实验 JDBC 与 PL/pgSQL

一、实验目的

1. 掌握使用 Java 连接数据库的基本步骤，并对数据库进行各种数据操作。
2. 掌握 PL/pgSQL 函数的结构，并掌握 PL/pgSQL 函数的编写和调用。

二、实验环境

1. 硬件：处理器 Intel® Core™ i5-8300H CPU @2.30GHz 内存 8.00GB。
2. 操作系统：Windows 10 家庭中文版，64 位操作系统。
3. 数据库管理系统：PostgreSQL-pgAdmin 4。

三、实验内容

1. 理解 JDBC 连接 PostgreSQL 的基本步骤，使用 Java 语言编程实现对数据库的访 问，并对数据库进行各种操作。
2. 编写 PL/pgSQL 函数，输入参数，定义变量，输出相应的结果。
3. JDBC 和 PL/pgSQL 函数综合练习。

四、实验数据

应急预案指面对突发事件如自然灾害、重特大事故、环境公害及人为破坏的应急管理、指挥、救援计划等，是一种公文。通常一个应急预案由多个不同的编制单位协同编写，才能编制完成。应急预案包含预案编号（plan_id）,预案名（plan_name），针对的灾害类型（plan_disatype），针对的区域（plan_area），针对的灾害等级（plan_level），发布时间（plan_date）。应急预案编制的参与单位包含单位编号（depart_id），单位名称 (depart_name)，单位联系方式（depart_tel）。一个参与单位可能参与多个预案的编制， 一个预案需要多个参与单位协作完成。当参与单位完成编写应急预案时，会记录该单位 在应急预案编制中的职责（depart_respon）和工作量（workload）

五、实验作业

1. **使用 JDBC 编程，连接 Emgyplan 数据库，新建数据表 record2，record2 结构如下：**

属性	类型	是否为主键	是否外键
depart _id	int	是	否
plan_id	int	是	否
workload	int	否	否

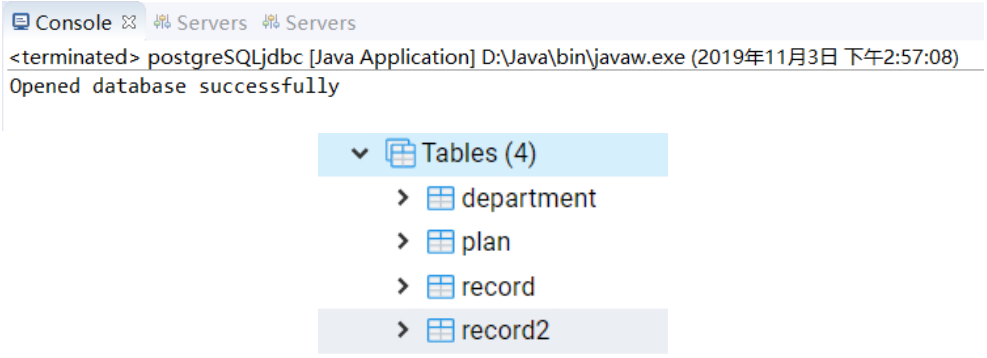
步骤一.配置好环境后，进入 eclipse 编写代码如下。

SQL 语句为：create table record2(depart_id int not null,plan_id int not null,workload int not null,primary key(depart_id,plan_id));

