사용자 가이드

AWS Schema Conversion Tool

버전 1.0.672

AWS Schema Conversion Tool: 사용자 가이드

Table of Contents

AWS SCT란 무엇인가요? 1
스키마 변환 개요 5
피드백 제공 6
설치, 확인 및 업데이트 8
설치 AWS SCT 8
AWS SCT 파일 다운로드 확인 9
파일의 체크섬 확인 AWS SCT 10
AWS SCT 페도라에서 RPM 파일 확인하기 11
AWS SCT 우분투에서 DEB 파일 확인하기 11
마이크로소프트 윈도우에서 AWS SCT MSI 파일 확인하기 12
필수 데이터베이스 드라이버 다운로드 12
Linux에 JDBC 드라이버 설치 15
전역 설정에 드라이버 경로 저장 16
업데이트 AWS SCT 17
AWS SCT CLI 18
AWS SCT 사용자 인터페이스 사용 19
프로젝트 창 19
AWS SCT 시작 21
프로젝트 생성 21
새 프로젝트 마법사 사용 22
프로젝트 저장 및 열기 25
서버 추가 25
오프라인 모드 사용 26
트리 필터 사용 28
..... 28
트리 필터의 파일 목록 가져오기 30
스키마 숨기기 31
데이터베이스 마이그레이션 평가 보고서 관리 32
스키마 변환 36
변환된 코드 적용 39
AWS 프로필 저장 40
AWS 보안 인증 정보 저장 40
프로젝트의 기본 프로필 설정 42
AWS 서비스 프로필 사용 권한 43

AWS Secrets Manager 사용 ... 44
데이터베이스 암호 저장 ... 45
파티션 테이블이 있는 프로젝트에 Union All 보기 사용 ... 45
키보드 바로 가기 ... 46
시작하기 ... 47
AWS SCT 소스 ... 49
Amazon RDS 연결 암호화 ... 50
Apache Cassandra를 소스로 사용 ... 53
Apache Cassandra에 소스로 연결 ... 53
Apache Hadoop을 소스로 사용 ... 54
Apache Hadoop을 소스로 사용하기 위한 사전 요구 사항 ... 55
Hive를 소스로 사용하기 위한 권한 ... 56
HDFS를 소스로 사용하기 위한 권한 ... 56
HDFS를 대상으로 사용하기 위한 권한 ... 57
Apache Hadoop에 소스로 연결 ... 57
Hive 및 HDFS에 연결 ... 59
Amazon EMR에 대상으로 연결 ... 62
Apache Oozie를 소스로 사용 ... 64
필수 조건 ... 65
Apache Oozie에 소스로 연결 ... 65
AWS Lambda 권한 ... 67
AWS Step Functions에 대상으로 연결 ... 69
Azure SQL Database를 소스로 사용 ... 70
Azure SQL Database를 사용하기 위한 권한 ... 70
Azure SQL Database에 소스로 연결 ... 71
IBM Db2 for z/OS를 소스로 사용 ... 72
Db2 for z/OS에 대한 사전 조건 ... 73
Db2 for z/OS에 대한 권한 ... 73
Db2 for z/OS에 소스로 연결 ... 74
MySQL을 대상으로 사용하기 위한 권한 ... 76
PostgreSQL을 대상으로 사용하기 위한 권한 ... 78
Db2 for z/OS에서 PostgreSQL로의 변환 설정 ... 78
IBM Db2 LUW를 소스로 사용 ... 80
Db2 LUW에 대한 권한 ... 80
Db2 LUW에 소스로 연결 ... 83
Db2 LUW를 PostgreSQL로 ... 85

Db2 LUW를 MySQL로 ... 87
MySQL을 소스로 사용 ... 88
MySQL에 대한 권한 ... 89
MySQL 소스에 연결 ... 89
PostgreSQL을 대상으로 사용하기 위한 권한 ... 92
Oracle Database를 소스로 사용 ... 92
Oracle에 대한 권한 ... 93
Oracle 소스에 연결 ... 93
Oracle에서 PostgreSQL ... 97
Oracle에서 MySQL로 ... 103
Oracle에서 Amazon RDS for Oracle로 ... 112
PostgreSQL을 소스로 사용 ... 119
PostgreSQL에 대한 권한 ... 119
PostgreSQL에 소스로 연결 ... 119
MySQL을 대상으로 사용하기 위한 권한 ... 121
SAP ASE(Sybase ASE)를 소스로 사용 ... 123
SAP ASE에 대한 권한 ... 123
SAP ASE 소스에 연결 ... 124
MySQL을 대상으로 사용하기 위한 권한 ... 126
SAP ASE에서 MySQL로 변환 설정 ... 128
PostgreSQL을 대상으로 사용하기 위한 권한 ... 128
SAP ASE에서 PostgreSQL로 변환 설정 ... 129
SQL Server를 소스로 사용 ... 130
Microsoft SQL Server에 대한 권한 ... 131
Microsoft SQL Server를 통한 Windows 인증 사용 ... 132
SQL Server에 소스 연결 ... 134
SQL Server에서 MySQL로 ... 136
SQL Server에서 PostgreSQL로 ... 141
SQL Server에서 Amazon RDS SQL Server로 ... 175
AWS SCT용 데이터 웨어하우스 소스 ... 177
Amazon Redshift를 소스로 사용 ... 177
Azure Synapse Analytics를 소스로 사용 ... 182
BigQuery를 소스로 사용 ... 187
Greenplum Database를 소스로 사용 ... 193
Netezza를 소스로 사용 ... 198
Oracle Data Warehouse를 소스로 사용 ... 207

Snowflake를 소스로 사용 ... 214
SQL Server Data Warehouse를 소스로 사용 ... 222
Teradata를 소스로 사용 ... 228
Vertica를 소스로 사용 ... 243
매핑 규칙 생성 ... 250
새 규칙 ... 250
규칙 관리 ... 251
가상 대상 ... 252
제한 사항 ... 253
변환 보고서 생성 ... 254
마이그레이션 평가 보고서 ... 254
데이터베이스 마이그레이션 평가 보고서 생성 ... 255
평가 보고서 보기 ... 256
평가 보고서 저장 ... 260
평가 보고서 구성 ... 262
다중 서버 평가 보고서 생성 ... 266
데이터베이스 스키마 변환 ... 275
마이그레이션 규칙 생성 ... 277
마이그레이션 규칙 생성 ... 278
마이그레이션 규칙 내보내기 ... 279
스키마 변환 ... 280
스키마 변환 ... 280
변환된 스키마 편집 ... 282
변환된 스키마 지우기 ... 283
수동 변환 처리 ... 284
소스 스키마 수정 ... 284
대상 스키마 수정 ... 284
변환된 스키마 업데이트 및 새로 고침 ... 285
스키마 저장 및 적용 ... 286
변환된 스키마 저장 ... 286
변환된 스키마 적용 ... 287
확장 팩 스키마 ... 287
스키마 비교 ... 288
관련 변환된 객체 ... 289
데이터 웨어하우스 스키마를 Amazon Redshift로 변환 ... 290
Amazon Redshift에 대한 권한 ... 291

최적화 전략 및 규칙 선택 ... 293
통계 수집 또는 업로드 ... 294
마이그레이션 규칙 생성 ... 295
마이그레이션 규칙 생성 ... 296
마이그레이션 규칙 내보내기 ... 297
스키마 변환 ... 298
스키마 변환 ... 298
변환된 스키마 편집 ... 300
변환된 스키마 지우기 ... 301
키 관리 및 사용자 지정 ... 302
관련 주제 ... 302
평가 보고서 생성 및 사용 ... 303
데이터베이스 마이그레이션 평가 보고서 생성 ... 303
요약 ... 304
작업 항목 ... 305
평가 보고서 저장 ... 305
수동 변환 처리 ... 306
소스 스키마 수정 ... 307
대상 스키마 수정 ... 307
변환된 스키마 업데이트 및 새로 고침 ... 307
변환된 스키마 저장 및 적용 ... 308
변환된 스키마를 파일에 저장 ... 308
변환된 스키마 적용 ... 309
확장 팩 스키마 ... 309
Python 라이브러리 ... 310
Amazon Redshift 최적화 ... 310
Amazon Redshift 데이터베이스 최적화 ... 310
ETL 프로세스 변환 ... 313
ETL 프로세스를 AWS Glue로 변환 ... 314
필수 조건 ... 315
AWS Glue 데이터 카탈로그 ... 316
제한 사항 ... 316
1단계: 새 프로젝트 생성 ... 318
2단계: AWS Glue 작업 생성 ... 319
AWS Glue용 Python API를 사용하여 ETL 프로세스 변환 ... 320
1단계: 데이터베이스 생성 ... 321

2단계: 연결 생성 321
3단계: AWS Glue 크롤러 생성 323
Informatica ETL 스크립트 변환 325
SSIS를 AWS Glue로 변환 330
지원되는 SSIS 구성 요소 333
SSIS를 AWS Glue Studio로 변환 335
필수 조건 336
AWS SCT 프로젝트에 SSIS 패키지 추가 338
SSIS 패키지 변환 339
AWS Glue Studio 작업 생성 339
SSIS 변환 평가 보고서 생성 341
지원되는 SSIS 구성 요소 342
Teradata BTEQ를 Amazon Redshift RSQL로 변환 343
AWS SCT 프로젝트에 BTEQ 스크립트 추가 344
BTEQ 스크립트에서 대체 변수 구성 345
BTEQ 스크립트 변환 345
BTEQ 스크립트 관리 346
BTEQ 스크립트 변환 평가 보고서 생성 347
변환된 BTEQ 스크립트 편집 및 저장 347
쉘 스크립트를 Amazon Redshift RSQL로 변환 348
AWS SCT 프로젝트에 쉘 스크립트 추가 349
쉘 스크립트에서 대체 변수 구성 350
쉘 스크립트 변환 350
쉘 스크립트 관리 351
쉘 스크립트 변환 평가 보고서 생성 352
변환된 쉘 스크립트 편집 및 저장 352
Teradata FastExport를 Amazon Redshift RSQL로 변환 353
AWS SCT 프로젝트에 FastExport 작업 스크립트 추가 353
FastExport 작업 스크립트에서 대체 변수 구성 354
FastExport 작업 스크립트 변환 356
FastExport 작업 스크립트 관리 356
FastExport 작업 스크립트 변환 평가 보고서 생성 357
변환된 FastExport 작업 스크립트 편집 및 저장 358
Teradata FastLoad를 Amazon Redshift RSQL로 변환 358
AWS SCT 프로젝트에 FastLoad 작업 스크립트 추가 359
FastLoad 작업 스크립트에서 대체 변수 구성 360

FastLoad 작업 스크립트 변환 361
FastLoad 작업 스크립트 관리 362
FastLoad 작업 스크립트 변환 평가 보고서 생성 363
변환된 FastLoad 작업 스크립트 편집 및 저장 363
Teradata MultiLoad를 Amazon Redshift RSQL로 변환 364
AWS SCT 프로젝트에 MultiLoad 작업 스크립트 추가 364
MultiLoad 작업 스크립트에서 대체 변수 구성 365
MultiLoad 작업 스크립트 변환 366
MultiLoad 작업 스크립트 관리 367
MultiLoad 작업 스크립트 변환 평가 보고서 생성 368
변환된 MultiLoad 작업 스크립트 편집 및 저장 369
빅 데이터 프레임워크 마이그레이션 370
Apache Hadoop을 Amazon EMR로 마이그레이션 370
개요 371
1단계: Hadoop 클러스터에 연결 372
2단계: 매핑 규칙 설정 372
3단계: 평가 보고서 생성 373
4단계: Apache Hadoop 클러스터를 Amazon EMR로 마이그레이션 375
CLI 스크립트 실행 376
마이그레이션 프로젝트 관리 376
Apache Oozie를 AWS Step Functions로 변환 378
개요 378
1단계: 소스 및 대상 서비스에 연결 380
2단계: 매핑 규칙 설정 380
3단계: 파라미터 구성 381
4단계: 평가 보고서 생성 382
5단계: Apache Oozie 워크플로를 AWS Step Functions로 변환 384
CLI 스크립트 실행 386
지원되는 노드 386
AWS SCT와 함께 AWS DMS 사용 388
AWS DMS에서 AWS SCT 복제 에이전트 사용 388
AWS DMS에서 AWS SCT 데이터 추출 에이전트 사용 388
AWS DMS와 함께 AWS SCT 사용 시 로깅 수준 높이기 388
데이터 웨어하우스에서 Amazon Redshift로 마이그레이션 390
필수 조건 392
Amazon S3 설정 393

IAM 역할 수임 ... 394
보안 설정 ... 395
구성 설정 ... 396
에이전트 설치 ... 396
에이전트 구성 ... 398
전용 복사 에이전트 설치 및 구성 ... 399
에이전트 시작 ... 401
에이전트 등록 ... 401
상담원의 정보 숨기기 및 복구하기 AWS SCT ... 402
데이터 마이그레이션 규칙 생성 ... 404
데이터 마이그레이션을 위한 추출기 및 복사 설정 변경 ... 405
데이터 정렬 ... 407
AWS SCT 작업 생성, 실행, 모니터링 ... 409
데이터 추출 작업 내보내기 및 가져오기 ... 412
AWS Snowball 에지 디바이스를 사용한 데이터 추출 ... 413
tep-by-step 및 Edge를 사용하여 AWS SCT 데이터를 마이그레이션하는 S 절차 AWS Snowball ... 414
데이터 추출 작업 출력 ... 417
가상 파티셔닝 사용 ... 418
가상 파티셔닝 생성 시 제한 ... 419
RANGE 파티션 유형 ... 419
LIST 파티션 유형 ... 420
DATE AUTO SPLIT 파티션 유형 ... 421
네이티브 파티셔닝 사용 ... 422
LOB 사용 ... 423
모범 사례 및 문제 해결 ... 424
애플리케이션 SQL 변환 ... 426
애플리케이션 SQL 변환 개요 ... 426
애플리케이션의 SQL 코드 변환 ... 427
일반 애플리케이션 변환 프로젝트 생성 ... 427
애플리케이션 변환 프로젝트 관리 ... 431
SQL 코드 분석 및 변환 ... 432
평가 보고서 생성 및 사용 ... 433
변환된 SQL 코드 편집 및 저장 ... 434
C# 애플리케이션의 SQL 코드 변환 ... 434
C# 애플리케이션 변환 프로젝트 생성 ... 434

C# 애플리케이션 SQL 코드 변환 436
변환된 애플리케이션 코드 저장 437
C# 애플리케이션 변환 프로젝트 관리 438
C# 애플리케이션 변환 평가 보고서 생성 438
C++ 애플리케이션의 SQL 코드 변환 440
C++ 애플리케이션 변환 프로젝트 생성 440
C++ 애플리케이션 SQL 코드 변환 441
변환된 애플리케이션 코드 저장 443
C++ 애플리케이션 변환 프로젝트 관리 443
C++ 애플리케이션 변환 평가 보고서 생성 445
Java 애플리케이션의 SQL 코드 변환 446
Java 애플리케이션 변환 프로젝트 생성 446
Java 애플리케이션 SQL 코드 변환 448
변환된 애플리케이션 코드 저장 449
Java 애플리케이션 변환 프로젝트 관리 449
Java 애플리케이션 변환 평가 보고서 생성 450
Pro*C 애플리케이션의 SQL 코드 변환 452
Pro*C 애플리케이션 변환 프로젝트 생성 452
Pro*C 애플리케이션 SQL 코드 변환 453
변환된 애플리케이션 코드 편집 및 저장 455
Pro*C 애플리케이션 변환 프로젝트 관리 455
Pro*C 애플리케이션 변환 평가 보고서 생성 456
확장 팩 사용 458
확장 팩 사용에 필요한 권한 459
확장 팩 스키마 사용 460
확장 팩용 사용자 지정 라이브러리 461
확장 팩 적용 461
AWS SCT 확장 팩의 Lambda 함수 사용 464
AWS Lambda 함수를 사용하여 데이터베이스 기능 에뮬레이션 464
확장 팩을 적용하여 Lambda 함수 지원 464
확장 팩 기능 구성 466
모범 사례 467
추가 메모리 구성 467
기본 프로젝트 폴더 467
데이터 마이그레이션 속도 향상 468
로깅 정보 증가 468

문제 해결 471
Oracle 소스 데이터베이스에서 객체를 로드할 수 없습니다. 471
경고 메시지 471
CLI 참조 473
필수 조건 473
대화형 모드 473
예제 475
CLI 시나리오 가져오기 475
예제 478
CLI 시나리오 편집 479
스크립트 모드 480
예제 481
참조 자료 481
릴리스 정보 482
릴리스 노트 — 676 482
릴리스 노트 — 675 487
릴리스 정보 – 674 489
릴리스 정보 – 673 496
릴리스 정보 – 672 501
릴리스 정보 – 671 509
릴리스 정보 – 670 518
릴리스 정보 – 669 523
릴리스 정보 – 668 527
릴리스 정보 – 667 534
릴리스 정보 – 666 538
릴리스 정보 – 665 543
릴리스 정보 – 664 546
릴리스 정보 – 663 549
릴리스 정보 – 662 552
릴리스 정보 – 661 557
릴리스 정보 – 660 561
릴리스 정보 – 659 564
릴리스 정보 – 658 569
릴리스 정보 – 657 574
릴리스 정보 – 656 578
릴리스 정보 – 655 581

릴리스 정보 – 654 584
릴리스 정보 – 653 587
릴리스 정보 – 652 589
릴리스 정보 – 651 592
릴리스 정보 – 650 594
릴리스 정보 – 649 596
릴리스 정보 – 648 599
릴리스 정보 – 647 600
릴리스 정보 – 646 602
릴리스 정보 – 645 603
릴리스 정보 – 644 605
릴리스 정보 – 642 607
릴리스 정보 – 641 608
릴리스 정보 – 640 609
릴리스 1.0.640 Oracle 변경 사항 610
릴리스 1.0.640 Microsoft SQL Server 변경 사항 615
릴리스 1.0.640 MySQL 변경 사항 619
릴리스 1.0.640 PostgreSQL 변경 사항 620
릴리스 1.0.640 Db2 LUW 변경 사항 624
릴리스 1.0.640 Teradata 변경 사항 624
기타 엔진과 관련된 릴리스 1.0.640 변경 사항 626
문서 기록 629
이전 업데이트 641
..... dcxlix

AWS Schema Conversion Tool이란 무엇인가요?

AWS Schema Conversion Tool (AWS SCT)를 사용하여 기존 데이터베이스 스키마를 한 데이터베이스 엔진에서 다른 데이터베이스 엔진으로 변환할 수 있습니다. 관계형 OLTP 스키마 또는 데이터 웨어하우스 스키마를 변환할 수 있습니다. 변환된 스키마는 Amazon Relational Database Service(RDS) MySQL, MariaDB, Oracle, SQL Server, PostgreSQL DB, Amazon Aurora DB 클러스터 또는 Amazon Redshift 클러스터에 적합합니다. 변환된 스키마는 Amazon EC2 인스턴스에서 데이터베이스와 함께 사용하거나 Amazon S3 버킷에 데이터로 저장할 수 있습니다.

AWS SCT는 Amazon S3 버킷 또는 다른 AWS 리소스에 연결할 때 Federal Information Processing Standards(FIPS)를 포함한 몇 가지 산업 표준을 지원합니다. 또한 AWS SCT는 Federal Risk and Authorization Management Program(FedRAMP)도 준수합니다. AWS 및 규정 준수 활동에 대한 자세한 내용은 규정 준수 프로그램 제공 AWS 범위 내 서비스 섹션을 참조하세요.

AWS SCT는 다음 OLTP 변환을 지원합니다.

원본 데이터베이스	대상 데이터베이스:
IBM Db2 for z/OS(버전 12)	Amazon Aurora MySQL-Compatible Edition(Aurora MySQL), Amazon Aurora PostgreSQL-Compatible Edition(Aurora PostgreSQL), MySQL, PostgreSQL 자세한 내용은 IBM Db2 for z/OS를 소스로 사용 섹션을 참조하세요.
IBM Db2 LUW(버전 9.1, 9.5, 9.7, 10.5, 11.1, 11.5)	Aurora MySQL, Aurora PostgreSQL, MariaDB, MySQL, PostgreSQL 자세한 내용은 IBM Db2 LUW를 소스로 사용 섹션을 참조하세요.
Microsoft Azure SQL Database	Aurora MySQL, Aurora PostgreSQL, MySQL, PostgreSQL 자세한 내용은 Azure SQL Database를 소스로 사용 섹션을 참조하세요.

원본 데이터베이스	대상 데이터베이스:
Microsoft SQL Server(버전 2008 R2, 2012, 2014, 2016, 2017, 2019, 2022)	Aurora MySQL, Aurora PostgreSQL, Babelfish for Aurora PostgreSQL(평가 보고서만 해당), MariaDB, Microsoft SQL Server, MySQL, PostgreSQL 자세한 내용은 SQL Server를 소스로 사용 섹션을 참조하세요.
MySQL(버전 5.5 이상)	Aurora PostgreSQL, MySQL, PostgreSQL 자세한 내용은 MySQL을 소스로 사용 섹션을 참조하세요. AWS SCT를 사용하지 않고 MySQL의 스키마 및 데이터를 Aurora MySQL DB 클러스터로 마이그레이션할 수 있습니다. 자세한 내용은 Amazon Aurora DB 클러스터로 데이터 마이그레이션을 참조하세요.
Oracle(버전 10.1 이상)	Aurora MySQL, Aurora PostgreSQL, MariaDB, MySQL, Oracle, PostgreSQL 자세한 내용은 Oracle Database를 소스로 사용 섹션을 참조하세요.
PostgreSQL(버전 9.1 이상)	Aurora MySQL, Aurora PostgreSQL, MySQL, PostgreSQL 자세한 내용은 PostgreSQL을 소스로 사용 섹션을 참조하세요.
SAP ASE(버전 12.5.4, 15.0.2, 15.5, 15.7, 16.0)	Aurora MySQL, Aurora PostgreSQL, MariaDB, MySQL, PostgreSQL 자세한 내용은 SAP ASE(Sybase ASE)를 소스로 사용 섹션을 참조하세요.

AWS SCT는 다음의 데이터 웨어하우스 변환을 지원합니다.

소스 데이터 웨어하우스	대상 데이터 웨어하우스
Amazon Redshift	Amazon Redshift 자세한 내용은 Amazon Redshift를 소스로 사용 섹션을 참조하세요.
Azure Synapse Analytics	Amazon Redshift 자세한 내용은 Azure Synapse Analytics를 소스로 사용 섹션을 참조하세요.
BigQuery	Amazon Redshift 자세한 내용은 BigQuery를 소스로 사용 섹션을 참조하세요.
Greenplum Database(버전 4.3 및 6.21)	Amazon Redshift 자세한 내용은 Greenplum Database를 소스로 사용 섹션을 참조하세요.
Microsoft SQL Server(버전 2008 이상)	Amazon Redshift 자세한 내용은 SQL Server Data Warehouse를 소스로 사용 섹션을 참조하세요.
Netezza(버전 7.0.3 이상)	Amazon Redshift 자세한 내용은 Netezza를 소스로 사용 섹션을 참조하세요.
Oracle(버전 10.1 이상)	Amazon Redshift 자세한 내용은 Oracle Data Warehouse를 소스로 사용 섹션을 참조하세요.
Snowflake(버전 3)	Amazon Redshift

소스 데이터 웨어하우스	대상 데이터 웨어하우스
	자세한 내용은 Snowflake를 소스로 사용 섹션을 참조하세요.
Teradata(버전 13 이상)	Amazon Redshift 자세한 내용은 Teradata를 소스로 사용 섹션을 참조하세요.
Vertica(버전 7.2.2 이상)	Amazon Redshift 자세한 내용은 Vertica를 소스로 사용 섹션을 참조하세요.

AWS SCT는 다음과 같은 데이터 NoSQL 데이터베이스 변환을 지원합니다.

원본 데이터베이스	대상 데이터베이스:
Apache Cassandra(버전 2.1.x, 2.2.16, 3.11.x)	Amazon DynamoDB 자세한 내용은 Apache Cassandra를 소스로 사용 섹션을 참조하세요.

AWS SCT는 다음과 같은 추출, 전환, 적재(ETL) 프로세스의 변환을 지원합니다. 자세한 내용은 ETL 프로세스 변환 섹션을 참조하세요.

소스	대상
Informatica ETL 스크립트	Informatica
Microsoft SQL Server Integration Services(SSIS) ETL 패키지	AWS Glue 또는 AWS Glue Studio
Teradata Basic Teradata Query(BTEQ)의 임베디드 명령이 포함된 쉘 스크립트	Amazon Redshift RSQL
Teradata BTEQ ETL 스크립트	AWS Glue 또는 Amazon Redshift RSQL

소스	대상
Terata FastExport 작업 스크립트	Amazon Redshift RSQL
Teradata FastLoad 작업 스크립트	Amazon Redshift RSQL
Teradata MultiLoad 작업 스크립트	Amazon Redshift RSQL

AWS SCT는 다음과 같은 빅 데이터 프레임워크 마이그레이션을 지원합니다. 자세한 내용은 빅 데이터 프레임워크 마이그레이션 섹션을 참조하세요.

소스	대상
Apache Hive(버전 0.13.0 이상)	Amazon EMR의 Hive
Apache HDFS	Amazon EMR의 Amazon S3 또는 HDFS
Apache Oozie	AWS Step Functions

스키마 변환 개요

AWS SCT는 소스 데이터베이스의 데이터베이스 스키마를 대상 Amazon RDS 인스턴스와 호환되는 형식으로 자동 변환할 수 있는 프로젝트 기반 사용자 인터페이스를 제공합니다. 소스 데이터베이스의 스키마를 자동으로 변환할 수 없는 경우 AWS SCT가 대상 Amazon RDS 데이터베이스에서 동일한 스키마를 생성할 수 있는 방법에 대한 지침을 제공합니다.

AWS SCT 설치 방법에 대한 자세한 내용은 설치, 확인 및 업데이트 AWS SCT 단원을 참조하십시오.

AWS SCT 사용자 인터페이스에 대한 소개는 AWS SCT 사용자 인터페이스 사용 단원을 참조하십시오.

변환 프로세스에 대한 자세한 정보는 AWS SCT를 사용하여 데이터베이스 스키마 변환 단원을 참조하십시오.

AWS SCT에는 기존 데이터베이스 스키마를 한 데이터베이스 엔진에서 다른 데이터베이스 엔진으로 변환하는 기능 외에도 다음과 같이 데이터 및 애플리케이션을 AWS 클라우드로 이전하는 데 도움이 되는 몇 가지 추가 기능이 있습니다.

- 데이터 추출 에이전트를 사용하면 데이터 웨어하우스로부터 데이터를 추출하여 Amazon Redshift로 마이그레이션할 준비를 할 수 있습니다. 데이터 추출 에이전트를 관리하려면 AWS SCT를 사용할 수 있습니다. 자세한 내용은 온프레미스 데이터 웨어하우스에서 Amazon Redshift로 데이터 마이그레이션 섹션을 참조하세요.
- AWS SCT를 사용하여 AWS DMS 엔드포인트 및 작업을 생성할 수 있습니다. AWS SCT로부터 이러한 작업을 실행하고 모니터링할 수 있습니다. 자세한 내용은 AWS SCT와 함께 AWS DMS 사용 섹션을 참조하세요.
- 데이터베이스 기능을 동등한 Amazon RDS 또는 Amazon Redshift 기능으로 변환할 수 없는 경우도 있습니다. AWS SCT 확장 팩 마법사를 통해 AWS Lambda 함수와 Python 라이브러리를 설치하고 변환되지 않는 기능을 에뮬레이트할 수 있습니다. 자세한 내용은 AWS SCT 확장 팩 사용 섹션을 참조하세요.
- AWS SCT를 사용하여 기존 Amazon Redshift 데이터베이스를 최적화할 수 있습니다. AWS SCT에서 사용자의 데이터베이스를 최적화하기 위한 정렬 키 및 배포 키를 추천합니다. 자세한 내용은 AWS SCT를 사용하여 Amazon Redshift 최적화 섹션을 참조하세요.
- AWS SCT를 사용하면 동일한 엔진을 실행하는 Amazon RDS DB 인스턴스로 기존 온프레미스 데이터베이스 스키마를 복사할 수 있습니다. 이 기능을 사용하면 클라우드로 이전하고 라이선스 유형을 변경하는 데 따르는 비용 절감의 가능성을 분석할 수 있습니다.
- AWS SCT를 사용하여 C++, C#, Java 또는 기타 애플리케이션 코드의 SQL을 변환할 수 있습니다. 변환된 SQL 코드를 보고, 분석하고, 편집하고, 저장할 수 있습니다. 자세한 내용은 AWS SCT를 사용하여 애플리케이션 SQL 변환 섹션을 참조하세요.
- AWS SCT를 사용하여 추출, 전환, 적재(ETL) 프로세스를 마이그레이션할 수 있습니다. 자세한 내용은 AWS Schema Conversion Tool을 사용하여 추출, 전환, 적재(ETL) 프로세스 변환 섹션을 참조하세요.

피드백 제공

AWS SCT에 대한 피드백을 제공할 수 있습니다. 버그 보고서를 제출하거나, 기능 요청을 제출하거나, 일반 정보를 제공할 수 있습니다.

AWS SCT에 대한 피드백을 제공하려면

1. AWS Schema Conversion Tool을 시작합니다.
2. 도움말 메뉴를 열고 Leave Feedback(피드백 남기기)을 선택합니다. Leave Feedback(피드백 남기기) 대화 상자가 열립니다.
3. Area(영역)에서 정보, Bug report(버그 보고서) 또는 Feature request(기능 요청)를 선택합니다.

4. Source database(소스 데이터베이스)에서 소스 데이터베이스를 선택합니다. 피드백이 특정 데이터베이스에 한정되지 않으면 Any(모두)를 선택합니다.
5. Target database(대상 데이터베이스)에서 대상 데이터베이스를 선택합니다. 피드백이 특정 데이터베이스에 한정되지 않으면 Any(모두)를 선택합니다.
6. 제목에 피드백 제목을 입력합니다.
7. 메시지에 피드백 내용을 입력합니다.
8. 전송을 선택하여 피드백을 제출합니다.

설치, 확인 및 업데이트 AWS SCT

AWS Schema Conversion Tool (AWS SCT) 는 프로젝트 기반 사용자 인터페이스를 제공하는 독립 실행형 응용 프로그램입니다. AWS SCT 마이크로소프트 윈도우, 페도라 리눅스, 우분투 리눅스에서 사용할 수 있습니다. AWS SCT 64비트 운영 체제에서만 지원됩니다.

AWS SCT 배포 파일의 올바른 버전을 받을 수 있도록 압축 파일을 다운로드한 후 확인 단계를 제공합니다. 제공된 단계를 사용하여 파일을 확인할 수 있습니다.

AWS SCT 독립 실행형 응용 프로그램과 명령줄 도구 모두로 사용할 수 있습니다. 명령줄 도구에 대한 자세한 내용은 을 참조하십시오. AWS SCT CLI

주제

- 설치 AWS SCT
- AWS SCT 파일 다운로드 확인
- 필수 데이터베이스 드라이버 다운로드
- 업데이트 AWS SCT
- AWS SCT CLI

설치 AWS SCT

다음 운영 AWS SCT 체제에 설치할 수 있습니다.

- Microsoft Windows 10
- Fedora Linux 36 이상
- Ubuntu Linux 18 이상

설치하려면 AWS SCT

1. 운영 체제의 링크를 사용하여 AWS SCT 설치 프로그램이 포함된 압축 파일을 다운로드합니다. 모든 압축 파일은 확장자가 .zip입니다. AWS SCT 설치 프로그램 파일의 압축을 풀면 운영 체제에 적합한 형식이 됩니다.

    - Microsoft Windows
    - Ubuntu Linux (.deb)
    - Fedora Linux (.rpm)

2. 다음과 같이 운영 체제의 AWS SCT 설치 프로그램 파일을 추출합니다.

운영 체제	파일 이름
Fedora Linux	`aws-schema-conversion-tool-1.0.` *`build-number`* `.x86_64.rpm`
Microsoft Windows	`AWS Schema Conversion Tool-1.0.` *`build-number`* `.msi`
Ubuntu Linux	`aws-schema-conversion-tool-1.0.` *`build-number`* `.deb`

3. 이전 단계에서 추출한 AWS SCT 설치 프로그램 파일을 실행합니다. 다음에 표시된 운영 체제별 지침을 사용하십시오.

운영 체제	설치 지침
Fedora Linux	파일을 다운로드한 폴더에서 다음 명령을 실행합니다. `sudo yum install aws-schema-conversion-tool-1.0.` *`build-number`* `.x86_64.rpm`
Microsoft Windows	파일을 두 번 클릭하여 설치 프로그램을 실행합니다.
Ubuntu Linux	파일을 다운로드한 폴더에서 다음 명령을 실행합니다. `sudo dpkg -i aws-schema-conversion-tool-1.0.` *`build-number`* `.deb`

4. 소스 및 대상 데이터베이스 엔진용 Java Database Connectivity(JDBC) 드라이버를 다운로드합니다. 지침 및 다운로드 링크는 필수 데이터베이스 드라이버 다운로드 단원을 참조하십시오.

이제 AWS SCT 애플리케이션 설정이 완료되었습니다. AWS SCT를 실행하려면 애플리케이션 아이콘을 두 번 클릭합니다.

AWS SCT 파일 다운로드 확인

배포 파일을 확인할 수 있는 몇 가지 방법이 있습니다. AWS SCT가장 간단한 방법은 파일의 체크섬을 AWS에서 공개한 체크섬과 비교하는 것입니다. 추가 보안 수준으로 다음 절차를 사용하여 파일이 설치되는 운영 체제에 따라 배포 파일을 확인할 수 있습니다.

이 단원에는 다음 주제가 포함되어 있습니다.

주제

- 파일의 체크섬 확인 AWS SCT
- AWS SCT 페도라에서 RPM 파일 확인하기
- AWS SCT 우분투에서 DEB 파일 확인하기
- 마이크로소프트 윈도우에서 AWS SCT MSI 파일 확인하기

파일의 체크섬 확인 AWS SCT

AWS SCT 압축 파일을 다운로드하거나 저장할 때 발생할 수 있는 오류를 감지하기 위해 파일 체크섬을 에서 제공한 값과 비교할 수 있습니다. AWS AWS 체크섬에는 SHA256 알고리즘을 사용합니다.

체크섬을 사용하여 AWS SCT 배포 파일을 확인하려면

1. 설치 섹션의 링크를 사용하여 AWS SCT 배포 파일을 다운로드합니다. 자세한 정보는 설치 AWS SCT을 참조하세요.
2. sha256Check.txt라는 최신 체크섬 파일을 다운로드합니다. 이 파일에는 최신 버전의 AWS SCT 체크섬이 포함되어 있습니다. 예를 들어 파일은 다음과 같을 수 있습니다.

   ```
   Fedora   b4f5f66f91bfcc1b312e2827e960691c269a9002cd1371cf1841593f88cbb5e6
   Ubuntu   4315eb666449d4fcd95932351f00399adb6c6cf64b9f30adda2eec903c54eca4
   Windows  6e29679a3c53c5396a06d8d50f308981e4ec34bd0acd608874470700a0ae9a23
   ```

3. 배포 파일이 포함된 디렉터리에서 운영 체제의 SHA256 확인 명령을 실행합니다. 예를 들어, Linux에서 다음 명령을 실행할 수 있습니다.

   ```
   shasum -a 256 aws-schema-conversion-tool-1.0.latest.zip
   ```

4. 명령의 결과를 sha256Check.txt 파일에 표시된 값과 비교합니다. 체크섬이 일치하면 배포 파일을 실행하는 것이 안전합니다. 체크섬이 일치하지 않으면 배포 파일을 실행하지 말고 AWS Support에 문의합니다.

AWS SCT 페도라에서 RPM 파일 확인하기

AWS 배포 파일 체크섬 외에도 다른 수준의 검증을 제공합니다. 배포 파일의 모든 RPM 파일은 AWS 개인 키로 서명됩니다. 퍼블릭 GPG 키는 amazon.com.public.gpg-key에서 볼 수 있습니다.

AWS SCT 페도라에서 RPM 파일을 확인하려면

1. 설치 섹션의 링크를 사용하여 AWS SCT 배포 파일을 다운로드하십시오.
2. AWS SCT 배포 파일의 체크섬을 확인합니다.
3. 배포 파일의 내용을 추출합니다. 확인할 RPM 파일의 위치를 찾습니다.
4. amazon.com.public.gpg-key 에서 퍼블릭 GPG 키를 다운로드합니다.
5. 다음 명령을 사용하여 퍼블릭 키를 RPM DB로 가져옵니다(적절한 권한이 있어야 합니다).

```
sudo rpm --import aws-dms-team@amazon.com.public.gpg-key
```

6. 다음 명령을 실행하여 가져오기가 성공했는지 확인합니다.

```
rpm -q --qf "%{NAME}-%{VERSION}-%{RELEASE} \n %{SUMMARY} \n" gpg-pubkey-
ea22abf4-5a21d30c
```

7. 다음 명령을 실행하여 RPM 서명을 확인합니다.

```
rpm --checksig -v aws-schema-conversion-tool-1.0.build number-1.x86_64.rpm
```

AWS SCT 우분투에서 DEB 파일 확인하기

AWS 배포 파일 체크섬 외에도 다른 수준의 검증을 제공합니다. 배포 파일의 모든 DEB 파일은 GPG 분리 서명으로 서명됩니다.

Ubuntu에서 AWS SCT DEB 파일을 확인하려면

1. 설치 섹션의 링크를 사용하여 AWS SCT 배포 파일을 다운로드하십시오.
2. AWS SCT 배포 파일의 체크섬 확인.
3. 배포 파일의 내용을 추출합니다. 확인할 DEB 파일의 위치를 찾습니다.
4. -1.0.latest.deb.asc에서 분리된 서명을 다운로드하십시오. aws-schema-conversion-tool
5. amazon.com.public.gpg-key 에서 퍼블릭 GPG 키를 다운로드합니다.
6. 다음 명령을 실행하여 GPG 퍼블릭 키를 가져옵니다.

```
gpg --import aws-dms-team@amazon.com.public.gpg-key
```

7. 다음 명령을 실행하여 서명을 확인합니다.

```
gpg --verify aws-schema-conversion-tool-1.0.latest.deb.asc aws-schema-conversion-
tool-1.0.build number.deb
```

마이크로소프트 윈도우에서 AWS SCT MSI 파일 확인하기

AWS 배포 파일 체크섬 외에도 다른 수준의 검증을 제공합니다. MSI 파일에는 서명했는지 확인할 수 있는 디지털 서명이 있습니다. AWS

Windows에서 AWS SCT MSI 파일을 확인하려면

1. 설치 섹션의 링크를 사용하여 AWS SCT 배포 파일을 다운로드합니다.
2. AWS SCT 배포 파일의 체크섬 확인.
3. 배포 파일의 내용을 추출합니다. 확인할 MSI 파일의 위치를 찾습니다.
4. Windows 탐색기에서 MSI 파일을 마우스 오른쪽 버튼으로 클릭하고 [속성]을 선택합니다.
5. 디지털 서명(Digital Signatures) 탭을 선택합니다.
6. 디지털 서명이 Amazon Services LLC의 서명인지 확인합니다.

필수 데이터베이스 드라이버 다운로드

제대로 AWS SCT 작동하려면 소스 및 대상 데이터베이스 엔진용 JDBC 드라이버를 다운로드하십시오. 가상 대상 데이터베이스 플랫폼을 사용하는 경우에는 대상 데이터베이스 엔진용 JDBC 드라이버를 다운로드할 필요가 없습니다. 자세한 정보는 가상 대상 사용을 참조하세요.

드라이버를 다운로드한 후 드라이버 파일의 위치를 제공합니다. 자세한 정보는 전역 설정에 드라이버 경로 저장을 참조하세요.

다음 위치에서 데이터베이스 드라이버를 다운로드할 수 있습니다.

> **Important**
>
> 사용 가능한 최신 버전의 드라이버를 다운로드합니다. 다음 표에는 에서 지원하는 데이터베이스 드라이버의 최저 버전이 나와 있습니다. AWS SCT

데이터베이스 엔진	드라이버	다운로드 위치
Amazon Aurora MySQL 호환 버전	`mysql-connector-java-5.1.6.jar`	https://www.mysql.com/products/connector/
Amazon Aurora PostgreSQL 호환 에디션	`postgresql-42.2.19.jar`	https://jdbc.postgresql.org/download/postgresql-42.2.19.jar
Amazon EMR	`HiveJDBC42.jar`	http://awssupportdatasvcs.com/bootstrap-actions/Simba/latest/
Amazon Redshift	`redshift-jdbc42-2.1.0.9.jar`	https://s3.amazonaws.com/redshift-downloads/drivers/jdbc/2.1.0.9/redshift-jdbc42-2.1.0.9.zip
Amazon Redshift Serverless	`redshift-jdbc42-2.1.0.9.jar`	https://s3.amazonaws.com/redshift-downloads/drivers/jdbc/2.1.0.9/redshift-jdbc42-2.1.0.9.zip
Apache Hive	`hive-jdbc-2.3.4-standalone.jar`	https://repo1.maven.org/maven2/org/apache/hive/hive-jdbc/2.3.4/hive-jdbc-2.3.4-standalone.jar
Azure SQL Database	`mssql-jdbc-7.2.2.jre11.jar`	https://docs.microsoft.com/en-us/sql/connect/jdbc/ release-notes-for-the -jdbc-드라이버? sql-server-ver뷰= 15 #72
Azure Synapse Analytics	`mssql-jdbc-7.2.2.jre11.jar`	https://docs.microsoft.com/en-us/sql/connect/jdbc/ release-notes-for-the -jdbc-driver? sql-server-ver뷰= 15 #72
Greenplum Database	`postgresql-42.2.19.jar`	https://jdbc.postgresql.org/download/postgresql-42.2.19.jar

데이터베이스 엔진	드라이버	다운로드 위치
IBM Db2 for z/OS	`db2jcc-db2jcc4.jar`	https://www.ibm.com/support/pages/db2 - - downloads-db2-zos jdbc-driver-versions-and
IBM Db2 LUW	`db2jcc-db2jcc4.jar`	https://www.ibm.com/support/pages/node/382667
MariaDB	`mariadb-java-client-2.4.1.jar`	https://downloads.mariadb.com/Connectors/java/connector-java-2.4.1/ mariadb-java-client-2.4.1.jar
Microsoft SQL Server	`mssql-jdbc-10.2.jar`	download-microsoft-jdbc-driverhttps://docs.microsoft.com/en-us/sql/connect/jdbc/ -? for-sql-server 조회= 15 sql-server-ver
MySQL	`mysql-connector-java-8.0.15.jar`	https://dev.mysql.com/downloads/connector/j/
Netezza	`nzjdbc.jar` 클라이언트 도구 소프트웨어를 사용합니다. 데이터 웨어하우스 버전 7.2.0과 하위 버전 호환되는 드라이버 버전 7.2.1을 다운로드합니다.	http://www.ibm.com/support/knowledgecenter/SSULQD_7.2.1/com.ibm.nz.datacon.doc/c_datacon_plg_overview.html
Oracle	`ojdbc8.jar` 드라이버 버전 8 이상이 지원됩니다.	https://www.oracle.com/database/technologies/jdbc-ucp-122-downloads.html
PostgreSQL	`postgresql-42.2.19.jar`	https://jdbc.postgresql.org/download/postgresql-42.2.19.jar
SAP ASE(Sybase ASE)	`jconn4.jar`	J커넥트 JDBC 드라이버

데이터베이스 엔진	드라이버	다운로드 위치
Snowflake	`snowflake-jdbc-3.9.2.jar` 자세한 내용은 JDBC 드라이버 다운로드/통합을 참조하세요.	https://repo1.maven.org/maven2/net/snowflake/snowflake-jdbc/3.9.2/snowflake-jdbc-3.9.2.jar

Teradata	`terajdbc4.jar`
	
	`tdgssconfig.jar`
	
	Teradata JDBC 드라이버 버전 16.20.00.11 이상의 경우 `tdgssconfig.jar` 파일이 필요하지 않습니다.
		https://downloads.teradata.com/download/connectivity/jdbc-driver

Vertica	`vertica-jdbc-9.1.1-0.jar` 드라이버 버전 7.2.0 이상이 지원됩니다.	https://www.vertica.com/client_drivers/9.1.x/9.1.1-0/vertica-jdbc-9.1.1-0.jar

Linux에 JDBC 드라이버 설치

다음 단계에 따라 Linux 시스템에 함께 사용할 JDBC 드라이버를 설치할 수 있습니다. AWS SCT

Linux 시스템에 JDBC 드라이버를 설치하려면

1. JDBC 드라이버를 저장할 디렉터리를 생성합니다.

```
PROMPT>sudo mkdir -p /usr/local/jdbc-drivers
```

2. 다음 명령을 사용하여 사용자의 데이터베이스 엔진용 JDBC 드라이버를 설치합니다.

데이터베이스 엔진	설치 명령
Amazon Aurora(MySQL 호환)	`PROMPT> cd /usr/local/jdbc-drivers` `PROMPT> sudo tar xzvf /tmp/mysql-connector-java-X.X.X.tar.gz`
Amazon Aurora(PostgreSQL 호환)	`PROMPT> cd /usr/local/jdbc-drivers` `PROMPT> sudo cp -a /tmp/postgresql-X.X.X.jre7.tar .`
Microsoft SQL Server	`PROMPT> cd /usr/local/jdbc-drivers` `PROMPT> sudo tar xzvf /tmp/sqljdbc_X.X.X_enu.tar.gz`
MySQL	`PROMPT> cd /usr/local/jdbc-drivers` `PROMPT> sudo tar xzvf /tmp/mysql-connector-java-X.X.X.tar.gz`

Oracle	`PROMPT> cd /usr/local/jdbc-drivers`
	`PROMPT> sudo mkdir oracle-jdbc`
	`PROMPT> cd oracle-jdbc`
	`PROMPT> sudo cp  -a /tmp/ojdbc8.jar .`

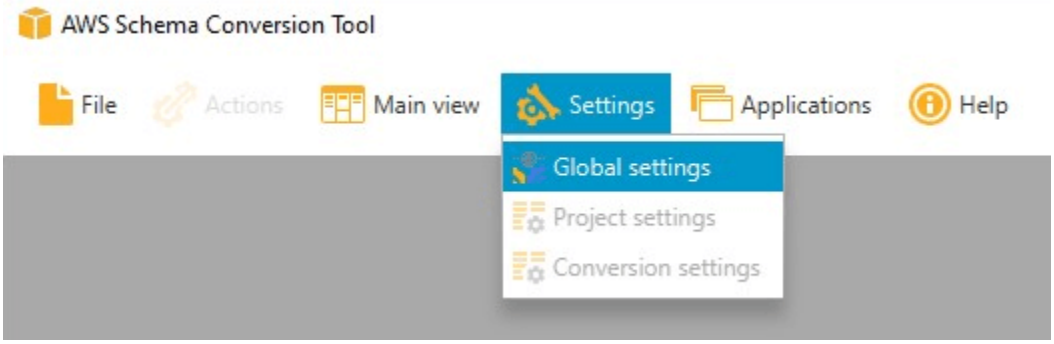
PostgreSQL	`PROMPT> cd /usr/local/jdbc-drivers` `PROMPT> sudo cp -a /tmp/postgresql-X.X.X.jre7.tar .`

전역 설정에 드라이버 경로 저장

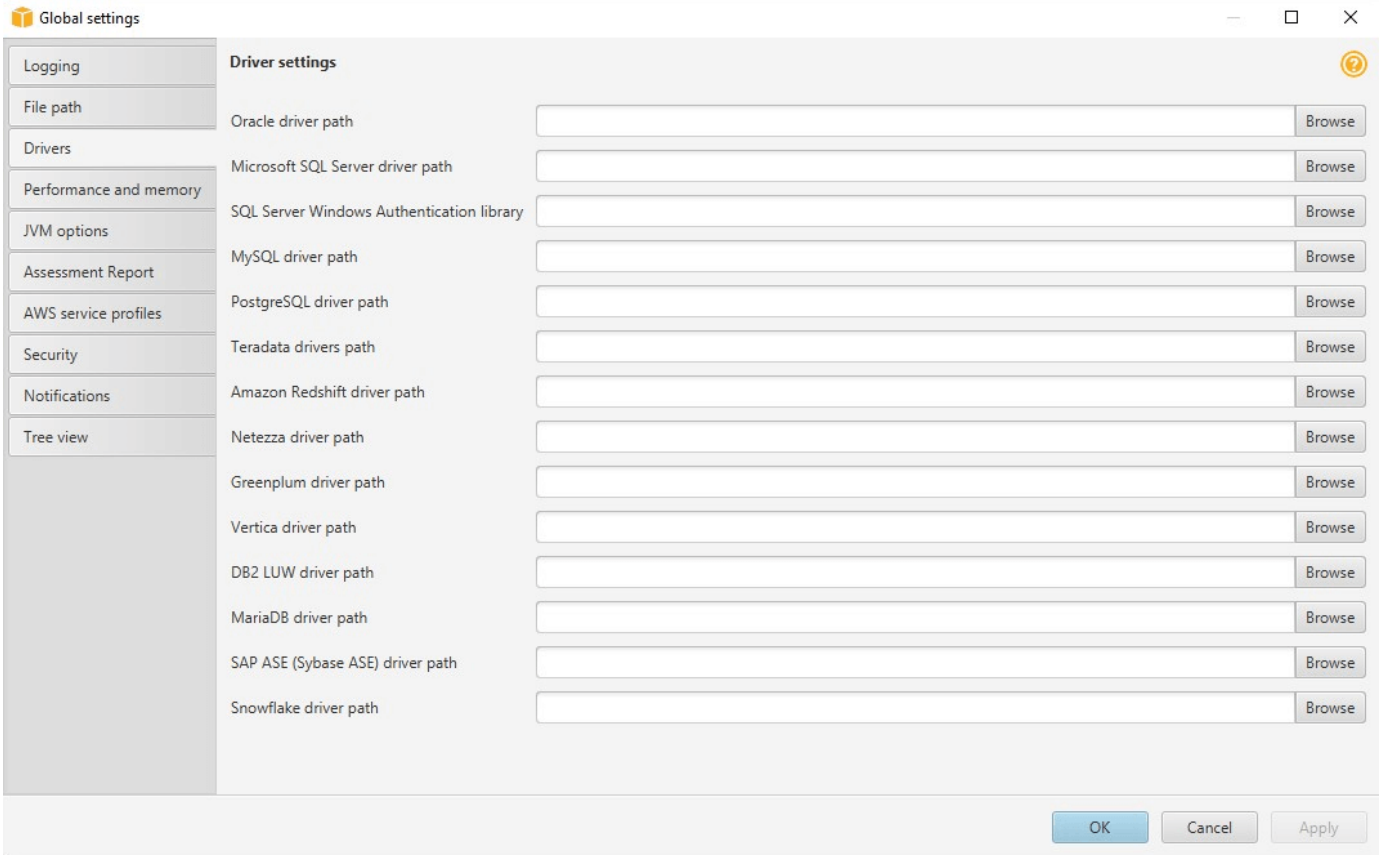
필요한 JDBC 드라이버를 다운로드하여 설치한 후 설정에서 드라이버의 위치를 전체적으로 설정할 수 있습니다. AWS SCT 드라이버 위치를 전역적으로 설정하지 않을 경우 데이터베이스에 연결할 때 애플리케이션이 드라이버 위치를 묻습니다.

드라이버 파일 위치를 업데이트하려면

1. 에서 AWS SCT설정을 선택한 다음 글로벌 설정을 선택합니다.



2. [Global settings]에서 [Drivers]를 선택합니다. 소스 데이터베이스 엔진 및 대상 Amazon RDS DB 인스턴스 데이터베이스 엔진용 JDBC 드라이버의 파일 경로를 추가합니다.



3. 드라이버 경로를 추가했으면 [OK]를 선택합니다.

업데이트 AWS SCT

AWS 새로운 AWS SCT 특징과 기능을 정기적으로 업데이트합니다. 이전 버전에서 업데이트하는 경우 새 AWS SCT 프로젝트를 만들고 사용 중인 데이터베이스 개체를 다시 변환하십시오.

에 대한 AWS SCT업데이트가 있는지 확인할 수 있습니다.

에 대한 업데이트를 확인하려면 AWS SCT

1. AWS SCT로그인하면 [도움말] 을 선택한 다음 [업데이트 확인] 을 선택합니다.
2. [Check for Updates] 대화 상자에서 [What's New]를 선택합니다. 링크가 나타나지 않으면 최신 버전을 가지고 있는 것입니다.

AWS SCT CLI

AWS SCT CLI를 다운로드하여 명령줄에서 사용할 수 있습니다. JAR을 다운로드하려면 다음 링크를 사용하십시오.

AWSSchemaConversionToolBatch.jar

AWS SCT 사용자 인터페이스 사용

다음 항목을 사용하여 AWS SCT 사용자 인터페이스로 작업할 수 있습니다. AWS SCT 설치에 대한 자세한 내용은 설치, 확인 및 업데이트 AWS SCT 섹션을 참조하세요.

주제

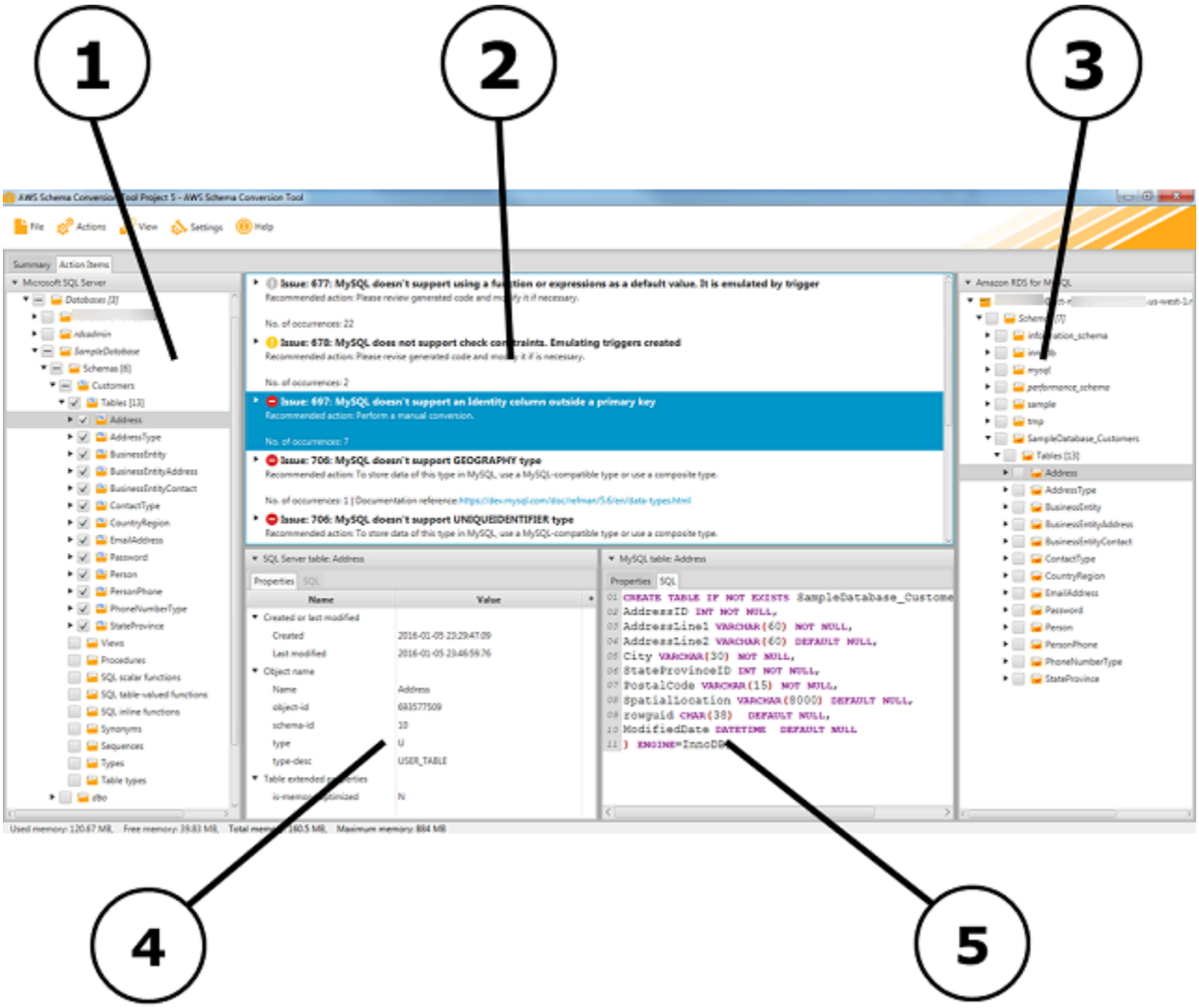
- AWS SCT 프로젝트 창
- AWS SCT 시작
- AWS SCT 프로젝트 생성
- AWS SCT에서 새 프로젝트 마법사 사용
- AWS SCT 프로젝트 저장 및 열기
- AWS SCT 프로젝트에 데이터베이스 서버 추가
- 오프라인 모드에서 AWS SCT 실행
- AWS SCT 트리 필터 사용
- AWS SCT 트리 보기에서 스키마 숨기기
- 데이터베이스 마이그레이션 평가 보고서 생성 및 검토
- 스키마 변환
- 변환된 스키마를 대상 DB 인스턴스에 적용
- AWS SCT에 AWS 서비스 프로필 저장
- AWS Secrets Manager 사용
- 데이터베이스 암호 저장
- 파티션 테이블이 있는 프로젝트에 UNION ALL 보기 사용
- AWS SCT의 키보드 바로 가기

AWS SCT 프로젝트 창

다음 그림은 스키마 마이그레이션 프로젝트를 만든 다음 스키마를 변환할 때 AWS SCT에서 볼 수 있습니다.

1. 왼쪽 패널에는 소스 데이터베이스의 스키마가 트리 보기로 표시됩니다. 데이터베이스 스키마는 “지연 로드”됩니다. 즉, 트리 보기에서 항목을 선택하면 AWS SCT는 소스 데이터베이스에서 현재 스키마를 가져와 표시합니다.

2. 상단 가운데 패널에는 대상 데이터베이스 엔진으로 자동 변환할 수 없는 소스 데이터베이스 엔진의 스키마 요소에 대한 작업 항목이 표시됩니다.
3. 오른쪽 패널에는 대상 DB 인스턴스의 스키마가 트리 보기로 표시됩니다. 데이터베이스 스키마는 “지연 로드”됩니다. 즉, 트리 보기에서 항목을 선택한 시점에서 AWS SCT가 대상 데이터베이스에서 현재 스키마를 가져와 표시합니다.



4. 왼쪽 하단 패널에서 스키마 요소를 선택하면 속성이 표시됩니다. 여기에는 소스 스키마 요소 및 해당 요소를 소스 데이터베이스에서 생성하는 SQL 명령이 설명되어 있습니다.
5. 오른쪽 하단 패널에서 스키마 요소를 선택하면 속성이 표시됩니다. 여기에는 대상 스키마 요소 및 해당 요소를 대상 데이터베이스에서 생성하는 SQL 명령이 설명되어 있습니다. 이 SQL 명령을 편집하고 업데이트된 명령을 프로젝트와 함께 저장할 수 있습니다.

AWS SCT 시작

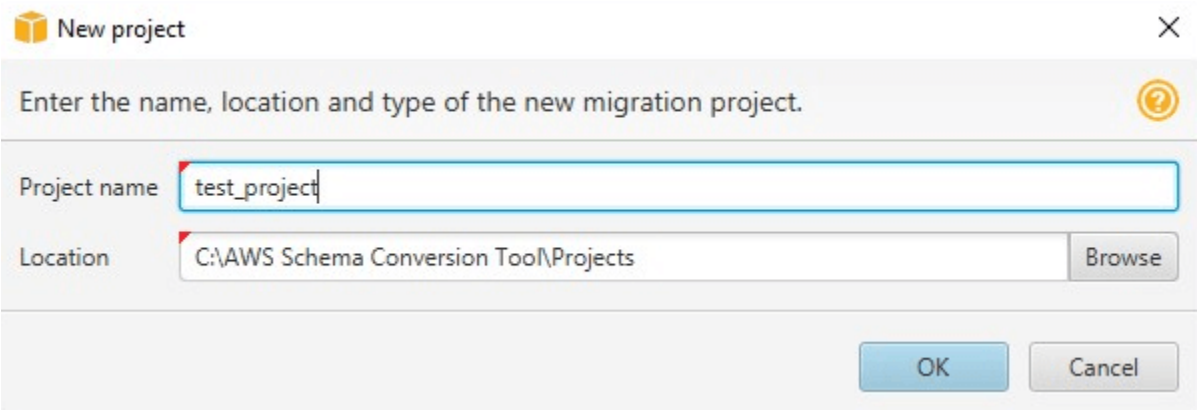
AWS Schema Conversion Tool을 시작하려면 애플리케이션 아이콘을 두 번 클릭합니다.

AWS SCT 프로젝트 생성

다음 절차에 따라 AWS Schema Conversion Tool 프로젝트를 생성합니다.

프로젝트를 생성하려면

1. AWS Schema Conversion Tool을 시작합니다.
2. 파일 메뉴에서 새 프로젝트를 선택합니다. 새 프로젝트 대화 상자가 나타납니다.



3. 컴퓨터에서 로컬로 저장되는 프로젝트의 이름을 입력합니다.
4. 로컬 프로젝트 파일의 위치를 입력합니다.
5. OK를 선택하여 AWS SCT 프로젝트를 생성합니다.
6. 소스 추가를 선택하여 AWS SCT 프로젝트에 새 소스 데이터베이스를 추가합니다. AWS SCT 프로젝트에 여러 소스 데이터베이스를 추가할 수 있습니다.
7. 대상 추가를 선택하여 AWS SCT 프로젝트에 새 대상 플랫폼을 추가합니다. AWS SCT 프로젝트에 여러 대상 플랫폼을 추가할 수 있습니다.
8. 왼쪽 패널에서 소스 데이터베이스 스키마를 선택합니다.
9. 오른쪽 패널에서 선택한 소스 스키마의 대상 데이터베이스 플랫폼을 지정합니다.
10. 매핑 생성을 선택합니다. 이 버튼은 소스 데이터베이스 스키마와 대상 데이터베이스 플랫폼을 선택한 후에 활성화됩니다. 자세한 내용은 매핑 규칙 생성 섹션을 참조하세요.

이제 AWS SCT 프로젝트가 설정되었습니다. 프로젝트를 저장하고, 데이터베이스 마이그레이션 평가 보고서를 생성하고, 소스 데이터베이스 스키마를 변환할 수 있습니다.

AWS SCT에서 새 프로젝트 마법사 사용

새 프로젝트 마법사를 사용하여 새 데이터베이스 마이그레이션 프로젝트를 생성할 수 있습니다. 이 마법사는 마이그레이션 대상을 결정하고 데이터베이스에 연결하는 데 도움을 줍니다. 또한 지원되는 모든 대상에 대해 마이그레이션이 얼마나 복잡할지 추정합니다. 마법사를 실행한 후 AWS SCT는 데이터베이스를 다른 대상으로 마이그레이션하기 위한 요약 보고서를 생성합니다. 이 보고서를 사용하여 가능한 대상을 비교하고 최적의 마이그레이션 경로를 선택할 수 있습니다.

새 프로젝트 마법사를 실행하려면

1. 소스 데이터베이스를 선택합니다.

   a. AWS Schema Conversion Tool을 시작합니다.

   b. 파일 메뉴에서 New project wizard를 선택합니다. Create a new database migration project 대화 상자가 열립니다.

   c. 소스 데이터베이스 연결 정보를 입력하려면 다음 지침을 사용합니다.

파라미터	작업
프로젝트 이름	컴퓨터에서 로컬로 저장되는 프로젝트의 이름을 입력합니다.
위치	로컬 프로젝트 파일의 위치를 입력합니다.
소스 유형	SQL database, NoSQL database 또는 ETL 옵션 중 하나를 선택합니다. 모든 마이그레이션 대상이 포함된 요약 보고서를 보려면 SQL database를 선택합니다.
[Source engine]	소스 데이터베이스 엔진을 선택합니다.
마이그레이션 전략	다음 옵션 중 하나를 선택합니다. • I want to switch engines and optimize for the cloud - 이 옵션은 소스 데이터베이스를 새 데이터베이스 엔진으로 변환합니다. • I want to keep the same engine but optimize for the cloud – 이 옵션은 데이터베이스 엔진을 그대로 유지하고 데이터베이스를 온프레미스에서 클라우드로 이동합니다.

파라미터	작업
	• I want to see a combined report for database engine switch and optimization for the cloud - 이 옵션은 사용 가능한 모든 마이그레이션 옵션의 마이그레이션 복잡성을 비교합니다. 모든 마이그레이션 대상이 포함된 집계된 평가 보고서를 보려면 마지막 옵션을 선택합니다.

d. 다음(Next)을 선택합니다. Connect to the source database 페이지가 열립니다.

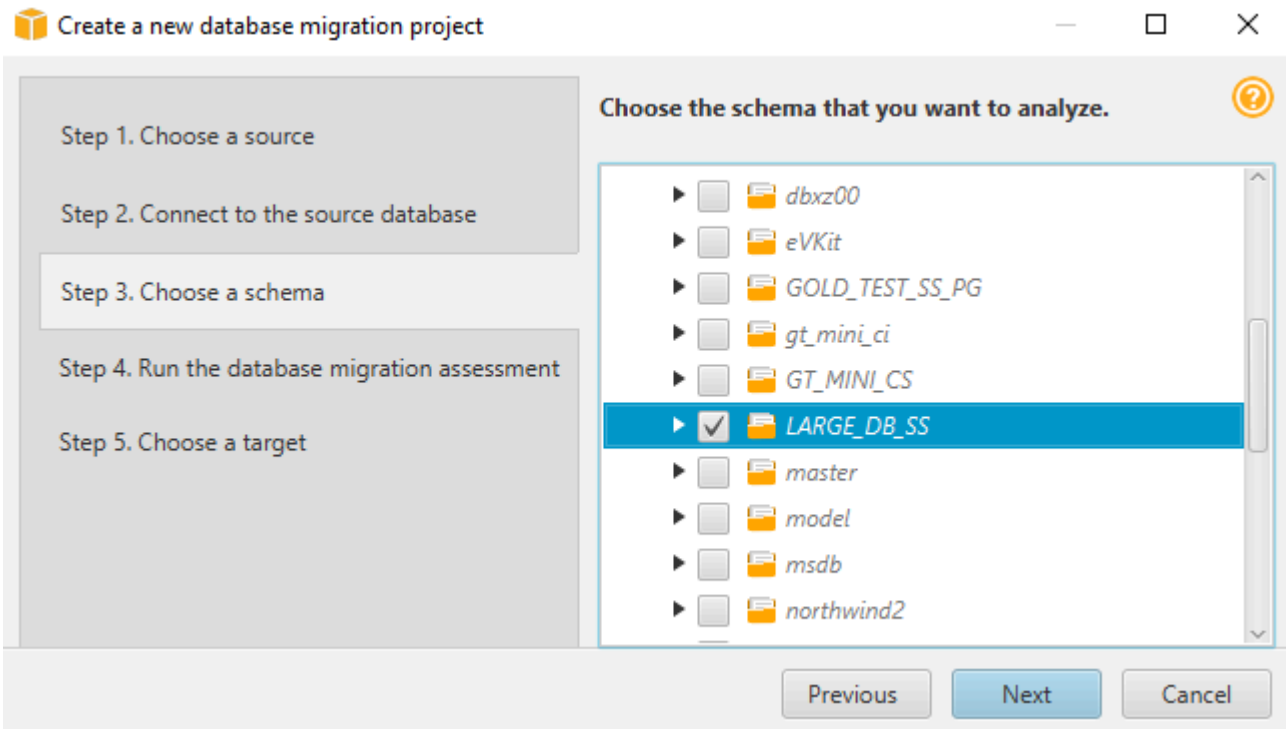
2. 소스 데이터베이스에 연결합니다.

a. 소스 데이터베이스 연결 정보를 제공합니다. 연결 파라미터는 소스 데이터베이스 엔진에 따라 달라집니다. 소스 데이터베이스 분석에 사용하는 사용자에게 해당 권한이 있어야 합니다. 자세한 내용은 AWS SCT 소스 섹션을 참조하세요.

b. 다음(Next)을 선택합니다. 스키마 선택 페이지가 열립니다.

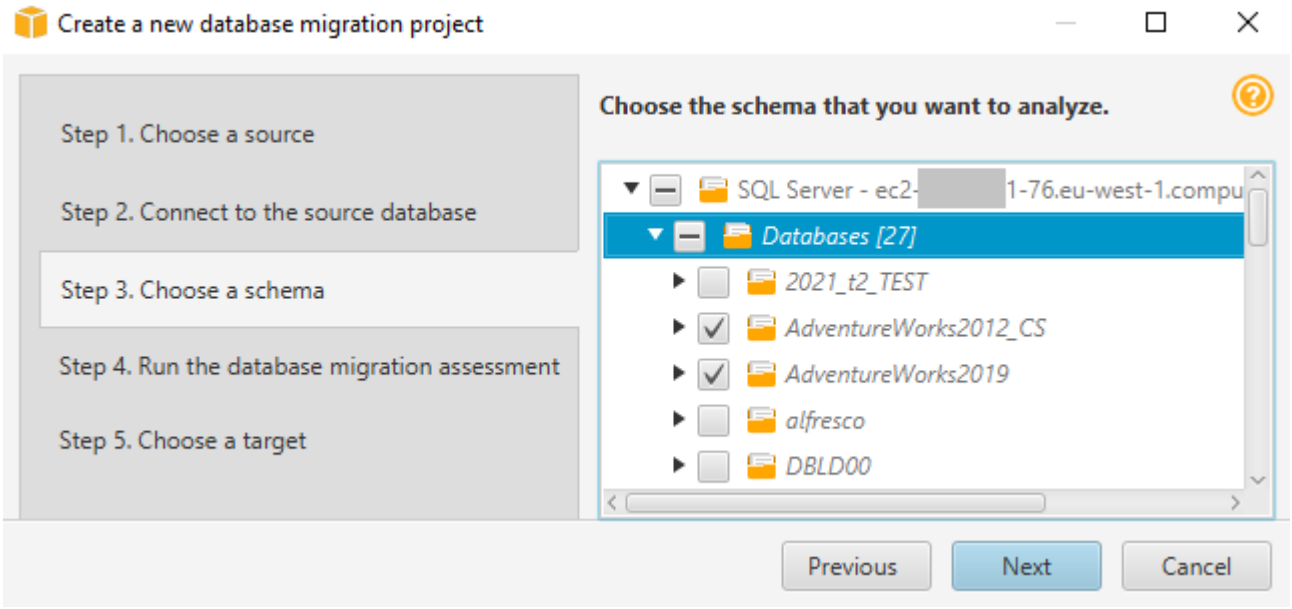
3. 데이터베이스 스키마를 선택합니다.

a. 평가할 스키마 이름에 대한 확인란을 선택하고 스키마 자체를 선택합니다. 스키마 이름을 선택하면 해당 스키마 이름이 파란색으로 강조 표시되고 다음 버튼을 사용할 수 있습니다.



b. 여러 데이터베이스 스키마를 평가하려면 모든 스키마에 대해 확인란을 선택한 다음 상위 노드를 선택합니다. 성공적인 평가를 위해서는 상위 노드를 선택해야 합니다. 예를 들어 소스 SQL

Server 데이터베이스의 경우 데이터베이스 노드를 선택합니다. 상위 노드의 이름은 파란색으로 강조 표시되고 다음 버튼을 사용할 수 있습니다.



c. 다음을 선택합니다. AWS SCT가 소스 데이터베이스 스키마를 분석하고 데이터베이스 마이그레이션 평가 보고서를 생성합니다. 소스 데이터베이스 스키마의 데이터베이스 객체 수는 평가를 실행하는 데 걸리는 시간에 영향을 줍니다. 완료되면 Run the database migration assessment 페이지가 열립니다.

4. 데이터베이스 마이그레이션 평가를 실행합니다.

   a. 다양한 마이그레이션 대상에 대한 평가 보고서를 검토 및 비교하거나 추가 분석을 위해 평가 보고서 파일의 로컬 사본을 저장할 수 있습니다.

   b. 데이터베이스 마이그레이션 평가 보고서의 로컬 사본을 저장합니다. 저장을 선택하고 파일을 저장할 폴더의 경로를 입력한 다음 저장을 선택합니다. AWS SCT가 평가 보고서 파일을 지정된 폴더에 저장합니다.

   c. 다음(Next)을 선택합니다. 대상 선택 페이지가 열립니다.

5. 대상 데이터베이스를 선택합니다.

   a. 대상 엔진에서 평가 보고서의 기준으로 사용하기로 결정한 대상 데이터베이스 엔진을 선택합니다.

   b. 대상 데이터베이스에 대한 연결 정보를 제공합니다. 표시되는 연결 파라미터는 선택한 대상 데이터베이스 엔진에 따라 달라집니다. 대상 데이터베이스에 지정된 사용자에게 필요한 권한이 있는지 확인합니다. 필요한 권한에 대한 자세한 내용은 AWS SCT 소스 및 Amazon Redshift를 대상으로 사용할 수 있는 권한에서 대상 데이터베이스의 권한을 설명하는 섹션을 참조하세요.

c. 완료를 선택합니다. AWS SCT가 프로젝트를 생성하고 매핑 규칙을 추가합니다. 자세한 내용은 매핑 규칙 생성 섹션을 참조하세요.

이제 AWS SCT 프로젝트를 사용하여 소스 데이터베이스 객체를 변환할 수 있습니다.

AWS SCT 프로젝트 저장 및 열기

다음 절차에 따라 AWS Schema Conversion Tool 프로젝트를 저장합니다.

프로젝트를 저장하려면

1. AWS Schema Conversion Tool을 시작합니다.
2. 파일 메뉴에서 Save project를 선택합니다.

   AWS SCT는 프로젝트를 생성할 때 지정된 폴더에 프로젝트를 저장합니다.

기존 AWS Schema Conversion Tool 프로젝트를 열려면 다음 절차를 따릅니다.

프로젝트를 열려면

1. 파일 메뉴에서 프로젝트 열기를 선택합니다. 열기 대화 상자가 나타납니다.
2. 프로젝트 폴더를 선택하고 Windows 스크립트 구성 요소(*.sct) 파일을 선택합니다.
3. AWS SCT가 프로젝트를 열지만 소스 및 대상 데이터베이스에 자동으로 연결되지는 않습니다. 데이터베이스 스키마 트리 상단에 있는 Connect to the server를 선택하여 소스 및 대상 데이터베이스에 연결합니다.

AWS SCT 버전 1.0.655 이전 버전으로 저장된 프로젝트를 열면 AWS SCT가 모든 소스 데이터베이스 스키마에 대한 매핑 규칙을 대상 데이터베이스 플랫폼에 자동으로 생성합니다. 다른 대상 데이터베이스 플랫폼을 추가하려면 기존 매핑 규칙을 삭제한 다음 새 매핑 규칙을 생성합니다. 매핑 규칙 생성에 대한 자세한 내용은 매핑 규칙 생성 섹션을 참조하세요.

AWS SCT 프로젝트에 데이터베이스 서버 추가

AWS Schema Conversion Tool 프로젝트에 여러 소스 및 대상 데이터베이스 서버를 추가할 수 있습니다.

프로젝트에 서버를 추가하려면

1. AWS Schema Conversion Tool을 시작합니다.
2. 새 프로젝트를 생성하거나 기존 프로젝트를 엽니다.
3. 메뉴에서 소스 추가를 선택하여 새 소스 데이터베이스를 추가합니다.
4. 데이터베이스 플랫폼을 선택하고 데이터베이스 연결 보안 인증 정보를 지정합니다. 소스 데이터베이스에 연결하는 방법에 대한 자세한 내용은 AWS SCT 소스 섹션을 참조하세요.

다음 절차에 따라 데이터베이스에 연결합니다.

데이터베이스에 연결하려면

1. 데이터베이스 서버에 대한 컨텍스트(마우스 오른쪽 버튼 클릭) 메뉴를 열고 연결 설정을 선택합니다.

   데이터베이스 스키마 트리 상단에서 Connect to the server를 선택할 수도 있습니다.
2. 소스 데이터베이스 서버에 연결하는 데 필요한 암호를 입력합니다.
3. Test Connection을 선택하여 AWS SCT가 소스 데이터베이스에 연결할 수 있는지 확인합니다.
4. 연결을 선택하여 소스 데이터베이스에 연결합니다.

AWS SCT 프로젝트에서 데이터베이스 서버를 제거하려면 다음 절차를 따릅니다.

데이터베이스 서버를 제거하려면

1. 제거할 데이터베이스 서버를 선택합니다.
2. 컨텍스트(마우스 오른쪽 버튼 클릭) 메뉴를 열고 프로젝트에서 제거를 선택합니다.

   AWS SCT는 선택한 데이터베이스 서버, 모든 매핑 규칙, 변환 결과 및 이 서버와 관련된 기타 메타데이터를 제거합니다.

오프라인 모드에서 AWS SCT 실행

오프라인 모드에서 AWS Schema Conversion Tool을 실행할 수 있습니다. 아래에서는 소스 데이터베이스에서 연결이 끊어진 경우 기존 AWS SCT 프로젝트를 사용하는 방법을 알아볼 수 있습니다.

AWS SCT에서는 다음 작업을 실행하기 위해 소스 데이터베이스에 연결할 필요가 없습니다.

- 매핑 규칙 추가
- 데이터베이스 마이그레이션 평가 보고서 생성
- 데이터베이스 스키마와 코드 변환
- 소스 및 변환된 코드 편집
- 소스 및 변환된 코드를 텍스트 파일에 SQL 스크립트로 저장

오프라인 모드에서 AWS SCT를 사용하려면 먼저 소스 데이터베이스에 연결하고 메타데이터를 로드한 다음 프로젝트를 저장합니다. 오프라인 모드에서 AWS SCT를 사용하려면 이 프로젝트를 열거나 소스 데이터베이스 서버와의 연결을 끊습니다.

오프라인 모드에서 AWS SCT를 실행하려면

1. AWS Schema Conversion Tool을 시작하고 새 프로젝트를 생성합니다. 자세한 내용은 AWS SCT 프로젝트 생성 섹션을 참조하세요.
2. 소스 데이터베이스 서버를 추가한 후 소스 데이터베이스에 연결합니다. 자세한 내용은 AWS SCT 프로젝트에 데이터베이스 서버 추가 섹션을 참조하세요.
3. 대상 데이터베이스 서버를 추가하거나 가상 대상 데이터베이스 플랫폼을 사용합니다. 자세한 내용은 가상 대상 사용 섹션을 참조하세요.
4. 매핑 규칙을 생성하여 소스 데이터베이스의 대상 데이터베이스 플랫폼을 정의합니다. 자세한 내용은 AWS SCT에서 매핑 규칙 생성 섹션을 참조하세요.
5. 보기를 선택하고 Main view를 선택합니다.
6. 소스 데이터베이스의 객체를 표시하는 왼쪽 패널에서 소스 데이터베이스 스키마를 선택합니다. 객체의 컨텍스트(마우스 오른쪽 버튼 클릭) 메뉴를 열고 Load schema를 선택합니다. 이 작업은 모든 소스 스키마 메타데이터를 AWS SCT 프로젝트에 로드합니다.

   보고서 생성 및 스키마 변환 작업도 모든 소스 스키마 메타데이터를 AWS SCT 프로젝트에 로드합니다. 컨텍스트 메뉴에서 이러한 작업 중 하나를 실행한 경우에는 Load schema 작업을 건너뜁니다.
7. 파일 메뉴에서 Save project를 선택하여 소스 데이터베이스 메타데이터를 프로젝트에 저장합니다.
8. Disconnect from the server를 선택하여 소스 데이터베이스와의 연결을 끊습니다. 이제 오프라인 모드에서 AWS SCT를 사용할 수 있습니다.

AWS SCT 트리 필터 사용

소스에서 대상으로 데이터를 마이그레이션하기 위해 AWS SCT는 소스 및 대상 데이터베이스의 모든 메타데이터를 트리 구조에 로드합니다. 이 구조는 AWS SCT에서 기본 프로젝트 창의 트리 보기로 나타납니다.

일부 데이터베이스는 트리 구조에 많은 객체가 포함될 수 있습니다. AWS SCT에서 트리 필터를 사용하면 소스 및 대상 트리 구조에서 객체를 검색할 수 있습니다. 트리 필터를 사용하면 데이터베이스를 변환할 때 변환되는 객체를 변경할 수 없습니다. 필터는 트리에 표시되는 항목만 변경합니다.

트리 필터는 AWS SCT가 미리 로드된 객체에서 작동합니다. 즉, 검색 중에는 AWS SCT가 데이터베이스에서 객체를 로드하지 않습니다. 이 방법을 사용하면 일반적으로 트리 구조에는 데이터베이스에 있는 객체 수보다 적은 수의 객체가 포함됩니다.

트리 필터에서 다음 사항에 유의합니다.

- 필터 기본값은 ANY입니다. 즉, 필터는 이름 검색을 사용하여 객체를 찾습니다.
- 하나 이상의 객체 유형을 선택하면 트리에 해당 유형의 객체만 표시됩니다.
- 필터 마스크를 사용하여 유니코드, 공백, 특수 문자 등 다양한 유형의 기호를 표시할 수 있습니다. “%” 문자는 모든 기호의 와일드카드입니다.
- 필터를 적용한 후에는 필터링된 객체의 수만 표시됩니다.

트리 필터를 만들려면

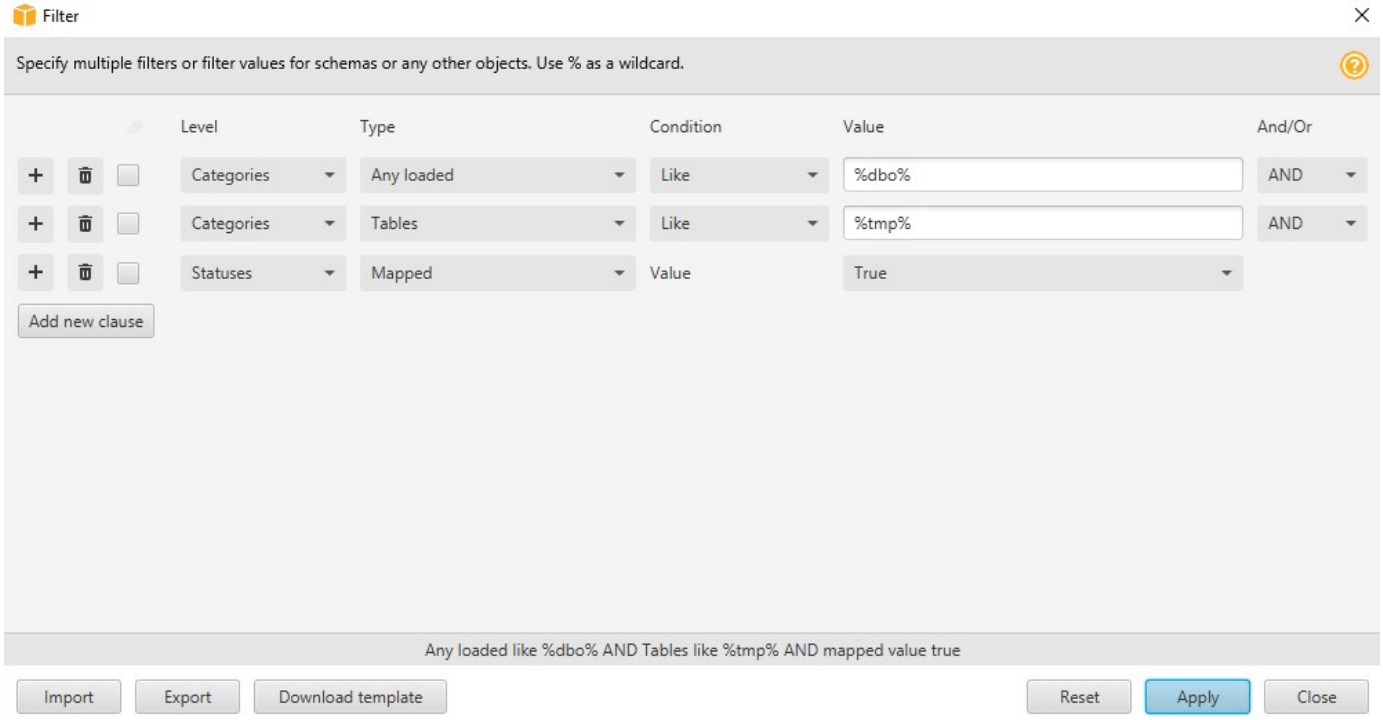
1. 기존 AWS SCT 프로젝트를 엽니다.
2. 트리 필터를 적용할 데이터베이스에 연결합니다.
3. 필터 아이콘을 선택합니다.

   현재는 적용된 필터가 없기 때문에 필터 실행 취소 아이콘이 회색으로 표시됩니다.
4. 필터 대화 상자에 다음 정보를 입력합니다. 대화 상자의 옵션은 각 데이터베이스 엔진마다 다릅니다.

AWS SCT 필터 옵션	작업
수준	범주별로 객체를 필터링하려면 범주를 선택합니다.

AWS SCT 필터 옵션	작업
	상태를 기준으로 객체를 필터링하려면 상태를 선택합니다.
유형	수준의 범주에서 필터링된 객체의 범주를 선택합니다. 모든 범주의 객체를 표시하려면 Any loaded를 선택합니다. 수준의 상태에서 필터링된 객체의 상태를 선택합니다. 다음 옵션 중 하나를 선택할 수 있습니다. • 변환된 모든 객체를 표시하려면 변환됨을 선택합니다. • 변환 문제가 있는 모든 객체를 표시하려면 작업 있음을 선택합니다. • 암호화된 모든 객체를 표시하려면 암호화됨을 선택합니다.
Condition	수준의 범주에서 Like와 Not like사이에서 필터링 조건을 선택합니다. 수준의 상태에서는 필터링 조건 옵션을 사용할 수 없습니다.
값	수준의 범주에서 값을 입력하여 이 값을 기준으로 트리를 필터링합니다. 모든 객체를 표시하려면 퍼센트(%)를 와일드카드로 사용합니다. 수준의 상태에서 True와 False 사이의 값을 선택합니다.
And/Or	여러 필터 절을 적용하려면 OR 또는 AND 논리 연산자를 선택합니다.



5. Add new clause를 선택하여 필터 절을 더 추가합니다. AWS SCT는 AND 또는 OR 논리 연산자를 사용하여 여러 필터 절을 적용할 수 있습니다.
6. Apply(적용)를 선택합니다. 적용을 선택하면 필터 실행 취소 아이콘(필터 아이콘 옆에 위치)이 활성화됩니다. 적용된 필터를 제거하려면 이 아이콘을 사용합니다.
7. [Close]를 선택하여 대화 상자를 닫습니다.

트리에 표시되는 스키마를 필터링하면 스키마를 변환할 때 변환되는 객체가 변경되지 않습니다. 필터는 트리에 표시되는 항목만 변경합니다.

트리 필터의 파일 목록 가져오기

세미콜론 구분자가 있는 쉼표로 구분된 값(CSV) 파일 또는 트리 필터에서 사용할 이름이나 값이 포함된 JSON 파일을 가져올 수 있습니다. 기존 AWS SCT 프로젝트를 열고 트리 필터를 적용할 데이터베이스에 연결한 다음 필터 아이콘을 선택합니다.

파일 예제를 다운로드하려면 템플릿 다운로드를 선택합니다. 파일 이름을 입력하고 저장을 선택합니다.

기존 필터 설정을 다운로드하려면 내보내기를 선택합니다. 파일 이름을 입력하고 저장을 선택합니다.

트리 필터에 대한 파일 목록을 가져오려면 가져오기를 선택합니다. 가져올 파일을 선택한 후 열기를 선택합니다. [Apply]를 선택한 다음 [Close]를 선택합니다.

CSV 파일은 세미콜론을 구분자로 사용하며 형식은 다음과 같습니다.

- `object_type`은 찾으려는 객체의 유형입니다.
- `database_name`은 이 객체가 있는 데이터베이스의 이름입니다.
- `schema_name`은 이 객체가 있는 스키마의 이름입니다.
- `object_name`은 객체 이름입니다.
- `import_type`은 필터에서 이 항목에 대해 `exclude` 또는 `include`를 수행하도록 지정합니다.

중첩 규칙과 같은 복잡한 필터링 사례를 설명하려면 JSON 파일을 사용합니다. JSON 파일의 형식은 다음과 같습니다.

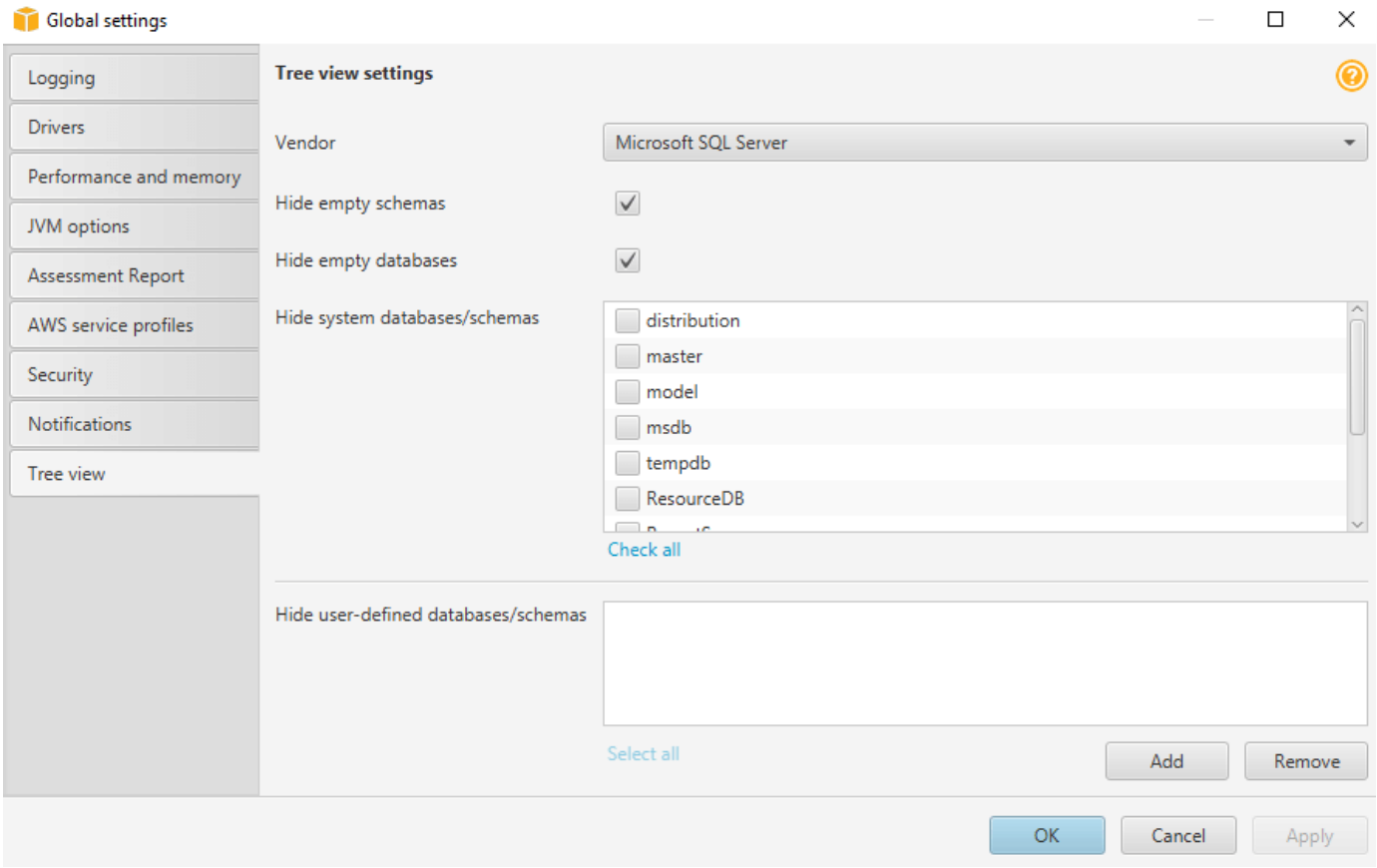
- `filterGroupType`은 여러 필터 절에 적용되는 필터 규칙 유형(`AND` 또는 `OR` 논리 연산자)입니다.
- `filterCategory`는 필터의 수준(범주 또는 상태)입니다.
- `names`는 범주 필터에 적용되는 객체 이름 목록입니다.
- `filterCondition`은 범주 필터에 적용되는 필터링 조건(`LIKE` 또는 `NOT LIKE`)입니다.
- `transformName`은 상태 필터에 적용되는 상태 이름입니다.
- `value`는 트리를 필터링하는 기준 값입니다.
- `transformValue`는 상태 필터에 적용되는 필터의 값(`TRUE` 또는 `FALSE`)입니다.

AWS SCT 트리 보기에서 스키마 숨기기

트리 보기 설정을 사용하여 AWS SCT 트리 보기에 표시할 스키마와 데이터베이스를 지정할 수 있습니다. 빈 스키마, 빈 데이터베이스, 시스템 데이터베이스, 사용자 정의 데이터베이스 및 스키마를 숨길 수 있습니다.

트리 보기에서 데이터베이스와 스키마를 숨기려면

1. AWS SCT 프로젝트를 엽니다.
2. 트리 보기에 표시할 데이터 스토어에 연결합니다.
3. 설정, 전역 설정, 트리 보기를 선택합니다.



4. Tree view settings 섹션에서 다음을 수행합니다.

   - 공급업체에서 데이터베이스 플랫폼을 선택합니다.
   - 선택한 데이터베이스 플랫폼의 빈 스키마를 숨기려면 빈 스키마 숨기기를 선택합니다.
   - 선택한 데이터베이스 플랫폼의 빈 데이터베이스를 숨기려면 빈 데이터베이스 숨기기를 선택합니다.
   - Hide system databases/schemas에서 시스템 데이터베이스와 스키마를 이름별로 선택하여 숨깁니다.
   - Hide user-defined databases/schemas에 숨기려는 사용자 정의 데이터베이스 및 스키마의 이름을 입력한 다음 추가를 선택합니다. 이름은 대/소문자를 구분하지 않습니다.

5. 확인(OK)을 선택합니다.

데이터베이스 마이그레이션 평가 보고서 생성 및 검토

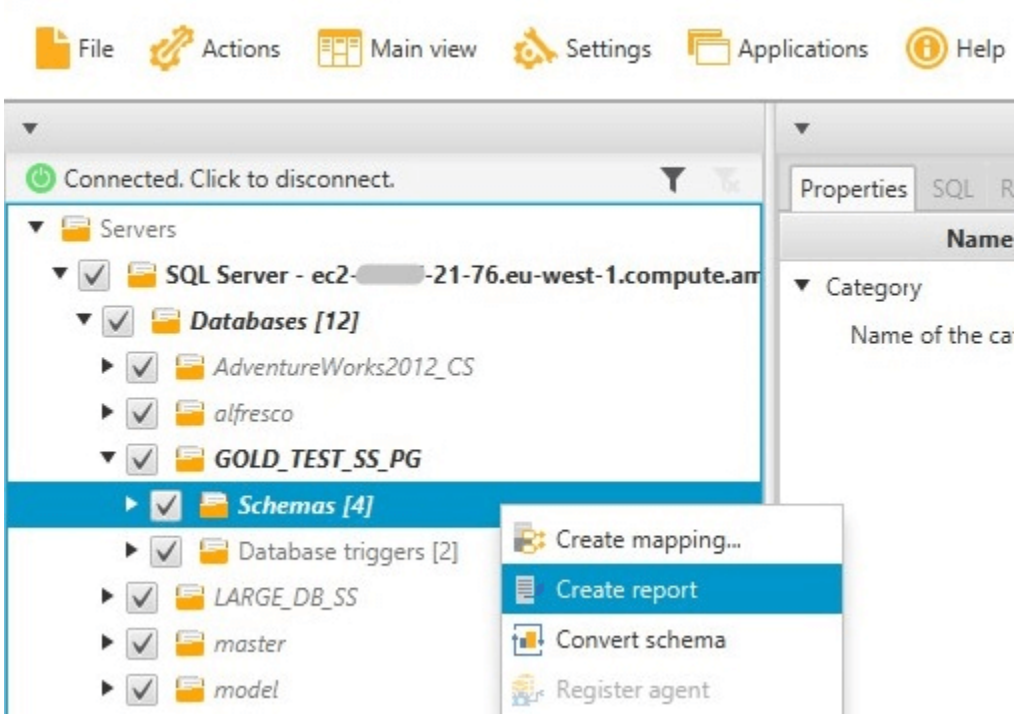
데이터베이스 마이그레이션 평가 보고서에는 대상 Amazon RDS DB 인스턴스의 엔진으로 자동 변환할 수 없는 스키마에 대한 모든 작업 항목이 요약되어 있습니다. 또한 이 보고서에는 대상 DB 인스턴스에 대해 동등한 코드를 작성하는 데 소요되는 예상 작업량도 포함되어 있습니다.

소스 데이터베이스와 대상 플랫폼을 프로젝트에 추가하고 매핑 규칙을 지정한 후 데이터베이스 마이그레이션 평가 보고서를 생성할 수 있습니다.

데이터베이스 마이그레이션 평가 보고서를 생성하고 보려면

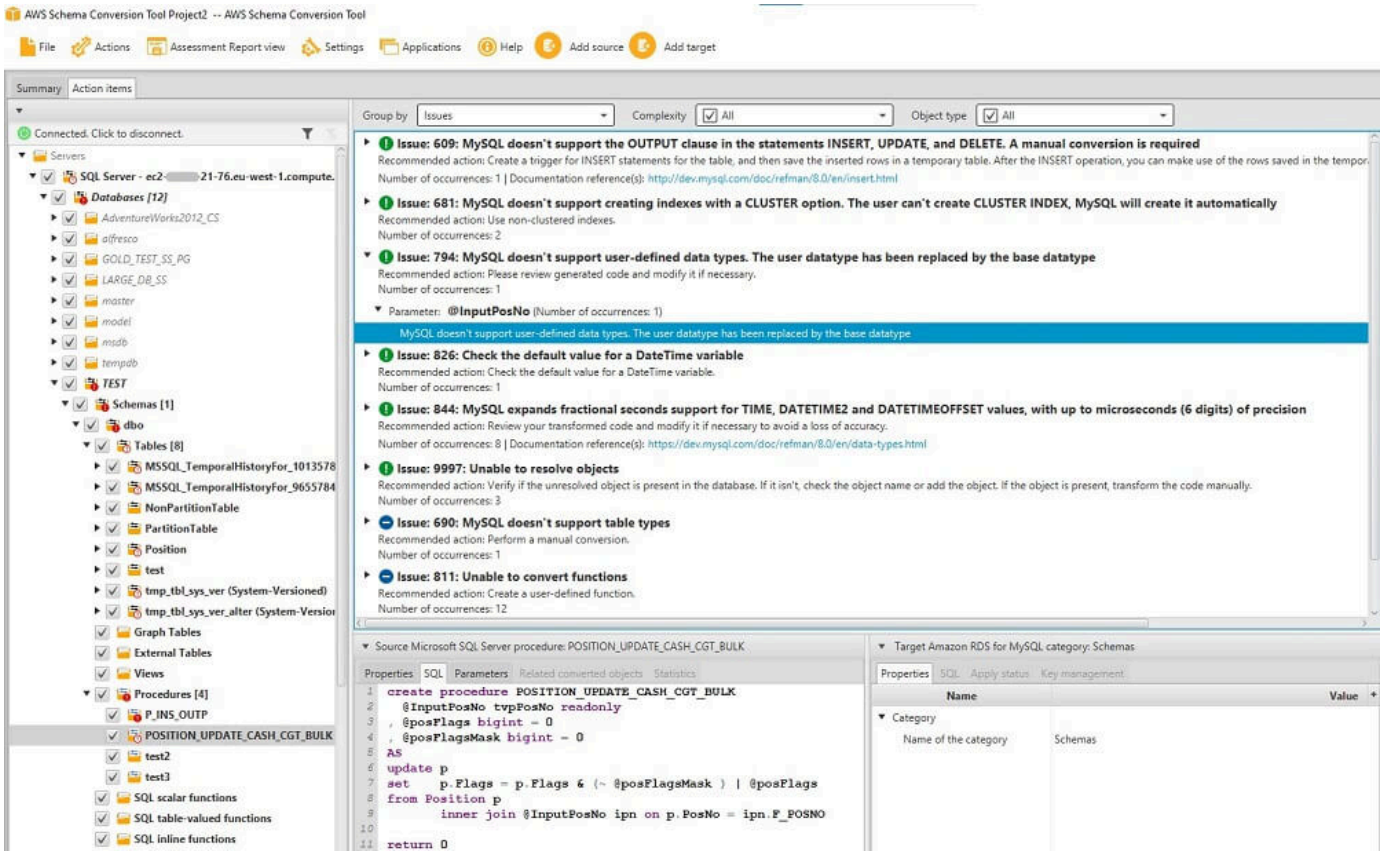
1. 평가 보고서를 생성하는 데 사용할 소스 데이터베이스 스키마에 대한 매핑 규칙을 생성했는지 확인합니다. 자세한 내용은 새 매핑 규칙 추가 섹션을 참조하세요.
2. 보기 메뉴에서 Main view를 선택합니다.
3. 소스 데이터베이스의 스키마를 표시하는 왼쪽 패널에서 평가 보고서를 생성할 스키마 객체를 선택합니다.

   평가 보고서를 만들 때 사용할 모든 스키마 객체의 확인란을 선택했는지 확인합니다.
4. 객체의 컨텍스트(마우스 오른쪽 버튼 클릭) 메뉴를 열고 보고서 생성을 선택합니다.



   평가 보고서 보기가 열립니다.
5. 작업 항목 탭을 선택합니다.

   작업 항목 탭에는 자동으로 변환할 수 없는 스키마를 설명하는 항목 목록이 표시됩니다. 목록에서 작업 항목 중 하나를 선택합니다. AWS SCT는 다음과 같이 작업 항목이 적용되는 스키마의 항목을 강조 표시합니다.

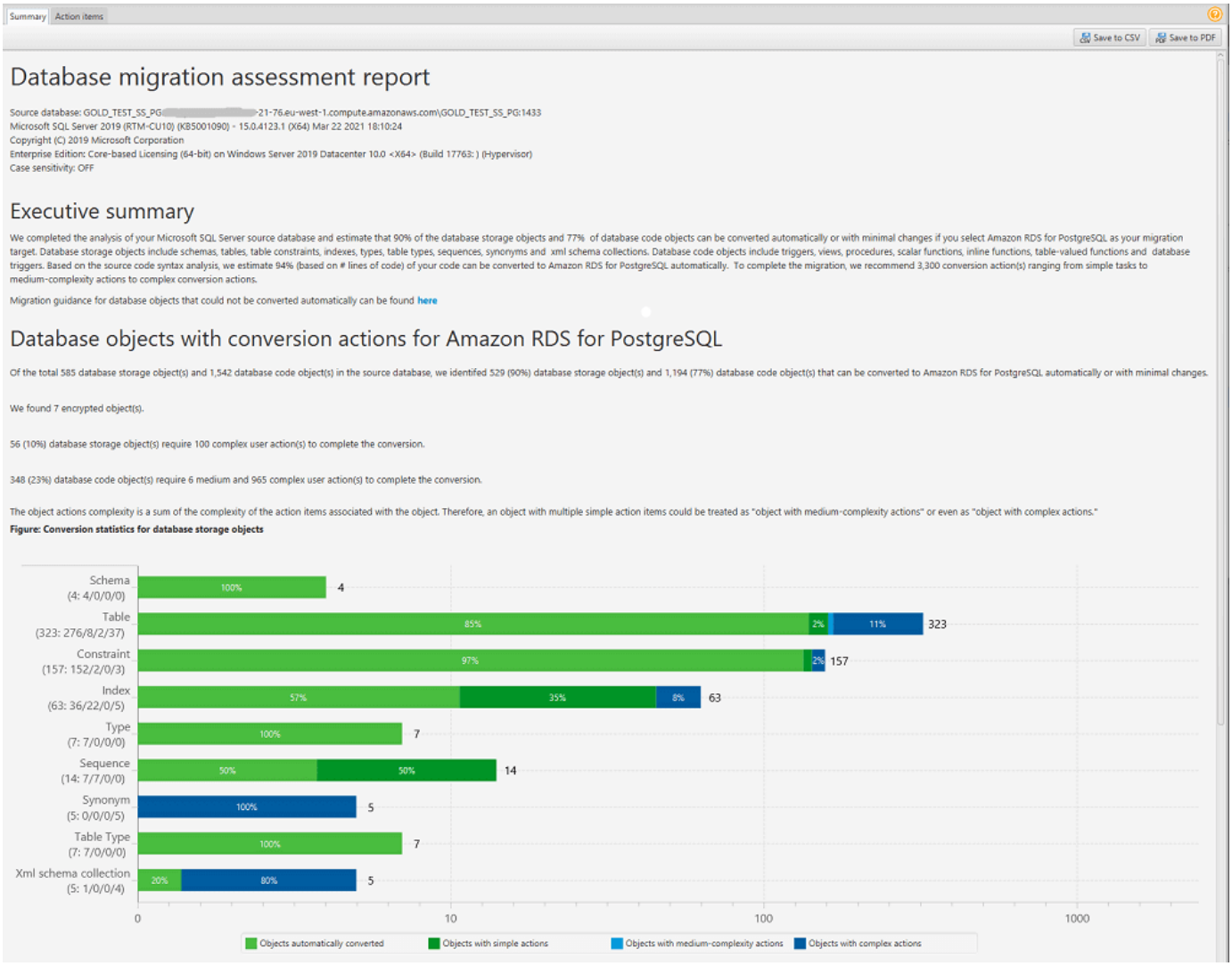


6. 요약 탭을 선택합니다.

   요약 탭에는 데이터베이스 마이그레이션 평가 보고서의 요약 정보가 표시됩니다. 자동으로 변환된 항목 수와 자동으로 변환되지 않은 항목 수가 표시됩니다. 또한 요약에는 소스 데이터베이스의 스키마와 동일한 스키마를 대상 DB 인스턴스에서 생성하는 데 소요되는 예상 시간도 포함됩니다.

   라이선스 평가 및 클라우드 지원 섹션에는 동일한 엔진을 실행하는 Amazon RDS DB 인스턴스로 기존 온프레미스 데이터베이스 스키마를 이동하는 방법에 대한 정보가 포함되어 있습니다. 예를 들어 라이선스 유형을 변경하려는 경우 보고서의 이 섹션에서는 현재 데이터베이스에서 제거할 기능을 설명합니다.

   평가 보고서 요약의 예는 다음과 같습니다.



7. 요약 탭을 선택한 후 Save to PDF를 선택합니다. 데이터베이스 마이그레이션 평가 보고서가 PDF 파일로 저장됩니다. PDF 파일에는 요약 정보와 작업 항목 정보가 모두 수록돼 있습니다.

   또한 Save to CSV를 선택하여 보고서를 CSV 파일로 저장할 수 있습니다. 이 옵션을 선택하면 AWS SCT에서 세 개의 CSV 파일을 생성합니다. 이러한 파일에는 다음과 같은 정보가 포함되어 있습니다.

   - 권장 작업이 포함된 변환 작업 항목 목록
   - 발생 작업 항목을 변환하는 데 필요한 예상 작업량이 포함된 변환 작업 항목 요약 정보
   - 예상 변환 시간을 기준으로 분류된 여러 작업 항목이 포함된 핵심 요약 정보

Database objects with conversion actions for Amazon RDS for PostgreSQL

Of the total 585 database storage object(s) and 1,542 database code object(s) in the source database, we identifed 529 (90%) database storage object(s) and 1,194 (77%) database code object(s) that can be converted to Amazon RDS for PostgreSQL automatically or with minimal changes.

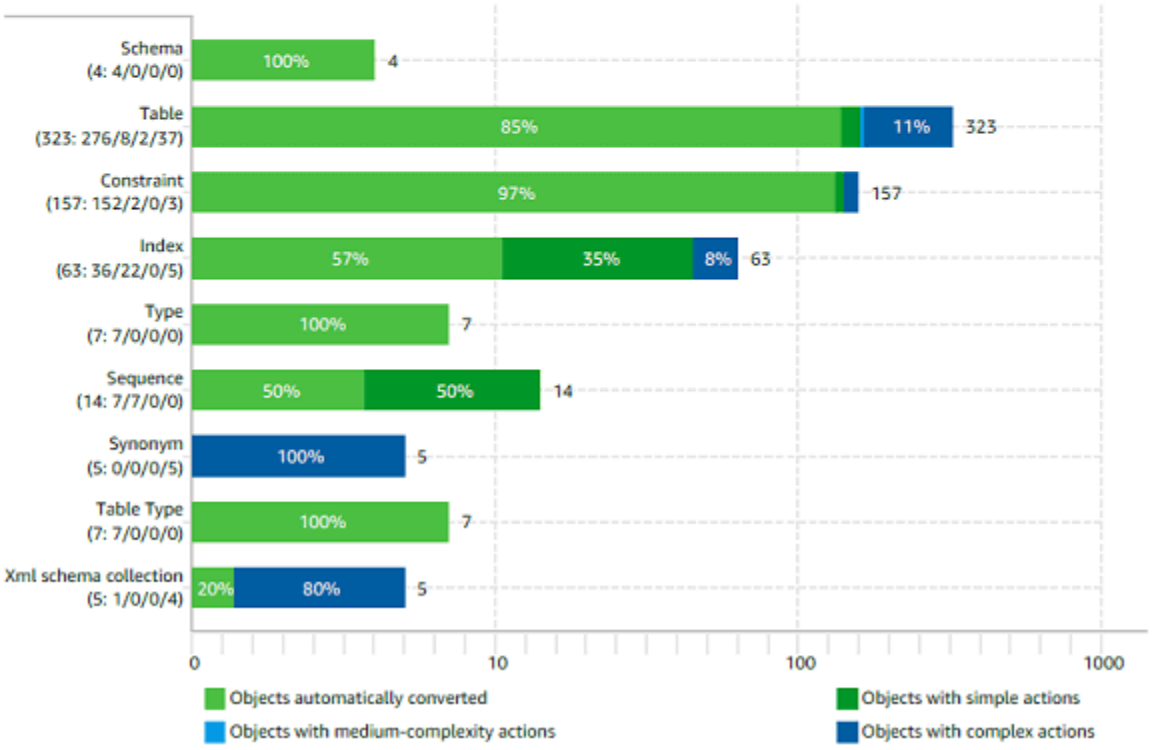
We found 7 encrypted object(s).

56 (10%) database storage object(s) require 100 complex user action(s) to complete the conversion.

348 (23%) database code object(s) require 6 medium and 965 complex user action(s) to complete the conversion.

The object actions complexity is a sum of the complexity of the action items associated with the object. Therefore, an object with multiple simple action items could be treated as "object with medium-complexity actions" or even as "object with complex actions."

Figure: Conversion statistics for database storage objects

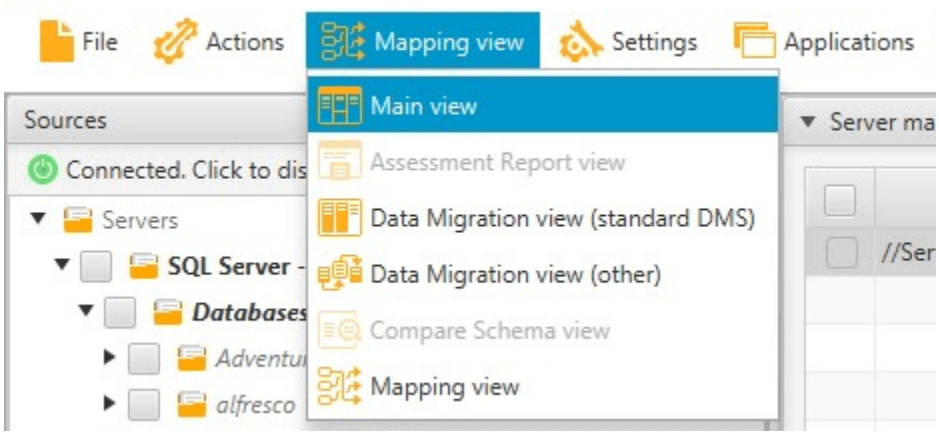


스키마 변환

프로젝트에 소스 및 대상 데이터베이스를 추가하고 매핑 규칙을 생성한 후 소스 데이터베이스 스키마를 변환할 수 있습니다. 스키마를 변환하려면 다음 절차를 따릅니다.

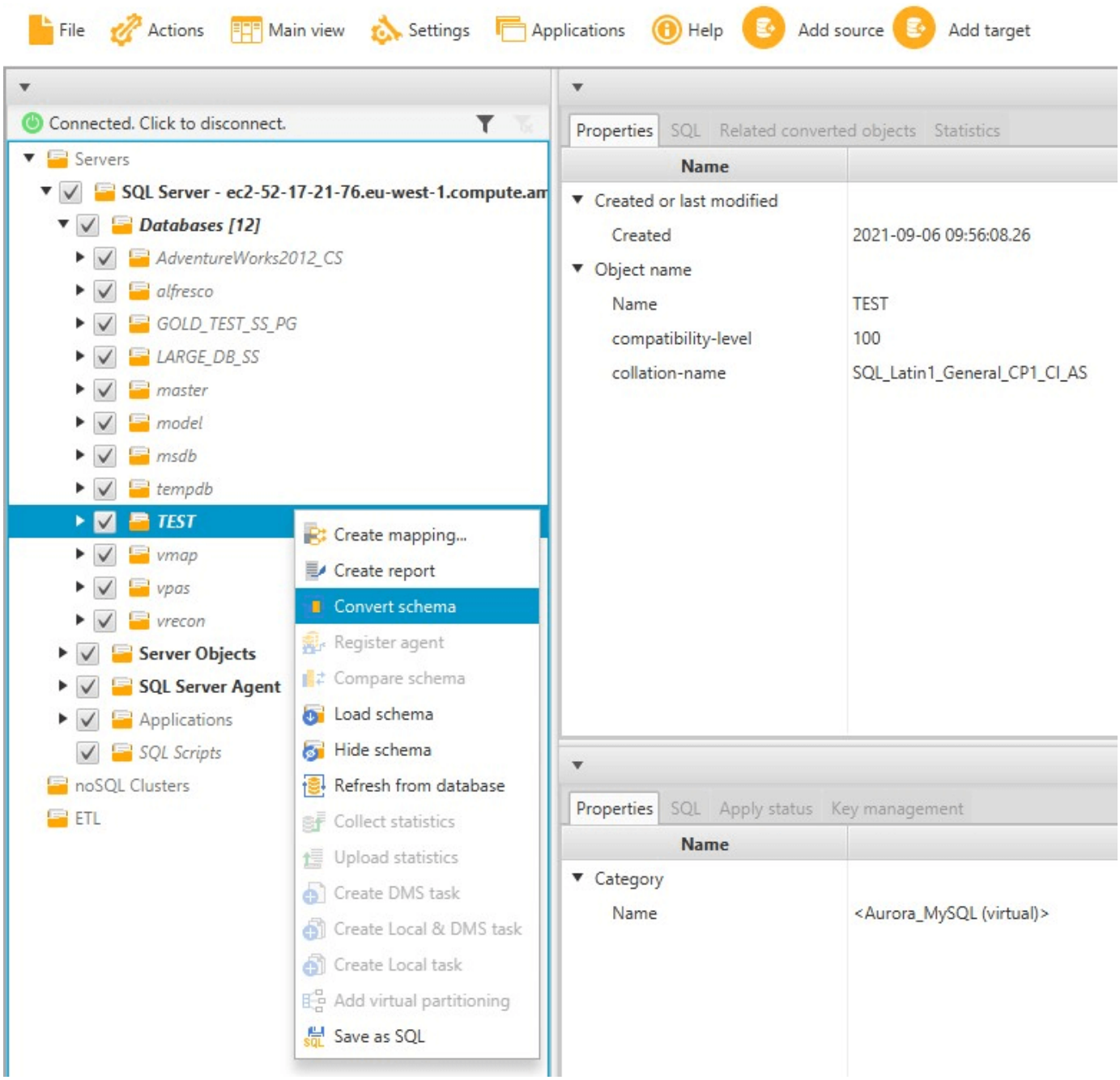
스키마를 변환하려면

1. 보기를 선택하고 Main view를 선택합니다.



2. 소스 데이터베이스의 스키마를 표시하는 왼쪽 패널에서 변환할 객체 이름에 해당하는 확인란을 선택합니다. 그 다음, 이 객체를 선택합니다. AWS SCT가 해당 객체 이름을 파란색으로 강조 표시합니다. 객체의 컨텍스트(마우스 오른쪽 버튼 클릭) 메뉴를 열고 스키마 변환을 선택합니다.

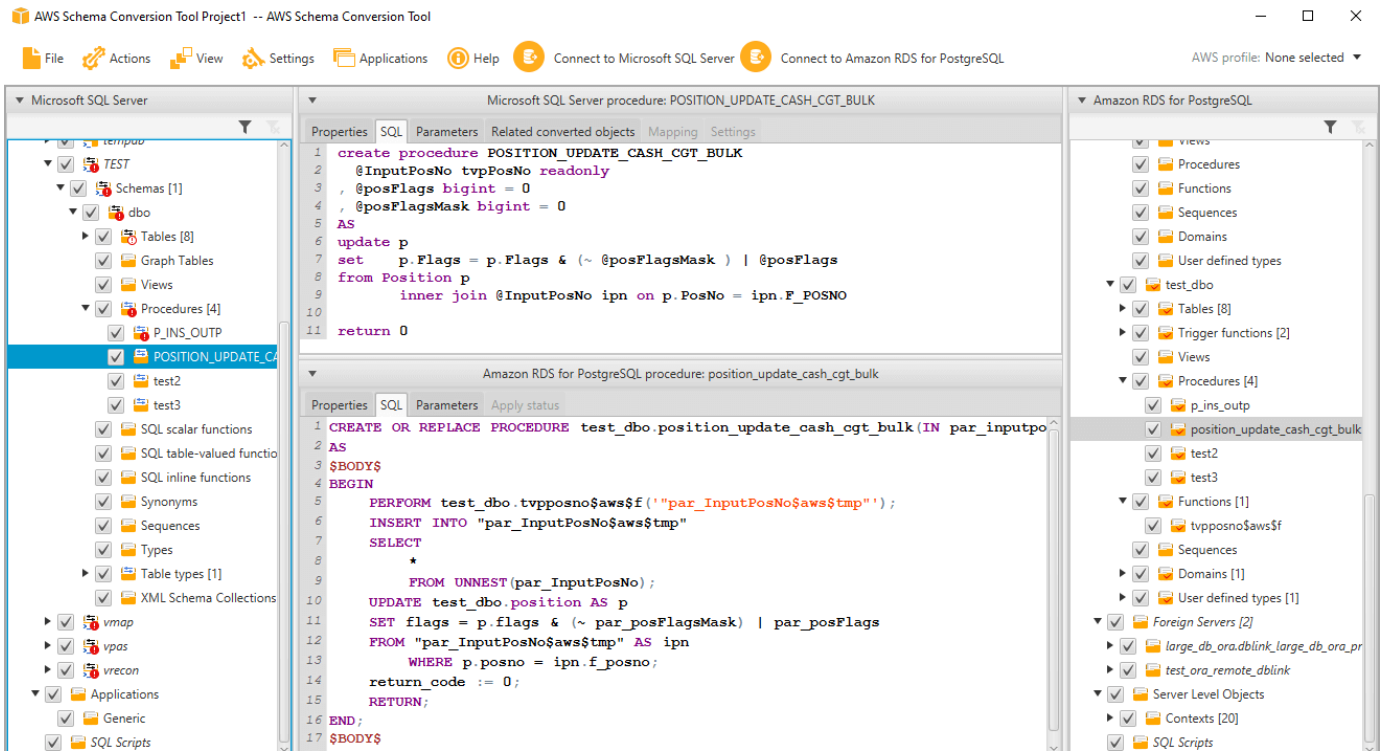
   여러 데이터베이스 객체를 변환하려면 모든 개체의 확인란을 선택합니다. 그 다음으로, 상위 노드를 선택합니다. 예를 들어 테이블에서 상위 노드는 Tables입니다. AWS SCT가 상위 노드의 이름을 파란색으로 강조 표시했는지 확인합니다. 상위 노드의 컨텍스트(마우스 오른쪽 버튼 클릭) 메뉴를 열고 스키마 변환을 선택합니다.



3. AWS SCT가 스키마 변환을 완료하면 프로젝트 오른쪽의 패널에서 제안된 스키마를 볼 수 있습니다.

   이 시점에서는 대상 데이터베이스 인스턴스에 스키마가 적용되지 않습니다. 계획된 스키마는 프로젝트의 일부입니다. 변환된 스키마 항목을 선택하면 대상 데이터베이스 인스턴스의 하단 중앙에 있는 패널에서 계획된 스키마 명령을 볼 수 있습니다.

   이 창에서 스키마를 편집할 수 있습니다. 편집된 스키마는 프로젝트의 일부로 저장되며 변환된 스키마를 적용하도록 선택하면 대상 데이터베이스 인스턴스에 기록됩니다.



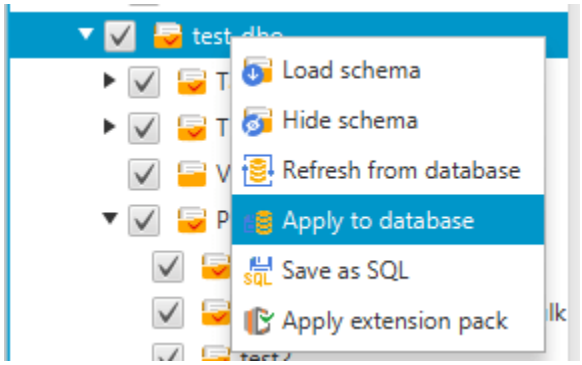
변환된 스키마를 대상 DB 인스턴스에 적용

변환된 데이터베이스 스키마를 대상 DB 인스턴스에 적용할 수 있습니다. 대상 DB 인스턴스에 스키마를 적용한 후 데이터베이스 마이그레이션 평가 보고서의 작업 항목을 기반으로 스키마를 업데이트할 수 있습니다.

> **Warning**
>
> 다음 절차는 기존 대상 스키마를 덮어씁니다. 실수로 스키마를 덮어쓰지 않도록 주의합니다. 이미 수정한 대상 DB 인스턴스의 스키마를 덮어쓰거나 해당 변경 내용을 덮어쓰지 않도록 주의해야 합니다.

변환된 데이터베이스 스키마를 대상 데이터베이스 인스턴스에 적용하려면

1. 프로젝트 오른쪽 패널 상단에서 Connect to the server를 선택하여 대상 데이터베이스에 연결합니다. 대상 데이터베이스에 이미 연결되어 있는 경우에는 이 단계를 건너뜁니다.
2. 프로젝트의 오른쪽 패널에서 대상 DB 인스턴스의 계획된 스키마를 표시하는 스키마 요소를 선택합니다.
3. 스키마 요소의 컨텍스트(마우스 오른쪽 버튼 클릭) 메뉴를 열고 Apply to database를 선택합니다.



변환된 스키마가 대상 DB 인스턴스에 적용됩니다.

AWS SCT에 AWS 서비스 프로필 저장

AWS SCT에 AWS 보안 인증 정보를 저장할 수 있습니다. AWS SCT는 AWS 서비스와 통합되는 기능을 사용할 때 보안 인증 정보를 사용합니다. 예를 들어, AWS SCT는 Amazon S3, AWS Lambda, Amazon Relational Database Service(RDS) 및 AWS Database Migration Service(AWS DMS)와 통합됩니다.

AWS SCT는 보안 인증이 필요한 기능에 액세스할 때 AWS 보안 인증 정보를 물어봅니다. 전역 애플리케이션 설정에 보안 인증 정보를 저장할 수 있습니다. AWS SCT가 보안 인증 정보를 묻는 메시지가 표시되면 저장된 보안 인증 정보를 선택할 수 있습니다.

전역 애플리케이션 설정에 여러 AWS 보안 인증 정보 세트를 저장할 수 있습니다. 예를 들어 테스트 시나리오에서 사용하는 보안 인증 정보 세트 하나와 프로덕션 시나리오에서 사용하는 다른 보안 인증 정보 세트를 저장할 수 있습니다. 또한 여러 AWS 리전에 대해 서로 다른 보안 인증 정보를 저장할 수 있습니다.

AWS 보안 인증 정보 저장

AWS 보안 인증 정보를 전역으로 저장하려면 다음 절차를 사용합니다.

AWS 보안 인증 정보를 저장하려면

1. AWS Schema Conversion Tool을 시작합니다.
2. 설정 메뉴를 열고 전역 설정을 선택합니다. 전역 설정 대화 상자가 나타납니다.
3. AWS service profiles를 선택한 다음 Add a new AWS service profile을 선택합니다.
4. 다음과 같이 AWS 정보를 입력합니다.

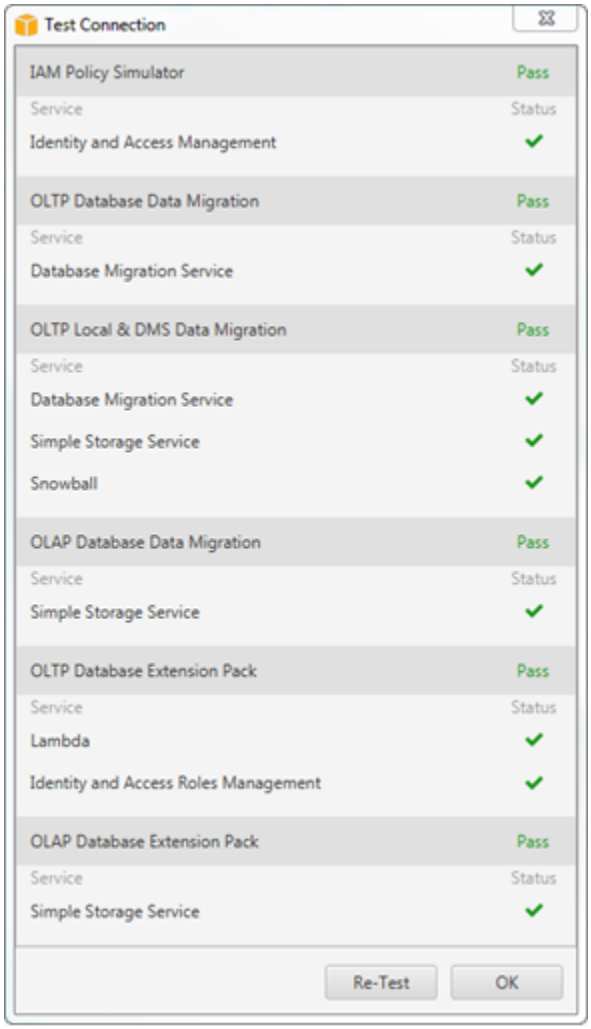
AWS SCT 옵션	작업
프로필 이름	프로필의 이름을 입력합니다.
AWS 액세스 키	AWS 액세스 키를 입력합니다.
AWS 비밀 키	AWS 비밀 액세스 키를 입력합니다. AWS 액세스 키에 대한 자세한 내용은 IAM 사용 설명서에서 액세스 키 관리를 참조하세요.
리전(Region)	프로필의 AWS 리전을 선택합니다.
Amazon S3 bucket folder	프로필의 Amazon S3 버킷을 선택합니다. Amazon S3에 연결하는 기능을 사용하는 경우에만 버킷을 지정해야 합니다. 필요한 권한에 대한 자세한 내용은 AWS 서비스 프로필 사용 권한 섹션을 참조하세요.

Federal Information Processing Standard(FIPS)의 보안 요구 사항을 준수해야 하는 경우 Use FIPS endpoint for S3을 선택합니다. FIPS 엔드포인트는 다음 AWS 리전에서 사용할 수 있습니다.

- US East (N. Virginia) Region
- 미국 동부(오하이오) 리전
- 미국 서부(캘리포니아 북부) 리전
- 미국 서부(오레곤) 리전

5. 연결 테스트를 선택하여 보안 인증 정보가 정확하고 활성 상태인지 확인합니다.

   연결 테스트 대화 상자가 나타납니다. 프로필에 연결된 각 서비스의 상태를 볼 수 있습니다. 통과는 프로필이 서비스에 성공적으로 액세스할 수 있음을 나타냅니다.



6. 프로필을 구성한 후 저장을 선택하여 프로필을 저장하거나 취소를 선택하여 변경 내용을 취소합니다.
7. 확인을 선택하여 전역 설정 대화 상자를 닫습니다.

프로젝트의 기본 프로필 설정

AWS SCT 프로젝트의 기본 프로필을 설정할 수 있습니다. 이렇게 하면 프로필에 저장된 AWS 보안 인증 정보가 프로젝트와 연결됩니다. 프로젝트를 연 상태에서 다음 절차에 따라 기본 프로필을 설정합니다.

프로젝트의 기본 프로필을 설정하려면

1. AWS Schema Conversion Tool을 시작하고 새 프로젝트를 생성합니다.
2. 설정 메뉴에서 프로젝트 설정을 선택합니다. 프로젝트 설정 대화 상자가 나타납니다.

3. Project environment 탭을 선택합니다.
4. Add a new AWS service profile을 선택하여 새 프로필을 추가합니다. 그런 다음, AWS service profile에서 프로젝트와 연결할 프로필을 선택합니다.
5. 확인을 선택하여 프로젝트 설정 대화 상자를 닫습니다. 취소를 선택하여 변경 사항을 취소할 수도 있습니다.

AWS 서비스 프로필 사용 권한

AWS 서비스 프로필에서 Amazon S3 버킷에 액세스하려면 다음과 같은 권한이 필요합니다.

- `s3:PutObject` - Amazon S3 버킷에 객체를 추가합니다.
- `s3:DeleteObject` – 객체의 null 버전을 제거하고 삭제 마커를 삽입합니다. 이 삭제 마커가 객체의 현재 버전이 됩니다.
- `s3:ListBucket` - Amazon S3 버킷에서 최대 1,000개의 객체를 반환합니다.
- `s3:GetObject` - Amazon S3 버킷에서 객체를 검색합니다.

다음 코드 예제는 사용자에게 이러한 권한을 부여하는 방법을 보여줍니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:DeleteObject",
                "s3:GetObject",
                "s3:ListBucket",
                "s3:PutObject"
            ],
            "Resource": [
                "*"
            ]
        }
    ]
}
```

AWS Secrets Manager 사용

AWS SCT는 AWS Secrets Manager에 저장된 데이터베이스 보안 인증을 사용할 수 있습니다. Secrets Manager에서 데이터베이스 연결 대화 상자의 모든 값을 채울 수 있습니다. Secrets Manager를 사용하려면 AWS 프로필을 AWS Schema Conversion Tool에 저장해야 합니다.

AWS Secrets Manager 사용에 대한 자세한 내용은 AWS Secrets Manager 사용 설명서의 AWS Secrets Manager란 무엇인가요?를 참조하세요. AWS 프로필 저장에 대한 자세한 내용은 AWS SCT에 AWS 서비스 프로필 저장 섹션을 참조하세요.

Secrets Manager에서 데이터베이스 보안 인증 정보를 검색하려면

1. AWS Schema Conversion Tool을 시작하고 새 프로젝트를 생성합니다.
2. 소스 추가 또는 대상 추가를 선택하여 프로젝트에 새 데이터베이스를 추가합니다.
3. 데이터베이스 플랫폼을 선택한 후 다음을 선택합니다.
4. AWS Secret에서 사용할 보안 암호를 선택합니다.
5. Populate를 선택합니다. 그러면 AWS SCT가 데이터베이스 연결 대화 상자의 모든 값을 입력합니다.
6. 연결 테스트를 선택하여 AWS SCT가 데이터베이스에 연결할 수 있는지 확인합니다.
7. 연결을 선택하여 데이터베이스에 연결합니다.

AWS SCT는 다음 구조를 갖는 보안 암호를 지원합니다.

```
{
  "username": "secret_user",
  "password": "secret_password",
  "engine": "oracle",
  "host": "secret_host.eu-west-1.compute.amazonaws.com",
  "port": "1521",
  "dbname": "ora_db"
}
```

이 구조에서는 `username` 및 `password` 값이 필수이고 다른 모든 값은 선택 사항입니다. Secrets Manager에 저장하는 값에는 모든 데이터베이스 보안 인증 정보가 포함되어야 합니다.

데이터베이스 암호 저장

AWS SCT 캐시에 데이터베이스 암호 또는 SSL 인증서를 저장할 수 있습니다. 암호를 저장하려면 연결을 생성할 때 Store Password를 선택합니다.

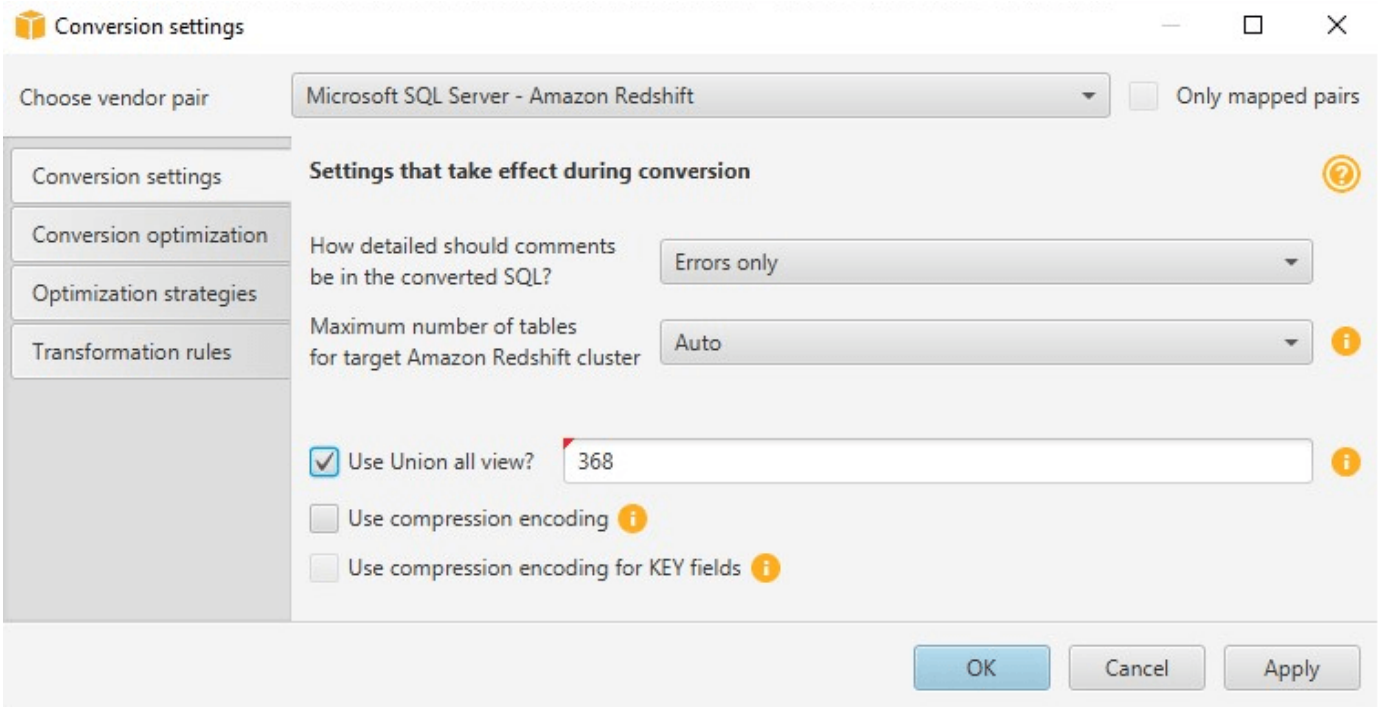
암호는 `seed.dat` 파일에서 임의로 생성된 토큰을 사용하여 암호화됩니다. 그런 다음, 암호가 사용자 이름과 함께 캐시 파일에 저장됩니다. `seed.dat` 파일을 분실하거나 파일이 손상된 경우 데이터베이스 암호가 잘못 암호화 해제될 수 있습니다. 이 경우 연결이 실패합니다.

파티션 테이블이 있는 프로젝트에 UNION ALL 보기 사용

소스 테이블이 분할된 경우 AWS SCT는 n개의 대상 테이블을 생성합니다. 여기서 n은 소스 테이블의 파티션 수입니다. AWS SCT는 대상 테이블을 기반으로 UNION ALL 보기를 생성하여 소스 테이블을 나타냅니다. AWS SCT 데이터 추출기를 사용하여 데이터를 마이그레이션하는 경우 소스 테이블 파티션은 별도의 하위 작업에 의해 병렬로 추출 및 로드됩니다.

프로젝트에 Union All 보기를 사용하려면

1. Start AWS SCT. 새 프로젝트를 생성하거나 기존 AWS SCT 프로젝트를 엽니다.
2. 설정 메뉴에서 변환 설정을 선택합니다.
3. 상단의 목록에서 OLAP 데이터베이스 페어를 선택합니다.
4. Use Union all view?를 켭니다.



5. 확인을 선택하여 설정을 저장하고 변환 설정 대화 상자를 닫습니다.

AWS SCT의 키보드 바로 가기

AWS SCT에서 사용할 수 있는 키보드 바로 가기는 다음과 같습니다.

키보드 바로 가기	설명
Ctrl+N	새 프로젝트를 생성합니다
Ctrl+O	기존 프로젝트를 엽니다.
Ctrl+S	열려 있는 프로젝트를 저장합니다.
Ctrl+W	마법사를 사용하여 새 프로젝트를 생성합니다.
Ctrl+M	새 다중 서버 평가를 생성합니다.
Ctrl+L	새 소스 데이터베이스를 추가합니다.
Ctrl+R	새 대상 데이터베이스를 추가합니다.
Ctrl+F4	열려 있는 프로젝트를 닫습니다.
F1	AWS SCT 사용 설명서를 엽니다.

AWS SCT 시작하기

AWS Schema Conversion Tool(AWS SCT)을 사용하여 소스 데이터베이스의 스키마를 변환할 수 있습니다. 소스 데이터베이스는 온프레미스 또는 Amazon EC2 인스턴스에서 실행되는 자체 관리형 엔진일 수 있습니다. AWS에서 호스팅하는 모든 지원 데이터베이스의 스키마로 소스 스키마를 변환할 수 있습니다. AWS SCT 애플리케이션은 프로젝트 기반 사용자 인터페이스를 제공합니다.

AWS SCT에서 수행하는 거의 모든 작업은 다음 단계부터 시작됩니다.

1. AWS SCT를 설치합니다. 자세한 내용은 설치, 확인 및 업데이트 AWS SCT 섹션을 참조하세요.
2. 필요한 경우 AWS SCT 에이전트를 설치합니다. AWS SCT 에이전트는 특정 마이그레이션 시나리오(예: 이기종 소스와 대상 간)에만 필요합니다. 자세한 내용은 온프레미스 데이터 웨어하우스에서 Amazon Redshift로 데이터 마이그레이션 섹션을 참조하세요.
3. AWS SCT 사용자 인터페이스를 숙지합니다. 자세한 내용은 AWS SCT 사용자 인터페이스 사용 섹션을 참조하세요.
4. AWS SCT 프로젝트를 생성합니다. 원본 및 대상 데이터베이스에 연결합니다. 소스 데이터베이스 연결에 대한 자세한 내용은 AWS SCT 소스 섹션을 참조하세요.
5. 매핑 규칙을 생성합니다. 매핑 규칙에 대한 자세한 내용은 AWS SCT에서 매핑 규칙 생성 섹션을 참조하세요.
6. 데이터베이스 마이그레이션 평가 보고서를 실행한 다음 검토합니다. 평가 보고서에 대한 자세한 내용은 데이터베이스 마이그레이션 평가 보고서 생성 및 검토 섹션을 참조하세요.
7. 소스 데이터베이스 스키마를 변환합니다. 변환되지 않는 항목에 대해 수행할 작업 및 특정 방식으로 변환해야 하는 항목을 매핑하는 방법 등과 같이 변환 시 염두에 두어야 할 몇 가지 사항이 있습니다. 소스 스키마 변환에 대한 자세한 내용은 AWS SCT를 사용하여 데이터베이스 스키마 변환 섹션을 참고하세요.

   데이터 웨어하우스 스키마를 변환하는 경우 변환을 수행하기 전에 고려해야 할 사항도 있습니다. 자세한 내용은 AWS SCT를 사용하여 데이터 웨어하우스 스키마를 Amazon Redshift로 변환 섹션을 참조하세요.
8. 대상에 스키마 변환을 적용합니다. 소스 스키마 변환 적용에 대한 자세한 내용은 변환된 코드 적용 섹션을 참조하세요.
9. 또한 AWS SCT를 사용하면 SQL 저장 프로시저 및 기타 애플리케이션 코드를 변환할 수 있습니다. 자세한 정보는 AWS SCT를 사용하여 애플리케이션 SQL 변환 섹션을 참조하세요.

AWS SCT를 사용하여 소스 데이터베이스의 데이터를 Amazon 관리형 데이터베이스로 마이그레이션할 수도 있습니다. 예를 보려면 온프레미스 데이터 웨어하우스에서 Amazon Redshift로 데이터 마이그레이션 섹션을 참조하세요.

AWS SCT 소스

AWS Schema Conversion Tool(AWS SCT)에서는 다음 소스 데이터베이스 및 데이터 웨어하우스의 스키마를 대상 데이터베이스 또는 데이터 웨어하우스로 변환할 수 있습니다. 권한, 연결 및 AWS SCT에서 대상 데이터베이스 또는 데이터 웨어하우스 용도로 변환할 수 있는 항목에 대한 자세한 내용은 아래 항목에서 세부 정보를 참조하세요.

암호화 정보

Amazon RDS 연결 암호화

데이터베이스 소스

- Apache Cassandra를 소스로 사용
- Azure SQL Database를 소스로 사용
- IBM Db2 for z/OS를 소스로 사용
- IBM Db2 LUW를 소스로 사용
- MySQL을 소스로 사용
- Oracle Database를 소스로 사용
- PostgreSQL을 소스로 사용
- SAP ASE(Sybase ASE)를 소스로 사용
- SQL Server를 소스로 사용

데이터 웨어하우스 소스

- Amazon Redshift를 소스로 사용
- Azure Synapse Analytics를 소스로 사용
- BigQuery를 소스로 사용
- Greenplum Database를 소스로 사용
- Netezza를 소스로 사용
- Oracle Data Warehouse를 소스로 사용
- Snowflake를 소스로 사용
- SQL Server Data Warehouse를 소스로 사용
- Teradata를 소스로 사용

- Vertica를 소스로 사용

빅 데이터 소스

- Apache Hadoop을 소스로 사용
- Apache Oozie를 소스로 사용

AWS SCT에서 Amazon RDS와 Amazon Aurora 연결 암호화

애플리케이션에서 Amazon RDS 또는 Amazon Aurora 데이터베이스로의 암호화된 연결을 열려면 AWS 루트 인증서를 특정 형태의 키 스토리지로 가져와야 합니다. Amazon RDS 사용 설명서의 SSL/TLS를 사용하여 DB 인스턴스에 대한 연결 암호화에서 AWS의 루트 인증서를 다운로드할 수 있습니다.

사용할 수 있는 옵션은 두 가지로, 모든 AWS 리전에 사용할 수 있는 루트 인증서와 이전 루트 인증서 및 새 루트 인증서를 모두 포함하는 인증서 번들입니다.

사용할 항목에 따라 다음 두 절차 중 하나의 단계를 따릅니다.

Windows 시스템 스토리지로 인증서를 가져오려면

1. 다음 소스 중 하나에서 인증서를 다운로드합니다.

   인증서 다운로드에 자세한 내용은 Amazon RDS 사용 설명서에서 SSL/TLS를 사용하여 DB 인스턴스에 대한 연결 암호화를 참조하세요.
2. Windows 검색 창에 **`Manage computer certificates`**를 입력합니다. 애플리케이션에서 컴퓨터를 변경하도록 허용할지 묻는 메시지가 표시되면 예를 선택합니다.
3. 인증서 창이 열리면 필요한 경우 인증서 목록을 볼 수 있도록 인증서 - 로컬 컴퓨터를 확장합니다. 신뢰할 수 있는 루트 인증 기관에 대한 컨텍스트(마우스 오른쪽 버튼 클릭) 메뉴를 열고 모든 작업, 가져오기를 선택합니다.
4. 다음을 선택한 다음 찾아보기를 선택하여 1단계에서 다운로드한 `*.pem` 파일을 찾습니다. 열기를 선택하여 인증서 파일을 선택하고 다음을 선택한 다음 완료를 선택합니다.

> **Note**
>
> `.pem`은 표준 인증서 확장자가 아니므로 파일을 찾으려면 브라우저 창에서 파일 유형을 모든 파일(*.*)로 변경합니다.

5. Microsoft 관리 콘솔에서 인증서를 확장합니다. 신뢰할 수 있는 루트 인증 기관을 확장한 다음 인증서를 선택하고 존재 여부를 확인할 인증서를 찾습니다. 인증서 이름은 Amazon RDS로 시작합니다.
6. 컴퓨터를 다시 시작합니다.

Java KeyStore로 인증서를 가져오려면

1. 다음 소스 중 하나에서 인증서를 다운로드합니다.

   인증서 다운로드에 자세한 내용은 Amazon RDS 사용 설명서에서 SSL/TLS를 사용하여 DB 인스턴스에 대한 연결 암호화를 참조하세요.
2. 인증서 번들을 다운로드한 경우에는 해당 인증서 번들을 개별 인증서 파일로 분할합니다. 이렇게 하려면 -----BEGIN CERTIFICATE-----로 시작하고 -----END CERTIFICATE-----로 끝나는 각 인증서 블록을 개별 *.pem 파일에 배치합니다. 각 인증서에 대한 개별 *.pem 파일을 생성한 후 인증서 번들 파일을 안전하게 제거할 수 있습니다.
3. 인증서를 다운로드한 디렉터리에서 명령 창 또는 터미널 세션을 열고 이전 단계에서 만든 모든 *.pem 파일에 대해 다음 명령을 실행합니다.

   ```
   keytool -importcert -file <filename>.pem -alias <filename>.pem -keystore storename
   ```

   Example

   다음 예제에서는 eu-west-1-bundle.pem 파일을 다운로드했다고 가정합니다.

   ```
   keytool -importcert -file eu-west-1-bundle.pem -alias eu-west-1-bundle.pem -
   keystore trust-2019.ks
   Picked up JAVA_TOOL_OPTIONS: -Dlog4j2.formatMsgNoLookups=true
   Enter keystore password:
   Re-enter new password:
   Owner: CN=Amazon RDS Root 2019 CA, OU=Amazon RDS, O="Amazon Web Services, Inc.",
    ST=Washington, L=Seattle, C=US
   Issuer: CN=Amazon RDS Root 2019 CA, OU=Amazon RDS, O="Amazon Web Services, Inc.",
    ST=Washington, L=Seattle, C=US
   Serial number: c73467369250ae75
   Valid from: Thu Aug 22 19:08:50 CEST 2019 until: Thu Aug 22 19:08:50 CEST 2024
   Certificate fingerprints:
            SHA1: D4:0D:DB:29:E3:75:0D:FF:A6:71:C3:14:0B:BF:5F:47:8D:1C:80:96
            SHA256:
    F2:54:C7:D5:E9:23:B5:B7:51:0C:D7:9E:F7:77:7C:1C:A7:E6:4A:3C:97:22:E4:0D:64:54:78:FC:70:AA:
   ```

```
Signature algorithm name: SHA256withRSA
Subject Public Key Algorithm: 2048-bit RSA key
Version: 3

Extensions:

#1: ObjectId: 2.5.29.35 Criticality=false
AuthorityKeyIdentifier [
KeyIdentifier [
0000: 73 5F 60 D8 BC CB 03 98   F4 2B 17 34 2E 36 5A A6  s_`......+.4.6Z.
0010: 60 FF BC 1F                                        `...
]
]

#2: ObjectId: 2.5.29.19 Criticality=true
BasicConstraints:[
  CA:true
  PathLen:2147483647
]

#3: ObjectId: 2.5.29.15 Criticality=true
KeyUsage [
  Key_CertSign
  Crl_Sign
]

#4: ObjectId: 2.5.29.14 Criticality=false
SubjectKeyIdentifier [
KeyIdentifier [
0000: 73 5F 60 D8 BC CB 03 98   F4 2B 17 34 2E 36 5A A6  s_`......+.4.6Z.
0010: 60 FF BC 1F                                        `...
]
]

Trust this certificate? [no]:  yes
Certificate was added to keystore
```

4. 키스토어를 AWS SCT에 트러스트 스토어로 추가합니다. 이렇게 하려면 기본 메뉴에서 설정, 전역 설정, 보안, 트러스트 스토어를 선택한 다음 Select existing trust store를 선택합니다.

   트러스트 스토어를 추가한 후 데이터베이스에 대한 AWS SCT 연결을 생성할 때 이 트러스트 스토어를 사용하여 SSL 지원 연결을 구성할 수 있습니다. AWS SCT 데이터베이스에 연결 대화 상자에서 Use SSL을 선택하고 이전에 입력한 트러스트 스토어를 선택합니다.

Apache Cassandra를 AWS SCT에 대한 소스로 사용

AWS SCT를 사용하여 키스페이스를 Apache Cassandra에서 Amazon DynamoDB로 변환할 수 있습니다.

Apache Cassandra에 소스로 연결

다음 절차에 따라 AWS Schema Conversion Tool을 사용하여 Apache Cassandra 소스 데이터베이스에 연결합니다.

Apache Cassandra 소스 데이터베이스에 연결하려면

1. AWS Schema Conversion Tool에서 소스 추가를 선택합니다.
2. Cassandra를 선택하고 다음을 선택합니다.

   소스 추가 대화 상자가 나타납니다.
3. 연결 이름에 데이터베이스의 이름을 입력합니다. AWS SCT는 왼쪽 패널의 트리에 이 이름을 표시합니다.
4. AWS Secrets Manager의 데이터베이스 보안 인증 정보를 사용하거나 수동으로 입력합니다.

   - Secrets Manager의 데이터베이스 보안 인증 정보를 사용하려면 다음 지침을 따릅니다.
     1. AWS Secret에서 보안 암호의 이름을 선택합니다.
     2. Populate를 선택하여 Secrets Manager에서 데이터베이스 연결 대화 상자에 있는 모든 값을 자동으로 채웁니다.

     Secrets Manager의 데이터베이스 보안 인증 사용에 대한 자세한 내용은 AWS Secrets Manager 사용 섹션을 참조하세요.
   - Apache Cassandra 소스 데이터베이스 연결 정보를 수동으로 입력하려면 다음 지침을 사용합니다.

파라미터	작업
[Server name]	소스 데이터베이스 서버의 DNS(Domain Name Service) 이름 또는 IP 주소를 입력합니다.
[Server port]	소스 데이터베이스 서버에 연결하는 데 사용되는 포트를 입력합니다.

파라미터	작업
User name 및 Password	소스 데이터베이스 서버에 연결하는 데 필요한 데이터베이스 보안 인증 정보를 입력합니다. 프로젝트에서 데이터베이스에 연결하도록 선택한 경우에만 AWS SCT에서 암호를 사용하여 소스 데이터베이스에 연결합니다. 소스 데이터베이스의 암호가 노출될 위험을 방지하기 위해 AWS SCT는 기본적으로 암호를 저장하지 않습니다. AWS SCT 프로젝트를 닫았다 다시 열 경우 필요에 따라 소스 데이터베이스에 연결하기 위한 암호를 입력하라는 메시지가 표시됩니다.
Use SSL(SSL 사용)	SSL(Secure Sockets Layer)을 사용하여 데이터베이스에 연결하려면 이 옵션을 선택합니다. SSL 탭에서 다음 추가 정보를 적절히 제공합니다. • 트러스트 스토어: 사용할 트러스트 스토어입니다. • 키 스토어: 사용할 키 스토어입니다.
Store password	AWS SCT는 SSL 인증서와 데이터베이스 암호를 저장할 안전한 볼트를 생성합니다. 이 옵션을 켜면 데이터베이스 암호를 저장하고 암호 입력 없이 빠르게 데이터베이스에 연결할 수 있습니다.

5. Test Connection을 선택하여 AWS SCT가 소스 데이터베이스에 연결할 수 있는지 확인합니다.
6. 연결을 선택하여 소스 데이터베이스에 연결합니다.

Apache Hadoop을 AWS SCT에 대한 소스로 사용

AWS SCT 명령줄 인터페이스(CLI)를 사용하여 Apache Hadoop에서 Amazon EMR로 마이그레이션할 수 있습니다. AWS SCT는 마이그레이션 중에 Amazon S3 버킷을 데이터의 임시 스토리지로 사용합니다.

AWS SCT는 소스 Apache Hadoop 버전 2.2.0 이상을 지원합니다. 또한 AWS SCT는 Apache Hive 버전 0.13.0 이상도 지원합니다.

AWS SCT는 대상 Amazon EMR 버전 6.3.0 이상을 지원합니다. 또한 AWS SCT는 대상 Apache Hadoop 버전 2.6.0 이상과 Apache Hive 버전 0.13.0 이상도 지원합니다.

주제

- Apache Hadoop을 소스로 사용하기 위한 사전 요구 사항
- Hive를 소스로 사용하기 위한 권한
- HDFS를 소스로 사용하기 위한 권한
- HDFS를 대상으로 사용하기 위한 권한
- Apache Hadoop에 소스로 연결
- 소스 Hive 및 HDFS 서비스에 연결
- Amazon EMR에 대상으로 연결

Apache Hadoop을 소스로 사용하기 위한 사전 요구 사항

AWS SCT CLI를 사용하여 Apache Hadoop에 연결하려면 다음과 같은 사전 요구 사항이 필요합니다.

- 마이그레이션 중에 데이터를 저장할 Amazon S3 버킷을 생성합니다. 그런 다음 데이터를 Amazon EMR HDFS로 복사하거나 Amazon S3을 Hadoop 워크로드용 데이터 리포지토리로 사용할 수 있습니다. 자세한 내용은 Amazon S3 사용 설명서의 버킷 생성을 참조하세요.
- `AmazonS3FullAccess` 정책을 사용하여 AWS Identity and Access Management(IAM) 역할을 생성합니다. AWS SCT는 이 IAM 역할을 사용하여 Amazon S3 버킷에 액세스합니다.
- AWS 비밀 키와 AWS 비밀 액세스 키를 기록해 둡니다. AWS 액세스 키에 대한 자세한 내용은 IAM 사용 설명서에서 액세스 키 관리를 참조하세요.
- 대상 Amazon EMR 클러스터를 생성 및 구성합니다. 자세한 내용은 Amazon EMR 관리 안내서에서 Amazon EMR 시작하기를 참조하세요.
- 소스 Apache Hadoop 클러스터에 `distcp` 유틸리티를 설치합니다. 또한 대상 Amazon EMR 클러스터에도 `s3-dist-cp` 유틸리티를 설치합니다. 데이터베이스 사용자에게 이러한 유틸리티를 실행할 권한이 있어야 합니다.
- s3a 프로토콜을 사용하도록 소스 Hadoop 클러스터의 `core-site.xml` 파일을 구성합니다. 이 작업을 수행하려면 `fs.s3a.aws.credentials.provider` 파라미터를 다음 값 중 하나로 설정합니다.
  - `org.apache.hadoop.fs.s3a.TemporaryAWSCredentialsProvider`
  - `org.apache.hadoop.fs.s3a.SimpleAWSCredentialsProvider`
  - `org.apache.hadoop.fs.s3a.AnonymousAWSCredentialsProvider`
  - `org.apache.hadoop.fs.s3a.auth.AssumedRoleCredentialProvider`

  `core-site.xml` 파일에 다음 코드 예제를 추가할 수 있습니다.

```
<property>
  <name>fs.s3a.aws.credentials.provider</name>
  <value>org.apache.hadoop.fs.s3a.SimpleAWSCredentialsProvider</value>
</property>
```

이전 예제에서는 이전 옵션 목록의 네 가지 옵션 중 하나를 보여줍니다. `core-site.xml` 파일에 `fs.s3a.aws.credentials.provider` 파라미터를 설정하지 않은 경우 AWS SCT에서 공급자를 자동으로 선택합니다.

Hive를 소스로 사용하기 위한 권한

Hive 소스 사용자에게 필요한 권한은 다음과 같습니다.

- 소스 데이터 폴더 및 소스 Amazon S3 버킷에 대한 `READ` 액세스 권한
- 중간 및 대상 Amazon S3 버킷에 대한 `READ+WRITE` 액세스 권한

마이그레이션 속도를 높이려면 ACID 트랜잭션 소스 테이블에 대해 압축을 실행하는 것이 좋습니다.

Amazon EMR Hive 대상 사용자에게 필요한 권한은 다음과 같습니다.

- 대상 Amazon S3 버킷에 대한 `READ` 액세스 권한
- 중간 Amazon S3 버킷에 대한 `READ+WRITE` 액세스 권한
- 대상 HDFS 폴더에 대한 `READ+WRITE` 액세스 권한

HDFS를 소스로 사용하기 위한 권한

HDFS를 소스로 사용하는 데 필요한 권한은 다음과 같습니다.

- NameNode의 경우, `EXECUTE`
- 마이그레이션 프로젝트에 포함할 모든 소스 폴더 및 파일의 경우, `EXECUTE+READ`
- Amazon S3로 마이그레이션하기 전에 Spark 작업을 실행하고 파일을 저장하기 위한 NameNode 내 `tmp` 디렉터리의 경우, `READ+WRITE`

HDFS에서는 모든 작업에 순회 액세스가 필요합니다. 순회 액세스에는 최종 경로 구성 요소를 제외한 경로의 모든 기존 구성 요소에 대한 EXECUTE 권한이 필요합니다. 예를 들어 /foo/bar/baz에 액세스하는 모든 작업의 경우 /, /foo 및 /foo/bar에 대한 EXECUTE 권한이 사용자에게 있어야 합니다.

다음 코드 예제는 소스 폴더 및 파일에 대한 EXECUTE+READ 권한과 tmp 디렉터리에 대한 READ +WRITE 권한을 부여하는 방법을 보여줍니다.

```
hadoop fs -chmod -R 744 /user/hdfs-data
hadoop fs -chmod -R 766 /tmp
```

HDFS를 대상으로 사용하기 위한 권한

Amazon EMR HDFS를 대상으로 사용하는 데 필요한 권한은 다음과 같습니다.

- 대상 Amazon EMR 클러스터의 NameNode인 경우, EXECUTE
- 마이그레이션 후 데이터를 저장할 대상 HDFS 폴더의 경우, READ+WRITE

Apache Hadoop에 소스로 연결

AWS SCT 버전 1.0.670 이상에서 Apache Hadoop을 소스로 사용할 수 있습니다. AWS SCT 명령줄 인터페이스(CLI)에서만 Hadoop 클러스터를 Amazon EMR로 마이그레이션할 수 있습니다. 시작하기 전에 AWS SCT의 명령줄 인터페이스를 숙지해야 합니다. 자세한 내용은 AWS SCT CLI 레퍼런스 섹션을 참조하세요.

AWS SCT CLI에서 Apache Hadoop에 연결하려면

1. 새 AWS SCT CLI 스크립트를 생성하거나 기존 시나리오 템플릿을 편집합니다. 예를 들어 HadoopMigrationTemplate.scts 템플릿을 다운로드하여 편집할 수 있습니다. 자세한 내용은 CLI 시나리오 가져오기 섹션을 참조하세요.
2. 드라이버 위치 및 로그 폴더와 같은 AWS SCT 애플리케이션 설정을 구성합니다.

   필요한 JDBC 드라이버를 다운로드하고 파일을 저장할 위치를 지정합니다. 자세한 내용은 필수 데이터베이스 드라이버 다운로드 섹션을 참조하세요.

   다음 코드 예제는 Apache Hive 드라이버에 경로를 추가하는 방법을 보여줍니다. 이 코드 예제를 실행하면 AWS SCT가 c:\sct 폴더에 로그 파일을 저장합니다.

   ```
   SetGlobalSettings
   ```

```
    -save: 'true'
    -settings: '{
        "hive_driver_file": "c:\\sct\\HiveJDBC42.jar",
        "log_folder": "c:\\sct",
        "console_log_folder": "c:\\sct"
    }'
/
```

Windows에서 이 예제와 다음 예제를 사용할 수 있습니다.

3. 새 AWS SCT 프로젝트를 생성합니다.

   다음 코드 예제는 `c:\sct` 폴더에 `hadoop_emr` 프로젝트를 만듭니다.

```
CreateProject
    -name: 'hadoop_emr'
    -directory: 'c:\sct'
/
```

4. 프로젝트에 소스 Hadoop 클러스터를 추가합니다.

   `AddSourceCluster` 명령을 사용하여 소스 Hadoop 클러스터에 연결합니다. `name`, `host`, `port`, `user`와 같은 필수 파라미터에 대한 값을 제공해야 합니다. 다른 파라미터는 선택 사항입니다.

   다음 코드 예제는 소스 Hadoop 클러스터를 추가합니다. 이 예제에서는 `HADOOP_SOURCE`를 소스 클러스터의 이름으로 설정합니다. 이 객체 이름을 사용하여 Hive 및 HDFS 서비스를 프로젝트에 추가하고 매핑 규칙을 생성합니다.

```
AddSourceCluster
    -name: 'HADOOP_SOURCE'
    -vendor: 'HADOOP'
    -host: 'hadoop_address'
    -port: '22'
    -user: 'hadoop_user'
    -password: 'hadoop_password'
    -useSSL: 'true'
    -privateKeyPath: 'c:\path\name.pem'
    -passPhrase: 'hadoop_passphrase'
/
```

   위 예제에서 *hadoop_address*를 Hadoop 클러스터의 IP 주소로 바꿉니다. 필요한 경우 port 옵션 값을 구성합니다. 그런 다음 *hadoop_user* 및 *hadoop_password*를 Hadoop 사용자의 이름 및

이 사용자의 암호로 바꿉니다. *path\name*에는 소스 Hadoop 클러스터의 PEM 파일 이름과 경로를 입력합니다.

5. CLI 스크립트를 저장합니다. 다음으로 Hive 및 HDFS 서비스에 대한 연결 정보를 추가합니다.

소스 Hive 및 HDFS 서비스에 연결

AWS SCT CLI를 사용하여 소스 Hive 및 HDFS 서비스에 연결할 수 있습니다. Apache Hive에 연결하려면 Hive JDBC 드라이버 버전 2.3.4 이상을 사용합니다. 자세한 내용은 필수 데이터베이스 드라이버 다운로드 섹션을 참조하세요.

AWS SCT는 hadoop 클러스터 사용자를 통해 Apache Hive에 연결합니다. 이렇게 하려면 `AddSourceClusterHive` 및 `AddSourceClusterHDFS` 명령을 사용합니다. 다음 방법 중 하나를 사용할 수 있습니다.

- 새 SSH 터널을 생성합니다.

  `createTunnel`에 **`true`**을 입력합니다. `host`에는 소스 Hive 또는 HDFS 서비스의 내부 IP 주소를 입력합니다. `port`에는 Hive 또는 HDFS 서비스의 서비스 포트를 입력합니다.

  그런 다음 `user` 및 `password`에 대한 Hive 또는 HDFS 보안 인증 정보를 입력합니다. SSH 터널에 대한 자세한 내용은 Amazon EMR 관리 안내서에서 로컬 포트 전달을 사용하여 프라이머리 노드에 SSH 터널 설정을 참조하세요.
- 기존 SSH 터널을 사용합니다.

  `host`에 **`localhost`**을 입력합니다. `port`에는 SSH 터널 파라미터에서 로컬 포트를 입력합니다.
- Hive 및 HDFS 서비스에 직접 연결합니다.

  `host`에는 소스 Hive 또는 HDFS 서비스의 IP 주소 또는 호스트 이름을 입력합니다. `port`에는 Hive 또는 HDFS 서비스의 서비스 포트를 입력합니다. 그런 다음 `user` 및 `password`에 대한 Hive 또는 HDFS 보안 인증 정보를 입력합니다.

AWS SCT CLI에서 Hive와 HDFS에 연결하려면

1. 소스 Hadoop 클러스터의 연결 정보가 포함된 CLI 스크립트를 엽니다. 이전 단계에서 정의한 Hadoop 클러스터의 이름을 사용해야 합니다.
2. 프로젝트에 소스 Hive 서비스를 추가합니다.

AddSourceClusterHive 명령을 사용하여 소스 Hive 서비스를 연결합니다. user, password, cluster, name, port와 같은 필수 파라미터에 대한 값을 제공해야 합니다. 다른 파라미터는 선택 사항입니다.

다음 코드 예제는 AWS SCT가 Hive 서비스에서 작업하는 데 사용할 터널을 생성합니다. 이 소스 Hive 서비스는 AWS SCT와 동일한 PC에서 실행됩니다. 이 예제에서는 이전 예제의 HADOOP_SOURCE 소스 클러스터를 사용합니다.

```
AddSourceClusterHive
    -cluster: 'HADOOP_SOURCE'
    -name: 'HIVE_SOURCE'
    -host: 'localhost'
    -port: '10005'
    -user: 'hive_user'
    -password: 'hive_password'
    -createTunnel: 'true'
    -localPort: '10005'
    -remoteHost: 'hive_remote_address'
    -remotePort: 'hive_port'
/
```

다음 코드 예제에서는 터널 없이 Hive 서비스에 연결합니다.

```
AddSourceClusterHive
    -cluster: 'HADOOP_SOURCE'
    -name: 'HIVE_SOURCE'
    -host: 'hive_address'
    -port: 'hive_port'
    -user: 'hive_user'
    -password: 'hive_password'
/
```

이전 예제에서 *hive_user* 및 *hive_password*를 Hive 사용자 이름과 이 사용자의 암호로 바꿉니다.

다음으로, *hive_address* 및 *hive_port*를 소스 Hadoop 클러스터의 NameNode IP 주소와 포트로 바꿉니다.

*hive_remote_address*에는 소스 Hive 서비스의 기본값 127.0.0.1 또는 NameNode IP 주소를 사용할 수 있습니다.

3. 프로젝트에 소스 HDFS 서비스를 추가합니다.

   AddSourceClusterHDFS 명령을 사용하여 소스 HDFS 서비스를 연결합니다. user, password, cluster, name, port와 같은 필수 파라미터에 대한 값을 제공해야 합니다. 다른 파라미터는 선택 사항입니다.

   사용자가 소스 HDFS 서비스에서 데이터를 마이그레이션하는 데 필요한 권한을 가지고 있어야 합니다. 자세한 내용은 Hive를 소스로 사용하기 위한 권한 섹션을 참조하세요.

   다음 코드 예제는 AWS SCT가 Apache HDFS 서비스에서 작업하는 데 사용할 터널을 생성합니다. 이 예제는 이전에 만든 HADOOP_SOURCE 소스 클러스터를 사용합니다.

```
AddSourceClusterHDFS
    -cluster: 'HADOOP_SOURCE'
    -name: 'HDFS_SOURCE'
    -host: 'localhost'
    -port: '9005'
    -user: 'hdfs_user'
    -password: 'hdfs_password'
    -createTunnel: 'true'
    -localPort: '9005'
    -remoteHost: 'hdfs_remote_address'
    -remotePort: 'hdfs_port'
/
```

   다음 코드에서는 터널 없이 Apache HDFS 서비스에 연결합니다.

```
AddSourceClusterHDFS
    -cluster: 'HADOOP_SOURCE'
    -name: 'HDFS_SOURCE'
    -host: 'hdfs_address'
    -port: 'hdfs_port'
    -user: 'hdfs_user'
    -password: 'hdfs_password'
/
```

   이전 예제에서 *hdfs_user* 및 *hdfs_password*를 HDFS 사용자 이름과 이 사용자의 암호로 바꿉니다.

   다음으로, *hdfs_address* 및 *hdfs_port*를 소스 Hadoop 클러스터의 NameNode IP 주소와 포트로 바꿉니다.

*hdfs_remote_address*에는 소스 Hive 서비스의 기본값 `127.0.0.1` 또는 NameNode IP 주소를 사용할 수 있습니다.

4. CLI 스크립트를 저장합니다. 다음으로, 대상 Amazon EMR 클러스터의 연결 정보와 마이그레이션 명령을 추가합니다.

Amazon EMR에 대상으로 연결

AWS SCT CLI를 사용하여 대상 Amazon EMR 클러스터에 연결할 수 있습니다. 이 작업을 수행하려면 인바운드 트래픽을 승인하고 SSH를 사용해야 합니다. 이 경우 AWS SCT에는 Amazon EMR 클러스터를 사용하는 데 필요한 모든 권한이 있어야 합니다. 자세한 내용은 Amazon EMR 관리 안내서에서 연결하기 전에 및 SSH를 사용하여 프라이머리 노드에 연결을 참조하세요.

AWS SCT는 hadoop 클러스터 사용자를 통해 Amazon EMR Hive에 연결합니다. Amazon EMR Hive에 연결하려면 Hive JDBC 드라이버 버전 2.6.2.1002 이상을 사용합니다. 자세한 내용은 필수 데이터베이스 드라이버 다운로드 섹션을 참조하세요.

AWS SCT CLI에서 Amazon EMR에 연결하려면

1. 소스 Hadoop 클러스터의 연결 정보가 포함된 CLI 스크립트를 엽니다. 대상 Amazon EMR 보안 인증 정보를 이 파일에 추가합니다.
2. 대상 Amazon EMR 클러스터를 프로젝트에 추가합니다.

   다음 코드 예제는 대상 Amazon EMR 클러스터를 추가합니다. 이 예제에서는 `HADOOP_TARGET`을 대상 클러스터의 이름으로 설정합니다. 이 객체 이름을 사용하여 Hive 및 HDFS 서비스와 Amazon S3 버킷 폴더를 프로젝트에 추가하고 매핑 규칙을 생성합니다.

```
AddTargetCluster
 -name: 'HADOOP_TARGET'
 -vendor: 'AMAZON_EMR'
 -host: 'ec2-44-44-55-66.eu-west-1.EXAMPLE.amazonaws.com'
 -port: '22'
 -user: 'emr_user'
 -password: 'emr_password'
 -useSSL: 'true'
 -privateKeyPath: 'c:\path\name.pem'
 -passPhrase: '1234567890abcdef0!'
 -s3Name: 'S3_TARGET'
 -accessKey: 'AKIAIOSFODNN7EXAMPLE'
 -secretKey: 'wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY'
```

```
 -region: 'eu-west-1'
 -s3Path: 'doc-example-bucket/example-folder'
/
```

이전 예제에서 AWS 리소스 이름과 Amazon EMR 연결 정보를 입력합니다. 여기에는 Amazon EMR 클러스터의 IP 주소, AWS 액세스 키, AWS 비밀 액세스 키 및 Amazon S3 버킷이 포함됩니다. 필요한 경우 port 변수 값을 구성합니다. 그런 다음 *emr_user* 및 *emr_password*를 Amazon EMR 사용자의 이름 및 이 사용자의 암호로 바꿉니다. *path\name*에는 대상 Amazon EMR 클러스터의 PEM 파일 이름과 경로를 입력합니다. 자세한 내용은 EMR 클러스터 액세스를 위한 PEM 파일 다운로드를 참조하세요.

3. 대상 Amazon S3 버킷을 프로젝트에 추가합니다.

   다음 코드 예제는 대상 Amazon S3 버킷을 추가합니다. 이 예제는 이전에 만든 HADOOP_TARGET 클러스터를 사용합니다.

```
AddTargetClusterS3
 -cluster: 'HADOOP_TARGET'
 -Name: 'S3_TARGET'
 -accessKey: 'AKIAIOSFODNN7EXAMPLE'
 -secretKey: 'wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY'
 -region: 'eu-west-1'
 -s3Path: 'doc-example-bucket/example-folder'
/
```

   이전 예제에서 AWS 액세스 키, AWS 비밀 액세스 키, Amazon S3 버킷을 입력합니다.

4. 프로젝트에 대상 Hive 서비스를 추가합니다.

   다음 코드 예제는 AWS SCT가 대상 Hive 서비스에서 작업하는 데 사용할 터널을 생성합니다. 이 예제는 이전에 만든 HADOOP_TARGET 대상 클러스터를 사용합니다.

```
AddTargetClusterHive
    -cluster: 'HADOOP_TARGET'
    -name: 'HIVE_TARGET'
    -host: 'localhost'
    -port: '10006'
    -user: 'hive_user'
    -password: 'hive_password'
    -createTunnel: 'true'
    -localPort: '10006'
    -remoteHost: 'hive_address'
```

```
    -remotePort: 'hive_port'
/
```

이전 예제에서 *hive_user* 및 *hive_password*를 Hive 사용자 이름과 이 사용자의 암호로 바꿉니다.

그 다음으로, *hive_address*를 기본값 127.0.0.1 또는 대상 Hive 서비스의 NameNode IP 주소로 바꿉니다. 그 다음으로, *hive_port*를 대상 Hive 서비스의 포트로 바꿉니다.

5. 프로젝트에 대상 HDFS 서비스를 추가합니다.

   다음 코드 예제는 AWS SCT가 Apache HDFS 서비스에서 작업하는 데 사용할 터널을 생성합니다. 이 예제는 이전에 만든 HADOOP_TARGET 대상 클러스터를 사용합니다.

