EL LENGUAJE SQL

Los sistemas relacionales más comunes de BD consultan y modifican la base por medio de un lenguaje denominado SQL (*Structured Query Language* = Lenguaje de Consulta Estructurado). Un núcleo importante de él equivale al álgebra relacional.

Originalmente, SQL se llamaba SEQUEL (*Structured English QUEry Language*) y fue diseñado e implementado por IBM Research como interfaz para un SGBD relacional experimental denominado SYSTEM R. Un esfuerzo conjunto de ANSI (American National Standards Institute) e ISO (International Standards Organization) ha dado lugar a una versión estándar de SQL (ANSI 1986) llamada SQL-86 o SQL1. Posteriormente, se ha desarrollado un estándar revisado y más expandido llamado SQL2 (también llamado SQL-92), que es el que se presenta a continuación. Ya existen planes para SQL3, que extiende SQL con conceptos de orientación a objetos y otros conceptos recientes de bases de datos, como se verá más adelante.

SQL es un lenguaje de BD global; cuenta con enunciados de definición, consulta y actualización de datos. Así pues, es tanto un lenguaje de definición de datos (LDD) como un lenguaje de manipulación de datos (LMD). Además, cuenta con mecanismos para definir vistas de la BD, para especificar seguridad y autorización, para definir restricciones de integridad, y para especificar controles de transacciones. También tiene reglas para insertar sentencias de SQL en lenguajes de programación de propósito general como C, Pascal o Java.

Existe una diferencia muy importante entre SQL y el modelo relacional formal: SQL permite que tablas (relaciones) tengan dos o más tuplas idénticas en todos los valores de sus atributos. Por tanto, en general, **una tabla de SQL no es un CONJUNTO de tuplas** ya que los conjuntos no permiten dos miembros idénticos; más bien, es un **MULTICONJUNTO** (bolsa) de tuplas.

SQL como DML (Lenguaje de Manipulación de Datos)

1 Consulta de datos

Antes de realizar ninguna consulta, veamos las tablas con las que vamos a trabajar. Se utilizan las tablas EMP (empleados) y DEPT (departamentos). Para ver su descripción utilizaremos la sentencia "`DESCRIBE tabla`". DESCRIBE no es una orden de SQL, es un comando del SQL*Plus de Oracle que nos indica las columnas (atributos) de una tabla, indicando además su tipo y si acepta o no valores nulos.

```
DESCRIBE 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)
```

siendo: EMPNO: código de empleado
ENAME: nombre de empleado
JOB: puesto de trabajo
MGR: código de supervisor
HIREDATE: fecha de contratación
SAL: salario
COMM: comisión
DEPTNO: número de departamento donde trabaja

```
DESCRIBE DEPT
```

```
Name                          Null?    Type
----------------------------- -------- ----
DEPTNO                                 NUMBER(2)
DNAME                                  VARCHAR2(14)
LOC                                    VARCHAR2(13)
```

siendo: DEPTNO: número de departamento
DNAME: nombre de departamento
LOC: localidad

El contenido de las tablas es:

empno	ename	job	mgr	hiredate	sal	comm	deptno
7369	SMITH	CLERK	7902	17/12/1980	800		20
7499	ALLEN	SALESMAN	7698	20/02/1981	1600	300	30
7521	WARD	SALESMAN	7698	22/02/1981	1250	500	30
7566	JONES	MANAGER	7839	02/04/1981	2975		20
7654	MARTIN	SALESMAN	7698	28/09/1981	1250	1400	30
7698	BLAKE	MANAGER	7839	01/05/1981	2850		30
7782	CLARK	MANAGER	7839	09/06/1981	2450		10
7788	SCOTT	ANALYST	7566	09/12/1982	3000		20
7839	KING	PRESIDENT		17/11/1981	5000		10
7844	TURNER	SALESMAN	7698	08/09/1981	1500	0	30
7876	ADAMS	CLERK	7788	12/01/1983	1100		20
7900	JAMES	CLERK	7698	03/12/1981	950		30
7902	FORD	ANALYST	7566	03/12/1981	3000		20
7934	MILLER	CLERK	7782	23/01/1982	1300		10

```
deptno dname          loc
    10 ACCOUNTING     NEW YORK
    20 RESEARCH       DALLAS
    30 SALES          CHICAGO
    40 OPERATIONS     BOSTON
```

El formato de la sentencia SELECT general es

```
SELECT columnas
       FROM tablas
       [WHERE condiciones_where]
       [GROUP BY columnas_group]
       [HAVING condiciones_having]
       [ORDER BY columnas_orden]
```

Esta sentencia SELECT ***no tiene relación con la operación SELECCIÓN σ del álgebra relacional***.

1.1 Selección de atributos

La consulta más sencilla es seleccionar todas las tuplas de una tabla:

"Selecciona los datos de todos los empleados"

```
SELECT * FROM EMP;
```

```
  EMPNO ENAME      JOB           MGR HIREDATE      SAL    COMM  DEPTNO
------- ---------- --------- ------- --------- ------- ------- -------
   7369 SMITH      CLERK        7902 17/12/1980     800              20
   7499 ALLEN      SALESMAN     7698 20/02/1981    1600     300      30
   7521 WARD       SALESMAN     7698 22/02/1981    1250     500      30
   7566 JONES      MANAGER      7839 02/04/1981    2975              20
   7654 MARTIN     SALESMAN     7698 28/09/1981    1250    1400      30
   7698 BLAKE      MANAGER      7839 01/05/1981    2850              30
   7782 CLARK      MANAGER      7839 09/06/1981    2450              10
   7788 SCOTT      ANALYST      7566 09/12/1982    3000              20
   7839 KING       PRESIDENT         17/11/1981    5000              10
   7844 TURNER     SALESMAN     7698 08/09/1981    1500       0      30
   7876 ADAMS      CLERK        7788 12/01/1983    1100              20
   7900 JAMES      CLERK        7698 03/12/1981     950              30
   7902 FORD       ANALYST      7566 03/12/1981    3000              20
   7934 MILLER     CLERK        7782 23/01/1982    1300              10
```

Se utiliza el asterisco (*) como comodín para seleccionar todos los campos. La sentencia anterior es equivalente a

```
SELECT EMPNO, ENAME, JOB, MGR, HIREDATE, SAL, COMM, DEPTNO
FROM EMP;
```

Se pueden seleccionar columnas individuales: ***"Lista todos los salarios"***

```
SELECT SAL FROM EMP;
```

```
    SAL
-------
    800
   1600
   1250
   2975
   1250
   2850
   2450
   3000
   5000
   1500
   1100
    950
   3000
   1300
```

Se pueden seleccionar constantes, por ejemplo la siguiente consulta:

```
SELECT 'SUELDO:', SAL
       FROM EMP
```

obtendría

```
'SUELDO SAL
------- ------
SUELDO:  800
SUELDO: 1600
SUELDO: 1250
SUELDO: 2975
SUELDO: 1250
SUELDO: 2850
SUELDO: 2450
SUELDO: 3000
SUELDO: 5000
SUELDO: 1500
SUELDO: 1100
SUELDO:  950
SUELDO: 3000
SUELDO: 1300
```

También se pueden hacer operaciones con los campos seleccionados, como

"***Nombre de todos los empleados y el doble del salario que tienen***"

```
SELECT ENAME, SAL*2
       FROM EMP
```

Si alguno de los atributos resultantes de la consulta puede tomar valores nulos, es posible visualizar un valor por defecto mediante el operador NVL:

NVL(campo, valor2)

devuelve el valor del campo salvo que sea nulo, en cuyo caso devuelve el valor *valor2*.

"Selecciona los nombres de los empleados, sus salarios y comisiones. Indicar con el valor -1 los casos en los que la comisión sea nula"

```
SELECT ENAME, SAL, NVL(COMM,-1) FROM EMP;
```

```
ENAME      SAL    COMM
---------- ------- -------
SMITH      800       -1
ALLEN      1600     300
WARD       1250     500
JONES      2975      -1
MARTIN     1250    1400
BLAKE      2850      -1
CLARK      2450      -1
SCOTT      3000      -1
KING       5000      -1
TURNER     1500       0
ADAMS      1100      -1
JAMES      950       -1
FORD       3000      -1
MILLER     1300      -1
```

Los gestores relacionales, y en concreto Oracle, no eliminan los duplicados cuando hacen un SELECT, como se ha mencionado anteriormente. Si se quieren eliminar hay que indicarlo explícitamente utilizando DISTINCT:

"Lista todos los salarios diferentes"

```
SELECT DISTINCT SAL FROM EMP;
```

```
   SAL
-------
    800
    950
   1100
   1250
   1300
   1500
   1600
   2450
   2850
   2975
   3000
   5000
```

La eliminación de duplicados es un proceso costoso, ya que es necesario que el gestor realice una ordenación de la tabla resultante para que las tuplas idénticas aparezcan contiguas. El tiempo necesario para la ordenación es, en muchas ocasiones, mayor que el necesario para realizar la consulta. Por ello, debemos utilizar la eliminación de duplicados con prudencia, si queremos que la consulta se ejecute rápidamente.

1.2 Ordenamiento de los resultados

Por defecto, SQL no ordena el resultado:

"Lista todos los empleados con sus salarios"

```
SELECT SAL, ENAME FROM EMP;
```

```
    SAL ENAME
------- ----------
    800 SMITH
   1600 ALLEN
   1250 WARD
   2975 JONES
   1250 MARTIN
   2850 BLAKE
   2450 CLARK
   3000 SCOTT
   5000 KING
   1500 TURNER
   1100 ADAMS
    950 JAMES
   3000 FORD
   1300 MILLER
```

Para ordenarlos, se utiliza la cláusula **ORDER BY**:

- `ORDER BY campo1 [ASC|DESC], campo2 [ASC|DESC], ...`
- Los órdenes se indican para cada campo. Si se indica **`ORDER BY campo1`** (u `ORDER BY campo1` **ASC**) la ordenación es ascendente, y si se indica **`ORDER BY campo1 DESC`** el orden es descendente.

"Lista todos los empleados con sus sueldos, ordenando el resultado primero por su salario y luego por el nombre en orden descendente"

```
SELECT SAL, ENAME
       FROM EMP
       ORDER BY SAL, ENAME DESC;
```

```
    SAL ENAME
------- ----------
    800 SMITH
    950 JAMES
   1100 ADAMS
   1250 WARD
   1250 MARTIN
   1300 MILLER
   1500 TURNER
   1600 ALLEN
   2450 CLARK
   2850 BLAKE
   2975 JONES
   3000 SCOTT
   3000 FORD
   5000 KING
```

1.3 Condiciones para restringir la consulta

1.3.1 Cláusula WHERE

Para restringir las tuplas o filas que se obtienen, se pueden imponer condiciones. Para ello se usa la cláusula **WHERE**. Cuando SQL aplica una cláusula WHERE a una tabla, se eliminan todas las filas de dicha tabla para las cuales la expresión de WHERE es falsa o nula.

