Database (DB)

A database is an organized collection of data, generally stored and accessed electronically from a computer.

Relational Database: A relational database organizes data into tables which can be linked (related) based on data common to each.

Database Management System (DBMS)

A DBMS is a special software program designed to define, manipulate, retrieve and mangage data in database. It is used to manage large amounts of information, handle security aspects and backups, importing/exporting data. The DBMS interacts with the Database and the application. Examples of DBMS are MySQL, PostgreSQL, MSSQL and Oracle Database.

SQL

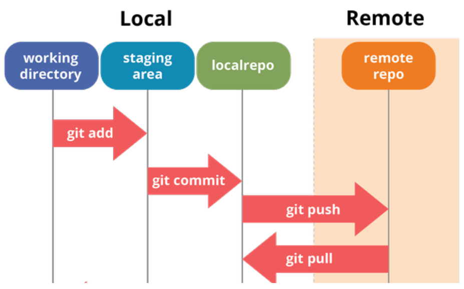
SQL (Structured Query Language) is a standardized language for interaction with the RDBMS (relational database management system). SQL is used to perfrom CRUD (create, retrieve, updata and delete) operations, as well as other administrative tasks (user management, security, backups). SQL is also used to define tables and structures. SQL code used on one RDBMS is not always portable to another without modifications.

Oracle

blalbalb

https://www.youtube.com/watch?v=rPfKXVmYuyg

Basic Terminology



Oracle Database 18.4.0 Express Edition (XE)

Build OracleDB Docker Image

The image for the Oracle Database is not available on DockerHub (only older versions are available and require login to Docker.

You need to clone the github repo on run a script that creates the Docker image.

```
git clone https://github.com/oracle/docker-images
cd ~/docker-images/OracleDatabase/SingleInstance/dockerfiles
./buildDockerImage.sh -v 18.4.0 -x
```

Clone the Repo, change directory and run the bash script to build an image for the Oracle database (version 18.4.0). The -x specifies that the image should be based on the free Express Edition.

Run OracleDB Docker Container

```
chmod o+w [project_folder]
```

In order to use volumes for database container you need to change the permissions of the project folder.

```
docker run -d --name OracleDB-18.4.0-xe -p 1521:1521 -p
5500:5500 -e ORACLE_PWD=example_password -v
`pwd`:/opt/oracle/oradata oracle/database:18.4.0-xe
```

Example for running a container in detached mode (-d) with volume mounted (-v).

```
docker run --name [container_name] \
-p [host_port]:1521 -p [host_port]:5500 \
-e ORACLE_SID=[your_SID] \
-e ORACLE_PDB=[your_PDB name] \
-e ORACLE_PWD=[your_database_passwords] \
-e ORACLE_CHARACTERSET=[your_character_set]  \
-v [<host mount point]:/opt/oracle/oradata \
oracle/database:18.4.0-xe
Parameters:
--name:          The name of the container (default: auto
                 generated)

-p:              The port mapping of the host port to the
                 container port. Two ports are exposed:
                 1521 (Oracle Listener), 5500 (OEM Express)

-e ORACLE_SID:   The Oracle Database SID (database name) that
                 should be used (default: XE)

-e ORACLE_PDB:   The Oracle Database PDB name that should be
                 used (default: XEPDB1)

-e ORACLE_PWD:   The Oracle Database SYS, SYSTEM and PDB_ADMIN
                 password (default: auto generated)

-e ORACLE_CHARACTERSET:    The character set to use when
                           creating the database
                           (default: AL32UTF8)

-v               The data volume to use for the database.
                 Has to be owned by the Unix user "oracle" or
                 set appropriately.
                 If omitted the database will not be persisted
                 over container recreation.
```

How to use the different parameters and flags when running a OracleDB container

```
# bash terminal (host machine)
docker exec -it [container_name] bash
# bash terminal (container)
tail -f tail -f /opt/oracle/cfgtoollogs/dbca/XE/XE.log
```

If the container is run in detached mode (-d) you can look at the progress of creating the DB container by using the command from above.

Connect to OracleDB from inside the Container

```
docker exec -it [container_name] bash
```

Open a bash terminal inside the container (command is executed on host machine)

```
# bash terminal (container)
sqlplus sys/example_password@//localhost:1521/XE as sysdba
```

Connect to OracleDB (don't forget to use your own password)

```
docker exec -it [container_name] sqlplus \
sys/[password]@//localhost:1521/XE as sysdba
```

With this command you can directly go to sqlplus, without first connecting to the container.

Connect to OracleDB Docker Container from another Container

You can use python (cx_Oracle) to connect to a OracleDB and run queries. In order to connect to the OracleDB you need Oracle Instant Client installed in the container from which you want to connect to the OracleDB container.

Thus, for a development session you need two containers, one on which the database is running and one in which the development will take place.