```
AddTargetClusterHDFS
    -cluster: 'HADOOP_TARGET'
    -name: 'HDFS_TARGET'
    -host: 'localhost'
    -port: '8025'
    -user: 'hdfs_user'
    -password: 'hdfs_password'
    -createTunnel: 'true'
    -localPort: '8025'
    -remoteHost: 'hdfs_address'
    -remotePort: 'hdfs_port'
/
```

   이전 예제에서 *hdfs_user* 및 *hdfs_password*를 HDFS 사용자 이름과 이 사용자의 암호로 바꿉니다.

   그 다음으로, *hdfs_address* 및 *hdfs_port*를 대상 HDFS 서비스의 NameNode의 프라이빗 IP 주소와 포트로 바꿉니다.

6. CLI 스크립트를 저장합니다. 그 다음으로, 매핑 규칙과 마이그레이션 명령을 추가합니다. 자세한 내용은 Apache Hadoop을 Amazon EMR로 마이그레이션 섹션을 참조하세요.

Apache Oozie를 AWS SCT에 대한 소스로 사용

AWS SCT 명령줄 인터페이스(CLI)를 사용하여 Apache Oozie 워크플로를 AWS Step Functions로 변환할 수 있습니다. Apache Hadoop 워크로드를 Amazon EMR로 마이그레이션한 후 AWS 클라우드의

네이티브 서비스를 사용하여 작업을 오케스트레이션할 수 있습니다. 자세한 내용은 Apache Hadoop을 소스로 사용 섹션을 참조하세요.

AWS SCT에서는 Oozie 워크플로를 AWS Step Functions로 변환한 후 AWS Lambda를 사용하여 AWS Step Functions에서 지원하지 않는 기능을 에뮬레이션합니다. 또한 AWS SCT는 Oozie 작업 속성을 AWS Systems Manager로 변환합니다.

Apache Oozie 워크플로를 변환하려면 AWS SCT 버전 1.0.671 이상을 사용해야 합니다. 또한 AWS SCT의 명령줄 인터페이스를 숙지해야 합니다. 자세한 내용은 AWS SCT CLI 레퍼런스 섹션을 참조하세요.

Apache Oozie를 소스로 사용하기 위한 사전 조건

AWS SCT CLI를 사용하여 Apache Oozie에 연결하려면 다음과 같은 사전 조건이 필요합니다.

- 상태 시스템 정의를 저장할 Amazon S3 버킷을 생성합니다. 이러한 정의를 사용하여 상태 시스템을 구성할 수 있습니다. 자세한 내용은 Amazon S3 사용 설명서의 버킷 생성을 참조하세요.
- `AmazonS3FullAccess` 정책을 사용하여 AWS Identity and Access Management(IAM) 역할을 생성합니다. AWS SCT는 이 IAM 역할을 사용하여 Amazon S3 버킷에 액세스합니다.
- AWS 비밀 키와 AWS 비밀 액세스 키를 기록해 둡니다. AWS 액세스 키에 대한 자세한 내용은 IAM 사용 설명서에서 액세스 키 관리를 참조하세요.
- AWS 보안 인증 정보와 Amazon S3 버킷 관련 정보를 전역 애플리케이션 설정의 AWS 서비스 프로필에 저장합니다. 그런 다음 AWS SCT는 이 AWS 서비스 프로필을 사용하여 AWS 리소스로 작업합니다. 자세한 내용은 AWS SCT에 AWS 서비스 프로필 저장 섹션을 참조하세요.

소스 Apache Oozie 워크플로로 작업하려면 AWS SCT에 소스 파일의 특정 구조가 필요합니다. 각 애플리케이션 폴더에는 `job.properties` 파일이 포함되어야 합니다. 이 파일에는 작업 속성의 키-값 페어가 포함되어 있습니다. 또한 각 애플리케이션 폴더에는 `workflow.xml` 파일이 포함되어야 합니다. 이 파일은 워크플로의 작업 노드와 제어 흐름 노드를 설명합니다.

Apache Oozie에 소스로 연결

다음 절차에 따라 Apache Oozie 소스 파일에 연결합니다.

AWS SCT CLI에서 Apache Oozie에 연결하려면

1. 새 AWS SCT CLI 스크립트를 생성하거나 기존 시나리오 템플릿을 편집합니다. 예를 들어 `OozieConversionTemplate.scts` 템플릿을 다운로드하여 편집할 수 있습니다. 자세한 내용은 CLI 시나리오 가져오기 섹션을 참조하세요.

2. AWS SCT 애플리케이션 설정을 구성합니다.

   다음 코드 예제에서는 애플리케이션 설정을 저장하고 프로젝트에 암호를 저장할 수 있도록 합니다. 이렇게 저장된 설정은 다른 프로젝트에서 사용할 수 있습니다.

```
SetGlobalSettings
    -save: 'true'
    -settings: '{
        "store_password": "true"
    }'
/
```

3. 새 AWS SCT 프로젝트를 생성합니다.

   다음 코드 예제는 `c:\sct` 폴더에 `oozie` 프로젝트를 만듭니다.

```
CreateProject
    -name: 'oozie'
    -directory: 'c:\sct'
/
```

4. `AddSource` 명령을 사용하여 소스 Apache Oozie 파일이 있는 폴더를 프로젝트에 추가합니다. `vendor` 파라미터에 대해 `APACHE_OOZIE` 값을 사용해야 합니다. 또한 필수 파라미터 `name` 및 `mappingsFolder`의 값을 제공해야 합니다.

   다음 코드 예제는 Apache Oozie를 AWS SCT 프로젝트에 소스로 추가합니다. 이 예제에서는 이름이 `OOZIE`인 소스 객체를 만듭니다. 이 객체 이름을 사용하여 매핑 규칙을 추가합니다. 이 코드 예제를 실행하면 AWS SCT가 `c:\oozie` 폴더를 사용하여 프로젝트에 소스 파일을 로드합니다.

```
AddSource
    -name: 'OOZIE'
    -vendor: 'APACHE_OOZIE'
    -mappingsFolder: 'c:\oozie'
/
```

   Windows에서 이 예제와 다음 예제를 사용할 수 있습니다.

5. `ConnectSource` 명령을 사용하여 소스 Apache Oozie 파일에 연결합니다. 이전 단계에서 정의한 소스 객체의 이름을 사용합니다.

```
ConnectSource
    -name: 'OOZIE'
```

```
    -mappingsFolder: 'c:\oozie'
/
```

6. CLI 스크립트를 저장합니다. 그 다음, AWS Step Functions 서비스의 연결 정보를 추가합니다.

확장 팩의 AWS Lambda 함수를 사용하기 위한 권한

AWS Step Functions에서 지원하지 않는 소스 함수의 경우 AWS SCT가 확장 팩을 생성합니다. 이 확장 팩에는 소스 함수를 에뮬레이션하는 AWS Lambda 함수가 포함되어 있습니다.

이 확장 팩을 사용하려면 다음 권한을 가진 AWS Identity and Access Management(IAM) 역할을 생성합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "lambda",
            "Effect": "Allow",
            "Action": [
                "lambda:InvokeFunction"
            ],
            "Resource": [
                "arn:aws:lambda:*:498160209112:function:LoadParameterInitialState:*",
                "arn:aws:lambda:*:498160209112:function:EvaluateJSPELExpressions:*"
            ]
        },
        {
            "Sid": "emr",
            "Effect": "Allow",
            "Action": [
                "elasticmapreduce:DescribeStep",
                "elasticmapreduce:AddJobFlowSteps"
            ],
            "Resource": [
                "arn:aws:elasticmapreduce:*:498160209112:cluster/*"
            ]
        },
        {
            "Sid": "s3",
            "Effect": "Allow",
            "Action": [
```

```
                "s3:GetObject"
            ],
            "Resource": [
                "arn:aws:s3:::*/*"
            ]
        }
    ]
}
```

확장 팩을 적용하려면 AWS SCT에 다음 권한을 가진 IAM 역할이 필요합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iam:GetRole",
                "iam:ListRolePolicies",
                "iam:CreateRole",
                "iam:TagRole",
                "iam:PutRolePolicy",
                "iam:DeleteRolePolicy",
                "iam:DeleteRole",
                "iam:PassRole"
            ],
            "Resource": [
                "arn:aws:iam::ACCOUNT_NUMBER:role/sct/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iam:GetRole",
                "iam:ListRolePolicies"
            ],
            "Resource": [
                "arn:aws:iam::ACCOUNT_NUMBER:role/
lambda_LoadParameterInitialStateRole",
                "arn:aws:iam::ACCOUNT_NUMBER:role/lambda_EvaluateJSPELExpressionsRole",
                "arn:aws:iam::ACCOUNT_NUMBER:role/
stepFunctions_MigratedOozieWorkflowRole"
            ]
```

```
        },
        {
            "Effect": "Allow",
            "Action": [
                "lambda:GetFunction",
                "lambda:CreateFunction",
                "lambda:UpdateFunctionCode",
                "lambda:DeleteFunction"
            ],
            "Resource": [
                "arn:aws:lambda:*:ACCOUNT_NUMBER:function:LoadParameterInitialState",
                "arn:aws:lambda:*:ACCOUNT_NUMBER:function:EvaluateJSPELExpressions"
            ]
        }
    ]
}
```

AWS Step Functions에 대상으로 연결

다음 절차에 따라 AWS Step Functions에 대상으로 연결합니다.

AWS SCT CLI에서 AWS Step Functions에 연결하려면

1. Apache Oozie 소스 파일의 연결 정보가 포함된 CLI 스크립트를 엽니다.
2. `AddTarget` 명령을 사용하여 AWS SCT 프로젝트에 마이그레이션 대상에 대한 정보를 추가합니다. `vendor` 파라미터에 대해 `STEP_FUNCTIONS` 값을 사용해야 합니다. 또한 필수 파라미터 `name` 및 `profile`의 값을 제공해야 합니다.

   다음 코드 예제는 AWS Step Functions를 AWS SCT 프로젝트에 소스로 추가합니다. 이 예제에서는 이름이 `AWS_STEP_FUNCTIONS`인 대상 객체를 만듭니다. 매핑 규칙을 생성할 때 이 객체 이름을 사용합니다. 또한 이 예제에서는 사전 조건 단계에서 생성한 AWS SCT 서비스 프로필을 사용합니다. *`profile_name`*을 프로필 이름으로 바꿔야 합니다.

   ```
   AddTarget
       -name: 'AWS_STEP_FUNCTIONS'
       -vendor: 'STEP_FUNCTIONS'
       -profile: 'profile_name'
   /
   ```

AWS 서비스 프로필을 사용하지 않는 경우, 필수 파라미터 `accessKey`, `secretKey`, `awsRegion`, `s3Path`의 값을 제공해야 합니다. 이러한 파라미터를 사용하여 AWS 비밀 액세스 키, AWS 비밀 키, AWS 리전 및 Amazon S3 버킷의 경로를 지정합니다.

3. `ConnectTarget` 명령을 사용하여 AWS Step Functions에 연결합니다. 이전 단계에서 정의한 대상 객체의 이름을 사용합니다.

   다음 코드 예제는 AWS 서비스 프로필을 사용하여 `AWS_STEP_FUNCTIONS` 대상 객체에 연결합니다. *`profile_name`*을 프로필 이름으로 바꿔야 합니다.

   ```
   ConnectTarget
       -name: 'AWS_STEP_FUNCTIONS'
       -profile: 'profile_name'
   /
   ```

4. CLI 스크립트를 저장합니다. 그 다음으로, 매핑 규칙과 마이그레이션 명령을 추가합니다. 자세한 내용은 Apache Oozie를 AWS Step Functions로 변환 섹션을 참조하세요.

Azure SQL Database를 AWS SCT에 대한 소스로 사용

AWS SCT를 사용하여 스키마, 코드 객체 및 애플리케이션 코드를 Azure SQL Database에서 다음 대상으로 변환할 수 있습니다.

- Amazon RDS for MySQL
- Amazon Aurora MySQL-Compatible Edition
- Amazon RDS for PostgreSQL
- Amazon Aurora PostgreSQL-Compatible Edition

주제

- Azure SQL Database를 소스로 사용하기 위한 권한
- Azure SQL Database에 소스로 연결

Azure SQL Database를 소스로 사용하기 위한 권한

Azure SQL Database를 소스로 사용하기 위해 필요한 권한은 다음과 같습니다.

- VIEW DEFINITION

- VIEW DATABASE STATE

변환하려는 스키마의 각 데이터베이스에 대해 권한 부여를 반복합니다.

대상 MySQL 및 PostgreSQL 데이터베이스에 필요한 권한은 다음 섹션에 설명되어 있습니다.

- MySQL을 대상 데이터베이스로 사용하기 위한 권한
- PostgreSQL을 대상 데이터베이스로 사용하기 위한 권한

Azure SQL Database에 소스로 연결

다음 절차에 따라 AWS Schema Conversion Tool을 사용하여 Azure SQL Database 소스 데이터베이스에 연결합니다.

Azure SQL Database 소스 데이터베이스에 연결하려면

1. AWS Schema Conversion Tool에서 소스 추가를 선택합니다.
2. Azure SQL Database를 선택하고 다음을 선택합니다.

   소스 추가 대화 상자가 나타납니다.
3. 연결 이름에 데이터베이스의 이름을 입력합니다. AWS SCT는 왼쪽 패널의 트리에 이 이름을 표시합니다.
4. AWS Secrets Manager의 데이터베이스 보안 인증 정보를 사용하거나 수동으로 입력합니다.

   - Secrets Manager의 데이터베이스 보안 인증 정보를 사용하려면 다음 지침을 따릅니다.
     1. AWS Secret에서 보안 암호의 이름을 선택합니다.
     2. Populate를 선택하여 Secrets Manager에서 데이터베이스 연결 대화 상자에 있는 모든 값을 자동으로 채웁니다.

     Secrets Manager의 데이터베이스 보안 인증 사용에 대한 자세한 내용은 AWS Secrets Manager 사용 섹션을 참조하세요.
   - Azure SQL Database 소스 데이터베이스 연결 정보를 수동으로 입력하려면 다음 지침을 사용합니다.

파라미터	작업
[Server name]	소스 데이터베이스 서버의 DNS(Domain Name Service) 이름 또는 IP 주소를 입력합니다.
데이터베이스	연결할 데이터베이스 이름을 입력합니다.
User name 및 Password	소스 데이터베이스 서버에 연결하는 데 필요한 데이터베이스 보안 인증 정보를 입력합니다. 프로젝트에서 데이터베이스에 연결하도록 선택한 경우에만 AWS SCT에서 암호를 사용하여 소스 데이터베이스에 연결합니다. 소스 데이터베이스의 암호가 노출될 위험을 방지하기 위해 AWS SCT는 기본적으로 암호를 저장하지 않습니다. AWS SCT 프로젝트를 닫았다 다시 열 경우 필요에 따라 소스 데이터베이스에 연결하기 위한 암호를 입력하라는 메시지가 표시됩니다.
Store password	AWS SCT는 SSL 인증서와 데이터베이스 암호를 저장할 안전한 볼트를 생성합니다. 이 옵션을 켜면 데이터베이스 암호를 저장하고 암호 입력 없이 빠르게 데이터베이스에 연결할 수 있습니다.

5. Test Connection을 선택하여 AWS SCT가 소스 데이터베이스에 연결할 수 있는지 확인합니다.
6. 연결을 선택하여 소스 데이터베이스에 연결합니다.

IBM Db2 for z/OS를 AWS SCT에 대한 소스로 사용

AWS SCT를 사용하여 스키마, 코드 객체 및 애플리케이션 코드를 IBM Db2 for z/OS에서 다음 대상으로 변환할 수 있습니다.

- Amazon RDS for MySQL
- Amazon Aurora MySQL-Compatible Edition
- Amazon RDS for PostgreSQL
- Amazon Aurora PostgreSQL-Compatible Edition

Db2 for z/OS를 소스 데이터베이스로 사용하기 위한 사전 조건

IBM Db2 for z/OS 버전 12 함수 수준 100 데이터베이스 버전은 IBM Db2 for z/OS 버전 12의 새로운 기능을 대부분 지원하지 않습니다. 이 데이터베이스 버전은 Db2 버전 11에 대한 폴백 및 Db2 버전 11과의 데이터 공유를 지원합니다. Db2 버전 11의 지원되지 않는 기능이 변환되지 않도록 하려면 IBM Db2 for z/OS 데이터베이스 함수 수준 500 이상을 AWS SCT에 대한 소스로 사용하는 것이 좋습니다.

다음 코드 예제를 사용하여 소스 IBM Db2 for z/OS 데이터베이스의 버전을 확인할 수 있습니다.

```
SELECT GETVARIABLE('SYSIBM.VERSION') as version FROM SYSIBM.SYSDUMMY1;
```

이 코드가 버전 `DSN12015` 이상을 반환해야 합니다.

다음 코드 예제를 사용하여 소스 IBM Db2 for z/OS 데이터베이스의 `APPLICATION COMPATIBILITY` 특수 레지스터 값을 확인할 수 있습니다.

```
SELECT CURRENT APPLICATION COMPATIBILITY as version FROM SYSIBM.SYSDUMMY1;
```

이 코드가 버전 `V12R1M500` 이상을 반환해야 합니다.

Db2 for z/OS를 소스 데이터베이스로 사용하기 위한 권한

Db2 for z/OS 데이터베이스에 연결하고 시스템 카탈로그 및 테이블을 읽는 데 필요한 권한은 다음과 같습니다.

- SELECT ON SYSIBM.LOCATIONS
- SELECT ON SYSIBM.SYSCHECKS
- SELECT ON SYSIBM.SYSCOLUMNS
- SELECT ON SYSIBM.SYSDATABASE
- SELECT ON SYSIBM.SYSDATATYPES
- SELECT ON SYSIBM.SYSDUMMY1
- SELECT ON SYSIBM.SYSFOREIGNKEYS
- SELECT ON SYSIBM.SYSINDEXES
- SELECT ON SYSIBM.SYSKEYCOLUSE
- SELECT ON SYSIBM.SYSKEYS
- SELECT ON SYSIBM.SYSKEYTARGETS

- SELECT ON SYSIBM.SYSJAROBJECTS
- SELECT ON SYSIBM.SYSPACKAGE
- SELECT ON SYSIBM.SYSPARMS
- SELECT ON SYSIBM.SYSRELS
- SELECT ON SYSIBM.SYSROUTINES
- SELECT ON SYSIBM.SYSSEQUENCES
- SELECT ON SYSIBM.SYSSEQUENCESDEP
- SELECT ON SYSIBM.SYSSYNONYMS
- SELECT ON SYSIBM.SYSTABCONST
- SELECT ON SYSIBM.SYSTABLES
- SELECT ON SYSIBM.SYSTABLESPACE
- SELECT ON SYSIBM.SYSTRIGGERS
- SELECT ON SYSIBM.SYSVARIABLES
- SELECT ON SYSIBM.SYSVIEWS

Db2 for z/OS 테이블을 PostgreSQL 파티션된 테이블로 변환하려면 다음과 같이 RUNSTATS 유틸리티를 사용하여 데이터베이스의 테이블스페이스 및 테이블에 대한 통계를 수집합니다.

```
LISTDEF YOURLIST INCLUDE TABLESPACES DATABASE YOURDB
RUNSTATS TABLESPACE
LIST YOURLIST
TABLE (ALL) INDEX (ALL KEYCARD)
UPDATE ALL
REPORT YES
SHRLEVEL REFERENCE
```

이전 예제에서 *YOURDB* 자리 표시자를 소스 데이터베이스의 이름으로 바꿉니다.

Db2 for z/OS에 소스로 연결

다음 절차에 따라 AWS SCT를 사용하여 Db2 for z/OS 소스 데이터베이스에 연결합니다.

IBM Db2 for z/OS 소스 데이터베이스에 연결하려면

1. AWS Schema Conversion Tool에서 소스 추가를 선택합니다.

2. IBM Db2 for z/OS를 선택하고 다음을 선택합니다.

   소스 추가 대화 상자가 나타납니다.

3. 연결 이름에 데이터베이스의 이름을 입력합니다. AWS SCT는 왼쪽 패널의 트리에 이 이름을 표시합니다.

4. AWS Secrets Manager의 데이터베이스 보안 인증 정보를 사용하거나 수동으로 입력합니다.

   - Secrets Manager의 데이터베이스 보안 인증 정보를 사용하려면 다음 지침을 따릅니다.

     1. AWS Secret에서 보안 암호의 이름을 선택합니다.
     2. Populate를 선택하여 Secrets Manager에서 데이터베이스 연결 대화 상자에 있는 모든 값을 자동으로 채웁니다.

     Secrets Manager의 데이터베이스 보안 인증 사용에 대한 자세한 내용은 AWS Secrets Manager 사용 섹션을 참조하세요.

   - IBM Db2 for z/OS 소스 데이터베이스 연결 정보를 수동으로 입력하려면 다음 지침을 사용합니다.

파라미터	작업
[Server name]	소스 데이터베이스 서버의 도메인 이름 시스템(DNS) 이름 또는 IP 주소를 입력합니다.
[Server port]	소스 데이터베이스 서버에 연결하는 데 사용되는 포트를 입력합니다.
위치	액세스할 Db2 위치의 고유한 이름을 입력합니다.
User name 및 Password	소스 데이터베이스 서버에 연결하는 데 필요한 데이터베이스 보안 인증 정보를 입력합니다. 프로젝트에서 데이터베이스에 연결하도록 선택한 경우에만 AWS SCT에서 암호를 사용하여 소스 데이터베이스에 연결합니다. 소스 데이터베이스의 암호가 노출될 위험을 방지하기 위해 AWS SCT는 기본적으로 암호를 저장하지 않습니다. AWS SCT 프로젝트를 닫았다 다시 열 경우 필요에 따라 소스 데이터베이스에 연결하기 위한 암호를 입력하라는 메시지가 표시됩니다.

파라미터	작업
Use SSL(SSL 사용)	SSL(Secure Sockets Layer)을 사용하여 데이터베이스에 연결하려면 이 옵션을 선택합니다. SSL 탭에서 다음 추가 정보를 적절히 제공합니다. • 트러스트 스토어: 인증서가 있는 트러스트 스토어의 위치입니다. 이 위치가 여기에 표시되도록 하려면 전역 설정에 추가해야 합니다.
Store password	AWS SCT는 SSL 인증서와 데이터베이스 암호를 저장할 안전한 볼트를 생성합니다. 이 옵션을 켜면 데이터베이스 암호를 저장하고 암호 입력 없이 빠르게 데이터베이스에 연결할 수 있습니다.
Db2 for z/OS driver path	소스 데이터베이스에 연결할 때 사용할 드라이버의 경로를 입력합니다. 자세한 내용은 필수 데이터베이스 드라이버 다운로드 섹션을 참조하세요. 드라이버 경로를 전역 프로젝트 설정에 저장할 경우 드라이버 경로가 연결 대화 상자에 표시되지 않습니다. 자세한 내용은 전역 설정에 드라이버 경로 저장 섹션을 참조하세요.

5. Test Connection을 선택하여 AWS SCT가 소스 데이터베이스에 연결할 수 있는지 확인합니다.
6. 연결을 선택하여 소스 데이터베이스에 연결합니다.

MySQL을 대상 데이터베이스로 사용하기 위한 권한

MySQL을 대상으로 사용하기 위해 필요한 권한은 다음과 같습니다.

- CREATE ON *.*
- ALTER ON *.*
- DROP ON *.*
- INDEX ON *.*
- REFERENCES ON *.*
- SELECT ON *.*
- CREATE VIEW ON *.*

- SHOW VIEW ON *.*
- TRIGGER ON *.*
- CREATE ROUTINE ON *.*
- ALTER ROUTINE ON *.*
- EXECUTE ON *.*
- SELECT ON mysql.proc
- INSERT, UPDATE ON AWS_DB2ZOS_EXT.*
- INSERT, UPDATE, DELETE ON AWS_DB2ZOS_EXT_DATA.*
- CREATE TEMPORARY TABLES ON AWS_DB2ZOS_EXT_DATA.*

다음 코드 예제를 사용하여 데이터베이스 사용자를 생성하고 권한을 부여할 수 있습니다.

```
CREATE USER 'user_name' IDENTIFIED BY 'your_password';
GRANT CREATE ON *.* TO 'user_name';
GRANT ALTER ON *.* TO 'user_name';
GRANT DROP ON *.* TO 'user_name';
GRANT INDEX ON *.* TO 'user_name';
GRANT REFERENCES ON *.* TO 'user_name';
GRANT SELECT ON *.* TO 'user_name';
GRANT CREATE VIEW ON *.* TO 'user_name';
GRANT SHOW VIEW ON *.* TO 'user_name';
GRANT TRIGGER ON *.* TO 'user_name';
GRANT CREATE ROUTINE ON *.* TO 'user_name';
GRANT ALTER ROUTINE ON *.* TO 'user_name';
GRANT EXECUTE ON *.* TO 'user_name';
GRANT SELECT ON mysql.proc TO 'user_name';
GRANT INSERT, UPDATE ON AWS_DB2ZOS_EXT.* TO 'user_name';
GRANT INSERT, UPDATE, DELETE ON AWS_DB2ZOS_EXT_DATA.* TO 'user_name';
GRANT CREATE TEMPORARY TABLES ON AWS_DB2ZOS_EXT_DATA.* TO 'user_name';
```

이전 예제에서 *user_name*을 사용자 이름으로 바꿉니다. 그런 다음 *your_password*를 안전한 암호로 바꿉니다.

Amazon RDS for MySQL을 대상으로 사용하려면 `log_bin_trust_function_creators` 파라미터를 true로 설정하고 `character_set_server`를 `latin1`로 설정합니다. 이들 파라미터를 구성하려면 새 DB 파라미터 그룹을 생성하거나 기존 DB 파라미터 그룹을 수정해야 합니다.

Aurora MySQL을 대상으로 사용하려면 log_bin_trust_function_creators 파라미터를 true로 설정하고 character_set_server를 latin1로 설정합니다. 또한 lower_case_table_names 파라미터를 true로 설정합니다. 이들 파라미터를 구성하려면 새 DB 파라미터 그룹을 생성하거나 기존 DB 파라미터 그룹을 수정해야 합니다.

PostgreSQL을 대상 데이터베이스로 사용하기 위한 권한

PostgreSQL을 대상으로 사용하려면 AWS SCT에 CREATE ON DATABASE 권한이 필요합니다. 각 대상 PostgreSQL 데이터베이스에 대해 이 권한을 부여해야 합니다.

Amazon RDS for PostgreSQL을 대상으로 사용하려면 AWS SCT에 rds_superuser 권한이 필요합니다.

변환된 공개 동의어를 사용하려면 데이터베이스 기본 검색 경로를 "$user", public_synonyms, public으로 변경합니다.

다음 코드 예제를 사용하여 데이터베이스 사용자를 생성하고 권한을 부여할 수 있습니다.

```
CREATE ROLE user_name LOGIN PASSWORD 'your_password';
GRANT CREATE ON DATABASE db_name TO user_name;
GRANT rds_superuser TO user_name;
ALTER DATABASE db_name SET SEARCH_PATH = "$user", public_synonyms, public;
```

이전 예제에서 *user_name*을 사용자 이름으로 바꿉니다. 그런 다음 *db_name*을 대상 데이터베이스의 이름으로 바꿉니다. 마지막으로 *your_password*를 안전한 암호로 바꿉니다.

PostgreSQL에서는 스키마 소유자 또는 superuser만 스키마를 삭제할 수 있습니다. 소유자는 스키마 소유자가 일부 객체를 소유하지 않은 경우에도 스키마 및 이 스키마에 포함된 모든 객체를 삭제할 수 있습니다.

여러 사용자를 통해 대상 데이터베이스를 변환하고 다른 스키마를 적용할 때 AWS SCT에서 스키마를 삭제할 수 없는 경우 오류 메시지가 표시될 수 있습니다. 이 오류 메시지가 표시되지 않도록 하려면 superuser 역할을 사용하세요.

Db2 for z/OS에서 PostgreSQL로의 변환 설정

Db2 for z/OS에서 PostgreSQL로의 변환 설정을 편집하려면 설정을 선택한 다음 변환 설정을 선택합니다. 상단 목록에서 Db2 for z/OS를 선택한 다음 Db2 for z/OS – PostgreSQL 또는 Db2 for z/OS – Amazon Aurora (PostgreSQL compatible)를 선택합니다. AWS SCT는 IBM Db2 for z/OS를 PostgreSQL로 변환하는 데 사용할 수 있는 모든 설정을 표시합니다.

AWS SCT의 경우, Db2 for z/OS에서 PostgreSQL로의 변환 설정에는 다음에 대한 옵션이 포함됩니다.

- 변환된 코드에서 작업 항목이 포함된 설명의 수를 제한합니다.

  Add comments in the converted code for the action items of selected severity and higher에서 작업 항목의 심각도를 선택합니다. AWS SCT가 선택된 심각도 이상의 작업 항목에 대한 변환된 코드에 설명을 추가합니다.

  예를 들어, 변환된 코드의 설명 수를 최소화하려면 오류만을 선택합니다. 변환된 코드의 모든 작업 항목에 대한 설명을 포함하려면 모든 메시지를 선택합니다.
- 대상 데이터베이스의 제약 조건에 대해 고유한 이름을 생성합니다.

  PostgreSQL에서 사용하는 모든 제약 조건 이름은 고유해야 합니다. AWS SCT는 테이블 이름이 포함된 접두사를 제약 조건 이름에 추가하여 변환된 코드에서 제약 조건의 고유한 이름을 생성할 수 있습니다. AWS SCT가 제약 조건에 대해 고유한 이름을 생성하도록 하려면 Generate unique names for constraints를 선택합니다.
- 변환된 코드에서 DML 문의 열 이름, 표현식 및 절 형식을 유지합니다.

  AWS SCT에서는 DML 문에 있는 열 이름, 표현식 및 절의 레이아웃을 소스 코드와 비슷한 위치 및 순서로 유지할 수 있습니다. 이렇게 하려면 Keep the formatting of column names, expressions, and clauses in DML statements에 대해 예를 선택합니다.
- 변환 범위에서 테이블 파티션을 제외합니다.

  AWS SCT는 변환 중에 소스 테이블의 모든 파티션을 건너뛸 수 있습니다. 이렇게 하려면 Exclude table partitions from the conversion scope를 선택합니다.
- 증가에 따라 파티셔닝된 테이블에 대해 자동 파티셔닝을 사용합니다.

  데이터 마이그레이션에서 AWS SCT는 지정된 크기보다 큰 모든 테이블을 자동으로 파티셔닝할 수 있습니다. 이 옵션을 사용하려면 Enforce partition of tables larger than을 선택하고 테이블 크기를 기가바이트 단위로 입력합니다. 그런 다음 파티션 수를 입력합니다. 이 옵션을 켜면 AWS SCT에서 DASD(Direct Access Storage Device) 크기를 고려합니다.

  AWS SCT에서는 파티션 수를 자동으로 결정할 수 있습니다. 이렇게 하려면 Increase the number of partitions proportionally를 선택하고 최대 파티션 수를 입력합니다.
- 동적 결과 세트를 refcursor 데이터 유형의 값 배열로 반환합니다.

AWS SCT는 동적 결과 세트를 반환하는 소스 프로시저를 추가 출력 파라미터로 열린 refcursor 배열이 포함된 프로시저로 변환할 수 있습니다. 이렇게 하려면 Use an array of refcursors to return all dynamic result sets를 선택합니다.

- 날짜 및 시간 값을 문자열 표현으로 변환하는 데 사용할 표준을 지정합니다.

  AWS SCT는 지원되는 산업 형식 중 하나를 사용하여 날짜 및 시간 값을 문자열 표현으로 변환할 수 있습니다. 이렇게 하려면 Use string representations of date values 또는 Use string representations of time values를 선택합니다. 그런 다음, 다음 표준 중 하나를 선택합니다.

  - 국제 표준 기구(ISO)
  - IBM 유럽 표준(EUR)
  - IBM 미국 표준(USA)
  - Japanese Industrial Standard Christian Era(JIS)

IBM Db2 LUW를 AWS SCT에 대한 소스로 사용

AWS SCT를 사용하여 스키마, SQL 언어로 된 코드 객체 및 애플리케이션 코드를 Db2 LUW(IBM Db2 for Linux, Unix, and Windows)에서 다음 대상으로 변환할 수 있습니다.

- Amazon RDS for MySQL
- Amazon Aurora MySQL-Compatible Edition
- Amazon RDS for PostgreSQL
- Amazon Aurora PostgreSQL-Compatible Edition
- Amazon RDS for MariaDB

AWS SCT는 소스로 Db2 LUW 버전 9.1, 9.5, 9.7, 10.1, 10.5, 11.1 및 11.5를 지원합니다.

Db2 LUW를 소스로 사용하기 위한 권한

Db2 LUW 데이터베이스에 연결하여 사용 가능한 권한을 확인하고 소스에 대한 스키마 메타데이터를 읽는 데 필요한 권한은 다음과 같습니다.

- 연결을 설정하는 데 필요한 권한:
  - CONNECT ON DATABASE
- SQL 문을 실행하는 데 필요한 권한:

  - EXECUTE ON PACKAGE NULLID.SYSSH200
- 인스턴스 수준 정보를 가져오는 데 필요한 권한:
  - EXECUTE ON FUNCTION SYSPROC.ENV_GET_INST_INFO
  - SELECT ON SYSIBMADM.ENV_INST_INFO
  - SELECT ON SYSIBMADM.ENV_SYS_INFO
- 역할, 그룹 및 권한을 통해 허용되는 권한을 점검하는 데 필요한 권한:
  - EXECUTE ON FUNCTION SYSPROC.AUTH_LIST_AUTHORITIES_FOR_AUTHID
  - EXECUTE ON FUNCTION SYSPROC.AUTH_LIST_GROUPS_FOR_AUTHID
  - EXECUTE ON FUNCTION SYSPROC.AUTH_LIST_ROLES_FOR_AUTHID
  - SELECT ON SYSIBMADM.PRIVILEGES
- 시스템 카탈로그 및 테이블에 필요한 권한:
  - SELECT ON SYSCAT.ATTRIBUTES
  - SELECT ON SYSCAT.CHECKS
  - SELECT ON SYSCAT.COLIDENTATTRIBUTES
  - SELECT ON SYSCAT.COLUMNS
  - SELECT ON SYSCAT.DATAPARTITIONEXPRESSION
  - SELECT ON SYSCAT.DATAPARTITIONS
  - SELECT ON SYSCAT.DATATYPEDEP
  - SELECT ON SYSCAT.DATATYPES
  - SELECT ON SYSCAT.HIERARCHIES
  - SELECT ON SYSCAT.INDEXCOLUSE
  - SELECT ON SYSCAT.INDEXES
  - SELECT ON SYSCAT.INDEXPARTITIONS
  - SELECT ON SYSCAT.KEYCOLUSE
  - SELECT ON SYSCAT.MODULEOBJECTS
  - SELECT ON SYSCAT.MODULES
  - SELECT ON SYSCAT.NICKNAMES
  - SELECT ON SYSCAT.PERIODS
  - SELECT ON SYSCAT.REFERENCES
  - SELECT ON SYSCAT.ROUTINEPARMS

  - SELECT ON SYSCAT.ROUTINES
  - SELECT ON SYSCAT.ROWFIELDS
  - SELECT ON SYSCAT.SCHEMATA
  - SELECT ON SYSCAT.SEQUENCES
  - SELECT ON SYSCAT.TABCONST
  - SELECT ON SYSCAT.TABLES
  - SELECT ON SYSCAT.TRIGGERS
  - SELECT ON SYSCAT.VARIABLEDEP
  - SELECT ON SYSCAT.VARIABLES
  - SELECT ON SYSCAT.VIEWS
  - SELECT ON SYSIBM.SYSDUMMY1
- SQL 문을 실행하려면 데이터베이스에서 활성화된 워크로드 중 하나 이상을 사용할 수 있는 권한이 사용자 계정에 있어야 합니다. 사용자에게 할당된 워크로드가 없는 경우에는 해당 사용자가 기본 사용자 워크로드에 액세스할 수 있어야 합니다.
  - USAGE ON WORKLOAD SYSDEFAULTUSERWORKLOAD

쿼리를 실행하려면 페이지 크기가 8K, 16K 및 32K인 시스템 임시 테이블스페이스를 생성해야 합니다 (테이블스페이스가 없는 경우). 임시 테이블스페이스를 생성하려면 다음 스크립트를 실행합니다.

```
CREATE BUFFERPOOL BP8K
  IMMEDIATE
  ALL DBPARTITIONNUMS
  SIZE AUTOMATIC
  NUMBLOCKPAGES 0
  PAGESIZE 8K;

CREATE SYSTEM TEMPORARY TABLESPACE TS_SYS_TEMP_8K
  PAGESIZE 8192
  BUFFERPOOL BP8K;

CREATE BUFFERPOOL BP16K
  IMMEDIATE
  ALL DBPARTITIONNUMS
  SIZE AUTOMATIC
  NUMBLOCKPAGES 0
  PAGESIZE 16K;
```

```
CREATE SYSTEM TEMPORARY TABLESPACE TS_SYS_TEMP_BP16K
  PAGESIZE 16384
  BUFFERPOOL BP16K;

CREATE BUFFERPOOL BP32K
  IMMEDIATE
  ALL DBPARTITIONNUMS
  SIZE AUTOMATIC
  NUMBLOCKPAGES 0
  PAGESIZE 32K;

CREATE SYSTEM TEMPORARY TABLESPACE TS_SYS_TEMP_BP32K
  PAGESIZE 32768
  BUFFERPOOL BP32K;
```

Db2 LUW에 소스로 연결

다음 절차에 따라 AWS Schema Conversion Tool을 사용하여 Db2 LUW 소스 데이터베이스에 연결합니다.

Db2 LUW 소스 데이터베이스에 연결하려면

1. AWS Schema Conversion Tool에서 소스 추가를 선택합니다.
2. Db2 LUW를 선택하고 다음을 선택합니다.

   소스 추가 대화 상자가 나타납니다.
3. 연결 이름에 데이터베이스의 이름을 입력합니다. AWS SCT는 왼쪽 패널의 트리에 이 이름을 표시합니다.
4. AWS Secrets Manager의 데이터베이스 보안 인증 정보를 사용하거나 수동으로 입력합니다.

   - Secrets Manager의 데이터베이스 보안 인증 정보를 사용하려면 다음 지침을 따릅니다.
     1. AWS Secret에서 보안 암호의 이름을 선택합니다.
     2. Populate를 선택하여 Secrets Manager에서 데이터베이스 연결 대화 상자에 있는 모든 값을 자동으로 채웁니다.

     Secrets Manager의 데이터베이스 보안 인증 사용에 대한 자세한 내용은 AWS Secrets Manager 사용 섹션을 참조하세요.
   - IBM Db2 LUW 소스 데이터베이스 연결 정보를 수동으로 입력하려면 다음 지침을 사용합니다.

파라미터	작업
[Server name]	소스 데이터베이스 서버의 도메인 이름 시스템(DNS) 이름 또는 IP 주소를 입력합니다.
[Server port]	소스 데이터베이스 서버에 연결하는 데 사용되는 포트를 입력합니다.
데이터베이스	Db2 LUW 데이터베이스의 이름을 입력합니다.
User name 및 Password	소스 데이터베이스 서버에 연결하는 데 필요한 데이터베이스 보안 인증 정보를 입력합니다. 프로젝트에서 데이터베이스에 연결하도록 선택한 경우에만 AWS SCT에서 암호를 사용하여 소스 데이터베이스에 연결합니다. 소스 데이터베이스의 암호가 노출될 위험을 방지하기 위해 AWS SCT는 기본적으로 암호를 저장하지 않습니다. AWS SCT 프로젝트를 닫았다 다시 열 경우 필요에 따라 소스 데이터베이스에 연결하기 위한 암호를 입력하라는 메시지가 표시됩니다.
Use SSL(SSL 사용)	SSL(Secure Sockets Layer)을 사용하여 데이터베이스에 연결하려면 이 옵션을 선택합니다. SSL 탭에서 다음 추가 정보를 적절히 제공합니다. • 트러스트 스토어: 인증서가 있는 트러스트 스토어의 위치입니다. 이 위치가 여기에 표시되도록 하려면 전역 설정에 추가해야 합니다.
Store password	AWS SCT는 SSL 인증서와 데이터베이스 암호를 저장할 안전한 볼트를 생성합니다. 이 옵션을 켜면 데이터베이스 암호를 저장하고 암호 입력 없이 빠르게 데이터베이스에 연결할 수 있습니다.

파라미터	작업
Db2 LUW driver path	소스 데이터베이스에 연결할 때 사용할 드라이버의 경로를 입력합니다. 자세한 내용은 필수 데이터베이스 드라이버 다운로드 섹션을 참조하세요. 드라이버 경로를 전역 프로젝트 설정에 저장할 경우 드라이버 경로가 연결 대화 상자에 표시되지 않습니다. 자세한 내용은 전역 설정에 드라이버 경로 저장 섹션을 참조하세요.

5. Test Connection을 선택하여 AWS SCT가 소스 데이터베이스에 연결할 수 있는지 확인합니다.
6. 연결을 선택하여 소스 데이터베이스에 연결합니다.

Db2 LUW를 Amazon RDS for PostgreSQL 또는 Amazon Aurora PostgreSQL-Compatible Edition으로 변환

IBM DB2 LUW를 PostgreSQL로 마이그레이션하는 경우, AWS SCT에서 Db2 LUW에 사용되는 다양한 트리거 문을 변환할 수 있습니다. 이러한 트리거 명령문에는 다음이 포함됩니다.

- 트리거 이벤트 – INSERT, DELETE, UPDATE 트리거 이벤트는 대상 테이블이나 대상 보기에 이벤트가 적용될 때마다 트리거된 작업이 실행되도록 지정합니다. INSERT, DELETE, UPDATE 이벤트를 조합하여 지정할 수 있지만 각 이벤트를 한 번씩만 지정할 수 있습니다. AWS SCT는 한 개 또는 여러 개의 트리거 이벤트를 지원합니다. 이벤트에 대해 PostgreSQL은 거의 동일한 기능을 제공합니다.
- OF COLUMN 이벤트 - 기본 테이블의 열 이름을 지정할 수 있습니다. 트리거는 열 이름 목록에서 식별된 열을 업데이트해야만 활성화됩니다. PostgreSQL에도 동일한 기능이 있습니다.
- 명령문 트리거 - 전체 명령문에 대해 트리거된 작업이 한 번만 적용되도록 지정합니다. BEFORE 트리거 또는 INSTEAD OF 트리거에는 이러한 유형의 트리거 세부 수준을 지정할 수 없습니다. 지정된 경우에는 영향을 받는 행이 없더라도 UPDATE 또는 DELETE 트리거가 활성화됩니다. PostgreSQL에도 이 기능이 있으며, 명령문 트리거에 대한 트리거 선언은 PostgreSQL과 Db2 LUW에서 동일합니다.
- 참조 절 - 변환 변수의 상관 관계 이름과 변환 테이블의 테이블 이름을 지정합니다. 상관 관계 이름은 트리거 SQL 작업의 영향을 받는 행 집합에서 특정 행을 식별합니다. 테이블 이름은 영향 받는 전체 행 집합을 식별합니다. 트리거 SQL 작업의 영향을 받는 각 행은 지정된 상관 관계 이름을 가진 열을 한정함으로써 트리거된 작업에 사용할 수 있습니다. PostgreSQL은 이 기능을 지원하지 않으며 NEW 또는 OLD 상관 관계 이름만 사용합니다.
- INSTEAD OF 트리거 - AWS SCT에서 이러한 트리거를 지원합니다.

Db2 LUW 파티션 테이블을 PostgreSQL 버전 10 파티션 테이블 변환

AWS SCT는 Db2 LUW 테이블을 PostgreSQL 10의 파티션 테이블로 변환할 수 있습니다. Db2 LUW 파티션 테이블을 PostgreSQL로 변환할 때는 다음과 같은 몇 가지 제한 사항이 있습니다.

- Db2 LUW에서 null이 허용되는 열을 포함하는 파티션 테이블을 생성할 수 있으며 NULL 값을 저장할 파티션을 지정할 수 있습니다. 하지만 PostgreSQL은 RANGE 파티셔닝에서 NULL 값을 지원하지 않습니다.
- Db2 LUW는 INCLUSIVE 또는 EXCLUSIVE 절을 사용하여 범위 경계 값을 설정할 수 있습니다. PostgreSQL은 시작 경계의 경우 INCLUSIVE, 끝 경계의 경우 EXCLUSIVE만 지원합니다. 변환된 파티션 이름은 <original_table_name>_<original_partition_name> 형식입니다.
- Db2 LUW에서 파티션 테이블의 기본 키 또는 고유 키를 생성할 수 있습니다. PostgreSQL에서는 각 파티션의 기본 키 또는 고유 키를 직접 생성해야 합니다. 기본 또는 고유 키 제약 조건은 상위 테이블에서 제거해야 합니다. 변환된 키 이름은 <original_key_name>_<original_partition _name> 형식입니다.
- Db2 LUW에서는 파티션 테이블에서 또는 파티션 테이블로의 외래 키 제약 조건을 생성할 수 있습니다. 하지만 PostgreSQL은 파티션 테이블에서 외래 키 참조를 지원하지 않습니다. 또한 PostgreSQL은 파티션 테이블에서 다른 테이블로의 외래 키 참조를 지원하지 않습니다.
- Db2 LUW의 파티션 테이블에 인덱스를 생성할 수 있습니다. 하지만 PostgreSQL에서는 각 파티션에 대한 인덱스를 직접 생성해야 합니다. 인덱스는 상위 테이블에서 제거해야 합니다. 변환된 인덱스 이름은 <original_index_name>_<original_partition_name> 형식입니다.
- 행 트리거는 파티션 테이블이 아닌 개별 파티션에 정의해야 합니다. 트리거는 상위 테이블에서 제거해야 합니다. 변환된 트리거 이름은 <original_trigger_name>_<original_partition_name> 형식입니다.

PostgreSQL을 대상으로 사용하기 위한 권한

PostgreSQL을 대상으로 사용하려면 AWS SCT에 `CREATE ON DATABASE` 권한이 필요합니다. 각 대상 PostgreSQL 데이터베이스에 대해 이 권한을 부여해야 합니다.

변환된 공개 동의어를 사용하려면 데이터베이스 기본 검색 경로를 `"$user", public_synonyms, public`으로 변경합니다.

다음 코드 예제를 사용하여 데이터베이스 사용자를 생성하고 권한을 부여할 수 있습니다.

```
CREATE ROLE user_name LOGIN PASSWORD 'your_password';
GRANT CREATE ON DATABASE db_name TO user_name;
ALTER DATABASE db_name SET SEARCH_PATH = "$user", public_synonyms, public;
```

이전 예제에서 *user_name*을 사용자 이름으로 바꿉니다. 그런 다음 *db_name*을 대상 데이터베이스의 이름으로 바꿉니다. 마지막으로 *your_password*를 안전한 암호로 바꿉니다.

PostgreSQL에서는 스키마 소유자 또는 superuser만 스키마를 삭제할 수 있습니다. 소유자는 스키마 소유자가 일부 객체를 소유하지 않은 경우에도 스키마 및 이 스키마에 포함된 모든 객체를 삭제할 수 있습니다.

여러 사용자를 통해 대상 데이터베이스를 변환하고 다른 스키마를 적용할 때 AWS SCT에서 스키마를 삭제할 수 없는 경우 오류 메시지가 표시될 수 있습니다. 이 오류 메시지가 표시되지 않도록 하려면 superuser 역할을 사용하세요.

Db2 LUW를 Amazon RDS for MySQL 또는 Amazon Aurora MySQL로 변환

IBM Db2 LUW 데이터베이스를 RDS for MySQL 또는 Amazon Aurora MySQL로 변환하는 경우 다음 사항에 유의합니다.

MySQL을 대상으로 사용하기 위한 권한

MySQL을 대상으로 사용하기 위해 필요한 권한은 다음과 같습니다.

- CREATE ON *.*
- ALTER ON *.*
- DROP ON *.*
- INDEX ON *.*
- REFERENCES ON *.*
- SELECT ON *.*
- CREATE VIEW ON *.*
- SHOW VIEW ON *.*
- TRIGGER ON *.*
- CREATE ROUTINE ON *.*
- ALTER ROUTINE ON *.*
- EXECUTE ON *.*
- SELECT ON mysql.proc
- INSERT, UPDATE ON AWS_DB2_EXT.*
- INSERT, UPDATE, DELETE ON AWS_DB2_EXT_DATA.*
- CREATE TEMPORARY TABLES ON AWS_DB2_EXT_DATA.*

다음 코드 예제를 사용하여 데이터베이스 사용자를 생성하고 권한을 부여할 수 있습니다.

```
CREATE USER 'user_name' IDENTIFIED BY 'your_password';
GRANT CREATE ON *.* TO 'user_name';
GRANT ALTER ON *.* TO 'user_name';
GRANT DROP ON *.* TO 'user_name';
GRANT INDEX ON *.* TO 'user_name';
GRANT REFERENCES ON *.* TO 'user_name';
GRANT SELECT ON *.* TO 'user_name';
GRANT CREATE VIEW ON *.* TO 'user_name';
GRANT SHOW VIEW ON *.* TO 'user_name';
GRANT TRIGGER ON *.* TO 'user_name';
GRANT CREATE ROUTINE ON *.* TO 'user_name';
GRANT ALTER ROUTINE ON *.* TO 'user_name';
GRANT EXECUTE ON *.* TO 'user_name';
GRANT SELECT ON mysql.proc TO 'user_name';
GRANT INSERT, UPDATE ON AWS_DB2_EXT.* TO 'user_name';
GRANT INSERT, UPDATE, DELETE ON AWS_DB2_EXT_DATA.* TO 'user_name';
GRANT CREATE TEMPORARY TABLES ON AWS_DB2_EXT_DATA.* TO 'user_name';
```

이전 예제에서 *user_name*을 사용자 이름으로 바꿉니다. 그런 다음 *your_password*를 안전한 암호로 바꿉니다.

Amazon RDS for MySQL 또는 Amazon RDS for Aurora MySQL을 대상으로 사용하려면 `lower_case_table_names` 파라미터를 1로 설정합니다. 이 값은 MySQL 서버가 테이블, 인덱스, 트리거 및 데이터베이스와 같은 객체 이름의 식별자를 대소문자 구분 없이 처리한다는 것을 의미합니다. 대상 인스턴스에서 이진 로깅을 활성화했다면 `log_bin_trust_function_creators` 파라미터를 1로 설정합니다. 이 경우 저장된 함수를 생성하기 위해 `DETERMINISTIC`, `READS SQL DATA` 또는 `NO SQL` 특성을 사용할 필요가 없습니다. 이들 파라미터를 구성하려면 새 DB 파라미터 그룹을 생성하거나 기존 DB 파라미터 그룹을 수정해야 합니다.

MySQL을 AWS SCT에 대한 소스로 사용

AWS SCT를 사용하여 스키마, 데이터베이스 코드 객체 및 애플리케이션 코드를 MySQL에서 다음 대상으로 변환할 수 있습니다.

- Amazon RDS for PostgreSQL
- Amazon Aurora PostgreSQL-Compatible Edition
- Amazon RDS for MySQL

자세한 내용은 다음 단원을 참조하세요.

주제

- MySQL을 소스 데이터베이스로 사용할 수 있는 권한
- MySQL 소스에 연결
- PostgreSQL을 대상 데이터베이스로 사용하기 위한 권한

MySQL을 소스 데이터베이스로 사용할 수 있는 권한

MySQL이 소스일 경우 필요한 권한은 다음과 같습니다.

- SELECT ON *.*
- SHOW VIEW ON *.*

MySQL 소스에 연결

다음 절차를 통해 AWS Schema Conversion Tool을 사용하여 MySQL 소스 데이터베이스에 연결합니다.

MySQL 소스 데이터베이스에 연결하려면

1. AWS Schema Conversion Tool에서 소스 추가를 선택합니다.
2. MySQL을 선택한 후 다음을 선택합니다.

   소스 추가 대화 상자가 나타납니다.
3. 연결 이름에 데이터베이스의 이름을 입력합니다. AWS SCT는 왼쪽 패널의 트리에 이 이름을 표시합니다.
4. AWS Secrets Manager의 데이터베이스 보안 인증 정보를 사용하거나 수동으로 입력합니다.

   - Secrets Manager의 데이터베이스 보안 인증 정보를 사용하려면 다음 지침을 따릅니다.
     1. AWS Secret에서 보안 암호의 이름을 선택합니다.
     2. Populate를 선택하여 Secrets Manager에서 데이터베이스 연결 대화 상자에 있는 모든 값을 자동으로 채웁니다.

     Secrets Manager의 데이터베이스 보안 인증 사용에 대한 자세한 내용은 AWS Secrets Manager 사용 섹션을 참조하세요.

- MySQL 소스 데이터베이스 연결 정보를 수동으로 입력하려면 다음 지침을 사용합니다.

파라미터	작업
[Server name]	소스 데이터베이스 서버의 도메인 이름 시스템(DNS) 이름 또는 IP 주소를 입력합니다. IPv6 주소 프로토콜을 사용하여 소스 MySQL 데이터베이스에 연결할 수 있습니다. 이렇게 하려면 다음 예제와 같이 대괄호를 사용하여 IP 주소를 입력해야 합니다. `[2001:db8:ffff:ffff:ffff:ffff:ffff:fffe]`
[Server port]	소스 데이터베이스 서버에 연결하는 데 사용되는 포트를 입력합니다.
User name 및 Password	소스 데이터베이스 서버에 연결하는 데 필요한 데이터베이스 보안 인증 정보를 입력합니다. 프로젝트에서 데이터베이스에 연결하도록 선택한 경우에만 AWS SCT에서 암호를 사용하여 소스 데이터베이스에 연결합니다. 소스 데이터베이스의 암호가 노출될 위험을 방지하기 위해 AWS SCT는 기본적으로 암호를 저장하지 않습니다. AWS SCT 프로젝트를 닫았다 다시 열 경우 필요에 따라 소스 데이터베이스에 연결하기 위한 암호를 입력하라는 메시지가 표시됩니다.

파라미터	작업
Use SSL(SSL 사용)	SSL(Secure Sockets Layer)을 사용하여 데이터베이스에 연결하려면 이 옵션을 선택합니다. SSL 탭에서 다음 추가 정보를 적절히 제공합니다. • Require SSL: SSL을 통해서만 서버에 연결하려면 이 옵션을 선택합니다. 따라서 Require SSL을 선택하면 서버가 SSL을 지원하지 않는 경우 서버에 연결할 수 없습니다. Require SSL을 선택하지 않고 서버가 SSL을 지원하지 않는 경우 SSL을 사용하지 않고 계속 서버에 연결할 수 있습니다. 자세한 내용은 보안 연결을 사용하도록 MySQL 구성을 참조하세요. • Verify server certificate: 트러스트 스토어를 사용하여 서버 인증서를 확인하려면 이 옵션을 선택합니다. • 트러스트 스토어: 인증서가 있는 트러스트 스토어의 위치입니다.
Store Password	AWS SCT는 SSL 인증서와 데이터베이스 암호를 저장할 안전한 볼트를 생성합니다. 이 옵션을 활성화하면 데이터베이스 암호를 저장하고 암호 입력 없이 빠르게 데이터베이스에 연결할 수 있습니다.
MySql driver path	소스 데이터베이스에 연결할 때 사용할 드라이버의 경로를 입력합니다. 자세한 내용은 필수 데이터베이스 드라이버 다운로드 섹션을 참조하세요. 드라이버 경로를 전역 프로젝트 설정에 저장할 경우 드라이버 경로가 연결 대화 상자에 표시되지 않습니다. 자세한 내용은 전역 설정에 드라이버 경로 저장 섹션을 참조하세요.

5. Test Connection을 선택하여 AWS SCT가 소스 데이터베이스에 연결할 수 있는지 확인합니다.
6. 연결을 선택하여 소스 데이터베이스에 연결합니다.

PostgreSQL을 대상 데이터베이스로 사용하기 위한 권한

PostgreSQL을 대상으로 사용하려면 AWS SCT에 `CREATE ON DATABASE` 권한이 필요합니다. 각 대상 PostgreSQL 데이터베이스에 대해 이 권한을 부여해야 합니다.

변환된 공개 동의어를 사용하려면 데이터베이스 기본 검색 경로를 `"$user", public_synonyms, public`으로 변경합니다.

다음 코드 예제를 사용하여 데이터베이스 사용자를 생성하고 권한을 부여할 수 있습니다.

```
CREATE ROLE user_name LOGIN PASSWORD 'your_password';
GRANT CREATE ON DATABASE db_name TO user_name;
ALTER DATABASE db_name SET SEARCH_PATH = "$user", public_synonyms, public;
```

이전 예제에서 *user_name*을 사용자 이름으로 바꿉니다. 그런 다음 *db_name*을 대상 데이터베이스의 이름으로 바꿉니다. 마지막으로 *your_password*를 안전한 암호로 바꿉니다.

PostgreSQL에서는 스키마 소유자 또는 `superuser`만 스키마를 삭제할 수 있습니다. 소유자는 스키마 소유자가 일부 객체를 소유하지 않은 경우에도 스키마 및 이 스키마에 포함된 모든 객체를 삭제할 수 있습니다.

여러 사용자를 통해 대상 데이터베이스를 변환하고 다른 스키마를 적용할 때 AWS SCT에서 스키마를 삭제할 수 없는 경우 오류 메시지가 표시될 수 있습니다. 이 오류 메시지가 표시되지 않도록 하려면 `superuser` 역할을 사용하세요.

Oracle Database를 AWS SCT에 대한 소스로 사용

AWS SCT를 사용하여 스키마, 데이터베이스 코드 객체 및 애플리케이션 코드를 Oracle Database에서 다음 대상으로 변환할 수 있습니다.

- Amazon RDS for MySQL
- Amazon Aurora MySQL-Compatible Edition
- Amazon RDS for PostgreSQL
- Amazon Aurora PostgreSQL-Compatible Edition
- Amazon RDS for Oracle
- Amazon RDS for MariaDB

소스가 Oracle 데이터베이스인 경우 설명을 PostgreSQL 데이터베이스와 같은 적절한 형식으로 변환할 수 있습니다. AWS SCT에서는 테이블, 보기 및 열에 관한 설명을 변환할 수 있습니다. 설명에는 아포스트로피가 포함될 수 있으며, AWS SCT에서는 문자열 리터럴에서와 마찬가지로 SQL 문을 변환할 때 두 개의 아포스트로피를 사용합니다.

자세한 내용은 다음을 참조하세요.

주제

- Oracle을 소스로 사용하기 위한 권한
- Oracle 소스에 연결
- Oracle을 Amazon RDS for PostgreSQL 또는 Amazon Aurora PostgreSQL로 변환
- Oracle을 Amazon RDS for MySQL 또는 Amazon Aurora MySQL로 변환
- Oracle에서 Amazon RDS for Oracle로 변환

Oracle을 소스로 사용하기 위한 권한

Oracle이 소스일 경우 필요한 권한은 다음과 같습니다.

- CONNECT
- SELECT_CATALOG_ROLE
- SELECT ANY DICTIONARY
- SELECT ON SYS.ARGUMENT$

Oracle 소스에 연결

다음 절차를 통해 AWS Schema Conversion Tool을 사용하여 Oracle 원본 데이터베이스에 연결합니다.

Oracle 소스 데이터베이스에 연결하려면

1. AWS Schema Conversion Tool에서 소스 추가를 선택합니다.
2. Oracle를 선택한 후 다음을 선택합니다.

   소스 추가 대화 상자가 나타납니다.
3. 연결 이름에 데이터베이스의 이름을 입력합니다. AWS SCT는 왼쪽 패널의 트리에 이 이름을 표시합니다.

4. AWS Secrets Manager의 데이터베이스 보안 인증 정보를 사용하거나 수동으로 입력합니다.

   - Secrets Manager의 데이터베이스 보안 인증 정보를 사용하려면 다음 지침을 따릅니다.

     1. AWS Secret에서 보안 암호의 이름을 선택합니다.
     2. Populate를 선택하여 Secrets Manager에서 데이터베이스 연결 대화 상자에 있는 모든 값을 자동으로 채웁니다.

     Secrets Manager의 데이터베이스 보안 인증 사용에 대한 자세한 내용은 AWS Secrets Manager 사용 섹션을 참조하세요.

   - Oracle 소스 데이터베이스 연결 정보를 수동으로 입력하려면 다음 지침을 사용합니다.

파라미터	작업

유형	데이터베이스 연결 유형을 선택합니다. 유형에 따라 다음의 추가 정보를 제공합니다.
	• SID
	  • 서버 이름: 소스 데이터베이스 서버의 도메인 이름 시스템(DNS) 이름 또는 IP 주소입니다.
	  • Server port: 소스 데이터베이스 서버에 연결하는 데 사용되는 포트입니다.
	  • Oracle SID: Oracle System ID(SID)입니다. Oracle SID를 확인하려면 Oracle 데이터베이스에 다음 쿼리를 제출합니다.
	  `SELECT sys_context('userenv','instance_name') AS SID FROM dual;`
	• 서비스 이름
	  • Server name: 소스 데이터베이스 서버의 DNS 이름 또는 IP 주소입니다.
	  IPv6 주소 프로토콜을 사용하여 소스 Oracle 데이터베이스에 연결할 수 있습니다. 이렇게 하려면 다음 예제와 같이 대괄호를 사용하여 IP 주소를 입력해야 합니다.
	  `[2001:db8:ffff:ffff:ffff:ffff:ffff:fffe]`
	  • Server port: 소스 데이터베이스 서버에 연결하는 데 사용되는 포트입니다.
	  • 서비스 이름: 연결할 Oracle 서비스의 이름입니다.
	• TNS alias
	  • TNS file path: TNS(Transparent Network Substrate) 이름 연결 정보가 저장된 파일의 경로입니다.
	  TNS 파일을 선택하면 AWS SCT가 파일의 모든 Oracle 데이터베이스 연결을 TNS alias 목록에 추가합니다.

파라미터	작업
	Oracle Real Application Clusters(RAC)에 연결하려면 이 옵션을 선택합니다. • TNS alias: 소스 데이터베이스에 연결하는 데 사용할 이 파일의 TNS 별칭입니다. • TNS connect identifier • TNS connect identifier: 등록된 TNS 연결 정보의 식별자입니다.
User name 및 Password	소스 데이터베이스 서버에 연결하는 데 필요한 데이터베이스 보안 인증 정보를 입력합니다. Oracle 데이터베이스에 처음으로 연결할 때는 Oracle Driver 파일(ojdbc8.jar)의 경로를 입력해야 합니다. 이 파일은 http://www.oracle.com/technetwork/database/features/jdbc/index-091264.html에서 다운로드하실 수 있습니다. 다운로드를 완료하려면 무료 Oracle Technical Network 웹 사이트에 등록해야 합니다. AWS SCT는 선택한 드라이버를 향후 모든 Oracle 데이터베이스 연결에 사용합니다. 드라이버 경로는 전역 설정에서 드라이버 탭을 사용하여 수정할 수 있습니다. 프로젝트에서 데이터베이스에 연결하도록 선택한 경우에만 AWS SCT에서 암호를 사용하여 소스 데이터베이스에 연결합니다. 소스 데이터베이스의 암호가 노출될 위험을 방지하기 위해 AWS SCT는 기본적으로 암호를 저장하지 않습니다. AWS SCT 프로젝트를 닫았다 다시 열 경우 필요에 따라 소스 데이터베이스에 연결하기 위한 암호를 입력하라는 메시지가 표시됩니다.

파라미터	작업
Use SSL(SSL 사용)	SSL(Secure Sockets Layer)을 사용하여 데이터베이스에 연결하려면 이 옵션을 선택합니다. SSL 탭에서 다음 추가 정보를 적절히 제공합니다. • SSL 인증: 인증서별 SSL 인증을 사용하려면 이 옵션을 선택합니다. 설정, 전역 설정, 보안에서 트러스트 스토어와 키 스토어를 설정합니다. • 트러스트 스토어: 사용할 트러스트 스토어입니다. • 키 스토어: 사용할 키 스토어입니다.
Store password	AWS SCT는 SSL 인증서와 데이터베이스 암호를 저장할 안전한 볼트를 생성합니다. 데이터베이스 암호를 저장하고 암호 입력 없이 빠르게 데이터베이스에 연결하려면 이 옵션을 선택합니다.
Oracle driver path	소스 데이터베이스에 연결할 때 사용할 드라이버의 경로를 입력합니다. 자세한 내용은 필수 데이터베이스 드라이버 다운로드 섹션을 참조하세요. 드라이버 경로를 전역 프로젝트 설정에 저장할 경우 드라이버 경로가 연결 대화 상자에 표시되지 않습니다. 자세한 내용은 전역 설정에 드라이버 경로 저장 섹션을 참조하세요.

5. Test Connection을 선택하여 AWS SCT가 소스 데이터베이스에 연결할 수 있는지 확인합니다.
6. 연결을 선택하여 소스 데이터베이스에 연결합니다.

Oracle을 Amazon RDS for PostgreSQL 또는 Amazon Aurora PostgreSQL로 변환

Oracle 데이터베이스를 RDS for PostgreSQL 또는 Amazon Aurora PostgreSQL로 변환하는 경우 다음 사항에 유의합니다.

주제

- PostgreSQL을 대상 데이터베이스로 사용하기 위한 권한
- Oracle에서 PostgreSQL로 변환 설정

- Oracle 시퀀스 변환
- Oracle ROWID 변환
- Oracle 동적 SQL 변환
- Oracle 파티션 변환

Oracle 시스템 객체를 PostgreSQL로 변환할 때 AWS SCT는 다음 테이블에 나온 대로 변환을 수행합니다.

Oracle 시스템 객체	설명	변환된 PostgreSQL 객체
V$VERSION	Oracle 데이터베이스에 있는 핵심 라이브러리 구성 요소의 버전 번호 표시	aws_oracle_ext.v$version
V$INSTANCE	현재 인스턴스의 상태를 나타내는 보기	aws_oracle_ext.v$instance

AWS SCT를 사용하여 Oracle SQL*Plus 파일을 psql로 변환할 수 있습니다. psql은 PostgreSQL의 터미널 기반 프런트 엔드입니다. 자세한 내용은 AWS SCT를 사용하여 애플리케이션 SQL 변환 섹션을 참조하세요.

PostgreSQL을 대상 데이터베이스로 사용하기 위한 권한

PostgreSQL을 대상으로 사용하려면 AWS SCT에 `CREATE ON DATABASE` 권한이 필요합니다. 각 대상 PostgreSQL 데이터베이스에 대해 이 권한을 부여해야 합니다.

변환된 공개 동의어를 사용하려면 데이터베이스 기본 검색 경로를 `"$user", public_synonyms, public`으로 변경합니다.

다음 코드 예제를 사용하여 데이터베이스 사용자를 생성하고 권한을 부여할 수 있습니다.

```
CREATE ROLE user_name LOGIN PASSWORD 'your_password';
GRANT CREATE ON DATABASE db_name TO user_name;
ALTER DATABASE db_name SET SEARCH_PATH = "$user", public_synonyms, public;
```

이전 예제에서 *user_name*을 사용자 이름으로 바꿉니다. 그런 다음 *db_name*을 대상 데이터베이스의 이름으로 바꿉니다. 마지막으로 *your_password*를 안전한 암호로 바꿉니다.

Amazon RDS for PostgreSQL을 대상으로 사용하려면 AWS SCT에 `rds_superuser` 권한이 필요합니다.

PostgreSQL에서는 스키마 소유자 또는 `superuser`만 스키마를 삭제할 수 있습니다. 소유자는 스키마 소유자가 일부 객체를 소유하지 않은 경우에도 스키마 및 이 스키마에 포함된 모든 객체를 삭제할 수 있습니다.

여러 사용자를 통해 대상 데이터베이스를 변환하고 다른 스키마를 적용할 때 AWS SCT에서 스키마를 삭제할 수 없는 경우 오류 메시지가 표시될 수 있습니다. 이 오류 메시지가 표시되지 않도록 하려면 `superuser` 역할을 사용하세요.

Oracle에서 PostgreSQL로 변환 설정

Oracle에서 PostgreSQL로의 변환 설정을 편집하려면 AWS SCT에서 설정을 선택한 다음 변환 설정을 선택합니다. 상단 목록에서 Oracle을 선택한 다음 Oracle – PostgreSQL을 선택합니다. AWS SCT는 Oracle에서 PostgreSQL로의 변환에 사용할 수 있는 모든 설정을 표시합니다.

AWS SCT의 경우, Oracle에서 PostgreSQL로의 변환 설정에는 다음에 대한 옵션이 포함됩니다.

- 변환된 코드에서 작업 항목이 포함된 설명의 수를 제한합니다.

  Add comments in the converted code for the action items of selected severity and higher에서 작업 항목의 심각도를 선택합니다. AWS SCT가 선택된 심각도 이상의 작업 항목에 대한 변환된 코드에 설명을 추가합니다.

  예를 들어, 변환된 코드의 설명 수를 최소화하려면 오류만을 선택합니다. 변환된 코드의 모든 작업 항목에 대한 설명을 포함하려면 모든 메시지를 선택합니다.
- AWS SCT가 Oracle 구체화된 뷰를 PostgreSQL의 테이블 또는 구체화된 뷰로 변환할 수 있습니다. Materialized view conversion as에서 소스 구체화된 뷰를 변환하는 방법을 선택합니다.
- PostgreSQL에서 지원하지 않는 파라미터가 있는 `TO_CHAR`, `TO_DATE` 및 `TO_NUMBER` 함수를 포함하는 경우 소스 Oracle 코드를 사용하여 작업합니다. 기본적으로 AWS SCT는 이러한 파라미터의 사용을 변환된 코드로 에뮬레이션합니다.

  소스 Oracle 코드에 PostgreSQL이 지원하는 파라미터만 포함되어 있는 경우 기본 PostgreSQL `TO_CHAR`, `TO_DATE` 및 `TO_NUMBER` 함수를 사용할 수 있습니다. 이 경우, 변환된 코드는 더 빠르게 작동합니다. 이러한 파라미터만 포함하려면 다음 값을 선택합니다.
  - Function TO_CHAR() does not use Oracle specific formatting strings
  - Function TO_DATE() does not use Oracle specific formatting strings
  - Function TO_NUMBER() does not use Oracle specific formatting strings

- 소스 Oracle 데이터베이스가 NUMBER 데이터 형식의 기본 또는 외래 키 열에 정수 값만 저장하는 경우를 해결하기 위해 AWS SCT에서 이러한 열을 BIGINT 데이터 형식으로 변환할 수 있습니다. 이 방식을 적용하면 변환된 코드의 성능이 향상됩니다. 이 방법을 사용하려면 Convert NUMBER primary / foreign key columns to BIGINT ones를 선택합니다. 데이터 손실을 방지하려면 소스에서 이러한 열에 부동 소수점 값이 포함되지 않도록 해야 합니다.
- 소스 코드에서 비활성화된 트리거와 제약 조건을 건너뜁니다. 이렇게 하려면 Ignore disabled triggers and constraints를 선택합니다.
- AWS SCT를 사용하여 동적 SQL이라고 하는 문자열 변수를 변환합니다. 데이터베이스 코드는 이러한 문자열 변수의 값을 변경할 수 있습니다. AWS SCT가 이 문자열 변수의 최신 값을 항상 변환하도록 하려면 Convert the dynamic SQL code that is created in called routines를 선택합니다.
- PostgreSQL 버전 10 이전에서 프로시저를 지원하지 않는 문제를 해결합니다. PostgreSQL에서 프로시저를 사용하는 데 익숙하지 않은 경우 AWS SCT는 Oracle 프로시저를 PostgreSQL 함수로 변환할 수 있습니다. 이렇게 하려면 프로시저를 함수로 변환을 선택합니다.
- 발생한 작업 항목에 대한 추가 정보를 확인합니다. 이렇게 하기 위해 Add on exception raise block for migration issues with the next severity levels를 선택하여 확장 팩에 특정 함수를 추가할 수 있습니다. 그런 다음 사용자 정의 예외를 발생시킬 심각도 수준을 선택합니다.
- 자동으로 생성된 이름이 있는 제약 조건을 포함할 수 있는 소스 Oracle 데이터베이스를 사용하여 작업합니다. 소스 코드에서 이러한 이름을 사용하는 경우 Convert the system generated constraint names using the source original names를 선택해야 합니다. 소스 코드에서 이러한 제약 조건을 사용하지만 해당 이름은 사용하지 않는 경우 이 옵션을 선택 취소하여 변환 속도를 높입니다.
- 데이터베이스와 애플리케이션이 서로 다른 시간대에서 실행되는지 여부를 확인합니다. 기본적으로 AWS SCT는 변환된 코드로 시간대를 에뮬레이션합니다. 하지만 데이터베이스와 애플리케이션이 동일한 시간대를 사용하는 경우에는 이 에뮬레이션이 필요하지 않습니다. 이 경우 Time zone on the client side matches the time zone on server를 선택합니다.
- 소스 데이터베이스와 대상 데이터베이스가 서로 다른 시간대에서 실행되는지 여부를 확인합니다. 서로 다른 시간대에서 실행되는 경우, SYSDATE 내장 Oracle 함수를 에뮬레이션하는 함수가 소스 함수와 비교해 다른 값을 반환합니다. 소스 함수와 대상 함수가 동일한 값을 반환하도록 하려면 Set default time zone for SYSDATE emulation을 선택합니다.
- orafce 확장의 함수를 변환된 코드에서 사용합니다. 이렇게 하려면 Use orafce implementation에서 사용할 함수를 선택합니다. orafce에 대한 자세한 내용은 GitHub에서 orafce를 참조하세요.

Oracle 시퀀스 변환

AWS SCT가 Oracle에서 PostgreSQL로 시퀀스를 변환합니다. 시퀀스를 사용하여 무결성 제약 조건을 유지하는 경우 마이그레이션된 시퀀스의 새 값이 기존 값과 겹치지 않도록 해야 합니다.

변환된 시퀀스를 소스 데이터베이스의 마지막 값으로 채우려면

1. Oracle을 소스로 사용하여 AWS SCT 프로젝트를 엽니다.
2. 설정을 선택한 다음 변환 설정을 선택합니다.
3. 상단 목록에서 Oracle을 선택한 다음 Oracle – PostgreSQL을 선택합니다. AWS SCT는 Oracle에서 PostgreSQL로의 변환에 사용할 수 있는 모든 설정을 표시합니다.
4. Populate converted sequences with the last value generated on the source side를 선택합니다.
5. 확인을 선택하여 설정을 저장하고 변환 설정 대화 상자를 닫습니다.

Oracle ROWID 변환

Oracle 데이터베이스에서 ROWID 유사 열(pseudocolumn)에는 테이블 행의 주소가 들어 있습니다. ROWID 의사 열은 Oracle 전용이므로, AWS SCT가 ROWID 의사 열을 PostgreSQL의 데이터 열로 변환할 수 있습니다. 이 변환을 사용하면 ROWID 정보를 유지할 수 있습니다.

ROWID 의사 열을 변환할 때 AWS SCT는 `bigint` 데이터 형식을 사용하여 데이터 열을 생성할 수 있습니다. 프라이머리 키가 없는 경우 AWS SCT는 ROWID 열을 프라이머리 키로 설정합니다. 프라이머리 키가 있으면 AWS SCT는 고유한 제약을 포함하는 ROWID 열을 설정합니다.

소스 데이터베이스 코드에 숫자 데이터 형식을 사용하여 실행할 수 없는 ROWID 연산이 포함되어 있는 경우 AWS SCT는 `character varying` 데이터 형식으로 데이터 열을 생성할 수 있습니다.

프로젝트에 Oracle ROWID용 데이터 열을 생성하려면

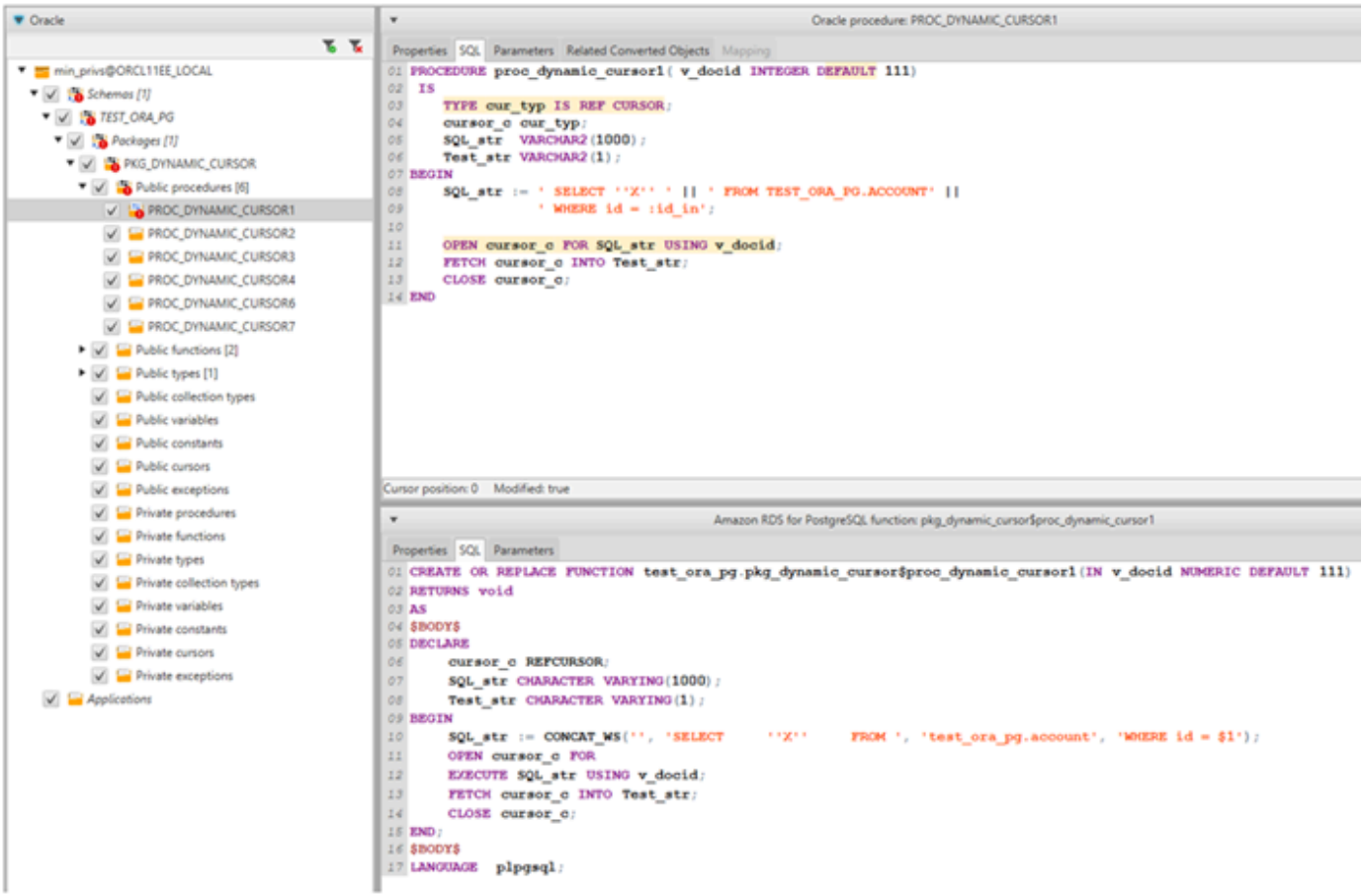
1. Oracle을 소스로 사용하여 AWS SCT 프로젝트를 엽니다.
2. 설정을 선택한 다음 변환 설정을 선택합니다.
3. 상단 목록에서 Oracle을 선택한 다음 Oracle – PostgreSQL을 선택합니다. AWS SCT는 Oracle에서 PostgreSQL로의 변환에 사용할 수 있는 모든 설정을 표시합니다.
4. 행 ID 생성에서 다음 중 하나를 수행합니다.
   - 숫자 데이터 열을 생성하려면 Generate as identity을 선택합니다.
   - 문자 데이터 열을 생성하려면 Generate as character domain type을 선택합니다.
5. 확인을 선택하여 설정을 저장하고 변환 설정 대화 상자를 닫습니다.

Oracle 동적 SQL 변환

Oracle은 동적 SQL을 구현하는 두 가지 방법을 제공합니다. 하나는 EXECUTE IMMEDIATE 문을 사용하는 것이고 다른 하나는 DBMS_SQL 패키지의 프로시저를 호출하는 것입니다. 소스 Oracle 데이터베이스에 동적 SQL이 포함된 객체가 있는 경우 AWS SCT를 사용하여 Oracle 동적 SQL 문을 PostgreSQL로 변환합니다.

Oracle 동적 SQL을 PostgreSQL로 변환하려면

1. Oracle을 소스로 사용하여 AWS SCT 프로젝트를 엽니다.
2. Oracle 소스 트리 보기에서 동적 SQL을 사용하는 데이터베이스 객체를 선택합니다.
3. 객체에 대한 컨텍스트(마우스 오른쪽 버튼 클릭) 메뉴를 열고 스키마 변환을 선택한 후 객체(있는 경우)를 교체하는 데 동의합니다. 아래 스크린샷에는 동적 SQL을 사용하는 Oracle 프로시저 아래에 변환된 프로시저가 나와 있습니다.



Oracle 파티션 변환

AWS SCT에서는 현재 다음과 같은 파티셔닝 방법을 지원합니다.

- Range
- 목록
- 다중 열 범위
- 해시
- 복합(list-list, range-list, list-range, list-hash, range-hash, hash-hash)

Oracle을 Amazon RDS for MySQL 또는 Amazon Aurora MySQL로 변환

변환된 MySQL 코드에서 Oracle 데이터베이스 함수를 에뮬레이션하려면 AWS SCT에서 Oracle-MySQL 확장 팩을 사용합니다. 확장 팩에 대한 자세한 내용은 AWS SCT 확장 팩 사용 섹션을 참조하세요.

주제

- MySQL을 대상 데이터베이스로 사용하기 위한 권한
- Oracle에서 MySQL로의 변환 설정
- 마이그레이션 고려 사항
- Oracle의 WITH 문을 RDS for MySQL 또는 Amazon Aurora MySQL로 변환

MySQL을 대상 데이터베이스로 사용하기 위한 권한

MySQL을 대상으로 사용하기 위해 필요한 권한은 다음과 같습니다.

- CREATE ON *.*
- ALTER ON *.*
- DROP ON *.*
- INDEX ON *.*
- REFERENCES ON *.*
- SELECT ON *.*
- CREATE VIEW ON *.*
- SHOW VIEW ON *.*
- TRIGGER ON *.*
- CREATE ROUTINE ON *.*
- ALTER ROUTINE ON *.*

- EXECUTE ON *.*
- CREATE TEMPORARY TABLES ON *.*
- AWS_LAMBDA_ACCESS
- INSERT, UPDATE ON AWS_ORACLE_EXT.*
- INSERT, UPDATE, DELETE ON AWS_ORACLE_EXT_DATA.*

MySQL 데이터베이스 버전 5.7 이하를 대상으로 사용하는 경우 AWS_LAMBDA_ACCESS 대신 I INVOKE LAMBDA *.* 권한을 부여합니다. MySQL 데이터베이스 버전 8.0 이상의 경우 AWS_LAMBDA_ACCESS 권한을 부여합니다.

다음 코드 예제를 사용하여 데이터베이스 사용자를 생성하고 권한을 부여할 수 있습니다.

```
CREATE USER 'user_name' IDENTIFIED BY 'your_password';
GRANT CREATE ON *.* TO 'user_name';
GRANT ALTER ON *.* TO 'user_name';
GRANT DROP ON *.* TO 'user_name';
GRANT INDEX ON *.* TO 'user_name';
GRANT REFERENCES ON *.* TO 'user_name';
GRANT SELECT ON *.* TO 'user_name';
GRANT CREATE VIEW ON *.* TO 'user_name';
GRANT SHOW VIEW ON *.* TO 'user_name';
GRANT TRIGGER ON *.* TO 'user_name';
GRANT CREATE ROUTINE ON *.* TO 'user_name';
GRANT ALTER ROUTINE ON *.* TO 'user_name';
GRANT EXECUTE ON *.* TO 'user_name';
GRANT CREATE TEMPORARY TABLES ON *.* TO 'user_name';
GRANT AWS_LAMBDA_ACCESS TO 'user_name';
GRANT INSERT, UPDATE ON AWS_ORACLE_EXT.* TO 'user_name';
GRANT INSERT, UPDATE, DELETE ON AWS_ORACLE_EXT_DATA.* TO 'user_name';
```

이전 예제에서 *user_name*을 사용자 이름으로 바꿉니다. 그런 다음 *your_password*를 안전한 암호로 바꿉니다.

MySQL 데이터베이스 버전 5.7 이하를 대상으로 사용하는 경우 `GRANT AWS_LAMBDA_ACCESS TO 'user_name'` 대신 `GRANT INVOKE LAMBDA ON *.* TO 'user_name'`을 사용합니다.

Amazon RDS for MySQL 또는 Aurora MySQL을 대상으로 사용하려면 `lower_case_table_names` 파라미터를 1로 설정합니다. 이 값은 MySQL 서버가 테이블, 인덱스, 트리거 및 데이터베이스와 같은 객체 이름의 식별자를 대소문자 구분 없이 처리한다는 것을 의미합니다. 대상 인스턴스에서 이진 로깅을 활성화했다면 `log_bin_trust_function_creators` 파라미터를 1로 설정합니다. 이 경우 저장

된 함수를 생성하기 위해 DETERMINISTIC, READS SQL DATA 또는 NO SQL 특성을 사용할 필요가 없습니다. 이들 파라미터를 구성하려면 새 DB 파라미터 그룹을 생성하거나 기존 DB 파라미터 그룹을 수정해야 합니다.

Oracle에서 MySQL로의 변환 설정

Oracle에서 MySQL로의 변환 설정을 편집하려면 AWS SCT에서 설정을 선택한 다음 변환 설정을 선택합니다. 상단 목록에서 Oracle을 선택한 다음 Oracle – MySQL을 선택합니다. AWS SCT는 Oracle에서 MySQL로의 변환에 사용할 수 있는 모든 설정을 표시합니다.

AWS SCT의 경우, Oracle에서 MySQL로의 변환 설정에는 다음에 대한 옵션이 포함됩니다.

- 변환된 코드에서 작업 항목이 포함된 설명의 수를 제한합니다.

  Add comments in the converted code for the action items of selected severity and higher에서 작업 항목의 심각도를 선택합니다. AWS SCT가 선택된 심각도 이상의 작업 항목에 대한 변환된 코드에 설명을 추가합니다.

  예를 들어, 변환된 코드의 설명 수를 최소화하려면 오류만을 선택합니다. 변환된 코드의 모든 작업 항목에 대한 설명을 포함하려면 모든 메시지를 선택합니다.
- 소스 Oracle 데이터베이스가 ROWID 의사 열을 사용할 수 있지만 MySQL이 유사한 기능을 지원하지 않는 문제를 해결합니다. AWS SCT는 ROWID 의사 열을 변환된 코드로 에뮬레이션할 수 있습니다. 이렇게 하려면 행 ID 생성?에서 Generate as identity를 선택합니다.

  소스 Oracle 코드에서 ROWID 의사 열을 사용하지 않는 경우 행 ID 생성?에서 Don't generate를 선택합니다. 이 경우, 변환된 코드는 더 빠르게 작동합니다.
- MySQL에서 지원하지 않는 파라미터가 있는 TO_CHAR, TO_DATE 및 TO_NUMBER 함수를 포함하는 경우 소스 Oracle 코드를 사용하여 작업합니다. 기본적으로 AWS SCT는 이러한 파라미터의 사용을 변환된 코드로 에뮬레이션합니다.

  소스 Oracle 코드에 PostgreSQL이 지원하는 파라미터만 포함되어 있는 경우 기본 MySQL TO_CHAR, TO_DATE 및 TO_NUMBER 함수를 사용할 수 있습니다. 이 경우, 변환된 코드는 더 빠르게 작동합니다. 이러한 파라미터만 포함하려면 다음 값을 선택합니다.
  - Function TO_CHAR() does not use Oracle specific formatting strings
  - Function TO_DATE() does not use Oracle specific formatting strings
  - Function TO_NUMBER() does not use Oracle specific formatting strings
- 데이터베이스와 애플리케이션이 서로 다른 시간대에서 실행되는지 여부를 확인합니다. 기본적으로 AWS SCT는 변환된 코드로 시간대를 에뮬레이션합니다. 하지만 데이터베이스와 애플리케이션이

동일한 시간대를 사용하는 경우에는 이 에뮬레이션이 필요하지 않습니다. 이 경우 Time zone on the client side matches the time zone on server를 선택합니다.

마이그레이션 고려 사항

Oracle을 RDS for MySQL 또는 Aurora MySQL로 변환할 때 명령문이 실행되는 순서를 변경하려면 GOTO 문과 레이블을 사용할 수 있습니다. GOTO 문 뒤에 오는 PL/SQL 문은 건너뛰며 프로세스는 레이블에서 계속됩니다. GOTO 문과 레이블은 프로시저, 배치(batch), 명령문 블록 내 어디든 사용할 수 있습니다. 또한 다음 GOTO 문에도 사용할 수 있습니다.

MySQL은 GOTO 문을 사용하지 않습니다. AWS SCT는 GOTO 문을 포함하는 코드를 변환할 때 BEGIN…END 또는 LOOP…END LOOP 문을 사용하도록 명령문을 변환합니다.

다음 테이블에는 AWS SCT가 GOTO 문을 어떻게 변환하는지에 대한 예가 나와 있습니다.

Oracle 문	MySQL 문

BEGIN
....
statement1;
....
GOTO label1;
statement2;
....
label1:
Statement3;
....
END
	BEGIN
	label1:
	BEGIN
	....
	statement1;
	....
	LEAVE label1;
	statement2;
	....
	END;
	Statement3;
	....
	END

BEGIN .... statement1; .... label1: statement2; .... GOTO label1;	BEGIN .... statement1; .... label1: LOOP statement2; ....

Oracle 문	MySQL 문
statement3; .... statement4; .... END	ITERATE label1; LEAVE label1; END LOOP; statement3; .... statement4; .... END

BEGIN
....
statement1;
....
label1:
statement2;
....
statement3;
....
statement4;
....
END
	BEGIN
	....
	statement1;
	....
	label1:
	BEGIN
	statement2;
	....
	statement3;
	....
	statement4;
	....
	END;
	END

Oracle의 WITH 문을 RDS for MySQL 또는 Amazon Aurora MySQL로 변환

Oracle의 WITH 절(subquery_factoring)을 사용하여 서브쿼리 블록에 이름(쿼리 이름)을 할당할 수 있습니다. 그런 다음 쿼리 이름을 지정하여 이 서브쿼리 블록을 쿼리의 여러 위치에서 참조할 수 있습니다. 서브쿼리 블록에는 링크나 파라미터(local, procedure, function, package)가 없으므로 AWS SCT는 이 절을 보기 또는 임시 테이블로 변환합니다.

이 절을 임시 테이블로 변환하면 서브쿼리에 대한 반복된 참조를 보다 효율화할 수 있습니다. 그 이유는 데이터를 각 참조에서 요구하지 않고 임시 테이블에서 데이터를 쉽게 검색할 수 있기 때문입니다. 보기 또는 임시 테이블을 추가로 사용하여 이를 에뮬레이션할 수 있습니다. 뷰 이름은 <procedure_name>$<subselect_alias> 형식을 사용합니다.

다음 테이블에서 예제를 찾을 수 있습니다.

Oracle 문	MySQL 문

```
CREATE PROCEDURE
 TEST_ORA_PG.P_WITH_SELECT_V
ARIABLE_01
     (p_state IN NUMBER)
AS
  l_dept_id NUMBER := 1;
BEGIN
FOR cur IN
           (WITH dept_empl(id, name,
 surname,
              lastname, state, dept_id)
              AS
                  (
                    SELECT id, name,
 surname,
                     lastname, state,
 dept_id
                      FROM test_ora_
pg.dept_employees
                     WHERE state =
 p_state AND
                       dept_id =
 l_dept_id)
            SELECT id,state
              FROM dept_empl
            ORDER BY id)  LOOP
  NULL;
END LOOP;
```

```
CREATE PROCEDURE test_ora_pg.P_WITH
_SELECT_VARIABLE_01(IN par_P_STATE
 DOUBLE)
BEGIN
    DECLARE var_l_dept_id DOUBLE
 DEFAULT 1;
    DECLARE var$id VARCHAR (8000);
    DECLARE var$state VARCHAR (8000);
    DECLARE done INT DEFAULT FALSE;
    DECLARE cur CURSOR FOR SELECT
        ID, STATE
        FROM (SELECT
            ID, NAME, SURNAME,
 LASTNAME, STATE, DEPT_ID
            FROM TEST_ORA_PG.DEPT_E
MPLOYEES
            WHERE STATE = par_p_sta
te AND DEPT_ID = var_l_dept_id) AS
 dept_empl
        ORDER BY ID;
    DECLARE CONTINUE HANDLER FOR NOT
 FOUND
        SET done := TRUE;
    OPEN cur;

    read_label:
    LOOP
        FETCH cur INTO var$id, var
$state;

        IF done THEN
            LEAVE read_label;
        END IF;

        BEGIN
        END;
    END LOOP;
    CLOSE cur;
END;
```

Oracle 문

```
CREATE PROCEDURE
 TEST_ORA_PG.P_WITH_SELECT_R
EGULAR_MULT_01
AS
BEGIN

 FOR cur IN  (
               WITH dept_empl AS
                   (
                        SELECT id,
 name, surname,
                         lastname,
 state, dept_id
                          FROM
 test_ora_pg.dept_employees
                         WHERE state =
 1),
                    dept AS
                   (SELECT id deptid,
 parent_id,
                      name deptname
                      FROM test_ora_
pg.department
                   )
                SELECT dept_empl
.*,dept.*
                 FROM dept_empl, dept
                 WHERE dept_empl
.dept_id = dept.deptid
              ) LOOP
              NULL;
            END LOOP;
```

MySQL 문

```
CREATE VIEW TEST_ORA_PG.`P_WIT
H_SELECT_REGULAR_MULT_01$dept_empl
 `(id, name, surname, lastname, state,
 dept_id)
AS
(SELECT id, name, surname, lastname,
 state, dept_id
   FROM test_ora_pg.dept_employees
  WHERE state = 1);

CREATE VIEW TEST_ORA_PG.`P_WIT
H_SELECT_REGULAR_MULT_01$dept
 `(deptid, parent_id,deptname)
AS
(SELECT id deptid, parent_id, name
 deptname
   FROM test_ora_pg.department);


CREATE PROCEDURE test_ora_pg.P_WITH
_SELECT_REGULAR_MULT_01()
BEGIN
    DECLARE var$ID DOUBLE;
    DECLARE var$NAME VARCHAR (30);
    DECLARE var$SURNAME VARCHAR (30);
    DECLARE var$LASTNAME VARCHAR (30);
    DECLARE var$STATE DOUBLE;
    DECLARE var$DEPT_ID DOUBLE;
    DECLARE var$deptid DOUBLE;
    DECLARE var$PARENT_ID DOUBLE;
    DECLARE var$deptname VARCHAR
 (200);
    DECLARE done INT DEFAULT FALSE;
    DECLARE cur CURSOR FOR SELECT
        dept_empl.*, dept.*
        FROM TEST_ORA_PG.`P_WIT
H_SELECT_REGULAR_MULT_01$dept_empl
          ` AS dept_empl,
            TEST_ORA_PG.`P_WIT
H_SELECT_REGULAR_MULT_01$dept
          ` AS dept
```

Oracle 문	MySQL 문

	WHERE dept_empl.DEPT_ID = dept.DEPTID;
	DECLARE CONTINUE HANDLER FOR NOT FOUND
	SET done := TRUE;
	OPEN cur;
	
	read_label:
	LOOP
	FETCH cur INTO varID, varNAME, var$SURNAME,
	var$LASTNAME, var$STATE, var$DEPT_ID, var$deptid,
	var$PARENT_ID, var$deptname;
	
	IF done THEN
	LEAVE read_label;
	END IF;
	
	BEGIN
	END;
	END LOOP;
	CLOSE cur;
	END;
	
	call test_ora_pg.P_WITH_SELECT_REGULAR_MULT_01()

Oracle 문

```
CREATE PROCEDURE
  TEST_ORA_PG.P_WITH_SELECT_V
AR_CROSS_02(p_state IN NUMBER)
AS
   l_dept_id NUMBER := 10;
BEGIN
 FOR cur IN  (
               WITH emp AS

                    (SELECT id, name,
 surname,
                      lastname, state,
 dept_id
                       FROM test_ora_
pg.dept_employees
                      WHERE dept_id >
 10
                    ),
                    active_emp AS
                    (
                      SELECT id
                        FROM emp
                       WHERE emp.state
 = p_state
                    )

                SELECT *
                  FROM active_emp

             ) LOOP
        NULL;
  END LOOP;

END;
```

MySQL 문

```
CREATE VIEW TEST_ORA_PG.`P_WIT
H_SELECT_VAR_CROSS_01$emp
    `(id, name, surname, lastname,
 state, dept_id)
AS
(SELECT
       id, name, surname, lastname,
       state, dept_id
  FROM TEST_ORA_PG.DEPT_EMPLOYEES
  WHERE DEPT_ID > 10);


CREATE PROCEDURE
   test_ora_pg.P_WITH_SELECT_V
AR_CROSS_02(IN par_P_STATE DOUBLE)
BEGIN
    DECLARE var_l_dept_id DOUBLE
 DEFAULT 10;
    DECLARE var$ID DOUBLE;
    DECLARE done INT DEFAULT FALSE;
    DECLARE cur CURSOR FOR SELECT *
                               FROM
 (SELECT

 ID

 FROM
                               TEST_ORA_
PG.
                                `P_WITH_S
ELECT_VAR_CROSS_01$emp` AS emp

 WHERE emp.STATE = par_p_state)
                                     AS
 active_emp;
    DECLARE CONTINUE HANDLER FOR NOT
 FOUND
        SET done := TRUE;
    OPEN cur;

    read_label:
```

Oracle 문	MySQL 문

	LOOP
	FETCH cur INTO var$ID;
	
	IF done THEN
	LEAVE read_label;
	END IF;
	
	BEGIN
	END;
	END LOOP;
	CLOSE cur;
	END;

Oracle에서 Amazon RDS for Oracle로 변환

Oracle 스키마와 코드를 Amazon RDS for Oracle로 마이그레이션할 경우 몇 가지 사항을 고려해야 합니다.

- AWS SCT가 객체 트리에 디렉터리 객체를 추가할 수 있습니다. 디렉터리 객체는 서버 파일 시스템의 물리적 디렉터리를 나타내는 논리적 구조입니다. DBMS_LOB, UTL_FILE, DBMS_FILE_TRANSFER, DATAPUMP 유틸리티 등과 같은 패키지를 통해 디렉터리 객체를 사용할 수 있습니다.
- AWS SCT는 Oracle 테이블스페이스를 Amazon RDS for Oracle DB 인스턴스로 변환하도록 지원합니다. Oracle은 데이터를 테이블스페이스에 논리적으로 저장하고, 해당 테이블스페이스와 연결된 데이터 파일에 물리적으로 저장합니다. Oracle에서는 데이터 파일 이름을 사용하여 테이블스페이스를 만들 수 있습니다. Amazon RDS는 데이터 파일, 로그 파일 및 제어 파일에 대해서만 Oracle Managed Files(OMF)를 지원합니다. AWS SCT는 변환 중에 필요한 데이터 파일을 생성합니다.
- AWS SCT는 서버 수준 역할과 권한을 변환할 수 있습니다. Oracle 데이터베이스 엔진에는 역할 기반 보안이 사용됩니다. 역할이란 사용자에 대해 부여하거나 취소할 수 있는 권한 모음입니다. Amazon RDS의 사전 정의된 역할인 DBA는 일반적으로 Oracle 데이터베이스 엔진에 대한 모든 관리 권한을 허용합니다. 아래의 권한들은 Oracle 엔진을 사용하는 Amazon RDS DB 인스턴스의 DBA 역할에는 사용할 수 없습니다.
  - 데이터베이스 변경
  - 시스템 변경
  - 디렉터리 생성

  - 권한 부여
  - 역할 부여
  - 외부 작업 생성

  Amazon RDS for Oracle 사용자 역할에 고급 필터링 및 열 권한을 포함하여 다른 모든 권한을 부여할 수 있습니다.

- AWS SCT는 Oracle 작업을 Amazon RDS for Oracle에서 실행할 수 있는 작업으로 변환하도록 지원합니다. 이 변환에는 다음과 같은 몇 가지 제한 사항이 있습니다.
  - 실행 작업은 지원되지 않습니다.
  - ANYDATA 데이터 유형을 인수로 사용하는 일정 작업은 지원되지 않습니다.
- Oracle 실제 애플리케이션 클러스터(RAC) 단일 노드는 Oracle Database 11g Release 2와 함께 제공된 Oracle Database Enterprise Edition에 속한 옵션입니다. Amazon RDS for Oracle은 RAC 기능을 지원하지 않습니다. 고가용성을 위해서는 Amazon RDS 다중 AZ를 사용하십시오.

  다중 AZ 배포에서 Amazon RDS는 자동으로 서로 다른 가용 영역에 동기식 예비 복제본을 프로비저닝하고 유지합니다. 기본 DB 인스턴스는 가용 영역 전체에서 대기 복제본으로 동기식으로 복제됩니다. 이 기능은 데이터 중복을 제공하고, I/O 중지를 없애고, 시스템 백업 중에 지연 시간 스파이크를 최소화합니다.
- Oracle Spatial에는 Oracle 데이터베이스에서 공간 데이터의 저장, 검색, 업데이트 및 쿼리를 신속하게 실행할 수 있는 SQL 스키마 및 기능이 있습니다. Oracle Locator에는 인터넷 및 무선 서비스 기반 애플리케이션과 파트너 기반 GIS 솔루션을 지원할 때 필요한 기능이 있습니다. Oracle Locator는 Oracle Spatial의 제한된 서브셋입니다.

  Oracle Spatial 및 Oracle Locator 기능을 사용하려면 SPATIAL 옵션 또는 LOCATOR 옵션(함께 사용할 수 없음)을 DB 인스턴스의 옵션 그룹에 추가합니다.

  Amazon RDS for Oracle DB 인스턴스에서 Oracle Spatial 및 Oracle Locator를 사용하기 위해서는 몇 가지 전제 조건이 있습니다.

  - 인스턴스가 Oracle Enterprise Edition 버전 12.1.0.2.v6 이상 또는 11.2.0.4.v10 이상을 사용해야 합니다.
  - 인스턴스가 VPC(Virtual Private Cloud) 내에 있어야 합니다.
  - 인스턴스가 Oracle 기능을 지원할 수 있는 DB 인스턴스 클래스를 사용해야 합니다. 예를 들어 db.m1.small, db.t1.micro, db.t2.micro 또는 db.t2.small DB 인스턴스 클래스에는 Oracle Spatial이 지원되지 않습니다. 자세한 내용은 Oracle을 위한 DB 인스턴스 클래스 지원을 참조하세요.

  - 인스턴스에 대해 자동 마이너 버전 업그레이드를 활성화해야 합니다. CVSS 점수가 9 이상인 보안 취약성 또는 발표된 기타 보안 취약성이 있을 경우 Amazon RDS는 DB 인스턴스를 최신 Oracle PSU로 업데이트합니다. 자세한 내용을 알아보려면 다음 섹션을 참조하세요.

    Oracle DB 인스턴스 설정
  - DB 인스턴스가 버전 11.2.0.4.v10 이상인 경우 XMLDB 옵션을 설치해야 합니다. 자세한 내용을 알아보려면 다음 섹션을 참조하세요.

    Oracle XML DB
  - Oracle의 Oracle Spatial 라이선스가 있어야 합니다. 자세한 내용은 Oracle 설명서의 Oracle Spatial and Graph를 참조하십시오.
- Data Guard는 Oracle Database Enterprise Edition에 포함되어 있습니다. 고가용성을 위해서는 Amazon RDS 다중 AZ 기능을 사용하십시오.

  다중 AZ 배포에서 Amazon RDS는 자동으로 서로 다른 가용 영역에 동기식 예비 복제본을 프로비저닝하고 유지합니다. 기본 DB 인스턴스는 가용 영역 전체에서 대기 복제본으로 동기식으로 복제됩니다. 이 기능은 데이터 중복을 제공하고, I/O 중지를 없애고, 시스템 백업 중에 지연 시간 스파이크를 최소화합니다.
- AWS SCT는 Amazon RDS for Oracle로 마이그레이션할 때 Oracle DBMS_SCHEDULER 객체 변환을 지원합니다. AWS SCT 평가 보고서는 일정 객체를 변환할 수 있는지 여부를 표시합니다. Amazon RDS에서 일정 객체를 사용하는 방법에 대한 자세한 내용은 Amazon RDS 설명서 를 참조하십시오.
- Oracle에서 Oracle용 Amazon RDS로 변환하는 경우 DB Links를 지원합니다. 데이터베이스 링크는 특정 데이터베이스 내 스키마 객체로서, 사용자는 이 객체를 통해 다른 데이터베이스에 있는 객체에 액세스할 수 있습니다. 다른 데이터베이스가 Oracle 데이터베이스이어야 할 필요는 없습니다. 하지만 Oracle 데이터베이스가 아닌 데이터베이스에 액세스하려면 Oracle Heterogeneous Services를 사용해야 합니다.

  데이터베이스 링크를 생성하면 이 링크를 SQL 문에서 사용하여 다른 데이터베이스에 있는 테이블, 보기 및 PL/SQL 객체를 참조할 수 있습니다. 데이터베이스 링크를 사용하려면 테이블, 보기 또는 PL/SQL 객체 이름에 `@dblink`를 붙입니다. SELECT 문을 사용해 다른 데이터베이스에 있는 테이블 또는 보기를 쿼리할 수 있습니다. Oracle 데이터베이스 링크를 사용하는 방법에 대한 자세한 정보는 Oracle 설명서를 참조하십시오.

  Amazon RDS에서 데이터베이스 링크를 사용하는 방법에 대한 자세한 정보는 Amazon RDS 설명서를 참조하십시오.

- AWS SCT 평가 보고서에서 변환을 위한 서버 측정치를 제공합니다. Oracle 인스턴스에 대한 이러한 측정치에는 다음이 포함됩니다.
  - 대상 DB 인스턴스의 컴퓨팅 및 메모리 용량
  - 지원되지 않는 Oracle 기능(예: Amazon RDS가 지원하지 않는 Real Application Clusters 등)
  - 디스크 읽기/쓰기 로드
  - 평균 총 디스크 처리량
  - 서버 정보(서버 이름, OS, 호스트 이름, 문자 집합 등)

RDS for Oracle을 대상으로 사용하기 위한 권한

Amazon RDS for Oracle로 마이그레이션하려면 권한이 있는 데이터베이스 사용자를 생성합니다. 다음과 같은 코드 예제를 사용할 수 있습니다.

```
CREATE USER user_name IDENTIFIED BY your_password;

-- System privileges
GRANT DROP ANY CUBE BUILD PROCESS TO user_name;
GRANT ALTER ANY CUBE TO user_name;
GRANT CREATE ANY CUBE DIMENSION TO user_name;
GRANT CREATE ANY ASSEMBLY TO user_name;
GRANT ALTER ANY RULE TO user_name;
GRANT SELECT ANY DICTIONARY TO user_name;
GRANT ALTER ANY DIMENSION TO user_name;
GRANT CREATE ANY DIMENSION TO user_name;
GRANT ALTER ANY TYPE TO user_name;
GRANT DROP ANY TRIGGER TO user_name;
GRANT CREATE ANY VIEW TO user_name;
GRANT ALTER ANY CUBE BUILD PROCESS TO user_name;
GRANT CREATE ANY CREDENTIAL TO user_name;
GRANT DROP ANY CUBE DIMENSION TO user_name;
GRANT DROP ANY ASSEMBLY TO user_name;
GRANT DROP ANY PROCEDURE TO user_name;
GRANT ALTER ANY PROCEDURE TO user_name;
GRANT ALTER ANY SQL TRANSLATION PROFILE TO user_name;
GRANT DROP ANY MEASURE FOLDER TO user_name;
GRANT CREATE ANY MEASURE FOLDER TO user_name;
GRANT DROP ANY CUBE TO user_name;
GRANT DROP ANY MINING MODEL TO user_name;
GRANT CREATE ANY MINING MODEL TO user_name;
GRANT DROP ANY EDITION TO user_name;
```

```
GRANT CREATE ANY EVALUATION CONTEXT TO user_name;
GRANT DROP ANY DIMENSION TO user_name;
GRANT ALTER ANY INDEXTYPE TO user_name;
GRANT DROP ANY TYPE TO user_name;
GRANT CREATE ANY PROCEDURE TO user_name;
GRANT CREATE ANY SQL TRANSLATION PROFILE TO user_name;
GRANT CREATE ANY CUBE TO user_name;
GRANT COMMENT ANY MINING MODEL TO user_name;
GRANT ALTER ANY MINING MODEL TO user_name;
GRANT DROP ANY SQL PROFILE TO user_name;
GRANT CREATE ANY JOB TO user_name;
GRANT DROP ANY EVALUATION CONTEXT TO user_name;
GRANT ALTER ANY EVALUATION CONTEXT TO user_name;
GRANT CREATE ANY INDEXTYPE TO user_name;
GRANT CREATE ANY OPERATOR TO user_name;
GRANT CREATE ANY TRIGGER TO user_name;
GRANT DROP ANY ROLE TO user_name;
GRANT DROP ANY SEQUENCE TO user_name;
GRANT DROP ANY CLUSTER TO user_name;
GRANT DROP ANY SQL TRANSLATION PROFILE TO user_name;
GRANT ALTER ANY ASSEMBLY TO user_name;
GRANT CREATE ANY RULE SET TO user_name;
GRANT ALTER ANY OUTLINE TO user_name;
GRANT UNDER ANY TYPE TO user_name;
GRANT CREATE ANY TYPE TO user_name;
GRANT DROP ANY MATERIALIZED VIEW TO user_name;
GRANT ALTER ANY ROLE TO user_name;
GRANT DROP ANY VIEW TO user_name;
GRANT ALTER ANY INDEX TO user_name;
GRANT COMMENT ANY TABLE TO user_name;
GRANT CREATE ANY TABLE TO user_name;
GRANT CREATE USER TO user_name;
GRANT DROP ANY RULE SET TO user_name;
GRANT CREATE ANY CONTEXT TO user_name;
GRANT DROP ANY INDEXTYPE TO user_name;
GRANT ALTER ANY OPERATOR TO user_name;
GRANT CREATE ANY MATERIALIZED VIEW TO user_name;
GRANT ALTER ANY SEQUENCE TO user_name;
GRANT DROP ANY SYNONYM TO user_name;
GRANT CREATE ANY SYNONYM TO user_name;
GRANT DROP USER TO user_name;
GRANT ALTER ANY MEASURE FOLDER TO user_name;
GRANT ALTER ANY EDITION TO user_name;
GRANT DROP ANY RULE TO user_name;
```

```
GRANT CREATE ANY RULE TO user_name;
GRANT ALTER ANY RULE SET TO user_name;
GRANT CREATE ANY OUTLINE TO user_name;
GRANT UNDER ANY TABLE TO user_name;
GRANT UNDER ANY VIEW TO user_name;
GRANT DROP ANY DIRECTORY TO user_name;
GRANT ALTER ANY CLUSTER TO user_name;
GRANT CREATE ANY CLUSTER TO user_name;
GRANT ALTER ANY TABLE TO user_name;
GRANT CREATE ANY CUBE BUILD PROCESS TO user_name;
GRANT ALTER ANY CUBE DIMENSION TO user_name;
GRANT CREATE ANY EDITION TO user_name;
GRANT CREATE ANY SQL PROFILE TO user_name;
GRANT ALTER ANY SQL PROFILE TO user_name;
GRANT DROP ANY OUTLINE TO user_name;
GRANT DROP ANY CONTEXT TO user_name;
GRANT DROP ANY OPERATOR TO user_name;
GRANT DROP ANY LIBRARY TO user_name;
GRANT ALTER ANY LIBRARY TO user_name;
GRANT CREATE ANY LIBRARY TO user_name;
GRANT ALTER ANY MATERIALIZED VIEW TO user_name;
GRANT ALTER ANY TRIGGER TO user_name;
GRANT CREATE ANY SEQUENCE TO user_name;
GRANT DROP ANY INDEX TO user_name;
GRANT CREATE ANY INDEX TO user_name;
GRANT DROP ANY TABLE TO user_name;
GRANT SELECT_CATALOG_ROLE TO user_name;
GRANT SELECT ANY SEQUENCE TO user_name;

-- Database Links
GRANT CREATE DATABASE LINK TO user_name;
GRANT CREATE PUBLIC DATABASE LINK TO user_name;
GRANT DROP PUBLIC DATABASE LINK TO user_name;


-- Server Level Objects (directory)
GRANT CREATE ANY DIRECTORY TO user_name;
GRANT DROP ANY DIRECTORY TO user_name;
-- (for RDS only)
GRANT EXECUTE ON RDSADMIN.RDSADMIN_UTIL TO user_name;

-- Server Level Objects (tablespace)
GRANT CREATE TABLESPACE TO user_name;
GRANT DROP TABLESPACE TO user_name;
```

```
-- Server Level Objects (user roles)
/* (grant source privileges with admin option or convert roles/privs as DBA) */

-- Queues
grant execute on DBMS_AQADM to user_name;
grant aq_administrator_role to user_name;

-- for Materialized View Logs creation
GRANT SELECT ANY TABLE TO user_name;

-- Roles
GRANT RESOURCE TO user_name;
GRANT CONNECT TO user_name;
```

이전 예제에서 *user_name*을 사용자 이름으로 바꿉니다. 그런 다음 *your_password*를 안전한 암호로 바꿉니다.

Oracle에서 Amazon RDS for Oracle로 변환할 때 제한 사항

Oracle 스키마와 코드를 Amazon RDS for Oracle로 마이그레이션할 경우 몇 가지 제한 사항을 고려해야 합니다.

- Amazon RDS의 사전 정의된 역할인 DBA는 일반적으로 Oracle 데이터베이스 엔진에 대한 모든 관리 권한을 허용합니다. 아래의 권한들은 Oracle 엔진을 사용하는 Amazon RDS DB 인스턴스의 DBA 역할에는 사용할 수 없습니다.
  - 데이터베이스 변경
  - 시스템 변경
  - 디렉터리 생성
  - 권한 부여
  - 역할 부여
  - 외부 작업 생성

  다른 모든 권한을 Oracle RDS 사용자 역할에 부여할 수 있습니다.
- Amazon RDS for Oracle은 기존 방식의 감사, DBMS_FGA 패키지를 사용한 세분화된 감사 및 Oracle Unified Auditing을 지원합니다.
- Amazon RDS for Oracle은 CDC(변경 데이터 캡처)를 지원하지 않습니다. 데이터 마이그레이션 도중이나 그 이후에 CDC를 수행하려면 AWS Database Migration Service를 사용합니다.

PostgreSQL을 AWS SCT에 대한 소스로 사용

AWS SCT를 사용하여 스키마, 데이터베이스 코드 객체 및 애플리케이션 코드를 PostgreSQL에서 다음 대상으로 변환할 수 있습니다.

- Amazon RDS for MySQL
- Amazon Aurora MySQL-Compatible Edition
- Amazon RDS for PostgreSQL
- Amazon Aurora PostgreSQL-Compatible Edition

자세한 내용은 다음 단원을 참조하세요.

주제

- PostgreSQL을 소스 데이터베이스로 사용할 수 있는 권한
- PostgreSQL에 소스로 연결
- MySQL을 대상 데이터베이스로 사용하기 위한 권한

PostgreSQL을 소스 데이터베이스로 사용할 수 있는 권한

PostgreSQL을 소스로 사용하기 위해 필요한 권한은 다음과 같습니다.

- CONNECT ON DATABASE *<database_name>*
- USAGE ON SCHEMA *<database_name>*
- SELECT ON ALL TABLES IN SCHEMA *<database_name>*
- SELECT ON ALL SEQUENCES IN SCHEMA *<database_name>*

PostgreSQL에 소스로 연결

다음 절차에 따라 AWS Schema Conversion Tool을 사용하여 PostgreSQL 소스 데이터베이스에 연결합니다.

PostgreSQL 소스 데이터베이스에 연결하려면

1. AWS Schema Conversion Tool에서 소스 추가를 선택합니다.
2. PostgreSQL을 선택한 후 다음을 선택합니다.

소스 추가 대화 상자가 나타납니다.

3. 연결 이름에 데이터베이스의 이름을 입력합니다. AWS SCT는 왼쪽 패널의 트리에 이 이름을 표시합니다.
4. AWS Secrets Manager의 데이터베이스 보안 인증 정보를 사용하거나 수동으로 입력합니다.

   - Secrets Manager의 데이터베이스 보안 인증 정보를 사용하려면 다음 지침을 따릅니다.

     1. AWS Secret에서 보안 암호의 이름을 선택합니다.
     2. Populate를 선택하여 Secrets Manager에서 데이터베이스 연결 대화 상자에 있는 모든 값을 자동으로 채웁니다.

     Secrets Manager의 데이터베이스 보안 인증 사용에 대한 자세한 내용은 AWS Secrets Manager 사용 섹션을 참조하세요.
   - PostgreSQL 소스 데이터베이스 연결 정보를 수동으로 입력하려면 다음 지침을 사용합니다.

파라미터	작업
[Server name]	소스 데이터베이스 서버의 도메인 이름 시스템(DNS) 이름 또는 IP 주소를 입력합니다. IPv6 주소 프로토콜을 사용하여 소스 PostgreSQL 데이터베이스에 연결할 수 있습니다. 이렇게 하려면 다음 예제와 같이 대괄호를 사용하여 IP 주소를 입력해야 합니다. `[2001:db8:ffff:ffff:ffff:ffff:ffff:fffe]`
[Server port]	소스 데이터베이스 서버에 연결하는 데 사용되는 포트를 입력합니다.
데이터베이스	PostgreSQL 데이터베이스의 이름을 입력합니다.
User name 및 Password	소스 데이터베이스 서버에 연결하는 데 필요한 데이터베이스 보안 인증 정보를 입력합니다. 프로젝트에서 데이터베이스에 연결하도록 선택한 경우에만 AWS SCT에서 암호를 사용하여 소스 데이터베이스에 연결합니다. 소스 데이터베이스의 암호가 노출될 위험을 방지하기 위해 AWS SCT는 기본적으로 암호를 저장하지 않습니다.

파라미터	작업
	AWS SCT 프로젝트를 닫았다 다시 열 경우 필요에 따라 소스 데이터베이스에 연결하기 위한 암호를 입력하라는 메시지가 표시됩니다.
Use SSL(SSL 사용)	SSL(Secure Sockets Layer)을 사용하여 데이터베이스에 연결하려면 이 옵션을 선택합니다. SSL 탭에서 다음 추가 정보를 적절히 제공합니다. • Verify server certificate: 트러스트 스토어를 사용하여 서버 인증서를 확인하려면 이 옵션을 선택합니다. • 트러스트 스토어: 인증서가 있는 트러스트 스토어의 위치입니다. 이 위치가 전역 설정 섹션에 표시되도록 하려면 해당 위치를 추가해야 합니다.
Store password	AWS SCT는 SSL 인증서와 데이터베이스 암호를 저장할 안전한 볼트를 생성합니다. 이 옵션을 활성화하면 데이터베이스 암호를 저장하고 암호 입력 없이 빠르게 데이터베이스에 연결할 수 있습니다.
PostgreSQL driver path	소스 데이터베이스에 연결할 때 사용할 드라이버의 경로를 입력합니다. 자세한 내용은 필수 데이터베이스 드라이버 다운로드 섹션을 참조하세요. 드라이버 경로를 전역 프로젝트 설정에 저장할 경우 드라이버 경로가 연결 대화 상자에 표시되지 않습니다. 자세한 내용은 전역 설정에 드라이버 경로 저장 섹션을 참조하세요.

5. Test Connection을 선택하여 AWS SCT가 소스 데이터베이스에 연결할 수 있는지 확인합니다.
6. 연결을 선택하여 소스 데이터베이스에 연결합니다.

MySQL을 대상 데이터베이스로 사용하기 위한 권한

PostgreSQL에서 마이그레이션할 때 MySQL을 대상으로 사용하기 위해 필요한 권한은 다음과 같습니다.

- CREATE ON *.*
- ALTER ON *.*

- DROP ON *.*
- INDEX ON *.*
- REFERENCES ON *.*
- SELECT ON *.*
- CREATE VIEW ON *.*
- SHOW VIEW ON *.*
- TRIGGER ON *.*
- CREATE ROUTINE ON *.*
- ALTER ROUTINE ON *.*
- EXECUTE ON *.*
- INSERT, UPDATE ON AWS_POSTGRESQL_EXT.*
- INSERT, UPDATE, DELETE ON AWS_POSTGRESQL_EXT_DATA.*
- CREATE TEMPORARY TABLES ON AWS_POSTGRESQL_EXT_DATA.*

다음 코드 예제를 사용하여 데이터베이스 사용자를 생성하고 권한을 부여할 수 있습니다.

```
CREATE USER 'user_name' IDENTIFIED BY 'your_password';
GRANT CREATE ON *.* TO 'user_name';
GRANT ALTER ON *.* TO 'user_name';
GRANT DROP ON *.* TO 'user_name';
GRANT INDEX ON *.* TO 'user_name';
GRANT REFERENCES ON *.* TO 'user_name';
GRANT SELECT ON *.* TO 'user_name';
GRANT CREATE VIEW ON *.* TO 'user_name';
GRANT SHOW VIEW ON *.* TO 'user_name';
GRANT TRIGGER ON *.* TO 'user_name';
GRANT CREATE ROUTINE ON *.* TO 'user_name';
GRANT ALTER ROUTINE ON *.* TO 'user_name';
GRANT EXECUTE ON *.* TO 'user_name';
GRANT INSERT, UPDATE ON AWS_POSTGRESQL_EXT.* TO 'user_name';
GRANT INSERT, UPDATE, DELETE ON AWS_POSTGRESQL_EXT_DATA.* TO 'user_name';
GRANT CREATE TEMPORARY TABLES ON AWS_POSTGRESQL_EXT_DATA.* TO 'user_name';
```

이전 예제에서 *user_name*을 사용자 이름으로 바꿉니다. 그런 다음 *your_password*를 안전한 암호로 바꿉니다.

Amazon RDS for MySQL 또는 Aurora MySQL을 대상으로 사용하려면 `lower_case_table_names` 파라미터를 1로 설정합니다. 이 값은 MySQL 서버가 테이블, 인덱스, 트리거 및 데이터베이스와 같은 객체 이름의 식별자를 대소문자 구분 없이 처리한다는 것을 의미합니다. 대상 인스턴스에서 이진 로깅을 활성화했다면 `log_bin_trust_function_creators` 파라미터를 1로 설정합니다. 이 경우 저장된 함수를 생성하기 위해 `DETERMINISTIC`, `READS SQL DATA` 또는 `NO SQL` 특성을 사용할 필요가 없습니다. 이들 파라미터를 구성하려면 새 DB 파라미터 그룹을 생성하거나 기존 DB 파라미터 그룹을 수정해야 합니다.

SAP ASE(Sybase ASE)를 AWS SCT에 대한 소스로 사용

AWS SCT를 사용하여 스키마, 데이터베이스 코드 객체 및 애플리케이션 코드를 SAP(Sybase) Adaptive Server Enterprise(ASE)에서 다음 대상으로 변환할 수 있습니다.

- Amazon RDS for MySQL
- Amazon Aurora MySQL-Compatible Edition
- Amazon RDS for MariaDB
- Amazon RDS for PostgreSQL
- Amazon Aurora PostgreSQL-Compatible Edition

자세한 내용은 다음 단원을 참조하세요.

주제

- SAP ASE를 소스 데이터베이스로 사용할 수 있는 권한
- SAP ASE(Sybase) 소스에 연결
- MySQL을 대상 데이터베이스로 사용하기 위한 권한
- SAP ASE에서 MySQL로 변환 설정
- PostgreSQL을 대상 데이터베이스로 사용하기 위한 권한
- SAP ASE에서 PostgreSQL로 변환 설정

SAP ASE를 소스 데이터베이스로 사용할 수 있는 권한

SAP ASE 데이터베이스를 소스로 사용하려면 데이터베이스 사용자를 생성하고 권한을 부여해야 합니다. 이 작업을 수행하려면 다음 단계를 수행합니다.

데이터베이스 사용자 생성 및 구성

1. 소스 데이터베이스에 연결합니다.
2. 다음 명령을 사용하여 데이터베이스 사용자를 생성합니다. 새 사용자의 암호를 입력합니다.

```
USE master
CREATE LOGIN min_privs WITH PASSWORD <password>
sp_adduser min_privs
grant select on dbo.spt_values to min_privs
grant select on asehostname to min_privs
```

3. 마이그레이션하려는 모든 데이터베이스에 대해 다음 권한을 부여합니다.

```
USE <database_name>
sp_adduser min_privs
grant select on dbo.sysusers to min_privs
grant select on dbo.sysobjects to min_privs
grant select on dbo.sysindexes to min_privs
grant select on dbo.syscolumns to min_privs
grant select on dbo.sysreferences to min_privs
grant select on dbo.syscomments to min_privs
grant select on dbo.syspartitions to min_privs
grant select on dbo.syspartitionkeys to min_privs
grant select on dbo.sysconstraints to min_privs
grant select on dbo.systypes to min_privs
grant select on dbo.sysqueryplans to min_privs
```

SAP ASE(Sybase) 소스에 연결

다음 절차에 따라 AWS Schema Conversion Tool을 사용하여 SAP ASE 소스 데이터베이스에 연결합니다.

SAP ASE 소스 데이터베이스에 연결하려면

1. AWS Schema Conversion Tool에서 소스 추가를 선택합니다.
2. SAP ASE를 선택하고 다음을 선택합니다.

   소스 추가 대화 상자가 나타납니다.
3. 연결 이름에 데이터베이스의 이름을 입력합니다. AWS SCT는 왼쪽 패널의 트리에 이 이름을 표시합니다.

4. AWS Secrets Manager의 데이터베이스 보안 인증 정보를 사용하거나 수동으로 입력합니다.

   - Secrets Manager의 데이터베이스 보안 인증 정보를 사용하려면 다음 지침을 따릅니다.

     1. AWS Secret에서 보안 암호의 이름을 선택합니다.
     2. Populate를 선택하여 Secrets Manager에서 데이터베이스 연결 대화 상자에 있는 모든 값을 자동으로 채웁니다.

     Secrets Manager의 데이터베이스 보안 인증 사용에 대한 자세한 내용은 AWS Secrets Manager 사용 섹션을 참조하세요.

   - SAP ASE 소스 데이터베이스 연결 정보를 수동으로 입력하려면 다음 지침을 사용합니다.

파라미터	작업
[Server name]	소스 데이터베이스 서버의 도메인 이름 시스템(DNS) 이름 또는 IP 주소를 입력합니다.
[Server port]	소스 데이터베이스 서버에 연결하는 데 사용되는 포트를 입력합니다.
데이터베이스	SAP ASE 데이터베이스의 이름을 입력합니다.
User name 및 Password	소스 데이터베이스 서버에 연결하는 데 필요한 데이터베이스 보안 인증 정보를 입력합니다. **Note** 프로젝트에서 데이터베이스에 연결하도록 선택한 경우에만 AWS SCT에서 암호를 사용하여 소스 데이터베이스에 연결합니다. 소스 데이터베이스의 암호가 노출될 위험을 방지하기 위해 AWS SCT는 기본적으로 암호를 저장하지 않습니다. AWS SCT 프로젝트를 닫았다 다시 열 경우 필요에 따라 소스 데이터베이스에 연결하기 위한 암호를 입력하라는 메시지가 표시됩니다.

파라미터	작업
Use SSL(SSL 사용)	SSL(Secure Sockets Layer)을 사용하여 데이터베이스에 연결하려면 이 옵션을 선택합니다. SSL 탭에서 다음 추가 정보를 적절히 제공합니다. • Verify server certificate: 트러스트 스토어를 사용하여 서버 인증서를 확인하려면 이 옵션을 선택합니다. • 트러스트 스토어: 인증서가 있는 트러스트 스토어의 위치입니다.
Store password	AWS SCT는 SSL 인증서와 데이터베이스 암호를 저장할 안전한 볼트를 생성합니다. 이 옵션을 활성화하면 데이터베이스 암호를 저장하고 암호 입력 없이 빠르게 데이터베이스에 연결할 수 있습니다.
SAP ASE driver path	소스 데이터베이스에 연결할 때 사용할 드라이버의 경로를 입력합니다. 자세한 내용은 필수 데이터베이스 드라이버 다운로드 섹션을 참조하세요. 드라이버 경로를 전역 프로젝트 설정에 저장할 경우 드라이버 경로가 연결 대화 상자에 표시되지 않습니다. 자세한 내용은 전역 설정에 드라이버 경로 저장 섹션을 참조하세요.

5. Test Connection을 선택하여 AWS SCT가 소스 데이터베이스에 연결할 수 있는지 확인합니다.
6. 연결을 선택하여 소스 데이터베이스에 연결합니다.

MySQL을 대상 데이터베이스로 사용하기 위한 권한

MySQL을 대상으로 사용하기 위해 필요한 권한은 다음과 같습니다.

- CREATE ON *.*
- ALTER ON *.*
- DROP ON *.*
- INDEX ON *.*
- REFERENCES ON *.*
- SELECT ON *.*

- CREATE VIEW ON *.*
- SHOW VIEW ON *.*
- TRIGGER ON *.*
- CREATE ROUTINE ON *.*
- ALTER ROUTINE ON *.*
- EXECUTE ON *.*
- INSERT, UPDATE ON AWS_SAPASE_EXT.*
- CREATE TEMPORARY TABLES ON AWS_SAPASE_EXT.*

다음 코드 예제를 사용하여 데이터베이스 사용자를 생성하고 권한을 부여할 수 있습니다.

```
CREATE USER 'user_name' IDENTIFIED BY 'your_password';
GRANT CREATE ON *.* TO 'user_name';
GRANT ALTER ON *.* TO 'user_name';
GRANT DROP ON *.* TO 'user_name';
GRANT INDEX ON *.* TO 'user_name';
GRANT REFERENCES ON *.* TO 'user_name';
GRANT SELECT ON *.* TO 'user_name';
GRANT CREATE VIEW ON *.* TO 'user_name';
GRANT SHOW VIEW ON *.* TO 'user_name';
GRANT TRIGGER ON *.* TO 'user_name';
GRANT CREATE ROUTINE ON *.* TO 'user_name';
GRANT ALTER ROUTINE ON *.* TO 'user_name';
GRANT EXECUTE ON *.* TO 'user_name';
GRANT INSERT, UPDATE ON AWS_SAPASE_EXT.* TO 'user_name';
GRANT CREATE TEMPORARY TABLES ON AWS_SAPASE_EXT.* TO 'user_name';
```

이전 예제에서 *user_name*을 사용자 이름으로 바꿉니다. 그런 다음 *your_password*를 안전한 암호로 바꿉니다.

Amazon RDS for MySQL 또는 Aurora MySQL을 대상으로 사용하려면 `lower_case_table_names` 파라미터를 1로 설정합니다. 이 값은 MySQL 서버가 테이블, 인덱스, 트리거 및 데이터베이스와 같은 객체 이름의 식별자를 대소문자 구분 없이 처리한다는 것을 의미합니다. 대상 인스턴스에서 이진 로깅을 활성화했다면 `log_bin_trust_function_creators` 파라미터를 1로 설정합니다. 이 경우 저장된 함수를 생성하기 위해 `DETERMINISTIC`, `READS SQL DATA` 또는 `NO SQL` 특성을 사용할 필요가 없습니다. 이들 파라미터를 구성하려면 새 DB 파라미터 그룹을 생성하거나 기존 DB 파라미터 그룹을 수정해야 합니다.

SAP ASE에서 MySQL로 변환 설정

SAP ASE에서 MySQL로의 변환 설정을 편집하려면 설정을 선택한 다음 변환 설정을 선택합니다. 상단 목록에서 SAP ASE를 선택한 다음 SAP ASE – MySQL 또는 SAP ASE – Amazon Aurora (MySQL compatible)를 선택합니다. AWS SCT는 SAP ASE에서 PostgreSQL로의 변환에 사용할 수 있는 모든 설정을 표시합니다.

AWS SCT의 경우, SAP ASE에서 MySQL로의 변환 설정에는 다음에 대한 옵션이 포함됩니다.

- 변환된 코드에서 작업 항목이 포함된 설명의 수를 제한합니다.

  Add comments in the converted code for the action items of selected severity and higher에서 작업 항목의 심각도를 선택합니다. AWS SCT가 선택된 심각도 이상의 작업 항목에 대한 변환된 코드에 설명을 추가합니다.

  예를 들어, 변환된 코드의 설명 수를 최소화하려면 오류만을 선택합니다. 변환된 코드의 모든 작업 항목에 대한 설명을 포함하려면 모든 메시지를 선택합니다.
- 변환된 코드에서 소스 데이터베이스 객체의 정확한 이름을 사용합니다.

  기본적으로 AWS SCT는 데이터베이스 객체, 변수 및 파라미터의 이름을 소문자로 변환합니다. 이러한 이름에서 원래의 대/소문자를 유지하려면 Treat source database object names as case sensitive를 선택합니다. 소스 SAP ASE 데이터베이스 서버에서 대소문자를 구분한 객체 이름을 사용하는 경우 이 옵션을 선택합니다.

PostgreSQL을 대상 데이터베이스로 사용하기 위한 권한

PostgreSQL을 대상으로 사용하려면 AWS SCT에 `CREATE ON DATABASE` 권한이 필요합니다. 각 대상 PostgreSQL 데이터베이스에 대해 이 권한을 부여해야 합니다.

변환된 공개 동의어를 사용하려면 데이터베이스 기본 검색 경로를 `"$user", public_synonyms, public`으로 변경합니다.

다음 코드 예제를 사용하여 데이터베이스 사용자를 생성하고 권한을 부여할 수 있습니다.

```
CREATE ROLE user_name LOGIN PASSWORD 'your_password';
GRANT CREATE ON DATABASE db_name TO user_name;
ALTER DATABASE db_name SET SEARCH_PATH = "$user", public_synonyms, public;
```

이전 예제에서 *user_name*을 사용자 이름으로 바꿉니다. 그런 다음 *db_name*을 대상 데이터베이스의 이름으로 바꿉니다. 마지막으로 *your_password*를 안전한 암호로 바꿉니다.

PostgreSQL에서는 스키마 소유자 또는 superuser만 스키마를 삭제할 수 있습니다. 소유자는 스키마 소유자가 일부 객체를 소유하지 않은 경우에도 스키마 및 이 스키마에 포함된 모든 객체를 삭제할 수 있습니다.

여러 사용자를 통해 대상 데이터베이스를 변환하고 다른 스키마를 적용할 때 AWS SCT에서 스키마를 삭제할 수 없는 경우 오류 메시지가 표시될 수 있습니다. 이 오류 메시지가 표시되지 않도록 하려면 superuser 역할을 사용하세요.

SAP ASE에서 PostgreSQL로 변환 설정

SAP ASE에서 PostgreSQL로의 변환 설정을 편집하려면 설정을 선택한 다음 변환 설정을 선택합니다. 상단 목록에서 SAP ASE를 선택한 다음 SAP ASE – PostgreSQL 또는 SAP ASE – Amazon Aurora (PostgreSQL compatible)를 선택합니다. AWS SCT는 SAP ASE에서 PostgreSQL로의 변환에 사용할 수 있는 모든 설정을 표시합니다.

AWS SCT의 경우, SAP ASE에서 PostgreSQL로의 변환 설정에는 다음에 대한 옵션이 포함됩니다.

- 변환된 코드에서 작업 항목이 포함된 설명의 수를 제한합니다.

  Add comments in the converted code for the action items of selected severity and higher에서 작업 항목의 심각도를 선택합니다. AWS SCT가 선택된 심각도 이상의 작업 항목에 대한 변환된 코드에 설명을 추가합니다.

  예를 들어, 변환된 코드의 설명 수를 최소화하려면 오류만을 선택합니다. 변환된 코드의 모든 작업 항목에 대한 설명을 포함하려면 모든 메시지를 선택합니다.
- 변환된 코드의 스키마 이름에 사용할 템플릿을 정의합니다. Schema name generation template에서 다음 옵션 중 하나를 선택합니다.
  - <source_db> – PostgreSQL에서 SAP ASE 데이터베이스 이름을 스키마 이름으로 사용합니다.
  - <source_schema> – PostgreSQL에서 SAP ASE 스키마 이름을 스키마 이름으로 사용합니다.
  - <source_db>_<schema> – PostgreSQL에서 SAP ASE 데이터베이스와 스키마 이름의 조합을 스키마 이름으로 사용합니다.
- 변환된 코드에서 소스 데이터베이스 객체의 정확한 이름을 사용합니다.

  기본적으로 AWS SCT는 데이터베이스 객체, 변수 및 파라미터의 이름을 소문자로 변환합니다. 이러한 이름에서 원래의 대/소문자를 유지하려면 Treat source database object names as case

sensitive를 선택합니다. 소스 SAP ASE 데이터베이스 서버에서 대소문자를 구분한 객체 이름을 사용하는 경우 이 옵션을 선택합니다.

대소문자를 구분하는 작업의 경우 AWS SCT는 데이터베이스 객체 이름을 소문자로 변환하지 않아도 됩니다. 이렇게 하려면 Avoid casting to lowercase for case sensitive operations를 선택합니다.

- SAP ASE의 여러 테이블에서 이름이 같은 인덱스를 사용할 수 있도록 합니다.

  PostgreSQL에서는 스키마에서 사용하는 모든 인덱스 이름이 고유해야 합니다. AWS SCT가 모든 인덱스에 대해 고유한 이름을 생성하도록 하려면 인덱스에 대한 고유 이름 생성을 선택합니다.

Microsoft SQL Server를 AWS SCT에 대한 소스로 사용

AWS SCT를 사용하여 스키마, 데이터베이스 코드 객체 및 애플리케이션 코드를 SQL Server에서 다음 대상으로 변환할 수 있습니다.

- Amazon RDS for MySQL
- Amazon Aurora MySQL-Compatible Edition
- Amazon RDS for PostgreSQL
- Amazon Aurora PostgreSQL-Compatible Edition
- Amazon RDS for SQL Server
- Amazon RDS for MariaDB

> **Note**
>
> AWS SCT에서는 Amazon RDS for SQL Server를 소스로 사용하는 기능이 지원되지 않습니다.

AWS SCT를 사용하면 다음 설명과 같이 SQL Server에서 Babelfish for Aurora PostgreSQL로 스키마, 데이터베이스 코드 객체 및 애플리케이션 코드를 마이그레이션하기 위한 평가 보고서를 생성할 수 있습니다.

주제

- Microsoft SQL Server를 소스로 사용하기 위한 권한
- Microsoft SQL Server를 소스로 사용할 때 Windows 인증 사용
- SQL Server에 소스 연결

- SQL Server를 MySQL로 변환
- SQL Server를 PostgreSQL로 변환
- SQL Server를 Amazon RDS for SQL Server로 변환

Microsoft SQL Server를 소스로 사용하기 위한 권한

Microsoft SQL Server를 소스로 사용하기 위해 필요한 권한은 다음과 같습니다.

- VIEW DEFINITION
- VIEW DATABASE STATE

`VIEW DEFINITION` 권한을 사용하면 퍼블릭 액세스 권한을 가진 사용자가 객체 정의를 볼 수 있습니다. AWS SCT는 `VIEW DATABASE STATE` 권한을 사용하여 SQL Server Enterprise 버전의 기능을 확인합니다.

변환하려는 스키마의 각 데이터베이스에 대해 권한 부여를 반복합니다.

또한 `master` 데이터베이스에 다음 권한을 부여합니다.

- VIEW SERVER STATE
- VIEW ANY DEFINITION

AWS SCT는 `VIEW SERVER STATE` 권한을 사용하여 서버 설정 및 구성을 수집합니다. 엔드포인트를 보려면 `VIEW ANY DEFINITION` 권한을 부여해야 합니다.

Microsoft Analysis Services에 관한 정보를 읽으려면 `master` 데이터베이스에서 다음 명령을 실행합니다.

```
EXEC master..sp_addsrvrolemember @loginame = N'<user_name>', @rolename = N'sysadmin'
```

이전 예제에서 *<user_name>* 자리 표시자를 이전에 권한을 부여한 사용자의 이름으로 바꿉니다.

SQL Server 에이전트에 관한 정보를 읽으려면 사용자를 `SQLAgentUser` 역할에 추가합니다. `msdb` 데이터베이스에서 다음 명령을 실행합니다.

```
EXEC sp_addrolemember <SQLAgentRole>, <user_name>;
```

앞의 예제에서 *<SQLAgentRole>* 자리 표시자를 SQL Server 에이전트 역할의 이름으로 바꿉니다. 그런 다음, *<user_name>* 자리 표시자를 이전에 권한을 부여한 사용자의 이름으로 바꿉니다. 자세한 내용은 Amazon RDS 사용 설명서의 SQLAgentUser 역할에 사용자 추가를 참조하세요.

로그 전달을 감지하려면 msdb 데이터베이스에 대한 SELECT on dbo.log_shipping_primary_databases 권한을 부여합니다.

DDL 복제의 알림 방식을 사용하려면 소스 데이터베이스에 대해 RECEIVE ON *<schema_name>*.*<queue_name>* 권한을 부여합니다. 이 예제에서는 *<schema_name>* 자리 표시자를 데이터베이스의 스키마 이름으로 바꿉니다. 그런 다음, *<queue_name>* 자리 표시자를 대기열 테이블 이름으로 바꿉니다.

Microsoft SQL Server를 소스로 사용할 때 Windows 인증 사용

Windows 기반 인트라넷에서 애플리케이션을 실행하는 경우 데이터베이스 액세스에 Windows 인증을 사용할 수 있습니다. Windows 인증은 운영 체제 스레드에 설정된 현재 Windows ID를 사용하여 SQL Server 데이터베이스에 액세스합니다. 그런 다음 Windows ID를 SQL Server 데이터베이스 및 권한에 매핑할 수 있습니다. Windows 인증을 사용하여 SQL Server에 연결하려면 애플리케이션에서 사용하는 Windows ID를 지정해야 합니다. 또한 SQL Server 데이터베이스에 대한 Windows ID 액세스 권한을 부여해야 합니다.

SQL Server에는 Windows 인증 모드와 혼합 모드의 두 가지 액세스 모드가 있습니다. Windows 인증 모드에서는 Windows 인증을 활성화하고 SQL Server 인증을 비활성화합니다. 혼합 모드에서는 Windows 인증과 SQL Server 인증을 모두 사용할 수 있습니다. Windows 인증은 항상 사용할 수 있으며 비활성화할 수 없습니다. Windows 인증에 대한 자세한 내용은 Microsoft Windows 설명서를 참조하세요.

다음은 TEST_DB에 사용자를 생성하는 것을 보여주는 예입니다.

```
USE [TEST_DB]
CREATE USER [TestUser] FOR LOGIN [TestDomain\TestUser]
GRANT VIEW DEFINITION TO [TestUser]
GRANT VIEW DATABASE STATE TO [TestUser]
```

JDBC 연결에서 Windows 인증 사용

JDBC 드라이버는 Windows 이외의 운영 체제에서 사용되는 경우 Windows 인증을 지원하지 않습니다. Windows 이외의 운영 체제에서 SQL Server에 연결할 때는 사용자 이름 및 암호와 같은 Windows 인증 보안 인증 정보가 자동으로 지정되지 않습니다. 이 경우 애플리케이션은 SQL Server 인증을 대신 사용해야 합니다.

JDBC 연결 문자열에서 Windows 인증을 사용하여 연결하려면 `integratedSecurity` 파라미터를 지정해야 합니다. JDBC 드라이버는 `integratedSecurity` 연결 문자열 파라미터를 통해 Windows 운영 체제에서 Windows 통합 인증을 지원합니다.

통합 인증을 사용하려면

1. JDBC 드라이버를 설치합니다.
2. JDBC 드라이버가 설치된 컴퓨터의 Windows 시스템 경로에 있는 디렉터리에 `sqljdbc_auth.dll` 파일을 복사합니다.

   `sqljdbc_auth.dll` 파일은 다음 위치에 설치됩니다.

   <installation directory>\sqljdbc_<version>\<language>\auth\

Windows 인증을 사용하여 SQL Server 데이터베이스에 연결을 시도하는 경우 "이 드라이버는 통합 인증에 맞게 구성되지 않았습니다"라는 오류가 발생할 수 있습니다. 이 문제는 다음 작업을 수행하여 해결할 수 있습니다.

- 다음과 같이 JDBC 설치 경로를 가리키는 두 가지 변수를 선언합니다.

  `variable name: SQLJDBC_HOME; variable value: D:\lib\JDBC4.1\enu`(sqljdbc4.jar이 있는 위치);

  `variable name: SQLJDBC_AUTH_HOME; variable value: D\lib\JDBC4.1\enu\auth\x86`(32비트 OS를 실행하는 경우) 또는 `D\lib\JDBC4.1\enu\auth\x64`(64비트 OS를 실행하는 경우). `sqljdbc_auth.dll`이 있는 위치입니다.
- JDK/JRE가 실행 중인 폴더에 `sqljdbc_auth.dll`을 복사합니다. lib 폴더, bin 폴더 등에 복사할 수 있습니다. 예를 들어 다음 폴더에 복사할 수 있습니다.

  ```
  [JDK_INSTALLED_PATH]\bin;
  [JDK_INSTALLED_PATH]\jre\bin;
  [JDK_INSTALLED_PATH]\jre\lib;
  [JDK_INSTALLED_PATH]\lib;
  ```
- JDBC 라이브러리 폴더에는 SQLJDBC4.jar 파일만 있어야 합니다. 이 폴더에서 다른 sqljdbc*.jar 파일을 모두 제거(또는 다른 폴더로 복사)합니다. 드라이버를 프로그램의 일부로 추가하는 경우 사용할 드라이버로 SQLJDBC4.jar만 추가해야 합니다.
- 애플리케이션의 해당 폴더에 sqljdbc_auth.dll 파일을 복사합니다.

> **Note**
>
> 32비트 Java Virtual Machine(JVM)을 실행 중인 경우 x86 폴더에 있는 sqljdbc_auth.dll 파일을 사용합니다. 이는 운영 체제가 x64 버전인 경우에도 마찬가지입니다. x64 프로세서에서 64비트 JVM을 실행하는 경우에 x64 폴더의 sqljdbc_auth.dll 파일을 사용합니다.

SQL Server 데이터베이스에 연결할 때 인증 옵션으로 Windows 인증 또는 SQL Server 인증을 선택할 수 있습니다.

SQL Server에 소스 연결

다음 절차에 따라 AWS Schema Conversion Tool을 사용하여 Microsoft SQL Server 소스 데이터베이스에 연결합니다.

Microsoft SQL Server 소스 데이터베이스에 연결하려면

1. AWS Schema Conversion Tool에서 소스 추가를 선택합니다.
2. Microsoft SQL Server를 선택한 후 다음을 선택합니다.

   소스 추가 대화 상자가 나타납니다.
3. 연결 이름에 데이터베이스의 이름을 입력합니다. AWS SCT는 왼쪽 패널의 트리에 이 이름을 표시합니다.
4. AWS Secrets Manager의 데이터베이스 보안 인증 정보를 사용하거나 수동으로 입력합니다.

   - Secrets Manager의 데이터베이스 보안 인증 정보를 사용하려면 다음 지침을 따릅니다.
     1. AWS Secret에서 보안 암호의 이름을 선택합니다.
     2. Populate를 선택하여 Secrets Manager에서 데이터베이스 연결 대화 상자에 있는 모든 값을 자동으로 채웁니다.

     Secrets Manager의 데이터베이스 보안 인증 사용에 대한 자세한 내용은 AWS Secrets Manager 사용 섹션을 참조하세요.
   - Microsoft SQL Server 소스 데이터베이스 연결 정보를 수동으로 입력하려면 다음 지침을 사용합니다.

파라미터	작업
[Server name]	소스 데이터베이스 서버의 DNS(Domain Name Service) 이름 또는 IP 주소를 입력합니다. IPv6 주소 프로토콜을 사용하여 소스 SQL Server 데이터베이스에 연결할 수 있습니다. 이렇게 하려면 다음 예제와 같이 대괄호를 사용하여 IP 주소를 입력해야 합니다. `[2001:db8:ffff:ffff:ffff:ffff:ffff:fffe]`
[Server port]	소스 데이터베이스 서버에 연결하는 데 사용되는 포트를 입력합니다.
인스턴스 이름	SQL Server 데이터베이스의 인스턴스 이름을 입력합니다. 인스턴스 이름을 찾으려면 SQL Server 데이터베이스에서 쿼리 `SELECT @@servername;` 을 실행합니다.
인증	Windows 인증 및 SQL Server 인증에서 인증 유형을 선택합니다.
User name 및 Password	소스 데이터베이스 서버에 연결하는 데 필요한 데이터베이스 보안 인증 정보를 입력합니다. 프로젝트에서 데이터베이스에 연결하도록 선택한 경우에만 AWS SCT에서 암호를 사용하여 소스 데이터베이스에 연결합니다. 소스 데이터베이스의 암호가 노출될 위험을 방지하기 위해 AWS SCT는 기본적으로 암호를 저장하지 않습니다. AWS SCT 프로젝트를 닫았다 다시 열 경우 필요에 따라 소스 데이터베이스에 연결하기 위한 암호를 입력하라는 메시지가 표시됩니다.

파라미터	작업
Use SSL(SSL 사용)	SSL(Secure Sockets Layer)을 사용하여 데이터베이스에 연결하려면 이 옵션을 선택합니다. SSL 탭에서 다음 추가 정보를 적절히 제공합니다. • Trust server certificate: 서버 인증서를 신뢰하려면 이 옵션을 선택합니다. • 트러스트 스토어: 인증서가 있는 트러스트 스토어의 위치입니다. 이 위치가 전역 설정 섹션에 표시되도록 하려면 해당 위치를 추가해야 합니다.
Store password	AWS SCT는 SSL 인증서와 데이터베이스 암호를 저장할 안전한 볼트를 생성합니다. 이 옵션을 활성화하면 데이터베이스 암호를 저장하고 암호 입력 없이 빠르게 데이터베이스에 연결할 수 있습니다.
Sql Server Driver Path	소스 데이터베이스에 연결할 때 사용할 드라이버의 경로를 입력합니다. 자세한 내용은 필수 데이터베이스 드라이버 다운로드 섹션을 참조하세요. 드라이버 경로를 전역 프로젝트 설정에 저장할 경우 드라이버 경로가 연결 대화 상자에 표시되지 않습니다. 자세한 내용은 전역 설정에 드라이버 경로 저장 섹션을 참조하세요.
Windows Authentication library	`sqljdbc_auth.dll` 파일의 경로를 입력합니다. 기본적으로 이 파일은 다음 위치에 설치됩니다. *`<installation directory of the JDBC driver>`*`sqljdbc_`*`<version>`*` \`*`<language>`*` \auth\`

5. Test Connection을 선택하여 AWS SCT가 소스 데이터베이스에 연결할 수 있는지 확인합니다.
6. 연결을 선택하여 소스 데이터베이스에 연결합니다.

SQL Server를 MySQL로 변환

변환된 MySQL 코드에서 Microsoft SQL Server 데이터베이스 함수를 에뮬레이션하려면 AWS SCT에서 SQL Server-MySQL 확장 팩을 사용합니다. 확장 팩에 대한 자세한 내용은 AWS SCT 확장 팩 사용 섹션을 참조하세요.

주제

- MySQL을 대상 데이터베이스로 사용하기 위한 권한
- SQL Server에서 MySQL로의 변환 설정
- 마이그레이션 고려 사항

MySQL을 대상 데이터베이스로 사용하기 위한 권한

MySQL을 대상으로 사용하기 위해 필요한 권한은 다음과 같습니다.

- CREATE ON *.*
- ALTER ON *.*
- DROP ON *.*
- INDEX ON *.*
- REFERENCES ON *.*
- SELECT ON *.*
- CREATE VIEW ON *.*
- SHOW VIEW ON *.*
- TRIGGER ON *.*
- CREATE ROUTINE ON *.*
- ALTER ROUTINE ON *.*
- EXECUTE ON *.*
- INSERT, UPDATE ON AWS_SQLSERVER_EXT.*
- INSERT, UPDATE, DELETE ON AWS_SQLSERVER_EXT_DATA.*
- CREATE TEMPORARY TABLES ON AWS_SQLSERVER_EXT_DATA.*

다음 코드 예제를 사용하여 데이터베이스 사용자를 생성하고 권한을 부여할 수 있습니다.

```
CREATE USER 'user_name' IDENTIFIED BY 'your_password';
GRANT CREATE ON *.* TO 'user_name';
GRANT ALTER ON *.* TO 'user_name';
GRANT DROP ON *.* TO 'user_name';
GRANT INDEX ON *.* TO 'user_name';
GRANT REFERENCES ON *.* TO 'user_name';
```

```
GRANT SELECT ON *.* TO 'user_name';
GRANT CREATE VIEW ON *.* TO 'user_name';
GRANT SHOW VIEW ON *.* TO 'user_name';
GRANT TRIGGER ON *.* TO 'user_name';
GRANT CREATE ROUTINE ON *.* TO 'user_name';
GRANT ALTER ROUTINE ON *.* TO 'user_name';
GRANT EXECUTE ON *.* TO 'user_name';
GRANT INSERT, UPDATE ON AWS_SQLSERVER_EXT.* TO 'user_name';
GRANT INSERT, UPDATE, DELETE ON AWS_SQLSERVER_EXT_DATA.* TO 'user_name';
GRANT CREATE TEMPORARY TABLES ON AWS_SQLSERVER_EXT_DATA.* TO 'user_name';
```

이전 예제에서 *user_name*을 사용자 이름으로 바꿉니다. 그런 다음 *your_password*를 안전한 암호로 바꿉니다.

MySQL 데이터베이스 버전 5.7 이하를 대상으로 사용하는 경우 다음 명령을 실행합니다. MySQL 데이터베이스 버전 8.0 이상에서는 이 명령이 더 이상 사용되지 않습니다.

```
GRANT SELECT ON mysql.proc TO 'user_name';
```

Amazon RDS for MySQL 또는 Amazon RDS for Aurora MySQL을 대상으로 사용하려면 `lower_case_table_names` 파라미터를 1로 설정합니다. 이 값은 MySQL 서버가 테이블, 인덱스, 트리거 및 데이터베이스와 같은 객체 이름의 식별자를 대소문자 구분 없이 처리한다는 것을 의미합니다. 대상 인스턴스에서 이진 로깅을 활성화했다면 `log_bin_trust_function_creators` 파라미터를 1로 설정합니다. 이 경우 저장된 함수를 생성하기 위해 `DETERMINISTIC`, `READS SQL DATA` 또는 `NO SQL` 특성을 사용할 필요가 없습니다. 이들 파라미터를 구성하려면 새 DB 파라미터 그룹을 생성하거나 기존 DB 파라미터 그룹을 수정해야 합니다.

SQL Server에서 MySQL로의 변환 설정

SQL Server에서 MySQL로의 변환 설정을 편집하려면 AWS SCT에서 설정을 선택한 다음 변환 설정을 선택합니다. 상단 목록에서 SQL Server를 선택한 다음 SQL Server – MySQL을 선택합니다. AWS SCT는 SQL Server에서 MySQL로의 변환에 사용할 수 있는 모든 설정을 표시합니다.

AWS SCT의 경우, SQL Server에서 MySQL로의 변환 설정에는 다음에 대한 옵션이 포함됩니다.

- 변환된 코드에서 작업 항목이 포함된 설명의 수를 제한합니다.

  Add comments in the converted code for the action items of selected severity and higher에서 작업 항목의 심각도를 선택합니다. AWS SCT가 선택된 심각도 이상의 작업 항목에 대한 변환된 코드에 설명을 추가합니다.

예를 들어, 변환된 코드의 설명 수를 최소화하려면 오류만을 선택합니다. 변환된 코드의 모든 작업 항목에 대한 설명을 포함하려면 모든 메시지를 선택합니다.

- 소스 SQL Server 데이터베이스에서 EXEC의 출력을 테이블에 저장할 수 있도록 하려면 AWS SCT가 이 기능을 에뮬레이션하기 위한 임시 테이블 및 추가 프로시저를 생성합니다. 이 에뮬레이션을 사용하려면 Create additional routines for handling open datasets를 선택합니다.

마이그레이션 고려 사항

SQL Server 스키마를 MySQL로 마이그레이션할 때는 다음 사항을 고려합니다.

- MySQL은 MERGE 문을 지원하지 않습니다. 하지만 AWS SCT는 변환 중에 INSERT ON DUPLICATE KEY 절과 UPDATE FROM and DELETE FROM 문을 사용하여 MERGE 문을 에뮬레이션할 수 있습니다.

  INSERT ON DUPLICATE KEY를 사용하여 올바르게 에뮬레이션하기 위해서는 대상 MySQL 데이터베이스에 고유한 제약 조건 또는 프라이머리 키가 있어야 합니다.
- GOTO 문과 레이블을 사용하여 명령문 실행 순서를 변경할 수 있습니다. GOTO 문 뒤에 오는 Transact-SQL 문은 건너뛰며 프로세스는 레이블에서 계속됩니다. GOTO 문과 레이블은 프로시저, 배치(batch), 명령문 블록 내 어디든 사용할 수 있습니다. GOTO 문을 중첩할 수도 있습니다.

  MySQL은 GOTO 문을 사용하지 않습니다. AWS SCT는 GOTO 문을 포함하는 코드를 변환할 때 BEGIN…END 또는 LOOP…END LOOP 문을 사용하도록 명령문을 변환합니다. 다음 테이블에는 AWS SCT가 GOTO 문을 어떻게 변환하는지에 대한 예가 나와 있습니다.

SQL Server 문	MySQL 문

BEGIN
....
statement1;
....
GOTO label1;
statement2;
....
label1:
Statement3;
....
END
	BEGIN
	label1:
	BEGIN
	....
	statement1;
	....
	LEAVE label1;
	statement2;
	....
	END;
	Statement3;
	....

SQL Server 문	MySQL 문
	END
BEGIN .... statement1; .... label1: statement2; .... GOTO label1; statement3; .... statement4; .... END	BEGIN .... statement1; .... label1: LOOP statement2; .... ITERATE label1; LEAVE label1; END LOOP; statement3; .... statement4; .... END

BEGIN
....
statement1;
....
label1:
statement2;
....
statement3;
....
statement4;
....
END
	BEGIN
	....
	statement1;
	....
	label1:
	BEGIN
	statement2;
	....
	statement3;
	....
	statement4;
	....
	END;
	END

- MySQL은 다중 명령문 테이블 반환 함수를 지원하지 않습니다. AWS SCT는 임시 테이블을 만들고 이러한 임시 테이블을 사용하도록 명령문을 다시 작성하여 변환 중에 테이블 반환 함수를 시뮬레이션합니다.

SQL Server를 PostgreSQL로 변환

AWS SCT에서는 SQL Server에서 PostgreSQL로의 확장 팩을 사용할 수 있습니다. 이 확장 팩은 변환된 PostgreSQL 코드에서 SQL Server 데이터베이스 함수를 에뮬레이션합니다. SQL Server에서 PostgreSQL로의 확장 팩을 사용하여 SQL Server 에이전트 및 SQL Server Database Mail을 에뮬레이션할 수 있습니다. 확장 팩에 대한 자세한 내용은 AWS SCT 확장 팩 사용 섹션을 참조하세요.

주제

- PostgreSQL을 대상 데이터베이스로 사용하기 위한 권한
- SQL Server에서 PostgreSQL로 변환 설정
- SQL Server 파티션을 PostgreSQL 버전 10 파티션으로 변환
- 마이그레이션 고려 사항
- AWS SCT 확장 팩을 사용하여 PostgreSQL에서 SQL Server 에이전트 에뮬레이션
- AWS SCT 확장 팩을 사용하여 PostgreSQL에서 SQL Server Database Mail 에뮬레이션

PostgreSQL을 대상 데이터베이스로 사용하기 위한 권한

PostgreSQL을 대상으로 사용하려면 AWS SCT에 `CREATE ON DATABASE` 권한이 필요합니다. 각 대상 PostgreSQL 데이터베이스에 대해 이 권한을 부여해야 합니다.

변환된 공개 동의어를 사용하려면 데이터베이스 기본 검색 경로를 `"$user", public_synonyms, public`으로 변경합니다.

다음 코드 예제를 사용하여 데이터베이스 사용자를 생성하고 권한을 부여할 수 있습니다.

```
CREATE ROLE user_name LOGIN PASSWORD 'your_password';
GRANT CREATE ON DATABASE db_name TO user_name;
ALTER DATABASE db_name SET SEARCH_PATH = "$user", public_synonyms, public;
```

이전 예제에서 *user_name*을 사용자 이름으로 바꿉니다. 그런 다음 *db_name*을 대상 데이터베이스의 이름으로 바꿉니다. 마지막으로 *your_password*를 안전한 암호로 바꿉니다.

PostgreSQL에서는 스키마 소유자 또는 superuser만 스키마를 삭제할 수 있습니다. 소유자는 스키마 소유자가 일부 객체를 소유하지 않은 경우에도 스키마 및 이 스키마에 포함된 모든 객체를 삭제할 수 있습니다.

여러 사용자를 통해 대상 데이터베이스를 변환하고 다른 스키마를 적용할 때 AWS SCT에서 스키마를 삭제할 수 없는 경우 오류 메시지가 표시될 수 있습니다. 이 오류 메시지가 표시되지 않도록 하려면 `superuser` 역할을 사용하세요.

SQL Server에서 PostgreSQL로 변환 설정

SQL Server에서 PostgreSQL로의 변환 설정을 편집하려면 설정을 선택한 다음 변환 설정을 선택합니다. 상단 목록에서 SQL Server를 선택한 다음 SQL Server – PostgreSQL을 선택합니다. AWS SCT는 SQL Server에서 PostgreSQL로의 변환에 사용할 수 있는 모든 설정을 표시합니다.

AWS SCT의 경우, SQL Server에서 PostgreSQL로의 변환 설정에는 다음에 대한 옵션이 포함됩니다.

- 변환된 코드에서 작업 항목이 포함된 설명의 수를 제한합니다.

  Add comments in the converted code for the action items of selected severity and higher에서 작업 항목의 심각도를 선택합니다. AWS SCT가 선택된 심각도 이상의 작업 항목에 대한 변환된 코드에 설명을 추가합니다.

  예를 들어, 변환된 코드의 설명 수를 최소화하려면 오류만을 선택합니다. 변환된 코드의 모든 작업 항목에 대한 설명을 포함하려면 모든 메시지를 선택합니다.
- SQL Server의 여러 테이블에서 이름이 같은 인덱스를 사용할 수 있도록 합니다.

  PostgreSQL에서는 스키마에서 사용하는 모든 인덱스 이름이 고유해야 합니다. AWS SCT가 모든 인덱스에 대해 고유한 이름을 생성하도록 하려면 인덱스에 대한 고유 이름 생성을 선택합니다.
- SQL Server 프로시저를 PostgreSQL 함수로 변환합니다.

  PostgreSQL 버전 10 이하에서는 프로시저를 지원하지 않습니다. PostgreSQL에서 프로시저를 사용하는 데 익숙하지 않은 고객을 위해 AWS SCT는 프로시저를 함수로 변환할 수 있습니다. 이렇게 하려면 프로시저를 함수로 변환을 선택합니다.
- 테이블에서 `EXEC` 출력을 에뮬레이션하려면

  소스 SQL Server 데이터베이스는 `EXEC`의 출력을 테이블에 저장할 수 있습니다. AWS SCT는 임시 테이블과 추가 프로시저를 생성하여 이 기능을 에뮬레이션합니다. 이 에뮬레이션을 사용하려면 Create additional routines for handling open datasets를 선택합니다.
- 변환된 코드의 스키마 이름에 사용할 템플릿을 정의합니다. Schema name generation template에서 다음 옵션 중 하나를 선택합니다.
  - <source_db> – PostgreSQL에서 SQL Server 데이터베이스 이름을 스키마 이름으로 사용합니다.
  - <source_schema> – PostgreSQL에서 SQL Server 스키마 이름을 스키마 이름으로 사용합니다.

  - <source_db>_<schema> – PostgreSQL에서 SQL Server 데이터베이스와 스키마 이름의 조합을 스키마 이름으로 사용합니다.
- 소스 객체 이름의 대소문자를 그대로 사용합니다.

  객체 이름을 소문자로 변환하지 않으려면 Avoid casting to lower case for case sensitive operations를 선택합니다. 이 옵션은 대상 데이터베이스에서 대소문자 구분 옵션을 활성화한 경우에만 적용됩니다.
- 소스 데이터베이스의 파라미터 이름을 유지합니다.

  변환된 코드의 파라미터 이름에 큰따옴표를 추가하려면 원본 파라미터 이름 유지를 선택합니다.

SQL Server 파티션을 PostgreSQL 버전 10 파티션으로 변환

Microsoft SQL Server 데이터베이스를 Amazon Aurora PostgreSQL-Compatible Edition(Aurora PostgreSQL) 또는 Amazon Relational Database Service for PostgreSQL(Amazon RDS for PostgreSQL)로 변환할 때는 다음 사항에 유의합니다.

SQL Server에서는 파티션 함수를 사용하여 파티션을 생성합니다. SQL Server 파티션 테이블을 PostgreSQL 버전 10 파티션 테이블로 변환할 때는 다음과 같은 몇 가지 잠재적 문제에 유의해야 합니다.

- SQL Server에서는 NOT NULL 제약 조건 없이 열을 사용하여 테이블을 분할할 수 있습니다. 이 경우 모든 NULL 값은 맨 왼쪽 파티션으로 이동합니다. PostgreSQL은 RANGE 파티셔닝에서 NULL 값을 지원하지 않습니다.
- SQL Server에서는 파티션 테이블의 프라이머리 키와 고유 키를 만들 수 있습니다. PostgreSQL에서는 각 파티션의 프라이머리 키 또는 고유 키를 직접 생성할 수 있습니다. 따라서 PostgreSQL로 마이그레이션할 때 PRIMARY 또는 UNIQUE KEY 제약 조건을 상위 테이블에서 제거해야 합니다. 결과 키 이름은 <original_key_name>_<partition_number> 형식입니다.
- SQL Server를 사용하면 파티션 테이블에서 또는 파티션 테이블에 외래 키 제약 조건을 만들 수 있습니다. PostgreSQL은 파티션 테이블을 참조하는 외래 키를 지원하지 않습니다. 또한 PostgreSQL은 파티션 테이블에서 다른 테이블로의 외래 키 참조를 지원하지 않습니다.
- SQL Server를 사용하면 파티션 테이블의 인덱스를 만들 수 있습니다. PostgreSQL에서는 각 파티션에 대해 직접 인덱스를 생성해야 합니다. 따라서 PostgreSQL로 마이그레이션할 때 해당 인덱스를 상위 테이블에서 제거해야 합니다. 결과 인덱스 이름은 <original_index_name>_<partition_number> 형식입니다.
- PostgreSQL은 파티션된 인덱스를 지원하지 않습니다.

마이그레이션 고려 사항

SQL Server 스키마에서 PostgreSQL로 마이그레이션할 때 고려해야 할 사항:

- PostgreSQL에서는 스키마의 모든 객체 이름(인덱스 포함)이 고유해야 합니다. 인덱스 이름은 기본 테이블의 스키마에서 고유해야 합니다. SQL Server에서는 서로 다른 테이블에서 인덱스 이름이 동일할 수 있습니다.

  인덱스 이름의 고유성을 보장하기 위해 AWS SCT는 인덱스 이름이 고유하지 않은 경우 고유한 인덱스 이름을 생성하는 옵션을 제공합니다. 이렇게 하려면 프로젝트 속성에서 Generate unique index names(고유한 인덱스 이름 생성) 옵션을 선택합니다. 이 옵션은 기본적으로 활성화되어 있습니다. 이 옵션을 활성화하면 IX_table_name_index_name 형식을 사용하여 고유한 인덱스 이름이 생성됩니다. 이 옵션을 비활성화하면 인덱스 이름이 변경되지 않습니다.
- GOTO 문과 레이블을 사용하여 문 실행 순서를 변경할 수 있습니다. GOTO 문 뒤에 오는 Transact-SQL 문은 건너뛰며 프로세스는 레이블에서 계속됩니다. GOTO 문과 레이블은 프로시저, 배치(batch), 문 블록 내 어디든 사용할 수 있습니다. GOTO 문은 중첩될 수 있습니다.

  PostgreSQL은 GOTO 명령문을 사용하지 않습니다. AWS SCT는 GOTO 문을 포함하는 코드를 변환할 때 BEGIN…END 문 또는 LOOP…END LOOP 문을 사용하도록 명령문을 변환합니다. 다음 테이블에는 AWS SCT가 GOTO 문을 어떻게 변환하는지에 대한 예가 나와 있습니다.

  SQL Server GOTO 문 및 변환된 PostgreSQL 문

SQL Server 문	PostgreSQL 문

`BEGIN`
`....`
`statement1;`
`....`
`GOTO label1;`
`statement2;`
`....`
`label1:`
`Statement3;`
`....`
`END`
	`BEGIN`
	` label1:`
	` BEGIN`
	`  ....`
	`  statement1;`
	`  ....`
	`  EXIT label1;`
	`  statement2;`
	`  ....`
	` END;`
	`  Statement3;`
	`  ....`
	`END`

`BEGIN`	`BEGIN`

SQL Server 문	PostgreSQL 문
.... statement1; .... label1: statement2; .... GOTO label1; statement3; .... statement4; .... END	.... statement1; .... label1: LOOP statement2; .... CONTINUE label1; EXIT label1; END LOOP; statement3; .... statement4; .... END

BEGIN
....
statement1;
....
label1:
statement2;
....
statement3;
....
statement4;
....
END
	BEGIN
	....
	statement1;
	....
	label1:
	BEGIN
	statement2;
	....
	statement3;
	....
	statement4;
	....
	END;
	END

- PostgreSQL은 MERGE 문을 지원하지 않습니다. AWS SCT는 다음과 같은 방식으로 MERGE 문의 동작을 에뮬레이션합니다.
  - INSERT ON CONFLICT 생성.
  - UPDATE FROM DML 문 사용(예: MERGE without a WHEN NOT MATCHED 절)
  - CURSOR 사용(예: MERGE with DELETE 절) 또는 복잡한 MERGE ON 조건문 사용
- AWS SCT는 Amazon RDS가 대상일 때 객체 트리에 데이터베이스 트리거를 추가할 수 있습니다.
- AWS SCT는 Amazon RDS가 대상일 때 객체 트리에 서버 수준 트리거를 추가할 수 있습니다.

- SQL Server는 `deleted` 및 `inserted` 테이블을 자동으로 생성하고 관리합니다. 메모리에 상주하는 이러한 임시 테이블을 사용하여 특정 데이터 수정에 따른 영향을 테스트하고 DML 트리거 동작의 조건을 설정할 수 있습니다. AWS SCT는 DML 트리거 명령문 내에서 이러한 테이블의 사용을 변환할 수 있습니다.
- AWS SCT는 Amazon RDS가 대상일 때 연결된 서버를 객체 트리에 추가할 수 있습니다.
- Microsoft SQL Server에서 PostgreSQL로 마이그레이션하면 내장된 SUSER_SNAME 함수가 다음과 같이 변환됩니다.
  - SUSER_SNAME - 보안 식별 번호(SID)와 연결된 로그인 이름을 반환합니다.
  - SUSER_SNAME(<server_user_sid>) – 지원되지 않습니다.
  - SUSER_SNAME() CURRENT_USER – 현재 실행 컨텍스트의 사용자 이름을 반환합니다.
  - SUSER_SNAME(NULL) – NULL을 반환합니다.
- 테이블 반환 함수 변환이 지원됩니다. 테이블 반환 함수는 테이블을 반환하며 쿼리에서 테이블을 대신할 수 있습니다.
- PATINDEX에서는 유효한 모든 텍스트 및 문자 데이터 형식의 지정된 표현식에서 패턴이 처음으로 발생하는 시작 위치를 반환하고, 패턴을 찾을 수 없는 경우 0을 반환합니다. SQL Server에서 Amazon RDS for PostgreSQL로 변환하는 경우 AWS SCT는 PATINDEX를 사용하는 애플리케이션 코드를 aws_sqlserver_ext.patindex(<pattern character>, <expression character varying>)로 대체합니다.
- SQL Server에서 사용자 정의 테이블 유형은 테이블 구조의 정의를 나타내는 유형입니다. 사용자 정의 테이블 유형을 사용하여 저장 프로시저 또는 함수에 테이블-값 파라미터를 선언합니다. 또한 사용자 정의 테이블 유형을 사용하여 저장 프로시저 또는 함수의 본문 또는 배치에 사용할 테이블 변수를 선언할 수 있습니다. AWS SCT는 임시 테이블을 생성하여 PostgreSQL에서 이 유형을 에뮬레이션했습니다.

SQL Server에서 PostgreSQL로 변환하는 경우 AWS SCT는 SQL Server 시스템 객체를 PostgreSQL에서 인식 가능한 객체로 변환합니다. 다음 표에서는 시스템 객체가 어떻게 변환되는지 보여줍니다.

MS SQL Server 사용 사례	PostgreSQL 대체
SYS.SCHEMAS	AWS_SQLSERVER_EXT.SYS_SCHEMAS
SYS.TABLES	AWS_SQLSERVER_EXT.SYS_TABLES
SYS.VIEWS	AWS_SQLSERVER_EXT.SYS_VIEWS

MS SQL Server 사용 사례	PostgreSQL 대체
SYS.ALL_VIEWS	AWS_SQLSERVER_EXT.SYS_ALL_VIEWS
SYS.TYPES	AWS_SQLSERVER_EXT.SYS_TYPES
SYS.COLUMNS	AWS_SQLSERVER_EXT.SYS_COLUMNS
SYS.ALL_COLUMNS	AWS_SQLSERVER_EXT.SYS_ALL_COLUMNS
SYS.FOREIGN_KEYS	AWS_SQLSERVER_EXT.SYS_FOREIGN_KEYS
SYS.SYSFOREIGNKEYS	AWS_SQLSERVER_EXT.SYS_SYSFOREIGNKEYS
SYS.FOREIGN_KEY_COLUMNS	AWS_SQLSERVER_EXT.SYS_FOREIGN_KEY_COLUMNS
SYS.KEY_CONSTRAINTS	AWS_SQLSERVER_EXT.SYS_KEY_CONSTRAINTS
SYS.IDENTITY_COLUMNS	AWS_SQLSERVER_EXT.SYS_IDENTITY_COLUMNS
SYS.PROCEDURES	AWS_SQLSERVER_EXT.SYS_PROCEDURES
SYS.INDEXES	AWS_SQLSERVER_EXT.SYS_INDEXES
SYS.SYSINDEXES	AWS_SQLSERVER_EXT.SYS_SYSINDEXES
SYS.OBJECTS	AWS_SQLSERVER_EXT.SYS_OBJECTS
SYS.ALL_OBJECTS	AWS_SQLSERVER_EXT.SYS_ALL_OBJECTS
SYS.SYSOBJECTS	AWS_SQLSERVER_EXT.SYS_SYSOBJECTS
SYS.SQL_MODULES	AWS_SQLSERVER_EXT.SYS_SQL_MODULES
SYS.DATABASES	AWS_SQLSERVER_EXT.SYS_DATABASES
INFORMATION_SCHEMA.SCHEMATA	AWS_SQLSERVER_EXT.INFORMATION_SCHEMA_SCHEMATA

MS SQL Server 사용 사례	PostgreSQL 대체
INFORMATION_SCHEMA.VIEWS	AWS_SQLSERVER_EXT.INFORMATION_SCHEMA_VIEWS
INFORMATION_SCHEMA.TABLES	AWS_SQLSERVER_EXT.INFORMATION_SCHEMA_TABLES
INFORMATION_SCHEMA.COLUMNS	AWS_SQLSERVER_EXT.INFORMATION_SCHEMA_COLUMNS
INFORMATION_SCHEMA.CHECK_CONSTRAINTS	AWS_SQLSERVER_EXT.INFORMATION_SCHEMA_CHECK_CONSTRAINTS
INFORMATION_SCHEMA.REFERENTIAL_CONSTRAINTS	AWS_SQLSERVER_EXT.INFORMATION_SCHEMA_REFERENTIAL_CONSTRAINTS
INFORMATION_SCHEMA.TABLE_CONSTRAINTS	AWS_SQLSERVER_EXT.INFORMATION_SCHEMA_TABLE_CONSTRAINTS
INFORMATION_SCHEMA.KEY_COLUMN_USAGE	AWS_SQLSERVER_EXT.INFORMATION_SCHEMA_KEY_COLUMN_USAGE
INFORMATION_SCHEMA.CONSTRAINT_TABLE_USAGE	AWS_SQLSERVER_EXT.INFORMATION_SCHEMA_CONSTRAINT_TABLE_USAGE
INFORMATION_SCHEMA.CONSTRAINT_COLUMN_USAGE	AWS_SQLSERVER_EXT.INFORMATION_SCHEMA_CONSTRAINT_COLUMN_USAGE
INFORMATION_SCHEMA.ROUTINES	AWS_SQLSERVER_EXT.INFORMATION_SCHEMA_ROUTINES
SYS.SYSPROCESSES	AWS_SQLSERVER_EXT.SYS_SYSPROCESSES
sys.system_objects	AWS_SQLSERVER_EXT.SYS_SYSTEM_OBJECTS

AWS SCT 확장 팩을 사용하여 PostgreSQL에서 SQL Server 에이전트 에뮬레이션

SQL Server 에이전트는 SQL Server 작업을 실행하는 Microsoft Windows 서비스입니다. SQL Server 에이전트는 일정에 따라, 특정 이벤트 발생 시 또는 필요에 따라 작업을 실행합니다. SQL Server 에이전트에 대한 자세한 내용은 Microsoft 기술 문서를 참조하세요.

PostgreSQL은 SQL Server 에이전트와 동일한 기능을 제공하지 않습니다. SQL Server 에이전트 기능을 에뮬레이션하기 위해 AWS SCT는 확장 팩을 생성합니다. 이 확장 팩은 AWS Lambda와 Amazon CloudWatch를 사용합니다. AWS Lambda는 일정을 관리하고 작업을 실행하는 데 사용되는 인터페이스를 구현합니다. Amazon CloudWatch는 예약 규칙을 유지 관리합니다.

AWS Lambda와 Amazon CloudWatch는 JSON 파라미터를 사용하여 상호 작용합니다. 이 JSON 파라미터의 구조는 다음과 같습니다.

