Mengoptimalkan kinerja kueri Postgre SQL

AWS Bimbingan Preskriptif

AWS Bimbingan Preskriptif: Mengoptimalkan kinerja kueri Postgre SQL

Table of Contents

Mengoptimalkan kinerja kueri PostgreSQL

Amazon Web Services (kontributor)

April 2024 (riwayat dokumen)

PostgreSQL adalah sistem database objek-relasional open source yang kuat, fleksibel, dan andal. Ada banyak cara untuk mengoptimalkan kinerja kueri PostgreSQL. Proses mengoptimalkan kueri tergantung pada kasus penggunaan. Mengetahui rencana kueri saat ini dapat membantu Anda mengidentifikasi dan memahami masalah apa pun dan membuat perubahan yang diperlukan. Terkadang, Anda mungkin perlu menganalisis tabel untuk menjaga statistik database tetap up to date. Pengoptimal PostgreSQL akan menggunakan statistik tersebut untuk menjalankan kueri lebih cepat. Panduan ini berfokus pada praktik terbaik untuk meningkatkan kinerja kueri PostgreSQL.

Panduan ini mengasumsikan bahwa Anda memiliki Amazon Relational Database Service (Amazon RDS) yang sudah ada untuk instance database PostgreSQL atau Amazon Aurora PostgreSQL yang kompatibel dengan PostgreSQL.

Kasus penggunaan untuk penyetelan kinerja kueri

Panduan ini mencakup lima kasus penggunaan, dengan penjelasan dan contoh:

- Kolasi
- Ketidakcocokan tipe data
- Panggilan fungsi dalam `SELECT` pernyataan
- `IN` atau `EXISTS`
- Subkueri atau Ekspresi Tabel Umum (CTE)

Setiap kasus penggunaan memberikan rincian rencana awal, bagaimana menganalisis rencana untuk mengidentifikasi masalah, dan solusi. Menerapkan kasus penggunaan ini biasanya menghasilkan waktu respons yang lebih cepat untuk kueri, pengurangan beban di server, dan efisiensi sistem yang ditingkatkan secara keseluruhan. Peningkatan tersebut dapat menghasilkan pengalaman pengguna yang lebih baik dan peningkatan keandalan sistem.

Rencana kueri EXPLY

PostgreSQL menyediakan `EXPLAIN` opsi `EXPLAIN ANALYZE` dan untuk mengembalikan rencana kueri dengan rincian tentang bagaimana kueri akan dijalankan.

Pernyataan EXPLY

`EXPLAIN`Pernyataan tersebut mengembalikan rencana kueri yang dihasilkan oleh perencana PostgreSQL untuk pernyataan yang diberikan. Rencana kueri menunjukkan hal berikut:

- Bagaimana tabel yang terlibat dalam pernyataan akan dipindai (misalnya, dengan pemindaian indeks atau pemindaian berurutan)
- Bagaimana beberapa tabel akan digabungkan (misalnya, bergabung dengan hash, gabungan gabungan, atau gabungan loop bersarang)

Memahami rencana sangat penting ketika meningkatkan kinerja kueri. Setelah Anda memahami rencana, Anda dapat fokus pada di mana kueri terlalu lama dan mengambil tindakan untuk mengurangi waktu.

Menggunakan EXPLY ANALYSIS

Dalam PostgreSQL`EXPLAIN`, hanya akan menghasilkan rencana untuk pernyataan yang diberikan. Jika Anda menambahkan `ANALYZE` kata kunci, `EXPLAIN` akan mengembalikan rencana, menjalankan kueri, dan menampilkan runtime aktual dan jumlah baris untuk setiap langkah. Ini sangat diperlukan untuk menganalisis kinerja kueri.

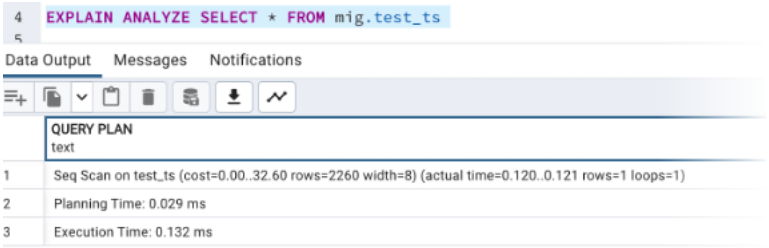
> **Important**
>
> Saat menggunakan`EXPLAIN ANALYZE`, berhati-hatilah dengan`INSERT`,`UPDATE`, dan`DELETE`.

Cara membaca rencana kueri EXPLOW

Sebuah rencana query PostgreSQL adalah struktur pohon yang terdiri dari beberapa node. Rencana `EXPLAIN` kueri menunjukkan langkah-langkah yang digunakan mesin database untuk menjalankan kueri. Paket kueri memberikan informasi berikut:

- Jenis operasi yang dilakukan, seperti pemindaian sekuensial, pemindaian indeks, atau gabungan loop bersarang.
- Label, seperti`Seq Scan`,`Index Scan`, atau`Nested Loop`, untuk menggambarkan operasi yang sedang dilakukan.
- Nama tabel atau indeks yang sedang diproses oleh kueri.
- Kolom biaya dan baris dengan informasi tentang perkiraan biaya dalam unit komputasi yang sewenang-wenang dan jumlah baris yang diproses.
- Kondisi filter dari setiap filter diterapkan pada operasi, seperti `where` kondisi.
- Representasi visual dari langkah-langkah, dengan setiap operasi ditampilkan sebagai simpul dan panah yang menghubungkan operasi. Urutan operasi ditunjukkan dari kiri ke kanan, dengan operasi sebelumnya dimasukkan ke dalam operasi selanjutnya.

Tangkapan layar berikut menunjukkan rencana kueri untuk pemindaian berurutan.



Perkiraan biaya (`cost=0.00..32.60 rows=2260 width=8`) berarti bahwa PostgreSQL mengharapkan bahwa kueri akan membutuhkan 32,60 unit komputasi untuk mengembalikan hasil.

`0.00`Nilainya adalah biaya di mana node ini dapat mulai bekerja (dalam hal ini, waktu startup untuk kueri). `rows`Nilainya adalah perkiraan jumlah baris yang akan dikembalikan oleh pemindaian sekuensial. `width`Nilainya adalah perkiraan ukuran dalam byte dari baris yang dikembalikan.

Karena contoh ditampilkan `EXPLAIN` dengan `ANALYZE` opsi, kueri dijalankan, dan informasi waktu ditangkap. Hasilnya (`actual time=0.120..0.121 rows=1 loops=1`) berarti sebagai berikut:

- Pemindaian sekuensial dijalankan satu kali (`loops`nilainya).
- Pemindaian mengembalikan satu baris.
- Waktu sebenarnya adalah 0,12 milidetik.

Kasus penggunaan untuk menyetel kueri

Panduan ini mencakup kasus penggunaan berikut untuk menyetel kinerja kueri:

- Kolasi
- Ketidakcocokan tipe data
- Panggilan fungsi dalam `SELECT` pernyataan
- `IN` atau `EXISTS`
- Subkueri atau Ekspresi Tabel Umum (CTE)

Untuk menguji penyetelan kinerja untuk kasus penggunaan kinerja kueri ini, gunakan database yang ada dan contoh data yang disediakan oleh panduan ini. Contoh menggunakan data untuk maskapai XX fiktif. Untuk menyiapkan contoh data, jalankan kode contoh berikut:

```
--Creating required tables along with data.

--Creating user and schema
create user perf_user;
create schema perf_user AUTHORIZATION perf_user;
set search_path to perf_user;


--Table1:

CREATE TABLE IF NOT EXISTS perf_user.rnr_expiry_date
(
    airline_iata_code character(2) COLLATE pg_catalog."default",
    pnr_number character varying(15) COLLATE pg_catalog."default" NOT NULL,
    calculated_pnr_expiry_date timestamp(0) without time zone,
    row_num bigint,
    arc_expiry_date timestamp(0) without time zone,
    status character varying(10) COLLATE pg_catalog."default"
);

insert into perf_user.rnr_expiry_date
select 'XX' , upper(substring(concat(md5(random()::text), md5(random()::text)), 0,
 7)),'2023-01-01 00:00:00',generate_series(1,100000), '2023-02-02 00:00:00' ,null;
```

```
CREATE INDEX rnr_expiry_date_idx1 ON perf_user.rnr_expiry_date (row_num ASC NULLS
 LAST);

CREATE INDEX rnr_expiry_date_idx2  ON perf_user.rnr_expiry_date (airline_iata_code
 COLLATE pg_catalog."default" ASC NULLS LAST, pnr_number COLLATE pg_catalog."default"
 ASC NULLS LAST);

CREATE INDEX rnr_expiry_date_idx3  ON perf_user.rnr_expiry_date (pnr_number ASC NULLS
 LAST);

vacuum analyze perf_user.rnr_expiry_date;

---------------
--Table2:

CREATE TABLE IF NOT EXISTS perf_user.rnr_segment_pax
(
    airline_iata_code character varying(6) COLLATE pg_catalog."default" NOT NULL,
    pnr_number character varying(15) COLLATE pg_catalog."default" NOT NULL,
    segment_pax_id numeric(25,0) NOT NULL,
    oandd_id numeric(25,0) NOT NULL,
    segment_id numeric(25,0) NOT NULL,
    cabin_class character varying(15) COLLATE pg_catalog."default",
    pax_id numeric(25,0) NOT NULL,
    ticket_number character varying(25) COLLATE pg_catalog."default",
    ticket_type character varying(10) COLLATE pg_catalog."default",
    archive_status smallint NOT NULL DEFAULT (0)::smallint,
    certificate_number character varying(100) COLLATE pg_catalog."default",
    loyalty_number character varying(25) COLLATE pg_catalog."default",
    arc_expiry_date timestamp(0) without time zone,
    CONSTRAINT rnr_segment_pax_pk PRIMARY KEY (airline_iata_code, pnr_number,
 segment_id, pax_id),
    CONSTRAINT rnr_segment_pax_ck1 CHECK (ticket_type::text = ANY (ARRAY['E'::character
 varying::text, 'A'::character varying::text, 'C'::character varying::text,
 'M'::character varying::text, 'I'::character varying::text]))
);

insert into perf_user.rnr_segment_pax (airline_iata_code, pnr_number, segment_pax_id,
 oandd_id, segment_id, pax_id, ticket_type, arc_expiry_date )
select 'XX',upper(substring(concat(md5(random()::text), md5(random()::text)), 0, 7)),
generate_series(1,10000000),generate_series(1,10000000),
generate_series(1,10000000),generate_series(1,10000000),'A','2023-01-01 00:00:00';
```

```
insert into perf_user.rnr_segment_pax (airline_iata_code, pnr_number, segment_pax_id,
 oandd_id, segment_id, pax_id, ticket_type, arc_expiry_date )
select 'XX',upper(substring(concat(md5(random()::text), md5(random()::text)), 0, 7)),
generate_series(10000001,20000000),generate_series(10000001,20000000),
generate_series(10000001,20000000),generate_series(10000001,20000000),'I','2023-01-01
 00:00:00';

insert into perf_user.rnr_segment_pax (airline_iata_code, pnr_number, segment_pax_id,
 oandd_id, segment_id, pax_id, ticket_type, arc_expiry_date)
select 'XX',upper(substring(concat(md5(random()::text), md5(random()::text)), 0, 7)),
generate_series(20000001,30000000),generate_series(20000001,30000000),
generate_series(20000001,30000000),generate_series(20000001,30000000),'E','2023-01-01
 00:00:00';

insert into perf_user.rnr_segment_pax (airline_iata_code, pnr_number, segment_pax_id,
 oandd_id, segment_id, pax_id, ticket_type, arc_expiry_date)
select 'XX',upper(substring(concat(md5(random()::text), md5(random()::text)), 0, 7)),
generate_series(30000001,40000000),generate_series(30000001,40000000),
generate_series(30000001,40000000),generate_series(30000001,40000000),'M','2023-01-01
 00:00:00';

CREATE INDEX rnr_segment_pax_idx1
    ON perf_user.rnr_segment_pax USING btree
    (loyalty_number COLLATE pg_catalog."default" ASC NULLS LAST, airline_iata_code
 COLLATE pg_catalog."default" ASC NULLS LAST, arc_expiry_date ASC NULLS LAST);

CREATE INDEX IF NOT EXISTS rnr_segment_pax_pn_idx1
    ON perf_user.rnr_segment_pax USING btree
    (pnr_number COLLATE pg_catalog."default" ASC NULLS LAST);

CREATE INDEX IF NOT EXISTS rnr_segment_pax_seq_idx1
    ON perf_user.rnr_segment_pax USING btree
    (segment_id ASC NULLS LAST);

vacuum analyze perf_user.rnr_segment_pax;



-----------------------------------------

--Table3:

CREATE TABLE IF NOT EXISTS perf_user.rnr_segment
(
    airline_iata_code character varying(6) COLLATE pg_catalog."default" NOT NULL,
```

```
    pnr_number character varying(15) COLLATE pg_catalog."C" NOT NULL,
    segment_id numeric(25,0) NOT NULL,
    oandd_id numeric(25,0),
    price_id numeric(25,0),
    flight_carrier character varying(6) COLLATE pg_catalog."default" ,
    flight_number integer ,
    flight_suffix character varying(1) COLLATE pg_catalog."default" ,
    flight_date_ltc timestamp(0) without time zone ,
    airline_company_code character varying(6) COLLATE pg_catalog."default",
    bd_airport_code character varying(5) COLLATE pg_catalog."default" ,
    off_airport_code character varying(5) COLLATE pg_catalog."default" ,
    segment_status character varying(50) COLLATE pg_catalog."default" ,
    flight_status character varying(30) COLLATE pg_catalog."default",
    flight_type character varying(15) COLLATE pg_catalog."default",
    cabin_class character varying(15) COLLATE pg_catalog."default",
    arc_expiry_date timestamp(0) without time zone,
    oandd_dep_date_ltc timestamp(0) without time zone,
    added_time timestamp(6) without time zone,
    dep_date_ltc timestamp(0) without time zone ,
    arr_date_utc timestamp(0) without time zone,
    dep_date_utc timestamp(0) without time zone,
    origin character varying(5) COLLATE pg_catalog."default",
    destination character varying(5) COLLATE pg_catalog."default",
    CONSTRAINT rnr_segment_pk PRIMARY KEY (pnr_number, segment_id, airline_iata_code)
);

insert into perf_user.rnr_segment (airline_iata_code, pnr_number, segment_id,
 FLIGHT_CARRIER,FLIGHT_NUMBER,FLIGHT_SUFFIX,FLIGHT_DATE_LTC)
select 'XX',
upper(substring(concat(md5(random()::text), md5(random()::text)), 0, 7)),
generate_series(1,10000000),'XX',110,'*','2023-01-01 00:00:00';

insert into perf_user.rnr_segment (airline_iata_code, pnr_number, segment_id,
 FLIGHT_CARRIER, FLIGHT_NUMBER, FLIGHT_SUFFIX, FLIGHT_DATE_LTC)
select 'XX',
upper(substring(concat(md5(random()::text), md5(random()::text)), 0, 7)),
generate_series(10000001,20000000),'XX',120,'*','2023-01-01 00:00:00';

insert into perf_user.rnr_segment (airline_iata_code, pnr_number, segment_id,
 FLIGHT_CARRIER, FLIGHT_NUMBER,FLIGHT_SUFFIX,FLIGHT_DATE_LTC)
select 'XX',
upper(substring(concat(md5(random()::text), md5(random()::text)), 0, 7)),
generate_series(20000001,30000000),'XX',130,'*','2023-01-01 00:00:00';
```

```
insert into perf_user.rnr_segment (airline_iata_code, pnr_number, segment_id,
 FLIGHT_CARRIER,FLIGHT_NUMBER,FLIGHT_SUFFIX,FLIGHT_DATE_LTC)
select 'XX',
upper(substring(concat(md5(random()::text), md5(random()::text)), 0, 7)),
generate_series(30000001,40000000),'XX',140,'*','2023-01-01 00:00:00';

insert into perf_user.rnr_segment (airline_iata_code, pnr_number, segment_id,
 FLIGHT_CARRIER, FLIGHT_NUMBER, FLIGHT_SUFFIX, FLIGHT_DATE_LTC)
select 'XX',
upper(substring(concat(md5(random()::text), md5(random()::text)), 0, 7)),
generate_series(40000001,50000000),'XX',150,'*','2023-01-01 00:00:00';

insert into perf_user.rnr_segment (airline_iata_code, pnr_number, segment_id,
 FLIGHT_CARRIER, FLIGHT_NUMBER, FLIGHT_SUFFIX, FLIGHT_DATE_LTC)
select 'XX',
upper(substring(concat(md5(random()::text), md5(random()::text)), 0, 7)),
generate_series(50000001,60000000),'XX',160,'*','2023-01-01 00:00:00';


CREATE INDEX rnr_segment_idx1  ON perf_user.rnr_segment USING btree
    (flight_date_ltc ASC NULLS LAST, bd_airport_code COLLATE pg_catalog."default"
 ASC NULLS LAST, off_airport_code COLLATE pg_catalog."default" ASC NULLS LAST,
 flight_number ASC NULLS LAST, flight_carrier COLLATE pg_catalog."default" ASC NULLS
 LAST, flight_suffix COLLATE pg_catalog."default" ASC NULLS LAST, airline_iata_code
 COLLATE pg_catalog."default" ASC NULLS LAST, arc_expiry_date ASC NULLS LAST);

CREATE INDEX rnr_segment_idx2
    ON perf_user.rnr_segment USING btree
    (dep_date_ltc ASC NULLS LAST, flight_number ASC NULLS LAST, bd_airport_code COLLATE
 pg_catalog."default" ASC NULLS LAST, off_airport_code COLLATE pg_catalog."default" ASC
 NULLS LAST, flight_carrier COLLATE pg_catalog."default" ASC NULLS LAST, flight_suffix
 COLLATE pg_catalog."default" ASC NULLS LAST, arc_expiry_date ASC NULLS LAST);

CREATE INDEX rnr_segment_idx3
    ON perf_user.rnr_segment USING btree
    (pnr_number COLLATE pg_catalog."default" ASC NULLS LAST, arr_date_utc ASC NULLS
 LAST, airline_iata_code COLLATE pg_catalog."default" ASC NULLS LAST, arc_expiry_date
 ASC NULLS LAST);

CREATE INDEX rnr_segment_idx4
    ON perf_user.rnr_segment USING btree
    (dep_date_utc ASC NULLS LAST, added_time ASC NULLS LAST, airline_iata_code COLLATE
 pg_catalog."default" ASC NULLS LAST, arc_expiry_date ASC NULLS LAST);
```

```
CREATE INDEX rnr_segment_idx5
    ON perf_user.rnr_segment USING btree
    (origin COLLATE pg_catalog."default" ASC NULLS LAST, destination COLLATE
 pg_catalog."default" ASC NULLS LAST, oandd_dep_date_ltc ASC NULLS LAST,
 airline_iata_code COLLATE pg_catalog."default" ASC NULLS LAST, arc_expiry_date ASC
 NULLS LAST);

CREATE INDEX rnr_segment_idx6
    ON perf_user.rnr_segment USING btree
    (pnr_number COLLATE pg_catalog."default" ASC NULLS LAST, oandd_id ASC NULLS LAST,
 segment_id ASC NULLS LAST, airline_iata_code COLLATE pg_catalog."default" ASC NULLS
 LAST, arc_expiry_date ASC NULLS LAST);


vacuum analyze perf_user.rnr_segment;



-------------------------------------

--Table4:

CREATE TABLE IF NOT EXISTS perf_user.rnr_seat_numbers
(
    airline_iata_code character varying(6) COLLATE pg_catalog."default" NOT NULL,
    pnr_number character varying(15) COLLATE pg_catalog."default" NOT NULL,
    segment_id numeric(25,0) NOT NULL,
    pax_id numeric(25,0) NOT NULL,
    seat_id numeric(25,0) NOT NULL,
    bd_airport_code character varying(5) COLLATE pg_catalog."default",
    off_airport_code character varying(5) COLLATE pg_catalog."default",
    seat_number character varying(5) COLLATE pg_catalog."default",
    seat_status character varying(20) COLLATE pg_catalog."default",
    ssr_id character varying(100) COLLATE pg_catalog."default",
    archive_status smallint DEFAULT (0)::smallint,
    seat_alloc_id numeric(25,0),
    archive_date timestamp(0) without time zone,
    seat_attribute_code character varying(201) COLLATE pg_catalog."default",
    channel_code character varying(20) COLLATE pg_catalog."default",
    arc_expiry_date timestamp(0) without time zone,
    CONSTRAINT rnr_seat_numbers_pk PRIMARY KEY (pnr_number, segment_id, pax_id,
 seat_id, airline_iata_code)
);
```

```
insert into perf_user.rnr_seat_numbers (pnr_number, segment_id, pax_id, seat_id,
 airline_iata_code)
select upper(substring(concat(md5(random()::text), md5(random()::text)), 0, 7)),
generate_series(1,10000000),generate_series(1,10000000),generate_series(1,10000000),'XX';

insert into perf_user.rnr_seat_numbers (pnr_number, segment_id, pax_id, seat_id,
 airline_iata_code)
select upper(substring(concat(md5(random()::text), md5(random()::text)), 0, 7)),
generate_series(10000001,20000000),generate_series(10000001,20000000),generate_series(10000001,

insert into perf_user.rnr_seat_numbers (pnr_number, segment_id, pax_id, seat_id,
 airline_iata_code)
select upper(substring(concat(md5(random()::text), md5(random()::text)), 0, 7)),
generate_series(20000001,30000000),generate_series(20000001,30000000),generate_series(20000001,

insert into perf_user.rnr_seat_numbers (pnr_number, segment_id, pax_id, seat_id,
 airline_iata_code)
select upper(substring(concat(md5(random()::text), md5(random()::text)), 0, 7)),
generate_series(30000001,40000000),generate_series(30000001,40000000),generate_series(30000001,

vacuum Analyze perf_user.rnr_seat_numbers;

--Table5:
CREATE TABLE IF NOT EXISTS perf_user.test_veh
(
    test_veh_id bigint NOT NULL,
    oiltype_id bigint,
    vehicle_id character varying(50) COLLATE pg_catalog."default",
    serviceprogram_id character varying(100) COLLATE pg_catalog."default",
    startdate timestamp without time zone,
    enddate timestamp without time zone,
    last_update_dt timestamp without time zone,
    CONSTRAINT test_veh_pkey PRIMARY KEY (test_veh_id),
    CONSTRAINT test_veh_oiltype_id_fkey FOREIGN KEY (oiltype_id)
        REFERENCES perf_user.oiltype (oiltype_id) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION,
    CONSTRAINT test_veh_oiltype_id_fkey1 FOREIGN KEY (oiltype_id)
        REFERENCES perf_user.oiltype (oiltype_id) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
);

CREATE INDEX IF NOT EXISTS test_veh_enddate_ind
```

```
    ON perf_user.test_veh USING btree
    (enddate ASC NULLS LAST);

CREATE INDEX IF NOT EXISTS test_veh_oiltype_id_ind
    ON perf_user.test_veh USING btree
    (oiltype_id ASC NULLS LAST);


--Table6:
CREATE TABLE IF NOT EXISTS perf_user.oiltype
(
    oiltype_id bigint NOT NULL,
    descr character varying(50) COLLATE pg_catalog."default",
    CONSTRAINT oiltype_pkey PRIMARY KEY (oiltype_id)
);

CREATE INDEX IF NOT EXISTS oiltype_oiltyp_in
    ON perf_user.oiltype USING btree
    (oiltype_id ASC NULLS LAST);



--Table7:
CREATE TABLE IF NOT EXISTS perf_user.serviceprogram
(
    serial bigint NOT NULL,
    serviceprogram_id character varying(50) COLLATE pg_catalog."default",
    progname character varying(150) COLLATE pg_catalog."default",
    CONSTRAINT serviceprogram_pkey PRIMARY KEY (serial)
);

CREATE INDEX IF NOT EXISTS progname_id_ind
    ON perf_user.serviceprogram USING btree
    (progname COLLATE pg_catalog."default" ASC NULLS LAST);

CREATE INDEX IF NOT EXISTS serviceprogram_id_ind
    ON perf_user.serviceprogram USING btree
    (serviceprogram_id COLLATE pg_catalog."default" ASC NULLS LAST);

--Table8:
CREATE TABLE IF NOT EXISTS perf_user.vehicleservicehistory
(
    v_id bigint NOT NULL,
    test_veh_id bigint,
```

```
    desc_1 character varying(50) COLLATE pg_catalog."default",
    start_date timestamp without time zone,
    end_date timestamp without time zone,
    CONSTRAINT vehicleservicehistory_pkey PRIMARY KEY (v_id)
);

CREATE INDEX IF NOT EXISTS veh_end_date_id_ind
    ON perf_user.vehicleservicehistory USING btree
    (end_date ASC NULLS LAST);

CREATE INDEX IF NOT EXISTS veh_ser_ind
    ON perf_user.vehicleservicehistory USING btree
    (test_veh_id ASC NULLS LAST);

CREATE INDEX IF NOT EXISTS vehicleservicehistory_v_id_ind
    ON perf_user.vehicleservicehistory USING btree
    (test_veh_id ASC NULLS LAST);


--Function creation
CREATE OR REPLACE FUNCTION perf_user.return_data()
    RETURNS character varying
    LANGUAGE 'plpgsql'
    COST 100
    VOLATILE PARALLEL UNSAFE
AS $BODY$
BEGIN
return 'EE9F41' ;
END;
$BODY$;
-----------------------------------------------
CREATE TABLE IF NOT EXISTS  ITEM_DETAILS
(
    ITEMID INTEGER,
    ORDID INTEGER,
    ITEMNAME CHARACTER VARYING(200)
);

CREATE TABLE IF NOT EXISTS  ORDER_DETAILS
(
    ORDID INTEGER,
    ORDNAME CHARACTER VARYING(200),
    ORDEREDPLACE CHARACTER VARYING(55)
);
```

```
CREATE TABLE IF NOT EXISTS  PAYMENT_DETAILS
(
    PAYID INTEGER,
    ORDID INTEGER,
    PAYPLACE CHARACTER VARYING(55)
);
```

