Vistas

1.- Introducción

Una vista no es más que ***una consulta almacenada a fin de utilizarla tantas veces como se desee***.

Una vista **NO contiene datos *sino la instrucción SELECT necesaria para crear la vista***, eso asegura que los datos sean coherentes al utilizar los datos almacenados en las tablas. Por todo ello, las vistas gastan muy poco espacio de disco.

Las vistas **se emplean para**:

- Realizar consultas complejas más fácilmente, ya que permiten dividir la consulta en varias partes.
- Proporcionar tablas con datos completos.
- Utilizar visiones especiales de los datos.
- Ser utilizadas como tablas que resumen todos los datos.
- Ser utilizadas como cursores de datos en los lenguajes procedimentales (como PL/SQL).

Hay **dos tipos de vistas**:

- ***Simples***: Las forman una sola tabla y no contienen funciones de agrupación. Su ventaja es que permiten siempre realizar operaciones DML sobre ellas.
- ***Complejas:*** Obtienen datos de varias tablas, pueden utilizar funciones de agrupación. No siempre permiten operaciones DML.

2.- Creación de Vistas

Sintaxis:

```
CREATE [OR REPLACE] [FORCE | NOFORCE] VIEW vista
        [ (alias [, alias2...] )]
AS consultaSELECT
[WITH CHECK OPTION [CONSTRAINT restricción]]
[WITH READ ONLY [CONSTRAINT restricción]]
```

- **OR REPLACE:** Si la vista ya existía, la cambia por la actual.
- **FORCE:** Crea la vista aunque los datos de la consulta SELECT no existan.
- **vista:** Nombre que se le da a la vista
- **alias:** Lista de alias que se establecen para las columnas devueltas por la consulta SELECT en la que se basa esta vista. El número de alias debe coincidir con el número de columnas devueltas por SELECT.

- **WITH CHECK OPTION:** Hace que sólo las filas que se muestran en la vista puedan ser añadidas (INSERT) o modificadas (UPDATE). La restricción que sigue a esta sección es el nombre que se le da a esta restricción de tipo CHECK OPTION.
- **WITH READ ONLY:** Hace que la vista sea de sólo lectura. Permite grabar un nombre para esta restricción.

Lo interesante de las vistas es que ***tras su creación se utilizan como si fueran una tabla***.

Ejemplo:

```
CREATE VIEW alumnosPorCurso(curso, numAlumnos)
AS  (
        SELECT c.nombre, count(*)
        FROM alumnos a JOIN cursos c USING (codCurso)
        GROUP BY c.nombre
);
```

La vista pasa a usarse como una **tabla normal**:

```
SELECT * FROM alumnosPorCurso;
```

La consulta anterior sobre la vista, podría mostrar los siguientes resultados:

Curso	NumAlumnos
1º ESO	25
2º ESO	28
3º ESO	24
4º ESO	29
1º BACH	32
2º BACH	33

Se pueden ***crear vistas a partir de otras vistas***:

Por ejemplo: a partir de los datos que devuelve la vista “alumnosPorCurso”, crear la vista “cursosMas25” con la información de los cursos que tengan más de 25 alumnos.

Ejemplo:

```
CREATE VIEW cursosMas25
AS (
        SELECT curso, numAlumnos
        FROM alumnosPorCurso
        WHERE numAlumnos > 25
);
```

```
SELECT * FROM cursosMas25;
```

La consulta anterior sobre la vista, podría mostrar los siguientes resultados:

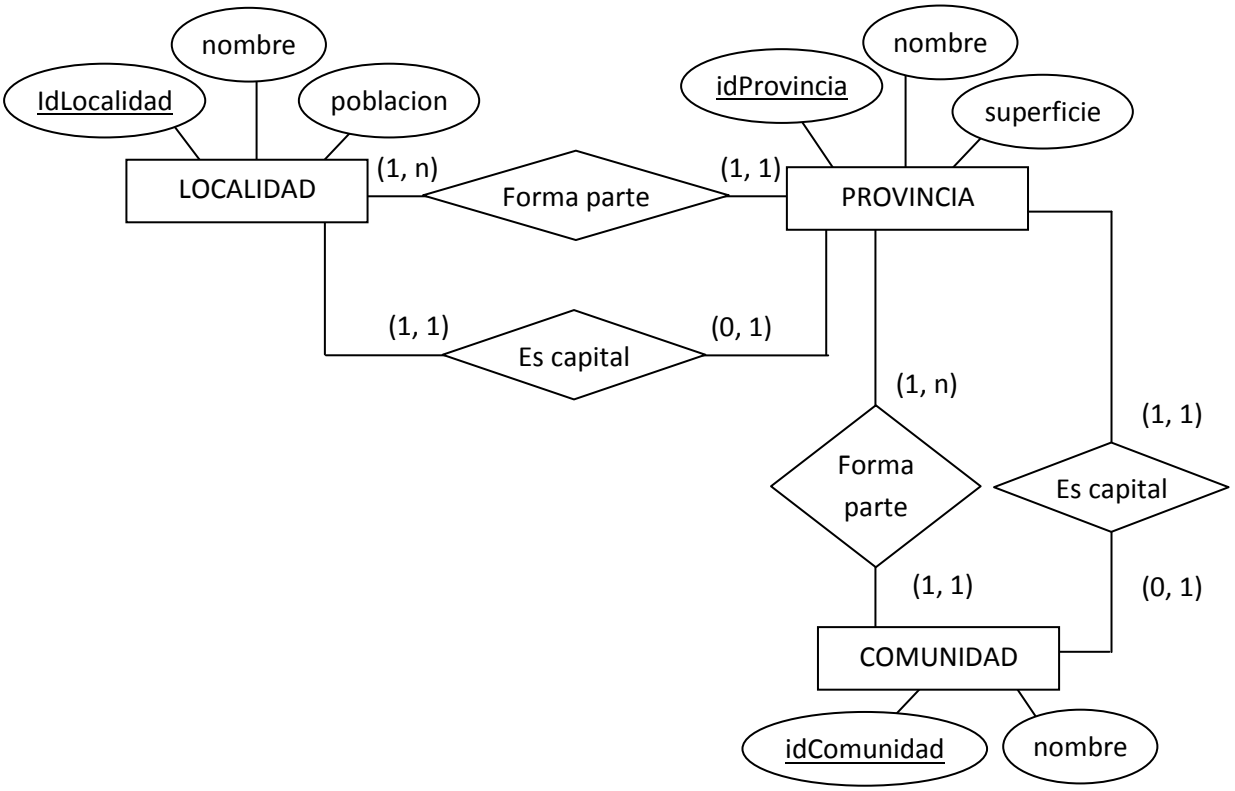
Curso	NumAlumnos
2º ESO	28
4º ESO	29
1º BACH	32
2º BACH	33

Otro ejemplo básico: crear una vista que muestre el salario más alto, la media de los salarios, el salario más bajo y la suma de los salarios de todos los empleados de cada departamento.

```
CREATE VIEW resumenDepartamentos (departamento, salarioMaximo, salarioMedio, salarioMinimo, sumaSalario)
AS (
        SELECT d.nombre, max(salario), avg(salario), min(salario), sum(salario)
        FROM empleados e JOIN departamentos d USING (codDep)
        GROUP BY d.nombre
);
```

Ejemplo complejo:

Teniendo en cuenta el siguiente modelo entidad/relación:



Redúcelo a tablas y da de alta información

Se podría crear la siguiente vista para tener la información relacionada entre comunidades, provincias y localidades. Sabiendo además qué provincia es la capital de la comunidad y qué localidad es la capital de la provincia:

```
CREATE VIEW resumen (comunidad, provincia, localidad,
                                    capitalComunidad, capitalProvincia)
AS (
        SELECT c.nombre,  p.nombre, l.nombre, p2.nombre, l2.nombre
        FROM comunidad c
                JOIN provincia p USING (idComunidad)
                JOIN localidad l USING (idProvincia)
                JOIN provincia p2 ON (c.idProvinciaCapital = p2.idProvincia)
                JOIN localidad l2 ON (p.idLocalidCapital = l2.idLocalidad)
        );
```