```
{
    "mode": mode,
    "parameters": {
        list of parameters
    },
    "callback": procedure name
}
```

이전 예제에서 *mode*는 작업 유형이고 *list of parameters*는 작업 유형에 따라 달라지는 파라미터 집합입니다. 또한 *procedure name*은 작업이 완료된 후 실행되는 프로시저의 이름입니다.

AWS SCT는 하나의 Lambda 함수를 사용하여 작업을 제어하고 실행합니다. CloudWatch 규칙은 작업 실행을 시작하고, 작업을 시작하는 데 필요한 정보를 제공합니다. CloudWatch 규칙이 트리거되면 규칙의 파라미터를 사용하여 Lambda 함수를 시작합니다.

프로시저를 호출하는 단순 작업을 생성하려면 다음 형식을 사용합니다.

```
{
    "mode": "run_job",
    "parameters": {
        "vendor": "mysql",
        "cmd": "lambda_db.nightly_job"
    }
}
```

여러 단계가 포함된 작업을 생성하려면 다음 형식을 사용합니다.

```
{
    "mode": "run_job",
    "parameters": {
        "job_name": "Job1",
        "enabled": "true",
        "start_step_id": 1,
        "notify_level_email": [0|1|2|3],
        "notify_email": email,
        "delete_level": [0|1|2|3],
        "job_callback": "ProcCallBackJob(job_name, code, message)",
        "step_callback": "ProcCallBackStep(job_name, step_id, code, message)"
    },
    "steps": [
        {
            "id":1,
            "cmd": "ProcStep1",
            "cmdexec_success_code": 0,
            "on_success_action": [|2|3|4],
            "on_success_step_id": 1,
            "on_fail_action": 0,
            "on_fail_step_id": 0,
            "retry_attempts": number,
            "retry_interval": number
        },
        {
            "id":2,
            "cmd": "ProcStep2",
            "cmdexec_success_code": 0,
            "on_success_action": [1|2|3|4],
            "on_success_step_id": 0,
            "on_fail_action": 0,
            "on_fail_step_id": 0,
            "retry_attempts": number,
            "retry_interval": number
        },
        ...
]
}
```

PostgreSQL에서 SQL Server 에이전트 동작을 에뮬레이션하기 위해 AWS SCT 확장 팩은 다음 테이블과 프로시저도 생성합니다.

PostgreSQL에서 SQL Server 에이전트를 에뮬레이션하는 테이블

SQL Server 에이전트를 에뮬레이션하기 위해 확장 팩은 다음 테이블을 사용합니다.

sysjobs

작업에 대한 정보를 저장합니다.

sysjobsteps

작업의 단계에 대한 정보를 저장합니다.

sysschedules

작업 예약에 대한 정보를 저장합니다.

sysjobschedules

개별 작업의 예약 정보를 저장합니다.

sysjobhistory

예약된 작업의 실행 관련 정보를 저장합니다.

PostgreSQL에서 SQL Server 에이전트를 에뮬레이션하는 프로시저

SQL Server 에이전트를 에뮬레이션하기 위해 확장 팩은 다음 프로시저를 사용합니다.

sp_add_job

새 작업을 추가합니다.

sp_add_jobstep

작업에 단계를 추가합니다.

sp_add_schedule

Amazon CloudWatch에 새 예약 규칙을 생성합니다. 여러 작업에서 이 예약 일정을 사용할 수 있습니다.

sp_attach_schedule

선택한 작업의 예약을 설정합니다.

sp_add_jobschedule

Amazon CloudWatch에서 작업에 대한 예약 규칙을 생성하고 이 규칙의 대상을 설정합니다.

sp_update_job

이전에 생성한 작업의 속성을 업데이트합니다.

sp_update_jobstep

작업의 단계 속성을 업데이트합니다.

sp_update_schedule

Amazon CloudWatch에서 예약 규칙의 속성을 업데이트합니다.

sp_update_jobschedule

지정된 작업의 예약 속성을 업데이트합니다.

sp_delete_job

작업을 삭제합니다.

sp_delete_jobstep

작업에서 작업 단계를 삭제합니다.

sp_delete_schedule

일정을 삭제합니다.

sp_delete_jobschedule

Amazon CloudWatch에서 지정된 작업의 예약 규칙을 삭제합니다.

sp_detach_schedule

예약과 작업 간의 연결을 제거합니다.

get_jobs, update_job

AWS Elastic Beanstalk과 상호 작용하는 내부 프로시저입니다.

sp_verify_job_date, sp_verify_job_time, sp_verify_job, sp_verify_jobstep, sp_verify_schedule, sp_verify_job_identifiers, sp_verify_schedule_identifiers

설정을 확인하는 내부 프로시저입니다.

PostgreSQL에서 SQL Server 에이전트를 에뮬레이션하는 프로시저의 구문

확장 팩의 `aws_sqlserver_ext.sp_add_job` 프로시저는 `msdb.dbo.sp_add_job` 프로시저를 에뮬레이션합니다. 소스 SQL Server 에이전트 프로시저에 대한 자세한 내용은 Microsoft 기술 문서를 참조하세요.

```
par_job_name varchar,
par_enabled smallint = 1,
par_description varchar = NULL::character varying,
par_start_step_id integer = 1,
par_category_name varchar = NULL::character varying,
par_category_id integer = NULL::integer,
par_owner_login_name varchar = NULL::character varying,
par_notify_level_eventlog integer = 2,
par_notify_level_email integer = 0,
par_notify_level_netsend integer = 0,
par_notify_level_page integer = 0,
par_notify_email_operator_name varchar = NULL::character varying,
par_notify_netsend_operator_name varchar = NULL::character varying,
par_notify_page_operator_name varchar = NULL::character varying,
par_delete_level integer = 0,
inout par_job_id integer = NULL::integer,
par_originating_server varchar = NULL::character varying,
out returncode integer
```

확장 팩의 `aws_sqlserver_ext.sp_add_jobstep` 프로시저는 `msdb.dbo.sp_add_jobstep` 프로시저를 에뮬레이션합니다. 소스 SQL Server 에이전트 프로시저에 대한 자세한 내용은 Microsoft 기술 문서를 참조하세요.

```
par_job_id integer = NULL::integer,
par_job_name varchar = NULL::character varying,
par_step_id integer = NULL::integer,
par_step_name varchar = NULL::character varying,
par_subsystem varchar = 'TSQL'::bpchar,
par_command text = NULL::text,
par_additional_parameters text = NULL::text,
par_cmdexec_success_code integer = 0,
par_on_success_action smallint = 1,
par_on_success_step_id integer = 0,
par_on_fail_action smallint = 2,
par_on_fail_step_id integer = 0,
par_server varchar = NULL::character varying,
par_database_name varchar = NULL::character varying,
par_database_user_name varchar = NULL::character varying,
par_retry_attempts integer = 0,
par_retry_interval integer = 0,
par_os_run_priority integer = 0,
par_output_file_name varchar = NULL::character varying,
```

```
par_flags integer = 0,
par_proxy_id integer = NULL::integer,
par_proxy_name varchar = NULL::character varying,
inout par_step_uid char = NULL::bpchar,
out returncode integer
```

확장 팩의 `aws_sqlserver_ext.sp_add_schedule` 프로시저는 `msdb.dbo.sp_add_schedule` 프로시저를 에뮬레이션합니다. 소스 SQL Server 에이전트 프로시저에 대한 자세한 내용은 Microsoft 기술 문서를 참조하세요.

```
par_schedule_name varchar,
par_enabled smallint = 1,
par_freq_type integer = 0,
par_freq_interval integer = 0,
par_freq_subday_type integer = 0,
par_freq_subday_interval integer = 0,
par_freq_relative_interval integer = 0,
par_freq_recurrence_factor integer = 0,
par_active_start_date integer = NULL::integer,
par_active_end_date integer = 99991231,
par_active_start_time integer = 0,
par_active_end_time integer = 235959,
par_owner_login_name varchar = NULL::character varying,
*inout par_schedule_uid char = NULL::bpchar,*
inout par_schedule_id integer = NULL::integer,
par_originating_server varchar = NULL::character varying,
out returncode integer
```

확장 팩의 `aws_sqlserver_ext.sp_attach_schedule` 프로시저는 `msdb.dbo.sp_attach_schedule` 프로시저를 에뮬레이션합니다. 소스 SQL Server 에이전트 프로시저에 대한 자세한 내용은 Microsoft 기술 문서를 참조하세요.

```
par_job_id integer = NULL::integer,
par_job_name varchar = NULL::character varying,
par_schedule_id integer = NULL::integer,
par_schedule_name varchar = NULL::character varying,
par_automatic_post smallint = 1,
out returncode integer
```

확장 팩의 aws_sqlserver_ext.sp_add_jobschedule 프로시저는 msdb.dbo.sp_add_jobschedule 프로시저를 에뮬레이션합니다. 소스 SQL Server 에이전트 프로시저에 대한 자세한 내용은 Microsoft 기술 문서를 참조하세요.

```
par_job_id integer = NULL::integer,
par_job_name varchar = NULL::character varying,
par_name varchar = NULL::character varying,
par_enabled smallint = 1,
par_freq_type integer = 1,
par_freq_interval integer = 0,
par_freq_subday_type integer = 0,
par_freq_subday_interval integer = 0,
par_freq_relative_interval integer = 0,
par_freq_recurrence_factor integer = 0,
par_active_start_date integer = NULL::integer,
par_active_end_date integer = 99991231,
par_active_start_time integer = 0,
par_active_end_time integer = 235959,
inout par_schedule_id integer = NULL::integer,
par_automatic_post smallint = 1,
inout par_schedule_uid char = NULL::bpchar,
out returncode integer
```

확장 팩의 aws_sqlserver_ext.sp_delete_job 프로시저는 msdb.dbo.sp_delete_job 프로시저를 에뮬레이션합니다. 소스 SQL Server 에이전트 프로시저에 대한 자세한 내용은 Microsoft 기술 문서를 참조하세요.

```
par_job_id integer = NULL::integer,
par_job_name varchar = NULL::character varying,
par_originating_server varchar = NULL::character varying,
par_delete_history smallint = 1,
par_delete_unused_schedule smallint = 1,
out returncode integer
```

확장 팩의 aws_sqlserver_ext.sp_delete_jobstep 프로시저는 msdb.dbo.sp_delete_jobstep 프로시저를 에뮬레이션합니다. 소스 SQL Server 에이전트 프로시저에 대한 자세한 내용은 Microsoft 기술 문서를 참조하세요.

```
par_job_id integer = NULL::integer,
par_job_name varchar = NULL::character varying,
par_step_id integer = NULL::integer,
```

```
out returncode integer
```

확장 팩의 aws_sqlserver_ext.sp_delete_jobschedule 프로시저는 msdb.dbo.sp_delete_jobschedule 프로시저를 에뮬레이션합니다. 소스 SQL Server 에이전트 프로시저에 대한 자세한 내용은 Microsoft 기술 문서를 참조하세요.

```
par_job_id integer = NULL::integer,
par_job_name varchar = NULL::character varying,
par_name varchar = NULL::character varying,
par_keep_schedule integer = 0,
par_automatic_post smallint = 1,
out returncode integer
```

확장 팩의 aws_sqlserver_ext.sp_delete_schedule 프로시저는 msdb.dbo.sp_delete_schedule 프로시저를 에뮬레이션합니다. 소스 SQL Server 에이전트 프로시저에 대한 자세한 내용은 Microsoft 기술 문서를 참조하세요.

```
par_schedule_id integer = NULL::integer,
par_schedule_name varchar = NULL::character varying,
par_force_delete smallint = 0,
par_automatic_post smallint = 1,
out returncode integer
```

확장 팩의 aws_sqlserver_ext.sp_detach_schedule 프로시저는 msdb.dbo.sp_detach_schedule 프로시저를 에뮬레이션합니다. 소스 SQL Server 에이전트 프로시저에 대한 자세한 내용은 Microsoft 기술 문서를 참조하세요.

```
par_job_id integer = NULL::integer,
par_job_name varchar = NULL::character varying,
par_schedule_id integer = NULL::integer,
par_schedule_name varchar = NULL::character varying,
par_delete_unused_schedule smallint = 0,
par_automatic_post smallint = 1,
out returncode integer
```

확장 팩의 aws_sqlserver_ext.sp_update_job 프로시저는 msdb.dbo.sp_update_job 프로시저를 에뮬레이션합니다. 소스 SQL Server 에이전트 프로시저에 대한 자세한 내용은 Microsoft 기술 문서를 참조하세요.

```
par_job_id integer = NULL::integer
```

```
par_job_name varchar = NULL::character varying
par_new_name varchar = NULL::character varying
par_enabled smallint = NULL::smallint
par_description varchar = NULL::character varying
par_start_step_id integer = NULL::integer
par_category_name varchar = NULL::character varying
par_owner_login_name varchar = NULL::character varying
par_notify_level_eventlog integer = NULL::integer
par_notify_level_email integer = NULL::integer
par_notify_level_netsend integer = NULL::integer
par_notify_level_page integer = NULL::integer
par_notify_email_operator_name varchar = NULL::character varying
par_notify_netsend_operator_name varchar = NULL::character varying
par_notify_page_operator_name varchar = NULL::character varying
par_delete_level integer = NULL::integer
par_automatic_post smallint = 1
out returncode integer
```

확장 팩의 `aws_sqlserver_ext.sp_update_jobschedule` 프로시저는 `msdb.dbo.sp_update_jobschedule` 프로시저를 에뮬레이션합니다. 소스 SQL Server 에이전트 프로시저에 대한 자세한 내용은 Microsoft 기술 문서를 참조하세요.

```
par_job_id integer = NULL::integer
par_job_name varchar = NULL::character varying
par_name varchar = NULL::character varying
par_new_name varchar = NULL::character varying
par_enabled smallint = NULL::smallint
par_freq_type integer = NULL::integer
par_freq_interval integer = NULL::integer
par_freq_subday_type integer = NULL::integer
par_freq_subday_interval integer = NULL::integer
par_freq_relative_interval integer = NULL::integer
par_freq_recurrence_factor integer = NULL::integer
par_active_start_date integer = NULL::integer
par_active_end_date integer = NULL::integer
par_active_start_time integer = NULL::integer
                par_active_end_time integer = NULL::integer
par_automatic_post smallint = 1
out returncode integer
```

확장 팩의 aws_sqlserver_ext.sp_update_jobstep 프로시저는 msdb.dbo.sp_update_jobstep 프로시저를 에뮬레이션합니다. 소스 SQL Server 에이전트 프로시저에 대한 자세한 내용은 Microsoft 기술 문서를 참조하세요.

```
par_job_id integer = NULL::integer
par_job_name varchar = NULL::character varying
par_step_id integer = NULL::integer
par_step_name varchar = NULL::character varying
par_subsystem varchar = NULL::character varying
par_command text = NULL::text
par_additional_parameters text = NULL::text
par_cmdexec_success_code integer = NULL::integer
par_on_success_action smallint = NULL::smallint
par_on_success_step_id integer = NULL::integer
par_on_fail_action smallint = NULL::smallint
par_on_fail_step_id integer = NULL::integer
par_server varchar = NULL::character varying
par_database_name varchar = NULL::character varying
par_database_user_name varchar = NULL::character varying
par_retry_attempts integer = NULL::integer
par_retry_interval integer = NULL::integer
par_os_run_priority integer = NULL::integer
par_output_file_name varchar = NULL::character varying
par_flags integer = NULL::integer
par_proxy_id integer = NULL::integer
par_proxy_name varchar = NULL::character varying
out returncode integer
```

확장 팩의 aws_sqlserver_ext.sp_update_schedule 프로시저는 msdb.dbo.sp_update_schedule 프로시저를 에뮬레이션합니다. 소스 SQL Server 에이전트 프로시저에 대한 자세한 내용은 Microsoft 기술 문서를 참조하세요.

```
par_schedule_id integer = NULL::integer
par_name varchar = NULL::character varying
par_new_name varchar = NULL::character varying
par_enabled smallint = NULL::smallint
par_freq_type integer = NULL::integer
par_freq_interval integer = NULL::integer
par_freq_subday_type integer = NULL::integer
par_freq_subday_interval integer = NULL::integer
par_freq_relative_interval integer = NULL::integer
par_freq_recurrence_factor integer = NULL::integer
```

```
par_active_start_date integer = NULL::integer
par_active_end_date integer = NULL::integer
par_active_start_time integer = NULL::integer
par_active_end_time integer = NULL::integer
par_owner_login_name varchar = NULL::character varying
par_automatic_post smallint = 1
out returncode integer
```

PostgreSQL에서 SQL Server 에이전트를 에뮬레이션하는 프로시저 사용 예제

새 작업을 추가하려면 다음과 같이 `aws_sqlserver_ext.sp_add_job` 프로시저를 사용합니다.

```
SELECT * FROM aws_sqlserver_ext.sp_add_job (
    par_job_name := 'test_job',
    par_enabled := 1::smallint,
    par_start_step_id := 1::integer,
    par_category_name := '[Uncategorized (Local)]',
    par_owner_login_name := 'sa');
```

새 작업 단계를 추가하려면 다음과 같이 `aws_sqlserver_ext.sp_add_jobstep` 프로시저를 사용합니다.

```
SELECT * FROM aws_sqlserver_ext.sp_add_jobstep (
    par_job_name := 'test_job',
    par_step_id := 1::smallint,
    par_step_name := 'test_job_step1',
    par_subsystem := 'TSQL',
    par_command := 'EXECUTE [dbo].[PROC_TEST_JOB_STEP1];',
    par_server := NULL,
    par_database_name := 'GOLD_TEST_SS');
```

단순 일정을 추가하려면 다음과 같이 `aws_sqlserver_ext.sp_add_schedule` 프로시저를 사용합니다.

```
SELECT * FROM aws_sqlserver_ext.sp_add_schedule(
    par_schedule_name := 'RunOnce',
    par_freq_type := 1,
    par_active_start_time := 233000);
```

작업 일정을 설정하려면 다음과 같이 `aws_sqlserver_ext.sp_attach_schedule` 프로시저를 사용합니다.

```
SELECT * FROM aws_sqlserver_ext.sp_attach_schedule (
    par_job_name := 'test_job',
    par_schedule_name := 'NightlyJobs');
```

작업에 대한 예약을 생성하려면 다음과 같이 aws_sqlserver_ext.sp_add_jobschedule 프로시저를 사용합니다.

```
SELECT * FROM aws_sqlserver_ext.sp_add_jobschedule (
    par_job_name := 'test_job2',
    par_name := 'test_schedule2',
    par_enabled := 1::smallint,
    par_freq_type := 4,
    par_freq_interval := 1,
    par_freq_subday_type := 4,
    par_freq_subday_interval := 1,
    par_freq_relative_interval := 0,
    par_freq_recurrence_factor := 0,
    par_active_start_date := 20100801,
    par_active_end_date := 99991231,
    par_active_start_time := 0,
    par_active_end_time := 0);
```

PostgreSQL에서 SQL Server 에이전트를 에뮬레이션하기 위한 사용 사례 예제

소스 데이터베이스 코드에서 SQL Server 에이전트를 사용하여 작업을 실행하는 경우 AWS SCT에 대한 SQL Server-PostgreSQL 확장 팩을 사용하여 이 코드를 PostgreSQL로 변환할 수 있습니다. 확장 팩은 AWS Lambda 함수를 사용하여 SQL Server 에이전트의 동작을 에뮬레이션합니다.

새 AWS Lambda 함수를 생성하거나 기존 함수를 등록할 수 있습니다.

새로운 AWS Lambda 함수를 생성하려면

1. AWS SCT의 대상 데이터베이스 트리에서 컨텍스트(마우스 오른쪽 버튼 클릭) 메뉴를 열고 Apply extension pack for를 선택한 다음, PostgreSQL을 선택합니다.

   확장 팩 마법사가 표시됩니다.
2. SQL Server Agent emulation service 탭에서 다음을 수행합니다.

   - AWS Lambda 함수 생성을 선택합니다.
   - Database login에 대상 데이터베이스 사용자의 이름을 입력합니다.
   - 데이터베이스 암호에 이전 단계에서 입력한 사용자 이름의 암호를 입력합니다.

- Python library folder에 Python 라이브러리 폴더의 경로를 입력합니다.
- AWS Lambda 함수 생성을 선택하고 다음을 선택합니다.

이전에 배포된 AWS Lambda 함수를 등록하려면

- 대상 데이터베이스에서 다음 스크립트를 실행합니다.

```
SELECT
    FROM aws_sqlserver_ext.set_service_setting(
        p_service := 'JOB',
        p_setting := 'LAMBDA_ARN',
        p_value := ARN)
```

이전 예제에서 *ARN*은 배포된 AWS Lambda 함수의 Amazon 리소스 이름(ARN)입니다.

다음 예제에서는 한 단계로 구성된 단순 작업을 생성합니다. 이 작업은 5분마다 이전에 생성된 `job_example` 함수를 실행합니다. 이 함수는 `job_example_table` 테이블에 레코드를 삽입합니다.

이 단순 작업을 생성하려면

1. 다음과 같이 `aws_sqlserver_ext.sp_add_job` 함수를 사용하여 작업을 생성합니다.

```
SELECT
    FROM aws_sqlserver_ext.sp_add_job (
        par_job_name := 'test_simple_job');
```

2. 다음과 같이 `aws_sqlserver_ext.sp_add_jobstep` 함수를 사용하여 작업 단계를 생성합니다.

```
SELECT
    FROM aws_sqlserver_ext.sp_add_jobstep (
        par_job_name := 'test_simple_job',
        par_step_name := 'test_simple_job_step1',
        par_command := 'PERFORM job_simple_example;');
```

작업 단계는 함수가 수행하는 작업을 지정합니다.

3. 다음과 같이 `aws_sqlserver_ext.sp_add_jobschedule` 함수를 사용하여 작업에 대한 스케줄러를 생성합니다.

```
SELECT
    FROM aws_sqlserver_ext.sp_add_jobschedule (
        par_job_name := 'test_simple_job',
        par_name := 'test_schedule',
        par_freq_type := 4, /* Daily */
        par_freq_interval := 1, /* frequency_interval is unused */
        par_freq_subday_type := 4, /* Minutes */
        par_freq_subday_interval := 5 /* 5 minutes */);
```

작업 단계는 함수가 수행하는 작업을 지정합니다.

이 작업을 삭제하려면 다음과 같이 `aws_sqlserver_ext.sp_delete_job` 함수를 사용합니다.

```
PERFORM aws_sqlserver_ext.sp_delete_job(
    par_job_name := 'PeriodicJob1'::character varying,
    par_delete_history := 1::smallint,
    par_delete_unused_schedule := 1::smallint);
```

AWS SCT 확장 팩을 사용하여 PostgreSQL에서 SQL Server Database Mail 에뮬레이션

SQL Server Database Mail을 사용하여 SQL Server 데이터베이스 엔진 또는 Azure SQL Managed Instance에서 사용자에게 이메일 메시지를 보낼 수 있습니다. 이러한 이메일 메시지는 쿼리 결과를 포함하거나 네트워크에 있는 리소스의 파일을 포함할 수 있습니다. SQL Server Database Mail에 대한 자세한 내용은 Microsoft 기술 문서를 참조하세요.

PostgreSQL은 SQL Server Database Mail과 동일한 기능을 제공하지 않습니다. SQL Server Database Mail 기능을 에뮬레이션하기 위해 AWS SCT는 확장 팩을 생성합니다. 이 확장 팩은 AWS Lambda와 Amazon Simple Email Service(Amazon SES)를 사용합니다. AWS Lambda는 Amazon SES 이메일 전송 서비스와 상호 작용할 수 있는 인터페이스를 사용자에게 제공합니다. 이 상호 작용을 설정하려면 Lambda 함수의 Amazon 리소스 이름(ARN)을 추가합니다.

새 이메일 계정의 경우 다음 명령을 사용합니다.

```
do
$$
begin
PERFORM sysmail_add_account_sp (
    par_account_name :='your_account_name',
    par_email_address := 'your_account_email',
```

```
    par_display_name := 'your_account_display_name',
    par_mailserver_type := 'AWSLAMBDA'
    par_mailserver_name := 'ARN'
);
end;
$$ language plpgsql;
```

Lambda 함수의 ARN을 기존 이메일 계정에 추가하려면 다음 명령을 사용합니다.

```
do
$$
begin
PERFORM sysmail_update_account_sp (
    par_account_name :='existind_account_name',
    par_mailserver_type := 'AWSLAMBDA'
    par_mailserver_name := 'ARN'
);
end;
$$ language plpgsql;
```

이전 예제에서 *ARN*은 Lambda 함수의 ARN입니다.

PostgreSQL에서 SQL Server Database Mail 동작을 에뮬레이션하기 위해 AWS SCT 확장 팩은 다음과 같은 테이블, 보기 및 프로시저를 사용합니다.

PostgreSQL에서 SQL Server Database Mail을 에뮬레이션하는 테이블

SQL Server Database Mail을 에뮬레이션하기 위해 확장 팩은 다음 테이블을 사용합니다.

sysmail_account

이메일 계정에 대한 정보를 저장합니다.

sysmail_profile

사용자 프로필에 대한 정보를 저장합니다.

sysmail_server

이메일 서버에 대한 정보를 저장합니다.

sysmail_mailitems

이메일 메시지 목록을 저장합니다.

sysmail_attachment

각 이메일 첨부 파일마다 하나의 행을 포함합니다.

sysmail_log

이메일 메시지 전송에 대한 서비스 정보를 저장합니다.

sysmail_profileaccount

사용자 프로필 및 이메일 계정에 대한 정보를 저장합니다.

PostgreSQL에서 SQL Server Database Mail을 에뮬레이션하는 보기

SQL Server Database Mail을 에뮬레이션하기 위해 AWS SCT는 PostgreSQL 데이터베이스에서 다음과 같은 보기를 생성하여 호환성을 보장합니다. 확장 팩에서는 이러한 보기를 사용하지 않지만 변환된 코드는 이러한 보기를 쿼리할 수 있습니다.

sysmail_allitems

모든 이메일 목록이 포함됩니다.

sysmail_faileditems

전송할 수 없는 이메일 목록이 포함됩니다.

sysmail_sentitems

보낸 이메일 목록이 포함됩니다.

sysmail_unsentitems

아직 전송되지 않은 이메일 목록이 포함됩니다.

sysmail_mailattachments

첨부 파일 목록이 포함됩니다.

PostgreSQL에서 SQL Server Database Mail을 에뮬레이션하는 프로시저

SQL Server Database Mail을 에뮬레이션하기 위해 확장 팩은 다음 프로시저를 사용합니다.

sp_send_dbmail

지정된 수신자에게 이메일을 보냅니다.

sysmail_add_profile_sp

새 사용자 프로필을 생성합니다.

sysmail_add_account_sp

SMTP(Simple Mail Transfer Protocol) 보안 인증 정보 등과 같은 정보를 저장하는 새로운 이메일 계정을 생성합니다.

sysmail_add_profileaccount_sp

지정된 사용자 프로필에 이메일 계정을 추가합니다.

sysmail_update_profile_sp

설명, 이름 등과 같은 사용자 프로필의 속성을 변경합니다.

sysmail_update_account_sp

기존 이메일 계정의 정보를 변경합니다.

sysmail_update_profileaccount_sp

지정된 사용자 프로필의 이메일 계정 정보를 업데이트합니다.

sysmail_delete_profileaccount_sp

지정된 사용자 프로필에서 이메일 계정을 제거합니다.

sysmail_delete_account_sp

이메일 계정을 삭제합니다.

sysmail_delete_profile_sp

사용자 프로필을 삭제합니다.

sysmail_delete_mailitems_sp

내부 테이블에서 이메일을 삭제합니다.

sysmail_help_profile_sp

사용자 프로필에 대한 정보를 표시합니다.

sysmail_help_account_sp

이메일 계정에 대한 정보를 표시합니다.

sysmail_help_profileaccount_sp

사용자 프로필과 연결된 이메일 계정 관련 정보를 표시합니다.

sysmail_dbmail_json

AWS Lambda 함수에 대한 JSON 요청을 생성하는 내부 프로시저입니다.

sysmail_verify_profile_sp, sysmail_verify_account_sp, sysmail_verify_addressparams_sp

설정을 확인하는 내부 프로시저입니다.

sp_get_dbmail, sp_set_dbmail, sysmail_dbmail_xml

더 이상 사용되지 않는 내부 프로시저입니다.

PostgreSQL에서 SQL Server Database Mail을 에뮬레이션하는 프로시저의 구문

확장 팩의 `aws_sqlserver_ext.sp_send_dbmail` 프로시저는 `msdb.dbo.sp_send_dbmail` 프로시저를 에뮬레이션합니다. 소스 SQL Server Database Mail 프로시저에 대한 자세한 내용은 Microsoft 기술 문서를 참조하세요.

```
par_profile_name varchar = NULL::character varying,
par_recipients text = NULL::text,
par_copy_recipients text = NULL::text,
par_blind_copy_recipients text = NULL::text,
par_subject varchar = NULL::character varying,
par_body text = NULL::text,
par_body_format varchar = NULL::character varying,
par_importance varchar = 'NORMAL'::character varying,
par_sensitivity varchar = 'NORMAL'::character varying,
par_file_attachments text = NULL::text,
par_query text = NULL::text,
par_execute_query_database varchar = NULL::character varying,
par_attach_query_result_as_file smallint = 0,
par_query_attachment_filename varchar = NULL::character varying,
par_query_result_header smallint = 1,
par_query_result_width integer = 256,
par_query_result_separator VARCHAR = ' '::character varying,
par_exclude_query_output smallint = 0,
par_append_query_error smallint = 0,
par_query_no_truncate smallint = 0,
par_query_result_no_padding smallint = 0,
out par_mailitem_id integer,
par_from_address text = NULL::text,
par_reply_to text = NULL::text,
out returncode integer
```

확장 팩의 aws_sqlserver_ext.sysmail_delete_mailitems_sp 프로시저는 msdb.dbo.sysmail_delete_mailitems_sp 프로시저를 에뮬레이션합니다. 소스 SQL Server Database Mail 프로시저에 대한 자세한 내용은 Microsoft 기술 문서를 참조하세요.

```
par_sent_before timestamp = NULL::timestamp without time zone,
par_sent_status varchar = NULL::character varying,
out returncode integer
```

확장 팩의 aws_sqlserver_ext.sysmail_add_profile_sp 프로시저는 msdb.dbo.sysmail_add_profile_sp 프로시저를 에뮬레이션합니다. 소스 SQL Server Database Mail 프로시저에 대한 자세한 내용은 Microsoft 기술 문서를 참조하세요.

```
par_profile_name varchar,
par_description varchar = NULL::character varying,
out par_profile_id integer,
out returncode integer
```

확장 팩의 aws_sqlserver_ext.sysmail_add_account_sp 프로시저는 msdb.dbo.sysmail_add_account_sp 프로시저를 에뮬레이션합니다. 소스 SQL Server Database Mail 프로시저에 대한 자세한 내용은 Microsoft 기술 문서를 참조하세요.

```
par_account_name varchar
par_email_address varchar
par_display_name varchar = NULL::character varying
par_replyto_address varchar = NULL::character varying
par_description varchar = NULL::character varying
par_mailserver_name varchar = NULL::character varying
par_mailserver_type varchar = 'SMTP'::bpchar
par_port integer = 25
par_username varchar = NULL::character varying
par_password varchar = NULL::character varying
par_use_default_credentials smallint = 0
par_enable_ssl smallint = 0
out par_account_id integer
out returncode integer
```

확장 팩의 aws_sqlserver_ext.sysmail_add_profileaccount_sp 프로시저는 msdb.dbo.sysmail_add_profileaccount_sp 프로시저를 에뮬레이션합니다. 소스 SQL Server Database Mail 프로시저에 대한 자세한 내용은 Microsoft 기술 문서를 참조하세요.

```
par_profile_id integer = NULL::integer,
par_profile_name varchar = NULL::character varying,
par_account_id integer = NULL::integer,
par_account_name varchar = NULL::character varying,
par_sequence_number integer = NULL::integer,
out returncode integer
```

확장 팩의 `aws_sqlserver_ext.sysmail_help_profile_sp` 프로시저는 `msdb.dbo.sysmail_help_profile_sp` 프로시저를 에뮬레이션합니다. 소스 SQL Server Database Mail 프로시저에 대한 자세한 내용은 Microsoft 기술 문서를 참조하세요.

```
par_profile_id integer = NULL::integer,
par_profile_name varchar = NULL::character varying,
out returncode integer
```

확장 팩의 `aws_sqlserver_ext.sysmail_update_profile_sp` 프로시저는 `msdb.dbo.sysmail_update_profile_sp` 프로시저를 에뮬레이션합니다. 소스 SQL Server Database Mail 프로시저에 대한 자세한 내용은 Microsoft 기술 문서를 참조하세요.

```
par_profile_id integer = NULL::integer,
par_profile_name varchar = NULL::character varying,
par_description varchar = NULL::character varying,
out returncode integer
```

확장 팩의 `aws_sqlserver_ext.sysmail_delete_profile_sp` 프로시저는 `msdb.dbo.sysmail_delete_profile_sp` 프로시저를 에뮬레이션합니다. 소스 SQL Server Database Mail 프로시저에 대한 자세한 내용은 Microsoft 기술 문서를 참조하세요.

```
par_profile_id integer = NULL::integer,
par_profile_name varchar = NULL::character varying,
par_force_delete smallint = 1,
out returncode integer
```

확장 팩의 `aws_sqlserver_ext.sysmail_help_account_sp` 프로시저는 `msdb.dbo.sysmail_help_account_sp` 프로시저를 에뮬레이션합니다. 소스 SQL Server Database Mail 프로시저에 대한 자세한 내용은 Microsoft 기술 문서를 참조하세요.

```
par_account_id integer = NULL::integer,
par_account_name varchar = NULL::character varying,
```

```
out returncode integer
```

확장 팩의 aws_sqlserver_ext.sysmail_update_account_sp 프로시저는 msdb.dbo.sysmail_update_account_sp 프로시저를 에뮬레이션합니다. 소스 SQL Server Database Mail 프로시저에 대한 자세한 내용은 Microsoft 기술 문서를 참조하세요.

```
par_account_id integer = NULL::integer,
par_account_name varchar = NULL::character varying,
par_email_address varchar = NULL::character varying,
par_display_name varchar = NULL::character varying,
par_replyto_address varchar = NULL::character varying,
par_description varchar = NULL::character varying,
par_mailserver_name varchar = NULL::character varying,
par_mailserver_type varchar = NULL::character varying,
par_port integer = NULL::integer,
par_username varchar = NULL::character varying,
par_password varchar = NULL::character varying,
par_use_default_credentials smallint = NULL::smallint,
par_enable_ssl smallint = NULL::smallint,
par_timeout integer = NULL::integer,
par_no_credential_change smallint = NULL::smallint,
out returncode integer
```

확장 팩의 aws_sqlserver_ext.sysmail_delete_account_sp 프로시저는 msdb.dbo.sysmail_delete_account_sp 프로시저를 에뮬레이션합니다. 소스 SQL Server Database Mail 프로시저에 대한 자세한 내용은 Microsoft 기술 문서를 참조하세요.

```
par_account_id integer = NULL::integer,
par_account_name varchar = NULL::character varying,
out returncode integer
```

확장 팩의 aws_sqlserver_ext.sysmail_help_profileaccount_sp 프로시저는 msdb.dbo.sysmail_help_profileaccount_sp 프로시저를 에뮬레이션합니다. 소스 SQL Server Database Mail 프로시저에 대한 자세한 내용은 Microsoft 기술 문서를 참조하세요.

```
par_profile_id integer = NULL::integer,
par_profile_name varchar = NULL::character varying,
par_account_id integer = NULL::integer,
par_account_name varchar = NULL::character varying,
out returncode integer
```

확장 팩의 aws_sqlserver_ext.sysmail_update_profileaccount_sp 프로시저는 msdb.dbo.sysmail_update_profileaccount_sp 프로시저를 에뮬레이션합니다. 소스 SQL Server Database Mail 프로시저에 대한 자세한 내용은 Microsoft 기술 문서를 참조하세요.

```
par_profile_id integer = NULL::integer,
par_profile_name varchar = NULL::character varying,
par_account_id integer = NULL::integer,
par_account_name varchar = NULL::character varying,
par_sequence_number integer = NULL::integer,
out returncode integer
```

확장 팩의 aws_sqlserver_ext.sysmail_delete_profileaccount_sp 프로시저는 msdb.dbo.sysmail_delete_profileaccount_sp 프로시저를 에뮬레이션합니다. 소스 SQL Server Database Mail 프로시저에 대한 자세한 내용은 Microsoft 기술 문서를 참조하세요.

```
par_profile_id integer = NULL::integer,
par_profile_name varchar = NULL::character varying,
par_account_id integer = NULL::integer,
par_account_name varchar = NULL::character varying,
out returncode integer
```

PostgreSQL에서 SQL Server Database Mail을 에뮬레이션하는 프로시저 사용 예제

이메일을 보내려면 다음과 같이 aws_sqlserver_ext.sp_send_dbmail 프로시저를 사용합니다.

```
PERFORM sp_send_dbmail (
    par_profile_name := 'Administrator',
    par_recipients := 'hello@rusgl.info',
    par_subject := 'Automated Success Message',
    par_body := 'The stored procedure finished'
);
```

다음 예제는 쿼리 결과가 포함된 이메일을 전송하는 방법을 보여줍니다.

```
PERFORM sp_send_dbmail (
    par_profile_name := 'Administrator',
    par_recipients := 'hello@rusgl.info',
    par_subject := 'Account with id = 1',
    par_query := 'SELECT COUNT(*)FROM Account WHERE id = 1'
);
```

다음 예제는 HTML 코드가 포함된 이메일을 전송하는 방법을 보여줍니다.

```
DECLARE var_tableHTML TEXT;
SET var_tableHTML := CONCAT(
    '<H1>Work Order Report</H1>',
    '<table border="1">',
    '<tr><th>Work Order ID</th><th>Product ID</th>',
    '<th>Name</th><th>Order Qty</th><th>Due Date</th>',
    '<th>Expected Revenue</th></tr>',
    '</table>'
);
PERFORM sp_send_dbmail (
    par_recipients := 'hello@rusgl.info',
    par_subject := 'Work Order List',
    par_body := var_tableHTML,
    par_body_format := 'HTML'
);
```

이메일을 삭제하려면 다음과 같이 `aws_sqlserver_ext.sysmail_delete_mailitems_sp` 프로시저를 사용합니다.

```
DECLARE var_GETDATE datetime;
SET var_GETDATE = NOW();
PERFORM sysmail_delete_mailitems_sp (
    par_sent_before := var_GETDATE
);
```

다음 예제는 가장 오래된 이메일을 삭제하는 방법을 보여줍니다.

```
PERFORM sysmail_delete_mailitems_sp (
    par_sent_before := '31.12.2015'
);
```

다음 예제는 전송할 수 없는 모든 이메일을 삭제하는 방법을 보여줍니다.

```
PERFORM sysmail_delete_mailitems_sp (
    par_sent_status := 'failed'
);
```

새 사용자 프로필을 생성하려면 다음과 같이 `aws_sqlserver_ext.sysmail_add_profile_sp` 프로시저를 사용합니다.

```
PERFORM sysmail_add_profile_sp (
    profile_name := 'Administrator',
    par_description := 'administrative mail'
);
```

다음 예제는 새 프로필을 생성하고 고유한 프로필 식별자를 변수로 저장하는 방법을 보여줍니다.

```
DECLARE var_profileId INT;
SELECT par_profile_id
    FROM sysmail_add_profile_sp (
        profile_name := 'Administrator',
        par_description := ' Profile used for administrative mail.')
    INTO var_profileId;

SELECT var_profileId;
```

새 이메일 계정을 생성하려면 다음과 같이 `aws_sqlserver_ext.sysmail_add_account_sp` 프로시저를 사용합니다.

```
PERFORM sysmail_add_account_sp (
    par_account_name :='Audit Account',
    par_email_address := 'dba@rusgl.info',
    par_display_name := 'Test Automated Mailer',
    par_description := 'Account for administrative e-mail.',
    par_mailserver_type := 'AWSLAMBDA'
    par_mailserver_name := 'arn:aws:lambda:us-west-2:555555555555:function:pg_v3'
);
```

사용자 프로필에 이메일 계정을 추가하려면 다음과 같이 `aws_sqlserver_ext.sysmail_add_profileaccount_sp` 프로시저를 사용합니다.

```
PERFORM sysmail_add_profileaccount_sp (
    par_account_name := 'Administrator',
    par_account_name := 'Audit Account',
    par_sequence_number := 1
);
```

PostgreSQL에서 SQL Server Database Mail을 에뮬레이션하기 위한 사용 사례 예제

소스 데이터베이스 코드에서 SQL Server Database Mail을 사용하여 이메일을 전송하는 경우 AWS SCT 확장 팩을 사용하여 이 코드를 PostgreSQL로 변환할 수 있습니다.

PostgreSQL 데이터베이스에서 이메일을 보내려면

1. AWS Lambda 함수를 생성 및 구성합니다.
2. AWS SCT 확장 팩을 적용합니다.
3. 다음과 같이 sysmail_add_profile_sp 함수를 사용하여 사용자 프로필을 생성합니다.
4. 다음과 같이 sysmail_add_account_sp 함수를 사용하여 이메일 계정을 생성합니다.
5. 다음과 같이 sysmail_add_profileaccount_sp 함수를 사용하여 이 이메일 계정을 사용자 프로필에 추가합니다.

```
CREATE OR REPLACE FUNCTION aws_sqlserver_ext.
proc_dbmail_settings_msdb()
RETURNS void
AS
$BODY$
BEGIN
PERFORM aws_sqlserver_ext.sysmail_add_profile_sp(
    par_profile_name := 'Administrator',
    par_description := 'administrative mail'
);
PERFORM aws_sqlserver_ext.sysmail_add_account_sp(
    par_account_name := 'Audit Account',
    par_description := 'Account for administrative e-mail.',
    par_email_address := 'dba@rusgl.info',
    par_display_name := 'Test Automated Mailer',
    par_mailserver_type := 'AWSLAMBDA'
    par_mailserver_name := 'your_ARN'
);
PERFORM aws_sqlserver_ext.sysmail_add_profileaccount_sp(
    par_profile_name := 'Administrator',
    par_account_name := 'Audit Account',
    par_sequence_number := 1
);
END;
$BODY$
LANGUAGE plpgsql;
```

6. 다음과 같이 sp_send_dbmail 함수를 사용하여 이메일을 보냅니다.

```
CREATE OR REPLACE FUNCTION aws_sqlserver_ext.
proc_dbmail_send_msdb()
RETURNS void
```

```
AS
$BODY$
BEGIN
PERFORM aws_sqlserver_ext.sp_send_dbmail(
    par_profile_name := 'Administrator',
    par_recipients := 'hello@rusgl.info',
    par_body := 'The stored procedure finished',
    par_subject := 'Automated Success Message'
);
END;
$BODY$
LANGUAGE plpgsql;
```

모든 사용자 프로필에 대한 정보를 보려면 다음과 같이 sysmail_help_profile_sp 프로시저를 사용합니다.

```
SELECT FROM aws_sqlserver_ext.sysmail_help_profile_sp();
```

다음 예제는 특정 사용자 프로필에 대한 정보를 표시합니다.

```
select from aws_sqlserver_ext.sysmail_help_profile_sp(par_profile_id := 1);
select from aws_sqlserver_ext.sysmail_help_profile_sp(par_profile_name :=
 'Administrator');
```

모든 이메일 계정에 대한 정보를 보려면 다음과 같이 sysmail_help_account_sp 프로시저를 사용합니다.

```
select from aws_sqlserver_ext.sysmail_help_account_sp();
```

다음 예제는 특정 이메일 계정에 대한 정보를 표시합니다.

```
select from aws_sqlserver_ext.sysmail_help_account_sp(par_account_id := 1);
select from aws_sqlserver_ext.sysmail_help_account_sp(par_account_name := 'Audit
 Account');
```

사용자 프로필과 연결된 모든 이메일 계정에 대한 정보를 보려면 다음과 같이 sysmail_help_profileaccount_sp 프로시저를 사용합니다.

```
select from aws_sqlserver_ext.sysmail_help_profileaccount_sp();
```

다음 예제는 식별자, 프로필 이름 또는 계정 이름을 기준으로 레코드를 필터링합니다.

```
select from aws_sqlserver_ext.sysmail_help_profileaccount_sp(par_profile_id := 1);
select from aws_sqlserver_ext.sysmail_help_profileaccount_sp(par_profile_id := 1,
 par_account_id := 1);
select from aws_sqlserver_ext.sysmail_help_profileaccount_sp(par_profile_name :=
 'Administrator');
select from aws_sqlserver_ext.sysmail_help_profileaccount_sp(par_account_name := 'Audit
 Account');
```

사용자 프로필 이름 또는 설명을 변경하려면 다음과 같이 sysmail_update_profile_sp 프로시저를 사용합니다.

```
select aws_sqlserver_ext.sysmail_update_profile_sp(
    par_profile_id := 2,
    par_profile_name := 'New profile name'
);
```

이메일 계정 설정을 변경하려면 다음과 같이 ysmail_update_account_sp 프로시저를 사용합니다.

```
select from aws_sqlserver_ext.sysmail_update_account_sp (
    par_account_name := 'Audit Account',
    par_mailserver_name := 'arn:aws:lambda:region:XXXXXXXXXXXX:function:func_test',
    par_mailserver_type := 'AWSLAMBDA'
);
```

SQL Server를 Amazon RDS for SQL Server로 변환

SQL Server 스키마와 코드를 Amazon RDS for SQL Server로 마이그레이션할 경우 몇 가지 사항을 고려해야 합니다.

- AWS SCT는 SQL Server 에이전트를 변환하여 Amazon RDS for SQL Server 인스턴스에서 예약, 알림 및 작업을 제공할 수 있습니다. 변환 후 Amazon RDS for SQL Server DB 인스턴스를 SSRS(SQL Server Reporting Service), SSAS(SQL Server Analysis Services), SSIS(SQL Server Integration Services)와 함께 사용할 수 있습니다.
- 현재 Amazon RDS는 SQL Server Service Broker 또는 CREATE ENDPOINT 명령을 실행하는 데 필요한 추가 T-SQL 엔드포인트를 지원하지 않습니다.
- Amazon RDS는 연결된 서버를 제한적으로 지원합니다. 연결된 서버를 사용하는 SQL Server 애플리케이션 코드를 변환할 때 AWS SCT에서는 애플리케이션 코드를 변환하지만, 사용자는 이 변환된 코드를 실행하기 전에 먼저 연결 서버를 사용하는 객체의 동작을 검토해야 합니다.

- 상시 가동 기능이 사용됩니다.
- AWS SCT 평가 보고서에서 변환을 위한 서버 측정치를 제공합니다. SQL Server 인스턴스에 대한 이러한 측정치에는 다음이 포함됩니다.
  - 데이터 미러링이 사용됩니다.
  - SQL Server 로그 전달이 구성되었습니다.
  - 장애 조치 클러스터가 사용됩니다.
  - Database Mail이 구성되었습니다.
  - 전체 텍스트 검색 서비스가 사용됩니다. Amazon RDS for SQL Server는 전체 텍스트 검색이 제한적이며 의미 체계 검색을 지원하지 않습니다.
  - 데이터 품질 서비스(DQS)가 설치되었습니다. Amazon RDS는 DQS를 지원하지 않으므로 Amazon EC2 인스턴스에 SQL Server를 설치하는 것이 좋습니다.

RDS for SQL Server를 대상으로 사용하기 위한 권한

RDS for SQL Server로 마이그레이션하려면 데이터베이스 사용자를 생성하고 각 데이터베이스에 필요한 권한을 부여합니다. 다음과 같은 코드 예제를 사용할 수 있습니다.

```
CREATE LOGIN user_name WITH PASSWORD 'your_password';

USE db_name
CREATE USER user_name FOR LOGIN user_name
GRANT VIEW DEFINITION TO user_name
GRANT VIEW DATABASE STATE TO user_name
GRANT CREATE SCHEMA TO user_name;
GRANT CREATE TABLE TO user_name;
GRANT CREATE VIEW TO user_name;
GRANT CREATE TYPE TO user_name;
GRANT CREATE DEFAULT TO user_name;
GRANT CREATE FUNCTION TO user_name;
GRANT CREATE PROCEDURE TO user_name;
GRANT CREATE ASSEMBLY TO user_name;
GRANT CREATE AGGREGATE TO user_name;
GRANT CREATE FULLTEXT CATALOG TO user_name;
GRANT CREATE SYNONYM TO user_name;
GRANT CREATE XML SCHEMA COLLECTION TO user_name;
```

이전 예제에서 *user_name*을 사용자 이름으로 바꿉니다. 그런 다음 *db_name*을 대상 데이터베이스의 이름으로 바꿉니다. 마지막으로 *your_password*를 안전한 암호로 바꿉니다.

AWS Schema Conversion Tool용 데이터 웨어하우스 소스

AWS SCT는 다음 소스 데이터 웨어하우스의 스키마를 지원되는 대상으로 변환할 수 있습니다. 권한, 연결 및 AWS SCT에서 대상 데이터베이스 또는 데이터 웨어하우스 용도로 변환할 수 있는 항목에 대한 자세한 내용은 아래에서 세부 정보를 참조하세요.

주제

- Amazon Redshift를 AWS SCT에 대한 소스로 사용
- Azure Synapse Analytics를 AWS SCT에 대한 소스로 사용
- BigQuery를 AWS SCT에 대한 소스로 사용
- Greenplum Database를 AWS SCT에 대한 소스로 사용
- Netezza를 AWS SCT에 대한 소스로 사용
- Oracle Data Warehouse를 AWS SCT에 대한 소스로 사용
- Snowflake를 AWS SCT에 대한 소스로 사용
- Microsoft SQL Server Data Warehouse를 AWS SCT에 대한 소스로 사용
- Teradata를 AWS SCT에 대한 소스로 사용
- Vertica를 AWS SCT에 대한 소스로 사용

Amazon Redshift를 AWS SCT에 대한 소스로 사용

AWS SCT를 사용하여 Amazon Redshift 클러스터를 최적화할 수 있습니다. AWS SCT는 Amazon Redshift 클러스터의 배포 및 정렬 키 선택에 대한 권장 사항을 제공합니다. Amazon Redshift 최적화 프로젝트는 소스와 대상이 서로 다른 Amazon Redshift 클러스터를 가리키는 AWS SCT 프로젝트로 간주할 수 있습니다.

Amazon Redshift를 소스 데이터베이스로 사용하기 위한 권한

Amazon Redshift를 소스로 사용하기 위해 필요한 권한은 다음과 같습니다.

- USAGE ON SCHEMA *<schema_name>*
- SELECT ON ALL TABLES IN SCHEMA *<schema_name>*
- SELECT ON PG_CATALOG.PG_STATISTIC
- SELECT ON SVV_TABLE_INFO
- SELECT ON TABLE STV_BLOCKLIST
- SELECT ON TABLE STV_TBL_PERM

- SELECT ON SYS_SERVERLESS_USAGE
- SELECT ON PG_DATABASE_INFO
- SELECT ON PG_STATISTIC

이전 예제에서 *<schema_name>* 자리 표시자를 소스 스키마의 이름으로 바꿉니다.

Amazon Redshift를 대상으로 사용하기 위해 필요한 권한에 대한 자세한 내용은 Amazon Redshift를 대상으로 사용할 수 있는 권한 섹션을 참조하세요.

Amazon Redshift에 소스로 연결

다음 절차에 따라 AWS Schema Conversion Tool을 사용하여 Amazon Redshift 소스 데이터베이스에 연결합니다.

Amazon Redshift 소스 데이터베이스에 연결하려면

1. AWS Schema Conversion Tool에서 소스 추가를 선택합니다.
2. Amazon Redshift를 선택한 후 다음을 선택합니다.

   소스 추가 대화 상자가 나타납니다.
3. 연결 이름에 데이터베이스의 이름을 입력합니다. AWS SCT는 왼쪽 패널의 트리에 이 이름을 표시합니다.
4. AWS Secrets Manager의 데이터베이스 보안 인증 정보를 사용하거나 수동으로 입력합니다.

   - Secrets Manager의 데이터베이스 보안 인증 정보를 사용하려면 다음 지침을 따릅니다.
     1. AWS Secret에서 보안 암호의 이름을 선택합니다.
     2. Populate를 선택하여 Secrets Manager에서 데이터베이스 연결 대화 상자에 있는 모든 값을 자동으로 채웁니다.

     Secrets Manager의 데이터베이스 보안 인증 사용에 대한 자세한 내용은 AWS Secrets Manager 사용 섹션을 참조하세요.
   - Amazon Redshift 소스 데이터베이스의 연결 정보를 입력하려면 다음 지침을 사용합니다.

파라미터	작업
[Server name]	소스 데이터베이스 서버의 도메인 이름 시스템(DNS) 이름 또는 IP 주소를 입력합니다.

파라미터	작업
[Server port]	소스 데이터베이스 서버에 연결하는 데 사용되는 포트를 입력합니다.
데이터베이스	Amazon Redshift 데이터베이스의 이름을 입력합니다.
User name 및 Password	소스 데이터베이스 서버에 연결하는 데 필요한 데이터베이스 보안 인증 정보를 입력합니다. 프로젝트에서 데이터베이스에 연결하도록 선택한 경우에만 AWS SCT에서 암호를 사용하여 소스 데이터베이스에 연결합니다. 소스 데이터베이스의 암호가 노출될 위험을 방지하기 위해 AWS SCT는 기본적으로 암호를 저장하지 않습니다. AWS SCT 프로젝트를 닫았다 다시 열 경우 필요에 따라 소스 데이터베이스에 연결하기 위한 암호를 입력하라는 메시지가 표시됩니다.
Use SSL(SSL 사용)	SSL(Secure Sockets Layer)을 사용하여 데이터베이스에 연결하려면 이 옵션을 선택합니다. SSL 탭에서 다음 추가 정보를 적절히 제공합니다. • Verify server certificate: 트러스트 스토어를 사용하여 서버 인증서를 확인하려면 이 옵션을 선택합니다. • 트러스트 스토어: 인증서가 있는 트러스트 스토어의 위치입니다. 이 위치가 여기에 표시되도록 하려면 전역 설정에 추가해야 합니다. Amazon Redshift의 SSL 지원에 대한 자세한 내용은 연결을 위한 보안 옵션 구성을 참조하세요.
Store password	AWS SCT는 SSL 인증서와 데이터베이스 암호를 저장할 안전한 볼트를 생성합니다. 이 옵션을 켜면 데이터베이스 암호를 저장하고 암호 입력 없이 빠르게 데이터베이스에 연결할 수 있습니다.

파라미터	작업
Redshift driver path	소스 데이터베이스에 연결할 때 사용할 드라이버의 경로를 입력합니다. 자세한 내용은 필수 데이터베이스 드라이버 다운로드 섹션을 참조하세요. 드라이버 경로를 전역 프로젝트 설정에 저장할 경우 드라이버 경로가 연결 대화 상자에 표시되지 않습니다. 자세한 내용은 전역 설정에 드라이버 경로 저장 섹션을 참조하세요.

5. Test Connection을 선택하여 AWS SCT가 소스 데이터베이스에 연결할 수 있는지 확인합니다.
6. 연결을 선택하여 소스 데이터베이스에 연결합니다.

Amazon Redshift 최적화 설정

Amazon Redshift 최적화 설정을 편집하려면 AWS SCT에서 설정을 선택한 다음 변환 설정을 선택합니다. 상단 목록에서 Amazon Redshift를 선택한 다음 Amazon Redshift – Amazon Redshift를 선택합니다. AWS SCT는 Amazon Redshift 최적화에 사용할 수 있는 모든 설정을 표시합니다.

AWS SCT의 Amazon Redshift 최적화 설정에는 다음과 같은 옵션이 포함됩니다.

- 변환된 코드에서 작업 항목이 포함된 설명의 수를 제한합니다.

  Add comments in the converted code for the action items of selected severity and higher에서 작업 항목의 심각도를 선택합니다. AWS SCT가 선택된 심각도 이상의 작업 항목에 대한 변환된 코드에 설명을 추가합니다.

  예를 들어, 변환된 코드의 설명 수를 최소화하려면 오류만을 선택합니다. 변환된 코드의 모든 작업 항목에 대한 설명을 포함하려면 모든 메시지를 선택합니다.
- AWS SCT가 대상 Amazon Redshift 클러스터에 적용할 수 있는 최대 테이블 수를 설정합니다.

  The maximum number of tables for the target Amazon Redshift cluster에서 AWS SCT가 Amazon Redshift 클러스터에 적용할 수 있는 테이블 수를 선택합니다.

  Amazon Redshift에는 여러 클러스터 노드 유형에 사용하는 테이블을 제한하는 할당량이 있습니다. 자동을 선택하면 AWS SCT가 노드 유형에 따라 대상 Amazon Redshift 클러스터에 적용되는 테이블 수를 결정합니다. 값을 수동으로 선택할 수도 있습니다. 자세한 내용은 Amazon Redshift 관리 가이드의 Amazon Redshift의 할당량 및 제한 섹션을 참조하세요.

Amazon Redshift 클러스터가 저장할 수 있는 수보다 테이블 수가 많아도 AWS SCT는 모든 소스 테이블을 변환합니다. AWS SCT는 변환된 코드를 프로젝트에 저장하며 대상 데이터베이스에는 적용하지 않습니다. 변환된 코드를 적용할 때 테이블의 Amazon Redshift 클러스터 할당량에 도달하면 AWS SCT에서 경고 메시지가 표시됩니다. 또한 AWS SCT는 테이블 수가 한도에 도달할 때까지 대상 Amazon Redshift 클러스터에 테이블을 적용합니다.

- 마이그레이션 전략을 선택합니다.

  AWS에서는 최적화 프로젝트의 소스 및 대상으로 여러 클러스터를 사용하도록 권장합니다. Amazon Redshift 최적화 프로세스를 시작하기 전에 소스 Amazon Redshift 클러스터의 사본을 생성합니다. 이 사본에 소스 데이터를 포함하거나 빈 클러스터를 생성할 수 있습니다.

  소스 클러스터의 데이터를 대상 클러스터에 포함하려면 Migration strategy에서 Migration to a copy를 선택합니다.

  최적화 제안을 검토하려면 Migration strategy에서 Migration to a clean slate를 선택합니다. 이러한 제안을 수락한 후에는 소스 데이터를 대상 클러스터로 마이그레이션합니다.

- Amazon Redshift 테이블 열에 압축을 적용합니다. 이렇게 하려면 Use compression encoding을 선택합니다.

  AWS SCT는 기본 Amazon Redshift 알고리즘을 사용하여 열에 압축 인코딩을 자동으로 할당합니다. 자세한 내용은 Amazon Redshift 데이터베이스 개발자 안내서의 압축 인코딩을 참조하세요.

  기본적으로 Amazon Redshift는 정렬 및 배포 키로 정의된 열에 압축을 적용하지 않습니다. 이 동작을 변경하여 이러한 열에 압축을 적용할 수 있습니다. 이렇게 하려면 Use compression encoding for KEY columns를 선택합니다. Use compression encoding 옵션을 선택한 경우에만 이 옵션을 선택할 수 있습니다.

- 자동 테이블 최적화 작업을 수행합니다.

  자동 테이블 최적화는 테이블 디자인을 자동으로 최적화하는 Amazon Redshift의 자체 조정 프로세스입니다. 자세한 내용은 Amazon Redshift 데이터베이스 개발자 안내서의 자동 테이블 최적화 작업을 참조하세요.

  자동 테이블 최적화에만 사용하려면 왼쪽 창에서 Optimization strategies를 선택합니다. 그런 다음 Use Amazon Redshift automatic table tuning을 선택하고 Initial key selection strategy에서 없음을 선택합니다.

- 전략을 사용하여 정렬 및 배포 키를 선택합니다.

Amazon Redshift 메타데이터, 통계 정보 또는 두 옵션을 모두 사용하여 정렬 및 배포 키를 선택할 수 있습니다. Optimization strategies 탭의 Initial key selection strategy에서 다음 옵션 중 하나를 선택합니다.

- 메타데이터 사용, 통계 정보 무시
- 메타데이터 무시, 통계 정보 사용
- 메타데이터 및 통계 정보 사용

선택한 옵션에 따라 최적화 전략을 선택할 수 있습니다. 그런 다음 각 전략에 대해 값(0~100)을 입력합니다. 이러한 값은 각 전략의 가중치를 정의합니다. AWS SCT는 이러한 가중치 값을 사용하여 각 규칙이 배포 및 정렬 키 선택에 미치는 영향을 정의합니다. 기본값은 AWS 마이그레이션 모범 사례를 기반으로 합니다.

Find small tables 전략에서 작은 테이블의 크기를 정의할 수 있습니다. Min table row count 및 Max table row count에 테이블의 최소 및 최대 행 수를 입력하여 작은 테이블로 정의합니다. AWS SCT는 작은 테이블에 ALL 배포 스타일을 적용합니다. 이 경우 전체 테이블의 사본이 모든 노드에 배포됩니다.

- 전략 세부 정보를 구성합니다.

  각 최적화 전략의 가중치를 정의하는 것 외에 최적화 설정도 구성할 수 있습니다. 이 작업을 수행하려면 Conversion optimization을 선택합니다.

  - Sort key columns limit에 정렬 키의 최대 열 수를 입력합니다.
  - Skewed threshold value에 열에 대한 스큐된 값의 백분율(0~100)을 입력합니다. AWS SCT는 스큐 값이 임계값보다 큰 열을 배포 키의 후보 목록에서 제외합니다. AWS SCT는 열의 스큐된 값을 전체 레코드 수에 대한 가장 일반적인 값 발생 횟수의 백분율로 정의합니다.
  - Top N queries from the query history table에 분석할 가장 자주 사용되는 쿼리의 수(1~100)를 입력합니다.
  - Select statistics user에서 쿼리 통계를 분석하려는 데이터베이스 사용자를 선택합니다.

Azure Synapse Analytics를 AWS SCT에 대한 소스로 사용

AWS SCT를 사용하여 스키마, 코드 객체 및 애플리케이션 코드를 Azure Synapse Analytics에서 Amazon Redshift로 변환할 수 있습니다.

Azure Synapse Analytics를 소스 데이터베이스로 사용하기 위한 권한

Azure Synapse Analytics 데이터 웨어하우스를 소스로 사용하려면 다음과 같은 권한이 필요합니다.

- VIEW DEFINITION
- VIEW DATABASE STATE

스키마를 변환하려는 각 데이터베이스에 권한을 적용합니다.

Azure Synapse Analytics에 소스로 연결

다음 절차에 따라 AWS Schema Conversion Tool을 사용하여 Azure Synapse Analytics 데이터 웨어하우스에 연결합니다.

Azure Synapse Analytics 데이터 웨어하우스에 소스로 연결하려면

1. AWS Schema Conversion Tool에서 소스 추가를 선택합니다.
2. Azure Synapse Analytics를 선택한 후 다음을 선택합니다.

   소스 추가 대화 상자가 나타납니다.
3. 연결 이름에 데이터베이스의 이름을 입력합니다. AWS SCT는 왼쪽 패널의 트리에 이 이름을 표시합니다.
4. AWS Secrets Manager의 데이터베이스 보안 인증 정보를 사용하거나 수동으로 입력합니다.

   - Secrets Manager의 데이터베이스 보안 인증 정보를 사용하려면 다음 지침을 따릅니다.

     1. AWS Secret에서 보안 암호의 이름을 선택합니다.
     2. Populate를 선택하여 Secrets Manager에서 데이터베이스 연결 대화 상자에 있는 모든 값을 자동으로 채웁니다.

     Secrets Manager의 데이터베이스 보안 인증 사용에 대한 자세한 내용은 AWS Secrets Manager 사용 섹션을 참조하세요.
   - Azure Synapse Analytics 데이터 웨어하우스의 연결 정보를 수동으로 입력하려면 다음 지침을 사용합니다.

파라미터	작업
[Server name]	소스 데이터베이스 서버의 DNS(Domain Name Service) 이름 또는 IP 주소를 입력합니다.
SQL pool	Azure SQL 풀의 이름을 입력합니다.

파라미터	작업
User name 및 Password	소스 데이터베이스 서버에 연결하는 데 필요한 데이터베이스 보안 인증 정보를 입력합니다. 프로젝트에서 데이터베이스에 연결하도록 선택한 경우에만 AWS SCT에서 암호를 사용하여 소스 데이터베이스에 연결합니다. 소스 데이터베이스의 암호가 노출될 위험을 방지하기 위해 AWS SCT는 기본적으로 암호를 저장하지 않습니다. AWS SCT 프로젝트를 닫았다 다시 열 경우 필요에 따라 소스 데이터베이스에 연결하기 위한 암호를 입력하라는 메시지가 표시됩니다.
Use SSL(SSL 사용)	SSL(Secure Sockets Layer)을 사용하여 데이터베이스에 연결하려면 이 옵션을 선택합니다. SSL 탭에서 다음 추가 정보를 적절히 제공합니다. • Trust server certificate: 서버 인증서를 신뢰하려면 이 옵션을 선택합니다. • 트러스트 스토어: 전역 설정에서 설정한 트러스트 스토어입니다.
Store password	AWS SCT는 SSL 인증서와 데이터베이스 암호를 저장할 안전한 볼트를 생성합니다. 이 옵션을 켜면 데이터베이스 암호를 저장한 후 암호를 입력하지 않고 빠르게 데이터베이스에 연결할 수 있습니다.

5. Test Connection을 선택하여 AWS SCT가 소스 데이터베이스에 연결할 수 있는지 확인합니다.
6. 연결을 선택하여 소스 데이터베이스에 연결합니다.

Azure Synapse Analytics에서 Amazon Redshift로의 변환 설정

Azure Synapse Analytics에서 Amazon Redshift로의 변환 설정을 편집하려면 AWS SCT에서 설정을 선택한 다음 변환 설정을 선택합니다. 상단 목록에서 Azure Synapse를 선택한 다음 Azure Synapse – Amazon Redshift를 선택합니다. AWS SCT는 Azure Synapse Analytics에서 Amazon Redshift로의 변환에 사용할 수 있는 모든 설정을 표시합니다.

AWS SCT에서 Azure Synapse Analytics를 Amazon Redshift로 변환하는 설정에는 다음과 같은 옵션이 포함됩니다.

- 변환된 코드에서 작업 항목이 포함된 설명의 수를 제한합니다.

  Add comments in the converted code for the action items of selected severity and higher에서 작업 항목의 심각도를 선택합니다. AWS SCT가 선택된 심각도 이상의 작업 항목에 대한 변환된 코드에 설명을 추가합니다.

  예를 들어, 변환된 코드의 설명 수를 최소화하려면 오류만을 선택합니다. 변환된 코드의 모든 작업 항목에 대한 설명을 포함하려면 모든 메시지를 선택합니다.

- AWS SCT가 대상 Amazon Redshift 클러스터에 적용할 수 있는 최대 테이블 수를 설정합니다.

  The maximum number of tables for the target Amazon Redshift cluster에서 AWS SCT가 Amazon Redshift 클러스터에 적용할 수 있는 테이블 수를 선택합니다.

  Amazon Redshift에는 여러 클러스터 노드 유형에 사용하는 테이블을 제한하는 할당량이 있습니다. 자동을 선택하면 AWS SCT가 노드 유형에 따라 대상 Amazon Redshift 클러스터에 적용되는 테이블 수를 결정합니다. 값을 수동으로 선택할 수도 있습니다. 자세한 내용은 Amazon Redshift 관리 가이드의 Amazon Redshift의 할당량 및 제한 섹션을 참조하세요.

  Amazon Redshift 클러스터가 저장할 수 있는 수보다 이 숫자가 많아도 AWS SCT는 모든 소스 테이블을 변환합니다. AWS SCT는 변환된 코드를 프로젝트에 저장하며 대상 데이터베이스에는 적용하지 않습니다. 변환된 코드를 적용할 때 테이블의 Amazon Redshift 클러스터 할당량에 도달하면 AWS SCT에서 경고 메시지가 표시됩니다. 또한 AWS SCT는 테이블 수가 한도에 도달할 때까지 대상 Amazon Redshift 클러스터에 테이블을 적용합니다.

- 소스 테이블의 파티션을 Amazon Redshift의 개별 테이블로 마이그레이션합니다. 이 작업을 수행하려면 Use the UNION ALL view를 선택하고 AWS SCT가 단일 소스 테이블에 대해 생성할 수 있는 대상 테이블의 최대 수를 입력합니다.

  Amazon Redshift는 테이블 파티셔닝을 지원하지 않습니다. 이 동작을 에뮬레이션하고 쿼리를 더 빠르게 실행하기 위해 AWS SCT는 소스 테이블의 각 파티션을 Amazon Redshift의 개별 테이블로 마이그레이션할 수 있습니다. 그런 다음 AWS SCT는 이러한 모든 테이블의 데이터를 포함하는 보기를 생성합니다.

  AWS SCT는 소스 테이블의 파티션 수를 자동으로 결정합니다. 소스 테이블 파티셔닝 유형에 따라서는 이 숫자가 Amazon Redshift 클러스터에 적용할 수 있는 테이블의 할당량을 초과할 수 있습니다. 이 할당량에 도달하지 않도록 하려면 AWS SCT가 단일 소스 테이블의 파티션에 대해 생성할 수 있는 최대 대상 테이블 수를 입력합니다. 기본 옵션은 368개 테이블이며, 이는 1년 중 366일 동안의 파티션과 `NO  RANGE` 및 `UNKNOWN` 파티션에 대한 테이블 두 개를 나타냅니다.

- Amazon Redshift 테이블 열에 압축을 적용합니다. 이렇게 하려면 Use compression encoding을 선택합니다.

  AWS SCT는 기본 Amazon Redshift 알고리즘을 사용하여 열에 압축 인코딩을 자동으로 할당합니다. 자세한 내용은 Amazon Redshift 데이터베이스 개발자 안내서의 압축 인코딩을 참조하세요.

  기본적으로 Amazon Redshift는 정렬 및 배포 키로 정의된 열에 압축을 적용하지 않습니다. 이 동작을 변경하여 이러한 열에 압축을 적용할 수 있습니다. 이렇게 하려면 Use compression encoding for KEY columns를 선택합니다. Use compression encoding 옵션을 선택한 경우에만 이 옵션을 선택할 수 있습니다.

Azure Synapse Analytics에서 Amazon Redshift로의 변환 최적화 설정

Azure Synapse Analytics에서 Amazon Redshift로의 변환 최적화 설정을 편집하려면 AWS SCT에서 설정을 선택한 다음 변환 설정을 선택합니다. 상단 목록에서 Azure Synapse를 선택한 다음 Azure Synapse – Amazon Redshift를 선택합니다. 왼쪽 창에서 Optimization strategies를 선택합니다. AWS SCT는 Azure Synapse Analytics에서 Amazon Redshift로의 변환에 대한 변환 최적화 설정을 표시합니다.

AWS SCT에서 Azure Synapse Analytics에서 Amazon Redshift로의 변환 최적화 설정에는 다음과 같은 옵션이 포함됩니다.

- 자동 테이블 최적화 작업을 수행합니다. 이 작업을 수행하려면 Use Amazon Redshift automatic table tuning을 선택합니다.

  자동 테이블 최적화는 테이블 디자인을 자동으로 최적화하는 Amazon Redshift의 자체 조정 프로세스입니다. 자세한 내용은 Amazon Redshift 데이터베이스 개발자 안내서의 자동 테이블 최적화 작업을 참조하세요.

  자동 테이블 최적화만 사용하려면 Initial key selection strategy에서 없음을 선택합니다.
- 전략을 사용하여 정렬 및 배포 키를 선택합니다.

  Amazon Redshift 메타데이터, 통계 정보 또는 두 옵션을 모두 사용하여 정렬 및 배포 키를 선택할 수 있습니다. Optimization strategies 탭의 Initial key selection strategy에서 다음 옵션 중 하나를 선택합니다.
  - 메타데이터 사용, 통계 정보 무시
  - 메타데이터 무시, 통계 정보 사용
  - 메타데이터 및 통계 정보 사용

선택한 옵션에 따라 최적화 전략을 선택할 수 있습니다. 그런 다음 각 전략에 대해 값(0~100)을 입력합니다. 이러한 값은 각 전략의 가중치를 정의합니다. AWS SCT는 이러한 가중치 값을 사용하여 각 규칙이 배포 및 정렬 키 선택에 미치는 영향을 정의합니다. 기본값은 AWS 마이그레이션 모범 사례를 기반으로 합니다.

Find small tables 전략에서 작은 테이블의 크기를 정의할 수 있습니다. Min table row count 및 Max table row count에 테이블의 최소 및 최대 행 수를 입력하여 작은 테이블로 정의합니다. AWS SCT는 작은 테이블에 ALL 배포 스타일을 적용합니다. 이 경우 전체 테이블의 사본이 모든 노드에 배포됩니다.

- 전략 세부 정보를 구성합니다.

  각 최적화 전략의 가중치를 정의하는 것 외에 최적화 설정도 구성할 수 있습니다. 이 작업을 수행하려면 Conversion optimization을 선택합니다.

  - Sort key columns limit에 정렬 키의 최대 열 수를 입력합니다.
  - Skewed threshold value에 열에 대한 스큐된 값의 백분율(0~100)을 입력합니다. AWS SCT는 스큐 값이 임계값보다 큰 열을 배포 키의 후보 목록에서 제외합니다. AWS SCT는 열의 스큐된 값을 전체 레코드 수에 대한 가장 일반적인 값 발생 횟수의 백분율로 정의합니다.
  - Top N queries from the query history table에 분석할 가장 자주 사용되는 쿼리의 수(1~100)를 입력합니다.
  - Select statistics user에서 쿼리 통계를 분석하려는 데이터베이스 사용자를 선택합니다.

  또한 Optimization strategies 탭에서 Find small tables 전략을 위한 작은 테이블의 크기를 정의할 수 있습니다. Min table row count 및 Max table row count에 테이블의 최소 및 최대 행 수를 입력하여 작은 테이블로 간주합니다. AWS SCT는 작은 테이블에 ALL 배포 스타일을 적용합니다. 이 경우 전체 테이블의 사본이 모든 노드에 배포됩니다.

BigQuery를 AWS SCT에 대한 소스로 사용

AWS SCT를 사용하여 스키마, 코드 객체 및 애플리케이션 코드를 BigQuery에서 Amazon Redshift로 변환할 수 있습니다.

BigQuery를 소스로 사용하기 위한 권한

AWS SCT에서 BigQuery 데이터 웨어하우스를 소스로 사용하려면 서비스 계정을 생성합니다. Google Cloud에서 애플리케이션은 서비스 계정을 사용하여 승인된 API 호출을 수행합니다. 서비스 계정은 사

용자 계정과 다릅니다. 자세한 내용은 Google Cloud Identity and Access Management 문서에서 서비스 계정을 참조하세요.

서비스 계정에 다음 역할을 부여해야 합니다.

- `BigQuery Admin`
- `Storage Admin`

`BigQuery Admin` 역할은 프로젝트 내 모든 리소스를 관리할 수 있는 권한을 제공합니다. AWS SCT는 이 역할을 사용하여 마이그레이션 프로젝트에 BigQuery 메타데이터를 로드합니다.

`Storage Admin` 역할은 데이터 객체 및 버킷에 대한 모든 제어 권한을 부여합니다. `Cloud Storage`에서 이 역할을 찾을 수 있습니다. AWS SCT는 이 역할을 사용하여 BigQuery에서 데이터를 추출한 다음 Amazon Redshift에 로드합니다.

서비스 계정 키 파일을 생성하려면

1. https://console.cloud.google.com/에서 Google Cloud 관리 콘솔에 로그인합니다.
2. BigQuery API 페이지에서 활성화를 선택합니다. API Enabled가 표시된 경우에는 이 단계를 건너뜁니다.
3. 서비스 계정 페이지에서 프로젝트를 선택한 다음 Create service account를 선택합니다.
4. Service account details 페이지에서 서비스 계정 이름을 설명하는 값을 입력합니다. Create and continue를 선택합니다. Grant this service account access to the project 페이지가 열립니다.
5. 역할 선택에서 BigQuery를 선택한 다음 BigQuery Admin을 선택합니다.
6. 다른 역할 추가를 선택합니다. 역할 선택에서 Cloud Storage를 선택한 다음 Storage Admin을 선택합니다.
7. 계속을 선택한 다음 완료를 선택합니다.
8. 서비스 계정 페이지에서 생성된 서비스 계정을 선택합니다.
9. 키를 선택한 다음 키 추가에서 새 키 생성을 선택합니다.
10. JSON을 선택한 다음 생성을 선택합니다. 프라이빗 키를 저장할 폴더를 선택하거나 브라우저에서 다운로드에 사용할 기본 폴더를 선택합니다.

BigQuery 데이터 웨어하우스에서 데이터를 추출하기 위해 AWS SCT는 Google Cloud Storage 버킷 폴더를 사용합니다. 데이터 마이그레이션을 시작하기 전에 이 버킷을 생성합니다. Create Local task 대화 상자에 Google Cloud Storage 버킷 폴더의 경로를 입력합니다. 자세한 내용은 AWS SCT 작업 생성, 실행, 모니터링 섹션을 참조하세요.

BigQuery에 소스로 연결

다음 절차에 따라 AWS Schema Conversion Tool을 사용하여 소스 BigQuery 프로젝트에 연결합니다.

BigQuery 소스 데이터 웨어하우스에 연결하려면

1. AWS Schema Conversion Tool에서 소스 추가를 선택합니다.
2. BigQuery를 선택한 후 다음을 선택합니다.

   소스 추가 대화 상자가 나타납니다.
3. 연결 이름에 BigQuery 프로젝트의 이름을 입력합니다. AWS SCT는 왼쪽 패널의 트리에 이 이름을 표시합니다.
4. 키 경로에는 서비스 계정 키 파일의 경로를 입력합니다. 이 파일 생성에 대한 자세한 내용은 BigQuery를 소스로 사용하기 위한 권한 섹션을 참조하세요.
5. Test Connection을 선택하여 AWS SCT가 소스 BigQuery 프로젝트에 연결할 수 있는지 확인합니다.
6. 연결을 선택하여 소스 BigQuery 프로젝트에 연결합니다.

BigQuery를 AWS SCT에서 소스로 사용 시 적용되는 제한 사항

AWS SCT에서 BigQuery를 소스로 사용하는 경우 적용되는 제한 사항은 다음과 같습니다.

- AWS SCT는 분석 함수의 하위 쿼리 변환을 지원하지 않습니다.
- BigQuery `SELECT AS STRUCT` 및 `SELECT AS VALUE` 문을 변환하는 데는 AWS SCT를 사용할 수 없습니다.
- AWS SCT는 다음과 같은 유형의 함수 변환을 지원하지 않습니다.
  - Approximate aggregate
  - Bit
  - Debugging
  - Federated query
  - Geography
  - 해시
  - Mathematical
  - Net
  - Statistical aggregate

  - UUID
- AWS SCT는 문자열 함수의 변환을 제한적으로 지원합니다.
- AWS SCT는 UNNEST 연산자 변환을 지원하지 않습니다.
- AWS SCT에서 상관 관계가 있는 조인 연산은 변환할 수 없습니다.
- AWS SCT는 QUALIFY, WINDOW, LIMIT, OFFSET 절의 변환을 지원하지 않습니다.
- 재귀 공통 테이블 표현식을 변환하는 데는 AWS SCT를 사용할 수 없습니다.
- AWS SCT는 VALUES 절 내에 하위 쿼리가 있는 INSERT 문 변환을 지원하지 않습니다.
- AWS SCT는 중첩 필드 및 반복 레코드에 대한 UPDATE 문 변환을 지원하지 않습니다.
- STRUCT 및 ARRAY 데이터 형식을 변환하는 데는 AWS SCT를 사용할 수 없습니다.

BigQuery에서 Amazon Redshift로의 변환 설정

BigQuery에서 Amazon Redshift로의 변환 설정을 편집하려면 AWS SCT에서 설정을 선택한 다음 변환 설정을 선택합니다. 상단 목록에서 Google BigQuery를 선택한 다음 Google BigQuery – Amazon Redshift를 선택합니다. AWS SCT는 BigQuery에서 Amazon Redshift로의 변환에 사용할 수 있는 모든 설정을 표시합니다.

AWS SCT에서 BigQuery를 Amazon Redshift로 변환하는 설정에는 다음과 같은 옵션이 포함됩니다.

- 변환된 코드에서 작업 항목이 포함된 설명의 수를 제한합니다.

  Add comments in the converted code for the action items of selected severity and higher에서 작업 항목의 심각도를 선택합니다. AWS SCT가 선택된 심각도 이상의 작업 항목에 대한 변환된 코드에 설명을 추가합니다.

  예를 들어, 변환된 코드의 설명 수를 최소화하려면 오류만을 선택합니다. 변환된 코드의 모든 작업 항목에 대한 설명을 포함하려면 모든 메시지를 선택합니다.
- AWS SCT가 대상 Amazon Redshift 클러스터에 적용할 수 있는 최대 테이블 수를 설정합니다.

  The maximum number of tables for the target Amazon Redshift cluster에서 AWS SCT가 Amazon Redshift 클러스터에 적용할 수 있는 테이블 수를 선택합니다.

  Amazon Redshift에는 여러 클러스터 노드 유형에 사용하는 테이블을 제한하는 할당량이 있습니다. 자동을 선택하면 AWS SCT가 노드 유형에 따라 대상 Amazon Redshift 클러스터에 적용되는 테이블 수를 결정합니다. 값을 수동으로 선택할 수도 있습니다. 자세한 내용은 Amazon Redshift 관리 가이드의 Amazon Redshift의 할당량 및 제한 섹션을 참조하세요.

Amazon Redshift 클러스터가 저장할 수 있는 수보다 이 숫자가 많아도 AWS SCT는 모든 소스 테이블을 변환합니다. AWS SCT는 변환된 코드를 프로젝트에 저장하며 대상 데이터베이스에는 적용하지 않습니다. 변환된 코드를 적용할 때 테이블의 Amazon Redshift 클러스터 할당량에 도달하면 AWS SCT에서 경고 메시지가 표시됩니다. 또한 AWS SCT는 테이블 수가 한도에 도달할 때까지 대상 Amazon Redshift 클러스터에 테이블을 적용합니다.

- Amazon Redshift 테이블 열에 압축을 적용합니다. 이렇게 하려면 Use compression encoding을 선택합니다.

  AWS SCT는 기본 Amazon Redshift 알고리즘을 사용하여 열에 압축 인코딩을 자동으로 할당합니다. 자세한 내용은 Amazon Redshift 데이터베이스 개발자 안내서의 압축 인코딩을 참조하세요.

  기본적으로 Amazon Redshift는 정렬 및 배포 키로 정의된 열에 압축을 적용하지 않습니다. 이 동작을 변경하여 이러한 열에 압축을 적용할 수 있습니다. 이렇게 하려면 Use compression encoding for KEY columns를 선택합니다. Use compression encoding 옵션을 선택한 경우에만 이 옵션을 선택할 수 있습니다.

BigQuery에서 Amazon Redshift로의 변환 최적화 설정

BigQuery에서 Amazon Redshift로의 변환 최적화 설정을 편집하려면 AWS SCT에서 설정을 선택한 다음 변환 설정을 선택합니다. 상단 목록에서 Google BigQuery를 선택한 다음 Google BigQuery – Amazon Redshift를 선택합니다. 왼쪽 창에서 Optimization strategies를 선택합니다. AWS SCT는 BigQuery에서 Amazon Redshift로의 변환에 대한 변환 최적화 설정을 표시합니다.

AWS SCT에서 BigQuery를 Amazon Redshift로 변환하는 변환 최적화 설정에는 다음과 같은 옵션이 포함됩니다.

- 자동 테이블 최적화 작업을 수행합니다. 이 작업을 수행하려면 Use Amazon Redshift automatic table tuning을 선택합니다.

  자동 테이블 최적화는 테이블 디자인을 자동으로 최적화하는 Amazon Redshift의 자체 조정 프로세스입니다. 자세한 내용은 Amazon Redshift 데이터베이스 개발자 안내서의 자동 테이블 최적화 작업을 참조하세요.

  자동 테이블 최적화만 사용하려면 Initial key selection strategy에서 없음을 선택합니다.
- 전략을 사용하여 정렬 및 배포 키를 선택합니다.

Amazon Redshift 메타데이터, 통계 정보 또는 두 옵션을 모두 사용하여 정렬 및 배포 키를 선택할 수 있습니다. Optimization strategies 탭의 Initial key selection strategy에서 다음 옵션 중 하나를 선택합니다.

- 메타데이터 사용, 통계 정보 무시
- 메타데이터 무시, 통계 정보 사용
- 메타데이터 및 통계 정보 사용

선택한 옵션에 따라 최적화 전략을 선택할 수 있습니다. 그런 다음 각 전략에 대해 값(0~100)을 입력합니다. 이러한 값은 각 전략의 가중치를 정의합니다. AWS SCT는 이러한 가중치 값을 사용하여 각 규칙이 배포 및 정렬 키 선택에 미치는 영향을 정의합니다. 기본값은 AWS 마이그레이션 모범 사례를 기반으로 합니다.

Find small tables 전략에서 작은 테이블의 크기를 정의할 수 있습니다. Min table row count 및 Max table row count에 테이블의 최소 및 최대 행 수를 입력하여 작은 테이블로 정의합니다. AWS SCT는 작은 테이블에 ALL 배포 스타일을 적용합니다. 이 경우 전체 테이블의 사본이 모든 노드에 배포됩니다.

- 전략 세부 정보를 구성합니다.

  각 최적화 전략의 가중치를 정의하는 것 외에 최적화 설정도 구성할 수 있습니다. 이 작업을 수행하려면 Conversion optimization을 선택합니다.

  - Sort key columns limit에 정렬 키의 최대 열 수를 입력합니다.
  - Skewed threshold value에 열에 대한 스큐된 값의 백분율(0~100)을 입력합니다. AWS SCT는 스큐 값이 임계값보다 큰 열을 배포 키의 후보 목록에서 제외합니다. AWS SCT는 열의 스큐된 값을 전체 레코드 수에 대한 가장 일반적인 값 발생 횟수의 백분율로 정의합니다.
  - Top N queries from the query history table에 분석할 가장 자주 사용되는 쿼리의 수(1~100)를 입력합니다.
  - Select statistics user에서 쿼리 통계를 분석하려는 데이터베이스 사용자를 선택합니다.

  또한 Optimization strategies 탭에서 Find small tables 전략을 위한 작은 테이블의 크기를 정의할 수 있습니다. Min table row count 및 Max table row count에 테이블의 최소 및 최대 행 수를 입력하여 작은 테이블로 간주합니다. AWS SCT는 작은 테이블에 ALL 배포 스타일을 적용합니다. 이 경우 전체 테이블의 사본이 모든 노드에 배포됩니다.

Greenplum Database를 AWS SCT에 대한 소스로 사용

AWS SCT를 사용하여 스키마, 코드 객체 및 애플리케이션 코드를 Greenplum Database에서 Amazon Redshift로 변환할 수 있습니다.

Greenplum Database를 소스로 사용하기 위한 권한

Greenplum Database를 소스로 사용하기 위해 필요한 권한은 다음과 같습니다.

- CONNECT ON DATABASE *<database_name>*
- USAGE ON SCHEMA *<schema_name>*
- SELECT ON *<schema_name>.<table_name>*
- SELECT ON SEQUENCE *<schema_name>.<sequence_name>*

이전 예제에서 다음과 같이 자리 표시자를 바꿉니다.

- *database_name*을 소스 데이터베이스의 이름으로 바꿉니다.
- *schema_name*을 소스 스키마의 이름으로 바꿉니다.
- *table_name*을 소스 테이블의 이름으로 바꿉니다.
- *sequence_name*을 시퀀스 이름으로 바꿉니다.

Greenplum Database에 소스로 연결

다음 절차에 따라 AWS SCT를 사용하여 Greenplum 소스 데이터베이스에 연결합니다.

Greenplum 소스 데이터베이스에 연결하려면

1. AWS Schema Conversion Tool에서 소스 추가를 선택합니다.
2. SAP ASE를 선택하고 다음을 선택합니다.

   소스 추가 대화 상자가 나타납니다.
3. 연결 이름에 데이터베이스의 이름을 입력합니다. AWS SCT는 왼쪽 패널의 트리에 이 이름을 표시합니다.
4. AWS Secrets Manager의 데이터베이스 보안 인증 정보를 사용하거나 수동으로 입력합니다.

   - Secrets Manager의 데이터베이스 보안 인증 정보를 사용하려면 다음 지침을 따릅니다.
     1. AWS Secret에서 보안 암호의 이름을 선택합니다.

2. Populate를 선택하여 Secrets Manager에서 데이터베이스 연결 대화 상자에 있는 모든 값을 자동으로 채웁니다.

Secrets Manager의 데이터베이스 보안 인증 사용에 대한 자세한 내용은 AWS Secrets Manager 사용 섹션을 참조하세요.

- Greenplum 소스 데이터베이스 보안 인증 정보를 수동으로 입력하려면 다음 지침을 사용합니다.

파라미터	작업
[Server name]	소스 데이터베이스 서버의 도메인 이름 시스템(DNS) 이름 또는 IP 주소를 입력합니다.
[Server port]	소스 데이터베이스 서버에 연결하는 데 사용되는 포트를 입력합니다.
데이터베이스	Greenplum 데이터베이스의 이름을 입력합니다.
User name 및 Password	소스 데이터베이스 서버에 연결하는 데 필요한 데이터베이스 보안 인증 정보를 입력합니다. 프로젝트에서 데이터베이스에 연결하도록 선택한 경우에만 AWS SCT에서 암호를 사용하여 소스 데이터베이스에 연결합니다. 소스 데이터베이스의 암호가 노출될 위험을 방지하기 위해 AWS SCT는 기본적으로 암호를 저장하지 않습니다. AWS SCT 프로젝트를 닫았다 다시 열 경우 필요에 따라 소스 데이터베이스에 연결하기 위한 암호를 입력하라는 메시지가 표시됩니다.
Use SSL(SSL 사용)	SSL(Secure Sockets Layer)을 사용하여 데이터베이스에 연결하려면 이 옵션을 선택합니다. SSL 탭에서 다음 추가 정보를 적절히 제공합니다. • Verify server certificate: 트러스트 스토어를 사용하여 서버 인증서를 확인하려면 이 옵션을 선택합니다. • 트러스트 스토어: 인증서가 있는 트러스트 스토어의 위치입니다.

파라미터	작업
Store password	AWS SCT는 SSL 인증서와 데이터베이스 암호를 저장할 안전한 볼트를 생성합니다. 이 옵션을 켜면 데이터베이스 암호를 저장하고 암호 입력 없이 빠르게 데이터베이스에 연결할 수 있습니다.
Greenplum Database driver path	소스 데이터베이스에 연결할 때 사용할 드라이버의 경로를 입력합니다. 자세한 내용은 필수 데이터베이스 드라이버 다운로드 섹션을 참조하세요. 드라이버 경로를 전역 프로젝트 설정에 저장할 경우 드라이버 경로가 연결 대화 상자에 표시되지 않습니다. 자세한 내용은 전역 설정에 드라이버 경로 저장 섹션을 참조하세요.

5. Test Connection을 선택하여 AWS SCT가 소스 데이터베이스에 연결할 수 있는지 확인합니다.
6. 연결을 선택하여 소스 데이터베이스에 연결합니다.

Greenplum에서 Amazon Redshift로의 변환 설정

Greenplum에서 Amazon Redshift로의 변환 설정을 편집하려면 AWS SCT에서 설정을 선택한 다음 변환 설정을 선택합니다. 상단 목록에서 Greenplum을 선택한 다음 Greenplum – Amazon Redshift를 선택합니다. AWS SCT는 Greenplum에서 Amazon Redshift로의 변환에 사용할 수 있는 모든 설정을 표시합니다.

AWS SCT에서 Greenplum을 Amazon Redshift로 변환하는 설정에는 다음과 같은 옵션이 포함됩니다.

- 변환된 코드에서 작업 항목이 포함된 설명의 수를 제한합니다.

  Add comments in the converted code for the action items of selected severity and higher에서 작업 항목의 심각도를 선택합니다. AWS SCT가 선택된 심각도 이상의 작업 항목에 대한 변환된 코드에 설명을 추가합니다.

  예를 들어, 변환된 코드의 설명 수를 최소화하려면 오류만을 선택합니다. 변환된 코드의 모든 작업 항목에 대한 설명을 포함하려면 모든 메시지를 선택합니다.
- AWS SCT가 대상 Amazon Redshift 클러스터에 적용할 수 있는 최대 테이블 수를 설정합니다.

  The maximum number of tables for the target Amazon Redshift cluster에서 AWS SCT가 Amazon Redshift 클러스터에 적용할 수 있는 테이블 수를 선택합니다.

Amazon Redshift에는 여러 클러스터 노드 유형에 사용하는 테이블을 제한하는 할당량이 있습니다. 자동을 선택하면 AWS SCT가 노드 유형에 따라 대상 Amazon Redshift 클러스터에 적용되는 테이블 수를 결정합니다. 값을 수동으로 선택할 수도 있습니다. 자세한 내용은 Amazon Redshift 관리 가이드의 Amazon Redshift의 할당량 및 제한 섹션을 참조하세요.

Amazon Redshift 클러스터가 저장할 수 있는 수보다 이 숫자가 많아도 AWS SCT는 모든 소스 테이블을 변환합니다. AWS SCT는 변환된 코드를 프로젝트에 저장하며 대상 데이터베이스에는 적용하지 않습니다. 변환된 코드를 적용할 때 테이블의 Amazon Redshift 클러스터 할당량에 도달하면 AWS SCT에서 경고 메시지가 표시됩니다. 또한 AWS SCT는 테이블 수가 한도에 도달할 때까지 대상 Amazon Redshift 클러스터에 테이블을 적용합니다.

- 소스 테이블의 파티션을 Amazon Redshift의 개별 테이블로 마이그레이션합니다. 이 작업을 수행하려면 Use the UNION ALL view를 선택하고 AWS SCT가 단일 소스 테이블에 대해 생성할 수 있는 대상 테이블의 최대 수를 입력합니다.

  Amazon Redshift는 테이블 파티셔닝을 지원하지 않습니다. 이 동작을 에뮬레이션하고 쿼리를 더 빠르게 실행하기 위해 AWS SCT는 소스 테이블의 각 파티션을 Amazon Redshift의 개별 테이블로 마이그레이션할 수 있습니다. 그런 다음 AWS SCT는 이러한 모든 테이블의 데이터를 포함하는 보기를 생성합니다.

  AWS SCT는 소스 테이블의 파티션 수를 자동으로 결정합니다. 소스 테이블 파티셔닝 유형에 따라서는 이 숫자가 Amazon Redshift 클러스터에 적용할 수 있는 테이블의 할당량을 초과할 수 있습니다. 이 할당량에 도달하지 않도록 하려면 AWS SCT가 단일 소스 테이블의 파티션에 대해 생성할 수 있는 최대 대상 테이블 수를 입력합니다. 기본 옵션은 368개 테이블이며, 이는 1년 중 366일 동안의 파티션과 `NO RANGE` 및 `UNKNOWN` 파티션에 대한 테이블 두 개를 나타냅니다.
- Amazon Redshift 테이블 열에 압축을 적용합니다. 이렇게 하려면 Use compression encoding을 선택합니다.

  AWS SCT는 기본 Amazon Redshift 알고리즘을 사용하여 열에 압축 인코딩을 자동으로 할당합니다. 자세한 내용은 Amazon Redshift 데이터베이스 개발자 안내서의 압축 인코딩을 참조하세요.

  기본적으로 Amazon Redshift는 정렬 및 배포 키로 정의된 열에 압축을 적용하지 않습니다. 이 동작을 변경하여 이러한 열에 압축을 적용할 수 있습니다. 이렇게 하려면 Use compression encoding for KEY columns를 선택합니다. Use compression encoding 옵션을 선택한 경우에만 이 옵션을 선택할 수 있습니다.

Greenplum에서 Amazon Redshift로의 변환 최적화 설정

Greenplum에서 Amazon Redshift로의 변환 최적화 설정을 편집하려면 AWS SCT에서 설정을 선택한 다음 변환 설정을 선택합니다. 상단 목록에서 Greenplum을 선택한 다음 Greenplum – Amazon Redshift를 선택합니다. 왼쪽 창에서 Optimization strategies를 선택합니다. AWS SCT는 Greenplum에서 Amazon Redshift로의 변환에 대한 변환 최적화 설정을 표시합니다.

AWS SCT에서 Greenplum을 Amazon Redshift로 변환하는 변환 최적화 설정에는 다음과 같은 옵션이 포함됩니다.

- 자동 테이블 최적화 작업을 수행합니다. 이 작업을 수행하려면 Use Amazon Redshift automatic table tuning을 선택합니다.

  자동 테이블 최적화는 테이블 디자인을 자동으로 최적화하는 Amazon Redshift의 자체 조정 프로세스입니다. 자세한 내용은 Amazon Redshift 데이터베이스 개발자 안내서의 자동 테이블 최적화 작업을 참조하세요.

  자동 테이블 최적화만 사용하려면 Initial key selection strategy에서 없음을 선택합니다.
- 전략을 사용하여 정렬 및 배포 키를 선택합니다.

  Amazon Redshift 메타데이터, 통계 정보 또는 두 옵션을 모두 사용하여 정렬 및 배포 키를 선택할 수 있습니다. Optimization strategies 탭의 Initial key selection strategy에서 다음 옵션 중 하나를 선택합니다.
  - 메타데이터 사용, 통계 정보 무시
  - 메타데이터 무시, 통계 정보 사용
  - 메타데이터 및 통계 정보 사용

  선택한 옵션에 따라 최적화 전략을 선택할 수 있습니다. 그런 다음 각 전략에 대해 값(0~100)을 입력합니다. 이러한 값은 각 전략의 가중치를 정의합니다. AWS SCT는 이러한 가중치 값을 사용하여 각 규칙이 배포 및 정렬 키 선택에 미치는 영향을 정의합니다. 기본값은 AWS 마이그레이션 모범 사례를 기반으로 합니다.

  Find small tables 전략에서 작은 테이블의 크기를 정의할 수 있습니다. Min table row count 및 Max table row count에 테이블의 최소 및 최대 행 수를 입력하여 작은 테이블로 정의합니다. AWS SCT는 작은 테이블에 ALL 배포 스타일을 적용합니다. 이 경우 전체 테이블의 사본이 모든 노드에 배포됩니다.
- 전략 세부 정보를 구성합니다.

각 최적화 전략의 가중치를 정의하는 것 외에 최적화 설정도 구성할 수 있습니다. 이 작업을 수행하려면 Conversion optimization을 선택합니다.

- Sort key columns limit에 정렬 키의 최대 열 수를 입력합니다.
- Skewed threshold value에 열에 대한 스큐된 값의 백분율(0~100)을 입력합니다. AWS SCT는 스큐 값이 임계값보다 큰 열을 배포 키의 후보 목록에서 제외합니다. AWS SCT는 열의 스큐된 값을 전체 레코드 수에 대한 가장 일반적인 값 발생 횟수의 백분율로 정의합니다.
- Top N queries from the query history table에 분석할 가장 자주 사용되는 쿼리의 수(1~100)를 입력합니다.
- Select statistics user에서 쿼리 통계를 분석하려는 데이터베이스 사용자를 선택합니다.

또한 Optimization strategies 탭에서 Find small tables 전략을 위한 작은 테이블의 크기를 정의할 수 있습니다. Min table row count 및 Max table row count에 테이블의 최소 및 최대 행 수를 입력하여 작은 테이블로 간주합니다. AWS SCT는 작은 테이블에 ALL 배포 스타일을 적용합니다. 이 경우 전체 테이블의 사본이 모든 노드에 배포됩니다.

Netezza를 AWS SCT에 대한 소스로 사용

AWS SCT를 사용하여 스키마, 코드 객체 및 애플리케이션 코드를 Netezza에서 Amazon Redshift로 변환할 수 있습니다.

Netezza를 소스로 사용하기 위한 권한

Netezza를 소스로 사용하기 위해 필요한 권한은 다음과 같습니다.

- select on system.definition_schema.system view
- select on system.definition_schema.system table
- select on system.definition_schema.management table
- list on *<database_name>*
- list on *<schema_name>*
- list on *<database_name>*.all.table
- list on *<database_name>*.all.external table
- list on *<database_name>*.all.view
- list on *<database_name>*.all.materialized view
- list on *<database_name>*.all.procedure

- list on *<database_name>*.all.sequence
- list on *<database_name>*.all.function
- list on *<database_name>*.all.aggregate

이전 예제에서 다음과 같이 자리 표시자를 바꿉니다.

- *database_name*을 소스 데이터베이스의 이름으로 바꿉니다.
- *schema_name*을 소스 스키마의 이름으로 바꿉니다.

AWS SCT는 다음과 같은 시스템 테이블과 보기에 대한 액세스 권한이 필요합니다. 위 목록의 `system.definition_schema.system view` 및 `system.definition_schema.system tables`에 대한 액세스 권한을 부여하는 대신 다음과 같은 객체에 대한 액세스 권한을 부여할 수 있습니다.

- select on system.definition_schema._t_aggregate
- select on system.definition_schema._t_class
- select on system.definition_schema._t_constraint
- select on system.definition_schema._t_const_relattr
- select on system.definition_schema._t_database
- select on system.definition_schema._t_grpobj_priv
- select on system.definition_schema._t_grpusr
- select on system.definition_schema._t_hist_config
- select on system.definition_schema._t_object
- select on system.definition_schema._t_object_classes
- select on system.definition_schema._t_proc
- select on system.definition_schema._t_type
- select on system.definition_schema._t_user
- select on system.definition_schema._t_usrobj_priv
- select on system.definition_schema._vt_sequence
- select on system.definition_schema._v_aggregate
- select on system.definition_schema._v_constraint_depends

- select on system.definition_schema._v_database
- select on system.definition_schema._v_datatype
- select on system.definition_schema._v_dslice
- select on system.definition_schema._v_function
- select on system.definition_schema._v_group
- select on system.definition_schema._v_obj_relation
- select on system.definition_schema._v_obj_relation_xdb
- select on system.definition_schema._v_procedure
- select on system.definition_schema._v_relation_column
- select on system.definition_schema._v_relation_keydata
- select on system.definition_schema._v_relobjclasses
- select on system.definition_schema._v_schema_xdb
- select on system.definition_schema._v_sequence
- select on system.definition_schema._v_synonym
- select on system.definition_schema._v_system_info
- select on system.definition_schema._v_sys_constraint
- select on system.definition_schema._v_sys_object_dslice_info
- select on system.definition_schema._v_sys_user
- select on system.definition_schema._v_table
- select on system.definition_schema._v_table_constraint
- select on system.definition_schema._v_table_dist_map
- select on system.definition_schema._v_table_organize_column
- select on system.definition_schema._v_table_storage_stat
- select on system.definition_schema._v_user
- select on system.definition_schema._v_view
- select on system.information_schema._v_relation_column
- select on system.information_schema._v_table
- select on $hist_column_access_*

Netezza에 소스로 연결

다음 절차에 따라 AWS Schema Conversion Tool을 사용하여 Netezza 소스 데이터베이스에 연결합니다.

Netezza 소스 데이터베이스에 연결하려면

1. AWS Schema Conversion Tool에서 소스 추가를 선택합니다.
2. Netezza를 선택한 후 다음을 선택합니다.

   소스 추가 대화 상자가 나타납니다.
3. 연결 이름에 데이터베이스의 이름을 입력합니다. AWS SCT는 왼쪽 패널의 트리에 이 이름을 표시합니다.
4. AWS Secrets Manager의 데이터베이스 보안 인증 정보를 사용하거나 수동으로 입력합니다.

   - Secrets Manager의 데이터베이스 보안 인증 정보를 사용하려면 다음 지침을 따릅니다.
     1. AWS Secret에서 보안 암호의 이름을 선택합니다.
     2. Populate를 선택하여 Secrets Manager에서 데이터베이스 연결 대화 상자에 있는 모든 값을 자동으로 채웁니다.

     Secrets Manager의 데이터베이스 보안 인증 사용에 대한 자세한 내용은 AWS Secrets Manager 사용 섹션을 참조하세요.
   - Netezza 소스 데이터베이스 연결 정보를 수동으로 입력하려면 다음 지침을 사용합니다.

파라미터	작업
[Server name]	소스 데이터베이스 서버의 도메인 이름 시스템(DNS) 이름 또는 IP 주소를 입력합니다.
[Server port]	소스 데이터베이스 서버에 연결하는 데 사용되는 포트를 입력합니다.
User name 및 Password	소스 데이터베이스 서버에 연결하는 데 필요한 데이터베이스 보안 인증 정보를 입력합니다. 프로젝트에서 데이터베이스에 연결하도록 선택한 경우에만 AWS SCT에서 암호를 사용하여 소스 데이터베이스에 연결합니다. 소스 데이터베이스의 암호가 노출될 위험을 방지하

파라미터	작업
	기 위해 AWS SCT는 기본적으로 암호를 저장하지 않습니다. AWS SCT 프로젝트를 닫았다 다시 열 경우 필요에 따라 소스 데이터베이스에 연결하기 위한 암호를 입력하라는 메시지가 표시됩니다.
Store password	AWS SCT는 SSL 인증서와 데이터베이스 암호를 저장할 안전한 볼트를 생성합니다. 이 옵션을 켜면 데이터베이스 암호를 저장하고 암호 입력 없이 빠르게 데이터베이스에 연결할 수 있습니다.
Netezza driver path	소스 데이터베이스에 연결할 때 사용할 드라이버의 경로를 입력합니다. 자세한 내용은 필수 데이터베이스 드라이버 다운로드 섹션을 참조하세요. 드라이버 경로를 전역 프로젝트 설정에 저장할 경우 드라이버 경로가 연결 대화 상자에 표시되지 않습니다. 자세한 내용은 전역 설정에 드라이버 경로 저장 섹션을 참조하세요.

5. Test Connection을 선택하여 AWS SCT가 소스 데이터베이스에 연결할 수 있는지 확인합니다.
6. 연결을 선택하여 소스 데이터베이스에 연결합니다.

지속적인 데이터 복제 구성

Netezza 데이터베이스 스키마를 변환하여 Amazon Redshift 데이터베이스에 적용한 후, AWS SCT 데이터 추출 에이전트를 사용하여 데이터를 마이그레이션할 수 있습니다. 에이전트가 데이터를 추출하여 Amazon S3 버킷에 업로드합니다. 그런 다음 AWS SCT를 사용하여 Amazon S3에서 Amazon Redshift로 데이터를 복사할 수 있습니다.

마이그레이션 프로세스 중에 소스 데이터베이스의 데이터가 변경되는 경우 AWS SCT 데이터 추출 에이전트를 사용하여 진행 중인 변경 사항을 캡처할 수 있습니다. 그러면 초기 데이터 마이그레이션을 완료한 후 대상 데이터베이스에 이러한 진행 중인 변경 사항을 복제할 수 있습니다. 이 프로세스를 지속적인 데이터 복제 또는 변경 데이터 캡처(CDC)라고 합니다.

Netezza에서 Amazon Redshift로의 마이그레이션을 위한 지속적인 데이터 복제를 구성하려면

1. 소스 데이터베이스에서 기록 데이터베이스를 생성합니다. Netezza 명령줄 인터페이스(CLI)에서 다음 코드 예제를 사용할 수 있습니다.

```
nzhistcreatedb -d history_database_name -t query -v 1 -u load_user -o histdb_owner
 -p your_password
```

이전 예제에서 *history_database_name*을 기록 데이터베이스의 이름으로 바꿉니다. 그 다음으로, *load_user*를 데이터베이스에 기록 데이터를 로드하기 위해 정의한 사용자 이름으로 바꿉니다. 그런 다음, *histdb_owner*를 기록 데이터베이스의 소유자로 정의한 사용자 이름으로 바꿉니다. 이 사용자를 이미 생성하고 CREATE DATABASE 권한을 부여했는지 확인합니다. 마지막으로 *your_password*를 안전한 암호로 바꿉니다.

2. 기록 로깅을 구성합니다. 이 작업을 수행하려면 다음 코드 예제를 사용합니다.

```
CREATE HISTORY CONFIGURATION history_configuration_name HISTTYPE QUERY
    DATABASE history_database_name USER load_user PASSWORD your_password COLLECT
 PLAN, COLUMN
    LOADINTERVAL 1 LOADMINTHRESHOLD 0 LOADMAXTHRESHOLD 0 STORAGELIMIT 25
    LOADRETRY 2 VERSION 1;
```

이전 예제에서 *history_configuration_name* 및 *history_database_name*을 기록 구성 및 기록 데이터베이스의 이름으로 바꿉니다. 그 다음으로, *load_user*를 데이터베이스에 기록 데이터를 로드하기 위해 정의한 사용자 이름으로 바꿉니다. 그런 다음 *your_password*를 안전한 암호로 바꿉니다.

3. 기록 데이터베이스의 모든 테이블에 읽기 권한을 부여합니다. 다음 코드 예제를 사용하여 SELECT 권한을 부여할 수 있습니다.

```
GRANT SELECT ON history_database_name.ALL.TABLE TO your_user;
```

이전 예제에서 *history_database_name*을 기록 데이터베이스의 이름으로 바꿉니다. 다음으로, *your_user*를 Netezza 데이터베이스 작업에 필요한 최소 권한을 가진 사용자 이름으로 바꿉니다. AWS SCT에서 이 데이터베이스 사용자의 보안 인증 정보를 사용합니다.

4. 소스 스키마의 각 테이블에 대한 통계를 수집하여 열의 카디널리티에 대한 정보를 가져옵니다. 다음 명령을 사용하여 기록 데이터베이스에 통계를 생성할 수 있습니다.

```
GENERATE STATISTICS on "schema_name"."table_name";
```

이전 예제에서 *schema_name* 및 *table_name*을 데이터베이스 스키마 및 테이블의 이름으로 바꿉니다.

5. 다음 쿼리를 실행하여 사전 조건을 완료해야 합니다.

```
SELECT COUNT(*)
 FROM history_database_name.history_schema_name."$hist_column_access_N";
```

이전 예제에서 *history_database_name* 및 *history_schema_name*을 기록 데이터베이스 및 스키마의 이름으로 바꿉니다. 그 다음, *N*을 기록 데이터베이스의 버전 번호로 바꿉니다. 기록 데이터베이스 버전에 대한 자세한 내용은 IBM Netezza 설명서를 참조하세요.

6. 데이터 추출 에이전트를 설치합니다. 자세한 내용은 추출 에이전트 설치 섹션을 참조하세요.

   모든 추출기 인스턴스에 대해 `settings.properties` 파일의 `{working.folder}` 파라미터가 동일한 폴더를 가리키는지 확인합니다. 이 경우 추출기가 CDC 세션을 조정하고 모든 하위 작업에 대한 단일 트랜잭션 지점을 사용할 수 있습니다.

7. 데이터 추출 에이전트를 등록합니다. 자세한 내용은 에 추출 에이전트 등록 AWS Schema Conversion Tool 섹션을 참조하세요.

8. CDC 작업을 생성합니다. 자세한 내용은 AWS SCT 작업 생성, 실행, 모니터링 섹션을 참조하세요.

   a. AWS SCT에서 프로젝트를 엽니다. 왼쪽 창에서 소스 테이블을 선택합니다. 컨텍스트(마우스 오른쪽 버튼 클릭) 메뉴를 열고 Create local task를 선택합니다.

   b. 작업 이름에 데이터 마이그레이션 작업을 설명하는 이름을 입력합니다.

   c. Migration mode에서 Extract, upload, and copy를 선택합니다.

   d. Enable CDC를 선택합니다.

   e. CDC settings 탭을 선택하고 CDC 세션의 범위와 일정을 정의합니다.

   f. Test task를 선택하여 작업 폴더, Amazon S3 버킷 및 Amazon Redshift 데이터 웨어하우스에 연결할 수 있는지 확인합니다.

   g. 생성을 선택하여 작업을 생성합니다.

   h. 작업 탭을 선택하고 목록에서 작업을 선택한 다음 시작을 선택합니다.

9. AWS SCT 작업은 대상 데이터베이스의 트랜잭션 일관성을 유지합니다. 데이터 추출 에이전트는 소스의 트랜잭션을 트랜잭션 ID 순서대로 복제합니다.

   마이그레이션 세션이 중지되거나 실패한 경우 CDC 처리도 중지됩니다.

Netezza에서 Amazon Redshift로의 변환 설정

Netezza에서 Amazon Redshift로의 변환 설정을 편집하려면 AWS SCT에서 설정을 선택한 다음 변환 설정을 선택합니다. 상단 목록에서 Netezza를 선택한 다음 Netezza – Amazon Redshift를 선택합니다. AWS SCT는 Netezza에서 Amazon Redshift로의 변환에 사용할 수 있는 모든 설정을 표시합니다.

AWS SCT에서 Netezza를 Amazon Redshift로 변환하는 설정에는 다음과 같은 옵션이 포함됩니다.

- 변환된 코드에서 작업 항목이 포함된 설명의 수를 제한합니다.

  Add comments in the converted code for the action items of selected severity and higher에서 작업 항목의 심각도를 선택합니다. AWS SCT가 선택된 심각도 이상의 작업 항목에 대한 변환된 코드에 설명을 추가합니다.

  예를 들어, 변환된 코드의 설명 수를 최소화하려면 오류만을 선택합니다. 변환된 코드의 모든 작업 항목에 대한 설명을 포함하려면 모든 메시지를 선택합니다.

- AWS SCT가 대상 Amazon Redshift 클러스터에 적용할 수 있는 최대 테이블 수를 설정합니다.

  The maximum number of tables for the target Amazon Redshift cluster에서 AWS SCT가 Amazon Redshift 클러스터에 적용할 수 있는 테이블 수를 선택합니다.

  Amazon Redshift에는 여러 클러스터 노드 유형에 사용하는 테이블을 제한하는 할당량이 있습니다. 자동을 선택하면 AWS SCT가 노드 유형에 따라 대상 Amazon Redshift 클러스터에 적용되는 테이블 수를 결정합니다. 값을 수동으로 선택할 수도 있습니다. 자세한 내용은 Amazon Redshift 관리 가이드의 Amazon Redshift의 할당량 및 제한 섹션을 참조하세요.

  Amazon Redshift 클러스터가 저장할 수 있는 수보다 이 숫자가 많아도 AWS SCT는 모든 소스 테이블을 변환합니다. AWS SCT는 변환된 코드를 프로젝트에 저장하며 대상 데이터베이스에는 적용하지 않습니다. 변환된 코드를 적용할 때 테이블의 Amazon Redshift 클러스터 할당량에 도달하면 AWS SCT에서 경고 메시지가 표시됩니다. 또한 AWS SCT는 테이블 수가 한도에 도달할 때까지 대상 Amazon Redshift 클러스터에 테이블을 적용합니다.

- Amazon Redshift 테이블 열에 압축을 적용합니다. 이렇게 하려면 Use compression encoding을 선택합니다.

  AWS SCT는 기본 Amazon Redshift 알고리즘을 사용하여 열에 압축 인코딩을 자동으로 할당합니다. 자세한 내용은 Amazon Redshift 데이터베이스 개발자 안내서의 압축 인코딩을 참조하세요.

  기본적으로 Amazon Redshift는 정렬 및 배포 키로 정의된 열에 압축을 적용하지 않습니다. 이 동작을 변경하여 이러한 열에 압축을 적용할 수 있습니다. 이렇게 하려면 Use compression encoding for

KEY columns를 선택합니다. Use compression encoding 옵션을 선택한 경우에만 이 옵션을 선택할 수 있습니다.

Netezza에서 Amazon Redshift로의 변환 최적화 설정

Netezza에서 Amazon Redshift로의 변환 최적화 설정을 편집하려면 AWS SCT에서 설정을 선택한 다음 변환 설정을 선택합니다. 상단 목록에서 Netezza를 선택한 다음 Netezza – Amazon Redshift를 선택합니다. 왼쪽 창에서 Optimization strategies를 선택합니다. AWS SCT는 Netezza에서 Amazon Redshift로의 변환에 대한 변환 최적화 설정을 표시합니다.

AWS SCT에서 Netezza를 Amazon Redshift로 변환하는 변환 최적화 설정에는 다음과 같은 옵션이 포함됩니다.

- 자동 테이블 최적화 작업을 수행합니다. 이 작업을 수행하려면 Use Amazon Redshift automatic table tuning을 선택합니다.

  자동 테이블 최적화는 테이블 디자인을 자동으로 최적화하는 Amazon Redshift의 자체 조정 프로세스입니다. 자세한 내용은 Amazon Redshift 데이터베이스 개발자 안내서의 자동 테이블 최적화 작업을 참조하세요.

  자동 테이블 최적화만 사용하려면 Initial key selection strategy에서 없음을 선택합니다.
- 전략을 사용하여 정렬 및 배포 키를 선택합니다.

  Amazon Redshift 메타데이터, 통계 정보 또는 두 옵션을 모두 사용하여 정렬 및 배포 키를 선택할 수 있습니다. Optimization strategies 탭의 Initial key selection strategy에서 다음 옵션 중 하나를 선택합니다.
  - 메타데이터 사용, 통계 정보 무시
  - 메타데이터 무시, 통계 정보 사용
  - 메타데이터 및 통계 정보 사용

  선택한 옵션에 따라 최적화 전략을 선택할 수 있습니다. 그런 다음 각 전략에 대해 값(0~100)을 입력합니다. 이러한 값은 각 전략의 가중치를 정의합니다. AWS SCT는 이러한 가중치 값을 사용하여 각 규칙이 배포 및 정렬 키 선택에 미치는 영향을 정의합니다. 기본값은 AWS 마이그레이션 모범 사례를 기반으로 합니다.

  Find small tables 전략에서 작은 테이블의 크기를 정의할 수 있습니다. Min table row count 및 Max table row count에 테이블의 최소 및 최대 행 수를 입력하여 작은 테이블로 정의합니다. AWS SCT는

작은 테이블에 ALL 배포 스타일을 적용합니다. 이 경우 전체 테이블의 사본이 모든 노드에 배포됩니다.

- 전략 세부 정보를 구성합니다.

  각 최적화 전략의 가중치를 정의하는 것 외에 최적화 설정도 구성할 수 있습니다. 이 작업을 수행하려면 Conversion optimization을 선택합니다.

  - Sort key columns limit에 정렬 키의 최대 열 수를 입력합니다.
  - Skewed threshold value에 열에 대한 스큐된 값의 백분율(0~100)을 입력합니다. AWS SCT는 스큐 값이 임계값보다 큰 열을 배포 키의 후보 목록에서 제외합니다. AWS SCT는 열의 스큐된 값을 전체 레코드 수에 대한 가장 일반적인 값 발생 횟수의 백분율로 정의합니다.
  - Top N queries from the query history table에 분석할 가장 자주 사용되는 쿼리의 수(1~100)를 입력합니다.
  - Select statistics user에서 쿼리 통계를 분석하려는 데이터베이스 사용자를 선택합니다.

  또한 Optimization strategies 탭에서 Find small tables 전략을 위한 작은 테이블의 크기를 정의할 수 있습니다. Min table row count 및 Max table row count에 테이블의 최소 및 최대 행 수를 입력하여 작은 테이블로 간주합니다. AWS SCT는 작은 테이블에 ALL 배포 스타일을 적용합니다. 이 경우 전체 테이블의 사본이 모든 노드에 배포됩니다.

Oracle Data Warehouse를 AWS SCT에 대한 소스로 사용

AWS SCT를 사용하면 스키마, 코드 객체 및 애플리케이션 코드를 Oracle Data Warehouse에서 Amazon Redshift로 변환하거나 조합하여 사용되는 Amazon Redshift 및 AWS Glue로 변환할 수 있습니다.

Oracle Data Warehouse를 소스로 사용하기 위한 권한

Oracle Data Warehouse를 소스로 사용하기 위해 필요한 권한은 다음과 같습니다.

- connect
- select_catalog_role
- select any dictionary

Oracle Data Warehouse에 소스로 연결

다음 절차에 따라 AWS Schema Conversion Tool을 사용하여 Oracle Data Warehouse 소스 데이터베이스에 연결합니다.

Oracle Data Warehouse 소스 데이터베이스에 연결하려면

1. AWS Schema Conversion Tool에서 소스 추가를 선택합니다.
2. Oracle를 선택한 후 다음을 선택합니다.

   소스 추가 대화 상자가 나타납니다.
3. 연결 이름에 데이터베이스의 이름을 입력합니다. AWS SCT는 왼쪽 패널의 트리에 이 이름을 표시합니다.
4. AWS Secrets Manager의 데이터베이스 보안 인증 정보를 사용하거나 수동으로 입력합니다.

   - Secrets Manager의 데이터베이스 보안 인증 정보를 사용하려면 다음 지침을 따릅니다.

     1. AWS Secret에서 보안 암호의 이름을 선택합니다.
     2. Populate를 선택하여 Secrets Manager에서 데이터베이스 연결 대화 상자에 있는 모든 값을 자동으로 채웁니다.

     Secrets Manager의 데이터베이스 보안 인증 사용에 대한 자세한 내용은 AWS Secrets Manager 사용 섹션을 참조하세요.
   - Oracle 소스 데이터 웨어하우스 연결 정보를 수동으로 입력하려면 다음 지침을 사용합니다.

파라미터	작업

유형	데이터베이스 연결 유형을 선택합니다. 유형에 따라 다음의 추가 정보를 제공합니다.
	• SID
	  • 서버 이름: 소스 데이터베이스 서버의 도메인 이름 시스템(DNS) 이름 또는 IP 주소입니다.
	  • Server port: 소스 데이터베이스 서버에 연결하는 데 사용되는 포트입니다.
	  • Oracle SID: Oracle System ID(SID)입니다. Oracle SID를 확인하려면 Oracle 데이터베이스에 다음 쿼리를 제출합니다.
	`SELECT sys_context('userenv','instance_name') AS SID FROM dual;`
	• Service Name

파라미터	작업
	• Server name: 소스 데이터베이스 서버의 DNS 이름 또는 IP 주소입니다. • Server port: 소스 데이터베이스 서버에 연결하는 데 사용되는 포트입니다. • Service Name: 연결할 Oracle 서비스의 이름입니다. • TNS alias • TNS file path: TNS(Transparent Network Substrate) 이름 연결 정보가 저장된 파일의 경로입니다. • TNS file path: 소스 데이터베이스에 연결하는 데 사용할 이 파일의 TNS 별칭입니다. • TNS connect identifier • TNS connect identifier: 등록된 TNS 연결 정보의 식별자입니다.
User name 및 Password	소스 데이터베이스 서버에 연결하는 데 필요한 데이터베이스 보안 인증 정보를 입력합니다. 프로젝트에서 데이터베이스에 연결하도록 선택한 경우에만 AWS SCT에서 암호를 사용하여 소스 데이터베이스에 연결합니다. 소스 데이터베이스의 암호가 노출될 위험을 방지하기 위해 AWS SCT는 기본적으로 암호를 저장하지 않습니다. AWS SCT 프로젝트를 닫았다 다시 열 경우 필요에 따라 소스 데이터베이스에 연결하기 위한 암호를 입력하라는 메시지가 표시됩니다.

파라미터	작업
Use SSL(SSL 사용)	SSL(Secure Sockets Layer)을 사용하여 데이터베이스에 연결하려면 이 옵션을 선택합니다. SSL 탭에서 다음 추가 정보를 적절히 제공합니다. • SSL 인증: 연결에 SSL 인증을 사용하려면 이 옵션을 선택합니다. • 트러스트 스토어: 인증서가 있는 트러스트 스토어의 위치입니다. • Key store: 프라이빗 키와 인증서가 포함된 키 스토어의 위치입니다. 이 값은 SSL 인증을 선택한 경우 필수이고, 그렇지 않은 경우 선택입니다.
Store password	AWS SCT는 SSL 인증서와 데이터베이스 암호를 저장할 안전한 볼트를 생성합니다. 이 옵션을 켜면 데이터베이스 암호를 저장하고 암호 입력 없이 빠르게 데이터베이스에 연결할 수 있습니다.
Oracle driver path	소스 데이터베이스에 연결할 때 사용할 드라이버의 경로를 입력합니다. 자세한 내용은 필수 데이터베이스 드라이버 다운로드 섹션을 참조하세요. 드라이버 경로를 전역 프로젝트 설정에 저장할 경우 드라이버 경로가 연결 대화 상자에 표시되지 않습니다. 자세한 내용은 전역 설정에 드라이버 경로 저장 섹션을 참조하세요.

5. Test Connection을 선택하여 AWS SCT가 소스 데이터베이스에 연결할 수 있는지 확인합니다.
6. 연결을 선택하여 소스 데이터베이스에 연결합니다.

Oracle Data Warehouse에서 Amazon Redshift로의 변환 설정

Oracle Data Warehouse에서 Amazon Redshift로의 변환 설정을 편집하려면 AWS SCT에서 설정을 선택한 다음 변환 설정을 선택합니다. 상단 목록에서 Oracle을 선택한 다음 Oracle – Amazon Redshift를 선택합니다. AWS SCT는 Oracle Data Warehouse에서 Amazon Redshift로의 변환에 사용할 수 있는 모든 설정을 표시합니다.

AWS SCT에서 Oracle Data Warehouse를 Amazon Redshift로 변환하는 설정에는 다음과 같은 옵션이 포함됩니다.

- 변환된 코드에서 작업 항목이 포함된 설명의 수를 제한합니다.

  Add comments in the converted code for the action items of selected severity and higher에서 작업 항목의 심각도를 선택합니다. AWS SCT가 선택된 심각도 이상의 작업 항목에 대한 변환된 코드에 설명을 추가합니다.

  예를 들어, 변환된 코드의 설명 수를 최소화하려면 오류만을 선택합니다. 변환된 코드의 모든 작업 항목에 대한 설명을 포함하려면 모든 메시지를 선택합니다.

- AWS SCT가 대상 Amazon Redshift 클러스터에 적용할 수 있는 최대 테이블 수를 설정합니다.

  The maximum number of tables for the target Amazon Redshift cluster에서 AWS SCT가 Amazon Redshift 클러스터에 적용할 수 있는 테이블 수를 선택합니다.

  Amazon Redshift에는 여러 클러스터 노드 유형에 사용하는 테이블을 제한하는 할당량이 있습니다. 자동을 선택하면 AWS SCT가 노드 유형에 따라 대상 Amazon Redshift 클러스터에 적용되는 테이블 수를 결정합니다. 값을 수동으로 선택할 수도 있습니다. 자세한 내용은 Amazon Redshift 관리 가이드의 Amazon Redshift의 할당량 및 제한 섹션을 참조하세요.

  Amazon Redshift 클러스터가 저장할 수 있는 수보다 이 숫자가 많아도 AWS SCT는 모든 소스 테이블을 변환합니다. AWS SCT는 변환된 코드를 프로젝트에 저장하며 대상 데이터베이스에는 적용하지 않습니다. 변환된 코드를 적용할 때 테이블의 Amazon Redshift 클러스터 할당량에 도달하면 AWS SCT에서 경고 메시지가 표시됩니다. 또한 AWS SCT는 테이블 수가 한도에 도달할 때까지 대상 Amazon Redshift 클러스터에 테이블을 적용합니다.

- 소스 테이블의 파티션을 Amazon Redshift의 개별 테이블로 마이그레이션합니다. 이 작업을 수행하려면 Use the UNION ALL view를 선택하고 AWS SCT가 단일 소스 테이블에 대해 생성할 수 있는 대상 테이블의 최대 수를 입력합니다.

  Amazon Redshift는 테이블 파티셔닝을 지원하지 않습니다. 이 동작을 에뮬레이션하고 쿼리를 더 빠르게 실행하기 위해 AWS SCT는 소스 테이블의 각 파티션을 Amazon Redshift의 개별 테이블로 마이그레이션할 수 있습니다. 그런 다음 AWS SCT는 이러한 모든 테이블의 데이터를 포함하는 보기를 생성합니다.

  AWS SCT는 소스 테이블의 파티션 수를 자동으로 결정합니다. 소스 테이블 파티셔닝 유형에 따라서는 이 숫자가 Amazon Redshift 클러스터에 적용할 수 있는 테이블의 할당량을 초과할 수 있습니다. 이 할당량에 도달하지 않도록 하려면 AWS SCT가 단일 소스 테이블의 파티션에 대해 생성할 수 있

는 최대 대상 테이블 수를 입력합니다. 기본 옵션은 368개 테이블이며, 이는 1년 중 366일 동안의 파티션과 NO RANGE 및 UNKNOWN 파티션에 대한 테이블 두 개를 나타냅니다.

- Amazon Redshift에서 지원하지 않는 날짜/시간 형식 요소를 사용하여 TO_CHAR, TO_DATE, TO_NUMBER 등의 데이터 유형 서식 설정 함수를 변환합니다. 기본적으로 AWS SCT는 확장 팩 함수를 사용하여 변환된 코드에서 지원되지 않는 이러한 형식 요소의 사용을 에뮬레이션합니다.

  Oracle의 날짜/시간 형식 모델에는 Amazon Redshift의 날짜/시간 형식 문자열과 비교하여 더 많은 요소가 포함되어 있습니다. 소스 코드에 Amazon Redshift가 지원하는 날짜/시간 형식 요소만 포함된 경우 변환된 코드에 확장 팩 함수가 필요하지 않습니다. 변환된 코드에서 확장 팩 함수를 사용하지 않으려면 Datetype format elements that you use in the Oracle code are similar to datetime format strings in Amazon Redshift를 선택합니다. 이 경우, 변환된 코드는 더 빠르게 작동합니다.

  Oracle의 숫자 형식 모델에는 Amazon Redshift의 숫자 형식 문자열과 비교하여 더 많은 요소가 포함되어 있습니다. 소스 코드에 Amazon Redshift가 지원하는 숫자 형식 요소만 포함된 경우 변환된 코드에 확장 팩 함수가 필요하지 않습니다. 변환된 코드에서 확장 팩 함수를 사용하지 않으려면 Numeric format elements that you use in the Oracle code are similar to numeric format strings in Amazon Redshift를 선택합니다. 이 경우, 변환된 코드는 더 빠르게 작동합니다.
- Oracle LEAD 및 LAG 분석 함수를 변환합니다. 기본적으로 AWS SCT가 각 LEAD 및 LAG 함수에 대한 작업 항목을 발생시킵니다.

  소스 코드가 이러한 함수의 오프셋에 대해 기본값을 사용하지 않는 경우 AWS SCT는 NVL 함수를 사용하여 이러한 함수의 사용을 에뮬레이션할 수 있습니다. 이렇게 하려면 Use the NVL function to emulate the behavior of Oracle LEAD and LAG functions를 선택합니다.
- Amazon Redshift 클러스터에서 기본 키와 고유 키의 동작을 에뮬레이션하려면 Emulate the behavior of primary and unique keys를 선택합니다.

  Amazon Redshift는 고유 키와 기본 키를 적용하지 않으며 정보 제공 목적으로만 사용합니다. 코드에서 이러한 제약 조건을 사용하는 경우에는 AWS SCT가 변환된 코드에서 해당 동작을 에뮬레이션해야 합니다.
- Amazon Redshift 테이블 열에 압축을 적용합니다. 이렇게 하려면 Use compression encoding을 선택합니다.

  AWS SCT는 기본 Amazon Redshift 알고리즘을 사용하여 열에 압축 인코딩을 자동으로 할당합니다. 자세한 내용은 Amazon Redshift 데이터베이스 개발자 안내서의 압축 인코딩을 참조하세요.

  기본적으로 Amazon Redshift는 정렬 및 배포 키로 정의된 열에 압축을 적용하지 않습니다. 이 동작을 변경하여 이러한 열에 압축을 적용할 수 있습니다. 이렇게 하려면 Use compression encoding for

KEY columns를 선택합니다. Use compression encoding 옵션을 선택한 경우에만 이 옵션을 선택할 수 있습니다.

Oracle Data Warehouse에서 Amazon Redshift로의 변환 최적화 설정

Oracle Data Warehouse에서 Amazon Redshift로의 변환 최적화 설정을 편집하려면 AWS SCT에서 설정을 선택한 다음 변환 설정을 선택합니다. 상단 목록에서 Oracle를 선택한 다음 Oracle – Amazon Redshift를 선택합니다. 왼쪽 창에서 Optimization strategies를 선택합니다. AWS SCT는 Oracle Data Warehouse에서 Amazon Redshift로의 변환에 대한 변환 최적화 설정을 표시합니다.

AWS SCT에서 Oracle Data Warehouse를 Amazon Redshift로 변환하는 변환 최적화 설정에는 다음과 같은 옵션이 포함됩니다.

- 자동 테이블 최적화 작업을 수행합니다. 이 작업을 수행하려면 Use Amazon Redshift automatic table tuning을 선택합니다.

  자동 테이블 최적화는 테이블 디자인을 자동으로 최적화하는 Amazon Redshift의 자체 조정 프로세스입니다. 자세한 내용은 Amazon Redshift 데이터베이스 개발자 안내서의 자동 테이블 최적화 작업을 참조하세요.

  자동 테이블 최적화만 사용하려면 Initial key selection strategy에서 없음을 선택합니다.
- 전략을 사용하여 정렬 및 배포 키를 선택합니다.

  Amazon Redshift 메타데이터, 통계 정보 또는 두 옵션을 모두 사용하여 정렬 및 배포 키를 선택할 수 있습니다. Optimization strategies 탭의 Initial key selection strategy에서 다음 옵션 중 하나를 선택합니다.
  - 메타데이터 사용, 통계 정보 무시
  - 메타데이터 무시, 통계 정보 사용
  - 메타데이터 및 통계 정보 사용

  선택한 옵션에 따라 최적화 전략을 선택할 수 있습니다. 그런 다음 각 전략에 대해 값(0~100)을 입력합니다. 이러한 값은 각 전략의 가중치를 정의합니다. AWS SCT는 이러한 가중치 값을 사용하여 각 규칙이 배포 및 정렬 키 선택에 미치는 영향을 정의합니다. 기본값은 AWS 마이그레이션 모범 사례를 기반으로 합니다.

  Find small tables 전략에서 작은 테이블의 크기를 정의할 수 있습니다. Min table row count 및 Max table row count에 테이블의 최소 및 최대 행 수를 입력하여 작은 테이블로 정의합니다. AWS SCT는

작은 테이블에 ALL 배포 스타일을 적용합니다. 이 경우 전체 테이블의 사본이 모든 노드에 배포됩니다.

- 전략 세부 정보를 구성합니다.

  각 최적화 전략의 가중치를 정의하는 것 외에 최적화 설정도 구성할 수 있습니다. 이 작업을 수행하려면 Conversion optimization을 선택합니다.

  - Sort key columns limit에 정렬 키의 최대 열 수를 입력합니다.
  - Skewed threshold value에 열에 대한 스큐된 값의 백분율(0~100)을 입력합니다. AWS SCT는 스큐 값이 임계값보다 큰 열을 배포 키의 후보 목록에서 제외합니다. AWS SCT는 열의 스큐된 값을 전체 레코드 수에 대한 가장 일반적인 값 발생 횟수의 백분율로 정의합니다.
  - Top N queries from the query history table에 분석할 가장 자주 사용되는 쿼리의 수(1~100)를 입력합니다.
  - Select statistics user에서 쿼리 통계를 분석하려는 데이터베이스 사용자를 선택합니다.

  또한 Optimization strategies 탭에서 Find small tables 전략을 위한 작은 테이블의 크기를 정의할 수 있습니다. Min table row count 및 Max table row count에 테이블의 최소 및 최대 행 수를 입력하여 작은 테이블로 간주합니다. AWS SCT는 작은 테이블에 ALL 배포 스타일을 적용합니다. 이 경우 전체 테이블의 사본이 모든 노드에 배포됩니다.

Snowflake를 AWS SCT에 대한 소스로 사용

AWS SCT를 사용하여 스키마, 코드 객체 및 애플리케이션 코드를 Snowflake에서 Amazon Redshift로 변환할 수 있습니다.

Snowflake를 소스 데이터베이스로 사용할 수 있는 권한

권한이 있는 역할을 생성하고 SECURITYADMIN 역할 및 SECURITYADMIN 세션 컨텍스트를 사용하여 이 역할에 사용자 이름을 부여할 수 있습니다.

다음 예제에서는 최소 권한을 생성하여 min_privs 사용자에게 부여합니다.

```
create role role_name;
grant role role_name to role sysadmin;
grant usage on database db_name to role role_name;
grant usage on schema db_name.schema_name to role role_name;
grant usage on warehouse datawarehouse_name to role role_name;
grant monitor on database db_name to role role_name;
grant monitor on warehouse datawarehouse_name to role role_name;
```

```
grant select on all tables in schema db_name.schema_name to role role_name;
grant select on future tables in schema db_name.schema_name to role role_name;
grant select on all views in schema db_name.schema_name to role role_name;
grant select on future views in schema db_name.schema_name to role role_name;
grant select on all external tables in schema db_name.schema_name to role role_name;
grant select on future external tables in schema db_name.schema_name to role role_name;
grant usage on all sequences in schema db_name.schema_name to role role_name;
grant usage on future sequences in schema db_name.schema_name to role role_name;
grant usage on all functions in schema db_name.schema_name to role role_name;
grant usage on future functions in schema db_name.schema_name to role role_name;
grant usage on all procedures in schema db_name.schema_name to role role_name;
grant usage on future procedures in schema db_name.schema_name to role role_name;
create user min_privs password='real_user_password'
DEFAULT_ROLE = role_name DEFAULT_WAREHOUSE = 'datawarehouse_name';
grant role role_name to user min_privs;
```

이전 예제에서 다음과 같이 자리 표시자를 바꿉니다.

- *role_name*을 읽기 전용 권한이 있는 역할의 이름으로 바꿉니다.
- *db_name*을 소스 데이터베이스의 이름으로 바꿉니다.
- *schema_name*을 소스 스키마의 이름으로 바꿉니다.
- *datawarehousename*을 필수 데이터 웨어하우스의 이름으로 바꿉니다.
- 최소 권한을 가진 사용자 이름으로 *min_privs*를 바꿉니다.

DEFAULT_ROLE 및 DEFAULT_WAREHOUSE 파라미터는 키를 구분합니다.

Amazon S3에 대한 보안 액세스 구성

Amazon S3 버킷의 보안 및 액세스 관리 정책을 통해 Snowflake는 S3 버킷에 액세스하고, S3 버킷에서 데이터를 읽고, S3 버킷에 데이터를 쓸 수 있습니다. Snowflake STORAGE INTEGRATION 객체 유형을 사용하여 프라이빗 Amazon S3 버킷에 대한 보안 액세스를 구성할 수 있습니다. Snowflake 스토리지 통합 객체는 Snowflake ID 및 액세스 관리 엔터티에 인증 책임을 위임합니다.

자세한 내용은 Snowflake 설명서에서 Amazon S3에 액세스하기 위한 Snowflake 스토리지 통합 구성을 참조하세요.

Snowflake에 소스로 연결

다음 절차에 따라 AWS Schema Conversion Tool을 사용하여 소스 데이터베이스에 연결합니다.

Snowflake 소스 데이터베이스에 연결하려면

1. AWS Schema Conversion Tool에서 소스 추가를 선택합니다.
2. Snowflake를 선택한 후 다음을 선택합니다.

   소스 추가 대화 상자가 나타납니다.
3. 연결 이름에 데이터베이스의 이름을 입력합니다. AWS SCT는 왼쪽 패널의 트리에 이 이름을 표시합니다.
4. AWS Secrets Manager의 데이터베이스 보안 인증 정보를 사용하거나 수동으로 입력합니다.

   - Secrets Manager의 데이터베이스 보안 인증 정보를 사용하려면 다음 지침을 따릅니다.
     1. AWS Secret에서 보안 암호의 이름을 선택합니다.
     2. Populate를 선택하여 Secrets Manager에서 데이터베이스 연결 대화 상자에 있는 모든 값을 자동으로 채웁니다.

     Secrets Manager의 데이터베이스 보안 인증 사용에 대한 자세한 내용은 AWS Secrets Manager 사용 섹션을 참조하세요.
   - Snowflake 소스 데이터 웨어하우스 연결 정보를 수동으로 입력하려면 다음 지침을 사용합니다.

파라미터	작업
[Server name]	소스 데이터베이스 서버의 도메인 이름 시스템(DNS) 이름 또는 IP 주소를 입력합니다.
[Server port]	소스 데이터베이스 서버에 연결하는 데 사용되는 포트를 입력합니다.
데이터베이스	Snowflake 데이터베이스의 이름을 입력합니다.
User name 및 Password	소스 데이터베이스 서버에 연결하는 데 필요한 데이터베이스 보안 인증 정보를 입력합니다. AWS SCT는 사용자가 명시적으로 요청하는 경우에만 암호를 암호화된 형식으로 저장합니다.
Use SSL(SSL 사용)	SSL(Secure Sockets Layer)을 사용하여 데이터베이스에 연결하려면 이 옵션을 선택합니다. SSL 탭에서 다음 추가 정보를 적절히 제공합니다.

파라미터	작업
	• 프라이빗 키 경로: 프라이빗 키의 위치입니다. • Passphrase: 프라이빗 키의 암호입니다. Snowflake의 SSL 지원에 대한 자세한 내용은 연결을 위한 보안 옵션 구성을 참조하세요.
Store password	AWS SCT는 SSL 인증서와 데이터베이스 암호를 저장할 안전한 볼트를 생성합니다. 이 옵션을 설정하면 데이터베이스 암호를 저장할 수 있습니다. 이렇게 하면 암호를 입력하지 않고도 데이터베이스에 빠르게 연결할 수 있습니다.
Snowflake driver path	소스 데이터베이스에 연결할 때 사용할 드라이버의 경로를 입력합니다. 자세한 내용은 필수 데이터베이스 드라이버 다운로드 섹션을 참조하세요. 드라이버 경로를 전역 프로젝트 설정에 저장할 경우 드라이버 경로가 연결 대화 상자에 표시되지 않습니다. 자세한 내용은 전역 설정에 드라이버 경로 저장 섹션을 참조하세요.

5. Test Connection을 선택하여 AWS SCT가 소스 데이터베이스에 연결할 수 있는지 확인합니다.
6. 연결을 선택하여 소스 데이터베이스에 연결합니다.

Snowflake를 소스로 사용 시 적용되는 제한 사항

AWS SCT에서 Snowflake를 소스로 사용하는 경우 제한 사항은 다음과 같습니다.

- 객체 식별자는 객체 유형과 상위 객체의 컨텍스트 내에서 고유해야 합니다.

  데이터베이스

  스키마 식별자는 데이터베이스 내에서 고유해야 합니다.

  스키마

  테이블 및 보기와 같은 객체 식별자는 스키마 내에서 고유해야 합니다.

  테이블/보기

  열 식별자는 테이블 내에서 고유해야 합니다.

- 대형(large) 및 배수(xlarge) 클러스터 노드 유형에 대한 최대 테이블 수는 9,900개입니다. 8xlarge 클러스터 노드 유형에 대한 최대 테이블 수는 100,000개입니다. 이 제한에는 임시 테이블이 포함되며, 모두 쿼리 처리 또는 시스템 유지 관리 중에 Amazon Redshift에서 사용자 정의하거나 생성할 수 있습니다. 자세한 내용은 Amazon Redshift 클러스터 관리 안내서의 Amazon Redshift 할당량 섹션을 참조하세요.
- 저장 프로시저의 경우 입력 및 출력 인수의 최대 개수는 32개입니다.

Snowflake의 소스 데이터 형식

다음에서는 AWS SCT를 사용할 때 지원되는 Snowflake 소스 데이터 형식과 Amazon Redshift 대상에 대한 기본 매핑을 확인할 수 있습니다.

Snowflake 데이터 형식	Amazon Redshift 데이터 형식
NUMBER	NUMERIC(38)
NUMBER(p)	If p is =< 4, then SMALLINT If p is => 5 and =< 9, then INTEGER If p is => 10 and =< 18, then BIGINT If p is => 19 then NUMERIC(p)
NUMBER(p, 0)	If p is =< 4, then SMALLINT If p is => 5 and =< 9, then INTEGER If p is => 10 and =< 18, then BIGINT If p is => 19 then: NUMERIC(p,0)
NUMBER(p, s)	If p is => 1 and =< 38, and if s is => 1 and =< 37, then NUMERIC(p,s)
FLOAT	FLOAT
TEXT	VARCHAR(MAX)

Snowflake 데이터 형식	Amazon Redshift 데이터 형식
최대 16,777,216바이트의 유니코드 문자, 문자당 최대 4바이트	
TEXT(p) 최대 65,535바이트의 유니코드 문자, 문자당 최대 4바이트	If p is =< 65,535 then, VARCHAR(p)
TEXT(p) 최대 16,777,216바이트의 유니코드 문자, 문자당 최대 4바이트	If p is => 65,535 and =< 16,777,216 then, VARCHAR(MAX)
BINARY 최대 8,388,608바이트의 싱글 바이트 문자, 문자당 1바이트	VARCHAR(MAX)
BINARY(p) 최대 65,535바이트의 싱글 바이트 문자, 문자당 1바이트	VARCHAR(p)
BINARY(p) 최대 8,388,608바이트의 싱글 바이트 문자, 문자당 1바이트	VARCHAR(MAX)
BOOLEAN	BOOLEAN
DATE	DATE
TIME 00:00:00에서 23:59:59.999999999 사이의 시간 값	VARCHAR(18)

Snowflake 데이터 형식	Amazon Redshift 데이터 형식
TIME(f) 00:00:00에서 23:59:59.9(f) 사이의 시간 값	VARCHAR(n) – 9 + dt-attr-1
TIMESTAMP_NTZ	TIMESTAMP
TIMESTAMP_TZ	TIMESTAMPTZ

Snowflake에서 Amazon Redshift로의 변환 설정

Snowflake에서 Amazon Redshift로의 변환 설정을 편집하려면 AWS SCT에서 설정을 선택한 다음 변환 설정을 선택합니다. 상단 목록에서 Snowflake를 선택한 다음 Snowflake – Amazon Redshift를 선택합니다. AWS SCT는 Snowflake에서 Amazon Redshift로의 변환에 사용할 수 있는 모든 설정을 표시합니다.

AWS SCT에서 Snowflake를 Amazon Redshift로 변환하는 설정에는 다음과 같은 옵션이 포함됩니다.

- 변환된 코드에서 작업 항목이 포함된 설명의 수를 제한합니다.

  Add comments in the converted code for the action items of selected severity and higher에서 작업 항목의 심각도를 선택합니다. AWS SCT가 선택된 심각도 이상의 작업 항목에 대한 변환된 코드에 설명을 추가합니다.

  예를 들어, 변환된 코드의 설명 수를 최소화하려면 오류만을 선택합니다. 변환된 코드의 모든 작업 항목에 대한 설명을 포함하려면 모든 메시지를 선택합니다.
- AWS SCT가 대상 Amazon Redshift 클러스터에 적용할 수 있는 최대 테이블 수를 설정합니다.

  The maximum number of tables for the target Amazon Redshift cluster에서 AWS SCT가 Amazon Redshift 클러스터에 적용할 수 있는 테이블 수를 선택합니다.

  Amazon Redshift에는 여러 클러스터 노드 유형에 사용하는 테이블을 제한하는 할당량이 있습니다. 자동을 선택하면 AWS SCT가 노드 유형에 따라 대상 Amazon Redshift 클러스터에 적용되는 테이블 수를 결정합니다. 값을 수동으로 선택할 수도 있습니다. 자세한 내용은 Amazon Redshift 관리 가이드의 Amazon Redshift의 할당량 및 제한 섹션을 참조하세요.

  Amazon Redshift 클러스터가 저장할 수 있는 수보다 이 숫자가 많아도 AWS SCT는 모든 소스 테이블을 변환합니다. AWS SCT는 변환된 코드를 프로젝트에 저장하며 대상 데이터베이스에는 적용

하지 않습니다. 변환된 코드를 적용할 때 테이블의 Amazon Redshift 클러스터 할당량에 도달하면 AWS SCT에서 경고 메시지가 표시됩니다. 또한 AWS SCT는 테이블 수가 한도에 도달할 때까지 대상 Amazon Redshift 클러스터에 테이블을 적용합니다.

- Amazon Redshift 테이블 열에 압축을 적용합니다. 이렇게 하려면 Use compression encoding을 선택합니다.

  AWS SCT는 기본 Amazon Redshift 알고리즘을 사용하여 열에 압축 인코딩을 자동으로 할당합니다. 자세한 내용은 Amazon Redshift 데이터베이스 개발자 안내서의 압축 인코딩을 참조하세요.

  기본적으로 Amazon Redshift는 정렬 및 배포 키로 정의된 열에 압축을 적용하지 않습니다. 이 동작을 변경하여 이러한 열에 압축을 적용할 수 있습니다. 이렇게 하려면 Use compression encoding for KEY columns를 선택합니다. Use compression encoding 옵션을 선택한 경우에만 이 옵션을 선택할 수 있습니다.

Snowflake에서 Amazon Redshift로의 변환 최적화 설정

Snowflake에서 Amazon Redshift로의 변환 최적화 설정을 편집하려면 AWS SCT에서 설정을 선택한 다음 변환 설정을 선택합니다. 상단 목록에서 Snowflake를 선택한 다음 Snowflake – Amazon Redshift를 선택합니다. 왼쪽 창에서 Optimization strategies를 선택합니다. AWS SCT는 Snowflake에서 Amazon Redshift로의 변환에 대한 변환 최적화 설정을 표시합니다.

AWS SCT에서 Snowflake를 Amazon Redshift로 변환하는 변환 최적화 설정에는 다음과 같은 옵션이 포함됩니다.

- 자동 테이블 최적화 작업을 수행합니다. 이 작업을 수행하려면 Use Amazon Redshift automatic table tuning을 선택합니다.

  자동 테이블 최적화는 테이블 디자인을 자동으로 최적화하는 Amazon Redshift의 자체 조정 프로세스입니다. 자세한 내용은 Amazon Redshift 데이터베이스 개발자 안내서의 자동 테이블 최적화 작업을 참조하세요.

  자동 테이블 최적화만 사용하려면 Initial key selection strategy에서 없음을 선택합니다.
- 전략을 사용하여 정렬 및 배포 키를 선택합니다.

  Amazon Redshift 메타데이터, 통계 정보 또는 두 옵션을 모두 사용하여 정렬 및 배포 키를 선택할 수 있습니다. Optimization strategies 탭의 Initial key selection strategy에서 다음 옵션 중 하나를 선택합니다.

  - 메타데이터 사용, 통계 정보 무시

- 메타데이터 무시, 통계 정보 사용
- 메타데이터 및 통계 정보 사용

선택한 옵션에 따라 최적화 전략을 선택할 수 있습니다. 그런 다음 각 전략에 대해 값(0~100)을 입력합니다. 이러한 값은 각 전략의 가중치를 정의합니다. AWS SCT는 이러한 가중치 값을 사용하여 각 규칙이 배포 및 정렬 키 선택에 미치는 영향을 정의합니다. 기본값은 AWS 마이그레이션 모범 사례를 기반으로 합니다.

Find small tables 전략에서 작은 테이블의 크기를 정의할 수 있습니다. Min table row count 및 Max table row count에 테이블의 최소 및 최대 행 수를 입력하여 작은 테이블로 정의합니다. AWS SCT는 작은 테이블에 ALL 배포 스타일을 적용합니다. 이 경우 전체 테이블의 사본이 모든 노드에 배포됩니다.

- 전략 세부 정보를 구성합니다.

  각 최적화 전략의 가중치를 정의하는 것 외에 최적화 설정도 구성할 수 있습니다. 이 작업을 수행하려면 Conversion optimization을 선택합니다.

  - Sort key columns limit에 정렬 키의 최대 열 수를 입력합니다.
  - Skewed threshold value에 열에 대한 스큐된 값의 백분율(0~100)을 입력합니다. AWS SCT는 스큐 값이 임계값보다 큰 열을 배포 키의 후보 목록에서 제외합니다. AWS SCT는 열의 스큐된 값을 전체 레코드 수에 대한 가장 일반적인 값 발생 횟수의 백분율로 정의합니다.
  - Top N queries from the query history table에 분석할 가장 자주 사용되는 쿼리의 수(1~100)를 입력합니다.
  - Select statistics user에서 쿼리 통계를 분석하려는 데이터베이스 사용자를 선택합니다.

  또한 Optimization strategies 탭에서 Find small tables 전략을 위한 작은 테이블의 크기를 정의할 수 있습니다. Min table row count 및 Max table row count에 테이블의 최소 및 최대 행 수를 입력하여 작은 테이블로 간주합니다. AWS SCT는 작은 테이블에 ALL 배포 스타일을 적용합니다. 이 경우 전체 테이블의 사본이 모든 노드에 배포됩니다.

Microsoft SQL Server Data Warehouse를 AWS SCT에 대한 소스로 사용

AWS SCT를 사용하면 스키마, 코드 객체 및 애플리케이션 코드를 Microsoft SQL Server DW에서 Amazon Redshift로 변환하거나 조합하여 사용되는 Amazon Redshift 및 AWS Glue로 변환할 수 있습니다.

Microsoft SQL Server Data Warehouse를 소스로 사용할 수 있는 권한

Microsoft SQL Server 데이터 웨어하우스를 소스로 사용하기 위해 필요한 권한은 다음과 같습니다.

- VIEW DEFINITION
- VIEW DATABASE STATE
- SELECT ON SCHEMA :: *<schema_name>*

이전 예제에서 *<source_schema>* 자리 표시자를 소스 source_schema의 이름으로 바꿉니다.

변환하려는 스키마의 각 데이터베이스에 대해 권한 부여를 반복합니다.

또한 다음 권한을 부여하고 마스터 데이터베이스에서 권한 부여를 실행합니다.

- VIEW SERVER STATE

SQL Server Data Warehouse를 소스로 사용 시 적용되는 제한 사항

Microsoft SQL Server Parallel Data Warehouse(PDW)를 소스로 사용하는 것은 현재 지원되지 않습니다.

SQL Server Data Warehouse에 소스로 연결

다음 절차에 따라 AWS Schema Conversion Tool을 사용하여 SQL Server Data Warehouse 소스 데이터베이스에 연결합니다.

SQL Server Data Warehouse 소스 데이터베이스에 연결하려면

1. AWS Schema Conversion Tool에서 소스 추가를 선택합니다.
2. Microsoft SQL Server를 선택한 후 다음을 선택합니다.

   소스 추가 대화 상자가 나타납니다.
3. 연결 이름에 데이터베이스의 이름을 입력합니다. AWS SCT는 왼쪽 패널의 트리에 이 이름을 표시합니다.
4. AWS Secrets Manager의 데이터베이스 보안 인증 정보를 사용하거나 수동으로 입력합니다.

   - Secrets Manager의 데이터베이스 보안 인증 정보를 사용하려면 다음 지침을 따릅니다.
     1. AWS Secret에서 보안 암호의 이름을 선택합니다.

2. Populate를 선택하여 Secrets Manager에서 데이터베이스 연결 대화 상자에 있는 모든 값을 자동으로 채웁니다.

Secrets Manager의 데이터베이스 보안 인증 사용에 대한 자세한 내용은 AWS Secrets Manager 사용 섹션을 참조하세요.

- Microsoft SQL Server 소스 데이터 웨어하우스 연결 정보를 수동으로 입력하려면 다음 지침을 사용합니다.

파라미터	작업
[Server name]	소스 데이터베이스 서버의 DNS(Domain Name Service) 이름 또는 IP 주소를 입력합니다.
[Server port]	소스 데이터베이스 서버에 연결하는 데 사용되는 포트를 입력합니다.
인스턴스 이름	SQL Server 데이터 웨어하우스의 인스턴스 이름을 입력합니다.
User name 및 Password	소스 데이터베이스 서버에 연결하는 데 필요한 데이터베이스 보안 인증 정보를 입력합니다. 프로젝트에서 데이터베이스에 연결하도록 선택한 경우에만 AWS SCT에서 암호를 사용하여 소스 데이터베이스에 연결합니다. 소스 데이터베이스의 암호가 노출될 위험을 방지하기 위해 AWS SCT는 기본적으로 암호를 저장하지 않습니다. AWS SCT 프로젝트를 닫았다 다시 열 경우 필요에 따라 소스 데이터베이스에 연결하기 위한 암호를 입력하라는 메시지가 표시됩니다.
Use SSL(SSL 사용)	SSL(Secure Sockets Layer)을 사용하여 데이터베이스에 연결하려면 이 옵션을 선택합니다. SSL 탭에서 다음 추가 정보를 적절히 제공합니다. • Trust server certificate: 서버 인증서를 신뢰하려면 이 옵션을 선택합니다. • 트러스트 스토어: 전역 설정에서 설정한 트러스트 스토어입니다.

파라미터	작업
Store password	AWS SCT는 SSL 인증서와 데이터베이스 암호를 저장할 안전한 볼트를 생성합니다. 이 옵션을 켜면 데이터베이스 암호를 저장하고 암호 입력 없이 빠르게 데이터베이스에 연결할 수 있습니다.
SQL Server driver path	소스 데이터베이스에 연결할 때 사용할 드라이버의 경로를 입력합니다. 자세한 내용은 필수 데이터베이스 드라이버 다운로드 섹션을 참조하세요. 드라이버 경로를 전역 프로젝트 설정에 저장할 경우 드라이버 경로가 연결 대화 상자에 표시되지 않습니다. 자세한 내용은 전역 설정에 드라이버 경로 저장 섹션을 참조하세요.

5. Test Connection을 선택하여 AWS SCT가 소스 데이터베이스에 연결할 수 있는지 확인합니다.
6. 연결을 선택하여 소스 데이터베이스에 연결합니다.

SQL Server Data Warehouse에서 Amazon Redshift로의 변환 설정

SQL Server Data Warehouse에서 Amazon Redshift로의 변환 설정을 편집하려면 AWS SCT에서 설정을 선택한 다음 변환 설정을 선택합니다. 상단 목록에서 Microsoft SQL Server를 선택한 다음 Microsoft SQL Server – Amazon Redshift를 선택합니다. AWS SCT는 SQL Server Data Warehouse에서 Amazon Redshift로의 변환에 사용할 수 있는 모든 설정을 표시합니다.

AWS SCT에서 SQL Server Data Warehouse를 Amazon Redshift로 변환하는 설정에는 다음과 같은 옵션이 포함됩니다.

- 변환된 코드에서 작업 항목이 포함된 설명의 수를 제한합니다.

  Add comments in the converted code for the action items of selected severity and higher에서 작업 항목의 심각도를 선택합니다. AWS SCT가 선택된 심각도 이상의 작업 항목에 대한 변환된 코드에 설명을 추가합니다.

  예를 들어, 변환된 코드의 설명 수를 최소화하려면 오류만을 선택합니다. 변환된 코드의 모든 작업 항목에 대한 설명을 포함하려면 모든 메시지를 선택합니다.
- AWS SCT가 대상 Amazon Redshift 클러스터에 적용할 수 있는 최대 테이블 수를 설정합니다.

The maximum number of tables for the target Amazon Redshift cluster에서 AWS SCT가 Amazon Redshift 클러스터에 적용할 수 있는 테이블 수를 선택합니다.

Amazon Redshift에는 여러 클러스터 노드 유형에 사용하는 테이블을 제한하는 할당량이 있습니다. 자동을 선택하면 AWS SCT가 노드 유형에 따라 대상 Amazon Redshift 클러스터에 적용되는 테이블 수를 결정합니다. 값을 수동으로 선택할 수도 있습니다. 자세한 내용은 Amazon Redshift 관리 가이드의 Amazon Redshift의 할당량 및 제한 섹션을 참조하세요.

Amazon Redshift 클러스터가 저장할 수 있는 수보다 이 숫자가 많아도 AWS SCT는 모든 소스 테이블을 변환합니다. AWS SCT는 변환된 코드를 프로젝트에 저장하며 대상 데이터베이스에는 적용하지 않습니다. 변환된 코드를 적용할 때 테이블의 Amazon Redshift 클러스터 할당량에 도달하면 AWS SCT에서 경고 메시지가 표시됩니다. 또한 AWS SCT는 테이블 수가 한도에 도달할 때까지 대상 Amazon Redshift 클러스터에 테이블을 적용합니다.

- 소스 테이블의 파티션을 Amazon Redshift의 개별 테이블로 마이그레이션합니다. 이 작업을 수행하려면 Use the UNION ALL view를 선택하고 AWS SCT가 단일 소스 테이블에 대해 생성할 수 있는 대상 테이블의 최대 수를 입력합니다.

  Amazon Redshift는 테이블 파티셔닝을 지원하지 않습니다. 이 동작을 에뮬레이션하고 쿼리를 더 빠르게 실행하기 위해 AWS SCT는 소스 테이블의 각 파티션을 Amazon Redshift의 개별 테이블로 마이그레이션할 수 있습니다. 그런 다음 AWS SCT는 이러한 모든 테이블의 데이터를 포함하는 보기를 생성합니다.

  AWS SCT는 소스 테이블의 파티션 수를 자동으로 결정합니다. 소스 테이블 파티셔닝 유형에 따라서는 이 숫자가 Amazon Redshift 클러스터에 적용할 수 있는 테이블의 할당량을 초과할 수 있습니다. 이 할당량에 도달하지 않도록 하려면 AWS SCT가 단일 소스 테이블의 파티션에 대해 생성할 수 있는 최대 대상 테이블 수를 입력합니다. 기본 옵션은 368개 테이블이며, 이는 1년 중 366일 동안의 파티션과 `NO  RANGE` 및 `UNKNOWN` 파티션에 대한 테이블 두 개를 나타냅니다.

- Amazon Redshift 테이블 열에 압축을 적용합니다. 이렇게 하려면 Use compression encoding을 선택합니다.

  AWS SCT는 기본 Amazon Redshift 알고리즘을 사용하여 열에 압축 인코딩을 자동으로 할당합니다. 자세한 내용은 Amazon Redshift 데이터베이스 개발자 안내서의 압축 인코딩을 참조하세요.

  기본적으로 Amazon Redshift는 정렬 및 배포 키로 정의된 열에 압축을 적용하지 않습니다. 이 동작을 변경하여 이러한 열에 압축을 적용할 수 있습니다. 이렇게 하려면 Use compression encoding for KEY columns를 선택합니다. Use compression encoding 옵션을 선택한 경우에만 이 옵션을 선택할 수 있습니다.

SQL Server Data Warehouse에서 Amazon Redshift로의 변환 최적화 설정

SQL Server Data Warehouse에서 Amazon Redshift로의 변환 최적화 설정을 편집하려면 AWS SCT에서 설정을 선택한 다음 변환 설정을 선택합니다. 상단 목록에서 Microsoft SQL Server를 선택한 다음 Microsoft SQL Server – Amazon Redshift를 선택합니다. 왼쪽 창에서 Optimization strategies를 선택합니다. AWS SCT는 SQL Server Data Warehouse에서 Amazon Redshift로의 변환에 대한 변환 최적화 설정을 표시합니다.

AWS SCT에서 SQL Server Data Warehouse를 Amazon Redshift로 변환하는 변환 최적화 설정에는 다음과 같은 옵션이 포함됩니다.

- 자동 테이블 최적화 작업을 수행합니다. 이 작업을 수행하려면 Use Amazon Redshift automatic table tuning을 선택합니다.

  자동 테이블 최적화는 테이블 디자인을 자동으로 최적화하는 Amazon Redshift의 자체 조정 프로세스입니다. 자세한 내용은 Amazon Redshift 데이터베이스 개발자 안내서의 자동 테이블 최적화 작업을 참조하세요.

  자동 테이블 최적화만 사용하려면 Initial key selection strategy에서 없음을 선택합니다.
- 전략을 사용하여 정렬 및 배포 키를 선택합니다.

  Amazon Redshift 메타데이터, 통계 정보 또는 두 옵션을 모두 사용하여 정렬 및 배포 키를 선택할 수 있습니다. Optimization strategies 탭의 Initial key selection strategy에서 다음 옵션 중 하나를 선택합니다.
  - 메타데이터 사용, 통계 정보 무시
  - 메타데이터 무시, 통계 정보 사용
  - 메타데이터 및 통계 정보 사용

  선택한 옵션에 따라 최적화 전략을 선택할 수 있습니다. 그런 다음 각 전략에 대해 값(0~100)을 입력합니다. 이러한 값은 각 전략의 가중치를 정의합니다. AWS SCT는 이러한 가중치 값을 사용하여 각 규칙이 배포 및 정렬 키 선택에 미치는 영향을 정의합니다. 기본값은 AWS 마이그레이션 모범 사례를 기반으로 합니다.

  Find small tables 전략에서 작은 테이블의 크기를 정의할 수 있습니다. Min table row count 및 Max table row count에 테이블의 최소 및 최대 행 수를 입력하여 작은 테이블로 정의합니다. AWS SCT는 작은 테이블에 ALL 배포 스타일을 적용합니다. 이 경우 전체 테이블의 사본이 모든 노드에 배포됩니다.
- 전략 세부 정보를 구성합니다.

각 최적화 전략의 가중치를 정의하는 것 외에 최적화 설정도 구성할 수 있습니다. 이 작업을 수행하려면 Conversion optimization을 선택합니다.

- Sort key columns limit에 정렬 키의 최대 열 수를 입력합니다.
- Skewed threshold value에 열에 대한 스큐된 값의 백분율(0~100)을 입력합니다. AWS SCT는 스큐 값이 임계값보다 큰 열을 배포 키의 후보 목록에서 제외합니다. AWS SCT는 열의 스큐된 값을 전체 레코드 수에 대한 가장 일반적인 값 발생 횟수의 백분율로 정의합니다.
- Top N queries from the query history table에 분석할 가장 자주 사용되는 쿼리의 수(1~100)를 입력합니다.
- Select statistics user에서 쿼리 통계를 분석하려는 데이터베이스 사용자를 선택합니다.

또한 Optimization strategies 탭에서 Find small tables 전략을 위한 작은 테이블의 크기를 정의할 수 있습니다. Min table row count 및 Max table row count에 테이블의 최소 및 최대 행 수를 입력하여 작은 테이블로 간주합니다. AWS SCT는 작은 테이블에 `ALL` 배포 스타일을 적용합니다. 이 경우 전체 테이블의 사본이 모든 노드에 배포됩니다.

Teradata를 AWS SCT에 대한 소스로 사용

AWS SCT를 사용하면 스키마, 코드 객체 및 애플리케이션 코드를 Teradata에서 Amazon Redshift로 변환하거나 조합하여 사용되는 Amazon Redshift 및 AWS Glue로 변환할 수 있습니다.

Teradata를 소스로 사용할 수 있는 권한

Teradata를 소스로 사용하기 위해 필요한 권한은 다음과 같습니다.

- SELECT ON DBC
- SELECT ON SYSUDTLIB
- SELECT ON SYSLIB
- SELECT ON *<source_database>*
- CREATE PROCEDURE ON *<source_database>*

이전 예제에서 *<source_database>* 자리 표시자를 소스 데이터베이스의 이름으로 바꿉니다.

AWS SCT가 소스 데이터베이스의 모든 프로시저에 대해 HELP PROCEDURE를 수행하려면 CREATE PROCEDURE 권한이 필요합니다. AWS SCT는 소스 Teradata 데이터베이스에 새 개체를 생성하는 데 이 권한을 사용하지 않습니다.

Teradata에 소스로 연결

다음 절차에 따라 AWS Schema Conversion Tool을 사용하여 Teradata 소스 데이터베이스에 연결합니다.

Teradata 소스 데이터베이스에 연결하려면

1. AWS Schema Conversion Tool에서 소스 추가를 선택합니다.
2. Teradata를 선택한 후 다음을 선택합니다.

   소스 추가 대화 상자가 나타납니다.
3. 연결 이름에 데이터베이스의 이름을 입력합니다. AWS SCT는 왼쪽 패널의 트리에 이 이름을 표시합니다.
4. AWS Secrets Manager의 데이터베이스 보안 인증 정보를 사용하거나 수동으로 입력합니다.

   - Secrets Manager의 데이터베이스 보안 인증 정보를 사용하려면 다음 지침을 따릅니다.

     1. AWS Secret에서 보안 암호의 이름을 선택합니다.
     2. Populate를 선택하여 Secrets Manager에서 데이터베이스 연결 대화 상자에 있는 모든 값을 자동으로 채웁니다.

     Secrets Manager의 데이터베이스 보안 인증 사용에 대한 자세한 내용은 AWS Secrets Manager 사용 섹션을 참조하세요.
   - Teradata 소스 데이터베이스 연결 정보를 수동으로 입력하려면 다음 지침을 사용합니다.

파라미터	작업
연결 이름	데이터베이스의 이름을 입력합니다. AWS SCT는 왼쪽 패널의 트리에 이 이름을 표시합니다.
[Server name]	소스 데이터베이스 서버의 도메인 이름 시스템(DNS) 이름 또는 IP 주소를 입력합니다.
[Server port]	소스 데이터베이스 서버에 연결하는 데 사용되는 포트를 입력합니다.
데이터베이스	Teradata 데이터베이스의 이름을 입력합니다.

파라미터	작업
User name 및 Password	소스 데이터베이스 서버에 연결하는 데 필요한 데이터베이스 보안 인증 정보를 입력합니다. 프로젝트에서 데이터베이스에 연결하도록 선택한 경우에만 AWS SCT에서 암호를 사용하여 소스 데이터베이스에 연결합니다. 소스 데이터베이스의 암호가 노출될 위험을 방지하기 위해 AWS SCT는 기본적으로 암호를 저장하지 않습니다. AWS SCT 프로젝트를 닫았다 다시 열 경우 필요에 따라 소스 데이터베이스에 연결하기 위한 암호를 입력하라는 메시지가 표시됩니다.
Store password	AWS SCT는 SSL 인증서와 데이터베이스 암호를 저장할 안전한 볼트를 생성합니다. 이 옵션을 켜면 데이터베이스 암호를 저장하고 암호 입력 없이 빠르게 데이터베이스에 연결할 수 있습니다.
Encrypt data	데이터베이스와 교환하는 데이터를 암호화하려면 이 옵션을 선택합니다. 이 옵션을 선택하면 포트 번호 443을 사용하여 AWS SCT와 Teradata 데이터베이스 간에 암호화된 데이터를 전송합니다.
Teradata driver path	소스 데이터베이스에 연결할 때 사용할 드라이버의 경로를 입력합니다. 자세한 내용은 필수 데이터베이스 드라이버 다운로드 섹션을 참조하세요. 드라이버 경로를 전역 프로젝트 설정에 저장할 경우 드라이버 경로가 연결 대화 상자에 표시되지 않습니다. 자세한 내용은 전역 설정에 드라이버 경로 저장 섹션을 참조하세요.

5. Test Connection을 선택하여 AWS SCT가 소스 데이터베이스에 연결할 수 있는지 확인합니다.
6. 연결을 선택하여 소스 데이터베이스에 연결합니다.

Teradata 소스에서 LDAP 인증 사용

Windows에서 Microsoft Active Directory를 실행하는 Teradata 사용자를 위한 Lightweight Directory Access Protocol(LDAP) 인증을 설정하려면 다음 절차를 사용합니다.

다음 절차에서 Active Directory 도메인은 `test.local.com`입니다. Windows 서버는 `DC`이며 기본 설정으로 구성되어 있습니다. 다음 스크립트는 `test_ldap` Active Directory 계정을 생성하며 이 계정은 `test_ldap` 암호를 사용합니다.

Windows에서 Microsoft Active Directory를 실행하는 Teradata 사용자에 대한 LDAP 인증을 설정하려면

1. `/opt/teradata/tdat/tdgss/site` 디렉터리에서 `TdgssUserConfigFile.xml` 파일을 편집합니다. LDAP 섹션을 다음과 같이 변경합니다.

