UNIVERSIDAD DE MÁLAGA | uma.es

ADaMaSBio: Adviser of Database Management Systems with Biological Databases

Samuel Pérez Fernández
Marco Muñoz Pérez
Juan Luis Onieva Zafra

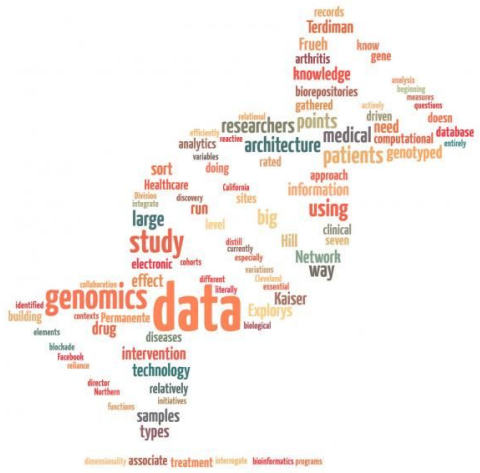
UNIVERSIDAD DE MÁLAGA

Índice

UNIVERSIDAD DE MÁLAGA | uma.es

1. Introducción
2. Diseño
3. Implementación
4. Resultados
5. Conclusión

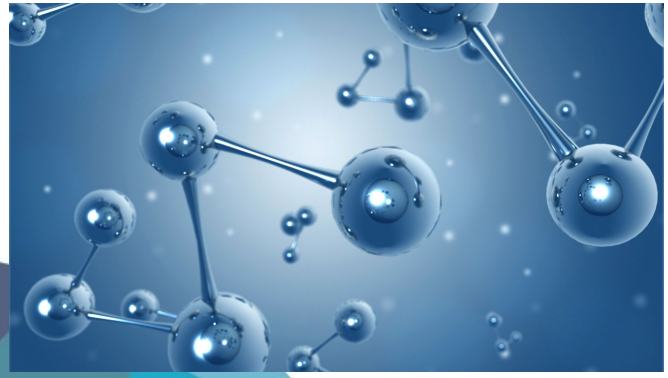
1. Introducción



Objetivos

UNIVERSIDAD DE MÁLAGA | uma.es

1. Administrar base de datos biológica a través de diferentes gestores.
2. Analizar rendimiento y eficiencia de cada uno.
3. Compararlos entre ellos.

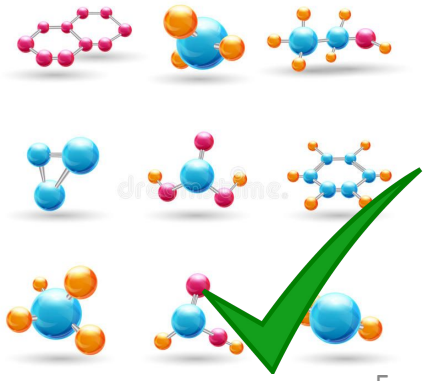


ChEBI

UNIVERSIDAD DE MÁLAGA | uma.es

La base de datos que vamos a implementar es ChEBI.
¿Qué es ChEBI?

1. Base de datos sobre entidades químicas de interés biológico.
2. Moléculas que intervienen en procesos biológicos.
3. Moléculas “pequeñas.”



ChEBI

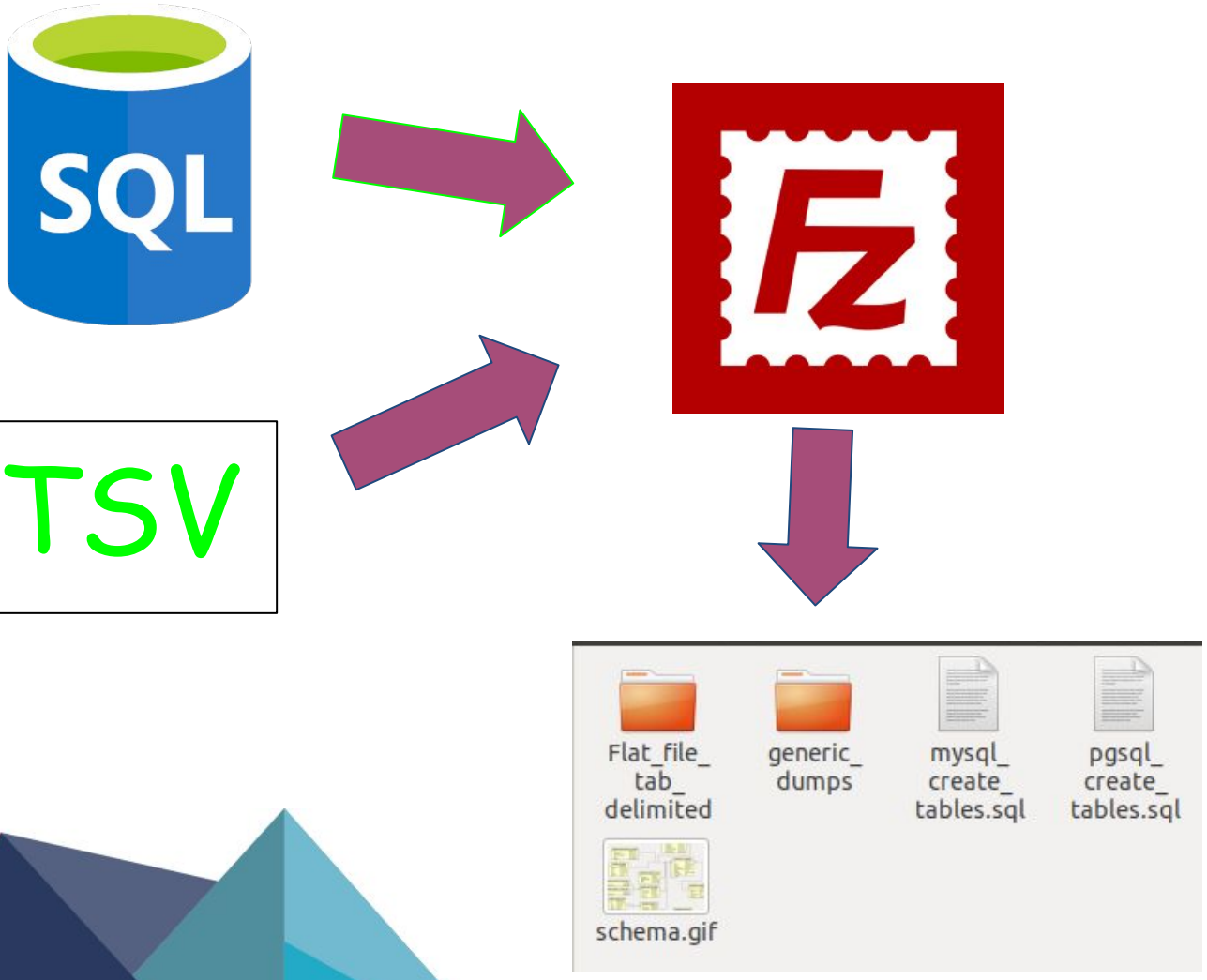
1. Iniciado en 2002 por el (EBI).
2. Primera publicación en 2004.
3. Contiene 12000 entidades moleculares.
4. Puede considerarse “pequeña”.
5. Sin embargo, se asegura tener información estandarizada y no ambigua.

