

Veri Kontrol Dili (DCL-Data Control Language)

Veritabanı üzerinde verilere erişimi kontrol etmek amacıyla kullandığımız komutlardır. Bir kullanıcıya verilen izinler ile alakalı komutlardır.

Komut	İşlevi
GRANT	Kullanıcıya veritabanı erişim yetkisi verir.
REVOKE	GRANT ile verilen yetkiyi geri alır.

ORACLE' da Bütünlük Kısıtları

PL/SQL diline geçmeden önce bütünlük kısıtlarını vermek önem arz etmektedir. Veri tabanı içerisinde verilerin tutulacağı tabloları oluştururken ya da sonradan tablolar üzerinde yanlış ve istenmeyen veri girişlerini engellemek açısından bütünlük kısıtları bulunmaktadır. Bunlar MS SQL Server'da ya da diğer VYTS'lerde de bulunmaktadır. Oracle'da 5 çeşit bütünlük kısıtı bulunmaktadır. Bunlar:

- **Primary Key**
 - Birincil Anahtar atamak için kullanılan kısıttır. Kısıtın verildiği kolona artık benzersiz ve tekrarsız verilerin girilmesi gerekmektedir. Bu alan NULL girişe izin vermez.
- **Foreign Key**
 - İlişkili tablolarda ana tabloda primary key ya da unique key alanını referans tutan alan için foreign key kısıtlaması verilir. Böylece iki tablo arasında ilişki kurulmuş olur.
- **Unique Key**
 - İlgili alana tekrarsız kayıtların girilmesini sağlar. Primary Key kısıtlamasına benzer fakat NULL girişe izin verir.
- **Check**

- İstenilen alana giriş şartı verilebilir. Böylece şarta uymayan verilerin girişi engellenir. Öğrenci Not alanına 0 ile 100 arası değer vermek buna güzel bir örnektir.

- **Not Null**

- Null değerlerin girişini iptal eder. Böylece o alana zorunlu olarak değer girilmesi gerekmektedir. Varsayılan değer bir alan için NULL'dur.

Not: DEFAULT ifadesi Oracle'da kullanılmaktadır. Tanımlama bloğunda bir değişkene ilk değeri atayabiliriz ya da bunun yerine default kullanabiliriz. Örnek vermek gerekirse;

Declare

```
sayi number(5):=0;
/* Bunun yerine */-- Açıklama Satırı
sayi number(5) DEFAULT 0;
```

Örnek: Tüm kısıtlamaları içeren bir öğrenci bilgilerini tutan tablo oluşturunuz.

```
create table ogrenci
(
  Ogr_Id number(7),
  Ad varchar2(30) not null,
  Soyad varchar2(30) not null,
  TcNo char(11),
  Adres varchar2(50),
  BolumNo number(2),
  dogtar date,
  Constraint dmgl primary key(Ogr_Id),
  Constraint dmgl2 foreign key(BolumNo) references bol
  Constraint dmgl3 check(extract(year from dogtar)>199
  Constraint dmgl4 unique key(TcNo)
);

create table bolum
( bno number(2),
  ad varchar2(30),
  Constraint dmgl primary key(bno)
);
```

Bunun yanında bu kısıtları kaldırabilir ya da sonradan tabloya ekleyebiliriz. Bunun için DROP ve ALTER komutlarını kullanmamız gerekir. Örnek vermek gerekirse;

Kısıtlamaların Sonradan Eklenmesi	Kısıtlamaların Kaldırılması
Alter Table ogrenci Add Constraint dmgl Primary Key (Ogr_Id);	Alter Table ogrenci Drop Constraint dmgl3;

PL/SQL

PL/SQL(Procedural Language Extension of SQL) , Oracle şirketi tarafından 1990'ların başında SQL ile yapacağımız sınırlı işlemleri geliştirmek için ortaya çıkmıştır.

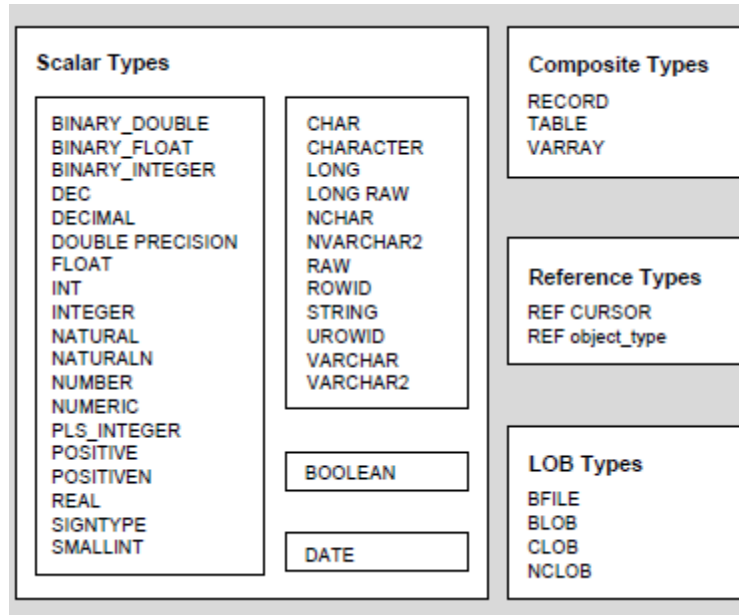
SQL komutlarını çalıştırabildiği gibi bunun yanında Oracle veri tabanı sistemleri için tetikleyiciler, yordamlar ve fonksiyonlar yazmak üzere geliştirilmiş, programlamada akış kontrollerini ve değişkenleri kullanmamıza imkân sağlayan bir dildir. Model olarak ADA dili örnek alınmıştır. Bu yüzden söz dizimlerinde ve işlemlerde PASCAL dilinin yapısına aşina olan kullanıcılar, PL/SQL üzerinde zorluk çekmeyecektir. Şu an dünyada da yoğun olarak kullanılmaktadır.

