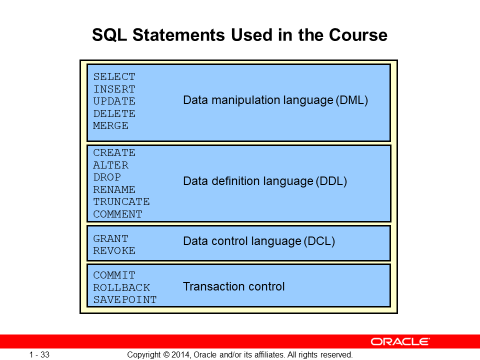
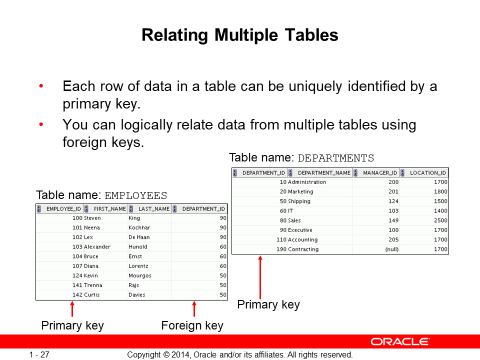
ORACLE SQL



Her tablo, tam olarak bir varlığı tanımlayan verileri içerir. Örneğin ÇALIŞANLAR tablosu, çalışanlarla ilgili bilgileri içerir. Veri kategorileri her tablonun üst kısmında listelenmiştir ve bireysel durumlar aşağıda listelenmiştir. Bir tablo formatı kullanarak bilgileri kolayca görselleştirebilir, anlayabilir ve kullanabilirsiniz.

Farklı varlıklarla ilgili veriler farklı tablolarda depolandığından, belirli bir soruyu yanıtlamak için iki veya daha fazla tabloyu birleştirmeniz gerekebilir. Örneğin, bir çalışanın çalıştığı bölümün yerini bilmek isteyebilirsiniz. Bu senaryoda, ÇALIŞANLAR tablosundan (çalışanlarla ilgili verileri içeren) ve BÖLÜMLER tablosundan (departmanlarla ilgili bilgileri içeren) bilgilere ihtiyacınız vardır. Bir RDBMS ile yabancı anahtarlar kullanarak bir tablodaki verileri diğerindeki verilerle ilişkilendirebilirsiniz. Yabancı anahtar, aynı tablodaki veya başka bir tablodaki birincil anahtara başvuran bir sütundur (veya bir dizi sütundur).

Bilgileri ayrı, yönetilebilir birimlerde düzenlemek için bir tablodaki verileri diğerindeki verilerle ilişkilendirme özelliğini kullanabilirsiniz. Çalışan verileri, ayrı bir tabloda saklanarak departman verilerinden mantıksal olarak ayrı tutulabilir.

Birincil Anahtarlar ve Yabancı Anahtarlar için Yönergeler

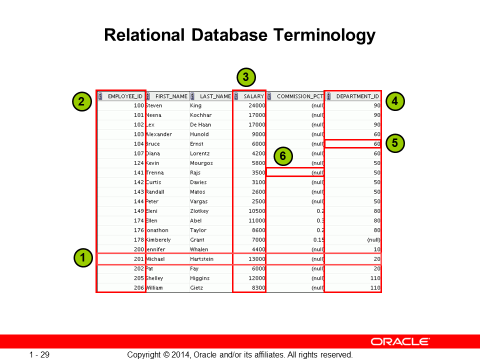
Birincil anahtarda yinelenen değerler kullanamazsınız. Yani unique olmalı

Birincil anahtarlar genellikle değiştirilemez.

Yabancı anahtarlar veri değerlerine dayanır ve tamamen mantıksal (fiziksel değil) işaretçilerdir.

Bir yabancı anahtar değeri, mevcut bir birincil anahtar değeriyle veya benzersiz anahtar değeriyle eşleşmelidir; aksi halde null olmalıdır.

Bir yabancı anahtar, bir birincil anahtara veya benzersiz bir anahtar sütununa başvurmalıdır.



İlişkisel bir veritabanı bir veya daha fazla tablo içerebilir. Tablo, bir RDBMS'nin temel depolama yapısıdır. Bir tablo, çalışanlar, faturalar veya müşteriler gibi gerçek dünyadaki bir şey hakkında gerekli tüm verileri tutar.

Slayt, ÇALIŞANLAR tablosunun veya ilişkisinin içeriğini gösterir. Rakamlar şunları gösterir:

1. Belirli bir çalışan için gereken tüm verileri temsil eden tek bir satır(Row) (veya demet). Tablodaki her satır, yinelenen satırlara izin vermeyen bir birincil anahtarla tanımlanmalıdır. Satırların sırası önemsizdir; veriler alındığında satır sırasını belirtin.

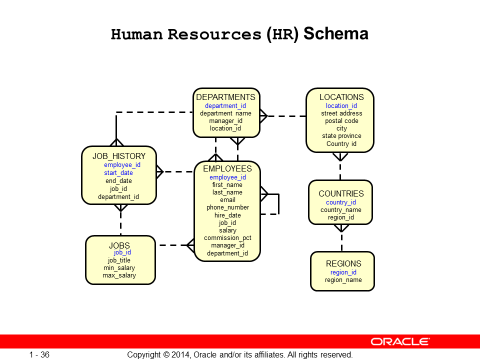
2. Çalışan numarasını içeren bir sütun(column) veya nitelik. Çalışan numarası, ÇALIŞANLAR tablosunda benzersiz bir çalışanı tanımlar. Bu örnekte, çalışan numarası sütunu birincil anahtar olarak belirlenmiştir. Birincil anahtar bir değer içermeli ve değer benzersiz olmalıdır.

3. Anahtar değer olmayan bir sütun. Bir sütun, bir tablodaki bir tür veriyi temsil eder; bu örnekte, veriler tüm çalışanların maaşlarıdır. Veri depolanırken sütun sırası önemsizdir; veriler alındığında sütun sırasını belirtin

4. Aynı zamanda bir yabancı anahtar olan departman numarasını içeren bir sütun. Yabancı anahtar, tabloların birbiriyle nasıl ilişkili olduğunu tanımlayan bir sütundur. Yabancı anahtar, aynı tablodaki veya başka bir tablodaki birincil anahtarı veya benzersiz bir anahtarı ifade eder. Örnekte, DEPARTMENT\_ID, DEPARTMENTS tablosundaki bir departmanı benzersiz olarak tanımlar.

5. Bir satır ve bir sütunun kesiştiği yerde bir alan bulunabilir. İçinde sadece bir değer olabilir.

6. Bir alanın içinde değer olmayabilir. Buna boş değer denir. ÇALIŞANLAR tablosunda COMMISSION\_PCT (komisyon) alanında sadece satış temsilcisi rolünde olan çalışanların değeri vardır.



İnsan Kaynakları (İK) Şeması Açıklama

İnsan Kaynakları (İK) şeması, Oracle Veritabanına kurulabilen Oracle Örnek Şemalarının bir parçasıdır. Bu kurstaki uygulama oturumları, İK şemasındaki verileri kullanır.

Tablo Açıklamaları

REGIONS, Amerika, Asya vb. gibi bir bölgeyi temsil eden satırları içerir.

COUNTRIES, her biri bir bölgeyle ilişkilendirilmiş ülkeler için satırlar içerir.

YERLER, belirli bir ülkedeki bir şirketin belirli bir ofisinin, deposunun veya üretim sahasının belirli adresini içerir.

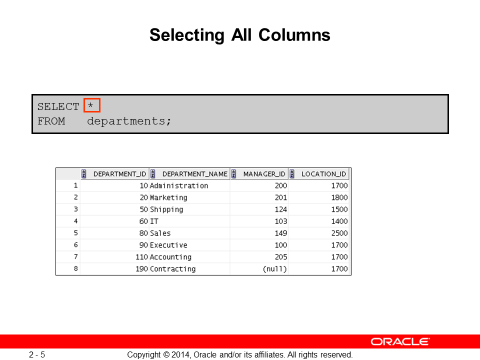
DEPARTMENTS, çalışanların çalıştığı departmanlarla ilgili ayrıntıları gösterir. Her departmanın ÇALIŞANLAR tablosunda departman yöneticisini temsil eden bir ilişkisi olabilir.

ÇALIŞANLAR, bir departmanda çalışan her bir çalışanın ayrıntılarını içerir. Bazı çalışanlar herhangi bir departmana atanamayabilir.

JOBS, her çalışanın yapabileceği iş türlerini içerir.

JOB\_HISTORY, çalışanların iş geçmişini içerir. Bir çalışan bir iş içinde departman değiştirirse veya bir departman içinde iş değiştirirse, bu tabloya çalışanın önceki iş bilgileriyle birlikte yeni bir satır eklenir.

SELECT sorgusu



yıldız işaretiyle (\*) SELECT anahtar sözcüğünü izleyerek bir tablodaki tüm veri sütunlarını görüntüleyebilirsiniz. Slayttaki örnekte, DEPARTMENTS tablosu dört sütun içerir: DEPARTMENT\_ID, DEPARTMENT\_NAME, MANAGER\_ID ve LOCATION\_ID. Tablo, her departman için bir tane olmak üzere sekiz satır içerir.

Tablodaki tüm sütunları SELECT anahtar sözcüğünden sonra listeleyerek de görüntüleyebilirsiniz. Örneğin, aşağıdaki SQL ifadesi (slayttaki örnek gibi),

SELECT department\_id, department\_name, manager\_id, location\_id

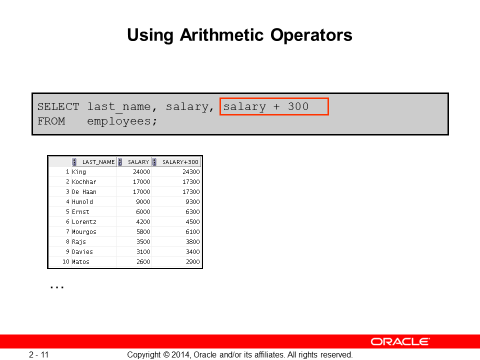
FROM departments**;**

SELECT deyimini, sütun adlarını virgülle ayırarak belirterek tablonun belirli sütunlarını görüntülemek için kullanabilirsiniz. Slayttaki örnek, BÖLÜMLER tablosundaki tüm departman numaralarını ve yer numaralarını görüntüler. SELECT yan tümcesinde, çıktıda görünmelerini istediğiniz sırayla istediğiniz sütunları belirtin. Örneğin, departman numarasından önce konumu görüntülemek için (soldan sağa), aşağıdaki ifadeyi kullanırsınız:

SELECT location\_id, department\_id

FROM departments

Sql yazarken genellikle anahtar kelimeler(SELECT FROM DELETE UPDATE VS) büyük harfle yazılır tablo adı sütün adı küçük harfle yazılır.kücük büyük harfe duyarlı değil yani tablo adım CITY olsun ben cıty dersemde hata vermez



Slayttaki örnek, tüm çalışanlar için 300$'lık bir maaş artışını hesaplamak için toplama operatörünü kullanır. Slayt ayrıca çıktıda MAAŞ+300 sütununu görüntüler.

Sonuç olarak hesaplanan MAAŞ+300 sütununun ÇALIŞANLAR tablosunda yeni bir sütun olmadığına dikkat edin; sadece gösterim amaçlıdır. Varsayılan olarak, yeni bir sütunun adı, onu oluşturan hesaplamadan gelir - bu durumda maaş+300.

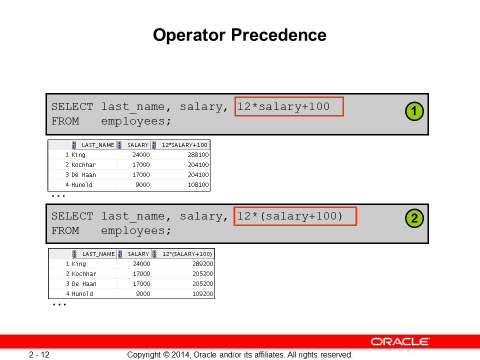
Not: Oracle sunucusu, aritmetik operatöründen önceki ve sonraki boşlukları yok sayar.

Öncelik Kuralları

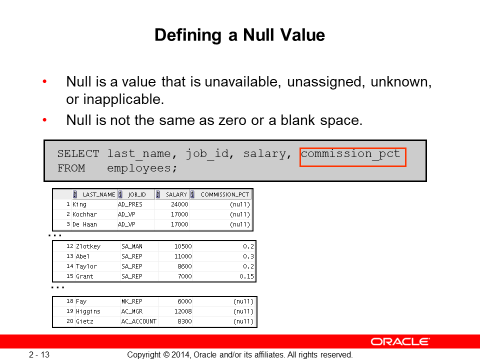
Çarpma ve bölme, toplama ve çıkarmadan önce gerçekleşir.

Aynı önceliğe sahip operatörler soldan sağa doğru değerlendirilir.

Parantezler, varsayılan önceliği geçersiz kılmak veya ifadeyi netleştirmek için kullanılır.



Parantez kullanarak istediğim gibi işlem önceliğini değiştirebiliriz



NULL değeri tanımlama

Null, kullanılamayan, atanmamış, bilinmeyen veya uygulanamayan bir değerdir.

Null, sıfır veya boşlukla aynı şey değildir.

Null değerler matematiksel bir işleme girerse sonuç null olur

Yukarıda görüldüğü gibi commission \_pct değerlerinden bazıları null ben şöyle bir sorgu yazsam

Select fırst\_name,salary,salary\*12\*commission\_pct from employees as prim

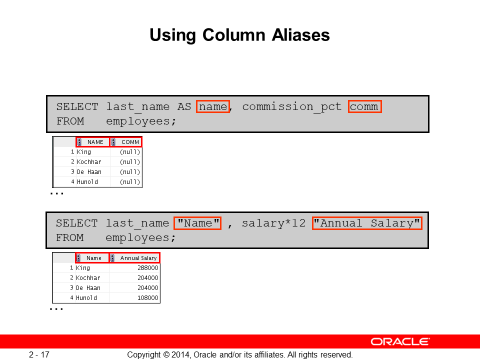
Çıktıda commission\_pct değerleri nuıll olanlar için prim kolonunda değerler “–“ olur commision\_pct değeri null olmayanlar için hesaplanan değer olur

Bir satırda belirli bir sütun için bir veri değeri yoksa, bu değerin NULL olduğu veya bir boş içerdiği söylenir.

NULL değerine sahip sütunlar bir SELECT sorgusunda seçilebilir ve bir aritmetik ifadenin parçası olabilir. NULL değerleri kullanan herhangi bir aritmetik ifade NULL ile sonuçlanır.

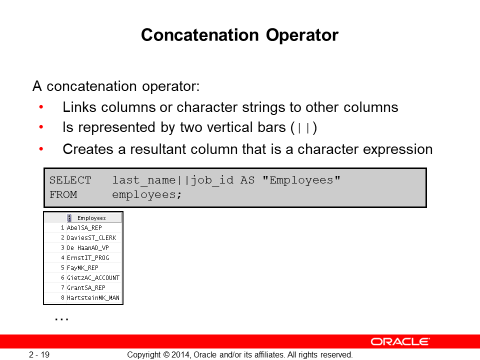
Herhangi bir veri türünün sütunları boş değerler içerebilir. Ancak, bazı kısıtlamalar (not null ve Primary Key) sütunda boş değerlerin kullanılmasını engeller.

Slayt örneğinde, ÇALIŞANLAR tablosunun COMMISSION\_PCT sütununda yalnızca bir satış yöneticisinin veya satış temsilcisinin komisyon kazanabileceğine dikkat edin. Diğer çalışanlar komisyon kazanma hakkına sahip değildir. Bir null bu gerçeği temsil eder.

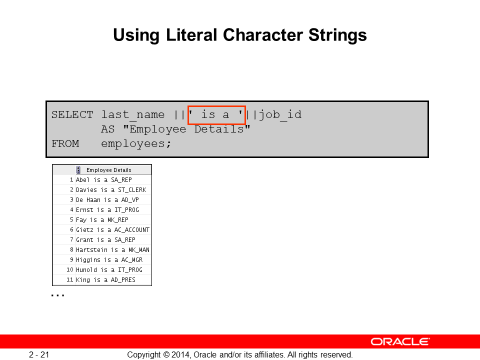


İlk örnekte as ile çıktıda gözükecek sütün adını değiştiriyoruz name yazmamıza rağmen çıktıda büyük harflerle NAME yazıyor çünkü sütün adları çıktıda varsayılan olarak büyük harfle gösteriliyor

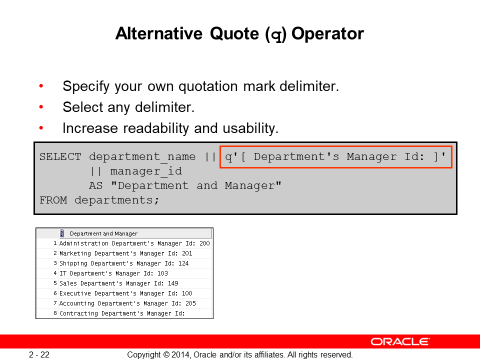
2.örnekte as değil de “ Name“ kullandık buda tırnağın içinde ne yazıyorsa olduğu gibi gösteriyor



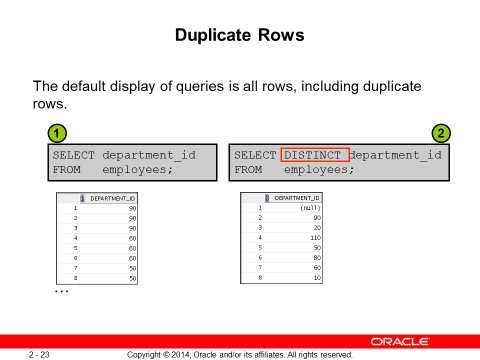
Concatenation karakteri(||) sütünları birleştirir

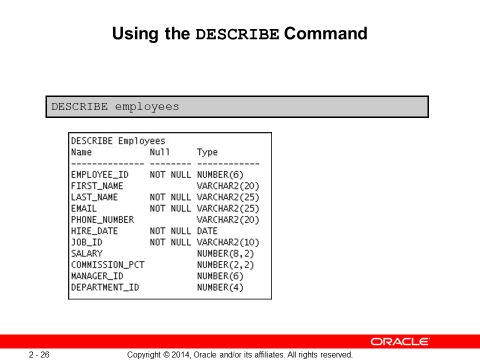


Bu şekilde daha iyi oldu



Bunu önceki örnekteki gibi tek tırnakta yazamadım çünkü ifadede bir tırnak var zaten tek tırnakla yazsaydım Departmen’s dediğimde tırnağı kapatacaktı kısacası çıktıda satırda olan ifadede tırnak varsa q operatörünü kullancaz

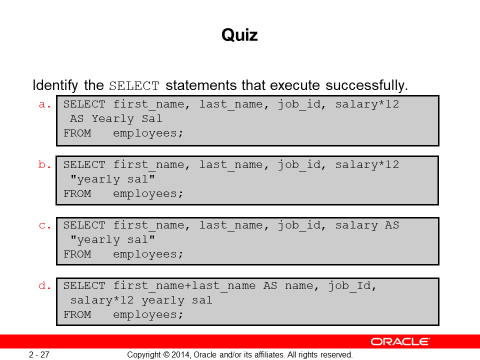


Yukarıda employees tablosundaki tüm department idleri getiriyo değerler aynı olsun olmasın hepsini getiriyo distinct koyduğumda tekrarlanan değerleri getirmiyor çalışanlar tablosunda aslında 20 kayıt var 1.de 20 kaydıda getiriyo 2.de tekrarlananları getirmiyor toplamda 7 farklı departman olduğu için 7 kayıt geliyor sayıları attım kafadan

Slayttaki örnek, DESCRIBE komutunu kullanarak ÇALIŞANLAR tablosunun yapısı hakkında bilgi görüntüler.

Ortaya çıkan ekranda Null, bu sütunun değerlerinin bilinmeyebileceğini belirtir. NOT NULL, bir sütunun veri içermesi gerektiğini belirtir. Tür, bir sütunun veri türünü görüntüler.

Veri türleri aşağıdaki tabloda açıklanmıştır:



Cevab b ve c

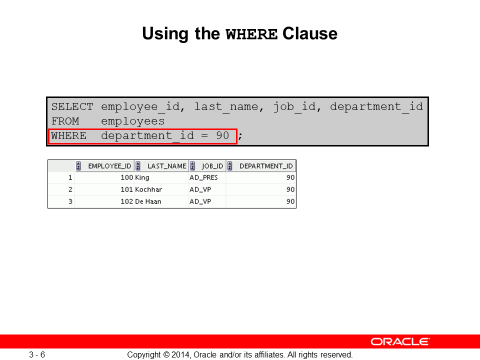
A şıkkında YEARLY SAL yazıyo arada boşuk olmaması gerekiyor

D şıkkı A şıkkıyla aynı sadece AS yok ama yine boşluk var boşluk odluğunda mutlaka “ “ tırnak içinde yazmamız gerekir AS yazsakta olur yazmasakta

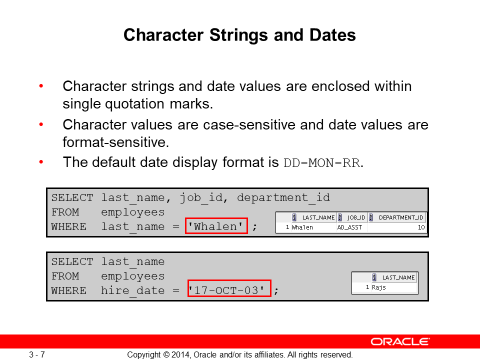
3.Slayt Veriyi kısıtlama ve sıralama

Verileri şunlarla kısıtlayabiliriz

* + - Where =, <=, BETWEEN, IN, LIKE, and NULL conditions
    - Logical conditions using AND, OR, and NOT operators



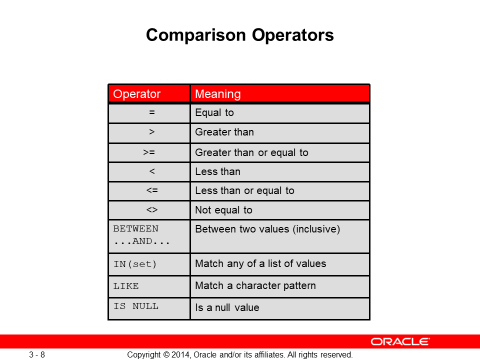
Where in olduğu yerde as kullanılmaz



Karakter stringleri ve date değerleri tek tırnak içinde yazılır küçük büyük harfe duyarlı neyse o olacak

Date için DD-MON-RR tipinde varsayılan olarak bu şekilde belirtmemiz gerekir

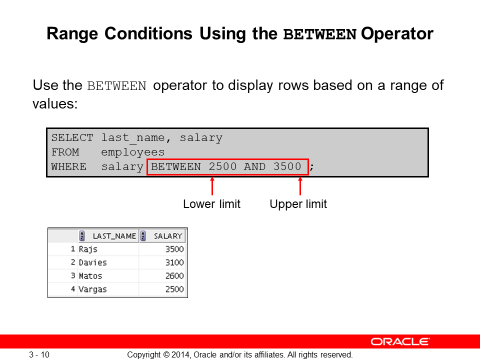
KARŞILAŞTIRMA OPERATÖRLERİ

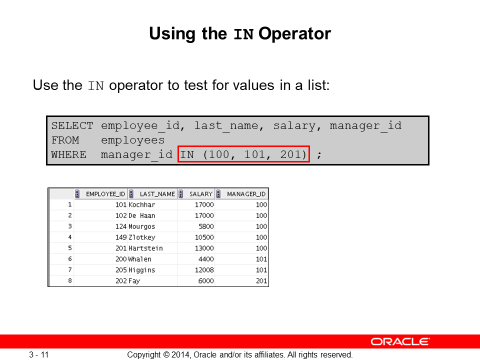


SELECT last\_name, salary

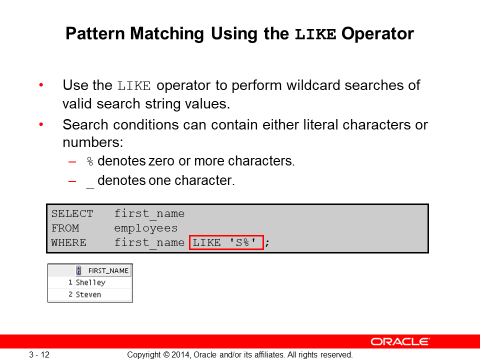
FROM employees

WHERE salary <= 3000 ;



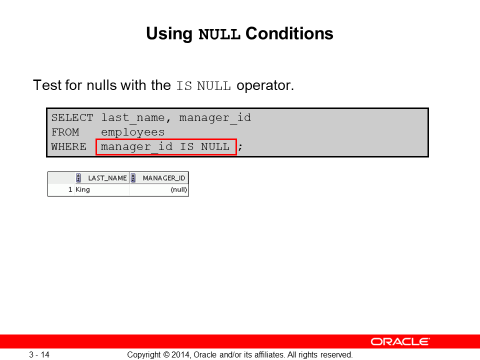


Manager id si 100 101 201 olan çalışanların istenilen bilgilerini getirir

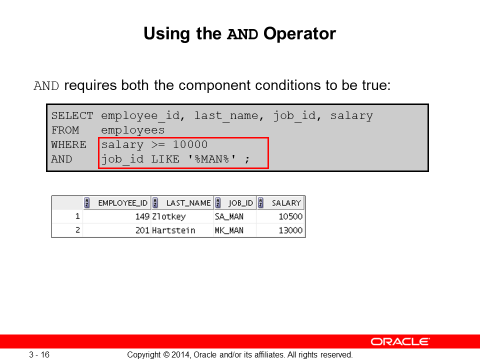


First\_name i S ile başlayanları getirir

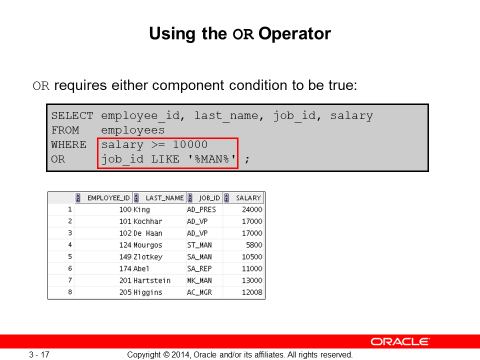
LIKE '\_o%' 2.harfi o ile başlayanları getirir

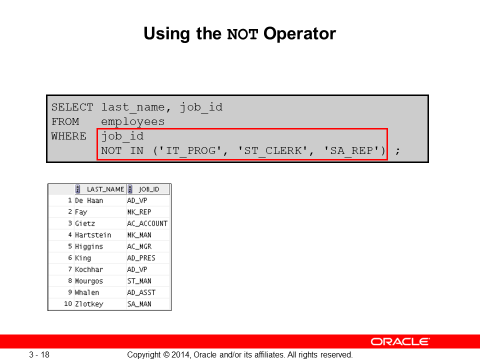


Manager id si null olanı getirir yani patronu

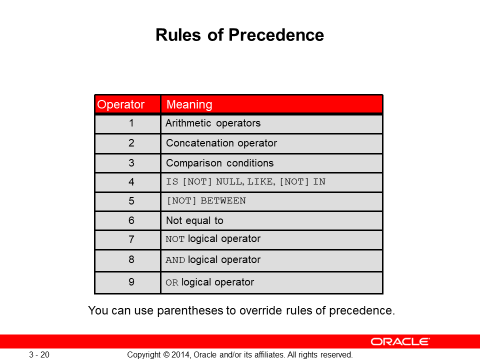


Salary si 10000 den ve job id sinde MAN içeren çalışanların istenilen bilgilerini getiriyo

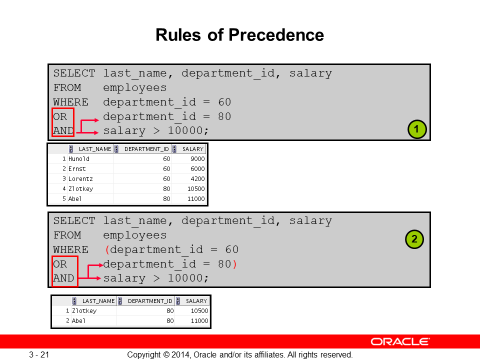


Maaşı 1000 den büyük tüm kayıtları ve job id sinde MAN içeren tüm kaytıları getirir veya her ikisine uyan tüm kayıtları getirir

Job id si IT PROG ST CLERK SA\_REP olmayan kayıtların last\_name ve job id sini getirir



Öncelik kuralları, ifadelerin değerlendirildiği ve hesaplandığı sırayı belirler. Slayttaki tablo, varsayılan öncelik sırasını listeler. Ancak, önce hesaplamak istediğiniz ifadelerin etrafında parantezler kullanarak varsayılan sırayı geçersiz kılabilirsiniz.



1. AND Operatörünün Önceliği: Örnek

Bu örnekte iki koşul vardır:

İlk koşul, departman kimliğinin 80 olması ve maaşın 10.000 dolardan fazla olmasıdır.

İkinci koşul, departman kimliğinin 60 olmasıdır.

Bu nedenle, SELECT ifadesi aşağıdaki gibidir:

"Bir çalışanın departman kimliği 80 ise ve 10.000 dolardan fazla kazanıyorsa veya çalışanın departman kimliği 60 ise satırı seçin. Yani istediğimiz gibi değil ben departman id si 60 veya 80 olan ve 1000 dolardan fazla maaş alan kişileri istiyom bunun için parantez kullancam"

2. Parantez Kullanımı: Örnek

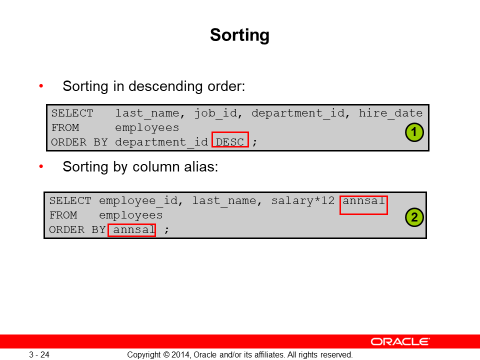
Bu örnekte iki koşul vardır:

İlk koşul, departman kimliğinin 80 veya 60 olmasıdır.

İkinci koşul, maaşın 10.000 dolardan fazla olmasıdır.

Bu nedenle, SELECT ifadesi aşağıdaki gibidir:

"Bir çalışanın departman kimliği 80 veya 60 ise ve çalışan 10.000 dolardan fazla kazanıyorsa satırı seçin."



Varsayılan olarak ascending olarak sıralıyo küçükten büyüğe yani tarihi de öyle 1992 1995 ten önce geliyor desc büyükten küçüğe

Varsayılan sıralama düzeni artandır:

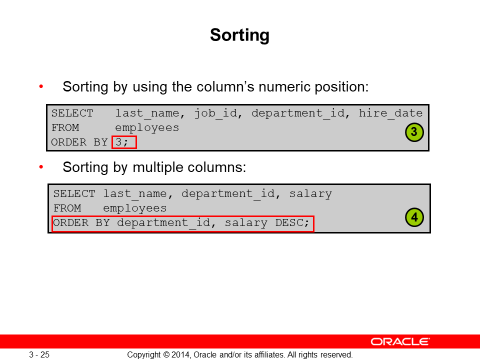
Sayısal değerler, önce en düşük değerlerle görüntülenir (örneğin, 1 ila 999).

Tarih değerleri, en erken değer en başta olacak şekilde görüntülenir (örneğin, 01-JAN-92, 01-JAN-95'ten önce).

Karakter değerleri alfabetik sırayla görüntülenir (örneğin, önce "A" ve son olarak "Z").

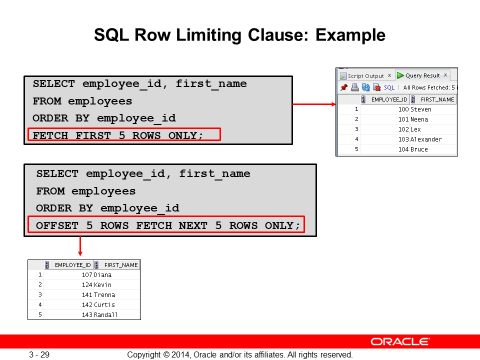
Null değerler, artan diziler için en son ve azalan diziler için ilk olarak görüntülenir.

Ayrıca, içinde olmayan bir sütuna göre de sıralayabilirsiniz.



