SQL لغة الاستعلام البنيوية Structural query language

Relational Data Base قواعد البيانات العلائقية أوراكل

يمكن تقسيم عبارات لغة SQL بالشكل التالي :

Statement	Description وصف
Select	استرجاع البيانات من قاعدة البيانات
Insert	DML _ Data manipulation Language وتعرف بلغة معالجة البيانات تستخدم لإدخال صفوف جديدة أو تغيير في صفوف موجودة وحذف صفوف غير ضرورية من جدول في قاعدة البيانات
Update	
Delete	
Create	DDL _ Data definition language وتعرف بلغة تعريف البيانات (التعامل مع الكائنات) وتستخدم لتثبيت و تغيير تركيبات أو أبنية البيانات في الجدول بشكل عام
Alter	
Drop	
Rename	
Truncate	
Grant	DCL _ Data control language تعرف بلغة التحكم بالبيانات وعن طريقها يتم تحديد السماحيات بالدخول على قاعدة البيانات أو التغيير في بنيتها
Revoke	

عبارة الاختيار Select الأساسية (Basic select statement) :

الصيغة العامة :

Select {*, Column name , ----}
From Table name:

Examples : **أمثلة**

١) اختيار كافة الأعمدة
Select all column

```
1) 1. SQL> select *
   2.        from emp ;
```

٢) اختيار بعض الأعمدة بأسمائها
Selecting specific column

```
2) 1.SQL> Select deptno ,dname ,loc
   2.        from emp ;
```

٣) استخدام التعابير الحسابية

Arithmetic Operators

```
3) 1. SQL> select ename,sal,sal+300
   2.        from emp;
```

جدول الرموز الحسابية :

Operator الرمز	Description وصف
+	**Add**
-	**Subtract**
*	**Multiply**
/	**Divide**

أسبقية العمليات الحسابية (الضرب - القسمة – الجمع الطرح)

Using Parentheses (استخدام الأقواس): يمكن استخدام الأقواس لتحديد أسبقية العملية الحسابية

مثلاً : 12*(sal+100)=1260 تختلف عن 12*sal+100=160 وذلك باعتبار sal =5

القيمة NULL أي عدم الوجود أو باطلة : وهي قيمة غير موجودة وغير محددة Unassigned وغير معروفة Unknown وغير قابلة للاستعمال و التطبيق Inapplicable وهي لا تشبه الصفر أو الفراغ وأي عملية حسابية على القيمة NULL تنتج القيمة NULL نفسها .

٤) استخدام أسماء مستعارة للأعمدة Using column aliases

- إما أن نستعمل AS أو نكتب الاسم القديم والاسم الجديد مباشرة بعده و بينهما فراغ :

```
4) 1.SQL>select ename As name ,sal  salary
   2.        from emp;
```

-أو نضع الاسم الجديد ضمن فواصل علوية و خاصة في اللغة العربية كما يلي :

```
   1.SQL>select ename  "name" , sal*12  "annual salary"
   2.        from emp;
```

٥) استخدام السلاسل الحرفية البسيطة Using Literal Character String

يمكننا دمج السلاسل الحرفية لحقلين أو أكثر ووضع فراغات بينها باستخدام الرمز || كما يلي :

```
5) 1.SQL>select ename ||' '|| job  as  "Employee Details"
   2.        from emp;
```

وفي هذه الحالة يجب استخدام AS حتماً

كما يمكننا دمج الصفوف بدون فراغات

```
   1.SQL>select ename || job  as  "Employees"
   2.        from emp;
```

٦)الصفوف المزدوجة (المكررة) Duplicate Rows

عند عرض الجدول يقوم البرنامج بعرض كل الصفوف والصفوف المكررة يمكن استخدام عبارة Distinct لإزالة الصفوف المكررة

إزالة الصفوف المكررة

```
6) 1.SQL>select  Distinct deptno
   2.        from emp;
```

وصف بنية الجدول Displaying Table Structure :

يمكننا وصف بنية الجدول باستخدام العبارة DESCRIBE مع اسم الجدول ويمكن اختصار العبارة بكتابة DESC والذي يعطينا وصف للجدول ولأنواع البيانات في الحقول والقيم الباطلة أو الفارغة كما يلي :

```
SQL> Desc dept
```

بعض أنواع البيانات : **SOME DATA TYPE**

NUMBER (P,S) : النوع العددي حيث P أكبر طول رقم قبل الفاصلة العشرية و S أكبرطول رقم بعد الفاصلة العشرية .

VARCAR(S) : نوع بيانات حرفي أو رمزي أكبر عدد لحروفه S .

DATE : بيانات من النوع تاريخ ويأخذ أكبر قيمة له 31/ December /9999 .

CHAR(S) : بيانات نوع حرفي تأخذ طول ثابت S .

أوامر التحرير في لغة الـ SQL PLUS — SQL*PLUS EDITING COMMANDS

I. الأمر APPEND وبعده نص (يمكن اختصاره بحرف A) : يقوم هذا الأمر بإضافة النص إلى آخر السطر الحالي أو السطر الأخير في عبارة SQL السابقة بمعنى آخر في حالة وضع عبارة ناقصة وتنفيذها يمكن إكمالها بعد التنفيذ باستخدام الأمر A .

II. الأمر CHANGE (اختصاراً C) : كما يلي **C /old text/new text/** يقوم بتغيير النص OLD إلى النص NEW في عبارة SQLالأخيرة .

III. الأمر CHNGE /text/ أو C /text/ : يحذف النص text نهائياً من عبارة SQL الأخيرة أي يستبدلها بفراغ .

IV. الأمر CLEAR BUFFER أو اختصارا CL BUFF : يحذف كل السطور أو التعليمات الموجودة ضمن الحاجز أو المصد (الـ Buffer).

V. الأمر DEL: يقوم بحذف السطر الحالي أي السطر الأخير في عبارة SQL الأخيرة .

VI. الأمر RUN أو R: يقوم هذا الأمر بعرض وتشغيل عبارة SQL الأخيرة والموجودة ضمن المصد الـ Buffer .

أوامر الملفات في لغة الـ SQL PLUS — SQL*PLUS FILE COMMANDS

i. الأمر SAVE Filename أو SAV Filename : يقوم بحفظ تعليمة الـ SQL الأخيرة ضمن الملف المسمى ؟ والذي يمكن استدعاؤه وتنفيذ التعليمة ضمنه باستخدام الأمر التالي GET .

ii. الأمر GET Filename : يقوم باستدعاء تعليمة الـ SQL المخزنة ضمن الملف النصي المذكور .

iii. الأمر START Filename أو STA Filename : يقوم مباشرة بتنفيذ تعليمة الـ SQL المخزنة في الملف المذكور ويمكن الاستعاضة عنه بالأمر **@filename** .

iv. الأمر EDIT أو اختصارا ED : يقوم بتحرير آخر تعليمة SQL ضمن محرر نصوص المفكرة عندها يمكن إصلاحها ثم حفظها وتنفيذها من جديد باستخدام الأمر R .

v. الأمر EDIT Filename أو ED Filename : يقوم بتحرير الملف الذي قمنا بحفظ تعليمة SQL ضمنه مسبقاً ضمن محرر نصوص .

vi.الأمر SPOOL Filename أو اختصاراً SPO Filename : يقوم هذا الأمر بتخزين نتائج الاستفسار في ملف ، بإضافة **OFF** إلى الأمر فإنه يقوم بإغلاق الملف ، أما بإضافة **OUT** فإنه يقوم بإغلاق الملف وإرسال النتائج إلى طابعة النظام . SPO filename|OFF أو SPO filename|out .

vii. الأمر EXIT يقوم بالخروج من واجهة الـ **SQL*PLUS .**

عبارة WHERE (Were Clause) :

LIMITING ROWS SELECTED — تحديد الصفوف المختارة

Restrict the rows returned by using the where clause _ Where تحديد الصفوف التي تظهر باستخدام عبارة

الصيغة العامة

```
SQL>select Distinct -------
        From  table name
        Where  conditions ;
```

مثال :

```
SQL>select ename , job , deptno
        from  emp
        Where  job= 'clerk' ;
```

COMPARISON OPERATORS — رموز (عوامل) المقارنة

Example :

```
SQL>Select ename , sal , comm
From  emp
Where  sal<=comm ;
```

Operator	Meaning المعنى
=	Equal to يساوي
>	Greater than أكبر من
>=	Greater than or equal to أكبر أو يساوي
<	Less than أقل من
<=	Less than or equal to أقل أو يساوي
< >	Not equal to لا يساوي

OTHER عوامل مقارنة أخرى COMPARISON OPERATORS

Examples :

```
SQL>Select empno , enam , mgr, deptno
 From  emp
Where ename in ('ford',allen') ;
Or Where sal between 1000 and 1500;
Or Where mgr in (7902,7566,7788) ;
 Or Where mgr is null ;
```

Operator	Meaning المعنى
Between…and…	Between tow values بين قيمتين
IN (list)	Match any of a list of values List يطابق مع كل قيمة ضمن القائمة
LIKE	Mach a character pattern يطابق نموذج الحرف
IS NULL	Is a null value كقيمة باطلة

Using The Like Operator — استخدامات معامل **LIKE**

Operator	Meaning and Description المعنى والتوصيف
%	**(per cent) Represent any sequence of zero or more characters** تمثل أي سلسلة مكونة من عدة أحرف أو ولا حرف
_	**(under score) Represent any single characters** تمثل أي حرف وحيد

Example :

```
SQL>select ename , hiredate
        From emp
        Where hiredate like '%81' ;
 Or    Where  ename  like  '_ A%' ;
```

LOGICAL OPERATOTS المعاملات المنطقية

Operator	Meaningالمعنى
AND	Returns true if both component condition is true يعيد قيمة صحيحة عندما يكون كل من الشرطين صحيحاً
OR	Returns true if either component condition is true يعيد قيمة صحيحة عندما يكون أحد الشرطين صحيحاً
NOT	Returns true if the following condition is false يعيد قيمة صحيحة عندما تكون العبارة خاطئة

Example :
SQL>select * from emp
Where sal >= 100
AND Job='clerk' ;

AND true table **جدول الحقيقة للمعامل AND**

Unknown	False	True	**AND**
Unknown	False	True	True
False	False	False	False
Unknown	False	Unknown	Unknown

Example :
SQL>select * from emp
Where sal >= 100
OR Job='clerk' ;

OR true table **جدول الحقيقة للمعامل OR**

Unknown	False	True	**OR**
True	True	True	True
Unknown	False	True	False
Unknown	Unknown	True	Unknown

Example :
SQL>select * from emp
Where Job is **NOT** in ('clerk','manager');

NOT true table **جدول الحقيقة للمعامل NOT**

Unknown	False	True	**NOT**
Unknown	True	False	

Example :
SQL>select ename,job,sal
From emp
Where Job = 'salesman'
Or Job = 'president'
And sal > 1500 ;

Rules of Precedence **قوانين أسبقية المعاملات**

Operator المعامل	رقم الأولوية
All comparison operators كافة معاملات المقارنة	١
NOT	٢
AND	٣
OR	٤

عبارة Order By (Order by Clause) :

Sort rows with the order by clause

اختيار الصفوف باستخدام عبارة Order By
تأتي عبارة Order By في آخر عبارة الاختيار Select .
مثال :

SQL>select ename , job , hiredate
from emp
Order By hiredate ;

تقوم عبارة Order By بترتيب طريقة ظهور الصفوف المسترجعة

Order By specifies the order in witch the retrieved rows are displayed .if order by clause not used the sort order is undefined and the Oracle server may not fetch rows in the same order for the same query twice .

إذا لم تستخدم عبارة الترتيب Order By تصبح طريقة الترتيب غير محددة وربما لا يقوم مخدم أوراكل بإعادة نفس نتيجة الاستفسار بنفس الترتيب مرتين .

استخدام DESC مع عبارة الترتيب Order By (الترتيب بشكل تنازلي): Sorting with descending order

افتراضياً تقوم عبارة Order by بترتيب الصفوف بشكل تصاعدي وباستخدام DESC مع عبارة Order by يصبح الترتيب بشكل تنازلي .

The default sort order is ascending . طريقة الترتيب الافتراضية هي بشكل تصاعدي

مثال :

```
SQL>select ename , job , hiredate
     from  emp
     Order by hiredate DESC ;
```

Notices :

Numeric values are displayed with the lowest values first	تظهر القيم الرقمية بدءاً بأقل قيمة ثم الأكبر والأكبر
Date values are displayed with the earliest values first	تظهر قيم التواريخ حسب أبكر قيمة أولاً
Character values are displayed in alphabetical order	تظهر القيم النصية حسب التسلسل الأبجدي
Null values are displayed last for ascending sequences and first for descending sequences	تظهر القيم الفارغة أخيراً في الترتيبات التصاعدية وأولاً في الترتيبات التنازلية

Sorting by Multiple columns — الترتيب وفق حقول متعددة أي حسب أكثر من عمود

مثال :

```
SQL>select ename , deptno , sal
     from  emp
     Order by deptno , sal  DESC ;
```

You can sort by a column that is not in the select list . يمكنك الترتيب وفق عمود غير موجود ضمن قائمة الحقول المختارة

The sort limit is the number of columns in the given table. يمكن الترتيب وفق عدد حقول موافق لعدد حقول الجدول المرتب

SQL Functions — توابع SQL

معظم الدوال التالية مخصصة لنسخة الـ SQL التابعة لأوراكل حصراً

Most of the functions are specific to Oracle version of SQL .

Two Types of SQL Functions — **نوعي دوال الـ SQL**

1-Single – Row Functions — ١. توابع وحيدة الصف

2-Multiple – Row Functions — ٢. توابع متعددة الصفوف

١. التوابع وحيدة الصف : تعمل هذه التوابع بصف واحد فقط وتعيد نتيجة واحدة لكل صف

1-**Single–Row Functions** : These functions operate on single row only and return one result per row.

٢. التوابع متعددة الصفوف : وهي تركيب مجموعات من الصفوف تعطي نتيجة واحدة لكل مجموعة صفوف .

2- Multiple – Row Functions :these functions multiplate groups of rows to give one result per group of rows .

Character Functions — الدوال الحرفية

1-**Case Conversion Functions :** (Lower , Upper , Initcap) — ١. دوال تحويل الحالة

2-**Character Manipulation Functions** :(Concat , Substr ,Length , Instr, Lpad) — ٢. دوال معالجة الأحرف

Function	Purpose الهدف
Lower (اسم العمود)	Convert alpha character values to lowercase يقوم بتحويل الرموز والأحرف إلى أحرف صغيرة
Upper(اسم العمود)	Convert alpha character values to uppercase يقوم بتحويل الأحرف إلى أحرف كبيرة
Initcap(اسم العمود)	Convert alpha character values to uppercase for the first letter of each word and all other letters in lowercase يقوم بتحويل الحرف الأول من كل كلمة على حرف كبير وبقية الأحرف من كل كلمة صغيرة

Function	Purpose الهدف
Concat(اسم العمود١ , Expre.. (اسم العمود٢,...	يقوم Concatenates the first character value to the second character value بدمج التعبير في العمود الأول بسلسلة واحدة مع التعبير في العمود الثاني ويمكن إضافة نص وهو مرادف للرمز || الذي رأيناه سابقاً
Substr(column,6,4)	يقوم هذا التابع بالقص ابتداءً من الحرف رقم ٦ وينتهي بعد ٤ حروف بتعبير آخر Substr(column,start,count) إذا لم يحدد العدد أي القيمةCount يعطينا الأحرف بدءاً منstart حتى نهاية الكلمة
Length(column)	Return the number of character in value يعيد عدد أحرف كل قيمة من قيم الحقل المحدد column
Instr(column,'set',start,n)	يقوم بالبحث عن حرف أو كلمة ضمن سلسلة أحرف يمثل column اسم الحقل وset الحرف أو مجموعة الأحرف المراد البحث عنها ويجب أن تكون بين فاصلتين علويتين وstart رقم الحرف الذي يبدأ منه البحث وn رقم التكرار لأننا قد نقوم بالبحث عن التكرار الثاني أو الرابع للحرف أو الكلمة
Lpad(column,length,'set')	يقوم بإضافة أحرف أو رموز إلى يسار الكلمة يمثل column اسم الحقل و length طول السلسلة الحرفية المطلوب بعد الإضافة وset الحرف أو مجموعة الأحرف المراد إضافتها
Rpad(column,length,'set')	يقوم بإضافة أحرف أو رموز إلى يمين الكلمة يمثل column اسم الحقل و length طول السلسلة الحرفية المطلوب بعد الإضافة وset الحرف أو مجموعة الأحرف المراد إضافتها

ملاحظة : يجب التمييز بين الأحرف الكبيرة والصغيرة ضمن التوابع السابقة أي في حالة البحث عن حرف صغير فإننا لا نجده إذا كان بهيئة حرف كبير .

مثال ١ :

```
SQL> select 'the job title for' || Initcap(ename) ||
       'is' || Lower(Job) As Employee Details
       from  emp ;
```

مثال ٢ :

```
SQL> select Lpad(ename,10,'F')||Rpad(ename,10,'J')
       As Name
       from  emp ;
```

مثال ١ :

```
SQL>select  ename , Instr(ename,'E',1,1)
       As Number
       from  emp ;
```

١ الأولى تمثل رقم الحرف الذي يبدأ منه البحث وهنا يبدأ البحث من الحرف الأول و ١ الثانية تعبر عن رقم التكرار للحرف ويعيد التابع رقم مكان توضع الحرف المطلوب أو الكلمة المطلوبة والموضوعة ضمن فواصل علوية .

Using The Round Function — استخدام دالة التدوير (التقريب)

The round function rounds the column expression or value to n decimal places .

تقوم دالة التدوير بتقريب العدد إلى n فاصلة عشرية أو عدد عشري .

مثال ١ :

```
SQL> select  Round ( 45.923 , 2) , Round( 45.923 , -1 )
       from  dual ;
```

Round (45.923 , 0**)** = 46 , **Round (** 45.923 , 2**)** = 45.92 , **Round (** 45.923 , -1**)** = 50 ,
Round (499.923 , -2**)** = 500 , **Round (** 45.923 , 1**)** = 45.9

If the second argument is 0 or is missing the value is rounded to zero decimal places . If the second argument is 2 the value is rounded to two decimal places . If the second argument is -2 the value is rounded to two decimal places to the left .

إذا كان المعامل الثاني لدالة التدوير مساوياً الصفر أو غير موجود يتم حذف الأرقام العشرية ويتم التقريب إلى أول رقم صحيح وإذا كان المعامل الثاني لدالة التدوير مساوياً للعدد 2 يتم تقريب الرقم إلى أقرب رقمين بعد الفاصلة وفي حالة كون المعامل الثاني مساوياً لـ 2- يتم تقريب الرقم منزلتين عشريتين إلى اليسار .

Using The TRUNC Function — استخدام دالة القطع (البتر)

The TRUNC function truncates the column expression or value to n decimal places .

تقوم دالة القطع بقطع العدد إلى n مرتبة أو فاصلة عشرية .

مثال ١ :

```
SQL> select  TRUNC ( 45.923 , 2) , TRUNC ( 45.923 , -1 )
       from  dual ;
```

TRUNC (45.923**)** = 45 , **TRUNC (** 45.923 , 2**)** = 45.92 , **TRUNC (** 45.923 , -1**)** = 40 ,
TRUNC (499.923 , -2**)** = 400 , **Round (** 45.923 , 1**)** = 45.9

The TRUNC function works with arguments similar to those of the Round function . If the second argument is 0 or is missing the value is truncate to zero decimal places . If the second argument is 2 the value is truncate to two decimal places . If the second argument is -2 the value is truncate to two decimal places to the left .

تعمل دالة القطع تعمل نفس عمل دالة التدوير مع المعاملات إذا كان المعامل الثاني لدالة القطع مساوياً الصفر أو غير موجود يتم بتر الأرقام العشرية وإذا كان المعامل الثاني لدالة التدوير مساوياً للعدد 2 يتم قطع الرقم إلى أول رقمين بعد الفاصلة وفي حالة كون المعامل الثاني مساوياً لـ 2- يتم القطع منزلتين عشريتين إلى اليسار .

Using The MOD Function — استخدام دالة باقي القسمة (MOD)

The MOD function finds the remainder of value1 divided by value2 .

تقوم دالة باقي القسمة بإيجاد باقي قسمة القيمة١ على القيمة٢ .

مثال ١ :

```
SQL> select  ename , sal , comm, MOD(sal , comm)
       from  dual
       Where  job = 'SALESMAN' ;
```

Working with Dates — العمل مع التواريخ (Dates)

Oracle stores dates in an internal numeric format : century , year , month , day , hours ,minutes , seconds .

يخزن أوراكل التواريخ بشكل عددي داخلي : قرن، سنة ، شهر ، يوم ، ساعات ، دقائق ، ثواني .

The default date format is DD-MON-YY — الشكل الافتراضي للتاريخ هو DD-MON-YY

SYSDATE is a function returning current date an d time . — SYSDATE إجراء يعيد التاريخ والوقت الحالي

سنقوم باستخدام جدول يسمى DUAL لإظهار نتيجة هذا الإجراء وهو جدول وهمي فارغ يحتوي على حقل حرفي واحد باسم dummy .ويحتوي على قيمة وحيدة هي X يستخدم هذا الجدول لإعادة قيمة وحيدة وهو منشأ من قبل المستخدم SYSTEM

Valid oracle dates are between January 1.4712 B.C and December 31.9999 A.D

يمكنك استخدام التابع أو الإجراء SYSDATE مع أي جدول وذلك باستخدام عبارة select واختيار اسم للحقل الجديد أو بدون اسم

```
SQL> select  SYSDATE  today from emp ;
SQL> select  SYSDATE  from dual ;
```

أو

Arithmetic With Dates — العمليات الحسابية على التواريخ

تشمل العمليات التالية :

-Add or subtract a number to or from a date for a resultant date value — إضافة أو حذف رقم من أو إلى تاريخ لإنتاج تاريخ

-Subtract two dates to find the number of days between those dates . — طرح تاريخين لإيجاد عدد الأيام بينهما

-Add hours to a date by dividing the number of hours by 24 — إضافة ساعات للتاريخ بتقسيم عدد الساعات على ٢٤

يمكنك إنجاز العمليات الحسابية على التاريخ باستخدام المعاملات الحسابية كالجمع والطرح لأن التاريخ يخزن بشكل رقم كما يمكنك إضافة أو طرح رقم ثابت من وإلى التاريخ .

You can perform calculation using arithmetic operators such as additional and subtraction . You can add and subtract number constants as well as dates .

You can perform the following operations:

العمليةOperation	النتيجةResult	الوصفDescription
Date + number	Date	إضافة عدد أيام إلى التاريخ
Date - number	Date	طرح عدة أيام من التاريخ
Date - Date	Number of days	طرح تاريخ من آخر
Date + number/24	Date	إضافة عدد من الساعات إلى التاريخ

مثال ١ :

```
SQL> select  ename , (sysdate-hiredate)/ 7 weeks
     from  emp
     Where  deptno = 10 ;
```

يظهر هذا المثال أسماء وعدد أسابيع العمل لجميع الموظفين أو العمال في القسم رقم ١٠ لأن الفرق بين التاريخ الحالي الناتج عن التابع Sysdate وتاريخ الاستخدام للعامل Hiredate يعطي المدة الزمنية بالأيام وبالقسمة على ٧ تكون النتيجة بالأسابيع .

DATE FUNCTIONS	
FUNCTION	**DESCRIPTION**
MONTHS_BETWEEN(D1,D2)	عدد الأشهر بين تاريخين D1 و D2
ADD_MONTHS(D1,n)	إضافة n شهر إلى التاريخ D1
NEXT_DAY(DATE ,'CH')	يعطينا تاريخ أول يوم اسمه CH بعد التاريخ Date
LAST_DAY(DATE)	يعطي تاريخ آخر يوم من الشهر المذكور
ROUND(DATE , 'DAY' **or** 'MONTH' **or** 'YEAR')	في حالة Day يقوم التابع بتقريب التاريخ إلى تاريخ أقرب يوم أحد من التاريخ الحالي وفي حالة Month يقوم بتقريب التاريخ إلى تاريخ أقرب أول شهر من التاريخ الحالي أما في حالة Year فإنه يقرب التاريخ إلى أقرب أول يوم من أقرب عام
TRUNC(DATE , 'DAY' **or** 'MONTH' **or** 'YEAR')	في حالة Day يقوم التابع بتقريب التاريخ إلى تاريخ أقرب يوم أحد قبل التاريخ الحالي وفي حالة Month يقوم بتقريب التاريخ إلى تاريخ أقرب أول شهر قبل التاريخ الحالي أما في حالة Year فإنه يقرب التاريخ إلى تاريخ بداية العام الحالي

أمثلة عن استخدام توابع التواريخ :

مثال ١ :

```
SQL> select  empno , hiredate,
     Months_between(sysdate,hiredate) Tenure,
     Add_months(hiredate,6)   Review ,
     Next_day(hiredate,'Friday') , Last_day(hiredate)
     from  emp
     Where  Months_between(sysdate,hiredate )<200 ;
```

For all employees employed for fewer than 200 month .display the employee number ,hiredate , number of months employed , six-month review date , first Friday after hire date ,and last day of the month when hired .

من أجل جميع الموظفين الذين تم استخدامهم لأقل من ٢٠٠ شهر قم بإظهار رقم الموظف وتاريخ الاستخدام (التوظيف) وعدد أشهر الاستخدام و تاريخ التوظيف مضاف إليه ٦ أشهر وتاريخ أول يوم جمعة بعد الاستخدام وتاريخ آخر يوم من شهر الاستخدام

أمثلة :

ROUND('25-JUL-95' , 'MONTH') →→→→ 01-AUG-95
ROUND('25-JUL-95' , 'YEAR') →→→→→ 01-JAN-96
TRUNC('25-JUL-95' , 'MONTH') →→→→ 01-JUL-95
TRUNC ('25-JUL-95' , 'YEAR') →→→→→ 01-JAN-95

مثال :

```
SQL> select  empno , hiredate,
     Round (hiredate , 'MONTH') ,
 Trunc ( hiredate , 'MONTH')
     from  emp
     Where  hiredate LIKE '%82'  ;
```

EMPNO	HIREDATE	ROUND(HI	TRUNC(HI
----------	----------	--------	------
٨٢/٠١/٠١	٨٢/٠٢/٠١	٨٢/٠١/٢٣	٧٩٣٤

Compare the hire dates for all employees who started in 1982 display the employee number ,hire date , and month started using the ROUND and TRUNC functions.

CONVERSION FUNCTIONS توابع التحويل

Implicit Datatype Conversion التحويل الضمني لأنواع البيانات

For assignments the Oracle can automatically convert the following :
من أجل المهام يستطيع أوراكل تحويل ما يلي بشكل أوتوماتيكي :

FROM	TO
VARCHAR2 or CHAR محرفي	NUMBER رقم
VARCHAR2 or CHAR محرفي	DATE تاريخ
NUMBER رقم	VARCHAR2 محرفي
DATE تاريخ	VARCHAR2 محرفي

الفرق بين CHAR و VARCHAR2 أن العرض الأعظمي لعمود من النمط CHAR ٢٠٠٠ محرف أما العرض الأعظمي لعمود من النمط VARCHAR2 فيصل إلى 4000 محرف .

For expression evaluation the Oracle Server can automatically convert the following :
من أجل عبارة التقدير يستطيع أوراكل تحويل ما يلي بشكل أوتوماتيكي :

FROM	TO
VARCHAR2 or CHAR محرفي	NUMBER رقم
VARCHAR2 or CHAR محرفي	DATE تاريخ

NOTE: CHAR to NUMBER conversions succeed only if the character string represents a valid number .

CHAR to DATE conversions succeed only if the character string has the default format DD-MON-YY
يتم التحويل التلقائي من المحارف إلى الأرقام بنجاح في حالة كون المحارف بصورة أرقام كما يتم التحويل التلقائي الضمني من المحارف إلى التاريخ عندما يكون لسلسلة المحارف الشكل الافتراضي للتاريخ DD-MON-YY .

Explicit Datatype Conversion التحويل الصريح لأنواع البيانات

SQL provides three functions to convert a value from one datatype to another :
تحوي لغة الـ.SQL ثلاثة دوال تحويل لأنواع البيانات وهي :

FUNCTION الدالة	Purpose الهدف
TO_CHAR(number or date ,*'fmt'*)	Converts number or date value to a VARCHAR2 character string with format model *'fmt'* تحويل الأرقام أو التواريخ إلى سلسلة محرفية من الشكل *'fmt'*
TO_NUMBER(char , *'fmt'*)	Converts a character string containing digits to a number with the optional format model *'fmt'* تحويل السلسلة المحرفية التي تحتوي أرقام إلى رقم من الشكل *'fmt'*
TO_DATE(char , *'fmt'*)	Converts a character string representing a date to a date value according to the *fmt* specified (If *fmt* is omitted format is DD-MON-YY) تحويل السلسلة الحرفية التي تمثل التاريخ إلى قيمة تاريخ حسب الشكل *fmt* (بحيث يكون شكل النمط *fmt* بالشكل الافتراضي لأوراكل وهو DD-MON-YY)

TO_CHAR Function with dates **TO_CHAR** (date , *'fmt'*) to display date in a specific format .
The format model *fmt* must be enclosed in single quotation marks and is case sensitive and separated From the date value by a comma
يجب أن تكون صيغة التاريخ المطلوبة *fmt* محصورة بين علامتي اقتباس مفردتين ' *fmt* ' و مفصولة عن التاريخ بفاصلة .

