NULL;-- *işlem yapmadan devam eder.*

END IF;

END;

Örnek: A grubundaki takımların puanlarının ortalaması 10'dan küçük ise A grubundaki takımlara 3'er puan ekleyen yoksa işlem yapmadan devam eden kodu yazınız.

```
DECLARE
toplam NUMBER := 0;
BEGIN
select avg(puan) into toplam
FROM TAKIMLAR
where GNAME='A';

IF toplam < 10 THEN
 UPDATE TAKIMLAR
 SET PUAN = PUAN+3
 WHERE GNAME ='A';
ELSE
 NULL;
END IF;
END;
```

PL/SQL'de İş Blokları(Transaction)

PL/SQL'de transactionlar sistem tarafından yönetilen bir yapıdır. Veri bütünlüğünü ve doğruluğunu sağlamak amaçlı iş blokları oluşturulur. İstenildiği takdirde kullanıcılarda transcation blokları oluşturabilirler. Transaction mantığı için en güzel örnek, bir ATM'den para çekme işlemi sırasında bir banka bir transaction oluşturur. Parayı çekip işleminiz bitmeden bir elektrik kesilmesi ya da başka bir sorunla karşılaştığında sistem hemen bloke olacak ve yapılan işlemler geri alınacaktır. Genel olarak üç ifadesi vardır. Bunlar:

- **COMMIT:** Yapılan işlemlerin doğruluğu sonucu işleme onay verilmesi işlemidir.
- **ROLLBACK:** Yapılan işlemler sonucunda bir sorunla karşılaşıldığında istenilen noktaya ya da işlemin başından tüm yapılanları geri alma işlemidir.
- **SAVEPOINT:** İş bloğu içerisinde istenilen yere dönüş noktaları koyulabilir. Buna örnek bilgisayar oyunlarında her geçilen seviyeden sonra kayıt ederek, tekrar oyuna girdiğinizde o seviyeden ya da kayıtlar içinden istediğiniz yerden oyuna devam etmeyi sağlayan işlemdir.

Örnek: Bu örneğimiz transaction işlemlerinin hepsini içeren bir örnek için tasarlanmıştır. Anlatımlı bir örnektir.

```
DECLARE
takim number:=100;

BEGIN
savepoint Evrel;

update TAKIMLAR
set puan=puan+3
where gname='B';

insert into Takimlar values(takim,'Man. United','03/10/1908','B','1000000',0);
commit;

EXCEPTION
WHEN DUP_VAL_ON_INDEX THEN
rollback to Evrel;

END;
```

Bir İşlem Noktası oluşturuluyor.

Bir İşlemler onaylanıyor.

Böyle bir aykırı durum oluştuğunda Evre1 noktasına geri döndür

Not: Rollback to Evre1 yerine direk **Rollback** yazarsak yine aynı işlemi yapardı. İşlemlerin hepsi geri alınırdır. Fakat aynı şey değildir. Buna dikkat etmeliyiz. **Evre1** noktası işlem gövdesinin ortasında olsa oraya kadar işlemler geri alacaktır.

İndeksler(Indexes)

İndeks yapıları özellikle tablolar üzerinde bir kaydı aramak için çok kullanılan yapılardır. Bunun yanında SQL cümlelerinin performansını arttırmak için tablolarda çok sorgulanan alanlara tanımlanırlar. SQL sorgularında bu tanımlanan alanların SELECT veya WHERE kısmında bulunmaları gerekmektedir. Bir tabloda indeks olmaması, işlemlerde tüm tablo verilerinin taranması anlamına gelmektedir. Bu istenmeyen bir durumdur. Bunun yanında INSERT cümleciklerinde İndeksler sistemi yavaşlatmaktadır. Sebebi girilen kayıta göre tüm indekslerin güncellenmesi gereğidir. Oracle'ın indeks yapısı default olarak B-Tree yapısıdır. İndeks tanımlandığında, tanımlı alanlar bir segmentte tutulur. Aranan değer B-Tree arama algoritmasına göre bulunarak onun ROWID değerine göre tablodan değer çekilir.

