



DIPLOMADO
**Desarrollo de sistemas con
tecnología Java**

Módulo 2

Manejo de bases de datos con Java

Carlos Eligio Ortiz

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 (DML).
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
- IntelliJ IDEA

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.**
 - **Consistencia.**
 - **Integridad.**
 - **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

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 tengan el mejor diseño posible.
- 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:

- Matrícula
- 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

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

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 nombredebasedatos;

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

nombreredecampo 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

nombreredcampo tipodedatos UNIQUE
(al definir un campo)

O

CONSTRAINT unique_nombradelconstraint UNIQUE
(nombreredcampo)
(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)
)
```

AUTO INCREMENT

- Permite hacer a un campo numérico un valor consecutivo.
- Solo puede haber uno por tabla.
- Debe ser definida como llave (primaria o secundaria).
- En un INSERT se puede asignar al campo auto_increment, y el siguiente registro tomará ese valor +1.

```
CREATE TABLE tabla (
    definición de campo PRIMARY KEY AUTO_INCREMENT,
    definición de demás campos
)
```

```
ALTER TABLE tabla AUTO_INCREMENT = 9;      # Valor del 9 en adelante
```

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)
);
```

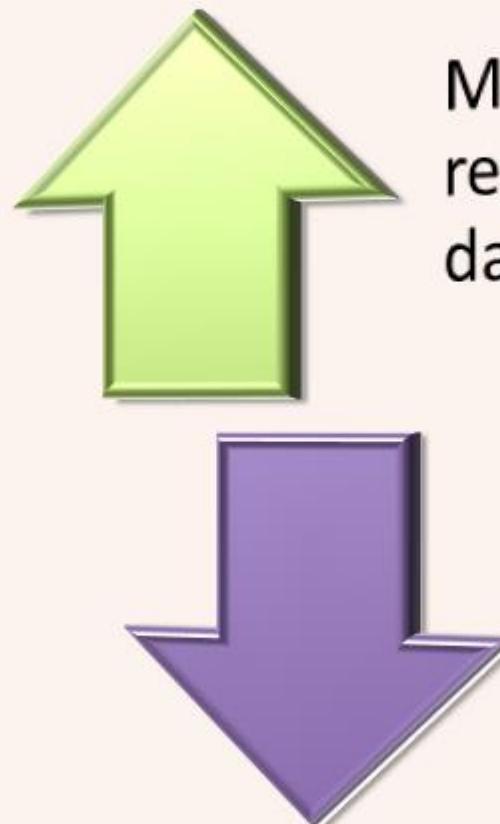
```
CREATE TABLE Renta (
    idRenta int PRIMARY KEY,
    idUsuario int,
    idPelicula int,
    fechaPrestamo datetime,
    diasPrestamo tinyint,
    precio smallmoney,
    CONSTRAINT fk_UR FOREIGN KEY (idUsuario)
    REFERENCES Usuarios (idUsuario),
    CONSTRAINT fk_PR FOREIGN KEY
    (idPelicula)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

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

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



Mejora el tiempo de respuesta al extraer datos

Disminuye el rendimiento durante la actualización de la tabla

Creación de índices

- CREATE INDEX nombreindice ON tabla(campos);

4. Uso de bases de datos MariaDB

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,...,  
campoN=valorN
```

```
UPDATE tabla SET campo1=valor1, campo2=valor2,...,  
campoN=valorN 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)**
- **STDDEV(campo numérico)**
- **STDDEVP(campo numérico)**
- **VAR(campo numérico)**
- **VARP(campo numérico)**
- **COUNT(campo ó *)**

Operaciones de conjunto

- UNION, UNION ALL
- INTERSECT
- EXCEPT

Primer SELECT
OPERACIÓN
Segundo SELECT;

Funciones

De conversión

- CAST (expresión AS nuevoTipo)
- CONVERT (expresión USING nuevoTipo)

Numéricas

- ABS ()
- FLOOR () y CEILING ()
- LOG () y LOG10 ()
- PI ()
- POWER (x, y)
- ROUND (número, posicionesDecimales)
- RAND ()
- SQRT ()

Funciones de cadena

- ASCII (c) y CHAR (n)
- INSTR (aBuscar, dondeBuscar)
- CONCAT (c1, c2, ..., cn)
- LEFT (c, n) y RIGHT (c, n)
- LTRIM() y RTRIM()
- CHAR_LENGTH(s) # Longitud en caracteres
- LOWER() y UPPER() / LCASE() Y UCASE()
- REPLACE (c, qué, conQué)
- REPEAT (c, n)
- REVERSE()
- SPACE (n) = REPLICATE (' ', n)
- STUFF (c, Inicio, Cuantos, cInsertar)
- SUBSTRING (c, inicio, cuantos)

Funciones de fecha

| unit Value | Expected expr Format |
|--------------------|--|
| SECOND | SECONDS |
| MINUTE | MINUTES |
| HOUR | HOURS |
| DAY | DAYS |
| WEEK | WEEKS |
| MONTH | MONTHS |
| QUARTER | QUARTERS |
| YEAR | YEARS |
| SECOND_MICROSECOND | 'SECONDS.MICROSECONDS' |
| MINUTE_MICROSECOND | 'MINUTES:SECONDS.MICROSECONDS' |
| MINUTE_SECOND | 'MINUTES:SECONDS' |
| HOUR_MICROSECOND | 'HOURS:MINUTES:SECONDS.MICROSECONDS' |
| HOUR_SECOND | 'HOURS:MINUTES:SECONDS' |
| HOUR_MINUTE | 'HOURS:MINUTES' |
| DAY_MICROSECOND | 'DAYS HOURS:MINUTES:SECONDS.MICROSECONDS' |
| DAY_SECOND | 'DAYS HOURS:MINUTES:SECONDS' |
| DAY_MINUTE | 'DAYS HOURS:MINUTES' |
| DAY_HOUR | 'DAYS HOURS' |
| YEAR_MONTH | 'YEARS-MONTHS' |

- **DATE_ADD(f, INTERVAL n part)**)
- **DATEDIFF(fFinal, fInicio) # dif. en días**
- **TIMEDIFF(hInicio, hFinal) # regresa TIME**
- **TIMESTAMPDIFF(unidad, dtInicio, dtFinal) # dif. en unidades**
- **DATE_FORMAT(fecha, formato)**
- **DAY(), MONTH(), YEAR()**
- **NOW()**

