Introducción a las Bases de Datos TUPAR

Cursada 2008

Clase 7: Vistas

Facultad de Ciencias Exactas
Universidad Nac. Centro de la Pcia. de Bs. As.

Vistas

- Una relación que está realmente almacenada en la BD → *Tabla base (o básica)*
- Una vista es una tabla virtual → relación que es definida en términos de otras tablas y/o vistas.
- Es un elemento del esquema de la BD (catálogo)
- Puede ser accedida mediante consultas al igual que cualquier otra tabla.
- Habitualmente no materializada → las tuplas de una vista se calculan cuando el usuario hace una consulta sobre ella.

Definición de Vistas

CREATE VIEW *nombre_vista* [n_col1, …, n_coln]
AS *expresión_tabla*
[WITH [CASCADED | LOCAL] CHECK OPTION]

Donde,

- *nombre_vista*: nombre de la vista
- [nom_col1 ,....]: nombres de columna de la vista
- *expresión_tabla:* consulta que define la relación derivada (SELECT)
- WITH CHECK OPTION: impide que se realice una actualización sobre la vista que viole su definición.

Eliminación de Vistas

DROP VIEW *nombre_vista* <opción>

permite eliminar una vista del esquema de la BD.

- **<opción>**

  → Cascade: se elimina cualquier definición de vista o restricción a la que haga referencia

  → Restrict: se rechaza si hay objetos del esquema que la mencionan.

Ventajas de las Vistas

o Simplifican la percepción que los usuarios tienen de la base de datos → presenta la información necesaria y oculta el resto

o Permiten definir consultas frecuentes para no especificarlas cada vez que se utilizan.

o Presentan diferentes datos a diferentes usuarios, → importante cuando usuarios con distintos intereses y destrezas manejan la misma base de datos.

Desventajas de las Vistas

o Actualizaciones VISTA → Tabla BASE restringidas

o Restricciones estructurales → la estructura de una vista se determina en el momento de su creación. Si los componentes cambian, éstos no son considerados.

o El estándar impone restricciones sobre la creación y utilización de vistas, por ejemplo: una vista agrupada no puede ser combinada con otras tablas o vistas para formar una nueva vista.

o Rendimiento: el proceso de resolución de la vista puede exigir el acceso a múltiples tablas cada vez que se accede a ella ⇒ recursos de procesamiento adicionales! → técnicas alternativas de mantenimiento de vistas

Aplicaciones de Vistas: Seguridad

Definiendo diferentes vistas y otorgando privilegios selectivamente sobre ellas, puede restringirse el acceso de los usuarios a ciertos subconjuntos de datos:

- Vistas sobre determinadas columnas, ocultando otras reservadas para usuarios específicos → por ej. Antecedentes penales.
- Vistas sobre determinadas filas, ocultando otras reservadas para usuarios específicos → por ej. Películas no aptas para el público infantil.

Aplicaciones de Vistas: Seguridad

(cont.)

- Vistas sobre determinadas columnas y filas, ocultando otras filas y columnas reservadas para usuarios específicos.
- Se pueden definir vistas sobre conjuntos de tablas (ensambles), seleccionando luego filas y columnas.
- Se puede restringir el acceso a subconjuntos de otra vista o combinaciones de vistas y tablas básicas.

Seguridad: Autorización en SQL usando Vistas

GRANT *access*
ON *view*
TO *user_list*

Ejemplos:

```
GRANT SELECT ON VistaClientes TO Pedro, Ana
GRANT INSERT, UPDATE ON VistaVideos TO Juan
```

Aplicaciones de Vistas: Seguridad

Ejemplo: Definición de un esquema externo para el departamento DCyS sobre la BD de docentes:

```
CREATE VIEW Materias_DCyS AS
SELECT IdMateria, nombre, cuatrim, hs_teor, hs_pract FROM Materia
WHERE dep='DCyS'

CREATE VIEW Profesor_DCyS AS
SELECT Nro_Leg, nombre, TE, categ FROM Profesor
WHERE dep='DCyS'

CREATE VIEW Docencia_DCyS AS
SELECT Nro_leg, IdMateria, hs_teor, hs_pract FROM Docencia
WHERE Nro_leg IN (SELECT Nro_leg FROM Profesor WHERE dep='DCyS')
```

- *Privacidad: Los usuarios de cada Depto sólo tendrían autorización para consultar el esquema externo correspondiente a su departamento.*

Ventajas de las Vistas: Indep. de los Datos

- Pueden ocultar a los usuarios cambios de estructura en las tablas reales, si no son necesarios para sus aplicaciones.

EJEMPLO: La tabla ALUMNOS se divide en dos: una para los datos personales ALUMNO_PERS y otra para los académicos ALUMNO_ACAD.

La original puede ser obtenida mediante una vista que sea el ensamble de las nuevas → puede llamarse ALUMNOS.

Cualquier pieza de código que antes se refería a la tabla ALUMNOS, ahora se refiere a la vista ALUMNOS → y los usuarios no notan la diferencia !

