<devAcademy>

Bases de datos JAVA

BASES DE DATOS

- La API por excelencia usada para conectarse a un SGBD es JDBC
- Cada vendor o marca de base de datos, proporciona un driver específico para la interacción con su Base de datos desde JAVA

JDBC

- Para encontrar el dirver JDBC apropiado, debemos conocer antes la versión de la BD con la vamos a trabajar
- Al final, el Driver, no será más que una Clase Java, que nos proporciona acceso transparente a la base de datos, usando para ello los métodos descritos en el API

Cadena de CONEXIÓN

- Para conectarnos a una base de datos, una vez indicado el driver, debemos proporcionar la siguiente información
  - SERVIDOR //ip donde está la BD
  - PUERTO//aplicación dentro del servidor
  - NOMBRE USUARIO
  - CONTRASEÑA

Clases principales

- DriverManager: Es la encargada de solicitar conexiones entre mi aplicación y la base de datos
- Connection: Es la clase que usaremos para conectarnos.

Clases principales

- Statement: Es la clase que emplearemos para ejecutar nuestras instrucciones SQL hacia la base datos.

  Métodos importantes

  executeQuery() //para ejecutar consultas

  execute()//para ejecutar operaciones de borrado, insercción o actualización

Clases principales

- ResultSet: Es la clase que emplearemos para gestionar los resultados de nuestras conusultas
- Deberemos iterar sobre él, para ir recorriendo los resultados de nuestras consultas

Clases principales

- ResultSet: Nos permite seleccionar los campos de la tabla por el nombre o por la posición

  getString (“id”) //nos da el campo id

  getString (1)//nos da el campo número 1

Clases principales

- PreparedStatement: Esta clase nos permite utilizar instrucciones con parámetros, que indicaremos como símbolos de interrogación (?)

Clases principales

- PreparedStatement: Debemos indicar la sentencia en el momento de creación del objeto PrepedStatement

```
PreparedStatement pstmt =
conn.prepareStatement("Select * from
employees where employee_id = ?");
pstmt.setInt(1, 110);
rset = pstmt.executeQuery();
```