```

1 SELECT YEAR(NOW()), MONTH (NOW()), DAY (NOW());
2 SELECT DATEDIFF(NOW(), DATE_ADD(now(), INTERVAL 1 DAY));
3 SELECT NOW(), DATE_ADD(now(), INTERVAL 1 DAY);

```

| Specifier | Description |
|-----------|--|
| %a | Abbreviated weekday name (Sun..Sat) |
| %b | Abbreviated month name (Jan..Dec) |
| %c | Month, numeric (0..12) |
| %D | Day of the month with English suffix (0th, 1st, 2nd, 3rd, ...) |
| %d | Day of the month, numeric (00..31) |
| %e | Day of the month, numeric (0..31) |
| %f | Microseconds (000000..999999) |
| %H | Hour (00..23) |
| %h | Hour (01..12) |
| %I | Hour (01..12) |
| %i | Minutes, numeric (00..59) |
| %j | Day of year (001..366) |
| %k | Hour (0..23) |
| %l | Hour (1..12) |
| %M | Month name (January..December) |
| %m | Month, numeric (00..12) |
| %p | AM or PM |
| %r | Time, 12-hour (hh:mm:ss followed by AM or PM) |
| %s | Seconds (00..59) |

Vistas

CREATE VIEW **nombre** AS SELECT ...

...

SELECT * FROM **nombre**

```
# Creación de vista
CREATE VIEW personajesF AS
    SELECT nombre, profesion FROM friends WHERE sexo='F';

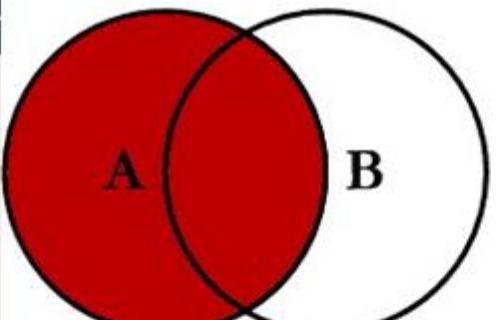
# Modificación de vista
ALTER VIEW personajesF AS
    SELECT nombre, profesion FROM friends WHERE sexo='M';

# Eliminación de vista
DROP VIEW personajesF;

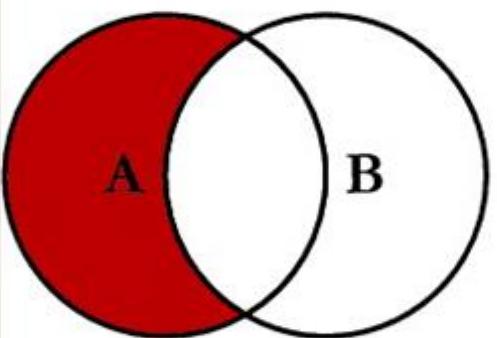
DELETE FROM Friends WHERE nombre= 'Phoebe Buffay';

# Uso de vista
SELECT nombre FROM personajesF ORDER BY 1;
```

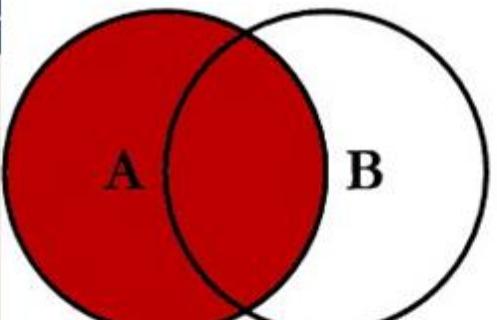
Joins



```
SELECT <select_list>
FROM TableA A
LEFT JOIN TableB B
ON A.Key = B.Key
```

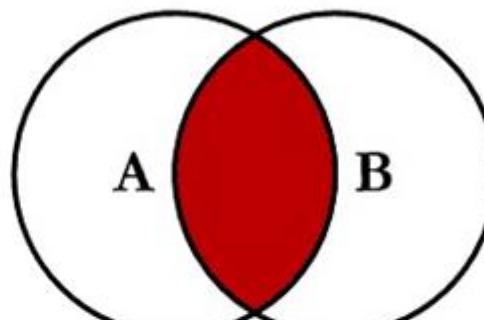


```
SELECT <select_list>
FROM TableA A
LEFT JOIN TableB B
ON A.Key = B.Key
WHERE B.Key IS NULL
```

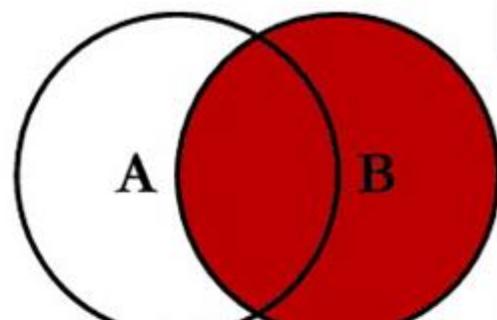


