UNIVERSIDAD COOPERATIVA DE COLOMBIA

Gestión de la Información

Gustavo Adolfo Gómez Gómez
MSc. Gestión, aplicación y desarrollo de software
2024

www.ucc.edu.co

Gestión de la Información

Unidad 3: Intervenir diseños, plataformas y soportes informáticos a los modelos de
gestión de información de la organización.

Tema: Gestionar el entorno de la base de datos.

Gestionar el entorno de la base de datos.

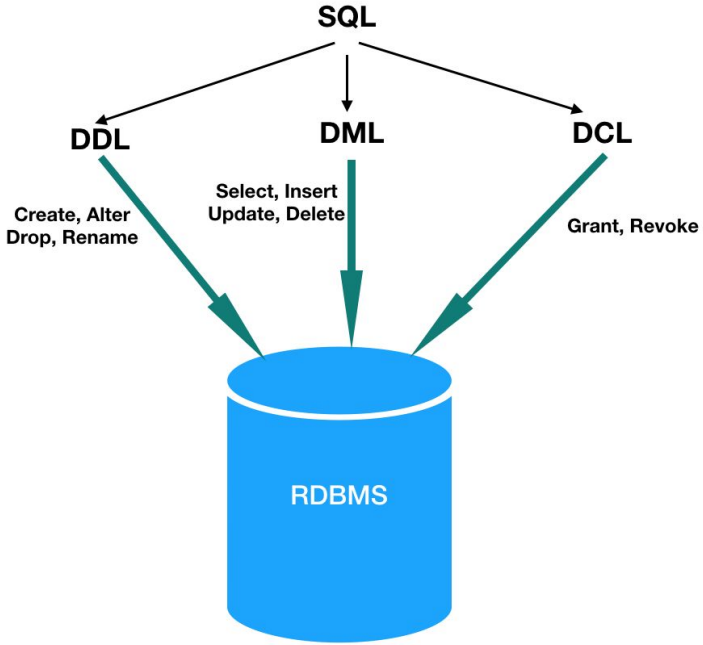
DCL. Usuarios y privilegios

Esquema de Permisos

DCL: Data Control Language

Lenguaje de la seguridad y el acceso a los datos

- Autenticación:
  - Quién: qué usuario se autentica
  - Dónde: a qué base de datos se conecta
  - Qué: cuáles acciones puede realizar



Esquema de usuarios

Usuarios

- Administrador por defecto: root

Nota: igual que linux

Esquema de usuarios

Crear un usuario

- Nombre de usuario
- Origen de la conexión: servidor, dirección IP, subred, etc
- Contraseña

```
CREATE USER 'miusuario'@localhost IDENTIFIED BY 'mipassword';
```

Esquema de usuarios

Seleccionar usuarios y roles

```
use mysql;
select User, Host from user;
```

User	Host
mariadb.sys	localhost
root	localhost
root	desktop-er6pdf2
root	127.0.0.1
root	::1
app1	localhost

Esquema de usuarios

Eliminar un usuario

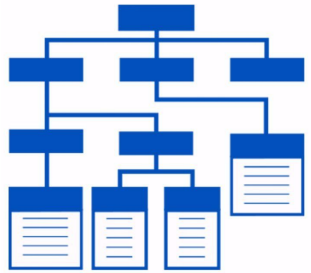
```
DROP USER usuario@dominio;
```

Esquema de Privilegios

Privilegios

La administración de privilegios y usuarios en MySQL se realiza a través de las sentencias:

- GRANT. Otorga privilegios a un usuario, en caso de no existir, se creará el usuario.
- REVOKE. Elimina los privilegios de un usuario existente.
- SET PASSWORD. Asigna una contraseña.
- DROP USER. Elimina un usuario.



