Módulo 0: **Emacs** (es un editor de texto con una gran cantidad de funciones, muy popular entre programadores y usuarios técnicos.)

Módulo 1: SQL Básico
- Generalidades de SQL plus

SQL*PLUS es una herramienta de Oracle que reconoce y envía sentencias SQL al servidor Oracle para su ejecución.

- Contiene su propio lenguaje de comandos.
- Permite abreviatura de palabras claves de SQL*PLUS.
- Permite guardar y recuperar sentencias SQL en archivos.
- Desde la línea de comandos:

SQL*PLUS
Comandos de Edición

- A[PPEND] texto
- C[HANGE] /antiguo /nuevo
- C[HANGE] /texto /
- CL[EAR] BUFF[ER]
- DEL [n] [m]
- I[INPUT] [texto]
- L[IST] [n] [m]
- R[UN]
- n [texto]
- 0 texto

SQL*PLUS
Comandos de Ficheros

- SAV[E] nombre_fichero[.ext] [REP[LACE] | APP[END]]
- GET nombre_fichero[.ext]
- STA[RT] nombre_fichero[.ext]
- @nombre_fichero[.ext]
- ED[IT] nombre_fichero[.ext]
- SPO[OL] [nombre_fichero[.ext] | OFF | OUT]
- EXIT

¿Qué es SQL.

Structured Query Language

- Establecido como el lenguaje de base de datos relacional estándar.
- Existen numerosos productos que soportan SQL, cada uno de ellos con pequeñas diferencias sin apenas importancia (p.ej. Oracle).

- El SQL estándar es el publicado por ANSI e ISO.

Características de S.Q.L.

- Lenguaje de definición de datos (DDL)
  - Create, Alter, Drop.
- Lenguaje de manipulación de datos (DML)
  - Insert, Update, Delete.
- Lenguaje de control de datos (DCL)
  - Grant, Revoke.
- Control de transacciones
  - Commit, Rollback, Savepoint
- Restricciones de integridad
  - Referencial, datos.

Expresiones aritméticas

- **Operadores:**
  - Suma (+)
  - Resta (-)
  - Multiplicación (*)
  - División (/)

Precedencia de operadores: * / + -

Operadores misma prioridad se evalúan de izq. a derecha.

Paréntesis sobreescriben reglas de precedencia.

Ejemplos:

```
SELECT  5+8,  8-5,   8*4, 10/2  FROM DUAL
```

Operador de concatenación

Representado por dos barras verticales: ||

Vincula columnas o cadenas de caracteres.

Crea una columna resultado que es una expresión de tipo carácter.

Ejemplo:

```
SELECT 'Alfredo'||40 AS Password FROM DUAL;
```

Visualización de estructura de tablas

Comando Describe (Visualización Estructura de Tablas)

Oracle nos proporciona un comando que resulta muy útil cuando queremos conocer la estructura de una tabla, las columnas que la forman y su tipo y restricciones. Este comando toma una mayor importancia según nos alejemos del momento de creación de una tabla.

La sintásis es la siguiente

DESCRIBE tabla

Y un ejemplo de su utilización se puede ver al describir la definición de las dos tablas creadas antes. Como no es una sentencia SQL no necesita el ';' al final. También se puede abreviar como `DESC`.

Funciones.-

Funciones de valores simples:

1. ABS(n)= Devuelve el valor absoluto de (n).
2. CEIL(n)= Obtiene el valor entero inmediatamente superior o igual a "n".
3. FLOOT(n) = Devuelve el valor entero inmediatamente inferior o igual a "n".
4. MOD (m, n)= Devuelve el resto resultante de dividir "m" entre "n".
5. NVL (valor, expresión)= Sustituye un valor nulo por otro valor.
6. POWER (m, exponente)= Calcula la potencia de un numero.
7. ROUND (numero [, m])= Redondea números con el numero de dígitos de precisión indicados.
8. SIGN (valor)= Indica el signo del "valor".
9. SQRT(n)= Devuelve la raíz cuadrada de "n".
10. TRUNC (numero, [m])= Trunca números para que tengan una cierta cantidad de dígitos de precisión.
11. VAIRANCE (valor)= Devuelve la varianza de un conjunto de valores la varianza es un termino estadistio y expresa la desviación tipica o Standard de determinado valor.

Funciones de grupos de valores:

1. AVG(n)= Calcula el valor medio de "n" ignorando los valores nulos.

2. COUNT (* | Expresión)= Cuenta el numero de veces que la expresión evalúa algún dato con valor no nulo. La opción "*" cuenta todas las filas seleccionadas.
3. MAX (expresión)= Calcula el máximo.
4. MIN (expresión)= Calcula el mínimo.
5. SUM (expresión)= Obtiene la suma de los valores de la expresión.
6. GREATEST (valor1, valor2...)= Obtiene el mayor valor de la lista.
7. LEAST (valor1, valor2...)= Obtiene el menor valor de la lista.

Funciones que devuelven valores de caracteres:

1. CHR(n) = Devuelve el carácter cuyo valor en binario es equivalente a "n".
2. CONCAT (cad1, cad2)= Devuelve "cad1" concatenada con "cad2".
3. LOWER (cad)= Devuelve la cadena "cad" en minúsculas.
4. UPPER (cad)= Devuelve la cadena "cad" en mayúsculas.
5. INITCAP (cad)= Convierte la cadena "cad" a tipo titulo.
6. LPAD (cad1, n[,cad2])= Añade caracteres a la izquierda de la cadena hasta que tiene una cierta longitud.
7. RPAD (cad1, n[,cad2])= Añade caracteres a la derecha de la cadena hasta que tiene una cierta longitud.
8. LTRIM (cad [,set])= Suprime un conjunto de caracteres a la izquierda de la cadena.
9. RTRIM (cad [,set])= Suprime un conjunto de caracteres a la derecha de la cadena.
10. REPLACE (cad, cadena_busqueda [, cadena_sustitucion])= Sustituye un carácter o caracteres de una cadena con 0 o mas caracteres.
11. SUBSTR (cad, m [,n])= Obtiene parte de una cadena.
12. TRANSLATE (cad1, cad2, cad3)= Convierte caracteres de una cadena en caracteres diferentes, según un plan de sustitución marcado por el usuario.

Funciones que devuelven valores numéricos:

1. ASCII(cad)= Devuelve el valor ASCII de la primera letra de la cadena "cad".
2. INSTR (cad1, cad2 [, comienzo [,m]])= Permite una búsqueda de un conjunto de caracteres en una cadena pero no suprime ningún carácter después.
3. LENGTH (cad)= Devuelve el numero de caracteres de cad.

Funciones para el manejo de fechas:

1. SYSDATE= Devuelve la fecha del sistema.

2. ADD_MONTHS (fecha, n)= Devuelve la fecha "fecha" incrementada en "n" meses.
3. LASTDAY (fecha)= Devuelve la fecha del último día del mes que contiene "fecha".
4. MONTHS_BETWEEN (fecha1, fecha2)= Devuelve la diferencia en meses entre las fechas "fecha1" y "fecha2".
5. NEXT_DAY (fecha, cad)= Devuelve la fecha del primer día de la semana indicado por "cad" después de la fecha indicada por "fecha".

Formatos de Fecha.-

La función `TO_CHAR(<fecha>,<formato>)` traduce una fecha/hora (o parte de ella) a una cadena de caracteres, y `TO_DATE(<fecha>,<formato>)` transforma una cadena de caracteres a una fecha, hora o combinación fecha/hora.

Por ejemplo, `TO_DATE('02/08/2002', 'DD/MM/YYYY')` daría la fecha 2 de agosto de 2002. Si suponemos que un atributo `FECHA` almacena esa misma fecha a las 3 de la tarde, `TO_CHAR(FECHA, 'DD-MON-YYYY HH:MI AM')` devolvería el string `02-AGO-2002 03:00 PM`. El formato es una cadena de caracteres en la que se indica el formato siguiendo las claves que se muestran en la Tabla 2.1. Esta tabla no incluye absolutamente todos los formatos, pero sí un buen número de ellos, que deberían ser más que suficientes para un uso normal.

Tabla 2.1: Formatos de fecha en Oracle

	Siglos y años
`CC`	Siglo
`SCC`	Siglo. Si es AC (Antes de Cristo), lleva un signo –
`YYYY`	Año, formato de 4 dígitos
`SYYY`	Año, formato de 4 dígitos. Si es AC lleva un signo –
`YY`	Año, formato de 2 dígitos
`YEAR`	Año, escrito en letras y en inglés (por ejemplo, 'TWO THOUSAND TWO')
`SYEAR`	Ídem, pero si es AC lleva el signo –
`BC`	Antes o Después de Cristo (AC o DC) para usar con los anteriores, por ejemplo `YYYY BC`

Meses

Q	Trimestre: Ene-Mar=1, Abr-Jun=2, Jul-Sep=3, Oct-Dic=4
MM	Número de mes (1-12)
RM	Número de mes en números romanos (I-XII)
MONTH	Nombre del mes completo rellenado con espacios hasta 10 espacios (SEPTIEMBRE)
FMMONTH	Nombre del mes completo, sin espacios adicionales
MON	Tres primeras letras del mes: ENE, FEB,...

Semanas

WW	Semana del año (1-52)
W	Semana del mes (1-5)

Días

DDD	Día del año (1-366)
DD	Día del mes (1-31)
D	Día de la semana (1-7)
DAY	Nombre del día de la semana rellenado a 9 espacios (MIÉRCOLES)
FMDAY	Nombre del día de la semana, sin espacios
DY	Tres primeras letras del nombre del día de la semana
DDTH	Día (ordinal): 7TH
DDSPTH	Día ordinal en palabra, en inglés: SEVENTH

horas

HH	Hora del día (1-12)
HH12	Hora del día (1-12)
HH24	Hora del día (1-24)
SPHH	Hora del día, en palabra, inglés: SEVEN
AM	am o pm, para usar con HH, como 'HH:MI am'
PM	am o pm
A.M.	a.m. o p.m.
P.M.	a.m. o p.m.

