



Veri Tabanı Yönetim Sistemleri

Bölüm 8– Gelişmiş Yapılar, Görünüm ve Tetikleyiciler

Dr. Öğr. Üyesi Elvan DUMAN

Bölüm 8

Gelişmiş Yapılar, Görünüm ve Tetikleyiciler

Bölüm 8 - Amaçlar



- ▶ SQL'de karar yapıları (IF, CASE) ve koşullu işlemleri kullanabilmek,
- ▶ CONCAT, SUBSTRING, LENGTH gibi string fonksiyonları ile metin işlemlerini yapabilmek
- ▶ ROUND, FLOOR, CEIL, MOD gibi sayısal fonksiyonların kullanım mantığını kavramak.
- ▶ View, Trigger ve Procedure yapılarında sık kullanılan temel fonksiyonları tanımak.
- ▶ VIEW temel yapısını ve kullanım alanlarını anlamak
- ▶ Tetikleyici temel yapısını ve kullanım alanlarını anlamak

MySQL Gelişmiş Yapılar

Değişken Tanımlama ve Atama



Procedure veya trigger içinde değişken kullanılabilir.

```
DECLARE ortalama DOUBLE;  
SET ortalama = 75;
```

MySQL Gelişmiş Yapılar



Metin (String) İşlemleri

CONCAT – Birden çok metni birleştirir

Trigger ve view içinde sık kullanılır (log yazmak, görüntü üretmek)

```
SELECT CONCAT(ad, ' ', soyad) AS tam_ad FROM ogrenci;
```

LENGTH / CHAR_LENGTH – Metnin karakter sayısını verir

```
SELECT LENGTH(ad) FROM ogrenci;
```

MySQL Gelişmiş Yapılar



Metin (String) İşlemleri

SUBSTRING – Metinden belirli kısmı alır

```
SELECT SUBSTRING(ad, 1, 3) FROM ogrenci;  
SELECT SUBSTRING('Elvan Duman', 1, 5); -- Sonuç: 'Elvan'
```

UPPER / LOWER – Metni büyük/küçük harfe çevirir

```
SELECT UPPER(ad) FROM ogrenci;
```

MySQL Gelişmiş Yapılar

Sayısal İşlemleri



ROUND– Yuvarlama yapar. Ayrıca ondalıklı kısım belirtilerek de yuvarlama yapılabilir.

```
SELECT ROUND(ortalama, 2) FROM notlar;
```

CEIL / FLOOR – Yukarı-aşağı yuvarlama

```
SELECT CEIL(89.1); -- → 90  
SELECT FLOOR(89.9); -- → 89
```

MOD – Mod alma

```
SELECT MOD(10, 3); -- → 1
```

MySQL Gelişmiş Yapılar

Tarih ve Zaman Fonksiyonları



CURRENT_DATE, NOW

```
SELECT CURRENT_DATE(); -- Sadece tarih
SELECT NOW();           -- Tarih + saat
```

DATE_ADD / DATE_SUB

```
SELECT DATE_ADD(NOW(), INTERVAL 1 MONTH);
```

DATEDIFF

```
SELECT DATEDIFF('2025-06-01', '2025-04-21'); -- → 41 gün
```

MySQL Gelişmiş Yapılar

Kullanışlı Sistem Fonksiyonları



Fonksiyon	Açıklama
UUID()	Benzersiz kimlik üretir
RAND()	Rastgele sayı üretir
LAST_INSERT_ID()	Son eklenen satırın ID'si
NOW()	Güncel tarih-saat
USER()	SQL oturumu kullanan kullanıcıyı döner

MySQL Gelişmiş Yapılar

Örnek Sorgu



```
SELECT
  ogr_no,
  CONCAT(ad, ' ', soyad) AS tam_ad,
  CASE
    WHEN ders_notu >= 90 THEN 'AA'
    WHEN ders_notu >= 80 THEN 'BA'
    ELSE 'FF'
  END AS harf_notu,
  ROUND(ders_notu * 0.05, 2) AS katkı_puani
FROM ogrenci_notlari;
```

MySQL Gelişmiş Yapılar

Karar Verme ve Koşul Yapıları



IF, ELSEIF, ELSE (sadece procedure, trigger içinde kullanılır)

- Nerede kullanılır?: BEGIN...END blokları içinde (örneğin trigger'da)

```
IF puan >= 90 THEN
    SET harf_notu = 'AA';
ELSEIF puan >= 80 THEN
    SET harf_notu = 'BA';
ELSE
    SET harf_notu = 'FF';
END IF;
```

MySQL Gelişmiş Yapılar



CASE (her yerde kullanılabilir – SELECT, UPDATE, vb.)

