

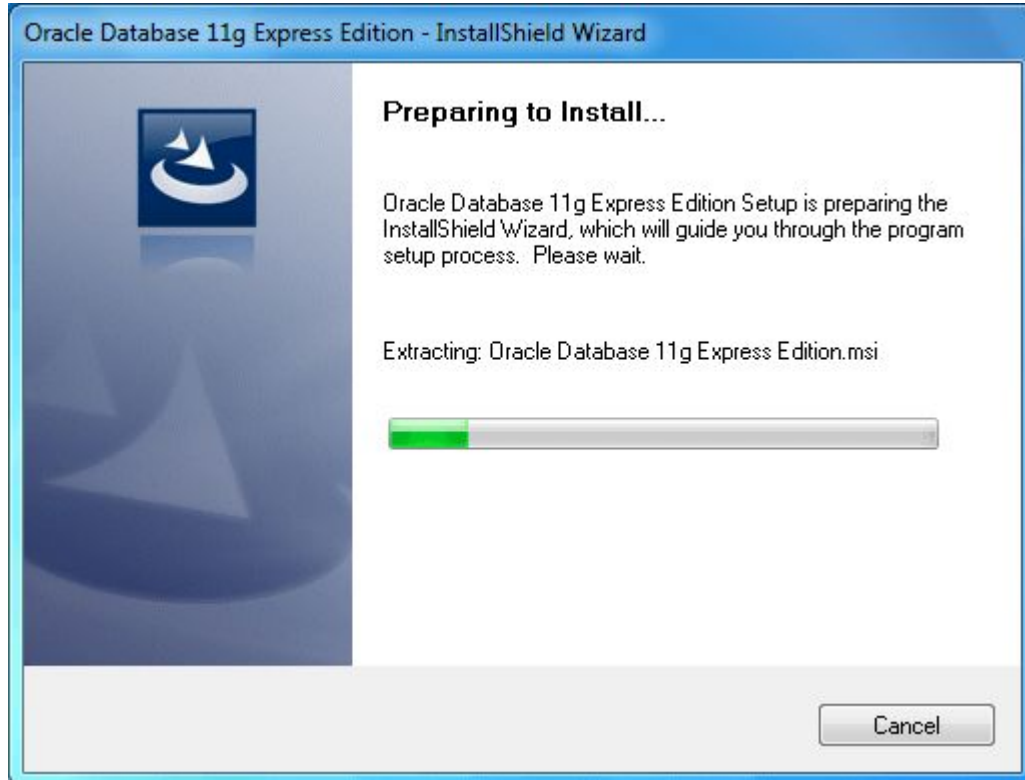
Veri tabanı programlama

Onur İNAN

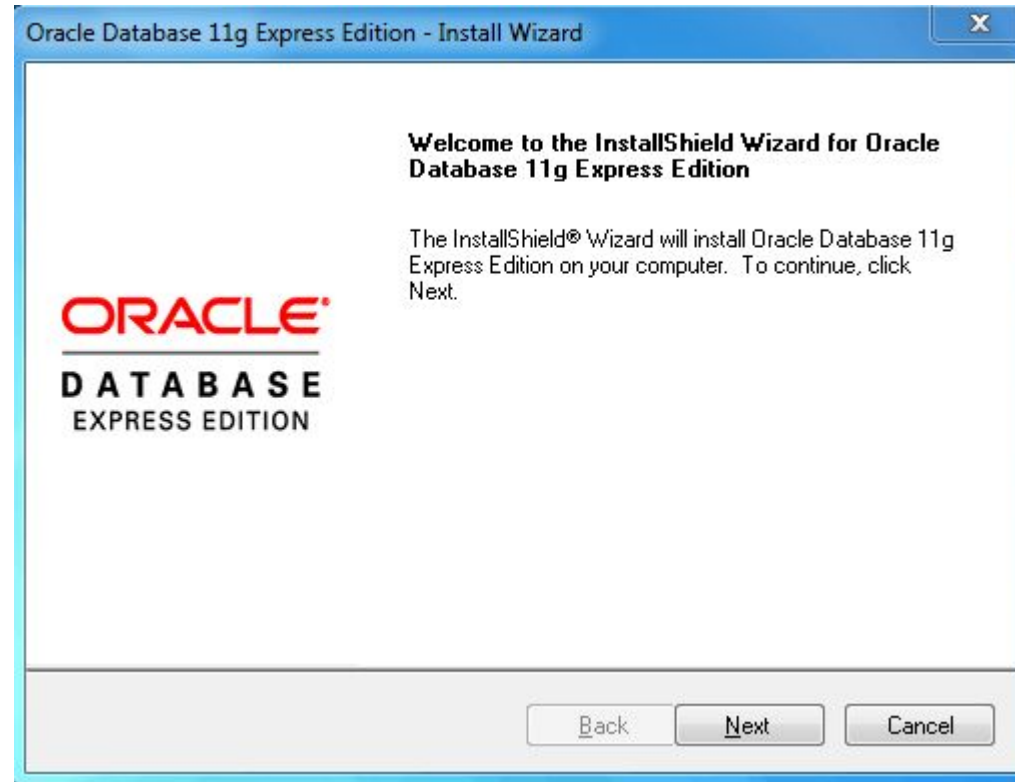
Araçlar:

- Oracle Express Sürümü
- <https://www.oracle.com/database/technologies/xe-downloads.html>
- <https://www.oracle.com/database/technologies/xe-prior-release-downloads.html>
- Sql developer yada Pl/Sql Developer

