

Veritabanı Yönetim Sistemleri

Dönem Projesi

Proje Adı: Kafka ve Debezium Tabanlı
Gerçek Zamanlı Otel Verisi Entegrasyonu

Hazırlayanlar:

Furkan Adığuzel - 170423505

Muhammed Berat Öner - 171421009

Enes Bulut - 170424507

Giriş - Rapor Senaryosu

Bu projede, farklı oteller tarafından kullanılan bağımsız uygulamaların kendi veritabanları üzerinde ürettiği verilerin, merkezi bir tavan şirket veri ambarında toplanması senaryosu ele alınmıştır. Her otel, kendi operasyonel süreçlerine uygun olarak tasarlanmış uygulamalar ve bu uygulamalara ait ilişkisel veritabanları üzerinden rezervasyon, müşteri, oda ve ödeme gibi kritik iş verilerini üretmektedir.

Çalışmanın ilk aşamasında, otel uygulamalarında kullanılan mevcut veritabanı tabloları ve bu tabloların mimarisi incelenmiş, ilişkisel modelleme prensiplerine uygun bir tablo yapısı oluşturulmuştur. Bu kapsamında, her bir iş nesnesi için birincil anahtarlar tanımlanmış, tablolar arası ilişkiler yabancı anahtarlar ile kurgulanarak veri bütünlüğü sağlanmıştır.

İkinci aşamada ise, farklı otel uygulamalarında üretilen bu verilerin, merkezi veri ambarına **gerçek zamanlı ve güvenilir** bir şekilde aktarılabilmesi amacıyla modern bir veri entegrasyon mimarisi tasarlanmıştır. Bu mimaride, Apache Kafka mesajlaşma altyapısı temel alınmış; kaynak veritabanlarındaki değişikliklerin yakalanması için Debezium Change Data Capture (CDC) yaklaşımı kullanılmıştır. Kafka Connect ve Confluent bileşenleri aracılığıyla, veritabanlarında gerçekleşen ekleme, güncelleme ve silme işlemleri anlık olarak izlenmiş ve hedef veri ambarına aktarılabilir hâle getirilmiştir.

Bu raporun devamında sırasıyla; öncelikle otel uygulamalarına ait veritabanı mimarisi ve tablolar detaylandırılacak, ardından kurulan Kafka–Debezium–Confluent tabanlı CDC mimarisi açıklanacak ve son olarak bu mimarinin merkezi veri ambarı senaryosundaki katkıları değerlendirilecektir.

Uygulamalara ait Veritabanı Mimarisi ve Tabloların Detaylandırılması - Örnek DML Sorguları

(Veritabanı diagram'ı .zip dosyasında gönderilmiştir, ekran görüntüsü rapora eklenecektir.)

Bu bölümde, otel rezervasyon senaryosu için tasarlanan veritabanı mimarisi ve tabloların yapısı sunulmaktadır. Veritabanı, ilişkisel model esas alınarak oluşturulmuş olup tablolar arasında birincil anahtar (Primary Key) ve yabancı anahtar (Foreign Key) ilişkileri tanımlanmıştır. Böylece veri bütünlüğü ve tutarlılığı sağlanmıştır.

Hotels Tablosu

Hotels tablosu, sistemde yer alan otellere ait temel bilgileri tutmak amacıyla tasarlanmıştır.

Her otel kaydı benzersiz bir **hotel_id** değeri ile tanımlanmakta olup, otelin adı, bulunduğu şehir ve yıldız bilgileri bu tabloda saklanmaktadır.

