Bases de Datos / Elementos de B.D.

Introducción a SQL y MySQL

Departamento de Ciencias e Ingeniería de la Computación
Universidad Nacional del Sur

Lenguaje SQL

- SQL: Structured Query Language
- SQL es un lenguaje de manipulación de datos (***DML***) y un lenguaje de definición de datos (***DDL***).
- Es un lenguaje **Procedural** muy cercano al algebra relacional.
- Existen muchos dialectos y varios estándares SQL86, SQL89, SQL92, SQL99, SQL2003, SQL2005, SQL2008, SQL2011 y SQL2016.

Capacidades del Lenguaje

- ***DDL*** - Lenguaje de definición de datos.
  - Definición de relaciones y vistas.
  - Definición de usuarios y privilegios.
  - Definición de reglas de integridad.

- ***DML*** - Lenguaje de manipulación de datos.
  - Almacenar (insertar) datos.
  - Consultar datos almacenados.
  - Modificar el contenido de los datos almacenados.

MySQL

- MySQL es un Sistema De Manejo de Bases de Datos (**SMDB**) que utiliza SQL como LDD y LMD.
- **Open Source**: El código fuente está disponible. Cualquiera puede usarlo y modificarlo.
- Es **gratuito**, bajo las restricciones de la Licencia General Pública GNU (***Community Server***)
- MySQL se puede descargar de: http://dev.mysql.com/downloads/mysql/

  Opción 1: windows (X86, 64-bit), ZIP archive
  Opción 2: windows (X86, 32 & 64-bit), MSI Installer

Instalación de MySQL

- Atención! desinstalar primero cualquier versión anterior instalada de MySQL, solo si está instalada como un servicio del sistema. Ver sección 2.3 del manual (ingles: refman-8.0-en.a4.pdf ó español: refman-5.0-es.a4.pdf). Mas documentación en http://dev.mysql.com/doc/
- Opción 1: Descomprimir mysql-8.X.XX-winXX.zip en c:\, luego ejecutar c:\...\bin\mysqld --initialize-insecure
- Opción 2: Ejecutar mysql-8.X.XX-winXX.msi y seguir las instrucciones.
- Todos los ejemplos presentados en esta clase hacen referencia a la carpeta "bin" de MySQL. Nos referiremos a ella de manera general como: **…\bin**

Iniciar el servidor

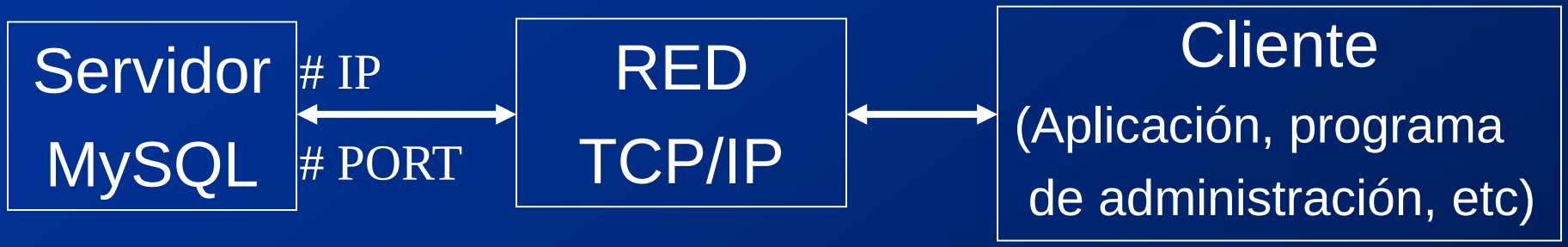
- El Servidor se puede iniciar y detener manualmente desde la linea de comandos (cmd). Hay diferentes opciones, podemos ejecutar:

  …\bin\mysqld ó

  …\bin\mysqld - -console (muestra los mensajes\ errores por la consola)
- Para detener el servidor ejecutar:
  …\bin\mysqladmin -u root shutdown
- Otra alternativa es instalar MySQL como un servicio de Windows. De esta forma, el servidor se iniciará y detendrá automaticamente cuando windows se inicie y se apague respectivamente.
- ver sección 2.3 del manual.

