

See discussions, stats, and author profiles for this publication at: <https://www.researchgate.net/publication/320531402>

Oracle Veri Tabanı Yönetim Sistemi & PL/SQL

Chapter · September 2013

CITATIONS

0

READS

39,933

1 author:



Deniz Mertkan Gezin

Trakya University

97 PUBLICATIONS 1,292 CITATIONS

SEE PROFILE

ORACLE Veri Tabanı Yönetim Sistemi

&

PL/SQL

Yrd. Doç. Dr. Deniz Mertkan GEZGİN

"Güzel Ablam Pınar'a ithafen"

İçindekiler

ORACLE Veri Tabanı Yönetim Sistemi	4
ORACLE Sürümleri	5
ORACLE VTYS Yapısı	6
Fiziksel Bölüm	8
Veri Dosyaları(Data Files)	8
Kontrol Dosyaları (Control Files)	8
Log Dosyaları (Log Files)	8
Mantıksal Bölüm	9
Veri Blokları (Data Blocks).....	10
Genişlemeler (Extends)	10
Parçalar (Segments).....	10
ORACLE Veritabanı Hafıza Yapısı	10
ORACLE Database 10g Express Kurulumu	12
ORACLE Veritabanı Yönetim Araçları	16
ORACLE Sql Developer 3.2.2 Kurulumu.....	18
ORACLE Database 10g Üzerinde Kullanıcı Oluşturma	22
ORACLE Database 10g Üzerinde Örnek Veri Tabanı.....	25
Örnek Veritabanının SQL Kodları	26
DDL (Data Definition Language) Kodları	26
DML (Data Manipulation Language) Kodları	27
Örnek Veritabanının İlişkisel Diyagramı	28
SQL Yapısal Sorgulama Dili.....	29
Veri Tanımlama Dili (DDL- Data Definition Language).....	29
Veri İşleme Dili (DML - Data Manipulation Language)	29
İşlem Kontrol Dili (TCL-Transaction Control Language)	30
Veri Kontrol Dili (DCL-Data Control Language).....	31
ORACLE’ da Bütünlük Kısıtları	31
PL/SQL.....	32
PL/SQL’de Veri Tipleri.....	33
PL/SQL Sayısal Veri Tipleri	34
PL/SQL’de Operatörler	37
PL/SQL’de Atama İşlemleri.....	38

PL/SQL'DE Değişkenler Ve Sabitler.....	38
PL/SQL'de Blok Yapısı	39
PL/SQL'de Sıra Yapısı (Sequence).....	41
PL/SQL'de Hata İşleme(Error Handling)	41
PL/SQL'de Akış Kontrol Yapıları.....	44
IF –THEN Yapısı	45
IF - THEN – ELSE Yapısı.....	46
IF - THEN – ELSEIF Yapısı	46
CASE Yapısı	47
LOOP Yapısı	48
EXIT Yapısı	48
EXIT- WHEN Yapısı	49
WHILE – LOOP Yapısı	50
FOR – LOOP Yapısı	50
GOTO Yapısı	51
NULL Yapısı	51
PL/SQL'de İş Blokları(Transaction)	52
İndeksler(Indexes)	53
Prosedürler(Yordamlar).....	54
Fonksiyonlar(Functions)	55
Görünümler (Views).....	56
Tetikleyiciler(Triggers)	58
İmleçler(Cursors).....	60
Paketler(Packages)	62
Kullanıcılar ve Roller(Users and Roles).....	64
Bölüm Sonu Soruları	66
Kaynakça	67

İlişkisel Veritabanı Yönetim Sistemleri (Relational Database Management Systems - RDBMS) büyük miktarlardaki verilerin güvenli bir şekilde tutulabildiği, bilgilere hızlı erişim imkânlarının sağlandığı, bilgilerin bütünlük içerisinde tutulabildiği ve birden fazla kullanıcıya aynı anda bilgiye erişim imkânının sağlandığı programlardır. **Oracle** veritabanı da bir ilişkisel veri tabanı yönetim sistemidir. Oracle dışında kullanılan Veri Tabanı Yönetim Sistemleri (VTYS) DB2, MS SQL Server, Sybase, Informix, MySQL, FireBird Postrage gibi sıralanabilir. **Oracle** Veritabanı Dünyada ve Türkiye de büyük kurumsal uygulamalarda en çok tercih edilen veritabanlarından birisidir. Özellikle büyük ölçekli firmalar, bankalar için lider konumdadır.

Oracle veritabanının özellikleri şunlardır:

- Büyük miktarda veri tutabilmekte ve verilerin depolandığı alanları ayarlama imkânı vermektedir.(Sayfalama)
- Aynı anda çok sayıda kullanıcıya verilerin bütünlüğünü bozmadan hizmet verebilmektedir. (Transaction)
- Günler ve haftalar boyu hiç kapatılmadan çalışabilmektedir.
- İşletim sistemi, veri erişim dilleri ve ağ iletişim protokolleri standartlarıyla uyumludur.
- Yetkisiz erişimleri engelleme ve kullanıcı kontrolünü desteklemektedir.
- Bütünlüğü veritabanı düzeyinde sağlayabilmektedir, böylece daha az kod yazılmaktadır.
- İstemci/Sunucu mimarisinin bütün avantajlarını kullanabilmektedir. Ağ trafiğini azaltmaktadır.

Oracle ile ilk defa karşılaşan kullanıcılar genellikle sadece bir veritabanı ve SQL gibi yapısal sorgulama dili ile çalışan uygulamalarla karşılaşmayı umarlar. Oysaki yukarıda da belirttiğimiz gibi Oracle bir ilişkisel veritabanı yönetim sistemidir. Yani bir programlama dili ve sadece bir veri tabanı değildir. Fakat Oracle tarafından geliştirilen ve Oracle'ın kendi uygulama geliştirme araçları içerisinde kullanılan bir programlama dili vardır. Oracle ürünleri genellikle büyük çaplı veri kontrolünü gerektiren uygulamalarda kullanılır. Öncelikle bir ağda Oracle veritabanı sadece sunucu olarak adlandırılan bilgisayara yüklenir. Bu sunucu Oracle'ın desteklediği Unix, Linux ya da Windows işletim sistemiyle çalışıyor olabilir. Yani Oracle'ın

faklı işletim sistemleri için farklı sürümleri vardır. Bu sunucu bilgisayara kurulan veritabanı üzerinde tablolar, indeksler, eşanlıklar, tablo uzayları ve ihtiyaç duyulan kayıtlı prosedürler oluşturulur. Oracle'ın bu veritabanına erişerek uygulama programı geliştirmeye yarayan diğer ürünleri de istemci bilgisayarlara kurulur. Tabi sunucu bilgisayara da isteğe bağlı olarak bu ürünler kurulabilir. Fakat Oracle veri tabanı yönetimini, PL/SQL dilini öğrenmek ve işlem yapmak isteyen kullanıcılar için bu kitapta da anlatılacak ve üzerinde işlem yapılacak olan ORACLE 10g(grid) Express sürümü tercih edilmektedir.

ORACLE Sürümleri

Oracle Veritabanı hali hazırda her biri farklı gelişme senaryoları ve amaca uygun beş adet sürümden oluşmaktadır. Bunun yanında Oracle özellikli uygulama amaçları için kapsam genişlemesinde birkaç veritabanı opsiyonları, paketleri ve ürünler sunmaktadır.

Oracle Database Standard Edition One

Oracle Database Standard Edition One sürümü görülmemiş kolay kullanım, güç, çalışma grupları için performans, bölüm düzeyi ve Web uygulamaları kullanım sunar. Küçük işletmeler için tek sunucu ortamlardan yüksek dağıtımli şube ortamlarına, kritik iş uygulamaları oluşturmak için gerekli tüm olanakları sağlamaktadır.

Oracle Database Standard Edition

Oracle Database Standard Edition One sürümü görülmemiş kolay kullanım, güç, ve Oracle Database Standard Edition One sürümünün performansı ile birlikte büyük makineler için destek ve Oracle Real Application Clusters (Oracle RAC) kümeleme servisleri gibi özellikleri sunar. Oracle RAC, Oracle Database 10g'den önceki Standard Edition sürümlerinde mevcut değildir ya da sürümlerin özellikleri ile uyumlu değildir.

Oracle Database Enterprise Edition

Oracle Database Enterprise Edition, yüksek hacimli çevrimiçi işlem (OLTP) uygulamaları, sorgu-yoğun veri ambarları ve yoğun talepli İnternet uygulamaları gibi kritik uygulamalar için gerekli performans, kullanılabilirlik, ölçeklenebilirlik ve güvenliği sağlar. Oracle Database Enterprise Edition Oracle Database tüm bileşenleri içerir.

Oracle Database Express Edition

Oracle XE ya da Oracle Express Edition, Oracle veri tabanının giriş seviyesi sürümü olup özellikle geliştirme amaçlı olarak iyi bir veri tabanına ihtiyacı olan ve Oracle öğrenmek isteyen kişi ve kurumlar için temel niteliktedir. Oracle XE tamamen bedava, makinenizde kullanabilir hatta ticari olan uygulamalarınızla bile paketlenabilir. Fakat 4 GB'tan daha fazla veri tutmaması ve 1 GB'tan daha fazla RAM kullanamaması sürümün dezavantajlarından. Şu anki versiyonu 11g Windows ve Linux üzerinde 32 bit ve 64 bit olarak mevcut durumdadır.

Oracle Database Personal Edition

Oracle Database Personal Edition Oracle Database Standard Edition One, Oracle Database Standard Edition ve Oracle Database Enterprise Edition ile tam uyumluluk gerektiren tek kullanıcı geliştirme ve dağıtım ortamlarını destekler. Personal Edition yalnızca Windows platformlarında mevcuttur. Yönetim Paketleri Personal Edition içine dâhil değildir.

Son olarak anlatılan bölümde Oracle veritabanının sürümleri ele alınmıştır. Kitap daha çok Oracle Veri Tabanı Yönetimi değil PL/SQL dili üzerine olduğu için ve eğitim amaçlı olduğundan kitapta Oracle XE 10g sürümü seçilmiştir. Daha fazla sürüm bilgisi ve karşılaştırma için aşağıdaki internet adresine bakabilirsiniz.

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

ORACLE VTYS Yapısı

Oracle Veritabanı, küme olarak kabul edilen verilerin bir araya toplanmasından oluşur. Bir Veritabanının amacı birbiriyle ilişkili bilgilerin depolanması ve sorgulanmasıdır. Bir veritabanı sunucu (VYTS olarak da söyleyebiliriz), büyük miktardaki veriye çok kullanıcı bir ortamda, birçok kullanıcının aynı anda aynı veriye ulaşabilmelerini güvenilir bir şekilde sağlamakla görevlidir. Bunu aynı zamanda yüksek bir performansla gerçekleştirir. Bir veritabanı sunucusu aynı zamanda izinsiz erişimi engeller ve hata durumundan kurtulmak için gereken en uygun çözümleri sağlar. Oracle Veritabanı fiziksel(physical) ve mantıksal(logical) olmak üzere iki yapıdan oluşur. Fiziksel ve mantıksal yapı birbirinden ayrı olduğu için verinin fiziksel olarak saklanma şekli mantıksal yapıya erişimi etkilemez.