PL/SQL'in Özelliklerini maddelersek;

- 1-PL/SQL yordamsal bir yapıya sahiptir.(if/else, goto, Loop gibi yapılar içerir).
- 2-Tek seferde istediğiniz kadar işlemi veri tabanına göndereceğinizden her seferinde SQL sorgular çalıştırmaktan daha hızlı toplam sonucu alabilir.
- 3- Çalıştırmış olduğunuz SQL kodlarını, debug etme, loglama gibi işlemleri PL/SQL kodlarıyla ya da yapılarla sağlayabilir.
- 4-Oracle Forms, Oracle Report gibi yararlı araçlarla bütünleşebilir.
- 5-Hata İşleme(Exception handling) ile bu durumlarda farklı işlemler tanımlayabilir ya da loglama yapabilir.
- 6- Birçok Editör ile PL/SQL kodları yazılabilir.
- 7- Fonksiyon, yordam, trigger tanımlamaları gibi birçok özelliğiyle uygulamalarda büyük artı sağlar.
- 8- Ağ trafiği konusunda uzmandır. PL/SQL kodları veritabanında saklandığı için sorgular için istemciden tekrar tekrar veri tabanına gönderilmediği için ağ trafiği oluşmaz.
- 9- Oracle birçok platformda çalışabilir.(Unix, Windows, Linux gibi.)
- 10- SQL dilinin tüm komutları ve veri tipleri desteklenmektedir.

PL/SQL'de Veri Tipleri

Her programlama dilinde olduğu gibi Oracle veri tabanında da veriler için tablo oluşturduğumuzda, tablonun her bir sütunu için bir veri tipi tanımlanır. Oracle'da birçok veri tipine sahiptir.



PL/SQL Veri Tipleri

PL/SQL Sayısal Veri Tipleri

Karakter Veri Tipleri

VARCHAR2: 4000 taneye kadar karakter tutabilen veri tipidir. Eğer saklanacak verinin başında ya da sonunda boşluk karakteri varsa bunları silerek tutar. Böylece boşluk için fazladan yer tutmamış olur.

NVARCHAR2: Değişken uzunlukta 4000 taneye kadar Unicode karakter tutabilen veri tipidir. Eğer saklanacak verinin başında ya da sonunda boşluk karakteri varsa bunları silerek tutar.

CHAR: Sabit uzunluklu 2000 taneye kadar karakter/byte tutabilen veri tipidir. Eğer sayı ile ifade edilen karakterden az girilirse, oracle boşluk karakteri ekleyerek sabit uzunluğa kadar getirir. Char(20) ile Char(20 Byte) aynıdır. Sabit uzunluk için bir değer girilmediğinde default 1 kabul edilir.

NCHAR: Sabit uzunlukta 2000 taneye kadar Unicode karakter tutabilen veri tipidir. Unicode karakterler dil değişse bile aynı şekilde karakterin kaydedilmesini sağlar. Bir standartlaşma yapmış olur.

Sayı Veri Tipleri

NUMBER: Number veri tipiyle fixed-point ve floating point sayıları saklanabilir. Bununla birlikte integer, float, decimal gibi sayılarda number veri tipiyle saklanabilir. Tam kısım 38 basamak , ondalık kısmın basamak sayısı da -130 ile 125 arasındadır.

DECLARE

```
dmg_integer NUMBER(6);  
dmg_scale_3 NUMBER(6,3);  
dmg_real NUMBER;  
BEGIN  
END;
```

FLOAT: Number veri tipinin alt tipi olarak düşünebilir. Oracle 11g versiyonu, Float yerine Binary_Float ve Binary_Double veri tipinin kullanılmasını önermektedir.

BINARY FLOAT: 5 byte yer tutar ve 32-bit floating point sayıları saklamak için kullanılır.

BINARY DOUBLE: 9 byte yer kaplar ve 64-bit floating-point sayıların saklanması için kullanılabilir.

Boolean Veri Tipleri

Boolean, mantıksal veri tipi **TRUE**, **FALSE** ve **NULL** olarak üç adet değer alır. Direkt tanımlama bloğunda atama yapılabilir. 'TRUE' gibi bir ifade hataya sebep olmaktadır.

DECLARE

```
dmg_boolean BOOLEAN; -- Bu kısımda ya da BEGIN-END arasında atama yapılabilir.  
BEGIN  
dmg_boolean := 'TRUE'; -- hataya sebep olur.  
END;
```

Tarih ve Zaman Data Tipleri

Tarih ve Zaman veri tipi DATE, TIMESTAMP ve INTERVAL tipleri içerir.