1.de department\_id ye göre sıralıcak ve belirtmediğim için default olarak asc yani artan sırada sıralıcak

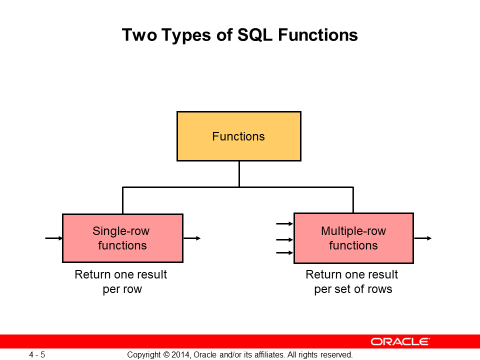
2.de önce department id ye göre asc olarak sıralıcak sonra maaşa göre büyüğünü önce yazcak yani department id si 110 olan 2 çalışan olsun hangisinin maaşı daha büyükse önce o yazılcak



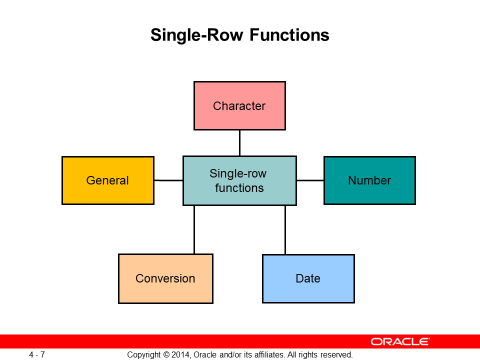
İlk kod örneği, en düşük employee\_id sahip ilk beş çalışanı döndürür.

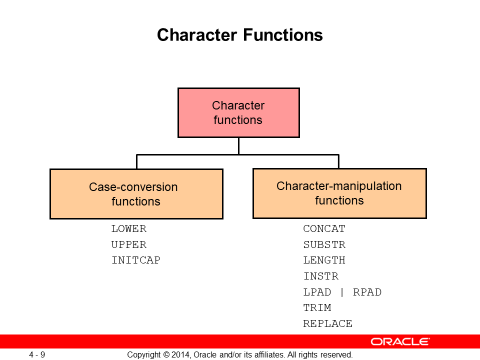
İkinci kod örneği, bir sonraki en düşük çalışan\_kimliği kümesine sahip beş çalışanı döndürür. Yani ilk 5 ten sonraki 5 satırı getirir.

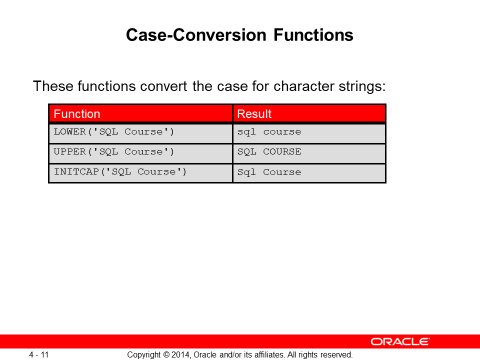
Not: Çalışan\_kimliği, çalışanın kuruluşa katıldığı tarihe göre sırayla atanırsa, bu örnekler bize kıdem açısından ilk 5 çalışanı ve ardından 6-10 arası çalışanı verir.



2 tip sql fonksiyonları var



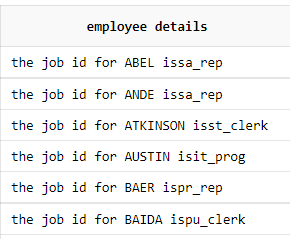


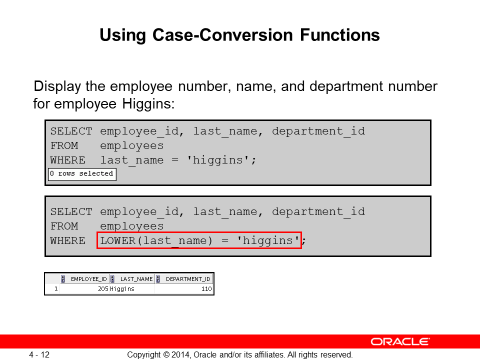


SELECT 'The job id for '||UPPER(last\_name)||' is '

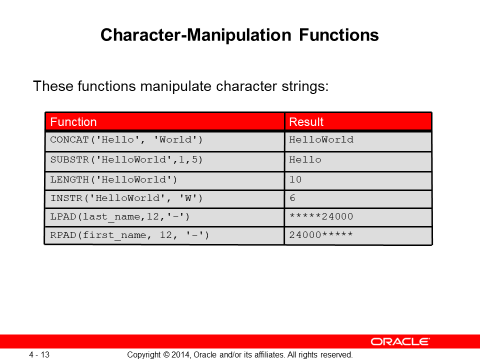
||LOWER(job\_id) AS "EMPLOYEE DETAILS"

FROM employees;



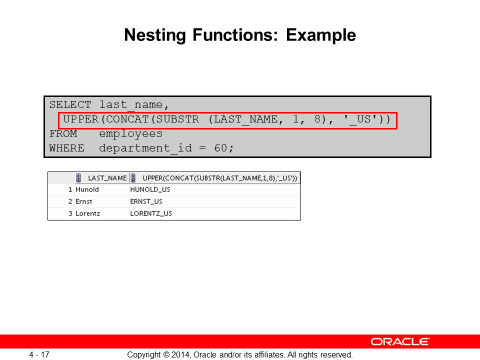


İlk örnekte higgins olarak aratıyom ama kayıt yapılırken Higgins olara kayıt yapılmış ve bu yüzden bir kayıt getirmiyor bu yüzden 2. Örnekteki gibi soy ismi LOWER fonksiyonu kullanarak küçük harfe çeviriyom artık where şartında küçük harfle yazarak kaydı getirebilirim

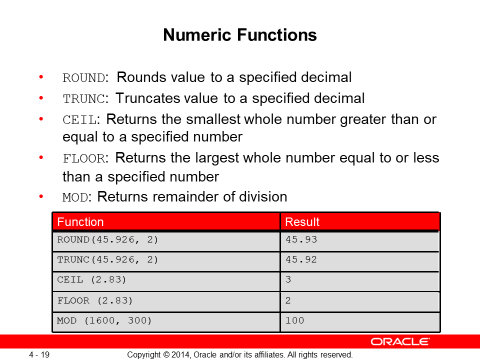


Burdada karakter yönetme fonksiyonları LPAD : Bir karakter ifadesi ile n karakter uzunluğunda sola dolgulu bir ifade döndürür

Rpad de sağa doğru



İç içe fonksiyona örnek lastname in ilk 8 karakterini alır(last name 8 den az karakterden oluşuyorsa atıyom 5 karakterli o zaman 5 karakteri de alır) sonrada onu \_US ile birleştirir sonrada hepsini büyük harfe çevirir 3 fonksiyon 1 arada Result1 = SUBSTR (LAST\_NAME, 1, 8) Result2 = CONCAT(Result1, '\_US') 3. Upper



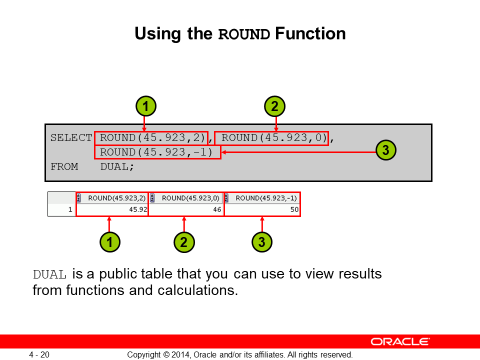
Burda da sayı fonksiyonları var

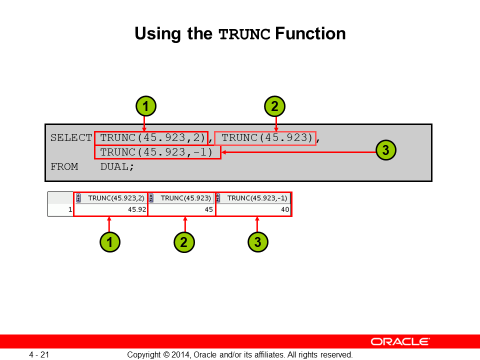
Round :değeri belirtilen ondalık sayıya yuvarlar

Trunc:değeri belirtilen ondalık basmağı keser.

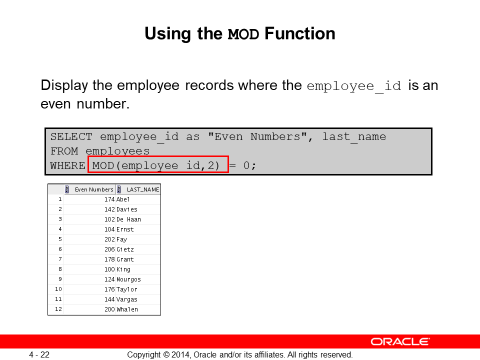
Ceıl:belirtilen ifadeden büyük veya eşit en küçük tam sayıyı verir.

FLOOR: belirtilen ifadeden küçük veya eşit en en büyük tam sayıyı verir.

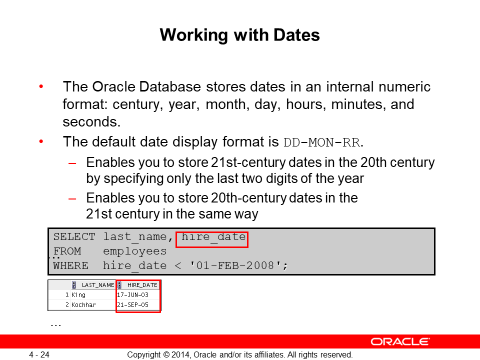




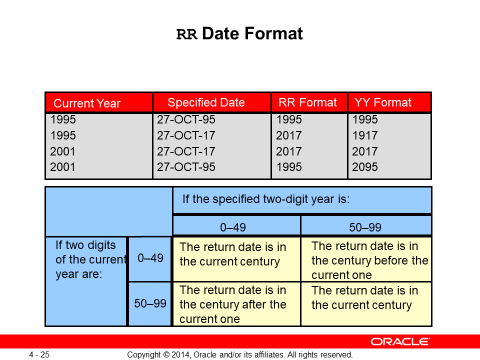
Dual tablosu sql de hesaplamalar ve fonksiyonlar için kullanabileceğimiz bir public table

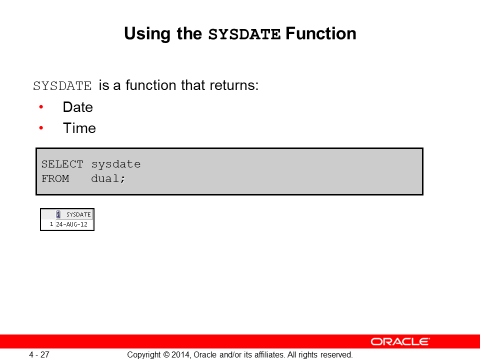


Employee id si çift olan çalışanların istenilen bilgilerini getirir.

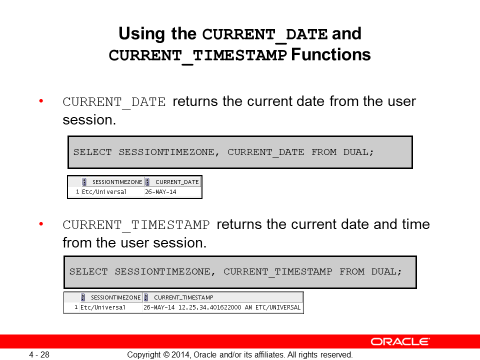


2008 den önce işe başlayan çalışanların istenilen bilgilerini getirir.





SYSDATE fonksiyonu anlık olarak tarihi döndürür

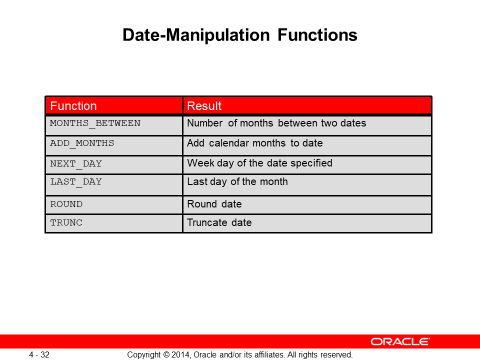


CURRENT\_DATE anlık tarihi user session dan alır kullanıcının sistemdeki saati.

CURRENT\_TIMESTAMP te hem tarih hem saati hemde sessionu döndürür.



Aritmetik işlemler yapabiliriz date fonksiyonları ile sistem tarihinden (kodu yazdığımız tarih) çalışanının işe başladığı tarihi çıkarıp(bu işlem bize gün sayısını vercek) 7 ye böldüğümüzde çalışanın kaç haftadır çalıştığını bulabiliriz dept idsi 90 olanlar için



**MONTHS\_BETWEEN(*date1, date2*): formatı bu şekilde 2 tarih arasındaki ay sayısını verir sonuç negatifte olabilir pozitifte date1 date 2 den önce geliyosa negative**

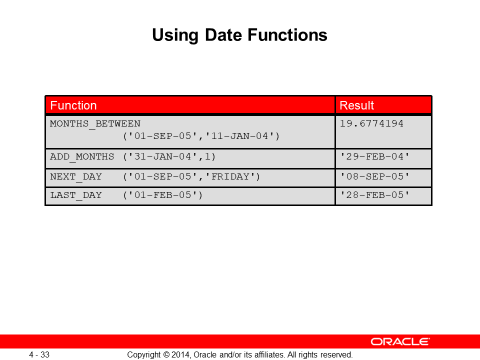
**ADD\_MONTHS(*date, n*): formatı bu şekilde belirtilen tarihin üstüne n defa ay ekliyo n negative olabilir n integer olmalı**

**NEXT\_DAY(*date,* '*char*'): formatı bu şekilde atıyom char yerine Friday yazdım bir tarih verdim o tarihten sonar en yakın Cuma gününün tarihini verir .**

**LAST\_DAY(*date*): formatı böyle bir tarih vercem atyıom 13-OCT-21 diye bana bu ayın son tarihini varcek 31-OCT-21 diye.**

**ROUND(*date*[,'*fmt*']):belirtilen tarihi en yakın tarihe yuvarlar.**

**TRUNC(*date*[, '*fmt*']):**

****

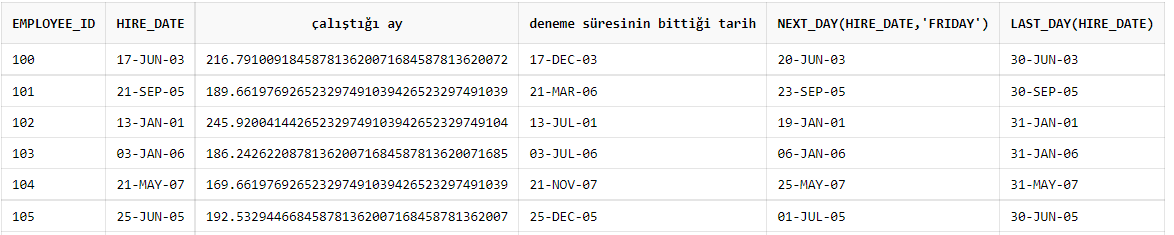
**Kullanımı böyle**

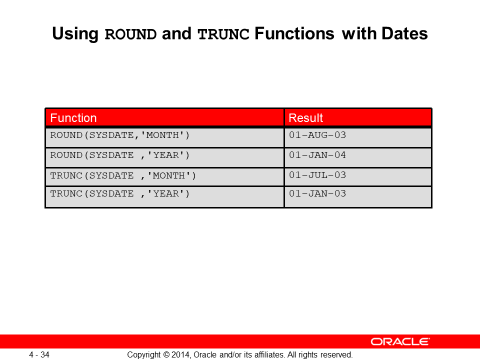
**ÖRNEK:**

**150 aydan fazla çalışan kişilerin employee\_id sini hire date ini kaç aydır çalıştığını hire date den sonraki 6.ayını (yani deneme süresi) hire date ten sonraki ilk Cuma gününün tarihini hire date in son tarihini getir.**

**SELECT employee\_id ,hire\_date,MONTHS\_BETWEEN(SYSDATE,hire\_date) "çalıştığı ay",ADD\_MONTHS(hire\_date,6) "deneme süresinin bittiği tarih",NEXT\_DAY(hire\_date,'FRIDAY'),LAST\_DAY(hire\_date)**

**FROM hr. employees WHERE MONTHS\_BETWEEN(SYSDATE,hire\_date)>150**

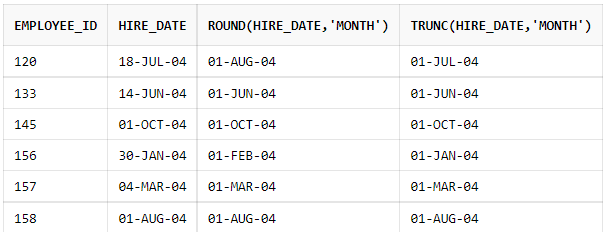


****

**Example: Compare the hire dates for all employees who started in 2004. Display the employee number, hire date, and starting month using the ROUND and TRUNC functions**

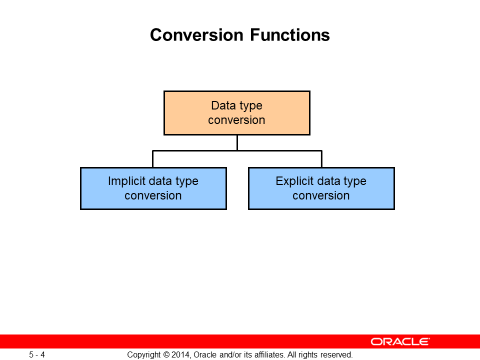
**SELECT employee\_id ,hire\_date,ROUND(hire\_date,'MONTH'),TRUNC(hire\_date,'MONTH') FROM hr.employees**

**WHERE hire\_date LIKE '%04'**

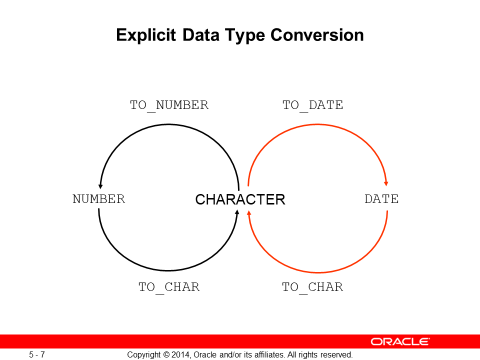


Çıktıda görüldüğü gibi Round month yaptığımızda belrtilen tarihe en yakın ayı veriyo (ayın ilk gününü tarih cinsinden)

Trunc month yaptığımızda verilen tarihin bulunduğu ayın ilk gününü veriyo

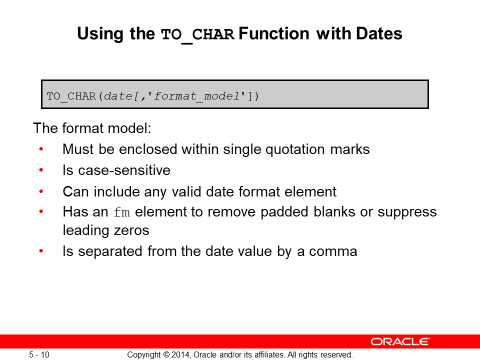


Veri tiplerini dönüştürme 2 şekilde implicit ve explicit



Varchar2 yada chardan numbere **,numberdan chara varchar2ye**

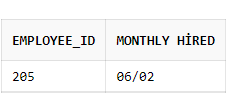
**Varchar2 yada chardan date e,date deb varchar2 ye yada chara dönüşüm yapılabilir**



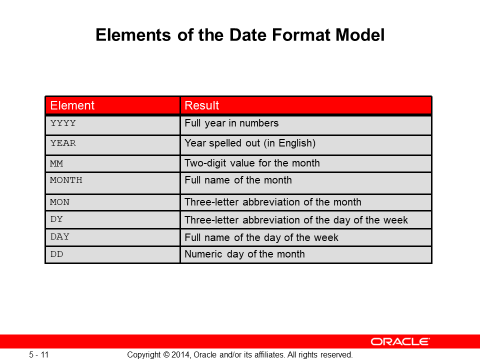
TO\_CHAR fonksiyonun formatı yukarıdaki gibi

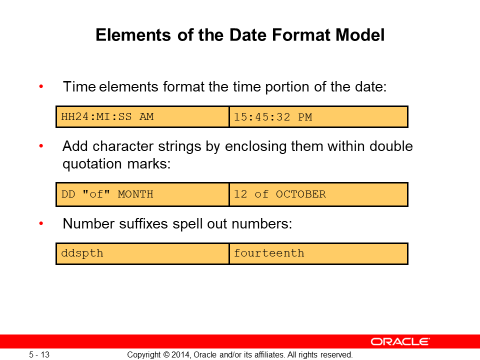
Biçim modeli tek tırnak içine alınmalıdır ve büyük/küçük harf duyarlıdır. Biçim modeli, herhangi bir geçerli tarih biçimi öğesini içerebilir. Ancak tarih değerini biçim modelinden virgülle ayırdığınızdan emin olun. Çıktıdaki gün ve ay adları otomatik olarak boşluklarla doldurulur. Dolgulu boşlukları kaldırmak veya baştaki sıfırları bastırmak için doldurma modu fm öğesini kullanın.

Örnek SELECT employee\_id,TO\_CHAR(hire\_date,'MM/YY') AS "MONTHLY HİRED" FROM hr.employees WHERE last\_name='Higgins'

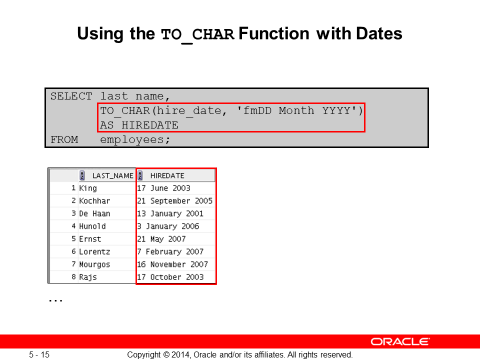


Görüldüğü gibi date formatını chara MM/YY ye çevirdi



Burdada fonksiyona yazcağımız parametreler. 

Örnek çıktılar



fm i yazmasam month dan sonra boşluk bırakıyo fm i yazarsam eşit aralıkla çıktı geliyor

**Modify the example in the slide to display the dates in a format that appears as “Seventeenth of June 2003 12:00:00 AM**

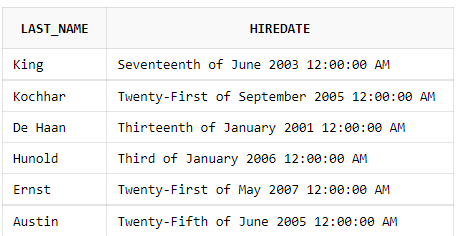
SELECT last\_name,

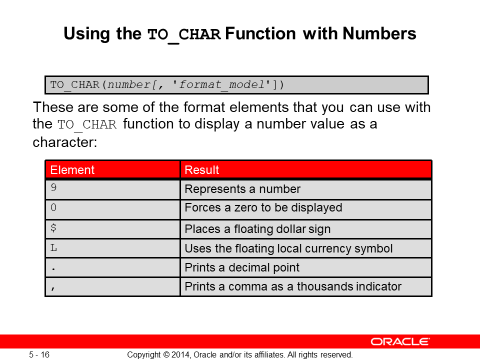
TO\_CHAR(hire\_date,

'fmDdspth "of" Month YYYY fmHH:MI:SS AM')

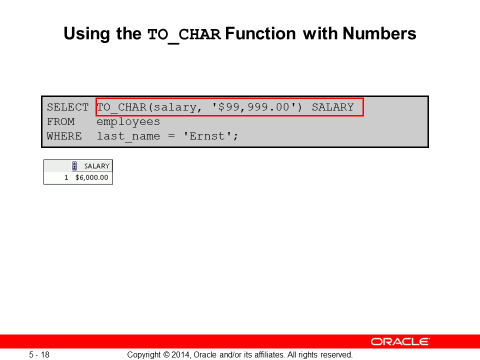
HIREDATE

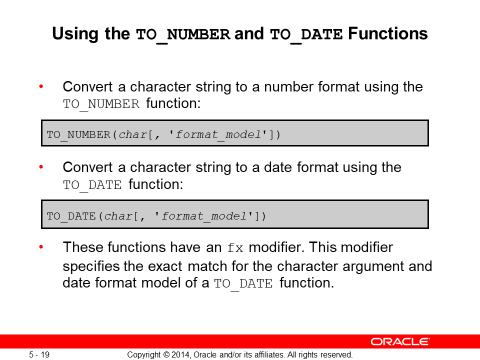
FROM employees;





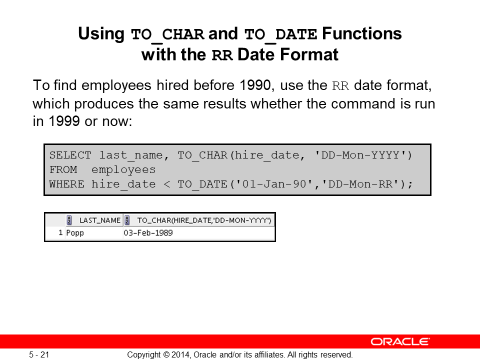
Şimdi TO\_CHAR fonksiyonu ile numberları chara dönüştürme

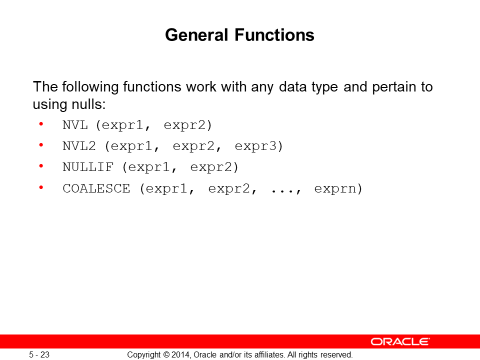


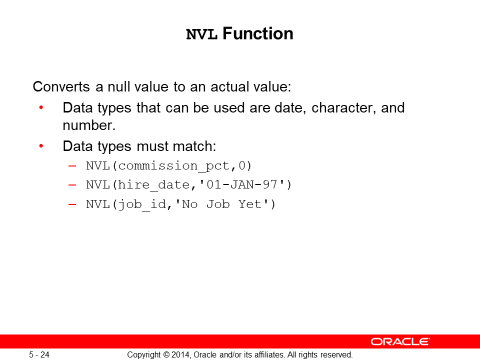


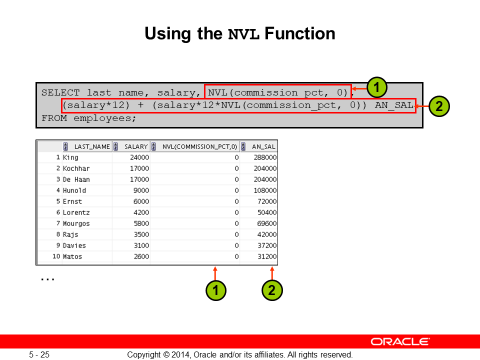
Şimdi charı number ve date veri tipine dönüştürme

1990'dan önce işe alınan çalışanları bulmak için, komut 1999'da veya şimdi çalıştırılsa da aynı sonuçları veren RR tarih biçimini kullanın:









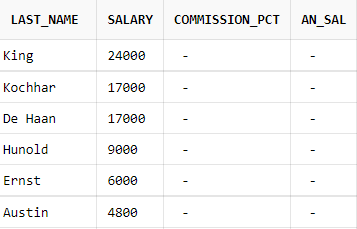
Yıllık tazminatın yalnızca komisyon kazanan çalışanlar için hesaplandığına dikkat edin. Bir ifadedeki herhangi bir sütun değeri null ise, sonuç null olur. Tüm çalışanların değerlerini hesaplamak için, aritmetik operatörü uygulamadan önce boş değeri bir sayıya dönüştürmeniz gerekir. Slayttaki örnekte, boş değerleri sıfıra dönüştürmek için NVL işlevi kullanılır.

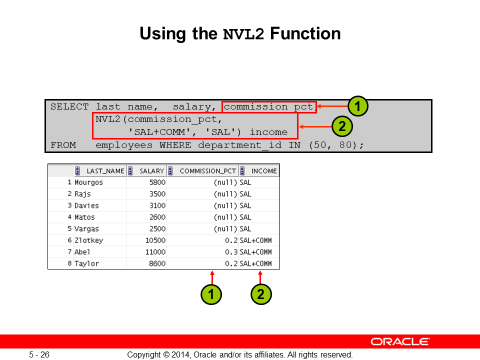
Tüm çalışanların yıllık ücretlerini hesaplamak için aylık maaşı 12 ile çarpmanız ve ardından komisyon yüzdesini sonuca eklemeniz gerekir:

SELECT last\_name, salary, commission\_pct,

(salary\*12) + (salary\*12\*commission\_pct) AN\_SAL

FROM employees;





NVL2 işlevi ilk ifadeyi inceler. İlk ifade boş değilse, NVL2 işlevi ikinci ifadeyi döndürür. İlk ifade null ise, üçüncü ifade döndürülür.

Sözdizimi

NVL2(ifade1, ifade2, ifade3)

Sözdiziminde:

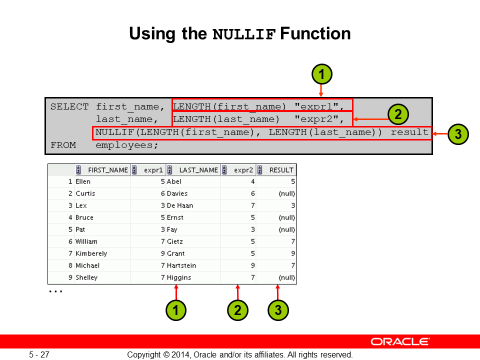
ifade1, boş değer içerebilecek kaynak değer veya ifadedir.

ifade2, ifade1 boş değilse döndürülen değerdir

ifade3, ifade1 boşsa döndürülen değerdir

Slaytta gösterilen örnekte COMMISSION\_PCT sütunu incelenmektedir. Bir değer algılanırsa, SAL+COMM'un metin değişmez değeri döndürülür. COMMISSION\_PCT sütunu boş bir değer içeriyorsa, SAL'ın metin değişmez değeri döndürülür.

Not: İfade1 bağımsız değişkeni herhangi bir veri türü olabilir, ancak ifade2 ve ifade3 aynı veri türü olmalıdır.



NULLIF işlevi iki ifadeyi karşılaştırır.

Sözdizimi

NULLIF (ifade1, ifade2)

Sözdiziminde:

NULLIF, ifade1 ve ifade2'yi karşılaştırır. Eşitlerse, işlev null değerini döndürür. Değillerse, işlev ifade1'i döndürür. Ancak, ifade1 için değişmez değeri NULL belirtemezsiniz.

Slaytta gösterilen örnekte ÇALIŞANLAR tablosundaki ilk adın uzunluğu ÇALIŞANLAR tablosundaki soyadının uzunluğu ile karşılaştırılır. Adların uzunlukları eşit olduğunda, boş bir değer görüntülenir. Adların uzunlukları eşit olmadığında, ilk adın uzunluğu görüntülenir.

**Using the COALESCE Function:**

COALESCE işlevi, listedeki ilk boş olmayan ifadeyi döndürür.

Sözdizimi

COALESCE (ifade1, ifade2, ... ifade)

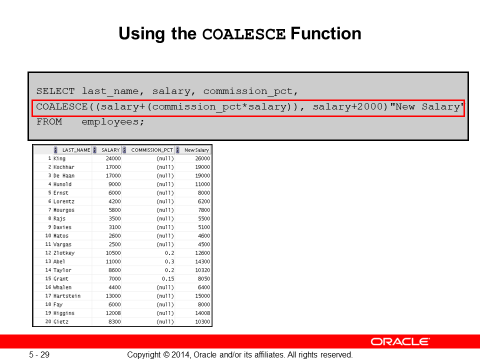
Sözdiziminde:

ifade1, boş değilse bu ifadeyi döndürür

ifade2, ilk ifade boşsa ve bu ifade boş değilse bu ifadeyi döndürür

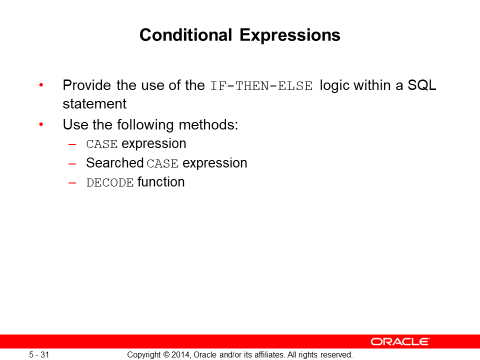
exprn, önceki ifadeler boşsa bu ifadeyi döndürür

Tüm ifadelerin aynı veri türünde olması gerektiğini unutmayın.



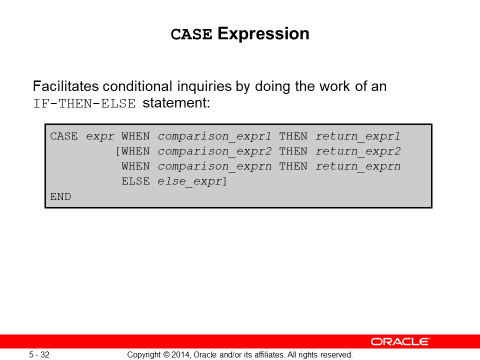
Slayttaki örnekte komisyon almayan çalışanlarınız için 2.000 TL maaş zammı vermek istediğinizde, komisyon alan çalışanlarınız için sorguda mevcut eklenen maaşa eşit yeni maaş hesaplanmalıdır. komisyon tutarına.

