PostgreSQL从菜鸟到专家 Windows中安装PostgreSQL

四月 30, 2012 by [admin](http://www.mysqlops.com/author/admin) · [Leave a Comment](http://www.mysqlops.com/2012/04/30/postgresql-windows.html#respond)

**在Windows中安装PostgreSQL**

在本小节开始前，先给Windows用户一些好消息。虽然PostgreSQL是为类UNIX平台开发的，它却是可移植的。

现在已经可能在Windows上写PostgreSQL客户端程序，而且从7.1版开始，PostgreSQL可以编译安装和作为一个PostgreSQL服务器运行在Windows NT 4，2000，XP和Server 2003中。

从PostgreSQL 8.0开始，已经有了Windows本地版本了，为服务端和客户端程序提供了Windows的安装程序，这让在Windows下的安装非常轻松。在8.0版之前，Windows用户需要安装一些软件来在Windows上提供UNIX功能。

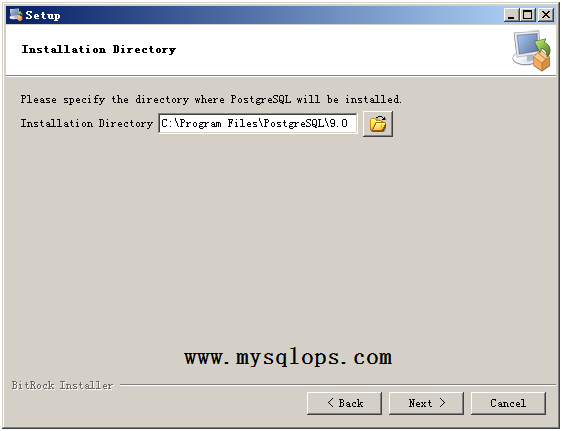
注：PostgreSQL 8.0支持Windows 2000，Windows XP和Windows Server 2003，由于Windows 95，98和Me不提供一些它需要的功能所以它无法在这些平台运行。它可以在Windows NT上跑起来，但必须手工安装，因为PostgreSQL的安装程序无法在Windows NT上正常运行。

看起来像Linux一样的开源的操作系统平台是PostgreSQL一样的开源数据库的自然归宿。实际上，我们不推荐在桌面版的Windows上运行生产用的数据库，但在Windows上安装有它的优点。例如在客户程序所在的机器上拥有PostgreSQL的工具套件例如psql在测试新装数据库和查找连接故障的时候非常有用。虽然你不需要在Windows上运行服务程序。在开发机上运行数据库服务器可以避免开发人员需要共享服务器实例而带来的问题。

使用Windows安装程序

EnterpriseDB公司（公司网站：http://www.enterprisedb.com）提供了Windows版的PostgreSQL安装程序。安装程序的名字类似于postgresql-9.0.1-1-windows.exe。安装程序需要系统安装了2.0或更高版本的Windows安装器。Windows XP以及以后的版本中已经包含了恰当的版本。如果有必要，Windows安装期可以从微软的官方网站http://www.microsoft.com下载（搜索“Windows Installer redistributable”）。

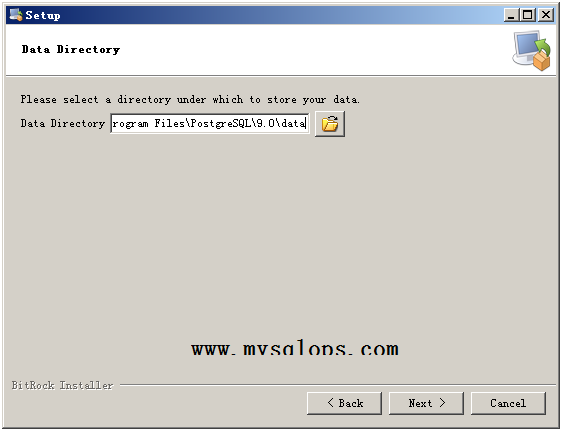
双击这个文件就可以启动安装程序，进入安装向导。在选择安装语言和阅读安装注记后，你将看到选择安装目录对话框，如图3-3所示。

[](http://www.mysqlops.com/wp-content/uploads/2012/04/postgresql-windows-1.png)

**图3-3 PostgreSQL安装位置**

在这一步，你可以选择你的PostgreSQL应用程序安装路径。通过点击打开文件夹按钮，选择安装路径。默认的安装路径为C:\Program Files\PostgreSQL\9.0。