Esquema de Privilegios

La sentencia GRANT

La sintaxis simplificada de grant consta de tres secciones. No puede omitirse ninguna, y es importante el orden de las mismas:

- **grant** lista de privilegios
- **on** base de datos.tabla
- **to** usuario

Ejemplo:

```
grant update, insert, select
on ucc.precios
to visitante@localhost ;
```

Esquema de Privilegios

La sentencia GRANT

Ejemplo:

```
grant update, insert, select
on ucc.precios
to visitante@localhost ;
```

Nota: El comando grant crea la cuenta si no existe y, si existe, agrega los privilegios especificados.

Esquema de Privilegios

La sentencia GRANT

Es posible asignar una contraseña a la cuenta al mismo tiempo que se crea y se le otorgan privilegios:

```
grant update, insert, select
on ucc.precios
to visitante@localhost identified by ’nuevaClave’;
```

Esquema de Privilegios

La sentencia GRANT

Incluso es posible asignar a varios usuarios al mismo tiempo:

```
grant update, insert, select
on ucc.precios
to visitante@localhost,
yo@localhost identified by ‘nuevaclave’,
tu@equipo.remoto.com;
```

Esquema de Privilegios

Especificación de lugares origen de la conexión

MySQL proporciona mecanismos para permitir que el usuario realice su conexión desde diferentes equipos dentro de una red específica, sólo desde un equipo, o únicamente desde el propio servidor.

```
grant update, insert, select
on ucc.precios
to aplicacionX@'%.ucc.edu.co';
```

Nota: El carácter % se utiliza de la misma forma que en el comando **like**.
Nota 2: Por lo tanto y al igual que en **like**, puede utilizarse el carácter '_'.

Esquema de Privilegios

Especificación de lugares origen de la conexión

Entonces, para permitir la entrada desde cualquier equipo en Internet?

Escribiríamos:

to aplicacionX…..?

Esquema de Privilegios

Especificación de lugares origen de la conexión

Los hosts (anfitriones) válidos también se pueden especificar con sus direcciones IP.

Ejemplo:

to app1@192.168.128.10

to app2@’192.168.128.%’

Nota: Los caracteres ’ %’ y ’_’ no se permiten en los nombres de los usuarios

Esquema de Privilegios

Especificación de bases de datos y tablas

limitar los privilegios a bases de datos, tablas y columnas.

Ejemplo:

```
grant all
on ucc.*
to 'visitante'@'localhost';
```

privilegios sobre todas las tablas de la base de datos **ucc**

Esquema de Privilegios

Especificación de bases de datos y tablas

limitar los privilegios a bases de datos, tablas y columnas.

Ejemplo dentro de la base de datos en cuestión:

```
mysql> use demo;

grant all
on *
to 'visitante'@'localhost';
```

privilegios sobre todas las tablas de la base de datos **demo**

Esquema de Privilegios

Especificación de bases de datos y tablas

Opciones para la clausula on del comando grant

Opción	Significado
.	Todas las bases de datos y todas las tablas
base.*	Todas las tablas de la base de datos especificada
tabla	Tabla especificada de la base de datos en uso
*	Todas las tablas de la base de datos en uso

Esquema de Privilegios

Especificación de columnas

Columnas sobre las que se otorgan privilegios con el comando **grant**

Ejemplo

```
grant select, update (id_cliente, nombre_cliente)
on ucc.cliente
to visitante@localhost;
```

Esquema de Privilegios

Especificación de columnas

Se puede especificar privilegios diferentes para cada columna o grupos de columnas

```
grant grant update(ic_cuent), select (telefono, email)
on ucc.cliente
to visitante@localhost;
```

Esquema de Privilegios

Tipos de privilegios

MySQL proporciona una gran variedad de tipos de privilegios

- **Privilegios relacionados con tablas**: alter, create, delete, drop, index, insert, select, update

- **Algunos privilegios administrativos:** file, proccess, super reload, replication client, grant option, shutdown

- **Algunos privilegios para fines diversos:** lock tables, show databases, create temporary tables.

