SQL

MÓDULO: Gestión de Bases de Datos
CICLO: Grado Superior – Administración de Sistemas

Elaborado por/*Egilea*: Javier Seara
Fecha de edición/*Argitalpen-data*: Septiembre 2010 / Irailak 2010

ÍNDICE

DDL (Lenguaje de Definición de Datos)

Este sub-lenguaje de SQL es utilizado para la creación y mantenimiento de la estructura de la Base de Datos.

Algunos de los elementos que genera son:

- ✓ Base de Datos (Instancia en Oracle)
- ✓ Tablas
- ✓ Columnas
- ✓ Índices
- ✓ Restricciones (Constraints)
- ✓ Sinónimos
- ✓ Vistas
- ✓ Usuarios
- ✓ Secuenciadores

Nombre de los objetos de una Base de Datos

Los nombres de los objetos en SQL se llaman identificadores y tienen algunas reglas que cumplir en cuanto a su composición:

- ✓ Un identificador puede tener entre 1 y 30 caracteres
- ✓ No pueden tener comillas.
- ✓ Los identificadores deben empezar por un carácter alfabético (no numérico) y los siguientes caracteres pueden ser alfabéticos, dígitos, "_", "&" y "#".
- ✓ Una palabra reservada (palabras que forman parte de la sintaxis) del lenguaje no puede ser utilizada como identificador.
- ✓ En un mismo esquema, una tabla y una vista no pueden tener el mismo nombre, porque se dice que están en el mismo ámbito. Los índices que están en otro ámbito, sí podrían llamarse como una tabla o como una vista.

Símbolos utilizados en las descripciones sintácticas de las órdenes SQL

[ ] → Aquellos elementos que se encuentran entre corchetes son opcionales.
Ej: ls [directorio]

{ } → Constituye la obligatoriedad de incluir uno de los elementos encerrados entre llaves. Dichos elementos estarán separados por " | ".
Ej: {while | for}

| → Se utiliza para separar los diferentes elementos opcionales u obligatorios.
Ej: [while | for]

... → Indica que el elemento anterior a los puntos puede aparecer más veces en las sintaxis.
Ej: WHILE condicion do
instruccion;
...
end
← más de una instrucción

,... → Como el caso anterior, pero cada nueva aparición del elemento debe ir precedida por una coma.
Ej: Definición de variables
Identificador_variable ,... tipo_variable ;

palabra → Las palabras en minúscula corresponden con identificadores SQL, expresiones, etc ...
Ej: **CREATE TABLE** nombre_tabla

PALABRA → Las palabras en mayúscula y en negrita son reservadas del lenguaje SQL y, por tanto, invariables.
Ej: **CREATE TABLE**

Subrayado → Si no se especifica otra opción, la palabra o palabras subrayadas se toman por defecto.
Ej: **SELECT** [**ALL** | **DISTINCT**]
Si no se coge ninguno se pone por defecto ALL, porque está subrayado.

Negrita → Cualquier signo de las convenciones que se encuentre en negrita son obligatorias en la sintaxis.

Tablas

Las operaciones que se pueden realizar sobre las tablas son:

- ✓ Creación
- ✓ Modificación de la estructura
- ✓ Renombrado
- ✓ Borrado

Creación de tablas

En la creación de las tablas deben especificarse dos cosas:

- ➢ Definición de columnas
- ➢ Restricciones de integridad

Para definir una columna habrá que especificar el nombre y el tipo de dato que se va a almacenar en ese campo de la tabla.

TIPOS DE DATOS

Tipo de Dato	Descripción
NUMBER (p,s)	Numéricos con precisión p y escala s (parte entera y decimal) (p de 1 a 38)
VARCHAR2 (n)	Cadena de caracteres de longitud variable, como máximo, n (Máximo 4000 bytes y mínimo 1 byte)
VARCHAR (n)	Cadena de caracteres de longitud variable, como máximo, n
CHAR (n)	Cadena de caracteres de longitud fija, n Entre 1 y 2000 caracteres
DATE	Fecha
LONG	Cadena de caracteres hasta un máximo de 2 GB. (Se emplea para textos o descripciones de tamaño grande y variable)
RAW (n)	Es para cadenas de datos binarios de hasta 2000 bytes
LONG RAW	Cadena de datos binarios de longitud variable de hasta 2 GB (almacenamiento de gráficos, sonidos,...)
ROWID	Direcciones de Fila (registro)

CREACIÓN DE TABLA

CREATE TABLE nombre_tabla

(nombre_columna tipo_columna [restricciones_columna],
[nombre_columna tipo_columna [restricciones_columna]]
,...

[restricción_tabla ,...]