DATE: Date veri tipi saniye, dakika, saat, gün, ay, yıl ve yüzyıl bilgisini tutar.

TIMESTAMP: Date ve timestamp veri tipi arasındaki tek fark timestamp de saniye kısmı kesirli ifade içerir. Default da 6 hanesi vardır, fakat 0-9 arasında değer verebilir.

TIMESTAMP WITH TIME ZONE: Timestamp'den tek farkı time zone bilgisini de saklar.

TIMESTAMP WITH LOCAL TIME ZONE: Saat, tarih ve Time zone bilgisini veritabanından değil programın çalıştığı istemciden alır.

INTERVAL YEAR TO MONTH: Bu veri tipi, yıl ve ay cinsinden aralık belirler.

INTERVAL DAY TO SECOND: Gün, saat, dakika ve saniye cinsinden zaman aralığını saklayan veri tipidir.

LONG ve RAW Veri Tipleri

LONG: Long veri tipi 2 GB'a kadar değişken uzunlukta karakter bilgisi tutan bir tiptir. Kullanımı Oracle Database 11g tarafından önerilmemektedir. Bunun yerine CLOB veri tipini kullanabiliriz.

RAW(sayı): Binary bilgileri saklamak için kullanılır. Maksimum boyutu 2000 byte'tır.

LONG RAW: RAW ile aynı sayılır fakat maksimum boyut belirtilmektedir. Bir tabloda Long ve Long RAW veri tipinden sadece 1 tane kolon tanımlayabiliriz. Bu yüzden Oracle LOB veri tiplerini kullanmayı önermektedir.

Büyük Nesne Veri Tipleri

BLOB: Maksimum 4 GB boyutunda herhangi bir yapısı olmayan verileri saklamak için kullanılabilir. Bunlar yazı, resim, video ve koordinat verileri tutulabilir.

BFILE: Bu veri tipi işletim sisteminde saklamak ve işletim sistemindeki dosyalara veritabanından erişmek için kullanılır.

CLOB (CHARACTER LARGE OBJECT): 4 GB'a kadar değişken uzunlukta veriler tutabilmektedir.

BLOB (BINARY LARGE OBJECT): CLOB' benzer bir olmakla birlikte içinde unicode veriler tutan maksimum büyüklüğü 4 GB olan bir veri tipidir.

ROWID Veri Tipleri

ROWID: Satırların fiziksel adreslerini saklar. Satırların, tablolarda birincil anahtarlarına göre mantıksal olarak yerlerini tutar. Bu değer SQL cümlesi içerisinde diğer kolonlarla birlikte ROWID yazılarak öğrenilir.

UROWID: İndex tabanlı ve yabancı tabloların adreslerini tutar.

ANSI, DB/2,SQL/DS Veri Tipleri

Oracle’da tablolarda listelenen ANSI veri tipleri de kullanılabilir.Oracle’da tablolarda listelenen ANSI veri tipleri de kullanılabilir.

SQL/DS veya DB2 Veri Tipi	Oracle Veri Tipi
Character(n)	Char(n)
Varchar(n)	Varchar(n)
Long Varchar(n)	Long
Decimal(p,s)	Number(p,s)
Integer, Smallint	Number(38)
Float(b)	Number

ANSI SQL Veri Tipi	Oracle Veri Tipi
Character(n), Char(n)	Char(n)
Character Varying(n), Char Varying(n)	Varchar(n)
National Character(n), National Char(n), Nchar(n)	Nchar(n)
National Character Varying(n), National Char Varying(n), Nchar Varying(n)	Nvarchar2(n)
Numeric(p,s), Decimal(p,s)	Number(p,s)
Integer, Smallint, Int	Number(38)
Float(b), Real, Double	Number

PL/SQL’de Operatörler

Diğer programlama dillerinde olduğu gibi özellikle akış kontrollerinde kullanılan birden çok operatör PL/SQL’de de bulunmaktadır. Kitabımızda da bunların bir listesi verilecektir. Bu liste yazacağınız kodlar için bir referans niteliği taşıyacaktır.

Operatör	İşlevi	Operatör	İşlevi
+	Toplama	**	Üs Alma
-	Çıkarma	=	Eşitlik
/	Bölme	>	Büyüklik

*	Çarpma	<	Küçüklük
< >	Eşit Değil	>=	Büyük Eşit
!=	Eşit Değil	<=	Küçük Eşit
^=	Eşit Değil		Birleştirme
..	Aralık	:=	Atama

PL/SQL’de Atama İşlemleri

PL/SQL dilinde bilinen iki şekilde atama vardır. Bunlardan ilki “:=” operatörü kullanarak diğeri ise INTO deyimi ile yapılmaktadır. Blok içerisinde SELECT ile birlikte INTO kullanılarak, SELECT sorgusundan dönen sonucu aynı tipteki diğeri değişkene atayabiliriz. Örnekle göstermek istersek, bir anonim blok içerisinde tanımlama ve atama işlemlerini açıklayalım.

```
DECLARE
ucret int;
```

```
BEGIN
```

```
  select maas into ucret
  from Personel
 where ad='deniz';
```

```
WHILE ucret<2500 LOOP
```

```
  update Personel
  set maas=ucret*1.1
  where ad='deniz';

  ucret:=ucret*1.1;
```

```
END LOOP;
```

Integer tipinde **ucret** isimli bir değişken tanımlanmaktadır.