Not: Çıktıyı inceleyin. Yeni Maaş sütununda komisyon almayan çalışanlar için 2.000 TL artırılan maaş, komisyon alan çalışanlar için Yeni Maaş sütunu maaşa eklenen hesaplanan komisyon tutarını gösterir.



Bir SQL deyiminde koşullu işlemeyi (IF-THEN-ELSE mantığı) uygulamak için kullanılan iki yöntem CASE ifadesi ve DECODE işlevidir.

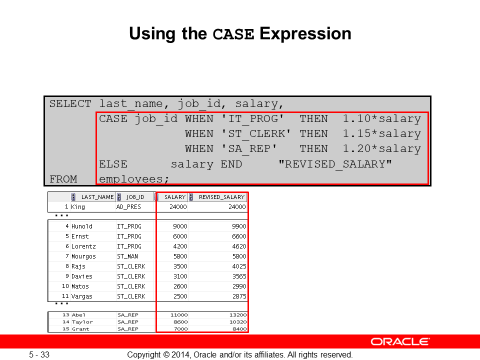
Not: CASE ifadesi ANSI SQL ile uyumludur. DECODE işlevi, Oracle sözdizimine özgüdür.

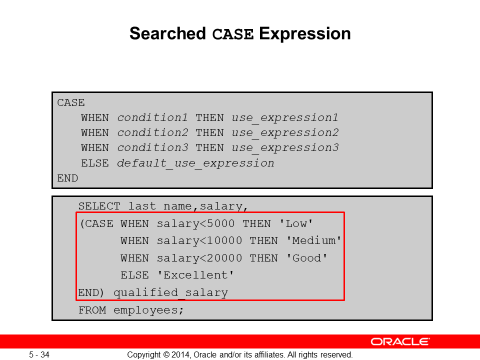


CASE ifadeleri, prosedürleri çağırmak zorunda kalmadan SQL deyimlerinde IF-THEN-ELSE mantığını kullanmanızı sağlar.

Basit bir CASE ifadesinde, Oracle sunucusu, ifadenin karşılaştırma\_ifadesine eşit olduğu ilk WHEN ... THEN çiftini arar ve dönüş\_ifadesini döndürür. WHEN ... THEN çiftlerinden hiçbiri bu koşulu karşılamıyorsa ve bir ELSE yan tümcesi varsa, Oracle sunucusu else\_expr döndürür. Aksi takdirde, Oracle sunucusu bir boş değer döndürür. Tüm dönüş\_ifadeleri ve başka\_ifade için NULL değişmez değerini belirtemezsiniz.

İfade ve karşılaştırma\_ifadesi, CHAR, VARCHAR2, NCHAR veya NVARCHAR2,NUMBER,BINARY\_FLOAT veya BINARY\_DOUBLE olabilen aynı veri türünde olmalı veya tümü sayısal bir veri türüne sahip olmalıdır. Tüm dönüş değerleri (dönüş\_ifadesi) aynı veri türünde olmalıdır.



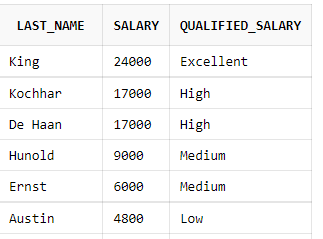
Slayttaki SQL deyiminde JOB\_ID değerinin kodu çözülür. JOB\_ID IT\_PROG ise maaş artışı %10'dur; JOB\_ID ST\_CLERK ise maaş artışı %15'tir; JOB\_ID SA\_REP ise maaş artışı %20'dir. Diğer tüm iş rolleri için maaşta artış yoktur. Aynı ifade DECODE işleviyle de yazılabilir. 

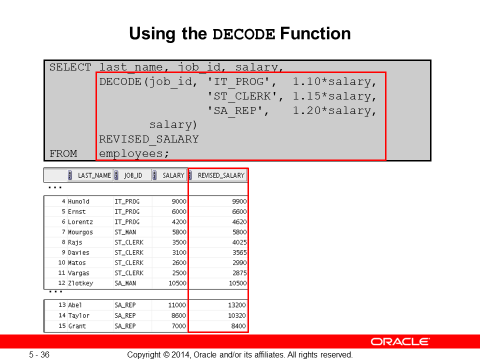
Aranan bir CASE ifadesinde, arama, listelenen koşulun bir tekrarı bulunana kadar soldan sağa yapılır ve ardından dönüş ifadesini döndürür. Hiçbir koşul doğru bulunamazsa ve bir ELSE yan tümcesi varsa, ELSE yan tümcesindeki dönüş ifadesi döndürülür; aksi takdirde, bir NULL döndürülür. Aranan CASE, WHEN seçeneklerinin her biri altındaki koşulları bağımsız olarak değerlendirir.

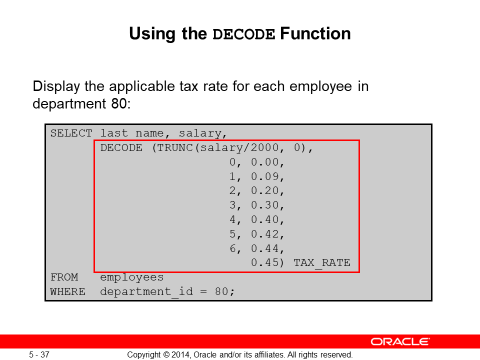
CASE ifadesi ile aranan CASE ifadesi arasındaki fark, aranan CASE ifadesinde, WHEN anahtar kelimesinden sonra bir karşılaştırma\_ifadesi yerine bir koşul veya yüklem belirtmenizdir.

Hem basit hem de aranan CASE ifadeleri için, tüm dönüş\_ifadeleri ya aynı veri tipine sahip olmalıdır CHAR, VARCHAR2, NCHAR veya NVARCHAR2,NUMBER,BINARY\_FLOAT veya BINARY\_DOUBLE veya hepsinin sayısal bir veri tipi olmalıdır.

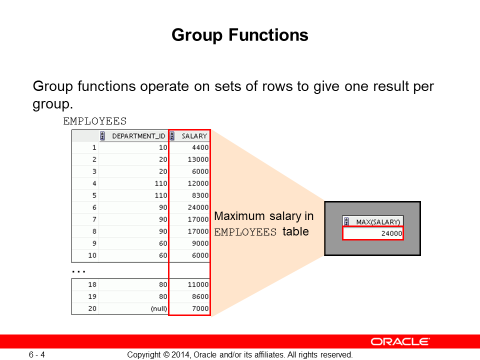
Slayttaki kod, aranan CASE ifadesinin bir örneğidir. Çıktısı böyle







GROUP FUNCTIONS



Tek sıralı işlevlerin aksine, grup işlevleri, grup başına bir sonuç verecek şekilde satır kümeleri üzerinde çalışır. Bu setler, tüm tabloyu veya gruplara ayrılmış tabloyu içerebilir.



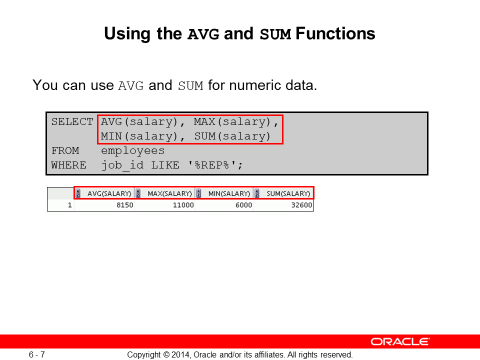


Grup işlevlerini kullanma yönergeleri:

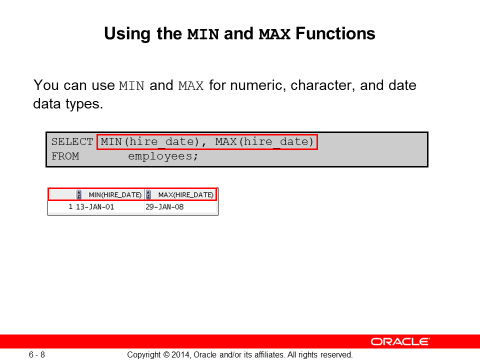
DISTINCT, işlevin yalnızca yinelenmeyen değerleri dikkate almasını sağlar; ALL, kopyalar da dahil olmak üzere her değeri dikkate almasını sağlar. Varsayılan ALL dur ve bu nedenle belirtilmesi gerekmez.

İfade bağımsız değişkeni olan işlevlerin veri türleri CHAR, VARCHAR2, NUMBER veya DATE olabilir.

Tüm grup işlevleri boş değerleri yok sayar. Boş değerlerin yerine bir değer koymak için NVL, NVL2, COALESCE, CASE veya DECODE işlevlerini kullanın.



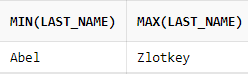
AVG, SUM, MIN ve MAX işlevlerini sayısal verileri depolayabilen sütunlara karşı kullanabilirsiniz. Slayttaki örnek, tüm satış temsilcileri için ortalama, en yüksek, en düşük ve aylık maaşların toplamını gösterir.

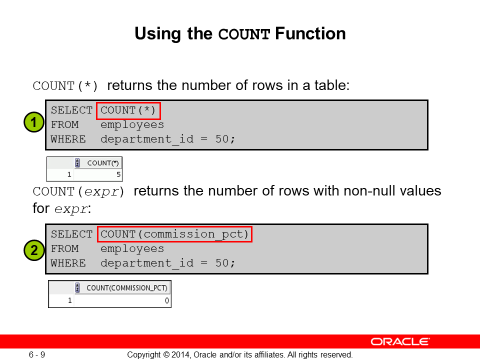


Sayısal, karakter ve tarih veri türleri için MAX ve MIN işlevlerini kullanabilirsiniz. Slayttaki örnek, en genç ve en kıdemli çalışanları gösterir. Aşağıdaki örnek, tüm çalışanların alfabetik bir listesinde en baştaki çalışan soyadını ve en sondaki çalışan soyadını gösterir:

SELECT MIN(last\_name), MAX(last\_name)

FROM employees;



Not: AVG, SUM, VARIANCE ve STDDEV işlevleri yalnızca sayısal veri türleri ile kullanılabilir. MAX ve MIN, LOB veya LONG veri türleri ile kullanılamaz

COUNT işlevinin üç biçimi vardır:

COUNT(\*)

COUNT(ifade)

COUNT(DISTINCT ifade)

COUNT(\*), sütunlardan herhangi birinde yinelenen satırlar ve boş değerler içeren satırlar dahil, SELECT ifadesinin ölçütlerini karşılayan bir tablodaki satır sayısını döndürür. SELECT deyiminde bir WHERE yan tümcesi varsa, COUNT(\*), WHERE yan tümcesindeki koşulu sağlayan satır sayısını döndürür.

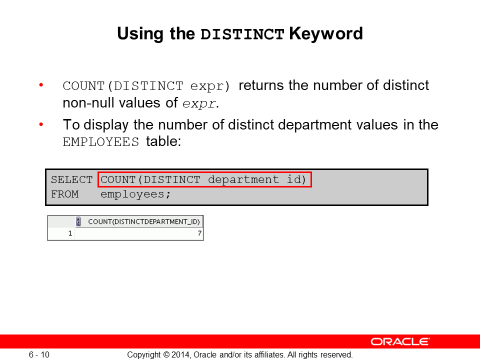
Buna karşılık, COUNT(ifade), ifade ile tanımlanan sütunda bulunan boş olmayan değerlerin sayısını döndürür.

COUNT(DISTINCT ifade), ifade ile tanımlanan sütunda bulunan benzersiz, boş olmayan değerlerin sayısını döndürür.

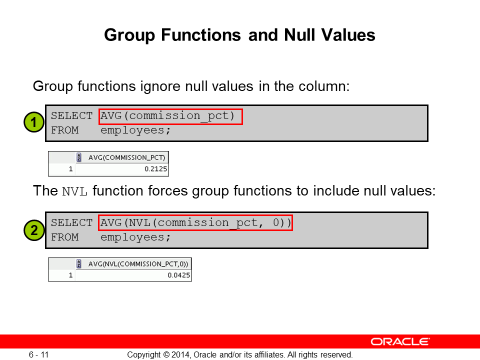
Örnekler

1. Slayttaki örnek, departman 50'deki çalışan sayısını göstermektedir.

2. Slayttaki örnek, 50 numaralı departmanda komisyon kazanabilecek çalışan sayısını göstermektedir.



Bir sütundaki yinelenen değerlerin sayımını bastırmak için DISTINCT anahtar sözcüğünü kullanın. Slayttaki örnek, EMPLOYEES tablosundaki farklı departman değerlerinin sayısını görüntüler.



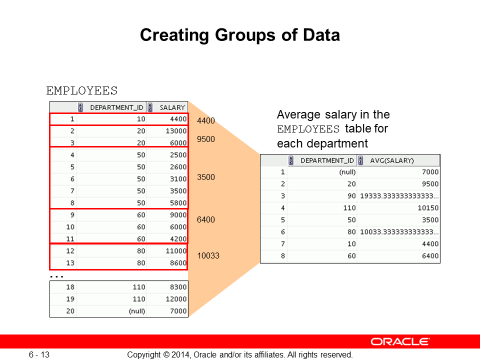
Tüm grup fonksiyonları, sütundaki null değerleri yok sayar.

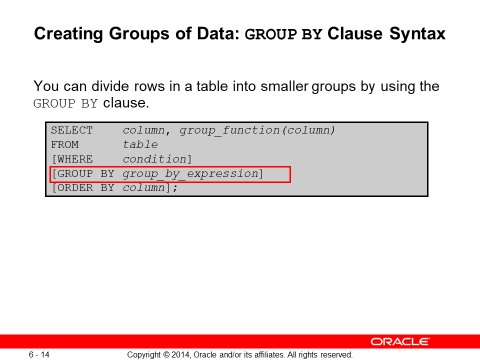
Ancak, NVL işlevi, grup fonksiyonlarına null değerleri eklemeye zorlar.

Örnekler

1. Ortalama, yalnızca COMMISSION\_PCT sütununda geçerli bir değerin depolandığı tablodaki satırlara göre (yani null olmayanlar için) hesaplanır. Ortalama, tüm çalışanlara ödenen toplam komisyonun, komisyon alan (dört) çalışan sayısına bölünmesiyle hesaplanır.

2. Ortalama, COMMISSION\_PCT sütununda boş değerler saklanıp saklanmadığından bağımsız olarak, tablodaki tüm satırlara göre hesaplanır. Ortalama, tüm çalışanlara ödenen toplam komisyonun şirketteki toplam çalışan sayısına (20) bölünmesiyle hesaplanır. ASLINDA benim istediğim doğru olan 2.kod





Bir tablodaki satırları gruplara ayırmak için GROUP BY yan tümcesini kullanabilirsiniz. Ardından, her grup için özet bilgi döndürmek için grup işlevlerini kullanabilirsiniz.

Sözdiziminde:

group\_by\_expression Değerleri satırları gruplamak için temel belirleyen sütunları belirtir

Yönergeler

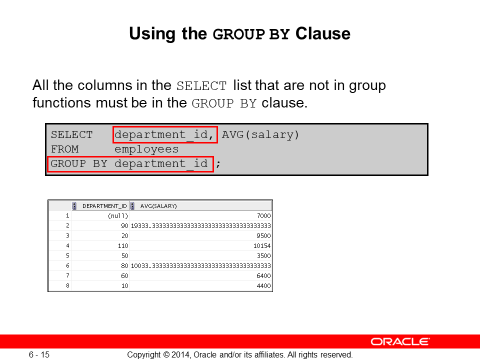
Bir SELECT yan tümcesine bir grup işlevi eklerseniz, bağımsız sütun GROUP BY yan tümcesinde görünmediği sürece tek tek sütunu da seçemezsiniz. GROUP BY yan tümcesine sütun listesini dahil etmezseniz bir hata iletisi alırsınız.

Bir WHERE yan tümcesi kullanarak, satırları gruplara ayırmadan önce hariç tutabilirsiniz.

SELECT deyiminde sütunu bir ifadeyle değiştirebilirsiniz.

GROUP BY yan tümcesindeki sütunları eklemelisiniz.

GROUP BY yan tümcesinde bir sütun diğer adı kullanamazsınız



**GROUP BY yan tümcesini kullanırken, SELECT listesindeki grup fonksiyonları olmayan tüm sütunların GROUP BY yan tümcesine dahil edildiğinden emin olun.** Slayttaki örnek, departman numarasını ve her departman için ortalama maaşı gösterir. GROUP BY yan tümcesi içeren bu SELECT deyimi şu şekilde değerlendirilir:

SELECT yan tümcesi, alınacak sütunları aşağıdaki gibi belirtir:

ÇALIŞANLAR tablosundaki departman numarası sütunu

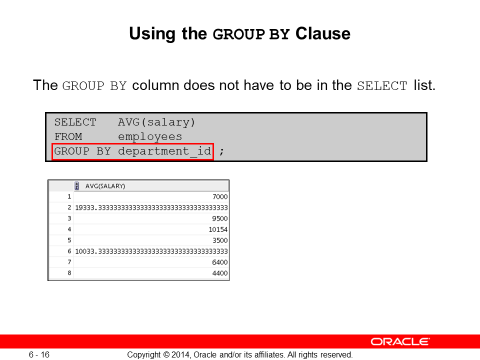
GROUP BY deyiminde belirttiğiniz gruptaki tüm maaşların ortalaması

FROM yan tümcesi, veritabanının erişmesi gereken tabloları belirtir: EMPLOYEES tablosu.

WHERE yan tümcesi, alınacak satırları belirtir. WHERE yan tümcesi olmadığından, tüm satırlar varsayılan olarak alınır.

GROUP BY yan tümcesi, satırların nasıl gruplanacağını belirtir. Satırlar departman numarasına göre gruplanmıştır, bu nedenle maaş sütununa uygulanan AVG işlevi, her departman için ortalama maaşı hesaplar.

Not: Sorgu sonuçlarını artan veya azalan düzende sıralamak için sorguya ORDER BY yan tümcesini ekleyin.



GROUP BY sütununun SELECT yan tümcesinde olması gerekmez. Örneğin, slayttaki SELECT ifadesi, ilgili departman numaralarını görüntülemeden her departman için ortalama maaşları görüntüler. Ancak departman numaraları olmadan sonuçlar anlamlı görünmüyor.

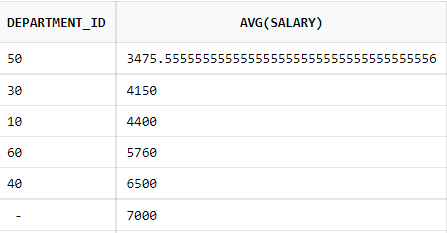
ORDER BY yan tümcesinde grup işlevini de kullanabilirsiniz:

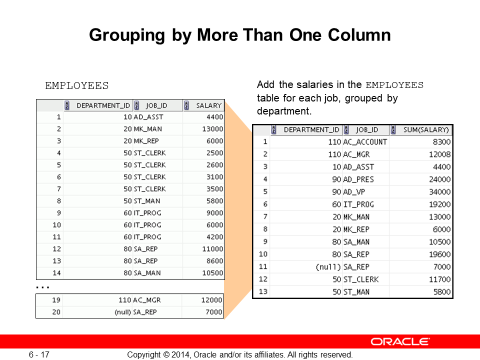
SELECT department\_id, AVG(salary)

FROM employees

GROUP BY department\_id

ORDER BY AVG(salary);





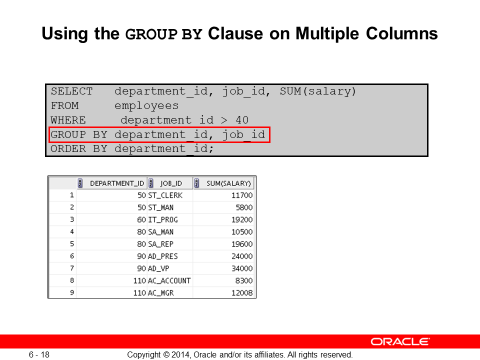
Bazen, gruplar içindeki gruplar için sonuçları görmeniz gerekir. Slayt, her departmandaki her bir iş unvanına ödenen toplam maaşı gösteren bir raporu gösterir.

ÇALIŞANLAR tablosu, önce departman numarasına, ardından o gruplama içindeki iş unvanına göre gruplandırılır. Örneğin, 50 numaralı departmandaki dört stok memuru birlikte gruplandırılır ve gruptaki tüm stok memurları için tek bir sonuç (toplam maaş) üretilir.

Aşağıdaki SELECT ifadesi, slaytta gösterilen sonucu döndürür

SELECT department\_id,job\_id, COUNT(job\_id),SUM(salary) FROM hr.employees GROUP BY department\_id,job\_id ORDER BY job\_id

COUNT(job id yi de bne ekledim bu sayede her job id de kaç kişi çalışıyo onu buluyoruz



Birden çok GROUP BY sütununu listeleyerek gruplar ve alt gruplar için özet sonuçları döndürebilirsiniz. GROUP BY yan tümcesi satırları gruplandırır ancak sonuç kümesinin sırasını garanti etmez. Gruplamaları sıralamak için ORDER BY yan tümcesini kullanın.

Slayttaki örnekte, GROUP BY yan tümcesi içeren SELECT ifadesi aşağıdaki gibi değerlendirilir:

SELECT yan tümcesi, alınacak sütunu belirtir:

ÇALIŞANLAR tablosundaki DEPARTMENT\_ID

ÇALIŞANLAR tablosundaki JOB\_ID

GROUP BY deyiminde belirttiğiniz gruptaki tüm maaşların toplamı

FROM yan tümcesi, veritabanının erişmesi gereken tabloları belirtir: EMPLOYEES tablosu.

WHERE yan tümcesi, sonuç kümesini DEPARTMENT\_ID'nin 40'tan büyük olduğu satırlara indirger.

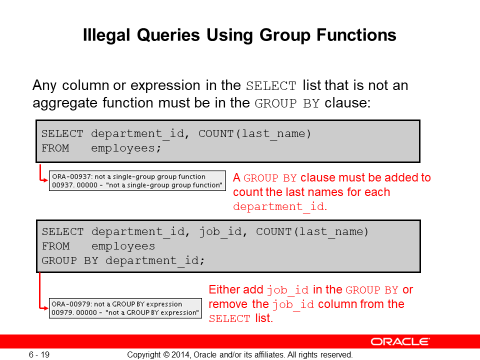
GROUP BY yan tümcesi, ortaya çıkan satırları nasıl gruplamanız gerektiğini belirtir:

İlk olarak, satırlar DEPARTMENT\_ID'ye göre gruplandırılır.

İkinci olarak, satırlar DEPARTMENT\_ID gruplarında JOB ID'ye göre gruplandırılır.

ORDER BY deyimi, sonuçları DEPARTMENT\_ID'ye göre sıralar.

Not: SUM işlevi, her DEPARTMENT\_ID grubundaki sonuç kümesindeki tüm iş kimlikleri için maaş sütununa uygulanır. Ayrıca SA\_REP satırının döndürülmediğini unutmayın. Bu satırın DEPARTMENT\_ID değeri NULL'dur ve bu nedenle WHERE koşulunu karşılamıyor.

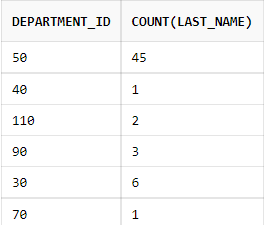


Aynı SELECT ifadesinde tek tek öğelerin (DEPARTMENT\_ID) ve grup işlevlerinin (COUNT) bir karışımını kullandığınızda, tek tek öğeleri (bu durumda DEPARTMENT\_ID) belirten bir GROUP BY yan tümcesi eklemelisiniz. GROUP BY yan tümcesi yoksa, "tek gruplu grup işlevi değil" hata mesajı görünür ve bir yıldız işareti (\*) sorunlu sütunu gösterir. GROUP BY yan tümcesini ekleyerek slayttaki ilk örnekteki hatayı düzeltebilirsiniz:

SELECT department\_id, count(last\_name)

FROM employees

GROUP BY department\_id;

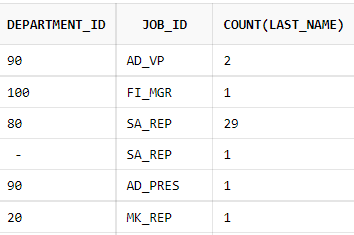


SELECT listesindeki bir toplama işlevi olmayan herhangi bir sütun veya ifade GROUP BY yan tümcesinde olmalıdır. Slayttaki ikinci örnekte, JOB\_ID ne GROUP BY deyiminde ne de bir grup işlevi tarafından kullanılıyor, bu nedenle “GROUP BY ifadesi değil” hatası var. İkinci slayt örneğindeki hatayı GROUP BY yan tümcesine JOB\_ID ekleyerek düzeltebilirsiniz. Aynı zamanda burda her departmanda kaç kişi çalışıyo onu da bulduk

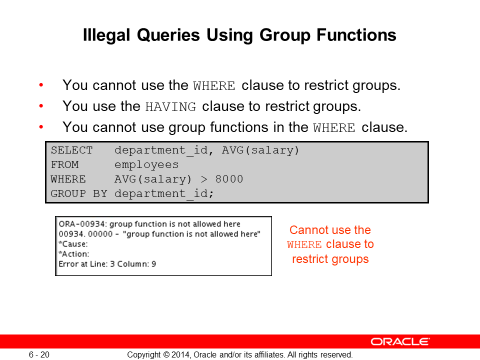
SELECT department\_id, job\_id, COUNT(last\_name)

FROM employees

GROUP BY department\_id, job\_id;



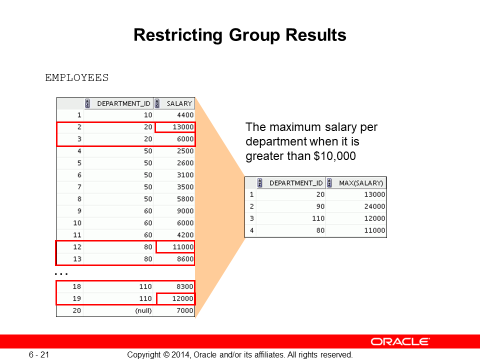
Bu örnekte aslında her meslekte (job\_ıd ) kaç kişi çalışıyor onu bulduk



WHERE yan tümcesi grupları kısıtlamak için kullanılamaz. Slayttaki örnekteki SELECT ifadesi, ortalama maaşı 8.000 dolardan fazla olan departmanların ortalama maaşlarının görüntülenmesini kısıtlamak için WHERE yan tümcesini kullandığından bir hatayla sonuçlanır.kısacası group by fonksiyonlarına(avg, max, mın, sum ….) şart verceksek having kullanıcaz

Ancak, grupları kısıtlamak için HAVING yan tümcesini kullanarak örnekteki hatayı düzeltebilirsiniz:

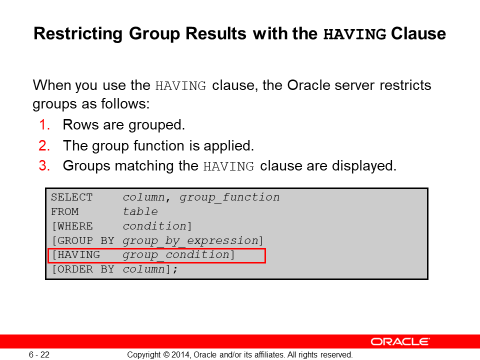
SELECT department\_id, AVG(salary) FROM employees GROUP BY department\_id HAVING AVG(salary) > 8000;



Grupları kısıtlamak için HAVING yan tümcesini, seçtiğiniz satırları kısıtlamak için WHERE yan tümcesini kullandığınız gibi kullanırsınız. Maksimum maaşı 10.000 ABD Dolarından fazla olan bölümlerin her birinde maksimum maaşı bulmak için aşağıdakileri yapmanız gerekir:

1. Bölüm numarasına göre gruplandırarak her bölüm için ortalama maaşı bulun.

2. Grupları maksimum maaşı 10.000 dolardan fazla olan departmanlarla sınırlayın.



Görüntülenecek grupları belirtmek için HAVING yan tümcesini kullanırsınız, böylece grupları toplu bilgi temelinde daha da kısıtlarsınız.

Sözdiziminde group\_condition, belirtilen koşulun doğru olduğu gruplara döndürülen satır gruplarını kısıtlar.

HAVING yan tümcesini kullandığınızda Oracle sunucusu aşağıdaki adımları gerçekleştirir:

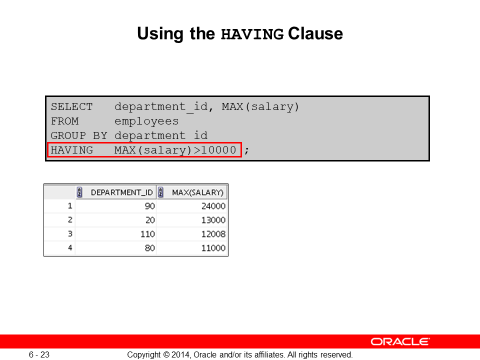
1. Satırlar gruplandırılmıştır.

2. Grup işlevi, gruba uygulanır.

3. HAVING yan tümcesindeki kriterlere uyan gruplar görüntülenir.

HAVING yan tümcesi GROUP BY yan tümcesinden önce gelebilir, ancak daha mantıklı olduğu için önce GROUP BY yan tümcesini yerleştirmeniz önerilir. SELECT listesindeki gruplara HAVING deyimi uygulanmadan önce gruplar oluşturulur ve grup fonksiyonları hesaplanır.

Not: WHERE yan tümcesi satırları kısıtlarken, HAVING yan tümcesi grupları kısıtlar.

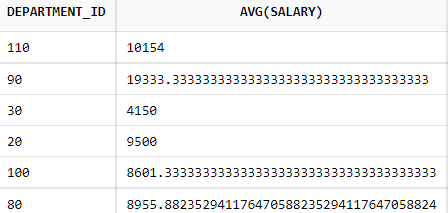


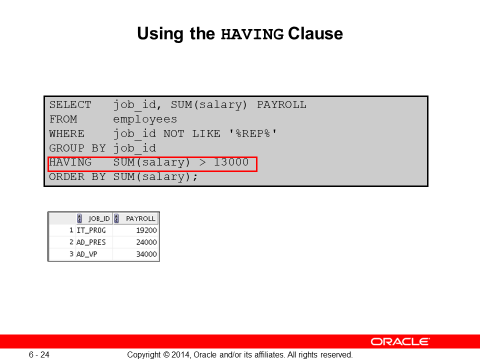
Slayttaki örnek, maksimum maaşı 10.000 ABD Dolarından fazla olan departman numaralarını ve maksimum maaşlarını gösterir.

SELECT listesinde bir grup fonksiyonu kullanmadan GROUP BY yan tümcesini kullanabilirsiniz. Satırları bir grup işlevinin sonucuna göre kısıtlarsanız, HAVING yan tümcesinin yanı sıra bir GROUP BY yan tümcesine sahip olmanız gerekir.

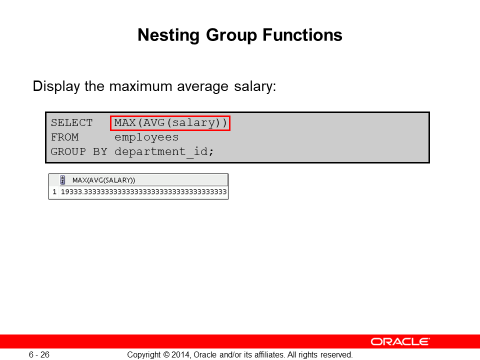
Aşağıdaki örnek, maksimum maaşı 10.000 ABD Dolarından fazla olan departmanların departman numaralarını ve ortalama maaşlarını gösterir:

SELECT department\_id,AVG(salary) FROM hr.employees GROUP BY department\_id HAVING MAX(SALARY)>10000





Slayttaki örnek, toplam maaş bordrosu 13.000 doları aşan her iş için JOB\_ID ve toplam aylık maaşı gösterir. Örnek, satış temsilcilerini hariç tutar ve listeyi toplam aylık maaşa göre sıralar. REP null du onu almak istemedi herhalde



Grup işlevleri, iki işlev derinliğinde iç içe yerleştirilebilir. Slayttaki örnek, her DEPARTMENT\_ID için ortalama maaşı hesaplar ve ardından maksimum ortalama maaşı görüntüler.

Grup işlevleri yuvalanırken GROUP BY yan tümcesinin zorunlu olduğunu unutmayın.

QUİZ

Grup işlevleri ve GROUP BY yan tümcesi için iki yönergeyi tanımlayın.

A- GROUP BY yan tümcesinde bir sütun diğer adı kullanamazsınız.