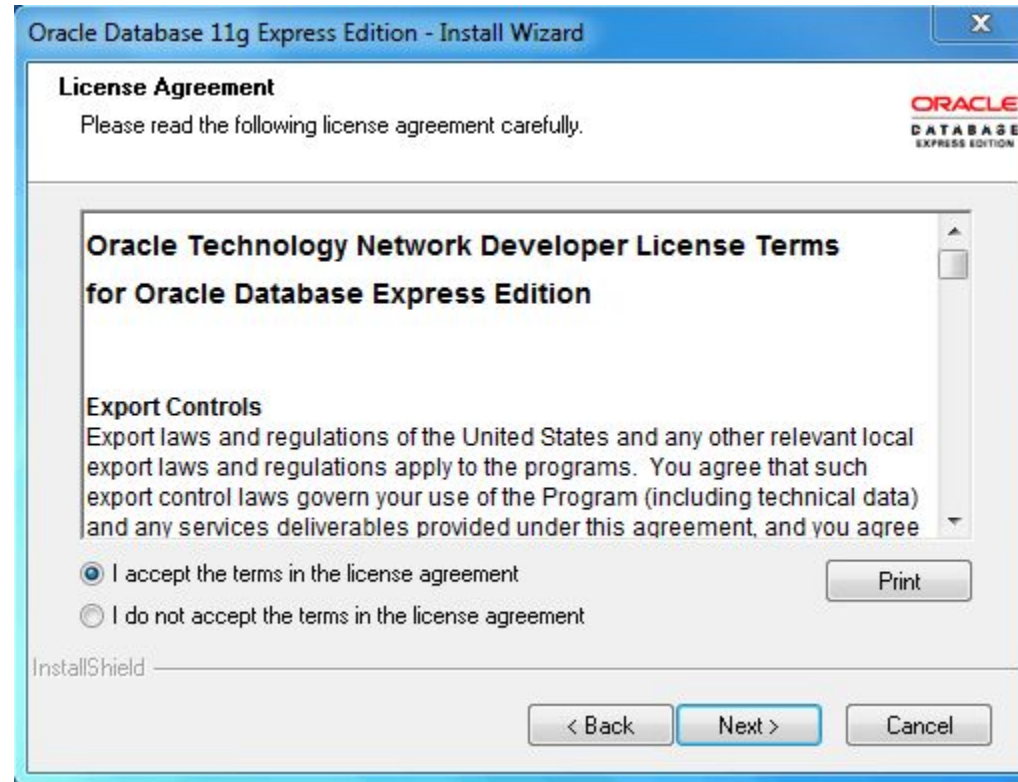
ORACLE EXPRESS KURULUM



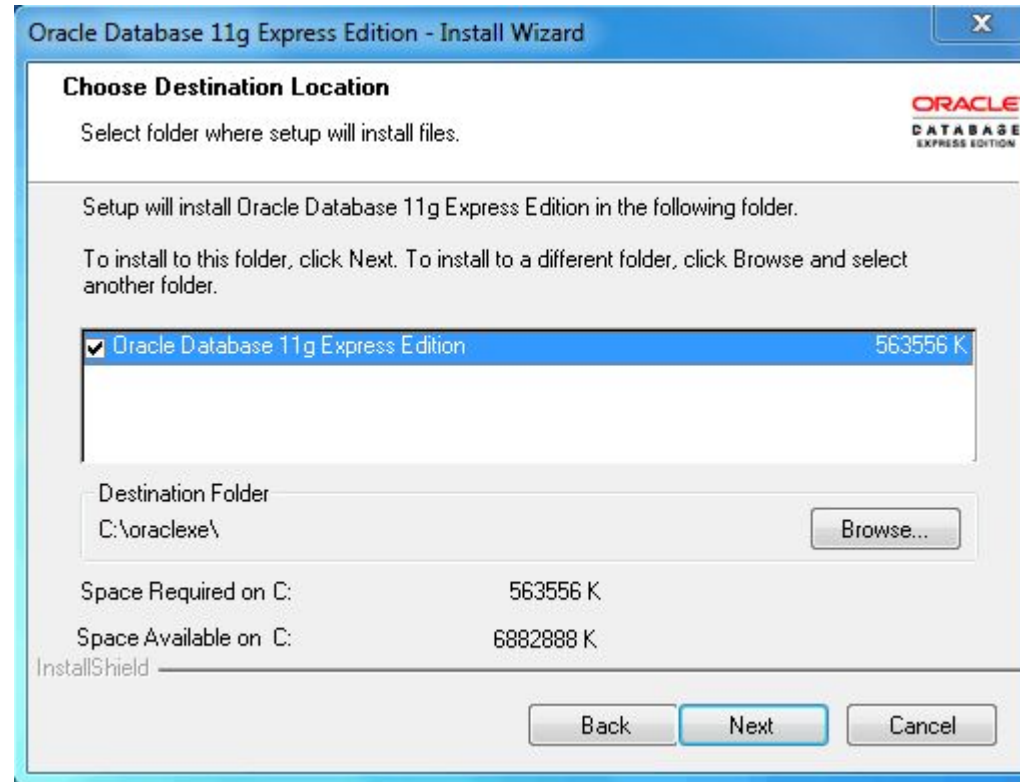
ORACLE EXPRESS KURULUM



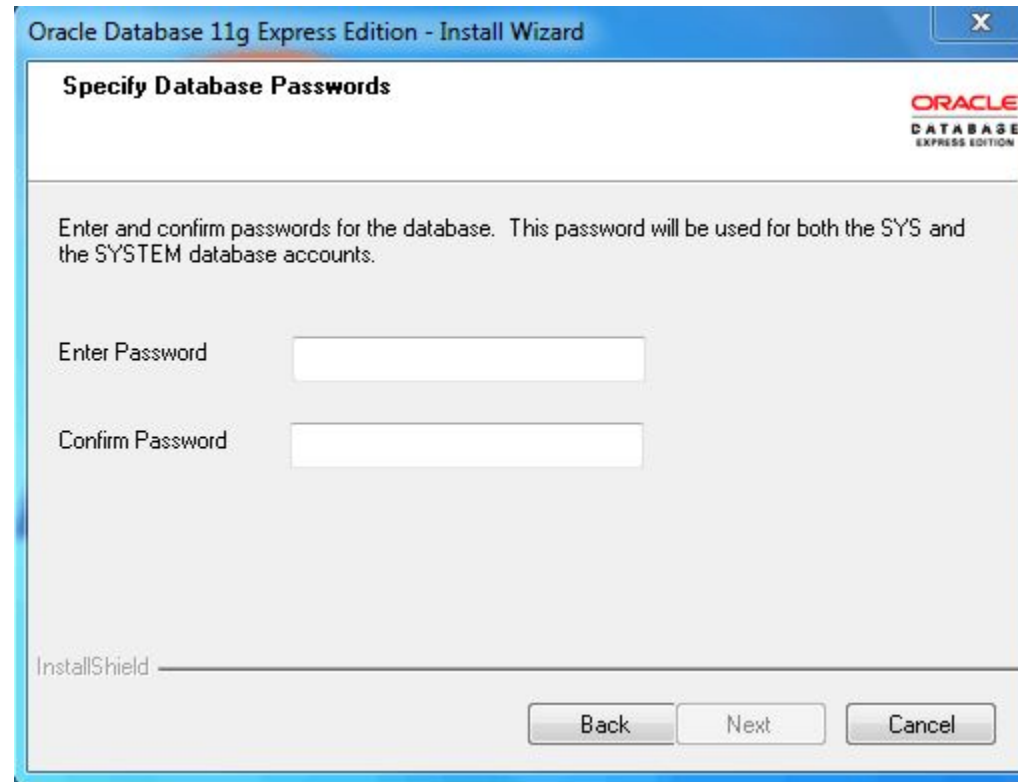
ORACLE EXPRESS KURULUM



ORACLE EXPRESS KURULUM



ORACLE EXPRESS KURULUM



Oracle Database 11g Express Edition - Install Wizard

Specify Database Passwords

ORACLE
DATABASE
EXPRESS EDITION

Enter and confirm passwords for the database. This password will be used for both the SYS and the SYSTEM database accounts.

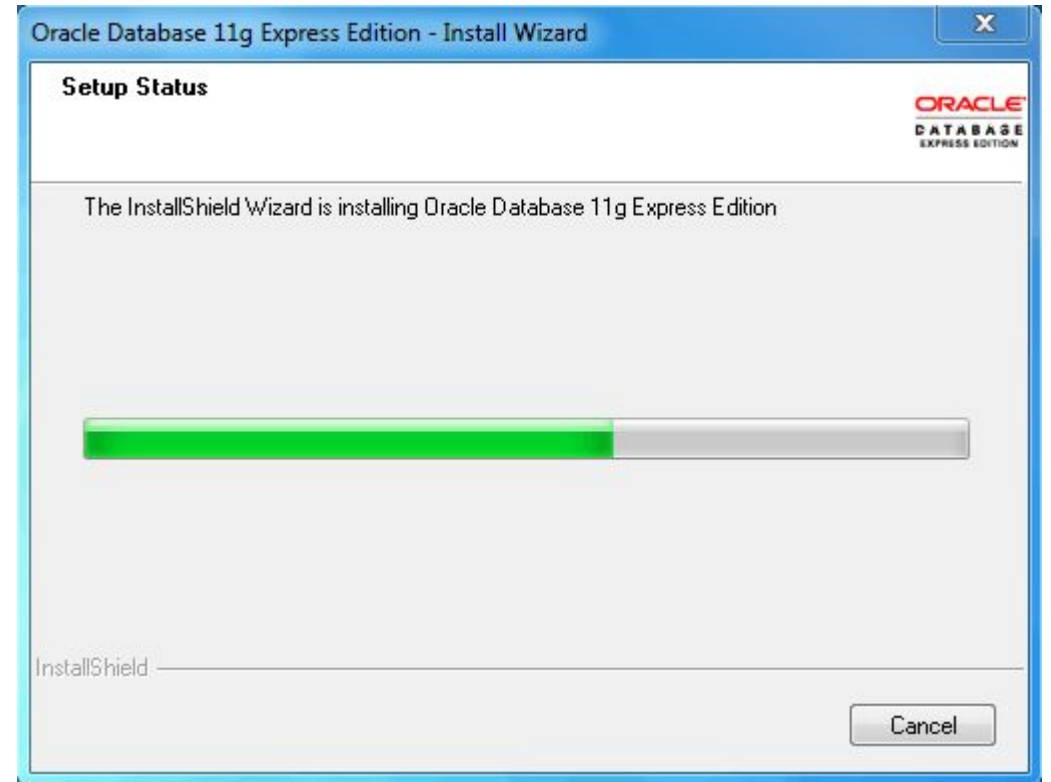
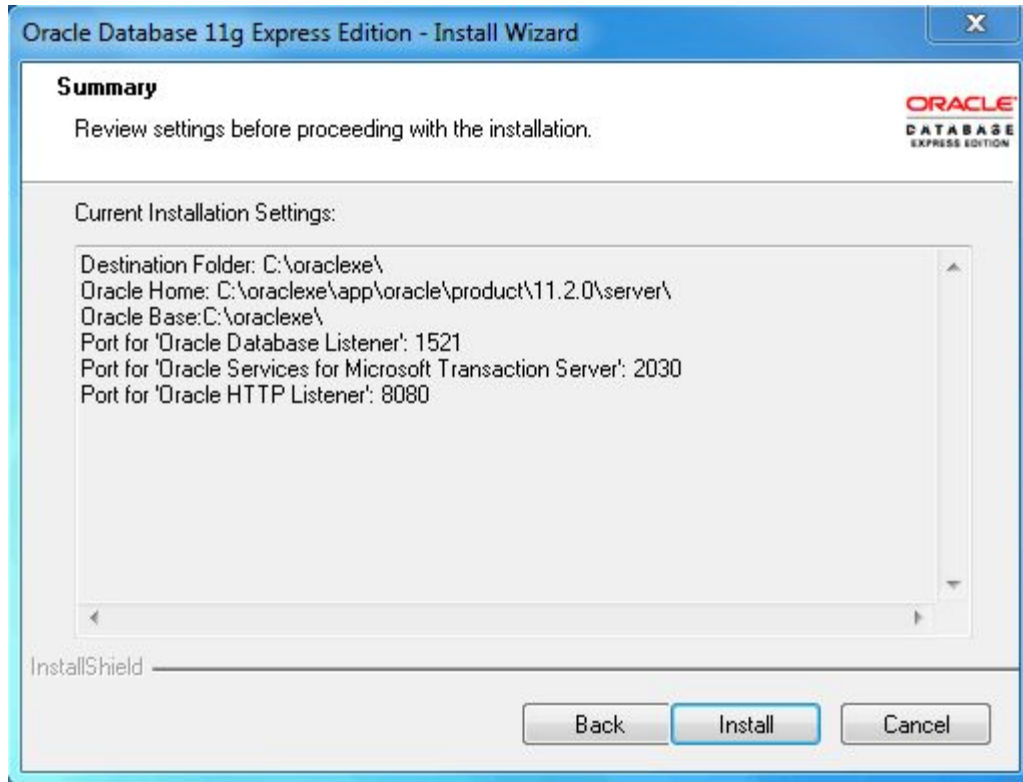
Enter Password

Confirm Password

InstallShield

Back Next Cancel

ORACLE EXPRESS KURULUM



Create table Foreign KEY

```
create table ogrenci(  
    ogrno int not null,  
    adisoyad varchar(25),  
    fakulte varchar(25),  
    bolum varchar(25),  
    CONSTRAINT pkkey1 PRIMARY  
    KEY(ogrno))
```

```
create table derskaydi(  
    ogrno int,  
    dersno varchar(11),  
    vize int,  
    final int,  
    CONSTRAINT fkkey1 FOREIGN KEY(ogrno) REFERENCES Ogrenci(ogrno) on delete cascade, --on update cascade  
    CONSTRAINT fkkey2 FOREIGN KEY(dersno) REFERENCES ders(dersno) on delete cascade) --on update cascade )
```

```
create table ders(  
    dersno varchar(11) not null,  
    dersadi varchar(25),  
    kredi int,  
    saat int,  
    CONSTRAINT pkkey2 PRIMARY  
    KEY(dersno))
```

Create table otomatik artış (mysql)

```
create table personel(  
    sicil int not null AUTO_INCREMENT=1,  
    ad varchar(25),  
    soyad varchar(25),  
    adres varchar(250),  
    sehir varchar(250),  
    UNIQUE(sicil)  
)
```

Sequence

- Create sequence sequence_adi
MINVALUE en_düşük_değer
MAXVALUE en_yuksek_değer
START WITH ilk_değer
INCREMENT BY artis_miktari
CYCLE / NOCYCLE

Insert into tablo(sicil,ad,soyad) values (sayac.nextval,'AAA','BBB')

Create table Foreign KEY (kütüphane)

```
create table kitap(  
    kitapid int not null,  
    adi varchar(25),  
    yazar varchar(25),  
    yer varchar(25),  
    CONSTRAINT pkkey1 PRIMARY  
    KEY(kitapid))
```

```
create table odunc(  
    kitapid int,  
    uyeno varchar(11),  
    bastar date,  
    bittar date,  
    CONSTRAINT fkkeykit1 FOREIGN KEY(kitapid) REFERENCES Kitap(kitapid) on delete cascade, --on update cascade  
    CONSTRAINT fkkeykit2 FOREIGN KEY(uyeno) REFERENCES uye(uyeno) on delete cascade, --on update cascade )
```

```
create table uye(  
    uyeno varchar(11) not null,  
    adsoyad varchar(25),  
    adres varchar(25),  
    tel varchar(25),  
    CONSTRAINT pkkeykut2 PRIMARY  
    KEY(uyeno))
```

ALTER

- Daha önce oluşturulmuş veri tabanı nesnesinin özelliğini değiştirmek için kullanılır. Yapılmak istenen değişiklik parametre olarak verilir.