```
SELECT <select_list>
FROM TableA A
LEFT JOIN TableB B
ON A.Key = B.Key
WHERE B.Key IS NULL
```

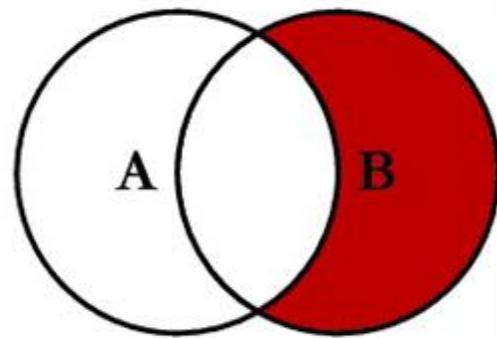
SQL JOINS



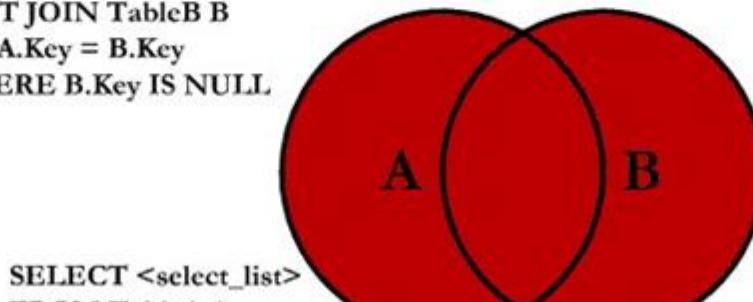
```
SELECT <select_list>
FROM TableA A
INNER JOIN TableB B
ON A.Key = B.Key
```



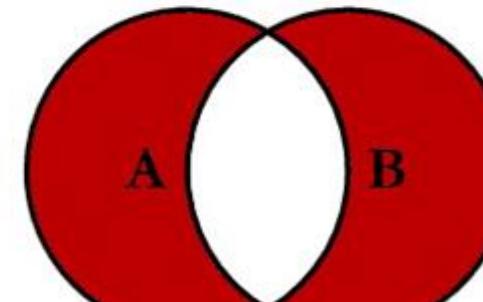
```
SELECT <select_list>
FROM TableA A
RIGHT JOIN TableB B
ON A.Key = B.Key
```



```
SELECT <select_list>
FROM TableA A
RIGHT JOIN TableB B
ON A.Key = B.Key
WHERE A.Key IS NULL
```



```
SELECT <select_list>
FROM TableA A
FULL OUTER JOIN TableB B
ON A.Key = B.Key
```



```
SELECT <select_list>
FROM TableA A
FULL OUTER JOIN TableB B
ON A.Key = B.Key
WHERE A.Key IS NULL
OR B.Key IS NULL
```

Funciones

```
DELIMITER //  
CREATE FUNCTION nombre (parámetros de entrada) RETURNS tipo  
BEGIN  
    ...  
    RETURN (valor);  
END//  
DELIMITER ;
```

```
1 DROP FUNCTION IF EXISTS cuadrado;  
2 DELIMITER //  
3 CREATE FUNCTION cuadrado (entrada DOUBLE) RETURNS DOUBLE  
4 BEGIN  
5     DECLARE resultado DOUBLE;  
6     SET resultado = entrada * entrada;  
7     RETURN (resultado);  
8 END//  
9 DELIMITER ;  
10  
11 SELECT cuadrado (2), cuadrado (3); cuadrado (2) cuadrado (3)
```

Procedimiento almacenado

```
DELIMITER //  
CREATE PROCEDURE procedimiento (parámetros de entrada)
```

```
BEGIN
```

```
...
```

```
END//
```

```
DELIMITER ;
```

```
DROP PROCEDURE IF EXISTS listaEntidades;  
DELIMITER //  
CREATE PROCEDURE listaEntidades (IN limite INT)  
BEGIN  
    SELECT nombre, clientes  
    FROM estados  
    WHERE clientes <= limite  
    ORDER BY clientes ;  
|  
END//  
DELIMITER ;  
  
