Veri Tabanı Sistemleri ve Uygulamaları

Hafta 13 – Transaction, Trigger, Zamanlanmış olaylar

Muğla Üniversitesi
Teknoloji Fakültesi
Bilişim Sistemleri Mühendisliği

Yrd.Doç.Dr. Serkan BALLI

Transaction (İşlem)

- Transaction, bir veya daha fazla SQL ifadesinden meydana gelen tek bir işlemdir.
- Yani, SQL ifadelerinin tamamı bir bütün olarak düşünülür ve daha küçük iş parçalarına ayrılamaz.
- Ardı ardına gelen ve birbiriyle bağlantılı birden fazla işlemin tek bir işlem olarak kullanılmasını sağlar

Transaction

- Transaction, içerdiği SQL ifadelerinin tamamını gerçekleştirir veya hiçbirini gerçekleştirmez.
- İşlemlerin bir tanesi gerçekleşmezse işlemlerin hiç biri gerçekleşmemiş sayılır.

Transaction

- **Banka örneği:** Müşteri kendi hesabından başka bir hesaba 7500 TL para transferi gerçekleştiriyor.
- Bu işlem iki adımdan oluşur:
  1. Transfer yapanın hesabından 7500 TL çıkarılır
  2. Transfer yapılan diğer hesaba 7500 TL eklenir.

Transaction

- Bu işlemleri gerçekleştirecek iki SQL ifadesi:

1. UPDATE hesaplar SET Bakiye=Bakiye-7500 WHERE hesap_no=1122

2. UPDATE hesaplar SET Bakiye=Bakiye+7500 WHERE hesap_no=1155

Transaction

- Birinci işlem gerçekleştikten sonra herhangi bir sebeple ikinci işlem gerçekleşmezse ciddi sorunlar oluşur.
- Gönderenin hesabından 7500 TL düşer ama alıcının hesabına bu para aktarılmaz.
- Bu tür sorunları önlemek için transaction yapısı kullanılır.

Transaction

- Transaction her iki işlemi de tek bir işlem gibi ele alır.
- Birisi gerçekleşmezse diğer işlem de yok sayılır. Yani gerçekleşen işlemler geri alınır (rollback).
- Eğer işlemlerin tamamı sorunsuz gerçekleşirse tüm işlemler kalıcı (commit) hale gelir.

Transaction

- Transaction yapıları transaction logları adı verilen yapıları kullanır.
- Bu yöntemde transaction başladıktan sonra ilgili veriler diskten geçici belleğe alınır.
- İstenilen değişiklikler geçici bellekteki veriler üzerinde yapılır
- Daha sonra yapılan değişikliklerin aynısını içeren transaction logları diske yazılır.
- İlgili transaction sorunsuz gerçekleşirse disk üzerindeki veriler de güncellenir ve transaction işlemi biter.

Transaction

- Transaction işlemi başarılı ise COMMIT ifadesi ile değişiklikler kalıcı hale getirilir.
- Eğer işlemler esnasında bir sorun olmuşsa ROLLBACK ifadesi ile veriler ilk baştaki değişiklik yapılmamış haline geri döner.

Transaction

- Transaction işlemi START TRANSACTION ifadesi ile başlar

START TRANSACTION
Ekleme, silme ve güncellemeleri yap
Hata oluştuysa **ROLLBACK**
Hata oluşmadıysa **COMMIT**

Transaction

- MySQL çalıştırılan sorguları otomatik olarak COMMIT eder
- Bunu kapatmak için

SET Auto_Commit=0 yapılır.

- Kapalı olduğu durumda manuel olarak COMMIT yapılması gerekir. Yapılmazsa gerçekleşen işlemler kalıcı olmaz.

Transaction (Örnek-1)

```
CREATE TABLE stok(
id INT(11) Primary key,
ad Varchar(25),
adet Smallint(3)
)
INSERT INTO stok (id, ad, adet)
VALUES (1, 'Lenovo', 125), (2, 'Sony',
432);
```

Transaction (Örnek-1)

```
CREATE TABLE satis_bilgileri(
id INT(11) Primary key,
adet Smallint(3),
Tarih timestamp
)
INSERT INTO satis_bilgileri (id, adet)
VALUES (1,3), (2, 1);
```

Transaction (Örnek-1)

Lenovo marka üründen 3 adet satılıyor

START TRANSACTION;

