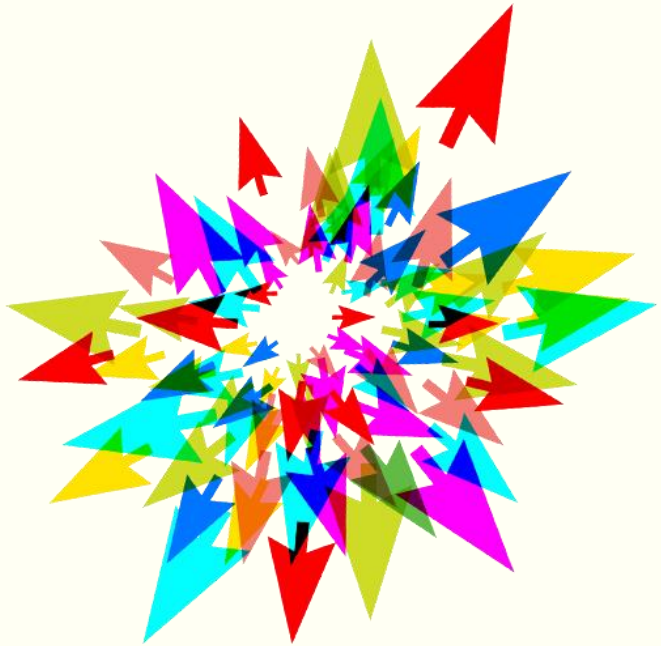
DATABASES

Introduction to databases on Cedar

By Wolfgang Richter E-mail: wolfgang@sfu.ca
and Ata Roudgar E-mail: aroudgar@sfu.ca

Database administrators/analysts for Compute Canada and SFU



Workshop topics to be covered

- Short introduction to SQL
- Overview of the two database servers on Cedar
- Shallow dive into MySQL on Cedar
- Deeper dive into PostgreSQL on Cedar
- For copy of these slides:

  https://westgrid.github.io/ubcSummerSchool2019/4-materials.html

Short Introduction to SQL

- Why databases?
- Frontends
- Basic commands

Short Introduction to SQL

Why databases?

- From Wikipedia: A **database management system** (**DBMS**) is a computer program (or more typically, a suite of them) designed to manage a database, a large set of structured data, and run operations on the data requested by numerous users.
- From Wikipedia: A **database** is an organized collection of data. A relational database, more restrictively, is a collection of schemas, tables, queries, reports, views, and other elements.
- For researchers, they can use a DBMS such as MySQL or Postgres (aka PostgreSQL) to set up their own databases, upload data to them, and issue queries against the data to extract a subset for insight or further processing.

Short Introduction to SQL

Frontends available on cedar.computecanada.ca

- mysql
  - A utility program to connect to the Cedar MySQL server.
  - Can be used to execute SQL commands interactively or in batch.
- psql
  - A utility program to connect to the Cedar Postgres server.
  - Can be used to execute SQL commands interactively or in batch.
- Scripting/programming languages
  - Examples: Python, C++, Perl
  - Write a script to connect to the database server, issue SQL commands, return results, work with the results, etc.

Short Introduction to SQL

Basic commands - Exercise

Point your browser at w3schools.com/sql

Lets you learn and try out basic SQL commands that is used in the major database systems such as MySQL, and Postgres.

- Click on a SQL topic or command in the left column.
- Get an explanation along with examples of the command and an opportunity to try out variations of the command. If the SQL command varies between different types of DBMS's, it will show you the differences.

Short Introduction to SQL

Basic commands - Exercise

w3schools.com/sql -- click on “SQL Intro”

- What is SQL?
  - SQL stands for Structured Query Language
  - SQL lets you access and manipulate databases
  - SQL became a standard of the American National Standards Institute (ANSI) in 1986, and of the International Organization for Standardization (ISO) in 1987
- SQL is a Standard - BUT....
  - Although SQL is an ANSI/ISO standard, there are different versions of the SQL language.
  - However, to be compliant with the ANSI standard, they all support at least the major commands (such as SELECT, UPDATE, DELETE, INSERT, WHERE) in a similar manner.
  - **Note:** Most of the SQL database programs also have their own proprietary extensions in addition to the SQL standard!