CALL listaEntidades(500000);
```

| nombre | clientes |
|---------------------|----------|
| Campeche | 8479 |
| Chiapas | 9986 |
| Colima | 10334 |
| Nayarit | 10826 |
| Tlaxcala | 17197 |
| Quintana Roo | 20095 |
| Aguascalientes | 23051 |
| Baja California Sur | 26357 |
| Morelos | 26447 |
| Zacatecas | 27424 |
| Durango | 30793 |
| Yucatán | 32584 |

5. Integración de bases de datos con Java

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)

Descarga de MariaDB

<https://mariadb.org/mariadb/all-releases/>

The screenshot shows the MariaDB All Releases page. On the left, there's a sidebar with a seal logo and a list of releases from 11.0 down to 10.3. The main content area has two sections: '10.11' and '10.10'. Each section contains a table with 'Name' and 'Release date' columns. The '10.11' section shows 10.11.3 (2023-05-10), 10.11.2 (2023-02-16), 10.11.1 (2022-11-17), and 10.11.0 (2022-09-26). The '10.10' section shows 10.10.4 (2023-05-10), 10.10.3 (2023-02-16), 10.10.2 (2022-11-17), and 10.10.1 (2022-09-26). To the right, there are filters for 'MariaDB Server Version' (set to 10.11.3), 'Operating System' (Windows), 'Architecture' (x86_64), and 'Package Type' (MSI Package). A large blue 'Download' button is at the bottom.

MariaDB Server – All releases

- 11.0
- 10.11
- 10.10
- 10.9
- 10.8
- 10.7
- 10.6
- 10.5
- 10.4
- 10.3

10.11

| Name | Release date |
|---------|--------------|
| 10.11.3 | 2023-05-10 |
| 10.11.2 | 2023-02-16 |
| 10.11.1 | 2022-11-17 |
| 10.11.0 | 2022-09-26 |

10.10

| Name | Release date |
|---------|--------------|
| 10.10.4 | 2023-05-10 |
| 10.10.3 | 2023-02-16 |
| 10.10.2 | 2022-11-17 |
| 10.10.1 | 2022-09-26 |

MariaDB Server Version

MariaDB Server 10.11.3

Display older releases:

Operating System

Windows

Architecture

x86_64

Package Type

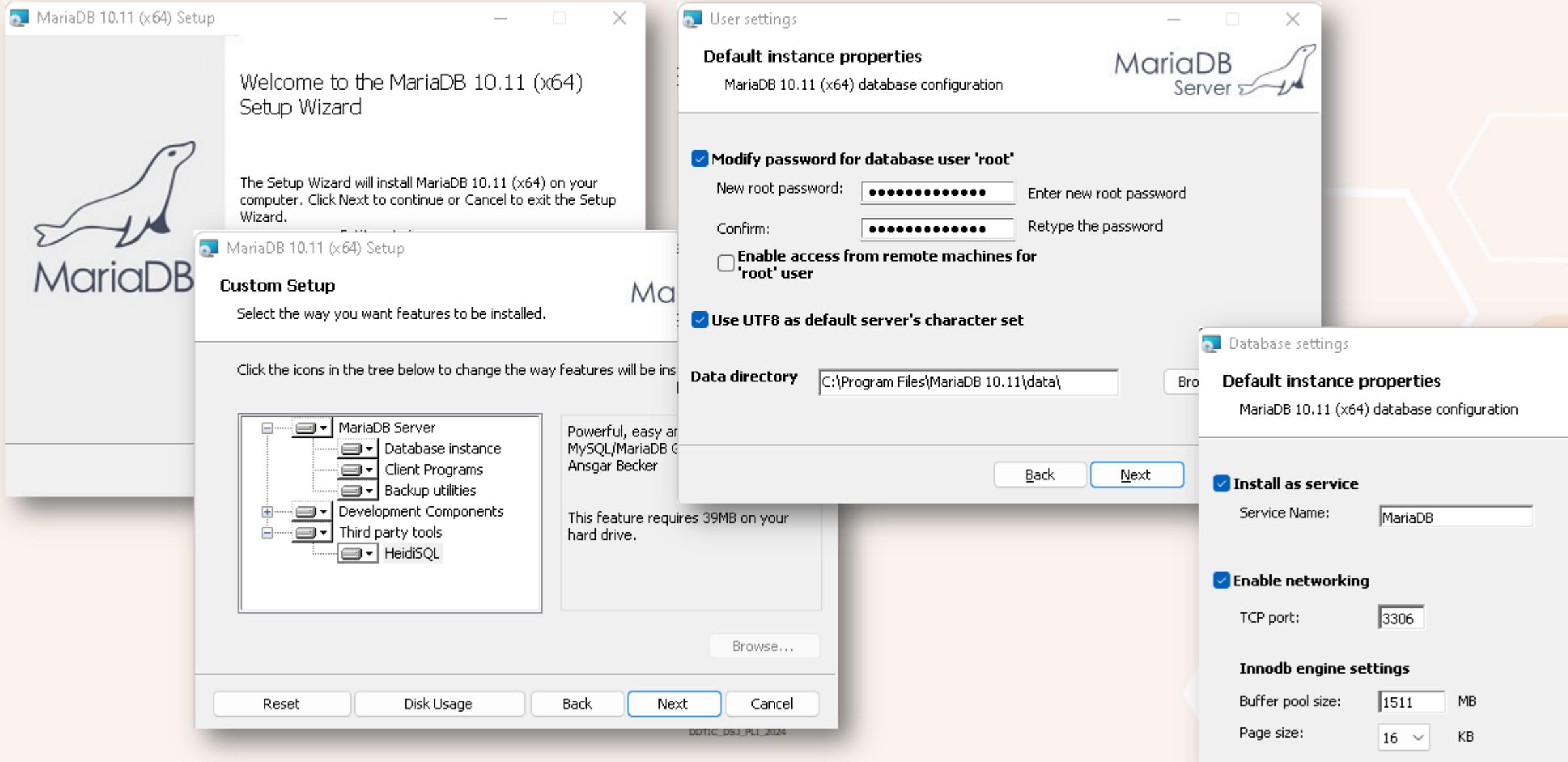
MSI Package

Download

Mirror

GigeNET - Chicago, IL

Instalación de MariaDB



Descarga del JDBC para MariaDB en

<https://mariadb.com/kb/en/about-mariadb-connector-j/>

<https://mariadb.org/connector-java/all-releases/>