PL/SQL'de indeks tanımlamak için söz dizimi;

CREATE [UNIQUE | BITMAP] INDEX *adı – Tekillik ya da BITMAP indeksleme yapısı seçilebilir.*

ON *tablo adı (sütun1,stüun2,..)*

[REVERSE];-- *Ters anahtar indeksi oluşturur.*

Örnek: Takımlar tablosunda takım adı üzerinde bir index oluşturalım.

```
CREATE INDEX takim_Ad_ndx
ON TAKIMLAR(isim);
```

Not: Bir tablo oluşturulduğunda birincil anahtar alanına sistem tarafından indeks oluşturulur. **DROP** komutu ile indeks yok edilebilir, **ALTER** komutu ile üzerinde değişiklikler yapılabilir.

Prosedürler(Yordamlar)

Prosedürler ya da başka deyişle yordamlar alt programlar olarak bilinirler. Birden fazla değer döndürebilirler. Belirli işlevleri yeri getirmek amaçlı kullanılırlar. **CREATE PROCEDURE** ile bir yordam oluşturularak **BEGIN – END** bloğu arasında istenen işlemler, kodlar yürütülebilir.

CREATE OR REPLACE PROCEDURE *adı*
(*Parametreler, değişkenler*) *– Parametrelerin modu (IN, OUT, IN OUT) verilebilir.*
AUTHID *current_user – Hangi kullanıcının bu procedure üzerinde yetkisi olduğunu belirtir*
AS
BEGIN *– Bloğun başı*
İşlemler, komutlar
[EXCEPTION] *-- İstenmeyen bir durum oluştuğunda Exception kısmı çalışacaktır.*
Aykırı durumlar
END *isim (opsiyeneldir)***;** *--Bloğun sonu*

Örnek: Dışarıdan değer olarak takımın adı ve grubunu alan, geriye bu takımın puan değerini döndüren procedure kodunu yazınız.

Örnek: Girilen karakter tipindeki iki ifadeyi yan yana yazan prosedürü kodlayınız.

```
CREATE OR REPLACE PROCEDURE puan_goster
(
takim IN VARCHAR2,
grup IN varchar2,
dmg out number
)
AS
BEGIN

select puan into dmg
from TAKIMLAR
where gname=grup and isim=takim;

EXCEPTION
WHEN VALUE_ERROR THEN
dbms_output.put_line('Değerlerde Sorun Var..');

END puan_goster;
```

```
CREATE OR REPLACE PROCEDURE Sayı_Ekleme
(
sayı IN VARCHAR2, yeni OUT VARCHAR2
)
AS
BEGIN
yeni := sayı || sayı;
EXCEPTION
WHEN VALUE_ERROR THEN
dbms_output.put_line('Değerlerde Bir Sorun Var..');
END;
```

Not: || operatörü iki karakter ya da string ifadeyi yan yana yazmayı sağlar. Karakterleri birleştirir.

Fonksiyonlar(Functions)

Fonksiyonlar yapı olarak prosedürlere benzeyen alt yordamlardandır. Fakat RETURN kelimesi ile söz dizimleri birbirinden ayrılır. Bir değeri hesaplayıp döndürmek amaçlıdırlar. CREATE FUNCTION komutu ile oluşturulurlar. Fonksiyon içinde RETURN ifadesine nerede rastlanırsa fonksiyon orada sonlanır. Fonksiyonlar SQL cümleleri ile de kullanılabilir.

FUNCTION *adı*
RETURN *[veri türü]—Dönecek değerin veri türü belirtilir.*
IS
BEGIN – *Bloğun başlangıcı*
komutlar
RETURN *[değer];--Dönecek değer buraya yazılır*
[EXCEPTION] -- *İstenmeyen bir durum oluştuğunda Exception kısmı çalışacaktır.*
END; -- *Bloğun Sonu*

Örnek: Dışarıdan değer olarak takımın adı girildiğinde ‘A’ grubunda ise bütçesine %30 turlama ücreti yansımış şekilde geri dönen fonksiyon kodunuz yazınız.

Örnek: Girilen sayının karekökünü döndüren basit ve anlaşılır kodu yazınız.

```
CREATE OR REPLACE  FUNCTION TUR_UCRET (takim varchar)
RETURN number
IS
ucret number;
BEGIN

select (to_number(butce)*1.3) into ucret
from TAKIMLAR
where gname='A' and isim=takim;

  RETURN ucret;

END TUR_UCRET;
```

```
CREATE OR REPLACE FUNCTION karekok (sayi NUMBER)

RETURN NUMBER

IS

BEGIN

RETURN POWER(sayi,2);

END;
```

Not: to_number parametresi karakter tipindeki bir alanı sayıya çevirir. **POWER** fonksiyonu ise bir sayının istenilen sayıda üssünü alır.

