Getting Started with PostgreSQL

eryar@163.com

Abstract. PostgreSQL is an excellent implementation of relational database, fully featured, open source, and free to use. Nearly nontrivial computer applications manipulate large amounts of data, and a lot of applications are written primarily to deal with data rather than perform calculations. Some writers estimate that 80% of all application development in the world today is connected in some way to complex data stored in a database, so databases are very important foundation to many applications. This article mainly about the usage of SQL shell of PostgreSQL(psql).

Key Words. Database, PostgreSQL, psql

1. Introduction

PostgreSQL 是一款开源的关系－对象数据库，其授权方式为 BSD 形式的开源协议，比 OpenCASCADE 的 LGPL 协议更为自由。将这些高质量的开源产品组合一下，应该可以创造出实用的软件，提高工作效率。

如下图所示为一款产于英国剑桥的工厂辅助设计管理系统 PDMS 的主要界面：

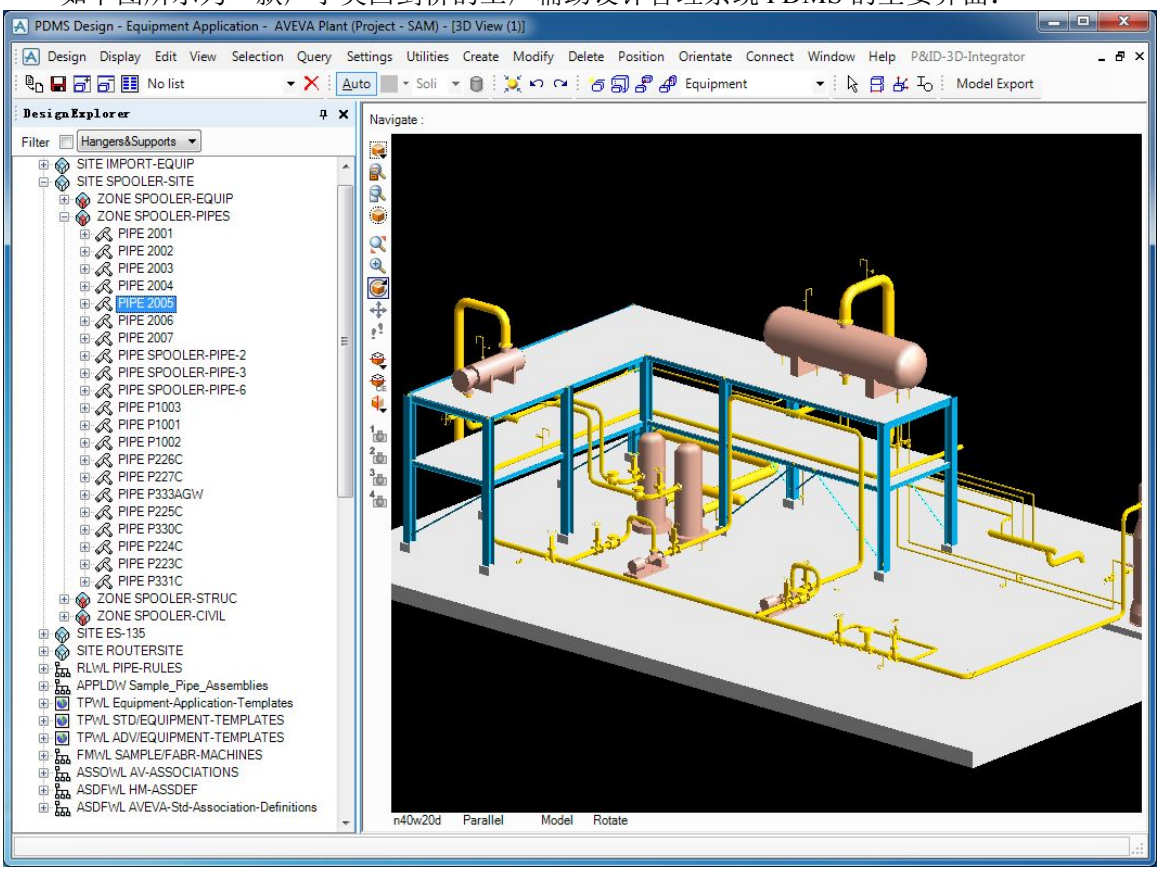


Figure 1.1 AVEVA Plant(PDMS) GUI

像 AVEVA Plant(PDMS)这样的产品，最终的结果都是以数据库的形式将模型及其他信息保存。因此，需要有数据库管理系统来对这些数据进行管理。不管是以树形的方式，还是以三维模型的方式，都是其统一数据模型的一种表现形式。基于 Observer 设计模式定义：

定义对象间的一对多的依赖关系，当一个对象的状态发生变化时，所有依赖于它的对象都得到通知，并自动更新。

由 Observer 定义可知，树形显示及三维显示都是数据模型的 Observer。不管是在树上修改还是在三维模型中以直观的交互方式修改模型，根本上还是修改了数据模型。并且在一个 Observer 中修改了数据模型，另一个 Observer 中会得到通知并自动更新和数据模型保持一致。其核心数据模型归根到底是由一个高性能的数据库来管理，由此可见数据库管理系统的重要性。

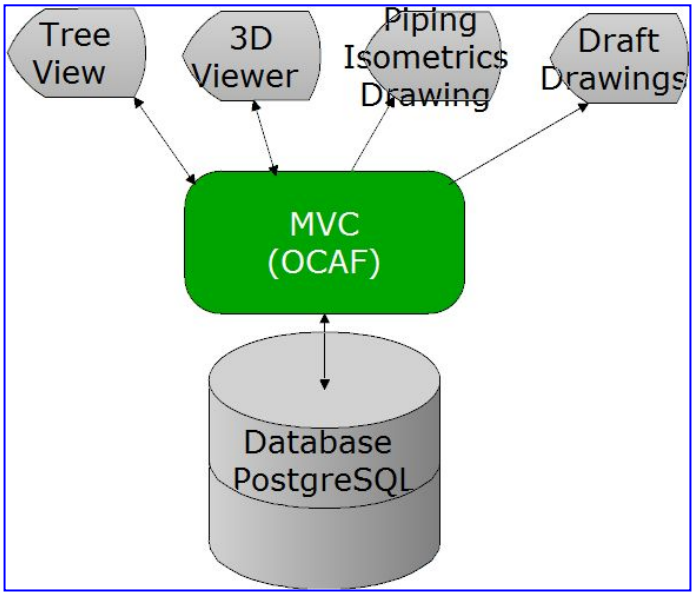


Figure 1.2 PDMS Architecture

我认为的 PDMS 软件架构如图 1.2 所示，树形视图、三维视图及管道 ISO 图和安装图等都是数据模型的观察者。因为对数据模型的存储非常重要，所以数据管理系统：数据库的需求就显而易见。但是对于应用开发而言，提供一个 MVC 框架来统一数据的增、删、改的接口更为重要。因为其好处更多：