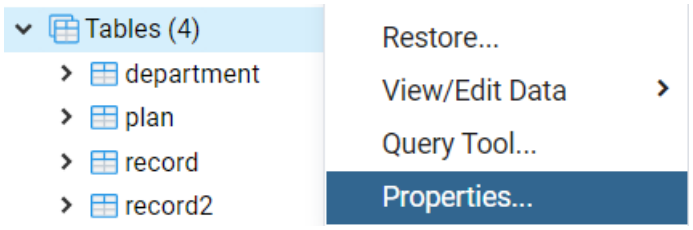
postgreSQLjdbc.java

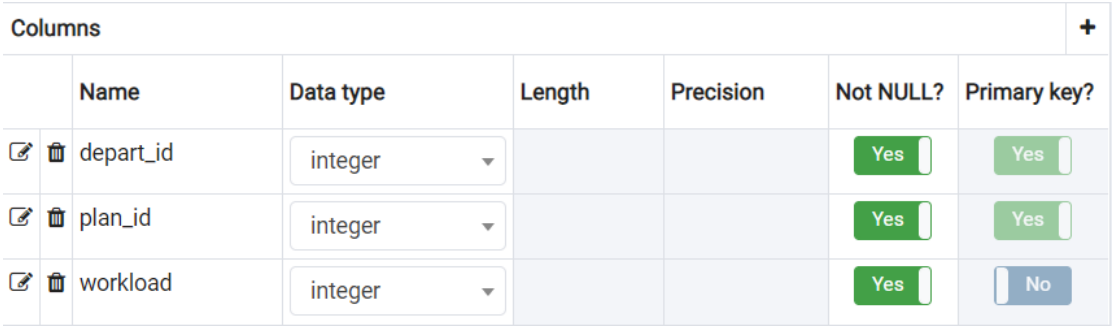
```java
import java.sql.Connection;
public class postgreSQLjdbc {

    public static void main(String[] args) {
        Connection c = null;
        try {
            Class.forName("org.postgresql.Driver");
            c = DriverManager .getConnection("jdbc:postgresql://localhost:5432/Emgyplan", "postgres", "123456" );
            Statement stmt = c.createStatement(); //使用连接对象创建处理对象
            //执行SQL语句
            stmt = c.createStatement();
            String sql = "CREATE TABLE record2("+ //将要执行的SQL语句写成字符串形式
                    "depart_id INT NOT NULL,"+
                    "plan_id INT NOT NULL," +
                    "workload INT NOT NULL,"+
                    "PRIMARY KEY(depart_id,plan_id));";
            stmt.executeUpdate(sql);
            stmt.close(); //关闭相关对象
            c.close();
        } catch (Exception e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
            System.err.println(e.getClass().getName()+": "+e.getMessage());
            System.exit(0);
        }
        System.out.println("Opened database successfully");
    }
}
```

