



11^a
Emisión

**DIPLOMADO
Desarrollo de Sistemas
con Tecnología Java**

Módulo 3

Manejo de bases de datos con Java

Carlos Eligio Ortiz Maldonado

carloseligio@ortizm.com



UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO
Dirección General de Cómputo y de Tecnologías de información y Comunicación
Dirección de Docencia en TIC



Educación
Continua
1971 - 2021

Temario

1. Conceptos asociados a las bases de datos.
2. Modelo relacional.
3. Creación de bases de datos y tablas (DDL).
4. Uso de bases de datos MySQL.
5. Integración de bases de datos con el lenguaje de programación Java.



Software a utilizar

- Motor de bases de datos
MariaDB
- Heidi SQL
- JDK de Java
- Eclipse

ConeCTOR JDBC MariaDB-Java
Software para diagramar (draw.io)



1. Conceptos asociados a las bases de datos

1. Redundancia
2. Consistencia
3. Integridad
4. Seguridad
5. Independencia lógica de los datos
6. Independencia física de los datos



Información

- **Relevante** (Utilidad notoria, real o potencial).
- **Comprensible** (Fácil de entender).
- **Fiable** (Sin errores y de fuentes confiables).
- **Oportuna** (Disponibilidad en el momento preciso).
- **Redituable** (Que el Costo-Beneficio de Obtención y su Utilidad sean viables)
- **Verificable** (Comprobable en cualquier momento).



Conceptos de bases de datos

- Bases de datos
- Tabla, fila, columna
- SQL
- Reglas de Codd
- **Redundancia.** Control de información repetida.
- **Consistencia.** Que los datos sigan las reglas del negocio.
- **Integridad.**
 - De campo.
 - De entidad.
 - Referencial.
- **Seguridad.**

Independencia de los datos

- Independencia física
 - Discos
 - Versiones
 - Otras bases de datos
 - Etc.
- Independencia lógica
 - Cambios a la estructura
 - Modificación de relaciones
 - Reordenamiento de campos
 - Etc.

2. Modelo relacional

- 1. Elementos del modelo relacional**
- 2. Índices**

¿Qué se diseña?

- Base de datos: Motor, ubicación, juego de caracteres
- Tabla: Información que contiene, relaciones, cardinalidad
- Fila/Tupla/Renglón: Granularidad
- Columna/Atributo/Campo: Tipo de datos, dominio, restricciones



Consideraciones

A nivel de tabla

- Contenido
- Granularidad
- Relaciones.
- Cardinalidad
- Grado
- Índices

A nivel de campo

- Tipo de dato
- Longitud
- Dominio
- Restricciones
 - Nulos
 - Valores únicos
 - Dominio
 - Llaves



Llaves

- Principal
- Candidatas
- Secundarias
- Foráneas



Llaves o claves candidatas

Es el atributo o atributos que *podrían* servir como llaves primarias. Una llave candidata debe cumplir dos condiciones:

1. **Unicidad:** no pueden existir dos tupías con el mismo valor en todos los atributos que forman la llave candidata.
2. **Irreductibilidad o Minimalidad:** no existe ningún otro subconjunto de la llave que cumpla la regla de unicidad.



Llave principal

- Campo o campos que hacen de cada registro un registro único.
- El motor de base de datos se encarga de que nunca se repitan los valores de la llave principal.
- Con saber la llave principal se puede tener acceso al registro.
- En muy pocas situaciones se podrán tener tablas sin llave principal explícita.
- Si no hay una combinación de campos que haga único el registro, se podrá añadir una llave artificial, aunque puede ser error de diseño.



Llave secundaria

- Llaves candidatas que no se eligieron como primaria, es decir, tienen todas las características para ser claves primarias, pero que por alguna razón no fueron tomadas como tal.



Llave foránea

- Es una clave primaria en otra relación, estas representan las asociaciones entre las diferentes entidades, es decir, son claves que están siendo compartidas por dos tablas para formar una relación entre ellas.



Diagrama de Entidad-Relación

- Representación gráfica del modelo de la base de datos.
- Representa a las Entidades (tablas) y sus Relaciones.



Formas normales

- Es un proceso de descomposición sin pérdida, para lograr que nuestras bases de datos estén lo más óptimas posibles.
- Son 3 las principales.



Primera forma normal

Se asegura que la tabla sea representación fiel de una relación, libre de grupos repetitivos.

