Laboratuvar 5
BİRLEŞTİRME VE ALT SORGULAR

Veritabanında verileri saklamak için birçok tablo bulunmaktadır. Bazı zamanlar ihtiyaç duyulan bilgileri almak için birden fazla tablodan yararlanmak gerekir. Bu gibi durumlarda ilgili tablolara erişmek için SQL'in birleştirme yeteneği kullanılır. Aslında birleştirme özelliği ilişkisel veri tabanlarının en önemli özelliğidir. Birleştirme özelliği sayesinde kullanıcı ihtiyaçlarına uyan ileri düzeyde SQL cümleleri yazma imkanı olur.

Zaman zaman değişik nedenlerden dolayı SQL cümleleri içerisinde başka SQL cümleleri yazma ihtiyacı olur. Bu şekilde SQL cümlesi içerisinde yer alan SQL cümlelerine altsorgu denir.

Basit Birleştirme İşlemleri

Birleştirme işlemi yapılırken iki tabloyu ilişkilendirmek için en çok kullanılan operatör (=) eşittir. Eğer iki tabloyu ilişkilendirirken eşittir (=) operatörü kullanılıyorsa bu birleştirme işlemine eşitlik birleştirmesi (*equijoin*) denir. Bu çeşit birleştirme işleminde belirtilen sütunlardaki verileri eşit olan kayıtlar birleştirilir. Bu basit birleştirme işlemi ayrıca dahili birleştirme (*inner join*) olarak da bilinir. Çünkü sadece eşitlenen kayıtlar sorgu sonucunda döner.

Örneğin aşağıda Employees ve Departments tabloları kullanılan eşitlik birleştirmesi ile verilerin tablolarıdan çekiliyor. Burada dikkat edilmesi gereken her iki tabloda da DEPARTMENT_ID sütunlarının bulunmasıdır. Eşitlik işlemi bu sütunlar ile yapılmıştır. Sorgu çalışırken DEPARTMENT_ID değeri her iki tabloda da aynı olan kayıtları birleştirerek getirmiştir.

```
SQL> desc employees
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 EMPLOYEE_ID                               NOT NULL NUMBER(6)
 FIRST_NAME                                         VARCHAR2(20)
 LAST_NAME                                 NOT NULL VARCHAR2(25)
 EMAIL                                     NOT NULL VARCHAR2(25)
 PHONE_NUMBER                                       VARCHAR2(20)
 HIRE_DATE                                 NOT NULL DATE
 JOB_ID                                    NOT NULL VARCHAR2(10)
 SALARY                                             NUMBER(8,2)
 COMMISSION_PCT                                     NUMBER(2,2)
 MANAGER_ID                                         NUMBER(6)
 DEPARTMENT_ID                                      NUMBER(4)

SQL> desc departments
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 DEPARTMENT_ID                             NOT NULL NUMBER(4)
 DEPARTMENT_NAME                           NOT NULL VARCHAR2(30)
 MANAGER_ID                                         NUMBER(6)
 LOCATION_ID                                        NUMBER(4)

SQL> select employees.last_name, departments.department_name
  2  from employees , departments
  3  where employees.department_id=departments.department_id;
```

```
LAST_NAME                 DEPARTMENT_NAME
------------------------- ------------------------------
OConnell                   Shipping
Grant                      Shipping
Whalen                     Administration
Hartstein                  Marketing
Fay                        Marketing
Mavris                     Human Resources
Baer                       Public Relations
Higgins                    Accounting
Gietz                      Accounting
King                       Executive
Kochhar                    Executive
```

Burada DEPARTMENT_ID sütunu belirtilirken sütunun başında tablo ismi ile belirtilmiş olduğuna dikkat ediniz. Sütunun isimlerini bu şekilde belirtmek sütunun hangi tabloya ait olduğu konusundaki karışıklığı önleyeceği gibi SQL cümlesinin okunabilirliğini de arttırır. Eğer aynı sütun ismine sahip birden fazla tablo varsa her bir sütun ismi tablosu ile birlikte belirtilmelidir.

Üç veya daha fazla tablolu birleştirme işlemi yer alan bir SQL cümlesi çalıştırıldığı zaman, Oracle aşağıdaki adımları adımları izler:

1. Oracle öncelikle iki tabloyu belirtilen koşullara göre karşılaştırıp birleştirir.
2. Oluşturulan bu yeni sonucu bir sonraki tablo ile belirtilen koşullara göre birleştirir.
3. Bu işlem tüm tabloların birleştirme işlemi tamamlanıncaya kadar tekrarlanır.