Minutos y segundos

MI	Minutos (0-59)
SS	Segundos (0-59)

`SSSS` Segundos después de medianoche (0-86399)

Además de estas palabras clave, el formato puede incluir espacios y los signos de puntuación `-/,.;:`. Cualquier otro carácter debe ir `"entre comillas dobles"`.

Funciones de conversión:

1. TO_CHAR = Transforma un tipo DATE ó NUMBER en una cadena de caracteres.
2. TO_DATE = Transforma un tipo NUMBER ó CHAR en DATE.
3. TO_NUMBER = Transforma una cadena de caracteres en NUMBER.

Módulo 2: Trabajo con varias tablas

Concepto de JOIN

- Un JOIN se utiliza para consultar datos de más de una tabla
- La condición de JOIN se escribe en la cláusula WHERE.
- Si existen columnas con el mismo nombre en las tablas seleccionadas, se deberán nombrar los campos
- Ejemplo:

Tipos de JOIN

* Existen dos tipos principales de JOIN:

EQUIJOIN Join sobre dos o más tablas, por igualdad de campos.

NON- EQUIJOIN Por desigualdad, sin correspondencia directa entre campos de tablas. La relación se puede establecer mediante criterios de rango (<, >, BETWEEN, ...)

* Y dos más adicionales:

OUTER JOIN Para ver, también, las filas que no complen la condición de Join. El operador de un Outer Join es el signo más (+), en el "lado" del join que es deficiente en información.

SELF JOIN Combinación de una tabla consigo misma

Ejemplos de Equijoin

```
select emp.empno, emp.ename, emp.deptno, dept.empno, dept.loc
from  emp, dept where emp.empno=dept.deptno;

select emp.empno, emp.ename, emp.deptno, dept.empno, dept.loc
from  emp JOIN dept ON emp.empno=dept.deptno;

(Hay un error en este ejemplo)
```

Ya que la columna DEPTNO es igual en ambas tablas, ésta debe ir prefijada por el nombre de la tabla para evitar la ambigüedad tambien se puede utilizar un Alias.

Ejemplo de Non- Equijoins

- `select e.ename, e.sal, s.grade from emp e, salgrade s where e.sal BETWEEN s.losal and s.hisal ;`
- `Select e.ename, e.sal, s.grade from emp e JOIN salgrade s ON e.sal BETWEEN s.losal and s.hisal`

En este ejemplo se han usado alias de tablas (*e* para la tabla emp y s para la tabla salgrade).

Ejemplo de Outer Join

- `select e.ename, d.deptno, d.dname from emp e, dept d where e.deptno(+)=d.deptno order by e.deptno;`
- `select e.ename, d.deptno, d.dname from emp e LEFT OUTER JOIN dept d ON e.deptno=d.deptno order by e.deptno;`

En este ejemplo se muestran los números y nombres de todos los departamentos, incluidos aquellos que no tienen empleado.

- Si se le añade: **AND emp.deptno is null**, sólo se mostrarían las no coincidencias.

Módulo 3: Funciones agregadas y Subconsultas

Funciones de Grupo (I)

- AVG ([DISTINCT | ALL] n)
  - Valor promedio de n.
- COUNT ({* | [DISTINCT | ALL |] expr})
  - Cantidad de filas con expr no nulo. Con * se cuentan todas las filas incluyendo duplicadas y valores nulos.
- MAX ([DISTINCT | ALL] expr)
  - Valor máximo de expr.
- MIN ([DISTINCT | ALL] expr)
  - Valor mínimo de expr., ignorando los valores nulos.

Ejemplos:

```
SELECT 'Alfredo'||40 AS Password FROM DUAL; SELECT MIN(Sal) FROM Emp;
```

Funciones de grupo y Nulos

- Las funciones de grupo IGNORAN los valores nulos de las columnas.
- ¿Qué resultado obtendríamos se calculamos la media de la comisión de los empleados?

NVL y funciones de grupo

- Esta media no es correcta porque se han ignorado las filas cuya comisión es nula.
- Solución: Uso de la función NVL para forzar a las funciones de grupo que admitan los valores nulos.

Uso de GROUP BY (I)

- Si se incluye una función de grupo en una cláusula SELECT, no se puede seleccionar resultados individuales a menos que la columna aparezca en la cláusula GROUP BY.
- No se pueden usar alias en GROUP BY.
- Por defecto, tras un GROUP BY, las filas se ordenan de forma ascendente
- Ejemplo:

```
SELECT deptno, AVG(sal) FROM emp GROUP BY deptno;
```

Uso de GROUP BY (II)

- La columna referenciada por GROUP BY no es necesario seleccionarla.
- Todas las columnas mencionadas en la SELECT que no son funciones de grupo, tienen que estar en la cláusula GROUP BY.
- Se pueden formar agrupaciones sobre múltiples columnas:

```
SELECT deptno, job, sum(sal) FROM emp GROUP BY deptno, job;
```

Consultas no válidas

- Cualquier columna o expresión en la SELECT que no sea una función agregada, tiene que ser especificada en la cláusula GROUP BY
- SQL> SELECT deptno, COUNT(ename) FROM emp;
- No puede usar una cláusula WHERE para restringir grupos. Utilice la cláusula HAVING para restringir grupos.
  SQL> SELECT deptno, AVG(sal) FROM emp WHERE AVG(sal) > 2000 GROUP BY deptno;

Cláusula HAVING

- Use la cláusula HAVING para restringir grupos:
  - Los registros son agrupados
  - Se aplica la función de grupo
  - Los grupos que se corresponden con la cláusula HAVING se visualizan (condición TRUE).
- HAVING puede preceder a GROUP BY, pero se recomienda que se ponga en primer lugar GROUP BY porque es más lógico. (1º se

calculan grupos y posteriormente se calcula HAVING sobre esos gpos.).

Ejemplo:

```
SELECT DeptNo, COUNT(*) AS Cantidad FROM Emp GROUP BY DeptNo HAVING
COUNT(*) > 1
```

***Manejo de Valores Nulos**

El valor Null almacenado en una columna puede confundir en la realización de consultas la sumatoria de dos valores uno nulo y otro normal va a dar nulo. Pensando que daria el otro valor por tanto para manejar posibles valores nulos es recomendable usar la Funcion NVL con el valor que se quiere conseguir en columnas Nulas.

Ej: `SELECT NVL(Sal,0)*0.5 FROM employee;`
(En este caso si salario es Nulo sacara un 5% de 0 Y dara 0 de lo contrario dara NULL.

- Sentencias DML:

* Sentencias DML son:
  - o INSERT Añade registros a una tabla.
  - o UPDATE Modifica registros existentes de una tabla.
  - o DELETE Elimina registros existentes de una tabla.

La Sentencia INSERT

INSERT INTO TABLA (COLUMNAS) VALUES (VALOR)

- Mediante esta sentencia sólo se inserta un registro cada vez.
- El nombre de las columnas es opcional. Si se omiten se deben colocar los valores en el orden que las columnas tienen en la tabla.
- Caracteres y fechas entre comillas simples.

Inserción de Valores Nulos

- Método Implícito: Omitir la columna en la lista:

```
INSERT INTO dept (deptno, dname)VALUES (45, 'jaguar');
```

Método Explícito: Especificar NULL o el string vacío (‘’), para cadenas y fechas, en la lista de VALUES:

```
INSERT INTO dept VALUES (75, 'honda', null);
```

Inserción Valores Especiales

* SYSDATE registra la fecha y hora actual:

```
insert into emp (empno, ename, job, mgr, hiredate, sal, comm, deptno)
values (7196, 'AZUL, 'VENDEDOR', 7782, sysdate, 2007, null, 10);
```

*USERID inserta el nombre del usuario actual

Inserción de registro de otra tabla

- Se escribe el comando INSERT con una subconsulta.
- No usar la cláusula VALUES.
- Deben coincidir el número de columnas de INSERT con el de la subconsulta

  insert into managers (empno, ename, sal, hiredate) select empno, name, sal,hiredate from emp where job='MANAGER';

La Sentencia UPDATE

UPDATE <nombre- tabla>
SET <columna1> = valor1 [, <columna2> = valor2 ...?
[WHERE <condición>?

Actualiza los campos correspondientes junto con los valores que se le asignen, en el subconjunto de filas que cumplan la condición de selección. Si no se pone condición de selección, la actualización se da en todas las filas de la tabla.
Si se desea actualizar a nulos, se asignará el valor NULL.

- Los registros a modificar se especifican por medio de la cláusula WHERE.
- Si se omite WHERE se modificarían todos los registros de la tabla.

Modificación con sub- consultas

- P.ej.: Modificar el oficio y departamento del empleado 7698, con los valores correspondientes actualmente al empleado 7499:

```
    update emp
          set (job, deptno)= (select job, deptno from emp  where emp-
no=7499)
       where empno = 7698;
```

La Sentencia DELETE

DELETE FROM <nombre- tabla>
[WHERE <condición>?

Si no se pone condición de selección, borra todas las filas de la tabla.

- Los registros a eliminar se especifican en la cláusula WHERE.
- Si se omite WHERE se borrarán todos los registros de la tabla.

