6 April SQL

▼ JOIN

Birbiri ile ilişkili birden çok tablodan veri çekme işlemlerinden birisidir. İkincisi ise subquery'ler.

```
SELECT students.name, students.exam, students.score, tests.passing_score
FROM students
INNER JOIN tests ON students.exam = tests.exam;

/* sorgu sonucunda sütunlar hangi sırayla belirtildi ise o sırayla çıkar. INNER
yazılmazsa sadece JOIN yazılırsa da INNER JOIN olarak çalışır (default'u bu şekilde).
Burada altta yazdığımız = de soldaki foregin key sağdaki ise primary key olmak zorunda
SQLite zorunluluğu!!!!.FK ile PK nın ortak olanlarını alır.*/
```

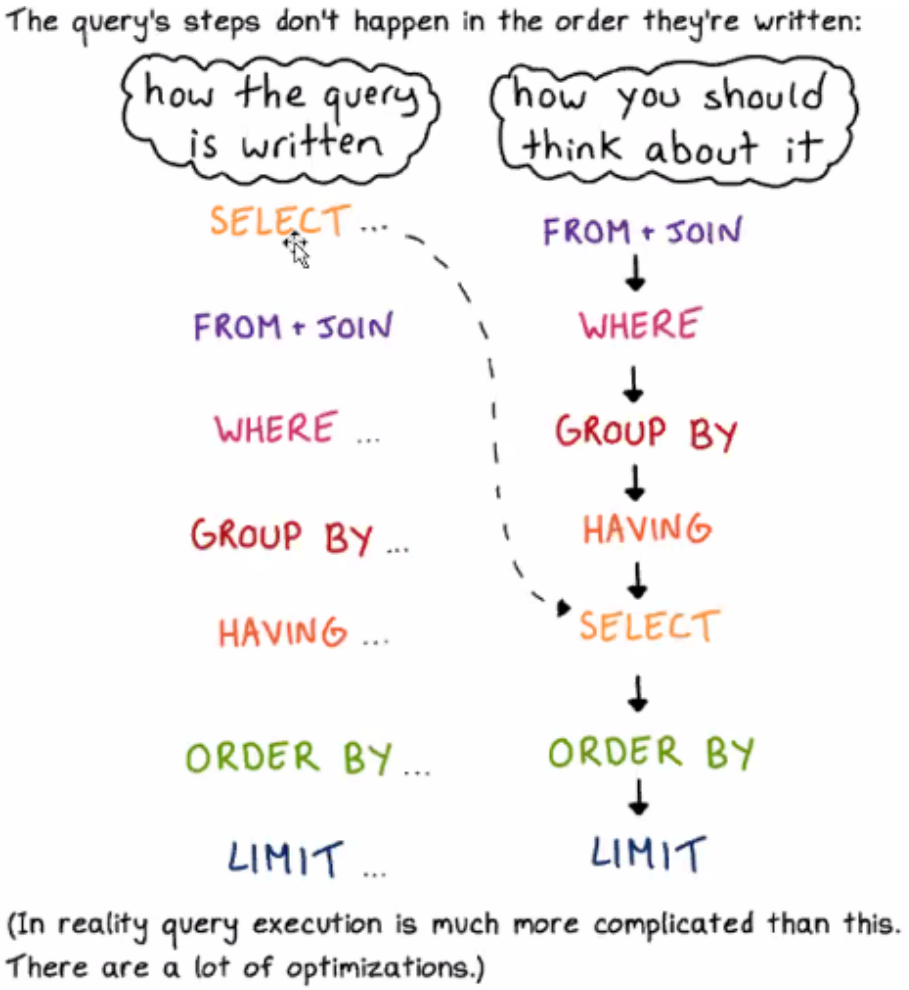
```
/*=====================================================
 JOINS
=====================================================*/

-- Join islemleri farkli tablolardan secilen sutunlar ile yeni bir tablo
-- olusturmak icin kullanilabilir.
--
-- JOIN islemleri Iliskisel Veritabanlari icin cok onemli bir ozelliktir. Çunku
-- Foreign Key'ler ile iliskili olan tablolardan istenilen sutunlari cekmek
-- icin JOIN islemleri kullanilabilir.

-- Standart SQL'de en çok kullanılan Join islemleri:
-- 1) FULL JOIN:  Tablodaki tum sonuclari gosterir
-- 2) INNER JOIN:  Tablolardaki ortak olan sonuc kumesini gosterir
-- 3) LEFT JOIN:  Ilk tabloda (Sol) olan bütün verilerini gosterir
-- 4) RIGHT JOIN: Sadece Ikinci tabloda olan tum sonuclari gosterir.

--  NOT: SQLite Sadece INNER, LEFT VE CROSS JOIN İşlemlerini desteklemektedir.

/*=====================================================*/
```