"Lista los datos de los empleados cuyo salario sea 800"

```
SELECT * FROM EMP
       WHERE SAL=800;
```

```
 EMPNO ENAME      JOB           MGR HIREDATE      SAL    COMM  DEPTNO
------- ---------- --------- ------- --------- ------- ------- -------
   7369 SMITH      CLERK        7902 17/12/1980     800              20
```

En la cláusula WHERE hay distintos tipos de condiciones*:*

- **>, >=, <, <=, =, <>**: Estos operadores se pueden utilizar con todos los tipos de datos, para compararlos según una relación de orden. Si los datos son números sigue la ordenación normal, si son cadenas de caracteres sigue la ordenación ASCII, si son fechas la ordenación temporal, etc.
- Cuando se realizan comparaciones con cadenas de caracteres o con fechas, estas deben estar entre comillas simples('):

"Lista los empleados contratados ('hiredate' es la fecha de contratación) después del 17 de diciembre de 1980"

```
SELECT * FROM EMP
       WHERE HIREDATE > '17/12/1980';
```

```
  EMPNO ENAME      JOB           MGR HIREDATE      SAL    COMM  DEPTNO
------- ---------- --------- ------- --------- ------- ------- -------
   7499 ALLEN      SALESMAN     7698 20/02/1981    1600     300       30
   7521 WARD       SALESMAN     7698 22/02/1981    1250     500       30
   7566 JONES      MANAGER      7839 02/04/1981    2975               20
   7654 MARTIN     SALESMAN     7698 28/09/1981    1250    1400       30
   7698 BLAKE      MANAGER      7839 01/05/1981    2850               30
   7782 CLARK      MANAGER      7839 09/06/1981    2450               10
   7788 SCOTT      ANALYST      7566 09/12/1982    3000               20
   7839 KING       PRESIDENT         17/11/1981    5000               10
   7844 TURNER     SALESMAN     7698 08/09/1981    1500       0       30
   7876 ADAMS      CLERK        7788 12/01/1983    1100               20
   7900 JAMES      CLERK        7698 03/12/1981     950               30
   7902 FORD       ANALYST      7566 03/12/1981    3000               20
   7934 MILLER     CLERK        7782 23/01/1982    1300               10
```

1.3.2 Selección de rangos: operador BETWEEN

- *WHERE campo BETWEEN limite_inferior AND limite_superior.*
- Es equivalente a (*campo>=limite_inferior and campo <=limite_superior*)

"Lista los empleados cuyo número esté entre el 7499 y el 7654"

```
SELECT * FROM EMP
        WHERE EMPNO  BETWEEN 7499 AND 7654;
```

```
  EMPNO ENAME      JOB           MGR HIREDATE      SAL    COMM  DEPTNO
------- ---------- --------- ------- --------- ------- ------- -------
   7499 ALLEN      SALESMAN     7698 20/02/1981    1600     300      30
   7521 WARD       SALESMAN     7698 22/02/1981    1250     500      30
   7566 JONES      MANAGER      7839 02/04/1981    2975              20
   7654 MARTIN     SALESMAN     7698 28/09/1981    1250    1400      30
```

Las condiciones se unen con conectivas lógicas AND y OR, y se puede usar también la negación, NOT.

"Lista los empleados cuyo número esté entre el 7499 y el 7654 y que ganen más de 1250"

```
SELECT * FROM EMP
       WHERE EMPNO  BETWEEN 7499 AND 7654
       AND   SAL > 1250;
```

```
  EMPNO ENAME      JOB           MGR HIREDATE      SAL    COMM  DEPTNO
------- ---------- --------- ------- --------- ------- ------- -------
   7499 ALLEN      SALESMAN     7698 20/02/1981    1600     300      30
   7566 JONES      MANAGER      7839 02/04/1981    2975              20
```

Además, las condiciones unidas por AND, OR y NOT admiten paréntesis (la prioridad, de mayor a menor, es NOT, AND y OR).

"Lista los empleados cuyo número esté entre el 7499 y el 7654 y que ganen más de 1250 o su nombre sea WARD"

```
SELECT * FROM EMP
       WHERE EMPNO  BETWEEN 7499 AND 7654
       AND (SAL > 1250 OR ENAME='WARD');
```

```
  EMPNO ENAME      JOB           MGR HIREDATE      SAL    COMM  DEPTNO
------- ---------- --------- ------- --------- ------- ------- -------
   7499 ALLEN      SALESMAN     7698 20/02/1981    1600     300      30
   7521 WARD       SALESMAN     7698 22/02/1981    1250     500      30
   7566 JONES      MANAGER      7839 02/04/1981    2975              20
```

1.3.3 Operador (NOT) LIKE

Se usa solo con cadenas de caracteres. Es una comparación de igualdad pero admite comodines:

- **%**: se puede sustituir por cualquier número de caracteres (0 o más)
- _: se sustituye por 1 carácter[1]

Por ejemplo, *ename LIKE '_'* buscaría los nombres de 1 carácter,

mientras que *ename LIKE '_ _%'* buscaría los nombres con 2 o más caracteres.

"Busca los empleados con cinco letras o menos"

```
SELECT * FROM EMP
       WHERE ENAME NOT LIKE '_ _ _ _ _ _%'
```

```
EMPNO ENAME      JOB           MGR HIREDATE      SAL    COMM  DEPTNO
------- ---------- --------- ------- --------- ------- ------- -------
   7369 SMITH      CLERK        7902 17/12/1980     800              20
   7499 ALLEN      SALESMAN     7698 20/02/1981    1600     300      30
   7521 WARD       SALESMAN     7698 22/02/1981    1250     500      30
   7566 JONES      MANAGER      7839 02/04/1981    2975              20
   7698 BLAKE      MANAGER      7839 01/05/1981    2850              30
   7782 CLARK      MANAGER      7839 09/06/1981    2450              10
   7788 SCOTT      ANALYST      7566 09/12/1982    3000              20
   7839 KING       PRESIDENT         17/11/1981    5000              10
   7876 ADAMS      CLERK        7788 12/01/1983    1100              20
   7900 JAMES      CLERK        7698 03/12/1981     950              30
   7902 FORD       ANALYST      7566 03/12/1981    3000              20
```

Busca los empleados con 6 letras o más y, mediante NOT LIKE, obtiene el complementario: 5 letras o menos.

1.3.4 Operador IN

Se puede usar con cualquier tipo de datos, y busca la pertenencia de un elemento en un conjunto. Es equivalente a "=ANY".

"Lista los empleados cuyo número sea 7499 ó 7566"

```
SELECT * FROM EMP
       WHERE EMPNO IN (7499,7566)
```

```
  EMPNO ENAME      JOB           MGR HIREDATE      SAL    COMM  DEPTNO
------- ---------- --------- ------- --------- ------- ------- -------
   7499 ALLEN      SALESMAN     7698 20/02/1981    1600     300      30
   7566 JONES      MANAGER      7839 02/04/1981    2975              20
```

campo IN *R* da como resultado Cierto si, y sólo si, *campo* es igual a uno de los valores de *R*.

Más adelante se verán otras posibilidades para el uso de IN con subconsultas.

[1] En estas notas, cuando se utilice LIKE con subrayados (_), se separarán por espacios para poder contarlos, de lo contrario sólo se vería una línea. Si se escribe la sentencia SELECT, se deben omitir los espacios, a no ser que explícitamente se indique lo contrario.

1.3.5 Operador NOT IN

Es equivalente al operador "<> ALL". *campo* NOT IN *R* da como resultado Cierto si, y sólo si, *campo* NO es igual a ningún valor de *R*.

Nunca se evalúa como Cierto si algún miembro del conjunto es nulo. Es decir: **deptno NOT IN (5, 15, nulo)** se evalúa como:

deptno <> 5 AND deptno <> 15 AND deptno <> nulo

e, independientemente del resultado de deptno <> 5 AND deptno <> 15 dará como resultado FALSO o DESCONOCIDO, ya que

FALSO AND DESCONOCIDO = FALSO

CIERTO AND DESCONOCIDO = DESCONOCIDO

El tratamiento de valores nulos exige definir:

- operaciones de comparación
- operaciones aritméticas
- operaciones algebraicas
- funciones de agregación

de forma específica para el caso de que alguno de los operandos tome valores nulos, y obliga también a introducir nuevos operadores especiales, utilizando lógica trivaluada.

1.4 Funciones de agregación

Existen funciones que permiten calcular, desde una sentencia SQL, sumas, medias aritméticas, etc. de datos. Todas ellas, excepto COUNT(*) ignoran cualquier nulo en sus argumentos. Algunas de las más importantes se presentan a continuación

1.4.1 Funciones para datos numéricos

- **AVG** ([DISTINCT] Atributo): media aritmética
- **MIN** (Atributo): valor mínimo
- **MAX** (Atributo): valor máximo

- **SUM** ([DISTINCT] Atributo): suma
- **STDDEV** (Atributo): desviación estándar
- **VARIANCE** (Atributo): varianza
- **ABS** (Atributo): valor absoluto

- **CEIL** (Atributo): entero más próximo por encima
- **FLOOR** (Atributo): entero más próximo por debajo

- **MOD** (Atributo, Atributo2): resto de dividir *Atributo* entre *Atributo2*. Si *Atributo2*=0 devuelve *Atributo*
- **POWER** (Atributo, Atributo2): *Atributo* $^{\text{Atributo2}}$
- **ROUND** (Atributo [, m]): redondeo de *Atributo* a *m* dígitos
- **TRUNC** (Atributo [, m]): trunca *Atributo* a *m* dígitos

- **SIGN** (Atributo):
  - Si *Atributo* < *0* devuelve (-1)
  - Si *Atributo* = *0* devuelve 0
  - Si *Atributo* > *0* devuelve 1
- **SQRT** (Atributo): raíz cuadrada del valor del *Atributo*

- **COUNT** ([DISTINCT] Atributo): Cuenta el número de filas *donde expresión no es nulo*
- **COUNT** (*): Cuenta el número de filas *incluyendo aquellas con nulos*

"Lista el sueldo mínimo, máximo y medio de los empleados, y el número total de éstos"

```
SELECT MIN(SAL),MAX(SAL),AVG(SAL), COUNT(*)
       FROM EMP
```

```
MIN(SAL) MAX(SAL) AVG(SAL) COUNT(*)
-------- -------- -------- --------
     800     5000  2073.21       14
```

1.4.2 Funciones para datos de tipo carácter

- **INITCAP** (Atributo): pone a mayúscula la primera letra de cada palabra contenida en *Atributo*.
- **LOWER** (Atributo): transforma *Atributo* a minúsculas.
- **UPPER** (Atributo): transforma *Atributo* a mayúsculas.
- **LTRIM** (Atributo [, caracter]): elimina el carácter *caracter* (o blancos) por la izquierda hasta encontrar el primer carácter que no está en *caracter*.
- **RTRIM** (Atributo [, caracter]): elimina el carácter *caracter* (o blancos) por la derecha hasta encontrar el primer carácter que no está en *caracter*.
- **SUBSTR** (Atributo, m[, n]): devuelve una porción de *Atributo* comenzando en el carácter *m* y en *n* caracteres de longitud.
- **LENGTH** (Atributo): devuelve la longitud de *Atributo.*

1.4.3 Funciones para datos de tipo fecha

Existe una amplia lista de funciones para ayudar a la manipulación de datos de tipo fecha. La información sobre la fecha se encuentra en una tabla del diccionario de datos, denominada *dual*. Las funciones más importantes son:

- **sysdate**: devuelve la fecha y hora actual.
  - Ej: select sysdate from dual;
    - Resultado: 28-FEB-03 si el día actual es 28 de febrero de 2003
- **last_day:** último día del mes
  - Ej: select last_day(sysdate) from dual;
    - Resultado: 31-MAR-03 si el día actual es 12 de marzo de 2003
- **add_months(*d, n*):** suma o resta *n* meses a partir de la fecha *d*
  - Ej: select add_months(sysdate, 2) from dual;
    - Resultado: 18-MAY-03 si el día actual es 18 de marzo de 2003
- **months_between(*f, s*):** diferencia en meses entre la fecha *f* y la fecha *s*
  - Ej: select months_between(sysdate, '12-MAR-03') from dual;
    - Resultado: 13 si el mes actual es abril de 2003
- **next_day(*d, day*):** la fecha del día especificado de la semana después del día actual
  - Ej: select next_day(sysdate, 'Lunes') from dual;
    - Resultado: 20-OCT-03 si el día actual es 14 de octubre de 2003

1.5 Funciones de conversión

- **TO_CHAR** (Atributo[, formato]): convierte *Atributo (numérico* o *fecha)* a un string con el formato especificado.

```
Ej:  select to_char (17145, '$099,999')
     from dual;

     select to_char (hiredate, 'Month DD, YYYY')
     from emp
     where ename = 'SMITH';
```

Posibles máscaras para fecha: Y, YY, YYY, YYYY, YEAR (en letra)
MONTH (nombre completo), MON (3 letras), MM (01-12)
DDD (1-366), DD (1-31), D(1-7), DAY (en letra), DY (3 letras)

- **TO_DATE** (Atributo[, formato]): convierte *Atributo* a fecha con el formato especificado.
  Ej: select to_date ('12-DEC-03')
  from dual;
  Ej: select to_date ('20031227', 'YYYYMMDD')
  from dual;
  devuelve la fecha 27-DEC-03

- **TO_NUMBER** (Atributo[, formato]): convierte *Atributo* a un número.

