BMB202. Veritabanı Yönetimi

Ders 8.
Stored Procedure (Saklı Yordamlar), Trigger (Tetikleyiciler), Transactions, Yetkilendirme (Authorization)

Erdinç Uzun
NKÜ Çorlu Mühendislik Fakültesi
Bilgisayar Mühendisliği Bölümü

Dersin Planı

- Stored Procedure (Saklı Yordamlar)
- Trigger (Tetikleyiciler)
- Transactions
- Yetkilendirme (Authorization)

Stored Procedure (Saklı Yordamlar)

- Store Procedure (SP)
  - dışarıdan, bir veya birden fazla parametre alarak,
  - bu parametreler ile veritabanı üzerinde belirli işlemleri yerine getiren,
  - yaptığı işlem sonucu olarak da geriye değer döndürebilen

  yapılardır.
- Bir SP, SQL cümleleri içerir.
- Bir SP, MySQL kurulu bilgisayar üzerinde başka bir deyişle sunucu üzerinde tutulur.

SP

- SP tanım olarak, veritabanı kataloğunda saklanan hazır derlenmiş sql kod bloklarıdır.
- Stored procedure'ler birkere yazılıp derlendikten sonra artık istenildiği gibi çağrılabilir.
- Mysql, 5.0 versiyonundan sonra SP'ye destek vererek daha esnek ve güçlü bir yapıya kavuşmuştur.

Neden SP?

- SP’ler normal SQL sorguları gibi her çağrıldığında veritabanı motoru tarafından SQL sorgusunun doğruluğu (derleme işlemi) kontrol edilmez.
- SP ler bir defa oluşturulduktan sonra çağırdığımızda derlenme ihtiyacı duymadan, yapmasını istediğimiz işlemi yapar ya da işlem sonuçlarını çıktı olarak bize geri döndürür.

SP avantajları

- Saklı yordamlar, uygulama ve veritabanı sunucusu arasındaki trafiği azaltır.
  - Uzun bir sql cümleciğinin ağda bir makineden diğer bir makineye gitmesi yerine sadece saklı yordamın ismi gitmektedir.
- SP'ler veritabanı için oldukça güvenli sorgu çalıştırma yapıları oluştururlar.
  - SP bazlı kullanıcı yetkilendirmesi yapılabilir.
- SP'ler uygulama ile veritabanı sunucusu arasındaki iletişimi en aza indirmektedir.
  - Uzun SQL sorgularını sunucuya göndermek yerine SP'nin sadece ismi gitmektedir.
- SP'ler bir defa derlendikten (oluşturulduktan) sonra tekrar tekrar çağrılabilir.

MySQL SP
SP için tablo oluşturma ve Veri Ekleme

```
1  CREATE TABLE `test`.`rehber` (
2    `idrehber` INT NOT NULL AUTO_INCREMENT,
3    `ad` VARCHAR(20) NULL,
4    `soyad` VARCHAR(20) NULL,
5    PRIMARY KEY (`idrehber`));
```

```
1  INSERT INTO `test`.`rehber` (`ad`, `soyad`) VALUES ('Ali', 'Gültekin');
2  INSERT INTO `test`.`rehber` (`ad`, `soyad`) VALUES ('Feyyaz', 'Uçar');
3  INSERT INTO `test`.`rehber` (`ad`, `soyad`) VALUES ('Metin', 'Tekin');
4  INSERT INTO `test`.`rehber` (`ad`, `soyad`) VALUES ('Pascal', 'Nouma');
5  INSERT INTO `test`.`rehber` (`ad`, `soyad`) VALUES ('İlhan', 'Mansız');
```

	idrehber	ad	soyad
	1	Ali	Gültekin
	2	Feyyaz	Uçar
	3	Metin	Tekin
▶	4	Pascal	Nouma
	5	İlhan	Mansız
*	NULL	NULL	NULL

SP

```
1
2   USE `test`;
3   DROP procedure IF EXISTS `test`.`test_pro`;
4
5   DELIMITER $$
6   USE `test`$$
7   CREATE DEFINER=`root`@`localhost` PROCEDURE `test_pro`()
8   BEGIN
9
10  END$$
11
12  DELIMITER ;
13  ;
```

- Use kullanılacak veritabanı ismini belirler.
- DROP, eğer sp var ise siler ve yeni sp'yi kaydeder.
- DELIMITER $$ ifadesi saklı yordama özgü bir ifade değildir. MySQL in ayırıcı karakteri olan ; (noktalı virgül) değiştirmek için kullanılır. Bu sayede SP içerisinde birden fazla SQL sorgusu yazabiliriz. DEFINER ifadesi saklı yordamın hangi kullanıcı tarafından kullanılacağını belirlemek için kullanılır.