- Ekleme

ALTER TABLE tablo ADD sutunadi özellikler

- Silme

ALTER TABLE tablo DROP COLUMN sutunadi özellikler

- Değiştirme

ALTER TABLE tablo ALTER COLUMN sutunadi özellikler

Alter

```
ALTER TABLE OGRENCI ADD DOGTAR date not null;
```

```
ALTER TABLE OGRENCI ADD UNIQUE(BOLUM);
```

```
ALTER TABLE OGRENCI ADD CONSTRAINT ucde UNIQUE(FAKULTE);
```

```
ALTER TABLE OGRENCI ADD PRIMARY KEY (OGRNO)
```

Yada

```
ALTER TABLE OGRENCI ADD
```

```
CONSTRAINT Pk_key PRIMARY KEY (OGRNO,AD)
```

ALTER

- ALTER TABLE tderskaydi ADD CONSTRAINT fkkey2 FOREIGN KEY(dersno) REFERENCES ders(dersno) on delete cascade
- ALTER TABLE odunc ADD CONSTRAINT fkkeykit2 FOREIGN KEY(uyeno) REFERENCES uye(uyeno) on delete cascade
- ALTER TABLE tderskaydi ADD CONSTRAINT chk_tdrs CHECK (not1>0 and not1<=100)

ALTER

- ALTER TABLE ogrenci modify Dogtar varchar(15) null
- ALTER TABLE *OGRENCI* DROP COLUMN *Dogtar*
- ALTER TABLE *OGRENCI* DROP CONSTRAINT *adi*

DROP

DROP TABLE tablo_adı

DROP TABLE OGRENCI

DROP DATABASE Veritabanıadı

TRUNCATE TABLE Tablo_adı

TRUNCATE TABLE OGRENCI1

Veri İşleme Dili (DML)

- SELECT
- INSERT
- UPDATE
- DELETE

Select

- Tablo veya tablolardan istenen verilerin seçimi için kullanılır. Listeleme işlemi sırasında kayıtlarda herhangi bir değişim meydana gelmez.
- `Select` sütunlar `from` `tablo_adi`
- Sütunlar: İstenilen sütunları aralarına virgül konarak listelenebilir. * kullanılırsa tüm kayıtlar getirilir.
- `Tablo_adi` : Seçim yapılacak tablo veya tablolar çağrılır.

Select

- Sorgulanacak tablonun her sütunu yazılabilir.
- Bir veya birden fazla tablo ile sorgu yapılabilir.
- Kayıtlar belli koşullara göre sorgulanabilir.
- Sorgulama esnasında dönen değerler birleştirilip, eklemeler yapılabilir.
- Sorgudan dönen değerlere matematiksel işlemler yapılabilir.
- Sorgudan dönen işlemler sıralanabilir.
- Ortak sütunlara sahip tablolar birleştirilip, birden fazla tablolarda sorgular yapılabilir.
- Karmaşık sorgular elde etmek için iç içe select ifadeleri kullanılabilir.

Select (tablo ismi kutuk)

Ogrno	Ad	soyad	Fakulte	Bolum
071213001	Murat	Kaçar	Müh.Mim	Bilgisayar
071212002	Şaban	Solmaz	Müh.Mim	Elektrik
071413003	Sabri	Çağlar	Eğitim	Türkçe
071513004	Burak	Yılmaz	Fen	Matematik

```
Create table kutuk (  
Ogrno varchar2(12),  
Ad varchar2(12),  
Soyad varchar2(12),  
Fakulte varchar2(50),  
Bolum varchar2(50),  
CONSTRAINT pkkey_kut1 PRIMARY KEY(ogrno)  
)
```

```
Select * from kutuk
```

```
Select ogrno,ad from kutuk
```

```
Select * from kutuk where ogrno='071213001'
```

Select

- Select ifadesi ile yapılan sorgularda belirli bir koşula bağlı yapıların seçilmesi isteniyorsa tablo isminin ardından 'where' ifadesi ile koşul belirtilir.
- Where ifadesinde karşılaştırma operatörü (>,<,<=,>=,NOT) kullanılarak oluşturulur.
- Birden fazla koşul varsa birbirine bağlamak için AND veya OR operatörü kullanılır.
- In ifadesiyle veri kümesi seçilmesi için koşul ifadeleri ve like ifadesi ile joker karakter anlamına gelen _ (alt çizgi) tek karakterlik herhangi bir şey anlamına gelir. % ifadesi kendisinden sonra herhangi bir şey anlamına gelir.

Select

- Select * from kutuk where ad='Murat' AND Soyad='Kaçar'
- Select * from kutuk where ad='Murat' OR Soyad='Solmaz'
- Select * from kutuk where ogrno like '171213%'
- Select * from kutuk where ogrno like '17__13002'

insert

- Veri tabanı sistemlerinde herhangi bir tabloya yeni veri girişi için **insert into** ifadesi kullanılır. Eklenecek veri karakter ise " tırnak işaretleri içinde numerik değer ise olduğu gibi yazılır.
- Insert into tablo values (deger1,deger2,deger3,...)
Bütün değerlerin girilmesi gerekiyor.
- Insert into tablo (veri sütunları) values (değerler)
Primary key ve not null ifadeleri yazılmalı diğer sütunlar yazılmasa da olur.

Insert into

- Insert into values ('171213008','Veli','Kaçar','Muh.Mim',null)
- Insert into (ogrno,adi,soyadi,fakultesi) values ('171213009','Selim','Uyar','Muh.Mim')

Update

- Tablo içinde var olan bilgileri değiştirmek için kullanılır.

Update tablo_ismi

Set sütun1=değer,sütun2=değer1

Where şartlar

Update Ogrenciler

Set ogr_ad='VELİ'

Where ogr_ad='ONUR'

Delete

- Tablo içerisindeki kayıtların tamamını yada belirli bir koşula uyanlarını silmek için kullanılır.

- Delete From Tabloadi where sartlar

delete ogrenciler where ogr_ad='ONUR'

- Truncate table tabloadi
- Truncate table ogrenciler

Gruplandırarak Sorgulama

Gruplandırma işlemi bir tablo içersin de ortak özelliklere sahip satırları kapsamaktadır. Ortak özelliğe satır tek bir satır bile ortak özelliğe sahip olur.

Satırlarda birden fazla satır kümesiyle işlem yapılacaksa grup fonksiyonları kullanılır. Grup fonksiyonları tablonun tamamına uygulandığı gibi group by komutu ile belli bir gruba da uygulanabilir.

Select *sütunad from tabload*

Where *şart*

Group by *grupnacak sutun adları*

Having *Şart*

Order by *Sıralacak sütün adları*

Gruplandırma işleminde kullanılan fonksiyonlar

AVG(): Sorgu sonucuna göre gruplanan sütuna göre fonksiyona gönderilen sütunun ortalamasını bulur.

Count(): Sorgu sonucuna göre gruplanan sütuna göre içerdikleri satır sayısını bulur.

Max(): Sorgu sonucuna göre gruplanan sütuna göre fonksiyona gönderilen sütunun en büyük değerini bulur.

Min(): Sorgu sonucuna göre gruplanan sütuna göre fonksiyona gönderilen sütunun en küçük değerini bulur.

Sum():Sorgu sonucuna göre gruplanan sütuna göre fonksiyona gönderilen sütunun toplam değerini bulur.

Gruplandırarak Sorgulama

```
select count(),max(yas),min(yas),sum(yas),avg(yas) from ogrenci
```

Fakültelere göre

```
select fak,count(),max(yas),min(yas),sum(yas),avg(yas) from ogrenci
```

Group by fak

Bölüme göre

```
select bol,count(),max(yas),min(yas),sum(yas),avg(yas) from ogrenci
```

Group by bol

Birden fazla sütun ile gruptama

- Birden fazla sütuna göre yapılacak gruptama işlemleri tek sütuna yapılan işlemlerle aynı kurallara tabiidir.
- Birden fazla sütuna göre gruptamalarda aralarına virgül konarak gruptama yapılır.

Select fakulte,bolum,count(*) from ogrenci

Group by fakulte,bolum

Gruplandırma işlemlerinde koşul ifadeleri

- Koşullu gruplandırma işlemi yapılırken, koşul herhangi bir fonsiyon sonucuna göre değil sütünün ismine göre yapılırsa where ifadesinin içine yazılır.
- Gruplandırma yapılırken şart fonksiyona bağlı ise having kullanılarak yapılır.

```
Select bolum,count(*) from ogrenci  
Group by bolum  
Having count(*)>20
```

```
Select bolu,count(*) from ogrenci  
Where bolum like '%B0il%'  
Group by bolum
```


Birden Fazla Tablo üzerinde Sorgulama

- Birleştirme işlemlerinde join kullanılabileceği gibi where işlemlerinde ortak kolon ifadeleri yazılarak da birleştirme işlemleri yapılabilir.
- Birleştirme işlemleri view üzerinde de yapılabilir.
- Birleştirilen sütunlarda veri tipi uyumlu olmalıdır.
- Birleştirilen sütunların aynı isme sahip olması gerekmez.
- Sorguların daha hızlı çalışması için sütunlara index tanımlanır.