Kompleks Birleştirme İşlemleri

Birleştirme işlemi yapılırken iki tabloyu ilişkilendirmenin yanında sorgu sonucu dönen kayıtları sınırlandırmak için de WHERE kısmında koşullar yazılabilir. Bu tip birleştirme işlemleri kompleks birleştirme olarak bilinir.
Örnek:

```
SQL> select employees.last_name, departments.department_name
  2  from employees , departments
  3  where employees.department_id=departments.department_id and
  4  upper(employees.job_id) IN ('IT_PROG', 'IT_MAN');
```

Tablo Kısaadlarının Kullanılması

Önceki bölümlerde sütun isimlerinde kısaadlar kullanılabildiğini öğrenmiştik. Bunun gibi SQL cümlelerinde tablolara da kısaadlar vermek ve bunları kullanmak mümkündür. Tablo kısaadlarını belirtmek için tablo isminden sonra bir boşluk bırakılarak kısaad yazılır. Bir önceki SQL in tablo kısaadları ile yazılmış şekli aşağıdadır.

```
SQL> select e.last_name, d.department_name
  2  from employees e, departments d
  3  where e.department_id=d.department_id and
  4  upper(employees.job_id) IN ('IT_PROG', 'IT_MAN');
```

OUTER JOIN

Tablolar arasında INNER JOIN birleştirmesi yapıldığı zaman ilişki kurmak için belirtilen sütunların verileri birebir aynı olduğu zaman bu kayıtlar birleştirilerek getirilmekte idi. Ancak bir tabloda yer alan kayıta karşılık diğer tabloda eşleneck bir kayıt bulunmaz ise bu kayıt da sorgu sonucunda getirilmez. Diğer bir deyişle INNER JOIN kullanıldığı zaman sadece diğer tabloda eşleniği olan kayıtlar karşımıza gelmektedir. Bazı durumlarda karşı tabloda kayıtların eşleniği olmasa da getirilmesi istenir.Bu durumlarda Oracle bize OUTER JOIN mekanizması sunmaktadır. OUTER JOIN için geleneksel Oracle sözdiziminde eksik kayıtların bulunduğu tablo tarafında ilişki ifadelerinde (+) işareti konur.

SORGU 1:

```
select e.last_name, e.department_id, d.department_name
from employees e, departments d
where e.department_id=d.department_id(+)
```

SORGU1 de herhangi bir bölüme dahil olmayan çalışanlar da çıktı sonucunda listelenecektir.

SORGU 2:

```
select e.last_name, e.department_id, d.department_name
from employees e, departments d
where e.department_id(+)=d.department_id
```

SORGU2 de ise çalışanı olmayan tüm bölümler de çıktı sonucunda listelenecektir.

Eşitsizlik Birleştirmesi (Non-Equijoin)

Eşitlik birleştirmesinin (*equijoin*) daha önce = operatörü ile yapıldığını öğrendik. Eşitsizlik birleştirmesi = operatörü haricinde herhangi bir operatör kullanılması ile yapılır. Örneğin <=, >=, <> (Sırasıyla; küçük-eşit, büyük-eşit, eşit-değil) operatörleri gibi.

Örnek:

```
SQL> select e.last_name, d.department_name
  2  from employees e, departments d
  3  where e.department_id < > d.department_id and
  4  upper(employees.job_id) IN ('IT_PROG', 'IT_MAN');
```

Alt Sorgular

Alt sorgu bir sotgu içerisindeki sorgu demektir. Alt sorgu problemin bir kısmının çözümünü verir, diğer kısmı ise ana sorguda çözüme kavuşur. Örneğin Employees tablomuzda yer alan çalışanlardan kimler yine Employees tablomuzda yer alan 'OLSEN' soyisimli çalışandan yüksek maaş almaktadır diye bir soru yöneltildiği zaman soruyu iki parça halinde değerlendirmemiz gerekir. Birinci parça; Olsen ne kadar maaş almaktadır, İkinci parça ise kimer Olsen'in maaşından fazla almaktadır.

Tek-Kayıt Döndüren Alt Sorgular

Tek kayıt alt sorguları sorgu sonucunda sadece bir kayıt döndürürler. Tek kayıt alt sorgularını ana sorgu ile birleştirirken eşittir (=) vb. tek kayıt operatörleri kullanılabilir.

Örneğin:

```
SQL> select first_name,last_name, salary
  2  from employees
  3  where salary = (select max(salary)
  4                  from employees);
```

Bu sorguda ilk önce Employees tablosundan alt sorgu ile en yüksek ücret değeri alınmaktadır. Daha sonra ana sorguda bu en yüksek ücreti alan kişi listelenmiştir.

Çoklu-Kayıt Döndüren Alt Sorgular

Çoklu kayıt alt sorguları sorgu sonucunda birden fazla kayıt döndüren sorgulardır. Çoklu kayıt sorgularını ana sorgu ile birleştirirken çoklu kayıt operatörleri (IN, ANY, EXISTS, ALL vb.) kullanılır. Örneğin aşağıdaki sorguda alt sorgu birden fazla kayıt döndürmektedir. Sonuçta ana sorguda alt sorgu sonucunda dönen kayıtlardaki koşullara uyan kayıtları getirmektedir.

```
SQL> Select department_id
       From  employees
       Where upper(last_name)='TAYLOR';

DEPARTMENT_ID
-------------
         60
         70
 SQL>
  1  select last_name
  2  from employees
  3  where department_id IN (select department_id
  4                          from employees
  5                          where upper(last_name)='TAYLOR';



LAST_NAME
-------------------------
Abel
Ande
Austin
Baer
Banda
Bates
Bernstein
Bloom
Cambrault
```

Yukarıdaki sorguda listelenen çalışanlar 'TAYLOR' soyisimli çalışan ile ayni bölümde çalışanlardır. IN operatörünü kullanmamızın amacı da içteki sorgunun birden fazla sonuç döndürmesinden kaynaklanmaktadır.