

# Gestión de Bases de Datos

## Tema 3: Diseño Físico de Bases de Datos en ORACLE



IES Gonzalo Nazareno  
**CONSEJERÍA DE EDUCACIÓN**


Raúl Ruiz Padilla  
rruizp@gmail.com  
Diciembre 2011

---

© Raúl Ruiz Padilla, Diciembre de 2011

Algunos derechos reservados.  
Este artículo se distribuye bajo la licencia  
“Reconocimiento-CompartirIgual 3.0 España” de Creative  
Commons, disponible en  
<http://creativecommons.org/licenses/by-sa/3.0/es/deed.es>

Este documento (o uno muy similar)  
esta disponible en (o enlazado desde)  
<http://informatica.gonzalonazareno.org>



# Índice

---

1. Introducción a SQL.
2. Fundamentos de SQL. Operadores y Funciones.
3. Creación básica de tablas.
4. Restricciones.
5. Modificación de tablas.
6. Borrado de tablas.
7. Uso de sinónimos.



# 1. Introducción a SQL.

---

- SQL. Structured Query Language. 1970 Codd. Consultas, actualizaciones, definicion de datos y control en BD.

- Para administradores, desarrolladores y usuarios.

- Se especifica que se quiere, no los pasos que hay que dar para obtenerlo.

- Interactivamente, introduciendo las órdenes desde terminal, se obtienen resultados.


- Embebido en PL/SQL o Java, por ejemplo.



# 1. Introducción a SQL.

## MySQL vs. ORACLE

---

- En MySQL, existen diferentes bases de datos dentro del servidor. Esto viola el modelo ANSI/SPARC y causa problemas de redundancia de tablas.
  - Así, hay que crear una base de datos y, dentro de ella, crear las tablas tras poner en uso la base de datos correspondiente.
  - En ORACLE existe UNA única base de datos con toda la información relevante de la empresa.
  - No obstante, las tablas pertenecen a usuarios concretos, que pueden conceder permisos sobre ellas a otros usuarios. No deben crearse tablas como usuario SYSTEM.
  - Al conjunto de objetos (tablas, vistas, etc.) de un usuario se le llama esquema del usuario.
- 

# 1. Introducción a SQL.

## Tipos de Sentencias.

---

### .DDL

.CREATE, DROP, ALTER TABLE.

.CREATE,....., VIEW.

.CREATE,....., INDEX.

.CREATE, ....., SYNONYM

### .DML

.SELECT, UPDATE, DELETE, INSERT

### .DCL

.GRANT, REVOKE (conceder, suprimir privilegios)

.COMMIT, ROLLBACK



## 2. Fundamentos de SQL.

### Creación básica de usuarios en ORACLE.

---

- Para crear un usuario en ORACLE, hay que entrar con un usuario que tenga el privilegio de crear usuarios y ejecutar la siguiente sentencia:

CREATE USER *nombreusuario*

IDENTIFIED BY *contraseña*;

- Un usuario recién creado no puede hacer nada, ni siquiera conectarse. Hay que asignarle privilegios o roles (conjuntos de privilegios):

GRANT CONNECT TO *nombreusuario*;

GRANT RESOURCE TO *nombreusuario*;

- Para conectarse a la base de datos:

CONNECT *nombreusuario*;



## 2. Fundamentos de SQL.

### Tipos de Datos en ORACLE.

---

- **VARCHAR2**(tamaño), cadenas long variable, max 4000 bytes
- **CHAR**(tamaño) cadenas long fija, max 2000 bytes
- **NUMBER**(precision, escala)
- **LONG**, cadenas long variable, max 2 gigabytes
- **DATE**, **fechas**, siglo/año/mes/dia/hora/minutos/segundos
- **RAW**(tamaño), cadenas de bytes, máximo 2000bytes.
- **LONG RAW**, graficos, sonidos, 2 gigabytes.
- **ROWID**, cadena hexadecimal, direccion de fila en tabla.
- **CLOB, NCLOB, BLOB**: Objetos binarios de más de dos gigabytes.