Descarga de ChEBI

UNIVERSIDAD DE MÁLAGA | uma.es

Portal ChEBI → Descargas

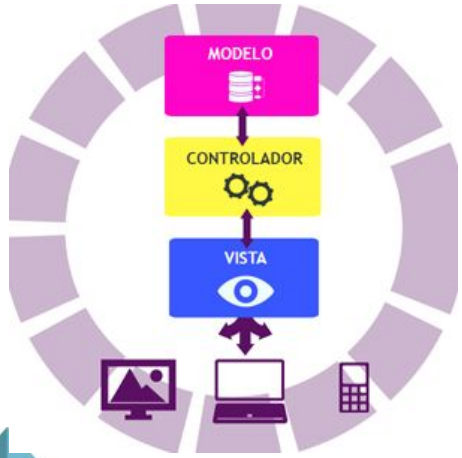
Descarga de ChEBI

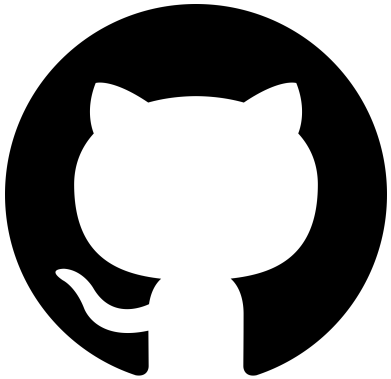


2. Diseño

UNIVERSIDAD DE MÁLAGA | uma.es

- Aplicación de escritorio
- MVC
- GitHub





2. Diseño

- Interfaz gráfica: Java Swing

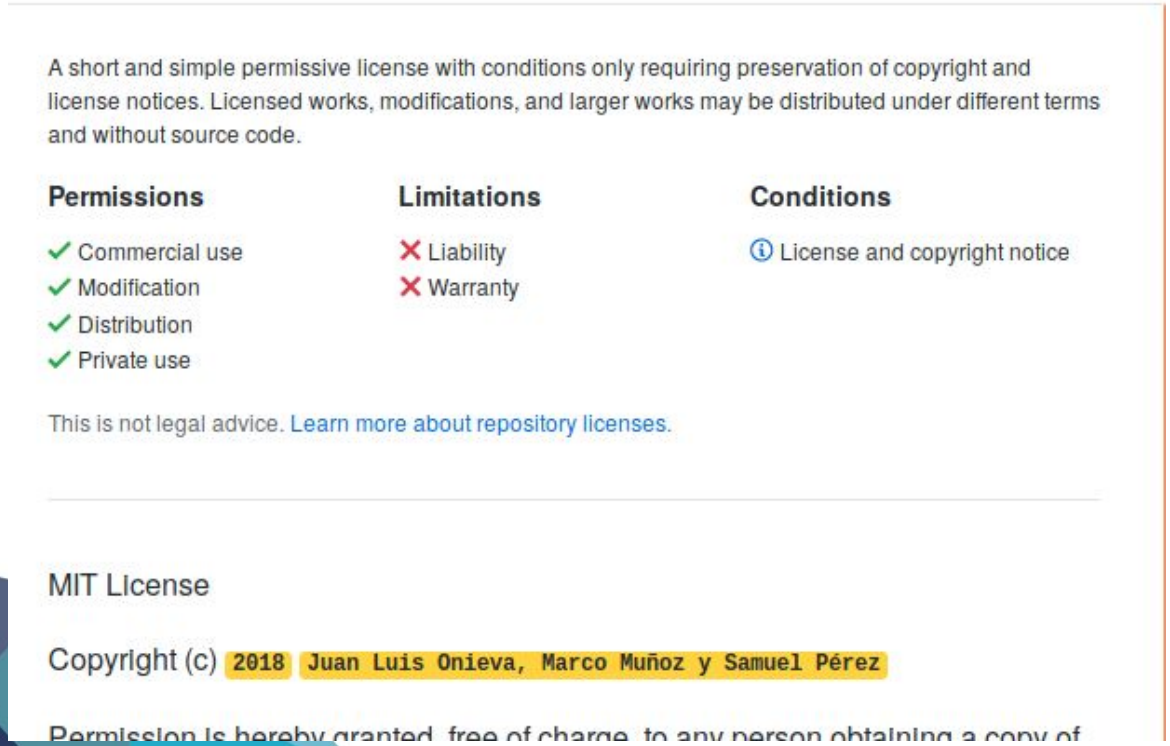
- Control de dependencias: Maven

Maven™

2. Diseño

- GitHub

https://github.com/JuanluOnieva/BDB2018



A short and simple permissive license with conditions only requiring preservation of copyright and license notices. Licensed works, modifications, and larger works may be distributed under different terms and without source code.

