

How-To: Work with the Java DB (Derby) Database in Netbeans 12

This document demonstrates how to set up a connection to [Java DB](#) database in NetBeans IDE. Once a connection is made, you can begin working with the database in the IDE, allowing you to create tables, populate them with data, run SQL statements and queries, and more.

The Java DB database is Sun's supported distribution of [Apache Derby](#). Java DB is a fully transactional, secure, standards-based database server, written entirely in Java, and fully supports SQL, JDBC API, and Java EE technology. The Java DB database is packaged with the [GlassFish](#) application server, and is included in [JDK 6](#) as well. For more information on Java DB database, consult the [official documentation](#).

Java DB is installed when you install JDK 7 or JDK 8 (except on Mac OS X). If you are using Mac OS X you can download and install [Java DB](#) manually or use the Java DB that is installed by Java EE version of the NetBeans IDE installer.

Configuring the Database

If you have the GlassFish Server registered in your NetBeans IDE installation, Java DB will already be registered for you. Therefore, you can skip ahead to [Starting the Server and Creating a Database](#).

If you downloaded the GlassFish server separately and need help registering it in NetBeans IDE, see **Registering a GlassFish Server Instance** in the IDE's Help Contents (F1).

If you just downloaded Java DB on its own, perform the following steps.

1. Run the self-extracting file. A folder named 'javadb' will be created in the same location as the file. If you just downloaded Java DB and want to have the database server reside in a different location than where it was extracted to, you should relocate it now.

2. On your system, create a new directory to be used as a home directory for the individual instances of the database server. For example, you can create this folder in the Java DB root directory (javadb) or in any other location.

Before continuing further, it is important to understand the components found in Java DB's root directory:

- The `demo` subdirectory contains the demonstration programs.
- The `bin` subdirectory contains the scripts for executing utilities and setting up the environment.
- The `javadoc` subdirectory contains the API documentation that was generated from source code comments.
- The `docs` subdirectory contains the Java DB documentation.
- The `lib` subdirectory contains the Java DB jar files.

Registering the Database in NetBeans IDE

Now that the database is configured, perform the following steps to register Java DB in the IDE.

1. In the Services window, right-click the Java DB Database node and choose Properties to open the Java DB Settings dialog box.
2. For the Java DB Installation text field, enter the path to the Java DB root directory (javadb) that you specified in the previous step.
3. For Database Location, use the default location if a location is already provided. Click OK

For example, the default location might look like `c:\Documents and Settings\username\.netbeans-derby` on a Windows machine.

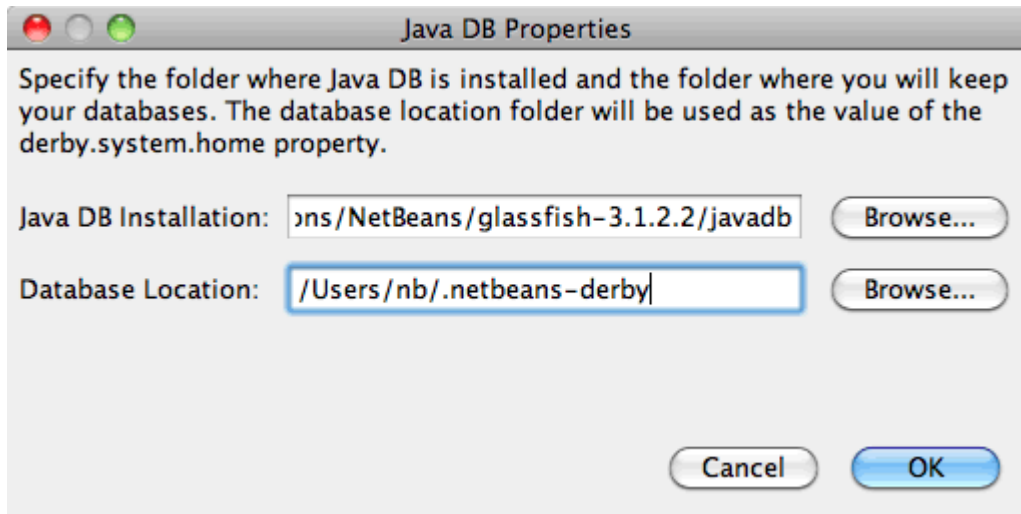
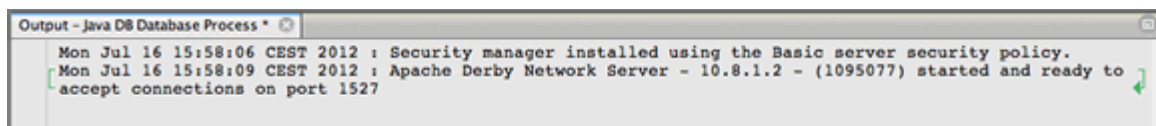


Figure 1. example of default settings of Java DB server and database
If the Database Location field is empty you will need to set the path to the directory that contains your databases. You will need to create a directory for the databases if no directory exists.

