Lab 1 - SQL&RA setup

CompSci 316
Fall 2022
TA: Yuxi Liu
Roadmap

1. Setup VM
2. Practice SQL on Pgweb
3. Practice RA on RATest
Virtual Machine instruction

https://courses.cs.duke.edu/fall22/compsci316d/instructions/vm/

Step 1: Reserve a VM: https://vcm.duke.edu/
Welcome to Virtual Computing Manager!

Virtual Computing Manager is a service providing the Duke community with easy access to virtual software packages, and semester-long virtual machine (VM) reservations. Access specialized software without installing it on your own computer, host your own server for development projects and coursework, or customize your own environment to use for the semester.

My Reservations

VIRTUAL MACHINES

vcm- : vm.duke.edu

Virtual Machines (aka VMs)

Your Duke VM is like having a second computer that lives in OIT. You can log into and use your VM from your own machine.
- Run Windows or Linux
- Install zero, one or multiple apps for free

Virtual Software (aka Containers)

A Container lets you use a desktop software application in your browser without installing it on your machine or your VM.
- Simple to use
- Launch an app in a click!
- Use anywhere you can run a browser

check the vm
THIS VM WILL POWER DOWN AUTOMATICALLY

To make efficient use of the shared VCM resources and reduce our carbon footprint, this VM will be powered down every morning at 6:00 AM. Click the Power on button to turn your VM back on before you log into it. If your VM is being used as a server or runs very long computations and must remain on continuously, you can un-check the Automatic power down check box above to opt out of automatic shutdowns.

General Information

- Hostname: vcm-15837.vm.duke.edu
- Operating System: Ubuntu18 Server
- Base memory: 2 GB
- Processors: 2
- Extra info: Request: https://clockworks.oit.duke.edu/vm_request/119995 Deployed from OVF packer-ubuntu-18.04-server-amd64(content version 13) on content library FE-VM-Manage © 2020-08-05 09:04:59-0400 ClockworksUrl: https://clockworks.oit.duke.edu/vms/31578
- VM Status: complete

Users

- Username: vcm
- Admin username: vcm
- Admin password: [Enter password here]

check username and password
To prevent your VM auto power down everyday

1. Uncheck Automatic power downs (it may ask you a reason)
Step 2: Access to the Machine

Mac/Linux:

In terminal, type “ssh vcm@vcm-xxxxx.vm.duke.edu”

(vcm-xxxxx.vm.duke.edu is your hostname)

Enter password for the admin user (vcm)

Another choice is use non-admin account

netid@vcm-xxxxx.vm.duke.edu

Password is your netid’s password
If Windows: Download MobaXterm from the official website

MobaXterm Home Edition

Download MobaXterm Home Edition (current version):

- MobaXterm Home Edition v22.1 (Portable edition)

Download previous stable version:
- MobaXterm Portable v22.0
- MobaXterm Installer v22.0

You can also get early access to the latest features and improvements by downloading MobaXterm Preview version.

MobaXterm Preview Version

By downloading MobaXterm software, you accept MobaXterm terms and conditions

You can download the third party plugins and components sources here

If you use MobaXterm inside your company, you should consider subscribing to MobaXterm Professional Edition: your subscription will give you access to professional support and to the “Customizer” software. This customizer will allow you to generate personalized versions of MobaXterm including your own logo, your default settings and your welcome message.

Please contact us for more information.
Click SSH
1. \texttt{vcm@vcm-xxx.vm.duke.edu}

2. OK
VM Setup - download & run setup script

1. Download the setup script:
   a. `wget -N https://courses.cs.duke.edu/fall22/compsci316d/static/init.sh`

2. Run the script:
   a. `bash init.sh`
VM Setup - add SSH Key to your GitLab

1. Pause at this screen
2. Copy your public key (everything starting from “ssh-rsa” to your duke email address)
3. Open a browser, go to gitlab.oit.duke.edu, login with your Duke credentials
4. Click on the icon on upper right corner, Go to setting → SSH Keys
5. Paste the public key under Key, give it a title, doesn’t matter what the title is, don’t have to set expiration date
6. Add Key
7. Go back to your terminal and hit [Enter]
8. Answer yes if asked to continue to connect
VM Setup - Install necessary tools

1. init.sh downloads a repository from GitLab
   a. They are under path /opt/dbcourse on your VM
2. Run the following command:
   a. `/opt/dbcourse/sync.sh`
   b. This would install necessary tools for setup
3. Just wait for the command to finish (it takes a while)
4. Once it is finished, reboot to adopt the changes:
   a. `sudo reboot now`
   b. This would terminate SSH connection, just wait for the reboot and re-ssh
VM Setup - Final Step

1. Run the command:
   a. `/opt/dbcourse/examples/db-beers/setup.sh`

2. Once done, you should see output similar to the screenshot

3. Go to the following URL:
   a. `vcm-XXXXX.vm.duke.edu:8081`
   b. `XXXXX` is the number of your VM
   c. Must include the port number at the end
Pgweb

In web browser, enter vcm-xxxxx.vm.duke.edu:8081 (27266, 28337, 28273)
Step 1. Write your SQL query

Step 2. Click ‘Run Query’

Step 3. Expect to see the result
SQL example 1:

What beer(s) does the bar – “Satisfaction” serve?
SQL example 1:

What beer(s) does the bar – “Satisfaction” serve?

Expect to see:

```
1. SELECT beer
2. FROM serves
3. WHERE bar = 'Satisfaction'
```

SQL query

Query result
SQL example 2:

Find addresses of all bars that drinker – “Ben” frequents?
SQL example 2:

Find addresses of all bars that drinker “Ben” frequents?