UPDATE satis_bilgileri SET adet = adet+3 WHERE id = 1;

UPDATE stok SET adet = adet-3 WHERE id = 1;

COMMIT;

Transaction (Örnek-2)

```
CREATE TABLE ogrenciler (
id INT PRIMARY KEY AUTO_INCREMENT,
ad VARCHAR(8)
);
```

Transaction (Örnek-2)

INSERT INTO ogrenciler (ad) VALUES ('Ayşe'), ('Fatma');

START TRANSACTION;

UPDATE ogrenciler SET ad='Defne' WHERE id = 1;

UPDATE ogrenciler SET ad='Elif' WHERE id = 2;

ROLLBACK

-> **Hiçbir işlem gerçekleşmez**

Transaction (Örnek-2)

```
INSERT INTO ogrenciler (ad) VALUES
('Ayşe'), ('Fatma');

START TRANSACTION;
UPDATE ogrenciler SET ad='Defne' WHERE id = 1;
UPDATE ogrenciler SET ad='Elif' WHERE id = 2;
COMMIT
```

->**İşlem gerçekleşir**

PHP ile Transaction Örneği

- **Banka örneği:** Müşteri kendi hesabından başka bir hesaba 7500 TL para transferi gerçekleştiriyor.

1. UPDATE hesaplar
   SET Bakiye=Bakiye-7500
   WHERE hesap_no=1122
2. UPDATE hesaplar
   SET Bakiye=Bakiye+7500
   WHERE hesap_no=1155

PHP ile Transaction Örneği

```
<?php
$mysqli = new mysqli("localhost", "kullanici", "parola", "veritabani");

//transaction basliyor
/* autocommit to kapatılıyor */
$mysqli->autocommit(FALSE);

//sorgular hazirlaniyor
$sorgu1 = $mysqli->query("UPDATE hesaplar SET Bakiye=Bakiye-7500 WHERE hesap_no=1122");
$sorgu2 = $mysqli->query("UPDATE hesaplar SET Bakiye=Bakiye+7500 WHERE hesap_no=1155");

if (!$sorgu1 or !$sorgu2 )
{
        //hata var, geri aliyoruz
        $mysqli->rollback();
}
else
{
        //hata yok, islem tamamlaniyor
        $mysqli->commit();
}
$mysqli->close();
?>
```

Transaction (RollBack)

Bazı SQL komutları (DDL) rollback ile geri alınamaz:

- CREATE DATABASE,
- CREATE TABLE,
- DROP DATABASE,
- DROP TABLE
- ALTER TABLE

Trigger (Tetikleyici)

- Veritabanı üzerinde herhangi bir işlem gerçekleştirildiğinde otomatik olarak başka bir işlemin yapılması isteniyorsa **Trigger** kullanılır.
- Yani bir işlem başka bir işlemi tetikler.
- Tetikleyiciler; INSERT, UPDATE ve DELETE ifadeleri için kullanılabilir.
- SELECT ifadesi tablo üzerinde bir değişiklik yapmadığı için SELECT için tetikleyici tanımlanmaz.

Trigger (Tetikleyici)

- Trigger, ekleme (INSERT), güncelleme (UPDATE) veya silme (DELETE) olayları öncesinde (BEFORE) veya sonrasında (AFTER) gerçekleşmektedir.

Trigger (Tetikleyici)

- MYSQL tek bir tablo üzerinde maksimum 6 adet tetikleyiciyi destekler:
  - Before Insert
  - After Insert
  - Before Update
  - After Update
  - Before Delete
  - After Delete

Trigger oluşturma

CREATE TRIGGER

trigger_adı trigger_zamanı trigger_olayı

ON tablo_adı **FOR EACH ROW**

trigger_tanımlaması

Trigger oluşturma (Örnek)

Yeni bir çalışan kaydı girildiği anda son güncelleme zamanını güncelleyen bir tetikleyici için:

Çalışanlar tablosu:

```
CREATE TABLE calisanlar (
calisan_id int(11) NOT NULL,
adi varchar(25) COLLATE utf8_turkish_ci DEFAULT NULL,
soyadi varchar(25) COLLATE utf8_turkish_ci DEFAULT NULL,
son_guncelleme datetime DEFAULT NULL,
PRIMARY KEY (`calisan_id`)
)
```

Trigger oluşturma (Örnek)