مثال :

```
SQL> select  empno , TO_CHAR( hiredate , 'MM/YY' ) Month_Hired
     from  emp
     Where  ename = ' BLAKE '  ;
```

Elements of Date Format Model **عناصر نموذج شكل التاريخ**

Element	Description
YYYY	Full year in numbers كامل التاريخ بالأرقام
YEAR	Year spelled out التاريخ مكتوب بالكلمات
MM	Two digit value for month قيمة التاريخ من رقمين
MONTH	Full name of the month اسم الشهر كامل
DY	Three letter abbreviation of the day of the week ثلاثة أحرف اختصار من اسم اليوم
DAY	Full name of the day اسم اليوم كاملاً
SCC or CC	Century prefixes BC date with القرن الذي يقع التاريخ ضمنه
Y,YYY	Year with a comma in the position رقم العام مع فاصلة بنفس الموضع
YYY or YY or Y	Last three ,two , or one digits of year آخر ٣ أرقام أو رقمين أو رقم من العام
SYYYY	رقم العام مع وجود الإشارة مثل 1000 B.C هو 1000-
IYYY or IYY or IY or I	Four ,three ,two or one digit year based on the ISO standard . أربع أو ثلاث خانات أو خانتين أو خانة واحدة للعام من المقياس ISO
BC or AD	يظهر المؤشر BC أو AD بحسب التاريخ و لهما نفس العمل - BC or AD indicator
B.C. or A.D.	نفس السابق مع علامات توقف (نقاط) – BC or AD indicator with Periods
Q	رقم الربع من العام الذي يقع – Quarter of year
MON	Three letter abbreviation - اسم الشهر ثلاثة أحرف اختصار
RM	Roman mineral month - رقم الشهر بالترميز الروماني
WW or W	Week of year or month – رقم الأسبوع في السنة أو الشهر
DDD or DD or D	Day of year ,month , or day - رقم اليوم في السنة أو الشهر أو الأسبوع
J	عدد الأيام منذ ٣١ كانون الأول عام ٤٧١٣ قبل الميلاد

Use the formats listed in the following tables to display time information and literals and to change numerals to spelled numbers

استخدم قائمة الأشكال الموجودة في الجدول التالي لإظهار معلومات الوقت والأحرف و تحويل الأعداد إلى أرقام مقروءة

Element	Description
AM or PM	Meridian indicator – مؤشر منتصف النهار
A.M. or P.M.	Meridian indicator with periods - مؤشر منتصف النهار مع علامات توقف
HH or HH12 or HH24	الساعة في اليوم و ترميز اليوم ١٢ ساعة (١-١٢) أو ٢٤ ساعة (٠-٢٤)
MI	الدقيقة في الساعة (٠-٥٩)
SS	الثانية في الدقيقة (٠-٥٩)
SSSSS	Seconds past midnight - عدد الثواني منذ منتصف الليل
TH	Ordinal number - (4TH – 24TH) لاحقة لعدد ما مثل
SP	Spelled out number (DDSP for FOUR) – لاحقة لعدد تجبره على كتابة العدد بشكل نص
SPTH or THSP	Spelled out number (DDSPTH for FOURTH) لاحقة تركيبية تجمع بين الاثنين

Using the TO_CHAR Function with dates **استخدام تابع التحويل To_Char مع التواريخ**

مثال :

```
SQL> select  ename ,
        TO_CHAR( hiredate , 'MM/YY' ) Month_Hired
        from  emp ;
```

Example : Modify the slide example to display the dates in a format that appears as Seventh of February 1981 08:00:00 AM .

حول التاريخ ليظهر بالشكل Seven of February 1981 08:00:00 AM

```
SQL> select  ename ,
        TO_CHAR(hiredate , 'fmDdspth  "of" Month  yyyy fmHH:MI:SS AM' )
        From emp ;
```

TO_CHAR Function with Numbers **استخدام تابع التحويل To_Char مع الأعداد**

TO_CHAR(number , *'fmt'*)

Use these formats with the TO_CHAR function to display a number value as a character:

استخدم الأشكال التالية مع التابع TO_CHAR لتظهر القيم الرقمية كأحرف :

٩	Represents a number تمثل الرقم
٠	Forces a zero to displayed تجبر الصفر على الظهور
$	Places a floating dollar sign يظهر علامة الدولار
L	Use the floating local currency symbol رمز الفاصلة العائمة
.	Prints a decimal point يطبع الفاصلة العشرية
,	Prints a thousand indicator يطبع مؤشر الآلاف

مثال :

```
SQL> select  TO_CHAR( sal , '$99,999 ' ) SALARY
       FROM emp
       Where  ename = 'SCOTT' ;
```

When working with number values such as a character string you should convert those numbers to the character datatype using the TO_CHAR function which translates a value of NUMBER datatype to VARCHAR2 datatype . This technique is specially useful with concatenation .

عند العمل مع القيم الرقمية على أنها أحرف عليك تحويل تلك الأرقام إلى النمط الحرفي باستخدام التابع TO_CHAR والذي يحول نمط البيانات العددية إلى النمط الحرفي VARCHAR2 وهذه التقنية مفيدة بشكل خاص في السلاسل .

TO_NUMBER and TO_DATE Functions

Convert a character string to a number format using the TO_NUMBER function .

TO_NUMBER(char , 'fmt') تحويل السلسلة الحرفية إلى هيئة رقمية باستخدام التابع TO_NUMBER

Convert a character string to a date format using the TO_DATE function .

TO_DATE(char , 'fmt') تحويل السلسلة الحرفية إلى هيئة تاريخ باستخدام التابع

مثال :

```
SQL> select  ename , hiredate
       FROM emp
       Where  hiredate = TO_DATE( 'February  22 ,1981 ' ,'Month  dd , YYYY') ;
```

NVL function **تابع تعويض القيمة الخالية**

Convert null to an actual value يحول القيمة الخالية إلى قيمة حقيقية

Datatypes that can be used are date, character, and number.أنواع المعطيات التي يستخدم معها التواريخ والأحرف والأرقام

EX: **NVL(**comm,0**)** , **NVL(**hiredate,'01-JAN-97'**)** , **NVL(**job ,'No Job Yet'**)**

مثال :

```
SQL> select  ename , sal , comm , (sal*12)+NVL(comm.,0)
       FROM emp ;
```

Note: If any column value in an expression is null the result is null .

إذا كانت القيمة في أي عمود خالية وأجرينا عليها عملية حسابية تنتج لدينا القيمة الخالية .

DECODE function — تابع البحث والمطابقة وإجراء العمليات بشكل منفصل

الشكل العام

DECODE(Value , if 1 ,then 1, if 2 , then 2 , if 3 , then 3 ,….,else)

- The DECODE function decodes an expression in a way similar the IF-THEN-ELSE logic used in various languages . The DECODE function decodes expression after comparing it to each search value . If the expression is the same as search result is returned .If the default value is omitted a null value is returned where a search value dose not mach any of the result values .

يقوم التابع DECODE بحل العبارة الجبرية بطريقة مشابهة لعبارة IF-THEN-ELSE المنطقية والتي تستعمل بشكل في لغات متنوعة . يحل التابع العبارة الجبرية بعد مقارنتها مع كل قيمة من قيم البحث إذا كان التعبير مطابق لقيمة البحث يعيد قيمة . أما إذا كانت القيمة الأصلية محذوفة أو فارغة تعاد قيمة فارغة وذلك عندما لا تتطابق قيمة البحث مع أي قيمة من قيم البحث .

مثال :

```
SQL> select  job , sal
      DECODE( job, 'ANALYST' , sal*1.1 ,
                   'CLERK' , sal*1.15 ,
                   'MANAGER' , sal*1.20 ,
                              sal )
                REVISED_SALARY
      FROM emp ;
```

Display the applicable tax rate for each employee in department 30 .

استعراض معدل الضريبة الملائم لكل موظف في القسم ٣٠ .

مثال :

```
SQL> select  ename , sal
      DECODE(TRUC(sal/1000,0) ,
                    0 , 0.00 ,
                    1 , 0.09 ,
                    2 , 0.20 ,
                    3 , 0.30 ,
                    4 , 0.40 ,
                    5 , 0.42 ,
                    6 , 0.44 ,
                       0.45 )  TAX_RATE
      FROM emp
      WHERE  deptno = 30 ;
```

NESTING FUNCTIONS SAMPLES — **أمثلة عن التوابع المتداخلة**

مثال1 :

```
SQL> select  ename ,
      NVL(TO_CHAR(mgr) , 'No Manager')
       From    emp
       Where  mgr  Is NULL ;
```

مثال2 :

```
SQL> select  TO_CHAR(NEXT_DAY(ADD_MONTHS
             (hiredate , 6 ) , 'FRIDAY' ) ,
              'fmDAY , Month  ddth , YYYY)
              " Next 6 Month Review "
       From    emp
       Order By  hiredate ;
```

Displaying data from multiple tables عرض البيانات من جداول متعددة

- -Write SELECT statements to access data from more than one tables using equality and none quality joins
- View data that generally does not meet a join condition by using outer joins .
- Join a table to itself

- كتابة عبارة الاختيار SELECT للوصول إلى البيانات من أكثر من جدول باستخدام الربط المتساوي وغير المتساوي .
- عرض البيانات التي لا تلتقي عادة بشرط ربط باستخدام الروابط الخارجية .
- ربط الجدول بنفسه

Use a join to query data from more than one table استخدام الربط للاستفسار عن البيانات من أكثر من جدول

```
SELECT   table1.column ,  table2.column
FROM     table1 , table2
WHERE    table1.coulumn1 = table2.coulumn2 ;
```

WHERE قم بكتابة شرط الربط ضمن عبارة Write the join condition in the WHERE clause

Prefix the column name with the table name when the same column name appears in more than one table .

يجب أن يسبق اسم الحقل اسم الجدول المحتوى ضمنه إذا كان الحقل يظهر أو موجود في أكثر من جدول .

Types of Joins أنواع الربط

There are two main types of join conditions: هناك نوعان رئيسيان من حالات الربط :

- Equijoins . - الربط المتساوي .
- Non-equijoins . - الربط الغير متساوي .

Additional join methods include the following : تتضمن طرق الربط الإضافية ما يلي :

+ Outer joins . + الربط الخارجي .
+ Self joins . + الربط الذاتي .
+ Set operators . + مجموعة المشغلات .

الربط المتساوي (الداخلي) Equijoins or Inner Joins :

يجب أن يكون هناك حقل مشترك على الأقل بين الجدولين المرتبطين بحيث تكون القيم في الحقول المتقابلة متساوية من كلا الجدولين .
هذا النوع من الربط يشكل تتمة للمفتاح الأساسي والثانوي .

EMP

EMPNO	ENAME	DEPTNO
7369	SMITH	20
7499	ALLEN	30
7521	WARD	30
7566	JONES	20
7654	MARTIN	30
7698	BLAKE	30
7782	CLARK	10
7788	SCOTT	20
7839	KING	10
7844	TURNER	30
7876	ADAMS	20

Foreign Key

DEPT

DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON

Primary Key

Retrieving records with Equijoins **استرجاع السجلات مع الربط المتساوي**

مثال :

```
SQL> select  emp , empno , emp.ename , emp.deptno ,
                Dept.deptno , dept.loc ,
                From emp , dept
                Where  emp.deptno=dept.deptno ;
```

The Where clause specifies how the tables are to be joined .

AND حالات بحث إضافية باستخدام المعامل Additional search conditions Using the AND operator

In addition to the join , you may have criteria for your WHERE clause .

بالإضافة إلى الربط قد تحتاج إلى بعض المعايير من أجل عبارة WHERE .

مثال :

```
SQL> select  empno , ename , emp.deptno , loc
                FROM  emp , dept
                WHERE  emp.deptno = dept.deptno
                AND       INITCAP(ename) = 'king' ;
```

Using Table Aliases **استخدام الأسماء المستعارة للجداول**

Simplify queries by using table aliases . تبسيط الاستعلامات باستخدام الأسماء المستعارة للجداول

You can use table aliases instead of table names . Just as a column aliases gives a column another name .a table aliases give a table another name . Table aliases help to keep SQL code smaller , therefore using less memory.

يمكنك استخدام اسم مستعار للجدول بدلاً من اسم الجدول تماماً مثل الأسماء المستعارة للحقول أو الأعمدة . تساعد الأسماء المستعارة للجداول في جعل كود الـ SQL أصغر والذي يستعمل ذاكرة أقل .

مثال :

```
SQL> select  emp.empno , emp.ename , emp.deptno ,
                Dept.deptno , dept.loc
        FROM                emp , dept
        WHERE        emp.deptno = dept.deptno ;
```

بغد تطبيق الأسماء المستعارة على الجداول يصبح المثال السابق بالشكل التالي :

مثال :

```
SQL> select  e.empno , e.ename , e.deptno ,
                d.deptno , d.loc
        FROM                emp e , dept d
        WHERE        e.deptno = d.deptno ;
```

Join more than two tables **الربط لأكثر من جدولين**

مثال :

```
SQL> select  c.name , i.itemid , i.itemotot , o.total ,
        FROM                customer  c , ord  o , item  i
       WHERE        c.custid = o.custid
        AND         o.ordid = i.ordid
        AND         c.name = 'TKP SPORT SHOP' ;
```

حيث الجداول بالشكل التالي :

CUSTOMER

NAME	CUSTID
JOCKSPORTS	100
THE SPORT SHOP	101
VOLLYRITE	102
JUST TENNIS	103
K+T SPORTS	105
SHAPE UP	106

ORD

CUSTID	ORDID
101	610
102	611
104	612
106	601
102	602
106	605

ITEM

ORDID	ITEMID
610	3
611	1
612	1
611	1
601	1
602	1

الربط الغير متساوي Non-Equijoins :

لا يشترط وجود حقل أو أكثر متساوي من الجدولين ولكن يوجد علاقة ما بين حقل أو أكثر من الجدول الأول و حقل أو أكثر من الجدول الثاني و هذه العلاقة ليست مساواة .

The relationship between tables is obtained using an operator other than equal (=) .

Retrieving records with Non-Equijoins **استرجاع السجلات مع الربط الغير متساوي**

في المثال التالي لدينا جدولين الأول EMP والثاني SALGRADE وتوجد علاقة بين الحقل SAL من الجدول EMP والحقلين LOSAL وHISAL وهي أن أي قيمة في الحقل SAL تقع بين القيمتين LOSAL و HISAL .

مثال :

```
SQL> select   e.ename , e.sal ,  s.grade ,
              From      emp e , salgrade s
               Where    e.sal
             Between  s.losat  and  s.hisat  ;
```

الربط الخارجي Outer joins :

You use an Outer join to also see rows that does not usually meet the join condition .

Outer join operator is the plus sign (+) .

يستخدم الربط الخارجي لإظهار الصفوف التي لا تظهر لأنها لا تحقق شرط الربط وكمثال على ذلك المثال القادم إذا استخدمنا الربط المتساوي وأردنا معرفة أسماء الموظفين وأقسامهم من الجدولين DEPT وEMP بحيث يتساوى رقم القسم DEPTNO من الجدول الأول مع رقم القسم من الثاني نجد أن قسم العمليات OPERATIONS ورقمه لا يظهران لدينا في نتيجة الاستعلام وذلك لعدم وجود أشخاص يعملون في ذلك القسم وبالتالي رقم القسم في جدول الموظفينEMP غير موجود وفي حالة أردنا ظهور رقم القسم نستخدم الربط الخارجي والذي يستخدم المعامل (+) كما يلي :

مثال : باستخدام الربط المتساوي

```
SQL> select   e.ename , e.deptno ,  d.dname ,
              From      emp e , dept d
             WHERE  e.deptno = d.deptno ;
```

مثال : و باستخدام الربط الخارجي يصبح كما يلي :

```
SQL> select   e.ename , d.deptno ,  d.dname
              From      emp e , dept d
             WHERE  e.deptno(+) = d.deptno
             ORDER BY   e.deptno ;
```

الربط الذاتي Self joins (Join the table to itself) :

تحتاج في بعض الأحيان إلى ربط الجدول بنفسه فمثلاً لإيجاد اسم مدير كل موظف من الجدول عليك ربط الجدول EMP بنفسه بالشكل التالي حيث MANAGER , WORKER أسماء مستعارة للجدول نفسه EMP :

```
SQL> select worker.ename||' works for '||manager.ename
              From      emp worker , emp manager
               Where    worker.mgr = manager.empno ;
```

EMPNO	ENAME	MGR
٧٣٦٩	SMITH	7902
٧٤٩٩	ALLEN	7698
٧٥٢١	WARD	7698
٧٥٦٦	JONES	7839
٧٦٥٤	MARTIN	7698
٧٦٩٨	BLAKE	7839
٧٧٨٢	CLARK	7839
٧٧٨٨	SCOTT	7566
٧٨٣٩	KING	
٧٨٤٤	TURNER	7698
٧٨٧٦	ADAMS	7788

```
WORKER.ENAME||'WORKSFOR'||MANAGER.ENAME
-----------------------------------------------------------------
SMITH works for FORD
ALLEN works for BLAKE
WARD works for BLAKE
JONES works for KING
MARTIN works for BLAKE
BLAKE works for KING
CLARK works for KING
SCOTT works for JONES
TURNER works for BLAKE
ADAMS works for SCOTT
JAMES works for BLAKE
```

Aggregating Data Using Group Functions: تحصيل البيانات باستخدام توابع قيمة المجموعة

Group functions: Unlike single-row functions operate on sets of rows to give one result per group . These sets may be the whole table or the table split into groups.

توابع قيمة المجموعة : لا تشبه التوابع وحيدة الصف وهي تؤثر على مجموعات من الصفوف لتعطي نتيجة واحدة لكل مجموعة ولكن هذه المجموعات ممكن تكون الجدول كاملاً أو الجدول مقسم إلى مجموعات .

Types of Group Functions **أنماط من توابع قيمة المجموعة**

Function	Description
AVG([DISTICT|ALL] **n)**	Average value of n ignoring null values قيمة المتوسط لـn قيمة ويقوم بتجاهل القيم الفارغة
COUNT(* [DISTICT|ALL] expr **)**	Number of rows where expr evaluates to something other than null (Count all selected rows using * , including duplicates and rows with nulls) يعيد هذا التابع عدد الصفوف في حال كانت القيم مغايرة للقيمة الفارغة null (استخدم * مع التابع ليعيد عدد كل الصفوف إضافة إلى الصفوف المكررة والقيم الفارغة NULL)
MAX([DISTICT|ALL] expr **)**	Maximum value of expr , ignoring null values . أكبر قيمة من قيم expr ويتجاهل القيم الفارغة
MIN([DISTICT|ALL] expr **)**	Minimum value of expr , ignoring null values أصغر قيمة من قيم expr ويتجاهل القيم الفارغة
STDDEV([DISTICT|ALL] x **)**	Standard deviation of n ignoring null values الانحراف المعياري لـn قيمة يتجاهل القيم الفارغة
SUM([DISTICT|ALL] n **)**	Sum values of n , ignoring null values مجموع n قيمة ويتجاهل القيم الفارغة
VARIANCE([DISTICT|ALL] x **)**	Variance of n , ignoring null values التباين لـ n قيمة ويتجاهل القيم الفارغة

- Distinct makes the function consider only nonduplicate values ; ALL makes it consider every value including duplicates . The default is ALL and therefore does not need to be specified .
- The datatypes for the arguments may be CHAR, VARCHAR2, NUMBER, DATE where *exp* is listed.
- All group functions except COUNT(*) ignore null values .To substitute a value for null values . use the NVL function .
- The Oracle Server implicitly sorts the result set in ascending order when using a GROUP BY clause. To override this default ordering. DESC can be used in the ORDER BY clause.

- الأمر Distinct يجعل التابع يأخذ بعين الاعتبار القيم الغير مكررة فقط . أما ALL فيجعل التابع يأخذ بعين الاعتبار كل قيمة بما فيها القيم المكررة . ALL عبارة عن أمر افتراضي ولذلك لست بحاجة لإدخاله.
- أنماط البيانات يمكن أن تكون أي من التالي CHAR , VARCHAR2 , NUMBER , DATE وذلك حسب التعبيرexpr .
- جميع توابع قيمة المجموعة ما عدا (*)COUNT تتجاهل القيم الفارغة null ولكي تقوم باستبدال القيم الفارغة عليك استخدام تابع الاستبدال NVL .
- يقوم مزود أوراكل بشكل ضمني بترتيب مجموعة النتائج بشكل تصاعدي عندما تقوم باستخدام عبارة الترتيب ORDER BY وحتى تقوم بتغيير نمط الترتيب الافتراضي عليك باستخدام الأمر DESC مع عبارة الترتيب ORDER BY .

SAMPLES : أمثلة :

مثال :
```
SQL> select   AVG(sal), MAX(SAL), MIN(sal), MAX(sal)
              From      emp
              WHERE  job LIKE  'SALES%' ;
```

مثال :
```
SQL> select   MIN(ename), MAX(ename)
              From      emp ;
```

باستخدام التابعين MAX و MIN مع البيانات الحرفية تكون النتيجة حسب الترتيب الأبجدي كما في المثال السابق يكون الناتج كما يلي :

```
MIN(ENAME)      MAX(ENAME)
------- ---------      ----------
ADAMS                  WARD
```

مثال :
```
SQL> select   MIN(hiredate), MAX(hiredate)
              From      emp ;
```

مثال :
```
SQL> select   COUNT( * )
              From      emp
              WHERE   deptno = 30 ;
```

مثال :
```
SQL> select   COUNT( comm )
```

```
                From      emp
                WHERE  deptno = 30 ;
```

مثال :

```
SQL> select    COUNT( deptno )
                From      emp ;
```

مثال :

```
SQL> select    COUNT(DISTINCT( deptno) )
                From      emp ;
```

Group functions ignore null values in the column. تتجاهل توابع قيمة المجموعة القيمة الفارغة لكن لا تتجاهل الصفر

مثال : قيمة المتوسط الحسابي للعمود COMM

```
SQL> select    AVG ( comm. )
                From      emp ;
```

The NVL function forces group functions to include null values.

يجبر التابع NVL توابع التصنيف بأن تتضمن القيمة الفارغة كما في المثال التالي .

مثال : باستخدام الربط المتساوي

```
SQL> select  AVG ( NVL (comm. ))
                From      emp ;
```

Creating Groups of DATA: GROUP BY Clause

إنشاء مجموعات بيانات باستخدام عبارة التجميع GROUP BY :

You can use the GROUP BY clause to divide the rows in a table into groups. You can then use the group functions to return summary information for each group.

يمكنك استخدام عبارة GROUP BY لتقسيم الصفوف ضمن الجدول إلى مجموعات . يمكنك عندها استخدام توابع المجموعة لتحصل على خلاصة معلومات عن كل مجموعة .

- You must include the columns in the GROUP BY clause.
- You cannot use the column alias in the GROUP BY clause.

يجب تضمين الأعمدة في عبارة GROUP BY كما لا يمكنك استخدام الأسماء المستعارة معها .

All columns in the SELECT list that are not in group functions must be in the GROUP BY clause.

جميع الأعمدة المذكورة في قائمة عبارة الاختيار SELECT والتي لم تطبق عليها التوابع السابقة يجب أن تذكر ضمن عبارة GROUP BY كما في المثال التالي :

مثال :

```
SQL> select    deptno , AVG ( sal )
                From      emp
                GROUP BY  deptno ;
```

DEPTNO	AVG(SAL)
10	2916.66667
20	2175
30	1566.66667

The GROUP BY column does not have to be in the SELECT list .

ليس من الضروري أن يكون العمود المذكور في عبارة GROUP BY موجوداً ضمن قائمة الأعمدة المذكورة في عبارة SELECT .

مثال :

```
SQL> select    AVG ( sal )
                From      emp
                GROUP BY  deptno ;
```

AVG(SAL)
2916.66667
2175
1566.66667

You can use the group function in the ORDER BY clause.

يمكنك استخدام توابع التصفية ضمن عبارة GROUP BY .

مثال :

```
SQL> select    deptno, AVG ( sal )
                From      emp
                GROUP BY  deptno
                ORDER BY  AVG(SAL) ;
```

DEPTNO	AVG(SAL)
30	1566.66667
20	2175
10	2916.66667

Using the GROUP BY Clause on Multiple Columns

استخدام عبارة GROUP BY مع أعمدة متعددة :

You can return summary results for groups and subgroups by listing more than one GROUP BY column
You can determine the default sort order of the results by the order of columns in the GROUP BY clause
يمكنك الحصول على ملخص عن المجموعات والمجموعات الثانوية بإدخال أكثر من عمود ضمن عبارة GROUP BY .
كما يمكنك أن تقوم بتغيير طريقة الترتيب الافتراضية للنتائج بترتيب الأعمدة ضمن عبارة GROUP BY .

مثال :

```
SQL> select    deptno, job, SUM ( sal )
         From      emp
         GROUP BY  deptno, job ;
```

DEPTNO	JOB	SUM (SAL)
١٠	CLERK	1300
١٠	MANAGER	2450
١٠	PRESIDENT	5000
٢٠	ANALYST	6000
٢٠	CLERK	1900
٢٠	MANAGER	2975
٣٠	CLERK	950
٣٠	MANAGER	2850
٣٠	SALESMAN	5600

Any column or expression in the SELECT list that is not an aggregate function must be in the GROUP BY clause.
أي عمود أو تعبير ضمن قائمة عبارة SELECT وليس تابع تجميع يجب أن يكون موجوداً ضمن عبارة GROUP BY وإلا ينتج خطأ ويكون عليك تصحيحه ليصبح مثل المثال التالي :

مثال :

```
SQL> select    deptno, COUNT ( ename )
         From      emp
         GROUP BY  deptno ;
```

DEPTNO	COUNT (ENAME)
١٠	3
٢٠	5
٣٠	6

You cannot use the WHERE clause to restrict groups.
You use the HAVING clause to restrict groups.
ليس بإمكانك استخدام عبارة WHERE من أجل تقييد المجموعات ولكنك تستخدم بدلاً عنها عبارة HAVING .
تعمل تعليمة HAVING عملاً مشابهاً لعمل تعليمة WHERE مع الفرق أنها مرتبطة بنتيجة تنفيذ توابع مجموعة أي أنها تأتي مع توابع المجموعة وعبارة GROUP BY كما يلي :

مثال :

```
SQL> select    deptno, AVG ( sal )
         From      emp
         GROUP BY  deptno
         HAVING    AVG (sal) > 2000 ;
```

DEPTNO	AVG(SAL)
١٠	٢٩١٦,٦٦٦٦٧
٢٠	٢١٧٥

Excluding Group Results: HAVING Clause

استخدام عبارة HAVING لاستثناء مجموعات من النتائج:

You use the HAVING clause to specify which groups are to be displayed. Therefore you further restrict the groups on the basis of aggregate information.
تستخدم عبارة HAVING لتخصيص المجموعات التي نريد إظهارها ولذلك من المستحسن حصر المجموعات ضمن معلومات التجميع .

مثال :

```
SQL> select    deptno, MAX ( sal )
         From      emp
         GROUP BY  deptno
         HAVING    MAX (sal) > 2000 ;
```

DEPTNO	MAX(SAL)
١٠	٥٠٠٠
٢٠	٣٠٠٠
٣٠	٢٨٥٠

You can use the GROUP BY clause without using a group function in the SELECT list.
If you restrict rows based on the result of a group function. You must have a GROUP BY clause as well as the HAVING clause.
بإمكانك استخدام عبارة GROUP BY بدون استخدام تابع تجميع ضمن عبارة SELECT .
إذا كنت تريد تقييد الصفوف ضمن نتيجة تابع التجميع عليك أن تستخدم GROUP BY بالإضافة إلى عبارة HAVING .

مثال :

```
SQL> select    deptno, AVG ( sal )
               From      emp
               GROUP BY  deptno
               HAVING    MAX (sal) > 2900 ;
```

```
DEPTNO   AVG(SAL)
--------  --------------
 ١٠       ٢٩١٦,٦٦٦٦٧
 ٢٠       ٢١٧٥
```

مثال :

```
SQL> select    job, SUM ( sal )  PAYROLL
               From      emp
               WHERE  job not like 'sales%'
               GROUP BY  job
               HAVING    SUM (sal) > 5000
               ORDER  BY  SUM(sal) ;
```

```
JOB            PAYROLL
----------      ---------
SALESMAN        5600
ANALYST         6000
MANAGER         8275
```

Nesting Group Functions

تداخل توابع قيمة المجموعة

مثال :

```
SQL> select    MAX(AVG ( sal ) )
               From      emp
               GROUP BY  deptno ;
```

```
MAX(AVG(SAL))
-------------------
٢٩١٦,٦٦٦٦٧
```

Subqueries الاستفسارات الجزئية

الصيغة العامة

```
Select  select_list
From  table
Where  expr  operator
                (select    select_list
                 From      table ) ;
```

- The subquery (inner query) executes once before the main query .
- The result of the subquery is used by the main query (outer query).

- الاستفسار الجزئي (الداخلي) ينفذ أولاً قبل الاستفسار الرئيسي.
- تستخدم نتيجة الاستفسار الجزئي أو الداخلي من قبل الاستفسار الرئيسي أو الخارجي.

A subquery is a SELECT statement that is embedded in a clause of another SELECT statement. You can build powerful statements out of simple ones by using subqueries. They can be very useful when you need to select rows from a table with a condition that depends on the data in the table itself.

الاستفسار الجزئي هو عبارة SELECT مغلفة بعبارة SELECT أخرى. يمكنك بناء عبارات فعالة خارج عبارات بسيطة باستخدام الاستفسارات الجزئية.تكون مفيدة جداً عندما تريد اختيار صفوف من جدول مع شرط يعتمد على بيانات التابع نفسه.

مثال :

```
SQL> Select    ename
       From      emp
       WHERE  sal >
                 ( SELECT  sal
                   From emp
                   WHERE  empno = 7566 ) ;
```

2975

ENAME
SCOTT
KING
FORD

Guidelines for Using Subqueries **إرشادات عامة لاستخدام الاستفسارات الجزئية**

- Enclose subqueries in parentheses.
- Place subqueries on the right slide of the comparison operator.
- Do not add an ORDER BY clause to a subquery.
- Use single-row operators with single-row subquery.
- Use multiple-row operators with multiple-row subqueries.

- قم بوضع الاستعلامات الجزئية ضمن أقواس .
- قم بوضع الاستعلامات الجزئية في الجهة اليمنى من معاملات المقارنة.
- لا تقم بإضافة عبارة ORDER BY إلى الاستفسار الجزئي.
- استخدم المعاملات وحيدة الصف مع الاستعلام الجزئي وحيد الصف.
- استخدم المعاملات متعددة الصفوف مع الاستعلامات الجزئية متعددة الصفوف.

Types of subqueries أنماط الاستفسارات الجزئية

- Single-row subquery: query that return only one row from the inner SELECT statement.
- Multiple-row subquery: query that return more than one row from the inner SELECT statement.
- Multiple -column subquery: query that return more than one column from the inner SELECT statement.

- الاستفسار الجزئي الوحيد الصف : هو استفسار يعيد صف واحد فقط من عبارة SELECT الداخلية .
- الاستفسار الجزئي متعدد الصفوف : هو استفسار يعيد أكثر من صف من عبارة SELECT الداخلية .
- الاستفسار الجزئي متعدد الأعمدة : هو استفسار أكثر من صف من عبارة SELECT الداخلية .

Single-row subqueries: الاستفسار الجزئي وحيد الصف

Use single-row comparison operators

Operator	Meaning المعنى
=	Equal to يساوي
>	Greater than أكبر من
>=	Greater than or equal to أكبر أو يساوي
<	Less than أقل من
<=	Less than or equal to أقل أو يساوي
< >	Not equal to لا يساوي

مثال : قم بعرض الموظفين الذين يطابق عملهم عمل الموظف رقم ٧٣٦٩

```
SQL> Select   ename, job
        From      emp
        WHERE  job =
                  (select   job
                       From   emp
                       WHERE  empno = 7369 ) ;
```

CLERK

ENAME	JOB
SMITH	CLERK
ADAMS	CLERK
JAMES	CLERK
MILLER	CLERK

مثال : قم بعرض الموظفين الذين يطابق عملهم عمل الموظف رقم ٧٣٦٩ و رواتبهم تساوي راتب الموظف رقم ٧٨٧٦

```
SQL> Select   ename, job
        From      emp
        WHERE  job =
                  (select   job
                       From   emp
                       WHERE  empno = 7369 )
        AND      SAL  >
                  (select   sal
                       From   emp
                       WHERE  empno = 7876 ) ;
```

CLERK

1100

ENAME	JOB
MILLER	CLERK

Using Group Functions in a subquery — استخدام توابع التجميع ضمن الاستفسار الجزئي

مثال :

```
SQL> Select   ename, job, sal
        From      emp
        WHERE  sal =
                  (select   MIN(sal)
                    FROM       emp  );
```

800

ENAME	JOB	SAL
SMITH	CLERK	800

HAVING clause with subqueries — استخدام عبارة HAVING مع الاستفسار الجزئي

- The Oracle Server executes subqueries first.
- The Oracle Server returns results into the HAVING clause of the main query.

