PostgreSQL

Lesson 1: PostgreSQL – An Introduction

Lesson Objectives

In this lesson, you will learn about:

- What is PostgreSQL?
- Features of PostgreSQL
- Architecture of PostgreSQL
- Creating Database
- PostgreSQL Datatypes
- Creating tables

What is PostgreSQL?

- PostgreSQL is a general purpose and object-relational database management system
- It is the most advanced open source database system
- PostgreSQL was designed to run on UNIX-like platforms
- It was designed to be portable so that it could run on various platforms such as Mac OS X, Solaris, and Windows.
- PostgreSQL requires very minimum maintained efforts because of its stability
- If you develop applications based on PostgreSQL, the total cost of ownership is low in comparison with other database management systems.

History of PostgreSQL

- PostgreSQL, originally called Postgres, was created at UCB by a computer science professor Michael Stonebraker
- Stonebraker started Postgres in 1986 as a follow up project to its predecessor, Ingres
- Postgres was developed between 1986-1994, a project meant to break new ground in database concepts such as exploration of "object relational" technologies
- In 1995, two Ph.D. students from Stonebraker's lab, Andrew Yu and Jolly Chen, replaced Postgres' POSTQUEL query language with an extended subset of SQL. They renamed the system to Postgres95

History of PostgreSQL

- In 1996, Postgres95 departed from academia and started a new life in the open source world when a group of dedicated developers outside of Berkeley, saw the promise of the system, and devoted themselves to its continued development
- With many new features and enhancements, the database system took its current name: PostgreSQL

Features of PostgreSQL

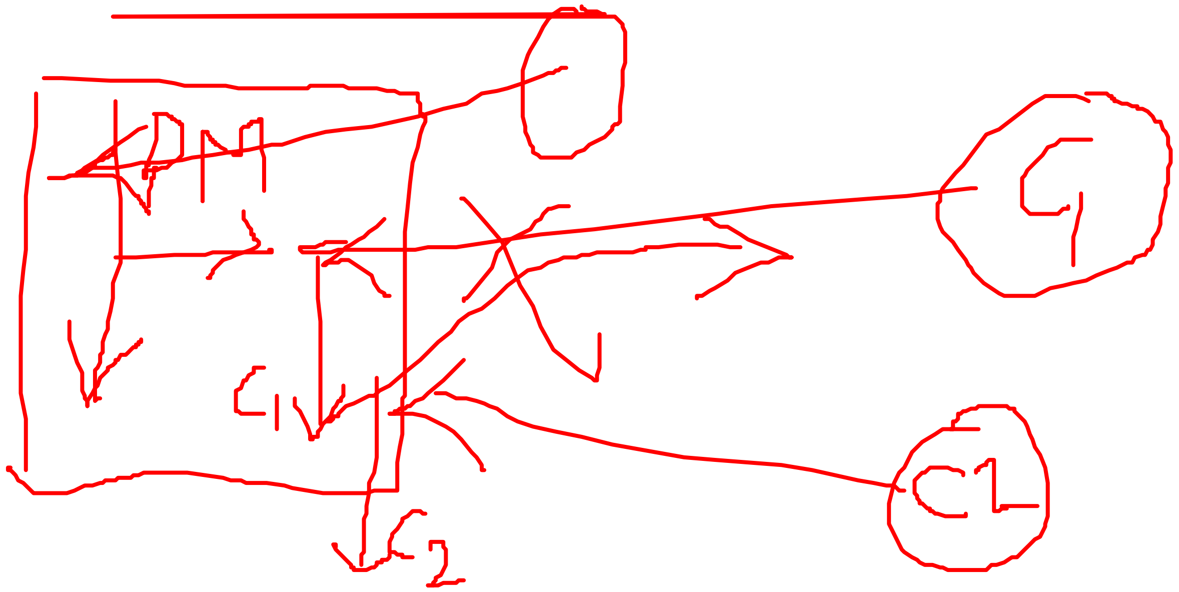
- User-defined types
- Table inheritance
- Sophisticated locking mechanism
- Foreign key referential integrity
- Views, rules, subquery
- Nested transactions (save points)
- Multi-version concurrency control (MVCC)
- Asynchronous replication