步骤二.执行该代码，输出如下。在 pgAdmin4 上检查表创建成功。



步骤三.查看表的属性。右击 record2，选择 properties 查看表结构。在新窗口 columns 一栏查看表的主键。





2. 使用 JDBC 编程，向 record2 中插入如下数据。

depart _id	plan_id	workload
1	2	3
2	5	2

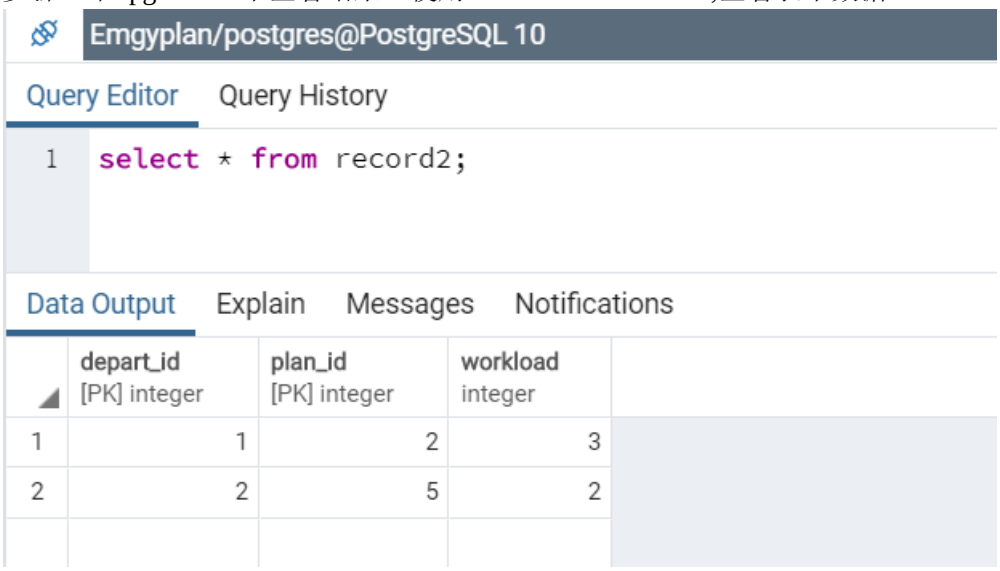
步骤一. 编写 java 代码如下。

SQL 语句为：insert into record2(depart_id,plan_id,workload) values(1,2,3),(2,5,2);

```java
package test;
import java.sql.Connection;
public class postgreSQLjdbc {

    public static void main(String[] args) {
        Connection c = null;
        try {
            Class.forName("org.postgresql.Driver");
            c = DriverManager .getConnection("jdbc:postgresql://localhost:5432/Emgyplan", "postgres", "123456" );
            Statement stmt = c.createStatement(); //使用连接对象创建处理对象
            //执行SQL语句
            stmt = c.createStatement();
            String sql = "insert into record2 (depart_id,plan_id,workload) values"+ //将要执行的SQL语句写成字符串形式
                    "(1,2,3),"+
                    "(2,5,2);";
            stmt.executeUpdate(sql);
            stmt.close(); //关闭相关对象
            c.close();
        } catch (Exception e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
            System.err.println(e.getClass().getName()+": "+e.getMessage());
            System.exit(0);
        }
        System.out.println("Opened database successfully");
    }
}
```

步骤二.执行该代码。输出如下。

步骤三.在 pgAdmin 中查看结果。使用 select * from record2;查看表中数据。



插入成功！

3. **在 PG 中，创建一个函数 add_workload（depart_id, addworkload），实现对给定单位 的工作量增给定值。例如：add_workload（2,5）是给 2 号部门的工作量添加 5。**

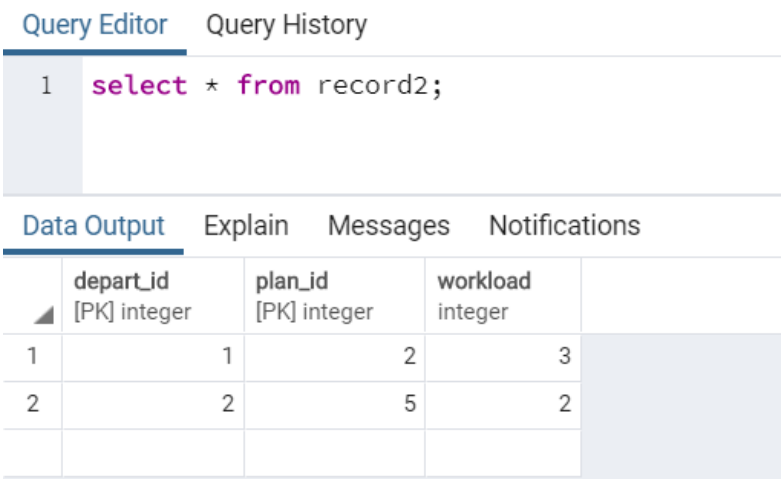
步骤一.在 pgAdmin 进入 Tools-Query Tool，编写函数如下。

SQL 语句为：

```
create or replace function add_workload(int,int)
returns int
as
$$
begin
    update record2 set workload=workload+$2 where depart_id = $1;
    return 0;
end;
$$
language plpgsql;
```

步骤二.点击 F5 运行，完成函数的定义。

步骤三.查看表 record2 内当前数据。



步骤四.调用函数 add_workload(2,5)，查看表内数据的变化。

Query Editor　Query History

```
1  select add_workload(2,5);
2  select * from record2;
```