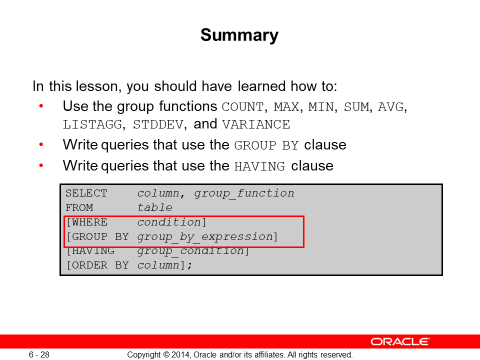
B-GROUP BY sütunu SELECT yan tümcesinde olmalıdır.

C-Bir WHERE yan tümcesi kullanarak, satırları gruplara ayırmadan önce hariç tutabilirsiniz.

D-GROUP BY yan tümcesi satırları gruplandırır ve sonuç kümesinin sırasını sağlar.

E- Bir SELECT yan tümcesine bir grup işlevi(fonksiyonu) eklerseniz, bir GROUP BY yan tümcesi eklemelisiniz.

CEVAP a ve c



AVG, COUNT, MAX, MIN, SUM, LISTAGG, STDDEV ve VARIANCE gibi SQL'de kullanılabilen birkaç grup işlevi vardır.

GROUP BY yan tümcesini kullanarak alt gruplar oluşturabilirsiniz. Ayrıca, gruplar HAVING yan tümcesi kullanılarak kısıtlanabilir.

Bir ifadede HAVING ve GROUP BY yan tümcelerini WHERE yan tümcesinden sonra yerleştirin. GROUP BY ve HAVING yan tümcelerinin WHERE yan tümcesinden sonraki sırası önemli değildir. WHERE yan tümcesini takip ettikleri sürece, önce GROUP BY yan tümcesine ya da HAVING yan tümcesine sahip olabilirsiniz. ORDER BY deyimini sonuna yerleştirin.

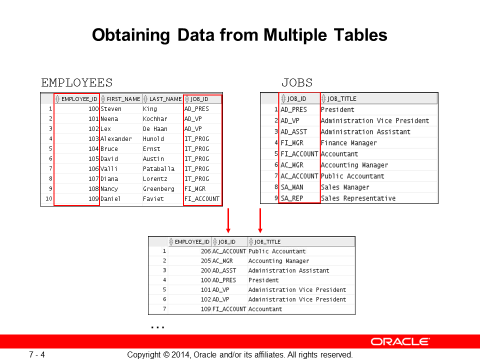
Oracle sunucusu, yan tümceleri aşağıdaki sırayla değerlendirir:

1. İfade bir WHERE yan tümcesi içeriyorsa, sunucu aday satırları oluşturur.

2. Sunucu, GROUP BY yan tümcesinde belirtilen grupları tanımlar.

3. HAVING deyimi, HAVING deyimindeki grup kriterlerini karşılamayan sonuç gruplarını daha da kısıtlar.

Not: Grup işlevlerinin tam listesi için 12c veritabanı için Oracle Database SQL Language Reference'a bakın.



Joinle tablo birleştirmek bir çok tablodan veri elde etmek

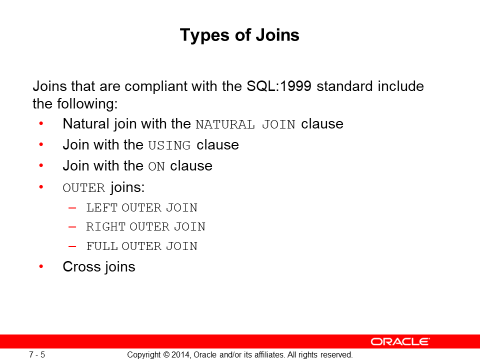
Bazen birden fazla tablodan veri kullanmanız gerekir. Slayttaki örnekte, rapor iki ayrı tablodan verileri görüntüler:

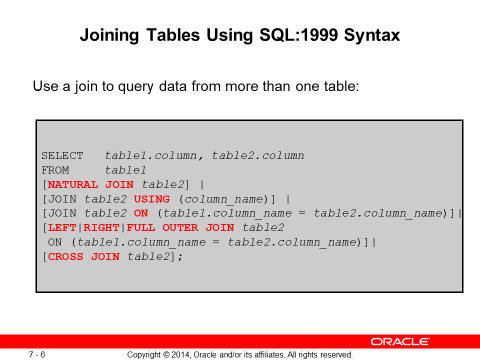
Employee\_id, ÇALIŞANLAR tablosunda bulunur.

Job\_id hem ÇALIŞANLAR hem de JOBS tablolarında bulunur.

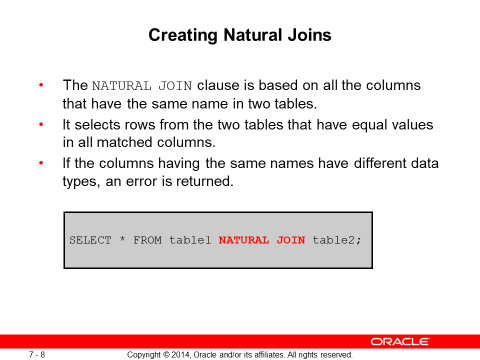
Job\_title, JOBS tablosunda bulunur.

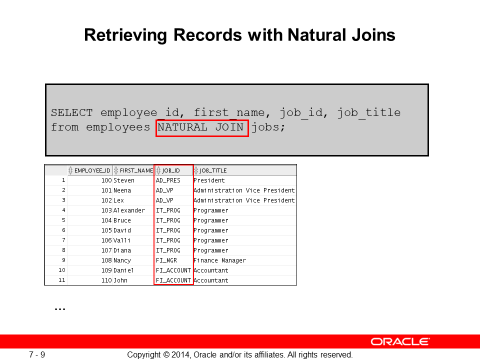
Raporu oluşturmak için ÇALIŞANLAR ve İŞ ​​tablolarını bağlamanız ve her ikisinden de verilere erişmeniz gerekir.





Sözdiziminde: tablo1.sütun, tabloyu ve verilerin alındığı sütunu belirtir NATURAL JOIN, aynı sütun adına dayalı iki tabloyu birleştirir JOIN table2 sütun\_adı KULLANARAK, sütun adına dayalı bir equijoin gerçekleştirir JOIN table2 ON table1.column\_name = table2.column\_name, ON yan tümcesindeki koşula göre bir equijoin gerçekleştirir LEFT/RIGHT/FULL OUTER, OUTER birleşimlerini gerçekleştirmek için kullanılır CROSS JOIN, iki tablodan bir Kartezyen ürün döndürür

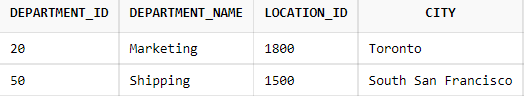


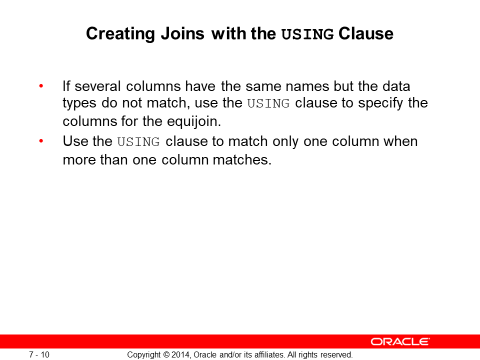
Veri türleri ve adları eşleşen iki tablodaki sütunlara göre tabloları otomatik olarak birleştirebilirsiniz. Bunu NATURAL JOIN anahtar sözcüklerini kullanarak yaparsınız.  
NATURAL JOIN yan tümcesi, iki tabloda aynı ada sahip tüm sütunları temel alır. Tüm eşleşen sütunlarda eşit değerlere sahip iki tablodan satırları seçer. Aynı ada sahip sütunların farklı veri türleri varsa, bir hata döndürülür. 

Slayttaki örnekte, JOBS tablosu, her iki tabloda da aynı ada sahip tek sütun olan JOB\_ID sütunu ile ÇALIŞANLAR tablosuna birleştirilmiştir. Diğer ortak sütunlar mevcut olsaydı, birleştirme hepsini kullanırdı.

WHERE Cümlesiyle Doğal Birleştirmeler

Doğal bir birleştirmeyle ilgili ek kısıtlamalar, bir WHERE yan tümcesi kullanılarak uygulanır. Aşağıdaki örnek, çıktı satırlarını DEPARTMENT\_ID'si 20 veya 50'ye eşit olan satırlarla sınırlar: 2 tabloda da location \_id ortak

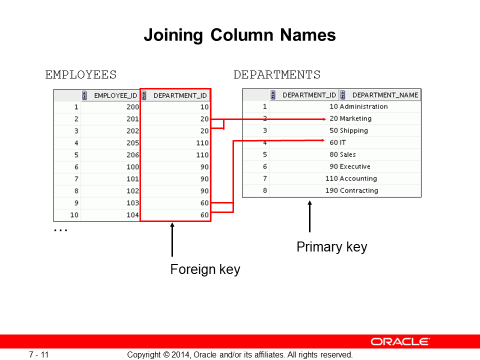




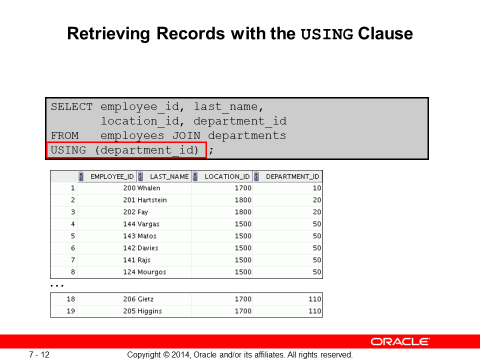
Birkaç sütun aynı ada sahipse, ancak veri türleri eşleşmiyorsa, equijoin sütunlarını belirtmek için USING yan tümcesini kullanın.

Birden fazla sütun eşleştiğinde yalnızca bir sütunu eşleştirmek için USİNG yan tümcesini kullanın.

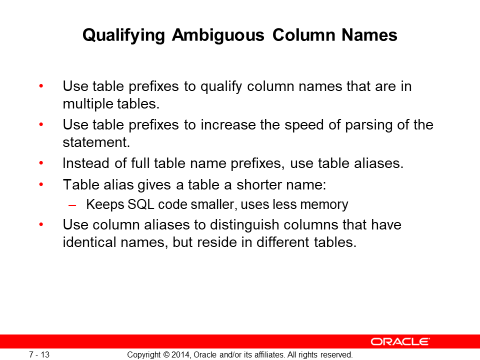
NATURAL JOIN, tabloları birleştirmek için eşleşen adlara ve veri türlerine sahip tüm sütunları kullanır. USING yan tümcesi, yalnızca bir equijoin için kullanılması gereken sütunları belirtmek için kullanılabilir.



Bir çalışanın departman adını belirlemek için EMPLOYEES tablosundaki DEPARTMENT\_ID sütunundaki değeri DEPARTMENTS tablosundaki DEPARTMENT\_ID değerleriyle karşılaştırırsınız. EMPLOYEES ve DEPARTMENTS tabloları arasındaki ilişki bir equijoin'dir; yani her iki tablodaki DEPARTMENT\_ID sütunundaki değerler eşit olmalıdır. Sıklıkla, bu tür birleştirme, birincil ve yabancı anahtar tamamlayıcılarını içerir.



Slayttaki örnekte ÇALIŞANLAR ve BÖLÜMLER tablolarındaki DEPARTMENT\_ID sütunları birleştirilir ve böylece çalışanın çalıştığı departmanın LOCATION\_ID'si gösterilir.



İki veya daha fazla tabloyu birleştirirken, belirsizliği önlemek için sütun adlarını tablo adıyla nitelemeniz gerekir. Tablo önekleri olmadan, SEÇ listesindeki DEPARTMENT\_ID sütunu, DEPARTMENTS tablosundan veya ÇALIŞANLAR tablosundan olabilir. Sorgunuzu yürütmek için tablo önekini eklemeniz gerekir. İki tablo arasında ortak sütun adları yoksa, sütunları nitelemeye gerek yoktur. Ancak, Oracle sunucusuna sütunları tam olarak nerede bulacağını söylediğiniz için tablo önekini kullanmak ifadenin ayrıştırma hızını artırır.

Ancak, tablo adlarıyla sütun adlarını nitelemek, özellikle tablo adları uzunsa, zaman alıcı olabilir. Bunun yerine tablo takma adlarını kullanabilirsiniz. Sütun diğer adının sütuna başka bir ad vermesi gibi, tablo diğer adı da tabloya başka bir ad verir. Tablo takma adları, SQL kodunu daha küçük tutmaya yardımcı olur, bu nedenle daha az bellek kullanır.

Tablo adı tam olarak, ardından bir boşluk ve ardından tablo diğer adı ile belirtilir. Örneğin, ÇALIŞANLAR tablosuna e takma adı ve BÖLÜMLER tablosuna d takma adı verilebilir.

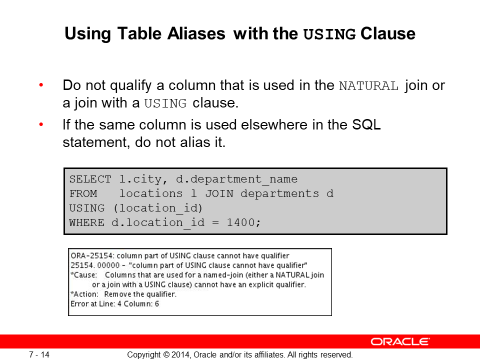
Yönergeler

Tablo takma adları en fazla 30 karakter uzunluğunda olabilir, ancak daha kısa takma adlar uzun olanlardan daha iyidir.

FROM yan tümcesinde belirli bir tablo adı için bir tablo diğer adı kullanılıyorsa, bu tablo diğer adı, SELECT deyimi boyunca tablo adı için değiştirilmelidir.

Tablo takma adları anlamlı olmalıdır.

Tablo diğer adı yalnızca geçerli SELECT ifadesi için geçerlidir.



NATURAL birleştirmede veya USING yan tümcesi ile birleştirmede kullanılan bir sütunu nitelendirmeyin.

Aynı sütun SQL deyiminde başka bir yerde kullanılıyorsa, onu takma ad vermeyin.

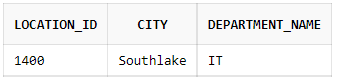
USING yan tümcesine katılırken, USING yan tümcesinde kullanılan bir sütunu niteleyemezsiniz. Ayrıca, bu sütun SQL deyiminin herhangi bir yerinde kullanılıyorsa, ona takma ad veremezsiniz. Örneğin, slaytta bahsedilen sorguda, sütun USING yan tümcesinde kullanıldığından, WHERE yan tümcesinde location\_id sütununa takma ad vermemelisiniz.

USING yan tümcesinde başvurulan sütunlar, SQL deyiminin hiçbir yerinde niteleyiciye (tablo adı veya diğer ad) sahip olmamalıdır. Örneğin, aşağıdaki ifade geçerlidir:

SELECT location\_id, l.city, d.department\_name

FROM locations l JOIN departments d USING (location\_id)

WHERE location\_id = 1400;



Her iki tabloda da ortak olan ancak USING yan tümcesinde kullanılmayan sütunların önüne bir tablo diğer adı konulmalıdır yukarıdaki kodda location\_id nin önünde herhangi bir ön ek yok çünkü using te onu kullanıyom; aksi takdirde, "belirsiz bir şekilde tanımlanmış sütun" hatası alırsınız.

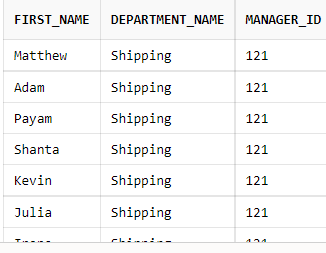
Aşağıdaki açıklamada, manager\_id, hem çalışanlar hem de departmanlar tablosunda bulunur; manager\_id bir tablo diğer adıyla önek değilse yani d.manager\_id gibi, "belirsiz bir şekilde tanımlanmış sütun" hatası verir.

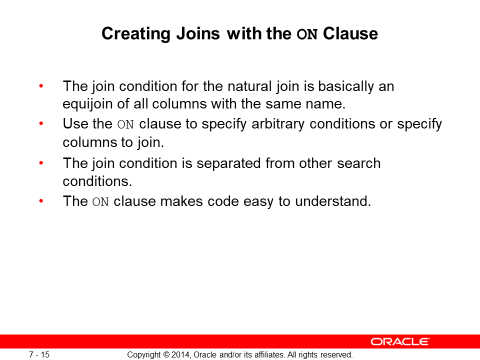
Aşağıdaki ifade geçerlidir:

SELECT first\_name, department\_name, d.manager\_id

FROM hr.employees e JOIN hr.departments d USING (department\_id)

WHERE department\_id = 50;





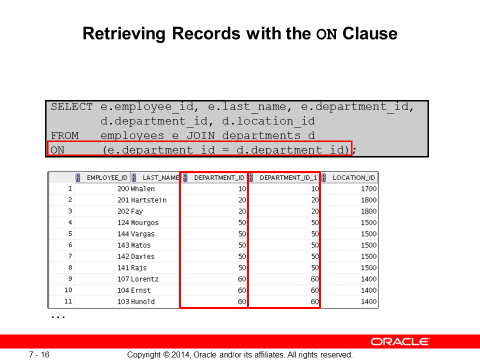
Natural joın için birleştirme koşulu, temelde aynı ada sahip tüm sütunların bir eş birleştirmesidir.

İsteğe bağlı koşulları belirtmek veya birleştirilecek sütunları belirtmek için ON yan tümcesini kullanın.

Birleştirme koşulu, diğer arama koşullarından ayrılır.

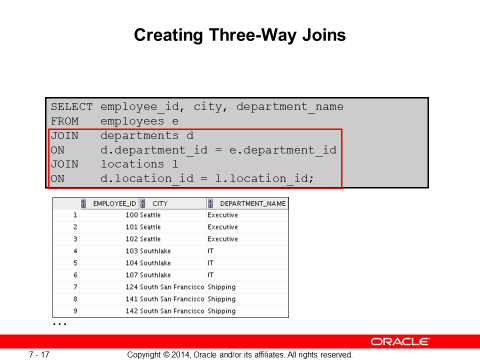
ON yan tümcesi kodun anlaşılmasını kolaylaştırır.   
Bir birleştirme koşulu belirtmek için ON yan tümcesini kullanın. Bununla, WHERE yan tümcesindeki herhangi bir arama veya filtre koşulundan ayrı birleştirme koşulları belirtebilirsiniz.

En kullanışlı yöntem birleştirmede on yöntemi



Bu örnekte, ÇALIŞANLAR ve DEPARTMENTS tablosundaki DEPARTMENT\_ID sütunları ON yan tümcesi kullanılarak birleştirilmiştir. ÇALIŞANLAR tablosundaki bir departman kimliği, BÖLÜMLER tablosundaki bir departman kimliğine eşit olduğunda, satır döndürülür. Tablo diğer adı, eşleşen column\_names'i nitelemek için gereklidir.

Farklı adlara sahip sütunları birleştirmek için ON yan tümcesini de kullanabilirsiniz. Birleştirilen sütunların etrafındaki parantezler, slayttaki örnekte olduğu gibi (e.department\_id = d.department\_id) isteğe bağlıdır. Yani ON e.department\_id = d.department\_id bile çalışacaktır.



Üç yollu birleşim, üç tablonun birleşimidir. Burada yapılacak ilk katılım EMPLOYEES JOIN DEPARTMENTS'dir. İlk birleştirme koşulu, ÇALIŞANLAR ve DEPARTMENTS içindeki sütunlara başvurabilir, ancak LOCATIONS içindeki sütunlara başvuramaz. İkinci birleştirme koşulu, üç tablonun tümünden sütunlara başvurabilir. Not: Slayttaki kod örneği, USING yan tümcesi ile de gerçekleştirilebilir:

SELECT e.employee\_id, l.city, d.department\_name

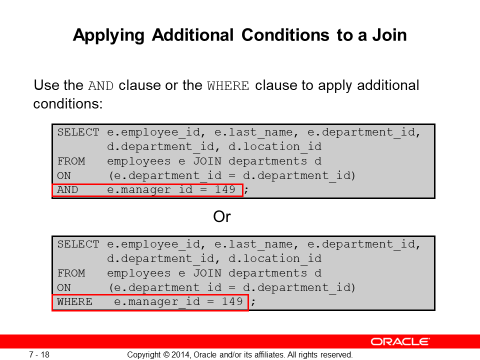
FROM employees e

JOIN departments d

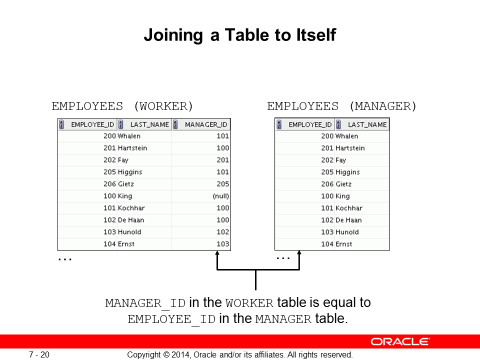
USING (department\_id)

JOIN locations l

USING (location\_id);



Birleştirme için ek koşullar uygulayabilirsiniz. Gösterilen örnek, ÇALIŞANLAR ve BÖLÜMLER tablolarında bir birleştirme gerçekleştirir ve ayrıca yalnızca yönetici kimliği 149 olan çalışanları görüntüler. ON deyimine ek koşullar eklemek için AND deyimleri ekleyebilirsiniz. Alternatif olarak, ek koşulları uygulamak için bir WHERE yan tümcesi kullanabilirsiniz. Her iki sorgu da aynı çıktıyı üretir.



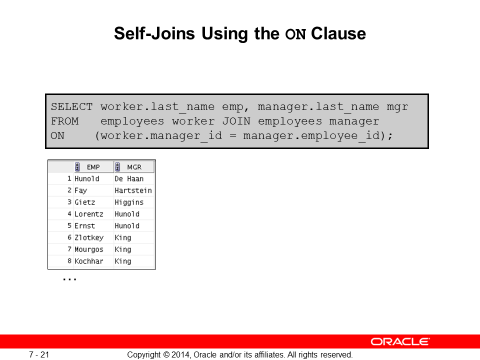
Bazen bir masaya kendi başına katılmanız gerekir. Her çalışanın yöneticisinin adını bulmak için ÇALIŞANLAR tablosuna kendisine katılmanız veya kendi kendine katılma işlemi yapmanız gerekir. Örneğin, Ernst'in yöneticisinin adını bulmak için şunları yapmanız gerekir:

LAST\_NAME sütununa bakarak ÇALIŞANLAR tablosunda Ernst'i bulun

MANAGER\_ID sütununa bakarak Ernst'in yönetici numarasını bulun. Ernst'in yönetici numarası 103'tür.

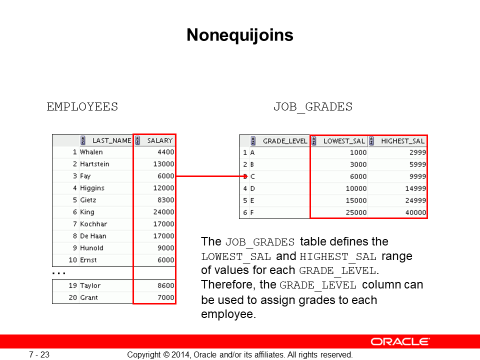
LAST\_NAME sütununa bakarak EMPLOYEE\_ID 103'e sahip yöneticinin adını bulun. Hunold'un çalışan sayısı 103'tür, yani Hunold, Ernst'in yöneticisidir.

Bu süreçte tabloya iki kez bakıyorsunuz. Tabloya ilk baktığınızda LAST\_NAME sütununda Ernst'i ve MANAGER\_ID değeri 103'ü buluyorsunuz. İkinci kez EMPLOYEE\_ID sütununda 103'ü ve LAST\_NAME sütununda Hunold'u buluyorsunuz. Tabloya iki kere bakacağımız için self joın yapıcaz



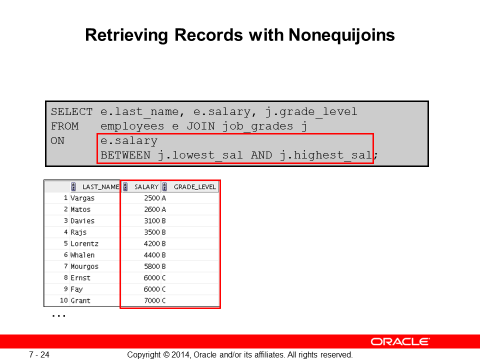
ON yan tümcesi, aynı tablo içinde veya farklı bir tabloda farklı adlara sahip sütunları birleştirmek için de kullanılabilir.

Gösterilen örnek, EMPLOYEE\_ID ve MANAGER\_ID sütunlarına dayalı olarak EMPLOYEES tablosunun kendi kendine katılmasıdır.



Bir nonequijoin, eşitlik operatörü dışında bir şey içeren bir birleştirme koşuludur.

EMPLOYEES tablosu ile JOB\_GRADES tablosu arasındaki ilişki, bir nonequijoin örneğidir. ÇALIŞANLAR tablosundaki MAAŞ sütunu, JOB\_GRADES tablosunun LOWEST\_SAL ve HIGHEST\_SAL sütunlarındaki değerler arasında değişir. Bu nedenle, her çalışan maaşına göre derecelendirilebilir. İlişki, eşitlik (=) operatörü dışında bir operatör kullanılarak elde edilir.



Slayttaki örnek, bir çalışanın maaş derecesini değerlendirmek için eşit olmayan bir birleşme oluşturur. Maaş, düşük ve yüksek maaş aralıklarının herhangi bir çifti arasında olmalıdır.

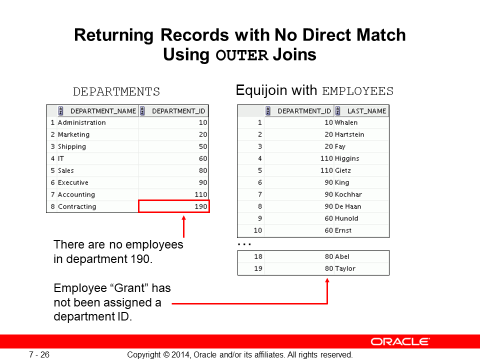
Bu sorgu yürütüldüğünde tüm çalışanların tam olarak bir kez göründüğüne dikkat etmek önemlidir. Listede hiçbir çalışan tekrarlanmaz. Bunun iki nedeni vardır:

JOB\_GRADES tablosundaki satırların hiçbiri çakışan notlar içermiyor. Yani bir çalışanın maaş değeri, maaş notu tablosundaki satırlardan birinin sadece düşük maaş ve yüksek maaş değerleri arasında yer alabilir.

Çalışanların maaşlarının tamamı iş notu tablosunda belirtilen limitler dahilindedir. Yani hiçbir çalışan, LOWEST\_SAL sütununda yer alan en düşük değerden daha az veya HIGHEST\_SAL sütununda yer alan en yüksek değerden daha fazla kazanmaz.

Not: Diğer koşullar (<= ve >= gibi) kullanılabilir, ancak BETWEEN en basitidir. BETWEEN koşulunu kullanırken önce düşük değeri ve en son yüksek değeri belirtmeyi unutmayın. Oracle sunucusu BETWEEN koşulunu bir çift AND koşuluna çevirir. Bu nedenle, BETWEEN'i kullanmanın performans avantajı yoktur, ancak yalnızca mantıksal basitlik için kullanılmalıdır.

Tablo takma adları, olası belirsizlik nedeniyle değil, performans nedenleriyle slayt örneğinde belirtilmiştir.



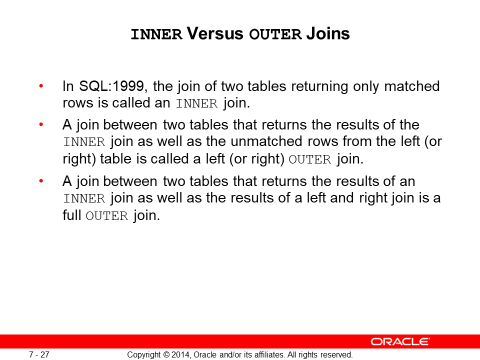
Bir satır bir birleştirme koşulunu karşılamıyorsa, sorgu sonucunda satır görünmez.

Slayt örneğinde, sonucu sağda döndürmek için ÇALIŞANLAR ve DEPARTMENTS tablolarında basit bir equijoin koşulu kullanılmıştır. Sonuç kümesi aşağıdakileri içermez:

Departman Kimliği 190, çünkü ÇALIŞANLAR tablosunda bu departman kimliğine sahip çalışan yok

Bu çalışana bir departman kimliği atanmadığı için soyadı Grant olan çalışan

Herhangi bir çalışanı olmayan departman kaydını veya atanmış bir departmanı olmayan çalışanları döndürmek için bir OUTER birleştirme kullanabilirsiniz.



SQL:1999'da, yalnızca eşleşen satırları döndüren iki tablonun birleşimine INNER birleşim adı verilir.

Sol (veya sağ) tablodaki eşleşmeyen satırların yanı sıra INNER birleşiminin sonuçlarını döndüren iki tablo arasındaki birleşim, sol (veya sağ) OUTER birleşim olarak adlandırılır.

Bir INNER birleşiminin sonuçlarını ve bir sol ve sağ birleşimin sonuçlarını döndüren iki tablo arasındaki birleşim, tam bir OUTER birleşimidir.

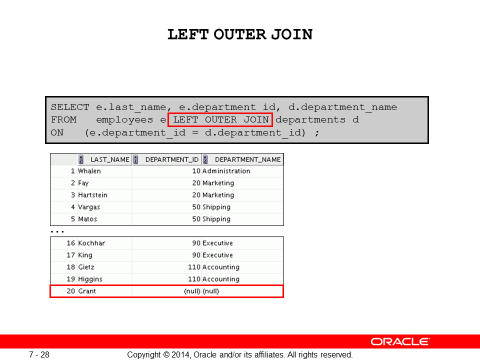
Tabloları NATURAL JOIN, USING veya ON yan tümceleriyle birleştirmek, INNER birleşimle sonuçlanır. Çıktıda eşleşmeyen satırlar görüntülenmez. Eşleşmeyen satırları döndürmek için bir OUTER birleştirme kullanabilirsiniz. Bir OUTER birleştirme, birleştirme koşulunu karşılayan tüm satırları döndürür ve aynı zamanda, diğer tablodaki hiçbir satırın birleştirme koşulunu karşılamadığı bir tablodaki satırların bir kısmını veya tamamını döndürür.

Üç tür OUTER birleşim vardır:

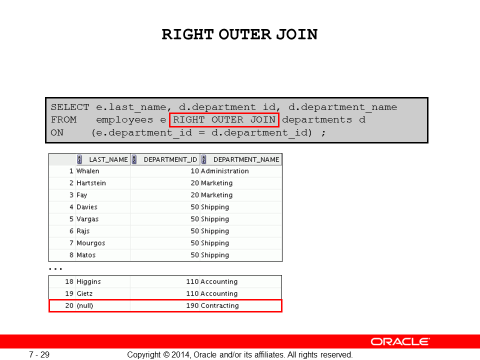
LEFT OUTER

RIGHT OUTER

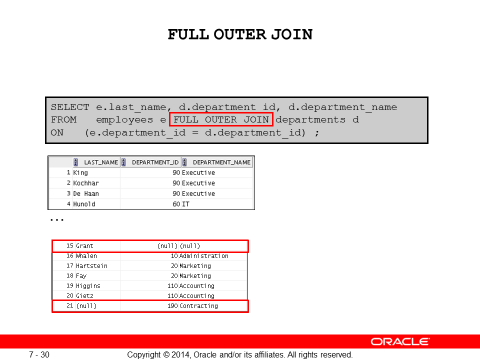
FULL OUTER



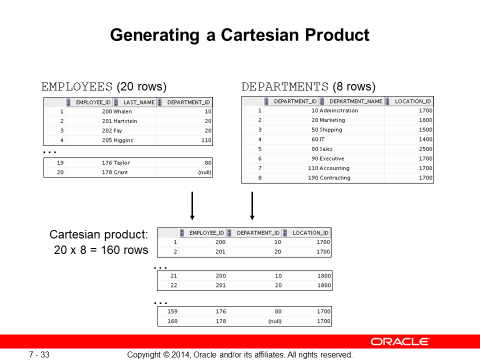
Bu sorgu, DEPARTMENTS tablosunda eşleşme olmasa bile, soldaki tablo olan EMPLOYEES tablosundaki tüm satırları alır. Mesela grant ın deaprtman ıd si ve department\_name i yok ama onuda getirdi null tipinde



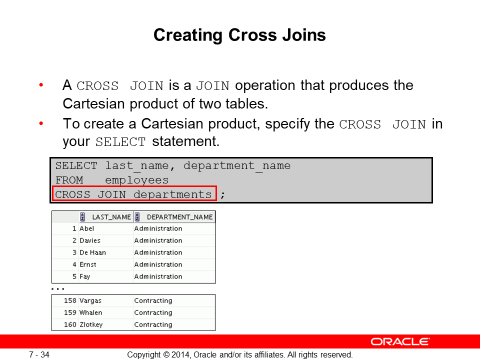
Bu sorgu, EMPLOYEES tablosunda eşleşme olmasa bile sağdaki tablo olan DEPARTMENTS tablosundaki tüm satırları alır.yukarıda 190 id li dpartmanın hiç çalışanı yok ama onuda getirdi last\_name null şekilde



**Bu sorgu, DEPARTMENTS tablosunda eşleşme olmasa bile EMPLOYEES tablosundaki tüm satırları alır. Ayrıca ÇALIŞANLAR tablosunda eşleşme olmasa bile DEPARTMENTS tablosundaki tüm satırları alır. Left ve right joinin birleşimi gibi ayrıca left outer joın yazmaya gerek yok direkt left joın de yazsak olur.**

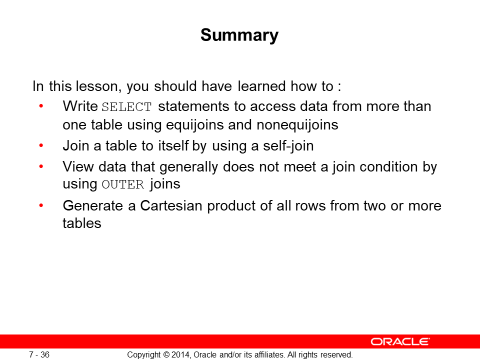
****

Bir birleştirme koşulu atlanırsa bir Kartezyen ürün oluşturulur. Slayttaki örnek, ÇALIŞANLAR ve BÖLÜMLER tablolarından çalışan soyadını ve departman adını görüntüler. Hiçbir birleştirme koşulu belirtilmediği için, ÇALIŞANLAR tablosundaki tüm satırlar (20 satır), DEPARTMENTS tablosundaki tüm satırlarla (8 satır) birleştirilir, böylece çıktıda 160 satır oluşturulur.