Görünümler (Views)

Görünümler(View), tablolardaki verilerin bir veya birden fazla tablodan istenildiği şekilde bir arada sunulmasını sağlayan tanımlanmış sorgulardır. Aynı tablolar gibi satırlar ve sütunları içeren sanal tablolar olarak da tanımlanabilir. Tablolardan fakı ise tablolar verileri tutarak veri tabanında yer kaplarlar fakat view veri depolamaz, yer tutmazlar. Sadece SQL sorgusu veri tabanın veri sözlüğünde tutulmaktadır. Bunun yanında içerisinde veri barındırabilen ve sürekli olarak güncellenebilen görüntü türüne **materialized view** denmektedir. View ile bir ya da birkaç tablodan seçtiğimiz verileri okuyabiliriz hatta bazı durumlarda veri girişi, veri güncelleme de yapabiliriz. Aslında View için ana tablolardan veri getiren bir işaretçi olarak da ifade edebiliriz. View yapısının bize avantajları listelersek;

- Birden çok tabloyla(join işlemi) çalışırken uzun ve karmaşık SQL ifadelerinden kurtulmayı sağlar.
- Veriye ulaşmada performans arttırır.
- Verinin istenmeyen tarafına erişimini sınırlar. Buna bir örnek vermek gerekirse bir personel tablosuna erişen şirketin müdürü tüm personel alanlarını listeleyebilirken, müdür yardımcısı ve diğer yetkililerin personel tablosundan maaş alanını görmesi istenmeyebilir, işte o zaman view oluşturarak diğer yetkililerin personel tablosunun yerine view'i sorgulamasını sağlayabiliriz

- Veriyi kontrol altında tutar.

  Görünümü daha iyi anlayabilmek için temel tablo ve görünüm ilişkisini anlatan sıradaki şekle bakabilirsiniz.

Temel Tablo

	TKNO	ISIM	KUR_TAR	GNAME	BUTCE	PUAN
1	1	Real Madrid	01/01/1880	A	1000000	3
2	2	Beşiktaş	21/01/1901	A	1000000	0
3	3	Cska Moskova	18/01/1900	A	1100000	0
4	14	Chealsea	01/08/1910	B	800000	1
5	13	Juventus	11/01/1920	B	10000000	1
6	11	Dmg United	03/10/1978	B	2000000	3

Görünüm(view)

	TKNO	ISIM	PUAN
1	1	Real Madrid	3
2	2	Beşiktaş	0
3	3	Cska Moskova	0
4	14	Chealsea	1
5	13	Juventus	1
6	11	Dmg United	3

View yapısının söz dizimi aşağıdaki gibidir.

CREATE [OR REPLACE] VIEW *görüntü ismi*

(takma isimler)
AS

SQL Sorgusu
[WITH [READ ONLY |CHECK OPTION | CHECK OPTION CONSTRAINT *kısıtlama ismi***]]**

Örnek: A ve B grubundaki takımların isim ve puanlarını listeleyen view oluşturunuz.

Örnek: A ve B grubundaki takımların isim ve puanlarını azalan sırada ve takma isim kullanarak listeleyen view oluşturunuz.

```
CREATE VIEW PuanList
AS

SELECT isim,puan

FROM TAKIMLAR

WHERE GNAME in ('A','B');
```

```
CREATE VIEW PuanList (TakimAdi , Puan)
AS

SELECT isim,puan

FROM TAKIMLAR

WHERE GNAME in ('A','B')

ORDER BY puan desc;
```

Örnek: A grubundaki takımların adlarını ve maç sonuçlarını listeleyen view oluşturunuz.

```
CREATE VIEW  OR REPLACE Macsonucu
AS
SELECT t1.ISIM,m.GOL_EVSAHIBI,t2.ISIM,m.GOL_KONUK

FROM TAKIMLAR t1, TAKIMLAR t2, MACLAR m

WHERE m.TKM_ID_EV=t1.TKNO and t2.TKNO=m.TKM_ID_KONUK and t1.GNAME='A'

WITH READ ONLY;
```

Not: Bunun yanında view silmek için DROP VIEW ve değiştirmek için ALTER VIEW komutu kullanılmaktadır. DROP ve ALTER komutları kitap içinde anlatılmıştır.