- 接口统一，便于程序开发及给用户一致的操作，便于用户轻松掌握软件；
- 只有基于统一的接口，才能基于此提供 Undo/Redo 功能；
- 便于与 Tcl, Python 等脚本语言绑定，为程序提供二次开发功能；
- 基于脚本绑定，为程序提供自动化测试，有力保证软件质量；

综上所述可知，OpenCASCADE 提供的 OCAF 框架在思想上对于软件开发的重要性。不过本文主要记录如何在 Windows 平台上使用另一款高质量的开源数据库管理系统 PostgreSQL。理解 PostgreSQL，就解决了软件底层数据的管理，为软件开发的数据模型提供根本保障。

2. Creating User Records

在 Windows 系统中，打开命令窗口并将 PostgreSQL 程序所在目录置为当前路径，然后运行 createuser.exe 程序，如下图所示：

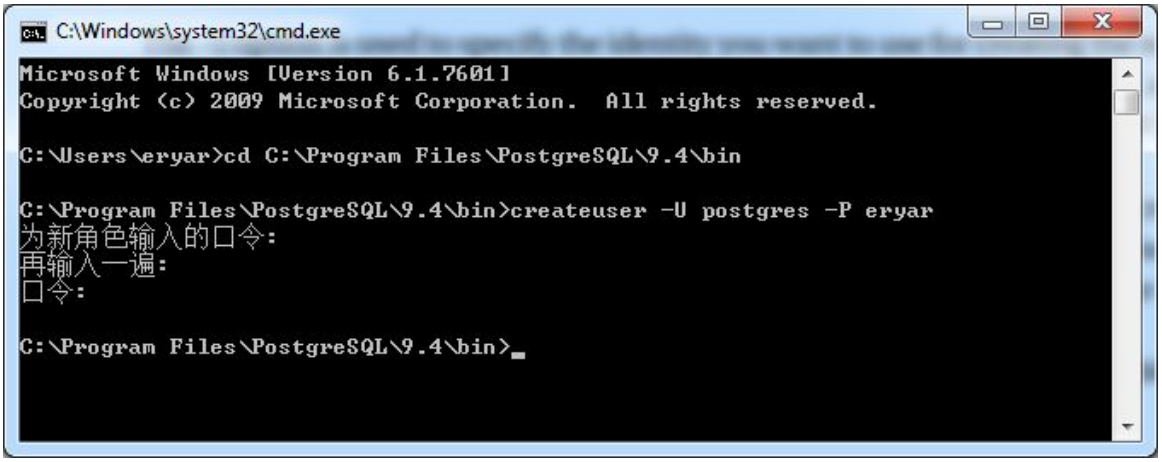


Figure 2.1 Create User by createuser.exe

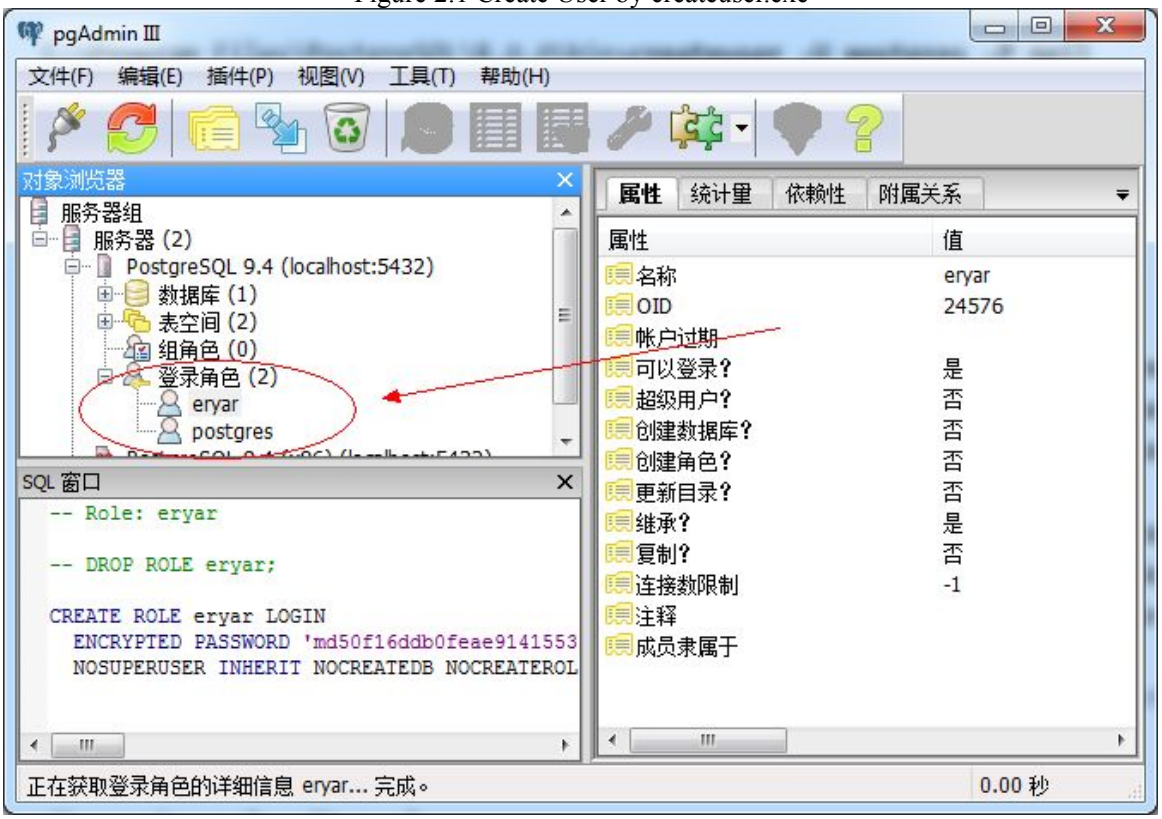


Figure 2.2 View user in pgAdmin

－U 选项用来指定创建新用户时使用的账号，必须为 PostgreSQL 具有创建用户权限的用户；
－P 选项为使用程序 createuser 创建的新用户的用户名；