1. No hay orden arriba-a-abajo en las filas (log).
2. No hay orden izquierda-derecha en las columnas (días de la semana).
3. No hay filas duplicadas (incluir llave artificial).
4. Cada intersección de fila-columna contiene un valor del dominio aplicable y nada más (no nulos). (Integridad y Consistencia)
5. Cada campo ya no se puede subdividir otros.
6. Los campos no-clave deben identificarse por la clave (Dependencia Funcional)



Grupos repetitivos

Más de un campo para un mismo tipo de dato:

- Teléfono 1, Telefono 2, etc.
- Lunes, Martes, Miércoles, ... Domingo
- Habilidad 1, Habilidad 2, Habilidad 3.
- MontoEfectivo, MontoPuntos, MontoTDC, MontoTDB

En esos casos es necesario crear una nueva tabla (con los grupos repetitivos) en una relación 1:n



Ejemplo de casos donde no se cumple la 1N

- Una tabla sin llave primaria.
- Una vista cuya definición exige que los resultados sean retornados en un orden particular, de modo que el orden de la fila sea un aspecto intrínseco y significativo de la vista.
- Una tabla con por lo menos un atributo que pueda ser nulo.



Segunda forma normal

Debe cumplir con 1FN, y

- Cualquier campo no-llave depende de toda la clave primaria (y de las candidatas) en vez de solo de una parte de ella.
- Cuando una tabla en 1FN no tiene ninguna clave candidata compuesta (claves candidatas consistiendo en más de un atributo), la tabla está automáticamente en 2FN.



Ejemplo de caso donde no se cumple la 2N

Considerar la tabla de materias cursadas por alumno siguiente:

- Matricula
- Clave de materia
- Salón
- Nombre de alumno
- Calificación

Salón depende sólo de Materia y Nombre de Alumno depende sólo de Matrícula



¿Qué hacer?

Alumnos:

- Matrícula
- Nombre de alumno

Materias cursadas con

- Clave de Materia
- Aula

Tabla intermedia (transitiva) con

- Matrícula
- Clave de Materia
- Calificación final



Otro ejemplo

Considerar la tabla de recolecciones siguiente:

- Día
- Recolector
- Clima (soleado, lluvioso)
- Bicho recolectado
- Hora de recolección

¿Qué hacer?



Tercera forma normal

Debe cumplir con 2FN, y

- No existe ninguna dependencia funcional transitiva entre los atributos que no son clave, es decir: una dependencia funcional $X \rightarrow Y$ en un esquema de relación R es una dependencia transitiva si hay un conjunto de atributos Z que no es un subconjunto de alguna clave de R, donde se mantiene $X \rightarrow Z$ y $Z \rightarrow Y$.
- Cada atributo no-clave "debe proporcionar un hecho sobre la clave, de la clave entera, y nada más excepto que de la clave".



Ejemplo de caso donde no se cumple la 3N

- Considerar la tabla de mejores promedios anuales por carrera:
 - Año
 - Carrera
 - Nombre de alumno con mejor promedio
 - Fecha de nacimiento del alumno con mejor promedio.
- Fecha de nacimiento depende del alumno, no de ningún campo de la llave primaria (Año-Carrera), por lo que no está en 3FN.



¿Qué hacer?

Tabla de Mejores promedios:

- Año
- Carrera
- Nombre de alumno con mejor promedio (FK)

Tabla de Alumnos

- Nombre de alumno
- Fecha de nacimiento



Otro ejemplo

Considerar la tabla de Ventas:

- Ticket
- CveCliente (FK)
- CveVendedor (FK)
- Monto
- Cliente-Frecuente (Si/No).

¿Qué campo o campos no cumplen con la 3N?



3. Creación de bases de datos y tablas (DDL)



Procesamiento de consultas

Análisis sintáctico y semántico (se parte en unidades lógicas)

Análisis del mejor camino (plan de ejecución)

Ejecuta *query* sobre la base de datos

Se transforman los datos al formato final requerido

El lenguaje de consulta estructurado (Structured Query Language) SQL

Lenguaje con el cual el cliente opera/consulta/actualiza/administra la base de datos. Consiste en 4 sublenguajes:

- Data Definition Language o DDL (CREATE, ALTER, DROP)
- Data Manipulation Language o DML (INSERT, UPDATE, DELETE)
- Data Control Language o DCL (GRANT, REVOKE)
- Transaction Control o TCL (COMMIT, ROLLBACK)



Comandos DDL para bases de datos

- CREATE DATABASE nombredebasedatos;
- DROP DATABASE nombredebasedatos;
- USE nombrededabsedatos;



Creación de tablas

```
CREATE TABLE nombredetabla  
(  
    definición de campo 1,  
    definición de campo 2,  
    ...  
    definición de campo n  
) ;
```

Definición de campo simplificada

nombredelcampo tipodedato



Ejemplo

```
CREATE TABLE peliculas  
(  
    titulo          varchar (100),  
    director        varchar (100),  
    genero          varchar (100),  
    ano             smallint,  
    clasificacion  varchar (20),  
    protagonistas   varchar (300)  
)
```