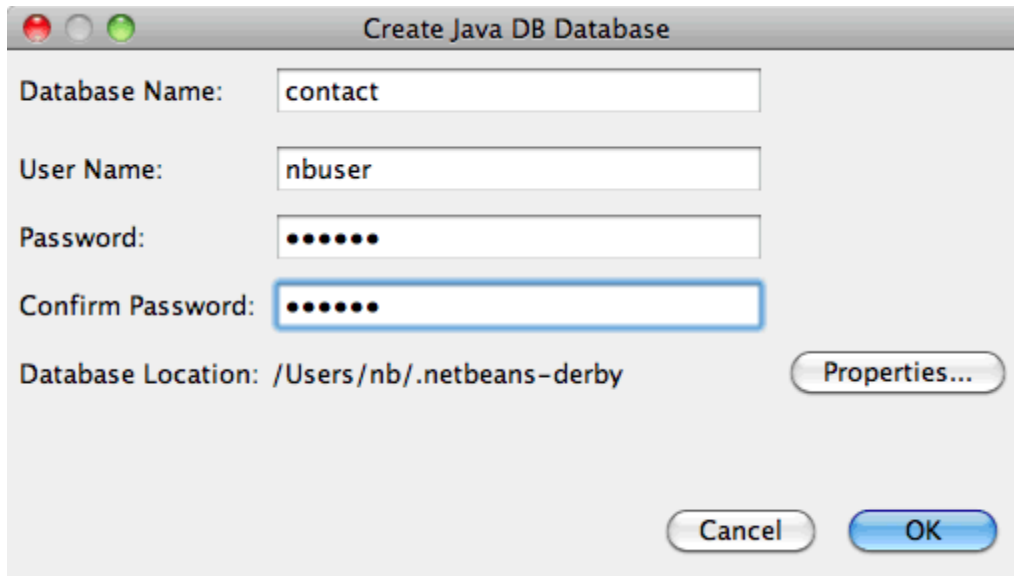
Starting the Server and Creating a Database

The Java DB Database menu options are displayed when you right-click the Java DB node in the Services window. This contextual menu items allow you to start and stop the database server, create a new database instance, as well as register database servers in the IDE (as demonstrated in the previous step). To start the database server:

1. In the Services window, right-click the Java DB node and choose Start Server. Note the following output in the Output window, indicating that the server has started:



2. Right-click the Java DB node and choose Create Database to open the Create Java DB Database dialog.
3. Type contact for the Database Name.
4. Type nbuser for the User Name and Password. Click OK.



The Database Location is the default location set during installation of Java DB from GlassFish. If you installed Java DB separately, this location might be different.

After you create the database, if you expand the Databases node in the Services window you can see that the IDE created a database connection and that the database was added to the list under the Java DB node.


Connecting to the Database

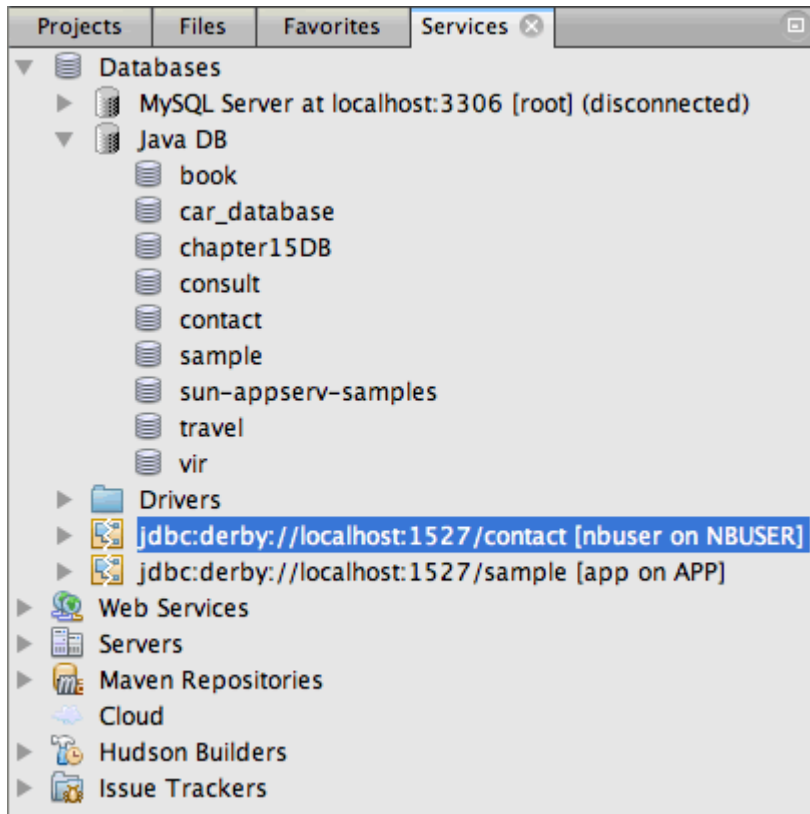
So far, you have successfully started the the database server and created a database instance named `contact` in the IDE. In the Services window of the IDE you can perform the following common tasks on database structures.