1. tablo DDL - DML sorgusu ve ekran görüntüsü:

```
1 ✓ CREATE TABLE hotels (
2     hotel_id SERIAL PRIMARY KEY,
3     hotel_name VARCHAR(100) NOT NULL,
4     city VARCHAR(50) NOT NULL,
5     star_rating INTEGER CHECK (star_rating BETWEEN 1 AND 5)
6 );
```



```
[2025-12-18 22:14:16] completed in 10 ms
sourcedb.public> CREATE TABLE hotels (
    hotel_id SERIAL PRIMARY KEY,
    hotel_name VARCHAR(100) NOT NULL,
    city VARCHAR(50) NOT NULL,
    star_rating INTEGER CHECK (star_rating BETWEEN 1 AND 5)
)
[2025-12-18 22:14:17] completed in 6 ms
```

```
1 ✓ INSERT INTO hotels (hotel_name, city, star_rating) VALUES
2   ( hotel_name 'Grand Palace', city 'Istanbul', star_rating 5),
3   ( hotel_name 'Sea View Resort', city 'Antalya', star_rating 5),
4   ( hotel_name 'Mountain Lodge', city 'Bursa', star_rating 4),
5   ( hotel_name 'City Center Hotel', city 'Ankara', star_rating 4),
6   ( hotel_name 'Sunshine Hotel', city 'Izmir', star_rating 3),
7   ( hotel_name 'Royal Stay', city 'Istanbul', star_rating 5),
8   ( hotel_name 'Blue Bay Hotel', city 'Antalya', star_rating 4),
9   ( hotel_name 'Urban Inn', city 'Eskisehir', star_rating 3),
10  ( hotel_name 'Green Park Hotel', city 'Bolu', star_rating 4),
11  ( hotel_name 'Historic House', city 'Safranbolu', star_rating 3);
```

15 ✓ select * from hotels

Output sourcedb.public.hotels ×

Tx: Auto DDL

	hotel_id	hotel_name	city	star_rating
1	1	Grand Palace	Istanbul	5
2	2	Sea View Resort	Antalya	5
3	3	Mountain Lodge	Bursa	4
4	4	City Center Hotel	Ankara	4
5	5	Sunshine Hotel	Izmir	3
6	6	Royal Stay	Istanbul	5
7	7	Blue Bay Hotel	Antalya	4
8	8	Urban Inn	Eskisehir	3
9	9	Green Park Hotel	Bolu	4
10	10	Historic House	Safranbolu	3

Customers Tablosu

Customers tablosu, rezervasyon yapan müşterilere ait kimlik ve iletişim bilgilerini içermektedir.

Tablo, her müşteri için benzersiz bir **customer_id** birincil anahtarına sahiptir. E-posta alanı tekil (unique) olacak şekilde tasarılanarak mükerrer kayıtların önüne geçilmiştir.

2. tablo DDL - DML sorgusu ve ekran görüntüsü:

```
1 ✓ CREATE TABLE customers (
2     customer_id SERIAL PRIMARY KEY,
3     first_name VARCHAR(50) NOT NULL,
4     last_name VARCHAR(50) NOT NULL,
5     email VARCHAR(100) UNIQUE NOT NULL
6 );
7
```

```
1 ✓ INSERT INTO customers (first_name, last_name, email) VALUES
2     ('Ahmet', 'Yilmaz', 'ahmet.yilmaz@mail.com'),
3     ('Mehmet', 'Kaya', 'mehmet.kaya@mail.com'),
4     ('Ayse', 'Demir', 'ayse.demir@mail.com'),
5     ('Fatma', 'Celik', 'fatma.celik@mail.com'),
6     ('Ali', 'Sahin', 'ali.sahin@mail.com'),
7     ('Zeynep', 'Kara', 'zeynep.kara@mail.com'),
8     ('Murat', 'Aslan', 'murat.aslan@mail.com'),
9     ('Elif', 'Aydin', 'elif.aydin@mail.com'),
10    ('Emre', 'Koc', 'emre.koc@mail.com'),
11    ('Selin', 'Oz', 'selin.oz@mail.com');
```

The screenshot shows a database client interface with a query editor at the top containing the SQL command: `select * from customers`. Below the query editor is a table titled "sourcedb.public.customers". The table has four columns: `customer_id`, `first_name`, `last_name`, and `email`. The data consists of 10 rows, each representing a customer with a unique ID, first name, last name, and email address.

	<code>customer_id</code>	<code>first_name</code>	<code>last_name</code>	<code>email</code>
1	1	Ahmet	Yilmaz	ahmet.yilmaz@mail.com
2	2	Mehmet	Kaya	mehmet.kaya@mail.com
3	3	Ayse	Demir	ayse.demir@mail.com
4	4	Fatma	Celik	fatma.celik@mail.com
5	5	Ali	Sahin	ali.sahin@mail.com
6	6	Zeynep	Kara	zeynep.kara@mail.com
7	7	Murat	Aslan	murat.aslan@mail.com
8	8	Elif	Aydin	elif.aydin@mail.com
9	9	Emre	Koc	emre.koc@mail.com
10	10	Selin	Oz	selin.oz@mail.com

Rooms Tablosu

Rooms tablosu, otellerde bulunan odalara ait bilgileri tutmaktadır. Oda numarası, oda tipi ve gecelik ücret bilgileri bu tabloda yer almaktadır, her oda kaydı `room_id` alanı ile benzersiz olarak tanımlanmıştır.