当然，也可以在 pgAdmin 中直接创建用户，图形化的界面更为直观。

3. Creating the Database

在 Windows 系统中创建数据库使用是程序 createdb.exe，用法如下：

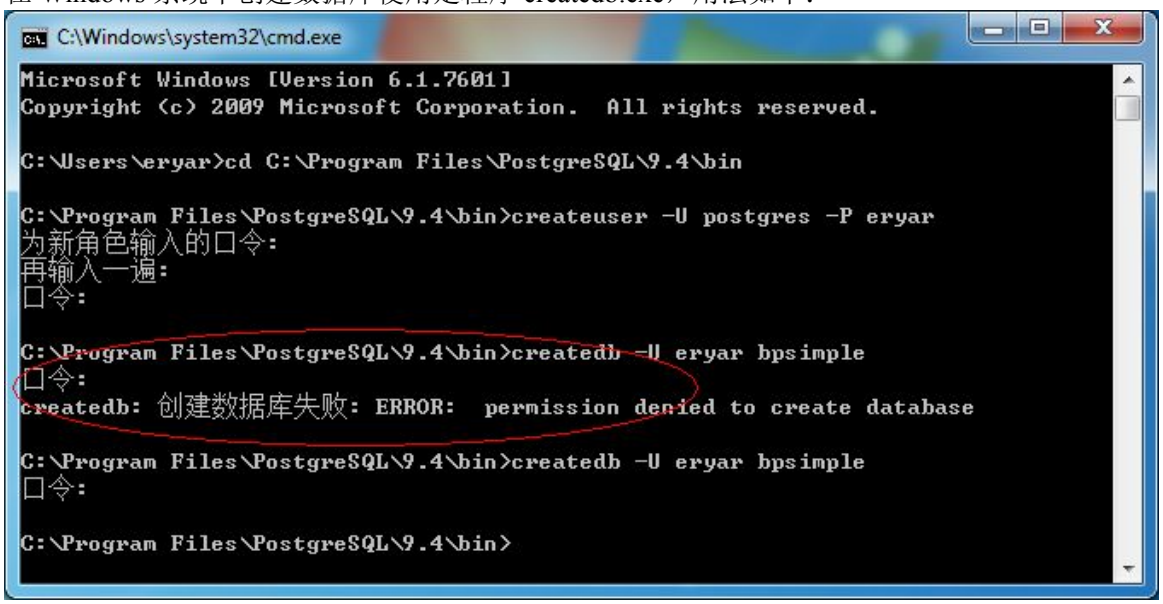


Figure 3.1 Create the Database by createdb.exe

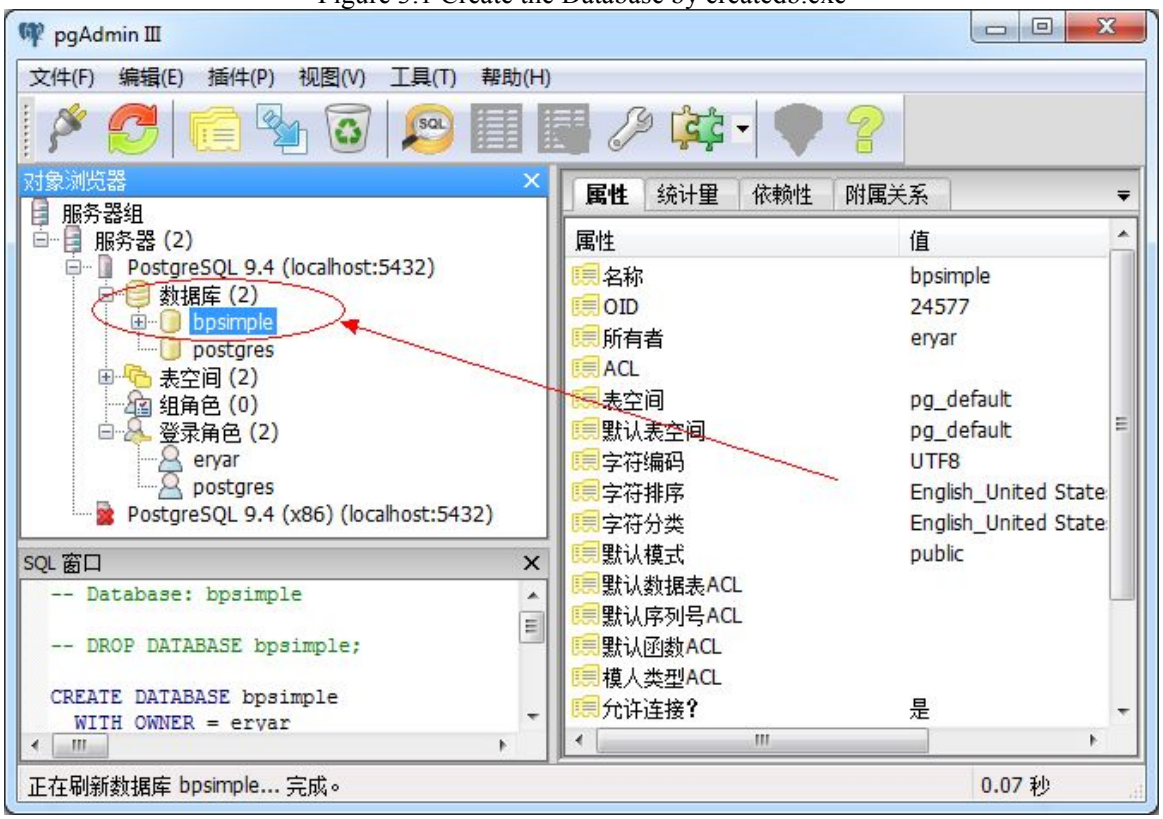


Figure 3.2 Databases in pgAdmin

新版本 9.4 的 createdb.exe 创建出来的用户没有询问是否有创建新数据库的权限。修改后即可。成功创建数据库后，就可以输入以下命令来连接了：

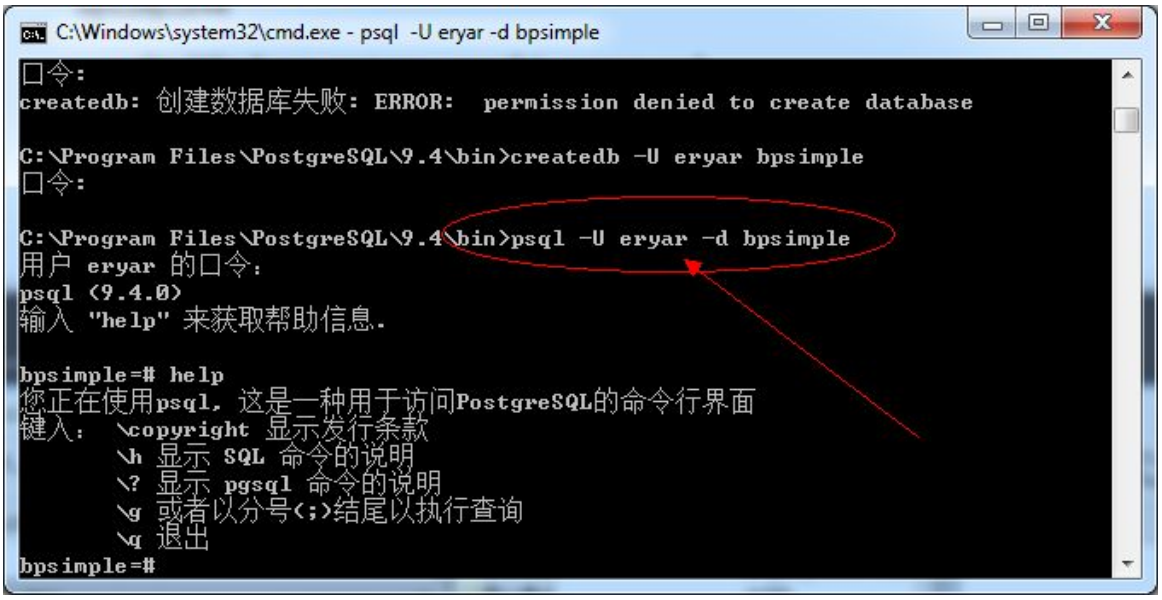


Figure 3.3 Connect to the database in psql

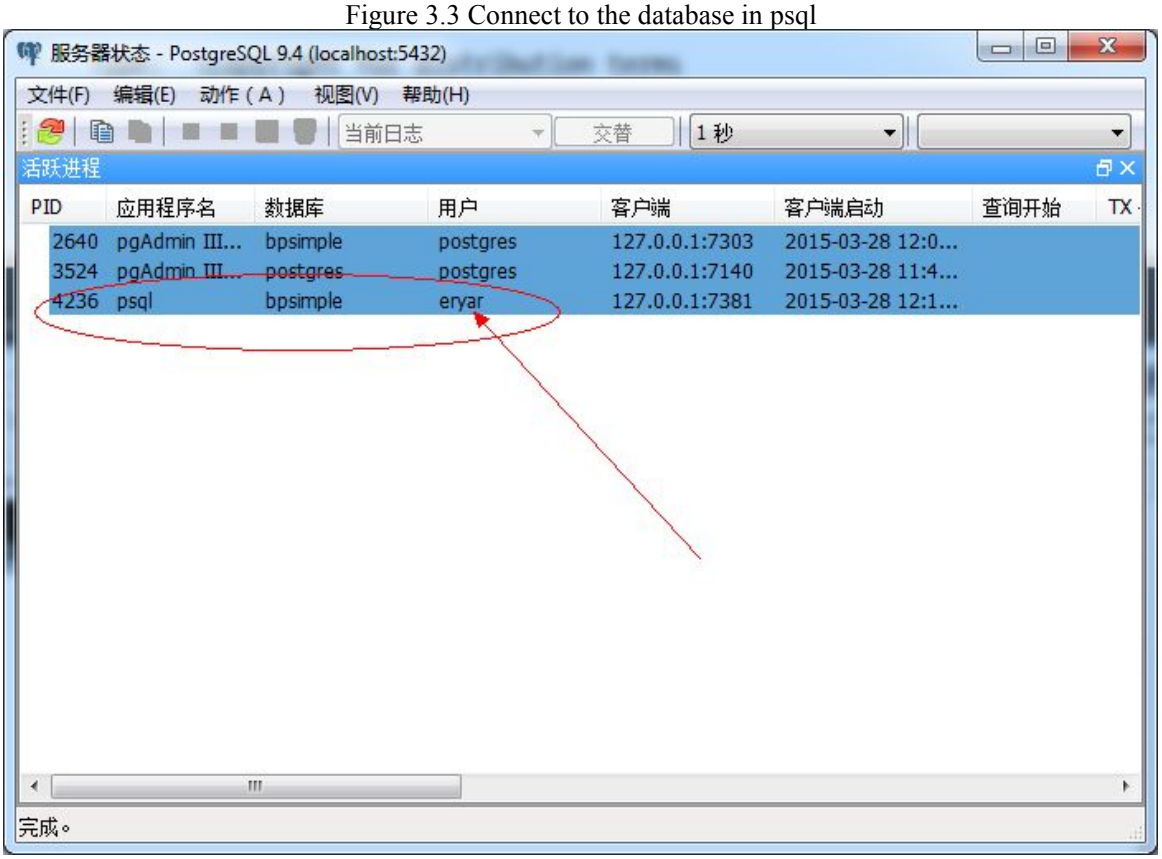


Figure 3.4 Server status

如图 3.4 所示，连接成功后，会从服务器状态中看到相关的连接信息。

4. Creating the Tables

连接到数据库后，psql 提供了一些基本命令，如下表 4.1 所示：

Command	Description
\?	Get a help message
\do	List operators
\dt	List tables
\dT	List types
\h <*cmd*>	Get help on a SQL command; replace <*cmd*> with the actual command
\i <*filename*>	Execute commands read from the filename <*filename*>
\r	Reset the buffer (discard any typing)
\q	Quit psql

Table 4.1 Basic psql Commands

由上表可知，可以使用 psql 的\i 命令来执行相关的表格创建、插入数据等操作。

```
-- customer table
CREATE TABLE customer
(
customer_id serial ,
title char(4) ,
fname varchar(32) ,
