الأكاديمية العربية الدولية

المقررات الجامعية

www.aiacademy.info | care@aiacademy.info

كتـــاب في

لغة الإستعـلام المهيكل

SQL & SQL* PLUS

إعـداد /

ماهر محمد أحمد الرياشي

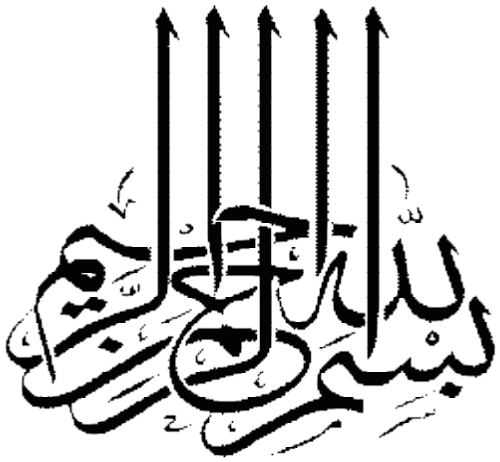
لغة الاستعلام المهيكل

SQL & SQL * PLUS
(oracle 9i)

الجزء الأول

إعداد /
ماهر محمد أحمد الرياشي

8 / 2009



إهداء

إلى أبي وأمي .... إجلالاً وإكباراً

الذين تعبوا من أجل تحصيلي على العلم السامي

إلى من تقاسمت معهم حلو الحياة ومرها .... إخواني وأخواتي

إلى أهلي و أحبابي ... إلى أصدقائي الذين

قاموا بتشجيعي ودعمي بعمل وانزال هذا الكتاب

بثقة واقتدار..

إلى كافة زملائي وزميلاتي في جامعة صنعاء بكلية العلوم

قسم الحاسوب وخاصة الدفعة الثامنة والتاسعة

أهدي هذا الجهد المتواضع

مقدمة الكتاب

الحمد لله والصلاة والسلام على رسول الله ، محمد بن عبد الله ، وآله وصحبه ومن والاه ....

وبعد :

هذا الكتاب يتناول مواضيع عديدة عن أوامر لغة (sql) ، وحاولت التركيز على أهم المواضيع التي قد تفيد القارئ ، ولا يخفى على أحد أن هذه اللغة غزيرة في معلوماتهـــا، ولذلك قمت بالتركيز على الدوال الأكثر استخداماً ، والدوال الأكثر متطلباً للمبرمجين والدارسين على حد سواء، ويحتوي هذا الكتـــــاب على أكثر من (220) مثال ، وهذه الأمثلة عبارة عن أمثلة توضيحيه لاستخدام الدوال، وكذلك بعض الأخطاء التي قد تحدث عن كتابة الأوامر .

وفي الأخير هذا الكتاب تجربة بسيطة لي يحتوي على بعض الأخطاء الغير مقصودة ، فالإنسان دائماً ما يخطى ، وهذا الكتاب مازال قيد التطوير فمن وجد فيه أخطاء أو لديه آراء فلا يتردد في إبلاغي عبر بريدي الإلكتروني :

mmah_1550@hotmail.com
mmah_1550@yahoo.com

وإن كنت قد أصبت بشيء فهذا من الله ، وإن كنت قد أخطئت في شيء فهذا من نفسي ومن الشيطان .

ولا تنسونا من دعوة صالحة في ظهر الغيب لي ولوالدي ولجميع المسلمين .

ماهر محمد أحمد الرياشي

-2009-

المحتويات

أعداد / ماهر محمد أحمد الرياشي

(sql & sql * plus)

الفصل الأول :

التعريف بـــ sql & sql * plus

أعداد / ماهر محمد أحمد الرياشي

1-1 أقسام ووظائف لغة (sql)

Sql (structure query language)

وتعني الـــ sql : لغة الاستعلام بالترتيب المهيكل ، وهي عبارة عن تعليمات برمجية تستخدم للتعامل مع البيانات .

وتنقسم إلى خمسة أقسام كالتالي :

1- Data Retrieval

و تتضمن أمر الاسترجاع **select** ، والذي يصنف ضمن لغة **DML** ، ولكن نظراً لأهمية هذا الأمر خصص له هذا القسم .

2- DML (Data Manipulating language)

وتختص بالتعامل مع البيانات في الجداول ضمن منطقة العمل التمهيدية (الاحتياطية) .

3- Transection command

وتختص بعملية نقل البيانات بين منطقة العمل التمهيدية ، وقاعدة البيانات .

4- DDL (Data Definition language)

وتختص بتوصيف البيانات ضمن منطقة العمل الأساسية (قاعدة البيانات) ، وتقوم بتنفيذ التعليمات .

5- DCL (Data Control language)

وتختص بمنح وسحب الصلاحيات و أوامرها .

منطقة العمل التمهيدية (الإحتياطية)

هي عبارة عن منطقة توضع فيها نسخ من البيانات المراد إضافتها ، أو تعديلها ، أو إلغاؤها من و إلى قاعدة البيانات .

أهم وظائف لغة (sql) :

- لغة قياسية من لغات الحاسب الخاصة بمعهد (**ANSI**) **American National Standards Institute** .
- تمكنك من الدخول لقاعدة البيانات .
- تمكنك من استخراج البيانات من قاعدة البيانات .
- تمكنك من إضافة بيانات إلى قاعدة البيانات .
- تمكنك من الحذف والتعديل على البيانات الموجود في قاعدة البيانات .

ملاحظات

- لغة (sql) غير حساسة للحروف الصغيرة أو الكبيرة .
- لغة (sql) لا تهتم بالمسافات الفارغة (البيضاء) .
- لغة (sql) تنتهي بفاصلة منقوطة (;) .

2-1 محرر sql * plus

وهو المحرر الخاص بلغة (sql) ، والذي يتم من خلاله تنفيذ أوامر لغة (sql) ، وحفظها واسترجاعها داخل قاعدة البيانات .

ومن أوامرها : ـــ

الأمر	الوصف
CONNECT	الاتصال بقاعدة البيانات .
DISC	لقطع الاتصال بقاعدة البيانات .
SHOW	أمر استعراض للمستخدم أو أوامر بيئة النظام .
GET	استدعاء ملف محفوظ من سابق .
SET	أمر تفعيل و إيقاف وإعداد بيئة SQL * PLUS .
ED	الانتقال للمحرر و تعديل الأمر .
RUN أو R	يكتب الأمر ويعيد تنفيذه .
/	ينفذ الأمر دون إعادة كتابته .
@	لتنفيذ أمر أو ملف محفوظ من سابق .
START	لتنفيذ ملف مكتوب من سابق .
DEL	إلغاء السطر المستعرض أو كل السطور في الأمر .
SAVE FILENAME	لحفظ أمر باسم .
SAVE FILENAME APPEND	لحفظ أمر في ملف سابق (إضافة للملف) .
SAVE FILENAME REPLACE	لحفظ أمر في ملف سابق مع حذف الأوامر .
QUIT	لإيقاف تنفيذ أمر .
EXIT	للخروج من محرر اللغة .

3-1 العمل مع بيئة sql & sql * plus

للبدء في العمل مع لغة (sql) يجب أن تقوم بتحميل قواعد البيانات oracle في أي إصدار من إصداراته على جهازك .

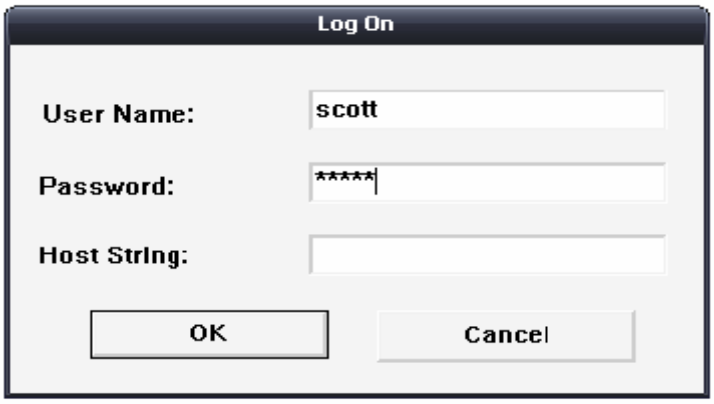
و لتشغيل البرنامج :

من قائمة ابدأ ← كافة البرامج ← oracle – orahome92 ← Application Development ← sql plus .

وستظهر الشاشة التالية حيث يتم كتابة اسم المستخدم ، وكلمة السر الخاصة به ، واسم قاعدة البيانات التي ستعمل عليها .

ملاحظة :

- إذا كنت ستعمل على نفس قاعدة البيانات الحالية ، فليس من الضروري كتابة اسم قاعدة البيانات .
- إذا كنت لم تغير كلمة السر للمستخدمين عند التحميل ، فسوف يكون الدخول لقاعدة البيانات ،كما في الأمثلة المبينة لاحقاً .



إذا كانت البيانات صحيحة ، يتم الدخول إلى الشاشة البيضاء الخاصة بلغة sql ، أما في حالة كتابة البيانات ثلاث مرات متتالية بطريقة خاطئة فإن البرنامج سيغلق تلقائياً .

يتم كتابة الأمر المطلوب ، وبحسب تركيبته ، حيث أن الأوامر تكتب أما استعراض بيانات ، أو إنشاء جداول ، أو ..... الخ .

ولاستعراض كافة الجداول الموجودة في المستخدم نستخدم الأمر التالي :

```
SQL> select * from cat;

TABLE_NAME                     TABLE_TYPE
------------------------------ -----------
BONUS                          TABLE
DEPT                           TABLE
EMP                            TABLE
SALGRADE                       TABLE
```

أو الأمر :

```
SQL> select * from user_catalog;

TABLE_NAME                     TABLE_TYPE
------------------------------ -----------
BONUS                          TABLE
DEPT                           TABLE
EMP                            TABLE
SALGRADE                       TABLE
```

وبعد كتابة الأمر نضغط على زر **ENTER** ، فإذا كان الأمر صحيح (لا يوجد به أخطاء) سوف يقوم بتنفيذ الأمر كما في الأمثلة السابقة ، أما في حالة وجود خطأ فسوف يقوم بإظهار رسالة توضح نوع الخطأ .
و توضع علامة (*) لتشير إلى موقع الخطأ كما في المثال التالي :

```
SQL> select * from user_catlog;
select * from user_catlog
              *
ERROR at line 1:
ORA-00942: table or view does not exist
```

نلاحظ في المثال أننا كتبنا كلمة **catalog** بالطريقة الخاطئة ، لذلك قام بإظهار رسالة خطأ مع توضيح مكان الخطأ ، ولتصحيح الأخطاء في حالة وجودها ، أو لتعديل الأمر نستخدم الأمر التالي :

c / الكلمة المراد تعديلها / الكلمة الجديدة

ثم نضغط مفتاح **ENTER** بدون وضع فاصلة منقوطة .

```
SQL> c/user_catlog/user_catalog
  1* select * from user_catalog
```


أعداد / ماهر محمد أحمد الرياشي

4-1 حفظ الأمر والرجوع إليه عند الحاجة من خلال sql * plus

يتم الحفظ بواسطة الأمر **save**

اسم الملف **save**

```
SQL> save maher;
Created file maher.sql
```

ويحفظ الملف عادة بالامتداد **filename.sql**

وعند الحاجة لاستخدام الأمر مره أخرى يستدعى بالأمر **GET**

اسم الملف **GET**

```
SQL> GET maher;
  1* select * from user_catalog
```

أما لإضافة الأوامر لملف سابق نستخدم الأمر

append اسم الملف **save**

```
SQL> save maher append;
Appended file to maher.sql
```

أما لحفظ الأوامر الجديدة بملف سابق ، وعدم الاحتفاظ بالأوامر السابقة نستخدم الأمر

replace اسم الملف **save**

```
SQL> save maher replace;
Wrote file maher.sql
```

عملية حفظ الأوامر المنفذة مع نتائجها الصحيحة والخاطئة في ملف يتم تسميته بامتداد **sql**. وذلك قبل كتابة وتنفيذ الأوامر المطلوب حفظها ، وذلك بالطريقة التالية :

```
spool  filename.sql فتح وتخزين الملف المؤقت
..... كتابة الأوامر
.....
Spool off          إغلاق ملف التخزين المؤقت
```

ولن يتم الحفظ إلا بعد استخدام **spool off** .

أمر تنظيف الشاشة

```
SQL> clear screen
```

أو الأمر :

```
SQL> clear scr
```

5-1 سماحيات المستخدمين والاتصال بقاعدة البيانات

يوجد في قاعدة البيانات عدد من المستخدمين ويمكن استعراضهم بالأمر :

```
SQL> SELECT * FROM ALL_USERS;

USERNAME                       USER_ID CREATED
------------------------------ ---------- ---------
SYS                                 0 12-MAY-02
SYSTEM                              5 12-MAY-02
OUTLN                              11 12-MAY-02
DBSNMP                             19 12-MAY-02
WMSYS                              21 12-MAY-02
ORDSYS                             30 12-MAY-02
ORDPLUGINS                         31 12-MAY-02
MDSYS                              32 12-MAY-02
CTXSYS                             33 12-MAY-02
XDB                                35 12-MAY-02
ANONYMOUS                          36 12-MAY-02

USERNAME                       USER_ID CREATED
------------------------------ ---------- ---------
WKSYS                              39 12-MAY-02
WKPROXY                            40 12-MAY-02
ODM                                42 12-MAY-02
ODM_MTR                            43 12-MAY-02
OLAPSYS                            44 12-MAY-02
RMAN                               60 12-MAY-02
HR                                 46 12-MAY-02
OE                                 47 12-MAY-02
PM                                 48 12-MAY-02
SH                                 49 12-MAY-02
QS_ADM                             51 12-MAY-02

USERNAME                       USER_ID CREATED
------------------------------ ---------- ---------
QS                                 52 12-MAY-02
QS_WS                              53 12-MAY-02
QS_ES                              54 12-MAY-02
QS_OS                              55 12-MAY-02
QS_CBADM                           56 12-MAY-02
QS_CB                              57 12-MAY-02
QS_CS                              58 12-MAY-02
SCOTT                              59 12-MAY-02

30 rows selected.
```

في المثال السابق تم عرض أسماء المستخدمين ، وأرقامهم ، وتاريخ إنشاءهم .

وهناك عدد من المستخدمين ، ويوجد لكل مستخدم كلمة سر ، وسماحيات خاصة به .

نعرف منهم :

اسم المستخدم	كلمة السر	السماحيات
scott	tiger	Connect and resource
demo	demo	Connect and resource
system	manager	DBA

لمعرفة المستخدم الحالي في قاعدة البيانات نستخدم أحد الأمرين التاليين :

```
SQL> select user from dual;

USER
------------------------------
SCOTT

SQL> show user;
USER is "SCOTT"
```

ولعرض معلومات حول الجداول التي يتملكها المستخدمون في قاعدة البيانات نستخدم الأمر التالي :

```
SQL> select * from user_tables;
```

ويوجد في كل مستخدم دليل خاص به ، يحتوي على أسماء الجداول التي إنشاءها المستخدم ، ولاستعراض الجداول الموجودة في المستخدم نستخدم الأمر التالي :

```
SQL> select * from cat;

TABLE_NAME                     TABLE_TYPE
------------------------------ -----------
BONUS                          TABLE
DEPT                           TABLE
EMP                            TABLE
SALGRADE                       TABLE
```

حيث الـــــ **cat** اختصار لـــ **catalog** والتي تعني الدليل .

وإذا أردنا ربط قاعدة البيانات بمستخدم آخر في نفس صفحة **sql** نستخدم الطرق التالية :

connect كلمة السر / اسم المستخدم

```
SQL> connect system / manager ;
Connected.
```

أو بالطريقة التالية :

```
SQL> connect
Enter user-name: system
Enter password: *******
Connected.
```

وإذا أردنا قطع الاتصال بقاعدة البيانات مع بقاء شاشة المحرر موجودة

نستخدم الأمر **DISC** ثم نضغط الزر **ENTER**

```
SQL> DISC
SQL> select * from cat;
SP2-0640: Not connected
```

نلاحظ في المثال السابق أنه تم قطع الاتصال بقاعدة البيانات ، وعند كتابة أي أمر بعد القطع تظهر رسالة **Not connected** أي غير متصل .