Permissions

- ✓ Commercial use
- ✓ Modification
- ✓ Distribution
- ✓ Private use

Limitations

- ✗ Liability
- ✗ Warranty

Conditions

- ⓘ License and copyright notice

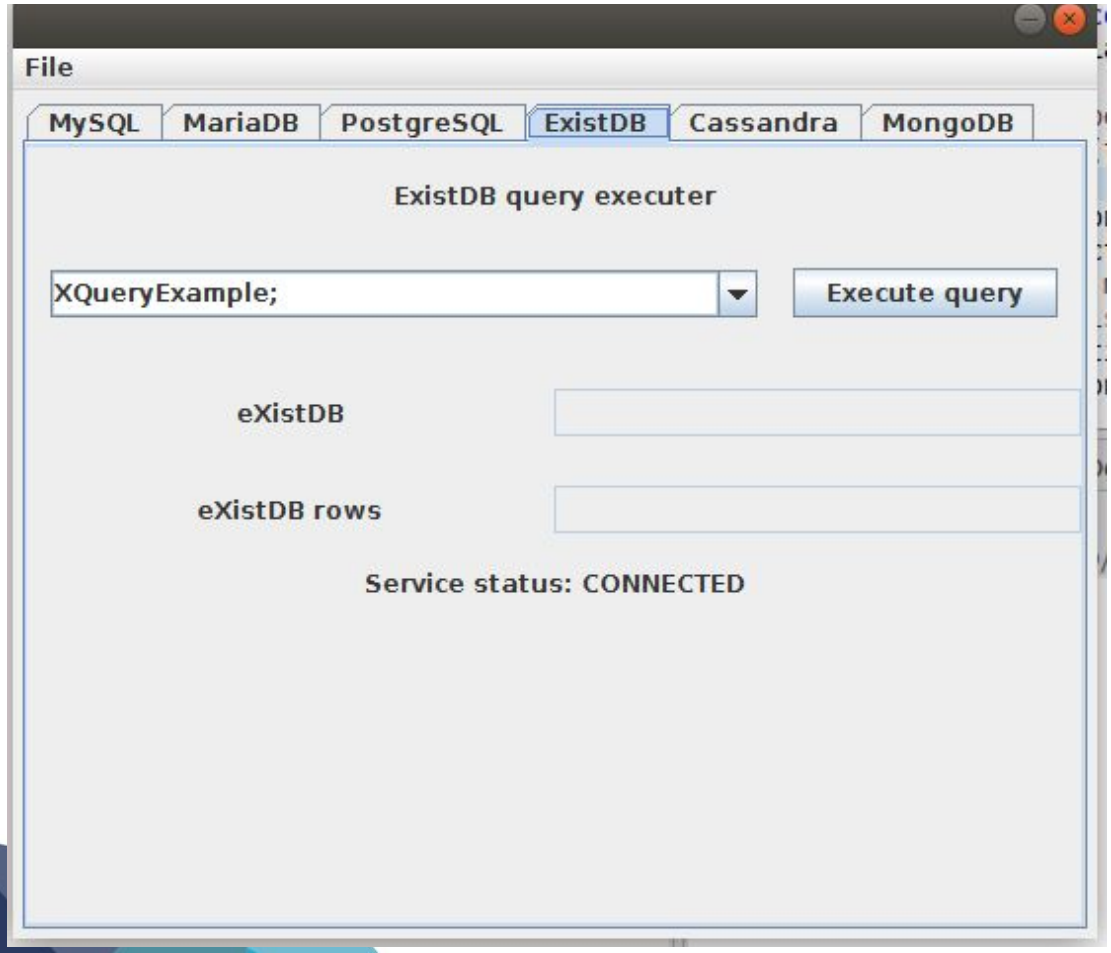
This is not legal advice. Learn more about repository licenses.

MIT License

Copyright (c) 2018 Juan Luis Onieva, Marco Muñoz y Samuel Pérez

Permission is hereby granted, free of charge, to any person obtaining a copy of

2. Diseño


File
MySQL
MariaDB
PostgreSQL
ExistDB
Cassandra
MongoDB
ExistDB query executer
XQueryExample;
Execute query
eXistDB
eXistDB rows
Service status: CONNECTED

3. Implementación

UNIVERSIDAD DE MÁLAGA | uma.es

- Docker
- SGBD: Relacionales, XML y NoSQL
  - ¿Por qué?
  - Despliegue de la BD
- Conexión con nuestra app

Docker es un proyecto de código abierto capaz de automatizar el despliegue de aplicaciones dentro de contenedores de software.

Permite instalar fácilmente sistemas con preinstalaciones de todo tipo de software de código abierto.