) [**TABLESPACE** espacio_de tabla] ;

RESTRICCIÓN DE INTEGRIDAD (CONSTRAINT)

Las restricciones de integridad a definir son:

- [NOT] NULL
- DEFAULT
- PRIMARY KEY
- UNIQUE
- FOREIGN KEY
- CHECK

Sintaxis de **restricción de tabla**:

[CONSTRAINT nombre_restricción]
{
{UNIQUE | PRIMARY KEY} (lista_columnas1)

| FOREIGN KEY (lista_columna2)
REFERENCES tabla_referenciada [(lista_columnas3)]
[ON DELETE CASCADE]

| CHECK (condición)
}
[DISABLE]

Donde:

- UNIQUE Palabra reservada que indica que se va a añadir una clave alternativa a la tabla.
- PRIMARY KEY Palabra reservada que indica que se va a añadir la clave primaria a la tabla
- Lista_columnas1 Columnas que definen la clave primaria (separadas por comas)
- FOREIGN KEY Indica la creación de una clave extranjera (referencial) para la tabla.
- Lista_columnas2 Columnas que componen la clave extranjera.
- REFERENCES Palabra reservada que indica que se va a nombrar la tabla referenciada en la clave extranjera.
- Tabla_referenciada Nombre de la tabla referenciada
- Lista_columna3 Columnas referenciadas (si la clave extranjera hace referencia a una clave primaria no hace falta ponerlos).
- ON DELETE CASCADE Permite el borrado de las filas de la tabla referenciada, heredando la acción a las filas de las tablas donde están definidas las claves extranjeras.
- CHECK (condición) Condición que deben cumplir los valores introducidos en la columna especificada.

  Donde Condición es una lista de subcondiciones separadas por operadores AND , OR, NOT. Las condiciones válidas son las siguientes:

  expresión operador_relación expresión

  operador_relación ⏩ >, <, != (<>), >= , <=

  expresión [NOT] BETWEEN expr1 and expr2

  expresión [NOT] LIKE 'literal'

  Comodines: % Cualquier cadena de caracteres.
  _ Un carácter obligatorio.

  expresión [NOT] IN (lista_valores)

Sintaxis de **restricción de columna**:

```
[ CONSTRAINT nombre_restricción ]
  {
        DEFAULT valor
      | [NOT] NULL
      | { UNIQUE | PRIMARY KEY }
      | REFERENCES tabla_referenciada [(nombre_columna_ref)]
        [ON  DELETE CASCADE]
      | CHECK (condición)
  }
  [ DISABLE ]
```

Donde:

- DEFAULT valor: Valor por defecto asignado a una columna.

  valor : constante numérica o alfanumérica.
  Si la columna es de tipo fecha o carácter, éste tiene que ir entre comillas.

- NOT NULL: La columna no admite varios valores nulos.

- NULL: La columna admite valores nulos (por defecto).

NOTA: **Creación de tabla mediante consulta (DDL)**

Creación de tablas: (mediante tabla derivada de un SELECT)

```
CREATE TABLE tabla [(columnas)]
AS SELECT...
```

MODIFICACIÓN DE TABLA

Un cambio de estructura puede consistir en las siguientes operaciones:

- Agregar una nueva columna a la tabla (ADD).
- Agregar nuevas restricciones (Cláusula ADD).
- Modificar una columna (Cláusula MODIFY).
- Para modificar una restricción, debe borrarse e introducirse de nuevo.
- Suprimir restricciones (DROP).

Sintaxis:

```
ALTER TABLE nombre_tabla
    {
        ADD ({nueva_columna | restricción_tabla}
                ,...)
      | MODIFY (modifica_columna
                ,...)
      | DROP COLUMN columna

      | DROP CONSTRAINT nombre_restricción

    } ...

    [ [ENABLE cláusula] [DISABLE cláusula] ]
```

Donde:

nueva_columna ::=

nombre_columna tipo_columna [DEFAULT expresión] [restricción_columna...]

modifica_columna ::=

nombre_columna [tipo_columna] [DEFAULT expresión] [restricción_columna...]

RENOMBRADO DE TABLA

Sintaxis:

```
RENAME nombre_tabla TO nuevo_nombre_tabla
```

BORRADO DE TABLA

Sintaxis:

```
DROP TABLE nombre_tabla [ CASCADE CONSTRAINTS ]
```

Índices

Los motivos de definición de un índice son:

- Aumentar la velocidad de acceso a la tabla reduciendo el tiempo de respuesta.
- Asegurar la identificación unívoca de filas de una tabla.

El sublenguaje DDL del SQL contempla únicamente dos operaciones respecto a los índices: creación y borrado.

Respecto al diseño de los índices podemos tener en cuenta las siguientes características de funcionamiento:

- La adición indiscriminada de índices puede ralentizar las operaciones de inserción y actualización de filas.
- Deben considerarse los criterios restrictivos (cláusula WHERE), así como la ordenación deseada de las consultas más frecuentes, para así crear los índices adecuados.

CREACIÓN DE ÍNDICE

Sintaxis:

```
CREATE [UNIQUE] INDEX nombre_índice ON Nombre_Tabla
( columna [ASC | DESC] [,columna [ASC | DESC] [,...] )
```

BORRADO DE ÍNDICES

Sintaxis:

```
DROP INDEX nombre_índice
```

Vistas (Views)

- Una vista es una tabla derivada a la que se asigna un nombre.
- Una vista, podrá tratarse como una tabla de la base de datos, pero a diferencia de las tablas no contiene datos físicos.
- El contenido de una vista es evaluado cada vez que se la invoca en una instrucción SQL.

CREACIÓN DE VISTAS

Sintaxis:

```
CREATE [ OR REPLACE ] VIEW nombre_vista [ (columnas) ]
AS instrucción_select
```

BORRADO DE VISTAS

Sintaxis:

```
DROP VIEW nombre_vista
```

OPERACIONES SOBRE VISTAS

Las operaciones que se pueden realizar sobre vistas son las mismas que las que se llevan a cabo sobre las tablas.

