



2. Altres objectes

NF2. Llenguatge SQL : DDL

UF2: Llenguatge SQL: DML i DDL

Desenvolupament d'Aplicacions Multiplataforma

M02 – Bases de dades. Versió 1.0

© M^a Carmen Brito Ruiz

2.1. Objetos de la base de datos

2.2. Vistas

2.2.1. Creación y eliminación de vistas

2.2.2. Algunos ejemplos de vistas

2.3. Nombres alternativos: definición para tablas y vistas

2.4. Índices

2.4.1. Creación y eliminación de índices

2.4.2. Algunos ejemplos de índices

2.5. Secuencias

2.6. Tablas del sistema

2.7. Usuarios

2.1. Objetos de la base de datos

Objeto	Descripción
Tabla	Unidad básica de almacenamiento.
Vista	Representación lógica de datos de una o más tablas.
Secuencia	Genera valores de clave primaria (autonuméricos).
Índice	Mejora el rendimiento de consultas.
Sinónimo	Nombre alternativo para un objeto.

2.2. Vistas

Las **vistas**, también llamadas tablas virtuales, no contienen datos almacenados, sino que combinan datos seleccionados de una o varias tablas.

2.2.1. Creación y eliminación de vistas

Para crear una vista tenemos el mandato CREATE [OR REPLACE] VIEW, cuya sintaxis es la siguiente:

```
CREATE [OR REPLACE] VIEW <nombre_vista>
  (atributo1, atributo2, atributo3)
AS
  SELECT <campo1, campo2, campoN>
  FROM <nombre_tabla>
  [WHERE condición]
  [GROUP BY]
  [ORDER BY];
```

➤ Vista que combine dos columnas:

Para crear una vista que combine dos columnas (dos campos) de una tabla, sería la siguiente sentencia:

```
CREATE [OR REPLACE] VIEW <nombre_vista>
(atributo1, atributo2, atributo3, atributo4)
AS
SELECT campo1, campo2+campo3, campo4, campo5
FROM <nombre_tabla>;
```

➤ Vista que combine varias filas y columnas:

Para crear una vista que combine varias filas y columnas de una o más de una tabla, tenemos la siguiente sintaxis:

```
CREATE [OR REPLACE] VIEW <nombre_vista>
(atributo1, atributo2, atributo3)
AS
SELECT campo1, campo2_tabla1, campo3_tabla2
FROM <nombre_tabla1>,<nombre_tabla2>
[WHERE nombre_tabla1.condición=
      nombre_tabla2.condición];
```

➤ Vista que combine todas las filas y columnas de una tabla:

Para crear una vista que combine varias filas y columnas de una o más de una tabla, tenemos la siguiente sintaxis:

```
CREATE [OR REPLACE] VIEW <nombre_vista>
AS
SELECT *
FROM <nombre_tabla1>,<nombre_tabla2>
[WHERE nombre_tabla1.condición=
        nombre_tabla2.condición];
```

➤ Consultar vista:

Las vistas, se pueden visualizar como si fuera una tabla; para ello, tenemos la siguiente sintaxis:

```
SELECT * FROM <nombre_vista>;
```

➤ Eliminar vista:

Las vistas se suprimen automáticamente con la eliminación de las tablas base usadas para crearlas o bien, con DROP VIEW. La sintaxis es la siguiente:

```
DROP VIEW <nombre_vista>;
```


➤ Modificar vista:

Para modificar una vista, se ha de inclur en el CREATE VIEW la cláusula OR REPLACE. La sintaxis es la siguiente:

```
CREATE [OR REPLACE] VIEW <nombre_vista>
AS
  SELECT <campo1, campo2, campoN>
  FROM <nombre_tabla>
  [WHERE condición]
  [GROUP BY]
  [ORDER BY];
```

➤ Denegar operaciones DML en una vista

Cuando se modifica datos de una vista, se modifica estos datos en la tabla física. Es decir, si elimino i/o modifico datos de una vista, quedará afectada la tabla a la que está asociada.

Por tanto, hemos de usar cláusulas para restringir las modificaciones y/o eliminaciones:

WITH CHECK OPTION CONSTRAINT

WITH READ ONLY

Ejemplo:

SELECT * FROM dept;

DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON

CREATE OR REPLACE VIEW vista_dept
AS

SELECT *
FROM dept;

Vista creada.

DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON

SELECT * FROM vista_dept;

DELETE FROM vista_dept
WHERE deptno = 40;

1 fila suprimida.

DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO

SELECT * FROM vista_dept;

SELECT * FROM dept;

DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO

➤ Denegar operaciones DML en una vista

Reglas para realizar operaciones DML en una vista:

- Si es una vista simple, se puede realizar operaciones DML.
- No se puede eliminar una fila si la vista contiene: funciones de grupo, cláusula GROUP BY, la cláusula DISTINCT y ROWNUM.
- No se puede modificar datos en una vista, si contiene: funciones de grupo, cláusula GROUP BY, la cláusula DISTINCT y ROWNUM. Además, de columnas definidas por expresiones.
- No se puede agregar datos a través de una vista, si contiene: funciones de grupo, cláusula GROUP BY, la cláusula DISTINCT y ROWNUM. Además, de columnas definidas por expresiones y columnas NOT NULL en las tablas base que no estén seleccionadas por la vista.

➤ Denegar operaciones DML en una vista ⇒ WITH CHECK OPTION

La cláusula especifica que las inserciones y las modificaciones realizadas mediante la vista no pueden crear filas que la vista no pueda seleccionar. Por tanto, permite que las restricciones de integridad y las comprobaciones de validación de datos se fuercen en los datos que se insertan o se actualizan.

La sintaxis es:

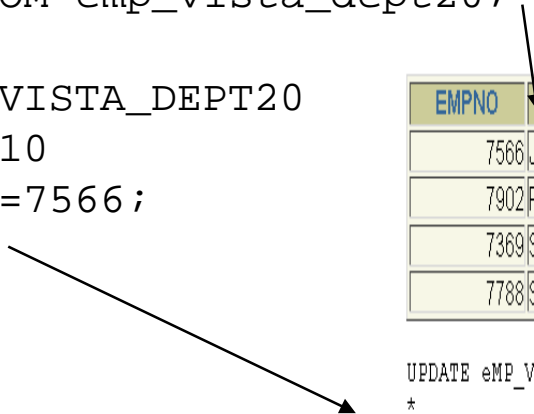
```
CREATE [OR REPLACE] VIEW <nombre_vista>
(atributo1, atributo2, atributo3)
AS
SELECT <campo1, campo2, campoN>
FROM <nombre_tabla>
[WHERE condición]
WITH CHECK OPTION CONSTRAINT nombre_restriccion;
```

Ejemplo WITH CHECK OPTION

```
CREATE OR REPLACE VIEW emp_vista_dept20
AS
  SELECT *
  FROM emp
  WHERE deptno=20
  WITH CHECK OPTION CONSTRAINT empvista20_ck;
```

```
SELECT * FROM emp_vista_dept20;
```

```
UPDATE eMP_VISTA_DEPT20
SET deptno=10
WHERE empno=7566;
```



EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7566	JONES	MANAGER	7839	02/04/81	2975		20
7902	FORD	ANALYST	7566	03/12/81	3000		20
7369	SMITH	CLERK	7902	17/12/80	800		20
7788	SCOTT	ANALYST	7566	09/12/82	3000		20

```
UPDATE eMP_VISTA_DEPT20
*
```

ERROR en línea 1:
ORA-01402: violación de la cláusula WHERE en la vista WITH CHECK OPTION

➤ Denegar operaciones DML en una vista ⇒ WITH READ ONLY

En este caso cualquier intento de eliminar una fila de una vista con una restricción de sólo lectura, dará error.

La sintaxis es:

```
CREATE [OR REPLACE] VIEW <nombre_vista>
    (atributo1, atributo2, atributo3)
AS
    SELECT <campo1, campo2, campoN>
    FROM <nombre_tabla>
    [WHERE condición]
    WITH READ ONLY;
```

Ejemplo WITH READ ONLY

```
CREATE OR REPLACE VIEW emp_vista_dept10
```

```
AS
```

```
  SELECT *
```

```
  FROM emp
```

```
  WHERE deptno=10
```

```
  WITH READ ONLY;
```

```
SELECT *
```

```
FROM emp_vista_dept10;
```

```
DELETE FROM emp_vista_dept10
```

```
WHERE empno = 7839;
```

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7839	KING	PRESIDENT		17/11/81	5000		10
7782	CLARK	MANAGER	7839	09/06/81	2450		10
7876	ADAMS	CLERK	7788	12/01/83	1100		10
7934	MILLER	CLERK	7782	23/01/82	1300		10