lname varchar(32) NOT NULL,
addressline varchar(64) ,
town varchar(32) ,
zipcode char(10) NOT NULL,
phone varchar(16) ,
CONSTRAINT customer_pk PRIMARY KEY(customer_id)
);

-- item table
CREATE TABLE item
(
item_id serial ,
description varchar(64) NOT NULL,
cost_price numeric(7,2) ,
sell_price numeric(7,2) ,
CONSTRAINT item_pk PRIMARY KEY(item_id)
);

-- orderinfo table
CREATE TABLE orderinfo
(
orderinfo_id serial ,
customer_id integer NOT NULL,
date_placed date NOT NULL,
```

```
date_shipped date ,
shipping numeric(7,2) ,
CONSTRAINT orderinfo_pk PRIMARY KEY(orderinfo_id)
);

-- stock table
CREATE TABLE stock
(
item_id integer NOT NULL,
quantity integer NOT NULL,
CONSTRAINT stock_pk PRIMARY KEY(item_id)
);

-- orderline table
CREATE TABLE orderline
(
orderinfo_id integer NOT NULL,
item_id integer NOT NULL,
quantity integer NOT NULL,
CONSTRAINT orderline_pk PRIMARY KEY(orderinfo_id, item_id)
);

-- barcode table
CREATE TABLE barcode
(
barcode_ean char(13) NOT NULL,
item_id integer NOT NULL,
CONSTRAINT barcode_pk PRIMARY KEY(barcode_ean)
);
```