3. tablo DDL - DML sorgusu ve ekran görüntüsü:

The screenshot shows a database client interface with a query editor containing the SQL command to create the 'rooms' table. The command is as follows:

```
CREATE TABLE rooms (
    room_id SERIAL PRIMARY KEY,
    room_number VARCHAR(10) NOT NULL,
    room_type VARCHAR(30) NOT NULL,
    price_per_night NUMERIC(10,2) NOT NULL
);
```

```
1 ✓ INSERT INTO rooms (room_number, room_type, price_per_night) VALUES
2   ( room_number '101', room_type 'Single', price_per_night 1500),
3   ( room_number '102', room_type 'Double', price_per_night 2200),
4   ( room_number '103', room_type 'Suite', price_per_night 4500),
5   ( room_number '201', room_type 'Single', price_per_night 1600),
6   ( room_number '202', room_type 'Double', price_per_night 2300),
7   ( room_number '203', room_type 'Suite', price_per_night 4800),
8   ( room_number '301', room_type 'Single', price_per_night 1700),
9   ( room_number '302', room_type 'Double', price_per_night 2400),
10  ( room_number '303', room_type 'Suite', price_per_night 5000),
11  ( room_number '401', room_type 'Family', price_per_night 5500);
12
```

```
1 ✓ select * from rooms
```

	room_id	room_number	room_type	price_per_night
1	1	101	Single	1500.00
2	2	102	Double	2200.00
3	3	103	Suite	4500.00
4	4	201	Single	1600.00
5	5	202	Double	2300.00
6	6	203	Suite	4800.00
7	7	301	Single	1700.00
8	8	302	Double	2400.00
9	9	303	Suite	5000.00
10	10	401	Family	5500.00

Payments Tablosu

Payments tablosu, rezervasyonlara ait ödeme bilgilerini saklamak için oluşturulmuştur.

Ödeme yöntemi, ödeme durumu ve ödeme tarihi gibi alanlar bu tabloda tutulmaktadır. Her ödeme işlemi **payment_id** birincil anahtarı ile temsil edilmektedir.

4. tablo DDL - DML sorgusu ve ekran görüntüsü:

```
1 ✓ CREATE TABLE payments (
2     payment_id SERIAL PRIMARY KEY,
3     payment_method VARCHAR(30) NOT NULL,
4     payment_status VARCHAR(30) NOT NULL,
5     payment_date DATE NOT NULL
6 );
```

```
1 ✓ INSERT INTO payments (payment_method, payment_status, payment_date) VALUES
2     ( 'Credit Card', 'Completed', '2025-01-05' ),
3     ( 'Credit Card', 'Completed', '2025-01-06' ),
4     ( 'Cash', 'Completed', '2025-01-07' ),
5     ( 'Bank Transfer', 'Completed', '2025-01-08' ),
6     ( 'Credit Card', 'Completed', '2025-01-09' ),
7     ( 'Cash', 'Completed', '2025-01-10' ),
8     ( 'Credit Card', 'Completed', '2025-01-11' ),
9     ( 'Bank Transfer', 'Completed', '2025-01-12' ),
10    ( 'Credit Card', 'Completed', '2025-01-13' ),
11    ( 'Cash', 'Completed', '2025-01-14' );
12
```

The screenshot shows a PostgreSQL terminal window with the following content:

```

1 ✓ select * from payments
2

```

The output pane displays the results of the query:

	payment_id	payment_method	payment_status	payment_date
1	1	Credit Card	Completed	2025-01-05
2	2	Credit Card	Completed	2025-01-06
3	3	Cash	Completed	2025-01-07
4	4	Bank Transfer	Completed	2025-01-08
5	5	Credit Card	Completed	2025-01-09
6	6	Cash	Completed	2025-01-10
7	7	Credit Card	Completed	2025-01-11
8	8	Bank Transfer	Completed	2025-01-12
9	9	Credit Card	Completed	2025-01-13
10	10	Cash	Completed	2025-01-14

Reservations Tablosu

Reservations tablosu, veritabanındaki ana ve merkezi tablo olarak tasarlanmıştır. Bu tablo; **Hotels**, **Customers**, **Rooms** ve **Payments** tabloları ile yabancı anahtar ilişkileri kurarak rezervasyon işlemlerini temsil etmektedir. Check-in ve check-out tarihleri ile toplam rezervasyon tutarı bu tabloda saklanmaktadır.

