

ABP BASE DE DATOS

INTEGRANTES

Jesús David Montiel Vergara

Perez Arrieta Boris Yesid

PROFESOR

Jhon Carlos Arrieta

CARTAGENA, BOLÍVAR

22/04/2021

VISTAS

Una vista es una tabla virtual basada en una tabla u otra vista, no contiene datos en sí misma, pero es como una ventana a través de la cual se pueden ver o cambiar los datos de las tablas. Podemos representar con ellas subconjuntos lógicos o combinación de datos, las tablas sobre las cuales se basa una vista se llaman tablas base.

¿ Por qué usar Vistas ?

- Para restringir el acceso a la B.D.
- Para realizar consultas complejas de manera fácil.
- Para obtener una independencia de los datos.
- Para presentar diferentes vistas de los mismos datos.

MySQL:

Creación: Se crea una vista con todos las filas y columnas de la tabla_basada.

CREATE VIEW VistaAcessoAS SELECT * FROM acceso;

Actualizar: A diferencia de la anterior, en esta utilizamos CREATE OR REMPLACE VIEW

CREATE OR REPLACE VIEW nombre_vista_crear AS SELECT * FROM tabla_basada;

Eliminar: Para eliminar una vista simplemente se utiliza el comando DROP VIEW y el nombre de la vista:

DROP VIEW nombre_vista_crear;

Oracle:

Creación: Crear una vista sobre la tabla que uno la quiere hacer;

```
CREATE VIEW nombre_vista_crear
AS

SELECT nombre,
    apellidos,
    matricula
FROM tabla_basada;
```

ALTER VIEW: Si queremos modificar la definición de la vista se utiliza la sentencia ALTER VIEW, de forma muy parecida a como se hacían con las tablas.

```
ALTER VIEW nombre_vista_crear

AS
(
SELECT nombre,
    apellidos,
    matricula,
    fx_alquiler,
    fx_devolucion

FROM tabla_basada;
```

Eliminar: También podemos eliminar las vistas a través de la sentencia DROP VIEW. para eliminar la vista que hemos creado.

```
DROP VIEW nombre_vista_creada;
```

PostgreSQL:

Crear: Para crear una vista en este motor es totalmente igual a MySQL. Respetando el estándar básico de SQL

```
CREATE VIEW nombre_vista_crear AS SELECT * FROM tabla_basada;
```

Eliminar: También podemos eliminar las vistas a través de la sentencia DROP VIEW. para eliminar la vista que hemos creado.

```
DROP VIEW nombre_vista_creada;
```

SQL Server:

Creación: Podemos crear la vista usando la sentencia CREATE VIEW. Una vista se puede crear desde una sola tabla o varias tablas.

```
CREATE VIEW view_name AS

SELECT column1, column2....

FROM table_name

WHERE condition;
```

Universidad del sinú

Creación de vistas en MySQL

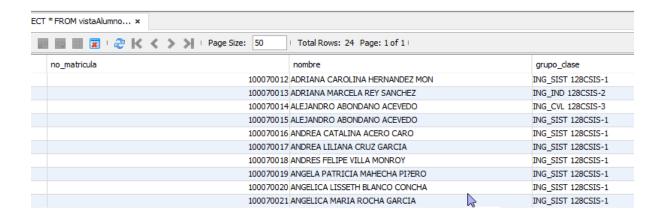
```
CREATE VIEW vistaAcceso AS SELECT * FROM acceso; --Creación de la vista
SELECT * FROM vistaAcceso; --Mostramos el contenido de la vista que se acaba de crear
```

Resultado:

T *FROM vistaAcesso ×	
	ize: 50 Total Rows: 47 Page: 1 of 1
codigo	nombre
100070012	ADRIANA CAROLINA HERNANDEZ MONTERROZA
100070013	ADRIANA MARCELA REY SANCHEZ
100070014	ALEJANDRO ABONDANO ACEVEDO
100070015	ALEJANDRO ABONDANO ACEVEDO
100070016	ANDREA CATALINA ACERO CARO
100070017	ANDREA LILIANA CRUZ GARCIA
100070018	ANDRES FELIPE VILLA MONROY
100070019	ANGELA PATRICIA MAHECHA PI?EROS
100070020	ANGELICA LISSETH BLANCO CONCHA
100070021	ANGELICA MARIA ROCHA GARCIA
100070022	ANGIE TATIANA FERN?NDEZ MART?NEZ
100070023	TATIANA ANGIE MART?NEZ FERN?NDEZ
100070024	CAMILO VILLAMIZAR ARISTIZABAL

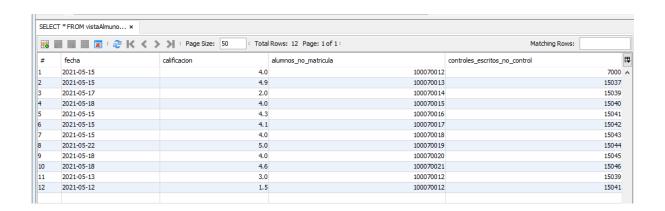
CREATE VIEW vistaAlumnos AS SELECT * FROM alumnos; --Creación de la vista SELECT * FROM vistaAlumnos; --Mostramos el contenido de la vista que se acaba de crear

Resultado:



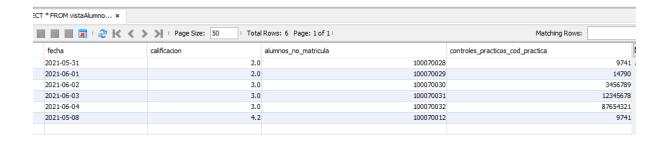
CREATE VIEW vistaAlmunosControlesEscritros AS SELECT * FROM alumnos_controles_escritos; --Creación de la vista SELECT * FROM vistaAlmunosControlesEscritros; --Mostramos el contenido de la vista que se acaba de crear

Resultado:



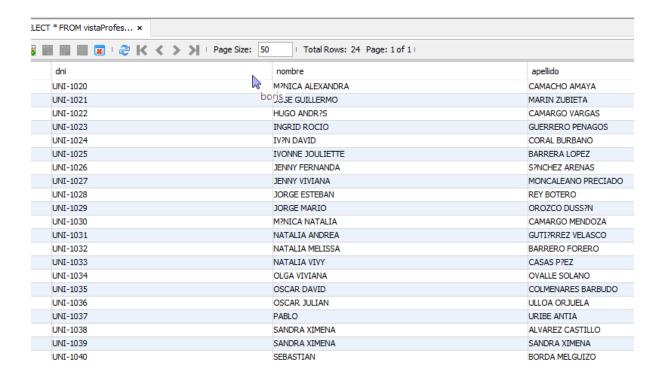
CREATE VIEW vistaAlumnosControlesPracticos AS SELECT * FROM alumnos_controles_practicos; -- Creación de la vista SELECT * FROM vistaAlumnosControlesPracticos; -- Mostramos el contenido de la vista que se acaba de crear

Resultado:



CREATE VIEW vistaProfesores AS SELECT * FROM profesores; -- Creación de la vista SELECT * FROM vistaProfesores; --Mostramos el contenido de la vista que se acaba de crear

Resultado:



Creación de vistas en SQL Server

CREATE VIEW vistaAlumnos AS SELECT * FROM alumnos; --Creación de la vista

Resultado:

Messages

Commands completed successfully.