Short Introduction to SQL

Basic commands – Exercise

- What Can SQL do?
  - SQL can execute queries against a database
  - SQL can retrieve data from a database
  - SQL can insert records in a database
  - SQL can update records in a database
  - SQL can delete records from a database
  - SQL can create new databases
  - SQL can create new tables in a database
  - SQL can create stored procedures in a database
  - SQL can create views in a database
  - SQL can set permissions on tables, procedures, and views

Short Introduction to SQL

Basic SQL commands - Exercise

- Click on each of these items in the left column to get a feel for basic SQL operations (w3schools.com/sql):
  - SQL Create Table – to create a new table
  - SQL Insert Into – to insert rows into a table
  - SQL Select – to search a table
  - SQL Where – to include a condition in a search
  - SQL Order By – to order the results of a search
  - SQL Delete – to delete rows from a table
  - SQL Update – to update rows in a table

Overview of the two database servers on Cedar

Postgres – cedar-pgsql-vm

- Ref:
  CC database servers: docs.computecanada.ca/wiki/Database_servers
  Navigate to the ***Cedar PostgreSQL Server*** section
- Need to be connected via SSH to a Cedar node such as cedar.computecanada.ca to be able to access this server.
- Description: General purpose server for the researcher wanting to set up SQL tables and issue SQL commands against them.
- Server full name: cedar-pgsql-vm.int.cedar.computecanada.ca
  Short name: cedar-pgysql-vm (can be used instead usually)
- Version: PostgreSQL version 10.1. PostGIS version 2.4 extension available for your database upon request.
- Documentation: https://www.postgresql.org and https://postgis.net/documentation (PostGIS documentation)

Overview of the two database servers on Cedar

MySQL – cedar-mysql-vm

- Ref:
  CC database servers: docs.computecanada.ca/wiki/Database_servers
  Navigate to the ***Cedar MySQL Server*** section.

  MySQL (MariaDB version) Knowledge Base: **http://www.mariadb.com** Click on "Knowledge Base" link on the page.
- Need to be connected via SSH to a Cedar node such as cedar.computecanada.ca to be able to access this server.
- Description: General purpose server for the researcher wanting to set up SQL tables and issue SQL commands against them.
- Server full name: cedar-mysql-vm.int.cedar.computecanada.ca
  Short name: cedar-mysql-vm (can be used instead usually)
- Version: MariaDB version 10.2 Community Edition
- Documentation: **http://www.mariadb.com** click on "Knowledge Base".

Overview of the two database servers on Cedar

Database Server Performance features

Our database servers are configured for high performance:

- Database server disk is SSD (solid state device) which makes for high performance I/O .
- Several gigs of RAM for the query cache means most data is fetched from high speed RAM in a query rather than from disk.
- Maintenance jobs run on the database servers to do nightly backups on all the databases and monitor the health of the system.

Shallow dive into MySQL on Cedar - Overview

Your MySQL account and database

- To be able to use the MySQL server, a MySQL id has to be set up for you on the cedar-mysql-vm server.
- The MySQL id will match your Compute Canada id but have its own password.
- The MySQL id will have privileges that allow you to create and administer your own MySQL databases.
- To request access to the Cedar MySQL server, send a request to support@computecanada.ca with the following information:
  - Your Compute Canada username
  - Amount of database space needed for your project
- A special configuration file .my.cnf for your access will be set up under your home directory owned by root and readable only to you. For example, it contains your MySQL password so you don't have to supply it when connecting to your id.

Shallow dive into MySQL on Cedar - Overview

MySQL (MariaDB flavor) Knowledge Base:

- Point browser at: http://www.mariadb.com
  and click on “Knowledge Base” link on the page.

Some relative recent new features offered in the MariaDB flavour of MySQL:

- New JSON functions available that allow you to work with data in the JSON format.
- CREATE OR REPLACE TABLE command to conveniently combine the DROP IF TABLE EXISTS and CREATE TABLE command into one.
- Materialized views. Flexviews is a materialized views implementation for MariaDB.

  A materialized view is similar to regular view, except that the results are stored into an actual database table, not a virtual one. The result set is effectively cached for a period of time. When the underlying data changes the view becomes stale. Because of this, materialized views must be frequently "refreshed" to bring them up-to-date.