DELETE FROM emp_vista_dept10
*

ERROR en línea 1:

ORA-01752: no se puede suprimir de la vista sin una tabla reservada mediante clave

2.2.2. Algunos ejemplos de vistas

a) *Crear una vista que contenga los socios cuya cuota sea mayor a 3000 ptas. Ha de aparecer el código del socio, nombre, apellidos y dirección.*

```
CREATE VIEW vista_soc_cuota  
  (Codigo, Nombre, Apellidos, Direccion)  
AS  
  
  SELECT cod_soc, nombre, apellidos, direccion  
  FROM socio  
  
  WHERE cuota<3000;
```

b) *Crear una vista de todas las películas que tenemos en nuestra base de datos.*

```
CREATE OR REPLACE VIEW vista_películas
AS
SELECT *
FROM película;
```

c) *Crear una vista de todas las películas (código y título) que tenemos en nuestra base de datos.*

```
CREATE OR REPLACE VIEW vista_películas2
(Codigo, Título)
AS
SELECT cod_pel, título
FROM película;
```

2.3. Nombres alternativos: definición para tablas y vistas

La idea es definir nombres que se pueden usar para sustituir nombres de tablas y de vistas, con sentencias SQL. Se usa para simplificar y acortar las entradas.

La sintaxis es la siguiente:

```
CREATE [PUBLIC] SYNONYM nombre_sinonimo  
FOR nombre_tabla/nombre_vista;
```

Para eliminar el nombre alternativo que se le ha asignado, tenemos la siguiente sintaxis:

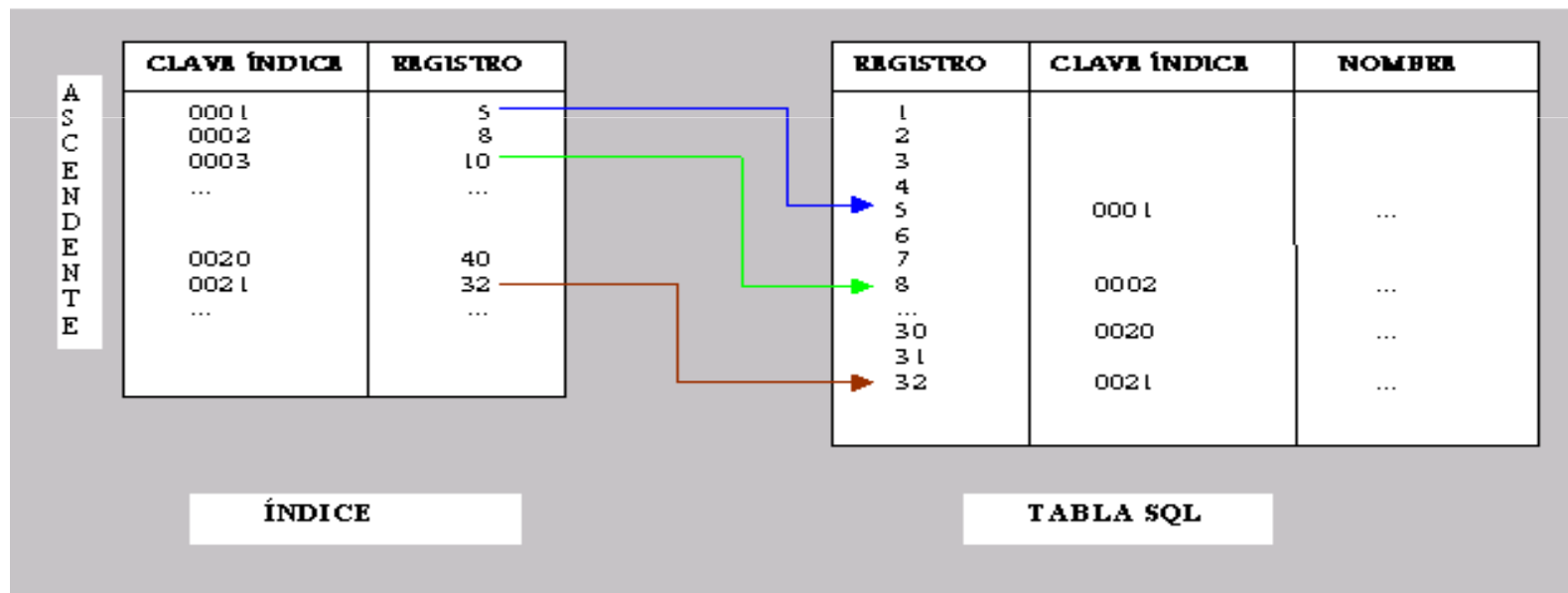
```
DROP SYNONYM nombre_sinonimo;
```

Ejemplo para una tabla:

```
CREATE SYNONYM depart FOR departments;
```

2.4. Índices

Los índices se crean a partir de la columna o columnas más empleadas en la especificación de sentencias. Estos **índices** se usan en SQL por la misma razón que en los ficheros de bases de datos de dBASE, ACCESS, ORACLE, etc.; para tener un “**acceso rápido a los datos**”.



2.4.1. Creación y eliminación de índices

➤ Crear índice a partir de una columna:

```
CREATE INDEX nombre_indice  
ON nombre_tabla (campo);
```

Destacar que a la hora de crear el índice...

- Por defecto, se define en orden ascendente. Si se desea un orden descendente, se ha de indicar y la sintaxis es la siguiente:

```
CREATE INDEX nombre_indice DESC  
ON nombre_tabla (campo);
```

- Si se usa más de una columna en la clave de índice no es posible especificar ASC y DESC a la vez.

➤ Crear índex únic:

En algún momento, nos interesa especificar que el índice ha de contener valores únicos (impidiendo introducir filas duplicadas), para ello tenemos la siguiente sintaxis:

```
CREATE UNIQUE INDEX nombre_indice  
ON <nombre_tabla> (campo);
```

➤ Eliminar índice:

Para eliminar un índice creado previamente, tenemos la siguiente sintaxis:

```
DROP INDEX alias_campo;
```

2.4.2. Algunos ejemplos de índices

a) Crear un índice para el campo *apellidos* de la tabla *Socio*.

```
CREATE INDEX ind_apell  
ON socio (apellidos);
```

b) Crear un índice único para el campo población de la tabla *Socio*.

```
CREATE UNIQUE INDEX ind_poblac  
ON socio (poblacion);
```

c) Eliminar el índice del campo *población* de la tabla *Socio*.

```
DROP INDEX ind_poblac;
```

2.5. Secuencias

Una secuencia se usa para generar de manera automática números únicos y por tanto, se recomienda usar este valor para claves primarias.

La sintaxis es la siguiente:

```
CREATE SEQUENCE nombre_secuencia
[INCREMENT BY n]          /* incremento de la secuencia */
[START WITH n]            /* valor inicial de la secuencia */
[{MAXVALUE n} | NOMAXVALUE] /* valor máximo de la secuencia */
[{MINVALUE n} | NOMINVALUE] /* valor mínimo de la secuencia */
[{CYCLE | NOCYCLE}]       /* especifica si la secuencia continua
                           generando valores cuando llega al máximo */
[{CACHE n} | NOCACHE]     /* especifica los valores que preasigna Oracle
                           Server y cuántos mantiene en memoria.
                           Por defecto es 20 valores en caché */
```




Ejemplo de creación de una secuencia

```
CREATE SEQUENCE dept_seq_deptno  
    INCREMENT BY 10  
    START WITH 100  
    MAXVALUE 9999  
    NOCACHE  
    NOCYCLE ;
```

➤ Comprobar la secuencia:

```
SELECT sequence_name, min_value, max_value,  
       increment_by, last_number  
FROM user_sequences;
```

SEQUENCE_NAME	MIN_VALUE	MAX_VALUE	INCREMENT_BY	LAST_NUMBER
DEPT_SEQ_DEPTNO	1	9999	10	100

➤ Cláusulas NEXTVAL y CURRVAL de las secuencias:

NEXTVAL: devuelve el siguiente valor de secuencia disponible.

CURRVAL: devuelve el valor actual de la secuencia.

Ejemplo:

```
INSERT INTO dept
VALUES (dept_seq_deptno.nextval, 'INFORMATICA', 'BCN');
```

```
INSERT INTO dept
VALUES (dept_seq_deptno.nextval, 'MARKETING', 'BCN');
```

```
SELECT dept_seq_deptno.currval
FROM dual;          /* devuelve el siguiente valor de la secuencia */
```

➤ Modificar una secuencia:

```
ALTER SEQUENCE nombre_secuencia  
    INCREMENT BY valor_nuevo  
    MAXVALUE valor_nuevo  
    NOCACHE  
    NOCYCLE;
```

➤ Eliminar una secuencia:

```
DROP SEQUENCE nombre_secuencia;
```

2.6. Tablas del sistema

Las tablas base del sistema son las tablas subyacentes que almacenan los metadatos para una base de datos específica. Hay muchas tablas del sistema, en esta ocasión vamos a estudiar las tablas donde se almacenan los objetos que hemos estudiado en nuestro módulo.