Kasus penggunaan 1 - Koleksi

Dalam database, pemeriksaan adalah seperangkat aturan untuk menentukan bagaimana data diurutkan dan dibandingkan. Sebuah pemeriksaan biasanya diterapkan pada bagaimana data teks diurutkan dalam bahasa yang berbeda untuk pengindeksan untuk membuat perbandingan antara nilai teks. Bahasa yang berbeda memiliki set dan urutan karakter yang berbeda. Dengan pemeriksaan, Anda dapat mengurutkan data karakter untuk bahasa tertentu dengan menggunakan aturan yang menentukan urutan karakter yang benar. Anda juga dapat menentukan yang berikut:

- Sensitivitas kasus
- Tanda aksen
- Jenis karakter Kana
- Penggunaan simbol atau tanda baca
- Lebar karakter
- Penyortiran kata

Mungkin ada dampak kinerja jika kolom gabungan menggunakan pemeriksaan yang berbeda. Contoh query berikut menggunakan tiga tabel, dengan pemeriksaan yang berbeda untuk kolom join.

Nama tabel	Nama kolom
`rnr_segment`	`pnr_number character varying(15) COLLATE pg_catalog."C" NOT NULL`
`rnr_segment_pax`	`pnr_number character varying(15) COLLATE pg_catalog."default" NOT NULL`

rnr_seat_numbers	pnr_number character varying(15) COLLATE pg_catalog."default" NOT NULL

```
EXPLAIN ANALYZE SELECT
A.PNR_NUMBER,
A.PAX_ID,
A.SEGMENT_ID,
B.OANDD_ID,
C.SEAT_ID,
C.BD_AIRPORT_CODE,
C.OFF_AIRPORT_CODE,
C.SEAT_NUMBER ,
B.CABIN_CLASS ,
A.SEGMENT_PAX_ID,
C.SEAT_ALLOC_ID,
C.SSR_ID,
C.SEAT_ATTRIBUTE_CODE
from
RNR_SEGMENT_PAX A,
RNR_SEGMENT B,
RNR_SEAT_NUMBERS C
where
B.AIRLINE_IATA_CODE = 'XX'
and B.FLIGHT_CARRIER = 'XX'
and B.FLIGHT_NUMBER = 140
and B.FLIGHT_SUFFIX ='*'
and B.FLIGHT_DATE_LTC = TO_DATE('01-JAN-2023', 'DD-MON-YYYY')
and A.AIRLINE_IATA_CODE = B.AIRLINE_IATA_CODE
and A.PNR_NUMBER = B.PNR_NUMBER
and A.SEGMENT_ID = B.SEGMENT_ID
and C.AIRLINE_IATA_CODE = B.AIRLINE_IATA_CODE
and C.PNR_NUMBER = B.PNR_NUMBER
and C.SEGMENT_ID = B.SEGMENT_ID
and A.PAX_ID = C.PAX_ID
and B.PNR_NUMBER in ('9F1588','E37DE0','04E82B','813D11','BFF10F');
```

Rencana kueri untuk kueri sebelumnya menggunakan pemindaian urutan pada `rnr_seat_numbers` tabel meskipun tabel tersebut memiliki indeks yang tepat pada kolom yang digabungkan. Perencana

tidak menggunakan pemindaian indeks karena kolom yang digabungkan ini menggunakan kumpulan yang berbeda:

```
Nested Loop  (cost=1112.14..927363.51 rows=1 width=833) (actual time=5395.367..5397.253
 rows=0 loops=1)
  Join Filter: (((b.pnr_number)::text = (a.pnr_number)::text) AND (b.segment_id =
 a.segment_id))
  ->  Gather  (cost=1111.58..670766.48 rows=1 width=843) (actual
 time=5395.367..5397.251 rows=0 loops=1)
        Workers Planned: 2
        Workers Launched: 2
        ->  Hash Join  (cost=111.58..669766.38 rows=1 width=843) (actual
 time=5388.992..5388.993 rows=0 loops=3)
              Hash Cond: (((c.pnr_number)::text = (b.pnr_number)::text) AND
 (c.segment_id = b.segment_id))
              ->  Parallel Seq Scan on rnr_seat_numbers c  (cost=0.00..582154.96
 rows=16666637 width=760) (actual time=0.008..2963.019 rows=13333333 loops=3)
                    Filter: ((airline_iata_code)::text = 'XX'::text)
              ->  Hash  (cost=111.52..111.52 rows=4 width=86) (actual time=0.121..0.121
 rows=2 loops=3)
                    Buckets: 1024  Batches: 1  Memory Usage: 9kB
                    ->  Index Scan using rnr_segment_pk on rnr_segment b
 (cost=0.56..111.52 rows=4 width=86) (actual time=0.082..0.116 rows=2 loops=3)
                          Index Cond: (((pnr_number)::text = ANY
 ('{9F1588,E37DE0,04E82B,813D11,BFF10F}'::text[])) AND ((airline_iata_code)::text =
 'XX'::text))
                          Filter: (((flight_carrier)::text = 'XX'::text) AND
 (flight_number = 140) AND ((flight_suffix)::text = '*'::text) AND (flight_date_ltc =
 to_date('01-JAN-2023'::text, 'DD-MON-YYYY'::text)))
                          Rows Removed by Filter: 20
  ->  Index Scan using rnr_segment_pax_pk on rnr_segment_pax a  (cost=0.56..256597.02
 rows=1 width=28) (never executed)
        Index Cond: (((airline_iata_code)::text = 'XX'::text) AND (segment_id =
 c.segment_id) AND (pax_id = c.pax_id))
        Filter: ((c.pnr_number)::text = (pnr_number)::text)
Planning Time: 0.982 ms
Execution Time: 5397.314 ms
```

Untuk mengubah pemeriksaan kolom tabel dari `"C"` bahasa ke pemeriksaan default yang disediakan oleh PostgreSQL, jalankan pernyataan berikut, lalu analisis tabel: `alter`

```
alter table rnr_segment alter column pnr_number type character varying(15) COLLATE
 pg_catalog."default";
```

```
Analyze rnr_segment;
```

Paket kueri sekarang menggunakan pemindaian indeks, dan runtime berkurang.

```
Nested Loop  (cost=1.69..146.63 rows=1 width=833) (actual time=0.155..0.155 rows=0
 loops=1)
  ->  Nested Loop  (cost=1.13..145.89 rows=1 width=111) (actual time=0.154..0.155
 rows=0 loops=1)
        ->  Index Scan using rnr_segment_pk on rnr_segment b  (cost=0.56..111.51 rows=4
 width=86) (actual time=0.048..0.097 rows=2 loops=1)
              Index Cond: (((pnr_number)::text = ANY
 ('{9F1588,E37DE0,04E82B,813D11,BFF10F}'::text[])) AND ((airline_iata_code)::text =
 'XX'::text))
              Filter: (((flight_carrier)::text = 'XX'::text) AND (flight_number =
 140) AND ((flight_suffix)::text = '*'::text) AND (flight_date_ltc = to_date('01-
JAN-2023'::text, 'DD-MON-YYYY'::text)))
              Rows Removed by Filter: 20
        ->  Index Scan using rnr_segment_pax_pk on rnr_segment_pax a  (cost=0.56..8.58
 rows=1 width=28) (actual time=0.027..0.027 rows=0 loops=2)
              Index Cond: (((airline_iata_code)::text = 'XX'::text) AND
 ((pnr_number)::text = (b.pnr_number)::text) AND (segment_id = b.segment_id))
  ->  Index Scan using rnr_seat_numbers_pk on rnr_seat_numbers c  (cost=0.56..0.72
 rows=1 width=760) (never executed)
        Index Cond: (((pnr_number)::text = (a.pnr_number)::text) AND (segment_id =
 a.segment_id) AND (pax_id = a.pax_id) AND ((airline_iata_code)::text = 'XX'::text))
Planning Time: 1.432 ms
Execution Time: 0.207 ms
```

Kasus penggunaan 2 — Ketidakcocokan tipe data

Memilih tipe data yang tepat berdasarkan data membantu memberikan keseimbangan optimal antara ukuran penyimpanan dan kinerja.

Contoh query berikut menggunakan `pnr_number` kolom untuk menggabungkan dua tabel. `pnr_number`Kolom memiliki tipe data yang berbeda dalam tabel yang berbeda.

Nama tabel	Nama kolom dan tipe data
`perf_user.rnr_segment_pax`	`pnr_number character varying(6)`

```
perf_user.rnr_expiry_date            pnr_number character(2)
```

```
EXPLAIN ANALYZE UPDATE perf_user.RNR_SEGMENT_PAX x SET ARC_EXPIRY_DATE =
 y.ARC_EXPIRY_DATE
       FROM (SELECT AIRLINE_IATA_CODE, PNR_NUMBER, ARC_EXPIRY_DATE, 0+row_num ROW_NUM
               FROM perf_user.RNR_EXPIRY_DATE
              WHERE airline_iata_code = 'XX'
                AND row_num BETWEEN (1*5000)+0 AND (1+1)*5000) y
         WHERE  x.airline_iata_code = y.airline_iata_code
             AND  x.PNR_NUMBER =y.PNR_NUMBER;

------------------------------------------------------------------------------------------

Update on rnr_segment_pax x  (cost=290.97..1104986.32 rows=15515 width=460) (actual
 time=14574.118..14574.120 rows=0 loops=1)
  ->  Hash Join  (cost=290.97..1104986.32 rows=15515 width=460) (actual
 time=16.967..14101.983 rows=11953 loops=1)
       Hash Cond: ((x.pnr_number)::text = (rnr_expiry_date.pnr_number)::text)
        ->  Seq Scan on rnr_segment_pax x  (cost=0.00..954539.00 rows=40000320
 width=446) (actual time=0.011..9702.989 rows=40000000 loops=1)
              Filter: ((airline_iata_code)::bpchar = 'XX'::bpchar)
        ->  Hash  (cost=225.37..225.37 rows=5248 width=24) (actual time=16.540..16.541
 rows=5001 loops=1)
              Buckets: 8192  Batches: 1  Memory Usage: 338kB
              ->  Index Scan using rnr_expiry_date_idx1 on rnr_expiry_date
 (cost=0.29..225.37 rows=5248 width=24) (actual time=3.102..15.331 rows=5001 loops=1)
                    Index Cond: ((row_num >= 5000) AND (row_num <= 10000))
                    Filter: (airline_iata_code = 'XX'::bpchar)
Planning Time: 4.445 ms
Execution Time: 14574.322 ms
```

Saat Anda menjalankan`EXPLAIN ANALYZE`, perencana menggunakan pemindaian urutan `rnr_segment_pax` alih-alih pemindaian indeks meskipun kolom yang digunakan dalam gabungan memiliki indeks. Perencana tidak menggunakan pemindaian indeks karena kolom yang digunakan dalam gabungan memiliki panjang yang berbeda.