Por lo general se utilizan para realizar consultas (SELECT), aunque también es posible realizar operaciones de inserción, actualización y borrado si se cumplen ciertas restricciones.

- Consulta. El modo de consultar una vista es el mismo que para las tablas.

```
SELECT lista_select
FROM nombre_vista
WHERE condición
```

Sinónimos

Cuando tenemos acceso a las tablas de otro usuario y deseamos consultarlas, es preciso dar el nombre del usuario propietario antes del nombre de la tabla.

*Ej. : SELECT * FROM almacen.PROVINCIAS*

Sin embargo, podemos crear sinónimos para hacer referencia a la tabla sin necesidad de especificar el nombre de usuario propietario por delante.

*Ej. : SELECT * FROM nombresinónimo_provincias*

CREACIÓN DE SINÓNIMOS

Sintaxis:

CREATE [PUBLIC] SYNONYM sinónimo FOR [usuario.] nombretabla

Donde:

PUBLIC hace que el sinónimo este disponible para todos los usuarios. (sólo usuarios DBA)

BORRADO DE SINÓNIMOS

Sintaxis:

DROP [PUBLIC] SYNONYM [usuario.] sinónimo

QL (Lenguaje de Consultas – Query Language)

CARACTERÍSTICAS:

- La única forma de leer las filas contenidas en las tablas de una B.D., es mediante la instrucción SELECT del sub-lenguaje de consultas de SQL.
- El resultado de esta consulta es una tabla derivada compuesta por un determinado número de columnas.
- Una instrucción SELECT es una operación entre tablas cuyo resultado puede ser interpretado como otra tabla (<< tabla derivada >>).
- Esta tabla sería el producto cartesiano de las tablas contenidas en la cláusula FROM.
- En caso de existir una cláusula WHERE se eliminaría de la tabla del resultado todas aquellas filas que no cumpliesen las condiciones especificadas en dicha cláusula.

Sintaxis:

```
SELECT [ALL | DISTINCT] lista_select
FROM lista_tablas
[WHERE {condición_comparación | condición_subselect }]
[GROUP BY lista_grupos]
[HAVING condición_grupos]
[ORDER BY columna_order [ASC | DESC] [,...]]
```

Cláusula SELECT

Esta cláusula indica las columnas que contendrá la tabla derivada obtenida como resultado de la ejecución de la instrucción SELECT.

Sintaxis:

SELECT [ALL | DISTINCT] lista_select
FROM lista_tablas

Donde:

ALL→ Palabra reservada que indica que se seleccionarán todas las filas que satisfagan las condiciones de consulta definidas, sin eliminar las filas duplicadas.

DISTINCT → Palabra reservada que indica que se eliminarán aquellas filas duplicadas del resultado de la selección.

lista_select → Representa aquel conjunto de atributos sobre los cuales se desea realizar la operación de proyección.
NOTA: En la mayoría de las veces serán columnas de las tablas

Los diferentes atributos estarán separados por comas.

La lista de elementos posibles son:

- * (asterisco) Indica que se deben seleccionar todas las columnas de las tablas especificadas en la cláusula FROM.

- Tabla.* → Este elemento indica que se seleccionarán todas las columnas de la tabla especificada.

- expresión → una expresión en SQL es cualquier dato constante o variable, o la combinación de varios de ellos mediante operadores que, tras un proceso de evaluación produce un resultado.

  NOTA : En la mayoría de los casos los operandos utilizados en la formación de estas expresiones serán <<columnas>> de las tablas de la BD.

  Los operadores aritméticos y de formato que pueden emplearse en expresiones de tipo SQL son los siguientes: +, *, -, / , || Concatenación

- expresión [alias] → Un alias es el nombre que se asignará a una >>expresión>> en una instrucción SELECT.
  Un alias se asigna a cualquiera de las expresiones vistas anteriormente.

Las expresiones válidas en SQL son las siguientes:

- ✓ columna Nombre de columna de tabla.
- ✓ número Número expresado como una constante en instrucciones SQL.
- ✓ 'Literal' Cadena de texto entre comillas (''). Este literal se expresa como constante en expresiones SQL y puede representar una fecha o cadena de caracteres alfanuméricos.
- ✓ - expresión En expresiones numéricas invierte el signo.
- ✓ (expresión) Evalúa en primer lugar la expresión entre paréntesis.
- ✓ expresión operador expresión Cualquier combinación de expresiones mediante los operadores aritméticos y de formato indicados anteriormente.
- ✓ función (expresiones) Cualquiera de las funciones o valores agregados admitidos por SQL
- ✓ subselect Tabla derivada devuelta por una instrucción SELECT que servirá como condicionante de una expresión.

Cláusula FROM

Determina la tabla o lista de las tablas, de las que se desean leer los datos.

Si se especifica más de una tabla en esta cláusula, la tabla derivada resultante será el producto cartesiano de estas si no se enlazan en la cláusula WHERE.

FROM lista_tablas → lista de tablas separadas por comas.
Donde cada una de las tablas puede especificarse con la siguiente sintaxis:

tabla [alias]

tabla → Nombre de la tabla perteneciente a la base de datos en curso de la cual se van a extraer datos.
alias → nombre alternativo asignado a la tabla.