Tetikleyiciler(Triggers)

Tetikleyiciler (trigger) , veri tabanında bir işlem yapıldıktan sonra ve öncesinde ya da süreçler içinde işleme alınan PL/SQL blokları ya da işlemleridir. Tetikleyici veritabanına yük bindirir. Bunun için tablo kısıtlamaları ya da diğer nesnelerle yapabileceğimiz işlemleri tetikleyici kullanarak yapmamak gerekir. Fakat bunun yanında tetikleyiciler ile bir çok işlem yapılabilir. Bunlar için senaryolar vermek gerekirse;

- Bazı saatlerde veri tabanında yapılan işlemlere izin vermemek olabilir.
- Yılı dolan personelin maaşına otomatik zam yapmak verilebilir.
- Veri eklenildiğinde, silindiğinde ya da güncelliğinde yapılacak işlemler verilebilir.

Tetikleyiciler genel olarak üç grupta toplanabilir.

- Before, After Tetikleyicileri
  - Kullanıcı giriş, çıkışları, DDL cümleleri (CREATE, ALTER ve DROP, veritabanı ve şema) ve DML cümleleri (DELETE, UPDATE, INSERT, tablo işlemleri) için tasarlanmıştır.
- Instead Of Tetikleyicileri
  - Görünümler için tasarlanmıştır.
- Sistem Tetikleyicileri
  - Bir sistem olayında tasarlanmıştır. Bunlar veritabanının başlatılması, sonlandırılması ya da sunucu(server) hata mesajlarıdır.

Tetikleyiciler konusunda, bu kitapta Before ve After tetikleyicileri üzerine örnekler verilmiştir. Daha fazla bilgi ve örnek için http://otn.oracle.com/tech/pl_sql sitesine bakabilirsiniz.

CREATE OR REPLACE TRIGGER *adı*

AFTER | BEFORE | INSERT | INSTEAD OF *olay adı*

ON *tablo adı*

FOR EACH ROW -- *Her satır için çalıştırılır.*

Şart (WHEN ile istenilen şart olursa tetikle denebilir..)

BEGIN

Yapılacak işlemler, komutlar

END;

Örnek: Maçlar tablosuna yeni maç girildiğinde sonuçlara bakarak takımların puanlarını güncelleyen tetikleyici kodunu yazınız.

```
CREATE OR REPLACE TRIGGER PuanEkleme
     AFTER INSERT ON MACLAR
     FOR EACH ROW
     BEGIN
      IF (:new.GOL_EVSAHIBI > :new.GOL_KONUK) then
       update TAKIMLAR
       SET puan=puan+3
       where TKNO=(:new.TKM_ID_EV);
      END IF;

      IF( :new.GOL_EVSAHIBI < :new.GOL_KONUK)then
       update TAKIMLAR
       SET puan=puan+3
       where TKNO=(:new.TKM_ID_KONUK);

       ELSE
       update TAKIMLAR
       SET puan=puan+1
       where TKNO in (:new.TKM_ID_EV,:new.TKM_ID_KONUK);
       END IF;
```

Örnek: İlk örnek STADLAR tablosundan veri sildirmeyen Before tetikleyicisi, diğer örnek ise Cumartesi ve Pazar günleri silme, ekleme ve güncelleme işlemini engelleyen BEFORE tetikleyicisi örneğidir.

```
CREATE OR REPLACE TRIGGER yasaklama

BEFORE DELETE ON STADLAR

BEGIN

 dbms_output.put_line('Veri Silemezsiniz');
  Rollback;

END;
```

```
CREATE OR REPLACE TRIGGER YASAK_İSLEM
BEFORE INSERT OR DELETE OR UPDATE ON PERSONEL

BEGIN

  IF(to_char(sysdate,'DY')IN ('PAZ','CMT')) THEN
  dbms_output.put_line ('Hafta sonları işlem yasağı');
  rollback;
  END IF;

END;
```

İmleçler(Cursors)

İmleçler, PL/SQL'de çok kayıt içeren tablolar üzerinde faydalıdır. Amacı birçok kaydın hafızaya getirilmesi ve üzerinde işlem yapılmasıdır. Oracle bu işlem için hafıza yer açmaktadır. İki çeşit imleç mevcuttur. Bunlar **Kapalı(Implicit)** ve **Açık(Explicit)** imleç tipleridir.