```
AuthorizationSupported="no"

LdapServerName="DC.test.local.com"
LdapServerPort="389"
LdapServerRealm="test.local.com"
LdapSystemFQDN="dc= test, dc= local, dc=com"
LdapBaseFQDN="dc=test, dc=local, dc=com"
```

2. 다음과 같이 구성을 실행하여 변경 사항을 적용합니다.

```
#cd /opt/teradata/tdgss/bin
#./run_tdgssconfig
```

3. 다음 명령을 실행하여 구성을 테스트합니다.

```
# /opt/teradata/tdat/tdgss/14.10.03.01/bin/tdsbind -u test_ldap -w test_ldap
```

   다음과 같이 출력됩니다

```
LdapGroupBaseFQDN: dc=Test, dc=local, dc=com
LdapUserBaseFQDN: dc=Test, dc=local, dc=com
LdapSystemFQDN: dc= test, dc= local, dc=com
LdapServerName: DC.test.local.com
LdapServerPort: 389
LdapServerRealm: test.local.com
LdapClientUseTls: no
LdapClientTlsReqCert: never
LdapClientMechanism: SASL/DIGEST-MD5
LdapServiceBindRequired: no
LdapClientTlsCRLCheck: none
LdapAllowUnsafeServerConnect: yes
```

```
UseLdapConfig: no
AuthorizationSupported: no
FQDN: CN=test, CN=Users, DC=Anthem, DC=local, DC=com
AuthUser: ldap://DC.test.local.com:389/CN=test1,CN=Users,DC=test,DC=local,DC=com
DatabaseName: test
Service: tdsbind
```

4. 다음 명령을 사용하여 TPA를 다시 시작합니다.

```
#tpareset -f "use updated TDGSSCONFIG GDO"
```

5. 다음과 같이 Active Directory와 동일한 사용자를 Teradata 데이터베이스에서 생성합니다.

```
CREATE USER test_ldap AS PERM=1000, PASSWORD=test_ldap;
GRANT LOGON ON ALL TO test WITH NULL PASSWORD;
```

Active Directory에서 LDAP 사용자의 사용자 암호를 변경하는 경우 LDAP 모드에서 Teradata에 연결하는 동안 이 새 암호를 지정합니다. DEFAULT 모드에서 LDAP 사용자 이름과 암호를 사용하여 Teradata에 연결합니다.

소스 Teradata 데이터 웨어하우스에서 통계 수집 구성

소스 Teradata 데이터 웨어하우스를 변환하려면 AWS SCT에서 통계를 사용하여 변환된 Amazon Redshift 데이터 웨어하우스를 최적화합니다. AWS SCT에서 통계를 수집하거나 통계 파일을 업로드할 수 있습니다. 자세한 내용은 통계 수집 또는 업로드 섹션을 참조하세요.

AWS SCT가 데이터 웨어하우스에서 통계를 수집할 수 있도록 하려면 다음 사전 작업을 완료합니다.

Teradata 데이터 웨어하우스에서 통계를 수집하려면

1. 다음 쿼리를 실행하여 데이터 웨어하우스의 모든 테이블에 대한 통계를 다시 수집합니다.

```
collect summary statistics on table_name;
```

   이전 예제에서 *table_name*을 소스 테이블의 이름으로 바꿉니다. 변환하는 각 테이블에 대해 쿼리를 반복합니다.

2. 다음 쿼리를 실행하여 사용자가 데이터 웨어하우스를 변환하는 데 사용할 계정 문자열을 결정합니다.

```
select * from dbc.accountinfo where username ='user_name'
```

3. 이전 예제의 계정 문자열을 사용하여 특정 사용자에 대한 쿼리 로깅을 켭니다.

```
BEGIN QUERY LOGGING WITH OBJECTS, SQL ON ALL ACCOUNT=('$M$BUSI$S$D$H');
```

또는 모든 데이터베이스 사용자에 대해 쿼리 로깅을 켤 수도 있습니다.

```
BEGIN QUERY LOGGING WITH SQL, OBJECTS LIMIT SQLTEXT=0 ON ALL;
```

데이터 웨어하우스 통계 수집을 완료한 후에는 쿼리 로깅을 끕니다. 다음 코드 예제를 사용하여 이 작업을 수행할 수 있습니다.

```
end query logging with explain, objects, sql on all account=(' $M$BUSI$S$D$H');
```

소스 Teradata 데이터 웨어하우스에서 오프라인 모드로 통계 수집

Teradata 데이터 웨어하우스에서 통계 수집을 구성한 후 AWS SCT 프로젝트에서 통계를 수집할 수 있습니다. 또는 Basic Teradata Query(BTEQ) 스크립트를 사용하여 오프라인 모드로 통계를 수집할 수 있습니다. 그런 다음 수집된 통계가 포함된 파일을 AWS SCT 프로젝트에 업로드할 수 있습니다. 자세한 내용은 통계 수집 또는 업로드 섹션을 참조하세요.

오프라인 모드로 Teradata 데이터 웨어하우스에서 통계를 수집하려면

1. 다음 콘텐츠가 포함된 `off-line_stats.bteq` 스크립트를 생성합니다.

```
.OS IF EXIST column-stats-tera.csv del /F column-stats-tera.csv
.OS IF EXIST table-stats-tera.csv del /F table-stats-tera.csv
.OS IF EXIST column-skew-script-tera.csv del /F column-skew-script-tera.csv
.OS IF EXIST column-skew-stats-tera.csv del /F column-skew-stats-tera.csv
.OS IF EXIST query-stats-tera.csv  del /F query-stats-tera.csv
.LOGON your_teradata_server/your_login, your_password
.EXPORT REPORT FILE = table-stats-tera.csv
.SET TITLEDASHES OFF
.SET WIDTH 10000

SELECT
    '"' || OREPLACE(COALESCE(c.DatabaseName, ''), '"', '""') || '";' ||
    '"' || OREPLACE(COALESCE(c.TableName, ''), '"', '""') || '";' ||
```

```
    '"' || TRIM(COALESCE(s.reference_count, '0')) || '";' ||
    '"' || TRIM(COALESCE(CAST(p.RowCount AS BIGINT), '0')) || '";' ||
    '"' || CAST(CAST(w.size_in_mb AS DECIMAL (38,1) FORMAT 'Z9.9') AS VARCHAR(38))
 || '";' ||
    '"' || TRIM(COALESCE(r.stat_fk_dep_count, '0')) || '";' ||
    '"' || CAST(CAST(current_timestamp(0) as timestamp(0) format 'YYYY-MM-
DDBHH:MI:SS') as VARCHAR(19)) || '"'
(TITLE
 '"database_name";"table_name";"reference_count";"row_count";"size_in_mb";"stat_fk_dep_coun
FROM (select databasename, tablename
        from DBC.tablesv
        where tablekind IN ('T','O')
        and databasename = 'your_database_name'
         ) c
left join
        (select DatabaseName, TableName, max(RowCount) RowCount
        from dbc.tableStatsv
        group by 1,2)p
on p.databasename = c.databasename
and p.tablename = c.tablename
left join
        (SELECT r.ChildDB as DatabaseName,
        r.ChildTable as TableName,
        COUNT(DISTINCT r.ParentTable) reference_count
        FROM DBC.All_RI_ChildrenV r
        GROUP BY r.ChildDB, r.ChildTable) s
on s.databasename = c.databasename
and s.tablename = c.tablename
left join
        (SELECT r.ParentDB as DatabaseName,
        r.ParentTable as TableName,
        COUNT(DISTINCT r.ChildTable) stat_fk_dep_count
        FROM DBC.All_RI_ParentsV r
        GROUP BY r.ParentDB, r.ParentTable) r
on r.databasename = c.databasename
and r.tablename = c.tablename
left join
        (select databasename, tablename,
        sum(currentperm)/1024/1024 as size_in_mb
        from dbc.TableSizeV
        group by 1,2) w
on w.databasename = c.databasename
and w.tablename = c.tablename
```

```
WHERE COALESCE(r.stat_fk_dep_count,0) + COALESCE(CAST(p.RowCount AS BIGINT),0) +
 COALESCE(s.reference_count,0) > 0;

.EXPORT RESET

.EXPORT REPORT FILE = column-stats-tera.csv
.SET TITLEDASHES OFF
.SET WIDTH 10000
    '"' || TRIM(COALESCE(CAST(t2.card AS BIGINT), '0')) || '";' ||

SELECT
 '"' || OREPLACE(COALESCE(trim(tv.DatabaseName), ''), '"', '""') || '";' ||
     '"' || OREPLACE(COALESCE(trim(tv.TableName), ''), '"', '""') || '";' ||
 '"' || OREPLACE(COALESCE(trim(tv.columnname), ''), '"', '""') || '";' ||
                         '"' || TRIM(COALESCE(CAST(t2.card AS BIGINT), '0')) ||
 '";' ||

 '"' || CAST(current_timestamp AS VARCHAR(19)) || '"' (TITLE
 '"database_name";"table_name";"column_name";"cardinality";"current_ts"')
FROM dbc.columnsv tv
LEFT JOIN
(
 SELECT
  c.DatabaseName AS DATABASE_NAME,
  c.TABLENAME  AS TABLE_NAME,
  c.ColumnName AS COLUMN_NAME,
  c.UniqueValueCount AS CARD
 FROM dbc.tablestatsv c
 WHERE c.DatabaseName = 'your_database_name'
 AND c.RowCount <> 0
) t2
ON tv.DATABASENAME = t2.DATABASE_NAME
AND tv.TABLENAME = t2.TABLE_NAME
AND tv.COLUMNNAME = t2.COLUMN_NAME
WHERE t2.card > 0;

.EXPORT RESET

.EXPORT REPORT FILE = column-skew-script-tera.csv
.SET TITLEDASHES OFF
.SET WIDTH 10000

SELECT
```

```
'SELECT CAST(''"' || TRIM(c.DatabaseName) || '";"' || TRIM(c.TABLENAME)  || '";"'
 || TRIM(c.COLUMNNAME) || '";"'' ||
TRIM(CAST(COALESCE(MAX(cnt) * 1.0 / SUM(cnt), 0) AS NUMBER FORMAT ''9.9999'')) ||
 '";"'' ||
CAST(CURRENT_TIMESTAMP(0) AS VARCHAR(19)) || '"'' AS VARCHAR(512))
AS """DATABASE_NAME"";""TABLE_NAME"";""COLUMN_NAME"";""SKEWED"";""CURRENT_TS"""
FROM(
SELECT COUNT(*) AS cnt
FROM "' || c.DATABASENAME || '"."' || c.TABLENAME ||
'" GROUP BY "' || c.COLUMNNAME || '") t' ||
 CASE WHEN ROW_NUMBER() OVER(PARTITION BY c.DATABASENAME
 ORDER BY c.TABLENAME DESC, c.COLUMNNAME DESC) <> 1
 THEN ' UNION ALL'
 ELSE ';' END (TITLE '--SKEWED--')
FROM dbc.columnsv c
INNER JOIN
(SELECT databasename, TABLENAME
FROM dbc.tablesv  WHERE tablekind = 'T'
AND  databasename = 'your_database_name') t
ON t.databasename = c.databasename
AND t.TABLENAME = c.TABLENAME
INNER JOIN
(SELECT databasename, TABLENAME, columnname FROM  dbc.indices GROUP BY 1,2,3
WHERE  TRANSLATE_CHK (databasename USING LATIN_TO_UNICODE) + TRANSLATE_CHK
 (TABLENAME USING LATIN_TO_UNICODE) + TRANSLATE_CHK (columnname USING
 LATIN_TO_UNICODE) = 0
) i
ON i.databasename = c.databasename
AND i.TABLENAME = c.TABLENAME
AND i.columnname = c.columnname
WHERE c.ColumnType NOT IN ('CO','JN','N','++','VA','UT','AN','XM','A1','BO')
ORDER BY c.TABLENAME, c.COLUMNNAME;

.EXPORT RESET

.EXPORT REPORT FILE = column-skew-stats-tera.csv
.SET TITLEDASHES OFF
.SET WIDTH 10000

.RUN FILE = column-skew-script-tera.csv

.EXPORT RESET

.EXPORT REPORT FILE = query-stats-tera.csv
```

```
.SET TITLEDASHES OFF
.SET WIDTH 32000

SELECT
  '"' || RTRIM(CAST(SqlTextInfo AS VARCHAR(31900)), ';') || '";"' ||
  TRIM(QueryCount) || '";"' ||
  TRIM(QueryId) || '";"' ||
  TRIM(SqlRowNo) || '";"' ||
  TRIM(QueryParts) || '";"' ||
  CAST(CURRENT_TIMESTAMP(0) AS VARCHAR(19)) || '"'
(TITLE
 '"query_text";"query_count";"query_id";"sql_row_no";"query_parts";"current_ts"')
  FROM
  (
    SELECT  QueryId,  SqlTextInfo, SqlRowNo, QueryParts, QueryCount,
    SUM(QueryFirstRow) OVER (ORDER BY QueryCount DESC, QueryId ASC, SqlRowNo ASC
    ROWS UNBOUNDED PRECEDING) AS topN
    FROM
    (SELECT QueryId,  SqlTextInfo, SqlRowNo, QueryParts, QueryCount,
      CASE WHEN
      ROW_NUMBER() OVER (PARTITION BY QueryCount, SqlTextInfo ORDER BY QueryId,
 SqlRowNo) = 1 AND SqlRowNo = 1
    THEN 1 ELSE 0 END AS QueryFirstRow
    FROM (
      SELECT q.QueryId,  q.SqlTextInfo, q.SqlRowNo,
      MAX(q.SqlRowNo) OVER (PARTITION BY q.QueryId) QueryParts,
      COUNT(q.SqlTextInfo) OVER (PARTITION BY q.SqlTextInfo) QueryCount
      FROM DBC.dbqlsqltbl q
      INNER JOIN
      (
        SELECT QueryId
        FROM DBC.DBQLogTbl t
        WHERE TRIM(t.StatementType) IN ('SELECT')
        AND TRIM(t.AbortFlag) = '' AND t.ERRORCODE = 0
        AND  (CASE WHEN 'All users' IN ('All users') THEN 'All users' ELSE
 TRIM(t.USERNAME) END) IN ('All users') --user_name list
        AND t.StartTime > CURRENT_TIMESTAMP - INTERVAL '30' DAY
        GROUP BY 1
      ) t
      ON q.QueryId = t.QueryId
      INNER JOIN
      (
        SELECT QueryId
        FROM DBC.QryLogObjectsV
```

```
        WHERE ObjectDatabaseName = 'your_database_name'
        AND ObjectType = 'Tab'
        AND CollectTimeStamp > CURRENT_TIMESTAMP - INTERVAL '30' DAY
        GROUP BY 1
      ) r
      ON r.QueryId = t.QueryId
      WHERE q.CollectTimeStamp > CURRENT_TIMESTAMP - INTERVAL '30' DAY
    ) t
  ) t
  WHERE SqlTextInfo NOT LIKE '%";"%'
  ) q
  WHERE
  QueryParts >=1
  AND topN <= 50
  ORDER BY QueryCount DESC, QueryId, SqlRowNo
  QUALIFY COUNT(QueryId) OVER (PARTITION BY QueryId) = QueryParts;

.EXPORT RESET

.LOGOFF

.QUIT
```

2. 이전 단계에서 생성한 BTEQ 스크립트를 실행하는 td_run_bteq.bat 파일을 생성합니다. 다음 내용을 이 파일에 사용합니다.

```
@echo off > off-line_stats1.bteq & setLocal enableDELAYedexpansion
@echo off > off-line_stats2.bteq & setLocal enableDELAYedexpansion

set old1=your_teradata_server
set new1=%1
set old2=your_login
set new2=%2
set old3=your_database_name
set new3=%3
set old4=your_password
set /p new4=Input %2 pass?

for /f "tokens=* delims= " %%a in (off-line_stats.bteq) do (
set str1=%%a
set str1=!str1:%old1%=%new1%!
>> off-line_stats1.bteq echo !str1!
)
```

```
for /f "tokens=* delims= " %%a in (off-line_stats1.bteq) do (
set str2=%%a
set str2=!str2:%old2%=%new2%!
>> off-line_stats2.bteq echo !str2!
)

type nul > off-line_stats1.bteq

for /f "tokens=* delims= " %%a in (off-line_stats2.bteq) do (
set str3=%%a
set str3=!str3:%old3%=%new3%!
>> off-line_stats1.bteq echo !str3!
)

type nul > off-line_stats2.bteq

for /f "tokens=* delims= " %%a in (off-line_stats1.bteq) do (
set str4=%%a
set str4=!str4:%old4%=%new4%!
>> off-line_stats2.bteq echo !str4!
)

del .\off-line_stats1.bteq

echo export starting...

bteq -c UTF8 < off-line_stats.bteq > metadata_export.log

pause
```

3. 이전 단계에서 생성한 배치 파일을 실행하는 runme.bat 파일을 생성합니다. 다음 내용을 이 파일에 사용합니다.

```
.\td_run_bteq.bat ServerName UserName DatabaseName
```

runme.bat 파일에서 *ServerName*, *UserName* 및 *DatabaseName*을 해당하는 값으로 바꿉니다.

그런 다음 runme.bat 파일을 실행합니다. Amazon Redshift로 변환하는 각 데이터 웨어하우스마다 이 단계를 반복합니다.

이 스크립트를 실행한 후 각 데이터베이스에 대한 통계가 포함된 파일 3개를 수신하게 됩니다. 이 파일을 AWS SCT 프로젝트에 업로드할 수 있습니다. 이 작업을 수행하려면 프로젝트의 왼쪽 패널에서 데이터 웨어하우스를 선택하고 컨텍스트(마우스 오른쪽 버튼 클릭) 메뉴를 엽니다. Upload Statistics를 선택합니다.

Teradata에서 Amazon Redshift로의 변환 설정

Teradata에서 Amazon Redshift로의 변환 설정을 편집하려면 AWS SCT에서 설정을 선택한 다음 변환 설정을 선택합니다. 상단 목록에서 Teradata를 선택한 다음 Teradata – Amazon Redshift를 선택합니다. AWS SCT는 Teradata에서 Amazon Redshift로의 변환에 사용할 수 있는 모든 설정을 표시합니다.

AWS SCT에서 Teradata를 Amazon Redshift로 변환하는 설정에는 다음과 같은 옵션이 포함됩니다.

- 변환된 코드에서 작업 항목이 포함된 설명의 수를 제한합니다.

  Add comments in the converted code for the action items of selected severity and higher에서 작업 항목의 심각도를 선택합니다. AWS SCT가 선택된 심각도 이상의 작업 항목에 대한 변환된 코드에 설명을 추가합니다.

  예를 들어, 변환된 코드의 설명 수를 최소화하려면 오류만을 선택합니다. 변환된 코드의 모든 작업 항목에 대한 설명을 포함하려면 모든 메시지를 선택합니다.
- AWS SCT가 대상 Amazon Redshift 클러스터에 적용할 수 있는 최대 테이블 수를 설정합니다.

  The maximum number of tables for the target Amazon Redshift cluster에서 AWS SCT가 Amazon Redshift 클러스터에 적용할 수 있는 테이블 수를 선택합니다.

  Amazon Redshift에는 여러 클러스터 노드 유형에 사용하는 테이블을 제한하는 할당량이 있습니다. 자동을 선택하면 AWS SCT가 노드 유형에 따라 대상 Amazon Redshift 클러스터에 적용되는 테이블 수를 결정합니다. 값을 수동으로 선택할 수도 있습니다. 자세한 내용은 Amazon Redshift 관리 가이드의 Amazon Redshift의 할당량 및 제한 섹션을 참조하세요.

  Amazon Redshift 클러스터가 저장할 수 있는 수보다 이 숫자가 많아도 AWS SCT는 모든 소스 테이블을 변환합니다. AWS SCT는 변환된 코드를 프로젝트에 저장하며 대상 데이터베이스에는 적용하지 않습니다. 변환된 코드를 적용할 때 테이블의 Amazon Redshift 클러스터 할당량에 도달하면 AWS SCT에서 경고 메시지가 표시됩니다. 또한 AWS SCT는 테이블 수가 한도에 도달할 때까지 대상 Amazon Redshift 클러스터에 테이블을 적용합니다.
- 소스 테이블의 파티션을 Amazon Redshift의 개별 테이블로 마이그레이션합니다. 이 작업을 수행하려면 Use the UNION ALL view를 선택하고 AWS SCT가 단일 소스 테이블에 대해 생성할 수 있는 대상 테이블의 최대 수를 입력합니다.

Amazon Redshift는 테이블 파티셔닝을 지원하지 않습니다. 이 동작을 에뮬레이션하고 쿼리를 더 빠르게 실행하기 위해 AWS SCT는 소스 테이블의 각 파티션을 Amazon Redshift의 개별 테이블로 마이그레이션할 수 있습니다. 그런 다음 AWS SCT는 이러한 모든 테이블의 데이터를 포함하는 보기를 생성합니다.

AWS SCT는 소스 테이블의 파티션 수를 자동으로 결정합니다. 소스 테이블 파티셔닝 유형에 따라서는 이 숫자가 Amazon Redshift 클러스터에 적용할 수 있는 테이블의 할당량을 초과할 수 있습니다. 이 할당량에 도달하지 않도록 하려면 AWS SCT가 단일 소스 테이블의 파티션에 대해 생성할 수 있는 최대 대상 테이블 수를 입력합니다. 기본 옵션은 368개 테이블이며, 이는 1년 중 366일 동안의 파티션과 `NO RANGE` 및 `UNKNOWN` 파티션에 대한 테이블 두 개를 나타냅니다.

- Amazon Redshift 테이블 열에 압축을 적용합니다. 이렇게 하려면 Use compression encoding을 선택합니다.

  AWS SCT는 기본 Amazon Redshift 알고리즘을 사용하여 열에 압축 인코딩을 자동으로 할당합니다. 자세한 내용은 Amazon Redshift 데이터베이스 개발자 안내서의 압축 인코딩을 참조하세요.

  기본적으로 Amazon Redshift는 정렬 및 배포 키로 정의된 열에 압축을 적용하지 않습니다. 이 동작을 변경하여 이러한 열에 압축을 적용할 수 있습니다. 이렇게 하려면 Use compression encoding for KEY columns를 선택합니다. Use compression encoding 옵션을 선택한 경우에만 이 옵션을 선택할 수 있습니다.

- 변환된 코드에서 `SELECT *` 문에 대해 명시적 열 목록을 사용하려면 Use explicit column declaration을 선택합니다.

- Amazon Redshift 클러스터에서 기본 키와 고유 키의 동작을 에뮬레이션하려면 Emulate the behavior of primary and unique keys를 선택합니다.

  Amazon Redshift는 고유 키와 기본 키를 적용하지 않으며 정보 제공 목적으로만 사용합니다. 코드에서 이러한 제약 조건을 사용하는 경우에는 AWS SCT가 변환된 코드에서 해당 동작을 에뮬레이션해야 합니다.

- 대상 Amazon Redshift 테이블의 데이터 고유성을 보장합니다. 이렇게 하려면 Emulate the behavior of SET tables를 선택합니다.

  Teradata는 `SET` 구문 요소를 기본 옵션으로 사용하여 테이블을 생성합니다. `SET` 테이블에는 중복된 행을 추가할 수 없습니다. 소스 코드에서 이 고유성 제약 조건을 사용하지 않는 경우에는 이 옵션을 끕니다. 이 경우, 변환된 코드는 더 빠르게 작동합니다.

소스 코드가 고유성 제약 조건으로 테이블에서 SET 옵션을 사용하는 경우 이 옵션을 켭니다. 이 경우 AWS SCT는 변환된 코드의 INSERT..SELECT 문을 다시 작성하여 소스 데이터베이스의 동작을 에뮬레이션합니다.

Teradata에서 Amazon Redshift로의 변환 최적화 설정

Teradata에서 Amazon Redshift로의 변환 최적화 설정을 편집하려면 AWS SCT에서 설정을 선택한 다음 변환 설정을 선택합니다. 상단 목록에서 Teradata를 선택한 다음 Teradata – Amazon Redshift를 선택합니다. 왼쪽 창에서 Optimization strategies를 선택합니다. AWS SCT는 Teradata에서 Amazon Redshift로의 변환에 대한 변환 최적화 설정을 표시합니다.

AWS SCT에서 Teradata를 Amazon Redshift로 변환하는 변환 최적화 설정에는 다음과 같은 옵션이 포함됩니다.

- 자동 테이블 최적화 작업을 수행합니다. 이 작업을 수행하려면 Use Amazon Redshift automatic table tuning을 선택합니다.

  자동 테이블 최적화는 테이블 디자인을 자동으로 최적화하는 Amazon Redshift의 자체 조정 프로세스입니다. 자세한 내용은 Amazon Redshift 데이터베이스 개발자 안내서의 자동 테이블 최적화 작업을 참조하세요.

  자동 테이블 최적화만 사용하려면 Initial key selection strategy에서 없음을 선택합니다.
- 전략을 사용하여 정렬 및 배포 키를 선택합니다.

  Amazon Redshift 메타데이터, 통계 정보 또는 두 옵션을 모두 사용하여 정렬 및 배포 키를 선택할 수 있습니다. Optimization strategies 탭의 Initial key selection strategy에서 다음 옵션 중 하나를 선택합니다.
  - 메타데이터 사용, 통계 정보 무시
  - 메타데이터 무시, 통계 정보 사용
  - 메타데이터 및 통계 정보 사용

  선택한 옵션에 따라 최적화 전략을 선택할 수 있습니다. 그런 다음 각 전략에 대해 값(0~100)을 입력합니다. 이러한 값은 각 전략의 가중치를 정의합니다. AWS SCT는 이러한 가중치 값을 사용하여 각 규칙이 배포 및 정렬 키 선택에 미치는 영향을 정의합니다. 기본값은 AWS 마이그레이션 모범 사례를 기반으로 합니다.

Find small tables 전략에서 작은 테이블의 크기를 정의할 수 있습니다. Min table row count 및 Max table row count에 테이블의 최소 및 최대 행 수를 입력하여 작은 테이블로 정의합니다. AWS SCT는 작은 테이블에 ALL 배포 스타일을 적용합니다. 이 경우 전체 테이블의 사본이 모든 노드에 배포됩니다.

- 전략 세부 정보를 구성합니다.

  각 최적화 전략의 가중치를 정의하는 것 외에 최적화 설정도 구성할 수 있습니다. 이 작업을 수행하려면 Conversion optimization을 선택합니다.

  - Sort key columns limit에 정렬 키의 최대 열 수를 입력합니다.
  - Skewed threshold value에 열에 대한 스큐된 값의 백분율(0~100)을 입력합니다. AWS SCT는 스큐 값이 임계값보다 큰 열을 배포 키의 후보 목록에서 제외합니다. AWS SCT는 열의 스큐된 값을 전체 레코드 수에 대한 가장 일반적인 값 발생 횟수의 백분율로 정의합니다.
  - Top N queries from the query history table에 분석할 가장 자주 사용되는 쿼리의 수(1~100)를 입력합니다.
  - Select statistics user에서 쿼리 통계를 분석하려는 데이터베이스 사용자를 선택합니다.

  또한 Optimization strategies 탭에서 Find small tables 전략을 위한 작은 테이블의 크기를 정의할 수 있습니다. Min table row count 및 Max table row count에 테이블의 최소 및 최대 행 수를 입력하여 작은 테이블로 간주합니다. AWS SCT는 작은 테이블에 ALL 배포 스타일을 적용합니다. 이 경우 전체 테이블의 사본이 모든 노드에 배포됩니다.

Vertica를 AWS SCT에 대한 소스로 사용

AWS SCT를 사용하여 스키마, 코드 객체 및 애플리케이션 코드를 Vertica에서 Amazon Redshift로 변환할 수 있습니다.

Vertica를 소스로 사용할 수 있는 권한

Vertica를 소스로 사용하기 위해 필요한 권한은 다음과 같습니다.

- USAGE ON SCHEMA *<schema_name>*
- USAGE ON SCHEMA PUBLIC
- SELECT ON ALL TABLES IN SCHEMA *<schema_name>*
- SELECT ON ALL SEQUENCES IN SCHEMA *<schema_name>*
- EXECUTE ON ALL FUNCTIONS IN SCHEMA *<schema_name>*

- EXECUTE ON PROCEDURE *<schema_name.procedure_name(procedure_signature)>*

이전 예제에서 다음과 같이 자리 표시자를 바꿉니다.

- *schema_name*을 소스 스키마의 이름으로 바꿉니다.
- *procedure_name*을 소스 프로시저의 이름으로 바꿉니다. 변환하려는 각 프로시저에 대해 권한 부여를 반복합니다.
- *procedure_signature*를 쉼표로 구분된 프로시저 인수 유형 목록으로 바꿉니다.

Vertica에 소스로 연결

다음 절차에 따라 AWS Schema Conversion Tool을 사용하여 Vertica 소스 데이터베이스에 연결합니다.

Vertica 소스 데이터베이스에 연결하려면

1. AWS Schema Conversion Tool에서 소스 추가를 선택합니다.
2. Vertica를 선택한 후 다음을 선택합니다.

   소스 추가 대화 상자가 나타납니다.
3. 연결 이름에 데이터베이스의 이름을 입력합니다. AWS SCT는 왼쪽 패널의 트리에 이 이름을 표시합니다.
4. AWS Secrets Manager의 데이터베이스 보안 인증 정보를 사용하거나 수동으로 입력합니다.

   - Secrets Manager의 데이터베이스 보안 인증 정보를 사용하려면 다음 지침을 따릅니다.

     1. AWS Secret에서 보안 암호의 이름을 선택합니다.
     2. Populate를 선택하여 Secrets Manager에서 데이터베이스 연결 대화 상자에 있는 모든 값을 자동으로 채웁니다.

     Secrets Manager의 데이터베이스 보안 인증 사용에 대한 자세한 내용은 AWS Secrets Manager 사용 섹션을 참조하세요.
   - Vertica 소스 데이터베이스 연결 정보를 수동으로 입력하려면 다음 지침을 사용합니다.

파라미터	작업
[Server name]	소스 데이터베이스 서버의 도메인 이름 시스템(DNS) 이름 또는 IP 주소를 입력합니다.
[Server port]	소스 데이터베이스 서버에 연결하는 데 사용되는 포트를 입력합니다.
데이터베이스	Vertica 데이터베이스의 이름을 입력합니다.
User name 및 Password	소스 데이터베이스 서버에 연결하는 데 필요한 데이터베이스 보안 인증 정보를 입력합니다. 프로젝트에서 데이터베이스에 연결하도록 선택한 경우에만 AWS SCT에서 암호를 사용하여 소스 데이터베이스에 연결합니다. 소스 데이터베이스의 암호가 노출될 위험을 방지하기 위해 AWS SCT는 기본적으로 암호를 저장하지 않습니다. AWS SCT 프로젝트를 닫았다 다시 열 경우 필요에 따라 소스 데이터베이스에 연결하기 위한 암호를 입력하라는 메시지가 표시됩니다.
Use SSL(SSL 사용)	SSL(Secure Sockets Layer)을 사용하여 데이터베이스에 연결하려면 이 옵션을 선택합니다. SSL 탭에서 다음 추가 정보를 적절히 제공합니다. • Verify server certificate: 트러스트 스토어를 사용하여 서버 인증서를 확인하려면 이 옵션을 선택합니다. • 트러스트 스토어: 전역 설정에서 설정한 트러스트 스토어입니다. • Key store: 전역 설정에서 설정한 키 스토어입니다.
Store password	AWS SCT는 SSL 인증서와 데이터베이스 암호를 저장할 안전한 볼트를 생성합니다. 이 옵션을 켜면 데이터베이스 암호를 저장하고 암호 입력 없이 빠르게 데이터베이스에 연결할 수 있습니다.

파라미터	작업
Vertica driver path	소스 데이터베이스에 연결할 때 사용할 드라이버의 경로를 입력합니다. 자세한 내용은 필수 데이터베이스 드라이버 다운로드 섹션을 참조하세요. 드라이버 경로를 전역 프로젝트 설정에 저장할 경우 드라이버 경로가 연결 대화 상자에 표시되지 않습니다. 자세한 내용은 전역 설정에 드라이버 경로 저장 섹션을 참조하세요.

5. Test Connection을 선택하여 AWS SCT가 소스 데이터베이스에 연결할 수 있는지 확인합니다.
6. 연결을 선택하여 소스 데이터베이스에 연결합니다.

Vertica에서 Amazon Redshift로의 변환 설정

Vertica에서 Amazon Redshift로의 변환 설정을 편집하려면 AWS SCT에서 설정을 선택한 다음 변환 설정을 선택합니다. 상단 목록에서 Vertica를 선택한 다음 Vertica – Amazon Redshift를 선택합니다. AWS SCT는 Vertica에서 Amazon Redshift로의 변환에 사용할 수 있는 모든 설정을 표시합니다.

AWS SCT에서 Vertica를 Amazon Redshift로 변환하는 설정에는 다음과 같은 옵션이 포함됩니다.

- 변환된 코드에서 작업 항목이 포함된 설명의 수를 제한합니다.

  Add comments in the converted code for the action items of selected severity and higher에서 작업 항목의 심각도를 선택합니다. AWS SCT가 선택된 심각도 이상의 작업 항목에 대한 변환된 코드에 설명을 추가합니다.

  예를 들어, 변환된 코드의 설명 수를 최소화하려면 오류만을 선택합니다. 변환된 코드의 모든 작업 항목에 대한 설명을 포함하려면 모든 메시지를 선택합니다.
- AWS SCT가 대상 Amazon Redshift 클러스터에 적용할 수 있는 최대 테이블 수를 설정합니다.

  The maximum number of tables for the target Amazon Redshift cluster에서 AWS SCT가 Amazon Redshift 클러스터에 적용할 수 있는 테이블 수를 선택합니다.

  Amazon Redshift에는 여러 클러스터 노드 유형에 사용하는 테이블을 제한하는 할당량이 있습니다. 자동을 선택하면 AWS SCT가 노드 유형에 따라 대상 Amazon Redshift 클러스터에 적용되는 테이블 수를 결정합니다. 값을 수동으로 선택할 수도 있습니다. 자세한 내용은 Amazon Redshift 관리 가이드의 Amazon Redshift의 할당량 및 제한 섹션을 참조하세요.

Amazon Redshift 클러스터가 저장할 수 있는 수보다 이 숫자가 많아도 AWS SCT는 모든 소스 테이블을 변환합니다. AWS SCT는 변환된 코드를 프로젝트에 저장하며 대상 데이터베이스에는 적용하지 않습니다. 변환된 코드를 적용할 때 테이블의 Amazon Redshift 클러스터 할당량에 도달하면 AWS SCT에서 경고 메시지가 표시됩니다. 또한 AWS SCT는 테이블 수가 한도에 도달할 때까지 대상 Amazon Redshift 클러스터에 테이블을 적용합니다.

- 소스 테이블의 파티션을 Amazon Redshift의 개별 테이블로 마이그레이션합니다. 이 작업을 수행하려면 Use the UNION ALL view를 선택하고 AWS SCT가 단일 소스 테이블에 대해 생성할 수 있는 대상 테이블의 최대 수를 입력합니다.

  Amazon Redshift는 테이블 파티셔닝을 지원하지 않습니다. 이 동작을 에뮬레이션하고 쿼리를 더 빠르게 실행하기 위해 AWS SCT는 소스 테이블의 각 파티션을 Amazon Redshift의 개별 테이블로 마이그레이션할 수 있습니다. 그런 다음 AWS SCT는 이러한 모든 테이블의 데이터를 포함하는 보기를 생성합니다.

  AWS SCT는 소스 테이블의 파티션 수를 자동으로 결정합니다. 소스 테이블 파티셔닝 유형에 따라서는 이 숫자가 Amazon Redshift 클러스터에 적용할 수 있는 테이블의 할당량을 초과할 수 있습니다. 이 할당량에 도달하지 않도록 하려면 AWS SCT가 단일 소스 테이블의 파티션에 대해 생성할 수 있는 최대 대상 테이블 수를 입력합니다. 기본 옵션은 368개 테이블이며, 이는 1년 중 366일 동안의 파티션과 `NO  RANGE` 및 `UNKNOWN` 파티션에 대한 테이블 두 개를 나타냅니다.
- Amazon Redshift 테이블 열에 압축을 적용합니다. 이렇게 하려면 Use compression encoding을 선택합니다.

  AWS SCT는 기본 Amazon Redshift 알고리즘을 사용하여 열에 압축 인코딩을 자동으로 할당합니다. 자세한 내용은 Amazon Redshift 데이터베이스 개발자 안내서의 압축 인코딩을 참조하세요.

  기본적으로 Amazon Redshift는 정렬 및 배포 키로 정의된 열에 압축을 적용하지 않습니다. 이 동작을 변경하여 이러한 열에 압축을 적용할 수 있습니다. 이렇게 하려면 Use compression encoding for KEY columns를 선택합니다. Use compression encoding 옵션을 선택한 경우에만 이 옵션을 선택할 수 있습니다.

Vertica에서 Amazon Redshift로의 변환 최적화 설정

Vertica에서 Amazon Redshift로의 변환 최적화 설정을 편집하려면 AWS SCT에서 설정을 선택한 다음 변환 설정을 선택합니다. 상단 목록에서 Vertica를 선택한 다음 Vertica – Amazon Redshift를 선택합니다. 왼쪽 창에서 Optimization strategies를 선택합니다. AWS SCT는 Vertica에서 Amazon Redshift로의 변환에 대한 변환 최적화 설정을 표시합니다.

AWS SCT에서 Vertica를 Amazon Redshift로 변환하는 변환 최적화 설정에는 다음과 같은 옵션이 포함됩니다.

- 자동 테이블 최적화 작업을 수행합니다. 이 작업을 수행하려면 Use Amazon Redshift automatic table tuning을 선택합니다.

  자동 테이블 최적화는 테이블 디자인을 자동으로 최적화하는 Amazon Redshift의 자체 조정 프로세스입니다. 자세한 내용은 Amazon Redshift 데이터베이스 개발자 안내서의 자동 테이블 최적화 작업을 참조하세요.

  자동 테이블 최적화만 사용하려면 Initial key selection strategy에서 없음을 선택합니다.
- 전략을 사용하여 정렬 및 배포 키를 선택합니다.

  Amazon Redshift 메타데이터, 통계 정보 또는 두 옵션을 모두 사용하여 정렬 및 배포 키를 선택할 수 있습니다. Optimization strategies 탭의 Initial key selection strategy에서 다음 옵션 중 하나를 선택합니다.
  - 메타데이터 사용, 통계 정보 무시
  - 메타데이터 무시, 통계 정보 사용
  - 메타데이터 및 통계 정보 사용

  선택한 옵션에 따라 최적화 전략을 선택할 수 있습니다. 그런 다음 각 전략에 대해 값(0~100)을 입력합니다. 이러한 값은 각 전략의 가중치를 정의합니다. AWS SCT는 이러한 가중치 값을 사용하여 각 규칙이 배포 및 정렬 키 선택에 미치는 영향을 정의합니다. 기본값은 AWS 마이그레이션 모범 사례를 기반으로 합니다.

  Find small tables 전략에서 작은 테이블의 크기를 정의할 수 있습니다. Min table row count 및 Max table row count에 테이블의 최소 및 최대 행 수를 입력하여 작은 테이블로 정의합니다. AWS SCT는 작은 테이블에 ALL 배포 스타일을 적용합니다. 이 경우 전체 테이블의 사본이 모든 노드에 배포됩니다.
- 전략 세부 정보를 구성합니다.

  각 최적화 전략의 가중치를 정의하는 것 외에 최적화 설정도 구성할 수 있습니다. 이 작업을 수행하려면 Conversion optimization을 선택합니다.
  - Sort key columns limit에 정렬 키의 최대 열 수를 입력합니다.
  - Skewed threshold value에 열에 대한 스큐된 값의 백분율(0~100)을 입력합니다. AWS SCT는 스큐 값이 임계값보다 큰 열을 배포 키의 후보 목록에서 제외합니다. AWS SCT는 열의 스큐된 값을 전체 레코드 수에 대한 가장 일반적인 값 발생 횟수의 백분율로 정의합니다.

- Top N queries from the query history table에 분석할 가장 자주 사용되는 쿼리의 수(1~100)를 입력합니다.
- Select statistics user에서 쿼리 통계를 분석하려는 데이터베이스 사용자를 선택합니다.

또한 Optimization strategies 탭에서 Find small tables 전략을 위한 작은 테이블의 크기를 정의할 수 있습니다. Min table row count 및 Max table row count에 테이블의 최소 및 최대 행 수를 입력하여 작은 테이블로 간주합니다. AWS SCT는 작은 테이블에 ALL 배포 스타일을 적용합니다. 이 경우 전체 테이블의 사본이 모든 노드에 배포됩니다.

AWS SCT에서 매핑 규칙 생성

단일 AWS SCT 프로젝트에 여러 소스 및 대상 데이터베이스를 추가할 수 있습니다. 이렇게 하면 여러 데이터베이스를 서로 다른 대상 플랫폼으로 마이그레이션할 때 프로젝트 관리가 간소화됩니다.

새 프로젝트를 만들고 소스 및 대상 데이터베이스를 추가한 후 매핑 규칙을 생성합니다. AWS SCT에서 마이그레이션 평가 보고서를 생성하고 데이터베이스 스키마를 변환하려면 하나 이상의 매핑 규칙이 필요합니다.

매핑 규칙은 소스 데이터베이스 스키마 또는 소스 데이터베이스와 대상 데이터베이스 플랫폼을 포함하는 소스-대상 페어를 설명합니다. 단일 AWS SCT 프로젝트에서 여러 매핑 규칙을 생성할 수 있습니다. 모든 소스 데이터베이스 스키마를 올바른 대상 데이터베이스 플랫폼으로 변환하려면 매핑 규칙을 사용합니다.

변환된 코드의 스키마 이름을 변경하려면 마이그레이션 규칙을 설정합니다. 예를 들어 마이그레이션 규칙을 사용하여 스키마 이름을 바꾸거나, 객체 이름에 접두사를 추가하거나, 열 데이터 정렬을 변경하거나, 데이터 유형을 변경할 수 있습니다. 변환된 코드에 이러한 변경 내용을 적용하려면 소스 스키마를 변환하기 전에 마이그레이션 규칙을 생성해야 합니다. 자세한 내용은 마이그레이션 규칙 생성 섹션을 참조하세요.

지원되는 데이터베이스 변환 페어에 대해서만 매핑 규칙을 생성할 수 있습니다. 지원되는 변환 페어 목록은 AWS SCT 소스 섹션을 참조하세요.

AWS SCT 버전 1.0.655 이전 버전으로 저장된 프로젝트를 열면 AWS SCT가 모든 소스 데이터베이스 스키마에 대한 매핑 규칙을 대상 데이터베이스 플랫폼에 자동으로 생성합니다. 다른 대상 데이터베이스 플랫폼을 추가하려면 기존 매핑 규칙을 삭제한 다음 새 매핑 규칙을 생성합니다.

주제

- 새 매핑 규칙 추가
- 매핑 규칙 관리
- 가상 대상 사용
- 단일 AWS SCT 프로젝트에서 여러 서버 사용 시 제한 사항

새 매핑 규칙 추가

단일 프로젝트에서 여러 매핑 규칙을 생성할 수 있습니다. AWS SCT는 매핑 규칙을 프로젝트의 일부로 저장합니다. 프로젝트를 연 상태에서 다음 절차에 따라 새 매핑 규칙을 추가합니다.

매핑 규칙을 생성하려면

1. 보기 메뉴에서 매핑 보기를 선택합니다.
2. 왼쪽 패널에서 매핑 규칙에 추가할 스키마 또는 데이터베이스를 선택합니다.
3. 오른쪽 패널에서 선택한 소스 스키마 또는 데이터베이스의 대상 데이터베이스 플랫폼을 선택합니다.

   가상 데이터베이스 플랫폼을 대상으로 선택할 수 있습니다. 자세한 내용은 가상 대상 사용 섹션을 참조하세요.
4. 매핑 생성을 선택합니다.

   AWS SCT가 새 매핑 규칙을 Server mappings 목록에 추가합니다.

모든 변환 페어에 대한 매핑 규칙을 추가합니다. 평가 보고서를 생성하거나 데이터베이스 스키마를 변환하려면 보기 메뉴에서 Main view를 선택합니다.

AWS SCT가 매핑 규칙에 속하는 모든 스키마 객체를 굵게 강조 표시합니다.

매핑 규칙 관리

기존 매핑 규칙을 필터링 또는 삭제하고 AWS Schema Conversion Tool(AWS SCT) 프로젝트에 새 매핑 규칙을 추가할 수 있습니다.

전체 소스 데이터베이스에 대한 매핑 규칙을 생성할 때는 AWS SCT에서 각 소스 데이터베이스 스키마에 대해 하나씩 매핑 규칙을 생성합니다. 수십 개의 스키마 또는 데이터베이스가 포함된 프로젝트의 경우 특정 스키마에 어떤 대상이 사용되는지 파악하기 어려울 수 있습니다. 스키마의 매핑 규칙을 빠르게 찾으려면 AWS SCT에서 다음 필터 옵션 중 하나 또는 여러 개를 사용합니다.

매핑 규칙을 필터링하려면

1. 보기 메뉴에서 매핑 보기를 선택합니다.
2. 소스 서버에서 소스 데이터베이스를 선택합니다.

   필터 기본값은 모두이며, AWS SCT가 모든 소스 데이터베이스의 매핑 규칙을 표시합니다.
3. 소스 스키마에 소스 스키마 이름을 입력합니다. 스키마 이름에 있는 여러 기호를 바꾸려면 퍼센트(%)를 와일드카드로 사용합니다.

   필터 기본값은 %이며, AWS SCT가 모든 소스 데이터베이스 스키마 이름의 매핑 규칙을 표시합니다.

4. Has migration rules에서 예를 선택하여 데이터 마이그레이션 규칙이 생성되는 매핑 규칙을 표시합니다. 데이터 마이그레이션 규칙이 없는 매핑 규칙을 표시하려면 아니요를 선택합니다. 자세한 내용은 에서 데이터 마이그레이션 규칙 생성 AWS SCT 섹션을 참조하세요.

   필터 기본값은 모두이며, AWS SCT가 모든 매핑 규칙을 표시합니다.

5. Target servers에서 대상 데이터베이스를 선택합니다.

   필터 기본값은 모두이며, AWS SCT가 모든 대상 데이터베이스의 매핑 규칙을 표시합니다.

프로젝트를 연 상태에서 다음 절차에 따라 매핑 규칙을 삭제합니다. 매핑 규칙 추가에 대한 자세한 내용은 새 매핑 규칙 추가 섹션을 참조하세요.

매핑 규칙을 삭제하려면

1. 보기 메뉴에서 매핑 보기를 선택합니다.
2. Server mappings에서 삭제할 매핑 규칙을 선택합니다.
3. Delete selected mappings를 선택합니다.

   AWS SCT가 선택된 매핑 규칙을 삭제합니다.

가상 대상 사용

AWS SCT가 소스 데이터베이스 스키마를 지원되는 대상 데이터베이스 플랫폼으로 변환하는 방법을 확인할 수 있습니다. 이 작업을 수행하기 위해 기존 대상 데이터베이스에 연결할 필요는 없습니다. 대신 매핑 규칙을 만들 때 오른쪽 패널에서 가상 대상 데이터베이스 플랫폼을 선택할 수 있습니다. 자세한 내용은 새 매핑 규칙 추가 섹션을 참조하세요. 가상 대상 데이터베이스 플랫폼 목록을 볼 수 있도록 오른쪽 패널에서 서버, NoSQL 클러스터 및 ETL 노드를 확장합니다.

AWS SCT는 다음과 같은 가상 대상 데이터베이스 플랫폼을 지원합니다.

- Amazon Aurora MySQL-Compatible Edition
- Amazon Aurora PostgreSQL-Compatible Edition
- Amazon DynamoDB
- Amazon Redshift
- Amazon Redshift 및 AWS Glue
- AWS Glue

- AWS Glue Studio
- Babelfish for Aurora PostgreSQL
- MariaDB
- Microsoft SQL Server
- MySQL
- Oracle
- PostgreSQL

Babelfish for Aurora PostgreSQL을 대상 데이터베이스 플랫폼으로 사용하는 경우 데이터베이스 마이그레이션 평가 보고서만 생성할 수 있습니다. 자세한 내용은 the section called “마이그레이션 평가 보고서” 섹션을 참조하세요.

가상 대상 데이터베이스 플랫폼을 사용하는 경우 변환된 코드를 파일에 저장할 수 있습니다. 자세한 내용은 the section called “변환된 스키마 저장” 섹션을 참조하세요.

단일 AWS SCT 프로젝트에서 여러 서버 사용 시 제한 사항

단일 AWS SCT 프로젝트에서 여러 서버를 사용하여 스키마를 변환할 때는 다음과 같은 제한 사항이 적용됩니다.

- 프로젝트에는 동일한 서버를 한 번만 추가할 수 있습니다.
- 서버 스키마를 특정 대상 스키마에 매핑할 수 없으며 대상 서버에만 매핑할 수 있습니다. AWS SCT는 변환 중에 대상 스키마를 생성합니다.
- 하위 수준 소스 객체를 대상 서버에 매핑할 수 없습니다.
- 하나의 소스 스키마를 프로젝트에 있는 하나의 대상 서버에만 매핑할 수 있습니다.
- 평가 보고서를 생성하거나, 스키마를 변환하거나, 데이터를 추출하려면 소스를 대상 서버에 매핑해야 합니다.

변환 보고서 생성

데이터베이스 전환을 계획할 때는 관련된 내용을 이해하는 데 도움이 되는 몇 가지 보고서를 만드는 것이 좋습니다. AWS Schema Conversion Tool를 사용하여 보고서를 생성할 수 있습니다.

AWS SCT를 사용하여 데이터베이스 마이그레이션 평가 보고서를 생성할 수 있습니다. 이 보고서는 스키마 변환 작업에 대한 요약 정보 및 대상 데이터베이스로 자동 변환할 수 없는 항목에 대한 세부 정보를 제공합니다. 이 보고서를 사용하면 AWS SCT를 사용하여 완료할 수 있는 프로젝트의 양과 변환을 완료하는 데 필요한 기타 사항을 평가할 수 있습니다. 평가 보고서를 생성하려면 AWS SCT에서 데이터베이스의 컨텍스트(마우스 오른쪽 버튼 클릭) 메뉴에 있는 보고서 생성을 사용합니다.

주제

- AWS SCT를 사용하여 마이그레이션 평가 보고서 작성

AWS SCT를 사용하여 마이그레이션 평가 보고서 작성

스키마 변환의 복잡성을 예측하기 위해 생성하는 평가 보고서는 AWS Schema Conversion Tool의 중요한 부분입니다. 이 데이터베이스 마이그레이션 평가 보고서는 모든 스키마 변환 작업을 요약하고 대상 DB 인스턴스의 DB 엔진으로 변환할 수 없는 스키마에 대한 작업 항목을 자세히 설명합니다. 애플리케이션에서 보고서를 보거나 쉼표로 구분된 값(CSV) 또는 PDF 파일로 내보낼 수 있습니다.

단일 프로젝트에 여러 소스 및 대상 데이터베이스를 추가하는 경우 AWS SCT에서는 모든 전환 쌍에 대한 보고서를 하나의 데이터베이스 마이그레이션 평가 보고서로 집계합니다.

가상 대상 데이터베이스 플랫폼을 사용하면 평가 보고서를 생성하고 선택한 데이터베이스 플랫폼으로의 마이그레이션 복잡성을 이해할 수 있습니다. 이 경우 대상 데이터베이스 플랫폼에 연결할 필요는 없습니다. 예를 들어 Babelfish for Aurora PostgreSQL을 가상 대상 데이터베이스 플랫폼으로 사용하여 데이터베이스 마이그레이션 평가 보고서를 생성할 수 있습니다. 가상 대상 데이터베이스 플랫폼에 대한 자세한 내용은 the section called “가상 대상”을 참조하세요.

마이그레이션 평가 보고서에는 다음 내용이 포함됩니다.

- 핵심 요약
- 라이선스 평가
- 대상에서 사용할 수 없는 소스 데이터베이스의 모든 기능을 나타내는 클라우드 지원
- 서버 객체 변환, 백업 제안, 연결된 서버 변경을 포함한 권장 사항

이 보고서에는 대상 DB 인스턴스에 대해 자동으로 변환할 수 없는 동등한 코드를 작성하는 데 드는 예상 작업량도 포함되어 있습니다.

AWS SCT를 사용하여 기존 스키마를 Amazon RDS DB 인스턴스로 마이그레이션하는 경우, 보고서를 사용하여 AWS 클라우드로 이전하기 위한 요구 사항을 분석하고 라이선스 유형을 변경할 수 있습니다.

주제

- 데이터베이스 마이그레이션 평가 보고서 생성
- 평가 보고서 보기
- 평가 보고서 저장
- 평가 보고서 구성
- 데이터베이스 마이그레이션을 위한 다중 서버 평가 보고서 생성

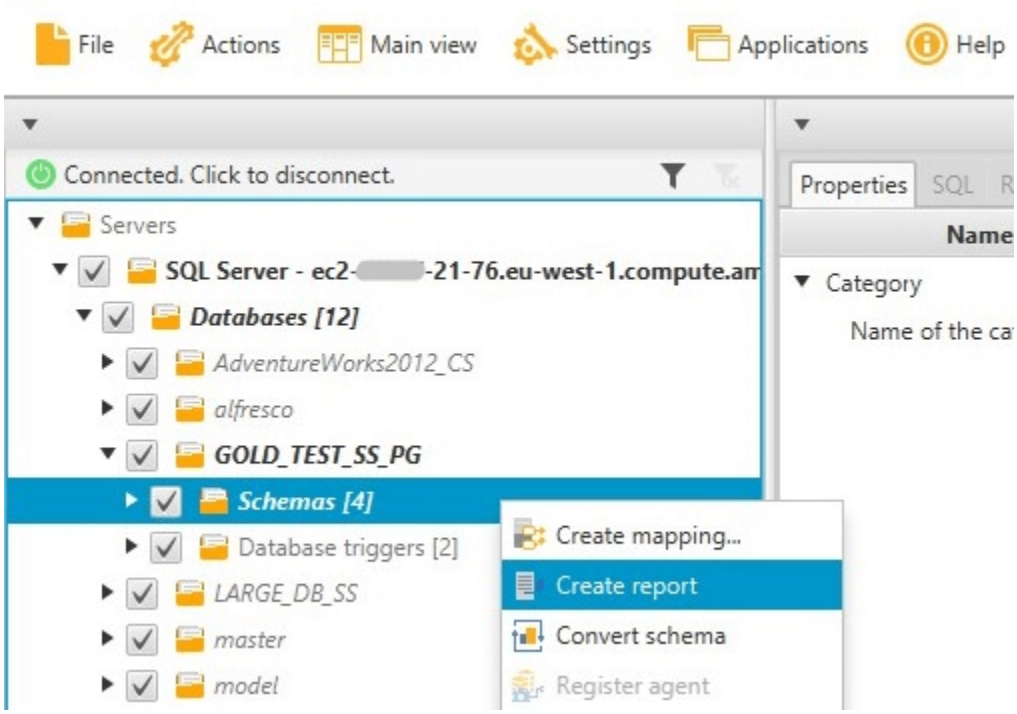
데이터베이스 마이그레이션 평가 보고서 생성

다음 절차에 따라 데이터베이스 마이그레이션 평가 보고서를 생성합니다.

데이터베이스 마이그레이션 평가 보고서를 생성하려면

1. 평가 보고서를 생성하는 데 사용할 소스 데이터베이스 스키마에 대한 매핑 규칙을 생성했는지 확인합니다. 자세한 내용은 새 매핑 규칙 추가 섹션을 참조하세요.
2. 보기 메뉴에서 Main view를 선택합니다.
3. 소스 데이터베이스 스키마를 표시하는 왼쪽 패널에서 평가 보고서를 생성할 스키마 객체를 선택합니다. 보고서에 여러 데이터베이스 스키마를 포함하려면 상위 노드(예: 스키마)를 선택합니다.

   평가 보고서를 만들 때 사용할 모든 스키마 객체의 확인란을 선택했는지 확인합니다.
4. 객체의 컨텍스트(마우스 오른쪽 버튼 클릭) 메뉴를 열고 보고서 생성을 선택합니다.



평가 보고서 보기

평가 보고서를 생성하면 평가 보고서 보기가 열리고 다음 탭이 표시됩니다.

- 요약
- 작업 항목

요약 탭에는 자동으로 변환되거나 변환되지 않은 항목이 표시됩니다.

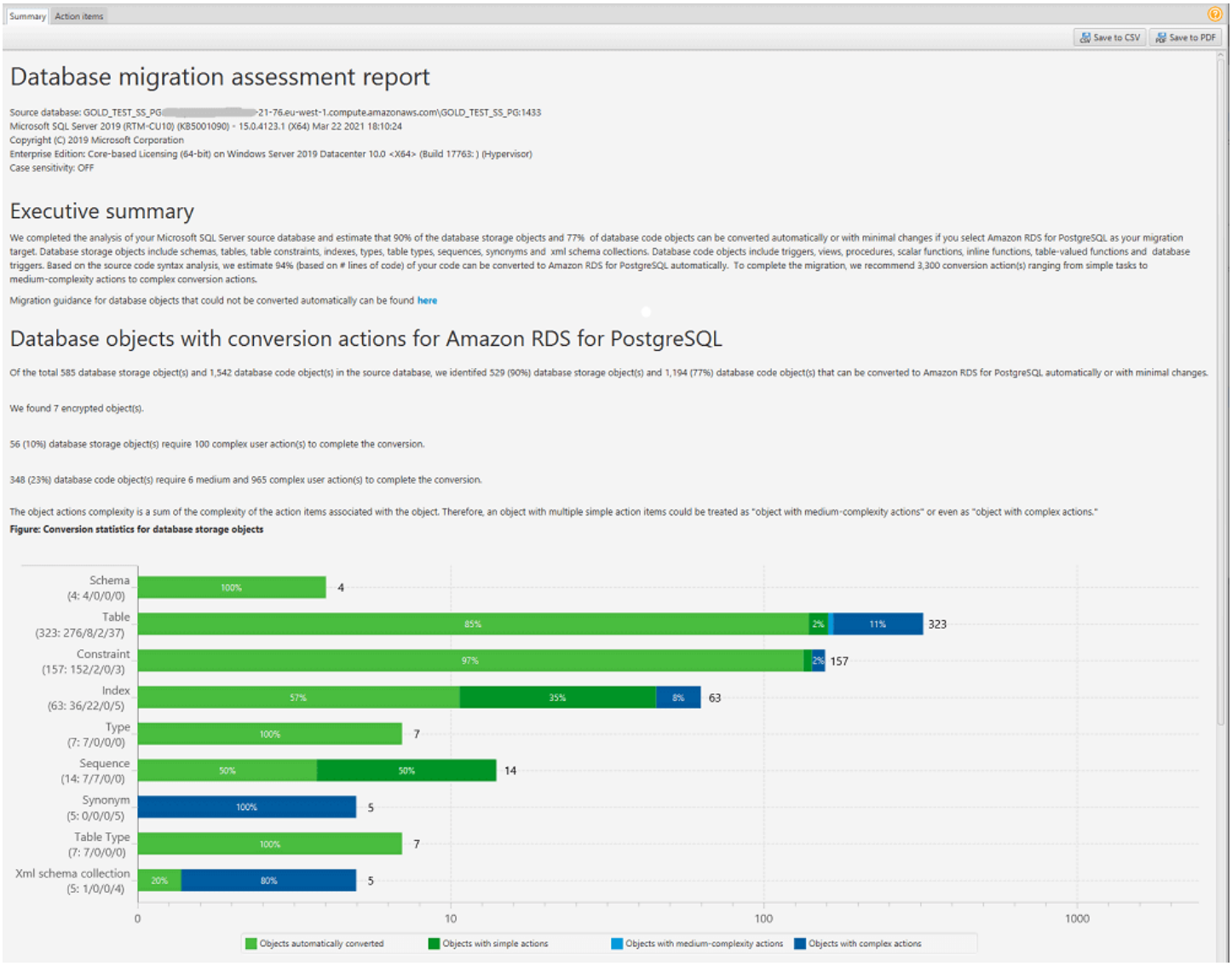
작업 항목 탭에는 자동으로 변환할 수 없는 항목과 이에 대한 권장 조치가 표시됩니다.

주제

- 평가 보고서 요약
- 평가 보고서 작업 항목
- 평가 보고서 경고 메시지

평가 보고서 요약

요약 탭에는 데이터베이스 마이그레이션 평가 보고서의 요약 정보가 표시됩니다. 자동으로 변환된 항목과 자동으로 변환되지 않은 항목이 표시됩니다.



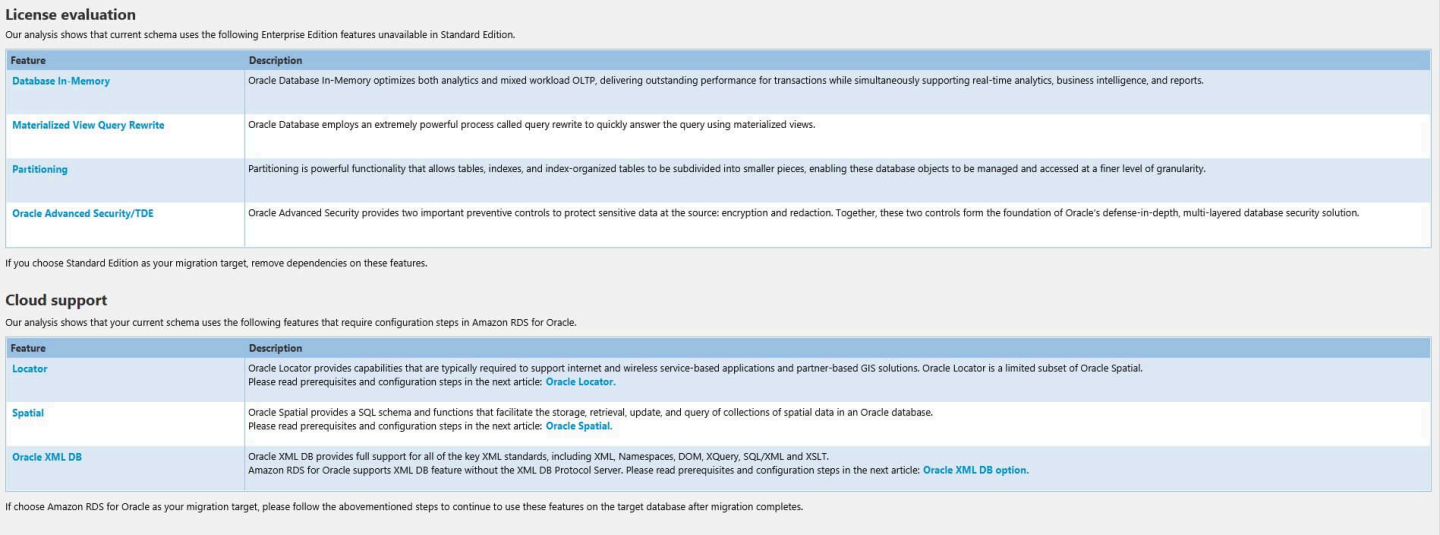
대상 데이터베이스 엔진으로 자동 변환할 수 없는 스키마 항목의 경우, 요약에는 대상 DB 인스턴스에서 소스에 있는 것과 동일한 스키마 항목을 생성하는 데 필요한 예상 작업량이 포함됩니다.

보고서에는 이러한 스키마 항목을 변환하는 데 걸리는 예상 시간이 다음과 같이 분류됩니다.

- 단순 - 2시간 이내에 완료할 수 있는 작업입니다.
- 중간 - 더 복잡하고 2~6시간 사이에 완료할 수 있는 작업입니다.
- Significant - 매우 복잡하며 완료하는 데 6시간 이상 걸리는 작업입니다.

라이선스 평가 및 클라우드 지원 섹션에는 동일한 엔진을 실행하는 Amazon RDS DB 인스턴스로 기존 온프레미스 데이터베이스 스키마를 이동하는 방법에 대한 정보가 포함되어 있습니다. 예를 들어 라이

선스 유형을 변경하려는 경우 보고서의 이 섹션에서는 현재 데이터베이스에서 제거해야 하는 기능을 설명합니다.



License evaluation

Our analysis shows that current schema uses the following Enterprise Edition features unavailable in Standard Edition.

Feature	Description
Database In-Memory	Oracle Database In-Memory optimizes both analytics and mixed workload OLTP, delivering outstanding performance for transactions while simultaneously supporting real-time analytics, business intelligence, and reports.
Materialized View Query Rewrite	Oracle Database employs an extremely powerful process called query rewrite to quickly answer the query using materialized views.
Partitioning	Partitioning is powerful functionality that allows tables, indexes, and index-organized tables to be subdivided into smaller pieces, enabling these database objects to be managed and accessed at a finer level of granularity.
Oracle Advanced Security/TDE	Oracle Advanced Security provides two important preventive controls to protect sensitive data at the source: encryption and redaction. Together, these two controls form the foundation of Oracle's defense-in-depth, multi-layered database security solution.

If you choose Standard Edition as your migration target, remove dependencies on these features.

Cloud support

Our analysis shows that your current schema uses the following features that require configuration steps in Amazon RDS for Oracle.

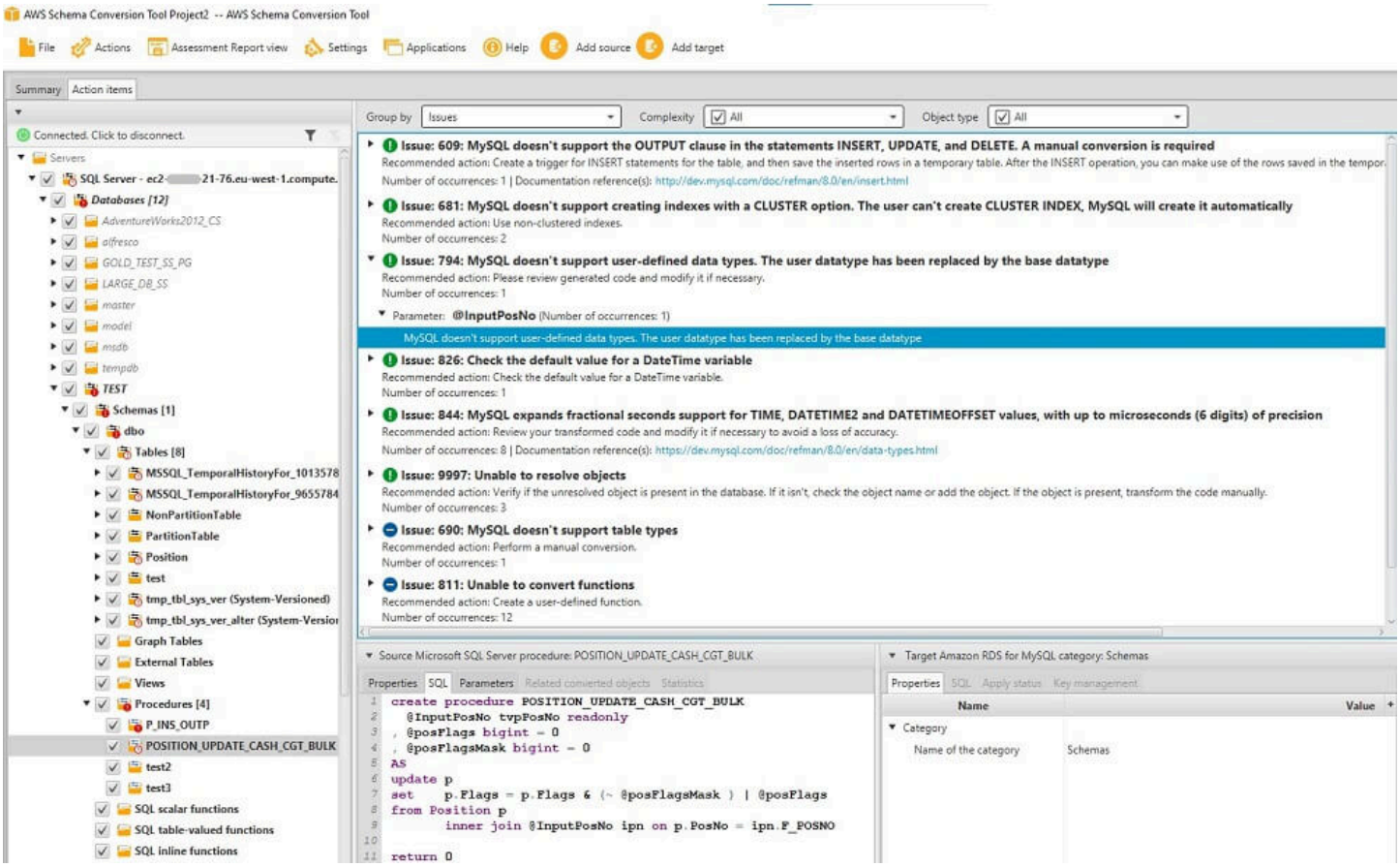
Feature	Description
Locator	Oracle Locator provides capabilities that are typically required to support internet and wireless service-based applications and partner-based GIS solutions. Oracle Locator is a limited subset of Oracle Spatial. Please read prerequisites and configuration steps in the next article: Oracle Locator.
Spatial	Oracle Spatial provides a SQL schema and functions that facilitate the storage, retrieval, update, and query of collections of spatial data in an Oracle database. Please read prerequisites and configuration steps in the next article: Oracle Spatial.
Oracle XML DB	Oracle XML DB provides full support for all of the key XML standards, including XML, Namespaces, DOM, XQuery, SQL/XML and XSLT. Amazon RDS for Oracle supports XML DB feature without the XML DB Protocol Server. Please read prerequisites and configuration steps in the next article: Oracle XML DB option.

If choose Amazon RDS for Oracle as your migration target, please follow the abovementioned steps to continue to use these features on the target database after migration completes.

평가 보고서 작업 항목

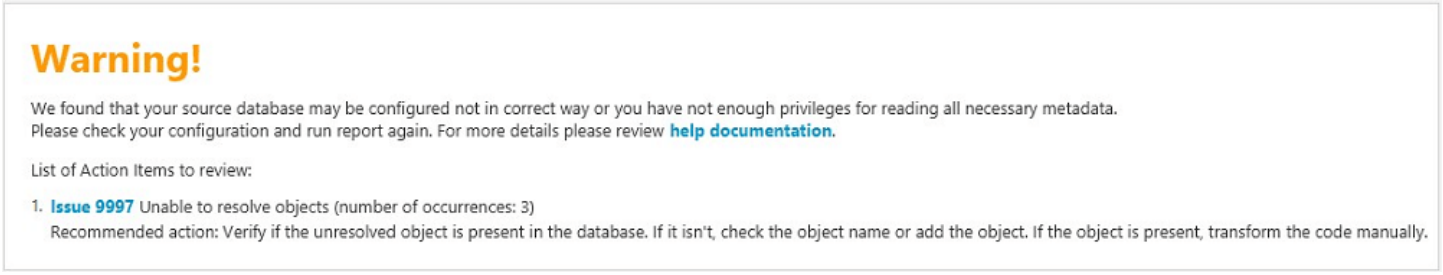
평가 보고서 보기에는 작업 항목 탭도 포함되어 있습니다. 이 탭에는 대상 Amazon RDS DB 인스턴스의 데이터베이스 엔진으로 자동 변환할 수 없는 항목 목록이 포함되어 있습니다. 목록에서 작업 항목을 선택하면 AWS SCT가 스키마에서 해당 작업 항목이 적용되는 항목을 강조 표시합니다.

또한 보고서에는 스키마 항목을 수동으로 변환하는 방법에 대한 권장 사항도 포함되어 있습니다. 예를 들어 평가를 실행한 후 데이터베이스/스키마에 대한 세부 보고서를 통해 작업 항목 변환을 위한 권장 사항을 설계하고 구현하는 데 필요한 작업량을 확인할 수 있습니다. 수동 변환 처리 방법 결정에 대한 자세한 내용은 AWS SCT에서 수동 변환 처리 섹션을 참조하세요.



평가 보고서 경고 메시지

다른 데이터베이스 엔진으로 변환하는 작업의 복잡성을 평가하려면 AWS SCT가 소스 데이터베이스의 객체에 액세스할 수 있어야 합니다. 스캔 중에 문제가 발생하여 SCT에서 평가를 수행할 수 없는 경우에는 전체 전환율이 감소했음을 나타내는 경고 메시지가 표시됩니다.



스캔 중에 AWS SCT에서 문제가 발생할 수 있는 이유는 다음과 같습니다.

- 데이터베이스에 연결된 사용자 계정으로는 필요한 모든 객체에 액세스할 수 없습니다.
- 스키마에 인용된 객체가 더 이상 데이터베이스에 존재하지 않습니다.
- SCT가 암호화된 객체를 평가하려고 합니다.

SCT에 필요한 데이터베이스 보안 권한 및 권한 부여에 대한 자세한 내용은 이 설명서의 해당 소스 데이터베이스 섹션에 대한 AWS SCT 소스를 참조하세요.

평가 보고서 저장

데이터베이스 마이그레이션 평가 보고서를 만든 후 데이터베이스 마이그레이션 평가 보고서의 로컬 사본을 PDF 파일 또는 CSV(쉼표로 구분된 값) 파일로 저장할 수 있습니다.

데이터베이스 마이그레이션 평가 보고서를 PDF 파일로 저장하려면

1. 상단 메뉴에서 보기를 선택한 다음, Assessment report view를 선택합니다.
2. 요약 탭을 선택합니다.
3. 오른쪽 상단에서 Save to PDF를 선택합니다.

데이터베이스 마이그레이션 평가 보고서를 CSV 파일로 저장하려면

1. 상단 메뉴에서 보기를 선택한 다음, Assessment report view를 선택합니다.
2. 요약 탭을 선택합니다.
3. 오른쪽 상단에서 Save to CSV를 선택합니다.

PDF 파일에는 다음 예제와 같이 요약 및 작업 항목 정보가 모두 포함되어 있습니다.

Database objects with conversion actions for Amazon RDS for PostgreSQL

Of the total 585 database storage object(s) and 1,542 database code object(s) in the source database, we identifed 529 (90%) database storage object(s) and 1,194 (77%) database code object(s) that can be converted to Amazon RDS for PostgreSQL automatically or with minimal changes.

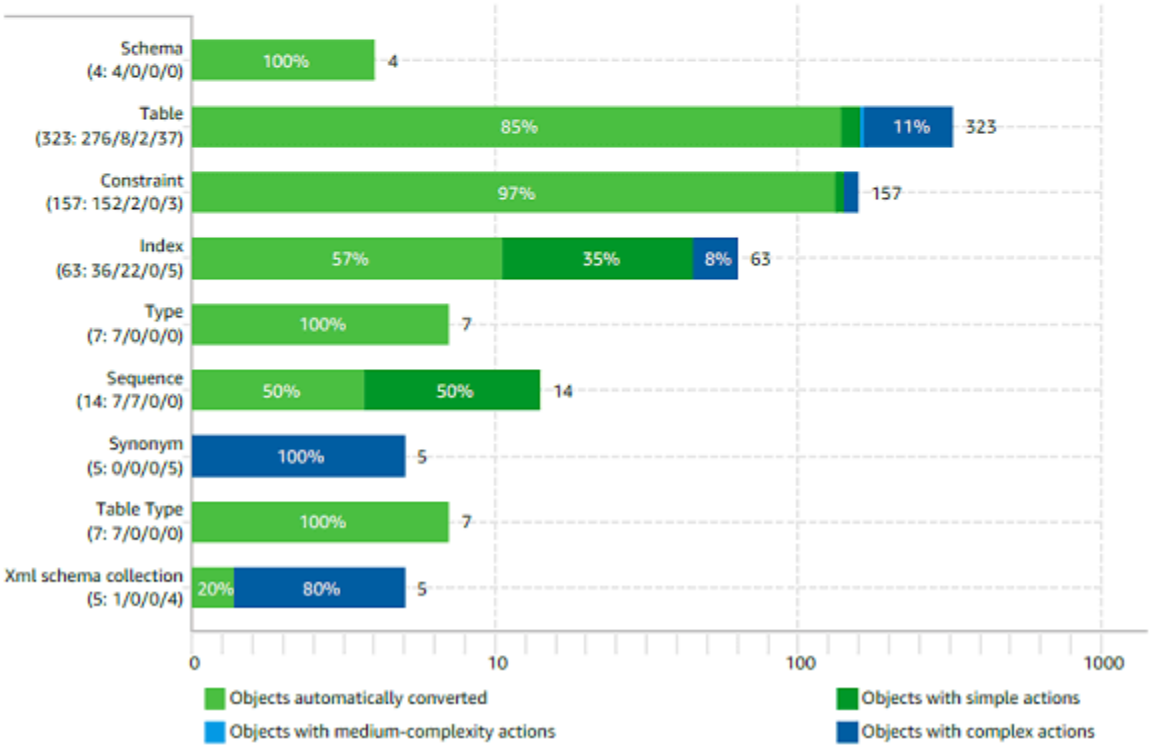
We found 7 encrypted object(s).

56 (10%) database storage object(s) require 100 complex user action(s) to complete the conversion.

348 (23%) database code object(s) require 6 medium and 965 complex user action(s) to complete the conversion.

The object actions complexity is a sum of the complexity of the action items associated with the object. Therefore, an object with multiple simple action items could be treated as "object with medium-complexity actions" or even as "object with complex actions."

Figure: Conversion statistics for database storage objects



Save to CSV 옵션을 선택하면 AWS SCT에서 세 개의 CSV 파일을 생성합니다.

첫 번째 CSV 파일에는 작업 항목에 대한 다음과 같은 정보가 포함됩니다.

- 범주
- 발생 - 항목의 파일 이름, 행 번호, 위치
- 작업 항목 번호
- 제목
- 그룹

- 설명
- 문서 참조
- 권장 조치
- 예상 복잡성

두 번째 CSV 파일은 이름에 `Action_Items_Summary` 접미사를 포함하며 모든 작업 항목의 발생 횟수에 대한 정보를 포함합니다.

다음 예제에서 Learning curve effort 열의 값은 각 작업 항목을 전환하는 방법을 설계하는 데 필요한 작업량을 나타냅니다. Effort to convert an occurrence of the action item 열의 값은 설계된 접근 방식에 따라 각 작업 항목을 변환하는 데 필요한 작업량을 나타냅니다. 필요한 노력의 수준을 나타내는 데 사용되는 값은 낮음(최소)에서 높음(최대)까지의 가중치 척도를 기반으로 합니다.

Schema	Action item	Number of occurrences	Learning curve efforts	Efforts to convert an occurrence of the action item
TEST.dbo	609	1	8	0.3
TEST.dbo	681	2	0.1	0.1
TEST.dbo	690	1	40	40
TEST.dbo	794	1	0	0.01
TEST.dbo	811	12	40	8
TEST.dbo	826	1	0	0.1
TEST.dbo	844	8	8	0.5
TEST.dbo	9997	3	0	0.3

세 번째 CSV 파일은 이름에 `Summary`가 포함되며 다음 정보를 포함하고 있습니다.

- 범주
- 객체 수
- 자동으로 변환된 객체
- 단순 작업이 포함된 객체
- 중간 복잡도의 작업이 포함된 객체
- 복잡한 작업이 포함된 객체
- 총 코드 줄 수

평가 보고서 구성

AWS SCT가 평가 보고서에 포함할 세부 정보의 양을 구성할 수 있습니다.

데이터베이스 마이그레이션 평가 보고서를 구성하려면

1. 설정 메뉴에서 전역 설정을 선택한 다음, 평가 보고서를 선택합니다.
2. 평가 보고서에서 단일 유형의 작업 항목 수를 제한하려면 Action item occurrences에서 First five issues only를 선택합니다. 평가 보고서에 각 유형의 모든 작업 항목을 포함하려면 모든 문제를 선택합니다.
3. 평가 보고서의 SQL 스크립트 파일 수를 *X*개로 제한하려면 SQL script analyzed files에서 List not more than ***X*** files를 선택합니다. 파일 수를 입력합니다. 평가 보고서에 모든 SQL 스크립트 파일을 포함하려면 List all analyzed files를 선택합니다.
4. 데이터베이스 마이그레이션 평가 보고서의 로컬 사본을 저장한 후 파일을 자동으로 열려면 Open reports after saving을 선택합니다. 자세한 내용은

데이터베이스 마이그레이션 평가 보고서를 만든 후 데이터베이스 마이그레이션 평가 보고서의 로컬 사본을 PDF 파일 또는 CSV(쉼표로 구분된 값) 파일로 저장할 수 있습니다.

데이터베이스 마이그레이션 평가 보고서를 PDF 파일로 저장하려면

1. 상단 메뉴에서 보기를 선택한 다음, Assessment report view를 선택합니다.
2. 요약 탭을 선택합니다.
3. 오른쪽 상단에서 Save to PDF를 선택합니다.

데이터베이스 마이그레이션 평가 보고서를 CSV 파일로 저장하려면

1. 상단 메뉴에서 보기를 선택한 다음, Assessment report view를 선택합니다.
2. 요약 탭을 선택합니다.
3. 오른쪽 상단에서 Save to CSV를 선택합니다.

PDF 파일에는 다음 예제와 같이 요약 및 작업 항목 정보가 모두 포함되어 있습니다.

Database objects with conversion actions for Amazon RDS for PostgreSQL

Of the total 585 database storage object(s) and 1,542 database code object(s) in the source database, we identifed 529 (90%) database storage object(s) and 1,194 (77%) database code object(s) that can be converted to Amazon RDS for PostgreSQL automatically or with minimal changes.

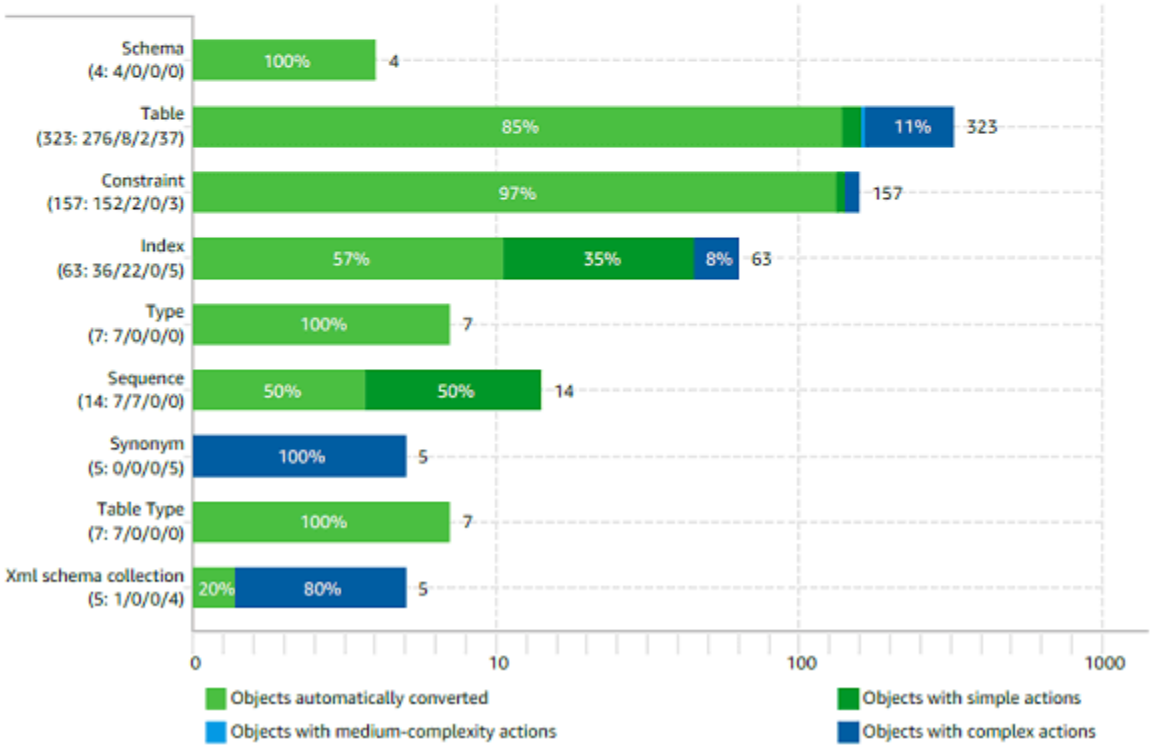
We found 7 encrypted object(s).

56 (10%) database storage object(s) require 100 complex user action(s) to complete the conversion.

348 (23%) database code object(s) require 6 medium and 965 complex user action(s) to complete the conversion.

The object actions complexity is a sum of the complexity of the action items associated with the object. Therefore, an object with multiple simple action items could be treated as "object with medium-complexity actions" or even as "object with complex actions."

Figure: Conversion statistics for database storage objects



Save to CSV 옵션을 선택하면 AWS SCT에서 세 개의 CSV 파일을 생성합니다.

첫 번째 CSV 파일에는 작업 항목에 대한 다음과 같은 정보가 포함됩니다.

- 범주
- 발생 - 항목의 파일 이름, 행 번호, 위치
- 작업 항목 번호
- 제목
- 그룹

- 설명

- 문서 참조
- 권장 조치
- 예상 복잡성

두 번째 CSV 파일은 이름에 `Action_Items_Summary` 접미사를 포함하며 모든 작업 항목의 발생 횟수에 대한 정보를 포함합니다.

다음 예제에서 Learning curve effort 열의 값은 각 작업 항목을 전환하는 방법을 설계하는 데 필요한 작업량을 나타냅니다. Effort to convert an occurrence of the action item 열의 값은 설계된 접근 방식에 따라 각 작업 항목을 변환하는 데 필요한 작업량을 나타냅니다. 필요한 노력의 수준을 나타내는 데 사용되는 값은 낮음(최소)에서 높음(최대)까지의 가중치 척도를 기반으로 합니다.

Schema	Action item	Number of occurrences	Learning curve efforts	Efforts to convert an occurrence of the action item
TEST.dbo	609	1	8	0.3
TEST.dbo	681	2	0.1	0.1
TEST.dbo	690	1	40	40
TEST.dbo	794	1	0	0.01
TEST.dbo	811	12	40	8
TEST.dbo	826	1	0	0.1
TEST.dbo	844	8	8	0.5
TEST.dbo	9997	3	0	0.3

세 번째 CSV 파일은 이름에 `Summary`가 포함되며 다음 정보를 포함하고 있습니다.

- 범주

- 객체 수
- 자동으로 변환된 객체
- 단순 작업이 포함된 객체
- 중간 복잡도의 작업이 포함된 객체
- 복잡한 작업이 포함된 객체
- 총 코드 줄 수

섹션을 참조하세요.

데이터베이스 마이그레이션을 위한 다중 서버 평가 보고서 생성

전체 환경에 가장 적합한 대상 방향을 결정하려면 다중 서버 평가 보고서를 생성합니다.

다중 서버 평가 보고서는 평가하려는 각 스키마 정의에 대해 제공된 입력 내용을 기반으로 여러 서버를 평가합니다. 스키마 정의에는 데이터베이스 서버 연결 파라미터와 각 스키마의 전체 이름이 포함됩니다. 각 스키마를 평가한 후 AWS SCT가 여러 서버 간의 데이터베이스 마이그레이션에 대한 요약된 집계 평가 보고서를 생성합니다. 이 보고서는 가능한 각 마이그레이션 대상의 예상 복잡성을 보여줍니다.

AWS SCT를 사용하여 다음 소스 및 대상 데이터베이스에 대한 다중 서버 평가 보고서를 생성할 수 있습니다.

원본 데이터베이스	대상 데이터베이스:
Amazon Redshift	Amazon Redshift
Azure SQL Database	Aurora MySQL, Aurora PostgreSQL, MySQL, PostgreSQL
Azure Synapse Analytics	Amazon Redshift
BigQuery	Amazon Redshift
Greenplum	Amazon Redshift
IBM Db2 for z/OS	Amazon Aurora MySQL-Compatible Edition(Aurora MySQL), Amazon Aurora PostgreSQL-Compatible Edition(Aurora PostgreSQL), MySQL, PostgreSQL
IBM Db2 LUW	Aurora MySQL, Aurora PostgreSQL, MariaDB, MySQL, PostgreSQL
Microsoft SQL Server	Aurora MySQL, Aurora PostgreSQL, Amazon Redshift, Babelfish for Aurora PostgreSQL, MariaDB, Microsoft SQL Server, MySQL, PostgreSQL
MySQL	Aurora PostgreSQL, MySQL, PostgreSQL

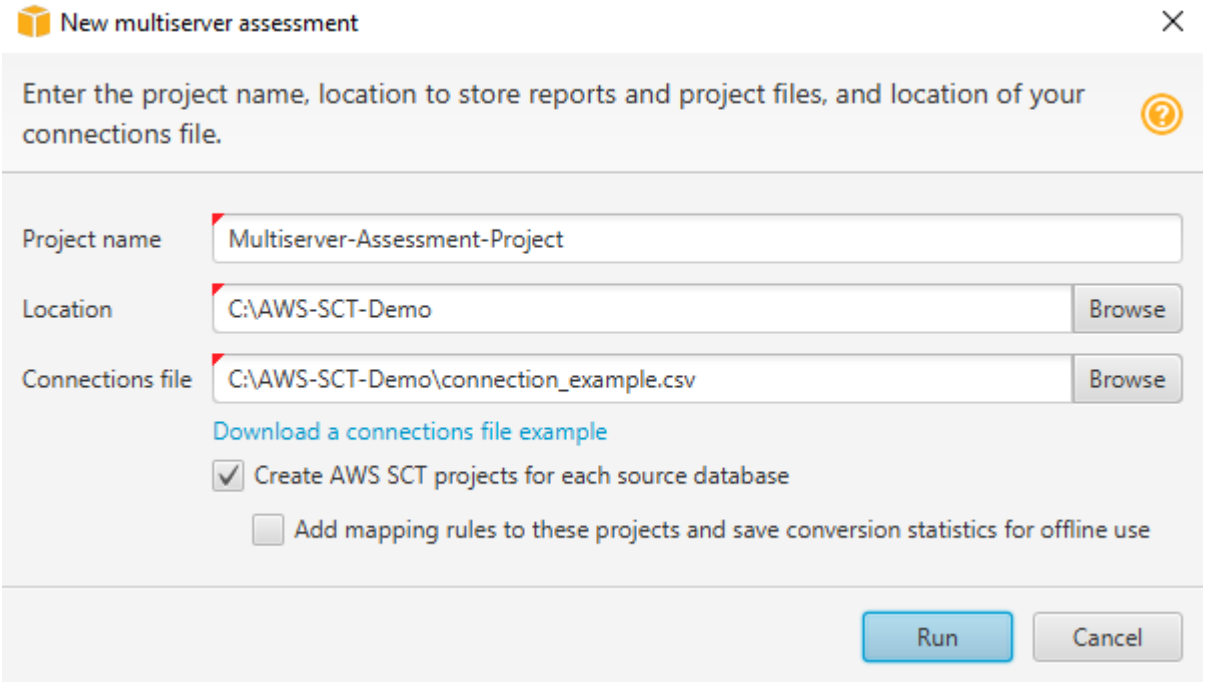
원본 데이터베이스	대상 데이터베이스:
Netezza	Amazon Redshift
Oracle	Aurora MySQL, Aurora PostgreSQL, Amazon Redshift, MariaDB, MySQL, Oracle, PostgreSQL
PostgreSQL	Aurora MySQL, Aurora PostgreSQL, MySQL, PostgreSQL
SAP ASE	Aurora MySQL, Aurora PostgreSQL, MariaDB, MySQL, PostgreSQL
Snowflake	Amazon Redshift
Teradata	Amazon Redshift
Vertica	Amazon Redshift

다중 서버 평가 수행

다음 절차에 따라 AWS SCT를 사용하여 다중 서버 평가를 수행합니다. 다중 서버 평가를 수행하기 위해 AWS SCT에서 새 프로젝트를 생성할 필요가 없습니다. 시작하기 전에 데이터베이스 연결 파라미터가 포함된 쉼표로 구분된 값(CSV) 파일이 준비되었는지 확인합니다. 또한 필요한 데이터베이스 드라이버를 모두 설치하고 AWS SCT 설정에서 드라이버 위치를 설정했는지 확인합니다. 자세한 내용은 필수 데이터베이스 드라이버 다운로드 섹션을 참조하세요.

다중 서버 평가를 수행하고 집계된 요약 보고서를 생성하려면

1. AWS SCT에서 파일, New multiserver assessment를 차례로 선택합니다. New multiserver assessment 대화 상자가 열립니다.



2. Download a connections file example을 선택하여 데이터베이스 연결 파라미터가 포함된 CSV 파일의 빈 템플릿을 다운로드합니다.
3. 프로젝트 이름, 위치(보고서 저장용), Connections file(CSV 파일)의 값을 입력합니다.
4. 평가 보고서를 생성한 후 마이그레이션 프로젝트를 자동으로 만들려면 Create AWS SCT projects for each source database를 선택합니다.
5. Create AWS SCT projects for each source database를 켠 상태에서 Add mapping rules to these projects and save conversion statistics for offline use를 선택할 수 있습니다. 이 경우 AWS SCT는 각 프로젝트에 매핑 규칙을 추가하고 소스 데이터베이스 메타데이터를 프로젝트에 저장합니다. 자세한 내용은 오프라인 모드에서 AWS SCT 실행 섹션을 참조하세요.
6. 실행을 선택합니다.

   데이터베이스 평가 속도를 나타내는 진행률 표시줄이 나타납니다. 대상 엔진 수는 평가 런타임에 영향을 미칠 수 있습니다.
7. 다음 메시지가 표시되면 예를 선택합니다. Full analysis of all Database servers may take some time. Do you want to proceed?

   다중 서버 평가 보고서가 생성되면 완료되었음을 나타내는 화면이 표시됩니다.
8. 집계된 요약 평가 보고서를 보려면 Open Report를 선택합니다.

기본적으로 AWS SCT에서는 모든 소스 데이터베이스에 대한 집계 보고서와 소스 데이터베이스의 각 스키마 이름에 대한 세부 평가 보고서를 생성합니다. 자세한 내용은 보고서 찾기 및 보기 섹션을 참조하세요.

Create AWS SCT projects for each source database 옵션을 켠 상태에서 AWS SCT는 각 소스 데이터베이스에 대해 빈 프로젝트를 생성합니다. 또한 AWS SCT는 이전에 설명한 대로 평가 보고서를 생성합니다. 이러한 평가 보고서를 분석하고 각 소스 데이터베이스의 마이그레이션 대상을 선택한 후 이러한 빈 프로젝트에 대상 데이터베이스를 추가합니다.

Add mapping rules to these projects and save conversion statistics for offline use 옵션을 켠 상태에서 AWS SCT는 각 소스 데이터베이스에 대한 프로젝트를 생성합니다. 이러한 프로젝트에는 다음 정보가 포함됩니다.

- 소스 데이터베이스 및 가상 대상 데이터베이스 플랫폼. 자세한 내용은 가상 대상 사용 섹션을 참조하세요.
- 이 소스–대상 쌍에 대한 매핑 규칙. 자세한 내용은 매핑 규칙 생성 섹션을 참조하세요.
- 이 소스-대상 쌍에 대한 데이터베이스 마이그레이션 평가 보고서
- 이 AWS SCT 프로젝트를 오프라인 모드에서 사용할 수 있게 해주는 소스 스키마 메타데이터. 자세한 내용은 오프라인 모드에서 AWS SCT 실행 섹션을 참조하세요.

입력 CSV 파일 준비

연결 파라미터를 다중 서버 평가 보고서의 입력으로 제공하려면 다음 예제와 같이 CSV 파일을 사용합니다.

```
Name,Description,Secret Manager Key,Server IP,Port,Service Name,Database name,BigQuery
 path,Source Engine,Schema Names,Use Windows Authentication,Login,Password,Use
 SSL,Trust store,Key store,SSL authentication,Target Engines
Sales,,,192.0.2.0,1521,pdb,,,ORACLE,Q4_2021;FY_2021,,user,password,,,,,POSTGRESQL;AURORA_POSTGR
Marketing,,,ec2-a-b-c-d.eu-
west-1.compute.amazonaws.com,1433,,target_audience,,MSSQL,customers.dbo,,user,password,,,,,AURO
HR,,,192.0.2.0,1433,,employees,,MSSQL,employees.%,true,,,,,,,AURORA_POSTGRESQL
Customers,,secret-name,,,,,,MYSQL,customers,,,,,,,,AURORA_POSTGRESQL
Analytics,,,198.51.100.0,8195,,STATISTICS,,DB2LUW,BI_REPORTS,,user,password,,,,,POSTGRESQL
Products,,,203.0.113.0,8194,,,,TERADATA,new_products,,user,password,,,,,REDSHIFT
```

위의 예에서는 세미콜론을 사용하여 `Sales` 데이터베이스의 두 스키마 이름을 구분합니다. 또한 세미콜론을 사용하여 `Sales` 데이터베이스의 두 대상 데이터베이스 마이그레이션 플랫폼을 구분합니다.

위의 예에서는 AWS Secrets Manager을 사용하여 `Customers` 데이터베이스에 연결하고 Windows 인증을 사용하여 HR 데이터베이스에 연결합니다.

새 CSV 파일을 만들거나 AWS SCT에서 CSV 파일용 템플릿을 다운로드하여 필요한 정보를 입력할 수 있습니다. CSV 파일의 첫 번째 행에 위의 예에 나온 것과 같은 열 이름이 포함되어 있는지 확인합니다.

입력 CSV 파일의 템플릿을 다운로드하려면

1. Start AWS SCT.
2. 파일을 선택한 다음, New multiserver assessment를 선택합니다.
3. Download a connections file example을 선택합니다.

템플릿에서 제공하는 다음과 같은 값이 CSV 파일에 포함되어 있는지 확인합니다.

- 이름 - 데이터베이스를 식별하는 데 도움이 되는 텍스트 레이블입니다. AWS SCT는 평가 보고서에 이 텍스트 레이블을 표시합니다.
- 설명 - 데이터베이스에 대한 추가 정보를 제공할 수 있는 선택적 값입니다.
- Secret Manager Key - AWS Secrets Manager에 데이터베이스 보안 인증 정보를 저장하는 보안 암호의 이름입니다. Secrets Manager를 사용하려면 AWS 프로필을 AWS SCT에 저장해야 합니다. 자세한 내용은 AWS Secrets Manager 사용 섹션을 참조하세요.

  > **Important**
  > 입력 파일에 서버 IP, 포트, 로그인 및 암호 파라미터가 포함된 경우 AWS SCT는 Secret Manager Key 파라미터를 무시합니다.

- 서버 IP – 소스 데이터베이스 서버의 DNS(Domain Name Service) 이름 또는 IP 주소입니다.
- 포트 - 소스 데이터베이스 서버에 연결하는 데 사용되는 포트입니다.
- 서비스 이름 - 서비스 이름을 사용하여 Oracle 데이터베이스에 연결하는 경우 연결할 Oracle 서비스의 이름입니다.
- 데이터베이스 이름 - 데이터베이스 이름입니다. Oracle 데이터베이스의 경우 Oracle System ID(SID)를 사용합니다.
- BigQuery path - 소스 BigQuery 데이터베이스의 서비스 계정 키 파일 경로입니다. 이 파일 생성에 대한 자세한 내용은 BigQuery를 소스로 사용하기 위한 권한 섹션을 참조하세요.
- 소스 엔진 - 소스 데이터베이스의 유형입니다. 다음 값 중 하나를 사용합니다.
  - AZURE_MSSQL - Azure SQL Database

- AZURE_SYNAPSE - Azure Synapse Analytics 데이터베이스
- GOOGLE_BIGQUERY - BigQuery 데이터베이스
- DB2ZOS - IBM Db2 for z/OS 데이터베이스
- DB2LUW - IBM Db2 LUW 데이터베이스
- GREENPLUM - Greenplum 데이터베이스
- MSSQL - Microsoft SQL Server 데이터베이스
- MYSQL - MySQL 데이터베이스
- NETEZZA - Netezza 데이터베이스
- ORACLE - Oracle 데이터베이스
- POSTGRESQL - PostgreSQL 데이터베이스
- REDSHIFT - Amazon Redshift 데이터베이스
- SNOWFLAKE - Snowflake 데이터베이스
- SYBASE_ASE - SAP ASE 데이터베이스
- TERADATA - Teradata 데이터베이스
- VERTICA - Vertica 데이터베이스

- 스키마 이름 - 평가 보고서에 포함할 데이터베이스 스키마의 이름입니다.

  Azure SQL Database, Azure Synapse Analytics, BigQuery, Netezza, SAP ASE, Snowflake 및 SQL Server의 경우 다음 형식의 스키마 이름을 사용합니다.

  db_name.*schema_name*

  *db_name*을 소스 데이터베이스의 이름으로 바꿉니다.

  *schema_name*을 소스 스키마의 이름으로 바꿉니다.

  `"database.name"."schema.name"`과 같이 점이 포함된 데이터베이스 또는 스키마 이름은 큰따옴표로 묶습니다.

  `Schema1;Schema2`와 같이 세미콜론을 사용하여 여러 스키마 이름을 구분합니다.

  데이터베이스 및 스키마 이름은 대/소문자를 구분합니다.

  데이터베이스 또는 스키마 이름에 있는 여러 기호를 바꾸려면 퍼센트(%)를 와일드카드로 사용합니다. 위 예제에서는 퍼센트(%)를 와일드카드로 사용하여 `employees` 데이터베이스의 모든 스키마를 평가 보고서에 포함시킵니다.

- Use Windows Authentication - Windows 인증을 사용하여 Microsoft SQL Server 데이터베이스에 연결하는 경우 true를 입력합니다. 자세한 내용은 Microsoft SQL Server를 소스로 사용할 때 Windows 인증 사용 섹션을 참조하세요.
- 로그인 – 소스 데이터베이스 서버에 연결하는 사용자 이름입니다.
- 암호 - 소스 데이터베이스 서버에 연결하는 데 필요한 암호입니다.
- Use SSL - SSL(Secure Sockets Layer)을 사용하여 소스 데이터베이스에 연결하는 경우 true를 입력합니다.
- 트러스트 스토어 - SSL 연결에 사용할 트러스트 스토어입니다.
- 키 스토어 - SSL 연결에 사용할 키 스토어입니다.
- SSL 인증 - 인증서를 통해 SSL 인증을 사용하는 경우 true를 입력합니다.
- Target Engines - 대상 데이터베이스 플랫폼입니다. 평가 보고서에서 하나 이상의 대상을 지정하려면 다음 값을 사용합니다.
  - AURORA_MYSQL - Aurora MySQL-Compatible 데이터베이스
  - AURORA_POSTGRESQL - Aurora PostgreSQL-Compatible 데이터베이스
  - BABELFISH - Babelfish for Aurora PostgreSQL 데이터베이스
  - MARIA_DB - MariaDB 데이터베이스
  - MSSQL - Microsoft SQL Server 데이터베이스
  - MYSQL - MySQL 데이터베이스
  - ORACLE - Oracle 데이터베이스
  - POSTGRESQL - PostgreSQL 데이터베이스
  - REDSHIFT - Amazon Redshift 데이터베이스

  `MYSQL;MARIA_DB`와 같이 세미콜론을 사용하여 여러 대상을 구분합니다. 대상 수는 평가를 실행하는 데 걸리는 시간에 영향을 줍니다.

보고서 찾기 및 보기

다중 서버 평가에서는 두 가지 유형의 보고서가 생성됩니다.

- 모든 소스 데이터베이스의 집계 보고서
- 소스 데이터베이스의 각 스키마 이름에 대한 대상 데이터베이스의 세부 평가 보고서

보고서는 New multiserver assessment 대화 상자의 위치에서 선택한 디렉터리에 저장됩니다.

세부 보고서에 액세스하려면 소스 데이터베이스, 스키마 이름 및 대상 데이터베이스 엔진별로 구성된 하위 디렉터리를 탐색할 수 있습니다.

집계 보고서에는 대상 데이터베이스의 변환 복잡도에 대한 정보가 네 개의 열로 표시됩니다. 해당 열에는 코드 객체, 스토리지 객체, 구문 요소의 변환 및 변환 복잡도에 대한 정보가 포함됩니다.

다음 예제에서는 두 Oracle 데이터베이스 스키마를 Amazon RDS for PostgreSQL로 변환하는 방법에 대한 정보를 보여줍니다.

Server IP address and port	Secret Manager key	Name	Description	Database name	Schema name	Code object conversion % for "Amazon RDS for PostgreSQL"	Storage object conversion % for "Amazon RDS for PostgreSQL"	Syntax Elements conversion % for "Amazon RDS for PostgreSQL"	Conversion Complexity for "Amazon RDS for PostgreSQL"
192.0.2.0:1521		Sales		ORCL	Q4_2021	97.78%	100.00%	98.76%	1
192.0.2.0:1521		Sales		pdb	FY_2021	82.35%	85.19%	99.24%	10

지정된 각각의 추가 대상 데이터베이스 엔진에 대해 동일한 네 개의 열이 보고서에 추가됩니다.

이 정보를 읽는 방법에 대한 자세한 내용은 다음을 참조하세요.

집계된 평가 보고서의 출력

AWS Schema Conversion Tool의 집계된 다중 서버 데이터베이스 마이그레이션 평가 보고서는 다음 열이 포함된 CSV 파일입니다.

- `Server IP address and port`
- `Secret Manager key`
- `Name`
- `Description`
- `Database name`
- `Schema name`
- `Code object conversion % for` *`target_database`*
- `Storage object conversion % for` *`target_database`*
- `Syntax elements conversion % for` *`target_database`*
- `Conversion complexity for` *`target_database`*

정보를 수집하기 위해 AWS SCT에서는 전체 평가 보고서를 실행한 다음 스키마별로 보고서를 집계합니다.

보고서에서 다음 세 필드는 평가를 기반으로 가능한 자동 변환 비율을 나타냅니다.

Code object conversion %

AWS SCT가 스키마에서 자동으로 변환하거나 최소한의 변경으로 변환할 수 있는 코드 객체의 비율입니다. 코드 객체에는 프로시저, 함수, 보기 등이 포함됩니다.

Storage object conversion %

SCT가 자동으로 변환하거나 최소한의 변경으로 변환할 수 있는 스토리지 객체의 비율입니다. 스토리지 객체에는 테이블, 인덱스, 제약 조건 등이 포함됩니다.

Syntax elements conversion %

SCT가 자동으로 변환할 수 있는 구문 요소의 비율입니다. 구문 요소에는 `SELECT`, `FROM`, `DELETE` 및 `JOIN` 절 등이 포함됩니다.

변환 복잡도 계산은 작업 항목의 개념을 기반으로 합니다. 작업 항목은 특정 대상으로 마이그레이션하는 동안 수동으로 수정해야 하는 소스 코드의 문제 유형을 나타냅니다. 작업 항목은 여러 번 발생할 수 있습니다.

가중치 척도는 마이그레이션 수행의 복잡도를 나타냅니다. 숫자 1은 가장 낮은 복잡도를 나타내고 숫자 10은 가장 높은 복잡도를 나타냅니다.

AWS SCT를 사용하여 데이터베이스 스키마 변환

AWS Schema Conversion Tool(AWS SCT)을 사용하여 기존 데이터베이스 스키마를 하나의 데이터베이스 엔진에서 다른 데이터베이스 엔진으로 변환할 수 있습니다. AWS SCT 사용자 인터페이스를 사용하여 데이터베이스를 변환하는 작업은 매우 간단할 수 있지만 변환을 수행하기 전에 고려해야 할 몇 가지 사항이 있습니다.

예를 들면, 다음을 수행하기 위해 AWS SCT를 사용할 수 있습니다.

- AWS SCT를 사용하면 동일한 엔진을 실행하는 Amazon RDS DB 인스턴스로 기존 온프레미스 데이터베이스 스키마를 복사할 수 있습니다. 이 기능을 사용하면 클라우드로 이전하고 라이선스 유형을 변경하는 데 따르는 비용 절감의 가능성을 분석할 수 있습니다.
- 데이터베이스 기능을 동등한 Amazon RDS 기능으로 변환할 수 없는 경우도 있습니다. Amazon Elastic Compute Cloud(Amazon EC2)에서 데이터베이스를 호스팅하고 자체 관리하는 경우 AWS 서비스를 대체하여 이러한 기능을 에뮬레이션할 수 있습니다.
- AWS SCT는 온라인 트랜잭션 처리(OLTP) 데이터베이스 스키마를 Amazon Relational Database Service(RDS) MySQL DB 인스턴스, Amazon Aurora DB 클러스터 또는 PostgreSQL DB 인스턴스로 변환하는 대부분의 프로세스를 자동화합니다. 소스 및 대상 데이터베이스 엔진에는 많은 특징과 기능이 포함되어 있으며 AWS SCT는 가능하다면 Amazon RDS DB 인스턴스에서 동일한 스키마를 생성하려고 합니다. 직접 변환이 불가능한 경우 AWS SCT는 수행할 수 있는 작업 목록을 제공합니다.

주제

- AWS SCT에서 마이그레이션 규칙 생성
- AWS SCT를 사용하여 스키마 변환
- AWS SCT에서 수동 변환 처리
- AWS SCT에서 변환된 스키마 업데이트 및 새로 고침
- AWS SCT에서 변환된 스키마 저장 및 적용
- 데이터베이스 스키마 비교
- 관련 변환된 객체 찾기

AWS SCT는 다음과 같은 온라인 트랜잭션 처리(OLTP) 변환을 지원합니다.

원본 데이터베이스	대상 데이터베이스:
IBM Db2 for z/OS(버전 12)	Amazon Aurora MySQL-Compatible Edition, Amazon Aurora PostgreSQL-Compatible Edition, MySQL, PostgreSQL
IBM Db2 LUW(버전 9.1, 9.5, 9.7, 10.5, 11.1, 11.5)	Aurora MySQL, Aurora PostgreSQL, MariaDB, MySQL, PostgreSQL
Microsoft Azure SQL Database	Aurora MySQL, Aurora PostgreSQL, MySQL, PostgreSQL
Microsoft SQL Server(버전 2008 R2 이상)	Aurora MySQL, Aurora PostgreSQL, Babelfish for Aurora PostgreSQL, MariaDB, Microsoft SQL Server, MySQL, PostgreSQL
MySQL(버전 5.5 이상)	Aurora PostgreSQL, MySQL, PostgreSQL AWS SCT를 사용하지 않고 MySQL의 스키마 및 데이터를 Aurora MySQL DB 클러스터로 마이그레이션할 수 있습니다. 자세한 내용은 Amazon Aurora DB 클러스터로 데이터 마이그레이션을 참조하세요.
Oracle(버전 10.2 이상)	Aurora MySQL, Aurora PostgreSQL, MariaDB, MySQL, Oracle, PostgreSQL
PostgreSQL(버전 9.1 이상)	Aurora MySQL, Aurora PostgreSQL, MySQL, PostgreSQL
SAP ASE(12.5, 15.0, 15.5, 15.7, 16.0)	Aurora MySQL, Aurora PostgreSQL, MariaDB, MySQL, PostgreSQL

데이터 웨어하우스 스키마 변환에 대한 자세한 내용은 AWS SCT를 사용하여 데이터 웨어하우스 스키마를 Amazon Redshift로 변환 섹션을 참조하세요.

데이터베이스 스키마를 Amazon RDS로 변환하려면 다음과 같은 개략적인 단계를 수행합니다.

- AWS SCT에서 마이그레이션 규칙 생성 – 이제 AWS SCT를 사용하여 스키마를 변환하기 전에 열의 데이터 형식을 변경하고 객체를 한 스키마에서 다른 스키마로 이동하고 객체 이름을 변경하는 규칙을 설정할 수 있습니다.
- AWS SCT를 사용하여 스키마 변환 – AWS SCT에서 사용자가 검토할 수 있도록 변환된 스키마의 로컬 버전을 생성하지만 준비가 될 때까지는 대상 DB 인스턴스에 적용되지 않습니다.
- AWS SCT를 사용하여 마이그레이션 평가 보고서 작성 – AWS SCT는 자동으로 변환할 수 없는 스키마 요소를 자세히 설명하는 데이터베이스 마이그레이션 평가 보고서를 생성합니다. 이 보고서를 사용하면 Amazon RDS DB 인스턴스에서 소스 데이터베이스와 호환되는 스키마를 만들어야 하는 위치를 식별할 수 있습니다.
- AWS SCT에서 수동 변환 처리 – 자동으로 변환할 수 없는 스키마 요소가 있는 경우 소스 스키마를 업데이트한 다음 다시 변환하거나, 대상 Amazon RDS DB 인스턴스에서 동일한 스키마 요소를 생성할 수 있습니다.
- AWS SCT에서 변환된 스키마 업데이트 및 새로 고침 - 소스 데이터베이스의 최신 스키마를 사용하여 AWS SCT 프로젝트를 업데이트할 수 있습니다.
- AWS SCT에서 변환된 스키마 저장 및 적용 - 준비가 되면 AWS SCT가 로컬 프로젝트의 변환된 스키마를 대상 Amazon RDS DB 인스턴스에 적용합니다.

AWS SCT에서 마이그레이션 규칙 생성

AWS SCT를 사용하여 스키마를 변환하기 전에 마이그레이션 규칙을 설정할 수 있습니다. AWS SCT에서 마이그레이션 규칙은 열의 데이터 유형 변경, 한 스키마에서 다른 스키마로 객체 이동, 객체 이름 변경 등의 변환 작업을 수행할 수 있습니다. 예를 들어 소스 스키마에 `test_TABLE_NAME`이라는 테이블 세트가 있다고 가정해 봅니다. 대상 스키마의 접두사 `test_`를 접두사 `demo_`로 변경하는 규칙을 설정할 수 있습니다.

> **Note**
>
> 서로 다른 소스 및 대상 데이터베이스 엔진에 대한 마이그레이션 규칙만 생성할 수 있습니다.

다음 작업을 수행하는 마이그레이션 규칙을 생성할 수 있습니다.

- 접두사 추가, 제거 또는 교체
- 접미사 추가, 제거 또는 교체
- 열 데이터 정렬 변경

- 데이터 유형 변경
- char, varchar, nvarchar 및 string 데이터 유형의 길이 변경
- 객체 이동
- 객체 이름 변경

다음 객체에 대한 마이그레이션 규칙을 생성할 수 있습니다.

- 데이터베이스
- Schema
- 표
- 열

마이그레이션 규칙 생성

마이그레이션 규칙을 생성하고 규칙을 프로젝트의 일부로 저장할 수 있습니다. 프로젝트를 연 상태에서 다음 절차를 사용하여 마이그레이션 규칙을 생성합니다.

마이그레이션 규칙을 생성하려면

1. 보기 메뉴에서 매핑 보기를 선택합니다.
2. Server mappings에서 소스 및 대상 서버 쌍을 선택합니다.
3. New migration rule을 선택합니다. 변환 규칙 대화 상자가 나타납니다.
4. 새 규칙 추가를 선택합니다. 규칙 목록에 새 행이 추가됩니다.
5. 규칙을 구성합니다.

    a. Name(이름)에 규칙의 이름을 입력합니다.

    b. For에서 규칙이 적용되는 객체 유형을 선택합니다.

    c. where에서 마이그레이션 규칙을 적용하기 전에 객체에 적용할 필터를 입력합니다. where 절은 like 절을 사용하여 평가됩니다. 정확한 이름을 입력하여 하나의 객체를 선택하거나 패턴을 입력하여 여러 객체를 선택할 수 있습니다.

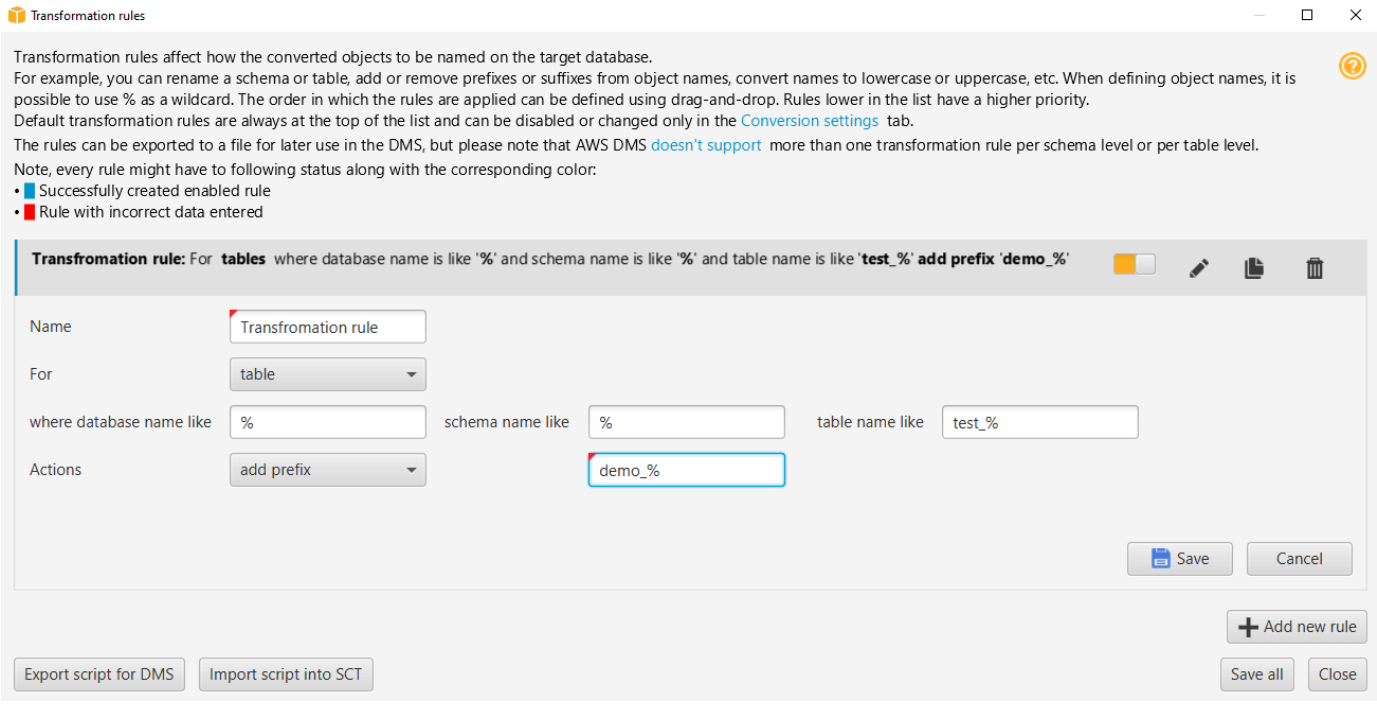
    where 절에 사용할 수 있는 필드는 객체 유형에 따라 다릅니다. 예를 들어, 객체 유형이 스키마인 경우 스키마 이름에 사용할 수 있는 필드는 하나뿐입니다.

    d. 작업에서 생성하려는 마이그레이션 규칙 유형을 선택합니다.

e. 규칙 유형에 따라 하나 또는 두 개의 추가 값을 입력합니다. 예를 들어, 객체의 이름을 바꾸려면 객체의 새 이름을 입력합니다. 접두사를 바꾸려면 이전 접두사와 새 접두사를 입력합니다.

   char, varchar, nvarchar 및 문자열 데이터 유형의 경우 곱셈 연산자를 사용하여 데이터 유형 길이를 변경할 수 있습니다. 예를 들어, `%*4` 값은 `varchar(10)` 데이터 유형을 `varchar(40)`으로 변환합니다.

6. 마이그레이션 규칙을 구성한 후 저장을 선택하여 규칙을 저장합니다. 취소를 선택하여 변경 사항을 취소할 수도 있습니다.



7. 규칙 추가, 편집 및 삭제를 완료한 후 모두 저장을 선택하여 변경 내용을 모두 저장합니다.
8. 닫기를 선택하여 변환 규칙 대화 상자를 닫습니다.

토글 아이콘을 사용하면 마이그레이션 규칙을 삭제하지 않고 끌 수 있습니다. 복사 아이콘을 사용하면 기존 마이그레이션 규칙을 복제할 수 있습니다. 연필 아이콘을 사용하면 기존 마이그레이션 규칙을 편집할 수 있습니다. 삭제 아이콘을 사용하면 기존 마이그레이션 규칙을 삭제할 수 있습니다. 마이그레이션 규칙 변경 내용을 저장하려면 모두 저장을 선택합니다.

마이그레이션 규칙 내보내기

AWS DMS를 사용하여 소스 데이터베이스의 데이터를 대상 데이터베이스로 마이그레이션하는 경우 마이그레이션 규칙에 대한 정보를 AWS DMS에 제공할 수 있습니다. 작업에 대한 자세한 내용은 AWS Database Migration Service 복제 작업 사용 섹션을 참조하세요.

마이그레이션 규칙을 내보내려면

1. AWS Schema Conversion Tool의 보기 메뉴에서 매핑 보기를 선택합니다.
2. Migration rules에서 마이그레이션 규칙을 선택한 다음, Modify migration rule를 선택합니다.
3. Export script for AWS DMS를 선택합니다.
4. 스크립트를 저장할 위치로 이동한 다음 저장을 선택합니다. 마이그레이션 규칙은 AWS DMS에서 사용할 수 있는 JSON 스크립트로 저장됩니다.

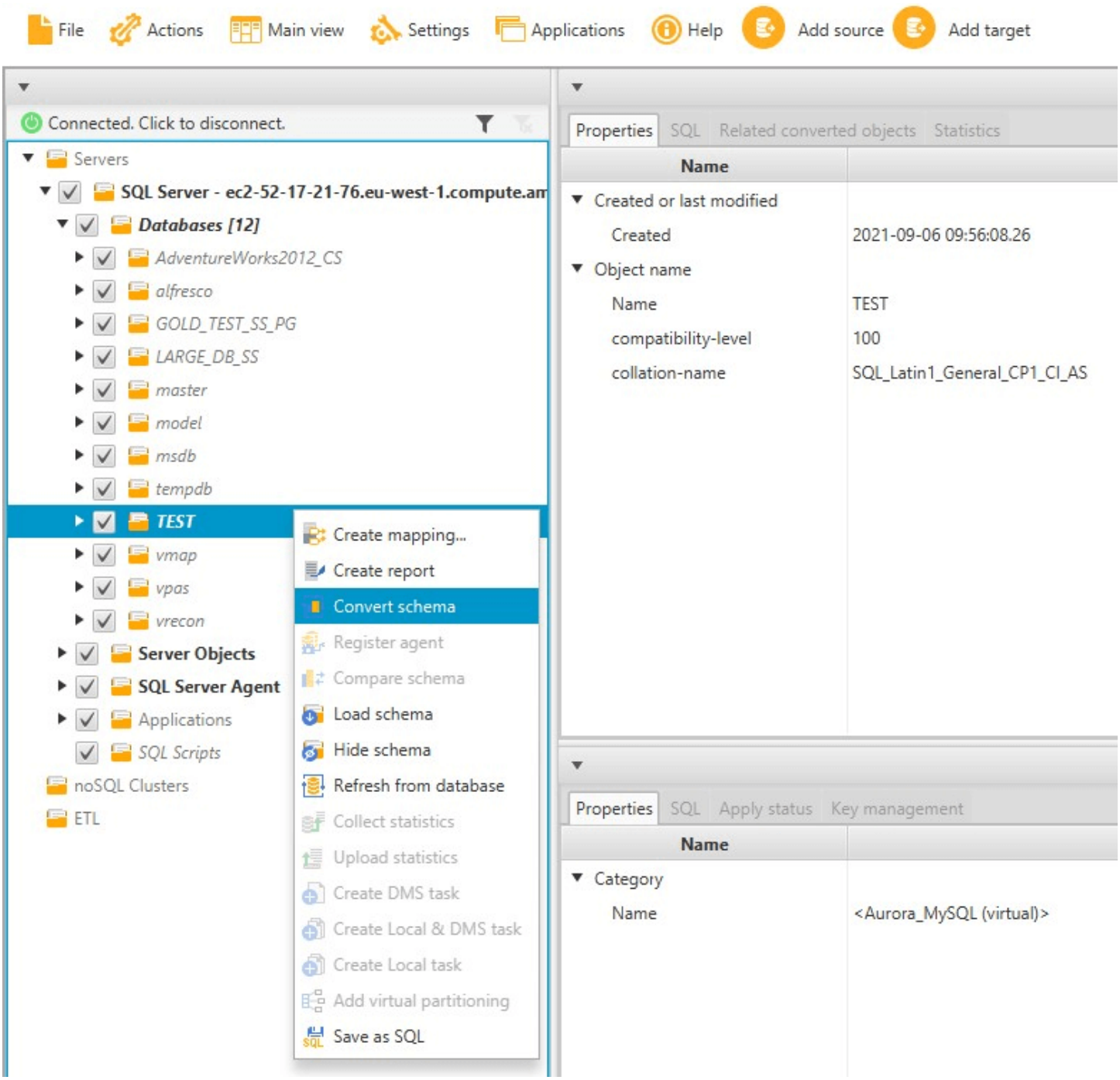
AWS SCT를 사용하여 스키마 변환

프로젝트를 소스 데이터베이스와 대상 Amazon RDS DB 인스턴스에 모두 연결하면 AWS Schema Conversion Tool 프로젝트가 소스 데이터베이스의 스키마를 왼쪽 패널에 표시합니다. 스키마는 트리 보기 형식으로 표시되며 트리의 각 노드는 지연 로드됩니다. 트리 보기에서 노드를 선택하면 이때 AWS SCT가 소스 데이터베이스의 스키마 정보를 요청합니다.

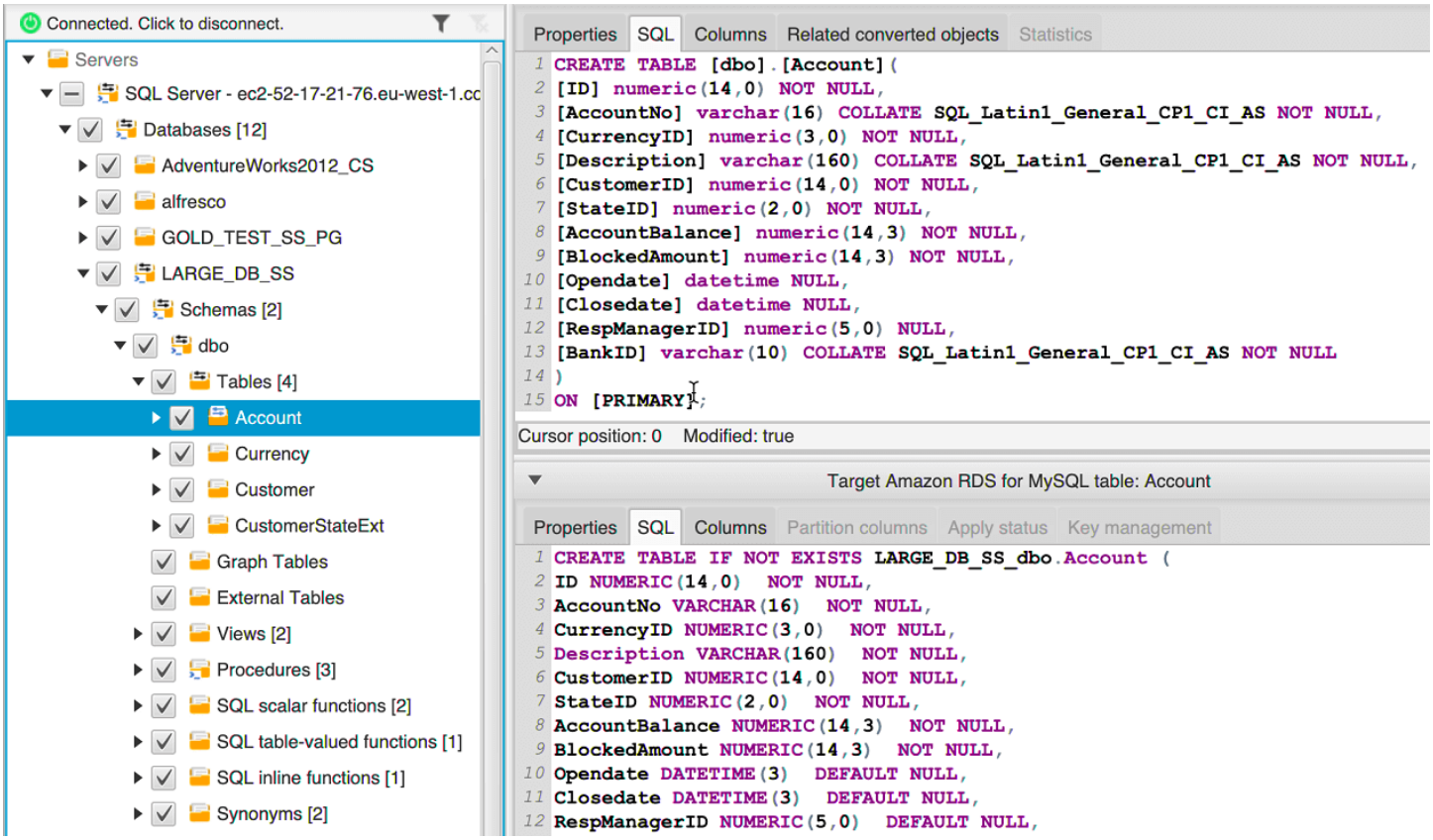
소스 데이터베이스에서 스키마 항목을 선택한 다음 해당 스키마를 대상 DB 인스턴스의 DB 엔진에 해당하는 동일한 스키마로 변환할 수 있습니다. 소스 데이터베이스에서 변환할 스키마 항목을 선택할 수 있습니다. 선택한 스키마 항목이 상위 항목에 따라 달라지는 경우, AWS SCT는 상위 항목에 대한 스키마도 생성합니다. 예를 들어, 변환할 테이블을 선택한다고 가정해 봅니다. 그럴 경우 AWS SCT는 테이블 및 테이블이 속한 데이터베이스에 대한 스키마를 생성합니다.

스키마 변환

소스 데이터베이스의 스키마를 변환하려면 변환할 스키마 이름의 확인란을 선택합니다. 그런 다음 프로젝트의 왼쪽 패널에서 이 스키마를 선택합니다. 그러면 AWS SCT가 스키마 이름을 파란색으로 강조 표시합니다. 스키마의 컨텍스트(마우스 오른쪽 버튼 클릭) 메뉴를 열고 아래와 같이 스키마 변환을 선택합니다.



소스 데이터베이스에서 스키마를 변환한 후 프로젝트의 왼쪽 패널에서 스키마 항목을 선택하고 프로젝트의 중앙 패널에서 변환된 스키마를 볼 수 있습니다. 하단 중앙 패널에는 다음과 같이 변환된 스키마를 생성하기 위한 속성 및 SQL 명령이 표시됩니다.



스키마를 변환한 후 프로젝트를 저장할 수 있습니다. 소스 데이터베이스의 스키마 정보는 프로젝트와 함께 저장됩니다. 이 기능을 사용하면 소스 데이터베이스에 연결하지 않고도 오프라인으로 작업할 수 있습니다. 소스 데이터베이스에 대해 데이터베이스에서 새로 고침을 선택하면 AWS SCT가 소스 데이터베이스에 연결하여 프로젝트의 스키마를 업데이트합니다. 자세한 내용은 AWS SCT에서 변환된 스키마 업데이트 및 새로 고침 섹션을 참조하세요.

자동으로 변환할 수 없는 항목의 데이터베이스 마이그레이션 평가 보고서를 생성할 수 있습니다. 평가 보고서는 자동으로 변환할 수 없는 스키마 항목을 식별하고 처리하는 데 유용합니다. 자세한 내용은 AWS SCT를 사용하여 마이그레이션 평가 보고서 작성 섹션을 참조하세요.

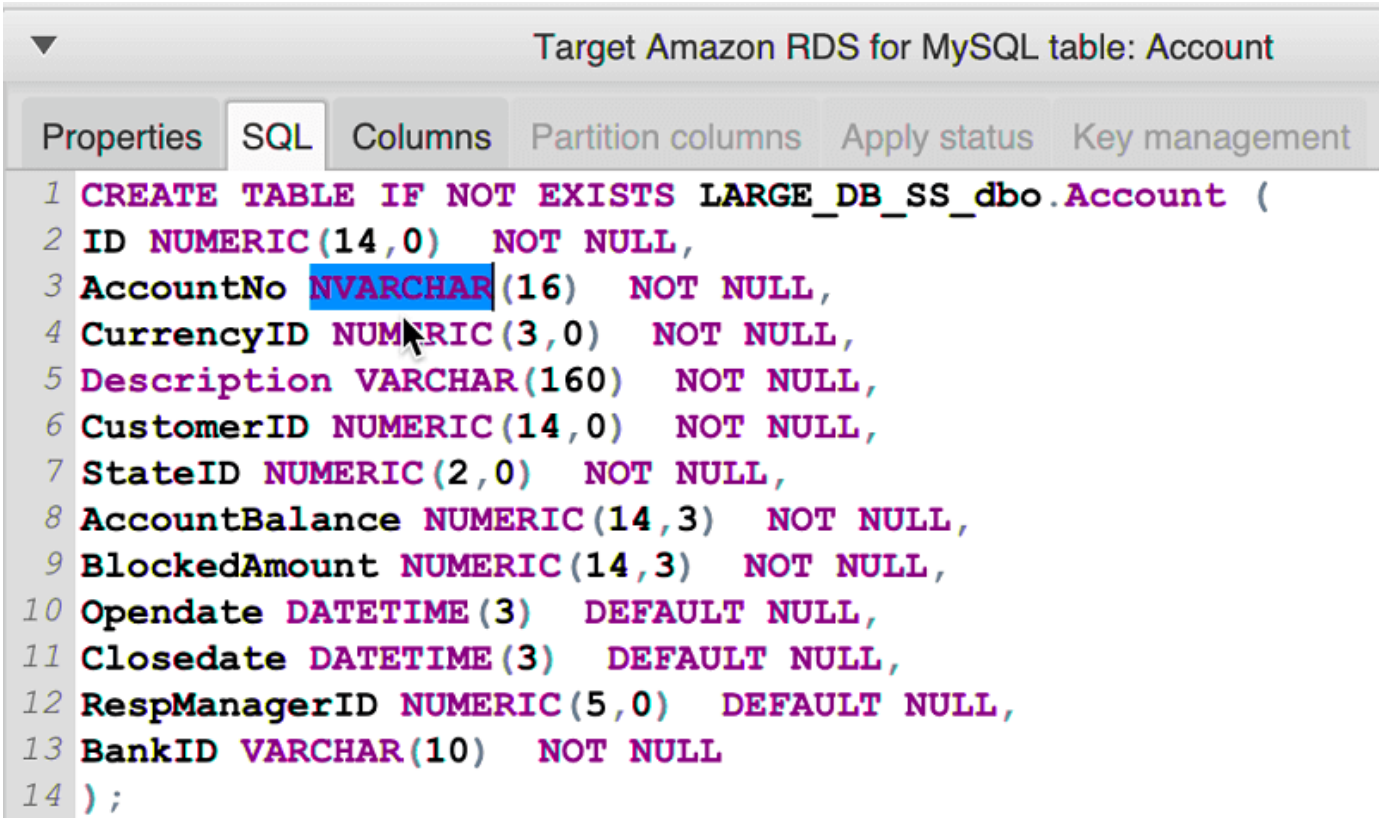
AWS SCT에서 변환된 스키마를 생성할 경우 대상 DB 인스턴스에 즉시 적용하지 않습니다. 대신 대상 DB 인스턴스에 적용할 준비가 될 때까지 변환된 스키마를 로컬에 저장합니다. 자세한 내용은 변환된 스키마 적용 섹션을 참조하세요.

변환된 스키마 편집

변환된 스키마를 편집하고 변경 내용을 프로젝트의 일부로 저장할 수 있습니다.

변환된 스키마를 편집하려면

1. 소스 데이터베이스의 스키마를 표시하는 왼쪽 패널에서 변환된 스키마를 편집할 스키마 항목을 선택합니다.
2. 선택한 항목에 대해 변환된 스키마를 표시하는 하단 중앙 패널에서 SQL 탭을 선택합니다.
3. SQL 탭에 표시된 텍스트에서 필요에 따라 스키마를 변경합니다. 프로젝트를 업데이트하면 스키마가 프로젝트에 자동으로 저장됩니다.



변환된 스키마에 대한 변경 내용은 업데이트 시 프로젝트와 함께 저장됩니다. 소스 데이터베이스에서 스키마 항목을 새로 변환하고 해당 항목에 대해 이전에 변환된 스키마로 업데이트한 경우 해당하는 기존 업데이트는 소스 데이터베이스를 기반으로 새로 변환된 스키마 항목으로 대체됩니다.

변환된 스키마 지우기

대상 DB 인스턴스에 스키마를 적용할 때까지 AWS SCT는 변환된 스키마를 프로젝트에 로컬로만 저장합니다. DB 인스턴스의 트리 보기 노드를 선택한 다음 데이터베이스에서 새로 고침을 선택하면 프로젝트에서 계획된 스키마를 지울 수 있습니다. 대상 DB 인스턴스에 스키마가 기록되지 않았으므로 데이터베이스를 새로 고치면 AWS SCT 프로젝트에서 계획된 스키마 요소가 제거되어 소스 DB 인스턴스에 있는 것과 일치하게 됩니다.

AWS SCT에서 수동 변환 처리

평가 보고서에는 대상 Amazon RDS DB 인스턴스의 데이터베이스 엔진으로 자동 변환할 수 없는 항목 목록이 포함되어 있습니다. 변환할 수 없는 각 항목의 경우 작업 항목 탭에는 작업 항목이 포함되어 있습니다.

다음과 같은 방법으로 평가 보고서의 작업 항목에 응답할 수 있습니다.

- 소스 데이터베이스 스키마 수정
- 대상 데이터베이스 스키마 수정

소스 스키마 수정

일부 항목의 경우 소스 데이터베이스의 데이터베이스 스키마를 자동으로 변환할 수 있는 스키마로 수정하는 것이 훨씬 쉬울 수 있습니다. 먼저 새로운 변경 내용이 애플리케이션 아키텍처와 호환되는지 확인하고 소스 데이터베이스의 스키마를 업데이트합니다. 마지막으로 업데이트된 스키마 정보로 프로젝트를 새로 고칩니다. 그런 다음 업데이트된 스키마를 변환하고 새 데이터베이스 마이그레이션 평가 보고서를 생성할 수 있습니다. 소스 스키마에서 변경된 항목에 대해서는 더 이상 작업 항목이 표시되지 않습니다.

이 프로세스의 장점은 소스 데이터베이스를 새로 고칠 때 항상 업데이트된 스키마를 사용할 수 있다는 것입니다.

대상 스키마 수정

일부 항목의 경우 변환된 스키마를 대상 데이터베이스에 적용한 다음 자동으로 변환할 수 없는 항목에 대해서는 대상 데이터베이스에 동일한 스키마 항목을 수동으로 추가하는 것이 훨씬 쉬울 수 있습니다. 스키마를 적용하여 대상 DB 인스턴스로 자동 변환할 수 있는 모든 스키마를 작성할 수 있습니다. 자세한 내용은 AWS SCT에서 변환된 스키마 저장 및 적용 섹션을 참조하세요.

대상 DB 인스턴스에 작성되는 스키마에는 자동으로 변환할 수 없는 항목이 포함되어 있지 않습니다. 대상 DB 인스턴스에 스키마를 적용한 후 소스 데이터베이스의 스키마와 동일한 스키마를 대상 DB 인스턴스에서 수동으로 생성할 수 있습니다. 데이터베이스 마이그레이션 평가 보고서의 작업 항목에는 동일한 스키마를 생성하는 방법에 대한 제안 사항이 포함되어 있습니다.

> **Warning**
>
> 대상 DB 인스턴스에서 스키마를 수동으로 생성하는 경우 수행하는 모든 수동 작업의 사본을 저장합니다. 프로젝트에서 변환된 스키마를 대상 DB 인스턴스에 다시 적용하면 수행한 수동 작업을 덮어씁니다.

경우에 따라서는 대상 DB 인스턴스에 동일한 스키마를 생성할 수 없습니다. 대상 DB 인스턴스의 DB 엔진에서 사용할 수 있는 기능을 사용하려면 애플리케이션과 데이터베이스의 일부를 다시 설계해야 할 수 있습니다. 자동으로 변환할 수 없는 스키마를 그냥 무시해도 되는 경우도 있습니다.

AWS SCT에서 변환된 스키마 업데이트 및 새로 고침

AWS Schema Conversion Tool 프로젝트에서 소스 스키마와 대상 스키마를 모두 업데이트할 수 있습니다.

- 소스 – 소스 데이터베이스의 스키마를 업데이트하면 AWS SCT가 프로젝트의 스키마를 소스 데이터베이스의 최신 스키마로 바꿉니다. 이 기능을 사용하면 소스 데이터베이스의 스키마가 변경된 경우 프로젝트를 업데이트할 수 있습니다.
- 대상 – 대상 Amazon RDS DB 인스턴스의 스키마를 업데이트하면 AWS SCT가 프로젝트의 스키마를 대상 DB 인스턴스의 최신 스키마로 바꿉니다. 대상 DB 인스턴스에 스키마를 적용하지 않은 경우에는 AWS SCT가 변환된 스키마를 프로젝트에서 지웁니다. 그런 다음 소스 데이터베이스의 스키마를 새로운 대상 DB 인스턴스로 변환할 수 있습니다.

데이터베이스에서 새로 고침을 선택하여 AWS SCT 프로젝트의 스키마를 업데이트할 수 있습니다.

> **Note**
>
> 스키마를 새로 고치면 AWS SCT가 필요할 때만 메타데이터를 로드합니다. 데이터베이스의 모든 스키마를 완전히 로드하려면 스키마의 컨텍스트(마우스 오른쪽 버튼 클릭) 메뉴를 열고 Load schema를 선택합니다. 예를 들어 이 옵션을 사용하여 데이터베이스의 모든 메타데이터를 한꺼번에 로드한 후 오프라인으로 작업할 수 있습니다.

AWS SCT에서 변환된 스키마 저장 및 적용

AWS Schema Conversion Tool에서 변환된 스키마를 생성할 경우(AWS SCT를 사용하여 스키마 변환 참조) 변환된 스키마를 대상 DB 인스턴스에 즉시 적용하지 않습니다. 대신 대상 DB 인스턴스에 적용할 준비가 될 때까지 변환된 스키마를 프로젝트에 로컬로 저장합니다. 이 기능을 사용하면 대상 DB 엔진으로 자동 변환할 수 없는 스키마 항목으로 작업할 수 있습니다. 자동으로 변환할 수 없는 항목에 대한 자세한 내용은 AWS SCT를 사용하여 마이그레이션 평가 보고서 작성 섹션을 참조하세요.

선택적으로 대상 DB 인스턴스에 스키마를 적용하기 전에 도구에서 변환된 스키마를 SQL 스크립트로 파일에 저장하도록 할 수 있습니다. 또한 도구에서 변환된 스키마를 대상 DB 인스턴스에 직접 적용하도록 할 수도 있습니다.

변환된 스키마를 파일에 저장

변환된 스키마를 텍스트 파일에 SQL 스크립트로 저장할 수 있습니다. 이 방법을 사용하면 AWS SCT에서 생성된 SQL 스크립트를 수정하여 도구가 자동으로 변환할 수 없는 항목을 처리할 수 있습니다. 그런 다음, 대상 DB 인스턴스에서 업데이트된 스크립트를 실행하여 변환된 스키마를 대상 데이터베이스에 적용할 수 있습니다.

변환된 스키마를 SQL 스크립트로 저장하려면

1. 스키마를 선택하고 컨텍스트(마우스 오른쪽 버튼 클릭) 메뉴를 엽니다.
2. SQL로 저장을 선택합니다.
3. 파일 이름을 입력하고 저장을 선택합니다.
4. 다음 옵션 중 하나를 사용하여 변환된 스키마를 저장합니다.

    - 단일 파일
    - Single file per stage
    - Single file per statement

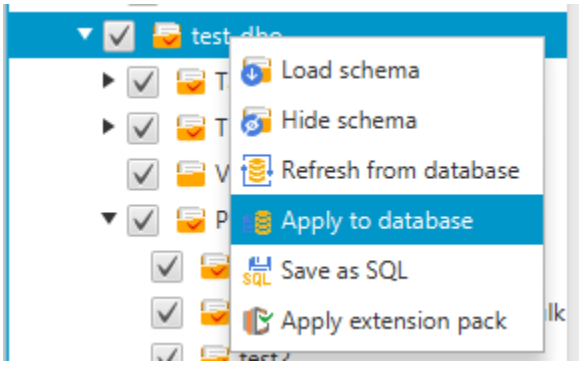
SQL 스크립트의 형식을 선택하려면

1. 설정 메뉴에서 프로젝트 설정을 선택합니다.
2. Save scripts를 선택합니다.
3. 공급업체에서 데이터베이스 플랫폼을 선택합니다.
4. Save SQL scripts to에서 데이터베이스 스키마 스크립트를 저장할 방법을 선택합니다.

5. 확인을 선택하여 설정을 저장합니다.

변환된 스키마 적용

변환된 스키마를 대상 Amazon RDS DB 인스턴스에 적용할 준비가 되면 프로젝트의 오른쪽 패널에서 스키마 요소를 선택합니다. 스키마 요소의 컨텍스트(마우스 오른쪽 버튼 클릭) 메뉴를 열고 아래와 같이 Apply to database를 선택합니다.



확장 팩 스키마

변환된 스키마를 대상 DB 인스턴스에 처음 적용하면 AWS SCT가 대상 DB 인스턴스에 추가적인 스키마를 추가합니다. 이 스키마는 변환된 스키마를 대상 DB 인스턴스에 쓸 때 필요한 소스 데이터베이스의 시스템 함수를 구현합니다. 이 스키마를 확장 팩 스키마라고 합니다.

확장 팩 스키마를 수정하지 않도록 합니다. 그렇지 않으면 대상 DB 인스턴스에 작성된 변환된 스키마에서 예상치 못한 결과가 발생할 수 있습니다. 스키마가 대상 DB 인스턴스로 완전히 마이그레이션되어 AWS SCT가 더 이상 필요하지 않은 경우 확장 팩 스키마를 삭제할 수 있습니다.

확장 팩 스키마는 소스 데이터베이스에 따라 다음과 같이 이름이 지정됩니다.

- IBM Db2 LUW: `aws_db2_ext`
- Microsoft SQL Server: `aws_sqlserver_ext`
- MySQL: `aws_mysql_ext`
- Oracle: `aws_oracle_ext`
- PostgreSQL: `aws_postgresql_ext`
- SAP ASE: `aws_sapase_ext`

자세한 내용은 AWS SCT 확장 팩의 AWS Lambda 함수 사용 섹션을 참조하세요.

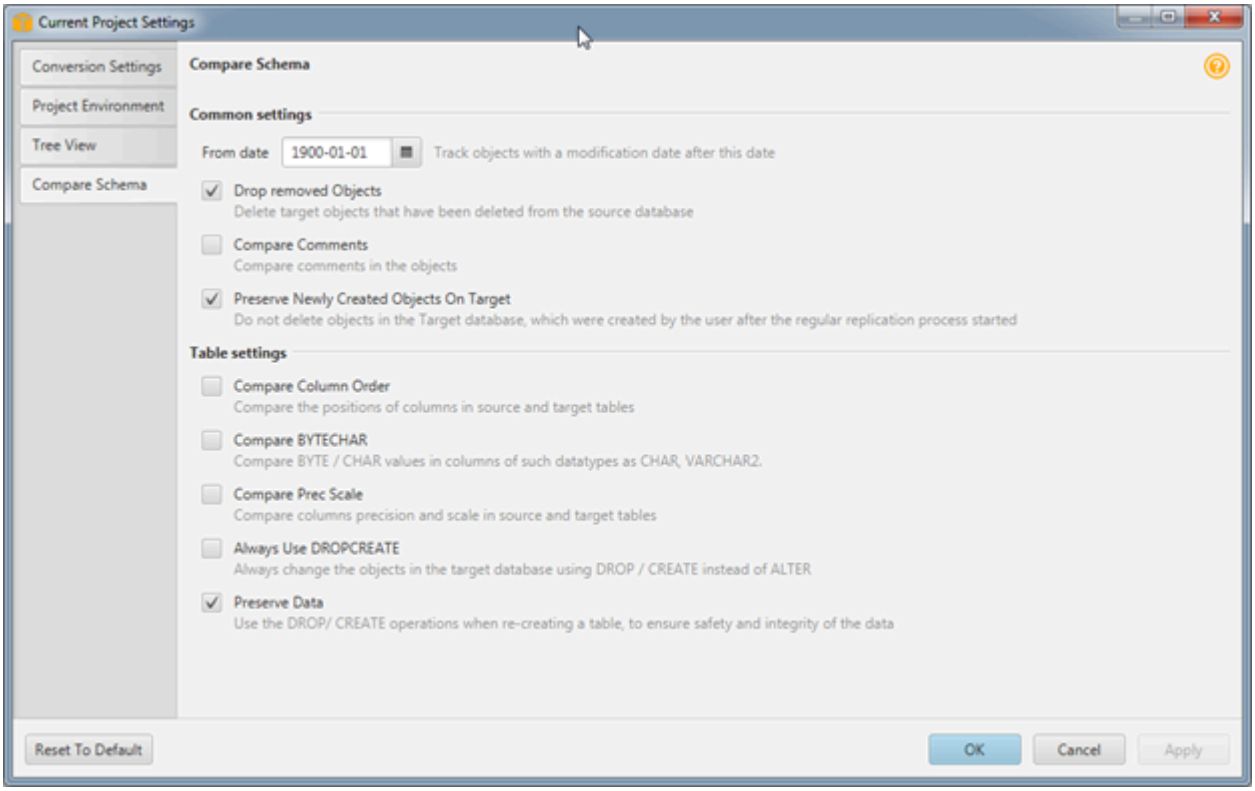
데이터베이스 스키마 비교

마이그레이션한 후 소스 또는 대상 스키마를 변경한 경우 AWS SCT를 사용하여 두 데이터베이스 스키마를 비교할 수 있습니다. 소스 스키마와 같은 버전 또는 이전 버전의 스키마를 비교할 수 있습니다.

다음과 같은 스키마 비교가 지원됩니다.

- Oracle 대 Oracle, 버전 12.1.0.2.0, 11.1.0.7.0, 11.2.0.1.0, 10
- SQL Server 대 SQL Server, 버전 2016, 2014, 2012, 2008 RD2, 2008
- PostgreSQL 대 PostgreSQL 및 Aurora PostgreSQL-Compatible Edition, 버전 9.6, 9.5.9, 9.5.4
- MySQL 대 MySQL, 버전 5.6.36, 5.7.17, 5.5

프로젝트 설정 페이지의 Compare Schema 탭에서 스키마 비교를 위한 설정을 지정합니다.



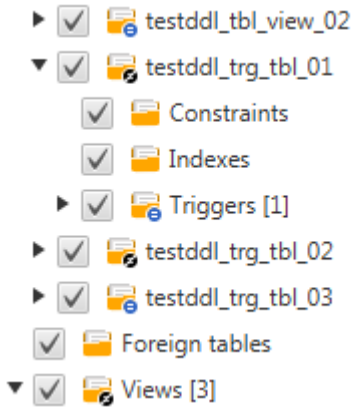
스키마를 비교하려면 스키마를 선택합니다. 그러면 AWS SCT가 두 스키마 간에 차이가 있는 객체와 그렇지 않은 객체를 표시합니다.

두 스키마를 비교하려면

1. 기존 AWS SCT 프로젝트를 열거나, 프로젝트를 만들고 소스 및 대상 엔드포인트에 연결합니다.

2. 비교하려는 스키마를 선택합니다.
3. 컨텍스트(마우스 오른쪽 버튼 클릭) 메뉴를 열고 Compare Schema를 선택합니다.

AWS SCT가 객체 아이콘에 검은색 원을 추가하여 두 스키마 간에 차이가 있는 객체를 나타냅니다.



스키마 비교 결과를 단일 객체, 객체의 단일 범주 또는 전체 스키마에 적용할 수 있습니다. 결과를 적용하려는 범주, 객체 또는 스키마 옆의 상자를 선택합니다.

관련 변환된 객체 찾기

스키마 변환 후 AWS SCT가 소스 데이터베이스의 한 스키마 객체에 대해 여러 개의 객체를 생성하는 경우가 있을 수 있습니다. 예를 들어, Oracle에서 PostgreSQL로 변환하는 경우 AWS SCT는 각 Oracle 트리거를 가져와서 PostgreSQL 대상의 트리거 및 트리거 함수로 변환합니다. 또한 AWS SCT가 Oracle 패키지 함수 또는 프로시저를 PostgreSQL로 변환하면 프로시저나 함수를 실행하기 전에 init 블록으로 실행해야 하는 동일한 함수와 INIT 함수가 생성됩니다.

다음 프로시저를 사용하면 스키마 변환 후에 생성된 모든 관련 객체를 볼 수 있습니다.

스키마 변환 중에 생성된 관련 객체를 보려면

1. 스키마 변환 후 대상 트리 보기에서 변환된 객체를 선택합니다.
2. Related Converted Objects 탭을 선택합니다.
3. 관련 대상 객체의 목록을 볼 수 있습니다.

AWS SCT를 사용하여 데이터 웨어하우스 스키마를 Amazon Redshift로 변환

AWS Schema Conversion Tool(AWS SCT)은 데이터 웨어하우스 스키마를 Amazon Redshift 데이터베이스 스키마로 변환하는 대부분의 프로세스를 자동화합니다. 소스 및 대상 데이터베이스 엔진은 다양한 특징과 기능을 가질 수 있으므로 AWS SCT에서는 가능한 경우 대상 데이터베이스에 동일한 스키마를 생성합니다. 직접 변환이 불가능한 경우 AWS SCT는 수행할 수 있는 작업 목록이 포함된 평가 보고서를 제공합니다. AWS SCT를 사용하면 키를 관리하고, 데이터 유형과 객체를 매핑하고, 수동 변환을 생성할 수 있습니다.

AWS SCT는 다음과 같은 데이터 웨어하우스 스키마를 Amazon Redshift로 변환할 수 있습니다.

- Amazon Redshift
- Azure Synapse Analytics(버전 10)
- BigQuery
- Greenplum Database(버전 4.3)
- Microsoft SQL Server(버전 2008 이상)
- Netezza(버전 7.0.3 이상)
- Oracle(버전 10.2 이상)
- Snowflake(버전 3)
- Teradata(버전 13 이상)
- Vertica(버전 7.2 이상)

온라인 트랜잭션 처리(OLTP) 데이터베이스 스키마 변환에 대한 자세한 내용은 AWS SCT를 사용하여 데이터베이스 스키마 변환 항목을 참조하세요.

데이터 웨어하우스 스키마를 변환하려면 다음 단계를 수행합니다.

1. 최적화 전략 및 규칙을 지정하고 AWS SCT에서 사용할 마이그레이션 규칙을 지정합니다. 열의 데이터 형식을 변경하고, 한 스키마에서 다른 스키마로 객체를 이동하고, 객체 이름을 변경하는 규칙을 설정할 수 있습니다.

   설정에서 최적화 및 마이그레이션 규칙을 지정할 수 있습니다. 최적화 전략에 대한 자세한 내용은 AWS SCT에서 사용할 최적화 전략 및 규칙 선택 항목을 참조하세요. 마이그레이션 규칙에 대한 자세한 내용은 AWS SCT에서 마이그레이션 규칙 생성 항목을 참조하세요.

2. AWS SCT에서 데이터 웨어하우스 변환 방식을 최적화할 수 있도록 소스 데이터 웨어하우스에 대한 통계를 제공합니다. 데이터베이스에서 직접 통계를 수집하거나 기존 통계 파일을 업로드할 수 있습니다. 데이터 웨어하우스 통계 제공에 대한 자세한 내용은 AWS SCT에 대한 통계 수집 또는 업로드 항목을 참조하세요.
3. 자동으로 변환할 수 없는 스키마 요소를 자세히 설명하는 데이터베이스 마이그레이션 평가 보고서를 생성합니다. 이 보고서를 사용하면 대상 데이터베이스에서 소스 데이터베이스와 호환되는 스키마를 수동으로 만들어야 하는 위치를 식별할 수 있습니다. 평가 보고서에 대한 자세한 내용은 AWS SCT를 사용하여 마이그레이션 평가 보고서 작성 섹션을 참조하세요.
4. 스키마를 변환합니다. AWS SCT에서 사용자가 검토할 수 있도록 변환된 스키마의 로컬 버전을 생성하지만 준비가 될 때까지는 대상 데이터베이스에 적용되지 않습니다. 변환에 대한 자세한 내용은 AWS SCT를 사용하여 스키마 변환 섹션을 참조하세요.
5. 스키마를 변환한 후 키를 관리하고 편집할 수 있습니다. 키 관리는 데이터 웨어하우스 변환의 핵심입니다. 키 관리에 대한 자세한 내용은 AWS SCT에서 키 관리 및 사용자 지정 섹션을 참조하세요.
6. 자동으로 변환할 수 없는 스키마 요소가 있는 경우 소스 스키마를 업데이트한 다음 다시 변환하거나, 대상 데이터베이스에서 동일한 스키마 요소를 생성할 수 있습니다. 스키마 요소의 수동 변환에 대한 자세한 내용은 AWS SCT에서 수동 변환 처리 섹션을 참조하세요. 소스 스키마 업데이트에 대한 자세한 내용은 AWS SCT에서 변환된 스키마 업데이트 및 새로 고침 섹션을 참조하세요.
7. 준비가 되면 변환된 스키마를 대상 데이터베이스에 적용할 수 있습니다. 변환된 스키마 저장 및 적용에 대한 자세한 내용은 AWS SCT에서 변환된 스키마 저장 및 적용 섹션을 참조하세요.

Amazon Redshift를 대상으로 사용할 수 있는 권한

Amazon Redshift를 대상으로 사용하기 위해 필요한 권한은 다음과 같습니다.

- CREATE ON DATABASE - 데이터베이스에 새 스키마를 생성할 수 있습니다.
- CREATE ON SCHEMA - 데이터베이스 스키마에 객체를 생성할 수 있습니다.
- GRANT USAGE ON LANGUAGE - 데이터베이스에 새 함수와 프로시저를 생성할 수 있습니다.
- GRANT SELECT ON ALL TABLES IN SCHEMA pg_catalog - Amazon Redshift 클러스터에 대한 시스템 정보를 사용자에게 제공합니다.
- GRANT SELECT ON pg_class_info - 사용자에게 테이블 배포 스타일에 대한 정보를 제공합니다.

다음 코드 예제를 사용하여 데이터베이스 사용자를 생성하고 권한을 부여할 수 있습니다.

```
CREATE USER user_name PASSWORD your_password;
GRANT CREATE ON DATABASE db_name TO user_name;
```

```
GRANT CREATE ON SCHEMA schema_name TO user_name;
GRANT USAGE ON LANGUAGE plpythonu TO user_name;
GRANT USAGE ON LANGUAGE plpgsql TO user_name;
GRANT SELECT ON ALL TABLES IN SCHEMA pg_catalog TO user_name;
GRANT SELECT ON pg_class_info TO user_name;
GRANT SELECT ON sys_serverless_usage TO user_name;
GRANT SELECT ON pg_database_info TO user_name;
GRANT SELECT ON pg_statistic TO user_name;
```

이전 예제에서 *user_name*을 사용자 이름으로 바꿉니다. 그런 다음 *db_name*을 대상 Amazon Redshift 데이터베이스의 이름으로 바꿉니다. 그 다음으로, *schema_name*을 Amazon Redshift 스키마의 이름으로 바꿉니다. 변환된 코드를 적용하거나 데이터를 마이그레이션할 각 대상 스키마에 대해 `GRANT CREATE ON SCHEMA` 작업을 반복합니다. 마지막으로 *your_password*를 안전한 암호로 바꿉니다.

대상 Amazon Redshift 데이터베이스에 확장 팩을 적용할 수 있습니다. 확장 팩은 객체를 Amazon Redshift로 변환할 때 필요한 소스 데이터베이스 함수를 에뮬레이션하는 추가 기능 모듈입니다. 자세한 내용은 AWS SCT 확장 팩 사용 섹션을 참조하세요.

이 작업을 수행하려면 사용자 대신 Amazon S3 버킷에 액세스할 수 있는 권한이 AWS SCT에 필요합니다. 이 권한을 제공하려면 다음 정책에 따라 AWS Identity and Access Management(IAM) 사용자를 생성합니다.

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Action":[
            "s3:DeleteObject",
            "s3:GetObject",
            "s3:ListBucket",
            "s3:PutObject"
         ],
         "Resource":[
            "arn:aws:s3:::aws-sct-*"
         ]
      },
      {
         "Effect":"Allow",
         "Action":[
            "s3:ListAllMyBuckets"
```

```
        ],
        "Resource":""
      }
   ]
}
```

AWS SCT에서 사용할 최적화 전략 및 규칙 선택

AWS Schema Conversion Tool에서 데이터 웨어하우스 스키마를 변환하는 방법을 최적화하려면 도구에서 사용할 전략과 규칙을 선택할 수 있습니다. 스키마를 변환하고 제안된 키를 검토한 후 규칙을 조정하거나 전략을 변경하여 원하는 결과를 얻을 수 있습니다.

최적화 전략 및 규칙을 선택하려면

1. 설정을 선택하고 프로젝트 설정을 선택합니다. [Current project settings] 대화 상자가 나타납니다.
2. 왼쪽 창에서 Optimization Strategies를 선택합니다. 최적화 전략은 기본값이 선택된 상태로 오른쪽 창에 표시됩니다.
3. Strategy Sector에서 사용하려는 최적화 전략을 선택합니다. 다음 중에서 선택할 수 있습니다.

   - Use metadata, ignore statistical information - 이 전략에서는 메타데이터의 정보만 최적화 결정에 사용됩니다. 예를 들어 소스 테이블에 둘 이상의 인덱스가 있는 경우 소스 데이터베이스 정렬 순서가 사용되고 첫 번째 인덱스는 이 배포 키가 됩니다.

   - Ignore metadata, use statistical information - 이 전략에서는 통계 정보에서만 최적화 결정을 내립니다. 이 전략은 통계가 제공되는 테이블 및 열에만 적용됩니다. 자세한 내용은 AWS SCT에 대한 통계 수집 또는 업로드 섹션을 참조하세요.

   - Use metadata and use statistical information - 이 전략에서는 메타데이터와 통계를 모두 최적화 결정에 사용합니다.

4. 최적화 전략을 선택한 후 사용할 규칙을 선택할 수 있습니다. 다음 중에서 선택할 수 있습니다.

   - Choose Distribution Key and Sort Keys using metadata
   - Choose fact table and appropriate dimension for collation
   - Analyze cardinality of indexes' columns

- Find the most used tables and columns from the query log table

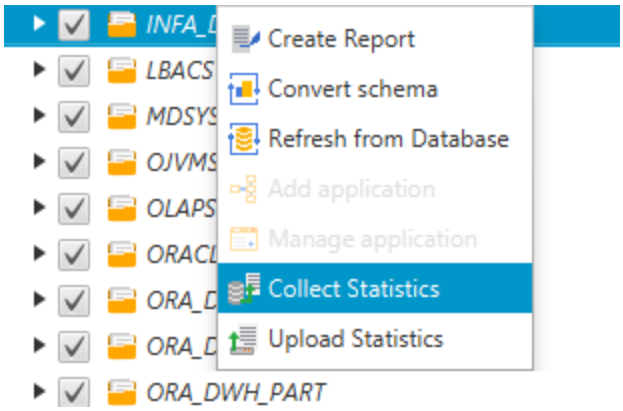
각 규칙에 대해 정렬 키의 가중치와 배포 키의 가중치를 입력할 수 있습니다. AWS SCT에서는 스키마를 변환할 때 선택한 가중치를 사용합니다. 나중에 제안된 키를 검토할 때 결과가 만족스럽지 않으면 여기로 돌아와 설정을 변경할 수 있습니다. 자세한 내용은 AWS SCT에서 키 관리 및 사용자 지정 섹션을 참조하세요.

AWS SCT에 대한 통계 수집 또는 업로드

AWS Schema Conversion Tool에서 데이터 웨어하우스 스키마를 변환하는 방법을 최적화하기 위해 도구에서 사용할 수 있는 소스 데이터베이스의 통계를 제공할 수 있습니다. 데이터베이스에서 직접 통계를 수집하거나 기존 통계 파일을 업로드할 수 있습니다.

통계를 제공 및 검토하려면

1. 프로젝트를 열고 소스 데이터베이스에 연결합니다.
2. 프로젝트의 왼쪽 패널에서 스키마 객체를 선택하고 해당 객체에 대한 컨텍스트(마우스 오른쪽 버튼 클릭) 메뉴를 엽니다. 다음과 같이 Collect Statistics 또는 Upload Statistics를 선택합니다.



3. 프로젝트의 왼쪽 패널에서 스키마 객체를 선택한 다음 통계 탭을 선택합니다. 객체에 대한 통계를 검토할 수 있습니다.

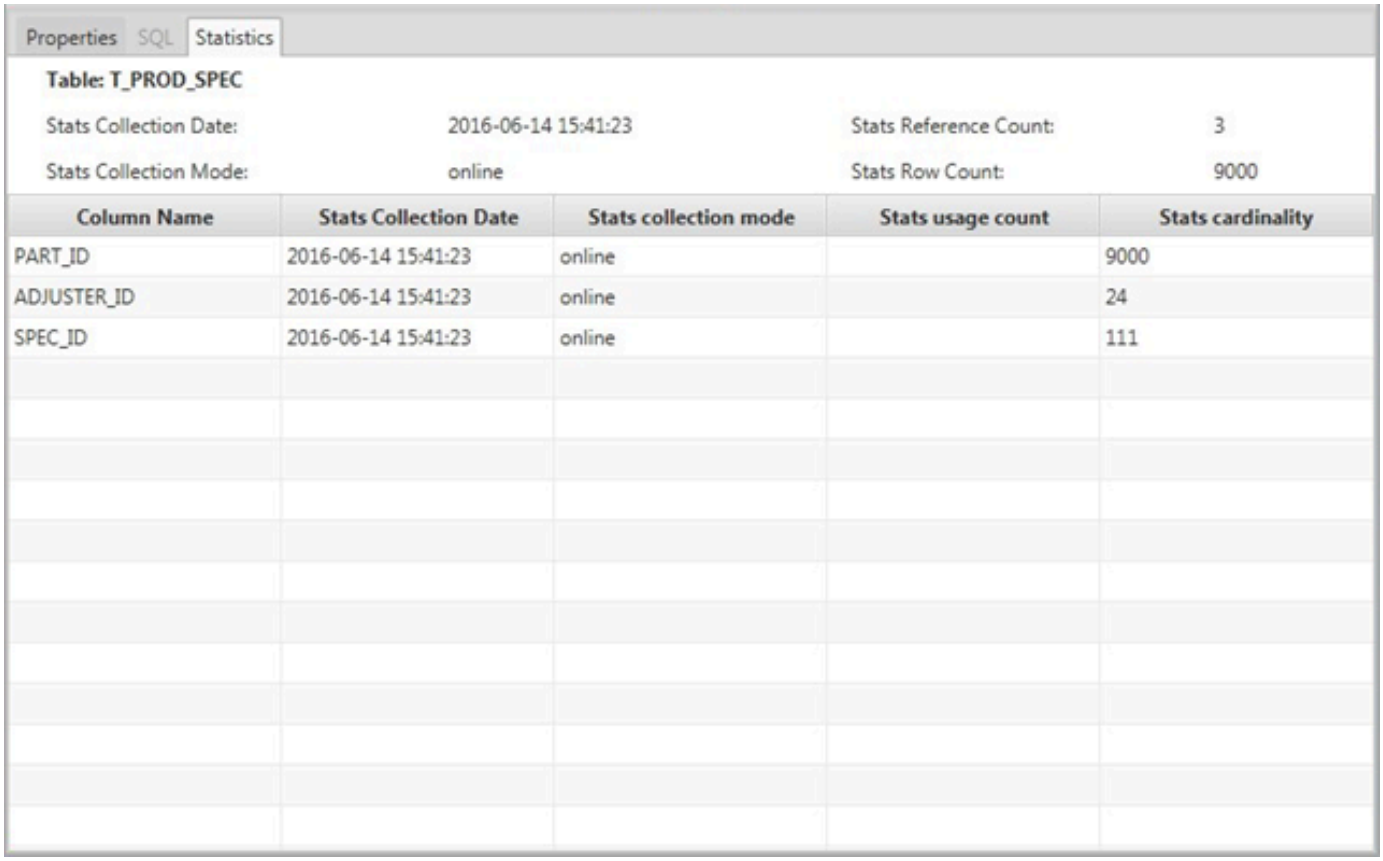

Properties SQL Statistics

Table: T_PROD_SPEC

Stats Collection Date: 2016-06-14 15:41:23 Stats Reference Count: 3

Stats Collection Mode: online Stats Row Count: 9000

Column Name	Stats Collection Date	Stats collection mode	Stats usage count	Stats cardinality
PART_ID	2016-06-14 15:41:23	online		9000
ADJUSTER_ID	2016-06-14 15:41:23	online		24
SPEC_ID	2016-06-14 15:41:23	online		111

나중에 제안된 키를 검토할 때 결과가 만족스럽지 않으면 추가 통계를 수집하고 이 절차를 반복할 수 있습니다. 자세한 내용은 AWS SCT에서 키 관리 및 사용자 지정 섹션을 참조하세요.

AWS SCT에서 마이그레이션 규칙 생성

AWS SCT를 사용하여 스키마를 변환하기 전에 마이그레이션 규칙을 설정할 수 있습니다. 마이그레이션 규칙은 열의 데이터 유형 변경, 한 스키마에서 다른 스키마로 객체 이동, 객체 이름 변경 등의 작업을 수행할 수 있습니다. 예를 들어 소스 스키마에 `test_TABLE_NAME`이라는 테이블 세트가 있다고 가정해 봅니다. 대상 스키마의 접두사 `test_`를 접두사 `demo_`로 변경하는 규칙을 설정할 수 있습니다.

> **Note**
>
> 서로 다른 소스 및 대상 데이터베이스 엔진에 대한 마이그레이션 규칙만 생성할 수 있습니다.

다음 작업을 수행하는 마이그레이션 규칙을 생성할 수 있습니다.

- 접두사 추가, 제거 또는 교체

- 접미사 추가, 제거 또는 교체
- 열 데이터 정렬 변경
- 데이터 유형 변경
- char, varchar, nvarchar 및 string 데이터 유형의 길이 변경
- 객체 이동
- 객체 이름 변경

다음 객체에 대한 마이그레이션 규칙을 생성할 수 있습니다.

- 데이터베이스
- Schema
- 표
- 열

마이그레이션 규칙 생성

마이그레이션 규칙을 생성하고 규칙을 프로젝트의 일부로 저장할 수 있습니다. 프로젝트를 연 상태에서 다음 절차를 사용하여 마이그레이션 규칙을 생성합니다.

마이그레이션 규칙을 생성하려면

1. 보기 메뉴에서 매핑 보기를 선택합니다.
2. Server mappings에서 소스 및 대상 서버 쌍을 선택합니다.
3. New migration rule을 선택합니다. 변환 규칙 대화 상자가 나타납니다.
4. 새 규칙 추가를 선택합니다. 규칙 목록에 새 행이 추가됩니다.
5. 규칙을 구성합니다.

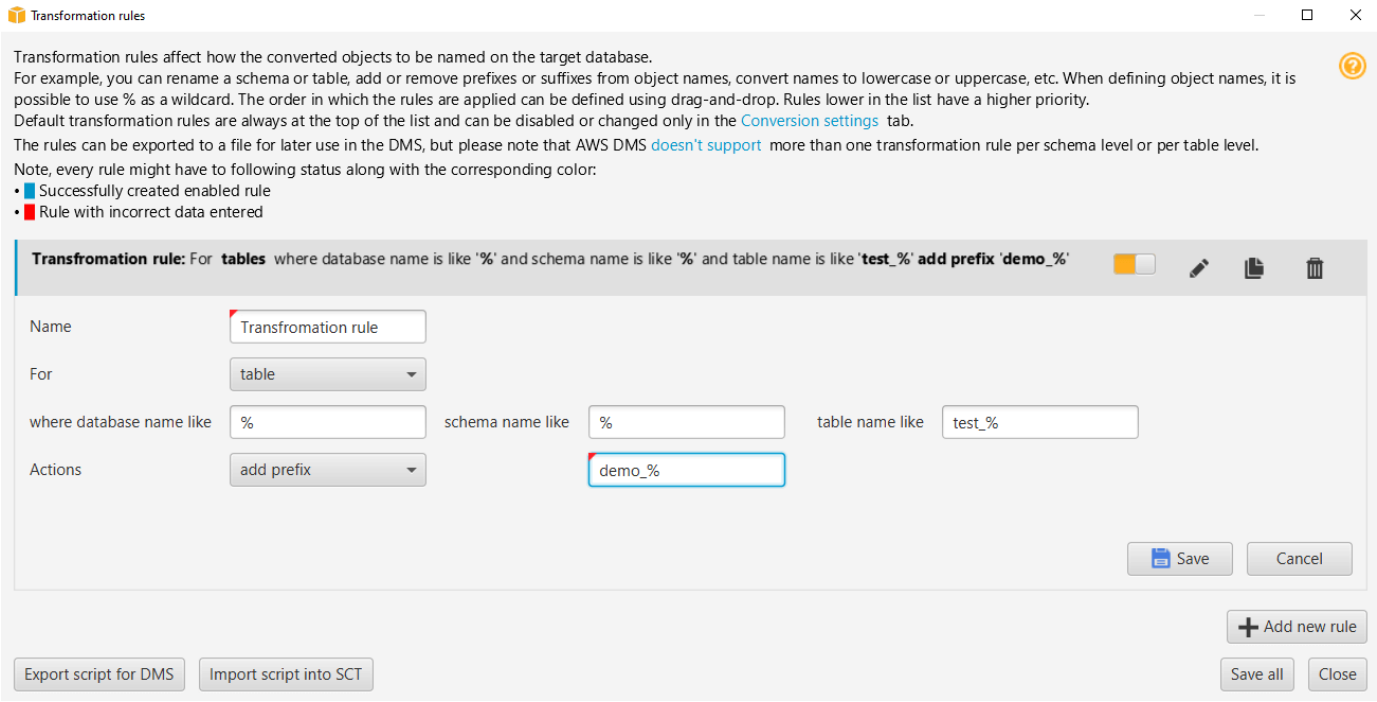
    a. Name(이름)에 규칙의 이름을 입력합니다.
    b. For에서 규칙이 적용되는 객체 유형을 선택합니다.
    c. where에서 마이그레이션 규칙을 적용하기 전에 객체에 적용할 필터를 입력합니다. where 절은 like 절을 사용하여 평가됩니다. 정확한 이름을 입력하여 하나의 객체를 선택하거나 패턴을 입력하여 여러 객체를 선택할 수 있습니다.

        where 절에 사용할 수 있는 필드는 객체 유형에 따라 다릅니다. 예를 들어, 객체 유형이 스키마인 경우 스키마 이름에 사용할 수 있는 필드는 하나뿐입니다.

d. 작업에서 생성하려는 마이그레이션 규칙 유형을 선택합니다.

e. 규칙 유형에 따라 하나 또는 두 개의 추가 값을 입력합니다. 예를 들어, 객체의 이름을 바꾸려면 객체의 새 이름을 입력합니다. 접두사를 바꾸려면 이전 접두사와 새 접두사를 입력합니다.