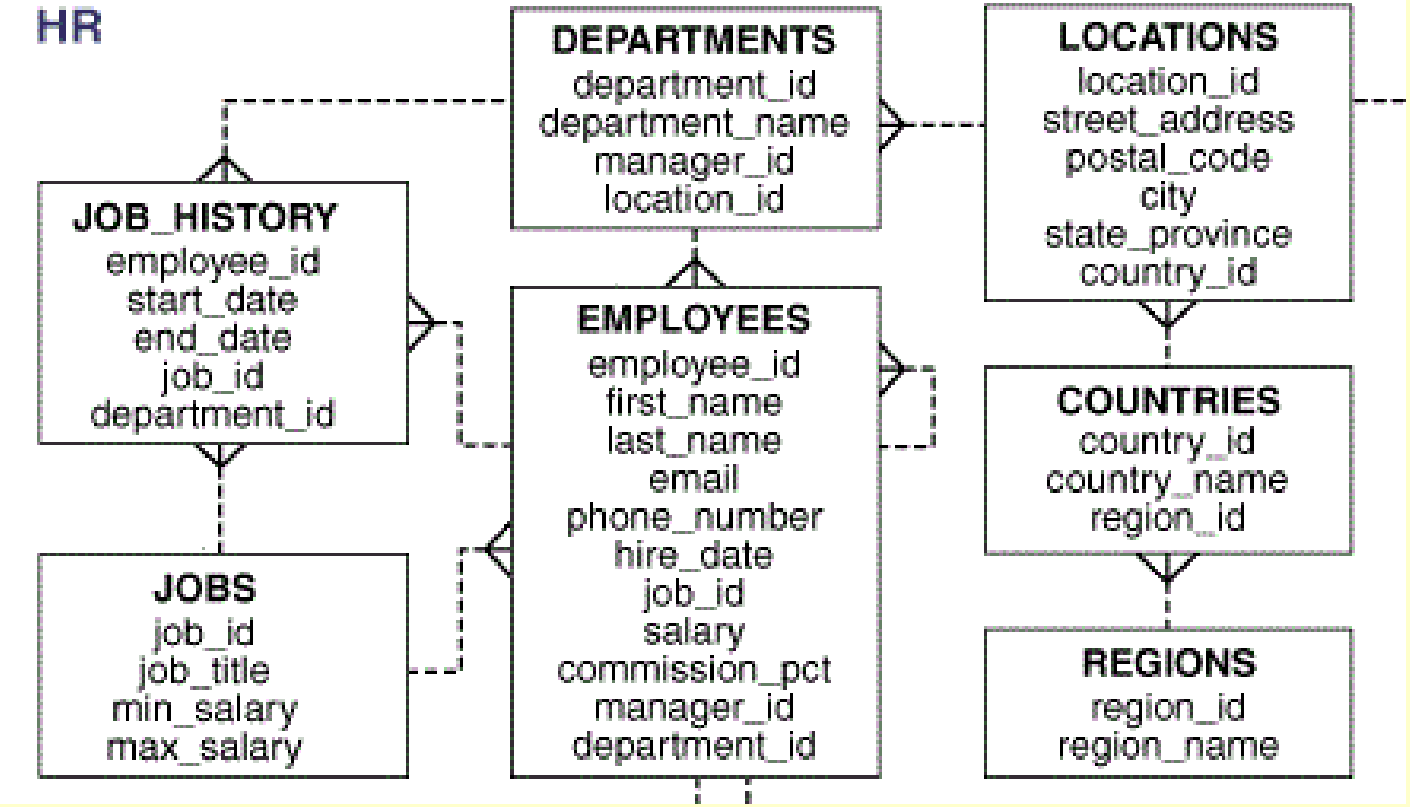
Liberando recursos

- Tanto las Connections, como los Statement y los ResultSet, son objetos que debemos cerrar después de su uso mediante el método close()
- Resulta vital, que empleemos una estructura try-catch-finally para aseguraros que realmente liberamos los recursos correctamente

MODELO E/R

- El Modelo E/R es un lenguaje formal que nos permite describir los objetos que conforman nuestra base de datos, así como definir la relación entre ellas

MODELO ESQUEMA HR



PRÁCTICA 1

- Realizar un programa que consulte los empleados con un salario mayor a 3000. Para cada empleado recuperado, crear un objeto Empleado e id conformando un ArrayList de empleados. Finalmente, ordenar la lista por el salario, de mayor a menor y mostrarla.

RDBMS

- Sistema de Gestión de Base de Datos Relacionales son las iniciales del acrónimo inglés RDBMS
- En él, las entidades que lo componen, guardan una serie de normas
- Oracle, Acces, MySQL, SQLServer, son ejemplos de implementaciones RDBMS

RDBMS Características

- Toda tabla posee una clave primaria, que identifica unívocamente a cada registro de la entidad. No puede repetirse (Primary Key)
- Cuando desde un atributo, campo o columna de una tabla se está guardando la PK de otra tabla, hablamos de que ése valor actúa como clave ajena (Foreign Key)

RDBMS Características

- Cuando un registro es eliminado o modificado, los registros que contienen alguna clave ajena pueden modificarse o eliminarse de forma automática
- Estas restricciones, quedan definidas por el administrador de la base de datos en el momento de definición de la clave ajena

RDBMS Características

- Lo que persigue la política de modificación transitiva y automática es que los entidades de la base de datos guarden una relación consistente y coherente. Esto se conoce por el nombre genérico de Integridad Referencial, que todo SGBDR debe cumplir

RDBMS Características

- Todos los RDBMS tienen su propio lenguaje para manipular las entidades y sus datos
- Además de su propio lenguaje, todas deben aceptar SQL (Standar Query Lenguage)

SQL Instrucciones

- La gestión de la base de datos, su definición y mantenimiento, corre a cargo de un administrador (DBA)
- Nosotros, como programadores, no debemos preocuparnos en exceso por la gestión de la BD en si, pero sí debemos conocer algunas instrucciones

SQL Instrucciones

- Las más importante son:
  - SELECT para diseñar consultas o querys, recuperar datos o también se emplea el anglicismo atacar la base de datos. En definitiva, para recuperar información.
  - INSERT para añadir nuevos registros a una entidad/tabla de la base de datos

SQL Instrucciones

- DELETE para eliminar registros/filas de una determinada entidad/tabla
- UPDATE para actualizar algunos campos o valores de un registro

SQL Instrucciones

- Todas estas instrucciones caen dentro del tipo de instrucciones de MANIPULACIÓN. El acrónimo en inglés es DML (Data Manipulation Language)
- El resto de instrucciones, empeladas para crear nuevos objetos y/o relaciones son las DDL (Data Definition Language), en principio, fuera de nuestro ámbito.