The screenshot shows the MariaDB website's "Client Libraries" section. On the left, there's a sidebar with links like Home, Open Questions, MariaDB Server, MariaDB MaxScale, MariaDB ColumnStore, and Connectors. The main content area has a large image of a seal above the title "About MariaDB". Below the title, it says "Lightweight, advanced connectors for high-performance data access and data streaming." It describes the MariaDB Connector/J as a JDBC driver for Java applications. It also mentions that this version is for Java 8, Java 11, and Java 17. There are two dropdown menus: "Product" set to "Java 8+ connector" and "Version" set to "3.1.0-GA". At the bottom, there's a teal bar with the connector type ("Java 8+ connector"), its URL ("https://dlm.mariadb.com/2678616/Connectors/java/connector-java-3.1.0/mariadb-java-client-3.1.0.jar"), its size ("621.53 KB"), and a prominent "Download" button.

MariaDB

Knowledge Base » Server & Client Software » Client Libraries

Home

Open Questions

MariaDB Server

MariaDB MaxScale

MariaDB ColumnStore

Connectors

About MariaDB

MariaDB Connector/J is used to standard JDBC API. The library

Lightweight, advanced connectors for high-performance data access and data streaming.

MariaDB Connector/J is a lightweight JDBC driver (Type 4) for building Java applications on top of MariaDB, complete with built-in connection pooling and encrypted connections via TLS/SSL. MariaDB Connector/J is LGPL.

This version of MariaDB Connector/J is for Java 8, Java 11, and Java 17.

Release notes

Show All Files

Product: Java 8+ connector

Version: 3.1.0-GA

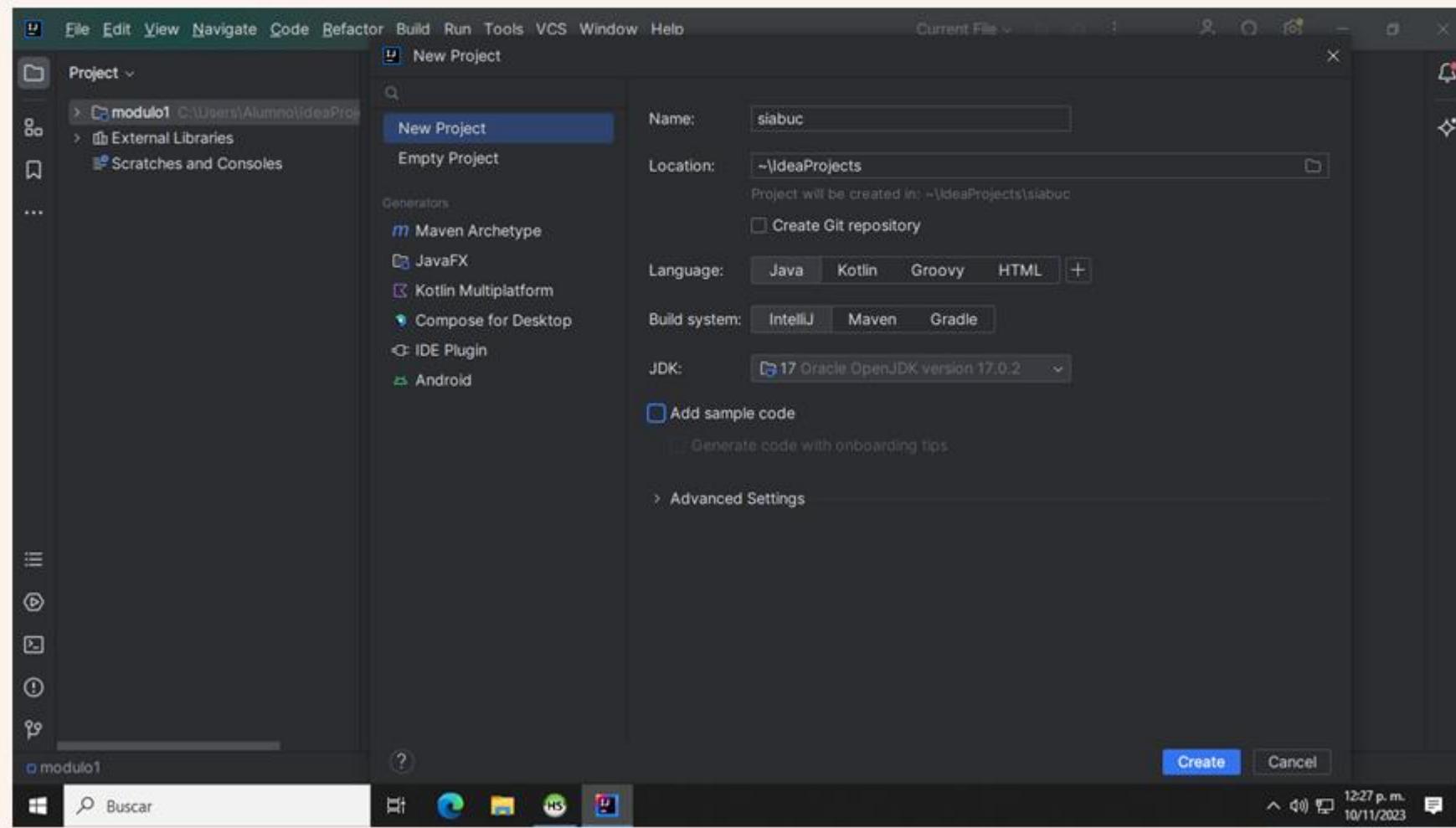
Java 8+ connector

<https://dlm.mariadb.com/2678616/Connectors/java/connector-java-3.1.0/mariadb-java-client-3.1.0.jar>

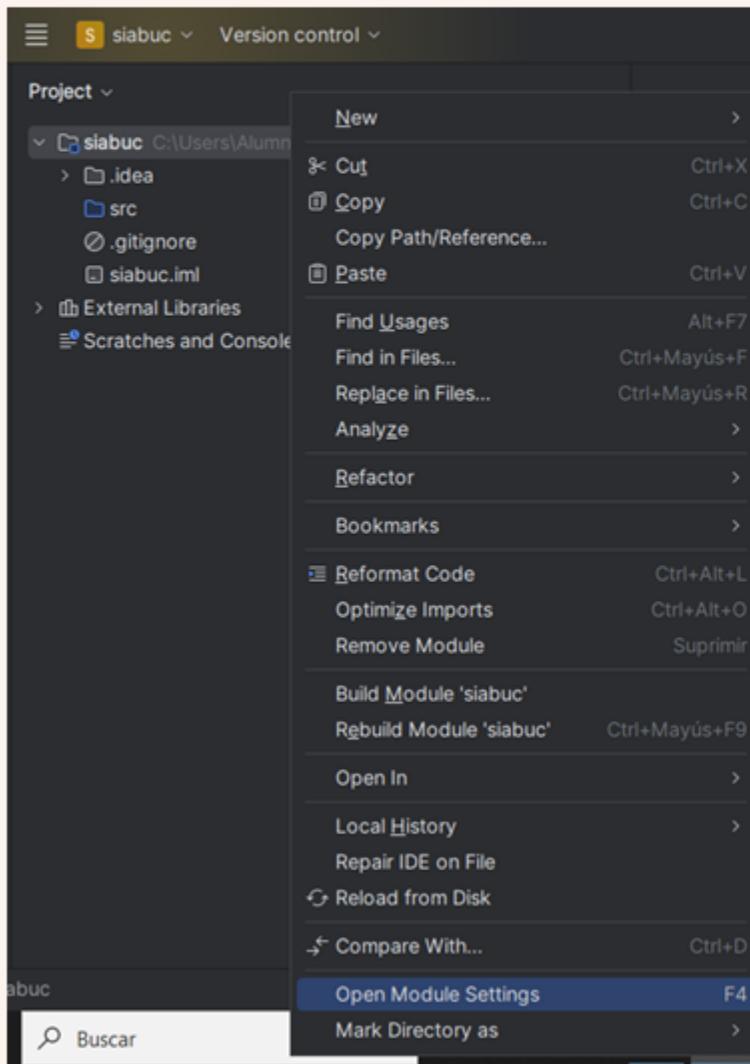
621.53 KB

Download

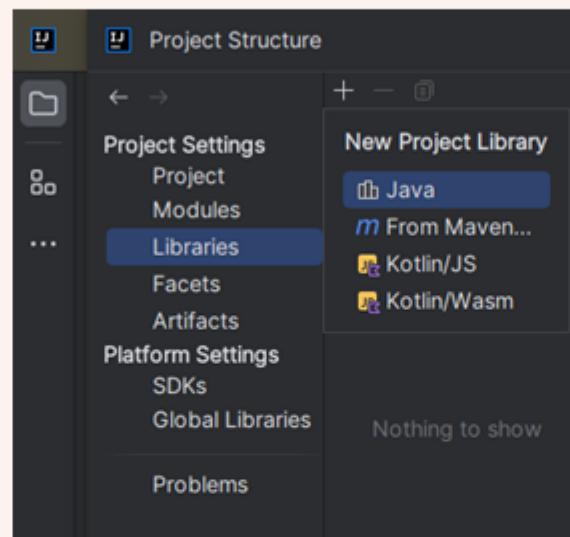
Creación de proyecto en IntelliJ Idea



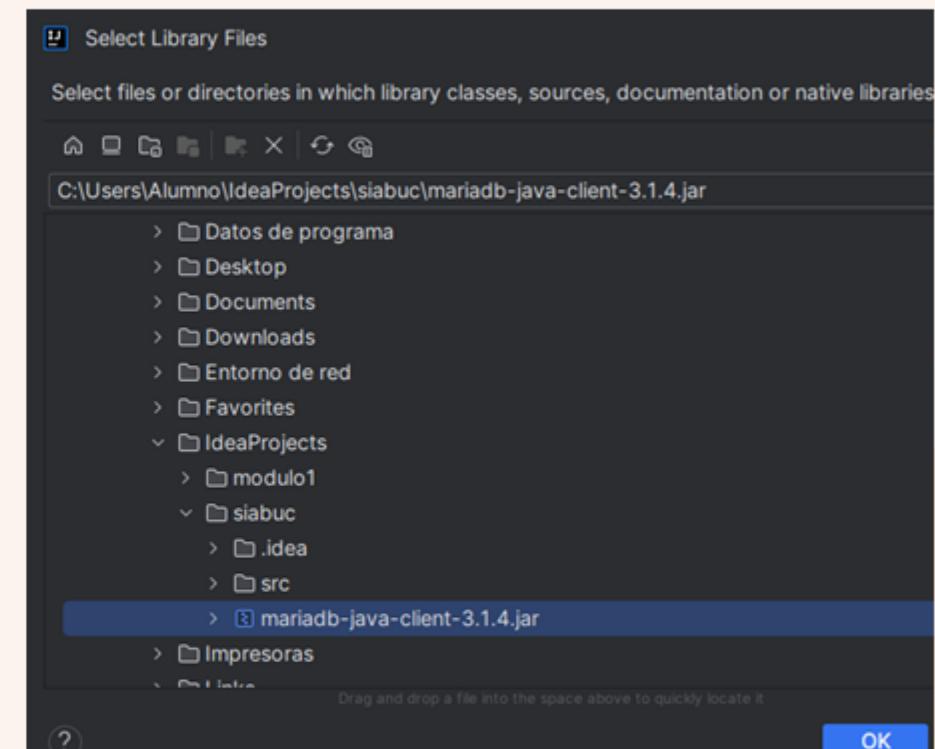
Configuración de proyecto para conectividad con BBDD



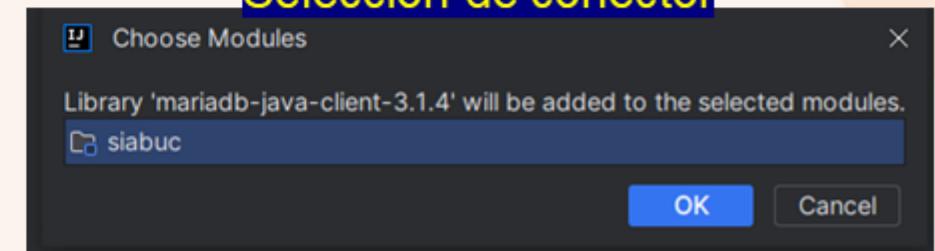
Open Module Settings



Libraries/New Project Library/Java



Selección de conector



OK

Cancel

Prueba de la conexión

© HolaBBDD.java ×

