

Offline Tutorial 3: A Brief Introduction to DBMS/PostgreSQL

Kevin Wang, kevinw@cse.ust.hk

August 2017

Why are we even reading this.

Throughout the course COMP3111 we are working on a chatbot (chatting-robot) on the instant messenger software LINE. One of the easiest way to do is using the example provided by LINE developer site and it is written in Java Enterprise Edition (J2EE). In order to complete the project you will need to pick up a set of skills, which is already *minimized* when we design the project, as listed below:

1. git, *a version controlling repository tools.*
2. Java, *an object oriented programming language.*
3. Database, *a program that manage your data.*
4. Spring, *a framework that provide web service in Java Enterprise Edition.*
5. Gradle, *a package management tools, like makefile, Ant, Maven.*
6. JUnit, *a testing suite for Java.*

As you can see we said “minimized” but there are still a lot to cover. It is impossible to cover all these contents in the lecture or tutorial. Besides, you should also develop the skill to pick up new contents therefore we are here. You are supposed to read this document at home and you will be given chance to practise them at lab. In any case you have encountered difficulty in reading this document, you can post your question on the forum or contact the TA for help.

Contents

1	Background	3
2	Essential Concept	3
3	Creating a User and a Database with pgAdmin	3
4	Data Manipulation	9
4.1	Creating Table	9
4.2	Dropping the table	10
4.3	Creating Data	10
4.4	Reading Data	11
4.5	Updating Data	11
4.6	Deleting Data	12
4.7	Joining Table	12
5	Working with Java	12
6	Self-Test	15

1 Background

We assume you have no background about database and what we are introducing here is just enough for our project purpose. We will not go through the design of database (e.g. E-R diagram, normal form, etc) and some examples illustrated here may not be the best considered as a good practise. Please enroll COMP3111 to learn database properly.

In this document we will use Windows 10 / PostgreSQL 9.6.3 / Java JDK 8 as a reference. You shall install PostgreSQL and JDK 8 on your machine. You don't need the PostgreSQL server for the project as ultimately your database server will be hosted on cloud. You need the database server on your computer for learning SQL and perhaps completing some lab works.

We will first illustrate how to perform data manipulation using command line mode. At the end of the document we will show how to do it with Java.

Learn More:

- [Course website of COMP3111](#)
- [Official Website of PostgreSQL](#)
- [TutorialPoints: PostgreSQL](#)
- [A list of books recommended by PostgreSQL website](#)

2 Essential Concept

A database can be viewed as a collection of data. A database management system (DBMS) is a general purpose software package that manages databases. Some well-known DBMS are Oracle SQL, Microsoft SQL Server, MySQL, PostgreSQL, MongoDB etc. In this chatbot project we are building a database that contains keywords-responses pair.

The structure of our database system can be explained as follows. We use PostgreSQL as our DBMS. In this DBMS, we have one database, let's call this **chatbotDB**. Inside this database, we have some tables. To begin with, we use only one table, called **phonebooktable**. Apparently this table is modelling a phonebook. Inside this table we will have a few columns, they are: **id**, **name**, **phone**. Making an analogy in excel, **chatbotDB** is like an excel file that contains one sheet called **phonebookTable**. On this sheet three columns are defined and they are **id**, **name**, **phone**.

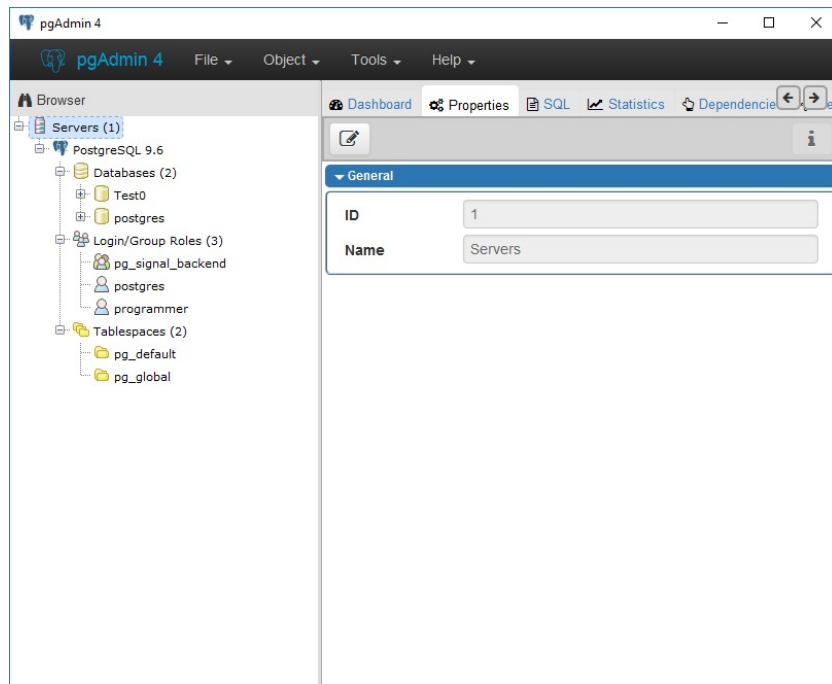
3 Creating a User and a Database with pgAdmin