Eliminación con subconsulta

- Utilice subconsultas en sentencias DELETE, para eliminar registros de una tabla, basados en valores de otra tabla:

```
delete from emp
where deptno=(select deptno
              from dept
              where dname='SALES');
```

- Objetos de la Base de datos

Aunque no es objeto de este curso conocer todos los objetos de una base de datos es importante que el estudiante que se esta iniciando en ORACLE conozca de su existencia, por tanto mencionare los principales;

1. Tablas
2. Indices
3. Constrains
4. Funciones
5. Procedures
6. Paquetes (Pakages)
7. Disparadores (Triguers)
8. Secuenciadores
9. DBLINK
10. Sinonimos
11. Etc…

Módulo 4: Creación y modificación de tablas

Definición de Datos (DDL)

Sentencias DDL son:

- CREATE TABLE Crea una tabla. Para ello el usuario debe de tener el privilegio CREATE TABLE.

  ```
  create table t (
   a number,
   b varchar2(10));
  ```

- Necesario tener privilegio CREATE TABLE.
- Ha de especificar:
  - Nombre de tabla
  - Para las columnas: nombre, tipo de dato y tamaño.

Reglas para los nombres

- Deben de comenzar con una letra.
- Pueden tener una longitud de 1 – 30 caracteres de largo.
- Deben contener solamente A-Z, a-z, 0-9, _, $ y #.
- No deben duplicar el nombre de otro objeto que sea propiedad del mismo usuario o schema.
- No debe ser una palabra reservada del servidor Oracle8.

Tipos de Datos

VARCHAR2(tamaño)	Dato carácter de longitud variable. Máx. 4000.
CHAR(tamaño)	Dato carácter de longitud fija. Máx. 255.
NUMBER(p,s)	Dato numérico de longitud variable.p entre 1..38; s entre 84..127
DATE	Valores de fecha y hora. Entre el 1 Enero 4712 A.C. Y el 31 Diciembre del 4712 D.C.
LONG	Dato carácter de long.variable hasta 2 Gb.
CLOB	Dato carácter "single-byte" de hasta 4 Gb.
RAW(tamaño) y LONG RAW	Datos Binarios según tamaño especificado y Datos Binarios de long.variable hasta 2 Gb.
BLOB	Datos Binarios hasta 4 Gb.
BFILE	Datos binarios almacenados en fich. Externo. Hasta 4 Gb.

Creación de tabla por subconsulta

- Se puede crear una tabla e insertar filas combinando el comando CREATE TABLE con la opción AS subconsulta.
- Es necesario hacer coincidir la cantidad de columnas especificadas con las de la subconsulta.
- Si no se indican nombres de columnas, éstas serán los mismos que los de la subconsulta.

```
create table c_libre as select * from emp;

create table c_libre2 as select sal*12 salario, comm, sal, ename from
emp;
```

ALTER TABLE

- Permite modificar la estructura definida para una tabla.
- La sentencia ALTER TABLE sirve para modificar la estructura de una tabla que ya existe. Mediante esta instrucción podemos añadir columnas nuevas, eliminar columnas. Ten cuenta que cuando eliminamos una columna se pierden todos los datos almacenados en ella.

- También nos permite crear nuevas restricciones o borrar algunas existentes. La sintaxis puede parecer algo complicada pero sabiendo el significado de las palabras reservadas la sentencia se aclara bastante; ADD (añade), ALTER (modifica), DROP (elimina), COLUMN (columna), CONSTRAINT (restricción).

Ejemplo.-

- ALTER TABLE prueba ADD col3 integer NOT NULL CONSTRAINT c1 UNIQUE
- ALTER TABLE prueba ADD CONSTRAINT col3 UNIQUE (cla)
- ALTER TABLE prueba DROP COLUMN col3

Añadir una Columna

- La nueva columna aparecerá en el último lugar de la tabla. No se puede especificar el orden.
- Puede añadir o modificar columnas, pero no eliminarlas de una tabla.
- Si la tabla ya contiene registros al añadir una nueva columna, ésta se inicializará con valores nulos para todos los registros.
- Puede definir una columna NOT NULL sólo si la tabla está vacía.

Modificar una Columna

- Puede cambiar el tipo de datos de una columna, su tamaño y valor por defecto
- Si cambia el valor por defecto, afectará sólo a sucesivas inserciones en la tabla.

DROP TABLE

Elimina una tabla (datos y estructura) y sus índices. No se puede hacer Rollback de esta sentencia.

- Se borra estructura, datos e índices de la tabla. Borrado Físico.
- No se puede hacer Rollback de la sentencia.
- Sólo el propietario de la tabla u otro usuario con el permiso DROP ANY TABLE puede eliminar una tabla.

drop table prueba

RENAME

Cambia el nombre de una tabla, vista, secuencia o sinónimo.

- Debe ser el propietario del objeto.

- Permite cambiar el nombre de una tabla, vista, secuencia o sinónimo.

La sintaxis es la siguiente:

RENAME {tabla | vista | sinónimo} to nuevoNombre ;

RENAME prueba to Nueva

Esta sentencia cambiará el nombre prueba por el nuevo, y a partir de este momento cualquier acceso al objeto por el nombre antiguo será respondido con un mensaje de error.

Conviene resaltar la diferencia entre el comando `SYNONYM` y el comando `RENAME`. Mientras que el primero mantiene el nombre original para acceder al objeto, el segundo elimina ese primer nombre sustituyéndolo por el nuevo.

Truncar tablas

TRUNCATE Borra Todos los Registros de una Tabla

TRUNCATE TABLE Nombre_Tabla [REUSE STORAGE]

La Opción REUSE STORAGE se utiliza para rehusar el espacio ocupado por la tabla, (Serie tema para un curso de DBA Avanzado)

- No se puede hacer Rollback de la sentencia.
- Sólo el propietario de la tabla u otro usuario con el permiso DELETE TABLE puede eliminar una tabla.

Restricciones

Las restricciones de los datos se imponen para asegurarnos que los datos cumplen con una serie de condiciones predefinidas para cada tabla. Estas restricciones ayudan a conseguir la integridad de referencia: todas las referencias dentro de una BD son válidas y todas las restricciones se han cumplido.

Las restricciones se van a definir acompañadas por un nombre, lo que permitirá activarlas o desactivarlas según sea el caso; o también mezcladas en la definiciones de las columnas de la tabla. A continuación vamos a describir cada una de las restricciones mencionadas.

`NOT NULL`

Establece la obligatoriedad de que esta columna tenga un valor no nulo. Se debe especificar junto a la columna a la que afecta. Los valores nulos no ocupan espacio, y son distintos a 0 y al espacio en blanco. Hay que tener cuidado con los valores nulos en las operaciones, ya que `1 * NULL` es igual a `NULL`.

`UNIQUE`

Evita valores repetidos en una columna, admitiendo valores nulos. Oracle crea un índice automáticamente cuando se habilita esta restricción y lo borra al deshabilitarse.

`DEFAULT`

Establece un valor por defecto para esa columna, si no se le asigna ninguno.

`CHECK`

Comprueba que se cumpla una condición determinada al rellenar esa columna. Esta condición sólo debe estar construida con columnas de esta misma tabla.

`PRIMARY KEY`

Establece el conjunto de columnas que forman la clave primaria de esa tabla. Se comporta como única y obligatoria sin necesidad de explicitarlo. Sólo puede existir una clave primaria por tabla. Puede ser referenciada como clave ajena por otras tablas. Crea un índice automáticamente cuando se habilita o se crea esta restricción. En Oracle, los índices son construidos sobre árboles B^+.

`FOREIGN KEY`

Establece que el contenido de esta columna será uno de los valores contenidos en una columna de otra tabla maestra. Esta columna marcada como clave ajena puede ser NULL. No hay límite en el número de claves ajenas. La clave ajena puede ser otra columna de la misma tabla. Se puede forzar que cuando una fila de la tabla maestra sea borrada, todas las filas de la tabla detalle cuya clave ajena coincida con la clave borrada se borren también. Esto se consigue añadiendo la coletilla `ON DELETE CASCADE` en la definición de la clave ajena.

Ejemplos:

```
ALTER TABLE prueba ADD col3 integer NOT NULL UNIQUE
```

*Desactivar una restricción

La Opción DISABLE CONSTRAINT es utilizada para deshabilitar una restricción.

ALTER Table_Name DISABLE CONSTRAINT Constraint_NAME

- Sólo el propietario de la tabla u otro usuario con el permiso ALTER puede deshabilitar un Constraint.

*Activar una restricción

La Opción ENABLE CONSTRAINT es utilizada para habilitar una restricción.

ALTER Table_Name ENABLE CONSTRAINT Constraint_NAME

- Sólo el propietario de la tabla u otro usuario con el permiso ALTER puede habilitar un Constraint.

***Consulta al diccionario de datos**

```
select owner, count(owner) Numero
 from dba_objects
 group by owner
 order by Numero desc
```

Diccionario de datos (incluye todas las vistas y tablas de la Base de Datos)

select * from dictionary

select table_name from dictionary

Muestra los datos de una tabla especificada (en este caso todas las tablas que lleven la cadena "EMPLO"

select * from ALL_ALL_TABLES where upper(table_name) like '%EMPLO%'

Tablas propiedad del usuario actual

select * from user_tables
Todos los objetos propiedad del usuario conectado a Oracle
select * from user_catalog

*Visualización de restricciones

Roles y privilegios por roles:

```
select * from role_sys_privs
```

Reglas de integridad y columna a la que afectan:

select constraint_name, column_name from sys.all_cons_columns

Tablas de las que es propietario un usuario, en este caso "HR":

```
SELECT table_owner, table_name from sys.all_synonyms where
table_owner like 'HR'
```

Otra forma más efectiva (tablas de las que es propietario un usuario):

```
SELECT DISTINCT TABLE_NAME
FROM ALL_ALL_TABLES
WHERE OWNER LIKE 'HR'
```

UNION.-

Combina los resultados de dos consultas. Las filas duplicadas que aparecen se reducen a una fila única.

