



# ACID Prensipleri(SQL)

## 1 Atomicity(Bölünmezlik):

### Terimler:

 **Transaction(İşlem):** SQL mantıksal işlem birimi

 **Rollback(Gerialma):** hatalı işlem sonucunda yapılan işlemi geri alır

 **Commit(Onaylama):** işlemde hata yok ise yapılan değişiklikleri kalıcı olarak kaydeder

### Açıklama:

Bir işlemin tamamen hatasız gerçekleşmesini veya hiç gerçekleşmemesini garanti eder.Eğer işlem sırasında herhangi bir hata ile karşılaşırsa işlem bütünüyle geri alınır.

### Örnek olarak:

```
BEGIN TRANSACTION;

-- A hesabından para çekme
UPDATE Accounts SET balance = balance - 500 WHERE account_id = 1

-- B hesabına para yatırma
UPDATE Accounts SET balance = balance + 500 WHERE account_id = 2



-- İşlem tamamlanırsa
COMMIT;


-- Eğer hata oluşursa
ROLLBACK;
```


Bu örnekte, A hesabından B hesabına para transferi gerçekleştiriliyor. Eğer işlem sırasında herhangi bir hata oluşursa (örneğin, A hesabında yeterli bakiye olmaması), ROLLBACK komutu ile tüm işlem geri alınır ve hesap bakiyeleri değişmeden kalır. İşlem başarılı olursa COMMIT komutu ile değişiklikler kalıcı hale getirilir.

## 2 Consistency(Tutarlılık):

### Terimler:


 **Integrity Constraints(Bütünlük Kısıtlamaları):** Veri Tabanında tutarlılık için kullanılan kurallar.Örneğin(PRIMARY KEY,FOREING KEY,CHECK kısıtlamalar ı.

 **Trigger(Tetikleyiciler):** Veritabanında belirlenen bir olay her çalıştığında otomatik olarak çalıştırılan özel SQL kod parçasıdır.

 **Validation(Doğrulama):** İşlem sırasında verilerin doğruluğunu kontrol edilmesidir

### Açıklama:

Veri tabanının önceden tanımlanmış kurallara uygun kalmasını sağlar.Veritabanında tutarsızlığa yer yoktur. Örneğin:

 Bir stok yönetiminde toplam stok değeri ile depodaki stokların toplamı birbirine eşit olmalıdır.Stokta olmayan bir ürünün siparişine izin verilmemelidir.

### Örnek Senaryo:

```
-- Bütünlük kısıtlamalarıyla tablo oluşturma
CREATE TABLE Customers (
    customer_id INT PRIMARY KEY, customer_name VARCHAR(50)
    NOT NULL, customer_email VARCHAR(50) UNIQUE
);

-- Tutarlılık kontrolüyle veri ekleme
BEGIN TRANSACTION;

-- Müşteri kaydı ekleme
INSERT INTO Customers (customer_id, customer_name, customer_e VALUES (1, 'Ahmet Yılmaz',
'ahmet.yilmaz@example.com');

-- Eğer tutarsızlık varsa geri al
ROLLBACK;
```

Bu örnekte, Customers tablosu oluştururken PRIMARY KEY, NOT NULL ve UNIQUE gibi bütünlük kısıtlamaları kullanılmıştır. Bu kısıtlamalar, verilerin tutarlılığını sağlar:

- PRIMARY KEY Her müşterinin benzersiz bir ID'si olmasını sağlar
- NOT NULL Müşteri adının boş olmamasını garanti eder
- UNIQUE Her müşterinin farklı bir e-posta adresine sahip olmasını sağlar

Eğer bu kısıtlamalardan herhangi biri ihlal edilirse, işlem otomatik olarak geri alınır ve veritabanı tutarlı durumda kalır.

### 3 Isolation( İzolasyon):

#### Terimler:

**Concurrency Control(Eş zamanlılık Kontrolü):** Veri tabanında birden fazla işlemin aynı anda gerçekleşmesi durumunda tutarlılığı sağlar.