Esquema de Privilegios

Tipos de privilegios

MySQL proporciona una gran variedad de tipos de privilegios

- **Privilegios relacionados con tablas**: alter, create, delete, drop, index, insert, select, update

- **Algunos privilegios administrativos:** file, proccess, super reload, replication client, grant option, shutdown

- **Algunos privilegios para fines diversos:** lock tables, show databases, create temporary tables.

Esquema de Privilegios

Tipos de privilegios

El privilegio ***all*** otorga todos los privilegios exceptuando el privilegio ***grant option***.

Y el privilegio ***usage*** no otorga ninguno, lo cual es útil cuando se desea, por ejemplo, simplemente cambiar la contraseña:

```
grant usage
on *.*
to visitante@localhost identified by 'nuevaClave';
```

Esquema de Privilegios

Tipos de privilegios

Tipo de privilegio	Operación que permite
all [privileges]	Otorga todos los privilegios excepto **grant option**
usage	No otorga ningún privilegio
alter	Privilegio para alterar la estructura de una tabla
create	Permite el uso de **create table**
delete	Permite el uso de **delete**
drop	Permite el uso de **drop table**
index	Permite el uso de **index** y **drop index**
insert	Permite el uso de **insert**
select	Permite el uso de **select**
update	Permite el uso de **update**
file	Permite le uso de **select . . . into outfile** y **load data infile**
process	Permite el uso de **show full procces list**
super	Permite la ejecución de comandos de supervisión
reload	Permite el uso de **flush**

Esquema de Privilegios

Tipos de privilegios

Tipo de privilegio	Operación que permite
replication client	Permite preguntar la localización de maestro y esclavo
replication slave	Permite leer los *binlog* del maestro
grant option	Permite el uso de **grant** y **revoke**
shutdown	Permite dar de baja al servidor
lock tables	Permite el uso de **lock tables**
show tables	Permite el uso de **show tables**
create temporary tables	Permite el uso de **create temporary table**

Esquema de Privilegios

Tipos de privilegios

Delegar el trabajo de administrar un servidor de bases de datos por otros usuarios, Esto se puede hacer en MySQL con el privilegio **grant option**:

```
grant all, grant option
on ucc.*
to administrador2@localhost;
```

Esquema de Privilegios

Tipos de privilegios

Delegar el trabajo de administrar un servidor de bases de datos por otros usuarios, Esto se puede hacer en MySQL con el privilegio **grant option**:

El mismo resultado se puede lograr así:

```
grant all
on ucc.*
to administrador2@localhost
with grant option;
```

Esquema de Privilegios

Limites de uso

Los recursos físicos del servidor siempre son limitados.

limitar a los usuarios el trabajo que pueden pedir al servidor con tres parámetros:

- Máximo número de conexiones por hora.
- Máximo número de consultas por hora.
- Máximo número de actualizaciones por hora

Esquema de Privilegios

Limites de uso

Los recursos físicos del servidor siempre son limitados.

Sintaxis:

```
grant all
on *.*
to
with MAX_CONECTIONS_PER_HOUR 3
MAX_QUERIES_PER_HOUR 300
MAX_UPDATES_PER_HOUR 30;
```

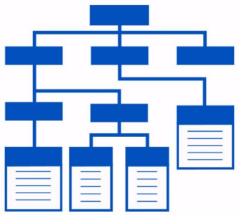
Esquema de Privilegios

Eliminar privilegios

El comando revoke permite eliminar privilegios otorgados con grant a los usuarios

Sintaxis:

```
revoke all
on *.*
from visitante@localhost;
```



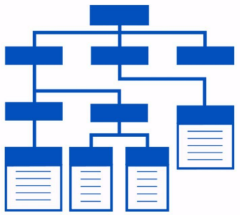
Esquema de Privilegios

Eliminar privilegios

El comando revoke permite eliminar privilegios otorgados con grant a los usuarios