Expect to see:

```
1  SELECT bar.address
2  FROM bar, frequents
3  WHERE bar.name = frequents.bar AND frequents.drinker = 'Ben'
```

Run Query  Explain Query

<table>
<thead>
<tr>
<th>address</th>
</tr>
</thead>
<tbody>
<tr>
<td>912 W. Main Street</td>
</tr>
</tbody>
</table>
Relational Algebra Recap (from yesterday)

Selection (to select rows) \( \sigma_{...} R \)
use \select_{...} R

Projection (to select columns) \( \pi_{\{} R \)
use \project_{\{} R

Join (to join with other tables under some conditions) \( R_1 \bowtie_{...} R_2 \)
use R1 \join_{...} R2

Union \( R_1 \cup R_2 \)
use R1 \union R2

Difference \( R_1 - R_2 \)
use R1 \diff R2

Intersection \( R_1 \cap R_2 \)
use R1 \intersect R2
Combining multiple operation to an expression

On paper/exam, you can either write as a tree or a formula using symbols.

For HW, you will write queries in radb form — \select..\project.. etc.
## Relational Operations


<table>
<thead>
<tr>
<th>Selection</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>\select_{\text{condition}} \text{input\_relation}</code></td>
</tr>
<tr>
<td>output the tuples in \text{input_relation} that satisfy \text{condition}</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Projection</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>\text{project\_list} \text{input\_relation}</code></td>
</tr>
<tr>
<td>output only the attributes in \text{attr_list} for each tuple in \text{input_relation} (duplicate output tuples are removed)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Theta-Join</th>
</tr>
</thead>
<tbody>
<tr>
<td>\text{input_relation_1} \text{\ join_list} \text{input_relation_2}</td>
</tr>
<tr>
<td>find pairs of tuples from \text{input_relation_1} and \text{input_relation_2} that satisfy \text{cond}, and for each such pair output the concatenation of the two tuples</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Natural join</th>
</tr>
</thead>
<tbody>
<tr>
<td>\text{input_relation_1} \text{\ join} \text{input_relation_2}</td>
</tr>
<tr>
<td>find pairs of tuples from \text{input_relation_1} and \text{input_relation_2} that agree on the values of all commonly named attributes, and for each pair output the concatenation of the two tuples (with only one copy of the commonly named attributes)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Set union, difference, and intersection</th>
</tr>
</thead>
<tbody>
<tr>
<td>\text{input_relation_1} \text{\ union} \text{input_relation_2}</td>
</tr>
<tr>
<td>output the union of \text{input_relation_1} and \text{input_relation_2}</td>
</tr>
</tbody>
</table>

| rename \{new\_attr\_names\} \text{input\_relation} |
| output the same \text{input\_relation} but with attributes renamed to \text{new\_attr\_names} |

| \text{rename\_list} \text{\ input\_relation} |
| output the same \text{input\_relation} but rename it to \text{new\_rel\_name} |

| \text{rename\_list} \text{\ input\_relation\_2 \ new\_attr\_names} |
| output the same \text{input\_relation\_2} but rename it to \text{new\_rel\_name} and its attributes to \text{new\_attr\_names} |
RATest

https://ratest.cs.duke.edu/ratest/

A web with the backend radb, where you can test your RA query.

We will try Q(e), Q(f) (, maybe Q(g)) next and you can take (a)-(d) as practice.
RA example 1:

- Bars that drinkers in address “101 W. Main Street” frequent

- Show in RATest
RA example 1:

Result: Correct / Incorrect with counter example

Expression tree

RA Test Status: Correct
RA example 2:

• Bars that drinkers in address “101 W. Main Street” do NOT frequent

• Hint 1. If I know how to get the bars those drinkers frequent, how can I know the bar they don’t frequent?

• Hint 2. Difference(-) might help
RA example 2:

- Bars that drinkers in address “101 W. Main Street” do NOT frequent

**Q(f)**

Bars that drinkers in address “101 W. Main Street” do not frequent

Input your RA query here

```sql
\rename_{\text{bar}}\{\text{project}_{\text{name}}\ \text{bar}\}
\text{diff}
\text{project}_{\text{name}}\{
\text{frequents}
\text{join}_{\text{frequents.drinker = drinker.name}}
\text{select}_{\text{address = ‘101 W. Main Street’}}
drinker
\} ;
```

Run your Q(f)!

RA Test Status: Correct
Let’s make sure that RADB on your VM works

1. RADB returns the result instead of whether it is correct
2. Go back to your vm (“ssh vcm@vcm-xxxxx.vm.duke.edu”)
3. Type in “radb beers”, then enter

```
[vcm@vcm-27266:~]$ radb beers
radb: A relational algebra interpreter over relational databases
Version 3.0.4 by Jun Yang <junyang@cs.duke.edu>
https://github.com/junyang/radb
ra>
```

1. Copy-paste what you write on RATest, then Enter

```
ra> \rename_{bar}(\project_{name} bar)
  .> \diff
  .> \project_{bar}{
      .> \frequents
      .> \join_{frequents.drinker = drinker.name}
      .> (\select_{address = '101 W. Main Street'} drinker)
      .> );
  (bar:string)
```

```
RA example 3:

• For each bar, find the drinkers who frequent it max no. times a week (among all drinkers who frequent that bar)

• Hint 1. Who do NOT visit a bar max no. of times?

• Whose times_of_weeks is lower than somebody else’s for a given bar

• Hint 2. Then, Difference(-) might help
RA example 3:

- For each bar, find the drinkers who frequent it max number times a week