Conectándose al Servidor

- MySQL sigue el modelo Cliente-Servidor.



- Permite definir usuarios con claves de acceso y con diferentes privilegios.
- Por defecto existe un usuario *root* con clave vacía y acceso total a todas las bases de datos. También existe un usuario vacío `´ con clave vacía.

Conectándose al Servidor: Cliente

- MySQL provee un **programa cliente** muy sencillo que permite conectarse al servidor y ejecutar consultas, desde la línea de comandos.
- El programa se llama ***mysql.exe*** y se encuentra en la carpeta **…\bin**.
- Para conectarse al servidor utilizando el ***cliente mysql*** ejecutar: **…\bin\mysql -u root**
  (**Nota**: el servidor debe estar corriendo)
- Una vez que se conectó, se pueden ingresar sentencias SQL después del símbolo ***mysql>*** .
- ver secciones 3.1 y 3.2 del manual.

Conectándose al Servidor: Seguridad

- Como el usuario *root* tienen una clave vacia, cualquiera puede conectarse al servidor y tener acceso a todas las bases de datos !!!
- Luego de instalar un servidor deberíamos cambiar la clave de acceso del usuario *root*.
- cambiamos la clave de root a ‘bd-dcic’:

  mysql> set password for root@localhost= ‘bd-dcic';
- para conectarnos al servidor con el usuario root:

  …\bin\mysql –u root -p

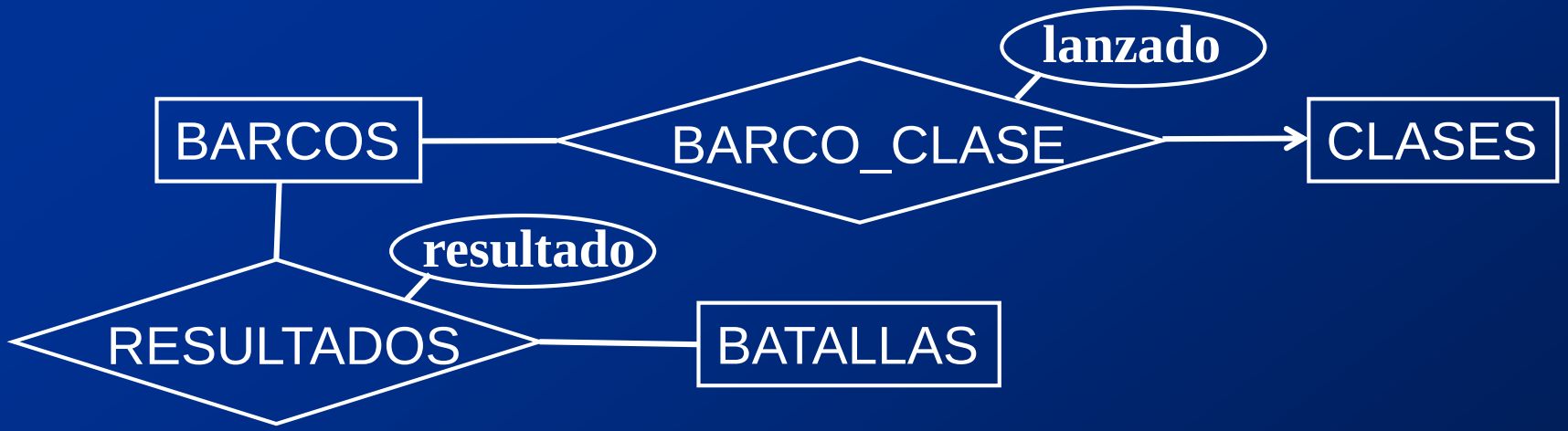
  Enter password: ******* (ingresamos bd-dcic)

Creando una base de datos

- Para crear una base de datos utilizando el cliente *mysql* hay dos opciones:
  - Ingresar las sentencias SQL una a una desde la línea de comandos, es decir, crear la base de datos y luego crear tabla por tabla. (ver sección 3.3 del manual).
  - Poner una secuencia de sentencias SQL en un archivo de texto, y ejecutar toda la secuencia de una vez. (**Batch Mode** - sección 3.5 del manual)