6. 마이그레이션 규칙을 구성한 후 저장을 선택하여 규칙을 저장합니다. 취소를 선택하여 변경 사항을 취소할 수도 있습니다.



7. 규칙 추가, 편집 및 삭제를 완료한 후 모두 저장을 선택하여 변경 내용을 모두 저장합니다.

8. 닫기를 선택하여 변환 규칙 대화 상자를 닫습니다.

토글 아이콘을 사용하면 마이그레이션 규칙을 삭제하지 않고 끌 수 있습니다. 복사 아이콘을 사용하면 기존 마이그레이션 규칙을 복제할 수 있습니다. 연필 아이콘을 사용하면 기존 마이그레이션 규칙을 편집할 수 있습니다. 삭제 아이콘을 사용하면 기존 마이그레이션 규칙을 삭제할 수 있습니다. 마이그레이션 규칙 변경 내용을 저장하려면 모두 저장을 선택합니다.

마이그레이션 규칙 내보내기

AWS Database Migration Service(AWS DMS)를 사용하여 소스 데이터베이스의 데이터를 대상 데이터베이스로 마이그레이션하는 경우 마이그레이션 규칙에 대한 정보를 AWS DMS에 제공할 수 있습니다. 작업에 대한 자세한 내용은 AWS Database Migration Service 복제 작업 사용 섹션을 참조하세요.

마이그레이션 규칙을 내보내려면

1. AWS Schema Conversion Tool의 보기 메뉴에서 매핑 보기를 선택합니다.
2. Migration rules에서 마이그레이션 규칙을 선택한 다음, Modify migration rule를 선택합니다.
3. Export script for AWS DMS를 선택합니다.
4. 스크립트를 저장할 위치로 이동한 다음 저장을 선택합니다. 마이그레이션 규칙은 AWS DMS에서 사용할 수 있는 JSON 스크립트로 저장됩니다.

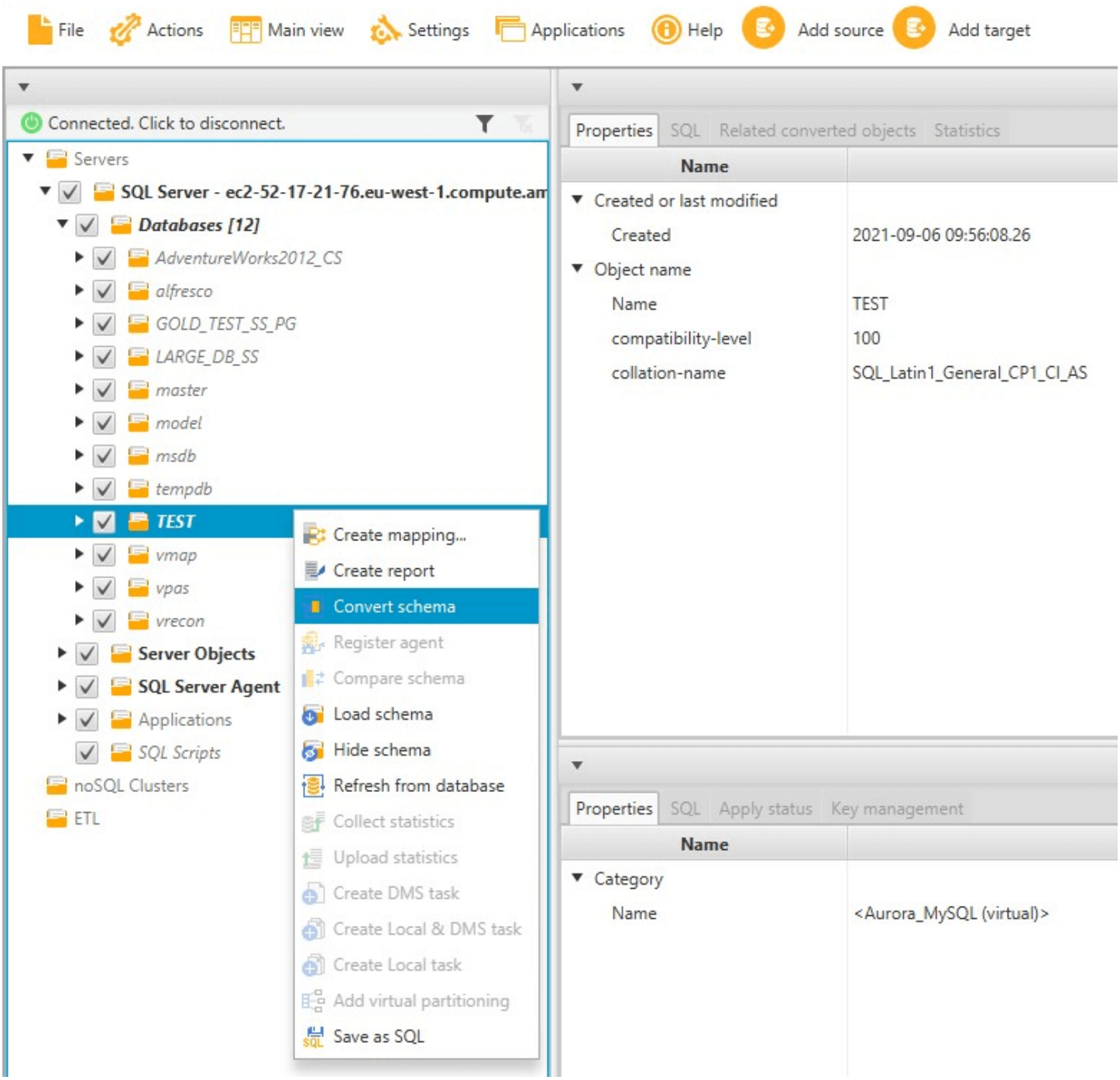
AWS SCT를 사용하여 스키마 변환

프로젝트를 소스 데이터베이스와 대상 데이터베이스에 모두 연결하면 AWS Schema Conversion Tool 프로젝트가 소스 데이터베이스의 스키마를 왼쪽 패널에 표시합니다. 스키마는 트리 보기 형식으로 표시되며 트리의 각 노드는 지연 로드됩니다. 트리 보기에서 노드를 선택하면 이때 AWS SCT가 소스 데이터베이스의 스키마 정보를 요청합니다.

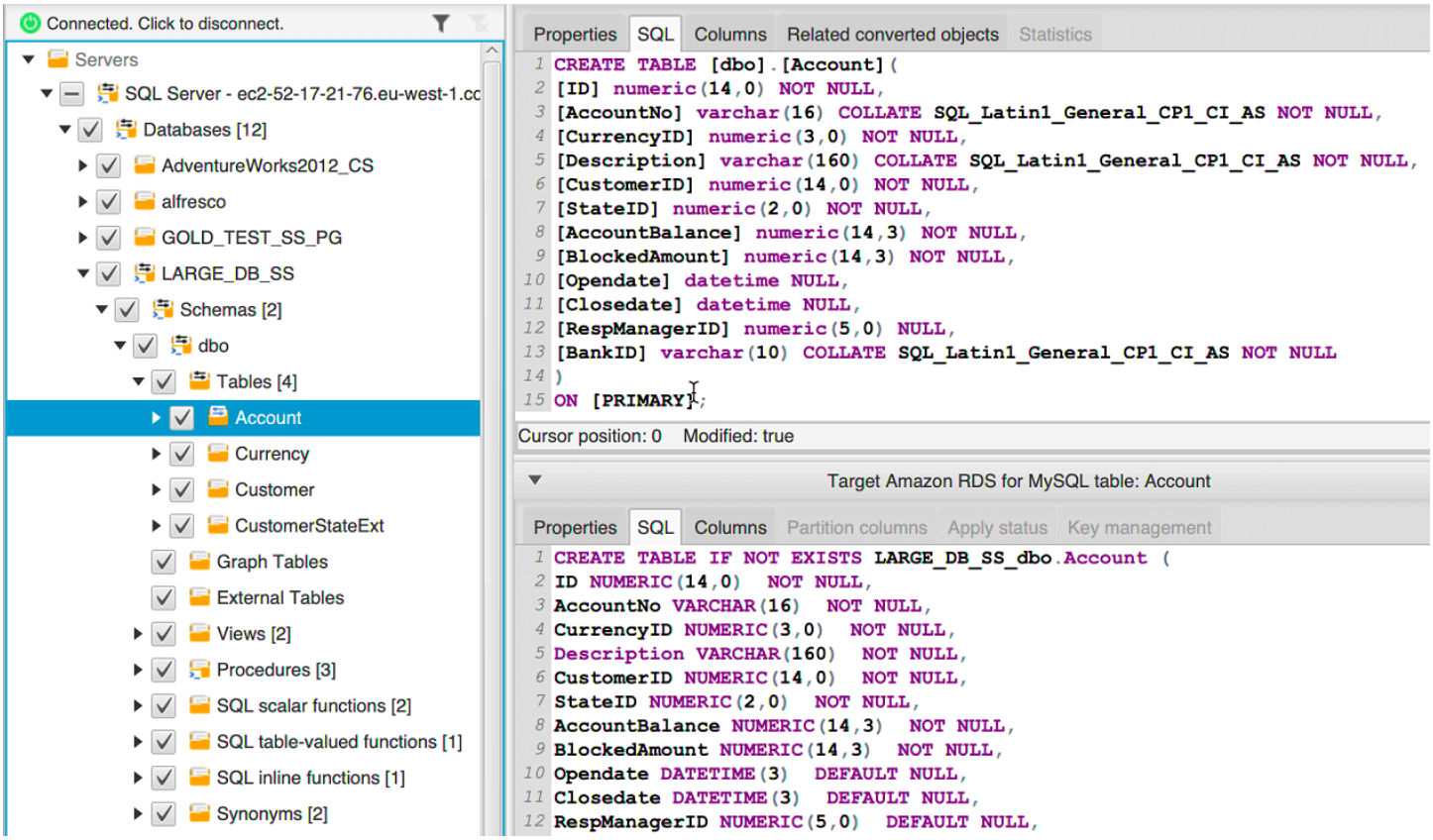
소스 데이터베이스에서 스키마 항목을 선택한 다음 해당 스키마를 대상 데이터베이스의 데이터베이스 엔진에 해당하는 동일한 스키마로 변환할 수 있습니다. 소스 데이터베이스에서 변환할 스키마 항목을 선택할 수 있습니다. 선택한 스키마 항목이 상위 항목에 따라 달라지는 경우, AWS SCT는 상위 항목에 대한 스키마도 생성합니다. 예를 들어, 테이블에서 변환할 열을 선택하면 AWS SCT는 해당 열, 해당 열이 있는 테이블 및 해당 테이블이 있는 데이터베이스에 대한 스키마를 생성합니다.

스키마 변환

소스 데이터베이스의 스키마를 변환하려면 변환할 스키마 이름의 확인란을 선택합니다. 그런 다음 프로젝트의 왼쪽 패널에서 이 스키마를 선택합니다. 그러면 AWS SCT가 스키마 이름을 파란색으로 강조 표시합니다. 스키마의 컨텍스트(마우스 오른쪽 버튼 클릭) 메뉴를 열고 아래와 같이 스키마 변환을 선택합니다.



소스 데이터베이스에서 스키마를 변환한 후 프로젝트의 왼쪽 패널에서 스키마 항목을 선택하고 프로젝트의 중앙 패널에서 변환된 스키마를 볼 수 있습니다. 하단 중앙 패널에는 다음과 같이 변환된 스키마를 생성하기 위한 속성 및 SQL 명령이 표시됩니다.



스키마를 변환한 후 프로젝트를 저장할 수 있습니다. 소스 데이터베이스의 스키마 정보는 프로젝트와 함께 저장됩니다. 이 기능을 사용하면 소스 데이터베이스에 연결하지 않고도 오프라인으로 작업할 수 있습니다. 소스 데이터베이스에 대해 데이터베이스에서 새로 고침을 선택하면 AWS SCT가 소스 데이터베이스에 연결하여 프로젝트의 스키마를 업데이트합니다. 자세한 내용은 AWS SCT에서 변환된 스키마 업데이트 및 새로 고침 섹션을 참조하세요.

자동으로 변환할 수 없는 항목의 데이터베이스 마이그레이션 평가 보고서를 생성할 수 있습니다. 평가 보고서는 자동으로 변환할 수 없는 스키마 항목을 식별하고 처리하는 데 유용합니다. 자세한 내용은 AWS SCT를 사용하여 마이그레이션 평가 보고서 작성 섹션을 참조하세요.

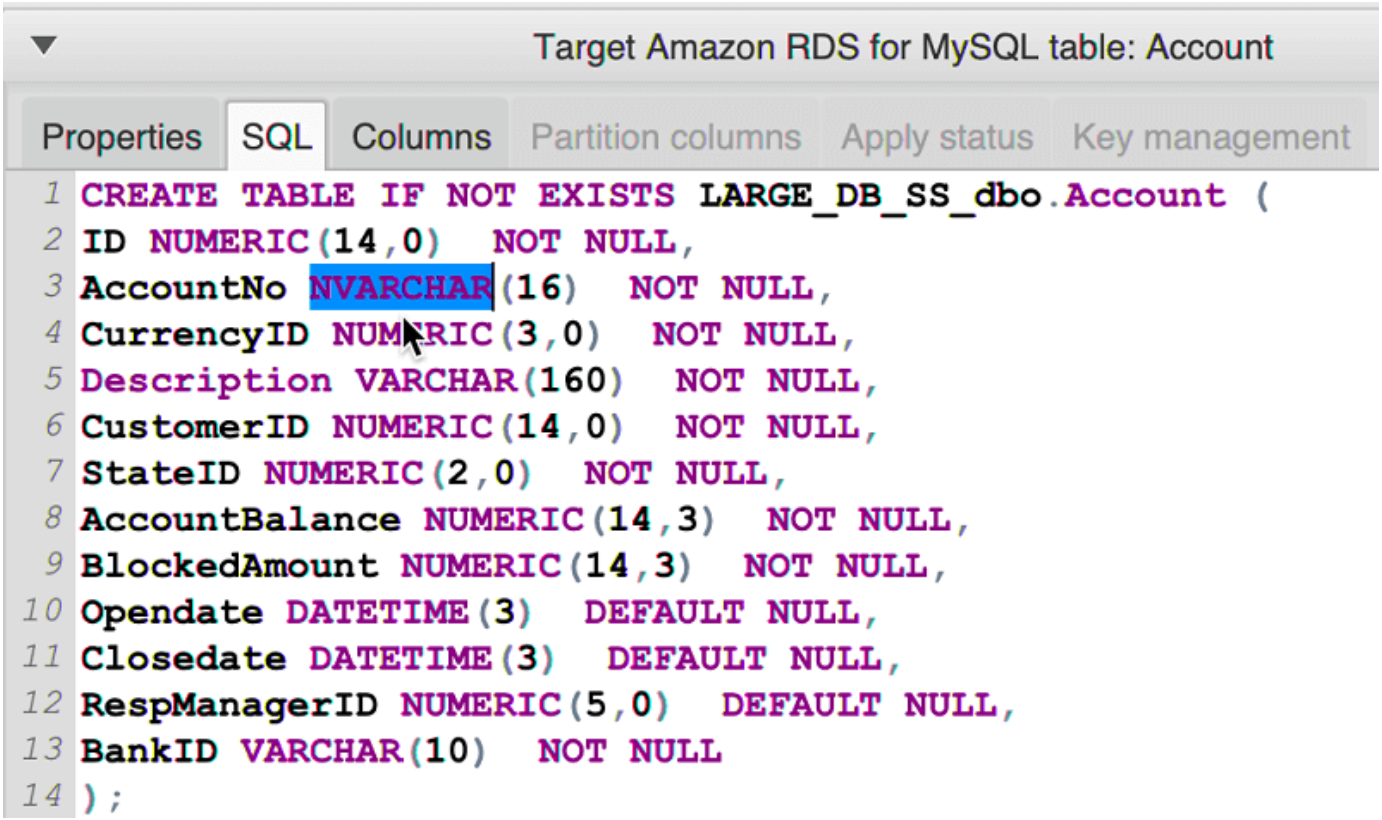
AWS SCT에서 변환된 스키마를 생성할 경우 대상 데이터베이스에 즉시 적용하지 않습니다. 대신 대상 데이터베이스에 적용할 준비가 될 때까지 변환된 스키마를 로컬에 저장합니다. 자세한 내용은 변환된 스키마 적용 섹션을 참조하세요.

변환된 스키마 편집

변환된 스키마를 편집하고 변경 내용을 프로젝트의 일부로 저장할 수 있습니다.

변환된 스키마를 편집하려면

1. 소스 데이터베이스의 스키마를 표시하는 왼쪽 패널에서 변환된 스키마를 편집할 스키마 항목을 선택합니다.
2. 선택한 항목에 대해 변환된 스키마를 표시하는 하단 중앙 패널에서 SQL 탭을 선택합니다.
3. SQL 탭에 표시된 텍스트에서 필요에 따라 스키마를 변경합니다. 프로젝트를 업데이트하면 스키마가 프로젝트에 자동으로 저장됩니다.



변환된 스키마에 대한 변경 내용은 업데이트 시 프로젝트와 함께 저장됩니다. 소스 데이터베이스에서 스키마 항목을 새로 변환하고 해당 항목에 대해 이전에 변환된 스키마로 업데이트한 경우 해당하는 기존 업데이트는 소스 데이터베이스를 기반으로 새로 변환된 스키마 항목으로 대체됩니다.

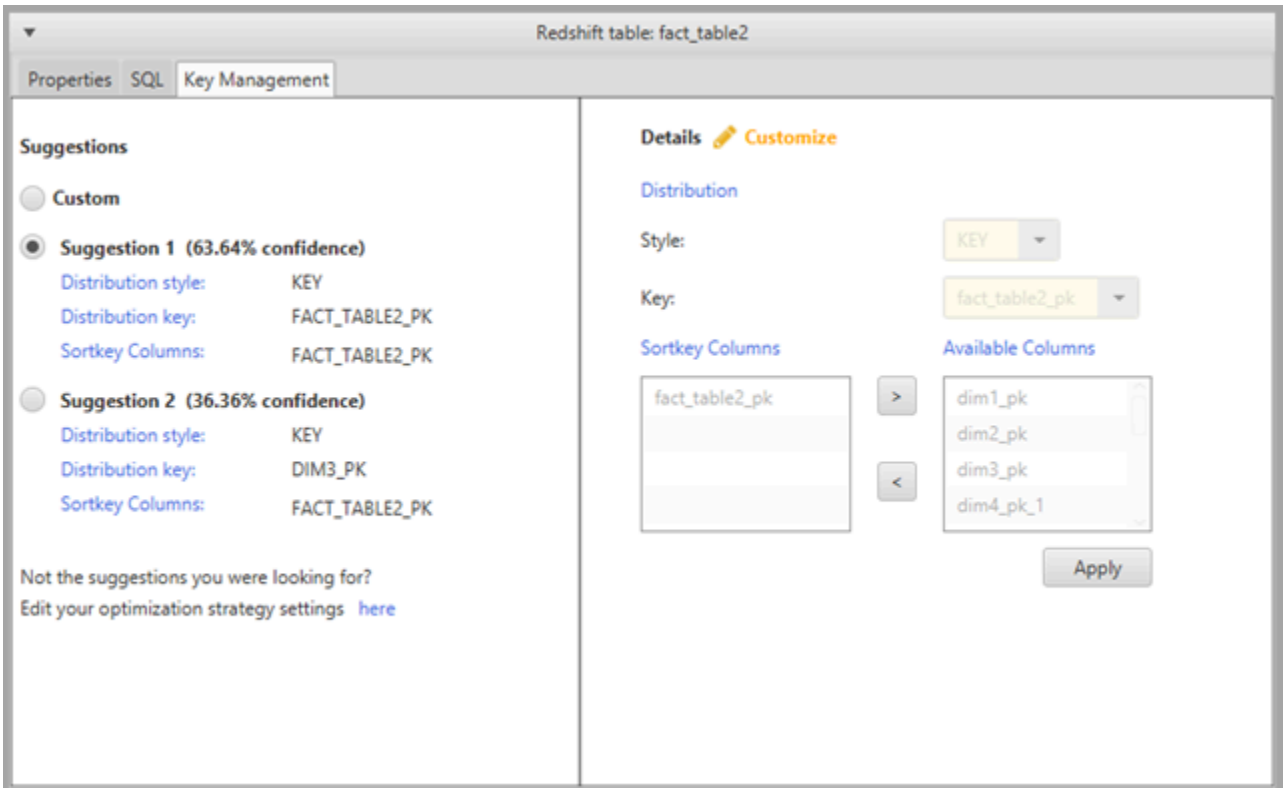
변환된 스키마 지우기

대상 데이터베이스에 스키마를 적용할 때까지 AWS SCT는 변환된 스키마를 프로젝트에 로컬로만 저장합니다. 대상 데이터베이스의 트리 보기 노드를 선택한 다음 데이터베이스에서 새로 고침을 선택하면 프로젝트에서 계획된 스키마를 지울 수 있습니다. 대상 데이터베이스에 스키마가 기록되지 않았으므로 데이터베이스를 새로 고치면 AWS SCT 프로젝트에서 계획된 스키마 요소가 제거되어 대상 데이터베이스에 있는 것과 일치하게 됩니다.

AWS SCT에서 키 관리 및 사용자 지정

AWS Schema Conversion Tool을 사용하여 스키마를 변환한 후 키를 관리하고 편집할 수 있습니다. 키 관리는 데이터 웨어하우스 변환의 핵심입니다.

키를 관리하려면 대상 데이터베이스에서 테이블을 선택하고 다음과 같이 키 관리 탭을 선택합니다.



왼쪽 창에는 주요 제안 사항과 각 제안 사항에 대한 신뢰도가 포함되어 있습니다. 제안 사항 중 하나를 선택하거나 오른쪽 창에서 편집하여 키를 사용자 지정할 수 있습니다.

키 선택 항목이 예상과 다를 경우 최적화 전략을 편집한 다음 변환을 다시 시도할 수 있습니다. 자세한 내용은 AWS SCT에서 사용할 최적화 전략 및 규칙 선택 섹션을 참조하세요.

관련 주제

- 최상의 정렬 키 선택
- 최상의 배포 스타일 선택

AWS SCT에서 평가 보고서 생성 및 사용

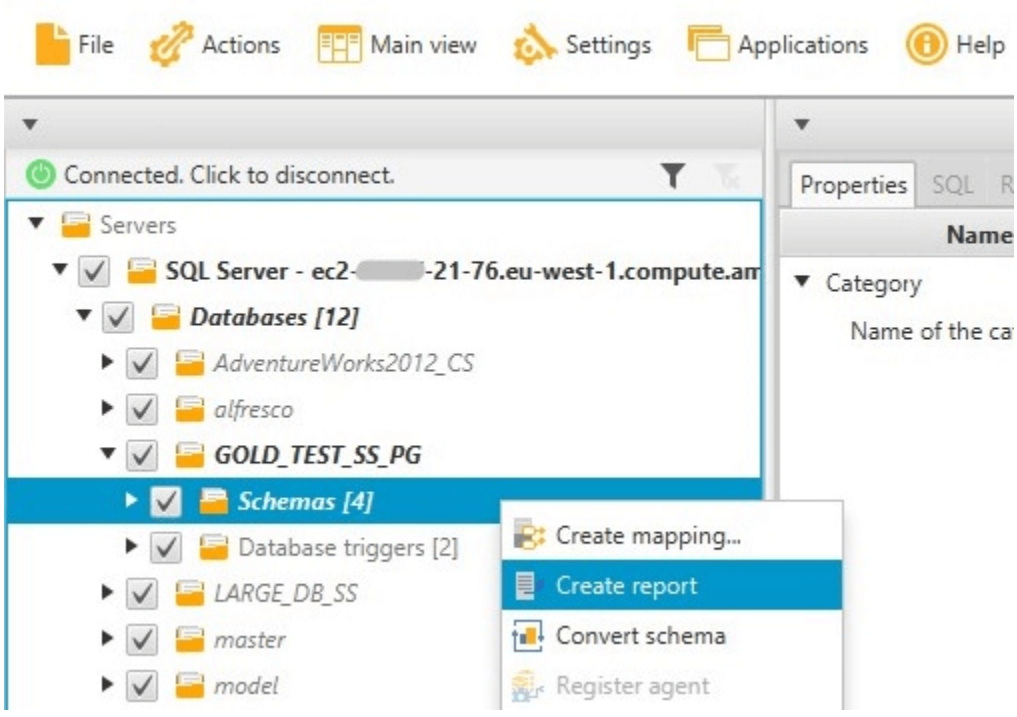
AWS Schema Conversion Tool은 스키마 변환을 지원하기 위해 데이터베이스 마이그레이션 평가 보고서를 생성합니다. 데이터베이스 마이그레이션 평가 보고서는 소스 데이터베이스에서 대상 데이터베이스로의 스키마 변환에 대한 중요한 정보를 제공합니다. 이 보고서는 모든 스키마 변환 작업을 요약하고 대상 데이터베이스의 DB 엔진으로 변환할 수 없는 스키마에 대한 작업 항목을 자세히 설명합니다. 또한 이 보고서에는 대상 데이터베이스에서 자동으로 변환할 수 없는 동등한 코드를 작성하는 데 드는 예상 작업량도 포함되어 있습니다.

데이터베이스 마이그레이션 평가 보고서 생성

다음 절차에 따라 데이터베이스 마이그레이션 평가 보고서를 생성합니다.

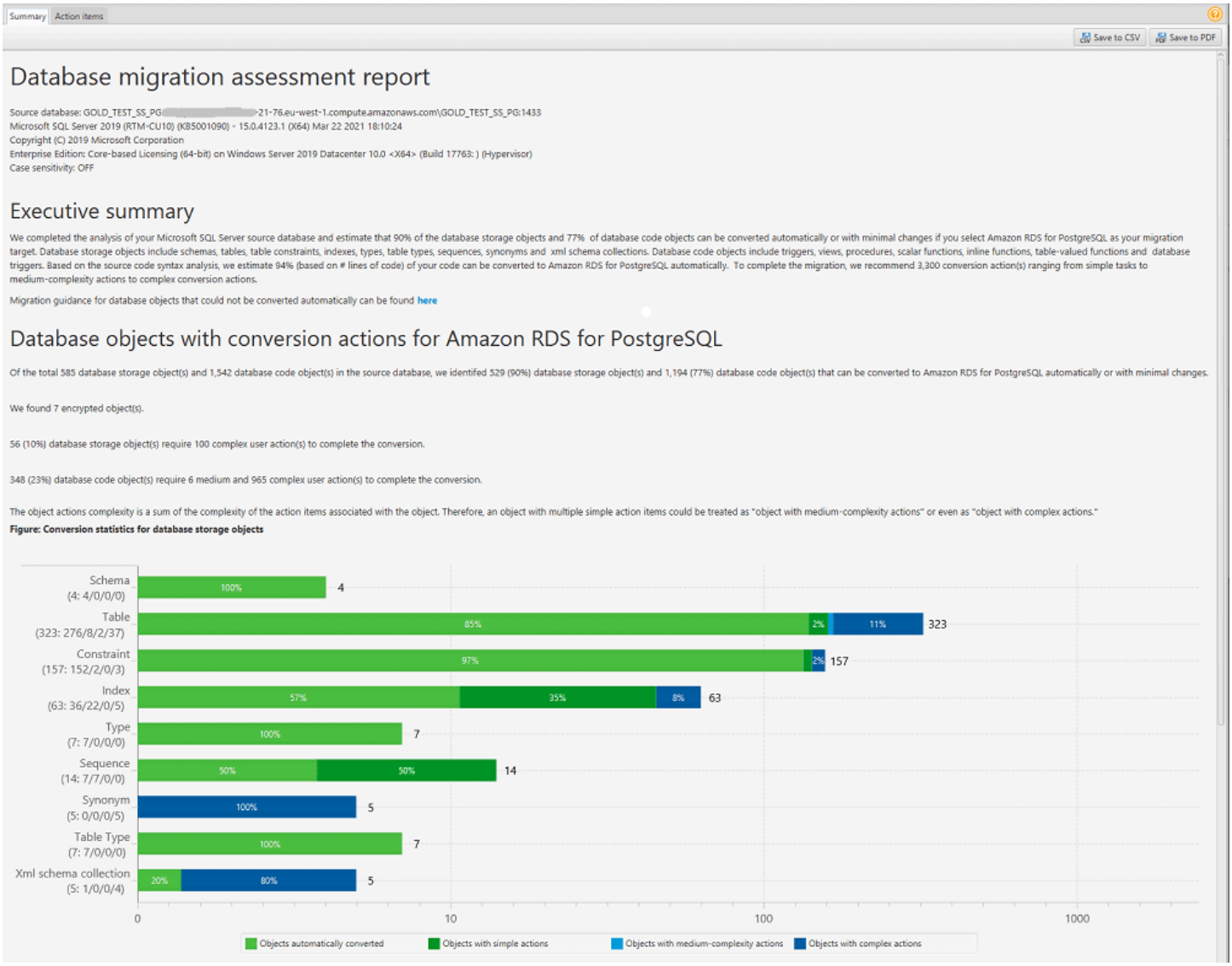
데이터베이스 마이그레이션 평가 보고서를 생성하려면

1. 소스 데이터베이스의 스키마를 표시하는 왼쪽 패널에서 평가 보고서를 생성할 스키마 객체를 선택합니다.
2. 객체의 컨텍스트(마우스 오른쪽 버튼 클릭) 메뉴를 열고 보고서 생성을 선택합니다.



평가 보고서 요약

평가 보고서를 생성하면 평가 보고서 보기가 열리고 요약 탭이 표시됩니다. 요약 탭에는 데이터베이스 마이그레이션 평가 보고서의 요약 정보가 표시됩니다. 자동으로 변환된 항목과 자동으로 변환되지 않은 항목이 표시됩니다.



대상 데이터베이스 엔진으로 자동 변환할 수 없는 스키마 항목의 경우, 요약에는 대상 DB 인스턴스에서 소스에 있는 것과 동일한 스키마 항목을 생성하는 데 필요한 예상 작업량이 포함됩니다.

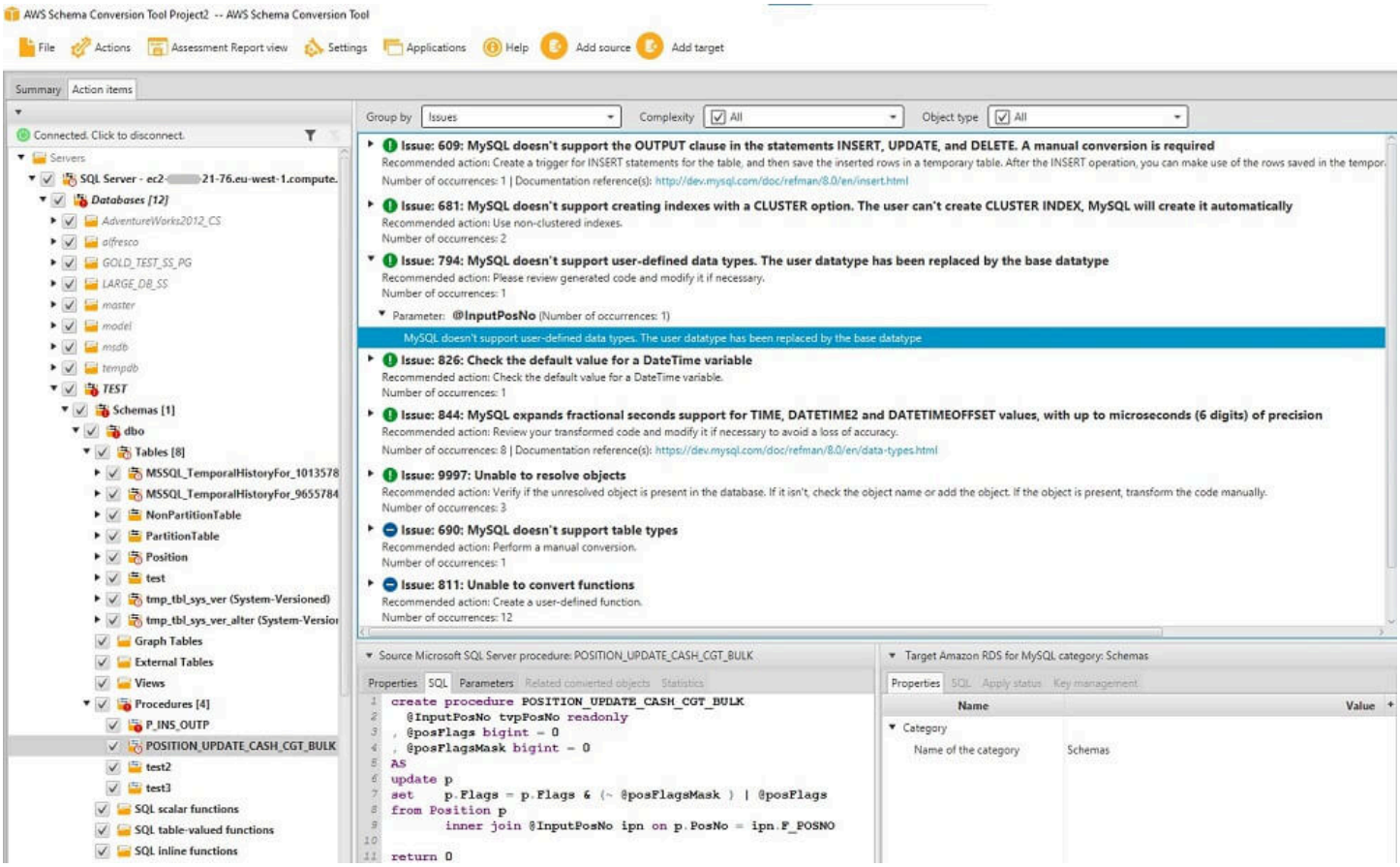
보고서에는 이러한 스키마 항목을 변환하는 데 걸리는 예상 시간이 다음과 같이 분류됩니다.

- 단순 - 1시간 내에 완료할 수 있는 작업입니다.
- 중간 - 더 복잡하고 1~4시간 사이에 완료할 수 있는 작업입니다.
- Significant - 매우 복잡하며 완료하는 데 4시간 이상 걸리는 작업입니다.

평가 보고서 작업 항목

평가 보고서 보기에는 작업 항목 탭도 포함되어 있습니다. 이 탭에는 대상 데이터베이스의 데이터베이스 엔진으로 자동 변환할 수 없는 항목 목록이 포함되어 있습니다. 목록에서 작업 항목을 선택하면 AWS SCT가 스키마에서 해당 작업 항목이 적용되는 항목을 강조 표시합니다.

또한 보고서에는 스키마 항목을 수동으로 변환하는 방법에 대한 권장 사항도 포함되어 있습니다. 수동 변환 처리 방법 결정에 대한 자세한 내용은 AWS SCT에서 수동 변환 처리 섹션을 참조하세요.



평가 보고서 저장

데이터베이스 마이그레이션 평가 보고서의 로컬 사본을 PDF 파일 또는 쉼표로 구분된 값(CSV) 파일로 저장할 수 있습니다. CSV 파일에는 작업 항목 정보만 포함되어 있습니다. PDF 파일에는 다음 예제와 같이 요약 및 작업 항목 정보가 모두 포함되어 있습니다.

Database objects with conversion actions for Amazon RDS for PostgreSQL

Of the total 585 database storage object(s) and 1,542 database code object(s) in the source database, we identifed 529 (90%) database storage object(s) and 1,194 (77%) database code object(s) that can be converted to Amazon RDS for PostgreSQL automatically or with minimal changes.

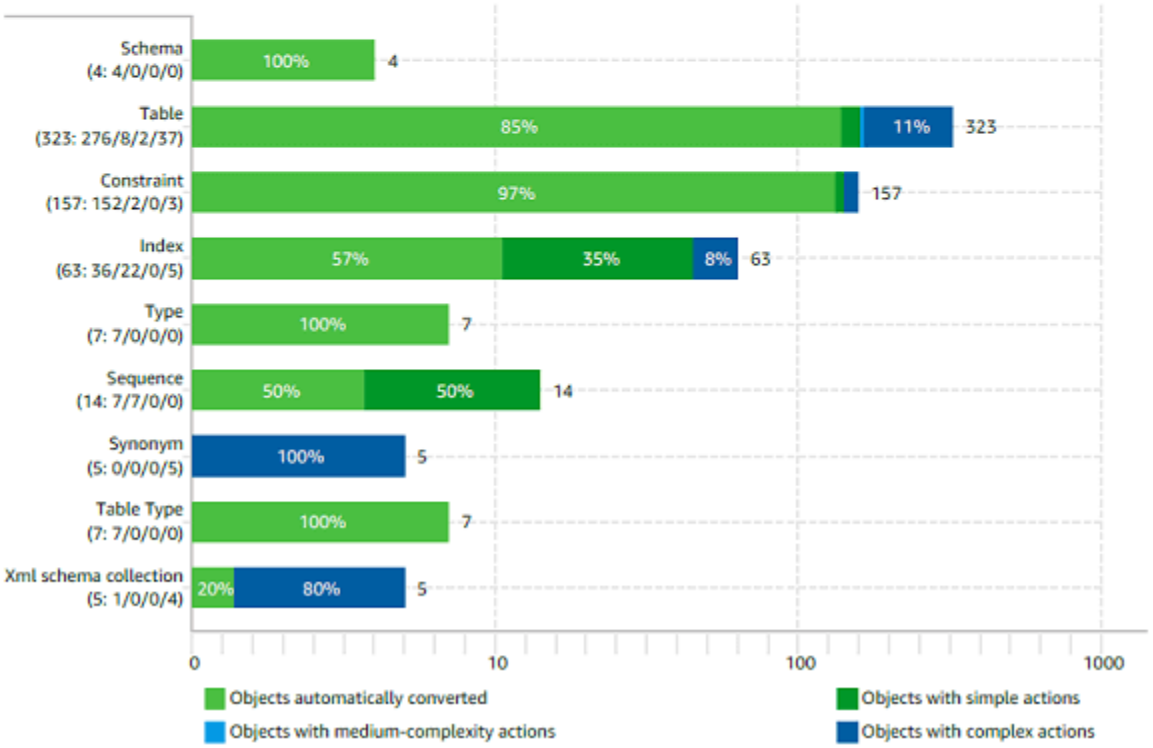
We found 7 encrypted object(s).

56 (10%) database storage object(s) require 100 complex user action(s) to complete the conversion.

348 (23%) database code object(s) require 6 medium and 965 complex user action(s) to complete the conversion.

The object actions complexity is a sum of the complexity of the action items associated with the object. Therefore, an object with multiple simple action items could be treated as "object with medium-complexity actions" or even as "object with complex actions."

Figure: Conversion statistics for database storage objects



AWS SCT에서 수동 변환 처리

평가 보고서에는 대상 데이터베이스의 데이터베이스 엔진으로 자동 변환할 수 없는 항목 목록이 포함되어 있습니다. 변환할 수 없는 각 항목의 경우 작업 항목 탭에는 작업 항목이 포함되어 있습니다.

다음과 같은 방법으로 평가 보고서의 작업 항목에 응답할 수 있습니다.

- 소스 데이터베이스 스키마 수정
- 대상 데이터베이스 스키마 수정

소스 스키마 수정

일부 항목의 경우 소스 데이터베이스의 데이터베이스 스키마를 자동으로 변환할 수 있는 스키마로 수정하는 것이 훨씬 쉬울 수 있습니다. 먼저 새로운 변경 내용이 애플리케이션 아키텍처와 호환되는지 확인하고 소스 데이터베이스의 스키마를 업데이트합니다. 마지막으로 업데이트된 스키마 정보로 프로젝트를 새로 고칩니다. 그런 다음 업데이트된 스키마를 변환하고 새 데이터베이스 마이그레이션 평가 보고서를 생성할 수 있습니다. 소스 스키마에서 변경된 항목에 대해서는 더 이상 작업 항목이 표시되지 않습니다.

이 프로세스의 장점은 소스 데이터베이스를 새로 고칠 때 항상 업데이트된 스키마를 사용할 수 있다는 것입니다.

대상 스키마 수정

일부 항목의 경우 변환된 스키마를 대상 데이터베이스에 적용한 다음 자동으로 변환할 수 없는 항목에 대해서는 대상 데이터베이스에 동일한 스키마 항목을 수동으로 추가하는 것이 훨씬 쉬울 수 있습니다. 스키마를 적용하여 대상 데이터베이스로 자동 변환할 수 있는 모든 스키마를 작성할 수 있습니다. 자세한 내용은 AWS SCT에서 변환된 스키마 저장 및 적용 섹션을 참조하세요.

대상 데이터베이스에 작성되는 스키마에는 자동으로 변환할 수 없는 항목이 포함되어 있지 않습니다. 대상 데이터베이스에 스키마를 적용한 후 소스 데이터베이스의 스키마와 동일한 스키마를 대상 데이터베이스에서 수동으로 생성할 수 있습니다. 데이터베이스 마이그레이션 평가 보고서의 작업 항목에는 동일한 스키마를 생성하는 방법에 대한 제안 사항이 포함되어 있습니다.

> **Warning**
>
> 대상 데이터베이스에서 스키마를 수동으로 생성하는 경우 수행하는 모든 수동 작업의 사본을 저장합니다. 프로젝트에서 변환된 스키마를 대상 데이터베이스에 다시 적용하면 수행한 수동 작업을 덮어씁니다.

경우에 따라서는 대상 데이터베이스에 동일한 스키마를 생성할 수 없습니다. 대상 데이터베이스의 엔진에서 사용할 수 있는 기능을 사용하려면 애플리케이션과 데이터베이스의 일부를 다시 설계해야 할 수 있습니다. 자동으로 변환할 수 없는 스키마를 그냥 무시해도 되는 경우도 있습니다.

AWS SCT에서 변환된 스키마 업데이트 및 새로 고침

AWS Schema Conversion Tool 프로젝트에서 소스 스키마와 대상 스키마를 모두 업데이트할 수 있습니다.

- 소스 – 소스 데이터베이스의 스키마를 업데이트하면 AWS SCT가 프로젝트의 스키마를 소스 데이터베이스의 최신 스키마로 바꿉니다. 이 기능을 사용하면 소스 데이터베이스의 스키마가 변경된 경우 프로젝트를 업데이트할 수 있습니다.
- 대상 – 대상 데이터베이스의 스키마를 업데이트하면 AWS SCT가 프로젝트의 스키마를 대상 데이터베이스의 최신 스키마로 바꿉니다. 대상 데이터베이스에 스키마를 적용하지 않은 경우에는 AWS SCT가 변환된 스키마를 프로젝트에서 지웁니다. 그런 다음 소스 데이터베이스의 스키마를 새로운 대상 데이터베이스로 변환할 수 있습니다.

데이터베이스에서 새로 고침을 선택하여 AWS SCT 프로젝트의 스키마를 업데이트할 수 있습니다.

AWS SCT에서 변환된 스키마 저장 및 적용

AWS Schema Conversion Tool에서 변환된 스키마를 생성할 경우(AWS SCT를 사용하여 스키마 변환 참조) 변환된 스키마를 대상 데이터베이스에 즉시 적용하지 않습니다. 대신 대상 데이터베이스에 적용할 준비가 될 때까지 변환된 스키마를 프로젝트에 로컬로 저장합니다. 이 기능을 사용하면 대상 데이터베이스 엔진으로 자동 변환할 수 없는 스키마 항목으로 작업할 수 있습니다. 자동으로 변환할 수 없는 항목에 대한 자세한 내용은 AWS SCT를 사용하여 마이그레이션 평가 보고서 작성 섹션을 참조하세요.

선택적으로 대상 데이터베이스에 스키마를 적용하기 전에 도구에서 변환된 스키마를 SQL 스크립트로 파일에 저장하도록 할 수 있습니다. 또한 도구에서 변환된 스키마를 대상 데이터베이스에 직접 적용하도록 할 수도 있습니다.

변환된 스키마를 파일에 저장

변환된 스키마를 텍스트 파일에 SQL 스크립트로 저장할 수 있습니다. 이 방법을 사용하면 AWS SCT에서 생성된 SQL 스크립트를 수정하여 도구가 자동으로 변환할 수 없는 항목을 처리할 수 있습니다. 그런 다음, 대상 DB 인스턴스에서 업데이트된 스크립트를 실행하여 변환된 스키마를 대상 데이터베이스에 적용할 수 있습니다.

변환된 스키마를 SQL 스크립트로 저장하려면

1. 스키마를 선택하고 컨텍스트(마우스 오른쪽 버튼 클릭) 메뉴를 엽니다.
2. SQL로 저장을 선택합니다.
3. 파일 이름을 입력하고 저장을 선택합니다.
4. 다음 옵션 중 하나를 사용하여 변환된 스키마를 저장합니다.

    - 단일 파일

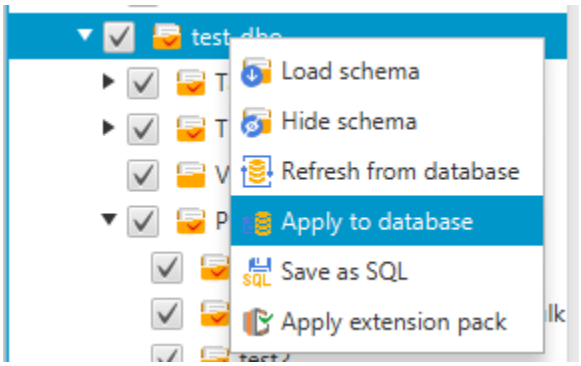
- Single file per stage
- Single file per statement

SQL 스크립트의 형식을 선택하려면

1. 설정 메뉴에서 프로젝트 설정을 선택합니다.
2. Save scripts를 선택합니다.
3. 공급업체에서 데이터베이스 플랫폼을 선택합니다.
4. Save SQL scripts to에서 데이터베이스 스키마 스크립트를 저장할 방법을 선택합니다.
5. 확인을 선택하여 설정을 저장합니다.

변환된 스키마 적용

변환된 스키마를 대상 데이터베이스에 적용할 준비가 되면 프로젝트의 오른쪽 패널에서 스키마 요소를 선택합니다. 스키마 요소의 컨텍스트(마우스 오른쪽 버튼 클릭) 메뉴를 열고 아래와 같이 Apply to database를 선택합니다.



확장 팩 스키마

변환된 스키마를 대상 DB 인스턴스에 처음 적용하면 AWS SCT가 대상 DB 인스턴스에 추가적인 스키마를 추가합니다. 이 스키마는 변환된 스키마를 대상 DB 인스턴스에 쓸 때 필요한 소스 데이터베이스의 시스템 함수를 구현합니다. 이 스키마를 확장 팩 스키마라고 합니다.

확장 팩 스키마를 수정하지 않도록 합니다. 그렇지 않으면 대상 DB 인스턴스에 작성된 변환된 스키마에서 예상치 못한 결과가 발생할 수 있습니다. 스키마가 대상 DB 인스턴스로 완전히 마이그레이션되어 AWS SCT가 더 이상 필요하지 않은 경우 확장 팩 스키마를 삭제할 수 있습니다.

확장 팩 스키마는 소스 데이터베이스에 따라 다음과 같이 이름이 지정됩니다.

- Greenplum: `aws_greenplum_ext`
- Microsoft SQL Server: `aws_sqlserver_ext`
- Netezza: `aws_netezza_ext`
- Oracle: `aws_oracle_ext`
- Snowflake: `aws_snowflake_ext`
- Teradata: `aws_teradata_ext`
- Vertica: `aws_vertica_ext`

자세한 내용은 AWS SCT 확장 팩 사용 섹션을 참조하세요.

Python 라이브러리

Amazon Redshift에서 사용자 지정 함수를 생성하려면 Python 언어를 사용합니다. AWS SCT 확장 팩을 사용하여 Amazon Redshift 데이터베이스용 Python 라이브러리를 설치합니다. 자세한 내용은 AWS SCT 확장 팩 사용 섹션을 참조하세요.

AWS SCT를 사용하여 Amazon Redshift 최적화

AWS Schema Conversion Tool을 사용하여 Amazon Redshift 데이터베이스를 최적화할 수 있습니다. Amazon Redshift 데이터베이스를 소스로 사용하고 테스트 Amazon Redshift 데이터베이스를 대상으로 사용하는 경우 AWS SCT는 데이터베이스를 최적화하기 위해 정렬 키와 배포 키를 사용하도록 권장합니다.

Amazon Redshift 데이터베이스 최적화

다음 절차에 따라 Amazon Redshift 데이터베이스를 최적화합니다.

Amazon Redshift 데이터베이스를 최적화하려면

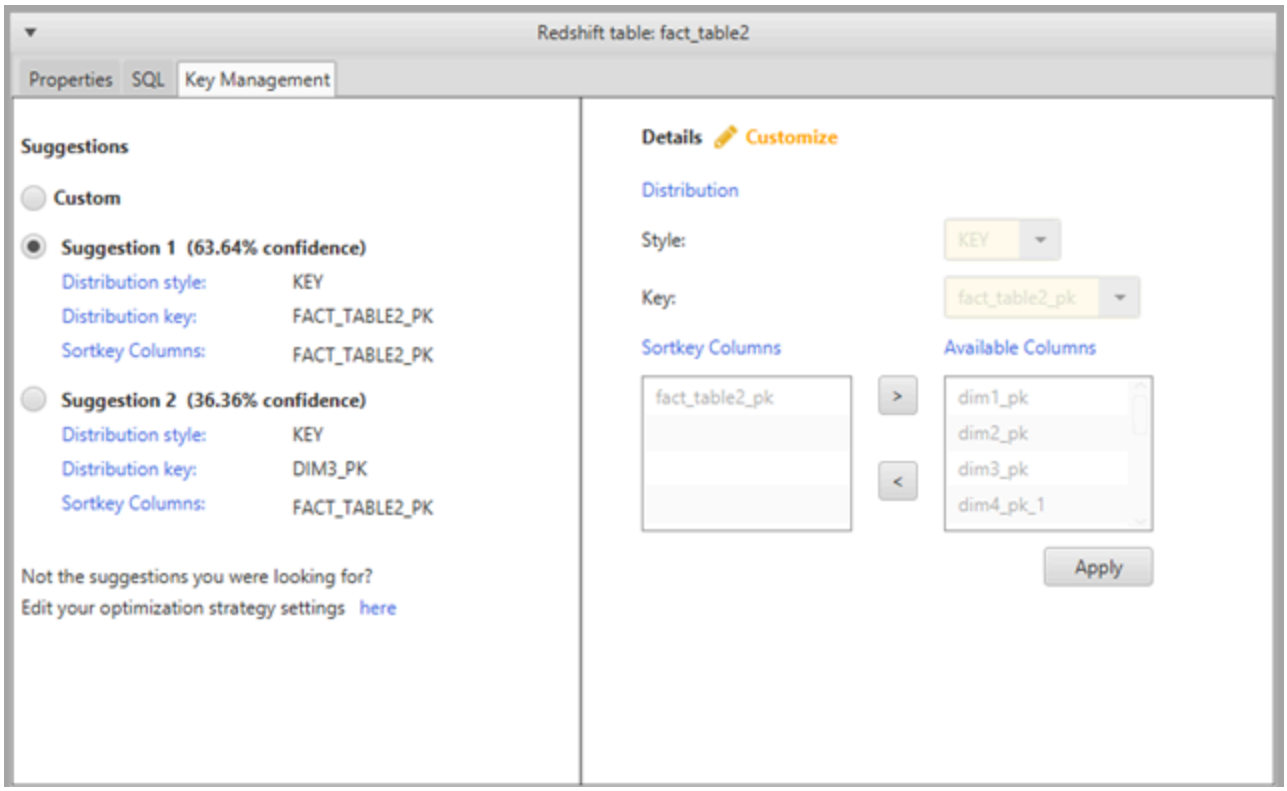
1. Amazon Redshift 클러스터의 수동 스냅샷을 백업으로 생성합니다. Amazon Redshift 클러스터를 최적화하고 변경 내용을 테스트한 후에 스냅샷을 삭제할 수 있습니다. 자세한 내용은 Amazon Redshift 스냅샷 섹션을 참조하세요.
2. 프로젝트의 왼쪽 패널에서 변환할 스키마 객체를 선택합니다. 객체의 컨텍스트(마우스 오른쪽 버튼 클릭) 메뉴를 열고 Collect Statistics를 선택합니다.

   AWS SCT는 통계를 사용하여 정렬 키 및 배포 키를 제안합니다.

3. 프로젝트의 왼쪽 패널에서 최적화할 스키마 객체를 선택합니다. 객체의 컨텍스트(마우스 오른쪽 버튼 클릭) 메뉴를 열고 Run Optimization을 선택합니다.

   AWS SCT가 정렬 키 및 배포 키를 제안합니다.

4. 제안 사항을 검토하려면 프로젝트의 왼쪽 패널에서 스키마 아래에 있는 테이블 노드를 확장하고 테이블을 선택합니다. 다음과 같이 키 관리 탭을 선택합니다.



   왼쪽 창에는 주요 제안 사항과 각 제안 사항에 대한 신뢰도가 포함되어 있습니다. 제안 사항 중 하나를 선택하거나 오른쪽 창에서 편집하여 키를 사용자 지정할 수 있습니다.

5. 최적화 제안 사항이 포함된 보고서를 생성할 수 있습니다. 보고서를 생성하려면 다음과 같이 수행합니다.

   a. 프로젝트의 왼쪽 패널에서 최적화한 스키마 객체를 선택합니다. 객체의 컨텍스트(마우스 오른쪽 버튼 클릭) 메뉴를 열고 보고서 생성을 선택합니다.

      보고서가 기본 창에서 열리고 요약 탭이 나타납니다. 최적화 제안 사항이 있는 객체의 개수가 보고서에 표시됩니다.

   b. 주요 제안 사항을 보고서 형식으로 보려면 작업 항목 탭을 선택합니다.

c. 최적화 보고서의 로컬 사본을 PDF 파일 또는 쉼표로 구분된 값(CSV) 파일로 저장할 수 있습니다. CSV 파일에는 작업 항목 정보만 포함되어 있습니다. PDF 파일에는 요약 정보와 작업 항목 정보가 모두 수록돼 있습니다.

6. 제안된 최적화를 데이터베이스에 적용하려면 프로젝트의 오른쪽 패널에서 객체를 선택합니다. 객체의 컨텍스트(마우스 오른쪽 버튼 클릭) 메뉴를 열고 Apply to database를 선택합니다.

AWS Schema Conversion Tool을 사용하여 추출, 전환, 적재(ETL) 프로세스 변환

AWS Schema Conversion Tool(AWS SCT)을 사용하여 추출, 전환, 적재(ETL) 프로세스를 마이그레이션할 수 있습니다. 이 유형의 마이그레이션에는 ETL 관련 비즈니스 로직의 변환이 포함됩니다. 이 로직은 소스 데이터 웨어하우스 내부 또는 별도로 실행되는 외부 스크립트에 있을 수 있습니다.

현재 AWS SCT에서는 다음 표에 나와 있는 것처럼 객체에 대한 ETL 스크립트를 AWS Glue 및 Amazon Redshift RSQL로 변환하는 기능을 지원합니다.

소스	대상
Informatica ETL 스크립트	Informatica
Microsoft SQL Server Integration Services(SSIS) ETL 패키지	AWS Glue 또는 AWS Glue Studio
Teradata Basic Teradata Query(BTEQ)의 임베디드 명령이 포함된 쉘 스크립트	Amazon Redshift RSQL
Teradata BTEQ ETL 스크립트	AWS Glue 또는 Amazon Redshift RSQL
Terata FastExport 작업 스크립트	Amazon Redshift RSQL
Teradata FastLoad 작업 스크립트	Amazon Redshift RSQL
Teradata MultiLoad 작업 스크립트	Amazon Redshift RSQL

주제

- AWS SCT를 사용하여 ETL 프로세스를 AWS Glue로 변환
- AWS SCT에서 AWS Glue용 Python API를 사용하여 ETL 프로세스 변환
- AWS SCT를 사용하여 Informatica ETL 스크립트 변환
- AWS SCT를 사용하여 SSIS를 AWS Glue로 변환
- AWS SCT를 사용하여 SSIS를 AWS Glue Studio로 변환
- AWS SCT를 사용하여 Teradata BTEQ 스크립트를 Amazon Redshift RSQL로 변환

- AWS SCT를 사용하여 임베디드 Teradata BTEQ 명령이 있는 쉘 스크립트를 Amazon Redshift RSQL로 변환
- AWS SCT를 사용하여 Teradata FastExport 작업 스크립트를 Amazon Redshift RSQL로 변환
- AWS SCT를 사용하여 Teradata FastLoad 작업 스크립트를 Amazon Redshift RSQL로 변환
- AWS SCT를 사용하여 Teradata MultiLoad 작업 스크립트를 Amazon Redshift RSQL로 변환

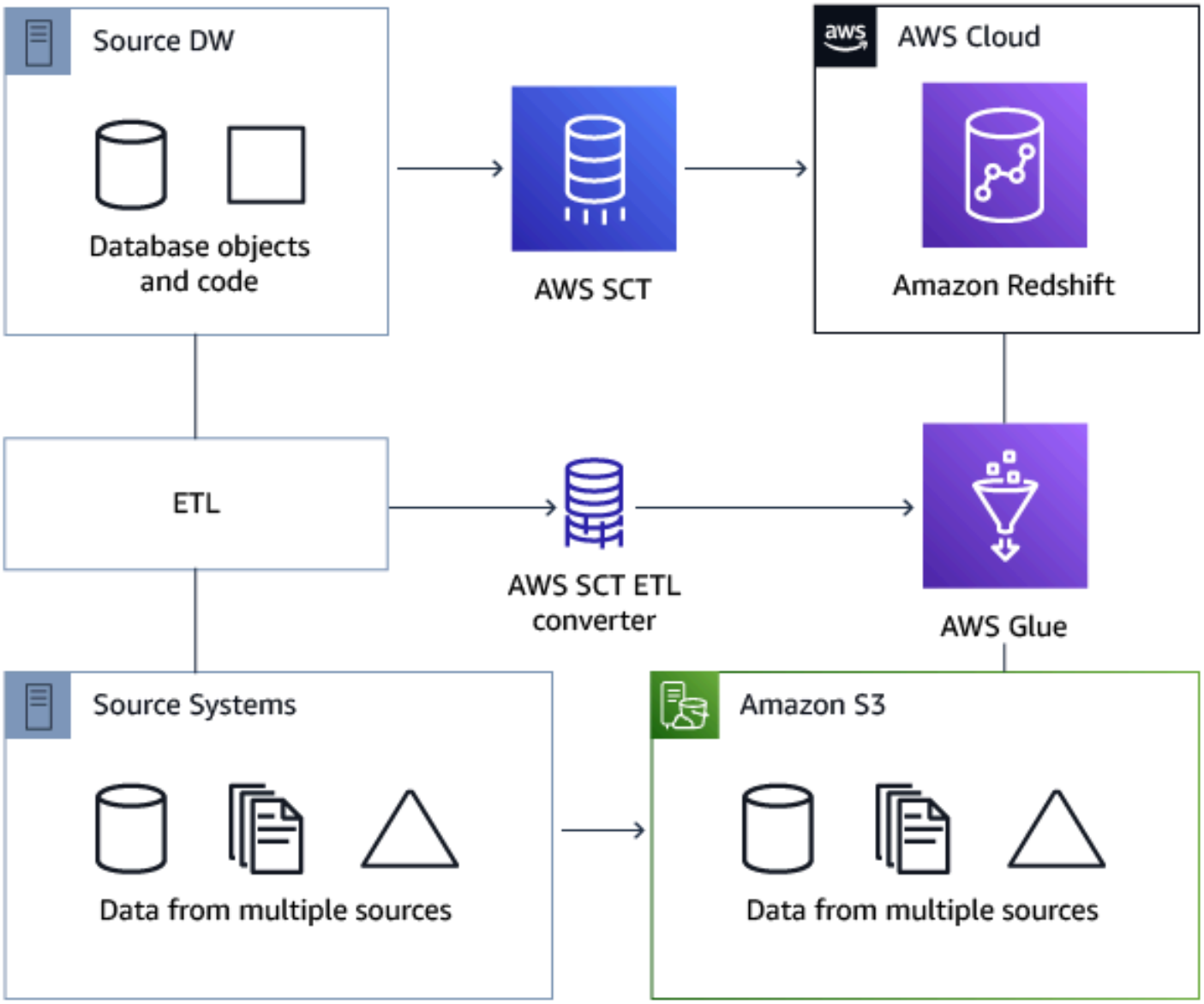
AWS SCT를 사용하여 ETL 프로세스를 AWS Glue로 변환

다음은 AWS SCT를 사용하여 ETL 스크립트를 AWS Glue로 변환하는 프로세스의 개요를 확인할 수 있습니다. 이 예제에서는 소스 데이터베이스 및 데이터 웨어하우스에서 사용되는 ETL 프로세스와 함께 Oracle 데이터베이스를 Amazon Redshift로 변환합니다.

주제

- 필수 조건
- AWS Glue 데이터 카탈로그 이해
- AWS Glue와 함께 AWS SCT를 사용하여 변환할 경우 제한 사항
- 1단계: 새 프로젝트 생성
- 2단계: AWS Glue 작업 생성

다음 아키텍처 다이어그램은 ETL 스크립트의 AWS Glue로의 변환을 포함하는 예제 데이터베이스 마이그레이션 프로젝트를 보여줍니다.



필수 조건

시작하기 전에 다음을 수행하세요.

- AWS로 마이그레이션하려는 모든 소스 데이터베이스를 마이그레이션합니다.
- 대상 데이터 웨어하우스를 AWS로 마이그레이션합니다.
- ETL 프로세스와 관련된 모든 코드의 목록을 수집합니다.
- 각 데이터베이스에 필요한 모든 연결 정보 목록을 수집합니다.

또한 AWS Glue에 사용자를 대신하여 다른 AWS 서비스에 액세스할 수 있는 권한이 필요합니다. AWS Identity and Access Management(IAM)을 사용하여 그러한 권한을 제공합니다. AWS Glue에 대한 IAM 정책을 생성했는지 확인합니다. 자세한 내용은 AWS Glue 개발자 안내서의 AWS Glueservice를 위한 IAM 정책 생성을 참조하세요.

AWS Glue 데이터 카탈로그 이해

AWS Glue는 변환 프로세스의 일부로 소스 및 대상 데이터베이스와 관련된 정보를 로드합니다. 이 정보는 트리라는 구조를 통해 범주별로 분류됩니다. 구조에는 다음이 포함됩니다.

- 연결 - 연결 파라미터
- 크롤러 - 크롤러 목록으로, 각 스키마당 하나의 크롤러가 지정됩니다.
- 데이터베이스 - 테이블을 보관하는 컨테이너
- 테이블 - 테이블의 데이터를 나타내는 메타데이터 정의
- ETL 작업 - ETL 작업을 수행하는 비즈니스 로직
- 트리거 - AWS Glue에서 ETL 작업이 실행되는 경우(온디맨드, 일정에 따라 또는 작업 이벤트에서 트리거)를 제어하는 로직

AWS Glue 데이터 카탈로그는 데이터의 위치, 스키마 및 런타임 지표의 인덱스입니다. AWS Glue 및 AWS SCT를 사용하는 경우 AWS Glue 데이터 카탈로그에는 AWS Glue에서 ETL 작업의 소스 및 대상으로 사용되는 데이터에 대한 참조가 포함됩니다. 데이터 웨어하우스를 생성하려면 이 데이터를 분류해야 합니다.

Data Catalog에서 이 정보를 사용하여 ETL 작업을 생성하고 모니터링합니다. 일반적으로 크롤러를 실행하여 데이터의 목록을 데이터 스토어로 가져가지만 메타데이터 테이블을 Data Catalog로 추가하는 다른 방법이 있습니다.

데이터 카탈로그에 테이블을 정의할 경우 해당 테이블을 데이터베이스에 추가합니다. 데이터베이스는 AWS Glue의 테이블을 정리하는 데 사용됩니다.

AWS Glue와 함께 AWS SCT를 사용하여 변환할 경우 제한 사항

AWS Glue과 함께 AWS SCT를 사용하여 변환할 경우 다음과 같은 제한 사항이 적용됩니다

리소스	기본 한도
각 계정의 데이터베이스 수	10,000개

각 데이터베이스의 테이블 수	100,000건
각 테이블의 파티션 수	1,000,000
각 테이블의 테이블 버전 수	100,000건
각 계정의 테이블 수	1,000,000
각 계정의 파티션 수	10,000,000
각 계정의 테이블 버전 수	1,000,000
각 계정의 연결 수	1,000
각 계정의 크롤러 수	25
각 계정의 작업 수	25
각 계정의 트리거 수	25
각 계정의 동시 작업 실행 개수	30
각 작업의 동시 작업 실행 개수	3
각 트리거의 작업 수	10
각 계정의 개발 엔드포인트 수	5
개발 엔드포인트에서 한 번에 사용할 수 있는 최대 데이터 처리 장치(DPU) 수	5
역할에서 한 번에 사용할 수 있는 최대 DPU 수	100

데이터베이스 이름 길이	무제한 Apache Hive와 같은 다른 메타데이터 스토어의 호환성을 고려해 이름은 소문자를 사용하도록 변경됩니다. Amazon Athena의 데이터베이스에 액세스하려고 할 경우에는 영숫자 문자와 밑줄만 포함된 이름을 제공합니다.
연결 이름 길이	무제한
크롤러 이름 길이	무제한

1단계: 새 프로젝트 생성

새 프로젝트를 생성하려면 다음과 같은 상위 단계를 따릅니다.

1. AWS SCT에서 새 프로젝트를 생성합니다. 자세한 내용은 AWS SCT 프로젝트 생성 섹션을 참조하세요.
2. 소스 및 대상 데이터베이스를 프로젝트에 추가합니다. 자세한 내용은 AWS SCT 프로젝트에 데이터베이스 서버 추가 섹션을 참조하세요.

   대상 데이터베이스 연결 설정에서 AWS Glue 사용을 선택했는지 확인합니다. 이 작업을 수행하려면 AWS Glue 탭을 선택합니다. Copy from AWS profile에서 사용하려는 프로필을 선택합니다. 프로필은 AWS 액세스 키, 비밀 키, Amazon S3 버킷 폴더를 자동으로 채워야 합니다. 자동으로 채워지지 않는 경우에는 이 정보를 직접 입력합니다. 확인을 선택하면 AWS Glue에서 객체를 분석하고 메타데이터를 AWS Glue 데이터 카탈로그에 로드합니다.

   보안 설정에 따라 계정에 서버의 일부 스키마에 대한 충분한 권한이 없다는 경고 메시지가 표시될 수 있습니다. 사용 중인 스키마에 대한 액세스 권한이 있다면 이 메시지를 무시해도 됩니다.
3. ETL을 가져올 준비를 완료하려면 소스 및 대상 데이터베이스에 연결합니다. 이를 위해서는 소스 또는 대상 메타데이터 트리에서 데이터베이스를 선택한 다음 Connect to the Server를 선택합니다.

AWS Glue가 ETL 변환에 도움이 되도록 소스 데이터베이스 서버와 대상 데이터베이스 서버에 각각 데이터베이스를 생성합니다. 대상 서버의 데이터베이스에는 AWS Glue 데이터 카탈로그가 포함되어 있습니다. 특정 객체를 찾으려면 소스 또는 대상 패널에서 검색 기능을 사용합니다.

특정 객체가 어떻게 변환되는지 보려면 변환할 항목을 찾은 다음 컨텍스트(마우스 오른쪽 버튼 클릭) 메뉴에서 스키마 변환을 선택합니다. AWS SCT가 선택된 이 개체를 스크립트로 변환합니다.

오른쪽 패널의 스크립트 폴더에서 변환된 스크립트를 검토할 수 있습니다. 현재, 스크립트는 AWS SCT 프로젝트의 일부로만 사용할 수 있는 가상 객체입니다.

변환된 스크립트로 AWS Glue 작업을 생성하려면 Amazon S3에 스크립트를 업로드합니다. Amazon S3에 스크립트를 업로드하려면 해당 스크립트를 선택한 다음 컨텍스트(마우스 오른쪽 버튼 클릭) 메뉴에서 Save to S3을 선택합니다.

2단계: AWS Glue 작업 생성

Amazon S3에 스크립트를 저장한 후 해당 스크립트를 선택한 다음 Configure AWS Glue Job을 선택하면 AWS Glue 작업을 구성할 마법사를 열 수 있습니다. 마법사를 사용하면 이 작업을 더 쉽게 설정할 수 있습니다.

1. 마법사의 첫 번째 탭인 Design Data Flow에서 이 작업에 포함할 실행 전략과 스크립트 목록을 선택할 수 있습니다. 각 스크립트의 파라미터를 선택할 수 있습니다. 올바른 순서로 실행되도록 스크립트를 재배열할 수도 있습니다.
2. 두 번째 탭에서는 작업의 이름을 지정하고 AWS Glue에 대한 설정을 직접 구성할 수 있습니다. 이 화면에서 다음 설정을 구성할 수 있습니다.
   - AWS Identity and Access Management(IAM) 역할
   - 스크립트 파일 이름 및 파일 경로
   - Amazon S3 관리형 키(SSE-S3)와 함께 서버 측 암호화를 사용하여 스크립트 암호화
   - 임시 디렉터리
   - 생성된 Python 라이브러리 경로
   - 사용자 Python 라이브러리 경로
   - 종속된 .jar 파일의 경로
   - 참조된 파일 경로
   - 각 작업 실행을 위한 동시 DPU 수
   - 최대 동시성
   - 작업 제한 시간(분)

- 지연 알림 임계값(분)
- 재시도 횟수
- 보안 구성
- 서버 측 암호화

3. 세 번째 단계 또는 탭에서는 대상 엔드포인트에 대해 구성된 연결을 선택합니다.

작업 구성을 마치면 AWS Glue 데이터 카탈로그의 ETL 작업 아래에 해당 작업이 표시됩니다. 작업을 선택하면 설정이 표시되므로 해당 설정을 검토하거나 편집할 수 있습니다. AWS Glue에서 새 작업을 생성하려면 작업의 컨텍스트(마우스 오른쪽 버튼 클릭) 메뉴에서 Create AWS Glue Job을 선택합니다. 이렇게 하면 스키마 정의가 적용됩니다. 디스플레이를 새로 고치려면 컨텍스트(마우스 오른쪽 버튼 클릭) 메뉴에서 데이터베이스에서 새로 고침을 선택합니다.

이제 AWS Glue 콘솔에서 작업을 볼 수 있습니다. 이렇게 하려면 AWS Management Console에 로그인하여 https://console.aws.amazon.com/glue/에서 AWS Glue 콘솔을 엽니다.

새 작업을 테스트하여 제대로 작동하는지 확인할 수 있습니다. 이를 위해서는 먼저 소스 테이블의 데이터를 확인하고 대상 테이블이 비어 있는지 확인합니다. 작업을 실행하고 다시 확인합니다. AWS Glue 콘솔에서 오류 로그를 볼 수 있습니다.

AWS SCT에서 AWS Glue용 Python API를 사용하여 ETL 프로세스 변환

다음 섹션에서는 Python에서 AWS Glue API 작업을 호출하는 변환에 대한 설명을 확인할 수 있습니다. 자세한 내용은 AWS Glue개발자 안내서에서 Python으로 AWS Glue ETL 스크립트 프로그래밍을 참조하세요.

주제

- 1단계: 데이터베이스 생성
- 2단계: 연결 생성
- 3단계: AWS Glue 크롤러 생성

1단계: 데이터베이스 생성

첫 번째 단계는 AWS SDK API를 사용하여 AWS Glue 데이터 카탈로그에 새 데이터베이스를 생성하는 것입니다. 데이터 카탈로그에 테이블을 정의할 경우 해당 테이블을 데이터베이스에 추가합니다. 데이터베이스는 AWS Glue에서 테이블을 구성하는 데 사용됩니다.

다음 예제는 AWS Glue용 Python API의 `create_database` 메서드를 보여줍니다.

```
response = client.create_database(
    DatabaseInput={
        'Name': 'database_name',
        'Description': 'description',
        'LocationUri': 'string',
        'Parameters': {
            'parameter-name': 'parameter value'
        }
    }
)
```

Amazon Redshift를 사용하는 경우 데이터베이스 이름은 다음과 같은 형식으로 구성됩니다.

```
{redshift_cluster_name}_{redshift_database_name}_{redshift_schema_name}
```

이 예제에서 Amazon Redshift 클러스터의 전체 이름은 다음과 같습니다.

```
rsdbb03.apq1mpqso.us-west-2.redshift.amazonaws.com
```

다음은 올바른 형식의 데이터베이스 이름의 예입니다. 이 경우 `rsdbb03`은 이름으로, 클러스터 엔드포인트 전체 이름의 첫 번째 부분입니다. 데이터베이스는 `dev`로 이름이 지정되고 스키마는 `ora_glue`입니다.

```
rsdbb03_dev_ora_glue
```

2단계: 연결 생성

AWS SDK API를 사용하여 데이터 카탈로그에 새 연결을 생성합니다.

다음 예제는 AWS Glue용 Python API의 `create_connection` 메서드 사용 방법을 보여줍니다.

```
response = client.create_connection(
    ConnectionInput={
        'Name': 'Redshift_abcde03.aabbcc112233.us-west-2.redshift.amazonaws.com_dev',
        'Description': 'Created from SCT',
        'ConnectionType': 'JDBC',
        'ConnectionProperties': {
            'JDBC_CONNECTION_URL': 'jdbc:redshift://aabbcc03.aabbcc112233.us-
west-2.redshift.amazonaws.com:5439/dev',
            'USERNAME': 'user_name',
            'PASSWORD': 'password'
        },
        'PhysicalConnectionRequirements': {
            'AvailabilityZone': 'us-west-2c',
            'SubnetId': 'subnet-a1b23c45',
            'SecurityGroupIdList': [
                'sg-000a2b3c', 'sg-1a230b4c', 'sg-aba12c3d', 'sg-1abb2345'
            ]
        }
    }
)
```

create_connection에서 사용되는 파라미터는 다음과 같습니다.

- Name(UTF-8 문자열) - 필수. Amazon Redshift의 경우 연결 이름은 Redshift_*<Endpoint-name>*_*<redshift-database-name>*과 같이 구성됩니다. 예: Redshift_abcde03_dev
- Description(UTF-8 문자열) - 연결에 대한 설명입니다.
- ConnectionType(UTF-8 문자열) - 필수. 연결의 유형입니다. 현재 JDBC만 지원하고 SFTP는 지원하지 않습니다.
- ConnectionProperties(dict) – 필수. JDBC 연결 URL, 사용자 이름 및 암호를 포함하여 이 연결의 파라미터로 사용되는 키-값 쌍의 목록입니다.
- PhysicalConnectionRequirements(dict) - 물리적 연결 요구 사항이며 다음을 포함합니다.
  - SubnetId(UTF-8 문자열) - 연결에 사용되는 서브넷의 ID입니다.
  - SecurityGroupIdList(list) – 연결에 사용되는 보안 그룹 ID 목록입니다.
  - AvailabilityZone(UTF-8 문자열) - 필수. 엔드포인트가 포함된 가용 영역입니다. 이 파라미터는 이제 사용되지 않습니다.

3단계: AWS Glue 크롤러 생성

다음으로 AWS Glue 크롤러를 생성하여 AWS Glue 카탈로그를 채웁니다. 자세한 내용은 AWS Glue 개발자 안내서의 크롤러를 사용하여 데이터 카탈로그 작성을 참조하세요.

크롤러를 추가하는 첫 번째 단계는 AWS SDK API를 사용하여 데이터 카탈로그에 새 데이터베이스를 생성하는 것입니다. 시작하기 전에 먼저 `delete_crawler` 작업을 사용하여 이전 버전의 데이터베이스를 삭제해야 합니다.

크롤러를 만들 때는 몇 가지 고려 사항이 적용됩니다.

- 크롤러 이름은 *<redshift_node_name>*_*<redshift_database_name>*_*<redshift_shema_name>* 형식을 사용합니다. 예: `abcde03_dev_ora_glue`
- 이미 존재하는 IAM 역할을 사용합니다. IAM 역할 생성에 대한 자세한 내용은 IAM 사용 설명서의 IAM 역할 생성을 참조하세요.
- 이전 단계에서 생성한 데이터베이스의 이름을 사용합니다.
- 필수 `ConnectionName` 파라미터를 사용합니다.
- `path` 파라미터에는 JDBC 대상의 경로를 사용합니다. 예: `dev/ora_glue/%`

다음 예제에서는 기존 크롤러를 삭제한 다음 AWS Glue용 Python API를 사용하여 새 크롤러를 생성합니다.

```
response = client.delete_crawler(
    Name='crawler_name'
)

response = client.create_crawler(
    Name='crawler_name',
    Role= 'IAM_role',
    DatabaseName='database_name',
    Description='string',
    Targets={
        'S3Targets': [
            {
                'Path': 'string',
                'Exclusions': [
                    'string',
                ]
```

```
            },
        ],
        'JdbcTargets': [
            {
                'ConnectionName': 'ConnectionName',
                'Path': 'Include_path',
                'Exclusions': [
                    'string',
                ]
            },
        ]
    },
    Schedule='string',
    Classifiers=[
        'string',
    ],
    TablePrefix='string',
    SchemaChangePolicy={
        'UpdateBehavior': 'LOG'|'UPDATE_IN_DATABASE',
        'DeleteBehavior': 'LOG'|'DELETE_FROM_DATABASE'|'DEPRECATE_IN_DATABASE'
    },
    Configuration='string'
)
```

크롤러를 생성한 후 실행하여 하나 이상의 데이터 스토어에 연결하고 데이터 구조를 결정하며 테이블을 데이터 카탈로그로 작성합니다. 다음과 같이 일정에 따라 크롤러를 실행할 수 있습니다.

```
response = client.start_crawler(
    Name='string'
)
```

이 예제에서는 Amazon Redshift를 대상으로 사용합니다. Amazon Redshift 데이터 형식은 크롤러가 실행된 후 다음과 같은 방식으로 AWS Glue 데이터 형식에 매핑됩니다.

Amazon Redshift 데이터 형식	AWS Glue 데이터 유형
smallint	smallint
integer	int
bigint	bigint

decimal	decimal(18,0)
decimal(p,s)	decimal(p,s)
real	double
double precision	double
부울	부울
char	문자열
varchar	문자열
varchar(n)	문자열
날짜	날짜
timestamp	timestamp
timestamptz	timestamp

AWS SCT를 사용하여 Informatica ETL 스크립트 변환

AWS SCT 명령줄 인터페이스(CLI)를 사용하여 Informatica ETL 스크립트를 변환하면 새 대상 데이터베이스에서 해당 스크립트를 사용할 수 있습니다. 이 변환에는 세 가지 주요 단계가 포함됩니다. 먼저 AWS SCT가 Informatica 객체에 포함된 SQL 코드를 변환합니다. 그런 다음 AWS SCT가 프로젝트에 지정된 마이그레이션 규칙에 따라 데이터베이스 객체의 이름을 변경합니다. 마지막으로 AWS SCT가 Informatica ETL 스크립트의 연결을 새로운 대상 데이터베이스로 리디렉션합니다.

Informatica ETL 스크립트를 AWS SCT 데이터베이스 변환 프로젝트의 일부로 변환할 수 있습니다. Informatica ETL 스크립트를 변환할 때는 소스 및 대상 데이터베이스를 프로젝트에 추가해야 합니다.

Informatica ETL 스크립트를 변환하려면 AWS SCT 버전 1.0.667 이상을 사용해야 합니다. 또한 AWS SCT의 명령줄 인터페이스를 숙지해야 합니다. 자세한 내용은 AWS SCT CLI 레퍼런스 섹션을 참조하세요.

AWS SCT를 사용하여 Informatica ETL 스크립트를 변환하려면

1. 새 AWS SCT CLI 스크립트를 생성하거나 기존 시나리오 템플릿을 편집합니다. 예를 들어 `InformaticConversionTemplate.scts` 템플릿을 다운로드하여 편집할 수 있습니다. 자세한 내용은 CLI 시나리오 가져오기 섹션을 참조하세요.
2. 소스 및 대상 데이터베이스에 필요한 JDBC 드라이버를 다운로드합니다. `SetGlobalSettings` 명령을 사용하여 이러한 드라이버의 위치를 지정합니다. 또한 AWS SCT가 로그 파일을 저장할 수 있는 폴더도 지정합니다.

   다음 코드 예제는 Oracle 및 PostgreSQL 드라이버의 경로를 AWS SCT 설정에 추가하는 방법을 보여줍니다. 이 코드 예제를 실행하면 AWS SCT가 `C:\sct_log` 폴더에 로그 파일을 저장합니다. 또한 AWS SCT가 콘솔 로그 파일을 `C:\Temp\oracle_postgresql` 폴더에 저장합니다.

   ```
   SetGlobalSettings
    -save: 'true'
    -settings: '{"oracle_driver_file": "C:\\drivers\\ojdbc8.jar",
    "postgresql_driver_file": "C:\\drivers\\postgresql-42.2.19.jar" }'
   /

   SetGlobalSettings
    -save: 'false'
    -settings: '{
   "log_folder": "C:\\sct_log",
   "console_log_folder": "C:\\Temp\\oracle_postgresql"}'
   /
   ```
3. 새 AWS SCT 프로젝트를 생성합니다. 프로젝트 이름과 위치를 입력합니다.

   다음 코드 예제는 `C:\Temp` 폴더에 `oracle_postgresql` 프로젝트를 만듭니다.

   ```
   CreateProject
    -name: 'oracle_postgresql'
    -directory: 'C:\Temp'
   /
   ```
4. 소스 및 대상 데이터베이스에 대한 연결 정보를 추가합니다.

   다음 코드 예제는 Oracle 및 PostgreSQL 데이터베이스를 AWS SCT 프로젝트의 소스 및 대상으로 추가합니다.

   ```
   AddSource
   ```

```
 -password: 'source_password'
 -port: '1521'
 -vendor: 'ORACLE'
 -name: 'ORACLE'
 -host: 'source_address'
 -database: 'ORCL'
 -user: 'source_user'
/
AddTarget
 -database: 'postgresql'
 -password: 'target_password'
 -port: '5432'
 -vendor: 'POSTGRESQL'
 -name: 'POSTGRESQL'
 -host: 'target_address'
 -user: 'target_user'
/
```

위 예제에서 *source_user* 및 *target_user*를 데이터베이스 사용자의 이름으로 바꿉니다. 그 다음으로, *source_password* 및 *target_password*를 암호로 바꿉니다. *source_address* 및 *target_address*에 소스 및 대상 데이터베이스 서버의 IP 주소를 입력합니다.

Oracle 데이터베이스 버전 19 이상에 연결하려면 AddSource 명령에 Oracle 서비스 이름을 사용합니다. 이렇게 하려면 -connectionType 파라미터를 추가하고 값을 'basic_service_name'으로 설정합니다. 그런 다음 -servicename 파라미터를 추가하고 값을 Oracle 서비스 이름으로 설정합니다. AddSource 명령에 대한 자세한 내용은 AWS Schema Conversion Tool 명령 참조를 확인하세요.

5. 각 소스 데이터베이스 스키마의 대상 데이터베이스 엔진을 정의하는 새 AWS SCT 매핑 규칙을 생성합니다. 자세한 내용은 AWS SCT에서 매핑 규칙 생성 섹션을 참조하세요.

   다음 코드 예제는 모든 소스 Oracle 데이터베이스 스키마를 포함하고 PostgreSQL을 마이그레이션 대상으로 정의하는 매핑 규칙을 생성합니다.

```
AddServerMapping
 -sourceTreePath: 'Servers.ORACLE'
 -targetTreePath: 'Servers.POSTGRESQL'
/
```

6. Informatica 소스 및 대상 XML 파일에 대한 연결 정보를 추가합니다.

다음 코드 예제는 C:\Informatica_source 및 C:\Informatica_target 폴더에서 Informatica XML 파일을 추가합니다.

```
AddSource
 -name: 'INFA_SOURCE'
 -vendor: 'INFORMATICA'
 -mappingsFolder: 'C:\Informatica_source'
/
AddTarget
 -name: 'INFA_TARGET'
 -vendor: 'INFORMATICA'
 -mappingsFolder: 'C:\Informatica_target'
/
```

7. 소스 Informatica XML 파일에 대한 대상 Informatica XML 파일을 정의하는 또 다른 매핑 규칙을 생성합니다.

   다음 코드 예제는 이전 예제에서 사용된 소스 및 대상 Informatica XML 파일을 포함하는 매핑 규칙을 생성합니다.

```
AddServerMapping
-sourceTreePath: 'ETL.INFA_SOURCE'
-targetTreePath: 'ETL.INFA_TARGET'
/
```

8. Informatica 연결 이름 참조에 해당하는 데이터베이스 서버 연결을 지정합니다.

   다음 코드 예제는 소스에서 새로운 대상 데이터베이스로의 Informatica ETL 스크립트 리디렉션을 구성합니다. 또한 이 예제에서 연결 변수도 구성합니다.

```
ConfigureInformaticaConnectionsRedirect
 -treePath: 'ETL.INFA_SOURCE.Files'
 -connections: '{
 "ConnectionNames": [
 {
  "name": "Oracle_src",
  "newName": "postgres",
  "treePath": "Servers.ORACLE"
 }
 ]
 "ConnectionVariables": [
```

```
 {
         "name": "$Source",
         "treePath": "Servers.ORACLE"
    }
    ]
 }'
/
```

9. 소스 데이터베이스 스키마와 Informatica ETL 스크립트를 변환합니다.

   다음 코드 예제는 모든 소스 Oracle 데이터베이스 스키마와 Informatica XML 파일을 변환합니다.

```
Convert
 -treePath: 'Servers.ORACLE.Schemas.%'
/
Convert
 -treePath: 'ETL.INFA_SOURCE.Files'
/
```

10. (선택 사항) 변환 프로젝트와 평가 보고서를 저장합니다. 이 보고서에는 변환 작업 항목과 각 항목을 해결하는 방법에 대한 권장 사항이 포함되어 있습니다.

    다음 코드 예제는 프로젝트를 저장하고 평가 보고서 사본을 `C:\Temp` 폴더에 PDF 파일로 저장합니다.

```
SaveProject
/
SaveReportPDF
 -treePath: 'ETL.INFA_SOURCE.Files'
 -file:'C:\Temp\Informatica.pdf'
/
```

11. 변환된 Informatica XML 파일을 저장합니다.

    다음 코드 예제는 변환된 XML 파일을 `C:\Temp` 폴더에 저장합니다. 이전 단계에서 `AddTarget` 명령을 사용하여 이 폴더를 지정했습니다.

```
SaveTargetInformaticaXML
-treePath: 'ETL.INFA_TARGET.Files'
/
```

12. 스크립트를 `.scts` 파일로 저장하고 AWS SCT CLI에서 `RunSCTBatch` 명령을 사용하여 실행합니다. 자세한 내용은 AWS SCT CLI 스크립트 모드 섹션을 참조하세요.

    다음 예제는 `C:\Temp` 폴더에서 `Informatica.scts` 스크립트를 실행합니다. Windows에서 이 예제를 사용할 수 있습니다.

    ```
    RunSCTBatch.cmd --pathtoscts "C:\Temp\Informatica.scts"
    ```

    소스 Informatica ETL 스크립트를 편집하는 경우에는 AWS SCT CLI 스크립트를 다시 실행합니다.

AWS SCT를 사용하여 SSIS를 AWS Glue로 변환

다음에서는 AWS SCT를 사용하여 Microsoft SQL Server Integration Services(SSIS) 패키지를 AWS Glue로 변환하는 방법을 알아볼 수 있습니다.

Microsoft SSIS 패키지를 AWS Glue로 변환하려면 AWS SCT 버전 1.0.642 이상을 사용해야 합니다. 또한 로컬 폴더에 ETL 패키지가 포함된 SSIS 프로젝트(`.dtsx`, `.conmgr` 및 `.params` 파일)가 있어야 합니다.

SSIS 서버를 설치할 필요는 없습니다. 변환 프로세스는 로컬 SSIS 파일을 사용하여 수행됩니다.

AWS SCT를 사용하여 SSIS 패키지를 AWS Glue로 변환하려면

1. AWS SCT에서 새 프로젝트를 생성하거나 기존 프로젝트를 엽니다. 자세한 내용은 the section called “프로젝트 생성” 섹션을 참조하세요.
2. 메뉴에서 소스 추가를 선택하여 프로젝트에 새 소스 SSIS 패키지를 추가합니다.
3. SQL Server Integration Services를 선택하고 다음을 입력합니다.

    - 연결 이름 - 연결에 대한 이름을 입력합니다. AWS SCT는 이 이름을 메타데이터 트리에 표시합니다.
    - SSIS packages folder - 패키지가 있는 SSIS 프로젝트 폴더의 경로를 선택합니다.

    AWS SCT가 로컬 폴더에서 프로젝트 파일(확장자가 `.dtsx`, `.conmgr` 또는 `.params`인 파일)을 읽고 구문 분석합니다. 그런 다음 AWS SCT 범주 트리로 구성합니다.
4. 메뉴에서 대상 추가를 선택하여 소스 SSIS 패키지를 변환할 새 대상 플랫폼을 추가합니다.
5. AWS Glue를 선택하고 다음 입력합니다.

- 연결 이름 - 연결에 대한 이름을 입력합니다. AWS SCT는 이 이름을 메타데이터 트리에 표시합니다.
- Copy from AWS profile - 사용할 프로필을 선택합니다.
- AWS 액세스 키 – AWS 액세스 키를 입력합니다.
- AWS 비밀 키 - AWS 비밀 키를 입력합니다.
- 리전 – 목록에서 사용하려는 AWS 리전을 선택합니다.
- Amazon S3 bucket folder - 사용하려는 Amazon S3 버킷의 폴더 경로를 입력합니다.

가상 AWS Glue 대상을 사용할 수 있습니다. 이 경우 연결 보안 인증 정보를 지정하지 않아도 됩니다. 자세한 내용은 the section called “가상 대상” 섹션을 참조하세요.

6. 소스 SSIS 패키지와 AWS Glue 대상을 포함하는 새 매핑 규칙을 생성합니다. 자세한 내용은 the section called “새 규칙” 섹션을 참조하세요.
7. 보기 메뉴에서 Main view를 선택합니다.
8. SSIS 트리 보기에서 Connection managers의 컨텍스트(마우스 오른쪽 버튼 클릭) 메뉴를 연 다음 연결 구성을 선택합니다.
9. 프로젝트 연결 관리자를 구성합니다.

   SSIS 연결 관리자의 연결 매핑을 구성하려면 해당 SSIS 연결 관리자의 AWS Glue 연결을 지정합니다. AWS Glue 연결이 이미 생성되어 있는지 확인합니다.

   a. 연결에서 Project connections을 선택합니다.
   b. Glue catalog connection에서 적절한 AWS Glue 연결을 선택합니다.
10. 패키지 연결 관리자를 구성합니다.

    a. 연결에서 패키지를 선택합니다.
    b. Glue catalog connection에서 적절한 AWS Glue 연결을 선택합니다.
    c. 패키지에 사용할 수 있는 모든 연결에 대해 이 작업을 반복합니다.
11. Apply(적용)를 선택합니다.
12. 패키지를 변환합니다. 소스 트리 보기에서 패키지를 찾습니다. 패키지의 컨텍스트(마우스 오른쪽 버튼 클릭) 메뉴를 연 다음 Convert package를 선택합니다.
13. 변환된 스크립트를 Amazon S3에 저장합니다. 대상 트리 보기에서 Package scripts를 찾습니다. 변환된 스크립트의 컨텍스트(마우스 오른쪽 버튼 클릭) 메뉴를 연 다음 Save to S3을 선택합니다.

14. AWS Glue 작업을 구성합니다. 대상 트리 보기에서 Package scripts를 찾습니다. 변환된 스크립트의 컨텍스트(마우스 오른쪽 버튼 클릭) 메뉴를 연 다음 Configure AWS Glue job을 선택합니다.
15. 3개의 구성 섹션을 완료합니다.

    a. Design data flow 섹션을 완료합니다.

    - Execution strategy - 작업에서 ETL 스크립트를 실행하는 방법을 선택합니다. 마법사에 지정된 순서대로 스크립트를 실행하려면 SEQUENTIAL을 선택합니다. 마법사에 지정된 순서를 무시하고 스크립트를 병렬로 실행하려면 PARALLEL을 선택합니다.
    - 스크립트 - 변환된 스크립트의 이름을 선택합니다.
    - 다음(Next)을 선택합니다.

    b. Job properties 섹션을 완료합니다.

    - 이름 – AWS Glue 작업의 이름을 입력합니다.
    - IAM 역할 – 작업을 실행하고 데이터 스토어에 액세스하는 데 사용되는 리소스에 권한을 부여하는 데 사용할 IAM 역할을 선택합니다.
    - 스크립트 파일 이름 - 변환된 스크립트의 이름을 입력합니다.
    - Script file S3 path - 변환된 스크립트의 Amazon S3 경로를 입력합니다.
    - Encrypt script using SSE-S3 – Amazon S3 관리형 암호화 키(SSE-S3)를 통한 서버 측 암호화를 사용하여 데이터를 보호하려면 이 옵션을 선택합니다.
    - 임시 디렉터리 - 중간 결과를 위한 Amazon S3 임시 디렉터리 경로를 입력합니다. AWS Glue 및 AWS Glue 기본 제공 변환에서 이 디렉터리를 사용하여 Amazon Redshift를 읽거나 씁니다.
    - AWS SCT는 Python 라이브러리의 경로를 자동으로 생성합니다. Generated python library path에서 이 경로를 검토할 수 있습니다. 자동으로 생성된 이 경로는 편집할 수 없습니다. 추가 Python 라이브러리를 사용하려면 User python library path에 경로를 입력합니다.
    - User python library path - 추가 사용자 Python 라이브러리의 경로를 입력합니다. 쉼표로 Amazon S3 경로를 구분합니다.
    - 종속된 jar 경로 - 종속된 jar 파일의 경로를 입력합니다. 쉼표로 Amazon S3 경로를 구분합니다.
    - 참조된 파일 경로 - 스크립트에 필요한 추가 파일(예: 구성 파일)의 경로를 입력합니다. 쉼표로 Amazon S3 경로를 구분합니다.
    - 최대 용량 – 이 작업이 실행될 때 할당할 수 있는 AWS Glue 데이터 처리 장치(DPU)의 최대 수입니다. 2~100의 정수를 입력할 수 있습니다. 기본값은 2입니다.

- 최대 동시성 - 이 작업에 허용되는 최대 동시 실행 수를 입력합니다. 기본값은 1입니다. 이 임계값에 도달하면 AWS Glue에서 오류가 반환됩니다.
- 작업 제한 시간(분) - 런어웨이 작업을 방지하기 위해 ETL 작업에 대한 제한 시간 값을 입력합니다. 배치 작업의 경우 기본값은 2,880분(48시간)입니다. 작업이 이 제한을 초과하면 작업 실행 상태가 `TIMEOUT`으로 변경됩니다.
- Delay 알림 임계값(분) - AWS SCT에서 지연 알림을 보내기 전의 임계값(분)을 입력합니다.
- 재시도 횟수 - 작업이 실패할 경우 AWS Glue에서 자동으로 작업을 재시작할 횟수(0~10)를 입력합니다. 제한 시간에 도달한 작업은 재시작되지 않습니다. 기본값은 0입니다.
- 다음(Next)을 선택합니다.

c. 필요한 연결을 구성합니다.

i. 모든 연결에서 필요한 AWS Glue 연결을 선택하고 Selected connections 목록에 추가합니다.

ii. [마침]을 클릭합니다.

16. 구성된 AWS Glue 작업을 생성합니다. 대상 트리 보기에서 ETL Jobs를 찾아 확장합니다. 구성한 ETL 작업의 컨텍스트(마우스 오른쪽 버튼 클릭) 메뉴를 연 다음 Create AWS Glue Job을 선택합니다.
17. AWS Glue 작업을 실행합니다.

a. https://console.aws.amazon.com/glue/에서 AWS Glue 콘솔을 엽니다.

b. 탐색 창에서 작업을 선택합니다.

c. 작업 추가를 선택한 다음 실행하려는 작업을 선택합니다.

d. 작업 탭에서 작업 실행을 선택합니다.

AWS SCT가 AWS Glue로 변환할 수 있는 SSIS 구성 요소

AWS SCT를 사용하여 데이터 흐름과 제어 흐름 구성 요소뿐만 아니라 컨테이너, 파라미터 및 변수도 변환할 수 있습니다.

지원되는 데이터 흐름 구성 요소는 다음과 같습니다.

- ADO NET Destination
- ADO NET Source
- Aggregate
- Cache Transformation

- Character Map Transformation
- Conditional Split Transformation
- Copy Column Transformation
- Data Conversion Transformation
- Derived Column Transformation
- Excel Destination
- Excel Source
- Export Column Transformation
- Flat File Destination
- Flat File Source
- Fuzzy Lookup Transformation
- Import Column Transformation
- Lookup Transformation
- Merge Join Transformation
- Merge Transformation
- Multicast Transformation
- ODBC Destination
- ODBC Source
- OLE DB Command Transformation
- OLE DB Destination
- OLE DB Source
- Percentage Sampling Transformation
- Pivot Transformation
- Raw File Destination
- Raw File Source
- RecordSet Destination
- Row Count Transformation
- Row Sampling Transformation

- Sort Transformation
- SQL Server Destination
- Union All Transformation
- Unpivot Transformation
- XML Source

지원되는 제어 흐름 구성 요소는 다음과 같습니다.

- 대량 삽입 작업
- 패키지 실행 작업
- SQL 실행 작업
- T-SQL 문 실행 작업
- 표현식 작업
- 파일 시스템 작업
- 운영자에게 알림 작업
- 메일 전송 작업

지원되는 SSIS 컨테이너는 다음과 같습니다.

- For Loop 컨테이너
- Foreach Loop 컨테이너
- Sequence 컨테이너

AWS SCT를 사용하여 SSIS를 AWS Glue Studio로 변환

AWS SCT를 사용하여 Microsoft SQL Server Integration Services(SSIS) 패키지를 AWS Glue Studio로 변환할 수 있습니다.

SSIS 패키지에는 연결 관리자, 작업, 제어 흐름, 데이터 흐름, 파라미터, 이벤트 핸들러, 변수 등과 같이 특정 추출, 전환, 적재(ETL) 작업을 실행하는 데 필요한 구성 요소가 포함되어 있습니다. AWS SCT는 SSIS 패키지를 AWS Glue Studio와 호환 가능한 형식으로 변환합니다. 소스 데이터베이스를 AWS 클라우드로 마이그레이션한 후 이러한 변환된 AWS Glue Studio 작업을 실행하여 ETL 작업을 수행할 수 있습니다.

Microsoft SSIS 패키지를 AWS Glue Studio로 변환하려면 AWS SCT 버전 1.0.661 이상을 사용해야 합니다.

주제

- 필수 조건
- AWS SCT 프로젝트에 SSIS 패키지 추가
- AWS SCT를 사용하여 SSIS 패키지를 AWS Glue Studio로 변환
- 변환된 코드를 사용하여 AWS Glue Studio 작업 생성
- AWS SCT를 사용하여 SSIS 패키지에 대한 평가 보고서 생성
- AWS SCT가 AWS Glue Studio로 변환할 수 있는 SSIS 구성 요소

필수 조건

이 섹션에서는 SSIS 패키지를 AWS Glue로 변환하기 위한 필수 조건에 대해 알아봅니다. 이러한 작업에는 계정에 필요한 AWS 리소스를 생성하는 작업이 포함됩니다.

AWS Identity and Access Management(IAM)를 사용하면 AWS Glue Studio에서 사용하는 리소스에 액세스하는 데 필요한 정책과 역할을 정의할 수 있습니다. 자세한 내용은 AWS Glue Studio 사용자를 위한 IAM 권한 섹션을 참조하세요.

AWS SCT가 소스 스크립트를 AWS Glue Studio로 변환한 후 변환된 스크립트를 Amazon S3 버킷에 업로드합니다. 이 Amazon S3 버킷을 생성하고 AWS 서비스 프로필 설정에서 이를 선택해야 합니다. Amazon S3 버킷 생성에 대한 자세한 내용은 Amazon Simple Storage Service 사용 설명서에서 첫 번째 S3 버킷 생성을 참조하세요.

AWS Glue Studio가 데이터 스토어에 연결할 수 있도록 하려면 사용자 지정 커넥터와 연결을 생성합니다. 또한 AWS Secrets Manager에 데이터베이스 보안 인증 정보를 저장합니다.

사용자 지정 커넥터를 생성하려면

1. 데이터 스토어의 JDBC 드라이버를 다운로드합니다. AWS SCT에서 사용하는 JDBC 드라이버에 대한 자세한 내용은 필수 데이터베이스 드라이버 다운로드 섹션을 참조하세요.
2. 이 드라이버 파일을 Amazon S3 버킷에 업로드합니다. 자세한 내용은 Amazon Simple Storage Service 사용 설명서에서 버킷에 객체 업로드를 참조하세요.
3. AWS Management Console에 로그인하고 https://console.aws.amazon.com/gluestudio/에서 AWS Glue Studio 콘솔을 엽니다.
4. 커넥터를 선택한 다음 Create custom connector를 선택합니다.

5. Connector S3 URL에서 Browse S3을 선택하고 Amazon S3 버킷에 업로드한 JDBC 드라이버 파일을 선택합니다.
6. 커넥터를 나타내는 이름을 입력합니다. 예를 들면 **SQLServer**을 입력합니다.
7. 커넥터 유형에서 JDBC를 선택합니다.
8. 클래스 이름에 JDBC 드라이버의 기본 클래스 이름을 입력합니다. SQL Server의 경우 **com.microsoft.sqlserver.jdbc.SQLServerDriver**를 입력합니다.
9. JDBC URL base에 JDBC 기본 URL을 입력합니다. JDBC 기본 URL의 구문은 소스 데이터베이스 엔진에 따라 달라집니다. SQL Server의 경우 **jdbc:sqlserver://$*<host>*:$*<port>*;databaseName=$*<dbname>*;user=$*<username>*;password=$*<password>*** 형식을 사용합니다.

   <host>, *<port>*, *<dbname>*, *<username>* 및 *<password>*를 적절한 값으로 바꿔야 합니다.
10. URL parameter delimiter에 세미콜론(;)을 입력합니다.
11. [커넥터 생성(Create connector)]을 선택합니다.

AWS Secrets Manager에 데이터베이스 보안 인증 정보를 저장하려면

1. AWS Management Console에 로그인하고 https://console.aws.amazon.com/secretsmanager/에서 AWS Secrets Manager 콘솔을 엽니다.
2. Store a new secret(새 보안 암호 저장)을 선택합니다.
3. 보안 암호 유형 선택(Choose secret type) 페이지에서 다음을 수행합니다.

   a. 보안 암호 유형에서 다른 유형의 보안 암호를 선택합니다.

   b. 키/값 페어에 **host**, **port**, **dbname**, **username** 및 **password** 키를 입력합니다.

      그런 다음 이들 키의 값을 입력합니다.
4. 보안 암호 구성 페이지에서 설명이 포함된 보안 암호 이름을 입력합니다. 예를 들면 **SQL_Server_secret**을 입력합니다.
5. 다음(Next)을 선택합니다. 그 다음, 교체 구성 페이지에서 다음을 다시 선택합니다.
6. Review(검토) 페이지에서 보안 암호 세부 정보를 검토한 후 Store(저장)를 선택합니다.

커넥터에 대한 연결을 생성하려면

1. AWS Management Console에 로그인하고 https://console.aws.amazon.com/gluestudio/에서 AWS Glue Studio 콘솔을 엽니다.

2. 연결을 생성할 커넥터를 선택한 다음 연결 생성을 선택합니다.
3. 연결 생성 페이지에서 연결을 설명하는 이름을 입력합니다. 예를 들면 **SQL-Server-connection**을 입력합니다.
4. AWS Secret에서, AWS Secrets Manager에서 생성한 보안 암호를 선택합니다.
5. 네트워크 옵션을 구성한 다음 연결 생성을 선택합니다.

이제 사용자 지정 커넥터를 사용하여 AWS Glue Studio 작업을 생성할 수 있습니다. 자세한 내용은 AWS Glue Studio 작업 생성 섹션을 참조하세요.

AWS SCT 프로젝트에 SSIS 패키지 추가

단일 AWS SCT 프로젝트에 여러 SSIS 패키지를 추가할 수 있습니다.

AWS SCT 프로젝트에 SSIS 패키지를 추가하려면

1. AWS SCT에서 새 프로젝트를 생성하거나 기존 프로젝트를 엽니다. 자세한 내용은 the section called “프로젝트 생성” 섹션을 참조하세요.
2. 메뉴에서 소스 추가를 선택한 다음 SQL Server Integration Services를 선택합니다.
3. 연결 이름에 SSIS 패키지의 이름을 입력합니다. AWS SCT는 왼쪽 패널의 트리에 이 이름을 표시합니다.
4. SSIS packages folder에 소스 SSIS 패키지가 있는 폴더의 경로를 입력합니다.
5. 메뉴에서 대상 추가를 선택하고 AWS Glue Studio를 선택합니다.

   AWS Glue Studio에 연결하기 위해 AWS SCT는 AWS 프로필을 사용합니다. 자세한 내용은 AWS SCT에 AWS 서비스 프로필 저장 섹션을 참조하세요.
6. 소스 SSIS 패키지와 AWS Glue Studio 대상을 포함하는 매핑 규칙을 생성합니다. 자세한 내용은 AWS SCT에서 매핑 규칙 생성 섹션을 참조하세요.
7. AWS Glue Studio 콘솔에서 AWS Glue Studio 연결을 생성합니다. 자세한 내용은 커넥터에 대한 연결 생성을 참조하세요.
8. 왼쪽 트리에서 Connection managers를 선택하고 컨텍스트(마우스 오른쪽 버튼 클릭) 메뉴를 연 다음 연결 구성을 선택합니다.

   AWS SCT에서 연결 구성 창이 표시됩니다.
9. 각 소스 SSIS 연결에 대해 AWS Glue Studio 연결을 선택합니다.

AWS SCT를 사용하여 SSIS 패키지를 AWS Glue Studio로 변환

다음에서는 AWS SCT를 사용하여 SSIS 패키지를 AWS Glue Studio로 변환하는 방법을 알아봅니다.

SSIS 패키지를 AWS Glue Studio로 변환하려면

1. AWS SCT 프로젝트에 SSIS 패키지를 추가합니다. 자세한 내용은 AWS SCT 프로젝트에 SSIS 패키지 추가 섹션을 참조하세요.
2. 왼쪽 패널에서 ETL 및 SSIS 노드를 확장합니다.
3. 패키지를 선택하고 컨텍스트(마우스 오른쪽 버튼 클릭) 메뉴를 연 다음 Convert package를 선택합니다.

   AWS SCT는 선택한 SSIS 패키지를 JSON 파일로 변환합니다. 이러한 JSON 객체는 DAG(방향성 비순환 그래프)로 노드를 표시합니다 오른쪽 트리의 Package DAGs 노드에서 변환된 파일을 찾습니다.
4. Package DAGs를 선택하고 컨텍스트(마우스 오른쪽 버튼 클릭) 메뉴를 연 다음 Save to Amazon S3을 선택합니다.

   이제 이러한 스크립트를 사용하여 AWS Glue Studio에서 작업을 생성할 수 있습니다.

변환된 코드를 사용하여 AWS Glue Studio 작업 생성

소스 SSIS 패키지를 변환한 후 변환된 JSON 파일을 사용하여 AWS Glue Studio 작업을 생성할 수 있습니다.

AWS Glue Studio 작업을 생성하려면

1. 오른쪽 트리에서 Package DAGs를 선택하고 컨텍스트(마우스 오른쪽 버튼 클릭) 메뉴를 연 후 Configure AWS Glue Studio job을 선택합니다.
2. (선택 사항) AWS Glue Studio에서 SSIS 함수를 에뮬레이션하는 확장 팩을 적용합니다.
3. Configure AWS Glue Studio job 창이 열립니다.

   Basic job properties 섹션을 완료합니다.

   - 이름 – AWS Glue Studio 작업의 이름을 입력합니다.
   - 스크립트 파일 이름 - 작업 스크립트의 이름을 입력합니다.
   - 작업 파라미터 – 파라미터를 추가하고 값을 입력합니다.

다음(Next)을 선택합니다.

4. Advanced job properties 섹션을 완료합니다.

   - IAM 역할 - AWS Glue Studio에 권한을 부여하고 데이터 스토어에 액세스하는 데 사용되는 IAM 역할을 선택합니다.
   - Script file S3 path - 변환된 스크립트의 Amazon S3 경로를 입력합니다.
   - 임시 디렉터리 - 중간 결과를 위한 Amazon S3 임시 디렉터리 경로를 입력합니다. AWS Glue Studio는 이 디렉터리를 사용하여 Amazon Redshift를 읽거나 씁니다.
   - AWS SCT는 Python 라이브러리의 경로를 자동으로 생성합니다. Generated python library path에서 이 경로를 검토할 수 있습니다. 자동으로 생성된 이 경로는 편집할 수 없습니다. 추가 Python 라이브러리를 사용하려면 User python library path에 경로를 입력합니다.
   - User python library path - 추가 사용자 Python 라이브러리의 경로를 입력합니다. 쉼표로 Amazon S3 경로를 구분합니다.
   - 종속된 jar 경로 - 종속된 `*.jar` 파일의 경로를 입력합니다. 쉼표로 Amazon S3 경로를 구분합니다.
   - 참조된 파일 경로 - 스크립트에 필요한 추가 파일(예: 구성 파일)의 경로를 입력합니다. 쉼표로 Amazon S3 경로를 구분합니다.
   - 작업자 유형 - `G.1X` 또는 `G.2X`를 선택합니다.

     `G.1X`를 선택하면 각 작업자가 1 DPU(4 vCPU, 16GB 메모리, 64GB 디스크)에 매핑합니다.

     `G.2X`를 선택하면 각 작업자가 2 DPU(8 vCPU, 32GB 메모리, 128GB 디스크)에 매핑합니다.
   - Requested number of workers - 작업이 실행될 때 할당되는 작업자 수를 입력합니다.
   - 최대 동시성 - 이 작업에 허용되는 최대 동시 실행 수를 입력합니다. 기본값은 1입니다. 이 임계값에 도달하면 AWS Glue에서 오류가 반환됩니다.
   - 작업 제한 시간(분) - 런어웨이 작업을 방지하기 위해 ETL 작업에 대한 제한 시간 값을 입력합니다. 배치 작업의 경우 기본값은 2,880분(48시간)입니다. 작업이 이 제한을 초과하면 작업 실행 상태가 `TIMEOUT`으로 변경됩니다.
   - Delay 알림 임계값(분) - AWS SCT에서 지연 알림을 보내기 전의 임계값(분)을 입력합니다.
   - 재시도 횟수 - 작업이 실패할 경우 AWS Glue에서 자동으로 작업을 재시작할 횟수(0~10)를 입력합니다. 제한 시간에 도달한 작업은 재시작되지 않습니다. 기본값은 0입니다.

   [마침]을 클릭합니다.

AWS Glue Studio가 선택된 AWS SCT 작업을 구성합니다.

5. 오른쪽 트리의 ETL jobs에서 구성된 작업을 찾을 수 있습니다. 구성된 작업을 선택하고 컨텍스트(마우스 오른쪽 버튼 클릭) 메뉴를 연 후 Create AWS Glue Studio job을 선택합니다.
6. Apply status를 선택하고 작업의 상태 값이 성공인지 확인합니다.
7. AWS Glue Studio 콘솔을 열고 새로 고침을 선택한 다음 작업을 선택합니다. 그런 다음 실행(Run)을 선택합니다.

AWS SCT를 사용하여 SSIS 패키지에 대한 평가 보고서 생성

ETL 마이그레이션 평가 보고서는 SSIS 패키지를 AWS Glue Studio와 호환 가능한 형식으로 변환하는 방법에 대한 정보를 제공합니다. 평가 보고서에는 SSIS 패키지의 구성 요소에 대한 작업 항목이 포함되어 있습니다. 이러한 작업 항목은 AWS SCT가 자동으로 변환할 수 없는 구성 요소를 보여줍니다.

ETL 마이그레이션 평가 보고서를 생성하려면

1. 왼쪽 패널의 ETL에서 SSIS 노드를 확장합니다.
2. 패키지를 선택하고 컨텍스트(마우스 오른쪽 버튼 클릭) 메뉴를 연 다음 보고서 생성을 선택합니다.
3. 요약 탭을 검토합니다. 여기에서 AWS SCT는 ETL 마이그레이션 평가 보고서의 핵심 요약 정보를 표시합니다. 여기에는 SSIS 패키지의 모든 구성 요소에 대한 변환 결과가 포함됩니다.
4. (선택 사항) ETL 마이그레이션 평가 보고서의 로컬 사본을 PDF 파일 또는 쉼표로 구분된 값(CSV) 파일로 저장합니다.

   - ETL 마이그레이션 평가 보고서를 PDF 파일로 저장하려면 오른쪽 상단에서 Save to PDF를 선택합니다.

     PDF 파일에는 스크립트 변환에 대한 요약 정보, 작업 항목 및 권장 사항이 포함됩니다.
   - ETL 마이그레이션 평가 보고서를 CSV 파일로 저장하려면 오른쪽 상단에서 Save to CSV를 선택합니다.

     AWS SCT는 세 개의 CSV 파일을 생성합니다. 이러한 파일에는 스크립트 변환에 필요한 작업 항목, 권장 작업 및 예상 수작업의 복잡성 등이 포함됩니다.
5. 작업 항목 탭을 선택합니다. 이 탭에는 AWS Glue Studio로의 수동 변환이 필요한 항목 목록이 포함되어 있습니다. 목록에서 작업 항목을 선택하면 AWS SCT가 소스 SSIS 패키지에서 해당 작업 항목이 적용되는 항목을 강조 표시합니다.

AWS SCT가 AWS Glue Studio로 변환할 수 있는 SSIS 구성 요소

AWS SCT를 사용하여 SSIS 데이터 흐름 구성 요소 및 파라미터를 AWS Glue Studio로 변환할 수 있습니다.

지원되는 데이터 흐름 구성 요소는 다음과 같습니다.

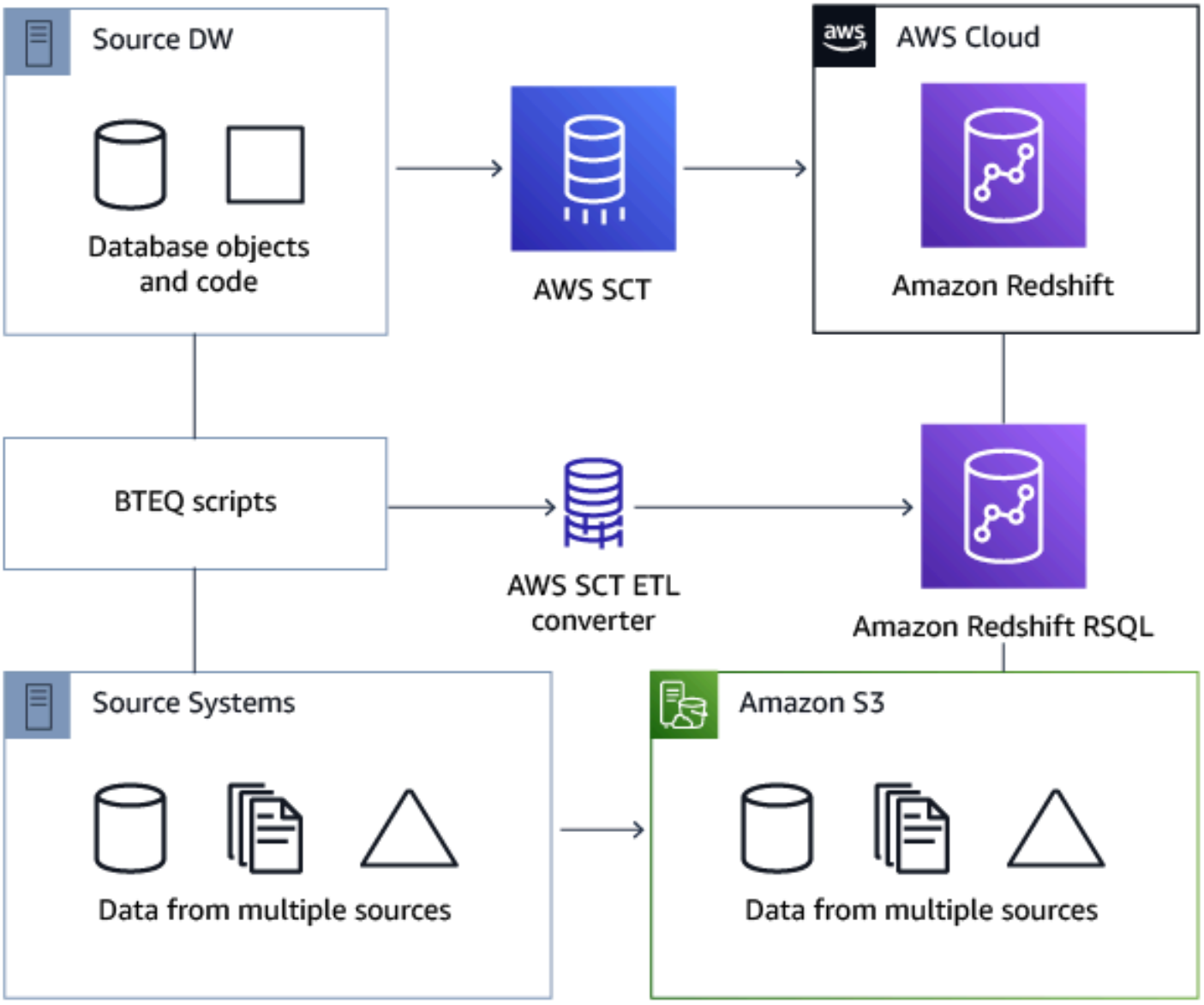
- ADO NET Destination
- ADO NET Source
- Aggregate
- 문자 맵
- 조건부 분할
- 열 복사
- 데이터 변환
- 파생된 영
- 조회
- 병합
- 병합 조인
- 멀티캐스트
- ODBCDestination
- ODBCSource
- OLEDBDestination
- OLEDBSource
- 행 수
- 정렬
- SQL Server Destination
- Union All

AWS SCT가 더 많은 SSIS 구성 요소를 AWS Glue로 변환할 수 있습니다. 자세한 내용은 AWS SCT가 AWS Glue로 변환할 수 있는 SSIS 구성 요소 섹션을 참조하세요.

AWS SCT를 사용하여 Teradata BTEQ 스크립트를 Amazon Redshift RSQL로 변환

AWS Schema Conversion Tool(AWS SCT)을 사용하여 Teradata Basic Teradata Query(BTEQ) 스크립트를 Amazon Redshift RSQL로 변환할 수 있습니다.

다음 아키텍처 다이어그램은 추출, 전환, 적재(ETL) 스크립트의 Amazon Redshift RSQL로의 변환을 포함하는 데이터베이스 마이그레이션 프로젝트를 보여줍니다.



주제

- AWS SCT 프로젝트에 BTEQ 스크립트 추가

- AWS SCT를 사용하여 BTEQ 스크립트에서 대체 변수 구성
- AWS SCT를 사용하여 Teradata BTEQ 스크립트를 Amazon Redshift RSQL로 변환
- AWS SCT를 사용하여 BTEQ 스크립트 관리
- AWS SCT를 사용하여 BTEQ 스크립트 변환 평가 보고서 생성
- AWS SCT를 사용하여 변환된 BTEQ 스크립트 편집 및 저장

AWS SCT 프로젝트에 BTEQ 스크립트 추가

단일 AWS SCT 프로젝트에 여러 스크립트를 추가할 수 있습니다.

AWS SCT 프로젝트에 BTEQ 스크립트를 추가하려면

1. AWS SCT에서 새 프로젝트를 생성하거나 기존 프로젝트를 엽니다. 자세한 내용은 the section called “프로젝트 생성” 섹션을 참조하세요.
2. 메뉴에서 소스 추가를 선택한 다음 Teradata를 선택하여 프로젝트에 소스 데이터베이스를 추가합니다. 자세한 내용은 Teradata를 소스로 사용 섹션을 참조하세요.
3. 메뉴에서 대상 추가를 선택하여 대상 Amazon Redshift 데이터베이스를 AWS SCT 프로젝트에 추가합니다.

   가상 Amazon Redshift 대상 데이터베이스 플랫폼을 사용할 수 있습니다. 자세한 내용은 가상 대상 사용 섹션을 참조하세요.
4. 소스 Teradata 데이터베이스와 Amazon Redshift 대상을 포함하는 새 매핑 규칙을 생성합니다. 자세한 내용은 새 매핑 규칙 추가 섹션을 참조하세요.
5. 보기 메뉴에서 Main view를 선택합니다.
6. 왼쪽 패널에서 스크립트 노드를 확장합니다.
7. BTEQ scripts를 선택하고 컨텍스트(마우스 오른쪽 버튼 클릭) 메뉴를 연 다음 Load scripts를 선택합니다.
8. Teradata BTEQ 스크립트의 소스 코드 위치를 입력하고 폴더 선택을 선택합니다.

   AWS SCT에서 Load scripts 창을 표시합니다.
9. 다음 중 하나를 수행하세요.

   a. Teradata BTEQ 스크립트에 대체 변수가 포함되어 있지 않은 경우 No substitution variables를 선택한 다음 확인을 선택하여 AWS SCT 프로젝트에 스크립트를 추가합니다.

b. Teradata BTEQ 스크립트에 대체 변수가 포함된 경우 해당 대체 변수를 구성합니다. 자세한 내용은 BTEQ 스크립트에서 대체 변수 구성 섹션을 참조하세요.

AWS SCT를 사용하여 BTEQ 스크립트에서 대체 변수 구성

Teradata BTEQ 스크립트에는 대체 변수가 포함될 수 있습니다. 예를 들어, 대체 변수가 있는 BTEQ 스크립트를 사용하여 여러 데이터베이스 환경에서 동일한 명령 세트를 실행할 수 있습니다. AWS SCT를 사용하여 BTEQ 스크립트에서 대체 변수를 구성할 수 있습니다.

대체 변수가 있는 BTEQ 스크립트를 실행하기 전에 모든 변수에 값을 할당해야 합니다. 이렇게 하려면 Bash 스크립트, UC4(Automic) 등과 같은 다른 도구 또는 애플리케이션을 사용할 수 있습니다. AWS SCT는 값을 할당한 후에만 대체 변수를 해석하고 변환할 수 있습니다.

BTEQ 스크립트에서 대체 변수를 구성하려면

1. AWS SCT 프로젝트에 BTEQ 스크립트를 추가합니다. 자세한 내용은 AWS SCT 프로젝트에 BTEQ 스크립트 추가 섹션을 참조하세요.

   스크립트를 추가할 때 Substitution variables are used를 선택합니다.
2. Define variable format에 스크립트의 모든 대체 변수와 일치하는 정규식을 입력합니다.

   예를 들어, 대체 변수 이름이 ${로 시작하고 }로 끝나는 경우 `\$\{\w+\}` 정규식을 사용합니다. 달러 기호 또는 퍼센트 기호로 시작하는 대체 변수를 일치시키려면 `\$\w+|\%\w+` 정규식을 사용합니다.

   AWS SCT의 정규식은 Java 정규식 구문을 따릅니다. 자세한 내용은 Java 설명서에서 java.util.regex 클래스 패턴를 참조하세요.
3. 확인을 선택하여 AWS SCT 프로젝트에 스크립트를 로드한 다음 확인을 선택하여 Load scripts 창을 닫습니다.
4. 변수를 선택하여 검색된 모든 대체 변수와 해당 값을 확인합니다.
5. 값에 대체 변수의 값을 입력합니다.

AWS SCT를 사용하여 Teradata BTEQ 스크립트를 Amazon Redshift RSQL로 변환

다음에서는 AWS SCT를 사용하여 BTEQ ETL 스크립트를 Amazon Redshift RSQL로 변환하는 방법을 알아봅니다.

Teradata BTEQ 스크립트를 Amazon Redshift RSQL로 변환하려면

1. AWS SCT 프로젝트에 BTEQ 스크립트를 추가합니다. 자세한 내용은 AWS SCT 프로젝트에 BTEQ 스크립트 추가 섹션을 참조하세요.
2. 대체 변수를 구성합니다. 자세한 내용은 BTEQ 스크립트에서 대체 변수 구성 섹션을 참조하세요.
3. 왼쪽 패널에서 스크립트 노드를 확장합니다.
4. 다음 중 하나를 수행하세요.

   - 단일 BTEQ 스크립트를 변환하려면 BTEQ scripts 노드를 확장하고 변환할 스크립트를 선택한 다음, 컨텍스트(마우스 오른쪽 버튼 클릭) 메뉴에서 Convert to RSQL을 선택합니다.
   - 여러 스크립트를 변환하려면 변환할 스크립트를 모두 선택해야 합니다. 그런 다음, BTEQ scripts를 선택하고 컨텍스트(마우스 오른쪽 버튼 클릭) 메뉴를 연 다음, Convert script에서 Convert to RSQL을 선택합니다.

   AWS SCT가 선택된 모든 Teradata BTEQ 스크립트를 Amazon Redshift RSQL과 호환되는 형식으로 변환합니다. 대상 데이터베이스 패널의 스크립트 노드에서 변환된 스크립트를 찾습니다.
5. 변환된 Amazon Redshift RSQL 스크립트를 편집하거나 저장합니다. 자세한 내용은 변환된 BTEQ 스크립트 편집 및 저장 섹션을 참조하세요.

AWS SCT를 사용하여 BTEQ 스크립트 관리

AWS SCT 프로젝트에 여러 BTEQ 스크립트를 추가하거나 BTEQ 스크립트를 제거할 수 있습니다.

AWS SCT 프로젝트에 BTEQ 스크립트를 더 추가하려면

1. 왼쪽 패널에서 스크립트 노드를 확장합니다.
2. BTEQ scripts 노드를 선택하고 컨텍스트(마우스 오른쪽 버튼 클릭) 메뉴를 엽니다.
3. Load scripts를 선택합니다.
4. 새 BTEQ 스크립트를 추가하고 대체 변수를 구성하는 데 필요한 정보를 입력합니다. 자세한 정보는 AWS SCT 프로젝트에 BTEQ 스크립트 추가 및 BTEQ 스크립트에서 대체 변수 구성 섹션을 참조하세요.

AWS SCT 프로젝트에서 BTEQ 스크립트를 제거하려면

1. 왼쪽 패널의 스크립트에서 BTEQ scripts 노드를 확장합니다.

2. 제거할 스크립트를 선택하고 컨텍스트(마우스 오른쪽 버튼 클릭) 메뉴를 엽니다.
3. 스크립트 삭제를 선택합니다.

AWS SCT를 사용하여 BTEQ 스크립트 변환 평가 보고서 생성

BTEQ 스크립트 변환 평가 보고서는 BTEQ 스크립트의 BTEQ 명령 및 SQL 문을 Amazon Redshift RSQL과 호환되는 형식으로 변환하는 방법에 대한 정보를 제공합니다. 평가 보고서에는 AWS SCT가 변환할 수 없는 BTEQ 명령 및 SQL 문에 대한 작업 항목이 포함되어 있습니다.

BTEQ 스크립트 변환 평가 보고서를 만들려면

1. 왼쪽 패널의 스크립트에서 BTEQ scripts 노드를 확장합니다.
2. 변환할 스크립트를 선택하고 컨텍스트(마우스 오른쪽 버튼 클릭) 메뉴를 엽니다.
3. 보고서 생성에서 Conversion to RSQL을 선택합니다.
4. 요약 탭을 검토합니다. 요약 탭에는 BTEQ 스크립트 평가 보고서의 요약 정보가 표시됩니다. 여기에는 BTEQ 스크립트의 모든 BTEQ 명령 및 SQL 문에 대한 변환 결과가 포함됩니다.
5. (선택 사항) BTEQ 스크립트 변환 평가 보고서의 로컬 사본을 PDF 파일 또는 쉼표로 구분된 값(CSV) 파일로 저장합니다.

   - BTEQ 스크립트 변환 평가 보고서를 PDF 파일로 저장하려면 오른쪽 상단에서 Save to PDF를 선택합니다.

     PDF 파일에는 스크립트 변환에 대한 요약 정보, 작업 항목 및 권장 사항이 포함됩니다.
   - BTEQ 스크립트 변환 평가 보고서를 CSV 파일로 저장하려면 오른쪽 상단에서 Save to CSV를 선택합니다.

     CSV 파일에는 스크립트 변환에 필요한 작업 항목, 권장 작업 및 예상 수작업의 복잡성 등이 포함됩니다.
6. 작업 항목 탭을 선택합니다. 이 탭에는 Amazon Redshift RSQL로의 수동 변환이 필요한 항목 목록이 포함되어 있습니다. 목록에서 작업 항목을 선택하면 AWS SCT가 소스 BTEQ 스크립트에서 해당 작업 항목이 적용되는 항목을 강조 표시합니다.

AWS SCT를 사용하여 변환된 BTEQ 스크립트 편집 및 저장

변환된 스크립트는 AWS SCT 프로젝트의 하단 패널에서 편집할 수 있습니다. AWS SCT는 편집된 스크립트를 프로젝트의 일부로 저장합니다.

변환된 스크립트를 저장하려면

1. 대상 데이터베이스 패널의 스크립트에서 RSQL scripts 노드를 확장합니다.
2. 변환된 스크립트를 선택하고 컨텍스트(마우스 오른쪽 버튼 클릭) 메뉴를 연 다음 스크립트 저장을 선택합니다.
3. 변환된 스크립트를 저장할 폴더의 경로를 입력하고 저장을 선택합니다.

   AWS SCT가 변환된 스크립트를 파일에 저장하고 이 파일을 엽니다.

AWS SCT를 사용하여 임베디드 Teradata BTEQ 명령이 있는 쉘 스크립트를 Amazon Redshift RSQL로 변환

AWS Schema Conversion Tool(AWS SCT)을 사용하여 임베디드 Teradata Basic Teradata Query(BTEQ) 명령이 있는 쉘 스크립트를 임베디드 Amazon Redshift RSQL 명령이 있는 쉘 스크립트로 변환할 수 있습니다.

AWS SCT는 쉘 스크립트에서 Teradata BTEQ 명령을 추출하여 Amazon Redshift와 호환되는 형식으로 변환합니다. Teradata 데이터베이스를 Amazon Redshift로 마이그레이션한 후에는 이러한 변환된 스크립트를 사용하여 새 Amazon Redshift 데이터베이스를 관리할 수 있습니다.

또한 AWS SCT를 사용하여 Teradata BTEQ ETL 스크립트가 있는 파일을 Amazon Redshift RSQL로 변환할 수 있습니다. 자세한 내용은 AWS SCT를 사용하여 Teradata BTEQ 스크립트를 Amazon Redshift RSQL로 변환 섹션을 참조하세요.

주제

- 임베디드 Teradata BTEQ 명령이 있는 쉘 스크립트를 AWS SCT 프로젝트에 추가
- AWS SCT를 사용하여 임베디드 Teradata BTEQ 명령이 있는 쉘 스크립트에서 대체 변수 구성
- AWS SCT를 사용하여 임베디드 Teradata BTEQ 명령이 있는 쉘 스크립트 변환
- AWS SCT를 사용하여 임베디드 Teradata BTEQ 명령이 있는 쉘 스크립트 관리
- AWS SCT를 사용하여 쉘 스크립트 변환을 위한 평가 보고서 생성
- AWS SCT를 사용하여 변환된 쉘 스크립트 편집 및 저장

임베디드 Teradata BTEQ 명령이 있는 쉘 스크립트를 AWS SCT 프로젝트에 추가

단일 AWS SCT 프로젝트에 여러 스크립트를 추가할 수 있습니다.

AWS SCT 프로젝트에 쉘 스크립트를 추가하려면

1. AWS SCT에서 새 프로젝트를 생성하거나 기존 프로젝트를 엽니다. 자세한 내용은 the section called “프로젝트 생성” 섹션을 참조하세요.
2. 메뉴에서 소스 추가를 선택한 다음 Teradata를 선택하여 프로젝트에 소스 데이터베이스를 추가합니다. 자세한 내용은 Teradata를 소스로 사용 섹션을 참조하세요.
3. 메뉴에서 대상 추가를 선택하고 대상 Amazon Redshift 데이터베이스를 AWS SCT 프로젝트에 추가합니다.

   가상 Amazon Redshift 대상 데이터베이스 플랫폼을 사용할 수 있습니다. 자세한 내용은 가상 대상 사용 섹션을 참조하세요.
4. 소스 Teradata 데이터베이스와 Amazon Redshift 대상을 포함하는 새 매핑 규칙을 생성합니다. 자세한 내용은 새 매핑 규칙 추가 섹션을 참조하세요.
5. 보기 메뉴에서 Main view를 선택합니다.
6. 왼쪽 패널에서 스크립트 노드를 확장합니다.
7. 쉘을 선택하고 컨텍스트(마우스 오른쪽 버튼 클릭) 메뉴를 연 다음 Load scripts를 선택합니다.
8. 임베디드 Teradata BTEQ 명령이 있는 소스 쉘 스크립트의 위치를 입력하고 폴더 선택을 선택합니다.

   AWS SCT에서 Load scripts 창을 표시합니다.
9. 다음 중 하나를 수행하세요.

   - 쉘 스크립트에 대체 변수가 포함되어 있지 않은 경우 No substitution variables를 선택한 다음 확인을 선택하여 AWS SCT 프로젝트에 스크립트를 추가합니다.
   - 쉘 스크립트에 대체 변수가 포함된 경우 해당 대체 변수를 구성합니다. 자세한 내용은 쉘 스크립트에서 대체 변수 구성 섹션을 참조하세요.

AWS SCT를 사용하여 임베디드 Teradata BTEQ 명령이 있는 쉘 스크립트에서 대체 변수 구성

쉘 스크립트에는 대체 변수가 포함될 수 있습니다. 예를 들어, 대체 변수가 있는 단일 스크립트를 사용하여 다양한 환경에서 데이터베이스를 관리할 수 있습니다. AWS SCT를 사용하여 쉘 스크립트에서 대체 변수를 구성할 수 있습니다.

쉘 스크립트에서 대체 변수가 있는 BTEQ 명령을 실행하기 전에 이 쉘 스크립트 내의 모든 변수에 값을 할당해야 합니다. AWS SCT는 값을 할당한 후에만 대체 변수를 확인하고 변환할 수 있습니다.

쉘 스크립트에서 대체 변수를 구성하려면

1. AWS SCT 프로젝트에 소스 쉘 스크립트를 추가합니다. 자세한 내용은 AWS SCT 프로젝트에 쉘 스크립트 추가 섹션을 참조하세요.

   스크립트를 추가할 때 Substitution variables are used를 선택합니다.
2. Define variable format에 스크립트의 모든 대체 변수와 일치하는 정규식을 입력합니다.

   예를 들어, 대체 변수 이름이 ${로 시작하고 }로 끝나는 경우 `\$\{\w+\}` 정규식을 사용합니다. 달러 기호 또는 퍼센트 기호로 시작하는 대체 변수를 일치시키려면 `\$\w+|\%\w+` 정규식을 사용합니다.

   AWS SCT의 정규식은 Java 정규식 구문을 따릅니다. 자세한 내용은 Java 설명서에서 java.util.regex 클래스 패턴를 참조하세요.
3. 확인을 선택하여 AWS SCT 프로젝트에 스크립트를 로드한 다음 확인을 선택하여 Load scripts 창을 닫습니다.
4. 변수를 선택하여 검색된 모든 대체 변수와 해당 값을 확인합니다.
5. 값에 대체 변수의 값을 입력합니다.

AWS SCT를 사용하여 임베디드 Teradata BTEQ 명령이 있는 쉘 스크립트 변환

다음에서는 AWS SCT를 사용하여 임베디드 Teradata BTEQ 명령이 있는 쉘 스크립트를 임베디드 Amazon Redshift RSQL 명령이 있는 쉘 스크립트로 변환하는 방법을 알아봅니다.

쉘 스크립트를 변환하려면

1. AWS SCT 프로젝트에 쉘 스크립트를 추가합니다. 자세한 내용은 AWS SCT 프로젝트에 쉘 스크립트 추가 섹션을 참조하세요.
2. 대체 변수를 구성합니다. 자세한 내용은 쉘 스크립트에서 대체 변수 구성 섹션을 참조하세요.
3. 왼쪽 패널에서 스크립트 노드를 확장합니다.
4. 다음 중 하나를 수행하세요.
   - 단일 쉘 스크립트에서 BTEQ 명령을 변환하려면 쉘 노드를 확장하고 변환할 스크립트를 선택한 다음, 컨텍스트(마우스 오른쪽 버튼 클릭) 메뉴에서 Convert script를 선택합니다.
   - 여러 스크립트를 변환하려면 변환할 스크립트를 모두 선택해야 합니다. 그 다음, 쉘을 선택하고 컨텍스트(마우스 오른쪽 버튼 클릭) 메뉴를 연 다음 Convert script를 선택합니다.
5. 확인(OK)을 선택합니다.

   AWS SCT가 선택된 쉘 스크립트의 BTEQ 명령을 Amazon Redshift RSQL과 호환되는 형식으로 변환합니다. 대상 데이터베이스 패널의 스크립트 노드에서 변환된 스크립트를 찾습니다.
6. 변환된 Amazon Redshift RSQL 스크립트를 편집하거나 저장합니다. 자세한 내용은 변환된 쉘 스크립트 편집 및 저장 섹션을 참조하세요.

AWS SCT를 사용하여 임베디드 Teradata BTEQ 명령이 있는 쉘 스크립트 관리

AWS SCT 프로젝트에 여러 쉘 스크립트를 추가하거나 쉘 스크립트를 제거할 수 있습니다.

AWS SCT 프로젝트에 새 쉘 스크립트를 추가하려면

1. 왼쪽 패널에서 스크립트 노드를 확장합니다.
2. 쉘 노드를 선택하고 컨텍스트(마우스 오른쪽 버튼 클릭) 메뉴를 엽니다.
3. Load scripts를 선택합니다.
4. 새 쉘 스크립트를 추가하고 대체 변수를 구성하는 데 필요한 정보를 입력합니다. 자세한 정보는 AWS SCT 프로젝트에 쉘 스크립트 추가 및 쉘 스크립트에서 대체 변수 구성 섹션을 참조하세요.

AWS SCT 프로젝트에서 쉘 스크립트를 제거하려면

1. 왼쪽 패널의 스크립트에서 쉘 노드를 확장합니다.

2. 제거할 스크립트를 선택하고 컨텍스트(마우스 오른쪽 버튼 클릭) 메뉴를 엽니다.
3. 스크립트 삭제를 선택합니다.

AWS SCT를 사용하여 쉘 스크립트 변환을 위한 평가 보고서 생성

쉘 스크립트 변환 평가 보고서는 BTEQ 명령 및 SQL 문 변환에 대한 정보를 제공합니다. 변환은 소스 스크립트에서 Amazon Redshift RSQL과 호환되는 형식으로 이루어집니다. 평가 보고서에는 AWS SCT가 변환할 수 없는 BTEQ 명령 및 SQL 문에 대한 작업 항목이 포함되어 있습니다.

쉘 스크립트 변환 평가 보고서를 만들려면

1. 왼쪽 패널의 스크립트에서 쉘 노드를 확장합니다.
2. 변환할 스크립트를 선택하고 컨텍스트(마우스 오른쪽 버튼 클릭) 메뉴를 연 다음, 보고서 생성을 선택합니다.
3. 요약 탭을 검토합니다. 요약 탭에는 쉘 스크립트 평가 보고서의 요약 정보가 표시됩니다. 여기에는 소스 스크립트의 모든 BTEQ 명령 및 SQL 문에 대한 변환 결과가 포함됩니다.
4. (선택 사항) 쉘 스크립트 변환 평가 보고서의 로컬 사본을 PDF 파일 또는 쉼표로 구분된 값(CSV) 파일로 저장합니다.

   - 쉘 스크립트 변환 평가 보고서를 PDF 파일로 저장하려면 오른쪽 상단에서 Save to PDF를 선택합니다.

     PDF 파일에는 스크립트 변환에 대한 요약 정보, 작업 항목 및 권장 사항이 포함됩니다.
   - 쉘 스크립트 변환 평가 보고서를 CSV 파일로 저장하려면 오른쪽 상단에서 Save to CSV를 선택합니다.

     CSV 파일에는 스크립트 변환에 필요한 작업 항목, 권장 작업 및 예상 수작업의 복잡성 등이 포함됩니다.
5. 작업 항목 탭을 선택합니다. 이 탭에는 Amazon Redshift RSQL로의 수동 변환이 필요한 항목 목록이 포함되어 있습니다. 목록에서 작업 항목을 선택하면 AWS SCT가 소스 쉘 스크립트에서 해당 작업 항목이 적용되는 항목을 강조 표시합니다.

AWS SCT를 사용하여 변환된 쉘 스크립트 편집 및 저장

변환된 스크립트는 AWS SCT 프로젝트의 하단 패널에서 편집할 수 있습니다. AWS SCT는 편집된 스크립트를 프로젝트의 일부로 저장합니다.

변환된 스크립트를 저장하려면

1. 대상 데이터베이스 패널의 스크립트에서 RSQL scripts 노드를 확장합니다.
2. 변환된 스크립트를 선택하고 컨텍스트(마우스 오른쪽 버튼 클릭) 메뉴를 연 다음 스크립트 저장을 선택합니다.
3. 변환된 스크립트를 저장할 폴더의 경로를 입력하고 저장을 선택합니다.

   AWS SCT가 변환된 스크립트를 파일에 저장하고 이 파일을 엽니다.

AWS SCT를 사용하여 Teradata FastExport 작업 스크립트를 Amazon Redshift RSQL로 변환

AWS Schema Conversion Tool(AWS SCT)을 사용하여 Teradata FastExport 작업 스크립트를 Amazon Redshift RSQL로 변환할 수 있습니다.

FastExport 작업 스크립트는 Teradata 데이터베이스에서 데이터를 선택하고 내보내는 FastExport 명령 및 SQL 문 세트입니다. AWS SCT는 FastExport 명령 및 SQL 문을 Amazon Redshift RSQL과 호환되는 형식으로 변환합니다. Teradata 데이터베이스를 Amazon Redshift로 마이그레이션한 후에는 이러한 변환된 스크립트를 사용하여 Amazon Redshift 데이터베이스에서 데이터를 추출할 수 있습니다.

주제

- AWS SCT 프로젝트에 FastExport 작업 스크립트 추가
- AWS SCT를 사용하여 Teradata FastExport 작업 스크립트에서 대체 변수 구성
- AWS SCT를 사용하여 Teradata FastExport 작업 스크립트 변환
- AWS SCT를 사용하여 Teradata FastExport 작업 스크립트 관리
- AWS SCT를 사용하여 Teradata FastExport 작업 스크립트 변환을 위한 평가 보고서 생성
- AWS SCT를 사용하여 변환된 Teradata FastExport 작업 스크립트 편집 및 저장

AWS SCT 프로젝트에 FastExport 작업 스크립트 추가

단일 AWS SCT 프로젝트에 여러 스크립트를 추가할 수 있습니다.

AWS SCT 프로젝트에 FastExport 작업 스크립트를 추가하려면

1. AWS SCT에서 새 프로젝트를 생성하거나 기존 프로젝트를 엽니다. 자세한 내용은 the section called “프로젝트 생성” 섹션을 참조하세요.

2. 메뉴에서 소스 추가를 선택한 다음 Teradata를 선택하여 프로젝트에 소스 데이터베이스를 추가합니다. 자세한 내용은 Teradata를 소스로 사용 섹션을 참조하세요.
3. 메뉴에서 대상 추가를 선택하고 대상 Amazon Redshift 데이터베이스를 AWS SCT 프로젝트에 추가합니다.

   가상 Amazon Redshift 대상 데이터베이스 플랫폼을 사용할 수 있습니다. 자세한 내용은 가상 대상 사용 섹션을 참조하세요.
4. 소스 Teradata 데이터베이스와 Amazon Redshift 대상을 포함하는 새 매핑 규칙을 생성합니다. 자세한 내용은 새 매핑 규칙 추가 섹션을 참조하세요.
5. 보기 메뉴에서 Main view를 선택합니다.
6. 왼쪽 패널에서 스크립트 노드를 확장합니다.
7. FastExport를 선택하고 컨텍스트(마우스 오른쪽 버튼 클릭) 메뉴를 연 다음 Load scripts를 선택합니다.
8. Teradata FastExport 작업 스크립트의 소스 코드 위치를 입력하고 폴더 선택을 선택합니다.

   AWS SCT에서 Load scripts 창을 표시합니다.
9. 다음 중 하나를 수행하세요.

   - Teradata FastExport 작업 스크립트에 대체 변수가 포함되어 있지 않은 경우 No substitution variables를 선택한 다음 확인을 선택하여 AWS SCT 프로젝트에 스크립트를 추가합니다.
   - Teradata FastExport 작업 스크립트에 대체 변수가 포함된 경우 해당 대체 변수를 구성합니다. 자세한 내용은 FastExport 작업 스크립트에서 대체 변수 구성 섹션을 참조하세요.

AWS SCT를 사용하여 Teradata FastExport 작업 스크립트에서 대체 변수 구성

Teradata FastExport 작업 스크립트에는 대체 변수가 포함될 수 있습니다. 예를 들어, 대체 변수가 있는 단일 스크립트를 사용하여 여러 데이터베이스의 데이터를 내보낼 수 있습니다. AWS SCT를 사용하여 Teradata 스크립트에서 대체 변수를 구성할 수 있습니다.

대체 변수가 있는 FastExport 작업 스크립트를 실행하기 전에 모든 변수에 값을 할당해야 합니다. 이렇게 하려면 Bash 스크립트, UC4(Automic) 등과 같은 다른 도구 또는 애플리케이션을 사용할 수 있습니다. AWS SCT는 값을 할당한 후에만 대체 변수를 해석하고 변환할 수 있습니다.

FastExport 작업 스크립트에서 대체 변수를 구성하려면

1. 소스 Teradata FastExport 작업 스크립트를 AWS SCT 프로젝트에 추가합니다. 자세한 내용은 AWS SCT 프로젝트에 BTEQ 스크립트 추가 섹션을 참조하세요.

   스크립트를 추가할 때 Substitution variables are used를 선택합니다.

2. Define variable format에 스크립트의 모든 대체 변수와 일치하는 정규식을 입력합니다.

   예를 들어, 대체 변수 이름이 ${로 시작하고 }로 끝나는 경우 \$\{\w+\} 정규식을 사용합니다. 달러 기호 또는 퍼센트 기호로 시작하는 대체 변수를 일치시키려면 \$\w+|\%\w+ 정규식을 사용합니다.

   AWS SCT의 정규식은 Java 정규식 구문을 따릅니다. 자세한 내용은 Java 설명서에서 java.util.regex 클래스 패턴를 참조하세요.

3. 확인을 선택하여 AWS SCT 프로젝트에 스크립트를 로드한 다음 확인을 선택하여 Load scripts 창을 닫습니다.

4. 왼쪽 패널에서 스크립트 노드를 확장합니다. FastExport를 선택한 다음 스크립트가 있는 폴더를 선택합니다. 컨텍스트(마우스 오른쪽 버튼 클릭) 메뉴를 연 다음 Substitution variables에서 Export variables를 선택합니다.

5. 한 스크립트의 대체 변수를 내보냅니다. 스크립트가 있는 폴더를 확장하고, 스크립트를 선택하고, 컨텍스트(마우스 오른쪽 버튼 클릭) 메뉴를 열고, Substitution variables에서 Export variables를 선택합니다.

6. 대체 변수를 저장할 쉼표로 구분된 값(CSV) 파일의 이름을 입력하고 저장을 선택합니다.

7. 이 CSV 파일을 열고 대체 변수의 값을 입력합니다.

   AWS SCT는 운영 체제에 따라 CSV 파일에 서로 다른 형식을 사용합니다. 파일의 값은 따옴표로 묶일 수도 있고 그렇지 않을 수도 있습니다. 대체 변수 값에는 파일의 다른 값과 동일한 형식을 사용해야 합니다. AWS SCT는 다른 형식의 값이 있는 CSV 파일은 가져올 수 없습니다.

8. CSV 파일을 저장합니다.

9. 왼쪽 패널에서 스크립트 노드를 확장합니다. FastExport를 선택한 다음 스크립트를 선택합니다. 컨텍스트(마우스 오른쪽 버튼 클릭) 메뉴를 연 다음 Substitution variables에서 Import variables를 선택합니다.

10. CSV 파일을 선택한 후 열기를 선택합니다.

11. 변수를 선택하여 검색된 모든 대체 변수와 해당 값을 확인합니다.

AWS SCT를 사용하여 Teradata FastExport 작업 스크립트 변환

다음에서는 AWS SCT를 사용하여 Teradata FastExport 작업을 Amazon Redshift RSQL로 변환하는 방법을 알아봅니다.

Teradata FastExport 작업 스크립트를 Amazon Redshift RSQL로 변환하려면

1. AWS SCT 프로젝트에 FastExport 작업 스크립트를 추가합니다. 자세한 내용은 AWS SCT 프로젝트에 FastExport 작업 스크립트 추가 섹션을 참조하세요.
2. 대체 변수를 구성합니다. 자세한 내용은 FastExport 작업 스크립트에서 대체 변수 구성 섹션을 참조하세요.
3. 왼쪽 패널에서 스크립트 노드를 확장합니다.
4. 다음 중 하나를 수행하세요.

   - 단일 FastExport 작업 스크립트를 변환하려면 FastExport 노드를 확장하고 변환할 스크립트를 선택한 다음, 컨텍스트(마우스 오른쪽 버튼 클릭) 메뉴에서 Convert script를 선택합니다.
   - 여러 스크립트를 변환하려면 변환할 스크립트를 모두 선택해야 합니다. 그 다음, FastExport를 선택하고 컨텍스트(마우스 오른쪽 버튼 클릭) 메뉴를 연 다음 Convert script를 선택합니다.

   AWS SCT가 선택된 모든 Teradata FastExport 작업 스크립트를 Amazon Redshift RSQL과 호환되는 형식으로 변환합니다. 대상 데이터베이스 패널의 스크립트 노드에서 변환된 스크립트를 찾습니다.
5. 변환된 Amazon Redshift RSQL 스크립트를 편집하거나 저장합니다. 자세한 내용은 변환된 FastExport 작업 스크립트 편집 및 저장 섹션을 참조하세요.

AWS SCT를 사용하여 Teradata FastExport 작업 스크립트 관리

Teradata FastExport 작업 스크립트를 여러 개 추가하거나 AWS SCT 프로젝트에서 FastExport 작업 스크립트를 제거할 수 있습니다.

AWS SCT 프로젝트에 새 FastExport 작업 스크립트를 추가하려면

1. 왼쪽 패널에서 스크립트 노드를 확장합니다.
2. FastExport 노드를 선택하고 컨텍스트(마우스 오른쪽 버튼 클릭) 메뉴를 엽니다.
3. Load scripts를 선택합니다.

4. 새 FastExport 작업 스크립트를 추가하고 대체 변수를 구성하는 데 필요한 정보를 입력합니다. 자세한 정보는 AWS SCT 프로젝트에 FastExport 작업 스크립트 추가 및 FastExport 작업 스크립트에서 대체 변수 구성 섹션을 참조하세요.

AWS SCT 프로젝트에서 FastExport 작업 스크립트를 제거하려면

1. 왼쪽 패널의 스크립트에서 FastExport 노드를 확장합니다.
2. 제거할 스크립트를 선택하고 컨텍스트(마우스 오른쪽 버튼 클릭) 메뉴를 엽니다.
3. 스크립트 삭제를 선택합니다.

AWS SCT를 사용하여 Teradata FastExport 작업 스크립트 변환을 위한 평가 보고서 생성

FastExport 작업 스크립트 변환 평가 보고서는 FastExport 스크립트의 FastExport 명령 및 SQL 문을 Amazon Redshift RSQL과 호환되는 형식으로 변환하는 방법에 대한 정보를 제공합니다. 평가 보고서에는 AWS SCT가 변환할 수 없는 FastExport 명령 및 SQL 문에 대한 작업 항목이 포함되어 있습니다.

Teradata FastExport 작업에 대한 스크립트 변환 평가 보고서를 생성하려면

1. 왼쪽 패널의 스크립트에서 FastExport 노드를 확장합니다.
2. 변환할 스크립트를 선택하고 컨텍스트(마우스 오른쪽 버튼 클릭) 메뉴를 연 다음, 보고서 생성을 선택합니다.
3. 요약 탭을 검토합니다. 요약 탭에는 FastExport 작업 스크립트 평가 보고서의 요약 정보가 표시됩니다. 여기에는 소스 스크립트의 모든 FastExport 명령 및 SQL 문에 대한 변환 결과가 포함됩니다.
4. FastExport 작업 스크립트 변환 평가 보고서의 로컬 사본을 PDF 파일 또는 쉼표로 구분된 값(CSV) 파일로 저장할 수 있습니다.

   a. FastExport 작업 스크립트 변환 평가 보고서를 PDF 파일로 저장하려면 오른쪽 상단에서 Save to PDF를 선택합니다.

      PDF 파일에는 스크립트 변환에 대한 요약 정보, 작업 항목 및 권장 사항이 포함됩니다.

   b. FastExport 작업 스크립트 변환 평가 보고서를 CSV 파일로 저장하려면 오른쪽 상단에서 Save to CSV를 선택합니다.

      CSV 파일에는 스크립트 변환에 필요한 작업 항목, 권장 작업 및 예상 수작업의 복잡성 등이 포함됩니다.

5. 작업 항목 탭을 선택합니다. 이 탭에는 Amazon Redshift RSQL로의 수동 변환이 필요한 항목 목록이 포함되어 있습니다. 목록에서 작업 항목을 선택하면 AWS SCT가 소스 FastExport 작업 스크립트에서 해당 작업 항목이 적용되는 항목을 강조 표시합니다.

AWS SCT를 사용하여 변환된 Teradata FastExport 작업 스크립트 편집 및 저장

변환된 스크립트는 AWS SCT 프로젝트의 하단 패널에서 편집할 수 있습니다. AWS SCT는 편집된 스크립트를 프로젝트의 일부로 저장합니다.

변환된 스크립트를 저장하려면

1. 대상 데이터베이스 패널의 스크립트에서 RSQL scripts 노드를 확장합니다.
2. 변환된 스크립트를 선택하고 컨텍스트(마우스 오른쪽 버튼 클릭) 메뉴를 연 다음 스크립트 저장을 선택합니다.
3. 변환된 스크립트를 저장할 폴더의 경로를 입력하고 저장을 선택합니다.

   AWS SCT가 변환된 스크립트를 파일에 저장하고 이 파일을 엽니다.

AWS SCT를 사용하여 Teradata FastLoad 작업 스크립트를 Amazon Redshift RSQL로 변환

AWS Schema Conversion Tool(AWS SCT)을 사용하여 Teradata FastLoad 작업 스크립트를 Amazon Redshift RSQL로 변환할 수 있습니다.

Teradata FastLoad 스크립트는 여러 세션을 사용하여 Teradata 데이터베이스의 빈 테이블에 데이터를 로드하는 명령 세트입니다. Teradata FastLoad는 일련의 Teradata FastLoad 명령과 SQL 문을 처리합니다. Teradata FastLoad 명령은 데이터 전송 시 세션 제어 및 데이터 처리를 제공합니다. SQL 문은 테이블을 생성, 유지 관리 및 삭제합니다.

AWS SCT는 Teradata FastLoad 명령 및 SQL 문을 Amazon Redshift RSQL과 호환되는 형식으로 변환합니다. Teradata 데이터베이스를 Amazon Redshift로 마이그레이션한 후에는 이러한 변환된 스크립트를 사용하여 Amazon Redshift 데이터베이스로 데이터를 로드할 수 있습니다.

주제

- AWS SCT 프로젝트에 FastLoad 작업 스크립트 추가
- AWS SCT를 사용하여 Teradata FastLoad 작업 스크립트에서 대체 변수 구성

- AWS SCT를 사용하여 Teradata FastLoad 작업 스크립트 변환
- AWS SCT를 사용하여 Teradata FastLoad 작업 스크립트 관리
- AWS SCT를 사용하여 Teradata FastLoad 작업 스크립트 변환을 위한 평가 보고서 생성
- AWS SCT를 사용하여 변환된 Teradata FastLoad 작업 스크립트 편집 및 저장

AWS SCT 프로젝트에 FastLoad 작업 스크립트 추가

단일 AWS SCT 프로젝트에 여러 스크립트를 추가할 수 있습니다.

AWS SCT 프로젝트에 FastLoad 작업 스크립트를 추가하려면

1. AWS SCT에서 새 프로젝트를 생성하거나 기존 프로젝트를 엽니다. 자세한 내용은 the section called “프로젝트 생성” 섹션을 참조하세요.
2. 메뉴에서 소스 추가를 선택한 다음 Teradata를 선택하여 프로젝트에 소스 데이터베이스를 추가합니다. 자세한 내용은 Teradata를 소스로 사용 섹션을 참조하세요.
3. 메뉴에서 대상 추가를 선택하고 대상 Amazon Redshift 데이터베이스를 AWS SCT 프로젝트에 추가합니다.

   가상 Amazon Redshift 대상 데이터베이스 플랫폼을 사용할 수 있습니다. 자세한 내용은 가상 대상 사용 섹션을 참조하세요.
4. 소스 Teradata 데이터베이스와 Amazon Redshift 대상을 포함하는 새 매핑 규칙을 생성합니다. 자세한 내용은 새 매핑 규칙 추가 섹션을 참조하세요.
5. 보기 메뉴에서 Main view를 선택합니다.
6. 왼쪽 패널에서 스크립트 노드를 확장합니다.
7. FastLoad를 선택하고 컨텍스트(마우스 오른쪽 버튼 클릭) 메뉴를 연 다음 Load scripts를 선택합니다.
8. 소스 Teradata FastLoad 작업 스크립트 위치를 입력하고 폴더 선택을 선택합니다.

   AWS SCT에서 Load scripts 창을 표시합니다.
9. 다음 중 하나를 수행하세요.

   - Teradata FastLoad 작업 스크립트에 대체 변수가 포함되어 있지 않은 경우 No substitution variables를 선택한 다음 확인을 선택하여 AWS SCT 프로젝트에 스크립트를 추가합니다.
   - Teradata FastLoad 작업 스크립트에 대체 변수가 포함된 경우 해당 대체 변수를 구성합니다. 자세한 내용은 FastLoad 작업 스크립트에서 대체 변수 구성 섹션을 참조하세요.

AWS SCT를 사용하여 Teradata FastLoad 작업 스크립트에서 대체 변수 구성

Teradata FastLoad 작업 스크립트에는 대체 변수가 포함될 수 있습니다. 예를 들어, 대체 변수가 있는 단일 스크립트를 사용하여 여러 데이터베이스에 데이터를 로드할 수 있습니다.

대체 변수가 있는 FastLoad 작업 스크립트를 실행하기 전에 모든 변수에 값을 할당해야 합니다. 이 작업을 수행하려면 Bash 스크립트, UC4(Automic) 등과 같은 다른 도구 또는 애플리케이션을 사용할 수 있습니다.

AWS SCT는 값을 할당한 후에만 대체 변수를 처리하고 변환할 수 있습니다. 소스 Teradata FastLoad 작업 스크립트의 변환을 시작하기 전에 모든 대체 변수에 값을 할당해야 합니다. AWS SCT를 사용하여 Teradata 스크립트에서 대체 변수를 구성할 수 있습니다.

FastLoad 작업 스크립트에서 대체 변수를 구성하려면

1. 소스 Teradata FastLoad 작업 스크립트를 AWS SCT 프로젝트에 추가할 경우 Substitution variables are used를 선택합니다. 이러한 스크립트의 추가에 자세한 내용은 AWS SCT 프로젝트에 FastLoad 작업 스크립트 추가 섹션을 참조하세요.
2. Define variable format에 스크립트의 모든 대체 변수와 일치하는 정규식을 입력합니다.

   예를 들어, 대체 변수 이름이 ${로 시작하고 }로 끝나는 경우 `\$\{\w+\}` 정규식을 사용합니다. 달러 기호 또는 퍼센트 기호로 시작하는 대체 변수를 일치시키려면 `\$\w+|\%\w+` 정규식을 사용합니다.

   AWS SCT의 정규식은 Java 정규식 구문을 따릅니다. 자세한 내용은 Java 설명서에서 java.util.regex 클래스 패턴를 참조하세요.
3. 확인을 선택하여 AWS SCT 프로젝트에 스크립트를 로드한 다음 확인을 선택하여 Load scripts 창을 닫습니다.
4. 왼쪽 패널에서 스크립트 노드를 확장합니다. FastLoad를 선택한 다음 스크립트가 있는 폴더를 선택합니다. 컨텍스트(마우스 오른쪽 버튼 클릭) 메뉴를 연 다음 Substitution variables에서 Export variables를 선택합니다.

   또한 한 스크립트의 대체 변수를 내보낼 수 있습니다. 스크립트가 있는 폴더를 확장하고, 스크립트를 선택하고, 컨텍스트(마우스 오른쪽 버튼 클릭) 메뉴를 열고, Substitution variables에서 Export variables를 선택합니다.
5. 대체 변수를 저장할 쉼표로 구분된 값(CSV) 파일의 이름을 입력하고 저장을 선택합니다.
6. 이 CSV 파일을 열고 대체 변수의 값을 입력합니다.

AWS SCT는 운영 체제에 따라 CSV 파일에 서로 다른 형식을 사용합니다. 파일의 값은 따옴표로 묶일 수도 있고 그렇지 않을 수도 있습니다. 대체 변수 값에는 파일의 다른 값과 동일한 형식을 사용해야 합니다. AWS SCT는 다른 형식의 값이 있는 CSV 파일은 가져올 수 없습니다.

7. CSV 파일을 저장합니다.
8. 왼쪽 패널에서 스크립트 노드를 확장합니다. FastLoad를 선택한 다음 스크립트를 선택합니다. 컨텍스트(마우스 오른쪽 버튼 클릭) 메뉴를 연 다음 Substitution variables에서 Import variables를 선택합니다.
9. CSV 파일을 선택한 후 열기를 선택합니다.
10. 변수를 선택하여 검색된 모든 대체 변수와 해당 값을 확인합니다.

AWS SCT를 사용하여 Teradata FastLoad 작업 스크립트 변환

다음에서는 AWS SCT를 사용하여 Teradata FastLoad 작업을 Amazon Redshift RSQL로 변환하는 방법을 알아봅니다.

Teradata FastLoad 작업 스크립트를 Amazon Redshift RSQL로 변환하려면

1. AWS SCT 프로젝트에 FastLoad 작업 스크립트를 추가합니다. 자세한 내용은 AWS SCT 프로젝트에 FastLoad 작업 스크립트 추가 섹션을 참조하세요.
2. 대체 변수를 구성합니다. 자세한 내용은 FastLoad 작업 스크립트에서 대체 변수 구성 섹션을 참조하세요.
3. 왼쪽 패널에서 스크립트 노드를 확장합니다.
4. 다음 중 하나를 수행하세요.

   - 단일 FastLoad 작업 스크립트를 변환하려면 FastLoad 노드를 확장하고 변환할 스크립트를 선택한 다음, 컨텍스트(마우스 오른쪽 버튼 클릭) 메뉴에서 Convert script를 선택합니다.
   - 여러 스크립트를 변환하려면 변환할 스크립트를 모두 선택해야 합니다. FastLoad를 선택하고 컨텍스트(마우스 오른쪽 버튼 클릭) 메뉴를 연 다음 Convert script를 선택합니다. 그런 다음, 다음 중 하나를 수행하세요.
     - Amazon S3에 소스 데이터 파일을 저장하는 경우 Source data file location에 대해 S3 object path를 선택합니다.

       소스 데이터 파일의 Amazon S3 bucket folder 및 Amazon S3 bucket for manifest file에 값을 입력합니다.

- Amazon S3에 소스 데이터 파일을 저장하지 않는 경우 Source data file location에 대해 Host address를 선택합니다.

  소스 데이터 파일의 URL or IP address of the host, Host user login name, Amazon S3 bucket for manifest file에 값을 입력합니다.

5. 확인(OK)을 선택합니다.

   AWS SCT가 선택된 모든 Teradata FastLoad 작업 스크립트를 Amazon Redshift RSQL과 호환되는 형식으로 변환합니다. 대상 데이터베이스 패널의 스크립트 노드에서 변환된 스크립트를 찾습니다.

6. 변환된 Amazon Redshift RSQL 스크립트를 편집하거나 저장합니다. 자세한 내용은 변환된 FastLoad 작업 스크립트 편집 및 저장 섹션을 참조하세요.

AWS SCT를 사용하여 Teradata FastLoad 작업 스크립트 관리

Teradata FastLoad 작업 스크립트를 여러 개 추가하거나 AWS SCT 프로젝트에서 FastLoad 작업 스크립트를 제거할 수 있습니다.

AWS SCT 프로젝트에 새 FastLoad 작업 스크립트를 추가하려면

1. 왼쪽 패널에서 스크립트 노드를 확장합니다.
2. FastLoad 노드를 선택하고 컨텍스트(마우스 오른쪽 버튼 클릭) 메뉴를 엽니다.
3. Load scripts를 선택합니다.
4. 새 FastLoad 작업 스크립트를 추가하고 대체 변수를 구성하는 데 필요한 정보를 입력합니다. 자세한 정보는 AWS SCT 프로젝트에 FastLoad 작업 스크립트 추가 및 FastLoad 작업 스크립트에서 대체 변수 구성 섹션을 참조하세요.

AWS SCT 프로젝트에서 FastLoad 작업 스크립트를 제거하려면

1. 왼쪽 패널의 스크립트에서 FastLoad 노드를 확장합니다.
2. 제거할 스크립트를 선택하고 컨텍스트(마우스 오른쪽 버튼 클릭) 메뉴를 엽니다.
3. 스크립트 삭제를 선택합니다.

AWS SCT를 사용하여 Teradata FastLoad 작업 스크립트 변환을 위한 평가 보고서 생성

FastLoad 작업 스크립트 변환 평가 보고서는 FastLoad 명령 및 SQL 문 변환에 대한 정보를 제공합니다. 변환은 소스 스크립트에서 Amazon Redshift RSQL과 호환되는 형식으로 이루어집니다. 평가 보고서에는 AWS SCT가 변환할 수 없는 FastLoad 명령 및 SQL 문에 대한 작업 항목이 포함되어 있습니다.

Teradata FastLoad 작업에 대한 스크립트 변환 평가 보고서를 생성하려면

1. 왼쪽 패널의 스크립트에서 FastLoad 노드를 확장합니다.
2. 변환할 스크립트를 선택하고 컨텍스트(마우스 오른쪽 버튼 클릭) 메뉴를 연 다음, 보고서 생성을 선택합니다.
3. 요약 탭을 검토합니다.

   요약 탭에는 FastLoad 작업 스크립트 평가 보고서의 요약 정보가 표시됩니다. 여기에는 소스 스크립트의 모든 FastLoad 명령 및 SQL 문에 대한 변환 결과가 포함됩니다.
4. (선택 사항) FastLoad 작업 스크립트 변환 평가 보고서의 로컬 사본을 PDF 파일 또는 쉼표로 구분된 값(CSV) 파일로 저장합니다.

   - FastLoad 작업 스크립트 변환 평가 보고서를 PDF 파일로 저장하려면 오른쪽 상단에서 Save to PDF를 선택합니다.

     PDF 파일에는 스크립트 변환에 대한 요약 정보, 작업 항목 및 권장 사항이 포함됩니다.
   - FastLoad 작업 스크립트 변환 평가 보고서를 CSV 파일로 저장하려면 오른쪽 상단에서 Save to CSV를 선택합니다.

     CSV 파일에는 스크립트 변환에 필요한 작업 항목, 권장 작업 및 예상 수작업의 복잡성 등이 포함됩니다.
5. 작업 항목 탭을 선택합니다. 이 탭에는 Amazon Redshift RSQL로의 수동 변환이 필요한 항목 목록이 포함되어 있습니다. 목록에서 작업 항목을 선택하면 AWS SCT가 소스 FastLoad 작업 스크립트에서 해당 작업 항목이 적용되는 항목을 강조 표시합니다.

AWS SCT를 사용하여 변환된 Teradata FastLoad 작업 스크립트 편집 및 저장

변환된 스크립트는 AWS SCT 프로젝트의 하단 패널에서 편집할 수 있습니다. AWS SCT는 편집된 스크립트를 프로젝트의 일부로 저장합니다.

변환된 스크립트를 저장하려면

1. 대상 데이터베이스 패널의 스크립트에서 RSQL scripts 노드를 확장합니다.
2. 변환된 스크립트를 선택하고 컨텍스트(마우스 오른쪽 버튼 클릭) 메뉴를 연 다음 스크립트 저장을 선택합니다.
3. 변환된 스크립트를 저장할 폴더의 경로를 입력하고 저장을 선택합니다.

   AWS SCT가 변환된 스크립트를 파일에 저장하고 이 파일을 엽니다.

AWS SCT를 사용하여 Teradata MultiLoad 작업 스크립트를 Amazon Redshift RSQL로 변환

AWS SCT를 사용하여 Teradata MultiLoad 작업 스크립트를 Amazon Redshift RSQL로 변환할 수 있습니다.

Teradata MultiLoad 작업 스크립트는 Teradata 데이터베이스의 일괄 유지 관리를 위한 명령 세트입니다. Teradata MultiLoad 가져오기 작업은 최대 5개의 서로 다른 테이블과 보기에서 다양한 삽입, 업데이트 및 삭제 작업을 수행합니다. Teradata MultiLoad 삭제 작업은 단일 테이블에서 많은 수의 행을 제거할 수 있습니다.

AWS SCT는 Teradata MultiLoad 명령 및 SQL 문을 Amazon Redshift RSQL과 호환되는 형식으로 변환합니다. Teradata 데이터베이스를 Amazon Redshift로 마이그레이션한 후에는 이러한 변환된 스크립트를 사용하여 Amazon Redshift 데이터베이스의 데이터를 관리합니다.

주제

- AWS SCT 프로젝트에 MultiLoad 작업 스크립트 추가
- AWS SCT를 사용하여 Teradata MultiLoad 작업 스크립트에서 대체 변수 구성
- AWS SCT를 사용하여 Teradata MultiLoad 작업 스크립트 변환
- AWS SCT를 사용하여 Teradata MultiLoad 작업 스크립트 관리
- AWS SCT를 사용하여 Teradata MultiLoad 작업 스크립트 변환을 위한 평가 보고서 생성
- AWS SCT를 사용하여 변환된 Teradata MultiLoad 작업 스크립트 편집 및 저장

AWS SCT 프로젝트에 MultiLoad 작업 스크립트 추가

단일 AWS SCT 프로젝트에 여러 스크립트를 추가할 수 있습니다.

AWS SCT 프로젝트에 MultiLoad 작업 스크립트를 추가하려면

1. AWS SCT에서 새 프로젝트를 생성하거나 기존 프로젝트를 엽니다. 자세한 내용은 the section called “프로젝트 생성” 섹션을 참조하세요.
2. 메뉴에서 소스 추가를 선택한 다음 Teradata를 선택하여 프로젝트에 소스 데이터베이스를 추가합니다. 자세한 내용은 Teradata를 소스로 사용 섹션을 참조하세요.
3. 메뉴에서 대상 추가를 선택하고 대상 Amazon Redshift 데이터베이스를 AWS SCT 프로젝트에 추가합니다.

   가상 Amazon Redshift 대상 데이터베이스 플랫폼을 사용할 수 있습니다. 자세한 내용은 가상 대상 사용 섹션을 참조하세요.
4. 소스 Teradata 데이터베이스와 Amazon Redshift 대상을 포함하는 새 매핑 규칙을 생성합니다. 자세한 내용은 새 매핑 규칙 추가 섹션을 참조하세요.
5. 보기 메뉴에서 Main view를 선택합니다.
6. 왼쪽 패널에서 스크립트 노드를 확장합니다.
7. MultiLoad를 선택하고 컨텍스트(마우스 오른쪽 버튼 클릭) 메뉴를 연 다음 Load scripts를 선택합니다.
8. 소스 Teradata MultiLoad 작업 스크립트 위치를 입력하고 폴더 선택을 선택합니다.

   AWS SCT에서 Load scripts 창을 표시합니다.
9. 다음 중 하나를 수행하세요.

   - Teradata MultiLoad 작업 스크립트에 대체 변수가 포함되어 있지 않은 경우 No substitution variables를 선택한 다음 확인을 선택하여 AWS SCT 프로젝트에 스크립트를 추가합니다.
   - Teradata MultiLoad 작업 스크립트에 대체 변수가 포함된 경우 해당 대체 변수를 구성합니다. 자세한 내용은 MultiLoad 작업 스크립트에서 대체 변수 구성 섹션을 참조하세요.

AWS SCT를 사용하여 Teradata MultiLoad 작업 스크립트에서 대체 변수 구성

Teradata MultiLoad 작업 스크립트에는 대체 변수가 포함될 수 있습니다. 예를 들어, 대체 변수가 있는 단일 스크립트를 사용하여 여러 데이터베이스에 데이터를 로드할 수 있습니다.

대체 변수가 있는 MultiLoad 작업 스크립트를 실행하기 전에 모든 변수에 값을 할당해야 합니다. 이 작업을 수행하려면 Bash 스크립트, UC4(Automic) 등과 같은 다른 도구 또는 애플리케이션을 사용할 수 있습니다.