Yeni bir çalışan kaydı girildiği anda son güncelleme zamanını güncelleyen bir tetikleyici için:

```
CREATE TRIGGER calisan_son_guncelleme
    BEFORE INSERT ON calisanlar
    FOR EACH ROW
    SET NEW.son_guncelleme = NOW();
```

Trigger silme

DROP TRIGGER calisan_son_guncelleme;

Hata verirse:

DROP TRIGGER IF EXISTS calisan_son_guncelleme;

Trigger (Çoklu SQL)

- Eğer tetikleyici ile birden fazla SQL komutu çalıştırılacaksa

```
BEGIN
……..
END
```

bloğu kullanılır

Trigger özellikler

- View’lar tetikleyicileri desteklemez
- Tetikleyici değiştirilemez, bu nedenle tetikleyiciyi kaldırıp tekrar oluşturmak gerekir.
- Veritabanında bulunan tüm tetikliyicileri görmek için:

  SHOW triggers

Tetikleyicilerin Saklandığı Konum

- MySQL klasöründeki data klasörünün içindeki çalışılan veritabanı klasöründe depolanırlar.

- TRN ve TRG uzantılıdır.
  - before_anket_soru_girisi.TRN-→tetikleyici tanımı
  - anket_sorular.TRG--< ilişkili tablo

Trigger Örnek:

```
CREATE TABLE anket_sorular (
   Soruno int(10) primary key,
   Soru varchar(250)
   );

CREATE TABLE yenisoru (
   Soru varchar(250),
   Tarihi datetime
   );
```

Trigger (INSERT)

- Örneğin ankete her girilen soruyu ve sorunun oluşturulma tarih/saatini başka bir tabloya kaydeden bir tetikleyici oluşturmak için:

Trigger (INSERT)

```
DELIMITER |
    CREATE TRIGGER before_anket_soru_girisi
    BEFORE INSERT ON anket_sorular
    FOR EACH ROW BEGIN
        INSERT INTO yenisoru (soru, Tarihi)
    values (NEW.soru, NOW());
    END
```

Trigger (INSERT)

Ankete yeni soru ekleyelim:

```
INSERT INTO anket_sorular (Soruno, Soru) values (1, 'Türkiyenin başkenti neresidir?');
```

Trigger (UPDATE)

- Örneğin anketteki bir soru değiştirildiğinde eski soruyu başka bir tabloya kaydeden bir tetikleyici oluşturmak için:

Trigger (UPDATE)

```
CREATE TABLE eskisoru (
   Soruno int(10),
   Soru varchar(250)
   );
```

Trigger (UPDATE)

```
DELIMITER |
CREATE TRIGGER update_anket_soru_girisi
BEFORE UPDATE ON anket_sorular
FOR EACH ROW BEGIN
    INSERT INTO eskisoru (soruno, soru) values (OLD.soruno, OLD.soru);
END
```

Trigger (UPDATE)

- UPDATE anket_sorular SET soru="Almanyanın başkenti neresidir?" where soruno=1;

Trigger (DELETE)

- Örneğin anketten silinen soruların sayısını başka bir tabloya kaydeden bir tetikleyici oluşturmak için:

Trigger (DELETE)

```
CREATE TABLE silinensorular (
   adet int(10)
   );

 INSERT INTO silinensorular
 SET adet=0;
```

Trigger (DELETE)

```
DELIMITER |
CREATE TRIGGER delete_anket_soru
BEFORE DELETE ON anket_sorular
FOR EACH ROW BEGIN
    UPDATE silinensorular
    SET adet=adet+1;
END
```

Trigger (DELETE)

```
DELETE FROM anket_sorular
WHERE soruno=1;
```

Zamanlanmış olaylar **Event Scheduler**

- Bir veya birden fazla SQL ifadesinin belli bir tarihte veya tarih aralığında otomatik olarak çalıştırılmasını sağlayan olaylara zamanlanmış olaylar denir.
- Örnek: Linux ortamında CRON
- Örnek: Windows ortamında Zamanlanmış Görevler

Zamanlanmış olaylar **Event Scheduler**

- Yani bir SQL komut dizisinin (INSERT, UPDATE, DELETE, STORED PROCEDURE, TRIGGER vb. olabilir) belirli bir tarih ve saatte otomatik olarak çalıştırılmasını sağlar.

