BASES DE DATOS II

# Práctica I

Grupo Miércoles A

04/02/2024

PABLO MORENO MUÑOZ 841972@unizar.es
ANDREI DUMBRAVA LUCA 844417@unizar.es
GUILLERMO BAJO LABORDA, 842748@unizar.es

Universidad Zaragoza
1542

Escuela de Ingeniería y Arquitectura Universidad Zaragoza

# Índice

# Configuración del entorno de trabajo

## VirtualBox

Se ha decidido instalar y configurar algunos de los diversos sistemas gestores de bases de datos que abordaremos en esta primera práctica de la asignatura en una máquina virtual. Se ha creado la máquina virtual linux sobre VirtualBox, sobre la cual se han instalado posteriormente los distintos SGBD. Se ha optado por utilizar la máquina virtual *VirtualBox CenOs 7.7 Minimal*, cuyo enlace ha sido facilitado por los docentes en el recurso de Moodle. Dado que la instalación mínima nos daba diversos problemas al instalar los SGBDs, finalmente optamos por la *Graphical Desktop Installation*.

Hubo que realizar ciertas modificaciones en la máquina. Primeramente, en *Configuración -> General -> Avanzado*, se escogió la opción de *Bidireccional* tanto en *Portapapeles compartido* como en *Arrastrar y soltar.* Esto facilitaría el copiado y pegado de ciertos comandos desde Windows, lo cual es obviamente opcional pero agiliza el trabajo.

Por otra parte, para poder instalar archivos cómodamente de internet, configuramos el adaptador 1 de red para que estuviera conectado a NAT: *Configuración -> Red -> Adaptador 1 -> Conectado a: NAT*.

Respecto al teclado, para poder usar los carácteres de nuestro teclado con normalidad, se cambió de inglés a español. Para ello: *Applications -> System Tools -> Settings -> Region & Language*, y en *Input Sources* añadimos español y eliminamos inglés. También ponemos en *Formats* “España(Español)”.

Una vez realizada esta configuración, estamos listos para comenzar a instalar los distintos sistemas gestores de bases de datos en la máquina virtual.

## Docker

Para algunas de las otras máquinas se ha decidido usar Docker para instalar y configurar algunos SGBD. La mayoría de estos estarán desplegados sobre esta

plataforma, pues es muy sencillo de utilizar y la mayoría de gestores tienen su propia imagen, la cual podemos utilizar y ejecutar, configurando algunos detalles.

Para utilizar esta aplicación hemos usado la instalación de Docker Desktop para Windows, que nos da una interfaz gráfica muy fácil de utilizar. Además, tenemos una distribución WSL que nos permite usar Linux, por lo que, habiendo descubierto que se podía integrar Docker con WSL, hemos aplicado en la configuración de Docker esta opción para poder utilizar los comandos de Linux para la ejecución de los containers. Para poder ejecutar los contenedores ya hemos tenido que usar las imágenes oficiales de cada SGBD que estarán explicados a continuación.

# Instalación y Administración Básica de los SGBD

## PostgreSql

Paso 1:

Antes de comenzar la instalación, tenemos que tener acceso de administrador en tu sistema

Primero tenemos que actualizar el sistema con el siguiente comando

```
Unset
sudo yum update
```

Paso 2: Instalación de PostgreSQL

Con el siguiente comando conseguimos instalar el repositorio de PostgreSql

```
Unset
sudo yum install -y
https://download.postgresql.org/pub/repos/yum/reporpms/EL-7-x86_6
4/pgdg-redhat-repo-latest.noarch.rpm
```

Después podemos conseguir descargar el SGBD de Postgresql

```
Unset
sudo yum install -y postgresql14-server
```

Posteriormente activamos el servicio de la base de datos y activamos el inicio automático de PostgreSql

```
Unset
sudo /usr/pgsql-14/bin/postgresql-14-setup initdb
sudo systemctl enable postgresql-14
sudo systemctl start postgresql-14
```

Paso 3: Configuración Inicial

PostgreSQL se instala con una configuración predeterminada que incluye un usuario llamado "postgres". Puedes acceder a la línea de comandos de PostgreSQL con este usuario ejecutando:

```
Unset
sudo -u postgres psql
```

Ahora mismo que estamos dentro de la base de datos con el usuario postgre

Paso 4: Comandos de creación y modificación de la base de datos

Hemos realizado un script '.sql' con un conjunto de comandos que permite crear un nuevo usuario y una base de datos, crear tablas y poblarlas con algún valor y realizar una consulta

Finalmente podemos configurar el servidor para asegurarnos una mayor seguridad permitiendo el acceso remoto desde ordenadores conectados a la misma red local.