Data Output　Explain　Messages　Notifications

	depart_id [PK] integer	plan_id [PK] integer	workload integer
1	1	2	3
2	2	5	7

depart_id 为 2 的部门工作量增加了 5，操作成功！

4. 使用 JDBC 编程，删除 record2 数据表的所有记录。

步骤一.编写 java 代码如下。

postgreSQLjdbc.java

```
1  package test;
2  import java.sql.Connection;
5  public class postgreSQLjdbc {
6
7      public static void main(String[] args) {
8          Connection c = null;
9          try {
10             Class.forName("org.postgresql.Driver");
11             c = DriverManager .getConnection("jdbc:postgresql://localhost:5432/Emgyplan", "postgres", "123456" );
12             Statement stmt = c.createStatement(); //使用连接对象创建处理对象
13             //执行SQL语句
14             stmt = c.createStatement();
15             String sql = "delete from record2"; //将要执行的SQL语句写成字符串形式
16             stmt.executeUpdate(sql);
17             stmt.close(); //关闭相关对象
18             c.close();
19         } catch (Exception e) {
20             // TODO Auto-generated catch block
21             e.printStackTrace();
22             System.err.println(e.getClass().getName()+": "+e.getMessage());
23             System.exit(0);
24         }
25         System.out.println("Opened database successfully");
26     }
27 }
28
29
```

步骤二.运行该程序，得到输出如下。在 pgAdmin 中查看结果。

Console　Servers　Servers

```
<terminated> postgreSQLjdbc [Java Application] D:\Java\bin\javaw.exe (2019年11月3日 下午4:18:32)
Opened database successfully
```

Query Editor　Query History

```
1  select * from record2;
```

Data Output　Explain　Messages　Notifications

	depart_id [PK] integer	plan_id [PK] integer	workload integer

SQL 语句为： delete from record2;

5. **使用 JDBC 编程，在 record2 中随机插入 3000 条数据，使得主键 depart_id 的值为 1-3000，依次递增；plan_id 的随机取值范围为 1-500，workload 的随机取值范 围为 1-50，以上数据均为整数。（用 SQL 写插入数据的代码，然后在 JDBC 中执行 SQL。）**

我们通过 java 动态地产生随机数，因此需要使用预备语句在 SQL 语句中保留插入的数据值。

步骤一.编写 java 代码如下。

postgreSQLjdbc.java

```
1  package test;
2  import java.sql.Connection;
7  public class postgreSQLjdbc {
8
9      public static void main(String[] args) {
10         Connection c = null;
11         try {
12             Class.forName("org.postgresql.Driver");
13             c = DriverManager .getConnection("jdbc:postgresql://localhost:5432/Emgyplan", "postgres", "123456" );
14             Random r=new Random(1);
15             for(int i=1;i<=3000;i++) {
16                 String sql="insert into record2 (depart_id,plan_id,workload) values(?,?,?);";
17                 PreparedStatement pStmt=c.prepareStatement(sql);
18                 int tmpplan_id=r.nextInt(500)+1;
19                 int tmpworkload=r.nextInt(50)+1;
20                 pStmt.setInt(1, i);
21                 pStmt.setInt(2, tmpplan_id);
22                 pStmt.setInt(3, tmpworkload);
23                 pStmt.executeUpdate();
24             }
25             c.close();
26         } catch (Exception e) {
27             // TODO Auto-generated catch block
28             e.printStackTrace();
29             System.err.println(e.getClass().getName()+": "+e.getMessage());
30             System.exit(0);
31         }
32         System.out.println("Opened database successfully");
33     }
34 }
35
```