- يقوم مزود أوراكل بإنجاز الاستفسارات الجزئية أو الداخلية أولاً.
- يعيد مزود أوراكل النتائج إلى عبارة HAVING التابعة للاستفسار الرئيسي (الخارجي).

مثال :

```
SQL> Select   deptno, MIN( sal)
        From      emp
       GROUP BY  deptno
        HAVING  MIN(sal) >
                  (select   MIN(sal)
                    FROM       emp
                    WHERE  deptno = 20  );
```

800

DEPTNO	MIN(SAL)
١٣٠٠	١٠
٩٥٠	٣٠

مثال : أوجد العمل الموافق لأقل معدل راتب : Find the job with the lowest average salary

```
SQL> Select   job,  AVG( sal)
        From      emp
       GROUP BY  job
        HAVING  AVG(sal) =
                  (select   MIN(AVG(sal))
                    FROM       emp
                    GROUP BY   job  );
```

١٠٣٧,٥

JOB	AVG(SAL)
CLERK	1037.5

مثال خاطئ : الخطأ في المثال التالي أننا استخدمنا معامل وحيد الصف مع استفسار جزئي متعدد الصفوف هذا يعني أن الاستفسار الداخلي سوف يعيد أكثر من صف :

```
SQL> Select   empno,  ename
     From     emp
     WHERE  sal =(select   MIN(sal)   FROM      emp
                                      GROUP BY  deptno );
```

ERROR
خطأ

Multiple-Row Subqueries الاستفسارات الجزئية متعددة الصفوف (التي تعيد أكثر من صف)

Operator	Meaning المعنى
IN	Equal to any member in the list مساوٍ لأي قيمة ضمن القائمة
ANY	Compare value to each value returned by the subquery يقارن القيمة مع أي قيمة من القيم المعادة من قبل الاستفسار الجزئي
ALL	Compare value to every value returned by the subquery يقارن القيمة مع كل قيمة معادة من قبل الاستفسار الجزئي

Use Multiple-row comparison operators قم باستخدام معاملات المقارنة متعددة الصفوف

Subqueries that return more than one row are called Multiple-row subqueries. You use a multiple-row operator instead of single row operator. With a multiple-row subquery. The multiple-row operator expects one or more value.

تسمى الاستفسارات الجزئية التي تعيد أكثر من صف بالاستفسارات متعددة الصفوف. تستخدم المعاملات متعددة الصفوف بالإضافة إلى المعاملات وحيدة الصف مع الاستفسارات متعددة الصوف. المعاملات متعددة الصفوف تتوقع قيمة أو أكثر.

يقوم المعامل IN بالمقارنة مع كل قيمة من القيم المعادة من الاستفسار الجزئي ويعيد كل نتيجة على حدة .

Using IN operator in Multiple-Row Subqueries استخدام المعامل IN مع الاستفسارات الجزئية متعددة الصفوف

يقوم المعامل IN بالمقارنة مع كل قيمة من القيم المعادة من الاستفسار الجزئي ويعيد كل نتيجة على حدة

مثال : أوجد العامل الذي يحصل على أقل راتب في كل قسم:

```
SQL> Select   ename,  sal, deptno
     From     emp
     WHERE    sal  IN
                 (select   MIN (sal)
                   FROM      emp
                  GROUP BY   deptno  );
```

(800, 950, 1300)

ENAME	SAL	DEPTNO
SMITH	800	20
JAMES	950	30
MILLER	1300	10

Using ANY operator in Multiple-Row Subqueries استخدام المعامل ANY مع الاستفسارات الجزئية متعددة الصفوف

يقوم المعامل ANY بالمقارنة مع كل قيمة من القيم المعادة من الاستفسار الجزئي ويعيد كل نتيجة على حدة

مثال :

```
SQL> Select   empno, ename,  job
     From     emp
     WHERE    sal  < ANY
                 (select      sal
                  FROM      emp
                  WHERE     job = 'CLERK'  )
     AND    job <> 'clerk' ;
```

(800, 950, 1300, 1100)

EMPNO	ENAME	JOB
٧٥٢١	WARD	SALESMAN
٧٦٥٤	MARTIN	SALESMAN

Using ALL operator in Multiple-Row Subqueries استخدام المعامل ALL مع الاستفسارات الجزئية متعددة الصفوف

يقوم المعامل ALL بالمقارنة مع كل القيم المعادة من الاستفسار الجزئي .

مثال : قم بعرض رقم الموظف واسمه و عمله للموظفين الذين تكون رواتبهم أكبر من معدل الرواتب لكل قسم

```
SQL> Select   empno, ename,  job
     From     emp
     WHERE    sal  > ALL
                 (select      AVG( sal)
                  FROM      emp
              GROUP BY     deptno );
```

(1566.6667, 2175, 2916.6667)

EMPNO	ENAME	JOB
٧٥٦٦	JONES	MANAGER
٧٧٨٨	SCOTT	ANALYST
٧٨٣٩	KING	PRESIDENT
٧٩٠٢	FORD	ANALYST

الفرق بين المعاملات : مع المعامل IN يقوم الاستفسار الرئيسي بالمقارنة مع كل قيمة من القيم المعادة ويعيد النتائج المطابقة لكل قيمة من قيم القائمة التي يعيدها الاستفسار الجزئي أما مع المعامل ANY فيقوم الاستفسار الرئيسي بالمقارنة مع كل قيمة من القيم المعادة على حدة ويعيد النتائج بشكل منفصل لكل قيمة من القيم التي يعيدها الاستفسار الجزئي أما مع المعامل ALL فيقوم الاستفسار الرئيسي بالمقارنة مع كل القيم المعادة مجتمعة ويعيد النتائج التي تحقق الشرط على جميع القيم التي يعيدها الاستفسار الجزئي .

Multiple-Column Subqueries — الاستفسارات الجزئية متعددة الأعمدة

So far you have written single-row subqueries and multiple-row subqueries where only one column was compared in the WHERE clause or HAVING clause of the SELECT statement. If you want to compare two or more columns. You must write a compare WHERE clause using logical operators. Multiple-column subqueries enable you to combine duplicate WHERE conditions into a single WHERE clause.

لقد قمت سابقاً بكتابة استفسارات جزئية وحيدة الصف واستفسارات جزئية متعددة الصفوف والتي يكون فيها عمود واحد يقارن ضمن عبارة WHERE أو ضمن عبارة HAVING التابعة لعبارة SELECT . إذا كنت تريد مقارنة عمودين أو أكثر عليك كتابة عبارة المقارنة الخاصة بـ WHERE باستخدام المعاملات المنطقية. تمكنك الاستفسارات الجزئية متعددة الأعمدة من ضم شروط متعددة ضمن عبارة WHERE واحدة.

Example: Display the order number, product number and quantity of any item in which the product number and quantity match both the product number and quantity of an item in order 605.

يقوم المثال التالي بعرض رقم الترتيب ورقم المنتج والكمية لكل عنصر يكون فيه رقم المنتج والكمية مماثلين (كلاهما) لرقم المنتج والكمية الموافقة للعنصر ذو الترتيب ٦٠٥ .

مثال :

```
SQL> Select   ordid, prodid,  qty
        From      item
        WHERE     (prodid, qty ) IN
                        (select      prodid, qty
                         FROM      item
                         WHERE     ordid = 605  )
        AND     ordid <> 605 ;
```

ORDID	PRODID	QTY
٦١٧	100861	100
٦١٧	100870	500
٦١٦	102130	10

هذا يعني أنه توجد ثلاث عناصر بترتيب آخر تحتوي على نفس رقم المنتج والكمية الموجودة ضمن الترتيب رقم ٦٠٥.

Column Comparisons — مقارنات العمود

يوجد نوعان من المقارنة في حالتنا هذه المقارنة الذكية للزوج (PAIRWISE) كما في المثال السابق و التي تكون كما في الشكل :

Nonpairwise

PRODID	QTY
100863	100
100861	100
102130	10
100890	5
100870	500
101860	50

Pairwise

PRODID	QTY
100863	100
100861	100
102130	10
100890	5
100870	500
101860	50

والمقارنة الثانية تسمى المقارنة الغير ذكية للزوج (NONPAIRWISE)كما في الشكل السابق والمثال التالي :

Example: Display the order number, product number and quantity of any item in which the product number and quantity match any product number and any quantity of an item in order 605.

يقوم المثال التالي بعرض رقم الترتيب ورقم المنتج والكمية لكل عنصر يكون فيه رقم المنتج والكمية مماثلين لأي رقم منتج أي كمية موافقة للعنصر ذو الترتيب ٦٠٥ .

مثال :

```
SQL> Select    ordid, prodid,  qty
        From      item
        WHERE     Prodid  IN  (select     prodid
                                FROM      item
                                WHERE     ordid = 605  )
        AND     qty   IN  (select     qty
                                FROM      item
                                WHERE     ordid = 605  )
        AND     ordid <> 605 ;
```

هذا يعني أنه يوجد ١٦ عنصر بترتيب آخر مماثلين لأي رقم منتج أو أي كمية موجودة ضمن الترتيب رقم ٦٠٥.

ORDID	PRODID	QTY
٦٠٩	1008٧٠	5
٦١٦	100861	10
٦١٦	10٢١٣٠	10
٦٢١	100861	10
٦١٨	1008٧٠	10
٦١٨	100861	50
٦١٦	1008٧٠	50
٦١٧	100861	100
٦١٩	10٢١٣٠	100
٦١٥	100870	100
٦١٧	10٠٨٦٠	100
٦٢١	1008٧٠	100
٦١٧	10٢١٣٠	100

Null Values in a Subquery — القيم الفارغة في الاستفسار الجزئي

Return Nulls in the Resulting Set of a Subquery إعادة القيم الفارغة ضمن مجموعة النتائج التي يعيدها الاستفسار الجزئي

مثال :

```
SQL> Select    employee.ename
        From      emp  employee
        WHERE     employee.empno    NOT IN
                        (select     manager.mgr
                          FROM      emp manager );
```

No rows selected

لا يعيد المثال السابق أي صف وذلك لأن أحد القيم التي يعيدها الاستفسار الجزئي أو الداخلي هي قيمة فارغة NULL والسبب في ذلك أن أي نتيجة مقارنة مع القيمة الفارغة تعطينا قيمة فارغة NULL. ولذلك مع أي استفسار جزئي قد ينتج قيم فارغة لا تستخدم معامل المقارنة NOT IN لأنه يكافئ ALL=! و لكن باستخدام المعامل IN لن يكون هناك مشكلة لأن هذا المعامل يكافئ ANY= كما في المثال التالي:

مثال :

```
SQL> Select    employee.ename
        From      emp  employee
        WHERE     employee.empno    IN
                        (select     manager.mgr
                          FROM      emp manager );
```

ENAME
JONES
BLAKE
CLARK
SCOTT
KING
FORD

Using a Subquery in the FROM Clause — استخدام الاستفسار الجزئي ضمن عبارة FROM

مثال :

```
SQL> Select    a.ename, a.sal ,a.deptno , b.salavg
        From      emp  a ,( SELECT  deptno , avg(sal) salavg
                            FROM   emp
                            GROUP BY  deptno  )  b
        WHERE     a.demptno = b.deptno
           AND   a.sal > b.salavg ;
```

ENAME	SAL	DEPTNO	SALAVG
KING	5000	10	2916.66667
FORD	3000	20	2175
SCOTT	3000	20	2175
JONES	2975	20	2175
ALLEN	1600	30	1566.66667
BLAKE	2850	30	1566.66667

التقارير التفاعلية Interactive Reports

المتحولات البديلة (متحولات الإبدال) Substitution Variables

- Use SQL*Plus substitution variables to temporarily store values.
- تستخدم متحولات التبديل من أجل التخزين المؤقت للقيم.
  - Single ampersand (&). العلامة
  - Double ampersand (&&).العلامة المزدوجة
  - DEFINE and ACCEPT commands. الأوامر
- Pass variable values between SQL statements. تمرير قيم المتحولات بين عبارات الـ SQL
- Dynamically alter headers and footers. التغيير الديناميكي للعناوين(الرؤوس) والذيول (الهوامش السفلية)

Using the & substitution variable — استخدام المتحول البديل &

Use the variable prefixed with an ampersand (**&**) to prompt the user for a value.

قم باستخدام المتحول البديل مع وضع العلامة **&** في بدايته وذلك ليقوم بدعوة المستخدم لإدخال القيمة .

Notation	Description
&user_variable	Indicates a variable in a SQL statement; if the variable does not exist. SQL*Plus prompts the user for a value (SQL*Plus discards a new variable once it is used) يدل على المتحول في عبارة SQL .إذا لم يكن المتحول مدخلاً تقوم SQL بحث المستخدم على إدخال قيمة المتحول (تقوم SQL بالاستغناء عن القيمة المدخلة عند استخدامها مرة واحدة

مثال :

```
SQL> Select    empno, ename ,sal , deptno
       From      emp
       WHERE     empno = &employee_num;
```

```
Enter value for employee_num : 7367

EMPNO    ENAME     SAL     DEPTNO
----------  ----------  --------  ----------
 7369      SMITH     800      20
```

ويكون الخرج بالشكل السابق حيث يطالبنا بإدخال قيمة المتحول البديل نقوم بإدخال أحد أرقام الموظفين وليكن ٧٣٦٧ .

Using the SET VERIFY Command — استخدام وتفعيل أمر التحقق

Toggling the display of the text of a command before and after SQL*Plus replaces substitution variables with values.

يقوم هذا الأمر بالتأكيد على عرض النص الخاص بالأمر قبل وبعد أن تقوم SQL بتبديل متحولات البدل بقيمها.

مثال :

```
SQL> SET VERIFY ON
SQL> Select    empno, ename ,sal , deptno
       From      emp
       WHERE     empno = &employee_num;
```

```
Enter value for employee_num : 7367
old   3: where empno = &emplyee_num
new  3: where empno = 7369
EMPNO    ENAME     SAL     DEPTNO
----------  ----------  --------  ----------
 7369      SMITH     800      20
```

بمعنى آخر يقوم هذا الأمر بإعطاء تقرير عن اسم المتحول قبل التغيير وقيمته بعد التغيير. وعندما نريد إيقاف مجموعة التأكيد نكتب التعليمة SET VERIFY OFF .

Character and Data Values with Substitution Variables المحارف وقيم البيانات مع المتحولات البديلة

Use single quotation marks for data and character values.

قم باستخدام فواصل علوية مع البيانات والقيم الحرفية (المحارف).

مثال :

```
SQL> Select    ename , deptno, sal*12
       From      emp
     WHERE     job = '&job_title' ;
```

```
Enter value for job_title: ANALYST

ENAME        DEPTNO     SAL*12
----------   ----------  ----------
SCOTT          20         36000
FORD           20         36000
```

Specifying Column Names, Expressions, and Text at Runtime

أسماء أعمدة مختارة ، تعابير مختارة ، ونصوص مختارة في وضع التنفيذ

Use substitution variables to supplement the following:

قم باستخدام متحولات البدل لاستكمال ما يلي :

- WHERE condition
- ORDER BY clause
- Column expression.
- Table name
- Entire select statement

شرط WHERE وعبارة ORDER BY و تعبير العمود و اسم الجدول و عبارة SELECT تامة.

Not only can you use the substitution variables in the where clause of a SQL statement. But also these variables can be used to substitute column names, expressions, or text.

لا تستخدم متحولات البدل فقط ضمن عبارة WHERE الخاصة بعبارة SQL ولكن هذه المتحولات تستخدم لاستبدال أسماء الأعمدة و التعابير والنص.

مثال :

```
SQL> Select    empno, &column_name
       From      emp
     WHERE     &condition  ;
```

```
Enter value for column_name: job
Enter value for condition: deptno = 10
EMPNO          JOB
-----------    ---------
7782           MANAGER
7839           PRESIDENT
7934           CLERK
```

مثال :

```
SQL> Select    empno, ename, job, &column_name
       From      emp
     WHERE     &condition
     ORDER BY   &order_column ;
```

```
Enter value for column_name: sal
Enter value for condition: sal>=3000
Enter value for order_column: ename
EMPNO   ENAME     JOB          SAL
------  ----------  ---------  -------
 7902   FORD      ANALYST      3000
 7839   KING      PRESIDENT    5000
 7788   SCOTT     ANALYST      3000
```

Using the && substitution variable — استخدام المتحول البديل &&

Use the double-ampersand (&&) if you want to reuse the variable value without prompting the user each time.

قم باستخدام العلامة المزدوجة (&&) مع المتحول البديل في حالة أردت إعادة استخدام قيمة المتحول بدون إبلاغ المستخدم في كل مرة.

مثال :

```
SQL> Select   empno, ename, job, &&column_name
        From     emp
        ORDER BY  & column_name ;
```

```
Enter value for column_name: deptno

EMPNO      ENAME       DEPTNO
------     ----------  ----------
 7782      CLARK        10
 7839      KING         10
 7934      MILLER       10
 7369      SMITH        20
 7876      ADAMS        20
 7902      FORD         20
……
14 rows selected
```

في هذه الحالة يقوم المتحول البديل بحفظ القيمة المدخلة ضمنه من أول تنفيذ

Defining User Variables — تعريف متحولات خاصة بالمستخدم

Command الأمر	Description الوصف
DEFINE variable - value	Creates a CHAR datatype user variable and assigns a value to it. إنشاء متحول حرفي خاص با لمستخدم وتحديد قيمة له
DEFINE variable	Display the variable, its value, and its datatype عرض المتحول وقيمته ونوع بياناته
DEFINE	Display all user variables with value and datatype عرض جميع متحولات المستخدم مع قيمها وأنواعها
ACCEPT (see syntax on next slide)	Reads the line of user input and store it in a variable قراءة السطر المدخل من المستخدم وتخزينه ضمن المتحول

The DEFINECommand — الأمرDEFINE

Variables remains defined until you either:

المتحولات تبقى معرفة حتى تقوم بأحد الأمرين :

-Use the UNDEFINE command to clear it.

- استخدام الأمر UNDEFINE لمسحها .

- Exit SQL*Plus.

- الخروج من بيئة SQL*Plus .

To define variables for every session modify your **login.sql** file so that the variables are created at startup.

لتبقى المتحولات معرفة في كل مرة تريد فيها الدخول قم بتعديل الملف **Login.sql** عندها تتولد المتحولات تلقائياً عند الإقلاع.

مثال : إنشاء متحول يحمل اسم الشقة

Create a variable to hold the department name

```
SQL> DEFINE   deptname = sales
SQL> DEFINE   deptname
```

```
DEFINE  DEPTNAME       = "sales" (CHAR)
```

Use the variable as you would any other variable

مثال : قم باستخدام المتحول كأي متحول آخر

```
SQL> Select  *
        FROM     dept
     Where dname = UPPER( '&deptname') ;
```

DEPTNO	DNAME	LOC
٣٠	SALES	CHICAGO

To erase the variable you use the UNDEFINE command

لمحو المتحول أو إزالته نستخدم الأمر UNDEFINE

```
SQL> UNDEFINE    deptname
SQL> DEFINE    deptname
```

Symbol deptname is UNDEFINED

ACCEPT الأمر — The ACCEPT Command

الشكل العام:

ACCEPT variable **[**datatype**]** **[FORMAT** format**]**
[PROMPT text**]** **[HIDE]**

variable is the name of the variable that stores a value (If it does not exist SQL*Plus creates it)

variable اسم المتحول الذي يقوم بتخزين القيمة (إذا لم تقم بإنشائه تقوم SQL*Plus بإنشائه تلقائياً)

datatype is NUMBER, CHAR, or DATE (CHAR has a maximum length limit of 240 bytes. DATE checks against a format model, and the datatype is CHAR)

datatype نوع البيانات عددية أو حرفية أو تاريخ (البيانات الحرفية تأخذ طول أعظمي ٢٤٠ بايت، والتاريخ يأخذ الشكل المطبق عليه ضمن format ونوع بياناته حرفية)

FOR[MAT] ***format*** specifies the format model—for example, A10 or 9.999

PROMPT ***text*** displays the text before the user can enter the value.

PROMPT *text* عرض النص الذي يقوم بإخبار المستخدم بإدخال القيمة قبل إدخالها.

HIDE suppresses what the user enters--- for example, a password.

HIDE يقوم بإخفاء القيمة المدخلة من قبل المستخدم يستخدم على سبيل المثال لكلمة المرور.

- ACCEPT: Read user input and store it in a variable.
- Creates a customized prompt when accepting user input.
- Explicitly defines a NUMBER or DATE datatype variable.
- Hides user input for security reasons.

- يقوم هذا الأمر بقراءة القيمة التي يدخلها المستخدم ثم يقوم بتخزينها في المتحول.
- يستخدم لوضع عبارة اختيارية من أجل إخبار المستخدم بعملية الإدخال.
- يستخدم للتحديد الصريح لنوع بيانات المتحول إن كان رقم أو تاريخ.
- و يستخدم لإخفاء القيم المدخلة من قبل المستخدم لأسباب أمنية (للحفاظ على السرية).

مثال :

```
SQL>ACCEPT dept PROMPT ' Provide the department name : '
        Select    *
        From      dept
WHERE   dname = UPPER( '&dept') ;
```

Provide the department name : **Sales**

DEPTNO	DNAME	LOC
30	SALES	CHICAGO

Customizing the SQL*Plus Environment — تخصيص بيئة الاس كيو ال بلس

الشكل العام: **SET** system_variable value

system_variable is a variable that controls one aspect of the session environment .
value is a value for the system variable .

system_variable متحول يتحكم بمظهر واحد من مظاهر بيئة الجلسة الحالية.
Value قيمة متحول النظام.

- باستخدام أمر SET يمكنك التحكم بإعدادات الجلسة الحالية .
- يمكن التأكد من الإعدادات باستخدام أمر التأكد SHOW .
- يمكنك استخدام الأمر SHOW ALL لرؤية كافة متحولات النظام وأوضاعها.

مثال :

```
SQL> SET  ECHO ON
SQL> SHOW  ECHO
```

echo ON

حيث Echo مسؤول عن التكرار

SET Command Variables — متحولات أمر التفعيل (التحديد) SET

The underscore values indicate default values — الخط السفلي في الجدول موضوع تحت القيم الافتراضية

SET Variable and Value	الوصف Description
ARRAYSIZE {20|n}	Set the database data fetch size تحديد حجم البيانات المستخرجة من قاعدة البيانات
CLOSEP {_|text}	Sets text to be printed between columns (The default is single space) ضبط النص الذي نريد طباعته بين الأعمدة (الافتراضي فراغ وحيد)
FEED[BACK] {6 |n |OFF|ON}	Display a number of records returned by a query when the query selects at least n records. Example (13 rows selected.)في نهاية نتيجة الاستعلام يعرض عدد السجلات المعادة من الاستعلام عندما يقوم الاستعلام باختيار n سجل كحد أدنى
HEA[DING] {OFF|ON}	مسؤول عن إظهار وإخفاء أسماء الأعمدة في نتيجة الاستعلام
LIN[ESIZE] {80| n }	ضبط وتحديد عدد الأحرف الأعظمي لكل سطر حتى n حرف وبذلك يمكن إظهار الجدول كاملاً إذا كانت السجلات طويلة أو عدد الأعمدة كبير مثلاً : SET LIN 100
LONG {80| n }	تحديد الطول الأعظمي لعرض قيم البيانات الطويلة LONG
PAGES[IZE] {24| n }	تحديد عدد الأسطر الأعظمي في كل صفحة من صفحات الخرج بذلك يمكن تحديد عدد صفوف معينة لكل جزء من الخرج أي يمكن تجزيء الناتج إلى جداول بدل من جدول
PAU[SE] {OFF|ON| text}	Allow you to control scrolling of your terminal (You must press [Return] after seeing each pause) تمكنك من التحكم بتحريك الطرف السفلي
TERM[OUT] {OFF|ON}	Determines whether output is displayed on screen يحدد فيما إذا كان الخرج قد ظهر على الشاشة
ECHO {OFF|ON}	مسؤول عن التكرار

- The *login.sql* file contains standard set and other SQL*Plus commands that are implemented at login.
- You can modify *login.sql* to contain additional set commands .
- يحتوي الملف login.sql الأوامر القياسية بالإضافة إلى أوامر أخرى من SQL*Plus والتي تنفذ عند الدخول .
- بإمكانك القيام بتعديل ملف الدخول login.sql ليحوي أوامر إضافية.

SQL*Plus Format Commands — أوامر الشكل في الـ SQL*Plus

Command	الوصف Description
COL[UMN] [*column option]*	Control column formats التحكم بتصميمات العمود بإضافة خيار كما في الفقرة التالية
TTI[TLE] [text|OFF|ON]	Specifies a header to appear at the top of each page تخصيص عنوان يظهر في أعلى كل صفحة
BTI[TLE] [text|OFF|ON]	Specifies a footer to appear at the bottom of each page of the report تخصيص هامش سفلي يظهر في أسفل كل صفحة
BRE[AK] [ON report element]	Suppresses duplicate value and sections rows of data with line feeds طمس وإخفاء القيم المضاعفة ومقاطع من الصفوف بسطر فارغ

تكتب هذه الأوامر مع معاملاتها مباشرة بدون سابق كما يلي :

```
SQL>TTI  "Students Table"
SQL>BTI  "END Report"
```

The COLUMN Command — أوامر العمود

Controls display of a column — تحكمات بطريقة عرض العمود

COL[UMN] [{column|alias} [option]]

COLUMN Command Options — جدول خيارات أمر العمود

Option الخيار	Description الوصف
CLE[AR]	مسح أي إعداد لشكل العمود مثال: **COL** ename **CLE**
FOR[MAT] *format*	تغيير طريقة عرض بيانات العمود
HEA[DING] *text*	تغيير طريقة عرض عنوان العمود (رأس العمود) مثال: **COL** ename **HEA** employee
JUS[TIFY] {*align*}	يجعل اسم العمود (الرأس) وليس البيانات يكون في اليسار أو الوسط أو اليمين حسب الإتجاه مثال: **COL** job **JUS CENTER**
NOPRI[NT]	إخفاء أعمدة ضمن التقرير مثال : **COL** sal **NOPRI**
NUL[L] *text*	اختيار قيمة لتعبئتها في التقرير مكان القيم الفارغة مثال : **COL** comm **NUL** 0
PRI[NT]	إظهار العمود المخفي مثال : COL sal **PRI**
TRU[NCATED]	يقوم بقطع السلسلة في نهاية السطر الأول من العرض
WRA[PPED]	التفاف نهاية السلسلة إلى السطر التالي (السطور ملتفة)

أمثلة :

إنشاء رؤوس (عناوين) للأعمدة :

```
SQL>COLUMN ename  HEADING 'Employee|Name'  FORMAT  A15
SQL>COLUMN sal  JUSTIFY  LEFT  FORMAT  $99,990.00
SQL>COLUMN mgr  FORMAT  999999999  NULL  'No manager'
SQL>SELECT ename,  mgr , sal  FROM  emp ;
```

وعندها يصبح ناتج ما يلي :

```
Employee
Name            MGR          SAL
---------------  ------------  ----------------
SMITH            7902          $800.00
ALLEN            7698          $1,600.00
WARD             7698          $1,250.00
JONES            7839          $2,975.00
MARTIN           7698          $1,250.00
BLAKE            7839          $2,850.00
CLARK            7839          $2,450.00
SCOTT            7566          $3,000.00
KING        No manager         $5,000.00
TURNER           7698          $1,500.00
ADAMS            7788          $1,100.00
JAMES            7698          $950.00
FORD             7566          $3,000.00
MILLER           7782          $1,300.00
```

COLUMN Format Models نماذج لشكل العمود

Element	Description	Example	Result
An	يقوم بإعدادالعرض حسب قيمة n كما في المثال السابقA15	N/A	N/A
9	أقرب عدد صحيح	999999	1234
0	يضع صفر يساري إجباري	099999	01234
$	علامة دولار $	$9999	$1234
L	رمز للعملة المحلية مهما يكن	L9999	L1234
.	الفاصلة العشرية	9999.99	1234.00
,	فاصلة الآلاف	9.999	1,234

أمثلة :

❖ عرض الإعدادات الحالية للعمود ENAME

Display the current setting for the ENAME column

```
SQL>COLUMN ename
```

```
COLUMN   ename ON
HEADING  'Employee'
FORMAT   A15
```

❖ عرض الإعدادات الحالية لكافة الأعمدة

Display the current setting for all columns

```
SQL>COLUMN
```

❖ مسح الإعدادات الحالية للعمود ENAME

Clear setting for the ENAME column

```
SQL>COLUMN ename CLEAR
```

❖ مسح الإعدادات الحالية لكافة الأعمدة

Clear setting for all columns

```
SQL> CLEAR   COLUMN
SQL> CLE   COL
```

أو اختصاراً

❖ إخفاء القيم المضاعفة

Suppresses duplicate value

```
SQL>BREAK ON ename ON job
```

❖ لاقتطاع صفوف أو وضع صفوف فارغة بين سجلات التقرير

To section out rows at break value

```
SQL>BREAK ON ename SKIP4 ON job SKIP2
```

❖ مسح كل إعدادات تعليمة الـ BREAK

Clear all BREAK setting

```
SQL>CLEAR BREAK
```

❖ وضع عنوان للتقرير

Set the report header

```
SQL>TTITLE 'Salary|Report'
```

❖ وضع هامش سفلي للتقرير

Set the report footer

```
SQL>BTITLE ' Confidential '
```

Creating a Script File to Run a Report إنشاء ملف نصي لحفظ وتشغيل التقرير

A. Create the SQL SELECT statement at the SQL prompt. Ensure that the data required for the report is accurate before you save the statement to a file and apply formatting commands. Ensure that the relevant ORDER BY clause is included if you intend to use breaks.

B. Save the SELECT statement to a script file.

C. Edit the script file to enter the SQL*Plus commands.

D. Add the required formatting commands before the SELECT statement. Be certain not to place SQL*Plus commands within the SELECT statement.

E. Verify that the SELECT statement is followed by a run character. Either a semicolon (**;**) or a slash (/).

F. Add the format-clearing SQL*Plus commands after the run character. As an alternative, you can call a reset file that contains all the format-clearing commands.

G. Save the script file with your changes.

H. In SQL*Plus, run the scripting file by entering START *filename* or *@filename*. This command is required to read and execute the script file.

A. قم بإنشاء عبارة SQL ضمن ملقن SQL. تأكد أن البيانات المطلوبة للتقرير دقيقة قبل حفظ التعليمة إلى ملف وتفعيل أوامر التنسيقات . تأكد من وجود عبارة ORDER BY مناسبة إذا كنت تنوي استخدام أوامر الكسر BREAK .