Tipos de datos

- Números (enteros, punto fijo, punto flotante)
- Caracteres
- Fecha
- Otros



Enteros

Tipo de dato	Con signo	Sin signo	Tamaño
TINYINT	-128 a +127	0-255	1 byte
SMALLINT	-32768 A +32767	0-65535	2 bytes
MEDIUMINT	-2^{23} a $2^{23}-1$		3 bytes
INT / INTEGER	-2^{31} a $2^{31}-1$		4 bytes
BIGINT	-2^{63} a $2^{63}-1$		8 bytes
BIT			1 byte para 8 bits



DECIMAL (punto fijo)

- Se controla el número de enteros y de decimales (precisión)
Decimal (precisión, escala)

donde:

precisión → número de dígitos enteros y decimales

escala → número de decimales



Reales (punto flotante)

Tipo	Rango
FLOAT	-3.4E+38 a -1.17-38 1.17-38 a 3.4E+38.
DOUBLE	-1.79E+308 a -2.22E-308 2.22E-308 a 1.79E+308

Ejemplo

```
CREATE TABLE PruebaNumeros  
(  
    CampoDecimal decimal(5,2) ,  
    CampoInt int ,  
    CampoDouble double  
) ;
```

```
INSERT INTO PruebaNumeros VALUES (123.4, 123.4, 123.4);  
SELECT * FROM PruebaNumeros;
```

```
DROP TABLE PruebaNumeros;
```

Caracteres

Tipo	Tamaño máximo
CHAR (n)	n=255, si se omite n=1
VARCHAR (n)	2^{16} caracteres
TINYTEXT, TEXT, MEDIUMTEXT, LONGTEXT	

Ejemplo

```
CREATE TABLE PruebaCaracteres  
(  
    CampoChar CHAR(20),  
    CampoVarChar VARCHAR(20),  
    CampoText TEXT  
) ;
```

```
INSERT INTO PruebaCaracteres VALUES ("UNIVERSIDAD",  
"NACIONAL", "AUTÓNOMA");
```

```
SELECT * FROM PruebaCaracteres;
```

```
DROP TABLE PruebaCaracteres;
```



Fechas

Tipo	Ejemplo
DATE	'2022-03-04', '2023-2-3'
TIME	'12:03:04', '9:6:3'
DATETIME	'2022-03-04 16:10:09'

Ejemplo

```
CREATE TABLE PruebaFechas
(
    CampoDate date,
    CampoTime time,
    CampoDatetime datetime);
##  
INSERT INTO PruebaFechas VALUES (now(), now(), now());  
SELECT * FROM PruebaFechas;  
  
DROP TABLE PruebaFechas;
```

Integridad de datos

- NULL / NOT NULL
- DEFAULT
- CHECK
- CONSTRAINT
- PRIMARY KEY
- FOREIGN KEY
- AUTO_INCREMENT

Creación de datos por omisión (DEFAULT)

- Opción DEFAULT al definir un campo

nombredecampo tipodedatos DEFAULT valor

- Ejemplo

```
sexo char(1) DEFAULT 'M'
```



Ejemplo

```
DROP TABLE peliculas;
CREATE TABLE peliculas
(
    titulo varchar(100),
    director varchar (100),
    genero varchar(100),
    ano smallint DEFAULT 2000,
    clasificacion varchar(20) DEFAULT 'A',
    protagonistas varchar (300)
);
INSERT INTO peliculas (titulo) VALUES ('Titanic');
SELECT * FROM peliculas;
```



Ejemplo

```
DROP TABLE peliculas;
CREATE TABLE peliculas
(
    titulo varchar(100) NOT NULL,
    director varchar (100) NULL,
    genero varchar(100) NULL,
    anoEstreno smallint DEFAULT 2000 NULL CHECK (anoEstreno > 1900),
    clasificacion varchar(20) NULL DEFAULT 'A',
    protagonistas varchar (300) NULL DEFAULT ''
) ;
```

Ejemplo, continuación.

```
INSERT INTO peliculas (titulo, anoEstreno) VALUES  
( 'Titanic' , 1880 );
```

```
INSERT INTO peliculas (titulo, anoEstreno) VALUES  
( 'Vaselina' , 1980 );
```

```
SELECT * FROM peliculas;
```

¿Qué sucede?



¿Y si se quiere hacer un CHECK con varios campos?

...

```
anoGrabacion smallint DEFAULT 2000 NULL CHECK  
(anoGrabacion <= anoEstreno),  
anoEdicion    smallint DEFAULT 2000 NULL CHECK  
(anoGrabacion >= anoGrabacion AND anoGrabacion <=  
anoEstreno),
```

...