INTO deyimi ile SQL sorgusundan dönen **maas** verisi **ucret** isimli bir değişkene atanmaktadır.

ucret isimli değişkenin içine “:=” ile yeni bir değer atanmaktadır.

PL/SQL’DE Değişkenler Ve Sabitler

PL/SQL’de diğeri programlama dillerinde olduğu gibi değişkenler, sabitler, semboller ve bunların yazım biçimleri bulunmaktadır. Fakat ilk olarak şu bilinmelidir ki PL/SQL’de küçük büyük harf ayrımı yoktur. Sadece bu ayrım karakterler için geçerlidir. Örnek vermek gerekirse;

- Takım_Id number(4) ile takım_id number(4) tanımlamaları aynıdır.
- “Deniz Mertkan GEZGİN” ile “deniz mertkan gezgin” ifadeleri aynı değildir.

PL/SQL’de kod yazılırken farklı veri tiplerinde değişkenler tanımlanabilir. Değişkenler tanımlanırken değer atanabilir ya da sonradan program akışı içerisinde de değer atanabilir. Tanımlama ve atama işlemine üç şekilde örnek vermek gerekirse;

Tanımlama İşlemi	Tanımlama ve İlk Değer Atama İşlemi	Tanımlama ve İlk Değer Atama İşlemi (Not Null ifadesi Seçimlidir.)
DECLARE ucret int ;	DECLARE ucret int :=500;	DECLARE ucret int not null :=500;

PL/SQL’de değişken ya da sabit tanımlarken kullanabileceğimiz karakterler, sayılar ve semboller listelenmektedir.

- Sayılar 0..9
- Harfler a..z ve A..Z
- Boşluk, Sekme ve Enter karakterleri
- Semboller + - * / < > = ! ^ ; : @ % , “ \$ # _ { [] } ‘ () ? & | ~

PL/SQL’de sabit tanımlarken kullanacağımız deyim ise CONSTAT deyimidir.

```
DECLARE
ucret constant int:=5000;
```

Kodu ile **ucret** değeri 5000 değeri ile sabitlenmiştir.

PL/SQL’de Blok Yapısı

PL/SQL’de bloklar birbiriyle ilgili kodların, programların birlikte yorumlandığı yapılardır. Blok yapıları üç tip olarak karşımıza çıkmaktadır. Anonim, Yordam ve Fonksiyon blokları olan bu bloklar PL/SQL programında kullanılabilir. En Basit PL/SQL blok yapısı olan anonim blok yapısının söz dizimi aşağıdaki gibidir.

[DECLARE] -Tanımlama alanıdır.

-Değişkenler, kursörler, tanımlı hata bildirimleri vs.

BEGIN

-SQL Cümleleri, IF-Else yapıları

-PL/SQL Cümleleri

[EXCEPTION]

-Hata oluştuğunda yapılacak işlemler ya da mesajlar

END;

Anonim Bloklar;

- Anonim bloklar bir isme sahip değildir. Bloklara isim vermek zorunlu değildir.
- Çalıştırılması için tüm blok çağrılmalıdır.
- Veritabanında saklanmazlar.
- Bir defaya mahsus kullanımda genelde tercih edilirler.
- Eğer hazırlanan bir blok yeniden kullanılmak istiyorsa, sql uzantılı dosyalara saklanıp yeniden kullanılabilir.

Diğer blok yapıları ise Fonksiyon ve Prosedür blok yapılarıdır.

Yordamlar için blok yapısı;

PROCEDURE- *adı*

IS

BEGIN

komutlar

[EXCEPTION] – *Aykırı durumlar*

END;

Yordam Bloklar;

- Prosedürlerin bir ismi vardır.
- Veritabanında, tablo ya da index gibi bir şema altında depolanıp saklanabilirler.
- Diğer programlar ya da scriptler içinde kullanılabilirler.
- Ayrıca isimleri ile de çağrılıp işlevini yerine getirebilirler.
- Birden çok değer getirebilirler.

Fonksiyonlar için blok yapısı;

FUNCTION- *adı*

RETURN [*veri türü*]

IS

BEGIN

komutlar

RETURN [*değer*];

[EXCEPTION]

END;

- Fonksiyonlar da prosedürler gibi, veritabanında saklanıp, çeşitli yerlerde çağrılabilirler.
- Ancak fonksiyonlar tek bir değer döndürürler ve bir veriye return edilmeleri gerekir.
- Bu blokların içinde de bloklar olabilir, nested bloklar denmektedir.

PL/SQL'de Sıra Yapısı (Sequence)

Sıralar(Sequence), PL/SQL'de otomatik şekilde istenilen alana sırayla tanımlandığı gibi ve ölçüde değer atamak için tanımlanırlar. İki parametresi mevcuttur. Bunlar **NextVal** ve **CurrVal** parametreleridir. NextVal parametresi bir sonraki sayıyı üretmek için, CurrVal ise en son verilen sayıyı göstermek için kullanılır. Sıra yapısının söz dizimini aşağıdaki gibidir.