- creating, deleting, modifying tables
- populating tables with data
- viewing tabular data
- executing SQL statements and queries

In order to begin working with the `contact` database, you need to create a connection to it. To connect to the `contact` database perform the following steps.


1. Expand the Databases node in the Services window and locate the new database and the database connection nodes.

The database connection node() is displayed under the Databases node. The name of the database is displayed under the Java DB node.



You will also see the `sample [app on APP]` database connection that is the default database schema.

1. Right-click the **contact** database connection node (`jdbc:derby://localhost:1527/contact [nbusser on NBUSER]`) and choose Connect.

The connection node icon appears whole (), signifying that the connection was successful.

1. Create a convenient display name for the database by right-clicking the database connection node (`jdbc:derby://localhost:1527/contact [nbusser on NBUSER]`) and choosing Rename. Type `contact DB` in the text field and click OK.

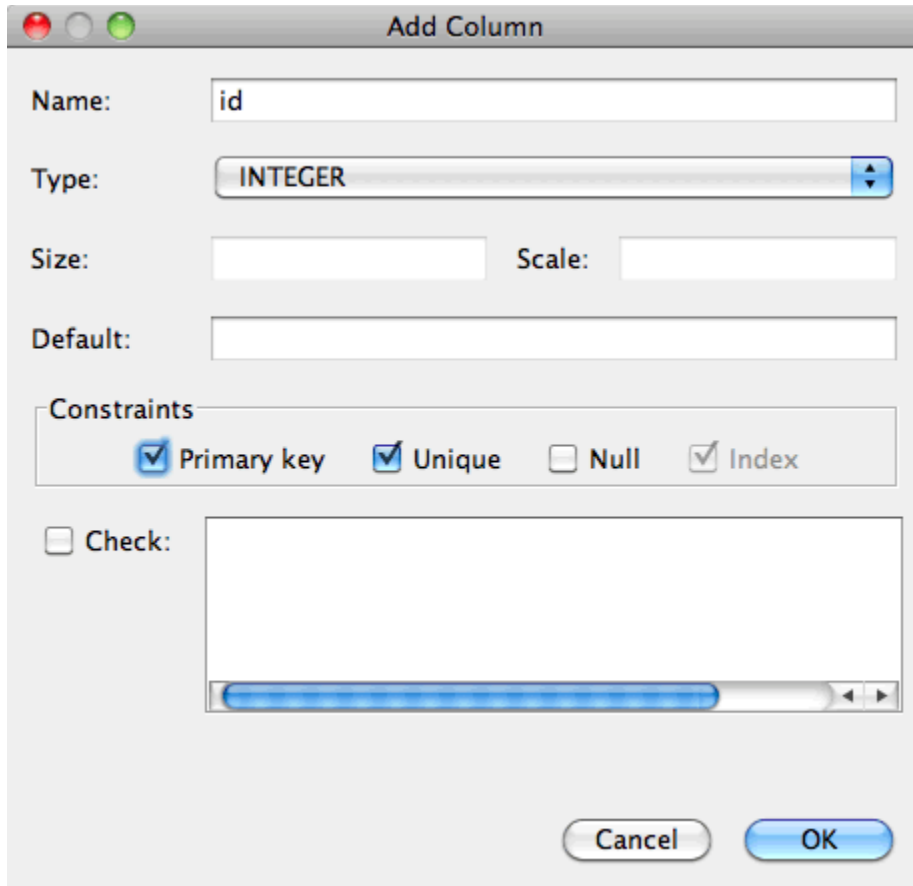
Creating Tables

The `contact` database that you just created is currently empty. It does not yet contain any tables or data. In NetBeans IDE you can add a database table by either using the Create Table dialog, or by inputting an SQL statement and running it directly from the SQL Editor. You can explore both methods:

- [Using the Create Table Dialog](#)
- [Using the SQL Editor](#)

Using the Create Table Dialog

1. Expand the `contact` DB connection node and note that there are several schema subnodes. The `app` schema is the only schema that applies to this tutorial. Right-click the `APP` node and choose `Set as Default Schema`.
2. Expand the `APP` node and note that there are three subfolders: `Tables`, `Views` and `Procedures`. Right-click the `Tables` node and choose `Create Table` to open the `Create Table` dialog box.
3. In the `Table Name` text field, type `FRIENDS`.
4. Click `Add Column`. The `Add Column` dialog box appears.
5. For `Column Name`, enter `id`. For `Data Type`, select `INTEGER` from the drop-down list.
6. Under `Constraints`, select the `Primary Key` checkbox to specify that this column is the primary key for your table. All tables found in relational databases must contain a primary key. Note that when you select the `Primary Key` check box, the `Index` and `Unique` check boxes are also automatically selected and the `Null` check box is deselected. This is because primary keys are used to identify a unique row in the database, and by default are used as the table index. Because all rows must be identified, primary keys cannot contain a `Null` value.

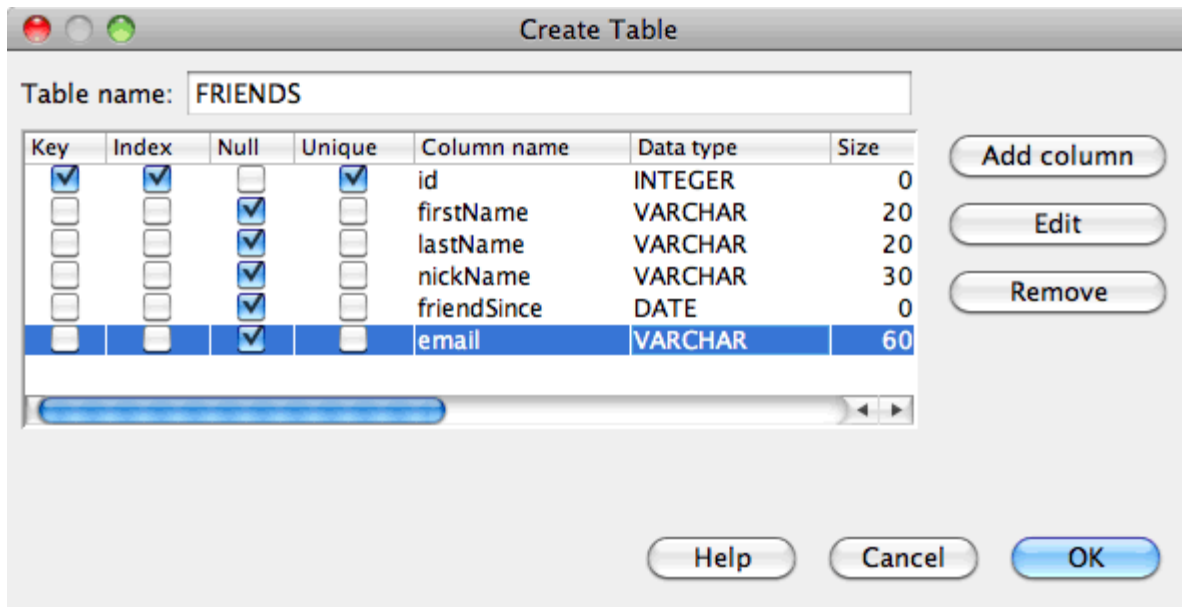




1. Repeat this procedure now by specifying fields as shown in the table below:

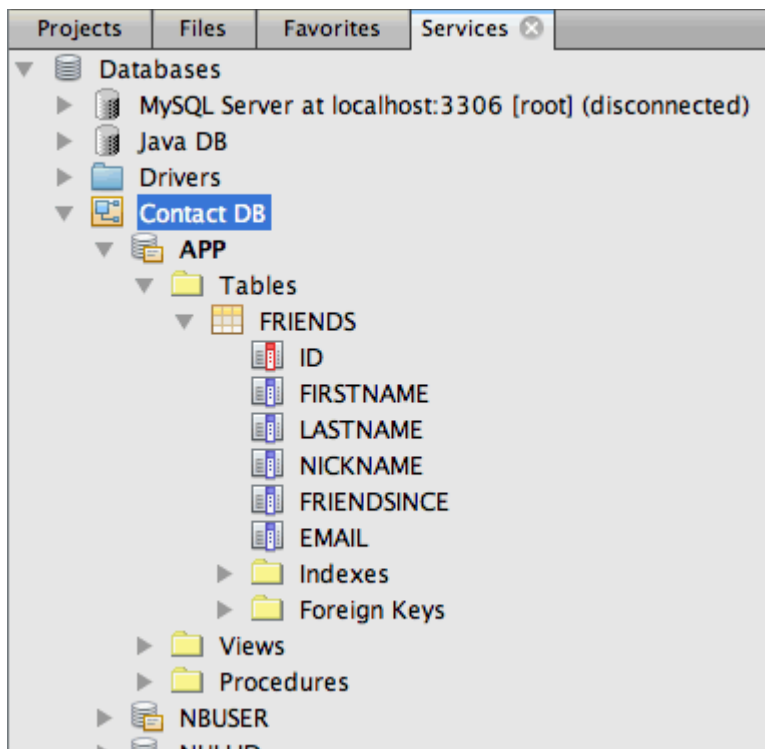
Key	Index	Null	Unique	Column name	Data type	Size
[checked]	[checked]	[checked]		id	INTEGER	0 [checked]
			[checked]	firstName	VARCHAR20	
				lastName	VARCHAR20	
[checked]			[checked]	nickName	VARCHAR30	
				friendSince	DATE	

You are creating a table named `FRIENDS` that holds the following data for each contact record:

- **First Name**
- **Last Name**
- **Nick Name**
- **Friend Since Date**
- **Email Address**



1. When you are sure that your Create Table dialog contains the same specifications as those shown above, click OK. The IDE generates the FRIENDS table in the database, and you can see a new FRIENDS table node () display under the Tables node. Beneath the table node the columns (fields) are listed, starting with the primary key ().




Using the SQL Editor:


1. In the Service window, either right-click the contact DB connection node or the Tables node beneath it and choose Execute Command. A blank canvas opens in the SQL Editor in the main window.
2. Enter the following query in the SQL Editor. This is a table definition for the COLLEAGUES table you are about to create:

```
CREATE TABLE "COLLEAGUES" (  
    "ID" INTEGER not null primary key,  
    "FIRSTNAME" VARCHAR(30),  
    "LASTNAME" VARCHAR(30),  
    "TITLE" VARCHAR(10),  
    "DEPARTMENT" VARCHAR(20),  
    "EMAIL" VARCHAR(60)  
);
```

Statements and queries formed in the SQL Editor are parsed in Structured Query Language. SQL adheres to strict syntax rules which you should be familiar with when working in the IDE's editor. SQL syntax can also differ depending on the database management system. See the [JavaDB Reference Manual](#) for specific guidelines.

1. Click the Run SQL () button in the task bar at the top of the editor (Ctrl-Shift-E) to execute the query. In the Output window (Ctrl-4), a message displays indicating that the statement was successfully executed.

```
Executed successfully in 0.162 s, 0 rows affected.  
Line 1, column 1  
  
Execution finished after 0.162 s, 0 error(s) occurred.
```

1. To verify changes, right-click the contact DB connection node in the Services window and choose Refresh. This updates the Runtime UI component to the current status of the specified database. This step is necessary when running queries from the SQL Editor in NetBeans IDE. Note that the new COLLEAGUES table node () now displays under Tables in the Services window.

Adding Table Data

Now that you have created one or more tables in the contact database, you can start populating it with data. There are several ways that you can add records to your table.

- [Write an SQL statement](#) in the SQL Editor that supplies a value for every field present in the table schema.
- [Use the SQL Editor](#) to add records to the table.
- [Use an external SQL script](#) to import records to the table.

Read the sections below to learn how to use all these methods of populating the FRIENDS table with data.

Running an SQL Statement

1. Expand the Tables under the contact DB node in the Services window, right-click the FRIENDS table and choose Execute Command to open the SQL Editor window.
2. In the SQL Editor, enter the following statement.

```
INSERT INTO APP.FRIENDS VALUES (1, 'Theodore', 'Bagwell', 'T-Bag', '2004-12-25', 'tbag@foxriver.com')
```

While you are typing, you can use the SQL Editor code completion.

1. Right-click inside the SQL Editor and choose Run Statement. The Output window displays a message indicating that the statement was successfully executed.
2. To verify that the new record has been added to the FRIENDS table, right-click the FRIENDS table node in the Services window and choose View Data.