5. tablo DDL - DML sorgusu ve ekran görüntüsü:

```

1 ✓ CREATE TABLE reservations (
2     reservation_id SERIAL PRIMARY KEY,
3     hotel_id INTEGER NOT NULL,
4     customer_id INTEGER NOT NULL,
5     room_id INTEGER NOT NULL,
6     payment_id INTEGER NOT NULL,
7     check_in_date DATE NOT NULL,
8     check_out_date DATE NOT NULL,
9     total_price NUMERIC(12,2) NOT NULL,
10    CONSTRAINT fk_reservation_hotel
11        FOREIGN KEY (hotel_id) REFERENCES hotels (hotel_id),
12    CONSTRAINT fk_reservation_customer
13        FOREIGN KEY (customer_id) REFERENCES customers (customer_id),
14    CONSTRAINT fk_reservation_room
15        FOREIGN KEY (room_id) REFERENCES rooms (room_id),
16    CONSTRAINT fk_reservation_payment
17        FOREIGN KEY (payment_id) REFERENCES payments (payment_id)
18 );

```

```

1 ✓ INSERT INTO reservations (
2     hotel_id, customer_id, room_id, payment_id,
3     check_in_date, check_out_date, total_price
4 ) VALUES
5     ( hotel_id 1, customer_id 1, room_id 1, payment_id 1, check_in_date '2025-02-01', check_out_date '2025-02-05', total_price 6000),
6     ( hotel_id 2, customer_id 2, room_id 2, payment_id 2, check_in_date '2025-02-03', check_out_date '2025-02-06', total_price 6600),
7     ( hotel_id 3, customer_id 3, room_id 3, payment_id 3, check_in_date '2025-02-05', check_out_date '2025-02-08', total_price 13500),
8     ( hotel_id 4, customer_id 4, room_id 4, payment_id 4, check_in_date '2025-02-07', check_out_date '2025-02-10', total_price 4800),
9     ( hotel_id 5, customer_id 5, room_id 5, payment_id 5, check_in_date '2025-02-09', check_out_date '2025-02-12', total_price 6900),
10    ( hotel_id 6, customer_id 6, room_id 6, payment_id 6, check_in_date '2025-02-11', check_out_date '2025-02-14', total_price 14400),
11    ( hotel_id 7, customer_id 7, room_id 7, payment_id 7, check_in_date '2025-02-13', check_out_date '2025-02-15', total_price 3400),
12    ( hotel_id 8, customer_id 8, room_id 8, payment_id 8, check_in_date '2025-02-15', check_out_date '2025-02-18', total_price 7200),
13    ( hotel_id 9, customer_id 9, room_id 9, payment_id 9, check_in_date '2025-02-17', check_out_date '2025-02-20', total_price 15000),
14    ( hotel_id 10, customer_id 10, room_id 10, payment_id 10, check_in_date '2025-02-19', check_out_date '2025-02-22', total_price 16500);

```

1 ✓ select * from reservations

Output sourcedb.public.reservations

reservation_id	hotel_id	customer_id	room_id	payment_id	check_in_date	check_out_date	total_price
1	1	1	1	1	2025-02-01	2025-02-05	6000
2	2	2	2	2	2025-02-03	2025-02-06	6600
3	3	3	3	3	2025-02-05	2025-02-08	13500
4	4	4	4	4	2025-02-07	2025-02-10	4800
5	5	5	5	5	2025-02-09	2025-02-12	6900
6	6	6	6	6	2025-02-11	2025-02-14	14400
7	7	7	7	7	2025-02-13	2025-02-15	3400
8	8	8	8	8	2025-02-15	2025-02-18	7200
9	9	9	9	9	2025-02-17	2025-02-20	15000
10	10	10	10	10	2025-02-19	2025-02-22	16500

DML Sorgusu Örnekleri

Belirli bir şehirdeki oteller için yapılan rezervasyonları listeler:

```
1 ✓ SELECT
2     h.hotel_name,
3     h.city,
4     r.reservation_id,
5     r.check_in_date,
6     r.check_out_date,
7     r.total_price
8 FROM reservations r
9 JOIN hotels h 1..n<->1: ON r.hotel_id = h.hotel_id
10 WHERE h.city = 'Istanbul';
```