## 2. Fundamentos de SQL.

### Operadores en ORACLE.

---

- Aritméticos: +, -, \*, /
- Comparación: =, >, >=, <, <=, <>, !=
- Lógicos: NOT, AND , OR
- De comparación de cadenas: LIKE
- De pertenencia a un conjunto: IN
- De pertenencia a un rango: BETWEEN



## 2. Fundamentos de SQL.

### Operadores en ORACLE. Operadores lógicos.

- Los operadores lógicos sirven para unir expresiones y evaluarlas de forma combinada.

Exp oper Exp , donde Exp puede ser a su vez Exp oper Exp

- Cada uno funciona de una manera distinta:

Exp AND Exp: Es cierto si ambas son ciertas.

Exp OR Exp: Es cierto si al menos una es cierta.

NOT Exp: Es cierto si la expresión es falsa.



## 2. Fundamentos de SQL.

### Operadores en ORACLE. Comparación de cadenas.

- Las cadenas siempre van entre comillas simples.
- Dentro de las comillas se distinguen mayúsculas y minúsculas. No lo olvides.
- Se usa el signo igual si quiero comprobar si las cadenas son idénticas. Ej: nombre = 'Pepe'
- Si quiero comprobar si la cadena sigue un patrón, se emplea el operador LIKE y los caracteres comodines:
  - % para cualquier conjunto de caracteres.
  - \_ para cualquier carácter individual.



## 2. Fundamentos de SQL.

### Operadores en ORACLE. Comparación de cadenas.

•Ejemplos:

•Nombre like 'P%'

•Nombre like '%X%'

•Nombre like '%R'

•DNI like '2%'

•Nombre like '\_\_ M'

•Valor like 'S\_'

•Nombre like '%V\_'



## 2. Fundamentos de SQL.

### Operadores en ORACLE. Pertenencia a un conjunto.

---

• Para comprobar si un valor se encuentra dentro de un conjunto de valores se emplea el operador IN.

• Ejemplo:

Sexo IN ('Varón','Mujer')

Nombre IN ('Pepe','Juan','Eva')

• Se usa frecuentemente combinado con el operador de negación (NOT)

• Ejemplo:

Dpto NOT IN ('10','20','30')

• **Actividad:** Escribe las expresiones de ejemplo usando sólo operadores lógicos.

## 2. Fundamentos de SQL.

### Operadores en ORACLE. Pertenencia a un rango.

---

• Para comprobar si un valor se encuentra dentro de un rango de valores se emplea el operador BETWEEN.

• Ejemplo:

Sueldo BETWEEN 1000 AND 2000;

• Se puede usar combinado con el operador de negación (NOT)

• Ejemplo:

edad NOT BETWEEN '20' AND '40'



## 2. Fundamentos de SQL.

### Funciones SQL en ORACLE.

---

• Son pequeños programas incluidos en ORACLE que reciben uno o varios valores llamados parámetros y devuelven un resultado.

• Hay 5 tipos:

• Aritméticas.

• De cadena de caracteres.

• De manejo de fechas.

• De conversión.

• Otras funciones.



## 2. Fundamentos de SQL.

### Funciones SQL en ORACLE. Aritméticas(I)

---

.De valores simples:

**ABS(n)**= Devuelve el valor absoluto de (n).

**CEIL(n)**=Obtiene el valor entero inmediatamente superior o igual a "n".

**FLOOR(n)** = Devuelve el valor entero inmediatamente inferior o igual a "n".

**MOD (m, n)**= Devuelve el resto resultante de dividir "m" entre "n".

**NVL (valor, expresión)**= Sustituye un valor nulo por otro valor.

**POWER (m, exponente)**= Calcula la potencia de un numero.

**ROUND (numero [, m])**= Redondea números con el numero de decimales indicado.

**SIGN (valor)**= Indica el signo del "valor".

**SQRT(n)**= Devuelve la raíz cuadrada de "n".

**TRUNC (numero, [m])**= Trunca números para que tengan ciertos decimales.

**VARIANCE (valor)**= Devuelve la varianza de un conjunto de valores.





## 2. Fundamentos de SQL.

### Funciones SQL en ORACLE. Aritméticas(II)

---

•Ejemplos:

```
SELECT ABS( - 10000) from DUAL;
```

```
SELECT CEIL(20.3), CEIL(16), CEIL(-20.3), CEIL(-16) FROM DUAL ;
```

```
SELECT FLOOR(20.3), FLOOR(16), FLOOR(-20.3), FLOOR(-16) FROM DUAL ;
```

```
SELECT MOD(11,4), MOD(10, -15), MOD(-10, -3), MOD(10.4, 4.5) FROM DUAL;
```

```
SELECT SAL, COMM, (SAL + NVL(COMM, 0)) FROM SCOTT.EMP;
```

```
SELECT POWER(2, 4), POWER(2, -4), POWER(3.5, 2.4), POWER(4.5, 2) FROM DUAL;
```

```
SELECT ROUND(1.56, 1), ROUND(1.56), ROUND(1.2234, 2), ROUND(1.2676, 3) FROM DUAL;
```

```
SELECT ROUND(145.5, -1), ROUND(145.5, -2), ROUND(145.5, -3) FROM DUAL;
```

```
SELECT SIGN(-10), SIGN(10) FROM DUAL;
```

```
SELECT SQRT(25), SQRT(25.6) FROM DUAL;
```

```
SELECT TRUNC(1.5634, 1), TRUNC(1.1684, 2), TRUNC(1.662) FROM DUAL;
```

```
SELECT TRUNC(187.98, -1), TRUNC(187.98, -2), TRUNC(187.98, -3) FROM DUAL;
```



## 2. Fundamentos de SQL.

### Funciones SQL en ORACLE. Aritméticas(III)

---

.De grupos de valores:

**AVG(expresión):** Calcula el valor medio de "n" ignorando los valores nulos.

**COUNT (\* | Expresión):** Cuenta el numero de veces que la expresión evalúa algún dato con valor no nulo. La opción "\*" cuenta todas las filas seleccionadas.

**MAX (expresión):** Calcula el máximo.

**MIN (expresión):** Calcula el mínimo.

**SUM (expresión):** Obtiene la suma de los valores de la expresión.



## 2. Fundamentos de SQL.

### Funciones SQL en ORACLE. Aritméticas(IV)

---

•Ejemplos de funciones aritméticas de grupos de valores:

```
SELECT AVG(SAL) FROM EMP;
```

```
SELECT COUNT(*) FROM EMP;
```

```
SELECT COUNT(COMM) FROM EMP;
```

```
SELECT MAX(SAL) FROM EMP;
```

```
SELECT MAX(ENAME) FROM EMP;
```

```
SELECT MIN(SAL) FROM EMP;
```

```
SELECT MIN(ENAME) FROM EMP;
```

```
SELECT SUM(SAL) FROM EMP;
```

```
SELECT VARIANCE(SAL) FROM EMP;
```

```
SELECT COUNT(DISTINCT(JOB)) FROM EMP;
```



## 2. Fundamentos de SQL.

### Funciones SQL en ORACLE. De cadenas(I)

---

•Funciones que devuelven cadenas o caracteres:

**CHR(n)** = Devuelve el carácter cuyo valor ASCII es equivalente a "n".

**CONCAT (cad1, cad2)**= Devuelve "cad1" concatenada con "cad2".

**LOWER (cad)**= Devuelve la cadena "cad" en minúsculas.

**UPPER (cad)**= Devuelve la cadena "cad" en mayúsculas.

**INITCAP (cad)**= Convierte la cadena "cad" a tipo titulo.

**LPAD (cad1, n[,cad2])**= Añade caracteres a la izquierda de la cadena hasta que tiene una cierta longitud.

**RPAD (cad1, n[,cad2])**= Añade caracteres a la derecha de la cadena hasta que tiene una cierta longitud.

**LTRIM (cad [,set])**= Suprime un conjunto de caracteres a la izquierda de la cadena.

**RTRIM (cad [,set])**= Suprime un conjunto de caracteres a la derecha de la cadena.

**REPLACE (cad, cadena\_busqueda [, cadena\_sustitucion])**= Sustituye un carácter o caracteres de una cadena con 0 o mas caracteres.