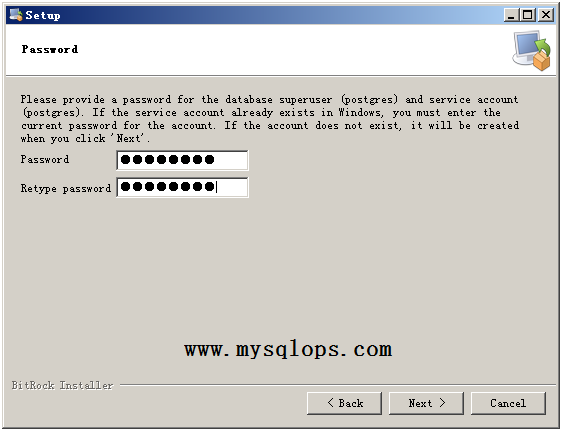
点击下一步后，安装向导进入数据目录设置页面。如图3-4所示：

[](http://www.mysqlops.com/wp-content/uploads/2012/04/postgresql-windows-2.png)

**图3-4 PostgreSQL数据目录**

这一步，你可以选择你的PostgreSQL数据存放的路径。通过点击打开文件夹按钮，选择存放路径。默认的存放路径为C:\Program Files\PostgreSQL\9.0\data。

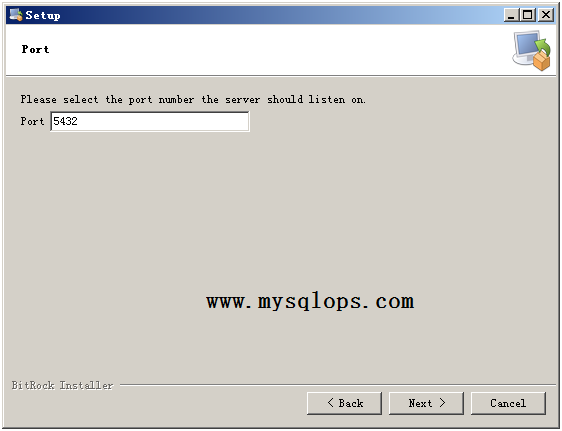
点击下一步，安装程序进入填写密码页面，如图3-5所示：

[](http://www.mysqlops.com/wp-content/uploads/2012/04/postgresql-windows-3.png)

**图3-5 输入postgres用户密码**

和UNIX系统一样，PostgreSQL不允许以管理员身份运行程序，这避免了拥有管理员权限的用户运行接受网络连接的服务程序潜在的安全问题。如果发现了PostgreSQL的安全若得并被使用溢出攻击，这样做的最坏情况只是PostgreSQL管理的数据被损坏，而不是整台服务器被攻陷。安装程序会使用一个叫postgres的用户运行服务程序。在这一步，安装向导将记录postgres用户的密码。如果这台机器已经安装过PostgreSQL或者已经存在postgres用户，则输入的密码必须是这个用户现在的密码，否则，安装向导将建立postgres用户并设置密码为输入的密码。输入两次相同的密码以确保你记得你输入的密码。

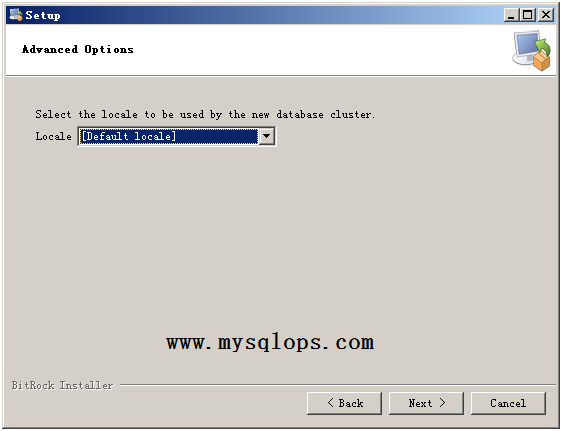
点击下一步，安装向导要求你填入服务监听的端口，如图3-6所示：

[](http://www.mysqlops.com/wp-content/uploads/2012/04/postgresql-windows-4.png)

**图3-6 输入服务监听端口**

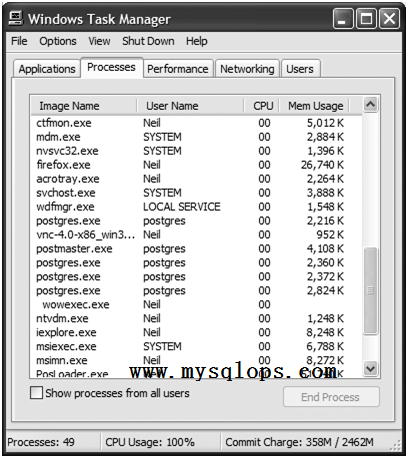
和UNIX下一样，默认的端口为5432，你可以设置为任何你想使用的端口。

下一步将设置新数据库的区域，如图3-7所示：

[](http://www.mysqlops.com/wp-content/uploads/2012/04/postgresql-windows-5.png)