Follow the instructions below to create a database.

1. Start pgAdmin 4 which has an icon looks like an elephant. You should be prompted with a login panel. Enter the password you have chosen when you install the pro-

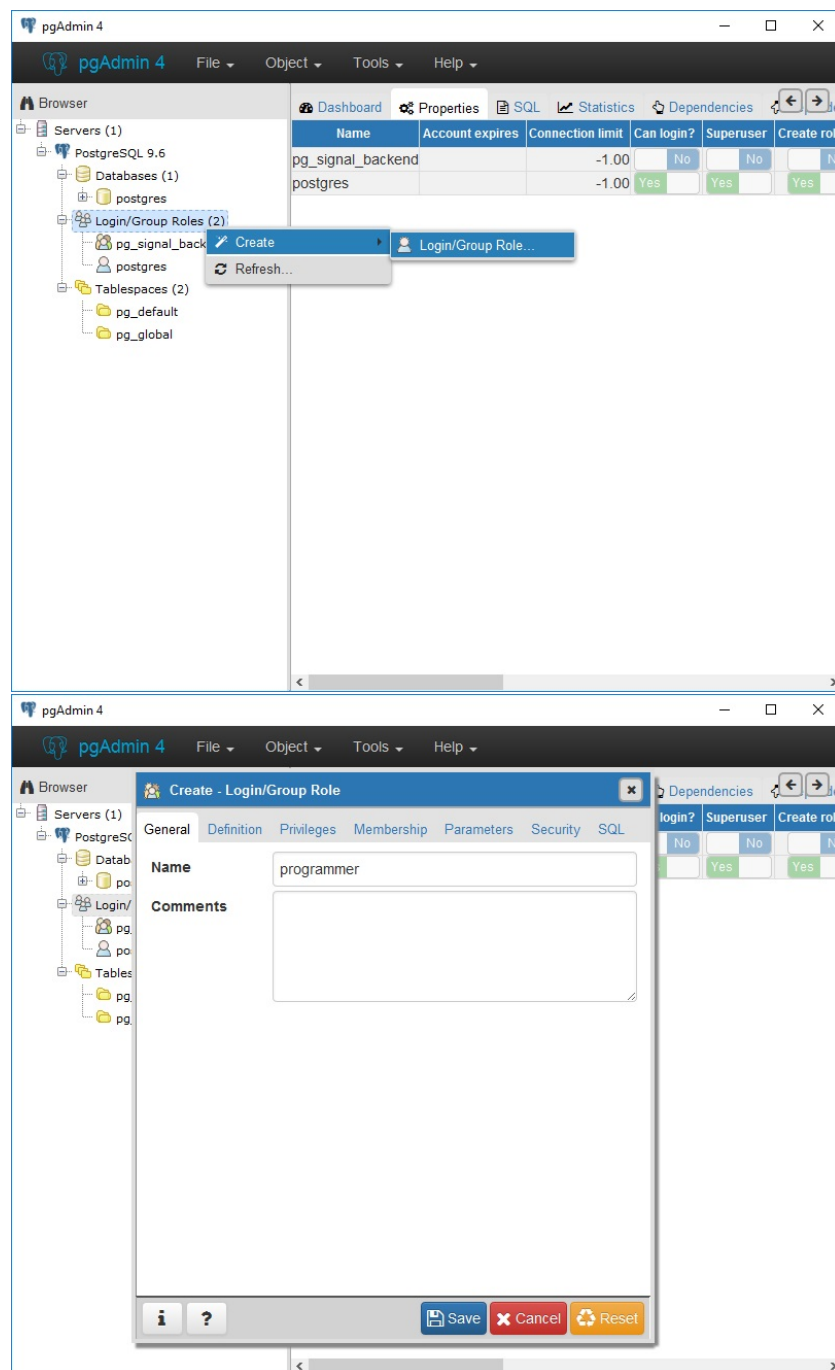
gram. If you fail this step, please create the user and the database with command line directly.