Where ile birleştirme

- Birleştirme işlemleri ilk aşamalarda where ile yapılmaktaydı. Daha sonra JOIN veya where ifadeleriyle yapılmaya başlanmıştır.

```
Select ISBN,yazar_adi,yazar_soyadi from kitap_yazar a,Yazarlar b  
Where a.yazarno=b.yazarno
```

Inner Join

- İki veya daha fazla tablonun ortak sütunlarının içerdiği verileri kontrol ederek birleştirme işlemi yapar. Inner join yerine sadece join yazmakta yeterlidir. İstenirse birleştirilen tablo where ifadesiyle özelleştirilebilir.

```
Select tablo1.sütunadi,tablo2.sütunadi  
from tablo1 inner join tablo2  
on tablo1.ortaksutun karşılaştırma operatörü tablo2.ortaksutun
```

Karşılaştırma operatörü çoğunlukla '=' operatörüdür. Ama diğer karşılaştırma operatörleri de (=,<>,>,>=,<,<=) kullanılabilir.

Outer Join

- Inner join tablolaradaki ilişkili satırları gösterirken ilişkisiz satırları göstermez. Outer join ile ilişkili olmayan satırlarda gösterilir.
- Üç Grupta incelenir :
 - Left Outer Join
 - Right Outer Join
 - Full Outer Join

Left Outer Join

- İfadenin solundaki tablo belirleyici rol oynamaktadır. Bu tablonun diğer tablodaki bilgilerle ilişkisi bulunsun yada bulunmasın tablodaki veriler görüntülenir.

Select tablo1.sütunadi,tablo2.sütunadi

from tablo1 left outer join tablo2

on tablo1.ortaksutun karşılaştırma operatörü tablo2.ortaksutun

- And ve Or operatörleriyle birden fazla koşul verilebilir.

Right Outer Join

- İfadenin sağındaki tablo(tablo2) belirleyici rol oynamaktadır. Bu tablonun diğer tablodaki bilgilerle ilişkisi bulunsun yada bulunmasın tablodaki veriler görüntülenir.

Select tablo1.sütunadi,tablo2.sütunadi

from tablo1 Right outer join tablo2

on tablo1.ortaksutun karşılaştırma operatörü tablo2.ortaksutun

- And ve Or operatörleriyle birden fazla koşul verilebilir.

Full Outer Join

- Diğer gördüğümüz iki join işleminin birleşim kümesidir. İki tablonda tüm satırları listelenir. İlişkili olmayan değerlerin sütunu null olur.

```
Select tablo1.sütunadi,tablo2.sütunadi  
from tablo1 Full outer join tablo2  
on tablo1.ortaksutun karşılaştırma operatörü tablo2.ortaksutun
```

- And ve Or operatörleriyle birden fazla koşul verilebilir.

Cross Join

- Birleştirilen tablolardaki tüm satırların listelenmesi sağlanır. Sorgu sonucu iki tablonun satırlarının çarpımı kadar satırdan oluşmaktadır.
- Ortak sütunu olma zorunluluğu yoktur.

```
Select tablo1.sütunadi,tablo2.sütunadi  
from tablo1 cross join tablo2
```


Natural Join

- Inner join ' e benzerdir. Bu tür birleştirmelerde koşul ifadesi belirtmeye gerek yoktur. Ortak sütunlar otomatik birleştirilir. Bu birleştirme ile satır tekrarı olmaz.

```
Select tablo1.sütunadi,tablo2.sütunadi  
from tablo1 natural join tablo2
```

Self Join

- Birleştirme işlemi tek tablodaki verileri kullanarak da yapılır. Diğer join işlemlerinin ortak kullanımındır.

```
Select a.perid,a.ad,b.ad 'Yönetici'  
from personel a left outer join personel b  
on a.perid=b.perid
```

Ortak Sütunu Bulunmayan Tablo Birleştirme

- Ortak sütunu bulunan tablolar where veya Join işlemiyle birleştirilirler. Eğer birleştirme yapılacak tablolarda ortak sütun yoksa birleştirme işleminde koşul olarak eşitleme yerine between ifadesi kullanılır.
- Select a.ogrno,a.dersno,a.hkar from ogrnotlar a,notlar b
Where a.notu between b.ilc and b.son

İki sorgunun kesişimi (Intersect)

- Aynı sütun isimlerine sahip iki veya daha fazla sorgudan dönen değerlerin kesişim kümesini almak için kullanılır.

Select * from tablo1

Intersect

Select * from tablo2

İki sorgunun farkı(Except/Minus)

- Aynı sütun isimlerine sahip iki sorgudan dönen değerlerin fark kümesini almak için kullanılır.

Select * from tablo1

Except yada Minus

Select * from tablo2

İki sorguyu Birleştirme (Union ve Union All)

- İki tablodan aynı anda veri almak için kullanılır. Tablodan çekilen sütunların veri tipleri aynı içeriği farklı olması gerekir.
- İki sorgu birleştğinde aynı veri her iki tabloda da varsa her ikisi de listelenir.

Select * from tablo1

Union yada Union All

Select * from tablo2

Alt Sorgular

- Tabloların birleştirilmesi haricinde bu zamana kadar anlatılan sorgular tek bir sorgudan oluşmaktadır.
- Alt sorgular, bir tablo içerisinde başka bir sql sorgusu kullanılarak oluşturulan sql ifadeleridir.
- Alt sorgular, çoklu tablolardan değer almanın farklı bir yoludur.
- Alt sorgular SELECT, INSERT, UPDATE ve DELETE sql ifadeleriyle =, >, <, >=, <= gibi karşılaştırma operatörleri veya in, not in gibi veri grubu içeren ifadelerle kullanılabilir.

Alt Sorgularda Dikkat Edilecek Noktalar

- Alt sorgu içerisinde listelenecek sütun sayısı birden fazla olmamalıdır.

select ad,soyad from öğrenci ----- >> yanlış

Select ad from öğrenci -----<?> doğru

- Alt sorgular parantez içine alınmalıdır.
- Order by veya group by içerisinde alt sorgu kullanılamaz.
- Alt sorgu ve temel sorgu içerisinde sütun isimleri takma isimler verilerek kullanılabilir.
- Temel sorgunun koşul kısmında karşılaştırma operatörleri kullanıldığı zaman tek değer dönmesi gerekir. Birden fazla değer dönerse In veya Not IN gibi koşul cümleleri kullanılır

Alt Sorgu Oluşturulması

- Select sütunlar from tablo where koşul (select sütun from tablo)

select * from öğrenci where fakulteno = (select fakno from fakülte where fakadi='Müh.mim.')

Select a.*, (select adi from fakülte b where b.fakno=a.fakulteno) from öğrenci a

- Insert into yeni_öğrenci Select a.*, (select adi from fakülte b where b.fakno=a.fakulteno) from öğrenci a

Update öğrenci a

Set a.fakulteno=(select b.fakno from bolum b where b.bolumno=a.bolumno and b.arşiv=0)

Where a.bolumno =(select bolumno from bolum where adi='BİLGİSAYAR MUH.' and arşiv=1)

IN ve NOT IN

- Select * from öğrenci where bolumno in (3,6,8,9)
- Select * from öğrenci where bolumno in (select bolumno from bolum where adi like '%Bilgisayar%')

Group by ifadesi

- Select * from bolum where bolumno in (select bolumno from öğrenci group by bolum having count(*)<20)

Any - ALL

- Select * from derskaydi where gnnott <any (select gnnott from derskayit where ogrno like '12%')
- Where vizenot > any(10,30,50) → 10 dan büyük olması yeterli
- Where vizenot > all(20,30,50) → 50 den büyük olması yeterli
- Select * from derskaydi where gnnott > all (select gnnott from derskayit where ogrno like '12%')

Sql – Yapısal sorgulama Dili

- Yapısal sorgulama dili anlamana gelen Sql ilişkisel veri tabanlarında çok geniş kullanım alanına sahiptir.
- Sql ile kullanıcılar veri tabanı ile iletişim kurmaktadır.
- Veri tabanlarının tamamı bu dili kullandığı için standart haline gelmiştir.
- Günümüzde kullanılan programlama dillerinin tamamı sql komutlarını desteklemektedir.
- Standart sql komutlarında fonksiyon, döngü ve karşılaştırma ifadeleri kullanılmamaktadır.

Sql – Yapısal sorgulama Dili

- Fonksiyon, döngü gibi işlemlerin yapılabilmesi için PL/SQL ve T-SQL sorgulama dili geliştirilmiştir.
- Bu diller standart sql dilini içerdiği gibi eğer, döngü gibi ifadeleri de içermektedirler.
- PL/SQL :Oracle firması tarafından geliştirilmiştir. Temel sql komutlarının yanında akış ve değişkenlerin kullanımına olanak sağlar.
- T-sql : Microsoft ve sysbase tarafından geliştirilmiştir. Temel sql komutları yanında akış ve değişkenlerin kullanımına olanak sağlar.

Pl/sql Dili

- Sql ifadeleri programlamaya yönelik ifadeleri barındırmadığı için belli işlemlerde yetersiz kalmaktadır. Oracle veri tabanı bu eksikliği gidermek amacıyla veri tabanına Pl/Sql 'i geliştirmiştir.
- PL/SQL'in açılımı SQL'in prosedürel dil uzantısı şeklindedir. (Procedural Language Extension of SQL). PL/SQL genel olarak Sql dilinin çeşitli programlama dillerinin özellikleriyle birleştirilmesidir. Bu kombinasyon programlama dillerinden bazı özellikleri miras edindiği gibi (örneğin For döngüsü, if then else kondisyonel karşılaştırmaları, While döngüleri) içinde Sql sorguları kullanıla bilinir.

PL/SQL'in Avantajları:

Blok Yapılar:

PL/SQL kod bloklarını içerir. Bunlar iç içe konumlandırılabilir. Her blok bir görev veya mantıksal bir yapı içerebilir veya onlardan oluşur. Bu PL/SQL blokları veritabanında kaydedilip saklanabilir.