```
1 import java.sql.Connection;
2 import java.sql.DriverManager;
3 import java.sql.SQLException;
4 import java.sql.Statement;
5 public class HolaBBDD {
6     public static void main(String[] args) {
7         // Base de datos
8         String DB_URL = "jdbc:mariadb://localhost/noviembre04";    // MariaDB
9         // Credenciales
10        String usuario = "root", contrasena = "Mar14D3";
11        // Variables para la conexión y ejecución
12        Connection conexion;
13        Statement instrucion = null;
14        // Otras variables
15        int registrosAfectados = 0;
16        try{
17            System.out.println("Connectando a la base de datos...");
18            conexion = DriverManager.getConnection(DB_URL,usuario,contrasena);
19            System.out.println("Creación de una instrucción INSERT ...");
20            instrucion = conexion.createStatement();
21            String sql = "INSERT INTO numeros VALUES (1,2,3,4)";
22            System.out.println("Código ... "+sql);
23            instrucion.executeUpdate(sql); // Ejecución del comando
24            registrosAfectados= instrucion.getUpdateCount(); // ¿Cuántos registros se afectaron?
25            // Se cierran todos los objetos
26            System.out.println ("Registros afectados "+ registrosAfectados);
27            instrucion.close();
28            conexion.close();
29        } catch (SQLException e) {
30            System.out.println ("ERROR SQL: "+e.getErrorCode()+"//"+e.getMessage());
31        }
32        catch (Exception e) {
33            System.out.println ("ERROR GENERAL: "+e);
34        }
35        System.out.println ("Final de todo");
36    }
37 }
```