Deeper dive into PostgreSQL on Cedar - Preparation

Your Postgres account and database

- To be able to use the Postgres server, a Postgres id and database has to be set up for you on the cedar-pgsql-vm server.
- The Postgres id will match your Compute Canada id and when used from a node on Cedar does not require that you supply a password. The Database name will contain your id in the name followed by "_db" (e.g. bob_db) and be exclusively available to you to administer.
- If you are signed up for this course, we should have already set up an id and database for you. Otherwise we will try to set it up on the fly so you can do the workshop now. The general procedure to get a Postgres id and database is to send a request to support@computecanada.ca with the following information:
  - Your Compute Canada username
  - Amount of database space needed for your project

Deeper dive into PostgreSQL on Cedar - Preparation

Postgres Knowledge Base:

- Point browser at: https://www.postgresql.org/docs
  and **c**lick on the “10” link under the “ONLINE MANUALS” heading.

Some recent new features offered by our version of Postgres

- Declarative table partitioning. (Table partitioning means internally the table can be broken into pieces. Declarative means you can tell it how to split up the pieces).
- Improved query parallelism. (A query internally may be broken into parts that can run in parallel to give faster results)
- Significant general performance improvements.
- Stronger password authentication based on SCRAM-SHA-256.

Deeper dive into PostgreSQL on Cedar - Preparation

Exercise - Set up your Postgres environment under your Cedar home directory

- Use an ssh client (such as Putty in Windows) to connect to your Compute Canada id on cedar.computecanada.ca .
- You will be using the “psql” client tool but the default version is old. You will want to upload a more recent one available. Issue the following commands:

```
$ psql --version
$ module load postgresql
$ psql --version
```

  To avoid having to type these commands each time you login to use Postgres, insert the following command into your Bash init file:

```
$ cd
$ vi .bashrc
…
module load postgresql
…
```

Deeper dive into PostgreSQL on Cedar - Preparation

- To avoid having to supply parameters to the “psql” client, issue commands like this:

```
$ export PGHOST=cedar-pgsql-vm.int.cedar.computecanada.ca
$ export PGDATABASE=[user]_db

#-- Also insert these into your Bash init file:
$ vi .bashrc
…
export PGHOST=cedar-pgsql-vm.int.cedar.computecanada.ca
export PGDATABASE=[user]_db
…
```

Deeper dive into PostgreSQL on Cedar - Preparation

Exercise – Run a set of commands to set up a Postgres table and populate it

- Set up your own copy of the workshop scripts that we use here:

```
$ cd
$ cp -r /home/wolfgang/db_workshop db_workshop
```

- Issue the commands to set up the table and populate it:

```
#-- Navigate to sample workshop scripts
$ cd
$ cd db_workshop/postgres

#-- View a prepared set of SQL commands to create a table of
#-- Canadian city info and upload data to it
$ more canada-city.sql

#-- Execute the SQL commands:
#-- It should display a total count of 5521 records
$ psql < canada-city.sql
```

Deeper dive into PostgreSQL on Cedar – psql Client

Exercise – Run a set of commands to set up a Postgres table and populate it

Useful interactive Postgres command to check things out:

```
#-- Start up interactive "psql" client
$ psql

-- Get list of all the databases on the server (look for yours)
=> \list     (or \l)
-- List tables and views in your database
=> \dt
-- Show fields in a table
=> \d+ Canada_city;
-- Count number of rows in your table
=> select count(*) from canada_city;
-- Quit
=> \quit   <-- (or \q)
```

Deeper dive into PostgreSQL on Cedar – psql Client

Database server Exercise – Query optimization

In Postgres (as in other database systems such as MySQL), you can create indexes to improve SQL commands and can use the EXPLAIN command to verify it.

```
#-- View a SQL SELECT command followed by an "explain" of it
$ more demo1_explain.sql

#-- Execute the commands and view the Query Plan info
$ psql < demo1_explain.sql

#-- Issue a command to create an index on "population" field
$ more demo1_index.sql
$ psql < demo1_index.sql

#-- Issue commands again to show lower cost for the Query
#-- Plan
psql < demo1_explain.sql
```

