TCL İFADELERİ VE A.C.I.D KURALLARI

Öğretim Görevlisi A. Berika VAROL MALKOÇOĞLU



İçindekiler

- TCL Nedir?
- TCL örnekleri
- A.C.I.D Nedir?



TCL

- TCL (İşlem Kontrol Dili Transaction Control Language): DML ile gerçekleştirdiğimiz işlemleri yönettiğimiz ve kontrol ettiğimiz dildir.
 - **COMMIT**: Yapılanları kayıt eder.
 - SAVEPOINT: Daha sonra geri dönülecek bir dönüş noktası belirler.
 - ROLLBACK: Mevcut işlemi geri alır ve değişikliklerini iptal eder.



TRANSACTION Nedir?

- Daha küçük parçalara ayrılamayan en küçük işlem yığını
- Hepsi tek bir SQL ifadesiymiş gibi yürütülen SQL komutları kümesidir.
- Transaction'ın tamamlanabilmesi ve verilerdeki değişikliklerin kaydedilebilmesi için transaction'da yer alan tüm SQL komutlarının gerçekleştirilmesi gerekir.



TRANSACTION Nedir?

- Transaction bloğu ya hep ya hiç mantığı ile çalışır.
- Ya tüm işlemler düzgün olarak gerçekleşir ve geçerli kabul edilir.
- Ya da bir kısım işlemler yolunda gitse bile, blok sona ermeden bir işlem yolunda gitmese hiçbir işlem olmamış gibi kabul edilir.
- Eğer bir transaction herhangi bir nedenle tamamlanmazsa:
 - Transaction'dan ötürü veri kümesinde meydana gelen değişiklikler iptal edilir.
 - Veritabanı transaction başlamadan önceki orijinal durumuna geri döndürülür.



TRANSACTION Oluşturma

- Transaction başlatmak için:
 - START TRANSACTION;
 - BEGIN
 - BEGIN WORK
- Transaction'ı sonlandırmak için:
 - Commit(başarılı)
 - Rollback(başarısız)



AUTOCOMMIT

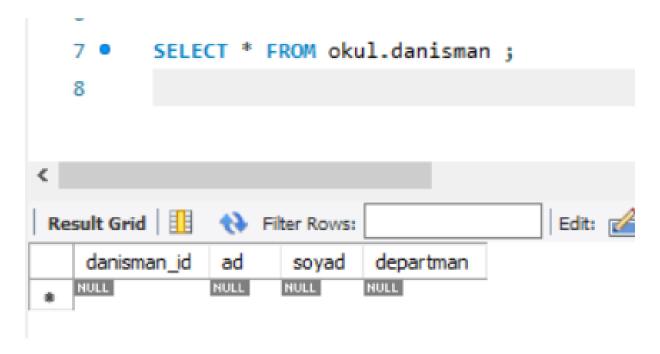
- AUTOCOMMIT özelliği default olarak 1 yani aktif olarak gelir.
- AUTOCOMMIT aktif olduğu için tablolar üzerindeki her hareketimiz otomatik olarak anında COMMIT edilmiş olur.
- Böylece yaptığımız işlemi geri alma şansımız olmaz.
- Yanlış yaptığımız bir UPDATE işlemi kalıcı olarak tablolara yazılmış olur.



AUTOCOMMIT



• Yapılan DML işlemlerini kalıcı hale getirmek için kullanılır.