Docker Compose can be used to get those two containers running.

```
Chmod o+w [project_folder]
```

Change Permission in order to use volumes

Dockerfile

```
# Dockerfile:

# use ubuntu18.04 with CUDA10.1 as base image
FROM nvidia/cuda:10.1-base-ubuntu18.04

# install anaconda3 (dockerfile from continuumio/anaconda3
ENV LANG=C.UTF-8 LC_ALL=C.UTF-8
ENV PATH /opt/conda/bin:$PATH

RUN apt-get update --fix-missing && apt-get install -y wget
bzip2 ca-certificates \
    libglib2.0-0 libxext6 libsm6 libxrender1 \
    git mercurial subversion

RUN wget --quiet https://repo.anaconda.com/archive/Anaconda3-
2020.02-Linux-x86_64.sh -O ~/anaconda.sh && \
    /bin/bash ~/anaconda.sh -b -p /opt/conda && \
    rm ~/anaconda.sh && \
    ln -s /opt/conda/etc/profile.d/conda.sh
/etc/profile.d/conda.sh && \
    echo ". /opt/conda/etc/profile.d/conda.sh" >> ~/.bashrc
&& \
    echo "conda activate base" >> ~/.bashrc

RUN apt-get install -y curl grep sed dpkg && \
    TINI_VERSION=`curl
https://github.com/krallin/tini/releases/latest | grep -o
"/v.*\"" | sed 's:^..\(.*\).$:\1:'` && \
    curl -L "https://github.com/krallin/tini/releases/download/
v${TINI_VERSION}/tini_${TINI_VERSION}.deb" > tini.deb && \
    dpkg -i tini.deb && \
    rm tini.deb && \
    apt-get clean

# install oracle instant client
#https://www.youtube.com/watch?v=_wab4By7P78

RUN apt-get update && apt-get install -y libaio1 wget
RUN apt-get install -y unzip

WORKDIR /opt/oracle
RUN wget
https://download.oracle.com/otn_software/linux/instantclient/19
600/instantclient-basic-linux.x64-19.6.0.0.0dbru.zip && \
     unzip instantclient-basic-linux.x64-19.6.0.0.0dbru.zip
&& \
    rm -f instantclient-basic-linux.x64-19.6.0.0.0dbru.zip
RUN cd /opt/oracle/instantclient_19_6 && rm -f *jdbc* *occi*
*mysql* *README *jar uidrvci genezi adrci
RUN echo /opt/oracle/instantclient_19_6 >
/etc/ld.so.conf.d/oracle-instantclient.conf && ldconfig

#RUN python -m pip install cx_Oracle # already specified in
requirements.txt

ENTRYPOINT [ "/usr/bin/tini", "--" ]


# copy requirements file
COPY requirements.txt /project/
# set working directory
WORKDIR /project/



# install dependencies
RUN pip install -r requirements.txt

# copy the project
COPY . /project/

# open jupyter notebook
CMD jupyter-notebook --ip=0.0.0.0 --allow-root
```

This Dockfile uses the Ubuntu18.04 with CUDA10.1 as a base image and installs Anaconda3, Oracle Instant Client and installs the dependencies (make sure that the requirements file includes cx_Oracle).

Thus, the Image create from this dockerfile can be used to run Development containers with access to jupyter notebooks, gpu acceleration and to a OracleDB.

```
Docker build -t [Dev_image_name] .
```

Builds an image from the Dockerfile. The -t is used to specify the name of the image.

Docker-compose.yml and db.env file

```
# docker-compose.yml
version: '2'
services:
        database:
                image: [OracleDB_image_name]
                # volumes:
                #        - `pwd`:/opt/oracle/oradata

                ports:
                        - 1521:1521
                        - 5500:5500

                # environment:
                #        - ORACLE_PWD=[your_password]
                env_file:
                         - db.env

        DevEnv:
                image: [Dev_image_name]
                volumes:
                        - ./Dev/:/project

                ports:
                        - 8888:8888
                depends_on:
                        - database
```

Using a docker-compose.yml file you can start multiple containers. The service “DevEnv” in this case is based on a image called [Dev_image_name] and depends on the service called “database”. The service “database” is based on the image called [OracleDB_image_name]. Also don’t forget to set you password.

```
# db.env
ORACLE_PWD=example_password
```

The db.env file specifies the environment variables. It is used by docker-compose when creating the database. Here you can specifiy things like the password. In this case the password “example_password” is used for the SYS, SYSTEM ad PDBADMIN accounts.

```
cocker-compose up
```

Start the containers (as specified in the docker-compose.yml)

Connecting to Database via cx_Oracle

cx_Oracle is a python package that can be used to interact with OracleDB.

```
Import cx_Oracle
db=cx_Oracle.connect(“sys/example_password@//database:1521/XE”,
mode=cx_Oracle.SYSDBA)
```

Connect to the database containers. The connection string has the following format: “[user]/[user-password]@//[service-name]:[port]/[SID]”

The [service-name] is specified in the docker-compose.yml (in this case its “database”)

Create read-only user

```
docker exec -it [container_name] sqlplus \
sys/[password]@//localhost:1521/XE as sysdba

SQL> create user c##[read_only_user] identified by [password];
SQL> grant create session to c##[read_only_user];
SQL> grant select any table to c##[read_only_user];
SQL> conn c##[read_only_user]/[password]@localhost/XE
```

Connection string: c##jw_readonly/readonlypwd@localhost/XE