INDICE.

1.Introduccion a sql.

El lenguaje SQL permite la comunicación con el sistema gestor de base de datos. Fue desarrollado sobre un prototipo de gestor de b.d.d relacionales denominados system r
Diseñado por IBM desarrollado en los años 70, alrededor del 1979 oracle corp. Presento la primera implementación comercial de SQL.
Entre las principales características de SQL se puede destacar que es un lenguaje para todo tipo de usuarios: admin. , desarrolladores y usuarios finales(normales). El usuario que emplea sql especifica que quiere, no donde ni como, y le permite hacer cualquier consulta de datos.
El lenguaje sql nos va a permitir consultar datos a la base de datos, crear , actualizar y eliminar datos de los objetos de la base de datos, controlar a la base de datos y a los objetos, dependiendo de las tareas podemos clasificar las sentencias salen 2 grandes apartados:

-**Sentencias DDL**(data description language): se trata del lenguaje con el que se crea y mantiene la estructura de la base de datos. Y sirve para realizar las siguientes tareas:

1-crear un objeto de la base de datos: tablas, vistas, procedimientos, etc..(orden **create**)
2-eliminar un objeto de la base de datos(orden **drop**)
3-modificar un objeto e la base de datos(orden **alter**)
4-conceder privilegios sobre un objeto de la base de datos(orden **grant**)
5-retirar privilegios sobre un objeto de la base de datos(orden **revoque**)

-**Sentencias DML**(data manipulation language): este lenguaje esta formado por un conjunto de sentencias que sirven para manipular los datos contenidos en la base de datos y nos permite realizar la siguientes operaciones:

1-Insert: insertar filas de datos en una tabla.
2-Update: actualizar filas de datos de una tabla.
3-Delete: eliminar filas de datos de una tabla.
4-Select: recuperar filas de datos de una tabla.

2-Tipos de datos.

Oracle soporta los siguientes tipos de datos:

-**CHAR**: este tipo de datos permite almacenar caracteres de longitud fija, entre 1 y 255(depende de las versiones). La longitud de la cadena se coloca entre paréntesis. Tiene las siguientes características:
-las columnas tienen longitud fija.
-si se introduce una cadena de menor longitud, queda definida, se rellenara con blancos a la derecha, hasta que quede completa.
-si se introduce una cadena mayor a la longitud definida, oracle devolverá un error.

-**VARCHAR2(n):** almacena cadenas de caracteres de longitud variable. La longitud máxima que se puede definir, es de 2000 caracteres(según versiones).
La longitud de la cadena se define entre paréntesis. Tiene las siguientes características:
-las columnas tienen una longitud variable.
-si se introduce una cadena de menor longitud que la definida, se almacenará con es longitud y no se rellenara con blancos ni con otro carácter cualquiera a la derecha hasta completar la longitud definida.
-si se introduce una cadena de mayor longitud que la definida, oracle devolverá un error.

-**NUMBER(p, e):** este tipo almacena datos numéricos tanto enteros como decimales. Soporta 38 dígitos de precisión(p). para especificar columnas numéricas se utiliza el identificador NUMBER. Por ejemplo: SAL _ MEDIO number(9,2), de este modo definimos una columna SAL _ MEDIO, como numérica de 9 dígitos, de los cuales 2 son decimales. Se pueden especifica números enteros usando como formato NUMBER(precisión). Por ejemplo, SAL _ MEDIO number(15).
-**LONG:** almacena caracteres de longitud variable que contengan hasta 2GB de información .
-**RAW:** es igual que el tipo de datos varchar2 pero en binario; su longitud maxima dependiendo de versiones, es de 255 caracteres.
-**LONGRAW:** es igual que el tipo longpero en binario y soporta datos multimedia.
-**TIPODATE:** se usa para almacenar información de fechas. Para cada tipo date se almacenan: siglo/año/mes/dia/hora/minutos/segundos --
el formato de la fecha se puede cambiar mediante un ALTERSESION.

3. INTRODUCCIÓN A SQL PLUS.

3.1 COMANDOS SQL PLUS DE EDICIÓN

Al trabajar con SQL PLUS se dispone de un buffer de edición que contiene la última sentencia SQL que se intentó ejecutar, mientras una sentencia está en el buffer se puede modificar por un conjunto de comandos de edición. SQL > ED, este comando invocará al editor del sistema (notepad.exe) que abrirá un fichero asociado al buffer de edición.

- El comando L [IST], lista una o varias líneas del buffer. Si se utiliza sin argumentos lista todas las líneas.
Como argumentos de LIST, pueden usarse:
L(n) -> lista la línea indicada en n, se pone como línea activa.
L 4 7 -> lista las líneas desde la 4 a la 7.
L LAST -> lista la ultima línea del buffer.
L* -> lista la línea actual (activa).
L -> visualiza el contenido del buffer.
La última línea listada es la línea activa. SQL PLUS indica que es la línea activa con un asterisco detrás del número de línea. La línea activa es sobre la que se pueden realizar operaciones de edición.

- El comando DEL, elimina la línea activa.

```
SQL> l2
  2* dir, apellidocomision
SQL> del
```

- I [NPUT] texto, añade una línea detrás de la línea activa.

```
SQL> l1
  1* select oficio, salario
SQL> i dir, apellido
```

- C [HANGE] /texto1/texto2, nos permite modificar una línea activa, cambia texto1 por texto2. Si se omite el texto2, elimina texto1.

```
4* where dept_no=10
SQL> c /10/30
  4* where dept_no=30
```

- A [PPEND] texto, añade algo al final de la línea activa.

```
SQL> l2
  2* dir, apellido
SQL> a, comision
  2* dir, apellido, comision
```

R[UN]→nos enseña el contenido del buffer y lo ejecuta

3.2 FICHEROS DE COMANDOS

Desde SQL Plus es posible salvar uno o más comandos en un fichero denominado fichero de comandos. Una vez creado el fichero de comandos se puede cargar, editar o ejecutar.
El contenido del buffer SQL lo podemos guardar en un fichero con el comando SAVE.

Sino damos extensión se guarda como .sql

- SAVE Fichero REPLACE – permite reemplazar el fichero existente.
- SAVE Fichero APPEND – se añade el contenido del buffer al final del fichero.

Con el comando EDIT invocamos a un editor externo con el que podemos crear y editar ficheros de comandos:

- EDIT [Nombre fichero]

Si utilizamos el comando EDIT sin poner el nombre del fichero se edita el buffer SQL. EDIT coloca el buffer editado en un fichero de nombre AFIEDT.BUF en el directorio de trabajo.

Se puede cargar un fichero en el buffer a través del comando GET.

- GET Nombre fichero.

Si queremos que además de cargarlo en el buffer se ejecute, se utilizará el comando START Nombre fichero.

3.3 CONSULTA DE DATOS.

Para recuperar información o lo que es lo mismo para realizar cualquier consulta utilizaremos la sentencia select. El usuario emplea esta sentencia con el nivel de complejidad apropiado para él: especifica lo que quiere sin indicar ni donde ni como y su formato es:

SELECT [ALL | DISTINCT]
[Expresa columna1, expresa columna2, expresa columna3| *]

FROM [nombre de tabla1, nombre de tabla2…]
[WHERE condicion]
[ORDER BY expr_columna {DESC / ASC }...]

FROM: especifica la tabla o la lista de tablas de las que se recuperan los datos,

WHERE: obtiene las filas que cumplen la condición expresada. La complejidad de la condición es prácticamente ilimitada. El formato de la condición es: expresión, operador, expresión.
Las expresiones pueden ser una constante, una expresión aritmética, un valor nulo o un nombre de columna.
Se pueden construir condiciones múltiples usando los operadores lógicos (and , or y not).
Se pueden emplear paréntesis para forzar el orden de evaluación.

ORDER by: especifica el criterio de clasificación del resultado de la consulta. ASC indica ordenación ascendente y des descendente.

ALL: recupera todas las filas aunque algunas estén repetidas. Es la opción por comisión.

DISTINCT: solo recupera las filas que son distintas.

Ejercicio:

1.Sacar el oficio de la tabla emple, cuando el departamento sea menor a 20:

```
SQL> select oficio from emple where dept_no<20;

OFICIO
----------
DIRECTOR
PRESIDENTE
EMPLEADO
```

2.Traer a pantalla, el nombre y el sueldo de lso trabajadores, ordenados por departamento.

```
SQL> select dept_no,apellido,salario from emple order by dept_no;

   DEPT_NO APELLIDO      SALARIO
---------- ---------- ----------
        10 CEREZO         318500
        10 REY            650000
        10 MUÑOZ          169000
        20 SÁNCHEZ        104000
        20 ALONSO         143000
        20 FERNÁNDEZ      390000
        20 GIL            390000
        20 JIMÉNEZ        386750
        30 ARROYO         208000
        30 NEGRO          370500
        30 MARTÍN         162500
        30 JIMENO         123500
        30 TOVAR          195000
```

30 SALA 162500

3.Seleccionar las filas de la tabla emple, cuyo trabajo es director y su salario es superior a 350000:

SQL> select * from emple where salario>350000 and oficio='DIRECTOR';

EMP_NO	APELLIDO	OFICIO	DIR	FECHA_AL	SALARIO	COMISION	DEPT_NO
7566	JIMÉNEZ	DIRECTOR	7839	02/04/81	386750		20
7698	NEGRO	DIRECTOR	7839	01/05/81	370500		30

4. Vamos a sacar todo el archivo emple ordenadas ascendentemente por oficio y descendentemente por dir.

SQL> select * from emple order by oficio asc,dir desc;

EMP_NO	APELLIDO	OFICIO	DIR	FECHA_AL	SALARIO	COMISION	DEPT_NO
7788	GIL	ANALISTA	7566	09/11/81	390000		20
7902	FERNÁNDEZ	ANALISTA	7566	03/12/81	390000		20
7566	JIMÉNEZ	DIRECTOR	7839	02/04/81	386750		20
7698	NEGRO	DIRECTOR	7839	01/05/81	370500		30
7782	CEREZO	DIRECTOR	7839	09/06/81	318500		10
7369	SÁNCHEZ	EMPLEADO	7902	17/12/80	104000		20
7876	ALONSO	EMPLEADO	7788	23/09/81	143000		20
7934	MUÑOZ	EMPLEADO	7782	23/01/82	169000		10
7900	JIMENO	EMPLEADO	7698	03/12/81	123500		30
7839	REY	PRESIDENTE		17/11/81	650000		10
7499	ARROYO	VENDEDOR	7698	20/02/80	208000	39000	30
7654	MARTÍN	VENDEDOR	7698	29/09/81	162500	182000	30
7844	TOVAR	VENDEDOR	7698	08/09/81	195000	0	30
7521	SALA	VENDEDOR	7698	22/02/81	162500	65000	30

Para sacar los registros de un campo, sin que esten repetidos hay que usar la sentecia distinc:

SQL> select distinct oficio from emple;

OFICIO

ANALISTA
DIRECTOR
EMPLEADO
PRESIDENTE
VENDEDOR

ALIAS DE COLUMNAS:

- Permite utilizar como nombres de columnas los nombres de los atributos para poder configurar cabeceras con otras cadenas de caracteres.

SQL> select deptno "nciudad" from dept; → Cambia el nombre del campo por el que yo le pongo entre comillas, pero solo para esa sentencia.

SQL> select empno, ename, sal, sal+(sal*3/100) "SALNUEVO" from emp; → Le pone un nuevo nombre a la columna creada.

3.4. OPERADORES ARITMÉTICOS.

Seleccionar dir,apellido y el salario y crear una nueva columna que sea el salario mas el 3% del salario actual.

```
SQL> select dir,apellido,salario, salario+(salario*3/100) "nueva"
  2  from emple;

SQL> select dir,apellido,salario, salario+(salario*3/100) "nueva" from emple;

       DIR APELLIDO      SALARIO     nueva
---------- ----------    ----------  ----------
      7902 SÁNCHEZ        104000     107120
      7698 ARROYO         208000     214240
      7698 SALA           162500     167375
      7839 JIMÉNEZ        386750     398352,5
      7698 MARTÍN         162500     167375
      7839 NEGRO          370500     381615
      7839 CEREZO         318500     328055
      7566 GIL            390000     401700
           REY            650000     669500
```

Si hay valores null..en una operación matemática, hay que utilizar la funcion NVL:

```
SQL> select apellido, salario, comision, nvl(comision,0)+salario "total salario"
  2  from emple;


APELLIDO      SALARIO   COMISION  total salario
----------     ----------    ----------  -------------
SÁNCHEZ       104000                    104000
ARROYO        208000        39000       247000
SALA          162500        65000       227500
JIMÉNEZ       386750                    386750
MARTÍN        162500        182000      344500
NEGRO         370500                    370500
CEREZO        318500                    318500
GIL           390000                    390000
REY           650000                    650000
TOVAR         195000        0           195000
```

Al probar nvl con null, nos damos cuenta de que da error debido a que, cuando un registro es 0, tiene el fallo de no dividir ese registro entre todo lo demás, y crea fallo en la solución.

```
SQL> select ename, sal, comm, sal/nvl(comm,3) "total"
  2  from emp
  3  where comm>0 or comm is null;

ENAME          SAL       COMM      total
---------- ---------- ---------- ----------
SMITH             800                266,666667
ALLEN            1600        300    5,33333333
WARD             1250        500     2,5
JONES             2975               991,666667
MARTIN           1250       1400      ,892857143
BLAKE            2850                950
CLARK            2450                816,666667
SCOTT            3000                1000
KING             5000                1666,66667
ADAMS            1100                366,666667
JAMES            950                 316,666667
FORD             3000                1000
MILLER           1300                 433,333333
```

3.5.Operaciones de comparación y lógicas

OPERADORES	OPERACIÓN
=	Igual
>	Mayor
>=	Mayor o igual
<	Menor
=<	Igual o menor
!=	Distinto
<>	Distinto

Ejemplo:

1.Seleccionar la fila de los trabajadores ke tengan saldo igual o menor a 800.

```
SQL> select *
  2  from emp
  3  where sal <= 800;

     EMPNO    ENAME     JOB          MGR    HIREDATE       SAL      COMM    DEPTNO
----------    ----------    ---------      ----------    --------       ----------    ----------    ----------
      7369       SMITH     CLERK         7902      17/12/80       800                      20
```

2.seleccionar todos los empleados cuyo saldo sea distinto de 800:

```
SQL> select *
  2  from emp
  3  where sal != 800;

 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
      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         19/04/87      3000                  20
      7839 KING       PRESIDENT                   17/11/81      5000                  10
      7844 TURNER     SALESMAN       7698        08/09/81      1500       0          30
      7876 ADAMS      CLERK          7788         23/05/87      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
```

3.Seleccionar a los empleados con un salario menor o igual a 1500 y que su oficio sea vendedor:

```
SQL> select *
  2  from emp
  3  where sal<=1500 and job='SALESMAN';

 EMPNO    ENAME     JOB          MGR    HIREDATE       SAL      COMM     DEPTNO
----------      ----------      ---------          ----------    --------       ----------  ----------  ----------
      7521 WARD       SALESMAN       7698      22/02/81       1250            500            30
      7654 MARTIN     SALESMAN       7698     28/09/81       1250           1400            30
      7844 TURNER     SALESMAN       7698     08/09/81       1500           0               30


      7369 SMITH      CLERK          7902 17/12/80       800                20
      7788 SCOTT      ANALYST          7566 19/04/87       3000                 20
      7876 ADAMS      CLERK          7788 23/05/87       1100               20
      7900 JAMES      CLERK          7698 03/12/81       950                30
      7902 FORD       ANALYST          7566 03/12/81       3000                 20
```

3.6 OPERADORES DE COMPARACION DE CADENAS DE CARACTERES.

Para comparar cadenas de caracteres se había utilizado el igual. Pero este operador no sirve si queremos realizar consultas de este tipo:

- Obtener los datos de los empleados cuyo apellido comience por una ‘s’ o bien obtener los nombres de los alumnos que incluyan la palabra Pérez. Para especificar este tipo de consultas, en SQL se utiliza el operador ‘LIKE’ que permite obtener el resultado de la consulta mediante la utilización de los siguientes caracteres especiales:

%: es un comodín que representa cualquier cadena de caracteres de 0 a más.
_: es un marcador de posición y representa un carácter cualquiera.

Ejercicios:

1.Obtener aquellos apellidos que empiecen con m:

```
SQL> select *
  2  from emp
  3  where ENAME LIKE 'M%';

 EMPNO    ENAME      JOB              MGR          HIREDATE       SAL       COMM
DEPTNO
----------    ----------     ---------          ----------         --------          ----------     ----------  -----
-----
      7654    MARTIN     SALESMAN       7698          28/09/81          1250          1400        30
      7934     MILLER    CLERK            7782          23/01/82           1300                    10
```

2.Obtener los apellidos que tengan una L en la segunda posición.

```
SQL> select *
  2  from emp
  3  where ename like '_L%'
  4  ;

    EMPNO ENAME      JOB              MGR HIREDATE       SAL       COMM     DEPTNO
---------- ---------- --------- ---------- -------- ---------- ---------- ----------
      7499 ALLEN      SALESMAN        7698 20/02/81       1600        300         30
      7698 BLAKE      MANAGER         7839 01/05/81       2850                    30
      7782 CLARK      MANAGER         7839 09/06/81       2450                    10
```

3.Averiguar los empleados ke tienen una a en la primera posición y una l en su interior!

```
SQL> select *
  2  from emp
  3  where ename like 'A%L%';

    EMPNO ENAME      JOB              MGR HIREDATE       SAL       COMM     DEPTNO
---------- ---------- --------- ---------- -------- ---------- ---------- ----------
     7499 ALLEN      SALESMAN       7698 20/02/81       1600        300         30
```

3.7.Comprobación con conjunto de valores.

Permite comparar una columna o una expresión con una lista de valores utilizando el "**IN | BETWEEN**"

IN.- Nos permite comprobar si una expresión pertenece o no (NOT) a un conjunto de valores lo que hace posible la realización de comparaciones multiples, [NOT]IN(lista de valores separados por una coma).

WHERE [NOT] IN (lista de valores separados por comas)

Ejercicio:

1.Averiguar los datos de los empleado que pertenece al depto 10 ó 20.

```
SQL> select * from emp
  2  where deptno IN(10,20);

    EMPNO ENAME      JOB              MGR HIREDATE       SAL       COMM     DEPTNO
---------- ---------- --------- ---------- -------- ---------- ---------- ----------
     7369 SMITH      CLERK            7902 17/12/80        800                    20
     7566 JONES      MANAGER          7839 02/04/81       2975                    20
     7782 CLARK      MANAGER          7839 09/06/81       2450                    10
     7788 SCOTT      ANALYST          7566 19/04/87       3000                    20
     7839 KING       PRESIDENT             17/11/81       5000                    10
     7876 ADAMS      CLERK            7788 23/05/87       1100                    20
     7902 FORD       ANALYST          7566 03/12/81       3000                    20
     7934 MILLER     CLERK            7782 23/01/82       1300                    10
```

BETWEEN.- Comprueba si un valor esta comprendido o no (NOT) entre un rango de valores desde un valor inicial a un valor final.
[NOT] BETWEEN valor inicial and valor final.

2.Averiguar los datos de los empleado cuyo salario esta comprendido entre 1000 2000.

SQL> select * from emp

```
  2  where sal between 1000 and 2000;

     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
      7844 TURNER     SALESMAN       7698 08/09/81       1500          0         30
      7876 ADAMS      CLERK          7788 23/05/87       1100                    20
      7934 MILLER     CLERK          7782 23/01/82       1300                    10
```

3.8.SUBCONSULTAS.

En determinada ocasiones para realizar una operaciones de consulta necesitamos conocer los datos que devuelve otra consulta; si queremos obtener los datos de los empleados que tengan el mismo oficio que pepe, para comenzar tendremos que averiguar el oficio de pepe (1ª consulta) una vez conocido ese dato podemos averiguar los empleado que tienen el mismo oficio que pepe. Las subconsulta son aquellas sentencia select que forma parte de una clausula where de una sentencia select anterior. Una subconsulta consistira en incluir una declaracion select como parte de una consulta where y su formato es similar a esto.

SELECT……
FROM……
WHERE COLUMNA OPERADOR_COMPARATIVO(SELECT……
FROM……
WHERE…….);

Ejercicio:

1.Obtener el apellido de los empleados con el mismo oficio que gil.

```
SQL> select apellido from emple
  2  where oficio=(select oficio from emple where apellido='GIL');

APELLIDO
----------
GIL
FERNÁNDEZ
```

3.8.1 Consultas que genera valores simples/lista.

Son aquella que devuelven una fila o un valor simple. En la sub consulta anterior se extraía un valor simple. El signo igual es un valor de comparación simple si la sub consulta obtiene mas de una fila no devolvería un mensaje de error.

1.Se pretende obtener los apellidos de los empleados cuyo oficio coincida con algún oficio del departamento 20.

```
SQL> select apellido from emple
  2  where oficio in(select oficio from emple where dept_no=20);

APELLIDO
----------
GIL
FERNÁNDEZ
JIMÉNEZ
NEGRO
CEREZO
SÁNCHEZ
ALONSO
MUÑOZ
JIMENO
```

Las sub consultas que genera una lista de valores o filas necesita el uso del operador **IN** en la cláusula **WHERE.**

2.Consultar los datos de los empleados que trabajan en Madrid o Barcelona.

```
SQL> select * from emple
  2  where dept_no in(select dept_no from depart where loc in('MADRID','BARCELONA'));

   EMP_NO APELLIDO   OFICIO           DIR FECHA_AL    SALARIO   COMISION
DEPT_NO
---------- ---------- ---------- ---------- -------- ---------- ---------- ----------
     7369 SÁNCHEZ    EMPLEADO         7902 17/12/80     104000                    20
     7876 ALONSO     EMPLEADO         7788 23/09/81     143000                    20
     7902 FERNÁNDEZ  ANALISTA         7566 03/12/81     390000                    20
     7788 GIL        ANALISTA         7566 09/11/81     390000                    20
     7566 JIMÉNEZ    DIRECTOR         7839 02/04/81     386750                    20
     7499 ARROYO     VENDEDOR         7698 20/02/80     208000      39000         30
     7698 NEGRO      DIRECTOR         7839 01/05/81     370500                    30
     7654 MARTÍN     VENDEDOR         7698 29/09/81     162500     182000         30
     7900 JIMENO     EMPLEADO         7698 03/12/81     123500                    30
     7844 TOVAR      VENDEDOR         7698 08/09/81     195000                0   30
     7521 SALA       VENDEDOR         7698 22/02/81     162500      65000         30
```

3.9 Combinación de Tablas.

En algunos momentos nos encontraremos que una consulta necesita columna de varias tablas. En este caso las tablas se expresaran a la derecha de la cláusula from y su sintaxis quedaría de la siguiente forma:

SELECT columna de las tablas citadas en from
FROM tabla1, tabla2,…….
WHERE tabla1.columna1 = tabla2.columna2……...

