SQL Nedir?

SQL (Structured Query Language), ilişkisel veritabanlarını yönetmek için kullanılan bir dildir. SQL'in iki temel alt kümesi bulunur:

1. **DDL (Veri Tanımlama Dili)**
2. **DML (Veri İşleme Dili)**

DDL (Veri Tanımlama Dili):

- Veritabanı yapısını yönetir.
- Tablolar oluşturma, silme, değiştirme gibi işlemleri gerçekleştirir.
- Veritabanı şemasını tanımlar.

DDL komutları şunları içerir:

- **CREATE:** Yeni tablolar, dizinler, görünümler ve kullanıcılar oluşturur.
- **ALTER:** Mevcut veritabanı nesnelerinin yapısını değiştirir.
- **DROP:** Tablolar, dizinler, görünümler ve kullanıcıları siler.
- **GRANT:** Kullanıcılara belirli izinler verir.
- **REVOKE:** Kullanıcılardan izinleri kaldırır.

Örnek DDL komutu:

```
CREATE TABLE customers (
  id INT PRIMARY KEY AUTO_INCREMENT,
  name VARCHAR(255) NOT NULL,
  email VARCHAR(255) UNIQUE
);
```

DML (Veri İşleme Dili):

- Veritabanındaki verileri manipüle eder.
- Verilerin eklenmesi, silinmesi, güncellenmesi ve sorgulanması gibi işlemleri gerçekleştirir.

DML komutları şunları içerir:

- **INSERT:** Veritabanına yeni kayıtlar ekler.
- **UPDATE:** Mevcut kayıtların verilerini değiştirir.
- **DELETE:** Veritabanından kayıtları siler.
- **SELECT:** Veritabanından kayıtları sorgular ve sonuçları döndürür.

Örnek DML komutu:

```
INSERT INTO customers (name, email) VALUES ("Alice Smith", "alice@email.com");

UPDATE customers SET name = "John Doe" WHERE id = 1;
```

```
DELETE FROM customers WHERE email = "jane@email.com";

SELECT * FROM customers;  -- Tüm kayıtları seç
```

WHERE, ORDER BY, GROUP BY, HAVING

WHERE:

- Bir koşula göre verileri filtreler. Koşulu sağlayan sadece satırlar döndürülür.

```
SELECT * FROM users WHERE age >= 18;  -- 18 yaşından büyük kullanıcıları seç
```

ORDER BY:

- Sonuç kümesini bir veya daha fazla sütuna göre sıralar. ASC (artan) veya DESC (azalan) belirtilebilir.

```
SELECT * FROM products ORDER BY price ASC;   -- Ürünleri fiyata göre sırala (en
ucuz önce)
SELECT * FROM orders ORDER BY order_date DESC; -- Siparişleri tarihe göre sırala
(en yeni önce)
```

GROUP BY:

- Bir veya daha fazla sütuna göre verileri gruplar. Sıklıkla toplama fonksiyonları (COUNT, SUM, AVG) ile birlikte kullanılır.

```
SELECT category, COUNT(*) AS product_count
FROM products
GROUP BY category;  -- Kategoriye göre ürünleri say
```

HAVING:

- `GROUP BY`'dan sonra grupları filtreler. Gruplar üzerinde çalışır, bireysel satırlar üzerinde değil.

```
SELECT country, COUNT(*) AS customer_count
FROM customers
GROUP BY country
HAVING customer_count > 100;  -- 100'den fazla müşteri olan ülkeleri göster
```

Unutmayın:

- `WHERE`, gruplama (`GROUP BY`) öncesinde filtreleme yapar.
- `HAVING`, gruplama sonrasında grupları filtreler.

Fonksiyonlar

SQL'de veri manipülasyonu ve alımı için yaygın olarak kullanılan tüm fonksiyonlar bunlardır. İşte her birinin ayrıntıları:

Dize Fonksiyonları:

- **LOWER(dize):** Bir dizeyi küçük harfe dönüştürür.
- **UPPER(dize):** Bir dizeyi büyük harfe dönüştürür.

Örnek:

```
SELECT UPPER(name), LOWER(email) FROM users;
```

Toplama Fonksiyonları:

- **COUNT(*):** Bir tablo veya sonuç kümesindeki satır sayısını sayar (tüm sütunlar).
- **COUNT(sütun_adı):** Belirli bir sütundaki boş olmayan değerlerin sayısını sayar.
- **MAX(sütun_adı):** Bir sütundaki en büyük değeri döndürür.
- **MIN(sütun_adı):** Bir sütundaki en küçük değeri döndürür.
- **SUM(sütun_adı):** Bir sütundaki tüm değerlerin toplamını hesaplar (sadece sayısal veriler için).
- **AVG(sütun_adı):** Bir sütundaki tüm değerlerin ortalamasını hesaplar (sadece sayısal veriler için).

Örnek:

```
SELECT COUNT(*), SUM(price) FROM products;  -- Ürünleri say ve toplam fiyatı
hesapla
SELECT MAX(salary) FROM employees;  -- En yüksek maaşı bul
```

Tarih/Zaman Fonksiyonu:

- **EXTRACT(bölüm FROM tarih_zaman_ifadesi):** Bir tarih veya zaman değerinden belirli bir bölümü çıkarır. Geçerli bölümler arasında `YEAR`, `MONTH`, `DAY`, `HOUR`, `MINUTE`, `SECOND` vb. bulunur.