**图3-7 选择新建数据库的区域**

选择恰当的区域，点击下一步后，向导提示将开始安装数据库。再点击下一步，如果不出错，安装过程将进行并正常完成。完成安装后，数据库服务进程应该已经运行。服务进程postmaster.exe和postgres.exe可以在任务管理器中看到，如图3-8所示：

[](http://www.mysqlops.com/wp-content/uploads/2012/04/postgresql-windows-6.png)

**图3-8 PostgreSQL进程**

PostgreSQL程序和工具被安装到开始菜单，如图3-8所示：

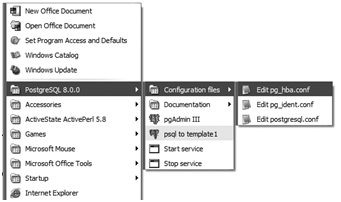
[](http://www.mysqlops.com/wp-content/uploads/2012/04/postgresql-windows-7.png)

图 3-8 PostgreSQL程序菜单

配置客户机访问

为了配置远程主机和用户可以连接到PostgreSQL服务，你需要编辑pg\_hba.conf文件。文件包含大量注释记录用于远程访问的选项。在我们的安装示例中，我们允许局域网中任何主机的任何用户访问服务器上的数据库。为了达到这个目的，我们添加以下的一行记录到文件尾：

host    all all 192.168.0.0/16  trust

这意味着所有IP地址由192.168开始的计算机可以访问所有的数据库。最简单的使配置生效的方法就是重启服务器。

Windows系统中的pg\_hba.conf文件和Linux和UNIX系统中的格式相同。其他的访问配置示例，请参考本章前面的“配置连接权限”小节。

建立示例数据库

现在我们已经让PostgreSQL运行起来了，我们将建立一个简单的数据库，我们给数据库起名为bpsimple，用来支持我们的客户订单表示例。这个数据库（与第8章建立的叫做bpfinal的改造版本）将在整个本书中使用。我们将在后面的章节全面讨论建立数据库和建表以及填充表的细节。在这里，我们只展示实现的步骤和SQL脚本，这样我们将有一个用于示例的数据库。在我们开始前，一个简单的检查PostgreSQL是否在运行的方法就是查找postmaster进程。在Windows系统中，通过任务管理器查找postmaster.exe进程。在UNIX和Linux系统中，运行以下的命令：

$ ps -el | grep post

如果有一个叫postmaster的进程在运行（名称可能是缩写显示），那么你已经在运行一个PostgreSQL服务程序了。

添加用户记录

在我们能建立一个数据库前，我们需要通过在系统中建立用户记录来告诉PostgreSQL有效的用户。PostgreSQL数据库系统的有效用户可以读数据，插入数据或者更新数据；建立自己的数据库以及控制对这些数据库管理的数据的访问。我们使用PostgreSQL的createuser工具建立用户记录。

在Linux和UNIX系统中，使用su命令（从root）变成PostgreSQL管理用户postgres。然后运行createuser注册用户。用户给出的用户登录名需要时有效的PostgreSQL用户。让我们为（UNIX/Linux的）用户neil建立数据库用户：

$ su

# su – postgres

pg$ /usr/local/pgsql/bin/createuser neil

Shall the new user be able to create databases? (y/n) y

Shall the new user be able to create new users (y/n) y

CREATE USER

pg$

在Windows系统中，打开一个命令行窗口并切换工作目录到PostgreSQL所在目录（在这里，我们按在默认位置：C:\Program Files\PostgreSQL\9.0），然后运行createuser.exe工具：

C:\Program Files\PostgreSQL\9.0\bin>createuser -U postgres -P neil

Enter password for new user:

Enter it again:

Shall the new user be allowed to create databases? (y/n) y

Shall the new user be allowed to create more new users? (y/n) y

Password:

CREATE USER

-U选项指出了你想要建立的新用户的编号。必须是PostgreSQL的用户才能建立用户，通常情况下就是postgres用户。-P选项通知createuser提示输入新用户的密码。

在这里，我们允许neil建立新数据库和建立新用户。本书中的一些例子中使用了另一个叫rick的用户，它允许建立数据库，但不允许新建用户。如果你想重新试试新建用户，现在就可以建立这个用户了。