Deeper dive into PostgreSQL on Cedar – psql Client

Database server Exercise – Query optimization

- Explains do not actually execute the command, so are ideal for doing against a command that you are thinking of issuing and looking to see the cost associated with it – before actually executing it.
- Rule of thumb on fields to index:
  - When doing a join, index each field involved in the join
  - In a sort, i.e. “order by”
  - In a conditional (i.e. WHERE clause)

Deeper dive into PostgreSQL on Cedar – psql Client

Database server Exercise – Using GROUP BY and HAVING

Example of using GROUP BY and HAVING to look for duplicated information.

```
#-- View a SQL SELECT command with GROUP BY
#-- and HAVING to find duplicate cities in the data
$ more demo1_groupby.sql

-----------------------------------------
select city,count(city) as "Count"
from canada_city
group by city
having count(city) > 1
;
-----------------------------------------
```

Deeper dive into PostgreSQL on Cedar – psql Client

Database server Exercise – Using GROUP BY and HAVING

Example of using GROUP BY and HAVING to look for duplicated information.

```
#-- Execute the command
$ psql < demo1_groupby.sql > demo1_groupby.rep

#-- View result of the command
$ more demo1_groupby.rep

#-- Many duplicates don't appear to have populations filled
#-- in. Check for duplicates for just those
$ more demo2_groupby.sql
$ psql < demo2_groupby.sql > demo2_groupby.rep
$ more demo2_groupby.rep
```

Deeper dive into PostgreSQL on Cedar - Python

Exercise – Python scripting

We show here a sample Python script that you can use to issue SQL commands on a Postgres server, fetch results, and work with them in Python.

Issue these commands:

```
#--
$ cd ~/db_workshop/postgres
#-- view the code
$ more sample_pg.py
#-- each time before using Python
$ source ~/python/bin/activate
#-- execute the program to verify that it will
#-- won't work because CC password is required for
#-- Postgres access from non-compute nodes
$ python sample_pg.py
```

Deeper dive into PostgreSQL on Cedar - Slurm

Exercise – Python scripts

Submit the Python script as a Slurm job

- Issue commands

```
$ cd ~/db_workshop/postgres
#-- View slurm script to execute sample Python script
#--- against your Postgres database
$ more pythonsample_pg.sh
-----------------------------------------
#!/bin/bash
#SBATCH --account=wgssubc-wa_cpu
#SBATCH --reservation=wgssubc-wr_cpu
#SBATCH --time=00:01:00
#SBATCH --job-name=test_pg
#SBATCH --output=%x-%j.out
source ~/python/bin/activate
python sample_pg.py
-----------------------------------------
$ sbatch pythonsample_pg.sh
```

** EXTRA DATABASE EXERCISES **

Database slides following contain the following extras that you can learn about and try out on your own by following the information on the following slides. It also includes some MySQL exercises for those who are interested in using MySQL database services.

Deeper dive into PostgreSQL on Cedar - Perl

Exercise – Perl scripts

We show here a sample Perl script that you can use to issue SQL commands on a Postgres server, fetch results, and work with them using basic Perl (no DBI).

Issue these commands:

```
#--
$ cd
$ cd db_workshop/postgres
#-- view the code
$ more sample_pg.pl
#-- execute the program to verify that it works
$ perl sample_pg.pl [database]
```

Deeper dive into PostgreSQL on Cedar - Perl

Exercise – Perl scripts

Submit the basic Perl script as a Slurm job

- Issue commands

```
$ cd ~/db_workshop/postgres
#-- Edit slurm script to execute sample Perl script
#-- against your Postgres database
$ vi perlsample_pg.sh
-----------------------------------------------
#!/bin/bash
#SBATCH --account=wgssubc-wa_cpu
#SBATCH --reservation=wgssubc-wr_cpu
#SBATCH --time=00:01:00
#SBATCH --job-name=test_pg
#SBATCH --output=%x-%j.out
$ perl sample_pg.pl [database] > sample_pg.rep
-----------------------------------------------

$ sbatch perlsample_pg.sh
```

Deeper dive into MySQL on Cedar - Preparation

Exercise - Set up your MySQL environment under your Cedar home directory

- Use an ssh client (such as Putty in Windows) to connect to your Compute Canada id on cedar.computecanada.ca .
- You will be using the "mysql" client tool but the default is old. You will want to upload a more recent one available. Issue the following commands:

```
$ mysql –version
$ module load mariadb
$ mysql –version
```