**Lock(Kilit)** İzolasyon seviyesine bağlı olarak belirli verilere ulaşımı önler.

#### Transaction Isolation Levels(İzolasyon Seviyeleri):

**Read Uncommitted:** En düşük seviyedir.Bir işlem diğer işlemin verilerini(uncommitted) görebilir.

**Read Committed:** Yalnızca tamamlanmış verileri(committed) görebiliriz.

**Repeatable Read:** İşlem sırasında okunan veriler değiştirilemez.

**Serializable** En büyük izolasyon seviyesidir.Tam sıralama sağlar.

#### Açıklama:

Isolation,işlemlerin karışmasını engeller.Örneğin bir ürün satıldığında stok bilgisi güncellenmeden başka müşteri aynı ürünü almaya çalışmamalıdır.

## Örnek Senaryo:

İki müşteri aynı ürünü satın alsın.

```
-- Müşteri 1 için işlem
SET TRANSACTION ISOLATION LEVEL SERIALIZABLE;

BEGIN TRANSACTION;

-- Stok kontrolü
IF (SELECT stock FROM Products WHERE product_id = 1) > 0 BEGIN

    -- Stok güncelleme
    UPDATE Products SET stock = stock - 1 WHERE product_id

    -- Sipariş oluşturma
    INSERT INTO Orders (customer_id, product_id, quantity) VALUES (1, 1, 1);

END

COMMIT;

-- Müşteri 2 için işlem (beklemede)
SET TRANSACTION ISOLATION LEVEL SERIALIZABLE;

BEGIN TRANSACTION;

-- Stok kontrolü
IF (SELECT stock FROM Products WHERE product_id = 1) > 0 BEGIN

    -- Stok güncelleme
    UPDATE Products SET stock = stock - 1 WHERE product_id

    -- Sipariş oluşturma
    INSERT INTO Orders (customer_id, product_id, quantity) VALUES (2, 1, 1);

END



COMMIT;
```


Bu örnekte, SERIALIZABLE izolasyon seviyesi sayesinde:

- İlk müşterinin işlemi tamamlanmadan ikinci müşteri stok bilgisini göremez
- Her iki işlem de birbirinden tamamen izole edilmiştir
- Stok tutarsızlığı oluşmaz
- İşlemler sırayla gerçekleşir

#### 4 Durability (Dayanıklılık):

##### Terimler:

 **Write-Ahead Logging**  **Yazmadan Önce Günlükleme**): İşlemlerin sonuçlarını veritabanından önce bir log dosyasına kaydeder.

 **Redo Log (Tekrar Günlüğü)**: Sistemin Çökmesi Durumunda tekrar başlatmaya yarayan log.

##### Açıklama:

Genellikle veri tabanı tarafından kullanılan loglar sayesinde bir işlem başarıyla tamamlandıktan sonra yapılan işlemin kalıcı olmasını sağlar. Sistem çökse bile işlemin sonuçları kaybolmaz. İşlem tamamlandığında veri disk üzerinde saklanır. Sistem çökmesi veya güç kaybı durumunda log dosyasından geri yükleme yapılır.

##### Örnek Senaryo:

Bir banka müşterisi hesabına para yükleme işlemini tamandıktan sonra sistem çökse bile yapılan işlem logda tutulur ve kaybolmaz.

```
UPDATE Accounts SET balance = balance + 1000 WHERE account_
```

```
-- İşlemi tamamla  
COMMIT;
```

Bu örnekte:

- Hesaba para yatırma işlemi transaction içinde gerçekleştirilir
- COMMIT komutu ile işlem tamamlanır ve değişiklikler kalıcı hale gelir
- İşlem log dosyasına kaydedilir
- Sistem çökse bile, log dosyası sayesinde işlem kaybı olmaz

### ACID'in Veri Tabanında Kullanılmasının Önemi:

**1≡ Veri Güvenliği:**Veri kaybını ve hatalı işlemlerin önüne geçer.

**2≡ Eş Zamanlı Durumlarda Kolaylık:**Birden fazla kullanıcının aynı anda yaptığı işlemlerde düzgün bir şekilde yönetilmeyi sağlar.

**3≡ Tutarlılık:**Veri tabanı kurallarına ve doğruluğuna uygun çalışmasını sağlar.

**4≡ Sistem çökmelerine karşı dayanıklılık:**Veri tabanı veya sistem çökse bile işlemler kaybolmaz.

Bu nedenlerden dolayı ACID veri tabanı güvenliğinde önemlidir.Özellikle e-ticaret,sağlık veya bankacılık işlemleri gibi alanlarda önemli rol oynar.