Reglas que hay que tener en cuenta:

1 Es posible reunir cuantas tablas se deseen.

2 En la cláusula select se puede solicitar columnas de todas las tablas.

3 Si hay columna con el mismo nombre en las distinta tabla de la cláusula from deben

4 identificarse especificando nombre de la tabla punto nombre de columna.

5 Si el nombre de una columna existe solo en una tabla no será necesario identificar nombre de tabla punto.

6 El criterio que se sigue para combinar las tablas se especificara en where. Si se omite esta cláusula (que especifica la condición de la combinación), el resultado será un PRODUCTO CARTESIANO, que recogerá todas las fila de una tabla con cada fila de la otra.

1.A partir de la tabla emple y depart obtener los siguientes datos de los empleados: APELLIDOS, OFICIO, NÚMERO DE EMPEADO DE EMPLEADO, NOMBRE DEL DEPARTAMENTO, LOCALIDAD.

```
SQL> SELECT APELLIDO,EMP_NO,OFICIO,DNOMBRE,LOC FROM EMPLE,DEPART
  2  WHERE EMPLE.DEPT_NO = DEPART.DEPT_NO;

APELLIDO   EMP_NO     OFICIO      DNOMBRE          LOC
---------- ---------- ---------- -------------- --------------
CEREZO       7782     DIRECTOR    CONTABILIDAD     SEVILLA
REY          7839      PRESIDENTE CONTABILIDAD     SEVILLA
MUÑOZ        7934     EMPLEADO    CONTABILIDAD     SEVILLA
SÁNCHEZ      7369     EMPLEADO    INVESTIGACIÓON   MADRID
ALONSO       7876     EMPLEADO    INVESTIGACIÓON   MADRID
FERNÁNDEZ    7902     ANALISTA    INVESTIGACIÓON   MADRID
GIL          7788     ANALISTA    INVESTIGACIÓON   MADRID
JIMÉNEZ      7566      DIRECTOR   INVESTIGACIÓON   MADRID
ARROYO       7499     VENDEDOR    VENTAS           BARCELONA
NEGRO        7698     DIRECTOR    VENTAS           BARCELONA
MARTÍN       7654     VENDEDOR    VENTAS           BARCELONA
JIMENO       7900     EMPLEADO    VENTAS            BARCELONA
TOVAR        7844     VENDEDOR    VENTAS           BARCELONA
SALA         7521     VENDEDOR    VENTAS           BARCELONA

14 filas seleccionadas.

SQL> SELECT APELLIDO,EMP_NO,OFICIO,DNOMBRE,LOC,EMPLE.DEPT_NO FROM
EMPLE,DEPART
  2  WHERE EMPLE.DEPT_NO = DEPART.DEPT_NO;

APELLIDO   EMP_NO     OFICIO      DNOMBRE          LOC            DEPT_NO
---------- ---------- ---------- -------------- -------------- ----------
CEREZO       7782     DIRECTOR    CONTABILIDAD     SEVILLA          10
REY          7839     PRESIDENTE  CONTABILIDAD     SEVILLA          10
MUÑOZ        7934     EMPLEADO    CONTABILIDAD     SEVILLA          10
SÁNCHEZ      7369     EMPLEADO    INVESTIGACIÓON   MADRID           20
ALONSO       7876     EMPLEADO  I NVESTIGACIÓON    MADRID           2
```

2.Realizar una consulta para obtener nombre de alumno, Las asignatura con cada una de sus notas.

```
SQL> select apenom,nombre,nota from alumnos,asignaturas,notas
  2  where alumnos.dni = notas.dni and asignaturas.cod = notas.cod;

APENOM                         NOMBRE                    NOTA
------------------------------ ------------------------  ----------
Alcalde García, Elena          Prog. Leng. Estr.          6
Alcalde García, Elena          Sist. Informáticos         5
Alcalde García, Elena          Análisis                   6
Cerrato Vela, Luis             FOL                        6
Cerrato Vela, Luis             Entornos Gráficos          4
Cerrato Vela, Luis             Aplic. Entornos 4ªGen      5
Cerrato Vela, Luis             RET                        8
Díaz Fernández, María          FOL                        8
Díaz Fernández, María          Entornos Gráficos          8
Díaz Fernández, María          Aplic. Entornos 4ªGen      9
Díaz Fernández, María          RET                        7

11 filas seleccionadas.
```

3.Obtener los nombre de los alumos matriculados en fol .

```
SQL> select apenom,nombre from alumnos,asignaturas,notas
  2  where alumnos.dni = notas.dni and asignaturas.cod = notas.cod and nombre='FOL';

APENOM                         NOMBRE
------------------------------ ------------------------
Cerrato Vela, Luis             FOL
Díaz Fernández, María          FOL
```

4. FUNCIONES.

4.1 Introducción.

Las Funciones se usan dentro de las expresiones y actuan con los valores de las columnas, variable o constantes. Generalmente generan dos tipos de resultados:

a)Unas producen un resultado que es una modificaron de la información original (poner en minusculas algo en mayuscula).
b)El resultados de otras indica la resultado (El numero de caracteres que tiene una columna.

Se utilizan en clausulas **SELECT**, **WHERE** y **ORDER BY.** Es posible el añadamiento de funciones.

4.2 Funciones Aritméticas.

Trabajan con datos de tipo numerico(incluye los digitos del 0 al 9). Los literales numéricos no se encierran entre comillas.
Trabajan con tres clases de numeros:Valores simple Grupo de valores y Lista de valores.

4.2.1 Funciones de Grupo de valores:

FUNCIONES	PROPOSITO
1. AVG (N) **2.** COUNT (*|EXPRESION) **3.** MAX (EXPRESION) **4.** MIN (EXPRESION) **5.** SUM (EXPRESION)	1 Calcula el valor numero n ignorando los valores nulos(saca la media). 2 (dos opciones) Cuenta el numero de veces que la expresión evalua un dato con valor no nulo, la opcion * cuenta todas las filas seleccionada. 3 calcula el maximo valor de la expresion mas alto 4 calcula el minmo valor de la expresión mas bajo 5 Suma los valores de la expresión seleccionados.

1. AVG.

Calcular el salario medio de lo empleados del departamento 20.

```
SQL> select avg(salario) from emple
  2  where dept_no=20;

AVG(SALARIO)
------------
     282750
```

2.COUNT.

Sacar la media de la comisiones cobradas por los empleado del dept_no 20.

```
SQL> select  count(*) from emple;

  COUNT(*)
----------
        14

SQL> select count(oficio) from emple;

COUNT(OFICIO)
-------------
          14
```

3.MAX.

Cual es el salario mas alto.

```
SQL> select max(salario) from emple;

MAX(SALARIO)
------------
      650000
```

Que apellido es el que alfabéticamente tiene el mayor valor de la tabla emple.

```
SQL> select max(apellido) from emple;

MAX(APELLI
----------
TOVAR
```

Obtener el salario y el apellido del empleado con apellido máximo de la tabla emple.

```
SQL> select salario,apellido from emple
  2  where apellido in(select max(apellido) from emple);

   SALARIO APELLIDO
---------- ----------
    195000 TOVAR
```

4.MIN.

Obtener los datos del empleado que tiene el minimo salario.

```
SQL> select * from emple
  2  where salario in(select min(salario) from emple);

    EMP_NO       APELLIDO   OFICIO         DIR   FECHA_ALT     SALARIO   COMISION    DEPT_NO
----------       ----------  ----------  ----------  ----------  ----------  ----------  ----------
      7369           SÁNCHEZ    EMPLEADO    7902    17/12/1980     104000                    20
```

5.SUM.

Averiguar el importe total de los empleados del departamento 30

```
SQL> select sum(salario) from emple where dept_no=30;

SUM (SALARIO)
-------------
 1222000
```

4.2.2. Funciones de valores simples.

Son funciones sencillas que trabajan con valores simple entendiendo por valor simple un numero, variable, una columna de la tabla.
Para probar una de estas funciones se suele utilizar la tabla **DUAL** cuya descripción es la siguiente.

```
SQL> desc dual
 Nombre                                    ¿Nulo?   Tipo
 ----------------------------------------- -------- ------------------------------------
 DUMMY                                              VARCHAR2(1)
SQL> select * from dual;
D
-
X

SQL> select sysdate from dual;

SYSDATE
--------
23/10/06
SQL> show user;
USER es "ALBERTO"
```

Función ABS(N).- Devuelve el valor absoluto de N.

1.Obtener el valor absoluto de -20.

```
SQL> select abs(-20) from dual;
  ABS(-20)
----------
        20
```

2.Obtener el valor absoluto del salario -10000000 y el apellido de los empleados de la tabla emple.

```
SQL>  select abs(salario-10000000)"sal_Absoluto",apellido from emple;

sal_Absoluto APELLIDO
------------ ----------
     9896000 SÁNCHEZ
     9792000 ARROYO
     9837500 SALA
     9613250 JIMÉNEZ
     9837500 MARTÍN
     9629500 NEGRO
     9681500 CEREZO
     9610000 GIL
```

Función CEIL(N).- Obtiene el valor entero inmediatamente superior o igual al "N".

Para numero positivos el valor superior de un numero decimal es el siguiente entero y el valor superior de un valor entero es el mismo.

Ejemplo:

3. CEIL(20.7) 20.2 16

SQL> select ceil(20.7),ceil(20.2),ceil(16) from dual;

```
CEIL(20.7) CEIL(20.2)   CEIL(16)
---------- ---------- ----------
        21         21         16
```

4.Para numero negativos el resultado seria el siguiente.

-20.7 -70.2 -16

-SQL> select ceil(-20.7),ceil(-70.2),ceil(-16) from dual;

```
CEIL(-20.7) CEIL(-70.2)  CEIL(-16)
----------- ----------- ----------
        -20         -70         -16
```

Función FLOOR(N).- Es lo contrario de CEIL(N), devuelve el valor entero inmediatamente inferior o igual a N.

SQL> select floor(20.7),floor(20.2),floor(16) from dual;

```
FLOOR(20.7) FLOOR(20.2)  FLOOR(16)
----------- ----------- ----------
         20          20         16
```

El **Floor** en negativo.

SQL> select floor(-20.7),floor(-70.2),floor(-16) from dual;

```
FLOOR(-20.7) FLOOR(-70.2) FLOOR(-16)
------------ ------------ ----------
         -21          -71         -16
```

Función MOD(M,N).- Devuelve el resto resultante de dividir m entre n.
.

11,4 11,0 10,15 -10,3 10,-3 10.4,4.5

SQL> select mod(11,4),mod(11,0),mod(10,15),mod(-10,3),mod(10,-3),mod(10.4,4.5) from dual;

```
 MOD(11,4)  MOD(11,0) MOD(10,15) MOD(-10,3) MOD(10,-3) MOD(10.4,4.5)
---------- ---------- ---------- ---------- ---------- -------------
         3         11         10         -1          1           1,4
```

Función NVL (Valor, Expresión).-Esta función se utiliza para sustituir un valor nulo por otro valor. Si “valor” es null es sustituido “Expresión” si no lo es la función devuelve “valor”. NVL se puede usar con cualquier tipo de datos numérico, carácter, Alfa numericos, etc.
“valor”, “Expresión” debe ser del mismo tipo, “Admiten valor diferente (en determinadas versiones)”.
Con esta función se evita los valores nulos en expresiones aritméticas, ya que en ellas siempre darían siempre un resultado nulo.

6.Partiendo de la tabla emple obtener el salario la comisión y el salario más la comisión de todos los empleados.

```
SQL> select salario,comision,salario+comision from emple;

   SALARIO   COMISION SALARIO+COMISION
---------- ---------- ----------------
    104000
    208000      39000           247000
    162500      65000           227500
    386750
    162500     182000           344500
    370500
    318500
    390000
    650000
    195000          0           195000
    143000
```

```
SQL> select salario,comision,nvl(salario,comision) from emple;

   SALARIO   COMISION NVL(SALARIO,COMISION)
---------- ---------- ----------------------
    104000                     104000
    208000      39000          208000
    162500      65000          162500
    386750                     386750
    162500     182000          162500
    370500                     370500
    318500                     318500
    390000                     390000
    650000                     650000
    195000          0          195000
    143000                     143000
```

```
SQL> select nvl(nombre,'zzz') from nombres;
NVL(NOMBRE,'ZZZ
---------------
PEDRO
JUAN
MARÍA
CLARA
zzz
zzz
JESÚS
```

Función POWER(m,exponente).- Esta función calcula la potencia de un numero, devuelve el valor de “m”, elevado a un “Exponente”

3,4 3,-4 -3,4 4.5,2.4 4.5,2

```
SQL> select power(3,4),power(3,-4),power(-3,4),power(4.5,2.4),power(4.5,2) from dual;

POWER(3,4) POWER(3,-4) POWER(-3,4) POWER(4.5,2.4) POWER(4.5,2)
---------- ----------- ----------- -------------- ------------
        81  ,012345679          81     36,9581338        20,25
```

Función ROUND(numero[,m]).- Esta función redondea los números con la cantidad indicada de dígitos de precisión. Devuelve el valor de “numero” redondeado “m decimales”. Si “m” es negativo el redondeo de dígitos se lleva a cabo a la izquierda del punto decimal. Si se omite “m” devuelve “numero” con cero decimales y redondeados.

1.5634,1 1.5634 1.2234 1.2234,2 1.2676,3

```
SQL> select round(1.5634,1),round(1.5634),round(1.2234),round(1.2234,2),round(1.2676,3) from
dual;

ROUND(1.5634,1) ROUND(1.5634) ROUND(1.2234) ROUND(1.2234,2) ROUND(1.2676,3)
--------------- ------------- ------------- --------------- ---------------
            1,6             2             1            1,22           1,268
```

Cantidades con valor negativos.

145.5,-1 145.5,-2 145.5,-3 141,-1 145,-1

```
SQL> select round(145.5,-1),round(145.5,-2),round(145.5,-3),round(141,-1),round(145,-1) from
dual;
ROUND(145.5,-1) ROUND(145.5,-2) ROUND(145.5,-3) ROUND(141,-1) ROUND(145,-1)
--------------- --------------- --------------- ------------- -------------
            150             100               0           140           150
```

Funcion SIGN(Valor).- Esta funcion indica el signo “Valor”. Si valor es menor que cero la funcion devuelve “-1”, y si valor es mayor que cero devuelve “0”.

```
SQL> select sign(5) from dual;

  SIGN(5)
----------
         1

SQL> select sign(-5) from dual;

  SIGN(-5)
----------
        -1
```

Funcion SQRT(N).- Devuelve la raiz cuadadra. El valor de n no puede ser negativo.

```
SQL> select sqrt(25),sqrt(25.6) from dual;
  SQRT(25) SQRT(25.6)
---------- ----------
         5 5,05964426
```

Funcion TRUNC(numero[,m]).- Trunca los numero para que tengan una cierta cantidad de digitos de precision. Devuelve “numero” Truncado “m” decimales. “m” pude ser negativo si lo es trunca por la izquierda del punto decimal. Si se omite “m” devuelve entre “numero” con cero decimales.

1.5634,1 1.1684,2 1.662

```
SQL> select trunc(1.5634,1),trunc(1.1684,2),trunc(1.662) from dual;
TRUNC(1.5634,1) TRUNC(1.1684,2) TRUNC(1.662)
--------------- --------------- ------------
            1,5            1,16            1
```

Valores Negativos.

```
            187.98,-1          187.98,-2          187.98,-3

SQL> select trunc(187.98,-1),trunc(187.98,-2),trunc(187.98,-3) from dual;
TRUNC(187.98,-1) TRUNC(187.98,-2) TRUNC(187.98,-3)
---------------- ---------------- ----------------
             180              100                0
```

4.2.3. Funciones de Lista.

Trabajan sobre un grupo de columnas dentro de una misma fila. Comparan los valores de cada una de las columnas en el interior de una fila para obtener el mayor o menor valor de la lista.

Función GREATEST(valor1, valor2, …..).-Obtiene el mayor valor de la lista.
Función LEAST(valor1, valor2, ……).-Obtiene el menor valor de la lista.

1.Obtener por cada alumno la mayor nota y menor nota de cada alumnos.

```
SQL> select nombre_alumno,
  2  greatest(nota1,nota2,nota3)"Mayor_nota"
  3  ,least(nota1,nota2,nota3)"Menor_nota"
  4  from notas_alumnos;

NOMBRE_ALUMNO            Mayor_nota Menor_nota
------------------------ ---------- ----------
Alcalde García, M. Luisa          5          5
Benito Martín, Luis               8          6
Casas Martínez, Manuel            7          5
Corregidor Sánchez, Ana           9          6
Díaz Sánchez, María
```

2.Hallar el mayor nombre alfabetico en Benito, Jorge, Andes e Isabel, y hallar el menor entre benito, Julia, Andres e Isabel.

```
SQL>  select greatest('benito','jorge','andres','isabel'),least('benito','julia','andres','isabel')
  2  from dual;
GREAT LEAST(
-----       ------
jorge       andres
```

3.Obtener el mayor y el menor valor de cada empleado entre salario y comision.

```
SQL> select greatest(salario,comision),least(salario,comision) from emple;

GREATEST(SALARIO,COMISION) LEAST(SALARIO,COMISION)
-------------------------- ----------------------

                208000                39000
                162500                65000

                182000               162500



                195000                    0
```

4.3 FUNCIONES DE CADENAS DE CARACTERES.

Esta funciones trabajan con datos tipo (CHAR) o (VARCHAR2). Estos datos incluyen cualquier dato alfanumerico: letra, numerico y caracteres especiales.
Los literales se deben encerrar entre apostrofe.

4.3.1 Funciones que devuelves valores carácter

Funciones que devuelven valores carácter.
Estas funciones devuelve un carácter o un conjunto de caracteres: una cadena en mayuscula, una cadena en minuscula y parte de una cadena, etc.

Funcion CHR(N).- Devuelve el carater cuyo valor binario es equivalente a ‘Ñ’.

1.Devolver las letra cuyo valor ASCII es 75 y 65.

```
SQL> select CHR(75),CHR(65) from dual;

C C
- -
K A
```

```
SQL> select chr(76),chr(66) from dual;
C C
- -
L B
```

Funcion CONCAT(cad1,cad2).- Es la funcion que concatena cadenas. Devuelve “cad1 concatenada con cad2”. Esta funcion es equivalente al “||”.

2.Obtener el apellido de la siguientes manera:
El apellido es:.. “apellido”.

```
SQL> select concat('El apellido es:..',apellido) from emple;
CONCAT('ELAPELLIDOES:..',AP
---------------------------
El apellido es:..SÁNCHEZ
El apellido es:..ARROYO
El apellido es:..SALA
El apellido es:..JIMÉNEZ
El apellido es:..MARTÍN
El apellido es:..NEGRO
El apellido es:..CEREZO
El apellido es:..GIL
El apellido es:..REY
El apellido es:..TOVAR
El apellido es:..ALONSO
El apellido es:..JIMENO
El apellido es:..FERNÁNDEZ
El apellido es:..MUÑOZ
```

3.Con el operador “||”:

```
SQL> select 'El apellido es:..' || apellido from emple;
'ELAPELLIDOES:..'||APELLIDO
---------------------------
El apellido es:..SÁNCHEZ
El apellido es:..ARROYO
El apellido es:..SALA
El apellido es:..JIMÉNEZ
El apellido es:..MARTÍN
El apellido es:..NEGRO
El apellido es:..CEREZO
El apellido es:..GIL
El apellido es:..REY
El apellido es:..TOVAR
El apellido es:..ALONSO
El apellido es:..JIMENO
El apellido es:..FERNÁNDEZ
El apellido es:..MUÑOZ
14 filas seleccionadas.
```

4.Obtener en una columna el apellido y el oficio de cada uno de los empleados de la tabla emple de la siguiente manera.

Apellido es Oficio

```
SQL> select concat(apellido,' es ') || oficio"Apellido es Oficio" from emple;
Apellido es Oficio
------------------------
SÁNCHEZ es EMPLEADO
ARROYO es VENDEDOR
SALA es VENDEDOR
JIMÉNEZ es DIRECTOR
MARTÍN es VENDEDOR
NEGRO es DIRECTOR
CEREZO es DIRECTOR
GIL es ANALISTA
REY es PRESIDENTE
TOVAR es VENDEDOR
ALONSO es EMPLEADO
JIMENO es EMPLEADO
FERNÁNDEZ es ANALISTA
MUÑOZ es EMPLEADO
14 filas seleccionadas.
```

5.Sacar por pantalla es siguiente texto: “ El empleado Numerio de empleado corresponde a apellido del empleado y cobra salario mas comision al mes”.

```
SQL> select concat(concat('El empleado ',emp_no),' corresponde a') || apellido
  2  || ' y cobra salario mas comision ' || concat((salario + nvl(comision,0)),' al mes.')"trabajado
r" from emple;

trabajador
-----------------------------------------------------------------------------------------------
El empleado 7369 corresponde aSÁNCHEZ y cobra salario mas comision 104000 al mes.
El empleado 7499 corresponde aARROYO y cobra salario mas comision 247000 al mes.
El empleado 7521 corresponde aSALA y cobra salario mas comision 227500 al mes.
El empleado 7566 corresponde aJIMÉNEZ y cobra salario mas comision 386750 al mes.
El empleado 7654 corresponde aMARTÍN y cobra salario mas comision 344500 al mes.
El empleado 7698 corresponde aNEGRO y cobra salario mas comision 370500 al mes.
El empleado 7782 corresponde aCEREZO y cobra salario mas comision 318500 al mes.
El empleado 7788 corresponde aGIL y cobra salario mas comision 390000 al mes.
El empleado 7839 corresponde aREY y cobra salario mas comision 650000 al mes.
El empleado 7844 corresponde aTOVAR y cobra salario mas comision 195000 al mes.
El empleado 7876 corresponde aALONSO y cobra salario mas comision 143000 al mes.
El empleado 7900 corresponde aJIMENO y cobra salario mas comision 123500 al mes.
El empleado 7902 corresponde aFERNÁNDEZ y cobra salario mas comision 390000 al mes.
El empleado 7934 corresponde aMUÑOZ y cobra salario mas comision 169000 al mes.

14 filas seleccionadas.
```

Funcion LOWER(cad).-Devuelve toda la cadena a minuscula.