Creando una base de datos: ejemplo

- Consideremos una B.D. sobre barcos y *batallas:*



BARCOS(nombre barco, id, capitan)

CLASES(clase, tipo, pais, nro caniones, calibre, desplazamiento)

BARCO CLASE(nombre barco, clase, lanzado)

BATALLAS(nombre batalla, fecha)

RESULTADOS(nombre barco, nombre batalla, resultado)

Creando una base de datos: ejemplo

- Para crear la base de datos anterior utilizaremos la opción de **batch mode**.
- Creamos un archivo de texto (**batallas.sql**) con la secuencia de sentencias necesarias para crear la base de datos.
- Luego desde el cliente ejecutamos:

  mysql> \. batallas.sql

Creando una base de datos: ejemplo

- **Nota**: si una de las sentencias produce un error, es posible que alguna de las otras sentencias de la secuencia se ejecuten con exito. Antes de ejecutar el archivo nuevamente, borrar la base de datos desde el cliente con:

  mysql> drop database batallas;

- El comando **drop database** no elimina los usuarios. Si la secuencia de sentencias crea algún usuario, es necesario eliminarlos mediante:

  mysql> drop user *nombre_usuario@host***;**

SQL: Lenguaje de Definición de Datos (*DDL*)

(En MySQL ver sección 13.1 del manual)

Estas operaciones sólo está autorizado a realizarlas el DBA o administrador de la base de datos

Instrucciones del LDD

- Para la definición de esquemas:

CREATE TABLE tab_nombre **(**
atrib1 **<tipo>** [<restricción>],
atrib2 **<tipo>** [<restricción>],
....
atribN **<tipo>** [<restricción>],
<restriccion_integridad_1>,
....
<restriccion_integridad_k>**)**;

Creación de Esquemas

- Definir el esquema de la tabla (atributos y sus tipos).
- Definir, si existen, restricciones sobre los atributos.
  - un valor no puede ser nulo (**not null**)
  - restricciónes de valores (Ej. **Unsigned, Unique**), subconjunto válido, etc.
- Definir restricciones a nivel de tabla
  - Clave primaria: **primary key** (A_1, ..., A_n)
  - Clave foránea:
    foreign key (A_1, ..., A_n) **references** R(B_1, ..., B_n)
  - Índices: **key**(A_1, ..., A_n), **index** (A_1, ..., A_n)

Primary key, key, index, unique

	Crea un índice	Admite valores nulos (NULL)	Valores únicos (No admite repetidos)	Mas de una por tabla
Primary Key	SI	NO	SI	NO
Key=index	SI	SI	NO (repetidos)	SI
Unique	NO	SI	SI	SI

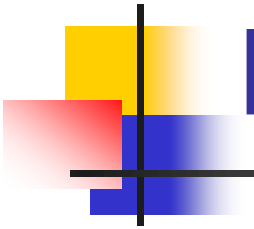
Instrucción ALTER TABLE

- Permite modificar el esquema de una tabla.

  ALTER TABLE <*nombre_tabla*>
  DROP *columna1,*
  MODIFY *columna2* <*modificación*>,
  ADD *column3* <tipo> [<*restricción*>],
  ADD/DROP CONSTRAINT *restricción1* ...

- En líneas generales la definición de las tablas debe ser estática, una vez creado todo el esquema para la base de datos se espera no tener que modificarlo.

Borrado de Esquemas



- La instrucción DROP TABLE *<tabla>* permite borrar el contenido y el esquema de una *tabla*.
- Para poder borrar una tabla la misma no debe estar referenciada por otra.
- De la misma forma, para poder borrar una columna con ALTER TABLE, esta no debe estar referenciada por otra tabla.

Lenguaje de Manipulación de Datos (DML)

(En MySQL ver sección 13.2 del manual)

Actualizando el contenido de la Base de Datos

Las siguientes instrucciones también son parte del DML y sirven para actualizar (agregar, borrar o modificar) el contenido de las relaciones.