Sintaxis:

```
REVOKE ALL PRIVILEGES
ON * . *
FROM 'nombre_usuario'@'localhost';
```



Esquema de Privilegios

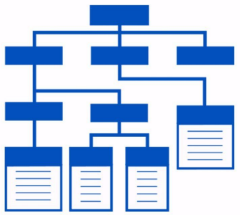
Consultar privilegios

MySQL almacena la información sobre los usuarios y sus privilegios en una base de datos como cualquier otra, cuyo nombre es **mysql**.

```
use mysql;

show tables;

desc user;
```



Esquema de Privilegios

Consultar privilegios

Es posible realizar modificaciones directamente sobre estas tablas y obtener los mismos resultados que si utilizáramos los comandos **grant**, **revoke**, **set password** ó **drop user**:

```
update user
set Password = password('nuevapasswd')
where User ='visitante' and Host = 'localhost';
```

flush privileges;

Nota: El comando **flush privileges** solicita a MySQL que vuelva a leer las tablas de

privilegios.

Esquema de Privilegios

Tablas de la base de datos *mysql*

Tabla	Contenido
user	Cuentas de usuario y sus privilegios globales
db	Privilegios sobres bases de datos
tables_priv	Privilegios sobre tablas
columns_priv	Privilegios sobre columnas
host	Privilegios de otros equipos anfitriones sobre bases de datos

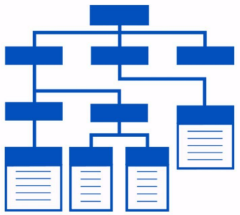
Nota: host es ahora: servers

Esquema de Privilegios

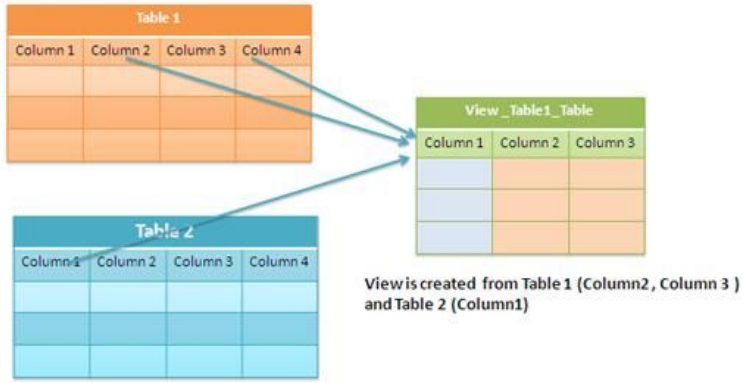
Consultar privilegios

Sintaxis:

```
SHOW GRANTS FOR 'app1'@localhost;
```



Procedimientos almacenados



Procedimientos almacenados

Los procedimientos almacenados son un conjunto de instrucciones SQL más una serie de estructuras de control que nos permiten dotar de cierta lógica al procedimiento

Para llamar a un procedimiento lo hacemos mediante la instrucción CALL. Desde un procedimiento podemos invocar a su vez a otros procedimientos o funciones.

Procedimientos almacenados

Un procedimiento almacenado, al igual cualquiera de los procedimientos que podamos programar en nuestras aplicaciones utilizando cualquier lenguaje, tiene:

- Un **nombre**
- **Lista de parámetros**. (opcional)
- **Contenido** (sección también llamada definición del procedimiento).

