EL NIVEL EXTERNO:VISTAS

1.- INTRODUCCIÓN.- En el nivel externo de la arquitectura ANSI/SPARC, la base de datos se percibe como una "vista externa", definida por el modelo externo. Los diferentes usuarios pueden tener distintas vistas externas; sin embargo, el término vista se reserva para designar a una tabla derivada; de esta forma para ANSI/SPARC es un conjunto de tablas, alguna de ellas base y otras vistas.

2.- VISTAS.- una vistas, se puede considerar como una tabla virtual, es decir, una tabla que en la realidad no existe por derecho propio, sino que se deriva de una o más tablas subyacentes. En otras palabras, no hay un archivo almacenado que de modo directo represente a la vista por si misma. En cambio en el directorio se almacena la definición de la vista, que muestra como esta se deriva de las tablas subyacentes.

En principio cualquier tabla derivable se puede definir como una vista. El proceso de derivación podría comprender proyectar alguna tabla de base sobre ciertos campos, reunir dos tablas o en realidad ejecutar cualquier sentencia sobre proyecciones, reuniones y operaciones semejantes sobre cualquier conjunto de tablas base. Una vista se crea al ejecutar la proposición del LDD DE SQL CREATE VIEW, cuya sintaxis es la siguiente (como siempre ponemos lo más importante, para más información buscar en la ayuda del gestor):

CREATE VIEW [esquema.]<nombre_vista>

[(nombre_campo [,nombre_campo…])] AS proposición Select

Ejemplo.

```
CREATE VIEW PROVEEDORES_LONDRES AS
SELECT S#,NOMS,ESTADO
FROM S
WHERE CIUDAD='LONDRES'
```

- Una vista se puede definir a partir de otra vista.
- Los nombres de los campos, pueden ser los mismos que los utilizados en las tablas originales, excepto si la vista es la unión de dos tablas, y estas comparten campos con los mismos nombres, los campos proceden de campos calculados en cuyo caso es necesario que sean especificados, pudiendo hacerse, renombrando los campos en la select o bien indicando el nombre de los mismos en el create view.

```
CREATE VIEW PARES_CIUDAD (SCIUDAD,PCIUDAD)
AS SELECT DISTINCT S.CIUDAD,P.CIUDAD
    FROM SPJ,P,S
   WHERE SPJ.S#=S.S# AND SPJ.P#=P.P#
```

O también:

```
CREATE VIEW PARES_CIUDAD
AS SELECT DISTINCT S.CIUDAD SCIUDAD,P.CIUDAD PCIUDAD
    FROM SPJ,P,S
   WHERE SPJ.S#=S.S# AND SPJ.P#=P.P#
```

```
CREATE VIEW CP (P#,SUMACTD) AS
SELECT P#, SUM(CTD)
FROM SPJ
GROUP BY P#
```

La proposición select de una vista, NO PUEDE INCLUIR ORDER BY, no obtener referencias a variables de programa.

Una vez que se ha creado la vista, el usuario, puede utilizarla como si fuera una tabla real, con arreglo a ciertas restricciones que veremos más adelante.

La ejecución correcta de una proposición CREATE VIEW da lugar a que la definición de la vista se almacene en el diccionario del sistema. La proposición select no se ejecuta en ese instante; en cambio lo que ocurre cuando el usuario hace una select (update o delete) a la vista, esa operación y la select de la vista, se combinan para formar una operación modificada. Por ejemplo:

```
SELECT * FROM PROVEEDORES_LONDRES WHERE ESTADO<50 ORDER BY S#
```

Esta combina con la definición de PROVEEDORES_LONDRES, de la siguiente forma:

```
SELECT S#,NOMS,ESTADO FROM S WHERE ESTADO>50 AND CIUDAD='LONDRES' ORDER BY S#
```

Es posible la definición de una vista a partir de otra, por ejemplo:

```
CREATE VIEW NOMBRE_LONDRES AS SELECT NOMS FROM PROVEEDORES_LONDRES
```

Para borrar una vista lo haremos con la sentencia DROP VIEW<nombre_vista>, y para actualizarla lo haremos con la sentencia Alter view. (MIRAR AYUDA).

Si la definición de una vista desaparece del diccionario, en ningún caso se verá afectada la tabla base, lo que si desaparecerán son las vista que se hubieran derivado de ella, así mismo la supresión de la tabla, conlleva la supresión de las vistas que se hubieran definido sobre ella.

Los campos de la vista, heredan los tipos de datos de la tabla base.