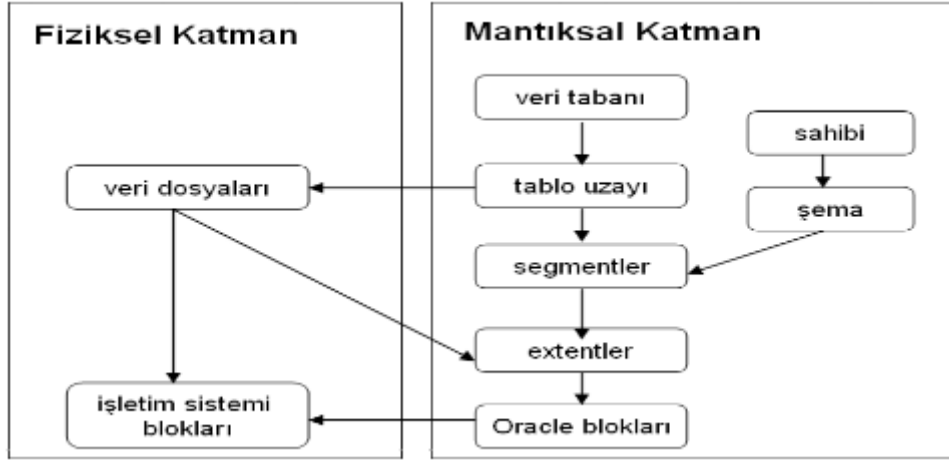
Fiziksel bölüm, işletim sistemi tarafından görülen aşağıdaki bölümlerden oluşmaktadır.

- Data File (Veri Dosyası)
- Control File (Kontrol Dosyası)
- Redo Log File (Log Dosyası)'dır.

Mantıksal Bölüm ise,

- Tablo uzayı (Tablespace)
- Tablolar (Table)
- Görüntüler (View)
- Sıralar (sequence),
- Eşanlamlar (synonym),
- İndeksler (index),
- Kümeler (cluster),
- Veritabanı bağlantıları (database link),
- Prosedürler (procedure),
- Fonksiyonlar (function),
- Paketlerden (package) oluşan şema nesnelerinden oluşmaktadır.

Veri tabanında her nesne bir kullanıcıya aittir. Bu nesneler kullanıcının sahip olduğu Tablo Uzaylarından(Tablespace) birinin içerisinden, fiziksel olarak da bir veri dosyası(data file) içinde tutulmaktadır. Şema() kavramı ise tablo, cluster, index, tetikleyiciler gibi mantıksal nesnelerin bulunduğu kısımdır.Bunun yanında Fiziksel bölüm işletim sistemi tarafından görülebilmesine rağmen, mantıksal bölüm ancak Oracle'a bağlanıp, SQL komutları çalıştırılarak görülebilmektedir. Yani, Oracle Kurulu herhangi bir makinede, SQL bilgisi olmayan bir insan, Oracle'ın sadece fiziksel bölümünü görebilmektedir.



Şekil - 1 Oracle Veritabanı yapısı

Fiziksel Bölüm

Fiziksel bölüm daha öncede ifade edildiği gibi veritabanını oluşturan işletim sistemi dosyalarıdır. Bir Oracle veritabanı fiziksel olarak bir ya da daha fazla veri dosyası, iki ya da daha fazla log dosyası, bir ya da daha fazla kontrol dosyasından oluşmaktadır.

Veri Dosyaları(Data Files)

Veri dosyaları veri tabanındaki tüm verileri tutan dosyalardır. Bir veri dosyası sadece bir veritabanına ait olabilir. Fakat bir veri tabanı sisteminde yüzlerce veri dosyası da olabilir. Tablo, indeks gibi mantıksal veritabanı yapılarının içerisindeki veriler fiziksel olarak veri dosyalarında tutulurlar. Bir veri dosyası kendisi için ayrılan alan dolduğunda, kendi sahip olduğu alanı artırabilecek özelliklere sahiptir. Bir ya da daha fazla veri dosyası mantıksal bir veritabanı depolama ünitesi olan bir tablo uzayını oluştururlar.

Kontrol Dosyaları (Control Files)

Tüm Oracle veritabanları kontrol dosyasına sahiptir. Bir kontrol dosyası veritabanı adı, veri dosyaları ve log dosyalarının adı ve diskteki yeri, veritabanının oluşturulma tarihi vb. veritabanı ile ilgili bilgileri tutar. Her veritabanı oturumu açıldığında Oracle bu dosyayı kontrol ederek gerekli bilgileri alır. Eğer veritabanında fiziksel bir değişim olursa(yeni bir log dosyası ya da veri dosyası oluşturulması gibi), yapılan değişiklikler Oracle tarafından otomatik olarak kontrol dosyalarına yansıtılır.

Log Dosyaları (Log Files)

Redo Log dosyaları olarak bilinen bu dosyaların amacı veriler üzerinde yapılan tüm değişiklikleri kaydetmektir. Eğer veri dosyalarına kalıcı olarak kaydedilmiş olan, değişikliğe uğramış kayıtlarda bir bozukluk olursa yapılan değişiklikler redo log dosyalarından sağlanabilir ve işlemler kaybolmaz. Birden fazla tekrarlanan bozukluk durumlarında redo log dosyalarının da bozulmasını engellemek için Oracle farklı diskler üzerinde redo log dosyalarının birden fazla kopyasının alınmasına olanak sağlar. Bir veritabanı işlemi sırasında elektrik kesilirse, bellekteki veriler veri dosyalarına kaydedilmeyecek ve verilerin kaybolması durumuyla karşılaşılacaktır. Oracle veritabanı tekrar açıldığında redo log dosyalarında yapılan son değişiklikler veri dosyalarına yansıtılarak verilerin kaybolması engellenir.

Mantıksal Bölüm

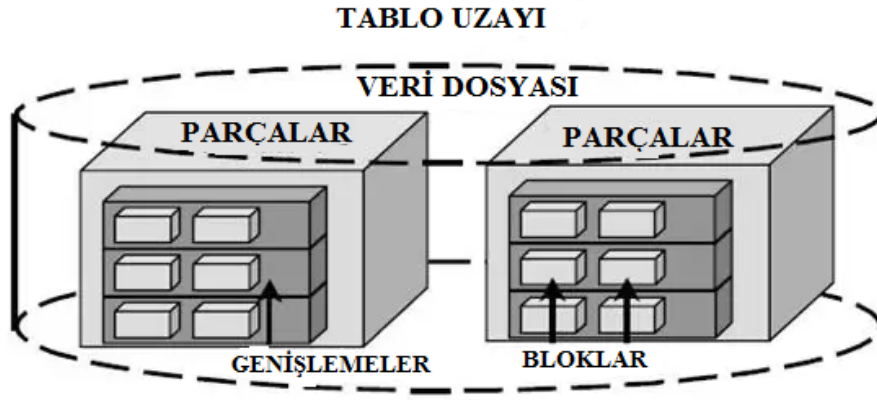
Oracle veritabanı, tablo uzayları olarak bilinen bir veya birden fazla mantıksal depolama birimlerine bölünmüştür. Tablo uzayları da dilimler olarak bilinen mantıksal depolama birimlerine bölünmüştür. Uzanımlar ise Oracle sisteminde en küçük veri birimi olan Veri Bloklarından oluşmuştur.

Mantıksal kısım başlıca 4'e ayrılmaktadır.

- Tablo Uzayları (Tablespaces)
- Veri Blokları (Data Blocks)
- Genişlemeler (Extends)
- Parçalar (Segments)

Tablespaces (Tablo Uzayları)

Her bir veritabanı bir ya da daha fazla tablespace'den oluşur. Buradaki verileri tutmak içinde her bir tablespace için bir ya da daha fazla veri dosyası oluşturulur. Her Oracle veritabanında kullanıcın isteği dışında kurulumu esnasında SYSTEM ve SYSAUX adlarında 2 adet tablespace oluşturulur. Bunlar aslında **smallfile tablespace** olarak adlandırılan küçük ölçekli alanlardır. Bunun dışında uygulamaya göre daha büyük ölçekli **bigfile tablespaceler** oluşturulabilir. Bu durumda özellikle 64bit sistemlerde çok büyük alanlar oluşturulmasına imkân vermektedir.(Maksimum 8 exabyte).Bu da onlarca küçük alan ile uğraşmaktan kullanıcıyı kurtarır.



Tablo Uzakı Yapısı

Veri Blokları (Data Blocks)

Oracle verileri, Veri Bloklarında saklar. Veri Bloklarına mantıksal bloklar, Oracle blokları veya sayfaları da denilmektedir. Disk üzerinde bir data block byte olarak belli bir alanı işgal eder. İşletim sistemlerinden bu blok yönetimini almışlardır. Özellikle Tablespace oluşturulurken beklenen süreç oluşturulan nesnesinin kendi blok yapısına dönüştürülme işlemidir. Bu alan DB_BLOCK_SIZE parametresi ile belirlenir.

Genişlemeler (Extends)

Belirli sayıda veri bloklarından meydana gelen mantıksal veritabanı depolama birimidir. Nesnelerin büyümesi için alınan yerlere bu ad verilir. Bir veya birden fazla Uzanım bir araya gelerek Dilimleri oluşturur. Dilimlerdeki boş alanlar tamamen dolduğu zaman, Oracle Dilimler için yeni Uzanımlar tahsis eder.

Parçalar (Segments)

Bir veya birden fazla genişlemeden meydana gelen diğer bir mantıksal depolama birimidir. Oracle tablolarındaki verileri sakladığı parçalar haricinde bir kaç farklı parça yapısı daha kullanılmaktadır. Bunlardan indekslere ait olan veriler için indeks parçaları, SQL komutları işletilirken duyulan ihtiyaçlar için Geçici(Temporary) Parçalar kullanılmaktadır.