Esnek bir yapıdır, tablo sorgularında sık kullanılır.

```
SELECT
  CASE
    WHEN puan >= 90 THEN 'AA'
    WHEN puan >= 80 THEN 'BA'
    ELSE 'FF'
  END AS harf_notu
FROM ogrenci_notlari;
```

MySQL Gelişmiş Yapılar

IF / CASE Örneği



```
-- CASE ile
SELECT CASE WHEN puan >= 60 THEN 'Geçti' ELSE 'Kaldı' END;

-- IF ile (procedure içinde kullanılır)
IF puan >= 60 THEN
    SET durum = 'Geçti';
ELSE
    SET durum = 'Kaldı';
END IF;
```

MySQL Görünüm (VIEW)



Temel Bilgiler

- ▶ View (Görünüm), bir veya daha fazla tablodan elde edilen sanal bir tablodur.
- ▶ Fiziksel veri içermez, sadece bir SQL sorgusunun sonucunu temsil eder.
- ▶ Veriler temel tablolardan çekilir, bu nedenle güncel veriyi yansıtır.
- ▶ Kullanıcılar için veri soyutlaması sağlar ve karmaşık sorguları sadeleştirir.

MySQL Görünüm (VIEW)



View Kullanım Amaçları

- ▶ Veri soyutlama: Kullanıcılara karmaşık sorgular yerine basit görünümler sunar.
- ▶ Güvenlik: Kullanıcıların sadece yetkili oldukları sütunlara erişmesini sağlar.
- ▶ Kod tekrarını azaltma: Tekrarlayan karmaşık sorgular view ile merkezi hâle getirilir.
- ▶ Raporlama: Sık kullanılan rapor sorguları görünüm hâline getirilebilir.

MySQL Görünüm (VIEW)

Nasıl Oluşturulur?



```
CREATE VIEW gorunum_adi AS  
SELECT sutun1, sutun2  
FROM tablo_adi  
WHERE kosul;
```

```
CREATE VIEW yayinevi_istatistik AS  
SELECT Y.yadi, COUNT(Y.id) AS sayisi  
FROM yevi AS Y  
LEFT JOIN kitap AS K ON K.yevi_id=Y.id  
GROUP BY Y.id;
```

```
CREATE VIEW aktif_ogrenciler  
AS  
SELECT ad, soyad, eposta  
FROM ogrenci  
WHERE durum = 1;
```

MySQL Görünüm (VIEW)



Kullanımı

- ▶ Normal tablo gibi sorgulanabilir:
 - `SELECT * FROM aktif_ogrenciler`
- ▶ Join yapılabilir, filtrelenebilir, sıralanabilir:
 - `SELECT * FROM aktif_ogrenciler
WHERE eposta LIKE '%@gmail.com'
ORDER BY ad;`

MySQL Görünüm (VIEW)



Veri Güncelleme

Aşağıdaki kısıtlı koşullarda, VIEW üzerinden UPDATE, INSERT, DELETE işlemleri yapılabilir.

- ▶ Tek tabloya dayanıyorsa
- ▶ DISTINCT, GROUP BY, JOIN, HAVING gibi ifadeler yoksa
- ▶ Alt sorgu ya da aggregate fonksiyonlar kullanılmamışsa

Aşağıdaki durumlarda view **güncellenemez**.

- ▶ JOIN içeren view'ler
- ▶ GROUP BY, HAVING, DISTINCT kullanımı
- ▶ LIMIT, UNION, AGGREGATE fonksiyonlar (SUM, AVG gibi)

MySQL Görünüm (VIEW)



Veri Güncelleme

Aşağıdaki kısıtlı koşullarda, VIEW üzerinden UPDATE, INSERT, DELETE işlemleri yapılabilir.

