SQL

- Vistas
- Disparadores
- Procedimientos Almacenados

VISTAS

Utilizadas para aumentar las posibilidades de acceso a tablas

CREATE VIEW *nombre_de_vista* **[<columnas]**
AS *<expresion de consulta>*

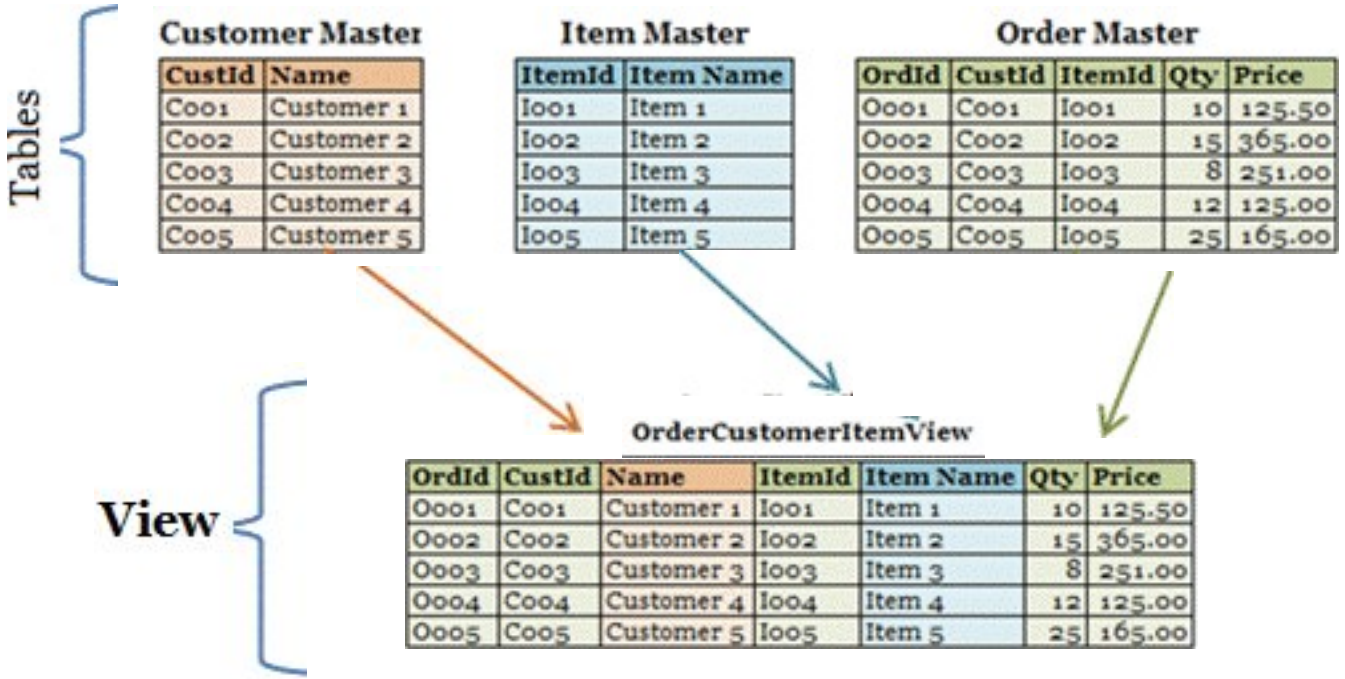
```
CREATE VIEW Rojos
AS SELECT B.bid, COUNT (*) AS scount
    FROM Botes B, Reservacion R
    WHERE  R.bid=B.bid AND
     B.color='rojo'  GROUP BY B.bid;
```

VISTAS

Utilizadas para aumentar las posibilidades de acceso a tablas

CREATE VIEW *nombre_de_vista* **[columnas]**
AS *<expresion de consulta>*

- Parecidas a las Tablas, la diferencia es que las vistas son creadas a partir de las tablas.
- Hace la programación simple.
- Usadas normalmente para crear reportes.
- Seguridad al compartir información.



```sql
drop view if exists  promedios_clientes;
create  view promedios_clientes as
select c.first_name as nombres, c.last_name as apellidos,
    avg(p.amount) as promedios from
customer as c
inner join payment as p on c.customer_id = p.customer_id
group by c.customer_id;

select * from promedios_clientes
order by promedios desc;
```

TRIGGERS (Disparadores)

- Procedimiento que se ejecuta cuando se cumple una condición establecida al realizar una operación de inserción (INSERT), actualización (UPDATE) o borrado (DELETE).
- Usados para mejorar la administración de la Base de datos, sin necesidad de contar con que el usuario ejecute la sentencia de SQL.
- Además, pueden generar valores de columnas, previene errores de datos, sincroniza tablas, modifica valores de una vista, etc.

Estructura básica -TRIGGERS

Llamada de activación: es la sentencia que permite "disparar" el código a ejecutar.