Funciones del SQL

Una función manipula los contenidos de una columna en una orden SQL. Cuando en una orden SQL se utiliza una función, el valor de la columna a la que se aplica la función cambia cuando se presenta en pantalla el valor de la columna.

Tabla DUAL: Es una tabla de trabajo de Oracle, creada para probar funciones o para hacer cálculos rápidos.

Funciones comunes con datos numéricos

Ceil (n)	Aproxima al entero mayor o igual al número n. Ej: select ceil(10.6) from dual;
Floor (n)	Devuelve el mayor entero igual o menor a n.
Abs (n)	Devuelve el valor absoluto de un número. Ej: select abs (-321) from dual;
Mod (m, n)	Devuelve el resto de dividir m entre n. Ej: select mod (7,5) from dual;
Power (m, n)	Devuelve el número m elevado a la potencia n. Ej: select power (3,2) from dual;
Round (m, n)	Redondea el número m con n dígitos a la derecha a partir del punto decimal. Ej: select round (3123.6789 ,2) from dual;
Sign(n)	Si n=0, devuelve 0 Si n>0, devuelve 1 Si n<0, devuelve –1 Ej: select sign (12) from dual;
Sqrt (n)	Devuelve la raíz cuadrada de n. Ej: Select sqrt (25) from dual;
Trunc (número, [m])	Trunca números para que tengan una cierta cantidad de dígitos de precisión.
Variance (valor)	Devuelve la varianza de un conjunto de valores.

Función NVL

NVL (valor, expresión)	Esta función se utiliza para sustituir un valor nulo por otro valor. Si "valor" es NULL, es sustituido por la "expresión"; si no lo es, la función devuelve "valor". NVL se puede usar con cualquier tipo de datos: numéricos, carácter, tipo fecha. Con esta función se evitan los valores nulos en expresiones aritméticas y funciones, ya que siempre darán como resultado un valor nulo.

Funciones comunes para datos de tipo carácter

initcap (cad)	Pone en mayúscula el primer carácter de cada cadena de caracteres. Ej: Select initcap ('sr garcia') from dual;
upper (cad)	Pasa a mayúsculas la cadena de caracteres completa.
lower (cad)	Pasa a minúsculas la cadena de caracteres completa. Ej: Select lower (población) from clientes;
Replace(cad, cad1, cad2)	En la cadena de caracteres "cad", cada ocurrencia de "cad1" se reemplaza por "cad2". Ej: Select replace ('panadero', 'pana', 'barren') from dual;
Substr (cad, m, n)	Extrae n caracteres de la cadena "cad' a partir de la posición m. Ej: Select substr ('ABCDEFD', 2, 3) from dual;
Length (cad)	Devuelve la longitud de la cadena. Ej: Select length ('supercalifragilisticoespialidoso') from dual;
Concat (cad1, cad2)	Devuelve "cad1" concatenada con "cad2".
Chr (n)	Devuelve el carácter cuyo valor en binario es equivalente a n.
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.
Instr(cad1, cad2, [,comienzo [, m]])	Devuelve la posición de la "m_ésima" ocurrencia de "cad2" en "cad1", empezando la búsqueda en la posición "comienzo".
Translate(cad, cad1, cad2)	Convierte caracteres de una cadena en caracteres diferentes, según un plan de sustitución marcado por el usuario.
ASCII (cad)	Devuelve el valor ASCII de la primera letra de la cadena.

Formatos comunes para datos de tipo fecha

Se pueden utilizar diferentes formatos de fecha. Dichos formatos se emplean para modificar el formato de presentación de una fecha.

YYYY	Año en cuatro cifras
YY	Año en dos últimas cifras
YEAR	Año en letras
MM	Mes del 0 al 12
MON	Mes 3 primeras letras
MONTH	Mes en letras
WW	Semana del año
W	Semana del mes
DDD	Día del año
DD	Día del mes
D	Día de la semana
DAY	Día de la semana en letras
J	El nº del día desde el comienzo de la era más o menos
HH	La hora
HH24	Horas de un día
MI	Minutos
SS	Segundos

Formatos para datos numéricos

9	Devuelve el valor con el número especificado de dígitos.
0	Devuelve el valor dejando ceros al principio.
S	Representa el signo.
D	Devuelve el carácter decimal en la opción especificada.
G	Devuelve el carácter de los miles en la opción especificada.
L	Devuelve el símbolo de la moneda local en la posición especificada.

Funciones de conversión de tipos de datos

To_char	Conversión a tipo carácter: a) Numérico → Carácter **to_char (**número [, formato]**)** b)Tipo fecha → Tipo carácter **to_char (**fecha [, formato]**)**
To_number	Conversión a tipo numérico: a) Carácter → Numérico **to_number (**cadena_de_números**)** b) Fecha → Numérico **to_number (to_char (**fecha**, 'J'))**
To_date	Conversión a tipo fecha: a) Carácter → fecha **to_date (**cadena**,** [formato]**)** b) Numérico → fecha **to_date (**número**, 'J')**

Funciones Comunes para datos de tipo fecha

sysdate	Devuelve la fecha y la hora actual. Ej. : select sysdate from dual
last_day (fecha)	Devuelve el último día del mes. Ej. : Select last_day(sysdate) from dual;
add_months (fecha, n)	Suma o resta n meses a partir de la fecha dada. Ej. : Select add_months(sysdate, 5) from dual;
months_between (fecha1, fecha2)	Diferencia de meses entre dos fechas.
next_day(fecha,día)	Devuelve la fecha del día especificado de la semana después de la fecha dada.