  To avoid having to type these commands each time you login to use MySQL, insert the following commands into your Bash init file:

```
$ cd
$ vi .bashrc
…
module load mariadb
…
```

- As mentioned, there is a .my.cnf file present under your home directory if a MySQL id has been set up for you. Verify this but be careful about viewing the contents in public as it contains your MySQL id password:

```
$ cd
$ ls –ld .my.cnf
```

Deeper dive into MySQL on Cedar - Preparation

Exercise – Use the mysql client to create your database

- Interactively: simply issue the command:

```
$ mysql
```

  You won't need to supply MySQL id and password and server name as the client automatically passes along server host, MySQL user, MySQL password from the .my.cnf file in your home directory.

- Create your database – making sure to prefix it with your id. For example, if your id is "jack", then this would work (MySQL commands are terminated with ";")

```
$ mysql
-- List your databases
> show databases;
-- Create the database with name prefixed with your id
> create database jack_db;
> quit;
```

Deeper dive into MySQL on Cedar - Preparation

Exercise – Run a set of commands to set up a MySQL table and populate it

- Set up your own copy of the workshop scripts that we use here:

```
$ cd
$ cp -r /home/wolfgang/db_workshop db_workshop
```

- Issue the commands to set up the table and populate it:

```
#-- Navigate to sample workshop scripts
$ cd
$ cd db_workshop/mysql

#-- See commands to create a table of Canadian
#-- city info and upload data in CSV format to it
$ more canada-city.sql

#-- see information about the sample data to be uploaded
$ more canada-city.doc

#-- Execute the following commands, substituting the
#-- name of your database for [database]
#-- If successful, it should show a total count of 5521
#-- display 5 rows of the new canada_city table
$ mysql -D [database] < canada-city.sql
```

Deeper dive into MySQL on Cedar - Preparation

Exercise – Run a set of commands to set up a MySQL table and populate it

Useful interactive MySQL commands to check things out:

```
#-- Start up interactive "mysql" client
$ mysql
-- Show your databases
> show databases;
-- Connect to your database
> use [database];
-- List tables in your database
> show tables;
-- Show names of fields in a table
> describe canada_city;
-- Count number of rows in your table
> select count(*) from canada_city;
-- Quit
> quit;
```

Deeper dive into MySQL on Cedar - Preparation

Database server Exercise – Query optimization using EXPLAIN and using indexes

```
#-- Example of issuing a SQL command followed by EXPLAIN
#-- of the command. Shows 5662 rows of the table processed
$ more demo1_explain.sql
$ mysql -D [database] < demo1_explain.sql | more

#-- Issue this command to create an index on “population”
#-- field
$ more demo1_index.sql
$ mysql -D [database] < demo1_index.sql

#-- Issue explain command to show now only 26 rows of
#-- the table
#-- now being accessed to get results
$ mysql -D [database] < demo1_explain.sql | more
```

Deeper dive into MySQL on Cedar – Preparation

Notes

- Explains do not actually execute the command, so are ideal for doing against a command that you are thinking of issuing and looking to see how much work it is likely to be doing – before actually executing it.
- Rule of thumb on fields to index:
  - When doing a join, index each field involved in the join
  - In a sort, i.e. “order by”
  - In a conditional (i.e. WHERE clause)

Deeper dive into MySQL on Cedar – Perl #1

Exercise – Perl scripts #1

We show here a sample Perl script that you can use to issue SQL commands on a MySQL server, fetch results, and work with them with basic Perl.

Issue these commands:

```
#--
$ cd
$ cd db_workshop/mysql
#-- view the code
$ more sample_mysql.pl
#-- execute the program to verify that it works
$ perl sample_mysql.pl [database]
```

Deeper dive into MySQL on Cedar – Perl #1

Exercise – Perl scripts #1

Submit the basic Perl script as a Slurm job

- Issue commands

```
$ cd ~/db_workshop/mysql
#-- Edit slurm script to execute sample Perl script
#-- against your MySQL database
$ vi perlsample_mysql.sh
---
#!/bin/bash
#SBATCH --account=wgssubc-wa_cpu
#SBATCH --reservation=wgssubc-wr_cpu
#SBATCH --time=00:01:00
#SBATCH --job-name=test_pg
#SBATCH --output=%x-%j.out
$ perl sample_mysql.pl [database] > sample_mysql.rep
---

$ sbatch perlsample_mysql.sh
```

Deeper dive into MySQL on Cedar – Perl #2

Exercise – Perl scripts #2

We show here a sample Perl script that you can use DBI to issue SQL commands on a MySQL server, fetch results, and work with them.