- Facil de instalar
- Facil de crear contenedores
- Mantiene la instalación aislada del resto del SO.
- Muy util para gestionar el tamaño de las instalaciones
- Facil de eliminar contenedores

SGBD Relacionales

UNIVERSIDAD DE MÁLAGA | uma.es

- Introducción
- MySQL
- MariaDB
- Postgres

SGBD Relacionales

UNIVERSIDAD DE MÁLAGA | uma.es

Introducción

a. Modelo relacional
b. SQL

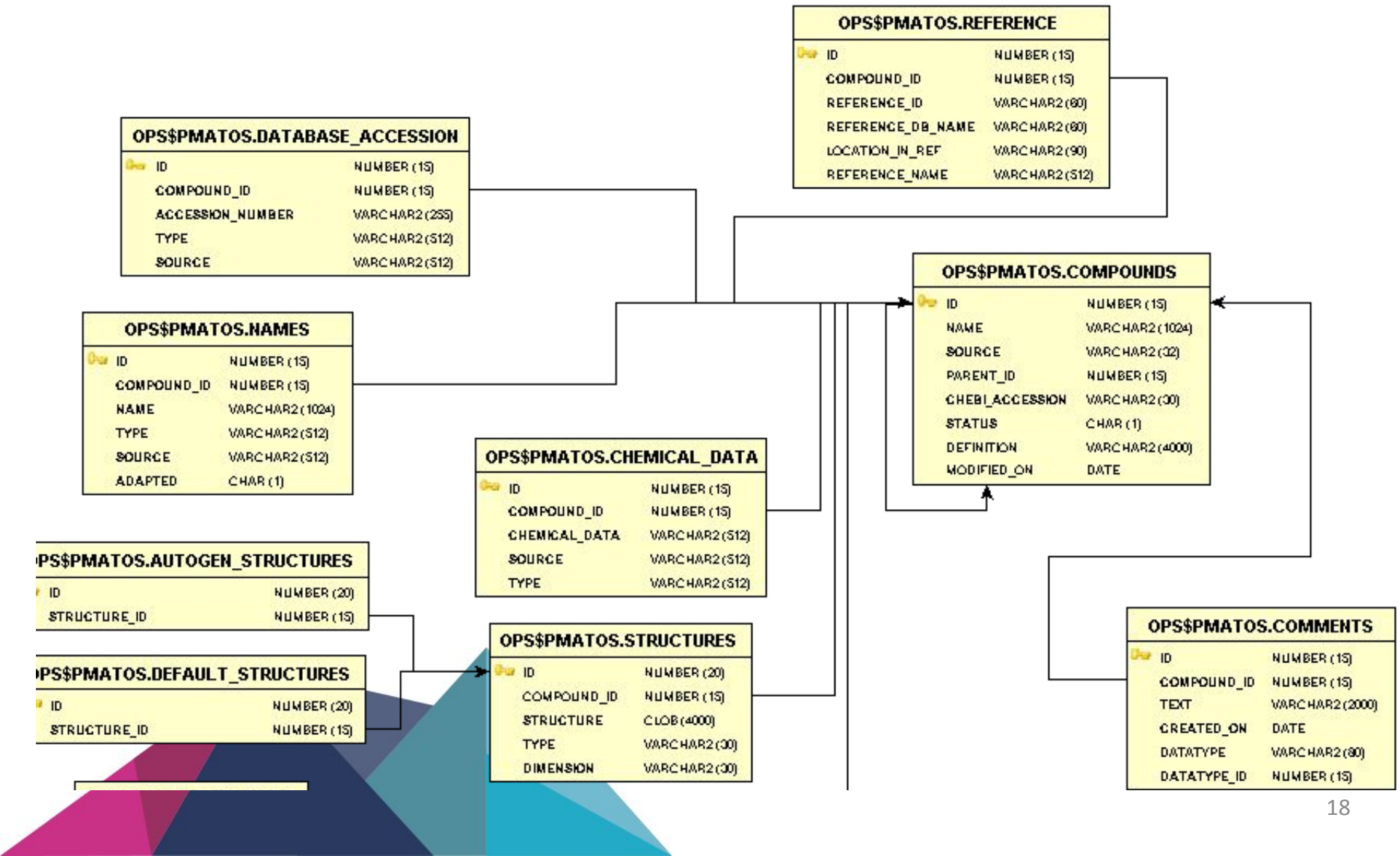
Ventajas

a. Evita duplicidad de registros
b. Integridad referencial
c. Favorece la normalización

Desventajas

a. Deficiencias con datos gráficos
b. No escalables horizontalmente

SGBD Relacionales



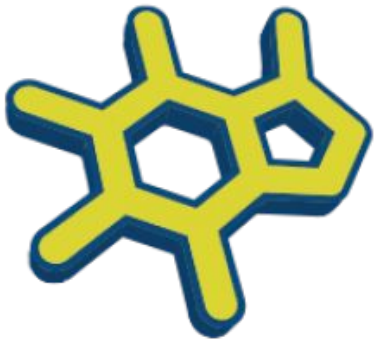
1. ¿Por qué MySQL?
   a. Popular
   b. Oracle
   c. Múltiples motores de almacenamiento

2. Instalación


```
user@localhost$ docker pull mysql
user@localhost$ docker run --name some-mysqldb
-e MYSQL_ROOT_PASSWORD=my-secret-pw -d
mysql:tag
user@localhost$ docker exec -it some-mysql bash
```

3. Despliegue de ChEBI

```
user@localhost$ mysql -u root -p chebi < script.sql
```



5. Optimización

a. Índices:


```
ALTER TABLE `table_name` ADD INDEX (`name`);
```

b. Motor de almacenamiento: MyISAM , InnoDB

```
ALTER TABLE `table_name` ENGINE =`MyISAM`;
ALTER TABLE `table_name` ENGINE =`InnoDB`;
```

1. ¿Por qué MariaDB?
   a. Derivado MySQL
   b. Múltiples motores de almacenamiento
      i. Aria
      ii. XtraDB
   c. Configuración personalizada

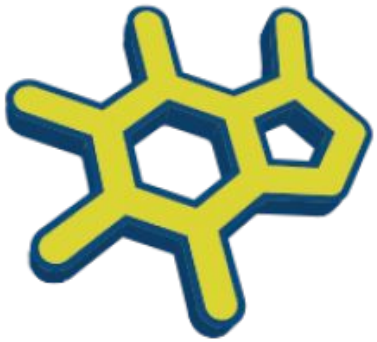
MariaDB

2. Instalación


```
user@localhost$ docker pull mariadb
user@localhost$ docker run --name some-mariadb
-e MYSQL_ROOT_PASSWORD=my-secret-pw -d
mariadb:tag
user@localhost$ docker exec -it some-mariadb
bash
```