UNION ALL.-

Como la anterior pero aparecerán nombres duplicados.

INTERSEC.-

Devuelve las filas que son iguales en ambas consultas. Todas las filas duplicadas serán eliminadas.

MINUS.-

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

Reglas para la utilización de operadores de conjunto:

- Las columnas de las dos consultas se relacionan en orden, de izquierda a derecha.
- Los nombres de columna de la primera sentencia "Select" no tiene porque ser los mismos que los nombres de la segunda.

- Los "Select" necesitan tener el mismo numero de columnas.
- Los tipos de datos deben coincidir, aunque la longitud no tiene que ser la misma.

VISTAS

Concepto de Vista.-

- Una vista es una tabla lógica basada en una tabla u otra vista.
- No contiene datos en sí misma, pero es como una ventana a través de la cual se pueden ver o cambiar los datos de las tablas.
- Podemos representar con ellas subconjuntos lógicos o combinaciones de datos.
- Las tablas sobre las cuales se basa una vista se llaman tablas base.
- Se almacenan en el Diccionario de Datos, USER_VIEWS.

¿Por qué usar Vistas?

- Para restringir el acceso a la B.D.
- Para realizar consultas complejas de manera fácil.
- Para obtener una independencia de los datos
- Para presentar diferentes vistas de los mismos datos.

CREATE VIEW vista [({columna ,}+] AS consulta ;

- FORCE: Crea la vista sin importar que la tabla base exista o no.
- WITH CHECK OPTION: Especifica que solamente las filas accesibles a la vista pueden ser insertadas o actualizadas.
- CONSTRAINT: Nombre asignado a la restricción CHECK OPTION.
- WITH READ ONLY: Asegura que ninguna operación DML pueda realizarse sobre esta vista.

Ejemplos creación de Vista

```
Create view emp_vista03 as select empno, ename, job from emp where
Deptno=10;

create view emp_vista04 as select empno Num_Empleado, ename Nombre,
sal Salario from emp where deptno=30;

create view emp_vista05 as select empno Num_Empleado, ename Nombre,
sal Salario from emp where deptno=30 WITH CHECK OPTION;
```

Borrado de una Vista

DROP VIEW vista ;

```
drop view  emp_vista05;
```

- Al borrar una vista no perderá los datos, porque la vista está basada en tablas subyacentes de la B.D.
- Únicamente el creador o un usuario con el privilegio DROP ANY VIEW puede eliminar una vista.

Limitaciones DML en Vistas

- Se pueden realizar operaciones DML sobre vistas simples.
- No se puede eliminar una fila si la vista contiene Funciones de grupo, una cláusula GROUP BY o el comando DISTINCT.
- No es posible modificar datos en la vista si contiene cualquiera de las condiciones anteriores, columnas definidas por expresiones o la pseudos- columna ROWNUM
- No se puede agregar datos si la vista contiene cualquiera de las condiciones anteriores o cualquier columna NOT NULL no incluida por la vista (tabla base).

***Creación de Secuencias**

A menudo es preciso generar números en forma ordenada para implementar, por ejemplo, una clave primaria en una tabla o garantizar que esos números no se repiten y van siempre en un orden predefinido por el desarrollador (no necesariamente secuenciales).

La forma tradicional de efectuar lo anterior sería almacenar el último número utilizado en un registro especial, bloquearlo, obtener el próximo valor, actualizar el registro, desbloquearlo y utilizar el número. Sin embargo, para eso Oracle implementa los objetos denominadas secuencias, que permiten hacer lo anterior de manera transparente para el usuario.

Cuando se define una secuencia se deben indicar, como mínimo, el valor de partida (valor mínimo) y el incremento.

La sintaxis de creación de una secuencia es la siguiente:

CREATE SEQUENCE nombre_secuencia
INCREMENT BY numero_incremento
START WITH numero_por_el_que_empezara
MAXVALUE valor_maximo | NOMAXVALUE
MINVALUE valor_minimo | NOMINVALUE
CYCLE | NOCYCLE
ORDER | NOORDER

Los parámetros significan lo siguiente:

- Increment by: Indica la cantidad de incremento de la secuencia.
- Start with: Es el valor de partida de la secuencia.
- Minvalue: Indica cuál será el valor mínimo de la secuencia.
- Maxvalue: Corresponde al valor máximo que puede tomar la secuencia.
- Nocycle: Es el valor por defecto para establecer si la secuencia deberá comenzar nuevamente a generar valores una vez que ha alcanzado el máximo.

Por ejemplo, si queremos crear una secuencia que empiece en 100 y se incremente de uno en uno utilizaremos la siguiente consulta SQL:

```
CREATE SEQUENCE incremento_id_cliente
INCREMENT BY 1
```

START WITH 100Para utilizar la secuencia, en primer lugar, crearemos una tabla de prueba (para insertar un registro y comprobar que la secuencia anterior funciona correctamente):

```
create table clientes (
codigo number not null primary key,
nombre varchar2(100) unique not null,
cif varchar2(15) unique,

fechaalta date)
```

Para utilizar la secuencia creada en una inserción de fila:

```
insert into clientes values (
incremento_id_cliente.NextVal,
'AjpdSoft',
'11225522F', sysdate)
```

Realizamos otra inserción para comprobar que el incremento es de 1:

```
insert into clientes values (
incremento_id_cliente.NextVal,
'Otro cliente',
'00000G', sysdate);
```

Como se puede observar en el ejemplo anterior, para obtener el siguiente valor de la secuencia almacenada se utiliza el comando: nombre_secuencia.NextVal.

Para comprobar que la secuencia ha funcionado en los inserts anteriores hacemos un SELECT a la tabla "clientes":

```
select * from clientes;
```

El resultado de este SELECT debe ser de dos registro con "codigo" 100 y 101:

- Creación de índices

Con la finalidad de acceder a los registros con mayor rapidez se utiliza la creación de índices y al igual que las tablas se crean mediante CREATE

```
CREATE INDEX Index_Name ON Table_Name USING Column_Name | UNIQUE
```

Ej: `CREATE INDEX IDX_Por_Nombre ON emp USING Noemp`

Creación de un Sinónimos

CREATE SYNONYM sinónimo FOR [usuario.]{tabla | vista} ;

Una primera utilidad de los sinónimos es la posibilidad de independizar las aplicaciones de los nombres físicos de las tablas que manejan. Asi, las aplicaciones harán referencia a un sinónimo de tabla, que en cada caso puede estar asociado a una tabla distinta.

Otra utilidad es la posibilidad de que un usuario acceda a las tablas de otro usuario como si fueran suyas, siempre que tenga permiso para hacerlo, si al definir el sinónimo incluye el nombre del usuario en la denominación de la tabla. Así si el usuarioA tiene permiso para leer el contenido de la tabla `emp` del usuarioB, entonces desde la ejecución de la sentencia `CREATE SYNONYM plantilla FOR usuarioB.emp` verá la tabla `usuarioB.emp` como `plantilla`.

create synonym pepe for prueba

- Simplifican el acceso a los objetos al crear otro nombre para un objeto (sinónimo).
- Hacen referencia a una tabla propia o de otro usuario.
- Permite acortar la longitud de los nombre de los objetos a la vez que elimina la necesidad de cualificar el objeto con un esquema.
- El DBA puede crear un sinónimo público accesible a todos los usuarios.

Eliminación de Sinónimos

```
drop synonym pepe
```

Sólo el DBA puede eliminar un sinónimo público.

Módulo 5: Subconsultas avanzadas

- - Subconsultas:
  La subconsulta se ejecuta una vez y antes de la consulta principal.
- El resultado de ella es usado por la consulta principal externa.

Guía Uso de Subconsultas

- Encierre las subconsultas entre paréntesis.
- No añada una cláusula ORDER BY a una subconsulta.
- Utilice operadores a nivel de fila para subconsultas que devuelvan solo una fila MONOREGISTRO.
- Utilice operadores que actúan sobre varios registros para subconsultas que devuelven más de una fila MULTIREGISTRO.

Subconsultas Mono- registro

- Devuelven un único registro.
- Se utilizan operadores de comparación (=, >, >=, <, <= y <>).

Ejemplo:

```
select ename, job, sal  from emp where job = (select job from emp
where empno=7369) and     sal >(select sal from emp where empno=7876);
```

Subconsultas Multi-registro

- Devuelven más de un registro
- Se utilizan comparadores multiregistro:
  - IN TRUE si se encuentra en la lista.
  - ANY (y sinónimo SOME) TRUE si la condición se cumple con algún registro de la lista devuelta por la subconsulta.
  - ALL TRUE si la condición se cumple con todos los registros de la lista devuelta por la subconsulta.
- El operador NOT puede ser utilizado con los operadores IN, ANY y ALL.