B. قم بحفظ تعليمة SELECT التي قمت بإنشائها ضمن ملف نصي.

C. قم بتحرير الملف النصي لإدخال أوامر SQL*Plus .

D. قم بإضافة أوامر التنسيقات قبل عبارة SELECT . كن واثقاً من عدم وضع أوامر SQL*Plus ضمن عبارة SELECT .

E. تحقق من أن عبارة SELECT متبوعة برمز تشغيل وهو أياً من الفاصلة المنقوطة (**;**) أو الخط المائل بالشكل (/).

F. قم بإضافة أوامر مسح التنسيق بعد رمز التشغيل، كأحد الخيارات غير أنه يمكنك استدعاء ملف إعادة الإفتراضيات reset الذي يحتوي بدوره على كافة أوامر مسح التنسيق .

G. إحفظ الملف النصي مع التغييرات.

H. ضمن SQL*Plus قم بتشغيل الملف النصي باستخدام الأمر START مع اسم الملف أو اختصاراً *@filename* هذا الأمر مطلوب لقراءة وتنفيذ الملف النصي.

مثال : قم بإنشاء تقرير يظهر نوع العمل واس وراتب كل موظف راتبه أقل من ٣٠٠٠$ قم بوضع عنوان للتقرير مكون من سطرين Employee Report وفي وسط وأسفل الصفحة أكتب Confidential قم بوضع عنوان آخر للعمود Job باسم Job Category يكون مقسوماً إلى سطرين . ثم قم بوضع عنوان آخر للعمود Ename باسم Employee .ثم قم بوضع عنوان آخر لعمود الراتب Sal باسم Salary وقم بتنسيقه إلى الشكل $2,500.0 :

الملف النصي Script File

```
SET PAGESIZE 37
SET LINESIZE 60
SET FEEDBACK OFF
TTITLE   'Employee|Report'
BTITLE   'Confidential'
BREAK ON  job
COLUMN  job      HEADING  'Job|Category'  FORMAT  A15
COLUMN  ename  HEADING  'Employee'    FORMAT  A15
COLUMN  sal       HEADING  'Salary'      FORMAT  $99,999.99
REM  **  Insert SELECT statement
SELECT       job,  ename,  sal
FROM          emp
WHERE       sal < 3000
ORDER BY    job,  ename
/
SET  FEEDBACK  ON
REM  clear all formatting commands
```

التقرير

```
Fri    Oct 12                                    page    1
                    Employee
                     Report

Job
Category          Employee          Salary
---------------   ---------------   --------------
CLERK             ADAMS             $1,100.00
                  JAMES               $950.00
                  MILLER            $1,300.00
                  SMITH               $800.00
MANAGER           BLAKE             $2,850.00
                  CLARK             $2,450.00
                  JONES             $2,975.00
SALESMAN          ALLEN             $1,600.00
                  MARTIN            $1,250.00
                  TURNER            $1,500.00
                  WARD              $1,250.00


                    Confidential
```

لغة معالجة البيانات

Data manipulation language (DML) is a core part of SQL . When you want to add, update, or delete data in the database, you execute a DML statement .
A collection of DML statements that from a logical unit of work is called a *transaction* .

لغة معالجة البيانات ويرمز لها اختصاراً DML تعتبر جزء رئيسي أو مركزي من لغة الـ SQL عندما تريد إضافة أو تعديل أو حذف بيانات من قاعدة البيانات عليك إنجاز عبارة DML .
مجموعة عبارات لغة DML من وحدة منطقية من العمل تسمى إجراء أو صفقة (معاملة تجارية).
على سبيل المثال إذا قام أحد عملاء بنك بتحويل مبلغ من الحساب المودع إلى حساب الشيكات (الجاري) هذا الإجراء ربما يكون مؤلفاً من ثلاث عمليات منفصلة . إنقاص الحساب المودع وزيادة الحساب الجاري وتسجيل عملية التحويل ضمن سجل التحويل . يجب أن يضمن مخدّم أوراكل إنجاز جميع عبارات SQL ليحافظ على الحسابات في توازن صحيح . وإذا أعاق شيء ما أحد هذه العبارات الخاصة بعملية التحويل يجب أن لا تنجز باقي العمليات .

The INSERT statement — عبارة الإدخال INSERT

Add new rows to a table by using the INSERT statement — إدخال صفوف جديدة باستخدام INSERT

الصيغة العامة:

INSERT INTO table [(column [, column….])]
Values (value **[,** value**….]) ;**

table is the name of table — *table* تعبر عن اسم الجدول
column is the name of the column in the table to populate — *column* تعبر عن اسم العمود ضمن الجدول للإدخال
value is the corresponding value for the column — *value* تعبر عن القيمة المناظرة للعمود

إرشادات : عند إدخال صف جديد عليك التقيد بما يلي :

- أدخل الصف الجديد بحيث يحتوي على قيم لكل عمود بدون استثناء .
- قم بترتيب القيم المدخلة حسب الترتيب الافتراضي للأعمدة في الجدول .
- بإمكانك ترتيب أسماء الأعمدة في الجدول بشكل اختياري لكن ترتيب القيم يجب أن يطابق ترتيب الأعمدة .
- قم بإحاطة القيم الموافقة للبيانات الحرفية والتواريخ بفواصل علوية (علامات اقتباس) مفردة .

مثال :

```
SQL> INSERT INTO  dept    (deptno, dname,  loc)
     VALUES               ( 50, 'DEVELOPMENT' , 'DETRIOT' ) ;
```

```
1 row created.
```

Inserting Special Value — إدخال قيم خاصة باستخدام التوابع

مثال :

```
SQL> INSERT INTO  emp    (empno, ename,  job, mgr, hiredate,
                          Sal, comm, deptno)
     VALUES              ( 7196, 'GREEN' , 'SALESMAN', 7782, SYSDATE,
                          2000, NULL, 10 )  ;
```

```
1 row created.
```

يقوم التابع SYSDATE في المثال السابق بإدراج قيمة التاريخ الحالي ضمن الحقل hiredate

للتحقق نضع :

```
SQL> SELECT  *  FROM  emp
     WHERE   empno = 7196 ;
```

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7196	GREEN	SALESMAN	7782	13/06/04	2000		10

Inserting Special Date Value — **إدخال قيم خاصة للتاريخ باستخدام التوابع**

مثال :

```
SQL> INSERT INTO      emp
     VALUES           ( 2296, 'ARMANO' , 'SALESMAN', 7782,
     TO_DATE( 'FEB  3,  97' ,  'MON  DD,  YY' ) , 1300, NULL, 10 ) ;
```

```
1 row created.
```

وبالتالي يمكننا استخدام توابع التواريخ ضمن تعليمة الإدخال INSERT .

Creating a Script with Customized Prompts — **إنشاء ملف نصي مع إدخال خاص**

الملف النصي Script File

```
ACCEPT   department_id  PROMPT 'Please enter the department number : '
ACCEPT   department_name  PROMPT 'Please enter the department name : '
ACCEPT   location   PROMPT 'Please enter the location : '
INSERT  INTO      dept (deptno, dname, loc)
VALUES            ( &department_id , '&department_name', '&location' ) ;
```

عند تشغيل هذا الملف يطلب منا إدخال قيم المتحولات واحد تلو الآخر ثم يقوم بإدخالها ضمن الجدول .

```
Please enter the department number : 90
Please enter the department name : PAYROL
Please enter the location : HOUSTON

1 row created.
```

The UPDATE statement — عبارة الإدخال UPDATE

Modify existing rows with the UODATE statement — تعديل صفوف موجودة باستخدام UPDATE

الصيغة العامة:

```
UPDATE     table
SET        column = value [ ,  column = value, …..]
[WHERE          condition ] ;
```

table is the name of table — *table* تعبر عن اسم الجدول

column is the name of the column in the table to populate — *column* تعبر عن اسم العمود ضمن الجدول للإدخال

value is the corresponding value for the column — *value* تعبر عن القيمة المناظرة للعمود

condition identifies the rows to be updated — *Condition* شرط يعين الصفوف التي سيتم تعديلها

- Specific row or rows are modified when you specify the WHERE clause.
- يتم تعديل صف واحد أو عدة صفوف محددة بشكل دقيق عند استخدام عبارة WHERE .

مثال :

```
SQL> UPDATE   emp
     SET      deptno = 20
     WHERE          empno = 7782 ;
```

```
1 row updated.
```

- All rows in the table are modified if you omit the WHERE clause.
- يتم تعديل كل الصفوف ضمن الجدول عند حذف عبارة WHERE .

مثال :

```
SQL> UPDATE   emp
     SET      deptno = 20 ;
```

```
14 rows updated.
```

Updating with Multiple-Column Subquery — التحديث باستخدام استفسار جزئي متعدد الأعمدة

مثال : تحديث عمل الموظف job ورقم الشقة deptno للموظف رقم ٧٦٩٨=empno لتكون مساوية للموظف رقم 7499 =empno

```
SQL> UPDATE   emp
     SET      ( job, deptno ) =
                    ( SELECT  job, deptno
                      FROM  emp
                      WHERE  empno = 7944 )
     WHERE          empno = 7698  ;
```

1 row updated.

- Multiple-column subqueries can be implemented in he SET clause of an UPDATE statement.
- الاستفسار المتعدد الأعمدة يمكن أن ينفذ ضمن عبارة STE التابعة لتعليمة UPDATE .

Updating Rows Based on Another Table — تحديث صفوف بناء على جدول ثاني

مثال :

```
SQL> UPDATE      employee
     SET  deptno =  ( SELECT   deptno
                      FROM  emp
                      WHERE  empno = 7788 )
     WHERE      job = ( SELECT  job
                      FROM     emp
                      WHERE    empno = 7788 ) ;
```

20

ANALYST

2 rows updated.

Updating Rows Integrity Constraint Error — تحديث الصفوف وخطأ قيد التكامل

مثال :

```
SQL> UPDATE      emp
     SET         deptno = ٥٥
     WHERE       deptno = 10 ;
```

رقم القسم ٥٥ غير موجود

```
UPDATE   emp
*
ERROR at line 1:
ORA-02291: integrity constraint (SCOTT.FK_DEPTNO)
violated – parent key not found
```

تم انتهاك قيد التكامل (...) مفتاح العنصر الرئيسي غير موجود

The DELETE statement — عبارة الحذف DELETE

Remove existing rows from a table by using the DELETE statement (حذف صفوف موجودة باستخدام DELETE)

الصيغة العامة:

```
DELETE   [FROM]     table
[WHERE              condition ] ;
```

table is the name of table — *table* تعبر عن اسم الجدول

condition identifies the rows to be deleted — *Condition* شرط يعين الصفوف التي سيتم حذفها

- Specific rows are deleted when you specify the WHERE clause.
- يتم حذف صفوف محددة بشكل دقيق عند استخدام عبارة WHERE .

مثال :

```
SQL> DELETE   FROM    dept
     WHERE            dname = 'DEVELOPMENT' ;
```

1 row deleted.

- All rows in the table are deleted if you omit the WHERE clause.
- يتم حذف كل الصفوف ضمن الجدول عند حذف عبارة WHERE .

مثال :

```
SQL> DELETE  FROM   dept ;
```

4 rows deleted.

مثال :

```
SQL> DELETE   FROM    emp
     WHERE          hiredate  > TO_DATE ( '01.01.97' , 'DD.MM.YY' ) ;
```

يقوم بحذف كل الموظفين الذين بدأوا بعد January 1. 1997

```
1 row deleted.
```

Deleting Rows Based on Another Table — حذف صفوف بناء على جدول ثاني

```
SQL> DELETE  FROM   emp
     WHERE       deptno  =
                 ( SELECT     deptno
                   FROM     dept
                 WHERE   dname  = 'SALES' ) ;
```

30

```
6 rows deleted.
```

Deleting Rows Integrity Constraint Error — حذف صفوف وخطأ قيد التكامل

مثال :

```
SQL> DELETE  FROM        dept
     WHERE        deptno = 10 ;
```

لا يمكنك حذف الصف الذي يحتوي على مفتاح أساسي يستخدم كمفتاح ثانوي في جدول آخر .

```
DELETE  FROM   dept
   *
ERROR at line 1:
ORA-02292: integrity constraint (SCOTT.FK_DEPTNO)
violated – child record found
```

تم انتهاك قيد التكامل (...) تم العثور على سجل عنصر فرعي

Database Transactions — إجراءات قاعدة البيانات

Consist of one of the following statements: يتألف من أحد العبارات التالية:

- DML (Data Manipulation Language)statements that make up one consistent change to the data.
- أي عدد من عبارات DML التي تقوم بتغيير واحد على البيانات أي التي يعاملها مزود أوراكل ككينونة واحدة أي كوحدة منطقية تشكل جزء واحد من العمل.
- One DDL (Data Definition Language) statement. تعليمة واحدة
- One DCL (Data Control Language) statement. تعليمة واحدة

عند تعديل قاعدة المعطيات بالإضافة أو التحديث يمكنك حفظ هذا التعديل أو عدم الحفظ وهذا مفيد عند اكتشاف أخطاء أثناء التعديل و عملية تثبيت التعديل أو التراجع تكتبان بالأوامر Commit للحفظ و Rollback للتراجع.

- تبدأ إجراءات قاعدة البيانات عندما تنفذ أول عبارة SQL قابلة للتنفيذ .
- وتنتهي مع حدوث أي مما يلي :

١. إذا تم تنفيذ أي من الأوامر COMMIT or ROLLBACK .
٢. إذا تم تنفيذ أي من تعليمات DDL or DCL والتي تعد بمثابة الحفظ الأوتوماتيكي Auto Commit .
٣. خروج المستخدم User Exits والذي يقوم بالحفظ تلقائياً .
٤. انهيار النظام أو فشل النظام .

عندما تنتهي أحد العمليات أو الإجراءات تقوم عبارة SQL التالية بشكل أوتوماتيكي ببدء الإجراء التالي .
كما وتقوم تعليمات DDL or DCL بالحفظ التلقائي ولذلك فإنها تقوم ضمنياً بإنهاء الإجراء.

Advantages of COMMIT and ROLLBACK Statements

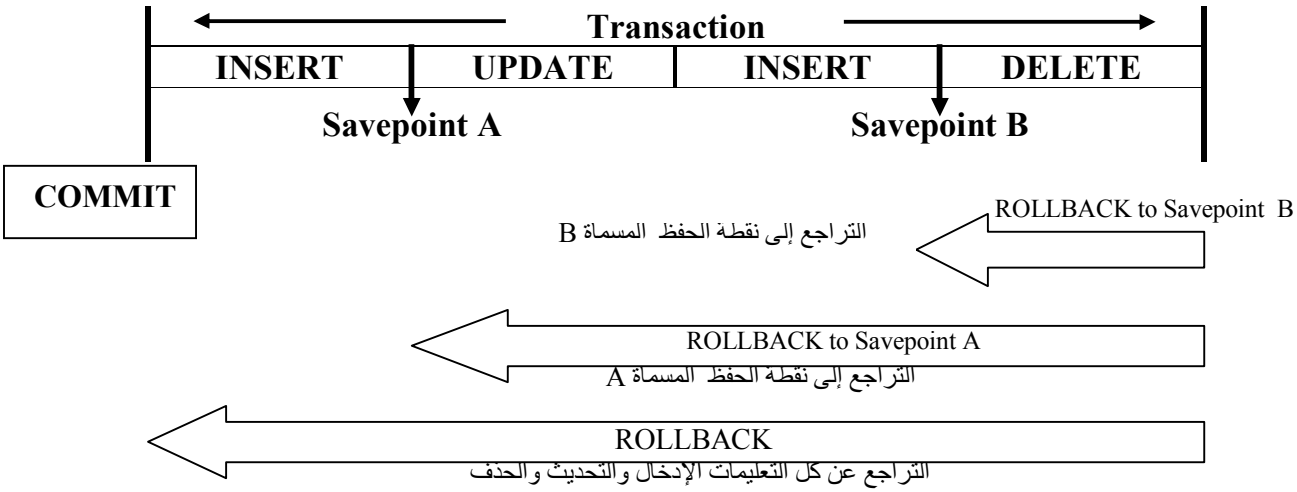
ميزات عبارتي COMMIT و ROLLBACK

- Ensure data consistency
- Preview data changes before making changes permanent
- Group logically related operations

- تضمن ثبات البيانات
- تسمح بمراجعة التغييرات قبل تثبيتها بشكل دائم
- تقوم بتجميع العمليات المنطقية

Controlling Transactions

عملية التحكم بالإجراءات



Statement	Description الوصف
COMMIT	Ends the current transaction by making all pending data changes permanent إنهاء الإجراء الحالي بجعل كل التغييرات المتعلقة بالبيانات ثابتة ودائمة(حفظ التغييرات)
SAVEPOINT name	Marks a savepoint within the current transaction تحديد نقطة حفظ خلال الإجراء الحالي
ROLLBACK [TO SAVEPOINT name]	ROLLBACK ends the current transaction by discarding all pending data changes :ROLLBACK TO SAVEPOINT name discards the savepoint and all subsequent changes التراجع عن الإجراء الحالي بإلغاء كافة التغييرات الأخيرة المتعلقة بالبيانات التراجع إلى نقطة الحفظ تعمل على تجاهل نقطة الحفظ وكافة التغييرات اللاحقة

- يحدث الحفظ الأوتوماتيكي Automatic commit مع حدوث أي مما يلي :
  ١. تنفيذ تعليمة DDL .
  ٢. تنفيذ تعليمة DCL .
  ٣. الخروج بشكل عادي من بيئة SQL*Plus بدون تنفيذ COMMIT أو ROLLBACK
- كما يحدث التراجع الأوتوماتيكي Automatic rollback بالإنهاء الغير طبيعي للـ SQL*Plus أو بانهيار النظام وفي هذه الحالة يقوم مزود أوراكل بالتراجع إلى آخر نقطة حفظ COMMIT مع المحافظة على سلامة الجداول.

ملاحظة : إن الأمر **AUTOCOMMIT** هو من تعليمات **DML** ويمكن استخدامه مع **SET** وتفعيله أو عدم تفعيله ON or OFF ولكن لا يمكنك عندها تنفيذ عملية التراجع ROLLBACK في حالة تفعيله ويكون افتراضيا على الوضع OFF .

SQL> SET AUTOCOMMIT ON

قبل تطبيق أحد عمليتي التثبيت أو التراجع COMMIT or ROLLBACK لا يستطيع أي مستخدم آخر مشاهدة نتائج عبارة DML المستخدمة من قبل المستخدم الحالي وتكون البيانات مقفلة بحيث لا يستطيع مستخدم آخر تغييرها.

أما بعد عملية التثبيت أو التراجع COMMIT or ROLLBACK يستطيع المستخدمين الآخرين مشاهدة النتائج أو التغييرات كما أن جميع نقاط الحفظ Savepoints تمحى .

Committing Data — تثبيت التغييرات على البيانات

مثال١ : إحداث تغييرات على البيانات

```
SQL> UPDATE   emp
     SET   deptno = 10
     WHERE        deptno = 7782  ;
```

```
1  row  updated .
```

تثبيت التغييرات على البيانات

```
SQL> COMMIT ;
```

```
Commit  complete
```

مثال2 : إحداث تغييرات على البيانات

```
SQL> INSERT INTO    dept  (deptno, dname, loc )
          VALUES    (50, 'ADVERTISING', 'MIAMI' ) ;
```

```
1  row  created
```

مثال2 : إحداث تغييرات على البيانات

```
SQL> UPDATE   emp
     SET   deptno = 10
     WHERE        deptno = 7782  ;
```

```
1  row  updated .
```

تثبيت التغييرات على البيانات

```
SQL> COMMIT ;
```

```
Commit  complete
```

مثال٣ : حذف بيانات جدول

```
SQL> DELETE FROM   emp ;
```

```
14  row deleted.
```

التراجع عن الحذف

```
SQL> ROLLBACK ;
```

```
Rollback  complete
```

Rolling Back Changes to a Marker — التراجع عن التغييرات حتى نقطة معلمة

- Create a marker in a current transaction by using the SAVEPOINT statement.
- Roll back to that marker by using the ROLLBACK TO SAVEPOINT statement.
- If you create a second savepoint with the same name as an earlier savepoint, the erlier savepoint is deleted

- إنشاء نقطة علامة ضمن الجلسة الحالية أو الإجراء الحالي باستخدام عبارة SAVEPOINT .
- التراجع للخلف حتى تلك النقطة باستخدام عبارة ROLLBACK TO SAVEPOINT .
- إذا قمت بإنشاء نقطة حفظ أخرى بنفس اسم الأولى فإن الأولى تلغى .

مثال :

```
SQL> UPDATE …
SQL> SAVEPOINT  update_done ;
```

```
Savepoint created .
```

```
SQL> INSERT …
SQL> ROLLBACK  TO update_done ;
```

```
Rollback complete .
```

يقوم المستخدم بعمليتي وصول إلى البيانات إما قراءة Reader أو كتابة Writer :
(SELECT statement) **or** (INSERT, UPDATE, DELETE statement)
تحتاج في هذه الحالة لقراءة متوافقة(**Read Consistency**) وثابتة للبيانات وذلك لحدوث ما يلي :
تضمن قراءة ثابتة للبيانات من قبل المستخدم الذي يقرأ والذي يكتب بنفس الوقت في قاعدة البيانات
لا يرى المستخدم القارىء البيانات التي تكون في طور التغيير .
يضمن المستخدم Writer أن التغييرات التي تحدث على قاعدة البيانات انتهت بطريقة متماسكة.
التغييرات التي تحدث من قبل Writer واحد لا تتقاطع أو تتعارض مع التغيرات التي تحدث من قبل مستخدم آخر Writer.

- The purpose of read consistency is to ensure that each user sees data as it existed at the last commit, before a DML operation started.
- الهدف من عملية القراءة المتوافقة هو ضمان أن كل مستخدم يرى البيانات كما هي موجودة عند آخر تثبيت لها وذلك قبل بدء أي عملية من عمليات لغة الـ DML.

Data Definition Language
لغة تعريف البيانات (التعامل مع الكائنات)

Database Objects - أشياء أو أغراض قاعدة البيانات	
Object الغرض	**Description الوصف**
Table الجدول	Basic unit of storage; composed of rows and columns وحدة التخزين الأساسية للبيانات مؤلفة من صفوف وأعمدة
View المنظور	Logically represents subsets of data from one or more table إجراء ثانوي تمثل (تصور) بشكل منطقي البيانات من جدول أو أكثر
Sequence المسلسل	Generates primary key values تولد قيم المفتاح الأساسي (**المسلسِل**)
Index الفهرس	Improve the performance of some queries يقوم بتحسين أداء بعض الاستفسارات
Synonym المرادف	Give alternative names to objects إعطاء أسماء بديلة للأشياء

Naming Conventions (Rules) **اصطلاحات (قوانين) التسمية**

Name database tables and columns according to the standard rules for naming any Oracle database object:
تسمية قاعدة البيانات والجداول والأعمدة خاضعة لقوانين قياسية خاصة بتسمية أغراض أوراكل كما يلي :

- Must begin with a letter and can be 1 – 30 characters long.
  يجب أن تبدأ أسماء الجداول والأعمدة بمحارف حصراً وبطول من ١ إلى ٣٠ محرف كحد أقصى.
- Must contain only A-Z , a – z , 0 – 9 , _ (underscore), $, and #. يجب أن تحوي فقط الرموز التالية
- Must not duplicate the name of another object owned by the same user.
  لا يجب أن يكون إسم الغرض التابع لنفس المستخدم مكرراً .
- Must not be an Oracle Server reserved word. يجب أن لا تكون أحد الكلمات المحجوزة لمزود أوراكل

The CREATE TABLE statement **عبارة إنشاء جدول CREATE TABLE**

بداية يجب أن يكون لديك امتياز إنشاء جدول ومساحة للتخزين ويجب أن تخصص اسم للجدول وأسماء الأعمدة وأنواع البيانات لكل منها وقياس كل عمود :

الصيغة العامة:

```
CREATE TABLE    [ schema .] table
                ( column  datatype [DEFAULT expr][ ,  … ] ) ;
```

schema is the same as the owner's name — *schema* تعبر عن نفس اسم المالك
table is the name of the table — *table* تعبر عن اسم الجدول
DEFAULT *expr* specifies a default value if a value is omitted in the INSERT statement
DEFAULT *expr* تخصيص قيمة افتراضية إذا كانت هذه القيمة قد حذفت من عبارة INSERT
column is the name of the column — *column* تعبر عن اسم العمود
datatype is the column's datatype and length — *Datatype* تحديد نوع البيانات وطولها الأعظمي

يمكن تخصيص قيمة افتراضية للعمود أثناء عملية الإدخال كما في المثال التالي :

……. Hiredate DATE DEFAULT SYSDATE , ……..

القيم المقبولة أو القانونية هي القيمة الحرفية أو التعابير أو توابع الـ SQL .
القيم الغير قانونية أو الغير مقبولة هي أسماء أعمدة أخرى أو الأعمدة الزائفة .
نوع بيانات القيمة الافتراضية يجب أن يكون مماثل لنوع بيانات العمود.

مثال :

```
SQL> CREATE  TABLE      dept
                ( deptno   NUMBER(2),
                  dname  VARCHAR2(14),
                  loc      VARCHAR2(13)   ) ;
```

Table created

Tables in the Oracle Database **الجداول في مزود أوراكل**

١. **جداول المستخدم (User Tables) :** وهي مجموعة من الجداول المنشأة والمحفوظة من قبل المستخدم وتحوي معلومات المستخدم (User Information).
٢. **قاموس المعطيات (Data Dictionary) :** وهي مجموعة من الجداول المنشأة والمحفوظة من قبل مخدم أوراكل والتي تحتوي معلومات عن البيانات (Database Information).

There are four categories of data dictionary views : each category has a distinct prefix witch reflects their intended use .

هناك أربعة من أصناف مناظر قواميس البيانات : كل صنف له بادئة واضحة تعكس الإستخدام المطلوب كما الجدول :

Prefix البادئة	Description الوصف
USER_	Contains objects owned by the user يحتوي الأغراض التي يملكها المستخدم
ALL_	يستطيع الوصول للأغراض التي منح المستخدم حقوق الوصول لها بالإضافة إلى الأغراض المملوكة من قبل المستخدم
DBA_	يعطي الإمتياز للدخول أو الوصول إلى أي غرض ضمن قاعدة البيانات
V$_	يعرض سلوك مزود قاعدة البيانات و القفل وهو متاح مبدئياً فقط لـ DBA

Querying the Data Dictionary الاستعلام عن قاموس المعطيات

- Describe tables owned by the user - وصف الجداول المملوكة للمستخدم

```
SQL> SELECT    *
     FROM      user_tables ;
```

- View distinct object types owned by the user - إظهار أنواع الأغراض التابعة للمستخدم بدون تكرار

```
SQL> SELECT    DISTINCT   object_type
     FROM      user_objects ;
```

```
OBJECT_TYPE
-----------
FUNCTION
INDEX
PROCEDURE
SEQUENCE
TABLE
```

- View tables, views, synonyms, sequences owned by the user . - إظهار الجداول والمناظير والمرادفات والمتسلسلات التابعة للمستخدم

```
SQL> SELECT    *
     FROM      user_catalog ;
```

The USER_CATALOG has a synonym called CAT. You can use this synonym instead of USER_CATALOG in SQL statement.

```
SQL> SELECT    *
     FROM      CAT ;
```

يمكن استخدام الاسم المرادف CAT بدلاً من USER_CATALOG .

```
TABLE_NAME                     TABLE_TYPE
------------------------------ ----------
BONUS                          TABLE
DEPT                           TABLE
EMP                            TABLE
MICROSOFTDTPROPERTIES          TABLE
MICROSOFTSEQDTPROPERTIES       SEQUENCE
SALGRADE                       TABLE
```

أنواع البيانات المتاحة في أوراكل DATATYPES	
Datatype النوع	Description الوصف
VARCHAR2(size)	نوع البيانات متحول حرفي متغير الطول يجب أن يكون size طول السلسلة بين حرف واحد ١على الأقل و٤٠٠٠ محرف على الأكثر
CHAR(size)	نوع البيانات متحول حرفي ثابت الطول يجب أن يكون size طول السلسلة بين حرف واحد ١على الأقل و٢٠٠٠ محرف على الأكثر
NUMBER(p , s)	متحول حرفي متغير الطول له الدقة P والمقياس S حيث Pأكبر طول لرقم قبل الفاصلة العشرية وS أكبر طول لرقم بعد الفاصلة العشرية من جهة اليمين يأخذ p مجال القيم بين ١ إلى ٣٨ أما S فيأخذ مجال القيم بين84- وبين ١٢٧
DATE	نوع البيانات تاريخ ووقت بين January 1. 4712B.C. وبين December 31.9999A.D.
LONG	نمط حرفي متغير الطول من البيانات يأخذ يصل حجمه إلى حتى ٢ جيجا بايت
CLOB	متحول حرفي يصل حجمه إلى حتى ٤ جيجا بايت
ROW	متحول ثنائي يأخذ طول محدد sizeيمكن جعله حتى ٢٠٠٠ كحد أقصى
LONG ROW	متحول ثنائي متغير الطول يصل حجمه حتى ٢ جيجا بايت
BLOB	بيانات ثنائية يصل حجمها حتى ٤ جيجا بايت
BFILE	بيانات ثنائية تخزن ضمن ملف خارجي يصل حجمها حتى ٤ جيجا بايت

Create a Table by Using a Subquery — إنشاء جدول باستخدام الاستفسار الجزئي

مثال:

```
SQL> CREATE  TABLE    dept30
 2       AS
 3       SELECT   empno, ename, sal*12  ANNSAL, hiredate
 4       FROM          emp
 5       WHERE         deptno = 30 ;
```

Table created

```
SQL> DESCRIBE    dept30
```

Name	Null?	Type
EMPNO	NOT NULL	NUMBER(4)
ENAME		VARCHAR2(10)
ANNSAL		NUMBER
HIREDATE		DATE

The ALTER TABLE statement — عبارة تعديل جدول ALTER TABLE

Use the ALTER TABLE statement to: Add a new column, modify an existing column, and define a default value for the new column.

تستخدم تعليمة ALTER TABLE لإضافة عمود جديد أو تعديل عمود موجود أو تحديد قيمة افتراضية للعمود جديد.

الصيغة العامة١:

```
ALTER TABLE    table
ADD            ( column  datatype [DEFAULT expr]
               [ ,  column  datatype].... )  ;
```

الصيغة العامة٢:

```
ALTER TABLE    table
MODIFY         ( column  datatype [DEFAULT expr]
               [ ,  column  datatype].... )  ;
```

table is the name of the table — *table* تعبر عن اسم الجدول

DEFAULT *expr* specifies a default value for a new column — تخصيص قيمة افتراضية للعمود الجديد

column is the name of the column — *column* تعبر عن اسم العمود

datatype is the datatype and length of the new column — *Datatype* تحديد نوع البيانات وطولها للعمود الجديد

Adding a Column — إضافة عمود

You use the ADD clause to add columns

نستخدم عبارة ADD لإضافة أعمدة

مثال:

```
SQL> ALTER  TABLE    dept30
 2       ADD              ( job VARCHAR2(9)) ;
```

Table altered.