Completion time: 2021-05-21T13:45:42.5843842-05:00

SELECT * FROM vistaAlumnos; --Mostramos el contenido de la vista que se acaba de crear

Resultado:

■	Results 🗐 Me	essages	
	no_matricula	nombre	grupo_clase
1	100070012	ADRIANA CAROLINA HERNANDEZ MONTERROZA	ING_SIST 128CSIS-1
2	100070013	ADRIANA MARCELA REY SANCHEZ	ING_IND 128CSIS-2
3	100070014	ALEJANDRO ABONDANO ACEVEDO	ING_CVL 128CSIS-3
4	100070015	ALEJANDRO ABONDANO ACEVEDO	ING_SIST 128CSIS-1
5	100070016	ANDREA CATALINA ACERO CARO	ING_SIST 128CSIS-1
6	100070017	ANDREA LILIANA CRUZ GARCIA	ING_SIST 128CSIS-1
7	100070018	ANDRES FELIPE VILLA MONROY	ING_SIST 128CSIS-1
8	100070019	ANGELA PATRICIA MAHECHA PIÑEROS	ING_SIST 128CSIS-1
9	100070020	ANGELICA LISSETH BLANCO CONCHA	ING_SIST 128CSIS-1
10	100070021	ANGELICA MARIA ROCHA GARCIA	ING_SIST 128CSIS-1
11	100070022	ANGIE TATIANA FERNÁNDEZ MARTÍNEZ	ING_SIST 128CSIS-1

CREATE VIEW vistaAcceso AS SELECT * FROM acceso; -- Creación de la vista

Resultado:



Commands completed successfully.

Completion time: 2021-05-21T13:45:42.5843842-05:00

SELECT * FROM vistaAcceso; --Mostramos el contenido de la vista que se acaba de crear

Resultado:

≡	Results	Ball W	essages		
	codigo)	nombre	pass	rol
1	10007	70012	ADRIANA CAROLINA HERNANDEZ MONTERROZA	12345	alumno
2	10007	70013	ADRIANA MARCELA REY SANCHEZ	12346	alumno
3	10007	70014	ALEJANDRO ABONDANO ACEVEDO	12347	alumno
4	10007	70015	ALEJANDRO ABONDANO ACEVEDO	12348	alumno
5	10007	70016	ANDREA CATALINA ACERO CARO	12349	alumno
6	10007	70017	ANDREA LILIANA CRUZ GARCIA	12350	alumno
7	10007	70018	ANDRES FELIPE VILLA MONROY	12351	alumno
8	10007	70019	ANGELA PATRICIA MAHECHA PIÑEROS	12352	alumno
9	10007	70020	ANGELICA LISSETH BLANCO CONCHA	12353	alumno
10	10007	70021	ANGELICA MARIA ROCHA GARCIA	12354	alumno
11	10007	70022	ANGIE TATIANA FERNÁNDEZ MARTÍNEZ	12355	alumno

Creación de vistas en SQL Server

CREATE VIEW vistaProfesoresAS SELECT * FROM profesores; -- Creación de la vista

Resultado:

Data Output Explain Messages Notifications

CREATE VIEW

Query returned successfully in 473 msec.

SELECT * FROM vistaAcceso; --Mostramos el contenido de la vista que se acaba de crear

Resultado:

Data Output Explain Messages Notifications				
4	dni character varying (15)	nombre character varying (30)	apellido character varying (30)	
1	UNI-1020	MàNICA ALEXANDRA	CAMACHO AMAYA	
2	UNI-1021	JOSE GUILLERMO	MARIN ZUBIETA	
3	UNI-1022	HUGO ANDRES	CAMARGO VARGAS	
4	UNI-1023	INGRID ROCIO	GUERRERO PENAGOS	
5	UNI-1024	IVμN DAVID	CORAL BURBANO	
6	UNI-1025	IVONNE JOUI IFTTE	BARRERA LOPEZ	

Transacciones

Una transacción es una unidad de trabajo compuesta por diversas tareas, cuyo resultado final debe ser que se ejecuten todas o ninguna de ellas. Una transacción tiene 4 propiedades fundamentales que son:

Atomicidad: Significa que una transacción es una unidad indivisible, es decir que se ejecuta o no se ejecuta

Creación de transacciones en MySQL con Java

Creamos los query:

Creamos la transacción:

En java, insertamos dentro de un try catch toda nuestra transacción:

```
try {
        miConexion = DriverManager.getConnection(url, user, pass); //Creamos la conexión con la BD.
        miConexion.setAutoCommit(false);
        Statement miStatement = miConexion.createStatement();
        miStatement.executeUpdate(insertarAlumnos); //Pasamos el query de insertar alumnos creado anteriormente.
        miStatement.executeUpdate(insertarAcceso); //Pasamos el query de insertar acceso creado anteriormente.
        miStatement.executeUpdate(insertarCalificacionControlEscrito); //Pasamos el query de insertar acceso calificaciones escritas.
        miStatement.executeUpdate(insertarCalificacionControlPractico); //Pasamos el query de insertar calificaciones practicas.
        miConexion.commit(); // Utilizamos este comando para guardar todos los cambios.
        resultado = "Todo Ok"; //Si no se encuentra ningún problema nos arroja como resultado "Todo ok".
     } catch (SQLException e) {
          resultado = "Ocurrió un error, no se hizo";//En caso de que haya algún problema nos mostrará este mensaje y NO se realiza la consulta.
          System.out.println("ERROR:
          if (miConexion != null) {
              try {
                 miConexion.rollback(); //Usamos rollback() para revertir todos los cambios que se hubieran hecho en la transacción.
              } catch (SQLException ex) {
                 Logger.getLogger(conexion.class.getName()).log(Level.SEVERE, null, ex);
```

Universidad del sinú

Por defecto, MySQL se ejecuta en modo autocommit. Esto significa que tan pronto como se ejecuta una sentencia se actualiza (modifica) la tabla. Para evitar lo anterior utilizamos el siguiente comando:

miConexion.setAutommit(false);

```
try {
   miConexion = DriverManager.getConnection(url, user, pass);
   miConexion.setAutoCommit(false);
   Statement miStatement = miConexion.createStatement();
   miStatement.executeUpdate(insertarAlumnos);
   miStatement.executeUpdate(insertarAcceso);
   miStatement.executeUpdate(insertarCalificacionControlEscrito);
   miStatement.executeUpdate(insertarCalificacionControlPractico);
   miConexion.commit();
   resultado = "Todo Ok";
} catch (SQLException e) {
   resultado = "Ocurrió un error, no se hizo";
   System.out.println("ERROR: " + e);
   if (miConexion != null) {
       trv {
           miConexion.rollback();
       } catch (SQLException ex) {
           Logger.getLogger(conexion.class.getName()).log(Level.SEVERE, null, ex);
```

```
public String registrarAlumno(int noMatricula, String nombre, String grupoClase, String fecha, float
calificacionEscrito, float calificacionPractico, int noControlEscrito, int noControlPractico) {
       String resultado;
       Connection miConexion = null;
       String insertarAlumnos = ("INSERT INTO alumnos(no matricula, nombre, grupo clase) VALUES ('" + noMatricula +
"','" + nombre + "', '" + grupoClase + "')");
       String insertarAcceso = ("INSERT INTO acceso(codigo, nombre, password, rol) VALUES ('" + noMatricula + "',
'" + nombre + "', '" + noMatricula + "', 'alumno')");
       String insertarCalificacionControlEscrito = ("INSERT INTO alumnos controles escritos(fecha, calificacion,
alumnos no matricula, controles escritos no control) VALUES ('" + fecha + "','" + calificacionEscrito + "', '" +
noMatricula + "', '" + noControlEscrito + "')");
       String insertarCalificacionControlPractico = ("INSERT INTO alumnos controles practicos(fecha, calificacion,
alumnos_no_matricula, controles_practicos_cod practica) VALUES ('" + fecha + "','" + calificacionPractico + "', '" +
noMatricula + "', '" + noControlPractico + "')");
       try {
            miConexion = DriverManager.getConnection(url, user, pass);
            miConexion.setAutoCommit(false);
            Statement miStatement = miConexion.createStatement();
            miStatement.executeUpdate(insertarAlumnos);
            miStatement.executeUpdate(insertarAcceso);
            miStatement.executeUpdate(insertarCalificacionControlEscrito);
            miStatement.executeUpdate(insertarCalificacionControlPractico);
            miConexion.commit();
            resultado = "Todo Ok";
        } catch (SQLException e) {
            resultado = "Ocurrió un error, no se hizo";
```