Probar



CONSTRAINT para hacer CHECK de varios campos

```
ALTER TABLE nombredetabla
```

```
ADD CONSTRAINT check_nombredelconstraint CHECK  
(condición)
```



Ejemplo

```
DROP TABLE peliculas;
CREATE TABLE peliculas
(
    titulo varchar(100) NOT NULL,
    director varchar (100) NULL,
    genero varchar(100) NULL,
    anoEstreno smallint DEFAULT 2000 NULL CHECK (anoEstreno >
1900),
    anoGrabacion smallint NULL,
    anoEdicion smallint NULL,
    clasificacion varchar(20) NULL DEFAULT "A",
    protagonistas varchar (200) NULL DEFAULT ""
);
```

Ejemplo, constraints

```
ALTER TABLE peliculas ADD CONSTRAINT  
check_validaAnoGrabacion CHECK (anoGrabacion <= anoEstreno);
```

```
ALTER TABLE peliculas ADD CONSTRAINT check_validaAnoEdicion  
CHECK (anoEdicion >= anoGrabacion AND anoEdicion <=  
anoEstreno);
```



Ejemplo, registros nuevos

Probar

```
INSERT INTO peliculas (titulo, anoEstreno, anoEdicion,  
anoGrabacion) VALUES ('Titanic', 1980, 1979, 1978);
```

```
INSERT INTO peliculas (titulo, anoEstreno, anoEdicion,  
anoGrabacion) VALUES ('Rocky I', 1980, 1979, 2000);
```

```
INSERT INTO peliculas (titulo, anoEstreno, anoEdicion,  
anoGrabacion) VALUES ('Vaselina', 1980, 1981, 1978);
```

¿Qué mensajes se obtuvieron?



CONSTRAINT CHECK en el CREATE TABLE

```
DROP TABLE peliculas;
CREATE TABLE peliculas
(
    titulo varchar(100) NOT NULL,
    director varchar (100) NULL,
    genero varchar(100) NULL,
    anoEstreno smallint DEFAULT 2000 NULL CHECK (anoEstreno>1900),
    anoGrabacion smallint NULL,
    anoEdicion    smallint NULL,
    clasificacion varchar(20) NULL DEFAULT 'A',
    protagonistas varchar (300) NULL DEFAULT '',
    CONSTRAINT check_validaAnoGrabacion CHECK (anoGrabacion <= anoEstreno),
    CONSTRAINT check_validaAnoEdicion CHECK (anoEdicion >= anoGrabacion AND
    anoEdicion <= anoEstreno) )
```



UNIQUE

nombredecampo tipodedatos UNIQUE
(al definir un campo)

O

CONSTRAINT unique_nombredelconstraint UNIQUE
(nombredecampo)
(en la sección de CONSTRAINT's del CREATE TABLE)



Tipos de llaves

- Candidatas
- Principal
- Secundarias
- Foráneas



PRIMARY KEY

```
CREATE TABLE tabla (
    definición de campo PRIMARY KEY, --Un solo campo
    definición de demás campos
)
```

o

```
CREATE TABLE tabla (
    definición de campos,
    CONSTRAINT pk_tabla PRIMARY KEY (campos)
)
```



FOREIGN KEY

```
CREATE TABLE tabla (
    campo tipo REFERENCES tablaPadre(llavePrimariaEnPadre), --Un solo campo
    definición de demás campos,
)
```

o

```
CREATE TABLE tabla (
    definición de campos,
    CONSTRAINT fk_padre FOREIGN KEY (campos) REFERENCES tablaPadre (llavePrimariaEnPadre)
)
```



Ejemplo de definición de FK

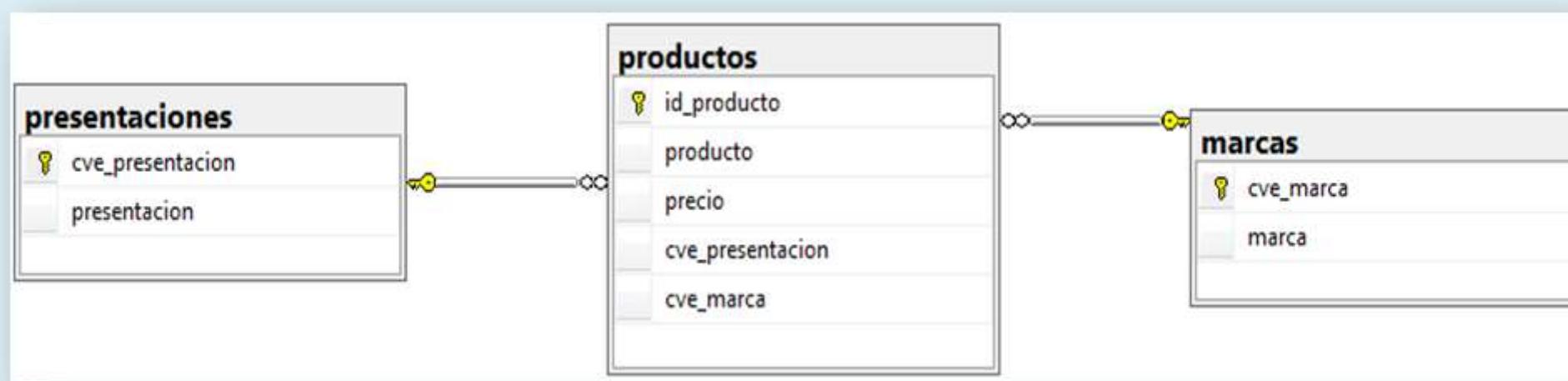


```
CREATE TABLE Usuarios (  
    idUsuario int PRIMARY KEY,  
    usuario varchar(100),  
    status char(1)  
);
```

```
CREATE TABLE Renta (  
    idRenta int PRIMARY KEY,  
    idUsuario int,  
    idPelicula int,  
    fechaPrestamo datetime,  
    diasPrestamo tinyint,  
    precio smallmoney,  
  
    CONSTRAINT fk_UR FOREIGN KEY (idUsuario)  
        REFERENCES Usuarios (idUsuario),  
  
    CONSTRAINT fk_PR FOREIGN KEY  
        (idPelicula) REFERENCES Peliculas  
        (idPelicula)  
);
```

```
CREATE TABLE Peliculas (  
    idPelicula int PRIMARY KEY,  
    pelicula varchar(100),  
    sinopsis varchar(200)  
);
```