SQL COMMIT-ROLLBACK

- Cuando ejecutamos una instrucción a través de un Statement, puede suceder que la cosa salga bien o que haya habido algún problema, como un fallo en la comunicación con el servidor, una violación de alguna restricción, etc..

SQL COMMIT-ROLLBACK

- Por ello, las modificaciones que implican la ejecución de nuestras instrucciones, no tienen un efecto real en la BD
- En principio, estamos trabajando en una copia, y hasta que no nos queda claro que todas las instrucciones han sido exitosas, no deberíamos darlas por hecho

SQL COMMIT-ROLLBACK

- Aquí tenemos el concepto de **transacción**. Una transacción, es un conjunto SQL de instrucciones relacionadas, que se ejecutan todas secuencialmente. O todas salen bien o si falla alguna, se aborta todo el proceso

SQL COMMIT-ROLLBACK

- Cuando tenemos la garantía de que una transacción ha sido exitosa, debemos realizar COMMIT sobre la conexión, para que nuestros cambios sean persistentes y se graben de verdad

SQL COMMIT-ROLLBACK

- Sin embargo, si algo falla, deberemos hacer ROLLBACK en la sección catch del tratamiento de la excepción, para deshacer las modificaciones que han intentado cometer las últimas instrucciones

SQL COMMIT-ROLLBACK

- ¿Hasta dónde deshace ROLLBACK?
  - O bien hasta la creación de la conexión
  - O bien hasta encontrar un punto de salva guarda o SAVEPOINT

    connection.rollback(savepoint);

SQL SAVEPOINT

- Cuando vamos a iniciar una nueva transacción y/o estamos seguros que nuestra BD está en un estado consistente, es coveniente fijar un punto de guarda a la conexión mediante el método setSavepoint();

SQL TRANSACCIONES

- Con esta forma de trabajar, empleadno Commit, Rollbacks y Savepoints, estamos preservando la integridad de la BD

- IMPORTANTE: Por defecto, la conexión tiene un autocommit a true (así que deberemos desactivarlo antes de empezar)

SQL VIEW

Concepto de Vista → Normalmente, nuestras entidades están representadas en la base de datos como una TABLA