وللخروج النهائي من المحرر نستخدم الأمر :

```
SQL> EXIT
```

6-1 مواصفات وبيانات الجداول

لاستعراض مواصفات الجدول نستخدم الأمر :

اسم الجدول **desc** أو **describe**

```
SQL> desc emp ;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 EMPNO                                     NOT NULL NUMBER(4)
 ENAME                                              VARCHAR2(10)
 JOB                                                VARCHAR2(9)
 MGR                                                NUMBER(4)
 HIREDATE                                           DATE
 SAL                                                NUMBER(7,2)
 COMM                                               NUMBER(7,2)
 DEPTNO                                             NUMBER(2)
```

نلاحظ في المثال السابق أنه تم عرض مواصفات جدول الموظفين (**emp**) الموجود في قاعدة بيانات المستخدم (**scott**) .

```
SQL> desc dept;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 DEPTNO                                    NOT NULL NUMBER(2)
 DNAME                                              VARCHAR2(14)
 LOC                                                VARCHAR2(13)
```

نلاحظ كذلك في المثال السابق أنه تم عرض مواصفات جدول الأقسام (**dept**) الموجود في قاعدة بيانات المستخدم (**scott**) .

```
SQL> describe dept;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 DEPTNO                                    NOT NULL NUMBER(2)
 DNAME                                              VARCHAR2(14)
 LOC                                                VARCHAR2(13)
```

ولاستعراض بيانات الجدول نستخدم الأمر :

اسم الجدول **select * from**

```
SQL> select * from emp;

     EMPNO ENAME      JOB              MGR HIREDATE         SAL       COMM     DEPTNO
---------- ---------- --------- ---------- --------- ---------- ---------- ----------
      7369 SMITH      CLERK           7902 17-DEC-80        800                    20
      7499 ALLEN      SALESMAN        7698 20-FEB-81       1600        300         30
      7521 WARD       SALESMAN        7698 22-FEB-81       1250        500         30
      7566 JONES      MANAGER         7839 02-APR-81       2975                    20
      7654 MARTIN     SALESMAN        7698 28-SEP-81       1250       1400         30
      7698 BLAKE      MANAGER         7839 01-MAY-81       2850                    30
      7782 CLARK      MANAGER         7839 09-JUN-81       2450                    10
      7788 SCOTT      ANALYST         7566 19-APR-87       3000                    20
      7839 KING       PRESIDENT            17-NOV-81       5000                    10
      7844 TURNER     SALESMAN        7698 08-SEP-81       1500          0         30
      7876 ADAMS      CLERK           7788 23-MAY-87       1100                    20

     EMPNO ENAME      JOB              MGR HIREDATE         SAL       COMM     DEPTNO
---------- ---------- --------- ---------- --------- ---------- ---------- ----------
      7900 JAMES      CLERK           7698 03-DEC-81        950                    30
      7902 FORD       ANALYST         7566 03-DEC-81       3000                    20
      7934 MILLER     CLERK           7782 23-JAN-82       1300                    10

14 rows selected.
```

نلاحظ في المثال السابق أنه تم عرض بيانات جدول الموظفين (**emp**) .

```
SQL> select * from dept;

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

نلاحظ أيضاً في المثال السابق تم عرض بيانات جدول الأقسام (**dept**) .

وبنفس الطريقة يمكن عرض مواصفات ، وبيانات الجداول الباقية في قاعدة البيانات .

وسوف أقوم بشرح محتويات الجدول (**emp**) ، والجدول (**dept**) ، وذلك لأننا سنستخدمهم كثيراً في أمثلتنا في الفصول القادمة :

```
SQL> select * from dept;

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

الحقل (**deptno**) : و يحتوي على الأقسام .

الحقل (**dname**) : و يحتوي على أسماء الأقسام .

الحقل (**loc**) : ويحتوي على موقع الأقسام .

```
SQL> select * from emp;

     EMPNO ENAME      JOB              MGR HIREDATE         SAL       COMM     DEPTNO
---------- ---------- --------- ---------- --------- ---------- ---------- ----------
      7369 SMITH      CLERK           7902 17-DEC-80        800                    20
      7499 ALLEN      SALESMAN        7698 20-FEB-81       1600        300         30
      7521 WARD       SALESMAN        7698 22-FEB-81       1250        500         30
      7566 JONES      MANAGER         7839 02-APR-81       2975                    20
      7654 MARTIN     SALESMAN        7698 28-SEP-81       1250       1400         30
      7698 BLAKE      MANAGER         7839 01-MAY-81       2850                    30
      7782 CLARK      MANAGER         7839 09-JUN-81       2450                    10
      7788 SCOTT      ANALYST         7566 19-APR-87       3000                    20
      7839 KING       PRESIDENT            17-NOV-81       5000                    10
      7844 TURNER     SALESMAN        7698 08-SEP-81       1500          0         30
      7876 ADAMS      CLERK           7788 23-MAY-87       1100                    20

     EMPNO ENAME      JOB              MGR HIREDATE         SAL       COMM     DEPTNO
---------- ---------- --------- ---------- --------- ---------- ---------- ----------
      7900 JAMES      CLERK           7698 03-DEC-81        950                    30
      7902 FORD       ANALYST         7566 03-DEC-81       3000                    20
      7934 MILLER     CLERK           7782 23-JAN-82       1300                    10

14 rows selected.
```

الحقل (**empno**) : ويحتوي على رقم الموظف .

الحقل (**ename**) : ويحتوي على اسم الموظف .

الحقل (**job**) : ويحتوي على الوظيفة .

الحقل (**mgr**) : ويحتوي على رقم مدير الموظف .

الحقل (**hiredate**) : ويحتوي على تاريخ التوظيف .

الحقل (**sal**) : ويحتوي على راتب الموظف .

الحقل (**comm**) : ويحتوي على عمولة الموظف .

الحقل (**deptno**) : ويحتوي على القسم الذي يتبعه الموظف .

– الجدول (dual) :

وهو جدول موجود داخل قاعدة البيانات ، وهو جدول وهمي حيث يتكون من حقل واحد من نوع حرفي طويل ، وبطول حرف واحد ويستخدم لإجراء العمليات التي ليست لها جدول أساسي مثل : دوال التاريخ ، المستخدم الحالي

```
SQL> desc dual;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 DUMMY                                              VARCHAR2(1)
```

1- 7 تمارين الفصل الأول

في هذا الفصل لا توجد تمارين

(sql & sql * plus)

الفصل الثاني :

جملة الاستعلام select

أعداد / ماهر محمد أحمد الرياشي

جملة الاستعلام select :

الشكل العام لجملة الاستعلام **select** هو :

```
select * or columns , aliases
from view or table
where condition
order by columns
```

(*) تعني جميع الحقول .

ملاحظات

– تكتب جملة **select** بالحروف الكبيرة أو الصغيرة .

– تفصل بين كل حقل وآخر علامة الفاصلة (,) .

2-1استرجاع الحقول بأسماء مستعارة

ونستخدم لذلك المعامل **as** أو المسافة كما في المثال التالي :

```
SQL> select ename as name
  2  from emp;

NAME
----------
SMITH
ALLEN
WARD
JONES
MARTIN
BLAKE
CLARK
SCOTT
KING
TURNER
ADAMS

NAME
----------
JAMES
FORD
MILLER

14 rows selected.
```

أو يمكن كتابة الأمر كالتالي :

```
SQL> select ename "name"
  2  from emp;

name
----------
SMITH
ALLEN
WARD
JONES
MARTIN
BLAKE
CLARK
SCOTT
KING
TURNER
ADAMS

name
----------
JAMES
FORD
MILLER

14 rows selected.
```

في المثال الأول قام باستعراض بيانات حقل (**ename**) تحت مسمى (**name**) باستخدام as ، وفي المثال الثاني قام بنفس العمل ولكن باستخدام المسافة .

الأخطاء :

```
SQL> select ename , as name
  2  form emp;
select ename , as name
               *
ERROR at line 1:
ORA-00936: missing expression
```

في المثال السابق ظهرت رسالة خطأ وذلك بسبب كتابة الفاصلة بين اسم الحقل والـ **as** والمفروض عدم كتابه الفاصلة .

```
SQL> select ename 'name'
  2  from emp;
select ename 'name'
             *
ERROR at line 1:
ORA-00923: FROM keyword not found where expected
```

في المثال السابق ظهرت رسالة خطأ ، وذلك بسبب كتابة النص داخل تنصيص مفرد ، والمفروض كتابة النصوص داخل تنصيص مزدوج .

2-2 استرجاع الحقول باستخدام العمليات الحسابية الأربع

يمكن استخدام العمليات الحسابية في جميع أجزاء جملة **sql** ، ماعدا الجزء الخاص بـــ **from** ، وأولويات التنفيذ الضرب والقسمة ، ثم الجمع والطرح .

مثال :

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

name           SALARY  annual salary
---------- ---------- ---------------
SMITH             800            9600
ALLEN            1600           19200
WARD             1250           15000
JONES            2975           35700
MARTIN           1250           15000
BLAKE            2850           34200
CLARK            2450           29400
SCOTT            3000           36000
KING             5000           60000
TURNER           1500           18000
ADAMS            1100           13200

name           SALARY  annual salary
---------- ---------- ---------------
JAMES             950           11400
FORD             3000           36000
MILLER           1300           15600

14 rows selected.
```

في المثال السابق تم عرض حقل (**ename**) تحت مسمى (**name**) ، وكذلك حقل (**sal**) تحت مسمى (**salary**) ، وكذلك قام بعملية حسابية حيث قام بحساب الراتب السنوي ، وذلك بضرب حقل الراتب (**sal**) في **12** ووضعه تحت مسمى (**annual salary**) أي الراتب السنوي .

مثال اخر :

```
SQL> select ename as name ,comm , comm / 4
  2  from emp;

NAME              COMM     COMM/4
---------- ---------- ----------
SMITH
ALLEN             300         75
WARD              500        125
JONES
MARTIN           1400        350
BLAKE
CLARK
SCOTT
KING
TURNER              0          0
ADAMS

NAME              COMM     COMM/4
---------- ---------- ----------
JAMES
FORD
MILLER

14 rows selected.
```

في المثال السابق تم عرض حقل (**ename**) تحت مسمى (**name**) ، وكذلك حقل العمولة (**comm**) ، وكذلك قام بعملية حسابية حيث قام بحساب العمولة مقسومة على أربعة .

3-2 استخدام الربط (||) بين الحقول

تستخدم الأداة (||) للربط بين الحقول كما في المثال التالي :

```
SQL> select ename || job as " employees "
  2  from emp;

 employees
--------------------
SMITHCLERK
ALLENSALESMAN
WARDSALESMAN
JONESMANAGER
MARTINSALESMAN
BLAKEMANAGER
CLARKMANAGER
SCOTTANALYST
KINGPRESIDENT
TURNERSALESMAN
ADAMSCLERK

 employees
--------------------
JAMESCLERK
FORDANALYST
MILLERCLERK

14 rows selected.
```

نلاحظ في المثال السابق أنه تم الربط بين حقل الاسم (**ename**) ، وحقل الوظيفة (**job**) تحت مسمى (**employees**) .

أعداد / ماهر محمد أحمد الرياشي

```
SQL> select (sal || comm ) * 10
  2  from emp;

(SAL||COMM)*10
--------------
          8000
      16003000
      12505000
         29750
     125014000
         28500
         24500
         30000
         50000
        150000
         11000

(SAL||COMM)*10
--------------
          9500
         30000
         13000

14 rows selected.
```

في المثال السابق تم الربط بين حقل الراتب (**sal**) ، وحقل العمولة (**comm**) ثم بعد الربط تم الضرب في 10

الأخطاء:

```
SQL> select sal || * comm as " total salary "
  2  from emp;
select sal || * comm as " total salary "
              *
ERROR at line 1:
ORA-00936: missing expression
```

في المثال السابق ظهرت رسالة خطأ ، وهو عدم إمكانية ربط حقل الراتب (**sal**) مع حقل العمولة (**comm**) مضروباً به .

4-2 استخدام (distinct) لمنع تكرار السجلات

يستخدم الأمر **distinct** لمنع تكرار السجلات (أي تكرار البيانات) .

مثال :

```
SQL> select deptno from emp;

    DEPTNO
----------
        20
        30
        30
        20
        30
        30
        10
        20
        10
        30
        20

    DEPTNO
----------
        30
        20
        10

14 rows selected.
```

في المثال السابق نلاحظ أن السجلات مكررة يعني يوجد الرقم **10** أكثر من مره ، وكذلك الرقم **20** و **30** ،

ولكن عند استخدام **distinct** نلاحظ عدم تكرار السجلات كما في المثال التالي :

```
SQL> select distinct deptno
  2  from emp;

    DEPTNO
----------
        10
        20
        30
```

الأخطاء:

```
SQL> select deptno distinct
  2  from emp;
select deptno distinct
              *
ERROR at line 1:
ORA-00923: FROM keyword not found where expected
```

في المثال السابق ظهرت رسالة خطأ وذلك بسبب تأخير كلمة **distinct** بعد الحقل وهي من المفروض أن تأتي قبل الحقل .

أعداد / ماهر محمد أحمد الرياشي

5-2 جملة الشرط where

وهي ضمن جملة الاستعلام **select** ، وتستخدم لحصر الصفوف المستدعاة على أساس شرط أو شروط معينة ، فإذا كان الشرط صحيح نحصل على ناتج الشرط ، أما إذا كان الشرط خاطئ تظهر الرسالة التالية :

```
no rows selected
```

أي انه لم يتم الحصول على أي صف .

وتتكون جملة الشرط **where** من أسماء حقول أو أسماء مستعارة للحقول ، ومعاملات مقارنة ، وقيم ثابتة قد تكون رقمية أو نصية ، وكذلك تعبيرات حسابية .

ويجب مراعاة ما يلي :

1- علامة التنصيص الفردية (' ') في حالة استخدام قيم نصية أو تاريخ .

2- حالة الأحرف كبيرة أو صغيرة بين علامتي التنصيص .

3- مراعاة شكل التاريخ المستخدم وفي **sql** يستخدم الشكل **DD – MON – YY** .

حيث :

DD هو اليوم : 09 .

MON هو الشهر : SEP اختصار لشهر سبتمبر .

YY هو السنة : 09 يعني 2009 .

جدول يوضح المعاملات المستخدمة مع جملة where

الوصـــــف	المعامـــــل
معاملات المقارنة .	= , > , < , > = , < = , < > , ! =
تستخدم لمقارنة قيمة أو أكثر بعدة قيم داخل القائمة.	in , any , all , < any , < all , > all
استدعاء البيانات بين قيمتين .	between value and value
استدعاء البيانات حسب مطابقة النص أو الحرف .	like (' %_')
استدعاء البيانات حسب الاسم الصوتي .	soundex (' excerption ')
استدعاء البيانات الخالية .	is null
المعاملات المنطقية .	and , or , not

ويمكن كذلك استخدام **not** كالتالي :

not between
not like
not in
is not null

أمثلة لاستخدام جملة الشرط (**where**) وبعض المعاملات :

```
SQL> select ename , job
  2  from emp
  3  where deptno = 10 ;

ENAME      JOB
---------- ---------
CLARK      MANAGER
KING       PRESIDENT
MILLER     CLERK
```

نلاحظ في المثال السابق أنه تم عرض حقل الاسم (**ename**) ، وحقل الوظيفة (**job**) للموظفين الذين يعملون في القسم (**deptno**) رقم **10** .

```
SQL> select ename , sal
  2  from emp
  3  where sal < = 2000 ;

ENAME             SAL
---------- ----------
SMITH             800
ALLEN            1600
WARD             1250
MARTIN           1250
TURNER           1500
ADAMS            1100
JAMES             950
MILLER           1300

8 rows selected.
```

نلاحظ في المثال السابق أنه تم عرض حقل الاسم (**ename**) ، وحقل الراتب (**sal**) للموظفين فقط الذين رواتبهم أقل أو يساوي **2000** .

```
SQL> select ename , sal
  2  from emp
  3  where sal between 2000 and 4000 ;

ENAME             SAL
---------- ----------
JONES            2975
BLAKE            2850
CLARK            2450
SCOTT            3000
FORD             3000
```

نلاحظ في المثال السابق أنه تم عرض حقل الاسم (ename) ، وحقل الراتب (sal) للموظفين فقط الذين رواتبهم بين 2000 و الـ 4000 .

المعامل like

يستخدم للبحث عن معنى معين داخل ثابت أو حقل نصي ، حيث أن العلامة % تعني أي حرف أو أحرف .

(' m%') تعني النصوص التي تبدأ بحرف m .

(' %m% ') تعني النصوص التي تحتوي على حرف m .

(' _m% ') تعني النصوص التي حرفها الثاني m .

(' __m% ') تعني النصوص التي حرفها الثالث m .

وهكذا .....

أمثلة :

```
SQL> select ename
  2  from emp
  3  where ename like 'M%';

ENAME
----------
MARTIN
MILLER
```

نلاحظ في المثال السابق انه تم عرض أسماء الموظفين الذين تبدأ أسماءهم بحرف M .

```
SQL> select ename
  2  from emp
  3  where ename like '_M%';

ENAME
----------
SMITH
```

نلاحظ في المثال السابق انه تم عرض أسماء الموظفين الذين حرفهم الثاني هو M .

الأخطاء :

```
SQL> select ename
  2  from emp
  3  where ename like 'm%';

no rows selected
```

هو في الواقع ليس خطأ ، ولكن نلاحظ في المثال السابق انه ظهرت رسالة بعدم وجود ما يطابق الشرط ، وذلك لكتابة حرف M بالحرف الصغير وهنا بما أن النص بين علامة التنصيص يجب مراعاة الحروف الصغيرة والكبيرة .

المعامل soundex

يستخدم للبحث عن النصوص التي تشبه نبرات الصوت أو نبرات الأحرف .

مثال :

```
SQL> select ename from emp
  2  where soundex(ename) = soundex ( 'SMOTH ' ) ;

ENAME
----------
SMITH
```

في المثال السابق نلاحظ إن نتيجة البحث عن (SMOTH) هو SMITH ، وذلك لوجود تشابه في نبرات الأحرف والصوت .

وفي المثال التالي كذلك هناك تشابه في نبرات الأحرف والصوت :

```
SQL> select job from emp
  2  where soundex(job) = soundex ( ' CLARK ' );

JOB
---------
CLERK
CLERK
CLERK
CLERK
```

وفي حالة عدم وجود تشابه في نبرات الأحرف والصوت تظهر رسالة (no rows selected)

أي لم يتم تحديد صفوف كالمثال التالي :

```
SQL> select ename from emp
  2  where soundex ( ename ) = soundex (' MAHER ' );

no rows selected
```

المعامل and

ويأخذ المعامل (and) أربع حالات كما في الجدول التالي :

جملة الشرط	جملة الشرط	ناتج الشرط
true	true	true
true	false	false
false	true	false
false	false	false

مثال :

```
SQL> select ename , sal , job
  2  from emp
  3  where sal > = 2000
  4  and deptno = 10 ;

ENAME             SAL JOB
---------- ---------- ---------
CLARK            2450 MANAGER
KING             5000 PRESIDENT
```

نلاحظ في المثال السابق أنه تم حصر الموظفين الذين رواتبهم اكبر من أو يساوي **2000** ، و يعملون في القسم رقم **10** .

(إذا كان أحد الشروط خطأ لا توجد نتائج وتوجد نتائج فقط إذا تحققت كل الشروط) .