Funciones de controles agregados

Count (*)	Devuelve el número de filas que cumplen la cláusula WHERE.
Count(distinct nom_colum)	Devuelve el nº de valores únicos (sin duplicados)en la columna especificada, de las filas que cumplen la cláusula WHERE.
Sum ([distinct] nom_colum)	Devuelve la suma de todos los valores de la columna especificada para las filas que cumplen la cláusula WHERE. Nota: Sólo para valores numéricos. En el caso de utilizar <<distinct>>, el resultado será la suma de los valores distintos de la columna especificada.
Avg ([distinct] nom_colum)	Devuelve el promedio de todos los valores de la columna especificada para las filas que cumplen la cláusula WHERE. Si utilizamos <<distinct>> el resultado será el promedio de los valores distintos.
Max (nom_colum)	Devuelve el valor máximo de la columna especificada de aquellas que cumplen la cláusula WHERE.
Min (nom_colum)	Devuelve el valor mínimo de la columna especificada de aquellas que cumplen la cláusula WHERE.

*Nota: En las funciones **sum(x), avg(x), max(x) y min(x)**
x puede ser una expresión en lugar de una columna

Función DECODE

DECODE (var**,** val1**,** cod1**,** val2**,** cod2..., valor_por_defecto**)**

Esta es una función IF – THEN – ELSE.

Si "var" **es igual** a cualquier valor de la lista ("val1", "val2",...), devuelve el correspondiente código ("cod1", "cod2",...).
En caso contrario se obtiene el valor señalado como valor por defecto ("valor_por_defecto").

"val" debe ser un dato del mismo tipo de "var".

Enlace de tablas (<<Join>>)

La operación de enlace (<<join>>) define el enlace entre tablas a través de una o más columnas en aquellas tablas cuyos valores cumplen la condición de enlace, siendo esta condición establecida mediante uno de los siguientes operadores:

=, >, <, >=, <=, (!=, <>)

Siendo el más habitual: **'='**

Por lo general el enlace (<<**join**>>) entre las tablas se realizará combinando la(s) columna(s) que componen la ***clave primaria*** de una tabla con la(s) que componen su correspondiente ***clave extranjera*** en la otra.

Ej:

```
SQL>  SELECT  clientes.empresa, albaranes.fecha_albaran
      FROM    clientes, albaranes
      WHERE   clientes.cliente = albaranes.cliente;
```

En este ejemplo se obtienen todos los clientes que tengan albaranes y sus correspondientes albaranes.

No obstante, puede haber clientes a los que aún no se les haya suministrado nada.

Éstos no aparecerán en el resultado, por lo que, en ciertos casos, podríamos decir que hay una pérdida de información, es decir, en el caso en el que se quiera listar los clientes tengan o no albaranes.

Para evitar esta pérdida de información, los lenguajes SQL disponen de los <<**outer join**>> cuya función es recuperar en una operación de enlace de tablas todas las filas de una tabla tengan o no enlace en la otra.

En SQL de Oracle, el outer join se indica con el signo '(+)', poniéndolo detrás del campo o campos de enlace de la tabla subordinada.

Ej:

```
SQL>  SELECT  clientes.empresa, albaranes.fecha_albaran
      FROM    clientes, albaranes
      WHERE   clientes.cliente = albaranes.cliente (+);
```

* En algunos SQL para hacer <<**outer join**>> se pone ***OUTER albaranes***

Características:

- Un <<**outer join**>> implica, al menos, dos tablas, una de las cuales es ***dominante*** (en la que se preserva la clave de enlace) y otra ***subordinada***.
- Todas las filas de la tabla dominante estarán presentes en la proyección resultante, **tengan o no enlace** con la tabla subordinada.
- Aquella fila que tenga enlace en dicha tabla se completará con **valores nulos** en las columnas correspondientes en la tabla subordinada.

Cláusula WHERE

Mediante esta cláusula se podrán especificar diferentes criterios de selección.

Aquellas filas que cumplan las condiciones indicadas serán las que formarán la tabla derivada.

Sintaxis:

WHERE condición

Donde:

condición → lista de una o más subcondiciones separadas por los operadores lógicos: <<AND>>, <<OR>>, <<NOT>>.

Existen dos tipos de condiciones de búsqueda:

- *Condición de comparación.*
- *Condición de subselección.*

I. Condición de comparación:

Puede definirse como cualquiera de las siguientes sintaxis:

- **expresión operador_relación expresión**

Esta condición devuelve verdadero o falso si ambas expresiones cumplen la relación establecida por el operador.

Donde:

expresión → es cualquier expresión válida en SQL
operador_relación → =, >, <, >=, <=, <> o !=

- **expresión [Not] BETWEEN expresión_desde AND expresión_hasta**

Esta condición devuelve Verdadero o Falso en caso de que el resultado de una expresión esté o no comprendido en el rango dado por otras dos, ambas inclusive.