将上述 sql 保存为 create_tables-bpsimple.sql，并在 psql 中执行，如下图所示：

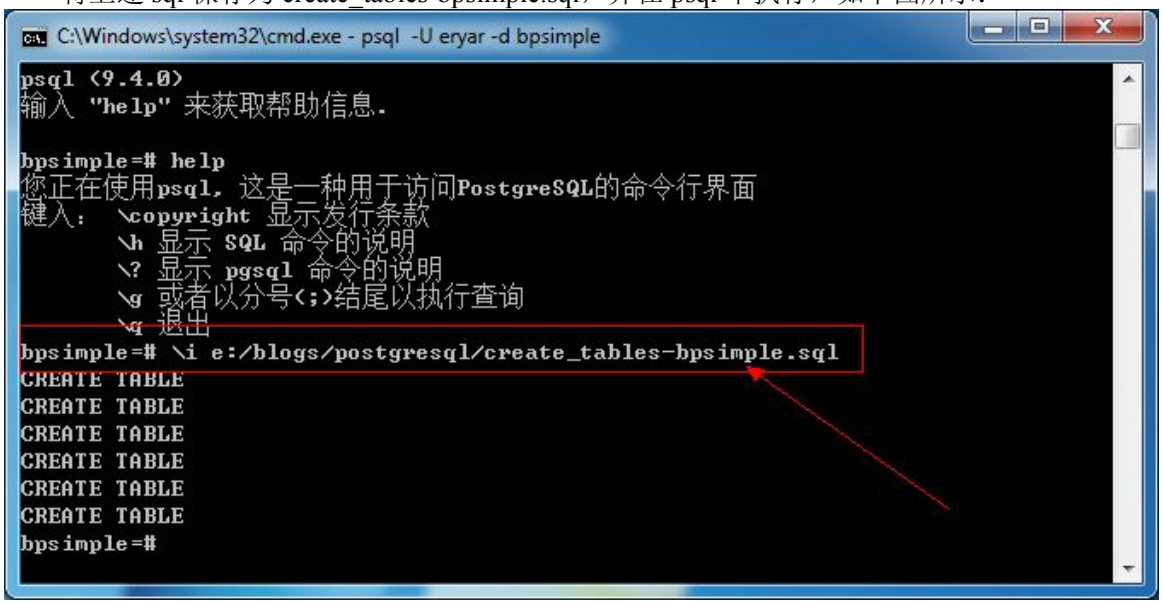


Figure 4.1 Create Tables by SQL File

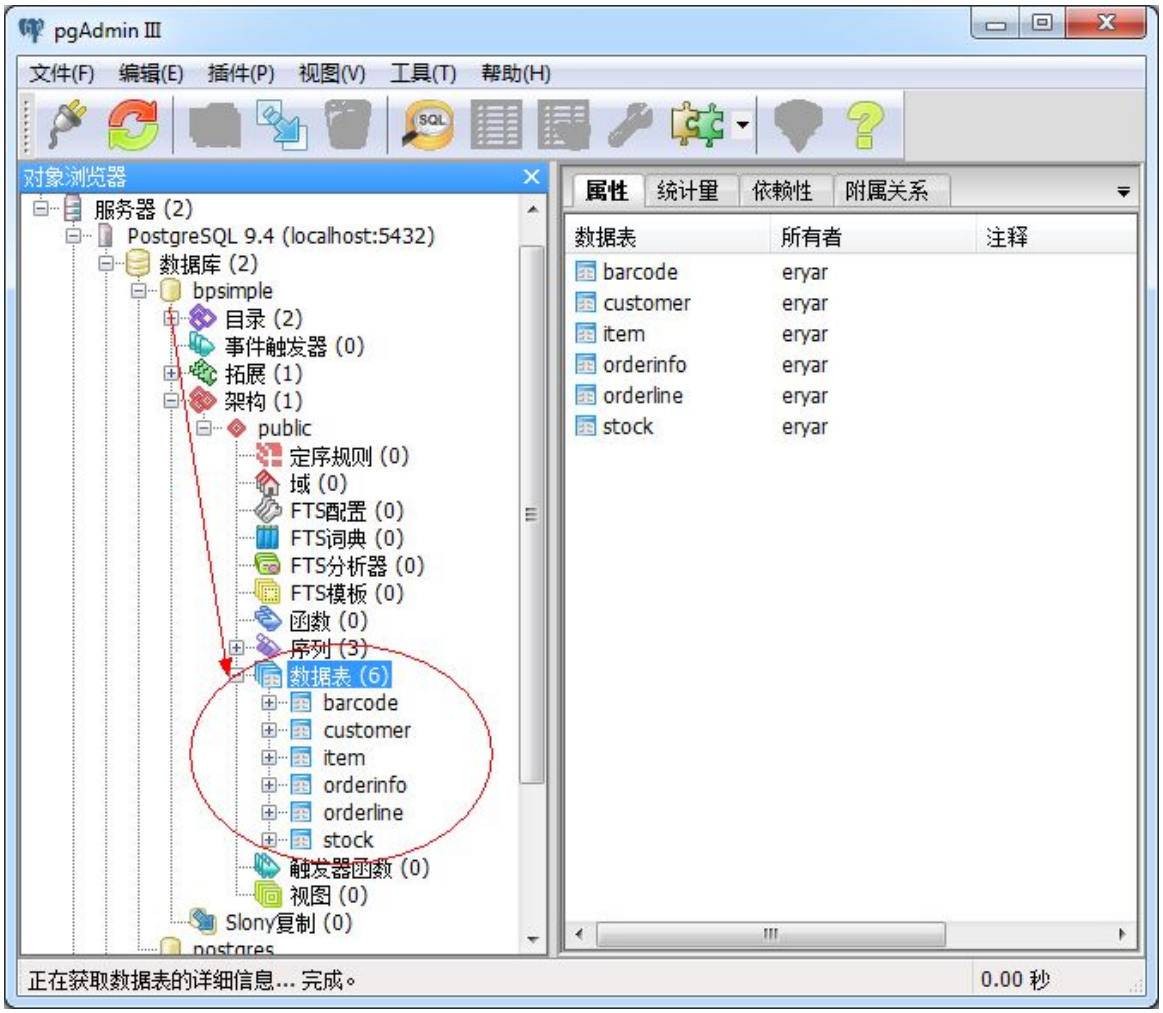


Figure 4.2 Tables in pgAdmin

5. Populating the Tables

与创建表的方式一样，将下述 SQL 保存为文件 pop_tablenames.sql，并在 psql 中执行\i 命令，将数据都插入到相应的表格中，如下所示：

```
-- customer table
INSERT INTO customer(title, fname, lname, addressline, town, zipcode, phone)
VALUES('Miss','Jenny','Stones','27 Rowan Avenue','Hightown','NT2 1AQ','023 9876');
INSERT INTO customer(title, fname, lname, addressline, town, zipcode, phone)
VALUES('Mr','Andrew','Stones','52 The Willows','Lowtown','LT5 7RA','876 3527');
INSERT INTO customer(title, fname, lname, addressline, town, zipcode, phone)
VALUES('Miss','Alex','Matthew','4 The Street','Nicetown','NT2 2TX','010 4567');
INSERT INTO customer(title, fname, lname, addressline, town, zipcode, phone)
VALUES('Mr','Adrian','Matthew','The Barn','Yuleville','YV67 2WR','487 3871');
INSERT INTO customer(title, fname, lname, addressline, town, zipcode, phone)
VALUES('Mr','Simon','Cozens','7 Shady Lane','Oakenham','OA3 6QW','514 5926');
INSERT INTO customer(title, fname, lname, addressline, town, zipcode, phone)
VALUES('Mr','Neil','Matthew','5 Pasture Lane','Nicetown','NT3 7RT','267 1232');
INSERT INTO customer(title, fname, lname, addressline, town, zipcode, phone)
VALUES('Mr','Richard','Stones','34 Holly Way','Bingham','BG4 2WE','342 5982');
INSERT INTO customer(title, fname, lname, addressline, town, zipcode, phone)
VALUES('Mrs','Ann','Stones','34 Holly Way','Bingham','BG4 2WE','342 5982');
INSERT INTO customer(title, fname, lname, addressline, town, zipcode, phone)
VALUES('Mrs','Christine','Hickman','36 Queen Street','Histon','HT3 5EM','342 5432');
INSERT INTO customer(title, fname, lname, addressline, town, zipcode, phone)
VALUES('Mr','Mike','Howard','86 Dysart Street','Tibsville','TB3 7FG','505 5482');
INSERT INTO customer(title, fname, lname, addressline, town, zipcode, phone)
VALUES('Mr','Dave','Jones','54 Vale Rise','Bingham','BG3 8GD','342 8264');
INSERT INTO customer(title, fname, lname, addressline, town, zipcode, phone)
VALUES('Mr','Richard','Neill','42 Thatched Way','Winersby','WB3 6GQ','505 6482');
INSERT INTO customer(title, fname, lname, addressline, town, zipcode, phone)
VALUES('Mrs','Laura','Hardy','73 Margarita Way','Oxbridge','OX2 3HX','821 2335');
INSERT INTO customer(title, fname, lname, addressline, town, zipcode, phone)
VALUES('Mr','Bill','O Neill','2 Beamer Street','Welltown','WT3 8GM','435 1234');
INSERT INTO customer(title, fname, lname, addressline, town, zipcode, phone)
VALUES('Mr','David','Hudson','4 The Square','Milltown','MT2 6RT','961 4526');

-- item table
INSERT INTO item(description, cost_price, sell_price)
VALUES('Wood Puzzle', 15.23, 21.95);
INSERT INTO item(description, cost_price, sell_price)
VALUES('Rubik Cube', 7.45, 11.49);
INSERT INTO item(description, cost_price, sell_price)
VALUES('Linux CD', 1.99, 2.49);
INSERT INTO item(description, cost_price, sell_price)
VALUES('Tissues', 2.11, 3.99);
INSERT INTO item(description, cost_price, sell_price)
VALUES('Picture Frame', 7.54, 9.95);
INSERT INTO item(description, cost_price, sell_price)
```

```
VALUES('Fan Small', 9.23, 15.75);
INSERT INTO item(description, cost_price, sell_price)