Output Result 21 ×

hotel_name	city	reservation_id	check_in_date	check_out_date	total_price
Grand Palace	Istanbul	1	2025-02-01	2025-02-05	6000.00
Royal Stay	Istanbul	6	2025-02-11	2025-02-14	14400.00

Her müşterinin hangi otelde, hangi tarihler arasında kaldığını gösterir:

```
1 ✓ SELECT
2     c.first_name,
3     c.last_name,
4     h.hotel_name,
5     r.check_in_date,
6     r.check_out_date
7 FROM reservations r
8 JOIN customers c 1..n<->1: ON r.customer_id = c.customer_id
9 JOIN hotels h 1..n<->1: ON r.hotel_id = h.hotel_id;
```

Output Result 22 ×

first_name	last_name	hotel_name	check_in_date	check_out_date
Ahmet	Yilmaz	Grand Palace	2025-02-01	2025-02-05
Mehmet	Kaya	Sea View Resort	2025-02-03	2025-02-06
Ayse	Demir	Mountain Lodge	2025-02-05	2025-02-08
Fatma	Celik	City Center Hotel	2025-02-07	2025-02-10
Ali	Sahin	Sunshine Hotel	2025-02-09	2025-02-12
Zeynep	Kara	Royal Stay	2025-02-11	2025-02-14
Murat	Aslan	Blue Bay Hotel	2025-02-13	2025-02-15
Elif	Aydin	Urban Inn	2025-02-15	2025-02-18
Emre	Koc	Green Park Hotel	2025-02-17	2025-02-20
Selin	Oz	Historic House	2025-02-19	2025-02-22

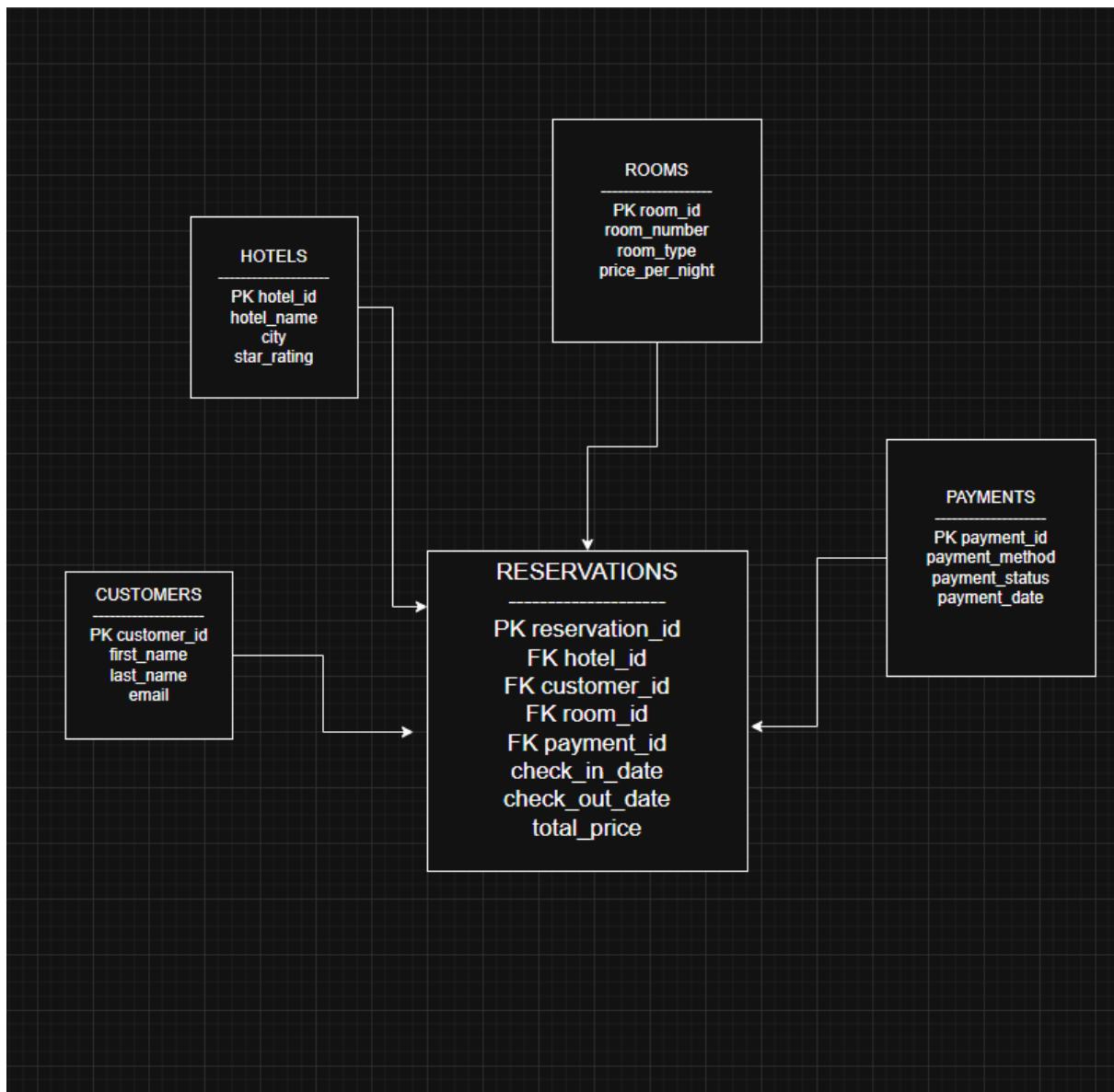
Her otelin toplam rezervasyon gelirini hesaplar:

```
1 ✓ SELECT
2     h.hotel_name,
3     COUNT(r.reservation_id) AS reservation_count,
4     SUM(r.total_price) AS total_revenue
5 FROM reservations r
6 JOIN hotels h 1..n<->1: ON r.hotel_id = h.hotel_id
7 GROUP BY h.hotel_name
8 ORDER BY total_revenue DESC;
```