AWS SCT는 값을 할당한 후에만 대체 변수를 처리하고 변환할 수 있습니다. 소스 Teradata MultiLoad 작업 스크립트의 변환을 시작하기 전에 모든 대체 변수에 값을 할당해야 합니다. AWS SCT를 사용하여 Teradata 스크립트에서 대체 변수를 구성할 수 있습니다.

MultiLoad 작업 스크립트에서 대체 변수를 구성하려면

1. 소스 Teradata MultiLoad 작업 스크립트를 AWS SCT 프로젝트에 추가할 경우 Substitution variables are used를 선택합니다. 이러한 스크립트의 추가에 자세한 내용은 AWS SCT 프로젝트에 MultiLoad 작업 스크립트 추가 섹션을 참조하세요.
2. Define variable format에 스크립트의 모든 대체 변수와 일치하는 정규식을 입력합니다.

   예를 들어, 대체 변수 이름이 ${로 시작하고 }로 끝나는 경우 \$\{\w+\} 정규식을 사용합니다. 달러 기호 또는 퍼센트 기호로 시작하는 대체 변수를 일치시키려면 \$\w+|\%\w+ 정규식을 사용합니다.

   AWS SCT의 정규식은 Java 정규식 구문을 따릅니다. 자세한 내용은 Java 설명서에서 java.util.regex 클래스 패턴를 참조하세요.
3. 확인을 선택하여 AWS SCT 프로젝트에 스크립트를 로드한 다음 확인을 선택하여 Load scripts 창을 닫습니다.
4. 변수를 선택하여 검색된 모든 대체 변수와 해당 값을 확인합니다.
5. 값에 대체 변수의 값을 입력합니다.

AWS SCT를 사용하여 Teradata MultiLoad 작업 스크립트 변환

다음에서는 AWS SCT를 사용하여 Teradata MultiLoad 작업을 Amazon Redshift RSQL로 변환하는 방법을 알아봅니다.

Teradata MultiLoad 작업 스크립트를 Amazon Redshift RSQL로 변환하려면

1. AWS SCT 프로젝트에 MultiLoad 작업 스크립트를 추가합니다. 자세한 내용은 AWS SCT 프로젝트에 MultiLoad 작업 스크립트 추가 섹션을 참조하세요.
2. 대체 변수를 구성하고 해당 값을 입력합니다. 자세한 내용은 MultiLoad 작업 스크립트에서 대체 변수 구성 섹션을 참조하세요.
3. 왼쪽 패널에서 스크립트 노드를 확장합니다.
4. 다음 중 하나를 수행하세요.

- 단일 MultiLoad 작업 스크립트를 변환하려면 MultiLoad 노드를 확장하고 변환할 스크립트를 선택한 다음, 컨텍스트(마우스 오른쪽 버튼 클릭) 메뉴에서 Convert script를 선택합니다.
- 여러 스크립트를 변환하려면 변환할 스크립트를 모두 선택해야 합니다. MultiLoad를 선택하고 컨텍스트(마우스 오른쪽 버튼 클릭) 메뉴를 연 다음 Convert script를 선택합니다.

5. 다음 중 하나를 수행하세요.

   - Amazon S3에 소스 데이터 파일을 저장하는 경우 Source data file location에 대해 S3 object path를 선택합니다.

     소스 데이터 파일에 대해 Amazon S3 bucket folder 및 Amazon S3 bucket for manifest file을 입력합니다.
   - Amazon S3에 소스 데이터 파일을 저장하지 않는 경우 Source data file location에 대해 Host address를 선택합니다.

     소스 데이터 파일에 대해 URL or IP address of the host, Host user login name, Amazon S3 bucket for manifest file을 입력합니다.

6. 확인(OK)을 선택합니다.

   AWS SCT가 선택된 모든 Teradata MultiLoad 작업 스크립트를 Amazon Redshift RSQL과 호환되는 형식으로 변환합니다. 대상 데이터베이스 패널의 스크립트 노드에서 변환된 스크립트를 찾습니다.

7. 변환된 Amazon Redshift RSQL 스크립트를 편집하거나 저장합니다. 자세한 내용은 변환된 MultiLoad 작업 스크립트 편집 및 저장 섹션을 참조하세요.

AWS SCT를 사용하여 Teradata MultiLoad 작업 스크립트 관리

Teradata MultiLoad 작업 스크립트를 여러 개 추가하거나 AWS SCT 프로젝트에서 MultiLoad 작업 스크립트를 제거할 수 있습니다.

AWS SCT 프로젝트에 새 MultiLoad 작업 스크립트를 추가하려면

1. 왼쪽 패널에서 스크립트 노드를 확장합니다.
2. MultiLoad 노드를 선택하고 컨텍스트(마우스 오른쪽 버튼 클릭) 메뉴를 엽니다.
3. Load scripts를 선택합니다.

4. 새 MultiLoad 작업 스크립트를 추가하고 대체 변수를 구성하는 데 필요한 정보를 입력합니다. 자세한 정보는 AWS SCT 프로젝트에 MultiLoad 작업 스크립트 추가 및 MultiLoad 작업 스크립트에서 대체 변수 구성 섹션을 참조하세요.

AWS SCT 프로젝트에서 MultiLoad 작업 스크립트를 제거하려면

1. 왼쪽 패널의 스크립트에서 MultiLoad 노드를 확장합니다.
2. 제거할 스크립트를 선택하고 컨텍스트(마우스 오른쪽 버튼 클릭) 메뉴를 엽니다.
3. 스크립트 삭제를 선택합니다.

AWS SCT를 사용하여 Teradata MultiLoad 작업 스크립트 변환을 위한 평가 보고서 생성

MultiLoad 작업 스크립트 변환 평가 보고서는 MultiLoad 명령 및 SQL 문 변환에 대한 정보를 제공합니다. 변환은 소스 스크립트에서 Amazon Redshift RSQL 명령 및 Amazon Redshift용 SQL 문으로 이루어집니다. 평가 보고서에는 AWS SCT가 변환할 수 없는 MultiLoad 명령 및 SQL 문에 대한 작업 항목이 포함되어 있습니다.

Teradata MultiLoad 작업에 대한 스크립트 변환 평가 보고서를 생성하려면

1. 왼쪽 패널의 스크립트에서 MultiLoad 노드를 확장합니다.
2. 평가 보고서를 생성할 스크립트를 선택하고 컨텍스트(마우스 오른쪽 버튼 클릭) 메뉴를 연 다음 보고서 생성을 선택합니다.
3. 요약 탭을 검토합니다. 요약 탭에는 MultiLoad 작업 스크립트 평가 보고서의 요약 정보가 표시됩니다. 여기에는 소스 스크립트의 모든 MultiLoad 명령 및 SQL 문에 대한 변환 결과가 포함됩니다.
4. (선택 사항) MultiLoad 작업 스크립트 변환 평가 보고서의 로컬 사본을 PDF 파일 또는 쉼표로 구분된 값(CSV) 파일로 저장합니다.

   - MultiLoad 작업 스크립트 변환 평가 보고서를 PDF 파일로 저장하려면 오른쪽 상단에서 Save to PDF를 선택합니다.

     PDF 파일에는 스크립트 변환에 대한 요약 정보, 작업 항목 및 권장 사항이 포함됩니다.
   - MultiLoad 작업 스크립트 변환 평가 보고서를 CSV 파일로 저장하려면 오른쪽 상단에서 Save to CSV를 선택합니다.

AWS SCT는 두 개의 CSV 파일을 생성합니다. 이러한 파일에는 스크립트 변환에 필요한 요약 정보, 작업 항목, 권장 작업 및 예상 수작업의 복잡성 등이 포함됩니다.

5. 작업 항목 탭을 선택합니다. 이 탭에는 Amazon Redshift RSQL로의 수동 변환이 필요한 항목 목록이 포함되어 있습니다. 목록에서 작업 항목을 선택하면 AWS SCT가 소스 MultiLoad 작업 스크립트에서 해당 작업 항목이 적용되는 항목을 강조 표시합니다.

AWS SCT를 사용하여 변환된 Teradata MultiLoad 작업 스크립트 편집 및 저장

변환된 스크립트는 AWS SCT 프로젝트의 하단 패널에서 편집할 수 있습니다. AWS SCT는 편집된 스크립트를 프로젝트의 일부로 저장합니다.

변환된 스크립트를 저장하려면

1. 대상 데이터베이스 패널의 스크립트에서 RSQL scripts 노드를 확장합니다.
2. 변환된 스크립트를 선택하고 컨텍스트(마우스 오른쪽 버튼 클릭) 메뉴를 연 다음 스크립트 저장을 선택합니다.
3. 변환된 스크립트를 저장할 폴더의 경로를 입력하고 저장을 선택합니다.

   AWS SCT가 변환된 스크립트를 파일에 저장하고 이 파일을 엽니다.

AWS Schema Conversion Tool을 사용한 빅 데이터 프레임워크 마이그레이션

AWS Schema Conversion Tool(AWS SCT)을 사용하여 빅 데이터 프레임워크를 AWS 클라우드로 마이그레이션할 수 있습니다.

현재 AWS SCT는 Hadoop 클러스터를 Amazon EMR 및 Amazon S3으로 마이그레이션할 수 있습니다. 이 마이그레이션 프로세스에는 Hive 및 HDFS 서비스가 포함됩니다.

또한 AWS SCT를 사용하면 Apache Oozie 오케스트레이션 워크플로를 AWS Step Functions로 변환하는 작업을 자동화할 수 있습니다.

주제

- AWS Schema Conversion Tool을 사용하여 Apache Hadoop을 Amazon EMR로 마이그레이션
- AWS Schema Conversion Tool을 사용하여 Apache Oozie를 AWS Step Functions로 변환

AWS Schema Conversion Tool을 사용하여 Apache Hadoop을 Amazon EMR로 마이그레이션

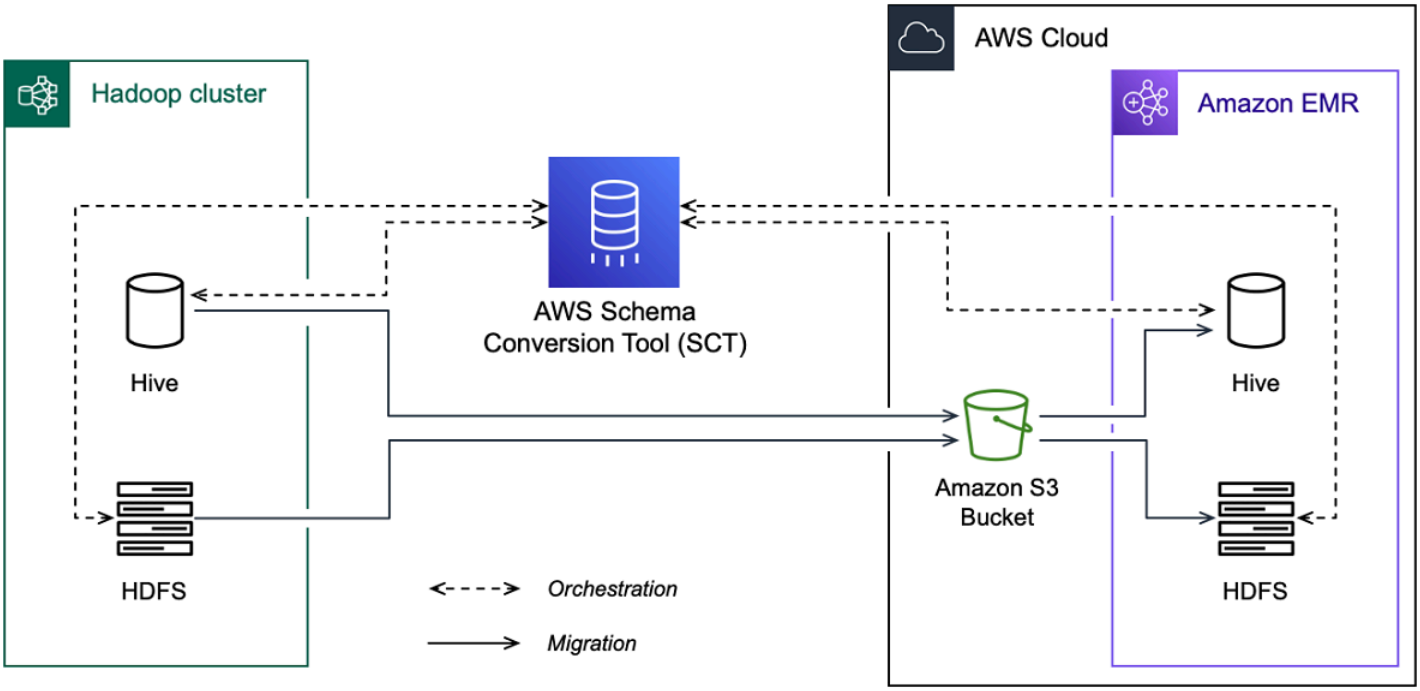
Apache Hadoop 클러스터를 마이그레이션하려면 AWS SCT 버전 1.0.670 이상을 사용해야 합니다. 또한 AWS SCT의 명령줄 인터페이스(CLI)를 숙지해야 합니다. 자세한 내용은 AWS SCT CLI 레퍼런스 섹션을 참조하세요.

주제

- 마이그레이션 개요
- 1단계: Hadoop 클러스터에 연결
- 2단계: 매핑 규칙 설정
- 3단계: 평가 보고서 생성
- 4단계: AWS SCT를 사용하여 Apache Hadoop 클러스터를 Amazon EMR로 마이그레이션
- CLI 스크립트 실행
- 빅 데이터 마이그레이션 프로젝트 관리

마이그레이션 개요

다음 이미지는 Apache Hadoop에서 Amazon EMR로의 마이그레이션에 대한 아키텍처 다이어그램을 보여줍니다.



AWS SCT는 소스 Hadoop 클러스터의 데이터 및 메타데이터를 Amazon S3 버킷으로 마이그레이션합니다. 다음으로, AWS SCT는 소스 Hive 메타데이터를 사용하여 대상 Amazon EMR Hive 서비스에서 데이터베이스 객체를 생성합니다. 경우에 따라서는 AWS Glue Data Catalog를 메타스토어로 사용하도록 Hive를 구성할 수도 있습니다. 이 경우 AWS SCT는 소스 Hive 메타데이터를 AWS Glue Data Catalog로 마이그레이션합니다.

그런 다음 AWS SCT를 사용하여 Amazon S3 버킷에서 대상 Amazon EMR HDFS 서비스로 데이터를 마이그레이션할 수 있습니다. 또는 Amazon S3 버킷에 데이터를 그대로 두고 Hadoop 워크로드를 위한 데이터 리포지토리로 사용할 수도 있습니다.

Hadoop 마이그레이션을 시작하려면 AWS SCT CLI 스크립트를 생성하여 실행합니다. 이 스크립트에는 마이그레이션을 실행하기 위한 전체 명령 세트가 포함되어 있습니다. Hadoop 마이그레이션 스크립트의 템플릿을 다운로드하여 편집할 수 있습니다. 자세한 내용은 CLI 시나리오 가져오기 섹션을 참조하세요.

Apache Hadoop에서 Amazon S3 및 Amazon EMR로 마이그레이션을 실행할 수 있도록 스크립트에는 다음 단계가 포함되어 있어야 합니다.

1단계: Hadoop 클러스터에 연결

Apache Hadoop 클러스터의 마이그레이션을 시작하려면 새 AWS SCT 프로젝트를 생성합니다. 그 다음, 소스 및 대상 클러스터에 연결합니다. 마이그레이션을 시작하기 전에 대상 AWS 리소스를 생성하고 프로비저닝해야 합니다.

이 단계에서는 다음 AWS SCT CLI 명령을 사용합니다.

- `CreateProject` – 새 AWS SCT 프로젝트를 생성합니다.
- `AddSourceCluster` - AWS SCT 프로젝트의 소스 Hadoop 클러스터에 연결합니다.
- `AddSourceClusterHive` - 프로젝트의 소스 Hive 서비스에 연결합니다.
- `AddSourceClusterHDFS` - 프로젝트의 소스 HDFS 서비스에 연결합니다.
- `AddTargetCluster` - 프로젝트의 대상 Amazon EMR 클러스터에 연결합니다.
- `AddTargetClusterS3` - Amazon S3 버킷을 프로젝트에 추가합니다.
- `AddTargetClusterHive` - 프로젝트의 대상 Hive 서비스에 연결합니다.
- `AddTargetClusterHDFS` - 프로젝트의 대상 HDFS 서비스에 연결합니다.

이러한 AWS SCT CLI 명령의 사용 예는 Apache Hadoop을 소스로 사용을 참조하세요.

소스 또는 대상 클러스터에 연결하는 명령을 실행하면 AWS SCT에서 이 클러스터에 대한 연결 설정을 시도합니다. 연결 시도가 실패하면 AWS SCT가 CLI 스크립트의 명령 실행을 중지하고 오류 메시지를 표시합니다.

2단계: 매핑 규칙 설정

소스 및 대상 클러스터에 연결한 후 매핑 규칙을 설정합니다. 매핑 규칙은 소스 클러스터의 마이그레이션 대상을 정의합니다. AWS SCT 프로젝트에 추가한 모든 소스 클러스터에 대해 매핑 규칙을 설정해야 합니다. 매핑 규칙에 대한 자세한 내용은 AWS SCT에서 매핑 규칙 생성 섹션을 참조하세요.

이 단계에서는 `AddServerMapping` 명령을 사용합니다. 이 명령은 소스 및 대상 클러스터를 정의하는 두 개의 파라미터를 사용합니다. 데이터베이스 객체의 명시적 경로 또는 객체 이름과 함께 `AddServerMapping` 명령을 사용할 수 있습니다. 첫 번째 옵션에는 객체의 유형과 이름을 포함합니다. 두 번째 옵션에는 객체 이름만 포함합니다.

- `sourceTreePath` - 소스 데이터베이스 객체의 명시적 경로입니다.

  `targetTreePath` - 대상 데이터베이스 객체의 명시적 경로입니다.

- `sourceNamePath` - 소스 객체의 이름만 포함하는 경로입니다.

  `targetNamePath` - 대상 객체의 이름만 포함하는 경로입니다.

다음 코드 예제는 소스 `testdb` Hive 데이터베이스 및 대상 EMR 클러스터의 명시적 경로를 사용하여 매핑 규칙을 생성합니다.

```
AddServerMapping
 -sourceTreePath: 'Clusters.HADOOP_SOURCE.HIVE_SOURCE.Databases.testdb'
 -targetTreePath: 'Clusters.HADOOP_TARGET.HIVE_TARGET'
/
```

Windows에서 이 예제와 다음 예제를 사용할 수 있습니다. Linux에서 CLI 명령을 실행하려면 운영 체제에 맞게 파일 경로를 업데이트해야 합니다.

다음 코드 예제는 객체 이름만 포함된 경로를 사용하여 매핑 규칙을 생성합니다.

```
AddServerMapping
 -sourceNamePath: 'HADOOP_SOURCE.HIVE_SOURCE.testdb'
 -targetNamePath: 'HADOOP_TARGET.HIVE_TARGET'
/
```

Amazon EMR 또는 Amazon S3을 소스 객체의 대상으로 선택할 수 있습니다. 각 소스 객체에 대해 단일 AWS SCT 프로젝트에서 하나의 대상만 선택할 수 있습니다. 소스 객체의 마이그레이션 대상을 변경하려면 기존 매핑 규칙을 삭제한 다음 새 매핑 규칙을 생성합니다. 매핑 규칙을 삭제하려면 `DeleteServerMapping` 명령을 사용합니다. 이 명령은 다음 두 파라미터 중 하나를 사용합니다.

- `sourceTreePath` - 소스 데이터베이스 객체의 명시적 경로입니다.
- `sourceNamePath` - 소스 객체의 이름만 포함하는 경로입니다.

`AddServerMapping` 및 `DeleteServerMapping` 명령에 대한 자세한 내용은 AWS Schema Conversion Tool CLI 참조를 확인하세요.

3단계: 평가 보고서 생성

마이그레이션을 시작하기 전에 평가 보고서를 생성하는 것이 좋습니다. 이 보고서에는 모든 마이그레이션 작업이 요약되어 있으며 마이그레이션 중에 나타날 작업 항목에 대해 자세히 설명합니다. 마이그레이션이 실패하지 않도록 하려면 마이그레이션 전에 이 보고서를 검토하여 작업 항목을 해결합니다. 자세한 내용은 마이그레이션 평가 보고서 섹션을 참조하세요.

이 단계에서는 CreateMigrationReport 명령을 사용합니다. 이 명령에서는 2개의 파라미터를 사용합니다. treePath 파라미터는 필수이고 forceMigrate 파라미터는 선택 사항입니다.

- treePath - 평가 보고서 사본을 저장할 소스 데이터베이스 객체의 명시적 경로입니다.
- forceMigrate - true로 설정하면 프로젝트에 동일한 객체를 참조하는 HDFS 폴더와 Hive 테이블이 포함되어 있는 경우에도 AWS SCT가 마이그레이션을 계속합니다. 기본값은 false입니다.

그런 다음 평가 보고서 사본을 PDF 또는 쉼표로 구분된 값(CSV) 파일로 저장할 수 있습니다. 이렇게 하려면 SaveReportPDF 또는 SaveReportCSV 명령을 사용합니다.

SaveReportPDF 명령은 평가 보고서 사본을 PDF 파일로 저장합니다. 이 명령에서는 4개의 파라미터를 사용합니다. file 파라미터는 필수이고 다른 파라미터는 선택 사항입니다.

- file - PDF 파일의 경로와 이름입니다.
- filter - 마이그레이션할 소스 객체의 범위를 정의하기 위해 이전에 만든 필터의 이름입니다.
- treePath - 평가 보고서 사본을 저장할 소스 데이터베이스 객체의 명시적 경로입니다.
- namePath - 평가 보고서 사본을 저장할 대상 객체의 이름만 포함된 경로입니다.

SaveReportCSV 명령은 평가 보고서를 3개의 CSV 파일로 저장합니다. 이 명령에서는 4개의 파라미터를 사용합니다. directory 파라미터는 필수이고 다른 파라미터는 선택 사항입니다.

- directory - AWS SCT가 CSV 파일을 저장하는 폴더의 경로입니다.
- filter - 마이그레이션할 소스 객체의 범위를 정의하기 위해 이전에 만든 필터의 이름입니다.
- treePath - 평가 보고서 사본을 저장할 소스 데이터베이스 객체의 명시적 경로입니다.
- namePath - 평가 보고서 사본을 저장할 대상 객체의 이름만 포함된 경로입니다.

다음 코드 예제는 평가 보고서 사본을 c:\sct\ar.pdf 파일에 저장합니다.

```
SaveReportPDF
 -file:'c:\sct\ar.pdf'
/
```

다음 코드 예제는 평가 보고서 사본을 c:\sct 폴더에 CSV 파일로 저장합니다.

```
SaveReportCSV
 -file:'c:\sct'
```

```
/
```

SaveReportPDF 및 SaveReportCSV 명령에 대한 자세한 내용은 AWS Schema Conversion Tool CLI 참조를 확인하세요.

4단계: AWS SCT를 사용하여 Apache Hadoop 클러스터를 Amazon EMR로 마이그레이션

AWS SCT 프로젝트를 구성한 후 온프레미스 Apache Hadoop 클러스터를 AWS 클라우드로 마이그레이션하기 시작합니다.

이 단계에서는 Migrate, MigrationStatus 및 ResumeMigration 명령을 사용합니다.

Migrate 명령은 소스 객체를 대상 클러스터로 마이그레이션합니다. 이 명령에서는 4개의 파라미터를 사용합니다. filter 또는 treePath 파라미터를 지정했는지 확인합니다. 다른 파라미터는 선택 사항입니다.

- filter - 마이그레이션할 소스 객체의 범위를 정의하기 위해 이전에 만든 필터의 이름입니다.
- treePath - 평가 보고서 사본을 저장할 소스 데이터베이스 객체의 명시적 경로입니다.
- forceLoad - true로 설정하면 마이그레이션 중에 AWS SCT가 데이터베이스 메타데이터 트리를 자동으로 로드합니다. 기본값은 false입니다.
- forceMigrate - true로 설정하면 프로젝트에 동일한 객체를 참조하는 HDFS 폴더와 Hive 테이블이 포함되어 있는 경우에도 AWS SCT가 마이그레이션을 계속합니다. 기본값은 false입니다.

MigrationStatus 명령은 마이그레이션 진행 상황에 대한 정보를 반환합니다. 이 명령을 실행하려면 name 파라미터에 마이그레이션 프로젝트의 이름을 입력합니다. CreateProject 명령에서 이 이름을 지정했습니다.

이 ResumeMigration 명령은 Migrate 명령을 사용하여 시작한 후 중단된 마이그레이션을 재개합니다. ResumeMigration 명령은 파라미터를 사용하지 않습니다. 마이그레이션을 재개하려면 소스 및 대상 클러스터에 연결해야 합니다. 자세한 내용은 마이그레이션 프로젝트 관리 섹션을 참조하세요.

다음 코드 예제는 소스 HDFS 서비스에서 Amazon EMR로 데이터를 마이그레이션합니다.

```
Migrate
 -treePath: 'Clusters.HADOOP_SOURCE.HDFS_SOURCE'
 -forceMigrate: 'true'
/
```

CLI 스크립트 실행

AWS SCT CLI 스크립트 편집을 완료한 후 확장자가 .scts인 파일로 저장합니다. 이제 AWS SCT 설치 경로의 app 폴더에서 스크립트를 실행할 수 있습니다. 이렇게 하려면 다음 명령을 사용합니다.

```
RunSCTBatch.cmd --pathtoscts "C:\script_path\hadoop.scts"
```

위 예제에서 *script_path*를 CLI 스크립트가 있는 파일의 경로로 바꿉니다. AWS SCT에서 CLI 스크립트 실행에 대한 자세한 내용은 스크립트 모드 항목을 참조하세요.

빅 데이터 마이그레이션 프로젝트 관리

마이그레이션을 완료한 후 나중에 사용할 수 있도록 AWS SCT 프로젝트를 저장하고 편집할 수 있습니다.

AWS SCT 프로젝트를 저장하려면 SaveProject 명령을 사용합니다. 이 명령은 파라미터를 사용하지 않습니다.

다음 코드 예제에서는 AWS SCT 프로젝트를 저장합니다.

```
SaveProject
/
```

AWS SCT 프로젝트를 열려면 OpenProject 명령을 사용합니다. 이 명령은 1개의 필수 파라미터를 사용합니다. file 파라미터에 대해 AWS SCT 프로젝트 파일의 경로와 이름을 입력합니다. CreateProject 명령에서 프로젝트 이름을 지정했습니다. OpenProject 명령을 실행하려면 프로젝트 파일 이름에 .scts 확장자를 추가해야 합니다.

다음 코드 예제는 c:\sct 폴더에서 hadoop_emr 프로젝트를 엽니다.

```
OpenProject
 -file: 'c:\sct\hadoop_emr.scts'
/
```

소스 및 대상 클러스터가 이미 프로젝트에 추가되어 있으므로 AWS SCT 프로젝트를 연 후 추가할 필요가 없습니다. 소스 및 대상 클러스터로 작업을 시작하려면 소스 및 대상 클러스터에 연결해야 합니다. 이렇게 하려면 ConnectSourceCluster 및 ConnectTargetCluster 명령을 사용합니다. 이들 명령은 AddSourceCluster 및 AddTargetCluster 명령과 동일한 파라미터를 사용합니다. 이러한 파라미터 목록은 변경하지 않은 상태로 CLI 스크립트를 편집하고 이러한 명령의 이름을 바꿀 수 있습니다.

다음 코드 예제는 소스 Hadoop 클러스터에 연결합니다.

```
ConnectSourceCluster
    -name: 'HADOOP_SOURCE'
    -vendor: 'HADOOP'
    -host: 'hadoop_address'
    -port: '22'
    -user: 'hadoop_user'
    -password: 'hadoop_password'
    -useSSL: 'true'
    -privateKeyPath: 'c:\path\name.pem'
    -passPhrase: 'hadoop_passphrase'
/
```

다음 코드 예제는 대상 Amazon EMR 클러스터에 연결합니다.

```
ConnectTargetCluster
 -name: 'HADOOP_TARGET'
 -vendor: 'AMAZON_EMR'
 -host: 'ec2-44-44-55-66.eu-west-1.EXAMPLE.amazonaws.com'
 -port: '22'
 -user: 'emr_user'
 -password: 'emr_password'
 -useSSL: 'true'
 -privateKeyPath: 'c:\path\name.pem'
 -passPhrase: '1234567890abcdef0!'
 -s3Name: 'S3_TARGET'
 -accessKey: 'AKIAIOSFODNN7EXAMPLE'
 -secretKey: 'wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY'
 -region: 'eu-west-1'
 -s3Path: 'doc-example-bucket/example-folder'
/
```

위 예제에서 *hadoop_address*를 Hadoop 클러스터의 IP 주소로 바꿉니다. 필요한 경우 port 변수 값을 구성합니다. 그런 다음 *hadoop_user* 및 *hadoop_password*를 Hadoop 사용자의 이름 및 이 사용자의 암호로 바꿉니다. *path\name*에는 소스 Hadoop 클러스터의 PEM 파일 이름과 경로를 입력합니다. 소스 및 대상 클러스터 추가에 대한 자세한 내용은 Apache Hadoop을 AWS SCT에 대한 소스로 사용 항목을 참조하세요.

소스 및 대상 Hadoop 클러스터에 연결한 후에는 Hive 및 HDFS 서비스와 Amazon S3 버킷에 연결해야 합니다. 이렇게 하려면 `ConnectSourceClusterHive`, `ConnectSourceClusterHdfs`,

`ConnectTargetClusterHive`, `ConnectTargetClusterHdfs` 및 `ConnectTargetClusterS3` 명령을 사용합니다. 이들 명령은 Hive 및 HDFS 서비스와 Amazon S3 버킷을 프로젝트에 추가하는 데 사용한 명령과 동일한 파라미터를 사용합니다. CLI 스크립트를 편집하여 명령 이름에서 `Add` 접두사를 `Connect`로 바꿉니다.

AWS Schema Conversion Tool을 사용하여 Apache Oozie를 AWS Step Functions로 변환

Apache Oozie 워크플로를 변환하려면 AWS SCT 버전 1.0.671 이상을 사용해야 합니다. 또한 AWS SCT의 명령줄 인터페이스(CLI)를 숙지해야 합니다. 자세한 내용은 AWS SCT CLI 레퍼런스 섹션을 참조하세요.

주제

- 변환 개요
- 1단계: 소스 및 대상 서비스에 연결
- 2단계: 매핑 규칙 설정
- 3단계: 파라미터 구성
- 4단계: 평가 보고서 생성
- 5단계: AWS SCT를 사용하여 Apache Oozie 워크플로를 AWS Step Functions로 변환
- CLI 스크립트 실행
- AWS SCT에서 AWS Step Functions로 변환할 수 있는 Apache Oozie 노드

변환 개요

Apache Oozie 소스 코드에는 작업 노드, 제어 흐름 노드 및 작업 속성이 포함됩니다. 작업 노드는 Apache Oozie 워크플로에서 실행하는 작업을 정의합니다. Apache Oozie를 사용하여 Apache Hadoop 클러스터를 오케스트레이션하면 작업 노드에 Hadoop 작업이 포함됩니다. 제어 흐름 노드는 워크플로 경로를 제어하는 메커니즘을 제공합니다. 제어 흐름 노드에는 `start`, `end`, `decision`, `fork`, `join` 등의 노드가 포함됩니다.

AWS SCT에서는 소스 작업 노드와 제어 흐름 노드를 AWS Step Functions로 변환합니다. AWS Step Functions에서는 Amazon States Language(ASL)로 워크플로를 정의합니다. AWS SCT는 ASL을 사용하여 상태 시스템을 정의합니다. 이 상태 시스템은 작업을 수행할 수 있는 상태, 다음으로 전환할 상태를 결정하는 상태, 오류를 표시하며 중지하는 상태 등의 상태 모음입니다. 그 다음으로 AWS SCT는 상태 시스템 정의가 포함된 JSON 파일을 업로드합니다. 그런 다음 AWS SCT가 AWS Identity and

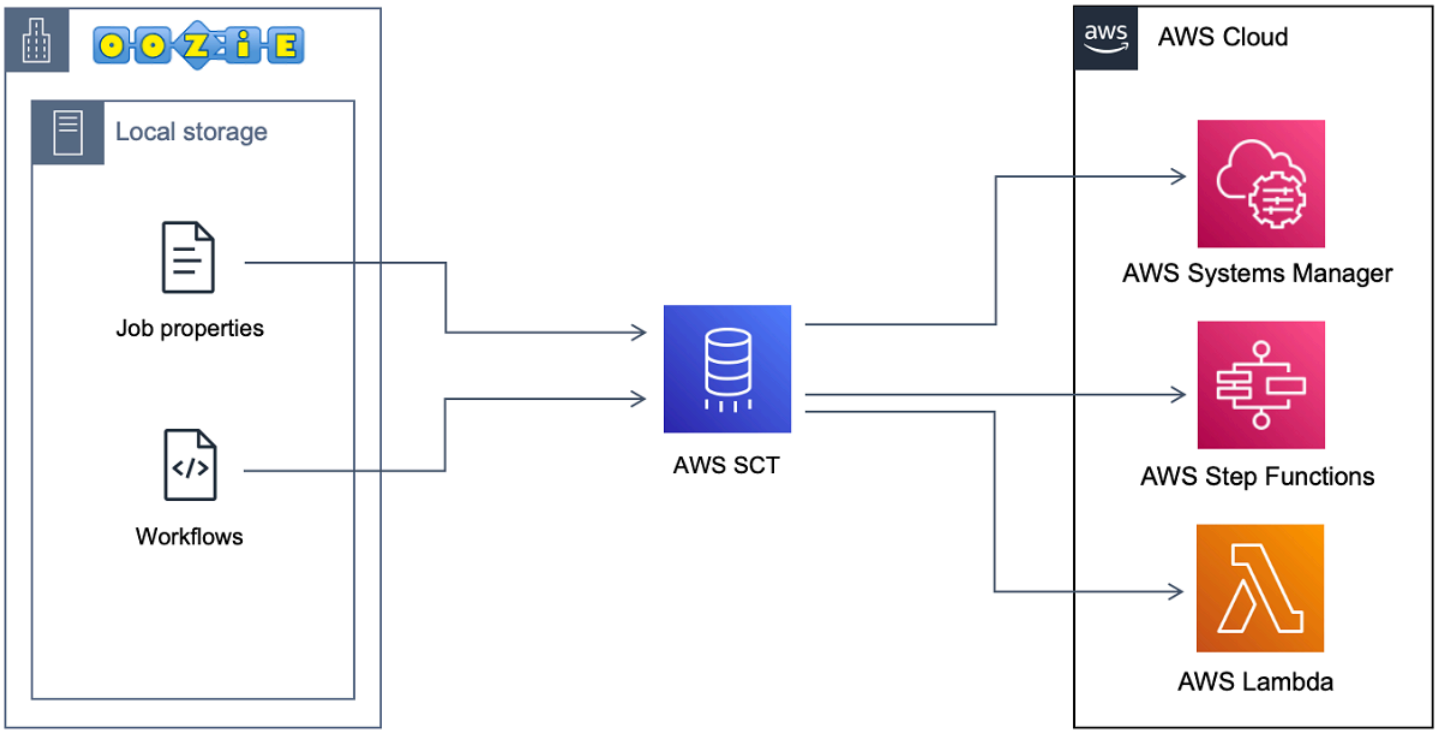
Access Management(IAM) 역할을 사용하여 AWS Step Functions에서 상태 시스템을 구성할 수 있습니다. 자세한 정보는 AWS Step Functions 개발자 설명서.의 AWS Step Functions란 무엇입니까?를 참조하세요.

또한 AWS SCT는 AWS Step Functions에서 지원하지 않는 소스 함수를 에뮬레이션하는 AWS Lambda 함수가 포함된 확장 팩을 생성합니다. 자세한 내용은 AWS SCT 확장 팩 사용 섹션을 참조하세요.

AWS SCT는 소스 작업 속성을 AWS Systems Manager로 마이그레이션합니다. AWS SCT는 파라미터 이름과 값을 저장하기 위해 AWS Systems Manager의 기능인 파라미터 스토어를 사용합니다. 자세한 내용은 AWS Systems Manager 사용 설명서의 AWS Systems Manager(이)란 무엇입니까? 섹션을 참조하세요.

AWS SCT를 사용하여 파라미터의 값과 이름을 자동으로 업데이트할 수 있습니다. Apache Oozie와 AWS Step Functions 간의 아키텍처 차이로 인해 파라미터를 구성해야 할 수도 있습니다. AWS SCT는 소스 파일에서 지정된 파라미터 이름 또는 값을 찾아서 새 값으로 바꿀 수 있습니다. 자세한 내용은 3단계: 파라미터 구성 섹션을 참조하세요.

다음 이미지는 AWS Step Functions로 Apache Oozie를 변환하는 아키텍처 다이어그램을 보여줍니다.



변환을 시작하려면 AWS SCT CLI 스크립트를 생성한 후 실행합니다. 이 스크립트에는 변환을 실행하기 위한 전체 명령 세트가 포함되어 있습니다. Apache Oozie 변환 스크립트의 템플릿을 다운로드하고 편집할 수 있습니다. 자세한 내용은 CLI 시나리오 가져오기 섹션을 참조하세요.

스크립트에 다음 단계가 포함되어야 합니다.

1단계: 소스 및 대상 서비스에 연결

Apache Oozie 클러스터의 변환을 시작하려면 새 AWS SCT 프로젝트를 생성합니다. 그 다음, 소스 및 대상 서비스에 연결합니다. 마이그레이션을 시작하기 전에 대상 AWS 리소스를 생성하고 프로비저닝해야 합니다. 자세한 내용은 Apache Oozie를 소스로 사용하기 위한 사전 조건 섹션을 참조하세요.

이 단계에서는 다음 AWS SCT CLI 명령을 사용합니다.

- `CreateProject` – 새 AWS SCT 프로젝트를 생성합니다.
- `AddSource` - AWS SCT 프로젝트에 소스 Apache Oozie 파일을 추가합니다.
- `ConnectSource` - 소스로 Apache Oozie에 연결합니다.
- `AddTarget` - AWS Step Functions를 마이그레이션 대상으로 프로젝트에 추가합니다.
- `ConnectTarget` - AWS Step Functions에 연결합니다.

이러한 AWS SCT CLI 명령의 사용 예는 Apache Oozie를 소스로 사용을 참조하세요.

`ConnectSource` 또는 `ConnectTarget` 명령을 실행하면 AWS SCT가 서비스에 대한 연결 설정을 시도합니다. 연결 시도가 실패하면 AWS SCT가 CLI 스크립트의 명령 실행을 중지하고 오류 메시지를 표시합니다.

2단계: 매핑 규칙 설정

소스 및 대상 서비스에 연결한 후 매핑 규칙을 설정합니다. 매핑 규칙은 소스 Apache Oozie 워크플로와 파라미터의 마이그레이션 대상을 정의합니다. 매핑 규칙에 대한 자세한 내용은 AWS SCT에서 매핑 규칙 생성 섹션을 참조하세요.

변환할 소스 및 대상 객체를 정의하려면 `AddServerMapping` 명령을 사용합니다. 이 명령에서는 2개의 파라미터(`sourceTreePath` 및 `targetTreePath`)를 사용합니다. 이러한 파라미터의 값에는 소스 및 대상 객체에 대한 명시적 경로가 포함됩니다. Apache Oozie를 AWS Step Functions로 변환하려면 이러한 파라미터를 `ETL`로 시작해야 합니다.

다음 코드 예제는 `OOZIE` 및 `AWS_STEP_FUNCTIONS` 객체에 대한 매핑 규칙을 생성합니다. 이전 단계에서 `AddSource` 및 `AddTarget` 명령을 사용하여 이러한 객체를 AWS SCT 프로젝트에 추가했습니다.

```
AddServerMapping
    -sourceTreePath: 'ETL.APACHE_OOZIE'
```

```
    -targetTreePath: 'ETL.AWS_STEP_FUNCTIONS'
/
```

AddServerMapping 명령에 대한 자세한 내용은 AWS Schema Conversion Tool 명령 참조를 확인하세요.

3단계: 파라미터 구성

소스 Apache Oozie 워크플로에서 파라미터를 사용하는 경우 AWS Step Functions로 변환한 후 해당 값을 변경해야 할 수 있습니다. 또한 AWS Step Functions에서 사용할 새 파라미터를 추가해야 할 수도 있습니다.

이 단계에서는 AddParameterMapping 및 AddTargetParameter 명령을 사용합니다.

소스 파일의 파라미터 값을 바꾸려면 AddParameterMapping 명령을 사용합니다. AWS SCT가 소스 파일을 스캔하고, 이름 또는 값으로 파라미터를 찾은 후 해당 값을 변경합니다. 단일 명령을 실행하여 모든 소스 파일을 스캔할 수 있습니다. 다음 목록의 처음 3개 파라미터 중 하나를 사용하여 스캔할 파일의 범위를 정의합니다. 이 명령은 최대 6개의 파라미터를 사용합니다.

- filterName - 소스 객체의 필터 이름입니다. CreateFilter 명령을 사용하여 필터를 생성할 수 있습니다.
- treePath - 소스 객체의 명시적 경로입니다.
- namePath - 특정 소스 객체의 명시적 경로입니다.
- sourceParameterName - 소스 파라미터의 이름입니다.
- sourceValue - 소스 파라미터의 값입니다.
- targetValue - 대상 파라미터의 값입니다.

다음 코드 예제에서는 값이 c:\oozie\hive.py와 동일한 모든 파라미터를 s3://bucket-oozie/hive.py 값으로 바꿉니다.

```
AddParameterMapping
 -treePath: 'ETL.OOZIE.Applications'
 -sourceValue: 'c:\oozie\hive.py'
 -targetValue: 's3://bucket-oozie/hive.py'
/
```

다음 코드 예제에서는 이름이 nameNode와 동일한 모든 파라미터를 hdfs://ip-111-222-33-44.eu-west-1.compute.internal:8020 값으로 바꿉니다.

```
AddParameterMapping
    -treePath: 'ETL.OOZIE_SOURCE.Applications'
    -sourceParameter: 'nameNode'
    -targetValue: 'hdfs://ip-111-222-33-44.eu-west-1.compute.internal:8020'
/
```

다음 코드 예제에서는 이름이 nameNode와 동일하고 값이 hdfs://ip-55.eu-west-1.compute.internal:8020과 동일한 모든 파라미터를 targetValue 파라미터의 값으로 바꿉니다.

```
AddParameterMapping
    -treePath: 'ETL.OOZIE_SOURCE.Applications'
    -sourceParameter: 'nameNode'
    -sourceValue: 'hdfs://ip-55-66-77-88.eu-west-1.compute.internal:8020'
    -targetValue: 'hdfs://ip-111-222-33-44.eu-west-1.compute.internal:8020'
/
```

소스 파일의 기존 파라미터 외에도 대상 파일에 새 파라미터를 추가하려면 AddTargetParameter 명령을 사용합니다. 이 명령은 AddParameterMapping 명령과 동일한 파라미터 세트를 사용합니다.

다음 코드 예제에서는 nameNode 파라미터 대신 clusterId 대상 파라미터를 추가합니다.

```
AddTargetParameter
    -treePath: 'ETL.OOZIE_SOURCE.Applications'
    -sourceParameter: 'nameNode'
    -sourceValue: 'hdfs://ip-55-66-77-88.eu-west-1.compute.internal:8020'
    -targetParameter: 'clusterId'
    -targetValue: '1234567890abcdef0'
/
```

AddServerMapping, AddParameterMapping, AddTargetParameter 및 CreateFilter 명령에 대한 자세한 내용은 AWS Schema Conversion Tool CLI 참조를 확인하세요.

4단계: 평가 보고서 생성

변환을 시작하기 전에 평가 보고서를 생성하는 것이 좋습니다. 이 보고서에는 모든 마이그레이션 작업이 요약되어 있으며 마이그레이션 중에 나타날 작업 항목에 대해 자세히 설명합니다. 마이그레이션이 실패하지 않도록 하려면 마이그레이션 전에 이 보고서를 검토하여 작업 항목을 해결합니다. 자세한 내용은 마이그레이션 평가 보고서 섹션을 참조하세요.

이 단계에서는 CreateReport 명령을 사용합니다. 이 명령에서는 2개의 파라미터를 사용합니다. 첫 번째 파라미터는 AWS SCT에서 평가 보고서를 생성하는 소스 개체를 설명합니다. 이 작업을 수행하려면 filterName, treePath 또는 namePath 파라미터 중 하나를 사용합니다. 이 파라미터는 필수입니다. 또한 선택적 부울 파라미터 forceLoad를 추가할 수 있습니다. 이 파라미터를 true로 설정하면 AWS SCT가 CreateReport 명령에 지정된 소스 객체의 모든 자식 객체를 자동으로 로드합니다.

다음 코드 예제에서는 소스 Oozie 파일의 Applications 노드에 대한 평가 보고서를 생성합니다.

```
CreateReport
    -treePath: 'ETL.APACHE_OOZIE.Applications'
/
```

그런 다음 평가 보고서 사본을 PDF 또는 쉼표로 구분된 값(CSV) 파일로 저장할 수 있습니다. 이렇게 하려면 SaveReportPDF 또는 SaveReportCSV 명령을 사용합니다.

SaveReportPDF 명령은 평가 보고서 사본을 PDF 파일로 저장합니다. 이 명령에서는 4개의 파라미터를 사용합니다. file 파라미터는 필수이고 다른 파라미터는 선택 사항입니다.

- file - PDF 파일의 경로와 이름입니다.
- filter - 마이그레이션할 소스 객체의 범위를 정의하기 위해 이전에 만든 필터의 이름입니다.
- treePath - 평가 보고서 사본을 저장할 소스 데이터베이스 객체의 명시적 경로입니다.
- namePath - 평가 보고서 사본을 저장할 대상 객체의 이름만 포함된 경로입니다.

SaveReportCSV 명령은 평가 보고서를 CSV 파일로 저장합니다. 이 명령에서는 4개의 파라미터를 사용합니다. directory 파라미터는 필수이고 다른 파라미터는 선택 사항입니다.

- directory - AWS SCT가 CSV 파일을 저장하는 폴더의 경로입니다.
- filter - 마이그레이션할 소스 객체의 범위를 정의하기 위해 이전에 만든 필터의 이름입니다.
- treePath - 평가 보고서 사본을 저장할 소스 데이터베이스 객체의 명시적 경로입니다.
- namePath - 평가 보고서 사본을 저장할 대상 객체의 이름만 포함된 경로입니다.

다음 코드 예제는 평가 보고서 사본을 c:\sct\ar.pdf 파일에 저장합니다.

```
SaveReportPDF
 -file:'c:\sct\ar.pdf'
/
```

다음 코드 예제는 평가 보고서 사본을 c:\sct 폴더에 CSV 파일로 저장합니다.

```
SaveReportCSV
 -file:'c:\sct'
/
```

CreateReport, SaveReportPDF 및 SaveReportCSV 명령에 대한 자세한 내용은 AWS Schema Conversion Tool CLI 참조를 확인하세요.

5단계: AWS SCT를 사용하여 Apache Oozie 워크플로를 AWS Step Functions로 변환

AWS SCT 프로젝트를 구성한 후 소스 코드를 변환하고 AWS 클라우드에 적용합니다.

이 단계에서는 Convert, SaveOnS3, ConfigureStateMachine 및 ApplyToTarget 명령을 사용합니다.

Migrate 명령은 소스 객체를 대상 클러스터로 마이그레이션합니다. 이 명령에서는 4개의 파라미터를 사용합니다. filter 또는 treePath 파라미터를 지정했는지 확인합니다. 다른 파라미터는 선택 사항입니다.

- filter - 마이그레이션할 소스 객체의 범위를 정의하기 위해 이전에 만든 필터의 이름입니다.
- namePath - 특정 소스 객체의 명시적 경로입니다.
- treePath - 평가 보고서 사본을 저장할 소스 데이터베이스 객체의 명시적 경로입니다.
- forceLoad - true로 설정하면 마이그레이션 중에 AWS SCT가 데이터베이스 메타데이터 트리를 자동으로 로드합니다. 기본값은 false입니다.

다음 코드 예제에서는 소스 Oozie 파일의 Applications 폴더에서 파일을 변환합니다.

```
Convert
    -treePath: 'ETL.APACHE_OOZIE.Applications'
/
```

SaveOnS3은 상태 시스템 정의를 Amazon S3 버킷으로 업로드합니다. 이 명령은 treePath 파라미터를 사용합니다. 이 명령을 실행하려면 상태 시스템 정의가 있는 대상 폴더를 이 파라미터의 값으로 사용합니다.

다음은 AWS_STEP_FUNCTIONS 대상 객체의 State machine definitions 폴더를 Amazon S3 버킷에 업로드합니다. AWS SCT에서는 필수 조건단계에서 AWS 서비스 프로필에 저장한 Amazon S3 버킷을 사용합니다.

```
SaveOnS3
    -treePath: 'ETL.AWS_STEP_FUNCTIONS.State machine definitions'
/
```

ConfigureStateMachine 명령은 상태 시스템을 구성합니다. 이 명령은 최대 6개의 파라미터를 사용합니다. 다음 목록의 처음 3개 파라미터 중 하나를 사용하여 대상 범위를 정의해야 합니다.

- filterName - 대상 객체의 필터 이름입니다. CreateFilter 명령을 사용하여 필터를 생성할 수 있습니다.
- treePath - 대상 객체의 명시적 경로입니다.
- namePath - 특정 대상 객체의 명시적 경로입니다.
- iamRole - 단계 시스템에 대한 액세스를 제공하는 IAM 역할의 Amazon 리소스 이름(ARN)입니다. 이 파라미터는 필수 항목입니다.

다음 코드 예제에서는 *role_name* IAM 역할을 사용하여 AWS_STEP_FUNCTIONS에 정의된 상태 시스템을 구성합니다.

```
ConfigureStateMachine
    -treePath: 'ETL.AWS_STEP_FUNCTIONS.State machine definitions'
    -role: 'arn:aws:iam::555555555555:role/role_name'
/
```

ApplyToTarget 명령은 변환된 코드를 대상 서버에 적용합니다. 이 명령을 실행하려면 filterName, treePath 또는 namePath 파라미터 중 하나를 사용하여 적용할 대상 객체를 정의합니다.

다음 코드 예제에서는 app_wp 상태 시스템을 AWS Step Functions에 적용합니다.

```
ApplyToTarget
    -treePath: 'ETL.AWS_STEP_FUNCTIONS.State machines.app_wp'
/
```

변환된 코드가 소스 코드와 동일한 결과를 생성하도록 AWS SCT 확장 팩을 사용할 수 있습니다. 이 확장 팩은 AWS Step Functions에서 지원되지 않는 Apache Oozie 함수를 에뮬레이션하는 AWS Lambda 함수 세트입니다. 이 확장 팩을 설치하려면 `CreateLambdaExtPack` 명령을 사용할 수 있습니다.

이 명령은 최대 5개의 파라미터를 사용합니다. `extPackId`에 대한 **`Oozie2SF`**를 사용해야 합니다. 이 경우 AWS SCT는 소스 Apache Oozie 함수의 확장 팩을 생성합니다.

- `extPackId` - Lambda 함수 세트의 고유 식별자입니다. 이 파라미터는 필수 항목입니다.
- `tempDirectory` - AWS SCT가 임시 파일을 저장할 수 있는 경로입니다. 이 파라미터는 필수 항목입니다.
- `awsProfile` - AWS 프로필의 이름입니다.
- `lambdaExecRoles` - Lambda 함수에 사용할 실행 역할의 Amazon 리소스 이름(ARN) 목록입니다.
- `createInvokeRoleFlag` - AWS Step Functions에 대해 실행 역할을 생성할지 여부를 나타내는 부울 플래그입니다.

확장 팩을 설치하고 사용하려면 필요한 권한을 제공해야 합니다. 자세한 내용은 확장 팩의 AWS Lambda 함수를 사용하기 위한 권한 섹션을 참조하세요.

`Convert`, `SaveOnS3`, `ConfigureStateMachine`, `ApplyToTarget` 및 `CreateLambdaExtPack` 명령에 대한 자세한 내용은 AWS Schema Conversion Tool CLI 참조를 확인하세요.

CLI 스크립트 실행

AWS SCT CLI 스크립트 편집을 완료한 후 확장자가 `.scts`인 파일로 저장합니다. 이제 AWS SCT 설치 경로의 app 폴더에서 스크립트를 실행할 수 있습니다. 이렇게 하려면 다음 명령을 사용합니다.

```
RunSCTBatch.cmd --pathtoscts "C:\script_path\oozie.scts"
```

위 예제에서 *`script_path`*를 CLI 스크립트가 있는 파일의 경로로 바꿉니다. AWS SCT에서 CLI 스크립트 실행에 대한 자세한 내용은 스크립트 모드 항목을 참조하세요.

AWS SCT에서 AWS Step Functions로 변환할 수 있는 Apache Oozie 노드

AWS SCT를 사용하여 Apache Oozie 작업 노드와 제어 흐름 노드를 AWS Step Functions로 변환할 수 있습니다.

지원되는 작업 노드는 다음과 같습니다.

- Hive 작업
- Hive2 작업
- Spark 작업
- MapReduce Streaming 작업
- Java 작업
- DistCp 작업
- Pig 작업
- Sqoop 작업
- FS 작업
- 쉘 작업

지원되는 제어 흐름 노드는 다음과 같습니다.

- Start 작업
- End 작업
- Kill 작업
- Decision 작업
- Fork 작업
- Join 작업

AWS SCT와 함께 AWS DMS 사용

AWS DMS에서 AWS SCT 복제 에이전트 사용

대규모 데이터베이스 마이그레이션에서 AWS SCT 복제 에이전트(aws-schema-conversion-tool-dms-agent)를 사용하여 온프레미스 데이터베이스에서 Amazon S3 또는 AWS Snowball Edge 디바이스로 데이터를 복사할 수 있습니다. 복제 에이전트는 AWS DMS와 함께 작동하며, AWS SCT가 닫혀 있는 동안 백그라운드에서 작동할 수 있습니다.

AWS Snowball Edge 작업 시 AWS SCT 에이전트는 AWS Snowball 디바이스로 데이터를 복제합니다. 그러면 디바이스가 AWS로 전송되고 데이터가 Amazon S3 버킷에 로드됩니다. 이 기간 동안 AWS SCT 에이전트는 계속 실행됩니다. 그런 다음 에이전트는 Amazon S3의 데이터를 가져와서 대상 엔드포인트에 복사합니다.

자세한 내용은 온프레미스 데이터 웨어하우스에서 Amazon Redshift로 데이터 마이그레이션 섹션을 참조하세요.

AWS DMS에서 AWS SCT 데이터 추출 에이전트 사용

Apache Cassandra에서 Amazon DynamoDB로 쉽게 마이그레이션할 수 있도록 도와주는 데이터 추출 에이전트(`aws-schema-conversion-tool-extractor`)를 AWS SCT에서 찾을 수 있습니다. Cassandra와 DynamoDB는 NoSQL 데이터베이스이지만 시스템 아키텍처 및 데이터 표현이 다릅니다. AWS SCT에서 마법사 기반 워크플로를 사용하면 Cassandra에서 DynamoDB로의 마이그레이션 프로세스를 자동화할 수 있습니다. AWS SCT는 AWS Database Migration Service(AWS DMS)와 통합되어 실제 마이그레이션을 수행합니다.

자세한 내용은 온프레미스 데이터 웨어하우스에서 Amazon Redshift로 데이터 마이그레이션 섹션을 참조하세요.

AWS DMS와 함께 AWS SCT 사용 시 로깅 수준 높이기

AWS Support와 협력해야 하는 경우 등과 같이 AWS SCT를 AWS DMS와 함께 사용할 때 로깅 수준을 높일 수 있습니다.

AWS SCT 및 필요한 드라이버를 설치한 후 AWS SCT 아이콘을 선택하여 애플리케이션을 엽니다. 업데이트 알림이 표시되면 프로젝트 완료 전 또는 후에 업데이트하도록 선택할 수 있습니다. 자동 프로젝트 창이 열리면 창을 닫고 프로젝트를 수동으로 생성합니다.

AWS SCT를 AWS DMS와 함께 사용할 때 로깅 수준을 높이려면

1. 설정 메뉴에서 전역 설정을 선택합니다.
2. 전역 설정 창에서 로깅을 선택합니다.
3. 디버그 모드에서 True를 선택합니다.
4. Message level 섹션에서 다음과 같은 유형의 로그를 수정할 수 있습니다.

   - 일반
   - 로더
   - 파서
   - 프린터
   - 확인자
   - 원격 측정
   - 변환기

   기본적으로 모든 메시지 수준이 정보로 설정됩니다.
5. 변경하려는 모든 메시지 수준 유형에 대한 로깅 수준을 선택합니다.

   - 추적(가장 상세한 로깅)
   - 디버깅
   - Info
   - 경고
   - 오류(가장 상세하지 않은 로깅)
   - 심각
   - 필수
6. 적용을 선택하여 프로젝트 설정을 수정합니다.
7. 확인을 선택하여 전역 설정 창을 닫습니다.

온프레미스 데이터 웨어하우스에서 Amazon Redshift로 데이터 마이그레이션

AWS SCT 에이전트를 사용하여 온프레미스 데이터 웨어하우스에서 데이터를 추출하고 Amazon Redshift로 마이그레이션할 수 있습니다. 에이전트는 데이터를 추출하여 Amazon S3 또는 대규모 마이그레이션의 경우 AWS Snowball Edge 디바이스에 데이터를 업로드합니다. 그런 다음 AWS SCT 에이전트를 사용하여 Amazon Redshift에 데이터를 복사할 수 있습니다.

또는 AWS Database Migration Service (AWS DMS) 를 사용하여 Amazon Redshift로 데이터를 마이그레이션할 수 있습니다. AWS DMS 의 장점은 지속적인 복제(변경 데이터 캡처)를 지원한다는 것입니다. 그러나 데이터 마이그레이션 속도를 높이려면 여러 AWS SCT 에이전트를 병렬로 사용하십시오. 테스트에 따르면 AWS SCT 에이전트는 15~ AWS DMS 35% 보다 빠르게 데이터를 마이그레이션합니다. 속도 차이는 데이터 압축, 테이블 파티션의 병렬 마이그레이션 지원, 다양한 구성 설정으로 인해 발생합니다. 자세한 내용은 AWS Database Migration Service대상으로 Amazon Redshift 데이터베이스 사용 섹션을 참조하세요.

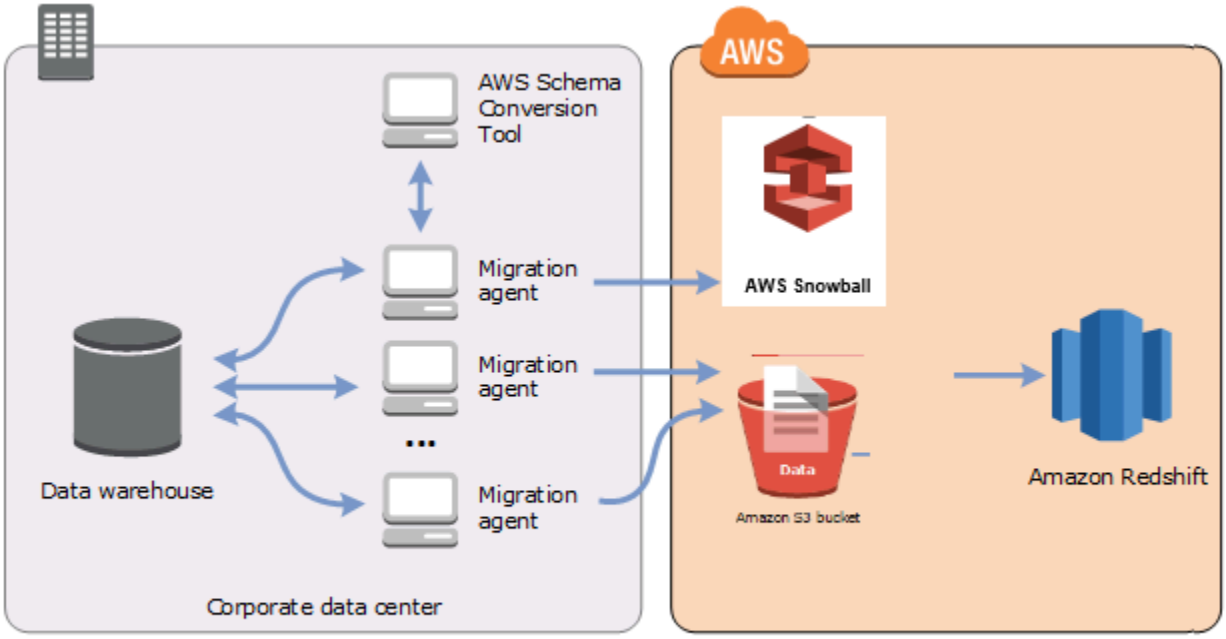
Amazon S3은 스토리지 및 검색 서비스입니다. Amazon S3에 객체를 저장하려면 저장할 파일을 Amazon S3 버킷에 업로드합니다. 파일을 업로드하면 객체 및 메타데이터에도 권한을 설정할 수 있습니다.

대규모 마이그레이션

대규모 데이터 마이그레이션에는 수 테라바이트의 정보가 포함될 수 있으며, 네트워크 성능과 이동해야 하는 데이터의 양 때문에 속도가 느려질 수 있습니다. AWS Snowball Edge는 소유 AWS 어플라이언스를 사용하여 데이터를 클라우드로 faster-than-network 빠른 속도로 전송하는 데 사용할 수 있는 서비스입니다. AWS AWS Snowball 에지 디바이스는 최대 100TB의 데이터를 저장할 수 있습니다. 256비트 암호화와 업계 표준 TPM (신뢰할 수 있는 플랫폼 모듈) 을 사용하여 데이터의 보안과 전체 chain-of-custody 보안을 모두 보장합니다. AWS SCT 에지 디바이스와 함께 작동합니다. AWS Snowball

AWS SCT 및 AWS Snowball Edge 장치를 사용하는 경우 데이터를 두 단계로 마이그레이션합니다. 먼저 를 AWS SCT 사용하여 데이터를 로컬에서 처리한 다음 해당 데이터를 AWS Snowball Edge 장치로 이동합니다. 그런 다음 AWS Snowball Edge 프로세스를 AWS 사용하여 디바이스를 전송한 다음 Amazon S3 버킷에 데이터를 AWS 자동으로 로드합니다. 다음으로, Amazon S3에서 데이터를 사용할 수 있게 되면 데이터를 Amazon Redshift로 마이그레이션하는 AWS SCT 데 사용합니다. 데이터 추출 AWS SCT 에이전트는 닫혀 있는 동안 백그라운드에서 작업할 수 있습니다.

다음 다이어그램은 지원되는 시나리오를 보여 줍니다.



데이터 추출 에이전트는 현재 다음과 같은 소스 데이터 웨어하우스에서 지원됩니다.

- Azure Synapse Analytics
- BigQuery
- Greenplum Database(버전 4.3)
- Microsoft SQL Server(버전 2008 이상)
- Netezza(버전 7.0.3 이상)
- Oracle(버전 10 이상)
- Snowflake(버전 3)
- Teradata(버전 13 이상)
- Vertica(버전 7.2.2 이상)

FIPS(Federal Information Processing Standard) 보안 요구 사항을 준수해야 하는 경우 Amazon Redshift용 FIPS 엔드포인트에 연결할 수 있습니다. FIPS 엔드포인트는 다음 AWS 지역에서 사용할 수 있습니다.

- 미국 동부(버지니아 북부) 리전(redshift-fips.us-east-1.amazonaws.com)
- 미국 동부(오하이오) 리전(redshift-fips.us-east-2.amazonaws.com)

- 미국 서부(캘리포니아 북부) 리전(redshift-fips.us-west-1.amazonaws.com)
- 미국 서부(오레곤) 리전(redshift-fips.us-west-2.amazonaws.com)

다음 항목의 정보를 사용하여 데이터 추출 에이전트를 사용하는 방법을 알아봅니다.

주제

- 데이터 추출 에이전트 사용을 위한 사전 조건
- 추출 에이전트 설치
- 추출 에이전트 구성
- 에 추출 에이전트 등록 AWS Schema Conversion Tool
- 상담원의 정보 숨기기 및 복구하기 AWS SCT
- 에서 데이터 마이그레이션 규칙 생성 AWS SCT
- 프로젝트 설정에서 추출기 및 복사 설정 변경
- 마이그레이션하기 전에 다음을 사용하여 데이터를 정렬합니다. AWS SCT
- AWS SCT 데이터 추출 작업 생성, 실행, 모니터링
- AWS SCT 데이터 추출 작업 내보내기 및 가져오기
- AWS Snowball Edge 장치를 사용한 데이터 추출
- 데이터 추출 작업 출력
- 가상 파티셔닝 사용: AWS Schema Conversion Tool
- 네이티브 파티셔닝 사용
- Amazon Redshift로 LOB 마이그레이션
- 데이터 추출 에이전트에 대한 모범 사례 및 문제 해결

데이터 추출 에이전트 사용을 위한 사전 조건

데이터 추출 에이전트를 사용하기 전에 Amazon Redshift에 필요한 권한을 대상으로 Amazon Redshift 사용자에게 추가합니다. 자세한 정보는 Amazon Redshift를 대상으로 사용할 수 있는 권한을 참조하세요.

그런 다음, Amazon S3 버킷 정보를 저장하고 SSL(Secure Sockets Layer) 트러스트 및 키 스토어를 설정합니다.

Amazon S3 설정

에이전트가 데이터를 추출한 후에는 Amazon S3 버킷에 업로드합니다. 계속하기 전에 AWS 계정 및 Amazon S3 버킷에 연결할 자격 증명을 제공해야 합니다. 자격 증명과 버킷 정보를 글로벌 애플리케이션 설정의 프로필에 저장한 다음 프로필을 AWS SCT 프로젝트에 연결합니다. 필요한 경우 전역 설정을 선택하여 새 프로필을 생성합니다. 자세한 정보는 AWS SCT에 AWS 서비스 프로필 저장을 참조하세요.

데이터를 대상 Amazon Redshift 데이터베이스로 마이그레이션하려면 AWS SCT 데이터 추출 에이전트가 사용자를 대신하여 Amazon S3 버킷에 액세스할 수 있는 권한이 필요합니다. 이 권한을 제공하려면 다음 정책에 따라 AWS Identity and Access Management (IAM) 사용자를 생성하십시오.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "s3:PutObject",
                "s3:DeleteObject",
                "s3:GetObject",
                "s3:GetObjectTagging",
                "s3:PutObjectTagging"
            ],
            "Resource": [
                "arn:aws:s3:::bucket_name/*",
                "arn:aws:s3:::bucket_name"
            ],
            "Effect": "Allow"
        },
        {
            "Action": [
                "s3:ListBucket",
                "s3:GetBucketLocation"
            ],
            "Resource": [
                "arn:aws:s3:::bucket_name"
            ],
            "Effect": "Allow"
        },
        {
            "Effect": "Allow",
            "Action": "s3:ListAllMyBuckets",
```

```
            "Resource": "*"
        },
        {
            "Action": [
                "iam:GetUser"
            ],
            "Resource": [
                "arn:aws:iam::111122223333:user/DataExtractionAgentName"
            ],
            "Effect": "Allow"
        }
    ]
}
```

이전 예제에서 *bucket_name*을 Amazon S3 버킷의 이름으로 바꿉니다. 그 다음, *111122223333:user/DataExtractionAgentName*을 IAM 사용자의 이름으로 바꿉니다.

IAM 역할 수임

보안을 강화하기 위해 AWS Identity and Access Management (IAM) 역할을 사용하여 Amazon S3 버킷에 액세스할 수 있습니다. 이를 위해 권한 없이 데이터 추출 에이전트의 IAM 사용자를 생성합니다. 그런 다음, Amazon S3 액세스를 지원하는 IAM 역할을 생성하고 이 역할을 수임할 수 있는 서비스 및 사용자 목록을 지정합니다. 자세한 내용은 IAM 사용 설명서에서 IAM 역할을 참조하세요.

Amazon S3 버킷에 액세스하도록 IAM 역할을 구성하려면

1. 새 IAM 사용자를 생성합니다. 사용자 보안 인증에 대해 프로그래밍 방식 액세스 유형을 선택합니다.
2. 데이터 추출 에이전트가 AWS SCT 제공하는 역할을 맡을 수 있도록 호스트 환경을 구성하십시오. 이전 단계에서 구성한 사용자가 데이터 추출 에이전트에서 보안 인증 공급자 체인을 사용할 수 있도록 지원하는지 확인합니다. 자세한 내용을 알아보려면 AWS SDK for Java 개발자 안내서의 보안 인증 사용을 참조하세요.
3. Amazon S3 버킷에 대한 액세스 권한이 있는 새 IAM 역할을 생성합니다.
4. 이전에 생성한 사용자가 역할을 맡을 수 있도록 하려면 이 역할의 트러스트 섹션을 수정합니다. 다음 예에서 *111122223333:user/DataExtractionAgentName*을 사용자의 이름으로 바꿉니다.

```
{
    "Effect": "Allow",
    "Principal": {
```

```
            "AWS": "arn:aws:iam::111122223333:user/DataExtractionAgentName"
        },
        "Action": "sts:AssumeRole"
}
```

5. redshift.amazonaws.com이 역할을 맡을 수 있도록 하려면 이 역할의 트러스트 섹션을 수정합니다.

```
{
    "Effect": "Allow",
    "Principal": {
        "Service": [
            "redshift.amazonaws.com"
        ]
    },
    "Action": "sts:AssumeRole"
}
```

6. Amazon Redshift 클러스터에 IAM 역할을 연결합니다.

이제 AWS SCT에서 데이터 추출 에이전트를 실행할 수 있습니다.

IAM 역할 수임을 사용하는 경우 데이터 마이그레이션이 다음과 같은 방식으로 작동합니다. 데이터 추출 에이전트는 보안 인증 공급자 체인을 사용하여 시작하고 사용자 보안 인증을 가져옵니다. 다음으로에서 AWS SCT데이터 마이그레이션 작업을 생성한 다음 데이터 추출 에이전트가 맡을 IAM 역할을 지정하고 작업을 시작합니다. AWS Security Token Service (AWS STS) 는 Amazon S3에 액세스하기 위한 임시 자격 증명을 생성합니다. 데이터 추출 에이전트는 이러한 보안 인증을 사용하여 Amazon S3에 데이터를 업로드합니다.

그런 다음 Amazon Redshift에 IAM 역할을 AWS SCT 제공합니다. 그러면 Amazon Redshift는 Amazon S3에 AWS STS 액세스하기 위한 새로운 임시 자격 증명을 받습니다. Amazon Redshift는 이러한 보안 인증을 사용하여 Amazon S3에서 Amazon Redshift 테이블로 데이터를 복사합니다.

보안 설정

AWS Schema Conversion Tool 및 추출 에이전트는 보안 소켓 계층 (SSL) 을 통해 통신할 수 있습니다. SSL을 활성화하려면 트러스트 스토어와 키 스토어를 설정합니다.

추출 에이전트와의 안전한 통신을 설정하려면

1. 를 시작합니다. AWS Schema Conversion Tool

2. 설정 메뉴를 열고 전역 설정을 선택합니다. 전역 설정 대화 상자가 나타납니다.
3. [Security]를 선택합니다.
4. Generate trust and key store를 선택하거나 Select existing trust store를 선택합니다.

   Generate trust and key store를 선택한 경우, 트러스트 및 키 스토어의 이름과 암호, 그리고 생성된 파일이 저장될 위치의 경로를 지정합니다. 이후 단계에서 이 파일을 사용합니다.

   Select existing trust store를 선택한 경우, 트러스트 및 키 스토어의 암호와 파일 이름을 지정합니다. 이후 단계에서 이 파일을 사용합니다.
5. 트러스트 스토어와 키 스토어를 지정한 후 확인을 선택하여 전역 설정 대화 상자를 닫습니다.

데이터 추출 에이전트의 환경 구성

단일 호스트에 여러 데이터 추출 에이전트를 설치할 수 있습니다. 그러나 하나의 호스트에서 하나의 데이터 추출 에이전트를 실행하는 것이 좋습니다.

데이터 추출 에이전트를 실행하려면 vCPU가 4개 이상 있고 메모리가 32GB 이상인 호스트를 사용해야 합니다. 또한 사용 가능한 최소 메모리를 4GB 이상으로 설정하십시오. AWS SCT 자세한 정보는 추가 메모리 구성을 참조하세요.

최적의 에이전트 호스트 구성과 수는 각 고객의 특정 상황에 따라 달라집니다. 마이그레이션할 데이터의 양, 네트워크 대역폭, 데이터 추출 시간 등의 요소를 고려해야 합니다. 먼저 PoC(개념 증명)를 수행한 후 이 PoC의 결과에 따라 데이터 추출 에이전트와 호스트를 구성할 수 있습니다.

추출 에이전트 설치

AWS Schema Conversion Tool를 실행 중인 컴퓨터와 별도로 개별 컴퓨터에 여러 추출 에이전트를 설치하는 것이 좋습니다.

추출 에이전트는 현재 다음 운영 체제에서 지원됩니다.

- Microsoft Windows
- Red Hat Enterprise Linux(RHEL) 6.0
- Ubuntu Linux(버전 14.04 이상)

다음 절차에 따라 추출 에이전트를 설치합니다. 추출 에이전트를 설치할 각 컴퓨터에서 이 절차를 반복해서 수행합니다.

추출 에이전트를 설치하려면

1. AWS SCT 설치 프로그램 파일을 아직 다운로드하지 않은 경우 의 지침에 따라 다운로드하십시오. 설치, 확인 및 업데이트 AWS SCT 설치 파일이 포함된.zip 파일에는 추출 에이전트 AWS SCT 설치 프로그램 파일도 들어 있습니다.
2. 최신 버전의 Amazon Corretto 11을 다운로드하여 설치합니다. 자세한 내용은 Amazon Corretto 11 사용 설명서의 Amazon Corretto 11 다운로드를 참조하세요.
3. 이름이 agents인 하위 폴더에서 추출 에이전트의 설치 관리자 파일을 찾습니다. 각 컴퓨터 운영 체제에서 추출 에이전트를 설치하기 위한 올바른 파일은 다음과 같습니다.

운영 체제	파일 이름
Microsoft Windows	`aws-schema-conversion-tool-extractor-2.0.1.`*`build-number`*`.msi`
RHEL	`aws-schema-conversion-tool-extractor-2.0.1.`*`build-number`*`.x86_64.rpm`
Ubuntu Linux	`aws-schema-conversion-tool-extractor-2.0.1.`*`build-number`*`.deb`

4. 설치 관리자 파일을 새 컴퓨터에 복사하여 추출 에이전트를 별도의 컴퓨터에 설치합니다.
5. 설치 관리자 파일을 실행합니다. 다음에 표시된 운영 체제별 지침을 사용하십시오.

운영 체제	설치 지침
Microsoft Windows	파일을 두 번 클릭하여 설치 프로그램을 실행합니다.

RHEL	파일을 다운로드 또는 이동한 폴더에서 다음 명령을 실행합니다.
	`sudo rpm -ivh aws-schema-conversion-tool-extractor-2.0.1.`*`build-number`*`.x86_64.rpm`
	`sudo ./sct-extractor-setup.sh --config`

Ubuntu Linux	파일을 다운로드 또는 이동한 폴더에서 다음 명령을 실행합니다. `sudo dpkg -i aws-schema-conversion-tool-extractor-2.0.1.`*`build-number`*`.deb`

운영 체제	설치 지침
	`sudo ./sct-extractor-setup.sh --config`

6. 다음을 선택하고 라이선스 계약에 동의한 후 다음을 선택합니다.
7. AWS SCT 데이터 추출 에이전트를 설치할 경로를 입력하고 다음을 선택합니다.
8. 설치를 선택하여 데이터 추출 에이전트를 설치합니다.

   AWS SCT 데이터 추출 에이전트를 설치합니다. 설치를 완료하려면 데이터 추출 에이전트를 구성하십시오. AWS SCT 구성 설정 프로그램을 자동으로 실행합니다. 자세한 정보는 추출 에이전트 구성을 참조하세요.
9. 데이터 추출 에이전트를 구성한 후 완료를 선택하여 설치 마법사를 닫습니다.

추출 에이전트 구성

다음 절차를 사용하여 추출 에이전트를 구성합니다. 추출 에이전트가 설치된 각 컴퓨터에서 이 절차를 반복합니다.

추출 에이전트를 구성하려면

1. 구성 설정 프로그램을 시작합니다.

   - Windows에서는 데이터 추출 에이전트를 설치하는 동안 구성 설정 프로그램을 자동으로 AWS SCT 시작합니다.

     필요한 경우 설정 프로그램을 수동으로 시작할 수 있습니다. 이렇게 하려면 Windows에서 `ConfigAgent.bat` 파일을 실행합니다. 에이전트를 설치한 폴더에서 이 파일을 찾을 수 있습니다.
   - RHEL 및 Ubuntu에서는 에이전트를 설치한 위치에서 `sct-extractor-setup.sh` 파일을 실행합니다.

   설정 프로그램에서 정보를 확인하는 프롬프트 메시지를 표시합니다. 각 프롬프트에는 기본값이 표시됩니다.
2. 각 프롬프트의 기본값을 그대로 사용하거나 새 값을 입력합니다.

   다음과 같은 정보를 지정합니다.

   - Listening port에 에이전트가 수신 대기하는 포트 번호를 입력합니다.

- Add a source vendor에 yes를 입력한 다음, 소스 데이터 웨어하우스 플랫폼을 입력합니다.
- JDBC 드라이버에 JDBC 드라이버를 설치한 위치를 입력합니다.
- 작업 폴더의 경우 AWS SCT 데이터 추출 에이전트가 추출된 데이터를 저장할 경로를 입력합니다. 작업 폴더는 에이전트와 다른 컴퓨터에 있을 수 있으며, 서로 다른 컴퓨터에 있는 여러 에이전트가 단일 작업 폴더를 공유할 수 있습니다.
- Enable SSL communication에 yes를 입력합니다.
- Key store에 키 스토어 파일의 위치를 입력합니다.
- Key store password에 키 스토어의 암호를 입력합니다.
- Enable client SSL authentication에 yes를 입력합니다.
- 트러스트 스토어에 트러스트 스토어 파일의 위치를 입력합니다.
- Trust store password에 트러스트 스토어의 암호를 입력합니다.

설치 프로그램이 추출 에이전트의 설정 파일을 업데이트합니다. 설정 파일은 이름이 `settings.properties`이며 추출 에이전트를 설치한 위치에 있습니다.

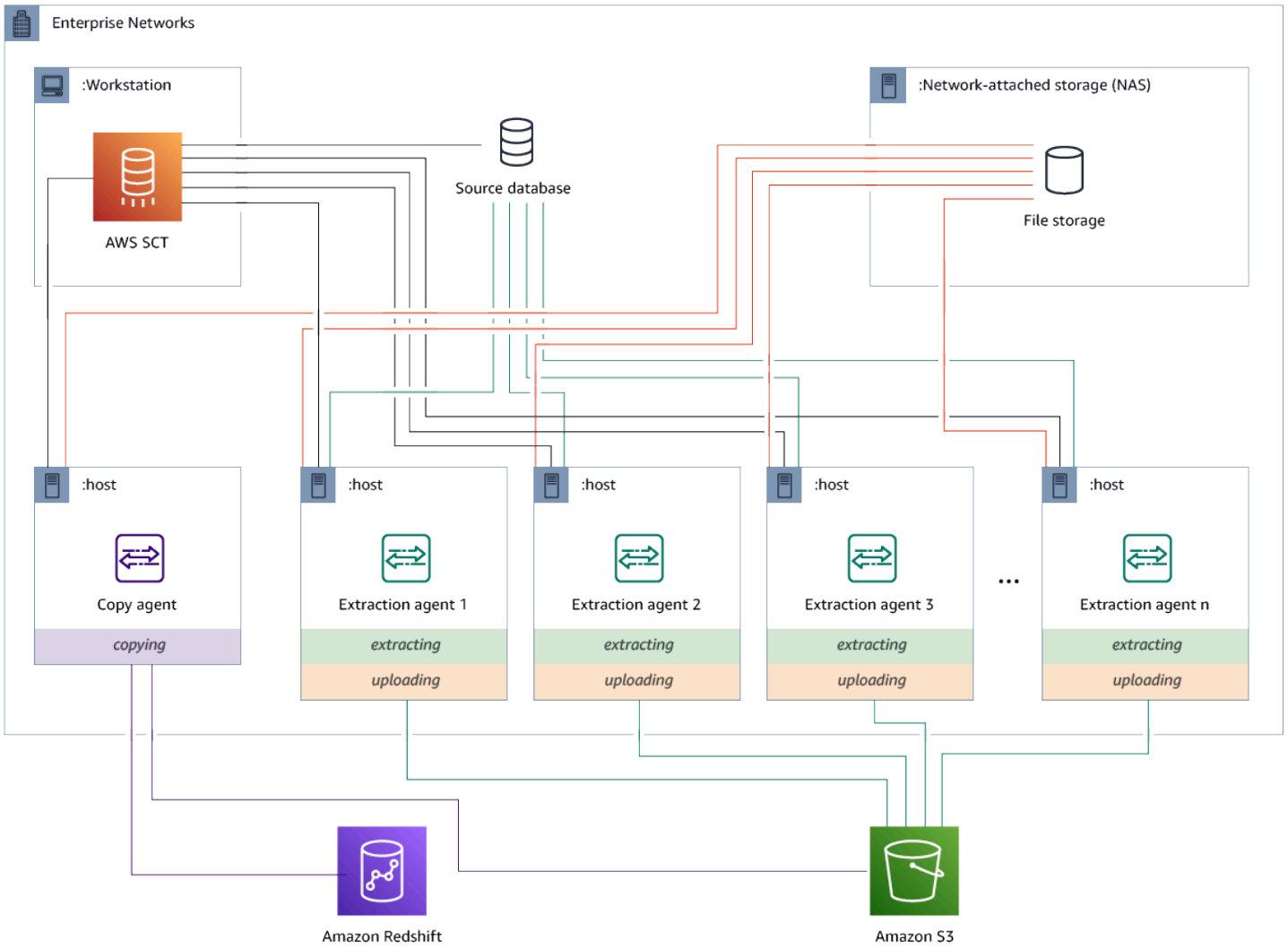
다음은 샘플 설정 파일입니다.

```
$ cat settings.properties
#extractor.start.fetch.size=20000
#extractor.out.file.size=10485760
#extractor.source.connection.pool.size=20
#extractor.source.connection.pool.min.evictable.idle.time.millis=30000
#extractor.extracting.thread.pool.size=10
vendor=TERADATA
driver.jars=/usr/share/lib/jdbc/terajdbc4.jar
port=8192
redshift.driver.jars=/usr/share/lib/jdbc/RedshiftJDBC42-1.2.43.1067.jar
working.folder=/data/sct
extractor.private.folder=/home/ubuntu
ssl.option=OFF
```

구성 설정을 변경하려면 텍스트 편집기를 사용하여 `settings.properties` 파일을 편집하거나 에이전트 구성을 다시 실행할 수 있습니다.

전용 복사 에이전트가 있는 추출 에이전트 설치 및 구성

공유 스토리지와 전용 복사 에이전트가 있는 구성에서 추출 에이전트를 설치할 수 있습니다. 다음은 이 시나리오를 설명하는 다이어그램입니다



이 구성은 소스 데이터베이스 서버가 최대 120개의 연결을 지원하고 네트워크에 충분한 스토리지가 연결된 경우 유용할 수 있습니다. 다음 절차를 사용하여 전용 복사 에이전트가 있는 추출 에이전트를 구성합니다.

추출 에이전트와 전용 복사 에이전트를 설치 및 구성하려면

1. 모든 추출 에이전트의 작업 디렉터리가 공유 스토리지의 동일한 폴더를 사용하는지 확인합니다.
2. 추출 에이전트 설치의 단계에 따라 추출기 에이전트를 설치합니다.
3. 추출 에이전트 구성의 단계에 따라 추출 에이전트를 구성하되 소스 JDBC 드라이버만 지정합니다.
4. 추출 에이전트 구성의 단계에 따라 전용 복사 에이전트를 구성하되 Amazon Redshift JDBC 드라이버만 지정합니다.

추출 에이전트 시작

다음 절차에 따라 추출 에이전트를 시작합니다. 추출 에이전트가 설치된 각 컴퓨터에서 이 절차를 반복합니다.

추출 에이전트는 리스너의 역할을 합니다. 이 절차를 사용하여 에이전트를 시작하면 해당 에이전트가 명령의 수신 대기를 시작합니다. 이후 섹션에서는 데이터 웨어하우스에서 데이터를 추출하는 명령을 에이전트로 보냅니다.

추출 에이전트를 시작하려면

- 추출 에이전트가 설치된 컴퓨터에서 운영 체제에 따라 아래에 나온 명령을 실행합니다.

운영 체제	시작 명령
Microsoft Windows	`StartAgent.bat` 배치 파일을 두 번 클릭합니다.
RHEL	에이전트를 설치한 폴더 경로에서 다음 명령을 실행합니다. `sudo initctl start sct-extractor`
Ubuntu Linux	에이전트를 설치한 폴더 경로에서 다음 명령을 실행합니다. 사용 중인 Ubuntu 버전에 적합한 명령을 사용합니다. Ubuntu 14.04: `sudo initctl start sct-extractor` Ubuntu 15.04 이상: `sudo systemctl start sct-extractor`

에이전트의 상태를 확인하려면 동일한 명령을 실행하고 `start`를 `status`로 바꿉니다.

에이전트를 중지하려면 동일한 명령을 실행하고 `start`를 `stop`으로 바꿉니다.

에 추출 에이전트 등록 AWS Schema Conversion Tool

를 사용하여 추출 에이전트를 관리합니다 AWS SCT. 추출 에이전트가 리스너의 역할을 합니다. 에서 AWS SCT지침을 받으면 데이터 웨어하우스에서 데이터를 추출합니다.

다음 절차를 사용하여 AWS SCT 프로젝트에 추출 에이전트를 등록하십시오.

추출 에이전트를 등록하려면

1. AWS Schema Conversion Tool를 시작하고 프로젝트를 엽니다.
2. 보기 메뉴를 열고 Data Migration view (other)를 선택합니다. 에이전트 탭이 나타납니다. 이전에 에이전트를 등록한 경우 AWS SCT 는 탭 상단의 그리드에 해당 에이전트를 표시합니다.
3. 등록(Register)을 선택합니다.

   에이전트를 AWS SCT 프로젝트에 등록한 후에는 동일한 에이전트를 다른 프로젝트에 등록할 수 없습니다. AWS SCT 프로젝트에서 에이전트를 더 이상 사용하지 않는 경우 등록을 취소할 수 있습니다. 그런 다음, 다른 프로젝트에 등록할 수 있습니다.
4. Redshift 데이터 에이전트를 선택하고 확인을 선택합니다.
5. 대화 상자의 연결 탭에 정보를 입력합니다.

   a. 설명에 에이전트의 설명을 입력합니다.

   b. 호스트 이름에 에이전트 컴퓨터의 호스트 이름 또는 IP 주소를 입력합니다.

   c. 포트에 에이전트가 수신 대기하는 포트 번호를 입력합니다.

   d. 등록을 선택하여 에이전트를 AWS SCT 프로젝트에 등록합니다.
6. AWS SCT 프로젝트에 여러 에이전트를 등록하려면 이전 단계를 반복합니다.

상담원의 정보 숨기기 및 복구하기 AWS SCT

AWS SCT 에이전트는 사용자 키 트러스트 저장소의 암호, 데이터베이스 계정, AWS 계정 정보 및 유사한 항목 등 상당한 양의 정보를 암호화합니다. `seed.dat`라는 특수 파일을 사용하여 이 작업을 수행합니다. 기본적으로 에이전트는 에이전트를 처음 구성한 사용자의 작업 폴더에 이 파일을 만듭니다.

여러 사용자가 에이전트를 구성하고 실행할 수 있으므로 `seed.dat`의 경로는 `settings.properties` 파일의 `{extractor.private.folder}` 파라미터에 저장됩니다. 에이전트가 시작되면 이 경로를 사용하여 에이전트가 작업을 수행하는 데이터베이스의 키 트러스트 스토어 정보에 액세스하는 데 사용할 `seed.dat` 파일을 찾을 수 있습니다.

다음과 같은 경우에는 에이전트가 저장한 암호를 복구해야 할 수 있습니다.

- 사용자가 `seed.dat` 파일을 분실했는데 AWS SCT 에이전트의 위치와 포트가 변경되지 않은 경우
- 사용자가 `seed.dat` 파일을 분실하고 AWS SCT 에이전트의 위치 및 포트가 변경된 경우 이 경우 일반적으로 에이전트가 다른 호스트 또는 포트로 마이그레이션되어 `seed.dat` 파일의 정보가 더 이상 유효하지 않기 때문에 변경이 발생합니다.

이러한 경우 에이전트를 SSL 없이 시작하면 에이전트가 시작된 후 이전에 만든 에이전트 스토리지에 액세스합니다. 그러면 Waiting for recovery 상태로 전환됩니다.

하지만 이런 경우에는 에이전트가 SSL를 사용하여 시작되어도 에이전트를 다시 시작할 수 없습니다. 이는 에이전트가 `settings.properties` 파일에 저장된 인증서의 암호를 해독할 수 없기 때문입니다. 이 유형의 시작에서는 에이전트가 시작되지 않습니다. 다음과 비슷한 오류가 로그에 기록됩니다. “SSL 모드를 활성화한 상태에서 에이전트를 시작할 수 없습니다. 에이전트를 다시 구성하십시오. 원인: 키 스토어 암호가 올바르지 않습니다.”

이 문제를 해결하려면 새 에이전트를 만들고 SSL 인증서에 액세스하는 데 기존 암호를 사용하도록 에이전트를 구성합니다. 이렇게 하려면 다음 절차를 사용하세요.

이 절차를 수행한 후에는 에이전트가 실행되어 복구 대기 상태로 전환됩니다. AWS SCT 복구 대기 중 상태인 에이전트에게 필요한 암호를 자동으로 보냅니다. 에이전트에 암호가 있으면 모든 작업이 다시 시작됩니다. AWS SCT 에서 추가적인 사용자 조치는 필요하지 않습니다.

에이전트를 재구성하고 SSL 인증서에 액세스하기 위한 암호를 복원하려면

1. 새 AWS SCT 에이전트를 설치하고 구성을 실행합니다.
2. 새 에이전트가 기존 에이전트 스토리지에서 작동하도록 하려면 `instance.properties` 파일의 `agent.name` 속성을 스토리지를 만들 때 사용한 에이전트의 이름으로 변경합니다.

   `instance.properties` 파일은 에이전트의 프라이빗 폴더에 저장되며, 이 폴더의 이름은 {*output.folder*}\dmt\{*hostName*}_{*portNumber*}\ 규칙에 따라 지정됩니다.
3. {*output.folder*}의 이름을 이전 에이전트의 출력 폴더 이름으로 변경합니다.

   이 시점에서도 여전히 이전 호스트와 포트의 기존 추출기에 액세스하려고 합니다. AWS SCT 따라서 액세스할 수 없는 추출기는 FAILED 상태가 됩니다. 그 다음, 호스트와 포트를 변경할 수 있습니다.
4. 요청 흐름을 새 에이전트로 리디렉션하도록 수정 명령을 사용하여 이전 에이전트의 호스트, 포트 또는 둘 모두 수정합니다.

새 에이전트에 ping을 보낼 AWS SCT 수 있는 경우 에이전트로부터 복구 대기 중 상태를 AWS SCT 받습니다. AWS SCT 그런 다음 에이전트의 암호를 자동으로 복구합니다.

에이전트 스토리지를 사용하는 각 에이전트는 {*output.folder*}\{*agentName*}\storage\에 있는 `storage.lck`라는 특수 파일을 업데이트합니다. 이 파일에는 에이전트의 네트워크 ID와 스토리지가 잠길 때까지의 시간이 포함되어 있습니다. 에이전트가 에이전트 스토리지로 작업할 경우

storage.lck 파일을 업데이트하고 스토리지 임대 기간을 5분마다 10분씩 연장합니다. 임대가 만료되기 전에는 다른 인스턴스가 이 에이전트 스토리지를 사용할 수 없습니다.

에서 데이터 마이그레이션 규칙 생성 AWS SCT

를 사용하여 데이터를 추출하기 전에 추출하는 데이터의 양을 줄이는 필터를 설정할 수 있습니다. AWS Schema Conversion ToolWHERE 절을 사용하여 데이터 마이그레이션 규칙을 만들면 추출하는 데이터를 줄일 수 있습니다. 예를 들어 단일 테이블에서 데이터를 선택하는 WHERE 절을 작성할 수 있습니다.

데이터 마이그레이션 규칙을 생성하고 필터를 프로젝트의 일부로 저장할 수 있습니다. 프로젝트를 연 상태에서 다음 절차를 사용하여 데이터 마이그레이션 규칙을 생성합니다.

데이터 마이그레이션 규칙을 만들려면

1. 보기 메뉴를 열고 Data Migration view (other)를 선택합니다.
2. Data migration rules를 선택한 다음, 새 규칙 추가를 선택합니다.
3. 데이터 마이그레이션 규칙을 구성합니다.
    a. 이름에 데이터 마이그레이션 규칙의 이름을 입력합니다.
    b. Where schema name is like에, 스키마에 적용할 필터를 입력합니다. 이 필터에서는 LIKE 절을 사용하여 WHERE 절을 평가합니다. 스키마 하나를 선택하려면 정확한 스키마 이름을 입력합니다. 여러 스키마를 선택하려면 “%” 문자를 와일드카드로 사용하여 스키마 이름에서 원하는 문자 수와 일치시킵니다.
    c. table name like에, 테이블에 적용할 필터를 입력합니다. 이 필터에서는 LIKE 절을 사용하여 WHERE 절을 평가합니다. 테이블 하나를 선택하려면 정확한 이름을 입력합니다. 여러 테이블을 선택하려면 “%” 문자를 와일드카드로 사용하여 테이블 이름에서 원하는 문자 수와 일치시킵니다.
    d. Where 절에 데이터를 필터링할 WHERE 절을 입력합니다.
4. 필터를 구성한 후 저장을 선택하여 필터를 저장하거나 취소를 선택하여 변경 내용을 취소합니다.
5. 필터 추가, 편집 및 삭제를 완료한 후 모두 저장을 선택하여 변경 내용을 모두 저장합니다.

필터를 삭제하지 않고 끄려면 토글 아이콘을 사용합니다. 기존 필터를 복제하려면 복사 아이콘을 사용합니다. 기존 필터를 삭제하려면 삭제 아이콘을 사용합니다. 필터 변경 내용을 저장하려면 모두 저장을 선택합니다.

프로젝트 설정에서 추출기 및 복사 설정 변경

의 프로젝트 설정 창에서 AWS SCT데이터 추출 에이전트 및 Amazon Redshift COPY 명령에 대한 설정을 선택할 수 있습니다.

이러한 설정을 선택하려면 설정, 프로젝트 설정을 선택한 다음, 데이터 마이그레이션을 선택합니다. 여기에서 추출 설정, Amazon S3 설정 및 복사 설정을 편집할 수 있습니다.

다음 테이블의 지침을 사용하여 추출 설정에 대한 정보를 제공합니다.

이 파라미터의 경우	조치
압축 형식	입력 파일의 압축 형식을 지정합니다. GZIP, BZIP2, ZSTD 또는 압축 없음 옵션 중 하나를 선택합니다.
구분 기호 문자	입력 파일의 필드를 구분하는 ASCII 문자를 지정합니다. 인쇄되지 않는 문자는 지원되지 않습니다.
NULL value as a string	데이터에 null 종결자가 포함된 경우 이 옵션을 켭니다. 이 옵션을 끄면 Amazon Redshift COPY 명령에서 null을 레코드의 끝으로 간주하고 로드 프로세스를 종료합니다.
Sorting strategy	정렬을 사용하여 실패 지점부터 추출을 다시 시작합니다. Use sorting after the first fail (recommended), Use sorting if possible 또는 Never use sorting 등의 정렬 전략 중 하나를 선택합니다. 자세한 정보는 the section called “데이터 정렬”을 참조하세요.
Source temp schema	추출 에이전트가 임시 객체를 생성할 수 있는 소스 데이터베이스의 스키마 이름을 입력합니다.
Out file size (in MB)	Amazon S3에 업로드된 파일의 크기를 MB 단위로 입력합니다.
Snowball out file size (in MB)	업로드한 파일의 크기 (MB) 를 AWS Snowball입력합니다. 파일의 크기는 1~1,000MB일 수 있습니다.
Use automatic partitioning. Greenplum과 Netezza의 경우 지원되는 테이블의 최소 크기를 MB 단위로 입력	테이블 파티셔닝을 사용하려면 이 옵션을 켠 다음, Greenplum 및 Netezza 소스 데이터베이스를 분할할 테이블 크기를 입력합니다. Oracle에서 Amazon Redshift로 마이그레이션하는 경우 파티션을

이 파라미터의 경우	조치
	나눈 모든 테이블에 대해 하위 작업이 AWS SCT 생성되므로 이 필드를 비워 둘 수 있습니다.
Extract LOBs	소스 데이터베이스에서 큰 객체(LOB)를 추출하려면 이 옵션을 켭니다. LOB에는 BLOB, CLOB, NCLOB, XML 파일 등이 포함됩니다. AWS SCT 추출 에이전트는 모든 LOB에 대해 데이터 파일을 생성합니다.
Amazon S3 bucket LOBs folder	AWS SCT 추출 에이전트가 LOB를 저장할 위치를 입력합니다.
Apply RTRIM to string columns	추출된 문자열 끝에서 지정된 문자 세트를 트리밍하려면 이 옵션을 켭니다.
Keep files locally after upload to Amazon S3	데이터 추출 에이전트가 Amazon S3에 파일을 업로드한 후 로컬 시스템에 파일을 유지하려면 이 옵션을 켭니다.

다음 테이블의 지침을 사용하여 Amazon S3 설정에 대한 정보를 제공합니다.

이 파라미터의 경우	조치
프록시 사용	프록시 서버를 사용하여 Amazon S3에 데이터를 업로드하려면 이 옵션을 켭니다. 그런 다음 데이터 전송 프로토콜을 선택하고 호스트 이름, 포트, 사용자 이름 및 암호를 입력합니다.
[엔드포인트 유형]	Federal Information Processing Standard(FIPS) 엔드포인트를 사용하려면 FIPS를 선택합니다. Virtual Private Cloud(VPC) 엔드포인트를 사용하려면 VPCE를 선택합니다. 그런 다음 VPC 엔드포인트에 VPC 엔드포인트의 도메인 이름 시스템(DNS)을 입력합니다.
Keep files on Amazon S3 after copying to Amazon Redshift	추출된 파일을 Amazon Redshift에 복사한 후 Amazon S3에 유지하려면 이 옵션을 켭니다.

다음 테이블의 지침을 사용하여 복사 설정에 대한 정보를 제공합니다.

이 파라미터의 경우	조치
Maximum error count	로드 오류 개수를 입력합니다. 작업이 이 한도에 도달하면 AWS SCT 데이터 추출 에이전트는 데이터 로드 프로세스를 종료합니다. 기본값은 0입니다. 즉, AWS SCT 데이터 추출 에이전트는 오류에 관계없이 데이터 로드를 계속합니다.
Replace not valid UTF-8 characters	유효하지 않은 UTF-8 문자를 지정된 문자로 바꾸고 데이터 로드 작업을 계속하려면 이 옵션을 켭니다.
Use blank as null value	공백 문자로 구성된 빈 필드를 null로 로드하려면 이 옵션을 켭니다.
Use empty as null value	비어 있는 CHAR 및 VARCHAR 필드를 null로 로드하려면 이 옵션을 켭니다.
Truncate columns	데이터 유형의 사양에 맞게 열의 데이터를 자르려면 이 옵션을 켭니다.
Automatic compression	복사 작업 중에 압축 인코딩을 적용하려면 이 옵션을 켭니다.
Automatic statistics refresh	복사 작업 종료 시 통계를 새로 고치려면 이 옵션을 켭니다.
Check file before load	Amazon Redshift에 데이터 파일을 로드하기 전에 해당 데이터 파일을 검증하려면 이 옵션을 켭니다.

마이그레이션하기 전에 다음을 사용하여 데이터를 정렬합니다. AWS SCT

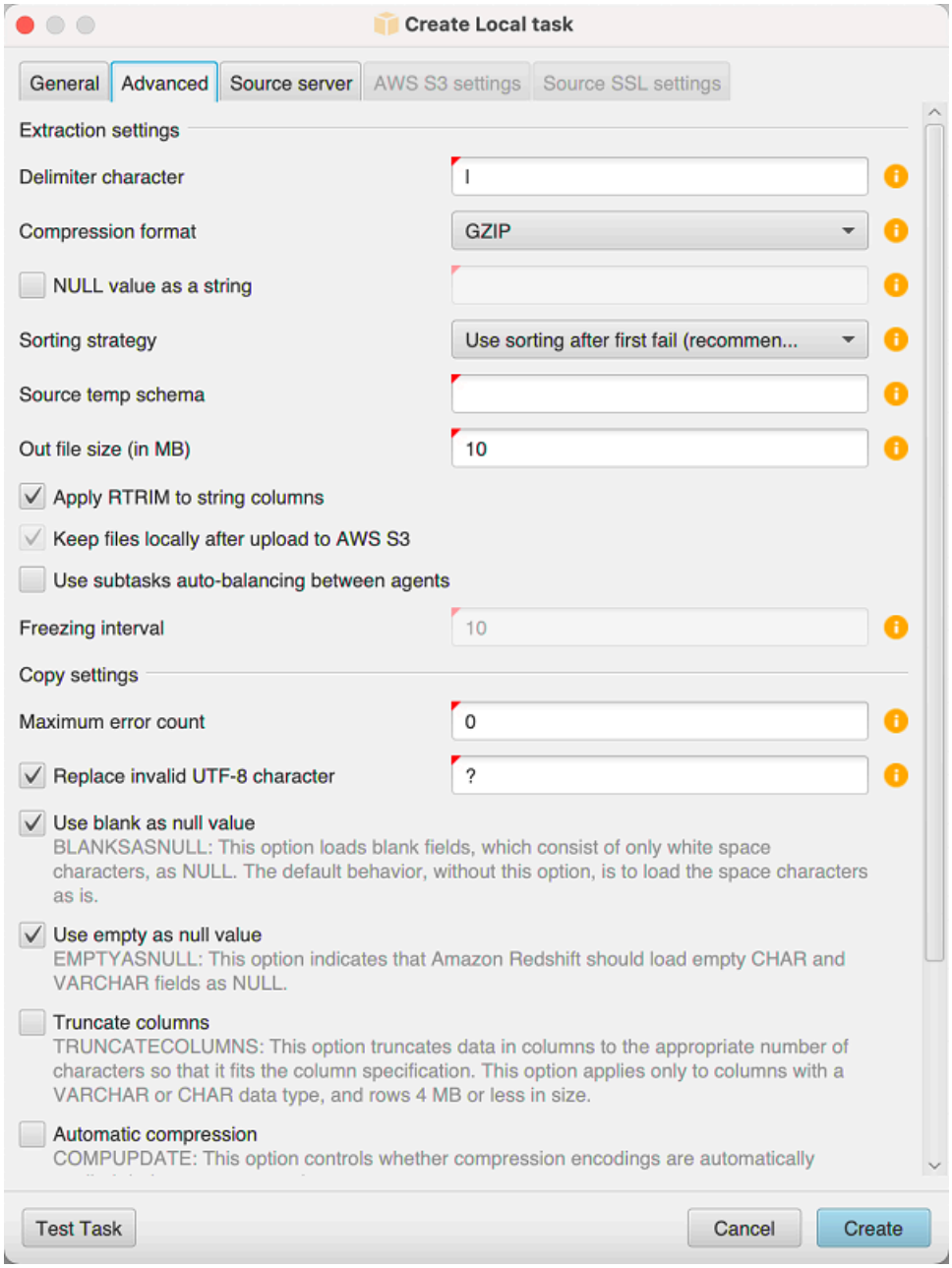
마이그레이션 전에 데이터를 정렬하면 몇 가지 AWS SCT 이점이 있습니다. 데이터를 먼저 정렬하면 실패 후 마지막으로 저장한 지점에서 추출 에이전트를 다시 시작할 AWS SCT 수 있습니다. 또한 데이터를 Amazon Redshift로 마이그레이션하고 데이터를 먼저 AWS SCT 정렬하는 경우 Amazon Redshift에 데이터를 더 빨리 삽입할 수 있습니다.

이러한 이점은 데이터 추출 쿼리를 AWS SCT 생성하는 방법과 관련이 있습니다. 경우에 따라 이러한 쿼리에 DENSE_RANK 분석 함수를 AWS SCT 사용합니다. 하지만 DENSE_RANK는 추출 결과 데이터세트를 정렬하는 데 많은 시간과 서버 리소스를 사용할 수 있으므로 이 데이터 세트 없이도 제대로 작동합니다. AWS SCT

마이그레이션하기 전에 다음을 사용하여 데이터를 정렬하려면 AWS SCT

1. AWS SCT 프로젝트를 엽니다.
2. 객체의 컨텍스트(마우스 오른쪽 버튼 클릭) 메뉴를 열고 Create Local task를 선택합니다.
3. 고급 탭을 선택하고 Sorting strategy에서 다음 옵션 중 하나를 선택합니다.

   - Never use sorting – 추출 에이전트가 DENSE_RANK 분석 함수를 사용하지 않으며 오류가 발생할 경우 처음부터 다시 시작합니다.
   - Use sorting if possible – 테이블에 프라이머리 키 또는 고유한 제약 조건이 있는 경우 추출 에이전트는 DENSE_RANK를 사용합니다.
   - Use sorting after first fail (recommended) - 추출 에이전트가 먼저 DENSE_RANK를 사용하지 않고 데이터를 가져오려고 합니다. 첫 번째 시도에 실패하면 추출 에이전트가 DENSE_RANK를 사용하여 쿼리를 다시 작성하고 실패 시 해당 위치를 보존합니다.



4. 아래에서 설명한 대로 추가 파라미터를 설정하고 생성을 선택하여 데이터 추출 작업을 생성합니다.

AWS SCT 데이터 추출 작업 생성, 실행, 모니터링

다음 절차를 사용하여 데이터 추출 작업을 작성, 실행 및 모니터링합니다.

에이전트에 작업을 할당하고 데이터를 마이그레이션하려면

1. 에서 AWS Schema Conversion Tool스키마를 변환한 후 프로젝트의 왼쪽 패널에서 하나 이상의 테이블을 선택합니다.

   모든 테이블을 선택할 수 있지만 성능상의 이유로 그렇게 하지 않는 것이 좋습니다. 데이터 웨어하우스의 테이블 크기에 따라 여러 테이블에 대해 여러 작업을 생성하는 것이 좋습니다.
2. 각 테이블에 대한 컨텍스트(마우스 오른쪽 버튼 클릭) 메뉴를 연 다음, 태스크 생성을 선택합니다. Create Local task 대화 상자가 열립니다.
3. 작업 이름에 해당 작업의 이름을 입력합니다.
4. Migration mode에서 다음 중 하나를 선택합니다.

   - Extract only - 데이터를 추출한 후 로컬 작업 폴더에 데이터를 저장합니다.
   - Extract and upload - 데이터를 추출한 후 Amazon S3에 데이터를 업로드합니다.
   - Extract, upload and copy - 데이터를 추출하고 Amazon S3에 데이터를 업로드한 다음, Amazon Redshift 데이터 웨어하우스로 복사합니다.
5. 암호화 유형에서 다음 중 하나를 선택합니다.

   - 없음 - 전체 데이터 마이그레이션 프로세스에서 데이터 암호화를 끕니다.
   - CSE_SK - 대칭 키를 사용한 클라이언트측 암호화를 통해 데이터를 마이그레이션합니다. AWS SCT 가 자동으로 암호화 키를 생성하고 Secure Sockets Layer(SSL)를 사용하여 이 키를 데이터 추출 에이전트에 전송합니다. 데이터 마이그레이션 중에는 AWS SCT 에서 큰 객체(LOB)를 암호화하지 않습니다.
6. 큰 객체를 추출하려면 Extract LOBs를 선택합니다. 큰 객체를 추출할 필요가 없으면 해당 확인란을 선택 취소할 수 있습니다. 그러면 추출하는 데이터의 양이 줄어듭니다.
7. 작업에 대한 세부 정보를 보려면 Enable task logging을 선택합니다. 작업 로그를 사용하여 문제를 디버깅할 수 있습니다.

   작업 로깅을 활성화하는 경우 보려는 세부 정보 수준을 선택합니다. 수준은 다음과 같으며, 각 수준에는 이전 수준의 모든 메시지가 포함됩니다.

   - ERROR – 가장 적은 양의 세부 정보입니다.
   - WARNING
   - INFO
   - DEBUG
   - TRACE – 가장 많은 양의 세부 정보입니다.

8. 에서 BigQuery 데이터를 내보내려면 Google Cloud Storage 버킷 폴더를 AWS SCT 사용합니다. 데이터 추출 에이전트가 이 폴더에 소스 데이터를 저장합니다.

   Google Cloud Storage 버킷 폴더의 경로를 입력하려면 고급을 선택합니다. Google CS bucket folder에 버킷 이름과 폴더 이름을 입력합니다.

9. 데이터 추출 에이전트 사용자 역할을 맡으려면 Amazon S3 설정을 선택합니다. IAM role에, 사용할 역할의 이름을 입력합니다. 지역의 경우 이 역할의 AWS 리전 를 선택합니다.

10. Test task를 선택하여 작업 폴더, Amazon S3 버킷 및 Amazon Redshift 데이터 웨어하우스에 연결할 수 있는지 확인합니다. 확인 방법은 선택한 마이그레이션 모드에 따라 달라집니다.

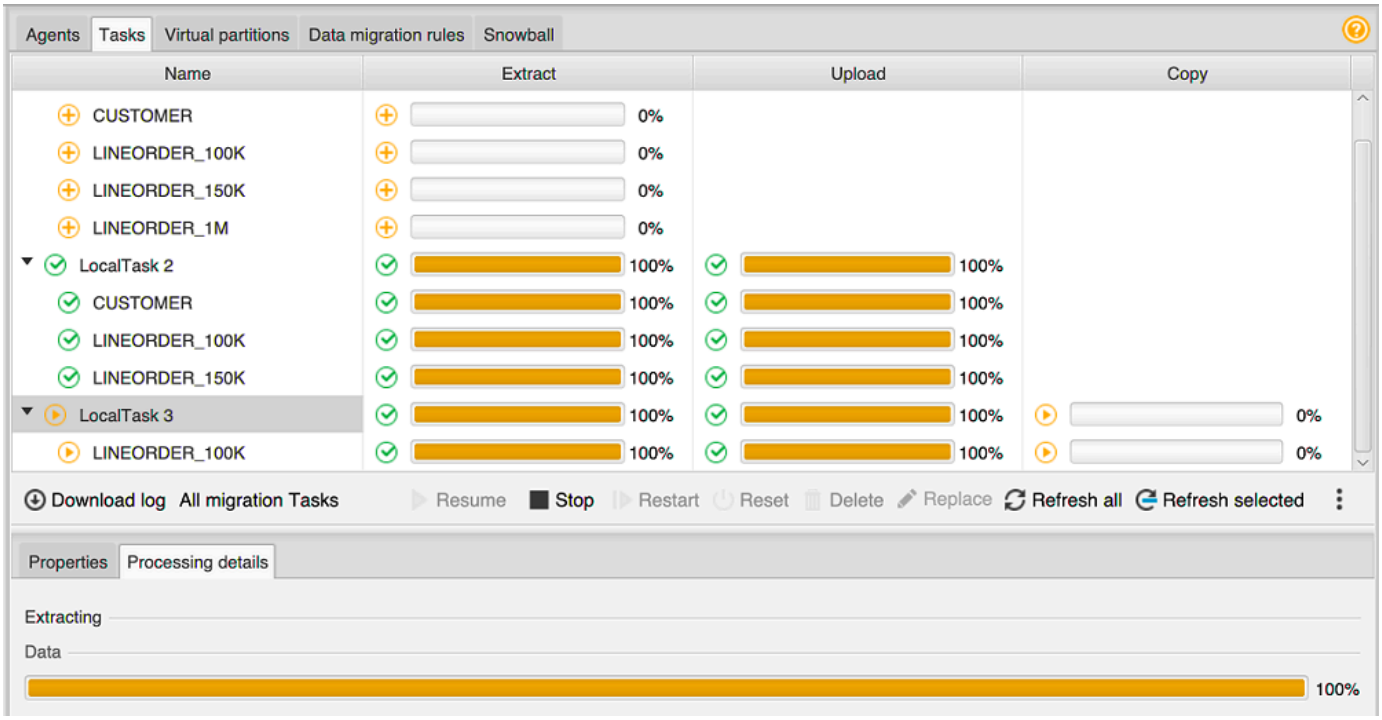
11. 생성을 선택하여 작업을 생성합니다.

12. 이전 단계를 반복하여 마이그레이션하려는 모든 데이터에 대한 작업을 생성합니다.

작업을 실행 및 모니터링하려면

1. 보기에서 Data Migration view를 선택합니다. 에이전트 탭이 나타납니다.

2. 작업 탭을 선택합니다. 작업이 다음과 같이 상단 그리드에 표시됩니다. 위쪽 그리드에서는 작업 상태를 확인하고 아래쪽 그리드에서는 하위 작업 상태를 확인할 수 있습니다.



3. 위쪽 그리드에서 작업을 선택하고 확장합니다. 선택한 마이그레이션 모드에 따라 작업이 추출, 업로드, 복사로 구분되어 표시됩니다.

4. 작업에 대해 시작을 선택하여 해당 작업을 시작합니다. 작업이 진행되는 동안 작업의 상태를 모니터링할 수 있습니다. 하위 작업은 병렬로 실행됩니다. 또한 추출, 업로드 및 복사도 병렬로 실행됩니다.
5. 작업을 설정할 때 로깅을 활성화한 경우에는 로그를 볼 수 있습니다.

    a. 로그 다운로드를 선택합니다. 로그 파일이 있는 폴더의 이름과 함께 메시지가 나타납니다. 메시지를 닫습니다.

    b. 태스크 세부 정보 탭에 링크가 표시됩니다. 링크를 선택하여 로그 파일이 있는 폴더를 엽니다.

종료할 수 AWS SCT있으며 상담원과 작업은 계속 실행됩니다. AWS SCT 나중에 다시 열어 작업 상태를 확인하고 작업 로그를 볼 수 있습니다.

데이터 추출 작업을 로컬 디스크에 저장한 후 내보내기와 가져오기를 사용하여 같은 프로젝트 또는 다른 프로젝트에 복원할 수 있습니다. 작업을 내보내려면 프로젝트에 추출 작업이 하나 이상 생성되어 있어야 합니다. 단일 추출 작업 또는 프로젝트에서 만든 모든 작업을 가져올 수 있습니다.

추출 작업을 내보내면 해당 작업에 대한 별도의 `.xml` 파일이 AWS SCT 생성됩니다. `.xml` 파일에는 작업의 메타데이터 정보(예: 작업 속성, 설명, 하위 작업 등)가 저장됩니다. `.xml` 파일에 추출 작업 처리에 대한 정보는 포함되지 않습니다. 작업을 가져오면 다음과 같은 정보가 다시 생성됩니다.

- 작업 진행 상황
- 하위 작업 및 스테이지 상태
- 하위 작업 및 스테이지별 추출 에이전트 분포
- 작업 및 하위 작업 ID
- [Task name]

AWS SCT 데이터 추출 작업 내보내기 및 가져오기

AWS SCT 내보내기와 가져오기를 사용하여 한 프로젝트의 기존 작업을 빠르게 저장하고 다른 프로젝트 (또는 같은 프로젝트) 에 복원할 수 있습니다. 데이터 추출 작업을 내보내고 가져오려면 다음 절차를 따릅니다.

데이터 추출 작업을 내보내고 가져오려면

1. 보기에서 Data Migration view를 선택합니다. 에이전트 탭이 나타납니다.
2. 작업 탭을 선택합니다. 표시되는 그리드에 작업이 나열됩니다.

3. 작업 목록 아래의 오른쪽 하단 모서리에 있는 세로로 정렬된 세 개의 점(줄임표 아이콘)을 선택합니다.
4. 팝업 메뉴에서 내보내기 작업을 선택합니다.
5. 작업 내보내기 `.xml` 파일을 AWS SCT 배치할 폴더를 선택합니다.

   AWS SCT 파일 이름 형식이 인 작업 내보내기 파일을 만듭니다*TASK-DESCRIPTION_TASK-ID*`.xml`.
6. 작업 목록 아래의 오른쪽 하단에 있는 세로로 정렬된 세 개의 점(줄임표 아이콘)을 선택합니다.
7. 팝업 메뉴에서 가져오기 작업을 선택합니다.

   소스 데이터베이스에 연결된 프로젝트로 추출 작업을 가져올 수 있으며, 프로젝트에는 등록된 활성 추출 에이전트가 하나 이상 있습니다.
8. 내보낸 추출 작업에 사용할 `.xml` 파일을 선택합니다.

   AWS SCT 파일에서 추출 작업의 매개변수를 가져오고, 작업을 만들고, 추출 에이전트에 작업을 추가합니다.
9. 추가로 데이터 추출 작업을 내보내고 가져오려면 이 단계를 반복합니다.

이 프로세스가 끝나면 내보내기와 가져오기가 완료되고 데이터 추출 작업을 사용할 준비가 됩니다.

AWS Snowball Edge 장치를 사용한 데이터 추출

AWS SCT AWS Snowball Edge를 사용하는 프로세스에는 여러 단계가 있습니다. 마이그레이션에는 데이터 추출 에이전트를 AWS SCT 사용하여 에지 디바이스로 데이터를 이동하는 로컬 작업과, AWS Snowball Edge 디바이스에서 Amazon S3 버킷으로 데이터를 AWS 복사하는 중간 작업이 포함됩니다. AWS Snowball 프로세스는 Amazon S3 버킷에서 Amazon Redshift로 데이터 AWS SCT 로드를 완료합니다.

이 개요 다음 섹션에서는 이러한 각 step-by-step 작업에 대한 가이드를 제공합니다. 이 절차에서는 사용자가 전용 컴퓨터에 데이터 추출 에이전트를 AWS SCT 설치하고 구성하고 등록했다고 가정합니다.

AWS Snowball Edge를 사용하여 로컬 데이터 저장소의 데이터를 데이터 저장소로 마이그레이션하려면 다음 단계를 수행하십시오. AWS

1. AWS Snowball 콘솔을 사용하여 AWS Snowball Edge 작업을 생성합니다.
2. 로컬 전용 Linux 시스템을 사용하여 AWS Snowball Edge 디바이스의 잠금을 해제하십시오.
3. 에서 새 프로젝트를 생성합니다 AWS SCT.

4. 데이터 추출 에이전트를 설치 및 구성합니다.
5. Amazon S3 버킷에 사용할 권한을 생성하고 설정합니다.
6. AWS SCT 프로젝트로 AWS Snowball 작업을 가져옵니다.
7. AWS SCT에 데이터 추출 에이전트를 등록합니다.
8. 에서 로컬 작업을 생성합니다 AWS SCT.
9. AWS SCT에서 데이터 마이그레이션 작업을 실행하고 모니터링합니다.

AWS SCT 및 tep-by-step AWS Snowball Edge를 사용하여 데이터를 마이그레이션하는 S 절차

다음 섹션에서는 마이그레이션 단계에 대한 자세한 정보를 제공합니다.

1단계: AWS Snowball 에지 작업 생성

AWS Snowball Edge 개발자 안내서의 AWS Snowball Edge Job 생성 섹션에 설명된 단계에 따라 AWS Snowball 작업을 생성합니다.

2단계: AWS Snowball Edge 장치 잠금 해제

에이전트를 설치한 시스템에서 Snowball Edge 디바이스의 잠금을 해제하고 자격 증명을 제공하는 명령을 실행합니다. AWS DMS 이 명령을 실행하면 AWS DMS 에이전트 호출이 AWS Snowball Edge 디바이스에 연결되었는지 확인할 수 있습니다. AWS Snowball Edge 디바이스 잠금 해제에 대한 자세한 내용은 Snowball Edge 잠금 해제를 참조하십시오.

```
aws s3 ls s3://<bucket-name> --profile <Snowball Edge profile> --endpoint http://
<Snowball IP>:8080 --recursive
```

3단계: 새 프로젝트 생성 AWS SCT

다음으로 새 AWS SCT 프로젝트를 생성합니다.

에서 새 프로젝트를 만들려면 AWS SCT

1. 시작 AWS Schema Conversion Tool. 파일 메뉴에서 새 프로젝트를 선택합니다. 새 프로젝트 대화 상자가 나타납니다.
2. 컴퓨터에서 로컬로 저장되는 프로젝트의 이름을 입력합니다.
3. 로컬 프로젝트 파일의 위치를 입력합니다.

4. OK를 선택하여 AWS SCT 프로젝트를 생성합니다.
5. 소스 추가를 선택하여 AWS SCT 프로젝트에 새 소스 데이터베이스를 추가합니다.
6. Add target (대상 추가) 를 선택하여 AWS SCT 프로젝트에 새 대상 플랫폼을 추가합니다.
7. 왼쪽 패널에서 소스 데이터베이스 스키마를 선택합니다.
8. 오른쪽 패널에서 선택한 소스 스키마의 대상 데이터베이스 플랫폼을 지정합니다.
9. 매핑 생성을 선택합니다. 이 버튼은 소스 데이터베이스 스키마와 대상 데이터베이스 플랫폼을 선택한 후에 활성화됩니다.

4단계: 데이터 추출 에이전트 설치 및 구성

AWS SCT 데이터 추출 에이전트를 사용하여 데이터를 Amazon Redshift로 마이그레이션합니다. 설치하기 위해 다운로드한.zip 파일에는 추출 에이전트 설치 AWS SCT프로그램 파일이 포함되어 있습니다. Windows, Red Hat Enterprise Linux 또는 Ubuntu에 데이터 추출 에이전트를 설치할 수 있습니다. 자세한 정보는 추출 에이전트 설치을 참조하세요.

데이터 추출 에이전트를 구성하려면 소스 및 대상 데이터베이스 엔진을 입력합니다. 또한 데이터 추출 에이전트를 실행하는 컴퓨터에 소스 및 대상 데이터베이스용 JDBC 드라이버를 다운로드했는지 확인합니다. 데이터 추출 에이전트는 이 드라이버를 사용하여 소스 및 대상 데이터베이스에 연결합니다. 자세한 정보는 필수 데이터베이스 드라이버 다운로드을 참조하세요.

Windows에서는 데이터 추출 에이전트 설치 관리자가 명령 프롬프트 창에서 구성 마법사를 시작합니다. Linux에서는 에이전트를 설치한 위치에서 `sct-extractor-setup.sh` 파일을 실행합니다.

5단계: Amazon S3 버킷에 AWS SCT 액세스하도록 구성

Amazon S3 버킷 구성에 대한 자세한 내용은 Amazon Simple Storage Service 사용 설명서의 버킷 개요를 참조하세요.

6단계: AWS SCT 프로젝트로 AWS Snowball 작업 가져오기

AWS SCT 프로젝트를 AWS Snowball Edge 기기와 연결하려면 AWS Snowball 작업을 가져오세요.

AWS Snowball 작업을 가져오려면

1. 설정 메뉴를 열고 전역 설정을 선택합니다. 전역 설정 대화 상자가 나타납니다.
2. AWS 서비스 프로필을 선택한 다음, 작업 가져오기를 선택합니다.
3. AWS Snowball 직업을 선택하세요.

4. AWS Snowball IP를 입력합니다. 자세한 내용은 AWS Snowball 사용 설명서에서 IP 주소 변경을 참조하세요.
5. AWS Snowball 포트를 입력하세요. 자세한 내용은 AWS SnowballAWS Snowball Edge 개발자 가이드의 에지 디바이스에서 AWS 서비스를 사용하는 데 필요한 포트를 참조하십시오.
6. AWS Snowball 액세스 키와 AWS Snowball 비밀 키를 입력합니다. 자세한 내용은 AWS Snowball 사용 설명서의 AWS Snowball에서 권한 부여 및 액세스 제어를 참조하세요.
7. 적용을 선택하고 확인을 선택합니다.

7단계: 데이터 추출 에이전트 등록 AWS SCT

이 섹션에서는 데이터 추출 에이전트를 AWS SCT에 등록합니다.

데이터 추출 에이전트를 등록하려면

1. 보기 메뉴에서 Data migration view (other)를 선택한 다음, 등록을 선택합니다.
2. 설명에 데이터 추출 에이전트의 이름을 입력합니다.
3. 호스트 이름에 데이터 추출 에이전트를 실행하는 컴퓨터의 IP 주소를 입력합니다.
4. 포트에는 구성된 수신 포트를 입력합니다.
5. 등록(Register)을 선택합니다.

8단계: 로컬 작업 생성

다음으로, 마이그레이션 작업을 생성합니다. 이 작업에는 하위 작업 두 개가 포함됩니다. 하나의 하위 작업은 원본 데이터베이스의 데이터를 AWS Snowball Edge 어플라이언스로 마이그레이션합니다. 또 다른 하위 작업은 어플라이언스에서 Amazon S3 버킷으로 로드하는 데이터를 가져와서 이를 대상 데이터베이스로 마이그레이션합니다.

마이그레이션 작업을 생성하려면

1. 보기 메뉴에서 Data migration view (other)를 선택합니다.
2. 소스 데이터베이스의 스키마를 표시하는 왼쪽 패널에서 마이그레이션할 스키마 객체를 선택합니다. 객체의 컨텍스트(마우스 오른쪽 버튼 클릭) 메뉴를 열고 Create local task를 선택합니다.
3. 작업 이름에 데이터 마이그레이션 작업을 설명하는 이름을 입력합니다.
4. Migration mode에서 Extract, upload, and copy를 선택합니다.
5. Amazon S3 settings를 선택합니다.

6. Use Snowball을 선택합니다.
7. 데이터 추출 에이전트가 데이터를 저장할 수 있는 Amazon S3 버킷의 폴더와 하위 폴더를 입력합니다.
8. 생성을 선택하여 작업을 생성합니다.

9단계: 에서 데이터 마이그레이션 작업 실행 및 모니터링 AWS SCT

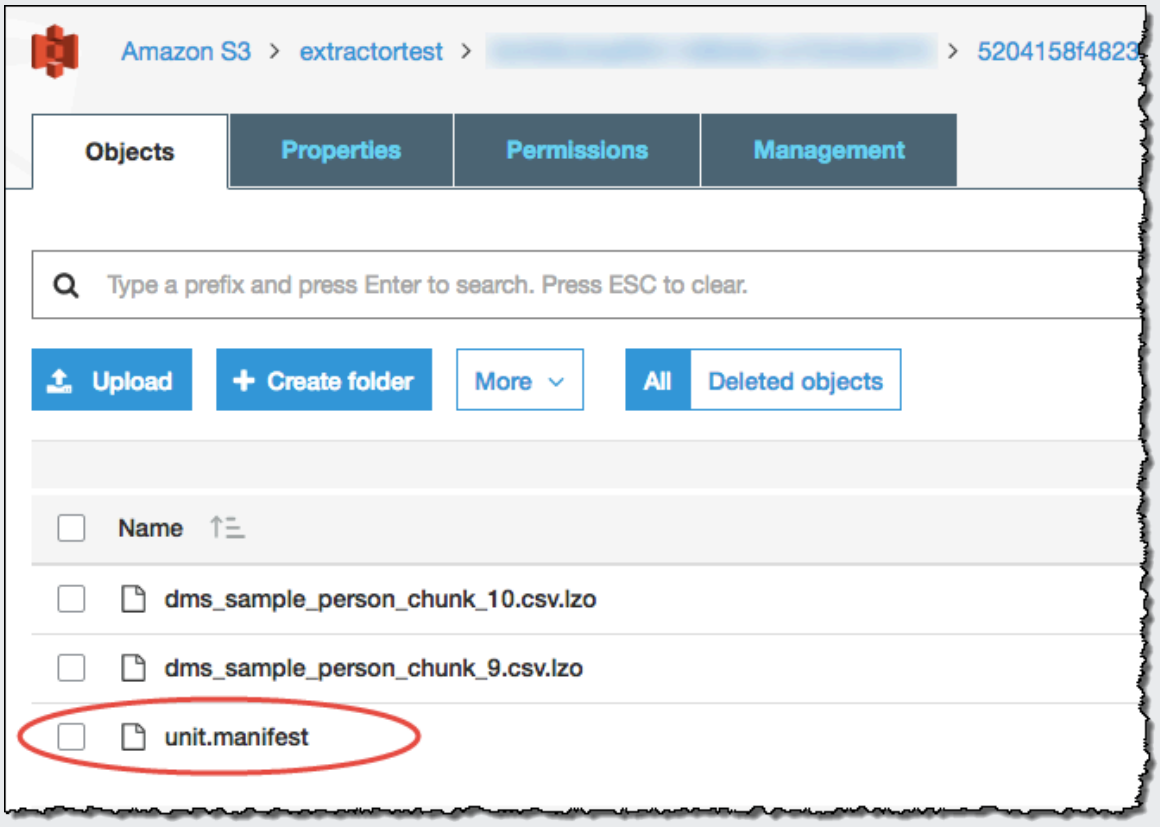
데이터 마이그레이션 작업을 시작하려면 시작을 선택합니다. 원본 데이터베이스, Amazon S3 버킷, AWS Snowball 디바이스 및 대상 데이터베이스에 대한 연결을 설정했는지 확인하십시오 AWS.

작업 탭에서 데이터 마이그레이션 작업과 해당 하위 작업을 모니터링하고 관리할 수 있습니다. 데이터 마이그레이션 진행 상황을 확인하고 데이터 마이그레이션 작업을 일시 중지하거나 다시 시작할 수 있습니다.

데이터 추출 작업 출력

마이그레이션 작업이 완료되면 데이터가 준비됩니다. 선택한 마이그레이션 모드 및 데이터 위치에 따라 작업 진행 방법을 결정하려면 다음 정보를 사용합니다.

마이그레이션 모드	데이터 위치
추출, 업로드 및 복사	데이터는 Amazon Redshift 데이터 웨어하우스에 이미 있습니다. 데이터가 있는지 확인한 후 사용할 수 있습니다. 자세한 내용은 클라이언트 도구 및 코드에서 클러스터에 연결을 참조하세요.
추출 및 업로드	추출 에이전트가 데이터를 Amazon S3 버킷에 파일로 저장했습니다. Amazon Redshift COPY 명령을 사용하여 Amazon Redshift에 데이터를 로드할 수 있습니다. 자세한 내용은 Amazon Redshift 설명서의 Amazon S3에서 데이터 로드를 참조하세요. Amazon S3 버킷에는 설정된 추출 작업에 해당하는 여러 폴더가 있습니다. Amazon Redshift로 데이터를 로드할 때 각 작업에서 생성한 매니페스트 파일의 이름을 지정합니다. 매니페스트 파일은 다음과 같이 Amazon S3 버킷의 작업 폴더에 나타납니다.

마이그레이션 모드	데이터 위치
	
추출만	추출 에이전트가 데이터를 작업 폴더에 파일로 저장했습니다. Amazon S3 버킷에 데이터를 수동으로 복사한 다음, 추출 및 업로드를 위한 명령을 사용하여 작업을 진행합니다.

가상 파티셔닝 사용: AWS Schema Conversion Tool

필터링 규칙을 사용하여 테이블 데이터의 가상 파티션을 생성하는 하위 작업을 만들면 분할되지 않은 큰 테이블을 최적으로 관리할 수 있습니다. AWS SCT에서는 마이그레이션된 데이터를 위한 가상 파티션을 만들 수 있습니다. 특정 데이터 유형에 사용할 수 있는 세 가지 파티션 유형이 있습니다.

- RANGE 파티션 유형은 숫자 및 날짜/시간 데이터 유형에서 작동합니다.
- LIST 파티션 유형은 숫자, 문자 및 날짜/시간 데이터 유형에서 작동합니다.
- DATE AUTO SPLIT 파티션 유형은 숫자, 날짜 및 시간 데이터 유형에서 작동합니다.

AWS SCT 파티션 생성 시 제공한 값을 검증합니다. 예를 들어 NUMERIC 데이터 유형으로 열을 분할하려고 하는데 다른 데이터 유형의 값을 제공하면 AWS SCT 오류가 발생합니다.

또한 Amazon Redshift로 데이터를 마이그레이션하는 데 사용하는 AWS SCT 경우, 네이티브 파티셔닝을 사용하여 대형 테이블의 마이그레이션을 관리할 수 있습니다. 자세한 정보는 네이티브 파티셔닝 사용을 참조하세요.

가상 파티셔닝 생성 시 제한

가상 파티션을 만들 때는 다음과 같은 제한 사항이 있습니다.

- 분할되지 않은 테이블에만 가상 파티셔닝을 사용할 수 있습니다.
- 데이터 마이그레이션 보기에서만 가상 파티셔닝을 사용할 수 있습니다.
- UNION ALL VIEW 옵션은 가상 파티셔닝에서 사용할 수 없습니다.

RANGE 파티션 유형

RANGE 파티션 유형은 숫자 및 날짜/시간 데이터 유형의 열 값 범위를 기반으로 데이터를 분할합니다. 이 파티션 유형은 WHERE 절을 생성하며 각 파티션의 값 범위를 제공합니다. 분할된 열의 값 목록을 지정하려면 값 상자를 사용합니다. .csv 파일을 사용하여 값 정보를 로드할 수 있습니다.

RANGE 파티션 유형은 파티션 값의 양쪽 끝에서 기본 파티션을 만듭니다. 이러한 기본 파티션은 지정된 파티션 값보다 작거나 큰 데이터를 모두 포착합니다.

예를 들어, 제공된 값 범위를 기반으로 여러 개의 파티션을 만들 수 있습니다. 다음 예에서는 LO_TAX의 파티셔닝 값을 지정하여 여러 파티션을 생성합니다.

```
Partition1: WHERE LO_TAX <= 10000.9
Partition2: WHERE LO_TAX > 10000.9 AND LO_TAX <= 15005.5
Partition3: WHERE LO_TAX > 15005.5 AND LO_TAX <= 25005.95
```

RANGE 가상 파티션을 만들려면

1. 열기. AWS SCT
2. Data Migration view (other) 모드를 선택합니다.
3. 가상 파티셔닝을 설정할 테이블을 선택합니다. 테이블에 대한 컨텍스트(마우스 오른쪽 버튼 클릭) 메뉴를 열고 Add virtual partitioning을 선택합니다.

4. Add virtual partitioning 대화 상자에 다음과 같이 정보를 입력합니다.

옵션	작업
파티션 유형	RANGE를 선택합니다. 선택한 유형에 따라 대화 상자 UI가 달라집니다.
열 이름	분할하려는 열을 선택합니다.
열 유형	열의 값에 대한 데이터 유형을 선택합니다.
값	새 값 상자에 각 값을 입력한 다음, 더하기 기호를 선택하여 값을 추가하면 새 값을 추가할 수 있습니다.
파일에서 로드	(선택 사항) 파티션 값이 포함된 .csv 파일의 이름을 입력합니다.

5. 확인을 선택합니다.

LIST 파티션 유형

LIST 파티션 유형은 숫자, 문자, 날짜/시간 데이터 형식의 열 값을 기반으로 데이터를 분할합니다. 이 파티션 유형은 WHERE 절을 생성하며 각 파티션의 값을 제공합니다. 분할된 열의 값 목록을 지정하려면 값 상자를 사용합니다. .csv 파일을 사용하여 값 정보를 로드할 수 있습니다.

예를 들어, 제공된 값을 기반으로 여러 개의 파티션을 만들 수 있습니다. 다음 예에서는 LO_ORDERKEY의 파티셔닝 값을 지정하여 여러 파티션을 생성합니다.

```
Partition1: WHERE LO_ORDERKEY = 1
Partition2: WHERE LO_ORDERKEY = 2
Partition3: WHERE LO_ORDERKEY = 3
…
PartitionN: WHERE LO_ORDERKEY = USER_VALUE_N
```

또한 지정된 파티션에 포함되지 않은 값에 대한 기본 파티션을 생성할 수도 있습니다.

마이그레이션에서 특정 값을 제외하려는 경우 LIST 파티션 유형을 사용하여 소스 데이터를 필터링할 수 있습니다. 예를 들어, LO_ORDERKEY = 4를 사용하여 행을 생략하려 한다고 가정합니다. 이 경우 파티션 값 목록에 4 값이 포함되지 않도록 하고 Include other values를 선택하지 않아야 합니다.

LIST 가상 파티션을 만들려면

1. 열기 AWS SCT.
2. Data Migration view (other) 모드를 선택합니다.
3. 가상 파티셔닝을 설정할 테이블을 선택합니다. 테이블에 대한 컨텍스트(마우스 오른쪽 버튼 클릭) 메뉴를 열고 Add virtual partitioning을 선택합니다.
4. Add virtual partitioning 대화 상자에 다음과 같이 정보를 입력합니다.

옵션	작업
파티션 유형	LIST를 선택합니다. 선택한 유형에 따라 대화 상자 UI가 달라집니다.
열 이름	분할하려는 열을 선택합니다.
새 값	파티셔닝 값 세트에 값을 추가하려면 여기에 해당 값을 입력합니다.
Include other values	파티셔닝 기준에 맞지 않는 모든 값이 저장되는 기본 파티션을 만들려면 이 옵션을 선택합니다.
파일에서 로드	(선택 사항) 파티션 값이 포함된 .csv 파일의 이름을 입력합니다.

5. 확인을 선택합니다.

DATE AUTO SPLIT 파티션 유형

DATE AUTO SPLIT 파티션 유형은 RANGE 파티션을 자동으로 생성하는 방법입니다. 데이터 자동 분할을 사용하면 파티셔닝 속성, 시작 위치 및 종료 위치, 값 사이의 범위 크기를 지정할 수 있습니다. AWS SCT 그런 다음 AWS SCT 에서 파티션 값을 자동으로 계산합니다.

DATA AUTO SPLIT은 범위 파티션 생성과 관련된 많은 작업을 자동화합니다. 이 방법을 사용하는 것과 범위 파티셔닝 사이의 단점은 파티션 경계에 대해 얼마나 많은 제어가 필요한가 하는 것입니다. 자동 분할 프로세스는 항상 동일한 크기(균일한)의 범위를 생성합니다. 범위 파티셔닝을 사용하면 특정 데이터 분포에 필요한 경우 각 범위의 크기를 변경할 수 있습니다. 예를 들어, 일별, 주별, 격주, 월별 등을 사용할 수 있습니다.

```
Partition1: WHERE LO_ORDERDATE >= ‘1954-10-10’ AND LO_ORDERDATE < ‘1954-10-24’
Partition2: WHERE LO_ORDERDATE >= ‘1954-10-24’ AND LO_ORDERDATE < ‘1954-11-06’
Partition3: WHERE LO_ORDERDATE >= ‘1954-11-06’ AND LO_ORDERDATE < ‘1954-11-20’
```

```
…
PartitionN: WHERE LO_ORDERDATE >= USER_VALUE_N AND LO_ORDERDATE <= ‘2017-08-13’
```

DATE AUTO SPLIT 가상 파티션을 만들려면

1. 열기 AWS SCT.
2. Data Migration view (other) 모드를 선택합니다.
3. 가상 파티셔닝을 설정할 테이블을 선택합니다. 테이블에 대한 컨텍스트(마우스 오른쪽 버튼 클릭) 메뉴를 열고 Add virtual partitioning을 선택합니다.
4. Add virtual partitioning 대화 상자에 다음과 같이 정보를 입력합니다.

옵션	작업
파티션 유형	DATE AUTO SPLIT을 선택합니다. 선택한 유형에 따라 대화 상자 UI가 달라집니다.
열 이름	분할하려는 열을 선택합니다.
시작일	시작 날짜를 입력합니다.
종료일	종료 날짜를 입력합니다.
간격	간격 단위를 입력하고 해당 단위의 값을 선택합니다.

5. 확인을 선택합니다.

네이티브 파티셔닝 사용

데이터 마이그레이션 속도를 높이기 위해 데이터 추출 에이전트는 원본 데이터 웨어하우스 서버의 기본 테이블 파티션을 사용할 수 있습니다. AWS SCT 그린플럼, 네테자, 오라클에서 Amazon Redshift로의 마이그레이션을 위한 네이티브 파티셔닝을 지원합니다.

예를 들어, 프로젝트를 생성한 후 스키마에 대한 통계를 수집하고 마이그레이션을 위해 선택한 테이블의 크기를 분석할 수 있습니다. 지정된 크기를 초과하는 테이블의 경우 기본 파티셔닝 메커니즘을 트리거합니다. AWS SCT

네이티브 파티셔닝을 사용하려면

1. 파일을 열고 AWS SCT새 프로젝트를 선택합니다. 새 프로젝트 대화 상자가 나타납니다.

2. 새 프로젝트를 만들고, 소스 및 대상 서버를 추가하고, 매핑 규칙을 생성합니다. 자세한 정보는 AWS SCT 프로젝트 생성을 참조하세요.
3. 보기를 선택하고 Main view를 선택합니다.
4. 프로젝트 설정에서 데이터 마이그레이션 탭을 선택합니다. Use automatic partitioning을 선택합니다. Greenplum 및 Netezza 소스 데이터베이스의 경우 지원되는 테이블의 최소 크기를 MB 단위로 입력합니다(예: 100). AWS SCT 가 비어 있지 않은 각 네이티브 파티션에 대해 개별 마이그레이션 하위 작업을 자동으로 생성합니다. Oracle에서 Amazon Redshift로 마이그레이션하는 경우 파티션을 나눈 모든 테이블에 대해 하위 작업을 AWS SCT 생성합니다.
5. 소스 데이터베이스의 스키마를 표시하는 왼쪽 패널에서 스키마를 선택합니다. 객체의 컨텍스트(마우스 오른쪽 버튼 클릭) 메뉴를 열고 Collect statistics를 선택합니다. Oracle에서 Amazon Redshift로 데이터를 마이그레이션하는 경우 이 단계를 건너뛸 수 있습니다.
6. 마이그레이션할 테이블을 모두 선택합니다.
7. 필요한 수의 에이전트를 등록합니다. 자세한 정보는 에 추출 에이전트 등록 AWS Schema Conversion Tool을 참조하세요.
8. 선택한 테이블에 대한 데이터 추출 작업을 생성합니다. 자세한 정보는 AWS SCT 데이터 추출 작업 생성, 실행, 모니터링을 참조하세요.

   큰 테이블이 하위 작업으로 분할되어 있는지, 각 하위 작업이 소스 데이터 웨어하우스의 한 부분에 있는 테이블의 일부를 나타내는 데이터 세트와 일치하는지 확인합니다.
9. AWS SCT 데이터 추출 에이전트가 원본 테이블에서 데이터 마이그레이션을 완료할 때까지 마이그레이션 프로세스를 시작하고 모니터링하십시오.

Amazon Redshift로 LOB 마이그레이션

Amazon Redshift는 대용량 바이너리 객체(LOB) 저장을 지원하지 않습니다. 하지만 하나 이상의 LOB를 Amazon AWS SCT Redshift로 마이그레이션해야 하는 경우 마이그레이션을 수행할 수 있습니다. 이 작업을 수행하기 위해 AWS SCT 가 Amazon S3 버킷을 사용하여 LOB를 저장하고 Amazon Redshift에 저장된 마이그레이션된 데이터에 Amazon S3 버킷의 URL을 기록합니다.

LOB를 Amazon Redshift로 마이그레이션하려면

1. 프로젝트를 엽니다. AWS SCT
2. 소스 및 대상 데이터베이스에 연결합니다. 대상 데이터베이스에서 메타데이터를 새로 고치고 변환된 테이블이 있는지 확인합니다.
3. 작업에서 Create local task를 선택합니다.

4. Migration mode에서 다음 중 하나를 선택합니다.

   - Extract and upload - 데이터를 추출한 후 Amazon S3에 데이터를 업로드합니다.
   - Extract, upload and copy - 데이터를 추출하고 Amazon S3에 데이터를 업로드한 다음, Amazon Redshift 데이터 웨어하우스로 복사합니다.
5. Amazon S3 settings를 선택합니다.
6. Amazon S3 bucket LOBs folder에서 LOB를 저장하려는 Amazon S3 버킷의 폴더 이름을 입력합니다.

   AWS 서비스 프로필을 사용하는 경우 이 필드는 선택 사항입니다. AWS SCT 프로필의 기본 설정을 사용할 수 있습니다. 다른 Amazon S3 버킷을 사용하려면 여기에 경로를 입력합니다.
7. 프록시 서버를 사용하여 Amazon S3에 데이터를 업로드하려면 프록시 사용 옵션을 켭니다. 그런 다음 데이터 전송 프로토콜을 선택하고 호스트 이름, 포트, 사용자 이름 및 암호를 입력합니다.
8. Federal Information Processing Standard(FIPS) 엔드포인트를 사용하려면 엔드포인트 유형에서 FIPS를 선택합니다. Virtual Private Cloud(VPC) 엔드포인트를 사용하려면 VPCE를 선택합니다. 그런 다음 VPC 엔드포인트에 VPC 엔드포인트의 도메인 이름 시스템(DNS)을 입력합니다.
9. 추출된 파일을 Amazon Redshift로 파일을 복사한 후 Amazon S3에 이러한 파일을 유지하려면 Keep files on Amazon S3 after copying to Amazon Redshift 옵션을 켭니다.
10. 생성을 선택하여 작업을 생성합니다.

데이터 추출 에이전트에 대한 모범 사례 및 문제 해결

다음은 추출 에이전트 사용에 대한 모범 사례와 문제 해결 제안 사항입니다.

문제	문제 해결 제안
성능이 느림	성능을 개선하기 위해 다음이 권장됩니다. • 여러 에이전트를 설치합니다. • 데이터 웨어하우스와 가까운 컴퓨터에 에이전트를 설치합니다. • 단일 에이전트 작업에서 모든 테이블을 실행하지 않도록 합니다.
경합 지연	너무 많은 에이전트가 동시에 데이터 웨어하우스에 액세스하지 않도록 합니다.

문제	문제 해결 제안
에이전트가 일시적으로 작동 중지됨	에이전트가 작동 중지되면 AWS SCT에 각 작업의 상태가 실패한 것으로 표시됩니다. 기다리면 경우에 따라 에이전트가 복구될 수 있습니다. 이런 경우에는 AWS SCT에서 작업 상태가 업데이트됩니다.
에이전트가 영구적으로 작동 중지됨	에이전트를 실행하는 컴퓨터가 영구적으로 작동 중지되는 경우 해당 에이전트가 작업을 실행 중이면 새 에이전트를 대체하여 작업을 계속할 수 있습니다. 원래 에이전트의 작업 폴더가 원래 에이전트와 같은 컴퓨터에 없는 경우에만 새 에이전트를 대체할 수 있습니다. 새 에이전트를 대체하려면 다음을 수행합니다. • 새 컴퓨터에 에이전트를 설치합니다. • 원래 에이전트와 동일한 설정(포트 번호와 작업 폴더 포함)으로 새 에이전트를 구성합니다. • 에이전트를 시작합니다. 에이전트가 시작되면 작업이 사용 가능한 새 에이전트를 검색하고 새 에이전트에서 계속 실행됩니다.

AWS SCT를 사용하여 애플리케이션 SQL 변환

한 엔진에서 다른 엔진으로 데이터베이스 스키마를 변환할 때는 이전 데이터베이스 엔진 대신 새 데이터베이스 엔진과 상호 작용하도록 애플리케이션의 SQL 코드도 업데이트해야 합니다. 변환된 SQL 코드를 보고, 분석하고, 편집하고, 저장할 수 있습니다.

AWS Schema Conversion Tool(AWS SCT)을 사용하여 C++, C#, Java 또는 기타 애플리케이션 코드로 된 SQL을 변환할 수 있습니다. Oracle에서 PostgreSQL로 변환하는 경우 AWS SCT를 사용하여 SQL*Plus 코드를 PSQL로 변환할 수 있습니다. 또한 Oracle에서 PostgreSQL로 변환하는 경우 AWS SCT를 사용하면 C#, C++, Java 및 Pro*C 애플리케이션에 포함된 SQL 코드를 변환할 수도 있습니다.

주제

- 애플리케이션 SQL 변환 개요
- AWS SCT를 사용하여 애플리케이션의 SQL 코드 변환
- AWS SCT를 사용하여 C# 애플리케이션의 SQL 코드 변환
- AWS SCT를 사용하여 C++ 애플리케이션의 SQL 코드 변환
- AWS SCT를 사용하여 Java 애플리케이션의 SQL 코드 변환
- AWS SCT를 사용하여 Pro*C 애플리케이션의 SQL 코드 변환

애플리케이션 SQL 변환 개요

애플리케이션의 SQL 코드를 변환하려면 다음과 같은 개략적인 단계를 수행합니다.

- 애플리케이션 변환 프로젝트 생성 - 애플리케이션 변환 프로젝트는 데이터베이스 스키마 변환 프로젝트의 하위 프로젝트입니다. 각 데이터베이스 스키마 변환 프로젝트에는 하위 애플리케이션 변환 프로젝트가 하나 이상 포함될 수 있습니다. 자세한 내용은 AWS SCT에서 일반 애플리케이션 변환 프로젝트 생성 섹션을 참조하세요.
- SQL 코드 분석 및 변환 - AWS SCT가 애플리케이션을 분석하고, SQL 코드를 추출하고, 검토 및 편집할 수 있도록 변환된 SQL의 로컬 버전을 생성합니다. 이 도구는 준비가 완료될 때까지 애플리케이션의 코드를 변경하지 않습니다. 자세한 내용은 AWS SCT에서 SQL 코드 분석 및 변환 섹션을 참조하세요.
- 애플리케이션 평가 보고서 생성 - 애플리케이션 평가 보고서는 소스 데이터베이스 스키마의 애플리케이션 SQL 코드를 대상 데이터베이스 스키마로 변환하는 데에 대한 중요한 정보를 제공합니다. 자세한 내용은 AWS SCT에서 AWS SCT 평가 보고서 생성 및 사용 섹션을 참조하세요.

- 변환된 SQL 코드 편집, 변경 내용 적용 및 저장 - 평가 보고서에는 자동으로 변환할 수 없는 SQL 코드 항목 목록이 포함됩니다. 이러한 항목의 경우 SQL 코드를 수동으로 편집하여 변환을 수행할 수 있습니다. 자세한 내용은 AWS SCT를 사용하여 변환된 SQL 코드 편집 및 저장 섹션을 참조하세요.

AWS SCT를 사용하여 애플리케이션의 SQL 코드 변환

AWS SCT를 사용하여 애플리케이션에 포함된 SQL 코드를 변환할 수 있습니다. 일반 AWS SCT 애플리케이션 변환기는 애플리케이션 코드를 일반 텍스트로 취급합니다. 또한 애플리케이션 코드를 스캔하고 정규 표현식을 사용하여 SQL 코드를 추출합니다. 이 변환기는 다양한 유형의 소스 코드 파일을 지원하며 모든 프로그래밍 언어로 작성된 애플리케이션 코드에서 작동됩니다.

일반 애플리케이션 변환기에는 다음과 같은 제한 사항이 있습니다. 애플리케이션의 프로그래밍 언어에 특정한 애플리케이션 로직에 대해서는 자세히 설명하지 않습니다. 또한 일반 변환기는 함수, 파라미터, 로컬 변수 등과 같은 다른 애플리케이션 객체의 SQL 문을 지원하지 않습니다.

애플리케이션 SQL 코드 변환을 개선하려면 언어별 애플리케이션 SQL 코드 변환기를 사용합니다. 자세한 내용은 C# 애플리케이션의 SQL 코드 변환, Java 애플리케이션의 SQL 코드 변환, Pro*C 애플리케이션의 SQL 코드 변환 섹션을 참조하세요.

AWS SCT에서 일반 애플리케이션 변환 프로젝트 생성

AWS Schema Conversion Tool에서 애플리케이션 변환 프로젝트는 데이터베이스 스키마 변환 프로젝트의 하위 프로젝트입니다. 각 데이터베이스 스키마 변환 프로젝트에는 하위 애플리케이션 변환 프로젝트가 하나 이상 포함될 수 있습니다.

> **Note**
>
> AWS SCT에서는 다음과 같은 소스 및 대상 간의 변환을 지원하지 않습니다.
>
> - Oracle에서 Oracle로
> - PostgreSQL에서 PostgreSQL 또는 Aurora PostgreSQL로
> - MySQL에서 MySQL로
> - SQL Server에서 SQL Server로
> - Amazon Redshift에서 Amazon Redshift로
> - SQL Server에서 Babelfish로
> - SQL Server Integration Services에 AWS Glue로

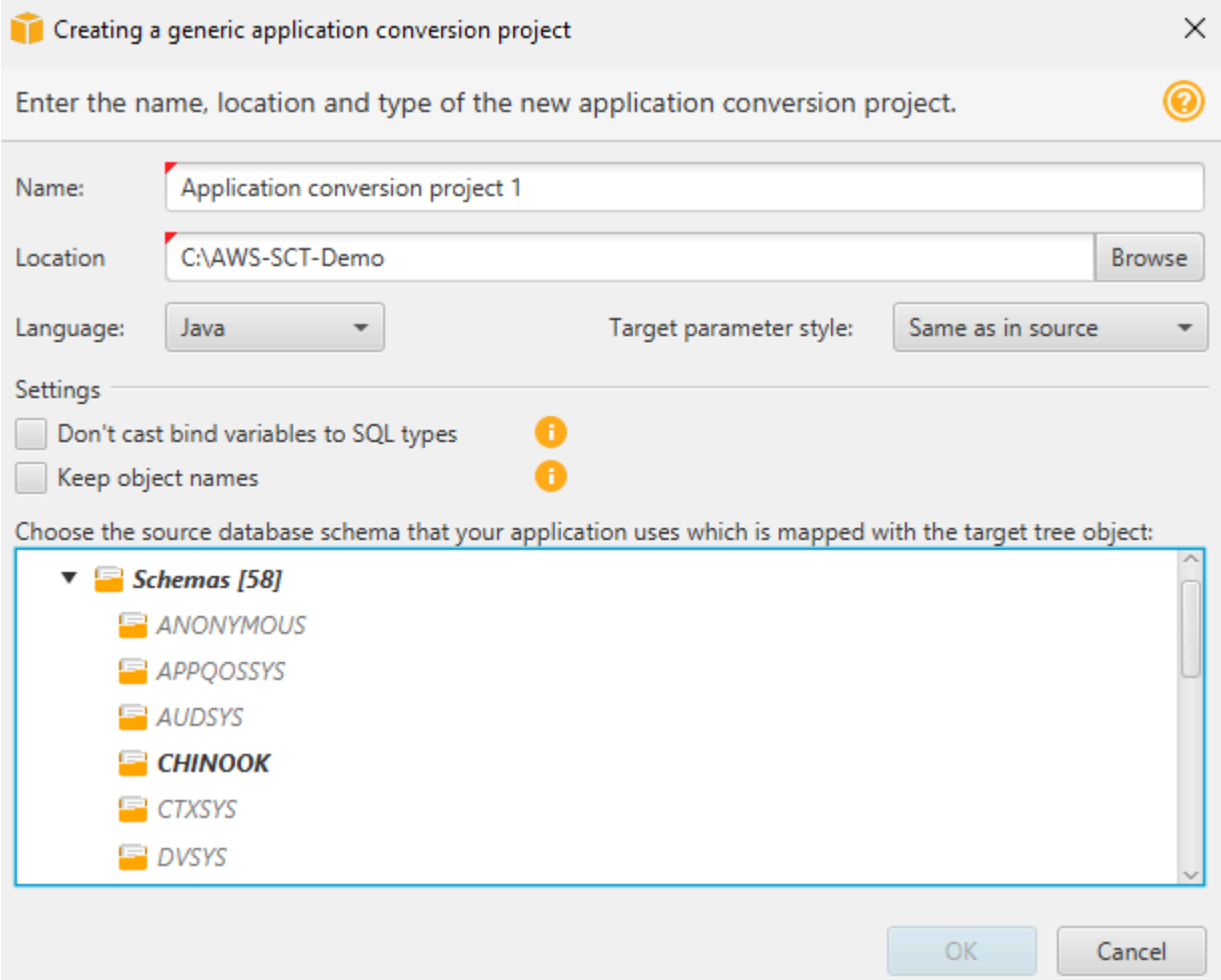
- Apache Cassandra에서 Amazon DynamoDB로

다음 절차에 따라 일반 애플리케이션 변환 프로젝트를 생성합니다.

애플리케이션 변환 프로젝트를 생성하려면

1. AWS Schema Conversion Tool의 애플리케이션 메뉴에서 New generic application을 선택합니다.

   New application conversion project 대화 상자가 나타납니다.



2. 다음 프로젝트 정보를 추가합니다.

이 파라미터의 경우	조치
이름	애플리케이션 변환 프로젝트의 이름을 입력합니다. 각 데이터베이스 스키마 변환 프로젝트에는 하위 애플리케이션 변환 프

이 파라미터의 경우	조치
	로젝트가 하나 이상 있을 수 있으므로 나중에 프로젝트를 더 추가할 경우 적합한 이름을 선택합니다.
위치	애플리케이션 소스 코드의 위치를 입력합니다.
언어	다음 중 하나를 선택합니다. • Java • C++ • C# • 모두
Target parameter style	변환된 코드의 바인드 변수에 사용할 구문을 선택합니다. 데이터베이스 플랫폼마다 바인드 변수에 다른 구문을 사용합니다. 다음 옵션 중 하나를 선택합니다. • Same as in source • Positional (?) • Indexed (:1) • Indexed ($1) • Named (@name) • Named (:name) • Named (&name) • Named ($name) • Named (#name) • Named (!name!)
Choose the source database schema	소스 트리에서 애플리케이션이 사용하는 스키마를 선택합니다. 이 스키마가 매핑 규칙의 일부인지 확인합니다.

3. Don't cast bind variables to SQL types를 선택하여 바인드 변수 유형이 SQL 유형으로 변환되지 않도록 합니다. 이 옵션은 Oracle에서 PostgreSQL로의 변환에서만 지원됩니다.

   예를 들어 소스 애플리케이션 코드에는 다음과 같은 Oracle 쿼리가 포함됩니다.

```
SELECT * FROM ACCOUNT WHERE id = ?
```

Don't cast bind variables to SQL types를 선택하면 AWS SCT가 이 쿼리가 다음과 같이 변환합니다.

```
SELECT * FROM account WHERE id = ?
```

Don't cast bind variables to SQL types를 선택 취소하면 AWS SCT가 바인드 변수 유형을 NUMERIC 데이터 유형으로 변경합니다. 변환 결과는 다음과 같습니다.