Ejemplo

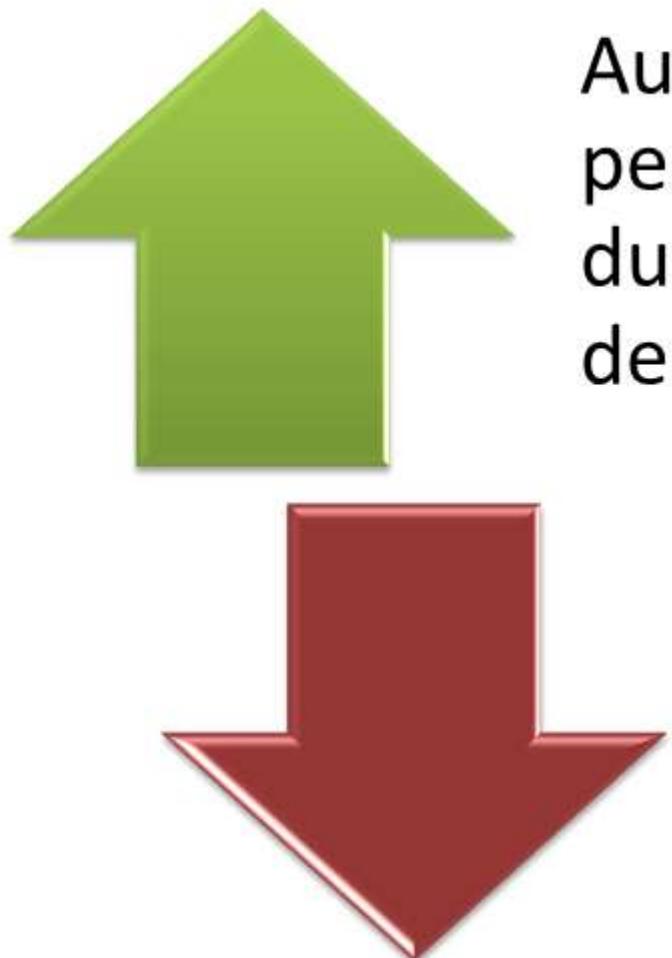


Índices

- Estructura que permite acceder a la información de las tablas de manera más rápida.
- Se requiere espacio para su almacenamiento y tiempo para su mantenimiento.
- Un índice siempre está asociado a una tabla.
- Siempre estará definido por uno o más *campos llave* que son la forma en la que se organiza la información.



Ventajas y desventajas de los índices



Aumenta
performance
durante la extracción
de datos

Disminuye el
rendimiento durante
la actualización de la
tabla

Creación de índices

- CREATE INDEX nombredeindice ON tabla(campos);

4. Uso de bases de datos MySQL

1. Introducción al Lenguaje estructurado de consultas (SQL)
2. Sentencias del lenguaje SQL (DDL, DML, DCL)
3. Inserción, actualización y eliminación de la información
4. Selección de la información



Agregar registros (INSERT)

INSERT INTO tabla VALUES (valores para todos los campos);

o

INSERT INTO tabla (campos) VALUES (valores);

Ejemplo de INSERT

```
INSERT INTO marcas VALUES (1, 'Bayer');
```

```
INSERT INTO presentaciones VALUES (10, 'Tableta');
```

```
INSERT INTO productos (id_producto, producto, precio,  
cve_presentacion, cve_marca ) VALUES (123,  
'Aspirinas', 30, 10, 1);
```

```
INSERT INTO productos (id_producto, precio,  
cve_presentacion, cve_marca ) VALUES (23, 80, 10, 1);
```

```
SELECT * FROM productos;
```