- ▶ Tek tabloya dayanıyorsa
- ▶ DISTINCT, GROUP BY, JOIN, HAVING gibi ifadeler yoksa
- ▶ Alt sorgu ya da aggregate fonksiyonlar kullanılmamışsa

Aşağıdaki durumlarda view **güncellenemez**.

- ▶ JOIN içeren view'ler
- ▶ GROUP BY, HAVING, DISTINCT kullanımı
- ▶ LIMIT, UNION, AGGREGATE fonksiyonlar (SUM, AVG gibi)

▶

```
CREATE OR REPLACE VIEW aktif_ogrenciler AS  
SELECT ad, soyad FROM musteriler WHERE aktif = 1;
```

MySQL Görünüm (VIEW)



Veri Güncelleme

```
CREATE OR REPLACE VIEW aktif_ogrenciler AS  
SELECT ad, soyad FROM musteriler WHERE aktif = 1;
```

Görünüm Silme (DROP VIEW)

```
DROP aktif_ogrenciler
```

MySQL Görünüm (VIEW)



Dikkat Edilmesi Gereken Hususlar

- ▶ Performans: Her çağrıldığında SQL çalışır, önbellekleme yok.
- ▶ Karmaşık view zincirleri performansı düşürebilir.
- ▶ İndeks içermezler, sadece temel tablodaki indekslerden faydalanır.
- ▶ Veri tutarlılığı açısından güncel tablolarla eşzamanlıdır.

MySQL Tetikleyici (Trigger)



Tanım ve Bilgiler

- ▶ Trigger (Tetikleyici), bir tabloda INSERT, UPDATE veya DELETE işlemleri gerçekleştiğinde otomatik olarak çalışan önceden tanımlanmış SQL kodlarıdır.
- ▶ Olay (event) tabanlı çalışır: İlgili veri değiştiğinde tetiklenir.
- ▶ Veri bütünlüğünü korur, arka planda otomatik işlemler yapılmasını sağlar.

MySQL Tetikleyici (Trigger)



Ne amaçla kullanılır.

- ▶ Otomatik log (kayıt) tutma
- ▶ Veri üzerinde doğrulama / kısıtlama yapma
- ▶ Başka tablolarda eş zamanlı güncelleme
- ▶ İş kurallarını veri tabanı seviyesinde uygulama (veri tabanında mı uygula maliyım yoksa programla dili seviyesinde mi? Sorusunu sorarak)
 - Veri tabanında yap : Veri bütünlüğü kritikse (not null, unique vb)
 - Uygulama tarafında yap: Kurallar sık değişiyorsa, performans açısından vb maliyetliyse, iş akışına bağlı koşullar var ise (örn: sipariş onayı için önce stok kontrolü yapılması gerekiyorsa)

MySQL Tetikleyici (Trigger)

Kullanım Yapısı



```
CREATE TRIGGER tetikleyici_adı
{BEFORE | AFTER} {INSERT | UPDATE | DELETE}
ON tablo_adı
FOR EACH ROW
BEGIN
    -- çalışacak SQL kodları
END;
```

MySQL Tetikleyici (Trigger)



Kullanım Türleri

Her işlem türü için 2 tetikleme zamanı vardır, toplam 6 çeşit trigger:

İşlem Türü	BEFORE	AFTER
INSERT	BEFORE INSERT	AFTER INSERT
UPDATE	BEFORE UPDATE	AFTER UPDATE
DELETE	BEFORE DELETE	AFTER DELETE

MySQL Tetikleyici (Trigger)



Log Tetikleyicisi Örneği

```
CREATE TABLE loglar (  
    id INT AUTO_INCREMENT PRIMARY KEY,  
    mesaj TEXT,  
    zaman TIMESTAMP DEFAULT  
CURRENT_TIMESTAMP  
);
```

```
CREATE TRIGGER log_ekle  
AFTER INSERT ON musteriler  
FOR EACH ROW  
BEGIN  
    INSERT INTO loglar  
    VALUES (CONCAT('Yeni müşteri eklendi:  
' , NEW.ad));  
END;
```

