CECS 323 LAB DATAGRIP & DERBY INSTALLATION

OBJECTIVE:

Walk you through the process of installing DataGrip and the Derby Relational Database Management System on your machine so that you can perform the labs in the rest of the semester.

INTRODUCTION:

There are many Relational Database Management Systems (RDBMs) available today. All of them implement the ANSI standards to one extent or another. Derby is an open-source Java based RDBMS that is lightweight to install, and free of charge to use.

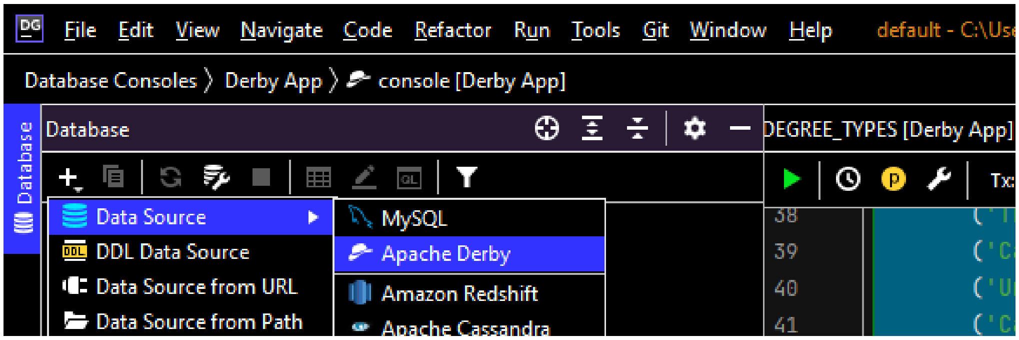
There are also many Integrated Development Environments (IDEs) that “sit on top of” the RDBMS to provide tools that make the RDBMS easier to work with. Some of those IDEs are proprietary, like the MySQL workbench that just works with MySQL. Many IDEs are multi-purpose because they will work with several RDBMs. NetBeans and IntelliJ are examples of this later class of IDEs. Both IDEs provide the developer with all of the tools that they need to write 3rd generation language code in languages like Java, as well as tools for directly manipulating the database itself.

DataGrip is offered by the same company that makes IntelliJ, and IntelliJ includes the functions and interface of DataGrip. We will use DataGrip during the first part of this semester to build tables, populate those tables, and manipulate the data in those tables directly. This will give us a “leg up” when we start using IntelliJ for the Java Persistence API coding project later in the semester.

The vendor for both DataGrip and IntelliJ is JetBrains. JetBrains offers free licenses for both products to students and instructors.

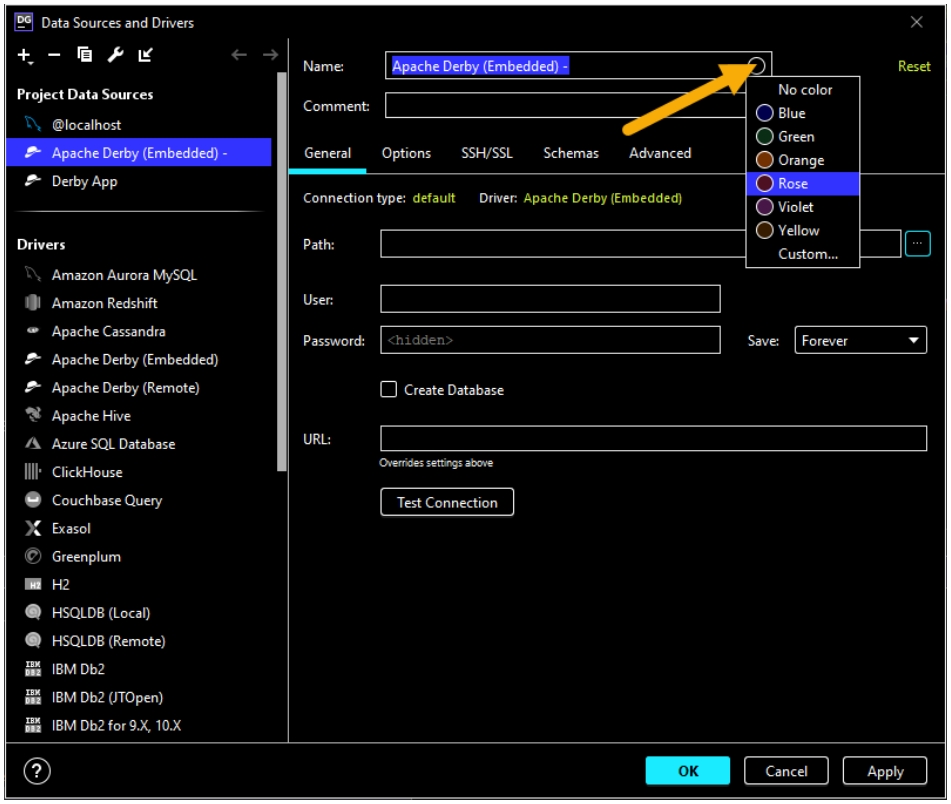
PROCEDURE:

1. Install DataGrip:
   a. Go to https://www.jetbrains.com/shop/eform/students and fill out the form to apply.
   b. Go to: https://www.jetbrains.com/datagrip/quick-start for a quick tour of getting started using DataGrip.
2. Build your first Derby Data Source
   a. Once you have installed and opened DataGrip, press Alt+1 to open the Database window within DataGrip.
   b. Press the “+” key in the upper left-hand corner of the Database window, and select the Apache Derby database as your Data Source:



c. That will put you into a dialog to define your Derby database. Derby comes in two flavors: server and embedded. The server style has a separate process that interacts with the database. We will use the embedded style since it is simpler to configure.

i. The dialog allows you to apply a color to your data sources. Once you start adding several of them to DataGrip, the coloring is a handy way to keep them straight. You can see the button to select the color highlighted by an arrow here:



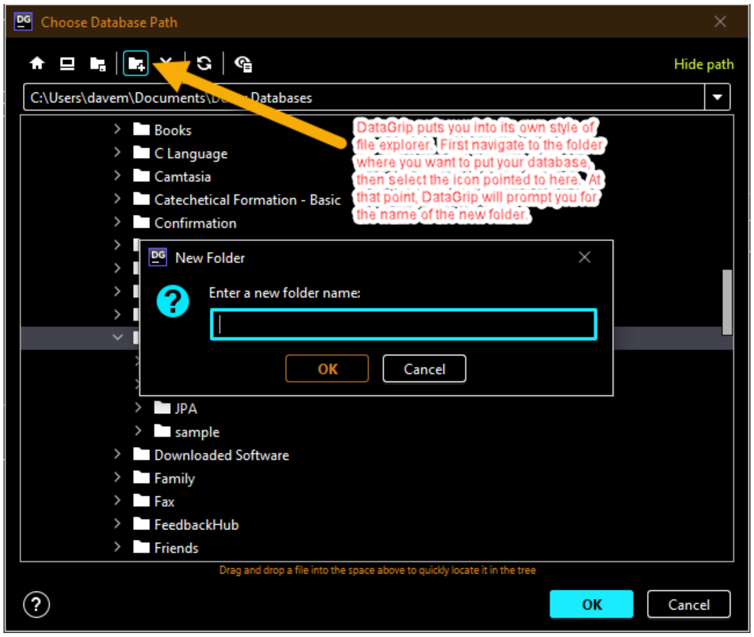
Name: Apache Derby (Embedded) -

Connection type: default Driver: Apache Derby (Embedded)

Password: <hidden>

Save: Forever

ii. The path points to the folder where the Derby database files will go. Select the ellipsis (…) to the right of the path control. That will produce a file explorer window like:



C:\Users\davem\Documents\D...\Databases

DataGrip puts you into its own style of file explorer. First navigate to the folder where you want to put your database, then select the icon pointed to here. At that point, DataGrip will prompt you for the name of the new folder.

Enter a new folder name:

Drag and drop a file into the space above to quickly locate it in the tree

iii. Click the folder with the "+" in the lower right-hand corner as indicated above, and enter the folder name for your new database. DataGrip will create that folder for you. You can then add another folder under that if you want to. Select the folder where you want to construct the database and select "OK".

   1. Note that Derby will **create a new folder** that you point to in the path parameter, and it will place the database files in **that** folder.
   2. If you try to create a new database in an existing folder, Derby will **not** create the database. Derby needs to have that folder all to itself, and the easiest way for it to do that is to create the folder as part of the database creation process.
   3. To remove a Derby database, simply delete its folder. There is no other metadata captured about it elsewhere.

iv. The name of the Data Source will default to the name of the folder, but you can override the name of the Data Source if you wish.

v. Derby does not require that you log into the database with any authentication at all if you do not want that. Since your Derby database will reside on your PC, security is not going to be terribly important, so you can leave the User and the Password fields blank for now.