Ubah kolom tabel agar tipe data tetap sama untuk kedua tabel yang terlibat dalam kondisi gabungan, lalu analisis tabel:

```
alter table perf_user.rnr_expiry_date alter column airline_iata_code type character
 varying(6) ;
```

```
analyze perf_user.rnr_expiry_date;
```

Sekarang tabel memiliki panjang yang sama pada kedua kolom yang digunakan dalam kondisi bergabung.

Jalankan lagi `EXPLAIN ANALYZE`. Perencana melakukan pemindaian indeks, yang meningkatkan kinerja kueri secara signifikan.

```
Update on rnr_segment_pax x  (cost=0.86..59733.09 rows=14637 width=460) (actual
 time=416.653..416.654 rows=0 loops=1)
  ->  Nested Loop  (cost=0.86..59733.09 rows=14637 width=460) (actual
 time=0.103..91.106 rows=11953 loops=1)
       ->  Index Scan using rnr_expiry_date_idx1 on rnr_expiry_date
 (cost=0.29..212.69 rows=4951 width=24) (actual time=0.025..3.023 rows=5001 loops=1)
             Index Cond: ((row_num >= 5000) AND (row_num <= 10000))
             Filter: ((airline_iata_code)::text = 'XX'::text)
       ->  Index Scan using rnr_segment_pax_pk on rnr_segment_pax x  (cost=0.56..11.99
 rows=3 width=446) (actual time=0.014..0.016 rows=2 loops=5001)
             Index Cond: (((airline_iata_code)::text = 'XX'::text) AND
 ((pnr_number)::text = (rnr_expiry_date.pnr_number)::text))
Planning Time: 0.310 ms
Execution Time: 416.696 ms
```

Gunakan kasus 3 - Panggilan fungsi dalam pernyataan SELECT

Memanggil fungsi dalam `where` klausa dapat mengurangi kinerja kueri saat fungsi tersebut `VOLATILE` dan Anda tidak menggunakan `select` kata kunci saat memanggil fungsi:

```
Select * from tab_name where FieldName = FunctionName(parameters);
```

Pemindaian indeks berjalan jika `select` pernyataan digunakan saat memanggil fungsi:

```
Select * from tab_name where FieldName = ( select FunctionName(parameters) );
```

`pnr_number`Bidang memiliki indeks dalam `rnr_expiry_date` tabel. Indeks digunakan saat membandingkan nilai dalam `where` klausa.

```
explain analyze select * from perf_user.rnr_expiry_date where pnr_number= 'EE9F41';
```

```
"Index Scan using rnr_expiry_date_idx3 on rnr_expiry_date  (cost=0.29..8.31 rows=1
 width=72) (actual time=0.020..0.021 rows=1 loops=1)"
"  Index Cond: ((pnr_number)::text = 'EE9F41'::text)"
"Planning Time: 0.063 ms"
"Execution Time: 0.038 ms"
```

Pemindaian sekuensial dilakukan ketika fungsi dipanggil tanpa `select` kata kunci bahkan ketika indeks tersedia di lapangan.

```
explain analyze select * from perf_user.rnr_expiry_date where pnr_number=
 perf_user.return_data();

"Seq Scan on rnr_expiry_date  (cost=0.00..27084.00 rows=1 width=72) (actual
 time=0.112..135.917 rows=1 loops=1)"
"  Filter: ((pnr_number)::text = (perf_user.return_data())::text)"
"  Rows Removed by Filter: 99999"
"Planning Time: 0.053 ms"
"Execution Time: 136.803 ms"
```

Pemindaian indeks dilakukan ketika fungsi dipanggil dengan `select` kata kunci.

```
explain analyze select * from perf_user.rnr_expiry_date where pnr_number= (select
 perf_user.return_data() );

"Index Scan using rnr_expiry_date_idx3 on rnr_expiry_date  (cost=0.55..8.57 rows=1
 width=72) (actual time=0.058..0.061 rows=1 loops=1)"
"  Index Cond: ((pnr_number)::text = ($0)::text)"
"  InitPlan 1 (returns $0)"
"    ->  Result  (cost=0.00..0.26 rows=1 width=32) (actual time=0.021..0.022 rows=1
 loops=1)"
"Planning Time: 0.147 ms"
"Execution Time: 0.111 ms"
```

Kasus penggunaan 4 — IN atau EXISTS

Jika kueri memiliki `IN` atau `NOT IN` operator, sebaiknya periksa rencana kueri untuk mengonfirmasi bahwa indeks yang tepat sedang digunakan. Jika indeks yang tepat tidak digunakan dan kinerja kueri membutuhkan waktu lebih lama dari yang diharapkan, coba tulis ulang kueri menggunakan `NOT EXISTS` kondisi `EXISTS` atau.

Perhatikan contoh berikut, yang menggunakanNOT IN:

```
EXPLAIN ANALYZE SELECT
  TEST_VEH.TEST_VEH_ID,
  TEST_VEH.VEHICLE_ID,
  TEST_VEH.SERVICEPROGRAM_ID,
  TEST_VEH.STARTDATE,
  TEST_VEH.ENDDATE,
  TEST_VEH.OILTYPE_ID
FROM PERF_USER.TEST_VEH TEST_VEH
JOIN  PERF_USER.OILTYPE OT ON  OT.OILTYPE_ID =TEST_VEH.OILTYPE_ID
JOIN  PERF_USER.SERVICEPROGRAM SP ON  SP.SERVICEPROGRAM_ID = TEST_VEH.SERVICEPROGRAM_ID
WHERE SP.PROGNAME  = '18FCE8FDAF365BB'
  AND OT.OILTYPE_ID  =3
  AND TEST_VEH.ENDDATE  IS NOT NULL
  AND TEST_VEH.TEST_VEH_ID NOT IN
                      (SELECT TEST_VEH_ID
                              FROM PERF_USER.VEHICLESERVICEHISTORY
                              WHERE  TEST_VEH_ID > 1
                      );
------------------------------------------------------------------------
"Nested Loop  (cost=1009.16..1188860356305.01 rows=1 width=76) (actual
 time=37299.891..37347.853 rows=0 loops=1)"
"  ->  Gather  (cost=1009.16..1188860356303.88 rows=1 width=76) (actual
 time=37299.890..37347.849 rows=0 loops=1)"
"        Workers Planned: 2"
"        Workers Launched: 2"
"        ->  Hash Join  (cost=9.16..1188860355303.78 rows=1 width=76) (actual
 time=37286.742..37286.751 rows=0 loops=3)"
"              Hash Cond: ((test_veh.serviceprogram_id)::text =
 (sp.serviceprogram_id)::text)"
"              ->  Parallel Index Scan using test_veh_oiltype_id_ind on test_veh
 (cost=0.56..1188860351273.04 rows=1072570 width=76) (actual time=37276.290..37276.292
 rows=1 loops=3)"
"                    Index Cond: (oiltype_id = 3)"
"                    Filter: ((enddate IS NOT NULL) AND (NOT (SubPlan 1)))"
"                    Rows Removed by Filter: 0"
"                    SubPlan 1"
"                      ->  Materialize  (cost=0.00..1025071.31 rows=33333332 width=8)
 (actual time=0.418..23201.432 rows=25001498 loops=4)"
"                            ->  Seq Scan on vehicleservicehistory
 (cost=0.00..728195.65 rows=33333332 width=8) (actual time=0.416..13249.975
 rows=25001498 loops=4)"
"                                  Filter: (test_veh_id > 1)"
```

```
"              ->  Hash  (cost=8.58..8.58 rows=1 width=11) (actual time=9.045..9.046
 rows=0 loops=3)"
"                    Buckets: 1024  Batches: 1  Memory Usage: 8kB"
"                    ->  Index Scan using progname_id_ind on serviceprogram sp
 (cost=0.56..8.58 rows=1 width=11) (actual time=9.043..9.044 rows=0 loops=3)"
"                          Index Cond: ((progname)::text = '18FCE8FDAF365BB'::text)"
"  ->  Seq Scan on oiltype ot  (cost=0.00..1.12 rows=1 width=8) (never executed)"
"        Filter: (oiltype_id = 3)"
"Planning Time: 37.696 ms"
"Execution Time: 37366.335 ms"
```

Kueri membutuhkan waktu lebih dari 37 detik 366 milidetik untuk mengambil 4 juta catatan.

Rencana kueri menyatakan bahwa pemindaian urutan dilakukan pada tabel yang digunakan dalam subquery`vehicleservicehistory`. Pemindaian urutan menghasilkan sejumlah besar catatan. Untuk masing-masing catatan di subquery, kueri melakukan pemindaian tabel lengkap, yang menyebabkan masalah kinerja.

Untuk menghindari pemindaian urutan pada subquery, tulis ulang subquery untuk menggunakan subquery yang berkorelasi dengan. `NOT EXISTS` Subquery yang berkorelasi akan menggunakan pemindaian indeks dan pengurangan jumlah pemindaian tabel:

```
EXPLAIN ANALYZE SELECT
  TEST_VEH.TEST_VEH_ID,
  TEST_VEH.VEHICLE_ID,
  TEST_VEH.SERVICEPROGRAM_ID,
  TEST_VEH.STARTDATE,
  TEST_VEH.ENDDATE,
  TEST_VEH.OILTYPE_ID
FROM PERF_USER.TEST_VEH TEST_VEH
JOIN  PERF_USER.OILTYPE OT ON  OT.OILTYPE_ID =TEST_VEH.OILTYPE_ID
JOIN  PERF_USER.SERVICEPROGRAM SP ON  SP.SERVICEPROGRAM_ID = TEST_VEH.SERVICEPROGRAM_ID
WHERE SP.PROGNAME  = '18FCE8FDAF365BB'
       AND OT.OILTYPE_ID  =3
       AND TEST_VEH.ENDDATE  IS NOT NULL
       AND  NOT EXISTS
                  (SELECT TEST_VEH_ID
                          FROM PERF_USER.VEHICLESERVICEHISTORY
                          WHERE
 TEST_VEH.TEST_VEH_ID=VEHICLESERVICEHISTORY.TEST_VEH_ID
                          AND TEST_VEH_ID > 1
                  );
------------------------------------------------------------------------
```

```
"Nested Loop Anti Join  (cost=1009.03..936146.10 rows=1 width=76) (actual
 time=12.693..12.810 rows=0 loops=1)"
"  ->  Nested Loop  (cost=1008.59..936141.78 rows=1 width=76) (actual
 time=12.692..12.809 rows=0 loops=1)"
"        ->  Gather  (cost=1008.59..936140.64 rows=1 width=76) (actual
 time=12.691..12.807 rows=0 loops=1)"
"              Workers Planned: 2"
"              Workers Launched: 2"
"              ->  Hash Join  (cost=8.59..935140.54 rows=1 width=76) (actual
 time=0.773..0.774 rows=0 loops=3)"
"                    Hash Cond: ((test_veh.serviceprogram_id)::text =
 (sp.serviceprogram_id)::text)"
"                    ->  Parallel Seq Scan on test_veh  (cost=0.00..927087.67
 rows=2145139 width=76) (actual time=0.672..0.672 rows=1 loops=3)"
"                          Filter: ((enddate IS NOT NULL) AND (oiltype_id = 3))"
"                          Rows Removed by Filter: 7"
"                    ->  Hash  (cost=8.58..8.58 rows=1 width=11) (actual
 time=0.040..0.040 rows=0 loops=3)"
"                          Buckets: 1024  Batches: 1  Memory Usage: 8kB"
"                          ->  Index Scan using progname_id_ind on serviceprogram sp
 (cost=0.56..8.58 rows=1 width=11) (actual time=0.039..0.040 rows=0 loops=3)"
"                                Index Cond: ((progname)::text =
 '18FCE8FDAF365BB'::text)"
"        ->  Seq Scan on oiltype ot  (cost=0.00..1.12 rows=1 width=8) (never executed)"
"              Filter: (oiltype_id = 3)"
"  ->  Index Only Scan using veh_ser_ind on vehicleservicehistory  (cost=0.44..4.32
 rows=1 width=8) (never executed)"
"        Index Cond: ((test_veh_id = test_veh.test_veh_id) AND (test_veh_id > 1))"
"        Heap Fetches: 0"
"Planning Time: 11.115 ms"
"Execution Time: 12.871 ms"
```

Setelah modifikasi, kueri membutuhkan waktu kurang dari 13 ms untuk memproses 4 juta catatan

Menurut rencana kueri dari kueri yang dimodifikasi, tabel `vehicleservicehistory` dapat memiliki pemindaian indeks. Menggunakan pemindaian indeks mengurangi biaya dan jumlah baris yang terpengaruh. Dengan cara ini, Anda dapat mengurangi runtime kueri dan meningkatkan kinerjanya.

Kasus penggunaan 5 - Subquery atau CTE

Common Table Expressions (CTE) membantu memecah kueri besar menjadi kueri yang lebih kecil. Ini membuat seluruh kueri lebih mudah dipelihara.

Gabungan subquery digantikan oleh gabungan CTE, yang lebih mudah dibaca karena kueri diberi nama dan dipisahkan di dalam bagian CTE. Ini sangat membantu ketika ukuran kueri bertambah dan kueri menjadi lebih sulit dipertahankan. Selain itu, hasil CTE di PostgreSQL terwujud. Jika Anda memanggil CTE di beberapa tempat, definisi kueri yang sebenarnya akan dijalankan hanya satu kali. Hasilnya akan disimpan dalam memori. Anda dapat menggunakan ini untuk logika kompleks apa pun yang harus digunakan di beberapa tempat dalam kueri yang sama. Masukkan logika itu ke dalam CTE, dan panggil CTE beberapa kali.

Misalnya, pelanggan menggunakan kueri aplikasi inline dengan banyak subquery dalam kueri. Subquery disaring oleh nilai parameter masukan yang dikirim dari aplikasi.

```
EXPLAIN  ANALYZE
SELECT * FROM
ORDER_DETAILS A
WHERE A.ORDID IN  (SELECT ORDID FROM PAYMENT_DETAILS)
AND A.ORDID IN  (SELECT ORDID FROM   ITEM_DETAILS  )
AND  A.ORDID = 1000000;
```

```
"Nested Loop Semi Join  (cost=3000.00..194258.21 rows=5 width=74) (actual
 time=201.605..747.945 rows=5 loops=1)"
"  ->  Nested Loop Semi Join  (cost=2000.00..135040.47 rows=5 width=74) (actual
 time=146.016..666.779 rows=5 loops=1)"
"        ->  Gather  (cost=1000.00..78580.31 rows=5 width=74) (actual
 time=58.893..463.570 rows=5 loops=1)"
"              Workers Planned: 2"
"              Workers Launched: 2"
"              ->  Parallel Seq Scan on order_details a  (cost=0.00..77579.81 rows=2
 width=74) (actual time=165.627..549.702 rows=2 loops=3)"
"                    Filter: (ordid = 1000000)"
"                    Rows Removed by Filter: 1666665"
"        ->  Materialize  (cost=1000.00..56460.07 rows=3 width=4) (actual
 time=17.424..40.638 rows=1 loops=5)"
"              ->  Gather  (cost=1000.00..56460.06 rows=3 width=4) (actual
 time=87.113..203.178 rows=1 loops=1)"
"                    Workers Planned: 2"
"                    Workers Launched: 2"
"                    ->  Parallel Seq Scan on payment_details  (cost=0.00..55459.76
 rows=1 width=4) (actual time=174.431..423.792 rows=1 loops=3)"
"                          Filter: (ordid = 1000000)"
"                          Rows Removed by Filter: 1333002"
"  ->  Materialize  (cost=1000.00..59217.64 rows=4 width=4) (actual time=11.117..16.231
 rows=1 loops=5)"
```

```
"        ->  Gather  (cost=1000.00..59217.62 rows=4 width=4) (actual
 time=55.581..81.148 rows=1 loops=1)"
"              Workers Planned: 2"
"              Workers Launched: 2"
"              ->  Parallel Seq Scan on item_details  (cost=0.00..58217.22 rows=2
 width=4) (actual time=287.030..411.004 rows=1 loops=3)"
"                    Filter: (ordid = 1000000)"
"                    Rows Removed by Filter: 1333080"
"Planning Time: 0.266 ms"
"Execution Time: 747.986 ms"
```

Setelah memodifikasi subkueri dengan menggunakan CTE dan menambahkan filter sehingga hanya kumpulan baris yang diperlukan yang diambil, kinerja kueri meningkat.

```
EXPLAIN  ANALYZE
WITH PAYMENT AS
 (
  SELECT * FROM PAYMENT_DETAILS WHERE  ORDID = 1000000
 ),
ITEM AS
(SELECT * FROM ITEM_DETAILS  WHERE  ORDID = 1000000)
SELECT * FROM
ORDER_DETAILS A JOIN PAYMENT B
ON A.ORDID=B.ORDID
JOIN ITEM C ON B.ORDID=C.ORDID
```

```
"Nested Loop  (cost=3000.00..194258.91 rows=60 width=166) (actual time=586.410..732.918
 rows=80 loops=1)"
"  ->  Nested Loop  (cost=2000.00..115677.83 rows=12 width=92) (actual
 time=456.760..457.083 rows=16 loops=1)"
"        ->  Gather  (cost=1000.00..59217.62 rows=4 width=48) (actual
 time=153.802..154.060 rows=4 loops=1)"
"              Workers Planned: 2"
"              Workers Launched: 2"
"              ->  Parallel Seq Scan on item_details  (cost=0.00..58217.22 rows=2
 width=48) (actual time=85.417..249.045 rows=1 loops=3)"
"                    Filter: (ordid = 1000000)"
"                    Rows Removed by Filter: 1333332"
"        ->  Materialize  (cost=1000.00..56460.07 rows=3 width=44) (actual
 time=75.738..75.753 rows=4 loops=4)"
"              ->  Gather  (cost=1000.00..56460.06 rows=3 width=44) (actual
 time=302.947..303.005 rows=4 loops=1)"
```

```
"                    Workers Planned: 2"
"                    Workers Launched: 2"
"                    ->  Parallel Seq Scan on payment_details  (cost=0.00..55459.76
 rows=1 width=44) (actual time=184.609..294.784 rows=1 loops=3)"
"                          Filter: (ordid = 1000000)"
"                          Rows Removed by Filter: 1333332"
"  ->  Materialize  (cost=1000.00..78580.34 rows=5 width=74) (actual time=8.103..17.238
 rows=5 loops=16)"
"        ->  Gather  (cost=1000.00..78580.31 rows=5 width=74) (actual
 time=129.641..275.795 rows=5 loops=1)"
"              Workers Planned: 2"
"              Workers Launched: 2"
"              ->  Parallel Seq Scan on order_details a  (cost=0.00..77579.81 rows=2
 width=74) (actual time=78.556..268.994 rows=2 loops=3)"
"                    Filter: (ordid = 1000000)"
"                    Rows Removed by Filter: 1666665"
"Planning Time: 0.108 ms"
"Execution Time: 732.953 ms"
```

Ini adalah pengamatan dari data contoh. Saat Anda menjalankan kueri pada kumpulan data besar, perbedaan kinerja akan sangat tinggi.