Kapalı imleçler, sistem tarafından Select, Delete, Insert ve Update SQL cümleleri için açılan imleçlerdir. Açık imleç ise kullanıcı tarafından tanımlanan imleç tipidir.

Kapalı imleçler, PL/SQL bloklarında yazılan SQL cümleleri için sistem tarafından tanımlanır demiştik ve bunun özelliklerini taşımaktadır. Bu Özellikler:

Özellik	**Görevi**
SQL%ROWCOUNT	İşlem gören kayıt sayısını tutar.
SQL%ISOPEN	İmleç açıksa **true**, değilse **false** değeri tutar.
SQL%FOUND	İşlem sonucu kayıt dönerse **true**, dönmez ise **false** değerini tutar.
SQL%NOTFOUND	İşlem sonucu kayıt dönmez ise **true,** yoksa **false** değerini tutar.

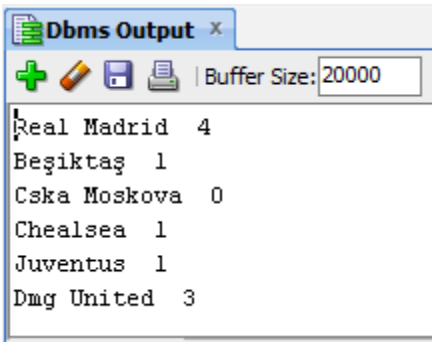
Örnek: Grup tablosundan veri silindiğinde silinen kaç kayıt olduğunu gösteren blok kodunu yazınız.

```
DECLARE
silinen number(2);
BEGIN

DELETE
FROM GRUPLAR
WHERE G_NAME in ('D','H');

silinen:=SQL%ROWCOUNT;
dbms_output.put_line (silinen||'kayıt silinmiştir');

END;
```

Bunun yanında açık imleçler, kullanıcılar tarafından özellikle güncelleme ve diğer işlemler için tanımlanırlar. Kayıtları getirmek için döngü içinde özellikle de For döngüsünü kullanmak çok faydalı olacaktır. Açık imleçlerde kontrol için üç komut özellik mevcuttur. Bunlar OPEN, FETCH ve CLOSE komutlarıdır. DECLARE kısmında tanımlanan imleç blok içerisinde OPEN komutu ile açılır. FETCH komutu ile kayıtlar tek tek çağırılıp, işlenir. CLOSE komutu ile imleç kapatılır. Bu işlemi bir örnek üzerinde görelim.

Örnek: Puanı 9'dan küçük olan takımların adlarını ve puanlarını yan yana yazan kodu yazınız.(Cursor kullanılacak.)

```
DECLARE
CURSOR cl IS

SELECT isim,puan FROM TAKIMLAR WHERE puan < 9;

BEGIN
FOR takım IN cl
LOOP
dbms_output.put_line(takım.isim ||'  '|| takım.puan);
END LOOP;

END;
```



Örnek: Puanı 9'dan büyük olan takımların isim ve puan bilgilerini oluşturulan başka bir tabloya atan kodu yazınız.(Cursor kullanılacak.)

```
DECLARE

CURSOR c1 IS
SELECT isim,puan FROM TAKIMLAR WHERE puan > 9;
takimlist c1%rowtype;

BEGIN
OPEN c1;
LOOP
FETCH c1 into takımlist;
EXIT WHEN c1%NOTFOUND;

INSERT INTO TurAtlama values(takimlist.isim,takimlist.puan);

dbms_output.put_line('İŞLEM BAŞARILI ');
COMMIT;
END LOOP;

END;
```

Paketler(Packages)

Paketler, mantıksal olarak ilişkili PL/SQL tipleri, alt programları gruplayan bir şema nesnesidir. Çoğu zaman sıklıkla ya da belirli bir işlem için kullanılan birden fazla veri tabanı işlemimiz var ise (procedure, function vb.) bu işlemlerin hem kullanımı hem de kolay takibi için paket yapısı kullanılır. Bize avantajları modülerlik, kolay uygulama dizaynı, bilgi saklama, iyi performans ve fonksiyonellik ekleme olarak sıralanabilir. Paketler genellikle iki kısımdan oluşur. Bunlar gövde(body) ve paket spec(specification) ifadeleridir. Spec, uygulamalarımız için ara yüzdür ve tipler, değişkenler, sabitler, aykırı durumlar, imleçler ve alt programlar için kullanılabilir. Gövde bölümü ise imleç ve alt programların tanımlandığı bölümdür.

