# Db2 Query Parallel Launcher

**Luis Reina**

**Version 1.0**

**April 2020**

# Index

# 1 Introduction.

This document describes how to implement a **Db2 Query Parallel Launcher** to maintain always a certain number of SQL running in parallel.

The Query Launcher receives 1 parameter to determine how many queries you want to have running in parallel and the program will ensure you always have this number of queries running in parallel until all the queries you have provided, in files, are executed.

It based in Linux scripts so any one can implement it easily. You can copy and paste from this document to create the necessary scripts.

# 2 Preparing your Queries

## 2.1 Putting your SQLs into files

First you have to create the files with the SQLs you want to run.

Each file may contain 1 or more SQLs. But the SQLs inside each file will be run sequentially.

End each SQL sentence with a semicolon ;

Example:

- **q1.sql** that contains any SQL , for example:

  SELECT TABNAME FROM SYSCAT.TABLES;

- **q2.sql** that contains any SQL, for example

  SELECT TABSCHEMA FROM SYSCAT.TABLES

…….

- **qn.sql**

## 2.2 Create the Queries list file

Create an additional file **query_list.out** with all the queries file names, generate it like this:

```
# from the same directory you put your query files, execute:
ls q*.sql > query_list.out

# remove the .sql extension:
sed -i "s/.sql//" query_list.out
```

# 3 Create the launch script for just 1 file.

Create in the same folder a file **launch_one.sh** with this content:

```
#!/bin/bash
# if your database name is not bludb, change the name for the correct one.
rm -f $1.out
db2batch -d bludb -f $1.sql -i complete -iso ur -o r 1 -r $1.out
```

db2batch is a db2 utility that gather interesting info about the execution of SQLs

I have use the following options (you can see others in the Knowledge Center):

- **-i complete** => Give the time broken down by parts (fetch, execute,...)
- **-iso ur** => Execute using Uncommited Read (dirty read).
- **-o r 1** => Fetch all the rows (real query) but only show 1 in the output. This way we simulate a real execution and we don't store big results
- **-r $1.out** => Output file

- You must give execute permissions to this shell script:

```
chmod 777 launch_one.sh
```

- You can test this script by executing:

```
# The argument is the file name without the .sql extension.
./launch_one.sh  q1
```

It will execute the SQL contained in the **q1.sql** file and will leave the output in **q1.out** where you can see the duration of the SQLs.

# 4 Create the auxiliary functions script

Create a **functions** file with this content (copy/paste):

```
#############################################################
# Auxiliary shell script functions that will be used from
#  the query launcher
#############################################################

function contar {
  jobs -r | wc -l | tr -d " "
}

function set_paralelismo_max {
 max_jobs=$1
}

function get_paralelismo {
  [[ max_jobs ]] && {
    echo max_jobs
    echo 0
  }
  echo 1
}

function disponible() {
  (( $(contar) < $max_jobs )) && return 0
  return 1
}

function esperar() {
  while true;
   do
    disponible && break
    sleep 0.5s
  done
}
```

# 5 Create the launcher.sh file

Create a **launcher.sh** file with this content (copy/paste):

```
#!/bin/bash
# This script receive an integer argument with the number of queries to be run in parallel.
source ./functions
set_paralelismo_max $1
while read -r query
do
  esperar
  ./launch_one.sh $query &
done < query_list.out
```

# 6 Running the query launcher

- Put all the files that has been created through this document in a directory, here is the list:
  - q1.sql, q2.sql,…, qn.sql
  - query_list.out (this one can be created with the provided commands)
  - launch_one.sh
  - functions
  - launcher.sh

- Give execution permissions to the shell scripts
  chmod 777 launch_one.sh
  chmod 777 launcher.sh

- Run the launcher with some degree of parallelism (e.g. 3 queries in parallel)
  **./launcher.sh 3**

- You can use db2top to monitor the concurrency.