> Yazım

ve

çalıştırma

sırası



▼ Subqueries

İnner query de (iç sorguda) bir sorgu yaptırıyoruz ve onu Outer query de (dış sorguda) kullanmamızı sağlar

```
SELECT column_name
FROM table_1, table_2
WHERE column_name OPERATOR (
      SELECT column_name
      FROM table_1, table_2);
```

1. Single-row subqueries ⇒ <,>, >=,<> gibi işaretler kullanıyoruz.
2. Multiple-row subqueries ⇒ birden fazla değer döndüğü için “IN” veya “NOT IN” kullanıyoruz.

Bugünkü Çalışma

```
/*=================================================
 JOINS
===================================================*/

-- Join islemleri farkli tablolardan secilen sutunlar ile yeni bir tablo
-- olusturmak icin kullanilabilir.
--
-- JOIN islemleri Iliskisel Veritabanlari icin cok onemli bir ozelliktir. Çunku
-- Foreign Key'ler ile iliskili olan tablolardan istenilen sutunlari cekmek
-- icin JOIN islemleri kullanilabilir.

-- Standart SQL'de en çok kullanılan Join islemleri:
-- 1) FULL JOIN:  Tablodaki tum sonuclari gosterir
-- 2) INNER JOIN:  Tablolardaki ortak olan sonuc kumesini gosterir
-- 3) LEFT JOIN:  Ilk tabloda (Sol) olup digerinde olmayan sonuclari gosterir
-- 4) RIGHT JOIN: Sadece Ikinci tabloda olan tum sonuclari gosterir.

--  NOT: SQLite Sadece INNER, LEFT VE CROSS JOIN İşlemlerini desteklemektedir.

/*=================================================*/


/* araclar.db adındaki veritabanını kullanarak Markalar ve Siparisler
tablolarındaki marka_id'si ayni olan kayıtların marka_id, marka_adi,
siparis_adedi ve siparis_tarihi bilgilerini  listeleyen bir sorgu yaziniz.*/

SELECT * from "araclar".markalar; /*markalar adlı başka tablo olursa
bu şekilde kullanılması uygun olur.*/
SELECT * FROM siparisler;

 SELECT markalar.marka_adi,
    markalar.marka_id,
     siparisler.siparis_adedi,
     siparisler.siparis_tarihi
FROM markalar
INNER JOIN siparisler
ON markalar.marka_id = siparisler.marka_id;

-- Daha kısa olarak;
SELECT m.*,
     s.siparis_adedi,
     s.siparis_tarihi
FROM markalar m   -- as kullanmamıza gerek yok. Uzun olmasın diye tercih edilir.
INNER JOIN siparisler s
ON m.marka_id = s.marka_id;


/* Markalar ve Siparisler tablolarındaki tüm araç markalarının siparis
bilgilerini (marka_id,marka_adi,siparis_adedi,siparis_tarihi) listeleyen
bir sorgu yaziniz.*/
```

```
SELECT m.marka_id,
     m.marka_adi,
     s.siparis_adedi,
     s.siparis_tarihi
FROM markalar m
LEFT JOIN siparisler s
ON m.marka_id = s.marka_id;

/* Chinook veritabanındaki tracks tablosunda bulunan her bir şarkının
türünü (genre) listeleyiniz.*/

SELECT  t.name,
    g.name
FROM tracks t
INNER JOIN genres g
ON t.GenreId = g.GenreId;


/* invoice tablosundaki faturaların her birinin müşteri adını (FirstName),
soyadını (lastName), fatura tarihini (InvoiceDate) ve fatura meblağını
(total) listeleyen sorguyu yazınız */

SELECT c.FirstName,
     c.LastName,
     i.InvoiceDate,
     i.total