Sistemde kayıtlı paketler bulunmaktadır. Bunlar:

- **DBMS_OUTPUT** – *en çok bilinen, verileri ekranda göstermek için gerekli paket.*
- **DBMS_PIPE – PIPE ile haberleşmeyi sağlar**
- **DBMS_LOCK** – Kullanıcı kilitlerini yönetir.
- **DBMS_ALERT**—*Spesifik veri tabanı değerleri değiştiğinde veri tabanı tetikleyicileri için alert kullanabilirsiniz..*
- **UTL_FILE** – *PL/SQL kodlarını İşletim sistemi dosyalarına yazmak ve okumak için.*
- **UTL_HTTP** – *PL/SQL kodlarıyla http protokolü belirtme çizgileri oluşturmak.*

Paket yapısının söz dizimi aşağıdaki gibidir.

CREATE [OR REPLACE] PACKAGE *isim*
  [AUTHID {CURRENT_USER | DEFINER}]
  {IS | AS}
  [PRAGMA SERIALLY_REUSABLE;]
  [collection_type_tanımlama ...]
  [record_type_ tanımlama ...]
  [subtype_ tanımlama ...]
  [collection_ tanımlama ...]
  [constant_ tanımlama...]
  [exception_ tanımlama ...]
  [object_ tanımlama ...]
  [record_ tanımlama ...]
  [variable_ tanımlama ...]
  [cursor_spec ...]
  [function_spec ...]
  [procedure_spec ...]
  [call_spec ...]
  [PRAGMA RESTRICT_REFERENCES(assertions) ...]
END [*paket_ismi*];

[CREATE [OR REPLACE] PACKAGE BODY paket_ismi **{IS | AS}**
  [PRAGMA SERIALLY_REUSABLE;]
  [collection_type_tanımlama ...]
  [record_type_tanımlama ...]
  [subtype_ tanımlama ...]
  [collection_ tanımlama ...]
  [constant_ tanımlama ...]
  [exception_ tanımlama ...]
  [object_ tanımlama ...]
  [record_ tanımlama ...]
  [variable_ tanımlama ...]
  [cursor_body ...]
  [function_spec ...]
  [procedure_spec ...]
  [call_spec ...]
[BEGIN
  --*Cümlelerin sırası*
END [*paket_ismi*];]

- **Örnek:** İçerisinde iki değer ve işlem girilip toplama ve çıkarma işlemi yapılan paket kodunu yazınız.

  / Paket Spec tanımlanıyor. */*

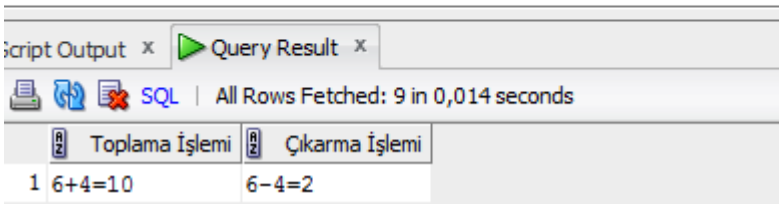
```
CREATE OR REPLACE PACKAGE dort_islem IS

FUNCTION hesapla(isleml NUMBER, islem2 NUMBER, islem3 VARCHAR2) RETURN VARCHAR2;

END dort_islem;
```

/ Paket gövdesi tanımlanıyor */*

```
CREATE OR REPLACE PACKAGE BODY dort_islem IS

FUNCTION hesapla(isleml NUMBER, islem2 NUMBER, islem3 VARCHAR2)
RETURN VARCHAR2 IS
RESULT VARCHAR2(500);
BEGIN
IF islem3 = '+' THEN
RESULT := to_char(isleml) || '+' || to_char(islem2) || '=' || (isleml + islem2);
ELSIF islem3 = '-' THEN
RESULT := to_char(isleml) || '-' || to_char(islem2) || '=' || (isleml - islem2);
END IF;

RETURN(RESULT);
END;

BEGIN
  NULL;
END dort_islem;
```