1.6 Agrupamiento

Las funciones de agregación se suelen utilizar combinadas con la cláusula de agrupamiento **GROUP BY**, que agrupa el resultado por una serie de atributos:

"Lista los números de departamento y la suma de los salarios de cada uno de ellos"

```
SELECT DEPTNO, SUM(SAL)
       FROM EMP
        GROUP BY DEPTNO;
```

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

Lógicamente, no se pueden seleccionar atributos que no se puedan agrupar por los atributos indicados en el GROUP BY. Por ejemplo, la sentencia

```
SELECT ENAME, DEPTNO, SUM(SAL) -- ¡¡¡Error!!!
       FROM EMP
        GROUP BY DEPTNO;
```

daría un error, ya que en cada departamento el nombre del empleado no se puede agrupar por departamento.

Cada expresión de SELECT debe ser:

1. Una constante o una función sin parámetros (p.ej: SYSDATE)
2. Contener una función de agrupación (SUM,COUNT, MAX, ...)
3. Emparejar con una expresión del GROUP BY

1.6.1 Restricciones en los agrupamientos

Cuando se selecciona un conjunto de atributos agrupados por uno o más atributos, se pueden imponer condiciones a los grupos (es decir, condiciones a los atributos que se están seleccionando). Es la cláusula **HAVING**, que sería el equivalente a la cláusula WHERE pero aplicada a los grupos. Es decir, elimina los grupos para los cuales la expresión HAVING da como resultado FALSO o DESCONOCIDO. Por ejemplo,

"Lista la suma de los sueldos agrupada por departamentos, pero sólo aquellos en los que la suma sea mayor que 10000, o que el departamento sea el 30"

```
SELECT SUM(SAL), DEPTNO
       FROM EMP
       GROUP BY DEPTNO
       HAVING SUM(SAL)>10000
              OR DEPTNO=30;
```

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

1.7 Consultas a más de una tabla (θ-joins)

Hasta ahora hemos visto consultas que se aplican a una tabla. Para realizar consultas a más de una, basta con indicar las tablas en la cláusula FROM y añadir las condiciones necesarias en la cláusula WHERE.

"Lista los empleados y los nombres de departamento a que pertenecen"

```
SELECT ENAME, DNAME
       FROM EMP, DEPT
       WHERE EMP.DEPTNO = DEPT.DEPTNO;
```

```
ENAME      DNAME
---------- --------------
CLARK      ACCOUNTING
KING       ACCOUNTING
MILLER     ACCOUNTING
SMITH      RESEARCH
ADAMS      RESEARCH
FORD       RESEARCH
SCOTT      RESEARCH
JONES      RESEARCH
ALLEN      SALES
BLAKE      SALES
MARTIN     SALES
JAMES      SALES
TURNER     SALES
WARD       SALES
```

La consulta anterior hace un join natural entre las tablas EMP y DEPT, y luego proyecta sólo los nombres de los empleados y departamentos. Los nombres de los atributos, en caso de que las tablas tengan atributos con el mismo nombre, irán precedidos por el nombre de la tabla y un punto, como en `EMP.DEPTNO`. Si no existe confusión posible pueden indicarse sin el nombre de la tabla, como por ejemplo `DNAME`, que sólo existe en la tabla DEPT.

Además, tanto para las tablas como los atributos pueden especificarse **sinónimos**, simplemente indicando el atributo o tabla seguido de su sinónimo: La siguiente consulta obtendría exactamente el mismo resultado que la anterior, simplemente en el encabezado se indicaría "EMPLEADO" y "DEPARTAMENTO" en vez de "ENAME" y "DNAME":

```
SELECT ENAME Empleado, DNAME Departamento
       FROM EMP E, DEPT D
        WHERE E.DEPTNO = D.DEPTNO
```

La utilidad principal de los sinónimos es que podemos referenciar a la misma tabla por varios sinónimos, con lo que realmente tratamos con varias copias de una tabla.

1.8 Subconsultas

El lenguaje SQL permite el anidamiento de consultas. Veamos un ejemplo.

"Datos de los empleados del departamento 30"

```
SELECT * FROM EMP
        WHERE DEPTNO=30;
```

```
   EMPNO ENAME      JOB           MGR HIREDATE      SAL    COMM  DEPTNO
------- ---------- --------- ------- --------- ------- ------- -------
   7499 ALLEN      SALESMAN     7698 20/02/81    1600     300      30
   7521 WARD       SALESMAN     7698 22/02/81    1250     500      30
   7654 MARTIN     SALESMAN     7698 28/09/81    1250    1400      30
   7698 BLAKE      MANAGER      7839 01/05/81    2850              30
   7844 TURNER     SALESMAN     7698 08/09/81    1500       0      30
   7900 JAMES      CLERK        7698 03/12/81     950              30
```

Sin embargo, si queremos saber los datos de los empleados del departamento de ventas (SALES), y no sabemos a qué número de departamento corresponde, podemos utilizar la siguiente consulta:

```
SELECT * FROM EMP
        WHERE EMP.DEPTNO=(SELECT DEPTNO FROM DEPT
                                  WHERE DNAME='SALES');
```

```
  EMPNO ENAME      JOB           MGR HIREDATE      SAL    COMM  DEPTNO
------- ---------- --------- ------- --------- ------- ------- -------
   7499 ALLEN      SALESMAN     7698 20/02/1981    1600     300      30
   7521 WARD       SALESMAN     7698 22/02/1981    1250     500      30
   7654 MARTIN     SALESMAN     7698 28/09/1981    1250    1400      30
   7698 BLAKE      MANAGER      7839 01/05/1981    2850              30
   7844 TURNER     SALESMAN     7698 08/09/1981    1500       0      30
   7900 JAMES      CLERK        7698 03/12/1981     950              30
```

En ella hay una consulta de primer nivel en la que seleccionamos los datos del empleado, y en la cláusula WHERE utilizamos una nueva (sub)consulta en la que averiguamos el número de departamento correspondiente a "SALES", para poder así hacer la restricción.

Nótese que utilizamos el operador de igualdad en `EMP.DEPTNO=(SELECT...)` porque tenemos la seguridad de que la subconsulta nos va a devolver una sola tupla. Si el resultado pueden ser varias tuplas, utilizaremos el operador **IN** (o **NOT IN** para obtener el conjunto complementario):

"Lista los nombres de empleados de los departamentos SALES y RESEARCH"

```
SELECT ENAME FROM EMP E
WHERE E.DEPTNO IN (SELECT DEPTNO
                     FROM DEPT
                     WHERE DNAME IN ('SALES','RESEARCH'));
```

```
ENAME
----------
SMITH
ADAMS
FORD
SCOTT
JONES
ALLEN
BLAKE
MARTIN
JAMES
TURNER
```

```
WARD
```
Si el resultado de la subconsulta fuese una tabla sin filas, el resultado sería DESCONOCIDO.

1.9 Consultas correlacionadas

Se trata de subconsultas que se evalúan para cada fila de la instrucción SQL exterior (y no para la consulta entera):

```
SELECT atributos
FROM tabla1
WHERE expresion OPERADOR (SELECT atributos
                          FROM tabla2
                          WHERE tabla1.atrib1 operador tabla2.atrib2)
```

Permite responder a cuestiones cuya respuesta depende del valor en cada fila de la consulta más exterior. Por ejemplo, ¿qué empleados ganan más que el salario medio de su departamento? donde para cada tupla padre se calcula el salario medio de su departamento

Este tipo de consultas también se utilizan cuando se desea actualizar (o borrar) filas de una tabla basadas en filas de otra tabla.

1.10 La cláusula (NOT) EXISTS

La cláusula **EXISTS** (o **NOT EXISTS**) comprueba si una subconsulta devuelve algún valor (EXISTS) o no devuelve ninguno (NOT EXISTS). Por ejemplo,

"Lista los nombres de los departamentos que no hayan contratado a nadie el 3 de diciembre de 1981"

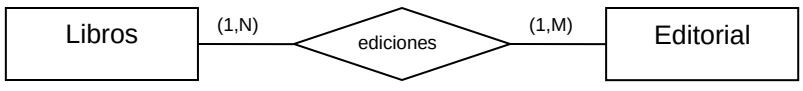
```
SELECT D.DNAME
       FROM DEPT D
       WHERE NOT EXISTS   (SELECT * FROM EMP E
                           WHERE E.DEPTNO=D.DEPTNO
                           AND HIREDATE ='03/12/1981')
```

```
DNAME
--------------
ACCOUNTING
OPERATIONS
```

EXISTS y NOT EXISTS casi siempre se usan junto con una consulta anidada correlacionada. En el caso anterior, la consulta de primer nivel busca en la tabla de departamentos los nombres y, para cada fila, comprueba —mediante la subconsulta— que para ese número de departamento no existan empleados que hayan sido contratados el 3 de diciembre de 1981.

En general, EXISTS(Q) devuelve CIERTO si hay *por lo menos una tupla* en el resultado de la consulta Q y devuelve FALSO en caso contrario. Por otro lado, NOT EXISTS(Q) devuelve CIERTO si *no hay tuplas en el resultado* de la consulta Q y devuelve FALSO en caso contrario.

Mediante la cláusula NOT EXISTS es posible implementar la operación de DIVISIÓN. Veámoslo con un ejemplo: Supongamos un modelo de datos de la forma:



Supongamos que las tablas a que da lugar tienen el siguiente contenido:

LIBROS

CodL	Título
X	Titulo1
Y	Titulo2
Z	Titulo1

EDICIONES

CodL	CodEdit
X	ED1
X	ED2
Y	ED1
Z	ED1
Z	ED2

EDITORIAL

CodEdit	Editorial
ED1	Editorial1
ED2	Editorial2

si deseamos "***encontrar los libros que han sido publicados por todas las editoriales***"

el resultado debería ser una nueva relación de la forma:

EDICIONES ÷ EDITORIAL

CodL
X
Z

Una sentencia SQL que permite obtener este resultado es la siguiente:

```
SELECT TITULO
FROM LIBROS
WHERE NOT EXISTS (  (SELECT CODEDIT FROM EDITORIAL)
                    MINUS
                    (SELECT CODEDIT FROM EDICIONES
                           WHERE EDICIONES.CODL = LIBROS.CODL))
```

Como resultado de la sentencia se obtienen los *títulos tales que no hay ninguna editorial que no haya realizado una edición de los mismos*.

Si el operador MINUS no está implementado en el gestor, podría utilizarse otra expresión de la forma:

```
SELECT TITULO
FROM LIBROS
WHERE NOT EXISTS (SELECT CODEDIT FROM EDITORIAL
                   WHERE NOT EXISTS ( SELECT CODEDIT FROM EDICIONES
                                       WHERE EDICIONES.CODEDIT=EDITORIAL.CODEDIT
                                          AND EDICIONES.CODL=LIBROS.CODL))
```

1.11 Operadores de conjunto

Combinan los resultados de múltiples consultas en un único resultado. Ambas consultas deben tener el mismo esquema (el mismo número y tipo de datos).

- **UNION**: combina todas las filas distintas de ambas consultas. A diferencia de SELECT, que por omisión conserva los duplicados, esta operación los suprime. Para evitarlo es necesario poner la palabra clave ALL.
- **INTERSECT**: filas comunes a ambas consultas. INTERSECT está incluido en el estándar SQL/92, pero algunos sistemas no lo soportan. En cualquier caso, esta operación se puede rescribir utilizando el operador IN.
- **MINUS**: filas de la primera consulta que no existe en la segunda. MINUS está incluido en el estándar SQL/92, pero algunos sistemas no lo soportan. En cualquier caso, esta operación se puede rescribir utilizando el operador NOT IN.