3. Despliegue de ChEBI

```
user@localhost$ mysql -u root -p chebi < script.sql
```



5. Optimización

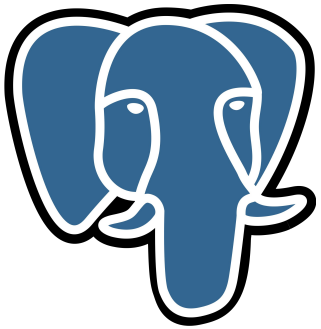
a. Índices:

```
ALTER TABLE `table_name` ADD INDEX (`name`);
```

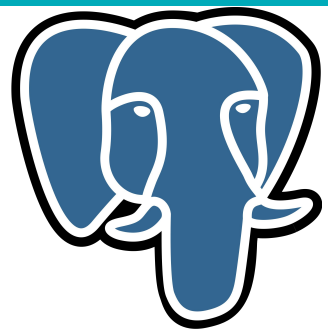
b. Motor de almacenamiento: Aria == Maria

```
ALTER TABLE `table_name` ENGINE =`MARIA`;
```

1. ¿Por qué PostgreSQL?
   a. Gran escalabilidad
   b. Software libre
   c. Estabilidad y confiabilidad
   d. Acceso concurrente multiversión



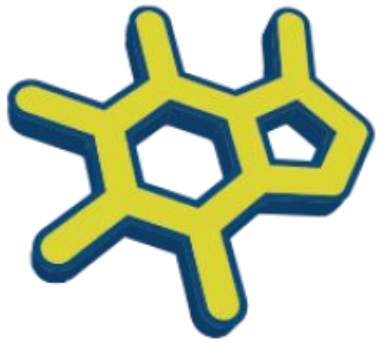
2. Instalación

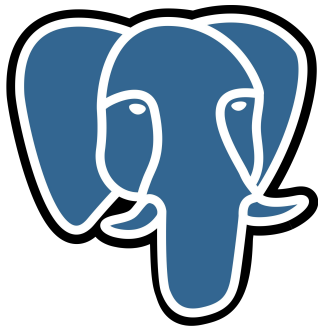


```
user@localhost$ docker pull postgres
user@localhost$ docker run --name some-postgres
-e MYSQL_ROOT_PASSWORD=my-secret-pw -d
postgres:tag
user@localhost$ docker exec -it some-postgres
bash
```

3. Despliegue de ChEBI

```
user@localhost$ postgres psql -f script.sql
chebi
```





5. Optimización

a. Índices:

```
CREATE INDEX index_name ON table_name(atribut);
```

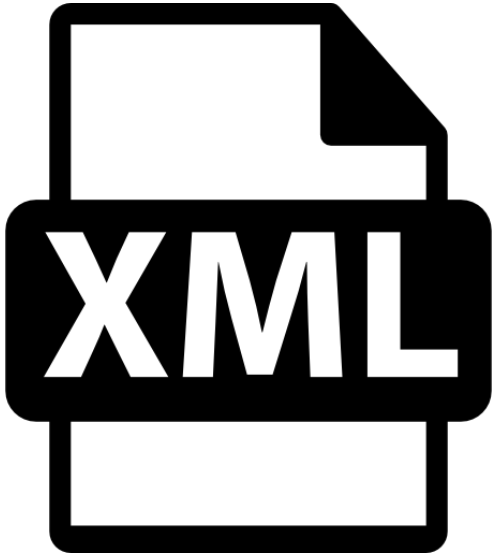
Table engine name(s)	Description	Resume: do we need something like this in PostgreSQL?
InnoDB	Provides index-organized tables where previous versions of rows are displaced into undo tablespace. Secondary indexes are indirect with separate versioning.	Yes. Index-organized tables, undo tablespace and indirect indexes are useful features. It seems, that at least some of them could be implemented using pluggable storage API.
MyISAM	Table engine without transactions and recovery.	No. We have unlogged tables if needed.
CSV, Federated, Cassandra, CONNECT, SphinxSE	These table engines are accessing either local or remote foreign data sources.	No, because we have FDWs.
MGR_Myisam, Merge	These engines allow defining tables which are union of primary tables set. Such union is updatable: updates are pushed down to primary tables.	No, because we have updatable view, partitioning, table inheritance etc.
Archive	Storage for archive data: append-only, compressed storage	Yes, archive storage would be useful.
Blackhole	Table engine which silently "eats" all the inserted data. It have useful usecases in proxy which reduces network traffic in partial replication by filtering replication traffic on master side.	No, our logical decoding already provides such capability. Moreover, there is Blackhole FDW with same functionality.
Mroonga	Mroonga implements non-transactional tables with FTS, geospatial and other indexes.	No, because in PostgreSQL new index types should be implemented as index access methods.
OQGraph	Table engine which allows querying and indexing of graph. OQGraph let users query some kind of view while primary data is stored in another table.	No, because this table engine is not intended to store primary data. If we would need similar solution, we should implement it using index access methods, views, materialized views and so on.
Sequence	This table engine provides functional analogue of generate_series() function.	No, generate_series() is completely sufficient for that purpose.
Aria, XtraDB	Enhanced versions of MyISAM and InnoDB from MariaDB.	No, we should evade fragmentation if possible.
TokuDB, RocksDB	These engines provide write-optimized tables. TokuDB implements fractal trees while RocksDB implements LSM trees.	Yes, write-optimized tables are nice to have.
ScaleDB, Spider	Provide clusters built in the table engines.	No, pluggable storage doesn't seem to be appropriate mechanism for clustering.
Memory	Memory table engine implements non-persistent tables which resides completely in memory. Besides shortcomings of MySQL implementation, in-memory engine could give us following benefits: faster in-memory operations bypass buffer manager; optimized work with disk for persistent in-memory tables due to full data snapshots and row-level WAL.	Yes, it would be nice to have in-memory tables if they would perform faster. Somebody could object that PostgreSQL is on-disk database which shouldn't utilize in-memory storage. But users would be interested in such storage engine if it would give serious performance advantages.