Output Result 25 ×

	hotel_name	reservation_count	total_revenue
1	Historic House	1	16500
2	Green Park Hotel	1	15000
3	Royal Stay	1	14400
4	Mountain Lodge	1	13500
5	Urban Inn	1	7200
6	Sunshine Hotel	1	6900
7	Sea View Resort	1	6600
8	Grand Palace	1	6000
9	City Center Hotel	1	4800
10	Blue Bay Hotel	1	3400

Veritabanı Mimarisi ER Diagram'ı



1. Projenin Amacı

Bu projede, bir **otel rezervasyon sistemlerinin tek veritabanında toplanması**, **oteller zinciri sahibi şirketin veri ambarı tasarımında kullanılabilirlik** senaryosu üzerinden, **PostgreSQL kaynak veritabanında (Source DB)** oluşan kayıtların **Debezium + Apache Kafka** kullanılarak **gerçek zamanlı (CDC – Change Data Capture)** biçimde **hedef PostgreSQL veritabanına (Target DB)** aktarılması amaçlanmıştır.

Proje, modern veri mimarilerinde yaygın olarak kullanılan:

- **Event-driven**
- **Log-based CDC**
- **Loose coupling (gevşek bağlı mimari)**

yaklaşımlarını uygulamalı olarak göstermektedir.

2. Kullanılan Teknolojiler

Bileşen	Açıklama
PostgreSQL 16	Kaynak ve hedef veritabanı
Debezium	PostgreSQL WAL log'larını okuyarak CDC sağlayan source connector
Apache Kafka	Değişiklik olaylarının (event) taşıdığı mesajlaşma altyapısı

Kafka Connect	Source ve Sink connector'ların çalıştığı platform
Confluent JDBC Sink Connector	Kafka'daki event'leri hedef PostgreSQL'e yazmak için
Docker & Docker Compose	Tüm ortamın izole ve taşınabilir şekilde çalıştırılması

3. Sistem Mimarisi (Genel Bakış)

Akış aşağıdaki şekilde çalışmaktadır:

1. **Source PostgreSQL** üzerinde hotel_booking_events tablosuna veri eklenir.
2. PostgreSQL, bu değişiklikleri **WAL (Write-Ahead Log)** üzerine yazar.
3. **Debezium PostgreSQL Source Connector**, WAL log'larını okuyarak değişiklikleri **event** haline getirir.
4. Bu event'ler **Apache Kafka topic**'ine yazılır.
5. **Confluent JDBC Sink Connector**, Kafka topic'ini dinleyerek gelen verileri alır.
6. Veriler **Target PostgreSQL** üzerindeki hotel_booking_events tablosuna aktarılır.