1.TARDE

SQL> select lower('TARDE') from dual;

LOWER
-----
tarde

2.Obtener el apellido en minuscula de los empleado de una empresa.

SQL> select lower(apellido) from emple;
LOWER(APEL
----
sánchez
arroyo
sala
jiménez
martín
negro
cerezo
gil
rey
tovar
alonso
jimeno
fernández
muñoz

Funcion UPPER(cad).-Devuelve toda la cadena convertida a mayuscula.

3. Tarde

SQL> select upper('tarde') from dual;
UPPER

TARDE

SQL> select lower(nombre_alumno)"minusculas",upper(nombre_alumno)"mayusculas" from notas_alumnos;

minusculas mayusculas
---------- ----------
alcalde garcía, m. luisa ALCALDE GARCÍA, M. LUISA
benito martín, luis BENITO MARTÍN, LUIS
casas martínez, manuel CASAS MARTÍNEZ, MANUEL
corregidor sánchez, ana CORREGIDOR SÁNCHEZ, ANA
díaz sánchez, maría DÍAZ SÁNCHEZ, MARÍA

Funcion INITCAP(cad).- Convierte la primera letra de cada palabra a mayuscula y el resto a minuscula.

4.ADMINISTRACION DE SISTEMAS INFORMATICOS.

```
SQL> select initcap('ADMINISTRACION DE SISTEMAS INFORMATICOS') FROM DUAL;
INITCAP('ADMINISTRACIONDESISTEMASINFORM
---------------------------------------
Administracion De Sistemas Informaticos
```

```
SQL> select initcap(nombre_alumno) from notas_alumnos;

INITCAP(NOMBRE_ALUMNO)
------------------------
Alcalde García, M. Luisa
Benito Martín, Luis
Casas Martínez, Manuel
Corregidor Sánchez, Ana
Díaz Sánchez, María
```

Funcion LPAD(cad1,n[,cad2]).- Esta función añade caracteres a la izquierda de cad1, hasta que alcance una cierta longitud. Devuelve cad1 en longitud de N y ajustado a la derecha. Cad2 es la cadena con la que se rellena por la izquierda, cad1 puede una columna de una tabla o cualquier literal si cad2 se suprime asume como carácter de relleno el blanco.

```
SQL> select lpad('x',5,'*'),lpad('x',7,'*.') from dual;
LPAD( LPAD('X
----- -------
****x *.*.*.x
```

5.Para cada fila de la tabla notas_alumnos obtener el nombre del alumno en una longitud de 30 caracteres rellenandolo por la izquierda por punto.

```
SQL> select lpad(nombre_alumno,30,'.')"rellena con puntos" from notas_alumnos;
rellena con puntos
------------------------------
......Alcalde García, M. Luisa
.....Benito Martín, Luis
.....Casas Martínez, Manuel
.....Corregidor Sánchez, Ana
.....Díaz Sánchez, María
```

Funcion RPAD(cad1,n[,cad2]).-

```
SQL> select rpad('x',5,'*'),rpad('x',7,'*.') from dual;
RPAD( RPAD('X
----- -------
x**** x*.*.*.
```

```
SQL> select rpad(nombre_alumno,30,'.')"rellena con puntos" from notas_alumnos;
rellena con puntos
------------------------------
Alcalde García, M. Luisa.....
Benito Martín, Luis.....
Casas Martínez, Manuel.....
Corregidor Sánchez, Ana.....
Díaz Sánchez, María.....
```

Funcion LTRIM(cad[,set]).- Suprime un conjunto de caracteres a la izquierda de la cadena. El nombre de la cadena puede ser el nombre de una columna o de una tabla o una cadena literal, y SET es la colección de caracteres que se van a suprimir. Devuelve entre “cad” con el grupo e caracteres “SET” omitidos por la cadena de caracteres. Es el segundo parámetro se omite devuelve la misma cadena. Por defecto si la cadena contiene blanco a la izquierda y se omite el segundo parámetro la funcion devuelve la cadena sin blanco a la izquierda.

6.Utilizar la funcion ltrim sin el segundo parámetro y con una cadena con blanco a la izquierda.

```
SQL> select ltrim('     hola') from  dual;
LTRI
----
hola
SQL> select ltrim('    hola') || ltrim('    adios') from dual;
LTRIM('HO
---------
holaadios
```

7.Suprimir los carateres ‘a’ y ‘b’ a la izquierda de la cadena.

```
SQL> select ltrim('abaAabUNIDAD IV - FUNCIONES','ab') from dual;

LTRIM('ABAAABUNIDADIV-FU
------------------------
AabUNIDAD IV – FUNCIONES
```

Funcion RTRIM(cad[,set]).- Suprime un cnjunto de caracteres a la derecha de la cadena. El nombre de la cadena puede ser el nombre de una columna o de una tabla o una cadena literal, y SET es la colección de caracteres que se van a suprimir. Devuelve entre “cad” con el grupo e caracteres “SET” omitidos por la cadena de caracteres. Es el segundo parámetro se omite devuelve la misma cadena. Por defecto si la cadena contiene blanco a la derecha y se omite el segundo parámetro la funcion devuelve la cadena sin blanco a la derecha.

8.‘PROBEMOS CON ab’

```
SQL> select rtrim('PROBEMOS CON ab','ab') from dual;
RTRIM('PROBEM
-------------
PROBEMOS CON
```

9.En la tabla mistextos de la columna titulo quitar los puntos y comilla a la derecha y las comillas a la izquierda.

```
SQL> select ltrim(rtrim(titulo,'.'),'')"titulos rectificados" from mistextos;
titulos rectificados
--------------------------------
METODOLOGÍA DE LA PROGRAMACIÓN
INFORMÁTICA BÁSICA
SISTEMAS OPERATIVOS
SISTEMAS DIGITALES
MANUAL DE C
```

Funcion REPLACE(cad,cadena_busqueda[,cadena_sustitucion]).- Sustituye un carácter o varios caracteres de una cadena con 0 o mas caracteres. Devuelve cad con cada ocurrencia de encadena de búsqueda sustituida por cadena sustitución.

10.Sustituir la letra ‘o’ * ‘a’ en la cadena ‘blanco y negro’.

```
SQL> select replace('blanco y negro','o','a') from dual;

REPLACE('BLANC
--------------
blanca y negra
```

11.Sustituir la letra ‘o’ * una ‘as’ en la cadena ‘blanco y negro’.

```
SQL> select replace('blanco y negro','o','as') from dual;
REPLACE('BLANCOY
----------------
blancas y negras
```

NOTA: Si no ponemos nada en cadena de sustitución se sustituira la cadena de búsqueda por nada(NULL).

Funcion SUBSTR(cad,inicio[,n]).- Extrae una parte de la cadena. Devuelve la subscadena de cad que abarca desde la posición indicada en inicio hasta tantos caracteres como indique el numero ‘n’. si se omite ‘n’ devuelve la cadena desde la primera posición indicada en inicio. El valor de ‘n’ no puede ser inferior a 1. el valor de inicio puede ser negativo cuyo caso devuelve la cadena empesando por el final yendo de derecha hacia izquierda.

11.Partiendo de la cadena ABCDEFG obtener 2 caracteres a partir de la 3ª posición.

```
SQL> select substr('ABCDEFG',3,2) from dual;
SU
--
CD
```

12.Obtener dos caracteres de la misma posición empezando por el fin de la cadena.

```
SQL> select substr('ABCDEFG',-3,2) from dual;
SU
--
EF
```

13.Partiendo de la misma cadena obtener la nueva cadena a partir de la posición cuarta.

```
SQL> select substr('ABCDEFG',4) from dual;
SUBS
----
DEFG
```

14.Visualizar el apellido de cada empleado con su primera letra.

```
SQL> select substr(apellido,1,1) from emple;
S
-
S
A
S
J
M
N
C
G
R
T
A
J
F
M
```

Funcion TRANSLATE(cad1,cad2,cad3).- Esta función convierte caracteres de una cadena en caracteres diferentes de acuerdo con un plan de substitución que marca el usuario. Devuelve “cad1” con los caracteres encontrado en “cad2” y substituido por los caracteres de “cad3”. Cualquier carácter que no este en “cad2” permanece como estaba.

15.‘SQL PLUS’,’SQL’,123’

```
SQL> select translate('SQL PLUS','SQL',123) from dual;
TRANSLAT
--------
123 P3U1
```

Para realizar esta funcion comprueba la posición de los caracteres de “cad2”->(SQL) con los de la “cad3”->(123) y sustituye la s por el 1, la q por el 2 y la l por el 3.

```
SQL> select translate('LOS PILARES DE LA TIERRA','LAEIOU','laeiou')"frase cambiada" from
dual;
frase cambiada
------------------------
loS PilaReS De la TieRRa
```

15.Deducir los apellidos de los trabajadores ke tienen el mismo oficio ke gil;

```
SQL> SELECT APELLIDO
  2  FROM EMPLE
  3  WHERE OFICIO=(SELECT OFICIO
  4  FROM EMPLE
  5  WHERE APELLIDO='GIL');

APELLIDO
----------
GIL
FERNÁNDEZ
```

4.4 Funciones para el manejo de fecha.

Oracle almacena datos de tipo fecha (Date), y posee una utilidad para formatear las fechas muy amplias. Tiene un formato por omisión: ‘**DD-MON-YY**’ pero con la función **TO_CHAR** es posible mostrar la fecha de cualquier modo. Los literales de fechas deben de encerrarse siempre entre comilla simples.
El tipo de dato DATE se almacena en un formato especial que incluye ‘MES/DIA/AÑO/HORA/MIN/SEG’.

Funcion SYDATE.- Esta funcion devuelve la fecha del sistema.

```
SQL> select sysdate from dual;

SYSDATE
--------
30/10/06
```

Funcion ADD-MONTHS(fecha .n).- Esta funcion devuelve la fecha incrementada en n meses.

```
SQL> select add_months(sysdate,7) from dual;

ADD_MONT
--------
30/05/06
```

1.utilizando emple, suma 2 meses a la fecha de alta de todos los empleados!!

```
SQL> select add_months(fecha_alt,2) from emple;

ADD_MONT
--------
17/02/81
20/04/80
22/04/81
02/06/81
29/11/81
01/07/81
09/08/81
09/01/82
17/01/82
08/11/81
23/11/81
03/02/82
03/02/82
23/03/82


SQL> select add_months(fecha_alt,-10) from emple;

ADD_MONT
--------
17/02/80
20/04/79
22/04/80
02/06/80
29/11/80
01/07/80
09/08/80
09/01/81
17/01/81
08/11/80
23/11/80
03/02/81
```

LAST_DAY(fecha):devuelve la fecha dle ultimo dia del mes que contiene fecha :

2.Averiguar el último dia del mes de la sig. Fecha 5/2/00 y 5/2/01:

```
SQL> select last_day('5/2/00') from dual;

LAST_DAY
--------
29/02/00

SQL> select last_day('5/2/01'),last_day( '5/2/00') from dual;

LAST_DAY  LAST_DAY
--------  --------
28/02/01  29/02/00
```

3.Obtener de la tabla emple el ultimo dia de mes para cada una de las fechas de alta:

```
SQL> select last_day(fecha_alt) from emple;

LAST_DAY
--------
31/12/80
29/02/80
28/02/81
30/04/81
30/09/81
31/05/81
30/06/81
30/11/81
30/11/81
30/09/81
30/09/81
31/12/81
31/12/81
31/01/82
```

MONTHS_BETWEEN(fecha1,fecha2)→devuelve la diferencia en meses entre las 2 fechas.puede ser un numero decimal.

```
SQL>  select months_between('5/1/00','12/8/33') from dual;

MONTHS_BETWEEN('5/1/00','12/8/33')
----------------------------------
                  -403,22581
```

4.Los mese que hemos vivido:

```
SQL>  select months_between(sysdate,'5/1/1985') from dual;

MONTHS_BETWEEN(SYSDATE,'5/1/1985')
----------------------------------
                  261,828455
```

5.Los años que tengo:

```
SQL> select months_between(sysdate,'5/1/1985')/12 from dual;

MONTHS_BETWEEN(SYSDATE,'5/1/1985')/12
-------------------------------------
                   21,8190406
```

NEXT_DAY(fecha,cad)→Devuelve la fecha del primer dia de la semana indicado por cad, despues de la fecha indicada por fecha;
El dia de la semana en cad se indica por su nombre es decir: lunes(Monday), martes(Tuesday),…..

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

NEXT_DAY
--------
02/11/06
```

4.5. FUNCIONES DE CONVERSION.

Son aquellas que cambian los objetos de una forma especial es decir , transforman un tipo de datos en otro.
Las funciones de conversión elementales son:

TO_CHAR(fecha,formato)→convierte una fecha (de tipo DATE) a tipo varchar2 en el formato especificado. El formato es una cadena de caracteres que puede incluir las mascaras de formato definidas en el control de formato de fechas y donde es posible incluir literales definidos por nosotros encerrados entre comillas dobles.

Mascaras de formato numerico:
Cc—valor del siglo
yy—ultimos 2 digitos del año

yyyy-año
q—nº del trimestre
ww—numero de semana del año
w—numero de semana del mes
mm—numero de mes
dd—numero del dia del mes
d—numero del dia de la semana
hh24—la hora
mi—minutos
ss—segundos

mascara formato aracteres:
year—año
month—mes/si se pone mon solo salen las 3 primeras letras
day—dia de la semana

```
SQL> select to_char(sysdate,'cc/q/yyyy/mm/dd hh24:mi:ss') from dual;

TO_CHAR(SYSDATE,'CC/Q/YY
------------------------
21/4/2006/10/30 16:47:41


SQL>  select to_char(sysdate,'day   mon/yyyy    hh24:mi:ss') from dual;

TO_CHAR(SYSDATE,'DAYMON/YYYYHH24
--------------------------------
lunes       oct/2006    16:53:53
```

7.Sumarle 5 horas a la hora actual:

```
SQL>  select to_char(sysdate,'hh24')+5 from dual;

TO_CHAR(SYSDATE,'HH24')+5
-------------------------
                       21


SQL>  select sysdate,to_char(sysdate+5/24,'hh24:mi:ss') from dual;

SYSDATE  TO_CHAR(
-------- --------
30/10/06 22:06:21


SQL>  select sysdate,to_char(sysdate+60*24,'hh24:mi:ss') from dual;

SYSDATE  TO_CHAR(
-------- --------
30/10/06 17:07:25
suma a los minutos;


SQL>  select sysdate,to_char(sysdate+0/24+5/60+10/60/60,'hh24:mi:ss') from dual;

SYSDATE  TO_CHAR(
-------- --------
30/10/06 19:15:58
suma horas, minutos,segundos.
```

7.A partir de la tabla emple obtener la fecha de alta, formateada, de manera ke aparezca el nombre del mes, el numero de dia del mes y el año:

```
SQL> select to_char(fecha_alt,'month/dd/yyyy') from emple;

TO_CHAR(FECHA_ALT,
------------------
diciembre /17/1980
febrero   /20/1980
febrero   /22/1981
abril     /02/1981
septiembre/29/1981
mayo      /01/1981
junio     /09/1981
noviembre /09/1981
noviembre /17/1981
septiembre/08/1981
septiembre/23/1981
diciembre /03/1981
diciembre /03/1981
enero     /23/1982
```

NOTA:Si ponemos month entero, nos sale el nombre completo del mes, si se acorta nos salen menos letras!!

8.Obtener a fecha de alta de modo ke aparezca el nombre del mes con 3 letras en mayuscula, el numero del dia del año,el ultimo digito del añoy los tres ultimos digitos del año!!

SQL> select to_char(**fecha_alt,'MON/ddd/y/yyy'**) from emple;

```
TO_CHAR(FECHA
-------------
DIC/352/0/980
FEB/051/0/980
FEB/053/1/981
ABR/092/1/981
SEP/272/1/981
MAY/121/1/981
JUN/160/1/981
NOV/313/1/981
NOV/321/1/981
SEP/251/1/981
SEP/266/1/981
DIC/337/1/981
DIC/337/1/981
ENE/023/2/982
```

NOTA: Si ponemos mon en mayusculas nos muestra el mes en mayusculas, ddd es el dia del año,y el ultimo digito del año,yyy los 3 ultimos digitos del año;

9.Fecha de hoy con el sig formato:

hoy es lunes,30 de octubre de 2006

SQL> select to_char(sysdate,'"hoy es" day dd "de" month "de" yyyy') from dual;

```
TO_CHAR(SYSDATE,'"HOYES"DAYDD"DE"MONTH"DE
------------------------------------------
hoy es lunes     30 de octubre    de 2006
```

TO_NUMBER(cadena[,formato])→convierte la cadena “cadena” a tipo number, según el formato especificado. La cadena ha de contener números, el carácter decimal o el signo menos a la izquierda. No puede haber espacios entre los números ni otros caracteres.

10.Transformar en numerico ese valor alfanumerico:

SQL> select to_number('-123456') from dual;

```
TO_NUMBER('-123456')
--------------------
            -123456
```

```
SQL>  select to_number('123.45','999.99') from dual;

TO_NUMBER('123.45','999.99')
----------------------------
                      123,45
```

```
SQL>  select to_number('123.456','999G999') from dual;

TO_NUMBER('123.456','999G999')
------------------------------
                        123456
```

NOTA:Fijarse que quita el punto decimal!!por ke no lo considera como tal!!

TO_DATE(cad,'formato')→convierte cad de tipo varchar2 o char a un valor de tipo date(fecha).
El formato de fecha elegido es "formato".
Convertirla cadena 01012006:

```
SQL> select to_date('01012006') from dual;

TO_DATE(
--------
01/01/06
```

11.Transformar a fecha:

```
SQL> select to_date('010103') from dual;

TO_DATE(
--------
01/01/03
```

12.Obtener el nombre del mes a partir de la cadena 01012001:

```
SQL>  select to_char (to_date('01012001'),'month') from dual;

TO_CHAR(TO
----------
enero
```

TO_CHAR (número, formato) esta función convierte un numero de tipo NUMBER a tipo VARCHAR2.

Elemento	Ejemplo	Descripción
9	999	Devuelve el valor con el número especificado de dígitos. Si es positivo deja un espacio. Devuelve el valor con el número especificado de dígitos con el signo menos si es negativo. Si el valor tiene ceros a la izquierda los deja en blanco, excepto si el valor es 0.
0	990	
S	S99 999S	S representa el signo. Devuelve el valor con el signo + si es positivo o con el signo – si es negativo
, (coma)	9,999	Devuelve la , en la posición especificada (carácter de los miles)
.(punto)	99.99	
RN	RN	Devuelve el valor en números romanos. RN devuelve el valor en mayúsculas y N en minus.

SQL> select to_char (1,'999'), to_char (-1,'999'), to_char (01,'999'), to_char (0,'999') "To_Char" from dual;

```
TO_C  TO_C  TO_C  To_C
----  ----  ----  ----
 1    -1     1    0
```

1. Devuelve ceros a la izquierda, signos,

SQL> select to_char (10,'0999'), to_char (10,'990'),to_char (10,'90090') from dual;

```
TO_CH   TO_C   TO_CHA
-----   ----   ------
0010    10     0010
```

SQL> select to_char (-55,'999S'), to_char (-55,'S999'), to_char (55,'S999'), to_char (55,'999S') from dual;

```
TO_CH   TO_C   TO_CHA  TO_CH
----    ----   ----    ----
55-     -55    +55     55+
```

ALTER SESSION

SQL> alter session set nls_numeric_characters="/*";

Sesión modificada.

SQL> select to_char (12345.67,'999G999D999') from dual;

TO_CHAR(1234

```
------------
12*345/670

SQL> alter session set nls_date_format='dd-mm-yyyy';

Sesión modificada.

SQL> select sysdate from dual;

SYSDATE
----------
31-10-2006
```

TO_NUMBER (cadena [, formato]) → convierte la cadena 'cadena' a tipo NUMBER según el formato especificado. La cadena a de contener números, el carácter decimal(.) o el signo menos a la izquierda(-). No puede haber espacios entre los números, ni otros caracteres.

2.transformar en numérico: (-123456) ('123,456','999,99')
En el segundo hay que cambiar las ',' de decimales por '.', o bien hacer un alter session.

```
SQL> select to_number(-123456) from dual;

TO_NUMBER(-123456)
------------------
          -123456



SQL> select to_number('123.45','999.99')from dual;

TO_NUMBER('123.45','999.99')
----------------------------
                  123,45


*SQL> select to_number('123,99','999D99')from dual;

  TO_NUMBER('123,99','999D99')
  ----------------------------
                  123,99


*SQL> select to_number('123.456','999G999')from dual;

  TO_NUMBER('123.456','999G999')
  ------------------------------
                    123456
```

TO_DATE(cad, formato) → convierte 'cad' de tipo VARCHAR2 o CHAR a un valor de tipo 'date'. El formato de fecha elegido es 'fromato'.

Ejercicio: convertir la cadena 01012006

```
SQL> select to_date('01012006')from dual;
```

```
TO_DATE(
--------
01/01/06
```

3.transformar a fecha la cadena de caracteres 010103

```
SQL> select to_date('010103')from dual;

TO_DATE(
--------
01/01/03
```

4.obtener el nombre del mes a partir de la cadena 01012001

```
SQL> select to_char(to_date('01/01/06'),'month')from dual;

TO_CHAR(TO
----------
enero
```

5.Obtener el nombre del día, el nombre del mes, el día y el año a partir de la cadena '12121997'

```
SQL> select to_char(to_date('12121997'),'day/month/yyyy')from dual;

TO_CHAR(TO_DATE('12121997
-------------------------
viernes  /diciembre /1997
```

4.6. OTRAS FUNCIONES:

DECODE (var ,val1,cad1,val2,cad2,…….,valor por defecto)→esta funcion sustituye un valor por otro. Si 'var' es igual a cualquier valor de la lista (val1,val2…..) devuelve el correspondiente codigo(cod1,cod2), en caso contrario devuelve el valor señalado por defecto.
Seleccionar todas la filas y codificar el oficio. Si el oficio es presidente codificarlo con un 1, si es empleado, con un 2 y en cualquier otro caso con un 5:

```
SQL>  select oficio, decode(oficio,'PRESIDENTE',1,'EMPLEADO',2,5) "codigo" from emple;

OFICIO          codigo
----------      ----------
EMPLEADO          2
VENDEDOR          5
VENDEDOR          5
DIRECTOR          5
VENDEDOR          5
DIRECTOR          5
DIRECTOR          5
ANALISTA          5
PRESIDENTE        1
```

1.Dada la tabla librería visualizar todas las filas sustituyendo dibujo, diseño y labores-hogar; en cualquiier otro caso dejar el tema como esta.

