AIM: Programs on Hydra Tool

1. Install and configure Hydra and at-least Database engines.

Introduction: Hydra is an open-source Python framework that simplifies the development of research and other complex applications. The key feature is the ability to dynamically create a hierarchical configuration by composition and override it through config files and the command line. The name Hydra comes from its ability to run multiple similar jobs - much like a Hydra with multiple heads.

Versions
Hydra supports Linux, Mac and Windows.
Use the version switcher in the top bar to switch between documentation versions.

Version	Python Versions
1.3 (Stable)	3.6 - 3.11
1.2	3.6 - 3.10
1.1	3.6 - 3.9
1.0	3.6 - 3.8
0.11	2.7, 3.5 - 3.8

Install and configure Hydra and at-least Database engines.

Step1: search as www.python.org, which is the official Python website.
Step2: Download Python Executable Installer (Python 3.9.1 version.)
Step3: select "Windows x86-64 executable installer" if your system is 64bit system.
Step 4: Once the executable file download is complete, you can open it to install Python
Or
GOTO downloads and click on executable file
Step 5:Click on Run, which will start the installation process.
Step 6:if you want to save the installation file in a different location, click on Customize installation; otherwise, continue with Install Now. Also, select the checkbox at the bottom to Add Python 3.7 to PATH.
Step 7: Once the installation is complete, the below pop-up box will appear: Setup was successful.
Step8: Now that the installation is complete, you need to verify that everything is working fine. Go to Start and search for Python.
Step 9:You can see Python 3.7 (64-bit) and IDLE. Let's open IDLE, which is the short form for Integrated Development Environment, and run a simple print statement.
Step 10: Python also has a command-line interpreter that works similarly to Python IDLE. Let's print "Hello world" in the command-line
LIBRARIES INSTALLED FOLLOW THE BELOW PRAOCESS

Pip :must be manually installed on Windows. You might need to use the correct version of the file from pypa.io if you're using an earlier version of Python or pip. Get the file and save it to a folder on your PC.
Step 1: Download the **get-pip.py (https://bootstrap.pypa.io/get-pip.py)** file and store it in the same directory as python is installed.
Step 2: Change the current path of the directory in the command line to the path of the directory where the above file exists.
Step 3: get-pip.py is a bootstrapping script that enables users to install pip in Python environments. Run the command given below:

python get-pip.py

Step 4: Now wait through the installation process. Voila! pip is now installed on your system.
Verification of the installation process
pip -V
or
pip –version
Adding PIP to Windows Environment Variables:
If you are facing any path error then you can follow the following steps to add the pip to your PATH.
You can follow the following steps to set the Path:
Go to System and Security > System in the Control Panel once it has been opened.
On the left side, click the advanced system settings link.
Then select Environment Variables.
Double-click the PATH variable under System Variables.
Click New, and add the directory where pip is installed, e.g. C: Python33Scripts, and select OK.
Step1: install hydra framework using pip
Configure Hydra:

Step 1: install pyyaml using pip

YAML is a data serialization format designed for human readability and interaction with scripting languages. PyYAML is a YAML parser and emitter for Python. PyYAML features a complete YAML 1.1 parser, Unicode support, pickle support, capable extension API, and sensible error messages. PyYAML supports standard YAML tags and provides Python-specific tags that allow representing an arbitrary Python object. PyYAML is applicable for a broad range of tasks from complex configuration files to object serialization and persistence.

Step2: install omegaconf using pip

OmegaConf is a hierarchical configuration system, with support for merging configurations from multiple sources (YAML config files, dataclasses/objects and CLI arguments) providing a consistent API regardless of how the configuration was created.

Setp3: **Install at-least Database engines. (mysql)**

Follow the mysql installation steps and create user name: root and password: root.

Step 4: configure with MYSQL database engine

Step 1: open notepad writes your database credentials follow like below

hydra configures with mysql database engine.
Step2: save file with extension of .yaml in C:\Users\TKRCET\.ipython\profile_default\db path .

Step3: Access your database into hydra framework using function @hydra. main. Open notepad

Step2: save file with extension of .py in C:\Users\TKRCET\.ipython\profile_default\db path

WEEK 2

AIM: Create and Establish a Database connection on POSTGRE, SQL & MYSQL Database engines.

Create and Establish a Database connection on MYSQL:

Step1: create a database connection using pip install mysql-connector-python

```
Step 2: Check connection using MySQL Connector

import mysql.connector
```

Establish a database Connection:

```
# Importing module

import mysql.connector

# Creating connection object

mydb = mysql.connector.connect(

   host = "localhost",

   user = "root",

   password = "root"

)

# Printing the connection object

print(mydb)
```

Database connection on POSTGRE:

Create database in POSTGRESQL using Command
CREATE DATABASE aiml;
Connect to the database using command
\c aiml;
Create table using Command
CREATE TABLE finalyear1(rollno varchar(10),name varchar(10));

WEEK-3

AIM: Fetch Tables, Columns and Most frequent values in Hydra application

Creating MySQL Database:

Following example establishes connection with MYSQL and creates a database in it.

```
import mysql.connector

cursor = mydb.cursor()

#Doping database MYDATABASE if already exists.

cursor.execute("DROP database IF EXISTS MyDatabase")

#Preparing query to create a database

db = "CREATE database aiml";

#Creating a database

cursor.execute(db)

#Retrieving the list of databases

print("List of databases: ")

cursor.execute("SHOW DATABASES")

print(cursor.fetchall())

#Closing the connection

conn.close()
```

output:

```
[('aiml',), ('information_schema',), ('mysql',), ('performance_schema',), ('sys',)]

>>>
```

The cursor.MySQLCursor class provides three methods namely fetchall(), fetchmany() and, fetchone() where,

The fetchall() method retrieves all the rows in the result set of a query and returns them as list of tuples. (If we execute this after retrieving few rows it returns the remaining ones).

The fetchone() method fetches the next row in the result of a query and returns it as a tuple.

The fetchmany() method is similar to the fetchone() but, it retrieves the next set of rows in the result set of a query, instead of a single row.

Fetch Tables, Columns

```
import hydra

import mysql.connector

conn = mysql.connector.connect(user='root', password='root', host='localhost', database='aiml')
```

```
cursor = conn.cursor()

sql = '''SELECT * from finalyear'''

cursor.execute(sql)

 result = cursor.fetchone();

print(result)
```

OUTPUT:

(602, 'saranya')

WEEK 4

AIM: Generate relation summary in Hydra

The code has been written on an Ubuntu 18.04 machine with Java 10 and postgreSQL v9.6

Installation Instructions (with Eclipse):

1. Extract the "hydra" folder from the downloaded zip.

2. File->Import->Projects from Folder or Archive.

3. Select codd-data-gen folder inside hydra folder.

4. Add resources folder, present inside codd-data-gen folder, in the build path.

5. Add the lib folder, present inside the codd-data-gen folder, in the libraries path if not automatically added.

Note: If there is an exception concerning Z3, you may have to install it separately. (Install Z3 library configured for Java. (https://github.com/Z3Prover/z3).) Ensure the library is in the path.

PostgreSQL Setup:

Create the desired database in potgreSQL v9.6.

Open the postresql.conf file in the database location and add the following lines:

```
enable_bitmapscan=off

enable_indexscan=off

enable_nestloop=off

enable_sort=off
```

Running Instructions:

1. Open hydra/codd-data-gen/resources/cdgclient/postgres.properties file; update connection string, username, password and dbname to establish database connection.

2. Extract the "test" folder from the downloaded zip in the directory containing the hydra folder.

3. The input to Hydra is given as query plans in JSON format. To run the queries and obtain the plans: (a) Add the input queries as SQL files in the test/sqlqueries folder; (b) list these queries in the test/sqlqueries/index file. A sample of three queries q1.sql,q2.sql,q3.sql is present in the folder, along with the sample index file.

4. The query plans in JSON format are generated automatically in the test/ea folder. List the plans to be given as input in the test/ea/index file. A sample of three plans, q1.json, q2.json, q3.json, is already present in the folder, along with a sample index file.

5. Execute codd-data-gen/src/in/ac/iisc/cds/dsl/cdg/main/Main.java

Note: After successfully running Hydra, the final output, in the form of a database summary, is obtained in the test/databasesummary folder.

Query Assumptions:

1. Each query is of the form: SELECT * FROM <tablenames> WHERE <filter-predicates>.

2. All join conditions are between Primary Key-Foreign Key.

3. Filter predicates on Non-Key Columns.

Data Assumptions:

No Null values in Foreign Key columns.

Step1: After creating Tables for generating a summary relation summary use this HYDRA tool

Step 2: In This HYDRA tool select Client

Step 3:select database Engine from pre-saved connections using username & password.

Step 4: select input workload queries, execution plans metadata information and output location.
Step 5: select input workload queries, execution plans metadata information and output location.
Step 6: input data received message from vendo
Step 7: select table data which want to generate summary

Step 8: select input data folder, empty folder for saving database summary and scale factor.
Step 9:solving a constraint problem per relation

Step 10: generate relation summary

Programs on CODD

CODD is an easy-to-use graphical tool for the automated creation, verification, retention, scaling and porting of database meta-data configurations. It is written entirely in Java and is operational on a suite of industrial-strength database engines -- currently, DB2, Oracle, SQL Server, SQL-MX and PostgreSQL are supported.

The effective design and testing of database engines and applications is predicated on the ability to easily construct alternative scenarios with regard to the database contents.

The following metadata processing modes are available in CODD:

Metadata Construct Mode

In this mode, the user can create metadata statistics from scratch without requiring the existence of any corresponding prior database instance. Apart from a form-based interface to enter the metadata values, a graphical interface is also provided for visually specifying the data distributions of the column values in the relational tables.