**SUBSTR (cad, m [,n])**= Obtiene parte de una cadena.



## 2. Fundamentos de SQL.

### Funciones SQL en ORACLE. De cadenas(II)

---

•Ejemplos de funciones de cadenas que devuelven cadenas:

```
SELECT CHR(75), CHR(65) FROM DUAL;
```

```
SELECT CONCAT('El nombre es: ', ename) FROM EMP;
```

```
SELECT LOWER('oRACLe Y sql'), UPPER('oRACLe Y sql'), INITCAP('oRACLe Y sql') FROM DUAL;
```

```
SELECT LPAD(ENAME, 20, '.'), RPAD(ENAME, 20, '.') FROM EMP;
```

```
SELECT LTRIM('      HOLA'), RTRIM('      ADIOS') FROM DUAL;
```

```
SELECT REPLACE ('BLANCO Y NEGRO', 'O', 'AS') FROM DUAL;
```

```
SELECT SUBSTR(ENAME, 1, 1) FROM EMP;
```



## 2. Fundamentos de SQL.

### Funciones SQL en ORACLE. De cadenas(III)

---

•Funciones que devuelven valores numéricos:

**ASCII(cad)**= Devuelve el valor ASCII de la primera letra de la cadena "cad".

**INSTR (cad1, cad2 [, comienzo [,m]])**= Permite una búsqueda de un conjunto de caracteres en una cadena, devolviendo el carácter de comienzo de la subcadena dentro de la cadena principal.

**LENGTH (cad)**= Devuelve el numero de caracteres de cad.

•Ejemplos:

```
SELECT ASCII('A') FROM DUAL;
```

```
SELECT INSTR('II VUELTA CICLISTA A TALAVERA', 'TA', 3,2) FROM DUAL;
```

```
SELECT INSTR('II VUELTA CICLISTA A TALAVERA', 'A', -1) FROM DUAL;
```

```
SELECT ENAME, LENGTH(ENAME) FROM EMP;
```



## 2. Fundamentos de SQL.

### Funciones SQL en ORACLE. De fechas.

---

•Funciones de manejo de fechas:

**SYSDATE:** Devuelve la fecha del sistema.

**ADD\_MONTHS (fecha, n):** Devuelve la fecha "fecha" incrementada en "n" meses.

**LAST\_DAY (fecha):** Devuelve la fecha del último día del mes que contiene "fecha".

**MONTHS\_BETWEEN (fecha1, fecha2):** Devuelve la diferencia en meses entre las fechas "fecha1" y "fecha2".

**NEXT\_DAY (fecha, cad):** Devuelve la fecha del primer día de la semana indicado por "cad" después de la fecha indicada por "fecha".

Ejemplos:

```
SELECT SYSDATE FROM DUAL;
```

```
SELECT ENAME, HIREDATE, ADD_MONTHS(HIREDATE, 12) FROM EMP
```

```
SELECT ENAME, HIREDATE, LAST_DAY(HIREDATE) FROM EMP
```

```
SELECT MONTHS_BETWEEN(SYSDATE, HIREDATE) FROM EMP;
```

```
SELECT NEXT_DAY(SYSDATE, 'JUEVES') FROM DUAL
```



## 2. Fundamentos de SQL.

### Funciones SQL en ORACLE. De conversión. (I)

---

•Funciones de conversión de un tipo a otro:

**TO\_CHAR:** Transforma un tipo DATE ó NUMBER en una cadena de caracteres.

*TO\_CHAR(fecha, 'formato')*

*TO\_CHAR(numero, 'formato')*

**TO\_DATE:** Transforma un tipo NUMBER ó CHAR en DATE.

*TO\_DATE (cadena, 'formato')*

**TO\_NUMBER:** Transforma una cadena de caracteres en NUMBER.

*TO\_NUMBER (cadena [, 'formato'])*





## 2. Fundamentos de SQL.

### Funciones SQL en ORACLE. De conversión. (II)

---

• Ejemplos de funciones de conversión de un tipo a otro:

```
SELECT TO_CHAR(HIREDATE, 'month DD, YYYY') FROM EMP;
SELECT TO_CHAR(HIREDATE, 'mon ddd, yyyy') FROM EMP;
ALTER SESSION SET NLS_DATE_FORMAT = 'DD/month/YYYY HH24 :MI :SS';
SELECT SYSDATE FROM DUAL;
ALTER SESSION SET NLS_DATE_LANGUAGE = French;
SELECT TO_CHAR(SYSDATE, '"HOY ES " DAY " , " DD " DE "
                MONTH " DE " YYYY') AS FECHA FROM DUAL;
SELECT TO_CHAR(12345.67, '999G999D999') FROM DUAL;
ALTER SESSION SET NLS_NUMERIC_CHARACTERS = ', .';
SELECT TO_CHAR(123, 'L999') FROM DUAL;
SELECT TO_NUMBER('-123456'), TO_NUMBER('123,99', '999D99') FROM DUAL;
ALTER SESSION SET NLS_DATE_FORMAT = 'DD/MM/YYYY';
SELECT TO_DATE('01012006') FROM DUAL;
SELECT TO_CHAR(TO_DATE('01012006', 'DDMMYYYY'), 'MONTH') FROM DUAL;
```

### 3. Creación básica de tablas.

---

Tendremos en cuenta los siguientes aspectos:

Elección del **NOMBRE DE TABLA**, debe identificar su contenido, máximo 30 caracteres y único. No será palabra reservada Oracle. Comenzará por letra. Ej: ***ALUMNOS***

Elección de los **NOMBRES DE COLUMNA**, han de ser autodescriptivos. Ejs: ***DNI***, ***NOMBRE***.

Elección de los **TIPOS DE DATOS** y **TAMAÑOS** de las columnas. Solo se usan tipos numéricos si van a realizarse operaciones aritméticas con la columna.

Definición de las **RESTRICCIONES** necesarias: claves primarias y ajenas, columnas obligatorias, valores por defecto, rangos de valores, etc.



### 3. Creación básica de tablas. Sintaxis.

---

```
CREATE TABLE nombre_tabla  
(colum_1          tipo_dato( ),  
  colum_2          tipo_dato( ),  
  .....  
) ;
```

Ejemplo:

```
CREATE TABLE alumnos_07  
(num_matri        NUMBER(6),  
  nombre           VARCHAR2(15),  
  fecha_nac        DATE,  
  direccion         VARCHAR2(50)  
) ;
```



### 3. Creación básica de tablas. Vistas del D.Datos.

---

**USER\_TABLES**, información sobre tablas propiedad del usuario.

CAT es un sinónimo de USER\_TABLES.

**ALL\_TABLES**, información sobre las tablas que son propiedad del usuario o el usuario tiene permisos sobre ellas.

**DBA\_TABLES**, información sobre todas las tablas existentes. Solo se puede consultar con privilegios de administración.

Para conocer las columnas de una tabla basta con introducir el siguiente comando:

DESC nombretabla.



## 4. Restricciones.

### Integridad de datos e integridad referencial.

**.INTEGRIDAD DE DATOS:** Los datos antes de almacenarse en BD han de cumplir ciertas condiciones establecidas por las restricciones.

**.INTEGRIDAD REFERENCIAL:** Impiden que en una tabla se introduzca un valor que no tenga su correspondencia en la tabla de la que proviene.

**.Las restricciones** hacen que el usuario o las aplicaciones tengan menos trabajo a la hora de manipular los datos y sea Oracle el encargado de las tareas de mantenimiento de la integridad de la BD.

**.Es fundamental** dar un nombre a las restricciones para poder manejarlas posteriormente. No pueden existir dos restricciones con el mismo nombre, ni siquiera en tablas diferentes.



## 4. Restricciones.

### Restricción de clave primaria.

---

- Clave primaria: Columna o conjunto de columnas que identifican unívocamente cada fila de una tabla.
- Única y obligatoria.
- Solo se define una por tabla.
- Si la tabla tiene una clave primaria compuesta por varias columnas, no se crean varias claves primarias, sino una que incluye a todas las columnas.
- Automáticamente se crea un índice que agiliza el acceso a la tabla por ese campo.