PERO …. La independencia es sólo parcial
→ la tabla ALUMNOS puede ser actualizada … pero la vista ALUMNOS, en general NO !

Creación de Vistas

- Una vista es equivalente a una consulta que ha sido 'guardada'
- El SELECT que la genera puede tener cualquier complejidad
- La vista puede ser usada en cualquier otro SELECT
- Puede ser obtenida a partir de varias tablas y/o vistas

```
CREATE VIEW Pericos AS
  SELECT * FROM Animal
  WHERE (Category = 'loro') AND
  (Color = 'verde') AND (habla = 'si');

SELECT Avg(Precio), max(Precio)
FROM Pericos
WHERE (Tamano LIKE "%mediano%");

CREATE VIEW Mascotas AS
    SELECT P.Nombre, P.Precio,
    V.FechaVenta, V.IdTienda
    FROM Pericos P, Ventas V
    WHERE P.IdAnimal = V.IdAnimal;
```

Consultas sobre Vistas

```
CREATE VIEW Clientes AS
  (SELECT nombre-sucursal, nombre-clte
   FROM Depositante, CtaCte
   WHERE Depositante.NroCuenta = CtaCte.NroCuenta)
    UNION
  (SELECT nombre-sucursal, nombre-clte
  FROM Deudor, Prestamo
  WHERE Deudor.NroDeudor = Prestamo.NroDeudor)
```

Obtener los clientes de la sucursal 'Centro'

SELECT *nomre-clte*
FROM *Clientes* **WHERE** *nombre-sucursal* = 'Centro'

Acceso a una Vista

Puede ser consultada como cualquier tabla base

- Pero hay limitaciones (muchas!) para actualizar vistas → dificultad para propagar las actualizaciones en cascada hacia los componentes.
- Acceso a los datos necesariamente debe limitarse:
  - Usuarios deben identificarse → sistema de *autenticación* (passwords)
  - Cada usuario tiene limitaciones para manipular los datos – *autorización*
- SQL provee herramientas de autorización pero no de autenticación (propietarias de cada sistema)

Actualización de Vistas

Las vistas reciben las actualizaciones provenientes de sus componentes:

Tabla Básica → Vista

Si las vistas son actualizables deben propagar las actualizaciones a las tablas básicas:

Vista→ Tabla Básica

Actualización de Vistas

Actualización datos de la BD → las vistas deben ser actualizadas.

. Recálculo
. Mantenimiento Incremental: computa y aplica sólo los cambios incrementales a las vistas

o Modificación de vistas → problemas inherentes de ambigüedad.

o Suprimir una tupla en una vista → 'borrarla de la tabla básica' ? o 'actualizar alguna columna para que ya no sea seleccionada para la vista'?

Actualización de Vistas

(cont.)

o Insertar una fila en una vista → 'insertar una tupla en la tabla base'? o 'actualizar una tupla existente para que ahora pueda ser seleccionada para la vista'?

o Actualizar una tupla en una vista que involucra join puede cambiar la semántica de otras columnas que no son proyectadas por la vista.

Actualización de Vistas

Las vistas no mantienen COPIAS de los datos → cuando se modifica una vista, se están modificando las tablas base.

Cuándo es posible sin ambigüedades?

o No afecta más de una tabla.
o Si la vista deriva de más de una tabla, pero la actualización afecta sólo a una.

Actualización de Vistas

(cont.)

- Si no afecta una columna que contiene información derivada.
- Si no causa error afectando tablas que no tienen valores por defecto definidos o que no aceptan nulos para alguna columna.
- Se verifica si WITH CHECK OPTION ha sido especificado.

Vistas actualizables: en síntesis

Vista sobre una tabla básica:

• el sistema traducirá la actualización sobre la vista en una operación de actualización sobre la relación básica

→ siempre que no se viole ninguna restricción de integridad definida sobre dicha relación

Vistas actualizables: en síntesis

Vista sobre una concatenación de relaciones:

• la actualización sólo puede modificar UNA de las tablas básicas
• la actualización modificará la relación básica que cumpla la propiedad de *conservación de la clave* (‘Key preserved’, aquella relación tal que su clave primaria podría ser también clave de la vista)
• la actualización no se realizará si viola alguna de las restricciones definidas sobre la relación básica que se va a actualizar

Vistas Actualizables de una tabla básica

- Una vista obtenida a partir de una única tabla básica es actualizable si
  - conserva una clave (primaria o alternativa)
  - no contiene funciones de agregación
  - no incluye DISTINCT
  - No incluye subconsultas en el SELECT
- Vista obtenida a partir de varias tablas básicas es actualizable si
  - la actualización altera la tabla correspondiente a la Key Preserved

Vistas Actualizables: ejemplos

CREATE VIEW Proveedor_ciudad (id_prov, ciudad)**AS**
SELECT id_prov, ciudad
FROM PROVEEDORES;

ACTUALIZABLE

CREATE VIEW Situación_ciudad (situación, ciudad)**AS**
SELECT situación, ciudad
FROM PROVEEDORES;

NO ACTUALIZABLE

CREATE VIEW EnvioProveedor (id_prov, cantTotal) **AS**
SELECT id_prov, SUM (cant)
FROM ENVIO
GROUP BY id_prov;

NO ACTUALIZABLE

Actualización de Vistas

Cuando se inserta o actualiza una tupla en una tabla base, puede ocurrir que esta tupla ya no pertenezca a la vista.