Prosedürel Dil Yeteneği:

PL/SQL içinde çeşitli programlama yapıları eklenip kullanılabılır.

Performans:

PL/SQL motoru çoklu SQL ifadelerini aynı anda işleyip ağdaki trafiği azaltabilir.

Hata ve İstisna İdaresi:

PL/SQL programı hataları verimli bir şekilde idare edebilir. Bir kere hatayla karşılaşıldığında buna karşılık gelen uygun aksiyonları gerçekleştirip, hata mesajlarını ekranda gösterebilmektedir.

- En temel Pl/Sql ifadesi aşağıda belirtilen kısımlardan oluşur.

DECLARE

Tanımlama bölümü

BEGIN

Gerçekleştirilecek işlemler

EXCEPTION

Hata durumunda gerçekleştirilecek işlemler

END;

Örnek kod:

Declare

```
    mesaj varchar2(256):='Merhaba dünya';
```

Begin

```
    Dbms_output.put_line(mesaj);
```

End;

Birden fazla blok kullanımı :

DECLARE

Tanımlama bölümü

BEGIN

Gerçekleştirilecek işlemler

DECLARE

BEGIN

END;

EXCEPTION

Hata durumunda gerçekleştirilecek işlemler

END;

Değişken Tanımlama

- Değişkenler, programın veya kodlarının icra süresince belirli bir değer tutan ve istenildiği zaman tuttuğu değeri değiştirilebilen programlama elemanlarıdır.
- Pl/Sql 'de declare bölümü içerisinde tanımlanır.

Değişken isimlendirme kuralları

- Değişkenleri ilk karakteri harf olmak üzere; sonraki karakterler harf, rakam ve '_' den oluşabilir.
- Değişken türleri Türkçe karakter içermez.
- Sql için kullanılan ifadeler değişken değişken adı olarak kullanılamaz.
- Değişken isimlerinde büyük küçük harf ayrımı yoktur.
- Değişken isimleri içinde boşluk karakteri olamaz.

Değişken tanımlama

Declare

Değişken_adı [CONSTANT] veri_tipi [Default değer] [Not null]
[:=ilk_değer]

Değişkenler içinde tutulacak değere göre belirlenmelidir. Sayısal değerler number, karakterler varchar olarak tanımlanır.

[...] ifadeleri arasındaki ifadeler isteğe bağlıdır.

Değişkenler ayrı satırlarda tanımlanmalıdır.

Değişken tanımlama

Declare

numara1,numara2 number; --?yanlış tanımlama

Declare

Numara1 number;

Numara2 number;

Declare

Adi_soyadi varchar(20):='XXX YYY';

Değişkene değer atama

- Değişken := atanacak_değer
- Declare
 - say1 number;
 - say2 number;
 - toplam number;
- Begin
- Say1:=5;
- Say2:=6;
- Toplam:=say1+say2;
- dbms_output.put_line('Toplama sonucu : ' || toplam);
- End;

Değişkene değer atama

Sorgu sonucundan değer atanacak ise

```
Select sutunadi into Değişken_ismi from tablo_adi;
```

Declare

```
    kitapsay number;
```

Begin

```
Select count(*) into kitapsay from kitaplar;
```

```
End;
```

Açıklama Ekleme

Pl/Sql kodları arasına açıklama satırı aşağıdaki şekillere göre eklenir.

-- işareti tek satırlık açıklamalarda kullanılır.

/* */ işaretleri birden fazla satır içeren açıklamalarda kullanılır.

DBMS_OUTPUT.PUT_LINE

Ekrana bilgi yazdırmak için kullanılır.

Pl/Sql bloğundan önce

Set Serveroutput on ifadesinin yazılması gerekir.

Set serveroutput on

Declare

deger1 number:= 150;

deger2 number:= 75;

Begin

DBMS_output.put_line('Sonuc toplamı :');

DBMS_output.put_line(deger1+deger2);

End;

Dışarıdan Bilgi Alma

Dışarıdan verilen ifadenin karesini alan bir pl/sql kodu nasıl yazılır.

- Declare
- Deger1 number:= °er_gir;
- Begin
- DBMS_output.put_line(deger1 || "'in Karesi:' || power(deger1,2));
- End;

Exception veri kontrolü

- Declare
- ogrencino varchar(20):=&ogrenci_no;
- Adsoyad varchar(40);
- Begin
- Select f.adsoyad into adsoyad from ogrenci f
- Where ogrno=ogrencino;
- Dbms_output.put_line('Öğrencinin adı soyadı:' || adsoyad);
- Exception
- When No_Data_found then
- Dbms_output.put_line('!! Öğrenci bulunamamıştır !!');
- End;

%Rowtype

- Record tipi değişkenler bir tablonun içerdiği bir satırın tüm bilgilerini içerecek şekilde de tanımlayabilir.
- `Degisken tablo_adi%ROWTYPE;`

Değişken bilgisine ulaşmak için
`degisken.kolon_ismi` şeklinde bilgiye ulaşılır.

Declare

ogrencino varchar(20):=&ogrencino;

bilgi ogrenci%rowtype;

Begin

Select * into bilgi from ogrenci f

Where ogrno=&ogrencino;

Dbms_output.put_line('Adsoyad :'| | bilgi.adsoyad);

Exception

When No_Data_found then

Dbms_output.put_line('!! Öğrenci bulunamamıştır !!');

End;

Pl/sql Akış Kontrolleri

- Tüm programlama dillerinde kullanılan if,while v.b akış kontrolleri bu dil içerisinde de vardır.
- Koşullu ifadeler :

Bir işlemin bir veya daha fazla koşula bağlı olduğu işlemlerde gerçekleştirilir. Koşullu ifadeler için IF ve CASE yapısı kullanılır.

- Döngüler:

Grup işlemlerin birden fazla tekrar etmesi istenilen durumlarda kullanılır. Döngüler için Loop,while,for yapısı kullanılır.

IF....then Yapısı

- Bir veya daha fazla pl/sql ifadesinin gerçekleşmesinin koşula bağlı olduğu durumlarda kullanılır.

IF koşul_ifadesi THEN

 Koşul Doğru ise yapılacaklar....

END IF;

Örnek if kodu

- Declare
- degal number(2):=°al;
- Begin
-
- if degal>50 then
- Dbms_output.put_line('!! Girdiğin sayı 50 den büyüktür !!');
- end if;
- End;

- Declare
- degbul number(2);
- Begin
-
- select count(*) into degbul from ogrenci;
- if degbul>20 then
- dbms_output.put_line('Öğrenci sayısı 20 yi aştı');
- end if;
- End;

If...Then...Else

- Koşula bağlı işlemlerde koşul gerçekleştiğinde yapılması gereken işlemler olduğu gibi koşul gerçekleşmediğinde de yapılması gereken işlemler için kullanılır.

- Declare
- degbul number(2);
- Begin
-
- select count(*) into degbul from ogrenci;
- if degbul>20 then
- dbms_output.put_line('Öğrenci sayısı 20 yi aştı');
- else
- dbms_output.put_line('sınıf mevcudu :'| | degbul);
- end if;
- End;

Case

- Birden fazla koşul gerektiren işlemlerde karşılaştırmayı kolaylaştırmak için kullanılır. Oracle 9i ile birlikte kullanılmaya başlanmıştır.
- Sorgulamalar sırasında sql ifadelerinde decode komutu yerine daha gelişmiş koşul sorgulamaları yapılmasına olanak sağlar.
- Case yapısı ile birlikte geçerli bir sql ifadesi de kullanılabilir. Örneğin SELECT,UPDATE,DELETE gibi ifadeler kullanılabilir.

Case örnek

- Declare
- degal number(2):=°al;
- Begin
-
- case
- when degal>=90 then
- Dbms_output.put_line('Harf karşılığı :AA');
- when degal>=85 and degal<90 then
- Dbms_output.put_line('Harf karşılığı :BA');
- when degal>=60 and degal<70 then
- Dbms_output.put_line('Harf karşılığı :CC');
- when degal>=55 and degal<60 then
- Dbms_output.put_line('Harf karşılığı :DC');
- else
- Dbms_output.put_line('Harf karşılığı :FF');
- end case;
-
- End;

- select case
 - when count(*)<5 then 'sınıf 5kişiden az'
 - when count(*)>5 and count(*)<10 then 'sınıf 5 - 10 arası'
 - else
 - '10 dan çok'
 - end
- from ogrenci

Loop

- Belirli bir koşul gerçekleşene kadar tekrar etmesi istenilen işlemler için kullanılır. Eğer koşul gerçekleşirse döngüden çıkılır. Birden fazla loop döngüsü iç içe kullanılabilir.

Loop

İfadeler

End Loop

Exit yapısı

- If yapısı ile koşulun gerçekleşip gerçekleşmediği kontrol edilir. Koşul gerçekleşince exit ifadesi ile koşul sonlandırılır.

Loop

İfade 1

If koşul then

Exit;

End if;

End Loop

- Declare
- Say number(2):=0;
- Begin
- loop
- say:=say+1;
- Dbms_Output.put_line('sayac : ' || say);
- if say=5 then
- exit;
- end if;
- end loop;
- End;

Exit when

- Belirtilen koşul gerçekleştiğinde döngü sonlandırılır.
- Declare
- Say number(2):=0;
- degal number(2):=&DeGal1;
- Begin
- loop
- say:=say+1;
- Dbms_Output.put_line('sayac :'| |say);
- Exit when say=degal;
- end loop;
- End;

While Döngüsü

- Bir kodun yada kod bloğunu koşul gerçekleştiği sürece devam etmesini sağlar.