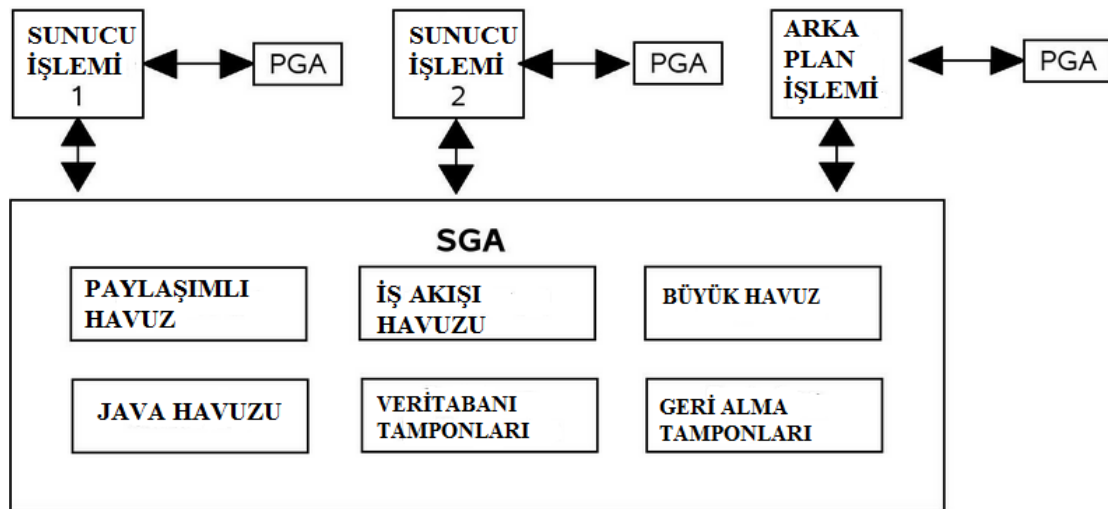
ORACLE Veritabanı Hafıza Yapısı

Oracle'da 2 tane hafıza yapısı mevcuttur. Bunlar da PGA(Program Global Area) ve SGA(System Global Area)'dır.

PGA, sunucu işlemleri için gereken veri ve kontrol bilgilerini tutmak için tahsis edilen tampon bellektir. Oracle tarafından bir sunucu işlemi başladığında otomatik olarak tahsis edilir. İçinde tutulacak bilgi oracle konfigürasyonuna bağlıdır.

SGA, Hafızanın veritabanı instance'ı ile ilgili veri ve kontrol bilgisinin bulunduğu yerdir. SGA tek bir oracle instance'ın ihtiyaç duyduğu veri ve kontrol bilgilerinin tutulduğu paylaşıma açık bellek alanıdır. Oracle oturumu açıldığı zaman tahsis edilir, kapatıldığı zaman sisteme geri verilir. Sunucu ve arka plan işlemleri tarafından paylaşılır. En iyi performansı sağlaması için SGA'nın ana bellek boyutlarını aşmama koşuluyla olabildiğince büyük olması gerekir. Bu sayede bellek de daha fazla veri tutulur ve disk okuma yazmasını azaltır. SGA' de veri dosyalarından okunan veri blokları, paylaşımlı SQL bellek alanları, kullanıcı bilgileri, değişiklik ara belleği bulunmaktadır. SGA'yı 6 bölüm olarak inceleyebiliriz. Bunlar:

- Büyük Havuz(Large Pool)
- Paylaşılmış Havuz(Shared Pool)
- Geri Alma Tamponları(Redo Log Buffer)
- Veritabanı Tamponları(Database Buffer Cache)
- İş Akışı Havuzu(Streams Pool)
- Java Havuzu (Java Pool)



Oracle Temel Hafıza Yapısı

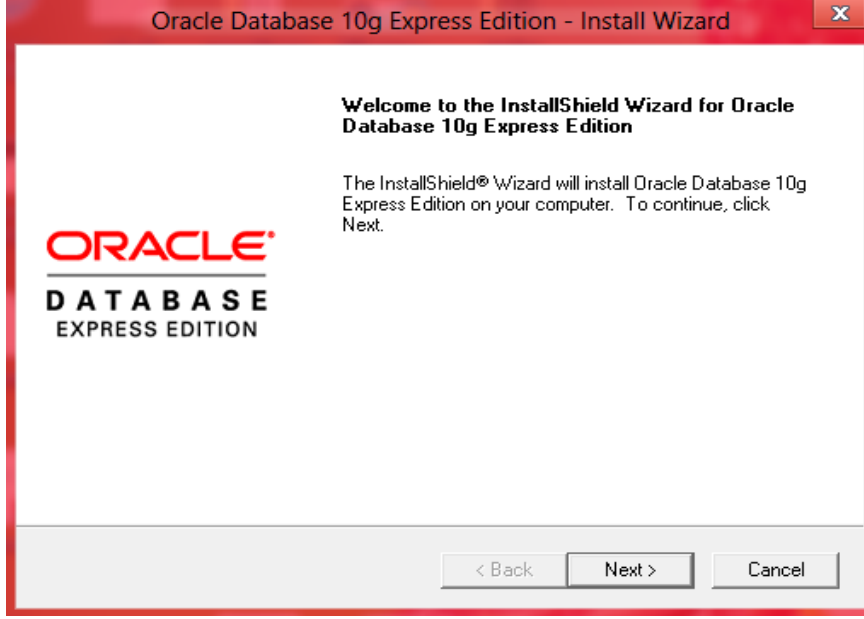
Bu kitap daha çok SQL ile programlama amacını gütmektedir. Bu yüzden Veri tabanı yönetimi ve Oracle veritabanı yapısının detaylı incelemesi kitapta bulunmamaktadır. Kitabın sonunda verilen kaynakları takip ederek Oracle yapısı hakkında daha fazla bilgi alabilirsiniz.

ORACLE Database 10g Express Kurulumu

Oracle Database Express son sürümü 11g sürümüdür. Aşağıdaki linkten, Oracle'ın kendi sitesinden bedava olarak indirebilirsiniz. Fakat kitapta bir önceki sürümü olan 10g XE kullanılacaktır. Kurulumu başlamadan önce Şekil -1'deki ekrandan gösterilen aşağıdaki linkten <http://www.oracle.com/technetwork/database/express-edition/downloads/index.html> işletim sistemimizin desteklediği ilgili Oracle Database Setup dosyası indirilir.

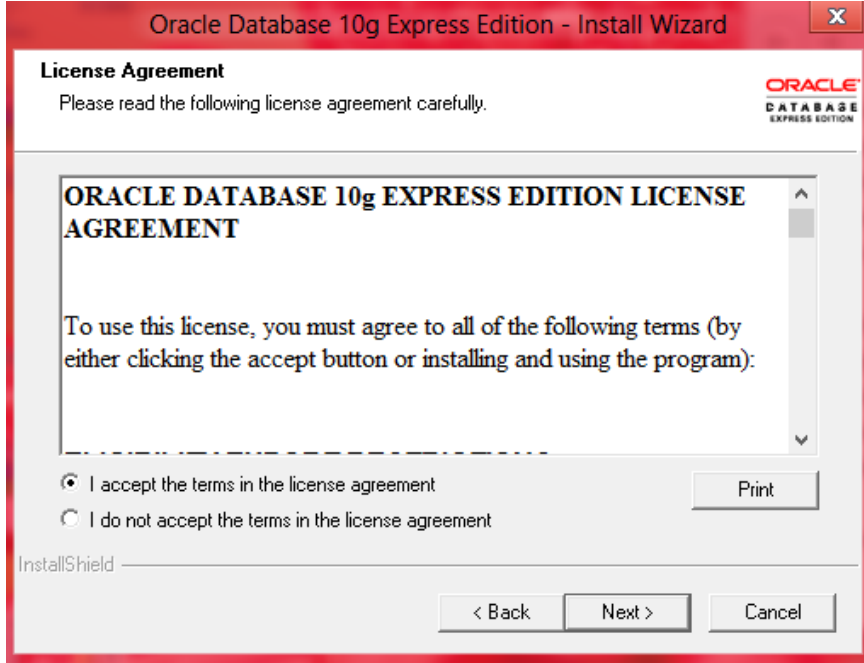


Oracle Database Download Adresi



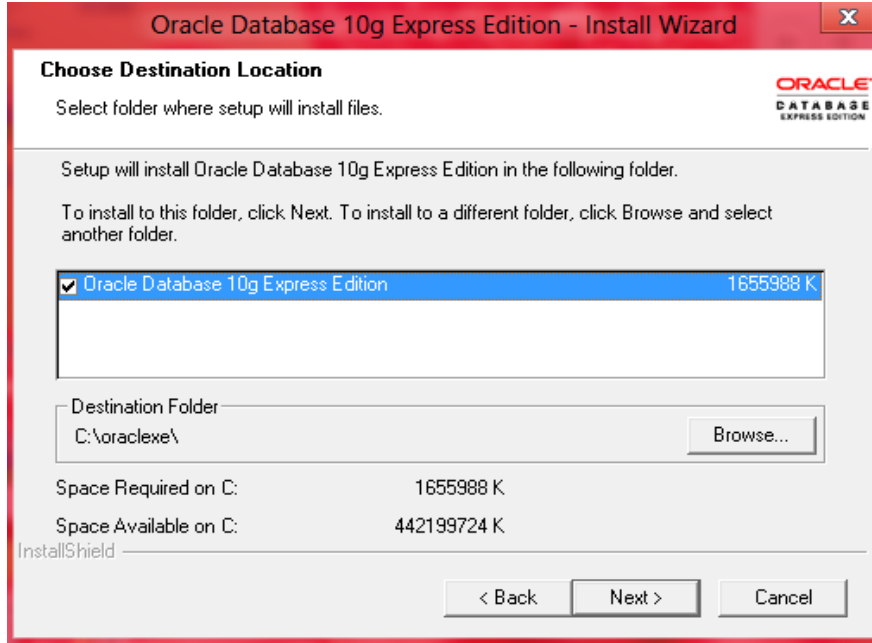
İlk Kurulum Ekranı

İnternet aracılığıyla indirdiğimiz ya da başka bir medyada bulunan Oracle 10g Express Setup dosyasını çalıştırdığımız zaman ilk karşımıza şekil 2'deki kurulum ekranı gelir. Next butonuna basılarak bir sonraki adıma geçilir.



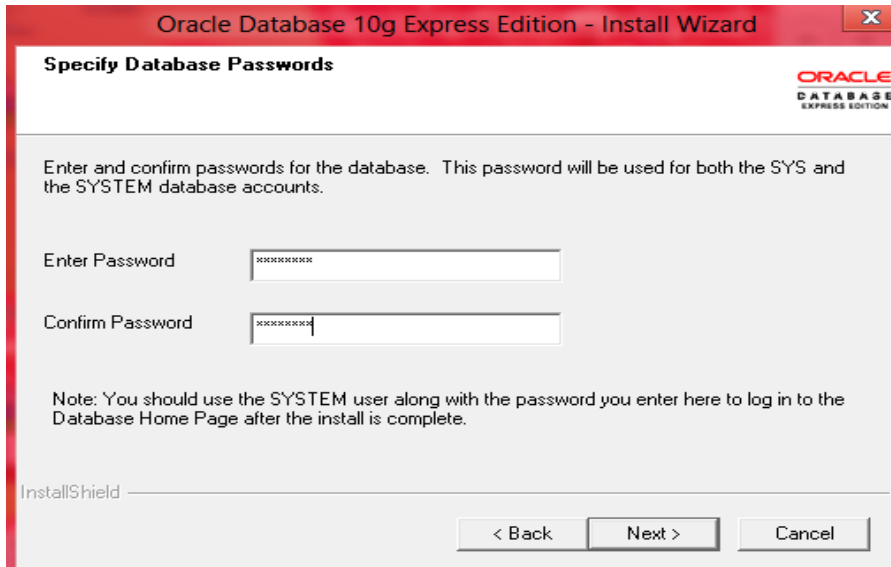
Lisans Kabul Ekranı

Bu adımda kurulumlarda klasik olarak karşımıza çıkan lisans sözleşmesini kabul etmemiz ekran gelecektir. *"I accept the terms in the licence agreement"* yazılı seçeneği işaretledikten sonra Next butonu ile bir sonraki ekrana geçilir.