يصبح العمود الجديد في نهاية الجدول كما يلي :

```
SQL> SELECT    *
 2       FROM             dept30 ;
```

EMPNO	ENAME	ANNSAL	HIREDATE	JOB
----	--------	----------	----------	------
٧٤٩٩	ALLEN	19200	20/02/81	
٧٥٢١	WARD	15000	22/02/81	
٧٦٥٤	MARTIN	15000	28/09/81	
٧٨٤٤	TURNER	18000	08/09/81	
٧٩٠٠	JAMES	11400	03/12/81	

Modifying a Column — تعديل عمود

You can change a column's datatype, size, and default value.

بإمكانك تغيير نوع بيانات العمود و الحجم والقيمة الافتراضية

لتعديل عمود تستخدم عبارة MODIFY مع تعليمة ALTER TABLE

مثال:

```
SQL> ALTER  TABLE    dept30
 2       MODIFY           ( ename VARCHAR2(15)) ;
```

Table altered.

- يمكنك تغيير عرض أو دقة الأعمدة الحاوية على بيانات عددية.
- يمكن إنقاص عرض العمود إذا احتوى العمود على قيم فارغة فقط NULL أو إذا لم يحوي الجدول أي صفوف (فارغ).
- يمكن تغيير نوع البيانات إذا احتوى العمود قيم فارغة NULL.
- يمكن تحويل نوع البيانات CHAR إلى النوع VARCHAR2 أو بالعكس إذا احتوى العمود على قيم فارغة NULL أو إذا لم تقوم بتغيير الحجم size .

Dropping a Table — عبارة إسقاط (حذف) جدول DROP TABLE

- All data and structure in the table is deleted. — - جميع البيانات و البنى في الجدول ستمحى.
- Any pending transactions are committed. — - أي إجراءات قيد الانتظار سيتم تثبيتها.
- All indexes are dropped — - جميع الفهارس ستحذف.
- You cannot roll back this statement. — - لا يمكن التراجع عن هذه التعليمة.
- Any views and synonyms will remains but are invalid. — - تبقى المناظير والمرادفات لكنها تعتبر باطلة.

الصيغة العامة:

DROP TABLE table **;**

مثال:

```
SQL> DROP  TABLE    dept30;
```

Table dropped.

Changing the name of an Object — إعادة تسمية غرض باستخدام RENAME

لتغيير اسم جدول أو منظور أو متسلسلة أو مرادف عليك تنفيذ تعليمة RENAME .

الصيغة العامة:

RENAME *old_name* **TO** *new_name* **;**

مثال:

```
SQL> RENAME  dept  TO    department;
```

Table renamed.

Truncating a Table — بتر الجدول باستخدام تعليمة TRUNCATE

تقوم تعليمة TRUNCATE TABLE بحذف كافة الصفوف أو السجلات من الجدول والتخلي عن مساحة التخزين المستخدمة لذلك الجدول كما أنه لا يمكنك استعادة الصفوف أو التراجع عن الصفوف المحذوفة باستخدام التعليمة TRUNCATE كما يمكنك حذف صفوف اختيارياً باستخدام تعليمة DELETE :

الصيغة العامة:

TRUNCATE TABLE table ;

مثال:

```
SQL> TRUNCATE TABLE department ;
```

```
Table truncated.
```

Adding Comments to a table

إضافة تعليق للجدول باستخدام COMMENT

يمكنك إضافة تعليقات للجدول أو العمود باستخدام التعليمة COMMENT .

الصيغة العامة:

```
COMMENT TO TABLE   table|column table.column
                IS 'text' ;
```

table is the name of the table — *table* تعبر عن اسم الجدول

column is the name of the column in a table — *column* تعبر عن اسم عمود في الجدول

text is the text of the comment — *text* تعبر عن نص التعليق

مثال:

```
SQL> COMMENT ON TABLE emp
     IS 'Employee Information' ;
```

```
Comment craeated.
```

Comments can be viewed through the data dictionary views:

يظهر التعليق خلال مناظر قاموس المعطيات باستخدام ما يلي :

- ALL_COL_COMMENTS
- USER_COL_COMMENTS
- ALL_TAB_COMMENTS
- USER_TAB_COMMENTS

مثال: بعد إدراج التعليق السابق نجد

```
SQL> SELECT *
     FROM   user_tab_comments ;
```

TABLE_NAME	TABLE_TYPE	COMMENTS
BONUS	TABLE	
DEPT	TABLE	
EMP	TABLE	Employee Information
MICROSOFTDTPROPERTIES	TABLE	
SALGRADE	TABLE	

القيود-Constraints

What Are Constraints ? ما هي القيود

- Constraints enforce rules at the table level. - تقوم القيود بفرض القوانين على مستوى الجدول .
- Constraints prevent the deletion of a table if there are dependencies.
- القيود تمنع الحذف من الجدول إذا كان هناك تبعيات.

The following constraint types are valid in Oracle **يبين الجدول التالي أنواع القيود المتاحة في أواكل**

Constraint القيد	Description الوصف
NOT NULL	Specifies that this column may not contain a null values لا يمكن أن يحوي العمود المقيد بهذا القيد قيم فارغة
UNIQUE **المفتاح الفريد**	Specifies a column or combination of columns whose values must be unique for all rows in the table هو عبارة عن تركيب من عمود أو مجموعة أعمدة التي تكون قيمها وحيدة من أجل جميع الصفوف في الجدول
PRIMARY KEY **المفتاح الأساسي**	Uniquely identifies each row of the table يعرف بشكل فريد كل صف من الجدول وهو أحد المفاتيح الفريد لكنه أعطي بعض الميزات الخاصة
FOREIGN KEY **المفتاح الخارجي (الغريب)**	Establishes and enforces a foreign key relationship between the column and a column of the referenced table يؤسس و يؤكد على علاقة مفتاح ثانوي بين العمود الحالي وعمود من الجدول المرجع
CHECK **قيد الإختبار**	Specifies a condition that must be true يوصّف الشرط بوجوب كونه صحيحاً

Defining Constraints تعريف القيود

الصيغة العامة:

```
CREATE  TABLE   [schema.] table
                (column datatype  [DEFAULT  expr]
                [ column_constraint] , ….
                [ table_constraint ][ ,….]) ;
```

schema is the same as the owner's name — *schema* تعبر عن نفس اسم المالك

table is the name of the table — *table* تعبر عن اسم الجدول

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

DEFAULT *expr* تخصيص قيمة افتراضية إذا كانت هذه القيمة قد حذفت من عبارة INSERT

column is the name of the column — *column* تعبر عن اسم العمود

datatype is the column's datatype and length — *Datatype* تحديد نوع البيانات وطولها الأعظمي

Column constraint is an integrity constraint as part of the column definition

column constraint قيد العمود يعبر عن كمال القيد كجزء من تعريف العمود .

Table constraint is an integrity constraint as part of the table definition

table constraint قيد الجدول يعبر عن كمال القيد كجزء من تعريف الجدول .

مثال :

```
SQL> CREATE  TABLE      emp (
                 empno      NUMBER(4),
                 ename      VARCHAR2(10),
                 …….
                 deptno     NUMBER(7 , 2)  NOT NULL,
                 CONSTRAINT  emp_empno_pk   PRIMARY KEY (EMPNO)  ) ;
```

Table created

- عادة نقوم بإنشاء القيود في نفس وقت إنشاء الجدول إلا أنه يمكننا إضافة القيود إلى الجدول بعد إنشائه .
- تنشأ القيود على مستويين مستوى العمود ومستوى الجدول :

على مستوى العمود: يكون الشكل العام

Column [**COMSTRAINT** constraint_name] constraint_type

على مستوى الجدول : يكون الشكل العام

Column , …..
[**COMSTRAINT** constraint_name] constraint_type
(column , …..) ,

The NOT NULL Constraint — القيد NOT NULL

Defined at the column level — وهو يعرف على مستوى العمود

مثال :

```
SQL> CREATE  TABLE     emp (
                 empno      NUMBER(4),
                 ename      VARCHAR2(10)   NOT NULL,
                 job        VARCHAR2(9),
                 mgr        NUMBER(4),
                 hiredate   DATE,
                 sal        NUMBER(7 , 2),
                 comm       NUMBER(7 , 2),
                 deptno     NUMBER(7 , 2)  NOT NULL ) ;
```

كما يمكنك تخصيص اسم للقيد عند إنشاء القيد وذلك بتعديل التعليمة السابقة كما يلي :

```
……. deptno      NUMBER(7 , 2)
        CONSTRAINT  emp_deptno_nn  NOT NULL ….
```

يمكننا إعطاء القيود أسماء ويفضل أن يكون لها علاقة بالجدول وبالحقل كما في السابق وإذا لم نفعل فإن أوراكل يتولى هذه المهمة بتوليد أسماء خاصة .

The UNIQUE KEY Constraint — قيد المفتاح الفريد

Defined at either the column level or the table level — وهو يعرف على أيٍّ من مستوى العمود أو مستوى الجدول

و هو عبارة عن تركيب من عمود أو عدة أعمدة بحيث تكون قيمة هذا المفتاح وحيدو حسب كل سطر في الجدول وإذا كان القيد NOT NULL غير معرف على هذا العمود فإن العمود يمكن أن يحوي القيمة NULL الفارغة .

مثال :

```
SQL> CREATE  TABLE     dept (
                 deotno     NUMBER(2),
                 dname      VARCHAR2(14)   ,
                 loc        VARCHAR2(13),
                CONSTRAINT  dept_dname_uk  UNIQUE (dname) ) ;
```

Note : The Oracle Server enforces the UNIQUE KEY constraint by implicitly creating a unique index on the unique key.

ملاحظة : ينفذ مزود أوراكل قيد المفتاح الوحيد بإنشاء فهرس وحيد ضمنياً على المفتاح الفريد.

The PRIMARY KEY Constraint — قيد المفتاح الأساسي

Defined at either the column level or the table level — وهو يعرف على أيٍّ من مستوى العمود أو مستوى الجدول

و هو عبارة عن أحد المفاتيح الفريدة و لكننا قمنا بإعطائه بعض الميزات الخاصة ويمكن تعريف مفتاح أساسي واحد في الجدول و لا يمكنه احتواء القيمة NULL .

يمكننا تعريف المفتاح كمفتاح أساسي أو فريد إذا كان مكوناً من عمود واحد بواسطة قيد على عمود بدلاً من قيد على جدول .

مثال :

```
SQL> CREATE  TABLE     dept (
                 deotno     NUMBER(2),
                 dname      VARCHAR2(14)   ,
                 loc        VARCHAR2(13),
                CONSTRAINT  dept_dname_uk  UNIQUE (dname)
                CONSTRAINT  dept_deptno_pk  PRIMARY KEY (deptno) ) ;
```

ينشأ فهرس وحيد بشكل تلقائي من أجل عمود المفتاح الأساسي

The FRREIGN KEY Constraint — قيد المفتاح الخارجي (الغريب)

Defined at either the column level or the table level — وهو يعرف على أيٍّ من مستوى العمود أو مستوى الجدول
و هو عبارة عن تركيب من مجموعة أعمدة قيمها موجودة في المفتاح الأساسي لجدول آخر .

مثال :

```
SQL> CREATE  TABLE     emp (
                empno      NUMBER(4),
                ename      VARCHAR2(10)   NOT NULL,
                job        VARCHAR2(9),
                mgr        NUMBER(4),
                hiredate   DATE,
                sal        NUMBER(7 , 2),
                comm       NUMBER(7 , 2),
                deptno     NUMBER(7 , 2)  NOT NULL,
        CONSTRAINT  emp_deptno_fk  FOREIGN KEY (deptno) REFERENCES  dept  (deptno) ) ;
```

يمكن الرجوع للمفتاح الأساسي في نفس الجدول أو في جدول آخر باستخدام تعليمة المراجع REFERENCES كما في المثال السابق حيث أن الحقل deptno والذي هو عبارة عن مفتاح خارجي في الجدول emp يرجع إلى العمود deptno والذي هو عبارة عن مفتاح أساسي في الجدول dept وبالتالي عندما تريد حذف أسطر من الجدول الأب فإنه من الضروري حذف الأسطر المرتبطة به من الجدول الأب وذلك باستخدام التعليمة ON DELETE CASCADE والتي تضاف إلى تعليمةREFERENCES والتي تجبر أوراكل على حذف الأسطر المرتبطة من الجدول الابن عندما تحذف أسطر من الجدول الأب (الأساسي).

FOREIGN KEY Constraint Keywords	
REFERENCES	Identifies the table and column in the parent table تحدد الجدول والعمود المرجع من الجدول الأب
ON DELETE CASCADE	Allows deletion in the parent table and deletion of the dependent rows in the child table تقوم بحذف الصفوف من الجدول الابن والمرتبطة مع الصفوف المحذوفة من الجدول الأب

The CHECK Constraint — قيد الاختبار

Defines a condition that each row must satisfy — وهو يحدد الشرط الذي يجب أن تحققه كافة الصفوف
Expressions that are not allowed: — التعابير الغير مسموح بها :
لا يمكنك استخدام هذه القيود على الأعمدة الغير حقيقية مثل :

- References to CURRVAL, NEXTVAL, LEVEL, and ROWNUM pseudo columns
- Calls to SYSDATE, UID, USER, and USERENV functions.: استدعاء التوابع التالية
- Queries that refer to other values in other rows.الاستفسارات التي تشير إلى قيم أخرى في صفوف أخرى

مثال :

```
SQL> CREATE  TABLE     dept (
                deotno     NUMBER(2),
                dname      VARCHAR2(14)   ,
                loc        VARCHAR2(13),
               CONSTRAINT  emp_dname_ck  CHECK (deptno  BETWEEN 10 and 99) )
;
```

Adding a Constraint — إضافة قيد

- Add or drop, but not modify a constraint
- Enable or disable constraints
- Add a NOT NULL constraint by using the MODIFY clause.

- يمكن إضافة أو حذف قيد ولكن لا يمكن التعديل
- تمكين أو إلغاء تمكين القيود
- يمكن إضافة القيد NOT NULL باستخدام عبارة MODIFY

الصيغة العامة

ALTER TABLE table
ADD [**CONSTRAINT** constraint] type (column) ;

table	is the name of the table	*table* تعبر عن اسم الجدول
constraint	is the name of the constraint	*constraint* تعبر عن اسم القيد
type	is the constraint type	*type* تعبر عن نوع القيد
column	is the name of the column	*column* تعبر عن اسم العمود المتأثر بالقيد

مثال :

```
SQL> ALTER   TABLE      emp
     ADD     CONSTRAINT emp_mgr_fk
             FOREIGN KEY (mgr) REFERENCES emp (empno);
```

Table altered.

Dropping a Constraint — إسقاط (إنهاء) قيد

- Remove the manager constraint from the EMP table

- حذف قيد عمود المدراء من جدول EMP

مثال :

```
SQL> ALTER   TABLE      emp
     DROP    CONSTRAINT emp_mgr_fk ;
```

Table altered.

- حذف قيد المفتاح الأساسي من جدول Dept وإسقاط قيد المفتاح الخارجي المرتبط به من جدول EMP وهو العمود Deptno باستخدام التعليمة CASCADE و التي تقوم بإسقاط القيود من الجداول الأبناء والمرتبطة مع القيد المحذوف من الجدول الأب كما في المثال التالي :

مثال :

```
SQL> ALTER   TABLE      dept
     DROP    PRIMARY KEY   CASCADE ;
```

Table altered.

Disabling Constraints — إلغاء تمكين القيود

- Execute the **DISABLE** clause of the **ALTER TABLE** statement to deactivate an integrity constraint.
  يتم تنفيذ عبارة DISABLE مع عبارة ALTER TABLE لتبطل عمل القيد.
- Apply the **CASCADE** option to disable dependent integrity constraints.
  يتم تفعيل خيار CASCADE لتبطل عمل القيود المرتبطة مع قيد الجدول الأب.

الصيغة العامة

ALTER TABLE table
DISABLE CONSTRAINT constraint [CASCADE] ;

table	is the name of the table	*table* تعبر عن اسم الجدول
constraint	is the name of the constraint	*constraint* تعبر عن اسم القيد

مثال :

```
SQL> ALTER   TABLE      emp
     DISABLE   CONSTRAINT emp_empno_pk  CASCADE ;
```

Table altered.

Enabling Constraints — تمكين القيود

- Execute the **ENABLE** clause of the **ALTER TABLE** statement to activate an integrity constraint.

يتم تنفيذ عبارة ENABLE مع عبارة ALTER TABLE لتفعيل عمل القيد.

- A UNIQUE or PRIMARY KEY index is automatically created if you enable a UNIQUE KEY or PRIMARY KEY constraint .

يتم إنشاء فهرس المفتاح الرئيسي والمفتاح الفريد تلقائياً عند تفعيل أي من هذين القيدين.

الصيغة العامة

```
ALTER TABLE  table
ENABLE  CONSTRAINT  constraint    ;
```

table is the name of the table — *table* تعبر عن اسم الجدول

constraint is the name of the constraint — *constraint* تعبر عن اسم القيد

مثال :

```
SQL> ALTER   TABLE      emp
     ENABLE    CONSTRAINT emp_empno_pk   ;
```

```
Table altered.
```

Viewing Constraints — عرض القيود

Query the USER_CONSTRAINTS table to view all constraint definitions and names.

لإظهار كافة أسماء و تعاريف القيود يجب إجراء استفسار على جدول القيود USER_CONSTRAINTS كما في المثال التالي:

مثال :

```
SQL> SELECT     constraint_name, constraint_type, search_condition
       FROM     USER_CONSTRAINTS
     WHERE     table_name = 'EMP'   ;
```

```
CONSTRAINT_NAME     C     SEARCH_CONDITION
------------------------------  -----  ------------------------------
EMP_MGR_FK          R
PK_EMP              P
FK_DEPTNO           R
```

ملاحظة : القيود التي لم تعطى اسم من قبل مالك الجدول يقوم أوراكل بإعطائها أسماء مثل SYS_C00674

بالنسبة للأحرف C يعبر عن القيد CHECK والحرف P يعبر عن القيد PRIMARY KEY والحرف R يعبر عن المرجع و الحرف U يعبر عن القيد UNIQUE key .

Viewing the Columns Associated with Constraints — عرض الأعمدة مرافقة للقيود

مثال :

```
SQL> SELECT     constraint_name, column_name
       FROM     user_cons_columns
     WHERE     table_name = 'EMP'   ;
```

```
CONSTRAINT_NAME        COLUMN_NAME
------------------------------  -----------
FK_DEPTNO                DEPTNO
PK_EMP                   EMPNO
```

Views- المناظير(المشاهد)

What Is a View ? ما هو المنظور

You can present logical subsets or combinations of data by creating views of tables. A view is a logical table based on a table or another view. A view contains no data of its own but is like a window through witch data from tables can be viewed or changed. The tables on witch a view is based are called based tables. The view is stored as a SELECT statement in the data dictionary.

بإمكانك إظهارأو تقديم مجموعة منطقية أو مركبة من البيانات بإنشاء مناظير للجداول. المنظور عبارة عن جدول منطقي مبني على جدول أو منظور آخر. المنظور لا يحوي بيانات خاصة به وإنما هو عبارة عن نافذة يظهر من خلالها بيانات جداول يمكن عرضها أو تغييرها. الجداول التي بني على أساسها المنظور تمسى الجداول القاعدة. يخزن المنظور بشكل يشبه عبارة الإختيار SELECT ضمن قاموس البيانات.

المنظور : هو طريقة لإخفاء المنطق الذي أدى إلى ربط الجداول التي تشكل قاعدة المنظور.

Why use View ? لماذا تستخدم المناظير

- To restrict database access. - لتقييد عملية الوصول أو الدخول إلى البيانات .
- To make a complex queries easy. - لجعل الإستعلامات المعقدة أبسط .
- To allow data in independence. - السماح باستقلال البيانات .
- To present different views of the same data. - لتقديم مناظير أو مشاهد مختلفة لنفس البيانات .

Simple Views and Complex Views **مقارنة بين المناظير البسيطة والمناظير المعقدة**

Feature الميزة	Simple Views	Complex Views
Number of tables عدد الجداول	One	One or more
Contain functions يحتوي على توابع	NO	Yes
Contain groups of data يحتوي على مجموعات من البيانات	NO	Yes
DML through view يحتوي تعليمات لغة التعامل مع البيانات	YES	Not always

Creating a View إنشاء منظور

- يحوي المنظور استفسار جزئي كجزء أساسي أثناء إنشاء المنظور.
- الاستفسار الجزئي يمكن أن يحوي تركيب عبارة SELECT معقدة.
- لا يمكن أن يحوي الاستفسار الجزئي عبارة ORDER BY.

الصيغة العامة:

```
CREATE [ OR REPLACE ] [ FORCE|NOFORCE ] VIEW view
        [(alias [ , alias] …) ]
    AS subquery
[ WITH CHECK OPTIN [ CONSTRAINT constraint ] ]
[ WITH READ ONLY ]
```

OR REPLACE re-creates the view if it already exists. إعادة إنشاء المنظور إذا كان قد أنشأ قبلاً.

FORCE creates the view regardless of whether or not the base tables exist

إنشاء المنظور بغض النظر عن كون الجداول الأساسية أو قاعدة المنظور منشأة.

NOFORCE creates the view only if the base tables exist

إنشاء المنظور فقط إذا كانت الجداول الأساسية أو قاعدة المنظور قد تم إنشاؤها وهو الإفتراضي.

view is the name of the view. *view* اسم المنظور

alias specifies names for the expressions selected by the view's query.

تحدد أسماء مستعارة للتعابير المختارة من قبل استفسار المنظور.

subquery is a complete SELECT statement . *Subquery* تعليمة select كاملة

WITH CHECK OPTION specifies that only rows accessible to the view can be inserted or updated .

تعين الصفوف المتصلة بالمنظور والتي يمكن فقط إدخالها أو تعديلها.

constraint is the name assigned to the CHECK OPTION constraint .

الاسم المخصص لقيد الإختيار CHECK OPTION

WITH READ ONLY ensures that no DML operations can be performed on this view .
تضمن عدم إنجاز أي عملية DML في هذا المنظور.

مثال :

```
SQL> CREATE VIEW    empvu10
     AS SELECT      empno, ename, job
     FROM           emp
     WHERE    deptno = 10 ;
```

```
View created.
```

تابع المثال السابق : وصف المنظور السابق

```
SQL> DESCRIBE   empvu10
```

```
Name          Null?            Type
------------  ---------------- --------------------
EMPNO         NOT NULL         NUMBER(4)
ENAME                          VARCHAR2(10)
```

- The subquery that defines a view can contain complex SELECT syntax, including joins, groups, and subqueries.
- يمكن أن يحوي الاستفسار الجزئي الذي يعرف المنظور على عبارة SELECT مركبة تضم الروابط و عمليا ت التجميع واستفسارات جزئية .
- The subquery that defines the view cannot contain an ORDER BY clause. The ORDER BY clause is specified when you retrieve data from the view .
- لا يمكن أن يحوي الاستفسار الجزئي الذي يعرف المنظور على عبارةORDER BY . لأن عبارة ORDER BY مخصصة لعملية استرجاع البيانات من المنظور.
- If you do not specify a constraint name for a view created with the CHECK OPTION, the system will assign a default name in the format SYS_Cn .
- إذا لم تقم بتخصيص اسم للقيد المنشأ للمنظور يقوم النظام بتحديد اسم افتراضي يأخذ الشكل العام SYS_Cn .
- You can use the OR REPLACE option to change the definition of the view without dropping and re-creating it .
- يمكنك تغيير تعريف المنظور بدون حذفه أو إعادة إنشاءه وذلك باستخدام التعليمة OR REPLACE .

إنشاء منظور باستخدام أسماء مستعارة للأعمدة Column Aliases ضمن الاستفسار الجزئي (الداخلي): كما في المثال التالى

مثال :

```
SQL> CREATE VIEW    salvu30
     AS SELECT      empno EMPLOYEE_NUMBER , ename NAME ,
                    sal SALARY
     FROM           emp
     WHERE    deptno = 30 ;
```

```
View created.
```

Retrieving Data from the View : **استعادة البيانات من المنظور :**

```
SQL> SELECT    *
     FROM  salvu30 ;
```

```
EMPLOYEE_NUMBER    NAME        SALARY
-------------------------------  ----------  ------------
              7499  ALLEN        1600
              7521  WARD         1250
              7654  MARTIN       1250
        ............
```

Modifying a View — تعديل المنظور

يمكن تعديل المنظور باستخدام تعليمة CREATE OR REPLACE VIEW
في المثال التالي نقوم بتعديل المنظور السابق والمسمى EMPVU10 وإضافة اسم مستعار لكل عمود كما يلي :

مثال :

```
SQL> CREATE OR REPLACE VIEW    empvu10
                               ( employee_number , employee_name , job_title )
     AS  SELECT                empno , ename , job
     FROM                      emp
     WHERE     deptno  = 10  ;
```

View created.

- Column aliases in the CREATE VIEW clause are listed in the same order as the columns in the subquery.
- ترتب الأسماء المستعرة في عبارة CREATE VIEW بنفس ترتيب الأعمدة ضمن الاستفسار الجزئي.

Creating a Complex View — إنشاء منظور مركب

- Create a complex view that contains group functions to display values from tow tables.
- إنشاء منظور مركب يحوي توابع تجميع لعرض البيانات من جدولين كما في المثال التالي:

مثال :

```
SQL> CREATE  VIEW     dept_sum_vu
                      ( name , minsal , maxsal , avgsal )
     AS  SELECT       d.dname , MIN(e.sal) , MAX(e.sal) , AVG(e.sal)
     FROM             emp  e , dept  d
     WHERE            e.deptno  = d.deptno
     GROUP BY         d.dname   ;
```

View created.

استرجاع البيانات من المنظور dept_sum_vu

```
SQL> SELECT    *
     FROM  dept_sum_vu  ;
```

NAME	MINSAL	MAXSAL	AVGSAL
ACCOUNTING	1300	5000	2916.66667
RESEARCH	800	3000	2175
SALES	950	2850	1566.66667

Rules for Performing DML Operations On a View — قواعد تطبيق تعليمات DML على المنظور

- You can perform DML operations on simple views.
- يمكنك تطبيق تعليمات DML (لغة معالجة البيانات) على المناظير البسيطة.
- You cannot remove a row if the view contains the following:
- لا يمكنك حذف أي صف أو سطر من السطور إذا احتوى على أيٍّ مما يلي:
  - Group functions . توابع تجميع
  - A GROUP BY clause. عبارة الترتيب
  - The DISTINCT keyword .التعليمة الخاصة بإلغاء التكرار
- You cannot modify data in a view if it contains:
- لا يمكن تعديل البيانات داخل منظور إذا احتوت أيٍّ مما يلي :
  - Any of the conditions mentioned in the previous slide. أي من الشروط المذكورة في السابق
  - Columns defined by expressions. أعمدة معرفة بتعابير
  - The ROWNIM pseudocolumn.
- You cannot add data:
  - The view contains any of the conditions mentioned above or in the previous slide .
    - إذا احتوى المنظور أي من الشروط السابقة أو اللاحقة .
  - There are NOT NULL columns in the base tables that are not selected by the view.
    - إذا كانت هنالك أعمدة يشترط أن لا تكون فارغة في الجداول التي تشكل قاعدة المنظور ولم يتم اختيار هذه الأعمدة.

Using the WITH CHECK OPTION Clause — استخدام عبارة خيار التحقق

يمكنك أن تضمن بقاء تعليمة الـ DML المطبقة على المنظور ضمن مجال المنظور باستخدام عبارة WITH CHECK OPTION

مثال :

```
SQL> CREATE OR REPLACE  VIEW     empvu20
     AS  SELECT      *
     FROM                emp
     WHERE               deptno  = 20
     WITH CHECK OPTION CONSTRAINT     empvu20_ck   ;
```

```
View created.
```

- ستفشل أي محاولة لتغيير رقم القسم deptno في أي سطر ضمن المنظور لأنه يتعدى على قيد خيار التحقق WITH CHECK OPTION وبالتالي إذا قمنا بمحاولة تعديل المنظور السابق فإننا نحصل على رسالة خطأ .

مثال :

```
SQL> UPDATE        empvu20
     SET            deptno = 10
     WHERE         empno = 7788 ;
```

```
ERROR at line 1 :
ORA-01402: view WITH CHECK OPTION where-clause violation
```

Denying DML Operations — رفض عمليات الـ DML

- يمكنك منع حدوث أي عملية DML عند إضافة الخيار **WITH READ ONLY** إلى تعريف المنظور.

مثال :

```
SQL> CREATE OR REPLACE VIEW     empvu10
                    ( employee_number , employee_name , job_title )
     AS  SELECT          empno , ename , job
     FROM                emp
     WHERE     deptno  = 10
     WITH READ ONLY    ;
```

```
View created.
```

- لو أننا قمنا بأي محاولة مثلاً لحذف سطر من المنظور ستكون النتيجة رسالة خطأ:

مثال :

```
SQL> DELETE  FROM        empvu10
     WHERE        employee_number  =  7782 ;
```

```
ERROR at line 1 :
ORA-01752: Cannot delete from view without exactly one key-preserved table
```

Removing a View — حذف (إسقاط) المنظور

- Remove a view without losing data because a view is based on underling tables on the database.
- حذف المنظور بدون فقدان البيانات وذلك لأن المنظور مبني على أساس جداول تابعة في قاعدة البيانات.

الصيغة العامة:

DROP VIEW view **;**

view is the name of the view. — *view* اسم المنظور

مثال :

```
SQL> DROP VIEW     empvu10 ;
```

```
View dropped.
```

Sequence- المسلسِل
Index- الفهرس
Synonym- المرادف

What Is a Sequence ? ما هو المسلسل

- Automatically generates unique numbers.
- Is a sharable object.
- Is typically used to create a primary key value.
- Replaces application code.
- Speeds up the efficiency of accessing sequence values when cached in memory.

- يولد بشكل آلي أعداداً فريدة .
- هو غرض قابل للمشاركة .
- يستخدم بشكل نموذجي لتوليد قيم المفتاح الأساسي .
- يستبدل رمز التطبيق.
- تقوم بتسريع فعالية دخول قيم السلسلة عندما يخبأها في الذاكرة.

The CREATE SEQUENCE Statement عبارة إنشاء مسلسل

تعريف قيم متسلسلة تقوم بتوليد أرقام متسلسلة بشكل تلقائي :

الصيغة العامة

```
CREATE SEQUENCE  sequence
        [ INCREMENT BY n ]
        [ START WITH n ]
        [ { MAXVALUE n | NOMAXVALUE } ]
        [ { MINVALUE n | NOMINVALUE } ]
        [ { CYCLE |  NOCYCLE } ]
        [ { CACHE n | NOCACHE } ]  ;
```

sequence is the name of the sequence generator ***sequence*** تعبر عن اسم مولد المسلسل

INCREMENT BY n تحدد الفاصل بين الأرقام السلسلة في حال كون السلسلة عبارة عن أرقام صحيحة وفي حالة إهمال هذه العبارة فإن السلسلة تتزايد بمقدار ١ .