VALUES('Fan Large', 13.36, 19.95);
INSERT INTO item(description, cost_price, sell_price)
VALUES('Toothbrush', 0.75, 1.45);
INSERT INTO item(description, cost_price, sell_price)
VALUES('Roman Coin', 2.34, 2.45);
INSERT INTO item(description, cost_price, sell_price)
VALUES('Carrier Bag', 0.01, 0.0);
INSERT INTO item(description, cost_price, sell_price)
VALUES('Speakers', 19.73, 25.32);

-- orderinfo table
INSERT INTO orderinfo(customer_id, date_placed, date_shipped, shipping)
VALUES(3,'03-13-2000','03-17-2000', 2.99);
INSERT INTO orderinfo(customer_id, date_placed, date_shipped, shipping)
VALUES(8,'06-23-2000','06-24-2000', 0.00);
INSERT INTO orderinfo(customer_id, date_placed, date_shipped, shipping)
VALUES(15,'09-02-2000','09-12-2000', 3.99);
INSERT INTO orderinfo(customer_id, date_placed, date_shipped, shipping)
VALUES(13,'09-03-2000','09-10-2000', 2.99);
INSERT INTO orderinfo(customer_id, date_placed, date_shipped, shipping)
VALUES(8,'07-21-2000','07-24-2000', 0.00);


-- stock table
INSERT INTO stock(item_id, quantity) VALUES(1,12);
INSERT INTO stock(item_id, quantity) VALUES(2,2);
INSERT INTO stock(item_id, quantity) VALUES(4,8);
INSERT INTO stock(item_id, quantity) VALUES(5,3);
INSERT INTO stock(item_id, quantity) VALUES(7,8);
INSERT INTO stock(item_id, quantity) VALUES(8,18);
INSERT INTO stock(item_id, quantity) VALUES(10,1);

-- orderline table
INSERT INTO orderline(orderinfo_id, item_id, quantity) VALUES(1, 4, 1);
INSERT INTO orderline(orderinfo_id, item_id, quantity) VALUES(1, 7, 1);
INSERT INTO orderline(orderinfo_id, item_id, quantity) VALUES(1, 9, 1);
INSERT INTO orderline(orderinfo_id, item_id, quantity) VALUES(2, 1, 1);
INSERT INTO orderline(orderinfo_id, item_id, quantity) VALUES(2, 10, 1);
INSERT INTO orderline(orderinfo_id, item_id, quantity) VALUES(2, 7, 2);
INSERT INTO orderline(orderinfo_id, item_id, quantity) VALUES(2, 4, 2);
INSERT INTO orderline(orderinfo_id, item_id, quantity) VALUES(3, 2, 1);
INSERT INTO orderline(orderinfo_id, item_id, quantity) VALUES(3, 1, 1);
INSERT INTO orderline(orderinfo_id, item_id, quantity) VALUES(4, 5, 2);
INSERT INTO orderline(orderinfo_id, item_id, quantity) VALUES(5, 1, 1);
INSERT INTO orderline(orderinfo_id, item_id, quantity) VALUES(5, 3, 1);
```

```
-- barcode table
INSERT INTO barcode(barcode_ean, item_id) VALUES('6241527836173', 1);
INSERT INTO barcode(barcode_ean, item_id) VALUES('6241574635234', 2);
INSERT INTO barcode(barcode_ean, item_id) VALUES('6264537836173', 3);
INSERT INTO barcode(barcode_ean, item_id) VALUES('6241527746363', 3);
INSERT INTO barcode(barcode_ean, item_id) VALUES('7465743843764', 4);
INSERT INTO barcode(barcode_ean, item_id) VALUES('3453458677628', 5);
INSERT INTO barcode(barcode_ean, item_id) VALUES('6434564564544', 6);
INSERT INTO barcode(barcode_ean, item_id) VALUES('8476736836876', 7);
INSERT INTO barcode(barcode_ean, item_id) VALUES('6241234586487', 8);
INSERT INTO barcode(barcode_ean, item_id) VALUES('9473625532534', 8);
INSERT INTO barcode(barcode_ean, item_id) VALUES('9473627464543', 8);
INSERT INTO barcode(barcode_ean, item_id) VALUES('4587263646878', 9);
INSERT INTO barcode(barcode_ean, item_id) VALUES('9879879837489', 11);
INSERT INTO barcode(barcode_ean, item_id) VALUES('2239872376872', 11);
```