Pertanyaan yang Sering Diajukan

Temukan jawaban atas pertanyaan yang sering diajukan tentang tuning kinerja kueri.

Apa itu MENJELASKAN?

`EXPLAIN`adalah kata kunci yang Anda masukkan ke kueri PostgreSQL (,`SELECT`,`UPDATE`,`INSERT`) untuk menghasilkan rencana kueri. `DELETE` Rencana kueri PostgreSQL merinci bagaimana database bermaksud menjalankan kueri. Paket ini mencakup informasi tentang urutan pemindaian tabel, penggunaan indeks, dan gabungan.

Gunakan rencana kueri untuk mengidentifikasi potensi kemacetan, mengoptimalkan kueri, dan meningkatkan kinerja secara keseluruhan. Saat meninjau rencana kueri, pertimbangkan faktor-faktor berikut:

- Pendekatan akses tabel
- Bergabunglah dengan pendekatan
- Kondisi filter
- Urutkan operasi
- Penggunaan indeks
- Paralelisme
- Statistik
- Estimasi biaya
- Baris diambil dari setiap langkah
- Distribusi data

Untuk informasi selengkapnya tentang EXPLOW, lihat dokumentasi PostgreSQL.

Apa itu EXPLY ANALYSIS?

Saat Anda menambahkan `EXPLAIN ANALYZE` kueri dan menjalankan kueri, PostgreSQL menjalankan kueri dan mengembalikan statistik rencana kueri dan runtime. Runtime aktual, baris yang diproses dari setiap langkah, dan informasi relevan lainnya ditampilkan bersama dengan

rencana kueri. Menggunakan `EXPLAIN ANALYZE` pada database produksi harus dilakukan dengan hati-hati, karena menjalankan kueri dapat memengaruhi kinerja database selama analisis.

Untuk informasi selengkapnya tentang EXPLOW ANALYSIS, lihat dokumentasi PostgreSQL.

Apa itu pemeriksaan di PostgreSQL?

Dalam PostgreSQL, pemeriksaan adalah seperangkat aturan untuk menentukan bagaimana string dibandingkan dan diurutkan. Pengumpulan mendefinisikan urutan karakter dipertimbangkan dalam perbandingan, dengan mempertimbangkan aturan dan konversi khusus bahasa.

Untuk informasi selengkapnya tentang pemeriksaan, lihat dokumentasi PostgreSQL.

Apa itu CTE?

Dalam database PostgreSQL, Common Table Expression (CTE) adalah kumpulan hasil sementara bernama yang dapat Anda referensikan. CTE menyediakan cara untuk membuat kueri SQL yang lebih mudah dibaca dan modular dengan memecah logika kompleks menjadi unit bernama yang lebih kecil.

Untuk informasi selengkapnya tentang CTE, lihat dokumentasi PostgreSQL.

Apa kategori fungsi di PostgreSQL?

Setiap fungsi PostgreSQL memiliki klasifikasi volatilitas, dengan kemungkinan, atau: `VOLATILE` `STABLE` `IMMUTABLE`

- VOLATILE — `VOLATILE` Fungsi dapat melakukan apa saja, termasuk memodifikasi database. Itu dapat mengembalikan hasil yang berbeda pada panggilan berturut-turut dengan argumen yang sama. Pengoptimal tidak membuat asumsi tentang perilaku fungsi tersebut. Kueri yang menggunakan fungsi volatile akan mengevaluasi kembali fungsi di setiap baris di mana nilainya diperlukan.
- STABLE — `STABLE` Fungsi tidak dapat memodifikasi database. Ini dijamin untuk mengembalikan hasil yang sama dengan argumen yang sama untuk semua baris dalam satu pernyataan. Saat Anda menggunakan klasifikasi ini, pengoptimal dapat mengoptimalkan beberapa panggilan fungsi ke satu panggilan. Secara khusus, aman untuk menggunakan ekspresi yang berisi fungsi seperti itu dalam kondisi pemindaian indeks. (Karena pemindaian indeks akan mengevaluasi nilai

perbandingan hanya satu kali, bukan satu kali di setiap baris, tidak valid untuk menggunakan `VOLATILE` fungsi dalam kondisi pemindaian indeks.)

- IMMUTABLE — `IMMUTABLE` Fungsi tidak dapat memodifikasi database dan dijamin untuk mengembalikan hasil yang sama dengan argumen yang sama selamanya. Saat Anda menggunakan klasifikasi ini, pengoptimal dapat mengevaluasi fungsi terlebih dahulu saat kueri memanggilnya dengan argumen konstan. Misalnya, kueri seperti `SELECT ... WHERE x = 2 + 2` dapat disederhanakan saat dilihat`SELECT ... WHERE x = 4`, karena fungsi yang mendasari operator penambahan integer ditandai`IMMUTABLE`.

`VOLATILE`adalah default jika `CREATE FUNCTION` perintah tidak menentukan kategori. Untuk informasi selengkapnya tentang jenis fungsi, lihat dokumentasi PostgreSQL.

Sumber daya

Referensi

- JELASKAN
- Menggunakan EXPLORE
- Support Collation
- DENGAN Kueri (Ekspresi Tabel Umum)

Panduan

- Aktivitas pemeliharaan untuk database PostgreSQL di Amazon RDS dan Amazon Aurora untuk menghindari masalah kinerja
- Menyetel parameter PostgreSQL di Amazon RDS dan Amazon Aurora

Kontributor

Kontributor dokumen ini meliputi:

- Tirumala Dasari, Konsultan Utama - Database, AWS
- Veeranjaneyulu Grandhi, Konsultan Utama - Database, AWS
- Vamsikrishna Jammula, Konsultan - Database, AWS
- Srinivas Potlachervoo, Konsultan Utama Senior - Database, AWS
- Naga Srinivas Reddy Ravulapati, Konsultan - Database, AWS

Riwayat dokumen

Tabel berikut menjelaskan perubahan signifikan pada panduan ini. Jika Anda ingin diberi tahu tentang pembaruan masa depan, Anda dapat berlangganan umpan RSS.

Perubahan	Deskripsi	Tanggal
Publikasi awal	—	April 23, 2024

AWS Glosarium Panduan Preskriptif

Berikut ini adalah istilah yang umum digunakan dalam strategi, panduan, dan pola yang disediakan oleh Panduan AWS Preskriptif. Untuk menyarankan entri, silakan gunakan tautan Berikan umpan balik di akhir glosarium.

Nomor

7 Rs

Tujuh strategi migrasi umum untuk memindahkan aplikasi ke cloud. Strategi ini dibangun di atas 5 Rs yang diidentifikasi Gartner pada tahun 2011 dan terdiri dari yang berikut:

- Refactor/Re-Architect — Memindahkan aplikasi dan memodifikasi arsitekturnya dengan memanfaatkan sepenuhnya fitur cloud-native untuk meningkatkan kelincahan, kinerja, dan skalabilitas. Ini biasanya melibatkan porting sistem operasi dan database. Contoh: Migrasikan database Oracle lokal Anda ke Amazon Aurora Postgre -Compatible Edition. SQL
- Replatform (angkat dan bentuk ulang) — Pindahkan aplikasi ke cloud, dan perkenalkan beberapa tingkat pengoptimalan untuk memanfaatkan kemampuan cloud. Contoh: Migrasikan database Oracle lokal Anda ke Amazon Relational Database Service (RDSAmazon) untuk Oracle di. AWS Cloud
- Pembelian kembali (drop and shop) - Beralih ke produk yang berbeda, biasanya dengan beralih dari lisensi tradisional ke model SaaS. Contoh: Migrasikan sistem manajemen hubungan pelanggan (CRM) Anda ke Salesforce.com.
- Rehost (lift dan shift) — Pindahkan aplikasi ke cloud tanpa membuat perubahan apa pun untuk memanfaatkan kemampuan cloud. Contoh: Migrasikan database Oracle lokal Anda ke Oracle pada instance EC2 di. AWS Cloud
- Relokasi (hypervisor-level lift and shift) — Pindahkan infrastruktur ke cloud tanpa membeli perangkat keras baru, menulis ulang aplikasi, atau memodifikasi operasi yang ada. Anda memigrasikan server dari platform lokal ke layanan cloud untuk platform yang sama. Contoh: Migrasi a Microsoft Hyper-V aplikasi untuk AWS.
- Pertahankan (kunjungi kembali) - Simpan aplikasi di lingkungan sumber Anda. Ini mungkin termasuk aplikasi yang memerlukan refactoring besar, dan Anda ingin menunda pekerjaan itu sampai nanti, dan aplikasi lama yang ingin Anda pertahankan, karena tidak ada pembenaran bisnis untuk memigrasikannya.

- Pensiun — Menonaktifkan atau menghapus aplikasi yang tidak lagi diperlukan di lingkungan sumber Anda.

A

ABAC

Lihat kontrol akses berbasis atribut.

layanan abstrak

Lihat layanan terkelola.

ACID

Lihat atomisitas, konsistensi, isolasi, daya tahan.

migrasi aktif-aktif

Metode migrasi database di mana database sumber dan target tetap sinkron (dengan menggunakan alat replikasi dua arah atau operasi penulisan ganda), dan kedua database menangani transaksi dari menghubungkan aplikasi selama migrasi. Metode ini mendukung migrasi dalam batch kecil yang terkontrol alih-alih memerlukan pemotongan satu kali. Ini lebih fleksibel tetapi membutuhkan lebih banyak pekerjaan daripada migrasi aktif-pasif.

migrasi aktif-pasif

Metode migrasi database di mana database sumber dan target disimpan dalam sinkron, tetapi hanya database sumber yang menangani transaksi dari menghubungkan aplikasi sementara data direplikasi ke database target. Basis data target tidak menerima transaksi apa pun selama migrasi.

fungsi agregat

SQLFungsi yang beroperasi pada sekelompok baris dan menghitung nilai pengembalian tunggal untuk grup. Contoh fungsi agregat meliputi `SUM` dan`MAX`.

AI

Lihat kecerdasan buatan.

AIOps

Lihat operasi kecerdasan buatan.

anonimisasi

Proses menghapus informasi pribadi secara permanen dalam kumpulan data. Anonimisasi dapat membantu melindungi privasi pribadi. Data anonim tidak lagi dianggap sebagai data pribadi.

anti-pola

Solusi yang sering digunakan untuk masalah berulang di mana solusinya kontra-produktif, tidak efektif, atau kurang efektif daripada alternatif.

kontrol aplikasi

Pendekatan keamanan yang memungkinkan penggunaan hanya aplikasi yang disetujui untuk membantu melindungi sistem dari malware.

portofolio aplikasi

Kumpulan informasi rinci tentang setiap aplikasi yang digunakan oleh organisasi, termasuk biaya untuk membangun dan memelihara aplikasi, dan nilai bisnisnya. Informasi ini adalah kunci untuk penemuan portofolio dan proses analisis dan membantu mengidentifikasi dan memprioritaskan aplikasi yang akan dimigrasi, dimodernisasi, dan dioptimalkan.

kecerdasan buatan (AI)

Bidang ilmu komputer yang didedikasikan untuk menggunakan teknologi komputasi untuk melakukan fungsi kognitif yang biasanya terkait dengan manusia, seperti belajar, memecahkan masalah, dan mengenali pola. Untuk informasi lebih lanjut, lihat Apa itu Kecerdasan Buatan?

operasi kecerdasan buatan (AIOps)

Proses menggunakan teknik pembelajaran mesin untuk memecahkan masalah operasional, mengurangi insiden operasional dan intervensi manusia, dan meningkatkan kualitas layanan. Untuk informasi selengkapnya tentang cara AIOps digunakan dalam strategi AWS migrasi, lihat panduan integrasi operasi.

enkripsi asimetris

Algoritma enkripsi yang menggunakan sepasang kunci, kunci publik untuk enkripsi dan kunci pribadi untuk dekripsi. Anda dapat berbagi kunci publik karena tidak digunakan untuk dekripsi, tetapi akses ke kunci pribadi harus sangat dibatasi.

atomisitas, konsistensi, isolasi, daya tahan () ACID

Satu set properti perangkat lunak yang menjamin validitas data dan keandalan operasional database, bahkan dalam kasus kesalahan, kegagalan daya, atau masalah lainnya.

kontrol akses berbasis atribut () ABAC

Praktik membuat izin berbutir halus berdasarkan atribut pengguna, seperti departemen, peran pekerjaan, dan nama tim. Untuk informasi selengkapnya, lihat ABAC AWS di dokumentasi AWS Identity and Access Management (IAM).

sumber data otoritatif

Lokasi di mana Anda menyimpan versi utama data, yang dianggap sebagai sumber informasi yang paling dapat diandalkan. Anda dapat menyalin data dari sumber data otoritatif ke lokasi lain untuk tujuan memproses atau memodifikasi data, seperti menganonimkan, menyunting, atau membuat nama samaran.

Zona Ketersediaan

Lokasi berbeda di dalam Wilayah AWS yang terisolasi dari kegagalan di Availability Zone lainnya dan menyediakan konektivitas jaringan latensi rendah yang murah ke Availability Zone lainnya di Wilayah yang sama.

AWS Kerangka Adopsi Cloud (AWS CAF)

Kerangka pedoman dan praktik terbaik AWS untuk membantu organisasi mengembangkan rencana yang efisien dan efektif untuk bergerak dengan sukses ke cloud. AWS CAFMengatur panduan ke dalam enam bidang fokus yang disebut perspektif: bisnis, orang, tata kelola, platform, keamanan, dan operasi. Perspektif bisnis, orang, dan tata kelola fokus pada keterampilan dan proses bisnis; perspektif platform, keamanan, dan operasi fokus pada keterampilan dan proses teknis. Misalnya, perspektif masyarakat menargetkan pemangku kepentingan yang menangani sumber daya manusia (SDM), fungsi kepegawaian, dan manajemen orang. Untuk perspektif ini, AWS CAF berikan panduan untuk pengembangan, pelatihan, dan komunikasi orang untuk membantu mempersiapkan organisasi untuk adopsi cloud yang sukses. Untuk informasi lebih lanjut, lihat situs AWS CAFweb dan AWS CAFwhitepaper.

AWS Kerangka Kualifikasi Beban Kerja ()AWS WQF

Alat yang mengevaluasi beban kerja migrasi database, merekomendasikan strategi migrasi, dan memberikan perkiraan kerja. AWS WQFdisertakan dengan AWS Schema Conversion Tool (AWS SCT). Ini menganalisis skema database dan objek kode, kode aplikasi, dependensi, dan karakteristik kinerja, dan memberikan laporan penilaian.

B

bot buruk

Bot yang dimaksudkan untuk mengganggu atau membahayakan individu atau organisasi.

BCP

Lihat perencanaan kontinuitas bisnis.

grafik perilaku

Pandangan interaktif yang terpadu tentang perilaku dan interaksi sumber daya dari waktu ke waktu. Anda dapat menggunakan grafik perilaku dengan Amazon Detective untuk memeriksa upaya logon yang gagal, API panggilan mencurigakan, dan tindakan serupa. Untuk informasi selengkapnya, lihat Data dalam grafik perilaku di dokumentasi Detektif.

sistem big-endian

Sistem yang menyimpan byte paling signifikan terlebih dahulu. Lihat juga endianness.

klasifikasi biner

Sebuah proses yang memprediksi hasil biner (salah satu dari dua kelas yang mungkin). Misalnya, model ML Anda mungkin perlu memprediksi masalah seperti “Apakah email ini spam atau bukan spam?” atau “Apakah produk ini buku atau mobil?”

filter mekar

Struktur data probabilistik dan efisien memori yang digunakan untuk menguji apakah suatu elemen adalah anggota dari suatu himpunan.

deployment biru/hijau

Strategi penyebaran tempat Anda membuat dua lingkungan yang terpisah namun identik. Anda menjalankan versi aplikasi saat ini di satu lingkungan (biru) dan versi aplikasi baru di lingkungan lain (hijau). Strategi ini membantu Anda dengan cepat memutar kembali dengan dampak minimal.

bot

Aplikasi perangkat lunak yang menjalankan tugas otomatis melalui internet dan mensimulasikan aktivitas atau interaksi manusia. Beberapa bot berguna atau bermanfaat, seperti perayap web yang mengindeks informasi di internet. Beberapa bot lain, yang dikenal sebagai bot buruk, dimaksudkan untuk mengganggu atau membahayakan individu atau organisasi.

botnet

Jaringan bot yang terinfeksi oleh malware dan berada di bawah kendali satu pihak, yang dikenal sebagai bot herder atau operator bot. Botnet adalah mekanisme paling terkenal untuk skala bot dan dampaknya.

cabang

Area berisi repositori kode. Cabang pertama yang dibuat dalam repositori adalah cabang utama. Anda dapat membuat cabang baru dari cabang yang ada, dan Anda kemudian dapat mengembangkan fitur atau memperbaiki bug di cabang baru. Cabang yang Anda buat untuk membangun fitur biasanya disebut sebagai cabang fitur. Saat fitur siap dirilis, Anda menggabungkan cabang fitur kembali ke cabang utama. Untuk informasi selengkapnya, lihat Tentang cabang (GitHub dokumentasi).

akses break-glass

Dalam keadaan luar biasa dan melalui proses yang disetujui, cara cepat bagi pengguna untuk mendapatkan akses ke Akun AWS yang biasanya tidak memiliki izin untuk mengaksesnya. Untuk informasi lebih lanjut, lihat indikator Implementasikan prosedur break-glass dalam panduan Well-Architected AWS .

strategi brownfield

Infrastruktur yang ada di lingkungan Anda. Saat mengadopsi strategi brownfield untuk arsitektur sistem, Anda merancang arsitektur di sekitar kendala sistem dan infrastruktur saat ini. Jika Anda memperluas infrastruktur yang ada, Anda dapat memadukan strategi brownfield dan greenfield.

cache penyangga

Area memori tempat data yang paling sering diakses disimpan.

kemampuan bisnis

Apa yang dilakukan bisnis untuk menghasilkan nilai (misalnya, penjualan, layanan pelanggan, atau pemasaran). Arsitektur layanan mikro dan keputusan pengembangan dapat didorong oleh kemampuan bisnis. Untuk informasi selengkapnya, lihat bagian Terorganisir di sekitar kemampuan bisnis dari Menjalankan layanan mikro kontainer di whitepaper. AWS

perencanaan kelangsungan bisnis () BCP

Rencana yang membahas dampak potensial dari peristiwa yang mengganggu, seperti migrasi skala besar, pada operasi dan memungkinkan bisnis untuk melanjutkan operasi dengan cepat.