```
Connectando a la base de datos...
Creación de una instrucción INSERT ...
Código ... INSERT INTO numeros VALUES (1,2,3,4)
Registros afectados 1
Final de todo

Process finished with exit code 0
```

| | |
|-------------------|-----------------------|
| 21 | SELECT * FROM numeros |
| 22 | |
| < | |
| numeros (1r x 4c) | |
| edad | estatura |
| 1 | 2 |
| peso | multa |
| 3.000 | 4 |

```
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

Ejemplo

```
CREATE TABLE cat_producto (
    id int(11) NOT NULL,
    producto varchar(100) NOT NULL,
    precio float NOT NULL,
    estatus int(11) NOT NULL
) ENGINE=InnoDB
```

```
import java.sql.*; //Para conectarse a una base de datos

public class EjemploJDBC {
    // Definición de variables de conexión
    static final String JDBC_DRIVER = "com.mysql.jdbc.Driver";
    static final String DB_URL = "jdbc:mysql://localhost/prueba";

    // Database credentials
    static final String USER = "root";
    static final String PASS = "";

    public static void main(String[] args) {
        try {
            pruebaSELECT();
            pruebaINSERT();
            pruebaUPDATE();
            pruebaDELETE();
        } catch (SQLException e) {
            System.out.println ("Excepción de SQL");
        }
        catch (Exception e) {
            System.out.println ("Otro tipo de Excepción");
        }
    }
}
```

Ejemplo, continuación

```
27  private static void pruebaDELETE() throws SQLException {
28      Connection conn = null;
29      Statement stmt = null;
30      int registrosAfectados = 0;
31      try{
32          System.out.println ("DELETE");
33          //Registro de la clase JDBC driver (solo para versiones antiguas de la JVM)
34          //    Class.forName(JDBC_DRIVER);
35
36          //Se abre una conexión
37          System.out.println("Connectando a la base de datos...");
38          conn = DriverManager.getConnection(DB_URL,USER,PASS);
39
40          //Ejecución del query
41          System.out.println("Creación de una instrucción INSERT ...");
42          stmt = conn.createStatement();
43          String sql;
44          sql = "DELETE FROM cat_producto WHERE precio <50";           //UPDATE
45
46          stmt.executeUpdate(sql); //Ejecución del comando
47          registrosAfectados= stmt.getUpdateCount(); //¿Cuántos registros de afectaron?
48          //Se cierran todos los objetos
49
50          stmt.close();
51          conn.close();
52      }catch(SQLException se){
53          //Manejo de errores de SQL
54          se.printStackTrace();
55      }catch(Exception e){
56          //Manejo del resto de excepciones
57          e.printStackTrace();
58      } finally {
59
60          //Intentará cerrar lo que no se pudo cerrar antes
61          try{
62              if(stmt!=null)
63                  stmt.close();
64          }catch(SQLException se2){}//No hace nada
65          try{
66              if(conn!=null)
67                  conn.close();
68          }catch(SQLException se){
69              se.printStackTrace();
70          }
71
72      } // try
73      System.out.println("Fin del DELETE. Se borraron " + registrosAfectados + " registros.\n");
74  }
```