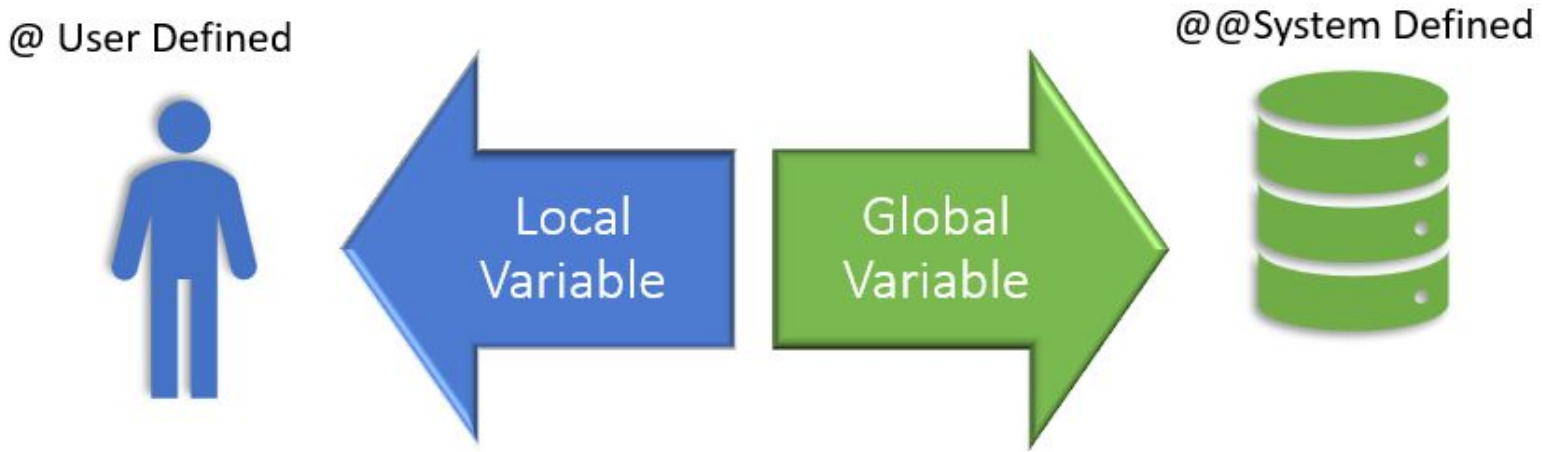
El contenido puede estar compuesto por

1. **instrucciones sql**
2. **estructuras de control**
3. **declaración de variables locales**
4. **control de errores**
5. **etcétera.**

Procedimientos almacenados

Variables locales

SET @var=value



Procedimientos almacenados

Sintaxis de un procedimiento almacenado:

CREATE PROCEDURE nombre (parámetro)

[características] definición

Ó

```
DELIMITER //
    CREATE PROCEDURE Reset_animal_count()
    MODIFIES SQL DATA
    UPDATE animal_count SET animals = 0;
// DELIMITER ;
```

Procedimientos almacenados

Como un procedimiento almacenado puede tener muchos comandos SQL entre las palabras claves **begin** y **end** se debe indicar a MySQL que no ejecute dichos puntos y coma.

```
create procedure NOMBREPROCEDIMIENTO()
DELIMITER //
begin
    INSTRUCCIONES;
end //
DELIMITER ;
```

Procedimientos almacenados

Con parámetros

```
CREATE PROCEDURE RetirarDinero  (
        valorRetirar DECIMAL(6,2),
        cajero VARCHAR(50),
        cliente_id INTEGER
)
 MODIFIES SQL DATA
 BEGIN
        UPDATE Cuenta
        SET saldo = saldo - valorRetirar
        WHERE id_cliente = cliente_id;

        INSERT INTO bitacora_retiros VALUES (valorRetirar, cajero, cliente_id);

 END;
```

Procedimientos almacenados

De consultas

```
create procedure listar_clientes()
 begin
   select * clientes;
 end;
```