```
SQL> select ename , sal , job
  2  from emp
  3  where sal > = 2000
  4  and deptno = 40 ;

no rows selected
```

نلاحظ في المثال السابق أنه تحقق الشرط الأول ، ولكن لم يتحقق الشرط الثاني فلم تظهر أي نتائج .

المعامل or

ويأخذ المعامل (**or**) أربع حالات كما في الجدول التالي :

جملة الشرط	جملة الشرط	ناتج الشرط
true	true	true
true	false	true
true	true	false
false	false	false

مثال :

```
SQL> select ename , sal , job
  2  from emp
  3  where sal > = 2000
  4  or deptno = 10 ;

ENAME             SAL JOB
---------- ---------- ---------
JONES            2975 MANAGER
BLAKE            2850 MANAGER
CLARK            2450 MANAGER
SCOTT            3000 ANALYST
KING             5000 PRESIDENT
FORD             3000 ANALYST
MILLER           1300 CLERK

7 rows selected.
```

نلاحظ في المثال السابق أنه تم حصر الموظفين الذين رواتبهم أكبر من أو يساوي **2000** أو يعملون في القسم رقم **10**

(إذا كان أحد الشروط صحيح أو كلاهما إذاً يوجد نتائج أما إذا كان كل الشروط خاطئة فلا توجد نتائج) .

```
SQL> select ename , sal , job
  2  from emp
  3  where sal > = 6600
  4  or deptno = 40;

no rows selected
```

نلاحظ في المثال السابق أن كل الشروط لم تتحقق ، حيث أنه لا يوجد موظفين في جدول (**emp**) رواتبهم أكبر أو تساوي **6600** وكذلك لا يوجد موظفين يتبعون القسم رقم **40** لذلك لم تظهر أي نتائج .

المعامل not

ويأخذ المعامل (**not**) حالتان كما في الجدول التالي :

جملة الشرط	ناتج الشرط
true	false
false	true

مثال :

```
SQL> select ename from emp
  2  where not soundex(ename) = soundex ('JENES ' );

ENAME
----------
SMITH
ALLEN
WARD
MARTIN
BLAKE
CLARK
SCOTT
KING
TURNER
ADAMS
FORD

ENAME
----------
MILLER

12 rows selected.
```

نلاحظ في المثال السابق أنه تم عرض جميع الموظفين ، ماعدا الموظفين التي نبرات الأحرف والصوت تشابه (**JENES**) وهم (**JAMES**) و (**JONES**) .

مثال آخر :

```
SQL> select ename , sal
  2  from emp
  3  where sal not between 1000 and 3000 ;

ENAME             SAL
---------- ----------
SMITH             800
KING             5000
JAMES             950
```

نلاحظ في المثال السابق أنه تم حصر الموظفين الذين رواتبهم ليست بين **1000** و **3000** .

6-2 جملة الترتيب order by

تستخدم لترتيب الصفوف الناتجة إما تصاعدياً **ascending** ، أو تنازلياً **descending** ، وتكتب في نهاية جملة **select** .

مثال :

```
SQL> select ename , job ,deptno
  2  from emp
  3  order by deptno ;

ENAME      JOB            DEPTNO
---------- --------- ----------
CLARK      MANAGER            10
KING       PRESIDENT          10
MILLER     CLERK              10
SMITH      CLERK              20
ADAMS      CLERK              20
FORD       ANALYST            20
SCOTT      ANALYST            20
JONES      MANAGER            20
ALLEN      SALESMAN           30
BLAKE      MANAGER            30
MARTIN     SALESMAN           30

ENAME      JOB            DEPTNO
---------- --------- ----------
JAMES      CLERK              30
TURNER     SALESMAN           30
WARD       SALESMAN           30

14 rows selected.
```

نلاحظ في المثال السابق أن الترتيب كان على حسب الأقسام (**deptno**) ، وإن الترتيب كان تصاعدياً (وذلك لأنه الوضع الافتراضي) .

أعداد / ماهر محمد أحمد الرياشي

مثال أخر :

```
SQL> select ename , job , deptno , hiredate
  2  from emp
  3  where deptno = 10
  4  order by hiredate;

ENAME      JOB           DEPTNO HIREDATE
---------- --------- ---------- ---------
CLARK      MANAGER           10 09-JUN-81
KING       PRESIDENT         10 17-NOV-81
MILLER     CLERK             10 23-JAN-82
```

نلاحظ في المثال السابق أن تم عرض بيانات الاسم والوظيفة والقسم وتاريخ التوظيف للموظفين الذين يعملون في القسم رقم **10** مرتباً تصاعدياً بحسب تاريخ التوظيف (**hiredate**) .

وفي حالة أننا نريد أن نرتب ترتيباً تنازلياً نستخدم الأمر **desc** كما في المثال التالي :

```
SQL> select ename , job , deptno
  2  from emp
  3  order by deptno desc;

ENAME      JOB           DEPTNO
---------- --------- ----------
ALLEN      SALESMAN          30
WARD       SALESMAN          30
MARTIN     SALESMAN          30
JAMES      CLERK             30
TURNER     SALESMAN          30
BLAKE      MANAGER           30
SMITH      CLERK             20
FORD       ANALYST           20
ADAMS      CLERK             20
JONES      MANAGER           20
SCOTT      ANALYST           20

ENAME      JOB           DEPTNO
---------- --------- ----------
CLARK      MANAGER           10
KING       PRESIDENT         10
MILLER     CLERK             10

14 rows selected.
```

نلاحظ في المثال السابق أن الترتيب كان على حسب الأقسام (**deptno**) ، وإن الترتيب كان تنازلياً ، وذلك لاستخدام الأمر **desc** الذي يقوم بعكس الترتيب الافتراضي (تصاعدي) .

مثال أخر :

```
SQL> select ename , job , deptno , sal
  2  from emp
  3  order by deptno , sal desc ;

ENAME      JOB           DEPTNO        SAL
---------- --------- ---------- ----------
KING       PRESIDENT          10       5000
CLARK      MANAGER            10       2450
MILLER     CLERK              10       1300
SCOTT      ANALYST            20       3000
FORD       ANALYST            20       3000
JONES      MANAGER            20       2975
ADAMS      CLERK              20       1100
SMITH      CLERK              20        800
BLAKE      MANAGER            30       2850
ALLEN      SALESMAN           30       1600
TURNER     SALESMAN           30       1500

ENAME      JOB           DEPTNO        SAL
---------- --------- ---------- ----------
WARD       SALESMAN           30       1250
MARTIN     SALESMAN           30       1250
JAMES      CLERK              30        950

14 rows selected.
```

نلاحظ في المثال السابق أنه يمكننا الترتيب بأكثر من حقل ، ففي المثال تم الترتيب بحسب القسم (**deptno**) و بحسب الراتب (**sal**) تنازلياً ، حيث يقوم بالترتيب أولاً بحسب القسم ثم ترتيب كل قسم بحسب الراتب .

7-2 دوال (sql functions)

1- دوال الصف الواحد single row function

2- الدوال التجميعية multiple row function

1- دوال الصف الواحد single row function : تعطي قيمة واحدة من صف واحد .

ومنها :

أعداد / ماهر محمد أحمد الرياشي

1- دوال حرفية :

دوال حرفية	
الدالة	الوصف
lower	التحويل إلى حروف صغيرة .
upper	التحويل إلى حروف كبيرة .
initcap	تحويل الحرف الأول كبير والباقي صغير .
concat	وصل السلاسل مع بعضها البعض .
substr	قطع السلاسل بطول معين .
length	إيجاد عدد حروف السلسلة .
instr	موقع حرف معين في السلسلة .
lpad	ضبط الحروف من جهة اليسار بملئ الفراغ بحرف معين .
rpad	ضبط الحروف من جهة اليمين بملئ الفراغ بحرف معين .
trim	قطع حرف معين من بداية أو نهاية الكلمة فقط .
ltrim	قطع حروف من يسار السلسلة .
rtrim	قطع حروف من يمين السلسلة .

2- دوال رقمية :

دوال رقمية	
الدالة	الوصف
round	دالة التقريب .
trunc	دالة القص دون تقريب .
mod	باقي القسمة .

3- دوال التاريخ:

دوال التاريخ	
الدالة	الوصف
sysdate	التاريخ الحالي .
months_between	عدد الأشهر بين تاريخين .
add_months	إضافة أشهر للتاريخ المعطى .
next_day	اليوم التالي .
last_day	آخر يوم من الشهر .
round	دالة التقريب للتواريخ .
trunc	دالة القص دون تقريب .
new_time	دالة المنطقة الزمنية .

4- دوال التحويل :

دوال التحويل الأساسية	
الدالة	الوصف
to_char	التحويل إلى حرف .
to_number	التحويل إلى رقم .
to_date	التحويل إلى تاريخ .

أمثلة للدوال الحرفية :

وتأتي مع حقول جملة الاستعلام **select** أو مع جملة الشرط **where** .

1- lower (التحويل إلى حروف صغيرة) :

```
SQL> select lower ('MAHER')
  2  from dual;

LOWER
-----
maher
```

2- upper (التحويل إلى حروف كبيرة) :

```
SQL> select upper ('maher')
  2  from dual;

UPPER
-----
MAHER
```

3- initcap (تحويل الحرف الأول كبير والباقي صغير) :

```
SQL> select initcap ('maher')
  2  from dual;

INITC
-----
Maher
```

ملاحظة

- يمكن استخدام أكثر من دالة في نفس جملة الاستعلام .

مثال :

```
SQL> select lower ('MAHER'), upper ('maher')
  2  from dual;

LOWER UPPER
----- -----
maher MAHER
```

4- concat (وصل السلاسل) :

```
SQL> select concat ('ma','her')
  2  from dual;

CONCA
-----
maher
```

5- substr (قطع السلاسل):

تقوم بعملية قطع السلاسل ، ولكن بطول معين .

```
SQL> select substr ('maher',1,3)
  2  from dual;

SUB
---
mah
```

نلاحظ في المثال السابق أنه تم قطع السلسلة على حسب ما تم تحديده ، ففي المثال تم تحديد من الحرف الأول إلى الحرف الثالث .

6- length (إيجاد عدد حروف السلسة) :

```
SQL> select length('maher')
  2  from dual;

LENGTH('MAHER')
---------------
              5
```

7- instr (إيجاد موقع حرف معين في السلسلة) :

```
SQL> select instr('maher','a')
  2  from dual;

INSTR('MAHER','A')
------------------
                 2
```

في المثال السابق يتم تحديد السلسلة ، وكذلك الحرف الذي يراد معرفة موقعه في السلسلة .

8- lpad (لضبط الحروف من جهة اليسار بملئ الفراغ برمز معين) :

```
SQL> select lpad('maher',8,'#')
  2  from dual;

LPAD('MA
--------
###maher
```

في المثال السابق تم تحديد عدد الخانات يساوي **8** ، وبما أن عدد حروف الاسم (**maher**) يتكون من خمسة حروف تم ملئ الفراغ من جهة اليسار بالرمز (#) .

مثال أخر :

```
SQL> select lpad('maher',10,'*')
  2  from dual;

LPAD('MAHE
----------
*****maher
```

في المثال السابق تم تحديد عدد الخانات يساوي **10** ، وبما أن عدد حروف الاسم (**maher**) يتكون من خمسة حروف تم ملئ الفراغ من جهة اليسار بالرمز (*) .

9- rpad (لضبط الحروف من جهة اليمين بملئ الفراغ برمز معين) :

```
SQL> select rpad('maher',8,'#')
  2  from dual;

RPAD('MA
--------
maher###
```

في المثال السابق تم تحديد عدد الخانات يساوي **8** ، وبما أن عدد حروف الاسم (**maher**) يتكون من خمسة حروف تم ملئ الفراغ من جهة اليمين بالرمز (#) .

10- trim (لقطع حروف واحد فقط من بداية أو نهاية الكلمة) :

```
SQL> select trim ('M' from 'MAHER')
  2  from dual;

TRIM
----
AHER
```

في المثال السابق تم قطع حرف واحد من بداية الكلمة ، ونلاحظ أننا عندما نقطع نكتب الحرف المراد قطعه بين علامتي تنصيص مفرده ، ونستخدم الكلمة (**from**) ، ثم الكلمة المراد القطع منها ، وكذلك تكتب بين علامتي تنصيص مفرده .

مثال أخر :

```
SQL> select trim ('R' from 'MAHER')
  2  from dual;

TRIM
----
MAHE
```

في المثال السابق تم قطع حرف واحد من نهاية الكلمة .

```
SQL> select trim ('H' from 'MAHER')
  2  from dual;

TRIM(
-----
MAHER
```

نلاحظ في المثال السابق أننا قمنا باختيار حرف يقع في وسط الكلمة ، ولكن لم يتم حذفه وذلك لأن الدالة (**trim**) تحذف فقط حرف واحد من بداية أو نهاية الكلمة .

11- rtrim (لقطع حروف أو حروف من جهة اليمين) :

```
SQL> select rtrim ('maher','er')
  2  from dual;

RTR
---
mah
```

في المثال السابق تم قطع حرفين من جهة اليمين ، ونلاحظ أننا لم نستخدم الكلمة (**from**) .

12- ltrim (لقطع حروف أو حروف من جهة اليسار) :

```
SQL> select ltrim ('maher','mah')
  2  from dual;

LT
--
er
```

في المثال السابق تم قطع ثلاثة حروف من جهة اليسار .

أمثلة للدوال الرقمية :

وتأتي مع حقول جملة الاستعلام **select** أو مع جملة الشرط **where** .

1- round (دالة التقريب) :

```
SQL> select round(15.734,2)
  2  from dual;

ROUND(15.734,2)
---------------
          15.73
```

في المثال السابق تم تقريب العدد إلى رقمين عشريين بحسب عدد الخانات المعطاة .

مثال أخر :

```
SQL> select round(15.734,0)
  2  from dual;

ROUND(15.734,0)
---------------
             16
```

في المثال السابق تم تقريب العدد بدون رقم عشري ، وذلك بوضعه يساوي **0** .

مثال أخر :

```
SQL> select round(15.734,-1)
  2  from dual;

ROUND(15.734,-1)
----------------
              20
```

في المثال السابق تم وضع القيمة تساوي (**1–**) ، فيتم التقريب إلى عدد صحيح .

2– trunc (دالة القص دون التقريب) :

```
SQL> select trunc (15.734,2)
  2  from dual;

TRUNC(15.734,2)
---------------
          15.73
```

في المثال السابق تم قص العدد إلى رقمين عشريين بحسب عدد الخانات المعطاة .

مثال أخر :

```
SQL> select trunc (15.734,0)
  2  from dual;

TRUNC(15.734,0)
---------------
             15
```

في المثال السابق تم قص الرقم العشري .

مثال أخر :

```
SQL> select trunc (15.734,-1)
  2  from dual;

TRUNC(15.734,-1)
----------------
              10
```

في المثال السابق تم القص إلى أقرب عدد صحيح .

3- mod (تقوم بإيجاد باقي القسمة) :

```
SQL> select ename , sal , comm , mod(sal,comm)
  2  from emp
  3  where ename = 'WARD';

ENAME             SAL       COMM MOD(SAL,COMM)
---------- ---------- ---------- -------------
WARD              1250        500           250
```

في المثال السابق تم احتساب باقي قسمة الراتب (**sal**) على العمولة (**comm**) .

أمثلة لدوال التاريخ :

1- sysdate (يعطي التاريخ الحالي للنظام) :

```
SQL> select sysdate from dual;

SYSDATE
---------
23-JUL-09
```

في المثال السابق أعطى التاريخ الحالي الموجود في النظام .

2- month_between (لمعرفة عدد الأشهر بين التاريخين المعطيين) :

```
SQL> select months_between ( '18-nov-89','09-may-89')
  2  from dual;

MONTHS_BETWEEN('18-NOV-89','09-MAY-89')
---------------------------------------
                             6.29032258
```

في المثال السابق تم احتساب عدد الأشهر بين تاريخين معطيين ، ونلاحظ أن النتيجة كانت بالشهر ، وأجزاء من الشهر .

الأخطاء:

```
SQL> select months_between ('09-may-89','18-nov-89')
  2  from dual;

MONTHS_BETWEEN('09-MAY-89','18-NOV-89')
---------------------------------------
                            -6.2903226
```

في المثال السابق نلاحظ أن النتيجة كانت بالسالب ، وذلك لوضع التاريخ الأصغر قبل التاريخ الأكبر ، ويجب أن يكون التاريخ الأكبر هو الأول .

3- add_months (لإضافة عدد من الأشهر للتاريخ المعطى) :

```
SQL> select sysdate, add_months ( sysdate , 4 )
  2  from dual;

SYSDATE   ADD_MONTH
--------- ---------
23-JUL-09 23-NOV-09
```

في المثال السابق تم إضافة أربعه أشهر لتاريخ النظام الحالي .

4- next_day (لمعرفة تاريخ يوم من أيام الأسبوع وذلك باستخدام تاريخ النظام) :

```
SQL> select sysdate , next_day (sysdate , 'SUNDAY')
  2  from dual;

SYSDATE   NEXT_DAY(
--------- ---------
23-JUL-09 26-JUL-09
```

في المثال السابق تم عرض اليوم الحالي من النظام ، وكم يصادف تاريخ يوم الأحد .

الأخطاء:

```
SQL> select sysdate , next_day ( sysdate , ' sunday ' )
  2  from dual;
select sysdate , next_day ( sysdate , ' sunday ' )
                                      *
ERROR at line 1:
ORA-01846: not a valid day of the week
```

في المثال السابق ظهرت رسالة الخطأ بسبب كتابة يوم الأحد بالحروف الصغيرة ، ويجب كتابتها بالحروف الكبيرة .

5- last_day (تستخدم لمعرفة أخر يوم من الشهر في التاريخ المعطى) :

```
SQL> select last_day ('18-nov-89')
  2  from dual;

LAST_DAY(
---------
30-NOV-89
```

في المثال السابق تم عرض اليوم الأخير من التاريخ المعطى .

6- round (دالة تقريب التواريخ) :

```
SQL> select round (to_date ('18-nov-89'),'MONTH')
  2  from dual;

ROUND(TO_
---------
01-DEC-89
```

في المثال السابق تم تقريب التاريخ إلى الشهر التالي ، وذلك لان اليوم اكبر من النصف ، وإذا كان اليوم أقل من النصف ، فإن التقريب يكون إلى بداية نفس الشهر كما في المثال التالي :

```
SQL> select round (to_date ('09-may-89'), 'MONTH')
  2  from dual;

ROUND(TO_
---------
01-MAY-89
```

7- trunc (دالة القص للتواريخ) :

وهي دالة قص بمعنى تحول التاريخ المعطى إلى بداية الشهر أو العام من نفس الشهر ، وسنة التاريخ المعطى .