Ejemplo subc. Multi-registro

```
select ename, sal, deptno from emp where sal in(select min(sal)from
emp group by deptno);

select ename, sal, job from emp where sal < any (select sal from emp
where job = 'CLERK');

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

Subconsultas. en cláusula FROM

- Puede utilizar una subconsulta en una cláusula FROM de una sentencia SELECT:

```
select a.ename, a.sal, a.deptno, b.salavg from emp a, (select deptno,
avg(sal)salavg from emp group by deptno)b where a.deptno=b.deptno and
a.sal > b.salavg;
```

Este ejemplo muestra los nombres, salarios, núm. Departamentos y media de salarios, de todos los empleados que cobran más que la media de salarios de su departamento.

Control de Datos (DCL)

Estas sentencias se completan con los comandos de control de transacción (DCL), las cuales aseguran la consistencia de los datos.

o COMMIT
Finaliza la transacción actual haciendo que todos los cambios pendientes pasen a ser permanentes.

o ROLLBACK
Finaliza la transacción en curso descartando todos los cambios pendientes.

o SAVEPOINT

Establece una "marca" dentro de la transacción en curso, usada por COMMIT o ROLLBACK.

Concepto sobre PL/SQL

Aunque esto seria objeto de otro curso, vamos a introducir un poco sobre PL/SQL

Existen procedimientos Anonimos y Almacenados, Funciones Triguers y Paquetes, solamente nos limitaremos a presentar un ejemplo de un PL Anonimo:

```
BEGIN
   FOR i IN(SELECT ename, sal from Emp) LOOP
          DBMS_OUTPUT.PUT_LINE('Nombre: '||Ename||' Salario: ||Sal);
  END LOOP END;
```

Prácticas y Ejercicios SQL - ORACLE

Objetivo:

Esta practica llevara al estudiantes a un nivel dominante de SQL en cualquier motor de bases datos, al finalizar esta practica el estudiante será capaz de manejar un 99% el estándar SQL, Select, Update, Delete, Drop, Insert, Sequence, Vista, Table, Synonym , Group By, Order By, Sub-Consultas, Vista, Index, Alter, Commit, Rollback, Funciones etc.

1. **Mostrar código, nombre y sueldo de los empleados del departamento 30.**

```
      SELECT empno, ename, sal
 FROM emp
 WHERE deptno = 30;
```

2. **Mostrar nombre, cargo y sueldo para todos los empleados excepto los gerentes.**

```
SELECT ename, job, sal
 FROM emp
    WHERE job != 'MANAGER';
```

3. **Mostrar nombre, cargo y código de departamento para los empleados contratados entre el 1-ENE-1982 y el 1-ENE-1983.**

```
SELECT ename, job, deptno, hiredate
 FROM emp
 WHERE hiredate BETWEEN '01-JAN-82' AND '01-JAN-83';
```

4. **Mostrar nombre, cargo y código de departamento para todos los oficinistas y analistas.**

```
SELECT ename, job, deptno
 FROM emp
    WHERE job IN ('CLERK', 'ANALYST');
```

5. **Mostrar nombre, cargo, código de departamento y fecha de ingreso para los empleados cuyo nombre comienza con M.**

```
SELECT ename, job, deptno, hiredate
 FROM emp
 WHERE ename LIKE 'M%';
```

6. **Mostrar nombre, cargo, código de departamento y fecha de ingreso para los empleados cuyo nombre comienza por J, continúa con 2 letras y termina en ES.**

```
SELECT ename, job, deptno, hiredate
 FROM emp
    WHERE ename LIKE 'J_ _ES';
```

7. **Mostrar nombre, cargo y sueldo para los gerentes que ganen $1500 o más, así como para todos los vendedores.**

```
SELECT ename, job, sal
 FROM emp
 WHERE sal >= 1500 AND job='MANAGER' OR job='SALESMAN';
```

8. **Mostrar nombre, cargo y sueldo para los gerentes y los vendedores que ganen $1500 o más.**

```
SELECT ename, job, sal
 FROM emp
 WHERE sal>=1500 AND (job='MANAGER' OR job='SALESMAN');
```

9. **Mostrar sueldo, cargo y nombre de los empleados del departamento 10 ordenados según el sueldo en forma ascendente.**

```
SELECT sal, job, ename
 FROM emp
 WHERE deptno = 10
 ORDER BY sal;
```

10. **Mostrar sueldo, cargo y nombre de los empleados del departamento 10 ordenados según el sueldo en forma descendente.**

```
SELECT sal, job, ename
 FROM emp
 WHERE deptno = 10
 ORDER BY sal DESC;
```

11. **Mostrar nombre, cargo y sueldo de los trabajadores del departamento 30. Ordenar el resultado por cargo. Si existe más de un empleado con el mismo cargo, ordenar por sueldo en orden descendente, y finalmente por nombre.**

```
SELECT ename, job, sal
 FROM emp
 WHERE deptno = 30
 ORDER BY job, sal DESC, ename;
```

12. **Mostrar nombre, sueldo y cargo de los empleados del departamento 10, en orden ascendente de sueldos y usando la posición de la columna.**

```
SELECT ename, sal, job
 FROM emp
 WHERE deptno = 10
 ORDER BY 2;
```

13. **Mostrar los nombres de las tablas del usuario.**

```
SELECT table_name
 FROM user_tables;
```

14. **Mostrar el nombre, salario, comisión y total de pagos para todos los vendedores cuya comisión sea mayor que el 25% de su salario.**

```
SELECT ename, sal, comm, sal+comm
 FROM emp
 WHERE job = 'SALESMAN'
 AND comm > .25*sal
 ORDER BY 4;
```

15. **Mostrar el nombre, cargo, sueldo, comisión e ingresos totales para los empleados del departamento 30.**

```
SELECT ename, job, sal, comm, sal+comm
 FROM emp
 WHERE deptno=30;

SELECT ename, job, sal, comm, NVL(sal,0) + NVL(comm,0)
 FROM emp
 WHERE deptno = 30;
```

16. **Mostrar nombre, sueldo, comisión e ingresos totales para todos los vendedores. Ordenar por ingresos totales.**

```
SELECT ename, sal, comm, 12*(sal+comm) TOTAL
 FROM emp
 WHERE job = 'SALESMAN'
 ORDER BY 12*(sal+comm);
```

17. **Mostrar nombre, sueldo, comisión y salario anual más un mes de comisión ingresos totales para todos los vendedores. Ordenar por esta última columna.**

```
SELECT ename, sal, comm, 12*sal+comm TOTAL
 FROM emp
 WHERE job = 'SALESMAN'
 ORDER BY 4;
```

18. **Listar el nombre, sueldo mensual, sueldo diario (basado en un mes de 22 días hábiles), y sueldo diario redondeado al dólar entero más cercano, para Allen y Jones.**

```
SELECT ename, sal, sal/22, ROUND(sal/22,0)
 FROM emp
 WHERE ename IN ('ALLEN', 'JONES');
```

19. **Listar el nombre, sueldo mensual, sueldo diario (basado en un mes de 22 días hábiles), y sueldo diario truncado al dólar entero más cercano hacia abajo, para Allen y Jones.**

```
SELECT ename, sal, sal/22, TRUNC(sal/22,0)
 FROM emp
 WHERE ename IN ('ALLEN', 'JONES');
```

20. **Mostrar el nombre, fecha de ingreso y fecha de terminación del período de prueba (90 días) para los empleados del departamento 10.**

```
SELECT ename, hiredate, hiredate + 90 "PRUEBA"
 FROM emp
 WHERE deptno = 10;
```

21. **Mostrar el nombre, fecha de ingreso y fecha de un aumento de sueldo a los 6 meses de la contratación para los empleados del departamento 10.**

```
SELECT ename, hiredate, ADD_MONTHS(hiredate,6)
"AUMENTO"
 FROM emp
    WHERE deptno = 10;
```

22. **Mostrar el nombre y número de semanas de trabajo para los empleados del departamento 20.**

```
SELECT ename, (SYSDATE - hiredate)/7 SEMANAS
 FROM emp
 WHERE deptno = 20;
```

23. **Mostrar las fechas de ingreso para los empleados del departamento 20 en formato DD de Mes YYYY.**

```
SELECT TO_CHAR(hiredate, 'fmDD " de " Month YYYY) "Ingreso"
 FROM emp
 WHERE deptno = 30;
```

24. **Mostrar los departamentos y su localización, unidos por un guión.**

```
SELECT dname || ' - ' || loc DEPARTAMENTOS
 FROM dept;
```

25. **Mostrar el nombre y cargo de todos los empleados llamados Ward, con mayúscula inicial.**

```
SELECT INITCAP(ename) NOMBRE, job
 FROM emp
 WHERE UPPER(ename) = 'WARD';
```

26. **Mostrar las 5 primeras letras del nombre del departamento, y la localidad entera.**

```
    SELECT SUBSTR(dname, 1, 5) DEPT, loc
 FROM dept;
```

27. **Mostrar la comisión para los empleados del departamento 30, usando signo de pesos, comas en los miles y dos cifras decimales.**

```
SELECT ename EMPLEADO, TO_CHAR(comm, '$9,990.99') COMISION
 FROM emp
 WHERE deptno = 30;
```

28. **Mostrar el nombre, salario mensual y comisión para los vendedores; incluir una columna que muestre, entre el salario y la comisión, el más grande de los dos.**

```
SELECT ename, sal, NVL(comm,0), GREATEST(sal, comm)
 FROM emp
 WHERE job = 'SALESMAN';
```

29. **Mostrar para los vendedores, el promedio de salario, el salario más alto y el más bajo.**

```
SELECT AVG(sal), MAX(sal), MIN(sal)
 FROM emp
 WHERE job = 'SALESMAN';
```

30. **Mostrar el número de filas de la tabla EMP, y el número de empleados cuya comisión no es nula.**

```
SELECT COUNT(*) EMPLEADOS, COUNT(comm) "CON COMISION"
 FROM emp;
```

31. **Mostrar cada departamento, seguido de su número de empleados.**

```
SELECT deptno, COUNT(*)
FROM emp
 GROUP BY deptno;
```

32. **Mostrar el número de empleados para cada cargo en cada departamento.**

```
SELECT deptno, job, COUNT(*)
 FROM emp
 GROUP BY deptno, job;
```

33. Mostrar el salario promedio anual para todos los cargos que tengan más de dos empleados.