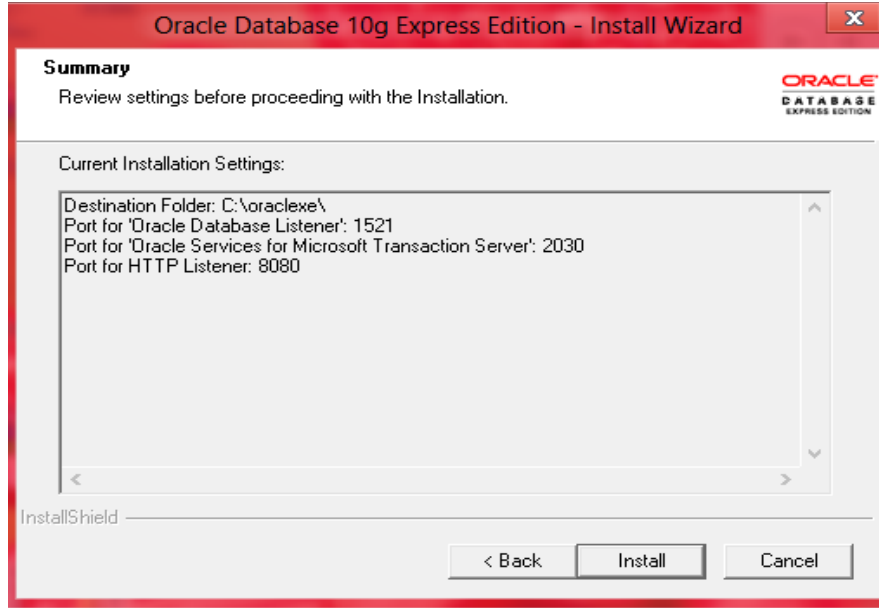
Oracle'ın Nereye Kurulacağını Gösteren Ekran

Bu adımda veritabanının bilgisayarınızda hangi disk bölümüne kurulacağı seçilecektir. Kurulumun yapılacağı disk üzerindeki boş alan, gerekli alan ve programın kaplayacağı alan gösterilmektedir. Herhangi bir değişiklik yapılabilir fakat yapılmaz ise Next butonuna basarak bir sonraki adıma geçilir.



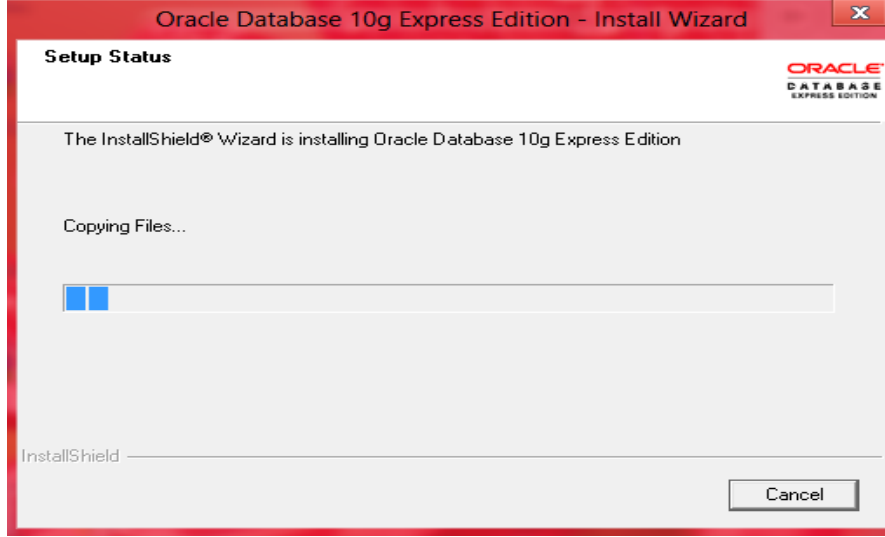
Sys ve System Kullanıcıları için Şifre Oluşturma Ekran

Bu adımda veritabanına ilk bağlanırken kullanılan *SYS* veya *SYSTEM* kullanıcısına ait şifre bilgisinin girilmesi istenmektedir. Boş geçilemez ve çözülmesi zor bir şifre seçilmesi güvenlik açısından da önemlidir. Şifre bilgisi veritabanına bağlanmak için gerekli olmazsa olmaz bir işlemdir.



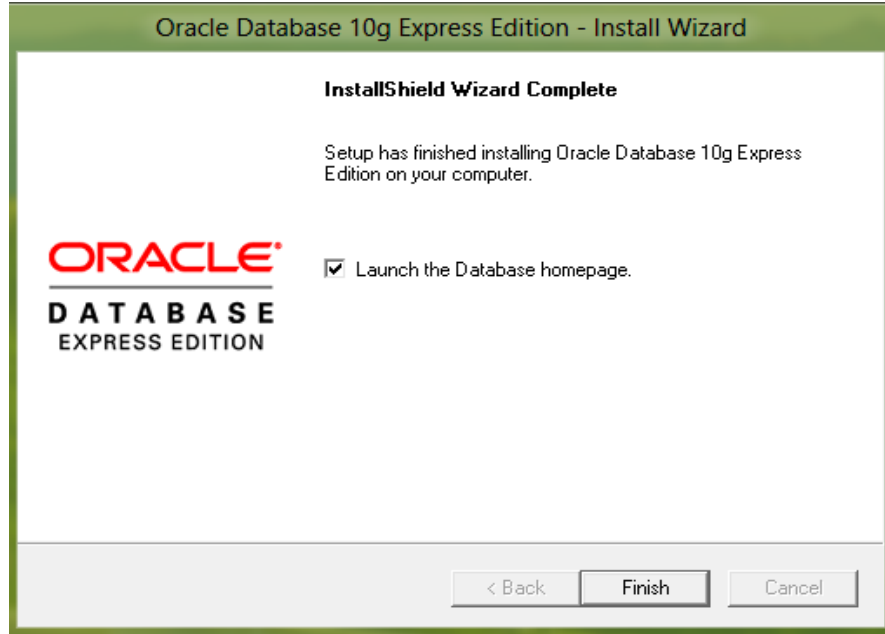
Yapılan Kurulum İşlemleri için Özet Ekranı

Bu adımda kurulumun nereye yapılacağı, Oracle veritabanının hangi portları kullanarak işlem yapacağı gösterilmektedir. İstemci bilgisayarlarda bu fazla önemli değildir fakat Server işletim sistemlerinde, güven duvarına takılmaması için ilgili portlar önem arz etmektedir.



Dosya Kopyalama Ekranı

Bu adımda kurulumun yapıldığı dosyalar bilgisayara kopyalanmaktadır.



Kurulum Tamamlanma Ekranı

Bu ekranı gördüğümüzde artık Oracle Database 10g Express Edition bilgisayarınıza başarı ile kurulmuş demektir. Finish Butonuna basarak kurulumu tamamlanmaktadır.

ORACLE Veritabanı Yönetim Araçları

Oracle veri tabanını SQL ile programlamak, geliştirmek ve yönetmek için doğru ve güvenilir bir ortam sağlayan yazımlar Veritabanı Yönetim Araçları(VTYA) olarak bilinir. Bu

yazılımlar veri tabanı geliştiricileri için kolaylık sağlayan, onların karşısına çıkabilecek sorunlarda onların işlemleri ve tasarımları daha iyi yönetebilmelerini sağlayan yazılımlardır. Piyasada ve Oracle'ın kendisinin ürettiği 3.Parti birçok SQL programlama ve geliştirme araçları mevcuttur. Bu araçların ortak özellikleri ya da olması gereken özellikleri:

- İyi bir SQL editörü olmaları
- Hata Ayıklama(Debugging) sisteminin olması
- Güvenlik Yönetiminin olması
- Oturum Gezginin olması
- Çoklu Dil desteğinin olması
- Veri aktarım işlemlerinin olması (Import/Export)
- Performans izleme işleminin olması
- Nesne Gezginin olması vs. olarak verebiliriz.

Bu özelliklere sahip yazılımlardan siz öğrencilerin ve Oracle database geliştiricileri için seçimleri bu kitaba koymak istedik. Size en iyi tanıdık gelecek olan örnek olması açısından Microsoft SQL Server'ın Enterprise Manager ya da yenisi Management Stdio verilebilir. Şimdi ise Oracle Database VTYA'larına örnekler verelim.

- SmartDBA Cockpit(BMC - www.bmc.com)
- DBTools (SoftTree Technologies - www.softtreetech.com)
- RapidSQL(Embarcadero - www.embarcadero.com)
- PL/SQL Developer (Allround Automations - www.allroundautomations.com)
- DBA Connect(DataSparc, Inc. - www.datasparc.com)
- SQLInsigth (Isidian Technologies, Inc. - www.isidian.com)
- KeepTool (www.keeptool.com)
- Toad (Quest Software - www.toadsoft.com)
- Oracle SQL Developer (www.oracle.com)

Seçim olarak Toadsoft firmasının developer programı editör seçimi olarak piyasada daha popüler. Fakat kitapta ilerleyen bölümlerde Oracle'ın Java üzerinde çalışan **Oracle SQL Developer** programı kullanılacaktır.