Usuarios en MySQL

Crear:

La sentencia para crear un usuario en MySQL es la siguiente:

```
CREATE USER 'usuario'@'localhost' IDENTIFIED BY 'password';
```

Con el comando "CREATE USER" le indicamos que queremos crear un usuario y seguido con comillas simples ('') se indica el nombre del usuario a crear, en este caso el nombre fue: 'JM-BP' seguido de un arroba (@) y se indica el nombre del servidor igualmente dentro de comillas simples como en nuestro caso que fue 'localhost' y seguido el comando "IDENTIFIED BY" para indicarle la contraseña que queremos establecer la cual la debemos escribir dentro de comillas simples como en nuestro ejemplo que fue: '123456'.

```
CREATE USER 'JM-BP'@'localhost' IDENTIFIED BY '123456';
```

Resultado:

```
mysql> create user 'JM-BP' identified by '123456';
Query OK, 0 rows affected (0.07 sec)
```

Privilegios:

Ahora recordar que solamente se creó el usuario, falta establecer privilegios a este.

Para ello utilizamos el comando "GRANT" y en nuestro ejemplo lo seguimos con "ALL" lo cual indica que va a tener todos los privilegios (SELECT, INSERT, UPDATE, DELETE, CREATE, DROP, ETC) en la tabla que le indiquemos con el comando "PRIVILEGES ON" y seguido el nombre de la base de datos en que queremos establecer esos privilegios, en nuestro caso nada más elegimos la base de datos *examenes*. En caso de querer que estos privilegios se apliquen en todas las bases de datos, solamente se coloca el símbolo asterisco (*) y seguido un punto (.) y a continuación en qué tablas, en nuestro ejemplo lo colocamos para todas las tablas con el símbolo asterisco (*) y "TO" para especificar a qué usuario es al que se lo vamos a aplicar, en este caso al mismo que creamos en el punto anterior.

```
GRANT ALL PRIVILEGES ON examenes.* TO 'JM-BP'@'localhost';
```

Resultado:

```
mysql> grant all privileges on examenes.* to 'JM-BP'@'localhost';
Query OK, 0 rows affected (0.02 sec)
```

Por último queda aplicar el comando "FLUSH PRIVILEGES" para refrescar los privilegios.

```
FLUSH PRIVILEGES;
```

Resultado:

```
mysql> flush privileges;
Query OK, 0 rows affected (0.08 sec)
```

Eliminar:

Para eliminar un usuario simplemente utilizamos el siguiente comando donde indicamos el nombre del usuario y nombre del servidor. En nuestro ejemplo utilizamos el usuario 'JM-BP' y el servidor 'localhost'.

```
DROP USER 'JM-BP'@'localhost';
```

Resultado:

```
MariaDB [(none)]> DROP USER 'JM-BP'@'localhost';
Query OK, 0 rows affected (0.067 sec)
```

Eliminar privilegios:

Para eliminar privilegios utilizamos el comando "REVOKE" seguido del "ALL" que cumple la misma función de la explicada al momento de establecer los privilegios. Luego con "PRIVILEGES ON" le indicamos si queremos quitar los privilegios en todas las bases de datos y tablas como en el ejemplo con los símbolos asterisco (*) sabiendo que el primer asteriscos bases de datos y el segundo asterisco tablas seguido del "FROM" e indicamos el nombre del usuario y servidor donde aplicaremos la revocación de privilegios.

```
REVOKE ALL PRIVILEGES ON *.* FROM 'JM-BP'@'localhost';
```

Resultado:

MariaDB [(none)]> REVOKE ALL PRIVILEGES ON *.* FROM 'JM-BP'@'localhost';
Query OK, O rows affected (0.069 sec)

Usuarios en SQL Server

Crear:

Para crear un usuario en SQL Server se utiliza el comando CREATE LOGIN seguido del nombre del usuario que en este ejemplo es: 'jmbp' y WITH PASSWORD para establecer la contraseña que en este caso sería: '123456'.