步骤二.执行该代码，在 pgAdmin 中查看数据添加结果。

Query Editor | Query History

```
1  select * from record2;
```

Data Output | Explain | Messages | Notifications

	depart_id [PK] integer	plan_id [PK] integer	workload integer
1	1	486	39
2	2	348	14
3	3	255	5
4	4	435	7
5	5	479	49
6	6	70	24
7	7	318	14
8	8	63	35
9	9	93	13
10	10	97	40
11	11	377	33
12	12	311	50
13	13	175	10
14	14	299	4
15	15	438	3
16	16	206	5

Query Editor | Query History

```
1  select count(*) from record2;
```

Data Output | Explain | Messages | Notifications

	count bigint
1	3000

插入数据的 SQL 语句为：
insert into record2 (depart_id,plan_id,workload) values (?,?,?);其中?标识预备语句中暂未被替换的值，在执行前会将其替换为具体的值。

查询的 SQL 语句为：
select * from record2;　　以及　　select count(*) from record2;

6. **使用 JDBC 编程，不使用嵌套查询：预案参与单位的平均工作量最大的预案 id 和 平均工作量。(编写函数，函数中使用除嵌套查询外的其它 SQL 查询；然后通过 JDBC 执行调用该函数的语句。比如可以使用 SQL 查询相关数据，然后用 Java 求 平均工作量最大的值)**

考虑完成整个查询需要两步操作：1.首先获取每个 plan_id 的平均工作量；2.获取工作量中的最大值，并输出其 plan_id。由于不允许使用嵌套查询而要求使用函数，因此很自然地想到用函数完成其中一步查询。将结果集返回，通过 java 语句处理该结果集得到答案。我们编写函数完成对第一步的查询。
SQL 语句如下：
select plan_id,avg(workload) from record2 group by plan_id;
由于我们使用函数，整体的 SQL 语句如下：

```
create or replace function showavg()
returns table(planid int,avg_workload numeric)
as
$$
declare id int;
begin
    return query(select plan_id,avg(workload) from record2 group by plan_id);
end;
$$
language plpgsql;
```

步骤一.在 pgAdmin 中定义函数如下（附函数运行结果）。

Query Editor　Query History

```
1  create or replace function showavg()
2  returns table(planid int,avg_workload numeric)
3  as
4  $$
5  declare id int;
6  begin
7      return query(select plan_id,avg(workload) from record2 group by plan_id);
8  end;
9  $$
10 language plpgsql;
11 select showavg();
```

Data Output　Explain　Messages　Notifications

	showavg record
1	(87,23.6666666666666667)
2	(184,20.1666666666666667)
3	(477,22.2500000000000000)
4	(273,16.2500000000000000)

步骤二.编写 java 代码如下。运行改代码，结果如下。

postgreSQLjdbc.java

```
1  package test;
2  import java.sql.Array;
9  public class postgreSQLjdbc {
10
11     public static void main(String[] args) {
12         Connection c = null;
13         try {
14             Class.forName("org.postgresql.Driver");
15             c = DriverManager .getConnection("jdbc:postgresql://localhost:5432/Emgyplan", "postgres", "123456" );
16             Statement stmt = c.createStatement(); //使用连接对象创建处理对象
17             //执行SQL语句
18             stmt = c.createStatement();
19             String sql = "select * from showavg()"; //将要执行的SQL语句写成字符串形式
20             ResultSet rst=stmt.executeQuery(sql);
21             ArrayList<Integer> idlist=new ArrayList<Integer>();
22             ArrayList<Float> wllist=new ArrayList<Float>();
23             while(rst.next()) {
24                 int id=rst.getInt("planid");
25                 idlist.add(id);
26                 float avg_workload=rst.getFloat("avg_workload");
27                 wllist.add(avg_workload);
28             }
29             stmt.close(); //关闭相关对象
30             float max_workload=Collections.max(wllist);
31             for(int i=0;i<idlist.size();i++) {
32                 if(wllist.get(i)==max_workload)
33                     System.out.println("plan_id="+idlist.get(i)+",max_workload="+max_workload+"\n");
34             }
35             c.close();
36         } catch (Exception e) {
37             // TODO Auto-generated catch block
38             e.printStackTrace();
```