La vista pasa a usarse como una "**tabla normal**":

```
SELECT DISTINCT comunidad, capitalComunidad
FROM resumen;
```

La creación de la vista del ejemplo es compleja ya que hay relaciones complicadas, pero una vez creada la vista, se le pueden hacer ***consultas como si se tratara de una tabla normal***.

Incluso se puede utilizar el ***comando DESCRIBE*** sobre la vista para mostrar la estructura de los campos que forman la vista o utilizarse como subconsulta en los comandos UPDATE o DELETE.

Si creamos la vista anterior utilizando ***alias*** en la ***consulta almacenada:***

```
CREATE OR REPLACE VIEW resumen
AS (
        SELECT c.nombre "comunidad",  p.nombre "provincia", l.nombre "localidad",
                p2.nombre "capitalComunidad", l2.nombre "capitalProvincia"
        FROM comunidad c
                JOIN provincia p USING (idComunidad)
                JOIN localidad l USING (idProvincia)
                JOIN provincia p2 ON (c.idProvinciaCapital = p2.idProvincia)
                JOIN localidad l2 ON (p.idLocalidCapital = l2.idLocalidad)
        );
```

A la hora de consultar la información de la vista habría que indicar los campos entre comillas dobles:

```
SELECT DISTINCT "comunidad", "capitalComunidad"
FROM resumen;
```

MODIFICAR INFORMACIÓN A PARTIR DE UNA VISTA

Si se modifican los datos mostrados a partir de una vista, los datos que realmente se modifican son los de la tabla base, ya que la vista lo único que contiene es la consulta almacenada de los datos.

Se puede ***insertar, actualizar o eliminar*** datos de una tabla a través de una vista, teniendo en cuenta las siguientes condiciones:

- La inserción, actualización o eliminación ***no puede afectar a más de una tabla***. Pueden modificarse y eliminarse datos de una vista que combina varias tablas pero la modificación o eliminación solamente debe afectar a una sola tabla.

- ***No*** se pueden cambiar los ***campos resultados de un cálculo***.

- ***Pueden generar errores si afectan a campos a las que la vista no hace referencia***. Ejemplo: si se intenta dar de alta un registro en una vista que consulta una tabla que tiene campos not null que no están incluidos en la vista.

Ejemplos:

Creamos las siguientes tablas en Oracle:

```
create table curso(
        codCurso number(3) constraint cur_pk primary key,
        nombre varchar2(30) constraint cur_nom_nn not null
);

create table alumno(
        codAlumno number(5) constraint alu_cod_pk primary key,
        nombre varchar2(30) constraint alu_nom_nn null,
        priApe varchar2(30) constraint alu_pri_nn not null,
        segApe varchar2(30),
        codCurso number(3) constraint alu_cur_fk references curso(codCurso)
);
```

Podemos crear una vista para tener los datos de los alumnos y el curso en el que se han matriculado:

```
create or replace view alumnosCursos(nombreAlum, priApe, segApe, nombreCurso)
as (
        select a.nombre, priApe, segApe, c.nombre
        from curso c join alumno a  using (codCurso)
);
```

Con esta vista:

```
insert into alumnosCursos(nombreAlum, priApe, segApe) values ('santiago', 'sanchez', 'garcia');

        // Error: no tenemos el código del alumno (campo obligatorio en la tabla
        // Alumno). No podríamos dar de alta.

update alumnosCursos set  priApe = 'martin' where nombreCurso = '2º ESO';

        // Actualización de los datos de la tabla Alumno. Funciona sin problemas, ya
        // que sólo se intenta actualizar los datos de una tabla.

update alumnosCursos set  priApe = 'martin', nombreCurso = '1º BACH'
where nombreCurso = '2º ESO';

        // Error: intentamos actualizar datos de dos tablas (tablas Alumnos y Cursos)

delete from alumnosCursos;

        // Se eliminan los datos de los alumnos de la vista. No se elimina ningún curso.
        // En la tabla alumnos continuarán los datos de los alumnos que no estuviesen
        // matriculados en ningún curso.
```