https://wiki.postgresql.org/wiki/Future_of_storage

SGBDs XML

UNIVERSIDAD DE MÁLAGA | uma.es

1. INTRODUCCIÓN
2. EXISTDB
3. VENTAJAS
4. INCONVENIENTES
5. INSTALACIÓN
6. DISEÑO XML
7. DISEÑO CONSULTAS XQUERY



existdb

XML

UNIVERSIDAD DE MÁLAGA | uma.es

1. Lenguaje de marcas.
2. Es extensible.
3. No tiene tags predefinidos.
4. Fácilmente procesable por personas y ordenadores.

eXistDB

UNIVERSIDAD DE MÁLAGA | uma.es

1. Código abierto
2. Emplea XQuery
3. También soporta: JSON y HTML.

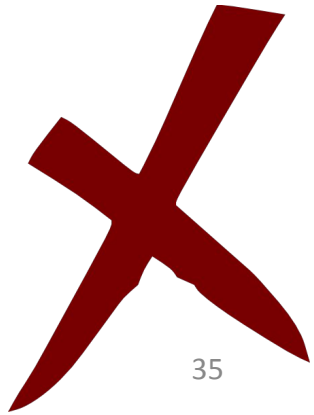
eXistDB

Ventajas

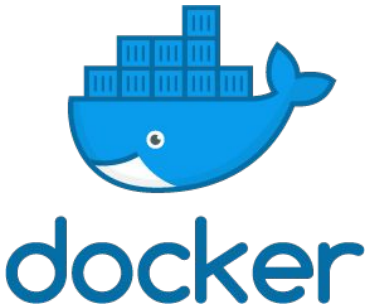
1. Flexibilidad a la hora de modificar diseños existentes.
2. Fácilmente actualizable.
3. Contenido auto descriptivo, no depende de esquemas.

Desventajas

1. Carece de Integridad referencial.
2. Lenguaje de consulta menos profundo/complejo que el lenguaje SQL.
3. Redundancia debido a los tags.



Instalación




```
user@localhost$ docker pull
evolvedbinary/exist-db:eXist-4.1.0
user@localhost$ docker run -it -p 9080:8080
-p 9443:8443
evolvedbinary/exist-db:eXist-4.1.0
user@localhost$ docker exec -it
evolvedbinary/exist-db:eXist-4.1.0
```

Diseño XML

```
<TABLE name="names">
    <ID value="...">
        <NAME>...</NAME>
        <SYNONYM>...</SYNONYM>
        ...
        <IUPAC_NAME>...</IUPAC_NAME>
        ...
    </ID>
    ...
</TABLE>
```

```
<TABLE name="compounds">
    <ID value="...">

<chebi_accession>...</chebi_accession>
        <name>...</name>
        <definition>...</definition>
        <star>...</star>
        <secondary_id>...</secondary_id>
    </ID>
</TABLE>
```

existdb

Diseño Consultas XQUERY

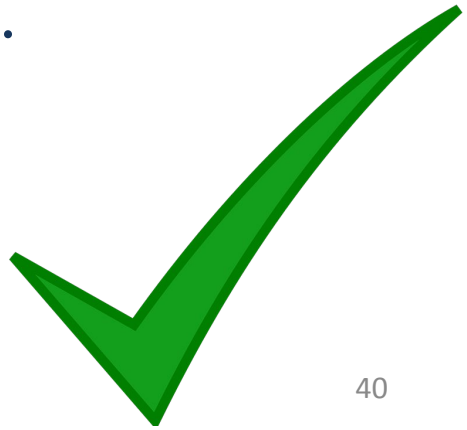
Ejemplo de una consulta simple en XQuery:

```
1   for $x in doc("db/tablasXML/compounds.xml")/table/id
    where $x/name="water" or $x/definition="water"
    return ($x/name,$x/star,$x/chebi_accession)
```

1. Introducción
2. Ventajas
3. Desventajas
4. Sistemas disponibles

Ventajas del NoSQL

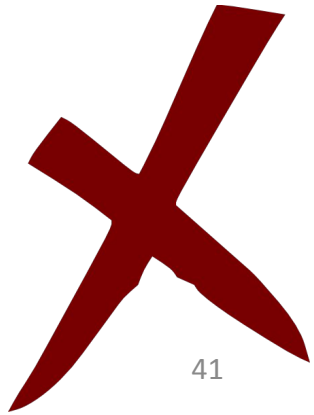
- Estos sistemas responden a las necesidades de escalabilidad horizontal.
- Pueden manejar enormes cantidades de datos.
- No generan cuellos de botella.
- Pueden ejecutarse en clusters de bajo coste.



Desventajas del NoSQL

- Al ser de código abierto, el soporte y la evolución del proyecto pueden ser una desventaja.

- Problemas de compatibilidad. Cada base de datos NoSQL tiene su propia API, las interfaces de consultas son únicas y tienen peculiaridades.



Sistema NoSQL de código abierto **orientado a documentos**.

En lugar de guardar los datos en tablas como se hace en las base de datos relacionales, MongoDB guarda estructuras de datos en documentos **BSON** (similar a JSON).