Console Servers Servers

```
<terminated> postgreSQLjdbc [Java Application] D:\Java\bin\javaw.exe (2019年11月3日 下午8:11:36)
plan_id=84,max_workload=50.0

plan_id=279,max_workload=50.0

Opened database successfully
```

可编辑格式的 SQL 语句为：

```
create or replace function showavg()
returns table(planid int,avg_workload numeric)
as
$$
declare id int;
begin
    return query(select plan_id,avg(workload) from record2 group by plan_id);
end;
$$
language plpgsql;
```

7. 使用 JDBC 编程，使用函数、嵌套查询：预案参与单位的平均工作量最大的预案 id 和平均工作量。（编写函数，函数中可以使用 SQL 嵌套查询；然后通过 JDBC 调用 执行调用该函数的语句）

由于允许使用嵌套查询，我们可以直接在数据库层面完成所有的筛选工作。
通过定义函数，直接返回平均工作量最大的预案 id 及其平均工作量。
在 java 代码中，只需对结果进行打印输出。

步骤一.在 pgAdmin 中编写函数如下（包括执行结果）。

Query Editor | Query History

```
1  create or replace function showmaxavg()
2  returns table(planid int,max_workload numeric)
3  as
4  $$
5  begin
6      return query(with showavg(planid,avg_workload) as
7                   (select plan_id,avg(workload) from record2 group by plan_id)
8                   select * from showavg
9                  where avg_workload=(select max(avg_workload) from showavg));
10 end;
11 $$
12 language plpgsql;
13 select * from showmaxavg();
14
```

Data Output | Explain | Messages | Notifications

	planid integer	max_workload numeric
1	84	50.0000000000000000
2	279	50.0000000000000000

步骤二.编写 java 代码如下。执行该代码。

postgreSQLjdbc.java

```
1  package test;
2  import java.sql.Array;
9  public class postgreSQLjdbc {
10
11     public static void main(String[] args) {
12         Connection c = null;
13         try {
14             Class.forName("org.postgresql.Driver");
15             c = DriverManager .getConnection("jdbc:postgresql://localhost:5432/Emgyplan", "postgres", "123456" );
16             Statement stmt = c.createStatement(); //使用连接对象创建处理对象
17             //执行SQL语句
18             stmt = c.createStatement();
19             String sql = "select * from showmaxavg()"; //将要执行的SQL语句写成字符串形式
20             ResultSet rst=stmt.executeQuery(sql);
21             ArrayList<Integer> idlist=new ArrayList<Integer>();
22             ArrayList<Float> wllist=new ArrayList<Float>();
23             while(rst.next()) {
24                 int id=rst.getInt("planid");
25                 float max_workload=rst.getFloat("max_workload");
26                 System.out.println("plan_id="+id+",max_workload="+max_workload+"\n");
27             }
28             stmt.close(); //关闭相关对象
29             c.close();
30         } catch (Exception e) {
31             // TODO Auto-generated catch block
32             e.printStackTrace();
33             System.err.println(e.getClass().getName()+": "+e.getMessage());
34             System.exit(0);
35         }
36         System.out.println("Opened database successfully");
37     }
38 }
```

Console | Servers | Servers

```
<terminated> postgreSQLjdbc [Java Application] D:\Java\bin\javaw.exe (2019年11月3日 下午8:34:25)
plan_id=84,max_workload=50.0

plan_id=279,max_workload=50.0

Opened database successfully
```

可编辑 SQL 语句为：

```
create or replace function showmaxavg()
returns table(planid int,max_workload numeric)
as
```

```
$$
begin
    return query(with showavg(planid,avg_workload) as
                 (select plan_id,avg(workload) from record2 group by plan_id)
                 select * from showavg
                where avg_workload=(select max(avg_workload) from showavg));
end;
$$
language plpgsql;
```