Primero, se ha editado el archivo de configuración /var/lib/pgsql/14/data/postgresql.conf:

```
listen_addresses = 'localhost'
```

Se reiniciar el servicio PostgreSQL mediante:

```
Unset
sudo systemctl restart postgresql-14
```

## Oracle XE

Para la instalación y configuración del SGBD Oracle XE, se han utilizado contenedores de Docker. Inicialmente, se optó por descargar la imagen oficial desde Docker Desktop y runnearla, pero esto dió diversidad de problemas, por lo que finalmente se procedió a hacerlo mediante la terminal WSL, como mostraremos en los pasos a continuación.

Paso 1:
Primeramente, se ha de abrir una terminal, e introducir el siguiente comando para hacer pull a la imagen oficial de Oracle XE en docker:

```
Unset
docker pull container-registry.oracle.com/database/express:latest
```

Paso 2:
Esperamos a que se complete el comando anterior, y ya podemos ejecutar el siguiente comando que correrá la imagen y creará un contenedor con el nombre que le indiquemos como parámetro.

```
Unset
docker run -d --name <nombre-db>
container-registry.oracle.com/database/express:21.3.0-xe
```

Paso 3:

Para poder empezar a introducir sentencias SQL y poder empezar a utilizar la imagen de Oracle XE que acabamos de ejecutar, ejecutamos el siguiente comando:

```
Unset
docker exec -it mibaseoracle sqlplus / as sysdba
```

Llegados a este punto ya se puede crear una base de datos e interactuar con la misma, así como distintos usuarios y roles e incluso un superusuario. Junto a esta memoria se ha entregado un script *Comandos_OracleXE.sql* con diversas sentencias de creación, actualización, borrado, y más con las que se ha verificado el correcto funcionamiento del SGBD. Tambien se incluye *Borrar_Base_Ejemplo_OracleXE.sql*

para borrar las tablas, usuarios y roles creados en el ejemplo. Se comentará esto más en detalle en la sección de Generación de Datos y Pruebas.

Finalmente, se procede a comprobar que se permite el acceso remoto a los datos desde otro ordenador conectado a la misma red local. Para ello, se ha instalado Oracle SQL Developer. En el navegador de conexiones, se hace click derecho en *Connections* y seleccionamos *New Database Connection.* Introducimos los siguientes parámetros: nombre de conexión *XE*, nombre de usuario *pr1-oracle-xe*, contraseña *pr1-oracle-xe-psswd*, nombre de host *127.0.0.1*, puerto *1521* default y SID *XE*. Clicamos en *Connect* y se abre el entorno de trabajo de SQL, desde el que podemos comenzar a introducir las sentencias SQL deseadas.

## Cassandra

Para descargar este SGBD hemos usado la imagen oficial de ésta misma en Docker.

Ahora vamos a describir paso a paso lo que se ha realizado para configurarlo todo:

Paso 1:

Lanzamiento en la terminal de los comandos para iniciar la ejecución del contenedor

```
Unset
docker pull cassandra
docker run --name cassandra -p 9042:9042 --rm --name dbeaver-cassandra -d cassandra
## Entramos dentro del server
docker exec -ti dbeaver-cassandra bash
```

Paso 2: Configuración del Authenticator y Authorizer

Dentro de la máquina, se ha descargado un editor de texto como nano y se ha editado */etc/cassandra/cassandra.yaml* para cambiar el Authenticator a PasswordAuthenticator y el Authorizer a CassandraAuthorizer

```
Unset
apt-get update
apt-get install nano
## Editamos los auth
nano /etc/cassandra/cassandra.yaml
exit
## Reiniciamos el contenedor
docker restart dbeaver-cassandra
```

Paso 3: Conexión remota del cliente SQL

Se ha descargado un cliente como DBeaver para poder conectarse a la base de datos del contenedor. Se ha usado una licencia de estudiante y se ha realizado una conexión a la base con keyspace : system, ip: *127.0.0.1*, puerto: 9042 y usuario y contraseña: dbeaver-cassandra. Con esta configuración podríamos usar este cliente y lanzar sentencias SQL a nuestro sistema cassandra. Posteriormente lanzaremos sentencias para crear nuevos usuarios, keyspaces, tablas, etc.