While koşul ifadesi LOOP

Tekrarlanması istenen işlemler

End loop

While örnek kod

- Declare
- Tut number(2):=°er_al;
- Sayac number:=1;
- deg number:=1;
- Begin
- While sayac<tut loop
- deg:=deg*sayac;
- Dbms_output.put_line('sayac' || deg);
- Sayac:=sayac+1;
- End loop;
- End;

For döngüsü

- Diğer döngüler gibi sayacın artırılması için bir kod yazılması gerekmez. Sayısal olarak artırımları otomatik kendisi yapar.
- Begin
- For sayac in 1..10 loop
- Dbms_output.put_line(sayac);
- End loop;
- End;

For örnek

- Declare
- Bit number:=&bitiş_deger;
- Sonuc number:=0;
- Begin
- For sayac in reverse 1..bit loop
- Dbms_output.put_line(sayac);
- End loop
- End;

Tablo for örneği

- Begin
- For sayac in (select * from ogrenci) loop
- if sayac.ogrno='1' then
- dbms_output.put_line('Sayın : ' || sayac.adsoyad || ' Harç borcunuzu ödeyin');
- else
- *--insert into derskaydi values (sayac.ogrno,'1213101',2017,-1,-1,-1);*
- dbms_output.put_line('Sayın : ' || sayac.adsoyad);
- End if;
- End loop;
- End;

Döngü Denetimleri

- Döngü tamamlanmadan kullanıcı tarafından bir koşul gerçekleşirse döngüyü sonlandırmak için LOOP döngüsünde olduğu gibi EXIT ifadesi kullanılır. EXIT ifadesi if yapısı yada EXIT WHEN yapısı ile kullanılabilir.
- Döngünün belirli bir adımını atlamak için CONTINUE veya GOTO ifadeleri kullanılır.

- begin
- for sayac in 1..10 loop
- dbms_output.put_line(sayac);
- *--if sayac=5 then*
- *--exit;*
- *--end if;*
- exit when sayac=5;
- end loop;
- end ;

- Döngüler oluşturulurken etiketler verilebilir. İç içe döngüler kullanılırken döngü etiketleri tercih edilmelidir.
- <<etiket1>>
- LOOP
- <<etiket2>>
- LOOP
- EXIT etiket1 When
- EXIT etiket2 When
- EXIT When
- end LOOP etiket2;
- Exit when
- End loop etiket1;

- begin
- <<satir>>
- for sayac in 1..10 loop
- <<sutun>>
- for sayac2 in 1..10 loop
- exit satir when sayac=5;
- dbms_output.put_line('Sayac1:' || sayac || 'sayac2:' || sayac2);
- end loop sutun;
- end loop satir;
- end ;

Continue örneği

- Declare
- sayac number:=0;
- begin
- loop
- dbms_output.put_line('yazım1:' || sayac);
- sayac:=sayac+1;
- if sayac<3 then
- continue;
- end if;
- exit when sayac=5;
- dbms_output.put_line('yazım2:' || sayac);
- end loop;
- end;

Etikete goto

- Declare
- sayac number:=0;
- begin
- loop
- dbms_output.put_line('yazım1:' || sayac);
- <<atla>>
- sayac:=sayac+1;
- if sayac<3 then
- goto atla;
- end if;
- if sayac=4 then
- null;
- else
- dbms_output.put_line('merhaba');
- end if;
- exit when sayac=5;
- dbms_output.put_line('yazım2:' || sayac);
- end loop;
- --null;
- end;

Exception

- When kuraldışı1 then
- gerçekleşecek işlem
- When kuraldışı2 then
- gerçekleşecek işlem
- When others then
- gerçekleşecek işlem

Exception

- `No_data_found` :select into ifadesinde herhangi bir sonuç dönmediğinde gerçekleşen durumda devreye girer.
- `Too_many_rows`:select into ifadesinde birden fazla sonuç dönerse oluşan hata durumudur.
- `Zero_divide` :Kodda sıfıra bölme hatası oluşması sırasında gerçekleşir.
- `Program_error`: Pl/Sql programı içerisinde dahili bir hata meydana geldiğinde gerçekleşir.
- `Value_error`: Geçersiz bir tip dönüşümünde veya değişken tipinin boyut uyumsuzluğunda gerçekleşir.

Declare

ogrencino varchar(20):=&ogrencino;

bilgi ogrenci%rowtype;

Begin

Select * into bilgi from ogrenci f

Where ogrno=&ogrencino;

Dbms_output.put_line('Adsoyad :'| | bilgi.adsoyad);

Exception

When No_Data_found then

Dbms_output.put_line('!! Öğrenci bulunamamıştır !!');

End;

- Declare
- ogrencino varchar(20):=&ogrencino;
- adi varchar(20);
- bilgi ogrenci%rowtype;
- Begin
- Select f.adisoyad into adi from ogrenci f
- Where ogrno=&ogrencino;
- Dbms_output.put_line('Adsoyad :'| |adi);
- Exception
- When No_Data_found then
- Dbms_output.put_line('!! Öğrenci bulunamamıştır !!');
- Dbms_output.put_line('!! Öğrenci bulunamamıştır 2222!!');
- insert into hatalog values('veri bulunamadı :'| |ogrencino);
- commit;
- End;

- Declare
- `ogrencino varchar(20):=&ogrencino;`
- `bilgi ogrenci%rowtype;`
- Begin
- `Select * into bilgi from ogrenci f;`
- `-- Where ogrno=&ogrencino;`
- `Dbms_output.put_line('Adsoyad :'| | bilgi.adsoyad);`
- Exception
- When too_many_rows then
- `Dbms_output.put_line('!! Birden fazla kayıt vardır!!');`
- When No_data_found then
- `Dbms_output.put_line('!! Kayıt Bulunamadı!!');`
- End;

- Declare
- deger1 number:=°er_A1;
- deger2 number:=°er_A2;
- sonuc number:=0;
- Begin
- Sonuc:=deger1/deger2;
- Dbms_Output.put_line(sonuc);
- exception
- when zero_divide then
- Dbms_Output.put_line('sıfıra bölme hatası');
- end;

- Declare
- deger1 number(4,2);
- sonuc number:=0;
- Begin
- deger1:=°er_A11;
- Dbms_Output.put_line(deger1);
- exception
- when value_error then
- Dbms_Output.put_line('Değer hatası');
- end;

- Dup_val_on_index : Unique olarak tanımlanan bir sütuna sütun içerisinde var olan bir değer girilmesiyle oluşan hata...
- Begin
- insert into ogrenci values('1','FFF GGG',35);
- Exception
- when dup_val_on_index then
- Dbms_Output.put_line('Eklenmek istenen numara önceden eklenmiş');
- end;

Kullanıcı tanımlı exception

- Program içerisinde bazı kural dışı durumlar için önceden tanımlı exception lar bulunmayabilir.
- Örneğin veri tabanının bir sütununu not girişi olarak number tipinde tanımladık. Bu sütuna negatif bir değer girildiğinde hata mesajı vermesini istiyorsak kendi exceptionlarımızı tanımlamamız gerekiyor.

- Declare
- Neg_deger exception;
- deg number:=°er_al;
- Begin
- if deg<0 then
- raise Neg_deger;
- else
- Dbms_Output.put_line('Değer negatif değildir... :');
- end if;
- Exception
- when neg_deger then
- Dbms_Output.put_line('Negatif değer giremezsiniz');
- When others then
- Dbms_Output.put_line('Negatif değer giremezsiniz');
- end;

- Declare
- frg_hat exception;
- Pragma exception_init(frg_hat,-2292);
- Begin
- Delete ogrenci where ogrno=1;
- Exception
- when frg_hat then
 - Dbms_Output.put_line('alt kategorileri önceden silmeniz gerekiyor...');
- end;

Raise_application_error

- Özel bir yordamdır.
- Oracle benzer hata mesajları oluşturmak için kullanılır.

- Declare
- deg number:=°er_al;
- Begin
- if deg<0 then
- raise_application_error(-20000,'Negatif değer girilemez.');
- else
- Dbms_Output.put_line('Değer negatif değildir... :');
- end if;
- end;

- Declare
- `deg varchar(10):=°_al;`
- `tut ogrenci%rowtype;`
- begin
- `select * into tut from ogrenci where ogrno=deg;`
- `dbms_output.put_line(tut.adsoyad);`
- Exception
- when No_data_found then
- `raise_application_error(-20001,'kayıt bulunamamıştır');`
- end;

Collection tipler

- Veri kümelerini bellekte işleyebilmek için yapılar oluşturmaya ihtiyaç vardır.
- Düzenli olarak gruplandırılmış veri kümelerini içerir.
- Dizi yapısına benzer yapılardır.
- Her eleman benzersiz bir index değerine sahiptir.

3 farklı collection vardır.

Associative Arrays

Nested Tables

Varrays

Associative Arrays

- Collection içerisinde tutulan değerlerin sayısal veya karakter türünde indeks değerleri olabilir.
- Type tip_adi is table of veritipi
- Index by [binary_integer | Pls_integer | varchar2(n)]
- Table of dan sonra collection da tutulacak verinin, veri tipini ; index by ifadesinden sonrada collection için kullanılacak index bilgilerinin veri tipi belirlenir.

Associative Arrays

- Declare
- Type ilkod is table of varchar2(20)
- index by varchar2(20);
- il ilkod;
- begin
- il('ADANA'):= '01';
- il('izmir'):= '35';
- il('KONYA'):= '42';
- dbms_output.put_line(il('KONYA'));
- end;

Collectionlar için hazır fonksiyonlar:

- Count: Eleman sayısını verir.
- Delete: Bir veya daha fazla elemanı siler.
- Exists(n): Bir elemanın bulunup bulunmadığını kontrol eder.
- First: İlk elemanın indexini döndürür.
- Prior: Bulunulan noktanın öncesinin index değerini döndürür.
- Next: Bulunulan noktanın sonrasının index değerini döndürür.
- Extend(n): Yeni eklenecek elemanlara yer açmak için kullanılır.

Nested Tables

- Tek boyutlu dizilere benzeyen, tutacağı eleman sayısı dinamik olarak değişir.