SP

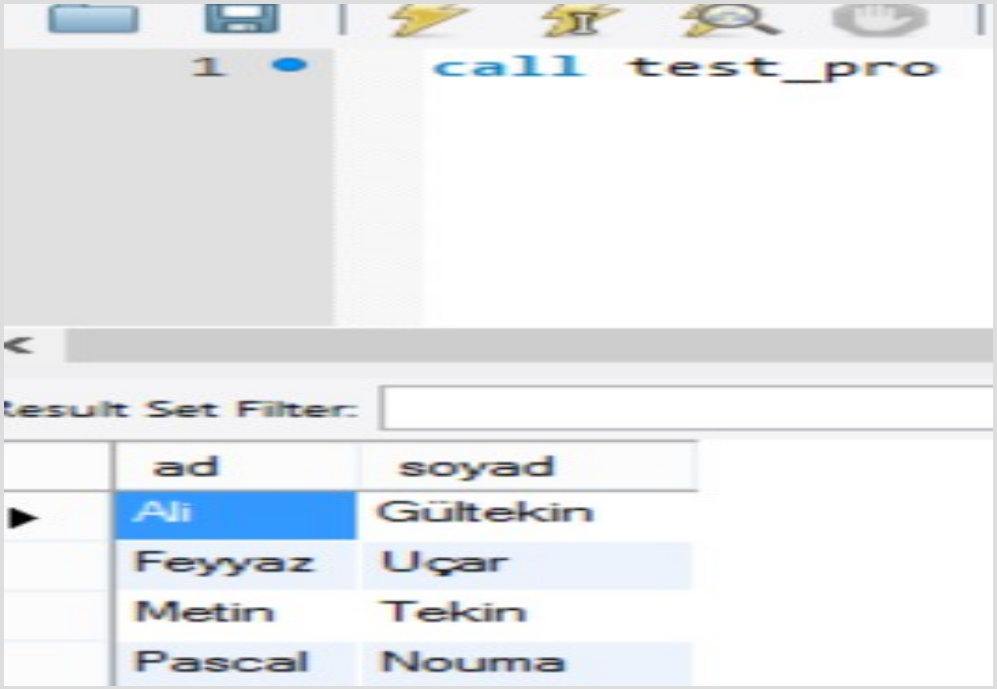
```
1
2   USE `test`;
3   DROP procedure IF EXISTS `test`.`test_pro`;
4
5   DELIMITER $$
6   USE `test`$$
7   CREATE DEFINER=`root`@`localhost` PROCEDURE `test_pro`()
8   BEGIN
9
10  END$$
11
12  DELIMITER ;
13  ;
```

- CREATE PROCEDURE ifadesi ile kullanılarak SP oluşturmaya başlanır. Crate ifadesi devamında SP ismi belirtilir. Eğer istenirse SP isminden sonra parantezler içerisinde parametreler tanımlanabilir.
- Saklı yordamın gövdesi BEGIN ve END bloğu ile belirtilir. BEGIN ve END arası saklı yordamın local (yerel) bilinirlik alanına ilişkindir. Yani BEGIN ve END arasına yazılan her sql komutu saklı yordama dahildir.
- DEFINER ifadesi saklı yordamın hangi kullanıcı tarafından kullanılacağını belirlemek için kullanılır.

İlk SP

```
5   DELIMITER $$
6
7   CREATE DEFINER=`root`@`localhost` PROCEDURE `test_pro`()
8   BEGIN
9   select ad, soyad from rehber;
10  END
```

- Sorrgumuzu BEGIN – END arasına yazıyoruz.
- Call procedure ismi ile SP'yi çağrıyoruz.



Parametreli SP'ler

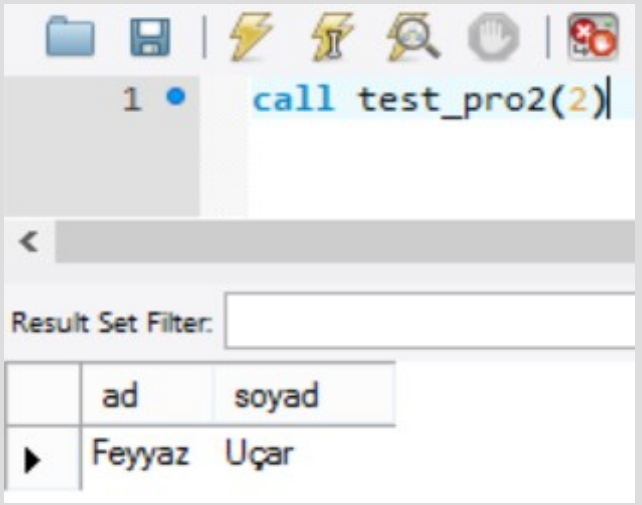
- SP'ler genellikle parametreli olurlar. Parametre ile saklı yordama
  - değer gönderebilir,
  - değer alabiliriz.
- Parametreler, SP'leri daha esnek ve kullanılabilir kılar.
- Mysql de SP parametreleri üç yöntem belirtecinden birini alabilir. Bu yöntem belirteçleri
  - IN,
  - OUT,
  - INOUT

  olabilir.

IN Yöntem Belirteci

- IN yöntem belirtecini biz parametreye sadece değer göndereceksek kullanırız.
- Yani IN yöntem belirteci ile gönderdiğimiz değer SP'nin parametre değişkenine kopyalanır ve SP'nin gövdesi içerisinde bilinir durumdadır.