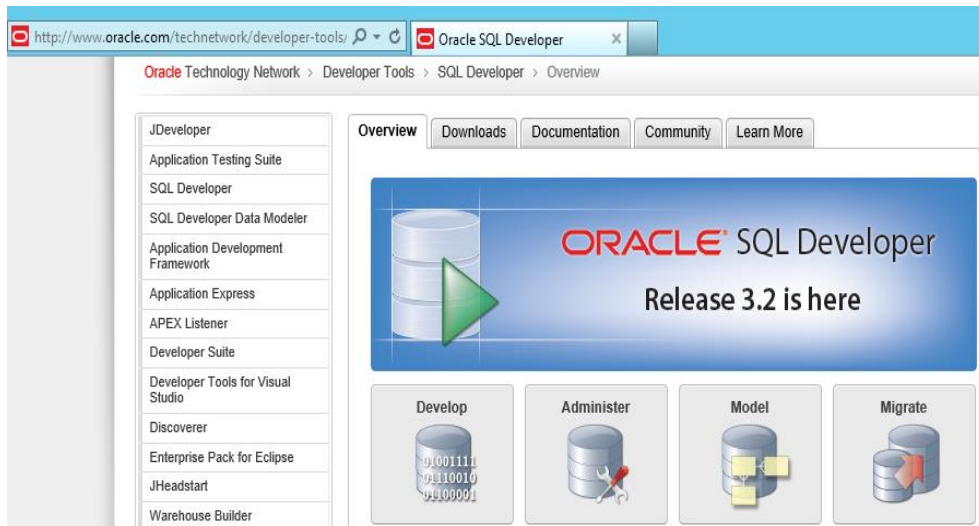
ORACLE Sql Developer 3.2.2 Kurulumu

Oracle SQL Developer, Oracle Veritabanı geliřtirmeyi ve yönetimini basitleřtiren ücretsiz bir entegre geliřtirme ortamıdır. SQL Developer da hareket için komple uçtan uca sizin PL / SQL uygulamaları geliřtirme, sorguları ve komut çalıştırmak için bir çalışma sayfası, veritabanı yönetimi için bir DBA konsolu, raporlama yüzü, tam bir veri modelleme çözümü ile bir platform sunmaktadır. Oracle için 3.parti bir yazılım olarak karşımıza çıkmaktadır. Bu kitapta, son sürüm olan Oracle Sql Developer 3.2.2 kullanılmıştır.

Oracle SQL Developer'ı aşağıdaki linkten indirebilirsiniz.

<http://www.oracle.com/technetwork/developer-tools/sql-developer/downloads/index.html>

Şekil 9'da görüldüğü gibi işletim sisteminize göre bir platform seçmelisiniz.



Oracle Sql Developer Sayfası

Yukarıdaki adresten giriş yaptığımızda yeni sürüm hakkında bir ön bakış ile karşılaşırız. Bu kısmı inceledikten sonra yan sekmede bulunan *Downloads* bölümünden bizim işletim sistemimize uygun sürümü seçip indirebilirsiniz.

[Overview](#) [Downloads](#) [Documentation](#) [Community](#) [Learn More](#)

**Oracle SQL Developer 3.2.2 (3.2.20.09.87)**
November 1, 2012

Thank you for accepting the OTN License Agreement; you may now download this software.

- [3.2.2 Release Notes](#)
- [3.2.2 Bugs Fixed](#)
- [New Features](#)
- [Documentation](#)

Platform

Windows 32-bit - zip file <i>includes</i> the JDK1.6.0_35	Download 209 M
Windows 32-bit - Installation Notes	Download 174 M
Windows 64-bit - Installation Notes	Download 174 M
Mac OS X - Installation Notes	Download 173 M
Linux RPM - Installation Notes	Download 173 M
Other Platforms - Installation Notes	Download 174 M

Oracle Sql Developer Platformları



[Oracle](#) Welcome NOT_FOUND Account Sign Out Help Select Country/Region Communities I am a... I want to...
PRODUCTS AND SERVICES SOLUTIONS DOWNLOADS STORE SUPPORT TRAINING PARTNERS ABOUT


Oracle Technology Network > Java > Java SE > Downloads

[Overview](#) [Downloads](#) [Documentation](#) [Community](#) [Technologies](#) [Training](#)

Java SE Downloads

[Latest Release](#) [Next Release \(Early Access\)](#) [Embedded Use](#) [Previous Releases](#)

**DOWNLOAD** 
Java Platform (JDK) 7u10

**DOWNLOAD** 
JavaFX 2.2.4

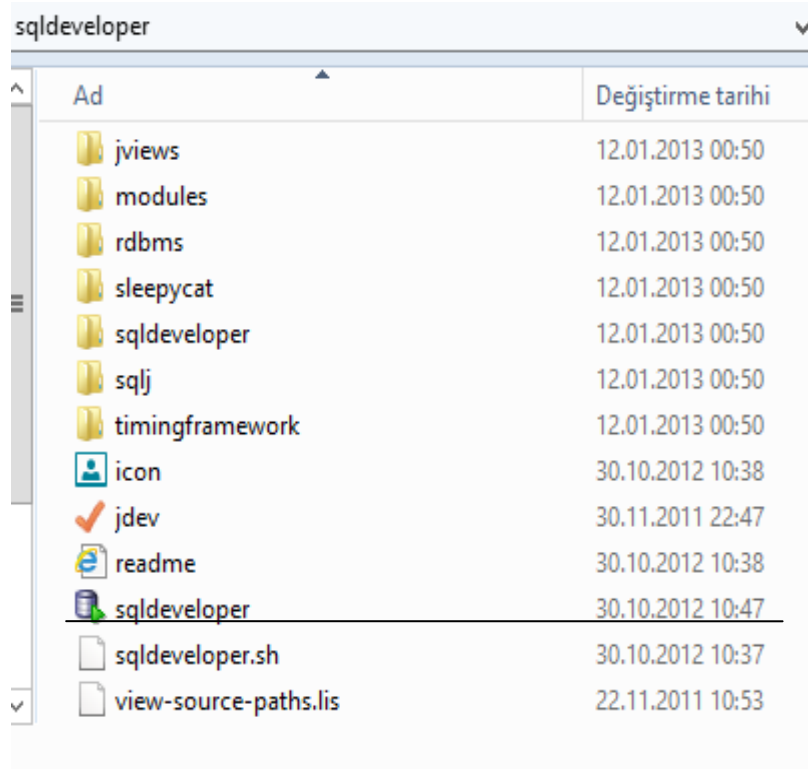
**DOWNLOAD** 
JDK 7u10 + NetBeans

Oracle Java SE İndirmeleri

Sql Developer'ın çalıştırılabilmesi için gerekli, Java programının kurulu olduğu bölümü sizden istemektedir. Bilgisayarımızda Java Kurulu ise Browse butonu ile Exe dosyasını gösterebilir veya aşağıdaki adresinden indirebiliriz.

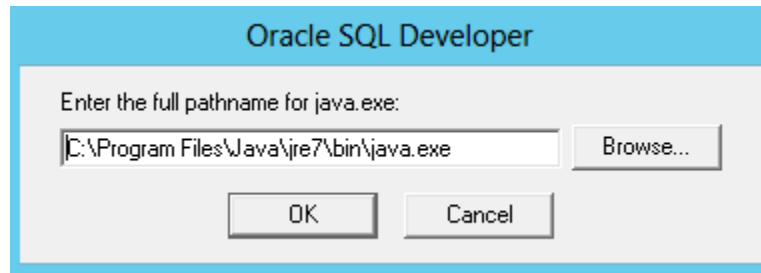
<http://www.oracle.com/technetwork/java/javase/downloads/index.html>

Oracle Sql Developer İndirildikten sonra ana klasör tıklanır ve içindeki *sqldeveloper* uygulama dosyası çalıştırılır.

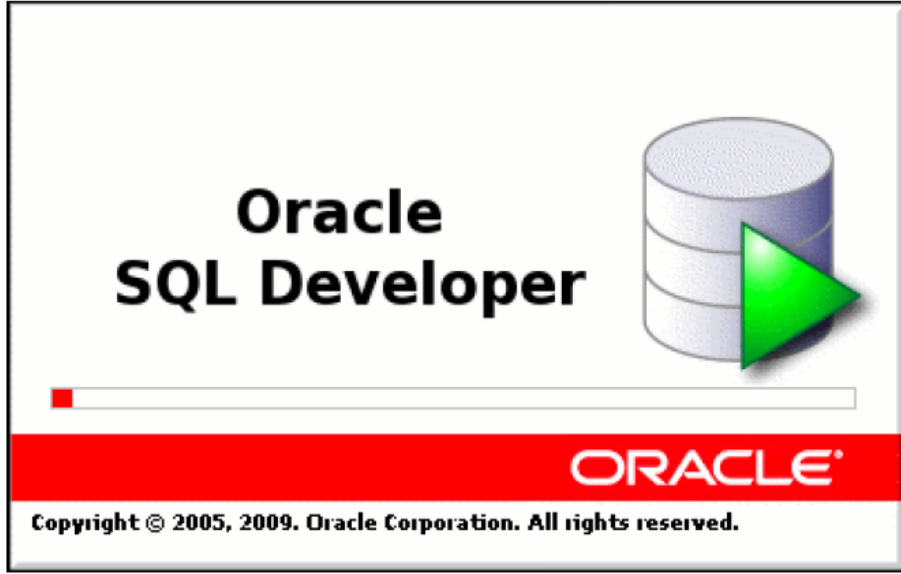


Oracle SQL Developer için Java Yolu Gösterme

Programı çalıştırdığımızda bize Java.exe'nin yolunu göstermemiz için bir ekran çıkacaktır. Buradan Java.exe dosyası gösterilir.Yolu gösterdikten sonra OK butonuna bastığımızda Oracle Sql Developer programı yüklenmeye başlayacaktır.