START WITH n يحدد أول رقم يتم توليده في السلسلة (بداية السلسلة) وفي حالة إهمال هذه العبارة فإن السلسلة تبدأ بالرقم ١

MAXVALUE n تحدد أكبر رقم يمكن أو تولده السلسلة .

NOMAXVALUE تحدد أكبر قيمة ممكن الوصول إليها وهي تساوي 10^27 للترتيب التصاعدي وتساوي 1- للترتيب التنازلي.

MINVALUE n تحدد أصغر رقم في السلسلة .

NOMAXVALUE تحدد أصغر قيمة ممكن الوصول إليها وهي تساوي ١ للترتيب التصاعدي وتساوي (10^27)– للترتيب التنازلي.

CYCLE | NOCYCLE تحدد فيما إذا السلسلة ستستمر بتوليد القيم المتسلسلة حتى وإن بلغت الحد الأعلى أو الأدنى المخصص لها أو أن تتوقف عند ذلك الحد NOCYCLE وهو الخيار الافتراضي .

CACHE n | NOCACHE تحدد عدد القيم التي يقوم مزود أوراكل بتخصيصها مسبقاً ووضعها في الذاكرة ويقوم مزود أواكل افتراضيا بتخصيص مسبق لـ ٢٠ قيمة .

مثال : إنشاء سلسلة باسم DEPT_DEPTNO لتستخدم من أجل المفتاح الأساسي في الجدول DEPT ولا تقم بتفعيل الخيار CYCLE

```
SQL> CREATE  SEQUENCE  dept_deptno
        INCREMENT BY   1
        START   WITH     91
        MAXVALUE        100
        NOCACHE
        NOCYCLE   ;
```

```
Sequence created.
```

Confirming Sequences التثبت من المسلسل

يمكنك التحقق من قيم المسلسل ضمن جدول قاموس بيانات المسلسلات USER_SEQUENCES كما يلي :

```
SQL> SELECT   sequence_name , min_value , max_value , increment_by , last_number
     FROM     user_sequences ;
```

بحيث أن العمود last_number يعرض رقم المسلسل اللاحق المتاح

SEQUENCE_NAME	MIN_VALUE	MAX_VALUE	INCREMENT_BY	LAST_NUMBER
DEPT_DEPTNO	1	100	1	91

NEXTVAL and CURRVAL Pseudocolumns — الأعمدة الزائفة

- NEXTVAL returns the next available sequence value.
- NEXTVAL يعيد قيمة المسلسل المتاح التالي (اللاحق) كما يعيد قيمة المسلسل في كل مرة يتم فيها الرجوع إليه حتى بالنسبة لمستخدمين مختلفين .
- CURRVAL obtains the current sequence value. يحصل على قيمة المسلسل الحالي

Rules for using NEXTVAL and CURRVAL — **قوانين استخدام NEXTVAL و CURRVAL**

يمكنك استخدام NEXTVAL و CURRVAL ضمن ما يلي :

- ضمن قائمة عبارة SELECT والتي ليست جزء من الاستفسار الجزئي .
- ضمن قائمة عبارة SELECT في تعليمة INSERT .
- ضمن عبارة VALUES التابعة لتعليمة INSERT .
- ضمن عبارة SET التابعة لتعليمة UPDATE .

لا يمكنك استخدام NEXTVAL و CURRVAL ضمن ما يلي :

- ضمن قائمة عبارة SELECT الخاصة بالمنظور VIEW .
- ضمن عبارة SELECT مع الكلمة المفتاحية DISTINCT .
- ضمن عبارة SELECT مع التعليمات GROUP BY ، HAVING ، ORDER BY.
- ضمن استفسار جزئي داخل تعليمات SELECT أو DELETE أو UPDATE .
- ضمن القيمة الافتراضية DEFAULT exp ضمن تعليمتي CREATE TABLE أو ALTER TABLE .

Using a Sequence — استخدام المسلسل

سنقوم بإدخال اسم قسم جديد هو "MARKETING" ضمن جدول الأقسام كما يلي :

مثال :

```
SQL> INSERT INTO   dept ( deptno , dname , loc )
     VALUES        ( dept_deptno.NEXTVAL ,
                   ' MARKETING ' , ' SAN DIEGO ' ) ;
```

```
1 row created.
```

إظهار القيمة المضافة ضمن الحقل deptno بواسطة NEXTVAL وذلك باستخدام CURRVAL وذلك ضمن الجدول الفارغ DUAL :

```
SQL> SELECT   dept_deptno. CURRVAL
     FROM     dual ;
```

```
CURRVAL
--------------
      ٩١
```

و السبب هو أننا قد وضعنا القيمة ٩١ كقيمة بدء في المثال صفحة ٤٨ باستخدام التعليمة START WITH 91

أو باستعراض الجدول dept كما يلي :

```
SQL> SELECT   *
     FROM     dept
     WHERE    dname = ' MARKETING ' ;
```

DEPTNO	DNAME	LOC
٩١	MARKETING	SAN DIEGO

- إن وضع قيم المسلسل في الذاكرة يسمح بتسريع الوصول إلى تلك القيم .
- تحدث الثغرات (الفجوات) ضمن قيم المسلسل إذا حدث أي مما يلي :

- حدوث عملية تراجع Rollback .
- انهيار النظام .
- في حالة استخدام المسلسل ضمن جدول آخر.

○ يمكن عرض المسلسل التالي المتاح إذا تم إنشاؤه مع NOCACH بالاستعلام عن جدول UER_SEQUENCES .

Modifying a Sequence — تعديل المسلسل

- Change the increment value , maximum value , minimum value , cycle option , or cache option .
  - تعديل قيمة فارق الزيادة بين قيم المسلسل و القيمة العظمى والقيمة الصغرى و خيار الذاكرة Cache .

الصيغة العامة

```
ALTER SEQUENCE  sequence
        [ INCREMENT BY n ]
        [ { MAXVALUE n | NOMAXVALUE } ]
        [ { MINVALUE n | NOMINVALUE } ]
        [ { CYCLE |  NOCYCLE } ]
        [ { CACHE n | NOCACHE } ]  ;
```

مثال : تعديل المسلسل المسمى DEPT_DEPTNO وذلك بتعديل القيمة العظمى **:**

```
SQL> ALTER  SEQUENCE  dept_deptno
        INCREMENT BY   1
        MAXVALUE       999999
        NOCACHE
        NOCYCLE   ;
```

```
Sequence altered.
```

- حتى تقوم بعملية تعديل المسلسل يجب أن تملك الامتياز أو الترخيص لهذه العملية بالإضافة إلى أنه فقط أرقام المسلسل الجديدة أو المستقبلية هي التي تتأثر بالتعديل كما أنه من أجل بدء المسلسل بقيمة جديدة أي تغيير قيمة البدء START WITH عليك حذف المسلسل ثم إعادة إنشاؤه كما أنه لا يمكنك أثناء التعديل وضع قيمة عظمى تكون أصغر من القيمة الحالية أي الموجودة .

مثال : تعديل المسلسل المسمى DEPT_DEPTNO وذلك بتعديل القيمة العظمى بقيمة أصغر من آخر قيمة ٩١ موجودة بالجدول **:**

```
SQL> ALTER  SEQUENCE  dept_deptno
        INCREMENT BY   1
        MAXVALUE       90
        NOCACHE
        NOCYCLE   ;
```

```
ALTER  SEQUENCE  dept_deptno
*
ERROR at line 1:
ORA-04009: MAXVALUE cannot be made to be less
than the current value
```

Removing a Sequence — حذف (إزالة) المسلسل

يمكن إزالة المسلسل من قاموس البيانات data dictionary باستخدام تعليمة DROP :

الصيغة العامة

```
DROP  SEQUENCE  sequence
```

مثال : حذف المسلسل المسمى DEPT_DEPTNO **:**

```
SQL> DROP  SEQUENCE  dept_deptno ;
```

```
Sequence dropped.
```

Index- الفهرس

What Is a Index ? — ما هو الفهرس

- Is a schema object .
- Is used by the Oracle Server to speed up the retrieval of rows by using a pointer .
- Can reduce disk I/O by using rapid path access method to locate the data quickly .
- Is independent of the table it indexes .
- Is used and maintained automatically by the Oracle Server .

- عبارة عن غرض مخطط .
- يستخدم من قبل مزود أوراكل ليسرع عملية استرجاع الصفوف باستخدام المؤشر .
- يمكنه تصغير القرص في عملية الإدخال والإخراج باستخدام طريق الوصول السريع لتحديد مكان البيانات بسرعة .
- يستخدم ويحفظ آلياً من قبل خادم أوراكل .

كيف تنشأ الفهارس :

- ينشأ الفهرس الفريد بشكل أوتوماتيكي عندما نضع أحد القيود UNIQUE أو PRIMARY KEY ضمن تعريف الجدول .
- أو يدوياً بحيث يمكن للمستخدمين إنشاء فهارس غير فريدة على الأعمدة لتسرع من وقت عملية الوصول إلى الصفوف .

نقوم باختيار الفهرس على أنه إما عمود يحوي قيم غير مكررة ويسمى بالفريد ومن الممكن أن لا يكون المفتاح الأساسي للجدول أو عمود يحوي قيم مكررة ويسمى الفهرس الغير فريد و هو عمود يستخدم بشكل كبير ضمن تعليمة WHERE أي يتم الحصول على الصفوف عن طريقه والفهرس في بنية مثل البنية الشجرية .

Create Index — إنشاء فهرس

يمكن إنشاء فهرس على عمود أو أكثر :

الصيغة العامة:

```
CREATE  INDEX  index
  ON  table ( column [ , column ] … )  ;
```

index is the name of the index. — *index* اسم الفهرس

table is the name of the table. — *table* اسم الجدول

column is the name of the column in the table to be indexed. — *column* اسم العمود المفهرس في الجدول

مثال : قم بتحسين سرعة استعلام الوصول إلى عمود ENAME من الجدول EMP **:**

```
SQL> CREATE  INDEX  emp_ename_idx
       ON           emp(ename)  ;
```

Index created .

When to Create an Index — متى نقوم بإنشاء الفهارس

١. نستخدم الفهرس على العمود الذي يستخدم بشكل كبير ضمن عبارة WHERE أو في شرط الربط Join condition .
٢. على العمود الذي يحوي مجال واسع من القيم.
٣. على العمود الذي يحوي عدد كبير من القيم الفارغة.
٤. على عمودين أو أكثر يستخدمان بشكل كبير ضمن عبارة WHERE أو في شرط الربط Join condition .
٥. على الجدول الكبير (الضخم) والمطبق عليه استفسارات يُتوقع أن تعيد أقل من ٢ إلى ٤ % من إجمالي الصفوف .

When Not to Create an Index — متى لا نقوم بإنشاء الفهارس

١. الجدول الصغير .
٢. على أعمدة لا تستخدم في الغالب كشرط ضمن الاستفسار .
٣. أغلب الاستفسارات المتوقع أن تعيد أكثر من ٢ إلى ٤ % من إجمالي الصفوف .
٤. الجدول الذي يتم تحديثه كثيراً .

Confirming Indexes — التأكد من الفهارس (التثبت)

إن منظور قاموس البيانات المسمى USER_INDEXES يحتوي اسم الفهرس كما يحوي صفة كونه وحيداً أو لا UNIQUE or NONUNIQUE كما أن المنظور USER_IND_COLUMNS يحتوي اسم الفهرس واسم الجدول واسم العمود كما يلي :

مثال :

```
SQL> SELECT   ic.index_name , ic.column_name ,
              Ic. column_position  col_pos , ix.uniqueness
     FROM        user_indexes  ix , user_ind_columns  ic
     WHERE       ic.index_name = ix.index_name
     AND         ic.table_name = ' EMP '  ;
```

INDEX_NAME	COLUMN_NAME	COL_POS	UNIQUENESS
PK_EMP	EMPNO	1	UNIQUE
EMP_ENAME_IDX	ENAME	1	NONUNIQUE

Removing an Index — حذف (إسقاط) فهرس

يتم حذف الفهرس من قاموس البيانات Data dictionary باستخدام التعليمة DROP :

الصيغة العامة

```
DROP  INDEX  index
```

مثال : حذف المسلسل المسمى DEPT_DEPTNO :

```
SQL> DROP  INDEX  emp_ename_idx  ;
```

```
Index dropped.
```

حتى تقوم بإسقاط الفهرس يجب أن تكون المالك للفهرس أو أن يكون لديك الامتياز DROP ANY INDEX .

المرادف - Synonym

What Is a Synonym ? — ما هو المرادف

- Simplify access to objects by creating a synonym (another name for an object) .

- Refer to a table owned by another user.
- Shorten lengthy object names.

- يمكنك تبسيط عملية الوصول إلى الأغراض بإنشاء الرديف (وهو اسم آخر للغرض).
- يشير إلى جدول مملوك من قبل مستخدم آخر.
- يقوم بتقصير أطوال أسماء الأغراض.
- يقوم بإنشاء رديف لاسم جدول أو منظور محلي أو جدول أو منظور بعيد.

الصيغة العامة:

CREATE [PUBLIC] SYNONYM synonym
FOR object **;**

PUBLIC create a synonym accessible to all user . PUBLIC يفعل هذا الخيار إمكانية وصول كافة المستخدمين للرديف
synonym is the name of the synonym to be created. *synonym* اسم الرديف
object identifies the object for which the synonym is created.
object اسم الغرض الذي نريد إنشاء رديف له.

مثال : قم بإنشاء اسم مقصر للمنظور المسمى dept_sum_vu**:**
Create a shortened name for the DEPT_SUM_VU view.

```
SQL> CREATE SYNONYM d_sum
     FOR            dept_sum_vu ;
```

Synonym created.

Drop a Synonym — حذف (إسقاط) مرادف

يتم حذف المرادف من قاموس البيانات Data dictionary باستخدام التعليمة DROP :

الصيغة العامة

DROP SYNONYM synonym

مثال : حذف المرادف المسمى d_sum **:**

```
SQL> DROP SYNONYM d_sum ;
```

Synonym dropped.

- The DBA can create public synonym accessible to all user.
- يمكن لمدير قاعدة البيانات إنشاء رديف عام متاح لكافة المستخدمين باستخدام التعليمة PUBLIC .

مثال : نقوم في المثال التالي بإنشاء مرادف عام باسم DEPT للجدول DEPT التابع للمستخدم Alice :

```
SQL> CREATE PUBLIC SYNONYM d_sum
     FOR            alice.dept ;
```

Synonym created.

- يمكن فقط لمدير قاعدة البيانات حذف رديف عام PUBLIC .

مثال : حذف المرادف المسمى DEPT **:**

```
SQL> DROP SYNONYM dept ;
```

Synonym dropped.

Controlling User Access _ التحكم بوصول المستخدم

In multiple user environment you want to maintain security of the database access and use. With Oracle Server database security you can do the following :

- Control database access.

- Give access to specific object in the database.
- Confirm given and received privileges with the Oracle data dictionary.
- Create a synonym for database objects.

Database security can be classified into two categories : system security and data security .
System security covers access and use of the database at the system level such as username and password, disk space allocated to users , and system operations allowed by the user .
Database security covers access and use of the database objects and the actions that those users can have on the objects.

في بيئة متعددة المستخدمين لا بد من الحفاظ على أمن وسرية الدخول إلى البيانات واستخدامها. مع الأمن الخاص بقاعدة بيانات خادم أوراكل يمكنك القيام بما يلي :

- التحكم بالوصول إلى قاعدة البيانات .
- إعطاء ميزة الوصول إلى غرض معين من أغراض قاعدة البيانات.
- التأكد من إعطاء وتسلم أو تلقي الامتيازات عن طريق قاموس بيانات أوراكل .
- إنشاء رديف أو مرادف لأغراض قاعدة البيانات.

أمن قاعدة البيانات يمكن أن يصنف ضمن فئتين : أمن النظام system security وأمن البيانات data security .
أمن النظام : يشمل عملية الوصول واستخدام قاعدة البيانات على مستوى النظام مثل اسم المستخدم و كلمة المرور ، سعة القرص المخصصة للمستخدمين ، عمليات النظام المسموح بها من قبل المستخدم .
أمن البيانات : يشمل عملية الوصول واستخدام أغراض البيانات والأحداث التي يمكن لمستخدمين أن يؤثروا بها على هذا الغرض .

Privileges الامتيازات (السماحيات)

- Database security : (system security – data security).
- System privileges : Gain access to the database.
- Object privileges : Manipulate the content of the database objects.
- Schema : Collection of objects , such as tables, views, and sequences.

- أمن قاعدة البيانات : (أمن النظام – أمن البيانات).
- امتيازات (سماحيات) النظام : الحصول على إذن للوصول إلى قاعدة البيانات .
- امتيازات الأغراض : يتعامل مع محتويات أغراض قاعدة البيانات .
- المخطط : مجموعة من الأغراض مثل الجداول والمناظير و المسلسلات .

الامتيازات : تمثل الحق بتنفيذ عبارات SQL خاصة ، يقوم مدير قاعدة البيانات والذي يملك مستوى عالي من قابلية منح الامتيازات بمنح السماحيات للمستخدمين للوصول إلى قاعدة البيانات
كما يمكن أن يعطى المستخدمون امتياز منح سماحيات إضافية لمستخدمين آخرين .

System Privileges — امتيازات النظام

- More than 80 privileges are available for users and roles أكثر من ٨٠ امتياز متاح للمستخدمين والوظائف .
- The DBA has high-level system privileges :
  - ⇒ Create new users إنشاء مستخدمين .
  - ⇒ Remove tables حذف الجداول .
  - ⇒ Back up tables إنشاء نسخة احتياطية للجداول .

System Privilege	Operations Authorized العمليات المرخص بها
CREATE USER	هذا الامتياز من وظائف مدير قواعد البيانات DBA
DROP USER	إسقاط (حذف) مستخدم ثاني
DROP ANY TABLE	حذف جدول من أي خطة
BACKUP ANY TABLE	إنشاء نسخة احتياطية لأي جدول في أي خطة باستخدام التصدير

Creating Users — إنشاء مستخدمين

يقوم مدير قاعدة البيانات DBA بإنشاء المستخدمين باستخدام عبارة CREATE USER لا يملك المستخدم أي امتيازات في المرحلة الحالية يقوم عندها المدير بمنح عدد من الامتيازات للمستخدم ، تحدد هذه الامتيازات ما الذي يمكن للمستخدم القيام به على مستوى قاعدة البيانات :

الصيغة العامة

CREATE USER user
IDENTIFIED BY passwords ;

مثال : إنشاء المستخدم scott و كلمة المرور له tiger :

```
SQL> CREATE USER scott
     IDENTIFIED BY tiger ;
```

User created.

user is the name of the user to be created. — *user* اسم المستخدم

password specifies that the user must login with this password. — *password* كلمة سر الدخول

User System Privileges — امتيازات النظام للمستخدم

يمكن لمدير قاعدة البيانات DBA منح المستخدم امتيازات نظام محددة عندما يتم إنشاؤه كما يلي :

الصيغة العامة

CRANT privilege [, privilege ….]
TO user [, user …] ;

- An application developer may have the following system privileges :
  - ⇒ CREATE SESSION إنشاء جلسة
  - ⇒ CREATE TABLE إنشاء جدول
  - ⇒ CREATE SEQUENCE إنشاء مسلسل ضمن خطة مستخدم
  - ⇒ CREATE VIEW إنشاء منظور ضمن خطة المستخدم
  - ⇒ CREATE PROCEDURE إنشاء تابع أو إجراء مخزن أو رزمة ضمن خطة المستخدم

Creating and Granting Privileges to a Role — إنشاء ومنح امتيازات لوظيفة

وظيفة أوراكل (Oracle Role) : هو مجموعة من الامتيازات أو نوع من أنواع الولوج إلى المعلومات التي يحتاجها المستخدم والتي تتعلق بالمسؤوليات والوظائف المتاحة له.تحوي وظيفة مدير قاعدة المعطيات كل سماحيات و امتيازات النظام بما فيها إعطاء الامتيازات للآخرين .

كما يمكنك إعطاء مجموعة امتيازات لوظيفة ثم منح هذه الوظيفة لمستخدم أو أكثر وبالتالي أي مستثمر يمكنه مباشرة إعطاء سماحيات أو امتيازات لمستثمر آخر.

مثال : إنشاء الوظيفة manager :

```
SQL> CREATE ROLE manager ;
```

Role created.

مثال : منح السماحيات للوظيفة manager :

```
SQL> GRANT create table , create view
     TO manager ;
```

Grant succeeded .

مثال : منح للوظيفة manager للمستخدمين BLAKE و CLARK:

```
SQL> GRANT manager to BLAKE , CLEARK ;
```

Grant succeeded .

Changing Your Password — تغيير كلمة السر

- You can change your password by using the ALTER USER statement.

الصيغة العامة

ALTER USER user **IDENTIFIED BY** passwords ;

user is the name of the user to be created. — *user* اسم المستخدم

password specifies the new password. — *password* كلمة السر الجديدة

مثال : تغيير كلمة السر للمستخدم scott :

```
SQL> ALTER USER scott
     IDENTIFIED BY lion ;
```

User altered .

Granting System Privileges — منح سماحيات النظام

يمكن لمدير قاعدة المعطيات أن يقوم بمنح امتيازات محددة للمستخدم .

مثال : منح بعض امتيازات النظام للمستخدم scott :

```
SQL> GRANT create table , create sequence , create view
     TO scott ;
```

Grant succeeded

What is a Role ? — ما هي الوظيفة

- A role is a named group of related privileges that can be granted to the user. This method makes granting and revoking privileges easier to perform and maintain .
- A user can have access to several roles , and several users can be assigned the same role. Roles typically are created for a database application.

- الوظيفة هي عبارة عن اسم لمجموعة امتيازات يمكن أن تمنح لمستخدم ، تجعل هذه الطريقة عملية منح وسحب الامتيازات أسهل في التنفيذ والإبقاء .
- يمكن أن يمتلك للمستخدم الوصول إلى وظائف متعددة . كما يمكن تخصيص نفس الوظيفة لعدد من المستخدمين ، تم إنشاء الوظائف نموذجياً من أجل تطبيق قاعدة البيانات .
-

الصيغة العامة

CREATE ROLE role ;

role is the name of the role to be created. — *role* اسم الوظيفة المنشأة

Object Privileges - امتيازات الأغراض				
Object privilege	**Table**	**View**	**Sequence**	**Procedure**
ALTER	☒		☒	
DELETE	☒	☒		
EXECUTE				☒
INDEX	☒			
INSERT	☒	☒		
REFERENCES	☒			
SELECT	☒	☒	☒	
UPDATE	☒	☒		

Object Privileges — امتيازات الأغراض

- Object privileges vary from object to object .
- An owner has all the privileges on the object .
- An owner can give specific privilege on that owner's object .

- تتنوع امتيازات الغرض من غرض إلى آخر .
- المالك أو صاحب الغرض يملك جميع الامتيازات على الغرض التابع له .
- يستطيع المالك إعطاء سماحية معينة على غرض تابع له.

الصيغة العامة

Grant *object_priv* **[(***columns***)]**
ON *object*
TO { *user*|*role*|PUBLIC }
[WITH GRANT OPTION] ;

object_priv is an object privilege to be granted. — *object_priv* امتياز الغرض الممنوح
ALL specifies all object privileges. — ALL تحدد كافة امتيازات الغرض
ON ***object*** is the object on witch the privileges are granted. — **ON** *object* اسم الغرض الممنوح وظيفة ما
TO identifies to whom the privilege is granted. — **TO** يحدد المستخدم الذي منح إليه الامتياز
PUBLIC grant object privileges to all users. — PUBLIC منح سماحيات الغرض لكافة المستخدمين
[WITH GRANT OPTION] allows the grantee to grant the object privileges to other users and roles .
يسمح هذا الخيار للشخص الممنوح بمنح امتيازات الغرض لمستخدمين أو وظائف آخرين .

مثال١ : منح امتيازات الاستفسار على الجدول EMP للمستخدمين sue و rich **:**

```
SQL> GRANT   select
     ON      emp
     TO      sue , rich   ;
```

Grant succeeded

مثال٢ : منح الامتيازات لتحديث أعمدة معينة للمستخدم والوظيفة**:**

```
SQL> GRANT   update (dname , loc)
     ON      dept
     TO      scott , manger   ;
```

Grant succeeded

حتى تقوم بمنح امتيازات على غرض ما يجب أن يكون هذا الغرض من ضمن الخطة Schema الخاصة بك أو أن تكون قد منحت الخيار WITH GRANT OPTION والذي يمنحك الحق بتمرير أو منح امتيازات الغرض لمستخدمن آخرين أو لوظائف أخرى .

مثال٣ : منح المستخدم الحق بتمرير الامتيازات باستخدام الخيار WITH GRANT OPTION **:**

```
SQL> GRANT    select , insert
     ON       dept
     TO       scott
     WITH GRANT OPTION   ;
```

Grant succeeded

مثال4 : السماح لكافة المستخدمين ضمن النظام بالاستفسار عن الجدول DEPT الخاص بالمستخدم Alice باستخدام الكلمة المفتاحية PUBLIC والتي تجعل مالك الجدول من منح حق الوصول للجدول لكافة المستخدمين **:**

```
SQL> GRANT    select
     ON       alice.dept
     TO       PUBLIC ;
```

Grant succeeded

بعض الجداول الخاصة بقاموس البيانات	
Data Dictionary Table	**Description- الوصف**
ROLE_SYS_PRIVS	System privileges granted to roles - يحوي امتيازات النظام الممنوحة للوظائف
ROLE_TAB_PRIVS	Table privileges granted to roles – يحوي امتيازات الجدول الممنوحة للوظائف
USER_ROLE_PRIVS	Roles accessible by the user - يحوي الوظائف التي يمكن الوصول إليها من قبل المستخدم
USER_TAB_PRIVS_MADE	يحوي سماحيات الغرض الممنوحة على أغراض المستخدم Object privileges granted on the user's objects
USER_ TAB _PRIVS_RECD	Object privileges granted to the user - يحوي امتيازات الغرض الممنوحة للمستخدم
USER_COL_PRIVS_MADE	يحوي سماحيات الغرض الممنوحة على أعمدة من غرض المستخدم Object privileges granted on the columns of the user's objects
USER_COL_PRIVS_RECD	يحوي سماحيات الغرض الممنوحة للمستخدم على أعمدة محددة Object privileges granted to the user on specific columns

Revoking Object Privileges — سحب (إلغاء) امتيازات الغرض

- نستخدم التعليمة REVOKE لسحب الامتيازات الممنوحة لمستخدمين آخرين .كما ان الامتيازات الممنوحة باستخدام الخيار WITH GRANT OPTION يتم سحبها أيضاً بنفس التعليمة .

الصيغة العامة

```
REVOKE    { privilege [  , privilege … ]|ALL }
ON        object
FROM      { user [, user …]|role|PUBLIC }
  [ CASCADE CONSTRAINTS ]   ;
```

CASCADE CONSTRAINTS is required to remove any referential integrity constraints made to the object by means of the REFERENCES privilege .
هذا الخيار مطلوب لحذف أي قيود مرجعية منشأة للغرض بواسطة امتياز المراجع .

مثال : سحب الامتيازين Select و Insert على الجدول DEPT والممنوحين للمستخدم SCOTT :

```
SQL> REVOKE   select , insert
     ON       dept
     FROM     scott  ;
```

Revoke succeeded

- إذا منح المستخدم امتياز ما مع الخيار WITH GRANT OPTION يستطيع عندها أن يمنح هذا الامتياز أيضاً ،وعندها يمكن حدوث سلسلة طويلة من عملية منح الامتيازات لكن لا يسمح بعملية المنح بشكل دائري . عند سحب المالك الامتياز الذي منحه لمستخدم تقوم تعليمة سحب الامتياز REVOKE بسحب كافة الامتيازات الممنوحة بناء على هذا الامتياز.

مثال : إذا قام مستخدم ما A بمنح امتياز الاستفسار SELECT إلى مستخدم ثاني B مع خيار المنح WITH GRANT OPTION يستطيع المستخدم B عندها أن يمنح هذا الامتياز مع خيار المنح WITH GRANT OPTION إلى مستخدم ثالث C وهكذا إلى مستخدم رابع C وفي حالة قرر المستخدم A سحب هذا الامتياز من المستخدم B عندها يتم سحب الامتياز الممنوح إلى كل من المستخدمين B وC .

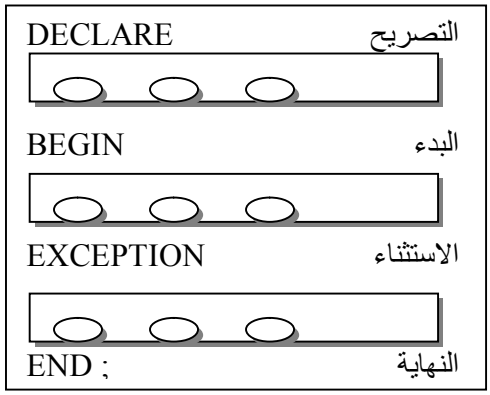
Procedural Language/SQL - (PL/SQL)

About PL/SQL — حول لغة الإجراءات PL/SQL

- PL/SQL is an extension to SQL with design features of programming languages.
- Data manipulation and query statements of SQL are included within procedural units of code.
- تعتبر لغة PL/SQL إمتداد للغة الـ SQL بالإضافة إلى ميزات التصميم الخاصة بلغات البرمجة.
- تكون تعليمات الاستفسار ومعالجة البيانات الخاصة بـ SQL متضمنة ضمن وحدات إجرائية تحوي كود .

Benefits of PL/SQL — ميزات وفوائد PL/SQL

Modularize program development — الصيغة القياسية لبرنامج التطوير



- **من ضمن ميزات لغة PL/SQL أنك تحصل على ميزة الكفاءات الإجرائية والغير متاحة في لغة الـ SQL .**

PL/SQL Block Structure — بنية كتلة PL/SQL

Every unit of PL/SQL comprises one ore more blocks. These blocks can be entirely separate or nested one within another. The basic unit (procedures, functions, and anonymous blocks) that make up a PL/SQL program are logical blocks, witch can contain any number of nested subblocks. Therefore, one block can represent a small part of another block, which in turn can be part of the whole unit of code.

كل وحدة من PL/SQL تتضمن واحدة أو أكثر من الكتل. يمكن ان تكون هذه الكتل منفصلة كلياً أو متداخلة الواحدة بالأخرى . الوحدة الأساسية والتي هي عبارة عن (إجراءات، توابع، و كتل مجهولة الاسم تعطى تلقائياً الاسم anonymous) التي تصنع برنامج الـ PL/SQL هي كتل منطقية، والتي يمكن أن تحتوي أي عدد كان من الكتل الجزئية المتداخلة. لذلك كتلة واحدة يمكن أن تمثل جزء صغيرمن كتلة أخرى ، والتي يمكن تباعاً أن تكون جزء من الوحدة الكاملة للكود .