Paso 4: Conexión desde Docker

Estando dentro de Docker, podemos lanzar cqlsh -u cassandra -p cassandra y realizar las mismas sentencias y operaciones, y produciremos los mismos resultados. Por lo que ya la decisión está en manos de la persona que la use.

Paso 5: Creación de usuarios con roles, keyspaces,…

Para realizar estas funciones vamos a crear un script que hace varias de estas cosas de forma automática, y que si ejecutamos sabemos que funcionará.

```
Unset
#!/bin/bash

# Crear un superusuario
docker exec -it dbeaver-cassandra cqlsh -u cassandra -p cassandra -e "CREATE USER new_superuser WITH PASSWORD 'new_password' SUPERUSER;"

# Crear la estructura básica del espacio de datos
docker exec -it dbeaver-cassandra cqlsh -u new_superuser -p new_password -e "CREATE KEYSPACE mykeyspace WITH replication = {'class':'SimpleStrategy', 'replication_factor' : 1};"
docker exec -it dbeaver-cassandra cqlsh -u new_superuser -p new_password -e "USE mykeyspace; CREATE TABLE Car( id uuid PRIMARY KEY, model text, description text, color text);"

# Crear usuarios y roles con distinto acceso
docker exec -it dbeaver-cassandra cqlsh -u new_superuser -p new_password -e "CREATE ROLE reader WITH PASSWORD = 'reader_password' AND LOGIN = true;"
docker exec -it dbeaver-cassandra cqlsh -u new_superuser -p new_password -e "CREATE ROLE writer WITH PASSWORD = 'writer_password' AND LOGIN = true;"
docker exec -it dbeaver-cassandra cqlsh -u new_superuser -p new_password -e "GRANT MODIFY ON KEYSPACE mykeyspace TO writer;"
docker exec -it dbeaver-cassandra cqlsh -u new_superuser -p new_password -e "GRANT SELECT ON KEYSPACE mykeyspace TO reader;"

# Probar los roles
echo "INSERT INTO mykeyspace.Car (id, model, description, color) VALUES (uuid(), 'Model S', 'Tesla Model S', 'Red');" | docker exec -i dbeaver-cassandra cqlsh -u writer -p writer_password
echo "SELECT * FROM mykeyspace.Car;" | docker exec -i dbeaver-cassandra cqlsh -u reader -p reader_password

# Borrar todo
```

```
docker exec -it dbeaver-cassandra cqlsh -u new_superuser -p new_password -e "DROP KEYSPACE mykeyspace;"
docker exec -it dbeaver-cassandra cqlsh -u new_superuser -p new_password -e "DROP ROLE reader;"
docker exec -it dbeaver-cassandra cqlsh -u new_superuser -p new_password -e "DROP ROLE writer;"
docker exec -it dbeaver-cassandra cqlsh -u dbeaver-cassandra -p dbeaver-cassandra -e "DROP USER new_superuser;"
```

**Se ha incluido el docker exec para poder lanzarlo desde fuera del contenedor, si se desea realizar desde dentro, ejecutar lo mismo sin esto. Si se está en DBeaver, ejecutar tan sólo las sentencias que están entre “”.*

## IBM DB2

Paso 1: Descarga del Instalador:

Se inicia el proceso de instalación de IBM DB2 accediendo al sitio web oficial de IBM y obteniendo el instalador tras crear una cuenta.

Paso 2 Descompresión del Archivo Tar:

Una vez descargado, el archivo tar que contiene el instalador se descomprime utilizando el siguiente comando:

```
Unset
tar -xvf v11.5.9_linuxx64_server_dec.tar
```

Paso 3: Verificación de Prerrequisitos:

Se ejecuta el script db2prereqcheck con el parámetro -v seguido de la versión correspondiente, por ejemplo, 11.5.0.0, para confirmar que se cumplen todos los prerrequisitos necesarios para la instalación.

```
Unset
./db2prereqcheck -v 11.5.0.0
```

Paso 4: Instalación de Paquetes Necesarios:

Se procede a instalar los paquetes necesarios utilizando el administrador de paquetes de Cent0s 7.
Se ejecuta el comando sudo yum install <nombre-paquete>.