```
TEMA           DECODE
-------------- ------
Informática
Economía
Deportes
Filosofía
Dibujo
Medicina
Biología
Geología
Sociedad
Labores
Jardinería
```

NOTA: Funciona bien, ya que en esta tabla los temas estan como char[15] y nosotros solo hemos ocupado lo que ocupa la palabra, si añadimos al la funcion los espacios suficientes para llenar los 15, si sale;

VSIZE(expresión)→ devuelve el numero de bytes ke ocupa esa expresión.

Averiguar el numero de bytes ke ocupa kada apellido de la tabla emple de los empleados del departamento 10;

```
SQL> select apellido,vsize(apellido) from emple where dept_no=10;

APELLIDO   VSIZE(APELLIDO)
---------- ---------------
CEREZO                   6
REY                      3
MUÑOZ                    5
```

USER→Que usuario somos.

```
SQL>  SELECT USER FROM DUAL;

USER
------------------------------
MONICA
```

```
SQL> show user;
USER es "MONICA"
```

UID:

SQL> select user, uid from dual: Nos dice nuestro uid!!

```
USER                           UID
------------------------------ ----------
MONICA                             80
```

5.CLAUSULAS AVANZADAS DE SELECCIÓN

5.1. Introducción.

En esta unidad nos ocuparemos de varias clausulas que acompañan a la consulta select lo. Nos permiitra agrupar filas de una tabla según las condicion para obtener resultado referente a las filas agrupadas.
Tambien podremos ralizar combinaciones de tabla algunas filas que no tengan correspondencia con la otra tabla.
Podremos combinar resultados de varias sentencia SELECT, utilizando operadores de conjunto.

5.2. Agrupación de elementos. GRUOP BY y HAVING.-

Hemos utilizados la sentencia select para recuperar filas de una tabla y la clausulas where el numero de filas seleccionadas . Se han utilizados funciones de grupo para trabajar con conjunto de filas: Media aritmetica (AVG), suma de salarios (SUM). Esto se á realizado pensando en que este conjunto de filas era un grupo.

Ejemplo

```
SELECT DEPT_NO, AVG(SALARIO)
FROM EMPLE
GRUOP BY DEPT_NO;
```

```
SQL> SELECT DEPT_NO,AVG(SALARIO)
  2  FROM EMPLE
  3  GROUP BY DEPT_NO;

   DEPT_NO AVG(SALARIO)
---------- ------------
        10   379166,667
        20        282750
        30   203666,667
```

La sentencia select quedaria de la siguiente manera:

SELECT COLUMNA……

FROM ……...
GROUP BY ……..
HAVING CONDICION ……...
ORDER BY ……..

Los datos seleccionados en la sentencias select que llevan el group by deben ser una constante, (), o una columna seleccionada en el group by .
La cláusula group by sirve para calcular propiedades de uno o mas conjuntos de filas. GROUP BY controla que las filas de la tabla original sehan agrupadas por una tabla temporal. Del mismo modo que existe la selección de búsqueda where para filas individuales hay también una condición de búsqueda HAVING para grupos de filas. La cláusula having se emplea para controlar cuales de los conjuntos de filas se quiere visualizar. No existe la cláusula having sin un goup by.
1.Visualizar el numero de empleado que hay por departamento de la tabla emple.

```
SQL> select dept_no,count(apellido)
  2  from emple
  3  group by dept_no;

   DEPT_NO COUNT(APELLIDO)
---------- ---------------
        10               3
        20               5
        30               6


SQL> select dept_no,count(apellido)"numero"
  2  from emple
  3  group by dept_no
  4  HAVING count(apellido)>4;

   DEPT_NO     numero
---------- ----------
        20          5
        30          6


SQL> select dept_no,count(apellido)"numero"
  2  from emple
  3  group by dept_no
  4  HAVING count(apellido)>4
  5  order by dept_no desc;

   DEPT_NO     numero
---------- ----------
        30          6
        20          5
```

Cuando se utiliza la cláusula order by con columna y funciones de grupo se á de tener en cuenta que esta se ejecuta detrás de where group by y having.
La evaluacion de la clausulas en tiempo de ejecución se efectúan en el siguiente orden:

Where selecciona filas
Group by agrupas estas fila
Having filtra los grupo selección, elimina grupos.
Orderr by clasifica los grupos. Ordena las filas.

2.Considerando las tablas de emple y depart , obtener la suma de salario, el salario máximo y el salario mínimo por cada departamento. Las salidas formateadas y utilizando alias de columna.

```
SQL> select dept_no,sum(salario),max(salario),min(salario)
  2  from emple
  3  group by dept_no
  4  order by to_char(sum(salario),'999G999D999');

   DEPT_NO SUM(SALARIO) MAX(SALARIO) MIN(SALARIO)
---------- ------------ ------------ ------------
        10      1137500       650000       169000
        20      1413750       390000       104000
        30      1222000       370500       123500
```

3.Obtener los nombres de departamento que tengan mas de 4 personas trabajando.

```
SQL> select dept_no, dnombre from depart
  2   where dept_no in(select dept_no from emple group by dept_no having
count(dept_no)>4);

   DEPT_NO DNOMBRE
---------- --------------
        20 INVESTIGACIÓON
        30 VENTAS
```

5.3.Combinación externa (outer joins).

Existe una variedad de combinación de tablas llamada outer join que nos permite seleccionar algunas filas de la otra tabla con la que se combina; el formato es el siguiente:

SELECT tabla1.columna1,tabla1.columna2,tabla2.columna1,……
FROM tabla1, tabla2
WHERE tabla1.columna1= tabla2.columna2(+) ;

Esta select seleccionará, todas las filas de la tabla1 aunque no tengan correspondencia con las filas de la tabla 2. Se señalara con el símbolo (+) detrás de la columna de la tabla2 en la cláusula where; el reto de columnas de la tabla2, se rellenara con nulos.

```
SQL> select emple.dept_no, depart.dept_no,emple.apellido,depart.dnombre
```

```
  2  from emple,depart
  3  where emple.dept_no=depart.dept_no(+);

   DEPT_NO    DEPT_NO APELLIDO   DNOMBRE
---------- ---------- ---------- --------------
        10         10 CEREZO     CONTABILIDAD
        10         10 REY        CONTABILIDAD
        10         10 MUÑOZ      CONTABILIDAD
        20         20 SÁNCHEZ    INVESTIGACIÓON
        20         20 ALONSO     INVESTIGACIÓON
        20         20 FERNÁNDEZ  INVESTIGACIÓON
        20         20 GIL        INVESTIGACIÓON
        20         20 JIMÉNEZ    INVESTIGACIÓON
        30         30 ARROYO     VENTAS
        30         30 NEGRO      VENTAS
        30         30 MARTÍN     VENTAS
        30         30 JIMENO     VENTAS
        30         30 TOVAR      VENTAS
        30         30 SALA       VENTAS

14 filas seleccionadas.
```

NOTA: aquí hemos puesto el (+) pero no sale nada, debido que hemos colocado el (+) en mal sitio.

```
SQL> select emple.dept_no,depart.dept_no,emple.apellido,depart.dnombre
  2  from emple,depart
  3  where depart.dept_no=emple.dept_no(+);

   DEPT_NO    DEPT_NO APELLIDO   DNOMBRE
----------          ---------- ---------- --------------
        10         10 CEREZO              CONTABILIDAD
        10         10 REY                 CONTABILIDAD
        10         10 MUÑOZ               CONTABILIDAD
        20         20 SÁNCHEZ                    INVESTIGACIÓON
        20         20 ALONSO                     INVESTIGACIÓON
        20         20 FERNÁNDEZ           INVESTIGACIÓON
        20         20 GIL                 INVESTIGACIÓON
        20         20 JIMÉNEZ             INVESTIGACIÓON
        30         30 ARROYO                     VENTAS
        30         30 NEGRO               VENTAS
        30         30 MARTÍN                     VENTAS
        30         30 JIMENO              VENTAS
        30         30 TOVAR               VENTAS
        30         30 SALA                VENTAS
        40                                       PRODUCCIÓN
```

NOTA: hay que ponerle el (+) a aquella tabla que le falte el dato para que sean iguales y rellene con nulos, los datos que faltan para que así salga la correspondencia de la tabla que si tiene el dato.

5.4. Unión, intersección y minus.

Los operadores relacionales unión intersect y minus son operadores de conjunto. Los conjuntos son las filas resultantes de cualquier sentencia select que permiten combinar los resultados de varias select para obtener un único resultado.
El formato de select para estos operadores es:

SELECT… FROM… WHERE…
Operador de conjunto.
SELECT… FROM… WHERE…

UNION combina los resultados de 2 consultas. Las filas duplicadas que aparezcan se reducen a una fila única. Su formato es:

SELECT col1, col2 FROM tabla1 WHERE condicion
UNION
SELECT col1, col2 FROM tabla2 WHERE condicion

1.Disponemos de 3 tablas, alum contiene los nombres de los alumnos que actualmente están en el centro. Nuevos contiene los nombres de los futuros alumnos, y antiguos contiene los nombres de antiguos alumnos del centro:

Visualizar los nombres de los alumnos actuales y de los futuros alumnos.

SQL>select nombre from alum union select nombre from nuevos;

```
NOMBRE
--------------------
ANA
ERNESTO
JUAN
LUISA
MAITE
MARÍA
PEDRO
RAQUEL
SOFÍA
```

UNION ALL combina los resultados de las 2 consultas. Cualquier duplicación de filas que se den en el resultado final aparecerá en la consulta.

INTERSECT devuelve las filas que son iguales en ambas consultas. Todas las filas duplicadas se eliminaran antes del resultado final.

2.Obtener los nombres de los alumnos que están actualmente en el centro y que estuvieron en el centro hace un tiempo.

SQL>select nombre from alum intersect select nombre from antiguos;

```
NOMBRE
--------------------
```

ERNESTO
MARÍA

MINUS devuelve aquellas filas que están en la primera select y no están en la segunda. Las filas duplicadas del primer conjunto se reducirán a una fila única antes de que empiece la comparación con el otro conjunto.

3.Obtener los nombres y la localidad de los alumnos que están actualmente en el centro y que nunca estuvieron anteriormente en el.

SQL>select nombre from alum minus select nombre from antiguos;

NOMBRE

ANA
JUAN
LUISA
PEDRO
RAQUEL

4.Seleccionar los nombre de la tabla alum que estén en nuevos y no estén en antiguos.

SQL>select nombre from alum intersect select nombre from nuevos minus select nombre from antiguos;

NOMBRE

ANA
JUAN

5.4.1- Reglas para los operadores de conjunto.

Estos operadores se pueden manejar con consultas a diferentes tablas con la siguiente regla:

1. Las columnas de las dos columnas se relacionan en orden de izquierda a derecha.
2. Los nombres de columna de la primera sentencia select no tienen por que ser los mismo que los nombre de la segunda.
3. Las select necesitan tener el mismo número de columnas.
4. Los tipos de datos deben coincidir aunque la longitud no tiene porque se r la misma.

6. MANIPULACION DE DATOS: INSERT, UPDATE Y DELETE.

6.1- Inserción de datos. Orden INSERT.

La orden INSERT nos permite añadir filas de datos en una tabla. Su formato es este:

INSERT INTO nombre_tabla [(columna [, columna])]
VALUES (valor[,valor…]);

Nombre_tabla es la tabla donde se vana insertar filas.
[(columna [, columna])] representa las columnas donde se van a introducir datos / valores. Si las columnas no se especifican en la cláusula INSERT, se consideran por defecto todas las columnas de la tabla
(valor[,valor…]) representa los valores que se van a dar en la columnas. Los valores se deben corresponder con cada una de las columnas que aparecen. Además deben coincidir con el tipo de datos definido para la columna. Cualquier columna que no este en la lista de columnas recibirá el valor NULL, siempre y cuando no este definida como NOT NULL, en cuyo caso INSERT nos anunciara el error. Si no se da la lista de columnas, se han de introducir valores en todas las columnas

2 Demos de alta a una profesora con apellidos Quiroga Martín, A. Isabel, de especialidad informático y con un código de centro 45.
3 insert into profesores (cod_centro, apellidos, especialidad) values (45, 'Quiroga Martín, A.Isabel', 'INFORMÁTICA');

1 fila creada.

Al dar de alta una fila ejecutando la sentencia INSERT se emite el mensaje de 1 fila creada, con el que se indica que se ha insertado correctamente. Conviene observar que:

Las columnas a las que damos valores se identifican por su nombre.
La asociación columna-valor es posicional.
Los valores que se dan alas columnas deben coincidir con el tipo de datos definido.
Los valores constantes de tipo carácter (alfanumérico) han de ir encerrados en comillas simples. Los de tipo fecha también.
Insertar a un profesor que no tiene código de centro de apellidos Seco Jiménez, Ernesto de la especialidad de lengua.
No se puede hacer la inserción porque el código en not null y no le pasamos nada. Por lo tanto es imposible crearla.

1.Insertar a un profesor de apellidos González Toledo código de centro 22 DNI 12345678 y especialidad historia.

SQL>insert into profesores (cod_centro,dni,apellidos,especialidad) values (22,12345678,'Gonzalez Toledo','HISTORIA');

```
COD_CENTRO       DNI APELLIDOS                    ESPECIALIDAD
---------- ---------- ------------------------------ ----------------
        10    1112345 Martínez Salas, Fernando      INFORMÁTICA
        10    4123005 Bueno Zarco, Elisa            MATEMÁTICAS
        10    4122025 Montes García, M.Pilar        MATEMÁTICAS
        15    9800990 Ramos Ruiz, Luis              LENGUA
        15    1112345 Rivera Silvestre, Ana         DIBUJO
        15    8660990 De Lucas Fdez, M.Angel        LENGUA
        22    7650000 Ruiz Lafuente, Manuel         MATEMÁTICAS
        45   43526789 Serrano Laguía, María         INFORMÁTICA
        22   12345678 Gonzalez Toledo               HISTORIA
```

6.1.1.inserción con select

Si añadimos a insert una consulta es decir una sentencia select , se añaden, tantas filas como devuelve la consulta. Le formato es el siguiente:

INSERT INTO nomb_tabla [(columna1[, columna2]-)]
SELECT (columna1[,columna1]…. | *)
From nom_tabla [clausulas];

```
SQL>  insert into profesores  (cod_centro,dni,apellidos,especialidad)
  2  values(99,87654321,'aaa,bb','historia');

1 fila creada.

SQL> select * from profesores;

COD_CENTRO       DNI APELLIDOS                    ESPECIALIDAD
---------- ---------- ------------------------------ ----------------
        10    1112345 Martínez Salas, Fernando      INFORMÁTICA
        10    4123005 Bueno Zarco, Elisa            MATEMÁTICAS
        10    4122025 Montes García, M.Pilar        MATEMÁTICAS
        15    9800990 Ramos Ruiz, Luis              LENGUA
        15    1112345 Rivera Silvestre, Ana         DIBUJO
        15    8660990 De Lucas Fdez, M.Angel        LENGUA
        22    7650000 Ruiz Lafuente, Manuel         MATEMÁTICAS
        45   43526789 Serrano Laguía, María         INFORMÁTICA
        99   87654321 aaa,bb                        historia
```

Para copiar de una tabla a otra, se procede a hacer lo siguiente:

1.insertar todas las filas con el dept_no = a 30 de la tabla emple a emple30.

```
SQL> insert into emple30
  2  select emp_no,apellido,oficio,dir,fecha_alt,salario,comision,dept_no
  3  from emple
```

```
  4  where dept_no=30;
```

6 filas creadas.

En el insert nosrefeimos a la tabla a la ke hay que copiar,en el select hacemos referencia a todas las columnas(si van a ser todas, se pone *),en el from ponemos la tabla de la que copiamos y en el where la clausula con la que copiaremos ciertas columnas.

```
SQL> select * from emple30;

    EMP_NO APELLIDO   OFICIO           DIR FECHA_ALT     SALARIO   COMISION
DEPT_NO
---------- ---------- ---------- ---------- ---------- ---------- ---------- ----------
      7499 ARROYO     VENDEDOR         7698 20/02/1980     208000      39000         30
      7521 SALA       VENDEDOR         7698 22/02/1981     162500      65000         30
      7654 MARTÍN     VENDEDOR         7698 29/09/1981     162500     182000         30
      7698 NEGRO      DIRECTOR         7839 01/05/1981     370500                    30
      7844 TOVAR      VENDEDOR         7698 08/09/1981     195000          0         30
      7900 JIMENO     EMPLEADO         7698 03/12/1981     123500                    30
```

6 filas seleccionadas.

ROLLBACK;→se utiliza para borrar lo creado en la tabla anteriormente.

2.Insertar en la tabla nombres, en la columna nombre el apellido de los empleados de la tabla emple, ke sean de Dep.=20:

```
SQL> insert into nombres (nombre)
  2  select (apellido)
  3  from emple
  4  where dept_no=20;
```

5 filas creadas.

3.Insertar a un empleado de apellido Quiroga con numero de empleado
1115 en la tabla emple.los restantes datos del empleado seran los mismos que los de gil y la fecha de alta es la de hoy.

```
SQL>  insert into emple
  2  select 1115,'quiroga',oficio,dir,sysdate,salario,comision,dept_no
  3  from emple
  4  where apellido='GIL';
```

1 fila creada.

```
SQL>  select * from emple;

    EMP_NO APELLIDO   OFICIO           DIR FECHA_ALT     SALARIO   COMISION
DEPT_NO
---------- ---------- ---------- ---------- ---------- ---------- ---------- ----------
```

```
7369 SÁNCHEZ    EMPLEADO        7902 17/12/1980     104000              20
7499 ARROYO     VENDEDOR        7698 20/02/1980     208000      39000       30
7521 SALA       VENDEDOR        7698 22/02/1981     162500      65000       30
7566 JIMÉNEZ    DIRECTOR        7839 02/04/1981     386750              20
7654 MARTÍN     VENDEDOR        7698 29/09/1981     162500      182000      30
7698 NEGRO      DIRECTOR        7839 01/05/1981     370500              30
7782 CEREZO     DIRECTOR        7839 09/06/1981     318500              10
7788 GIL        ANALISTA        7566 09/11/1981     390000              20
7839 REY        PRESIDENTE          17/11/1981      650000              10
7844 TOVAR      VENDEDOR        7698 08/09/1981     195000      0       30
7876 ALONSO     EMPLEADO        7788 23/09/1981     143000              20
7900 JIMENO     EMPLEADO        7698 03/12/1981     123500              30
7902 FERNÁNDEZ  ANALISTA        7566 03/12/1981     390000              20
7934 MUÑOZ      EMPLEADO        7782 23/01/1982     169000              10
1115 quiroga    ANALISTA        7566 14/11/2006     390000              20
```

6.3.Modificacion UPDATE

Esta orden se utiliza para actualizar valores de columnas (tanto a una como a varias filas de la tabla; formato:

UPDATE nom_tabla
SET columna1= var1,…,columnaN = varN
WHERE condición;

Sin condicion, cambia todas las filas de la columna, con condicion, cambia la fila que le digamos.

```
SQL> update emple
  2  set oficio='escuchante';

15 filas actualizadas.

SQL> select * from emple;

   EMP_NO APELLIDO   OFICIO          DIR FECHA_ALT     SALARIO   COMISION
DEPT_NO
---------- ---------- ---------- ---------- ---------- ---------- ---------- ----------
7369 SÁNCHEZ    escuchante      7902 17/12/1980     104000              20
7499 ARROYO     escuchante      7698 20/02/1980     208000      39000       30
7521 SALA       escuchante      7698 22/02/1981     162500      65000       30
7566 JIMÉNEZ    escuchante      7839 02/04/1981     386750              20
7654 MARTÍN     escuchante      7698 29/09/1981     162500      182000      30
7698 NEGRO      escuchante      7839 01/05/1981     370500              30
7782 CEREZO     escuchante      7839 09/06/1981     318500              10
7788 GIL        escuchante      7566 09/11/1981     390000              20
7839 REY        escuchante          17/11/1981      650000              10
7844 TOVAR      escuchante      7698 08/09/1981     195000      0       30
7876 ALONSO     escuchante      7788 23/09/1981     143000              20
```

1.Ponerle a gil el oficio de inmob:

```
SQL> update emple
  2  set oficio='inmob'
  3  where apellido='GIL';

1 fila actualizada.
```

2.Cambiar la direccion del cod_centro 22, en la tabla centros, a peñon ,15:
Y el nº de plazas a 295:

```
SQL> update centros
  2  set direccion='c/peñon 15', num_plazas=295
  3  where cod_centro=22;

1 fila actualizada.
```

NOTA: Recordar siempre poner el where, ya que si no lo ponemos cambiara todas las filas de las columnas a las que nos hayamos referido.

6.3.1.UPDATE con SELECT.

Se puede incluir unas subconsultas en una sentencia update. La select puede estar contenida en la where o puede formar parte de set; el formato:

a)

```
 UPDATE nom_tabla
SET columna1=var1,…….,colmnaN=varN
WHERE columna3 =(SELECT…….);
```

b)

```
UPDATE nom_table
SET (columna1,columna2,…..)=(SELECT  col1,col2,….)
WHERE condicion;
```

1.Igualar la direccion y el numero de plazas del cod_centro 10 a los valores del las columnas correspondientes que estan almacenadas para el cod_centro 50.