```
SELECT job, 12*AVG(sal)
 FROM emp
 GROUP BY job
 HAVING COUNT(*) > 2;
```

34. Mostrar los departamentos y el total de nómina para esos departamentos, teniendo en cuenta que dicho total debe exceder $8,000 y que se debe excluir del total a todo el personal de oficina. La lista debe ordenarse por este total.

```
SELECT deptno, SUM(sal)
 FROM emp
 WHERE job != 'CLERK'
 GROUP BY deptno
 HAVING SUM(sal) > 8000
 ORDER BY SUM(sal);
```

35. Vamos ha crear una tabla.

```
CREATE TABLE gnuEmpleo (
 id NUMBER(4),
 gnombre CHAR(10),
 gtrabajo CHAR(9),
 idmanger NUMBER(4),
 gnfecha DATE,
 gsalario NUMBER(7,2),
 gcomission NUMBER(7,2),
    gdept NUMBER(2) NOT NULL);
```

36. Vamos a crear una tabla con la estructura de otra ya creada.

```
CREATE TABLE GNU
 AS SELECT empno, ename, hiredate
 FROM emp
 WHERE deptno = 10;
```

37. La siguiente orden crea una tabla vacía basada en la estructura de otra (se especifica algo en el WHERE que sea siempre falso):

```
CREATE TABLE vacia_como_emp
 AS SELECT *
 FROM emp
 WHERE 1 = 2;
```

38. Añadir una columna a la tabla HDATES:

```
ALTER TABLE hdates
 ADD (manager NUMBER(4));
```

39. Incrementar el ancho de la columna ENAME de 10 a 18 caracteres:

```
ALTER TABLE hdates
 MODIFY (ename CHAR(18));
```

40. Para borrar la tabla.

```
DROP TABLE vacia_como_emp;
```

41. Crear una vista que contenga el código, nombre y cargo de todos los empleados del departamento 10.

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

42. Borrar la vista que se acabó de crear.

```
DROP VIEW empvu10;
```

43. CREACION DE UNA VISTA CON ALIAS EN LAS COLUMNAS:

```
CREATE VIEW empvu10 (id_number, employee, title)
 AS SELECT empno, ename, job
 FROM emp
 WHERE deptno = 10;

CREATE VIEW salvu10
AS SELECT empno, ename, sal*12 ANNUAL_SALARY
 FROM EMP WHERE deptno = 10;
```

44. Crear una vista que contenga todas las columnas de la tabla EMP para el departamento 20, con opción de chequeo:

```
CREATE VIEW empvu20
 AS SELECT *
 FROM emp
 WHERE deptno=20
 WITH CHECK OPTION;
```

Nota: Ahora trate de actualizar la vista.

45. Dar a todos privilegios sobre la tabla EMP.

```
GRANT ALL
 ON emp
 TO PUBLIC;
```

46. Dar privilegio de SELECT a Jones sobre la tabla DEPT.

```
GRANT SELECT
 ON dept
 TO jones;
```

47. Dar privilegio de SELECT a Jones sobre la tabla DEPT, y darle a Jones la posibilidad de dar ese privilegio a otros usuarios.

```
GRANT SELECT
 ON dept
 TO jones
 WITH GRANT OPTION;
```

48. Crear un sinónimo DATES para la tabla HDATES de Ward.

```
CREATE SYNONYM dates
 FOR ward.hdates;
```

49. Crear un índice simple sobre la columna ENAME de EMP.

```
CREATE INDEX i_emp_name
 ON emp(ename);
```

50. Consulta en la que se emplea el índice:

```
SELECT *
 FROM emp
 WHERE ename = 'SMITH';
```

51. Consulta en la que no se emplea el índice:

```
SELECT *
 FROM emp;
```

52. Creación de un índice concatenado (sobre varias columnas):

```
CREATE INDEX i_empno_ename
 ON emp(empno,ename);
```

53. Borrar el índice recién creado.

```
DROP INDEX i_emp_ename;
```

54. Crear una secuencia simple para el campo EMPNO iniciando con el número 8000.

```
CREATE SEQUENCE s_emp_empno START WITH 8000;
```

55. Crear una secuencia simple para DEPTNO, iniciando con el número 100.

```
CREATE SEQUENCE s_dept_deptno START WITH 100;
```

56. Mostrar el siguiente número disponible en la secuencia S_EMP_EMPNO, e incrementar su valor:

```
SELECT s_emp_empno.nextval
 FROM dual;
```

57. Mostrar el número actual de la secuencia S_DEPT_DEPTNO, e incrementar su valor:

```
SELECT s_dept_deptno.currval
 FROM dual;
```

58. Borrar la secuencia creada para la tabla DEPT:

```
DROP SEQUENCE s_dept_deptno;
```

59. Insertar el departamento de Finanzas, localizado en Los Angeles, en la tabla DEPT con código 50.

```
INSERT INTO dept
 VALUES (50, 'FINANCE', 'LOS ANGELES');
```

60. Insertar valores en todas las columnas de la tabla EMP para un nuevo empleado.

```
INSERT INTO emp
 VALUES(1234, 'EMMETT', 'SALESMAN', 7698, SYSDATE, 2000,
 NULL, 30);
```

61. Insertar valores sólo en algunas columnas de la tabla EMP para un nuevo empleado.

```
INSERT INTO emp (empno, ename, hiredate, sal, deptno)
 VALUES(S_EMP_EMPNO.nextval,'LERNER','01-JAN-92',2000, 30);
```

62. Insertar datos para un departamento nuevo en la tabla DEPT, preguntando los valores.

```
INSERT INTO dept
VALUES (&DEPTNO, '&DNAME', '&LOC');
```

63. Cambiar al empleado con código 7566 a un puesto de vendedor en el departamento.

```
UPDATE emp
 SET job = 'SALESMAN', deptno = 30
 WHERE empno = 7566;
```

64. Transferir al empleado 7788 (Scott) a Ventas y darle un aumento de sueldo del 5%.

```
UPDATE emp
 SET job = 'SALESMAN', sal = sal*1.05, deptno = 30 WHERE empno = 7788;
```

65. Cambiar el nombre del cargo de vendedor: SALESMAN por SALES, en todas las filas donde esté.

```
UPDATE emp
 SET job = 'SALES'
 WHERE job = 'SALESMAN';
```

66. Deshacer el cambio anterior.

```
UPDATE emp
 SET job = 'SALESMAN'
 WHERE job = 'SALES';
```

67. Borrar al empleado con código 1234 (Emmett).

```
DELETE FROM emp
 WHERE empno = 1234;
```

68. Borrar al empleado con código 1234 (Emmett).

```
DELETE FROM emp
 WHERE empno = 1234;
```

69. Borrar TODOS los empleados de la tabla EMP (¡OJO!).

```
DELETE FROM emp;
```

Realice, Analice y determine el resultado de las siguientes sentencias.

```
create table S ( S# char(3), SNAME Char(10), Status Int, City
char(10));
create table P (P# char(3), PNAME Char(10), Color Char(10), Weight
Int, City
Char(10));
create table SP (S# char(3), P# Char(3), Qty Int);
create table J ( J# char(3), JNAME Char(10), City char(10));
create table SPJ (S# char(3), P# Char(3), J#Char(3), Qty Int);
insert into S values ('S1', 'Smith', 20, 'London');
insert into S values ('S2', 'Jones', 10, 'Paris');
insert into S values ('S3', 'Blake', 30, 'Paris');
insert into S values ('S4', 'Clark', 20, 'London');
insert into S values ('S5', 'Adams', 30, 'Athens');
insert into P values ('P1', 'Nut', 'Red', 12, 'London');
insert into P values ('P2', 'Bolt', 'Green', 17, 'Paris');
insert into P values ('P3', 'Screw', 'Blue', 17, 'Rome');
insert into P values ('P4', 'Screw', 'Red', 14, 'London');
insert into P values ('P5', 'Cam', 'Blue', 12, 'Paris');
insert into P values ('P6', 'Cog', 'Red', 19, 'London');
insert into J values ('J1', 'Sorter', 'Paris');
insert into J values ('J2','Punch', 'Rome');
insert into J values ('J3', 'Reader', 'Athens');
insert into J values ('J4', 'Console', 'Athens');
insert into J values ('J5', 'Collator', 'London');
insert into J values ('J6', 'Terminal', 'Oslo');
insert into J values ('J7', 'Tape', 'London');
insert into SP values ('S1', 'P1', 300);
insert into SP values ('S1', 'P2', 200);
insert into SP values ('S1', 'P3', 400);
insert into SP values ('S1', 'P4', 200);
insert into SP values ('S1', 'P5', 100);
insert into SP values ('S1', 'P6', 100);
insert into SP values ('S2', 'P1', 300);
insert into SP values ('S2', 'P2', 400);
insert into SP values ('S3', 'P2', 200);
insert into SP values ('S4', 'P2', 200);
insert into SP values ('S4', 'P4', 300);
insert into SP values ('S4', 'P5', 400);
insert into SPJ values ('S1', 'P1', 'J1', 200);
insert into SPJ values ('S1', 'P1', 'J4', 700);
insert into SPJ values ('S2', 'P3', 'J1', 400);
```