Agregar filas

- Existen varias formas de agregar datos a una tabla. Sea *R* representa una relación con atributos A_1, ..., A_n y V_i representa un valor
- Sin especificar los atributos:

INSERT INTO *R* **VALUES** *(*V_1, ..., V_n*)*

- Especificando todos los atributos

INSERT INTO *R(*A_1, ..., A_n*)* **VALUES** *(*V_1, ..., V_n*)*

- Especificando algunos atributos en cualquier orden. Los demás atributos toman el valor por defecto de su tipo.

INSERT INTO *R(*A_1, ..., A_k*)* **VALUES** *(*V_1, ..., V_k*)*

Borrar Filas de una tabla

- Borrar todas las tuplas de una tabla:

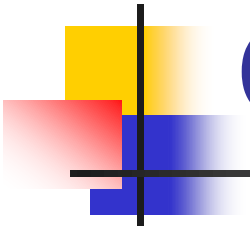
DELETE FROM *R;*

```
DELETE FROM barcos;
```

- Para borrar algunas tuplas que satisfacen cierta condición

DELETE FROM *R*
WHERE *<condición>;*

```
DELETE FROM batallas WHERE nombre_batalla="North Cape";
```



Observaciones sobre DELETE

- La operación DELETE sin cláusula de condición borra todas las filas de una tabla, **no borra su esquema.**
- La operación de DELETE será exitosa siempre que no se violen las restricciones preexistentes (por ej. de llaves foráneas).

Actualizar filas de una tabla

- Actualizar todas las tuplas de una tabla:

UPDATE *R* **SET** *<nuevas asignaciones>;*

```
UPDATE barcos SET capitan = "SIMONE";
```

- Para actualizar algunas tuplas es necesario describirlas.

UPDATE *R* **SET** *<nuevas asignaciones>;*
WHERE *<condición>;*

```
UPDATE resultados SET resultado="hundido"
WHERE nombre_batalla="Guadalcanal";
```

Observaciones sobre UPDATE

- Al igual que en el caso de INSERT y DELETE esta operación solo será exitosa si como resultado de la modificación se siguen respetando todas las restricciones preexistentes

Recuperando el contenido de la Base de Datos (consultas)

Estructura Básica de una consulta SQL

- La estructura básica consiste de tres cláusulas: **SELECT, FROM** y **WHERE.**
- **SELECT**: atributos deseados.

  proyección del álgebra relacional.
- **FROM**: una o más tablas.

  producto cartesiano del álgebra relacional.
- **WHERE**: condición sobre las túplas.

  selección del álgebra relacional.

Semántica Operacional

- Una consulta típica SQL es de la forma:

  select A_1, A_2, ..., A_n
  from r_1, r_2, ..., r_m
  [**where** P]

  - A_i representa un atributo
  - r_i representa una relación
  - P es un predicado sobre los atributos de A_1, A_2, ..., A_m.

- Esta consulta se **expresa** y **resuelve** como en Algebra Relacional:

$$\Pi_{A_1, A_2, \ldots, A_n} (\sigma_P (r_1 \times r_2 \times \ldots \times r_m))$$

Proyección

- Proyectar todos los atributos del resultado: *

SELECT * FROM ...;

```
SELECT * FROM resultados;
```

- Proyectar algunos atributos del resultado: listar los atributos separados por coma

SELECT *A1, A2, …,An* **FROM ...;**

```
SELECT nombre_barco, nombre_batalla
FROM resultados;
```

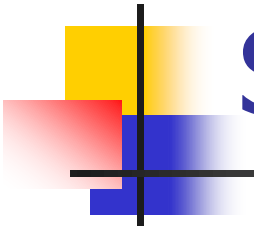
Semántica de Bolsa (Bag)

- Las cláusulas SELECT, FROM, WHERE utilizan **semántica Bag**, es decir, que no se eliminan las túplas duplicadas.
- Para utilizar una **semántica de conjunto (Set)** es necesario utilizar la cláusula DISTINCT luego del SELECT. Que elimina las túplas repetidas.