```
SQL> select trunc ( to_date ('18-nov-89'),'MONTH')
  2  from dual;

TRUNC(TO_
---------
01-NOV-89
```

في المثال السابق تم قص التاريخ المعطى إلى بداية الشهر .

مثال أخر :

```
SQL> select trunc (to_date ('18-nov-89') , 'YEAR')
  2  from dual;

TRUNC(TO_
---------
01-JAN-89
```

في المثال السابق تم قص التاريخ المعطى إلى بداية السنة .

8- new_time (دالة المنطقة الزمنية) :

وهي دالة المنطقة الزمنية تستدعي التاريخ ضمن منطقة زمنية معينة ، وفي المثال التالي (HST) هي المنطقة الزمنية للتاريخ ، و (EST) هي المنطقة الزمنية المقابلة لها ، والمطلوب إظهار التاريخ على أساسها .

```
SQL> select hiredate , new_time ( hiredate , 'EST','HST')
  2  from emp
  3  where ename = 'WARD' ;

HIREDATE  NEW_TIME(
--------- ---------
22-FEB-81 21-FEB-81
```

1- التحويل من بيانات التاريخ إلى بيانات حرفية والشكل العام هو :

to_char(date ,'fmt')

```
SQL>  select sysdate , to_char ( sysdate , 'DD/MM/YYY')
  2  from dual;

SYSDATE   TO_CHAR(S
--------- ---------
23-JUL-09 23/07/009
```

```
SQL> select sysdate , to_char (sysdate , 'DD "of" MM "of" YY')
  2  from dual;

SYSDATE   TO_CHAR(SYSDAT
--------- --------------
23-JUL-09 23 of 07 of 09
```

في المثاليين السابقين تم إظهار شكل التاريخ كبيانات حرفية بشكل مختلف عن صورة التاريخ الأساسية .

2- التحويل من بيانات رقمية إلى بيانات حرفية والشكل العام هو :

to_char(number ,'fmt')

```
SQL> select empno , to_char(sal,'$99,999') as salary
  2  from emp
  3  where deptno = 10 ;

     EMPNO SALARY
---------- --------
      7782   $2,450
      7839   $5,000
      7934   $1,300
```

في المثال السابق تم تحويل حقل الراتب إلى بيانات حرفية .

3- التحويل من بيانات حرفية إلى بيانات تاريخ والشكل العام هو :

to date (char ,'fmt')

```
SQL> select to_date ('NOVEMBER 18,1989','MONTH DD,YYYY')
  2  from dual;

TO_DATE('
---------
18-NOV-89
```

في المثال السابق تم تحويل البيانات الحرفية إلى بيانات تاريخ .

مثال أخر :

```
SQL> select to_date ('NOV 18,89','MON DD,YY')
  2  from dual;

TO_DATE('
---------
18-NOV-89
```

في المثال السابق تم أيضاً تحويل البيانات الحرفية إلى بيانات تاريخ ، ولكن بصيغة أخرى .

4- التحويل من بيانات حرفية إلى بيانات رقمية والشكل العام هو :

to_number (char ,'fmt')

تستخدم هذا الدالة لتحويل البيانات الحرفية التي تعبر عن رقم إلى بيانات رقمية ، كما في المثال التالي :

```
SQL> select to_number('1800',9999)
  2  from dual;

TO_NUMBER('1800',9999)
----------------------
                  1800
```

ملاحظة

- هناك تحويلات مختلفة تتم بين الأنواع ، ولكن لا يتم التحويل بين البيانات الرقمية ، وبيانات التاريخ والعكس .

توابع التحويل

يقوم بتغير الأغراض الخاصة به ومن هذه الدوال :

1- nvl (يحول القيمة الفارغة إلى قمة رقمية أو نصية أو تاريخ) :

```
SQL> select ename , sal - nvl (comm,0)
  2  from emp
  3  where deptno = 20 ;

ENAME      SAL-NVL(COMM,0)
---------- ---------------
SMITH                  800
JONES                 2975
SCOTT                 3000
ADAMS                 1100
FORD                  3000
```

في المثال السابق رغم أن بعض الموظفين حقل العمولة (**comm**) لديهم فارغ ، إلا أنه تم تحويل القيم الفارغة إلى قيم رقمية .

2- translate (ويعني الترجمة) :

ويستخدم لتبديل حروف عبارة حرفاً حرفاً حسب ما هو مطلوب ، والشكل العام هو :

translate(string , if , then)

```
SQL> select translate (3527644 , 234567890 , 'BCDEFGHIJ')
  2  from dual;

TRANSLA
-------
CEBGFDD
```

في المثال السابق تم البحث عن الرقم المراد ضمن (**if**) ، فإذا وجد الرقم يحدد مكان وروده ضمن العبارة ، ثم يبحث في نفس المكان من العبارة (**then**) ، ويبدل الحرف الذي وجده مع الرقم .

مثال أخر :

```
SQL> select translate ('LZGDQ','ABCDEFGHIJKLMNOPQRSTUVWXYZ','BCDEFGHIJKLMNOPQRSTUVWXYZA')
  2  from dual;

TRANS
-----
MAHER
```

في المثال السابق تم البحث عن الحروف المراده ضمن (**if**) ، فإذا وجد الحرف يحدد مكان وروده ضمن العبارة ، ثم يبحث في نفس المكان من العبارة (**then**) ، ويبدل الحرف الذي وجده مع الحرف الأخر .

3- decode

في **translate** كان التحويل حرف حرف ، أما في **decode** فإن التحويل يكون قيمة قيمة ، والشكل العام هو:

Decode (value , if 1 , then 1 , if 2 , then 2 , if 3 , then 3 , else ….)

```
SQL> select ename , deptno ,
  2  decode (deptno, 10 , 'ten ' , 20 , 'twenty' , 30 , 'thirty' , 'not found ' )
  3  from emp;

ENAME           DEPTNO DECODE(DEP
---------- ---------- ----------
SMITH               20 twenty
ALLEN               30 thirty
WARD                30 thirty
JONES               20 twenty
MARTIN              30 thirty
BLAKE               30 thirty
CLARK               10 ten
SCOTT               20 twenty
KING                10 ten
TURNER              30 thirty
ADAMS               20 twenty

ENAME           DEPTNO DECODE(DEP
---------- ---------- ----------
JAMES               30 thirty
FORD                20 twenty
MILLER              10 ten

14 rows selected.
```

في المثال السابق تم تحويل القيمة (10) إلى (ten) ، و (20) إلى (twenty) ، و (30) إلى (thirty) ، وفي حالة وجود قيم أخرى تحول إلى (not found) ، ولكن في المثال السابق لا يوجد قيم أخرى .

مثال أخر :

```
SQL> select deptno ,
  2  decode (deptno , 10 , 'ten' , 20 , 'twenty' , 30 , 'thirty' , 'not found')
  3  from dept;

    DEPTNO DECODE(DE
---------- ---------
        10 ten
        20 twenty
        30 thirty
        40 not found
```

في المثال السابق نلاحظ أنه تطرق إلى الخيار (else) (not found) ، وذلك لأن القيمة (40) لا يوجد لها تحويل .

4- ascii (يعطي قيمة الحرف أو الرمز في نظام الشفرة الأمريكية) :

```
SQL> select ascii('M'), ascii('R')
  2  from dual;

ASCII('M') ASCII('R')
---------- ----------
        77         82
```

في المثال السابق أعطي قيم الحروف (M,R) في نظام الشفرة الأمريكية .

مثال اخر :

```
SQL> select ascii('MAHER')
  2  from dual;

ASCII('MAHER')
--------------
            77
```

في المثال السابق أعطي قيمة أول حرف في السلسلة ، ولا يمكنه إعطاء أكثر من حرف في نفس الوقت .

5- chr (يعطي رمز الرقم في شفرة ascii) :

```
SQL> select chr(77),chr(82)
  2  from dual;

C C
- -
M R
```

في المثال السابق أعطى رمز الأرقام (77,82) في شفرة **ascii** .

2- الدوال التجميعية (دوال متعددة الصفوف) multiple row function : وتعطي قيمة واحدة من عدة صفوف ومنها :

دوال التجميع	
الوصف	الدالة
دالة التجميع (المجموع) .	**sum**
دوال أكبر وأقل قيمة وتستخدم للبيانات الرقمية والحرفية والتاريخ .	**max,min,greatest,least .**
دالة المتوسط (لحساب الوسط الحسابي) .	**avg**
دالة العد .	**count**
دالة الانحراف المعياري .	**stddevdevttion**
دالة التشتت .	**variance**

والدوال التجميعية تأتي مع حقول جملة الاستعلام **select** ومع جملة الشرط **where** .

أمثلة لبعض الدوال التجميعية :

1- sum (دالة المجموع) :

```
SQL> select sum(sal)
  2  from emp
  3  where deptno = 10 ;

  SUM(SAL)
----------
      8750
```

في المثال السابق تم عرض مجموع رواتب الموظفين في القسم رقم **10** .

2- max (دالة أكبر قيمة) :

```
SQL> select max(sal)
  2  from emp
  3  where deptno = 10 ;

  MAX(SAL)
----------
      5000
```

في المثال السابق تم عرض قيمة أكبر راتب في القسم رقم **10** .

3- min (دالة أقل قيمة) :

```
SQL> select min(sal)
  2  from emp
  3  where deptno = 10 ;

  MIN(SAL)
----------
      1300
```

في المثال السابق تم عرض قيمة أقل راتب في القسم رقم **10** .

4- greatest (دالة أكبر قيمة بين تاريخين أو نصيين معطيين) :

```
SQL> select greatest (to_date('09-may-1989'),to_date('18-nov-1989')) greatdate
  2  from dual;

GREATDATE
---------
18-NOV-89
```

في المثال السابق تم عرض التاريخ الأكبر بين التاريخين المعطيين .

مثال أخر :

```
SQL> select greatest ('MAHER','MOFEED')
  2  from dual;

GREATE
------
MOFEED
```

```
SQL> select greatest ('MOHAMMED','ABDULRHMAN')
  2  from dual;

GREATEST
--------
MOHAMMED
```

في المثاليين السابقين تم عرض النص الأكبر بحسب ترتيب الحروف ، وهنا (**z** أكبر من **a**) .

5- least (دالة أقل قيمة بين تاريخين أو نصيين معطيين) :

```
SQL> select least (to_date('09-may-1989'),to_date('18-nov-1989')) leastdate
  2  from dual;

LEASTDATE
---------
09-MAY-89
```

في المثال السابق تم عرض التاريخ الأقل بين التاريخيين المعطيين .

مثال أخر :

```
SQL> select least ('MAHER','MOFEED')
  2  from dual;

LEAST
-----
MAHER
```

```
SQL> select least ('MOHAMMED','ABDULRHMAN')
  2  from dual;

LEAST('MOH
----------
ABDULRHMAN
```

في المثاليين السابقين تم عرض النص الأقل بحسب ترتيب الحروف ، وهنا (a أقل من z) .

6- avg (دالة تعطي المتوسط) :

```
SQL> select avg(sal)
  2  from emp
  3  where deptno = 10 ;

  AVG(SAL)
----------
2916.66667
```

في المثال السابق تم إيجاد المتوسط الحسابي للرواتب في القسم رقم 10 .

7- count (دالة العد) :

```
SQL> select count(sal), count(comm), count(deptno)
  2  from emp;

COUNT(SAL) COUNT(COMM) COUNT(DEPTNO)
---------- ---------- -------------
        14          4            14
```

في المثال السابق تم عد الرواتب ، وعد العمولة ، وعد الأقسام ، ونلاحظ بأن (count) تقوم حتى بعد القيم المكررة في العمود (deptno) ، وفي العمود (sal) .

– ويمكن أيضاً استخدام **group by** مع الدوال التجميعية .

```
SQL> select deptno,sum(sal)
  2  from emp
  3  group by deptno
  4  order by sum(sal);

    DEPTNO   SUM(SAL)
---------- ----------
        10       8750
        30       9400
        20      10875
```

في المثال السابق تم عرض حقل القسم (**deptno**) ، ومجموع الرواتب ويكون التجميع على أساس حقل الأقسام مرتباً حسب حقل مجموع الرواتب .

الأخطاء:

```
SQL> select deptno,sum(sal)
  2  from emp
  3  where sum(sal)>1000
  4  group by deptno;
where sum(sal)>1000
      *
ERROR at line 3:
ORA-00934: group function is not allowed here
```

ظهور رسالة الخطأ في المثال السابق بسبب استخدام دالة (**sum**) في الجزء (**where**) ، والدوال التجميعية لا تأتي مع الجزء (**where**) .

و لتصحيح الخطأ يوجد جزء يسمى بـــ (**having**) ، وهذه الدالة خاصة بالدوال التجميعية ، وتأتي بعد (**group by**) .

وتعني (**having**) : أي أن ما يوجد عنده ينفذ .

```
SQL> select deptno , sum(sal)
  2  from emp
  3  group by deptno
  4  having sum(sal) > 8000 ;

    DEPTNO   SUM(SAL)
---------- ----------
        10       8750
        20      10875
        30       9400
```

مثال : (يستخدم جميع الأجزاء لجملة الاستعلام **select**)

```
SQL> select deptno , sum (sal) , avg (sal)
  2  from emp
  3  where deptno not like '10%'
  4  group by deptno
  5  having sum (sal) > 8500
  6  order by avg (sal);

    DEPTNO   SUM(SAL)   AVG(SAL)
---------- ---------- ----------
        30       9400 1566.66667
        20      10875       2175
```

في المثال السابق تم عرض الأقسام ، و مجموع الرواتب ، ومتوسط الرواتب ، بشرط أن القسم لا يكون **10** ، وكان التجميع على أساس رقم القسم الذي مجموع رواتبه اكبر من **8500** مرتباً بحسب متوسط الرواتب .

أي تم استخدام كل جميع أجزاء جملة الاستعلام **select** ، وهي أسماء الحقول ومصدر البيانات (**from**) والشرط الجزء (**where**) وجزء التجميع (**group by**) والجزء (**having**) والجزء (**order by**) .

8-2 عرض البيانات من أكثر من جدول

لعرض البيانات من أكثر من جدول لابد من عمل ربط بين الجداول المراد عرض البيانات منها ، ويكون ذلك الربط في جملة الاستعلام **select** .

– الربط : هو عبارة عن عمل علاقة بين جدولين أو أكثر في جملة الاستعلام (**select**) ، للحصول على بيانات من تلك الجداول .

– أنواع الربط :

1– الربط بالتساوي **equal join** .

2– الربط بعدم التساوي **non-equal join** .

3– الربط الخارجي **outer join** .

4– الربط الداخلي في نفس الجدول **self join** .

5– الربط بين أكثر من جدولين .

1- الربط بالتساوي equal join

يتم ربط جدولين أو أكثر عن طريق حقليين متساوين ، وعادة يتم بين الحقل الأول المفتاح الأساسي في الجدول الأول ، والمفتاح الأجنبي في الجدول الثاني .