Bu süreçte:

- Kaynak ve hedef sistemler **birbirinden bağımsızdır**
- Veri akışı **asenkron ve gerçek zamanlıdır**
- Uygulama tarafında ek kod yazmaya gerek yoktur

4. Kullanılan CDC Teknolojileri ve Connector Mimarisi

Bu projede PostgreSQL veritabanları arasında gerçek zamanlı veri aktarımı sağlamak amacıyla **Change Data Capture (CDC)** yaklaşımı kullanılmıştır. CDC mekanizması, kaynak veritabanında gerçekleşen tüm değişikliklerin güvenilir ve düşük maliyetli şekilde hedef sistemlere aktarılmasını sağlar.

Bu kapsamda aşağıdaki teknolojiler tercih edilmiştir:

- **Debezium PostgreSQL Source Connector**
- **Apache Kafka**
- **Confluent JDBC Sink Connector**
- **PostgreSQL (Source & Target)**

5. Debezium PostgreSQL Source Connector

Debezium PostgreSQL Source Connector, PostgreSQL veritabanında gerçekleşen değişiklikleri **WAL (Write-Ahead Log)** üzerinden izleyerek Kafka topic'lerine event olarak aktaran bir kaynaktır.

Source Connector'ın Temel Özellikleri

- PostgreSQL logical replication altyapısını kullanır
- WAL üzerinden okuma yaptığı için uygulama performansına ek yük bindirmez
- Snapshot ve streaming modlarını destekler
- Event'leri Kafka topic'leri üzerinden publish eder
- ACID özelliklerini koruyarak veri tutarlılığı sağlar

6. Source Connector Konfigürasyonu

Aşağıda projede kullanılan Debezium PostgreSQL Source Connector konfigürasyon dosyası yer almaktadır:

```
1  {
2      "name": "pg-source-hotel-events",
3      "config": {
4          "connector.class": "io.debezium.connector.postgresql.PostgresConnector",
5
6          "database.hostname": "postgres_source",
7          "database.port": "5432",
8          "database.user": "source_user",
9          "database.password": "source_pass",
10         "database.dbname": "sourcedb",
11
12         "plugin.name": "pgoutput",
13         "slot.name": "debezium_slot",
14         "publication.autocreate.mode": "filtered",
15
16         "topic.prefix": "pgdemo",
17         "table.include.list": "public.hotel_booking_events",
18
19         "snapshot.mode": "initial",
20
21         "tombstones.on.delete": "false",
22
23         "key.converter": "org.apache.kafka.connect.json.JsonConverter",
24         "value.converter": "org.apache.kafka.connect.json.JsonConverter",
25         "key.converter.schemas.enable": "false",
26         "value.converter.schemas.enable": "false"
27     }
28 }
```

7. Source Connector Parametrelerinin Açıklanması

Parametre	Açıklama
connector.class	Kullanılan Debezium PostgreSQL connector sınıfı
database.hostname	Kaynak PostgreSQL sunucusunun adresi
database.port	PostgreSQL port bilgisi
database.user	CDC için yetkilendirilmiş kullanıcı
database.password	Kullanıcı şifresi
database dbname	İzlenen veritabanı adı
plugin.name	PostgreSQL logical decoding plugin'i
slot.name	Logical replication slot adı
publication.autocreate.mode	İzlenen tablolar için publication oluşturma modu
topic.prefix	Kafka topic'leri için ön ek

table.include.list	CDC kapsamına alınan tablo
snapshot.mode	İlk çalışmada snapshot alım davranışı
tombstones.on.delete	Delete işlemlerinde tombstone üretimi
key/value.converter	Kafka mesajlarının serileştirme formatı

7. Apache Kafka'nın Rolü

Apache Kafka, bu mimaride **event streaming platformu** olarak görev yapmaktadır.

Kafka:

- Kaynak sistemden gelen değişiklikleri güvenli şekilde saklar
- Source ve Sink sistemleri birbirinden bağımsız hale getirir
- Yüksek hacimli veri akışına dayanıklıdır
- Event'lerin sıralı ve kalıcı şekilde tutulmasını sağlar

Bu sayede veri kaybı olmadan asenkron bir aktarım sağlanmaktadır.

