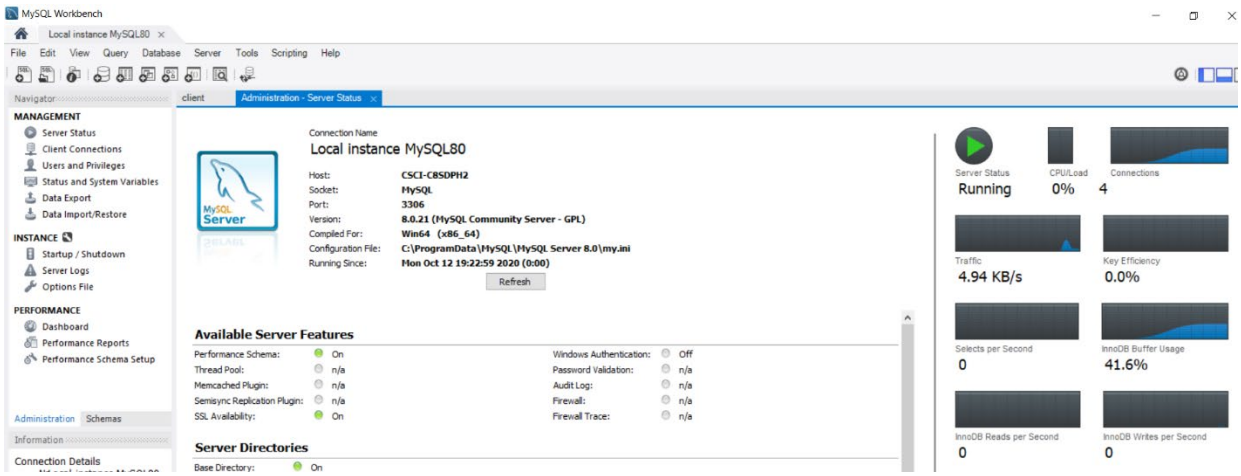
CSCI 332 Database Concepts
G. Pothering
Fall 2020
Laboratory 2 Database Creation in MySQL Using SQL Commands

In this lab we discuss how to create a database in MySQL RDBMS using the data definition features of SQL.

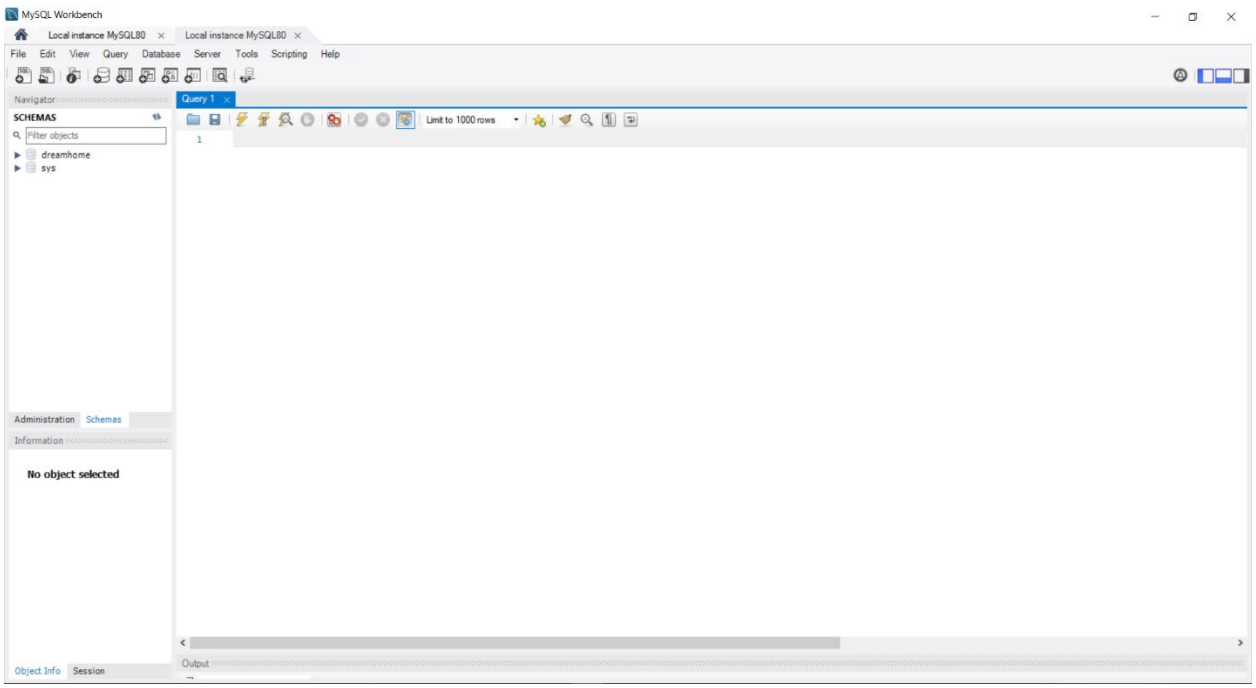
0. Start up the MySQL Workbench, click on the entry for a Local Instance connection, and then check to see that your local server is running.