  SELECT DISTINCT A_1, A_2, …, A_n
  FROM …

Selección de túplas en SQL

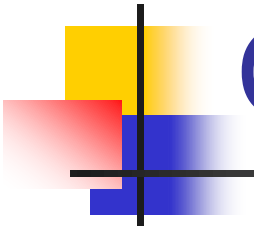
- Se implementa a través de las expresiones lógicas en el **WHERE**.
- Las expresiones se arman con los operadores de comparación **>**, **<**, **=**, **<>**, **>=**, **<=** y los operadores lógicos **AND**, **OR** y **NOT**.
- Los valores que se comparan pueden incluir atributos de las relaciones mencionadas en el FROM y/o constantes.
- Pueden usarse operadores aritméticos (+, -, etc) siempre que se respeten los tipos de los datos a comparar.



Selección: ejemplos

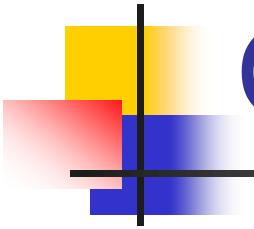
```
SELECT DISTINCT nombre_batalla
FROM resultados;

SELECT nombre_barco, nombre_batalla
FROM resultados
WHERE resultado="hundido";

SELECT nombre_barco
FROM resultados
WHERE resultado="hundido" AND
        nombre_batalla="North Atlantic";
```

Comparación de Strings

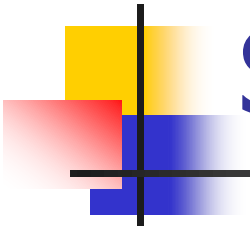


- Pueden realizarse comparaciones con operadores relacionales <, >, >=, <=, =, <> según el orden lexicográfico.
- También comparaciones de patrones:
  - <Atributo> **LIKE** <patrón>
  - <Atributo> **NOT LIKE** <patrón>
- El patrón es un string que puede contener dos caracteres especiales **%** y **_**

Comparación con LIKE



- s **LIKE** p [**ESCAPE** ‘<caracter>’]
- Los caracteres distintos de % y _ deben corresponderse con ellos mismos.
- El caracter **%** (en s) puede corresponderse con cualquier cadena de 0 o más caracteres (en p).
- El caracter **_** (en s) se corresponde con un único caracter cualquiera (en p)



Selección: ejemplos

```
SELECT *
FROM   batallas
WHERE fecha > "1942-01-01";

SELECT nombre_barco
FROM barcos
WHERE nombre_barco like 'Re%';

SELECT nombre_barco
FROM barcos
WHERE nombre_barco > 'R%';
```

SQL es *case-insensitive* excepto en los strings. MySQL se comporta según el conjunto de caracteres y regionalización configurados.

Múltiples Relaciones

- Si después de la palabra reservada FROM se enumera más de una relación, SQL hace el **producto cartesiano** entre ellas.

- Un producto cartesiano sin clausula WHERE generalmente da resultados sin significado semántico.

Multiples relaciones: ejemplos

```
SELECT *
FROM barcos, resultados;
```

```
SELECT *
FROM barcos, resultados
WHERE barcos.nombre_barco=resultados.nombre_barco;
```

```
SELECT barcos.nombre_barco, capitan, nombre_batalla, resultado
FROM barcos, resultados
WHERE barcos.nombre_barco=resultados.nombre_barco AND
      resultado="hundido";
```

Usuarios y Autorización

- La definición de usuarios y la asignación de privilegios nos permiten restringir el acceso sobre las bases de datos almacenadas en el servidor.
- Simplifica el uso del sistema y mejora la seguridad ya que permite a los usuarios acceder solo a los datos que necesitan.
- Por ejemplo, podemos definir un usuario con acceso sólo a ciertas tablas dentro de una o mas bases de datos. Para cada tabla podemos definir el tipo de acceso (leer , insertar, modificar, etc). (ver secciones 6.2, 6.3 y 13.7 de refman-8.0-en.a4.pdf o 5.6, 5.7 y 13.5 de refman-5.0-es.a4.pdf).