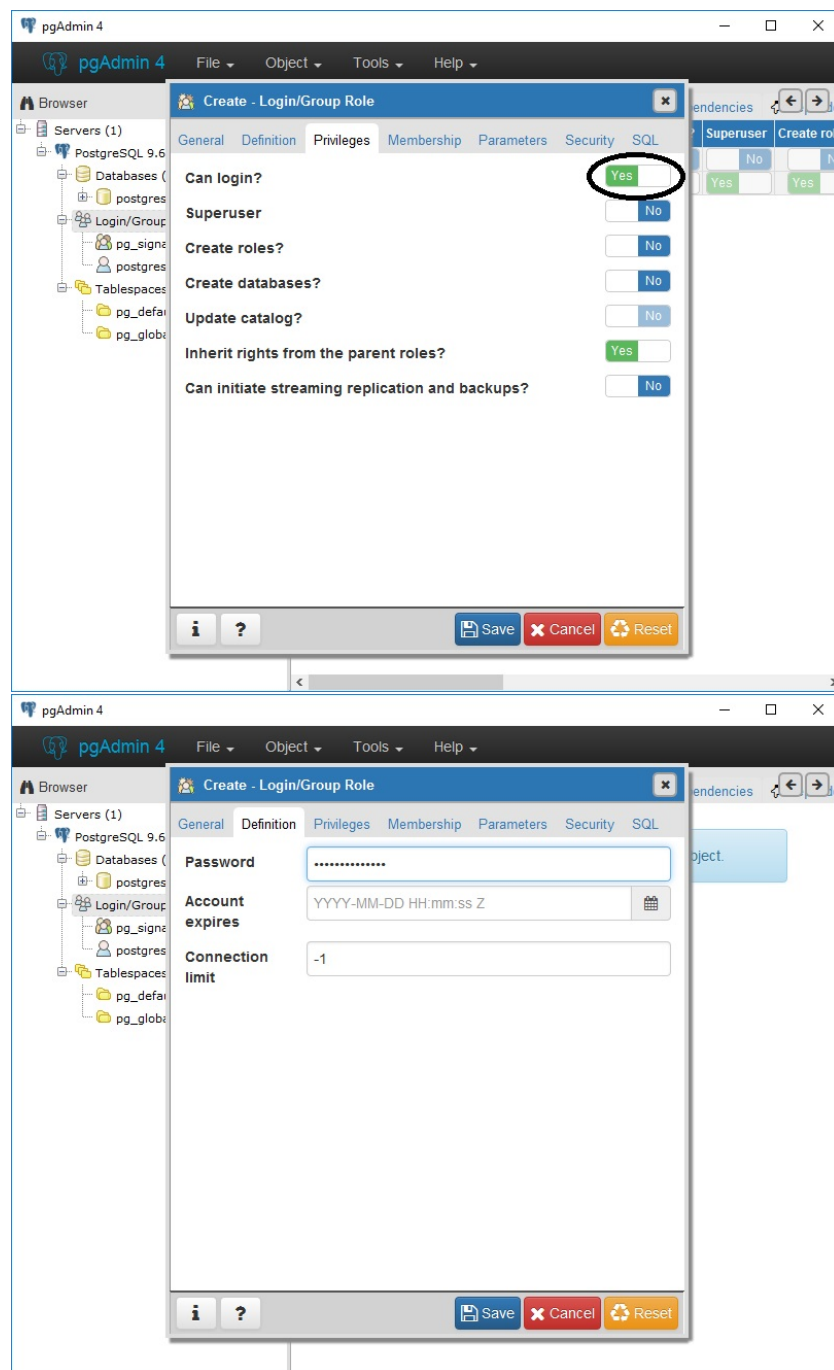
2. You shall see this



3. Now we are creating a username **programmer**

```
username: programmer
password: iamaprogrammer
privilege: Login
```

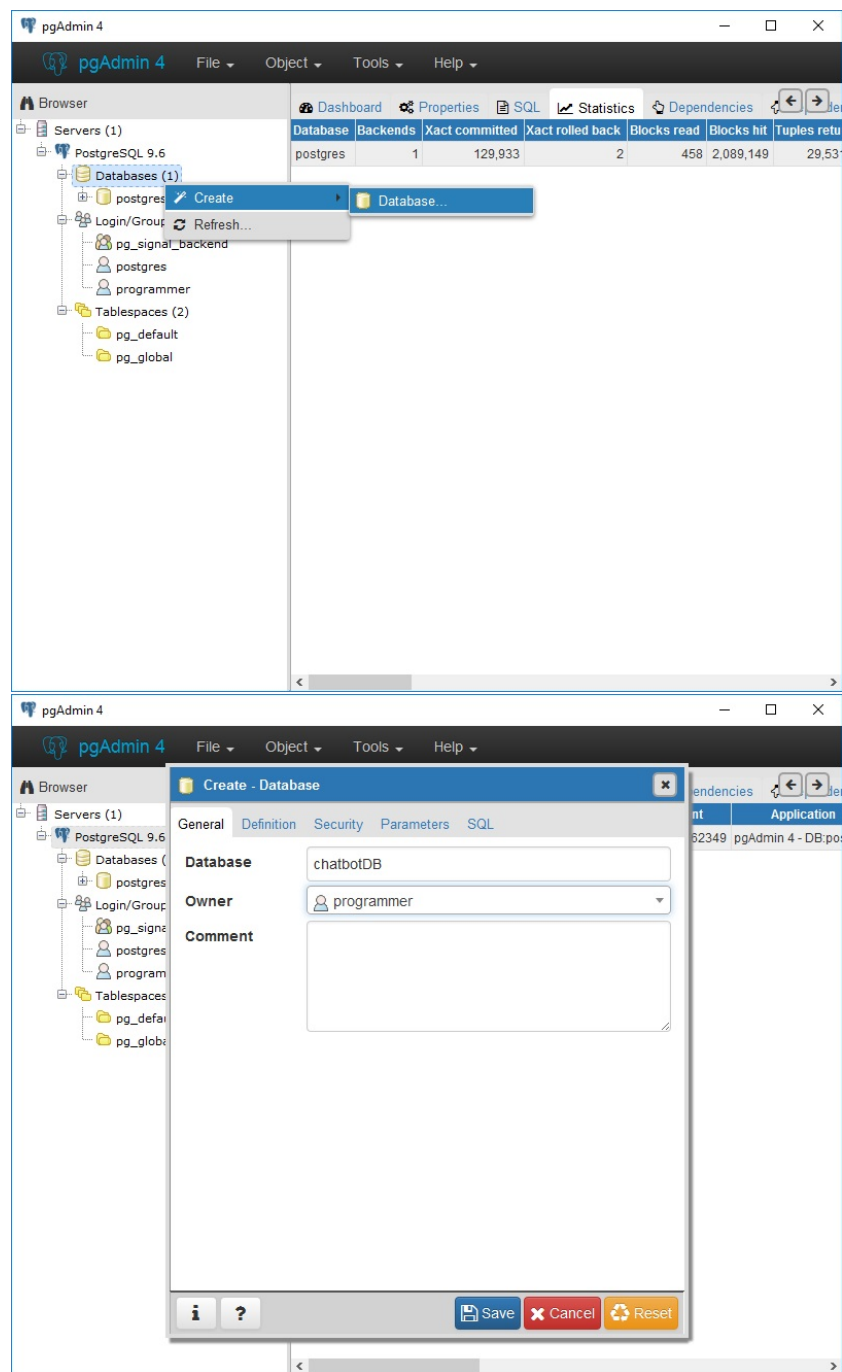




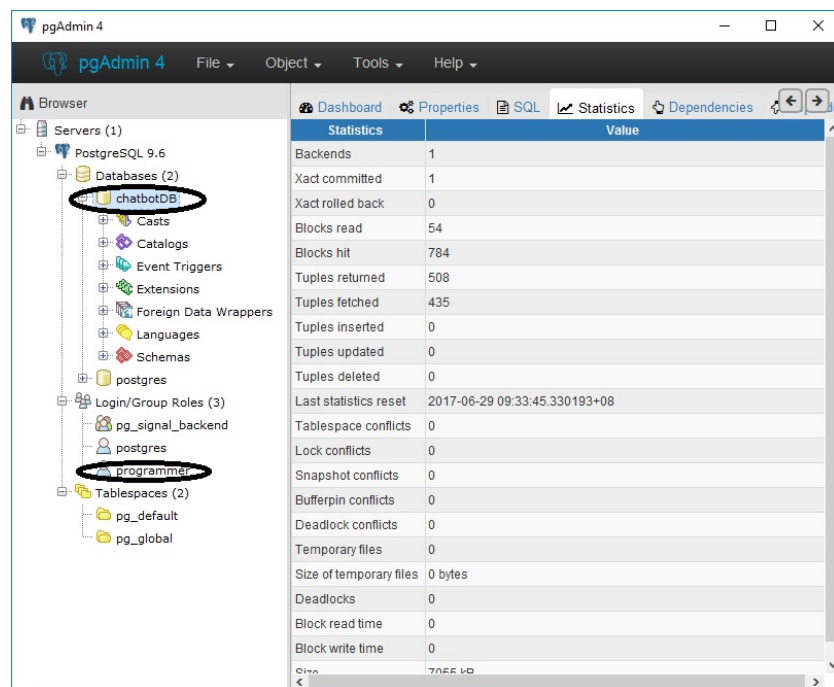
4. Next we are creating a database called `chatbotDB` and set `programmer` as the owner of the database.

Database name: `chatbotDB`

Owner: `programmer`



5. You should see this if you complete the process above. You are free to terminate pgAdmin.



Alternatively, you can create the user and the database using the SQL commands below. SQL commands are a set of common language for many different DBMS. Don't forget the terminating semicolon when you type it. Please refer to the next section about command line login server. Before you have created the user with login privilege, you should login using the admin account.

Create User

```
CREATE USER programmer WITH
    LOGIN
    NOSUPERUSER
    NOCREATEDB
    NOCREATEROLE
    INHERIT
    NOREPLICATION
    CONNECTION LIMIT -1