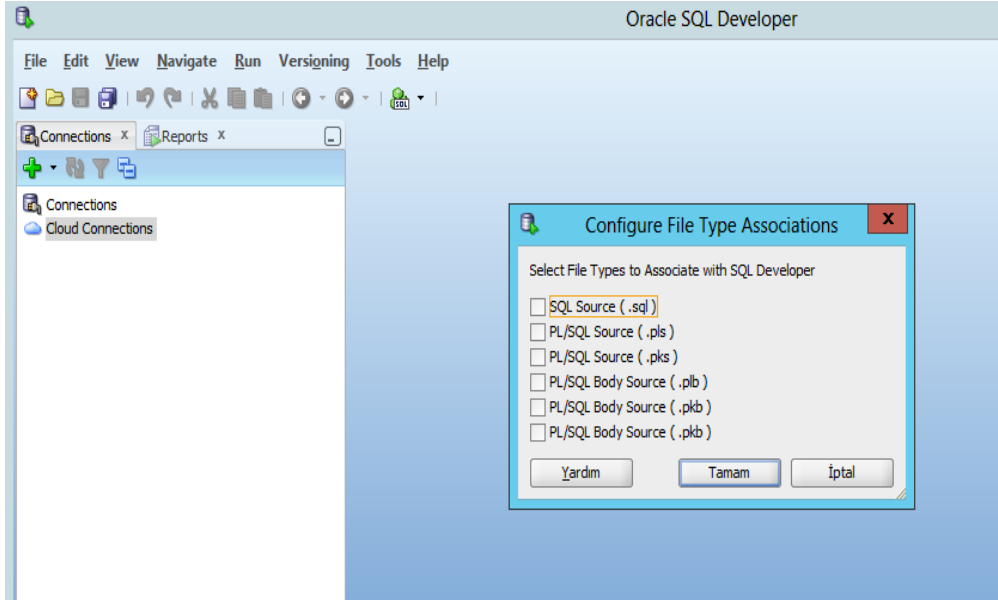


Oracle SQL Developer için Java Yolu Gösterme

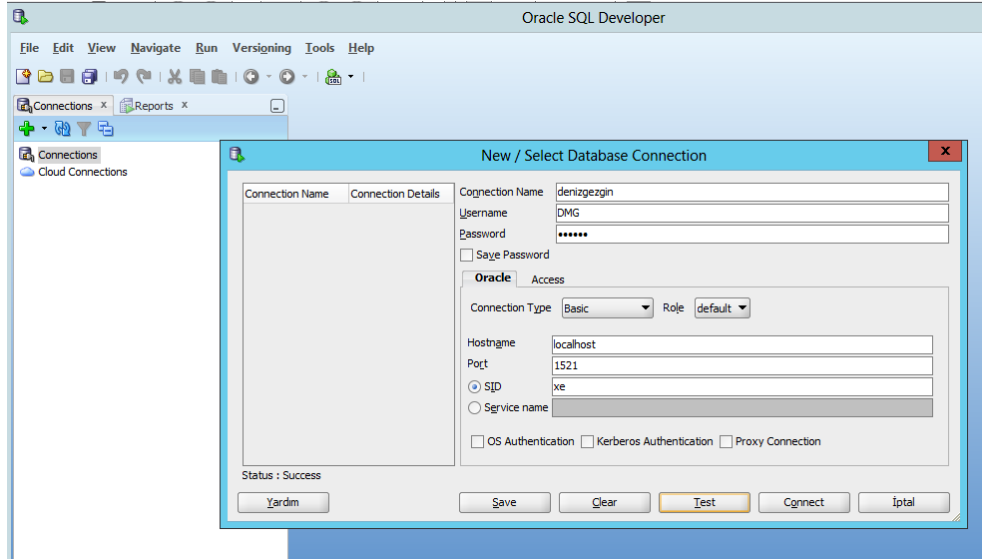


Oracle SQL Developer Programının Çalıştırılması Ekranı

Program yüklendikten sonra ilk ekran olarak karşımıza Şekil 15'teki ekran çıkacaktır. Sql Developer'ın dosya uzantılarının seçilebildiği, programın default olarak açacağı tipler gelecektir. İlgili seçimler yapıldıktan sonra ki biz tüm uzantıları seçiyoruz, Tamam Butonu ile bitirilebilirsiniz. Burada örnek olması açısından **.sql** uzantılı dosyaları boş bırakıp olanların MS SQL Server kullanıyorsanız Management Stdio ile açılmasını sağlayabilirsiniz.



Oracle SQL Developer'da dosya uzantıları seçim ekranı



Oracle SQL Developer'da bir Connection Oluşturma

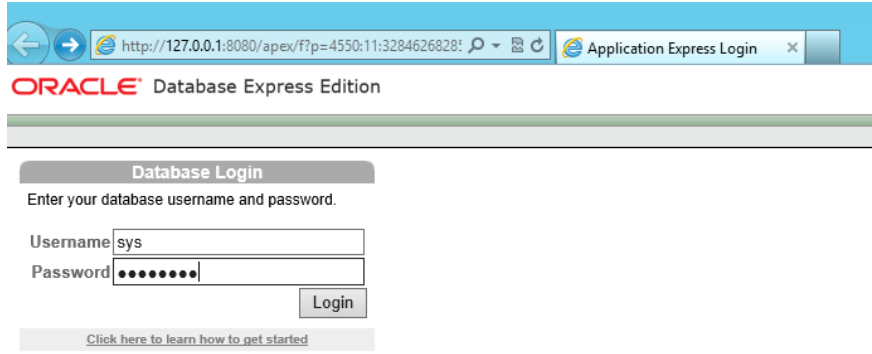
Dosya uzantılarını seçtikten sonra sıra *connection* işlemine gelmiştir. Bunun için bir kullanıcı adı ve şifre gerekmektedir. Biz kitap boyunca yarattığımız ve bir sonraki konuda anlatacağım kullanıcı oluşturma işlemi ile yarattığımız **DMG** kullanıcısı ile Oracle'a bağlantı sağlayacağız. Fakat **HR** kullanıcı isminde default olarak bir veri tabanı yada şemaya bağlanabilirsiniz. Yâda **Sys** veya **System** kullanıcısı ile **SYSDBA** rolünde de bağlantı sağlayabilirsiniz. Tüm izinli tablo ve nesnelere ulaşabilirsiniz. Burada yapılması gereken işlemler sırasıyla:

- Bir bağlantı ismi seçmek. Burada **denizgezzgin** verilmiştir.
- Oluşturulan ya da var olan bir kullanıcı ismi ve şifresi girilmelidir.
- Değişik bir yapılandırma olmadığı sürece Role ya da diğer seçenekler default bırakılmalıdır.
- Test butonuna basıp, çalışmanın başarılı olup olmadığı sorgulanmalıdır.
- Son olarak **Save** diyerek bilgileri bir dahaki bağlantı için kaydetmemiz kolaylık sağlayacaktır.

ORACLE Database 10g Üzerinde Kullanıcı Oluşturma

Oracle XE 10 g kurulumunda Sys ve System kullanıcısına bir şifre belirlemiştik. Bu kullanıcı adı, şifre ile sisteme Web konsolu üzerinden giriş yapalım. Bu işlemi Oracle XE 10g

kurduğumuzda sistemde beliren *Go To Database Homepage* simgesini tıklayarak ulaşabilirsiniz.



Oracle Database Homepage ile login işlemi

Sisteme giriş yaptıktan sonra Web konsolundan da veri tabanı üzerinde değişiklik yapabileceğimiz ekran karşımıza gelmektedir. Burada dört temel işlevsel nesne karşımıza çıkmaktadır. Bunları simge ve işlevlerine göre aşağıda bulabilirsiniz.



Administration



Object Browser

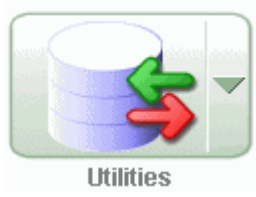


SQL

Veritabanı kullanıcı hesaplarını yönetmek, bellek, depolama ve ağ bağlantıları, monitör veritabanı aktivitesini yönetme ve yapılandırma bilgilerini görüntülemek.

Veri tabanı nesnelerini silme, oluşturma, view etme, değiştirme, browse etme. Hata raporlamadan yararlanırken tetikleyicileri, fonksiyonları, prosedürleri ve paketleri derleme ve edit işlemleri için PL/SQL editörünü kullanma.

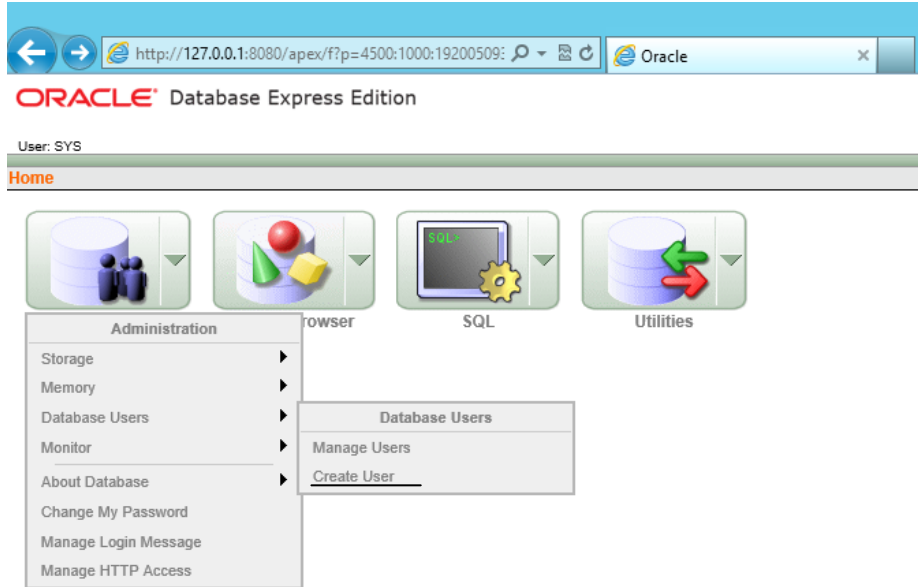
- Listelenen SQL araçlarına erişim :
- **SQL Commands**—Sql ve PL/SQL bloklarını çalıştırma. Scriptleri çalıştırma ve Sorguları kaydetme.
- **SQL Scripts**—Script dosyalarını oluşturma, silme, edit etme, çalıştırma ve view etme. Yerel Dosya Sisteminden scriptleri download ve upload etme.
- **Query Builder**—SQL sorguları için grafik bir ara yüz sunma. Sorgu sonuçlarını ve sorguları görüntüleme.



Veri Yükleme ve Geri yükleme, DDL üretme, view nesnesi raporları ve silinmiş veritabanı nesnelerini restore etme.

Burada eklemek gereken bir husus daha vardır. System kullanıcısı dışında başka bir kullanıcı ile sisteme giriş yapılırsa, bu dört işlevsel simgenin yanına Application Builder isimli bir icon daha belirecektir. Bu işlevi de Oracle uygulamalarını geliştirmek için kullanabilirsiniz.

Bu bilgileri verdikten sonra Administration bölümünden kullanıcı oluşturma işlemi için **Database Users** bölümüne girilip **Create User** seçeneği tıklanır.



Kullanıcı Oluşturma Ekranı-I

Create User seçeneği tıklandığında karşımıza oluşturulacak kullanıcı için bilgilerin girilmesi için gerekli ekran ve bunun altında veri tabanı üzerindeki rolü ve yapabileceği işlemlerin seçebileceği ekran gelmektedir. Biz bu kitapta DMG kullanıcıını oluşturduk. DMG kullanıcıına tüm izinleri verdik. Son olarak DMG kullanıcıını DBA(Database Administrator) olarak da işaretledik. Böylece tüm yetkiyi DMG kullanıcıına vermiş olduk. Bundan sonra Create butonuna basarak kullanıcıımızı oluşturup, Web konsolundan çıkış yaptık. Sizde aynı işlemleri kendi kullanıcı üzerinde yapabilir ya da kitap ile aynı bilgilerle gidebilirsiniz.

Oracle Database Express Edition

User: SYS

Home > Administration > Manage Database Users > Create Database User

Create Database User

* Username: DMG

* Password:

* Confirm Password:

Expire Password: ☐

Account Status: Unlocked

Default Tablespace: USERS

Temporary Tablespace: TEMP

User Privileges

Roles:

☒ CONNECT ☒ RESOURCE ☐ DBA

Direct Grant System Privileges:

☒ CREATE DATABASE LINK ☒ CREATE MATERIALIZED VIEW ☒ CREATE PROCEDURE

☒ CREATE PUBLIC SYNONYM ☒ CREATE ROLE ☒ CREATE SEQUENCE

☒ CREATE SYNONYM ☒ CREATE TABLE ☒ CREATE TRIGGER

☒ CREATE TYPE ☒ CREATE VIEW

[Check All](#) [Uncheck All](#)

Kullanıcı Oluşturma Ekranı-II

ORACLE Database 10g Üzerinde Örnek Veri Tabanı

Kitabın diğer bölümlerinde çok sayıda veri tabanı örnekleri verilmiştir. Oracle Bölümünde de PL/SQL komutlarını ve bloklarını çalıştırıp uygulamak için ilişkisel bir veri tabanı modeli üzerinde çalışacağız. Veritabanımız hakkında bilgi ve diyagram bu bölümde size verilecektir.