Si ***reemplazamos la vista anterior*** por esta otra, donde hemos incluido el código del alumno y el del curso:

```
create or replace view alumnosCursos(codAlumno, nombreAlum, priApe, segApe,
codCurso, nombreCurso)
as (
        select a.codAlumno, a.nombre, priApe, segApe, c.codCurso, c.nombre
        from curso c join alumno a  on (c.codCurso = a.codCurso)
);
```

Podríamos dar de alta alumnos sin ningún problema aunque no podríamos dar de alta cursos:

```
insert into alumnosCursos(codAlumno, nombreAlum, priApe, segApe)
                                        values (100, 'santiago', 'sanchez', 'garcia');
        // Podríamos dar de alta un alumno

insert into alumnosCursos(codCurso, nombreCurso)  values (998, '1º CFGS');

        // Error ORA-01779: no se puede modificar una columna que se corresponde
        // con una tabla no reservada por clave.

// Este error significa que no se pueden actualizar los datos de las tablas de la vista que
coincidan con más de una fila de otra tabla.

// Obtendríamos el mismo error si intentáramos realizar la siguiente actualización:

update alumnosCursos  set  nombreCurso = '1º BACH'
                                        where nombreCurso = '1º ESO';
```

Opción "with read only":

```
create or replace view alumnosCursos(codAlumno, nombreAlum, priApe, segApe,
codCurso, nombreCurso)
as (
        select a.codAlumno, a.nombre, priApe, segApe, c.codCurso, c.nombre
        from curso c join alumno a  on (c.codCurso = a.codCurso)
)
with read only;
```

- A partir de la vista anterior sólo se podrían consultar datos, no se permitiría ninguna modificación.

Opción "with check option":

```
create or replace view primerosAlumnos(codAlumno, nombreAlum, priApe)
as (
        select codAlumno, nombre, priApe
        from  alumno
        where     (codAlumno < 200)
);
```

Sobre esta vista se pueden dar de alta alumnos:

```
insert into primerosAlumnos(codAlumno, nombreAlum, priApe)
values (150, 'sara', 'santos');

// No hay  ningún problema en dar de alta el alumno con código 150

insert into primerosAlumnos(codAlumno, nombreAlum, priApe)
values (600, 'manolo, 'gomez');

// Tampoco hay ningún problema a la hora de dar de alta el alumno con código 600

// Si el usuario consulta los datos de la vista, no le va a aparecer el alumno Manolo
// Gomez, aunque sí estará dado de alta en la tabla Alumno.
```

Para resolver este problema se utiliza la opción **"with check option"** en la creación de la vista. Esta opción hace que las operaciones DML que se deseen efectuar sobre dicha vista deban pasar el control de integridad que corresponda a los criterios de definición de la propia vista:

```
create or replace view primerosAlumnos(codAlumno, nombreAlum, priApe)
as (
        select codAlumno, nombre, priApe
        from  alumno
        where     (codAlumno < 200)
)
with check option;
```

Ahora ya no se podrían dar de alta alumnos con un código superior a 200.

3.- Mostrar la lista de vistas

La vista del diccionario de datos de Oracle **USER_VIEWS** permite mostrar una ***lista de todas las vistas que posee el usuario actual***. Es decir, para saber qué vistas hay disponibles se usa:

```
SELECT * FROM USER_VIEWS;
```

La columna **TEXT** de esa vista ***contiene la sentencia SQL que se utilizó para crear la vista*** (sentencia que es ejecutada cada vez que se invoca a la vista).

4.- Borrar vistas

Se utiliza el **comando DROP VIEW**:

```
DROP VIEW nombreDeVista;
```