CREATE LOGIN jmbp WITH PASSWORD = '123456';

Privilegios:

Ahora solo falta agregar los privilegios.

Debemos ingresar a la base de datos donde queremos otorgarle los permisos:

USE examenes;

Utilizamos el comando GRANT y los privilegios que le daremos, en este ejemplo serán todos entonces por ello le sigue: ALL PRIVILEGES TO y se escribe el nombre del usuario que creamos en el punto anterior que sería jmbp.

GRANT ALL PRIVILEGES TO jmbp;

Eliminar:

Para eliminar un usuario simplemente utilizamos el siguiente comando donde indicamos el nombre del usuario. En nuestro ejemplo utilizamos el usuario jmbp.

DROP USER jmbp;

Usuarios en PostgreSQL

Crear:

Para crear el usuario en postgreSQL utilizamos la siguiente instrucción CREATE USER seguido del nombre del usuario a crear y colocarle la contraseña WITH PASSWORD y se le coloca la contraseña que desee en este caso se utiliza '123456'.

```
CREATE USER JMBP with password '123456';

Resultado:

postgres=# create user JMBP with password '123456';
CREATE ROLE
```

Para visualizar los usuarios creado colocamos \du y vemos que el usuario creado anteriormente no tiene ningún privilegio

Resultado:

```
Lista de roles

Nombre de rol | Atributos

jmbp |
postgres | Superusuario, Crear rol, Crear BD, Replicaci¾n, Ignora RLS
```

Para agregarle un privilegio a un usuario colocamos ALTER user seguido del usuario que se creó anteriormente en este caso JMBP dándole el privilegio de superusuario colocando WITH SUPERUSER;

```
ALTER USER JMBP WITH SUPERUSER;
```

Resultado:

```
postgres=# ALTER USER JMBP with superuser;
ALTER ROLE
```

Para darle privilegios al usuario creado al principio JMBP a una base de datos es parecido al motor de base de datos MySQL usamos GRANT ALL ON DATABASE seguido del nombre de la base de datos en este caso exámenes en el usuario creado TO JMBP

```
GRANT ALL ON DATABASE examenes TO JMBP;
```

Resultado:

postgres=# GRANT ALL ON DATABASE examenes to JMBP; GRANT postgres=#

Usuarios en ORACLE

Crear:

Para crear un usuario en ORACLE se utiliza el comando CREATE LOGIN seguido del nombre del usuario que en este ejemplo es: 'jmbp' y IDENTIFIED BY para establecer la contraseña que en este caso sería: 'contras'.

CREATE USER jmbp IDENTIFIED BY contras;

Privilegios:

Para agregar privilegios a este usuario utilizamos el comando GRANT y los privilegios que le daremos, en este ejemplo serán todos entonces por ello le sigue: ALL PRIVILEGES TO y se escribe el nombre del usuario que creamos en el punto anterior que sería jmbp.

GRANT ALL PRIVILEGES TO jmbp;

Eliminar:

Para eliminar un usuario simplemente utilizamos el siguiente comando donde indicamos el nombre del usuario. En nuestro ejemplo utilizamos el usuario jmbp.

DROP USER jmbp;

Eliminar privilegios:

Para quitar los privilegios al usuario creado anteriormente utilizaremos el comando "REVOKE" seguido del "ALL" que cumple la misma función de la explicada al momento de establecer los privilegios. Luego con "PRIVILEGES FROM" donde le diremos a cual usuario le eliminaremos los privilegios.

REVOKE ALL PRIVILEGES FROM jmbp;

Jesus Montiel - Boris Perez