```
SELECT * FROM account WHERE id = (?)::NUMERIC
```

4. Keep object names를 선택하여 스키마 이름이 변환된 객체 이름에 추가되지 않도록 합니다. 이 옵션은 Oracle에서 PostgreSQL로의 변환에서만 지원됩니다.

   예를 들어 소스 애플리케이션 코드에 다음과 같은 Oracle 쿼리가 포함되어 있다고 가정합니다.

   ```
   SELECT * FROM ACCOUNT
   ```

   Keep object names를 선택하면 AWS SCT에서 이 쿼리를 다음과 같이 변환합니다.

   ```
   SELECT * FROM account
   ```

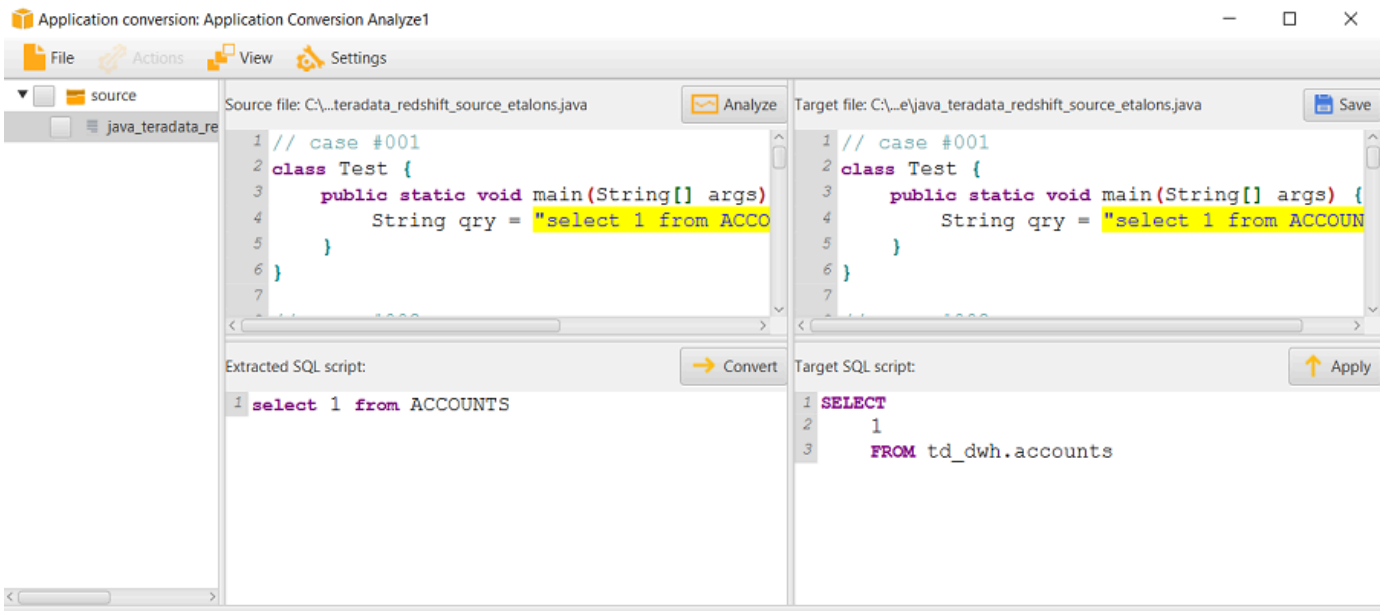
   Keep object names를 선택 취소하면 AWS SCT에서 스키마 이름을 테이블 이름에 추가합니다. 변환 결과는 다음과 같습니다.

   ```
   SELECT * FROM schema_name.account
   ```

   소스 코드에서 객체 이름에 상위 객체의 이름이 포함된 경우 AWS SCT에서 변환된 코드에 이 형식을 사용합니다. 이 경우 AWS SCT가 변환된 코드에 상위 객체의 이름을 추가하므로 Keep object names 옵션을 무시합니다.

5. 확인을 선택하여 애플리케이션 변환 프로젝트를 생성합니다.

   프로젝트 창이 열립니다.



AWS SCT에서 애플리케이션 변환 프로젝트 관리

기존 애플리케이션 변환 프로젝트를 열고 여러 애플리케이션 변환 프로젝트를 추가할 수 있습니다.

애플리케이션 변환 프로젝트를 생성하면 프로젝트 창이 자동으로 열립니다. 애플리케이션 변환 프로젝트 창을 닫은 후 나중에 다시 돌아올 수 있습니다.

기존 애플리케이션 변환 프로젝트를 열려면

1. 왼쪽 패널에서 애플리케이션 변환 프로젝트 노드를 선택하고 컨텍스트(마우스 오른쪽 버튼 클릭) 메뉴를 엽니다.
2. 애플리케이션 관리를 선택합니다.

애플리케이션 변환 프로젝트를 더 추가하려면

1. 왼쪽 패널에서 애플리케이션 변환 프로젝트 노드를 선택하고 컨텍스트(마우스 오른쪽 버튼 클릭) 메뉴를 엽니다.
2. 새 애플리케이션(New application)을 선택합니다.
3. 새 애플리케이션 변환 프로젝트를 생성하는 데 필요한 정보를 입력합니다. 자세한 내용은 일반 애플리케이션 변환 프로젝트 생성 섹션을 참조하세요.

AWS SCT에서 SQL 코드 분석 및 변환

다음 절차를 사용하여 AWS Schema Conversion Tool에서 SQL 코드를 분석하고 변환합니다.

SQL 코드를 분석 및 변환하려면

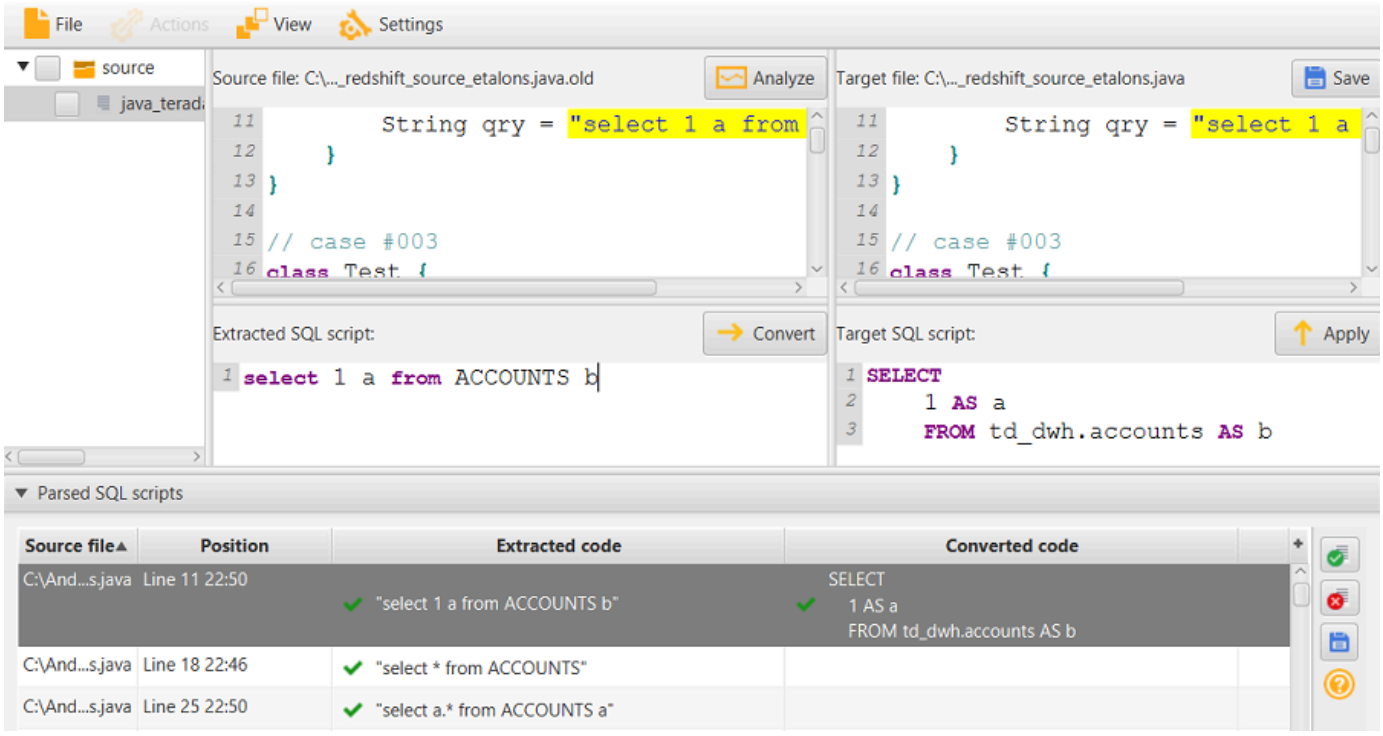
1. 기존 애플리케이션 변환 프로젝트를 열고 분석을 선택합니다.

   AWS SCT가 애플리케이션 코드를 분석하고 SQL 코드를 추출합니다. AWS SCT는 추출된 SQL 코드를 Parsed SQL scripts 목록에 표시합니다.

2. Parsed SQL scripts에서 추출된 SQL 코드를 검토할 항목을 선택합니다. AWS SCT는 선택한 항목의 코드를 Extracted SQL script 창에 표시합니다.

3. Extracted SQL script 창에서 변환을 선택하여 SQL 코드를 변환합니다. AWS SCT는 대상 데이터베이스와 호환되는 형식으로 코드를 변환합니다.

   변환된 SQL 코드를 편집할 수 있습니다. 자세한 내용은 변환된 SQL 코드 편집 및 저장 섹션을 참조하세요.



4. 애플리케이션 변환 평가 보고서를 생성하면 AWS SCT가 추출된 모든 SQL 코드 항목을 변환합니다. 자세한 내용은 평가 보고서 생성 및 사용 섹션을 참조하세요.

AWS SCT에서 AWS SCT 평가 보고서 생성 및 사용

애플리케이션 변환 평가 보고서는 애플리케이션 SQL 코드를 대상 데이터베이스와 호환되는 형식으로 변환하는 방법에 대한 정보를 제공합니다. 보고서에는 추출된 모든 SQL 코드, 변환된 모든 SQL 코드, AWS SCT에서 변환할 수 없는 SQL 코드에 대한 작업 항목이 자세히 설명되어 있습니다.

애플리케이션 변환 평가 보고서 생성

다음 절차에 따라 애플리케이션 변환 평가 보고서를 생성합니다.

애플리케이션 변환 평가 보고서를 생성하려면

1. 애플리케이션 변환 프로젝트 창의 작업 메뉴에서 보고서 생성을 선택합니다.

   AWS SCT에서 애플리케이션 변환 평가 보고서를 생성하고 애플리케이션 변환 프로젝트 창에서 엽니다.
2. 요약 탭을 검토합니다.

   아래에 나온 요약 탭에는 애플리케이션 평가 보고서의 요약 정보가 표시됩니다. 자동으로 변환된 SQL 코드 항목과 자동으로 변환되지 않은 항목을 보여줍니다.

3. SQL extraction actions를 선택합니다.

   AWS SCT가 소스 코드에서 추출할 수 없는 SQL 코드 항목 목록을 검토합니다.
4. SQL conversion actions를 선택합니다.

   AWS SCT가 자동으로 변환할 수 없는 SQL 코드 항목 목록을 검토합니다. 권장 작업을 사용하여 SQL 코드를 수동으로 변환합니다. 변환된 SQL 코드를 편집하는 방법에 대한 자세한 내용은 AWS SCT를 사용하여 변환된 SQL 코드 편집 및 저장 항목을 참조하세요.
5. (선택 사항) 보고서의 로컬 사본을 PDF 파일 또는 쉼표로 구분된 값(CSV) 파일로 저장합니다.

   - 오른쪽 상단에서 Save to PDF를 선택하여 보고서를 PDF 파일로 저장합니다.

     PDF 파일에는 애플리케이션 변환에 대한 요약 정보, 작업 항목 및 권장 사항이 포함되어 있습니다.

- 오른쪽 상단에서 Save to CSV를 선택하여 보고서를 CSV 파일로 저장합니다.

  CSV 파일에는 SQL 코드 변환에 필요한 작업 항목, 권장 작업 및 예상 수작업의 복잡성 등이 포함되어 있습니다.

AWS SCT를 사용하여 변환된 SQL 코드 편집 및 저장

평가 보고서에는 AWS SCT에서 변환할 수 없는 SQL 코드 항목 목록이 포함되어 있습니다. AWS SCT는 각 항목에 대한 작업 항목을 SQL conversion actions 탭에 생성합니다. 이러한 항목의 경우 SQL 코드를 수동으로 편집하여 변환을 수행할 수 있습니다.

다음 절차를 사용하여 변환된 SQL 코드를 편집하고 변경 내용을 적용한 다음 저장합니다.

변환된 SQL 코드를 편집하고, 변경 내용을 적용하고 저장하려면

1. 변환된 SQL 코드를 Target SQL script 창에서 직접 편집합니다. 변환된 코드가 표시되지 않으면 창을 클릭하여 입력할 수 있습니다.
2. 변환된 SQL 코드의 편집을 마친 후에는 적용을 선택합니다. 이 시점에서는 변경 내용이 메모리에 저장되지만 아직 파일에 기록되지 않습니다.
3. 저장을 선택하여 변경 내용을 파일에 저장합니다.

   저장을 선택하면 원본 파일에 덮어씁니다. 원본 애플리케이션 코드를 기록할 수 있도록 저장하기 전에 원본 파일의 사본을 만듭니다.

AWS SCT를 사용하여 C# 애플리케이션의 SQL 코드 변환

Oracle에서 PostgreSQL로 변환하는 경우 AWS Schema Conversion Tool(AWS SCT)을 사용하면 C# 애플리케이션에 포함된 SQL 코드를 변환할 수 있습니다. 이 특정 C# 애플리케이션 변환기는 애플리케이션 로직을 이해합니다. 또한 이 변환기는 함수, 파라미터, 로컬 변수 등과 같은 다양한 애플리케이션 객체에 있는 명령문을 수집합니다.

이러한 심층 분석 덕분에 C# 애플리케이션 SQL 코드 변환기는 일반 변환기보다 더 나은 변환 결과를 제공합니다.

AWS SCT에서 C# 애플리케이션 변환 프로젝트 생성

Oracle 데이터베이스 스키마를 PostgreSQL 데이터베이스 스키마로 변환하기 위한 C# 애플리케이션 변환 프로젝트를 생성할 수 있습니다. 소스 Oracle 스키마와 대상 PostgreSQL 데이터베이스를 포함하

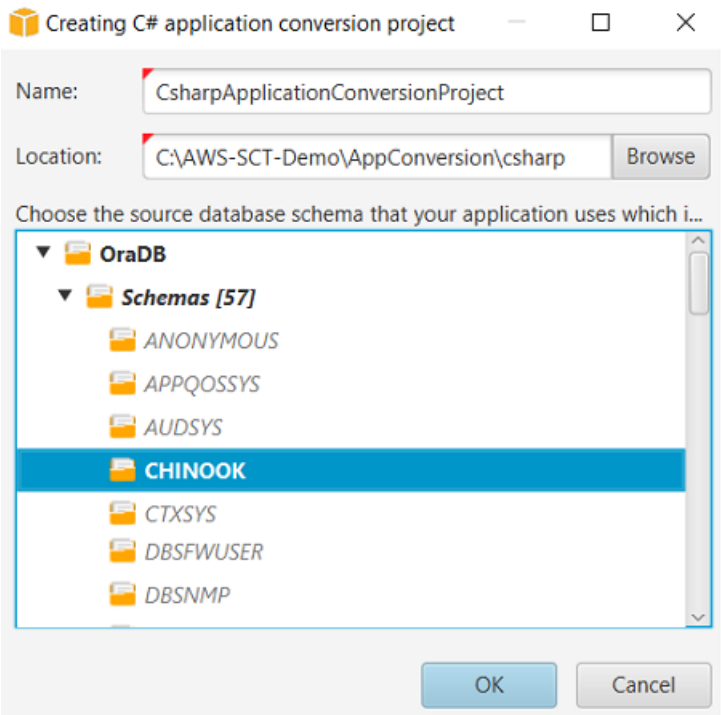
는 매핑 규칙을 프로젝트에 추가해야 합니다. 자세한 내용은 AWS SCT에서 매핑 규칙 생성 섹션을 참조하세요.

단일 AWS SCT 프로젝트에 여러 애플리케이션 변환 프로젝트를 추가할 수 있습니다. 다음 절차에 따라 C# 애플리케이션 변환 프로젝트를 생성합니다.

C# 애플리케이션 변환 프로젝트를 생성하려면

1. 데이터베이스 변환 프로젝트를 만들고 소스 Oracle 데이터베이스를 추가합니다. 자세한 정보는 AWS SCT 프로젝트 생성 및 AWS SCT 프로젝트에 데이터베이스 서버 추가 섹션을 참조하세요.
2. 소스 Oracle 데이터베이스와 대상 PostgreSQL 데이터베이스를 포함하는 매핑 규칙을 추가합니다. 매핑 규칙에 대상 PostgreSQL 데이터베이스를 추가하거나 가상 PostgreSQL 대상 데이터베이스 플랫폼을 사용할 수 있습니다. 자세한 정보는 AWS SCT에서 매핑 규칙 생성 및 가상 대상 사용 섹션을 참조하세요.
3. 보기 메뉴에서 Main view를 선택합니다.
4. 애플리케이션 메뉴에서 New C# application을 선택합니다.

   Creating a C# application conversion project 대화 상자가 나타납니다.



5. 이름에 C# 애플리케이션 변환 프로젝트의 이름을 입력합니다. 각 데이터베이스 스키마 변환 프로젝트에는 하위 애플리케이션 변환 프로젝트가 하나 이상 있을 수 있으므로 여러 프로젝트를 추가할 경우 적합한 이름을 선택합니다.

6. 위치에 애플리케이션의 소스 코드 위치를 입력합니다.
7. 소스 트리에서 애플리케이션이 사용하는 스키마를 선택합니다. 이 스키마는 매핑 규칙에 속해야 합니다. AWS SCT는 매핑 규칙에 속하는 스키마를 굵게 강조 표시합니다.
8. 확인을 선택하여 C# 애플리케이션 변환 프로젝트를 생성합니다.
9. 왼쪽 패널의 애플리케이션 노드에서 C# 애플리케이션 변환 프로젝트를 찾습니다.

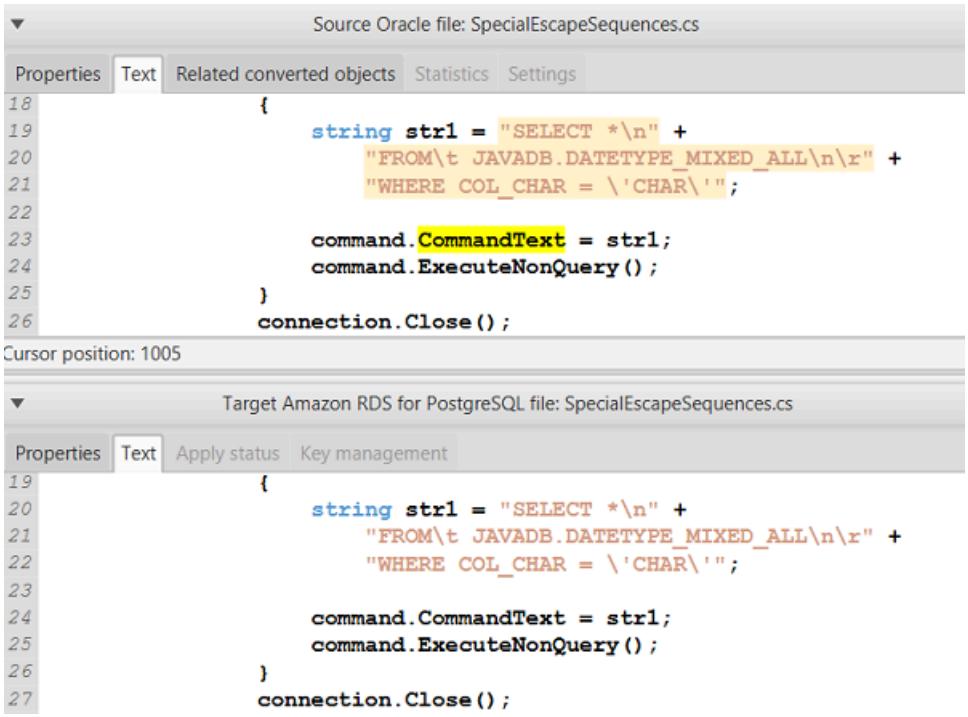
AWS SCT에서 C# 애플리케이션 SQL 코드 변환

AWS SCT 프로젝트에 C# 애플리케이션을 추가한 후 이 애플리케이션의 SQL 코드를 대상 데이터베이스 플랫폼과 호환되는 형식으로 변환합니다. 다음 절차를 사용하여 AWS Schema Conversion Tool에서 C# 애플리케이션에 포함된 SQL 코드를 분석하고 변환할 수 있습니다.

SQL 코드를 변환하려면

1. 왼쪽 패널의 애플리케이션에서 C# 노드를 확장합니다.
2. 변환할 애플리케이션을 선택하고 컨텍스트(마우스 오른쪽 버튼 클릭) 메뉴를 엽니다.
3. 변환을 선택합니다. AWS SCT가 소스 코드 파일을 분석하고, 애플리케이션 로직을 결정한 후 코드 메타데이터를 프로젝트에 로드합니다. 이 코드 메타데이터에는 C# 클래스, 객체, 메서드, 전역 변수, 인터페이스 등이 포함됩니다.

   AWS SCT는 대상 데이터베이스 패널에서 소스 애플리케이션 프로젝트와 유사한 폴더 구조를 만듭니다. 여기서 변환된 애플리케이션 코드를 검토할 수 있습니다.



4. 변환된 애플리케이션 코드를 저장합니다. 자세한 내용은 변환된 애플리케이션 코드 저장 섹션을 참조하세요.

C# 애플리케이션에는 다양한 소스 데이터베이스와 상호 작용하는 SQL 코드가 포함될 수 있습니다. 이러한 소스 데이터베이스 중 일부를 PostgreSQL로 마이그레이션할 수 있습니다. 이 경우 마이그레이션 범위에서 제외한 데이터베이스와 상호 작용하는 SQL 코드를 변환하지 않도록 해야 합니다. 변환 범위에서 C# 애플리케이션의 소스 파일을 제외할 수 있습니다. 이렇게 하려면 변환 범위에서 제외할 파일 이름의 확인란을 선택 취소합니다.

변환 범위를 변경한 후에도 AWS SCT는 계속 C# 애플리케이션의 모든 소스 파일에 대해 SQL 코드를 분석합니다. 그런 다음 AWS SCT는 변환 범위에서 제외한 모든 소스 파일을 대상 폴더에 복사합니다. 이 작업을 수행하면 변환된 애플리케이션 파일을 저장한 후 애플리케이션을 빌드할 수 있습니다.

AWS SCT를 사용하여 변환된 애플리케이션 코드 저장

다음 절차에 따라 변환된 애플리케이션 코드를 저장합니다.

변환된 애플리케이션 코드를 저장하려면

1. 대상 데이터베이스 패널의 애플리케이션에서 C# 노드를 확장합니다.
2. 변환된 애플리케이션을 선택하고 저장을 선택합니다.
3. 변환된 애플리케이션 코드를 저장할 폴더의 경로를 입력하고 폴더 선택을 선택합니다.

AWS SCT에서 C# 애플리케이션 변환 프로젝트 관리

여러 C# 애플리케이션 변환 프로젝트를 추가하거나, AWS SCT 프로젝트에서 애플리케이션 코드를 업데이트하거나, AWS SCT 프로젝트에서 C# 변환 프로젝트를 제거할 수 있습니다.

C# 애플리케이션 변환 프로젝트를 더 추가하려면

1. 왼쪽 패널에서 애플리케이션 노드를 확장합니다.
2. C# 노드를 선택하고 컨텍스트(마우스 오른쪽 버튼 클릭) 메뉴를 엽니다.
3. 새 애플리케이션(New application)을 선택합니다.
4. 새 C# 애플리케이션 변환 프로젝트를 생성하는 데 필요한 정보를 입력합니다. 자세한 내용은 C# 애플리케이션 변환 프로젝트 생성 섹션을 참조하세요.

소스 애플리케이션 코드를 변경한 후 AWS SCT 프로젝트로 업로드합니다.

업데이트된 애플리케이션 코드를 업로드하려면

1. 왼쪽 패널의 애플리케이션에서 C# 노드를 확장합니다.
2. 업데이트할 애플리케이션을 선택하고 컨텍스트(마우스 오른쪽 버튼 클릭) 메뉴를 엽니다.
3. 새로 고침을 선택한 다음, 예를 선택합니다.

   AWS SCT가 소스 파일의 애플리케이션 코드를 업로드하고 변환 결과를 제거합니다. AWS SCT에서 변경된 코드와 변환 결과를 유지하려면 새 C# 변환 프로젝트를 만듭니다.

C# 애플리케이션 변환 프로젝트를 제거하려면

1. 왼쪽 패널의 애플리케이션에서 C# 노드를 확장합니다.
2. 제거할 애플리케이션을 선택하고 컨텍스트(마우스 오른쪽 버튼 클릭) 메뉴를 엽니다.
3. 삭제를 선택한 다음, 확인을 선택합니다.

AWS SCT에서 C# 애플리케이션 변환 평가 보고서 생성

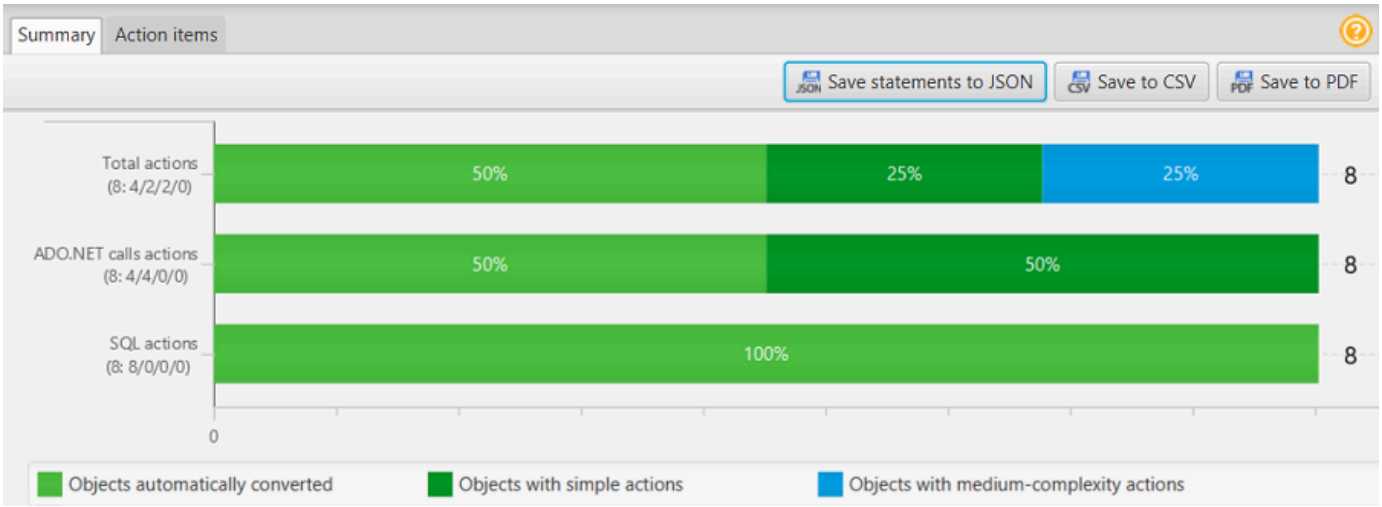
C# 애플리케이션 변환 평가 보고서는 C# 애플리케이션에 포함된 SQL 코드를 대상 데이터베이스와 호환되는 형식으로 변환하는 방법에 대한 정보를 제공합니다. 평가 보고서는 모든 SQL 실행 지점과 모든 소스 코드 파일에 대한 변환 세부 정보를 제공합니다. 평가 보고서에는 AWS SCT에서 변환할 수 없는 SQL 코드에 대한 작업 항목도 포함되어 있습니다.

다음 절차에 따라 C# 애플리케이션 변환 평가 보고서를 생성합니다.

C# 애플리케이션 변환 평가 보고서를 생성하려면

1. 왼쪽 패널의 애플리케이션에서 C# 노드를 확장합니다.
2. 변환할 애플리케이션을 선택하고 컨텍스트(마우스 오른쪽 버튼 클릭) 메뉴를 엽니다.
3. 변환을 선택합니다.
4. 보기 메뉴에서 Assessment report view를 선택합니다.
5. 요약 탭을 검토합니다.

   아래에 나온 요약 탭에는 C# 애플리케이션 평가 보고서의 요약 정보가 표시됩니다. 또한 모든 SQL 실행 지점과 모든 소스 코드 파일에 대한 변환 결과를 보여줍니다.



6. Save statements to JSON을 선택하여 C# 애플리케이션에서 추출된 SQL 코드를 JSON 파일로 저장합니다.
7. (선택 사항) 보고서의 로컬 사본을 PDF 파일 또는 쉼표로 구분된 값(CSV) 파일로 저장합니다.

   - 오른쪽 상단에서 Save to PDF를 선택하여 보고서를 PDF 파일로 저장합니다.

     PDF 파일에는 애플리케이션 변환에 대한 요약 정보, 작업 항목 및 권장 사항이 포함되어 있습니다.
   - 오른쪽 상단에서 Save to CSV를 선택하여 보고서를 CSV 파일로 저장합니다.

     CSV 파일에는 SQL 코드 변환에 필요한 작업 항목, 권장 작업 및 예상 수작업의 복잡성 등이 포함되어 있습니다.

AWS SCT를 사용하여 C++ 애플리케이션의 SQL 코드 변환

Oracle에서 PostgreSQL로 변환하는 경우 AWS SCT를 사용하면 C++ 애플리케이션에 포함된 SQL 코드를 변환할 수 있습니다. 이 특정 C++ 애플리케이션 변환기는 애플리케이션 로직을 이해합니다. 또한 이 변환기는 함수, 파라미터, 로컬 변수 등과 같은 다양한 애플리케이션 객체에 있는 명령문을 수집합니다.

이러한 심층 분석 덕분에 C++ 애플리케이션 SQL 코드 변환기는 일반 변환기보다 더 나은 변환 결과를 제공합니다.

AWS SCT에서 C++ 애플리케이션 변환 프로젝트 생성

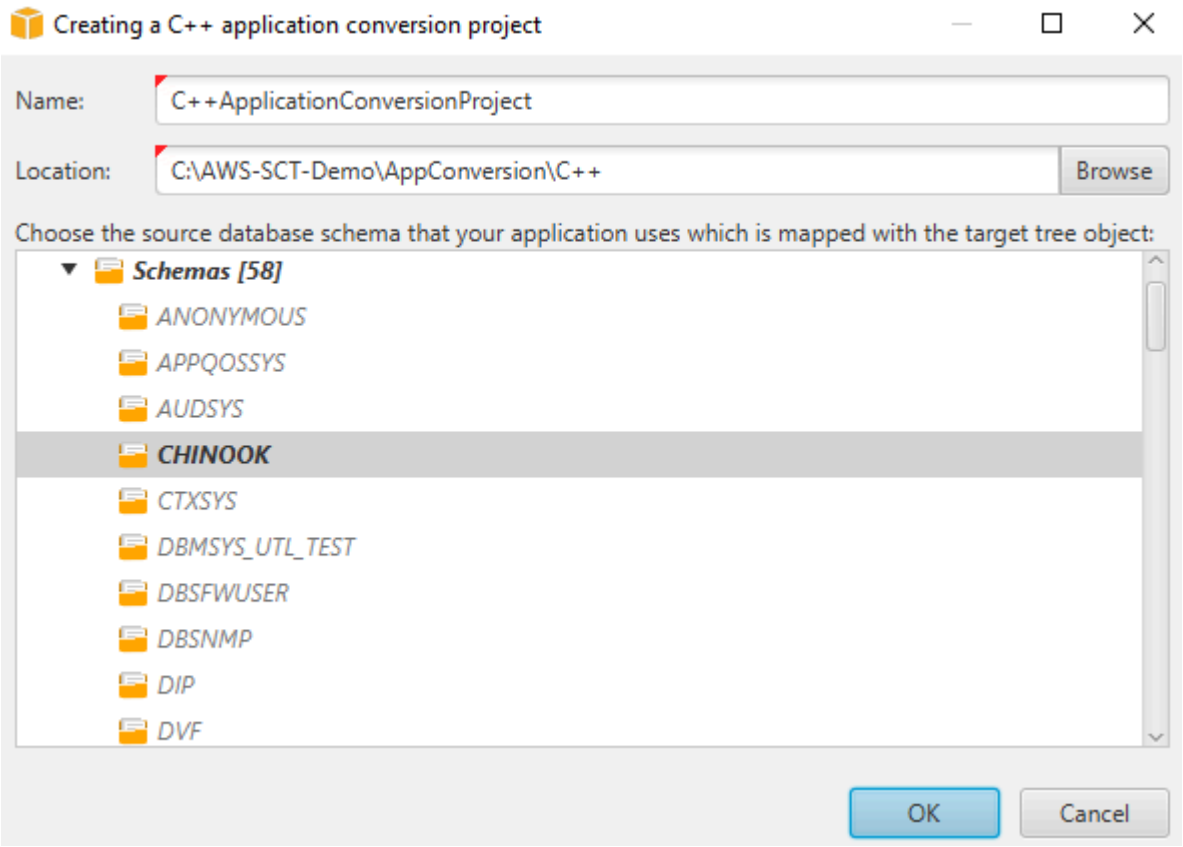
Oracle 데이터베이스 스키마를 PostgreSQL 데이터베이스 스키마로 변환하기 위한 C++ 애플리케이션 변환 프로젝트를 생성할 수 있습니다. 소스 Oracle 스키마와 대상 PostgreSQL 데이터베이스를 포함하는 매핑 규칙을 프로젝트에 추가해야 합니다. 자세한 내용은 AWS SCT에서 매핑 규칙 생성 섹션을 참조하세요.

단일 AWS SCT 프로젝트에 여러 애플리케이션 변환 프로젝트를 추가할 수 있습니다.

C++ 애플리케이션 변환 프로젝트를 생성하려면

1. 데이터베이스 변환 프로젝트를 만들고 소스 Oracle 데이터베이스를 추가합니다. 자세한 정보는 AWS SCT 프로젝트 생성 및 AWS SCT 프로젝트에 데이터베이스 서버 추가 섹션을 참조하세요.
2. 소스 Oracle 데이터베이스와 대상 PostgreSQL 데이터베이스를 포함하는 매핑 규칙을 추가합니다. 매핑 규칙에 대상 PostgreSQL 데이터베이스를 추가하거나 가상 PostgreSQL 대상 데이터베이스 플랫폼을 사용할 수 있습니다. 자세한 정보는 AWS SCT에서 매핑 규칙 생성 및 가상 대상 사용 섹션을 참조하세요.
3. 보기 메뉴에서 Main view를 선택합니다.
4. 애플리케이션 메뉴에서 New C++ application을 선택합니다.

   Creating a C++ application conversion project 대화 상자가 나타납니다.



5. 이름에 C++ 애플리케이션 변환 프로젝트의 이름을 입력합니다. 각 데이터베이스 스키마 변환 프로젝트에는 하위 애플리케이션 변환 프로젝트가 하나 이상 있을 수 있으므로 여러 프로젝트를 추가할 경우 적합한 이름을 선택합니다.
6. 위치에 애플리케이션의 소스 코드 위치를 입력합니다.
7. 소스 트리에서 애플리케이션이 사용하는 스키마를 선택합니다. 이 스키마는 매핑 규칙에 속해야 합니다. AWS SCT는 매핑 규칙에 속하는 스키마를 굵게 강조 표시합니다.
8. 확인을 선택하여 C++ 애플리케이션 변환 프로젝트를 생성합니다.
9. 왼쪽 패널의 애플리케이션 노드에서 C++ 애플리케이션 변환 프로젝트를 찾습니다.

AWS SCT에서 C++ 애플리케이션 SQL 코드 변환

AWS SCT 프로젝트에 C++ 애플리케이션을 추가한 후 이 애플리케이션의 SQL 코드를 대상 데이터베이스 플랫폼과 호환되는 형식으로 변환합니다. 다음 절차를 사용하여 AWS SCT에서 C++ 애플리케이션에 포함된 SQL 코드를 분석하고 변환할 수 있습니다.

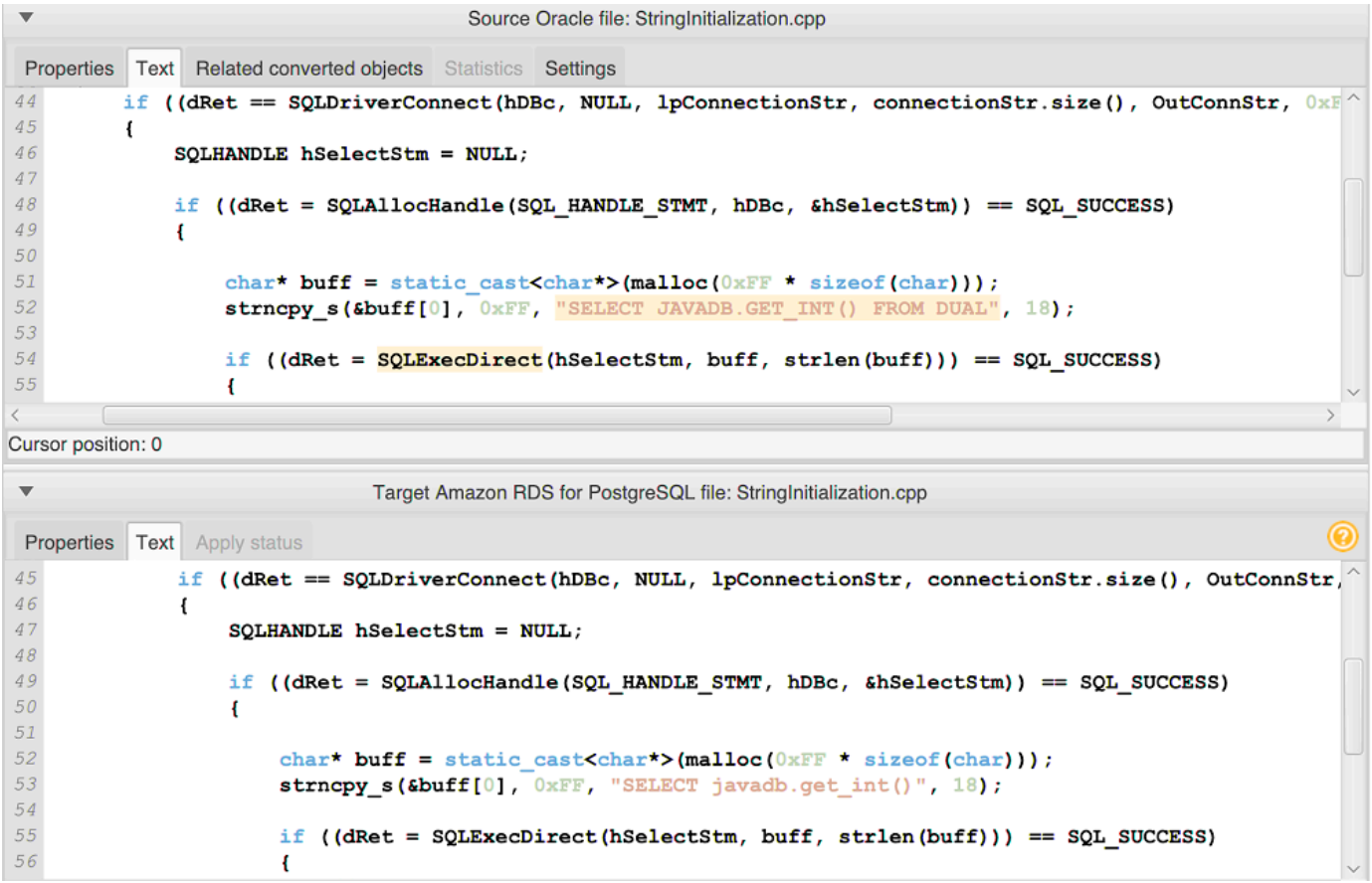
SQL 코드를 변환하려면

1. 왼쪽 패널의 애플리케이션에서 C++ 노드를 확장하고 변환할 애플리케이션을 선택합니다.

2. Source Oracle application project에서 설정을 선택합니다. 선택한 C++ 애플리케이션의 변환 설정을 검토하고 편집합니다. 또한 AWS SCT 프로젝트에 추가한 모든 C++ 애플리케이션의 변환 설정을 지정할 수 있습니다. 자세한 내용은 C++ 애플리케이션 변환 프로젝트 관리 섹션을 참조하세요.
3. 컴파일러 유형에서 C++ 애플리케이션의 소스 코드에 사용할 컴파일러를 선택합니다. AWS SCT는 Microsoft Visual C++, GCC(GNU Compiler Collection) 및 Clang과 같은 C++ 컴파일러를 지원합니다. 기본 옵션은 Microsoft Visual C++입니다.
4. User-defined macros에 C++ 프로젝트의 사용자 정의 매크로가 포함된 파일의 경로를 입력합니다. 이 파일이 `#define name value` 구조인지 확인합니다. 앞의 예와 마찬가지로, `value`는 선택적 파라미터입니다. 이 선택적 파라미터의 기본값은 `1`입니다.

   이 파일을 생성하려면 Microsoft Visual Studio에서 프로젝트를 열고 프로젝트, 속성, C/C++ 및 프리프로세서를 선택합니다. Preprocessor definitions에서 편집을 선택하고 이름 및 값을 새 텍스트 파일에 복사합니다. 그런 다음 파일의 각 문자열에 접두사 `#define` 을 추가합니다.
5. External include directories에 C++ 프로젝트에서 사용할 외부 라이브러리가 포함된 폴더의 경로를 입력합니다.
6. 왼쪽 창에서 변환할 애플리케이션을 선택하고 컨텍스트(마우스 오른쪽 버튼 클릭) 메뉴를 엽니다.
7. 변환을 선택합니다. AWS SCT가 소스 코드 파일을 분석하고, 애플리케이션 로직을 결정한 후 코드 메타데이터를 프로젝트에 로드합니다. 이 코드 메타데이터에는 C++ 클래스, 객체, 메서드, 전역 변수, 인터페이스 등이 포함됩니다.

   AWS SCT는 대상 데이터베이스 패널에서 소스 애플리케이션 프로젝트와 유사한 폴더 구조를 만듭니다. 다음과 같이 변환된 애플리케이션 코드를 검토할 수 있습니다.



8. 변환된 애플리케이션 코드를 저장합니다. 자세한 내용은 변환된 애플리케이션 코드 저장 섹션을 참조하세요.

AWS SCT를 사용하여 변환된 애플리케이션 코드 저장

다음 절차에 따라 변환된 애플리케이션 코드를 저장합니다.

변환된 애플리케이션 코드를 저장하려면

1. 대상 데이터베이스 패널의 애플리케이션에서 C++ 노드를 확장합니다.
2. 변환된 애플리케이션을 선택하고 저장을 선택합니다.
3. 변환된 애플리케이션 코드를 저장할 폴더의 경로를 입력하고 폴더 선택을 선택합니다.

AWS SCT에서 C++ 애플리케이션 변환 프로젝트 관리

여러 C++ 애플리케이션 변환 프로젝트를 추가하거나, 변환 설정을 편집하거나, C++ 애플리케이션 코드를 업데이트하거나, AWS SCT 프로젝트에서 C++ 변환 프로젝트를 제거할 수 있습니다.

C++ 애플리케이션 변환 프로젝트를 더 추가하려면

1. 왼쪽 패널에서 애플리케이션 노드를 확장합니다.
2. C++ 노드를 선택하고 컨텍스트(마우스 오른쪽 버튼 클릭) 메뉴를 엽니다.
3. 새 애플리케이션(New application)을 선택합니다.
4. 새 C++ 애플리케이션 변환 프로젝트를 생성하는 데 필요한 정보를 입력합니다. 자세한 내용은 C++ 애플리케이션 변환 프로젝트 생성 섹션을 참조하세요.

AWS SCT 프로젝트의 모든 C++ 애플리케이션 변환 프로젝트에 대한 변환 설정을 지정할 수 있습니다.

모든 C++ 애플리케이션의 변환 설정을 편집하려면

1. 설정 메뉴에서 프로젝트 설정을 선택한 다음 Application conversion을 선택합니다.
2. 컴파일러 유형에서 C++ 애플리케이션의 소스 코드에 사용할 컴파일러를 선택합니다. AWS SCT는 Microsoft Visual C++, GCC(GNU Compiler Collection) 및 Clang과 같은 C++ 컴파일러를 지원합니다. 기본 옵션은 Microsoft Visual C++입니다.
3. User-defined macros에 C++ 프로젝트의 사용자 정의 매크로가 포함된 파일의 경로를 입력합니다. 이 파일이 `#define name value` 구조인지 확인합니다. 앞의 예와 마찬가지로, `value`는 선택적 파라미터입니다. 이 선택적 파라미터의 기본값은 `1`입니다.

   이 파일을 생성하려면 Microsoft Visual Studio에서 프로젝트를 열고 프로젝트, 속성, C/C++ 및 프리프로세서를 선택합니다. Preprocessor definitions에서 편집을 선택하고 이름 및 값을 새 텍스트 파일에 복사합니다. 그런 다음 파일의 각 문자열에 접두사 `#define` 을 추가합니다.
4. External include directories에 C++ 프로젝트에서 사용할 외부 라이브러리가 포함된 폴더의 경로를 입력합니다.
5. 확인을 선택하여 프로젝트 설정을 저장하고 창을 닫습니다.

또는 각 C++ 애플리케이션 변환 프로젝트의 변환 설정을 지정할 수 있습니다. 자세한 내용은 C++ 애플리케이션 SQL 코드 변환 섹션을 참조하세요.

소스 애플리케이션 코드를 변경한 후 AWS SCT 프로젝트로 업로드합니다.

업데이트된 애플리케이션 코드를 업로드하려면

1. 왼쪽 패널의 애플리케이션에서 C++ 노드를 확장합니다.
2. 업데이트할 애플리케이션을 선택하고 컨텍스트(마우스 오른쪽 버튼 클릭) 메뉴를 엽니다.

3. 새로 고침을 선택한 다음, 예를 선택합니다.

   AWS SCT가 소스 파일의 애플리케이션 코드를 업로드하고 변환 결과를 제거합니다. AWS SCT에서 변경된 코드와 변환 결과를 유지하려면 새 C++ 변환 프로젝트를 만듭니다.

   또한 AWS SCT가 선택된 애플리케이션에 지정된 애플리케이션 변환 설정을 제거합니다. 업데이트된 애플리케이션 코드를 업로드하면 AWS SCT가 프로젝트 설정의 기본값을 적용합니다.

C++ 애플리케이션 변환 프로젝트를 제거하려면

1. 왼쪽 패널의 애플리케이션에서 C++ 노드를 확장합니다.
2. 제거할 애플리케이션을 선택하고 컨텍스트(마우스 오른쪽 버튼 클릭) 메뉴를 엽니다.
3. 삭제를 선택한 다음, 확인을 선택합니다.

AWS SCT에서 C++ 애플리케이션 변환 평가 보고서 생성

C++ 애플리케이션 변환 평가 보고서는 C++ 애플리케이션에 포함된 SQL 코드를 대상 데이터베이스와 호환되는 형식으로 변환하는 방법에 대한 정보를 제공합니다. 평가 보고서는 모든 SQL 실행 지점과 모든 소스 코드 파일에 대한 변환 세부 정보를 제공합니다. 평가 보고서에는 AWS SCT에서 변환할 수 없는 SQL 코드에 대한 작업 항목도 포함되어 있습니다.

C++ 애플리케이션 변환 평가 보고서를 생성하려면

1. 왼쪽 패널의 애플리케이션에서 C++ 노드를 확장합니다.
2. 변환할 애플리케이션을 선택하고 컨텍스트(마우스 오른쪽 버튼 클릭) 메뉴를 엽니다.
3. 변환을 선택합니다.
4. 보기 메뉴에서 Assessment report view를 선택합니다.
5. 요약 탭을 검토합니다.

   요약 탭에는 C++ 애플리케이션 평가 보고서의 요약 정보가 표시됩니다. 또한 모든 SQL 실행 지점과 모든 소스 코드 파일에 대한 변환 결과를 보여줍니다.
6. Save statements to JSON을 선택하여 Java 애플리케이션에서 추출된 SQL 코드를 JSON 파일로 저장합니다.
7. (선택 사항) 보고서의 로컬 사본을 PDF 파일 또는 쉼표로 구분된 값(CSV) 파일로 저장합니다.

   - 오른쪽 상단에서 Save to PDF를 선택하여 보고서를 PDF 파일로 저장합니다.

PDF 파일에는 애플리케이션 변환에 대한 요약 정보, 작업 항목 및 권장 사항이 포함되어 있습니다.

- 오른쪽 상단에서 Save to CSV를 선택하여 보고서를 CSV 파일로 저장합니다.

  CSV 파일에는 SQL 코드 변환에 필요한 작업 항목, 권장 작업 및 예상 수작업의 복잡성 등이 포함되어 있습니다.

AWS SCT를 사용하여 Java 애플리케이션의 SQL 코드 변환

Oracle에서 PostgreSQL로 변환하는 경우 AWS Schema Conversion Tool를 사용하면 Java 애플리케이션에 포함된 SQL 코드를 변환할 수 있습니다. 이 특정 Java 애플리케이션 변환기는 애플리케이션 로직을 이해합니다. 또한 이 변환기는 함수, 파라미터, 로컬 변수 등과 같은 다양한 애플리케이션 객체에 있는 명령문을 수집합니다.

이러한 심층 분석 덕분에 Java 애플리케이션 SQL 코드 변환기는 일반 변환기와 비교하여 더 나은 변환 결과를 제공합니다.

Java 애플리케이션이 MyBatis 프레임워크를 사용하여 데이터베이스와 상호 작용하는 경우, AWS SCT를 사용하여 MyBatis XML 파일 및 주석에 포함된 SQL 문을 변환할 수 있습니다. 이러한 SQL 문의 로직을 이해하기 위해 AWS SCT에서는 MyBatis 구성 파일을 사용합니다. AWS SCT는 애플리케이션 폴더에서 이 파일을 자동으로 검색하거나 이 파일의 경로를 수동으로 입력할 수 있습니다.

AWS SCT에서 Java 애플리케이션 변환 프로젝트 생성

Oracle 데이터베이스 스키마를 PostgreSQL 데이터베이스 스키마로 변환하기 위한 Java 애플리케이션 변환 프로젝트를 생성할 수 있습니다. 소스 Oracle 스키마와 대상 PostgreSQL 데이터베이스를 포함하는 매핑 규칙을 프로젝트에 추가해야 합니다. 자세한 내용은 AWS SCT에서 매핑 규칙 생성 섹션을 참조하세요.

단일 AWS SCT 프로젝트에 여러 애플리케이션 변환 프로젝트를 추가할 수 있습니다. 다음 절차에 따라 Java 애플리케이션 변환 프로젝트를 생성합니다.

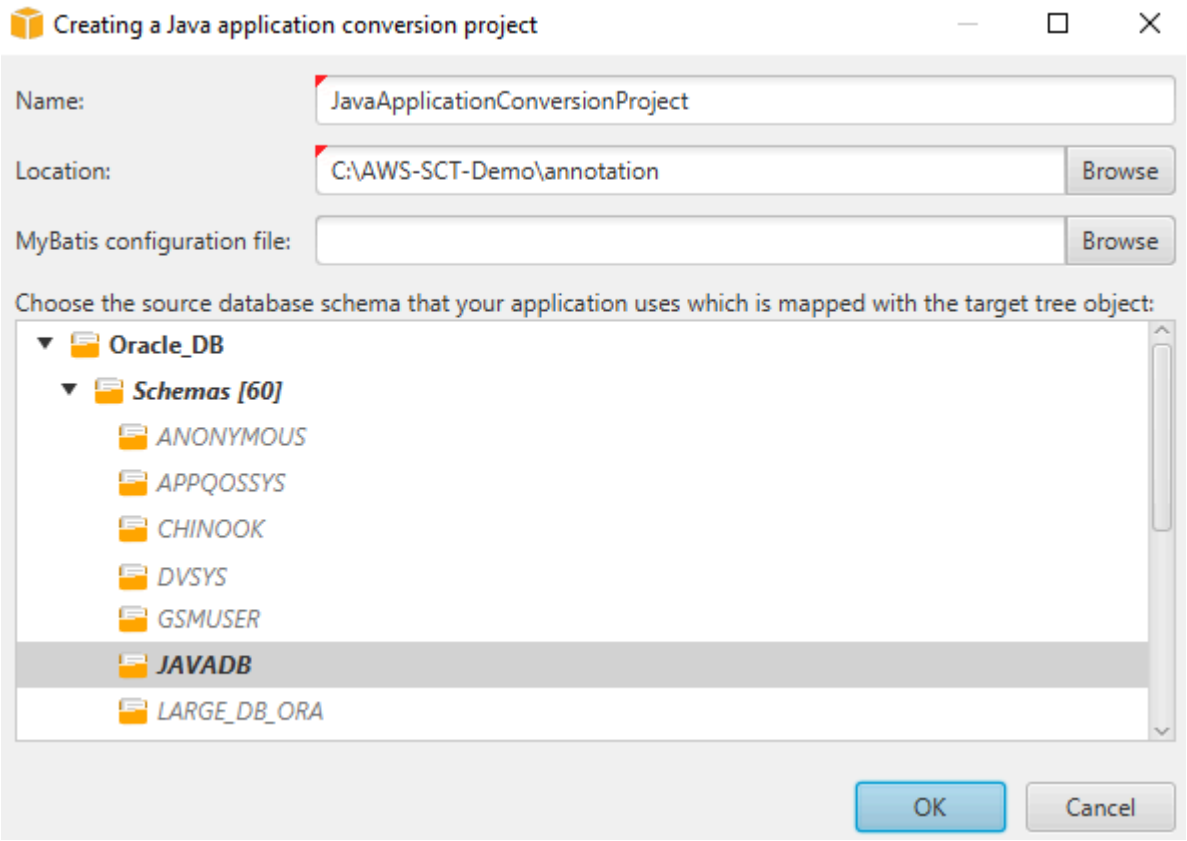
Java 애플리케이션 변환 프로젝트를 생성하려면

1. 데이터베이스 변환 프로젝트를 만들고 소스 Oracle 데이터베이스를 추가합니다. 자세한 정보는 AWS SCT 프로젝트 생성 및 AWS SCT 프로젝트에 데이터베이스 서버 추가 섹션을 참조하세요.
2. 소스 Oracle 데이터베이스와 대상 PostgreSQL 데이터베이스를 포함하는 매핑 규칙을 추가합니다. 매핑 규칙에 대상 PostgreSQL 데이터베이스를 추가하거나 가상 PostgreSQL 대상 데이터베이

스 플랫폼을 사용할 수 있습니다. 자세한 정보는 AWS SCT에서 매핑 규칙 생성 및 가상 대상 사용 섹션을 참조하세요.

3. 보기 메뉴에서 Main view를 선택합니다.
4. 애플리케이션 메뉴에서 New Java application을 선택합니다.

   Creating a Java application conversion project 대화 상자가 나타납니다.



5. 이름에 Java 애플리케이션 변환 프로젝트의 이름을 입력합니다. 각 데이터베이스 스키마 변환 프로젝트에는 하위 애플리케이션 변환 프로젝트가 하나 이상 있을 수 있으므로 여러 프로젝트를 추가할 경우 적합한 이름을 선택합니다.
6. 위치에 애플리케이션의 소스 코드 위치를 입력합니다.
7. (선택 사항) MyBatis configuration file에 MyBatis 구성 파일의 경로를 입력합니다. AWS SCT가 애플리케이션 폴더를 스캔하여 이 파일을 자동으로 검색합니다. 이 파일이 애플리케이션 폴더에 없거나, 여러 구성 파일을 사용하는 경우에는 경로를 수동으로 입력합니다.
8. 소스 트리에서 애플리케이션이 사용하는 스키마를 선택합니다. 이 스키마는 매핑 규칙에 속해야 합니다. AWS SCT는 매핑 규칙에 속하는 스키마를 굵게 강조 표시합니다.
9. 확인을 선택하여 Java 애플리케이션 변환 프로젝트를 생성합니다.
10. 왼쪽 패널의 애플리케이션 노드에서 Java 애플리케이션 변환 프로젝트를 찾습니다.

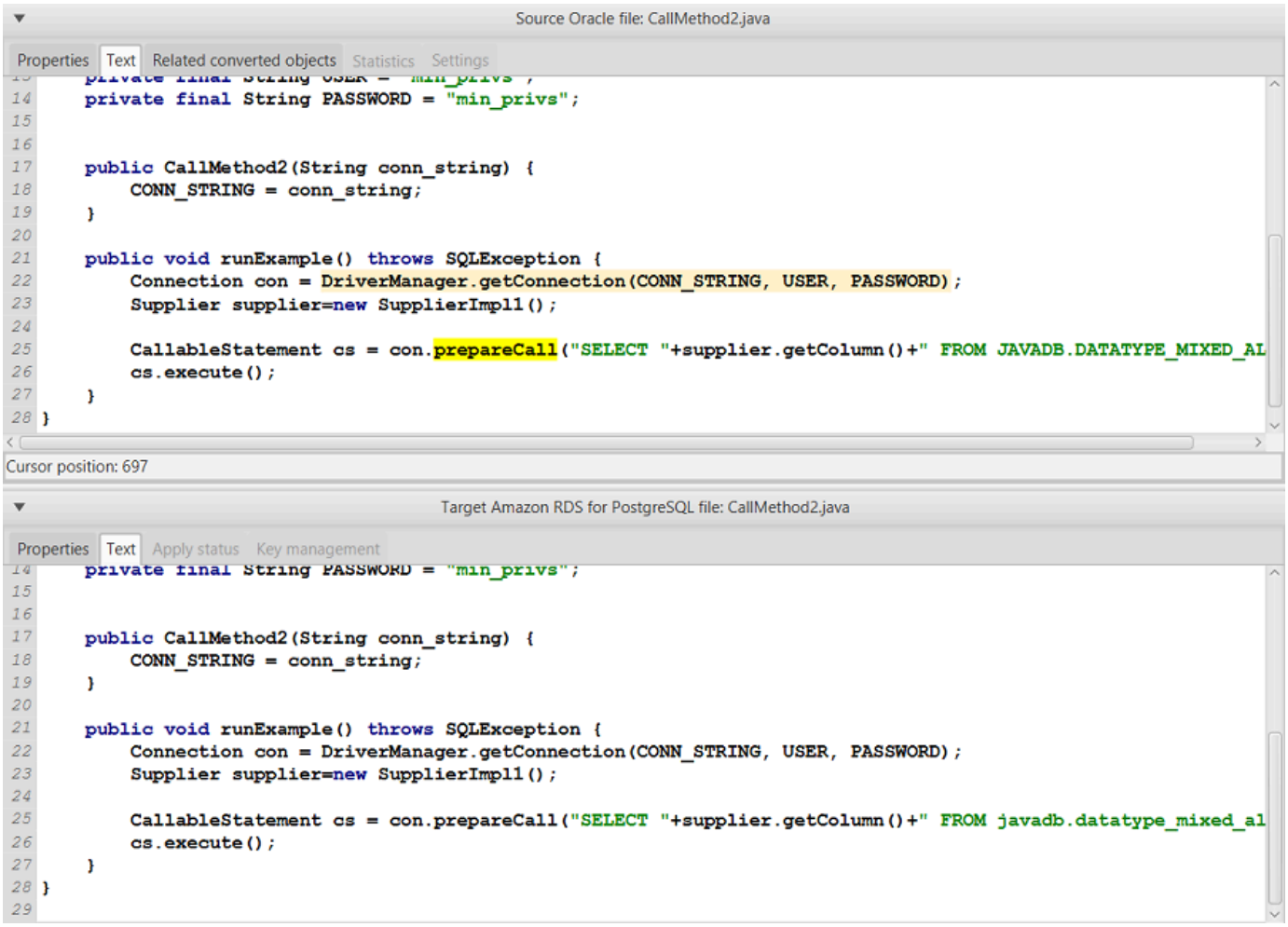
AWS SCT에서 Java 애플리케이션 SQL 코드 변환

AWS SCT 프로젝트에 Java 애플리케이션을 추가한 후 이 애플리케이션의 SQL 코드를 대상 데이터베이스 플랫폼과 호환되는 형식으로 변환합니다. 다음 절차를 사용하여 AWS Schema Conversion Tool에서 Java 애플리케이션에 포함된 SQL 코드를 분석하고 변환할 수 있습니다.

SQL 코드를 변환하려면

1. 왼쪽 패널의 애플리케이션에서 Java 노드를 확장합니다.
2. 변환할 애플리케이션을 선택하고 컨텍스트(마우스 오른쪽 버튼 클릭) 메뉴를 엽니다.
3. 변환을 선택합니다. AWS SCT가 소스 코드 파일을 분석하고, 애플리케이션 로직을 결정한 후 코드 메타데이터를 프로젝트에 로드합니다. 이 코드 메타데이터에는 Java 클래스, 객체, 메서드, 전역 변수, 인터페이스 등이 포함됩니다.

   AWS SCT는 대상 데이터베이스 패널에서 소스 애플리케이션 프로젝트와 유사한 폴더 구조를 만듭니다. 여기서 변환된 애플리케이션 코드를 검토할 수 있습니다.



4. 변환된 애플리케이션 코드를 저장합니다. 자세한 내용은 변환된 애플리케이션 코드 저장 섹션을 참조하세요.

Java 애플리케이션에는 다양한 소스 데이터베이스와 상호 작용하는 SQL 코드가 포함될 수 있습니다. 이러한 소스 데이터베이스 중 일부를 PostgreSQL로 마이그레이션할 수 있습니다. 이 경우 마이그레이션 범위에서 제외한 데이터베이스와 상호 작용하는 SQL 코드를 변환하지 않도록 해야 합니다. 변환 범위에서 Java 애플리케이션의 소스 파일을 제외할 수 있습니다. 이렇게 하려면 변환 범위에서 제외할 파일 이름의 확인란을 선택 취소합니다.

변환 범위를 변경한 후에도 AWS SCT는 계속 Java 애플리케이션의 모든 소스 파일에 대해 SQL 코드를 분석합니다. 그런 다음 AWS SCT는 변환 범위에서 제외한 모든 소스 파일을 대상 폴더에 복사합니다. 이 작업을 수행하면 변환된 애플리케이션 파일을 저장한 후 애플리케이션을 빌드할 수 있습니다.

AWS SCT를 사용하여 변환된 애플리케이션 코드 저장

다음 절차에 따라 변환된 애플리케이션 코드를 저장합니다.

변환된 애플리케이션 코드를 저장하려면

1. 대상 데이터베이스 패널의 애플리케이션에서 Java 노드를 확장합니다.
2. 변환된 애플리케이션을 선택하고 저장을 선택합니다.
3. 변환된 애플리케이션 코드를 저장할 폴더의 경로를 입력하고 폴더 선택을 선택합니다.

소스 Java 애플리케이션이 MyBatis 프레임워크를 사용하는 경우, 새 데이터베이스에서 작동하도록 구성 파일을 업데이트해야 합니다.

AWS SCT에서 Java 애플리케이션 변환 프로젝트 관리

여러 Java 애플리케이션 변환 프로젝트를 추가하거나, AWS SCT 프로젝트에서 애플리케이션 코드를 업데이트하거나, AWS SCT 프로젝트에서 Java 변환 프로젝트를 제거할 수 있습니다.

Java 애플리케이션 변환 프로젝트를 더 추가하려면

1. 왼쪽 패널에서 애플리케이션 노드를 확장합니다.
2. Java 노드를 선택하고 컨텍스트(마우스 오른쪽 버튼 클릭) 메뉴를 엽니다.
3. 새 애플리케이션(New application)을 선택합니다.
4. 새 Java 애플리케이션 변환 프로젝트를 생성하는 데 필요한 정보를 입력합니다. 자세한 내용은 Java 애플리케이션 변환 프로젝트 생성 섹션을 참조하세요.

소스 애플리케이션 코드를 변경한 후 AWS SCT 프로젝트로 업로드합니다.

업데이트된 애플리케이션 코드를 업로드하려면

1. 왼쪽 패널의 애플리케이션에서 Java 노드를 확장합니다.
2. 업데이트할 애플리케이션을 선택하고 컨텍스트(마우스 오른쪽 버튼 클릭) 메뉴를 엽니다.
3. 새로 고침을 선택한 다음, 예를 선택합니다.

   AWS SCT가 소스 파일의 애플리케이션 코드를 업로드하고 변환 결과를 제거합니다. AWS SCT에서 변경된 코드와 변환 결과를 유지하려면 새 Java 변환 프로젝트를 만듭니다.

소스 Java 애플리케이션이 MyBatis 프레임워크를 사용하는 경우, AWS SCT는 MyBatis 구성 파일을 사용하여 SQL 코드를 구문 분석합니다. 이 파일을 변경한 후 AWS SCT 프로젝트로 업로드합니다.

MyBatis 구성 파일의 경로를 편집하려면

1. 왼쪽 패널의 애플리케이션에서 Java 노드를 확장합니다.
2. 애플리케이션을 선택한 후 설정을 선택합니다.
3. 찾아보기를 선택한 다음 MyBatis 구성 파일을 선택합니다.
4. Apply(적용)를 선택합니다.
5. 왼쪽 패널에서 애플리케이션을 선택하고 컨텍스트(마우스 오른쪽 버튼 클릭) 메뉴를 연 다음 새로 고침을 선택합니다.

Java 애플리케이션 변환 프로젝트를 제거하려면

1. 왼쪽 패널의 애플리케이션에서 Java 노드를 확장합니다.
2. 제거할 애플리케이션을 선택하고 컨텍스트(마우스 오른쪽 버튼 클릭) 메뉴를 엽니다.
3. 삭제를 선택한 다음, 확인을 선택합니다.

AWS SCT에서 Java 애플리케이션 변환 평가 보고서 생성

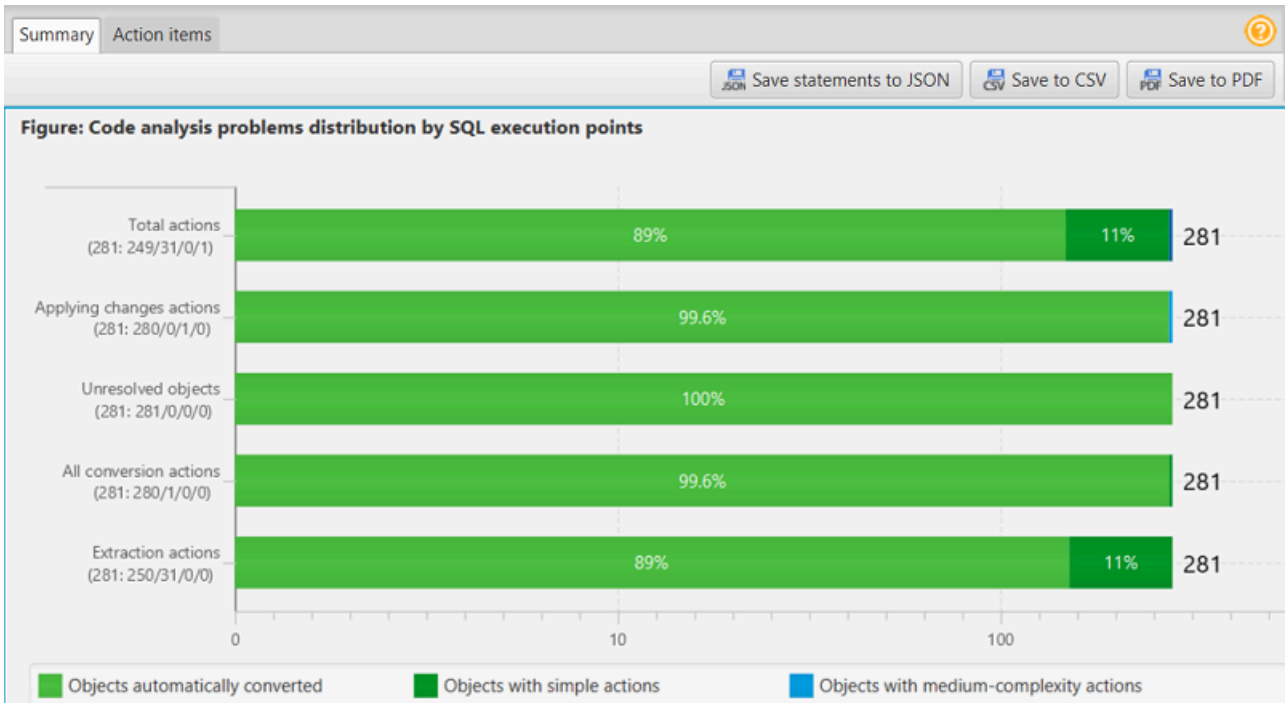
Java 애플리케이션 변환 평가 보고서는 Java 애플리케이션에 포함된 SQL 코드를 대상 데이터베이스와 호환되는 형식으로 변환하는 방법에 대한 정보를 제공합니다. 평가 보고서는 모든 SQL 실행 지점과 모든 소스 코드 파일에 대한 변환 세부 정보를 제공합니다. 평가 보고서에는 AWS SCT에서 변환할 수 없는 SQL 코드에 대한 작업 항목도 포함되어 있습니다.

다음 절차에 따라 Java 애플리케이션 변환 평가 보고서를 생성합니다.

Java 애플리케이션 변환 평가 보고서를 생성하려면

1. 왼쪽 패널의 애플리케이션에서 Java 노드를 확장합니다.
2. 변환할 애플리케이션을 선택하고 컨텍스트(마우스 오른쪽 버튼 클릭) 메뉴를 엽니다.
3. 변환을 선택합니다.
4. 보기 메뉴에서 Assessment report view를 선택합니다.
5. 요약 탭을 검토합니다.

   아래에 나온 요약 탭에는 Java 애플리케이션 평가 보고서의 요약 정보가 표시됩니다. 또한 모든 SQL 실행 지점과 모든 소스 코드 파일에 대한 변환 결과를 보여줍니다.



6. Save statements to JSON을 선택하여 Java 애플리케이션에서 추출된 SQL 코드를 JSON 파일로 저장합니다.
7. (선택 사항) 보고서의 로컬 사본을 PDF 파일 또는 쉼표로 구분된 값(CSV) 파일로 저장합니다.

   - 오른쪽 상단에서 Save to PDF를 선택하여 보고서를 PDF 파일로 저장합니다.

     PDF 파일에는 애플리케이션 변환에 대한 요약 정보, 작업 항목 및 권장 사항이 포함되어 있습니다.
   - 오른쪽 상단에서 Save to CSV를 선택하여 보고서를 CSV 파일로 저장합니다.

CSV 파일에는 SQL 코드 변환에 필요한 작업 항목, 권장 작업 및 예상 수작업의 복잡성 등이 포함되어 있습니다.

AWS SCT를 사용하여 Pro*C 애플리케이션의 SQL 코드 변환

Oracle에서 PostgreSQL로 변환하는 경우 AWS Schema Conversion Tool(AWS SCT)을 사용하면 Pro*C 애플리케이션에 포함된 SQL 코드를 변환할 수 있습니다. 이 특정 Pro*C 애플리케이션 변환기는 애플리케이션 로직을 이해합니다. 또한 이 변환기는 함수, 파라미터, 로컬 변수 등과 같은 다양한 애플리케이션 객체에 있는 명령문을 수집합니다.

이러한 심층 분석 덕분에 Pro*C 애플리케이션 SQL 코드 변환기는 일반 변환기와 비교하여 더 나은 변환 결과를 제공합니다.

AWS SCT에서 Pro*C 애플리케이션 변환 프로젝트 생성

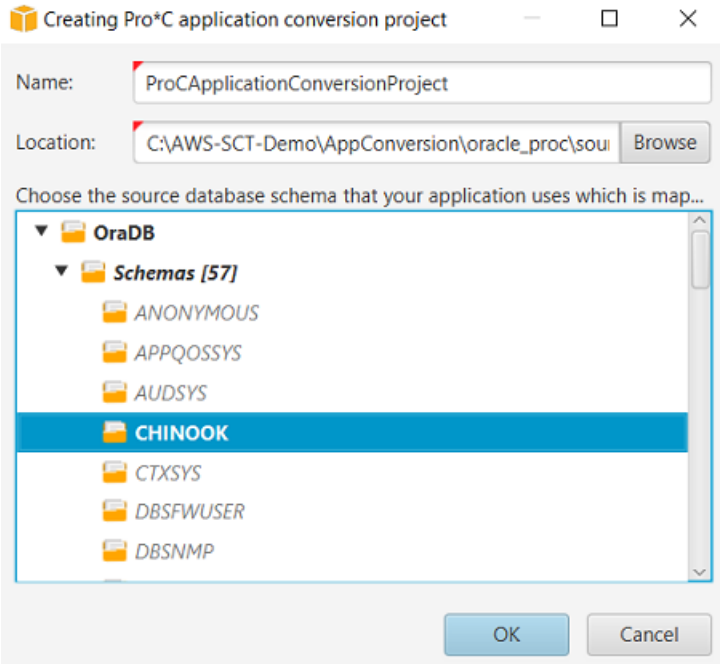
Oracle 데이터베이스 스키마를 PostgreSQL 데이터베이스 스키마로 변환하기 위한 Pro*C 애플리케이션 변환 프로젝트를 생성할 수 있습니다. 소스 Oracle 스키마와 대상 PostgreSQL 데이터베이스를 포함하는 매핑 규칙을 프로젝트에 추가해야 합니다. 자세한 내용은 AWS SCT에서 매핑 규칙 생성 섹션을 참조하세요.

단일 AWS SCT 프로젝트에 여러 애플리케이션 변환 프로젝트를 추가할 수 있습니다. 다음 절차에 따라 Pro*C 애플리케이션 변환 프로젝트를 생성합니다.

Pro*C 애플리케이션 변환 프로젝트를 생성하려면

1. 데이터베이스 변환 프로젝트를 만들고 소스 Oracle 데이터베이스를 추가합니다. 자세한 정보는 AWS SCT 프로젝트 생성 및 AWS SCT 프로젝트에 데이터베이스 서버 추가 섹션을 참조하세요.
2. 소스 Oracle 데이터베이스와 대상 PostgreSQL 데이터베이스를 포함하는 매핑 규칙을 추가합니다. 매핑 규칙에 대상 PostgreSQL 데이터베이스를 추가하거나 가상 PostgreSQL 대상 데이터베이스 플랫폼을 사용할 수 있습니다. 자세한 정보는 AWS SCT에서 매핑 규칙 생성 및 가상 대상 사용 섹션을 참조하세요.
3. 보기 메뉴에서 Main view를 선택합니다.
4. 애플리케이션 메뉴에서 New Pro*C application을 선택합니다.

   Creating a Pro*C application conversion project 대화 상자가 나타납니다.



5. 이름에 Pro*C 애플리케이션 변환 프로젝트의 이름을 입력합니다. 각 데이터베이스 스키마 변환 프로젝트에는 하위 애플리케이션 변환 프로젝트가 하나 이상 있을 수 있으므로 여러 프로젝트를 추가할 경우 적합한 이름을 선택합니다.
6. 위치에 애플리케이션의 소스 코드 위치를 입력합니다.
7. 소스 트리에서 애플리케이션이 사용하는 스키마를 선택합니다. 이 스키마는 매핑 규칙에 속해야 합니다. AWS SCT는 매핑 규칙에 속하는 스키마를 굵게 강조 표시합니다.
8. 확인을 선택하여 Pro*C 애플리케이션 변환 프로젝트를 생성합니다.
9. 왼쪽 패널의 애플리케이션 노드에서 Pro*C 애플리케이션 변환 프로젝트를 찾습니다.

AWS SCT에서 Pro*C 애플리케이션 SQL 코드 변환

AWS SCT 프로젝트에 Pro*C 애플리케이션을 추가한 후 이 애플리케이션의 SQL 코드를 대상 데이터베이스 플랫폼과 호환되는 형식으로 변환합니다. 다음 절차를 사용하여 AWS Schema Conversion Tool에서 Pro*C 애플리케이션에 포함된 SQL 코드를 분석하고 변환할 수 있습니다.

SQL 코드를 변환하려면

1. 왼쪽 패널의 애플리케이션에서 Pro*C 노드를 확장합니다.
2. 변환할 애플리케이션을 선택한 후 설정을 선택합니다.

    a. Global header file path에 애플리케이션 프로젝트에서 사용하는 헤더 파일의 경로를 입력합니다.

b. 변환된 코드의 모든 미해결 변수를 확인하려면 Interpret all unresolved host variables as를 선택합니다.

c. 변환된 SQL 코드에서 확장 팩 함수를 사용하려면 Use fixed-width string conversion function from the extension pack을 선택합니다. AWS SCT가 애플리케이션 프로젝트에 확장 팩 파일을 포함합니다.

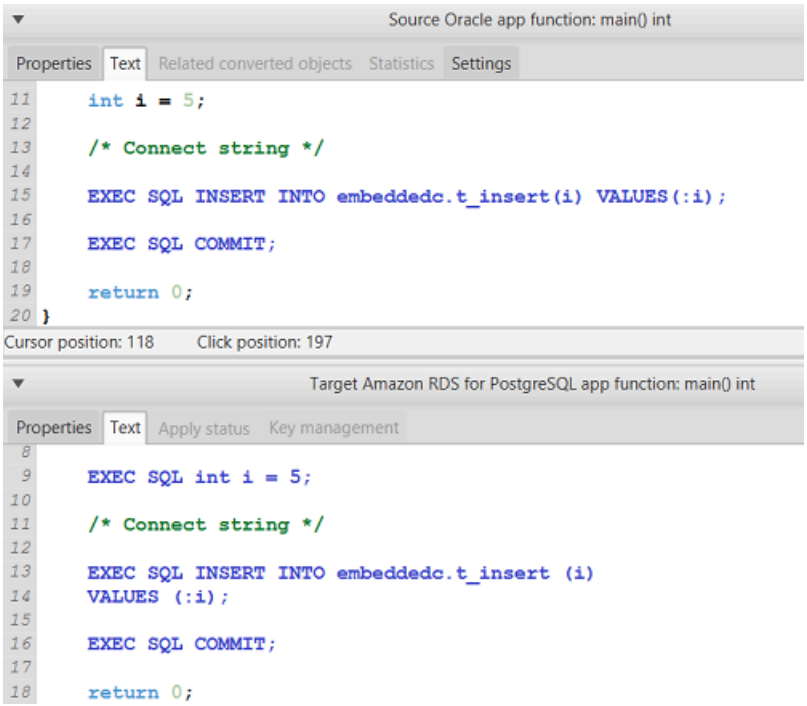
d. 대상 데이터베이스에 모든 익명 PL/SQL 블록에 대한 저장 프로시저를 만들려면 Transform anonymous PL/SQL blocks to standalone SQL calls or stored functions를 선택합니다. 그러면 AWS SCT가 변환된 애플리케이션 코드에 이러한 저장 프로시저의 실행을 포함합니다.

e. Oracle 데이터베이스 커서의 변환을 개선하려면 Use custom cursor flow를 선택합니다.

3. 왼쪽 패널에서 변환할 애플리케이션을 선택하고 컨텍스트(마우스 오른쪽 버튼 클릭) 메뉴를 엽니다.

4. 변환을 선택합니다. AWS SCT가 소스 코드 파일을 분석하고, 애플리케이션 로직을 결정한 후 코드 메타데이터를 프로젝트에 로드합니다. 이 코드 메타데이터에는 Pro*C 클래스, 객체, 메서드, 전역 변수, 인터페이스 등이 포함됩니다.

   AWS SCT는 대상 데이터베이스 패널에서 소스 애플리케이션 프로젝트와 유사한 폴더 구조를 만듭니다. 여기서 변환된 애플리케이션 코드를 검토할 수 있습니다.



5. 변환된 애플리케이션 코드를 저장합니다. 자세한 내용은 변환된 애플리케이션 코드 편집 및 저장 섹션을 참조하세요.

AWS SCT를 사용하여 변환된 애플리케이션 코드 편집 및 저장

변환된 SQL 문을 편집하고 AWS SCT를 사용하여 이 편집된 코드를 변환된 Pro*C 애플리케이션 코드에 포함할 수 있습니다. 다음 절차에 따라 변환된 SQL 코드를 편집합니다.

변환된 SQL 코드를 편집하려면

1. 왼쪽 패널의 애플리케이션에서 Pro*C 노드를 확장합니다.
2. 변환할 애플리케이션을 선택하고 컨텍스트(마우스 오른쪽 버튼 클릭) 메뉴를 연 후 변환을 선택합니다.
3. 보기 메뉴에서 Assessment report view를 선택합니다.
4. Save statements to CSV를 선택하여 Pro*C 애플리케이션에서 추출된 SQL 코드를 CSV 파일로 저장합니다.
5. 추출된 SQL 코드를 저장할 CSV 파일의 이름을 입력하고 저장을 선택합니다.
6. 추출된 SQL 코드를 편집합니다.
7. 보기 메뉴에서 Main view를 선택합니다.
8. 대상 데이터베이스 패널의 애플리케이션에서 Pro*C 노드를 확장합니다.
9. 변환된 애플리케이션을 선택하고 컨텍스트(마우스 오른쪽 버튼 클릭) 메뉴를 연 다음 Import statements from CSV를 선택합니다.
10. 예를 선택한 다음, 편집한 SQL 코드가 있는 파일을 선택하고 열기를 선택합니다.

AWS SCT가 변환된 SQL 문을 분할하여 소스 애플리케이션 코드의 적절한 객체에 배치합니다. 다음 절차에 따라 변환된 애플리케이션 코드를 저장합니다.

변환된 애플리케이션 코드를 저장하려면

1. 대상 데이터베이스 패널의 애플리케이션에서 Pro*C 노드를 확장합니다.
2. 변환된 애플리케이션을 선택하고 저장을 선택합니다.
3. 변환된 애플리케이션 코드를 저장할 폴더의 경로를 입력하고 폴더 선택을 선택합니다.

AWS SCT에서 Pro*C 애플리케이션 변환 프로젝트 관리

여러 Pro*C 애플리케이션 변환 프로젝트를 추가하거나, AWS SCT 프로젝트에서 애플리케이션 코드를 업데이트하거나, AWS SCT 프로젝트에서 Pro*C 변환 프로젝트를 제거할 수 있습니다.

Pro*C 애플리케이션 변환 프로젝트를 더 추가하려면

1. 왼쪽 패널에서 애플리케이션 노드를 확장합니다.
2. Pro*C 노드를 선택하고 컨텍스트(마우스 오른쪽 버튼 클릭) 메뉴를 엽니다.
3. 새 애플리케이션(New application)을 선택합니다.
4. 새 Pro*C 애플리케이션 변환 프로젝트를 생성하는 데 필요한 정보를 입력합니다. 자세한 내용은 Pro*C 애플리케이션 변환 프로젝트 생성 섹션을 참조하세요.

소스 애플리케이션 코드를 변경한 후 AWS SCT 프로젝트로 업로드합니다.

업데이트된 애플리케이션 코드를 업로드하려면

1. 왼쪽 패널의 애플리케이션에서 Pro*C 노드를 확장합니다.
2. 업데이트할 애플리케이션을 선택하고 컨텍스트(마우스 오른쪽 버튼 클릭) 메뉴를 엽니다.
3. 새로 고침을 선택한 다음, 예를 선택합니다.

   AWS SCT가 소스 파일의 애플리케이션 코드를 업로드하고 변환 결과를 제거합니다. AWS SCT에서 변경된 코드와 변환 결과를 유지하려면 새 Pro*C 변환 프로젝트를 만듭니다.

Pro*C 애플리케이션 변환 프로젝트를 제거하려면

1. 왼쪽 패널의 애플리케이션에서 Pro*C 노드를 확장합니다.
2. 제거할 애플리케이션을 선택하고 컨텍스트(마우스 오른쪽 버튼 클릭) 메뉴를 엽니다.
3. 삭제를 선택한 다음, 확인을 선택합니다.

AWS SCT에서 Pro*C 애플리케이션 변환 평가 보고서 생성

Pro*C 애플리케이션 변환 평가 보고서는 Pro*C 애플리케이션에 포함된 SQL 코드를 대상 데이터베이스와 호환되는 형식으로 변환하는 방법에 대한 정보를 제공합니다. 평가 보고서는 모든 SQL 실행 지점과 모든 소스 코드 파일에 대한 변환 세부 정보를 제공합니다. 평가 보고서에는 AWS SCT에서 변환할 수 없는 SQL 코드에 대한 작업 항목도 포함되어 있습니다.

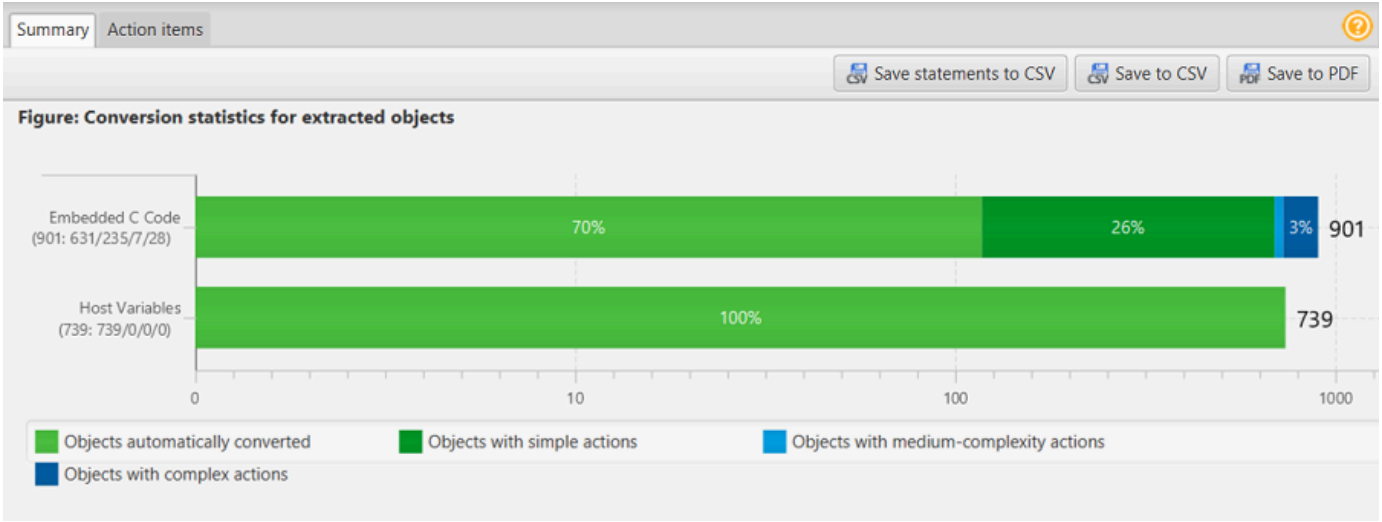
다음 절차에 따라 Pro*C 애플리케이션 변환 평가 보고서를 생성합니다.

Pro*C 애플리케이션 변환 평가 보고서를 생성하려면

1. 왼쪽 패널의 애플리케이션에서 Pro*C 노드를 확장합니다.

2. 변환할 애플리케이션을 선택하고 컨텍스트(마우스 오른쪽 버튼 클릭) 메뉴를 엽니다.
3. 변환을 선택합니다.
4. 보기 메뉴에서 Assessment report view를 선택합니다.
5. 요약 탭을 검토합니다.

   아래에 나온 요약 탭에는 Pro*C 애플리케이션 평가 보고서의 요약 정보가 표시됩니다. 또한 모든 SQL 실행 지점과 모든 소스 코드 파일에 대한 변환 결과를 보여줍니다.



6. Save statements to CSV를 선택하여 Pro*C 애플리케이션에서 추출된 SQL 코드를 쉼표로 구분된 값(CSV) 파일로 저장합니다.
7. (선택 사항) 보고서의 로컬 사본을 PDF 파일 또는 쉼표로 구분된 값(CSV) 파일로 저장합니다.

   - 오른쪽 상단에서 Save to PDF를 선택하여 보고서를 PDF 파일로 저장합니다.

     PDF 파일에는 애플리케이션 변환에 대한 요약 정보, 작업 항목 및 권장 사항이 포함되어 있습니다.
   - 오른쪽 상단에서 Save to CSV를 선택하여 보고서를 CSV 파일로 저장합니다.

     CSV 파일에는 SQL 코드 변환에 필요한 작업 항목, 권장 작업 및 예상 수작업의 복잡성 등이 포함되어 있습니다.

AWS SCT 확장 팩 사용

AWS SCT 확장 팩은 객체를 대상 데이터베이스로 변환할 때 필요한 소스 데이터베이스에 있는 함수를 에뮬레이션하는 추가 기능 모듈입니다. AWS SCT 확장 팩을 설치하려면 먼저 데이터베이스 스키마를 변환해야 합니다.

각 AWS SCT 확장 팩에는 다음과 같은 구성 요소가 포함되어 있습니다.

- DB 스키마 - 특정 온라인 트랜잭션 처리(OLTP) 및 온라인 분석 처리(OLAP) 데이터베이스 객체(예: 시퀀스)를 에뮬레이션하기 위한 SQL 함수, 프로시저 및 테이블을 포함합니다. 또한 원본 built-in-functions 데이터베이스에서 지원되지 않는 것을 에뮬레이션합니다. 이 스키마 이름의 형식은 `aws_`*`database_engine_name`*`_ext`와 같습니다.
- AWS Lambda 함수 (특정 OLTP 데이터베이스용) - 작업 예약 및 이메일 전송과 같은 복잡한 데이터베이스 기능을 에뮬레이션하는 AWS Lambda 함수를 포함합니다.
- OLAP 데이터베이스용 사용자 지정 라이브러리 - Microsoft SQL Server 통합 서비스 (SSIS) 추출, 변환 및 로드 (ETL) 스크립트를 또는 로 마이그레이션하는 데 사용할 수 있는 Java 및 Python 라이브러리 세트를 포함합니다. AWS Glue AWS Glue Studio

  Java 라이브러리에는 다음과 같은 모듈이 포함됩니다.

  - `spark-excel_2.11-0.13.1.jar` - Excel 소스 및 대상 구성 요소의 기능을 에뮬레이션합니다.
  - `spark-xml_2.11-0.9.0.jar`, `poi-ooxml-schemas-4.1.2.jar` 및 `xmlbeans-3.1.0.jar` - XML 소스 구성 요소의 기능을 에뮬레이션합니다.

  Python 라이브러리에는 다음과 같은 모듈이 포함됩니다.

  - `sct_utils.py` - 소스 데이터 유형을 에뮬레이션하고 Spark SQL 쿼리를 위한 파라미터를 준비합니다.
  - `ssis_datetime.py` - 날짜 및 시간 내장 함수를 에뮬레이션합니다.
  - `ssis_null.py` - `ISNULL` 및 `REPLACENULL` 내장 함수를 에뮬레이션합니다.
  - `ssis_string.py` - 문자열 내장 함수를 에뮬레이션합니다.

  이러한 라이브러리에 대한 자세한 내용은 AWS SCT 확장 팩용 사용자 지정 라이브러리 사용 섹션을 참조하세요.

AWS SCT 확장 팩은 다음 두 가지 방법으로 적용할 수 있습니다.

- AWS SCT 컨텍스트 메뉴에서 데이터베이스에 적용을 선택하여 대상 데이터베이스 스크립트를 적용할 때 확장 팩을 자동으로 적용할 수 있습니다. AWS SCT 다른 모든 스키마 개체를 적용하기 전에 확장 팩을 적용합니다.
- 확장 팩을 수동으로 적용하려면 대상 데이터베이스를 선택한 다음 컨텍스트(마우스 오른쪽 버튼 클릭) 메뉴에서 Apply extension pack for를 선택합니다. 대부분의 경우 자동 적용 방법으로 충분합니다. 하지만 팩을 실수로 삭제한 경우에는 수동으로 팩을 적용해야 할 수 있습니다.

대상 데이터 저장소에 AWS SCT 확장 팩을 적용할 때마다 구성요소를 덮어쓰고 이에 대한 알림이 AWS SCT 표시됩니다. 이러한 알림을 끄려면 설정, 전역 설정, 알림 및 Hide the extension pack replacement alert를 차례로 선택합니다.

Microsoft SQL Server에서 PostgreSQL로 변환하려면 AWS SCT에서 SQL Server를 PostgreSQL로 변환하는 확장 팩을 사용할 수 있습니다. 이 확장 팩은 SQL Server 에이전트와 SQL Server Database Mail을 에뮬레이션합니다. 자세한 내용은 확장 팩을 사용하여 PostgreSQL에서 SQL Server 에이전트 에뮬레이션 및 확장 팩을 사용하여 PostgreSQL에서 SQL Server Database Mail 에뮬레이션 섹션을 참조하세요.

다음에서 AWS SCT 확장 팩 사용에 대한 자세한 내용을 확인할 수 있습니다.

주제

- AWS SCT 확장 팩 사용 권한
- 확장 팩 스키마 사용
- AWS SCT 확장 팩용 사용자 지정 라이브러리 사용
- AWS SCT 확장 팩의 AWS Lambda 함수 사용
- AWS SCT 확장 팩의 함수 구성

AWS SCT 확장 팩 사용 권한

Amazon Aurora용 AWS SCT 확장 팩은 함수를 사용하여 메일 전송, 작업 예약, 대기열 및 기타 작업을 에뮬레이션합니다. AWS Lambda 대상 Aurora 데이터베이스에 AWS SCT 확장 팩을 적용하면 새 AWS Identity and Access Management (IAM) 역할과 인라인 IAM 정책이 AWS SCT 생성됩니다. 다음으로, 새 Lambda 함수를 AWS SCT 생성하고 아웃바운드 연결을 위해 Aurora DB 클러스터를 구성합니다. AWS Lambda이러한 작업을 실행하려면 다음과 같은 필수 권한을 IAM 사용자에게 부여해야 합니다.

- `iam:CreateRole`— 계정의 새 IAM 역할을 생성합니다. AWS

- `iam:CreatePolicy`— 계정에 AWS 대한 새 IAM 정책을 만들려면
- `iam:AttachRolePolicy` - 지정된 정책을 IAM 역할에 연결합니다.
- `iam:PutRolePolicy` - IAM 역할에 포함된 인라인 정책 문서를 업데이트합니다.
- `iam:PassRole` - 지정된 IAM 역할을 규칙 엔진에 전달합니다.
- `iam:TagRole` - IAM 역할에 태그를 추가합니다.
- `iam:TagPolicy` - IAM 정책에 태그를 추가합니다.
- `lambda:ListFunctions` - Lambda 함수 목록을 볼 수 있습니다.
- `lambda:ListTags` - Lambda 함수의 태그 목록을 볼 수 있습니다.
- `lambda:CreateFunction` – 새 Lambda 함수를 생성합니다.
- `rds:AddRoleToDBCluster` – IAM 역할을 Aurora DB 클러스터와 연결합니다.

Amazon Redshift용 AWS SCT 확장 팩은 변환된 객체를 Amazon Redshift에 적용할 때 필요한 소스 데이터 웨어하우스 기반 함수를 에뮬레이션합니다. 변환된 코드를 Amazon Redshift에 적용하기 전에 Amazon Redshift용 확장 팩을 적용해야 합니다. 이렇게 하려면 IAM 정책에 해당 `iam:SimulatePrincipalPolicy` 작업을 포함시킵니다.

AWS SCT IAM 정책 시뮬레이터를 사용하여 Amazon Redshift 확장 팩 설치에 필요한 권한을 확인합니다. IAM 사용자를 올바르게 구성했더라도 IAM 정책 시뮬레이터에 오류 메시지가 표시될 수 있습니다. 이는 IAM 정책 시뮬레이터의 알려진 문제입니다. 또한 IAM 정책 시뮬레이터는 IAM 정책에 해당 `iam:SimulatePrincipalPolicy` 작업이 없는 경우 오류 메시지를 표시합니다. 이러한 경우에는 오류 메시지를 무시하고 확장 팩 마법사를 사용하여 확장 팩을 적용할 수 있습니다. 자세한 정보는 확장 팩 적용을 참조하세요.

확장 팩 스키마 사용

데이터베이스 또는 데이터 웨어하우스 스키마를 변환할 때 AWS SCT 가 대상 데이터베이스에 추가 스키마를 추가합니다. 이 스키마는 대상 데이터베이스에 변환된 코드를 쓸 데 필요한 소스 데이터베이스의 SQL 시스템 함수를 구현합니다. 이 추가 스키마를 확장 팩 스키마라고 합니다.

OLTP 데이터베이스의 확장 팩 스키마는 소스 데이터베이스에 따라 다음과 같이 이름이 지정됩니다.

- Microsoft SQL Server: `AWS_SQLSERVER_EXT`
- MySQL: `AWS_MYSQL_EXT`
- Oracle: `AWS_ORACLE_EXT`
- PostgreSQL: `AWS_POSTGRESQL_EXT`

OLAP 데이터 웨어하우스 애플리케이션의 확장 팩 스키마는 다음과 같이 소스 데이터 스토어에 따라 이름이 지정됩니다.

- Greenplum: `AWS_GREENPLUM_EXT`
- Microsoft SQL Server: `AWS_SQLSERVER_EXT`
- Netezza: `AWS_NETEZZA_EXT`
- Oracle: `AWS_ORACLE_EXT`
- Teradata: `AWS_TERADATA_EXT`
- Vertica: `AWS_VERTICA_EXT`

AWS SCT 확장 팩용 사용자 지정 라이브러리 사용

원본 데이터베이스 기능을 대상 데이터베이스의 동등한 기능으로 변환할 AWS SCT 수 없는 경우도 있습니다. 관련 AWS SCT 확장 팩에는 대상 데이터베이스의 일부 소스 데이터베이스 기능을 에뮬레이션하는 사용자 지정 라이브러리가 포함되어 있습니다.

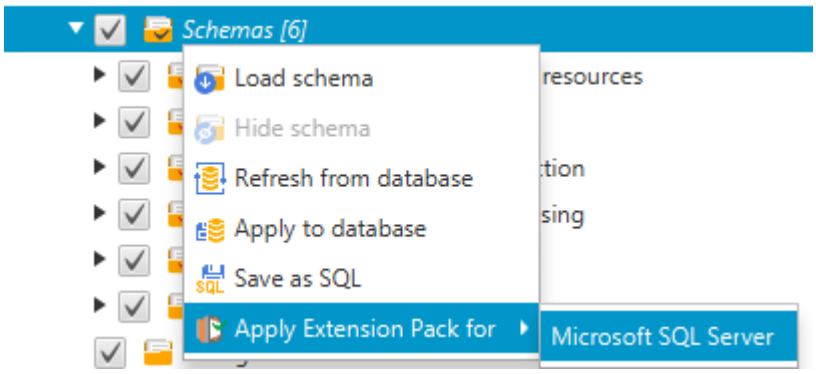
트랜잭션 데이터베이스를 변환하는 경우 AWS SCT 확장 팩의 AWS Lambda 함수 사용 섹션을 참조합니다.

확장 팩 적용

AWS SCT 확장 팩 마법사를 사용하거나 변환된 코드를 대상 데이터베이스에 적용할 때 확장 팩을 적용할 수 있습니다.

확장 팩 마법사를 사용하여 확장 팩을 적용하려면

1. 대상 데이터베이스 트리에서 컨텍스트 (마우스 오른쪽 버튼 클릭) 메뉴를 열고 확장 팩 적용을 선택한 다음 원본 데이터베이스 플랫폼을 선택합니다. AWS Schema Conversion Tool



확장 팩 마법사가 표시됩니다.

2. 시작 페이지를 읽은 후 다음을 선택합니다.
3. AWS profile settings 페이지에서 다음 작업을 수행합니다.

   - 확장 팩 스키마만 다시 설치하는 경우에는 Skip this step for now를 선택하고 다음을 선택합니다. Skip this step for now 옵션은 온라인 트랜잭션 처리(OLTP) 데이터베이스에만 사용할 수 있습니다.
   - 새 라이브러리를 업로드하는 경우 AWS 계정에 연결하는 데 필요한 보안 인증 정보를 제공합니다. 이 단계는 OLAP 데이터베이스 또는 ETL 스크립트를 변환할 때만 사용합니다. AWS Command Line Interface (AWS CLI) 자격 증명이 AWS CLI 설치되어 있는 경우 해당 자격 증명을 사용할 수 있습니다. 또한 이전에 전역 애플리케이션 설정의 프로필에 저장한 후 프로젝트와 연결한 보안 인증 정보를 사용할 수도 있습니다. 필요한 경우 글로벌 설정으로 이동을 선택하여 다른 프로필을 구성하거나 AWS SCT 프로젝트에 연결할 수 있습니다. 자세한 정보는 AWS SCT에 AWS 서비스 프로필 저장을 참조하세요.
4. 새 라이브러리를 업로드하는 경우 Library upload 페이지에서 I need to upload a library를 선택합니다. 이 단계는 OLAP 데이터베이스 또는 ETL 스크립트를 변환할 때만 사용합니다. 다음으로 Amazon S3 경로를 입력한 다음 Upload library to S3을 선택합니다.

   라이브러리를 이미 업로드한 경우 Library upload 페이지에서 I already have libraries uploaded, use my existing S3 bucket을 선택합니다. 다음으로, Amazon S3 경로를 입력합니다.

   완료되면 다음을 선택합니다.
5. Function emulation 페이지에서 Create extension pack을 선택합니다. 확장 팩 작업 상태가 포함된 메시지가 표시됩니다.

   모두 마쳤으면 완료를 선택합니다.

변환된 코드를 적용할 때 확장 팩을 적용하려면

1. AWS 서비스 프로필에 Amazon S3 버킷을 지정합니다. 이 단계는 OLAP 데이터베이스 또는 ETL 스크립트를 변환할 때만 사용합니다. 자세한 정보는 AWS SCT에 AWS 서비스 프로필 저장을 참조하세요.

   Amazon S3 버킷 정책에는 다음 권한이 포함되어야 합니다.

```
{
    "Version": "2012-10-17",
```

```
    "Statement": [
      {
        "Effect": "Allow",
        "Action": ["s3:ListBucket"],
        "Resource": ["*"]
      },
      {
        "Effect": "Allow",
        "Action": ["s3:PutObject"],
        "Resource": ["*"]
      },
      {
        "Effect": "Allow",
        "Action": ["iam:SimulatePrincipalPolicy"],
        "Resource": ["*"]
      },
      {
        "Effect": "Allow",
        "Action": ["iam:GetUser"],
        "Resource": ["arn:aws:iam::111122223333:user/DataExtractionAgentName"]
      }
    ]
}
```

이전 예제에서 *DataExtractionAgentName111122223333:user/#* IAM 사용자의 이름으로 바꾸십시오.

2. 소스 데이터 웨어하우스 스키마를 변환합니다. 자세한 정보는 데이터 웨어하우스 스키마를 Amazon Redshift로 변환을 참조하세요.
3. 오른쪽 창에서 변환된 스키마를 선택합니다.
4. 스키마 요소의 컨텍스트(마우스 오른쪽 버튼 클릭) 메뉴를 열고 Apply to database를 선택합니다.
5. AWS SCT 필수 구성 요소가 포함된 확장 팩을 생성하고 대상 트리에 스키마를 추가합니다. aws_*database_engine_name*_ext 그런 다음 변환된 코드와 확장 팩 스키마를 대상 데이터 웨어하우스에 AWS SCT 적용합니다.

   Amazon AWS Glue Redshift와 대상 데이터베이스 플랫폼을 함께 사용하는 경우 확장 팩에 스키마를 AWS SCT 추가합니다.

AWS SCT 확장 팩의 AWS Lambda 함수 사용

AWS SCT 이메일을 위한 Lambda 함수, 작업 예약 및 Amazon EC2에 호스팅되는 데이터베이스의 기타 기능을 포함하는 확장 팩을 제공합니다.

AWS Lambda 함수를 사용하여 데이터베이스 기능을 에뮬레이션합니다.

데이터베이스 기능을 동등한 Amazon RDS 기능으로 변환할 수 없는 경우도 있습니다. 예를 들어 Oracle은 UTL_SMTP를 사용하는 이메일 호출을 보내고 Microsoft SQL Server는 작업 스케줄러를 사용할 수 있습니다. Amazon EC2에서 데이터베이스를 호스팅하고 자체 관리하는 경우 서비스를 AWS 대체하여 이러한 기능을 에뮬레이션할 수 있습니다.

AWS SCT 확장 팩 마법사는 이메일, 작업 예약 및 기타 기능을 에뮬레이션하도록 Lambda 함수를 설치, 생성 및 구성하는 데 도움을 줍니다.

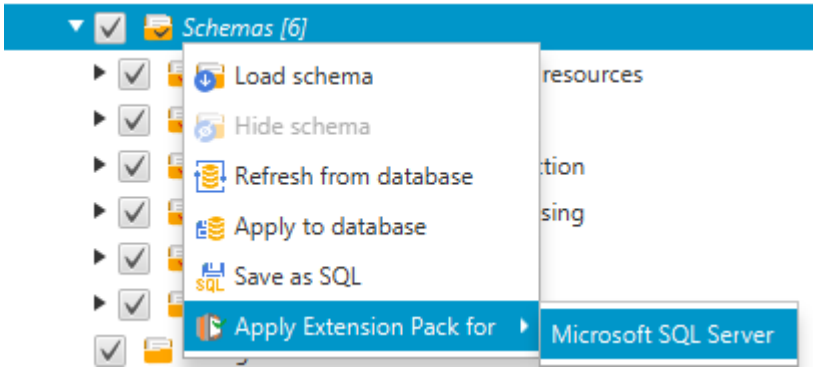
확장 팩을 적용하여 Lambda 함수 지원

확장 팩 마법사를 사용하거나 변환된 코드를 대상 데이터베이스에 적용할 때 확장 팩을 적용하여 Lambda 함수를 지원할 수 있습니다.

> **Important**
>
> AWS 서비스 에뮬레이션 기능은 Amazon EC2에 설치되고 자체 관리되는 데이터베이스에만 지원됩니다. 대상 데이터베이스가 Amazon RDS DB 인스턴스에 있는 경우에는 서비스 에뮬레이션 기능을 설치하지 않도록 합니다.

확장 팩 마법사를 사용하여 확장 팩을 적용하려면

1. 대상 데이터베이스 트리에서 컨텍스트 (마우스 오른쪽 버튼 클릭) 메뉴를 열고 확장 팩 적용을 선택한 다음 소스 데이터베이스 플랫폼을 선택합니다. AWS Schema Conversion Tool



확장 팩 마법사가 표시됩니다.

2. 시작 페이지를 읽은 후 다음을 선택합니다.
3. AWS profile settings 페이지에서 다음 작업을 수행합니다.

   - 확장 팩 스키마만 다시 설치하는 경우에는 Skip this step for now를 선택하고 다음을 선택합니다.
   - AWS 서비스를 설치하는 경우 연결에 필요한 자격 증명을 제공하십시오 AWS 계정. 자격 증명이 AWS CLI 설치되어 있으면 AWS CLI 자격 증명을 사용할 수 있습니다. 또한 이전에 전역 애플리케이션 설정의 프로필에 저장한 후 프로젝트와 연결한 보안 인증 정보를 사용할 수도 있습니다. 필요한 경우 Navigate to Project Settings를 선택하여 다른 프로필과 프로젝트를 연결합니다. 필요한 경우 전역 설정을 선택하여 새 프로필을 생성합니다. 자세한 정보는 AWS SCT에 AWS 서비스 프로필 저장을 참조하세요.
4. Email Sending Service 페이지에서 다음을 수행합니다.

   - 확장 팩 스키마만 다시 설치하는 경우에는 Skip this step for now를 선택하고 다음을 선택합니다.
   - AWS 서비스를 설치 중이고 기존 Lambda 함수가 있는 경우 이를 제공할 수 있습니다. 그렇지 않으면 마법사에서 자동으로 생성합니다. 완료되면 다음을 선택합니다.
5. Job Emulation Service 페이지에서 다음을 수행합니다.

   - 확장 팩 스키마만 다시 설치하는 경우에는 Skip this step for now를 선택하고 다음을 선택합니다.
   - AWS 서비스를 설치 중이고 기존 Lambda 함수가 있는 경우 이를 제공할 수 있습니다. 그렇지 않으면 마법사에서 자동으로 생성합니다. 완료되면 다음을 선택합니다.
6. Function emulation 페이지에서 Create extension pack을 선택합니다. 확장 팩 작업 상태가 포함된 메시지가 표시됩니다.

   모두 마쳤으면 완료를 선택합니다.

> **Note**
>
> 확장 팩을 업데이트하고 이전 확장 팩 구성 요소를 덮어쓰려면 최신 버전을 사용해야 합니다. AWS SCT자세한 정보는 설치, 확인 및 업데이트 AWS SCT을 참조하세요.

AWS SCT 확장 팩의 함수 구성

확장 팩에는 사용하기 전에 구성해야 하는 기능이 포함되어 있습니다. 상수는 서비스 팩이 사용하는 언어를 CONVERSION_LANG 정의합니다. 함수는 영어와 독일어로 사용할 수 있습니다.

언어를 영어 또는 독일어로 설정하려면 함수 코드를 다음과 같이 변경합니다. 다음 상수 선언을 찾아보십시오.

```
CONVERSION_LANG CONSTANT VARCHAR := '';
```

영어로 CONVERSION_LANG 설정하려면 줄을 다음과 같이 변경하십시오.

```
CONVERSION_LANG CONSTANT VARCHAR := 'English';
```

영어로 CONVERSION_LANG 설정하려면 줄을 다음과 같이 변경하십시오.

```
CONVERSION_LANG CONSTANT VARCHAR := 'Deutsch';
```

다음 기능에 대해 이 설정을 설정하십시오.

- aws_sqlserver_ext.conv_datetime_to_string
- aws_sqlserver_ext.conv_date_to_string
- aws_sqlserver_ext.conv_string_to_date
- aws_sqlserver_ext.conv_string_to_datetime
- aws_sqlserver_ext.conv_string_to_datetime
- aws_sqlserver_ext.parse_to_date
- aws_sqlserver_ext.parse_to_datetime
- aws_sqlserver_ext.parse_to_time

AWS SCT의 모범 사례

AWS Schema Conversion Tool(AWS SCT)을 사용하는 모범 사례 및 옵션에 대한 정보를 알아봅니다.

추가 메모리 구성

3,500개의 저장 프로시저가 있는 데이터베이스와 같은 대용량 데이터베이스 스키마를 변환하려면 AWS Schema Conversion Tool에서 사용할 수 있는 메모리 양을 구성할 수 있습니다.

AWS SCT에서 사용하는 메모리 양을 수정하려면

1. 설정 메뉴에서 전역 설정을 선택한 다음, JVM 옵션을 선택합니다.
2. Edit config file을 선택하고 텍스트 편집기를 선택하여 구성 파일을 엽니다.
3. `JavaOptions` 섹션을 편집하여 사용 가능한 최소 및 최대 메모리를 설정합니다. 다음 예제에서는 최소값을 4GB로, 최대값을 40GB로 설정합니다.

   ```
   [JavaOptions]
   -Xmx40960M
   -Xms4096M
   ```

   사용 가능한 최소 메모리를 4GB 이상으로 설정하는 것이 좋습니다.
4. 구성 파일을 저장하고 확인을 선택한 다음, AWS SCT를 다시 시작하여 변경 내용을 적용합니다.

기본 프로젝트 폴더 구성

AWS SCT에서는 프로젝트 폴더를 사용하여 프로젝트 파일, 평가 보고서 및 변환된 코드를 저장합니다. 기본적으로 AWS SCT는 모든 파일을 애플리케이션 폴더에 저장합니다. 다른 폴더를 기본 프로젝트 폴더로 지정할 수 있습니다.

기본 프로젝트 폴더를 변경하려면

1. 설정 메뉴에서 전역 설정을 선택한 다음, 파일 경로를 선택합니다.
2. Default project file path에 기본 프로젝트 폴더의 경로를 입력합니다.
3. [Apply]를 선택하고 [OK]를 선택합니다.

데이터 마이그레이션 속도 향상

1TB를 초과하는 데이터가 있는 테이블 세트와 같은 대규모 데이터 세트를 마이그레이션하려면 마이그레이션 속도를 높이는 것이 좋습니다. 데이터 추출 에이전트를 사용하는 경우 데이터 마이그레이션 속도는 다양한 요소에 따라 달라집니다. 이러한 요소에는 대상 Amazon Redshift 클러스터의 슬라이스 수, 마이그레이션 작업의 청크 파일 크기, 데이터 추출 에이전트를 실행하는 PC에서 사용할 수 있는 RAM 등이 포함됩니다.

데이터 마이그레이션 속도를 높이려면 프로덕션 데이터 중 작은 데이터 세트를 사용하여 여러 테스트 마이그레이션 세션을 실행하는 것이 좋습니다. 또한 크기가 500GB 이상인 SSD가 있는 PC에서 데이터 추출 에이전트를 실행하는 것이 좋습니다. 이러한 테스트 세션 중에 다른 마이그레이션 파라미터를 변경하고 디스크 사용률을 모니터링하여 최대 데이터 마이그레이션 속도를 보장하는 구성을 찾습니다. 그런 다음 이 구성을 사용하여 전체 데이터 세트를 마이그레이션합니다.

로깅 정보 증가

데이터베이스, 스크립트 및 애플리케이션 SQL을 변환할 때 AWS SCT에서 생성되는 로깅 정보를 증가시킬 수 있습니다. 로깅 정보가 증가하면 변환 속도가 느려질 수 있지만, 오류가 발생할 경우 변경 내용을 통해 AWS Support에 강력한 정보를 제공할 수 있습니다.

AWS SCT는 로컬 환경에 로그를 저장합니다. 이러한 로그 파일을 확인하고 문제 해결을 위해 AWS Support 또는 AWS SCT 개발자와 공유할 수 있습니다.

로깅 설정을 변경하려면

1. 설정 메뉴에서 전역 설정을 선택한 다음, 로깅을 선택합니다.
2. Log folder path에 사용자 인터페이스의 로그를 저장할 폴더를 입력합니다.
3. 콘솔 로그 폴더 경로에는 AWS SCT 명령줄 인터페이스(CLI)의 로그를 저장할 폴더를 입력합니다.
4. Maximum log file size (MB)에는 단일 로그 파일의 크기를 MB 단위로 입력합니다. 파일이 이 한도에 도달하면 AWS SCT에서 새 로그 파일이 생성됩니다.
5. Maximum number of log files에는 저장할 로그 파일 수를 입력합니다. 폴더의 로그 파일 수가 이 한도에 도달하면 AWS SCT가 가장 오래된 로그 파일을 삭제합니다.
6. Extractors log download path에 데이터 추출 에이전트 로그를 저장할 폴더를 입력합니다.
7. Cassandra extractor log path에 Apache Cassandra 데이터 추출 에이전트 로그를 저장할 폴더를 입력합니다.
8. 데이터 추출 에이전트를 사용할 때마다 AWS SCT에서 로그를 저장할 위치를 묻도록 하려면 Ask for a path before loading을 선택합니다.

9. 디버그 모드에서 True를 선택합니다. 표준 AWS SCT 로그에 문제가 없는 경우 추가 정보를 기록하려면 이 옵션을 사용합니다.
10. 로깅 정보를 증가시킬 주요 애플리케이션 모듈을 선택합니다. 다음 애플리케이션 모듈의 로깅 정보를 증가시킬 수 있습니다.

    - 일반
    - 로더
    - 파서
    - 프린터
    - 해석기
    - 원격 측정
    - 변환기
    - 유형 매핑
    - 사용자 인터페이스
    - 컨트롤러
    - 스키마 비교
    - 복제 데이터 센터
    - 애플리케이션 분석기

    위의 각 애플리케이션 모듈에 대해 다음 로깅 수준 중 하나를 선택합니다.

    - 추적 - 가장 자세한 정보입니다.
    - 디버그 - 시스템 전반의 흐름에 대한 세부 정보입니다.
    - 정보 - 시작 또는 종료와 같은 런타임 이벤트입니다.
    - 경고 - 더 이상 사용되지 않는 API의 사용, 잘못된 API 사용, 바람직하지 않거나 예상치 못한 기타 런타임 상황에 대한 정보입니다.
    - 오류 - 런타임 오류 또는 예상치 못한 상황에 대한 정보입니다.
    - 치명적 - 애플리케이션 종료로 이어지는 오류입니다.
    - 필수 - 가능한 가장 높은 수준의 오류입니다.

    기본적으로 디버그 모드를 켠 후 AWS SCT가 모든 애플리케이션 모듈에 대해 정보 로깅 수준을 설정합니다.

예를 들어 변환 중에 주요 문제 영역에서 도움이 필요하면 파서, 유형 매핑 및 사용자 인터페이스를 추적으로 설정합니다.

로그가 스트리밍되는 파일 시스템에서 정보가 너무 많아지는 경우에는 로그를 캡처하기에 충분한 공간이 있는 위치로 변경합니다.

AWS Support에 로그를 전송하려면 로그가 저장된 디렉터리로 이동하여 모든 파일을 관리 가능한 단일 .zip 파일로 압축합니다. 그런 다음 지원 사례와 함께 .zip 파일을 업로드합니다. 초기 분석이 완료되고 진행 중인 개발이 재개되면 디버그 모드를 false로 되돌려 상세 로깅을 제거합니다. 그런 다음 변환 속도를 높입니다.

> **Tip**
>
> 로그 크기를 관리하고 보고 문제를 간소화하려면 변환 성공 후 로그를 제거하거나 다른 위치로 이동합니다. 이 작업을 수행하면 관련 오류 및 정보만 AWS Support에 전송되므로 로그 파일 시스템이 가득 차는 것을 방지할 수 있습니다.

AWS SCT 문제 해결

다음에서는 AWS Schema Conversion Tool(AWS SCT) 관련 문제를 해결하는 방법에 대해 자세히 알아볼 수 있습니다.

Oracle 소스 데이터베이스에서 객체를 로드할 수 없습니다.

Oracle 데이터베이스에서 스키마를 로드하려고 하면 다음 오류 중 하나가 발생할 수 있습니다.

```
Cannot load objects tree.
```

```
ORA-00942: table or view does not exist
```

이러한 오류는 Oracle 데이터베이스에 연결하는 데 사용한 ID를 가진 사용자에게 AWS SCT에서 필요한 스키마를 읽을 수 있는 충분한 권한이 없기 때문에 발생합니다.

사용자에게 `select_catalog_role` 권한과 데이터베이스의 모든 사전에 대한 권한을 부여하여 문제를 해결할 수 있습니다. 이러한 권한은 AWS SCT에 필요한 보기 및 시스템 테이블에 대한 읽기 전용 액세스를 제공합니다. 다음 예제에서는 `min_privs`라는 사용자 ID를 생성하고 이 ID를 가진 사용자에게 Oracle 소스 데이터베이스의 스키마를 변환하는 데 필요한 최소 권한을 부여합니다.

```
create user min_privs identified by min_privs;
grant connect to min_privs;
grant select_catalog_role to min_privs;
grant select any dictionary to min_privs;
```

평가 보고서 경고 메시지

다른 데이터베이스 엔진으로 변환하는 작업의 복잡성을 평가하려면 AWS SCT가 소스 데이터베이스의 객체에 액세스할 수 있어야 합니다. 스캔 중에 AWS SCT에서 문제가 발생하여 평가를 수행할 수 없는 경우 경고 메시지가 표시됩니다. 이 메시지는 전체 변환율이 감소했음을 나타냅니다. 스캔 중에 AWS SCT에서 문제가 발생할 수 있는 이유는 다음과 같습니다.

- 데이터베이스 사용자가 필요한 모든 개체에 액세스할 수 없습니다. AWS SCT에 필요한 데이터베이스 보안 권한 및 권한 부여에 대한 자세한 내용은 이 설명서에서 해당 소스 데이터베이스의 AWS SCT 소스 섹션을 참조하세요.

- 스키마에 인용된 객체가 더 이상 데이터베이스에 존재하지 않습니다. 문제를 해결하는 데 도움이 되도록 SYSDBA 권한으로 연결한 후 해당 객체가 데이터베이스에 있는지 확인할 수 있습니다.
- SCT가 암호화된 객체를 평가하려고 합니다.

AWS SCT CLI 레퍼런스

이 섹션에서는 AWS SCT 명령줄 인터페이스 (CLI) 를 시작하는 방법을 설명합니다. 또한 이 섹션에서는 주요 명령 및 사용 모드에 대한 정보도 제공합니다. AWS SCT CLI 명령에 대한 전체 참조는 를 참조하십시오. 참조 자료

주제

- AWS SCT 명령줄 인터페이스를 사용하기 위한 사전 조건
- AWS SCT CLI 대화형 모드
- AWS SCT CLI 시나리오 가져오기
- AWS SCT CLI 시나리오 편집
- AWS SCT CLI 스크립트 모드
- AWS SCT CLI 참조 자료

AWS SCT 명령줄 인터페이스를 사용하기 위한 사전 조건

최신 버전의 Amazon Corretto 11을 다운로드하여 설치합니다. 자세한 내용은 Amazon Corretto 11 사용 설명서의 Amazon Corretto 11 다운로드를 참조하세요.

의 최신 버전을 다운로드하여 설치합니다. AWS SCT자세한 정보는 설치 AWS SCT을 참조하세요.

AWS SCT CLI 대화형 모드

AWS SCT 명령줄 인터페이스는 대화형 모드에서 사용할 수 있습니다. 이 모드에서는 콘솔에 명령을 하나씩 입력합니다. 이 대화형 모드를 사용하여 CLI 명령에 대해 자세히 알아보거나 가장 일반적으로 사용되는 CLI 시나리오를 다운로드할 수 있습니다.

원본 데이터베이스 스키마를 변환하려면 새 프로젝트 만들기 AWS SCT, 원본 및 대상 데이터베이스에 연결, 매핑 규칙 만들기, 데이터베이스 개체 변환 등의 시퀀스 작업을 실행합니다. 이 워크플로는 복잡할 수 있으므로 AWS SCT CLI 모드에서 스크립트를 사용하는 것이 좋습니다. 자세한 정보는 스크립트 모드을 참조하세요.

AWS SCT 설치 경로의 app 폴더에서 AWS SCT CLI 명령을 실행할 수 있습니다. Windows에서 기본 설치 경로는 `C:\Program Files\AWS Schema Conversion Tool\`입니다. 이 폴더에 `AWSSchemaConversionToolBatch.jar` 파일이 포함되어 있어야 합니다.

AWS SCT CLI 대화형 모드로 전환하려면 사전 요구 사항을 완료한 후 다음 명령을 사용하십시오.

```
java -jar AWSSchemaConversionToolBatch.jar -type interactive
```

이제 AWS SCT CLI 명령을 실행할 수 있습니다. 새로운 줄에 /로 명령을 끝내야 합니다. 또한 명령 파라미터 값 앞뒤에 곧은 작은따옴표(')를 사용해야 합니다.

> **Note**
>
> 앞의 명령이 Unexpected error를 반환하는 경우 다음을 수행합니다.
>
> ```
> java -Djdk.jar.maxSignatureFileSize=20000000 -jar
>  AWSSchemaConversionToolBatch.jar
> ```

AWS SCT CLI 대화형 모드에서 사용 가능한 명령 목록을 보려면 다음 명령을 실행합니다.

```
help
/
```

AWS SCT CLI 명령에 대한 정보를 보려면 다음 명령을 사용합니다.

```
help -command: 'command_name'
/
```

이전 예제에서 *command_name*을 명령 이름으로 바꿉니다.

AWS SCT CLI 명령의 파라미터에 대한 정보를 보려면 다음 명령을 사용합니다.

```
help -command: 'command_name' -parameters: 'parameters_list'
/
```

이전 예제에서 *command_name*을 명령 이름으로 바꿉니다. 그런 다음 *parameters_list*를 쉼표로 구분된 파라미터 이름 목록으로 바꿉니다.

AWS SCT CLI 대화형 모드에서 파일에서 스크립트를 실행하려면 다음 명령을 사용합니다.

```
ExecuteFile -file: 'file_path'
```

```
/
```

이전 예제에서 *file_path*를 스크립트가 있는 파일 경로로 바꿉니다. 파일 확장자는 .scts여야 합니다.

AWS SCT CLI 대화형 모드를 종료하려면 명령을 실행합니다. quit

예제

다음 예제는 Convert 명령에 대한 정보를 표시합니다.

```
help -command: 'Convert'
/
```

다음 예제는 Convert 명령의 두 파라미터에 대한 정보를 표시합니다.

```
help -command: 'Convert' -parameters: 'filter, treePath'
/
```

AWS SCT CLI 시나리오 가져오기

가장 일반적으로 사용되는 AWS SCT 시나리오를 가져오려면 GetCliScenario 명령을 사용할 수 있습니다. 대화형 모드에서 이 명령을 실행한 다음, 다운로드한 템플릿을 편집할 수 있습니다. 편집한 파일을 스크립트 모드에서 사용합니다.

GetCliScenario 명령은 선택한 템플릿 또는 사용 가능한 모든 템플릿을 지정된 디렉터리에 저장합니다. 템플릿에는 스크립트를 실행하기 위한 전체 명령 세트가 포함되어 있습니다. 이러한 템플릿에서 파일 경로, 데이터베이스 보안 인증 정보, 객체 이름 및 기타 데이터를 편집해야 합니다. 또한 사용하지 않는 명령은 제거하고 필요한 경우 스크립트에 새 명령을 추가해야 합니다.

GetCliScenario 명령을 실행하려면 사전 조건을 완료하고 AWS SCT CLI 대화형 모드로 전환합니다. 자세한 정보는 대화형 모드을 참조하세요.

그런 다음, 다음 구문을 사용하여 GetCliScenario 명령을 실행하고 AWS SCT 시나리오를 가져옵니다.

```
GetCliScenario -type: 'template_type' -directory: 'file_path'
/
```

이전 예제에서 *template_type*을 다음 표의 템플릿 유형 중 하나로 바꿉니다. 그런 다음, 스크립트를 다운로드하려는 폴더 경로로 *file_path*를 바꿉니다. 관리자 권한을 요청하지 않고도 이 폴더에 액세스할 AWS SCT 수 있는지 확인하세요. 또한 명령 파라미터 값 앞뒤에 곧은 작은따옴표(')를 사용해야 합니다.

모든 AWS SCT CLI 템플릿을 다운로드하려면 위의 명령을 옵션 없이 실행합니다. `-type`

다음 표에는 다운로드할 수 있는 AWS SCT CLI 템플릿 유형이 나와 있습니다. 각 템플릿에 대한 테이블에는 파일 이름과 스크립트를 사용하여 실행할 수 있는 작업에 대한 설명이 포함됩니다.

템플릿 유형	파일 이름	설명
BTEQ ScriptConversion	`BTEQScriptConversionTemplate.scts`	테라데이타 기본 테라데이타 쿼리 (BTEQ) FastExport FastLoad, 및 스크립트를 MultiLoad Amazon Redshift RSQL로 변환합니다. 자세한 정보는 ETL 프로세스 변환을 참조하세요.
ConversionApply	`ConversionTemplate.scts`	소스 데이터베이스 스키마를 변환하고 변환된 코드를 대상 데이터베이스에 적용합니다. 변환된 코드를 SQL 스크립트로 저장하고 평가 보고서를 저장할 수도 있습니다. 자세한 정보는 데이터베이스 스키마 변환을 참조하세요.
GenericAppConversion	`GenericApplicationConversionTemplate.scts`	일반 애플리케이션 변환기를 사용하여 애플리케이션에 내장된 SQL 코드를 변환합니다. AWS SCT 자세한 정보는 애플리케이션의 SQL 코드 변환을 참조하세요.
HadoopMigration	`HadoopMigrationTemplate.scts`	온프레미스 Hadoop 클러스터를 Amazon EMR로 마이그

템플릿 유형	파일 이름	설명
		레이션합니다. 자세한 정보는 Apache Hadoop을 AWS SCT에 대한 소스로 사용을 참조하세요.
HadoopResumeMigration	`HadoopResumeMigrationTemplate.scts`	온프레미스 Hadoop 클러스터를 Amazon EMR로의 중단된 마이그레이션을 재개합니다. 자세한 정보는 Apache Hadoop을 AWS SCT에 대한 소스로 사용을 참조하세요.
Informatica	`InformaticaConversionTemplate.scts`	Informatica 추출, 전환, 적재(ETL) 스크립트에 포함된 SQL 코드를 변환합니다. ETL 스크립트에서 소스 및 대상 데이터베이스에 대한 연결을 구성한 다음, 변환 후 변환된 스크립트를 저장합니다. 자세한 정보는 Informatica ETL 스크립트 변환을 참조하세요.
LanguageSpecificAppConversion	`LanguageSpecificAppConversionTemplate.scts`	AWS SCT 애플리케이션 변환기를 사용하여 C#, C++, Java 및 Pro*C 애플리케이션에 포함된 SQL 코드를 변환합니다. 자세한 정보는 애플리케이션 SQL 변환을 참조하세요.
OozieConversion	`OozieConversionTemplate.scts`	Apache Oozie 워크플로를 로 변환합니다. AWS Step Functions자세한 정보는 Apache Oozie를 AWS SCT에 대한 소스로 사용을 참조하세요.

템플릿 유형	파일 이름	설명
RedshiftAgent	`DWHDataMigrationTemplate.scts`	소스 데이터 웨어하우스 스키마를 변환하고 변환된 코드를 대상 Amazon Redshift 데이터베이스에 적용합니다. 그런 다음, 데이터 추출 에이전트를 등록하고 데이터 마이그레이션 작업을 생성한 후 시작합니다. 자세한 정보는 데이터 웨어하우스에서 Amazon Redshift로 마이그레이션을 참조하세요.
ReportCreation	`ReportCreationTemplate.scts`	여러 소스 데이터베이스 스키마에 대한 데이터베이스 마이그레이션 보고서를 생성합니다. 그런 다음, 이 보고서를 PDF 파일의 CSV로 저장합니다. 자세한 정보는 마이그레이션 평가 보고서을 참조하세요.
SQL ScriptConversion	`SQLScriptConversionTemplate.scts`	SQL*Plus 또는 TSQL 스크립트를 PL/SQL로 변환하고 변환된 스크립트를 저장합니다. 또한 평가 보고서도 저장합니다.

AWS SCT CLI 템플릿을 다운로드한 후 텍스트 편집기를 사용하여 소스 및 대상 데이터베이스에서 실행되도록 스크립트를 구성합니다. 그런 다음 AWS SCT CLI 스크립트 모드를 사용하여 스크립트를 실행합니다. 자세한 정보는 AWS SCT CLI 스크립트 모드을 참조하세요.

예제

다음 예제는 모든 템플릿을 `C:\SCT\Templates` 폴더에 다운로드합니다.

```
GetCliScenario -directory: 'C:\SCT\Templates'
/
```

다음 예제에서는 ConversionApply 작업에 필요한 템플릿을 C:\SCT\Templates 폴더에 다운로드합니다.

```
GetCliScenario -type: 'ConversionApply' -directory: 'C:\SCT\Templates'
/
```

AWS SCT CLI 시나리오 편집

시나리오 템플릿을 다운로드한 후 데이터베이스에서 실행할 수 있는 작업 스크립트를 가져오도록 구성합니다.

모든 템플릿에 대해 소스 및 대상 데이터베이스의 드라이버 경로를 제공해야 합니다. 자세한 정보는 필수 데이터베이스 드라이버 다운로드을 참조하세요.

소스 및 대상 데이터베이스의 데이터베이스 보안 인증 정보를 포함해야 합니다. 또한 변환 프로젝트의 소스-대상 페어를 설명하는 매핑 규칙을 설정해야 합니다. 자세한 정보는 매핑 규칙 생성을 참조하세요.

다음으로, 실행할 작업의 범위를 구성합니다. 사용하지 않는 명령을 제거하거나 새 명령을 스크립트에 추가할 수 있습니다.

예를 들어, 소스 Oracle 데이터베이스의 모든 스키마를 PostgreSQL로 변환할 계획이라고 가정해 보겠습니다. 데이터베이스 마이그레이션 평가 보고서를 PDF로 저장하고 변환된 코드를 대상 데이터베이스에 적용하도록 계획할 수 있습니다. 이 경우 ConversionApply 작업에 템플릿을 사용할 수 있습니다. 다음 절차를 사용하여 AWS SCT CLI 템플릿을 편집하십시오.

작업에 대한 AWS SCT CLI 템플릿을 편집하려면 **ConversionApply**

1. 다운로드한 ConversionTemplate.scts를 엽니다. 자세한 정보는 예제을 참조하세요.
2. 스크립트에서 -filter CreateFilter, -filter, ApplyToTarget SaveTargetSQL, SaveTargetSQ 및 SaveReportCSV LbyStatement 작업을 제거하고 변환합니다.
3. 작업 중인 oracle_driver_file의 경우 오라클 드라이버의 경로를 입력하십시오. SetGlobalSettings 그런 다음, postgresql_driver_file에 PostgreSQL 드라이버의 경로를 입력합니다.

   다른 데이터베이스 엔진을 사용하는 경우에는 설정에 해당 이름을 사용합니다. SetGlobalSettings작업에서 설정할 수 있는 글로벌 설정의 전체 목록은 의 글로벌 설정 매트릭스를 참조하십시오. 참조 자료

4. (선택 사항 CreateProject) 에는 프로젝트 이름과 로컬 프로젝트 파일 위치를 입력합니다. 기본값을 사용하여 작업을 진행하는 경우 AWS SCT 는 관리자 권한을 요청하지 않고도 `C:\temp` 폴더에 파일을 생성할 수 있어야 합니다.
5. AddSource에 원본 데이터베이스 서버의 IP 주소를 입력합니다. 또한 소스 데이터베이스 서버에 연결할 사용자 이름, 암호 및 포트를 입력합니다.
6. AddTarget에 대상 데이터베이스 서버의 IP 주소를 입력합니다. 또한 대상 데이터베이스 서버에 연결할 사용자 이름, 암호 및 포트를 입력합니다.
7. (선택 사항 AddServerMapping) 매핑 규칙에 추가할 원본 및 대상 데이터베이스 개체를 입력합니다. `sourceTreePath` 및 `targetTreePath` 파라미터를 사용하여 데이터베이스 객체의 경로를 지정할 수 있습니다. 선택적으로 `sourceNamePath` 및 `targetNamePath`를 사용하여 데이터베이스 객체의 이름을 지정할 수 있습니다. 자세한 내용은 참조 자료에서 서버 매핑 명령을 참조하세요.

   AddServerMapping작업의 기본값은 모든 소스 스키마를 대상 데이터베이스에 매핑합니다.
8. 파일을 저장한 다음, 스크립트 모드를 사용하여 해당 파일을 실행합니다. 자세한 정보는 스크립트 모드을 참조하세요.

AWS SCT CLI 스크립트 모드

AWS SCT CLI 스크립트를 생성하거나 템플릿을 편집한 후 명령을 사용하여 실행할 수 있습니다. `RunSCTBatch` CLI 스크립트를 사용하여 파일을 `.scts` 확장자로 저장해야 합니다.

AWS SCT 설치 경로의 app 폴더에서 AWS SCT CLI 스크립트를 실행할 수 있습니다. Windows에서 기본 설치 경로는 `C:\Program Files\AWS Schema Conversion Tool\`입니다. 이 폴더에 `RunSCTBatch.cmd` 또는 `RunSCTBatch.sh` 파일이 포함되어 있어야 합니다. 또한 이 폴더에는 `AWSSchemaConversionToolBatch.jar` 파일이 포함되어야 합니다.

운영 체제의 `PATH` 환경 변수에 `RunSCTBatch` 파일의 경로를 추가할 수도 있습니다. `PATH`환경 변수를 업데이트한 후 모든 폴더에서 AWS SCT CLI 스크립트를 실행할 수 있습니다.

AWS SCT CLI 스크립트를 실행하려면 Windows에서 다음 명령을 사용하십시오.

```
RunSCTBatch.cmd --pathtoscts "file_path"
```

이전 예제에서 *`file_path`*를 스크립트가 있는 파일 경로로 바꿉니다.

AWS SCT CLI 스크립트를 실행하려면 Linux에서 다음 명령을 사용합니다.

```
RunSCTBatch.sh --pathtoscts "file_path"
```

이전 예제에서 *file_path*를 스크립트가 있는 파일 경로로 바꿉니다.

이 명령에 데이터베이스 보안 인증 정보, 콘솔 출력의 세부 정보 수준 등과 같은 선택적 파라미터를 제공할 수 있습니다. 자세한 내용은 에서 AWS SCT 명령줄 인터페이스 참조를 다운로드하십시오. 참조 자료

예제

다음 예제는 `C:\SCT\Templates` 폴더에서 `ConversionTemplate.scts` 스크립트를 실행합니다. Windows에서 이 예제를 사용할 수 있습니다.

```
RunSCTBatch.cmd --pathtoscts "C:\SCT\Templates\ConversionTemplate.scts"
```

다음 예제는 `/home/user/SCT/Templates` 디렉터리에서 `ConversionTemplate.scts` 스크립트를 실행합니다. Linux에서 이 예제를 사용할 수 있습니다.

