Seguridad

Bibliografía:

Introducción a los sistemas de bases de datos - C.J. Date

Vistas

Una **tabla** percibida por el usuario puede ser:

–una tabla base

–una vista

- Una **tabla base** es real (existe físicamente).
- Una **vista** es una **tabla virtual** que no tiene existencia en sí misma sino **que se deriva de una o más tablas bases subyacentes**.
  - No existe en sí pero ante el usuario parece existir

Ejemplo

- Crear una vista de S de los buenos proveedores, cuya situación sea mayor que 15.

```
CREATE VIEW BUENOS_PROVEEDORES
  AS  SELECT  S#,  SITUACION,  CIUDAD
      FROM S
      WHERE SITUACION > 15;
```

S#	SNOMBRE	SITUACION	CIUDAD
S1	Salazar	20	Londres
S2	Jaimes	10	París
S3	Bernal	30	París
S4	Corona	20	Londres
S5	Aldana	30	Atenas

Para el **usuario** es como si **existiera** una tabla llamada BUENOS_PROVEEDORES

con la siguientes filas y columnas

S#	SITUACION	CIUDAD
S1	20	Londres
S3	30	París
S4	20	Londres
S5	30	Atenas

Vistas

- Es una **ventana** a través de la cual se ve la **tabla real S**
- Es una **ventana dinámica**:
  - las modificaciones hechas en S se verán automáticamente
- Los usuarios pueden trabajar en las vistas como si fueran una tabla real

Ejemplo de una **selección**

```
SELECT *
FROM BUENOS_PROVEEDORES
    WHERE CIUDAD <> ‘Londres’;
```

- Resultado:

S#	SITUACION	CIUDAD
S3	30	París
S5	30	Atenas

El sistema **convierte** esta **operación en una equivalente** realizada sobre la **tabla** (ó tablas) **subyacente**:

```
SELECT * FROM   S
WHERE CIUDAD <> 'Londres'
AND  SITUACION  > 15;
```

La conversión se hace:

- ***combinando*** el **SELECT DEL USUARIO**
- con la proposición **SELECT guardada en el catálogo** cuando se definió la vista

Ejemplo de **actualización**

```
UPDATE  BUENOS_PROVEEDORES
SET   SITUACION = SITUACION + 10
WHERE CIUDAD =  ‘París’;
```

- Será convertida en:

```
UPDATE   S
  SET   SITUACION = SITUACION + 10
  WHERE CIUDAD =  ‘París’
  AND  SITUACION  > 15;
```

- La **inserción** y **eliminación** se manejan en forma análoga

Definición de Vistas

Sintaxis:

```
CREATE  VIEW  nombre-vista [(columna
 [,columna] ....)]
      AS  subconsulta
      [WITH CHECK OPTION];
```

Definición de Vistas

Ejemplo:

- Crear una vista de los envíos, sumando la cantidad de cada parte.

```
CREATE VIEW PC ( P#,
 CANTTOT)
 AS SELECT  P#, SUM(CANT)
      FROM  SP
      GROUP BY  P#;
```

Observaciones

- Una **subconsulta** puede extraer **datos de otra vista**

```
CREATE VIEW
  PARTES_ROJAS_LONDRES
  AS SELECT  P#, PESO
    FROM  PARTESROJAS
    WHERE  CD = ‘Londres’;
```

Observaciones

- **Con opción de verificación:**
  - las operaciones UPDATE e INSERT se verificarán para **garantizar** que toda fila modificada o insertada **satisfaga** la condición de la vista.

Ejemplo

```
CREATE VIEW
  BUENOS_PROVEEDORES
  AS  SELECT  S#,  SITUACION,
  CIUDAD
    FROM S
    WHERE SITUACION > 15
    WITH CHECK OPTION;
```

- La BD se modificará sólo si se cumple el WHERE de la

Borrado de Vistas

- Para **borrar** una vista:

DROP VIEW nombre-vista;

Operaciones de DML sobre vistas

- No todas las vistas se pueden actualizar
  - –inserción
  - –eliminación
  - –modificación

Sean las siguientes dos vistas sobre una misma tabla:

1. CREATE VIEW S#_CIUDAD

AS SELECT S#, CIUDAD FROM S;

2. CREATE VIEW SITUACION_CIUDAD

AS SELECT SITUACION, CIUDAD FROM S;

Sean las siguientes dos vistas sobre una misma tabla:

1. CREATE VIEW S#_CIUDAD
AS SELECT S#, CIUDAD FROM S;

- Se puede **insertar** un nuevo registro
  - (S6, Roma), mediante la inserción real del registro **(S6,NULL,NULL,Roma)** en la tabla subyacente

Sean las siguientes dos vistas sobre una misma tabla:

1. CREATE VIEW S#_CIUDAD

AS SELECT S#, CIUDAD FROM S;

- Se puede **eliminar** un registro existente.
  - Por ejemplo: (S1,Londres) eliminando en realidad **(S1,Salazar,20,Londres)**

Sean las siguientes dos vistas sobre una misma tabla:

1. CREATE VIEW S#_CIUDAD
AS SELECT S#, CIUDAD FROM S;

- Se puede **modificar** un campo.
  - Por ejemplo: Londres por Roma para S# = 'S1'.

Sean las siguientes dos vistas sobre una misma tabla:

2. CREATE VIEW SITUACION_CIUDAD AS SELECT SITUACION, CIUDAD FROM S;

- En “teoría” 2 **no se puede actualizar**.