When you choose View Data, a query to select all the data from the table is automatically generated in the upper pane of the SQL Editor. The results of the statement are displayed in the lower pane of the SQL Editor. In this case, the FRIENDS table displays in the lower pane. Note that a new row has been added with the data you just supplied from the SQL statement.

select * from APP.FRIENDS

Page Size: 20 Total Rows: 1 Page: 1 of 1 Matching Rows:

#	ID	FIRSTNAME	LASTNAME	NICKNAME	FRIENDSINCE	EMAIL
1	1	Theodore	Bagwell	T-Bag	2004-12-25	tbag@foxriver.com

Using the SQL Editor

1. Right-click the FRIENDS table node and choose View Data (if you have not done this at the last step of the previous section).
2. Click the Insert Record(s) (Alt-I) button to add a row. The Insert Records dialog box appears.
3. Click in each cell and enter records. Note that for the cells with Date data type, you can choose a date from the calendar. Click OK when you are done.

Insert Record(s)

Press CTRL+Tab to exit data entry mode from the table. Press CTRL+0 to set NULL value and CTRL+1 to set DEFAULT value for a given column.

#	ID	FIRSTNAME	LASTNAME	NICKNAME	FRIENDSINCE	EMAIL
1	2	Justin	Smith	Just	2012-07-16	<NULL>

July 2012

Sun	Mon	Tue	Wed	Thu	Fri	Sat
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30	31				

Today is 16 July 2012

In the SQL Editor, you can sort the results by clicking on a row header, modify and delete existing records, and see the SQL script for the actions you are doing in the editor (the Show SQL Script command from the pop-up menu).

Deleting Tables

In the following step, you use an external SQL script to create a new COLLEAGUES table. However, you just created a COLLEAGUES table in the [Using the](#)

[SQL Editor](#) section above. In order to make it clear that the SQL script indeed creates a new table, you can delete the already created COLLEAGUES table now. To delete a database table perform the following steps.

1. Expand the Tables node under the database connection node in the Services window.
2. Right-click the table that you want to delete and choose Delete.

Using an External SQL Script

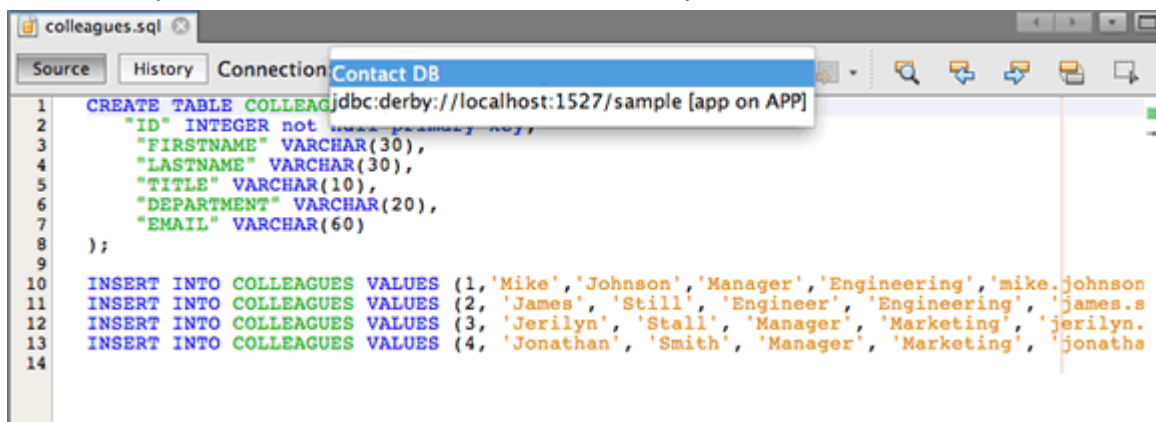
Issuing commands from an external SQL script is a popular way to manage your database. You may have already created an SQL script elsewhere, and want to import it into NetBeans IDE to run it on a specified database.

In this exercise the script will create a new table named COLLEAGUES and populate it with data. Perform the following steps to run the script on the contact database.

1. Download [colleagues.sql](#) to your local system
2. Choose File > Open File from the IDE's main menu. In the file browser navigate to the location of the saved colleagues.sql file and click Open. The script automatically opens in the SQL Editor.


Alternatively, you can copy the contents of [colleagues.sql](#) and then open the SQL editor and paste the contents of the file into the SQL editor.

1. Make sure your connection to Contact DB is selected from the Connection drop-down box in the tool bar at the top of the editor.