CREATE SEQUENCE *dizi_adı*

INCREMENT BY *tamsayı* – *Artış miktarını belirler*

START WITH *tamsayı* – *Başlangıç değeri belirler*

MAXVALUE *tamsayı* – *Maksimum değeri belirler*

CYCLE|NOCYCLE – *Sırada maksimuma gelinince başa dönülüp, dönmeyeceğini belirler.*

CACHE *tamsayı* |**NOCACHE**;-- *sıra numaralarının hafıza da ne kadar saklanacağını belirler.*

Örnek: OtomatikId isminde bir sequence oluşturup. Personel tablosuna veri girerken kullanımı gösteriniz.

```
Create Sequence OtomatikId      INSERT INTO PERSONEL VALUES (OtomatikId.nextval, 'Yusuf','Gezgin', 1000);
increment by 1
start with 1                      select OtomatikId.Currval
maxvalue 10000                   from personel;
nocycle
nocache;
```

PL/SQL'de Hata İşleme(Error Handling)

PL/SQL’de, ikaz ve hata şartları EXCEPTION(istisnai durum) olarak adlandırılır. İstisnalar sistem tarafından çalışma zamanında otomatik ya da kullanıcı tanımlıdır. Sistemde bulunan istisnai durumlara örnek olarak *division by zero* (sıfıra bölünme durumu) ve *out of memory* (hafıza taşması durumu) durumlarını verebiliriz. Kullanıcılarda PL/SQL’in tanımlama bloklarında bir istisnai durum oluşturabilirler.

Hata işlemede, bloklar içindeki kodlarda bir hata oluştuğunda, bir istisnai durum tanımlandıysa kod oraya yönlenererek gerekli hata kodunu ve ona karşılık gelen mesajı verir. Hata işlemenin, çalışma prensibi bu şekildedir. Hata işlemenin faydalarına geldiğimizde konu devamında yazılacak iki sözdizimini inceleyebilirsiniz.

Bu söz diziminde her SELECT ifadesinde ya da içinde hata kontrolü yapmalı kodu check etmeliyiz. Buda programın yavaşlamasına ve ekstra ifadeye ihtiyaç duymaktadır.

BEGIN

SELECT...

-- *'no data found' hatası için kontrol etmeli.*

SELECT...

-- *'no data found' hatası için kontrol etmeli.*

SELECT...

-- *'no data found' hatası için kontrol etmeli.*

Fakat bu söz diziminde tüm kodlar işlenirken bir hata oluştuğunda direk EXCEPTION bölümü devreye girerek hata mesajını ya da kodunu döndürecektir.

BEGIN

SELECT...

SELECT...

SELECT...

...

EXCEPTION

WHEN NO_DATA_FOUND THEN -- *Tüm 'no data found' hatalarını yakalar.*

Örnek: Takımın ismi girildiğinde bu takımın puanın tüm takımların puanına oranı nedir sorusuna cevap veren bir fonksiyon tanımlayalım.(Örnek **division by zero** durumunu göstermek için hayali bir örnektir.)

```
create or replace function Ornek1(ad varchar2)
return number
is
yuzde NUMBER(3,1);
BEGIN
declare
deger number(2);
BEGIN

    select puan into deger
    FROM TAKIMLAR
    WHERE isim=ad;

    SELECT max(puan)/deger INTO yuzde
    from TAKIMLAR;

    return yuzde;

EXCEPTION
    WHEN ZERO_DIVIDE THEN
        dbms_output.put_line ('Bu takım 0 çekmiştir');
END;
END;
```

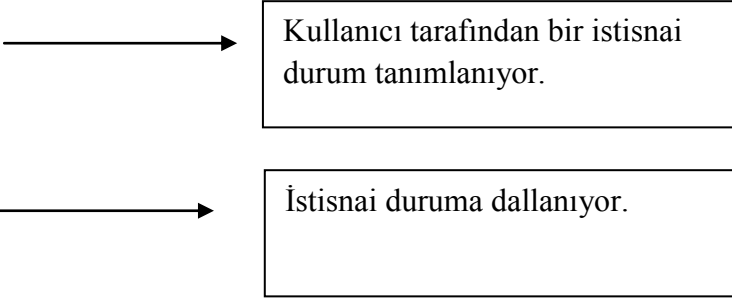
Örnek: Kullanıcı tanımlı istisnai durumları anlayabilmek için hazırlanmış bir örnektir. Bu örnek kullanım için tam isabetli bir seçim olmasa da yapının nasıl tanımlanacağını ve nasıl kullanılacağına güzel bir örnektir. Sayı değerimiz 5, bunun 10'dan küçük olma şartına göre tanımlama kısmında oluşturduğumuz dmğ_hata isimli istisnai duruma dallanma sağlıyoruz.

```
DECLARE
    dmğ_hata EXCEPTION;
    sayı NUMBER:=5;
BEGIN

    IF sayı<10 THEN
        RAISE dmğ_hata;
    END IF;
EXCEPTION

    WHEN dmğ_hata THEN
        dbms_output.put_line ('Gruptan Çıkamaz');

END;
```



Kullanıcı tarafından bir istisnai durum tanımlanıyor.

İstisnai duruma dallanıyor.

