**STORED PROCEDURES**

Es un conjunto de sentencias SQL que se almacenan en el servidor.

Ejemplo: CREATE PROCEDURE ObtenerEnvios()

BEGIN

SELECT \* FROM envios;

END

* Se lo crea indicando: nombre, parámetros (IN, OUT, INOUT) de ser necesario, cuerpo del procedimiento.
* Se lo invoca mediante: CALL nombres
* Eliminación: al borrar toda la base los SP se eliminan.
  + DROP PROCEDURE IF EXISTS nombres

Tiene acceso homogéneo, asegura la consistencia en las operaciones, pueden ayudar a la performance y sirve para consultas complejas.

* Privilegios que tiene que tener un usuario para:
  + Crearlo: CREATE ROUTINE
  + Modificar solo metadata (no parámetros, ni cuerpo): ALTER ROUTINE
  + Ejecutarlo: EXECUTE







* Parámetro IN: son parámetros de entrada. Se puede usar y modificar su valor dentro del SP, pero los cambios no se verán reflejados fuera de este.
* Parámetro OUT: son parámetros de salida. Se puede asignar un valor dentro del SP, y usarlo dentro del mismo. Los cambios se verán reflejados fuera del SP.
* Parámetro INOUT: son parámetros de entrada salida. Se puede usar y modificar su valor dentro del SP, y los cambios se verán reflejados fuera de este.
* Función LAST\_INSERT\_ID: retorna el valor del último auto incremental agregado inmediatamente anterior a su invocación. Si la inserción es errónea, el valor que retorna es indefinido.
* Estructura de control:



* Cursores: Permiten guardar en ellos valores obtenidos de ejecutar una sentencia SQL. Es posible recorrerlos e ir recuperando de a uno sus valores.
  + Operaciones:
    - DECLARE: permite declarar un cursor.
    - OPEN: permite abrir un cursor que haya sido declarado.
    - FETCH: permite recuperar el valor de un cursor que ya ha sido abierto previamente.
    - CLOSE: permite cerrar un cursor que ha sido al menos, declarado previamente.

**TRANSACCION**

Es una unidad de ejecución de un programa que accede y posiblemente modifica datos. Colección de operaciones que forman una única unidad lógica de trabajo. Tiene un inicio y un fin definido.

Un DBMS debe asegurar que la ejecución de las transacciones se realice adecuadamente a pesar de la existencia de fallos. O se ejecuta la transacción completa o no se ejecuta nada.

* Propiedades ACID:
  + Atomicidad: todas las operaciones de la transacción se ejecutan o no lo hacen ninguna de ellas.
  + Consistencia: la ejecución aislada de la transacción conserva la consistencia de la BD.
  + Aislamiento (isolation): cada transacción ignora el resto de las transacciones que se ejecutan concurrentemente en el sistema, actúa como si fuera única.
  + Durabilidad: una transacción terminada con éxito realiza cambios permanentes en la BD, incluso si hay fallos en el sistema.
* Estados de una transacción:
  + Activa: Es el estado inicial, la transacción permanece en este estado mientras se ejecuta.
  + Parcialmente comprometida: Después de ejecutarse la última operación.
  + Fallida: Luego de darse cuenta de que no puede continuar con la ejecución normal.
  + Abortada: Después de haber retrocedido la transacción y restablecido la base de datos a su estado anterior al comienzo de la transacción.
  + Comprometida: Al completarse con éxito



* Rollback: Revierte una transacción explícita o implícita hasta el inicio de la transacción o hasta un punto de retorno dentro de la transacción. Puede usar ROLLBACK TRANSACTION para borrar todas las modificaciones de datos realizadas desde el inicio de la transacción o hasta un punto de retorno. También libera los recursos que mantiene la transacción.

**TRIGGER**

Son disparadores que se pueden usar BEFORE (antes) o AFTER (después) de las operaciones de INSERT, DELETE y UPDATE.



* Donde:
  + nombre\_disp es el nombre que se le asigna
  + momento\_disp: BEFORE o AFTER
  + evento\_disparador: INSERT-DELETE-UPDATE
  + nombre\_tabla: tabla sobre la cual se generará el evento
  + sentencia\_disp: conjunto de sentencias del cuerpo del trigger
* Cuando el evento disparador es UPDATE, se puede usar:
  + OLD.nombreColumna (hace referencia al valor antes de actualizarse de la columna en cuestión)
  + NEW.nombreColumna (hace referencia al valor después de actualizarse de la columna en cuestión)
* Cuando el evento disparador es INSERT, se puede usar:
  + NEW.nombreColumna
* Cuando el evento disparador es DELETE, se puede usar:
  + OLD.nombreColumna
* Eliminar un trigger:
  + DROP TRIGGER nombreTrigger
  + Al eliminar una tabla, todos los triggers asociados son eliminados
* Privilegios para crear un trigger:



* Una columna OLD es de solo lectura y requiere privilegios de SELECT.
* Una columna de NEW requiere privilegio de SELECT, pero además, es posible modificarla si se usa con BEFORE, para ello es necesario además privilegio de UPDATE.
* Limitaciones:
  + No pueden ejecutar un stored procedure (call).
  + No pueden usar sentencias que explícita o implícitamente abran o cierren una transacción.

**VISTAS**

* Una vez creada, la definición de una vista es “congelada”. Esto significa que cambios posteriores a las tablas de la vista no afectaran la vista.
* Las vistas pertenecen a una base de datos, por lo que, si se elimina base, se elimina la vista.
* Los nombres de las columnas deben ser únicos.
* Una vez creada la vista, los datos se actualizan automáticamente cuando cambias las tablas referenciadas.
* No se pueden utilizar tablas temporales ni crear vistas temporales.
* No se pueden asociar triggers con las vistas.

Ejemplo:



**PERMISOS**

* ALL PRIVILEGES: como mencionamos previamente esto permite a un usuario de MySQL acceder a todas las bases de datos asignadas en el sistema.
* CREATE: permite crear nuevas tablas o bases de datos.
* DROP: permite eliminar tablas o bases de datos.
* DELETE: permite eliminar registros de tablas.
* INSERT: permite insertar registros en tablas.
* SELECT: permite leer registros en las tablas.
* UPDATE: permite actualizar registros seleccionados en tablas.
* GRANT OPTION: permite remover privilegios de usuarios.

Crear usuario: CREATE USER nombre@host IDENTIFIED BY password;

Darle permisos: GRANT [permiso] ON [nombre de bases de datos].[nombre de tabla] TO ‘[nombre de usuario]’@'localhost’;

Crear una tabla: CREATE TABLE nombre\_tabla (columna1 – dato, columna2 – dato, columna3 – dato … )

Editar una tabla: ALTER TABLE nombre\_tabla ADD, DROP o lo que quereamos editarle.

Eliminar una tabla: DROP TABLE nombre\_tabla