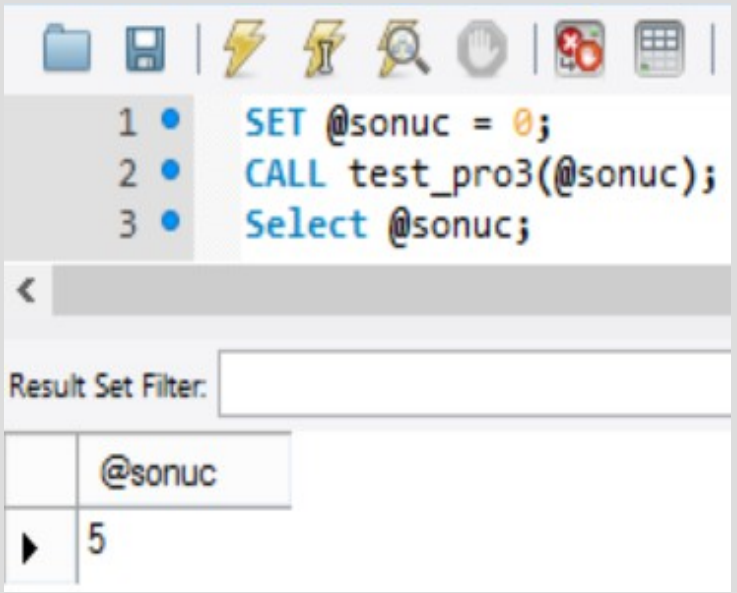
```
1
2   USE `test`;
3   DROP procedure IF EXISTS `test`.`test_pro2`;
4
5   DELIMITER $$
6   USE `test`$$
7   CREATE DEFINER=`root`@`localhost` PROCEDURE `test_pro2`(IN id INT)
8   BEGIN
9   select ad, soyad from rehber where idrehber = id;
10  END$$
11
12  DELIMITER ;
13  ;
```



OUT Yöntem Belirteci

- Bu yöntem belirteci ile tanımlanmış bir parametreye dışarıdan bir değişken gönderip SP'nin bu değişkene bir değer atamasını sağlarız. Sonunda da SP işlemesi bittiğinde bu değeri kullanabiliriz.

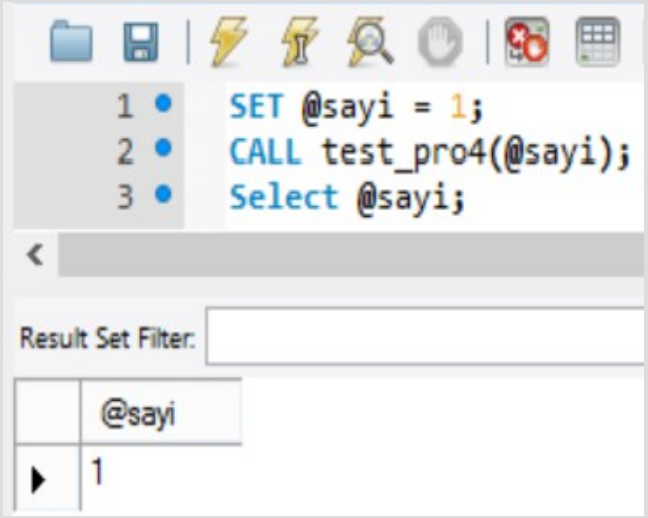
```
5   DELIMITER $$
6
7   CREATE DEFINER=`root`@`localhost` PROCEDURE `test_pro4`(OUT toplam INT)
8   BEGIN
9   select count(*) into toplam from rehber;
10  END
```



INOUT Yöntem Belirteci

- INOUT isminden de anlaşılabileceği gibi IN ve OUT yöntem belirteçlerinin birleşimidir.
- Başka bir deyişle, hem SP'e değer gönderilebilir hem de SP'den bir değer alınabilir. Bu yöntem diğer programlama dillerindeki referans parametre tipine benzemektedir. Yani değişken SP'e referansı vasıtası ile geçirilir.

```
5   DELIMITER $$
6
7   CREATE DEFINER=`root`@`localhost` PROCEDURE `test_pro4`(INOUT sayi INT)
8   BEGIN
9   SET sayi = (select count(*) from rehber where idrehber=1);
10  END
```



- SP’leri listelemek için:
  - SHOW PROCEDURE STATUS

  sorgusunu kullanabilirsiniz.

- SP’i silmek için ise aşağıdaki SQL sorgu şablonu kullanabilirsiniz
  - DROP PROCEDURE procedure_adi

Saklı yordamlarda Kontrol Deyimleri

- MySQL'de aşağıdaki kontrol deyimleri bulunur:
  - IF koşul deyimi
  - CASE koşul deyimi
  - WHILE döngü deyimi
  - REPAT – UNTIL döngü deyimi
  - LOOP döngü deyimi
  - LEAVE deyimi (diğer dillerdeki break benzeri.)
  - ITERATE deyimi (diğer dillerdeki continue benzeri.)
  - RETURN deyimi
- VTYS'ye göre kullanım farklılıkları olabilir.

Trigger (Tetikleyiciler)

- Trigger yani tetikleyici, ilişkisel veri tabanı yönetim sistemlerinde bir tabloda belirli olaylar meydana geldiği zaman yani ekleme, güncelleme, silme işlemlerinden biri gerçekleşmeden önce veya sonra çalışan ve belirli işlemleri kodlandığı şekilde yerine getiren yordamdır.
- Genel amacı veri bütünlüğünü korumaktır.

Tetikleyiciler ne zaman kullanılır?