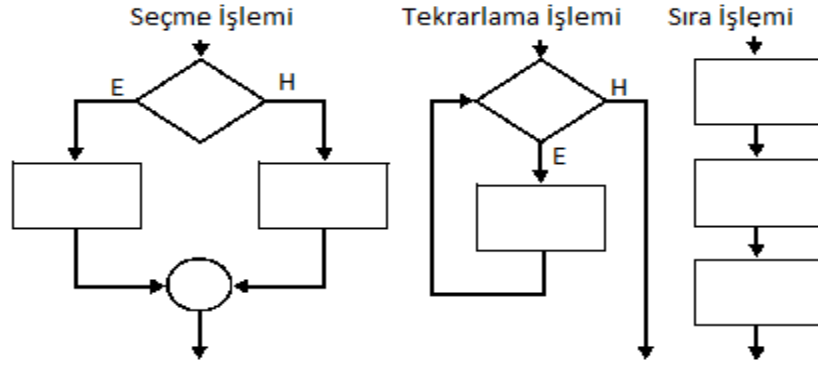
İstisnai Durum	Oracle Hata	SQL Hata Kodu
ACCESS_INT0_NULL	ORA-06530	-6530
CASE_NOT_FOUND	ORA-06592	-6592
COLLECTION_IS_NULL	ORA-06531	-6531
CURSOR_ALREADY_OPEN	ORA-06511	-6511
DUP_VAL_ON_INDEX	ORA-00001	-1
INVALID_CURSOR	ORA-01001	-1001
INVALID_NUMBER	ORA-01722	-1722
LOGIN_DENIED	ORA-01017	-1017
NO_DATA_FOUND	ORA-01403	+100
NOT_LOGGED_ON	ORA-01012	-1012
PROGRAM_ERROR	ORA-06501	-6501
ROWTYPE_MISMATCH	ORA-06504	-6504
SELF_IS_NULL	ORA-30625	-30625
STORAGE_ERROR	ORA-06500	-6500
SUBSCRIPT_BEYOND_COUNT	ORA-06533	-6533
SUBSCRIPT_OUTSIDE_LIMIT	ORA-06532	-6532
SYS_INVALID_ROWID	ORA-01410	-1410
TIMEOUT_ON_RESOURCE	ORA-00051	-51
TOO_MANY_ROWS	ORA-01422	-1422
VALUE_ERROR	ORA-06502	-6502
ZERO_DIVIDE	ORA-01476	-1476

PL/SQL Tanımlı İstisnai Durumlar

İstisnai durumların açıklamaları ve anlamları için Oracle firmasının kendi sitesine bakabilirsiniz. http://docs.oracle.com/cd/B10501_01/appdev.920/a96624/07_errs.htm

PL/SQL'de Akış Kontrol Yapıları

PL/SQL blokları içerisinde koşullu-koşulsuz dallanmalar ve döngüler kullanılabilir. PL/SQL'de üç tip kontrol yapısı vardır. Birincisi "IF" ve türevleri kontrol yapıları koşul için kullanılır. Yineleme, tekrarlama için "LOOP, WHILE, FOR" gibi kontrol yapıları ve son olarak GOTO ve NULL gibi basit dallanmalar yapan yapılar kullanılır. Şimdi şart yapılarından başlayalım.



Akış Kontrol Yapıları Çeşitleri

IF –THEN Yapısı

IF cümleleri, şartlara göre hangi komut ya da SQL cümlelerinin işletileceğini belirtilen kodlardır, ifadelerdir. En basit yapılardandır ve her programa dilinde bulunmaktadır. Bu yüzden programcıların ya da öğrencilerin aşına olduğu bir yapıdır. Zaten şunu bilmeliyiz ki tüm programlama dilleri ya da SQL vb. dillerin mantığı ve kodları birbirine benzemektedir. Sadece syntax dediğimiz söz dizimi farklı olmaktadır.

PL/SQL içerisinde üç tip "IF" yapısı kullanılır:

- IF - THEN yapısı, IF içinde belirtilen koşul sağlanırsa THEN – END IF arasındaki işlemler yapılmaktadır. Aksi takdirde işlemler yapılmadan bir sonraki işleme ya da komuta geçilmektedir.

IF şart THEN

Komutlar - SQL cümlelerini ifade eder.

END IF; - Yapının bittiğini gösterir.

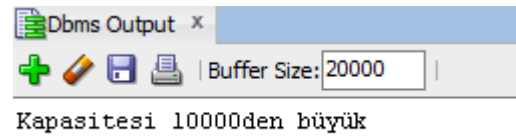
Bu yapı ile ilgili hemen bir örnek yapalım. Böylece pekişmesi sağlanabilir.

Örnek: İnönü stadının kapasitesi 10000'den büyük ise “Kapasitesi 10000'den büyük” mesajı yazan blok kodunu yazınız.

```

DECLARE
fark number(5);
BEGIN
    select kapasite into fark
    from Stadlar
    where isim='İnönü';
    if fark>10000 then
    DBMS_OUTPUT.put_line ('Kapasitesi 10000den büyük');
    end if;
END;

```



IF - THEN – ELSE Yapısı

- IF - THEN - ELSE yapısında ise IF içinde belirtilen koşul sağlanırsa THEN-ELSE arasındaki komutlar çalıştırılır ya da işlemler yapılır. Koşul doğru değil ise ELSE – END IF arasındaki işlemler yapılmaktadır.

IF şart THEN

Komutlar - SQL cümlelerini ifade eder.

ELSE

Komutlar - SQL cümlelerini ifade eder.