Örnek Zamanlanmış Olaylar

- Benzin fiyatlarının bu geceden itibaren güncellenmesi
- Süpermarkette bazı ürünlerin haftasonu indirime girmesi

Zamanlanmış olaylar

- MySQL ortamında bu özellik varsayılan olarak kapalı gelir.
- Durumunu görmek için

SELECT @@event_scheduler;

- Kapalı ise açmak için

```
SET GLOBAL event_scheduler = 1;
```

Zamanlanmış olaylar

Syntax:

```
CREATE
    [DEFINER = { user | CURRENT_USER }]
    EVENT
    [IF NOT EXISTS]
    event_name
    ON SCHEDULE schedule
    [ON COMPLETION [NOT] PRESERVE]
    [ENABLE | DISABLE | DISABLE ON SLAVE]
    [COMMENT 'comment']
    DO event_body;
```

Zamanlanmış olaylar

schedule:

```
    AT timestamp [+ INTERVAL interval] ...
  | EVERY interval
    [STARTS timestamp [+ INTERVAL interval] ...]
    [ENDS timestamp [+ INTERVAL interval] ...]
```

interval:

```
    quantity {YEAR | QUARTER | MONTH | DAY | HOUR | MINUTE |
              WEEK | SECOND | YEAR_MONTH | DAY_HOUR | DAY_MINUTE |
              DAY_SECOND | HOUR_MINUTE | HOUR_SECOND | MINUTE_SECOND}
```

Zamanlanmış olaylar

5 saniyede bir Zamanlar isimli tablomuza günün tarih ve
saatini ekleyen bir olay yazalım:

```
CREATE DATABASE Kronometre;

CREATE TABLE Zamanlar (zaman datetime
NOT NULL);
```

Zamanlanmış olaylar

```
CREATE EVENT zaman_guncelle
ON SCHEDULE
EVERY 5 second
COMMENT "5 sn de bir ekleme"
DO
INSERT INTO Zamanlar (zaman)
VALUES (now());
```

Zamanlanmış olaylar

SHOW EVENTS;

Silmek için:

DROP EVENT zaman_güncelle;

Zamanlanmış olaylar

Akaryakıt isimli bir veritabanında ürünler isimli bir tablomuz var.

```
CREATE TABLE `urunler` (
  `ID` int(5) NOT NULL,
  `Cinsi` varchar(20),
  `Fiyati` float DEFAULT NULL,
  PRIMARY KEY (`ID`)
)
```

Zamanlanmış olaylar

```
INSERT INTO `urunler` (`ID`, `Cinsi`, `Fiyati`) VALUES
(1, 'Benzin', 6.10), (2, 'Motorin', 5.63), (3, 'LPG', 3.35);
```

Urunler(ID, Cinsi, Fiyati)

```
+----+---------+--------+
| ID | Cinsi     | Fiyati     |
+----+---------+--------+
|  1 | Benzin  |    6.10  |
|  2 | Motorin |   5.63  |
|  3 | LPG       |   3.35 |
+----+---------+--------+
```

Zamanlanmış olaylar

Bu gece saat 00:00'dan itibaren Benzine 8 kuruş zam gelecek şekilde bir olay yazınız.

Zamanlanmış olaylar

```
CREATE EVENT benzine_zam_yap
ON SCHEDULE
AT "2018-04-25 00:00:00"
COMMENT "benzine 8 kuruş zam yap"
DO
UPDATE Urunler Set Fiyati=Fiyati+0.08
WHERE cinsi="Benzin";
```

Zamanlanmış olaylar

1 Mayıstan başlamak üzere 31 Mayısa kadar her hafta benzine 8 kuruş zam gelecek şekilde bir olay yazınız.

Zamanlanmış olaylar

```
CREATE EVENT benzine_zam_yap
ON SCHEDULE EVERY 1 WEEK
STARTS "2018-05-01 00:00:00"
ENDS "2018-05-31 00:00:00"
COMMENT " benzine 8 kuruş zam yap"
DO
UPDATE Urunler Set Fiyati=Fiyati+0.08
WHERE cinsi="Benzin";
```

Zamanlanmış olaylar

Mesajlar isimli tablodan her gün kontrol ederek 1 haftadan daha eski mesajları silmek için:

```
CREATE EVENT temizle
ON SCHEDULE every 1 day
DO
DELETE FROM mesajlar WHERE
mesajTarih <
DATE_SUB(NOW(), INTERVAL 1 WEEK);
```