- Çeşitli amaçlara uygun olarak tetikleyiciler kullanılabilir.
  - Değişiklikleri takip etmek
  - Birincil anahtar üretmek
  - Karmaşık iş kurallarını gerçekleştirmek
  - E-posta atmak gibi olayları otomatik olarak yapmak
  - Standart hata mesajlarının dışında bir hata mesajı elde etmek
  - Veritabanı erişimlerini takip edebilmek
  - Nesnede meydana gelebilecek değişiklikleri takip ve engellemektir

Tetikleyicilerin Çalışması

- Bir işlemin gerçekleşmesini veya gerçekleşmeye başlamasını tetikleyici ateşleyebilir. Tetikleyiciyi ateşleyen bir işlem ile karşılaşıldığında tetikleyici ile işlem bir blokta ele alınır. Bu blok, RAM'de bulunan geçici hafıza bloğudur. Tetikleyiciyi çağıran işlemi onaylamak anlamında hiçbir işlem yapmayabilir veya işlem başarılı olmadığında işlemi geçersiz kılabilir.

Tetikleyicilerin Çalışması

- Tetikleyici çalıştığı zaman Inserted ve Deleted adı verilen sahte tabloları kullanır. Bu tablolar tetikleyicinin ateşlendiği tabloyla eşdeğer alanlara sahiptir. Bunlar da mantıksal olarak RAM'de bulunur. Asıl tabloya bir kayıt eklendiğinde ve tetikleyici ateşlendiğinde bu kayıt Inserted tablosuna da eklenir. Tablodan bir kayıt silindiğinde silinen kayıt Deleted sahte tablosuna da eklenir. Update işlemi ise önce silme (Delete) ve ardından bir kayıt ekleme (Insert) olarak ele alınır. Bir kayıt güncellendiğinde asıl kayıt Deleted sahte tablosuna, değişen kayıt da Inserted sahte tablosuna yazılır.
- Inserted ve Deleted sahte tabloları RAM'deki hafıza bölgesinde elde edilir. Bundan dolayı, TRUNCATE TABLE gibi yazılmayan tablolara yansımayan değişiklikler tetikleyici tarafından yakalanamaz. Tetikleyiciler dışarıdan parametre almazlar. Ancak, sahte tablolar sayesinde son işlemden etkilenmekte olan kayıtlar tespit edilebilir.

Tetikleyici Türleri (MySQL)

- Mysql'de trigger nesneleri iki çeşit olarak bulunmaktadır.
  - Veritabanında nesneler üzeride yapılan değişikler için (Drop,Create,Alter),
  - Tablolar içersinde bulundukları kayıtlar, içerisinde yalpan değişler (Insert,Update,Delete) için kullanır.

Tetikleyici nasıl bir performans artışı sağlıyor?

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

Tetikleyici oluşturmak

CREATE TRIGGER triggername

[BEFORE|AFTER] [INSERT|UPDATE|DELETE] ON tablename

FOR EACH ROW {statement}

Örnek

- Yeni bir çalışan kaydı girildiği anda son güncelleme alanını 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

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

Tetikleyici Alias

Bağlam	New.field	Old. Field
Before Insert	Tüm alanlar	Desteklemez
After Insert	Desteklemez	Ekledikten sonra «eski» kayıt yoktur.
Before Update	Tüm alanlar	Desteklemez
After Update	Desteklemez	«Eski» kayıtların hepsi yeni güncellenmiştir.
Before Delete	Silmeden önce «yeni» kayıt yoktur	Desteklemez
After Delete	Desteklemez	Kayıtların «eski» değerleri silinmiştir

Tetikleyici Silme

- Kontrolsüz silme
  - DROP TRIGGER calisan_son_guncelleme;
- Kontrollü silme
  - DROP TRIGGER IF EXISTS calisan_son_guncelleme;

Tetikleyici Birden Fazla SQL içeriyorsa

- Eğer tetikleyici ile birden fazla işlem çalıştırılacaksa

  BEGIN

  ……..

  END

  bloğu kullanılır

Trigger
PHPMyAdmin üzerinden

- Aşağıdaki tabloları oluşturulalım.
- Amaç birinci tablodaki (stok) adet değiştiğinde sonuc bölümünü güncellemek.

İlk Trigger

```
CREATE TRIGGER ilk_trigger BEFORE INSERT ON
stok
FOR EACH ROW
INSERT INTO toplam VALUES(NEW.adet);
```

- Trigger ı tanımlarken BEFORE INSERT ON stok kod bloğu ile stok tablosunda ekleme(INSERT) işlemi gerçekleşmeden önce trigger ımız çalışacaktır. NEW anahtar kelimesi ile yeni girilen adet değerini almış oluyoruz ve aynısını toplam tablosundaki sonuc alanına eklemesini sağlıyoruz.