FROM invoices i
INNER JOIN customers c
ON i.CustomerId = c.CustomerId;

--group by çözümü. 412 olan row sayısı 57 e düştü. Aggregate fun. kullanılması
-- mantıklı olur.
SELECT c.FirstName,
     c.LastName,
     i.InvoiceDate,
     i.total  --sum(i.total) as total_amount
FROM invoices i
INNER JOIN customers c
ON i.CustomerId = c.CustomerId
GROUP by c.FirstName;

 /* invoice tablosundaki faturaların her birinin müşteri adını
(FirstName),soyadını(lastName) ve fatura meblağlarının
toplamının(total) 40 dolardan fazla olanlarını azalan sırada
listeleyen sorguyu yazınız */

SELECT c.FirstName,
     c.LastName,
     sum(i.total) as total_amount
FROM invoices i
INNER JOIN customers c
ON i.CustomerId = c.CustomerId
GROUP BY c.FirstName
```

```
HAVING total_amount > 40
ORDER BY total DESC;

--alternatif olarak CustomerId ye göre gruplandırırsak...
SELECT c.CustomerId,
     c.FirstName,
     c.LastName,
     sum(i.total) as total_amount
FROM invoices i
INNER JOIN customers c
ON i.CustomerId = c.CustomerId
GROUP BY c.CustomerId
HAVING total_amount > 40
ORDER BY total DESC;

/*================================================
 SUBQUERIES
================================================*/

/* albums tablosundaki Title sutunu 'Faceless' olan kaydın albumid'si elde
ederek tracks tablosunda bu değere eşit olan kayıtların bilgilerini SUBQUERY
yazarak listeyiniz. Listelemede trackid, name ve albumid bilgilerini
bulunmalıdır. */

SELECT TrackId, name, AlbumId --benim çözüm
FROM tracks
where AlbumId IN (select AlbumId
        FROM albums
        where Title='Faceless');

--hard-coded yöntem, tek bir blokla çözmeliyiz. subquery uygun.
SELECT AlbumId
FROM albums
WHERE Title = 'Faceless';

SELECT TrackId, name, AlbumId
FROM tracks
WHERE AlbumId = 88;
-- subquery yöntemi. = kullanmamızın sebebi tek değer döndüğünden dolayı
-- birden fazla değer dönecekse = yerine in kullanabilirdik
SELECT TrackId, name, AlbumId
FROM tracks
WHERE AlbumId = (
        SELECT AlbumId
        FROM albums
        WHERE Title = 'Faceless');

-- JOIN
SELECT t.TrackId,
     t.name,
     t.AlbumId
FROM tracks t
JOIN albums a
ON t.AlbumId = a.AlbumId
```

```
WHERE a.Title = 'Faceless';
--and li çözüm.
SELECT t.TrackId,
     t.name,
     t.AlbumId
FROM tracks t
JOIN albums a
ON t.AlbumId = a.AlbumId AND a.Title = 'Faceless';
```

ÖDEVLER

```
/* albums tablosundaki Title sutunu Faceless veya Let There Be Rock olan kayıtların
albumid'lerini elde ederek tracks tablosunda bu id'lere eşit olan kayıtların bilgiler
ini
SUBQUERY kullanarak listeyiniz. Listelemede trackid, name ve albumid bilgileri bulunm
alıdır. */
```

```
/* albums tablosundaki Title sutunu Faceless veya Let There Be Rock olan kayıtların
albumid'lerini elde ederek tracks tablosunda bu id'lere eşit olan kayıtların bilgiler
ini
JOIN kullanarak listeyiniz.Listelemede trackid, name ve albumid bilgileri bulunmalıdı
r. */
```