## 4. Restricciones.

### Restricción de clave primaria. Sintaxis.

---

A nivel de **columna**:

```
CREATE TABLE nombre_tabla  
(  
    nombrecolum_1 TIPO_DATO() [CONSTRAINT nombre_constraint] PRIMARY KEY,  
    .....  
);
```

o bien a nivel de **tabla**:

```
CREATE TABLE nombre_tabla  
(  
    colum_1          TIPO_DATO(),  
    .....  
[CONSTRAINT nombre_constraint] PRIMARY KEY(colum1[,...]),  
    .....  
);
```



## 4. Restricciones.

### Restricción de clave ajena.

---

Formada por:

Una o varias columnas asociadas a una clave primaria de otra o de la misma tabla.

Se pueden definir tantas claves ajenas o foráneas como sean necesarias.

El valor debe ser igual a un valor de la clave referenciada: regla de integridad referencial.





## 4. Restricciones.

### Restricción de clave ajena. Sintaxis.(I)


---

A nivel de columna:

La clave ajena se define en la descripción de la columna usando la palabra clave **REFERENCES**

```
CREATE TABLE nombre_tabla  
(  
column1 tipo_dato() [CONSTRAINT nombre_restriccion] REFERENCES nombre_tabla  
    [(columna)] [ON DELETE CASCADE]  
....  
);
```

**ON DELETE CASCADE** produce un borrado en cascada para garantizar que se mantiene la integridad referencial



## 4. Restricciones.


### Restricción de clave ajena. Sintaxis.(II)

---

A nivel de tabla:

La clave ajena se define **al final** de todas las columnas usando las cláusulas **FOREIGN KEY** y **REFERENCES**.

```
CREATE TABLE nombre_tabla
(
    column1      tipo_dato(),
    .....
    [CONSTRAINT nombre_restriccion]
    FOREIGN KEY(columna [, columna..])
    REFERENCES nombre_tabla [(columna [, columna..))]
    [ON DELETE CASCADE]
    ....
);
```



## 4. Restricciones.

### Restricción de obligatoriedad.

---

Asociada a una columna, significa que no puede tener valores nulos.

```
CREATE TABLE nombre_tabla  
(  
    column1 tipo_dato() [CONSTRAINT nombre_restriccion] NOT NULL,  
    ....  
);
```



## 4. Restricciones.

### Restricción de valor por defecto.

---

- Sirve para asignar valores por defecto a una columna, cuando no se especifica valor en la cláusula INSERT.
- Se pueden incluir constantes, funciones SQL y las variables UID y SYSDATE.
- Es el único tipo de restricción que no lleva nombre puesto que no puede ser violada.

Formato:

```
CREATE TABLE nombre_tabla
```

```
(
```

```
    column1      tipo_dato() DEFAULT valor_por defecto,
```

```
    ....
```

```
);
```



## 4. Restricciones.

### Restricción de verificación de condiciones.

---

• Sirve para comprobar el cumplimiento de ciertas condiciones en los valores que puede tomar una columna.

- No se pueden incluir subconsultas, ni UID, SYSDATE y USER.

- *Formato a nivel de columna:*

```
CREATE TABLE nombre_tabla
(
    column1 tipo_dato() [CONSTRAINT nombre_restriccion] CHECK (condicion),
    .....
);
```

- *Formato a nivel de tabla:*

```
CREATE TABLE nombre_tabla
(
    column1      tipo_dato(),
    .....
    [CONSTRAINT nombre_restriccion] CHECK (condicion),
    .....
);
```



## 4. Restricciones.

### Restricción de unicidad.

---

Evita valores repetidos en una columna.

Puede contener una o varias columnas, se crea automáticamente un índice por esa columna, admite valores NULL si no se especifica lo contrario.

Formato a nivel de columna:

```
CREATE TABLE nombre_tabla  
(  
    column1 tipo_dato() [CONSTRAINT nombre_restriccion] UNIQUE,  
    .....  
);
```

Formato a nivel de tabla:

```
CREATE TABLE nombre_tabla  
(  
    column1      tipo_dato(),  
    .....  
    [CONSTRAINT nombre_restriccion] UNIQUE (columna [,columna...]),  
    ..... );
```