输入命令如下图所示：

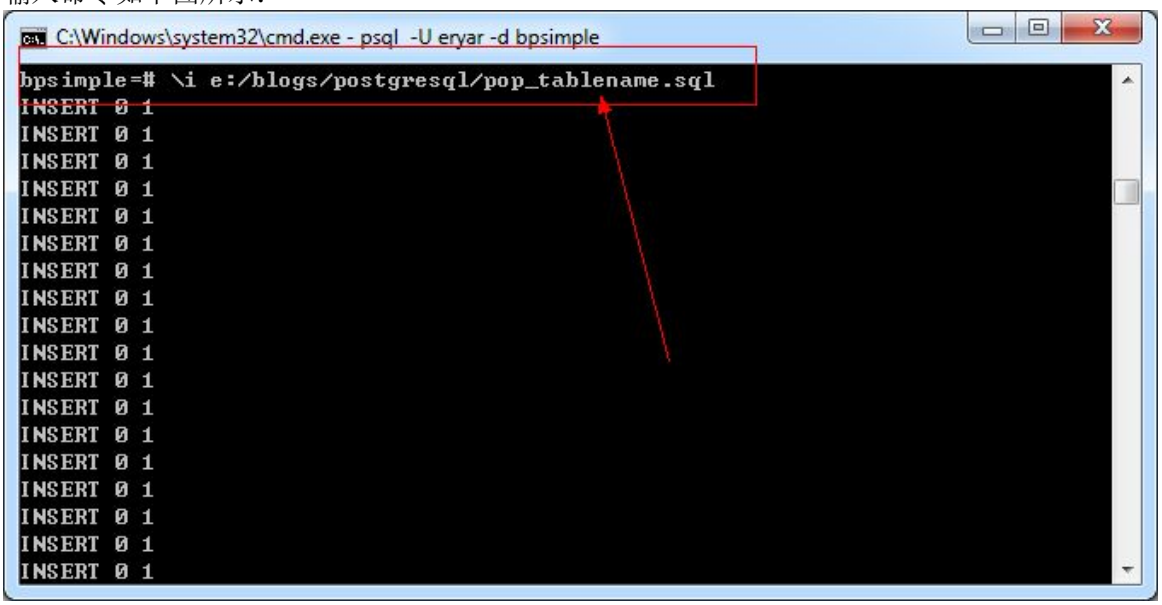


Figure 5.1 Insert data to tables

6. Accessing the Data

在 psql 中输入\dt 命令来查看数据库中的表格，如下图 6.1 所示：

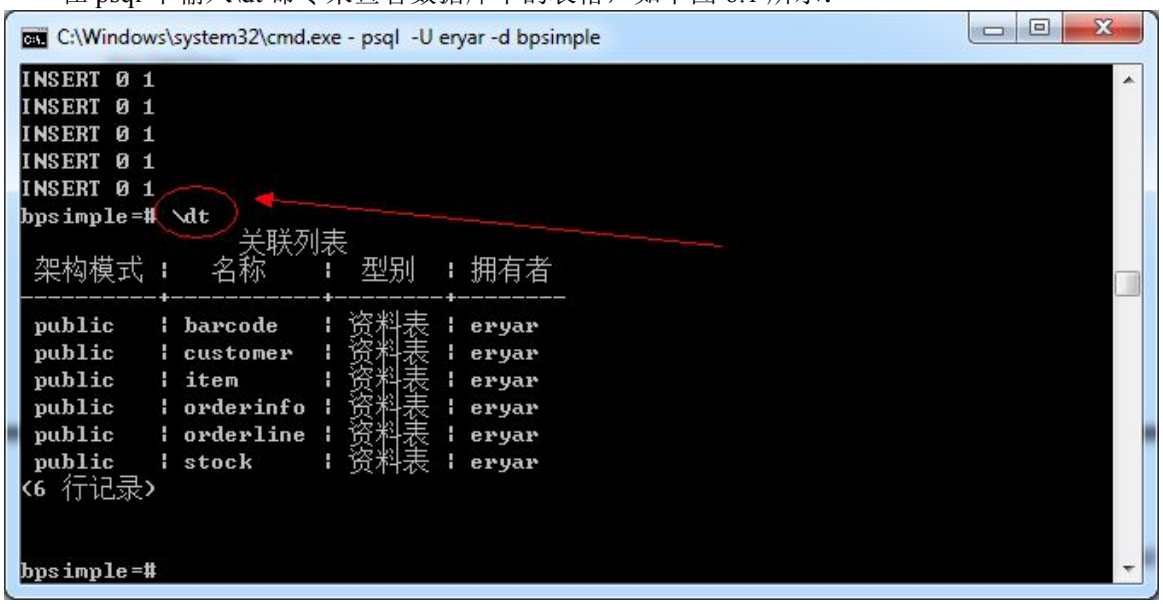


Figure 6.1 Use \dt command to list tables

插入数据后，可以用 SELECT 命令来简单查看一下数据，如下图所示：

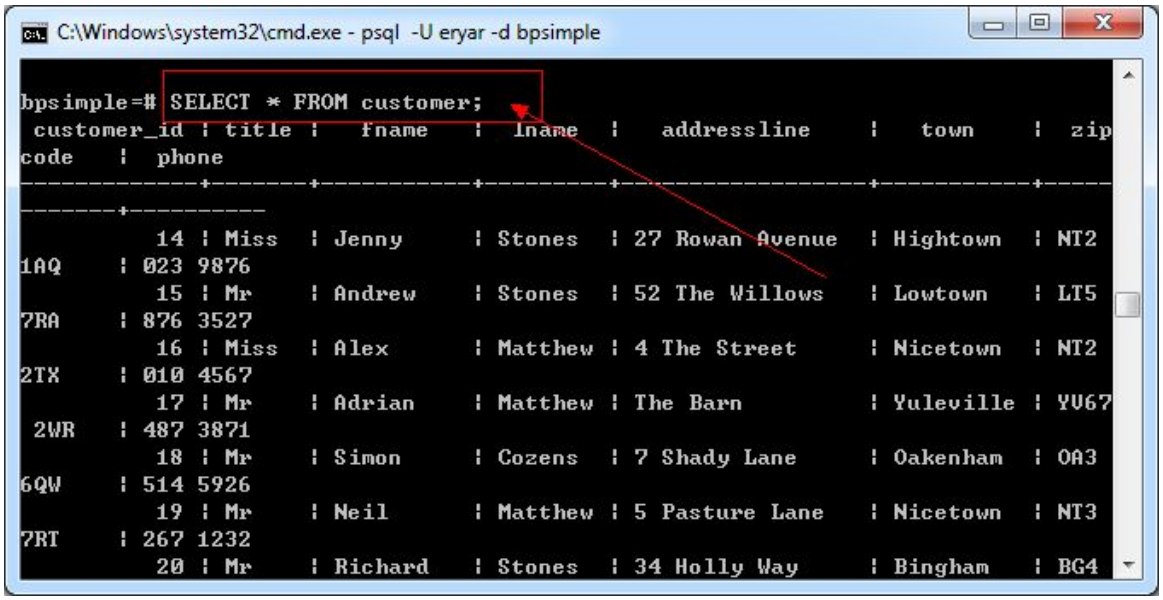
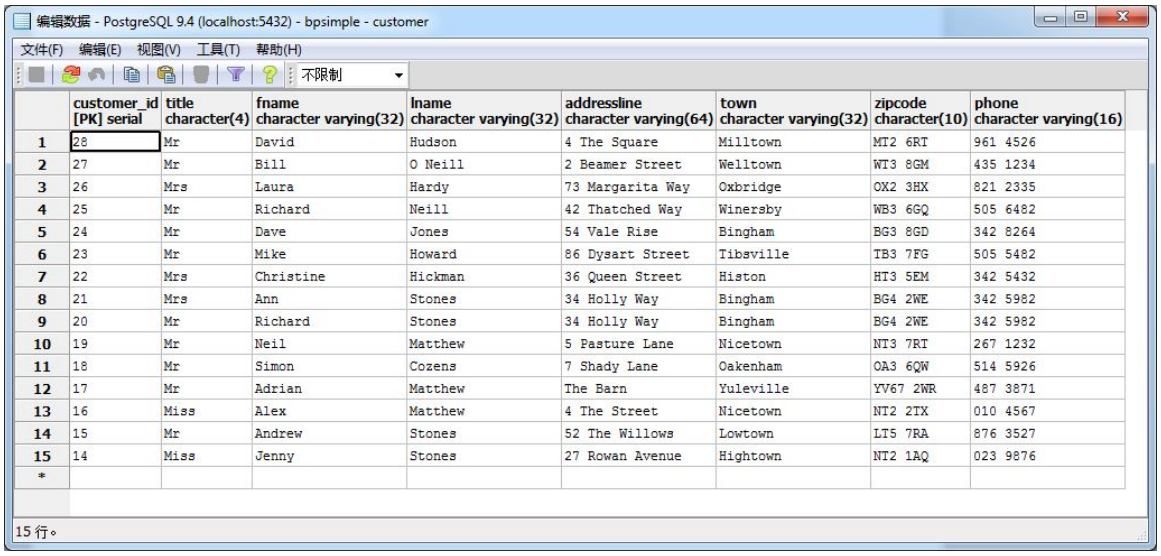


Figure 6.2 Query the data

也可以用同样的命令来查询其他表中的数据。当然也可以用 pgAdmin 来查看数据，如下图 6.3 所示：



	customer_id [PK] serial	title character(4)	fname character varying(32)	lname character varying(32)	addressline character varying(64)	town character varying(32)	zipcode character(10)	phone character varying(16)
1	28	Mr	David	Hudson	4 The Square	Milltown	MT2 6RT	961 4526
2	27	Mr	Bill	O Neill	2 Beamer Street	Welltown	WT3 8GM	435 1234
3	26	Mrs	Laura	Hardy	73 Margarita Way	Oxbridge	OX2 3HX	821 2335
4	25	Mr	Richard	Neill	42 Thatched Way	Winersby	WB3 6GQ	505 6482
5	24	Mr	Dave	Jones	54 Vale Rise	Bingham	BG3 8GD	342 8264
6	23	Mr	Mike	Howard	86 Dysart Street	Tibsville	TB3 7FG	505 5482