/ Sonuç olarak listelenmesi */*

```
SELECT dort_islem.hesapla(6, 4, '+') "Toplama İşlemi",
       dort_islem.hesapla(6, 4, '-') "Çıkarma İşlemi"
  FROM TAKIMLAR;
```



Kullanıcılar ve Roller(Users and Roles)

Oracle'da veri tabanına erişecek, tablolar üzerinde işlem yapabilecek, çeşitli rollerde kullanıcılar yaratılabilir. Bölümün başında Oracle 10g veri tabanı erişim web konsolundan DMG adlı bir kullanıcı oluşturup ona roller vermiştik. Fakat veri tabanını kurarken ilk giriş kullanıcısı olarak SYS ya da SYSTEM kullanıcısına bir şifre belirlemiştik. İşte burada bu iki kullanıcı, sistem yöneticisi olarak default gelmektedir. Bunun akabinde kullanıcılar

oluşturmuştuk. Kullanıcı oluşturma ve rol atama işlemlerini PL/SQL komutları ile de yapabiliriz. Bu konuyu adım adım kodlar ile göstermek istiyorum.

/ deniz adında şifresi a1234_ olan bir kullanıcı oluşturdum */*

```
create user deniz
identified by a1234_;
```

/ deniz kullanıcısının şifresini b1234_ olarak değiştirdim */*

```
alter user deniz
identified by b1234_;
```

/ Operate isminde bir rol oluşturdum*/*

/ Bu role personel tablosu üzerinde select ve insert yetkisi verdim */*

/ deniz kullanıcısına operate rolü verdim */*

```
create role operate;

grant select,insert on personel to operate;

grant operate to deniz;
```

/ operate rolüne TAKIMLAR tablosu üzerinde tüm yetkiler verilmiştir.*/*

/ deniz kullanıcısından personel tablosu üzerinden select izni alınmıştır. */*

/ operate rolü silinmiştir. */*

```
grant all on TAKIMLAR to operate;

revoke select on personel from deniz;

drop role operate;
```

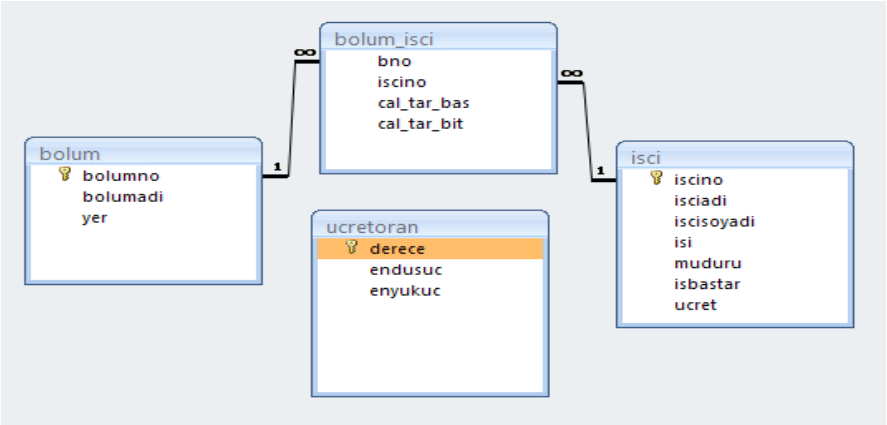
/ deniz kullanıcısına oturum açma ve veri tabanı yöneticisi rolü atanmıştır. */*

/ deniz kullanıcısı silinmiştir. */*

```
grant create session, dba
to deniz;

drop user deniz;
```

Not: Bu konuda tablo üzerindeki işlem yetkileri **Select, Insert, Update, Delete, References, Alter, Index**'ten oluşmaktadır. Veri tabanı üzerindeki yetkiler ve rollerden bazıları ise **Create Table, Create Cluster, Create Synonym, Create View, Create Sequence, Create Database Link, Create Session** ve önemli rol olan **DBA(Database Administrator)** rolüdür.

Bölüm Sonu Soruları



(3-10 arasındaki sorular yukarıdaki diyagrama göre cevaplanacaktır.)