END IF; - Yapının bittiğini gösterir.

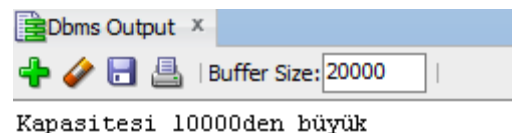
Örnek: 50 yıllık geçmişi olan kulüplerin adlarını tekrarsız ve artan listede sıralayan komutu yazınız.

Örnek: İnönü stadının kapasitesi 10000'den büyük ise “Kapasitesi 10000'den büyük” , değilse “Kapasitesi 10000'den küçük” mesajı yazan blok kodunu yazınız.

```

DECLARE
fark number(5);
BEGIN
    select kapasite into fark
    from Stadlar
    where isim='İnönü';
    if fark>10000 then
    DBMS_OUTPUT.put_line ('Kapasitesi 10000den büyük');
    else
    DBMS_OUTPUT.put_line ('Kapasitesi 10000den küçük');
    end if;
END;

```



IF - THEN – ELSEIF Yapısı

- IF - THEN – ELSEIF yapısında ise birden çok alternatif bulunmaktadır. Burada koşullar yukarıdan aşağıya doğru bakılarak, koşulların sağlanması durumunda ilgili işlemler yapılmaktadır. Söz dizimi aşağıdaki gibidir.

IF *şart* **THEN**

Komutlar - SQL cümlelerini ifade eder.

ELSEIF *şart* **THEN**

Komutlar - SQL cümlelerini ifade eder.

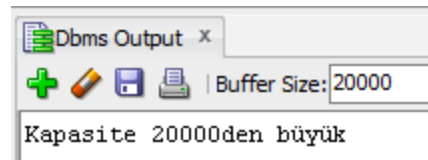
ELSE

Komutlar - SQL cümlelerini ifade eder.

END IF; - Yapının bittiğini gösterir.

Örnek: Maksimum kapasite hakkında şartlar içeren bir blok kodu yazınız.

```
DECLARE
fark number(5);
BEGIN
    select max(kapasite) into fark
    from Stadlar;
    if fark=10000 then
        DBMS_OUTPUT.put_line ('Kapasitesi 10000');
    else if fark=20000 then
        DBMS_OUTPUT.put_line ('Kapasitesi 20000');
    else
        DBMS_OUTPUT.put_line ('Kapasite 20000den büyük');
    end if;
END;
```



CASE Yapısı

CASE yapısı, IF yapısına benzer fakat çok seçenek yani çok şart olduğunda kullanılması daha uygun bir yapıdır. Birden çok IF yazmak yerine bir CASE bloğu ile içinde uygun şarttaki işlemler yapılır ya da ELSE seçeneği ile işlem bitirilir.

CASE *seçim*

WHEN *ifade A* **THEN** *işlem 1;*

WHEN *ifade B* **THEN** *işlem 2;*

WHEN *ifade C* **THEN** *işlem 3;*

WHEN *ifade D* **THEN** *işlem 4;*

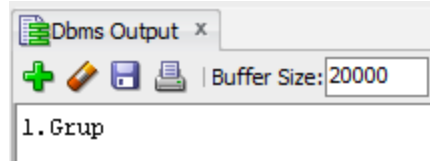
WHEN *ifade F* **THEN** *işlem 5;*

ELSE işlem son;

END CASE;

Örnek: Beşiktaş takımının sayı olarak kaçınıcı grupta olduğu kodu yazınız.

```
DECLARE
harf varchar2(1);
BEGIN
select gname into harf
from Takimler
where isim='Beşiktaş';
CASE harf
WHEN 'A' THEN dbms_output.put_line ('1.Grup');
WHEN 'B' THEN dbms_output.put_line ('2.Grup');
WHEN 'C' THEN dbms_output.put_line ('3.Grup');
WHEN 'D' THEN dbms_output.put_line ('4.Grup');
WHEN 'E' THEN dbms_output.put_line ('5.Grup');
ELSE dbms_output.put_line ('6.Grup');
END CASE;
END;
```



LOOP Yapısı

LOOP deyimi bir dizi işlemin birçok kez yapılmasını sağlamaktadır. PL/SQL içerisinde LOOP yapısından üç çeşit türemiştir ki bunlar LOOP, FOR ve WHILE döngüleridir. LOOP –END yapısının içine EXIT ifadesini koyduğumuzda orda döngüden çıkılır. Burada dikkat edilecek husus, EXIT ifadesi unutulursa, programcılar tarafından istenmeyen bir durum olan KISIR DÖNGÜ ya da SONSUZ DÖNGÜ durumu oluşmaktadır.

LOOP

.....

İşlemler

.....

END LOOP;

NOT: Bu yapı ile gerekli örnek EXIT yapısı bölümünde EXIT deyimi ile beraber verilecektir.

EXIT Yapısı

EXIT ifadesi döngüden çıkmak için kullanılan bir deyimdir. Kod çalıştırıldığında, döngü içinde EXIT gördüğünde derleyici işlemi yani döngüyü bitirip, döngüden sonra gelen deyimleri icra etmeye başlar.

LOOP

İşlemler

EXIT; -- Döngüden çıkılan yer.

END LOOP;

-- Akış buradan devam eder.