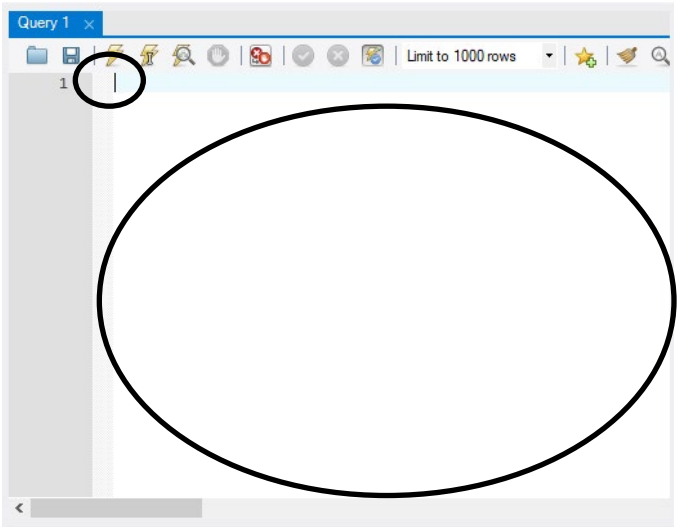
If it is not running, start it. ***See Lab 1 if you need help with this.***

Creating a New Database

1. Click on the Schemas tab (circled above) to bring change to a window that looks similar to the following. Note, I closed the Administration - Server Status window by clicking the "X" in the right-hand side of its tab.

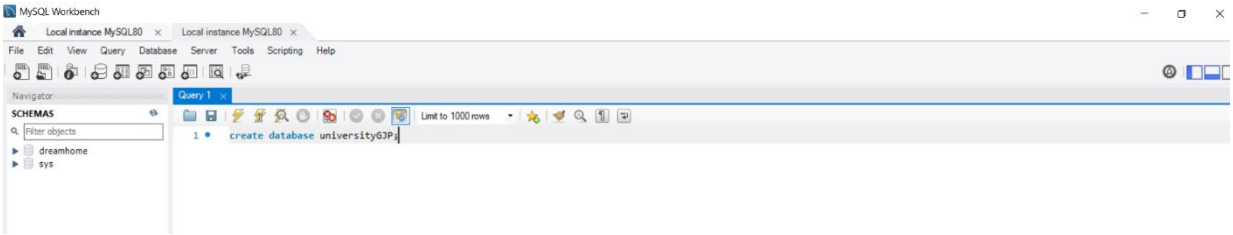


In the Query 1 panel (highlighted below via an inscribed ellipse; your panel may have a different name, depending on prior uses) you will see a cursor (circled above) indicating where you can enter commands both to create a database schema and to create tables in that schema.



We will create a database schema – **university***yourInitials (initials in lowercase)* using SQL data definition commands. In later steps we will create tables for this schema.

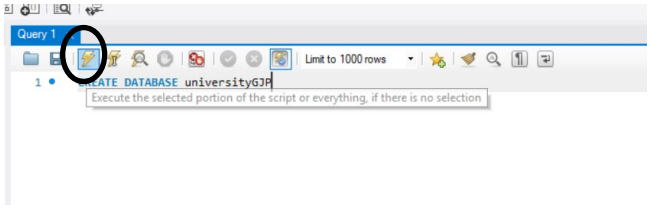
SQL has a CREATE DATABASE command that can be used to create new database schemas.