```
SQL> update centros set(direccion,num_plazas)=(select direccion,num_plazas from centros where
cod_centro=50)
  2    where cod_centro=10;

  1 fila actualizada
```

6.4.Borrado de filas: DELETE.

Para eliminar una o varia fila de una tabla se utiliza orden delete. La clausula where es fundamental para eliminar solo las filas deseadas. Sin la clausula where, delete borraria todas las filas.

El espacio usado por las filas que han sido borradas no se reutiliza a menos que se realice un EXPORT o un IMPORT. Su formato es:

DELETE [FROM] nomb_tabla WHERE condicion;

Ej: borarr el codigo de centro 50 de la tabla centros:

SQL> delete centros where cod_centro=50;

1 fila suprimida.

Borrar todas las filas:

SQL> delete centros;

5 filas suprimidas.

SQL> select * from centros;

ninguna fila seleccionada

NOTA: Hay que poner el where, por que si no se borrara todo.

1.Borrar los departamentos de la tabla depart con menos de 4 empleados.

SQL> delete depart
  2 where dept_no in (select dept_no from emple group by dept_no having count(dept_no)<4);

1 fila suprimida.

6.5.ROLLBACK, COMMIT, AUTOCOMMIT.

Supongamos que queremos borrar una fila de una tabla, y al teclear la orden SQL se nos olvida la cláusula where, automáticamente borramos todas las filas de la tabla. Oracle permite dar marcha atrás mediante la orden ROLLBACK siempre y cuando no hayamos validado los cambios en la base de datos, mediante la orden COMMIT.
Cuando se están realizando transacciones (insertar, actualizar y eliminar)los cambios no se aplicaran hasta que no se haga un COMMIT.
SQL plus, permite validar automáticamente las transacciones sin tener que indicarlo de forma explicita, para ello sirve el parámetro AUTOCOMMIT.

Para ver como se encuentra ese parámetro:
SQL> show autocommit
autocommit OFF

para activarlo:

SQL> set autocommit on
SQL> show autocommit

OFF, es el valor por omision si queremos que las transacciones tengan un valor definitivo sin realizar la validación commit se debe activar el parámetro AUTOCOMMIT

1.Insertar en la tabla depart el numero de departamento 80 que esta en Soria y su actividad es producción y que una vez que se de de alta esa fila quede validada.

```
SQL>  insert into depart (dept_no,dnombre,loc)
  2   values(80,'produccion','SORIA');

1 fila creada.

SQL> DELETE from depart where dept_no=80;

1 fila suprimida.
```

COMMIT implícito:

Hay varias ordenes SQL, que fuerzan a que se ejecute un COMMIT, sin necesidad de indicarlo:

-EXIT.
-CONNECT
-GRANT
-REUD KE
-CREATE TABLE
-CREATE VIEW
-DROP TABLE
-DROP VIEW

→consolida todo lo que hemos estado trabajando hasta ese momento.

ROLLBACK automático:

Si después d haber realizado cambios, reproduce un fallo del sistema (ej. se va la luz y no hemos validado el trabajo, oracle hace un rollback automático sobre cualquier trabajo no validado. Esto significa que tendremos que repetir el trabajo cuando pongamos en marcha la base de datos.

7. CREACIÓN, SUPRESIÓN Y MODIFICACIÓN DE TABLAS Y DE VISTAS EN LA EMPRESA

7.1. INTRODUCCION

Hasta este momento, se han consultado datos de una tabla a través de la sentencia select, se han insertado filas mediante insert, se han actualizado valores de columnas a través de update y se han eliminado filas a través de delete. Todo ello utilizando tablas ya generadas.

Este capítulo utilizará el lenguaje de descripción de datos (DDL) con las órdenes CREATE, DROP y ALTER. Create sirve para crear objetos en la base de datos (tablas, vistas sinónimos, etc.), drop permite eliminar un objeto y la orden alter permite modificar un objeto de la base de datos.

7.2. CREACIÓN DE UNA TABLA

Antes de crear una tabla conviene tener presentes los siguientes aspectos:

1. El nombre de la tabla. Debe ser un nombre que identifique su contenido. Por ejemplo: si llamamos a una tabla ALUMNOS es porque contendrá datos de los alumnos.
2. El nombre de cada columna de la tabla. Ha de ser un nombre descriptivo que permita identificar el contenido. DNI, apellidos, nombre, etc.
3. El tipo de dato y el tamaño que tendrá cada columna.
4. Las columnas que son obligatorias (NOT NULL) los valores por defecto es decir las restricciones de cada columna.

Las denominaciones de la tabla pueden tener de 1 a 30 caracteres.Ha de ser única, no puede ser una palabra reservada a oracle , su primer carácter tierne que ser alfabetico,; los demas pueden ser letras, numeros o subrayado.

Para crear una tabla se utiliza la orden CREATE TABLE,y su principal formato es:

```
CREATE TABLE nom_table
(
insert into tabla2
columna1 tipo_dato [NOT NULL],
columna2 tipo_dato [NOT NULL],
……..etc…….
)
[TABLESPACE nom.tablespace];
```

//no es obligatorio, si no se pone va al //tablespace del administrador(system);
//definir un espacio para tenerlo alli.

Columna 1 y columna 2 son el nombre de las columnas que contiene cada fila.
Tipo_dato indica el tipo de dato(var, varchar2, number,…) de cada columna.
Tablespace, indica en que espacio de tabla se va almacenar la tabla.

Not null, indica que la columna debe contener alguna información, nunca debe ser nula.

*select sysdate from dual-→para ver como esta puesta la fecha definida en oracle.

Crear una tabla:

```
SQL> create table tabla2
  2  (
  3  marca char(10),
  4  tipo varchar(10),
  5  velocidad number(5),
  6  capacidad number(3) not null,
  7  fecha_salida date
  8  )
  9  tablespace users;
```

DROP TABLE nom_tabla;→para borrar tablas.

Oracle soporta los siguientes tipos de datos:

-CHAR: este tipo de datos permite almacenar caracteres de longitud fija, entre 1 y 255(depende de las versiones). La longitud de la cadena se coloca entre paréntesis. Tiene las siguientes características:
-las columnas tienen longitud fija.
-si se introduce una cadena de menor longitud, queda definida, se rellenara con blancos a la derecha, hasta que quede completa.
-si se introduce una cadena mayor a la longitud definida, oracle devolverá un error.

-VARCHAR2(n): almacena cadenas de caracteres de longitud variable. La longitud máxima que se puede definir, es de 2000 caracteres(según versiones).
La longitud de la cadena se define entre paréntesis. Tiene las siguientes características:
-las columnas tienen una longitud variable.
-si se introduce una cadena de menor longitud que la definida, se almacenará con es longitud y no se rellenara con blancos ni con otro carácter cualquiera a la derecha hasta completar la longitud definida.
-si se introduce una cadena de mayor longitud que la definida, oracle devolverá un error.

-NUMBER(p, e): este tipo almacena datos numéricos tanto enteros como decimales. Soporta 38 dígitos de precisión(p). para especificar columnas numéricas se utiliza el identificador NUMBER. Por ejemplo: SAL _ MEDIO number(9,2), de este modo definimos una columna SAL _ MEDIO, como numérica de 9 dígitos, de los cuales 2 son decimales.
Se pueden especifica números enteros usando como formato NUMBER(precisión). Por ejemplo, SAL _ MEDIO number(15).

-LONG: almacena caracteres de longitud variable que contengan hasta 2GB de información .

-RAW: es igual que el tipo de datos varchar2 pero en binario; su longitud maxima dependiendo de versiones, es de 255 caracteres.

-LONGRAW: es igual que el tipo longpero en binario y soporta datos multimedia.

-TIPODATE: se usa para almacenar información de fechas. Para cada tipo date se almacenan:
siglo/año/mes/dia/hora/minutos/segundos --
el formato de la fecha se puede cambiar mediante un ALTERSESION.

Los usuarios pueden consultar las tablas creadas a traves de la vista **USER_TABLES**

SQL> DESC USER_TABLES

SQL> select TABLE_NAME, TABLESPACE_NAME FROM USER_TABLES;

*(NUESTRAS TABLAS)

Existen otras 2 vistas que permiten obrtener informacion de los objetos que son propiedad del usuario: **USER_OBJECTS, USER_CATALOG.:**

SQL> desc user_objects

SQL> DESC USER_CATALOG

SQL> SELECT * FROM USER_CATALOG;

```
TABLE_NAME                     TABLE_TYPE
------------------------------ -----------
ALUM                           TABLE
ALUMNOS                        TABLE
ANTIGUOS                       TABLE
ARTICULOS                      TABLE
ASIGNATURAS                    TABLE
CENTROS                        TABLE
COCHES                         TABLE
DEPART                         TABLE
EMPLE                          TABLE
EMPLE30                        TABLE
FABRICANTES                    TABLE
LEIDOS                         TABLE
LIBRERIA                       TABLE
LIBROS                         TABLE
MISTEXTOS                      TABLE
NACIMIENTOS                    TABLE
NOMBRES                        TABLE
NOTAS                          TABLE
NOTAS_ALUMNOS                  TABLE
NUEVOS                         TABLE
PARALEER                       TABLE

TABLE_NAME                     TABLE_TYPE
------------------------------ -----------
PEDIDOS                        TABLE
PERSONAL                       TABLE
PROFESORES                     TABLE
TIENDAS                        TABLE
VENTAS                         TABLE
```

7.2.1.integridad de datos.

Al almacenar datos en las tablas se pueden ajustar a una serie de restricciones predefinidas, por ejemplo: que una columna no pueda tener valores negativos, que una cadena de caracteres se deba almacenar en mayusculas, que el valor de una columna no pueda ser 0. La integridad hace referencia a que los datos de la bd han de ajustarse a una serie de restricciones antes de almacenar. Asi pues, una restricción de integridad sera una regla que restringe el rango de valores para una o mas columnas de una tabla.

Existe otro tipo de integridad que es la integridad referencial. Lo cual grantiza que los valores de una columna/s de una tabla dependa de los valores de otra columna/s de otra tabla.

La orden CREATE TABLE permitir definir las siguientes restricciones: claves primarias, claves ajenas, obligatoriedad, valores por defecto y verificación de condiciones.

Para definir una restricción en la orden create table se utiliza CONSTRAINT. Este puede restringir una o varias columnas de una misma tabla. Hay dos modos de especificar las restricciones: como parte de una definición de columna (restricción de columna) o al final una vez especificada todas las columnas (restricción de tabla).

El formato de la restricción a nivel de columna seria:

```
CREATE TABLE nombre_tabla
(columna1 tipo_dato
[CONSTRAINT nombre_restriccion]
[NOT NULL]
[UNIQUE]
[PRIMARY KEY]
[DEFAULT valor]
[REFERENCES nombre_tabla][(columna1[,columna2])]
        [ON DELETE CASCADE]
[CHECK condicion],
columna2 tipo_dato
.......);
```

Ejemplos:

```
Create table empleado
(nombre varchar2(25) primary key,
edad, number(2) check (edad between 18 and 35),
cod_provincia number(2) references provincias on delete cascade)
tablespace user
```

Las restricciones de la orden create table que aparecen al final de la definición de las columnas se diferencia del anterior en que se puede hacer referencia a varias columnas en una única restricción

```
CREATE TABLE nombre_tabla
(columna1 tipo_dato,
columna2 tipo_dato,
[CONSTRAINT nombre_restriccion]
{[UNIQUE]|[PRIMARY KEY](columnaN[,columnaN))];
```

```
create table empleado
(
nombre varchar2(25),
edad,number(2),
cod_provincia number(2),
primary key nombre,
check (edad between 18 and 35),
constraint refe referentes provincias on delete cascade
)
tablespace user;
```

Clave primaria. La restriccion primary key

Una clave primaria dentro de una tabla es una columna/s que identifican unívocamente a una fila. Debe ser unica, no nula y obligatoria. Como maximo podemos definir una clave primaria en cada tabla. Esta clave se puede referenciar por una columna o columnas de otra tabla. Cuando se crea una clave primaria, Oracle crea un indice asociado a ella para facilitar el acceso a la tabla

Ejemplos:

Crear una tabla B_pisos con las siguientes columnas: calle con varchar2 de 30, numero tipo number(3), piso number(2), puerta char de 1 , cod_postal de number 5.metros de number de 5. Comentarios de varchar 2 de 60.Cod_zona number de 2. dni varchar2(10).La clave primaria formada por numero, pisos y puerta.

Creada a nivel tabla:

```
SQL>create table b_pisos
(
calle varchar2(30),
numero number(3),
piso number(2),
puerta char(1),
cod_postal number(5),
metros number(5),
comentarios varchar2(60),
cod_zona number(2),
dni varchar2(10),
constraint usuario primary key (numero,piso,puerta)
);
```

Tabla creada.

Creada a nivel de columna:

SQL>Create table b_pisos(calle varchar2(30) primary key**,** numero number(3), piso number(2) puerta char(1),cod_postal number(5),metros number(5),comentarios varchar2(60),cod_zona number(2),dni varchar2(10)

Maestra
provincias
Cod_prov
Nom_prov

Detalle
Personas
DNI
Nombre
Población
Cod_prov
Dirección

Esta es la tabla maestra, y la primary key es el cod_prov por que será la clave ajena en la siguiente tabla:

```
SQL>  create table personas
  2   (
  3   dni number(7) primary key,
  4   nombre varchar2(10),
  5   direccion varchar2(10),
  6   poblacion varchar2(10),
  7   cod_prov number(7),
  8  constraint fkpersonas
  9   foreign key (cod_prov) references provincias on delete cascade
 10  )
 11  tablespace users;
```

En este caso la primary key es el DNI, ya que es el la clave primaria;
después de decir las filas que queremos hay que poner constraint nombre_restriccion (ya que esto sirve para poner las restricciones convenientes a la tabla);
al poner foreign key (nombre_clave_ajena) →esto es para indicarle que esta tabla dependerá de la tabla maestra a través de un campo que ha de ser igual en las 2 tablas, si se intenta meter un dato con un código no existente en la tabla maestra, la aplicaión dará error y no dejará introducir datos.
A continuación, se pone referentes nomb_tabla (aki indicamos de a ke tabla pertenece el foreing key) y por último, on delete cascade, significa que si se borra la clve principal de la tabla maestra, se borraran los datos de la tabla detalle que tengan ese mismo código que anteriormente se ha borrado.}

En la cláusula references indicamos la tabla a la cual remite la clave ajena y a la derecha de foreing key y entre paréntesis, indicamos la columna o columnas que forman parte de la tabla ajena

Si queremos borrar las tablas se ha de comenzar borrando la tabla personas y depués la tabla provincias. Si intentamos borrar la tabla provincias ants que la tabla personas, oracle emitirá n mensaje de error.

Si se desea borrar alguna provincia de la tabla provincias y que las filas correspondientes a la tabla personas con esa provincia sean eliminadas, se añadirá on delete cascade en la opción references.

OBLIGATORIEDAD. la restricción **NOT NULL**

Esta restricción asociada a una columna , significa que no puede tener valores nulos, es decir que ha de tener obligatoriamente un valor. En caso contrario causara una excepción .

CONSTRAINT nomb_restriccion NOT NULL

La restricción not null, se debe poner a nivel columna.
Cuando se viola una columna NOT NULL se ocasiona una excepción y aunque se de nombre a la restrcción, no aparecerán mensajes de restricción si no, simplemente un error.

1.Crear una tabla que se llame eje

```
SQL> create table eje
  2  (
  3  nif varchar2(7) not null,
  4  nombre varchar2(30) constraint nombre_nonulo not null
  5  ,
  6  edad number(2)
  7  )
  8  tablespace users;

insert eje
values (‘452211’);
```

Valores por defecto.

La especificación **DEFAULT**
Al crear una tabla se pueden asignar valores por defecto a las columnas. Si especificamos DEFAULT a una columnas le proporcionará un valor por omisión cuando el valor de la columna no se especifique en la cláusula insert. En la especificación DEFAULT es posible incluir : constantes, funciones SQL y las variables UID y SYSDATE.

Ej:

```
SQL> create table eje1
  2  (
  3  dni varchar2(8) not null,
  4  nombre varchar2(15) not null,
  5  edad number(2),
  6  fecha date default sysdate
  7  )
  8  tablespace users;

SQL> insert into eje1 (dni, nombre, edad)
  2  values ('1234', 'pepa', 21);

1 fila creada.
```

```
SQL> select * from eje1
```

```
  2  ;

DNI      NOMBRE              EDAD FECHA
-------- --------------- ---------- --------
1234     pepa                21 29/11/06
```

Verificación de condiciones.

La restricción **CHECK.**
Muchas columnas de tablas requieren valores limitados dentro del rango o el cumplimiento de ciertas condiciones.
Con la restricción de verificación de restricciones se puede para todas y cada una de las filas de la tabla.
La restricción CHECK actúa como la cláusula WHERE. En una clausula CHECK no se pueden incluir subconsultas ni ls seudo columnas SYSDATE, UID y USERS.

Crear tabla utilizando la restricción check:

```
SQL>  create table eje3
  2   (
  3   dni varchar2(7),
  4   nombre varchar2(15),
  5    usuario number(2),
  6    constraint ch_eje3 check (usuario>20)
  7  );

SQL> insert into eje3
  2  values('1234','pepe',22);

1 fila creada.

SQL> insert into eje3
  2  values('12546','pepe',15);
insert into eje3
*
ERROR en línea 1:
ORA-02290: restricción de control (MONICA.CH_EJE3) violada
```

→da error por que se ha violado el checkeo al meter un usuario <20…

Crear la tabla eje 4 con :
Dni varchar2(10),
Nombre varchar2(20),
Edad number(2),
Curso number(4),

Con las siguientes restricciones,:
El dni no puede ser nulo, la clave primaria es el dni, el nombre no puede ser nulo, la edad debe ser a 20 años,el nombre ha de estar en mayusculas y el curso solo puede ser 1,2 o 3.

```
SQL>    create table eje4
```

```
  2    (
  3     dni varchar2(10) not null,
  4    nombre varchar2(20) not null,
  5    edad number(2),
  6    curso number(4),
  7     constraint pk_eje4 primary key (dni),
  8     constraint ch_edeje4 check (edad<20),
  9     constraint up_nomeje4 check (nombre=upper(nombre)),
 10    constraint ch_cursoeje4 check (curso between 1 and 4)
 11   )
 12   tablespace users;
```

Tabla creada.

AHORA LOS INSERT CON ERRORES PARA SABER POR QUE:

```
SQL> insert into eje4
  2  values('1234','monica',15,2);
insert into eje4
*
ERROR en línea 1:
ORA-02290: restricción de control (MONICA.UP_NOMEJE4) violada


SQL> insert into eje4
  2  values('1234','MONICA',15,2);
```

1 fila creada.

```
SQL> insert into eje4
  2  values ('235','PEPE',21,5);
insert into eje4
*
ERROR en línea 1:
ORA-02290: restricción de control (MONICA.CH_CURSOEJE4) violada
```

La restricción **UNIQUE,** evita valores repetidos en la misma columna. Es similar a la primary key salvo que se pueden tener varias columnas UNIQUE en la misma tabla.
Al igual que en primary key cuando se define una restricción UNIQUE se crea automáticamente un índice.
Ej->

```
SQL> create table ejem1
  2   (
  3   dni varchar2(10) primary key,
  4   nombre varchar2(15) unique,
  5   edad number(2)
  6   )
  7   tablespace users;
```

Tabla creada.

```
SQL> desc ejem1
```

```
 Nombre                                    ¿Nulo?   Tipo
 ----------------------------------------- -------- ----------------------------
 DNI                                       NOT NULL VARCHAR2(10)
 NOMBRE                                             VARCHAR2(15)
 EDAD                                               NUMBER(2)

SQL> insert into ejem1
  2  values ('1234','jose',15);

1 fila creada.

SQL> insert into ejem1
  2  values ('1234','jose',15);

1 fila creada.

SQL> insert into ejem1
  2  values ('234567','jose',13);
insert into ejem1
*
ERROR en línea 1:
ORA-00001: restricción única (MONICA.SYS_C006255) violada
```

*en este caso el unique tiene que ser a los 2 valores a la vez….ya que si uno se pone igual y el otro unique no..se creara la fila, sin embargo si los 2 uniques coinciden dara error.

Ejemplo con restricciones
Tabla maestra:

```
SQL> run
  1   create table ejem8
  2   (
  3   dni number(7),
  4   nombre varchar2(15) not null,
  5   grupo number(2) default 3,
  6   sociedad varchar2(10),
  7   medico number(3) not null,
  8  constraint  pk_8dni primary key (dni),
  9  constraint ch_8gru check (grupo between 2 and 6),
 10  constraint un_8medico unique (medico),
 11  constraint fk_8soci unique (sociedad)
 12  )
 13* tablespace users
```

tabla secundaria:

```
SQL> create table ejem8b
  2  (
  3  dni number(7),
  4  fecha_alta date not null,
  5  pago number(3,2),
  6  constraint fk_8b_dni foreign key (dni)references ejem8 on delete cascade
  7  )
  8  tablespace users;
```

→estas dos tablas se unirán a través de foreign key en la secundaria y primary key en la maestra.

7.3.Supresión de tablas.

La orden sql DROP TABLE suprime una tabla de la base de datos. Cada usuario puede borrar sus propias tablas. Solo el administrador de la base de datos o algún usuario con privilegios suficientes, puede borrar las tablas de otro usuario. Al suprimir una tabla también se suprimen los índices y los privilegios asociados a ella. Las vistas y los sinónimos creados a partir de esta tabla dejan de funcionar peor siguen existiendo en la base de datos por lo que habría que eliminarlos.

El formato es:
DROP TABLE nombre_tab [cascade constraints]

7.3.1.Orden truncate

la orden truncate permite suprimir todas las filas de una tabla y liberar el espacio ocupado para otros usos pero no depsaparece la definición de la tabla. Es una orden del lenguaje DDL que no genera retroceso (ROLLBACK).
*la orden delete no libera espacio.

Su formato:
TRUNCATE TABLE nom_tabla;

7.4.Modificación de tablas.

Se modifican las tablas de 2 formas:
Cambiando la definición de la columna (modify) o añadiendo una columna a una tabla existente (add). La orden con la que se modifica una tabla es ALTER TABLE y el formato es el siguiente:

ALTER TABLE nom_tabla
{
[ADD (columna [,columna]…)]
[MODIFY (columna [,columna]…)]
[ADD CONSTRAINT restricción]
[DROP CONSTRAINT resticcion]
};

ADD añade una columna o mas al final de una tabla.
MODIFY modifica una o mas columnas existentes en la tabla.
ADD CONSTRAINT añade una restricción a la definición de la tabla.
DROP CONSTRAINT elimina una restricción de la tabla.

1.Ejemplo:

```
SQL> create table prueba
  2  (
  3  dni varchar2(10) not null,
  4  nombre varchar2(30) not null,
  5  edad number(2),
  6  curso number(2)
  7  )
  8  tablespace users;
```

A esta tabla vamos añadirle 2 columnas:
sexo number(2)
importe number(4)

```
SQL> alter table prueba
  2  add (sexo number(2), importe number(4))
  3  ;
```

Modificar la columna sexo, añadirle not null:

```
SQL> alter table prueba
  2    modify ( sexo number(2) not null);
```

Modificar y quitar el not null a sexo:

```
SQL> alter table prueba
  2    modify (sexo number(2) null);
```

cuando hay que añadir una columnas a una tabla se deben tener en cuenta los siguientes factores:
-si la columnas no esta definida como not null se le puede añadir en cualquier momento.
-Si la columna esta definida como not null será necesario dar valor a esa columna para cada una de las filas y a continuación modificar la columna a not null.

Modificar la columnas nombre dandole varchar(29)

```
SQL> alter table prueba
  4    modify (nombre varchar2(29));
```

Al modificar una columnas de una tabla hay que tener en cuenta:

-Se puede aumentar la longitud de la columna en cualquier momento.

-Es posible aumentar o disminuir el numero de posiciones decimales en una columnas tipo number.

-Si la columna es null, en todas las filas de la tabla se puede disminuir la longitud y el tipo de datos.

-La opción modify…..not null solo sera posible cuando la tabla no contenga ninguna fila con valor nulo en la columna que se modifique.

* DROP COLUMN nomb_columna

```
SQL> alter table prueba
  2  drop column sexo;
```

→minimo versión 9.

7.4.Adicción de restricciones

Con la orden alter table se pueden añadir restricciones a una tabla:

ALTER TABLE nom_tabla
ADD CONSTRAINT restricción;

```
SQL> alter table emple
  2  add constraint u_empape unique(apellido);
```

Para borrar la constraint.