Eliminar registros (DELETE)

```
TRUNCATE TABLE tabla;
```

```
DELETE FROM tabla;
```

```
DELETE FROM tabla WHERE condición;
```



Ejemplo

```
TRUNCATE TABLE marcas;          --Vacía tabla
```

```
TRUNCATE TABLE productos;      --Vacía tabla
```

```
DELETE FROM productos;         --Vacía tabla
```

```
DELETE FROM productos WHERE precio > 60;
```

Modificar registros (UPDATE)

UPDATE tabla SET campo1=valor1, campo2=valor2,...,
campo_n=valor_n

UPDATE tabla SET campo1=valor1, campo2=valor2,...,
campo_n=valor_n WHERE condición



Ejemplo

```
UPDATE marcas SET marca = 'J & J';
```

```
UPDATE productos SET precio = precio*1.1;
```

```
UPDATE productos SET precio = precio*0.9 WHERE  
cve_marca=1;
```

Extracción de datos (SELECT)

```
SELECT * FROM tabla;  
SELECT campos FROM tabla;  
SELECT alias.campos FROM tabla alias;  
SELECT campo AS nuevonombre ... FROM ...  
SELECT DISTINCT campo FROM tabla  
SELECT ... FROM tabla ORDER BY ...;  
SELECT * FROM tabla WHERE condición;
```

Operadores de comparación

=	Igualdad
<>	Diferente
>	Mayor que
>=	Mayor o igual que
<	Menor
<=	Menor o igual que

Operadores lógicos

AND	Y lógico
OR	O lógico
NOT	Negación



Operadores especiales

- **IN**

...WHERE campo IN (valor1, valor2, ... , valorn)

- **LIKE**

...WHERE string LIKE 'patrón'

 % Sustituye 0-n caracteres

 _ Sustituye UN caracter

 [] Permite definir un rango de caracteres [b-g]

[a,e,i,o,u]

 [^] Es la negación [^a] que No sea una a

- **BETWEEN**

...WHERE campo BETWEEN inicio AND final

- **IS NULL / IS NOT NULL**

...WHERE campo IS NULL

¿ IN (5,6,7,8) o BETWEEN 5 AND 8 ?



Agrupamiento de información

SELECT **campos** y funciones
de agregación
FROM tabla
WHERE condición
GROUP BY **campo1, campo2**
ORDER BY **campo1, campo2**

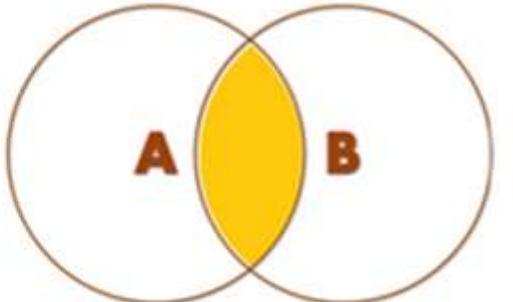
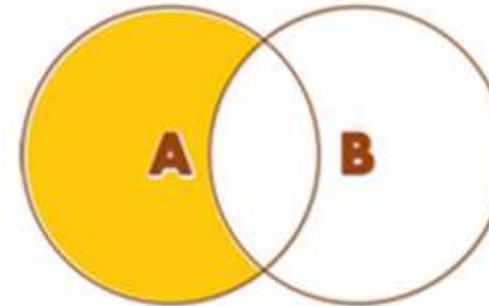
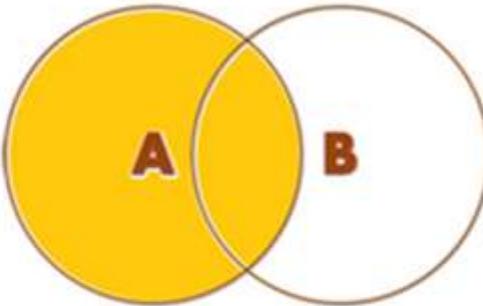
Funciones de agregación

- **MAX(campo)**
- **MIN(campo)**
- **SUM(campo numérico)**
- **AVG(campo numérico)**
- **STDEV(campo numérico)**
- **STDEVP(campo numérico)**
- **VAR(campo numérico)**
- **VARP(campo numérico)**
- **COUNT(campo ó *)**