C

CAF

Lihat Kerangka Adopsi AWS Cloud.

penyebaran kenari

Rilis versi yang lambat dan bertahap untuk pengguna akhir. Ketika Anda yakin, Anda menyebarkan versi baru dan mengganti versi saat ini secara keseluruhan.

CCoE

Lihat Cloud Center of Excellence.

CDC

Lihat mengubah pengambilan data.

ubah pengambilan data (CDC)

Proses melacak perubahan ke sumber data, seperti tabel database, dan merekam metadata tentang perubahan tersebut. Anda dapat menggunakan CDC untuk berbagai tujuan, seperti mengaudit atau mereplikasi perubahan dalam sistem target untuk mempertahankan sinkronisasi.

rekayasa kekacauan

Sengaja memperkenalkan kegagalan atau peristiwa yang mengganggu untuk menguji ketahanan sistem. Anda dapat menggunakan AWS Fault Injection Service (AWS FIS) untuk melakukan eksperimen yang menekankan AWS beban kerja Anda dan mengevaluasi responsnya.

CI/CD

Lihat integrasi berkelanjutan dan pengiriman berkelanjutan.

klasifikasi

Proses kategorisasi yang membantu menghasilkan prediksi. Model ML untuk masalah klasifikasi memprediksi nilai diskrit. Nilai diskrit selalu berbeda satu sama lain. Misalnya, model mungkin perlu mengevaluasi apakah ada mobil dalam gambar atau tidak.

Enkripsi sisi klien

Enkripsi data secara lokal, sebelum target Layanan AWS menerimanya.

Pusat Keunggulan Cloud (CCoE)

Tim multi-disiplin yang mendorong upaya adopsi cloud di seluruh organisasi, termasuk mengembangkan praktik terbaik cloud, memobilisasi sumber daya, menetapkan jadwal migrasi, dan memimpin organisasi melalui transformasi skala besar. Untuk informasi selengkapnya, lihat CCoEposting di Blog Strategi AWS Cloud Perusahaan.

komputasi cloud

Teknologi cloud yang biasanya digunakan untuk penyimpanan data jarak jauh dan manajemen perangkat IoT. Cloud computing umumnya terhubung ke teknologi edge computing.

model operasi cloud

Dalam organisasi TI, model operasi yang digunakan untuk membangun, mematangkan, dan mengoptimalkan satu atau lebih lingkungan cloud. Untuk informasi selengkapnya, lihat Membangun Model Operasi Cloud Anda.

tahap adopsi cloud

Empat fase yang biasanya dilalui organisasi ketika mereka bermigrasi ke AWS Cloud:

- Proyek — Menjalankan beberapa proyek terkait cloud untuk bukti konsep dan tujuan pembelajaran
- Foundation — Melakukan investasi dasar untuk meningkatkan adopsi cloud Anda (misalnya, membuat landing zone, mendefinisikanCCoE, membuat model operasi)
- Migrasi — Migrasi aplikasi individual
- Re-invention — Mengoptimalkan produk dan layanan, dan berinovasi di cloud

Tahapan ini didefinisikan oleh Stephen Orban dalam posting blog The Journey Toward Cloud-First & the Stages of Adoption di blog Strategi Perusahaan. AWS Cloud Untuk informasi tentang bagaimana kaitannya dengan strategi AWS migrasi, lihat panduan kesiapan migrasi.

CMDB

Lihat database manajemen konfigurasi.

repositori kode

Lokasi di mana kode sumber dan aset lainnya, seperti dokumentasi, sampel, dan skrip, disimpan dan diperbarui melalui proses kontrol versi. Repositori cloud umum termasuk GitHub atau Bitbucket Cloud. Setiap versi kode disebut cabang. Dalam struktur layanan mikro, setiap repositori

dikhususkan untuk satu bagian fungsionalitas. Pipa CI/CD tunggal dapat menggunakan beberapa repositori.

cache dingin

Cache buffer yang kosong, tidak terisi dengan baik, atau berisi data basi atau tidak relevan. Ini mempengaruhi kinerja karena instance database harus membaca dari memori utama atau disk, yang lebih lambat daripada membaca dari cache buffer.

data dingin

Data yang jarang diakses dan biasanya historis. Saat menanyakan jenis data ini, kueri lambat biasanya dapat diterima. Memindahkan data ini ke tingkat penyimpanan atau kelas yang berkinerja lebih rendah dan lebih murah dapat mengurangi biaya.

visi komputer (CV)

Bidang AI yang menggunakan pembelajaran mesin untuk menganalisis dan mengekstrak informasi dari format visual seperti gambar dan video digital. Misalnya, AWS Panorama menawarkan perangkat yang menambahkan CV ke jaringan kamera lokal, dan Amazon SageMaker AI menyediakan algoritme pemrosesan gambar untuk CV.

konfigurasi drift

Untuk beban kerja, konfigurasi berubah dari status yang diharapkan. Ini dapat menyebabkan beban kerja menjadi tidak patuh, dan biasanya bertahap dan tidak disengaja.

database manajemen konfigurasi (CMDB)

Repositori yang menyimpan dan mengelola informasi tentang database dan lingkungan TI, termasuk komponen perangkat keras dan perangkat lunak dan konfigurasinya. Anda biasanya menggunakan data dari CMDB dalam penemuan portofolio dan tahap analisis migrasi.

paket kesesuaian

Kumpulan AWS Config aturan dan tindakan remediasi yang dapat Anda kumpulkan untuk menyesuaikan kepatuhan dan pemeriksaan keamanan Anda. Anda dapat menerapkan paket kesesuaian sebagai entitas tunggal di Akun AWS dan Wilayah, atau di seluruh organisasi, dengan menggunakan templat. YAML Untuk informasi selengkapnya, lihat Paket kesesuaian dalam dokumentasi. AWS Config

integrasi berkelanjutan dan pengiriman berkelanjutan (CI/CD)

Proses mengotomatiskan sumber, membangun, menguji, pementasan, dan tahap produksi dari proses rilis perangkat lunak. CI/CD is commonly described as a pipeline. CI/CDdapat membantu

Anda mengotomatiskan proses, meningkatkan produktivitas, meningkatkan kualitas kode, dan memberikan lebih cepat. Untuk informasi lebih lanjut, lihat Manfaat pengiriman berkelanjutan. CD juga dapat berarti penerapan berkelanjutan. Untuk informasi selengkapnya, lihat Continuous Delivery vs Continuous Deployment.

CV

Lihat visi komputer.

D

data saat istirahat

Data yang stasioner di jaringan Anda, seperti data yang ada di penyimpanan.

klasifikasi data

Proses untuk mengidentifikasi dan mengkategorikan data dalam jaringan Anda berdasarkan kekritisan dan sensitivitasnya. Ini adalah komponen penting dari setiap strategi manajemen risiko keamanan siber karena membantu Anda menentukan perlindungan dan kontrol retensi yang tepat untuk data. Klasifikasi data adalah komponen pilar keamanan dalam AWS Well-Architected Framework. Untuk informasi selengkapnya, lihat Klasifikasi data.

penyimpangan data

Variasi yang berarti antara data produksi dan data yang digunakan untuk melatih model ML, atau perubahan yang berarti dalam data input dari waktu ke waktu. Penyimpangan data dapat mengurangi kualitas, akurasi, dan keadilan keseluruhan dalam prediksi model ML.

data dalam transit

Data yang aktif bergerak melalui jaringan Anda, seperti antara sumber daya jaringan.

jala data

Kerangka arsitektur yang menyediakan kepemilikan data terdistribusi dan terdesentralisasi dengan manajemen dan tata kelola terpusat.

minimalisasi data

Prinsip pengumpulan dan pemrosesan hanya data yang sangat diperlukan. Mempraktikkan minimalisasi data di dalamnya AWS Cloud dapat mengurangi risiko privasi, biaya, dan jejak karbon analitik Anda.

perimeter data

Satu set pagar pembatas pencegahan di AWS lingkungan Anda yang membantu memastikan bahwa hanya identitas tepercaya yang mengakses sumber daya tepercaya dari jaringan yang diharapkan. Untuk informasi selengkapnya, lihat Membangun perimeter data pada AWS.

prapemrosesan data

Untuk mengubah data mentah menjadi format yang mudah diuraikan oleh model ML Anda. Preprocessing data dapat berarti menghapus kolom atau baris tertentu dan menangani nilai yang hilang, tidak konsisten, atau duplikat.

asal data

Proses melacak asal dan riwayat data sepanjang siklus hidupnya, seperti bagaimana data dihasilkan, ditransmisikan, dan disimpan.

subjek data

Individu yang datanya dikumpulkan dan diproses.

gudang data

Sistem manajemen data yang mendukung intelijen bisnis, seperti analitik. Gudang data biasanya berisi sejumlah besar data historis, dan biasanya digunakan untuk kueri dan analisis.

bahasa definisi database (DDL)

Pernyataan atau perintah untuk membuat atau memodifikasi struktur tabel dan objek dalam database.

bahasa manipulasi database (DML)

Pernyataan atau perintah untuk memodifikasi (memasukkan, memperbarui, dan menghapus) informasi dalam database.

DDL

Lihat bahasa definisi database.

ansambel yang dalam

Untuk menggabungkan beberapa model pembelajaran mendalam untuk prediksi. Anda dapat menggunakan ansambel dalam untuk mendapatkan prediksi yang lebih akurat atau untuk memperkirakan ketidakpastian dalam prediksi.

pembelajaran mendalam

Subbidang ML yang menggunakan beberapa lapisan jaringan saraf tiruan untuk mengidentifikasi pemetaan antara data input dan variabel target yang diinginkan.

defense-in-depth

Pendekatan keamanan informasi di mana serangkaian mekanisme dan kontrol keamanan dilapisi dengan cermat di seluruh jaringan komputer untuk melindungi kerahasiaan, integritas, dan ketersediaan jaringan dan data di dalamnya. Saat Anda mengadopsi strategi ini AWS, Anda menambahkan beberapa kontrol pada lapisan AWS Organizations struktur yang berbeda untuk membantu mengamankan sumber daya. Misalnya, defense-in-depth pendekatan mungkin menggabungkan otentikasi multi-faktor, segmentasi jaringan, dan enkripsi.

administrator yang didelegasikan

Di AWS Organizations, layanan yang kompatibel dapat mendaftarkan akun AWS anggota untuk mengelola akun organisasi dan mengelola izin untuk layanan tersebut. Akun ini disebut administrator yang didelegasikan untuk layanan itu. Untuk informasi selengkapnya dan daftar layanan yang kompatibel, lihat Layanan yang berfungsi dengan AWS Organizations AWS Organizations dokumentasi.

deployment

Proses pembuatan aplikasi, fitur baru, atau perbaikan kode tersedia di lingkungan target. Deployment melibatkan penerapan perubahan dalam basis kode dan kemudian membangun dan menjalankan basis kode itu di lingkungan aplikasi.

lingkungan pengembangan

Lihat lingkungan.

kontrol detektif

Kontrol keamanan yang dirancang untuk mendeteksi, mencatat, dan memperingatkan setelah suatu peristiwa terjadi. Kontrol ini adalah garis pertahanan kedua, memperingatkan Anda tentang peristiwa keamanan yang melewati kontrol pencegahan di tempat. Untuk informasi selengkapnya, lihat Kontrol Detektif dalam Menerapkan kontrol keamanan pada. AWS

pemetaan aliran nilai pengembangan () DVSM

Sebuah proses yang digunakan untuk mengidentifikasi dan memprioritaskan kendala yang mempengaruhi kecepatan dan kualitas dalam siklus hidup pengembangan perangkat lunak. DVSMmemperluas proses pemetaan aliran nilai yang awalnya dirancang untuk praktik manufaktur

lean. Ini berfokus pada langkah-langkah dan tim yang diperlukan untuk menciptakan dan memindahkan nilai melalui proses pengembangan perangkat lunak.

kembar digital

Representasi virtual dari sistem dunia nyata, seperti bangunan, pabrik, peralatan industri, atau jalur produksi. Kembar digital mendukung pemeliharaan prediktif, pemantauan jarak jauh, dan optimalisasi produksi.

tabel dimensi

Dalam skema bintang, tabel yang lebih kecil yang berisi atribut data tentang data kuantitatif dalam tabel fakta. Atribut tabel dimensi biasanya bidang teks atau angka diskrit yang berperilaku seperti teks. Atribut ini biasanya digunakan untuk pembatasan kueri, pemfilteran, dan pelabelan set hasil.

musibah

Peristiwa yang mencegah beban kerja atau sistem memenuhi tujuan bisnisnya di lokasi utama yang digunakan. Peristiwa ini dapat berupa bencana alam, kegagalan teknis, atau akibat dari tindakan manusia, seperti kesalahan konfigurasi yang tidak disengaja atau serangan malware.

pemulihan bencana (DR)

Strategi dan proses yang Anda gunakan untuk meminimalkan downtime dan kehilangan data yang disebabkan oleh bencana. Untuk informasi selengkapnya, lihat Disaster Recovery of Workloads on AWS: Recovery in the Cloud in the AWS Well-Architected Framework.

DML

Lihat bahasa manipulasi basis data.

desain berbasis domain

Pendekatan untuk mengembangkan sistem perangkat lunak yang kompleks dengan menghubungkan komponennya ke domain yang berkembang, atau tujuan bisnis inti, yang dilayani setiap komponen. Konsep ini diperkenalkan oleh Eric Evans dalam bukunya, Domain-Driven Design: Tackling Complexity in the Heart of Software (Boston: Addison-Wesley Professional, 2003). Untuk informasi tentang cara menggunakan desain berbasis domain dengan pola ara pencekik, lihat Memodernisasi Microsoft lama. ASP NET(ASMX) layanan web secara bertahap dengan menggunakan container dan Amazon API Gateway.

DR

Lihat pemulihan bencana.

deteksi drift

Melacak penyimpangan dari konfigurasi dasar. Misalnya, Anda dapat menggunakan AWS CloudFormation untuk mendeteksi penyimpangan dalam sumber daya sistem, atau Anda dapat menggunakannya AWS Control Tower untuk mendeteksi perubahan di landing zone yang mungkin memengaruhi kepatuhan terhadap persyaratan tata kelola.

DVSM

Lihat pemetaan aliran nilai pengembangan.

E

EDA

Lihat analisis data eksplorasi.

EDI

Lihat pertukaran data elektronik.

komputasi tepi

Teknologi yang meningkatkan daya komputasi untuk perangkat pintar di tepi jaringan IoT. Jika dibandingkan dengan komputasi awan, komputasi tepi dapat mengurangi latensi komunikasi dan meningkatkan waktu respons.

pertukaran data elektronik () EDI

Pertukaran otomatis dokumen bisnis antar organisasi. Untuk informasi selengkapnya, lihat Apa itu Pertukaran Data Elektronik.

enkripsi

Proses komputasi yang mengubah data plaintext, yang dapat dibaca manusia, menjadi ciphertext.

kunci enkripsi

String kriptografi dari bit acak yang dihasilkan oleh algoritma enkripsi. Panjang kunci dapat bervariasi, dan setiap kunci dirancang agar tidak dapat diprediksi dan unik.

endianness

Urutan byte disimpan dalam memori komputer. Sistem big-endian menyimpan byte paling signifikan terlebih dahulu. Sistem little-endian menyimpan byte paling tidak signifikan terlebih dahulu.

titik akhir

Lihat titik akhir layanan.

layanan endpoint

Layanan yang dapat Anda host di cloud pribadi virtual (VPC) untuk dibagikan dengan pengguna lain. Anda dapat membuat layanan endpoint dengan AWS PrivateLink dan memberikan izin kepada prinsipal lain Akun AWS atau to AWS Identity and Access Management (IAM). Akun atau prinsipal ini dapat terhubung ke layanan endpoint Anda secara pribadi dengan membuat titik akhir antarmuka. VPC Untuk informasi selengkapnya, lihat Membuat layanan titik akhir di dokumentasi Amazon Virtual Private Cloud (AmazonVPC).

perencanaan sumber daya perusahaan (ERP)

Sistem yang mengotomatiskan dan mengelola proses bisnis utama (seperti akuntansi, MES, dan manajemen proyek) untuk suatu perusahaan.

enkripsi amplop

Proses mengenkripsi kunci enkripsi dengan kunci enkripsi lain. Untuk informasi selengkapnya, lihat Enkripsi amplop dalam dokumentasi AWS Key Management Service (AWS KMS).

lingkungan

Sebuah contoh dari aplikasi yang sedang berjalan. Berikut ini adalah jenis lingkungan yang umum dalam komputasi awan:

- Development Environment — Sebuah contoh dari aplikasi yang berjalan yang hanya tersedia untuk tim inti yang bertanggung jawab untuk memelihara aplikasi. Lingkungan pengembangan digunakan untuk menguji perubahan sebelum mempromosikannya ke lingkungan atas. Jenis lingkungan ini kadang-kadang disebut sebagai lingkungan pengujian.
- lingkungan yang lebih rendah — Semua lingkungan pengembangan untuk aplikasi, seperti yang digunakan untuk build awal dan pengujian.
- lingkungan produksi — Sebuah contoh dari aplikasi yang berjalan yang dapat diakses oleh pengguna akhir. Dalam pipa CI/CD, lingkungan produksi adalah lingkungan penyebaran terakhir.
- lingkungan atas — Semua lingkungan yang dapat diakses oleh pengguna selain tim pengembangan inti. Ini dapat mencakup lingkungan produksi, lingkungan praproduksi, dan lingkungan untuk pengujian penerimaan pengguna.

epik

Dalam metodologi tangkas, kategori fungsional yang membantu mengatur dan memprioritaskan pekerjaan Anda. Epik memberikan deskripsi tingkat tinggi tentang persyaratan dan tugas implementasi. Misalnya, epos AWS CAF keamanan termasuk manajemen identitas dan akses, kontrol detektif, keamanan infrastruktur, perlindungan data, dan respons insiden. Untuk informasi selengkapnya tentang epos dalam strategi AWS migrasi, lihat panduan implementasi program.