Nested Tables

- Declare
- *--Nested tablo oluşturuluyor.*
- Type aylar_tipi is table of varchar2(20);
- *-- nested e değişken oluşturuyoruz.*
- ay_deg aylar_tipi:=aylar_tipi();
- say number;
- begin
-
- ay_deg.extend(3); *-- collectionda 3 elemanlık yer açıldı.*
- ay_deg(1):='OCAK';
- ay_deg(2):='ŞUBAT';
- ay_deg(3):='MART';
-
- ay_deg.delete(2);
-
- Dbms_Output.put_line('Eleman Sayısı:' || ay_deg.count);
- Dbms_Output.put_line('Eleman Sayısı:' || ay_deg.last);
 say:=ay_deg.first;
- Dbms_Output.put_line('İlk Eleman:' || ay_deg(say));
- end;

Varrays

- Tutulacak elaman sayısı sabit olarak tanımlanan collection türüdür. Bu collectionda ekleme ve silme işlemleri sadece sonda yada başta yapılır.

- Type `tip_adi` is `varray (boyut) of veri_tipi;`

Veya

- Create or replace type `tip_adi` is `array(boyut) of veri_tipi;`

Varrays

- Declare
- Type aylar_tipi is varray(13) of varchar2(20);
- ay aylar_tipi;
- begin
- ay:=aylar_tipi('OCAK','ŞUBAT','MART','NİSAN','MAYIS','HAZİRAN','TEMMUZ','AĞUSTOS','EYLÜL','EKİM','KASIM','ARALIK');
- Dbms_Output.put_line('Eleman Sayısı:' || ay.count);
- ay.extend;
- ay(ay.last):='NULL';
- Dbms_Output.put_line('Eleman Sayısı:' || ay.count);
- Dbms_Output.put_line('2. ay:' || ay(2));
- end;

- *--bir kere oluşturuluyor :*
- *--create or replace type dizi is varray(12) of varchar2(20);*
- Declare
- urunler dizi:=dizi();
- Begin
- urunler.extend(4);
- urunler(1):='AAAA';
- urunler(2):='BBBB';
- urunler(3):='CCCC';
- urunler(4):='DDDD';
- Dbms_Output.put_line('3.urun:' || urunler(3));
- end;

- Declare
- type ogr is record (dogrno ogrnci.ogrno%type, dadsoy ogrnci.adsoyad%type, dfak ogrnci.fakultesi%type);
- type std is table of ogr index by binary_integer;
- degogr std;
- begin
- select * bulk collect into degogr from ogrnci;
- For i in 1..degogr.count loop
- dbms_output.put_line(degogr(i).dogrno || ' ' || degogr(i).dadsoy);
- end loop;
- end;

Cursor

- Word gibi belge yazım programlarında imleç nerde ise bilgi belgenin o bölümüne yazılır. Veri tabanlarında da imleç hangi satırda ise o satır üzerinde işlem yapılır.
- İmleçler sunucuya yük getirdiğinden gerekli yerlerde kullanılmalıdır.
- Select ifadesi ile kullanılır.
- İki tür imleç kullanılır.

İmplicit cursors (Kapalı imleçler)

- Sql ifadesi çalıştırıldığında bu cursor otomatik olarak oluşur.
- Kullanıcı müdahale edemez. Open,fetch ve close işlemleri otomatik olarak yapılır.

implicit cursors (Kapalı imleçler)

- Declare
- --s number;
- begin
- --s:=5/0;
- update ogrenci
- set fakulte=30
- where ogrno<-5;
- dbms_output.put_line(sql%rowcount);
- exception
- when others then
- dbms_output.put_line(sqlcode || sqlerrm);
- end;

Explicit cursor (Açık imleçler)

- İmleç bir select ifadesi ile tanımlanır.
 - Oluşturulan imleç ile veriler üzerinde gezinmek için imleç açılır.
 - Veri kümesinin son satırına gelinceye kadar tüm veriler taranır.
 - Veri kümesi üzerinde yapılan işlemler bittiğinde veri kümesi kapatılır.
-
- Cursor imlecadi is select ifadesi
 - Open imlecadi
 - Fetch imlecadi into pl/sql değişkenleri
 - Close imlecadi

Explicit cursor (Açık imleçler)

- Declare
- cursor ogr_imlec is select * from ogrenci;
- ogr_veri ogr_imlec%rowtype;
- begin
- open ogr_imlec;
- loop
- fetch ogr_imlec into ogr_veri;
- exit when ogr_imlec%notfound;
- dbms_output.put_line(ogr_veri.ogrno || ogr_veri.adisoyad);
- if ogr_veri.ogrno=3 then
- insert into derskaydi values(ogr_veri.ogrno,'1213103',-1,-1,-1);
- end if;
- end loop;
- close ogr_imlec;
- end;

imleç2

- Declare
- cursor ogr_imlec is select * from ogrenci;
- ogr_veri ogr_imlec%rowtype;
- der varchar2(50);
- begin
- open ogr_imlec;
- loop
- fetch ogr_imlec into ogr_veri;
- exit when ogr_imlec%notfound;
- dbms_output.put_line(ogr_veri.ogrno || ogr_veri.adisoyad);
- *-- if ogr_veri.ogrno=16 then*
- select max(dersno) into der from ders where dersno not in (select dersno from derskaydi);
- insert into derskaydi values(ogr_veri.ogrno,der,-1,-1,-1);
- insert into harc values(ogr_veri.ogrno,6,50);
- *-- end if;*
- end loop;
- close ogr_imlec;
- commit;
- end;

Explicit cursor (Açık imleçler)

- Declare
- `cursor ogr_imlec is select * from ogrenci;`
- begin
- For i in ogr_imlec loop
- `dbms_output.put_line(i.ogrno || i.adsoyad);`
- end loop;
- end;

İmleçlerde parametre kullanımı

- Declare
- cursor ogr_imlec (ogr in ogrenci.ogrno%TYPE) is select * from ogrenci where ogrno=ogr;
- begin
- For i in ogr_imlec(1) loop
- dbms_output.put_line(i.ogrno || i.adisoyad);
- end loop;
- end;

Procedures

- Veri tabanında tutulan yapılardır.
- Çalıştığı anda derlenen yapılardır.
- Sql yapılarına göre daha performanslı çalışır. Syntax,derleme kontrolleri oluşturulduğu sırada yapıldığı için hızlı çalışır.
- Ağ trafiğini azaltır.
- Parametrelili olarak çalıştığı için güvenliği artırır. Sql injection ları engeller.
- Grup çalışmasına yatkınlık

Procedures

- Create Or Replace Procedure yordam_ismi
- (Paremetre,paremetre,...) as
- Yerel tanımlar
- Begin
- Gerçekleşecek işlemler;
- Exception
- Kural dışı durumların kontrolü
- End yordam_ismi;

Procedure örnek

- create or replace procedure testproc as
- Cursor tut is select a.ogrno, a.adsoyad,b.dersno,b.not1 from ogrenci a,notlar b
- where a.ogrno=b.ogrno(+);
- Begin
- for imlec in tut loop
- dbms_output.put_line(imlec.ogrno||imlec.adsoyad);
- end loop;
- End testproc;

Procedure çalıştırılması

- begin
- testproc();
- end;

Procedure parametre kullanımı

- create or replace procedure testprocpar(ogrencino in varchar2, adsoy out varchar2,gn out number) as
- Begin
- *--sele a.adsoyad,b.not1 into adsoy,gn from ogrenci a,notlar b*
- select a.adsoyad,b.not1 into adsoy,gn from ogrenci a,notlar b
- where a.ogrno=b.ogrno(+) and a.ogrno=ogrencino;

- Exception
- when no_data_found then
- dbms_output.put_line ('Belirtilen kitap bulunamadi');

- End testprocpar;

Procedure Çağırılması

- Declare
- dogrno varchar2(15);
- dadsoy varchar2(15);
- dnot number;
- begin
- dogrno:=&dogrno;
- testprocpa(dogrno,dadsoy,dnot);
- dbms_output.put_line ('ogrenci adı:' || dadsoy);
- dbms_output.put_line ('ogrenci not' || dnot);
- end;

- create or replace procedure uyekitapbul(uyno in varchar2, kitapad out varchar2, kitapyaz out varchar) as
- Begin
- select a.kitadi,a.yazar into kitapad,kitapyaz from kitap a, odunc b where a.kitapno=b.kitapno and
- b.uyeno=uyno;
- insert into log values (uyno,'uyekitapbul',sysdate);
- commit;
- Exception
- when no_data_found then
- dbms_output.put_line ('Belirtilen kitap bulunamadı');
- End uyekitapbul;

Yordam Silmek

- Hazırlanmış olan yordamları Drop ifadesi ile sileriz.
- DROP PROCEDURE YORDAM_ISMI

FONKSİYONLAR

- Foksiyonlar, Procedure(yordam) ler gibi çalışır. Tek farkı dışarıya bir değer döndürmesidir.
- Paremetre olarak in,out,inout parametreleri alabilirler. Ama o zaman fonksiyon olma özünü kaybederler.

FONKSİYONLAR

- Foksiyonlar dışarıdan değer alabilmek için giriş parametrelerini kullanırlar.

