## Laboratuvar 7

# DML CÜMLELERİNİN KULLANIMI

Veritabanına very ekleme, mevcut verileri güncelleme ve slime işlemleri SQL Veri İşleme Dili (*DML- Data Manipulation Language*) kullanılarak yapılmaktadır.

DML cümlelerini kullanarak veriler üzerinde değişiklik yaptıktan sonar bu değişikliklerin kalıcı olması için COMMIT komutu çalıştırılır. COMMIT işleminin çalıştırılması ile o ana kadar yapılan değişikliklerin hepsi veritabanına işlenir ve kalıcı hale gelir. Yapılan değişiklikleri geri almak için ise ROLLBACK komutu kullanılır.

## TABLOYA KAYIT EKLEME

Bir veya daha fazla tabloya kayıt eklemek için INSERT cümlesi kullanılır. Eklenecek kayıt ya da kayıtlar doğrudan eklenebildiği gibi, bi alt sorgu tarafından alınarak da tabloya eklenebilir.

**Bir Tabloya Kayıt Ekleme**

Basit bir INSERT cümlesinin sözdizimi aşağıda verilmiştir.

***INSERT INTO tablo_adı [(sütun1, sütun2, sütun3,…)]***
***VALUES (değer1, değer2, değer3,…)***

**NOT:** INSERT cümleleri yazarkan sütun isimlerinin açıkça belirtilmesi ileride tabloya yeni sütun eklendiği zaman hata alınmasını engeller. Örneğin bir program yazmaktasınız ve program içerisinede TABLO_A'ya bir kayıt ekliyorsunuz.

```
SQL> INSERT INTO tablo_a
      VALUES (1, 'Test');
1 row created.
```

Daha sonra başka bir iş için ilgili tabloya bir sütun daha eklendiğini düşünelim. Bu durumda daha once geçerli olan SQL'imiz geçersiz hale gelecektir. Çünkü sütun isimleri belirtilmediği zaman tüm sütunlar için ilgili değerlerin belirtilmesi gerekir.

```
SQL> INSERT INTO tablo_a
      VALUES (1, 'Test');
INSERT INTO tablo_a VALUES (1, 'Test');
  *
ERROR at line 1:
ORA-00947 not enough values
```

Eğer SQL'imizi daha önceden sütun isimleri belirterek yazmış olsaydık böyle bir hata ile karşılaşmamış olacaktık ve programımız hala çalışmaya devam edecekti.

```
SQL> INSERT INTO tablo_a (kolon1, kolon2)
      VALUES (1, 'Test');
1 row created.
```

INSERT cümlesi yazarken bazı durumlarda tabloya eklenecek kayıtlar başka bir SELECT cümlesinin sonucu olabilir. SELECT cümlesinden dönen her bir kayıt sırasıyla tabloya eklenir. Tabii bu işlemin geçerli olması için SELECT cümlesinden dönen değerler ile verilerin ekleneceği tablonun sütun sıralaması ve tipleri uyumlu olmalıdır.

Görüntülere (*View*) aşağıdaki koşullarda kayıt eklenemez:

- Görüntüyü oluşturan SELECT cümlesinde DISTINCT operatörü kullanılmış ise,
- Görüntüyü oluşturan SELECT cümlesinde gruplama fonksiyonu kullanılmış ise,
- Görüntüyü oluşturan SELECT cümlesinde GROUP BY, ORDER BY veya CONNECT BY kullanılmış ise,
- Görüntüyü oluşturan SELECT cümlesinin SELECT kısmında başka bir alt sorgu kullanılmışş ise;

bu görüntü aracılığıyla tabloya kayıt eklenemez.

## TABLODAKİ KAYITLARI GÜNCELLEME

Tablolarda yer alan kayıtları güncellemek için UPDATE cümleleri kullanılır. Basit bir UPDATE cümlesinin sözdizimi aşagıda verilmiştir.

***UPDATE tablo_adı***
***SET [sütun1=değer1, sütun2=değer2,…]***
***[WHERE koşullar]***

## TABLODAKİ KAYITLARI SİLMEK

Tablolardaki kayıtları silmek için DELETE cümleleri kullanılır. Aşagıda DELETE cümlesi sözdizimi görülmektedir.

***DELETE [FROM] tablo_adı***
***[WHERE koşullar]***

Bu sözdiziminde FROM kelimesi ve WHERE koşul cümlesi opsiyoneldir. Eğer hiçbir WHERE koşulu kullanılmaz ise tablodaki tüm kayıtlar silinir. Koşul belirtilirse sadece koşula uyan kayıtlar silinir.