    PASSWORD 'iamaprogrammer';
```

Create Database

```
CREATE DATABASE "chatbotDB"
    WITH
    OWNER = programmer
    ENCODING = 'UTF8'
    CONNECTION LIMIT = -1;
```


Learn More:

- [Tutorial from pgadmin.org](http://pgadmin.org)

4 Data Manipulation

4.1 Creating Table

Next you need to login the system using command line. Press `Win + R` to launch the run dialog, and type `cmd` to launch the command prompt, a black screen allows you to type some command.

You will need to login to the database. Type the following in your command prompt.

```
psql -d chatbotDB -U programmer
```

`psql` is the command line client program that connect to PostgreSQL server. `-d chatbotDB` and `-U programmer` are two parameters which specifies the database and the user respectively.

You should see something like

```
Password for user programmer:
psql (9.6.3)
WARNING: Console code page (437) differs from Windows code page (1252)
         8-bit characters might not work correctly. See psql reference
         page "Notes for Windows users" for details.
Type "help" for help.
chatbotDB=>
```

There are several `psql` commands you can use. These are not SQL commands and only works if you are working with `psql`.

command	meaning
<code>\?</code>	display help
<code>\l</code>	List databases
<code>\d</code>	List all tables
<code>\d <TABLENAME></code>	Describe the table
<code>\e</code>	Edit your SQL query using external editor (notepad).
<code>\q</code>	Quit this program

We first create a table for the database `chatbotDB`. Type `\e` to open an editor and type the following things. You can also directly type the following in `psql`. Don't forget the terminating semicolon. When you close the editor the command will be executed.

```
CREATE TABLE phonebooktable (
```

```
    id int primary key,  
    name varchar(50),  
    phone varchar(8)  
);
```

You can verify your input by `\d` and `\d phonebooktable`. It is rather a bad idea to name your table or column with camel cases since some systems will automatically do casing for you while some other systems are actually case sensitive.