Restricción: es la condición necesaria para realizar el código. Esta restricción puede ser de tipo condicional o de tipo nulidad.

Acción a ejecutar: es la secuencia de instrucciones a ejecutar una vez que se han cumplido las condiciones iniciales.

SINTAXIS - TRIGGER

```
DELIMITER |
CREATE TRIGGER <trigger_name> <trigger_time> <trigger_event>
ON  <table_name>  FOR EACH
ROW  BEGIN
…
END
DELIMITER;
```

trigger_time: BEFORE or AFTER

trigger_event: INSERT, UPDATE OR DELETE

EJEMPLO

```
DELIMITER |

CREATE TRIGGER testref BEFORE INSERT ON test1
  FOR EACH ROW BEGIN
    INSERT INTO test2 SET a2 = NEW.a1;
    DELETE FROM test3 WHERE a3 = NEW.a1;
    UPDATE test4 SET b4 = b4 + 1 WHERE a4 = NEW.a1;
  END
|

DELIMITER ;
```

EJEMPLO

```
DELIMITER $$
CREATE TRIGGER update_limitecredito

   BEFORE INSERT ON PEDIDOS


   FOR EACH ROW BEGIN

   update CLIENTES
   SET LIMITECREDITO =

       LIMITECREDITO--new.IMPORTE  where

       CLIENTES.NUMCLIE = new.CLIE;

END$$

DELIMITER ;
```

EJEMPLO

```
DELIMITER |
CREATE TRIGGER tr1 BEFORE INSERT ON PAYMENTS
        FOR EACH ROW
        BEGIN
                IF new.amount < 100 THEN
                        SET new.amount = new.amount * 0.90;
                ELSEIF NEW.amount >= 100 THEN
                        SET NEW.amount = 0;
                END IF;
        END |
DELIMITER ;
```

Escriba un trigger que realice un descuento del 10% en los pagos si el monto es menor a $100 y, un descuento del 100% si es mayor o igual $100

EJEMPLO

```
DELIMITER ||
CREATE TRIGGER tr4 AFTER UPDATE ON PAYMENTS
    FOR EACH ROW
    BEGIN
        IF new.amount<0 THEN
            UPDATE customers SET creditLimit = new.amount
            WHERE customerNumber = new.customerNumber;
        ELSEIF new.amount > 100 THEN
            UPDATE customers SET creditLimit = old.amount+1
            WHERE customerNumber = new.customerNumber;
        END IF;
    END ||
DELIMITER ;
```

- Escriba un trigger que se ejecute después de que se actualicen datos en la tabla payments.
- El trigger debe actualizar el límite de crédito de acuerdo a la siguiente condición:
- Si el monto es menor a 0, el límite de crédito debe tomar el valor del nuevo monto.
- Si el monto es mayor a 100, el límite de crédito debe tomar el valor del monto actual + 1.

```
DROP TRIGGER IF EXISTS actualizar_promedio
DELIMITER $$
create trigger  actualizar_promedio after update on payment
for each row
begin
    update totalcliente set total =
    (select sum(amount) from payment where customer_id = new.customer_id)
    where customer_id = new.customer_id;
end
$$  DELIMITER ;


update payment set amount = 300 where payment_id = 1;
```

```
DROP TRIGGER IF EXISTS insertar_promedio;
DELIMITER $$
create trigger  insertar_promedio after insert on payment
for each row
begin
    declare pagos_existentes int;
    set @pagos_existentes = (select count(*) from payment where customer_id = new.customer_id);

    create temporary table if not exists tmp_log as select @pagos_existentes;
    insert into tmp_log select @pagos_existentes;

    if @pagos_existentes = 1 then
        insert into totalcliente values (new.customer_id, new.amount);
    else
        update totalcliente set total =
        (select sum(amount) from payment where customer_id = new.customer_id)
        where customer_id = new.customer_id;
    end if;
end;
$$ DELIMITER ;

insert into payment(customer_id, staff_id, rental_id, amount, payment_date, last_update) values(1,1,76,1.4, now(), now());
```

Stored Procedure

- Es un programa (o procedimiento) el cual es almacenado físicamente en una base de datos.
- La ventaja es que al ser ejecutado, en respuesta a una petición de usuario, es ejecutado directamente en el motor de bases de datos, el cual usualmente corre en un servidor separado.
- Como tal, posee acceso directo a los datos que necesita manipular y sólo necesita enviar sus resultados de regreso al usuario, deshaciéndose de la sobrecarga resultante de comunicar grandes cantidades de datos salientes y entrantes.

```
drop procedure if exists calcular_iva;

delimiter $$
create procedure calcular_iva(in iva decimal(3,2), in valor decimal(5,2), out resultado decimal)
begin
    select valor*iva into resultado;
end;
$$ delimiter ;


call calcular_iva(0.14, 100.34, @r);
select @r;
```