```
SQL> select emp.deptno, emp.ename , emp.job
  2    , dept.deptno,dept.loc
  3  from emp ,dept
  4  where emp.deptno = dept.deptno ;

    DEPTNO ENAME      JOB           DEPTNO LOC
---------- ---------- --------- ---------- -------------
        20 SMITH      CLERK             20 DALLAS
        30 ALLEN      SALESMAN          30 CHICAGO
        30 WARD       SALESMAN          30 CHICAGO
        20 JONES      MANAGER           20 DALLAS
        30 MARTIN     SALESMAN          30 CHICAGO
        30 BLAKE      MANAGER           30 CHICAGO
        10 CLARK      MANAGER           10 NEW YORK
        20 SCOTT      ANALYST           20 DALLAS
        10 KING       PRESIDENT         10 NEW YORK
        30 TURNER     SALESMAN          30 CHICAGO
        20 ADAMS      CLERK             20 DALLAS

    DEPTNO ENAME      JOB           DEPTNO LOC
---------- ---------- --------- ---------- -------------
        30 JAMES      CLERK             30 CHICAGO
        20 FORD       ANALYST           20 DALLAS
        10 MILLER     CLERK             10 NEW YORK

14 rows selected.
```

في المثال السابق تم عرض بيانات من جدولين (**emp , dept**) ، ويتم ذلك بتحديد اسم الجدول أمام الحقل المراد استدعاه ، وفي جزء الشرط (**where**) تم الربط بين الجدولين على أساس الحقل (**deptno**) الموجود في الجدولين ، وهذا الحقل هو مفتاح أساسي في الجدول (**dept**) .

ويمكن أيضاً عرض بيانات من جدولين بأسماء مستعارة كما في المثال التالي :

```
SQL> select e.deptno , e.ename , e.job ,
  2  d.deptno , d.loc
  3  from emp e , dept d
  4  where e.deptno = d.deptno ;

    DEPTNO ENAME      JOB           DEPTNO LOC
---------- ---------- --------- ---------- -------------
        20 SMITH      CLERK             20 DALLAS
        30 ALLEN      SALESMAN          30 CHICAGO
        30 WARD       SALESMAN          30 CHICAGO
        20 JONES      MANAGER           20 DALLAS
        30 MARTIN     SALESMAN          30 CHICAGO
        30 BLAKE      MANAGER           30 CHICAGO
        10 CLARK      MANAGER           10 NEW YORK
        20 SCOTT      ANALYST           20 DALLAS
        10 KING       PRESIDENT         10 NEW YORK
        30 TURNER     SALESMAN          30 CHICAGO
        20 ADAMS      CLERK             20 DALLAS

    DEPTNO ENAME      JOB           DEPTNO LOC
---------- ---------- --------- ---------- -------------
        30 JAMES      CLERK             30 CHICAGO
        20 FORD       ANALYST           20 DALLAS
        10 MILLER     CLERK             10 NEW YORK

14 rows selected.
```

2- الربط بعدم التساوي non-equal join

يستخدم هذا الربط عندما لا توجد علاقة مباشرة بين الجدولين المراد ربطهما .

```
SQL> select e.empno , e.ename , e.sal , s.grade
  2  from emp e , salgrade s
  3  where e.sal between s.losal and s.hisal ;

     EMPNO ENAME             SAL      GRADE
---------- ---------- ---------- ----------
      7369 SMITH             800          1
      7876 ADAMS            1100          1
      7900 JAMES             950          1
      7521 WARD             1250          2
      7654 MARTIN           1250          2
      7934 MILLER           1300          2
      7499 ALLEN            1600          3
      7844 TURNER           1500          3
      7566 JONES            2975          4
      7698 BLAKE            2850          4
      7782 CLARK            2450          4

     EMPNO ENAME             SAL      GRADE
---------- ---------- ---------- ----------
      7788 SCOTT            3000          4
      7902 FORD             3000          4
      7839 KING             5000          5

14 rows selected.
```

في المثال السابق تم استدعاء أكثر من جدول بأسماء مستعارة ، ويشترط أن يكون الراتب بين أقل قيمة ، وأكبر قيمة في جدول (**salgrade**) .

3- الربط الخارجي outer join

يستخدم هذا الربط عندما توجد بيانات في أحد الجداول ، ولكنها لا تظهر في حالة الربط بالتساوي .

```
SQL> select e.empno , e.ename , d.deptno ,d.dname
  2  from emp e , dept d
  3  where e.deptno(+) = d.deptno ;

     EMPNO ENAME          DEPTNO DNAME
---------- ---------- ---------- --------------
      7782 CLARK              10 ACCOUNTING
      7839 KING               10 ACCOUNTING
      7934 MILLER             10 ACCOUNTING
      7369 SMITH              20 RESEARCH
      7876 ADAMS              20 RESEARCH
      7902 FORD               20 RESEARCH
      7788 SCOTT              20 RESEARCH
      7566 JONES              20 RESEARCH
      7499 ALLEN              30 SALES
      7698 BLAKE              30 SALES
      7654 MARTIN             30 SALES

     EMPNO ENAME          DEPTNO DNAME
---------- ---------- ---------- --------------
      7900 JAMES              30 SALES
      7844 TURNER             30 SALES
      7521 WARD               30 SALES
                              40 OPERATIONS

15 rows selected.
```

في المثال السابق تم عرض البيانات من أكثر من جدول ، ونلاحظ أن في الربط بالتساوي رقم القسم **40** الموجود في جدول (**dept**) لم يكن ظاهر ، ولكن بعد إضافة الجزء (+) بجانب الحقل (**deptno**) تم إظهار هذا القسم .

4- الربط الداخلي في نفس الجدول self join

هو تقسيم الجدول الواحد إلى جدولين أو أكثر .

```
SQL> select w.ename || ' works for ' || m.ename
  2  from emp w , emp m
  3  where w.mgr = m.empno ;

W.ENAME||'WORKSFOR'||M.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

W.ENAME||'WORKSFOR'||M.ENAME
------------------------------
FORD works for JONES
MILLER works for CLARK

13 rows selected.
```

في المثال السابق تم عرض بيانات من جدول واحد ، ولكن تم تغير اسمه إلى اسميين مختلفين ، وذلك لكي يتم إجراء مقار
في الاشتراط ، وكأنه يقارن بين جدولين مختلفين .

ونلاحظ في المثال انه لم يتم عرض بيانات الموظف (**KING**) ، وذلك لأن الموظف لا يملك مدير (لا يعمل لأحد)

5- الربط بين أكثر من جدولين

للربط بين أكثر من جدولين لابد أن تتوفر علاقة ما بينهم جميعاً ، علماً بأنه لابد أن تكون جمل الشرط المستخدمة في عملية الربط تساوي (عدد الجداول – **1**) .

فإذا كان لدينا جدولين فلابد أن يتوفر شرط واحد لربطهما على الأقل ، وإذا كان لدينا ثلاثة جداول فلابد أن يتوفر شرطين لربطهما على الأقل وهكذا .....

ولابد من وضع المعامل **and** أو **or** بين هذه الشروط .

```
SQL> select e.ename , e.sal , d.dname , s.grade
  2  from emp e , dept d , salgrade s
  3  where e.deptno = d.deptno and
  4  e.sal between s.losal and s.hisal ;

ENAME             SAL DNAME              GRADE
---------- ---------- -------------- ----------
SMITH              800 RESEARCH               1
ADAMS             1100 RESEARCH               1
JAMES              950 SALES                  1
WARD              1250 SALES                  2
MARTIN            1250 SALES                  2
MILLER            1300 ACCOUNTING             2
ALLEN             1600 SALES                  3
TURNER            1500 SALES                  3
JONES             2975 RESEARCH               4
BLAKE             2850 SALES                  4
CLARK             2450 ACCOUNTING             4

ENAME             SAL DNAME              GRADE
---------- ---------- -------------- ----------
SCOTT             3000 RESEARCH               4
FORD              3000 RESEARCH               4
KING              5000 ACCOUNTING             5

14 rows selected.
```

في المثال السابق تم استدعاء البيانات من ثلاثة جداول ، وتم الربط بينهما في جزء الشرط (**where**) .

9-2 الاستعلامات الفرعية (subquery)

عندما نريد الاستعلام عن موظفين بدلالة موظف آخر كما في المثال التالي :

```
SQL> select sal from emp
  2  where ename = 'WARD';

       SAL
----------
      1250
```

وبعد ذلك الاستعلام عن الموظفين الذين رواتبهم أعلى من الموظف (**WARD**) :

```
SQL> select ename from emp
  2  where sal > 1250 ;

ENAME
----------
ALLEN
JONES
BLAKE
CLARK
SCOTT
KING
TURNER
FORD
MILLER

9 rows selected.
```

نلاحظ في المثال السابق أننا احتجنا استعلامين للحصول على النتيجة التي نريدها ، فكيف نستطيع أن نحصل على النتيجة باستعلام واحد .

سوف نعيد كتابة الاستعلامين السابقين في استعلام واحد كما في المثال التالي :

```
SQL> select ename , sal from emp
  2  where sal > ( select sal from emp
  3  where ename = 'WARD');

ENAME             SAL
---------- ----------
ALLEN            1600
JONES            2975
BLAKE            2850
CLARK            2450
SCOTT            3000
KING             5000
TURNER           1500
FORD             3000
MILLER           1300

9 rows selected.
```

مثال آخر :

```
SQL> select ename , deptno , mgr from emp
  2  where deptno = ( select deptno from emp
  3  where ename ='ALLEN');

ENAME          DEPTNO        MGR
---------- ---------- ----------
ALLEN              30       7698
WARD               30       7698
MARTIN             30       7698
BLAKE              30       7839
TURNER             30       7698
JAMES              30       7698

6 rows selected.
```

في المثال السابق نتيجة الاستعلام الفرعي هو القسم رقم **30** ، وهو القسم الذي يعمل به الموظف (**ALLEN**) .

ونلاحظ أن الاستعلام الرئيسي تم تنفيذه بعد الاستعلام الفرعي بشكل مستقل عن الاستعلام الفرعي ، والعلاقة فقط بنتيجة الاستعلام الفرعي .

ملاحظة

يتم تنفيذ الاستعلام الفرعي أولاً ، ثم ينفذ الاستعلام الرئيسي .

– أنواع الاستعلامات الفرعية :

1– استعلام فرعي أحادي الصف (**single – row subquery**) .

2– استعلام فرعي متعدد الصفوف (**multiple – row subquery**) .

3– استعلام فرعي متعدد الأعمدة (**multiple – column subquery**) .

ملاحظات

– يوضع الاستعلام الفرعي بين قوسين .

– يوضع الاستعلام الفرعي يمين معامل المقارنة .

وتكتب الاستعلامات الفرعية داخل جملة الاستعلام **select** :

بعد الجزء **where** ، وبعد الجزء **having** ، وبعد الجزء **from** .

1– الاستعلامات الفرعية الأحادية الصف (single – row subquery) :

وهي استعلامات دائماً تكون نتيجتها صف واحد فقط ، وتستخدم معها معاملات المقارنة (<,>,<=,>=,<>) .

```
SQL> select ename , sal , deptno
  2  from emp
  3  where deptno = ( select deptno from emp
  4  where ename = 'WARD');

ENAME              SAL     DEPTNO
---------- ---------- ----------
ALLEN            1600         30
WARD             1250         30
MARTIN           1250         30
BLAKE            2850         30
TURNER           1500         30
JAMES             950         30

6 rows selected.
```

في المثال السابق تم عرض أسماء ، ورواتب ، وأقسام الموظفين الذين يعملون في نفس قسم الموظف (**WARD**) .

مثال أخر :

```
SQL> select ename , sal from emp
  2  where sal >(select sal
  3  from emp where empno=7934);

ENAME              SAL
---------- ----------
ALLEN            1600
JONES            2975
BLAKE            2850
CLARK            2450
SCOTT            3000
KING             5000
TURNER           1500
FORD             3000

8 rows selected.
```

في المثال السابق تم عرض أسماء ، ورواتب الموظفين الذين مرتباتهم أعلى من مرتب الموظف الذي رقمه (**7934**) .

2- الاستعلامات الفرعية متعددة الصفوف (multiple-row subquery) :

وهي استعلامات ترجع دائماً بأكثر من صف ، وتستخدم معها معاملات المقارنة (in , =any , <any , exist , not exist, >any , =all , <all , >all) .

```
SQL> select ename , sal , deptno from emp
  2  where sal in (select max(sal)
  3  from emp group by deptno);

ENAME             SAL     DEPTNO
---------- ---------- ----------
BLAKE            2850         30
SCOTT            3000         20
FORD             3000         20
KING             5000         10
```

في المثال السابق تم عرض أسماء ، وأرقام ، ورواتب الموظفين الذين يأخذون رواتب مساوية لأعلى راتب في كل قسم .

مثال أخر :

```
SQL> select empno , ename , sal , job from emp
  2  where sal <all (select avg(sal)
  3  from emp group by deptno);

     EMPNO ENAME             SAL JOB
---------- ---------- ---------- ---------
      7369 SMITH             800 CLERK
      7521 WARD             1250 SALESMAN
      7654 MARTIN           1250 SALESMAN
      7844 TURNER           1500 SALESMAN
      7876 ADAMS            1100 CLERK
      7900 JAMES             950 CLERK
      7934 MILLER           1300 CLERK

7 rows selected.
```

في المثال السابق تم عرض أرقام ، و أسماء ، ورواتب ، ووظائف الموظفين الذين رواتبهم أقل من كل المتوسطات الحسابية للمرتبات في كل قسم .

3- الاستعلامات الفرعية متعددة الأعمدة (multiple-column subquery) :

وهي استعلامات ترجع بأكثر من عمود ، و أكثر من صف ، ويستخدم معها المعاملات متعددة الصفوف .

```
SQL> select e.empno , e.ename , e.sal , e.deptno , m.salavg
  2  from emp e , (select deptno , avg(sal) salavg
  3  from emp group by deptno ) m
  4  where e.deptno = m.deptno
  5  and e.sal >= m.salavg ;

     EMPNO ENAME             SAL     DEPTNO     SALAVG
---------- ---------- ---------- ---------- ----------
      7839 KING             5000         10 2916.66667
      7902 FORD             3000         20       2175
      7788 SCOTT            3000         20       2175
      7566 JONES            2975         20       2175
      7499 ALLEN            1600         30 1566.66667
      7698 BLAKE            2850         30 1566.66667

6 rows selected.
```

في المثال السابق تم استدعاء البيانات من الجدول (**emp**) ، وضع له اسم مستعار **e** ، ومن الاستعلام الفرعي الذي وضع له اسم مستعار **m** .

– استخدام المعاملات (union , intersect , minus) :

المعامل	الوصف
union	الاتحاد : ويعني البيانات الموجودة في الجدولين .
intersect	التقاطع : ويعني البيانات المشتركة في الجدولين .
minus	الفرق : ويعني البيانات الموجودة في الجدول الأول وغير موجودة في الجدول الثاني.

أمثلة :

```
SQL> select deptno from emp
  2  union
  3  select deptno from dept;

    DEPTNO
----------
        10
        20
        30
        40
```

في المثال السابق تم عرض رقم القسم من الجدولين باستخدام الاتحاد (**union**) .

```
SQL> select deptno from emp
  2  intersect
  3  select deptno from dept;

    DEPTNO
----------
        10
        20
        30
```

في المثال السابق تم عرض رقم القسم من الجدولين باستخدام التقاطع (**intersect**) .

```
SQL> select deptno from emp
  2  minus
  3  select deptno from dept;

no rows selected
```

في المثال السابق لم تعرض أي نتائج باستخدام (**minus**) ، والسبب انه كل أرقام الأقسام الموجودة في الجدول (**emp**) موجود في الجدول (**deptno**) .

```
SQL> select deptno from dept
  2  minus
  3  select deptno from emp;

    DEPTNO
----------
        40
```

في المثال السابق نلاحظ وجود نتائج والسبب أن رقم القسم **40** موجود في الجدول (**dept**) ، وغير موجود في الجدول (**emp**) .

```
SQL> select empno , sal from emp
  2  where sal in
  3  (select sal from emp where ename ='WARD'
  4  union
  5  select sal from emp where ename ='JAMES');

     EMPNO        SAL
---------- ----------
      7900        950
      7521       1250
      7654       1250
```

في المثال السابق تم استخدام المعامل (**union**) بين استعلامين فرعين .

2- 10 تمارين الفصل الثاني

1- قم بعرض أسماء ، ووظائف ، وأقسام الموظفين الذين وظائفهم مساوية لوظيفة صاحب الرقم **7566** ، ورواتبهم اكبر من رواتب الموظف صاحب الرقم **7369**.

2- قم بعرض أسماء و أرقام الأقسام ، واقل راتب يأخذه موظف فيها بحيث يكون اقل راتب فيها اكبر من اقل راتب في القسم رقم **30**.

3- قم بعرض اسم ، ووظيفة الموظف الذي وظيفته تساوي أحد هذه الوظائف (**SALESMAN , CLERK**) .

(sql & sql * plus)

الفصل الثالث :

لغة التعامل مع البيانات

Data Manipulating Language
(DML)

لغة التعامل مع البيانات (Data Manipulating language) :

هي لغة تتعامل مع البيانات داخل الجداول من حيث :

1- إضافة البيانات إلى الجداول ، ويستخدم لذلك الأمر (**insert**) .

2- تعديل بيانات الحقول في الجداول ، ويستخدم لذلك الأمر (**update**) .

3- إلغاء البيانات من الجداول ، ويستخدم لذلك الأمر (**delete**) .

1-3 إضافة البيانات إلى الجداول

لإضافة بيانات إلى الجداول يستخدم الأمر (**insert**) ، وتأخذ عدة صور والشكل العام هو :

```
insert into table name ( column1 , column2 , ……..)
values ( value1 , value2 , ……..) ;
```

وعند إضافة البيانات إلى الجداول :

- يجب أن تكون عدد القيم التي سيتم إضافتها مساوية لعدد الأعمدة في جملة **insert** .
- عند إدخال قيمة نصية أو تاريخ يجب وضعها بين علامتي تنصيص مفردة .
- يجب إدخال قيماً للأعمدة التي لا تقبل قيماً فارغة أو ستكون مفتاح أساسي .
- يجب عند الإدخال مراعاة ترتيب الحقول بنفس الترتيب الموجود في الجدول .

ملاحظة

- سوف أقوم بإنشاء جداول مشابهة لجداول المستخدم **scott** ، ونتعامل مع النسخ بدل من التعامل مع الجداول الأساسية حتى لا تتأثر هذه الجداول بالتغيرات التي ستطرأ عليها .