El motivo de la creación de vistas es:

- Garantizar la privacidad de los datos a los distintos usuarios, dando y quitando permisos con la sentencia GRANT y REVOKE (que consultaremos en la ayuda del sistema)
- Simplificar el acceso a los datos.
- Servir como puente para la resolución de consultas, que de otro modo no pueden realizarse.

3.- OPERACIONES DEL LMD SOBRE LAS VISTAS

Una vista, es una ventana sobre los datos reales, no una copia separada de esos datos. Los cambios en los datos son visibles a través de la vista y las operaciones contra la vista se convierten en operaciones sobre los datos reales. Esta operación, siempre es posible para una claúsula select, pero no ocurre lo mismo en las operaciones de actualización.

Una vista que va a aceptar actualizaciones debe derivarse en una única tabla base, y debe satisfacer las siguientes restricciones:

- Cada renglón de la vista, debe corresponder de forma diferente e identificable de manera única en la tabla base.
- Cada columna distinta de la vista, debe corresponder a una columna diferente e identificable en la tabla base.

En otras palabras, las únicas diferencias permitidas entre las vistas y las tablas son la en la primara las filas individuales y las columnas individuales se pueden omitir.

Ejemplo 1:

```
CREATE VIEW V1 AS
        SELECT DISTINCT COLOR,CIUDAD
        FROM P
```

V1 es una proyección de la tabla P sin duplicados, si la tabla P, es la que hemos tomado como ejemplo, la vista V1 es la siguiente (con fondo oscuro), y mostrando los valores de p# que corresponden a ese renglón:

V1

COLOR	CIUDAD	
Rojo	Londres	P1,P4,P6
Verde	París	P2
Azul	Roma	P3
Azul	París	P5

La vista V1, no puede soportar la sentencia Insert, pués se omitiría el valor de P# que es una clave primaria.

Actualizaciones y borrados, si permitiría, pero no sería conveniente, pues se verían afectadas varias filas, y no sabríamos que partes están afectadas.

Ejemplo 2.

```
CREATE VIEW V2 AS
SELECT P#, CIUDAD
FROM SPJ INNER JOIN S ON (S.S#=SPJ.S#)
```

V2, comprende la unión de dos tablas, S y SPJ, la extensión correspondiente a la consulta es la siguiente:

P#	CIUDAD	
P1	Londres	S1
P2	Londres	S1
P3	Londres	S1
P4	Londres	S1
P5	Londres	S1
P6	Londres	S1

P1	París	S2
P2	París	S2
P2	París	S3
P2	Londres	S4
P4	Londres	S4
P5	Londres	S4

V2, no puede admitir en absoluto, ninguna operación de actualización. Insertar una nueva pareja de (p#,ciudad) implicaría la inserción de un nuevo renglón en SPJ, para el cual se requiere el valor de S#, por ser clave primaria. La modificación de algún valor en la pareja, podría suponer modificaciones en SPJ, pero nunca estaría claro en cual, pues la pareja P2, `París' puede corresponder a valores en S2 o S3, y lo mismo ocurriría con los borrados. La situación sería peor, si se hubiera utilizado la cláusula distinct.

Ejemplo 3.

CREATE VIEW V3 (P#, SUMACTD) AS
SELECT P#,SUM(CTD) FROM SPJ GROUP BY P#

V3, comprende un group by y una función integrada. Sin considerar ninguna extensión, debe de resultar obvio que no se puede realizar ninguna operación de actualización. Si podría actualizarse P#, pero no sería recomendable, pues no sabemos qué es lo que actualizamos.

Ejemplo 4

CREATE VIEW PARTESROJAS AS
SELECT P#,NOMP,PESO,COLOR
FROM P
WHERE COLOR='ROJO'

PARTESROJAS, si satisface las restricciones y por tanto es actualizable. Sin embargo sirve para ilustrar tres puntos adicionales. Al principio se muestra la siguiente vista, similar al a tabla P:

P#	NOMP	PESO	COLOR
P1	Tuerca	12	Rojo
P4	Tornillo	14	Rojo
P6	Leva	19	Rojo

Una inserción en esta vista ocasionaría un valor NULL en ciudad, que no si está definido como obligatorio, no origina problema ninguno.

Un borrado, sabríamos exactamente qué partes borramos, lo mismo que las actualizaciones, que si lo hiciéramos en el campo color, sencillamente al volver a ejecutar una consulta a la vista, estas partes no saldrían.