Ejemplo:

CREATE VIEW alumnos_Tandil AS (SELECT * FROM Alumnos WHERE ciudad='Tandil')

Si se hace

UPDATE alumnos_Tandil SET ciudad='Rauch'

Todos los registros de la vista son actualizados con un valor diferente de ciudad y dejan de pertenecer a la vista.

Vistas Actualizables

Esto es legal, pero este efecto puede propagar errores inadvertidos por el usuario.

Solución →

WITH CHECK OPTION en la creación de la vista.

CREATE VIEW alumnos_Tandil AS (SELECT * FROM Alumnos WHERE ciudad='Tandil') WITH CHECK OPTION

Cualquier inserción o actualización que haga 'migrar' una tupla de la vista ≡ la tupla ya no satisface la condición del query que define la vista

→ error en tiempo de ejecución → rechaza la operación

Vistas Actualizables: With Check Option

Cláusula necesaria para vistas actualizables. Todos los INSERT y UPDATE sobre la vista serán verificados para asegurar que los datos satisfacen la definición de la vista → si no es así : Rollback.

LOCAL
Chequea la integridad de la vista corriente.

CASCADE (opción por defecto)
Chequea la integridad en la vista corriente y en toda otra dependiente de ella.

Actualización: ejemplos

```
CREATE VIEW Buenos_proveedores (prov, sit, ciudad )
AS SELECT id_prov, situación, ciudad
   FROM  PROVEEDORES
   WHERE  situación > 15
   WITH CHECK OPTION ;
```

Insert into Buenos_proveedores values (10, 20, 'Paris');

Update Buenos_proveedores set situacion=5 where prov=20; X

Insert into Buenos_proveedores values (8, 5, 'Roma'); X

Modificación de Vistas vía Triggers

- Se puede "interceptar" una modificación a una vista mediante un trigger instead-of → opción especial para vistas (recordar que los triggers se definen sobre tablas !)
- Triggers INSTEAD OF proveen una manera transparente para actualizar vistas que no pueden ser modificadas directamente vía las sentencias SQL DML.

Modificación de Vistas vía Triggers

- El trigger INSTEAD OF se dispara en lugar de ejecutar la sentencia disparadora → el trigger ejecuta UPDATE, INSERT, o DELETE directamente sobre las tablas subyacentes, en forma invisible para el usuario.
- Por defecto, triggers INSTEAD OF se activan 'for each row'.

Modificación de Vistas vía Triggers: Ejemplo

```
CREATE VIEW info_supervisor AS
        SELECT E.nombre,E.Nro_estud,E.Id_superv,P.id_oficina
        FROM   Estudiante E, Profesor P
        WHERE  E.Id_superv = P.legajo;

CREATE TRIGGER insert_info_supervisor
INSTEAD OF INSERT ON info_supervisor
REFERENCING NEW AS n – nuevo supervisor
FOR EACH ROW
BEGIN
IF NOT EXISTS (SELECT * FROM Estudiante E WHERE E.Nro_estud=n.Nro_estud)
THEN INSERT INTO Estudiante(Nro_estud,nombre,supervisor)
                VALUES(n.Nro_estud, n.nombre, n.Id_superv);
ELSE UPDATE Estudiante SET Estudiante.Id_superv = n.Id_superv
                WHERE Estudiante.Nro_estud = n.Nro_estud;
END IF;
IF NOT EXISTS (SELECT * FROM Profesor P WHERE P.legajo= n.Id_superv)
THEN  INSERT INTO Profesor VALUES(n.Id_superv, n.id_oficina);
ELSE UPDATE Profesor SET Profesor.id_oficina = n.id_oficina WHERE
    Profesor.legajo = n.id_superv;
END IF;
END;
```

Pueden plantearse triggers similares para las operaciones de UPDATE y DELETE.

Vistas Materializables

Una vista puede ser materializada → su contenido puede ser precomputado y almacenado por el DBMS

- Características:
  - Cuestiones de performance en la elaboración de la consulta
  - Confiabilidad
  - Mantenimiento de vistas
    - cómo mantener consistencia entre las tablas de la BD y los resultados materializados?
    - Actualización por regeneración o incremental?
    - Implementación de triggers para realizar estas funciones o programas o soporte por el DBMS.
  - Selección → cuáles conviene materializar?
  - Obtener resultados de consultas usando vistas → reescritura de consultas para utilizar los resultados materializados.

Actualización de Vistas: consideraciones

- Tener en cuenta acciones referenciales y la activación de triggers asociados
- Una actualización en una vista debería transformarse en el mismo tipo de operación en las tablas base
- Las reglas de actualización no deben dar por supuesto que las tablas están normalizadas
- Considerar las modificaciones como una sucesión de bajas + altas puede tener consecuencias indeseables.