```
insert into SPJ values ('S2', 'P3', 'J2', 200);
insert into SPJ values ('S2', 'P3', 'J3', 200);
insert into SPJ values ('S2', 'P3', 'J4', 500);
insert into SPJ values ('S2', 'P3', 'J5', 600);
insert into SPJ values ('S2', 'P3', 'J6', 400);
insert into SPJ values ('S2', 'P3', 'J7', 800);
insert into SPJ values ('S2', 'P5', 'J5', 100);
insert into SPJ values ('S3', 'P3', 'J1', 200);
insert into SPJ values ('S3', 'P4', 'J2', 500);
insert into SPJ values ('S4', 'P6', 'J3', 300);
insert into SPJ values ('S5', 'P2', 'J2', 200);
commit;
create unique index SIX1 on S ( S# );
create unique index PIX1 on P ( P# );
create unique index SPIX1 on SP ( S#, P# );
create unique index SPJIX1 on SPJ ( S#, P#, J# );
create index PCOLOR on P(Color);

1- select distinct P# from SPJ group by
p#,j# having avg(qty) > 320;

2- select s.* from S, SP where s.s#=Sp.P#
and status > 20 and qty > 40
union
Select s.* from s,sp, p where s.s#=sp.s# and sp.p#=p.p# and p.col-
or='Red';

3-select s.* from s, sp where s.s# =
sp.s# and status > 20 and city in (select city from j where j# =
'J1' or j# = 'J2');

4-SELECT +3 FROM DUAL;

5-SELECT -4 FROM DUAL;
6-SELECT SAL * 5 FROM EMP;

7-SELECT SAL + 200 FROM EMP;

8-SELECT SAL - 100 FROM EMP;

9-SELECT CONCAT (CONCAT (ENAME, ' is a '),job) FROM EMP WHERE SAL >
2000;

10-SELECT ENAME "Employee" FROM EMP WHERE SAL = 1500;

11-SELECT ENAME FROM EMP WHERE SAL ^= 5000;

12- SELECT ENAME "Employee", JOB "Title" FROM EMP WHERE SAL > 3000;

13-SELECT * FROM PRICE WHERE MINPRICE < 30;

14- SELECT * FROM PRICE WHERE MINPRICE >= 20;

15-SELECT ENAME FROM EMP WHERE SAL <= 1500;

16-SELECT * FROM EMP WHERE ENAME IN ('SMITH', 'WARD');

17-SELECT * FROM DEPT WHERE LOC = SOME ('NEW YORK','DALLAS');

18-SELECT * FROM DEPT WHERE LOC NOT IN ('NEW YORK', 'DALLAS');
```

```
19-SELECT * FROM emp WHERE sal >= ALL (1400, 3000);

20-SELECT ENAME, JOB FROM EMP WHERE SAL BETWEEN 3000 AND 5000;

21-SELECT * FROM EMP WHERE EXISTS (SELECT
ENAME FROM EMP WHERE MGR IS NULL);

22-SELECT * FROM EMP WHERE ENAME LIKE '%E%';

23-SELECT * FROM EMP WHERE COMM IS NOT NULL AND SAL > 1500;

24-SELECT * FROM EMP WHERE NOT (job IS NULL);

25-SELECT * FROM EMP WHERE NOT (sal BETWEEN 1000 AND 2000);

26-SELECT * FROM EMP WHERE job='CLERK' AND deptno=10;

27-SELECT * FROM emp WHERE job='CLERK' OR deptno=10;

28- SELECT * FROM (SELECT ENAME FROM EMP WHERE JOB = 'CLERK';
UNION
SELECT ENAME FROM EMP WHERE JOB = 'ANALYST');

29- SELECT * FROM (SELECT SAL FROM EMP WHERE JOB = 'CLERK'
UNION
SELECT SAL FROM EMP WHERE JOB = 'ANALYST');

30-SELECT * FROM (SELECT SAL FROM EMP WHERE
JOB = 'PRESIDENT'
MINUS
SELECT SAL FROM EMP WHERE JOB = 'MANAGER');

31-SELECT ENAME, DNAME FROM EMP, DEPT
WHERE DEPT.DEPTNO = EMP.DEPTNO (+);

32-SELECT ASCII('Q') FROM DUAL;

33-SELECT AVG(SAL) FROM EMP;

34-SELECT AVG (DISTINCT DEPTNO) FROM EMP;

35-SELECT AVG (ALL DEPTNO) FROM EMP;

36- SELECT CASE JOB
WHEN 'PRESIDENT' THEN 'The Honorable'
WHEN 'MANAGER' THEN 'The Esteemed'
ELSE 'The good'
END,
ENAME
FROM EMP;

37-SELECT CHR(68)||CHR(79)||CHR(71) "Dog" FROM DUAL;

38- SELECT CONCAT( CONCAT(ename, ' is a '), job) "Job" FROM emp WHERE
empno = 7900;
39-SELECT COUNT(*) "Total" FROM emp;

40- SELECT COUNT(job) "Count" FROM emp;

41-SELECT COUNT(DISTINCT job) "Jobs" FROM emp;
```

```
42-SELECT COUNT (ALL JOB) FROM EMP;

43-SELECT CURRENT_DATE FROM DUAL;

44-SELECT CURRENT_TIMESTAMP FROM DUAL;

45- SELECT TO_CHAR (CURRENT_TIMESTAMP, 'HH24:MM:SS, Day, Month, DD,
YYYY')FROM DUAL;

46-SELECT DECODE (deptno, 10, 'ACCOUNTING',
20, 'RESEARCH',
30, 'SALES',
40, 'OPERATIONS',
'NONE')
FROM DEPT;

47-SELECT FLOOR(15.7) "Floor" FROM DUAL;

48-SELECT INITCAP('the soap') "Capitals" FROM DUAL;

49-SELECT INSTR('CORPORATE FLOOR','OR',3,2) "Instring" FROM DUAL;

50-SELECT INSTRB('CORPORATE FLOOR','OR',5,2) "Instring in bytes" FROM
DUAL;

51-SELECT CURRENT_DATE - INTERVAL '8' MONTH FROM DUAL;

52-SELECT TO_CHAR (INTERVAL '6' DAY * 3) FROM DUAL;

53-SELECT LAST_DAY (SYSDATE) FROM DUAL;

54- SELECT SYSDATE,
LAST_DAY(SYSDATE) "Last",
LAST_DAY(SYSDATE) - SYSDATE "Days Left"
FROM DUAL;

55-SELECT LENGTH('CANDIDE') "Length in characters" FROM DUAL;

56-SELECT LPAD('Page1',15,'*.') "LPAD example" FROM DUAL;

57-SELECT LTRIM ('xyxXxyLAST WORD','xy') "LTRIM example" FROM DUAL;

58-SELECT MAX(SAL) FROM EMP;

59-SELECT MIN(SAL), MAX(SAL) FROM EMP;

60-SELECT NEXT_DAY('15-MAR-92','TUESDAY') "NEXT DAY" FROM DUAL;

61-SELECT ename, NVL(TO_CHAR(COMM),'NOT APPLICABLE') "COMMISSION"
FROM emp
WHERE deptno = 30;

62-SELECT sal+NVL(comm, 0) FROM EMP;

63-SELECT REPLACE('JACK and JUE','J','BL') "Changes" FROM DUAL;

64-SELECT ROUND(TO_DATE('27-OCT-92', 'DD-MONTH-RR'),'YEAR') "FIRST OF
THE YEAR" FROM DUAL;

 SELECT ROUND(TO_DATE('27-OCT-92'),'YEAR')
"FIRST OF THE YEAR" FROM DUAL;
```

```
65-SELECT ROUND (54.339, 2) FROM DUAL;

66-SELECT RPAD('ename',12,'ab') "RPAD example"
FROM emp
WHERE ename = 'TURNER';

67-SELECT STDDEV(sal) "Deviation" FROM emp;

68-SELECT SUBSTR('ABCDEFG',3,4) "Subs" FROM DUAL;

69-SELECT SUBSTRB('ABCDEFG',5,4) "Substring with bytes" FROM DUAL;

70-SELECT deptno, SUM(sal) TotalSalary FROM emp GROUP BY deptno;

71-SELECT TO_CHAR(SYSDATE, 'MM-DD-YYYY HH24:MI:SS') NOW FROM DUAL;

72-SELECT TO_CHAR(SYSDATE, 'MONTH-DAY-YEAR HH24:MI:SS') NOW FROM DUAL;

73-SELECT * FROM EMP WHERE ENAME = 'BLAKE';

74-UPDATE EMP SET SAL = SAL + TO_NUMBER('100.52','9,999.99') WHERE
ENAME = 'BLAKE';

75-SELECT * FROM EMP WHERE ENAME = 'BLAKE';

76- SELECT TRANSLATE('2KRW229', '0123456789ABCDEFGHIJKLMNOPQRSTU-
VWXYZ',                        '9876543210ZYXWVUTSRQPONMLKJIHGFEDCBA')
"Licence" FROM DUAL;

77-SELECT TRUNC(TO_DATE('27-OCT-92', 'DD-MON-YY'), 'YEAR') "First Of
The Year"
FROM DUAL;

78-SELECT TRUNC(15.79,1) "Truncate" FROM DUAL;

79-SELECT UPPER('Carol') FROM DUAL;

80-SELECT USER "User" FROM DUAL;

81-SELECT VARIANCE(sal) "Variance" FROM emp;

82-ALTER SEQUENCE eseq MAXVALUE 1500

83-ALTER SESSION
SET NLS_DATE_FORMAT = 'YYYY MM DD HH24:MI:SS';

84-SELECT TO_CHAR(SYSDATE) Today FROM DUAL;

85-ALTER TABLE emp
ADD (thriftplan NUMBER(7,2),
loancode CHAR(1));

86-ALTER TRIGGER reorder DISABLE;

87-ALTER TRIGGER reorder ENABLE;

88-CREATE USER todd IDENTIFIED BY tiger;

89-ALTER USER todd IDENTIFIED BY lion;
```

```
90-INSERT INTO dept VALUES (50, 'Marketing', 'TAMPA');