Para llamar a los procedimientos:

```
call listar_clientes();
```

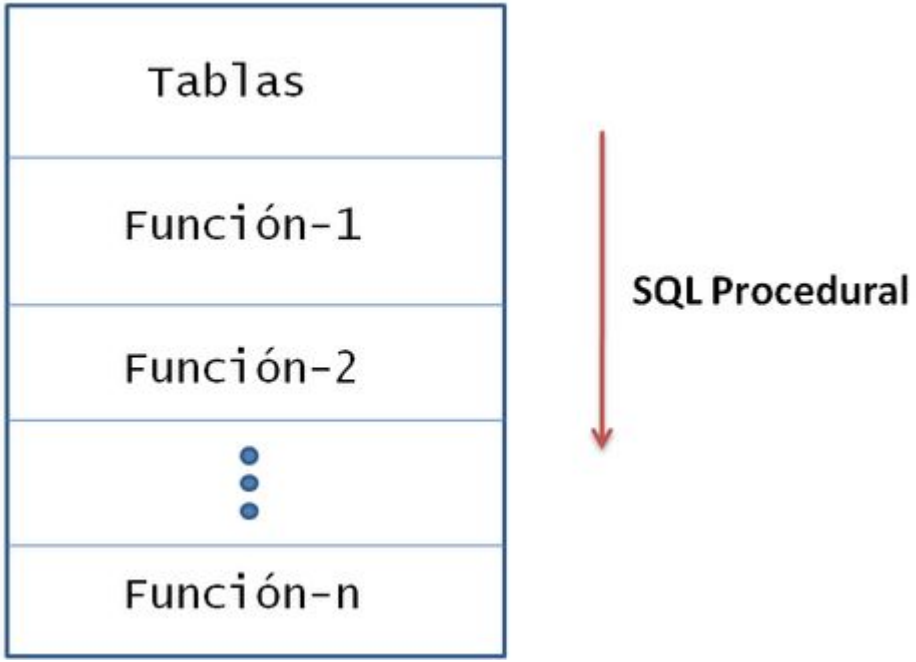
Procedimientos almacenados

Bloque de instrucción

BEGIN
...
Instruccion_1
Instruccion_2

[BREAK | CONTINUE]

...
END



Procedimientos almacenados

Instrucción IF

```
BEGIN
...
        IF ( expresion OPERADOR expresion)
                Expresion
        ELSE BEGIN
                Expresion
        END
...
END
```

Procedimientos almacenados

Instrucción WHILE

```
BEGIN
...
        WHILE ( expresion OPERADOR expresion)
        BEGIN
                Expresion
                ....
        END
...
END
```

Procedimientos almacenados

Crear un procedimiento almacenado que devuelva una consulta

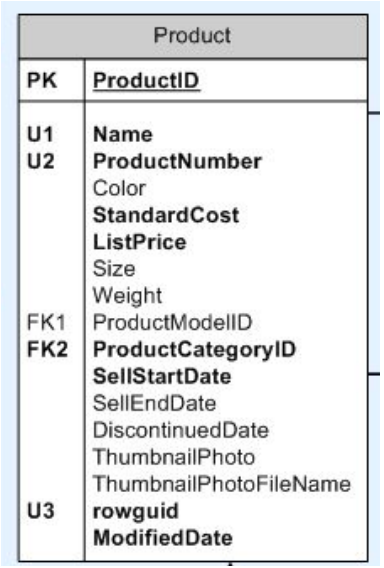
```
CREATE PROCEDURE sp_listaProductos AS
BEGIN
	SELECT
		Name,
			ListPrice
	FROM SalesLT.Product
	ORDER BY Name;
END;
```

Procedimientos almacenados

Ventajas creación Procedimientos almacenados

- Mejora rendimiento base de datos
- Control de flujo de ejecuciones

Ejemplo: Se requiere constantemente estar consultando los productos con su respectivo precio y ordenados por nombre



Product	
PK	**ProductID**
U1	**Name**
U2	**ProductNumber**
	Color
	StandardCost
	ListPrice
	Size
	Weight
FK1	ProductModelID
FK2	**ProductCategoryID**
	SellStartDate
	SellEndDate
	DiscontinuedDate
	ThumbnailPhoto
	ThumbnailPhotoFileName
U3	**rowguid**
	ModifiedDate