Slayttaki örnek, ÇALIŞANLAR ve BÖLÜMLER tablolarının Kartezyen ürününü üretmektedir.

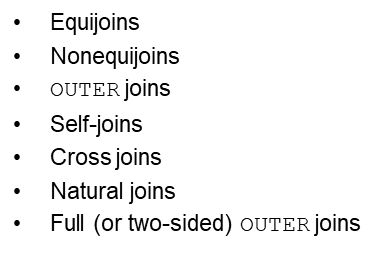
Kartezyen bir ürün oluşturmayı düşündüğünüzde, SELECT'inizde CROSS JOIN'i açıkça belirtmek iyi bir uygulamadır. Bu nedenle, bunun olmasını istediğiniz çok açıktır ve bu, eksik birleştirmelerin sonucu değildir.



Bu derste, şunları yapmayı öğrenmiş olmalısınız: Eşbirleşimleri ve eşbirleşim olmayanları kullanarak birden fazla tablodaki verilere erişmek için SELECT ifadeleri yazın Kendi kendine katılma özelliğini kullanarak bir tabloya katılın OUTER birleşimlerini kullanarak genellikle bir birleştirme koşulunu karşılamayan verileri görüntü leyin İki veya daha fazla tablodan tüm satırların Kartezyen ürününü üret

Tabloları birleştirmenin birden çok yolu vardır.

Birleşim Türleri



**Cartesian Products**

Bir Cartesian Products, tüm satır kombinasyonlarının görüntülenmesiyle sonuçlanır. Bu, ya WHERE yan tümcesi atlanarak ya da CROSS JOIN yan tümcesi belirtilerek yapılır.

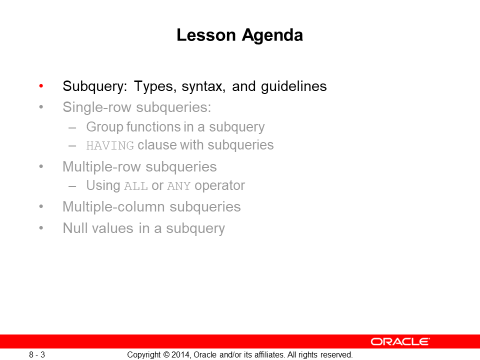
**Table Aliases**

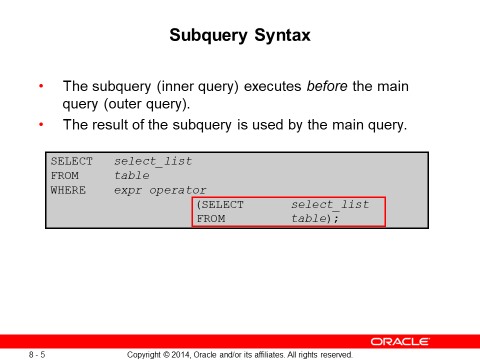
Tablo takma adları, veritabanı erişimini hızlandırır.

Belleği koruyarak SQL kodunu daha küçük tutmaya yardımcı olabilirler.

Sütun belirsizliğini önlemek için bazen zorunludurlar.

Subqueries





Alt sorgu (iç sorgu), ana sorgudan (dış sorgu) önce yürütülür. Alt sorgunun sonucu ana sorgu tarafından kullanılır. Alt sorgu, başka bir SELECT ifadesinin yan tümcesine katıştırılmış bir SELECT ifadesidir. Alt sorguları kullanarak basit ifadelerden güçlü ifadeler oluşturabilirsiniz. Tablodaki verilere bağlı bir koşula sahip bir tablodan satır seçmeniz gerektiğinde çok yararlı olabilirler.

Alt sorguyu aşağıdakiler dahil bir dizi SQL yan tümcesine yerleştirebilirsiniz:

WHERE yan tümcesi

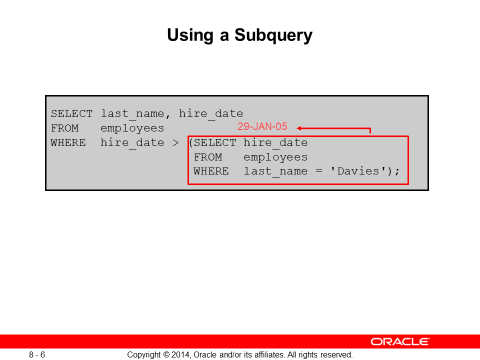
HAVING yan tümcesi

FROM yan tümcesi

Sözdiziminde:

operatör >, = veya IN gibi bir karşılaştırma koşulu içerir

Alt sorguya genellikle iç içe geçmiş bir SELECT, alt SELECT veya iç SELECT ifadesi denir. Alt sorgu genellikle ilk önce yürütülür ve çıktısı, ana (veya dış) sorgu için sorgu koşulunu tamamlamak için kullanılır.

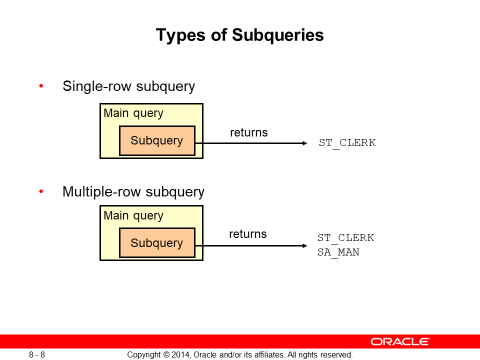


Slaytta, iç sorgu, çalışan Davies'in işe alınma tarihini belirler. Dış sorgu, iç sorgunun sonucunu alır ve bu sonucu Davies'ten sonra işe alınan tüm çalışanları görüntülemek için kullanır.

Bir alt sorgu parantez içine alınmalıdır.

Okunabilirlik için alt sorguyu karşılaştırma koşulunun sağ tarafına yerleştirin. Ancak, alt sorgu, karşılaştırma operatörünün her iki tarafında da görünebilir.

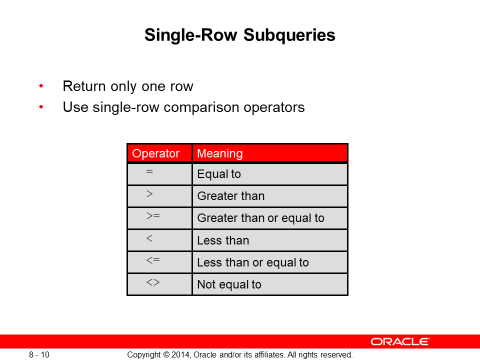
Alt sorgularda iki sınıf karşılaştırma koşulu kullanılır: tek satırlı operatörler ve çok satırlı operatörler



Tek satırlı alt sorgular: İç SELECT ifadesinden yalnızca bir satır döndüren sorgular

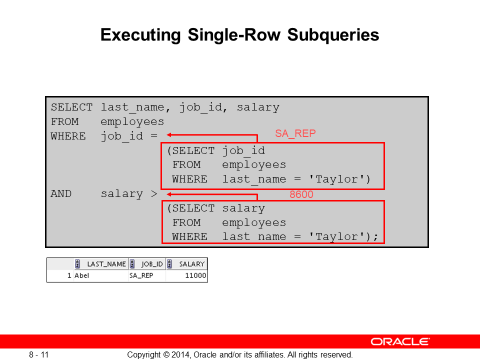
Birden çok satırlı alt sorgular: İç SELECT ifadesinden birden fazla satır döndüren sorgular

Not: İç SELECT ifadesinden birden fazla sütun döndüren sorgular olan çok sütunlu alt sorgular da vardır. Bunlar Oracle Database: SQL Workshop II kursunda ele alınmaktadır



Tek satırlı alt sorgu, iç SELECT ifadesinden bir satır döndüren sorgudur. Bu alt sorgu türü, tek satırlık bir operatör kullanır. Slayt, tek satırlı operatörlerin bir listesini verir. **Display the employees whose job ID is the same as that of employee 141: 141 employee id li adamın job id siyle aynı olanın last\_name ve job id sini getir**

**SELECT last\_name job\_id FROM hr.employees WHERE job\_id = (SELECT job\_id FROM hr.employees WHERE employee\_id=141**

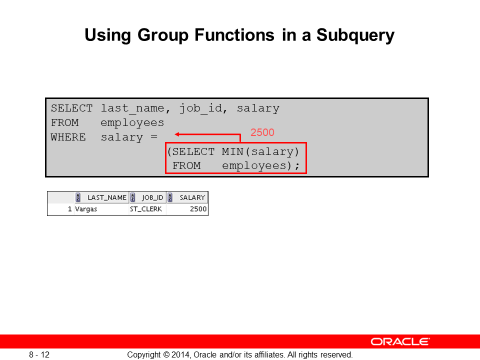


Bir SELECT ifadesi, bir sorgu bloğu olarak düşünülebilir. Slayttaki örnek, “Taylor” ile aynı işi yapan ancak ondan daha fazla maaş alan çalışanları göstermektedir.

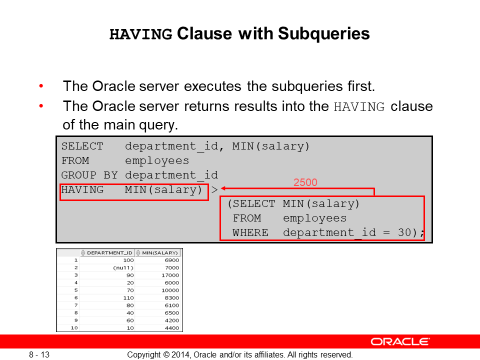
Örnek üç sorgu bloğundan oluşur: dış sorgu ve iki iç sorgu. Önce iç sorgu blokları yürütülür ve sırasıyla SA\_REP ve 8600 sorgu sonuçları üretilir. Dış sorgu bloğu daha sonra işlenir ve arama koşullarını tamamlamak için iç sorgular tarafından döndürülen değerleri kullanır.

Her iki iç sorgu da tek değerler döndürür (sırasıyla SA\_REP ve 8600), bu nedenle bu SQL deyimi tek satırlı alt sorgu olarak adlandırılır.

Not: Dış ve iç sorgular farklı tablolardan veri alabilir.

7

Tek bir satır döndürmek için bir alt sorguda bir grup işlevi kullanarak ana sorgudaki verileri görüntüleyebilirsiniz. Alt sorgu parantez içindedir ve karşılaştırma koşulundan sonra yer alır. Slayttaki örnek, maaşı asgari maaşa eşit olan tüm çalışanların çalışan soyadını, iş kimliğini ve maaşını gösterir. MIN grup işlevi, dış sorguya tek bir değer (2500) döndürür.



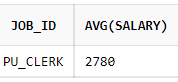
Department id si 30 olan departmandaki minimumum salaryden daha fazla maaş olan departmanların çalışanlarının en düşük maaşını getiriyo.

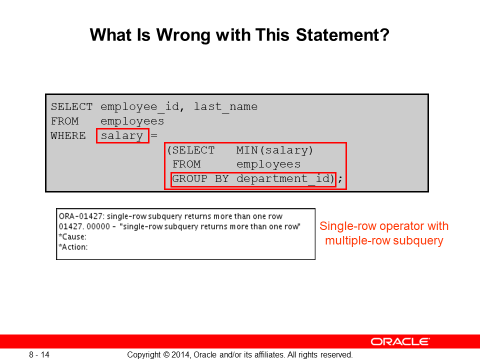
Alt sorguları yalnızca WHERE yan tümcesinde değil, HAVING yan tümcesinde de kullanabilirsiniz. Oracle sunucusu alt sorguyu yürütür ve sonuçlar ana sorgunun HAVING yan tümcesine döndürülür.

Slayttaki SQL ifadesi, asgari maaşı 30 numaralı departmandan daha yüksek olan tüm departmanları gösterir.

Ortalama maaşı en düşük olan job id ye ortalama maaşı eşit olan job id ve maaşı getir.

SELECT job\_id , AVG(salary) FROM hr.employees GROUP BY job\_id HAVİNG AVG(salary) =(SELECT MIN(AVG(salary))FROM hr.employees GROUP BY job\_id)



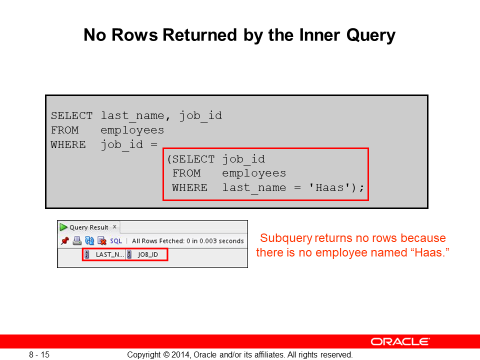


Alt sorgudan birden fazla kayıt geliyor ama biz sadece salary nin bir kaydın gelmesini istiyoz Tek satırlık bir alt sorgu için birden fazla satır döndürüldüğünde, alt sorgularda yaygın bir hata oluşur.

Slayttaki SQL deyiminde, alt sorgu bir GROUP BY yan tümcesi içerir; bu, alt sorgunun bulduğu her grup için bir tane olmak üzere birden çok satır döndüreceği anlamına gelir. Bu durumda, alt sorgunun sonuçları 4400, 6000, 2500, 4200, 7000, 17000 ve 8300'dür.

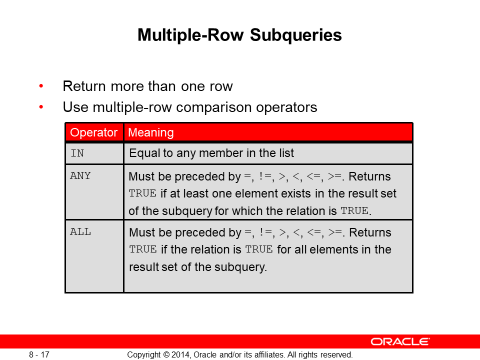
Dış sorgu bu sonuçları alır ve bunları WHERE yan tümcesinde kullanır. WHERE yan tümcesi, yalnızca bir değer bekleyen tek satırlı bir karşılaştırma operatörü olan eşit (=) operatörü içerir. = operatörü, alt sorgudan birden fazla değer kabul edemez ve bu nedenle hatayı oluşturur.

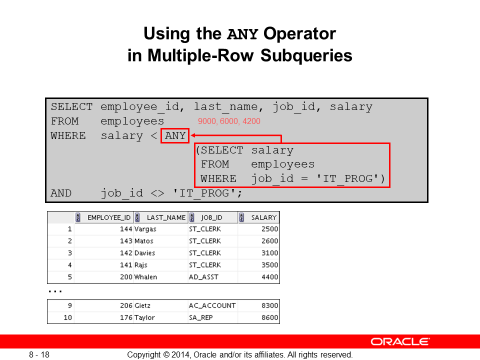
Bu hatayı düzeltmek için = operatörünü IN olarak değiştirin.

 Alt sorgularla ilgili diğer bir yaygın sorun, iç sorgu tarafından hiçbir satır döndürülmediğinde ortaya çıkar.

Slayttaki SQL deyiminde, alt sorgu bir WHERE yan tümcesi içerir. Muhtemelen amaç, adı Haas olan çalışanı bulmaktır. İfade doğrudur, ancak Haas adında bir çalışan olmadığı için yürütüldüğünde hiçbir satır seçmez. Bu nedenle, alt sorgu hiçbir satır döndürmez.

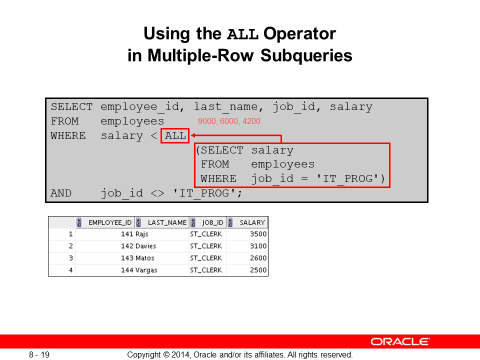
Dış sorgu, alt sorgunun (boş) sonuçlarını alır ve bu sonuçları WHERE yan tümcesinde kullanır. Dış sorgu, NULL'a eşit bir iş kimliğine sahip hiçbir çalışan bulamaz ve bu nedenle hiçbir satır döndürmez. Null değerine sahip bir iş varsa, iki boş değerin karşılaştırılması bir boş değer verdiği için satır döndürülmez; bu nedenle, WHERE koşulu doğru değildir.





ANY operatörü (ve eşanlamlısı SOME operatörü), bir değeri bir alt sorgu tarafından döndürülen her bir değerle karşılaştırır. Slayt örneği, BT programcısı olmayan ve maaşı herhangi bir BT programcısından daha az olan çalışanları gösterir. Bir programcının kazandığı maksimum maaş 9,000 dolardır.minimumda 4200 <ANY, maksimumdan daha az anlamına gelir. >ANY minimumdan daha fazlası anlamına gelir.

=ANY, IN ile eşdeğerdir. Şimdi yukarıdaki iç sorgu job idsi ıt olan çalışanların maaşını alır ve bu maaştan herhangi birinden daha az alan çalışanların istenilen bilgilerini getirir.mesela 4200 den küçük 2500 var 6000 den küçük 4400 var 9000 den küçük 8300 var yani 9000 6000 4200 den herhangi birinden küçükse salarysi where koşulu doğru oluyo.

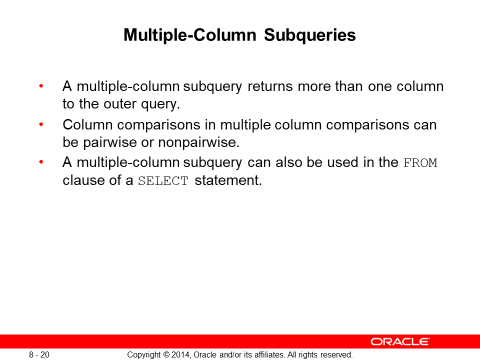


ALL da burda alt sorgudan dönen tüm değerlerden küçük olaması lazım.

ALL operatörü, bir değeri bir alt sorgu tarafından döndürülen her değerle karşılaştırır. Slayttaki örnek, maaşı IT\_PROG iş kimliğine sahip ve işi IT\_PROG olmayan tüm çalışanların maaşından daha az olan çalışanları gösterir.

>ALL, maksimumdan daha fazla anlamına gelir ve <ALL, minimumdan daha az anlamına gelir.

NOT operatörü IN, ANY ve ALL operatörleriyle birlikte kullanılabilir.



Çok sütunlu bir alt sorgu, dış sorguya birden fazla sütun döndürür ve dış sorgunun FROM, WHERE veya HAVING yan tümcesinde listelenebilir.

İki veya daha fazla sütunu karşılaştırmak istiyorsanız, mantıksal operatörleri kullanarak bileşik bir WHERE yan tümcesi yazmalısınız. Birden çok sütunlu alt sorgular, yinelenen WHERE koşullarını tek bir WHERE yan tümcesinde birleştirmenize olanak tanır.

IN operatörü, bir dizi değer içindeki bir değeri kontrol etmek için kullanılır. Değer listesi, bir alt sorgu tarafından döndürülen sonuçlardan gelebilir.

Syntax:

SELECT *column, column, ...*

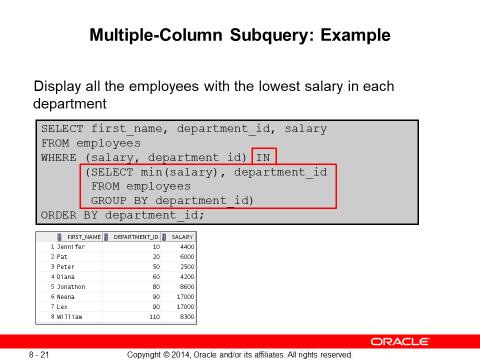
FROM *table*

WHERE (*column, column, ...) IN*

(SELECT *column, column, ...*

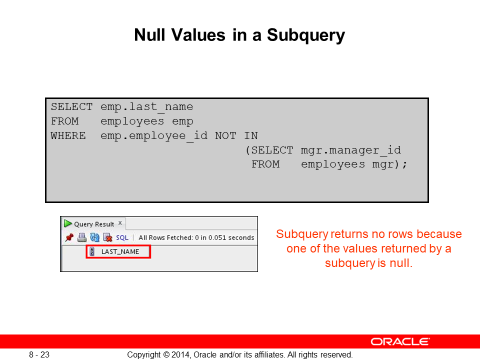
FROM *table*

WHERE *condition);*

**

*Slayttaki örnek, alt sorgu birden fazla* ***sütun*** *döndürdüğü için çok sütunlu bir alt sorgu örneğidir.*

*Önce iç sorgu yürütülür ve her departman için en düşük maaşı ve departman\_id yi döndürür. Ana sorgu bloğu daha sonra işlenir ve arama koşulunu tamamlamak için iç sorgu tarafından döndürülen değerleri kullanır.*

**

Slayttaki SQL ifadesi, astı olmayan tüm çalışanları görüntülemeye çalışır. Mantıksal olarak, bu SQL ifadesinin 12 satır döndürmesi gerekirdi. Ancak, SQL deyimi herhangi bir satır döndürmez. İç sorgu tarafından döndürülen değerlerden biri boş bir değerdir ve bu nedenle sorgunun tamamı satır döndürmez.

Bunun nedeni, bir boş değeri karşılaştıran tüm koşulların bir boş değerle sonuçlanmasıdır. Bu nedenle, boş değerlerin bir alt sorgunun sonuç kümesinin parçası olması muhtemel olduğunda, NOT IN operatörünü kullanmayın. NOT IN operatörü <> ALL ile eşdeğerdir.

IN operatörünü kullanırsanız, bir alt sorgunun sonuç kümesinin parçası olarak boş değerin sorun olmadığına dikkat edin. IN operatörü =ANY ile eşdeğerdir. Örneğin, managerı olan çalışanları görüntülemek için aşağıdaki SQL deyimini kullanın:

SELECT emp.last\_name

FROM employees emp

WHERE emp.employee\_id IN

(SELECT mgr.manager\_id

FROM employees mgr);

Alternatif olarak, üstü olmayan tüm çalışanları görüntülemek için alt sorguya bir WHERE yan tümcesi eklenebilir: yani bu sorgu ceo nun bilgilerini getircek ceo nun managerı yok

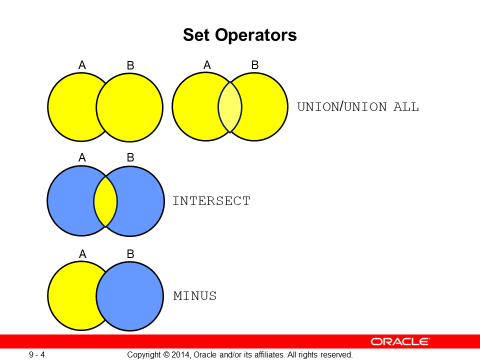
SELECT last\_name FROM employees

WHERE employee\_id NOT IN

(SELECT manager\_id

FROM employees

WHERE manager\_id IS NOT NULL);





Set operatörleri, iki veya daha fazla bileşen sorgusunun sonuçlarını tek bir sonuçta birleştirir. Küme operatörlerini içeren sorgulara bileşik sorgular denir.

Tüm küme operatörleri eşit önceliğe sahiptir. Bir SQL deyimi birden fazla küme operatörü içeriyorsa, parantez açıkça başka bir sıra belirtmiyorsa, Oracle sunucusu bunları soldan (üstten) sağa (alttan) doğru değerlendirir. INTERSECT operatörünü diğer küme operatörleriyle birlikte kullanan sorgularda değerlendirme sırasını açıkça belirtmek için parantez kullanmalısınız.

SET OPERATOR RULES

Sorguların SELECT listelerindeki ifadeler sayı ve veri türü olarak eşleşmelidir. UNION, UNION ALL, INTERSECT ve MINUS operatörlerini kullanan sorguların SELECT listelerinde aynı sayıda ve veri türünde sütunlara sahip olması gerekir. Bileşik sorgudaki sorguların SELECT listesindeki sütunların veri tipi tam olarak aynı olmayabilir. İkinci sorgudaki sütun, ilk sorgudaki karşılık gelen sütunla aynı veri türü grubunda (sayısal veya karakter gibi) olmalıdır.

Set operatörleri alt sorgularda kullanılabilir.

Diğer küme operatörleriyle INTERSECT operatörünü kullanan sorgularda değerlendirme sırasını belirtmek için parantez kullanmalısınız. Bu, INTERSECT operatörüne diğer küme operatörlerinden daha fazla öncelik verecek yeni ortaya çıkan SQL standartlarıyla uyumluluğu sağlar.

Bir sorgu, küme operatörlerini kullandığında, Oracle sunucusu, UNION ALL operatörü durumu dışında, yinelenen satırları otomatik olarak ortadan kaldırır. Çıktıdaki sütun adlarına, ilk SELECT deyimindeki sütun listesi tarafından karar verilir. Varsayılan olarak çıktı, SELECT yan tümcesinin ilk sütununun artan düzeninde sıralanır.

Bir bileşik sorgunun bileşen sorgularının SELECT listelerindeki karşılık gelen ifadeler, sayı ve veri türü olarak eşleşmelidir. Bileşen sorguları, seçme karakter verilerini sorgularsa, dönüş değerlerinin veri türü aşağıdaki gibi belirlenir:

Her iki sorgu da eşit uzunluktaki CHAR veri türü değerlerini seçerse, döndürülen değerler bu uzunlukta CHAR veri türüne sahiptir. Sorgular farklı uzunluklarda CHAR değerlerini seçerse, döndürülen değer, daha büyük CHAR değerinin uzunluğuyla VARCHAR2'dir.

Sorgulardan biri veya her ikisi VARCHAR2 veri türü değerlerini seçerse, döndürülen değerler VARCHAR2 veri türüne sahiptir.

Bileşen sorguları sayısal verileri seçerse, dönüş değerlerinin veri türü sayısal önceliğe göre belirlenir. Tüm sorgular NUMERİC türündeki değerleri seçerse, döndürülen değerler NUMERİC veri türüne sahiptir. Set operatörlerini kullanan sorgularda Oracle sunucusu, veri türü grupları arasında örtük dönüştürme gerçekleştirmez. Bu nedenle, bileşen sorgularının karşılık gelen ifadeleri hem karakter verilerine hem de sayısal verilere çözümlenirse, Oracle sunucusu bir hata döndürür.

Bu derste iki tablo kullanılır: EMPLOYEES tablosu ve RETIRED\_EMPLOYEES tablosu.

Benzersiz bir kimlik numarası, e-posta adresi, iş kimliği (ST\_CLERK, SA\_REP vb.), maaş, yönetici vb. gibi çalışan ayrıntılarını depolayan ÇALIŞANLAR tablosuna zaten aşinasınız.

RETIRED\_EMPLOYEES, şirketten ayrılan çalışanların detaylarını saklar.

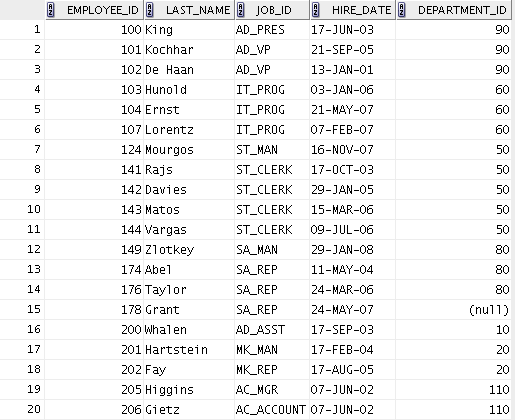
EMPLOYEES ve RETIRED\_EMPLOYEES tablolarından yapı ve veriler sonraki sayfalarda gösterilmektedir.

DESCRIBE employees

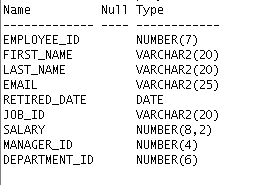


SELECT employee\_id, last\_name, job\_id, hire\_date, department\_id

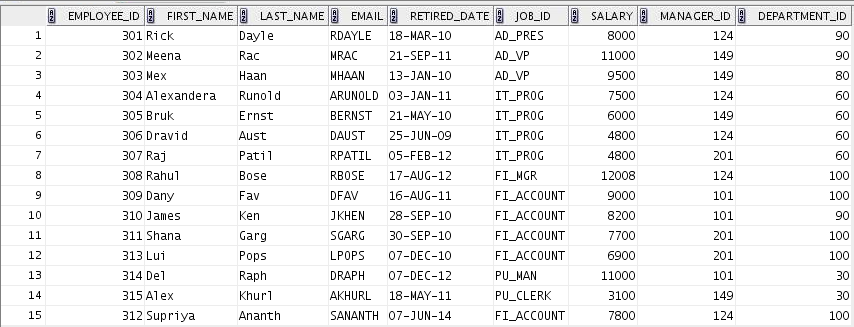
FROM employees;

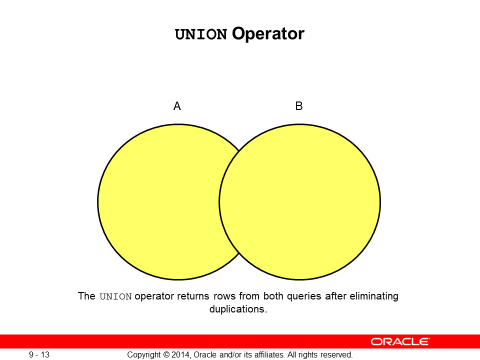


DESCRIBE **retired\_employees**



SELECT \* FROM retired\_employees;





UNION operatörü, herhangi bir sorgu tarafından seçilen tüm satırları döndürür. Birden çok tablodan tüm satırları döndürmek ve yinelenen satırları ortadan kaldırmak için UNION operatörünü kullanın.

Yönergeler: Seçilecek sütun sayısı aynı olmalıdır.

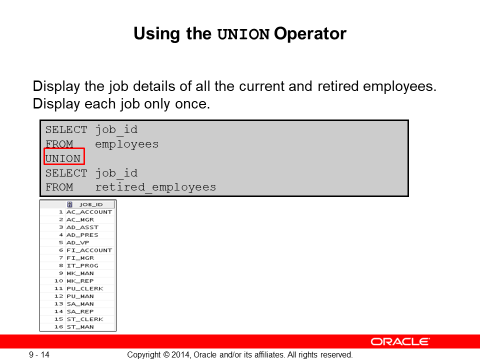
Seçilmekte olan sütunların veri türleri aynı veri türü grubunda (sayısal veya karakter gibi) olmalıdır.

Sütunların adlarının aynı olması gerekmez.

UNION, seçilen tüm sütunlar üzerinde çalışır.

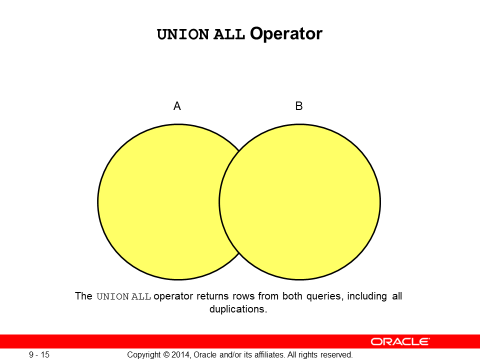
Yinelenen kontrol sırasında NULL değerler göz ardı edilmez.