91-CREATE VIEW EMP_SAL (Name, Job, Salary) AS SELECT ENAME, JOB, SAL
FROM EMP;

92-SELECT employees_seq.currval
FROM DUAL;

93-INSERT INTO employees
VALUES (employees_seq.nextval, 'John', 'Doe', 'jdoe',
'555-1212', TO_DATE(SYSDATE), 'PU_CLERK', 2500, null, null,
30);

94-CREATE TABLE t1 (c1 NUMBER c2 INTEGER);
CREATE USER a IDENTIFIED BY a;
GRANT SELECT, UPDATE ON t1 TO a;
ALTER TABLE t1 ADD c3 INT;
COMMIT;

95-SELECT LPAD(' ',2*(LEVEL-1)) || ename org_chart,
empno, mgr, job
FROM emp
START WITH job = 'PRESIDENT'
CONNECT BY PRIOR empno = mgr;

96-SELECT * FROM
(SELECT empno FROM emp ORDER BY empno)
WHERE ROWNUM < 11;

97-UPDATE emp
SET sal = 2000
WHERE ename = 'BLAKE';

98-SELECT 'ID=', EMPNO,
'Name=', ENAME,
'Dept=', DEPTNO
FROM EMP WHERE SAL >= 1300;

99-SELECT * FROM (SELECT ENAME FROM EMP WHERE JOB = 'CLERK'
UNION
SELECT ENAME FROM EMP WHERE JOB = 'ANALYST');

100-UPDATE EMP SET SAL = SAL * .45 WHERE JOB = 'PRESIDENT';

101- SELECT last_name, department_name
FROM employees e LEFT OUTER
JOIN departments d ON (e.department_id = d.department_id);

102- INSERT INTO employees
VALUES ('1000','John',NULL);

INSERT INTO employees (employee_id)
VALUES (1000);

INSERT INTO employees (employee_id, first_name, last_name)
VALUES ( 1000, 'John','');

103- GRANT select, insert, update
ON student_grades
TO manager
WITH GRANT OPTION;
```

```
104- SELECT distinct department_id
FROM employees
Where salary > ANY (SELECT AVG(salary)
FROM employees
GROUP BY department_id);

SELECT department_id
FROM employees
WHERE SALARY > ALL (SELECT AVG(salary)
FROM employees
GROUP BY department_id);

SELECT last_name
FROM employees
Where salary > ANY (SELECT MAX(salary)
FROM employees
GROUP BY department_id);

105- SELECT ord_id, cust_id, ord_total
FROM orders
Where ord_date IN (SELECT ord_date
FROM orders
WHERE cust_id = (SELECT cust_id
FROM customers
WHERE cust_name =
'Martin'));

106- ALTER TABLE students
ADD CONSTRAINT stud_id_pk PRIMARY KEY (student_id);

107- CREATE TABLE EMP_Mia
(empno NUMBER(4),
ename VARCNAR2(35),
deptno NUMBER(7,2)
CONSTRAINT emp_deptno_fk REFERENCES dept (deptno));

108- ALTER TABLE student_grades
ADD CONSTRAINT student_id_fk
FOREIGN KEY (student_id) REFERENCES students(student_id);

109- CREATE OR REPLACE VIEW emp_dept_vu AS
SELECT employee_id, employee_name,
department_name, manager_id
FROM employees e, departments d
WHERE e.department_id = d.department_id;

110- SELECT TO_CHAR(2000, '$0,000.00')
FROM dual;

SELECT TO_CHAR(2000, '$9,999.00')
FROM dual;

SELECT TO_CHAR (2000, '$9,999.99')
FROM dual;

111-UPDATE new_employees SET last_name = (Select last_name||
first_name
FROM employees
Where employee_id=180)
```

```
112- SELECT First_Name, department_name, city
FROM employees e, departments d, locations l
WHERE e.department_id = d.department_id
AND d.location_id = l.location_id
AND salary > 10000;

113- SELECT DEPARTMENT_ID, MIN(salary), MAX(salary)
FROM employees
GROUP BY DEPARTMENT_ID

114- CREATE VIEW emp_Vu AS
SELECT employee_id, emp_name, job_id
department_id
FROM employees
WHERE mgr_id IN (102, 120);

115- SELECT e.last_name "Emp_id", e.last_name "Employee",
e.salary, m.employee_id "Mgr_id", m.first_name "Manager"
FROM employees e, employees m
WHERE e.MANAGER_ID = m.employee_id
AND e.salary > 4000;
```

LABORATORIO- I

CREACION Y MANTENIMIENTO DE TABLAS Y VISTAS.

1. Cree una copia de la tabla EMP. Llámela EMPTEST.
2. Añada una columna nueva llamada SEX a la tabla EMPTEST con tipo de datos carácter y longitud 1.

3. Los usuarios han cambiado de opinión. SEX no debe guardar sólo una letra (F o M),
Sino la palabra completa (MALE, FEMALE). Incremente entonces el tamaño de la columna SEX.

4. Cree una vista llamada EMP_NO_MONEY con todas las columnas de la tabla EMP, excepto SAL y COMM.

5. Seleccione todas las columnas de la tabla USER_VIEWS del diccionario de datos.

6. Usando su vista EMP_NO_MONEY, haga una consulta que muestre todos los nombres y fechas de ingreso de los empleados.

7. Trate de introducir una consulta para mostrar el nombre y el salario del empleado usando la vista EMP_NO_MONEY. ¿Qué sucedió?

8. Cree una vista llamada EMP_DEPTNO_TEN que incluya únicamente el nombre del empleado, código de departamento, y código del empleado de la tabla EMP. Los únicos empleados que se podrán ver en esta vista serán los del departamento 10. Emplee los siguientes nombres de columnas: NAME, DNO, ENO.

9. Introduzca una consulta, usando la vista EMP_DEPTNO_TEN

10. Borre la vista EMP_DEPTNO_TEN.

LABORATORIO- II

CREACION DE INDICES Y SECUENCIAS

1. Cree un índice llamado I_EMP_EMPNO sobre la tabla EMP, de manera que no se permitan códigos de empleado repetidos. Se hará uso de este índice en el laboratorio del siguiente capítulo.

2. Cree una secuencia llamada S_DEPT_DEPTNO que inicie con el número 60 y que se incremente de 10 en 10. Se hará uso de esta secuencia en el laboratorio del siguiente capítulo.

3. Verifique que la secuencia haya sido creada, consultando la vista SER_SEQUENCES del diccionario de datos.

4. Emplee la tabla DUAL para mostrar el siguiente valor disponible de la secuencia S_DEPT_DEPTNO.

5. Repita el paso 4.

LABORATORIO de DDL- III

1. Crear una tabla con las columnas Nombre, Puesto, Cargo, salario.
2. Insertar en la tabla que usted creo datos tomados de la tabla emp.
3. Modifique la columna salario para que tenga number(10,2).
4. Borre de la tabla los empleados que Ganen Mas de 19000.
5. Agregar un campo que se llame Sexo.
6. Al campo salario agregar un Check para que no te permita salario 0.
7. Borre la columna Puesto.
8. Inserte solo los datos de la columna empno de la tabla emp en Puesto.
9. Borre toda la fila de la tabla.
10. Agregar un constraint que no acepte datos nulos y que no sea una llave primaria.
11. Renombre la Tabla para que ahora tenga el nombre de Usted.

LABORATORIO- IV

USO DE TIPOS DE DATOS Y FUNCIONES EN LAS CONSULTAS

Emplee la tabla EMP para hacer los siguientes ejercicios:

1. Muestre todos los departamentos con la primera letra en mayúscula y el resto en minúscula.

2. Para cada empleado, calcule el número de meses entre hoy y la fecha de contratación del empleado. Ordene los resultados por el número de meses que el empleado ha trabajado. Redondee el número de meses al número entero más cercano.

3. Muestre el nombre, los ingresos totales y la fecha de ingreso para todos los empleados cuyo salario es mayor que el doble de su comisión. Déle formato a la fecha, de manera que se vea así: *3rd of December, 1982*.

4. Muestre el nombre de todos los empleados en minúsculas y su sueldo redondeado en los cientos.
5. Muestre el número de años transcurridos desde la contratación de Ward.

6. Muestre los primeros 8 caracteres del nombre del empleado, con la primera letra mayúscula, seguidos de *trabaja en el departamento* y seguido del nombre del departamento. Muestre sólo aquellos empleados cuyo nombre contenga al menos una A.

7. Muestre todos los empleados que ganan comisión, sin usar `WHERE JOB = 'SALES'` (sin mostrar que son vendedores).

8. Escriba una consulta que produzca la siguiente salida para cada empleado:
ALLEN gana $ 1600 mensuales pero quiere $ 4800, es decir:
<Nombre Empleado> gana $ <sueldo> mensuales pero quiere $ <3 x sueldo>

LABORATORIO - V

USO DE LAS FUNCIONES DE GRUPO

Emplee las tablas NATION, DEPT y EMP para efectuar las siguientes consultas:

1. Muestre el año en que fueron contratados el empleado más nuevo y el más antiguo.

2. Muestre el número de empleados que hay en la tabla EMP.

3. Muestre el número de cargos diferentes que hay en la tabla EMP.

4. Muestre el número de cargos diferentes para cada departamento. ¡ACEPTE EL RETO! RESUELVA TAMBIEN ESTE EJERCICIO:

5. Muestre el área territorial total para tres grupos de países: los que empiezan con A, con

B y con C. Su salida deberá parecerse a ésta:

```
Area territorial
-------------------------------------------------
Países con A XXXXX
Países con B XXXXX
Países con C XXXXX
```