Paso 5: Re-verificación de Prerrequisitos:

Se realiza una segunda verificación de prerrequisitos para asegurarse de que todos los componentes necesarios estén instalados correctamente.

Paso 6: Ejecución del Instalador DB2:

Se ejecuta el script db2_install para comenzar el proceso de instalación de IBM DB2. Durante este proceso, se solicitará al usuario que acepte los términos de la licencia,

especifique el directorio de instalación y seleccione las opciones de configuración deseadas.

Paso 7: Inicio del Servicio DB2:

Se inicia el servicio de IBM DB2  escribiendo en la terminal el comando “db2”, lo que permite el acceso y la gestión de la base de datos.

## HBASE Y HADOOP

Para realizar la configuración de este SGBD, se ha intentado seguir las instrucciones proporcionadas en uno de los documentos de la práctica, pero el resultado ha sido un error que no llegaba a nuestra comprensión. Hemos intentado seguir otros tutoriales desde Internet para instalarlo en la máquina virtual de centOs 7, pero no funcionaba tampoco, por lo que he seguido un tutorial de GitHub de una imagen de Docker que está formada para lanzar tres nodos, uno master y dos slaves, que crean el cluster con Hadoop, y configuran todo lo necesario mediante scripts, permitiendo lanzar los nodos y conecta hbase correctamente, por lo que de esta forma se ha podido usar el SGBD, pero no se ha logrado acceder a él desde fuera de Docker, esa tarea queda pendiente, pero la base de datos se puede usar, ahora vamos a mostrar los pasos y probar una serie de consultas.

Paso 1: Clonar repositorio y construir imágenes del máster y los slaves:

```
Unset
git clone https://github.com/krejcmat/hadoop-hbase-docker.git
cd hadoop-hbase-docker
./build-image.sh hadoop-hbase-base
```

Paso 2: Iniciar contenedores y empezar Hadoop

```
Unset

./start-container.sh latest 3
## Check members of cluster
serf members
cd ~
./configure-members.sh
./start-hadoop.sh
```

Paso 3: Iniciar Hbase y esperar un momento a que se configure todo (es probable que al principio no funcione correctamente)

```
Unset
./start-hbase.sh
```

Con este script se habrá generado correctamente toda la configuración y se lanzará automáticamente el hbase shell, pudiendo añadir tablas y consultarlas.

Paso 4: Gestionar roles y permisos:

Para realizar esta acción, se necesitan añadir características de seguridad a los archivos de configuración de Hbase. Al hacerlo, todo esto se puede gestionar con comandos como estos:

```
Unset
create 'hbase:ejemplo', 'info'
put 'hbase:ejemplo', 'bbdd2', 'info:password', 'contraseña'
# Asignar permisos de solo lectura
grant 'bbdd2', 'R', 'mi_tabla'
```

# Generación de Datos y Pruebas

## PostgreSQL:

Existen varias estrategias y herramientas para la generación de datos en PostgreSQL.
1. Generadores de datos automáticos: Existen herramientas específicamente diseñadas para generar datos de forma automática.

- pgbench: Es una herramienta de benchmarking que viene con PostgreSQL. Aunque su enfoque principal es evaluar el rendimiento de PostgreSQL, también puede utilizarse para generar datos de prueba. Permite simular cargas de trabajo de múltiples usuarios y generar datos aleatorios para tablas predefinidas.

2. Herramientas de generación de datos personalizadas: Puedes crear tus propias herramientas o scripts para generar datos según tus necesidades específicas.

3. Herramientas de generación de datos de terceros: Además de las herramientas específicas para PostgreSQL, también hay herramientas de terceros que pueden generar datos para diferentes tipos de bases de datos, incluyendo PostgreSQL.

- DataGenerator: Es una herramienta gratuita y de código abierto que puede generar datos de prueba para una variedad de bases de datos, incluyendo PostgreSQL. Permite definir esquemas y configurar reglas para generar datos aleatorios.

## Oracle:

Nuevamente, para la generación de datos en Oracle existen diversas estrategias. Algunas de las principales son las siguientes:

1. Oracle Data Generator: Oracle ofrece una herramienta llamada Oracle Data Generator que puede generar grandes cantidades de datos de manera automatizada. Esta herramienta puede ser utilizada para generar datos de prueba para tablas predefinidas o personalizadas. A su vez, permite simular cargas de trabajo de múltiples usuarios y generar datos aleatorios según las especificaciones proporcionadas.