Varsayılan olarak çıktı, SELECT yan tümcesinin sütunlarına göre artan düzende sıralanır.order by asc yani



Mevcut ve emekli tüm çalışanların iş ayrıntılarını görüntüleyin. Her işi yalnızca bir kez görüntüleyin

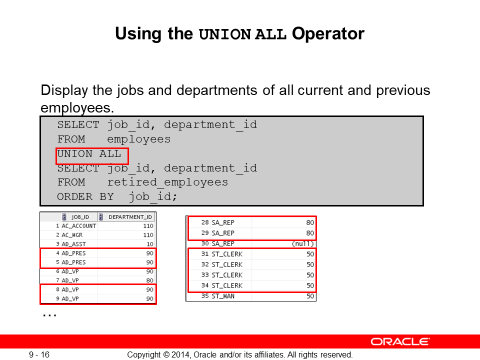
UNION operatörü, yinelenen kayıtları ortadan kaldırır. Hem EMPLOYEES hem de RETIRED\_EMPLOYEES tablolarında oluşan kayıtlar aynıysa, kayıtlar yalnızca bir kez görüntülenir.



Birden çok sorgudan tüm satırları döndürmek için UNION ALL operatörünü kullanın.

Yönergeler

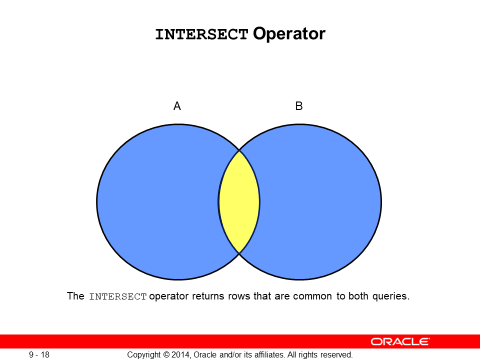
UNION ve UNION ALL için yönergeler, UNION ALL ile ilgili aşağıdaki iki istisna dışında aynıdır: UNION'dan farklı olarak, yinelenen satırlar elimine edilmez ve çıktı varsayılan olarak sıralanmaz



Örnekte 35 satır seçilmiştir. İki tablonun kombinasyonu toplam 35 satırdır. UNION ALL operatörü yinelenen satırları ortadan kaldırmaz. UNION, herhangi bir sorgu tarafından seçilen tüm farklı satırları döndürür. UNION ALL, tüm kopyalar dahil olmak üzere her iki sorgu tarafından seçilen tüm satırları döndürür. Şimdi UNION yan tümcesi ile yazılmış slayttaki sorguyu düşünün:

SELECT job\_id,department\_id  
 FROM employees  
 UNION  
 SELECT job\_id,department\_id  
 FROM retired\_employees  
 ORDER BY job\_id;

yukarıdaki sorgu 19 satır döndürür. Bunun nedeni, tüm yinelenen satırları ortadan kaldırmasıdır.



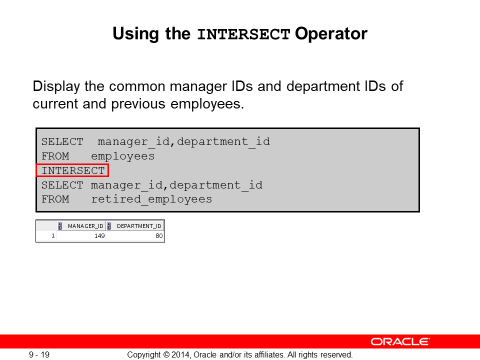
Birden çok sorguda ortak olan tüm satırları döndürmek için INTERSECT operatörünü kullanın.

Yönergeler

Sorgularda SELECT deyimleri tarafından seçilen sütunların sayısı ve veri türleri, sorguda kullanılan tüm SELECT deyimlerinde aynı olmalıdır. Bununla birlikte, sütunların adlarının aynı olması gerekmez.

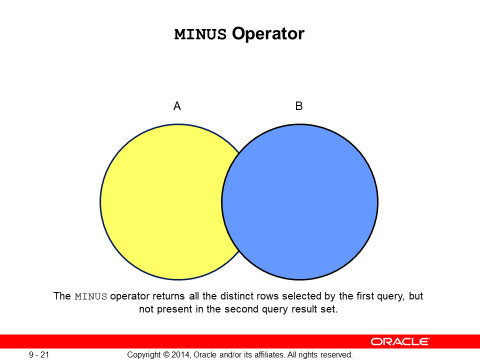
Kesişen tabloların sırasını tersine çevirmek sonucu değiştirmez.

INTERSECT, NULL değerleri yok saymaz.



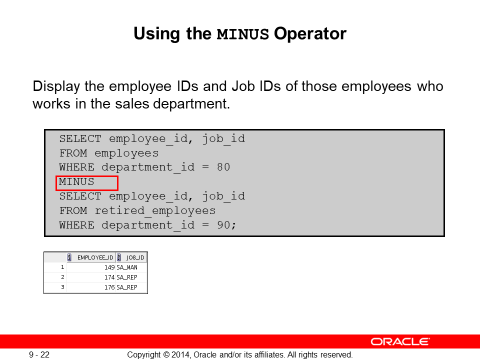
Mevcut ve önceki çalışanların ortak yönetici kimliklerini ve departman kimliklerini görüntüleyin.

Bu slayttaki örnekte, sorgu yalnızca her iki tabloda da seçili sütunlarda aynı değerlere sahip kayıtları döndürür



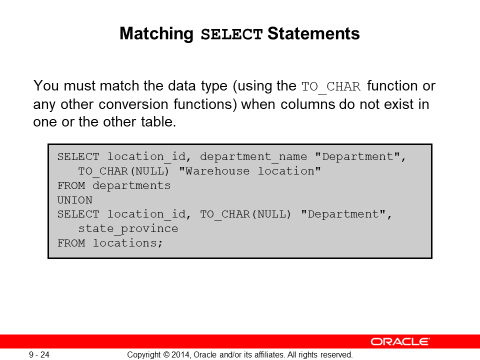
İlk sorgu tarafından seçilen, ancak ikinci sorgu sonuç kümesinde bulunmayan tüm farklı satırları döndürmek için EKSİ operatörünü kullanın (ilk SELECT ifadesi MINUS ikinci SELECT ifadesi).

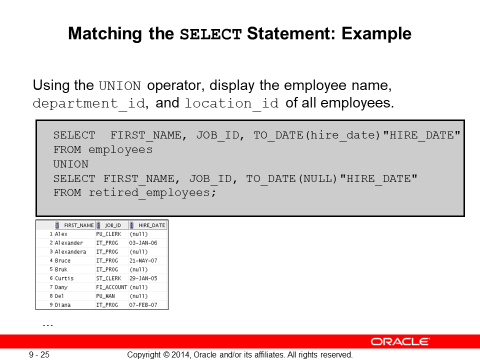
Not: Sorgularda kullanılan tüm SELECT deyimlerinde sütun sayısı aynı olmalı ve sorgularda SELECT deyimleri tarafından seçilen sütunların veri tipleri aynı veri tipi grubuna ait olmalıdır. Bununla birlikte, sütunların adlarının aynı olması gerekmez.



Satış departmanında çalışan bu çalışanların çalışan kimliklerini ve İş Kimliklerini görüntüleyin.?

Slayttaki örnekte, RETIRED\_EMLOYEES tablosundaki çalışan kimlikleri, EMPLOYEES tablosundakilerden çıkarılır. Sonuç kümesi, çıkarma işleminden sonra kalan çalışanları görüntüler; EMPLOYEES tablosunda bulunan, ancak RETIRED\_EMPLOYEES tablosunda bulunmayan satırlarla temsil edilirler. Satış departmanında çalışan personelin kayıtlarıdır.



Tablolardan birinde veya diğerinde sütunlar olmadığında, veri türünü (TO\_CHAR işlevini veya diğer dönüştürme işlevlerini kullanarak) eşleştirmeniz gerekir. Sorguların SELECT listelerindeki ifadelerin sayı olarak eşleşmesi gerektiğinden, bu kurala uymak için boş sütunları ve veri türü dönüştürme işlevlerini kullanabilirsiniz. Sütun listesini açıkça eşleştirmek için, her SELECT deyiminde seçili sütunların sayısı ve veri türüyle eşleşecek şekilde eksik konumlara NULL sütunlar ekleyebilirsiniz. Slaytta, warehouse adı, dummy sütun başlığı olarak verilmiştir. TO\_CHAR işlevi, ikinci sorgu tarafından alınan state\_province sütununun VARCHAR2 veri türünü eşleştirmek için ilk sorguda kullanılır. Benzer şekilde, ikinci sorgudaki TO\_CHAR işlevi, ilk sorgu tarafından alınan departman\_adı sütununun VARCHAR2 veri türünü eşleştirmek için kullanılır. 

EMPLOYEES ve RETIRED\_EMPLOYEES tablolarının birkaç ortak sütunu vardır (örneğin, EMPLOYEE\_ID, JOB\_ID ve DEPARTMENT\_ID). Ancak, sorgunun UNION operatörünü kullanarak FIRST\_NAME, JOB\_ID ve HIRE\_DATE'i görüntülemesini istiyorsanız, HIRE\_DATE'in yalnızca EMPLOYEES tablosunda bulunduğunu bilerek?

Slayttaki kod örneği, EMPLOYEES ve RETIRED\_EMPLOYEES tablolarındaki FIRST\_NAME ve JOB\_ID sütunlarıyla eşleşir. EMPLOYEES SELECT deyimindeki HIRE\_DATE sütunuyla eşleşmesi için RETIRED\_EMPLOYEES SELECT deyimine NULL eklenir.

Slaytta gösterilen sonuçlarda, RETIRED\_EMPLOYEES tablosundan bir kayda karşılık gelen çıktıdaki her satır HIRE\_DATE sütununda bir NULL içerir.

ORDER BY yan tümcesi bileşik sorguda yalnızca bir kez kullanılabilir. Kullanılırsa, ORDER BY yan tümcesi sorgunun sonuna yerleştirilmelidir. ORDER BY yan tümcesi, sütun adını veya bir diğer adı (alias)kabul eder. Varsayılan olarak çıktı, ilk SELECT sorgusunun ilk sütununda artan düzende sıralanır.

Not: ORDER BY yan tümcesi, ikinci SELECT sorgusunun sütun adlarını tanımıyor. Sütun adları üzerinde karışıklığı önlemek için, sütun konumlarına göre order by yaygın bir uygulamadır.

Örneğin, aşağıdaki ifadede çıktı artan job\_id sırasına göre gösterilecektir.

SELECT employee\_id, job\_id,salary

FROM employees

UNION

SELECT employee\_id, job\_id,0

FROM retired\_employees

ORDER BY 2;

SİPARİŞ BY atlarsanız, varsayılan olarak, çıktı employee\_id artan düzende sıralanır. Çıktıyı sıralamak için ikinci sorgudaki sütunları kullanamazsınız

SUMMARY

UNION operatörü, bileşik sorgudaki her sorgu tarafından seçilen tüm farklı satırları döndürür. Birden çok tablodan tüm satırları döndürmek ve yinelenen satırları ortadan kaldırmak için UNION operatörünü kullanın.

Birden çok sorgudan tüm satırları döndürmek için UNION ALL operatörünü kullanın. UNION operatörünün aksine, yinelenen satırlar elimine edilmez ve çıktı varsayılan olarak sıralanmaz.

Birden çok sorguda ortak olan tüm satırları döndürmek için INTERSECT operatörünü kullanın.

İkinci sorguda bulunmayan ilk sorgu tarafından döndürülen satırları döndürmek için MINUS operatörünü kullanın.

ORDER BY yan tümcesini yalnızca bileşik ifadenin en sonunda kullanmayı unutmayın.

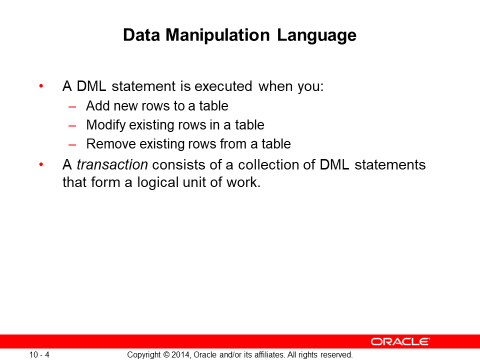
SELECT listelerindeki karşılık gelen ifadelerin sayı ve veri türü olarak eşleştiğinden emin olun.

After completing this lesson, you should be able to do the following:

Describe each data manipulation language (DML) statement

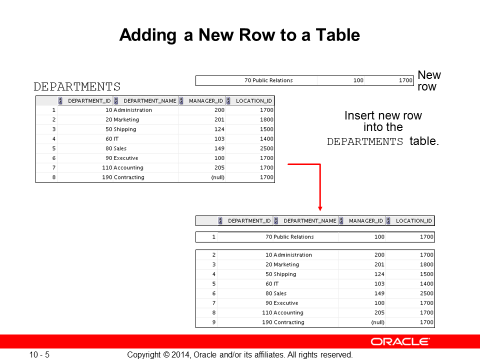
Control transactions DML statementlarını öğrencez LESSON AGENDA AŞAĞIDA

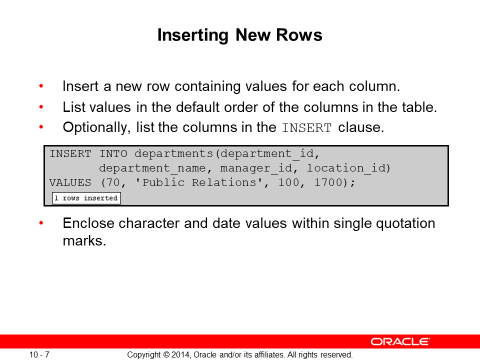




Aşağıdaki durumlarda bir DML ifadesi yürütülür: Bir tabloya yeni satırlar ekleyin Bir tablodaki mevcut satırları değiştirin Bir tablodan mevcut satırları kaldırma Bir işlem, mantıksal bir iş birimi oluşturan bir DML ifadeleri koleksiyonundan oluşur.

Veri işleme dili (DML), SQL'in temel bir parçasıdır. Veritabanına veri eklemek, güncellemek veya silmek istediğinizde bir DML ifadesi yürütürsünüz. Mantıksal bir iş birimi oluşturan DML ifadeleri koleksiyonuna işlem denir. Bir bankacılık veritabanı düşünün. Bir banka müşterisi bir tasarruf hesabından bir çek hesabına para transfer ettiğinde, işlem üç ayrı işlemden oluşabilir: tasarruf hesabının azaltılması, çek hesabının arttırılması ve işlemin işlem günlüğüne kaydedilmesi. Oracle sunucusu, hesapları uygun dengede tutmak için üç SQL ifadesinin de gerçekleştirildiğini garanti etmelidir. İşlemdeki deyimlerden birinin yürütülmesini engelleyen bir şey olduğunda, işlemin diğer deyimleri geri alınmalıdır. Notlar Bu dersteki DML ifadelerinin çoğu, tablodaki hiçbir kısıtlamanın ihlal edilmediğini varsayar. Kısıtlamalar bu derste daha sonra tartışılacaktır. SQL Developer'da, Komut Dosyasını Çalıştır simgesine tıklayın veya DML deyimlerini çalıştırmak için [F5]'e basın. Geri bildirim mesajları, Komut Dosyası Çıktısı sekmeli sayfasında gösterilecektir.





Her sütun için değerleri içeren yeni bir satır ekleyin.

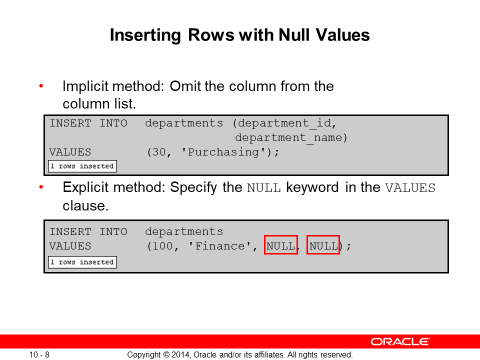
Tablodaki sütunların varsayılan sırasına göre değerleri listeleyin.

İsteğe bağlı olarak, INSERT yan tümcesindeki sütunları listeleyin. Yani valuesi tablodaki aynı sırayla ekliyosam insert yazmama ınsert ınto da kolon isimlerini girmeme gerek yok

Karakter ve tarih değerlerini tek tırnak içine alın. Her sütun için değerler içeren yeni bir satır ekleyebileceğiniz için, INSERT yan tümcesinde sütun listesi gerekli değildir. Ancak sütun listesini kullanmazsanız, değerler tablodaki sütunların varsayılan sırasına göre listelenmeli ve her sütun için bir değer sağlanmalıdır.

departmanları TANIMLAYIN

Anlaşılır olması için INSERT yan tümcesindeki sütun listesini kullanın. Karakter ve tarih değerlerini tek tırnak içine alın; ancak sayısal değerleri tek tırnak içine almanız önerilmez.



Implicit method: Sütunu sütun listesinden çıkarın.

Explicit method: VALUES yan tümcesinde NULL anahtar sözcüğünü belirtin.

DESCRIBE komutuyla Null durumunu doğrulayarak hedeflenen sütunda boş değerler kullanabildiğinizden emin olun.yani null kullandığım sütün null değeri kabul ediyomu

Oracle sunucusu, tüm veri türlerini, veri aralıklarını ve veri bütünlüğü kısıtlamalarını otomatik olarak uygular. Açıkça listelenmeyen herhangi bir sütun, kullanılan eksik sütunlar için varsayılan değerleriniz yoksa, yeni satırda boş bir değer alır.

Kullanıcı girişi sırasında oluşabilecek yaygın hatalar aşağıdaki sırayla kontrol edilir:

NOT NULL sütunu için zorunlu değer eksik mi ?

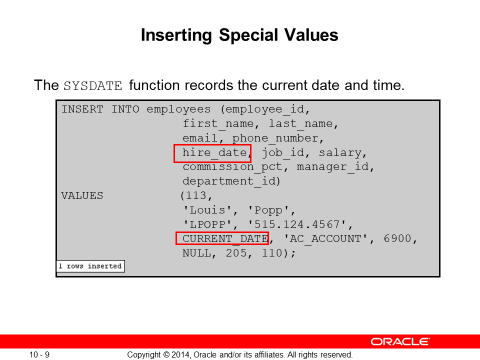
Herhangi bir benzersiz veya birincil anahtar kısıtlamasını ihlal eden yinelenen değer

Bir CHECK kısıtlamasını ihlal eden herhangi bir değer

Yabancı anahtar kısıtlaması için bilgi bütünlüğü korunur

Veri türü uyumsuzlukları veya bir sütuna sığmayacak kadar geniş değerler

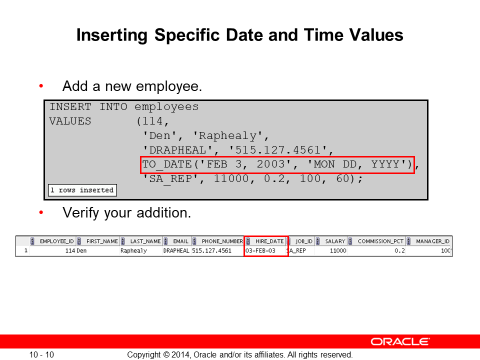
Not: INSERT ifadesini daha okunabilir ve güvenilir hale getirdiği veya hataya daha az açık hale getirdiği için sütun listesinin kullanılması önerilir.



İnsert ederken special değerler de kullanabilirm mesela hire date kolunu CURRENT\_DATE dersem şimdiki tarih ve zamanı atar.

eğerler girmek için fonksiyonları kullanabilirsiniz.

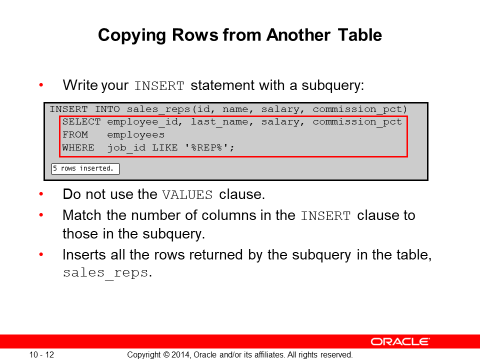
Slayt örneği, ÇALIŞANLAR tablosunda çalışan Popp için bilgileri kaydeder. HIRE\_DATE sütununda geçerli tarih ve saati sağlar. Oturum saat diliminde geçerli tarihi döndüren CURRENT\_DATE işlevini kullanır. Bir tabloya satır eklerken de USER fonksiyonalrını kullanabilirsiniz. USER fonksiyonları, geçerli kullanıcı adını kaydeder.



DD-MON-RR formatı genellikle bir tarih değeri eklemek için kullanılır. RR formatı ile sistem doğru yüzyılı otomatik olarak sağlar.

Tarih değerini DD-MON-YYYY formatında da sağlayabilirsiniz. Bu, yüzyılı açıkça belirttiği ve doğru yüzyılı belirtmenin dahili RR biçimi mantığına bağlı olmadığı için önerilir.

Varsayılan biçimden farklı bir biçimde bir tarih girilmesi gerekiyorsa (örneğin, başka bir yüzyıl veya belirli bir saat), TO\_DATE işlevini kullanmanız gerekir.

Slayttaki örnek, ÇALIŞANLAR tablosunda çalışan Raphealy için bilgileri kaydeder. HIRE\_DATE sütununu 3 Şubat 2003 olarak ayarlar. 

Değerlerin mevcut tablolardan türetildiği bir tabloya satır eklemek için INSERT ifadesini kullanabilirsiniz. Slayttaki örnekte, INSERT INTO ifadesinin çalışması için, CREATE TABLE ifadesini kullanarak sales\_reps tablosunu önceden oluşturmuş olmanız gerekir. CREATE TABLE “Veri Tanımlama Dili'ne Giriş” başlıklı derste işlenmiştir.

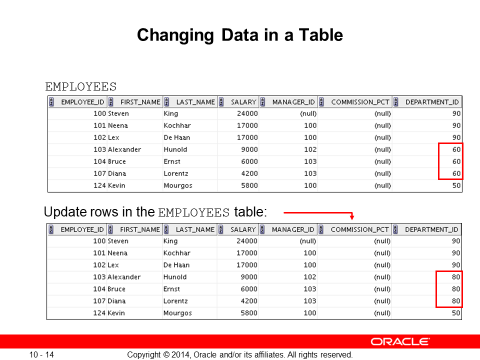
VALUES yan tümcesi yerine bir alt sorgu kullanırsınız.

INSERT yan tümcesinin sütun listesindeki sütun sayısı ve veri türleri, alt sorgudaki değer sayısı ve veri türleri ile eşleşmelidir. Alt sorgu tarafından döndürülen satır sayısına bağlı olarak sıfır veya daha fazla satır eklenir. Bir tablonun satırlarının bir kopyasını oluşturmak için alt sorguda SELECT \* kullanın:

INSERT INTO copy\_emp

SELECT \*

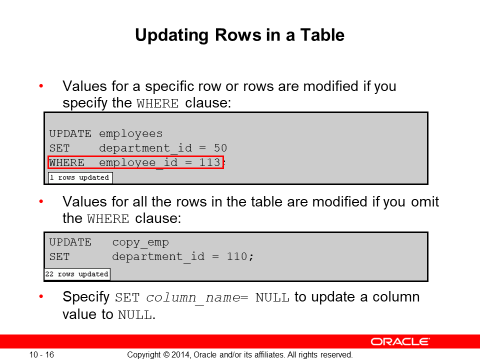
FROM employees;



Departmant id si 90 olan adamın department ıd sini 80 yapma

UPDATE

Not: Genel olarak, güncelleme için tek bir satırı tanımlamak için WHERE yan tümcesindeki primary key sütununu kullanın. Diğer sütunların kullanılması beklenmedik bir şekilde birkaç satırın güncellenmesine neden olabilir. Örneğin, ÇALIŞANLAR tablosunda tek bir satırın ismiyle tanımlanması, aynı ada sahip birden fazla çalışanın dönmesine neden olabilir.



WHERE yan tümcesini belirtirseniz, belirli bir satır veya satırların değerleri değiştirilir:

WHERE yan tümcesini atlarsanız, tablodaki tüm satırların değerleri değiştirilir:

Bir sütun değerini NULL olarak güncellemek için SET column\_name= NULL belirtin.

İlk örnekte tek satır seçili employee\_id primary key olduğu için where şartında omu kullandım ve tek bir satırın gelceğini garantiledim firsT\_name deseydim birden fazla satırı benim istemedğim kişiyide güncelleyebilirde.

UPDATE ifadesi, WHERE yan tümcesi belirtilmişse belirli bir satırın veya satırların değerlerini değiştirir. Slayttaki örnek, çalışan 113'ün (Popp) 50 numaralı bölüme transferini göstermektedir.

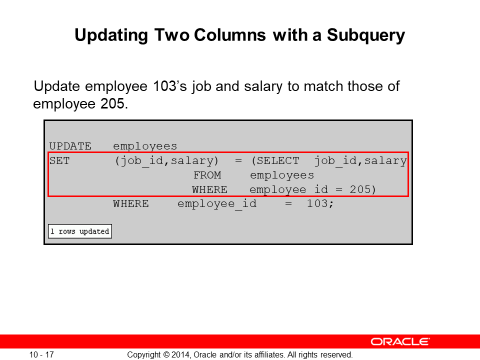
WHERE yan tümcesini atlarsanız, tablodaki tüm satırların değerleri değiştirilir. COPY\_EMP tablosundaki güncellenmiş satırları inceleyin.

SELECT last\_name, department\_id

FROM copy\_emp; diyerek güncellemeleri control edebiliriz.  
Örneğin, SA\_REP olan bir çalışan artık işini IT\_PROG olarak değiştirmiştir. Bu nedenle, JOB\_ID'sinin güncellenmesi ve komisyon alanının NULL olarak ayarlanması gerekiyor.

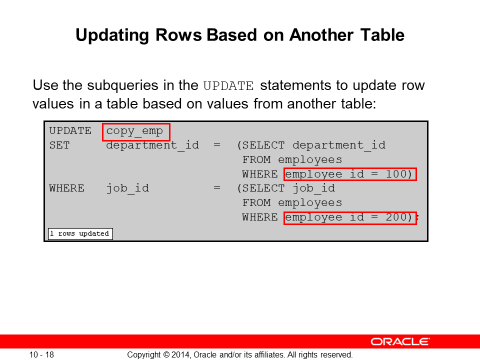
UPDATE employees SET job\_id = ‘IT\_PROG’,commission\_pct=NULL WHERE employee\_id=114

copy\_emp tablosunun dataları employees tablosunun dataları ile aynı

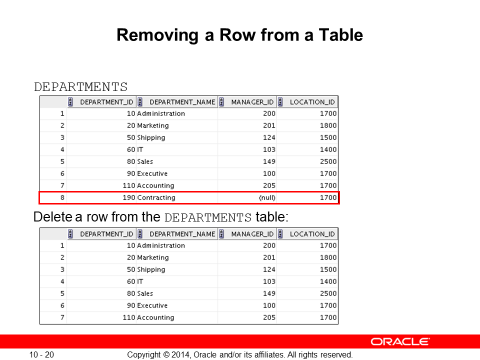


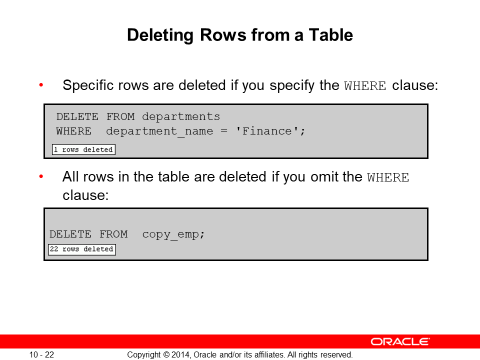
You can update multiple columns in the SET clause of an UPDATE statement by writing multiple subqueries. The syntax is as follows:

Çalışan 103'ün işini ve maaşını, çalışan 205'inkilerle eşleşecek şekilde güncelleyin.



employee\_id si 200 olan adamın job id si ne ise job id si o olan çalışanın departmanını employee\_idsi 100 olan adamın departmanı ne ise onla güncelle





DELETE deyiminde WHERE yan tümcesini belirterek belirli satırları silebilirsiniz. Slayttaki ilk örnek, BÖLÜMLER tablosundan muhasebe departmanını siler. SELECT deyimini kullanarak silinen satırları görüntüleyerek silme işlemini onaylayabilirsiniz.

SELECT \*

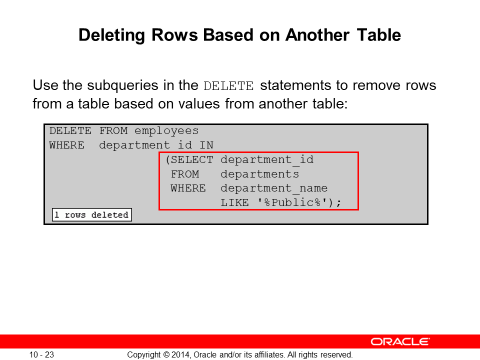
FROM departments

WHERE department\_name = 'Finance'; silindiğini burdan görebilirim

Ancak, WHERE yan tümcesini atlarsanız, tablodaki tüm satırlar silinir. Slayttaki ikinci örnek, WHERE yan tümcesi belirtilmediği için COPY\_EMP tablosundaki tüm satırları siler. Örnek WHERE yan tümcesinde tanımlanan satırları kaldırın.

DELETE FROM employees WHERE employee\_id = 114;

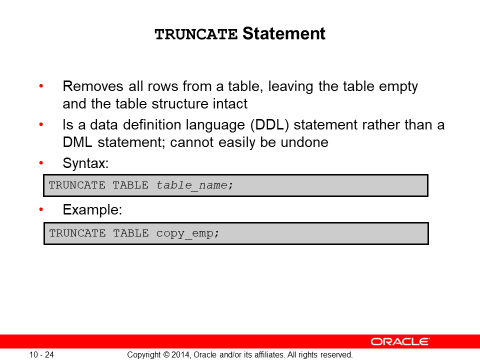
DELETE FROM departments WHERE department\_id IN (30, 40);



Başka bir tablodaki değerlere dayalı olarak bir tablodan satırları silmek için alt sorguları kullanabilirsiniz. Slayttaki örnek, departman adının Public dizesini içerdiği bir departmandaki tüm çalışanları siler.

Alt sorgu, Public dizesini içeren departman adına göre departman numarasını bulmak için DEPARTMENTS tablosunu arar. Alt sorgu daha sonra departman numarasını, bu departman numarasına dayalı olarak ÇALIŞANLAR tablosundan veri satırlarını silen ana sorguya besler.

Eşittir ve IN aynı şey eşittir de koysam çalışırdı



Bir tabloyu boşaltmanın daha etkili bir yöntemi TRUNCATE deyimini kullanmaktır. Bir tablodan veya kümeden tüm satırları hızla kaldırmak için TRUNCATE deyimini kullanabilirsiniz. TRUNCATE deyimiyle satırları kaldırmak, aşağıdaki nedenlerle DELETE deyimiyle satırları kaldırmaktan daha hızlıdır:

TRUNCATE deyimi bir veri tanımlama dili (DDL) deyimidir ve rollback bilgisi oluşturmaz. rollback bilgileri bu derste daha sonra ele alınacaktır.

Tablo bir başvuru bütünlüğü constraintin üst öğesiyse, yabi foreign keyi siliyosan tabloyu kesemezsiniz. TRUNCATE deyimini yayınlamadan önce constrainti devre dışı bırakmanız gerekir. Devre dışı bırakma constraint leri “DDL İfadelerine Giriş” başlıklı derste ele alınmıştır.

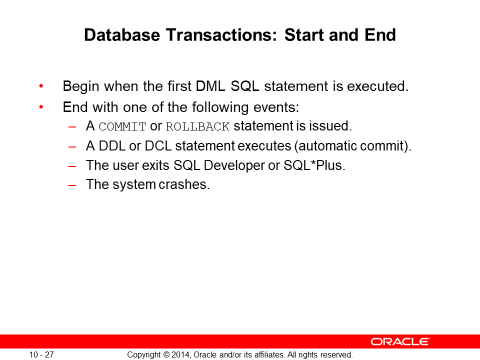
DATABASE TRANSACTIONS

Oracle sunucusu, işlemlere dayalı veri tutarlılığı sağlar. İşlemler, verileri değiştirirken size daha fazla esneklik ve kontrol sağlar ve kullanıcı işlemi hatası veya sistem hatası durumunda veri tutarlılığını sağlar.

İşlemler, verilerde tutarlı bir değişiklik oluşturan DML ifadelerinden oluşur. Örneğin, iki hesap arasındaki para transferinde, bir hesaptaki borç ve aynı tutarda başka bir hesaba alacak dahil edilmelidir. Her iki eylem de ya başarısız olmalı ya da birlikte başarılı olmalıdır; kredi borç olmadan taahhüt edilmemelidir.

TRANSACTION TYPE





Bir veritabanı işlemi ne zaman başlar ve biter?

Bir işlem, ilk DML ifadesi ile karşılaşıldığında başlar ve aşağıdakilerden biri gerçekleştiğinde sona erer:

Bir COMMIT veya ROLLBACK ifadesi yayınlanır.