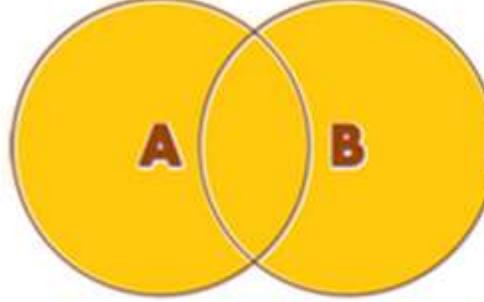


Joins

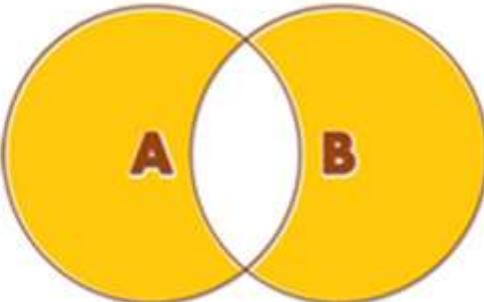
Left joins



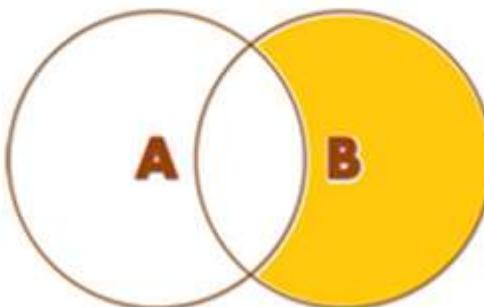
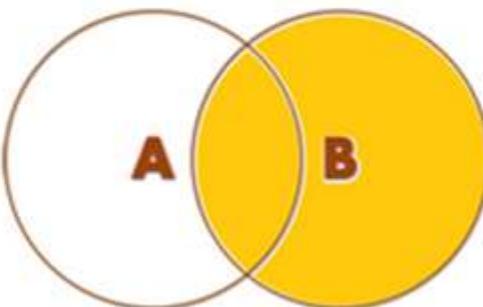
Inner join



Full Outer joins



Right joins



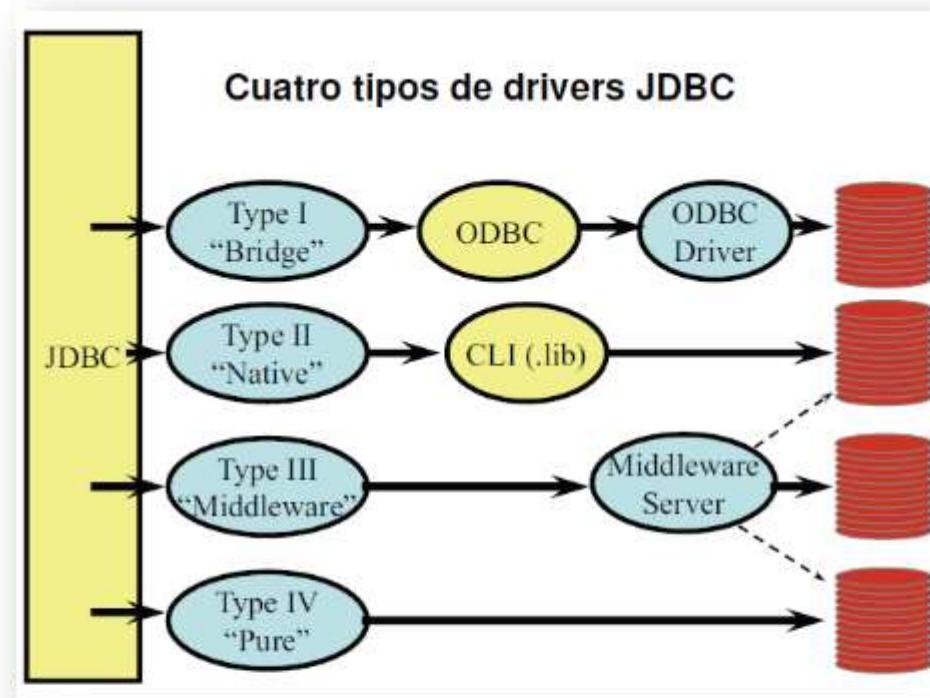
5. Integración de bases de datos con Java

- 1. API JDBC**
- 2. Creación de un programa con JDBC**



JDBC y tipos de controladores

- Interfaz que permite ejecutar instrucciones SQL en una bbdd.
- Puede ser “puente” (jdbc-odbc), Api nativa, Middleware o Net protocol.