Birden fazla işlem var ise

- Eğer ki girilen adetin %20'sinin yazılmasını istiyorsak aşağıdaki gibi bir yapı kullanmalıyız:

```
CREATE TRIGGER ilk_trigger BEFORE INSERT ON stok
FOR EACH ROW
BEGIN
SET NEW.adet = NEW.adet * 20 / 100;
INSERT INTO toplam VALUES(NEW.adet);
END;
```

- Yukarıda gördüğünüz gibi trigger ın açıklama(statement) kısmında birden fazla satır kullanacağımız zaman bu satırları BEGIN-END bloğu içerisine almak zorundayız. SET anahtar kelimesini değer atamaları yapmak için kullanıyoruz. Yukarıdaki örnekte NEW.adet değişkeninin değerine yeni değer atamak için en başında SET anahtar kelimesini kullandık. Kullanmasaydık syntax hatası ile karşılaşacaktık. Burada dikkat etmemiz gereken başka bir nokta ise satır ayıracı. İki türlü satır ayıracı mevcut. Birincisi sql sorgusunun bittiğini gösteren satır ayıracı, ikincisi kod yazarken yeni satıra geçmek için kullandığımız ; (noktalı virgül) olan satır ayıracı.

Birden fazla SQL trigger işlemi

```
CREATE TRIGGER ilk_trigger AFTER INSERT ON stok
FOR EACH ROW
BEGIN
DECLARE toplamSonuc INT DEFAULT 0;
SELECT SUM(adet) INTO toplamSonuc FROM stok;
INSERT INTO toplam VALUES(toplamSonuc);
END;
```

- Bu trigger ımızı oluştururken AFTER INSERT ON stok kod bloğunu kullandığımız için stok tablosuna ekleme yapıldıktan sonra trigger çalışacaktır. DECLARE anahtar kelimesi ile toplamSonuc adında INT tipinde ve varsayılan(DEDAULT) değeri sıfır olan bir değişken tanımlamış olduk. Daha sonra SUM fonksiyonu ile stok tablosundaki adet değerleri toplandı ve INTO anahtar kelimesi ile toplamSonuc değişkenine aktarıldı. En son olarak toplamSonuc değişkeni toplam tablosuna eklendi. Bu şekilde trigger ımız ekleme işleminden sonra çalışmış oldu.

Güncelleme işlemi için trigger

- Güncelleme(UPDATE) işlemi ile ilgili olacak. Trigger ımız aşağıdaki gibi olsun.

```
CREATE TRIGGER ilk_trigger AFTER UPDATE ON stok
FOR EACH ROW
BEGIN
DECLARE toplamSonuc INT DEFAULT 0;
SELECT SUM(adet) INTO toplamSonuc FROM stok;
INSERT INTO toplam VALUES(toplamSonuc);
END;
```

- Bu trigger ımız bir önceki trigger örneği ile aynı işlemleri yapıyor fakat AFTER UPDATE ON stok kod bloğunu kullandığımız için stok tablosunda güncelleme işlemi gerçekleştikten sonra çalışacaktır.

New - Old

- Daha önceki örneklerde NEW anahtar kelimesini kullanmıştık. Şimdi ise OLD anahtar kelimesini kullanalım. Örneğe geçmeden önce NEW anahtar kelimesi ile OLD anahtar kelimesi arasındaki farklardan bahsetmek istiyorum. NEW anahtar kelimesi ile istenilen değişkenin yeni değerine erişilir , OLD anahtar kelimesi ile eski değerine erişilir. OLD anahtar kelimesi güncelleme (UPDATE) ve silme (DELETE) işlemlerinde kullanılır. NEW anahtar kelimesi ise ekleme (INSERT) ve güncelleme (UPDATE) işlemlerinde kullanılır. Diğer işlemlerde kullanmak için trigger oluşturmaya çalışırsanız hata ile karşılaşacaksınızdır. Örneğimiz aşağıdaki gibi olsun:

```
CREATE TRIGGER ilk_trigger BEFORE UPDATE ON stok
FOR EACH ROW
BEGIN
INSERT INTO toplam VALUES(OLD.adet);
INSERT INTO toplam VALUES(NEW.adet);
END;
```

- Bu trigger ımızıda BEFORE UPDATE ON stok kod bloğunu kullandığımız için stok tablosunda güncelleme işlemi yapılmadan önce trigger ımız çalışacaktır. Güncelleme işleminde adet in değerini değiştirdiğimizi varsayalım. OLD anahtar kelimesi ile adet in eski değerine,NEW anahtar kelimesi ile adet in yeni değerine erişerek toplam tablosuna eklenmesini sağladık.

Silme işlemi ile değer güncelleme

- silme (DELETE) işlemi ile bir örnek inceleyelim. Örnek sorgumuz aşağıdaki gibi olsun.
  - CREATE TRIGGER ilk_trigger AFTER DELETE ON stok
  - FOR EACH ROW
  - INSERT INTO toplam VALUES(OLD.adet);
- Bu trigger ımızda AFTER DELETE ON stok kod bloğunu kullandığımız için stok tablosunda silme (DELETE) işlemi gerçekleştikten sonra çalışacaktır. OLD.adet ile silinen satırdaki adet değerine erişerek toplam tablosuna ekledik.