The DBI Perl modules need to be installed using CPAN. As part of another project, these modules are accessible to you by setting PERL5LIB appropriately as shown below via the "source perl5lib" command.

Issue these commands:

```
#--
$ cd ~/db_workshop/mysql
#-- view the code
$ more sample2_mysql.pl
#-- Set up DBI environment
$ source perl5lib
#-- execute the program to verify that it works
$ perl sample2_mysql.pl [user] [database]
```

Deeper dive into MySQL on Cedar – Perl #2

Exercise – Perl scripts #2

Submit the basic Perl script as a Slurm job

- Issue commands

```
$ cd ~/db_workshop/mysql
#-- Edit slurm script to execute sample Perl script
#-- against your MySQL database
$ vi perlsample2_mysql.sh
---
#!/bin/bash
#SBATCH --account=wgssubc-wa_cpu
#SBATCH --reservation=wgssubc-wr_cpu
#SBATCH --time=00:01:00
#SBATCH --job-name=test_pg
#SBATCH --output=%x-%j.out
$ source perl5lib
$ perl sample2_mysql.pl [user] [database] > sample2_mysql.rep
---

$ sbatch perlsample2_mysql.sh
```

Deeper dive into MySQL on Cedar - Python

Exercise – Python scripting

We show here a sample Python script that needs fixing to work.

Issue these commands:

```
#--
$ cd ~/db_workshop/mysql
#-- view and fix the code with your database name
$ vi sample_mysql.py
  --
  # Name of database to be accessed
  database = "yourdb";
  --
#-- each time before using Python
$ source ~/python/bin/activate
#-- execute the program to verify that it works
$ python sample_mysql.py
```

Deeper dive into MySQL on Cedar - Python

Exercise – Python script

Submit the basic Python script as a Slurm job

- Issue commands

```
$ cd ~/db_workshop/mysql
#-- Edit slurm script to execute sample Python script
#-- against your MySQL database
$ vi pythonsample_mysql.sh
---
#!/bin/bash
#SBATCH --account=wgssubc-wa_cpu
#SBATCH --reservation=wgssubc-wr_cpu
#SBATCH --time=00:01:00
#SBATCH --job-name=test_pg
#SBATCH --output=%x-%j.out
$ source ~/python/bin/activate
$ python sample_mysql.py

---

$ sbatch pythonsample_mysql.sh
```

Using Slurm During the Workshop in 2018

Using Slurm job reservations on Cedar and Graham during the workshop

To use the summer school reservations on the clusters, in addition to all other flags, you will need to pass the following flags to Slurm:

- For CPU jobs

```
--account=wgssubc-wa_cpu --reservation=wgssubc-wr_cpu
```

- For GPU jobs

```
--gres=gpu:1 --account=wgssubc-wa_gpu
    --reservation=wgssubc-wr_gpu
```

All instructors, registered attendees with CC accounts, and all guest accounts have been added to the reservations. If you need to add anyone else to a reservation, please contact Alex Razoumov.

Deeper dive into PostgreSQL on Cedar – Python in 2018

Exercise - Python scripting to access MySQL and Postgres databases

Set up Python environment if you had not already done this earlier.

```
$ cd
$ cd db_workshop/postgres
$ more python_env.sh
----
$ virtualenv ~/python
$ source ~/python/bin/activate
$ pip install psycopg2
----
$ source python_env.sh
```

Deeper dive into MySQL on Cedar – Python in 2018

Exercise - Python scripting to access MySQL and Postgres databases

Set up Python environment once only.

```
$ cd
$ cd db_workshop/mysql
$ more python_env.sh
----
$ virtualenv ~/python
$ source ~/python/bin/activate
$ pip install MySQL-python
$ pip install psycopg2
----
$ source python_env.sh
```