Bases de Datos

Ingeniería de Software

Laboratorio

L. en C.C. Miguel Angel Piña Avelino

PostgreSQL

¿Qué es PostgreSQL?

PostgreSQL es un sistema de gestión de bases de datos relacional, orientado a objetos y libre.

PostgreSQL

PostgreSQL se define como:

- Una poderosa base de datos open-source ***objeto-relacional***.
- Es totalmente compatible con caracteŕisticas **ACID** (*atomicidad*, *consistencia*, *isolación* y *durabilidad*).
- Con soporte para *llaves foráneas*, *uniones*, *vistas*, *triggers* y *procedimientos almacenados,* incluso *Common Table Expressions (CTE)*.
- Soporte de la mayoŕia de los tipos de datos del estándar ANSI SQL 2008.
- Interfaces para trabajar con varios lenguajes de programación (*C/C++*, *Java*, *Perl*, *.Net*, *Python*, etc.)

PostgreSQL

También tiene las siguientes caracteŕısticas:

- Ser una base de datos de clase empresarial
- Control de concurrencia multiversión
- Replicación aśincrona
- Transacciones anidadas
- Altamente escalable

PostgreSQL

Algunos límites y características generales que se incluyen en PostgreSQL.

Tamaño máximo de la base de datos	Ilimitado
Tamaño máximo de tabla	32TB
Tamaño máximo de fila	1.6TB
Tamaño máximo de campo	1GB
Máximo de filas por tabla	Ilimitado
Máximo de columnas por tabla	250 - 1600 dependiendo del tipo de la columna
Máximo de índices por tabla	Ilimitado

PostgreSQL

Podemos consultar más información acerca de la base de datos en:

https://www.postgresql.org/docs/10/index.html

Instalación en Ubuntu/Debian:

sudo apt install postgresql-10 postgresql-client-10

PostgreSQL

En caso de que no se encuentre en el repositorio la versión 10, podemos realizar lo siguiente:

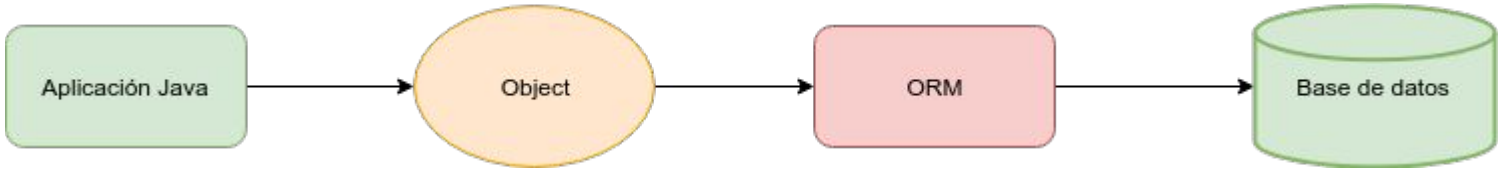
```
> sudo add-apt-repository "deb
http://apt.postgresql.org/pub/repos/apt/ xenial-pgdg main"
> wget --quiet -O -
https://www.postgresql.org/media/keys/ACCC4CF8.asc | sudo apt-key
add -
> sudo apt update
> sudo apt install postgresql-10 postgresql-client-10
```

Hibernate

Hibernate

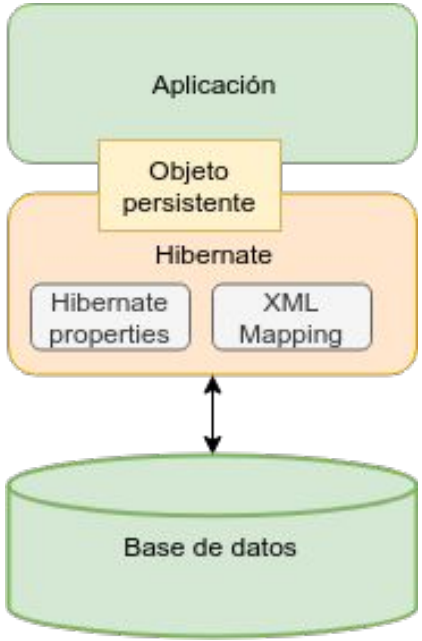
- Object-Relational Mapping o ORM es una técnica de programación para mapear un modelo de objetos de aplicación a tablas relacionales de una base de datos.
- Hibernate es una herramienta ORM basada en Java que provee un framework para transformar objetos en información para tablas relacionales y viceversa. Provee una implementación de Java Persistence API (JPA).
- El framework provee opciones para llevar datos de POJOs tradicionales a bases de datos tradicionales usando anotaciones de JPA y configuraciones basadas en XML.

Hibernate



Hibernate

Hay 4 capas en la arquitectura de Hibernate: Capa de aplicación Java, capa de aplicación Hibernate, capa Backend y capa de base de datos



Hibernate

Antes de crear algo con Hibernate, debemos conocer algunos elementos de la arquitectura de Hibernate.

Elemento	Descripción
SessionFactory	Es una fábrica de sesiones y es un cliente del proveedor de *Connection.* Mantiene un caché de segundo nivel para los datos.
Session	Provee una interfaz entre la aplicación y los datos de la base de datos. Es un objeto de corta vida y envuelve una conexión JDBC. También es una fábrica de Transaction, Query y Criteria. Matiene un caché de primer nivel para los datos. Provee métodos para INSERT, UPDATE y DELETE.
Transaction	Especifica la únidad atómica de trabajo y es un parámetro opcional.

Hibernate

Antes de crear algo con Hibernate, debemos conocer algunos elementos de la arquitectura de Hibernate.

Elemento	Descripción
ConnectionProvider	Es una fábrica de conexiones JDBC y abstrae la aplicación desde DriverManager o DataSource. Es un parámetro opcional.
TransactionFactory	Es una fábrica de transacciones. Es un parámetro opcional.

Integrando Hibernate en Maven

Agregamos al archivo pom.xml (en la sección de dependencias) lo siguiente:

```
<dependency>
  <groupId>org.hibernate</groupId>
  <artifactId>hibernate-core</artifactId>
  <version>5.4.1.Final</version>
</dependency>
```

```
$ git checkout b7d6b4a
```

Integrando Hibernate en Maven

Supongamos que tenemos el siguiente código en SQL que vamos a cargar en una base de datos llamada **test**.

```
begin;
drop schema if exists datos cascade;
create schema datos;
drop table if exists datos.user;
create table datos.user (
  id serial primary key,
  name text not null,
  created_by text not null,
  created_date date not null
);
commit;
```

```
$ git checkout ce842e3
```

Integrando Hibernate en Maven

Vamos a crear un par de clases. Una clase que representa el modelo de la base de datos (la tabla user) y otra tabla que es una clase de utilería para manipular los datos que vamos a insertar en la base de datos.

La clase modelo estará en src/main/java/com/miguel/modelo/User.java

```
git checkout bca64ce
```

La clase utilería estará en src/main/java/com/miguel/modelo/Utility.java

```
git checkout 7166abb
```

Integrando Hibernate en Maven

Agregamos la configuración de hibernate a través del archivo `src/main/resources/hibernate.cfg.xml` y utilizamos un archivo de ayuda para generar sessiones de hibernate `src/main/java/com/miguel/modelo/HibernateUtil.java`

```
git checkout de543f2
```

Y modificamos el registro que habíamos hecho anteriormente.

```
git checkout fe60203
```