Veri Tabanının ismi, **UEFA Şampiyonlar Ligi Maç Yönetimi** olarak karşımıza çıkmaktadır. Bu veritabanı ile sorgulanmak istenenler:

- Şampiyonlar Ligi takımlarının bilgileri.
- Maç yapılan Stat bilgileri.
- Takımlar arasında yapılan maçlar ve sonuç bilgileri.
- Gruplardaki takımlar ve Puan durumu bilgileridir.

Örnek Veritabanının SQL Kodları

DDL (Data Definition Language) Kodları

Gruplar Tablosunun Oluşturulması

```
Create Table Gruplar
( g_name varchar2(5),
  constraint pk_Gruplar primary key (g_name)
);
```

Stadlar Tablosunun Oluşturulması

```
Create Table Stadlar
(Std_Id number(4),
 isim varchar(50),
 kapasite number(4),
 ac_tar date,
 constraint pk_Stadlar primary key (Std_Id)
);
```

Takımlar Tablosunun Oluşturulması

```
Create Table Takimlar
(tkno numeric(10) not null,
 isim varchar2(50),
 kur_tar date,
 gname varchar2(5),
 butce varchar2(30),
 constraint pk_Takimlar primary key (tkno),
 constraint fk_Takimlar foreign key (gname) references Gruplar(g_name));
```

Maçlar Tablosunun Oluşturulması

```
Create Table Maclar
(mac_Id int not null,
 mac_tar date,
 gol_evsahibi smallint,
 gol_konuk smallint,
 tkm_Id_ev numeric(10),
 tkm_Id_konuk numeric(10),
 std_Id number(4),
 constraint pk_Maclar primary key (mac_Id),
 constraint fk_Takim1 foreign key (tkm_Id_ev) references Takimlar(tkno),
 constraint fk_Takim2 foreign key (tkm_Id_konuk) references Takimlar(tkno),
 constraint fk_stad foreign key (std_Id) references Stadlar(Std_Id)
);
```

Not: Yukarıdaki Takımlar tablosunu Create Table komutu ile oluştururken Puan alanını oluşturmadık. Bu alanı yine **ALTER Table** komutu ekleyebiliriz. Kod aşağıdaki gibidir. Bu kodlar kitabın önceki bölümlerinde anlatılmıştır.

Alter Table Takımlar

Add puan Integer;

DML (Data Manipulation Language) Kodları

Stadlar Tablosuna Veri Girilmesi

```
insert into Stadlar values (2345, 'İnönü', 50000, '02/01/1903');  
insert into Stadlar values (234, 'Şükrü Saraçoğlu', 40000, '10/01/1923');  
insert into Stadlar values (23, '24 Kasım', 3000, '12/12/1953');  
insert into Stadlar values (5645, 'Santiago Bernabeu', 55000, '02/01/1900');  
insert into Stadlar values (8014, 'Nou camp', 30000, '01/01/1920');
```

Gruplar Tablosuna Veri Girilmesi

```
insert into Gruplar values ('A');  
insert into Gruplar values ('B');  
insert into Gruplar values ('C');  
insert into Gruplar values ('D');  
insert into Gruplar values ('E');  
insert into Gruplar values ('F');
```

Takımlar Tablosuna Veri Girilmesi

```
insert into Takımlar values(1,'Real Madrid','01/01/1880','A','1000000',0);

insert into Takımlar values(2,'Beşiktaş','21/01/1901','A','1000000',0);

insert into Takımlar values(3,'Cska Moskova','18/01/1900','A','1100000',0);

insert into Takımlar values(14,'Chealsea','01/08/1910','B','800000',0);

insert into Takımlar values(13,'Juventus','11/01/1920','B','10000000',0);

insert into Takımlar values(11,'Dmg United','03/10/1978','B','2000000',0);
```

Maçlar Tablosuna Veri Girilmesi

Takımlar tablosunun verilerinin girdiğimizde puan alanı default (0) olarak girilmiştir. Fakat Maçlar tablosu girildiğinde bu alan ileride güncellenecektir.

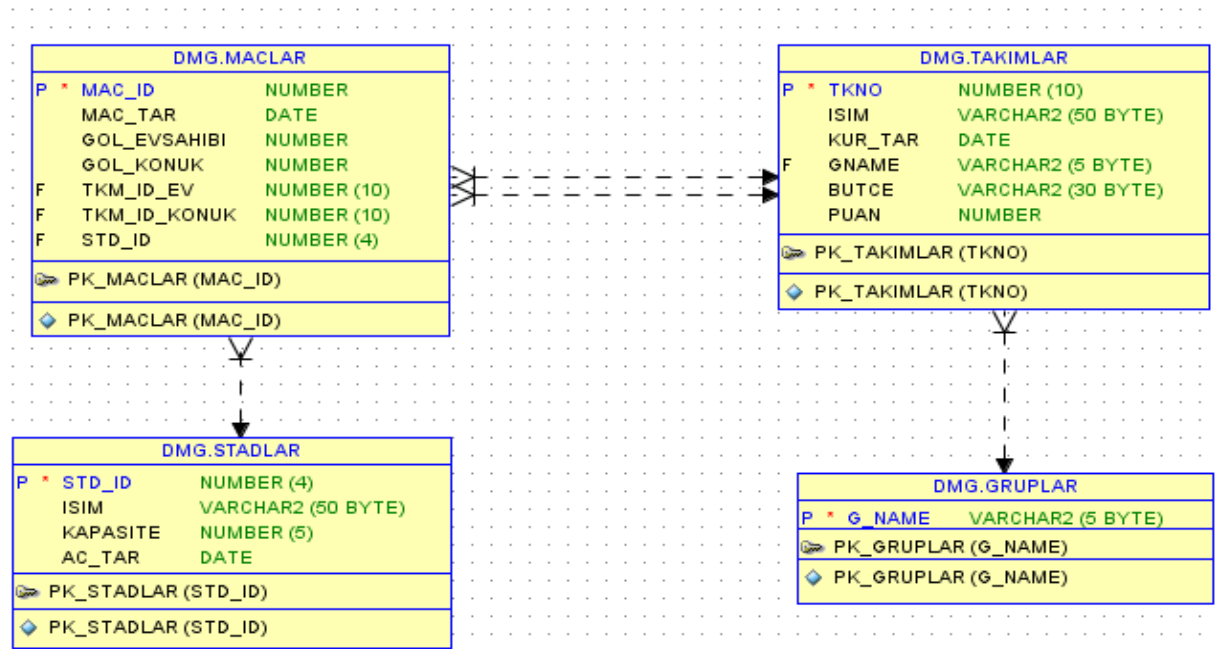
```
insert into Maclar values(23,'12/10/2010',5,3,1,2,5645);

insert into Maclar values(24,'12/12/2010',3,1,1,3,5645);

insert into Maclar values(20,'12/11/2011',1,1,14,13,5645);

insert into Maclar values(19,'19/11/2011',1,0,11,13,5645);
```

Örnek Veritabanının İlişkisel Diyagramı



SQL Yapısal Sorgulama Dili

Kitabın diğer bölümlerinde SQL ile ilgili komutlar işlenmiş ve örnekler verilmiştir. Bu bölümde de Oracle'ın veri sorgulama dili olan PL/SQL'e geçiş yaparken SQL özeti yapmak isterim. SQL, İlişkisel veritabanlarında veriye ulaşmak ve bu veriyi okumak, yorumlamak, değiştirmek ya da verinin depolandığı birimler ile her türlü işlemlerin yapıldığı evrensel veri tabanı kodlarıdır. SQL aslında dört kategoride toplanabilir.

Veri Tanımlama Dili (DDL- Data Definition Language)

Verinin tutulacağı tablo ya da veritabanı oluşturma komutları olarak bilinen komutları içeren SQL kısmıdır. Yapısal değişiklik yaparlar. Veriyle değil, verinin yerleştiği yapıyla uğraşırlar. Tanımlamalar bu kısımda yapılmaktadır. Bunlardan en genel kullanılanlar aşağıdaki komutlarıdır.

Komut	İşlevi
CREATE	Veritabanında nesne yaratır.
ALTER	Veritabanının yapısını değiştirir.
DROP	Veritabanından nesne siler.
TRUNCATE	Tablodaki kayıtları içerdikleri alan ile birlikte siler.
COMMENT	Yorum ekler.
RENAME	Nesnenin adını değiştirir.

Not: Bu komutları kullanmak yerine Veri tabanı yönetim konsolundan bu işlemleri görsel olarak yapmak daha kolaydır. Fakat bir veri tabanı uzmanı olacak kişinin tüm komutlara hâkim olması gerekmektedir.

Veri İşleme Dili (DML - Data Manipulation Language)

Verileri tabloya ekleme, tablodaki verileri silme ya da veriler üzerinde yapılan değişiklikler için kullanılan komutlarıdır. Bu komutlar program kodları içerisinde yada yazılan programlarda çağırılan prosedür ve fonksiyonların içinde çok kullanılmaktadır. Örnek olarak bir web sitesine kayıta girilen bilgiler veri tabanına **insert** komutu ile girilmektedir.

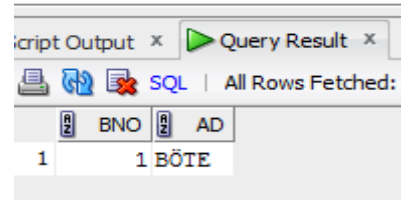
Komut	İşlevi
SELECT	Veritabanından kayıtları getirir.
INSERT	Tabloya kayıt ekler.
UPDATE	Tablodaki kayıtları günceller.
DELETE	Tablodan sadece kayıtları siler.
MERGE	UPSERT işlemi (ekleme veya güncelleme)
CALL	PL/SQL veya Java alt programı çalıştırır.

Not: Burada MERGE komutu ile ilgili bir örnek vermek isterim. Çünkü bu komut Oracle’da aktif olarak kullanılmaktadır. Amacı bir tabloya veri girilirken veri varsa güncelleme yoksa kayıt işlemini tek SQL sorgusu ile yapabiliriz.

Örnek: Bölüm tablosunda 1 numaralı kayıt var ise bunu güncelleyen yok ise BÖTE isimli bir kayıt girmesini sağlayan kodu yazınız.

```
merge into bolum b
using dual s on (b.bno=1)
when matched then
update set ad='BÖTE'
when not matched then
insert(bno,ad) values (1,'BÖTE');
```

```
select * from bolum;
```



BNO	AD
1	1 BÖTE

İşlem Kontrol Dili (TCL-Transaction Control Language)

Transaction veritabanındaki işlem ya da işlem bloğudur. Bu işlemlerin tablo ya da şema gibi depo birimleri ya da veriler üzerinde yapacaklar değişikliklerin geçerli olmasını ya da geri almayı sağlarız. Böylece transectionları kontrol etmiş oluruz. Buna örnek bir para transferi işlemidir. Para transferi yapılmadan **transaction** başlatılır. Bir sorun çıkmazsa sonunda **commit** ile bitirilir. Sorun çıkarsa işlem **rollback** ile geri çekilir.