```
SQL> insert into emp2 (empno,ename,job,mgr,hiredate,sal,comm,deptno)
  2  values(8000,'MAHER','MANAGER',7839,'18-NOV-89',4500,NULL,20);

1 row created.
```

في المثال السابق تم إضافة بيانات موظف للجدول (**emp2**) ، ونلاحظ أنه يجب علينا مراعاة الترتيب الموجود في الجدول عند الإدخال ، وللتأكد من أنه تم إضافة موظف نقوم بعرض بيانات الجدول (**emp2**) :

```
SQL> select * from emp2;

     EMPNO ENAME      JOB              MGR HIREDATE         SAL       COMM     DEPTNO
---------- ---------- --------- ---------- --------- ---------- ---------- ----------
      7369 SMITH      CLERK           7902 17-DEC-80        800                    20
      7499 ALLEN      SALESMAN        7698 20-FEB-81       1600        300         30
      7521 WARD       SALESMAN        7698 22-FEB-81       1250        500         30
      7566 JONES      MANAGER         7839 02-APR-81       2975                    20
      7654 MARTIN     SALESMAN        7698 28-SEP-81       1250       1400         30
      7698 BLAKE      MANAGER         7839 01-MAY-81       2850                    30
      7782 CLARK      MANAGER         7839 09-JUN-81       2450                    10
      7788 SCOTT      ANALYST         7566 19-APR-87       3000                    20
      7839 KING       PRESIDENT            17-NOV-81       5000                    10
      7844 TURNER     SALESMAN        7698 08-SEP-81       1500          0         30
      7876 ADAMS      CLERK           7788 23-MAY-87       1100                    20

     EMPNO ENAME      JOB              MGR HIREDATE         SAL       COMM     DEPTNO
---------- ---------- --------- ---------- --------- ---------- ---------- ----------
      7900 JAMES      CLERK           7698 03-DEC-81        950                    30
      7902 FORD       ANALYST         7566 03-DEC-81       3000                    20
      7934 MILLER     CLERK           7782 23-JAN-82       1300                    10
      8000 MAHER      MANAGER         7839 18-NOV-89       4500                    20

15 rows selected.
```

نلاحظ أنه قد تم إضافة بيانات الموظف في نهاية الجدول .

مثال آخر :

```
SQL> insert into dept2 ( deptno , dname , loc )
  2  values (50,'ENGINEER','SANA`A');

1 row created.
```

في المثال السابق تم إضافة قسم جديد للجدول (**dept2**) .

```
SQL> select * from dept2;

    DEPTNO DNAME          LOC
---------- -------------- -------------
        10 ACCOUNTING     NEW YORK
        20 RESEARCH       DALLAS
        30 SALES          CHICAGO
        40 OPERATIONS     BOSTON
        50 ENGINEER       SANA`A
```

وإذا أردنا في المثال السابق عدم كتابة بيانات في حقل الموقع نعيد صياغة المثال كالتالي :

```
SQL> insert into dept2 (deptno , dname , loc )
  2  values (50,'ENGINEER','');

1 row created.
SQL> select * from dept2;

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

صورة أخرى للإضافة :

```
SQL> insert into dept2
  2  values (60,'DOCTOR','ADEN');

1 row created.

SQL> select * from dept2;

    DEPTNO DNAME          LOC
---------- -------------- -------------
        10 ACCOUNTING     NEW YORK
        20 RESEARCH       DALLAS
        30 SALES          CHICAGO
        40 OPERATIONS     BOSTON
        50 ENGINEER       SANA`A
        60 DOCTOR         ADEN

6 rows selected.
```

في المثال السابق تم إضافة بيانات لحقل القسم ، واسم القسم ، والموقع ، بدون تحديد أسماء الحقول المراد الإضافة إليها ، ولكن يشترط هنا أن تحافظ على الترتيب الموجود في الجدول عند الإدخال .

صورة أخرى للإضافة :

```
SQL> insert into emp2
  2  values (&empno,'&ename','&job',&mgr,'&hiredate',&sal,&comm,&deptno);
Enter value for empno: 8001
Enter value for ename: MOFEED
Enter value for job: ENGINEER
Enter value for mgr: 8000
Enter value for hiredate: 5-DEC-90
Enter value for sal: 3500
Enter value for comm: NULL
Enter value for deptno: 20
old   2: values (&empno,'&ename','&job',&mgr,'&hiredate',&sal,&comm,&deptno)
new   2: values (8001,'MOFEED','ENGINEER',8000,'5-DEC-90',3500,NULL,20)

1 row created.
```

في المثال السابق تم إضافة بيانات لموظف واحد ، وإذا أردنا إضافة بيانات أخرى نستخدم الأمر (/) أو الأمر (**R**) لكي يقوم بإعادة كتابة دالة الإدخال كما في الأمثلة التالية :

```
SQL> /
Enter value for empno: 8002
Enter value for ename: MOHAMMED
Enter value for job: ENGINEER
Enter value for mgr: 8000
Enter value for hiredate: 9-MAY-89
Enter value for sal: 3500
Enter value for comm: NULL
Enter value for deptno: 20
old   2: values (&empno,'&ename','&job',&mgr,'&hiredate',&sal,&comm,&deptno)
new   2: values (8002,'MOHAMMED','ENGINEER',8000,'9-MAY-89',3500,NULL,20)

1 row created.

SQL> R
  1  insert into emp2
  2* values (&empno,'&ename','&job',&mgr,'&hiredate',&sal,&comm,&deptno)
Enter value for empno: 8003
Enter value for ename: ABDULRHMAN
Enter value for job: ENGINEER
Enter value for mgr: 8000
Enter value for hiredate: 12-SEP-88
Enter value for sal: 3500
Enter value for comm: NULL
Enter value for deptno: 20
old   2: values (&empno,'&ename','&job',&mgr,'&hiredate',&sal,&comm,&deptno)
new   2: values (8003,'ABDULRHMAN','ENGINEER',8000,'12-SEP-88',3500,NULL,20)

1 row created.
```

وللتأكد من نجاح إضافة البيانات نقوم بعرض بيانات جدول (**emp2**) .

```
SQL> select * from emp2;

     EMPNO ENAME      JOB              MGR HIREDATE         SAL       COMM     DEPTNO
---------- ---------- --------- ---------- --------- ---------- ---------- ----------
      7369 SMITH      CLERK           7902 17-DEC-80        800                    20
      7499 ALLEN      SALESMAN        7698 20-FEB-81       1600        300         30
      7521 WARD       SALESMAN        7698 22-FEB-81       1250        500         30
      7566 JONES      MANAGER         7839 02-APR-81       2975                    20
      7654 MARTIN     SALESMAN        7698 28-SEP-81       1250       1400         30
      7698 BLAKE      MANAGER         7839 01-MAY-81       2850                    30
      7782 CLARK      MANAGER         7839 09-JUN-81       2450                    10
      7788 SCOTT      ANALYST         7566 19-APR-87       3000                    20
      7839 KING       PRESIDENT            17-NOV-81       5000                    10
      7844 TURNER     SALESMAN        7698 08-SEP-81       1500          0         30
      7876 ADAMS      CLERK           7788 23-MAY-87       1100                    20

     EMPNO ENAME      JOB              MGR HIREDATE         SAL       COMM     DEPTNO
---------- ---------- --------- ---------- --------- ---------- ---------- ----------
      7900 JAMES      CLERK           7698 03-DEC-81        950                    30
      7902 FORD       ANALYST         7566 03-DEC-81       3000                    20
      7934 MILLER     CLERK           7782 23-JAN-82       1300                    10
      8001 MOFEED     ENGINEER        8000 05-DEC-90       3500                    20
      8002 MOHAMMED   ENGINEER        8000 09-MAY-89       3500                    20
      8003 ABDULRHMAN ENGINEER        8000 12-SEP-88       3500                    20
      8000 MAHER      MANAGER         7839 18-NOV-89       4500                    20

18 rows selected.
```

وتسمى الطريقة السابقة بطريقة (المتغيرات البديلة) substitution variables :

وهي عبارة عن مخزن مؤقت للبيانات ، ومن خلالها يتم تخزين قيم معينة داخل هذه المتغيرات ، وأثناء تنفيذ جملة (sql) يتم استبدال هذه المتغيرات بقيمتها ، ويتم تعريفها أثناء كتابة جملة (sql) ، وذلك بوضع العلامة (&) قبل اسم المتغير .

2-3 تعديل بيانات في جداول

لتعديل بيانات في جدول نستخدم الأمر (update) وتأخذ عدة صور والشكل العام هو :

```
update table name set column1= value , column2= value
where condition ;
```

```
SQL> update emp2 set sal = 4900
  2  where ename = 'MAHER';

1 row updated.
```

في المثال السابق تم تعديل حقل الراتب للموظف (MAHER) .

```
SQL> update dept2 set loc='TAIZ'
  2  where deptno = 50 ;

1 row updated.

SQL> select * from dept2;

    DEPTNO DNAME          LOC
---------- -------------- -------------
        10 ACCOUNTING     NEW YORK
        20 RESEARCH       DALLAS
        30 SALES          CHICAGO
        40 OPERATIONS     BOSTON
        50 ENGINEER       TAIZ
        60 DOCTOR         ADEN

6 rows selected.
```

في المثال السابق تم تعديل حقل الموقع (loc) للقسم رقم 50 .

صورة أخرى للتعديل :

```
SQL> update emp2 set
  2  job = (select job from emp2 where empno = 8000 )
  3  where empno = 8001;

1 row updated.
```

في المثال السابق تم استخدام الاستعلامات الفرعية في التعديل ، حيث تم تعديل حقل الوظيفة بحيث تساوي القيم الناتجة من الاستعلام الفرعي ، وهي وظيفة الموظف رقم **8000** ، و التعديل يتم للموظف رقم **8001** فيصبح وظيفة الموظف رقم **8001** كالتالي :

```
8001 MOFEED     MANAGER         8000 05-DEC-90       3500                    20
```

3-3 حذف بيانات من جداول

لحذف بيانات من جدول يستخدم الأمر (**delete**) وتأخذ عدة صور والشكل العام هو :

```
delete from table_name
where condition ;
```

ملاحظات

- يمكن التراجع عن الحذف باستخدام الأمر (**rollback**) .
- الحذف في الجدول للبيانات يتم على مستوى السجل كامل .
- إذا لم يحتوي أمر الحذف على جملة شرط فسوف يتم حذف بيانات جميع السجلات .

```
SQL> delete from dept2
  2  where deptno = 50 ;

1 row deleted.
```

وللتراجع عن الحذف :

```
SQL> rollback;

Rollback complete.
```

صورة أخرى للحذف :

```
SQL> delete from emp2
  2  where deptno = (select deptno from emp2
  3  where ename = 'KING');

3 rows deleted.
```

في المثال السابق تم حذف القسم الذي يعمل به الموظف (**KING**) .

الأمر truncate

يقوم هذا الأمر بنفس عمل الدالة **delete** ، إلا أنها تختلف في أنه لا يمكن التراجع عن حذف البيانات باستخدام الأمر **rollback** ، وكذلك لأنه أسرع من الأمر **delete** .

3- 4 تمارين الفصل الثالث

1- قم بإضافة البيانات التالية لجدول emp :

empno = 7777 , ename = أسمك , job = MANAGER , mgr = 7839,
hiredate = 15-SEP-09 , sal = 5000 , comm=NULL, deptno = 30 .

2- قم بتعديل الحقل (deptno) في المثال السابق إلى 20 .

3- قم بحذف البيانات السابقة .

أعداد / ماهر محمد أحمد الرياشي

(sql & sql * plus)

الفصل الرابع :

أوامر النقل

Transection command

1-4 أوامر النقل (transaction command)

وهو القسم الذي يختص بعملية نقل البيانات من منطقة العمل التمهيدية (الاحتياطية) ، وقاعدة البيانات ،

ومن هذه الأوامر :

الأمر	الوصف
commit	يستخدم هذا الأمر لعملية التخزين اليدوي .
Auto commit	يستخدم هذا الأمر لعملية التخزين الآلي .
rollback	يستخدم هذا الأمر للتراجع عن التغييرات التي طرأت على البيانات ، ولكن قبل استخدام أمر الحفظ commit .
savepoint	نقطة في البرنامج ويمكن التراجع عنها باستخدام الأمر rollback .

– الأمر commit :

يستخدم لعملية حفظ الأوامر بعد كتابتها .

```
SQL> insert into dept2 ( deptno , dname , loc )
  2  values (50,'ENGINEER','SANA'A');

1 row created.
```

```
SQL> commit;

Commit complete.
```

– الأمر auto commit :

يستخدم كذلك لعملية الخزن ، ولكن بطريقة آلية .

– الأمر rollback :

يستخدم هذا الأمر للتراجع عن التغييرات التي طرأت على البيانات ، ولكن قبل استخدام أمر الحفظ commit

.

```
SQL> rollback;

Rollback complete.
```

– الأمر savepoint :

نقطة في البرنامج ، ويمكن التراجع عنها باستخدام الأمر rollback .

4- 2 تمارين الفصل الرابع

في هذا الفصل لا توجد تمارين .

(sql & sql * plus)

الفصل الخامس :

لغة توصيف البيانات

Data Definition Language

لغة توصيف البيانات (DDL) Data Definition Language :

وهو القسم الذي يختص بتوصيف البيانات داخل قاعدة البيانات ، ويمكننا من إجراء عمليات الإنشاء (**create**) ، والتعديل (**alter**)، والإلغاء (**drop**) علي أي كائن داخل قاعدة البيانات .

1-5 العمليات على الجداول (tables)

– إنشاء الجداول (creating tables) :

وهي عملية إنشاء جداول داخل قاعدة البيانات ، ويتم ذلك من خلال الأمر (**create**) .

نوع البيانات	الوصف
number	تستخدم مع البيانات الرقمية .
date	تستخدم مع بيانات الوقت والتاريخ .
char	تستخدم مع البيانات الحرفية الثابتة الطول ، ويجب تحديد طولها وإلا سيحددها النظام بواحد .
varchar2	تستخدم مع البيانات الحرفية المتغيرة الطول .
long	تستخدم لتمثيل البيانات كبيرة الحجم التي تصل طولها إلى 2 جيجا بايت .
long row	تستخدم لتمثيل البيانات الكبيرة الحجم مثل الصور ، والتي تصل طولها إلى 4 جيجا بايت .
clob-blob	تستخدم لتمثيل البيانات الكبيرة الحجم مثل الصور ،والتي تصل طولها إلى أكثر من 4 جيجا بايت.
dfile	تستخدم لتخزين الملفات الكبيرة ، والتي يصل حجمها إلى أكثر من 4 جيجا بايت .

ملاحظات

– اسم الجدول لا يتجاوز **30** حرفاً ، ويمكن أن يكون حروف كبيرة أو صغيرة .

– يجب أن لا يكون اسم الجدول من الكلمات المحجوزة في أوراكل مثل (**create**) .

– أن لا يحمل رموز ، ولا فراغات ماعدا الرموز التالية (, , _ , $, #)

– اسم الجدول يبدأ بمتغير حرفي ، وليس رقمي .

– يجب أن لا يتكرر اسم الجدول أكثر من مره داخل قاعدة البيانات .

– المتغير الرقمي في الجدول يصل إلى **38** خانة ، وتحديد الطول ليس إجبارياً .

– المتغير الحرفي في الجدول يصل إلى **240** خانة ، وتحديد الطول إجبارياً .

– التاريخ يكون **9** خانات ، ولا يتم تحديد طوله عند الإنشاء .

– يجب أن لا يتكرر اسم الحقل أكثر من مره داخل الجدول .

الأمر create

يستخدم لإنشاء الجداول ، والشكل العام هو :

```
create table table name (
column1      data type
column2      data type
.............  .............
column(n)    data type );
```

فإذا أردنا إنشاء جدول مشابه تماماً لجدول الموظفين (**emp**) سيكون الإنشاء كالتالي :

```
SQL> create table employee(
  2  empno     number(4) NOT NULL,
  3  ename     varchar2(15),
  4  job       varchar2(15),
  5  mgr       number(4),
  6  hiredate  date,
  7  sal       number(5,2),
  8  comm      number(5,2),
  9  deptno    number(2)
 10  );

Table created.

SQL> desc employee;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 EMPNO                                     NOT NULL NUMBER(4)
 ENAME                                              VARCHAR2(15)
 JOB                                                VARCHAR2(15)
 MGR                                                NUMBER(4)
 HIREDATE                                           DATE
 SAL                                                NUMBER(5,2)
 COMM                                               NUMBER(5,2)
 DEPTNO                                             NUMBER(2)
```

في المثال السابق تم إنشاء الحقل (**empno**) من نوع البيانات الرقمية ، طول الحقل **4** ونلاحظ كتبنا أمامه (**NOT NULL**) أي هذا الحقل يجب عند إدخال البيانات أن لا يترك فارغاً ، الحقل الثاني (**ename**) من نوع البيانات النصية طول الحقل **15** ، الحقل الثالث (**job**) من نوع البيانات النصية كذلك طول الحقل **15** ، الحقل الرابع (**mgr**) من نوع البيانات الرقمية طول الحقل **4** ، الحقل الخامس (**hiredate**) من نوع بيانات التاريخ ونلاحظ عدم كتابة حجم لهذا الحقل ، الحقل السادس (**sal**) من نوع البيانات الرقمية وطول الحقل (**5,2**) أي خمس خانات صحيحة وخانتان عشريتان ، الحقل السابع (**comm**) من نوع البيانات الرقمية ، وطول الحقل (**5,2**) كذلك خمس خانات صحيحة وخانتان عشريتان ، الحقل الثامن والأخير (**deptno**) من نوع البيانات الرقمية وطول الحقل **2** .

ويمكن أيضاً إنشاء جدول مشابه لجدول (**emp**) بالطريقة التالية :

```
SQL> create table employee1 as select * from emp;

Table created.
```

ويمكن أن ننشى جدول مشابه لجدول (**emp**) ، ولكن بجزء من البيانات كما في المثال التالي :

```
SQL> create table employee2 as select * from emp
  2  where deptno = 10 and deptno = 20 ;

Table created.
```

في المثال السابق تم إنشاء جدول مشابه لجدول (**emp**) للأقسام **10** و **20** فقط .