2 Inserción, modificación y borrado de datos

2.1 Inserción de datos

Para insertar datos en una tabla se utiliza la sentencia INSERT. Hay dos formas de utilizar esta sentencia:

1) Los valores se insertan directamente:

```
INSERT INTO tabla (campo1, ..., campon)
       VALUES (valor1, ..., valorn)
```

Por ejemplo, "***Añadir un nuevo departamento SEGURIDAD sito en Madrid y con código 90***"

```
INSERT INTO DEPT (DEPTNO, DNAME, LOC)
             VALUES (90, 'SEGURIDAD', 'MADRID')
```

Si, como en este caso, se insertan todos los campos, no es necesario indicarlos, y la misma inserción se realizaría con

```
INSERT INTO DEPT
       VALUES (90, 'SEGURIDAD', 'MADRID')
```

Sin embargo, si no se insertan todos, sí es obligatorio indicar los campos. Si los valores son fechas o cadenas de caracteres, deben ir entrecomillados.

La posición de cada columna en la tabla se puede obtener mediante los atributos TABLE_NAME, COLUMN_NAME y COLUMN_ID de la vista USER_TAB_COLUMNS.

2) La segunda forma de insertar datos es hacerlo a través de una consulta:

```
INSERT INTO tabla
       < SENTENCIA SELECT >
```

Por ejemplo, suponiendo que tenemos definida una tabla BUENOSDEPT con la misma estructura que DEPT "***Insertar en BUENOSDEPT los departamentos con más de 3 empleados***"

```
INSERT INTO BUENOSDEPT
       SELECT DEPT.* FROM DEPT, EMP
              WHERE DEPT.DEPTNO=EMP.DEPTNO
                     GROUP BY DEPT.DEPTNO, DNAME, LOC
                     HAVING COUNT (*) >3
```

Ej: Las tuplas que se han utilizado para la explicación de la sentencia SELECT se han introducido en el sistema mediante las siguientes instrucciones:

```
--TUPLAS DE EMPLEADO
INSERT INTO EMP (EMPNO,ENAME, JOB, MGR, HIREDATE, SAL, DEPTNO)
       VALUES (7369,'SMITH','CLERK', 7902,'17/02/1980', 800,20);
INSERT INTO EMP (EMPNO,ENAME, JOB, MGR, HIREDATE, SAL, COMM, DEPTNO)
       VALUES (7499, 'ALLEN', 'SALESMAN',7698,'20/02/1981',1600,300,30);
INSERT INTO EMP (EMPNO,ENAME, JOB, MGR, HIREDATE, SAL, COMM, DEPTNO)
       VALUES (7521,'WARD','SALESMAN',7698,'22/02/1981',1250,500,30);
```

```
INSERT INTO EMP (EMPNO,ENAME, JOB, MGR, HIREDATE, SAL, DEPTNO)
       VALUES (7566,'JONES','MANAGER',7839,'02/04/1981',2975,20);
INSERT INTO EMP (EMPNO,ENAME, JOB, MGR, HIREDATE, SAL, COMM, DEPTNO)
       VALUES (7654,'MARTIN','SALESMAN',7698,'28/09/1981',1250,1400,30);
INSERT INTO EMP (EMPNO,ENAME, JOB, MGR, HIREDATE, SAL, DEPTNO)
       VALUES (7698,'BLAKE','MANAGER',7839,'01/05/1981',2850,30);
INSERT INTO EMP (EMPNO,ENAME, JOB, MGR, HIREDATE, SAL, DEPTNO)
       VALUES (7782,'CLARK','MANAGER',7839,'09/06/1981',2450,10);
INSERT INTO EMP (EMPNO,ENAME, JOB, MGR, HIREDATE, SAL, DEPTNO)
       VALUES (7788,'SCOTT','ANALYST',7566,'09/12/1982',3000,20);
INSERT INTO EMP (EMPNO,ENAME, JOB, HIREDATE, SAL, DEPTNO)
       VALUES (7839,'KING','PRESIDENT','17/11/1981',5000,10);
INSERT INTO EMP (EMPNO,ENAME, JOB, MGR, HIREDATE, SAL, COMM, DEPTNO)
       VALUES (7844,'TURNER','SALESMAN',7698, '08/09/1981',1500,0, 30);
INSERT INTO EMP (EMPNO,ENAME, JOB, MGR, HIREDATE, SAL, DEPTNO)
       VALUES (7876,'ADAMS','CLERK',7788,'12/01/1983',1100, 20);
INSERT INTO EMP (EMPNO,ENAME, JOB, MGR, HIREDATE, SAL, DEPTNO)
       VALUES (7900,'JAMES','CLERK',7698,'03/12/1981',950, 30);
INSERT INTO EMP (EMPNO,ENAME, JOB, MGR, HIREDATE, SAL, DEPTNO)
       VALUES (7902,'FORD','ANALYST',7566,'03/12/1981',3000,20);
INSERT INTO EMP (EMPNO,ENAME, JOB, MGR, HIREDATE, SAL, DEPTNO)
       VALUES (7934,'MILLER','CLERK',7782,'23/01/1982',1300,10);
```

```
-- TUPLAS DE DEPARTAMENTO
INSERT INTO DEPT VALUES     (10,'ACCOUNTING',     'NEW YORK');
INSERT INTO DEPT VALUES     (20,'RESEARCH',       'DALLAS');
INSERT INTO DEPT VALUES     (30,'SALES',          'CHICAGO');
INSERT INTO DEPT VALUES     (40,'OPERATIONS',     'BOSTON');
```

En este caso:

- la sentencia SELECT **no** puede tener ORDER BY
- las columnas no definidas se rellenan con nulos.

2.2 Modificación de datos

Para modificar los datos se utiliza la sentencia UPDATE. Hay varias formas de utilizarla:

```
      1) UPDATE tabla
            SET atributo1=valor1, ...
            [WHERE <condicion>]
      2) UPDATE tabla
            SET (<conjunto atributos>) = (<sentencia SELECT>)
            [WHERE <condicion>]
```

1) En la primera forma se indica individualmente el valor para cada campo, por ejemplo

"Cambiar el empleado 7499 al departamento 30 y modificar su salario a 6000"

```
      UPDATE EMP
            SET DEPTNO=30, SAL=6000
            WHERE EMPNO=7499
```

En todos los casos la cláusula WHERE es opcional, y su formato es el mismo que el de la sentencia SELECT. Si no se indica, todas las filas de la tabla serán modificadas.

2) Por último, en caso de que la cláusula SET contenga una subconsulta, debe devolver exactamente una fila para cada fila a actualizar. Si la consulta no devuelve filas, el valor se rellena con nulos.

Las subconsultas pueden referirse a la tabla que se está actualizando. En este caso, ORACLE evalúa la subconsulta una vez para cada fila actualizada (**actualización correlacionada**).

2.3 Borrado de datos

Para borrar tuplas de una tabla se utiliza la sentencia DELETE, cuyo formato general es

```
DELETE FROM tabla
       [WHERE <condicion>]
```

Como en el caso de UPDATE, la cláusula WHERE es una condición cuyo formato es el mismo que el de una sentencia SELECT, y si no se especifica ninguna se borrarán todas las tuplas de la tabla. Por ejemplo, "***borrar el departamento RESEARCH"***:

```
DELETE FROM DEPT
       WHERE DNAME='RESEARCH'
```

La condición puede incluir una consulta correlacionada.

EJERCICIOS

1. Para los empleados que tienen como director a algún otro empleado con número mayor que el suyo, obtener los que reciben el salario más de 1000 y menos de 2000, o están en el departamento 30.

```
SELECT * FROM EMP
WHERE ((SAL BETWEEN 1000 AND 2000) OR (DEPTNO = 30))
       AND (MGR > EMPNO))

empno ename      job          mgr hiredate       sal      comm deptno
  7499 ALLEN      SALESMAN    7698 20/02/1981   1600       300     30
  7521 WARD       SALESMAN    7698 22/02/1981   1250       500     30
  7654 MARTIN     SALESMAN    7698 28/09/1981   1250      1400     30
  7698 BLAKE      MANAGER     7839 01/05/1981   2850               30
```

2. Obtener el último empleado por orden alfabético.

```
SELECT MAX(ENAME) FROM EMP

(max)
WARD
```

3. ¿Quiénes reciben el salario más alto y más bajo, y cuáles son esos salarios? (RESOLVER DE 2 MODOS DIFERENTES)

```
SELECT ENAME,SAL    FROM EMP
       WHERE SAL=(SELECT MAX(SAL) FROM EMP)
          OR SAL=(SELECT MIN(SAL) FROM EMP)
ó
SELECT ENAME,SAL    FROM EMP
       WHERE SAL IN (SELECT MAX(SAL) FROM EMP
                      UNION
                       SELECT MIN(SAL) FROM EMP)

ename          sal
SMITH         800
KING         5000
```

4. Hallar la media de los salarios de los departamentos cuyo salario mínimo supera a 900, considerando los salarios inferiores a 5000. Además, dar el código y nombre de los departamentos.

```
SELECT D.DEPTNO, DNAME, AVG(SAL)
       FROM EMP E, DEPT D
       WHERE E.DEPTNO=D.DEPTNO     AND SAL<5000
       GROUP BY D.DEPTNO, DNAME
       HAVING MIN(SAL)>900

deptno dname                     (avg)
    30 SALES          1566,66666666667
    10 ACCOUNTING     1875
```

5. Halla los empleados cuyo salario total (salario + comisión) supera o coincide con la media del salario total (salario + comisión) de la empresa.

```
SELECT ENAME, (SAL + NVL(comm, 0)) salario    FROM EMP
       WHERE SAL >=(SELECT AVG(SAL + NVL(comm, 0)) FROM EMP)

 ename       salario
JONES        2975
BLAKE        2850
CLARK        2450
SCOTT        3000
KING         5000
FORD         3000
```

6. Obtén los empleados cuyo salario total (salario + comisión) supera al de sus compañeros de departamento.

```
SELECT empno, ename, sal, comm, (sal + NVL(comm, 0)) salario FROM EMP E
       WHERE SAL=(SELECT MAX(SAL + NVL(comm, 0)) FROM EMP F
                     WHERE F.DEPTNO=E.DEPTNO)

 empno ename      sal      comm         salario
  7698 BLAKE      2850                    2850
  7788 SCOTT      3000                    3000
  7839 KING       5000                   5000
  7902 FORD       3000                    3000
```

7. ¿Cuántos empleos diferentes, empleados y diferentes salarios encontramos en el departamento 20, y a qué cantidad asciende la suma de los salarios de dicho departamento?

```
SELECT COUNT(DISTINCT JOB) AS EMPLEOS, COUNT(EMPNO) AS EMPLEADOS,
        COUNT (DISTINCT SAL) AS SALARIOS, SUM(SAL) AS SUMA_SAL
       FROM EMP     WHERE DEPTNO=20

empleos      empleados    salarios     suma_sal
3            5            4            10875
```

8. Halla los departamentos que tienen más de tres empleados, e indica el número de empleados.

```
SELECT COUNT(*), DEPTNO   FROM EMP
       GROUP BY DEPTNO
       HAVING COUNT(*)>3

      (count(*)) deptno
               5     20
               6     30
```

9. Halla los empleados que tienen por lo menos un empleado a su mando, ordenados inversamente por nombre.

```
SELECT ENAME FROM EMP
       WHERE EMPNO IN(SELECT DISTINCT MGR FROM EMP)
ORDER BY ENAME DESC

ename
SCOTT
KING
JONES
FORD
CLARK
BLAKE
```