Create or replace function foksiyon_ismi

(parametreler)

Return veri_tipi As/is

Yerel tanımlar

Begin

Fonksiyon gerçekleştirme işlemleri

Return (değer);

End;

Fonksiyon örnek

- create or replace function buyukharf (gelen in varchar2) return varchar2
- is
- Tut varchar2(20);
- Begin
- Tut:=replace(gelen,'i','İ');
- Return upper(tut);
- End;

Fonksiyon örnek

- create or replace function Dersadibul (dersN in varchar2) return varchar2
- is
- Tut varchar2(40);
- Begin
- Select dersadi into tut from ders.ders a where a.dersno=DersN;
- Return (tut);
- Exception
- when no_data_found then
- Return 'bilinmiyor';
- End;

- create or replace function ogradibul (OgrN in varchar2) return varchar2
- is
- Tut varchar2(40);
- Begin
- select a.adisoyad into tut from ogrenci a where ogrno=OGRN;
- return buyukharf(tut);
- Exception
- when no_data_found then
- Return 'bilinmiyor';
- End;

Tablo sonuçlu fonksiyonlar

- Tablo sonuçlu fonksiyonlar geriye tek değer döndürmek yerine birden fazla sütun ve satırdan oluşan tablo değerleri döndürebilmektedir.
- Return ifadesi ile number,int,varchar2 gibi ifadeler döndürülebilir. Döndürülecek değer tablo ise

Tablo sonuçlu Fonksiyon (type tanım)

- create or replace Type derslertut as object
- (
 - DERSNO VARCHAR2(8),
 - YIL NUMBER,
 - ADI VARCHAR2(50),
 - KREDI NUMBER,
 - ECTS NUMBER
-);
- CREATE OR REPLACE Type derslertutsatir as table of derslertut;

- create or replace function Derslerfunc(fakno in varchar2) return derslertutsatir
- is
- sonuc derslertutsatir :=new derslertutsatir();

- Begin
- For satir in (select * from ders.dersler where substr(dersno,1,2)=Fakno)
- loop
- sonuc.extend;
- sonuc(sonuc.count):=new derslertut{
- satir.dersno,satir.yil,satir.adi,satir.kredi,satir.ects
- };
- end loop;
- return sonuc;

- End;

- Çağırılması:

select * from table(derslerfunc('12'));

- Create or replace type DRS as table of varchar2(70);

create or replace function Derslerism(fakno in varchar2) return Drs
pipelined

is

Begin

For satir in (select adi from ders.dersler where substr(dersno,1,2)=Fakno)

loop

PIPE ROW(satir.adi);

end loop;

return;

End;

select * from table(derslerism('12'))

Trigger

- create or replace trigger silme_yok
- before drop on database-- *schema system ile açılmadıysa*
- begin
- raise_application_error(-20000,'Tabloyu silme işlemi yasağı');
- end;

Veri tabanından çekilebilecek loglar

- select
- SYSDATE
- ,SYS_CONTEXT('USERENV','TERMINAL') TERMINAL
- ,SYS_CONTEXT('USERENV','SESSIONID') SESSIONID
- ,SYS_CONTEXT('USERENV','INSTANCE') INSTANCE
- ,SYS_CONTEXT('USERENV','ENTRYID') ENTRYID
- ,SYS_CONTEXT('USERENV','ISDBA') ISDBA
- ,SYS_CONTEXT('USERENV','CURRENT_USER') CURRENT_USER
- ,SYS_CONTEXT('USERENV','CURRENT_USERID') CURRENT_USERID
- ,SYS_CONTEXT('USERENV','SESSION_USER') SESSION_USER
- ,SYS_CONTEXT('USERENV','SESSION_USERID') SESSION_USERID
- ,SYS_CONTEXT('USERENV','PROXY_USER') PROXY_USER
- ,SYS_CONTEXT('USERENV','PROXY_USERID') PROXY_USERID
- ,SYS_CONTEXT('USERENV','DB_NAME') DB_NAME
- ,SYS_CONTEXT('USERENV','HOST') HOST
- ,SYS_CONTEXT('USERENV','OS_USER') OS_USER
- ,SYS_CONTEXT('USERENV','EXTERNAL_NAME') EXTERNAL_NAME
- ,SYS_CONTEXT('USERENV','IP_ADDRESS') IP_ADDRESS
- ,SYS_CONTEXT('USERENV','NETWORK_PROTOCOL') NETWORK_PROTOCOL
- ,SYS_CONTEXT('USERENV','AUTHENTICATION_TYPE') AUTHENTICATION_TYPE
- from dual

Log table

- Create table veritablog1 (
 - id number primary key,
 - yapan varchar2(40),
 - Nesne varchar2(40),
 - tarih date
 -)
- create sequence log_say;

- create or replace trigger silme_yok1
- before drop on database-- *schema system ile açılmadıysa*
- begin
- insert into veritablog
values(log_say.nextval,ora_dict_obj_owner,ora_dict_obj_name,sysdate);
- end;

- Create table oturumlog (
- yapan varchar2(40),
- islem varchar2(40),
- tarih date
-)

- create or replace trigger oturumac
- after logon on database-- *schema system ile açılmadıysa*
- begin
- insert into oturumlog values(ora_login_user,'Bağlandı',sysdate);
- end;

- Create table notlog10 (
- ogrno varchar2(40),
- dersno varchar2(40),
- Vize int,
- Final1 int,
- islem varchar2(40),
- tarih date,
- kimyapti varchar2(50),
- ipadresi varchar2(50)
-)

- create or replace trigger dersekleme6
- before insert on derskaydi-- *schema system ile açılmadıysa*
- For each row
- begin
- insert into notlog
values(:new.ogrno,:new.dersno,:new.vize,:new.final,
- 'ekleme',sysdate,ora_login_user,SYS_CONTEXT('USERENV','IP_ADDRESSES'));
- end;

- create or replace trigger derssilme
- before delete on derskaydi-- *schema system ile açılmadıysa*
- For each row
- begin
- insert into notlog values(:old.ogrno,:old.dersno,:old.vize,:old.final,
- 'silme',sysdate,ora_login_user);
- end;

- Create table dersnotlog (
- ogrno varchar2(40),
- dersno varchar2(40),
- islem varchar2(40),
- eski int,
- yeni int,
- tarih date
-)

- create or replace trigger dersduzeltme
- before update of vize,final on derskaydi-- *schema system ile açılmadıysa*
- For each row
- begin
- if :old.vize<>:new.vize then
- insert into dersnotlog values(:new.ogrno,:new.dersno,'vize',:old.vize,:new.vize,sysdate);
- end if;
- if :old.vize<>:new.vize then
- insert into dersnotlog values(:new.ogrno,:new.dersno,'vize',:old.vize,:new.vize,sysdate);
- end if;
- :new.gn :=((:new.vize*0.4)+(:new.final*0.6));
- end;

Package

- İçinde çok kullanılan function ve Procedure ler için kullanılır.
- İkiye ayrılır.
- Specification : Çağrıldığında public tanımların görüneceği bloktur.
- Private metodlar, yardımcı metodlar olarak body de kullanılır .
- Body : Specification da tanımlanan herşeyin execute edildiği alandır. Çağrıldığında Specificationdan tanımları alır. İşleteceği kodları body bloğunun içinden işletir. Bulamazsa hata verir.

- create or replace Package Packed _name
- As
- Procedure P_name()....
- Function P_name() return Varchar2();
- End;
- create or replace Package Body Packed _name
- As
- Procedure P_name()....
- As
- Begin
- End;
- Function P_name() return Varchar2();

- create or replace package deneme
- as
- procedure testprochesapla;
- function Dersadibul (dersN in varchar2) return varchar2;
- end;

- create or replace package body deneme
 - as
 - procedure testprochesapla as
 - Begin
 - update derskaydi
 - set gn=(vize*0.4)+(final*0.6)
 - where gn is null;
 - commit;
 - End;
-
- function Dersadibul (dersN in varchar2) return varchar2
 - is
 - Tut varchar2(40);
 - Begin
 - Select dersadi into tut from ders.ders a where a.dersno=DersN;
 - Return (tut);
 - Exception
 - when no_data_found then
 - Return 'bilinmiyor';
 - End;
 - end;

- Pl/Sql ile geliştirdiğimiz programlarda ihtiyacımıza göre bazı işlemlerin otomatik veya belirli zamanlarda tekrarlı olarak yapılmasını isteriz.
- Bu tarz gereksinimlerimiz için Oracle tarafından bize sağlanan **dbms_job** paketini kullanıp sistemde zamanlanmış görevler oluşturabiliriz.

JOB

- declare
- vJob number;
- begin
- dbms_job.submit(job => vJob,
- what => '
- begin
- testprochesaplajob();
- end;
- ',
- next_date => to_date('14/12/2022', 'dd/mm/yyyy'),
- interval => 'SYSDATE + 30/86400 ');
- commit;
- dbms_output.put_line('Oluşan JobId:' || vJob);
- end;

- declare
- vJob number;
- begin
- dbms_job.submit(job => vJob,
- what => '
- begin
- insert into denemeTable
- (Tarih, Aciklama)
- values
- (sysdate, 'Job ile bu tabloya insert edildi.');
- commit;
- end;
- ',
- next_date => to_date('12/05/2020', 'dd/mm/yyyy'),
- interval => 'TRUNC(SYSDATE+1)+(1/24)');
- commit;
- dbms_output.put_line('Oluşan JobId:' || vJob);
- end;

- select * from user_jobs;

- Her gün çalışsın: 'SYSDATE + 1'
- Haftada bir çalışsın: 'SYSDATE + 7'
- Her saat başında çalışsın: 'SYSDATE + 1/24'
- 10 dakikada bir çalışsın: 'SYSDATE + 10/1440'
- 30 saniyede: 'SYSDATE + 30/86400'
- Bir daha çalışmasın: 'NULL'
- **DBMS_JOB.WHAT:** Job'ın içeriğini değiştirebilen prosedür.

MYSQL

- DELIMITER \$\$
- DROP PROCEDURE IF EXISTS WhileLoopProc\$\$
- CREATE PROCEDURE WhileLoopProc()
- BEGIN
- DECLARE x INT;
- DECLARE str VARCHAR(255);
- SET x = 1;
- SET str = '';
- WHILE x <= 5 DO
- SET str = CONCAT(str, x, ',');
- SET x = x + 1;
- END WHILE;
- SELECT str;
- END\$\$
- DELIMITER ;

Saklı yordamdaki ilk satır olan DELIMITER // aslında saklı yordama ilişkin bir ifade değildir.

DELIMITER komutu mysql'in standart ayırıcı olan noktalı virgülü (;) değiştirmeye yarar.

Bu sayede saklı yordam içerisinde birden fazla sql komutu yazıp

bunları noktalı virgül(;) ile ayırabilir. Saklı yordamı yaratmak için CREATE PROCEDURE direktifi kullanılır. Bu direktifi saklı yordamın ismi takip eder. İsimlendirmedeki kabul görmüş genel

notasyon camel(deve) notasyonudur. (Her bir kelimenin ilk harfi

büyük olacak şekilde.) Yukarıdaki örnekte de görüldüğü gibi "GetAllClients"

. Saklı yordamın gövdesi BEGIN ve END bloğu ile belirtilir.

BEGIN ve END arası saklı yordamın local (yerel) bilinirlik alanına ilişkindir. Yani BEGIN ve END arasına yazılan her sql komutu saklı yordama dahildir.