ERP

Lihat perencanaan sumber daya perusahaan.

analisis data eksplorasi () EDA

Proses menganalisis dataset untuk memahami karakteristik utamanya. Anda mengumpulkan atau mengumpulkan data dan kemudian melakukan penyelidikan awal untuk menemukan pola, mendeteksi anomali, dan memeriksa asumsi. EDAdilakukan dengan menghitung statistik ringkasan dan membuat visualisasi data.

F

tabel fakta

Tabel tengah dalam skema bintang. Ini menyimpan data kuantitatif tentang operasi bisnis. Biasanya, tabel fakta berisi dua jenis kolom: kolom yang berisi ukuran dan yang berisi kunci asing ke tabel dimensi.

gagal cepat

Filosofi yang menggunakan pengujian yang sering dan bertahap untuk mengurangi siklus hidup pengembangan. Ini adalah bagian penting dari pendekatan tangkas.

batas isolasi kesalahan

Dalam AWS Cloud, batas seperti Availability Zone, Wilayah AWS, control plane, atau data plane yang membatasi efek kegagalan dan membantu meningkatkan ketahanan beban kerja. Untuk informasi selengkapnya, lihat Batas Isolasi AWS Kesalahan.

cabang fitur

Lihat cabang.

fitur

Data input yang Anda gunakan untuk membuat prediksi. Misalnya, dalam konteks manufaktur, fitur bisa berupa gambar yang diambil secara berkala dari lini manufaktur.

pentingnya fitur

Seberapa signifikan fitur untuk prediksi model. Ini biasanya dinyatakan sebagai skor numerik yang dapat dihitung melalui berbagai teknik, seperti Shapley Additive Explanations (SHAP) dan gradien terintegrasi. Untuk informasi lebih lanjut, lihat Interpretabilitas model pembelajaran mesin dengan. AWS

transformasi fitur

Untuk mengoptimalkan data untuk proses ML, termasuk memperkaya data dengan sumber tambahan, menskalakan nilai, atau mengekstrak beberapa set informasi dari satu bidang data. Hal ini memungkinkan model ML untuk mendapatkan keuntungan dari data. Misalnya, jika Anda memecah tanggal “2021-05-27 00:15:37" menjadi “2021", “Mei”, “Kamis”, dan “15", Anda dapat membantu algoritme pembelajaran mempelajari pola bernuansa yang terkait dengan komponen data yang berbeda.

beberapa tembakan mendorong

Memberikan sejumlah kecil contoh yang menunjukkan tugas dan output yang diinginkan sebelum memintanya untuk melakukan tugas serupa. LLM Teknik ini adalah aplikasi pembelajaran dalam konteks, di mana model belajar dari contoh (bidikan) yang tertanam dalam petunjuk. Beberapa bidikan dapat efektif untuk tugas-tugas yang memerlukan pemformatan, penalaran, atau pengetahuan domain tertentu. Lihat juga bidikan nol.

FGAC

Lihat kontrol akses berbutir halus.

kontrol akses berbutir halus () FGAC

Penggunaan beberapa kondisi untuk mengizinkan atau menolak permintaan akses.

migrasi flash-cut

Metode migrasi database yang menggunakan replikasi data berkelanjutan melalui pengambilan data perubahan untuk memigrasikan data dalam waktu sesingkat mungkin, alih-alih menggunakan pendekatan bertahap. Tujuannya adalah untuk menjaga downtime seminimal mungkin.

FM

Lihat model pondasi.

model pondasi (FM)

Jaringan saraf pembelajaran mendalam yang besar yang telah melatih kumpulan data besar-besaran data umum dan tidak berlabel. FMsmampu melakukan berbagai tugas umum, seperti memahami bahasa, menghasilkan teks dan gambar, dan berbicara dalam bahasa alami. Untuk informasi selengkapnya, lihat Apa itu Model Foundation.

G

AI generatif

Subset model AI yang telah dilatih pada sejumlah besar data dan yang dapat menggunakan prompt teks sederhana untuk membuat konten dan artefak baru, seperti gambar, video, teks, dan audio. Untuk informasi lebih lanjut, lihat Apa itu AI Generatif.

pemblokiran geografis

Lihat pembatasan geografis.

pembatasan geografis (pemblokiran geografis)

Di Amazon CloudFront, opsi untuk mencegah pengguna di negara tertentu mengakses distribusi konten. Anda dapat menggunakan daftar izinkan atau daftar blokir untuk menentukan negara yang disetujui dan dilarang. Untuk informasi selengkapnya, lihat Membatasi distribusi geografis konten Anda dalam dokumentasi. CloudFront

Alur kerja Gitflow

Pendekatan di mana lingkungan bawah dan atas menggunakan cabang yang berbeda dalam repositori kode sumber. Alur kerja Gitflow dianggap warisan, dan alur kerja berbasis batang adalah pendekatan modern yang lebih disukai.

gambar emas

Sebuah snapshot dari sistem atau perangkat lunak yang digunakan sebagai template untuk menyebarkan instance baru dari sistem atau perangkat lunak itu. Misalnya, di bidang manufaktur, gambar emas dapat digunakan untuk menyediakan perangkat lunak pada beberapa perangkat dan membantu meningkatkan kecepatan, skalabilitas, dan produktivitas dalam operasi manufaktur perangkat.

strategi greenfield

Tidak adanya infrastruktur yang ada di lingkungan baru. Saat mengadopsi strategi greenfield untuk arsitektur sistem, Anda dapat memilih semua teknologi baru tanpa batasan kompatibilitas dengan infrastruktur yang ada, juga dikenal sebagai brownfield. Jika Anda memperluas infrastruktur yang ada, Anda dapat memadukan strategi brownfield dan greenfield.

pagar pembatas

Aturan tingkat tinggi yang membantu mengatur sumber daya, kebijakan, dan kepatuhan di seluruh unit organisasi ()OUs. Pagar pembatas preventif menegakkan kebijakan untuk memastikan keselarasan dengan standar kepatuhan. Mereka diimplementasikan dengan menggunakan kebijakan kontrol layanan dan batas IAM izin. Detective guardrails mendeteksi pelanggaran kebijakan dan masalah kepatuhan, dan menghasilkan peringatan untuk remediasi. Mereka diimplementasikan dengan menggunakan AWS Config, AWS Security Hub, Amazon GuardDuty AWS Trusted Advisor, Amazon Inspector, dan pemeriksaan khusus AWS Lambda .

H

HA

Lihat ketersediaan tinggi.

migrasi database heterogen

Memigrasi database sumber Anda ke database target yang menggunakan mesin database yang berbeda (misalnya, Oracle ke Amazon Aurora). Migrasi heterogen biasanya merupakan bagian dari upaya arsitektur ulang, dan mengubah skema dapat menjadi tugas yang kompleks. AWS menyediakan AWS SCT yang membantu dengan konversi skema.

ketersediaan tinggi (HA)

Kemampuan beban kerja untuk beroperasi terus menerus, tanpa intervensi, jika terjadi tantangan atau bencana. Sistem HA dirancang untuk gagal secara otomatis, secara konsisten memberikan kinerja berkualitas tinggi, dan menangani beban dan kegagalan yang berbeda dengan dampak kinerja minimal.

modernisasi sejarawan

Pendekatan yang digunakan untuk memodernisasi dan meningkatkan sistem teknologi operasional (OT) untuk melayani kebutuhan industri manufaktur dengan lebih baik. Sejarawan

adalah jenis database yang digunakan untuk mengumpulkan dan menyimpan data dari berbagai sumber di pabrik.

data penahanan

Sebagian dari data historis berlabel yang ditahan dari kumpulan data yang digunakan untuk melatih model pembelajaran mesin. Anda dapat menggunakan data penahanan untuk mengevaluasi kinerja model dengan membandingkan prediksi model dengan data penahanan.

migrasi database homogen

Memigrasi database sumber Anda ke database target yang berbagi mesin database yang sama (misalnya, Microsoft SQL Server ke Amazon RDS untuk SQL Server). Migrasi homogen biasanya merupakan bagian dari upaya rehosting atau replatforming. Anda dapat menggunakan utilitas database asli untuk memigrasi skema.

data panas

Data yang sering diakses, seperti data real-time atau data translasi terbaru. Data ini biasanya memerlukan tingkat atau kelas penyimpanan berkinerja tinggi untuk memberikan respons kueri yang cepat.

perbaikan terbaru

Perbaikan mendesak untuk masalah kritis dalam lingkungan produksi. Karena urgensinya, perbaikan terbaru biasanya dibuat di luar alur kerja DevOps rilis biasa.

periode hypercare

Segera setelah cutover, periode waktu ketika tim migrasi mengelola dan memantau aplikasi yang dimigrasi di cloud untuk mengatasi masalah apa pun. Biasanya, periode ini panjangnya 1-4 hari. Pada akhir periode hypercare, tim migrasi biasanya mentransfer tanggung jawab untuk aplikasi ke tim operasi cloud.

I

IAc

Lihat infrastruktur sebagai kode.

kebijakan berbasis identitas

Kebijakan yang dilampirkan pada satu atau beberapa IAM prinsip yang mendefinisikan izin mereka dalam lingkungan. AWS Cloud

aplikasi idle

Aplikasi yang memiliki rata-rata CPU dan penggunaan memori antara 5 dan 20 persen selama periode 90 hari. Dalam proyek migrasi, adalah umum untuk menghentikan aplikasi ini atau mempertahankannya di tempat.

IIoT

Lihat Internet of Things industri.

infrastruktur yang tidak dapat diubah

Model yang menyebarkan infrastruktur baru untuk beban kerja produksi alih-alih memperbarui, menambal, atau memodifikasi infrastruktur yang ada. Infrastruktur yang tidak dapat diubah secara inheren lebih konsisten, andal, dan dapat diprediksi daripada infrastruktur yang dapat berubah. Untuk informasi selengkapnya, lihat praktik terbaik Deploy using immutable infrastructure di AWS Well-Architected Framework.

masuk (masuknya) VPC

Dalam arsitektur AWS multi-akun, a VPC yang menerima, memeriksa, dan merutekan koneksi jaringan dari luar aplikasi. Arsitektur Referensi AWS Keamanan merekomendasikan pengaturan akun Jaringan Anda dengan inbound, outbound, dan inspeksi VPCs untuk melindungi antarmuka dua arah antara aplikasi Anda dan internet yang lebih luas.

migrasi inkremental

Strategi cutover di mana Anda memigrasikan aplikasi Anda dalam bagian-bagian kecil alih-alih melakukan satu cutover penuh. Misalnya, Anda mungkin hanya memindahkan beberapa layanan mikro atau pengguna ke sistem baru pada awalnya. Setelah Anda memverifikasi bahwa semuanya berfungsi dengan baik, Anda dapat secara bertahap memindahkan layanan mikro atau pengguna tambahan hingga Anda dapat menonaktifkan sistem lama Anda. Strategi ini mengurangi risiko yang terkait dengan migrasi besar.

Industri 4.0

Sebuah istilah yang diperkenalkan oleh Klaus Schwab pada tahun 2016 untuk merujuk pada modernisasi proses manufaktur melalui kemajuan dalam konektivitas, data real-time, otomatisasi, analitik, dan AI/ML.

infrastruktur

Semua sumber daya dan aset yang terkandung dalam lingkungan aplikasi.

infrastruktur sebagai kode (IAc)

Proses penyediaan dan pengelolaan infrastruktur aplikasi melalui satu set file konfigurasi. IAc dirancang untuk membantu Anda memusatkan manajemen infrastruktur, menstandarisasi sumber daya, dan menskalakan dengan cepat sehingga lingkungan baru dapat diulang, andal, dan konsisten.

Internet of Things industri (IIoT)

Penggunaan sensor dan perangkat yang terhubung ke internet di sektor industri, seperti manufaktur, energi, otomotif, perawatan kesehatan, ilmu kehidupan, dan pertanian. Untuk informasi selengkapnya, lihat Membangun strategi transformasi digital Internet of Things (IIoT) industri.

inspeksi VPC

Dalam arsitektur AWS multi-akun, terpusat VPC yang mengelola inspeksi lalu lintas jaringan antara VPCs (dalam hal yang sama atau berbeda Wilayah AWS), internet, dan jaringan lokal. Arsitektur Referensi AWS Keamanan merekomendasikan pengaturan akun Jaringan Anda dengan inbound, outbound, dan inspeksi VPCs untuk melindungi antarmuka dua arah antara aplikasi Anda dan internet yang lebih luas.

Internet of Things (IoT)

Jaringan objek fisik yang terhubung dengan sensor atau prosesor tertanam yang berkomunikasi dengan perangkat dan sistem lain melalui internet atau melalui jaringan komunikasi lokal. Untuk informasi selengkapnya, lihat Apa itu IoT?

interpretabilitas

Karakteristik model pembelajaran mesin yang menggambarkan sejauh mana manusia dapat memahami bagaimana prediksi model bergantung pada inputnya. Untuk informasi lebih lanjut, lihat Interpretabilitas model pembelajaran mesin dengan. AWS

IoT

Lihat Internet of Things.

Perpustakaan informasi TI (ITIL)

Serangkaian praktik terbaik untuk memberikan layanan TI dan menyelaraskan layanan ini dengan persyaratan bisnis. ITILmenyediakan fondasi untukITSM.

Manajemen layanan TI (ITSM)

Kegiatan yang terkait dengan merancang, menerapkan, mengelola, dan mendukung layanan TI untuk suatu organisasi. Untuk informasi tentang mengintegrasikan operasi cloud dengan ITSM alat, lihat panduan integrasi operasi.

ITIL

Lihat perpustakaan informasi TI.

ITSM

Lihat manajemen layanan TI.

L

kontrol akses berbasis label () LBAC

Implementasi kontrol akses wajib (MAC) di mana pengguna dan data itu sendiri masing-masing secara eksplisit diberi nilai label keamanan. Persimpangan antara label keamanan pengguna dan label keamanan data menentukan baris dan kolom mana yang dapat dilihat oleh pengguna.

landing zone

Landing zone adalah AWS lingkungan multi-akun yang dirancang dengan baik yang dapat diskalakan dan aman. Ini adalah titik awal dari mana organisasi Anda dapat dengan cepat meluncurkan dan menyebarkan beban kerja dan aplikasi dengan percaya diri dalam lingkungan keamanan dan infrastruktur mereka. Untuk informasi selengkapnya tentang zona pendaratan, lihat Menyiapkan lingkungan multi-akun AWS yang aman dan dapat diskalakan.

model bahasa besar (LLM)

Model AI pembelajaran mendalam yang dilatih sebelumnya pada sejumlah besar data. An LLM dapat melakukan banyak tugas, seperti menjawab pertanyaan, meringkas dokumen, menerjemahkan teks ke dalam bahasa lain, dan menyelesaikan kalimat. Untuk informasi lebih lanjut, lihat Apa itu LLMs.

migrasi besar

Migrasi 300 atau lebih server.

LBAC

Lihat kontrol akses berbasis label.

hak istimewa paling sedikit

Praktik keamanan terbaik untuk memberikan izin minimum yang diperlukan untuk melakukan tugas. Untuk informasi selengkapnya, lihat Menerapkan izin hak istimewa terkecil dalam dokumentasi. IAM

angkat dan geser

Lihat 7 Rs.

sistem endian kecil

Sebuah sistem yang menyimpan byte paling tidak signifikan terlebih dahulu. Lihat juga endianness.

LLM

Lihat model bahasa besar.

lingkungan yang lebih rendah

Lihat lingkungan.

M

pembelajaran mesin (ML)

Jenis kecerdasan buatan yang menggunakan algoritma dan teknik untuk pengenalan pola dan pembelajaran. ML menganalisis dan belajar dari data yang direkam, seperti data Internet of Things (IoT), untuk menghasilkan model statistik berdasarkan pola. Untuk informasi selengkapnya, lihat Machine Learning.

cabang utama

Lihat cabang.

malware

Perangkat lunak yang dirancang untuk membahayakan keamanan atau privasi komputer. Malware dapat mengganggu sistem komputer, membocorkan informasi sensitif, atau mendapatkan akses yang tidak sah. Contoh malware termasuk virus, worm, ransomware, Trojan horse, spyware, dan keyloggers.

layanan terkelola

Layanan AWS yang AWS mengoperasikan lapisan infrastruktur, sistem operasi, dan platform, dan Anda mengakses titik akhir untuk menyimpan dan mengambil data. Amazon Simple Storage Service (Amazon S3) dan Amazon DynamoDB adalah contoh layanan terkelola. Ini juga dikenal sebagai layanan abstrak.

sistem eksekusi manufaktur (MES)

Sistem perangkat lunak untuk melacak, memantau, mendokumentasikan, dan mengendalikan proses produksi yang mengubah bahan baku menjadi produk jadi di lantai toko.

MAP

Lihat Program Percepatan Migrasi.

mekanisme

Proses lengkap di mana Anda membuat alat, mendorong adopsi alat, dan kemudian memeriksa hasilnya untuk melakukan penyesuaian. Mekanisme adalah siklus yang memperkuat dan meningkatkan dirinya sendiri saat beroperasi. Untuk informasi lebih lanjut, lihat Membangun mekanisme di AWS Well-Architected Framework.

akun anggota

Semua Akun AWS selain akun manajemen yang merupakan bagian dari organisasi di AWS Organizations. Akun dapat menjadi anggota dari hanya satu organisasi pada suatu waktu.

MES

Lihat sistem eksekusi manufaktur.

Transportasi Telemetri Antrian Pesan () MQTT

Protokol komunikasi ringan machine-to-machine (M2M), berdasarkan pola terbitkan/berlangganan, untuk perangkat IoT yang dibatasi sumber daya.

layanan mikro

Layanan kecil dan independen yang berkomunikasi dengan jelas APIs dan biasanya dimiliki oleh tim kecil yang mandiri. Misalnya, sistem asuransi mungkin mencakup layanan mikro yang memetakan kemampuan bisnis, seperti penjualan atau pemasaran, atau subdomain, seperti pembelian, klaim, atau analitik. Manfaat layanan mikro termasuk kelincahan, penskalaan yang fleksibel, penyebaran yang mudah, kode yang dapat digunakan kembali, dan ketahanan. Untuk

informasi selengkapnya, lihat Mengintegrasikan layanan mikro dengan menggunakan layanan tanpa AWS server.

arsitektur microservices

Pendekatan untuk membangun aplikasi dengan komponen independen yang menjalankan setiap proses aplikasi sebagai layanan mikro. Layanan mikro ini berkomunikasi melalui antarmuka yang terdefinisi dengan baik dengan menggunakan ringan. APIs Setiap layanan mikro dalam arsitektur ini dapat diperbarui, digunakan, dan diskalakan untuk memenuhi permintaan fungsi tertentu dari suatu aplikasi. Untuk informasi selengkapnya, lihat Menerapkan layanan mikro di AWS.