Procedimientos almacenados

Ejecutar Procedimiento

```
EXECUTE sp_listaProductos

EXEC sp_listaProductos
```

Procedimientos almacenados

Modificar Procedimiento

```
ALTER PROCEDURE sp_listaProductos    AS
      BEGIN
                  SELECT Name, ListPrice
                  FROM salesLT.Product
                  ORDER BY ListPrice;
      END;
```

Procedimientos almacenados

Eliminar Procedimiento

```
DROP PROCEDURE sp_listaProductos();
```

Mostrar un procedimiento:

```
SHOW CREATE PROCEDURE ucc.resetear;
```

```
SHOW PROCEDURE STATUS LIKE 'resetear'
```

Procedimientos almacenados

Procedimiento con parámetros

- Favorecen la reutilización
- Estructuran la lógica de datos

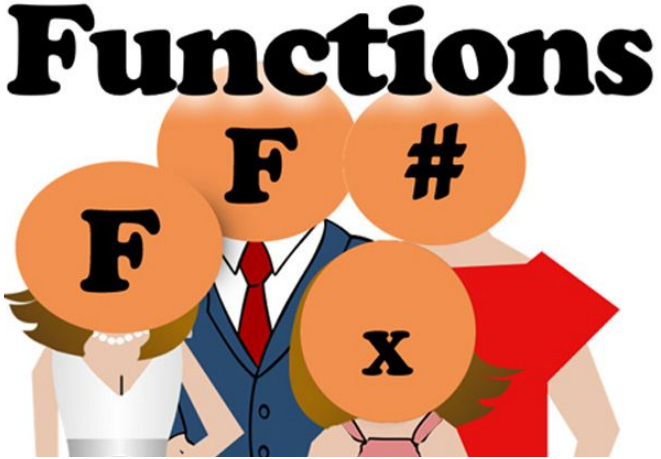
Ejemplo: Se requiere rápidamente estar consultando los productos con su respectivo precio dependiendo de su color

Procedimientos almacenados

Ejercicio Banco

Realizar un procedimiento almacenado sp_prestamosValor que pida un valor mínimo y un valor máximo y consulte los préstamos de clientes entre esos valores

Funciones



Funciones de Usuario

Las **funciones** son como Procedimientos definidos por el usuario que retornan un valor:

- **Escalar**

Encapsulan complejidad para su reutilización

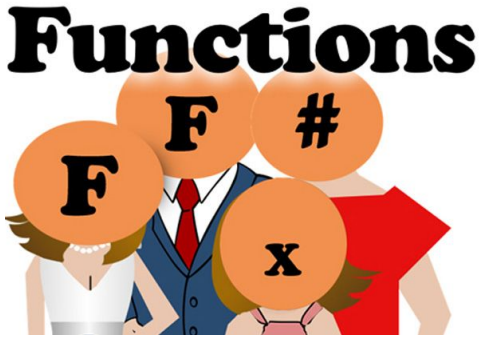
Pueden utilizar lógicas complejas internas (IF, WHILE, etc)

Incluso llamar otras funciones

Funciones de Usuario

Sintaxis básica funciones ESCALARES

```
CREATE FUNCTION nombre_funcion (parametros)
        RETURNS INT
BEGIN
        …. Sentencias …..
        RETURN valor
END
```



Funciones de Usuario

Sintaxis básica funciones ESCALARES

- **Nombre_funcion**. El nombre del esquema es opcional
- Lista de parámetros
- **Tipo de datos** para el retorno
- Incluir una respuesta con el **RETURN**

Funciones de Usuario

Ejemplo

Realizar una función de usuario para calcular el cuadrado de un número

```
delimiter //

CREATE FUNCTION cuadrado (s SMALLINT) RETURNS SMALLINT
        RETURN s*s;
//

delimiter ;
```

Funciones de Usuario

Llamar una función

```
SELECT cuadrado ( 4 ) as cuadrado;
```

Funciones de Usuario

Eliminar una función

```
DROP FUNCTION cuadrado;
```

Procedimientos almacenados

Ejercicio Banco

* Realizar una función de usuario que calcule el interés del 1
de un préstamo. Luego hacer la consulta de los préstamo con su respectivo interés.