Trigger listeleme ve silme

SHOW TRIGGERS

SQL sorgusu:

```
SHOW triggers
```

[Düzenle] [PHP Kodu oluştur] [Yenile]

Sorgu sonuçları işlemleri

Baskı görünümü Baskı görünümü (tüm metinler ile)

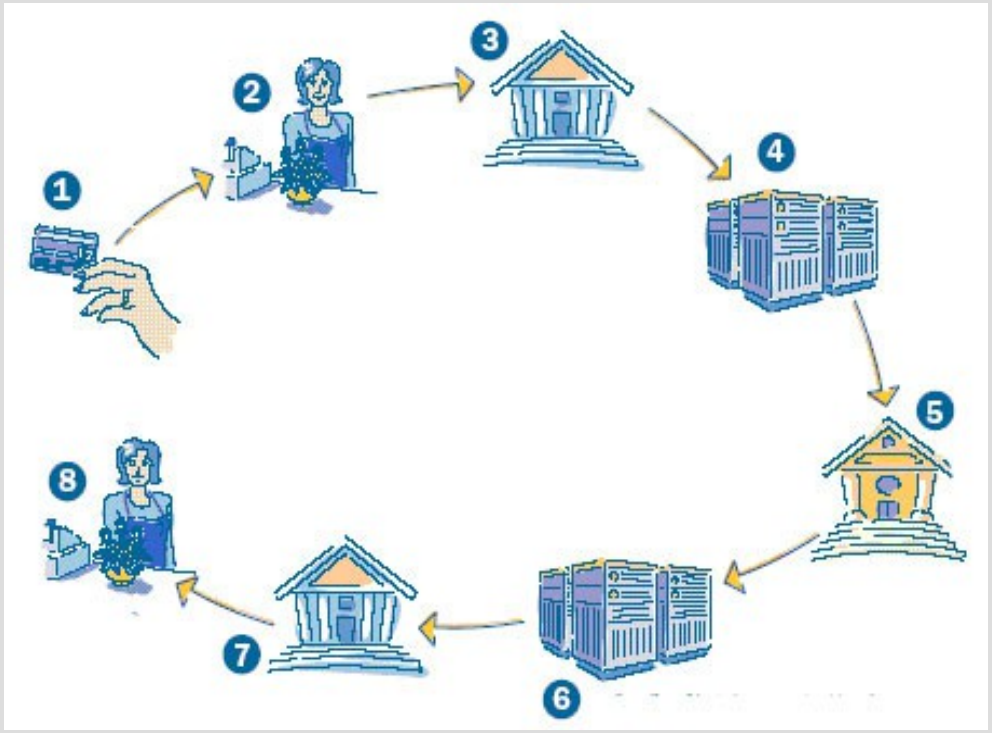
Trigger	Event	Table	Statement	Timing	Created	sql_mode	Definer	character_set_client	collation_connecti
ilk_trigger	INSERT	stok	BEGIN DECLARE toplamSonuc INT DEFAULT 0; SELECT ...	AFTER	*NULL*	NO_AUTO_CREATE_USER,NO_ENGINE_SUBSTITUTION	root@localhost	utf8	utf8_unicode_ci
deneme	DELETE	stok	BEGIN insert into toplam VALUES(OLD.adet); END	AFTER	*NULL*	NO_AUTO_CREATE_USER,NO_ENGINE_SUBSTITUTION	root@localhost	utf8	utf8_unicode_ci

Trigger ı silmek için kullanacağımız kod şablonunu aşağıdaki gibidir.

DROP TRIGGER trigger_adı

Transaction

- Birden çok işlemin bir arada yapıldığı durumlarda eğer parçayı oluşturan işlemlerden herhangi birinde sorun olursa tüm işlemleri iptal etmeyi sağlar.
- Örnek 1:
  - Kredi kartı ile alışveriş olayında transaction olayı vardır, siz marketten ürün alırken sizin hesabınızdan para düşülecek marketin hesabına para aktarılacaktır bu işlemde hata olmaması gerekir ve bu işlem transaction blogu sayesinde yapılır. Yani kodlarımız sırayla işler bu esnada bir sorun çıkarsa bütün işlem RollBack sayesinde iptal edilir.



Transaction
ya hep, ya hiç!!!

- Örnek 2:
  - Havale yapılan hesaptan para düşülür ve sonra paranın gideceği hesaba bu miktar eklenir. Peki havale yapılacak hesaptan para düşüldükten sonra bir şeyler olsa ve para düşülmesine rağmen alıcı hesaba para aktarılması gerçekleştirilmemiş olsa, elektrik gitse mesela, bu faciadır aslında :) Neyse ki transactionlar var :) rollback mekanızmalarıyla tüm işlemler geri alınır. Buradan anlayacağınız gibi daha küçük parçalara ayrılamayacak bir işlemdir bu, transaction işte budur. Eğer tüm işlemler başarıyla gerçekleşmişse commit komutu verilir, içeriden bir işlemcik gerçekleşmezse rollback edilir.

Transaction Özellikleri

- 4 ana özelliği vardır. Kısaca ACID: Atomicity – Consistency – Isolation – Durability kelimelerinin baş harflerinin birleşmesinden oluşur.
  - Atomicity: Bu prensibe göre bir transactionın başarılı olabilmesi için transactionın içindeki tüm işlemcikler başarılı şekilde gerçekleşmiş olmalıdır.
  - Consistency: İşlemler sonucunda çıkan verinin tutarlı olmasıdır, aynı girişte hep aynı çıktıyı üretmelidir.
  - Isolation: Transaction gerçekleşirken dışarıdan müdahaleye izin verilmemelidir, adı üstünde izole olmalıdır. Bir transaction başka bir transactiondan bağımsız olmalıdır.
  - Durability: Eğer transaction içinde bir hata olursa başa geri dönebilme yeteneğidir (rollback). Böylece devamlılık başarıyla sağlanabilir.