Program Percepatan Migrasi (MAP)

AWS Program yang menyediakan dukungan konsultasi, pelatihan, dan layanan untuk membantu organisasi membangun fondasi operasional yang kuat untuk pindah ke cloud, dan untuk membantu mengimbangi biaya awal migrasi. MAPmencakup metodologi migrasi untuk mengeksekusi migrasi lama dengan cara metodis dan seperangkat alat untuk mengotomatiskan dan mempercepat skenario migrasi umum.

migrasi dalam skala

Proses memindahkan sebagian besar portofolio aplikasi ke cloud dalam gelombang, dengan lebih banyak aplikasi bergerak pada tingkat yang lebih cepat di setiap gelombang. Fase ini menggunakan praktik terbaik dan pelajaran yang dipetik dari fase sebelumnya untuk mengimplementasikan pabrik migrasi tim, alat, dan proses untuk merampingkan migrasi beban kerja melalui otomatisasi dan pengiriman tangkas. Ini adalah fase ketiga dari strategi AWS migrasi.

pabrik migrasi

Tim lintas fungsi yang merampingkan migrasi beban kerja melalui pendekatan otomatis dan gesit. Tim pabrik migrasi biasanya mencakup operasi, analis dan pemilik bisnis, insinyur migrasi, pengembang, dan DevOps profesional yang bekerja di sprint. Antara 20 dan 50 persen portofolio aplikasi perusahaan terdiri dari pola berulang yang dapat dioptimalkan dengan pendekatan pabrik. Untuk informasi selengkapnya, lihat diskusi tentang pabrik migrasi dan panduan Pabrik Migrasi Cloud di kumpulan konten ini.

metadata migrasi

Informasi tentang aplikasi dan server yang diperlukan untuk menyelesaikan migrasi. Setiap pola migrasi memerlukan satu set metadata migrasi yang berbeda. Contoh metadata migrasi termasuk subnet target, grup keamanan, dan akun. AWS

pola migrasi

Tugas migrasi berulang yang merinci strategi migrasi, tujuan migrasi, dan aplikasi atau layanan migrasi yang digunakan. Contoh: Rehost migrasi ke Amazon EC2 dengan Layanan Migrasi AWS Aplikasi.

Penilaian Portofolio Migrasi (MPA)

Alat online yang menyediakan informasi untuk memvalidasi kasus bisnis untuk bermigrasi ke. AWS Cloud MPAmemberikan penilaian portofolio terperinci (ukuran kanan server, harga, TCO perbandingan, analisis biaya migrasi) serta perencanaan migrasi (analisis data aplikasi dan pengumpulan data, pengelompokan aplikasi, prioritas migrasi, dan perencanaan gelombang). MPAAlat ini (memerlukan login) tersedia gratis untuk semua AWS konsultan dan konsultan APN Mitra.

Penilaian Kesiapan Migrasi () MRA

Proses mendapatkan wawasan tentang status kesiapan cloud organisasi, mengidentifikasi kekuatan dan kelemahan, dan membangun rencana aksi untuk menutup kesenjangan yang teridentifikasi, menggunakan. AWS CAF Untuk informasi selengkapnya, lihat panduan kesiapan migrasi. MRAIni adalah tahap pertama dari strategi AWS migrasi.

strategi migrasi

Pendekatan yang digunakan untuk memigrasikan beban kerja ke. AWS Cloud Untuk informasi lebih lanjut, lihat entri 7 Rs di glosarium ini dan lihat Memobilisasi organisasi Anda untuk mempercepat migrasi skala besar.

ML

Lihat pembelajaran mesin.

modernisasi

Mengubah aplikasi usang (warisan atau monolitik) dan infrastrukturnya menjadi sistem yang gesit, elastis, dan sangat tersedia di cloud untuk mengurangi biaya, mendapatkan efisiensi, dan memanfaatkan inovasi. Untuk informasi selengkapnya, lihat Strategi untuk memodernisasi aplikasi di. AWS Cloud

penilaian kesiapan modernisasi

Evaluasi yang membantu menentukan kesiapan modernisasi aplikasi organisasi; mengidentifikasi manfaat, risiko, dan dependensi; dan menentukan seberapa baik organisasi dapat mendukung keadaan masa depan aplikasi tersebut. Hasil penilaian adalah cetak biru arsitektur target, peta

jalan yang merinci fase pengembangan dan tonggak untuk proses modernisasi, dan rencana aksi untuk mengatasi kesenjangan yang diidentifikasi. Untuk informasi lebih lanjut, lihat Mengevaluasi kesiapan modernisasi untuk aplikasi di. AWS Cloud

aplikasi monolitik (monolit)

Aplikasi yang berjalan sebagai layanan tunggal dengan proses yang digabungkan secara ketat. Aplikasi monolitik memiliki beberapa kelemahan. Jika satu fitur aplikasi mengalami lonjakan permintaan, seluruh arsitektur harus diskalakan. Menambahkan atau meningkatkan fitur aplikasi monolitik juga menjadi lebih kompleks ketika basis kode tumbuh. Untuk mengatasi masalah ini, Anda dapat menggunakan arsitektur microservices. Untuk informasi lebih lanjut, lihat Menguraikan monolit menjadi layanan mikro.

MPA

Lihat Penilaian Portofolio Migrasi.

MQTT

Lihat Transportasi Telemetri Antrian Pesan.

klasifikasi multiclass

Sebuah proses yang membantu menghasilkan prediksi untuk beberapa kelas (memprediksi satu dari lebih dari dua hasil). Misalnya, model ML mungkin bertanya "Apakah produk ini buku, mobil, atau telepon?" atau "Kategori produk mana yang paling menarik bagi pelanggan ini?"

infrastruktur yang bisa berubah

Model yang memperbarui dan memodifikasi infrastruktur yang ada untuk beban kerja produksi. Untuk meningkatkan konsistensi, keandalan, dan prediktabilitas, AWS Well-Architected Framework merekomendasikan penggunaan infrastruktur yang tidak dapat diubah sebagai praktik terbaik.

O

OAC

Lihat kontrol akses asal.

OAI

Lihat identitas akses asal.

OCM

Lihat manajemen perubahan organisasi.

migrasi offline

Metode migrasi di mana beban kerja sumber diturunkan selama proses migrasi. Metode ini melibatkan waktu henti yang diperpanjang dan biasanya digunakan untuk beban kerja kecil dan tidak kritis.

OI

Lihat integrasi operasi.

OLA

Lihat perjanjian tingkat operasional.

migrasi online

Metode migrasi di mana beban kerja sumber disalin ke sistem target tanpa diambil offline. Aplikasi yang terhubung ke beban kerja dapat terus berfungsi selama migrasi. Metode ini melibatkan waktu henti nol hingga minimal dan biasanya digunakan untuk beban kerja produksi yang kritis.

OPC-UA

Lihat Komunikasi Proses Terbuka - Arsitektur Terpadu.

Komunikasi Proses Terbuka - Arsitektur Terpadu (OPC-UA)

Protokol komunikasi machine-to-machine (M2M) untuk otomasi industri. OPC-UA menyediakan standar interoperabilitas dengan enkripsi data, otentikasi, dan skema otorisasi.

perjanjian tingkat operasional () OLA

Perjanjian yang menjelaskan apa yang dijanjikan oleh kelompok TI fungsional untuk diberikan satu sama lain, untuk mendukung perjanjian tingkat layanan (). SLA

tinjauan kesiapan operasional () ORR

Daftar pertanyaan dan praktik terbaik terkait yang membantu Anda memahami, mengevaluasi, mencegah, atau mengurangi ruang lingkup insiden dan kemungkinan kegagalan. Untuk informasi lebih lanjut, lihat Ulasan Kesiapan Operasional (ORR) dalam Kerangka Kerja AWS Well-Architected.

teknologi operasional (OT)

Sistem perangkat keras dan perangkat lunak yang bekerja dengan lingkungan fisik untuk mengendalikan operasi industri, peralatan, dan infrastruktur. Di bidang manufaktur, integrasi sistem OT dan teknologi informasi (TI) adalah fokus utama untuk transformasi Industri 4.0.

integrasi operasi (OI)

Proses modernisasi operasi di cloud, yang melibatkan perencanaan kesiapan, otomatisasi, dan integrasi. Untuk informasi selengkapnya, lihat panduan integrasi operasi.

jejak organisasi

Jejak yang dibuat oleh AWS CloudTrail itu mencatat semua peristiwa untuk semua Akun AWS dalam organisasi di AWS Organizations. Jejak ini dibuat di setiap Akun AWS bagian organisasi dan melacak aktivitas di setiap akun. Untuk informasi selengkapnya, lihat Membuat jejak untuk organisasi dalam CloudTrail dokumentasi.

manajemen perubahan organisasi (OCM)

Kerangka kerja untuk mengelola transformasi bisnis utama yang mengganggu dari perspektif orang, budaya, dan kepemimpinan. OCMmembantu organisasi mempersiapkan, dan transisi ke, sistem dan strategi baru dengan mempercepat adopsi perubahan, mengatasi masalah transisi, dan mendorong perubahan budaya dan organisasi. Dalam strategi AWS migrasi, kerangka kerja ini disebut percepatan orang, karena kecepatan perubahan yang diperlukan dalam proyek adopsi cloud. Untuk informasi lebih lanjut, lihat OCMpanduannya.

kontrol akses asal (OAC)

Di CloudFront, opsi yang disempurnakan untuk membatasi akses untuk mengamankan konten Amazon Simple Storage Service (Amazon S3) Anda. OACmendukung semua bucket S3 di semua Wilayah AWS, enkripsi sisi server dengan AWS KMS (SSE-KMS), dan dinamis `PUT` dan `DELETE` permintaan ke bucket S3.

identitas akses asal (OAI)

Di CloudFront, opsi untuk membatasi akses untuk mengamankan konten Amazon S3 Anda. Saat Anda menggunakanOAI, CloudFront buat prinsipal yang dapat diautentikasi oleh Amazon S3. Prinsipal yang diautentikasi dapat mengakses konten dalam bucket S3 hanya melalui distribusi tertentu. CloudFront Lihat juga OAC, yang menyediakan kontrol akses yang lebih terperinci dan ditingkatkan.

ORR

Lihat tinjauan kesiapan operasional.

OT

Lihat teknologi operasional.

keluar (jalan keluar) VPC

Dalam arsitektur AWS multi-akun, a VPC yang menangani koneksi jaringan yang dimulai dari dalam aplikasi. Arsitektur Referensi AWS Keamanan merekomendasikan pengaturan akun Jaringan Anda dengan inbound, outbound, dan inspeksi VPCs untuk melindungi antarmuka dua arah antara aplikasi Anda dan internet yang lebih luas.

P

batas izin

Kebijakan IAM manajemen yang dilampirkan pada IAM prinsipal untuk menetapkan izin maksimum yang dapat dimiliki pengguna atau peran. Untuk informasi selengkapnya, lihat Batas izin dalam IAM dokumentasi.

informasi yang dapat diidentifikasi secara pribadi () PII

Informasi yang, jika dilihat secara langsung atau dipasangkan dengan data terkait lainnya, dapat digunakan untuk menyimpulkan identitas individu secara wajar. Contohnya PII termasuk nama, alamat, dan informasi kontak.

PII

Lihat informasi yang dapat diidentifikasi secara pribadi.

buku pedoman

Serangkaian langkah yang telah ditentukan sebelumnya yang menangkap pekerjaan yang terkait dengan migrasi, seperti mengirimkan fungsi operasi inti di cloud. Buku pedoman dapat berupa skrip, runbook otomatis, atau ringkasan proses atau langkah-langkah yang diperlukan untuk mengoperasikan lingkungan modern Anda.

PLC

Lihat pengontrol logika yang dapat diprogram.

PLM

Lihat manajemen siklus hidup produk.

kebijakan

Objek yang dapat menentukan izin (lihat kebijakan berbasis identitas), menentukan kondisi akses (lihat kebijakanberbasis sumber daya), atau menentukan izin maksimum untuk semua akun dalam organisasi di (lihat kebijakan kontrol layanan). AWS Organizations

ketekunan poliglot

Secara independen memilih teknologi penyimpanan data microservice berdasarkan pola akses data dan persyaratan lainnya. Jika layanan mikro Anda memiliki teknologi penyimpanan data yang sama, mereka dapat menghadapi tantangan implementasi atau mengalami kinerja yang buruk. Layanan mikro lebih mudah diimplementasikan dan mencapai kinerja dan skalabilitas yang lebih baik jika mereka menggunakan penyimpanan data yang paling sesuai dengan kebutuhan mereka. Untuk informasi selengkapnya, lihat Mengaktifkan persistensi data di layanan mikro.

penilaian portofolio

Proses menemukan, menganalisis, dan memprioritaskan portofolio aplikasi untuk merencanakan migrasi. Untuk informasi selengkapnya, lihat Mengevaluasi kesiapan migrasi.

predikat

Kondisi kueri yang mengembalikan `true` atau`false`, biasanya terletak di `WHERE` klausa.

predikat pushdown

Teknik optimasi kueri database yang menyaring data dalam kueri sebelum transfer. Ini mengurangi jumlah data yang harus diambil dan diproses dari database relasional, dan meningkatkan kinerja kueri.

kontrol preventif

Kontrol keamanan yang dirancang untuk mencegah suatu peristiwa terjadi. Kontrol ini adalah garis pertahanan pertama untuk membantu mencegah akses tidak sah atau perubahan yang tidak diinginkan ke jaringan Anda. Untuk informasi selengkapnya, lihat Kontrol pencegahan dalam Menerapkan kontrol keamanan pada. AWS

principal

Entitas AWS yang dapat melakukan tindakan dan mengakses sumber daya. Entitas ini biasanya merupakan pengguna root untuk Akun AWS, IAM peran, atau pengguna. Untuk informasi selengkapnya, lihat Prinsip dalam istilah dan konsep Peran dalam IAM dokumentasi.

privasi berdasarkan desain

Pendekatan rekayasa sistem yang memperhitungkan privasi melalui seluruh proses pengembangan.

zona yang dihosting pribadi

Container yang menyimpan informasi tentang bagaimana Anda ingin Amazon Route 53 merespons DNS kueri untuk domain dan subdomainnya dalam satu atau lebih. VPCs Untuk informasi selengkapnya, lihat Bekerja dengan zona yang dihosting pribadi di dokumentasi Route 53.

kontrol proaktif

Kontrol keamanan yang dirancang untuk mencegah penyebaran sumber daya yang tidak sesuai. Kontrol ini memindai sumber daya sebelum disediakan. Jika sumber daya tidak sesuai dengan kontrol, maka itu tidak disediakan. Untuk informasi selengkapnya, lihat panduan referensi Kontrol dalam AWS Control Tower dokumentasi dan lihat Kontrol proaktif dalam Menerapkan kontrol keamanan pada AWS.

manajemen siklus hidup produk () PLM

Manajemen data dan proses untuk suatu produk di seluruh siklus hidupnya, mulai dari desain, pengembangan, dan peluncuran, melalui pertumbuhan dan kematangan, hingga penurunan dan penghapusan.

lingkungan produksi

Lihat lingkungan.

pengontrol logika yang dapat diprogram () PLC

Di bidang manufaktur, komputer yang sangat andal dan mudah beradaptasi yang memantau mesin dan mengotomatiskan proses manufaktur.

rantai cepat

Menggunakan output dari satu LLMprompt sebagai input untuk prompt berikutnya untuk menghasilkan respons yang lebih baik. Teknik ini digunakan untuk memecah tugas yang kompleks menjadi subtugas, atau untuk secara iteratif memperbaiki atau memperluas respons awal. Ini membantu meningkatkan akurasi dan relevansi respons model dan memungkinkan hasil yang lebih terperinci dan dipersonalisasi.

pseudonimisasi

Proses penggantian pengenal pribadi dalam kumpulan data dengan nilai placeholder. Pseudonimisasi dapat membantu melindungi privasi pribadi. Data pseudonim masih dianggap sebagai data pribadi.

publish/subscribe (pub/sub)

Pola yang memungkinkan komunikasi asinkron antara layanan mikro untuk meningkatkan skalabilitas dan daya tanggap. Misalnya, dalam layanan mikro berbasis MES, layanan mikro dapat mempublikasikan pesan peristiwa ke saluran yang dapat berlangganan oleh layanan mikro lainnya. Sistem dapat menambahkan layanan mikro baru tanpa mengubah layanan penerbitan.

Q

rencana kueri

Serangkaian langkah, seperti instruksi, yang digunakan untuk mengakses data dalam sistem database SQL relasional.

regresi rencana kueri

Ketika pengoptimal layanan database memilih rencana yang kurang optimal daripada sebelum perubahan yang diberikan ke lingkungan database. Hal ini dapat disebabkan oleh perubahan statistik, kendala, pengaturan lingkungan, pengikatan parameter kueri, dan pembaruan ke mesin database.

R

RACImatriks

Lihat bertanggung jawab, akuntabel, dikonsultasikan, diinformasikan () RACI.

RAG

Lihat Retrieval Augmented Generation.

ransomware

Perangkat lunak berbahaya yang dirancang untuk memblokir akses ke sistem komputer atau data sampai pembayaran dilakukan.

RASCImatriks

Lihat bertanggung jawab, akuntabel, dikonsultasikan, diinformasikan () RACI.

RCAC

Lihat kontrol akses baris dan kolom.

replika baca

Salinan database yang digunakan untuk tujuan read-only. Anda dapat merutekan kueri ke replika baca untuk mengurangi beban pada database utama Anda.

arsitek ulang

Lihat 7 Rs.

tujuan titik pemulihan (RPO)

Jumlah waktu maksimum yang dapat diterima sejak titik pemulihan data terakhir. Ini menentukan apa yang dianggap sebagai kehilangan data yang dapat diterima antara titik pemulihan terakhir dan gangguan layanan.

tujuan waktu pemulihan (RTO)

Penundaan maksimum yang dapat diterima antara gangguan layanan dan pemulihan layanan.

refactor

Lihat 7 Rs.