```
SQL> alter table emple
  2  drop constraint u_empape;
```

Tabla modificada.

```
SQL>
SQL>  select constraint_name,constraint_type,table_name from user_constraints
  2   where table_name='EMPLE';

CONSTRAINT_NAME              C TABLE_NAME
------------------------------ - ------------------------------
SYS_C004730                  C EMPLE
SYS_C004731                  C EMPLE
```

Se puede activar o desactivar las restricciones con las sentencias **ENABLE** y **DISABLE** :

```
SQL> ALTER TABLE emple
  2  disable constraint u_empape;
```

Para verlo:

```
SQL>  select constraint_name,constraint_type,table_name,status from user_constraints
  2  where table_name='EMPLE';

CONSTRAINT_NAME              C TABLE_NAME                 STATUS
```

```
------------------------------ - ------------------------------ --------
SYS_C004730                    C EMPLE                        ENABLED
SYS_C004731                    C EMPLE                        ENABLED
U_EMPAPE                       U EMPLE                        DISABLED
```

Para activarla:

```
SQL> ALTER TABLE emple
  2  enable constraint u_empape;
```

Tabla modificada.

```
SQL>  select constraint_name,constraint_type,table_name,status from user_constraints
  2  where table_name='EMPLE';
```

para verlo:

```
CONSTRAINT_NAME                C TABLE_NAME                   STATUS
------------------------------ - ------------------------------ --------
SYS_C004730                    C EMPLE                        ENABLED
SYS_C004731                    C EMPLE                        ENABLED
U_EMPAPE                       U EMPLE                        ENABLED
```

7.5. CREACIÓN Y USO DE VISTAS.

A veces para obtener datos de varias tablas se ha de construir una sentencia select completa y además si en otro momento necesitásemos realizar la misma consulta deberíamos construir de nuevo la sentencia select. Sería muy cómodo poder obtener los datos de cualquier consulta incluso las complejas con una misma sentencia select.

Para solucionar este problema se utilizan las vistas. Una vista es una tabla lógica que permite acceder a la información de una o varias tablas. No contiene información por si mismo, sino que su información esta basada en la que contienen otras tablas llamadas tablas base. Simplemente es una sentencia select. Si se suprime la tabla la vista asociada a ella se invalida. Las vistas tienen la misma estructura que una tabla (filas y columnas) y se trata de forma semejante. Su formato es:

CREATE [OR REPLACE] VIEW Nombre_vista
[(columna [,columna] ...)]
AS consulta.

- Nombre_vista: Es el nombre que le damos a la vista.
- [(columna [, columna]...)]: Son los nombres de las columnas que va a contener la vista. Si se omiten se asumen los nombre de las columnas
- AS consulta: Determina las columnas y las tablas que aparecerán en la vista.
- OR REPLACE: Crea de nuevo la vista si ya existía.

1.Crear una vista de Dep.30 con el oficio, el salario y el apellido de la tabla emple del departamento 30.

```
SQL> create view dept30 as (select apellido, oficio, salario from emple where dept_no=30);
```

```
SQL> desc dept30
 Nombre                                    ¿Nulo?   Tipo
 ----------------------------------------- -------- ------------
 APELLIDO                                           VARCHAR2(10)
 OFICIO                                             VARCHAR2(10)
 SALARIO                                            NUMBER(10)
```

SQL> create or replace view dept30 (apel, ofis, sal) as (select apellido, oficio, salario from emple where dept_no=30);

```
SQL> desc dept30
 Nombre                                    ¿Nulo?   Tipo
 ----------------------------------------- -------- ------------
 APEL                                               VARCHAR2(10)
 OFIS                                               VARCHAR2(10)
 SAL                                                NUMBER(10)
```

SQL> desc user_views
SQL> select view_name from user_views;

*** Para ver las vistas que tenemos creadas. ***

7.5.1. Borrado de vistas.

Es posible borrar las vistas a través de la orden drop view; su formato es:

DROP VIEW Nombre_vista

SQL> drop view prueba7;

7.5.2. Operaciones sobre vistas.

Las operaciones que se pueden realizar sobre vistas son las mismas que se llevan a cabo sobre las tablas: select, insert, update y delete.

- Consultas: Se utiliza el mismo procedimiento que para las tablas:
Select columna1, columna2… |* from Nombre_vista where condición.

- Actualización: Si una vista está basada en una sola tabla se pueden modificar las filas de la vista. En realidad lo que se hace es actualizar las filas de la tabla sobre la que está definida.

1. Modificar la vista dept30 modificando el apellido Martín almacenándolo en minúsculas y cambiando el salario a 200.000.

SQL> update dept30 set apel='martín', sal=200000 where apel='MARTÍN';

Existen una serie de restricciones a considerar en la actualización y la inserción de filas en una tabla a través de la vista:

- Borrado de filas a través de una vista: Para borrar filas de una tabla a través de una vista ésta se debe crear:
  - Con filas de una sola tabla.
  - Sin utilizar la cláusula group by ni distinct.
  - Sin usar funciones de grupo o referencias a pseudo columnas.

- Actualización de vistas a través de una vista: Para actualizar filas en una tabla a través de una vista esta debe estar definida según las restricciones anteriores y además ninguna de las columnas que se van a actualizar deberá haber sido definida como una expresión.

- Inserción de filas a través de una vista: se habrán de tener en cuenta todas las restricciones anteriores y además todas las columnas obligatorias de la tabla asociada deben estar presentes en la vista.

2. Creamos la vista vdep a partir de la tabla depart

SQL> create view vdep as (select * from depart);

- Insertamos en la vista una fila que tenga valor 55 en dept_no e informática en d_nombre.

SQL> insert into vdep (dept_no, dnombre) values (55, 'INFORMÁTICA');

- Borramos la fila insertada a través de la vista.

SQL> delete from vdep where dept_no=55;

- Vistas definidas sobre más de una tabla:

3.A partir de la tabla emple y depart creamos una vista que contenga las columnas emp_no, apellido, dept_no y d-nombre.

SQL> create view empdep as (select depart.dept_no, apellido, emp_no, dnombre from depart, emple where emple.dept_no=depart.dept_no);

7.6 CREACIÓN DE SINÓNIMOS.

Cuando se tiene acceso a las tablas de otro usuario y se desea consultarlas es preciso teclear antes del nombre de la tabla el nombre de su propietario. Mediante el uso de sinónimos podemos crearnos uno para referirnos a la tabla de otro usuario sin incluir su nombre.
Un sinónimo es un nuevo nombre que se puede dar a una lista o a una tabla. Con los sinónimos se pueden utilizar 2 nombres diferentes para referirse a un mismo objeto. El formato es el siguiente:

CREATE [PUBLIC] SYNONYM Nombre_sinónimo FOR Nombre_tabla

- PUBLIC: hace que el sinónimo esté disponible para todos los usuarios.

SQL> select synonym_name, table_name from user_synonyms;

SYNONYM_NAME TABLE_NAME

PGV	PABLO27

7.6.1 Borrado de sinónimos.

Del mismo modo que se crea un sinónimo se pueden borrar con la orden DROP SYNONYM. Su formato es:

DROP [PUBLIC] SYNONYM Nombre_sinónimo

SQL> drop synonym PGV;

7.7 CAMBIO DE NOMBRE.

RENAME es una orden de SQL que cambia el nombre de una tabla, vista o sinónimo. El nuevo nombre no puede ser una palabra reservada ni el nombre de un objeto que tenga creado el usuario. El formato es:

- RENAME Nombre_anterior Nombre_nuevo

Las restricciones de integridad, los índices y los permisos dados al objeto se transfieren automáticamente al nuevo objeto.

No se puede usar esta orden para renombrar sinónimos públicos, ni para renombrar columnas de tablas.

1.Crear un sinónimo de la tabla alumnos.

SQL> create synonym alumn for alumnos;

Sinónimo creado.

SQL> rename alumnos to talumnos;

SQL> select * from alumnos;
ERROR en línea 1:
ORA-00942: la tabla o vista no existe

SQL> select * from alumn;
ERROR en línea 1:
ORA-00980: la traducción de sinónimos ya no es válida

2. Renombrar una tabla y ver si siguen las restricciones.

SQL> rename eje4 to teje4;

Tabla cambiada de nombre.

SQL> select table_name, constraint_name from user_constraints;

8.Generación de informes en SQL plus.

8.1.Introducción.

El uso mas común de SQL plus es realizar consultas y obtener informes. Los informes se pueden escribir directamente desde el indicador SQL plus (de este modo al ejecutar una consulta se produce automáticamente una salida formateada). El inconveniente radica en que si queremos repetir el mismo informe se deberán teclear todas las ordenes de nuevo. La solución para ello es guardar todas las órdenes en un archivo y ejecutarlo cada vez que se necesite.

8.2.Uso del editor para crear archivos SQL.
(Extendido en tema 3.2)

Para guardar las ordenes en un archivo hay que teclear e el indicador de SQL la orden EDIT seguida del nombre del archivo en el que se van a guardar esas acciones.
Con EDIT se invoca al editor y por defecto toma el de bloc de notas. Si el editor avisa de que no encuentra el fichero pregunta si se desea crear.
Salvo que se indique una ruta en concreto para le archivo creado este se almacenara por defecto en el directorio bin de oracle.

8.3.Ordenes para generar informes.

8.3.1.Orden REMARK (REM).

Permite añadir comentarios y se puede colocar en cualquier parte del listado

8.3.2.Configuración de variables del entorno SQL plus con orden SET.

Se pueden consultar y cambiar el valor de las variables. Se pueden dar valor a estas variables con la orden SET y visualizar su valor utilizando SHOW.

SET variable valor

Ej:

```
SQL> set linesize 100
SQL> set pagesize 24
SQL> show linesize
linesize 100
SQL> show pagesize
pagesize 24

SQL> show headsep
headsep "|" (hex 7c)
```

SET HEADSEP |
El carácter que se coloca a la derecha de la orden identifica el carácter que divide un titulo superior o inferior o una cabecera o una columna en 2 o más líneas.

Ejemplo:
SQL> ttitle 'listado General de trabajadores | ======================'
SQL> select * from emple;

Lun Ene 08 página 1
listado General de trabajadores
==========

EMP_NO	APELLIDO	OFICIO	DIR	FECHA_AL	SALARIO	COMISION	DEPT_NO
7369	SÁNCHEZ	EMPLEADO	7902	17/12/80	104000		20
7499	ARROYO	VENDEDOR	7698	20/02/80	208000	39000	30
7521	SALA	VENDEDOR	7698	22/02/81	162500	65000	30
7566	JIMÉNEZ	DIRECTOR	7839	02/04/81	386750		20
7654	MARTÍN	VENDEDOR	7698	29/09/81	162500	182000	30
7698	NEGRO	DIRECTOR	7839	01/05/81	370500		30
7782	CEREZO	DIRECTOR	7839	09/06/81	318500		10
7788	GIL	ANALISTA	7566	09/11/81	390000		20
7839	REY	PRESIDENTE		17/11/81	650000		10
7844	TOVAR	VENDEDOR	7698	08/09/81	195000	0	30
7876	ALONSO	EMPLEADO	7788	23/09/81	143000		20
7900	JIMENO	EMPLEADO	7698	03/12/81	123500		30
7902	FERNÁNDEZ	ANALISTA	7566	03/12/81	390000		20
7934	MUÑOZ	EMPLEADO	7782	23/01/82	169000		10

SET LINESIZE

Con esta orden se define el máximo de caracteres que puede aparecer en una línea
Set linesize 100

SET PAGESIZE

Con esta orden se define el número de líneas por página incluyendo TTITLE y BTITLE.

SET NEWPAGE

Esta orden imprime líneas en blanco antes de la línea superior de cada pagina del informe. Esta línea contiene la fecha y el numero de página.

SET TERMOUT OFF y SET TERMOUT ON

Cuando se ejecuta un listado START este aparece por la pantalla. Mediante estas órdenes se puede evitar que aparezca sin olvidar que para no perder la información se debe incluir SPOOL.
Con SET TERMOUT OFF se evita que los datos aparezcan en pantalla.
Con SET TERMOUT ON los datos aparecen por pantalla.

8.3.3. Ordenes TTITLE y BTITLE (encabezados y pies de página).

Con estas órdenes se ponen títulos en la parte superior e inferior de cada página del informe.
Ej:

Queremos poner el titulo “Listado general de empleados durante el mes de enero”, subrayado con *.
Y en la parte inferior, a la izquierda, solo enero del 2007.

```
SQL> ttitle 'Listado general de trabajadores  durante el mes de Enero _
**********************************************'
SQL> btitle left 'enero de 2007'
SQL> select * from emple;

Lun Ene 08                                                    página    1
                Listado general de trabajadores  durante el mes de Enero
                   **********************************************

   EMP_NO APELLIDO   OFICIO          DIR FECHA_AL    SALARIO   COMISION
DEPT_NO
---------- ---------- ---------- ---------- -------- ---------- ---------- ----------
     7369 SÁNCHEZ    EMPLEADO        7902 17/12/80     104000                  20
     7499 ARROYO     VENDEDOR        7698 20/02/80     208000      39000       30
     7521 SALA       VENDEDOR        7698 22/02/81     162500      65000       30
     7566 JIMÉNEZ    DIRECTOR        7839 02/04/81     386750                  20
     7654 MARTÍN     VENDEDOR        7698 29/09/81     162500     182000       30
     7698 NEGRO      DIRECTOR        7839 01/05/81     370500                  30
     7782 CEREZO     DIRECTOR        7839 09/06/81     318500                  10
     7788 GIL        ANALISTA        7566 09/11/81     390000                  20
     7839 REY        PRESIDENTE           17/11/81     650000                  10
     7844 TOVAR      VENDEDOR        7698 08/09/81     195000          0       30
     7876 ALONSO     EMPLEADO        7788 23/09/81     143000                  20
     7900 JIMENO     EMPLEADO        7698 03/12/81     123500                  30
     7902 FERNÁNDEZ  ANALISTA        7566 03/12/81     390000                  20
     7934 MUÑOZ      EMPLEADO        7782 23/01/82     169000                  10


Enero de 2007
```

Se pueden utilizar 3 parámetro para indicar en que parte del informe deben aparecer los textos.
LEFT
CENTER
RIGHT

```
ttitle off
btitle off
```

Esto se utiliza para borrar los títulos y pies de pagina.
Se puede incluir una serie de variables con información del sistema, tanto en cabeceras (ttitle) como en pie de página (btitle) :

SQL.LNO=nº línea actual
SQL.PNO =nº pagina actual
SQL.RELEASE =versión actual de oracle
SQL.USER= nombre de usuario.

El guión (-) significa que se continua en la linea siguiente, es decir que lo que hagamos a continuación, tiene que ver con la misma linea.
Ejemplo:

Ttitle left ‘apellido de los empleados’ center ‘usuario: ‘ –
SQL.USER skip 1 left ‘pagina: ‘ format999 SQL.PNO

8.3.4.ORDEN COLUMN. Formato columnas .

Esta orden formatea el contenido de las columnas y sus cabeceras. Se puede usar directamente desde el indicador SQL plus en una sentencia select.
Ej.

COLUMN salario format 999,999,999.99 heading ‘Salario del empleado’;

Si se desea que desaparzca el formato definido se utilizará la orden clear columns
COLUMN siempre trabaja asi;

COLUMN nom_colum HEADING ‘cabecera|*’;
COLUMN nom_columna FORMAT A15
COLUMN nom_colum TRUNCATE;

Para definir la cabecera de la columna se emplea la orden heading. El texto que aparece a la derecha es la nueva cabecera que en este caso tiene el separador (|) indicando que la cabecera debe aparecer en 2 líneas.
En la segunda fila se define el formato para el contenido de la columna mediante la cláusula format. La A que aparece a la derecha de format indica que se da formato a una columna de tipo alfanumérico y el número que sigue a continuación representa la cantidad en caracteres.
Todas las cláusulas de formato para una columna se pueden incluir en una única orden.
Si una columna esta definida en la tabla con 18 caracteres tiene mas de 15 aparecerá en 2 líneas. Para evitar esto se recurre a la opción truncate, que trunca el contenido de la columna y solo se considera 15 caracteres.

SQL> column apellido format a5 truncate
SQL> select apellido from emple;

APELL

SÁNCH
ARROY
SALA
JIMÉN
MARTÍ
NEGRO
CEREZ
GIL
REY
TOVAR
ALONS
JIMEN
FERNÁ
MUÑOZ

Se puede dar formato a columnas que sean fruto de alguna expresión siempre y cuando se de un alias de columnas.

Ej:

```
SQL> column total format 9999,9999.9999
SQL> select apellido,salario,salario*12 "total" from emple;

APELLIDO      SALARIO          total
---------- ---------- ---------------
SÁNCHEZ       104000   124,8000.0000
ARROYO        208000   249,6000.0000
SALA          162500   195,0000.0000
JIMÉNEZ       386750   464,1000.0000
MARTÍN        162500   195,0000.0000
NEGRO         370500   444,6000.0000
CEREZO        318500   382,2000.0000
GIL           390000   468,0000.0000
REY           650000   780,0000.0000
TOVAR         195000   234,0000.0000
ALONSO        143000   171,6000.0000
JIMENO        123500   148,2000.0000
FERNÁNDEZ     390000   468,0000.0000
MUÑOZ         169000   202,8000.0000
```

*aquí estamos formateando una columna que no existe todavía, entonces para que esto funcione, hay que poner un alias a la columna que queremos manejar.

8.3.5.Orden BREAK ON. Ruptura de control.

Esta instrucción produce una ruptura de control por la columna que se desee. Una ruptura de control que permite obtener un resultado o varios sobre algo que tiene en común varias filas.
La orden BREAK ON necesita la claúsula ORDER BY de la sentencia SELECT para que la salida del listado o informe sea la deseada.
Ej:
BREAK ON articulo SKIP 2
Ejemplo:

```
SQL> break on dept_no skip 2
SQL> select dept_no,oficio from emple
  2  order by dept_no;

   DEPT_NO OFICIO
---------- ----------
        10 DIRECTOR
           PRESIDENTE
           EMPLEADO


        20 EMPLEADO
           EMPLEADO
           ANALISTA
           ANALISTA
           DIRECTOR
```

30 VENDEDOR
DIRECTOR
VENDEDOR

Para limpiar estas rupturas de control la instrucción es:
CLEAR BREAKS

Si queremos ver el numero que encontramos en el informe a la izquierda tendríamos que poner DUPLICATE.
Ej: break on dept_no duplicate

Solo puede haber una orden BREAK activa por lo que si se especifican varias rupturas han de especificarse todas las columnas en una sola orden.

8.3.6. Orden COMPUTE. Cálculos.

Con esta orden se hacen cálculos con grupos de filas seleccionadas.
El formato es:

COMPUTE {SUM,AVG,COUNT,MAX,MIN}
OF {expres | column}
ON {expre | row | page | report}
Los totales que calcula COMPUTE son los especificados en el BREAK ON. Esto significa que no puede usarse el COMPUTE sin BREAK.
Ej:
Break on columna1 skip 3
Compute sum of sal columna1
Compute sum of com on columna1

BREAK REPORT→ esta orden nos da el importe total de la columna que deseemos.

Ej:
SQL> break on report
SQL> compute sum of salario on report
SQL> select oficio,salario from emple;

OFICIO SALARIO
---------- ----------
EMPLEADO 104000
VENDEDOR 208000
VENDEDOR 162500
DIRECTOR 386750
VENDEDOR 162500
DIRECTOR 370500
DIRECTOR 318500

```
ANALISTA       390000
PRESIDENTE     650000
VENDEDOR       195000
EMPLEADO       143000
EMPLEADO       123500
ANALISTA       390000
EMPLEADO       169000
          ----------
sum           3773250
```

BREAK ON multiples:

Se pueden poner 2 opciones en el break:

```
SQL>  break on dept_no on report skip 2
SQL> compute sum of salario on dept_no
SQL> compute sum of salario on report
SQL>  select dept_no,salario from emple
  2  order by dept_no;

   DEPT_NO    SALARIO
---------- ----------
        10     318500
               650000
               169000
********** ----------
sum             1137500
        20     104000
               143000
               390000
               390000
               386750
********** ----------
sum             1413750
        30     208000
               370500
               162500
               123500
               195000
               162500
********** ----------
sum             1222000
          ----------

   DEPT_NO    SALARIO
---------- ----------
sum           3773250
```

Para ver lo que tenemos activo en estas últimas ordenes, simplemente poner la orden:

```
SQL> break
break en report ignorar 2 nodup
```

```
        en dept_no nodup

SQL> compute
COMPUTE sum LABEL 'sum' OF salario ON dept_no
COMPUTE sum LABEL 'sum' OF salario ON report
```

Si queremos cambiar el titulo de la función tendremos que usar la orden LABEL:

```
SQL> break on dept_no
SQL> compute sum label 'suma total' of salario on dept_no
SQL> select dept_no, salario from emple
  2  order by dept_no;

   DEPT_NO    SALARIO
---------- ----------
        10     318500
               650000
               169000
********** ----------
suma total    1137500
        20     104000
               143000
               390000
               390000
               386750
********** ----------
suma total    1413750
        30     208000
               370500
               162500
               123500
               195000
               162500
********** ----------
suma total    1222000
```

8.3.7. Orden SPOOL. Generador de informes.

Esta orden permite guardar en un archivo toda la salida SQL plus que se vaya produciendo sobre la pantalla hasta que se indique a SQL plus que pare.
Esta acción se realiza mediante la orden SPOOL OFF.
Por eso en el listado se utilizan la orden SPOLL para guardar toda la salida de un informe y poder utilizarlo en el momento que se quiera y cuantas veces se necesite. El listado de salida queda almacenado en el fichero que se indica a continuación de la orden SPOOL.

9.Administración de Oracle.

una SGA y una serie de procesos en background
un sgbd relacional y una instancia oracle.

Una base de datos oracle es un conjunto de información tratado como una unidad y una instancia oracle esta formada por un conjunto de procesos y estructura de memoria compartida que permiten definir almacenar y manipular la base de datos asi como controlar el acceso la concurrencia y el uso de la información garantizando la seguridad de la base de dato.

9.2. Herramienta de Oracle.

Oracle SERVER.- Es la motor de la base de dato. Permite almacenar grandes cantidades de datos proporcionando a los usuario un rápido acceso. Los datos se pueden compartir con varias aplicaciones y varios usuario. Oracle server admite las siguientes configuraciones:

1. basada en anfitrión: los usuarios se conectan al mismo ordenador en que se encuentra la base de datos.
2. Cliente/Servidor: los usuarios acceden a la base de datos desde su ordenador a través de una red. La base de datos esta en otro ordenador (servidor).
3. Proceso Distribuido: los usuarios acceden a la información que esta almacenada en mas de un ordenador. No tiene por que conocer la unidad física donde trabajan.

Oracle Office.- Se trata de un grupo de

Oracle Loador.- Permite introducir datos en una base de datos de forma rapida.

Oracle Designer 2000.- Es una herramienta que diseña programa e implementa y mantiene sistema.

Oracle Developer 2000.- Esta formado por los siguientes productos SQLPLUS, Lenguaje de consulta de la base de datos.

Oracle Forum.- Diseña pantallas para introducir datos y consultas.

Oracle report: Es un generador de informes.

Personal oracle: es la base de datos oracle para ordenadores personales en entorno DOS y Windows. Es una implementación de oracle server.

9.3.estructura de la base de datos.

Un SGBD relacional debe ser capaz de manejar la información estructurada en 3 niveles.
En la arquitectura ANSI los datos están estructurado sen 3 niveles, conceptual, externo e interno.
Los 2 primeros implican que el sistema relacional debe ser capaz de manejar el modelo relacional.
El tercer nivel (interno) obliga a que el sistema maneje las estructuras adecuadas para el almacenamiento real de los datos con el fin de conseguir un rendimiento adecuado en las operaciones de manipulación de datos.

En los manuales de oracle se dice que una base de datos tiene una estructura física y una estructura lógica. La física esta constituida por los ficheros del sistema operativo que dan soporte a los datos. Los TABLESPACE y los OBJETOS son los elementos que configuran la estructura lógica.

Estructura de base de datos oracle		Niveles ANSI
Estructura lógica	Vistas	Externo
	Tablas	Conceptual
	Tablespaces	
	Indices	
	Clusters	Interno
Estructura física	Ficheros datos	
	Fichero redo log	
	Ficheros de control	

Nota: la estructura lógica puede inducir a error, ya que hay elementos que corresponden tanto a nivel externo y conceptual como a nivel interno.

Los SGBD se apoyan en el sistema operativo para gestionar el acceso a los datos.
En oralce la estructura física se gestiona a través de las rutinas del sistema operativo y la estructura lógica es gestionada directamente pro el software de oracle. El SGBD soporta le lenguaje SQL tanto auto contenido como huesped.el usuario puede definir y manipular datos en los niveles externos y conceptual mediante el sql estándar.

9.4.Arquitectura de Oracle.

Partiendo de que una base de datos es un conjunto de archivos de datos y el software que los maneja los componentes de una base de datos oracle son:
- Archivos de datos (database files)
- Archivos de diario o transacciones (los files) (redo log file).
- Archivos de control (control files).

9.4.1.Componentes de la base de datos.

Archivo de datos: Toda la información de la base de datos se almacena fisicamente sobre dispositivos de acceso directo (discos,etc..), en archivos de datos:
Datos de usuario
Datos del sistema

El espacio de almacenamiento formado por los ficheros se agrupan en estructuras
Podemos tener un tablespace para ventas, otro para compras. Cada tablespace puede constatar de uno o varios archivos, pero un archivo de datos solo puede permanecer a un único tablespace.

Registros de rehacer o redolog: Se trata de archivos en los que oracle registra todas las modificaciones y transacciones (insert, delete y update) que se producen en la base de datos.

Esto permite recuperar la base de datasen caso de que se produzca un problema.
Un fichero redolog contiene:
Identificación de la transacción, dirección del bloque, número de fila, número de columna, valor anterior y valor nuevo.

Archivos de control: Un archivo de control contiene información sobre lso archivos asociados a la base de datos. Tienen todas las modificaciones importantes que se hagan en la estructura de la base de datos. Estos archivos de control, mantienen la integridad del archivo de datos por lo que es recomendable tener por lo menos 2 en previsión de que alguno se borre o deteriore. Los archivos de control tiene entre otras la siguiente información:
1- información sobre arranque o parada.
2-nombres de los archivos de la base de datos y de redolog.
3- Información sobre check points.
6-Fecha de la actualización y nombre de la fecha de datos.
5-estado on-line y off-line de los archivos
etc.….

9.4.2.Estructura de la memoria.

Al arrancar un SGBD Oracle y en cualquier máquina se asignan un área de memoria llamada SGA (System Global Area) y se inician una serie de procesos background.

El conjunto de procesos background y la SGA constituyen lo que se llama una Instancia de Oracle. La memoria y los procesos de una instancia gestionan los datos y facilitan el acceso de los usuarios a los mismos.

Es posible el número de varias instancias ejecutándose en una misma maquina. En este caso cada instancia accede a su propia base de datos física y cada una de ellas utilizará un conjunto distinto de ficheros de datos y de redo log.

Además de los procesos background existen otros denominados procesos servidores. Oracle crea procesos servidores para atender los procesos de usuario (personas que ejecutan el código de los programas de aplicación.
Una operación que desee realizar un acceso a la base de datos (proceso de ususario) será siempre gestionada por un proceso servidor.
Cuando el SGBD arranca un proceso servidor, le asigna un buffer en memoria denominado PGA (Program Gloval Area) para contener los datos y la información de control.
Cuando el proceso de usuario y el proceso servidor residen en distintas máquinas de una red se necesita la ejecución de SQL*NET tanto en la maquina servidor como en la máquina cliente.
SQL*NET es el software de oracle que actúa como interfaz como el protocolo usado por la red.

Área global de programa (PGA)

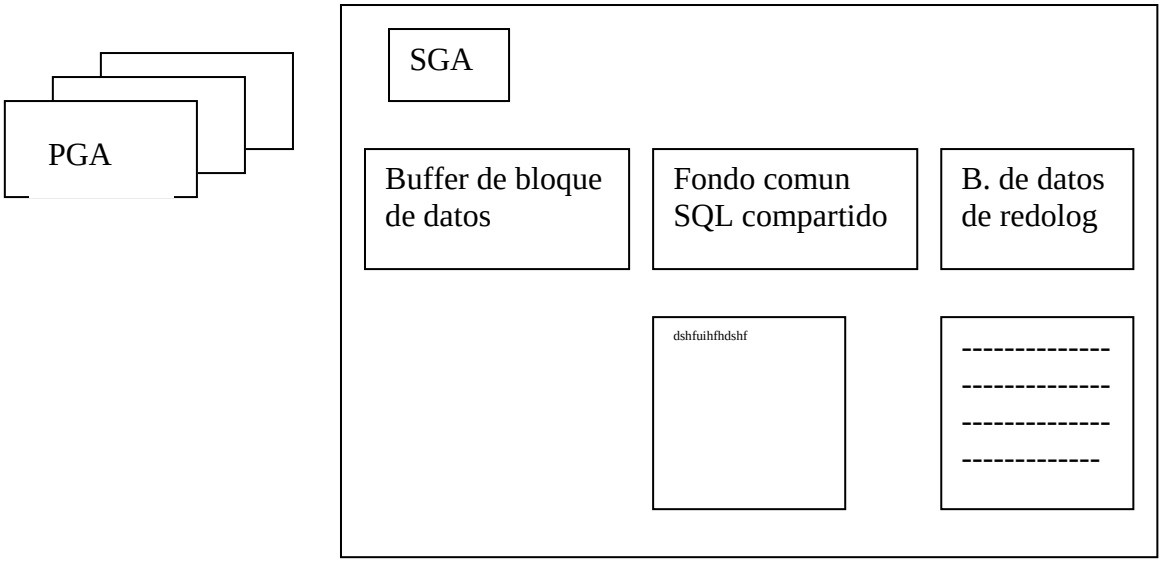
Es la zona de memoria utilizada pro un solo proceso de usuario de oracle y contiene datos e información del proceso.

Área global del sistema (SGA)

Permite la comunicación entre los diversos procesos entre el cliente y el servidor. También mantiene la información mas consultada de la base de datos y se compone de las siguientes formas:
1- Buffer de bloque de datos. Aquí oracle almacena los bloques de datos utilizados. Se puede decir que los usuarios acceden a los datos en esta zona de memoria.

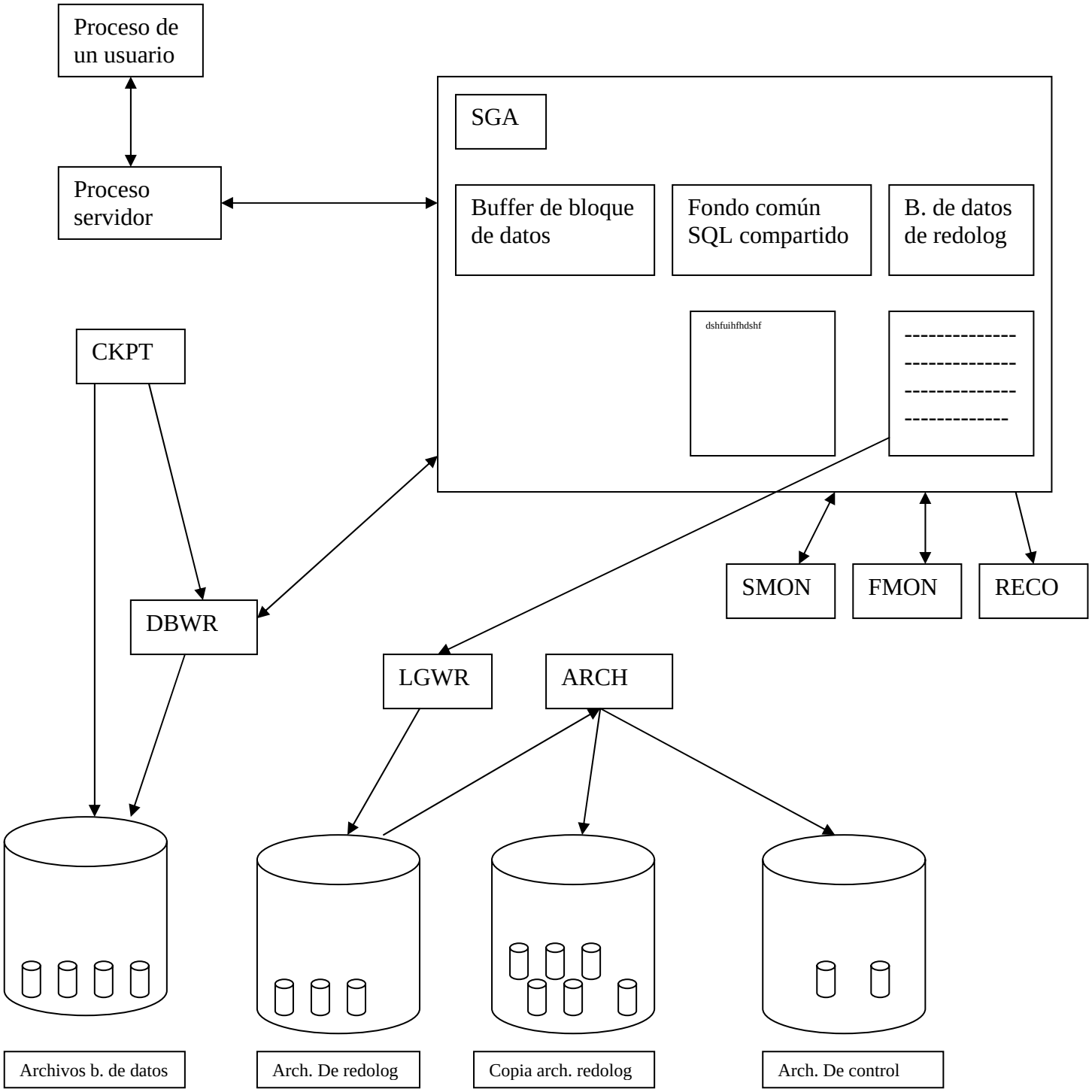
2- Buffer de registro de rehacer o redolog: Ese buffer registra las transacciones (insert, delete y update) o cambios efectuados en la base de datos antes de escribirse en los archivos del registro de rehacer.

3- Un fondo común SQL compartido: contiene las sentencias SQL ya ejecutadas sobre la base de datos. Si un usuario ejecuta una sentencia idéntica SQL se aprovecha todo el análisis de esa sentencia ya ejecutada para acelerar su ejecución.



9.4.3.Procesos de soporte de la base de datos.

Los procesos de usuario solicitan información a través de los procesos servidor. Los procesos de servidor toman las peticiones de los procesos de usuario y se comunican con la base de datos para satisfacer las necesidades del usuario.



Son los procesos background o procesos de fondo. Entre los mas importantes conviene citar:

- Escritor de bases de datos (database brite): este proceso es responsable de gestionar el contenido del buffer de datos de la SGA. Los bloques de los archivos de datos se almacenan en la SGA y realizan escritura de los bloques modificados en el archivo de datos. Cuando un usuario solicita

una petición cuyos datos no están en el buffer de datos el DBWR se encargará de llevar al buffer un nuevo bloque de información del archivo de datos.

- Punto de comprobación o control CKPT (check point) estos puntos provocan que el DBWR escriba en el archivo de datos todos los bloques que se hayan modificado desde el último punto de control y que actualice las cabeceras de los archivos de datos y los archivos de control. Se producen de forma automática siempre que se realiza el llenado de un registro de redolog.

- Escrito de registros LGWR (logbrite): gestiona la escritura del contenido del buffer de registros rehacer de la SGA a los archivos redolog.
Es el único proceso que escribe en lo archivos de registros de rehacer y el único que lee los buffer de ese registro.
Esta operación permite a oracle que pueda recuperarse en cualquier momento, si hay fallos. Los registros de los archivos de redolog deberán contener siempre la información anterior y la información actualizada.

- Supervisor de sistema SMON (system monitor):
El supervisor del sistema es un proceso obligatorio que se ocupa de todas las recuperaciones que sean precisas durante el arranque de la base de datos. Elimina todos los datos de las transacciones que el sistema ya no necesita y compacta los huecos libres en los ficheros de datos. Se activa de forma periódica para comprobar si su intervención es precisa.

- Supervisor de procesos PMON (proccess monitor):
Realiza una limpieza al terminar la ejecución de los procesos. Restaura las transacciones no válidas de los procesos de usuario que abortan. Liberando los bloques y los recursos de la SGA. Al igual que SMON se activa de forma periódica para comprobar si es necesaria su intervención.

- Archivador ARCH (arch iver):
Es opcional y archiva en disco o en cinta una copia de los ficheros redolog cuando estén llenos para una posible recuperación en caso de fallo de los discos. Se ejecuta en modo archivelog.

- Recuperador RECO (recoveder):
Es opcional. Recupera transacciones distribuidas dudosas y se usa en bases de datos oracle distribuidas.

Resumen

La base de datos de oracle mas elemental consta de:	Internamente las B. de datos contienen:
-Uno o mas archivos de datos.	-Varios usuarios/esquemas.
-Uno o mas archivos de control.	-Uno o mas espacios de tablas.
-Dos o mas archivos de redolog.	-Tablas de diccionario de datos.
	-objetos de usuario (tablas, vistas, etc..)

El servidor que accede a la b. de datos consta como mínimo de :

Un SGA que contiene:	-Proceso SMON
-Buffer de archivo de datos.	-Proceso PMON
-Buffer de registro de rehacer.	-Proceso DBWR
-Fondo común SQL.	-Proceso LGWR
	-Procesos de usuario los PGA asociados.

Ejemplo:

Queremos consultar la nota de un alumna luisa martinez en la asignatura de informatica.
Para ello hacemos una consulta a la tabla notas_alumnos de la base de datos a traves de las siguientes sentencias:

Select nota from notas_alumnos
Where asignatura=’INFORMATICA’ and nombre_alumno=’LUISA MARTINEZ’;

La sentencia Esq. Es un proceso de usuario.

1º el proceso de usuario pasa la sentencia sql al proceso del servidor por medio del SGA.

2º los procesos del servidor buscan en el fondo común SQL una versión ejecutable de la sentencia. Si la encuentra se ejecuta la sentencia sql. Si no se encuentra se procesa y se lleva la versión ejecutable al fondo común.

3º si los datos requeridos no están en el buffer del bloque de datos por medio del proceso DBWR se leen los archivos de datos y el contenido solicitado se coloca en los buffer del bloque de datos de la SGA .

4º una vez que los datos se encuentran en el buffer de bloque de datos el proceso de usuario puede leer la nota y enviársela al usuario.

Este es un ejemplo muy sencillo ya que ha sido simplemente una recuperación de información.

Supongamos que cambiamos la nota de informática de luisa martinez modificando la columna nota de la tabla notas_alumnos con el valor 7.

Update notas_alumnos set nota=7
where asignatura’INFORMÁTICA’ and nombre_alumno=’Luisa Martínez’;

Los pasos que da oracle con la sentencia update son los siguientes:

1º el proceso de usuario pasa la sentencia sql al proceso del servidor por medio del SGA.

2º los procesos del servidor buscan en el fondo común SQL una versión ejecutable de la sentencia. Si la encuentra se ejecuta la sentencia sql. Si no se encuentra se procesa y se lleva la versión ejecutable al fondo común.

3º si los datos requeridos no están en el buffer del bloque de datos por medio del proceso DBWR se leen los archivos de datos y el contenido solicitado se coloca en los buffer del bloque de datos de la SGA .

4º Se registran el valor antiguo de los datos en un segmento rollback (la nota antigua era 5).

5º se crea una transacción en el buffer de redolog (rehacer) .

6º se modifican los datos en el buffer del bloque de datos reflejando la nueva nota (un 7).

7º cuando se ejecute un COMMIT el LGWR escribirá los buffer de redolog a los archivos de redolog y se liberará la información de deshacer en el segmento rollback.

8º en algún momento se producirá un punto de comprobación o control (CKPT) y el DBWR escribirá en los archivos de datos todos los bloques que se hayan modificado desde el último punto de control.

9.4.4.¿Qué es una instancia oracle?

Una instancia de oracle es un conjunto de procesos background o de fondo (algunos opcionales) y una zona de memoria denominada SGA (system global area).

9.5.Gestión de seguridad.

La gestión de seguridad básicamente tiene que ver con la gestión de usuarios y con la concesión y supresión de privilegios a los usuarios. El DBA es el responsable de permitir o denegar el acceso a los usuarios, a determinados objetos o recursos de la base de datos. Se puede clasificar la seguridad de base de datos en 2 categorías, seguridad del sistema y seguridad de los datos.

Seguridad del sistema: incluye los mecanismos que controlan el acceso y uso de la base de datos a nivel sistema. Por ejemplo cada vez que se conecta un susario a la base de datos se controlará si tiene o no autorización para hacerlo, cuanto tiempo de conexión puede tener, cuantos intentos se le permiten para conectarse, etc.…

La seguridad de los datos: incluye mecanismos que controlan el acceso y uso de la base de datos a nivel objetos. Por ejemplo cada vez que un usuario acceda a un objeto (tabla, vista, etc….) los mecanismos de seguridad comprobaran si el usuario puede acceder a ese objeto y que tipo de operación puede hacer con el.

9.5.1.Usuarios.

Un usuario es un nombre definido en la base de datos que se puede conectar a el y acceder a determinados objetos según ciertas condiciones que define el administrador.
Asociado con cada usuario de la base de datos hay un esquema con el mismo nombre. **Un esquema es una colección lógica de objetos (tablas, vistas, sinónimos, funciones, etc.….)**. Por defecto cada usuario tiene acceso a los objetos de su esquema y puede acceder a los objetos de otro usuario siempre y cuando este otro usuario le haya concedido el privilegio de hacerlo.

Creación del usuario.

Cuando se instala una base de datos, se crean automáticamente 2 usuarios con los privilegios de administración.

Estos usuarios son: SYS ___CHANGE_ON_INSTALL
SYSTEM ___MANAGER

El usuarios SYS es el propietario de las tablas del diccionario de la tabla de datos, en ellas se almacena información sobre el resto de estructuras de la base de datos. Oracle maneja las tablas del usuario SYS. Ningún usuario aunque sea administrador puede modificar las tablas de SYS. Solo se debe conectar como SYS cuando las instrucciones de oracle lo exijan.

El diccionario de datos está formado por un conjunto de tablas y vistas en el TABLESPACE SYSTEM. Los usuarios tiene acceso de solo lectura a las vistas de este diccionario el cual se crea a la vez que la base de datos y es propiedad del usuario SYS.
Contiene…

Derechos y autorizaciones, restricciones, información sobre el espacio libre/ocupado e información de exportacion sobre otros objetos.

Los objetos del diccionario de datos a lso que el usuario puede acceder se encuentran en la vista **dictionary** que es propoiedad de sys.

El prefijo de una lista del diccionario indica el nivel de acceso user y all: accesibles para todos los usuarios.

DBA: solo el administrador puede utilizar estas vistas.

El usuario system lo crea oracle para realizar las tareas de administración.

No se suelen crear tablas de usuario en el esquema system. Para crear otros usuarios es preciso conectarse cono system ya que este posee los suficientes privilegios para crear usuarios:

CREATE USER nom_user IDENTIFIED BY contraseña.
[DEFAULT TABLESPACE espacio_tabla]
[TEMPORARY TABLESPACE espacio_tabla]
[QUOTA {entero {K | M } | UNLIMITED} ON spacio_tabla]
[PROFILE perfil];

default tablespace: asigna a un usuario el tablespace pro defecto para almacenar los objetos que cree.
Si no se asigna ningún tablespace por defecto será system.

Temporary tablespace: especifica el nombre del tablespace para trabajos temporales.
Si no se especifica ninguno, el tablespace pro defecto es system.

Tanto el espacio de tabla por defecto como el espacio de tabla temporal es recomendable que existan para evitar problemas sobre system

Cuota: asigna un espacio en megabyte o kilo byte en el espacio asignado. Si no se especifica esta cláusula el usuario no tiene cuota asignada y no podrá crear objetos en el tablespace.

Mediante el privilegio unlimited tablespace se dispone de recursos ilimitados a cualquier tablespace . est privilegio se da a través de la orden grant.

Profile: asigan un perfil al usuario. Si se omite oracle asigna el perfil pro omisión al usuario. Un perfil limita, el numero de sesiones concurrente de sesiones del usuario, limita el tiempo de la cpu, limita el tiempo de una sesión desconecta al usuario si sobrepasa cuanlkier tiempo asignado

Ejemplo:

SQL> create user A5