10. Obtén información sobre los empleados que tienen el mismo trabajo que los empleados que trabajen en Chicago.

```
SELECT * FROM EMP
       WHERE JOB IN (SELECT JOB FROM EMP E, DEPT D
                     WHERE E.DEPTNO=D.DEPTNO AND LOC ='CHICAGO')

 empno ename      job          mgr hiredate      sal    comm deptno
  7369 SMITH      CLERK       7902 17/12/1980    800            20
  7499 ALLEN      SALESMAN    7698 20/02/1981   1600    300     30
  7521 WARD       SALESMAN    7698 22/02/1981   1250    500     30
  7566 JONES      MANAGER     7839 02/04/1981   2975            20
  7654 MARTIN     SALESMAN    7698 28/09/1981   1250   1400     30
  7698 BLAKE      MANAGER     7839 01/05/1981   2850            30
  7782 CLARK      MANAGER     7839 09/06/1981   2450            10
  7844 TURNER     SALESMAN    7698 08/09/1981   1500      0     30
  7876 ADAMS      CLERK       7788 12/01/1983   1100            20
  7900 JAMES      CLERK       7698 03/12/1981    950            30
  7934 MILLER     CLERK       7782 23/01/1982   1300            10
```

11. Halla los nombres de los empleados que no son jefes. (RESOLVER DE 2 MODOS DIFERENTES)

```
SELECT ENAME FROM EMP E
 WHERE EMPNO NOT IN (SELECT DISTINCT MGR FROM EMP P
                     WHERE E.EMPNO = P.MGR)
ó
SELECT ENAME FROM EMP E
 WHERE NOT EXISTS (SELECT * FROM EMP P
                     WHERE E.EMPNO = P.MGR)
ó
SELECT ENAME FROM EMP
 WHERE EMPNO NOT IN (SELECT DISTINCT MGR FROM EMP
                     WHERE MGR IS NOT NULL)
ename
ALLEN
WARD
MARTIN
TURNER
ADAMS
JAMES
MILLER
SMITH
```

12. Calcula cuántos empleos hay en cada departamento y cual es la media anual del salario de cada departamento. Indica el nombre del departamento.

```
SELECT E.DEPTNO, DNAME, COUNT(DISTINCT JOB) AS EMPLEOS,
                  AVG(SAL)*12 AS SALARIO_ANUAL
       FROM EMP E, DEPT D
       WHERE E.DEPTNO=D.DEPTNO
       GROUP BY E.DEPTNO, DNAME

deptno dname            EMPLEOS    salario_anual
30     SALES               3           18800
20     RESEARCH            3           26100
10     ACCOUNTING          3           35000
```

13. Lista los empleados del departamento 30 y sus comisiones por orden descendente de comisión. En caso de que la comisión sea nula, escribir el texto “SIN COMISIÓN”

```
SELECT ENAME, NVL(TO_CHAR(comm), ‘SIN COMISION’) comision
       FROM EMP
       WHERE DEPTNO = 30
       ORDER BY COMISION DESC

ename      comision
WARD       500
ALLEN      300
MARTIN     1400
TURNER     0
BLAKE      SIN COMISION
JAMES      SIN COMISION
```

14. Halla el código y nombre de cada supervisor junto con el número de empleados a los que supervisa. Puede haber empleados sin supervisor; en este caso, se indicará solamente la cantidad de empleados y los demás campos (nombre y código del supervisor) quedarán con nulos.

```
SELECT J.ENAME AS JEFE, J.EMPNO AS CODIGOJEFE, COUNT(*) AS NUM_EMPLE
       FROM EMP E, EMP J
       WHERE E.MGR = J.EMPNO (+)
       GROUP BY J.ENAME, J.EMPNO

jefe       codigojefe         num_emple
CLARK            7782                1
SCOTT            7788                1
FORD             7902                1
                                     1
BLAKE            7698                5
KING             7839                3
JONES            7566                2
```

15. Hallar los empleados cuyo sueldo es el mayor de su departamento, indicando además su salario y el nombre del departamento.

```
SELECT ENAME, SAL, DNAME
       FROM EMP E, DEPT D
       WHERE E.DEPTNO=D.DEPTNO
       AND SAL=(SELECT MAX(SAL) FROM EMP H WHERE H.DEPTNO=E.DEPTNO)

ename         sal   dname
BLAKE        2850   SALES
FORD         3000   RESEARCH
SCOTT        3000   RESEARCH
KING         5000   ACCOUNTING
```

16. Hallar los números de los departamentos cuya suma de salarios sea la más alta, indicando dicha suma. (RESOLVER DE 2 MODOS DIFERENTES)

```
SELECT DEPTNO, SUM(SAL)   FROM EMP
       GROUP BY DEPTNO
       HAVING SUM(SAL)>=ALL(SELECT  SUM(SAL)  FROM EMP
                            GROUP BY DEPTNO)
ó
SELECT DEPTNO, SUM(SAL)   FROM EMP
       GROUP BY DEPTNO
       HAVING SUM(SAL)= (SELECT  MAX(SUM(SAL))  FROM EMP
                            GROUP BY DEPTNO)
deptno          (sum)
    20         10875
```

17. Listar los empleados que corresponden a los 4 mayores salarios.

```
SELECT * FROM EMP E
       WHERE 4> (SELECT COUNT(DISTINCT SAL) FROM EMP H
                     WHERE H.SAL > E.SAL)

 empno ename      job          mgr hiredate      sal      comm deptno
  7566 JONES      MANAGER     7839 02/04/1981   2975                20
  7698 BLAKE      MANAGER     7839 01/05/81   2850                  30
  7788 SCOTT      ANALYST     7566 09/12/1982   3000                20
  7839 KING       PRESIDENT        17/11/1981   5000                10
  7902 FORD       ANALYST     7566 03/12/1981   3000                20
```

18. Para departamentos con al menos 2 empleados y tal que la media del salario del departamento sea mayor que la media de salarios de la empresa, indíquese el código y nombre del departamento y la suma de salarios de sus empleados.

```
SELECT D.DEPTNO, DNAME, SUM(SAL)
       FROM EMP E, DEPT D
       WHERE E.DEPTNO=D.DEPTNO
       GROUP BY D.DEPTNO, DNAME
       HAVING AVG(SAL) > (SELECT AVG(SAL) FROM EMP)
       AND COUNT(*) >=2
deptno dname                (sum)
    20 RESEARCH              10875
    10 ACCOUNTING             8750
```

19. Listar las localidades que son sede de departamentos con empleados, y en las que trabajan al menos cuatro empleados, indicando el número de éstos.

```
SELECT E.DEPTNO, DNAME,LOC, COUNT(*)
       FROM EMP E, DEPT D
       WHERE E.DEPTNO=D.DEPTNO
       GROUP BY E.DEPTNO,DNAME,LOC
       HAVING COUNT(*)>=4

deptno dname          loc                (count(*))
    20 RESEARCH       DALLAS                      5
    30 SALES          CHICAGO                     6
```

20. Promocionar a Ward a Manager del departamento 20, e incrementar su salario en 100

```
UPDATE EMP
       SET JOB = 'MANAGER', SAL = SAL + 100, DEPTNO = 20
WHERE ename = 'WARD'

empno        ename        job         sal     comm   deptno
7521         WARD       MANAGER       1350     500       20
```

21. Cambiar los empleados que trabajan en Dallas o Detroit al departamento de Chicago. Su salario será ahora 1.5 veces el salario medio del nuevo departamento, y su comisión será 1.5 veces la comisión media del nuevo departamento.

```
UPDATE EMP SET DEPTNO = (SELECT DEPTNO        FROM DEPT
                           WHERE LOC = 'CHICAGO'),
        (SAL, COMM) =  (SELECT 1.5 * AVG(SAL), 1.5 * AVG(COMM)   FROM EMP
                           WHERE DEPTNO IN (SELECT DEPTNO FROM DEPT
                                            WHERE LOC = 'CHICAGO'))
WHERE DEPTNO IN (SELECT DEPTNO FROM DEPT
                    WHERE LOC = 'DALLAS' OR LOC = 'DETROIT')

 empno ename       job         sal     comm   deptno
  7369 SMITH      CLERK       2350      825       30
  7566 JONES      MANAGER     2350      825       30
  7788 SCOTT      ANALYST     2350      825       30
  7876 ADAMS      CLERK       2350      825       30
  7902 FORD       ANALYST     2350      825       30
```

SQL como DDL (Lenguaje de Definición de Datos)

Hasta ahora hemos utilizado el lenguaje SQL como un **DML** (Data Manipulation Language) o Lenguaje de Manipulación de Datos, es decir, para añadir, borrar, modificar o consultar datos. Pero además el SQL puede utilizarse como **DDL** (Data Definición Language) o Lenguaje de Definición de Datos; esto es, podemos utilizar SQL para crear, modificar o borrar diferentes objetos de bases de datos como tablas, vistas o índices.

3 Definición de Tablas

3.1 Creación de tablas

Una tabla es un objeto de la BD que almacena datos. El diccionario de datos mantiene información sobre cada tabla. Oracle utiliza su diccionario de datos para asegurar que se introduzca el tipo correcto de datos (numérico, carácter, fecha) en los diferentes atributos que la componen.

La sintaxis básica de creación de una tabla es la siguiente:

```
CREATE TABLE [esquema.]tabla
   (  atrib1 tipo1 [DEFAULT definicion] [NOT NULL],
    [,atrib2 tipo2 [DEFAULT definicion] [NOT NULL]]...,
    [CONSTRAINT nombre_restr UNIQUE|PRIMARY KEY (atrib1[,atrib2]...)],
    [CONSTRAINT nombre_restr FOREIGN KEY (atrib1 [,atrib2]...)
             REFERENCES [esquema.]tabla(atrib1[,atrib2]...)
             [ON DELETE SET NULL|CASCADE]],
    [CONSTRAINT nombre_restr CHECK condicion]
    [AS consulta])
```

TABLA: es un nombre de menos de 30 caracteres (letras, números, -). No es sensible a mayúsculas y minúsculas. Por lo general, el esquema SQL en el que se declaran las relaciones se especifica implícitamente por el entorno en que se ejecutan las sentencias CREATE TABLE. Como alternativa, podemos adjuntar explícitamente el nombre del esquema al nombre de la relación, separándolos por un punto:

esquema.tabla

La creación de una tabla engloba las definiciones de atributos y/o restricciones:

a) La DEFINICIÓN DE ATRIBUTOS se realiza dando el nombre del atributo (se ajusta a las mismas reglas que los nombres de tablas), su tipo y opcionalmente indicando que no acepta valores nulos.

Los principales tipos de Oracle son:

Numéricos:

- números enteros: INT[EGER] ([+|-] precis [K|M]) ó SMALLINT
- números reales: FLOAT (precis), REAL, DOUBLE PRECISION
- números con formato: DECIMAL (precis[,decim]), NUMERIC (precis[,decim]), donde *precis* es el número total de dígitos decimales y *decim* es la escala, es decir, el número de dígitos después del punto decimal.

Caracteres:

- VARCHAR2(n): admite letras, números o caracteres especiales. Se almacena en un formato de longitud variable. Su longitud máxima es de 4000 caracteres.
- CHAR(n): admite letras, números o caracteres especiales. Internamente, se almacena en formato de longitud fija. Su longitud máxima es de 2000 caracteres.

Moneda:

- MONEY(numero,decimales)

Fechas:

- DATE: campo de longitud fija de 7 bytes, que se utiliza para almacenar todas las fechas. La hora se almacena como parte de la fecha. Es importante tener esto en cuenta cuando se desea comparar dos fechas. El formato predeterminado de fecha es DD-MON-YY, donde DD es el día, MON es el mes e YY son los dos últimos dígitos del año.

Campos largos:

- RAW: campo de longitud variable para caracteres no interpretables por Oracle (cadenas alfanuméricas o binarias, secuencias de caracteres gráficos, etc.)
- LONG [RAW]: campo de longitud variable para datos de tipo RAW, de una longitud máxima de 2 Gb.
- BLOB: objeto binario de gran tamaño, de hasta 4 Gb de longitud.
- CLOB: objeto de caracteres de gran tamaño, de hasta 4 Gb de longitud.
- NCLOB: tipo de datos CLOB para conjuntos de caracteres multibyte, con una longitud máxima de 4 Gb.
- BFILE: archivo binario externo. El SO determina el tamaño máximo.