La partícula <<NOT>> invierte resultado.

- **expresión [NOT] IN (lista_valores)**

Esta condición devuelve verdadero o falso en caso de que el resultado de una expresión sea igual o no a uno de los valores incluidos en la <<lista de valores>>.

La partícula <<NOT>> invierte el resultado.

Donde:

lista_valores → Es una lista de valores separados por comas.

Estos valores serán numéricos o alfanuméricos, dependiendo el tipo de dato de <<expresión>>.

Los valores de los tipos de datos carácter y fecha tienen que ir entre comillas.

- **expresión [NOT] LIKE 'literal'**

Condición de comparación de expresiones de tipo alfanumérico, sólo aplicable a columnas de tipo carácter.

Devuelve Verdadero o falso en caso de que el resultado de una expresión se ajuste o no al patrón definido en <<literal>>.

Donde:

'literal' → Este literal puede contener dos caracteres especiales, pudiendo especificarse cada uno de ellos más de una vez.
Estos son:

% → El lugar donde aparezca este carácter se podrá sustituir por ningún carácter, por uno o por más de uno.

_(subrayado) → Este carácter indica que debe sustituirse obligatoriamente por un carácter en el lugar que ocupe.

- **expresión IS [NOT] NULL**

Devuelve verdadero o falso en caso de que el resultado de una expresión sea o no un valor nulo.

La partícula <<NOT>> invierte el resultado.

En caso de no utilizar este operador (is null) para la detección de valores nulos, no se recuperarán las filas cuyos valores en la columna a la que se aplica la condición sean nulos (<<NULL>>).

II. Condición de Subselección

Una condición de <<subselect>> consiste en comparar una expresión, que por lo general será una columna, con una tabla derivada devuelta por otra instrucción SELECT.

Existen tres formas de realizar condiciones de <<subselect>>:

- **expresión operador_relación {ALL | [ANY | SOME]} (subselect)**

  Esta condición devuelve verdadero o falso dependiendo de si se cumple o no cierta relación entre el resultado de una <<expresión>> y el resultado de un <<subquery>> en función del operador de relación que intervenga.

  Donde:

  expresión → cualquier expresión válida

  operador_relación → =, >, <, >=, <=, (!=, <>)

  subselect → Instrucción SELECT cuya <<lista_select>> sólo podrá contener una columna en la tabla derivada que genere.

  Esta columna tendrá que ser del mismo tipo de dato SQL que la columna que condiciona <<expresión>>.

  ALL → Palabra reservada que indica que el resultado de la comparación debe ser verdadero para cada uno de los valores devueltos por la <<subselect>>.

  ANY→ Palabra reservada que indica que el resultado de la comparación debe ser verdadero, al menos para uno de los valores devueltos por la instrucción <<subselect>>.

  SOME → sinónimo de **ANY**

- **Expresión [NOT] IN (subselect)**

  Esta condición devuelve verdadero o falso dependiendo de si cumple o no cierta relación entre <<expresión>> y el resultado de la expresión <<subselect>>.

  Donde:

  NOT → Invierte el resultado
  IN → Palabra reservada que pregunta si <<expresión>> es alguno de los valores devueltos por <<subselect>>.

NOTA:
1. La partícula <<IN>> puede simularse mediante <<=ANY>>.
2. La partícula <<NOT IN>> puede reemplazarse por <<!=ALL>> (<> ALL)

- **[NOT] EXISTS (subselect)**

  Esta condición devuelve Verdadero o Falso dependiendo de si existe o no alguna fila resultado de la instrucción <<subselect>>.

  La partícula NOT invertirá el resultado de la condición.

  Donde:

  EXISTS → Palabra reservada que evalúa la existencia o no de alguna fila en la tabla derivada de la instrucción <<subselect>>.

  Subselect→ Instrucción SELECT válida.
  La "lista select" de esta instrucción, puede contener más de una columna o expresión, ya que ésta no se utilizará para la comparación de expresiones de la condición.

Cláusula GROUP BY

Esta cláusula devuelve una fila a partir de un conjunto de filas que tienen un denominador común para las columnas indicadas en ella.

Sintaxis:

GROUP BY lista_grupos

Donde:

GROUP BY → Palabras reservadas

lista_grupos → Nombre de una columna o lista de nombres de columnas separados por comas, que determinan el grupo.

NOTAS:

1. En la <<lista_select>> de la instrucción SELECT no se pueden indicar columnas que no estén incluídas en la cláusula <<GROUP BY>>.

2. Cuando en la <<lista_select>> existen funciones de valores agregados junto a más columnas se debe utilizar la cláusula <<GROUP BY>>.

Cláusula HAVING

Va unida a la cláusula GROUP BY.

(Por lo general, sólo aparecerá si aparece el group by)

Establece condiciones de selección sobre los grupos determinados por la cláusula <<GROUP BY>>.

Sintaxis:

HAVING condición_grupos

Donde:

HAVING → Palabra reservada.

Condición_grupos →cualquier condición, pero es necesaria una función agregada dentro de la misma.

Cláusula ORDER BY

Esta cláusula ordena el resultado de la selección por el valor de una o más columnas de forma ascendente o de forma descendente.