Ejemplo, continuación

```
75
76     private static void pruebaUPDATE() throws SQLException {
77         Connection conn = null;
78         Statement stmt = null;
79         int registrosAfectados = 0;
80         try{
81             System.out.println ("UPDATE");
82             //Registro de la clase JDBC driver (solo para versiones antiguas de la JVM)
83             //    Class.forName(JDBC_DRIVER);
84
85             //Se abre una conexión
86             System.out.println("Connectando a la base de datos...");
87             conn = DriverManager.getConnection(DB_URL,USER,PASS);
88
89             //Ejecución del query
90             System.out.println("Creación de una instrucción INSERT ...");
91             stmt = conn.createStatement();
92             String sql;
93             sql = "UPDATE cat_producto SET precio = precio* 1.20";           //UPDATE
94             stmt.executeUpdate(sql); //Ejecución del comando
95             registrosAfectados= stmt.getUpdateCount(); //¿Cuántos registros se afectaron?
96
97             //Se cierran todos los objetos
98
99             stmt.close();
100            conn.close();
101        }catch(SQLException se){
102            //Manejo de errores de SQL
103            se.printStackTrace();
104        }catch(Exception e){
105            //Manejo del resto de excepciones
106            e.printStackTrace();
107        } finally {
108
109            //Intentará cerrar lo que no se pudo cerrar antes
110            try{
111                if(stmt!=null)
112                    stmt.close();
113            }catch(SQLException se2){ } //No hace nada
114            try{
115                if(conn!=null)
116                    conn.close();
117            }catch(SQLException se){
118                se.printStackTrace();
119            }
120
121        } // try
122
123        System.out.println("Fin del UPDATE. Se modificaron " + registrosAfectados + " registros.\n");
124
125    }
```

Ejemplo, continuación

```
125
126     private static void pruebaINSERT() throws SQLException {
127         Connection conn = null;
128         Statement stmt = null;
129         int registrosAfectados = 0;
130         try{
131             System.out.println ("INSERT");
132             //Registro de la clase JDBC driver (solo para versiones antiguas de la JVM)
133             //    Class.forName(JDBC_DRIVER);
134
135             //Se abre una conexión
136             System.out.println("Connectando a la base de datos...");
137             conn = DriverManager.getConnection(DB_URL,USER,PASS);
138
139             //Ejecución del query
140             System.out.println("Creación de una instrucción INSERT ...");
141             stmt = conn.createStatement();
142             String sql;
143             sql = "INSERT INTO cat_producto (id, producto, precio, estatus) VALUES (30, 'Nombre del producto', 33, 1)"; //INSERT que :
144             stmt.executeUpdate(sql); //Ejecución del INSERT
145             registrosAfectados= stmt.getUpdateCount(); //¿Cuántos registros de afectaron?
146             //Se cierran todos los objetos
147
148             stmt.close();
149             conn.close();
150         }catch(SQLException se){
151             //Manejo de errores de SQL
152             se.printStackTrace();
153         }catch(Exception e){
154             //Manejo del resto de excepciones
155             e.printStackTrace();
156         } finally {
157
158             //Intentará cerrar lo que no se pudo cerrar antes
159             try{
160                 if(stmt!=null)
161                     stmt.close();
162             }catch(SQLException se2){ //No hace nada
163             try{
164                 if(conn!=null)
165                     conn.close();
166             }catch(SQLException se){
167                 se.printStackTrace();
168             }
169
170         } // try
171
172         System.out.println("Fin del INSERT. Se adicionaron " + registrosAfectados + " registros.\n");
173     }
```

Ejemplo, continuación

```
177  private static void pruebaSELECT() throws SQLException {
178      Connection conn = null;
179      Statement stmt = null;
180      int registrosAfectados = 0;
181      try{
182          System.out.println ("SELECT");
183          //Registro de la clase JDBC driver (solo para versiones antiguas de la JVM)
184          //    Class.forName(JDBC_DRIVER);
185          //Se abre una conexión
186          System.out.println("Connectando a la base de datos SISPIBE...");
187          conn = DriverManager.getConnection(DB_URL,USER,PASS);
188          //Ejecución del query
189          System.out.println("Creación de una instrucción ...");
190          stmt = conn.createStatement();
191          String sql;
192          sql = "SELECT * FROM cat_producto";      //Select que se va a ejecutar
193          ResultSet rs = stmt.executeQuery(sql);    //Ejecución del query. En rs quedará el resultado del SELECT
194          registrosAfectados= stmt.getUpdateCount(); //¿Cuántos registros de afectaron?
195          //Recorrido del resultado
196          while(rs.next()){
197              //Se extraen los campos del registro y se cargan en variables de instancia
198              int id = rs.getInt("id");
199              int estatus = rs.getInt("estatus");
200              String nombre = rs.getString("producto");
201              float precio = rs.getFloat("precio");
202              //Despliega un registro
203              System.out.print("ID: " + id);
204              System.out.print(", Estatus: " + estatus);
205              System.out.print(", Nombre: " + nombre);
206              System.out.println(", Precio : $" + precio);
207          } //while
208          //Se cierran todos los objetos
209          rs.close();
210          stmt.close();
211          conn.close();
212      }catch(SQLException se){
213          //Manejo de errores de SQL
214          se.printStackTrace();
215      }catch(Exception e){
216          //Manejo del resto de excepciones
217          e.printStackTrace();
218      } finally {
219          //Intentará cerrar lo que no se pudo cerrar antes
220          try{
221              if(stmt!=null)
222                  stmt.close();
223          }catch(SQLException se2){}//No hace nada
224          try{
225              if(conn!=null)
226                  conn.close();
227          }catch(SQLException se){
228              se.printStackTrace();
229          }
230
231      } // try
232      System.out.println("Fin del SELECT. Se leyeron " + registrosAfectados + " registros.\n");
233  }
```

Contacto

Lic. Carlos Eligio Ortiz Maldonado

carloseligio@ortizm.com