7	22	Mrs	Christine	Hickman	36 Queen Street	Histon	HT3 5EM	342 5432
8	21	Mrs	Ann	Stones	34 Holly Way	Bingham	BG4 2WE	342 5982
9	20	Mr	Richard	Stones	34 Holly Way	Bingham	BG4 2WE	342 5982
10	19	Mr	Neil	Matthew	5 Pasture Lane	Nicetown	NT3 7RT	267 1232
11	18	Mr	Simon	Cozens	7 Shady Lane	Oakenham	OA3 6QW	514 5926
12	17	Mr	Adrian	Matthew	The Barn	Yuleville	YV67 2WR	487 3871
13	16	Miss	Alex	Matthew	4 The Street	Nicetown	NT2 2TX	010 4567
14	15	Mr	Andrew	Stones	52 The Willows	Lowtown	LT5 7RA	876 3527
15	14	Miss	Jenny	Stones	27 Rowan Avenue	Hightown	NT2 1AQ	023 9876
*								

15 行。

Figure 6.3 View and Edit Data in pgAdmin

7. Summary

通过对国外软件的简单介绍，说明了数据库管理系统在软件中的重要作用，并说明了在数据库与应用层之间的数据框架的重要性。

由于 PostgreSQL 基于类似 BSD 协议，开源且免费使用，很自由，所以选择 PostgreSQL 来学习数据库的知识。

通过使用 psql 来创建数据表及插入测试数据，便于对 PostgreSQL 做进一步的学习。

8. References

1. Neil Matthew, Richard Stones. Beginning Databases with PostgreSQL. Apress. 2005
2. Richard Blum. PostgreSQL 8 FOR Windows. The McGraw-Hill. 2007
3. http://www.postgresql.org/docs/books/