Esta tabla derivada sólo puede ordenarse por las columnas que intervienen en la <<selección>> de la instrucción SELECT.

Sintaxis:

ORDER BY columna [ASC/DESC] [, columna [ASC/DESC] [,...]]

Donde:

ORDER BY → palabra reservada

Columna→ Nombre de una columna o alias o la posición que ocupa la columna en la selecciónsiendo obligatorio este caso para las expresiones que no sean columnas simples.

ASC → Para ordenar de forma ascendente.

DESC → Para ordenar de forma descendente.

Cláusula UNION

Esta cláusula devuelve una tabla derivada compuesta por el resultado de varias instrucciones SELECT concatenadas o unidas. (Unión)

Sintaxis:

SELECT

[UNION [ALL]]

SELECT

[UNION [ALL]]

...

[ORDER BY ...]

Donde:

UNION→ palabra reservada que indica la unión de los resultados de la instrucciones SELECT que incluyan.

ALL→ La tabla derivada no contendrá filas duplicadas a menos que se especifique esta palabra reservada.

Las características de la cláusula UNION son las siguientes:

- El número de columnas y los tipos de las columnas correspondientes de todas las cláusulas <<lista_select>> de los SELECT deben ser idénticas.
- En la cláusula ORDER BY las columnas tienen que referenciarse por número y no por nombre.
- Los nombres de las columnas de la tabla derivada resultante coinciden con los de la primera instrucción SELECT.

Cláusula INTERSECT

Devuelve una tabla derivada compuesta por las filas que se obtienen en ambas instrucciones SELECT. (Intersección)

Sintaxis:

SELECT

[INTERSECT]

SELECT

[INTERSECT]

...

[ORDER BY]

Cláusula MINUS

Cuando se utiliza esta cláusula se obtienen todas las filas de la primera tabla derivada, menos las filas de dicha tabla que también estén en la segunda tabla derivada. (Resta)

Sintaxis:

SELECT

[MINUS]

SELECT

[MINUS]

...

[ORDER BY...]

DML (Lenguaje de Manipulación de Datos)

Este sublenguaje del SQL contempla las siguientes operaciones:

- Inserción de filas.
- Borrado de filas.
- Modificación de filas.

INSERCIÓN DE FILAS

SQL permite la inserción de valores sobre todas las columnas de una tabla o bien de parte de ellas.

La instrucción que permite la inserción de filas en las tablas es INSERT.

Esta instrucción tiene dos posibles sintaxis:

a) Inserción de una sola fila:

INSERT INTO tabla [(columnas)] VALUES (valores)

Donde:

tabla Nombre de una tabla de la B.D.

columnas Nombre de la columna o columnas pertenecientes a la tabla indicada anteriormente.

valores Valores a insertar en cada una de las columnas especificadas anteriormente.

b) Inserción de un bloque indeterminado de filas (tabla derivada):

INSERT INTO tabla [(columnas)] instrucción_select

Borrado de filas

SQL permite el borrado de filas de aquellas filas que cumplen las condiciones impuestas por la instrucción DELETE, cuya sintaxis es la siguiente:

Sintaxis:

```
DELETE    FROM      tabla
[WHERE    {condición_comparación | condición-subselect}]
```

Orden TRUNCATE

Podemos también borrar la totalidad de las filas de una tabla con la instrucción TRUNCATE.

Si utilizamos esta opción no podremos realizar un rollback posteriormente para deshacer los cambios realizados.

Sintaxis:

```
TRUNCATE  TABLE   nombre_tabla;
```

Modificación de Filas

SQL permite la modificación de todos los valores de todas las columnas de una tabla o bien de parte de ellos.

La instrucción que se usa es UPDATE:

Sintaxis:

```
UPDATE tabla
SET columna = expresión [, columna = expresión ] [,... ]
[ WHERE { condición_comparación  |  condición_subselect } ]
```

En caso de intentan actualizar una fila bloqueada por otro usuario, la instrucción UPDATE quedará a su vez bloqueada hasta que dicha fila se libere.

(DCL) Lenguaje de Control de Acceso a Datos

Dentro de la definición de una base de datos, el sublenguaje de control de acceso a datos del SQL contempla los siguientes procesos:

- Permisos
- Bloqueos

Uno de los aspectos a tener en cuenta en el diseño de una B.D. es definir quienes pueden acceder y a qué datos, así como el acceso a qué se autoriza.

PRIVILEGIOS DEL SISTEMA

Los privilegios del sistema son lo que dan derecho a ejecutar un tipo de comando SQL o a realizar alguna acción sobre objetos de un tipo especificado.
Algunos de estos privilegios son:

PRIVILEGIO DEL SISTEMA	OPERACIONES AUTORIZADAS
	SESSION
CREATE SESSION	Conectarnos a la base de datos.
ALTER SESSION	Manejar la orden ALTER SESSION.
	TABLE
CREATE TABLE	Crear tablas y generar índices sobre dichas tablas.
CREATE ANY TABLE	Crear una tabla en cualquier esquema (usuario).
ALTER ANY TABLE	Modificar una tabla en cualquier esquema.
SELECT ANY TABLE	Hacer SELECT en cualquier tabla.
INSERT ANY TABLE	Insertar filas en cualquier tabla.
UPDATE ANY TABLE	Modificar filas en cualquier tabla.
DELETE ANY TABLE	Borrar filas de cualquier tabla.
	VIEW
CREATE VIEW	Crear vistas en el esquema propio.
CREATE ANY VIEW	Crear vistas en cualquier esquema.
DROP ANY VIEW	Borrar vistas en cualquier esquema.
	ROLE
CREATE ROLE	Crear roles.
GRANT ANY ROLE	Dar permisos para cualquier rol de la base de datos
	PRIVILEGE
GRANT ANY PRIVILEGE	Conceder cualquier privilegio del sistema.
	USER
CREATE USER	Crear usuarios.
ALTER USER	Modificar cualquier usuario.
DROP USER	Eliminar usuarios.
	SYNONYM
CREATE SYNONYM	Crear sinónimos en nuestro esquema.
CREATE PUBLIC SYNONYM	Crear sinónimos públicos.
	SEQUENCE
CREATE SEQUENCE	Crear secuencias en nuestro esquema.

El usuario propietario (administrador) de la B.D., será el que conceda o retire los privilegios del sistema a los usuarios, con las instrucciones GRANT y REVOKE.

ROLES

Un rol es un conjunto de privilegios usando la orden **GRANT**.

El formato para crear un rol es:

CREATE ROLE Nombre_Rol

Para crear un rol se requiere el privilegio del sistema CREATE ROLE.

Para borrar un rol sería:

DROP ROLE Nombre_Rol

Existen 3 roles o niveles de acceso sobre la B.D. por defecto:

DBA son los administradores de la B.D. los que lo contienen y su acceso es absoluto en todos los niveles.

Un usuario DBA puede conceder o retirar cualquier permiso a cualquier otro usuario.
El creador de la B.D. se convierte automáticamente en usuario DBA.

Ej. :
SYSTEM, SYS

RESOURCE este nivel de acceso es el inmediatamente inferior a DBA. Y como característica principal permite la creación de procedimientos y triggers en la B.D..

CONNECT este es el nivel más bajo, permite realizar las operaciones necesarias para conectarse a la B.D., creación de los objetos básicos de una B.D., como son las tablas y vistas y su posterior manipulación.

Sintaxis: (Concesión de permios del sistema y roles)

GRANT {privilegio | rol} [, …] TO { PUBLIC | lista_usuarios | rol}

Sintaxis: (Retirada de permisos del sistema o roles)

REVOKE {privilegio | rol} [, …] FROM {PUBLIC |lista_usuarios| rol}

Donde:

PUBLIC Palabra reservada que indica la cesión de permisos a todos los usuarios.

lista_usuarios Lista de usuarios separados por comas.

PERMISOS SOBRE OBJETOS (TABLAS)

Se dispone de los siguientes privilegios sobre dichos objetos:

SELECT	Lectura de la tabla.
UPDATE	Actualización de la tabla.
INSERT	Inserción de las filas en la tabla.
DELETE	Borrado de filas en la tabla.
INDEX	Creación de índices para dicha tabla.
ALTER	Alteración de la estructura de la tabla.
ALL	Engloba todos los permisos anteriores.

Cada usuario propietario podrá conceder ó retirar los permisos sobre sus objetos a los demás usuarios.

Para conceder y retirar estos permisos a los usuarios sobre las tablas especificadas utilizaremos las instrucciones GRANT y REVOKE de la siguiente forma:

Sintaxis: (Concesión y retirada de los permisos sobre los objetos)

GRANT permisos ON tabla TO {PUBLIC | lista_usuarios | rol}

REVOKE permisos ON tabla FROM {PUBLIC | lista_usuarios | rol}

Donde:

permisos	lista de permisos separados por comas.
tabla	tabla sobre la que se conceden o retiran los permisos.
PUBLIC	Palabra reservada que indica la cesión de permisos a todos los usuarios.
lista_usuarios	Lista de usuarios separados por comas.

VISTAS CON INFORMACIÓN DE LOS PRIVILEGIOS

Para conocer los privilegios que han concedido o recibido los usuarios sobre los objetos o a nivel del sistema, podemos consultar las siguientes vistas del diccionario de datos:

SESSION_PRIVS: privilegios del usuario activo.

USER_SYS_PRIVS: privilegios del sistema asignados al usuario.

DBA_SYS_PRIVS: privilegios de sistema asignados a los usuarios o a los roles.

USER_TAB_PRIVS: concesiones sobre objetos que son propiedad del usuario, concedidos o recibidos por éste.

USER_TAB_PRIVS_MADE: concesiones sobre objetos que son propiedad del usuario (asignadas).

USER_TAB_PRIVS_RECD: concesiones sobre objetos que recibe el usuario.

INFORMACIÓN SOBRE ROLES EN EL DICCIONARIOS DE DATOS

Para saber a qué usuarios se ha concedido acceso a un rol, o los privilegios que se han concedido a un rol, se puede consultar las siguientes vistas:

ROLE_SYS_PRIVS: privilegios de sistema asignados a roles.

ROLE_TAB_PRIVS: privilegios sobre tablas aplicados a roles.

ROLE_ROLE_PRIVS: roles asignados a otros roles.

SESSION_ROLES: roles activos para el usuario.

USER_ROLE_PRIVS: roles asignados al usuario.