MySQL Tetikleyici (Trigger)



OLD ve NEW Anahtar Kelimeleri

Trigger içinde kullanılan özel referanslardır:

Kullanım	Açıklama
NEW.sutun	Yeni değer (INSERT ve UPDATE için)
OLD.sutun	Eski değer (UPDATE ve DELETE için)

MySQL Tetikleyici (Trigger)



BEFORE vs AFTER Trigger Farkı

- ▶ BEFORE: Veri veritabanına yazılmadan önce çalışır. Veri değiştirme veya doğrulama işlemleri için kullanılır.
- ▶ AFTER: Veri yazıldıktan sonra çalışır. Log tutma veya başka tabloya işlem yapma için idealdir.

```
CREATE TRIGGER maas_kontrol
BEFORE UPDATE ON personel
FOR EACH ROW
BEGIN
    IF NEW.maas < 0 THEN
        SET NEW.maas = 0;
    END IF;
END;
```

MySQL Tetikleyici (Trigger)

Tetikleyici Silme



```
DROP TRIGGER IF EXISTS tetikleyici_adı;
```

MySQL Tetikleyici (Trigger)



Hatırlatmalar

- ▶ Her tablo için aynı türde sadece bir trigger tanımlanabilir (örneğin, bir tabloda yalnızca bir BEFORE INSERT olabilir).
- ▶ Trigger içindeki işlemler başka triggerları tetikleyebilir. Dikkatli kullanılmalı (sonsuz döngü riski!).
- ▶ Performans maliyeti olabilir, özellikle büyük verilerde.

MySQL Tetikleyici (Trigger)



Örnekler

Ogrenci tablosuna kayıt yapıldığında randevu tarihi mevcut kayıt tarihinden 1 ay sonraya atılır.

```
DELIMITER $$
```

```
CREATE TRIGGER randevu_ayarla
```

```
BEFORE INSERT ON ogrenci
```

```
FOR EACH ROW
```

```
BEGIN
```

```
    SET NEW.randevu_tarihi = DATE_ADD(NEW.kayit_tarihi, INTERVAL 1 MONTH);
```

```
END $$
```

```
DELIMITER ;
```

MySQL Tetikleyici (Trigger)

Örnekler

Not girildiği zaman harf notu otomatik hesaplayan tetikleyici.



```
CREATE TRIGGER hesapla_harf_notu
BEFORE UPDATE ON ogrenci_notlari
FOR EACH ROW
BEGIN
    IF NEW.ders_notu IS NOT NULL THEN
        IF NEW.ders_notu BETWEEN 90 AND 100 THEN
            SET NEW.harf_notu = 'AA';
        ELSEIF NEW.ders_notu BETWEEN 85 AND 89 THEN
            SET NEW.harf_notu = 'BA';
        ELSEIF NEW.ders_notu BETWEEN 80 AND 84 THEN
            SET NEW.harf_notu = 'BB';
        ELSEIF NEW.ders_notu BETWEEN 75 AND 79 THEN
            SET NEW.harf_notu = 'CB';
        ELSEIF NEW.ders_notu BETWEEN 70 AND 74 THEN
            SET NEW.harf_notu = 'CC';
        ELSEIF NEW.ders_notu BETWEEN 65 AND 69 THEN
            SET NEW.harf_notu = 'DC';
        ELSEIF NEW.ders_notu BETWEEN 60 AND 64 THEN
            SET NEW.harf_notu = 'DD';
        ELSEIF NEW.ders_notu BETWEEN 50 AND 59 THEN
            SET NEW.harf_notu = 'FD';
        ELSEIF NEW.ders_notu BETWEEN 0 AND 49 THEN
            SET NEW.harf_notu = 'FF';
        ELSE
            SET NEW.harf_notu = 'DZ'; -- örneğin devamsızlık durumu
        END IF;
    END IF;
END$$
```

Bölüm sonu- Teşekkürler.

Dr. Öğr. Üyesi Elvan DUMAN – eduman@mehmetakif.edu.tr