Sean las siguientes dos vistas sobre una misma tabla:

2. CREATE VIEW SITUACION_CIUDAD
AS SELECT SITUACION, CIUDAD FROM S;

- Si tratamos de **insertar** un nuevo registro (40, Roma),
  - el sistema tendrá que insertar el registro **(NULL,NULL,40,Roma)** en la tabla subyacente
  - Esa operación **fracasará** pues los **S#** están **definidos NOT NULL**

Sean las siguientes dos vistas sobre una misma tabla:

2. CREATE VIEW SITUACION_CIUDAD
AS SELECT SITUACION, CIUDAD FROM S;

- Si tratamos de **eliminar** un registro existente.
  - Por ej. (20,Londres), el sistema no sabe cual, porque no se ha especificado el nro. de proveedor, porque no es parte de la vista

Sean las siguientes dos vistas sobre una misma tabla:

2. CREATE VIEW SITUACION_CIUDAD
AS SELECT SITUACION, CIUDAD FROM S;

- Si se quiere **modificar** un campo
  - por ej. (20,Londres) a (20, Roma), el sistema no sabe que registro modificar de la tabla subyacente.

Diferencia entre Vista 1 y Vista 2

- **Vista 1** incluye la **clave primaria** de la tabla subyacente

- **Vista 2** no la incluye.

Una vista se puede **actualizar**
si **conserva la clave primaria**

Observaciones

Si se hace una **actualización** como la siguiente

```
UPDATE BUENOS_PROVEEDORES
SET SITUACION = 5
WHERE  S# = ‘S1’;
```

- Si se acepta, **S1 no aparecerá más en la vista** (equivale a eliminarlo)

Observaciones

Si se hace una **inserción**

```
INSERT INTO BUENOS_PROVEEDORES (S#,
   SITUACION, CIUDAD)
   VALUES ('S8',  5,  'Estocolmo');
```

- Si se acepta, **S8 se creará pero no aparecerá más en la vista**

- La **opción CHECK OPTION** fue diseñada para **resolver estos problemas**.
  - Si se especifica, chequeará y no insertará en la vista
  - Si no se especifica, se aceptarán las actualizaciones pero no aparecerán en la vista.

Ventajas de las vistas

- Permiten a los usuarios ver los **mismos datos de distintas maneras al mismo tiempo**
- Se **simplifica la percepción del usuario**: muestra solo los datos que le interesan y no otros.
- Se cuenta con **seguridad** automática para datos ocultos (información no visible a través de una vista)

Seguridad e Integridad

- **Seguridad** se refiere a la **protección** de los datos contra una visualización, alteración o destrucción no autorizada.

- **Integridad** se refiere a la exactitud o **validez** de los datos.

Seguridad e Integridad

- **Seguridad** implica asegurar que los usuarios están ***autorizados*** para llevar a cabo lo que tratan de hacer.

- **Integridad** implica asegurar que lo que tratan de hacer es ***correcto.***

Seguridad: Consideraciones Generales

- El **DBA** se encargará de **especificar restricciones:**

  ¿quién puede tener acceso a qué?

- Un **usuario** dado tendrá **diferentes derechos de acceso** o autorizaciones **sobre diferentes objetos de información.**

- **Diferentes usuarios** pueden tener **diferentes derechos sobre el mismo objeto.**

El **SQL** cuenta con dos **características** independientes

para mantener la **seguridad**:

- El mecanismo de **vistas**.
- El **subsistema de autorización**, mediante el cual usuarios pueden:
  - **conceder** de manera selectiva y dinámica esos derechos a otros usuarios,
  - y después **revocar** esos derechos si lo desean.

- Las proposiciones de SQL del **subsistema de autorización** son

  GRANT (otorgar)

  REVOKE (revocar)

- El sistema las graba en el **catálogo** en forma de *restricciones de autorización*

- El **DBA** tiene **todos los derechos y puede concederlos** a otros

- Si el administrador concede a otro usuario U el derecho de crear algún objeto (vista o tabla),

  – entonces **U automáticamente tiene todo tipo de derecho sobre ese objeto**,

  – incluyendo el **derecho de conceder derechos**.

Sintaxis

GRANT comandos
ON TABLE tablas
TO usuarios
WITH GRANT OPTION;

Ejemplos:

```
GRANT SELECT ON TABLE S TO CARLOS;

GRANT SELECT, UPDATE (SITUACION,
   CIUDAD)
   ON TABLE S TO JULIA;

GRANT ALL ON TABLE S, P  TO MARIA, LUIS;

GRANT SELECT ON TABLE P TO PUBLIC;
```

Si U1 concede cierta autorización a U2,
el usuario U1 puede **revocarla** después

```
REVOKE comandos
ON TABLE tablas
FROM usuarios;
```

Ejemplos:

```
REVOKE  SELECT   ON TABLE  S  FROM
   CARLOS;

REVOKE  UPDATE    ON TABLE  S   FROM
   JULIA;

REVOKE  INSERT, DELETE   ON TABLE  SP
   FROM PEPE;

REVOKE  ALL    ON TABLE  S, P, SP  FROM
   MARIA;
```

- Derechos aplicables a **tablas** y **vistas** son:

SELECT, UPDATE, DELETE, INSERT

- Derechos aplicables solo a **tablas** son:

ALTER TABLE e INDEX.

- Si el usuario U1 tiene el **derecho de conceder cierta autorización A1** a otro usuario U2,
- tendrá también el **derecho de concederla** al usuario U2 "**con la opción GRANT**"

- **Usuario U1:**

  GRANT SELECT ON TABLE S TO U2 WITH GRANT OPTION;

- **Usuario U2:**

  GRANT SELECT ON TABLE S TO U3 WITH GRANT OPTION;

- **Usuario U3:**

  GRANT SELECT ON TABLE S TO U4 WITH GRANT OPTION;

- Si el usuario U1 emite ahora:

  REVOKE SELECT ON TABLE S FROM U2;

- La revocación se propagará y se revocarán todas automáticamente.