Despliegue

UNIVERSIDAD DE MÁLAGA | uma.es

Instalación y ejecución en Docker:

```
user@localhost$ docker pull mongo
user@localhost$ docker run --name
chebi_mongo -d mongo:latest
user@localhost$ docker exec -it chebi_mongo
/bin/bash -c mongo
```

Volcado de datos

UNIVERSIDAD DE MÁLAGA | uma.es

Mongo usa **colecciones**, un concepto similar a las tablas en SQL. Para importar datos en colecciones se debe primero crear una BD:

```
> use chebi_basic
> db.users.save({username:”admin”})
```

Volcado de datos

UNIVERSIDAD DE MÁLAGA | uma.es

A continuación, se importan los archivos de datos en formato **TSV**:

```
> mongoimport --db chebi_basic --collection...
```

Esta sintaxis se debe repetir para cada uno de los ficheros a importar, por lo que hemos creado un script en Python para automatizarlo.

Ver en GitHub

Cassandra es una base de datos NoSQL distribuida y basada en un modelo de almacenamiento de «**clave-valor**».

Su objetivo principal es la **escalabilidad lineal** y la **disponibilidad**. Los datos se pueden crear, eliminar y alterar **en tiempo de ejecución** sin bloquear actualizaciones y consultas.

Despliegue

UNIVERSIDAD DE MÁLAGA | uma.es

Instalación y ejecución en Docker:

```
user@localhost$ docker pull cassandra
user@localhost$ docker run --name chebi_cql
-d cassandra:latest
user@localhost$ docker exec -it
chebi_cassandra /bin/bash -c cqlsh
```

Spark

cassandra

Volcado de datos

Cassandra usa **keyspaces** para almacenar los datos.

```
> CREATE KEYSPACE chebi WITH
replication = {'class':
'SimpleStrategy',
'replication_factor': 1};
```

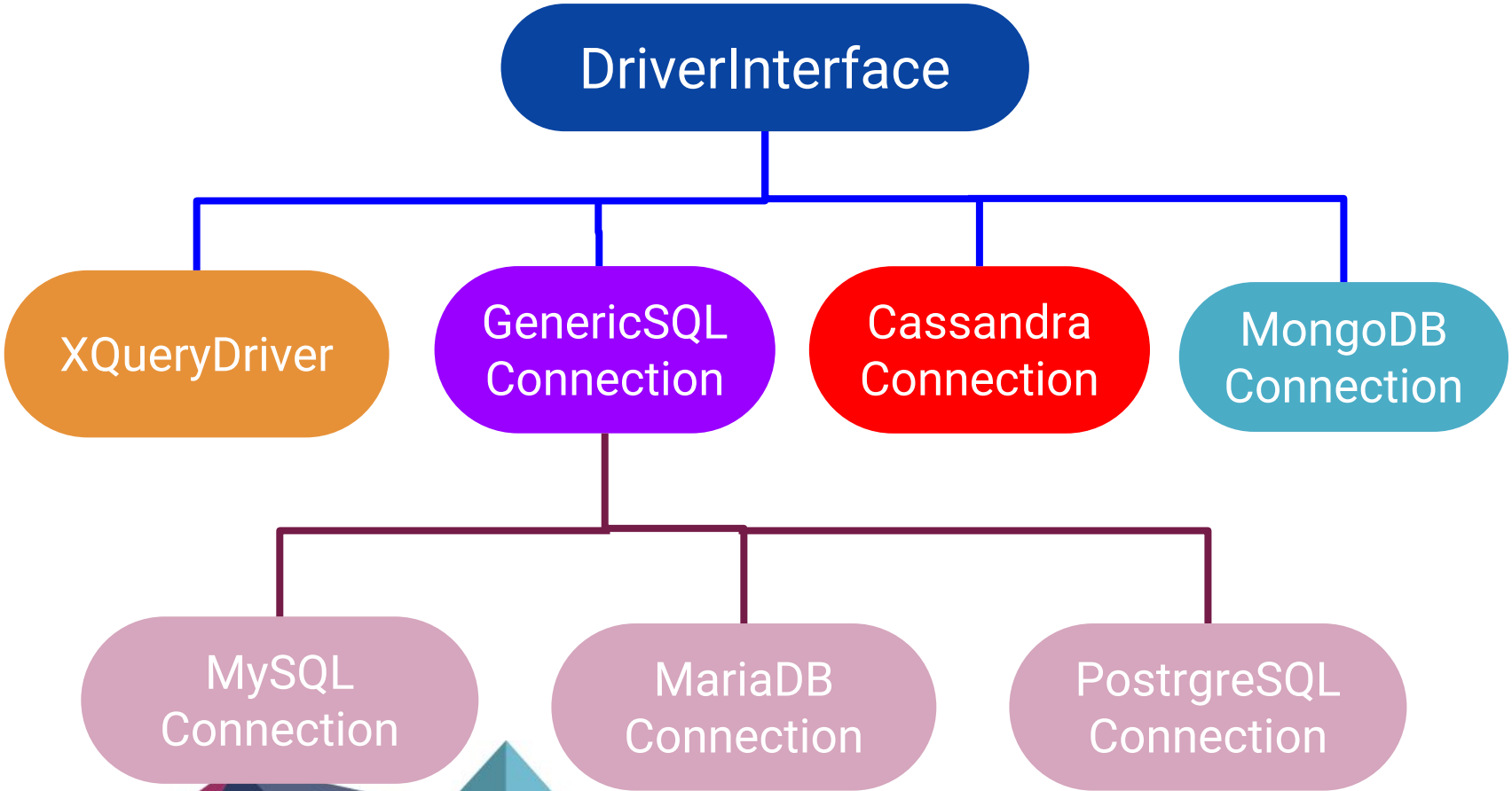
Volcado de datos

UNIVERSIDAD DE MÁLAGA | uma.es

A continuación, se importan los archivos de datos en formato **TSV**:

```
> COPY chebi_basic.chebiId_inchi(CHEBI_ID,
InChI) from 'chebiId_inchi.tsv' with delimiter=
'\t' and header = true;
```

Conexión


DriverInterface
XQueryDriver
GenericSQL Connection
Cassandra Connection
MongoDB Connection
MySQL Connection
MariaDB Connection
PostrgreSQL Connection