- catálogo
- restricciones
- índices
- sinónimos
- secuencias



➤ Consulta catàlogo

Con esta consulta se visualiza las tablas/vistas/sinónimos de la base de datos en la que estamos trabajando.

```
SELECT *  
FROM cat;
```

➤ Consulta restriccions

Para visualizar las restricciones, se ha de consultar la tabla USER_CONSTRAINTS.

```
SELECT constraint_name, constraint_type, search_condition  
FROM user_constraints;
```

```
SELECT constraint_name, constraint_type, search_condition  
FROM user_constraints  
WHERE table_name = 'EMP';
```

DESC USER_CONSTRAINTS.

Nombre	¿Nulo?	Tipo
OWNER	NOT NULL	VARCHAR2(30)
CONSTRAINT_NAME	NOT NULL	VARCHAR2(30)
CONSTRAINT_TYPE		VARCHAR2(1)
TABLE_NAME	NOT NULL	VARCHAR2(30)
SEARCH_CONDITION		LONG
R_OWNER		VARCHAR2(30)
R_CONSTRAINT_NAME		VARCHAR2(30)
DELETE_RULE		VARCHAR2(9)
STATUS		VARCHAR2(8)
DEFERRABLE		VARCHAR2(14)
DEFERRED		VARCHAR2(9)
VALIDATED		VARCHAR2(13)
GENERATED		VARCHAR2(14)
BAD		VARCHAR2(3)
RELY		VARCHAR2(4)
LAST_CHANGE		DATE
INDEX_OWNER		VARCHAR2(30)
INDEX_NAME		VARCHAR2(30)
INVALID		VARCHAR2(7)
VIEW_RELATED		VARCHAR2(14)

➤ Consulta las columnas asociadas a las restricciones:

Para visualizar las columnas asociadas a los nombres de restricciones, se usa la vista USER_CONS_COLUMNS.

```
SELECT constraint_name, column_name  
FROM user_cons_columns;
```

```
SELECT constraint_name, column_name  
FROM user_cons_columns;  
WHERE table_name = 'EMP';
```

➤ Consulta índices

Para visualizar los datos de los índices creados, se usa USER_INDEXES.

```
SELECT  index_name,  index_type,  table_owner,  table_name,  
        table_type  
FROM    user_indexes;
```

➤ Consulta sinónimos

Para visualizar los sinónimos, se usa USER_SYNONYMS.

```
SELECT *  
FROM user_synonyms;
```

```
DESC user_synonyms;
```

Nombre	¿Nulo?	Tipo
SYNONYM_NAME	NOT NULL	VARCHAR2(30)
TABLE_OWNER		VARCHAR2(30)
TABLE_NAME	NOT NULL	VARCHAR2(30)
DB_LINK		VARCHAR2(128)

➤ Consulta secuencias

Para visualizar los datos de las secuencias creadas, se usa USER_SEQUENCES.

```
SELECT sequence_name, min_value, max_value,  
       increment_by, last_number  
FROM user_sequences;
```

2.7. Usuarios

El acceso a usuarios es controlado por el DBA (Administrador de la base datos) que tiene privilegios del sistema de alto nivel, para poder realizar tareas como:

- Crear nuevos usuarios (**CREATE USER**),
- Eliminar usuarios (**DROP USER**),
- Eliminar tablas,
- Realizar copias de seguridad de las tablas, etc.

Por ejemplo en Oracle:

- Unidad básica de almacenamiento de una base de datos ⇒ Tabla.
- Las tablas se agrupan dentro ⇒ usuarios (esquemas).
- un usuario puede tener cero, una o muchas tablas y es propietario de ellas.
- al crear una base de datos se crean dos usuarios:

SYS y SYSTEM

Estos usuarios son propietarios de las tablas del diccionario de la base de datos. En caso de borrar alguno de estos usuarios, se puede corromper la base de datos.

Preguntes!!!!