```
RunSCTBatch.sh --pathtoscts "/home/user/SCT/Templates/ConversionTemplate.scts"
```

AWS SCT CLI 참조 자료

AWS Schema Conversion Tool 명령줄 인터페이스 (CLI) 에 대한 참조 자료는 다음 가이드에서 찾을 수 있습니다. CLI AWS Schema Conversion Tool 참조.

에 대한 릴리스 노트 AWS SCT

이 섹션에는 버전 AWS SCT 1.0.640부터 시작하는 릴리스 노트가 포함되어 있습니다.

빌드 676의 릴리스 노트 AWS SCT

소스	대상	새로운 기능, 향상된 기능 또는 수정된 기능	AWS DMS 스키마 변환 도구 (SCT) 의 가용성	AWS DMS 스키마 변환의 가용성
Oracle	PostgreSQL/Aurora PostgreSQL	다음 함수를 위한 새로운 내장 함수 에뮬레이션: • `SYS.UTL_RAW.BIT_AND(RAW,RAW)` • `XDB.DBMS_XSLPROCESSOR.CLOB2FILE(CLOB)` • `XDB.DBMS_XSLPROCESSOR.READ2CLOB(VARCHAR2)` • `SYS.UTL_RAW.BIT_OR(RAW,RAW)` • `SYS.UTL_RAW.BIT_COMPLEMENT(RAW)`	아니요	예
MS SQL Server	아마존 RDS SQL 서버	PDF `Database Mail not supported` 보고서에서 메시지가 제거되었습니다.	예	예
Oracle	PostgreSQL/Aurora PostgreSQL	파티션을 나눈 테이블에 대한 제약 조건 변환을 구현했습니다.	예	예

소스	대상	새로운 기능, 향상된 기능 또는 수정된 기능	AWS DMS 스키마 변환 도구 (SCT) 의 가용성	AWS DMS 스키마 변환의 가용성
Oracle	MySQL	테이블 변환의 AI-602 적용 가능성 검토	예	예
MS SQL Server	PostgreL/ Aurora PostgreL	이제 MERGE PostgreSQL 15.x의 명령문을 지원합니다.	예	예
모두	모두	구현된 JDBC 연결: 고급 속성	예	아니요
모두	모두	CLI: 명령 실패 수정 PrintOLAP TaskStatus	예	아니요
Teradat	Amazor Redshif	Teradata 스타일 데이터 유형 캐스트를 구현했습니다.	예	아니요
Teradat	Amazor Redshif	SQL/BTEQ에서 잘못된 변환이 수정되었습니다MERGE.	예	아니요
Teradat	Amazor Redshif	테라데이터 스타일 데이터 타입 캐스트를 구현했습니다.	예	아니요
Teradat	Amazor Redshif	함수 변환을 구현했습니다. LEAD/LAG	예	아니요
Teradat	Amazor Redshif	오류가 AI-9996 Transformer error occurred in statement 수정되었습니다.	예	아니요
Teradat	Amazor Redshif	오류가 수정되었습니다AI-9996 Transformer error in selectItem .	예	아니요

소스	대상	새로운 기능, 향상된 기능 또는 수정된 기능	AWS DMS 스키마 변환 도구 (SCT) 의 가용성	AWS DMS 스키마 변환의 가용성
Teradat	Amazor Redshif	부분 저장 프로시저에 대한 변환 구현: XbiDQM.SpCmprsnDly	예	아니요
Teradat	Amazor Redshif	별칭이 포함된 UNPIVOT 명령문을 구현했습니다.	예	아니요
Teradat	Amazor Redshif	여러 소스 테이블이 포함된 Delete 명령문을 구현했습니다.	예	아니요
Teradat	Amazor Redshif	에 대한 수정AI-9996 Transformer error occurred in functionCallExpression .	예	아니요
Teradat	Amazor Redshif	NORMALIZE 조항 변환을 구현했습니다.	예	아니요
Teradat	Amazor Redshif	하위 쿼리가 있는 DELETE 명령문의 잘못된 변환이 수정되었습니다.	예	아니요
Teradat	Amazor Redshif	오류가 AI-9996 Transformer error occurred in tableOperatorSource 수정되었습니다.	예	아니요
Teradat	Amazor Redshif	오류가 수정되었습니다AI-9996 Transformer error occurred in additiveExpression .	예	아니요
Teradat	Amazor Redshif	DBC 시스템 객체 변환을 구현했습니다.	예	아니요
Teradat	Amazor Redshif	암시적 조인 조건자를 사용한 업데이트의 해결 방법을 구현했습니다.	예	아니요

소스	대상	새로운 기능, 향상된 기능 또는 수정된 기능	AWS DMS 스키마 변환 도구 (SCT) 의 가용성	AWS DMS 스키마 변환의 가용성
Netezza	Amazon Redshift	CREATE MATERIALIZED VIEW 명령문 변환 오류가 수정되었습니다.	예	아니요
DB2Luw	PostgreSQL/ Aurora PostgreSQL	JDBC 확장 옵션 연결: 추가 연결 옵션이 추가되었습니다.	예	아니요
DB2Luw	PostgreSQL/ Aurora PostgreSQL	PostgreSQL 15.x에서 MERGE 스테이트먼트에 대한 지원이 추가되었습니다.	예	아니요
DB2Luw	PostgreSQL/ Aurora PostgreSQL	변환을 구현했습니다GLOBAL TEMPORARY TABLE.	예	아니요
DB2Luw	PostgreSQL/ Aurora PostgreSQL	변환을 구현했습니다USER DEFINED TYPES.	예	아니요
DB2Luw	MySQL	변환을 구현했습니다. GLOBAL TEMPORARY TABLE	예	아니요
DB2Luw	MySQL	변환을 구현했습니다. USER DEFINED TYPES	예	아니요

소스	대상	새로운 기능, 향상된 기능 또는 수정된 기능	AWS DMS 스키마 변환 도구 (SCT) 의 가용성	AWS DMS 스키마 변환의 가용성
DB2Luv	MySQL	변환을 구현했습니다. `USER DEFINED FUNCTIONS`	예	아니요
DB2Luv	MariaDB	변환을 구현했습니다. `GLOBAL TEMPORARY TABLE`	예	아니요
DB2Luv	MariaDB	변환을 구현했습니다. `USER DEFINED TYPES`	예	아니요
Sybase	모두	Kerberos 인증에 대한 지원이 추가되었습니다.	예	아니요
DB2Luv	PostgreSQL/ Aurora PostgreSQL	타겟에 대한 다중 버전 변환에 대한 지원이 추가되었습니다.	예	아니요
Azure SQL/ Microso SQL Server	PostgreSQL/ Aurora PostgreSQL	타겟에 대한 다중 버전 변환에 대한 지원이 추가되었습니다.	예	아니요
DB2Luv	PostgreSQL/ Aurora PostgreSQL	`MERGE`PostgreSQL 15.x에서 명령문에 대한 지원이 추가되었습니다.	예	아니요
Teradat	Amazor Redshif	지원되지 않는 함수 변경 변환이 수정되었습니다.	예	아니요

소스	대상	새로운 기능, 향상된 기능 또는 수정된 기능	AWS DMS 스키마 변환 도구 (SCT) 의 가용성	AWS DMS 스키마 변환의 가용성
모두	Amazor Redshif	데이터 추출기: 인덱싱된 열을 기준으로 파티셔닝을 구현했습니다.	예	아니요

빌드 675의 출시 노트 AWS SCT

소스	대상	새로운 기능, 향상된 기능 또는 수정된 기능	AWS DMS 스키마 변환의 가용성
Cassandra	DynamoD	대상 데이터 센터에서 Cassandra 설치가 실패하는 버그를 수정했습니다.	아니요
DB2 LUW	PostgreSQ L	동적 SQL: 준비 명령문: 동적 SQL을 사용하지 않는 해결 및 변환.	아니요
DB2 LUW	PostgreSQ L	특수 레지스터에 대한 지원이 추가되었습니다.	아니요
DB2 LUW	PostgreSQ L	익스텐션 팩 업데이트	아니요
Hadoop	Amazon EMR	rsa-sha2 프로토콜을 통한 하둡 클러스터 연결 지원이 추가되었습니다.	아니요
Microsoft SQL Server	Amazon Redshift	JDBC 드라이버가 구성되지 않았는데도 TLS를 강제 실행하는 문제를 수정했습니다.	아니요
Netezza	Amazon Redshift	구체화된 뷰 변환에 대한 지원이 추가되었습니다.	아니요
Oracle	Amazon Redshift	Amazon Redshift에 재귀 쿼리에 대한 지원이 추가되었습니다.	예

소스	대상	새로운 기능, 향상된 기능 또는 수정된 기능	AWS DMS 스키마 변환의 가용성
Oracle	PostgreSQL, Aurora PostgreSQL	NUMBER 데이터 유형이 잘못 변환되던 문제를 수정했습니다.	예
Oracle	Amazon Redshift	데이터 마이그레이션. 오라클 자동 파티셔닝. 테이블 프래그먼트 값의 만료 시간을 추가했습니다. 만료 시간은 72시간입니다. 만료가 발생하면 데이터 마이그레이션 작업이 생성될 때 데이터 조각이 재구축됩니다.	아니요
Oracle	Amazon Redshift	SCT 데이터 추출기: Amazon Redshift에 데이터를 업로드하는 방식을 변경했습니다. 기본적으로 추출기는 스테이징된 테이블을 생성하지 않습니다. 대신 모든 데이터 파일이 Amazon S3 버킷에 저장되면 추출기는 단일 COPY 명령을 사용하여 대상 테이블에 데이터를 복사합니다.	아니요
Oracle	Amazon Redshift	원시 데이터 유형을 VARBYTE 열로 마이그레이션하는 기능이 추가되었습니다.	아니요
Oracle	PostgreSQL, Aurora PostgreSQL	멀티 버전 변환	아니요
Oracle	PostgreSQL	PostgreSQL 15.x에 병합 명령문에 대한 지원이 추가되었습니다.	예
Oracle	PostgreSQL	PostgreSQL 15.x에 새로운 정규 표현식 함수에 대한 지원이 추가되었습니다.	예

소스	대상	새로운 기능, 향상된 기능 또는 수정된 기능	AWS DMS 스키마 변환의 가용성
Oracle	PostgreSQL, Aurora PostgreSQL	충돌 시 DO UPDATE 명령문은 제외된 별칭 없이 변환됩니다.	예
Teradata	Amazon Redshift	LEAD/LAG 함수에 대한 변환 지원이 추가되었습니다.	아니요
Teradata	Amazon Redshift	데이터 형식을 명시적으로 표시하여 데이터 유형 캐스팅을 개선했습니다.	아니요
Teradata	Amazon Redshift	타임/타임스탬프 표현식의 AT 'TIME ZONE' 절 변환이 개선되었습니다.	아니요
Teradata	Amazon Redshift	MERGE 문을 사용한 변환 프로시저 진행 시 AI-9996 발생	아니요

AWS SCT 빌드 674의 릴리스 노트

소스	대상	새로운 기능, 향상된 기능 또는 수정된 기능	AWS DMS 스키마 변환의 가용성
모두	모두	다양한 버그 수정 및 성능 향상	부분(지원되는 소스 및 대상 페어에만 해당)
Azure SQL/ Microsoft SQL Server	Amazon Redshift	스키마 평가/변환 중 사용자에게 혼동을 주는 “AI 18066: Unable to convert schema name” 메시지를 제거했습니다.	아니요

소스	대상	새로운 기능, 향상된 기능 또는 수정된 기능	AWS DMS 스키마 변환의 가용성
Azure SQL/ Microsoft SQL Server	Amazon RDS for MySQL / Amazon Aurora MySQL	반환 코드를 할당하지 않은 상태에서 잘못된 프로시저 변환	부분(스키마 변환은 현재 Azure SQL을 소스로 지원하지 않음)
Azure SQL/ Microsoft SQL Server	Amazon RDS for PostgreSQL / Amazon Aurora PostgreSQL	FOR XML PATH 절 변환의 일부 경우에 대한 AI9997 문제 수정	부분(스키마 변환은 현재 Azure SQL을 소스로 지원하지 않음)
Azure SQL/ Microsoft SQL Server	Amazon RDS for PostgreSQL / Amazon Aurora PostgreSQL	프로시저/함수 본문에서 값이 원래 배율로 반올림됨	부분(스키마 변환은 현재 Azure SQL을 소스로 지원하지 않음)
Azure SQL/ Microsoft SQL Server	Amazon RDS for PostgreSQL / Amazon Aurora PostgreSQL	EXECUTE 문 변환에 대한 다양한 개선 사항	부분(스키마 변환은 현재 Azure SQL을 소스로 지원하지 않음)

소스	대상	새로운 기능, 향상된 기능 또는 수정된 기능	AWS DMS 스키마 변환의 가용성
Azure SQL/ Microsoft SQL Server/ Azure Synapse	Amazon Redshift	다음 명령문 및 모드의 변환을 개선했습니다. • EXCEPTION BLOCK • AUTOCOMMIT • NONATOMIC • GROUPING SET • CUBE • ROLLUP	아니요
DB2 LUW	Amazon RDS for PostgreSQL / Amazon Aurora PostgreSQL	메타데이터 로드 SQL 쿼리에서 다양한 수정	아니요
DB2 LUW	Amazon RDS for PostgreSQL / Amazon Aurora PostgreSQL	트리거에서는 AI 9996이 예상되지 않습니다.	아니요

소스	대상	새로운 기능, 향상된 기능 또는 수정된 기능	AWS DMS 스키마 변환의 가용성
DB2 z/OS	Amazon RDS for PostgreSQL / Amazon Aurora PostgreSQL	ROWNUMBER 분석 함수	아니요
DB2 z/OS	Amazon RDS for PostgreSQL / Amazon Aurora PostgreSQL	16진 문자열 상수 지원	아니요
DB2 z/OS	Amazon RDS for PostgreSQL / Amazon Aurora PostgreSQL	메타데이터 로드 SQL 쿼리에서 다양한 수정	아니요

소스	대상	새로운 기능, 향상된 기능 또는 수정된 기능	AWS DMS 스키마 변환의 가용성
DB2 z/OS	Amazon RDS for PostgreSQL / Amazon Aurora PostgreSQL	NEXT VALUE FOR 시퀀스 참조 지원	아니요
DB2 z/OS	Amazon RDS for PostgreSQL / Amazon Aurora PostgreSQL	GET DIAGNOSTICS 문 DB2_NUMBER_ROWS 옵션 지원	아니요
DB2 z/OS	Amazon RDS for PostgreSQL / Amazon Aurora PostgreSQL	GET DIAGNOSTICS의 여러 명령문	아니요

소스	대상	새로운 기능, 향상된 기능 또는 수정된 기능	AWS DMS 스키마 변환의 가용성
DB2 z/OS	Amazon RDS for PostgreSQL / Amazon Aurora PostgreSQL	FOR 문 변환의 버그 수정됨	아니요
Oracle	Amazon RDS for MySQL / Amazon Aurora MySQL	패키지 함수의 파라미터 노드가 정의되지 않은 경우의 버그 수정	예
Oracle	Amazon RDS for PostgreSQL / Amazon Aurora PostgreSQL	확장 팩의 AWS_ORACLE_EXT.NEXT_DAY 함수에서 버그 수정	예
Oracle	Amazon RDS for PostgreSQL / Amazon Aurora PostgreSQL	Oracle의 외부 조인에서 “(+)” 변환과 관련된 여러 버그 수정	예

소스	대상	새로운 기능, 향상된 기능 또는 수정된 기능	AWS DMS 스키마 변환의 가용성
Oracle		Kerberos 인증 지원	아니요
SAP ASE	Amazon RDS for PostgreSQL / Amazon Aurora PostgreSQL	UPDATE 문의 FROM 절에서 둘 이상의 식별자를 변환할 때 발생하는 버그 수정	아니요
SAP ASE	Amazon RDS for PostgreSQL / Amazon Aurora PostgreSQL	여러 줄로 된 주석 및 명령문 변환 관련 버그 수정	아니요
SAP ASE		연결 시 ENCRYPT_PASSWORD 파라미터에 대한 지원 추가	아니요
Teradata	Amazon Redshift	지정된 스키마 이름을 사용하는 VOLATILE 테이블의 변환 기능 개선	아니요
Teradata	Amazon Redshift	복잡한 CTE의 잘못된 WHERE CLAUSE 변환	아니요
Teradata	Amazon Redshift	SCT 데이터 추출 에이전트를 사용하여 데이터를 마이그레이션할 때 INTERVAL 데이터 형식에 대한 지원을 추가했습니다.	아니요

소스	대상	새로운 기능, 향상된 기능 또는 수정된 기능	AWS DMS 스키마 변환의 가용성
Teradata BTEQ 스크립트	Amazon Redshift RSQL 스크립트	BTEQ에서 실행한 프로시저의 잘못된 파라미터 변환	아니요

AWS SCT 빌드 673의 릴리스 노트

소스	대상	새로운 기능, 향상된 기능 또는 수정된 기능
모두	모두	일반 버그 수정 및 성능 개선
Azure SQL/ Microsoft SQL Server	Aurora PostgreSQL/ Amazon RDS PostgreSQL	잘못된 함수 호출 변환 문제 해결
Azure SQL/ Microsoft SQL Server	Aurora PostgreSQL/ Amazon RDS PostgreSQL	FOR XML 절의 변환을 구현했습니다.
Azure SQL/ Microsoft SQL Server	Aurora PostgreSQL/ Amazon RDS PostgreSQL	잘못된 별칭이 있는 FOR XML 절의 변환

소스	대상	새로운 기능, 향상된 기능 또는 수정된 기능
Azure SQL/ Microsoft SQL Server	Aurora PostgreSQL/ Amazon RDS PostgreSQL	프로시저 매개 변수와 함께 문자열을 실행하는 EXECUTE 명령문을 변환하지 AWS SCT 않는 경우의 버그가 수정되었습니다.
Azure SQL/ Microsoft SQL Server	Aurora PostgreSQL/ Amazon RDS PostgreSQL	내부 조인을 사용한 UPDATE 문의 변환을 개선했습니다.
Azure Synapse	Amazon Redshift	OBJECT_ID 내장 함수의 잘못된 변환을 수정했습니다.
IBM DB2 for z/OS	Aurora PostgreSQL/ Amazon RDS PostgreSQL	다음 명령문 및 객체의 변환을 구현했습니다. • DECLARE TEMPORARY TABLE statement • DROP TABLE statement • 파티션 테이블의 PK 및 UNIQUE 제약 조건 • TIMESTAMPDIFF 함수 • TO_DATE 함수 • EBCDIC_STR 함수 • VARCHAR_FORMAT 함수
IBM DB2 for z/OS	Aurora PostgreSQL/ Amazon RDS PostgreSQL	변환 후 함수 기반 인덱스가 함수를 건너뛰는 버그를 수정했습니다.

소스	대상	새로운 기능, 향상된 기능 또는 수정된 기능
IBM DB2 for z/OS	Aurora PostgreSQL/ Amazon RDS PostgreSQL	변환 후 REPEAT 문이 AI 9996으로 종료되는 버그를 수정했습니다.
IBM DB2 for z/OS	Aurora PostgreSQL/ Amazon RDS PostgreSQL	FINAL TABLE 절이 9996으로 종료되는 버그를 수정했습니다.
IBM DB2 for z/OS	Aurora PostgreSQL/ Amazon RDS PostgreSQL	LOADER | 참조 제약 조건의 파티셔닝 키. AWS SCT 는 이제 파티션 테이블의 프라이머리 키와 고유 제약 조건을 보조 인덱스로 변환할 수 있습니다.
IBM DB2 for z/OS	Aurora PostgreSQL/ Amazon RDS PostgreSQL	PostgreSQL.VARCHAR_FORMAT 함수 지원
IBM DB2 for z/OS	Aurora PostgreSQL/ Amazon RDS PostgreSQL	CreateTransformationRule 및 ModifyTransformationRule SCT CLI 명령에서 데이터 정렬 변경을 구현했습니다.

소스	대상	새로운 기능, 향상된 기능 또는 수정된 기능
Greenplum	Amazon Redshift	변환 후 저장 프로시저가 잘못 호출되는 버그를 수정했습니다.
Hadoop	Amazon EMR	rsa-sha2 프로토콜을 사용하여 Hadoop 클러스터에 연결하는 지원을 추가했습니다.
Hadoop	Amazon EMR	비 Glue Hive 메타스토어가 있는 Amazon EMR에 대한 지원을 추가했습니다.
Oracle	Amazon Redshift	PRIOR 열이 SELECT 목록에 없는 재귀 쿼리가 잘못 변환되는 버그를 수정했습니다.
Oracle	Aurora PostgreSQL/ Amazon RDS PostgreSQL	연관 배열의 요소를 반환하도록 구현했습니다.
Oracle	Aurora PostgreSQL/ Amazon RDS PostgreSQL	괄호를 사용하는 UNPIVOT에서 예기치 않은 AI 9996 수정
Oracle	Aurora PostgreSQL/ Amazon RDS PostgreSQL	UNION ALL을 사용하는 UNPIVOT에서 예기치 않은 AI 9996 수정

소스	대상	새로운 기능, 향상된 기능 또는 수정된 기능
Oracle	Aurora PostgreSQL/ Amazon RDS PostgreSQL	`Number` 데이터 형식 변환에 대한 개선 사항
Oracle	Amazon Redshift 데이터 추출기	Oracle 테이블의 자동 파티셔닝 지원. 마이그레이션 작업 생성을 위한 최적화
Teradata	Amazon Redshift	`EXCEPTION BLOCK` 문에 대한 변환 구현
Teradata	Amazon Redshift	`ALL`, `ANY` 및 `SOME` 조건자를 Amazon Redshift로 변환하는 기능이 지원됩니다.
Teradata	Amazon Redshift	`QUALIFY` 조건자에 대한 기본 지원을 추가했습니다.
Teradata	Amazon Redshift	다음에 대한 변환을 개선했습니다. • 재귀 쿼리 • `GROUPING SET` • `CUBE` • `ROLLUP` • 암시적 조인이 포함된 `UPDATE` 문
OLAP 소스	Amazon Redshift 데이터 추출기	Amazon Redshift 데이터 추출기 작업의 중지/재개를 위한 CLI 명령을 구현했습니다.

소스	대상	새로운 기능, 향상된 기능 또는 수정된 기능
OLAP 소스	Amazon Redshift 데이터 추출기	마이그레이션 작업을 구성하는 동안 마이그레이션해야 하는 테이블 열을 선택할 수 있는 기능을 추가했습니다.

AWS SCT 빌드 672의 출시 노트

소스	대상	새로운 기능, 향상된 기능 또는 수정된 기능
모두	Amazon RDS for PostgreSQL	PostgreSQL 메이저 버전 15을 마이그레이션 대상으로 사용하는 기능에 대한 지원을 구현했습니다.
모두	Amazon Redshift	`PrintTaskStatus` AWS SCT 명령줄 인터페이스 (CLI) 에 데이터 마이그레이션 작업의 상태를 표시하는 새 명령을 추가했습니다.
모두	Amazon Redshift	데이터 추출 에이전트의 구성 흐름을 개선했습니다.
모두	Amazon Redshift	데이터 추출 에이전트가 하위 작업에 대한 정보를 표시하지 않는 오류를 해결했습니다.
Apache Oozie	AWS Step Functions	상태 시스템 정의를 변환된 코드에 스크립트로 저장하는 옵션을 추가했습니다.
Azure SQL Database Microsoft SQL Server	Aurora PostgreSQL PostgreSQL	`COALESCE`, `DATEADD`, `GETDATE` 및 `SUM` 함수 변환을 구현했습니다.
Azure SQL Database	Aurora PostgreSQL	`JOIN` 및 `OUTPUT` 절이 있는 `UPDATE` 문의 변환을 개선했습니다.

소스	대상	새로운 기능, 향상된 기능 또는 수정된 기능
Microsoft SQL Server	PostgreSQL	
Azure SQL Database Microsoft SQL Server	Aurora PostgreSQL PostgreSQL	SELECT TOP 1 WITH TIES 문 변환 중에 발생한 오류를 해결했습니다.
Azure SQL Database Microsoft SQL Server	Aurora PostgreSQL PostgreSQL	내장 함수의 FOR XML 절을 변환하는 동안 발생하는 여러 문제를 해결했습니다.
Greenplum	Amazon Redshift	네이티브 Amazon Redshift EXCEPTION 블록을 사용하여 GET DIAGNOSTICS 및 RAISE 문에 대한 변환을 구현했습니다.
Greenplum	Amazon Redshift	변환된 코드에 EXCEPTION 블록 지원을 추가하여 저장 프로시저의 변환을 개선했습니다.
IBM Db2 for z/OS	Aurora PostgreSQL PostgreSQL	시간 형식 템플릿이 있는 TO_CHAR 함수가 잘못 변환되는 오류를 수정했습니다.
IBM Db2 for z/OS	Aurora PostgreSQL PostgreSQL	중첩된 테이블 표현식에 대한 변환을 구현했습니다.

소스	대상	새로운 기능, 향상된 기능 또는 수정된 기능
IBM Db2 for z/OS	Aurora PostgreSQL PostgreSQL	GOTO, MERGE, REPEAT 및 SIGNAL 문에 대한 변환을 구현했습니다.
IBM Db2 for z/OS	Aurora PostgreSQL PostgreSQL	BEFORE 및 AFTER 방향 키워드가 있는 FETCH 문에 대한 변환을 구현했습니다.
IBM Db2 for z/OS	Aurora PostgreSQL PostgreSQL	FINAL TABLE 및 OLD TABLE 테이블 참조에 대한 변환을 구현했습니다.

소스	대상	새로운 기능, 향상된 기능 또는 수정된 기능
IBM Db2 for z/OS	Aurora PostgreSQL PostgreSQL	다음 함수의 변환을 구현했습니다. • ADD_MONTHS • 문자 데이터 형식의 파라미터를 사용하는 DAY • DAYOFWEEK • DAYS • DECODE • HOUR • LAST_DAY • LOCATE_IN_STRING • MICROSECOND • MINUTE • MONTH • ROUND • TIME • TIMESTAMP • TIMESTAMP_FORMAT • TRANSLATE • UNICODE_STR • XMLCAST • XMLELEMENT • XMLQUERY • XMLSERIALIZE • YEAR
IBM Db2 for z/OS	Aurora PostgreSQL PostgreSQL	JOIN 절에서 하위 쿼리의 별칭에 대한 변환을 개선했습니다.

소스	대상	새로운 기능, 향상된 기능 또는 수정된 기능
IBM Db2 for z/OS	Aurora PostgreSQL PostgreSQL	COALESCE 함수에 대한 변환을 개선했습니다.
IBM Db2 for z/OS	Aurora PostgreSQL PostgreSQL	EXPLICIT 인덱스에 대한 변환을 개선했습니다.
IBM Db2 for z/OS	Aurora PostgreSQL PostgreSQL	변환 중에 작업 항목 9997이 예기치 않게 나타나는 문제를 해결하기 위해 복합 표현식에서 열 이름에 대한 변환을 개선했습니다.
IBM Db2 for z/OS	Aurora PostgreSQL PostgreSQL	프라이머리 키 변환 및 고유 제약 조건을 개선했습니다.
IBM Db2 for z/OS	Aurora PostgreSQL PostgreSQL	변환 중에 작업 항목 9996이 예기치 않게 나타나는 문제를 해결하기 위해 INSERT 문에서 XMLTABLE 문에 대한 변환을 개선했습니다.

소스	대상	새로운 기능, 향상된 기능 또는 수정된 기능
IBM Db2 for z/OS	Aurora PostgreSQL PostgreSQL	SUBSTR 인수가 있는 함수를 변환하는 동안 작업 항목 9996이 예기치 않게 나타나는 문제를 해결했습니다.
IBM Db2 for z/OS	Aurora PostgreSQL PostgreSQL	CURRENT TIMESTAMP 특수 레지스터를 변환하는 동안 작업 항목 9996이 예기치 않게 나타나는 문제를 해결했습니다.
IBM Db2 for z/OS	Aurora PostgreSQL PostgreSQL	MERGE 문, 지원되지 않는 명령문 및 지원되지 않는 내장 함수를 변환하는 동안 작업 항목 9996이 예기치 않게 나타나는 문제를 해결했습니다.
Microsoft SQL Server	모두	Microsoft SQL Server 버전 2022를 소스로 사용하도록 지원을 추가했습니다.
Microsoft SQL Server	Aurora PostgreSQL PostgreSQL	문자열 연결 연산자를 사용하는 SELECT 명령문의 변환을 개선했습니다. AWS SCT 변환된 코드에서 STRING_AGG 함수를 사용합니다.
Microsoft SQL Server	Babelfish for Aurora PostgreSQL	Babelfish 기능 구성 파일의 새 버전 3.1.0에 대한 지원을 구현했습니다. 이 파일은 특정 Babelfish 버전에서 지원되거나 지원되지 않는 SQL 기능을 정의합니다.

소스	대상	새로운 기능, 향상된 기능 또는 수정된 기능
Netezza	Amazon Redshift	데이터 추출 에이전트가 지정된 CDC 지점에서 데이터 마이그레이션을 시작하지 않는 문제를 해결했습니다.
Oracle	모두	Oracle 데이터베이스 버전 19를 소스로 사용하도록 평가 보고서를 업데이트했습니다.
Oracle	Aurora PostgreSQL PostgreSQL	AWS SCT 확장 팩에 새 함수를 추가하여 `DBMS_OUTPUT` 패키지 변환을 구현했습니다.
Oracle	Aurora PostgreSQL PostgreSQL	연관 배열을 인수 또는 파라미터로 사용하는 함수 및 프로시저에 대한 변환을 구현했습니다.
Oracle	Aurora PostgreSQL PostgreSQL	`SELECT` 문에서 `DISTINCT` 절의 변환을 개선했습니다.
Oracle	Aurora PostgreSQL PostgreSQL	프라이머리 키 제약 조건의 이름이 테이블과 동일한 경우 해당 테이블의 변환을 개선했습니다.

소스	대상	새로운 기능, 향상된 기능 또는 수정된 기능
Oracle	Aurora PostgreSQL PostgreSQL	세 번째 파라미터가 있는 RAISE_APPLICATION_ERROR 프로시저의 변환을 개선했습니다.
Oracle	Aurora PostgreSQL PostgreSQL	해당되는 경우 마이그레이션 규칙이 NUMERIC 데이터 형식을 INTEGER로 자동 변경하지 않는 문제를 해결했습니다.
Oracle DW	Amazon Redshift	변환된 코드에서 네이티브 Amazon Redshift CONNECT BY 절에 대한 지원을 구현했습니다.
Oracle DW	Amazon Redshift	마이그레이션 범위의 각 테이블 또는 파티션에 대한 하위 작업을 자동으로 추가하여 데이터 마이그레이션을 개선했습니다. 이 방식은 파티셔닝 후 삽입된 데이터의 손실을 방지합니다.
Teradata	Amazon Redshift	재귀 보기의 변환 구현
Teradata	Amazon Redshift	네이티브 Amazon Redshift AUTOCOMMIT 트랜잭션 모드에 대한 지원을 추가하여 BTET 및 ANSI 트랜잭션 모드를 사용하는 저장 프로시저의 변환을 개선했습니다.
Teradata	Amazon Redshift	변환된 코드에 NONATOMIC 키워드를 추가하여 TERADATA 트랜잭션 구문을 사용하는 저장 프로시저의 변환을 개선했습니다.
Teradata	Amazon Redshift RSQL	변환된 코드에 AWS 액세스 키 ID와 비밀 액세스 키가 포함되는 문제가 해결되었습니다.

AWS SCT 빌드 671의 출시 노트

소스	대상	새로운 기능, 향상된 기능 또는 수정된 기능
모두	모두	Windows에서 프로젝트 파일을 저장할 권한이 AWS SCT 없던 오류를 수정했습니다.
모두	모두	다음 AWS SCT 명령줄 인터페이스 (CLI) 템플릿이 업데이트되었습니다. • BTEQ ScriptConversion • ConversionApply • HadoopMigration • HadoopResume마이그레이션 • Informatica AWS SCT CLI 템플릿에 대한 자세한 내용은 을 참조하십시오. CLI 시나리오 가져오기
모두	Amazon Redshift	명령줄 인터페이스 (CLI) 에서 확장 팩을 만들지 AWS SCT 않은 오류가 수정되었습니다.
모두	Amazon Redshift	AWS SCT 데이터 추출 에이전트가 명령줄 인터페이스 (CLI) 의 AWS Snowball 구성을 사용하지 않는 문제가 해결되었습니다.
Apache Oozie	AWS Step Functions	Apache Oozie에서 명령줄 인터페이스 (CLI) AWS Step Functions 모드로의 마이그레이션에 대한 지원을 구현했습니다. 이제 Hadoop 워크로드를 Amazon EMR로 마이그레이션한 후 워크플로 예약 시스템을 AWS 클라우드로 마이그레이션할 수 있습니다. 자세한 정보는 Apache Oozie를 AWS Step Functions로 변환을 참조하세요.
Azure SQL Database Microsoft SQL Server	Aurora PostgreSQL PostgreSQL	테이블 및 별칭에서 발생한 해석기 오류를 수정했습니다.

소스	대상	새로운 기능, 향상된 기능 또는 수정된 기능
Azure SQL Database Microsoft SQL Server	Aurora PostgreSQL PostgreSQL	`INDEX ON` 절의 변환을 구현했습니다.
Azure SQL Database Microsoft SQL Server	Aurora PostgreSQL PostgreSQL	예기치 못한 작업 항목이 발생하지 않도록 다음 객체의 변환을 개선했습니다. • 배치 문 • 표현식 목록 • 테이블 별칭 • 임시 테이블 • 트리거 • 사용자 변수
Azure SQL Database Microsoft SQL Server	Aurora PostgreSQL PostgreSQL	프로시저에서 발생한 구문 분석 오류를 해결했습니다.
Azure SQL Database Microsoft SQL Server	Aurora PostgreSQL PostgreSQL	변환된 함수 코드에서 임시 테이블의 잘못된 이름을 AWS SCT 사용하는 오류를 수정했습니다. `OBJECT_ID`

소스	대상	새로운 기능, 향상된 기능 또는 수정된 기능
Azure SQL Database Microsoft SQL Server	Aurora PostgreSQL PostgreSQL	다음 코드 요소를 변환하는 동안 작업 항목 9996이 예기치 않게 나타나는 문제를 해결했습니다. • CONVERT 함수 • DATEADD 함수 • 인라인 함수 내의 DELETE 문 • IF 문 • 열에서 INSERT 또는 UPDATE 작업 • RETURN 문 • 복잡한 쿼리 또는 함수가 있는 UPDATE 문
BigQuery	Amazon Redshift	다중 서버 평가 프로세스의 BigQuery 소스로 에 대한 지원이 추가되었습니다. 자세한 정보는 다중 서버 평가 보고서 생성을 참조하세요.
Hadoop	Amazon EMR	소스 데이터베이스에 연결하는 데 사용하는 지원되는 Apache Hive JDBC 드라이버 버전을 업데이트했습니다. 자세한 정보는 필수 데이터베이스 드라이버 다운로드을 참조하세요.
IBM Db2 for z/OS	Aurora PostgreSQL PostgreSQL	기본 키, 암시적 인덱스 등과 같은 소스 데이터베이스 개체를 AWS SCT 로드할 수 있도록 소스 메타데이터 로더를 개선했습니다.
IBM Db2 for z/OS	Aurora PostgreSQL PostgreSQL	암시적 커서에서 열에 대해 발생한 해석기 오류를 수정했습니다.

소스	대상	새로운 기능, 향상된 기능 또는 수정된 기능
IBM Db2 for z/OS	Aurora PostgreSQL PostgreSQL	변환된 코드에서 DML 문의 열 이름, 표현식 및 절 형식을 유지하기 위한 기능을 구현했습니다.
IBM Db2 for z/OS	Aurora PostgreSQL PostgreSQL	교차 스키마 외래 키에 대한 변환을 구현했습니다.
IBM Db2 for z/OS	Aurora PostgreSQL PostgreSQL	`LENGTH` 및 `VARCHAR` 함수에 대한 변환을 구현했습니다.
IBM Db2 for z/OS	Aurora PostgreSQL PostgreSQL	`LABEL ON` 및 `DECLARE CONDITION` 문에 대한 변환을 구현했습니다.
IBM Db2 for z/OS	Aurora PostgreSQL PostgreSQL	`OPTIMIZE FOR` 절이 있는 `SELECT` 문에 대한 변환을 구현했습니다.

소스	대상	새로운 기능, 향상된 기능 또는 수정된 기능
IBM Db2 for z/OS	Aurora PostgreSQL PostgreSQL	지원되는 모든 데이터 유형에 기본값을 추가하여 CREATE TABLE 문에 대한 변환을 개선했습니다.
IBM Db2 for z/OS	Aurora PostgreSQL PostgreSQL	INCREMENT BY 속성에 대한 변환을 개선했습니다.
IBM Db2 for z/OS	Aurora PostgreSQL PostgreSQL	변환 범위에서 테이블 파티션을 제외하는 기능을 추가하여 파티션 테이블에 대한 변환을 개선했습니다.
IBM Db2 for z/OS	Aurora PostgreSQL PostgreSQL	INCLUDE 열을 사용한 프라이머리 키 정의에 대한 변환을 개선했습니다.
IBM Db2 for z/OS	Aurora PostgreSQL PostgreSQL	SUBSTRING 함수에 대한 변환을 개선했습니다.

소스	대상	새로운 기능, 향상된 기능 또는 수정된 기능
IBM Db2 for z/OS	Aurora PostgreSQL PostgreSQL	SET 및 DECLARE HANDLER FOR 문에 대한 변환을 개선했습니다.
IBM Db2 for z/OS	Aurora PostgreSQL PostgreSQL	변수 데이터 유형에 대한 변환을 개선했습니다.
IBM Db2 for z/OS	Aurora PostgreSQL PostgreSQL	XMLTABLE 함수에 대한 변환을 개선했습니다.
IBM Db2 for z/OS	Aurora PostgreSQL PostgreSQL	변환된 객체를 대상 데이터베이스에 적용하는 순서(테이블, 파티션, 인덱스, 제약 조건, 외래 키, 트리거)를 구현하여 마이그레이션 흐름을 개선했습니다.
IBM Db2 for z/OS	Aurora PostgreSQL PostgreSQL	소스 코드에서 주석을 변환하는 동안 작업 항목 9996이 예기치 않게 나타나는 문제를 해결했습니다.

소스	대상	새로운 기능, 향상된 기능 또는 수정된 기능
IBM Db2 for z/OS	Aurora PostgreSQL PostgreSQL	FROM 절의 별칭을 변환하는 동안 작업 항목 9997이 예기치 않게 나타나는 문제를 해결했습니다.
IBM Db2 for z/OS	Aurora PostgreSQL PostgreSQL	커서 별칭을 변환하는 동안 작업 항목 9997이 예기치 않게 나타나는 문제를 해결했습니다.
Microsoft SQL Server	Aurora PostgreSQL PostgreSQL	변환된 코드가 ORDER BY 절이 있는 SELECT 문에 대해 다른 결과를 반환하는 오류를 수정했습니다. SQL Server와 PostgreSQL이 NULL 값을 서로 다르게 처리하므로 이제 변환된 코드에는 변환된 코드가 소스 코드와 동일한 순서로 결과를 반환하도록 하는 NULLS FIRST 또는 NULLS LAST절이 포함됩니다.
Microsoft SQL Server	Aurora PostgreSQL PostgreSQL	테이블 함수의 데이터 유형이 잘못 변환되는 문제를 해결했습니다.
MySQL	Amazon RDS for MySQL	변환된 코드에서 데이터베이스 객체 이름 주위에 작은따옴표(' ')가 예기치 않게 나타나는 문제를 해결했습니다.

소스	대상	새로운 기능, 향상된 기능 또는 수정된 기능
Oracle	Aurora PostgreSQL PostgreSQL	파티션 및 하위 파티션에 대한 정보를 표시하는 데 사용하는 Oracle 시스템 보기를 에뮬레이션할 수 있도록 확장 팩에 새 보기를 추가했습니다.
Oracle	Aurora PostgreSQL PostgreSQL	변환된 코드에 스키마 이름을 인수로 추가하도록 확장 팩에서 두 개의 함수를 업데이트했습니다.
Oracle	Aurora PostgreSQL PostgreSQL	사용자 인터페이스에서 응용 프로그램 코드를 새로 고친 후 C++ 응용 프로그램 변환에 올바른 매개 변수를 사용하지 AWS SCT 않는 오류가 수정되었습니다.
Oracle	Aurora PostgreSQL PostgreSQL	예기치 못한 예외가 발생하지 않도록 `CREATE TYPE` 문에 대한 변환을 개선했습니다.
Oracle	Aurora PostgreSQL PostgreSQL	중첩된 테이블에 대한 변환을 개선했습니다.

소스	대상	새로운 기능, 향상된 기능 또는 수정된 기능
Oracle	Aurora PostgreSQL PostgreSQL	패키지 객체에서 발생하는 구문 분석 오류를 해결했습니다.
Oracle	Aurora PostgreSQL PostgreSQL	이름 길이가 60자를 초과하는 경우 변환된 코드에서 개체 이름이 AWS SCT 예기치 않게 잘리는 문제가 해결되었습니다.
Oracle	Aurora PostgreSQL PostgreSQL	파티션 테이블의 행 수준 트리거가 잘못 변환되는 문제를 해결했습니다.
Oracle DW	Amazon Redshift	데이터 마이그레이션을 위한 자동 테이블 파티셔닝 지원을 구현했습니다. 데이터 마이그레이션 속도를 높이기 위해 가상 열의 값을 기준으로 큰 테이블이나 파티션을 자동으로 AWS SCT 분할할 수 있습니다. `ROWID` 자세한 정보는 네이티브 파티셔닝 사용을 참조하세요.
Teradata	Amazon Redshift	변환된 Amazon Redshift 코드에서 네이티브 `MERGE` 명령에 대한 지원을 구현했습니다. Amazon Redshift의 `MERGE` 명령에 대한 자세한 내용은 Amazon Redshift 데이터베이스 개발자 안내서에서 MERGE를 참조하세요.
Teradata	Amazon Redshift	명시적 테이블 이름을 사용하지 않는 `DELETE` 및 `UPDATE` 문에 대한 변환을 개선했습니다.
Teradata	Amazon Redshift	`IN` 및 `NOT IN` 문이 잘못 변환되는 문제를 해결했습니다.

빌드 670의 출시 노트 AWS SCT

소스	대상	새로운 기능, 향상된 기능 또는 수정된 기능
Azure SQL Database Microsoft SQL Server	Aurora PostgreSQL PostgreSQL	다음 코드 요소를 변환하는 동안 작업 항목 9996이 예기치 않게 나타나는 문제를 해결했었습니다. • INCLUDE 문 내의 CREATE INDEX 문 • DECLARE 문 • DECLARE ... TABLE 문 • LOOP 문 내에서 기본값을 사용하는 DECLARE • DELETE 문 • ALTER TABLE 문 내의 DROP CONSTRAINT 문 • EXECUTE AS CALLER 및 REVERT • IIF 문 • 표현식 목록 • MONTH() 함수 • UPDATE 문 • YEAR() 함수
Azure Synapse Analytics	Amazon Redshift	다중 서버 평가 프로세스의 소스로 Azure Synapse Analytics에 대한 지원을 추가했습니다. 자세한 정보는 다중 서버 평가 보고서 생성을 참조하세요.
Hadoop	Amazon EMR	명령줄 인터페이스(CLI) 모드에서 Hadoop 클러스터를 Amazon EMR로 마이그레이션하는 기능에 대한 지원을 구현했습니다. 자세한 정보는 빅 데이터 프레임워크 마이그레이션을 참조하세요.
IBM Db2 for z/OS	Aurora PostgreSQL PostgreSQL	소스 테이블 및 열에서 발생한 해석기 오류를 수정했습니다.

소스	대상	새로운 기능, 향상된 기능 또는 수정된 기능
IBM Db2 for z/OS	Aurora PostgreSQL PostgreSQL	CASE 표현식에 대한 변환을 구현했습니다.
IBM Db2 for z/OS	Aurora PostgreSQL PostgreSQL	CURRENT_DATE 참조를 특수 레지스터로 변환하도록 구현했습니다. Db2 for z/OS에서 특수 레지스터에 대한 참조는 현재 서버에서 제공하는 값에 대한 참조입니다.
IBM Db2 for z/OS	Aurora PostgreSQL PostgreSQL	DATE 및 POSSTR 함수에 대한 변환을 구현했습니다.
IBM Db2 for z/OS	Aurora PostgreSQL PostgreSQL	날짜/시간 상수에 대한 변환을 개선했습니다.
IBM Db2 for z/OS	Aurora PostgreSQL PostgreSQL	DATE, TIME, TIMESTAMP 및 TIMESTAMP WITH TIME ZONE 데이터 유형의 열에 대한 기본값의 변환을 개선했습니다.

소스	대상	새로운 기능, 향상된 기능 또는 수정된 기능
IBM Db2 for z/OS	Aurora PostgreSQL PostgreSQL	SELECT INTO 문을 변환하는 동안 작업 항목 9996이 예기치 않게 나타나는 문제를 해결했습니다.
Microsoft SQL Server	Aurora PostgreSQL PostgreSQL	DATEDIFF 함수에 대한 변환을 개선했습니다.
Microsoft SQL Server	Aurora PostgreSQL PostgreSQL	ISNULL 함수가 NULLIF로 변환되는 오류를 수정했습니다. 이로 인해 변환된 코드는 소스 코드와 다른 결과를 생성했습니다. 이제 ISNULL 함수를 로 AWS SCT COALESCE 변환합니다.
Netezza	Amazon Redshift	성공적으로 완료된 작업에 대해 실패 상태가 설정되는 문제를 해결하도록 데이터 추출 에이전트를 개선했습니다.
Netezza	Amazon Redshift	데이터 추출 에이전트로 데이터 마이그레이션을 시작한 후 하위 작업에서 엔드포인트를 변경하는 기능을 추가했습니다.
Microsoft SQL Server MySQL Oracle PostgreSQL	Aurora MySQL Aurora PostgreSQL MySQL PostgreSQL	IPv6 주소 프로토콜을 사용하여 데이터베이스에 연결하는 기능을 추가했습니다.

소스	대상	새로운 기능, 향상된 기능 또는 수정된 기능
Oracle	Amazon RDS for Oracle	작업 대기열의 작업을 예약하고 관리하는 DBMS_JOB 패키지에 대한 변환을 구현했습니다.
Oracle	Aurora PostgreSQL PostgreSQL	전역 중첩 테이블의 변환을 개선하기 위한 새 함수를 확장 팩에 추가했습니다. 이러한 새 함수는 소스 Oracle 코드에서 DELETE, EXTEND 및 TRIM 함수를 에뮬레이션합니다.
Oracle	Aurora PostgreSQL PostgreSQL	Java 애플리케이션에 포함된 SQL 코드의 변환 범위를 지정하는 기능을 추가했습니다. 이제 변환 범위에서 소스 애플리케이션 프로젝트의 하위 집합을 제외할 수 있습니다. 자세한 정보는 AWS SCT에서 Java 애플리케이션 SQL 코드 변환을 참조하세요.
Oracle	Aurora PostgreSQL PostgreSQL	함수 인덱스 내 연결 연산자(||)에 대한 변환을 개선했습니다.
Oracle	Aurora PostgreSQL PostgreSQL	단일 표현식의 괄호가 소스 코드에 포함되지 않는 IN 조건에 대한 변환을 개선했습니다.

소스	대상	새로운 기능, 향상된 기능 또는 수정된 기능
Oracle	Aurora PostgreSQL PostgreSQL	PostgreSQL에서 `MERGE` 문을 `INSERT ON CONFLICT`로 변환하는 기능을 개선했습니다.
Oracle	Aurora PostgreSQL PostgreSQL	프로시저 패키지에서 발생한 구문 분석 오류를 해결했습니다.
Oracle	Aurora PostgreSQL PostgreSQL	패키지를 변환하는 동안 작업 항목 5072가 예기치 않게 나타나는 문제를 해결했습니다.
Oracle DW	Amazon Redshift	변환된 코드를 대상 데이터베이스에 적용할 때 확장 팩을 AWS SCT 적용하지 않는 오류가 수정되었습니다.
Oracle DW	Amazon Redshift	확장 팩 마법사를 사용할 때 일부 확장 팩 파일이 적용되지 AWS SCT 않는 오류를 수정했습니다.
Oracle DW	Amazon Redshift	병렬로 실행 중인 작업이 500개 이상인 경우 AWS Snowball 로 데이터 마이그레이션을 처리할 수 AWS SCT 없는 문제가 해결되었습니다.
Oracle DW	Amazon Redshift	사용자 정의 형식의 사용자 정의 함수가 잘못 변환되는 문제를 해결했습니다.

AWS SCT 빌드 669의 릴리스 노트

소스	대상	새로운 기능, 향상된 기능 또는 수정된 기능
모두	모두	소스 데이터베이스에 가장 적합한 대상 데이터베이스 플랫폼을 결정하는 데 도움이 되는 다중 서버 평가 프로세스를 개선했습니다. 이제 입력 CSV (쉼표로 구분된 값) 파일에 데이터베이스 자격 증명을 제공하면 AWS Secrets Manager 키가 AWS SCT 무시됩니다. 자세한 정보는 다중 서버 평가 보고서 생성을 참조하세요.
모두	모두	비밀 양식을 사용하여 데이터베이스에 AWS Secrets Manager 연결할 때 다중 서버 평가 보고서에 원본 데이터베이스의 IP 주소가 포함되는 문제가 해결되었습니다.
모두	Amazon Redshift	운영 체제 및 사용 가능한 RAM에 따라 Java 가상 머신 (JVM) 설정의 자동 구성을 구현했습니다. AWS SCT 이 JVM을 사용하여 데이터 추출 에이전트 작업을 실행합니다.
모두	Amazon Redshift	Ubuntu에서 데이터 추출 에이전트가 시작되지 않는 문제를 해결했습니다.
모두	Amazon Redshift	Windows에서 `StartAgent.bat` 파일을 실행한 후 데이터 추출 작업이 시작되지 않는 문제를 해결했습니다.
Azure SQL Database Microsoft SQL Server	Aurora PostgreSQL PostgreSQL	인덱스에 대한 고유 이름 생성 옵션이 켜진 상태에서 열 이름이 잘못 변환되는 문제를 해결했습니다.
Greenplum	Amazon Redshift	프로시저에 `VOID`를 반환하는 함수에 대한 변환을 구현했습니다.
Greenplum	Amazon Redshift	원본 데이터베이스의 숫자 열에 숫자 (NaN) 값이 포함되지 않은 경우 데이터 마이그레이션이 실패하는 문제가 해결되었습니다. AWS SCT 데이터 추출 에이전트는 이제 NaN 값을 NULL로 대체합니다.

소스	대상	새로운 기능, 향상된 기능 또는 수정된 기능
IBM Db2 for z/OS	Aurora PostgreSQL PostgreSQL	CHAR 내장 함수를 변환하는 동안 DATE FORMAT 및 TIME FORMAT 옵션을 지정하는 새로운 변환 설정을 추가했습니다.
IBM Db2 for z/OS	Aurora PostgreSQL PostgreSQL	WITHOUT RETURN 절과 함께 선언된 사전 정의된 커서의 변환을 위한 작업 항목 8534를 추가했습니다. 커서가 결과 집합을 반환하지 않는 경우 변환된 코드의 커서 이름에 NULL 값을 AWS SCT 할당하고 작업 항목을 발생시킵니다.
IBM Db2 for z/OS	Aurora PostgreSQL PostgreSQL	소스 데이터베이스에 연결하는 동안 AWS SCT 를 식별하는 CURRENT CLIENT_APPLNAME 속성을 수정했습니다.
IBM Db2 for z/OS	Aurora PostgreSQL PostgreSQL	CHAR 내장 함수를 변환하는 동안 DATE FORMAT 및 TIME FORMAT 옵션을 지정하는 새로운 변환 설정을 구현했습니다.
IBM Db2 for z/OS	Aurora PostgreSQL PostgreSQL	BEGIN...END 블록 문에서 LEAVE 문의 변환을 구현했습니다.

소스	대상	새로운 기능, 향상된 기능 또는 수정된 기능
IBM Db2 for z/OS	Aurora PostgreSQL PostgreSQL	XMLPARSE, XMLTABLE 및 XMLNAMESPACES 함수의 변환을 구현했습니다.
IBM Db2 for z/OS	Aurora PostgreSQL PostgreSQL	CHAR 내장 함수의 변환을 개선했습니다.
IBM Db2 for z/OS	Aurora PostgreSQL PostgreSQL	커서 변환 개선
IBM Db2 for z/OS	Aurora PostgreSQL PostgreSQL	FOR 루프 문을 변환하는 동안 작업 항목 9996이 예기치 않게 나타나는 문제를 해결했습니다.
Microsoft SQL Server	Aurora PostgreSQL PostgreSQL	SELECT 문에서 테이블 유형 사용에 대한 변환을 개선했습니다.

소스	대상	새로운 기능, 향상된 기능 또는 수정된 기능
Microsoft SQL Server	Babelfish for Aurora PostgreSQL	Babelfish 기능 구성 파일의 새 버전 2.2.0에 대한 지원을 구현했습니다. 이 파일은 특정 Babelfish 버전에서 지원되거나 지원되지 않는 SQL 기능을 정의합니다.
Netezza	Amazon Redshift	진행 중인 데이터 복제 중에 대상 테이블에서 한 행이 삭제되지 않는 문제를 해결하도록 데이터 추출 에이전트를 개선했습니다.
Oracle	Amazon RDS for Oracle	Oracle Database Enterprise Edition 기능에 대한 변환을 개선했습니다.
Oracle	Aurora PostgreSQL PostgreSQL	`GROUPING_ID` 함수의 변환을 구현했습니다.
Oracle	Aurora PostgreSQL PostgreSQL	명령줄 인터페이스(CLI) 모드에서 사용자 지정 데이터 유형 매핑 지원을 추가하여 C# 애플리케이션에서 SQL 코드의 변환을 개선했습니다.
Oracle	Aurora PostgreSQL PostgreSQL	예기치 않은 작업 항목 9996이 발생하지 않도록 중첩 테이블의 변환을 개선했습니다.

소스	대상	새로운 기능, 향상된 기능 또는 수정된 기능
Oracle	Aurora PostgreSQL PostgreSQL	객체 생성자 호출이 잘못 변환되는 문제를 해결했습니다.
Oracle DW	Amazon Redshift	데이터 마이그레이션을 위한 기존 테이블 파티션 지원을 구현했습니다. 데이터 마이그레이션 속도를 높이기 위해 소스 테이블의 각 파티션에 대해 비어 있지 않은 하위 작업을 AWS SCT 만듭니다. 자세한 정보는 네이티브 파티셔닝 사용을 참조하세요.
Teradata	Amazon Redshift	`TIME WITH TIME ZONE AS TIMESTAMP` , `TIME WITH TIME ZONE AS CHAR` 및 `TIMESTAMP AS TIME WITH TIME ZONE` 인수가 있는 `CAST` 함수의 변환을 개선했습니다.
Teradata	Amazon Redshift	`FORMAT` 옵션을 통한 `CAST` 함수의 변환을 개선했습니다.
Teradata	Amazon Redshift	`CEIL` 함수가 변환되지 않는 문제를 해결했습니다.
Teradata	Amazon Redshift	`DELETE` 절이 있는 `MERGE` 문이 잘못 변환되는 문제를 해결했습니다.
Teradata	Amazon Redshift	날짜 및 형식 인수가 있는 `TO_CHAR` 함수가 잘못 변환되는 문제를 해결했습니다.

AWS SCT 빌드 668의 릴리스 노트

소스	대상	새로운 기능, 향상된 기능 또는 수정된 기능
모두	Amazon Redshift	마이그레이션 규칙에서 곱셈 연산자가 제대로 작동하지 않는 문제를 해결했습니다. 이러한 연산자를 사용하면 `char`, `varchar`,

소스	대상	새로운 기능, 향상된 기능 또는 수정된 기능
		nvarchar, string 데이터 유형의 길이를 변경할 수 있습니다. 자세한 정보는 마이그레이션 규칙 생성을 참조하세요.
Azure Synapse Analytics	Amazon Redshift	VARCHAR 인수가 있는 CONVERT 함수에 대한 지원을 구현했습니다.
Azure Synapse Analytics	Amazon Redshift	NOLOCK 절이 있는 SELECT 문의 변환을 개선했습니다.
Azure Synapse Analytics	Amazon Redshift	별칭 또는 SET 및 FROM 절이 있는 UPDATE 문의 변환을 개선했습니다.
Greenplum	Amazon Redshift	데이터 마이그레이션을 위한 자동 가상 파티셔닝을 구현했습니다. AWS SCT 는 GP_SEGMENT_ID 시스템 열을 사용하여 파티션을 생성합니다.
Greenplum	Amazon Redshift	RETURN QUERY 및 RETURN SETOF 절에 대한 지원을 구현했습니다.
Greenplum	Amazon Redshift	세 개의 파라미터가 있는 SUBSTRING 함수에 대한 지원을 구현했습니다.
IBM Db2 for z/OS	Aurora PostgreSQL PostgreSQL	LOCATE 파라미터가 있는 SUBSTR 함수에 대한 변환을 개선했습니다.

소스	대상	새로운 기능, 향상된 기능 또는 수정된 기능
IBM Db2 for z/OS	Aurora PostgreSQL PostgreSQL	REFCURSOR 변수의 배열을 사용하여 동적 결과 집합을 반환하는 옵션을 추가했습니다. 변환 설정에서 이 옵션을 선택하면 AWS SCT 가 변환된 코드에 추가적인 OUT 파라미터를 추가합니다.
IBM Db2 for z/OS	Aurora PostgreSQL PostgreSQL	FOR 루프 문에 대한 지원 구현
IBM Db2 for z/OS	Aurora PostgreSQL PostgreSQL	XMLPARSE 함수에 대한 지원을 구현했습니다. XMLPARSE 함수에서 공백 문자 스트라이핑을 위한 작업 항목 8541을 추가했습니다.
IBM Db2 for z/OS	Aurora PostgreSQL PostgreSQL	단일 BEGIN ... END 블록에서 여러 예외 핸들러에 대한 변환을 개선했습니다.
Microsoft SQL Server	Aurora PostgreSQL PostgreSQL	INSERT 및 DELETE 트리거에 대한 변환을 개선했습니다.

소스	대상	새로운 기능, 향상된 기능 또는 수정된 기능
Microsoft SQL Server	Aurora PostgreSQL PostgreSQL	중첩 프로시저 호출에 대한 변환을 개선했습니다.
Microsoft SQL Server	Aurora PostgreSQL PostgreSQL	테이블 유형에 대한 변환을 개선했습니다.
Microsoft SQL Server	Aurora PostgreSQL PostgreSQL	정수 값에 대해 비트 논리 NOT 연산이 잘못 변환되는 문제를 해결했습니다.
Microsoft SQL Server	Aurora PostgreSQL PostgreSQL	PostgreSQL 버전 8.0.2 이하에서 로컬 배열이 초기화되지 않는 문제를 해결했습니다.
Microsoft SQL Server	Aurora PostgreSQL PostgreSQL	WHEN NOT MATCHED BY SOURCE 절이 있는 MERGE 문이 잘못 변환되는 문제를 해결했습니다.
MySQL	Aurora MySQL	rds_superuser_role 역할에 의해 부여된 사용자 권한이 AWS SCT 잘못 결정되는 문제가 해결되었습니다.

소스	대상	새로운 기능, 향상된 기능 또는 수정된 기능
Netezza	Amazon Redshift	이름이 소문자로 된 데이터베이스 개체를 AWS SCT 올바르게 로드할 수 있도록 소스 메타데이터 로더를 개선했습니다.
Oracle	Aurora PostgreSQL PostgreSQL	로컬 중첩 테이블의 변환을 개선하기 위한 새 함수를 확장 팩에 추가했습니다. 이러한 새 함수는 소스 Oracle 코드에서 PRIOR, NEXT, LIMIT, FIRST, LAST, EXISTS, EXTEND, TRIM, DELETE, SET 함수를 에뮬레이션합니다. 자세한 정보는 확장 팩 사용을 참조하세요.
Oracle	Aurora PostgreSQL PostgreSQL	C# 애플리케이션의 변환 범위를 지정하는 기능을 추가했습니다. 이제 사용자가 변환 범위에서 소스 애플리케이션 프로젝트의 하위 집합을 제외할 수 있습니다.
Oracle	Aurora PostgreSQL PostgreSQL	컬렉션에서 COUNT 메서드에 대한 지원을 구현했습니다.
Oracle	Aurora PostgreSQL PostgreSQL	중첩된 테이블의 변수 및 생성자에 대한 지원을 구현했습니다.
Oracle	Aurora PostgreSQL PostgreSQL	RATIO_TO_REPORT 및 STANDARD_HASH 함수에 대한 지원 구현

소스	대상	새로운 기능, 향상된 기능 또는 수정된 기능
Oracle	Aurora PostgreSQL PostgreSQL	AWS SCT 확장 팩에 속하는 큰 객체(LOB)에 대한 변환을 개선했습니다.
Oracle	Aurora PostgreSQL PostgreSQL	로컬 컬렉션에 대한 변환을 개선했습니다.
Oracle	Aurora PostgreSQL PostgreSQL	열 이름에 테이블 이름이 포함되지 않는 `USING` 절이 있는 `JOIN` 문에 대한 변환을 개선했습니다.
Oracle	Aurora PostgreSQL PostgreSQL	`EMPTY_BLOB` 및 `EMPTY_CLOB` 함수의 변환을 구현했습니다.
Oracle	Aurora PostgreSQL PostgreSQL	C# 애플리케이션에서 위치 바인드 변수의 변환을 구현했습니다.

소스	대상	새로운 기능, 향상된 기능 또는 수정된 기능
SAP ASE	Aurora PostgreSQL PostgreSQL	다중 이벤트 트리거의 변환을 구현했습니다.
SAP ASE	Aurora PostgreSQL PostgreSQL	재귀 트리거의 변환을 구현했습니다.
SAP ASE	Aurora PostgreSQL PostgreSQL	`@@rowcount` 전역 변수를 사용한 트리거 변환을 개선했습니다.
SAP ASE	Aurora PostgreSQL PostgreSQL	`UPDATE` 문의 `SET` 절에서 집계 함수가 잘못 변환되는 문제를 해결했습니다.
SAP ASE	Aurora PostgreSQL PostgreSQL	`UPDATE` 문을 변환하는 동안 작업 항목 42702가 예기치 않게 나타나는 문제를 해결했습니다.

소스	대상	새로운 기능, 향상된 기능 또는 수정된 기능
SAP ASE	Aurora PostgreSQL PostgreSQL	CHAR 인수가 있는 CONVERT 함수가 잘못 변환되는 문제를 해결했습니다.
Snowflake	Amazon Redshift	데이터 추출 에이전트를 사용한 데이터 마이그레이션을 위한 소스로 Snowflake에 대한 지원이 추가되었습니다. AWS SCT 자세한 정보는 온프레미스 데이터 웨어하우스에서 Amazon Redshift로 데이터 마이그레이션을 참조하세요.
Teradata	Amazon Redshift	TIMESTAMP AS TIME WITH TIMEZONE 인수가 있는 CAST 함수의 변환을 개선했습니다.

빌드 667의 AWS SCT 릴리스 노트

소스	대상	새로운 기능, 향상된 기능 또는 수정된 기능
모두	모두	명령줄 인터페이스 (CLI) 모드에서 Informatica ETL (추출, 변환 및 로드) 스크립트에 대한 지원을 구현했습니다. AWS SCT Informatica ETL 스크립트를 새 대상 데이터베이스로 자동으로 리디렉션합니다. 또한 AWS SCT Informatica 객체에 내장된 객체 이름과 SQL 코드를 변환합니다. 자세한 정보는 Informatica ETL 스크립트 변환을 참조하세요.
모두	Amazon Redshift	Amazon Redshift에 대해 지원되는 최소 드라이버 버전을 2.1.0.9로 높였습니다. 자세한 정보는 필수 데이터베이스 드라이버 다운로드을 참조하세요.
Azure Synapse Analytics	Amazon Redshift	세 개의 날짜 및 시간 인수가 있는 CONVERT 함수의 변환을 개선하기 위해 확장 팩에 새로운 함수를 추가했습니다.

소스	대상	새로운 기능, 향상된 기능 또는 수정된 기능
Azure Synapse Analytics	Amazon Redshift	DATEDIFF 함수에 대한 변환을 개선했습니다.
Azure Synapse Analytics Microsoft SQL Server DW	Amazon Redshift	확장 팩 버전을 업데이트했습니다. 기존 AWS SCT 프로젝트에 최신 버전의 확장 팩을 적용해야 합니다. 자세한 정보는 확장 팩 사용을 참조하세요.
BigQuery	Amazon Redshift	필터링된 객체가 명령줄 인터페이스(CLI) 모드에서 변환되지 않는 문제를 해결했습니다.
Greenplum	Amazon Redshift	저장 프로시저에서 선언된 임시 테이블을 변환하지 AWS SCT 않던 오류를 수정했습니다.
Greenplum	Amazon Redshift	변환된 코드에서 열 인코딩 속성이 누락되는 오류를 수정했습니다.
Microsoft SQL Server	Aurora PostgreSQL PostgreSQL	INNER JOIN 절이 두 개 이상 있는 자체 참조 테이블에 대한 UPDATE 문의 변환을 구현했습니다.
Microsoft SQL Server	Aurora PostgreSQL PostgreSQL	SQL Server가 DML 트리거에 사용하는 inserted 및 deleted 임시 테이블에 대한 지원을 구현했습니다.

소스	대상	새로운 기능, 향상된 기능 또는 수정된 기능
Microsoft SQL Server	Aurora PostgreSQL PostgreSQL	다양한 데이터베이스 스키마에서 생성된 저장 프로시저의 사용자 정의 형식 변환을 개선했습니다. 데이터 유형을 찾을 AWS SCT 수 없고 작업 항목 9996이 표시되는 문제가 해결되었습니다.
Microsoft SQL Server	Aurora PostgreSQL PostgreSQL	변환된 코드에서 데이터베이스 객체 이름 주위에 대괄호([ ])가 예기치 않게 나타나는 문제를 해결했습니다.
Microsoft SQL Server	Aurora PostgreSQL PostgreSQL	`@@ROWCOUNT` 함수가 잘못 변환되는 문제를 해결했습니다.
Microsoft SQL Server DW	Amazon Redshift	`geometry` 및 `geography` 데이터 유형에 대한 지원을 구현했습니다.
Microsoft SQL Server DW	Amazon Redshift	변환된 코드의 데이터 형식 선언에서 `MAX` 키워드에 대한 지원을 구현했습니다.
Microsoft SQL Server DW	Amazon Redshift	`DATEADD` 함수에 대한 변환을 개선했습니다.

소스	대상	새로운 기능, 향상된 기능 또는 수정된 기능
Oracle	Aurora PostgreSQL PostgreSQL	MyBatis프레임워크에 대한 지원을 추가하여 Java 응용 프로그램의 SQL 코드 변환을 개선했습니다. 자세한 정보는 Java 애플리케이션의 SQL 코드 변환을 참조하세요.
Oracle	Aurora PostgreSQL PostgreSQL	MyBatis 프레임워크를 사용하는 Java 애플리케이션의 SQL 코드 변환이 개선되었습니다. 지원되지 않는 구문이 있는 SQL 코드에 대한 작업 항목 30411을 추가했습니다.
Oracle	Aurora PostgreSQL PostgreSQL	`typedef struct` 선언에 대한 지원을 추가하여 Pro*C 애플리케이션의 SQL 코드 변환을 개선했습니다.
Oracle	Aurora PostgreSQL PostgreSQL	`CROSS JOIN` 및 `LEFT JOIN` 문에 대한 지원을 구현했습니다.
Oracle	Aurora PostgreSQL PostgreSQL	`MERGE` 문의 변환을 개선했습니다. 변환된 코드에 삽입할 값이 누락되는 문제를 해결했습니다.

소스	대상	새로운 기능, 향상된 기능 또는 수정된 기능
Teradata	Amazon Redshift	AWS SCT 가 변환된 코드에서 사용하는 기본 열 압축 인코딩 설정을 기본 Amazon Redshift 설정과 일치하도록 변경했습니다. 자세한 내용은 Amazon Redshift 데이터베이스 개발자 안내서의 압축 인코딩을 참조하세요.
Teradata	Amazon Redshift	해당 `TIME` 데이터 유형을 사용하는 수학 연산이 잘못 변환되는 문제를 해결했습니다.
Teradata	Amazon Redshift RSQL	셸 스크립트 내에 있는 FastExport 코드의 변환을 구현했습니다.
Teradata BTEQ	Amazon Redshift RSQL	`COALESCE`및 `%data` 명령문을 AWS SCT 변환하지 않는 오류가 수정되었습니다.
Vertica	Amazon Redshift	사용자가 최적화 전략을 선택할 때의 변환 최적화 제안을 개선했습니다.

AWS SCT 빌드 666의 출시 노트

소스	대상	새로운 기능, 향상된 기능 또는 수정된 기능
Azure SQL Database Microsoft SQL Server	Aurora PostgreSQL PostgreSQL	`JOIN` 문 내에 있는 `ON` 절에서 발생하는 구문 분석 오류를 해결했습니다.
Azure Synapse Analytics	Amazon Redshift	날짜 및 시간 인수가 있는 `CONVERT` 함수의 변환을 개선하기 위해 확장 팩에 세 개의 새로운 함수를 추가했습니다.

소스	대상	새로운 기능, 향상된 기능 또는 수정된 기능
Azure Synapse Analytics	Amazon Redshift	시스템 데이터베이스 스키마를 로드할 수 있도록 AWS SCT 소스 메타데이터 로더를 개선했습니다.
Azure Synapse Analytics	Amazon Redshift	임시 테이블의 열에서 발생한 해석기 오류를 수정했습니다.
Azure Synapse Analytics	Amazon Redshift	`BINARY` 및 `VARBINARY` 데이터 유형을 `VARBYTE` 데이터 유형으로 변환하도록 구현했습니다.
Azure Synapse Analytics	Amazon Redshift	변환된 코드의 `TIME` 데이터 유형에 대한 지원을 구현했습니다.
Azure Synapse Analytics	Amazon Redshift	`COLLATE` 절의 변환을 개선했습니다. 기본 데이터베이스의 데이터 정렬을 사용하여 열을 변환하는 동안 작업 항목 31141이 예기치 않게 나타나는 문제를 해결했습니다.
BigQuery	Amazon Redshift	입력 파라미터를 변경하는 프로시저의 변환을 구현했습니다.
Greenplum	Amazon Redshift	Greenplum 6.x 데이터베이스와 호환되지 않는 쿼리를 AWS SCT 사용하는 문제가 해결되었습니다.
IBM Db2 for z/OS	Aurora PostgreSQL PostgreSQL	Db2 for z/OS에서 PostgreSQL로 예외 핸들러를 전송하여 `EXCEPTION` 섹션 변환을 개선했습니다.

소스	대상	새로운 기능, 향상된 기능 또는 수정된 기능
IBM Db2 for z/OS	Aurora PostgreSQL PostgreSQL	OPEN CURSOR 문의 변환을 개선했습니다.
Microsoft SQL Server	Aurora PostgreSQL PostgreSQL	CASE 표현식을 사용한 IIF 함수의 변환을 구현했습니다.
Microsoft SQL Server	Aurora PostgreSQL PostgreSQL	CREATE PROCEDURE 문에 BEGIN...END 블록이 포함되지 않은 경우 테이블 반환 파라미터가 있는 프로시저가 잘못 변환되는 문제를 해결했습니다.
Microsoft SQL Server	Aurora PostgreSQL PostgreSQL	SCOPE_IDENTITY 함수가 잘못 변환되는 문제를 해결했습니다.
Oracle	Amazon RDS for Oracle	Oracle 10g를 소스로 사용할 때 SELECT_CATALOG_ROLE 역할에서 발생한 로더 오류를 수정했습니다.
Oracle	Amazon RDS for Oracle	Oracle Scheduler 작업을 지원하도록 로더를 개선했습니다.

소스	대상	새로운 기능, 향상된 기능 또는 수정된 기능
Oracle	Aurora PostgreSQL PostgreSQL	USING 절이 있는 JOIN 문에 대한 변환을 구현했습니다.
Oracle	Aurora PostgreSQL PostgreSQL	소스 코드의 WHERE 절에 전역 변수가 포함되어 있는 변환된 코드의 성능을 개선했습니다.
Oracle	Aurora PostgreSQL PostgreSQL	프레임워크에 대한 지원을 추가하여 Java 애플리케이션의 SQL 코드 변환을 개선했습니다 MyBatis. 자세한 정보는 Java 애플리케이션의 SQL 코드 변환을 참조하세요.
Oracle DW	Amazon Redshift	PIVOT 및 UNPIVOT 관계 연산자의 변환을 구현했습니다.
Teradata	Amazon Redshift	JSON 객체를 사용하는 소스 코드가 변환되지 않는 오류를 수정했습니다.
Teradata	Amazon Redshift	삭제된 사용자가 만든 테이블이 제대로 로드되지 않는 오류를 수정했습니다.
Teradata	Amazon Redshift	INSTR 함수를 네이티브 Amazon Redshift STRPOS 함수로 변환하는 기능을 구현했습니다.
Teradata	Amazon Redshift	NVP 및 TRANSLATE 함수에 대한 변환을 구현했습니다.
Teradata	Amazon Redshift	COALESCE 표현식 변환을 개선했습니다.

소스	대상	새로운 기능, 향상된 기능 또는 수정된 기능
Teradata	Amazon Redshift	DECLARE CONDITION 문의 변환을 개선했습니다.
Teradata	Amazon Redshift	SECOND 구문 요소가 있는 EXTRACT 함수의 변환을 개선했습니다.
Teradata	Amazon Redshift	LOOP 문 내 SQLSTATE 및 SQLCODE 변수의 변환을 개선했습니다.
Teradata	Amazon Redshift	고유 인덱스 변환을 개선했습니다.
Teradata	Amazon Redshift	소수 정밀도가 3으로 설정된 CURRENT_TIMESTAMP 문을 변환하는 동안 작업 항목 9996이 예기치 않게 나타나는 문제를 해결했습니다.
Teradata	Amazon Redshift	문자열 리터럴에서 백슬래시가 잘못 변환되는 문제를 해결했습니다.
Teradata	Amazon Redshift	변환된 EXEC 문에 ADD CONSTRAINT 문의 잘못된 필드 이름이 포함되는 문제를 해결했습니다.
Teradata	Amazon Redshift	변환된 QUALIFY 하위 쿼리에 잘못된 하위 쿼리 이름이 포함되는 문제를 해결했습니다.
Teradata	Amazon Redshift	변환된 보기가 적용되지 않는 문제를 해결했습니다. 변환된 코드의 NULL 값에 대해 특정 데이터 유형으로의 명시적인 캐스팅을 추가했습니다.
Teradata	Amazon Redshift	날짜 및 시간 함수가 잘못 변환되는 문제를 해결했습니다.
Teradata	Amazon Redshift	16진 문자열 리터럴이 변환되지 않는 문제를 해결했습니다.

AWS SCT 빌드 665의 릴리스 노트

소스	대상	새로운 기능, 향상된 기능 또는 수정된 기능
Azure Synapse Analytics	Amazon Redshift	VARCHAR 인수가 있는 CONCAT 함수에 대한 변환을 구현했습니다.
Azure Synapse Analytics	Amazon Redshift	임시 테이블을 생성하고 스키마 이름을 포함하지 않는 CREATE TABLE 명령문의 변환이 개선되었습니다. AWS SCT 대상 데이터베이스에 이러한 임시 테이블을 저장하기 위한 dbo 스키마를 만듭니다.
Azure Synapse Analytics	Amazon Redshift	임시 테이블에서 실행하는 DROP TABLE 문의 변환을 개선했습니다.
Azure Synapse Analytics	Amazon Redshift	BEGIN...END 블록이 있는 OBJECT_ID 문에 대한 변환을 개선했습니다.
Azure Synapse Analytics	Amazon Redshift	블록 주석이 있는 저장 프로시저를 변환할 AWS SCT 수 없는 오류가 해결되었습니다.
BigQuery	Amazon Redshift	BigQuery 데이터 웨어하우스를 Amazon Redshift로 전환하도록 구현했습니다. 자세한 정보는 BigQuery를 AWS SCT에 대한 소스로 사용을 참조하세요.
Microsoft SQL Server	Aurora PostgreSQL PostgreSQL	여러 이벤트를 처리하고 SQL Server에서 inserted 및 deleted 시스템 테이블로 작업하는 트리거의 변환을 개선했습니다.
Microsoft SQL Server	Aurora PostgreSQL	SQL Server의 inserted 및 deleted 시스템 테이블에서 발생한 해석기 오류를 수정했습니다.

소스	대상	새로운 기능, 향상된 기능 또는 수정된 기능
	PostgreSQL	
Microsoft SQL Server	Babelfish for Aurora PostgreSQL	Babelfish 기능 구성 파일의 새 버전 2.1.0에 대한 지원을 구현했습니다. 이 파일은 특정 Babelfish 버전에서 지원되거나 지원되지 않는 SQL 기능을 정의합니다.
Oracle	Aurora MySQL MariaDB MySQL	`varchar2` 데이터 유형이 잘못 변환되는 문제를 해결했습니다.
Oracle	Aurora MySQL Aurora PostgreSQL MariaDB MySQL PostgreSQL	Oracle 데이터베이스 버전 12c 이상에서는 다음과 같은 확장 데이터 유형을 AWS SCT 지원합니다. • `VARCHAR2` • `NVARCHAR2` • `RAW` AWS SCT 이러한 데이터 유형에 대해 지원되는 최대 열 길이를 8,000바이트에서 32,767바이트로 늘렸습니다.
Oracle	Aurora PostgreSQL PostgreSQL	Oracle Event Processing 패키지에서 발생한 구문 분석 오류를 해결했습니다.
Teradata	Amazon Redshift	단일 `SELECT` 문의 여러 `RESET WHEN` 절에 대한 작업 항목 13214를 추가했습니다.

소스	대상	새로운 기능, 향상된 기능 또는 수정된 기능
Teradata	Amazon Redshift	예외 처리 블록 외부에 있는 SQLSTATE 변수에 대한 작업 항목을 추가했습니다.
Teradata	Amazon Redshift	ACTIVITY_COUNT 변수를 ROW_COUNT 로 변환하도록 구현했습니다.
Teradata	Amazon Redshift	내장 지오메트리 ST_TRANSFORM 함수의 변환을 구현했습니다.
Teradata	Amazon Redshift	WHERE 절이 없는 보기의 삭제 명령문 변환을 개선했습니다.
Teradata	Amazon Redshift	표현식의 CAST 연산자 변환을 개선했습니다.
Teradata	Amazon Redshift	GROUP BY 절의 변환을 개선했습니다.
Teradata	Amazon Redshift	INSTR 및 REGEXP_INSTR 내장 함수의 변환을 개선했습니다.
Teradata	Amazon Redshift	측면 열 별칭 참조가 잘못 변환되는 문제를 해결했습니다.
Teradata	Amazon Redshift	QUALIFY 하위 쿼리에서 열 이름이 잘못 변환되는 문제를 해결했습니다.
Teradata	Amazon Redshift	ERRORCODE 상태 값 키워드를 사용하여 .QUIT 명령을 변환하도록 구현했습니다.
Teradata BTEQ	Amazon Redshift RSQL	CREATE 문을 변환하는 동안 작업 항목 9996이 예기치 않게 나타나는 문제를 해결했습니다.
Teradata BTEQ	Amazon Redshift RSQL	END 문을 변환하는 동안 작업 항목 9998이 예기치 않게 나타나는 문제를 해결했습니다.

빌드 664의 릴리스 노트 AWS SCT

소스	대상	새로운 기능, 향상된 기능 또는 수정된 기능
모두	모두	AWS SCT에서 Amazon Redshift Serverless를 데이터베이스 마이그레이션 프로젝트의 소스 및 대상으로 사용하도록 하는 지원을 추가했습니다. Amazon Redshift Serverless에 연결하려면 Amazon Redshift JDBC 드라이버 버전 2.1.0.9 이상을 사용해야 합니다.
모두	모두	전환 설정 창의 사용자 인터페이스가 개선되었습니다. AWS SCT 이제 매핑 규칙이 생성된 데이터베이스 변환 쌍에 대한 설정만 표시합니다. 자세한 정보는 매핑 규칙 생성을 참조하세요.
모두	모두	작업 항목의 행 및 위치에 대한 중복 정보를 제거하도록 평가 보고서를 업데이트했습니다.
모두	Amazon Redshift	데이터 추출 작업에서 자동 메모리 밸런싱을 구현했습니다.
모두	Amazon Redshift	데이터 추출 에이전트가 AWS Snowball 디바이스에 연결할 수 없는 오류를 해결했습니다.
Azure SQL Database IBM Db2 for z/OS IBM Db2 LUW Microsoft SQL Server MySQL Oracle	Aurora MySQL Aurora PostgreSQL MariaDB MySQL PostgreSQL	데이터 추출 에이전트를 실행하는 플랫폼으로 SUSE Linux 15.3을 사용하도록 지원을 구현했습니다.

소스	대상	새로운 기능, 향상된 기능 또는 수정된 기능
PostgreSQL SAP ASE		
Azure Synapse Analytics	Amazon Redshift	DATEADD 함수에 대한 변환을 개선했습니다.
IBM Db2 for z/OS	Aurora PostgreSQL PostgreSQL	마이그레이션 규칙에서 열 데이터 정렬을 변경하는 기능을 추가했습니다.
Microsoft SSIS	AWS Glue AWS Glue Studio	사용자가 소스 스크립트를 선택할 때 발생하는 예기치 않은 오류를 해결했습니다.
Oracle	Aurora MySQL MariaDB MySQL	저장된 함수의 사용을 생성된 열 표현식으로 변환하도록 구현했습니다. AWS SCT MySQL은 저장된 함수를 생성된 열 표현식으로 사용하는 것을 지원하지 않기 때문에 트리거를 생성하여 이 동작을 에뮬레이션합니다.
Oracle	Aurora PostgreSQL PostgreSQL	UTL_MATCH 패키지의 함수를 확장 팩의 AWS SCT 일부로 변환하도록 구현했습니다.

소스	대상	새로운 기능, 향상된 기능 또는 수정된 기능
Oracle	Aurora PostgreSQL PostgreSQL	NULL 파라미터가 있는 REGEXP_LIKE 함수의 변환을 구현했습니다.
Oracle	Aurora PostgreSQL PostgreSQL	SYS_EXTRACT_UTC 함수에 대한 변환을 개선했습니다.
Oracle	Aurora PostgreSQL PostgreSQL	Wcscats, Wcscpys 및 Wcsncats 함수에 대한 지원을 구현하여 C++ 애플리케이션에서 SQL 코드 변환을 개선했습니다. 자세한 정보는 AWS SCT를 사용하여 C++ 애플리케이션의 SQL 코드 변환을 참조하세요.
Oracle DW Snowflake	Amazon Redshift	변환된 명령문에 열 데이터 유형으로의 명시적 값 변환이 포함되지 않는 문제를 해결했습니다. 이 문제는 다른 테이블의 쿼리 결과를 사용하는 명령문에서 발생했습니다.
Teradata	Amazon Redshift	마이그레이션 규칙에서 case sensitive 및 case insensitive 사이에 열 데이터 정렬을 변경하는 기능을 추가했습니다. 자세한 정보는 마이그레이션 규칙 생성을 참조하세요.
Teradata	Amazon Redshift	CREATE TABLE AS 문에서 발생한 해석기 오류를 수정했습니다.
Teradata	Amazon Redshift	COALESCE 표현식이 있는 내장 P_INTERSECT 함수가 변환되지 않는 오류를 수정했습니다.
Teradata	Amazon Redshift	Amazon Redshift에서 예약된 키워드를 사용하지 않도록 이름이 OID인 열의 _OID로의 변환을 구현했습니다.

소스	대상	새로운 기능, 향상된 기능 또는 수정된 기능
Teradata	Amazon Redshift	함수, 프로시저, 보기 및 매크로에 대한 RENAME 문의 변환을 구현했습니다.
Teradata	Amazon Redshift	Amazon Redshift에서 STROKE 함수를 SPLIT_PART 함수로 변환하는 기능을 구현했습니다.
Teradata	Amazon Redshift	INSTR 및 REGEXP_INSTR 시스템 함수의 변환을 개선했습니다.
Teradata	Amazon Redshift	TIME 데이터 유형에 대한 변환을 개선했습니다.
Teradata	Amazon Redshift	기본 및 보조 고유 인덱스 변환을 구현하여 SET 및 MULTISET 테이블의 에뮬레이션을 개선했습니다.
Teradata	Amazon Redshift	CHARACTER 함수에 대해 발생한 구문 분석 오류를 해결했습니다.
Teradata BTEQ	Amazon Redshift RSQL	사용자가 프로젝트에서 테라데이타 기본 테라데이타 쿼리 (BTEQ) 스크립트를 제거했을 때 발생하는 오류가 해결되었습니다. AWS SCT

빌드 663의 릴리스 노트 AWS SCT

소스	대상	새로운 기능, 향상된 기능 또는 수정된 기능
모두	모두	마이그레이션 규칙에 곱셈 연산자를 사용하여 char, varchar, nvarchar, string 데이터 유형의 길이를 변경하는 기능을 추가했습니다. 자세한 정보는 마이그레이션 규칙 생성을 참조하세요.
모두	모두	다중 서버 평가 보고서에서 세 개의 새로운 열에 대한 지원을 구현하고 입력 파일의 형식을 업데이트했습니다. 입력 파일의 업데이트된 템플릿을 AWS SCT의 최신 버전과 함께 사용해야 합니다. 자세한 정보는 데이터베이스 마이그레이션을 위한 다중 서버 평가 보고서 생성을 참조하세요.

소스	대상	새로운 기능, 향상된 기능 또는 수정된 기능
Azure Synapse Analytics	Amazon Redshift	`OBJECT_ID` 문의 변환을 개선했습니다.
Microsoft SQL Server	Babelfish for Aurora PostgreSQL	Babelfish for Aurora PostgreSQL 1.2.0을 데이터베이스 마이그레이션 평가 보고서의 대상 플랫폼으로 사용하기 위한 지원을 추가했습니다. 자세한 내용은 Amazon Aurora 사용 설명서에서 버전별 Babelfish에서 지원되는 기능을 참조하세요.
Microsoft SQL Server DW	Amazon Redshift	`AT TIME ZONE` 절에 대한 지원을 추가했습니다.
Microsoft SQL Server DW	Amazon Redshift	`BEGIN/END` 블록 외부의 명령문이 잘못 변환되는 문제를 해결했습니다.
Netezza	Amazon Redshift	`TIME` 데이터 유형의 변환이 개선되고 관련 내장 함수, 표현식 및 리터럴의 변환을 구현했습니다.
Oracle	Aurora PostgreSQL PostgreSQL	Oracle 10g를 소스로 사용할 때 발생한 로더 오류를 수정했습니다.
Oracle	Aurora PostgreSQL PostgreSQL	`OFFSET` 및 `FETCH` 절의 변환을 개선했습니다.

소스	대상	새로운 기능, 향상된 기능 또는 수정된 기능
Oracle	Aurora PostgreSQL PostgreSQL	OUT 파라미터에서 기본값을 사용하는 프로시저가 잘못 변환되는 문제를 해결했습니다.
Oracle DW	Amazon Redshift	Oracle 함수를 Amazon Redshift 사용자 정의 함수로 변환하는 기능을 개선했습니다.
Snowflake	Amazon Redshift	WITH 절의 변환을 개선했습니다.
Teradata	Amazon Redshift	CHAR 데이터 유형에 지원되지 않는 멀티바이트 문자에 대한 새 작업 항목 13209를 추가했습니다.
Teradata	Amazon Redshift	테이블이 완전히 로드되지 않는 로더 오류를 수정했습니다.
Teradata	Amazon Redshift	JOIN 조건의 내장 P_INTERSECT 함수가 변환되지 않는 변환기 오류를 수정했습니다.
Teradata	Amazon Redshift	이름에 특수 문자가 포함된 테이블에서 SELECT 문을 실행할 때 보기 이름이 잘못된 대/소문자로 변환되는 문제를 수정했습니다.
Teradata	Amazon Redshift	UNTIL_CHANGED 값이 PERIOD(DATE) 데이터 유형인 INSERT 문의 변환을 개선했습니다.
Teradata	Amazon Redshift	Amazon Redshift의 FORMAT 함수를 사용하여 내장 TO_CHAR 함수를 변환하는 기능을 개선했습니다.
Teradata	Amazon Redshift	변환된 코드가 소스 코드와 동일한 순서로 NULL 값을 반환하도록 내장 RANK 함수의 변환을 개선했습니다.
Teradata	Amazon Redshift	기본 또는 보조 고유 인덱스 등과 같은 고유 제약 조건의 변환을 개선했습니다.

AWS SCT 빌드 662의 릴리스 노트

소스	대상	새로운 기능, 향상된 기능 또는 수정된 기능
모두	모두	다중 서버 평가 보고서를 만들 때 각 소스 데이터베이스에 대한 AWS SCT 프로젝트를 자동으로 생성하는 기능을 추가했습니다. 이 옵션을 켜면 이러한 프로젝트에 매핑 규칙을 추가하고 오프라인에서 사용할 AWS SCT 수 있도록 전환 통계를 저장할 수 있습니다. 자세한 정보는 데이터베이스 마이그레이션을 위한 다중 서버 평가 보고서 생성을 참조하세요.
모두	모두	다중 서버 평가 보고서를 생성할 때 데이터베이스와 스키마 이름에서 퍼센트(%)를 와일드카드로 사용하는 지원 기능을 구현했습니다.
모두	Aurora MySQL Aurora PostgreSQL	모든 AWS Lambda 함수의 런타임을 Python 버전 3.9로 업데이트했습니다.
모두	Amazon Redshift	모든 데이터 추출 에이전트를 사용할 AWS SDK for Java 2.x수 있도록 업그레이드했습니다.
Azure SQL Database Microsoft SQL Server	Aurora PostgreSQL PostgreSQL	`NOT EXISTS` 절이 있는 `DELETE` 문의 변환을 개선했습니다.
Azure Synapse Analytics	Amazon Redshift	소스 데이터베이스 연결에 실패하는 오류를 해결했습니다.
IBM Db2 for z/OS	Aurora PostgreSQL	변환된 트리거 코드에 객체 별칭에 대한 두 개의 설명이 포함되는 오류를 해결했습니다.

소스	대상	새로운 기능, 향상된 기능 또는 수정된 기능
	PostgreSQL	
Microsoft SQL Server	Aurora PostgreSQL PostgreSQL	Treat database object name as case sensitive 옵션이 켜져 있을 때 이름에 대/소문자가 혼합된 객체의 변환을 개선했습니다.
Microsoft SQL Server DW Teradata	Amazon Redshift	`PIVOT` 및 `UNPIVOT` 관계 연산자의 변환을 구현했습니다.
Netezza	Amazon Redshift	`TIME` 데이터 유형의 변환을 구현했습니다.
Oracle	Aurora MySQL Aurora PostgreSQL MySQL PostgreSQL	`UTL_TCP.CRLF` 패키지 상수 변환을 구현했습니다.
Oracle	Aurora PostgreSQL PostgreSQL	변환 중에 가변 길이의 열에 대한 데이터 유형 길이가 유지되지 않는 확장 팩 문제를 수정했습니다.

소스	대상	새로운 기능, 향상된 기능 또는 수정된 기능
Oracle	Aurora PostgreSQL PostgreSQL	C++ 애플리케이션에서 SQL 코드 변환을 구현했습니다. 자세한 정보는 AWS SCT를 사용하여 C++ 애플리케이션의 SQL 코드 변환을 참조하세요.
Oracle	Aurora PostgreSQL PostgreSQL	전역 변수 및 연관 배열의 변환에 대한 대소문자를 구분한 이름 지정 지원 기능을 구현했습니다.
Oracle	Aurora PostgreSQL PostgreSQL	확장 팩의 `TO_CHAR`, `TO_DATE`, `TO_NUMBER` 함수 변환을 개선했습니다.
Oracle	Aurora PostgreSQL PostgreSQL	`TABLE()` 연산자의 변환을 개선했습니다.
Oracle DW	Amazon Redshift	프라이머리 키 변환 및 기타 제약 조건에 대한 지원을 추가했습니다.
Oracle DW	Amazon Redshift	조건문 변환 중에 작업 항목 12054가 표시되지 않는 문제를 수정했습니다.

소스	대상	새로운 기능, 향상된 기능 또는 수정된 기능
SAP ASE	Aurora PostgreSQL PostgreSQL	사용자 정의 유형의 열이 있는 테이블을 변환하는 동안 대상 트리에서 빈 이름이 있는 객체를 생성할 때 발생하는 오류를 해결했습니다.
SAP ASE	Aurora PostgreSQL PostgreSQL	스크립트, 루틴 등과 같은 저장된 객체에 대한 로더 오류를 수정했습니다.
Snowflake	Amazon Redshift	필요한 경우 작업 항목 22152가 표시되지 않고 변환 결과가 설명으로 AWS SCT 표시되는 문제를 수정했습니다.
Snowflake	Amazon Redshift	날짜 및 시간 함수의 변환을 개선하고 시간대 지원을 구현했습니다.
Snowflake	Amazon Redshift	`WITH` 절이 있는 비재귀적 공통 테이블 표현식(CTE)이 재귀적 CTE로 변환되는 문제를 해결했습니다.
Teradata	Amazon Redshift	조건에 테이블 링크가 있는 `UPDATE` 문의 변환을 개선했습니다.
Teradata	Amazon Redshift	`RENAME TABLE` 문의 변환을 개선했습니다.
Teradata	Amazon Redshift	평가 보고서가 있는 쉼표로 구분된 값(CSV) 파일에 빈 열이 나타나는 문제를 해결했습니다.
Teradata	Amazon Redshift RSQL	변환된 Basic Teradata Query(BTEQ) 매크로의 끝에 세미콜론이 누락되는 오류를 수정했습니다.

소스	대상	새로운 기능, 향상된 기능 또는 수정된 기능
Teradata	Amazon Redshift RSQL	CASE 문의 여러 데이터 유형 값에 대한 변환을 개선했습니다.
Teradata	Amazon Redshift RSQL	ESCAPE 문자가 있는 LIKE ANY 절의 변환을 개선했습니다.
Teradata	Amazon Redshift RSQL	INSERT 문의 CAST 함수 변환을 개선했습니다.
Teradata	Amazon Redshift RSQL	시간대 변환을 개선하고 시간대 리전 매핑을 구현했습니다.
Teradata	Amazon Redshift RSQL	쉘 스크립트를 BTEQ 스크립트로 변환하는 동안 작업 항목 9998이 예기치 않게 나타나는 문제를 해결했습니다.
Teradata	Amazon Redshift RSQL AWS Glue	대체 변수 값에 대해 500자 제한을 구현했습니다.
Vertica	Amazon Redshift	BINARY, VARBINARY , LONG BINARY, BYTEA 및 RAW 데이터 유형을 VARBYTE 데이터 유형으로 변환하도록 구현했습니다.
Vertica	Amazon Redshift	내장 함수 및 리터럴의 변환을 개선했습니다.

빌드 661의 AWS SCT 릴리스 노트

소스	대상	새로운 기능, 향상된 기능 또는 수정된 기능
모두	모두	매핑 보기에서 매핑 규칙을 검색하는 필터를 추가했습니다. 필터를 적용하면 서버 매핑 목록의 필터링 조건과 일치하는 규칙이 AWS SCT 표시됩니다. 자세한 정보는 매핑 규칙 관리을 참조하세요.
모두	모두	Apache Log4j를 버전 2.17.1로 업그레이드했습니다.
모두	Amazon Redshift	COPY 명령의 ENCRYPTED 절을 사용하여 Amazon Redshift로의 데이터 마이그레이션 지원을 추가했습니다.
모두	Amazon Redshift	데이터 추출 에이전트의 REST API를 개선했습니다. 업데이트된 REST API는 암호화 키, 암호화 유형 등과 같은 새로운 속성에 대한 지원을 추가합니다.
모두	Amazon Redshift	데이터 추출 에이전트에서 역할 수임을 구현했습니다. 이 업데이트를 통해 하위 작업의 분배를 개선하고 AWS SCT 가 지정된 역할의 무료 에이전트에게 작업을 할당할 수 있습니다.
모두	Amazon Redshift	Amazon Redshift에 확장 팩을 적용하기 전에 필요한 모든 구성 요소가 설치되었는지 확인하는 검사 기능을 구현했습니다.
Azure Synapse Analytics Microsoft SQL Server DW	Amazon Redshift	오류 처리를 위해 ERROR_LINE , ERROR_MESSAGE , ERROR_NUMBER , ERROR_PROCEDURE , ERROR_SEVERITY 및 ERROR_STATE 시스템 함수의 변환을 개선했습니다.
IBM Db2 for z/OS	Aurora MySQL Aurora PostgreSQL	AWS SCT에서 IBM Db2 for z/OS 버전 12를 데이터베이스 마이그레이션 프로젝트의 소스로 사용할 수 있는 지원 기능을 추가했습니다. 자세한 정보는 IBM Db2 for z/OS를 소스로 사용을 참조하세요.

소스	대상	새로운 기능, 향상된 기능 또는 수정된 기능
	MySQL PostgreSQL	
IBM Db2 LUW	모두	열 이름과 중복되는 루틴 매개 변수를 로드할 수 있도록 AWS SCT 원본 메타데이터 로더를 개선했습니다.
Microsoft Azure SQL Database Microsoft SQL Server	Aurora PostgreSQL PostgreSQL	SET NOCOUNT ON 집합 문을 사용하는 프로시저의 변환기 오류를 수정했습니다.
Microsoft Azure SQL Database Microsoft SQL Server	Aurora PostgreSQL PostgreSQL	입력 값이 사용자 정의 유형의 변수인 경우 CONCAT 함수 변환을 개선했습니다.
Microsoft Azure SQL Database Microsoft SQL Server	Aurora PostgreSQL PostgreSQL	DATEPART 함수가 잘못 변환되는 문제를 해결했습니다.
Microsoft SQL Server	Babelfish for Aurora PostgreSQL	Babelfish 기능 구성 파일의 새 버전에 대한 지원을 개선했습니다. 이 파일은 특정 Babelfish 버전에서 지원되거나 지원되지 않는 SQL 기능을 정의합니다.

소스	대상	새로운 기능, 향상된 기능 또는 수정된 기능
Microsoft SQL Server DW	Amazon Redshift	EXECUTE 문이 있는 프로시저가 잘못 변환되는 문제를 해결했습니다.
Microsoft SSIS	AWS Glue	작업 구성 마법사의 사용자 인터페이스를 개선했습니다. AWS SCT 이제 연결 구성 섹션에 사용 가능한 연결만 표시됩니다.
Microsoft SSIS	AWS Glue	변환 규칙이 패키지 작업 및 변수 규칙에 적용되지 않는 문제를 해결했습니다.
Microsoft SSIS	AWS Glue AWS Glue Studio	지원되지 않는 구성 요소에 대해 새로운 작업 항목 25042를 추가했습니다.
Microsoft SSIS	AWS Glue Studio	Microsoft SQL 서버 통합 서비스 (SSIS) 추출, 변환 및 로드 (ETL) 패키지를 로 변환하도록 구현했습니다. AWS Glue Studio자세한 정보는 SSIS를 AWS Glue Studio로 변환을 참조하세요.
Oracle	MariaDB	MINUS 연산자 변환과 관련된 문제를 수정했습니다.
Oracle	MariaDB	MariaDB의 sql_mode 시스템 변수가 Oracle인 경우 ROWNUM, SYS_GUID, TO_CHAR 및 ADD_MONTHS 함수의 변환을 개선했습니다.
Oracle	PostgreSQL	일반 애플리케이션 변환 프로젝트에서 바인드 변수 유형을 SQL 유형으로 변환하지 않도록 하는 옵션을 추가했습니다.
Oracle	PostgreSQL	일반 애플리케이션 변환 프로젝트에서 변환된 객체의 이름에 스키마 이름이 추가되지 않도록 하는 옵션을 추가했습니다.
Oracle	PostgreSQL	애플리케이션 SQL 코드 변환을 위한 ?x 바인드 변수 형식 지원을 추가했습니다.
Oracle DW	Amazon Redshift	RAW 데이터 유형을 VARBYTE 데이터 유형으로 변환하도록 구현했습니다.

소스	대상	새로운 기능, 향상된 기능 또는 수정된 기능
Teradata	Amazon Redshift	변환된 코드에서 SET 테이블을 에뮬레이션하는 옵션을 추가했습니다. 이 에뮬레이션의 경우 AWS SCT 지원 MIN 및 조건이 적용됩니다. MAX
Teradata	Amazon Redshift	다양한 데이터 유형의 파라미터가 있는 조인 작업의 변환을 개선했습니다. 이 업데이트를 통해 AWS SCT 이러한 작업을 변환하는 동안 변환 규칙을 적용할 수 있습니다.
Teradata	Amazon Redshift	GROUP BY 절이 잘못 변환되는 문제를 해결했습니다.
Teradata	Amazon Redshift	QUALIFY 절이 잘못 변환되는 문제를 해결했습니다.
Teradata	Amazon Redshift	FastExport 스크립트를 가져오는 동안 예상치 못한 오류가 발생하던 문제를 해결했습니다.
Teradata	Amazon Redshift RSQL	Teradata BTEQ 및 쉘 스크립트에서 변수 값을 편집하는 기능을 구현했습니다.
Teradata	Amazon Redshift RSQL	변환된 Teradata FastLoad 세션에서 매니페스트 스크립트가 누락되는 문제가 해결되었습니다.
Teradata	Amazon Redshift RSQL	변환된 스크립트의 유니폼 리소스 로케이터 (URL) 에서 매니페스트 파일의 확장자가 누락되는 문제가 해결되었습니다. FastLoad
Teradata BTEQ	Amazon Redshift RSQL	대체 변수가 있는 스크립트의 로더 오류를 수정했습니다.
Teradata BTEQ	Amazon Redshift RSQL	필요할 때 작업 항목 27022가 표시되지 않는 문제를 수정했습니다.

빌드 660의 AWS SCT 출시 노트

소스	대상	새로운 기능, 향상된 기능 또는 수정된 기능
모두	모두	다중 서버 평가 AWS Secrets Manager 보고서에 보안 소켓 계층 (SSL) 에 대한 지원이 추가되었습니다. 자세한 정보는 데이터베이스 마이그레이션을 위한 다중 서버 평가 보고서 생성을 참조하세요.
모두	모두	변환된 객체에 대한 통계 수집을 개선했습니다.
모두	PostgreSQL	PostgreSQL 메이저 버전 14와 MariaDB 10.6을 마이그레이션 대상으로 사용하는 기능에 대한 지원을 구현했습니다.
Azure Synapse Analytics	Amazon Redshift	변환된 객체의 이름에 대한 변환 로직을 개선했습니다.
Microsoft Azure SQL Database Microsoft SQL Server	Aurora PostgreSQL	XML 데이터 유형에 대한 변환을 개선했습니다.
Microsoft Azure SQL Database Microsoft SQL Server	Aurora PostgreSQL PostgreSQL	NOT LIKE 절이 잘못 변환되는 문제를 해결했습니다.
Microsoft Azure SQL Database	Aurora PostgreSQL PostgreSQL	OUTPUT 절을 포함하는 INSERT, DELETE 및 UPDATE 문이 있는 프로시저에 대한 변환기 오류를 수정했습니다.

소스	대상	새로운 기능, 향상된 기능 또는 수정된 기능
Microsoft SQL Server		
Microsoft Azure SQL Database Microsoft SQL Server	Aurora PostgreSQL PostgreSQL	`RETURN @@ROWCOUNT` 문을 사용하는 프로시저의 변환기 오류를 수정했습니다.
Microsoft SQL Server	모두	연결된 서버를 사용하는 프로시저의 변환을 개선했습니다.
Microsoft SQL Server	모두	다중 서버 평가 보고서에 Microsoft Windows 인증에 대한 지원을 추가했습니다.
Microsoft SQL Server	Aurora PostgreSQL PostgreSQL	테이블 값 생성자의 변환기 오류를 수정했습니다.
Microsoft SQL Server DW	아마존 Redshift와 AWS Glue	변환된 스크립트에 대한 올바른 경로를 포함하도록 추출, 전환, 적재 (ETL) 스크립트의 변환을 개선했습니다.
Microsoft SQL Server DW	Amazon Redshift	가상 및 실제 대상 데이터베이스 플랫폼에 대해 서로 다른 변환된 스크립트가 생성되는 문제를 해결했습니다.

소스	대상	새로운 기능, 향상된 기능 또는 수정된 기능
Oracle	PostgreSQL Aurora PostgreSQL	구체화된 뷰의 인덱스 변환에 대한 지원을 추가했습니다.
Oracle	PostgreSQL Aurora PostgreSQL	NOVALIDATE 옵션으로 PRIMARY KEY 및 UNIQUE 제약 조건을 변환할 때 작업 항목 5982가 표시되지 않는 문제를 수정했습니다.
Oracle DW	Amazon Redshift	변환된 스키마에 추가적인 범주가 표시되는 문제를 해결했습니다.
Teradata	Amazon Redshift	해결되지 않은 열을 CAST 함수의 인수로 변환할 때 작업 항목 13185가 표시되지 않는 문제를 수정했습니다.
Teradata	Amazon Redshift	변환된 코드에서 TRUNCATE 명령을 사용하도록 DELETE 및 DELETE ALL 문을 변환하는 기능을 개선했습니다.
Teradata	Amazon Redshift	SET 테이블 변환을 개선했습니다.
Teradata	Amazon Redshift	NORMALIZE 조건 변환을 개선했습니다.
Teradata	Amazon Redshift	데이터베이스 스토리지 객체 목록에서 데이터베이스 스키마 변환 통계를 제거하도록 평가 보고서를 업데이트했습니다.
Teradata	Amazon Redshift	FROM 절이 없는 UPDATE 문의 변환을 개선했습니다.
Teradata	Amazon Redshift	변환된 코드의 VARBYTE 데이터 유형에 대한 지원을 구현했습니다.

소스	대상	새로운 기능, 향상된 기능 또는 수정된 기능
Teradata BTEQ	AWS Glue	컨텍스트 메뉴에서 Convert to AWS Glue 옵션이 비활성화되는 문제를 해결했습니다.
Teradata BTEQ	Amazon Redshift RSQL	변환된 코드에서 데이터 유형이 누락되는 문제를 해결했습니다.
Teradata BTEQ	Amazon Redshift RSQL	변환된 코드에서 대체 변수가 잘못 인용되는 문제를 해결했습니다.
Teradata BTEQ	Amazon Redshift RSQL	스크립트에서 FastLoad 대체 변수를 값으로 변환할 때 발생하는 문제를 수정했습니다.
Vertica	Amazon Redshift	변환된 코드의 `TIME` 데이터 유형에 대한 지원을 구현했습니다.
Vertica	Amazon Redshift	`SELECT DISTINCT` 및 `ORDER BY` 표현식의 변환을 개선했습니다.
Vertica	Amazon Redshift	제약 조건 변환에 대한 지원을 추가했습니다.
Vertica	Amazon Redshift	평가 보고서가 쉼표로 구분된 값(CSV) 파일로 저장되지 않는 오류를 해결했습니다.

AWS SCT 빌드 659의 릴리스 노트

소스	대상	새로운 기능, 향상된 기능 또는 수정된 기능
모두	모두	여러 소스 데이터베이스에 대한 통합 평가 보고서를 생성하는 New project wizard를 개선했습니다.
모두	모두	여러 소스 및 대상 데이터베이스가 포함된 프로젝트에서 확장 팩이 생성되지 않는 문제를 수정했습니다.

소스	대상	새로운 기능, 향상된 기능 또는 수정된 기능
모두	모두	애플리케이션 소스 코드에 포함된 SQL 코드의 변환을 개선했습니다.
모두	모두	AWS SCT 명령줄 인터페이스의 여러 폴더에서 스크립트를 실행할 수 있는 기능을 추가했습니다.
모두	Amazon Redshift	사용자가 Amazon Redshift 가상 대상 데이터베이스 플랫폼을 사용하여 마이그레이션 프로젝트에서 Run optimization을 선택할 때 표시되는 경고 메시지를 개선했습니다.
모두	Aurora PostgreSQL	Aurora PostgreSQL-Compatible Edition의 PostgreSQL 메이저 버전 13을 마이그레이션 대상으로 사용하는 기능에 대한 지원을 구현했습니다.
모두	Amazon RDS for MySQL	기본적으로 대소문자를 구분하지 않는 코드 변환을 구현했습니다.
Azure Synapse Analytics	Amazon Redshift	명령줄 인터페이스에서 소스 데이터베이스 연결에 실패하는 오류를 해결했습니다.
Microsoft SQL Server	PostgreSQL Aurora PostgreSQL	조인 조건이 있는 `UPDATE` 문을 포함하는 프로시저의 변환을 개선했습니다.
Microsoft SQL Server	PostgreSQL Aurora PostgreSQL	등호 뒤에 값이 포함된 트리거, 저장 프로시저 및 함수의 변환을 개선했습니다.

소스	대상	새로운 기능, 향상된 기능 또는 수정된 기능
Microsoft SQL Server	PostgreSQL Aurora PostgreSQL	DELETE 문 및 OR 연산자를 사용하는 프로시저의 변환기 오류를 수정했습니다.
Microsoft SQL Server	PostgreSQL Aurora PostgreSQL	OUTPUT 절의 변환을 개선했습니다.
Microsoft SQL Server DW	아마존 Redshift와 AWS Glue	NUMERIC 데이터 유형에 대한 변환을 개선했습니다.
Microsoft SQL Server DW	Amazon Redshift	원래 테이블과 같은 이름의 테이블 별칭이 있는 보기의 변환을 개선했습니다.
Microsoft SSIS	AWS Glue	연결 구성 창에 AWS Glue 연결 자격 증명이 표시되지 않는 문제를 수정했습니다.
Netezza	Amazon Redshift	변경 데이터 캡처(CDC) 데이터 마이그레이션 작업을 매일 반복해서 실행하는 기능을 추가했습니다.
Netezza	Amazon Redshift	데이터 추출 에이전트를 등록 취소한 후 작업 탭이 비활성화되는 문제를 수정했습니다.
Netezza	Amazon Redshift	데이터 마이그레이션 에이전트의 등록 확인이 사용자 인터페이스에 표시되지 않는 문제를 수정했습니다.
Netezza	Amazon Redshift	로더 오류로 소스 데이터베이스 연결이 실패하는 문제를 수정했습니다.

소스	대상	새로운 기능, 향상된 기능 또는 수정된 기능
Netezza	Amazon Redshift	저장된 프로젝트를 연 후 데이터 마이그레이션 에이전트가 실행되지 않는 오류를 해결했습니다.
Oracle	Amazon RDS for Oracle	Oracle Unified Auditing 지원을 구현했습니다.
Oracle	PostgreSQL Aurora PostgreSQL	C# 애플리케이션에서 SQL 코드 변환을 구현했습니다. 자세한 정보는 C# 애플리케이션의 SQL 코드 변환을 참조하세요.
Oracle	PostgreSQL Aurora PostgreSQL	코드 변환 변경의 가시성을 개선하기 위해 대소문자를 구분하는 개체 이름에 대한 새로운 변환 로직을 구현했습니다. AWS SCT 객체 이름을 대문자로 소문자로 변환합니다. 반대의 경우도 마찬가지로, AWS SCT 가 소문자로 된 객체 이름을 대문자로 변환합니다. 다른 객체 이름 및 예약어는 변경 없이 변환됩니다.
Oracle	PostgreSQL Aurora PostgreSQL	NOT NULL 제약 조건 없이 해시 파티션을 변환할 수 있도록 개선했습니다.
Oracle	Aurora PostgreSQL	ENABLE NOVALIDATE 절을 통해 Oracle CHECK, FOREIGN KEY, NOT NULL 제약 조건의 변환을 지원하는 기능을 추가했습니다.
Oracle DW	Amazon Redshift	부동 소수점 숫자의 잘못된 값이 마이그레이션되는 문제를 수정했습니다.

소스	대상	새로운 기능, 향상된 기능 또는 수정된 기능
Oracle DW	아마존 Redshift와 AWS Glue	쉼표로 구분된 값(CSV) 파일의 데이터베이스 마이그레이션 평가 보고서에 빈 열이 있는 문제를 해결했습니다.
SAP ASE	PostgreSQL Aurora PostgreSQL	예기치 못한 변환 중단 문제를 수정했습니다.
Snowflake	Amazon Redshift	`VARIANT` 데이터 유형에 대한 변환을 개선했습니다.
Teradata	Amazon Redshift	`COLLECT STATISTICS` 문의 변환을 개선했습니다.
Teradata	Amazon Redshift	`PERIOD` 열이 있는 중첩된 보기를 변환할 때 작업 항목 9998이 표시되지 않는 문제를 수정했습니다.
Teradata	아마존 Redshift와 AWS Glue	저장된 프로젝트를 연 후 UI에 가상 AWS Glue 대상 플랫폼이 표시되지 않는 문제를 수정했습니다.
Teradata BTEQ	AWS Glue	저장된 프로젝트를 연 후 가상 AWS Glue 타겟 플랫폼으로의 변환이 지원되지 않던 문제를 수정했습니다.
Teradata BTEQ	Amazon Redshift RSQL	변환된 코드의 구문 강조 표시를 개선했습니다.
Teradata BTEQ	Amazon Redshift RSQL	업로드 후 파라미터 값 검사 기능을 구현했습니다. 지원되지 않는 값은 변수 탭에서 강조 표시됩니다.
Vertica	Amazon Redshift	집계 함수의 변환을 구현했습니다.

소스	대상	새로운 기능, 향상된 기능 또는 수정된 기능
Vertica	Amazon Redshift	프로젝션을 구체화된 뷰로 변환하는 기능을 구현하고 프로젝션의 소스 코드를 표시하는 UI를 개선했습니다.

AWS SCT 빌드 658의 릴리스 노트

소스	대상	새로운 기능, 향상된 기능 또는 수정된 기능
모두	모두	와 통합이 AWS Secrets Manager제공되었습니다. 이제 Secrets Manager에 저장된 데이터베이스 연결 보안 인증 정보를 사용할 수 있습니다.
모두	모두	AWS SCT 명령줄 인터페이스에 YAML 형식의 스크립트에 대한 지원이 추가되었습니다.
모두	Amazon Redshift	데이터 추출 에이전트에서 Amazon S3 인터페이스 엔드포인트(VPCE)에 대한 지원을 구현했습니다.
모두	Amazon Redshift	이미 지원되는 Amazon Redshift 및 조합 외에도 Amazon Redshift 가상 대상 데이터베이스 플랫폼에 대한 지원이 추가되었습니다. AWS Glue
Greenplum	Amazon Redshift	SQL로 저장 옵션이 변환된 SQL 코드를 파일에 저장하지 않는 문제를 수정했습니다.
IBM Db2 LUW Microsoft Azure SQL Database Microsoft SQL Server Oracle	Aurora MySQL	MySQL 8.0 호환성이 있는 Amazon Aurora MySQL-Compatible Edition의 새로운 기능을 지원하도록 변환을 개선했습니다.

소스	대상	새로운 기능, 향상된 기능 또는 수정된 기능
SAP ASE		
Microsoft SQL Server	Aurora MySQL Aurora PostgreSQL MySQL PostgreSQL	필요할 때 작업 항목 810이 표시되지 않는 문제를 수정했습니다.
Microsoft SQL Server	Aurora PostgreSQL PostgreSQL	UPDATE, DELETE 및 INSERT 문을 사용한 프로시저의 변환을 개선했습니다.
Microsoft SQL Server	Aurora PostgreSQL PostgreSQL	필요할 때 작업 항목 7810이 표시되지 않는 문제를 수정했습니다.
Microsoft SQL Server	Aurora PostgreSQL PostgreSQL	IF...ELSE 문 내에 중첩된 EXEC 문의 변환을 개선했습니다.

소스	대상	새로운 기능, 향상된 기능 또는 수정된 기능
Microsoft SQL Server	Aurora PostgreSQL PostgreSQL	인덱싱된 보기의 변환을 개선했습니다.
Netezza	Amazon Redshift	변경 데이터 캡처(CDC) 작업의 전체 로드 중에 실시간 트랜잭션을 추적하여 데이터 마이그레이션 에이전트를 개선했습니다. 이제 CDC 세션이 특정 시간에 시작되도록 예약된 경우 데이터 마이그레이션 작업을 중지할 수 있습니다. 또한 CDC에서 작업을 중지한 후 콘솔에서 오류 로깅 수준을 확인할 수 있습니다.
Oracle	모두	공유 옵션을 사용하여 객체를 AWS SCT 로드할 수 있도록 테이블 로더를 개선했습니다.
Oracle	Aurora PostgreSQL PostgreSQL	SYSDATE 함수 변환을 개선하고 변환 설정에서 시간대를 변경할 수 있는 기능을 추가했습니다.
Oracle	Aurora PostgreSQL PostgreSQL	동적 명령문이 변환되지 않는 문제를 해결했습니다.
Oracle	Aurora PostgreSQL PostgreSQL	시스템에서 생성한 이름이 변환된 코드에 포함되지 않는 문제를 수정했습니다.

소스	대상	새로운 기능, 향상된 기능 또는 수정된 기능
Oracle Oracle DW	Aurora PostgreSQL PostgreSQL	트리거 내에 중첩된 SELECT 문의 변환을 개선했습니다.
Oracle DW	Amazon Redshift	확장 팩의 TO_DATE, TO_TIMESTAMP , TO_TIMESTAMP_TZ 함수 변환을 개선했습니다.
Snowflake	Amazon Redshift	변환된 SQL 코드를 각 객체 또는 각 명령문에 대해 서로 다른 파일에 저장하는 옵션을 추가했습니다.
Teradata	Amazon Redshift	CONCAT 함수에 대한 변환을 개선했습니다.
Teradata	Amazon Redshift	WHERE 절 내에 중첩된 SELECT 문의 변환을 개선했습니다.
Teradata	Amazon Redshift	사용자가 테이블을 삭제하고 다시 생성한 후 SET 및 MULTISET 테이블이 잘못 변환되는 문제를 해결했습니다.
Teradata	Amazon Redshift	WITH 절이 포함된 프로시저의 변환을 개선했습니다.
Teradata	Amazon Redshift	DATE 데이터 유형에 대한 변환을 개선했습니다.
Teradata	Amazon Redshift RSQL	FastExport 스크립트 변환 중에 예상치 못한 변환기 오류가 발생하는 문제가 해결되었습니다.
Teradata BTEQ	Amazon Redshift RSQL	조인 인덱스를 구체화된 뷰로 변환하는 지원 기능을 추가했습니다.

소스	대상	새로운 기능, 향상된 기능 또는 수정된 기능
Teradata BTEQ	Amazon Redshift RSQL	여러 행을 포함하는 TITLE 정의의 변환에 대한 지원을 추가했습니다.
Teradata BTEQ	Amazon Redshift RSQL	지리 공간 데이터 유형의 크기가 변환되지 않는 문제를 해결했습니다.
Teradata BTEQ	Amazon Redshift RSQL	파라미터 이름이 소문자로 변환되는 문제를 수정했습니다.
Teradata BTEQ	Amazon Redshift RSQL	MACRO 문 내에 중첩된 저장 프로시저가 변환되지 않는 문제를 수정했습니다.
Vertica	Amazon Redshift	ALL 연산자의 변환을 개선했습니다.
Vertica	Amazon Redshift	변환 설정의 Use Union all view? 옵션이 적용되지 않는 문제를 해결했습니다.
Vertica	Amazon Redshift	TIME 및 TIME WITH TIMEZONE 데이터 유형에 대한 변환을 개선했습니다.
Vertica	Amazon Redshift	플렉스 테이블 로드 관련 문제를 해결했습니다.

해결된 문제:

- 일반적인 개선 사항

AWS SCT 빌드 657의 릴리스 노트

소스	대상	새로운 기능, 향상된 기능 또는 수정된 기능
모두	모두	보안 취약성 문제를 해결하기 위해 Apache Log4j를 버전 2.17로 업그레이드했습니다.
모두	Amazon Redshift	키 관리 통계가 AWS SCT 프로젝트에 저장되지 않았던 스키마 최적화 프로젝트를 개선했습니다.
Amazon Redshift	Amazon Redshift	서버 정보 업데이트 관련 문제를 수정했습니다.
Apache Cassandra	Amazon DynamoDB	AWS SCT 명령줄 인터페이스를 사용할 때 매핑 규칙 관련 문제가 해결되었습니다.
Apache Cassandra	Amazon DynamoDB	인증서의 업데이트된 제목으로 인해 마이그레이션 작업이 생성되지 않는 문제를 해결했습니다.
Microsoft SQL Server	Aurora PostgreSQL PostgreSQL	Microsoft SQL Server 프로시저를 동적 SQL로 변환하는 동안 작업 항목 7672가 표시되지 않도록 문제를 수정했습니다.
Azure SQL Database Microsoft SQL Server	Aurora PostgreSQL PostgreSQL	테이블 반환 함수의 변환을 개선했습니다.
Azure SQL Database Microsoft SQL Server	Aurora PostgreSQL PostgreSQL	기본 반환 값이 있는 저장 프로시저의 `OUT` 인수가 `INOUT` 인수로 변환되지 않는 문제를 해결했습니다.

소스	대상	새로운 기능, 향상된 기능 또는 수정된 기능
Greenplum	Amazon Redshift	`QueryLog` 테이블에서 가장 많이 사용되는 테이블과 열을 찾도록 최적화 전략을 개선했습니다.
Microsoft SQL Server	Aurora PostgreSQL PostgreSQL	다음 항목의 변환 관련 문제를 수정했습니다. • 문자열 연결 할당 연산자(+=) • `SCOPE_IDENTITY` 함수 • `varchar(max)` 데이터 유형
Microsoft SQL Server	Aurora PostgreSQL PostgreSQL	지원되지 않는 함수가 있는 보기의 변환을 개선했습니다.
Microsoft SQL Server	Aurora PostgreSQL PostgreSQL	지원되지 않는 함수가 다른 함수에 대한 인수로 잘못 변환되는 문제를 수정했습니다.
Microsoft SQL Server	Babelfish for Aurora PostgreSQL	전환 테이블 참조의 변환을 개선했습니다.
Microsoft SQL Server DW	Amazon Redshift	집계 함수 범주를 소스 데이터베이스 메타데이터 트리에 추가했습니다.

소스	대상	새로운 기능, 향상된 기능 또는 수정된 기능
Microsoft SQL Server DW	Amazon Redshift	TIME 데이터 유형에 대한 변환을 개선했습니다.
Azure Synapse Analytics Greenplum Netezza Microsoft SQL Server DW Snowflake Teradata	Amazon Redshift	가상 대상 데이터베이스 플랫폼을 사용할 때 DROP 및 CREATE 스크립트가 저장되지 않는 문제를 수정했습니다.
Microsoft SQL Server Integration Services	AWS Glue	소스 객체의 스크립트가 UI에 표시되지 않는 문제를 해결했습니다.
Netezza	Amazon Redshift	팩트 테이블 및 공동 배치에 적절한 차원을 선택하여 최적화 전략을 개선했습니다.
Oracle	Aurora PostgreSQL PostgreSQL	시퀀스 번호를 사용하는 Oracle 트리거가 올바르게 변환되도록 문제를 해결했습니다.

소스	대상	새로운 기능, 향상된 기능 또는 수정된 기능
Oracle	Aurora PostgreSQL PostgreSQL	퍼블릭 데이터베이스 링크가 포함된 보기 변환을 개선했습니다.
Oracle DW	Amazon Redshift	인덱스 열의 카디널리티 분석을 통해 최적화 전략을 개선했습니다.
Oracle DW	Amazon Redshift	문자열 연결을 사용하는 사용자 정의 스칼라 함수가 잘못 변환되는 문제를 수정했습니다.
Snowflake	Amazon Redshift	SQL로 저장 옵션이 UI에 표시되지 않는 문제를 수정했습니다.
Teradata	Amazon Redshift	`LOADER ERROR` 예외가 발생하여 통계 수집에 실패하는 문제를 수정했습니다.
Teradata	Amazon Redshift	UI에 보고서 생성 옵션이 표시되지 않는 문제를 수정했습니다.
Teradata	Amazon Redshift	`CAST` 함수에 대한 변환을 개선했습니다.
Teradata	Amazon Redshift	`ST_Line_Interpolate_Point` 에 대한 손상된 변환 문제를 수정했습니다.
Teradata	Amazon Redshift	Python 라이브러리 경로에서 예기치 않은 값을 제거했습니다.
Teradata	Amazon Redshift RSQL	여러 스크립트를 변환하는 동안 나타나는 리졸버 오류를 수정했습니다. FastLoad

소스	대상	새로운 기능, 향상된 기능 또는 수정된 기능
Teradata BTEQ	Amazon Redshift RSQL	DATABASE 명령 및 지오메트리 데이터 유형의 변환을 개선했습니다.
Teradata BTEQ	AWS Glue	UI에서 소스 및 대상 스크립트가 잘못 동기화되는 문제를 수정했습니다.

해결된 문제:

- 일반적인 개선 사항

빌드 656의 AWS SCT 릴리스 노트

소스	대상	새로운 기능, 향상된 기능 또는 수정된 기능
모두	모두	한 프로젝트 내의 여러 소스 및 대상 데이터베이스에 대한 지원을 추가했습니다. 이제 사용자는 동일한 프로젝트의 여러 데이터베이스 스키마와 대상 플랫폼을 일치시키는 매핑 규칙을 만들 수 있습니다.
모두	모두	가상 대상 데이터베이스 플랫폼에 대한 지원을 추가했습니다. 이제 사용자는 대상 데이터베이스에 연결하지 않아도 원본 데이터베이스 스키마가 어떻게 AWS SCT 변환되는지 확인할 수 있습니다.
모두	모두	UI 개선 사항: • 소스 및 대상 메타데이터 트리에 Connect to the server 및 Disconnect from the server 옵션을 추가했습니다. • AWS SCT 프로젝트에서 데이터베이스 서버를 제거하는 옵션을 추가했습니다.
Cassandra	Amazon DynamoDB	CASSANDRA_HOME 변수에서 cassandra.yaml 또는 conf 폴더 뒤에 슬래시(/)가 포함되지 않는 검색 문제를 해결했습니다.

소스	대상	새로운 기능, 향상된 기능 또는 수정된 기능
Cassandra	Amazon DynamoDB	Amazon Linux 2에 대한 Amazon Machine Image(AMI) 지원을 추가했습니다.
Cassandra	Amazon DynamoDB	Cassandra에 잘못된 키가 제공되었을 때 표시되는 오류 메시지를 개선했습니다.
Cassandra	Amazon DynamoDB	대상 데이터베이스의 버전에 따라 `cassandra-env.yaml` 파일의 속성을 변경하여 변환을 개선했습니다.
Cassandra	Amazon DynamoDB	대상 Cassandra Datacenter의 Java 버전을 1.8.0으로 증가시켰습니다.
Greenplum	Amazon Redshift	프로젝트 설정에서 최적화 전략을 개선했습니다.
Greenplum	Amazon Redshift	`An I/O error occurred while sending to the backend` 오류가 발생하여 객체가 데이터베이스에 적용되지 않는 데이터 마이그레이션 문제를 해결했습니다.
Greenplum Microsoft SQL Server DW	Amazon Redshift	UI에 `Apply RTRIM to string columns` 옵션이 표시되지 않는 문제를 해결했습니다.
Microsoft SQL Server	Babelfish for Aurora PostgreSQL	Babelfish for Aurora PostgreSQL을 대상 플랫폼으로 사용하기 위한 지원을 추가했습니다. 이제 사용자는 평가 보고서를 생성하여 SQL Server에서 Babelfish for Aurora PostgreSQL로의 마이그레이션을 평가할 수 있습니다.
Netezza	Amazon Redshift	프로젝트 설정에서 최적화 전략을 개선했습니다.

소스	대상	새로운 기능, 향상된 기능 또는 수정된 기능
SAP ASE	Aurora PostgreSQL PostgreSQL	인덱스에 고유한 이름을 생성하는 기능을 구현했습니다.
SAP ASE	Aurora PostgreSQL PostgreSQL	대상 스크립트에서 인덱스 열이 중복되는 문제를 수정했습니다.
Snowflake	Amazon Redshift	빈 스키마 숨기기, 빈 데이터베이스 숨기기 및 Hide system databases /schemas 옵션이 UI에 표시되지 않는 문제를 해결했습니다.
Teradata	Amazon Redshift RSQL	테라데이타 MultiLoad 작업 스크립트를 Amazon Redshift RSQL 스크립트로 변환하기 위한 지원이 추가되었습니다.
Teradata	Amazon Redshift RSQL	및 스크립트에서 대체 변수를 변환할 때 발생하는 문제를 수정했습니다. FastLoad FastExport
Teradata	Amazon Redshift RSQL	요약 탭에서 전환한 후 작업 항목 탭에 작업 항목이 표시되지 않는 문제를 수정했습니다.
Teradata	Amazon Redshift RSQL	FastExport 스크립트 변환 중에 보고서를 생성한 후 오류가 발생하는 문제가 해결되었습니다.
Teradata	Amazon Redshift RSQL	쉘 스크립트 변환 후의 형식 지정 문제를 해결했습니다.

소스	대상	새로운 기능, 향상된 기능 또는 수정된 기능
Teradata	Amazon Redshift RSQL	이제 변환된 스크립트에서 AI 13177이 주석으로 처리되도록 문제를 수정했습니다.
Teradata	Amazon Redshift	임시 테이블의 손상된 변환 문제를 수정했습니다.
Teradata	Amazon Redshift	`SET QUERY_BAND` 문의 변환을 개선했습니다.
Teradata	Amazon Redshift	`NORMALIZE` 작업의 손상된 변환 문제를 수정했습니다.
Vertica	Amazon Redshift	AI 17008의 설명을 개선했습니다.

해결된 문제:

- 일반적인 개선 사항

AWS SCT 빌드 655의 릴리스 노트

소스	대상	새로운 기능, 향상된 기능 또는 수정된 기능
Teradata	Amazon Redshift RSQL	사용 시 FastLoad 모든 평가 문제가 보고서에 표시되도록 문제를 수정했습니다. MultiLoad
Teradata	Amazon Redshift RSQL	테라데이타 FastExport 작업 스크립트를 Amazon Redshift RSQL 스크립트로 변환하기 위한 지원이 추가되었습니다.
Teradata	Amazon Redshift RSQL	를 사용할 때 매니페스트를 S3로 저장 작업을 오프라인 모드에서 활성화하도록 하는 문제를 수정했습니다. FastLoad

소스	대상	새로운 기능, 향상된 기능 또는 수정된 기능
Teradata	Amazon Redshift RSQL	와 같은 FastLoad 스크립트에 매핑 규칙이 적용되도록 하는 문제를 수정했습니다.
Greenplum	Amazon Redshift	Greenplum에 대해 지원되는 최소 드라이버 버전을 42.2.5로 높였습니다.
Greenplum	Amazon Redshift	드라이버 버전 42.2.5 이상에서 SSL을 통한 Greenplum 연결을 추가했습니다.
Oracle DW	Amazon Redshift	다른 UDF 내에서 사용자 정의 스칼라 함수(UDF)를 실행하기 위한 지원을 개선했습니다.
Oracle DW	Amazon Redshift	`Failed to compile udf` 오류가 발생하여 함수가 데이터베이스에 적용되지 않는 문제를 수정했습니다.
Oracle DW	Amazon Redshift	%ROWTYPE 파라미터에 대한 `pls-type` 등과 같은 적절한 형식 선언을 사용하여 변환을 개선했습니다.
Teradata	Amazon Redshift RSQL	정보 유형 평가 문제가 보고서에 표시되지 않는 문제를 해결했습니다.
Teradata	Amazon Redshift RSQL	일부 스크립트를 변환한 후 발생하는 변환기 오류를 해결했습니다.
Teradata	Amazon Redshift RSQL	이제 변환된 스크립트에서 문제가 주석으로 처리되도록 문제를 수정했습니다.
Teradata	Amazon Redshift	변환 후 'CAST' 대신 FastExport ->내보내기 -> 'null'이 표시되는 문제가 해결되었습니다.
Teradata	Amazon Redshift	드라이버 버전 1.2.43을 사용하는 경우 `Cause:[JDBC Driver]String index out of range: 0` 를 적용하면 확장 팩의 일부 기능이 실패하는 문제를 해결했습니다.

소스	대상	새로운 기능, 향상된 기능 또는 수정된 기능
Teradata	Amazon Redshift	SET 테이블 변환—insert-select 문에 SET 테이블 에뮬레이션을 추가했습니다.
Teradata	Amazon Redshift	CAST—추가 데이터 유형 캐스팅을 지원합니다.
Teradata	Amazon Redshift	"other_current_time_01"의 손상된 변환 문제를 수정했습니다.
Teradata	Amazon Redshift	테라데이타 FastExport — 아마존 Redshift RSQL: 테라데이타 명령의 향상된 변환 — 필드 FastExport
Teradata	Amazon Redshift	테라데이타 FastExport — 아마존 Redshift RSQL: 테라데이타 명령의 향상된 변환 — 레이아웃 FastExport
Oracle	PostgreSQL Aurora PostgreSQL	재변환 후 SAVE EXCEPTIONS STATEMENT가 있는 객체의 대상 스크립트가 변경되는 문제를 해결했습니다.
Oracle	PostgreSQL Aurora PostgreSQL	`proc_cursor_with_calc_columns` 변환 후 ORDER BY 절에 잘못된 필드가 지정되는 문제를 해결했습니다.
Oracle	PostgreSQL Aurora PostgreSQL	해결됨: ASSOCIATIVE COLLECTION 변환에는 추가 `aws_oracle_ext$array_id$temporary` 변수 선언이 필요합니다.

소스	대상	새로운 기능, 향상된 기능 또는 수정된 기능
Oracle	PostgreSQL Aurora PostgreSQL	해결됨: 동일한 테이블에서 소유한 INDEX와 동일한 이름으로 PRIMARY KEY를 사용하는 경우 잘못된 변환이 발생했습니다.

해결된 문제:

- 일반적인 개선 사항

빌드 AWS SCT 654의 릴리스 노트

소스	대상	새로운 기능, 향상된 기능 또는 수정된 기능
Oracle	PostgreSQL Aurora PostgreSQL	계층적 쿼리 의사 열, PRIOR 열 구문 분석 오류 문제를 해결했습니다.
Oracle	PostgreSQL Aurora PostgreSQL	슬래시와 별표(/*)가 포함된 다중 행 주석을 올바르게 변환하도록 문제를 해결했습니다.
Oracle	PostgreSQL Aurora PostgreSQL	확장 팩에 시스템 보기 USER_COL_COMMENTS 에뮬레이션을 추가했습니다.

소스	대상	새로운 기능, 향상된 기능 또는 수정된 기능
Oracle	PostgreSQL Aurora PostgreSQL	인용 리터럴의 변환을 개선했습니다.
DB2 LUW	PostgreSQL Aurora PostgreSQL	테이블, 보기, 별칭 또는 열의 설명에 레이블을 추가하거나 바꾸는 LABEL 문의 변환을 개선했습니다.
Oracle	None	SYS.USER$ 시스템 테이블을 DBA_USERS 보기로 대체하고 쿼리를 개선했습니다.
Oracle DW	Amazon Redshift	Oracle DW 메타데이터 쿼리를 업데이트했습니다.
Teradata	Amazon Redshift RSQL	셸, 테라데이타 및 테라데이타 기본 테라데이타 FastLoad 쿼리 (BTEQ) 스크립트를 Amazon Redshift RSQL 스크립트로 변환하기 위한 지원이 추가되었습니다.
Teradata BTEQ	Amazon Redshift RSQL	"merge_01"이 잘못 변환되는 문제를 해결했습니다.
Teradata BTEQ	Amazon Redshift RSQL	새로운 행의 스크립트 끝에 EOI(End or Identify)가 표시되도록 문제를 해결했습니다.
Azure Synapse	Amazon Redshift	Azure Synapse에 잘못된 암호를 입력했을 때 표시되는 오류 메시지를 개선했습니다.
Teradata	Amazon Redshift	Teradata 표준에 따라 올바른 별칭 이름을 전달하도록 UPDATE 문 변환을 개선했습니다.

소스	대상	새로운 기능, 향상된 기능 또는 수정된 기능
Teradata	Amazon Redshift	작업이 수신되지 않던 커서 변환 오류를 해결했습니다.
Teradata	Amazon Redshift	TD_NORMALIZE_OVERLAP 변환 시 행이 삭제되는 문제를 해결했습니다.
Teradata	Amazon Redshift	이제 향상된 TO_DATE 함수에 대한 엄격한 날짜 확인이 지원됩니다.
Teradata	Amazon Redshift	내장 함수 TO_NUMBER(n)의 변환을 개선했습니다.
Teradata	Amazon Redshift	메타데이터 트리에 스키마 범주가 표시되지 않는 문제를 해결했습니다.
Greenplum	Amazon Redshift	Greenplum 테이블의 가상 파티션을 생성할 때 목록에 GP_SEGMENT_ID 선택 항목을 추가했습니다.
Greenplum	Amazon Redshift	함수가 대상에 적용되지 않는 문제를 해결했습니다.
MS SQL Server DW	Amazon Redshift	AI 9996 없이 변환한 후 변환 오류가 발생하는 문제를 해결했습니다.
MS SQL Server DW	Amazon Redshift	확장 팩 마법사를 열면 오류가 기록되는 문제를 해결했습니다.
MS SQL Server DW	Amazon Redshift	Redshift Python 함수에 잘못된 주석 스타일이 사용되었을 때 발생하는 문제를 해결했습니다.
Netezza	Amazon Redshift	프로필이 있는 Netezza—Redshift 확장 팩을 만들지 못하는 문제가 해결되었습니다. AWS
Teradata	Amazon Redshift RSQL	SESSIONS 명령의 변환이 개선되었습니다. FastLoad

소스	대상	새로운 기능, 향상된 기능 또는 수정된 기능
Teradata	Amazon Redshift RSQL	FastLoad 스크립트 평가 보고서가 개선되었습니다.
Teradata	Amazon Redshift RSQL	FastLoad WRITER S3에 저장 작업을 구현했습니다.
Teradata	Amazon Redshift RSQL	스크립트 FastLoad 저장\ s3에 매니페스트 저장 버튼이 활성화되지 않던 문제가 해결되었습니다.
Teradata	Amazon Redshift RSQL	FastLoad multifile_script가 변환 후 예상 파일 세 개가 아닌 매니페스트 파일을 하나만 생성하는 문제가 해결되었습니다.
Teradata	Amazon Redshift RSQL	S3 경로에 추가 폴더가 표시되는 문제가 FastLoad 해결되었습니다.
Teradata	Amazon Redshift RSQL	S3 경로에 매니페스트 파일 이름이 잘못된 문제가 FastLoad 해결되었습니다.

해결된 문제:

- 일반적인 개선 사항

AWS SCT 빌드 653의 릴리스 노트

소스	대상	새로운 기능, 향상된 기능 또는 수정된 기능
Oracle	PostgreSQL	호출된 함수 또는 프로시저에서 생성된 동적 SQL을 변환하는 기능을 구현했습니다.

소스	대상	새로운 기능, 향상된 기능 또는 수정된 기능
	Aurora PostgreSQL	
Oracle	PostgreSQL Aurora PostgreSQL	동적 SQL 변환 개선: In-parameters를 바인드 변수로 사용.
Oracle DW 18, 19	Amazon Redshift	Oracle에서 Redshift로의 변환 개선 사항 구현: 향상된 변환 내장 기능. Aggregate LISTAGG, Analytic LISTAGG.
Oracle DW 18,19	Amazon Redshift	Oracle에서 Redshift로의 변환 개선 사항 구현: 새 기능 쿼리.
Vertica	Amazon Redshift	Vertica에서 Redshift로의 변환 개선 사항 구현: SSL=true일 때 SSL에서 JDBC로 연결.
MS SQL Server DW	Amazon Redshift	MS SQL Server에서 Redshift로의 변환 개선: 외부 테이블.
Teradata	Amazon Redshift	Teradata에서 Redshift로의 변환 개선: INTERVAL 데이터 유형 산술 연산.
Teradata	Amazon Redshift	Teradata에서 Redshift로의 변환 개선: 측면 열 별칭 지원.
Oracle	None	이제 다음 로더 쿼리에서 `SYS.USER$` 대신 `DBA_USERS` 를 사용합니다. • get-tree-path-list-by-name-path.sql • estimate-table-or-view-constraints-by-schema.sql • estimate-table-or-view-constraints-by-selected-schemas.sql

소스	대상	새로운 기능, 향상된 기능 또는 수정된 기능
Teradata	Amazon Redshift	SCT가 Teradata 매크로를 Redshift 저장 프로시저로 변환할 때 주석 정렬을 개선했습니다.
Oracle DW	Amazon Redshift	날짜/타임스탬프 형식 요소의 개선된 변환: `TO_DATE`, `TO_TIMESTAMP` 및 `TO_TIMESTAMP_TZ`
Teradata	Amazon Redshift	Teradata 커서 변환 오류를 해결했습니다.
Teradata	Amazon Redshift	변환 중에 `TD_NORMALIZE_OVERLAP` 의 속성이 삭제되는 문제를 해결했습니다.
Teradata	Amazon Redshift	SCT에서 쿼리를 변환할 때 `MAX` 함수가 무시되는 문제를 해결했습니다.
Teradata	Amazon Redshift	이제 SCT가 Teradata CHARACTERS 함수를 Redshift LENGTH 함수로 변환합니다.
Teradata	Amazon Redshift	이제 SCT가 가장 일반적으로 사용되는 형식에 대해 FORMAT을 TO_CHAR로 변환하도록 지원합니다.
모두	모두	암호화된 루틴의 변환을 개선했습니다.

해결된 문제:

- 일반적인 개선 사항

AWS SCT 빌드 652의 릴리스 노트

소스	대상	새로운 기능, 향상된 기능 또는 수정된 기능
Microsoft SQL Server	PostgreSQL	`sp_getapplock` 및 `sp_releaseapplock` 함수에 대한 앱 잠금 기능을 추가했습니다.

소스	대상	새로운 기능, 향상된 기능 또는 수정된 기능
None	Amazon Redshift	명령줄 인터페이스(CLI) 개선: 스크립트 명령 모드를 구현했습니다.
Oracle	PostgreSQL Aurora PostgreSQL	동적 SQL 내에서 루틴 파라미터 샘플링을 구현했습니다.
Oracle	PostgreSQL Aurora PostgreSQL	호출된 함수 또는 프로시저에서 생성된 동적 SQL에 대한 변환을 개선했습니다.
Microsoft SQL Server Oracle DB2 LUW	Aurora PostgreSQL	각 Lambda 함수는 정책을 통해 한 번만 배포 및 구성되며 일반적인 Lambda 함수는 가능한 모든 소스에서 재사용됩니다.
DB2 LUW	PostgreSQL	DB2 LUW를 소스로 사용하면 "9996 — Severity critical — Transformer error occurred"라는 오류 메시지가 발생하는 문제를 해결했습니다.
Teradata	Amazon Redshift	향후 Amazon Redshift 출시에서는 재귀 테이블 표현식이 지원됩니다.
Azure Synapse	Amazon Redshift	스키마 최적화 규칙을 구현했습니다.
Teradata	Amazon Redshift	Teradata 매크로에서 Redshift 저장 프로시저로의 시간대 변환을 지원합니다.

소스	대상	새로운 기능, 향상된 기능 또는 수정된 기능
Teradata	Amazon Redshift	PERIOD 값에 대한 산술 연산을 지원합니다.
Teradata	Amazon Redshift	Teradata 재귀적 공통 테이블 표현식(RECURSIVE CTE)의 변환을 지원합니다.
Teradata	Amazon Redshift	사용자 설정 `enable_case_sensitive_identifier` 를 통해 대소문자를 구분하는 식별자를 지원합니다. 따라서 “COLUMN_NAME”과 “Column_Name”은 서로 다른 열 이름이 됩니다.
Teradata	Amazon Redshift	십진수 필드가 동일한 정밀도로 변환되도록 십진수 데이터 유형 문제를 해결했습니다.
Teradata	Amazon Redshift	구간 산술 뺄셈이 올바르게 변환되도록 구간 산술 변환 문제를 해결했습니다.
Teradata	Amazon Redshift	Teradata NUMBER에서 DATE 유형으로의 캐스팅을 개선했습니다.
Teradata	Amazon Redshift	Teradata DATE에서 NUMBER 유형으로의 캐스팅을 개선했습니다.
Teradata BTEQ	Amazon Redshift	PERIOD 데이터 유형 변환을 개선했습니다.
Teradata	Amazon Redshift	이제 Teradata에서 올바르게 로드되도록 GEOMETRY 열이 있는 테이블의 메타데이터를 로드할 때 발생하는 문제를 해결했습니다.
Teradata	Amazon Redshift	Teradata 매크로를 Redshift 저장 프로시저로 변환할 때 merge 문 변환을 지원합니다.
Teradata	Amazon Redshift	Teradata에서 Redshift로 마이그레이션할 때 단순 매크로의 변환을 개선했습니다.
Teradata	Amazon Redshift	Teradata UPDATE 문의 변환이 Teradata 표준에 따라 올바른 별칭 이름을 전달하도록 개선되었습니다.

해결된 문제:

- 일반적인 개선 사항

AWS SCT 빌드 651의 릴리스 노트

소스	대상	새로운 기능, 향상된 기능 또는 수정된 기능
모두	모두	나열된 권장 전환 액션 항목에 대한 링크를 업데이트하도록 AWS SCT 보고서를 개선했습니다.
MS SQL Server	PostgreSQL	`STR()` 함수 변환에 대한 지원을 추가했습니다.
MS SQL Server	PostgreSQL	비트 EXOR 연산자(Microsoft SQL Server의 ^)를 PostgreSQL # 연산자로 변환하는 지원을 추가했습니다.
Oracle	PostgreSQL	PostgreSQL 타겟에서 AWS SCT 확장 팩 `aws_oracle_ext.UNISTR(null)` 함수가 중단되는 문제가 해결되었습니다. `NULL` AWS SCT 이제 이 작업을 처리합니다. `NULL`
Teradata BTEQ	Amazon Redshift RSQL	Amazon Redshift RSQL MERGE를 변환할 때 변환 오류가 발생하는 문제를 해결하기 위해 변환을 개선했습니다.
Oracle DW	Amazon Redshift	개선된 내장 기능을 구현했습니다.
Oracle DW	Amazon Redshift	자동 목록 파티셔닝(TBL_PART_LIST_AUTO), 복수 열 목록(TBL_PART_MULTI_LIST) 및 구간 참조(TBL_PART_RANGE_INTVAL_REF)를 비롯한 메타데이터 기능 기반 개선 사항을 추가했습니다.
없음	Amazon Redshift	`UNION ALL` 변환에 사용되는 물리적 파티션의 파티션 테이블 제한을 높였습니다.
Teradata	Amazon Redshift	평가 보고서의 범위에 대한 변환을 개선했습니다.

소스	대상	새로운 기능, 향상된 기능 또는 수정된 기능
Teradata	Amazon Redshift	복잡한 Teradata MACRO 변환을 개선했습니다.
Teradata	Amazon Redshift	지원되지 않는 SQL을 주석 처리할 때 Teradata 매크로를 Amazon Redshift 저장 프로시저로 변환하는 기능을 개선했습니다.
Teradata	Amazon Redshift	Teradata 매크로를 Amazon Redshift 저장 프로시저로 변환할 때 잘못된 별칭 이름 참조가 발생하는 문제를 해결했습니다.
Teradata	Amazon Redshift	Teradata `QUALIFY` 문의 변환을 개선했습니다.
Teradata	Amazon Redshift	Amazon Redshift에 주석을 전달하고 보기에서 수행된 변경 기록을 유지할 수 있도록 변환을 개선했습니다.
Teradata	Amazon Redshift	RESET WHEN 절로 인해 올바른 변환이 수행되지 않았던 문제를 해결했습니다.
Teradata BTEQ	Amazon Redshift	MERGE 문을 포함하는 BTEQ 스크립트의 변환을 개선했습니다.
Teradata	Amazon Redshift	PERIOD 데이터 유형 필드의 변환을 개선하기 위해 내장 함수를 추가했습니다.
Microsoft SQL Server	Amazon Redshift	TIME 데이터 유형에 대한 변환 데이터 유형 매핑을 개선했습니다.
모두	모두	AWS Schema Conversion Tool CLI 참조 설명서의 PDF 형식 초기 출판물에 대한 액세스를 추가했습니다. AWS Schema Conversion Tool CLI 참조를 참고하세요.

해결된 문제:

- 일반적인 개선 사항

AWS SCT 빌드 650의 릴리스 노트

소스	대상	새로운 기능, 향상된 기능 또는 수정된 기능
모두	모두	다음을 포함한 추출기 에이전트 사용을 업데이트 및 개선했습니다. • 공유 스토리지 및 전용 복사 에이전트와 함께 사용하기 위한 구성 • 한 프로젝트에서 다른 프로젝트로 데이터 추출 작업 내보내기 및 가져오기 • Azure SQL Data Warehouse(Azure Synapse)를 소스로 지원 • 네이티브 Netezza 파티셔닝 사용 자세한 정보는 온프레미스 데이터 웨어하우스에서 Amazon Redshift로 데이터 마이그레이션을 참조하세요.
모두	Amazon RDS PostgreSQL 13	AWS SCT 이제 Amazon RDS PostgreSQL 13을 타겟으로 지원합니다.
Microsoft SQL Server	Aurora PostgreSQL	Microsoft SQL Server 프로시저에서 Aurora PostgreSQL 대상으로 결과 세트를 변환하는 기능을 개선했습니다.
Oracle DW	Amazon Redshift	Oracle에서 Amazon Redshift로의 변환 개선 사항을 구현했습니다.
Oracle DW	Amazon Redshift	동적 SQL 문 변환에 대한 개선 사항을 구현했습니다.
Oracle DW	Amazon Redshift	SQL UDF 변환에 대한 개선 사항을 구현했습니다.

소스	대상	새로운 기능, 향상된 기능 또는 수정된 기능
Oracle DW	Amazon Redshift	외부 테이블 변환을 AWS SCT 지원하지 않는 메시지를 명확히 설명했습니다.
Oracle DW	Amazon Redshift	내장 변환 함수를 개선했습니다.
Teradata BTEQ	Amazon Redshift RSQL	GUI를 사용하는 동안 BTEQ 스크립트 내에서 대체 매개 변수를 처리하는 기능이 개선되었습니다. AWS SCT
Microsoft SQL Server DW Microsoft SQL Server Azure Azure Synapse	모두	Microsoft SQL Server, Azure, Azure Synapse에 대해 지원되는 최소 JDBC 드라이버 버전을 업그레이드했습니다.

해결된 문제:

- Teradata: 매크로 변환 추가 개선 [해결됨]
- 대상에서 특수 문자가 이스케이프되어 SQL 오류가 발생하고 이를 다시 배치하려면 재작업이 필요 [해결됨]
- 일반적인 개선 사항

빌드 649의 릴리스 노트 AWS SCT

소스	대상	새로운 기능, 향상된 기능 또는 수정된 기능
Microsoft SQL Server DW	Amazon Redshift	임시 테이블을 지원하도록 MSSQL에서 Amazon Redshift로의 변환을 개선했습니다.
Oracle DW	Amazon Redshift	다음과 같이 개선된 내장 함수를 구현했습니다. 변환 함수 • TO_BINARY_DOUBLE • TO_BINARY_FLOAT • TO_NUMBER • TO_DATE • TO_TIMESTAMP • TO_TIMESTAMP_TZ • TO_DSINTERVAL • TO_YMINTERVAL • VALIDATE_CONVERSION
Oracle DW	Amazon Redshift	다음과 같이 근사 쿼리 처리를 위해 개선된 함수를 구현했습니다. 집계 함수 • ANY_VALUE • APPROX_COUNT_DISTINCT •

소스	대상	새로운 기능, 향상된 기능 또는 수정된 기능
		APPROX_COUNT_DISTINCT_DETAIL • APPROX_COUNT_DISTINCT_AGG • LISTAGG • TO_APPROX_COUNT_DISTINCT
Teradata	Amazon Redshift	Teradata 자동 정렬 및 배포 키 선택에 대한 변환 개선 사항을 구현했습니다. DB 엔진은 배포 키와 정렬 키를 자동으로 선택합니다. Current projects settings > Optimization strategies > Initial Key Selection Strategy 대화 상자에 Use Amazon Redshift automatic table tuning 라디오 버튼을 도입했습니다.
Teradata	Amazon Redshift	AWS SCT Teradata에서 모든 테이블을 AWS SCT 로드할 수 있도록 테이블 로더를 개선했습니다.
Teradata	Amazon Redshift	Amazon Redshift가 단순한 WHERE NOT EXISTS 절을 포함하는 상호 연관된 하위 쿼리 패턴을 지원하도록 변환 개선 사항을 구현했습니다.
Teradata	Amazon Redshift	매크로에서 ECHO 명령 사용에 대한 지원을 추가했습니다.
DB2 LUW	PostgreSQL Aurora PostgreSQL	다음을 포함하여 DYNAMIC RESULTS SETS 변환에 대한 지원을 구현했습니다. • 커서 절 WITH RETURN/WITH RETURN TO CLIENT • DYNAMIC RESULT SETS 루틴 절 변환

소스	대상	새로운 기능, 향상된 기능 또는 수정된 기능
Microsoft SQL Server Oracle DB2 LUW SAP ASE	Aurora PostgreSQL	현재 Aurora RDS PostgreSQL을 대상으로 사용하기 위한 지원 기능을 구현했습니다.
Microsoft SQL Server Oracle DB2 LUW SAP ASE	MariaDB	MariaDB 10.5를 대상으로 사용하기 위한 지원 기능을 구현했습니다.
Microsoft SQL Server	MariaDB	삽입된 행의 결과 세트를 반환하는 INSERT-RETURNING 지원을 구현했습니다.
Oracle	Aurora PostgreSQL	Oracle에서 Aurora PostgreSQL로 변환하기 위한 XMLFOREST 함수 지원을 추가했습니다.

해결된 문제:

- 일반적인 개선 사항

빌드 648의 AWS SCT 출시 노트

소스	대상	새로운 기능, 향상된 기능 또는 수정된 기능
Oracle	PostgreSQL Amazon Aurora PostgreSQL 호환 에디션	Aurora PostgreSQL 확장 팩 사용자 지정 적용 모드 구현: 숫자/날짜 및 텍스트 유형에 대한 연산자
Oracle Microsoft SQL Server DB2 LUW	Aurora PostgreSQL	Aurora PostgreSQL Lambda Invoke 구성 구현: aws_lambda 확장 프로그램 생성, Aurora PostgreSQL 클러스터에 IAM 역할 할당 • 오라클—이메일, 작업, 대기열, 파일 WebAgent • DB2—이메일, 작업, 파일 • Microsoft SQL Server—이메일, 에이전트
Oracle	PostgreSQL	FORALL 문 변환 리팩터링 구현: • FORALL 문 • FORALL ... SAVE EXCEPTIONS • RETURNING INTO with BULK COLLECT • SQL%BULK_EXCEPTIONS 시스템 컬렉션
Oracle DW 18, 19	Amazon Redshift	Oracle에서 Amazon Redshift로의 변환 개선 사항 구현: 향상된 변환 내장 기능. Aggregate LISTAGG, Analytic LISTAGG.

소스	대상	새로운 기능, 향상된 기능 또는 수정된 기능
Oracle DW 18,19	Amazon Redshift	Oracle에서 Amazon Redshift로의 변환 개선 사항 구현: 새 기능 쿼리.
Vertica	Amazon Redshift	Vertica에서 Amazon Redshift로의 변환 개선 사항 구현: SSL=true일 때 SSL에서 JDBC로 연결.
Microsoft SQL Server DW	Amazon Redshift	Microsoft SQL Server에서 Redshift로의 변환 개선: 외부 테이블.
Teradata	Amazon Redshift	Teradata에서 Redshift로의 변환 개선: INTERVAL 데이터 유형 산술 연산.
Teradata	Amazon Redshift	Teradata에서 Redshift로의 변환 개선: 측면 열 별칭 지원.

해결된 문제:

- 일반적인 개선 사항

빌드 AWS SCT 647의 릴리스 노트

소스	대상	새로운 기능, 향상된 기능 또는 수정된 기능
Microsoft SQL Server	Microsoft SQL Server	이제 RDS는 데이터베이스 메일 기능을 지원합니다.
Microsoft SQL Server	MySQL	각 식별자 유형의 최대 이름 구현 — SQL Server에서 객체 이름(예: 테이블, 제약 조건, 열)의 최대 길이는 128자입니다. MySQL에서 객체 이름의 최대 길이는 64자입니다. 변환된 객체를 MySQL 데이터베이스에 쓰려면 이름을 줄여야 합니다. 잘라낸 후 이름이 중복되는 것을 방지하려면 원본 객체 이름의 “체크섬”을 새 이름에 추가해야 합니다. 64자보다 긴 이름은 다음과 같이 잘라냅니다.

소스	대상	새로운 기능, 향상된 기능 또는 수정된 기능

		`[first N chars]() + "" + [checksum]()`
		
		`[first N chars] = 64 - 1 - [length of checksum string]`
		
		예:
		`example_of_a_test_schema_with_a_name_length_greater_than_64_characters ?? example_of_a_test_schema_with_a_name_length_greater_than_64_9703`

Oracle	MySQL/Aurora MySQL	스토리지 객체에 대한 주석 로드 및 변환을 구현했습니다. 예: 테이블에 대한 주석 처리 및 테이블/보기 열에 대한 주석 처리
Teradata	Amazon Redshift	TIME 데이터 유형 변환에 대한 지원을 추가했습니다.
Teradata	Amazon Redshift	변환 개선 사항 — TD_NORMALIZE_OVERLAP을 구현했습니다.
Microsoft SQL Server DW	Amazon Redshift	변환 개선 사항 — WITH 절이 있는 SELECT, FROM이 없는 SELECT
모두	모두	AWS SCT 데이터 마이그레이션 서비스 평가자 (DMSA) — 이 새로운 기능을 사용하면 여러 서버를 평가하고 환경에 가장 적합한 목표 방향을 보여주는 요약 보고서를 받을 수 있습니다.
모두	모두	AWS SCT 마법사 — 이제 대상 비교에서 대상 간의 차이가 단일 테이블 보기로 표시됩니다.
모두	모두	트리 필터 UI — 재설계된 메타데이터 필터가 더 복잡한 필터링 패턴을 처리합니다.
모두	모두	평가 보고서 — 재설계된 경고 섹션이 문제에 대한 더 나은 설명과 더 명확한 이해를 제공합니다.

해결된 문제:

- 일반적인 개선 사항
- 데이터 추출기 — ConcurrentModificationException [RESOLVED] 오류가 발생하여 하위 작업이 실패했습니다.
- Microsoft SQL Server에서 MySQL로 — 최대 식별자 길이 [해결됨]

빌드 646의 AWS SCT 릴리스 노트

소스	대상	새로운 기능, 향상된 기능 또는 수정된 기능
Oracle	PostgreSQL	TM 형식 모델 구현을 개선했습니다.
Oracle	PostgreSQL	SP 형식 마스크 구현에서는 영어인 경우에만 SP 접미사에 대한 기본 지원을 제공합니다.
Oracle	PostgreSQL	Oracle 긴 객체 이름 처리 — AWS SCT 이제 대상 최대 식별자 길이 속성에 따라 Oracle 긴 객체 이름을 처리합니다.
	Amazon Redshift	Amazon Redshift 인코딩 AZ64 포함 AWS SCT — 일부 데이터 유형에 대한 압축 인코딩 AZ64 추가
Teradata	Amazon Redshift	암시적 트랜잭션 변환에 대한 지원을 추가했습니다.
Teradata	Amazon Redshift	Teradata 지리 공간 기본 제공 함수에 대한 지원 추가: `ST_LineString` 메서드
Greenplum	Amazon Redshift	Greenplum 시퀀스 변환 — 속성 탭에 최솟값, 최댓값, 증가, 주기 항목을 추가했습니다.
Greenplum	Amazon Redshift	해석기 — “char” 데이터 유형 처리 기능을 추가했습니다.
Greenplum	Amazon Redshift	문자 변환 길이 - 문자 유형에 대한 PL/pgSQL 변환을 업데이트했습니다.

소스	대상	새로운 기능, 향상된 기능 또는 수정된 기능
Greenplum	Amazon Redshift	Greenplum 배포 키 선택 시 테이블에 배포 키가 있지만 테이블을 무작위로 배포된 것으로 인식하고 가져올 AWS SCT 수 없는 문제가 해결되었습니다.
Teradata	Amazon Redshift	Teradata 커서 지원 — 커서 변환에 대한 지원을 추가했습니다.
Teradata	Amazon Redshift	ID 열 — ID 열 변환에 대한 지원을 추가했습니다.
Teradata	Amazon Redshift	INTERVAL 데이터 유형 — INTERVAL 데이터 유형 변환에 대한 지원을 추가했습니다.

해결된 문제:

- 일반적인 개선 사항
- Greenplum: 로그의 오류로 인해 변환을 실행할 수 없습니다. [해결됨]
- MSSQL — PostgreSQL: LAG 함수를 변환할 때 변환기 오류가 발생했습니다. [해결됨]
- MSSQL — PostgreSQL: SCOPE_IDENTITY [해결됨]
- AWS SCT DW 프로젝트에서 멈춤 [해결됨].
- AWS SCT [RESOLVED] 에서 열 이름의 추가 공백을 제거하려면 매핑 규칙이 필요합니다.

AWS SCT 빌드 645의 릴리스 노트

소스	대상	새로운 기능, 향상된 기능 또는 수정된 기능
Teradata	Amazon Redshift	Teradata의 정규화되지 않은 보기(보기 이름 또는 보기 내의 정규화되지 않은 객체)를 해결하기 위한 솔루션을 제공합니다.
Teradata	Amazon Redshift	노드를 계산하기 위한 ASCII 함수 지원을 추가했습니다.

소스	대상	새로운 기능, 향상된 기능 또는 수정된 기능
Teradata	Amazon Redshift	로 CHAR 정의된 테라데이타에서 멀티바이트 데이터를 발견하면 AWS SCT Amazon CHAR(N) VARCHAR(3*N) Redshift에서 해당 데이터로 변환됩니다.
Teradata	Amazon Redshift	날짜와 숫자 사이의 Teradata CAST 변환을 제공합니다. • SELECT Cast('2020-07-17' AS BIGINT) • SELECT Cast(20200630 - 19000000 AS DATE)
Teradata	Amazon Redshift	Teradata PERIOD 데이터 유형을 두 개의 Amazon Redshift TIMESTAMP 열로 변환하도록 지원합니다. • PERIOD(TIMESTAMP) • PERIOD(TIMESTAMP WITH TIMEZONE)
Teradata	Amazon Redshift	RESET WHEN 절을 사용하여 Teradata RANK 함수의 변환을 지원합니다.
Teradata	Amazon Redshift	명시적 데이터 유형 변환에서 CAST 지원 및 표현식에 대한 암시적 CAST 지원을 개선했습니다.
Teradata	Amazon Redshift	지원되지 않는 상관 관계가 있는 하위 쿼리 패턴을 보고합니다. 자세한 내용은 Amazon Redshift 데이터베이스 개발자 안내서에서 상관관계가 있는 하위 쿼리를 참조하세요.
none	Amazon Redshift	RA3 노드 유형에 대한 테이블 제한 지원을 개선했습니다.
Teradata	Amazon Redshift	Teradata 지리 공간 데이터 추출에 대한 지원을 개선했습니다. 자세한 내용은 Amazon Redshift 데이터베이스 개발자 안내서에서 Amazon Redshift에서 공간 데이터 쿼리를 참조하세요.

소스	대상	새로운 기능, 향상된 기능 또는 수정된 기능
Microsoft SQL Server	PostgreSQL	convert_procedures_to_function 옵션을 추가했습니다.

해결된 문제:

- 일반적인 개선 사항

빌드 644의 릴리스 노트 AWS SCT

릴리스 1.0.643의 변경 사항은 1.0.644 AWS SCT 릴리스에 병합되었습니다. AWS SCT

소스	대상	새로운 기능, 향상된 기능 또는 수정된 기능
Teradata	Amazon Redshift	여러 변환 개선 사항 • 테이블 별칭이 포함된 QUALIFY를 사용한 변환을 개선했습니다. • IN 연산자를 사용한 변환을 개선했습니다. • LIKE 연산자를 사용한 변환을 개선했습니다. • 변환된 코드에서 강조 표시 문제가 있는 변환을 개선했습니다. • SQL에서 WHERE, QUALIFY 절의 순서가 비정상적인 변환을 개선했습니다. • 프로시저 UPD_FT_SVC_TRANS_BH_CBH_IND 의 JOIN( ) 구성 변환 중에 발생한 변환기 오류를 수정했습니다. • 매크로를 저장 프로시저로 변환하는 기능을 개선했습니다.

소스	대상	새로운 기능, 향상된 기능 또는 수정된 기능
		제공된 sql/bteq 스크립트를 파싱하고 소스 코드에서 발견된 구문 구조 수에 대한 보고서를 생성할 수 있는 특수 AWS SCT CLI 명령이 추가되었습니다. • BTEQ 명령 개수 • HANDLERS 개수 • CAST 케이스 개수 • DML/DDL 사례 개수 • 업데이트 가능한 보기에서 DML 개수 평가 보고서 작업 항목 추가: Amazon Redshift에서는 사용자 지정 날짜 형식을 사용하는 Teradata 열이 지원되지 않습니다.
Oracle	PostgreSQL/Aurora PostgreSQL	확장 팩 설치 스크립트를 저장하는 기능을 추가했습니다. AI 5334의 심각도 수준을 변경했습니다. 레코드를 패키지 변수 `IMPLEMENTATION` 으로 사용하는 성능을 개선했습니다. `XMLAGG` 집계 함수 지원을 추가했습니다.
IBM Db2	PostgreSQL/Aurora PostgreSQL	스토리지 객체 구현에 대한 주석 로드 및 변환 기능을 추가했습니다.

소스	대상	새로운 기능, 향상된 기능 또는 수정된 기능
MS SQL DW	Amazon Redshift	변환 개선: PATINDEX 문제를 해결했습니다. UI 개선 사항: • 소스 트리 구현을 위한 SQL로 저장 옵션. • 여러 파일에 대한 스크립트 생성을 위한 로직을 추가했습니다.
Vertica	Amazon Redshift	UI 개선: 소스 트리 구현을 위한 SQL로 저장 옵션

해결된 문제:

- Teradata와 Amazon Redshift 간의 변환에 대한 전반적인 개선 사항
- 일반적인 버그 수정 및 UI 개선

빌드 642의 릴리스 노트 AWS SCT

AWS Schema Conversion Tool 릴리스 1.0.642에 대한 변경 사항.

> **Note**
>
> AWS Schema Conversion Tool (AWS SCT) 빌드 1.0.642 변경 사항은 윈도우, 우분투, 페도라에 적용됩니다. macOS용 1.0.642 빌드는 없습니다.

소스	대상	새로운 기능, 향상된 기능 또는 수정된 기능
Microsoft SSIS	AWS Glue	Microsoft SQL 서버 통합 서비스 (SSIS) ETL 패키지를 로 변환하도록 구현했습니다. AWS Glue자세한 정보는 AWS SCT를 사용하여 SSIS를 AWS Glue로 변환을 참조하세요.
Oracle	MariaDB/SQL MODE=C	WITH 절에서 PL/SQL 선언 섹션을 구현했습니다.

소스	대상	새로운 기능, 향상된 기능 또는 수정된 기능
	LE/MySQL/ Amazon Aurora MySQL	
Oracle	PostgreSQL/Aurora PostgreSQL	`DBMS_SESSION.RESET_PACKAGE` 및 `DBMS_SESSION.MODIFY_PACKAGE` 에 대한 지원이 추가되었습니다.
Vertica	Amazon Redshift	Vertica 데이터베이스에서 Amazon Redshift로 SQL 스크립트를 내보낼 수 있습니다.

해결된 문제:

- 평가 보고서 개선 사항
- 평가 보고서 UI 개선 사항
- UI에서 JVM 설정을 변경하는 기능 추가
- 일반적인 개선 사항

빌드 641의 AWS SCT 릴리스 노트

AWS Schema Conversion Tool 릴리스 1.0.641에 대한 변경 사항.

> **Note**
>
> AWS Schema Conversion Tool (AWS SCT) 빌드 1.0.641 변경 사항은 윈도우, 우분투, 페도라에 적용됩니다. macOS용 1.0.641 빌드는 없습니다.

소스	대상	새로운 기능, 향상된 기능 또는 수정된 기능
Oracle/ MS SQL/	모두	.csv 파일에서 시간 보고서 계산을 생성합니다.

소스	대상	새로운 기능, 향상된 기능 또는 수정된 기능
MySQL/PostgreSQL/Db2 LUW		
Teradata	Amazon Redshift	CSUM 함수에 대한 지원을 추가했습니다. Teradata 지리 공간 데이터 유형에 대한 지원을 추가했습니다.
Teradata	모두	IDENTITY 열 변환에 대한 지원을 추가했습니다.
Greenplum	Amazon Redshift	Greenplum 테이블 변환 중 배포 스타일 AUTO에 대한 지원을 추가했습니다.
SAP ASE	모두	.csv 파일에서 시간 보고서 계산을 생성합니다.

해결됨:

- 다양한 버그가 수정되었습니다.
- 다양한 성능 개선 사항

AWS SCT 빌드 640의 릴리스 노트

AWS SCT 릴리스 1.0.633, 1.0.634, 1.0.635, 1.0.636, 1.0.637, 1.0.638, 1.0.639 및 1.0.640의 변경 사항이 1.0.640 릴리스에 병합되었습니다. AWS SCT

> **Note**
>
> AWS SCT 빌드 1.0.640 변경 사항은 윈도우, 우분투, 페도라에 적용됩니다. macOS에는 적용되지 않습니다.
> 애플 macOS에는 AWS SCT 버전 1.0.640 이상을 설치할 수 없습니다. AWS SCT 버전 1.0.632는 애플 macOS에서의 설치를 지원하는 마지막 버전입니다.

아래 표에는 릴리스 1.0.640으로 결합된 AWS Schema Conversion Tool 버전의 기능과 버그 수정 목록이 나와 있습니다. 이러한 표에서는 소스 엔진별로 기능과 버그 수정이 그룹화되어 있습니다.

주제

- 릴리스 1.0.640 Oracle 변경 사항
- 릴리스 1.0.640 Microsoft SQL Server 변경 사항
- 릴리스 1.0.640 MySQL 변경 사항
- 릴리스 1.0.640 PostgreSQL 변경 사항
- 릴리스 1.0.640 Db2 LUW 변경 사항
- 릴리스 1.0.640 Teradata 변경 사항
- 기타 엔진과 관련된 릴리스 1.0.640 변경 사항

릴리스 1.0.640 Oracle 변경 사항

다음 표에는 Oracle을 소스 엔진으로 사용하는 경우 해당되는 1.0.640 빌드 변경 사항이 나와 있습니다.

소스	대상	새로운 기능, 향상된 기능 또는 수정된 기능
Oracle	PostgreSQL Aurora PostgreSQL	Java 및 Pro*C 애플리케이션에서 SQL 코드 변환을 구현했습니다.
Oracle	PostgreSQL Aurora PostgreSQL	WHERE 절에서 사용할 때 다음 함수의 성능이 향상되었습니다. • aws_oracle_ext.to_date • aws_oracle_ext.to_char • aws_oracle_ext.to_number • aws_oracle_ext.sysdate • aws_oracle_ext.sys_context
Oracle	RDS MariaDB 10.4	모든 OLTP(Online Transactional Processing) 공급업체에 대한 RDS MariaDB 10.4 지원을 추가했습니다.

소스	대상	새로운 기능, 향상된 기능 또는 수정된 기능
Oracle	PostgreSQL/Aurora PostgreSQL	DBMS_UTILITY.GET_TIME에 대한 지원을 추가했습니다. 다음 에뮬레이션을 추가했습니다. • DBMS_UTILITY.GET_TIME • DBMS_UTILITY.FORMAT_CALL_STACK • DBMS_UTILITY.CURRENT_INSTANCE
Oracle	MariaDB/MySQL/Aurora MySQL/Microsoft SQL Server Mode=Oracle/PostgreSQL/Aurora PostgreSQL/RDS Oracle	TABLE(DATA,EXTENDED DATA), VIEW(DATA,EXTENDED DATA) 및 SEQUENCE(DATA)에 대한 공유 절 지원을 추가했습니다.
Oracle	PostgreSQL/Aurora PostgreSQL/Oracle RDS	명시적 NULL 삽입에 DEFAULT를 적용하도록 열의 DEFAULT 정의를 확장할 수 있습니다. DEFAULT 절에 새로운 ON NULL 절이 있습니다. 이 새로운 절은 INSERT 문이 NULL로 평가되는 값을 할당하려고 할 때 지정된 기본 열 값을 할당하도록 데이터베이스에 지시합니다.
Oracle	MariaDB/MariaDB(SQL MODE=ORACLE)	삽입 시 자동으로 증분되는 “Identity 열”에 대한 지원을 추가했습니다.

소스	대상	새로운 기능, 향상된 기능 또는 수정된 기능
모두	모두	JDK 8에서 Amazon Corretto JDK 11로 업그레이드되었습니다. 다운로드 링크를 포함한 자세한 내용은 Amazon Corretto 11 사용 설명서에서 Amazon Corretto 11이란?을 참조하세요.
모두	모두	사용자 데이터베이스에서 발생할 수 있는 불일치에 대한 정보를 평가 보고서에 추가했습니다.
Oracle	MariaDB 10.2/MariaDB 10.3/MySQL/Aurora MySQL/PostgreSQL/Aurora PostgreSQL	`DEFAULT` 절에 새로운 `ON NULL` 절이 있습니다. 이 새로운 절은 INSERT 문이 `NULL`로 평가되는 값을 할당하려고 할 때 지정된 기본 열 값을 할당하도록 데이터베이스에 지시합니다.
Oracle	Oracle RDS/MySQL/Aurora MySQL/PostgreSQL/Aurora PostgreSQL	`IDENTITY` 열에 대한 지원을 추가했습니다.
Oracle	MySQL 8.x	CHECK 제약 조건에 대한 지원을 추가했습니다.

소스	대상	새로운 기능, 향상된 기능 또는 수정된 기능
Oracle	PostgreSQL/Aurora PostgreSQL	확장 팩 루틴을 사용한 ANYDATA IS NULL/IS NOT NULL 점검을 구현했습니다. XMLSequence의 TABLE 함수를 기반으로 쿼리에 사용되는 VALUE 함수의 에뮬레이션을 구현했습니다. 다음과 같은 기본 제공 루틴에 대한 DBMS_LOB 지원을 추가했습니다. • DBMS_LOB.CREATETEMPORARY • DBMS_LOB.FREETEMPORARY • DBMS_LOB.APPEND
모두	SQL Server	SQL Server 2019: 새로운 인덱스 속성 OPTIMIZE_FOR_SEQUENTIAL_KEY에 대한 지원을 추가했습니다. SQL Server 2017: 그래프 데이터베이스 노드 및 엣지 테이블 유형에 대한 지원을 추가했습니다. SQL Server 2016: 임시 테이블에 대한 지원을 추가했습니다.
모두	모두	가상 파티션으로 물리적 파티션을 재정의하는 기능을 구현했습니다. 데이터 웨어하우스 추출기가 생성된 가상 파티션에 따라 데이터를 추출합니다.
Oracle	Amazon Redshift	중첩 블록에서 커서 속성의 변환을 구현했습니다. Amazon Redshift는 컬렉션을 지원하지 않습니다. 관련 변수는 VARCHAR로 변환됩니다. 하나의 변수를 다른 변수에 할당하는 것 이외의 모든 컬렉션 작업(시작 및 컬렉션 요소 액세스 포함)은 거부됩니다. Amazon Redshift 배포 스타일 = AUTO를 구현했습니다.

소스	대상	새로운 기능, 향상된 기능 또는 수정된 기능
Oracle	PostgreSQL/Aurora PostgreSQL	Oracle의 예약되지 않은 단어가 PostgreSQL에서 예약되어 있으면 다음 사항이 적용됩니다. • 단어가 인용된 경우 대/소문자가 유지되고 따옴표로 묶여 있습니다. • 단어가 인용되지 않은 경우 대문자로 변환되고 따옴표로 묶입니다. 함수를 LTRIM, RTRIM 및 TRIM 함수에 대한 입력으로 사용하는 기능을 구현했습니다. SELECT DISTINCT, ORDER BY 표현식이 선택 목록에 나타나야 합니다. 기본값이 있는 매개 변수 뒤에 오는 커서 매개 변수의 경우 DEFAULT IS NULL 절을 AWS SCT 추가합니다. 소스 OUT 커서 파라미터는 IN 커서 파라미터로 변환됩니다. “변환 설정”에서 “패키지 변수 논리 구현” 옵션을 추가하여 패키지 변수를 다시 구현했습니다. 사용 가능한 설정은 “세션 변수” 및 “plv8 전역 객체”입니다. 기본값은 “세션 변수”입니다. dblink 및 pg_background를 사용하여 AUTONOMOUS_TRANSACTION pragma 지원을 구현했습니다.
Oracle	모두	SYS_%_TAB_COMMENTS 뷰를 구현했습니다.
Oracle	PostgreSQL	필터에 대한 변수 입력은 PostgreSQL에서 지원되지 않습니다. Oracle에서 PostgreSQL로 변환할 때 변수 필터가 있으면 예외가 보고됩니다.

소스	대상	새로운 기능, 향상된 기능 또는 수정된 기능
Oracle	Amazon Redshift	저장된 코드 FOR..LOOP 커서 변환 개선 사항을 구현했습니다. 기본 파라미터를 사용한 함수/프로시저의 저장된 코드 호출을 구현했습니다. WHERE 절 없이 별칭을 사용해 업데이트할 수 있는 저장된 코드 기능을 구현했습니다. 저장된 코드 함수가 SELECT FROM dual을 사용하여 추가 사례를 수행하도록 구현했습니다. 저장된 코드 Table%ROWTYPE 파라미터 및 패키지 변수를 구현했습니다. JAVA 및 외부 프로시저에 사용되는 저장된 코드를 구현했습니다. 저장된 코드에 표준 Oracle 패키지를 구현했습니다.

릴리스 1.0.640 Microsoft SQL Server 변경 사항

다음 표에는 Microsoft SQL Server를 소스 엔진으로 사용하는 경우 해당되는 1.0.640 빌드 변경 사항이 나와 있습니다.

소스	대상	새로운 기능, 향상된 기능 또는 수정된 기능
Microsoft Azure/ Microsoft SQL Server	PostgreSQL/Aurora PostgreSQL/MySQL/ Aurora MySQL	COLUMN STORE 인덱스에 대한 지원을 추가했습니다.
Microsoft SQL Server	RDS MariaDB 10.4	모든 OLTP(Online Transactional Processing) 공급업체에 대한 RDS MariaDB 10.4 지원을 추가했습니다.

소스	대상	새로운 기능, 향상된 기능 또는 수정된 기능
Azure/SQL Server	MariaDB/MySQL/Aurora MySQL/PostgreSQL/Aurora PostgreSQL	OPTIMIZE_FOR_SEQUENTIAL_KEY 인덱스 속성에 대한 지원을 추가했습니다.
Azure/SQL Server	MySQL/Aurora MySQL/PostgreSQL/Aurora PostgreSQL	데이터베이스 노드 및 엣지 테이블 유형에 대한 지원을 추가했습니다.
Azure/SQL Server	MariaDB/MySQL/Aurora MySQL/PostgreSQL/Aurora PostgreSQL	임시 테이블에 대한 지원을 추가했습니다.
모두	모두	JDK 8에서 Amazon Corretto JDK 11로 업그레이드되었습니다. 다운로드 링크를 포함한 자세한 내용은 Amazon Corretto 11 사용 설명서에서 Amazon Corretto 11이란?을 참조하세요.
모두	모두	사용자 데이터베이스에서 발생할 수 있는 불일치에 대한 정보를 평가 보고서에 추가했습니다.

소스	대상	새로운 기능, 향상된 기능 또는 수정된 기능
Azure/SQL Server	MySQL/Aurora MySQL/PostgreSQL/Aurora PostgreSQL/MariaDB	SQL Server 그래프 아키텍처에 대한 DML 처리 지원을 추가했습니다.
SQL Server	Aurora PostgreSQL	`par_` 접두사 없이 파라미터를 변환하는 옵션을 추가했습니다.
Azure/SQL Server	MySQL 8.x	CHECK 제약 조건에 대한 지원을 추가했습니다.
모두	SQL Server	SQL Server 2019: 새로운 인덱스 속성 OPTIMIZE_FOR_SEQUENTIAL_KEY에 대한 지원을 추가했습니다. SQL Server 2017: 그래프 데이터베이스 노드 및 엣지 테이블 유형에 대한 지원을 추가했습니다. SQL Server 2016: 임시 테이블에 대한 지원을 추가했습니다.
모두	모두	가상 파티션으로 물리적 파티션을 재정의하는 기능을 구현했습니다. 데이터 웨어하우스 추출기가 생성된 가상 파티션에 따라 데이터를 추출합니다.

소스	대상	새로운 기능, 향상된 기능 또는 수정된 기능
SQL Server	AWS Glue (파이썬 셸)	다음을 포함한 변환 개선 사항: • 기본 제공 함수를 Python.String으로 변환하는 기능을 구현했습니다. • 저장된 코드에서 EXECUTE 및 EXEC를 구현했습니다. • 테이블 유형의 사용을 구현했습니다.
Azure/SQL Server	PostgreSQL/Aurora PostgreSQL	$TMP 프로시저를 선택적으로 사용할 수 있도록 구현했습니다.
SQL Server	MySQL/ Aurora MySQL	날짜로 확장된 산술 연산. TOP(표현식) WITH TIES에 대한 생성 에뮬레이션 이제 생성된 refcursor out을 사용하여 프로시저를 호출한 후 refcursor가 닫힙니다. GLOBAL 격리 수준을 설정하는 것은 Aurora MySQL에서 지원되지 않습니다. 세션 범위만 변경할 수 있습니다. 트랜잭션의 기본 동작은 REPEATABLE READ와 일관된 읽기를 사용하는 것입니다. READ COMMITTED와 함께 사용하도록 설계된 애플리케이션을 수정해야 할 수 있습니다. 또는 기본값을 READ COMMITTED로 명시적으로 변경할 수 있습니다.

소스	대상	새로운 기능, 향상된 기능 또는 수정된 기능
SQL Server	AWS Glue (파이썬 셸)	SQL Server 문은 완전한 결과 집합을 생성하지만 결과를 한 번에 한 행씩 처리할 때 가장 잘 처리되는 경우가 있습니다. 결과 집합에서 커서를 열면 결과 집합을 한 번에 한 행씩 처리할 수 있습니다. 커서 데이터 형식을 사용하여 변수 또는 파라미터에 커서를 할당할 수 있습니다. Python은 흐름 제어로 SQL Server의 BEGIN 및 END를 지원하지 않지만 Transact-SQL 문 그룹을 실행할 수 있도록 저장된 코드에 일련의 Transact-SQL 문을 포함하는 기능을 구현했습니다. SQL Server LABEL 및 GOTO 문은 에서 AWS Glue지원되지 않습니다. AWS SCT 가 코드에서 레이블을 발견하면 해당 레이블을 건너뜁니다. AWS SCT 가 GOTO 문을 발견하면 이 문을 주석 처리합니다.
SQL Server	Amazon Redshift	IF ... ELSE 컨트롤을 구현하여 저장된 코드에 Transact-SQL 문의 조건부 처리를 구현했습니다. Transact-SQL 문 그룹을 블록으로 실행할 수 있도록 저장된 코드에 일련의 Transact-SQL 문을 포함하는 기능을 구현했습니다. 중첩된 BEGIN ... END 블록을 지원합니다. 저장된 코드에서 SET 및 SELECT를 구현했습니다. 테이블에 사용자 지정 정렬 키를 생성하여 Amazon Redshift(인덱스를 지원하지 않음)에서 CREATE INDEX를 구현했습니다.

릴리스 1.0.640 MySQL 변경 사항

다음 표에는 MySQL을 소스 엔진으로 사용하는 경우 해당되는 1.0.640 빌드 변경 사항이 나와 있습니다.

소스	대상	새로운 기능, 향상된 기능 또는 수정된 기능
MySQL	PostgreSQL 12.x	생성된 열에 대한 지원을 추가했습니다.

소스	대상	새로운 기능, 향상된 기능 또는 수정된 기능
모두	모두	JDK 8에서 Amazon Corretto JDK 11로 업그레이드되었습니다. 다운로드 링크를 포함한 자세한 내용은 Amazon Corretto 11 사용 설명서에서 Amazon Corretto 11이란?을 참조하세요.
모두	모두	사용자 데이터베이스에서 발생할 수 있는 불일치에 대한 정보를 평가 보고서에 추가했습니다.
MySQL	PostgreSQL/Aurora PostgreSQL 11.	다음에 대한 지원을 추가했습니다. • SQL 저장 프로시저 내의 내장 트랜잭션 • SQL 저장 프로시저를 호출하는 기능 • SQL 저장 프로시저를 생성하는 기능
모두	SQL Server	SQL Server 2019: 새로운 인덱스 속성 OPTIMIZE_FOR_SEQUENTIAL_KEY에 대한 지원을 추가했습니다. SQL Server 2017: 그래프 데이터베이스 노드 및 엣지 테이블 유형에 대한 지원을 추가했습니다. SQL Server 2016: 임시 테이블에 대한 지원을 추가했습니다.
모두	모두	가상 파티션으로 물리적 파티션을 재정의하는 기능을 구현했습니다. 데이터 웨어하우스 추출기가 생성된 가상 파티션에 따라 데이터를 추출합니다.

릴리스 1.0.640 PostgreSQL 변경 사항

다음 표에는 PostgreSQL을 소스 엔진으로 사용하는 경우 해당되는 1.0.640 빌드 변경 사항이 나와 있습니다.

소스	대상	새로운 기능, 향상된 기능 또는 수정된 기능
PostgreSQL	MySQL 8.x	

소스	대상	새로운 기능, 향상된 기능 또는 수정된 기능
		이제 MySQL은 열 값 대신 표현식 값을 인덱싱하는 기능 인덱스 키 파트 생성을 지원합니다. 기능 키 파트를 사용하면 JSON 값과 같이 인덱싱할 수 없는 값을 인덱싱할 수 있습니다. MySQL은 이제 Now CTE와 Recursive CTE를 지원합니다.
모두	모두	JDK 8에서 Amazon Corretto JDK 11로 업그레이드되었습니다. 다운로드 링크를 포함한 자세한 내용은 Amazon Corretto 11 사용 설명서에서 Amazon Corretto 11이란?을 참조하세요.
모두	모두	사용자 데이터베이스에서 발생할 수 있는 불일치에 대한 정보를 평가 보고서에 추가했습니다.
PostgreSQL 11.x	PostgreSQL/Aurora PostgreSQL 11.	다음에 대한 지원을 추가했습니다. • SQL 저장 프로시저 내의 내장 트랜잭션 • SQL 저장 프로시저를 호출하는 기능 • SQL 저장 프로시저를 생성하는 기능

소스	대상	새로운 기능, 향상된 기능 또는 수정된 기능
PostgreSQL	MySQL 8.x	이제 MySQL이 내림차순 인덱스를 지원합니다. 인덱스 정의의 DESC는 더 이상 무시되지 않으며 키 값이 내림차순으로 저장됩니다. 이제 MySQL이 데이터 형식 사양의 기본값으로 표현식을 사용할 수 있도록 지원합니다(예: 표현식을 BLOB, TEXT, GEOMETRY, 및 JSON 데이터 형식의 기본값으로 사용 가능). 이제 몇 가지 기존 집계 함수를 창 함수로 사용할 수 있습니다. • AVG() • BIT_AND() • BIT_OR() • BIT_XOR() • COUNT() • JSON_ARRAYAGG() • JSON_OBJECTAGG() • MAX() • MIN() • STDDEV_POP() • STDDEV() • STD() • STDDEV_SAMP() • SUM() • VAR_POP() • VARIANCE() • VAR_SAMP() MySQL은 쿼리의 각 행에 대해 해당 행과 관련된 행을 사용하여 계산을 수행하는 창 함수를 지원합니다. • CUME_DIST()

소스	대상	새로운 기능, 향상된 기능 또는 수정된 기능
		• DENSE_RANK() • FIRST_VALUE() • LAG() • LAST_VALUE() • LEAD() • NTH_VALUE() • NTILE() • PERCENT_RANK() • RANK() • ROW_NUMBER()
PostgreSQL	MySQL 8.x	CHECK 제약 조건에 대한 지원을 추가했습니다.
모두	SQL Server	SQL Server 2019: 새로운 인덱스 속성 OPTIMIZE_FOR_SEQUENTIAL_KEY에 대한 지원을 추가했습니다. SQL Server 2017: 그래프 데이터베이스 노드 및 엣지 테이블 유형에 대한 지원을 추가했습니다. SQL Server 2016: 임시 테이블에 대한 지원을 추가했습니다.
모두	모두	가상 파티션으로 물리적 파티션을 재정의하는 기능을 구현했습니다. 데이터 웨어하우스 추출기가 생성된 가상 파티션에 따라 데이터를 추출합니다.
PostgreSQL/Aurora PostgreSQL	모두	시스템 뷰 sysindexes 에뮬레이션을 추가했습니다. INTO가 지정되지 않고 프로시저에 SELECT 문이 있는 경우 대상 프로시저에 대해 refcursor 유형의 INOUT p_refcur 파라미터가 생성됩니다.

릴리스 1.0.640 Db2 LUW 변경 사항

다음 표에는 DB2 LUW를 소스 엔진으로 사용하는 경우 해당되는 1.0.640 빌드 변경 사항이 나와 있습니다.

소스	대상	새로운 기능, 향상된 기능 또는 수정된 기능
DB2 LUW	RDS MariaDB 10.4	모든 OLTP(Online Transactional Processing) 공급업체에 대한 RDS MariaDB 10.4 지원을 추가했습니다.
모두	모두	JDK 8에서 Amazon Corretto JDK 11로 업그레이드되었습니다. 다운로드 링크를 포함한 자세한 내용은 Amazon Corretto 11 사용 설명서에서 Amazon Corretto 11이란?을 참조하세요.
모두	모두	사용자 데이터베이스에서 발생할 수 있는 불일치에 대한 정보를 평가 보고서에 추가했습니다.
DB2 LUW	MySQL 8.0.17	CHECK 제약 조건에 대한 지원을 추가했습니다.
모두	SQL Server	SQL Server 2019: 새로운 인덱스 속성 OPTIMIZE_FOR_SEQUENTIAL_KEY에 대한 지원을 추가했습니다. SQL Server 2017: 그래프 데이터베이스 노드 및 엣지 테이블 유형에 대한 지원을 추가했습니다. SQL Server 2016: 임시 테이블에 대한 지원을 추가했습니다.
모두	모두	가상 파티션으로 물리적 파티션을 재정의하는 기능을 구현했습니다. 데이터 웨어하우스 추출기가 생성된 가상 파티션에 따라 데이터를 추출합니다.

릴리스 1.0.640 Teradata 변경 사항

다음 표에는 Teradata 소스 엔진을 사용하는 경우 해당되는 빌드 1.0.640 변경 사항이 나와 있습니다.

소스	대상	새로운 기능, 향상된 기능 또는 수정된 기능
Teradata	Amazon Redshift	MERGE 및 QUALIFY 문에 대한 지원을 추가했습니다. Teradata 문에서 LOCKING ROWS FOR ACCESS 절을 제거했습니다. CAST 함수에 대한 지원을 추가했습니다.
모두	모두	JDK 8에서 Amazon Corretto JDK 11로 업그레이드되었습니다. 다운로드 링크를 포함한 자세한 내용은 Amazon Corretto 11 사용 설명서에서 Amazon Corretto 11이란?을 참조하세요.
Teradata	Teradata	REGEXP_INSTR() 및 REGEXP_SUBSTR()의 개선 사항을 구현했습니다.
모두	모두	사용자 데이터베이스에서 발생할 수 있는 불일치에 대한 정보를 평가 보고서에 추가했습니다.
모두	SQL Server	SQL Server 2019: 새로운 인덱스 속성 OPTIMIZE_FOR_SEQUENTIAL_KEY에 대한 지원을 추가했습니다. SQL Server 2017: 그래프 데이터베이스 노드 및 엣지 테이블 유형에 대한 지원을 추가했습니다. SQL Server 2016: 임시 테이블에 대한 지원을 추가했습니다.
Teradata	모두	REGEXP_INSTR() 및 REGEXP_SUBSTR()에 대한 지원을 추가했습니다.
모두	모두	가상 파티션으로 물리적 파티션을 재정의하는 기능을 구현했습니다. 데이터 웨어하우스 추출기가 생성된 가상 파티션에 따라 데이터를 추출합니다.
Teradata	Amazon Redshift	프로젝트 설정, SQL로 저장 및 적용, 드롭다운 목록: 단일 파일/다중 파일의 설정을 사용하여 소스 트리의 SQL을 단계별로 단일 파일 또는 여러 파일에 저장하는 기능을 구현했습니다.

소스	대상	새로운 기능, 향상된 기능 또는 수정된 기능
		뷰 및 프로시저 변환의 개선 사항을 구현했습니다.
Teradata	모두	Teradata 버전 16.20에 대한 지원을 추가했습니다.

기타 엔진과 관련된 릴리스 1.0.640 변경 사항

다음 표에는 기타 소스 엔진을 사용하는 경우 해당되는 1.0.640 빌드 변경 사항이 나와 있습니다.

소스	대상	새로운 기능, 향상된 기능 또는 수정된 기능
Sybase	RDS MariaDB 10.4	모든 OLTP(Online Transactional Processing) 공급업체에 대한 RDS MariaDB 10.4 지원을 추가했습니다.
SAP ASE	MariaDB	다음을 구현했습니다. • MariaDB 10.4 • EXECUTE IMMEDIATE 문 • DEFAULT 정의 • CHECK 제약 조건 지원
SAP ASE	PostgreSQL 12.x	생성된 열에 대한 지원을 추가했습니다.
모두	모두	JDK 8에서 Amazon Corretto JDK 11로 업그레이드되었습니다. 다운로드 링크를 포함한 자세한 내용은 Amazon Corretto 11 사용 설명서에서 Amazon Corretto 11이란?을 참조하세요.
모두	모두	사용자 데이터베이스에서 발생할 수 있는 불일치에 대한 정보를 평가 보고서에 추가했습니다.
SAP ASE	MySQL 8.0.17	CHECK 제약 조건에 대한 지원을 추가했습니다.
모두	SQL Server	

소스	대상	새로운 기능, 향상된 기능 또는 수정된 기능
		SQL Server 2019: 새로운 인덱스 속성 OPTIMIZE_FOR_SEQUENTIAL_KEY에 대한 지원을 추가했습니다. SQL Server 2017: 그래프 데이터베이스 노드 및 엣지 테이블 유형에 대한 지원을 추가했습니다. SQL Server 2016: 임시 테이블에 대한 지원을 추가했습니다.
Vertica	Amazon Redshift	배포 스타일 = AUTO에 대한 지원을 추가했습니다.
모두	모두	가상 파티션으로 물리적 파티션을 재정의하는 기능을 구현했습니다. 데이터 웨어하우스 추출기가 생성된 가상 파티션에 따라 데이터를 추출합니다.
Amazon Redshift	Amazon Redshift	DML 문에서 지원되지 않는 기본 제공 함수는 자리 표시자 NULL로 대체됩니다.
Sybase	PostgreSQL	네이티브 함수에 대한 지원을 추가했습니다.
SAP ASE	MySQL/ Aurora MySQL	Aurora MySQL의 기본 격리 수준은 REPEATABLE READ입니다. GLOBAL 격리 수준을 설정하는 것은 Aurora MySQL에서 지원되지 않습니다. 세션 범위만 변경할 수 있습니다. 트랜잭션의 기본 동작은 REPEATABLE READ와 일관된 읽기를 사용하는 것입니다. READ COMMITTED와 함께 실행하도록 설계된 애플리케이션을 수정해야 할 수 있습니다. 또는 기본값을 READ COMMITTED로 명시적으로 변경할 수 있습니다.
SAP ASE	PostgreSQL	확장 팩 없이 CONVERT 함수(낙관적)에 대한 지원을 추가했습니다.

소스	대상	새로운 기능, 향상된 기능 또는 수정된 기능
SAP ASE	모두	시스템 뷰 sysindexes 에뮬레이션을 추가했습니다. INTO가 지정되지 않고 프로시저에 SELECT 문이 있는 경우 대상 프로시저에 대해 refcursor 유형의 INOUT p_refcur 파라미터가 생성됩니다.
Greenplum	Amazon Redshift	다음과 같이 CREATE TEMPORARY TABLE을 구현했습니다.

문서 기록

다음 표는 2018년 1월 이후 AWS Schema Conversion Tool(AWS SCT) 사용 설명서에서 변경된 중요 사항을 기술한 것입니다.

RSS 피드를 구독하면 이 설명서에 대한 업데이트 알림을 받을 수 있습니다.

변경 사항	설명	날짜
AWS SCT 빌드 #1.0.672	빌드 1.0.672는 Amazon RDS for PostgreSQL 15를 대상으로 지원하고 Microsoft SQL Server 버전 2022를 소스로 지원합니다. 또한 변환된 코드에 새로운 Amazon Redshift 기능에 대한 지원을 추가하고, IBM Db2 for z/OS 소스에 대한 여러 변환 개선 사항을 구현하고, 다양한 변환 문제를 해결합니다.	2023년 5월 8일
AWS SCT 빌드 #1.0.671	빌드 1.0.671은 Apache Oozie에서 AWS Step Functions로의 마이그레이션을 지원합니다. 또한 다중 서버 평가 프로세스의 소스로 BigQuery에 대한 지원을 추가합니다. 또한 IBM Db2 for z/OS에 대한 새로운 변환 설정을 소스로 추가하고 여러 변환 문제를 해결합니다.	2023년 3월 8일
AWS SCT 빌드 #1.0.670	빌드 1.0.670은 Hadoop에서 Amazon EMR로의 마이그레이션을 지원합니다. 또한 다중 서버 평가 프로세스의 소스로 Azure Synapse Analytics에 대한 지원을 추가합니다. 또한 Java 애플리케이션에 포함된	2023년 1월 23일

	SQL 코드의 변환을 개선하고 여러 변환 문제를 해결합니다.	
AWS SCT 빌드 #1.0.669	빌드 1.0.669는 Oracle 데이터 웨어하우스에서 데이터 마이그레이션에 필요한 네이티브 파티셔닝 지원을 구현합니다. 또한 다중 서버 평가 프로세스를 개선하고, 데이터 추출 에이전트에 새로운 기능을 추가하고, 여러 변환 문제를 해결합니다.	2022년 12월 19일
AWS SCT 빌드 #1.0.668	빌드 1.0.668은 Greenplum 데이터베이스에서 데이터를 마이그레이션하기 위한 자동 가상 파티셔닝을 구현하고 Snowflake 데이터베이스에서 Amazon Redshift로의 데이터 마이그레이션에 대한 지원을 추가합니다. 또한 C# 애플리케이션에 포함된 SQL 코드의 변환을 개선하고 여러 변환 문제를 해결합니다.	2022년 11월 16일
AWS SCT 빌드 #1.0.667	빌드 1.0.667은 Informatica 추출, 전환, 적재(ETL) 엔진을 마이그레이션 소스로 지원합니다. 또한 확장 팩 버전을 업데이트하고, Amazon Redshift에 지원되는 최소 드라이버 버전을 증가시키고, 여러 변환 문제를 해결합니다.	2022년 10월 13일

AWS SCT 빌드 #1.0.666	빌드 1.0.666은 MyBatis 프레임워크에 대한 지원을 추가하여 Java 애플리케이션의 변환을 개선합니다. 또한 확장 팩에 새 함수를 추가하고 소스 메타데이터 로더를 개선하며 여러 변환 문제를 해결합니다.	2022년 9월 20일
AWS SCT 빌드 #1.0.665	빌드 1.0.665는 BigQuery를 마이그레이션 소스로 지원합니다. Babelfish 기능 구성 파일의 새 버전에 대한 지원도 구현합니다. 또한 데이터 웨어하우스를 Amazon Redshift로 변환하는 기능을 개선하고 여러 변환 문제를 해결합니다.	2022년 8월 29일
AWS SCT 빌드 #1.0.664	빌드 1.0.664는 Amazon Redshift Serverless를 마이그레이션 소스 또는 대상으로 지원합니다. 또한 데이터 추출 작업에서 자동 메모리 밸런싱을 구현하고 AWS SCT가 AWS Snowball 디바이스에 연결할 수 없는 오류를 수정합니다. 또한 마이그레이션 규칙에 열 데이터 정렬을 변경하는 기능을 추가하고 사용자 인터페이스를 개선하고 여러 변환 문제를 해결합니다.	2022년 7월 14일

AWS SCT 빌드 #1.0.663	빌드 1.0.663은 Babelfish for Aurora PostgreSQL 1.2.0에 대한 지원을 추가하고 다중 서버 평가 보고서 기능을 개선합니다. 또한 마이그레이션 규칙에 새 기능을 추가하고 두 개의 로더 오류를 수정하며 여러 변환 문제를 해결합니다.	2022년 6월 20일
AWS SCT 빌드 #1.0.662	빌드 1.0.662는 C# 애플리케이션에서 SQL 코드 변환을 구현하고 다중 서버 평가 보고서 워크플로를 개선합니다. 또한 여러 가지 변환 개선 사항을 추가하고 여러 변환 문제를 해결합니다.	2022년 5월 19일
AWS SCT 빌드 #1.0.661	빌드 1.0.661은 IBM Db2 for z/OS를 마이그레이션 소스로 지원합니다. 또한 추출, 전환, 적재(ETL) 스크립트의 변환 지원을 AWS Glue Studio에 추가하고 여러 변환 문제를 해결합니다.	2022년 4월 21일
AWS SCT 빌드 #1.0.660	빌드 1.0.660은 PostgreSQL 메이저 버전 14와 MariaDB 10.6을 마이그레이션 대상으로 지원합니다. 또한 구체화된 뷰에 대한 Oracle 인덱스 변환에 대한 지원을 추가하고 여러 변환 문제를 해결합니다.	2022년 3월 21일

AWS SCT 빌드 #1.0.659	빌드 1.0.659는 Aurora PostgreSQL-Compatible Edition의 PostgreSQL 메이저 버전 13을 마이그레이션 대상으로 지원합니다. C# 애플리케이션에서 SQL 코드 변환을 구현하고 Oracle Unified Auditing 지원을 추가하고 여러 변환 문제를 해결합니다.	2022년 2월 21일
AWS SCT 빌드 #1.0.658	빌드 1.0.658은 AWS Secrets Manager와의 통합을 제공하고 Amazon Redshift 가상 대상 데이터베이스 플랫폼 지원을 추가합니다. 또한 여러 변환 개선 사항 및 버그 수정이 추가되었습니다.	2022년 1월 20일
AWS SCT 빌드 #1.0.657	빌드 1.0.657은 Microsoft SQL Server에서 Aurora PostgreSQL-Compatible Edition, Amazon RDS for PostgreSQL 및 기타 마이그레이션 대상으로의 변환을 개선합니다. 또한 다양한 사용자 인터페이스 개선 사항 및 버그 수정이 추가되었습니다.	2021년 12월 20일
AWS SCT 빌드 #1.0.656	빌드 1.0.656은 하나의 프로젝트에서 여러 소스 및 대상 데이터베이스를 지원합니다. 또한 변환, 최적화 전략, 일반 개선 사항 및 여러 버그 수정이 추가되었습니다.	2021년 11월 22일

AWS SCT 빌드 #1.0.655	빌드 1.0.655는 Teradata FastExport 작업 스크립트를 Amazon Redshift RSQL로 변환하는 기능을 구현하고 Greenplum에 지원되는 최소 드라이버 버전을 42.2.5로 증가했습니다. 또한 여러 개선 사항과 버그 수정을 추가합니다.	2021년 10월 18일
AWS SCT 빌드 #1.0.654	빌드 1.0.654는 쉘, Teradata FastLoad 및 Teradata Basic Teradata Query(BTEQ) 스크립트를 Amazon Redshift RSQL로 변환하는 기능을 구현합니다. 또한 여러 변환 문제를 해결하고 여러 개선 사항과 버그 수정을 추가합니다.	2021년 9월 16일
AWS SCT 빌드 #1.0.653	빌드 1.0.653은 호출된 함수 또는 프로시저에서 생성된 동적 SQL의 변환을 구현합니다. 또한 암호화된 루틴의 변환을 개선하고 여러 개선 사항과 버그 수정을 추가합니다.	2021년 8월 10일
AWS SCT 빌드 #1.0.652	빌드 1.0.652는 명령줄 인터페이스에서 스크립트 명령 모드를 구현하고 스키마 최적화 규칙을 구현합니다. 또한 여러 변환 및 성능 개선과 버그 수정을 추가합니다.	2021년 6월 30일
AWS SCT 빌드 #1.0.651	빌드 1.0.651은 여러 개선 사항과 버그 수정을 추가합니다. 또한 AWS Schema Conversion Tool CLI 참조의 초기 사본에 대한 액세스를 제공합니다.	2021년 6월 4일

AWS SCT 빌드 #1.0.650	빌드 1.0.650은 Amazon RDS for PostgreSQL 13을 대상 데이터베이스로 지원하는 기능을 구현하고 추출기 에이전트를 업데이트합니다. Microsoft SQL Server, Azure 및 Azure Synapse에 대해 지원되는 최소 JDBC 드라이버 버전도 업그레이드합니다. 또한 여러 변환 개선 사항 및 버그 수정을 추가합니다.	2021년 4월 30일
AWS SCT 빌드 #1.0.649	빌드 1.0.649는 MariaDB 10.5를 대상 데이터베이스로 지원하고 Oracle 내장 함수의 변환을 위한 함수 개선 사항을 구현합니다. 또한 여러 변환 및 성능 개선과 버그 수정을 추가합니다.	2021년 3월 29일
AWS SCT 빌드 #1.0.648	빌드 1.0.648은 여러 변환 개선 사항과 버그 수정을 추가합니다.	2021년 2월 22일
AWS SCT 빌드 #1.0.647	빌드 1.0.647은 Amazon RDS에서 Database Mail 기능에 대한 지원을 추가하고, 스토리지 객체에 대한 주석의 로드 및 변환을 구현합니다. 또한 AWS SCT Data Migration Service Assessor 및 AWS SCT 마법사를 추가하고 트리 필터 사용자 인터페이스를 구현합니다. 또한 평가 보고서의 재설계된 섹션과 여러 개선 사항 및 버그 수정이 추가되었습니다.	2021년 1월 15일

AWS SCT 빌드 #1.0.646	빌드 1.0.646은 INTERVAL 데이터 유형, ID 열 및 커서 변환에 대한 지원을 추가하고 여러 개선 사항과 버그 수정을 추가합니다.	2020년 12월 28일
AWS SCT 빌드 #1.0.645	빌드 1.0.645은 AWS Glue 변환에 대한 ETL SSIS 지원과 여러 개선 사항 및 버그 수정을 추가합니다.	2020년 11월 16일
AWS SCT 빌드 #1.0.643~1.0.644	빌드 1.0.644는 여러 가지 변환, 성능 및 사용자 인터페이스 개선 사항과 버그 수정을 추가합니다.	2020년 10월 14일
AWS SCT 빌드 #1.0.642	빌드 1.0.642는 Microsoft SQL Server Integration Services에서 AWS Glue로 ETL 패키지 변환을 구현하고 여러 개선 사항과 버그 수정을 추가합니다.	2020년 8월 28일
AWS SCT 빌드 #1.0.641	데이터 추출기에 대한 SSL 지원이 추가되었습니다. 또한 빌드에는 여러 개선 사항 및 수정 사항도 포함됩니다.	2020년 7월 17일
AWS SCT 빌드 #1.0.633~1.0.640	JDK 8에서 Amazon Corretto JDK 11로 업그레이드되었습니다. 다른 업그레이드, 변경 사항 및 수정 사항을 정리한 표가 추가되었습니다.	2020년 6월 22일
AWS WQF 가용성	AWS SCT에서 더 이상 AWS WQF(AWS Workload Qualification Framework) 도구의 다운로드를 제공하지 않습니다.	2020년 6월 19일

AWS SCT 빌드 #1.0.632	SCT UI - 스크립트를 적용할 때 발생하는 오류를 표시하는 새 탭이 추가되었습니다. 이제 SAP ASE에서 변환할 때 소스 트리를 SQL로 저장할 수 있습니다. PostgreSQL, Aurora PostgreSQL 또는 Redshift로 변환하는 작업에 대한 개선 사항입니다.	2019년 11월 19일
AWS SCT 빌드 #1.0.631 및 #1.0.630(통합됨)	Oracle의 ROWID와 Microsoft SQL Server 및 SAP ASE의 시스템 객체에 대한 지원이 개선되었습니다. SQL Server 스키마의 누락된 지정자에 대한 처리가 개선되었습니다. Greenplum에서 Redshift로의 변환에 대한 지원이 개선되었습니다. Amazon Redshift, MariaDB, MySQL 및 PostgreSQL로 이동할 때 저장된 코드의 변환에 대한 지원이 향상되었습니다.	2019년 9월 30일
AWS SCT 빌드 #1.0.629	Netezza에서 변환하기 위한 저장 프로시저에 대한 지원. Amazon Redshift, DynamoDB, MySQL 및 PostgreSQL로의 변환에 대한 지원이 개선되었습니다. 소스로 SAP ASE 12.5 지원 추가	2019년 8월 20일

AWS SCT 빌드 #1.0.628	DB2, SQL Server 및 Oracle에서 변환하기 위한 서비스 에뮬레이션에 대한 지원. 커서 및 저장 프로시저에 대한 추가 지원을 비롯한 Amazon Redshift로의 변환 개선	2019년 6월 22일
AWS SCT 빌드 #1.0.627	SQL Server에서 Amazon Redshift의 저장 프로시저로 변환 지원. PostgreSQL 11 및 MySQL 8.0으로 변환하는 작업에 대한 개선 사항.	2019년 5월 31일
AWS SCT 빌드 #1.0.626	PostgreSQL 11 및 MySQL 8.0은 현재 대상으로 지원됩니다. SAP ASE 15.5는 현재 대상으로 지원됩니다.	2019년 4월 26일
AWS SCT 빌드 #1.0.625	업데이트에는 Teradata BTEQ에서 AWS Glue로 변환하는 기능, Oracle 호환성 모드 지원을 이용한 MariaDB 10.3 변환 지원, SAP ASE 15.7 지원, 누락된 기능 에뮬레이션을 위한 서비스 구독 등이 포함됩니다.	2019년 3월 25일

AWS SCT 빌드 #1.0.624	업데이트에는 Oracle ETL에서 AWS Glue로 변환하는 기능 및 Microsoft SQL Server, Oracle, IBM Db2 LUW에서 Amazon RDS for MariaDB로 변환 지원 등이 포함됩니다. MySQL 호환성을 이용한 SAP ASE에서 RDS for MySQL 및 Amazon Aurora로 변환 지원도 추가했습니다. 또한 Oracle을 PostgreSQL로 변환할 때 Orafce 확장에 대한 지원도 추가했습니다.	2019년 2월 22일
AWS SCT 빌드 #1.0.623	업데이트에는 SAP ASE 데이터베이스 변환 기능과 T-SQL 스크립트, DML, DDL을 동등 코드나 구성 요소로 변환하는 기능이 포함됩니다. 변환 개선을 위한 Oracle 및 Microsoft SQL Server 에뮬레이션도 추가했습니다.	2019년 1월 25일
AWS SCT 빌드 #1.0.622	업데이트에는 데이터베이스 및 앱 수정 사항을 비롯해 전체 마이그레이션에 필요한 워크로드를 분석하는 Workload Qualification이 포함되어 있습니다.	2018년 12월 20일
AWS SCT 빌드 #1.0.621	업데이트에는 Aurora PostgreSQL 10을 대상으로 지원, 외부 테이블 옵션을 사용하여 Netezza에서 마이그레이션하는 기능 등이 포함됩니다.	2018년 11월 21일

AWS SCT 빌드 #1.0.620	업데이트에는 SQL 스크립트 저장 기능과 MySQL로 마이그레이션 시 Oracle 전역 커서 지원이 포함됩니다.	2018년 10월 22일
AWS SCT 빌드 #1.0.619	업데이트에는 Apache Cassandra에서 DynamoDB로 마이그레이션하기 위한 지원과 Vertica 9를 소스로 사용할 수 있는 지원이 포함되어 있습니다.	2018년 9월 20일
AWS SCT 빌드 #1.0.618	업데이트에는 확장된 평가 보고서, Oracle ROWID 변환 지원, SQL Server 사용자 정의 테이블 지원이 포함됩니다.	2018년 8월 24일
AWS SCT 빌드 #1.0.617	업데이트에는 확장된 평가 보고서, Oracle ROWID 변환 지원, SQL Server 사용자 정의 테이블 지원이 포함됩니다.	2018년 7월 24일
AWS SCT 빌드 #1.0.616	업데이트에는 Oracle에서 Amazon RDS for Oracle로 변환 시 RDS 지원, Oracle 일정 객체 변환, Oracle 작업 지원, 파티셔닝 및 Db2 LUW 버전 10.1 지원 등이 포함됩니다.	2018년 26월 6일
AWS SCT 빌드 #1.0.615	업데이트에는 SQL Server에서 PostgreSQL로 마이그레이션 GOTO 문 지원, PostgreSQL 10 파티셔닝 및 Db2 LUW 버전 10.1 지원이 포함됩니다.	2018년 5월 24일

AWS SCT 빌드 #1.0.614	업데이트에는 Oracle에서 Oracle DB Links로 마이그레이션 지원, SQL Server에서 PostgreSQL에 대응되는 인라인 함수, Oracle 시스템 객체 에뮬레이션이 포함됩니다.	2018년 25월 4일
AWS SCT 빌드 #1.0.613	업데이트에는 Db2 LUW 지원, SQL*Plus 파일 변환, SQL Server Windows 인증이 포함됩니다.	2018년 3월 28일
AWS SCT 빌드 #1.0.612	업데이트에는 사용자 지정 데이터 형식 매핑 지원, Oracle 10용 스키마 비교, Oracle에서 PostgreSQL로 전역 변수 변환이 포함됩니다.	2018년 2월 22일
AWS SCT 빌드 #1.0.611	업데이트에는 Oracle에서 PostgreSQL로 동적 문 변환 지원, 오류 메시지 선택을 통해 로그 파일 열기, 트리 보기에서 스키마를 숨기는 기능이 포함됩니다.	2018년 1월 23일

이전 업데이트

다음 표는 2018년 1월 이전 AWS Schema Conversion Tool(AWS SCT) 사용 설명서에서 변경된 중요 사항을 기술한 것입니다.

버전	변경 사항	설명	변경 날짜
1.0.608	Amazon S3용 FIPS 엔드포인트 지원	이제 FIPS 엔드포인트를 사용하여 Amazon S3 및 Amazon Redshift에 연결하도록 AWS SCT에 요청하여 미연방 정보 처리 표준 보안 요구 사항을 준수할 수 있습니다. 자세한 내	2017년 11월 17일

버전	변경 사항	설명	변경 날짜
		용은 AWS 보안 인증 정보 저장 섹션을 참조하세요.	
1.0.607	Amazon S3용 FIPS 엔드포인트 지원	이제 FIPS 엔드포인트를 사용하여 Amazon S3 및 Amazon Redshift에 연결하도록 AWS SCT에 요청하여 미연방 정보 처리 표준 보안 요구 사항을 준수할 수 있습니다. 자세한 내용은 AWS 보안 인증 정보 저장 섹션을 참조하세요.	2017년 10월 30일
1.0.607	데이터 추출 작업에서 LOB 무시 가능	이제 데이터 추출 작업을 생성할 때 대용량 객체(LOB)를 무시하여 추출하는 데이터 양을 줄일 수 있습니다. 자세한 내용은 AWS SCT 데이터 추출 작업 생성, 실행, 모니터링 섹션을 참조하세요.	2017년 10월 30일
1.0.605	데이터 추출 에이전트 작업 로그 액세스	이제 AWS Schema Conversion Tool 사용자 인터페이스에서 편리한 링크를 통해 데이터 추출 에이전트 작업 로그에 액세스할 수 있습니다. 자세한 내용은 AWS SCT 데이터 추출 작업 생성, 실행, 모니터링 섹션을 참조하세요.	2017년 8월 28일
1.0.604	컨버터 기능 향상	이종 마이그레이션에서 변환을 향상하기 위해 AWS Schema Conversion Tool 엔진을 개선했습니다.	2017년 6월 24일
1.0.603	데이터 추출 에이전트가 필터를 지원	이제 추출 에이전트가 데이터 웨어하우스에서 추출하는 데이터를 필터링할 수 있습니다. 자세한 내용은 에서 데이터 마이그레이션 규칙 생성 AWS SCT 섹션을 참조하세요.	2017년 6월 16일

버전	변경 사항	설명	변경 날짜
1.0.603	AWS SCT에서 추가 데이터 웨어하우스 버전을 지원	이제 AWS Schema Conversion Tool을 사용하여 Teradata 13 및 Oracle Data Warehouse 10 스키마를 그와 동등한 Amazon Redshift 스키마로 변환할 수 있습니다. 자세한 내용은 AWS SCT를 사용하여 데이터 웨어하우스 스키마를 Amazon Redshift로 변환 섹션을 참조하세요.	2017년 6월 16일
1.0.602	데이터 추출 에이전트가 추가 데이터 웨어하우스를 지원	이제 데이터 추출 에이전트를 사용해 Microsoft SQL Server 데이터 웨어하우스에서 데이터를 추출할 수 있습니다. 자세한 내용은 온프레미스 데이터 웨어하우스에서 Amazon Redshift로 데이터 마이그레이션 섹션을 참조하세요.	2017년 5월 11일
1.0.602	데이터 추출 에이전트가 데이터를 Amazon Redshift로 복사 가능	이제 데이터 추출 에이전트는 세 가지 업로드 모드를 제공합니다. 데이터 추출만, 데이터 추출 후 Amazon S3으로 업로드 또는 데이터 추출, 업로드 및 Amazon Redshift에 직접 복사 중에서 지정할 수 있습니다. 자세한 내용은 AWS SCT 데이터 추출 작업 생성, 실행, 모니터링 섹션을 참조하세요.	2017년 5월 11일
1.0.601	AWS SCT가 추가 데이터 웨어하우스를 지원	이제 AWS Schema Conversion Tool을 사용하여 Vertica 및 Microsoft SQL Server 스키마를 동등한 Amazon Redshift 스키마로 변환할 수 있습니다. 자세한 내용은 AWS SCT를 사용하여 데이터 웨어하우스 스키마를 Amazon Redshift로 변환 섹션을 참조하세요.	2017년 4월 18일

버전	변경 사항	설명	변경 날짜
1.0.601	데이터 추출 에이전트가 추가 데이터 웨어하우스를 지원	이제 데이터 추출 에이전트를 사용해 Greenplum, Netezza 및 Vertica 데이터 웨어하우스로부터 데이터를 추출할 수 있습니다. 자세한 내용은 온프레미스 데이터 웨어하우스에서 Amazon Redshift로 데이터 마이그레이션 섹션을 참조하세요.	2017년 4월 18일
1.0.601	데이터 추출 에이전트가 추가 운영 체제를 지원	이제 macOS 및 Microsoft Windows 운영 체제를 실행하는 컴퓨터에 데이터 추출 에이전트를 설치할 수 있습니다. 자세한 내용은 추출 에이전트 설치 섹션을 참조하세요.	2017년 4월 18일
1.0.601	데이터 추출 에이전트가 Amazon S3으로 자동 업로드	이제 데이터 추출 에이전트가 추출된 데이터를 자동으로 Amazon S3으로 업로드합니다. 자세한 내용은 데이터 추출 작업 출력 섹션을 참조하세요.	2017년 4월 18일
1.0.600	데이터 추출 에이전트	이제 데이터 웨어하우스로부터 데이터를 추출하여 Amazon Redshift에서 사용하도록 준비해 주는 데이터 추출 에이전트를 설치할 수 있습니다. AWS Schema Conversion Tool을 사용하여 에이전트를 등록하고 에이전트를 위한 데이터 추출 작업을 생성할 수 있습니다. 자세한 내용은 온프레미스 데이터 웨어하우스에서 Amazon Redshift로 데이터 마이그레이션 섹션을 참조하세요.	2017년 2월 16일
1.0.600	고객 피드백	이제 사용자는 AWS Schema Conversion Tool에 대한 피드백을 제공할 수 있습니다. 버그 보고서를 제출하거나, 기능 요청을 제출하거나, 일반 정보를 제공할 수 있습니다. 자세한 내용은 피드백 제공 섹션을 참조하세요.	2017년 2월 16일

버전	변경 사항	설명	변경 날짜
1.0.502	AWS DMS과(와)의 통합	이제 AWS Schema Conversion Tool을 사용하여 AWS DMS 엔드포인트 및 작업을 생성할 수 있습니다. AWS SCT로부터 이러한 작업을 실행하고 모니터링할 수 있습니다. 자세한 내용은 AWS SCT와 함께 AWS DMS 사용 섹션을 참조하세요.	2016년 12월 20일
1.0.502	대상 데이터베이스로서 Amazon Aurora(PostgreSQL)의 호환성	이제 AWS Schema Conversion Tool은 대상 데이터베이스로서 Amazon Aurora(PostgreSQL)의 호환성을 지원합니다. 자세한 내용은 AWS SCT를 사용하여 데이터베이스 스키마 변환 섹션을 참조하세요.	2016년 12월 20일
1.0.502	프로필 지원	이제 AWS Schema Conversion Tool에 여러 프로필을 저장하고 간편하게 전환하여 사용할 수 있습니다. 자세한 내용은 AWS SCT에 AWS 서비스 프로필 저장 섹션을 참조하세요.	2016년 12월 20일
1.0.501	Greenplum Database 및 Netezza 지원	이제 AWS Schema Conversion Tool을 사용하여 Greenplum Database 및 Netezza에서 Amazon Redshift로 데이터 웨어하우스 스키마를 변환할 수 있습니다. 자세한 내용은 AWS SCT를 사용하여 데이터 웨어하우스 스키마를 Amazon Redshift로 변환 섹션을 참조하세요.	2016년 11월 17일
1.0.501	Redshift 최적화	이제 AWS Schema Conversion Tool을 사용하여 Amazon Redshift 데이터베이스를 최적화할 수 있습니다. 자세한 내용은 AWS SCT를 사용하여 Amazon Redshift 최적화 섹션을 참조하세요.	2016년 11월 17일

버전	변경 사항	설명	변경 날짜
1.0.500	매핑 규칙	AWS Schema Conversion Tool를 사용하여 스키마를 변환하기 전에 이제 열의 데이터 형식을 변경하고 객체를 한 스키마에서 다른 스키마로 이동하며 객체 이름을 변경하는 규칙을 설정할 수 있습니다. 자세한 내용은 AWS SCT에서 마이그레이션 규칙 생성 섹션을 참조하세요.	2016년 10월 4일
1.0.500	클라우드로 이전	이제 AWS Schema Conversion Tool을 사용하여 기존의 온프레미스 데이터베이스 스키마를 동일한 엔진을 실행하는 Amazon RDS DB 인스턴스로 복사할 수 있습니다. 이 기능을 사용하면 클라우드로 이전하고 라이선스 유형을 변경하는 데 따르는 비용 절감의 가능성을 분석할 수 있습니다. 자세한 내용은 AWS SCT를 사용하여 마이그레이션 평가 보고서 작성 섹션을 참조하세요.	2016년 10월 4일
1.0.400	데이터 웨어하우스 스키마 변환	이제 AWS Schema Conversion Tool을 사용하여 Oracle 및 Teradata에서 Amazon Redshift로 데이터 웨어하우스 스키마를 변환할 수 있습니다. 자세한 내용은 AWS SCT를 사용하여 데이터 웨어하우스 스키마를 Amazon Redshift로 변환 섹션을 참조하세요.	2016년 7월 13일
1.0.400	애플리케이션 SQL 변환	이제 AWS Schema Conversion Tool을 사용하여 C++, C#, Java 또는 기타 애플리케이션 코드로 된 SQL을 변환할 수 있습니다. 자세한 내용은 AWS SCT를 사용하여 애플리케이션 SQL 변환 섹션을 참조하세요.	2016년 7월 13일

버전	변경 사항	설명	변경 날짜
1.0.400	새로운 기능	이제 AWS Lambda 함수 및 Python 라이브러리를 설치, 생성 및 구성하여 이메일, 작업 일정 및 기타 기능을 제공하는 데 활용할 수 있는 확장 팩 및 마법사가 AWS Schema Conversion Tool에 포함되었습니다. 자세한 정보는 AWS SCT 확장 팩의 AWS Lambda 함수 사용 및 AWS SCT 확장 팩용 사용자 지정 라이브러리 사용 섹션을 참조하세요.	2016년 7월 13일
1.0.301	SSL 지원	이제 AWS Schema Conversion Tool을 사용할 때 Secure Sockets Layer(SSL)를 사용하여 소스 데이터베이스에 연결할 수 있습니다.	2016년 5월 19일
1.0.203	새로운 기능	변환을 위한 소스 데이터베이스로 MySQL 및 PostgreSQL을 추가 지원합니다.	2016년 4월 11일
1.0.202	유지 관리 릴리스	대상 데이터베이스 엔진용으로 생성된 변환된 SQL의 편집 지원을 추가합니다. 소스 데이터베이스 및 대상 DB 인스턴스 트리 보기에 향상된 선택 기능을 추가합니다. TNS(Transparent Network Substrate) 이름을 사용하는 Oracle 소스 데이터베이스 연결에 대한 지원을 추가합니다.	2016년 3월 2일
1.0.200	유지 관리 릴리스	대상 데이터베이스 엔진으로 PostgreSQL을 추가 지원합니다. 변환된 스키마를 스크립트로 생성하고 그 스크립트를 파일로 저장한 다음 해당 스키마를 대상 DB 인스턴스에 적용하는 기능을 추가합니다.	2016년 1월 14일
1.0.103	유지 관리 릴리스	오프라인 프로젝트 기능, 신규 버전 확인 기능 및 메모리/성능 관리 기능을 추가합니다.	2015년 12월 2일

버전	변경 사항	설명	변경 날짜
1.0.101	유지 관리 릴리스	Create New Database Migration Project(데이터베이스 마이그레이션 프로젝트 새로 만들기) 마법사를 추가합니다. 데이터베이스 마이그레이션 평가 보고서를 PDF 파일로 저장하는 기능을 추가합니다.	2015년 10월 19일
1.0.100	미리 보기 릴리스	AWS Schema Conversion Tool 미리 보기 릴리스용 사용 설명서를 제공합니다.	2015년 10월 7일

기계 번역으로 제공되는 번역입니다. 제공된 번역과 원본 영어의 내용이 상충하는 경우에는 영어 버전이 우선합니다.