六、问题与思考

（1）使用 C 语言连接访问 PG 和使用 Java 连接访问的异同。

答：相同点：

1.交互的步骤相同。整体来看，C 语言连接访问 PG 与 java 连接访问，都需要先连接数据库，然后编写 SQL 语句，通过相关接口传递。

2.C 语言与 java 语言本身并不干预数据库的操作，C 语言和 java 语言尽向数据库传递需要的信息，获取信息后进行本地化的处理。无论是 java 还是 C，数据库层面的操作完全相同。

不同点：

C 提供编写 pgc 格式文件的方式嵌入 SQL 语句。而 Java 在代码中定义字符串来嵌入 SQL 语句。

（2）使用函数的优点有哪些？

答：优点包括但不限于：

1.将工作放置数据库上进行，客户端的维护较为简单。同时，方便对数据库集成操作进行管理。

2.提供更加便利的接口。客户端无需使用复杂的指令代码即可获取需要的数据信息。如第 6 题第 7 题对比。

七、实验体会

答：本次实验遇到的问题一如既往的多。

1.函数的运用上。函数的语法较为新颖，需要重新记忆理解。

2. 函数的返回值形式多样，对于不同的返回值，需要经过实践才能更好地处理数据。在第 6、7 题中，函数的返回值是表，表的每一行是 record（如下图 1）。如果在 java 中使用 SQL 语句只是调用函数：“select 函数名”，较难对得到的数据进行操作，甚至可能会数据丢失（如下图 2）。需要使用“select * from 函数名”得到具有列名的表。

Query Editor | Query History

```
1  create or replace function showavg()
2  returns table(planid int,avg_workload numeric)
3  as
4  $$
5  declare id int;
6  begin
7      return query(select plan_id,avg(workload) from record2 group by plan_id);
8  end;
9  $$
10 language plpgsql;
11 select showavg();
```

Data Output | Explain | Messages | Notifications

	showavg record
1	(87,23.6666666666666667)
2	(184,20.1666666666666667)
3	(477,22.2500000000000000)
4	(273,16.2500000000000000)
5	(394,22.4285714285714286)
6	(51,28.1250000000000000)
7	(272,20.1111111111111111)

图 1

postgreSQLjdbc.java

```
2  import java.sql.Array;
9  public class postgreSQLjdbc {
10
11     public static void main(String[] args) {
12         Connection c = null;
13         try {
14             Class.forName("org.postgresql.Driver");
15             c = DriverManager .getConnection("jdbc:postgresql://localhost:5432/Emgyplan", "postgres", "liujh123" );
16             Statement stmt = c.createStatement(); //使用连接对象创建处理对象
17             //执行SQL语句
18             stmt = c.createStatement();
19             String sql = "select showavg()"; //将要执行的SQL语句写成字符串形式
20             ResultSet rst=stmt.executeQuery(sql);
21             ArrayList<Integer> idlist=new ArrayList<Integer>();
22             ArrayList<Float> wllist=new ArrayList<Float>();
23             while(rst.next()) {
24                 Array record=rst.getArray(1);
25                 System.out.println(record);
26             }
27             stmt.close(); //关闭相关对象
28
29             c.close();
30         } catch (Exception e) {
31             // TODO Auto-generated catch block
32             e.printStackTrace();
33             System.err.println(e.getClass().getName()+": "+e.getMessage());
34             System.exit(0);
35         }
36         System.out.println("Opened database successfully");
37     }
38 }
39
```

Console | Servers | Servers

```
<terminated> postgreSQLjdbc [Java Application] D:\Java\bin\javaw.exe (2019年11月3日 下午7:46:10)
-7,23.6666666666666667
-84,20.1666666666666667
-77,22.2500000000000000
-73,16.2500000000000000
-94,22.4285714285714286
-1,28.1250000000000000
```

图 2