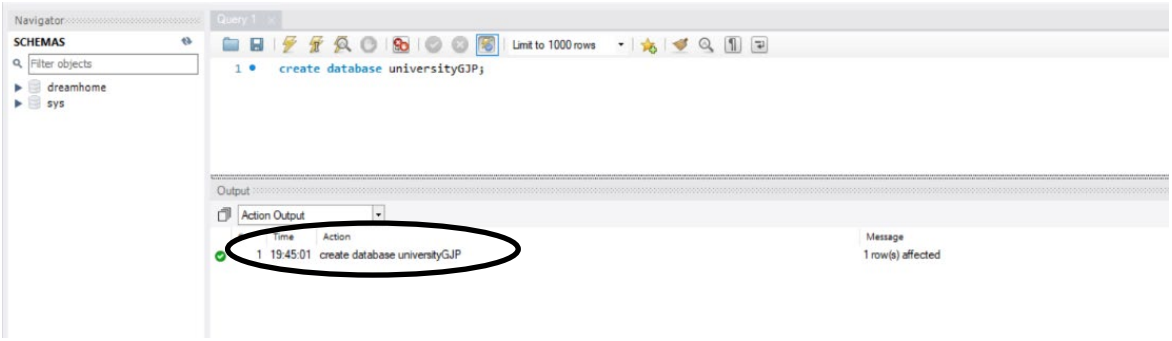
2. In the command panel shown above, enter the command

   ```
   CREATE DATABASE universitygjp;
   ```

   in the first line as we have done (except with your initials instead of mine), and then click the leftmost "lightning bolt" icon (circled below) to execute this command. From now on we'll call this the "***execute icon***," or simply say "***click execute***."



3. You will notice in the **Output** panel at the bottom of the window a message informing us, via the green check (circled in the window snapshot below) that the database was successfully created

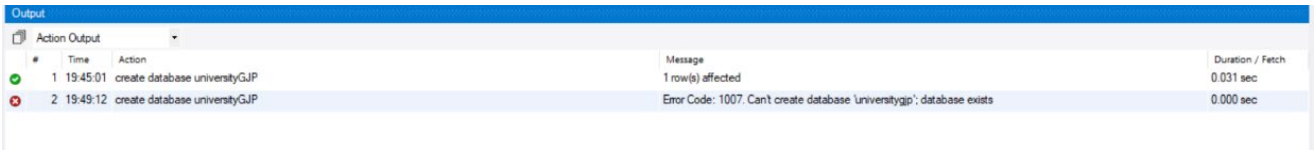


If you do not see the output panel you may have to go to the bottom of the window, place your cursor just above the divider labeled "Output" and the holding the left mouse-button drag upwards to expose the panel.

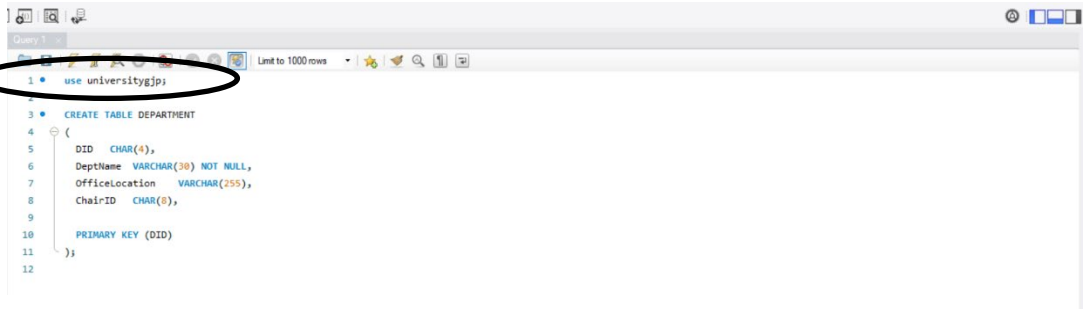
4. Next we are going to use an SQL data definition statement to create our first table, the **DEPARTMENT** table, for the **universitygjp** schema. We can't just issue a CREATE TABLE command, however, since the database management system has no way of knowing which schema the table is to be part of. Consequently we must issue a ***USE*** command to specify the schema we wish to use (again, don't forget to use ***your*** initials, not mine)

```
USE universitygjp;
```

For now, type this command so that it ***replaces the CREATE DATABASE command; otherwise Workbench will complain in the output panel, that the database already exists.