Komut	İşlevi
COMMIT	Yapılan işlemleri onaylar.
SAVEPOINT	Daha sonra geri dönülecek bir dönüş noktası belirler.
ROLLBACK	Son COMMIT’e kadar olan kısmı geri alır.

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.

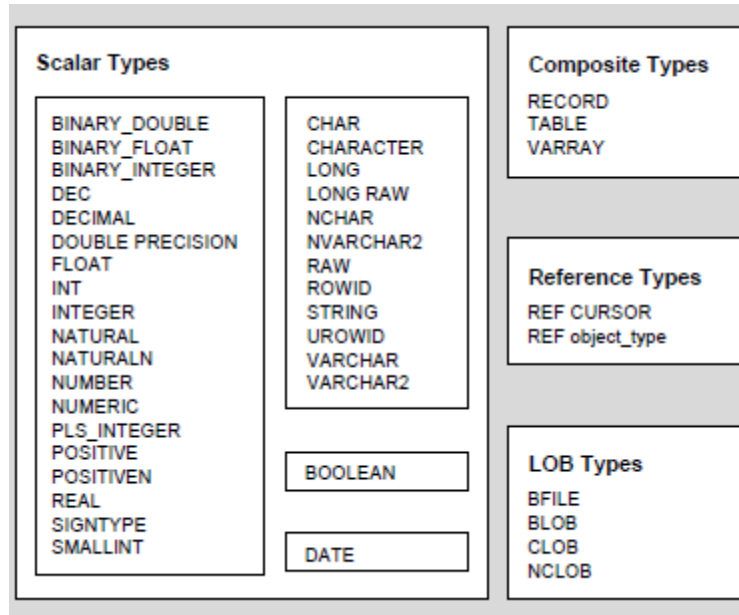
SQL komutlarını çalıştırabildiği gibi bunun yanında Oracle veri tabanı sistemleri için tetikleyiciler, yordamlar ve fonksiyonlar yazmak üzere geliştirilmiş, programlamada akış kontrollerini ve değişkenleri kullanmamıza imkân sağlayan bir dildir. Model olarak ADA dili örnek alınmıştır. Bu yüzden söz dizimlerinde 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;
```

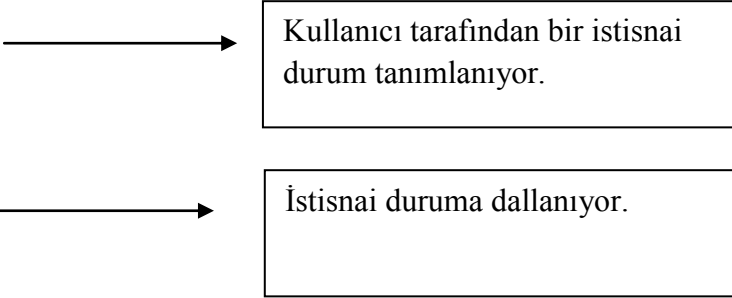
Örnek: Kullanıcı tanımlı istisnai durumları anlayabilmek için hazırlanmış bir örnektir. Bu örnek kullanım için tam isabetli bir seçim olmasa da yapının nasıl tanımlanacağını ve nasıl kullanılacağına güzel bir örnektir. Sayı değerimiz 5, bunun 10'dan küçük olma şartına göre tanımlama kısmında oluşturduğumuz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

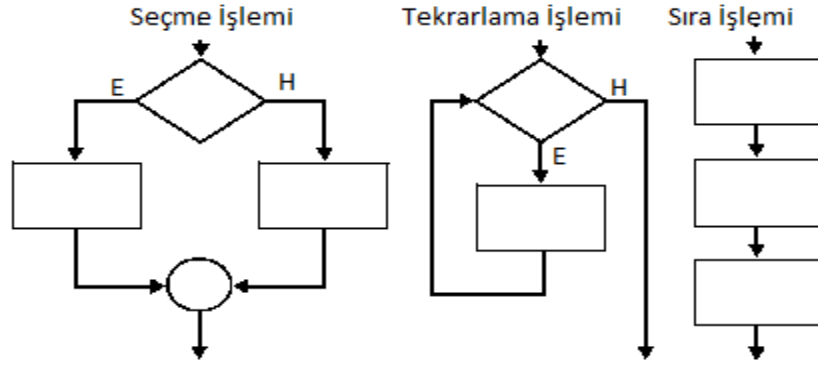
İstisnai Durum	Oracle Hata	SQL Hata Kodu
ACCESS_INT0_NULL	ORA-06530	-6530
CASE_NOT_FOUND	ORA-06592	-6592
COLLECTION_IS_NULL	ORA-06531	-6531
CURSOR_ALREADY_OPEN	ORA-06511	-6511
DUP_VAL_ON_INDEX	ORA-00001	-1
INVALID_CURSOR	ORA-01001	-1001
INVALID_NUMBER	ORA-01722	-1722
LOGIN_DENIED	ORA-01017	-1017
NO_DATA_FOUND	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.

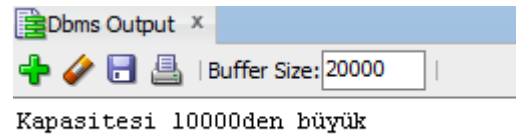
Bu yapı ile ilgili hemen bir örnek yapalım. Böylece pekişmesi sağlanabilir.

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.

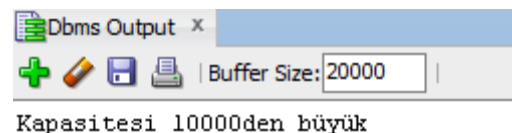
Örnek: 50 yıllık geçmişi olan kulüplerin adlarını tekrarsız ve artan listede sıralayan komutu yazınız.

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.

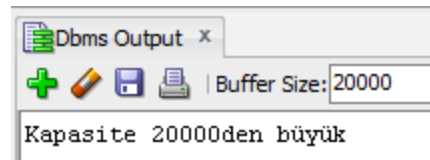
ELSE

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*;

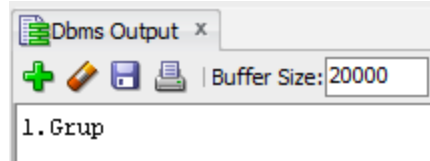
WHEN *ifade F* **THEN** *işlem 5*;

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.

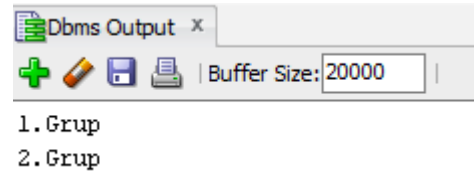
Örnek: Takımların bulundukları grupları tekrarsız bir şekilde rakam ile ekrana basan kodu yazınız.

```

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

```

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 transaction 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 işlemidir.

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 Takimler 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 olduğunda Evrel noktasına geri döndür

Not: Rollback to Evrel yerine direk **Rollback** yazarsak yine aynı işlemi yapardı. İşlemlerin hepsi geri alınır. Fakat aynı şey değildir. Buna dikkat etmeliyiz. **Evrel** 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. Aranılan 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,sütun2,..)*

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

Örnek: Takımlar tablosunda takım adı üzerinde bir index oluşturalım.

```
CREATE INDEX takım_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,
    dmğ OUT NUMBER
)
AS
BEGIN

    select puan into dmğ
    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 amaçlıdır. 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 olduğunda 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 farklı olarak tablolar verileri tutarak veri tabanında yer kaplarlar fakat view veri depolamaz, yer tutmazlar. Sadece SQL sorgusu veri tabanının 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 artırır.
- 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.
- Yıllı 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_ISLEM  
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.)

```

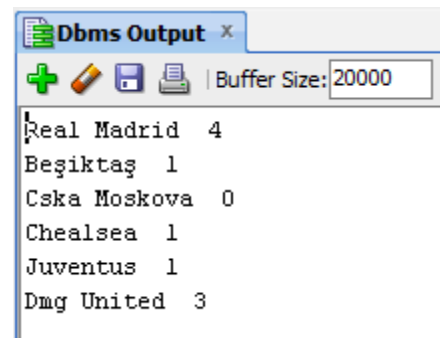
DECLARE
CURSOR c1 IS

SELECT isim,puan FROM TAKIMLAR WHERE puan < 9;

BEGIN
FOR takım IN c1
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 takimlist;
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(islem1 NUMBER, islem2 NUMBER, islem3 VARCHAR2) RETURN VARCHAR2;

END dort_islem;

/* Paket gövdesi tanımlanıyor */

CREATE OR REPLACE PACKAGE BODY dort_islem IS

FUNCTION hesapla(islem1 NUMBER, islem2 NUMBER, islem3 VARCHAR2)
RETURN VARCHAR2 IS
RESULT VARCHAR2(500);
BEGIN
IF islem3 = '+' THEN
RESULT := to_char(islem1) || '+' || to_char(islem2) || '=' || (islem1 + islem2);
ELSIF islem3 = '-' THEN
RESULT := to_char(islem1) || '-' || to_char(islem2) || '=' || (islem1 - 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;

```

Script Output x		Query Result x	
SQL All Rows Fetched: 9 in 0,014 seconds			
Toplama İşlemi		Çıkarma İşlemi	
1	6+4=10	6	4=2

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. */*

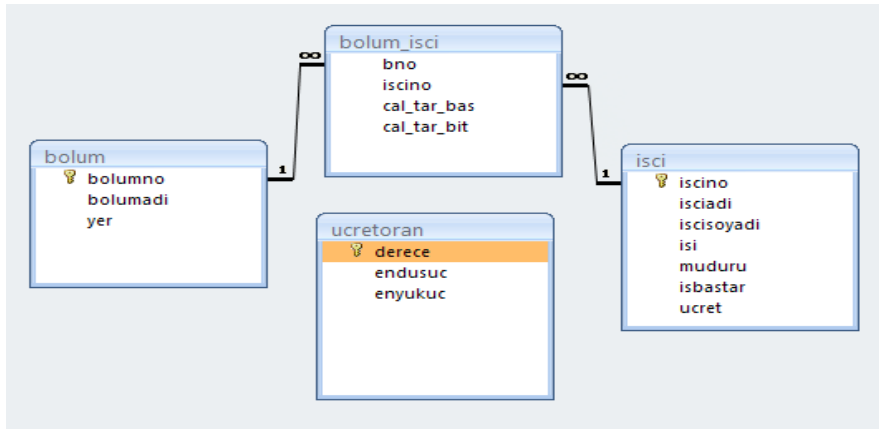
/ deniz kullanıcısı silinmiştir. */*

```
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/>