- **من ضمن ميزات لغة PL/SQL أنها قابلة للنقل والحمل Portable .**

أي أنه يمكنك نقل أي برنامج إلى أي بيئة تدعم لغة الـPL/SQL أو Oracle Server (نظام تشغيل أو برنامج) بمعنى آخر برنامج الـ PL/SQL يمكن أن يعمل في أي مكان يمكن أن يعمل فيه مزود أوراكل . كما يمكنك أيضاً نقل الكود بين خادم أوراكل وتطبيقاتك يمكنك كتابة رزم برامج محمولة Portable Program Packages و إنشاء مكتبات يمكن إعادة استخدامها في بيئات مختلفة .

- **من ضمن ميزات PL/SQL أنه يمكن التصريح عن المعرفات.**

يصرح عن المتحولات (المتغيرات) variables والمؤشرات cursors و الثوابت constants والاستثناءات exceptions ثم استخدامها ضمن عبارة SQL و عبارات إجرائية . التصريح عن المتغيرات ديناميكياً بالاعتماد على بنى المعطيات من الجداول والأعمدة في قاعدة المعطيات.

- **يمكن أن تبرمج بنى تحكم لغة إجرائية .**
- **يمكنها معالجة الأخطاء .**

PL/SQL Block Structure — بنية كتلة PL/SQL

- **DECLARE - Optional** — **التصريح - إختياري**
  - Variables, cursors, user-defined exceptions — - المتغيرات و المؤشرات واستثناءات المستخدم
- **BEGIN – Mandatory** — **البدء - إجباري**
  - SQL statements.
  - PL/SQL statements.
- **EXCEPTION - Optional** — **استثناء - إختياري**
  - Action to perform when errors occur. — - عمل ما يحدث في حالة حدوث أخطاء
- **END ; – Mandatory** — **إنهاء - إجباري**

```
DECLARE
  v-variable  VARCHAR2(5) ;
BEGIN
  SELECT   column_name
  INTO     v-variable
  FROM     table_name ;
EXCEPTION
  WHEN  exeption_name  THEN
  …..
END ;
```

- قم بوضع فاصلة منقوطة (;) semicolon في نهاية عبارة SQL أو عبارة PL/SQL .
- استخدم علامة الشرطة (/) slash لتنفيذ كتلة الـPL/SQL المجهولة ضمن المصد (buffer) ،في حالة تم إنجاز الكتلة بنجاح وبدون حصول أخطاء من أي نوع الرسالة التي ستظهر في الخرج بالشكل التالي :

```
PL/SQL procedure successfully completed
```

- قم بوضع علامة توقف (.) period لإغلاق مصد الـSQL*Plus . إن كتلة الـPL/SQL تعامل كعبارة واحدة مستمرة ضمن المصد والفاصل المنقوطة ضمن الكتلة لا توقف أو تشغل المصد .
- يسمى الخطأ في PL/SQL بالإستثناء .

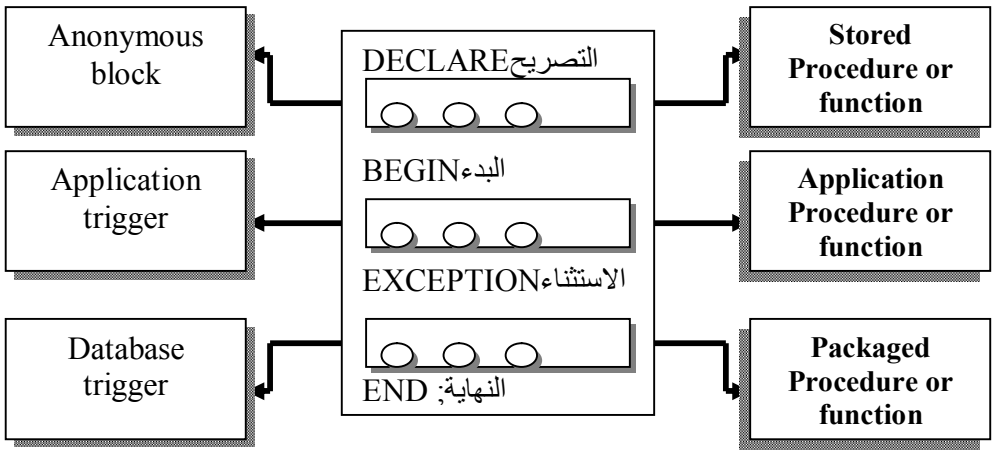
Anonymous	Procedure	Function

[DECLARE]
 	
BEGIN
---statements
[EXCEPTION]
 	
END ;
	PROCEDURE name
	IS
	
	BEGIN
	---statements
	[EXCEPTION]
	
	END ;
		FUNCTION name
		RETURN datatype
		IS
		BEGIN
		---statements
		RETURN value **;**
		[EXCEPTION]
		
		END ;

المجهول	**الإجراء**	**التابع**

Anonymous Blocks - الكتل المجهولة :
هي كتل ليس لها تسمية يتم التصريح عنها في لحظة تنفيذ التطبيق وتمريره إلى محرك PL/SQL لتنفيذه في زمن التشغيل .

Note : A function is similar to a procedure, except that a function must return a value.
ملاحظة : التابع أو الدالة مشابهة تماماً للإجراء باستثناء أن التابع يجب أن يعيد قيمة أما الإجراء فلا يستطيع إرجاع قيمة.



Program Constructs – تركيبات البرنامج

- **Anonymous block** : كتلة PL/SQL بدون اسم ضمن تطبيق أو مصدر بشكل تفاعلي .
- **Stored procedure or function** : عبارة عن كتلة PL/SQL لها اسم مخزنة في مزود أوراكل وهو يقبل بارامترات ويمكن ان يستدعى بالاسم بشكل متكرر .
- **Application procedure or function** : عبارة عن كتلة PL/SQL لها اسم مخزنة ضمن منشىء تطبيقات أوراكل Developer أو مكتبة مشتركة وهو يقبل بارامترات ويمكن ان يستدعى بالاسم بشكل متكرر.
- **Package (الرزمة)** : عبارة عن وحدة (Module) PL/SQL وهي مجموعة إجرائيات، توابع، متحولات أو عبارات SQL مجمعة سوية ضمن وحدة واحدة . يتم تحديد اسم لها .
- **Database trigger (القوادح)**: عبارة عن كتلة PL/SQL وهي أفعال أوتصرفات يجب أن تقوم بها قاعدة المعطيات عند حدوث أو تحقق حادث أو حوادث في قاعدة المعطيات وهي تكون مرافقة لجدول قاعدة البيانات.
- **Application trigger (القوادح)**: عبارة عن كتلة PL/SQL وهي أفعال أوتصرفات يقوم بها التطبيق عند حدوث أو تحقق حادث أو حوادث فيه وهي تكون مترافقة مع حدث يحدث للتطبيق وتعمل أوتوماتيكياً.

استخدام المتحولات Use of Variables

يمكنك في PL/SQL التصريح عن متحولات ومن ثم استخدامها في SQL وفي البعبارات الإجرائية في أي مكان يستخدم فيه التعبير :

١. **Temporary storage of data (التخزين المؤقت للبيانات) :**

يمكن أن تخزن البيانات بشكل مؤقت ضمن متحول أو أكثر لاستخدامها فيما بعد .

٢. **Manipulation of stored values(معالجة والتلاعب بالقيم المخزنة) :**

أي أنه يمكننا تطبيق العمليات الحسابية وعمليات أخرى بدون الدخول إلى قاعدة المعطيات .

٣. **Reusability (قابلية إعادة الاستخدام) :**

بعد التصريح يمكن استخدام المتحولات بشكل متكرر ضمن التطبيقات بسهولة .

٤. **Ease of maintenance (سهولة الصيانة)**

هناك ثلاث أنواع من البارامترات IN وهو الافتراضي و OUT و IN OUT يستخدم البارانتر IN لتمرير القيم إلى البرنامج الجزئي الذي يقوم باستدعائها . ويستخدم البارامتر OUT لإعادة القيم إلى البرنامج الجزئي المستدعي . ويستخدم البارامتر IN OUT لتمرير القيم الابتدائية إلى البرنامج الجزئي وإعادة القيم المحدثة إلى المستدعي .

أنماط المتحولات Types of Variables

➢ **تدعم PL/SQL أربع فئات من أنواع البيانات :**

- **Scalar وحيد البعد :** يحمل قيمة وحيدة ،أنماط البيانات الرئيسة هي تلك التي تتوافق مع أنماط أعمدة جداول خادم اوراكل وتدعم PL/SQL أيضاً المتحولات المنطقية Boolean .
- **Composite مركب :** مثل السجلات والتي تسمح بمجموعات من الحقول لتكون معرفة ومعالجة ضمن كتل PL/SQL .
- **Reference مرجعي :** يخبئ البيانات يستدعى بالمؤشرات.
- **LOB(large objects) الأغراض الكبيرة :** تستخدم لتخزين العناصر الكبيرة والمعلومات الكبيرة والتي تحفظ بشكل ثنائي مثل الصور وبعض الأغراض الكبيرة والتي قد يصل حجمها إلى 4GB تخزن في قاعدة المعطيات أو خارجها .

➢ **وهناك أنواع معطيات ليست تابعة لـ PL/SQL تسمى Non-PL/SQL variables**

بعض الأمثلة عن أنماط المعطيات :

- **TRUE** يمثل قيمة منطقية **Boolean** .
- **25-OCT-99** يمثل تاريخ **DATE** .
- الصور الفوتوغرافية تمثل **BLOB** وهو غرض كبير ممثل بشكل ثنائي قد يصل حجمه حتى 4GB يخزن في قاعدة المعطيات .
- نص الخطاب أو النص الطويل يمثل بالنوع **LONG ROW .**
- **٢٥٦١٢٠,٠٨** يمثل بنمط المعطيات **NUMBER** مع الدقة و القياس .
- الفيلم أو الفيديو يمثل بنمط المعطيات **BFILE** وهو ملف يحوي معلومات ممثلة ثنائياً للقراءة فقط يخزن خارج قاعدة المعطيات .
- إسم المدينة يمثل بنمط المعطيات **VARCHAR2** .

Declaring PL/SQL Variables — التصريح عن متحولات ضمن PL/SQL

الصيغة العامة

identifier **[CONSTANT]** datatype **[NOT NULL]**
[:= | DEFAULT **expr] ;**

identifier is the name of variable. — identifier يعبر عن اسم المتحول

CONSTANT constrains the variable so that its value cannot change.
CONSTANT تقيد المتغير وتجعله ثابتاً بحيث لا تتغير قيمته .

datatype is a scalar, composite , reference, or OLB datatype.
datatype نمط المتغير وحيد البعد أو مركب أو مرجعي او من الأغراض الكبيرة .

NOT NULL constrains the variable so that it must contain a value. NOT NULL تجبر المتحول أن يكون له قيمة

expr is any PL/SQL expression that can be literal, another variable….
expr أي تعبير من تعابير PL/SQL يمكن أن يكون حرفي أو متحول آخر أو أي تعبير متضمن توابع أو مشغلات operators .

مثال :

```
Declare
   v_hiredate      DATE ;
   v_deptno        NUMBER(2) NOT NULL := 10 ;
   v_location      VARCHAR2(13) := 'Atlanta' ;
   c_comm          CONSTANT NUMBER := 1400 ;
```

- التصريح عن كل متحول في سطر منفصل يجعل الكود سهل القراءة والإصلاح .
- إذا قمت باستخدام القيد NOT NULL فإنه يتوجب عليك تحديد قيمة .
- اسناد قيمة إلى المتحول يكون باستخدام معامل الإسناد (:=) وإذا لم تقم بإسناد قيمة إلى المتحول فإنه سيحوي القيمة الفارغة NULL في الوضع الإفتراضي وذلك حتى تقوم بإسناد قيمة له فيما بعد .
- في التصريح عن قيمة ثابتة يجب أن تسبق الكلمة المفتاحية CONSTANT نوع المعطيات كما في المثال التالي حيث يتم التصريح عن متحول عددي NUMBER من النوع REAL ثم اسناد القيمة ٥٠٠٠٠ إلى المتحول المسمى v_sal .

```
v_sal      CONSTANT REAL := 50000.00 ;
```

قوانين التسمية Naming Rules :

- يمكن أن يكون لمتحولين الاسم ذاته بشرط أن يكون كل منهما في كتل Blocks مختلفة .
- لا يجب ان يكون اسم المتحول مطابق لنفس اسم أعمدة الجدول المستخدم في الكتلة .
- قم بتبني قاعدة في التسمية كما يلي : قم باستخدام (v_) كبادئة لتمثيل المتحول *variable* والبادئة (g_) لتمثيل المتحول العام *global variable* وتجنب التسمية بأسماء تتعارض مع أغراض قاعدة البيانات كما في المثال التالي :

```
DECLARE
   v_hiredate        date ;
   g_deptno          number(2) NOT NULL := 10 ;
BEGIN
   …….
```

Assigning Values to Variables — إسناد قيم إلى المتحولات

الصيغة العامة

identifier := expr **;**

identifier is the name of scalar variable. — identifier يعبر عن اسم متحول وحيد البعد

مثال1 :
```
v_hiredate := '31-DEC-98' ;
```

مثال2 :
```
v_ename := 'Maduro' ;
```

- هناك طريقة أخرى لإسناد القيم إلى المتحولات وذلك باختيار select أو جلب fetch القيم إلى داخلها كما في المثال التالي والذي يحسب نسبة 10% علاوة (bonus) عند اختيار راتب الموظف salary :

مثال :

```
SELECT      sal * 0.10
INTO        v_bonus
FROM        emp
WHERE       empno = 7369 ;
```

وعندها يمكنك استخدام المتحول bonus في حسابات أخرى أو إدخال قيمته في قاعدة البيانات.

- استخدم معامل الإسناد (:=) للمتحولات التي تملك قيم غير نموذجية .

مثال :

```
v_hiredate      := to_date ( '15-SEP_1999' , 'DD_MON_YYYY' ) ;
```

- **DEFAULT** : يمكننك استخدام الكلمة المفتاحية DEFAULT للمتحولات التي تملك قيمة نموذجية .

مثال :

```
g_mgr   NUMBER(4)   DEFAULT 7839 ;
```

- **NOT NULL** : قم بفرض هذا القيد إذا كان المتحول يجب أن يمتلك قيمة حتماً ولا يمكنك اسناد القيمة الفارغة للمتحول .

مثال :

```
v_location   VARCHAR2(13)   NOT NULL := 'CHICAGO' ;
```

أنماط المتحولات وحيدة البعد — Scalar Datatypes

نمط المتحولات وحيدة البعد يحجز قيمة وحيدة لا تملك مكونات داخلية ، يمكن أن تكون المتحولات وحيدة البعد مصنفة ضمن ضمن أربعة فئات (number – character – date – Boolean)(رقم أو حرف أو تاريخ أو منطقي) .
تملك المتحولات من النمط الحرفي Character أو العددي Number أنماط جزئية وكمثال يعتبر النمطين INTEGER وPOSITIVE أنماط جزئية للنمط الأساسي NUMBER .

Base Scalar Datatypes - أنماط المتحولات وحيدة البعد الأساسية	
Datatype - النمط	**Description- الوصف**
VARCHAR2(maximum length**)**	نمط أساسي من المتحولات الحرفية **متغيرة الطول** يصل طولها إلى **32.767bytes** وليس لها حجم افتراضي
NUMBER[(precision , scale**)]**	نمط أساسي يمثل المتحولات الثابتة والممثلة بالفاصلة العائمة
DATE	نمط أساسي يمثل متحولات التاريخ والوقت تأخذ القيم بين **4712 B.C.** وبين **9999 A.D.**
CHAR[(maximum length**)]**	نمط أساسي يمثل متحولات بطول محرفي **ثابت الطول** يصل إلى **32,767bytes** إذا لم تخصص قيمة عظمى فإنه يأخذ افتراضياً القيمة ١
LONG	نمط أساسي يمثل متحولات بطول محرفي **متغير الطول** يصل إلى **32.760bytes** عرض العمود من النمط **LONG** يصل إلى **2,147,483,647 bytes**
LONG RAW	نمط أساسي يمثل بيانات ثنائية وسلاسل تصل إلى **32.760bytes** النمط **LONG ROW** غير مترجم ضمن PL/SQL
BOOLEAN	نمط أساسي يخزن واحدة من ثلاث قيم ممكنة تستخدم في الحسابات المنطقية (**TRUE-FALSE-NULL**)
BINARY_INTEGER	نمط أساسي للأعداد الصحيحة التي تقع بين **-2,147,483,647 bytes** **وبين** **2,147,483,647 bytes**
PLS_INTEGER	نمط أساسي للأعداد الصحيحة المؤشرة التي تقع بين **-2,147,483,647 bytes** وبين **2,147,483,647 bytes** يحتاج هذا النمط إلى مساحة تخزين أقل وأسرع من **NUMBER** و **BINARY_INTEGER**

أمثلة :

```
v_job          VARCHAR2(9) ;                          لتخزين عمل الموظف
v_count        BINARY_INTEGER := 0 ;                  لتخزين عداد مرات تكرار الحلقة ويعيد المتحول إلى ٠
v_total_sal    NUMBER( 9,2 ) := 0 ;                   لتجميع إجمالي الراتب للأقسام و إعادة المتحول إلى ٠
v_orderdate    DATE := SYSDATE + 7 ;                  يخزن تاريخ شحن الطلبية ويعيد المتحول إلى تاريخ أسبوع بعد اليوم
v_text_rate    CONSTANT NUMBER ( 3,2 ) := 8.25 ;      ثابت يخزن معدل الضريبة والذي لا يتغير
v_valid        BOOLEAN NOT NULL := TRUE ;             فيماإذا كانت البيانات صحيحة أو خطأ ويعيد المتحول إلى صح
```

THE %TYPE Attribute (الخاصية %TYPE):

عندما تقوم بالتصريح عن متحولات PL/SQL لتقوم باحتجاز قيم العمود يجب أن تكون متأكداً بأن نوع البيانات صحيح ودقيق ،لأنه إن لم يكن كذلك فإنه سيحدث خطأ في عبارة PL/SQL أثناء التنفيذ .

يمكنك استخدام الخاصية %TYPE للتصريح عن المتحول طبقاً لمتحول آخر مصرح عنه سابقاً أو طبقاً لعمود من قاعدة البيانات لأن هذه الخاصية تورث نمط العمود وهذا يعني انه إذا تم تغيير نمط العمود في الجدول فلن تحتاج لإجراء أي تعديل على إجرائية PL/SQLلأن التغيير يتم اوتوماتيكياً ليتناسب مع النمط الجديد وذلك أثناء التنفيذ .

غالباً ما تستخدم الخاصية %TYPE بشكل كبير عندما يتم تخزين القيمة في المتحول الذي سوف يتم اشتقاقه من جدول في قاعدة البيانات إذا أردت استخدام هذه الخصية في التصريح عن المتحولات عليك إسباقها بإسم جدول وعمود قاعدة البيانات . كما يلي :

أمثلة :

```
……
v_ename          emp.ename%TYPE ;          متحولات لتخزين اسم الموظف
v_balance        NUMBER( 7,2 ) ;           متحولات لتخزين الرصيد في الحساب البنكي
v_min_balance    v_balance%TYPE ;          أقل حساب والذي يبدأ من ١٠
……
```

الغاية من استخدام هذه الخاصية هو جعل تعريف الأنماط ضمن نص PL/SQL مستقل عن بنى المعطيات المستخدمة .

التصريح عن متحولات منطقية من النوع Boolean :

- فقط القيم (TRUE , FALSE , NULL) يمكن اسنادها أو تخصيصها للمتحول المنطقي .
- ترتبط المتحولات المنطقية بالمعاملات المنطقية (AND , OR , NOT) .
- ينتج عن المتحولات المنطقية إما TRUE أو FALSE أو NULL .
- يمكن استخدام التعابير الحسابية و الحرفية والتواريخ للحصول على قيمة منطقية Boolean .

أمثلة :

```
v_sal1 := 50000 ;
v_sal2 :=60000 ;
v_sal1 < v_sal2                                  ينتج هذا التعبير القيمة TRUE
v_comm_sal BOOLEAN := ( v_sal1 < v_sal2 )        التصريح عن متحول منطقي وإعطاؤه قيمة إبتدائية
```

أنماط المتحولات المركبة — Composite Datatypes

- جداول PL/SQL TABLES .
- سجلات PL/SQL RECORDS .

نمط البيانات المركبة والتي تعرف أيضاً بالمجموعات و تشمل الجدول TABLE والسجل RECORD والجدول المتداخل NESTED TABLE و الـ VARRAY . تستخدم السجلات لاسترجاع ومعالجة مجموعات البيانات كغرض واحد متكامل.

أنماط المتحولات الأغراض الكبيرة — LOB Datatypes

يمكن لمزود أوراكل ٨ باستخدام نمط المعطيات الأغراض الكبيرة تخزين كتل من البيانات الغير منظمة والغير مبنية مثل النصوص text والصور الفوتوغرافية graphic images وملفات الفيديو video clips و موجات الأصوات sound wave والتي يصل حجمها حتى 4 gigabytes :

- النمط **CLOB** (character large object) يستخدم لتخزين الكتل الضخمة من البيانات الحرفية مثل الوصفات Recipe .
- النمط **BLOB** (binary large object) يستخدم لتخزين الأغراض الثنائية الكبيرة مثل الصور Photo .
- النمط **BFILE** (binary file) يستخدم لتخزين الأغراض الثنائية الضخمة خارج قاعدة البيانات ضمن ملفات نظام التشغيل مثل الفيديو Movie .
- النمط **NCLOB** (national language character large object) يستخدم لتخزين الكتل الكبيرة .

Bind Variables متحولات الربط من النوع Bind مؤقت :

يصرح عن هذا النوع من المتحولات في بيئة مضيف وتستخدم لتمريرالقيم في زمن التنفيذ ، كل من الرقم أو الحرف ، داخل أو خارج رنامج واحد أو أكثر من برنامج PL/SQL ، والذي يمكنه استخدام المتحول كأي متحول آخر .

- للتصريح عن متحولات من النوع Bind في بيئة SQL*Plus يجب أن تستخدم الأمر **VARIABLE** على سبيل المثال يمكنك التصريح عن متحول رقمي NUMBER أو حرفي VARCHAR2 كما يلي :

```
VARIABLE return_code NUMBER
VARIABLE return_msg VARCHAR2(30)
```

لعرض القيمة الحالية للمتغير في بيئة SQL*Plus يجب أن تستخدم الأمر PRINT كما يلي :

```
SQL>VARIABLE g_n NUMBER
SQL>PRINT g_n
```

Referencing Non-PL/SQL Variables إسناد المتحولات من خارج PL/SQL:

تخزين الدخل السنوي لموظف ضمن متحول SQL*Plus مضيف :

```
:g_monthly_sal := v_sal / 12 ;
```

يجب إسباق عملية الإسناد بالرمز (:)

مثال : يقوم المثال التالي بحساب الراتب الشهري بالإعتماد على الراتب السنوي المدخل من قبل المستخدم يحوي النص التالي أوامر SQL*Plus ككتلة PL/SQL كاملة :

```
VARIABLE  g_monthly_sal   NUMBER
ACCEPT     p_annual_sal   PROMPT  ' Please inter the annual salary: '
DECLARE
    v_sal   NUMBER ( 9,2 )  := &p_annual_sal ;
BEGIN
   :g_monthly_sal  := v_sal / 12  ;
END ;
/
PRINT  g_monthly_sal
```

عند التنفيذ يطلب إدخال قيمة للمتحول p_annual_sal ولتكن ١٢٠٠٠ ثم ينتج القيمة التالية من الأمر الأخير :

```
G_MONTHLY_SAL
-------------------------
        1000
```

♦ يمكنك استخدام طريقة أخرى لطباعة النتيجة السابقة باستخدام الأمر DBMS_OUTPUT.PUT_LINE :

```
SET  SERVEROUTPUT   ON
ACCEPT     p_annual_sal   PROMPT  ' Please inter the annual salary: '

DECLARE
    v_sal   NUMBER ( 9,2 )  := &p_annual_sal ;
BEGIN
    v_sal  := v_sal / 12  ;
DBMS_OUTPUT.PUT_LINE ( 'The monthly salary is' || TO_CHAR(v_sal) ) ;
END ;
/
```

Writing Executable Statements كتابة تعابير قابلة للتنفيذ

بما أن PL/SQL هي امتداد SQL فإن قوانين البناء العام المتاحة لـ SQL تكون متاحةأيضاً في لغة PL/SQL .

- **Lexical units can be separated by : (Spaces – Delimiters – Identifiers – Literals – Comments)**

Delimiters - المعيِّنات :

Delimiters are simple or compound symbols that have special meaning to PL/SQL .

المعيِّنات عبارة عن رموز بسطة أو مركبة تملك معنى خاص بالنسبة لـ PL/SQL .

Simple Symbols		Compound Symbols	
Symbol	**Meaning**	**Symbol**	**Meaning**
+	Addition operator –معامل الجمع	<>	Relational operator –معامل علاقة
-	Subtraction/negation operator –معامل الطرح والنفي	!=	Relational operator –معامل علاقة
*	Multiplication operator –معامل الضرب	||	Concatenation operator –معامل السلسلة
/	Division operator –معامل القسمة	–	Single line comment indicator مؤشرتعليق وحيد السطر
=	Relational operator –معامل علاقة	/*	Beginning comment delimiter - مؤشر بداية التعليق
@	Remote access indicator –مؤشر الوصول عن بعد	*/	Ending comment delimiter - مؤشر نهاية التعليق
;	Statement terminator – مُنْهي العبارة	:=	Assignment operator – معامل الإسناد

Identifiers - المعرفات :

Identifiers are used to name PL/SQL program items and units, which include constants, variables, exceptions, cursors, cursor variables, subprograms, and packages .

تستخدم المعرفات لتسمية عناصر ووحدات برنامج الـPL/SQL والتي تحوي الثوابت والمتحولات و الإستثناءات والمؤشرات والمتحولات المشيرة والبرامج الجزئية والرزم .

- يمكن أن تحوي المؤشرات حتى ٣٠ حرف ولكن يجب أن تبدأ بحرف حصراً.
- لا تستخدم اسم المعرف كاسم للأعمدة ضمن جدول ضمن الكتلة .
- لا تستخدم الكلمات المحجوزة مثل SELECT كمعرفات إلا في حالة وضعها ضمن علامتي إقتباس كما يلي "select" .
- يجب أن تكتب الكلمات المحجوزة بحروف كبيرة لتسهيل إمكانية القراءة.

Literals - الحرفية :

- تضم الحروف الرمزية كافة الرموز القابلة للطباعة ضمن مجموعة رموز PL/SQL :أحرف Letters – أعداد Numerals – مسافات Spaces – ورموز خاصة Special symbols .
- يمكن أن تكون الحروف الرمزية قيمة بسيطة مثل 32.5- أو ترقيم علمي مثل 2E5 والتي تعني 10*2 مرفوعة للأس ٥ وتساوي = 200000 .
- يجب أن تحاط الحروف الرمزية بعلامات اقتباس فريدة كما يلي :'Henderson' := v_ename

Commenting Code — كتابة تعليق ضمن الشيفرة

- يجب إسباق التعليق وحيد السطر بخطين (--) two dashes .
- التعليق المتعدد السطور يتم وضعه ضمن العلامتين */ و /* .

مثال : حساب الدخل السنوي عن طريق الدخل الشهري

```
……
v_sal   NUMBER( 9,2 ) ;
BEGIN
   /* Compute the annual salary on the
      monthly salary input from the user */
  v_sal := &p_monthly_sal * 12 ;
END ; -- This is the end of  the transaction
```

SQL Functions in PL/SQL — توابع SQL المتاحة ضمن PL/SQL

أغلب الدوال المتاحة ضمن SQL متاحة أيضا ضمن تعابير PL/SQL :

- Single-row number functions . — التوابع الرقمية وحيدة الصف .
- Single-row character functions. — التوابع الرمزية وحيدة الصف.
- Datatype conversion functions. — توابع التحويل بين أنماط المعطيات.
- Data functions. — توابع المعطيات .

أما التوابع التالية فهي غير متاحة ضمن العبارات الإجرائية :

- DECODE . — تابع البحث والمطابقة .
- Group functions : AVG, MIN, MAX, COUNT, SUM, STDDEV, and VARIANCE .
  - توابع قيمة المجموعة المذكورة سابقاً مخصصة لمجموعات الصفوف في الجدول ولذلك فهي متاحة فقط ضمن عبارات SQL الموجودة ضمن كتل PL/SQL .

مثال ينتج خطأ : قم بتنفيذ عملية جمع كافة الأرقام المخزنة ضمن جدول PL/SQL يسمى NUMBER_TABLE :

```
v_total   := SUM ( number_table ) ;
```

PL/SQL Functions - توابع PL/SL :

تزودك لغة PL/SQL بالعديد من التوابع الفعالة والتي تساعدك في معالجة البيانات . تندرج هذه التوابع ضمن الفئات التالية :

- Error reporting - تقرير الخطأ .
- Number – الرقمية .
- Character – الرمزية .
- Conversion – التحويل .
- Date – التاريخ .
- Miscellaneous – متنوع .

مثال١ : بناء قائمة بريدية لشركة :

```
v_mailing_address :=  v_name||CHR(10)||
                      v_address||CHR(10)||v_state||
                      CHR(10)||v_zip  ;
```

- CHR is the SQL function that converts an ASCII code to its corresponding character, 10 is the code for a line feed.
- CHR هو تابع SQL يقوم بتحويل رمز الـASCII إلى رمزه المطابق أو الموافق، ١٠ هو رمز لامتداد السطر.

مثال٢ : تحويل اسم الموظف إلى حالة الأحرف الصغيرة :

```
v_ename  :=  LOWER (v_ename)
```

Datatype Conversion - التحويل بين أنماط المعطيات :

تزودك لغة PL/SQL بالعديد من التوابع الفعالة والتي تساعدك في معالجة البيانات . تندرج هذه التوابع ضمن الفئات التالية :

- تحويل البيانات إلى نمط معطيات قابل للمقارنة .
- أنماط البيانات المختلطة يمكن أن تسبب أخطاء وتؤثر على الأداء .
- توابع التحويل :
  - TO_CHAR(value , *fmt*) .
  - TO_DATE(value , *fmt*) .
  - TO_NUMBER(value , *fmt*) .

حيث value تعبر عن السلسلة الحرفية أو الرقم أو التاريخ و *fmt* شكل نموذج التحويل .

مثال :

```
DECLARE
      v_date   VARCHAR2(15)  ;
BEGIN
      SELECT  TO_CHAR (hiredate
                  'MON. DD, YYYY')
       INTO      v_date
      FROM     emp
      WHERE   empno = 7839  ;
END  ;
```

- في حالة تم التصريح عن المتحول v_date على أنه من نمط المعطيات date وقمنا بإسناد قيمة له بالشكل التالي :

```
v_date  :=  'January 13, 1998'  ;
```

فإنه ينتج خطأ تصنيف لتصحيحه نستخدم تابع لتحويل كما يلي :

```
v_date  := TO_DATE ('January 13, 1998' , 'Month DD, YYYY' ) ;
```

Nested Blocks and Variable Scope — الكتل المتداخلة والمجال المتغير

- يمكن للعبارات أن تتداخل حيثما تكون العبارة القابلة للتنفيذ مسموحة .
- الكتلة المتداخلة تصبح عبارة .
- يمكن لقسم الاستثناءات أن يحوي كتل متداخلة .
- مجال الغرض Scope هو منطقة البرنامج التي تشير إلى الغرض .