Örnek: deniz adlı personele maaşı 1500 TL değerinden yüksek olana kadar %10 oranında zaman yapan kodu yazınız.

Bu örnek için bir tablo oluşturalım. Bu tablomuzun adı **Personel** ve bu tablo **Per_Id**, **Ad**, **Soyad** ve **maaş** alanlarından oluşmaktadır.

```
DECLARE
ucret int;
BEGIN
select maas into ucret
from Personel
where ad='deniz';
LOOP
if ucret>1500 then
exit;
end if;

update Personel
set maas=ucret*1.1
where ad='deniz';

ucret:=ucret*1.1;

END LOOP;
END;
```

```
Create table Personel
( Per_Id number(5),
Ad varchar2(30),
Soyad varchar2(30),
maas int,
constraint pk_personel primary key(Per_Id)
)
```

EXIT- WHEN Yapısı

Bu EXIT yapısında ise döngünün bir koşul sonucu sonlandırılması istenmektedir. WHEN içerisinde belirtilen koşul doğru ise döngü kırılmaktadır. Temel IF yapısına benzemektedir fakat söz dizimi daha yalındır.

LOOP

İşlemler

EXIT WHEN Şart; -- Şart doğru ise Döngüden çıkılır.

END LOOP;

```
DECLARE
ucret int;
BEGIN
select maas into ucret
from Personel
where ad='deniz';

LOOP
exit when ucret>2500;

update Personel
set maas=ucret*1.1
where ad='deniz';

ucret:=ucret*1.1;

END LOOP;

END;
```

Örnek: deniz adlı personele maaşı 2500 TL değerinden yüksek olana kadar %10 oranında zaman yapan kodu yazınız.

WHILE – LOOP Yapısı

While döngüsü, şart sağlandığı sürece işlemlerin ya da komutların yapılmasını sağlar. Her defasında şart kontrol edilir. Şart tutmadığında döngüden çıkılarak, döngünün bittiği yerden program akışına devam eder.

WHILE *şart* LOOP

Komutlar
İşlemler

END LOOP;

Örnek: deniz adlı personele maaşı 2500 TL değerinden yüksek olana kadar %10 oranında zaman yapan kodu yazınız.

```
DECLARE
ucret int;
BEGIN
select maas into ucret
from Personel
where ad='deniz';

WHILE ucret<2500 LOOP

update Personel
set maas=ucret*1.1
where ad='deniz';

ucret:=ucret*1.1;

END LOOP;
```

FOR – LOOP Yapısı

FOR döngü yapısı programcılarının çok sevdiği ve özellikle iç içe döngü kullanımlarında, belirli sayılarda yapılan döngülerdir. Bazı parametreler almaktadır. Burada değişken, azalan ya da artan bir değişkendir. IN operatörü kontrol amaçlı tanımlanan değişkenin değişeceği sayı aralığının belirtildiği yerdir. REVERSE ise seçimlik olup, değerler büyükten küçüğe doğru azalacaksa kullanılır. Tam tersi işleminde kullanılmamaktadır.

FOR değişken (kontrol) **IN** [REVERSE] *değer1 .. değer2* **LOOP**

İşlemler
Komutlar

END LOOP;

Not: For döngüsünün daha yararlı örnekleri SQL Cümleleri ve Kursör yapılarında göze çarpmaktadır.

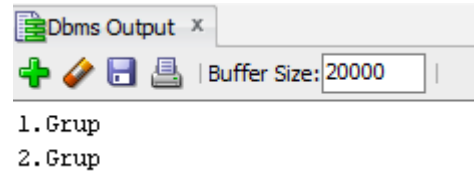
Örnek: Takımların bulundukları grupları tekrarsız bir şekilde rakam ile ekrana basan kodu yazınız.

```

DECLARE
sayi number(2);
BEGIN
    select count(Distinct(gname)) into sayi
    from takımlar;
FOR i IN 1..sayi LOOP
    dbms_output.put_line (i || '.Grup');
END LOOP;

END;

```



GOTO Yapısı

GOTO deyimi ile belirtilen, blok içerisinde tek olan etikete(label) dallanma yapılmaktadır. Programcılar tarafından fazla sevilmeyen bir yapıdır. Etiketler “<<” ile “>>” işaretleri ile yapılmaktadır. Bunun yerine diğer anlatılan yapılarla kodumuzu işlemek daha yararlıdır.

```

BEGIN
    İşlemler
    <<etiket>> -- Etiket oluşturuluyor.
    BEGIN
        İşlemler
    END;
    GOTO etiket; -- Etikete yönleniliyor.
END;

```

```

DECLARE
sayi number(2):=1;
BEGIN
    <<dmg>>
    if sayi<6 then
        sayi:=sayi+1;
        goto dm;
    end if;
END;

```

Örnek: Kodun başında 1 değeri verilen **sayi** değişkeninin şartta kadar 1 arttırılmasını sağlayan kodu yazınız.

NULL Yapısı

NULL ifadesinin aslında çok büyük bir fonksiyonu yoktur. Fakat NULL ile kodun akışında koşulun sağlanıp sağlanamadığı izlenmiş olunur. Hiçbir işlem gerçekleştirmez. Kod kaldığı yerden devam eder.

```

BEGIN
    İşlemler
    IF şart THEN
        İşlemler
    ELSE

```