CREATE gibi bir DDL deyimi yayınlanır.

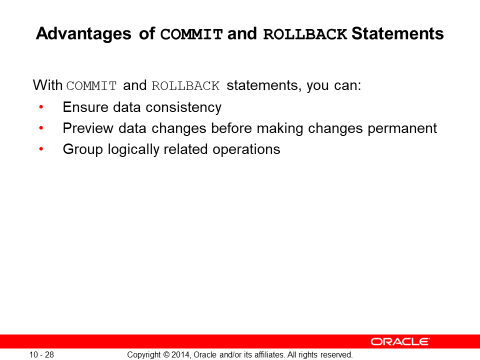
Bir DCL bildirimi yayınlandı.

Kullanıcı, SQL Developer veya SQL\*Plus'tan çıkar.

Bir makine arızalanır veya sistem çöker.

Bir işlem sona erdikten sonra, bir sonraki yürütülebilir SQL ifadesi otomatik olarak bir sonraki işlemi başlatır.

Bir DDL deyimi veya bir DCL deyimi otomatik olarak taahhüt edilir ve bu nedenle bir işlemi örtülü olarak sona erdirir.



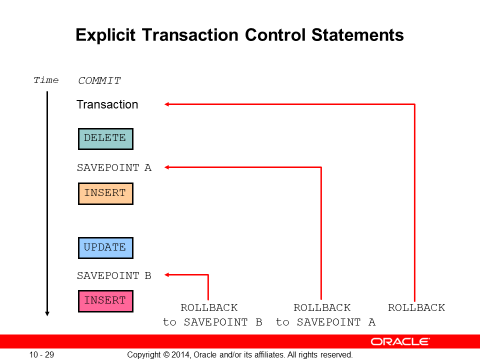
COMMIT ve ROLLBACK ifadeleriyle şunları yapabilirsiniz:

Veri tutarlılığını sağlayın

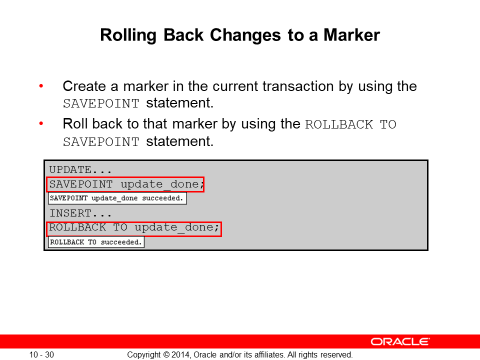
Değişiklikleri kalıcı hale getirmeden önce veri değişikliklerini önizleyin

Mantıksal olarak ilgili işlemleri gruplandırın

COMMIT ve ROLLBACK ifadeleriyle, verilerde kalıcı değişiklikler yapma kontrolünüz vardır.



COMMIT, SAVEPOINT ve ROLLBACK deyimlerini kullanarak işlemlerin mantığını kontrol edebilirsiniz.



Hareketi daha küçük bölümlere ayıran SAVEPOINT deyimini kullanarak mevcut işlemde bir işaret oluşturabilirsiniz. Ardından, ROLLBACK TO SAVEPOINT deyimini kullanarak, bu işaretleyicide bekleyen değişiklikleri geri atabilirsiniz.

Daha önceki bir SAVEPOINT aynı ada sahip ikinci bir SAVEPOINT oluşturursanız, önceki SAVEPOINT noktasının silineceğini unutmayın.

**Implicit Transaction Processing**

Aşağıdaki durumlarda otomatik bir taahhüt gerçekleşir: Bir DDL beyanı yayınlandı Bir DCL beyanı yayınlandı COMMIT veya ROLLBACK deyimlerini açıkça yayınlamadan SQL Developer veya SQL\*Plus'tan normal çıkış SQL Developer veya SQL\*Plus'ın anormal bir şekilde sonlandırılması veya bir sistem arızası olduğunda otomatik geri alma gerçekleşir.



Not: SQL\*Plus'ta AUTOCOMMIT komutu AÇIK veya KAPALI olarak değiştirilebilir. ON olarak ayarlanırsa, her bir DML ifadesi yürütülür yürütülmez işlenir. Değişiklikleri geri alamazsınız. KAPALI olarak ayarlanırsa, COMMIT ifadesi yine de açıkça verilebilir. Ayrıca, bir DDL ifadesi yayınlandığında veya SQL\*Plus'tan çıktığınızda COMMIT ifadesi yayınlanır. SET AUTOCOMMIT ON/OFF komutu SQL Developer'da atlanır. DML, yalnızca Otomatik tamamlama tercihini etkinleştirdiyseniz, SQL Developer'dan normal bir çıkışta işlenir. Autocommit'i etkinleştirmek için aşağıdakileri yapın:

Araçlar menüsünde Tercihler'i seçin. Tercihler iletişim kutusunda Veritabanı'nı genişletin ve Çalışma Sayfası Parametreleri'ni seçin.

Sağ bölmede, “SQL Çalışma Sayfasında Otomatik İşlem Yap” seçeneğini seçin. Tamam'ı tıklayın.

**Sistem Arızaları**

Bir işlem bir sistem hatası nedeniyle kesintiye uğradığında, işlemin tamamı otomatik olarak geri alınır. Bu, hatanın verilerde istenmeyen değişikliklere neden olmasını önler ve tabloları son işleme sırasındaki durumuna döndürür. Bu sayede Oracle sunucusu tabloların bütünlüğünü koruriSQL Developer'da, Dosya menüsünden Çıkış seçilerek oturumdan normal bir çıkış gerçekleştirilir. SQL\*Plus'ta, istemde EXIT komutu girilerek normal bir çıkış gerçekleştirilir. Pencerenin kapatılması anormal bir çıkış olarak yorumlanır.

**State of Data Before COMMIT or ROLLBACK**

Verilerin önceki durumu kurtarılabilir.

Geçerli oturum, SELECT deyimini kullanarak DML işlemlerinin sonuçlarını gözden geçirebilir.

Diğer oturumlar, geçerli oturum tarafından verilen DML ifadelerinin sonuçlarını görüntüleyemez.

Etkilenen satırlar kilitlenir; diğer oturum, etkilenen satırlardaki verileri değiştiremez.

İşlem sırasında yapılan her veri değişikliği işlem gerçekleşene kadar geçicidir.

COMMIT veya ROLLBACK deyimleri yayınlanmadan önce verilerin durumu aşağıdaki gibi tanımlanabilir:

Veri işleme işlemleri(DML) öncelikle veritabanı arabelleğini etkiler; bu nedenle, verilerin önceki durumu kurtarılabilir.

Mevcut oturum, tabloları sorgulayarak veri işleme işlemlerinin sonuçlarını inceleyebilir.

Diğer oturumlar, geçerli oturum tarafından yapılan veri işleme işlemlerinin sonuçlarını görüntüleyemez. Oracle sunucusu, her oturumun verileri son işlemde olduğu gibi görmesini sağlamak için okuma tutarlılığı kurar.

Etkilenen satırlar kilitlenir; diğer oturum, etkilenen satırlardaki verileri değiştiremez.

**State of Data After COMMIT**

COMMIT deyimini kullanarak bekleyen tüm değişiklikleri kalıcı hale getirin. Bir COMMIT ifadesinden sonra ne olur:

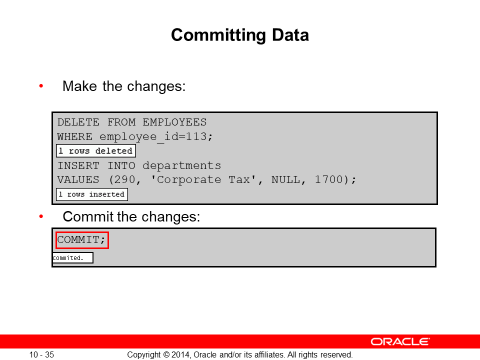
Veri değişiklikleri veritabanına yazılır.

Verilerin önceki durumu artık normal SQL sorgularında mevcut değildir.

Tüm oturumlar işlemin sonuçlarını görüntüleyebilir.

Etkilenen satırlardaki kilitler serbest bırakılır; satırlar artık yeni veri değişiklikleri gerçekleştirmek için diğer oturumlar için kullanılabilir.

Tüm kayıt noktaları silinir.



In the example in the slide, a row is deleted from the EMPLOYEES table and a new row is inserted into the DEPARTMENTS table. The changes are saved by issuing the COMMIT statement.

**Example**

Remove departments 290 and 300 in the DEPARTMENTS table and update a row in the EMPLOYEES table. Save the data change.

DELETE FROM employees WHERE department\_id IN(290,300)

UPDATE employees

SET department\_id = 80

WHERE employee\_id = 206;

COMMIT

**State of Data After ROLLBACK**

DELETE FROM copy\_emp;

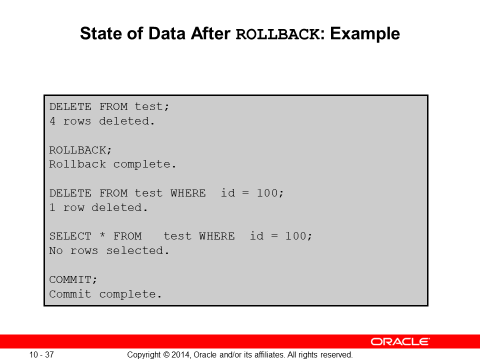
ROLLBACK ;

Aşağıdakilerle sonuçlanan ROLLBACK deyimini kullanarak bekleyen tüm değişiklikleri atın:

Veri değişiklikleri geri alındı.

Verilerin önceki durumu geri yüklenir.

Etkilenen satırlardaki kilitler serbest bırakılır.



TEST tablosundan bir kaydı kaldırmaya çalışırken yanlışlıkla tabloyu boşaltabilirsiniz. Ancak hatayı düzeltebilir, uygun bir ifadeyi yeniden düzenleyebilir ve veri değişikliğini kalıcı hale getirebilirsiniz.

**Statement-Level Rollback**

Yürütme sırasında tek bir DML ifadesi başarısız olursa, yalnızca bu ifade geri alınır.

Oracle sunucusu, implicit savepoint uygular.

Diğer tüm değişiklikler korunur.

Kullanıcı, bir COMMIT veya ROLLBACK deyimi yürüterek işlemleri açıkça sonlandırmalıdır.

Bir ifade yürütme hatası algılanırsa, işlemin bir kısmı örtük bir geri alma yoluyla atılabilir. Bir işlemin yürütülmesi sırasında tek bir DML ifadesi başarısız olursa, bunun etkisi ifade düzeyinde bir geri alma ile geri alınır, ancak işlemdeki önceki DML ifadeleri tarafından yapılan değişiklikler atılmaz. Kullanıcı tarafından açıkça taahhüt edilebilir veya geri alınabilirler.

Oracle sunucusu, herhangi bir DDL deyiminden önce ve sonra örtülü bir taahhüt verir. Bu nedenle, DDL ifadeniz başarıyla yürütülmese bile, sunucu bir taahhüt yayınladığı için önceki ifadeyi geri alamazsınız.

Bir COMMIT veya ROLLBACK ifadesi yürüterek işlemlerinizi açıkça sonlandırın.

**Read Consistency**

**Okuma tutarlılığı, verilerin her zaman tutarlı bir şekilde görüntülenmesini garanti eder.**

**Bir kullanıcı tarafından yapılan değişiklikler, başka bir kullanıcı tarafından yapılan değişikliklerle çelişmez.**

**Okuma tutarlılığı, aynı veriler üzerinde şunları sağlar:**

* + - **Readers do not wait for writers**
    - **Writers do not wait for readers**
    - **Writers wait for writers**

**Veritabanı kullanıcıları veritabanına iki şekilde erişir:**

**Okuma işlemleri (SELECT ifadesi)**

**Yazma işlemleri (INSERT, UPDATE, DELETE deyimleri)**

**Aşağıdakilerin gerçekleşmesi için okuma tutarlılığına ihtiyacınız vardır:**

**Veritabanı okuyucusu ve yazıcısı, verilerin tutarlı bir şekilde görüntülenmesini sağlar.**

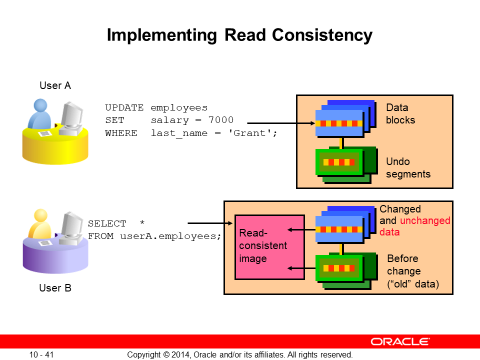
**Okuyucular, değiştirilme sürecinde olan verileri görmezler.**

**Yazarların, veritabanındaki değişikliklerin tutarlı bir şekilde yapılması sağlanır.**

**Bir yazar tarafından yapılan değişiklikler, başka bir yazar tarafından yapılan değişiklikleri bozmaz veya bunlarla çelişmez.**

**Okuma tutarlılığının amacı, bir DML işlemi başlamadan önce her kullanıcının verileri son işlemde olduğu gibi görmesini sağlamaktır.**

**Not: Aynı kullanıcı farklı oturumlarda oturum açabilir. Her oturum, aynı kullanıcılar olsalar bile, yukarıda açıklanan şekilde okuma tutarlılığını korur.**

****

**Okuma tutarlılığı otomatik bir uygulamadır. Veritabanının kısmi bir kopyasını geri alma bölümlerinde tutar. Tutarlı okuma görüntüsü, tablodaki taahhüt edilen verilerden ve değiştirilmekte olan ve henüz geri alma segmentinden kaydedilmeyen eski verilerden oluşturulur.**

**Veritabanı üzerinde bir ekleme, güncelleme veya silme işlemi yapıldığında, Oracle sunucusu verilerin değiştirilmeden önce bir kopyasını alır ve bir geri alma segmentine yazar.**

**Değişikliği yayınlayan dışındaki tüm okuyucular, veritabanını değişiklikler başlamadan önceki haliyle görür; geri alınan segmentin verilerin "anlık görüntüsünü" görüntülerler.**

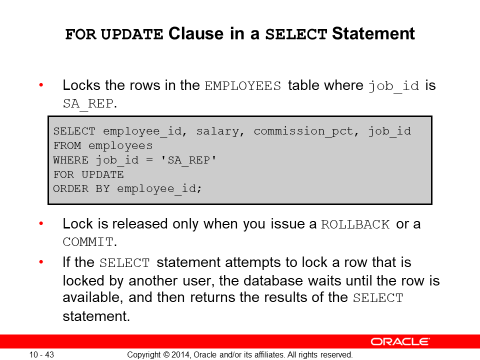
**Değişiklikler veritabanına işlenmeden önce, yalnızca verileri değiştiren kullanıcı, değişiklikleri içeren veritabanını görür. Diğer herkes, geri alma bölümünde anlık görüntüyü görür. Bu, veri okuyucularının şu anda değişime uğramayan tutarlı verileri okumasını garanti eder.**

**Bir DML ifadesi taahhüt edildiğinde, veritabanında yapılan değişiklik, taahhüt tamamlandıktan sonra bir SELECT ifadesi yayınlayan herkes tarafından görünür hale gelir. Geri alma segmenti dosyasındaki eski veriler tarafından işgal edilen alan, yeniden kullanım için serbest bırakılır.**

**İşlem geri alınırsa, değişiklikler geri alınır:**

**Geri alma bölümündeki verilerin orijinal, eski sürümü tabloya geri yazılır.**

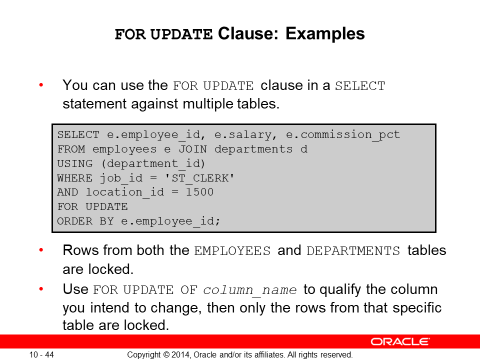
**Tüm kullanıcılar veritabanını işlem başlamadan önceki haliyle görür.**

****

**Bazı kayıtları sorgulamak için veritabanına karşı bir SELECT ifadesi yayınladığınızda, seçilen satırlara hiçbir kilit yerleştirilmez. Genel olarak, herhangi bir zamanda kilitlenen kayıtların sayısı (varsayılan olarak) mutlak minimumda tutulduğundan bu gereklidir: yalnızca değiştirilmiş ancak henüz commit edilmemiş kayıtlar kilitlenir. O zaman bile, diğerleri bu kayıtları değişiklikten önce göründükleri gibi okuyabileceklerdir (verinin “önceki görüntüsü”). Ancak, programınızda onları değiştirmeden önce bile bir dizi kaydı kilitlemek isteyebileceğiniz zamanlar vardır. Oracle, bu kilitlemeyi gerçekleştirmek için SELECT ifadesinin FOR UPDATE yan tümcesini sunar.**

**Bir SELECT...FOR UPDATE ifadesi yayınladığınızda, ilişkisel veritabanı yönetim sistemi (RDBMS), SELECT ifadesi tarafından tanımlanan tüm satırlarda otomatik olarak özel satır düzeyinde kilitler alır ve böylece kayıtları "yalnızca sizin değişiklikleriniz için" tutar. Siz bir Rollback veya COMMIT gerçekleştirmedikçe, başka hiç kimse bu kayıtlardan herhangi birini değiştiremez.**

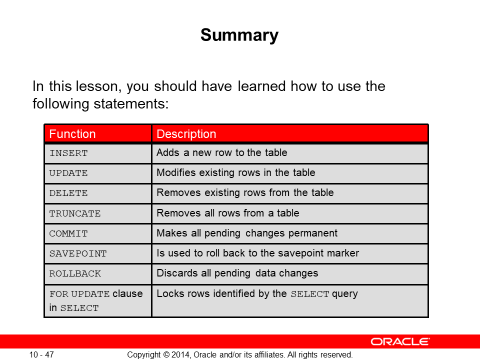
**Oracle sunucusuna tablo başka bir kullanıcı tarafından kilitlendiğinde beklememesini söylemek için FOR UPDATE yan tümcesine isteğe bağlı NOWAIT anahtar sözcüğünü ekleyebilirsiniz. Bu durumda, kontrol hemen programınıza veya SQL Developer ortamınıza döndürülür, böylece başka işler yapabilirsiniz veya tekrar denemeden önce bir süre bekleyin. NOWAIT yan tümcesi olmadan, kilitler diğer kullanıcı tarafından bir COMMIT veya bir ROLLBACK komutu aracılığıyla serbest bırakıldığında, tablo kullanılabilir duruma gelene kadar işleminiz engellenir.**



Slayttaki örnekte, ifade, JOB\_ID'si ST\_CLERK ve LOCATION\_ID'si 1500 olarak ayarlanmış ÇALIŞANLAR tablosundaki satırları kilitler ve LOCATION\_ID'deki departmanlar 1500 olarak ayarlanmış DEPARTMENTS tablosundaki satırları kilitler.

Değiştirmeyi düşündüğünüz sütunu nitelemek için FOR UPDATE OF column\_name kullanabilirsiniz. FOR UPDATE yan tümcesinin OF listesi, yalnızca seçili satırların sütunlarını değiştirmenizi kısıtlamaz. Kilitler hala tüm sıralara yerleştirilmiştir; sorguda sadece FOR UPDATE belirtirseniz ve OF anahtar sözcüğünden sonra bir veya daha fazla sütun eklemezseniz, veritabanı FROM yan tümcesinde listelenen tüm tablolarda tanımlanan tüm satırları kilitleyecektir.

Aşağıdaki ifade, yalnızca LOCATION\_ID 1500'de bulunan ST\_CLERK ile ÇALIŞANLAR tablosundaki satırları kilitler. DEPARTMENTS tablosunda hiçbir satır kilitlenmez:



Bu derste, INSERT, UPDATE, DELETE ve TRUNCATE deyimlerini kullanarak Oracle veritabanındaki verileri nasıl değiştireceğinizi ve ayrıca COMMIT, SAVEPOINT ve ROLLBACK deyimlerini kullanarak veri değişikliklerini nasıl kontrol edeceğinizi öğrenmiş olmalısınız. Yalnızca değişiklikleriniz için satırları kilitlemek için SELECT deyiminin FOR UPDATE yan tümcesini nasıl kullanacağınızı da öğrendiniz.

Oracle sunucusunun her zaman tutarlı bir veri görünümünü garanti ettiğini unutmayın.

INDTRODUCTION DATA DEFINITION LANGUAGE

Bu derste, veri tanımlama dili (DDL) ifadeleriyle tanışacaksınız. Basit tabloların nasıl oluşturulacağını, değiştirileceğini ve kaldırılacağının temellerini öğrenirsiniz. DDL'de bulunan veri türleri gösterilir ve şema kavramları tanıtılır. Kısıtlamalar bu derste tartışılmaktadır. DML işlemleri sırasında kısıtlamaları ihlal ederek oluşturulan istisna mesajları gösterilir ve açıklanır.

DATABASE OBJECTS

Oracle veritabanı birden fazla veri yapısı içerebilir. Her yapı, veritabanı geliştirmenin inşa aşamasında oluşturulabilmesi için veritabanı tasarımında ana hatlarıyla belirtilmelidir.

Tablo: Verileri depolar

View: Bir veya daha fazla tablodaki verilerin bir alt kümesidir.

Sequence: Sayısal değerler üretir

INDEX: Bazı sorguların performansını iyileştirir

Synonym: Bir nesneye alternatif bir ad verir

Oracle Tablo Yapıları

Kullanıcılar veritabanını kullanırken bile tablolar herhangi bir zamanda oluşturulabilir.

Bir tablonun boyutunu belirtmeniz gerekmez. Boyut, nihai olarak, bir bütün olarak veritabanına ayrılan alan miktarı ile tanımlanır. Bununla birlikte, bir tablonun zaman içinde ne kadar alan kullanacağını tahmin etmek önemlidir.

Tablo yapısı çevrimiçi olarak değiştirilebilir.

Not: Daha fazla veritabanı nesnesi mevcuttur, ancak bu kursta kapsanmamaktadır.

Veritabanı tablolarını ve sütunlarını, herhangi bir Oracle veritabanı nesnesini adlandırmak için standart kurallara göre adlandırırsınız.

Tablo adları ve sütun adları bir harfle başlamalı ve 1-30 karakter uzunluğunda olmalıdır.

Adlar yalnızca A–Z, a–z, 0–9, \_ (alt çizgi), $ ve # karakterlerini içermelidir (yasal karakterler, ancak kullanımları önerilmez).

Adlar, aynı Oracle sunucusu kullanıcısına ait başka bir nesnenin adını çoğaltmamalıdır.

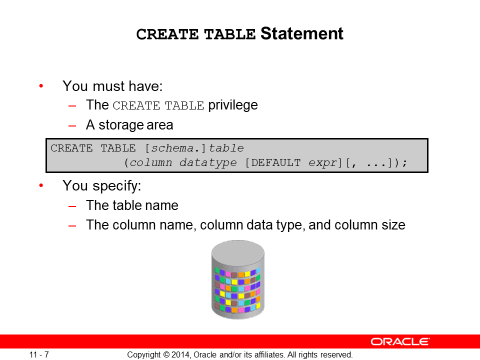
Adlar, Oracle sunucusu tarafından ayrılmış bir kelime olmamalıdır.

Bir nesnenin adını temsil etmek için alıntılanan tanımlayıcıları da kullanabilirsiniz. Alıntılanan bir tanımlayıcı çift tırnak (“”) ile başlar ve biter. Bir şema nesnesini tırnak içine alınmış bir tanımlayıcı kullanarak adlandırırsanız, o nesneye her başvurduğunuzda çift tırnak işareti kullanmanız gerekir. Alıntılanan tanımlayıcılar, bu önerilmese de ayrılmış kelimeler olabilir.

Adlandırma Yönergeleri

Tablolar ve diğer veritabanı nesneleri için açıklayıcı adlar kullanın.

Not: Adlar büyük/küçük harfe duyarlı değildir. Örneğin, EMPLOYEES, eMPloyees or eMpLOYEES ile aynı ad olarak kabul edilir. Ancak, alıntılanan tanımlayıcılar büyük/küçük harf duyarlıdır.



SQL CREATE TABLE ifadesini çalıştırarak verileri depolamak için tablolar oluşturursunuz. Bu ifade, Oracle Veritabanı yapılarını oluşturmak, değiştirmek veya kaldırmak için kullanılan SQL ifadelerinin bir alt kümesi olan DDL ifadelerinden biridir. Bu ifadeler veri tabanı üzerinde anında bir etkiye sahiptir ve ayrıca veri sözlüğündeki bilgileri de kaydederler.

Bir tablo oluşturmak için, kullanıcının CREATE TABLE ayrıcalığına ve içinde nesnelerin oluşturulacağı bir depolama alanına sahip olması gerekir. Veritabanı yöneticisi (DBA), kullanıcılara ayrıcalıklar vermek için veri kontrol dili (DCL) ifadelerini kullanır.

Sözdiziminde:

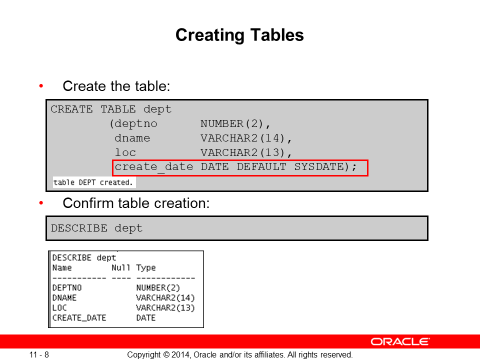
*schema* Is the same as the owner’s name

*table* Is the name of the table

DEFAULT *expr* Specifies a default value if a value is omitted in the INSERT statement

*column* Is the name of the column

*datatype* Is the column’s data type and length



Slayttaki örnek, dört sütunlu DEPT tablosunu oluşturur: DEPTNO, DNAME, LOC ve CREATE\_DATE. CREATE\_DATE sütununun varsayılan bir değeri vardır. INSERT ifadesi için bir değer sağlanmazsa, sistem tarihi otomatik olarak eklenir.

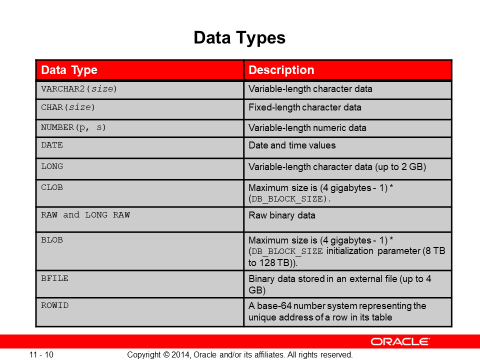
Tablonun oluşturulduğunu onaylamak için DESCRIBE komutunu çalıştırın.

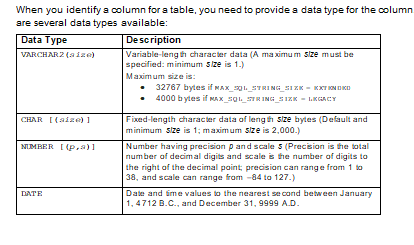
Tablo oluşturmak bir DDL deyimi olduğundan, bu deyim yürütüldüğünde otomatik bir kesinleştirme gerçekleşir.

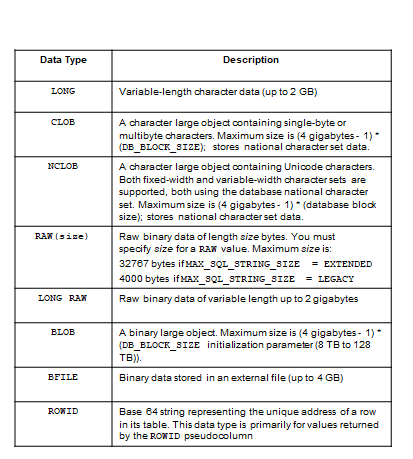
Not: Sahip olduğunuz tabloların listesini veri sözlüğünü sorgulayarak görüntüleyebilirsiniz. Örnek:

select table\_name from user\_tables;

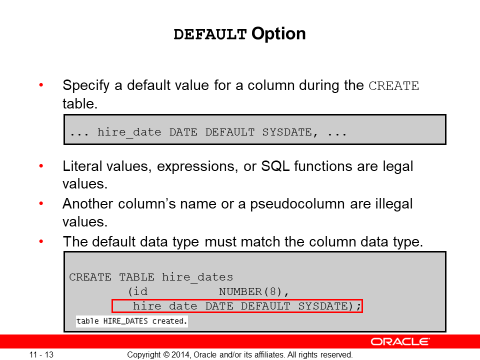
Veri sözlüğü görünümlerini kullanarak görünümler, dizinler vb. gibi diğer veritabanı nesneleri hakkında da bilgi bulabilirsiniz. Oracle Database: SQL Workshop II kursunda veri sözlükleri hakkında detaylı bilgi edineceksiniz.







Yönergeler Bir alt sorgu kullanılarak bir tablo oluşturulduğunda LONG sütunu kopyalanmaz. LONG sütunu, GROUP BY veya ORDER BY yan tümcesine dahil edilemez. Tablo başına yalnızca bir LONG sütun kullanılabilir. LONG sütununda herhangi bir kısıtlama tanımlanamaz. LONG sütun yerine CLOB sütunu kullanmak isteyebilirsiniz.



Bir tablo tanımladığınızda, DEFAULT seçeneğini kullanarak bir sütuna varsayılan bir değer verilmesi gerektiğini belirtebilirsiniz. Bu seçenek, sütun için bir değer olmadan bir satır eklendiğinde boş değerlerin sütunlara girmesini engeller. Varsayılan değer bir hazır bilgi, bir ifade veya bir SQL işlevi (SYSDATE veya USER gibi) olabilir, ancak değer başka bir sütunun veya bir sözde sütunun adı (NEXTVAL veya CURRVAL gibi) olamaz. Varsayılan ifade, sütunun veri türüyle eşleşmelidir.

Aşağıdaki örnekleri göz önünde bulundurun:

INSERT INTO hire\_dates values(45, NULL);

Yukarıdaki ifadede, varsayılan değer yerine null değeri ekleyecektir.

INSERT INTO hire\_dates(id) values(35);

Yukarıdaki ifadede hire datei varsayılan değer ne ise onu eklicek yani system tarihini

**CONSTRAINTS**

Constraınts, tablo düzeyinde kuralları zorunlu kılar.

Kısıtlamalar, veritabanının tutarlılığını ve bütünlüğünü sağlar.

Aşağıdaki kısıtlama türleri geçerlidir:

* + - NOT NULL
    - UNIQUE
    - PRIMARY KEY
    - FOREIGN KEY
    - CHECK

Oracle sunucusu, tablolara geçersiz veri girişini önlemek için kısıtlamalar kullanır.

Aşağıdakileri yapmak için kısıtlamaları kullanabilirsiniz:

Bir tablodaki her satır eklendiğinde, güncellendiğinde veya tablodan silindiğinde, tablodaki veriler üzerinde kuralları uygulayın. İşlemin başarılı olması için kısıtlamanın karşılanması gerekir.

Diğer tablolardan bağımlılıklar varsa, bir tablonun düşmesini önleyin.

**Constraint Guidelines**

Bir kısıtlamayı adlandırabilirsiniz veya Oracle sunucusu SYS\_Cn biçimini kullanarak bir ad oluşturur.

Aşağıdaki zamanlardan birinde bir kısıtlama oluşturun:

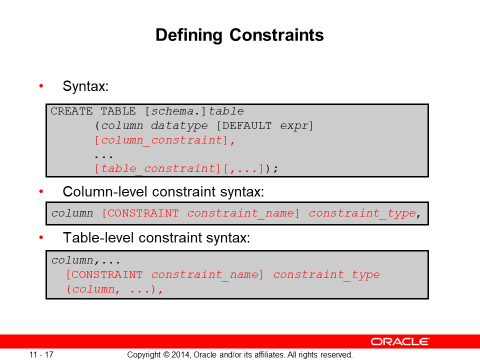
Tablonun oluşturulması ile aynı zamanda

Tablonun oluşturulmasından sonra

Sütun veya tablo düzeyinde bir kısıtlama tanımlayın.

Veri sözlüğünde bir kısıtlama görüntüleyin.

Tüm kısıtlamalar veri sözlüğünde saklanır. Eğer onlara bir a verirseniz, kısıtlamaların yeniden anlamlandırılması kolaydır. Kısıtlama adları, adın aynı kullanıcıya ait başka bir nesneyle aynı olmaması dışında, standart nesne adlandırma kurallarına uymalıdır. Kısıtlamanızı adlandırmazsanız, Oracle sunucusu SYS\_Cn biçiminde bir ad oluşturur, burada n bir tam sayıdır, böylece kısıtlama adı benzersiz olur.