2. Herramientas de generación de datos personalizadas:
Se pueden crear herramientas propias o scripts para generar datos en Oracle según las necesidades específicas. Estos scripts pueden utilizar sentencias SQL o lenguajes de programación como PL/SQL para insertar datos en las tablas de la base de datos.

3. Herramientas de generación de datos de terceros:

Además de las herramientas específicas para Oracle, también existen herramientas de terceros que pueden generar datos para bases de datos Oracle.

- DataFactory: Es una herramienta comercial que proporciona funcionalidades avanzadas para la generación de datos en Oracle y otras bases de datos. Permite definir reglas de generación de datos y generar datos aleatorios para tablas y columnas específicas.

## IBM DB2:

Para la generación de datos en IBM Db2, también hay varias estrategias y herramientas disponibles.
1. Herramientas de generación de datos incorporadas en Db2:

- db2generate: Db2 proporciona una utilidad llamada db2generate que se puede utilizar para generar datos de prueba. Esta utilidad puede crear tablas y cargarlas con datos de prueba basados en plantillas de generación de datos predefinidas.

2. Herramientas de generación de datos personalizadas:

- Scripts SQL personalizados: Al igual que con PostgreSQL, puedes escribir scripts SQL personalizados que generen datos

3. Herramientas de generación de datos de terceros:

- dbForge Data Generator for DB2: Es una herramienta comercial que proporciona una interfaz gráfica de usuario para generar datos de prueba para Db2. Permite definir reglas de generación de datos, y ofrece opciones avanzadas para la personalización.

## Cassandra:

Para este SGBD se ha creado un .cql que se introducirá con el comando:

```
Unset
cqlsh -u new_superuser -p new_password -f cassandra_poblado_script.cql
```

*cassandra_poblado_script.cql* tiene la siguiente estructura:

```
Unset
CREATE KEYSPACE IF NOT EXISTS test_keyspace WITH replication = {'class':
'SimpleStrategy', 'replication_factor': '1'};

USE test_keyspace;

CREATE TABLE IF NOT EXISTS test_table (
  id INT PRIMARY KEY,
  name TEXT
);

INSERT INTO test_table (id, name) VALUES (1, 'John');
INSERT INTO test_table (id, name) VALUES (2, 'Alice');

SELECT * FROM test_table;

DROP TABLE IF EXISTS test_table;

DROP KEYSPACE IF EXISTS test_keyspace;
```

Para esta SGBD tenemos varias formas de ejecutar el poblado de datos y consultas: Una de ellas es mediante un script como este, que se puede ejecutar desde fuera o iniciando sesión con cqlsh y ejecutando el script desde la terminal de cql. Además, teniendo la conexión a DBeaver se ha podido ejecutar también este script desde la aplicación, produciendo el mismo resultado. Otras formas que se pueden utilizar son la utilización de código usando distintas librerías para realizar la conexión y ejecutar las consultas.

## Hbase y Hadoop:

Para generar sentencias de Hbase para ejecutar hemos creado un script que introduce una tabla y varios valores, que ejecutamos desde fuera con hbase shell, hay que tener en cuenta que ejecuta cada comando por separado, por lo que puede tardar algo de tiempo.

```
Unset
echo "create 'alumnos', 'info'" | hbase shell

echo "put 'alumnos', 'andrei', 'info:nombre', 'Andrei'" | hbase shell
echo "put 'alumnos', 'andrei', 'info:edad', '20'" | hbase shell

echo "put 'alumnos', 'guillermo', 'info:nombre', 'Guillermo'" | hbase shell
echo "put 'alumnos', 'guillermo', 'info:edad', '20'" | hbase shell
```

```
echo "put 'alumnos', 'pablo', 'info:nombre', 'Pablo'" | hbase shell
echo "put 'alumnos', 'pablo', 'info:edad', '20'" | hbase shell

echo "scan 'alumnos'" | hbase shell
echo "get 'alumnos', 'andrei'" | hbase shell
echo "get 'alumnos', 'guillermo'" | hbase shell
echo "get 'alumnos', 'pablo'" | hbase shell

echo "disable 'alumnos'" | hbase shell
echo "drop 'alumnos'" | hbase shell
```