Specifically, the editable meta-data is comprised of statistics on the following entities: (a) relational tables (row cardinality, row length, number of disk blocks, etc.); (b) attribute columns (column width, number of distinct values, value distribution histograms, etc.); (c) attribute indexes (number of leaf blocks, clustering factor, etc.); and (d) system parameters (sort memory size, CPU utilization, etc.)

The metadata entered by the user is automatically validated for legality (correct type, valid range) and consistency (compatibility with other meta-data values).

Note: The Construct Mode is not currently available for SQL Server since it stores column distribution statistics in a proprietary internal format.

Metadata Retention Mode (aka Data Drop Mode)

This mode applies to pre-existing databases and allows the user to fully reclaim the storage currently occupied by the database instance without triggering any changes in the associated metadata.

Inter-Engine Metadata Transfer Mode

This mode aids in the automated transfer of metadata across different relational engines (e.g. from DB2 to Oracle), thereby facilitating the comparative study of alternative database offerings. Since the engines are not entirely compatible in their configurations, only a best-effort transfer is provided, with the remaining information to be explicitly entered by the user.

Metadata Scaling

A common activity in database engine testing exercises is to assess the behavior of the system on scaled versions of the original database. To cater to this need, standard industry decision-support benchmarks such as TPC-H and TPC-DS are available in a variety of scale factors. They essentially linearly scale the cardinalities of the user relations to provide a database of the desired size.

CODD supports the above-mentioned space-based scaling. In addition, it also incorporates a novel time-based scaling feature. Specifically, given a query workload $\mathbb{Q}$ and a scale factor α, the relational cardinalities are scaled such that the optimizer's estimated total cost of executing $\mathbb{Q}$ on the new database is scaled by α. As a secondary objective, it is also attempted, as far as possible, to have the execution time of each individual query in $\mathbb{Q}$ scaled by α.

The name of the CODD tool stems from the English word cod, whose archaic meanings include “empty shell” and “fake”, both of which are appropriate to our dataless context. It also coincidentally happens to be the name of Edgar Codd, the father of relational databases.

WEEK 1

AIM: Installation and configuration of CODD

INSTALLATION INSTRUCTIONS:

There are two processes involved in CODD, as shown in the figure:

(a) the CODD Metadata Processor, through which the user creates the desired metadata characteristics

using the data less modes;

(b) the Database Engine, which stores the metadata of the relational tables.

These two processes can all execute on the same machine or on different machines. The machine in

which the CODD Metadata Processor is run should support Java compilation and execution. A few third-party libraries for database connection and Time Scaling are required for CODD to function – the details

are given in License Information (for convenience, this support software (other than the library for Time

Scaling) is included with the CODD code-base in the full version).

CODD is completely written in Java and should, in principle, operate in a platform-independent manner. It

has been successfully tested on the following system and database environments:

System platforms:

(a) Windows 64-bit:Windows Vista Business 64-bit, Sun Ultra 24Intel® Core™2 QuadCore 3GHz, 8 GB Ram,

(b) Windows 64-bit: Windows 7 Professional, Intel® Core™i3 2.10GHz, 4 GB RAM

(c) Windows 64-bit: Windows 8/8.1, Intel® Core™i5 2.60GHz, 6 GB RAM

Database engines: DB2 9/10, Oracle 11g, SQL Server 2008, SQL/MX 3.1, PostgreSQL 9.3.

Installation Steps:

I Setup the Database Engine

1. Install a database engine. Database Engines can be obtained from the vendor websites and installed by following their installation steps.

2. Populate the database with data. This step may be needed only for certain modes of CODD (like Metadata Retain mode). Details can be found when we explain the different modes.

Note: CODD can be used with generic relational database schemas and SQL query templates.

The illustrative examples in the CODD documentation are with respect to the TPC-H benchmark.

II Download the CODD Software

1. From CODD Download, download the CODD code (version 1.0) as a zip file. Extract its contents. A directory called CODD1.0 in which the entire code-base is contained will be created.

All paths mentioned in this document and the supporting documentation are with reference to this directory.

2. Contents of the extracted zip file:

https://dsl.cds.iisc.ac.in/projects/CODD/downloads.php

The extracted CODD1.0 folder contains the source code, CODDDoc folder, folder named Libraries, which contains the associated library files and another folder named PostgreSQLModifications, which contains files required for PostgreSQL to work with CODD. (see PostgreSQL for details) All the necessary library files are included in the Libraries folder except SuanShu, which is needed for Time Scaling. Obtain the SuanShu library, license file and put it in the lib folder. License related information about the tool and libraries can be found in License Information

3. Read through the documentation given in the CODDDoc directory.

III Getting Started with the tool

Prerequisite: Java 1.7 or above. It must be present in the CLASSPATH Environment variable. i.e. Command "java -version" in the command prompt should give the version number more than 1.7.

1. Compiling and running the Main.java present in the folder iisc/dsl/codd/ starts the tool. We suggest the user to import CODD folder as a project in Netbeans/Eclipse IDE and execute the project to start the tool.

2. Running Main.java takes the user to a screen as in Figure 1, indicating that the tool has started successfully.