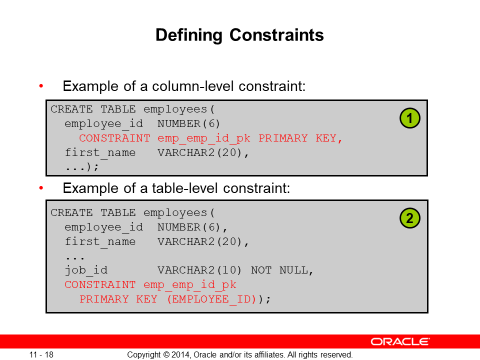


Kısıtlamalar, tablo oluşturulurken veya oluşturulduktan sonra tanımlanabilir. Sütun veya tablo düzeyinde bir kısıtlama tanımlayabilirsiniz. İşlevsel olarak, tablo düzeyinde bir kısıtlama, sütun düzeyinde bir kısıtlama ile aynıdır.

Slayt, bir tablo oluştururken kısıtlamaları tanımlamaya yönelik sözdizimini verir. Sütun düzeyinde veya tablo düzeyinde kısıtlamalar oluşturabilirsiniz. Sütun düzeyinde tanımlanan kısıtlamalar, sütun tanımlandığında dahil edilir. Tablo düzeyindeki kısıtlamalar, tablo tanımının sonunda tanımlanır ve kısıtlamanın bir dizi parantez içinde ait olduğu sütun veya sütunlara atıfta bulunmalıdır. İkisini birbirinden ayıran esas olarak sözdizimidir; aksi halde, işlevsel olarak, sütun düzeyinde bir kısıtlama, tablo düzeyinde bir kısıtlama ile aynıdır.

NOT NULL kısıtlamaları yalnızca sütun düzeyinde tanımlanabilir.

Birden fazla sütuna uygulanan kısıtlamalar tablo düzeyinde tanımlanmalıdır.

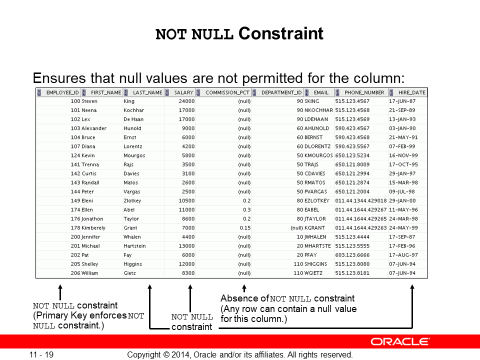


Kısıtlamalar genellikle tabloyla aynı anda oluşturulur. Kısıtlamalar, oluşturulduktan sonra bir tabloya eklenebilir ve ayrıca geçici olarak devre dışı bırakılabilir.

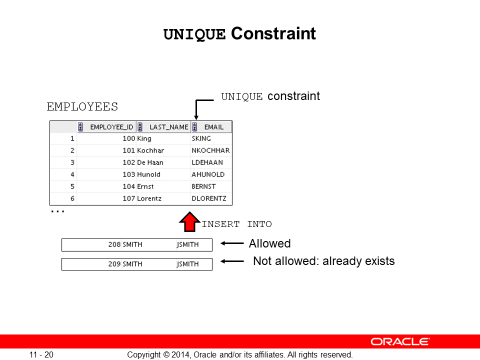
Slayttaki her iki örnek de EMPLOYEES tablosunun EMPLOYEE\_ID sütununda bir primary key kısıtlaması (CONSTRAINT) oluşturur.

1. İlk örnek, kısıtlamayı tanımlamak için sütun düzeyinde sözdizimini kullanır.

2. İkinci örnek, kısıtlamayı tanımlamak için tablo düzeyinde sözdizimini kullanır.



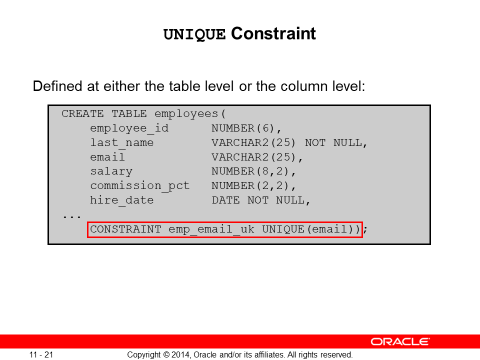
NOT NULL kısıtlaması, sütunun boş değer içermemesini sağlar. NOT NULL kısıtlaması olmayan sütunlar, varsayılan olarak boş değerler içerebilir. NOT NULL kısıtlamaları sütun düzeyinde tanımlanmalıdır. ÇALIŞANLAR tablosunda, EMPLOYEE\_ID sütunu, birincil anahtar olarak tanımlandığından NULL DEĞİL kısıtlamasını devralır (primary key odluğu için not null ve unique oluyo otomatikman). Aksi takdirde, LAST\_NAME, EMAIL, HIRE\_DATE ve JOB\_ID sütunları üzerinde NOT NULL kısıtlaması uygulanır.



BENZERSİZ anahtar bütünlüğü kısıtlaması, bir sütundaki veya bir sütun kümesindeki (anahtar) her değerin benzersiz olmasını gerektirir; yani, bir tablonun hiçbir iki satırı, belirtilen bir sütunda veya bir sütun kümesinde yinelenen değerlere sahip olamaz. BENZERSİZ anahtar kısıtlamasının tanımına dahil edilen sütuna (veya sütun kümesine) benzersiz anahtar denir. UNIQUE kısıtlaması birden fazla sütun içeriyorsa, bu sütun grubuna bileşik benzersiz anahtar denir.

BENZERSİZ(unıque) kısıtlamalar, aynı sütunlar için NOT NULL kısıtlamaları da tanımlamadığınız sürece boş değerlerin girişini sağlar. Aslında, boş değerler hiçbir şeye eşit olarak kabul edilmediğinden, herhangi bir sayıda satır, NOT NULL kısıtlaması olmayan sütunlar için boş değerler içerebilir. Bir sütundaki (veya bileşik bir UNIQUE anahtarının tüm sütunlarındaki) boş değer, her zaman UNIQUE kısıtlamasını karşılar.

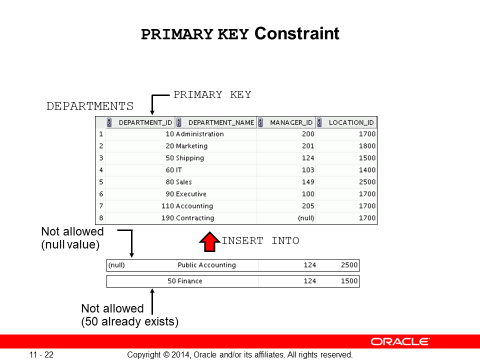
Not: Birden fazla sütundaki UNIQUE kısıtlamaları için arama mekanizması nedeniyle, kısmen boş bir bileşik UNIQUE anahtar kısıtlamasının boş olmayan sütunlarında aynı değerlere sahip olamazsınız.



UNIQUE CONSTRAINT, sütun düzeyinde veya tablo düzeyinde tanımlanabilir. Bileşik benzersiz bir anahtar oluşturmak istediğinizde kısıtlamayı tablo düzeyinde tanımlarsınız. Bir satırı UNIQUE şekilde tanımlayabilen tek bir öznitelik olmadığında bileşik anahtar tanımlanır. Bu durumda, birleşik değeri her zaman benzersiz olan ve satırları tanımlayabilen iki veya daha fazla sütundan oluşan benzersiz bir anahtarınız olabilir.

Slayttaki örnek, ÇALIŞANLAR tablosunun EMAIL sütununa UNIQUE CONSTRAINT uygular. Kısıtlamanın adı EMP\_EMAIL\_UK.

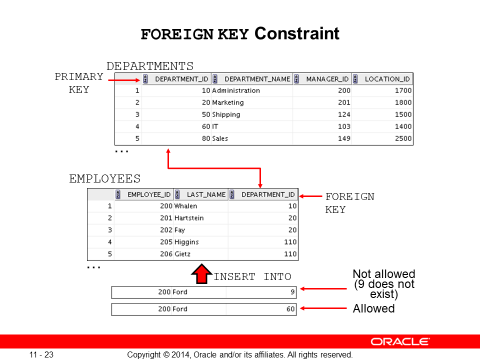
Not: Oracle sunucusu, benzersiz anahtar sütun veya sütunlarında örtük olarak benzersiz bir dizin oluşturarak BENZERSİZ kısıtlamayı uygular.



Bir PRIMARY KEY kısıtlaması, tablo için bir birincil anahtar(prımary key) oluşturur. Her tablo için yalnızca bir birincil anahtar oluşturulabilir. PRIMARY KEY kısıtlaması, bir tablodaki her satırı benzersiz şekilde tanımlayan bir sütun veya sütunlar kümesidir. Bu kısıtlama, sütun veya sütun kombinasyonunun benzersizliğini zorlar ve birincil anahtarın parçası olan hiçbir sütunun boş değer içermemesini sağlar.

Not: Benzersizlik, birincil anahtar kısıtlama tanımının bir parçası olduğundan, Oracle sunucusu, birincil anahtar sütunu veya sütunlarında örtük olarak benzersiz bir dizin oluşturarak benzersizliği zorlar.7

Yukarıdaki tabloya employee id si null olan bir değer girdiğimde izin ver mi yo aynı şekilde daha önce girilmşi bir id yi(50) eklemek istediğimde ona da izin ver miyo çünkü primary key olan sütün not null ve unıque olması lazım.



Employees tablosundaki department\_id foreign key ben employees bir kayıt ekledğimde departmant id si 9 diye dapartments tablosuna bakıyo department id si 9 olan kayıt var mı yok o yüzden izin vermiyor 60 a bakıyo var izin veriyo. YABANCI ANAHTAR (veya başvuru bütünlüğü) kısıtlaması, bir sütunu veya bir sütun kombinasyonunu yabancı anahtar olarak belirler ve aynı tablodaki veya farklı bir tablodaki birincil anahtar veya benzersiz bir anahtarla ilişki kurar.

Slayttaki örnekte DEPARTMENT\_ID, EMPLOYEES tablosunda (bağımlı veya alt tablo) yabancı anahtar olarak tanımlanmıştır; DEPARTMENTS tablosunun (başvurulan veya ana tablo) DEPARTMENT\_ID sütununa başvurur.

Yönergeler

Bir yabancı anahtar değeri, üst tablodaki mevcut bir değerle eşleşmeli veya NULL olmalıdır.

Yabancı anahtarlar veri değerlerine dayanır ve fiziksel işaretçiler yerine tamamen mantıksaldır.

YABANCI ANAHTAR kısıtlamaları, sütun veya tablo kısıtlaması düzeyinde tanımlanabilir. Tablo düzeyinde tanım kullanılarak bir bileşik yabancı anahtar oluşturulmalıdır.

Slayttaki örnek, ÇALIŞANLAR tablosunun DEPARTMENT\_ID sütununda, tablo düzeyinde sözdizimi kullanarak bir YABANCI ANAHTAR kısıtlaması tanımlar. Kısıtlamanın adı EMP\_DEPT\_FK'dir.

Yabancı anahtar, alt tabloda tanımlanır ve başvurulan sütunu içeren tablo, üst tablodur. Yabancı anahtar, aşağıdaki anahtar kelimelerin bir kombinasyonu kullanılarak tanımlanır:

YABANCI ANAHTAR tablo-kısıtlama düzeyinde alt tablodaki sütunu tanımlamak için kullanılır.

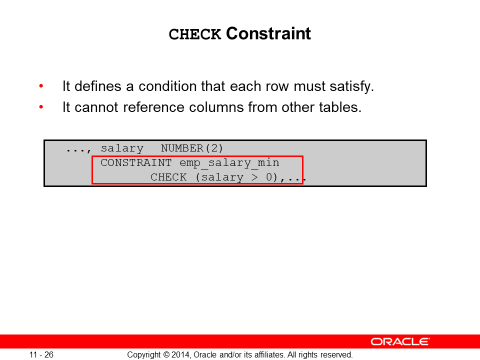
REFERENCES, üst tablodaki tabloyu ve sütunu tanımlar.

ON DELETE CASCADE, üst tablodaki bir satır silindiğinde, alt tablodaki bağımlı satırların da silindiğini belirtir.

ON DELETE SET NULL, üst tablodaki bir satır silindiğinde, yabancı anahtar değerlerinin null olarak ayarlandığını gösterir.

Varsayılan davranış, başvurulan verilerin güncellenmesine veya silinmesine izin vermeyen kısıtlama kuralı olarak adlandırılır.

ON DELETE CASCADE veya ON DELETE SET NULL seçenekleri olmadan, alt tabloda referans veriliyorsa üst tablodaki satır silinemez. Ve bu anahtar sözcük, sütun düzeyinde sözdiziminde kullanılamaz.

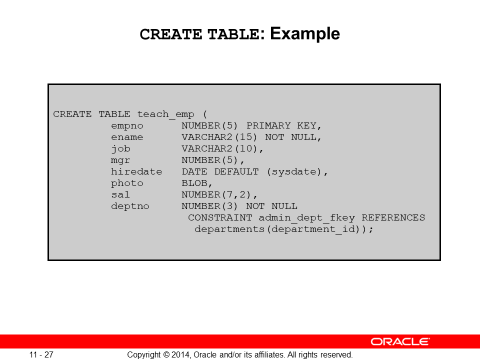


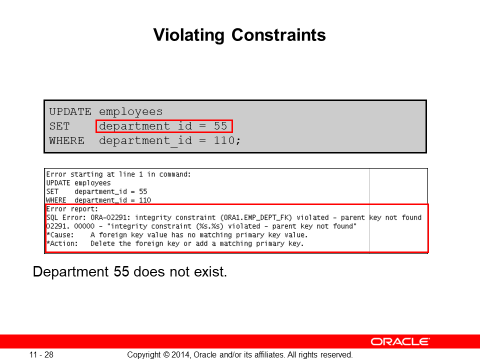
CHECK kısıtlaması, her satırın karşılaması gereken bir koşulu tanımlar. Kısıtlamayı karşılamak için tablodaki her satır, koşulu DOĞRU ya da bilinmiyor (boş değer nedeniyle) yapmalıdır.

Koşul, diğer satırlardaki diğer değerlere başvuran sorgular dışında, sorgu koşullarıyla aynı yapıları kullanabilir.

Tek bir sütun, tanımındaki sütuna başvuran birden çok CHECK kısıtlamasına sahip olabilir. Bir sütunda tanımlayabileceğiniz CHECK kısıtlamalarının sayısında bir sınırlama yoktur.

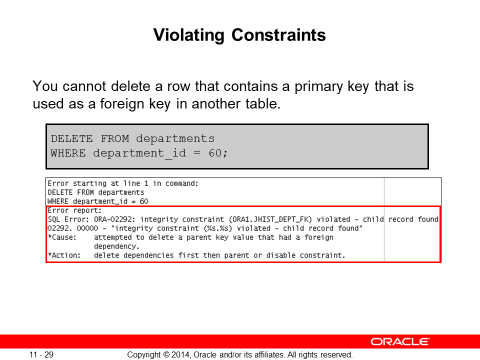
KONTROL kısıtlamaları sütun düzeyinde veya tablo düzeyinde tanımlanabilir.burda sütün düzeyinde tanımlanmış





Sütunlarda kısıtlamalarınız olduğunda, kısıtlama kuralını ihlal etmeye çalışırsanız bir hata döndürülür. Örneğin, bir bütünlük kısıtlamasına bağlı bir değere sahip bir kaydı güncellemeye çalışırsanız, bir hata döndürülür.

Slayttaki örnekte, 55 departman ana tablo, department tablosunda mevcut değildir ve bu nedenle ORA-02291 "üst anahtar bulunamadı" ihlali alırsınız.55 nolu department\_id department tablosunda department \_id kolonunda yok.



Departments tablosundaki department\_id employees tablosunda foreign key olarak olarak kullanıyor bu yüzden silemem.

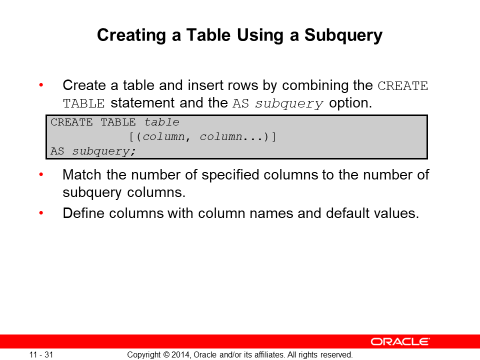
Bütünlük kısıtlamasına bağlı bir değere sahip bir kaydı silmeye çalışırsanız, bir hata döndürülür.

Slayttaki örnek, departman 60'ı BÖLÜMLER tablosundan silmeye çalışır, ancak bu departman numarası ÇALIŞANLAR tablosunda yabancı anahtar olarak kullanıldığından bir hatayla sonuçlanır. Silmeye çalıştığınız üst kayıtta alt kayıtlar varsa, ORA-02292 "alt kayıt bulundu" ihlali alırsınız.

70 numaralı departmanda çalışan olmadığı için aşağıdaki ifade işe yarar:

DELETE FROM departments

WHERE department\_id = 70;



Tablo oluşturmak için ikinci bir yöntem, hem tabloyu oluşturan hem de alt sorgudan döndürülen satırları ekleyen AS alt sorgu yan tümcesini uygulamaktır.

Sözdiziminde:

table tablonun adıdır

column Sütunun adı, varsayılan değer ve bütünlük kısıtlamasıdır

subquery Yeni tabloya eklenecek satır kümesini tanımlayan SELECT ifadesidir.

Yönergeler

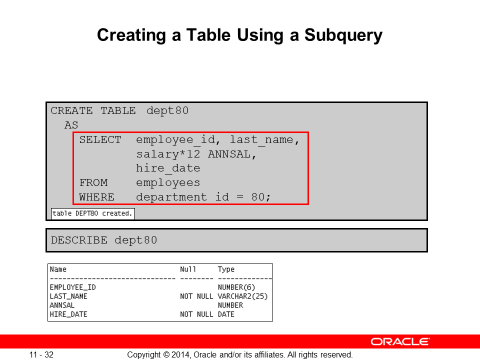
Tablo, belirtilen sütun adlarıyla oluşturulur ve SELECT deyimi tarafından alınan satırlar tabloya eklenir.

Sütun tanımı yalnızca sütun adını ve varsayılan değeri içerebilir.

Sütun özellikleri verilmişse, sütun sayısı, alt sorgu SELECT listesindeki sütun sayısına eşit olmalıdır.

Hiçbir sütun belirtimi verilmezse, tablonun sütun adları alt sorgudaki sütun adlarıyla aynıdır.

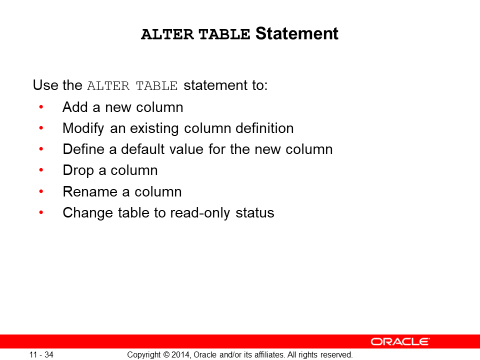
Sütun veri türü tanımları ve NOT NULL kısıtlaması yeni tabloya geçirilir. Yalnızca açık NOT NULL kısıtlamasının devralınacağını unutmayın. PRIMARY KEY sütunu, NOT NULL özelliğini yeni sütuna geçirmeyecektir. Diğer kısıtlama kuralları yeni tabloya geçirilmez. Ancak, sütun tanımına kısıtlamalar ekleyebilirsiniz



Slayttaki örnek, departman 80'de çalışan tüm çalışanların ayrıntılarını içeren DEPT80 adında bir tablo oluşturur. DEPT80 tablosunun verilerinin ÇALIŞANLAR tablosundan geldiğine dikkat edin.

DESCRIBE komutunu kullanarak bir veritabanı tablosunun varlığını doğrulayabilir ve sütun tanımlarını kontrol edebilirsiniz.

Ancak, bir ifade seçerken bir sütun diğer adı sağladığınızdan emin olun. SALARY\*12 ifadesine ANNSAL takma adı verilir. Takma ad olmadan aşağıdaki hata oluşturulur:



Bir tablo oluşturduktan sonra, aşağıdaki nedenlerden herhangi biri nedeniyle tablo yapısını değiştirmeniz gerekebilir:

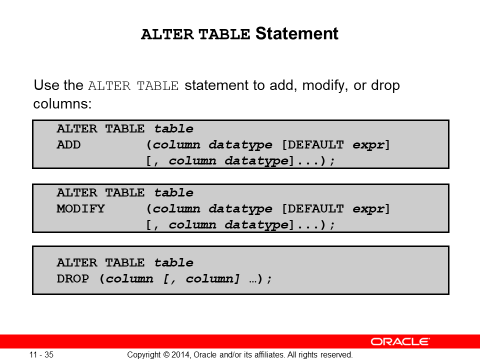
Bir sütunu atladınız.

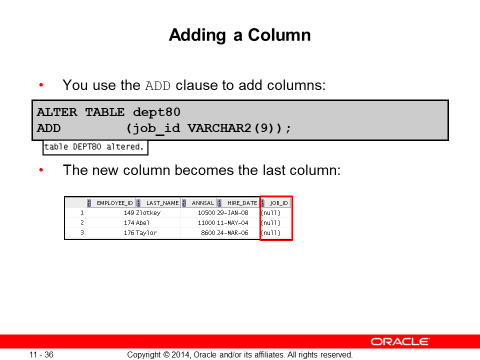
Sütun tanımınız veya adının değiştirilmesi gerekiyor.

Sütunları kaldırmanız gerekiyor.

Tabloyu salt okunur moda geçirmek istiyorsunuz

ALTER TABLE ifadesini kullanarak bunu yapabilirsiniz.





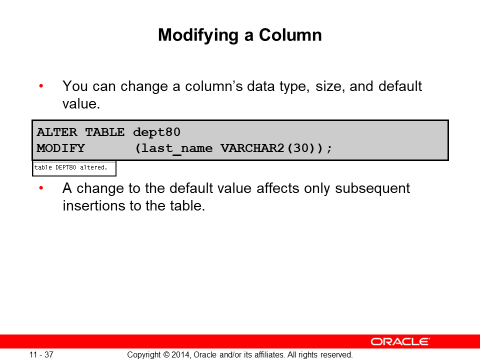
Sütun Ekleme Yönergeleri

Sütun ekleyebilir veya değiştirebilirsiniz.

Sütunun nerede görüneceğini belirtemezsiniz. Yeni sütun son sütun olur.

Slayttaki örnek, DEPT80 tablosuna JOB\_ID adlı bir sütun ekler. JOB\_ID sütunu, tablodaki son sütun olur.

Not: Bir sütun eklendiğinde bir tablo zaten satır içeriyorsa, yeni sütun başlangıçta boştur veya tüm satırlar için varsayılan değeri alır. Yalnızca varsayılan bir değer belirtirseniz, diğer sütunlarda veri içeren bir tabloya zorunlu bir NOT NULL sütunu ekleyebilirsiniz. Varsayılan değer olmadan boş bir tabloya NOT NULL sütunu ekleyebilirsiniz.



MODIFY yan tümcesi ile ALTER TABLE ifadesini kullanarak bir sütun tanımını değiştirebilirsiniz. Sütun değişikliği, bir sütunun veri türü, boyutu ve varsayılan değerindeki değişiklikleri içerebilir.

Yönergeler

Sayısal bir sütunun genişliğini veya kesinliğini artırabilirsiniz.

Karakter sütunlarının genişliğini artırabilirsiniz.

Aşağıdaki durumlarda bir sütunun genişliğini azaltabilirsiniz:

Sütun yalnızca boş değerler içeriyor

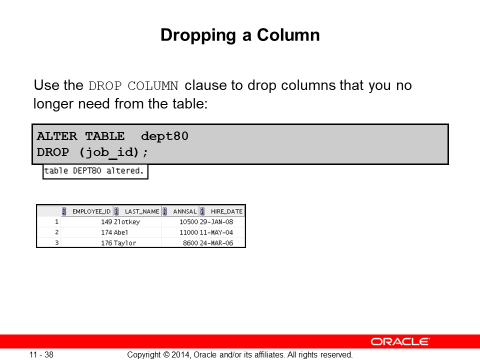
Tabloda satır yok

Sütun genişliğindeki azalma, o sütundaki mevcut değerlerden daha az değil

Sütun yalnızca boş değerler içeriyorsa veri türünü değiştirebilirsiniz. Bunun istisnası, sütunlardaki verilerle yapılabilecek CHAR-VARCHAR2 dönüşümleridir.

Yalnızca sütun boş değerler içeriyorsa veya boyutu değiştirmezseniz, bir CHAR sütununu VARCHAR2 veri türüne dönüştürebilir veya bir VARCHAR2 sütununu CHAR veri türüne dönüştürebilirsiniz.

Bir sütunun varsayılan değerinde yapılan bir değişiklik, yalnızca tabloya sonraki eklemeleri etkiler.



DROP COLUMN yan tümcesi ile ALTER TABLE ifadesini kullanarak bir tablodan bir sütun kaldırabilirsinzi.

Yönergeler

Sütun veri içerebilir veya içermeyebilir.

ALTER TABLE DROP COLUMN deyimi kullanılarak, bir seferde yalnızca bir sütun kaldıralabilir.

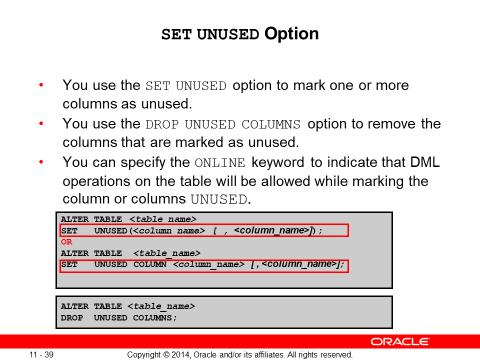
Tablo, değiştirildikten sonra içinde en az bir sütun kalmalıdır.

Bir sütun bırakıldıktan sonra kurtarılamaz.

Basamaklı seçenek eklenmedikçe, başka bir sütun tarafından başvurulan birincil anahtar bırakılamaz.

Sütunda çok sayıda değer varsa, bir sütunu bırakmak biraz zaman alabilir. Bu durumda, uzun süreli kilitlerden kaçınmak için kullanılmayacak şekilde ayarlamak ve sistemde daha az kullanıcı olduğunda bırakmak daha iyi olabilir.

Not: Bölümlenmiş bir tablonun bölümleme anahtarının bir parçasını oluşturan sütunlar veya dizinle düzenlenmiş bir tablonun PRIMARY KEY'in bir parçasını oluşturan sütunlar gibi belirli sütunlar asla bırakılamaz. İndeksle organize edilmiş tablolar ve bölümlenmiş tablolar hakkında daha fazla bilgi için Oracle Database Concepts ve Oracle Database Administrator's Guide'a bakın.



KULLANILMAMIŞ AYARLA (SET UNUSED) seçeneği, bir veya daha fazla sütunu kullanılmamış olarak işaretler, böylece sistem kaynaklarına olan talep daha düşük olduğunda kaldırılabilirler. Bu yan tümcenin belirtilmesi, tablodaki her satırdan hedef sütunları gerçekten kaldırmaz (yani, bu sütunlar tarafından kullanılan disk alanını geri yüklemez). Bu nedenle, yanıt süresi, DROP yan tümcesini çalıştırdığınızdan daha hızlıdır. Kullanılmayan sütunlar, sütun verileri tablonun satırlarında kalsa bile, atılmış gibi değerlendirilir. Bir sütun kullanılmamış olarak işaretlendikten sonra o sütuna erişiminiz olmaz. SELECT \* sorgusu, kullanılmayan sütunlardan veri almaz. Ayrıca, DESCRIBE deyimi sırasında kullanılmayan olarak işaretlenen sütunların adları ve türleri görüntülenmez ve tabloya kullanılmayan bir sütunla aynı ada sahip yeni bir sütun ekleyebilirsiniz. SET UNUSED bilgisi USER\_UNUSED\_COL\_TABS sözlük görünümünde saklanır.

ONLINE anahtar sözcüğünü, UNUSED sütun veya sütunları işaretlerken tablodaki DML işlemlerine izin verileceğini belirtmek için belirtebilirsiniz. Kod örneği, ONLINE anahtar sözcüğünü kullanarak sonsuza kadar kullanılmayacak bir sütun ayarlayan SET UNUSED COLUMN'un kullanımını gösterir.

ALTER TABLE dept80 SET UNUSED(hire\_date)ONLINE;

Not: Bir sütunu UNUSED olarak ayarlamak için yönergeler, bir sütunu kaldırmak için için olanlara benzer.

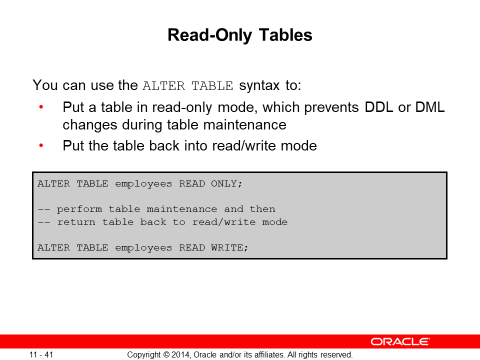
DROP UNUSUED COLUMNS Seçeneği

DROP UNUSED COLUMNS, şu anda kullanılmamış olarak işaretlenmiş tüm sütunları tablodan kaldırır. Tablodaki kullanılmayan sütunlardan fazladan disk alanını geri almak istediğinizde bu ifadeyi kullanabilirsiniz. Tablo kullanılmayan sütun içermiyorsa, ifade hatasız olarak döner

ALTER TABLE dept80

SET UNUSED (last\_name)

ALTER TABLE dept80 DROP UNUSED COLUMNS;

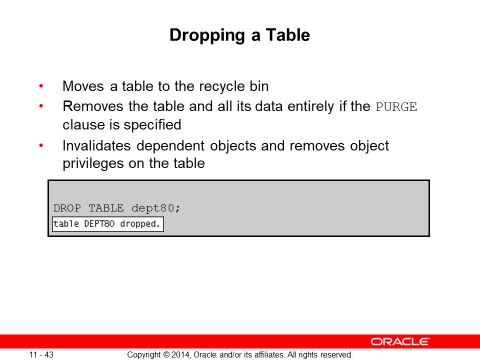


Bir tabloyu salt okunur modda yerleştirmek için SADECE OKUYUN(READY ONLY)'u belirtebilirsiniz. Tablo SADECE OKUMA modundayken, tabloyu etkileyen herhangi bir DML deyimi veya herhangi bir SELECT ... FOR UPDATE deyimi yayınlayamazsınız. Tablodaki herhangi bir veriyi değiştirmedikleri sürece DDL deyimleri yayınlayabilirsiniz. Tablo SADECE OKUMA modundayken tabloyla ilişkili dizinler üzerinde işlemlere izin verilir.

Salt okunur bir tabloyu okuma/yazma moduna döndürmek için READ/WRITE belirtin.

Not: SADECE OKUMA modunda olan bir tabloyu bırakabilirsiniz. DROP komutu yalnızca veri sözlüğünde yürütülür, bu nedenle tablo içeriğine erişim gerekli değildir. Tablo alanı yeniden okuma/yazma yapılıncaya kadar tablo tarafından kullanılan alan geri alınmayacaktır ve ardından blok segment başlıklarında gerekli değişiklikler yapılabilir ve bu böyle devam eder.

ALTER TABLE deyimi hakkında bilgi için Oracle Database 12c: SQL Workshop II başlıklı kursa bakın.



DROP TABLE ifadesi, bir tabloyu geri dönüşüm kutusuna taşır veya tabloyu ve tüm verilerini veritabanından tamamen kaldırır. PURGE yan tümcesini belirtmediğiniz sürece, DROP TABLE deyimi, diğer nesneler tarafından kullanılmak üzere tablo alanına geri bırakılan alanla sonuçlanmaz ve alan, kullanıcının alan kotasında sayılmaya devam eder. Bir tabloyu bırakmak, bağımlı nesneleri geçersiz kılar ve tablodaki nesne ayrıcalıklarını kaldırır.

Bir tabloyu bıraktığınızda(kaldırdığınızda,drop ettiğinizde), veritabanı tablodaki tüm verileri ve onunla ilişkili tüm indexleri kaybeder.

QUİZ

To do which three of the following can you use constraints?

* 1. Enforce rules on the data in a table whenever a row is inserted, updated, or deleted.
  2. Prevent the dropping of a table.
  3. Prevent the creation of a table.
  4. Prevent the creation of data in a table.

A b ve d cevap

Aşağıdakilerden hangi üçünü yapmak için kısıtlamaları kullanabilirsiniz?

a. Bir satır eklendiğinde, güncellendiğinde veya silindiğinde tablodaki veriler üzerinde kuralları uygulayın.

B. Bir tablonun kaldırılmasını engelleyin.

C. Bir tablonun oluşturulmasını engelleyin.

d. Bir tabloda veri oluşturulmasını engelleyin.

s