**Al final se eliminan las tablas para poder volver a ejecutar cada comando*

# Comentarios acerca de las licencias

## PostgreSQL:

PostgreSQL utiliza una licencia de código abierto llamada PostgreSQL License, que es una licencia permisiva similar a la licencia BSD. Esto significa que PostgreSQL es de uso gratuito y está disponible para su uso y modificación, tanto para fines comerciales como no comerciales. Los usuarios pueden utilizar PostgreSQL sin tener que pagar regalías ni tarifas de licencia.
Además, la licencia permite a los usuarios modificar el código fuente y distribuir las versiones modificadas, siempre y cuando se cumplan ciertas condiciones, como la inclusión de un aviso de copyright y la conservación de la licencia original.

## Oracle:

Las licencias de Oracle XE son fundamentales para establecer los términos y condiciones bajo los cuales se puede utilizar este software. Oracle XE está diseñado para ofrecer una opción gratuita para desarrolladores y empresas que buscan una solución de base de datos robusta pero de bajo costo. Una de las principales ventajas de Oracle XE es su disponibilidad gratuita para uso en entornos de desarrollo y distribución de aplicaciones. Esto permite a los desarrolladores utilizar la base de datos Oracle para crear y probar aplicaciones sin incurrir en costos de licencia.

Sin embargo, Oracle XE tiene limitaciones en cuanto a los recursos de la base de datos. Por ejemplo, puede haber restricciones en la cantidad máxima de memoria y capacidad de almacenamiento de datos que puede utilizar la base de datos. Además, las licencias de Oracle XE pueden incluir restricciones sobre su uso en entornos comerciales. Aunque el software es de uso gratis, puede haber limitaciones en su uso para aplicaciones comerciales, como el tamaño de la empresa o los ingresos generados por la aplicación. También es común encontrar cláusulas que prohíben el uso de Oracle XE en aplicaciones críticas para la empresa o aquellas que requieren un alto nivel de disponibilidad y rendimiento.

Es esencial revisar detenidamente las licencias de Oracle XE para comprender completamente los términos y condiciones de uso. Esto ayudará a evitar posibles problemas legales y garantizará un uso adecuado del software dentro de los límites establecidos por Oracle.

## IBM db2

IBM Db2, como producto comercial de IBM, está sujeto a una licencia propietaria. Esto significa que su uso está regulado por términos y condiciones establecidos por IBM, y generalmente implica el pago de una licencia o tarifa por parte de los

usuarios para utilizar el software. Los detalles específicos de la licencia de IBM Db2 pueden variar según la versión y el acuerdo específico con IBM. Se recomienda revisar la documentación oficial de IBM Db2 o consultar con un representante de ventas de IBM para obtener información precisa sobre las licencias aplicables.

## Cassandra:

Apache Cassandra está disponible bajo la licencia de Software Apache v2.0 y que es supervisada por un comité de gestión de proyecto (PMC), que orienta sus operaciones diarias, incluyendo versiones de desarrollo y producto de la comunidad.

La licencia Apache otorga permisos amplios para usar, modificar y usar el software, pues es una aplicación de free license. Al ser una licencia bastante sencilla, la responsabilidad del titular está bastante limitada en caso de daños causados por el software. También ofrece mucha flexibilidad en los esquemas de datos y un alto rendimiento, además de tolerancia a fallos, pues se pueden replicar datos de forma eficiente entre nodos.

## Apache HBase y Hadoop:

Hbase está disponible bajo la licencia de Software Apache v2.0, por lo que ofrece funcionalidades parecidas a las de Cassandra, pues sigue la misma licencia. Hadoop también la emplea, pero incluye componentes de terceros que tienen sus propias licencias.

# Esfuerzos invertidos

*Todas las horas de cada base de datos incluyen el tiempo de lectura de documentación, instalación de los SGBD y las pruebas pertinentes.

| | Miembros del grupo | | |
|---|---|---|---|
| Tarea | Guillermo Bajo | Andrei Dumbrava | Pablo Moreno |
| Hadoop y Hbase | 30min | 5h | 2h |
| Cassandra | 1h | 2h | 30min |
| Ibm DB2 | 30min | 1h | 3h |
| Postgresql | 3h | 2h | 3h |
| Oracle | 4h | 1h | 30min |
| Memoria | 3h | 3h | 3h |
| **Total** | **12h** | **14h** | **12h** |