Örnek:

```
SELECT EXTRACT(YEAR FROM hire_date) AS hire_year FROM employees;
```

Önemli Notlar:

- Dize fonksiyonları, metin verilerini arama, sıralama veya biçimlendirme amacıyla manipüle etmek için sıklıkla kullanılır.
- Toplama fonksiyonları genellikle `GROUP BY` ifadesiyle birlikte kullanılarak gruplar içinde veriyi özetlemek için kullanılır.
- `EXTRACT` fonksiyonu, belirli tarih veya zaman bileşenlerini çıkarmak için kullanışlıdır.

SQL'de Aritmetik ve Mantıksal Operatörler

SQL, hesaplamaları gerçekleştirmek için **aritmetik operatörler** ve sorgularınızda koşulları birleştirmek için **mantıksal operatörler** sunar.

Aritmetik Operatörler:

Bu operatörler, tablolarınızdaki sayısal veriler üzerinde matematiksel hesaplamalar yapmanıza olanak tanır.

- **+ (Toplama):** İki sayıyı toplar.
- **- (Çıkarma):** Bir sayıyı diğerinden çıkarır.
- *** (Çarpma):** İki sayıyı çarpar.
- **/ (Bölme):** Bir sayıyı diğerine böler.
- **% (Mod):** Bölme işleminden sonra kalanı döndürür.

Örnek:

```
SELECT product_id, price * quantity AS total_price
FROM order_items;  -- Toplam fiyatı hesapla (fiyat * miktar)
SELECT age - YEAR(CURDATE()) AS current_age
FROM users;  -- Mevcut yaşını hesapla (yaş - güncel yıl)
```

Mantıksal Operatörler:

Bu operatörler, koşulları birleştirerek sorgularınızı birden fazla kriterle sınırlamanıza yardımcı olur.

- **AND:** Her iki koşulun da doğru olması durumunda genel koşul doğru olur.
- **OR:** En az bir koşulun doğru olması durumunda genel koşul doğru olur.
- **NOT:** Bir koşulun doğruluk değerini tersine çevirir (doğru yanlış olur, yanlış doğru olur).

Örnek:

```
SELECT * FROM products WHERE price > 100 AND category = "Elektronik";  -- $100'den
pahalı "Elektronik" ürünler
SELECT * FROM users WHERE age >= 18 OR is_admin = 1;  -- 18 yaşından büyük
Kullanıcılar VEYA yöneticiler
SELECT * FROM orders WHERE shipped = 0 AND order_date > DATE_SUB(CURDATE(),
INTERVAL 7 DAY);  -- Son bir haftada gönderilmemiş siparişler
```

Önemli Notlar:

- SQL ifadelerinde aritmetik operatörler genellikle mantıksal operatörlerden daha önce değerlendirilir. Gerekirse doğru değerlendirme sırası için parantez kullanın.
- Mantıksal operatörler, karmaşık ve hedefe yönelik SQL sorguları oluşturmak için önemlidir.

Bu operatörleri anlayarak, sayısal verileri manipüle edebilir, koşulları birleştirebilir ve veritabanınızdan ihtiyaç duyduğunuz özel bilgileri alabilirsiniz.

Diğer Operatörler

Bu operatörler, SQL'in `WHERE` ifadesinde belirli kriterlere göre verileri filtrelemek için yaygın olarak kullanılır. İşte her birinin ayrıntıları:

BETWEEN:

- Bir değerin belirli bir aralığa (dahil) düşüp düşmediğini kontrol etmek için kullanılır.

```
SELECT * FROM products WHERE price BETWEEN 50 AND 100;  -- $50 ile $100 arasında
fiyatı olan ürünler
SELECT * FROM orders WHERE order_date BETWEEN '2024-04-01' AND '2024-04-10';  --
2024-04-01 ile 2024-04-10 arasında verilen siparişler
```

IN:

- Bir değerin belirli bir listedeki bir veya daha fazla değere eşit olup olmadığını kontrol eder.

```
SELECT * FROM customers WHERE country IN ('ABD', 'Kanada', 'Meksika');  -- ABD,
Kanada veya Meksika'dan müşteriler
SELECT * FROM products WHERE category IN ('Elektronik', 'Kitaplar', 'Oyuncaklar');
-- Bu kategorideki ürünler
```

LIKE:

- Metin verilerinde desen eşleştirmesi yapar. Herhangi bir karakter dizisini eşleştirmek için joker karakterleri (`%`) veya tek bir karakteri eşleştirmek için alt çizgi (`_`) kullanabilirsiniz.

```
SELECT * FROM users WHERE name LIKE '%Smith';  -- "Smith" içeren isimler
SELECT * FROM products WHERE product_name LIKE 'Kulaklık%';  -- "Kulaklık" ile
başlayan ürünler
SELECT * FROM emails WHERE subject LIKE 'Toplantı _%';  -- "Toplantı " ile
başlayan ve ardından herhangi bir kelime gelen e-postalar
```

IS NULL:

- Bir sütun değerinin null (eksik veya tanımsız) olup olmadığını kontrol eder.

```
SELECT * FROM customers WHERE email IS NULL;  -- E-posta adresi olmayan müşteriler
SELECT * FROM orders WHERE shipped_date IS NOT NULL;  -- Gönderilen siparişler
```

Önemli Notlar:

- `BETWEEN`, belirli bir aralık içindeki verileri filtrelemek için kullanışlıdır.
- `IN`, önceden tanımlanmış bir değer kümesine karşı kontrol sağlar.

- `LIKE`, metin aramalarında desen eşleştirmesi için esneklik sağlar.
- `IS NULL`, tablolarınızdaki eksik verileri belirlemenize yardımcı olur.

Bu operatörler, SQL sorgularınızda belirli verileri filtrelemek ve almak için yeteneklerinizi artırır.