ويمكن أيضاً إنشاء جدول بواسطة الاستعلام الفرعي ببعض الحقول كما في المثال التالي :

```
SQL> create table emp3 as
  2  select empno , ename , sal * 12 annual_salary , comm
  3  from emp;

Table created.
```

في المثال السابق تم إنشاء جدول مشابه لجدول (**emp**) ، ولكن ببعض الحقول ونلاحظ إنشاء حقل الراتب مضروباً في **12** تحت مسمى (**annual_salary**) .

```
SQL> select * from emp3 ;

     EMPNO ENAME      ANNUAL_SALARY       COMM
---------- ---------- ------------- ----------
      7369 SMITH              9600
      7499 ALLEN             19200        300
      7521 WARD              15000        500
      7566 JONES             35700
      7654 MARTIN            15000       1400
      7698 BLAKE             34200
      7782 CLARK             29400
      7788 SCOTT             36000
      7839 KING              60000
      7844 TURNER            18000          0
      7876 ADAMS             13200

     EMPNO ENAME      ANNUAL_SALARY       COMM
---------- ---------- ------------- ----------
      7900 JAMES             11400
      7902 FORD              36000
      7934 MILLER            15600

14 rows selected.
```

– التعديل في مواصفات الجداول (altering table) :

الأمر alter

وذلك باستخدام الأمر (**alter**) ، ويكون التعديل من حيث :

1– إضافة حقول للجدول.

والشكل العام هو :

1- alter table_name add (column_name type(size));

```
SQL> alter table dept2 add ( NOTES varchar2(50) ) ;

Table altered.
```

تم في المثال السابق إضافة حقل للجدول (**dept2**) .

```
SQL> select * from dept2;

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

2– تعديل حجم الحقل ونوع بياناته .

والشكل العام هو :

2- alter table_name modify (column_name type(size));

```
SQL> alter table emp2 modify (deptno number(3) );

Table altered.
```

في المثال السابق تم تعديل حجم الحقل (**deptno**) من **2** إلى **3** .

```
SQL> desc emp2;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 EMPNO                                              NUMBER(4)
 ENAME                                              VARCHAR2(10)
 JOB                                                VARCHAR2(9)
 MGR                                                NUMBER(4)
 HIREDATE                                           DATE
 SAL                                                NUMBER(7,2)
 COMM                                               NUMBER(7,2)
 DEPTNO                                             NUMBER(3)
```

الأخطاء:

```
SQL> alter table emp2 modify (deptno number(1) );
alter table emp2 modify (deptno number(1) )
                         *
ERROR at line 1:
ORA-01440: column to be modified must be empty to decrease precision or scale
```

في المثال السابق ظهرت رسالة خطأ ، وذلك بسبب تعديل الحقل إلى رقم أقل من الرقم الأول ، ولا يمكننا فعل ذلك إلى إذا كان الحقل فارغاً (لا يوجد به بيانات) ، أما إذا كان به بيانات فلا نستطيع تعديل الحقل إلى رقم أقل .

3- إلغاء الحقول.

والشكل العام هو :

3- alter table_name drop column column_name ;

```
SQL> alter table emp2  drop column comm ;

Table altered.
```

في المثال السابق تم حذف (إلغاء) حقل العمولة (**comm**) .

```
SQL> desc emp2;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 EMPNO                                              NUMBER(4)
 ENAME                                              VARCHAR2(10)
 JOB                                                VARCHAR2(9)
 MGR                                                NUMBER(4)
 HIREDATE                                           DATE
 SAL                                                NUMBER(7,2)
 DEPTNO                                             NUMBER(3)
```

4- تغير اسم الجدول .

والشكل العام هو :

4- rename old_name to new_name ;

```
SQL> rename emp2 to emp22;

Table renamed.
```

في المثال السابق تم تغير أسم الجدول (**emp2**) إلى الاسم (**emp22**) .

5- إلغاء الجداول.

والشكل العام هو :

5- drop table table_name ;

```
SQL> drop table emp22;

Table dropped.
```

في المثال السابق تم حذف (إلغاء) الجدول (**emp22**) .

6- إضافة تعليق للجدول.

والشكل العام هو :

6- comment on table table_name is ' statement ' ;

```
SQL> comment on table dept2
  2  is 'Department Information ';

Comment created.
```

في المثال السابق تم إضافة تعليق يوضح ما يحتويه جدول (**dept2**) .

ملاحظات

- يمكن إضافة عمود جديد في أي لحظة بشرط أن لا تعطى الصفة (**not null**) .
- في حالة تعديل طول الحقل إلى الأقل يجب أن يكون الحقل فارغ .
- لا يمكن إلغاء أكثر من حقل (عمود) في الأمر الواحد .
- يجب أن يبقى حقل (عمود) واحد على الأقل بعد عملية الإلغاء .
- عند إدراج تعليق يمكن حذفه ، وذلك من قاعدة البيانات بجعل قيمته خالية فقط .

أنواع الجداول في بيئة قواعد البيانات أوراكل :

1- جداول يتم إنشاؤها من قبل المستخدمين .

2- جداول يتم إنشاؤها من قبل (**oracle server**) ، وتنقسم إلى عدة فئات :

- الفئة التي تبدأ (**user_**) ، وتحتوي على معلومات حول الكائنات الخاصة بالمستخدمين .
- الفئة التي تبدأ (**all_**) ، وتحتوي على معلومات عن كل الجداول والعلاقات التي يمكن للمستخدم الدخول عليها .
- الفئة التي تبدأ (**dba_**) ، وتحتوي على معلومات خاصة بمديري قواعد البيانات ، ولا يمكن لأحد الدخول إليها أو استخدامها .

– القيود على الجداول (constraints) :

هي شروط معينة توضع على الجداول لتنظيم العمليات المختلفة التي تتم فيها .

– أنواع القيود (types of constraints) :

1– (not null) : يمنع ترك الحقل فارغاً ، وهو على مستوى الحقل فقط .

2– (unique) : يمنع تكرار القيم داخل الحقل الواحد .

3– (primary key) : لعمل مفتاح أساسي داخل الجدول (وهذا الحقل يمنع ترك الحقل فارغاً ويمنع أيضاً تكرار القيم داخل الحقل) .

4– (foreign key) : لعمل مفتاح ربط بين جدولين .

5– (check) : لاختبار قيمة حقل بحيث لا يقبل هذا الحقل إلا قيم حسب شرط معين .

6– (default) : لوضع قيمة افتراضية لحقل ما .

– إضافة قيود على الجداول عند الإنشاء :

القيد (not null)

ويستخدم هذا القيد للبيانات التي يجب أن يدخلها المستخدم (أي البيانات التي يجب أن لا تترك فارغة) .

```
SQL> create table maher(
  2  name     varchar(20)  not null,
  3  no       number(5),
  4  age      number(3)
  5  );

Table created.
```

نلاحظ في المثال السابق أنه تم وضع القيد (not null) أي لا يترك فارغاً ، للحقل (name) .

القيد (unique)

ويستخدم هذا القيد للحقول التي لا يراد أن تكرر البيانات فيها ويمكن أن يوضع على مستوى الحقل أو على مستوى الجدول .

– إنشاء قيد على مستوى الحقل :

```
SQL> create table dept3
  2  (
  3  deptno   number(2),
  4  dname    varchar2(17) unique ,
  5  loc      varchar2(11)
  6  );

Table created.
```

في المثال السابق تم إضافة قيد للحقل (dname) على مستوى الحقل .

– إنشاء قيد على مستوى الجدول :

```
SQL> create table dept4
  2  (
  3  deptno  number(2) ,
  4  dname   varchar2(17) ,
  5  loc     varchar2(11) ,
  6  constraint dept4_dname_uk unique(dname)
  7  );

Table created.
```

في المثال السابق تم إضافة قيد للحقل (**dname**) على مستوى الجدول .

القيد (primary key)

ويستخدم هذا القيد لعدم تكرار البيانات ، وعدم قبول القيم الفارغة يعني أنه يقوم بعمل القيدين السابقين ، ويمكن أن يوضع على مستوى الحقل أو على مستوى الجدول .

– إنشاء قيد على مستوى الحقل :

```
SQL> create table dept5
  2  (
  3  deptno  number(2),
  4  dname   varchar2(17) primary key ,
  5  loc     varchar2(11)
  6  );

Table created.
```

في المثال السابق تم إضافة قيد للحقل (**dname**) على مستوى الحقل .

– إنشاء قيد على مستوى الجدول :

```
SQL> create table dept6
  2  (
  3  deptno  number(2) ,
  4  dname   varchar2(17) ,
  5  loc     varchar2(11) ,
  6  constraint dept6_dname_pk primary key(dname)
  7  );

Table created.
```

في المثال السابق تم إضافة قيد للحقل (**dname**) على مستوى الجدول ، ويمكن أيضاً وضع أكثر من قيد أثناء إنشاء الجدول على مستوى الجدول كما في المثال التالي :

```
SQL> create table dept7
  2  (
  3  deptno  number(2) ,
  4  dname   varchar2(17) ,
  5  loc     varchar2(11) ,
  6  constraint dept7_deptno_uk unique(deptno) ,
  7  constraint dept7_dname_pk primary key(dname)
  8  );

Table created.
```

القيد (foreign key)

يستخدم هذا القيد عندما نريد ربط جدولين ببعض ، ولربط حقل في جدول في حقل في جدول أخر يجب أن يكون الحقل في الجدول الأخر مفتاح أساسي (**primary key**) ، ويمكن أن يوضع على مستوى الحقل أو على مستوى الجدول .

– إنشاء قيد على مستوى الحقل :

```
SQL> create table dept8
  2  (
  3  deptno  number(2) primary key ,
  4  dname   varchar2(17) ,
  5  loc     varchar2(11)
  6  );

Table created.
```

في المثال السابق تم وضع الحقل (**deptno**) مفتاح أساسي (**primary key**) .

```
SQL> create table emp1
  2  (
  3  empno   number(4),
  4  ename   varchar2(17),
  5  job     varchar2(11),
  6  mgr     number(4),
  7  hiredat date ,
  8  sal     number(5,2),
  9  comm    number(5,2),
 10  deptno  number(2) references dept (deptno)
 11  );

Table created.
```

ثم تم وضع مفتاح أجنبي (**foreign key**) على الحقل (**deptno**) الذي هو مفتاح أساسي (**primary key**) في الجدول (**dept8**) على مستوى الحقل .

- إنشاء قيد على مستوى الجدول :

```
SQL> create table dept9
  2  (
  3  deptno  number(2) ,
  4  dname   varchar2(17) ,
  5  loc     varchar2(11) ,
  6  constraint dept9_deptno_pk primary key(deptno)
  7  );

Table created.
```

في المثال السابق تم وضع الحقل (**deptno**) مفتاح أساسي (**primary key**) .

```
SQL> create table emp2
  2  (
  3  empno     number(4),
  4  ename     varchar2(17),
  5  job       varchar2(11),
  6  mgr       number(4),
  7  hiredate  date,
  8  sal       number(5,2),
  9  comm      number(5,2),
 10  deptno    number(2),
 11  constraint emp2_deptno_fk foreign key(deptno)
 12  references dept9 (deptno)
 13  );

Table created.
```

ثم تم وضع مفتاح أجنبي (**foreign key**) على الحقل (**deptno**) الذي هو مفتاح أساسي (**primary key**) في الجدول (**dept9**) على مستوى الجدول .

القيد (check)

ويستخدم لاختبار قيمة حقل بحيث لا يقبل هذا الحقل إلا قيم حسب شرط معين ، ويمكن أن يوضع على مستوى الحقل أو على مستوى الجدول .

- إنشاء قيد على مستوى الحقل :

```
SQL> create table dept10
  2  (
  3  deptno  number(2) check(deptno between 5 and 85 ),
  4  dname   varchar2(17),
  5  loc     varchar2(11)
  6  );

Table created.
```

في المثال السابق تم وضع قيد للحقل (**deptno**) ، وهو أن يكون القيمة المدخلة بين (**5**) و (**85**) ، ولا يقبل قيم خارج هذا الشرط .

– إنشاء قيد على مستوى الجدول :

```
SQL> create table dept11
  2  (
  3  deptno  number(2),
  4  dname   varchar2(17),
  5  loc     varchar2(11),
  6  constraint dept11_deptno_ck
  7  check ( deptno between 5 and 85)
  8  );

Table created.
```

القيد (default)

ونستفيد منه في وضع قيمة افتراضية لحقل ما كما في المثال التالي :

```
SQL> create table student
  2  (
  3  st_no   number(4),
  4  st_name varchar2(15),
  5  st_age  number(2) default 25
  6  );

Table created.
```

في المثال السابق تم إضافة القيد (**default**) للحقل (**st_age**) ، وهو بجعل قيمة الحقل الافتراضية **25** .

– إضافة قيود على الجداول بعد الإنشاء :

ويتم ذلك باستخدام الأمر (**alter**) والشكل العام هو :

```
alter table table_name
add constraint constraint_name   constraint_type ;
```

```
SQL> alter table dept2
  2  add constraint dept_deptno_pk
  3  primary key (deptno);

Table altered.
```

في المثال السابق تم إضافة مفتاح أساسي (**primary key**) للجدول (**dept2**) المنشئ سابقاً .

```
SQL> alter table emp
  2  add constraint emp_empno_fk
  3  foreign key(empno)
  4  references emp(empno);

Table altered.
```

في المثال السابق تم إضافة مفتاح أجنبي (**foreign key**) للجدول (**emp**) المنشئ سابقاً .

الأخطاء:

```
SQL> alter table dept
  2  add constraint dept_deptno_pk
  3  primary key (deptno);
primary key (deptno)
*
ERROR at line 3:
ORA-02260: table can have only one primary key
```

في المثال السابق ظهرت رسالة خطأ ، وذلك بسبب محاولة إنشاء مفتاح أساسي (**primary key**) لجدول له مفتاح أساسي .

ملاحظة

– لا يمكن للجدول أن يكون به أكثر من مفتاح أساسي .

– إزالة القيود من الجداول :

وهذه جدول للأوامر التي تستخدم مع إزالة القيود :

الأمر	الوصف
cascade	يقوم بإسقاط المتعلقات (القيم) .
disable	يستخدم للإلغاء تفعيل الأمر .
enable	يستخدم لتفعيل الأمر .

```
SQL> alter table dept2
  2  drop primary key;

Table altered.
```

في المثال السابق تم إلغاء المفتاح الأساسي (**primary key**) من الجدول (**dept2**) .

```
SQL> alter table emp
  2  drop constraint emp_empno_fk;

Table altered.
```

في المثال السابق تم إلغاء المفتاح الأجنبي (**foreign key**) من الجدول (**emp**) .

```
SQL> alter table dept2
  2  drop primary key cascade;

Table altered.
```

في المثال السابق تم إلغاء المفتاح الأساسي (**primary key**) وإسقاط المتعلقات (القيم) .

```
SQL> alter table dept3
  2  disable constraint dept3_dname_pk cascade;
Table altered.
```

في المثال السابق تم تعديل جدول (dept3) بعدم تفعيل قيم المفتاح الأساسي (primary key) وإسقاط المتعلقات .

```
SQL> alter table dept3
  2  enable constraint dept3_deptno_pk;

Table altered.
```

في المثال السابق تم تعديل جدول (dept3) بتفعيل قيم المفتاح الأساسي (primary key) .

- استعراض القيود :

كل قيد له رمز معين وهذه جدول يوضح هذه الرموز :

الرمز	الوصف
p	يعني أن القيد مفتاح أساسي (primary key) .
r	يعني أن القيد مفتاح أجنبي (foreign key) .
c	يعني أن القيد من نوع (check) أو (not null) .
u	يعني أن القيد من النوع (unique) .

```
SQL> select constraint_name , constraint_type
  2  from user_constraints
  3  where table_name = 'EMP' or table_name= 'DEPT' ;

CONSTRAINT_NAME                C
------------------------------ -
PK_DEPT                        P
PK_EMP                         P
FK_DEPTNO                      R
```

في المثال السابق تم عرض أسماء المفاتيح ، والروابط المتعلقة بالجدولين (emp) و (dept) من جدول (user_constraints) .

2-5 العمليات على المناظير (views)

- إنشاء المناظير (creating views) :

وهو عنصر ضمن قاعدة البيانات يستخدم لتمثيل الجدول الذي تكون منه ، وهو لا يتم فيه تخزين البيانات بشكل فعلي فقط يتم من خلاله التعامل مع بيانات الجدول الأساسي فأي تغيرات تحدث لا تتم في المنظور ، وإنما في بيانات الجدول الأساسي .

والشكل العام لإنشاء المناظير هو :

```
create view view_name as select
column1 , column2 , ……  from table_name ;
```

```
SQL> create view emp_view as
  2  select empno,ename , job , sal
  3  from emp
  4  where sal > 2500 ;

View created.
```

في المثال السابق تم إنشاء جدول نظرة مكون من أربعة حقول من الجدول (**emp**) بشرط فقط بيانات الموظفين الذين رواتبهم أكبر من **2500** .