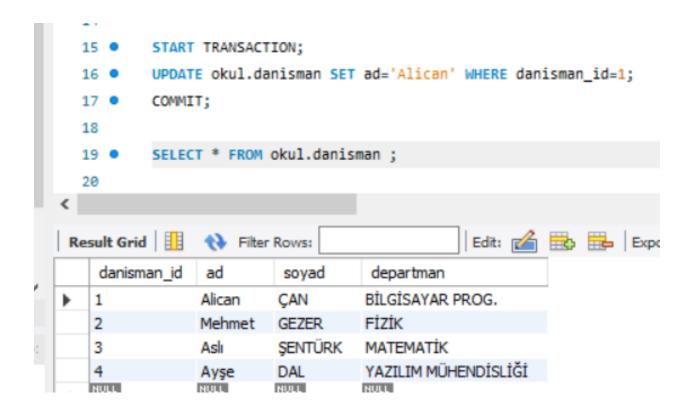


```
2 •
          START TRANSACTION;
          INSERT INTO okul.danisman VALUES (1, 'ALİ', 'ÇAN', 'BİLGİSAYAR PROG.');
  3 •
          INSERT INTO okul.danisman VALUES (2,'Mehmet','GEZER','FİZİK');
  4 •
          INSERT INTO okul.danisman VALUES (3, 'Aslı', 'ŞENTÜRK', 'MATEMATİK');
  5
          INSERT INTO okul.danisman VALUES (4, 'Ayşe', 'DAL', 'YAZILIM MÜHENDİSLİĞİ');
  6
          COMMIT;
                                               Edit: 🚄 🖶 🖶 | Export/Import: 📳 👸 | V
Result Grid
               Filter Rows:
   danisman_id
                ad
                         soyad
                                   departman
               ALİ
                                   BİLGİSAYAR PROG.
                        CAN
               Mehmet
                        GEZER
                                  FİZİK
                        SENTÜRK
                                  MATEMATİK
               Aslı
                                  YAZILIM MÜHENDİSLİĞİ
                        DAL
               Ayşe
               NULL
                        NULL
  NULL
                                  NULL
```



```
ŏ
  9 •
         START TRANSACTION;
          INSERT INTO okul.ogrenci VALUES (1000, 'Seyma', 'ÇAĞLAR', 3);
 10 •
         INSERT INTO okul.ogrenci VALUES (1001, 'Cigdem', 'GEZER', 3);
 11 •
 12 •
          INSERT INTO okul.ogrenci VALUES (1002, 'Süleyman', 'ARSLAN', 4);
         COMMIT;
 13 •
 14
 15
Result Grid
                                                Edit: 🕍 🛗 Export/Import:
                Filter Rows:
                                   danisman
   ogrenci_no
               ad
                          soyad
                         CAĞLAR
   1000
              Seyma
              Çiğdem
                         GEZER
   1001
                         ARSLAN
   1002
              Süleyman
              NULL
                         NULL
                                  HULL
  HULL
```







SAVE POINT

- Tanımlayıcının bir adıyla adlandırılmış bir işlem kayıt noktası ayarlar.
- Geçerli işlemin aynı adı taşıyan bir kayıt noktası varsa, eski kayıt noktası silinir ve yenisi belirlenir.
- Yapılan işlemlerin START TRANSACTION içerisinde yapılması gerekir.

SAVE POINT save_name;

ROLLBACK TO SAVEPOINT save_name;



SAVE POINT

```
INSERT INTO okul.ogrenci VALUES (1000, 'Şeyma', 'ÇAĞLAR', 3);
         INSERT INTO okul.ogrenci VALUES (1001, 'Çiğdem', 'GEZER', 3);
  9 •
         INSERT INTO okul.ogrenci VALUES (1002, 'Süleyman', 'ARSLAN',4);
 10 •
         COMMIT;
 11 •
12
         SELECT * FROM okul.ogrenci ;
 13 •
              Filter Rows:
                                          Edit: 🚄 📆 Export/Import: 🙀
Result Grid
   ogrenci no
             ad
                       soyad
                               danisman
                      CAĞLAR
  1000
             Şeyma
                      GEZER
  1001
             Çiğdem
                      ARSLAN
  1002
             Süleyman
  NULL
                      NULL
                               NULL
```

```
19
          START TRANSACTION;
 20 •
         UPDATE okul.ogrenci SET danisman=1 WHERE ogrenci_no=1000;
 21 •
         SAVEPOINT ogrenci_1000_danisman_atamasi;
 22 •
 23
         UPDATE okul.ogrenci SET danisman=1 WHERE ogrenci_no=1001;
 24
         SAVEPOINT ogrenci_1001_danisman_atamasi;
 25 •
 26
         ROLLBACK TO SAVEPOINT ogrenci_1000_danisman_atamasi;
 27 •
 28
         UPDATE okul.ogrenci SET danisman=2 WHERE ogrenci_no=1002;
 29
         SAVEPOINT ogrenci_1002_danisman_atamasi;
 30 •
         COMMIT;
 31 •
 32
         SELECT * FROM okul.ogrenci;
 33 •
 34
                                               Edit: 🔏 🖶 🖶 Export
Result Grid
               Filter Rows:
   ogrenci_no
                         soyad
                                   danisman
                         ÇAĞLAR
  1000
              Şeyma
              Çiğdem
   1001
                         GEZER
   1002
              Süleyman
                        ARSLAN
                                  2
                                  NULL
  NULL
              NULL
                        NULL
```