Transaction Şartları

- Öncelikle belirli şartları sağlamamız lazım, yani sorgu öbeğimiz insert, delete ya da updatelerden oluşmalı.
- DDL komutları (create, alter, drop, comment gibi tanımlayıcı komutlar) transactionlar içerisinde kullanılamaz.
- MySQL'in en büyük iki eksiği Stored Procedure ve Transaction kavramları sonunda 5.X versiyonlarına entegre edildi.
- MySQL'de, InnoDB ve BerkeleyDB de Transaction desteği vermektedir.

Transaction Adımları

- Transactiona başla
- Insert/update/delete komutlarını uygula
- Commit / Rollback yap

Authorization (Yetkilendirme)

- MySQL erişim haklarını mysql adlı özel bir veritabanında tutar. Bu veritabanı birtakım erişim hakkı tablolarından oluşur (grant tables):
  - user tablosu
  - db tabloasu
  - host tablosu
  - table-priv tablosu
- Veri değiştirme komutları ile mysql veritabanı üzerinde değişiklik yaparak kullanıcı hakları değiştirilebileceği gibi daha emniyetli metod olarak grant ve revoke komutları kullanılabilir.

MySQL veritabanındaki tablolar

- user tablosu
  - kullanıcıların sunucuya hangi host makinadan hangi haklarla bağlanabileceğini belirler. Bu tabloda bulunan haklar kullanıcıya global olarak verilir ve verilen haklar istisnasız tüm veritabanları üzerinde etkili olur. Host alanı ise tam tersine bağlanabilecek tüm hostları içermelidir. Diğer tablolarda bunların dışında host alanı girilmesi anlam taşımayacaktır.
- db tablosu
  - kullanıcıların belirli bir veritabanına hangi haklarla erişebileceğini gösterir. Global olarak sahip olunmayan hakları verebilir, ancak global olarak sahip olunan hakları kaldıramaz. Yani user tablosunda kullanıcıya verilen hakları geri alamaz ama belirli bir veritabanına ait olmak üzere yeni haklar verebilir. Burada host alanı user tablosunda verilen host aralığını daraltabilir. Yani bir kullanıcı user tablosundaki host aralığının kapsadığı bir host'tan sunucu bağlantısı kursa da, db tablosunda belirli bir veritabanına kullanıcının bu hosttan erişimine ilişkin bir satır (kayıt) yoksa veritabanına erişemez. Burada host alanı için için tipik olarak '%' kullanılabilir. Böylece sunucu bağlantısı sağlayabilen tüm hostlar veritabanına kabul edilirler.

MySQL veritabanındaki tablolar

- host tablosu
  - db tablosundaki 'host' alanına değer girilmezse anlam kazanır. Bu durumda ilgili veritabanına bağlanabilecek hostlar host tablosundan taranır ve erişim haklarına bakılır. Burada haklar açısından user ve db tabloları arasındaki ilişkiden farklı br durum vardır. Öyle ki burada db ve host tabloları arasında adeta bir AND işlemi yapılır.
- table-priv tablosu
  - tablo düzeyinde erişim haklarını belirler. (Sadece grant, revoke komutları ile değiştirilmeli) column-priv tablosu tablo kolonları düzeyinde erişim haklarını belirler. (Sadece grant, revoke komutları ile değiştirilmeli) Bu tablolarda kullanıcının bir erişim hakkı varsa daha alt düzey tablolara bakılmaz. Yapılan değişiklikleri etkinleştirmek için flush privileges komutu kullanılır.

Yetkilendirme Sınıflaması

- Yönetici hakları, MySQL veritabanının yönetimi için verilir. Fakat bu yetkiler geneldir, özellikle bir veritabanı için verilmez.
- Veritabanı yetkileri ise veritabanı bazında verilebilir, fakat veritabanı içindeki tüm objeler için geçerlidir. Aynı zamanda sadece bir veritabanı için verilebileceği gibi, tüm veritabanları için verilmesi de sağlanabilir.
- Bir de tablo, indeks, view, store prosedür`ler için verilen haklar vardır. Bu haklar ise belirli bir veritabanının herhangi bir objesi için de verilebilir, tüm veritabanlarının tüm objeleri için de verilebilir.

Yetki	Bulunduğu Yer	Etkilenenler
CREATE	Create_priv	V, T, I
DROP	Drop_priv	V ya da T
GRANT OPTION	Grant_priv	V, T, SP
REFERENCES	Reference_priv	V ya da T
EVENT	Event_priv	V
ALTER	Alter_priv	T
DELETE	Delete_priv	T
INDEX	Index_priv	T
INSERT	Insert_priv	T
SELECT	Select_priv	T
UPDATE	Update_priv	T
CREATE TEMPORARY TABLES	Create_tmp_table_priv	T
LOCK TABLES	Lock_tables	T
TRIGGER	Trigger_priv	T
CREATE VIEW	Screate_view_pric	VW
SHOW VIEW	Show_view_priv	VW
ALTER ROUTINE	Alter_routine_priv	SP
CREATE ROUTINE	Create_routine_priv	SP
EXECUTE	Execute_priv	SP
FILE	File_priv	Dosya erişimi (V)
CREATE USER	Create_user_priv	Y
PROCESS	Process_priv	Y
RELOAD	Reload_priv	Y
REPLICATION CLIENT	Repl_client_priv	Y
REPLICATION SLAVE	Repl_slave_priv	Y
SHOW DATABASES	Show_db_priv	Y
SHUTDOWN	Shutdown_priv	Y
SUPER	Super_priv	Y
ALL [PRIVILEGES]		Y
USAGE		Y