**NOT:** DELETE cümlesinde WHERE koşulu kullanmadığımız zaman tablodaki tüm kayıtlar silinir. Eğer tamamını sileceğimiz tablo oldukça büyük ise bu işlem cok uzun zaman alabilir. Hatta eğer veritabanı gerialma bölümü yeteri kadar büyük değil ise silme işleminin gerçekleştirilememesi gibi bir durumla karşılaştırılabilir. Bu gibi tablonun tamamının silineceği durumlarda TRUNCATE cümlesi kullanılabilir.
**Örneğin:**

**DELETE tablo_adı** yerine **TRUNCATE TABLE tablo_adı** kullanılabilir.

## TRUNCATE

TRUNCATE komutu bir tablodaki tüm kayıtları silen bir komuttur. WHERE koşulu olmayan DELETE cümlesine benzer. DELETE bir veri işleme dili (DML) komutu olmasına karşın TRUNCATE very tanımlama dili (DDL) komutudur ve bundan dolayı da TRUNCATE komutu DELETE komutundan farklı bir karakteristiğe sahiptir. DML ve DDL cümleleri arasındaki temel fark şudur: Bir DDL cümlesi çalıştırılırken dolaylı olarak (otomatikman) COMMIT işlemi gerçekleştirilir. Bu da sadece DDL ile yaptığımız değişkliği değil aynı

zamanda diğer bekleyen DML cümlesi kayıt değişikliklerini de işler, yani kaydeder. DDL cümleleri ile yapılan işlemler geri alınamaz.

Aşağıda TRUNCATE cümlesinin sözdizimi görülmektedir.
***TRUNCATE TABLE tablo_adı [[DROP|REUSE] STORAGE];***
Bu sözdiziminde DROP STORAGE varsayılan değerlerdir.

## TRUNCATE ve DELETE

TRUNCATE ve WHERE koşulu olamayan DELETE cümleleri birbirine benzemekle beraber aralarında yapısal farklılıklar vardır. Bu ikisi arasındaki farklar aşağıda belirtilmiştir.

- TRUNCATE gerek küçük gerekse büyük tablolar için oldukça hızlı çalışır. DELETE, ROLLBACK yapılabilmesi için geri alma bilgisi üretir fakat TRUNCATE geri alma bilgisi üretmez. Doğal olarak da TRUNCATE işlemi ROLLBACK ile geri alınamaz.
- TRUNCATE bir DDL cümlesidir ve tüm DDL cümlelerinde olduğu gibi dolaylı olarak işlem sonunda COMMIT işlemi gerçekleştirir. Eğer TRUNCATE işleminden önce verilerde değişiklik olmuş ise bunların hepsi veritabanına işlenmiş olur.
- Başka bir kullanıcının tablosu üzerinde TRUNCATE işleminin çalıştırabilmek için belirlenmiş bir nesne yetkilendirmesi yoktur. Bunun için DROP ANY TABLE yetkisinin kullanıcıya verilmesi gerekir.

**NOT:** TRUNCATE işlemi, bir tablonun **DROP TABLE tablo_adı** cümlesi ile düşürülüp sonra tekrar olşturulması işleminden farklıdır.

- DROP bu tabloya bağlı nesneleri geçersiz kılar. Örneğin bu tabloyu kullanan bir fonksiyon varsa bu geçersiz hale gelir. Yeniden derlenmesi gerekir.
- DROP tablo ile birlikte tabloya bağlı indeksleri, triggerleri ve referansları da düşürür.
- DROP işleminden sonra tablo tekrar oluşturulsa bile tablo ile ilgili yetkilerin yeniden kullanıcılara verilmesi gerekir.

## VERİTABANI İŞLERİ

Oracle'da veri tutarlılığı veritabanı işleri aracılığıyla sağlanır. Veritabanı işleri kayıtlrda yapılan değişiklikler üzerinde daha fazla kontrol ve esneklik sağlar. Bir işlemin başarısız olması ya da sistem hatası durumlarında veri tutarlılığını garanti eder.
Veritabanı işlerini birden fazla DML cümlesinin gruplanarak tek bir iş gibi çalıştırılması ve yapılan değişikliklerin hepsinin birden veritabanına işlenmesi ya da geri alınması şeklinde tanımlamak mümkündür. (*ATOMICITY özelliği*)

Bir veritabanı işi (*transaction*) aşağıdakilerden herhangi birini kapsamaktadır:

- Veritabanında tutarlı değişiklik yapan DML (*Veri İşleme Dili*) cümleleri.
- Bir DDL (Veri Tanımlama Dili) cümlesi.
- Bir DCL (Veri Kontrol Dili) cümlesi.

Bir veritabanı işi (*transaction*) ilk DML cümlesi ile başlar ve aşağıdakilerden biri ile sona erer:

- COMMIT ya da ROLLBACK cümlesi çalıştırıldığında

- Bir DDL cümlesi çalıştırıldığında (*DDL cümlelerinin dolaylı olarak COMMIT işlemini çalıştırdığını hatırlayınız.*)
- Bir DCL cümlesi çalıştırıldığında
- Kullanıcı SQL*Plus ya da iSQL*Plus'dan çıktığı zaman (*SQL*Plus ya da iSQL*plus'dan çıkışta otomatik olarak COMMIT işlemi çalıştırılır. SQL*Plus ya da iSQL*Plus'ın anormal bir şekilde sonlanmasında otomatik olarak ROLLBACK çalıştırılır.*)
- Kullanıcı makinesi ya da sistem çökerse. (*Otomatik olarak ROLLBACK çalıştırılır.)*

Bir veritabanı işi tamamlandıktan sonra bir sonraki SQL otomatik olarak yeni iş başlatır. DDL ve DCL cümleleri otomatik olarak veritabanına işlenir ve dolaylı olarak veritabanı işi sonlanır.