Algunas de esas tablas, son realmente vistas u objetos VIEW dentro del esquema

SQL VIEW

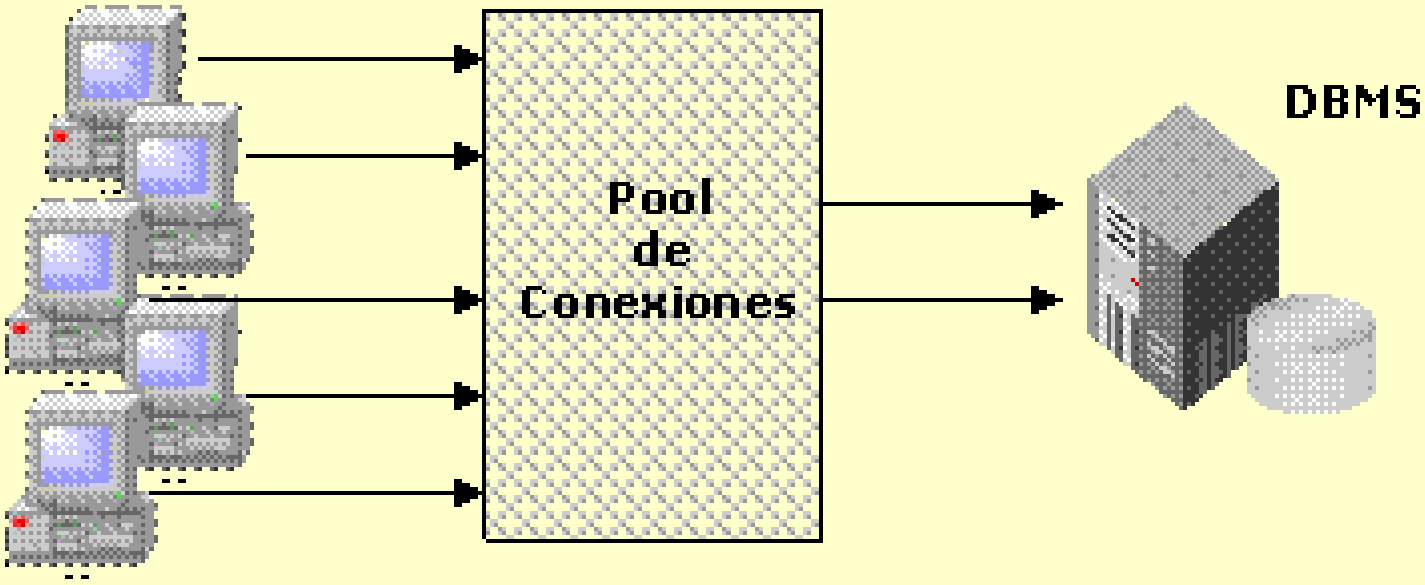
Una vista, es una entidad que sin tener suficiente entidad en el esquema E/R para ser una tabla, sí la tiene para un grupo de usuarios de la base de datos.

A efectos prácticos, funcionamos con ella como con otra tabla, aunque realmente, su comportamiento interno no sea tal.

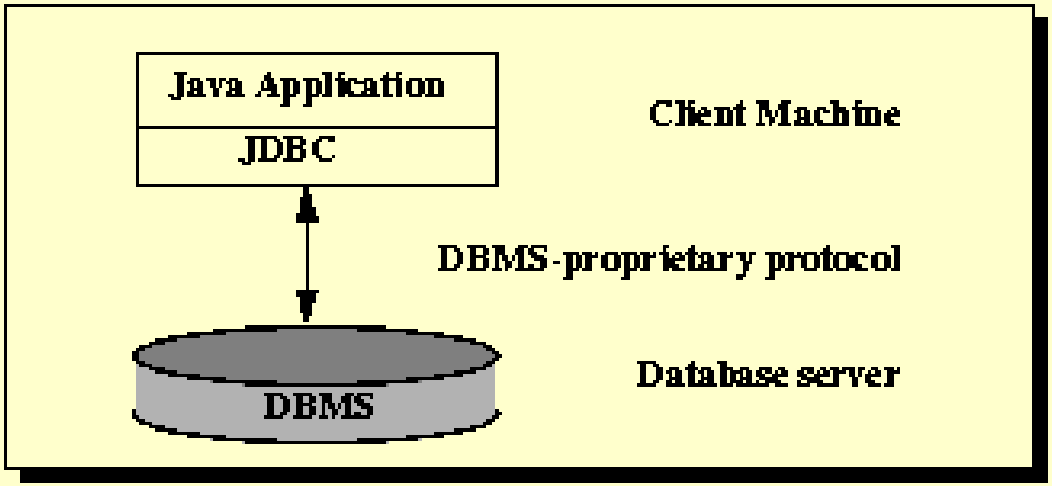
POOL de CONEXIONES

Concepto: Crear y cerrar conexiones, es un proceso costoso. Para mejorar el rendimiento y evitar que cada proceso tenga que gestionar la apertura y el cierre de la conexión con la base de datos surge el concepto de Pool de conexiones o simplemente Pool.