Oracle permite la **definición de tipos de datos** por parte del usuario:

```
CREATE TYPE tipo1 AS OBJECT (    nombre varchar2(40),
                                 domicilio varchar2(50));

CREATE TABLE EMPLEADO     (      empno  VARCHAR2(10),
                                 identificación      tipo1);
```

b) La DEFINICIÓN DE RESTRICCIONES DE INTEGRIDAD / SEMÁNTICAS: Se almacenan en el Diccionario de Datos. Permiten al diseñador restringir el rango de valores de una tabla. Las restricciones pueden ser de **columna** si afectan a una sola columna, o de **tabla** si afectan a una o más columnas. Pueden ser:

NOT NULL: la columna NO puede contener un valor nulo

[CONSTRAINT nombre_restr UNIQUE (col1 [, col2] ...): la(s) columna(s) NO pueden contener valores duplicados. Deben declararse como NOT NULL y NO pueden formar parte de la clave primaria. Con [CONSTRAINT nombre_restr] se puede dar nombre a la restricción. En este caso, el nombre de la restricción debe ser único dentro del esquema.

[CONSTRAINT nombre_restr PRIMARY KEY (atrib1 [, atrib2] ...): la(s) columna(s) forman la clave primaria. Por lo tanto, tienen valor UNICO y NO NULO.

[CONSTRAINT nombre_restr FOREIGN KEY (atrib1 [, atrib2] ...) REFERENCES tabla (atrib1 [, atrib2] ...): la(s) columna(s) forman una clave foránea. REFERENCES indica la tabla referenciada. Si los atributos clave de la tabla referenciada no tienen el mismo nombre que los atributos que forman la clave foránea, deben especificarse los nombres de los atributos clave.

Cada valor no nulo en esta(s) columna(s) deberá(n) tener un valor equivalente en una columna de la tabla referenciada. Es posible que una tabla haga referencia a sí misma.

[ON DELETE SET NULL|CASCADE]: si se borra una tupla en la tabla "principal"

- SET NULL: se coloca el valor nulo …
- CASCADE: se borran las tuplas …

en las tuplas de la tabla actual donde el elemento borrado es foráneo

[CONSTRAINT nombre_restr CHECK (condición)]: antes de que una fila sea insertada o borrada debe satisfacer *condición*.

```
CHECK (EMPNO BETWEEN 10 and 100)
CHECK (LOC IN ('ATENAS', 'LONDON', 'MADRID'))
CHECK (LOC <> 'ATENAS' OR DEPTNO = 20)
```

Los predicados que admite son: de comparación, BETWEEN, LIKE, IN, IS NULL /IS NOT NULL y ALL/ANY.

Por ejemplo, la creación de tablas de empleado y departamento podría ser la siguiente:

```
CREATE TABLE DEPT(
      DEPTNO                        NUMBER(4),
      DNAME                         VARCHAR2(14),
      LOC                           VARCHAR2(13),
          PRIMARY KEY (DEPTNO));
```

ó

```
CREATE TABLE DEPT(
      DEPTNO                        NUMBER(4) CONSTRAINT rest1 PRIMARY KEY,
      DNAME                         VARCHAR2(14),
      LOC                           VARCHAR2(13));
```

```
CREATE TABLE EMP (
      EMPNO                      NUMBER(4) NOT NULL ,
      ENAME                      VARCHAR2(10),
      JOB                        VARCHAR2(9),
      MGR                        NUMBER(4),
      HIREDATE                   DATE,
      SAL                        DECIMAL(7,2),
      COMM                       DECIMAL(7,2),
      DEPTNO                     NUMBER(4),
             PRIMARY KEY (EMPNO),
             FOREIGN KEY (DEPTNO) REFERENCES DEPT);
```

Como se ve en este ejemplo, DEPTNO es una clave foránea que referencia a DEPT, pero como el atributo clave de DEPT también se llama DEPTNO no hay que indicarlo explícitamente. La definición de la clave foránea podría haber sido:

```
DEPTNO CONSTRAINT rest2 REFERENCES dept(deptno)
```

En este caso no se indica el tipo de *deptno*; se le asigna el mismo que el campo donde es clave primaria.

[AS consulta] : Permite especificar una consulta para determinar los contenidos de la tabla. Las filas devueltas por la subconsulta se insertan en la tabla tras su creación.

3.2 Modificación de tablas

La definición de una tabla se puede modificar con la instrucción ALTER. Las acciones de alterar incluyen la adición / eliminación / modificación de una columna (atributo), y la adición / eliminación de restricciones de la tabla

```
ALTER TABLE [esquema.]tabla
      [ADD (       atrib1 tipo1 [DEFAULT definicion] [NOT NULL],
                   [,atrib2 tipo2 [DEFAULT definicion] [NOT NULL]]...)]
      [MODIFY (    atrib1 tipo1 [DEFAULT definicion] [NOT NULL],
                   [,atrib2 tipo2 [DEFAULT definicion] [NOT NULL]]...)]
      [ADD CONSTRAINT restricción de clave 1ª/única/foránea]
      [DROP PRIMARY KEY [CASCADE KEEP|DROP INDEX]]
      [DROP UNIQUE (col1 [,col2]...)]
      [DROP CONSTRAINT restricción [CASCADE]]
      [DROP COLUMN columna [CASCADE CONSTRAINTS]]
```

Si se añade un atributo, su valor es nulo para las filas existentes. Solo se pueden añadir columnas NOT NULL si no existen tuplas en el momento de la modificación.

Sólo se puede cambiar el tipo o disminuir el tamaño de una columna si tiene valores nulos en todas las columnas.

[DROP PRIMARY KEY [CASCADE KEEP|DROP INDEX]]: se elimina la restricción de clave primaria. Con CASCADE KEEP INDEX se mantiene el índice asociado (creado automáticamente por Oracle) mientras que con CASCADE DROP INDEX se elimina.

[DROP UNIQUE (col1 [,col2]...)]: se elimina la restricción de unicidad sobre las columnas indicadas.

[DROP CONSTRAINT restricción [CASCADE]]: se elimina una restricción y las restricciones donde la columna afectada es foránea. Si se omite CASCADE Oracle no la elimina si afecta a una clave primaria o columna única y existe alguna clave foránea que la referencie.

[DROP COLUMN columna [CASCADE CONSTRAINTS]]: se elimina una columna y todas las restricciones donde se encuentra. Si se omite CASCADE CONSTRAINTS Oracle no borra la tupla con clave primaria o columna única si cualquier clave foránea la referencia.

3.3 Borrado de tablas

Para borrar una tabla se utiliza la sentencia `DROP TABLE`

```
DROP TABLE [esquema.]tabla [CASCADE CONSTRAINTS]
```

[CASCADE CONSTRAINTS]: se elimina la tabla y todas las restricciones donde se encuentra alguna de sus columnas primarias y únicas. Si se omite esta cláusula la tabla no se borra si existen restricciones sobre sus columnas, y Oracle devuelve un error.

4 Definición de Índices

4.1 Creación de índices

Un índice sobre un atributo *A* de una relación es una estructura de datos que permite encontrar rápidamente las tuplas que poseen un valor fijo en ese atributo. Los índices generalmente facilitan las consultas en las que el atributo *A* se compara con una constante (*A=3*) o incluso (*A<3*).

Un índice adecuadamente situado en una tabla ayudará a la BD a recuperar más rápidamente los datos, sobre todo cuando las tablas son muy grandes. Los índices se asocian a una única

tabla. Podrá haber un índice primario (índice por el cual se ordena la tabla de datos) y múltiples índices secundarios.

Aunque la creación de índices no forma parte de los estándares de SQL2, en general los sistemas comerciales lo permiten. Para la creación de índices se utiliza `CREATE INDEX`.

```
CREATE [UNIQUE] INDEX [esquema.]indice
      ON tabla (atrib1 [ASC|DESC] [,atrib2 [ASC|DESC]]...)
      [NOSORT]
```

UNIQUE: indica que se trata de un índice único, es decir, no admite más de una tupla con el mismo valor en los atributos que forman el índice. Cuando se crea una tabla con clave primaria, automáticamente Oracle crea un índice único.

ON tabla (atrib1 [ASC|DESC][,atrib2 [ASC|DESC]]...): Es posible crear índices compuestos, que se forman con más de una columna. Se emplea en caso de columnas que siempre se consultan juntas.

NOSORT: por defecto, si el índice es primario Oracle ordena las filas de la tabla de datos de forma ascendente en base al atributo indexado. Si las tuplas ya están ordenadas en base al índice, se puede omitir la ordenación de Oracle con NOSORT. Esta cláusula sólo tiene sentido, por tanto, cuando se trata de un índice primario.

Ejemplo:

```
CREATE INDEX nomdept ON dept (dname)
```

Oracle determina qué índices se utilizarán en función de lo que se especifique en las cláusulas **WHERE** y **ORDER BY** de la consulta.

La selección de índices requiere que el diseñador de la BD haga un compromiso:

- la existencia de un índice en un atributo agiliza enormemente las consultas en que se especifica un valor de él.
- en cambio, todos los índices construidos para un atributo de alguna tabla hacen que inserciones, modificaciones y eliminaciones sean más complejas y lentas

A la hora de decidirse a crear los índices se debe estimar cuál será la combinación normal de consultas y otras operaciones sobre la BD. Si una tabla se consulta con mucha mayor frecuencia de lo que se modifica, conviene utilizar índices sobre los atributos utilizados como filtro de consulta más a menudo. Si las modificaciones son la operación predominante, hay que ser más cauto respecto a la creación de índices.

4.2 Borrado de índices

Para borrar un índice se utiliza `DROP INDEX`:

```
DROP INDEX [esquema.]indice
```

5 Sinónimos

Permiten crear nombres alternativos de vistas y tablas para omitir el creador de la tabla, la ubicación, o para darle otro nombre.

```
CREATE [PUBLIC] SYNONYM [esquema.]sinonimo
      FOR [esquema.]tabla
```

Por definición, el acceso al sinónimo es sólo para el creador del sinónimo. Con PUBLIC puede hacerse accesible a todos los usuarios.

6 Formato de la salida

A continuación se presentan una serie de comandos propios de la herramienta SQL WorkSheet y parámetros Oracle que permiten mejorar el aspecto y formato de los resultados de las consultas realizadas.

