Vistas.

Apuntes BD

Índice

Vistas

1.1 ¿Qué es una vista?

Una vista es una especie de tabla "virtual" que se crea a partir de una consulta SQL y se almacena en la base de datos. Los datos que se muestran en la vista son los resultados de la consulta que la define y están almacenados en las tablas reales de la base de datos.

Algunas de las principales ventajas que nos aporta el uso de vistas son las siguientes:

- Nos permiten convertir una consulta compleja en una tabla "virtual" para que sea más fácil trabajar con ella.
- Nos permiten ocultar columnas de una tabla real que no queremos que estén visibles a ciertos usuarios.

1.2 Crear o modificar una vista

```
CREATE
    [OR REPLACE]
    [ALGORITHM = {UNDEFINED | MERGE | TEMPTABLE}]
    [DEFINER = user]
    [SQL SECURITY { DEFINER | INVOKER }]
    VIEW view_name [(column_list)]
    AS select_statement
    [WITH [CASCADED | LOCAL] CHECK OPTION]
```

Ejemplo:

Crea una vista que muestre para cada uno de los pedidos, el código del pedido, la fecha, el nombre del cliente que realizó el pedido y el importe total del pedido.

```
CREATE OR REPLACE VIEW resumen_pedidos AS
SELECT
  pedido.codigo_pedido,
    pedido.fecha_pedido,
    cliente.nombre_cliente,
    SUM(detalle_pedido.cantidad * detalle_pedido.precio_unidad) AS total
FROM
  cliente INNER JOIN pedido
    ON cliente.codigo_cliente = pedido.codigo_cliente
    INNER JOIN detalle_pedido
    ON pedido.codigo_pedido = detalle_pedido.codigo_pedido
GROUP BY pedido.codigo_pedido
```

Cuando creamos una vista, es posible crear un alias para cada una las columnas. En el siguiente ejemplo se muestran los nombres que tendrán las columnas encerrados entre paréntesis.

```
CREATE OR REPLACE VIEW resumen_pedidos (codigo_pedido, fecha_pedido,
    nombre_cliente, total) AS
SELECT
  pedido.codigo_pedido,
    pedido.fecha_pedido,
    cliente.nombre_cliente,
    SUM(detalle_pedido.cantidad * detalle_pedido.precio_unidad)
FROM
  cliente INNER JOIN pedido
    ON cliente.codigo_cliente = pedido.codigo_cliente
    INNER JOIN detalle_pedido
    ON pedido.codigo_pedido = detalle_pedido.codigo_pedido
GROUP BY pedido.codigo_pedido
```

Referencias:

- Documentación oficial de `CREATE VIEW` en MySQL.

1.3 Eliminar una vista

```
DROP VIEW [IF EXISTS]
    view_name [, view_name] ...
    [RESTRICT | CASCADE]
```

Ejemplo:

```
DROP VIEW resumen_pedidos;
```

Referencias:

- Documentación oficial de `DROP VIEW` en MySQL.

1.4 Renombrar una vista

```
RENAME TABLE
    tbl_name TO new_tbl_name
    [, tbl_name2 TO new_tbl_name2] ...
```

Ejemplo:

```
RENAME TABLE old_table TO new_table;
```

Referencias:

- Documentación oficial de `RENAME TABLE` en MySQL.

1.5 Consultar el listado de vistas disponibles

```
SHOW FULL TABLES;
```

```
SHOW FULL TABLES
WHERE table_type = 'VIEW';
```

Referencias:

- Documentación oficial de `SHOW FULL TABLES` en MySQL.

1.6 Consultar la sentencia que se utilizó para crear una vista

```
SHOW CREATE VIEW view_name
```

Referencias:

- Documentación oficial de `SHOW CREATE VIEW` en MySQL.

1.7 Vistas con contenido actualizable con `INSERT`, `UPDATE` y `DELETE`

En algunos casos es posible actualizar el contenido de las tablas que se utilizan para crear una vista.

Para que una vista sea actualizable, es necesario que todas las columnas obligatorias de la tabla subyacente estén presentes en la vista.

Referencias:

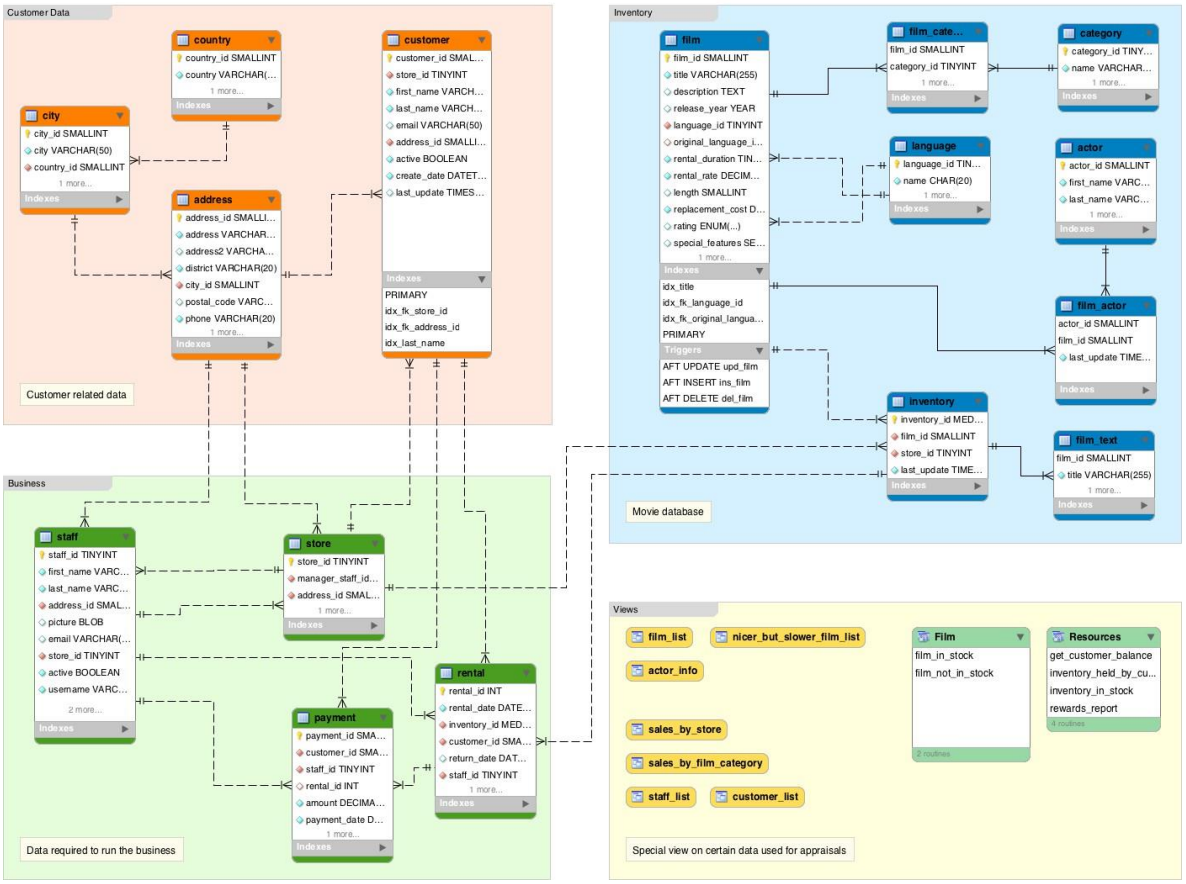
- Documentación oficial sobre vistas actualizables.

1.8 Ejemplos

1.8.1 Base de datos: Sakila

La base de datos `Sakila` está disponible en la página web oficial de MySQL. Se trata de una base de datos creada por **Mike Hillyers**.

1.8.2 Modelo entidad/relación



1.8.3 Base de datos para MySQL

La base de datos está disponible en la web oficial de MySQL.

1.8.4 Vistas utilizadas en la base de datos `Sakila`

A continuación se muestran algunas de las vistas que se han utilizado en la base de datos `Sakila`.

```
1  --
2  -- View structure for view `customer_list`
3  --
4
5  CREATE VIEW customer_list AS
```

```
SELECT
  cu.customer_id AS ID,
    CONCAT(cu.first_name, _utf8mb4' ', cu.last_name) AS name,
    a.address AS address,
    a.postal_code AS `zip code`,
  a.phone AS phone,
    city.city AS city,
    country.country AS country,
    IF(cu.active, _utf8mb4'active',_utf8mb4'') AS notes,
    cu.store_id AS SID
FROM
  customer AS cu JOIN address AS a
    ON cu.address_id = a.address_id
    JOIN city
    ON a.city_id = city.city_id
  JOIN country
    ON city.country_id = country.country_id;
```

```
--
-- View structure for view `film_list`
--

CREATE VIEW film_list AS
SELECT
  film.film_id AS FID,
    film.title AS title,
    film.description AS description,
    category.name AS category,
    film.rental_rate AS price,
  film.length AS length,
    film.rating AS rating,
    GROUP_CONCAT(CONCAT(actor.first_name, _utf8mb4' ', actor.last_name)
        SEPARATOR ', ') AS actors
FROM
  category LEFT JOIN film_category
    ON category.category_id = film_category.category_id
    LEFT JOIN film
    ON film_category.film_id = film.film_id
  JOIN film_actor
    ON film.film_id = film_actor.film_id
  JOIN actor
    ON film_actor.actor_id = actor.actor_id
GROUP BY film.film_id, category.name;
```