V = Veritabanları
T = Tablolar
SP = Store prosedürler
I = Indeksler
VW = Görüntüler (view)
Y = Yöneticiler

Kullanıcı Oluşturma

- Sisteme root kullanıcı ile bağlanalım.
- MySQL'de oluşturacağınız kullanıcılar için host parametresi de verebilirsiniz. Bu ayar "kullanıcı nereden (hosttan) bağlanabilir" sorusunun cevabıdır.
  - Localhost: Sadece MySQL'in bulunduğu makineden erişilir.
  - %: Sadece dışarıdan erişilir.
  - IP: Sadece sizin belirlediğiniz IP'lerden erişilir.

```
CREATE USER 'korsan'@'localhost' IDENTIFIED BY 'powerful_password';
```

```
CREATE USER 'korsan'@'%' IDENTIFIED BY 'another_password';
```

Kullanıcı adları aynı olan şifreleri farklı olan 2 farklı kullanıcı oluşturduk. Birisi "powerful_password" şifresi ile sadece içeriden, diğeri ise "another_password" şifresi ile sadece dışarıdan erişebilir durumdadır.

Kullanıcıları Listeleme / Silme

Kullanıcıları Listeleme

```
SELECT host, user FROM mysql.user;
```

Kullanıcıları Silme

```
DROP USER korsan;
```

```
DROP USER 'korsan'@'localhost';
DROP USER 'korsan'@'%';
```

```
DELETE FROM mysql.user WHERE user='korsan'
-- veya
DELETE FROM mysql.user WHERE user='korsan' AND host='localhost';
DELETE FROM mysql.user WHERE user='korsan' AND host='%';
```

MySQL'de Kullanıcı Yetkilendirme

```
GRANT ALL PRIVILEGES ON korsanDB.* TO 'korsan'@'localhost';
```

"korsan" kullanıcısı "local makineden bağlanarak", "korsanDB" veri tabanının tüm tablolarında (korsanDB.* ile ifade ediliyor) tüm işlemleri yapabilir.

```
GRANT SELECT, INSERT, UPDATE, DELETE ON korsanDB.users TO 'korsan'@'%';
```

"korsan" kullanıcısı "dışarıdaki bir makineden bağlanarak", "korsanDB" veri tabanının tüm users tablosunda (korsanDB.users) sadece "SELECT, INSERT, UPDATE, DELETE" işlemlerini yapabilir.

Kullanıcının şifresi yoksa aşağıdaki gibi şifre de oluşturabilirsiniz.

```
GRANT ALL PRIVILEGES ON korsanDB.* TO 'korsan'@'localhost' IDENTIFIED BY 'password';
```

Tüm yetkileri görüntülemek

```
SHOW GRANTS FOR 'korsan'@'localhost';
SHOW GRANTS FOR 'korsan'@'%';
```

MySQL Veritabanında dikkat edilecek hususlar

- Örneğin, “test” gibi kendi içinde ön tanımlı gelen ve bazen şifresiz de erişim sağlanabilen veritabanını silmeliyiz.
- “root” hesabı “no-password” olarak gelebiliyor. Bu yüzden tahmini zor bir şifre atamalıyız.
- “root” kullanıcısı için sadece “localhost” erişimi vermeliyiz.
- “anonymous” gibi ön tanımlı gelen kullanıcıları silmeliyiz.
- kullanıcı oluştururken ‘kullanıcı’@’host_name’ kısmında muhakkak, kullanıcının bağlanacağı IP adresini belirtmeliyiz. “%” gibi genel ifadeler kullanmak, hür türlü yerden bağlantı isteği yapılacağı anlamına gelecektir.
- Özellikle uygulama güvenliği tarafında, uygulamaya verilen kullanıcının hakları, yapacağı işlevler le sınırlandırılmalıdır. Örneğin; sadece SELECT yapacaksa, bunun dışında yetki verilmemesi gerekmektedir. Bu yüzden uygulamalar da kullandığımız kullanıcıların SELECT, UPDATE, INSERT v.s gibi temel yetkileri dışında, fazladan yetki vermemeliyiz.

Kaynaklar

- http://www.caglarozcan.com.tr/mysql-ve-sql-server-notlari/mysql-stored-procedure-sakli-yordam-kullanimi.html
- http://www.sinanakyazici.com/mysqlde-stored-procedure-sakli-yordam-kullanimi
- http://emircem.wordpress.com/2011/07/20/mysqlde-sakli-yordamlar/
- http://www.sinanakyazici.com/mysql-trigger-tetikleyici-kullanimi
- http://www.tutorialspoint.com/mysql/mysql-transactions.htm
- http://webguvenligi.org/dergi/MySQLKullaniciHaklari-Subat2010-BunyaminDemir.pdf
- http://www.cemdemir.net/veri-tabani/mysql-icin-kullanici-olusturma-ve-dogru-yetkilendirme-2223.html