Triggers - Disparadores

Triggers (Disparadores)

Ejecuciones automáticas en **respuesta** a un EVENTO de base de datos

- Disparadores DML
  - INSERT
  - UPDATE
  - DELETE
- Disparadores DDL
  - CREATE
  - ALTER
  - DROP
- Disparadores de sesión (LOGON)

Triggers (Disparadores)

Las ejecuciones se realizan antes ó después de que suceda el evento

- BEFORE: antes de que ocurra el evento
- AFTER: después de que ocurra el evento

El disparador tiene:

1. Un nombre
2. Un momento del disparador (after, before)
3. Un evento (INSERT)
4. Una tabla
5. Un conjunto de sentencias SQL

Triggers (Disparadores)

Crear un disparador DML

CREATE TRIGGER [schema_name.]**nombre_disparador**

AFTER {[INSERT],[UPDATE],[DELETE]} ON **tabla**

FOR EACH ROW

{sentencias sql}

Triggers (Disparadores)

Crear un disparador DML

```
CREATE TRIGGER increment_animal

AFTER INSERT ON animals

FOR EACH ROW

UPDATE animal_count SET animal_count.animals = animal_count.animals+1;
```

Triggers (Disparadores)

Crear un disparador DML

Al “suceder” el evento, el SGBD genera una tabla temporal para los valores insertados o eliminados en un momento ANTES y DESPUES de que ocurran

Triggers (Disparadores)

Ejemplo un disparador DML

Auditoría / Trazabilidad de operaciones:

Requisito de calidad: INTEGRIDAD de los datos

Cada registro en la base de datos que se modifica, elimina o inserta debe tener una traza de quién, cuándo y qué fue modificado.

Triggers (Disparadores)

Ejemplo un disparador DML

Crear tabla de auditoría:

```
CREATE TABLE auditoriaTransacciones(
        id_auditoria INT AUTOINCREMENT,
        fecha DATETIME NOT NULL,
        id_objeto INT,
        tabla VARCHAR(100),
        operation CHAR(3) NOT NULL,
        CHECK(operation = 'INS' or operation='DEL' or operation='UPD')
);
```

Triggers (Disparadores)

Ejemplo un disparador DML

Crear el disparador:

```
CREATE TRIGGER tgr_eliminarCliente
    AFTER DELETE ON ucc.clientes
    FOR EACH ROW
    BEGIN
        INSERT INTO ucc.auditoriaTransacciones
            (fecha, id_objeto, tabla, operation) values
            (NOW(), OLD.cliente_id, 'Cliente', 'DEL')
    END
```

Triggers (Disparadores)

Disparador más complejo

```
DELIMITER //

CREATE TRIGGER LosRatonesInvaden
AFTER INSERT ON Animales
FOR EACH ROW

BEGIN

        IF NEW.Nombre = 'Mouse' THEN
                UPDATE ConteoDeAnimales SET ConteoDeAnimales.Animales = ConteoDeAnimales.Animales +
100;
        ELSE
                UPDATE ConteoDeAnimales SET ConteoDeAnimales.Animales = ConteoDeAnimales.Animales + 1;
        END IF;

  END; //
DELIMITER ;
```

Triggers (Disparadores)

Taller:

- Desarrollar los triggers **delete** para las tablas: prestamo y cuenta
- Realizar tabla de auditoria en una base de datos nueva llamada: audit
  - Crear una tabla de auditoria por cada ejercicio
- Ir a cada base de datos de cada ejercicio:
  - Crear los disparadores Insert, Update y delete para 3 tablas