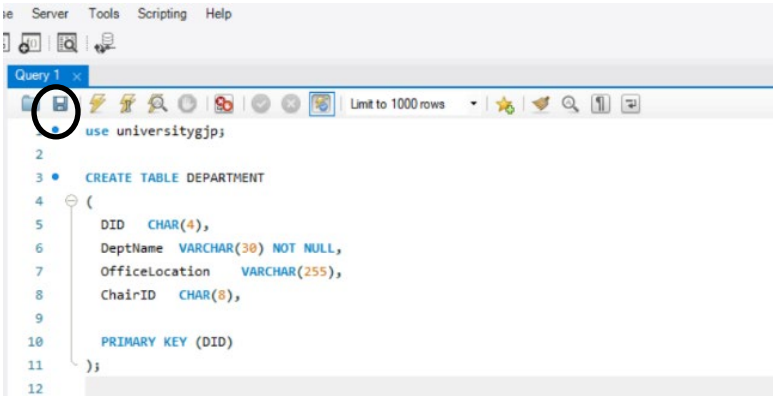
On subsequent lines we can now enter our first CREATE TABLE statement, which is taken from page 4 of Chapter 4 of our notes. When you have completed entering this code, the center panel should appear similar to the following. Note that I have placed a blank line after the USE command. You may do this, or not, as you wish. I find that it makes things easier to read.



this must ***replace*** the CREATE DATABASE line

If you **click on execute** (remember, it's the lightning bolt icon) you will create the table (assuming there are no errors). The Output panel will indicate this.

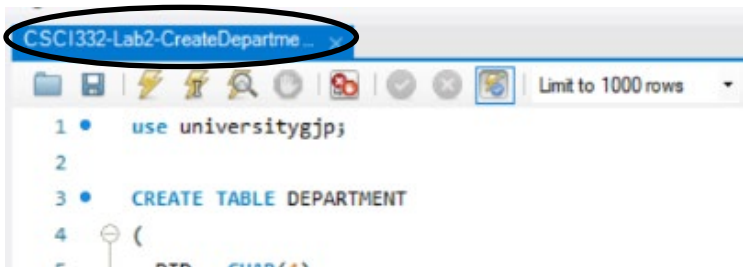
5. In order that I can check your progress through this laboratory we are going to save this work. To do so click on the **Save** icon in the panel in which you have been working (see below). Again, my panel is labeled ***Query 1***; some of you may have a panel labeled as something else.



This will bring up a Save SQL Script window in which you can supply a name for your script and navigate to a directory where you want to save it. I want you to name your script

CSCI332-Lab2-CreateDepartment-*YourInitials*

(a suffix .sql will automatically be appended for you so you don't have to type in that part). Be sure you save this script in a folder where you can find it later because you will eventually have to upload this to OAKS. Shown below is what this would look like for me. Click **Save.**



You will note that the name of the tab has been changed (circled below).

Note: If there are any errors in a line of our script, or if we attempt to create a database using the name of an existing database, or if we attempt to re-define an existing table, we will find out about them in the Output panel. This is always an important run-time feature, so consult with it regularly.

For the rest of this laboratory exercise we are going to complete the definition of the **university** schema. There are three ways you can procede:

i. You can delete the lines for CREATE TABLE DEPARTMENT and in their place enter **all** of the remaining CREATE TABLE statements. If you do this, save the entire script under the name **CSCI332-Lab2-RemainingTables-***yourInitials*.**sql**.

ii. Alternatively you can create the remaining tables one-at-a-time by overwriting the previous script in the same tab (but keep the ***USE university***... statement). If you do this, be sure to save each script before moving on to the next. Use the name format

CSCI332-Lab2-Create*Tablename-YourInitials*.sql

for example for the script for creating the FACULTY table, I would save it as

CSCI332-Lab2-CreateFaculty-gjp.sql

iii. You can create each script in its own SQL script tab and execute and save it from there. To do so go to ***File***, then select the ***New Query Tab*** entry (or simply hold the Control key and then hit the T key)

There is one problem with the first approach. If a table cannot be created because of a syntax error (perhaps you left out a comma or right parenthesis, or mistyped a type name) you can correct the error and rerun the sequence of commands, but if any tables were successfully created prior to the one that had the error, you will get an error on a subsequent run because these tables already exist. To help avoid such errors, there is a variation of the CREATE TABLE line that you can (indeed, should) use:

```
CREATE TABLE IF NOT EXISTS TableName[1]
```