```
SQL> desc emp_view ;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 EMPNO                                     NOT NULL NUMBER(4)
 ENAME                                              VARCHAR2(10)
 JOB                                                VARCHAR2(9)
 SAL                                                NUMBER(7,2)
```

ولاستعراض بيانات المنظور (**emp_view**) :

```
SQL> select * from emp_view ;

     EMPNO ENAME      JOB              SAL
---------- ---------- --------- ----------
      7566 JONES      MANAGER         2975
      7698 BLAKE      MANAGER         2850
      7788 SCOTT      ANALYST         3000
      7839 KING       PRESIDENT       5000
      7902 FORD       ANALYST         3000
```

مثال آخر :

```
SQL> create or replace view emp_view
  2  ( e_number , e_name , e_job , e_sal )
  3  as select empno , ename , job , sal
  4  from emp2
  5  where deptno = 10 or deptno = 20 ;

View created.
```

تم في المثال السابق إنشاء أو استبدال لجدول النظرة (**emp_view**) مع عمل أسماء لحقول المنظور قبل تنفيذ الاستعلام الفرعي .

– إلغاء المناظير (canceling view) :

ويتم ذلك باستخدام الأمر (**drop**) كما في المثال التالي :

```
SQL> drop view emp_view ;

View dropped.
```

3-5 العمليات على المتسلسلات (sequences)

– إنشاء المتسلسلات (creating sequences) :

هو عنصر ضمن قاعدة البيانات يستخدم لتوليد أعداد صحيحة غير مكررة يمكن إضافتها لحقول المفتاح الأساسي (**primary key**) في الجدول .

وهذا الجدول يوضح الأوامر المستخدمة مع المتسلسلات :

الأمر	الوصف
increment by	تحديد قيمة الزيادة في المتتالية .
Start with	تحديد الرقم التي تبدأ بها المتتالية .
maxvalue	تحديد أكبر قيمة تصل إليها المتتالية .
cache	لحفظ عدد من القيم .
nocache	لعدم حفظ قيمة .
Cycle	للعودة في العد .
nocycle	لعدم العودة للعد .

مثال :

```
SQL> create sequence emp2_empno
  2  increment by 1
  3  start with 1
  4  maxvalue 50
  5  nocache
  6  nocycle ;

Sequence created.
```

في المثال السابق تم عمل متتالية للحقل (**empno**) في الجدول (**emp**) حيث حددت الزيادة بـــ **1** ، ويبدأ العد من الرقم (**1**) ، وأكبر قيمة تصل إليها المتتالية هو (**50**) ، وكذلك عدم حفظ القيمة وعدم العودة في العد .

– تعديل المتسلسلات (altering sequences) :

ويستخدم لهذا الأمر (**alter**) .

مثال :

```
SQL> alter sequence emp2_empno
  2  increment by 2
  3  maxvalue 100;

Sequence altered.
```

في المثال السابق تم تعديل عمل المتتالية حيث جعل الزيادة بـــ **2** ، وأكبر قيمة تصل إليها المتتالية هو (**100**) .

– إلغاء المتسلسلات (canceling sequences) :

ويستخدم لهذا الأمر (**drop**) .

مثال :

```
SQL> drop sequence emp2_empno ;

Sequence dropped.
```

في المثال السابق تم إلغاء المتسلسلة الموجود في الجدول (**emp2**) للحقل (**empno**) .

4-5 العمليات على الفهارس (indexes)

– إنشاء الفهارس (creating indexes) :

الفهرس : هو عنصر ضمن قاعدة البيانات يستخدم لترتيب ، وفرز بيانات الجدول الأساسي على أساس حقل أو حقول معينة ويتم تخزينه في جدول يسمى جدول الفهرسة (**dba_indexes**) .

والشكل العام هو :

```
create index  index_name
on table_name ( column1 , ……. ) ;
```

مثال :

```
SQL> create index dept3_deptno_idx
  2  on dept3(deptno);

Index created.
```

في المثال السابق تم إضافة فهرس للجدول (**dept3**) على الحقل (**deptno**) .

ملاحظات

– عدد الفهارس الممكن استخدامها على جدول 16 حقل مفهرس بشكل مستقل .

– عند إنشاء فهرس لحقول تم إنشاء فهرس سابق لها ، فإن قواعد البيانات أوراكل سيستخدم الفهرس الموجود كمصدر للمعلومات للفهرس الجديد .

– الحقول الفارغة (null) لن تظهر في الفهرس .

– المستخدم لا يرى هذه الفهارس ، وإنما هي لتسريع عملية الاستعلام فقط .

– إلغاء الفهارس (canceling indexes) :

ويستخدم لهذا الأمر (drop) .

مثال :

```
SQL> drop index dept3_deptno_idx ;

Index dropped.
```

في المثال السابق تم إلغاء فهرس الجدول (dept3) على الحقل (deptno) .

5-5 العمليات على المرادفات (synonym)

– إنشاء المرادفات (creating synonyms) :

المرادفات : هو عنصر ضمن قاعدة البيانات ، وهو عبارة عن تسمية تعطى لجدول أو منظور يتم استخدامها بعد ذلك للرجوع لهذا الجدول أو المنظور .

والشكل العام هو :

```
create synonym   synonym_name
for table_name ;
```

مثال :

```
SQL> create synonym  s_dept3 for dept3 ;

Synonym created.
```

في المثال السابق تم إنشاء مرادف للجدول (dept3) .

– إلغاء المرادفات (canceling synonyms) :

ويستخدم لهذا الأمر (drop) .

مثال :

```
SQL> drop synonym s_dept3 ;

Synonym dropped.
```

في المثال السابق تم إلغاء مرادف الجدول (dept3) .

6-5 العمليات على المستخدمين (users)

– إنشاء مستخدمين (creating user) :

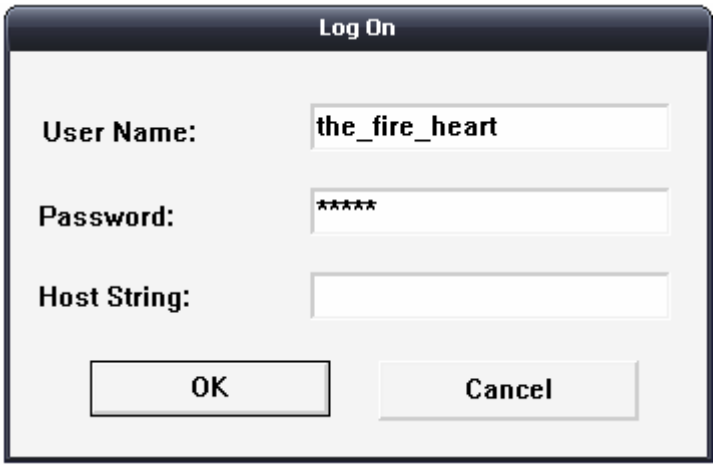
والشكل العام هو :

```
create user   user_name   identified by   password ;
```

عند إنشاء المستخدم يجب أن تكون متصل بمستخدم له صلاحيات عالية مثل (system) أو (sys) .

```
SQL> create user the_fire_heart identified by  maher ;

User created.
```



- التعديل على المستخدمين (altering users) :

```
SQL> alter user the_fire_heart  identified by  the_fire_heart ;

User altered.
```

في المثال السابق تم تغيير كلمة السر للمستخدم (the_fire_heart) .

ولمعرفة المساحة المتبقية للمستخدم على جدول مسافة معين نستخدم الأمر التالي :

```
SQL> select * from user_free_space ;

TABLESPACE_NAME                FILE_ID   BLOCK_ID      BYTES     BLOCKS RELATIVE_FNO
------------------------------ ---------- ---------- ---------- ---------- ------------
SYSTEM                              1      50529      65536          8            1
SYSTEM                              1      50857      65536          8            1
SYSTEM                              1      50873    2686976        328            1
UNDOTBS1                            2        329      65536          8            2
UNDOTBS1                            2        345     327680         40            2
UNDOTBS1                            2        393    4194304        512            2
UNDOTBS1                            2       1057     131072         16            2
UNDOTBS1                            2       1089      65536          8            2
UNDOTBS1                            2       1129      65536          8            2
UNDOTBS1                            2       1225  199688192      24376            2
CWMLITE                             3        737     262144         32            3

TABLESPACE_NAME                FILE_ID   BLOCK_ID      BYTES     BLOCKS RELATIVE_FNO
------------------------------ ---------- ---------- ---------- ---------- ------------
CWMLITE                             3       1233   10878976       1328            3
DRSYS                               4       1241   10813440       1320            4
EXAMPLE                             5      19025     131072         16            5
INDX                                6          9   26148864       3192            6
ODM                                 7       1193   11206656       1368            7
TOOLS                               8        777    4128768        504            8
USERS                               9          9   26148864       3192            9
XDB                                10       4857     196608         24           10

19 rows selected.
```

7-5 العمليات على النظام (system)

- وظائف النظام :

1- connect : وهي صلاحية الربط والاتصال بقاعدة البيانات ، وتعطى للمستخدمين العاديين الذين لا يحتاجون لإنشاء جداول ، وهي استخدام أوراكل ككل ، وتصبح هذه الصلاحية ذات معنى إذا وهب للمستخدم حق الوصول لجداول خاصة بمستخدمين آخرين ، وامتيازات تنفيذ الأوامر على هذه الجداول ، وبعد منح هذه الحقوق كلها فإن المستخدمين الذين يملكون الامتياز (**connect**) يمكنهم أيضاً إنشاء جداول ، ومناظير ، وسلاسل ، ومرادفات ، وربطه مع قواعد أخرى .

2- resource : (المصادر) وهي تعطى للمستخدمين الأكثر تعمقاً ، وهي حق إنشاء الجداول ، والسلاسل ، والمناظير ، والمرادفات ، و القوادح ، والفهارس ، والإجرائيان ، والتجمعات الخاصة بهم .

3- Dba : وتحوي كل السماحيات ، وكل امتيازات النظام بما فيها إعطاء الامتيازات و السماحيات لمستخدمين آخرين .

- وهناك سماحيات تعطى فقط لمدير قاعدة البيانات مثل :

- صلاحية التصدير لقاعدة البيانات (**exp_full_database**) .
- صلاحية الاستيراد لقاعدة البيانات (**imp_full_database**) .

- إنشاء وظائف للنظام (creating roles) :

ملاحظات

- يمكن إنشاء وظائف خاصة بالمستخدم في قاعدة بيانات أوراكل .
- يمكن إنشاء وظيفة ليس لها امتيازات ، والبدء بمنحها الامتيازات .
- يمكن منح الوظيفة كلمة سر سوء عند الإنشاء أو بعد منحها الصلاحيات أو قبل منحها الصلاحيات .
- عند إنشاء وظيفة يجب أن تكون متصل بمستخدم ذو صلاحيات عالية .

```
SQL> create role maher;

Role created.
```

في المثال السابق تم إنشاء وظيفة باسم (**maher**) دون وضع كلمة سر ، والوظائف الناتجة ليست لها أية صلاحيات عند الإنشاء .

```
SQL> alter role maher identified by the_fire_heart ;

Role altered.
```

في المثال السابق تم منح الوظيفة (**maher**) كلمة سر .

وتعتبر هذه الوظيفة غير نشطة حالياً ، وإذا أراد المستخدم تنشيط هذه الوظيفة يطلب إليه كلمة السر .

و لإعطاء الوظيفة امتيازات النظام يستخدم الأمر (**identified**) بدون (**by**) كما في المثال التالي :

```
SQL> alter role maher identified externally ;

Role altered.
```

ولحذف كلمة السر من الوظيفة نستخدم الأمر التالي :

```
SQL> alter role maher not identified ;

Role altered.
```

– تفعيل وإنهاء تفعيل الوظائف :

لتأهيل وظيفة غير معيارية نستخدم الأمر التالي :

```
SQL> set role maher ;

Role set.
```

لتأهيل كل الوظائف نستخدم الأمر التالي :

```
SQL> set role all;

Role set.
```

ولتأهيل كل الامتيازات للوظيفة (**maher**) نستخدم الأمر التالي :

```
SQL> set role all except maher ;

Role set.
```

وإذا كان للوظيفة كلمة سر يجب كتابتها .

```
SQL> set role maher identified by the_fire_heart ;

Role set.
```

ولإلغاء تفعيل وظيفة حالية نستخدم الأمر :

```
SQL> set role none ;

Role set.
```

أما لإلغاء وظيفة نستخدم الأمر التالي :

```
SQL> drop role maher ;

Role dropped.
```

8 -5 تمارين الفصل الخامس

-1

1- قم بإنشاء جدول باسمك ويحتوي على الحقول التالية :

- الرقم التسلسلي (s_no) من نوع البيانات الرقمية .
- الاسم (name) من نوع البيانات الحرفية .
- العمر (age) من نوع البيانات الرقمية .
- تاريخ الميلاد (birth_date) من نوع بيانات التاريخ .
- الدرجة (grade) من نوع البيانات الرقمية .

2- قم بتعديل اسم الحقل (grade) إلى (mark) .

3- قم بإضافة مفتاح أساسي (primary key) على حقل (s_no) .

4- قم بإضافة قيد التحقق (check) على حقل (age) حيث أن هذا الحقل يكون بين (18-38) .

5- قم بإنشاء جدول نظرة لجدولك .

-2

1- قم بإنشاء مستخدم (user) باسمك وضع لها كلمة السر (maher) .

2- قم بتغير كلمة السر إلى اسمك .

-3

1- قم بإنشاء وظيفة اسمها (stud) .

2- أمنح هذه الوظيفة كلمة سر .

3- أمنح هذه الوظيفة كل الامتيازات .

4- فعل الوظيفة ثم الغي تفعيلها .

5- أحذف كلمة السر من الوظيفة .

6- قم بإلغاء هذه الوظيفة .

(sql & sql * plus)

الفصل السادس

لغة التحكم في البيانات

Data Control Language

لغة التحكم في البيانات (DCL) Data Control Language :

وهي لغة تهتم بمنح وسحب الصلاحيات أو الامتيازات للمستخدمين (**users**) ، والوظائف (**roles**) .

6-1 منح الصلاحيات لمستخدم أو لوظيفة

ولمنح الصلاحيات للمستخدم أو للوظيفة نستخدم الأمر (**grant**) والشكل العام هو :

```
grant privilege1 , privilege2 , ……..
 on table_name to user_name or role_name ;
```

مثال :

```
SQL> grant connect , resource , dba to the_fire_heart;

Grant succeeded.
```

في المثال السابق تم منح صلاحية (**connect**) ، وصلاحية (**resource**) ، وصلاحية (**dba**) للمستخدم (**the_fire_heart**) .

```
SQL> connect the_fire_heart / maher ;
Connected.
```

والآن يمكننا أيضاً منح المستخدم (**the_fire_heart**) صلاحيات الاستيراد ، والتصدير كما في المثال التالي :

```
SQL> grant exp_full_database , imp_full_database
  2  to the_fire_heart ;

Grant succeeded.
```

مثال أخر :

```
SQL> grant create table to maher;

Grant succeeded.
```

في المثال السابق تم منح صلاحية إنشاء جداول (**create table**) للوظيفة (**maher**) .

وإذا أردنا أخذ صلاحيات وظيفة ومنحها لوظيفة أخرى :

1- نقوم بإنشاء وظيفة جديدة كما في المثال التالي :

```
SQL> create role alryashi ;

Role created.
```

2- نقوم بمنح هذه الوظيفة الجديدة صلاحيات الوظيفة (**maher**) كما في المثال التالي :

```
SQL> grant maher to alryashi ;

Grant succeeded.
```

ملاحظة

– يمكن منح صلاحية وظيفة لأكثر من وظيفة في نفس الوقت .

مثال أخر :

```
SQL> grant all on emp to public;

Grant succeeded.
```

في المثال السابق تم منح كل الصلاحيات الموجود على جدول (**emp**) لكل المستخدمين .

6- 2 سحب الصلاحيات من مستخدم أو وظيفة

ولسحب الصلاحيات على المستخدم أو الوظيفة نستخدم الأمر (**revoke**) والشكل العام هو :

```
revoke privilege1 , privilege2 , ……..
from table_name to user_name or role_name ;
```

مثال :

```
SQL> revoke connect , resource , dba from the_fire_heart ;

Revoke succeeded.
```

في المثال السابق تم سحب صلاحية (**connect**) ، وصلاحية (**resource**) ، وصلاحية (**dba**) من المستخدم (**the_fire_heart**) .

مثال أخر :

```
SQL> revoke create table from maher;

Revoke succeeded.
```

في المثال السابق تم سحب صلاحية إنشاء جداول (**create table**) للوظيفة (**maher**) .

6- 3 تمارين الفصل السادس

1- قم بمنح صلاحية (connect) والـ (resource) للمستخدم الذي قمت بإنشائه في الفص السابق .

2- قم بمنح صلاحيات (create table) و (alter table) للوظيفة (stud) التي قمت بإنشائه في الفصل السابق .

3- قم بسحب صلاحية (connect) والـ (resource) للمستخدم الذي قمت بإنشائه في الفص السابق .

4- قم بسحب صلاحيات (create table) و (alter table) للوظيفة (stud) التي قمت بإنشائه في الفصل السابق .

المصادر

1- sql international 9i التابع لشركة oracle .

2- موسوعة مبرمجي oracle 8i إعداد المهندس / ياسر رحال ، والمهندس / حسام عابد .

3- oracle 9i للمبتدئين تأليف / مايكل آبي ، مايك كوري ، ايان أبرامسون وترجمة د / خالد العامري .

4- لغة الاستعلام المهيكل (sql & sql * plus & pl/sql) oracle 9i أ / حلمي عبد الحليم الأسودي

5- كتاب (قواعد البيانات) التعليم المهني ، المملكة العربية السعودية .

الخاتمة

أتمنى من الله أن أكون قد وفقت في عملي هذا ، وأتمنى من الله أن ينال هذا الكتاب إعجابكم ورضاكم ، وفي الأخير أعتذر منكم على التقصير في بعض الأشياء .

الرجاء لا تنسونا من صالح الدعاء

السيرة الذاتية :

الاسم : ماهر محمد أحمد الرياشي

تاريخ الميلاد : 18 – 11 - 1989

المهنة : طالب في جامعة صنعاء – كلية العلوم

قسم الحاسوب – هندسة البرمجيات

المستوى : الثالث