```
  2   identified by A5
  3   default tablespace users
  4  quota 10 k on users;
```

Usuario creado.

Modificación de usuario.

SQL> alter user A5 identified B5;

Con este comando se puede cambiar la contraseña de un usuario, siendo uno system.

ALTER USER nom_usuario IDENTIFIED BY contraseña;

Para ver la vista de dba_users:

desc dba_users

la descripción de algunas partes :

SQL> select username,default_tablespace,temporary_tablespace from dba_users;

Fijarse en si el temporary es temp.

Las opciones dadas a un usuario en la orden create user se pueden modificar con la orden alter user.
Se puede cambiar la clave de acceso, el tablespace por defecto, el tablespace temporal, la cuota y el perfil.

ALTER USER nombre de usuario
[DEFAULT TABLESPACE espacio_tabla]
[TEMPORARY TABLESPACE espacio_tabla]
[QUOTA {entero {K | M } | UNLIMITED} ON spacio_tabla]
[PROFILE perfil];

Borrado de usuarios.

Se puede borrar un usuario de la base de datos incluido los objetos que contiene a través de la orden:

DROP USER nom_usuario;

Para borrar un usuario y sus objetos, poner a continuación de nom_usuario CASCADE.

9.5.2 PRIVILEGIOS.

Un privilegio es la capacidad de un usuario dentro de la base de datos para realizar determinada operaciones o acceder a determinado objeto de otro usuario. Ningun usuario puede llevar a cabo una operación si ante no se le ha concedido permiso para ello. Mediante la asignación de privilegios se permite o restringe el acceso a los datos lal realización de conculta y cambio en los datos la posibilidad de realizar funciones del sistema.

Cuando se crea un usuario para que este pueda empezar a funcionar es necesario darle determinado privilegios. Oracle ofrece vario roles o funciones. (CONNECT RESOURCE) ambos son roles. Son roles con los que un usuario empieza a caminar por oracle. Ademas de connect y resource exitsten otros privilegios o roles que se pueden asignar directamente a cualquier usuario: DBA, EXP_FULL_DATABASE, IMP_FULL_DATABASE

ROL CONNECT
ALTER
CREATE CLUSTER
CREATE DATABASE LINK
CREATE SESSION
CREATE SYNONYM
CREATE TABLE
CREATE VIEW

ROL RESOURCE
Create cluster
Create table

Hay dos tipos de privilejios que se pueden definir en una base de datos privilegios sobre objetos.

Privilegios sobre objetos:
Son los que permiten acceder y realizar cambio en los datos de otro usuario. Por ejemplo: El privilegio de consultar la tabla de otro usuario. Se dispone de los siguientes privilegios sobre tablas, vista, secuencia y procedure.

PRIVILEGIOS SOBRE OBJETOS	TABLAS	VISTAS	SECUENCIAS	PROCEDURES
ALTER	X		X	
DELETE	X	X		
EXECUTE				X
INDEX	X			
INSERT	X	X		

REFERENCES	X		
SELECT	X	X	X
UPDATE	X	X	

PRIVILEGIOS SOBRE OBJETOS	SENTENCIAS SQL PERMITIDAS
ALTER	ALTER OBJETO (TABLA O VISTA)
DELETE	DELETE FROM (OBJETO TABLA O VISTA)
EXECUTE	EXECUTE nombre.objeto (PROCEDURE)
INDEX	CREATE INDEX ON objeto (SOLO TABLAS)
INSERT	INSERT INTO OBJETO (TABLA O VISTAS)
REFERENCES	CREATE ó ALTER TABLE DEFINIENDO UNA INTEGRIDAD DE CLAVE AJENA (SOLO TABLA)
SELECT	SELECT FROM OBJETO (TABLA VISTA SECUENCIA)
UPDATE	UPDATE OBJETO (TABLA O VISTA)

La orden para dar privilegios es GRANT con el siguiente formato:
Solo sobre objetos.

GRANT { priv.objeto [, priv.objeto]….|ALL [privilegio]}
[(columna[,columna]……)]
[usuario.]objeto
ON [usuario.]objeto
TO (usuario|rol|PUBLIC) [,(usuario|rol|PUBLIC) ..]
[WITH GRANT OPTION]

ON especifica el objeto sobre el que se dan los privilegios.
TO identifica a los usuario o ROLES a los que se
ALL concede todos os privilegio sobre el objeto especificado.
WITH GRAN OPTION permite que el receptor del privilegio pueda asignar roles o permisos a otros usuarios.
PUBLIC asigna los privilegio a todos los usuarios actuarles y

Privilegio del Sistema : Son los que dan privilegio o realizan las acciones importantes del sistema.

Por eje,pribill

Sl partimos de la orden:

GRANT (privilegio|rol) [,privilegio|rol)…]
TO (usuario|rol|PUBLIC) [,usuario|rol|UBLIC])
[WITH ADMIN OPTION]

TO identifica a los usuarios o roles.
WITH ADMIN OPTION permite que el receptor de privilegios o rol pueda conceder ese privilegio a otro usuario o rol.

Es posible que los problemas que nos ha dado oracle al crear tabla se debe a que seguramente debemos de darle una cuota obligatoria al nuevo usuario que hemos creado, esto se da a partir de una versión del oracle.

Ejemplos con el grant:

//para dar el permiso de insert en una única columna a ese usuario.
SQL> grant insert (idioma) on saludo
  2 to man1;

//para que ese usuario pueda borrar en esa tabla.
SQL> grant delete on saludo
  2 to man1;

Retirada de privilegios: al igual que se conceden privilegios se pueden retirar y para ello se utiliza la orden REVOKE que retira privilegios o roles concedidos a los usuarios y privilegios concedidos a los roles.
El formato para retirar privilegios de objeto a los usuarios o roles es:

REVOKE {priv_objeto [, priv_objeto]….. |ALL [PRIVILEGES] }
ON [usuario.] objeto
From {usuario |rol |public} [{usuario |rol |public}]…..

Ó

REVOKE {priv_sistema| rol} [,{priv_sistema| rol}]
From {numero |rol | public} [,{numero |rol | public}]

Vistas con información de los privilegios:
SESSION_PRIVS, privilegios del usuario activo
USER_SYS_PRIVS privilegios del sistema asignados al usuario.
DBA_SYS_PRIVS, privilegios de sistema asignados a lso usuarios o a los roles(solo funciona con system).
USER_TAB_PRIV, concesiones sobre objetos que son propiedad del usuario concedidos o recibidos pro este.
USER_TAB_PRIVS_MADE, concesiones a objetos que son propiedad del usuario (asignadas).
USER_TABS_PRIVS_RECD, concesiones sobre objetos que recibe el usuario.
USER_COL_PRIVS, concesiones sobre columnas en las que el usuario es el propietario, asigna el privilegio o lo recibe.
USER_COL_PRIVS_MADE, todas las concesiones sobre columnas de objetos que son propiedad del usuario.
USER_COL_PRIVS_RECD , concesiones sobre columnas recibidas pro el usuario

9.5.3.Roles:

Supongamos que un conjunto de usuarios de un departamento de una empresa requiere el mismo conjunto de privilegios par trabajar sobre determinados datos.
Este conjunto de privilegios se puede agrupar en un rol de tal manera que es posible asignar el mismo rol a cada uno de los usuarios. Un rol o función es un conjunto de privilegios que recibe un nombre común para facilitar la tarea de asignar estos a los usuarios o a otros roles. Los privilegios de un rol, pueden ser de sistema o de objeto.

En primer lugar se crea el rol con la orden:

CREATE ROLE nom_rol

Y a continuación se asignan los privilegios con la orden grant.

SQL> create role paco;

Rol creado.

Insertar los permisos(se hace varios grant):

SQL> grant select, insert on emple to paco;

Concesión terminada correctamente.

SQL> grant insert on depart to paco;

Concesión terminada correctamente.

SQL> grant create session to paco;

Concesión terminada correctamente.

Ahora le damos los roles al usuario:

SQL> grant paco to jose;

SQL> desc system.depart;

Nombre	¿Nulo?	Tipo
DEPT_NO	NOT NULL	NUMBER(2)
DNOMBRE		VARCHAR2(14)
LOC		VARCHAR2(14)

SQL> insert into system.depart
  2 values(10,' Pérez , Luis ', 'Madrid');

1 fila creada.

Quitar un permiso al rol creado.

SQL> revoke select on depart from paco;

Conectarse con el usuario y probar:

```
SQL> select * from system.depart;
select * from system.depart
              *
ERROR en línea 1:
ORA-01031: privilegios insuficientes
```

Supresión de un rol:
La sentencia DROP ROLE, permite eliminar un rol de la base de datos. Oracle retira el rol concedido a todos los usuarios a los que se les concedió; el formato es:

```
DROP ROLE nomb_rol;
```

9.6.Gestión de Tablespaces.

Una base de datos está formada entre otras cosas por un conjunto de archivos de datos.
Oracle agrupa estos archivos utilizando un objeto denominado TABLESPACE o espacio de tabla. Se llama así por que contiene tablas de datos. Antes de introducir datos en una base de datos es necesario crear o disponer de un TABLESPACE y seguidamente las tablas en las que se van a introducir los datos. Estas tablas deben almacenarse en un tablespace.
Un tablespace es una unidad lógica de almacenamiento de datos representada físicamente por uno o más archivos de datos. Se recomienda no mezclar datos de diferentes aplicaciones en el mismo TABLESPACE, es decir, es conveniente crear un TABLESPACE para almacenar los datos de almacén, otro para los datos de compras, etc.…
Existen varios TABLESPACES que oracle crea al instalar la base de datos:

- SYSTEM
- USERS
- Etc…

Creación de un TABLESPACE: para crear un TABLESPACE, como mínimo se tiene que tener permisos de DBA o ser administrador, se usa la orden:

```
CREATE TABLESPACE nom_tablespace
DATAFILE ‘nom_archivo’ [SIZE entero {k|m}] [REUSE]
                [,‘nom_archivo’ [SIZE entero {k|m}] [REUSE] ]…..
[ DEFAULT STORAGE
( INITIAL tamaño
NEXT tamaño
MINIEXTENTS tamaño
MAXEXTENTS tamaño
PCTINCREASE valor) ]
[ ONLINE | OFFLINE ];
```

DATAFILE: especifica el nombre de archivo o archivos de que constará el tablespace.
SIZE entera: especifica el tamaño del tablespace que pude venir dado pro kilobytes(K) o megabytes (M).
REUSE: reutiliza el archivo si ya existe o crea el archivo si no existe.

DEFAULT STORAGE: define el almacenamiento pro omisión para todos los objetos que se creen en este espacio de tabla. Fija la cantidad de espacio si no se especifica en la sentencia create table.
INITIAL: extensión inicial. Especifica el tamaño en bites de la primera extensión del objeto. El tamaño se puede especificar en K o M. el valor por defecto son 10K.
NEXT: extensión siguiente. Especifica el tamaño de la siguiente extensión que se va a asignar al objeto. El tamaño se puede especificar en K o M y el valor por defecto son 10K.
MINIEXTENT: reserva extensiones adicionales mas allá de la extensión inicial que se da a la tabla. Este parámetro permite asignar una gran cantidad de espaciocuando se crea un espacio, incluso si el espacio disponible no está contiguo.el valor pro omisión es 1 que significa que solo se asigna la extensión inicial. Si el valor es mayor a 1, se calcula el tamaño de las extensiones siguientes basándose en los valores (parámetros) INITIAL, NEXT y PCTINCREASE.
MAXEXTENTS: es el numero total de extensiones incluida la primera que oracle puede asignar a un objeto. El valor depende del tamaño del bloque de datos. Ejemplo: si el tamaño del bloque es 2K, el valor es 121; si el tamaño del bloque es 4K el valor es 249.
PCTINCREASE: es un factor de crecimiento para las extensiones. El valor por omision es 50, lo que significa que cada extensión será un 50% mas grande que la extensión anterior.
ONLINE, OFFLINE: con ONLINE el tablespace está disponible después de crearlo y es el valor por defecto. OFFLINE impide su acceso.
Crear un TABLESPACE de 15megas llamado traba6
El tamaño inicial para el objeto que se cree en el tablespace (por ejemplo una tabla) es de 15k.
El tamaño de la siguiente extensión es también 15k, cada extensión siguiente será un 24% mas grande que el anterior. Asignamos a este tablespace 2 archivos traba61 y traba62.

```
SQL> create tablespace traba6
  2  datafile 'traba61' size 10 m,
  3   'traba62' size 5 m
  4  default storage
  5  (initial 15 k
  6  next 15 k
  7  pctincrease 24);
```

Tablespace creado.

Hay que poner la extensión .ora para que cree el archivo con esa extensión seguro!!!

```
SQL> desc dba_data_files
 Nombre                                    ¿Nulo?   Tipo
 ----------------------------------------- -------- ------------
 FILE_NAME                                          VARCHAR2(513)
 FILE_ID                                   NUMBER
 TABLESPACE_NAME                                    VARCHAR2(30)
 BYTES                                     NUMBER
 BLOCKS                                    NUMBER
 STATUS                                    VARCHAR2(9)
 RELATIVE_FNO                              NUMBER
 AUTOEXTENSIBLE                                     VARCHAR2(3)
 MAXBYTES                                  NUMBER
 MAXBLOCKS                                 NUMBER
 INCREMENT_BY                              NUMBER
 USER_BYTES                                NUMBER
 USER_BLOCKS                               NUMBER
```

SQL> select * from user_free_space;

```
TABLESPACE_NAME                FILE_ID   BLOCK_ID     BYTES     BLOCKS
RELATIVE_FNO
------------------------------ ---------- ---------- ---------- ---------- ------------
SYSTEM                              1      51201      65536          8          1
UNDOTBS1                            2        201     131072         16          2
UNDOTBS1                            2        225      65536          8          2
UNDOTBS1                            2        241     196608         24          2
UNDOTBS1                            2        297      65536          8          2
UNDOTBS1                            2        337      65536          8          2
UNDOTBS1                            2        657      65536          8          2
[…]
```

Con ello vemos el espacio que nos queda libre en los tablespace, y asi saber en cuales tenemos espacio.

BLOCK_ID: identificación del primer bloque libre.

BYTES: Número de bytes libres.

BLOCK: Número de bloques libres..

RELATIVE_FNO: Número relativo del fichero en la primera extensión del bloque.

SQL> select * from dba_free_space;

```
TABLESPACE_NAME                FILE_ID   BLOCK_ID     BYTES     BLOCKS
RELATIVE_FNO
------------------------------ ---------- ---------- ---------- ---------- ------------
SYSTEM                              1      51201      65536          8          1
UNDOTBS1                            2        201     131072         16          2
UNDOTBS1                            2        225      65536          8          2
UNDOTBS1                            2        241     196608         24          2
UNDOTBS1                            2        297      65536          8          2
UNDOTBS1                            2        337      65536          8          2
[…]
```

Este comando informa sobre las extensiones libres en los tablespace. Reservado para los administradores.

SQL> select tablespace_name,block_size,initial_extent,max_extents,pct_increase,min_extlen,status from dba_tablespaces;

```
TABLESPACE_NAME   BLOCK_SIZE INITIAL_EXTENT MAX_EXTENTS
PCT_INCREASE MIN_EXTLEN
------------------------------ ---------- -------------- ----------- ------------ ----------
STATUS
---------
SYSTEM                          8192        65536              2147483645              65536
```

```
ONLINE

UNDOTBS1                       8192         65536  2147483645              65536
ONLINE

TEMP                        8192       1048576                0    1048576
ONLINE

CWMLITE                        8192         65536  2147483645              65536
ONLINE
```

```
SQL> select * from dba_ts_quotas;

TABLESPACE_NAME                USERNAME                       BYTES  MAX_BYTES
BLOCKS MAX_BLOCKS
------------------------------ ------------------------------ ---------- ---------- ---------- ----------
ODM                            ODM                           5439488         -1        664         -1
ODM                            ODM_MTR                          4259840         -1        520         -1
CWMLITE                           OLAPSYS                          9764864         -1       1192         -1
TOOLS                           RMAN                           6291456         -1        768         -1
EXAMPLE                           HR                           1638400         -1        200         -1
EXAMPLE                           OE                           6946816         -1        848         -1
EXAMPLE                           PM                          14811136         -1       1808         -1
EXAMPLE                           SH                         125763584         -1      15352         -1
EXAMPLE                           QS_ADM                              0         -1          0         -1
EXAMPLE                           QS                           1572864         -1        192         -1
EXAMPLE                           QS_WS                            1179648         -1        144         -1
```

USERNAME: nombre de usuario.
BYTES: numeros de bytes utilizados por el usuario.
MAX_BYTES : numero maximo de bytes que tiene asignado el usuario
BLOCKS: numero de bloques usados.
MAX_BLOCKS: numero maximo de bloques.

*en MAX_BYTES y MAX_BLOCKS si el valor es -1 significa que tiene asignado un numero de bytes ilimitados.

Hay 3 parametros para borrar tablespace.

```
SQL> drop tablespace traba6 including contents and datafiles cascade constraints;
```

*al crear el tablespace creamos 2 ficheros, si la versión de oracle que usamos es servidor y cliente, en vez de personal, los ficheros del tablespace se encontrarán en el servidor; sin embargo, en la versión personal, los ficheros si se encontraran en el propio ordenador personal.

Para ver la dirección de los ficheros del tablespace, teclear:

SQL> select file_name,tablespace_name, bytes, blocks, status from dba_data_files;

Crear un tablespace y hacer una copia con el rename.

Pondremos el tablespace en etado offline, para que mientras lo usemos, nadie pueda entrar:

```
SQL> alter tablespace tra5
  2  offline;
```

al ponerlo offline no podemos añadir…asi que lo volvemos a poner online.

```
SQL> alter tablespace tra5
  2  add datafile 'f513.ora' size 2 m,
  3  'f523.ora' size 1 m;
```

Tablespace modificado.

Ahora directamente desde el SO haremos una copia de los ficheros con otro nombre!!!
Y renombrarlos:

```
SQL> alter tablespace tra5
  2  rename datafile 'tra51.ora','tra52.ora' to 'dtra51.ora','dtra52.ora';
```

Tablespace modificado.

9.7.Secuencias.

Una secuencia es un objeto de la base de datos que sirve para generar enteros únicos y es muy útil para generar automáticamente, valores para claves primarias, para crear una secuencia en el esquema propio. Para crear una secuencia en el esquema propio es necesario tener el privilegio CREATE SEQUENCE y su formato es el siguiente:

```
CREATE SEQUENCE nom_secuencia
[ INCREMENT BY entero]
[START WITH entero]
[MAXVALUE entero | NOMAXVALUE]
[MINVALUE entero | NOMINVALUE]
[CYCLE |NOCYCLE]
[ORDER | NOORDER];
```

Donde INCREMENT BY entero, especifica el intervalo de crecimiento de la secuencia.
Si se omite se asume valor 1.
START WITH entero: es el numero donde comienza la secuencia.
MAXVALUE entero: es el número mas alto que genera la secuencia.
MINVALUE entero: es el numero mas bajo que generará la secuencia.
CYCLE | NOCYCLE: cycle reanuda la secuencia cuando llega al máximo o al mínimo valor; y nocycle no la reanuda.

ORDER |NOORDER: order garantiza que los números de secuencia se generan en el orden requeridos. Noorder no lo garantiza.

Una vez creada la secuencia accedemos a ella mediante las pseudocolumnas CURRVAL, que devuelve el valor actual de la secuencia y NEXTVAL, que devuelve el siguiente valor e incrementa la secuencia. Para acceder a estos valores tenemos que poner el nombre de la sentencia un punto (.) y a continuación la pseudocolumna.

Creamos una secuencia, que empiece en 5 y ke vaya de 2 y el maximo 57

```
SQL> create sequence mo2
  2  increment by 2
  3  start with 5
  4  maxvalue 57
  5  minvalue 5;
```

y creamos una tabla en la que introduciremos en laparte numerica la secuenacia que hemos creado de la siguiente maner:

```
SQL> insert into saludo
  2  values(mo2.nextval,'hola');

SQL> insert into saludo
  2  values(mo2.nextval,'hello');

SQL> insert into saludo
  2  values(mo2.nextval,'ciaoo');
```

y se mostrará:

```
SQL> select * from saludo;

     PAÍS SALUDO
---------- ---------------
         5 hola
         7 hello
         9 ciaoo
```

si hacemos un nexval, te muestra el sig. Número y hace un salto cuando hagamos los insert:

```
SQL> select mo2.nextval from dual;

   NEXTVAL
----------
        11

SQL> insert into saludo
  2  values(mo2.nextval,'oliiii');

1 fila creada.

SQL>  select * from saludo;

     PAÍS SALUDO
---------- ---------------
         5 hola
         7 hello
```

9 ciaoo
13 oliiii