6.1 Formatear columnas

cambiar carácter de subrayado en la cabecera de columnas: `SET UNDERLINE =`

nombre de columna: `COLUMN atributo HEADING 'texto cabecera'`

visualizar en 2 líneas: `COLUMN atributo HEADING 'texto|cabecera'`

modificar tamaño de columna: `COLUMN atributo FORMAT A7`

¿qué pasa con los caracteres que no caben?

continúan en la línea siguiente: `COLUMN atributo FORMAT a7 WRAP`

desplaza la última palabra que no cabe: `COLUMN atributo FORMAT a7 WORD_WRAP`

los ignora: `COLUMN atributo FORMAT a7 TRUNCATE`

colocar máscara en columna:
```
COLUMN sal FORMAT 90,999
COLUMN mgr FORMAT 9,999
```

Tipos de máscaras numéricas:

9	99999	determina el ancho de presentación en función del número de dígitos especificado. Si el número se desborda, se muestra ######. No se visualizan los ceros a la izquierda.
0	09999	Presenta los ceros a la izquierda
$	$9999	Coloca el signo $ delante del número
B	B9999	Presenta un 0 como un blanco
,	99,999	Coloca una coma en la posición especificada
.	999.99	Coloca un punto decimal en la posición especificada y redondea el valor

alineación de la cabecera de columna:

derecha: `COLUMN atributo JUSTIFY RIGHT`

izquierda: `COLUMN atributo JUSTIFY LEFT`

centrada: `COLUMN atributo JUSTIFY CENTER`

```
SET UNDERLINE       =
COLUMN ename        HEADING 'Nombre | Empleado'
COLUMN loc          FORMAT A7 WORD_WRAP
COLUMN sal          FORMAT 99,999
COLUMN mgr          FORMAT 9,999

select * from emp;
select * from dept;
```

```
             Nombre
  EMPNO      Empleado   JOB          MGR    HIREDATE     SAL       COMM   DEPTNO
=======      ======     ========     ======= ========== ====      ======  =====
      7369   SMITH      CLERK        7,902  17/02/80        800           20
      7499   ALLEN      SALESMAN     7,698  20/02/81      1,600   300     30
      7521   WARD       SALESMAN     7,698  22/02/81      1,250   500     30
      7566   JONES      MANAGER      7,839  02/04/81      2,975           20
      7654   MARTIN     SALESMAN     7,698  28/09/81      1,250  1400     30
      7698   BLAKE      MANAGER      7,839  01/05/81      2,850           30
      7782   CLARK      MANAGER      7,839  09/06/81      2,450           10
      7788   SCOTT      ANALYST      7,566  09/12/82      3,000           20
      7839   KING       PRESIDENT           17/11/81      5,000           10
      7844   TURNER     SALESMAN     7,698  08/09/81      1,500    0      30
      7876   ADAMS      CLERK        7,788  12/01/83      1,100           20
      7900   JAMES      CLERK        7,698  03/12/81        950           30
      7902   FORD       ANALYST      7,566  03/12/81      3,000           20
      7934   MILLER     CLERK        7,782  23/01/82      1,300           10
14 filas seleccionadas.

DEPTNO       DNAME          LOC
=======      ======         ========
    10       ACCOUNTING     NEW
                            YORK
    20       RESEARCH       DALLAS
    30       SALES          CHICAGO
    40       OPERATIONS     BOSTON
4 filas seleccionadas.
```

"Reseteando el formato de la columna SAL"

```
COLUMN sal CLEAR
select * from emp;
```

```
             Nombre
  EMPNO      Empleado   JOB          MGR    HIREDATE       SAL    COMM   DEPTNO
=======      ======     ========     ======= ==========    =====  =====  =====
      7369   SMITH      CLERK        7,902  17/02/80         800          20
      7499   ALLEN      SALESMAN     7,698  20/02/81        1600    300   30
      7521   WARD       SALESMAN     7,698  22/02/81        1250    500   30
      7566   JONES      MANAGER      7,839  02/04/81        2975          20
      7654   MARTIN     SALESMAN     7,698  28/09/81        1250   1400   30
      7698   BLAKE      MANAGER      7,839  01/05/81        2850          30
      7782   CLARK      MANAGER      7,839  09/06/81        2450          10
      7788   SCOTT      ANALYST      7,566  09/12/82        3000          20
      7839   KING       PRESIDENT           17/11/81        5000          10
      7844   TURNER     SALESMAN     7,698  08/09/81        1500     0    30
      7876   ADAMS      CLERK        7,788  12/01/83        1100          20
      7900   JAMES      CLERK        7,698  03/12/81         950          30
      7902   FORD       ANALYST      7,566  03/12/81        3000          20
      7934   MILLER     CLERK        7,782  23/01/82        1300          10

14 filas seleccionadas.
```

"Ver los atributos de todas las columnas"

```
COLUMN
```

```
COLUMN   loc ON FORMAT   a7 word_wrap

COLUMN   ename ON HEADING  'Nombre|Empleado' headsep '|'

COLUMN   mgr ON FORMAT   9,999

COLUMN   sal ON FORMAT   99,999
```

"Borrar las definiciones de todas las columnas"

```
CLEAR COLUMNS
```

6.2 La instrucción Decode

Mediante la función **decode** es posible implementar de forma sencilla la estructura *if-then*.

```
DECODE (atributo, compara1, accion1, compara2, accion2, ..., acciónN)
```

La instrucción compara el contenido del atributo con el texto *compara1*. Si son iguales, entonces se lleva a cabo la acción. Si no son iguales, salta a la siguiente comparación. Si no se cumple ninguna de las comparaciones, entonces se realiza la acción indicada en *else*.

"Nombre de empleados y puestos que ocupan (en castellano)"

```
SELECT ename, DECODE (job,'CLERK','Oficinista','SALESMAN','Vendedor',
    'MANAGER','Gerente','ANALYST','Analista','PRESIDENT','Presidente',
        'Sin Puesto')
FROM emp
ORDER BY ename;
```

```
ENAME      JOB
---------- ---------
SMITH      Oficinista
ALLEN      Vendedor
WARD       Vendedor
JONES      Gerente
MARTIN     Vendedor
BLAKE      Gerente
CLARK      Gerente
SCOTT      Analista
KING       Presidente
TURNER     Vendedor
ADAMS      Oficinista
JAMES      Oficinista
FORD       Analista
MILLER     Oficinista
```

6.3 Variables de Usuario

definir variables: `DEFINE variable = valor`

confirmar definición de variable 'variable': `DEFINE variable`

```
DEFINE mi_atributo = JOB
DEFINE mi_tabla = EMP

select &mi_atributo, sal
from &mi_tabla
order by &mi_atributo;
```

```
antiguo   1: select &mi_atributo, sal
nuevo   1: select JOB, sal
antiguo   2: from &mi_tabla
nuevo   2: from EMP
antiguo   3: order by &mi_atributo
nuevo   3: order by JOB

JOB              SAL
========= ==========
ANALYST        3000
ANALYST        3000
CLERK           800
CLERK          1100
CLERK          1300
CLERK           950
MANAGER        2975
MANAGER        2450
MANAGER        2850
```

```
PRESIDENT      5000
SALESMAN       1600
SALESMAN       1250
SALESMAN       1500
SALESMAN       1250
14 filas seleccionadas.
```

"Conocer los valores de todas las variables"

```
DEFINE
```

```
DEFINE _CONNECT_IDENTIFIER = "bdalumno" (CHAR)
DEFINE _SQLPLUS_RELEASE = "902000100" (CHAR)
DEFINE _EDITOR         = "Notepad" (CHAR)
DEFINE _O_VERSION      = "Oracle9i Enterprise Edition Release 9.2.0.1.0 - Production
With the Partitioning, OLAP and Oracle Data Mining options
JServer Release 9.2.0.1.0 - Production" (CHAR)
DEFINE _O_RELEASE      = "902000100" (CHAR)

DEFINE MI_ATRIBUTO     = "JOB" (CHAR)
DEFINE MI_TABLA        = "EMP" (CHAR)
```

borrar variable: `UNDEFINE variable`

sustitución de variables: `&variable`

6.4 Ruptura de secuencia

La sentencia **break** permite manejar las rupturas de secuencia:

al cambiar un atributo: `BREAK ON columna_rupt1 [ON columna_rupt2]`

al final de un informe: `REPORT`

después de cada línea: `ROW`

Acciones cuando cambia el grupo:

saltar *n* líneas o a la sgte. página: `SKIP n | PAGE`

sin duplicados: `NODUP`

con duplicados: `DUP`

Para implementar la lógica de ruptura de secuencia en SQL*Plus se debe ordenar la consulta según la misma columna(s) en la que se especifica la ruptura de secuencia.

"Empleados cuyo salario no supere las 2500. Mostrar únicamente su nombre, salario y nº dpto al que pertenece. Ordenar la salida por nº dpto, dejando una línea en blanco entre departamentos y no repetir el nº"

```
BREAK ON deptno SKIP 1

select deptno, ename, sal
from emp
where sal < 2500
order by deptno;
```

```
    DEPTNO     ENAME             SAL
----------     ---------- ----------
        10     CLARK            2450
               MILLER           1300

        20     SMITH             800
               ADAMS            1100

        30     ALLEN            1600
               JAMES             950
               TURNER           1500
               WARD             1250
               MARTIN           1250
9 filas seleccionadas.
```

"Ver los BREAKS actuales"

```
BREAK
```

```
break en deptno ignorar 1 nodup
```

"Borrar las definiciones de BREAK"

```
CLEAR BREAKS
```

6.5 Cálculos sobre líneas

Es posible realizar cálculos con los valores de columna en la ruptura de secuencia. Basta con utilizar la sentencia:

```
COMPUTE funcion OF col1 [col2] ON columna_ruptura | REPORT
COMPUTE funcion1 [funcion2] OF atributo ON columna_ruptura
```

funcion: admite los valores SUM, MIN, MAX, AVG, STD (desviación estándar), VAR (varianza), COUNT (valores no nulos), NUM (nº filas)

REPORT: se realiza un cálculo total del informe. En este caso, para que el resultado de la operación se visualiza al final del informe es necesario utilizar `BREAK ON REPORT`.

"Borrar las definiciones de COMPUTE"

```
CLEAR COMPUTE
```

"Suma de salarios por departamento"

```
CLEAR BREAKS
CLEAR COMPUTE
BREAK ON deptno SKIP 1 nodup ON job nodup
COMPUTE SUM OF SAL on deptno

select deptno, job, ename, sal
from emp
order by deptno, job;
```

```
    DEPTNO JOB       ENAME             SAL
********** ********* ********** **********
        10 CLERK     MILLER           1300
           MANAGER   CLARK            2450
           PRESIDENT KING             5000
********** *********             ----------
sum                                   8750

        20 ANALYST   SCOTT            3000
                     FORD             3000
           CLERK     SMITH             800
                     ADAMS            1100
           MANAGER   JONES            2975
********** *********             ----------
sum                                  10875
```

```
        30 CLERK     JAMES             950
           MANAGER   BLAKE            2850
           SALESMAN  ALLEN            1600
                     MARTIN           1250
                     TURNER           1500
                     WARD             1250
********** ********* ----------
sum                                   9400

14 filas seleccionadas
```

"Suma de salarios por departamento y por puesto de trabajo dentro de cada uno"

```
CLEAR BREAKS
CLEAR COMPUTE

BREAK ON deptno SKIP 1 nodup ON job nodup

COMPUTE SUM OF SAL on deptno job

select deptno, job, ename, sal
from emp
order by deptno, job;
```

```
    DEPTNO JOB       ENAME             SAL
********** ********* ********** **********
        10 CLERK     MILLER            1300
           *********             ----------
           sum                         1300
           MANAGER   CLARK             2450
           *********             ----------
           sum                         2450
           PRESIDENT KING              5000
           *********             ----------
           sum                         5000
**********                       ----------
sum                                    8750

        20 ANALYST   SCOTT             3000
                     FORD              3000
           *********             ----------
           sum                         6000
           CLERK     SMITH              800
                     ADAMS             1100
           *********             ----------
           sum                         1900
           MANAGER   JONES             2975
           *********             ----------
           sum                         2975
**********                       ----------
sum                                   10875

        30 CLERK     JAMES              950
           *********             ----------
           sum                          950
           MANAGER   BLAKE             2850
           *********             ----------
           sum                         2850
           SALESMAN  ALLEN             1600
                     MARTIN            1250
                     TURNER            1500
                     WARD              1250
           *********             ----------
           sum                         5600
**********                       ----------
sum                                    9400


14 filas seleccionadas
```