In this table we have three columns. `id` is an integer type while it is also the primary key of this table. A primary key means it is a unique column and no duplicate record can be insert with the same primary key. `name` and `phone` are other two fields. They are strings with variable size. Yet, `name` cannot have more than 50 characters while `phone` cannot have more than 8 characters. We are not storing phone as int for a rather straight forward reason – to avoid trimming the leading zeros.

Learn More:

- You can actually create a table using pgAdmin. See [this link](#).

4.2 Dropping the table

In case you have some typo and you want to do it again. You can either modify a table by `alter` or `drop` it. For simplicity, we show how to drop a table and repeat the step above you can recreate the table. Note, dropping a table will erase all data in this table.

To drop a table, simply say:

```
DROP TABLE phonebooktable;
```

4.3 Creating Data

To create (insert) data we simply use the following command

```
INSERT INTO TABLE_NAME (column1, column2, column3,...columnN)  
VALUES (value1, value2, value3,...valueN);
```

If you are sure about your data order you can also skip the column name

```
INSERT INTO TABLE_NAME VALUES (value1,value2,value3,...valueN);
```

Try the following by typing

```
INSERT INTO phonebooktable VALUES  
(1, 'Kevin Wang', '23588834');
```

Note that we enclose a string by a single quote character. If you want to use the single quote character inside your string, for instance, Kevin's daughter, you can try

```
INSERT INTO phonebooktable VALUES
(2, 'Kevin' 's Daughter', 999');
```

Since `id` is the primary key, if you attempt to insert another record with the same `id` the database will reject this command.

4.4 Reading Data

We use the `select` query to retrieve data from a table. Say we select all data from `phonebooktable`:

```
SELECT * FROM phonebooktable;
```

Asterisk (*) refers to everything from the `phonebooktable`.

You can also select a subset of the columns like

```
SELECT name, phone as number FROM phonebooktable;
```

Here we select two column from the table and we also rename the column `phone` as `number`.

SQL is very powerful in filtering data. Please refer to the references below for more comprehensive use of `select` query. You can try the code below that perform a like match so that if the name contain the substring `Daugh` will be selected.

```
SELECT * FROM phonebooktable
WHERE name LIKE '%Daugh%';
```

Learn More:

- [Beginner SQL Tutorial](#)

4.5 Updating Data

We use the `update` query to modify data in a table. Unless you are very sure all rows need to be updated in the same way, it is very likely we will use a `where` clause in this query

```
UPDATE table_name
SET column_name1 = value1,
    column_name2 = value2, ...
[WHERE condition]
```

As an example, we want to update Kevin's phone number as 12345678

```
UPDATE phonebooktable
SET phone = '12345678'
where name = 'Kevin Wang';
```

4.6 Deleting Data

We use the `delete` query to delete data from a table. We don't want to delete all record so it is a must to use delete with a where clause.

```
DELETE FROM phonebooktable WHERE id=2;
```

This will delete the entry with id=2.

4.7 Joining Table

Suppose you have another table `calllogtable` that records how many times a phone-number is dialed. This table has the fields `phone` (primary key), `record`.

The follow query allows us to output a list with name, phone number, and the record of the number being dialed.

```
SELECT phonebooktable.name, phonebooktable.phone,
       calllogtable.phone, calllogtable.record
FROM phonebooktable, calllogtable
WHERE phonebooktable.phone = calllogtable.phone
```

This query is a little redundant but easy to understand. Four columns are selected from the two tables. With the `WHERE` clause all record retrieved will have equal `phonebooktable.phone` and `calllogtable.phone`.

The follow query will produce a list of call with name, phone number which have 5 times or more dial record.

```
SELECT p.name as name, p.phone as phone, c.record as record
FROM phonebooktable as p, calllogtable as c
WHERE p.phone = c.phone
AND    c.record >= 5
```

In this query we use `p`, `c` as the alias of `phonebooktable` and `calllogtable` respectively to make the query shorter.