Wilayah

Kumpulan AWS sumber daya di wilayah geografis. Masing-masing Wilayah AWS terisolasi dan independen dari yang lain untuk memberikan toleransi kesalahan, stabilitas, dan ketahanan. Untuk informasi selengkapnya, lihat Menentukan Wilayah AWS akun yang dapat digunakan.

regresi

Teknik ML yang memprediksi nilai numerik. Misalnya, untuk memecahkan masalah “Berapa harga rumah ini akan dijual?” Model ML dapat menggunakan model regresi linier untuk memprediksi harga jual rumah berdasarkan fakta yang diketahui tentang rumah (misalnya, luas persegi).

rehost

Lihat 7 Rs.

melepaskan

Dalam proses penyebaran, tindakan mempromosikan perubahan pada lingkungan produksi.

memindahkan

Lihat 7 Rs.

memplatform ulang

Lihat 7 Rs.

pembelian kembali

Lihat 7 Rs.

ketahanan

Kemampuan aplikasi untuk melawan atau pulih dari gangguan. Ketersediaan tinggi dan pemulihan bencana adalah pertimbangan umum ketika merencanakan ketahanan di. AWS Cloud Untuk informasi lebih lanjut, lihat AWS Cloud Ketahanan.

kebijakan berbasis sumber daya

Kebijakan yang dilampirkan ke sumber daya, seperti bucket Amazon S3, titik akhir, atau kunci enkripsi. Jenis kebijakan ini menentukan prinsip mana yang diizinkan mengakses, tindakan yang didukung, dan kondisi lain yang harus dipenuhi.

matriks yang bertanggung jawab, akuntabel, dikonsultasikan, diinformasikan () RACI

Matriks yang mendefinisikan peran dan tanggung jawab untuk semua pihak yang terlibat dalam kegiatan migrasi dan operasi cloud. Nama matriks berasal dari jenis tanggung jawab yang didefinisikan dalam matriks: bertanggung jawab (R), akuntabel (A), dikonsultasikan (C), dan diinformasikan (I). Jenis dukungan (S) adalah opsional. Jika Anda menyertakan dukungan, matriks disebut RASCImatriks, dan jika Anda mengecualikannya, itu disebut RACImatriks.

kontrol responsif

Kontrol keamanan yang dirancang untuk mendorong remediasi efek samping atau penyimpangan dari garis dasar keamanan Anda. Untuk informasi selengkapnya, lihat Kontrol responsif dalam Menerapkan kontrol keamanan pada AWS.

melestarikan

Lihat 7 Rs.

pensiun

Lihat 7 Rs.

Pengambilan Generasi Augmented () RAG

Teknologi AI generatif di mana LLMreferensi sumber data otoritatif yang berada di luar sumber data pelatihannya sebelum menghasilkan respons. Misalnya, RAG model mungkin melakukan pencarian semantik dari basis pengetahuan organisasi atau data kustom. Untuk informasi lebih lanjut, lihat Apa yang dimaksud dengan RAG.

rotasi

Proses memperbarui rahasia secara berkala untuk membuatnya lebih sulit bagi penyerang untuk mengakses kredensil.

kontrol akses baris dan kolom (RCAC)

Penggunaan SQL ekspresi dasar dan fleksibel yang telah menetapkan aturan akses. RCACterdiri dari izin baris dan topeng kolom.

RPO

Lihat tujuan titik pemulihan.

RTO

Lihat tujuan waktu pemulihan.

buku runbook

Satu set prosedur manual atau otomatis yang diperlukan untuk melakukan tugas tertentu. Ini biasanya dibangun untuk merampingkan operasi berulang atau prosedur dengan tingkat kesalahan yang tinggi.

D

SAML 2.0

Standar terbuka yang digunakan oleh banyak penyedia identitas (IdPs). Fitur ini memungkinkan sistem masuk tunggal (SSO) gabungan, sehingga pengguna dapat masuk ke AWS Management Console atau memanggil AWS API operasi tanpa Anda harus membuat pengguna untuk semua orang di IAM organisasi Anda. Untuk informasi lebih lanjut tentang federasi SAML berbasis 2.0, lihat Tentang federasi SAML berbasis 2.0 dalam dokumentasi. IAM

SCADA

Lihat kontrol pengawasan dan akuisisi data.

SCP

Lihat kebijakan kontrol layanan.

Rahasia

Dalam AWS Secrets Manager, informasi rahasia atau terbatas, seperti kata sandi atau kredensi pengguna, yang Anda simpan dalam bentuk terenkripsi. Ini terdiri dari nilai rahasia dan metadata-nya. Nilai rahasia dapat berupa biner, string tunggal, atau beberapa string. Untuk informasi selengkapnya, lihat Apa yang ada di rahasia Secrets Manager? dalam dokumentasi Secrets Manager.

keamanan dengan desain

Pendekatan rekayasa sistem yang memperhitungkan keamanan melalui seluruh proses pengembangan.

kontrol keamanan

Pagar pembatas teknis atau administratif yang mencegah, mendeteksi, atau mengurangi kemampuan pelaku ancaman untuk mengeksploitasi kerentanan keamanan. Ada empat jenis kontrol keamanan utama: preventif, detektif, responsif, dan proaktif.

pengerasan keamanan

Proses mengurangi permukaan serangan untuk membuatnya lebih tahan terhadap serangan. Ini dapat mencakup tindakan seperti menghapus sumber daya yang tidak lagi diperlukan, menerapkan praktik keamanan terbaik untuk memberikan hak istimewa paling sedikit, atau menonaktifkan fitur yang tidak perlu dalam file konfigurasi.

informasi keamanan dan manajemen acara (SIEM) sistem

Alat dan layanan yang menggabungkan sistem manajemen informasi keamanan (SIM) dan manajemen peristiwa keamanan (SEM). Sebuah SIEM sistem mengumpulkan, memantau, dan menganalisis data dari server, jaringan, perangkat, dan sumber lain untuk mendeteksi ancaman dan pelanggaran keamanan, dan untuk menghasilkan peringatan.

otomatisasi respons keamanan

Tindakan yang telah ditentukan dan diprogram yang dirancang untuk secara otomatis merespons atau memulihkan peristiwa keamanan. Otomatisasi ini berfungsi sebagai kontrol keamanan

detektif atau responsif yang membantu Anda menerapkan praktik terbaik AWS keamanan. Contoh tindakan respons otomatis termasuk memodifikasi grup VPC keamanan, menambal EC2 instans Amazon, atau memutar kredensil.

enkripsi sisi server

Enkripsi data di tujuannya, oleh Layanan AWS yang menerimanya.

kebijakan kontrol layanan (SCP)

Kebijakan yang menyediakan kontrol terpusat atas izin untuk semua akun di organisasi. AWS Organizations SCPsmenentukan pagar pembatas atau menetapkan batasan pada tindakan yang dapat didelegasikan oleh administrator kepada pengguna atau peran. Anda dapat menggunakan SCPs daftar izin atau daftar penolakan, untuk menentukan layanan atau tindakan mana yang diizinkan atau dilarang. Untuk informasi selengkapnya, lihat Kebijakan kontrol layanan dalam AWS Organizations dokumentasi.

titik akhir layanan

Titik masuk untuk sebuah Layanan AWS. URL Anda dapat menggunakan endpoint untuk terhubung secara terprogram ke layanan target. Untuk informasi selengkapnya, lihat Layanan AWS titik akhir di Referensi Umum AWS.

perjanjian tingkat layanan () SLA

Perjanjian yang menjelaskan apa yang dijanjikan tim TI untuk diberikan kepada pelanggan mereka, seperti waktu kerja dan kinerja layanan.

indikator tingkat layanan () SLI

Pengukuran aspek kinerja layanan, seperti tingkat kesalahan, ketersediaan, atau throughputnya.

tujuan tingkat layanan () SLO

Metrik target yang mewakili kesehatan layanan, yang diukur dengan indikator tingkat layanan.

model tanggung jawab bersama

Model yang menjelaskan tanggung jawab yang Anda bagikan AWS untuk keamanan dan kepatuhan cloud. AWS bertanggung jawab atas keamanan cloud, sedangkan Anda bertanggung jawab atas keamanan di cloud. Untuk informasi selengkapnya, lihat Model tanggung jawab bersama.

SIEM

Lihat informasi keamanan dan sistem manajemen acara.

satu titik kegagalan (SPOF)

Kegagalan dalam satu komponen penting dari aplikasi yang dapat mengganggu sistem.

SLA

Lihat perjanjian tingkat layanan.

SLI

Lihat indikator tingkat layanan.

SLO

Lihat tujuan tingkat layanan.

split-and-seed model

Pola untuk menskalakan dan mempercepat proyek modernisasi. Ketika fitur baru dan rilis produk didefinisikan, tim inti berpisah untuk membuat tim produk baru. Ini membantu meningkatkan kemampuan dan layanan organisasi Anda, meningkatkan produktivitas pengembang, dan mendukung inovasi yang cepat. Untuk informasi lebih lanjut, lihat Pendekatan bertahap untuk memodernisasi aplikasi di. AWS Cloud

SPOF

Lihat satu titik kegagalan.

skema bintang

Struktur organisasi database yang menggunakan satu tabel fakta besar untuk menyimpan data transaksional atau terukur dan menggunakan satu atau lebih tabel dimensi yang lebih kecil untuk menyimpan atribut data. Struktur ini dirancang untuk digunakan dalam gudang data atau untuk tujuan intelijen bisnis.

pola ara pencekik

Pendekatan untuk memodernisasi sistem monolitik dengan menulis ulang secara bertahap dan mengganti fungsionalitas sistem sampai sistem warisan dapat dinonaktifkan. Pola ini menggunakan analogi pohon ara yang tumbuh menjadi pohon yang sudah mapan dan akhirnya mengatasi dan menggantikan inangnya. Pola ini diperkenalkan oleh Martin Fowler sebagai cara untuk mengelola risiko saat menulis ulang sistem monolitik. Untuk contoh cara menerapkan pola ini, lihat Memodernisasi Microsoft lama. ASP NET(ASMX) layanan web secara bertahap dengan menggunakan container dan Amazon API Gateway.

subnet

Berbagai alamat IP di AndaVPC. Subnet harus berada di Availability Zone tunggal.

kontrol pengawasan dan akuisisi data () SCADA

Di bidang manufaktur, sistem yang menggunakan perangkat keras dan perangkat lunak untuk memantau aset fisik dan operasi produksi.

enkripsi simetris

Algoritma enkripsi yang menggunakan kunci yang sama untuk mengenkripsi dan mendekripsi data.

pengujian sintetis

Menguji sistem dengan cara yang mensimulasikan interaksi pengguna untuk mendeteksi potensi masalah atau untuk memantau kinerja. Anda dapat menggunakan Amazon CloudWatch Synthetics untuk membuat tes ini.

sistem prompt

Teknik untuk memberikan konteks, instruksi, atau pedoman LLMuntuk mengarahkan perilakunya. Permintaan sistem membantu mengatur konteks dan menetapkan aturan untuk interaksi dengan pengguna.

T

tag

Pasangan nilai kunci yang bertindak sebagai metadata untuk mengatur sumber daya Anda. AWS Tanda dapat membantu Anda mengelola, mengidentifikasi, mengatur, dan memfilter sumber daya. Untuk informasi selengkapnya, lihat Menandai AWS sumber daya Anda.

variabel target

Nilai yang Anda coba prediksi dalam ML yang diawasi. Ini juga disebut sebagai variabel hasil. Misalnya, dalam pengaturan manufaktur, variabel target bisa menjadi cacat produk.

daftar tugas

Alat yang digunakan untuk melacak kemajuan melalui runbook. Daftar tugas berisi ikhtisar runbook dan daftar tugas umum yang harus diselesaikan. Untuk setiap tugas umum, itu termasuk perkiraan jumlah waktu yang dibutuhkan, pemilik, dan kemajuan.

lingkungan uji

Lihat lingkungan.

pelatihan

Untuk menyediakan data bagi model ML Anda untuk dipelajari. Data pelatihan harus berisi jawaban yang benar. Algoritma pembelajaran menemukan pola dalam data pelatihan yang memetakan atribut data input ke target (jawaban yang ingin Anda prediksi). Ini menghasilkan model ML yang menangkap pola-pola ini. Anda kemudian dapat menggunakan model ML untuk membuat prediksi pada data baru yang Anda tidak tahu targetnya.

gerbang transit

Hub transit jaringan yang dapat Anda gunakan untuk menghubungkan jaringan Anda VPCs dan lokal. Untuk informasi selengkapnya, lihat Apa itu gateway transit dalam AWS Transit Gateway dokumentasi.

alur kerja berbasis batang

Pendekatan di mana pengembang membangun dan menguji fitur secara lokal di cabang fitur dan kemudian menggabungkan perubahan tersebut ke cabang utama. Cabang utama kemudian dibangun untuk pengembangan, praproduksi, dan lingkungan produksi, secara berurutan.

akses tepercaya

Memberikan izin ke layanan yang Anda tentukan untuk melakukan tugas di organisasi Anda di dalam AWS Organizations dan di akunnya atas nama Anda. Layanan tepercaya menciptakan peran terkait layanan di setiap akun, ketika peran itu diperlukan, untuk melakukan tugas manajemen untuk Anda. Untuk informasi selengkapnya, lihat Menggunakan AWS Organizations dengan AWS layanan lain dalam AWS Organizations dokumentasi.

penyetelan

Untuk mengubah aspek proses pelatihan Anda untuk meningkatkan akurasi model ML. Misalnya, Anda dapat melatih model ML dengan membuat set pelabelan, menambahkan label, dan kemudian mengulangi langkah-langkah ini beberapa kali di bawah pengaturan yang berbeda untuk mengoptimalkan model.

tim dua pizza

Sebuah DevOps tim kecil yang bisa Anda beri makan dengan dua pizza. Ukuran tim dua pizza memastikan peluang terbaik untuk berkolaborasi dalam pengembangan perangkat lunak.

U

waswas

Sebuah konsep yang mengacu pada informasi yang tidak tepat, tidak lengkap, atau tidak diketahui yang dapat merusak keandalan model ML prediktif. Ada dua jenis ketidakpastian: ketidakpastian epistemik disebabkan oleh data yang terbatas dan tidak lengkap, sedangkan ketidakpastian aleatorik disebabkan oleh kebisingan dan keacakan yang melekat dalam data. Untuk informasi lebih lanjut, lihat panduan Mengukur ketidakpastian dalam sistem pembelajaran mendalam.

tugas yang tidak terdiferensiasi

Juga dikenal sebagai angkat berat, pekerjaan yang diperlukan untuk membuat dan mengoperasikan aplikasi tetapi itu tidak memberikan nilai langsung kepada pengguna akhir atau memberikan keunggulan kompetitif. Contoh tugas yang tidak terdiferensiasi termasuk pengadaan, pemeliharaan, dan perencanaan kapasitas.

lingkungan atas

Lihat lingkungan.

V

menyedot debu

Operasi pemeliharaan database yang melibatkan pembersihan setelah pembaruan tambahan untuk merebut kembali penyimpanan dan meningkatkan kinerja.

kendali versi

Proses dan alat yang melacak perubahan, seperti perubahan kode sumber dalam repositori.

VPCmengintip

Koneksi antara dua VPCs yang memungkinkan Anda untuk merutekan lalu lintas dengan menggunakan alamat IP pribadi. Untuk informasi selengkapnya, lihat Apa yang VPC mengintip di VPC dokumentasi Amazon.

kerentanan

Kelemahan perangkat lunak atau perangkat keras yang membahayakan keamanan sistem.

W

cache hangat

Cache buffer yang berisi data saat ini dan relevan yang sering diakses. Instance database dapat membaca dari cache buffer, yang lebih cepat daripada membaca dari memori utama atau disk.

data hangat

Data yang jarang diakses. Saat menanyakan jenis data ini, kueri yang cukup lambat biasanya dapat diterima.

fungsi jendela

SQLFungsi yang melakukan perhitungan pada sekelompok baris yang berhubungan dengan catatan saat ini. Fungsi jendela berguna untuk memproses tugas, seperti menghitung rata-rata bergerak atau mengakses nilai baris berdasarkan posisi relatif dari baris saat ini.

beban kerja

Kumpulan sumber daya dan kode yang memberikan nilai bisnis, seperti aplikasi yang dihadapi pelanggan atau proses backend.

aliran kerja

Grup fungsional dalam proyek migrasi yang bertanggung jawab atas serangkaian tugas tertentu. Setiap alur kerja independen tetapi mendukung alur kerja lain dalam proyek. Misalnya, alur kerja portofolio bertanggung jawab untuk memprioritaskan aplikasi, perencanaan gelombang, dan mengumpulkan metadata migrasi. Alur kerja portofolio mengirimkan aset ini ke alur kerja migrasi, yang kemudian memigrasikan server dan aplikasi.

WORM

Lihat menulis sekali, baca banyak.

WQF

Lihat AWSKerangka Kualifikasi Beban Kerja.

tulis sekali, baca banyak (WORM)

Model penyimpanan yang menulis data satu kali dan mencegah data dihapus atau dimodifikasi. Pengguna yang berwenang dapat membaca data sebanyak yang diperlukan, tetapi mereka tidak dapat mengubahnya. Infrastruktur penyimpanan data ini dianggap tidak dapat diubah.

Z

eksploitasi zero-day

Serangan, biasanya malware, yang memanfaatkan kerentanan zero-day.

kerentanan zero-day

Cacat atau kerentanan yang tak tanggung-tanggung dalam sistem produksi. Aktor ancaman dapat menggunakan jenis kerentanan ini untuk menyerang sistem. Pengembang sering menyadari kerentanan sebagai akibat dari serangan tersebut.

bisikan zero-shot

LLMMemberikan instruksi untuk melakukan tugas tetapi tidak ada contoh (bidikan) yang dapat membantu membimbingnya. LLMHarus menggunakan pengetahuan pra-terlatih untuk menangani tugas. Efektivitas bidikan nol tergantung pada kompleksitas tugas dan kualitas prompt. Lihat juga beberapa bidikan yang diminta.

aplikasi zombie

Aplikasi yang memiliki rata-rata CPU dan penggunaan memori di bawah 5 persen. Dalam proyek migrasi, adalah umum untuk menghentikan aplikasi ini.

Terjemahan disediakan oleh mesin penerjemah. Jika konten terjemahan yang diberikan bertentangan dengan versi bahasa Inggris aslinya, utamakan versi bahasa Inggris.