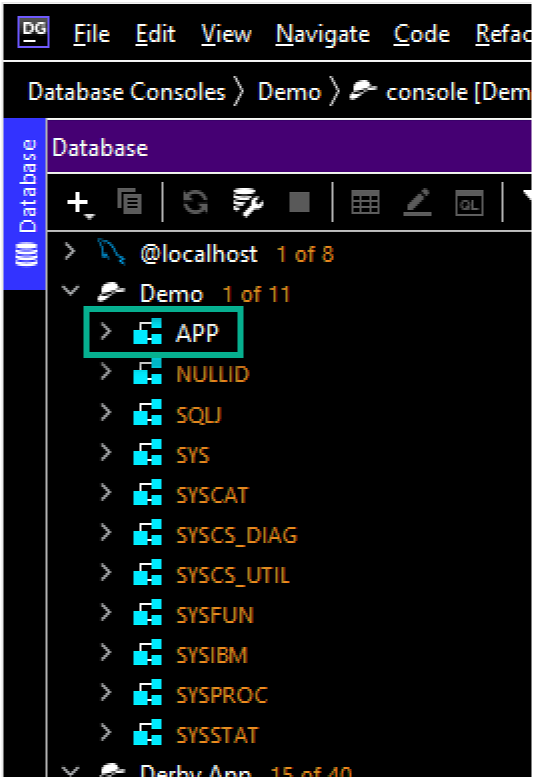
vi. The "Create Database" checkbox needs to be checked to tell DataGrip to instruct Derby to create the database for you.

vii. Finally, select the "Test Connection" button to make sure that you can connect to the new database.

   1. At this point, DataGrip will prompt you telling you that you need to download drivers. Let it do that for you. You only need to do it the first time that you access a Derby database.
   2. Select the "OK" button to finish the wizard.

d. Interacting with your new database:

i. In my case, I called my new database “Demo”. Click the > sign next to your database in the Database window to get:

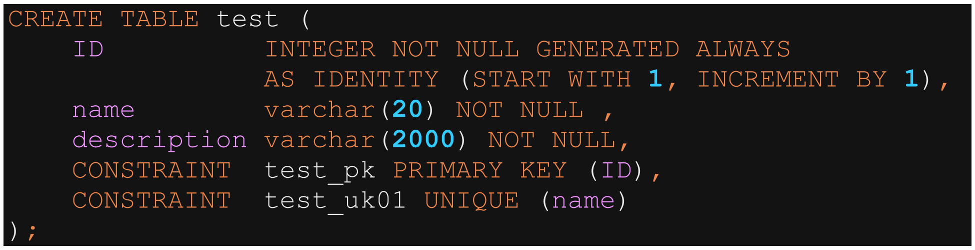


ii. I have highlighted the APP entry in the display for two reasons:

   1. In the absence of any other account, Derby will always create the APP schema and make that the default.
   2. The other entries shown here are schemas that Derby uses to administer the data about the data in the database. These schemas are collectively referred to as the data dictionary of the database.

iii. Note that you can create additional schemas of your own any time that you wish to. Personally, I have a schema for each lab that I have built in Derby to help me keep things organized. The scheme is a namespace within the database. Every object in the database, regardless of whether it is a table, a constraint, an index, a view, … belongs to **some** schema.

iv. In the console (large window in the middle of the DataGrip window, type in the following:



```
CREATE TABLE test (
    ID          INTEGER NOT NULL GENERATED ALWAYS
                AS IDENTITY (START WITH 1, INCREMENT BY 1),
    name        varchar(20) NOT NULL ,
    description varchar(2000) NOT NULL,
    CONSTRAINT  test_pk PRIMARY KEY (ID),
    CONSTRAINT  test_uk01 UNIQUE (name)
);
```

1. This creates a table with a surrogate ID, called ID curiously enough, that will uniquely identify the rows in this table. The database itself generates the values for the ID column, we do not have to supply those.
2. Name and description are additional columns that we do not want to every go without a value.
3. We establish name as a candidate key to make sure that we never get two of these with the same value for name.

v. Then type in:

```
INSERT INTO test (name, description)
VALUES       ('Diet Pepsi',
          'A refreshing beverage that won\''t leave you
bloated'),
         ('Mountain Dew',
          'A citrisy beverage with a caffeine wallop');
```

1. This inserts two rows into our demo table.

vi. Finally, select the data from the test table and cut/paste it into a document to turn in.

1. Enter the statement “select * from test;” and run it.
2. Right click in the output area in any of the cells.
3. Select the “export to clipboard” option.
4. Past from your clipboard into your report document for the lab.

WHAT TO TURN IN:

- The sample output from your select statement.