Usuarios y Autorización

- Los usuarios se identifican con un nombre y un dominio: *nombre@dominio*
- El dominio identifica desde donde puede conectarse el usuario al servidor de MySQL.
- ejemplos:
  - *root@localhost*, el usuario *root* solo puede conectarse desde la PC donde esta corriendo el servidor MySQL.
  - *uns@'%.uns.edu.ar'*, el usuario *uns* solo puede conectarse desde una PC del dominio *uns.edu.ar.*
  - *admin@'%'* (o *admin,* sin especificar dominio), el usuario *admin* puede conectarse desde cualquier dominio. Atencion!: se debe eliminar el usuario vacío (**`drop user ''@localhost`**) para poder conectarse usando este usuario desde localhost (ver sección 6.2.7. del manual refman-5.7-en.a4.pdf)

Usuarios y Vistas

- MySQL también permite definir vistas de la base de datos. (ver capítulo 19 del manual)
- Una vista es una tabla derivada (i.e. calculada en función de otras tablas o vistas) y persistente en la Base de Datos.
- Las vistas se crean a partir de una sentencia *select* que define su contenido.
- Podemos restringir a los usuario a acceder a una o mas vistas de una o mas bases de datos.

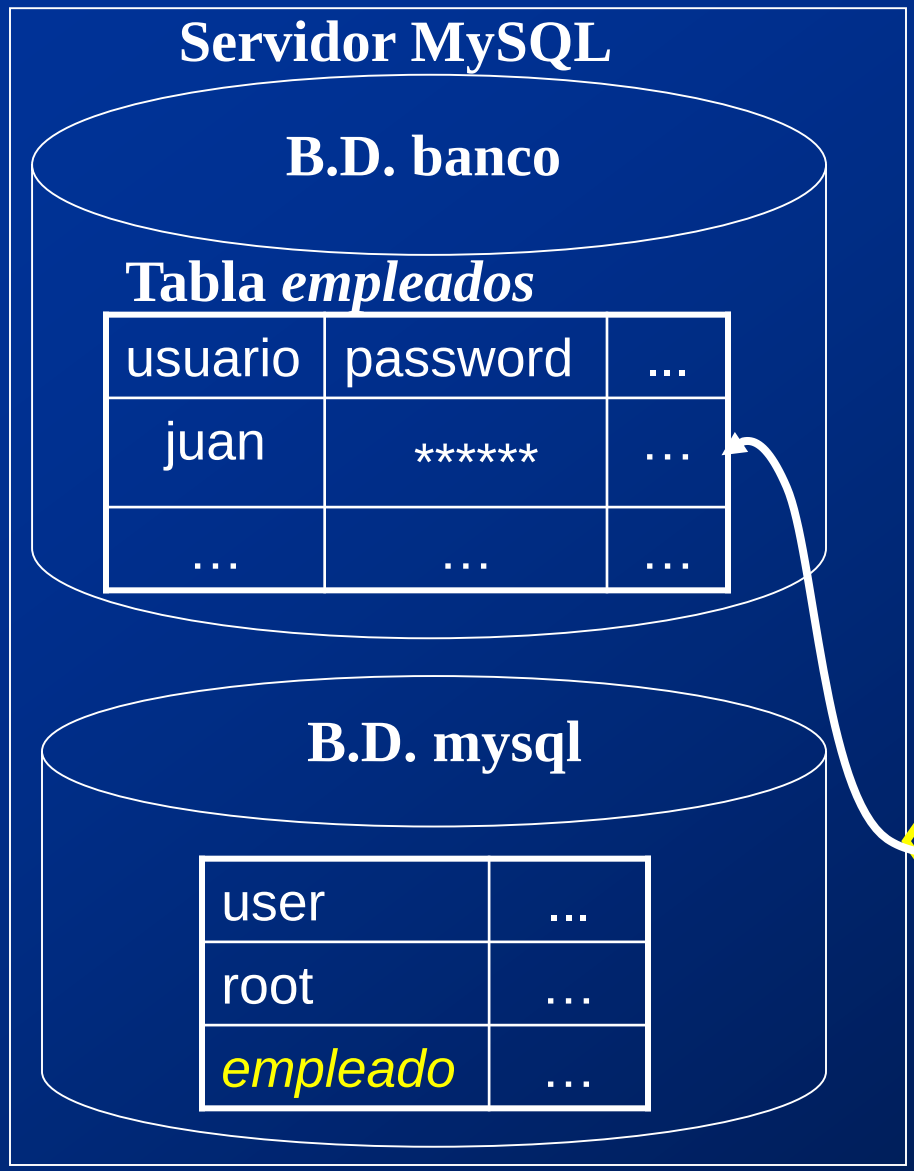
Usuarios y Autorización

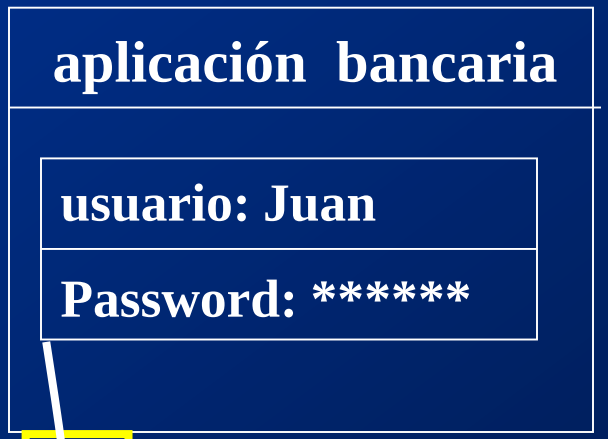
- Estos usuarios, son usuarios del servidor de MySQL en general y definen niveles o perfiles de acceso al servidor.
- Generalmente estos usuarios serán utilizados por las aplicaciones (no por personas) para acceder al Servidor.
- Un caso particular seria por ejemplo un usuario *admin*, utilizado por el administrador (persona) de la B.D. para tareas de mantenimiento
- Si un sistema requiere la definición de usuarios (personas), es conveniente definir estos usuarios dentro de la B.D. del sistema y manejar el acceso desde la/s aplicación/es utilizando usuarios definidos en el servidor.