Conexión

UNIVERSIDAD DE MÁLAGA | uma.es

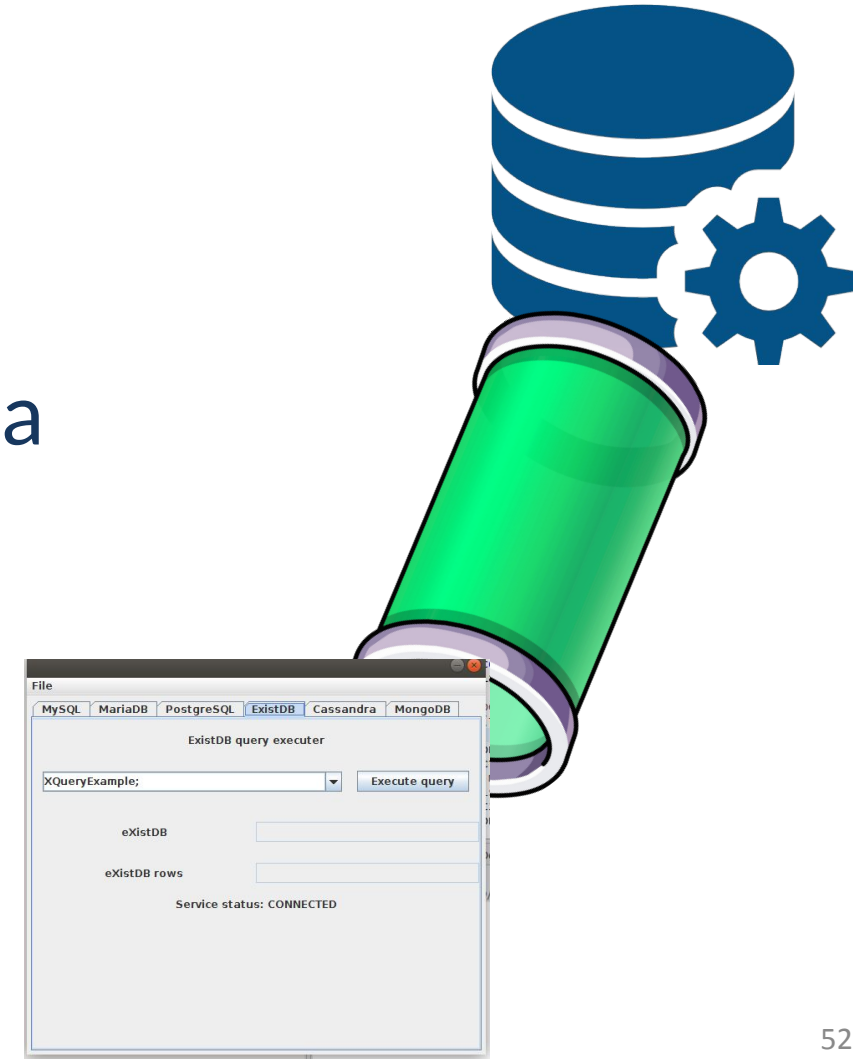
Interfaz

```
public interface DBDriverInterface {
    void executeQuery(String query) throws SQLException, IOException;
    String getTime();
    String getRows();
    boolean hasMoreDB();
    List<String> availableDbs();
    void connect(String db) throws SQLException;
    int dbAdapt(String dbName);
}
```

Conexión

Conectores:

- IP
- Puerto
- Usuario y contraseña
- Definir propiedades
- Ejecutar consulta
- Medir tiempos



4. Resultados

UNIVERSIDAD DE MÁLAGA | uma.es

Consulta compleja

SGBDs	Sín Índices	Con Índices	Con motor	Optimizado
MariaDb	1205 ms	890 ms	1205 ms	600 ms
MySQL	1312 ms	989 ms	1020 ms	799 ms
Postgres	1297 ms	975 ms	---	---

4. Resultados

UNIVERSIDAD DE MÁLAGA | uma.es

Consulta sencilla

SGBDs	Sín Índices	Con Índices	Con motor	Optimizado
MariaDb	350 ms	100 ms	157 ms	22 ms
MySQL	214 ms	113 ms	135 ms	32 ms
Postgres	176 ms	98 ms	---	---
eXistDB	143 ms			
MongoDB	102 ms			
Cassandra	212 ms			

Demo

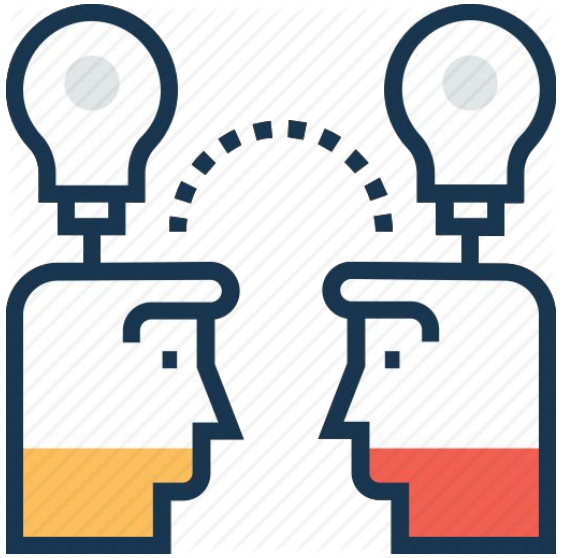
5. Conclusión

- SGBD Relacionales ⟶ MariaDB
  - Optimizaciones
- XML:
  - Intuitiva
- NoSQL:
  - ChEBI: Baja eficiencia
  - Potencial

5. Conclusión

UNIVERSIDAD DE MÁLAGA | uma.es

- Diversidad SGBD
- Otros lenguajes: CQL, XQuery
- GitHub



UNIVERSIDAD DE MÁLAGA

Gracias por

vuestra atención