9. Confluent JDBC Sink Connector

Confluent JDBC Sink Connector, Kafka topic'lerinde bulunan event'leri okuyarak hedef PostgreSQL veritabanına yazan bileşendir.

Sink Connector'ın Temel Özellikleri

- Kafka topic'lerini sürekli dinler
- Event'leri JDBC üzerinden hedef veritabanına aktarır
- Insert / Update işlemlerini otomatik yönetir
- UPSERT mantığı ile veri tutarlılığı sağlar
- Birden fazla veritabanını destekler

10. Sink Connector Konfigürasyonu

Projede kullanılan JDBC Sink Connector yapılandırması aşağıdaki gibidir:

```
1  {
2      "name": "pg-sink-hotel-booking-events",
3      "config": {
4          "connector.class": "io.confluent.connect.jdbc.JdbcSinkConnector",
5
6          "connection.url": "jdbc:postgresql://postgres_target:5432/targetdb",
7          "connection.user": "target_user",
8          "connection.password": "target_pass",
9
10         "topics": "pgdemo.public.hotel_booking_events",
11
12         "insert.mode": "upsert",
13         "pk.mode": "record_value",
14         "pk.fields": "event_id",
15
16         "auto.create": "true",
17         "auto.evolve": "true",
18
19         "batch.size": "3000",
20
21         "key.converter": "org.apache.kafka.connect.json.JsonConverter",
22         "value.converter": "org.apache.kafka.connect.json.JsonConverter",
23         "key.converter.schemas.enable": "false",
24         "value.converter.schemas.enable": "false"
25     }
26 }
27 }
```

11. Sink Connector Parametrelerinin Açıklanması

Parametre	Açıklama
connector.class	Confluent JDBC Sink Connector sınıfı

connection.url	Hedef PostgreSQL JDBC bağlantı adresi
connection.user	Hedef veritabanı kullanıcı adı
connection.password	Hedef veritabanı şifresi
topics	Dinlenen Kafka topic'i
insert.mode	Veri yazma stratejisi
pk.mode	Primary key alma yöntemi
pk.fields	Primary key alanı
auto.create	Hedef tabloların otomatik oluşturulması
auto.evolve	Şema değişikliklerinin otomatik uygulanması
batch.size	Toplu insert/update boyutu

12. Genel Değerlendirme

Bu proje kapsamında, farklı oteller tarafından kullanılan operasyonel veritabanlarında üretilen verilerin, merkezi bir veri ambarı yapısına **gerçek zamanlı ve güvenilir** bir şekilde aktarılabilmesini amaçlayan modern bir veri entegrasyon mimarisi tasarlanmış ve başarıyla uygulanmıştır.

Kurulan yapı ile birlikte, **PostgreSQL → PostgreSQL** veri aktarımı **Change Data Capture (CDC)** yaklaşımı kullanılarak sağlanmıştır. Kaynak veritabanında gerçekleşen **insert, update ve delete** işlemleri, uygulama katmanına herhangi bir ek yük bindirilmeden Debezium aracılığıyla yakalanmış ve Kafka altyapısı üzerinden hedef sisteme aktarılmıştır. Bu sayede, verilerin anlık olarak izlenebilmesi ve hedef sistemde güncel tutulması mümkün hale gelmiştir.

Debezium ve Confluent Kafka Connect bileşenleri kullanılarak oluşturulan CDC mimarisi, sistemin **ölçeklenebilir ve genişletilebilir** olmasını mümkün kılmıştır. Yeni tabloların, yeni kaynak sistemlerin veya yeni hedeflerin eklenmesi; mevcut yapıyı bozmadan, yalnızca yeni connector tanımları yapılarak gerçekleştirilebilmektedir. Bu esneklik, mimarinin uzun vadede sürdürülebilir olmasını sağlayan kritik bir avantajdır. Sonuç olarak bu projede kurulan yapı; günümüz kurumsal veri entegrasyonlarında yaygın olarak tercih edilen, **olay tabanlı (event-driven)** ve **gerçek zamanlı** veri işleme yaklaşımlarını temel alan modern bir mimari örneği sunmaktadır. Veri tutarlılığı, performans, esneklik ve ölçeklenebilirlik gibi temel gereksinimler göz önünde bulundurulduğunda, geliştirilen çözüm hem akademik hem de endüstriyel açıdan güçlü ve uygulanabilir bir model ortaya koymaktadır.