Usuarios y Autorización: ejemplo

- Supongamos que en una aplicación que accede a una B.D. *banco* de un sistema bancario, queremos definir usuarios para controlar el acceso de los empleados.
- La B.D. *banco contendrá* una tabla *empleados* con información de los empleados y su password.
- Definimos un usuario *empleado* en el servidor a través del cual se conectará la aplicación. El usuario *empleado* define el nivel de acceso de un empleado en general.
- La aplicación se encargará de controlar el acceso de los empleados (por ejemplo requiriendo el usuario y password) utilizando la información de tabla *empleados*.

Usuarios y Autorización: ejemplo





La aplicación se conecta al servidor por medio del usuario *empleado* que tiene acceso a parte de la B.D *banco* y le permite acceder a la tabla *empleados* de la B.D. *banco* para validar su identidad.

MariaDB

- MariaDB es un proyecto de código abierto para mantener y dar soporte a una versión gratuita de MySQL.
- Si bien MySQL prove una versión gratuita del servidor (**Community Edition),** para utilizarlo de manera comercial y recibir soporte se debe pagar una licencia (**Enterprise Edition**).
- MariaDB esta basado en MySQL y es compatible con algunas versions de MySQL

  https://mariadb.org/en/about/

  https://mariadb.com/kb/en/library/mariadb-vs-mysql-compatibility/

MariaDB

- Se puede descargar de *https://downloads.mariadb.org/*
- Opción zip file: descargar y descomprimir mariadb-X.X.XX-winXX.zip en (por ejemplo) c:\.
- Una vez descomprimido tiene la misma estructura de carpetas que MySQL y funciona exactamente igual. En la Carpeta …/bin se encuentra el servidor y el cliente como vimos en los ejemplos anteriores.