一旦你建立了拥有这些权限的PostgreSQL用户，你就有能力建立bpsimple数据库了。

建立数据库

在Linux和UNIX系统中建立数据库，回到你自己的用户（非root用户）并运行以下命令：

$ /usr/local/pgsql/bin/createdb bpsimple

CREATE DATABASE

$

在Windows系统中，运行createdb.exe命令：

C:\Program Files\PostgreSQL\8.4\bin>createdb -U neil bpsimple

Password:

CREATE DATABASE

你先可以使用交互式终端工具psql（从本地）连接到服务器了。在Linux和UNIX系统中，使用以下命令：

$ /usr/local/pgsql/bin/psql -U neil -d bpsimple

psql (8.4.5)

Type “help” for help.

bpsimple =#

在Windows系统中，使用这个命令：

C:\Program Files\PostgreSQL\8.4\bin>psql -U neil -d bpsimple

Password:

Welcome to psql 8.4.5, the PostgreSQL interactive terminal.

Type:  \copyright for distribution terms

\h for help with SQL commands

\? for help with psql commands

\g or terminate with semicolon to execute query

\q to quit

Warning: Console codepage (850) differs from windows codepage (1252)

8-bit characters will not work correctly. See PostgreSQL

documentation “Installation on Windows” for details.

bpsimple=#

你也可以选择Windows开始菜单中的psql项目连接到template1数据库，然后在psql中切换数据库：

template1=# \c bpsimple

You are now connected to database “bpsimple”.

bpsimple=#

现在你已经登录到PostgreSQL，准备好执行命令了。你可以使用\q命令退回到shell。

下一步，我们将使用一套SQL语句来建立和填充示例数据库。

建表

你可以在psql的命令提示符后面通过输入SQL命令在你的bpsimple数据库中建表。但是，更简单的办法是下载代码包并解压后，通过命令\i <文件名>执行这些命令（psql中的\i命令可以用来执行脚本文件，脚本文件是由数组SQL语句和其他PostgreSQL命令组成的文本文件）。由于命令都是纯文本，所以只要你愿意你都可以以通过你的文本编辑器编辑它。

输入下面的命令运行create\_tables-bpsimple.sql来建表：

bpsimple=# \i create\_tables-bpsimple.sql

CREATE TABLE

…

bpsimple=#

把所有的数据库模式（表，索引以及存储过程）写入脚本文件是一个好习惯。只有这样，如果数据库需要重新建立，你可以通过脚本完成。脚本也可以用于在任何模式需要更新时使用。

以下为建立我们需要的表的SQL（第2章设计的表）这些SQL可以在create\_tables-bpsimple.sql代码包里头找到：

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)

);

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)

);

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)

);

CREATE TABLE stock

(

item\_id          integer               NOT NULL,

quantity         integer               NOT NULL,

CONSTRAINT       stock\_pk PRIMARY KEY(item\_id)

);

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)

);

CREATE TABLE barcode

(

barcode\_ean      char(13)              NOT NULL,

item\_id          integer               NOT NULL,

CONSTRAINT       barcode\_pk PRIMARY KEY(barcode\_ean)

);

移除表

假设以后的某天，你需要删除所有的表并重新开始，命令集就在drop\_tables.sql，就像这样：

DROP TABLE barcode;

DROP TABLE orderline;

DROP TABLE stock;

DROP TABLE orderinfo;

DROP TABLE item;

DROP TABLE customer;

DROP SEQUENCE customer\_customer\_id\_seq;

DROP SEQUENCE item\_item\_id\_seq;

DROP SEQUENCE orderinfo\_orderinfo\_id\_seq;

警告你一声，如果你删掉表，你也将丢失表里的数据！

注：drop\_tables.sql脚本也明确地删除了叫做序列生成器的特殊的属性，它被PostgreSQL用来管理数据自动顺序增长的列。在PostgreSQL 8.0和以后的版本，序列生成器将在相关表被删除时被自动删除，但我们保留了命令用于兼容以前的版本。

如果你在建表后运行了这个脚本你需要在尝试往表里头填充数据前再次运行create\_tables-bpsimple.sql。

填充表

最后，我们需要往表里头添加一些数据，或者叫做填充表。示例数据在pop\_tablename.sql里头。如果你想使用自己的数据，你得到的结果将和本书提供的不同。所以，除非你确信，否则最好是使用我们提供的示例数据。

在这里使用换行符的必要性是为了将命令打印在打印页上。你可以每个每行一条命令。你必须包含结尾的分号，这告诉psql每行SQL的结束点。

customer表

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表

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);

barcode表

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);

orderinfo表

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);

orderline表

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);

stock表

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);

在PostgreSQL系统运行，数据库建立表建立以及填充数据后，我们准备好继续探索PostgreSQL的功能了。