## 4. Restricciones.

### Resumen de la sintaxis.

---

- A nivel de columna:

```
CREATE TABLE nombre_tabla  
( colum_1          TIPO_DATO( )  
[CONSTRAINT nombre_constraint]  
[NOT NULL] [UNIQUE] [PRIMARY KEY] [DEFAULT valor]  
[REFERENCES nombre_tabla [(colum_1 [,colum_n])]  
[ON DELETE CASCADE]]  
[CHECK condicion]),  
.....  
);
```

- A nivel de tabla:

```
CREATE TABLE nombre_tabla  
( colum_1          TIPO_DATO( ),  
.....,  
  colum_n          TIPO_DATO( ),  
[CONSTRAINT nombre_constraint]  
{[UNIQUE] | [PRIMARY KEY] (COLUMN1, )},  
[CONSTRAINT nombre_constraint]  
[FOREIGN KEY (COLUMN,...)]  
REFERENCES nombre_tabla [(colum...]  
[ON DELETE CASCADE]],  
[CONSTRAINT nombre_constraint]  
[CHECK (condicion)],  
.....  
);
```



## 4. Restricciones.

### Ejemplos.

- A nivel de columna:

**CREATE TABLE** empleados

```
(nombre          VARCHAR2(25)    CONSTRAINT pk_emp PRIMARY KEY,  
edad            NUMBER          CONSTRAINT edad_ok CHECK (edad BETWEEN 18 AND 35),  
cod_provincia   NUMBER(2)      CONSTRAINT fk_prov REFERENCES PROVINCIAS ON DELETE  
                                     CASCADE  
);
```

- A nivel de tabla:

**CREATE TABLE** empleados

```
(nombre          VARCHAR2(25),  
edad            NUMBER,  
cod_prov        NUMBER(2),  
CONSTRAINT pk_emple PRIMARY KEY(nombre),  
CONSTRAINT ck_edad_empleados CHECK(edad BETWEEN 18 AND 35),  
CONSTRAINT fk_emple_07 FOREIGN KEY(cod_prov) REFERENCES provincias ON DELETE CASCADE  
);
```



## 4. Restricciones.

### Consultas al Diccionario de Datos.

- En el diccionario de datos se almacena toda la información acerca de las restricciones existentes en la base de datos.
- Hay que saber consultar dicha información para ver qué restricciones tengo definidas en cada tabla. Para ello están las siguientes vistas:

**USER\_CONSTRAINTS** (restricciones de las tablas del usuario)

**ALL\_CONSTRAINTS** (restricciones de las tablas a las que puede acceder el usuario)

**DBA\_CONSTRAINTS** (todas las restricciones sobre todas las tablas)

Tipos de restricciones:

<b>C</b>	<b>CHECK</b>	<b>R</b>	<b>FOREIGN KEY</b>
<b>P</b>	<b>PRIMARY KEY</b>	<b>U</b>	<b>UNIQUE</b>



## 4. Restricciones.

### Consultas al Diccionario de Datos (II).

- Para saber sobre qué columnas de la tabla están definidas las restricciones debemos consultar las siguientes vistas del diccionario de datos:

**USER\_CONS\_COLUMNS** (restricciones en columnas de tablas de usuario)

**ALL\_CONS\_COLUMNS** (restricciones en columnas de tablas a las que puede acceder el usuario)

**DBA\_CONS\_COLUMNS** ( todas las restricciones de las columnas)



## 5. Modificación de tablas.

La orden **ALTER TABLE**, nos permitirá:

- Añadir, modificar o eliminar columnas de una tabla.
- Añadir o eliminar restricciones.
- Activar o desactivar restricciones

### Sintaxis:

```
ALTER TABLE nombre_tabla  
{ADD (columna)  
[MODIFY (column [....])]  
[DROP COLUMN (column....)]  
[ADD CONSTRAINT restricción]  
[DROP CONSTRAINT restricción]  
[DISABLE CONSTRAINT restricción]  
[ENABLE CONSTRAINT restricción]  
};
```



## 5. Modificación de tablas.

### Añadir columnas.

---

**ADD**, añadir columnas a tabla, tendremos en cuenta:

Si la columna **no tiene restricción NOT NULL**, se puede añadir en cualquier momento.