If we use this variation, then the statements to create the FACULTY table would look like the following

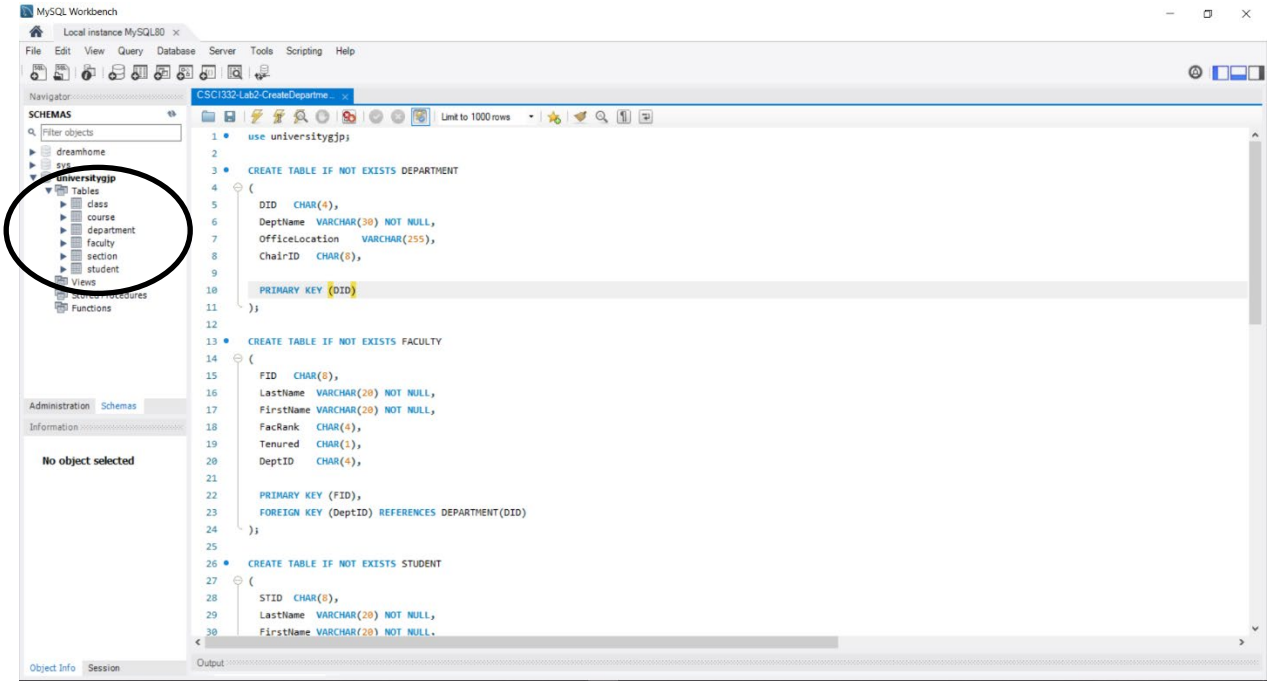
```
CREATE TABLE IF NOT EXISTS FACULTY
(
  FID              CHAR(8),
  LastName         VARCHAR(20)      NOT NULL,
  FirstName        VARCHAR(20)      NOT NULL,
  FacRank          CHAR(4),
  Tenured          CHAR(1),
  DeptID           CHAR(4),

  PRIMARY KEY (FID),
  FOREIGN KEY (DeptID) REFERENCES DEPARTMENT(DID)[2]
);
```

6. Proceed to create the remaining tables of the **university** database, using one of the above approaches. Note, effective with MySQL 8.0, the name "Rank" that we used in the Faculty table became a reserved word and cannot be used as an attribute name. We will change the name here to FacRank.

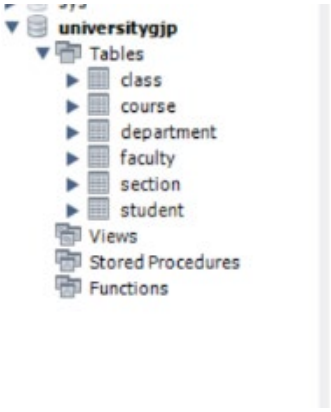
If your scripts executed successfully, you should see the new database and its tables appear in the *Schemata* panel on the right-hand side of the browser. If you do not see them, but you know the

[1] There is also a variation of the CREATE DATABASE command, **CREATE DATABASE IF NOT EXISTS** which can also be used to create a schema, and is also generally preferred command to use.