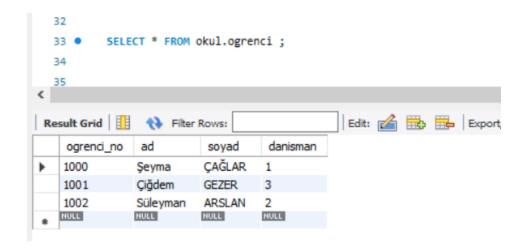


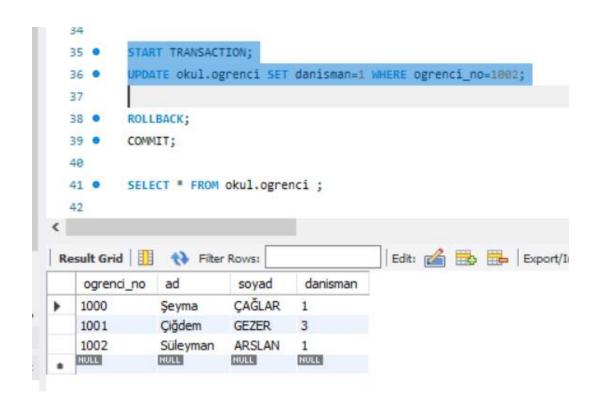
ROLLBACK

- ROLLBACK ile yapılmış bir işlemi geri alabiliriz.
- İşlem COMMIT edilse bile bir önceki haline geri dönüş yapılabilir.
- Fakat AUTOCOMMIT'in kağalı olması gerekir.
- Yapılan işlemlerin START TRANSACTION içerisinde yapılması gerekir.
- Aksi halde ROLLBACK ifadesi çalışmaz.



ROLLBACK







ROLLBACK

```
34
         START TRANSACTION;
 35 •
         UPDATE okul.ogrenci SET danisman=1 WHERE ogrenci_no=1002;
 36 •
 37 •
         ROLLBACK;
 38 •
         COMMIT;
 39
 40 •
         SELECT * FROM okul.ogrenci;
 41
                                              Edit: 🚄 🖶 🖶 Expo
               Filter Rows:
Result Grid
   ogrenci_no
                                  danisman
              ad
                        soyad
                        ÇAĞLAR
   1000
              Şeyma
   1001
              Çiğdem
                        GEZER
                                 3
              Süleyman
                        ARSLAN
   1002
                                 2
                                 NULL
             NULL
                        NULL
  NULL
```



VERITABANI GÜVENLİĞİ İÇİN A.C.I.D



A.C.I.D (Atomicity, Consistency, Isolation, Durability)

- Güvenli bir veritabanı veriler üstünde değişiklik yaparken ACID kuralını sağlamalıdır.
 - 1. Atomicity (Bölünmezlik)
 - 2. Consistency (Tutarlılık)
 - 3. Isolation (İzolasyon)
 - 4. Durability (Dayanıklılık)



Atomicity (Bölünmezlik)

- Transaction daha küçük parçalara ayrılamayan bir işlem birimi olarak ele alınır.
- Transaction bloğu yarım kalmaz.
- Ya hepsi gerçekleşmiş sayılır ya da hiçbir işlem gerçekleşmemiş gibi kabul edilerek en başa dönülür.
 - Veritabanları erişilemez olabilir.
 - Network problemi olabilir.
 - Herhangi bir hata oluşabilir.
- Bu durumda işlem geçersiz sayılacaktır.



Consistency (Tutarlılık)

- Transaction veritabanının yapısını bozmadan işlem bloğunu terk etmelidir.
- Yani ara işlemler yaparken ürettiği işlem parçacıklarının etkisini veritabanında bırakarak, transaction'ı sonlandıramaz.
- Örneğin, birinci hesaptan para azaltıp ikinci hesaba eklemeden transaction sonlandırılamaz.



Isolation (İzolasyon)

- Farklı transaction'lar birbirinden ayrık ele alınmalıdır.
- Aynı anda aynı veri üzerinde birden fazla Transaction değiştirme gereksinimi olabilir.
- Transaction'ların birbirlerinin işlemlerinden etkilenmemesi için işlemler Seri olarak yapılması gerekir.
- Her transaction için veritabanının yapısı ayrı ayrı korunmalıdır.
- İlk transaction tarafından yapılan değişiklikler, ikinci transaction'dan her an görülememeli, sadece bütün işlem gerçekleştiği anda ve bütünü bir anda görülmelidir.



Durability (Dayanıklılık)

- Tamamlanmış transaction'ın hatalara karşı esnek olması gerekir.
- Transaction sırasında fiziksel veya işlemsel bir hata olması durumunda sistemin kendisini bir önceki geçerli veri durumuna döndürebilme kabiliyetidir.
- Elektrik kesilmesi, CPU yanması, işletim sisteminin çökmesi bu kuralları uygulamaya engel olmamalıdır.
- Bunun içinde gerçekleşmiş ve başarılı olarak sonlanmış transaction'ın değişikliklerinin kalıcı olarak diske yansıtılması gerekir.