**Si tiene restricción NOT NULL:**

- 1- Se añade la columna sin restricción.
- 2- Se da valor a la columna en todas las filas.
- 3- Se añade la restricción NOT NULL




## 5. Modificación de tablas.

### Modificar columnas.

---

**MODIFY**, modifica una o más columnas existentes en una tabla. Tendremos en cuenta lo siguiente:

- 1- Se puede aumentar longitud de columna en cualquier momento.
  - 2- Si disminuimos la longitud esta no puede ser de menor tamaño que el valor máximo almacenado.
  - 3- Se puede aumentar o disminuir el valor de los decimales en columna **NUMBER**.
  - 4- Si columna es **NULL** en toda la tabla, se puede modificar la longitud y el tipo de dato.
  - 5- **MODIFY... NOT NULL**, cuando no haya ningún valor **NULO** en la columna que se modifica.
- 

## 5. Modificación de tablas.

### Eliminar columnas.

---

**DROP COLUMN**, para borrar una columna de la tabla.

Tendremos en cuenta:

- 1-No se pueden borrar todas las columnas de una tabla.
- 2-No se pueden borrar claves primarias, referenciadas por claves ajenas.



## 5. Modificación de tablas.

### Añadir y eliminar restricciones.

---

Se pueden añadir y borrar los siguientes tipos de restricciones:

PRIMARY KEY

NOT NULL

FOREIGN KEY

UNIQUE

CHECK

ALTER TABLE nombre\_tabla

ADD CONSTRAINT nombre\_constraint;

ALTER TABLE nombre\_tabla

DROP CONSTRAINT nombre\_constraint;



## 5. Modificación de tablas.

### Activar y desactivar restricciones.

---

Puede interesar desactivar temporalmente las restricciones si vamos a importar un gran volumen de datos que sabemos que ya cumplen las restricciones.

Para desactivarlas:

```
ALTER TABLE nombre_tabla  
DISABLE CONSTRAINT nombre_restriccion;
```

```
ALTER TABLE nombre_tabla  
DISABLE ALL CONSTRAINTS;
```

Para activarlas:

```
ALTER TABLE nombre_tabla  
ENABLE CONSTRAINT nombre_restriccion;
```

Las restricciones por defecto se activan al crearlas, aunque se pueden desactivar en el momento de crearlas.

```
ALTER TABLE emp  
ADD CONSTRAINT UNI_ename  
UNIQUE (ename) DISABLE;
```





## 6. Borrado de tablas.

### Orden DROP TABLE.

---

Formato:

`DROP TABLE [usuario.]nombre_tabla [CASCADE CONSTRAINTS];`

- Cada usuario puede borrar sus propias tablas.
- El DBA u otro usuario con privilegios de `DROP ANY TABLE`, podrán borrar cualquier tabla.
- Se borran también los índices y privilegios asociados a ellas.
- Las vistas y sinónimos dejan de funcionar, con lo que habría que borrarlos.
- `CASCADE CONSTRAINTS`, elimina las restricciones de integridad referencial que remitan a la clave primaria de la tabla borrada.



## 6. Borrado de tablas.

### Orden TRUNCATE TABLE.

---

Formato:

TRUNCATE TABLE [usuario.]nombre\_tabla [{REUSE|DROP} STORAGE];

- Elimina filas de una tabla, liberando el espacio asociado.
- No desaparece la definición de la tabla.
- Orden DDL, no es posible realizar un ROLLBACK.
- No activa disparadores DELETE.

No se puede truncar una tabla cuya clave primaria sea referenciada por una ajena, antes tendremos que desactivar las restricciones.



## 7. Uso de sinónimos.

---

Es posible crear un sinónimo para ciertos objetos de la base de datos.

```
CREATE [OR REPLACE] [PUBLIC] SYNONYM [esquema.]sinonimo  
FOR [esquema.]objeto;
```

Con la opción 'PUBLIC' se crea un sinónimo público accesible a todos los usuarios, siempre que tengan los privilegios adecuados para el mismo.

Sirve para no tener que usar la notación 'esquema.objeto' para referirse a un objeto que no es propiedad de usuario.

Ejemplo:

```
CREATE PUBLIC SYNONYM EMPLEADOS FOR SCOTT.EMP;
```