Example :

```
....
  x  BINARY_INTEGER ;
BEGIN                                  Scope of x
   ....
   DECLARE
      y  NUMBER ;
   BEGIN                               Scope of y
      ....
   END ;
   ....
END ;
```

في هذا المثال يمكن للمتحول المسمى y أن يرجع إلى المتحول x أما المتحول x فلا يمكن أن يرجع إلى المتحول y . أما إذا قمنا بإعطاء كلا المتحولين الاسم ذاته فإن قيمته تكون سارية المفعول فقط لمدة الكتلة المتداخلة .

Operators in PL/SQL — المعاملات في PL/SQL

- Logical منطقية .
- Arithmetic حسابية .
- Concatenation سلسلة (تسلسل) .
- Parentheses to control order of operations علامات حصر (أقواس) للتحكم بترتيب العمليات .

جميع المعاملات السابقة هي نفسها في SQL أما المعامل التالي فهو معامل PL/SQL:

- Exponential operator (**) المعامل الأسّي .

Order of Operations ترتيب العمليات : أسبقية المعاملات

Operator - المعامل	Operation - العملية
, NOT	أس ، نفي منطقي
+ , -	تطابق ، نفي
* , /	ضرب ، قسمة
+ , - , ||	جمع ، طرح ، تسلسل
= , != , < , > , <= , >= , IS NULL , LIKE , BETWEEN , IN	مقارنة
AND	توحيد (اقتران)
OR	تضمين

ملاحظة : من الضروري استخدام الأقواس مع التعابير المنطقية ولكنها لا تجعل النص سهل القراءة .

مثال ١ : قم بإنشاء تزايد من أجل الحلقة

Example1 : Increment the index for a loop.

```
v_count   :=  v_count  +  1 ;
```

مثال ٢ : وضع قيمة منطقية

Example2 : Set the value of Boolean flag.

```
v_equal   :=  (v_n1  =  v_n2)  ;
```

مثال ٣ : صادق على رقم الموظف إن كان يحوي قيمة. **Example3** : Validate an employee number if it contains a value

```
v_valid   :=  (v_empno IS NOT NULL )  ;
```

ملاحظة : أثناء عملك مع القيم الفارغة NULL يجب أن تتجنب بعض الأخطاء الشائعة من خلال وضع ما يلي في اعتبارك :

- المقارنات التي تتضمن القيمة NULL تكون نتيجتها NULL دائماً .
- تطبيق المعامل المنطقي NOT على القيمة NULL ينتج NULL .
- في عبارات التحكم الشرطية إذا نتجت عن الشرط القيمة NULL فإن السلسلة المترابطة لعبارتها لن تُنجز .

مثال : يقوم المثال التالي باستخدام متحول الربط g_salary :

VARIABLE g_salary **NUMBER**

```
DECLARE
    v_sal    emp.sal%TYPE ;
BEGIN
    SELECT   sal
    INTO     v_sal
    FROM     emp
    WHERE    empno = 7369 ;
   :g_salary  := v_sal ;
END ;
/
```

```
SQL>PRINT  g_salary

G_ SALARY
----------------
            800
```

R **كيف تجعل صيانة الكود أسهل :**

- قم بتوثيق الكود باستخدام التعليقات .
- التطوير لحالة الكود المصطلح عليها.
- تطوير اصطلاحات التسمية للمعرفات identifiers والأغراض الأخرى.
- تحسين قابلية القراءة بترك فراغات عند الكتابة في بداية الفقرة .

Code Conventions – اصطلاحات الشيفرة		
Category - الفئة	**Case convention-اصطلاح الحالة**	**Examples - أمثلة**
SQL statements	Uppercase	SELECT , INSERT
PL/SQL keywords	Uppercase	DECLARE , BEGIN , IF
Datatypes	Uppercase	VARCHAR2 , BOOLEAN
Identifiers and parameters	Lowercase	v_sal, emp_cursor, g_sal, p_empno
Database tables and columns	Lowercase	emp , orderdate , deptno

Code Naming Conventions – اصطلاحات التسمية في الكود		
Identifier - المعرف	**Naming convention-اصطلاح التسمية**	**Example - مثال**
Variable	v_name	v_sal
Constant	c_name	c_company_name
Cursor	*name*_cursor	emp_cursor
Exception	e_name	e_too_many
Table type	*name*_table_type	amount_table_type
Table	*name*_table	order_total_table
Record type	*name*_record_type	emp_record_type
Record	*name*_record	customer_record
SQL*Plus substitution variable (also referred to as substitution parameter)	p_name	p_sal
SQL*Plus global variable (also referred to as host or bind variable)	g_name	g_year_sal

R من أجل الوضوح يفضل تقسيم الشيفرة ووضع مسافات قبل السطور حسب مستويات الشيفرة كما في الأمثلة التالية:

```
DECLARE
    v_deptno      NUMBER(2) ;
    v_location    VARCHAR2(13) ;
BEGIN
    SELECT    deptno ,
              loc
    INTO      v_deptno ,
              v_location
    FROM      dept
   WHERE    dname = 'SALES' ;
END ;
```

```
BEGIN
    IF    x=0   THEN
                y :=1 ;
    END   IF ;
END ;
```

لنقم بمقارنة بسيطة بين طريقتي

```
IF    x>y   THEN
      v_max  := x ;
ELSE
      v_max  := y ;
END IF ;
```

```
=x; ELSE  v_max:=y; END IF ;
```

كتابة كود كما في المثال التالي :

Determining Variable Scope — تحديد مجال المتحول

لنأخذ التمرين التالي :
قيم كتلة الـPL/SQL في العرض السابق . ثم قم بتحديد قيم كل من المتحولات التالية وفقاً لقواعد المجال .

```
....
DECLARE
V_SAL          NUMBER( 7,2 ) := 60000 ;
V_COMM         NUMBER( 7,2 ) := V_SAL * .20 ;
V_MESSAGE      VARCHAR2(255) := ‘ eligible for commission ‘ ;
BEGIN …
  DECLARE
    V_SAL          NUMBER( 7,2 ) := 50000 ;
    V_COMM         NUMBER( 7,2 ) := 0 ;
    V_TOTAL_COMP   NUMBER( 7,2 ) := V_SAL + V_COMM ;
  BEGIN …
    V_MESSAGE    := ‘ CLERK not ‘||V_MESSAGE  ;
  END ;
    V_MESSAGE    := ‘ SALESMAN ‘||V_MESSAGE  ;
END ;
```

\ Evaluate the PL/SQL block on the slide. Determine each of the following values according to the rules of scoping.

- The value of **V_MESSAGE** in the subblock is :
  “CLERK not eligible for commission “ and the data type is **VARCHAR2 .**
- The value of **V_TOTAL_COMP** in the main block is :
  Illegal because V_TOTAL_COMP is not visible outside the subblock .
- The value of **V_COMM** in the main block is :
  “12000 “ and the data type is **NUMBER.**
- The value of **V_COMM** in the subblock is :
  “ 0 “ and the data type is **NUMBER.**
- The value of **V_MESSAGE** in the main block is :
  “ SALESMAN eligible for commission “ and the data type is **VARCHAR2 .**

مقارنة بين أنواع عبارات SQL وعبارات PL/SQL :

- لا تعتبر كتلة PL/SQL وحدة إجرائية . أوامر التثبيت والتراجع Commits ، Savepoints ، Rollbacks مستقلة عن الكتل . ولكن يمكنك إصدارها ضمن كتلة .
- لا تدعم PL/SQL أوامر لغة تعريف البيانات DDL مثل CREATE TABLE و ALTER TABLE و DROP TABLE .
- لا تدعم PL/SQL لغة التحكم بالبيانات DCL . مثل GRANT أو REVOKE .

SQL Statements in PL/SQL — تعليمات SQL ضمن PL/SQL

SELECT Statement in PL/SQL — عبارة الاختيار SELECT في PL/SQL

- استرجاع البيانات من قاعدة البيانات باستخدام عبارة SELECT :

الصيغة العامة

```
SELECT    select_list
INTO      {variable_name[ , variable_name] …
          | record_name }
FROM      table
WHERE     condition ;
```

Select_list قائمة من عمود واحد على الأقل يمكن أن تحوي تعابير SQL ، توابع الصفوف Row functions ، توابع قيمة المجموعة Group functions .

Variable_name متحول وحيد البعد ليقوم بحفظ (احتجاز) القيم المعادة .

Record_name مسجل PL/SQL ليقوم بحمل (احتجاز) القيم المعادة .

Table تخصيص اسم جدول من قاعدة البيانات .

condition مركب من أسماء أعمدة وتعابير و ثوابت و معاملات مقارنة تتضمن متحولات وثوابت PL/SQL .

عبارة INTO مطلوبة وإجبارية في PL/SQL عند استخدام تعليمة SELECT وتوضع بين عبارة SELECT وعبارة WHERE وذلك لتحديد أسماء المتحولات التي تقوم بحمل القيم التي تعيدها SQL والمتمثلة بتعليمة SELECT . يجب إعطاء متحول واحد لكل عنصر يتم اختياره ويجب أن ينسجم ترتيبها مع العناصر المختارة .

You use the **INTO** clause to populate either PL/SQL variables or host variables .

```
DECLARE
   v_deptno          NUMBER( 2 ) ;
   v_loc             VARCHAR2( 15 ) ;
BEGIN
    SELECT    deptno , loc
    INTO      v_deptno , v_loc
    FROM      dept
    WHERE     dname  = 'SALES' ;
….
END ;
```

الاستعلام يجب أن يعيد صف واحد وواحد فقط .
لأنه في حالتي إعادة أكثر من صف أو لم يعيد أي صف فإنه ينتج خطأ .
في تلك الحالتين يمكن استخدام الإستثنائين NO_DATA_FOUND أو TOO_MANY_ROWS ضمن قسم الاستثناءات **EXCEPTION** . والذي يمكن من خلاله القيام بإجراء معين في حالة حدوث أحد الأمرين .

Retrieving Data in PL/SQL — استرجاع البيانات في PL/SQL

- Retrieve the order date and the ship date for the specified order .
- استرجاع تاريخ الطلبية و تاريخ الشحن من أجل طلبية محددة .

```
DECLARE
   v_orderdate          ord.orderdate%TYPE ;
   v_shipdate           ord.shipdate%TYPE ;
BEGIN
    SELECT    orderdate, shipdate
    INTO      v_orderdate, v_shipdate
    FROM      ord
    WHERE     id  = 620 ;
….
END ;
```

بهذا يرث المتحول v_orderdate نوع بيانات العمود orderdate من الجدول ord لأننا استخدمنا %TYPE

كما يرث المتحول v_shipdate نوع بيانات العمود shipdate من الجدول ord لأننا استخدمنا %TYPE

- قم بإنهاء أي تعليمة SQL باستخدام الفاصلة المنقوطة (;) .
- يجب استخدام عبارة INTO مع تعليمة SELECT إن كانت مضمنة في PL/SQL .
- عبارة WHERE إختيارية ويمكن أن تستخدم لتخصيص المتحولات المدخلة والثوابت والحروف وتعابير PL/SQL .
- قم بتخصيص نفس عدد المتحولات المخرجة في عبارة INTO كأعمدة في عبارة SELECT . وكن متأكداً أنها متوافقة بالترتيب وأن نوع بياناتها متوافقة.

- استرجاع مجموع الرواتب لجميع الموظفين في قسم محدد .

```
DECLARE
   v_sum_sal            emp.sal%TYPE ;
   v_deptno             NUMBER  NOT NULL :=  10 ;
BEGIN
    SELECT    SUM(sal)  -- group  function
    INTO      v_sum_sal
    FROM      emp
    WHERE     deptno  = v_deptno ;
END ;
```

لا يمكن استخدام توابع قيمة المجموعة Group Functions ضمن تركيب عبارة

PL/SQL فهي تستخدم ضمن تعليمات SQL داخل أو خلال كتلة PL/SQL .

Manipulating Data in PL/SQL — معالجة البيانات في PL/SQL

Make changes to the database tables by using DML commands :

القيام بتغييرات على جداول قاعدة البيانات باستخدام أوامر DML :

- INSERT أمر إضافة صفوف جديدة من البيانات إلى الجدول .
- UPDQTE أمر التعديل يقوم بتعديل صفوف موجودة ضمن الجدول .
- DELETE أمر الحذف يحذف الصفوف التي لا نريدها من الجدول .

يمكنك إجراء تعديلات على قاعدة البيانات باستخدام أوامر DML ضمن PL/SQL . إن تضمين تعليمتي COMMIT أو ROLLBACK ضمن كود PL/SQL يحرر أقفال الصف وأقفال الجدول .

إدخال البيانات Inserting Data :

إضافة بيانات موظف جديد إلى الجدول emp :
في هذا المثال يجب أن يكون هناك مسلسل sequence منشأ على العمود empno باسم empno_sequence .

```
BEGIN
   INSERT   INTO    emp(empno, ename, job, deptno )
   VALUES ( empno_sequence.NEXTVAL ,  'HARDING' ,
                    'CLERK' ,  10 ) ;
END ;
```

تعديل البيانات Updating Data :

زيادة رواتب جميع الموظفين في الجدول emp والذين يعملون كمحللين analysts .
لإسناد قيم إلى متحولات نستخدم (:=) أما في حالة إسناد أعمدة نستخدم (=) دائماً .

```
DECLARE
   v_sal_increase              emp.sal%TYPE := 2000 ;
BEGIN
   UPDATE      emp
   SET         sal = sal + v_sal_increase
   WHERE       job = 'ANALYST' ;
END ;
```

حذف البيانات Deleting Data :

حذف الصفوف التي تخص القسم رقم ١٠ من الجدول emp .

```
DECLARE
   v _deptno              emp.deptno%TYPE := 10 ;
BEGIN
   DELETE      emp
   WHERE       deptno = v_deptno ;
END ;
```

- حذف طلبية محددة :

```
DECLARE
   v _ordid              ord.ordid%TYPE := 605 ;
BEGIN
   DELETE   FROM    item
   WHERE            ordid = v_ordid ;
END ;
```

Naming Conventions — استخدام اصطلاحات التسمية

? Use a naming convention to avoid ambiguity in the WHERE clause .
? Database columns and identifiers should have distinct names .
? Syntax errors can arise because PL/SQL checks the database first for column in the table .

? استخدم اصطلاح التسمية لتفادي أي إلتباس في عبارة WHERE .
? يجب أن يكون لأعمدة ومعرفات قاعدة البيانات أسماء متميزة .
? يمكن أن تحدث أخطاء نحوية لأن PL/SQL تتحقق من قاعدة البيانات من أجل العمود في الجدول .

ولذلك يجب تطبيق اصطلاحات التسمية المتفق عليها لتجنب حدوث الخطأ كما في المثال التالي :

```
DECLARE
   orderdate          ord.orderdate%TYPE ;
   shipdate           ord. shipdate%TYPE ;
   ordid              ord. ordid %TYPE := 601 ;
BEGIN
   SELECT   orderdate , shipdate
   INTO     orderdate , shipdate
   FROM     ord
   WHERE    ordid = ordid ;
END ;
SQL>
DECLARE
*
ERROR at line 1:
ORA-01422: exact fetch returns more than requested
number of rows
ORA-06512: at line 6
```

في هذا المثال ينتج استثناء تنفيذ وذلك لأن PL/SQL يقوم بالتحقق فيما إذا كان المعرف عبارة عن عمود في قاعدة البيانات وإن لم يكن فإنه من المفروض أن يكون معرف PL/SQL .
لا توجد إمكانية لحدوث إلتباس في عبارة SELECT لأن أي معرف ضمن تلك العبارة يجب أن يكون اسم عمود من قاعدة البيانات ولا توجد إمكانية لحدوث إلتباس في عبارة INTO لأن المعرفات ضمن عبارة INTO يجب أن تكون أسماء متحولات .
فقط ضمن عبارة WHERE يمكن أن يحدث إلتباس أو تشويش .

SQL Cursor — مؤشر SQL

- A cursor is a private SQL work area.
- There are two types of cursors :
  - Implicit cursors .
  - Explicit cursors.
- The Oracle Server uses implicit cursors to parse and execute your SQL statements.
- Explicit cursors are explicitly declared by the programmer.

- المؤشر هو منطقة عمل خاصة بـ SQL .
- هناك نوعين من المؤشرات :
  - مؤشرات ضمنية .
  - مؤشرات صريحة (واضحة) .
- يستخدم مزود أوراكل المؤشرات الضمنية لإعراب وتنفيذ تعليمات SQL.
- يتم التصريح عن المؤشرات الصريحة بوضوح من قبل المبرمج.

خواص مؤشر SQL - SQL Cursor Attributes :

يمكنك اختبار حصيلة تعليمات SQL باستخدام خواص المؤشر . حيث تتيح لك هذه الخواص أن تخمن ما الذي حدث عندما يتم استخدام المؤشر الضمني الأخير، تستخدم هذه الخواص كدالات ولا يمكنك استخدامها ضمن تعليمات SQL .

Cursor Attributes	
SQL%ROWCOUNT	عدد الصفوف المتأثرة بآخر تعليمة SQL – وهو قيمة صحيحة
SQL%FOUND	صفة منطقية تقدر بـTRUE إذا أثرت آخر تعليمة SQL على صف واحد أو أكثر
SQL%NOTFOUND	صفة منطقية تقدر بـTRUE إذا لم تؤثر آخر تعليمة SQL على أي صف
SQL%ISOPEN	دائماً تقدر بـFALSE لأن PL/SQL تغلق المؤشر الضمني مباشرة بعد تنفيذه

- قم بحذف الصفوف التي تحتوي رقم طلبية محدد من الجدول ITEM ، ثم قم بطباعة عدد الصفوف المحذوفة .

```
VARIABLE  rows_deleted   VARCHAR2(30)
DECLARE
  v_ordid    NUMBER := 605 ;
BEGIN
   DELETE  FROM  item
   WHERE              ordid = v_ordid ;
     :rows_deleted := (SQL%ROWCOUNT  ||
                                  '   rows deleted .  ' ) ;
END ;
/
PRINT  rows_deleted
```

Writing Control Structures — كتابة بنى التحكم

Controlling PL/SQL Flow of Execution — التحكم بتدفق تنفيذ PL/SQL

يمكنك التحكم بالتدفق المنطقي للتعليمات باستخدام عبارة IF الشرطية وبنى التحكم الحلقية :

❁ **عبارات IF الشرطية :**

- IF-THEN-END IF .
- IF-THEN-ELSE-END IF .
- IF-THEN-ELSIF-ENDIF .

الصيغة العامة لعبارة IF

```
IF  condition  THEN
      statements ;
[ELSIF  condition   THEN
      statements ; ]
[ELSE
      statements ; ]
END  IF ;
```

condition متحول منطقي أو تعبير ما تكون نتيجته TRUE أو FALSE أو NULL ويأتي بشكل متسلسل مع التعليمات التي تنفذ فقط إذا نتج عنه القيمة TRUE .

THEN عبارة ترافق التعبير المنطقي وتأتي بعدها التعليمات التي تنفذ في حالة تحقق الشرط.

statements يمكن أن تكون تعليمة SQL أو PL/SQL واحدة أو أكثر ويمكن أن تحوي عبارات IF متداخلة .

ELSIF كلمة مفتاحية يأتي بعدها تعبير منطقي يتم الوصول إليه إن كانت نتيجة التعبير الأول FALSE أو NULL.

ELSE كلمة مفتاحية يأتي بعدها مجموعة تعليمات تنفذ في حالة وصل التحكم إليها.

2 عبارة IF البسيطة Simple IF Statements :

مثال1: قم بوضع القيمة ٢٢ ضمن رقم المدير manager ID إذا كان اسم الموظف Osborne .

```
IF  v_ename = 'OSBORN'  THEN
    v_mgr  := 22 ;
END  IF ;
```

مثال2 : قم بوضع القيمة Salesman ضمن Job title و القيمة ٣٥ في رقم القسم Department number وعلاوة commission تصل إلى 20% إذا كان الاسم الأخير للموظف Miller .

```
IF  v_ename = 'MILLER'   THEN
    v_job        := 'SALESMAN' ;
    v_deptno  :=  35;
    v_new_comm  := sal * 0.20 ;
END  IF ;
....
```

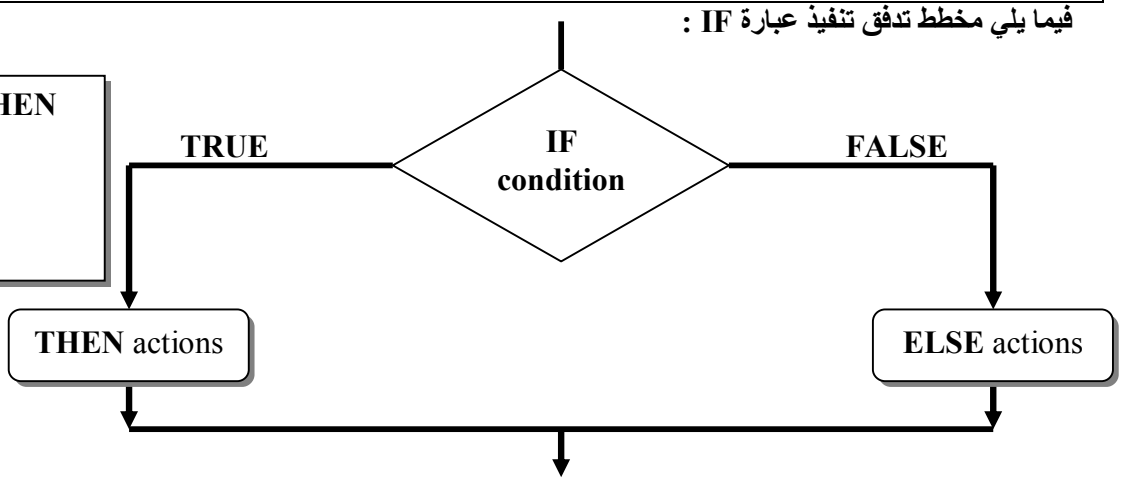
i. عند كتابة كود تذكر بأن **ELSIF** كلمة واحدة وأن **END IF** كلمتين .
ii. إن كانت نتيجة الشرطا المنطقي الأول **TRUE** تنفَّذ مجموعة التعليمات الأولى أما إن كانت نتيجته **FALSE** أو **NULL** يتم تخطي مجموعة التعليمات والإنتقال إلى الخطوة التالية كما يسمح بأي عدد من عبارات **ELSIF** .
iii. يمكن أن يكون هنالك على الأكثر عبارة **ELSE** واحدة .
iv. قم بترك فراغات قبل التعليمات المنفذة للوضوح .

عبارة IF الشرطية من الشكل IF-THEN-ELSE — IF-THEN-ELSE Statement

فيما يلي مخطط تدفق تنفيذ عبارة IF :

الشكل العام

```
IF  condition1  THEN
      statement1 ;
ELSE
      statement2 ;
END  IF ;
```



تعليمات IF المتداخلة :

يمكن أيضاً أن تضم مجموعة الأحداث الناتجة عن تعليمة IF الأولى تعليمات IF إضافية قبل أن تنجز الأحداث .يمكن أن تضم كل من عبارتي THEN و ELSE تعليمات IF .كل تعليمة من تعليمات IF المتداخلة يجب أن يتم إنهاؤها باستخدام END IF .

مثال ١: ضع إشارة flag للطلبيات Orders عندما يكون هناك أقل من خمسة أيام بين تاريخ الطلب order date وبين تاريخ الشحن ship date .

```
IF  v_shipdate – v_orderdate < 5    THEN
    v_ship_flag        := 'Acceptable'  ;
ELSE
    v_ship_flag        := 'Unacceptable'  ;
END  IF ;
....
```

مثال2 : قم بتغيير نوع عمل الموظف إلى Manager إذا كان اسم الموظف King . وإذا كان اسم الموظف غير King ضع نوع العمل Clerk .

```
IF  v_name  = 'KING'    THEN
    v_job        := 'MANAGER'  ;
ELSE
    v_job        := 'CLERK'  ;
END  IF ;
```

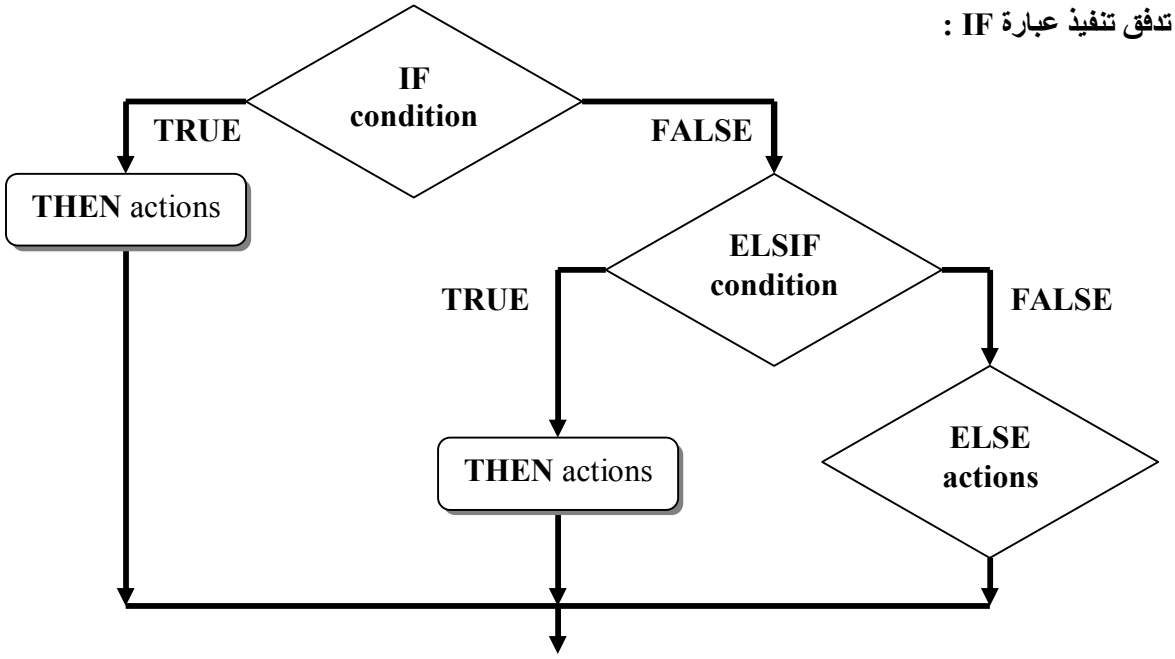
IF-THEN-ELSIF Statement — عبارة IF الشرطية من الشكل IF-THEN-ELSIF

عندما يكون ممكناً قم باستخدام عبارة ELSIF بالإضافة إلى تعليمات IF المتداخلة يصبح الكود أسهل للقراءة والفهم ويكون المنطق مميز بشكل واضح . إذا كان الحدث ضمن عبارة ELSE يتألف من تعليمة IF ثانية فإنه يكون من الأسهل استخدام عبارة ELSIF . وهذا يجعل الكود أسهل لأننا نقوم بحذف عبارات END IF المتداخلة في نهاية كل مجموعة من الشروط والأحداث .

الشكل العام

```
IF condition1 THEN
      statement1 ;
ELSIF condition2 THEN
      statement2 ;
ELSIF condition3 THEN
      Statement3 ;
END IF ;
```

فيما يلي مخطط تدفق تنفيذ عبارة IF :




```
....
IF v_deptno = 10 THEN
     v_comm      := 5000 ;
ELSIF v_deptno = 20 THEN
     v_comm      := 7500 ;
ELSE
     v_comm      := 2000 ;
END IF ;
....
```

EX1 :Determine an employee's bonus based upon the employee's department.

مثال ١ : قم بوضع علاوة للموظفين وذلك بالاعتماد على القسم أي حسب رقم قسم الموظف .

```
....
IF v_start > 100 THEN
     v_ start  := 2 * v_ start    ;
ELSIF v_ start  >= 50 THEN
     v_ start  := .5 * v_ start    ;
ELSE
     v_ start  := .1 * v_ start    ;
END IF ;
....
```

EX2 : For a given value, calculate a percentage of that value based on a condition.

مثال ٢ : من أجل قيمة مدخلة قم بحساب النسبة المئوية بناءً على الشرط .

Building Logical Conditions — بناء الشروط المنطقية

You can build a simple Boolean condition by combining number, character, or date expressions with a comparison operator . In general, handle null values with the IS NULL operator.

يمكنك بناء شرط منطقي بسيط وذلك بضم أرقام وأحرف و تعابير التاريخ باستخدام معامل مقارنة ' عموماً يتم التعامل مع القيم الفارغة NULL باستخدام المعامل IS NULL .

Any expression containing a null value evaluates to NULL, with the exception of a concatenated expression, witch treats the null value as an empty string.

أي تعبير يحوي قيمة فارغة يقدر أو ينتج قيمة فارغة NULL مع استثناء حالة ضم السلاسل والتي تعامل القيمة الفارغة على أنها سلسلة فارغة .

مثال ١ : ينتج عن كل من التعبيرين التاليين القيمة NULL إن كان المتحول v_sal ذو قيمة فارغة:

v_sal >1000 , v_sal * 1.1

مثال 2 : لا ينتج في هذا المثال القيمة NULL إن كانت قيمة المتحول v_string فارغة أي NULL :

'PL' || v_string || 'SQL'

Logic Tables — جداول المنطق

يمكن بناء شرط منطقي بسيط باستخدام معاملات المقارنة والتي تبين الجداول التالية القيم المنطقية التي تنتج عن تلك الشروط :

AND	True	False	NULL
True	True	False	NULL
False	False	False	False
NULL	NULL	False	NULL

OR	True	False	NULL
True	True	True	True
False	True	False	NULL
NULL	True	NULL	NULL

NOT	
True	False
False	True
NULL	NULL

مثال عن الشروط المنطقية : ما هي قيمة V_FLAG في كل حالة :

```
v_flag := v_order_flag AND v_available_flag ;
```

V_ORDER_FLAG	V_AVAILABLE_FLAG	V_FLAG
True	True	True
True	False	False
NULL	True	NULL
NULL	False	False

Iterative Control : LOOP Statements — التحكم التكراراي : التعابير الحلقية

- Loops repeat a statement or sequence of statements multiple times .
- There are three loop types :
  - Basic loop .
  - **FOR** loop .
  - **WHILE** loop .

- تقوم الحلقات بتكرار التعليمة أوسلسلة التعليمات عدد من المرات .
- يوجد ثلاث أنواع من الحلقات :
  - الحلقة الأساسية .
  - حلقة FOR .
  - حلقة WHILE .

الحلقة الأساسية – Basic Loop :

الصيغة العامة

```
LOOP                                  -- delimiter
    Statement1;                       -- statements
…..
    EXIT  [ WHEN CONDITION ] ;  --  EXIT statement
END LOOP ;                            -- delimiter
```