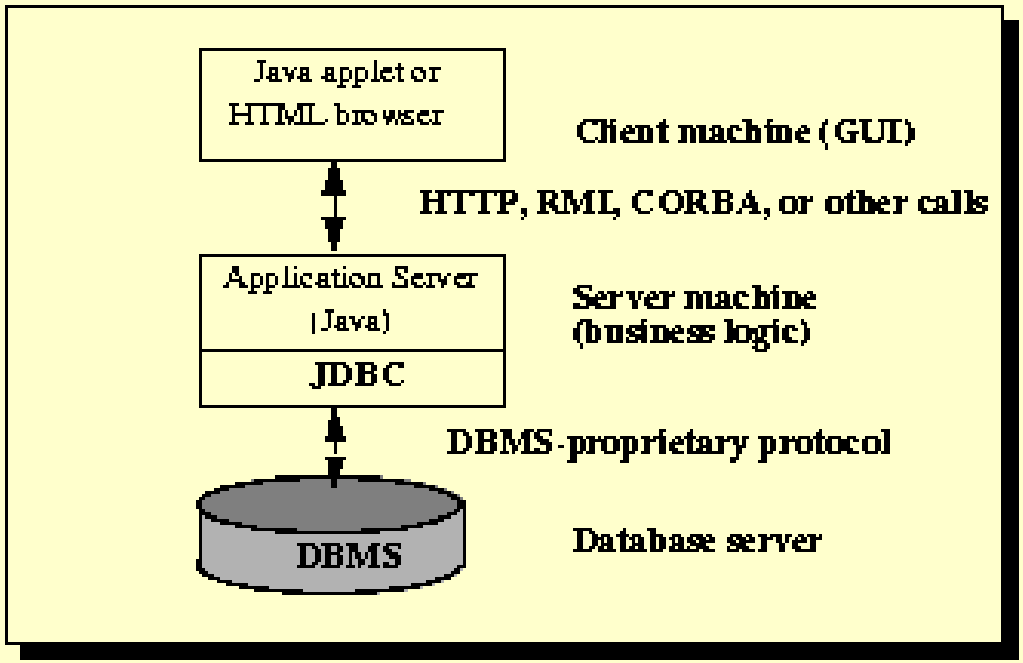
POOL de CONEXIONES



MODELO 2 CAPAS



MODELO 3 CAPAS



PATRÓN DTO DAO

- En este patrón, cada clase que representa una entidad, va a tener dos clases que lo representan:
  - Una a nivel de memoria (objeto)
  - Otra a nivel de base de datos (tabla)

Objeto DTO

- El objeto DTO (Data Type Object) será la clase java que representa una entidad en Memoria, con todos los atributos necesarios que identifican y definen a la clase para JAVA

Objeto DTO

- Los métodos que implementa al menos un objeto DTO son los getter y setter de sus atributos, así como una o varias versiones del constructor
- Con esto, nuestro DTO es a efectos prácticos un POJO (Plain Old Java Object)

Objeto DAO

- El objeto DAO (Data Access Object) será la clase java que representa una entidad en base de datos, con todos los atributos , columnas o campos que definen a un registro de la tabla de mi base de datos.

Objeto DAO

- Los métodos de una clase DAO, deberán contener todo lo relativo a la interacción de un objeto DTO en la base de datos
- Por tanto, la clase DAO, va a tener los métodos DML tanto para insertar un DTO en la base de datos, como para modificarlo, eliminarlo o recuperarlo

Objeto DAO

- Encapsulando (wrap) todo lo relativo a la base de datos en las clases DAO, logramos desacoplar nuestras clases Javas, con lo que tenga que ver con la base de datos

Objeto DAO

- Si mañana, por ejemplo, cambiamos de marca de base de datos o se realiza una mejora en el modo de acceso, estará sólo afectando a mis clases DAO, logrando mantener la vigencia de mis clases DTO

CRUD

- El comportamiento de los objetos DAO, puede categorizarse mediante este acrónimo: CRUD Create, Read, Update y Delete
- En otras palabras, como ya hemos descrito, nuestros objetos DAO crean, leen, actualizan y borran registros en la base de datos que equivalen a DTOS en memoria

CRUD - Interfaz

- Ya que el comportamiento de los objetos se puede generalizar, resulta más que apropiado modelarlo en Java usando una Interfaz que defina esos operadores
- Y a su vez, hacer que nuestros objetos DAO implementen esa interfaz

CRUD - Interfaz

```
public interface CRUD {

    create (GenericDTO ObjectDTO) throws Exception;

    update (GenericDTO ObjectDTO) throws Exception;

    read (String Condicion) throws Exception;

    delete (String Condicion) throws Exception;
```