6.6 Títulos y pies de página

Definición de títulos: `TTITLE|BTITLE posicion texto formato posicion texto formato`

valores de posición:

sangrado hasta la posición *n* de la línea actual: `COL n`

salto hasta el principio de la nueva línea *n*: `SKIP n`

alineación: `LEFT, RIGHT, CENTER`

presentar los datos en negrita: `BOLD`

nº línea/nº página actual: `SQL.LNO, SQL.PNO`

lista la definición de títulos existentes: `TTITLE`

eliminación de las definiciones de títulos y pies de página: `TTITLE OFF`

`BTITLE OFF`

```
CLEAR BREAKS
CLEAR COLUMNS
CLEAR COMPUTE
TTITLE OFF
BTITLE OFF
TTITLE CENTER '========================' SKIP 1           -
CENTER 'TITULO DEL INFORME' SKIP 1                        -
CENTER '========================' SKIP 1                  -
LEFT 'INFORME DE PERSONAL' RIGHT 'PAG:' FORMAT 099 SQL.PNO SKIP 2
BTITLE SKIP 2 CENTER ' FIN DEL INFORME '

SELECT * FROM EMP;
```

```
                                    ========================
                                       TITULO DEL INFORME
                                    ========================
INFORME DE PERSONAL                      PAG: 001

EMPNO ENAME      JOB              MGR HIREDATE        SAL       COMM    DEPTNO
***** ********** *********   ***** ******** ********** ********** **********
7369 SMITH      CLERK            7902 17/02/80        800                    20
7499 ALLEN      SALESMAN         7698 20/02/81       1600        300         30
7521 WARD       SALESMAN         7698 22/02/81       1250        500         30
7566 JONES      MANAGER          7839 02/04/81       2975                    20
7654 MARTIN     SALESMAN         7698 28/09/81       1250       1400         30
7698 BLAKE      MANAGER          7839 01/05/81       2850                    30
7782 CLARK      MANAGER          7839 09/06/81       2450                    10
7788 SCOTT      ANALYST          7566 09/12/82       3000                    20
7839 KING       PRESIDENT             17/11/81       5000                    10
7844 TURNER     SALESMAN         7698 08/09/81       1500          0         30
7876 ADAMS      CLERK            7788 12/01/83       1100                    20
7900 JAMES      CLERK            7698 03/12/81        950                    30
7902 FORD       ANALYST          7566 03/12/81       3000                    20
7934 MILLER     CLERK            7782 23/01/82       1300                    10

                                         FIN DEL INFORME
```

visualizar valores de atributos en títulos: Para referirse a un valor de atributo en el título hay que almacenar el valor en una variable previamente:

```
COLUMN atributo       NEW_VALUE | OLD_VALUE  variable
```

Con OLD_VALUE se toma el valor anterior antes de la ruptura. Es útil si se desea poner el valor de un atributo al final de una página.

El atributo a visualizar deberá estar en ORDER BY de la cláusula SELECT y en BREAK

"Relación de códigos de supervisor y empleados que supervisa"

```
CLEAR BREAKS
CLEAR COLUMN
CLEAR COMPUTE
TTITLE OFF
BTITLE OFF


COLUMN mgr NEW_VALUE var_mgr NOPRINT
TTITLE LEFT 'Supervisor: ' var_mgr SKIP 2
BREAK ON mgr SKIP PAGE

select mgr, ename, sal, deptno
from emp
where mgr is not null
order by mgr;
```

En este ejemplo cabe destacar ciertas particularidades. Por un lado, se utiliza el parámetro NOPRINT en la sentencia COLUMN para evitar que la columna (en este caso, *mgr*) no se imprima en la línea de detalle. Por otro lado, es necesario el parámetro SKIP PAGE en BREAK ON para que se modifique el título al cambiar de página.

El resultado queda de la forma:

```
Supervisor:       7566

ENAME            SAL          DEPTNO
----------     ----------     ----------
SCOTT             3000         20
FORD              3000         20

Supervisor:       7698

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

Supervisor:       7782

ENAME             SAL     DEPTNO
---------- ---------- ----------
MILLER           1300         10

Supervisor:       7788

ENAME             SAL     DEPTNO
---------- ---------- ----------
ADAMS            1100         20

Supervisor:       7839

ENAME             SAL     DEPTNO
---------- ---------- ----------
JONES            2975         20
CLARK            2450         10
BLAKE            2850         30

Supervisor:       7902

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

6.7 Establecimiento de parámetros

Establecer parámetros para la sesión actual:	`SET parámetro valor`
Los parámetros más destacables son:	
cambiar subrayado cabecera atributo:	`SET UNDERLINE =`
líneas entre el comienzo de página y el título:	`SET NEWPAGE [1|n]`
líneas por página desde el título:	`SET PAGESIZE [14|n]`
si n = 0 no se realizan saltos de página	
caracteres por línea:	`SET LINESIZE [80|n]`
filas recuperadas a la vez:	`SET ARRAY [20|n]`
COMMIT implícito tras cada sentencia SQL o bloque PL/SQL:	`SET AUTOCOMMIT[off|on]`
longitud para visualización de valores LONG:	`SET LONG [80|n]`
texto para representar los valores nulos:	`SET NULL texto`
carácter subrayado cabecera de columnas:	`SET UNDERLINE [cr|on|off]`
ver los valores actuales de un parámetro:	`SHOW parámetro`
visualizar todos los parámetros:	`SHOW PARAMETERS`

PRÁCTICA 1 (Utilización de sentencias DML y DDL)

APARTADO A): Utilizando las tablas descritas al principio de este documento, resolver las consultas números 1 al 34

APARTADO B):

1. Modelizar mediante un E/R un supuesto práctico que dé lugar a 4 entidades relacionadas
2. Transformar el modelo conceptual anterior a relacional
3. Mediante la herramienta SQL WorkSheet de Oracle:
   a. Crear las tablas relacionales derivadas de la transformación anterior
   b. Modificar el esquema de alguna de las tablas creadas
   c. Plantear 3 consultas que involucren más de una tabla
   d. Crear índices que agilicen las consultas anteriores
   e. Crear un sinónimo para alguna de las tablas creadas
   f. Rellenar las tablas con tuplas que permitan demostrar el correcto funcionamiento de las consultas. Ejecutar las consultas y almacenar sus resultados. Utilizar en las consultas el sinónimo creado en el punto anterior
   g. Borrar el contenido de las tablas
   h. Borrar las tablas
   i. Borrar los índices

APARTADO C): Resolver las siguientes preguntas:

1. Ordenar los empleados por su departamento y luego por orden descendente de su número. Separación por número de departamento, dejando una línea en blanco. Poner la fecha en formato día, mes (3 letras) y año completo.
   - Si no tiene comisión o no tiene supervisor, que ponga “NO” en el campo correspondiente
   - El salario que aparezca con la “,” de miles.
   - El nombre del empleo con la primera letra en mayúscula (solamente)
   - Cada tupla debe ocupar una única línea

2. Para los empleados que tienen como director a algún otro con número mayor que el suyo, obtener los que reciben el salario más de 1000 y menos de 2000, o están en el departamento 30. Utilizar variables tanto para los salarios como para el número de departamento.

3. Obtén información sobre los empleados que tienen el mismo trabajo que los empleados que trabajen en Chicago.
   - Poner en la cabecera el empleo.
   - Poner un título y un pie de informe.
   - Totalizar por empleo el salario total (salario + comisiones)
   - Mostrar únicamente los tres primeros caracteres del nombre de empleado

Condiciones generales de la práctica:

- Las prácticas son individuales.
- Para los tres apartados anteriores, deberán presentarse en papel todas las sentencias SQL junto con el resultado obtenido (en el caso del APARTADO A) las respuestas ya están incluidas en este documento), **escritas a mano**. Asimismo, para el APARTADO B) deberá incluirse el modelo E/R utilizado.

APARTADO C)

```
1.     clear columns
       clear breaks
       SET NULL '    NO'
       COLUMN sal FORMAT 99,999
       break on deptno skip 1
       select deptno, empno, ename, initcap(job), mgr,
                           to_char(hiredate,'DD MON YYYY'), sal, comm
       from emp
       order by deptno, empno desc;

DEPTNO  EMPNO ENAME  INITCAP(J   MGR    TO_CHAR(HIR     SAL       COMM
-----  ------ -----  ---------    -----  -----------  ------- ----------
10     7934   MILLER Clerk        7782   23 ENE 1982         1,300     NO
       7839   KING   President    NO     17 NOV 1981         5,000     NO
       7782   CLARK  Manager      7839   09 JUN 1981         2,450     NO

20     7902   FORD   Analyst      7566   03 DIC 1981         3,000     NO
       7876   ADAMS  Clerk        7788   12 ENE 1983         1,100     NO
       7788   SCOTT  Analyst      7566   09 DIC 1982         3,000     NO
       7566   JONES  Manager      7839   02 ABR 1981         2,975     NO
       7369   SMITH  Clerk        7902   17 FEB 1980           800     NO

30     7900   JAMES  Clerk        7698   03 DIC 1981           950     NO
       7844   TURNER Salesman     7698   08 SEP 1981         1,500        0
       7698   BLAKE  Manager      7839   01 MAY 1981         2,850     NO
       7654   MARTIN Salesman     7698   28 SEP 1981         1,250     1400
       7521   WARD   Salesman     7698   22 FEB 1981         1,250      500
       7499   ALLEN  Salesman     7698   20 FEB 1981         1,600      300
14 filas seleccionadas.


2.     DEFINE salario = SAL
       DEFINE departamento = DEPTNO
       select empno, ename, job, mgr, hiredate, &salario, comm, &departamento
       from emp
       where ((&salario between 1000 and 2000) or (&departamento = 30))
       and (mgr > empno);

antiguo  1:  select  empno,  ename,  job,  mgr,  hiredate,  &salario,  comm,
&departamento
nuevo   1: select empno, ename, job, mgr, hiredate, SAL, comm, DEPTNO
antiguo   3: where ((&salario between 1000 and 2000) or (&departamento = 30))
nuevo   3: where ((SAL between 1000 and 2000) or (DEPTNO = 30))

EMPNO ENAME   JOB          MGR          HIREDATE     SAL       COMM     DEPTNO
----- ------ -------      --------     -------      -----  ---------- -------
7499 ALLEN   SALESMAN        7698      20/02/81   1,600        300         30
7521 WARD    SALESMAN        7698      22/02/81   1,250        500
7654 MARTIN  SALESMAN        7698      28/09/81   1,250       1400
7698 BLAKE   MANAGER         7839      01/05/81   2,850         NO
       4 filas seleccionadas.

3.     SET PAGESIZE 20
       SET LINESIZE 80
       CLEAR COLUMNS
       CLEAR BREAKS
       TTITLE OFF
       BTITLE OFF

       COLUMN ename FORMAT A3 TRUNCATE
       COLUMN job NEW_VALUE var_job NOPRINT

       TTITLE LEFT 'PUESTO TRABAJO: ' var_job SKIP 2
       BTITLE CENTER 'FIN PUESTO' SKIP 2

       BREAK ON job SKIP PAGE
       COMPUTE SUM OF suma ON job
```

```
select empno, ename, job, deptno, sal, comm, sal + nvl(comm, 0) suma
from emp
where job in (select distinct (job)
              from emp b
              where b.deptno in (select deptno
                                 from dept where loc='CHICAGO'))
order by job, deptno;


columns limpiado
breaks limpiado
PUESTO TRABAJO: CLERK

 EMPNO       ENA     DEPTNO            SAL          COMM       SUMA
----------   ---    -------            -------      ------ ----------
      7934   MIL          10            1300        NO           1300
      7369   SMI          20             800        NO            800
      7876   ADA          20            1100        NO           1100
      7900   JAM          30             950        NO            950
                                     ---------- ---------- ----------
                                      4150        NO           4150

                                  FIN PUESTO

PUESTO TRABAJO: MANAGER


EMPNO        ENA     DEPTNO            SAL          COMM       SUMA
----------    ---       ---------- ---------- ---------- ----------
      7782   CLA          10            2450     NO           2450
      7566   JON          20            2975     NO           2975
      7698   BLA          30            2850     NO           2850
                                     ---------- ---------- ----------
                                     8275     NO           8275

                                  FIN PUESTO

PUESTO TRABAJO: SALESMAN

EMPNO        ENA     DEPTNO            SAL       COMM       SUMA
----------   ---    ----------     ---------- ---------- ----------
      7499   ALL          30           1600        300       1900
      7654   MAR          30           1250       1400       2650
      7844   TUR          30           1500          0       1500
      7521   WAR          30           1250        500       1750
                                  ---------- ---------- ----------
                                     5600       2200       7800

                                  FIN PUESTO
```