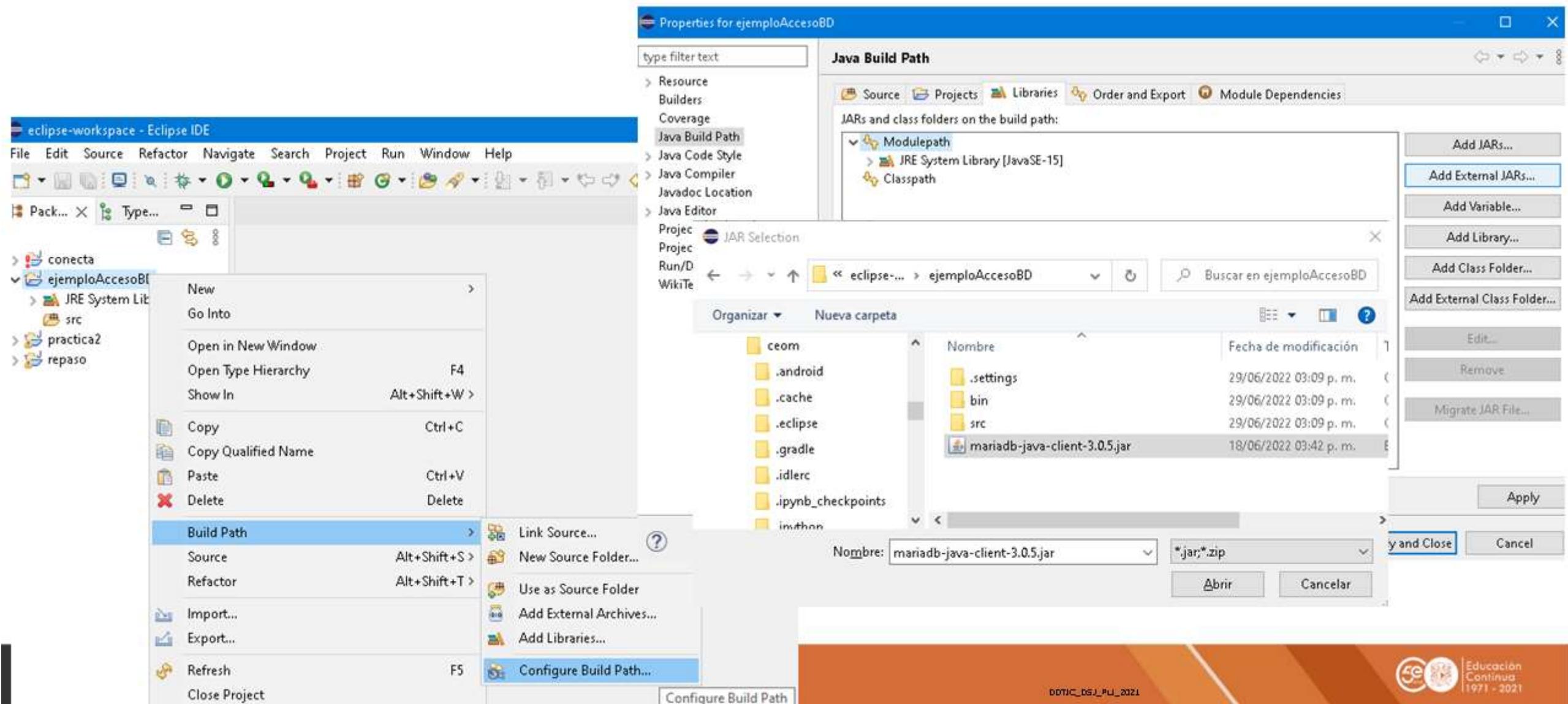


Procedimiento general de conexión con JDBC

1. Configurar JDBC específico de la base de datos en proyecto.
2. Especificar cadena de conexión
3. Conectarse por medio de Connection
4. Ejecutar instrucciones SQL con Statement
5. Consultar respuesta con ResultSet
6. Cerrar objetos (ResultSet, Statement, Connection)



Configurar JDBC en proyecto (Eclipse)



```
J ejempleado01.java X
2 import java.sql.Connection;
3 import java.sql.DriverManager;
4 import java.sql.Statement;
5 public class ejempleado01 {
6     public static void main(String[] args) {
7         String cadenaConexion = "jdbc:mariadb://localhost/ejercicio2";
8         // "jdbc:motor://servidor/base de datos";
9         try{
10             Connection oConn = DriverManager.getConnection(cadenaConexion, "root", "MaPassw");
11             // cadena de conexión, usuario, contraseña
12             Statement oStatem = oConn.createStatement();
13             String sql = "INSERT INTO tabla VALUES (1,'texto','2023-01-30')";
14             oStatem.executeUpdate(sql); //Ejecución del comando
15             int registros = oStatem.getUpdateCount(); //¿Cuántos registros se afectaron?
16             // Se cierran todos los objetos
17             oStatem.close();
18             oConn.close();
19             System.out.println ("Operaciones realizadas");
20         } catch (Exception e) {
21             } //Fin del catch()
22         } //Fin del main()
23     } //Fin de la clase
```

ejercicio2.tabla: 1 rows total (approximately)		
numero	texto	fecha
1	texto	2023-01-30

Problems Javadoc Declaration Console X

<terminated> ejempleado01 [Java Application] C:\Program Files\Java\jdk-15.0.1\bin\javaw.exe (29 jun. 2022 15:33:25 – 15:33:26) [pid: 12296]

Operaciones realizadas

Procesamiento de resultado

- El resultado de un query (en un objeto Resultset) es similar a un arreglo bidimensional.
- Se recorre con métodos como `next()`
- Regularmente se recorre dentro de un `while`



Contacto

Lic. Carlos Eligio Ortiz Maldonado

carloseligio@ortizm.com