1.) Oracle Veri Tabanı Yapısını açıklayınız.

2.) Veri Tabanı Yöneticisinin Görevlerini açıklayınız.

3.) PL/SQL bloklarını ve Döngü yapılarını açıklayınız.

4.)İşçi tablosunda kullanılmak üzere 1 ile 50000 arasında bir sequence oluşturunuz. İşçi tablosuna dışarıdan değer alacak şekilde veri giren PL/SQL kodunu yazınız.

5.) İşçi tablosundaki işçilerin ücretlerini ucretoran tablosundaki en düşük ve en yüksek ücret alanları ile karşılaştırılıp, bu işçilerin kıdemlerini bulan PL/SQL kodunu yazınız.

6.) İşçi numarası dışarıdan girilen işçinin bu zamana kadar ne kadar ücret aldığını ekrana yazan PL/SQL kodunu yazınız.

7.) Adı soyadı girilen işçi sistem de var ise maaşının, maaşların ortalamasından küçük olması durumunda, maaşını ortalama maaş şeklinde güncelleyiniz. Hata mesajlarını kodlayınız.

8.) Mertkan adlı bir kullanıcı oluşturup(şifre size ait), Bu kullanıcıya veritabanına bağlanma, oturum açma, tablo oluşturma hakkı veren kodu yazınız.

9.) En küçük ilk 5 işçi için bir cursor (imleç) tanımlayınız. Bu işçilerin adlarını tümü büyük harf olmak üzere listeleyen PL/SQL kodunu yazınız.

10.) Bilgi İşlem Bölümünde çalışan işçilerin (şuan) adının ilk hafi büyük, soyadı Büyük harfle ulanmış halde listeleyiniz. Böyle bir bölümde çalışan yoksa ona uygun sistem exception'ı yazınız.

Kaynakça

Kitap ve e-kitaplar

Oracle 10g, Murat OBALI, Pusula Yayıncılık,2007,İstanbul

Oracle 9i, Osman Murat ŞEN, Beta Yayıncılık,2004,İstanbul

PL/SQL User's Guide and Reference 10g Release 1 (10.1) , Part No. B10807-01, December 2003

Oracle 10g PL/SQL Programming, Scott Urman, Ron Hardman, Michael McLaughlin, Oracle Press,2004

Oracle® PL/SQL by Example, Benjamin Rosenzweig, Elena Silvestrova Rakhimov, Addison Wesley,2008

Web Adresleri

http://tonguc.oracleturk.org/index.php/2006/03/21/kahin-tarihihistory-of-oracle/

http://www.teknolojioku.com/forum/Konu-Oracle-Veritaban%C4%B1-kurulumu-Resimli-1529.html

http://www.javaturk.org/?tag=oracle-veri-tabani

http://docs.oracle.com/cd/B19306_01/license.102/b14199/editions.htm

http://www.oracle.com/us/products/database/enterprise-edition/comparisons/index.html

http://docs.oracle.com/cd/B25329_01/doc/admin.102/b25107/getstart.htm

Learning Oracle PL/SQL ,Bill Pribyl&Steven Feuerstein

Oracle PL/SQL Programming Steven Feuerstein with Bill Pribly

http://www.ceturk.com/oracle-veritabanina-kusbakisi/

http://www.ceturk.com/plsql-blok-yapilariveri-tipleri/

http://www.java2s.com/Tutorial/Oracle/0420__PL-SQL-Data-Types/BlockStructure.htm

http://datawarehouse.gen.tr/Makale.aspx?ID=443&seo=plsql-yazi-dizisi2

http://www.baskent.edu.tr/~tkaracay/etudio/ders/dbase/sql/pdfSQL/Introduction.pdf

http://muratimre.blogspot.com/2012/06/oracle-pl-sql-de-donguler-ve-kullanm.html

http://ftmnrtrkk.wordpress.com/2012/10/05/oracle-veri-tipleri/

http://www.bilgisayardershanesi.com/Y5330-oracle-view-yaratmak-ve-degistirmek--performans.html

http://sametcelikbicak.wordpress.com/2010/06/08/plsql-package-ornek/

http://www.techonthenet.com/oracle/roles.php

http://anovian.wordpress.com/2008/05/21/oracle-memory-structure/