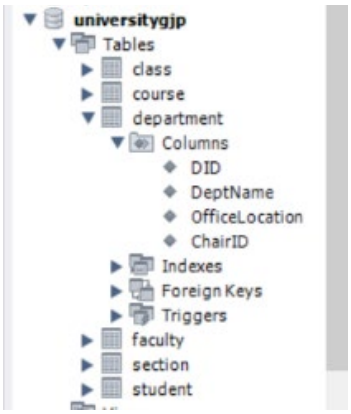
code executed successfully, you can try placing the cursor in the *Schemata* panel, right-click, and then select ***Refresh All.*** The following window shows this (circled), as well as the script tabs we created.



When you look in the ***Schemas*** panel in the left side of the Workbench window you may have noticed that the name of the "Department" table is in all lower-case, even though we defined it in upper-case. This is a default action that MySQL takes. We can force MySQL to use all upper-case by placing the table name in parentheses, but we shall not insist on this.



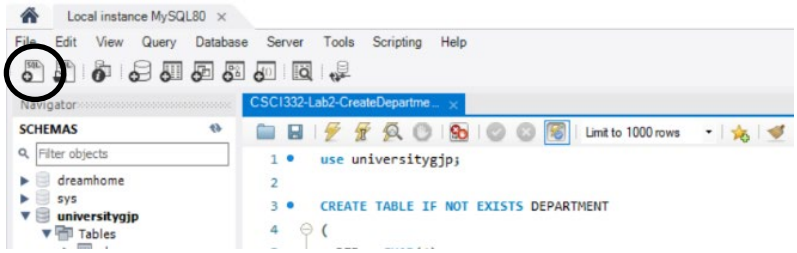
If you further expand **department** by clicking on the arrow to the left of its name and then doing the same for the ***Columns*** entry you will notice that the attribute/column names reflect the mixed cases we used when we specified them (see below)



Although officially SQL is case-insensitive when it comes to schema, table, and column names, some vendors default to, and enforce, case-sensitivity unless directed not to. We are not going to concern ourselves with this here (we have enough other issues to worry about), but do want to alert you to the issue.

To conclude this laboratory activity, we execute a final SQL script to alter the DEPARTMENT table to indicate that the attribute ChairID is a foreign key that references the FID attribute of the FACULTY table.

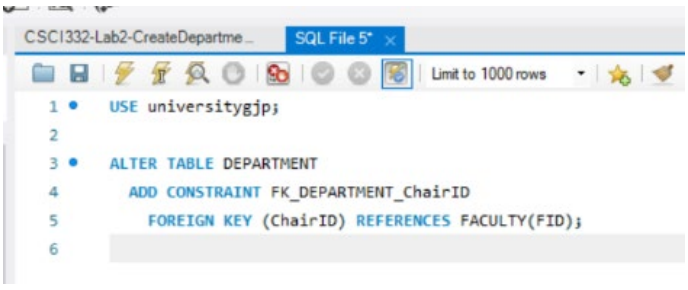
7. First create a new tab for entering an SQL script by clicking on the icon circled below



8. In the ensuing window, enter the following SQL script, taking care to precede it with a **USE university*yourInitials*;** statement.

```
USE universitygjp;

ALTER TABLE DEPARTMENT
  ADD CONSTRAINT FK_DEPARTMENT_ChairID
    FOREIGN KEY (ChairID) REFERENCES FACULTY(FID);
```

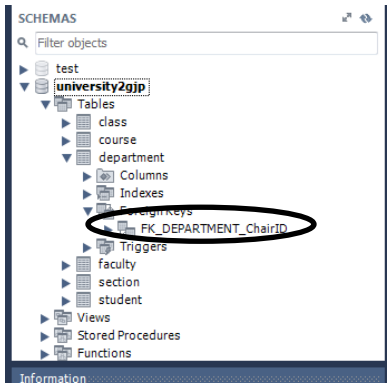
The tab is shown below



Save your script as

CSCI332-Lab2-AlterDepartment-*yourInitials*.sql.

If you now examine the **department** entry in the Object Browser panel and expand the Foreign Keys entry, you will see an entry for the foreign key we just created (circled below)



This concludes Laboratory 2. You should upload all of the script files you saved in this lab to the Lab 2 dropbox on OAKS.