The screenshot shows the NetBeans SQL Editor interface. The title bar indicates the file is 'colleagues.sql'. The 'Connection' dropdown menu is set to 'Contact DB'. The SQL script is as follows:

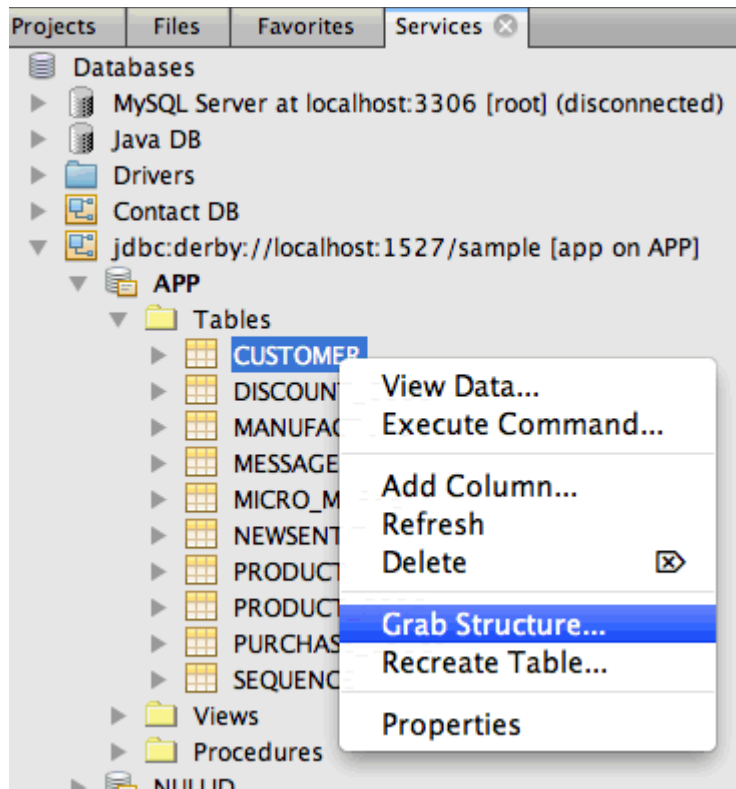
```
1 CREATE TABLE COLLEAGUES (
2   "ID" INTEGER NOT NULL,
3   "FIRSTNAME" VARCHAR(30),
4   "LASTNAME" VARCHAR(30),
5   "TITLE" VARCHAR(10),
6   "DEPARTMENT" VARCHAR(20),
7   "EMAIL" VARCHAR(60)
8 );
9
10 INSERT INTO COLLEAGUES VALUES (1, 'Mike', 'Johnson', 'Manager', 'Engineering', 'mike.johnson
11 INSERT INTO COLLEAGUES VALUES (2, 'James', 'Still', 'Engineer', 'Engineering', 'James.s
12 INSERT INTO COLLEAGUES VALUES (3, 'Jerilyn', 'Stall', 'Manager', 'Marketing', 'jerilyn.
13 INSERT INTO COLLEAGUES VALUES (4, 'Jonathan', 'Smith', 'Manager', 'Marketing', 'jonatha
14
```

1. Click the Run SQL () button in the SQL Editor's task bar. The script is executed against the selected database, and any feedback is generated in the Output window.
2. To verify changes, right-click the `contact` DB connection node in the Services window and choose Refresh. Note that the new `COLLEAGUES` table from the SQL script now displays as a table node under `contact` in the Services window.
3. To view the data contained in the new tables, right-click the `COLLEAGUES` table and choose View Data. In this manner, you can also compare the tabular data with the data contained in the SQL script to see that they match.

Recreating Tables from a Different Database

If you have a table from another database which you would like to recreate in the database you are working in from NetBeans IDE, the IDE offers a handy tool for this. You first need to have the second database registered in the IDE, similar to what was described at the beginning of this tutorial. For the purposes of this tutorial, use the `sample` database that comes packaged with Java DB. This process is essentially carried out in two parts: You first 'grab' the table definition of the selected table, then you can recreate the table in your chosen database:

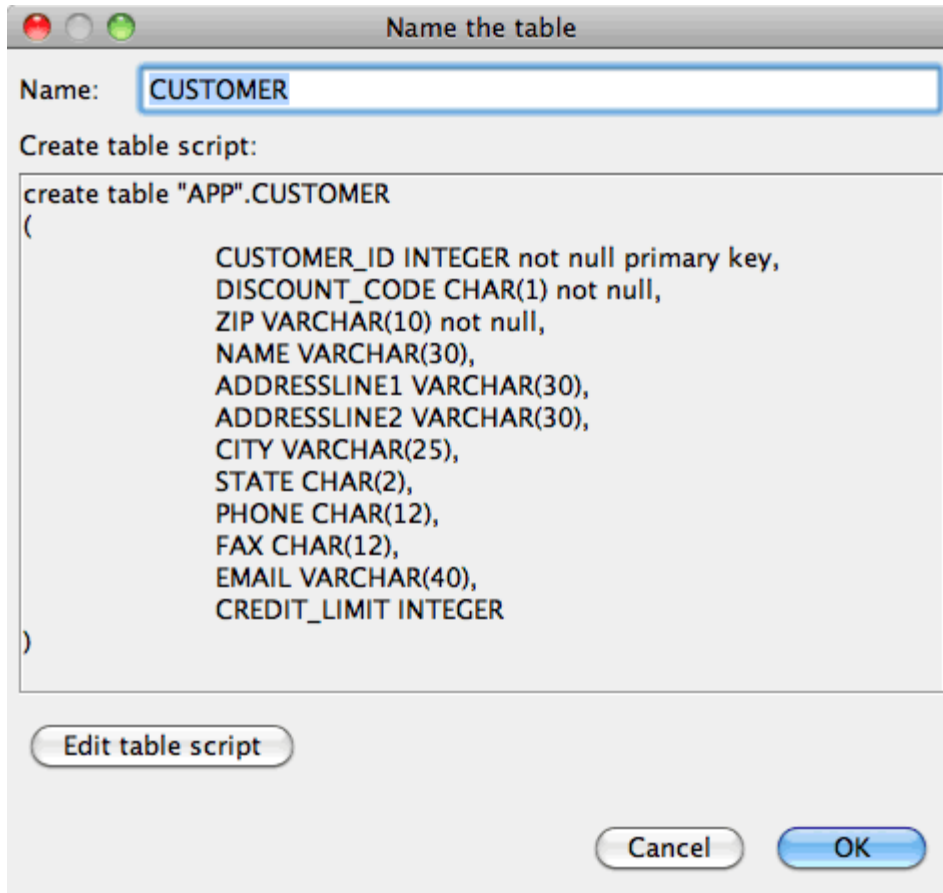
1. Connect to the `sample` database by right-clicking the connection node under the Databases node in the Services window and choosing Connect (username and password is `app`).
2. Expand the Tables node under the `sample` database connection, right-click the `CUSTOMER` table node and choose Grab Structure.



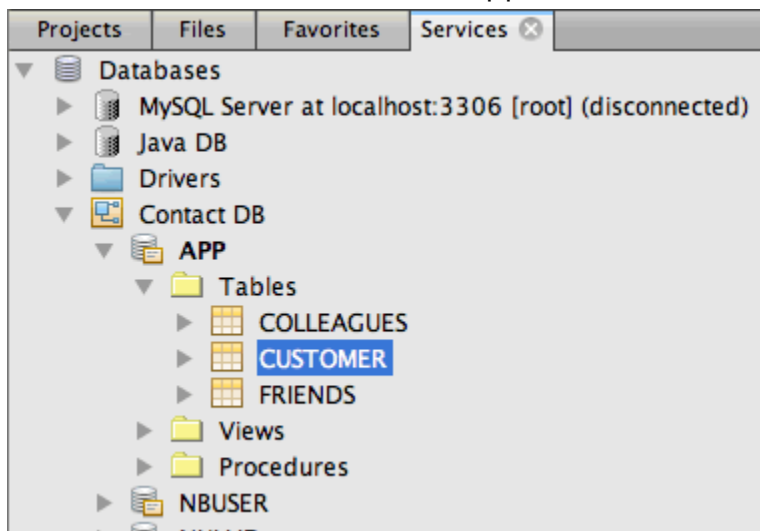
1. In the Grab Table dialog that opens, specify a location on your computer to save the grab file that will be created. Click Save.

The grab file records the table definition of the selected table.

1. Expand the APP schema node under the contact DB database connection, right-click the Tables node and choose Recreate Table to open the Recreate Table dialog box.
2. In the Recreate Table dialog box, navigate to the location where you saved the CUSTOMER grab file and click Open to open the Name the Table dialog box.



1. At this point you can change the table name or edit the table definition. Otherwise, click OK to immediately create the table in the contact database. A new CUSTOMER table node appears beneath the contact DB connection node.



If you view the data in the new `CUSTOMER` table you will see that there are no records in the database, but that the structure of the table is identical to the table that you grabbed.

Conclusion

This concludes the Working with the Java DB (Derby) Database tutorial. This tutorial demonstrated how to set up a connection to the Java DB database in NetBeans IDE. It then demonstrated how to create, view, modify and delete tables in the IDE's Services window. It also showed how work with the SQL Editor to add data to tables, and use the IDE's functionality allowing you to recreate tables using definitions from other databases.

Courtesy: <https://netbeans.apache.org/kb/docs/ide/java-db.html>

Modified: 2021.10.04.7.22.AM

Dököll Solutions, Inc.