Learn More:

- [W3Schools – SQL Tutorial](#)

5 Working with Java

Before going into this section, make sure you have some basic background about Java – at least finish reading Tutorial 1. We are going to create a desktop application connecting to the database. You need a JDK 1.8 and download a PostgreSQL JDBC driver from their [official website](#). (You can select “PostgreSQL JDBC 4.2 Driver, 42.1.1” if you have the same configuration as I do.)

The big picture is as follows:

1. We will create and initialize an object called `Connection` that connects to the database.
2. Then we **prepare** a SQL statement while leaving some parameters.
3. Then we set the parameters to the `PreparedStatement`.
4. The prepare statement will be executed and a `ResultSet` will be returned.
5. We iterate on the `ResultSet` and get the values we want.
6. Finally we close the `ResultSet`, `PreparedStatement`, and the `Connection`.

Assume we create a POJO¹ called `DatabaseApplication`. We include the following libraries so that we can deal with the database.

```
import java.sql.*;
import java.net.URISyntaxException;
```

Then we prepare the connection using the following code where the database is running in the `localhost` at port `5432` connecting to the database `chatbotDB`.

```
String username = "programmer";
String password = "iamaprogrammer";
String dbUrl = "jdbc:postgresql://localhost:5432/chatbotDB";
Connection connection = DriverManager.getConnection(dbUrl, username,
    ↪ password);
```

Then we create a `PreparedStatement` by the following code. This statement tries to find all entries that `name` contains the substring of an input parameter.²

```
PreparedStatement stmt = connection.prepareStatement(
    "SELECT id, name, phone FROM phonebooktable where name like concat('%', ?, '%')");
```

Assume the program takes an input from user and store the name to be search in the variable `inputName`. We complete the `PreparedStatement` by

¹POJO: means a plain old Java object, a regular Java object, don't worry.

²A legacy way in writing this statement is by string concatenation, like `statement = "SELECT * FROM phonebooktable where name like %" + parameter + "%"`. However, this is insecure against SQL injection and should be abandoned.

```
stmt.setString(1, inputName);
↪ //assume inputName contain the name to search
```

We execute the statement and obtain the result by

```
ResultSet rs = stmt.executeQuery();
while (rs.next()) {
    System.out.println("ID: " + rs.getInt(1) + "\tName: " +
        ↪ rs.getString(2) + "\tPhone: " + rs.getString(3));
}
```

Then we need to close the ResultSet, PreparedStatement, and the Connection by

```
rs.close();
stmt.close();
connection.close();
```

Yet, most of the above function call will throw exception and we need to handle that as well. So a complete working code is

```
import java.sql.*;
import java.net.URISyntaxException;

public class DatabaseApplication {
    public static void main(String argv[]) {
        try {
            String username = "programmer";
            String password = "iamaprogrammer";
            String dbUrl = "jdbc:postgresql://localhost:5432/chatbotDB";
            Connection connection = DriverManager.getConnection(dbUrl,
                ↪ username, password);
            PreparedStatement stmt = connection.prepareStatement(
                "SELECT id, name, phone FROM phonebooktable where name like concat('%', ?, '%')");
            stmt.setString(1, "vin"); //or some other variables
            ResultSet rs = stmt.executeQuery();
            while (rs.next()) {
                System.out.println("ID: " + rs.getInt(1) + "\tName: " +
                    ↪ rs.getString(2) + "\tPhone: " + rs.getString(3));
            }
            rs.close();
            stmt.close();
            connection.close();
        } catch (Exception e) {
            System.out.println(e);
        }
    }
}
```

When you run the program you need a postgresQL jdbc driver. Assume the driver you have download is called `postgresql-42.1.1.jar`, execute your program with the following line:

```
java -cp postgresql-42.1.1.jar;. DatabaseApplication
```

Run this line if you are running on Linux or Mac:

```
java -cp postgresql-42.1.1.jar:. DatabaseApplication
```

6 Self-Test

Try to perform the following tasks to get familiar with SQL.

1. Create a database and a user using pgAdmin.
2. Use psql, the command line tool, to create two tables in the database. The first table is named “students” with the columns ID (primary key) and Name which both are strings. The second table is named “computer” with two columns called Model and Price which are a string and a decimal number respectively.
3. Use psql to insert two rows to each table into the database. Try and see if you are able to insert duplicate record in each tables.
4. Repeat above by using a Java program. Try to print the error message when you fail to insert the record.