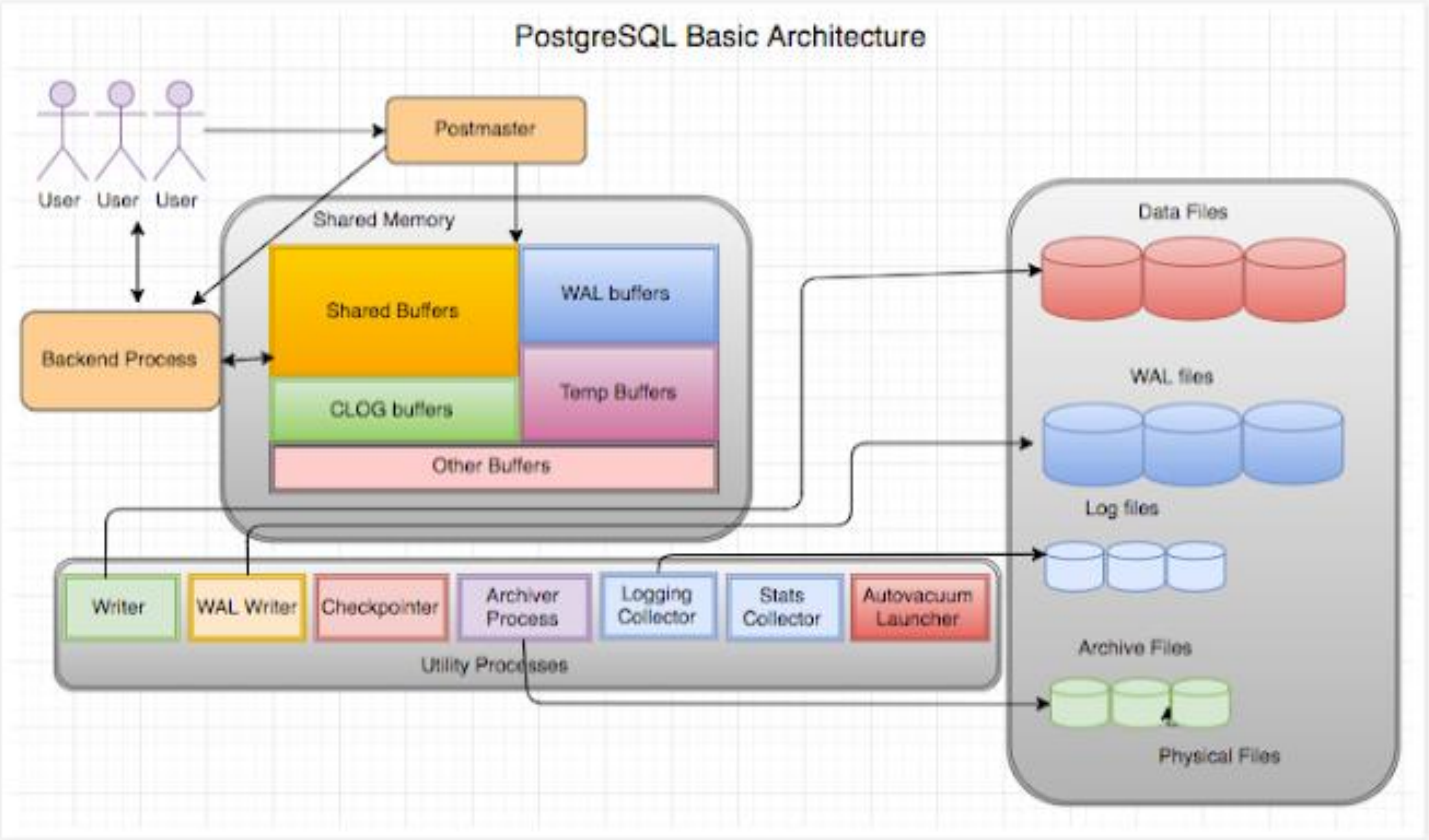
Architecture of PostgreSQL

- PostgreSQL uses client/server model
- PostgreSQL session consists of the following cooperating process:
  - A server process – which manages the database files, accepts connections to the database from client applications, and performs actions on the database on behalf of the clients. The database server program is called postmaster
  - The user’s client application that wants to perform database operations
  - Client applications can be very diverse in nature: a client could be a text-oriented tool, a graphical application, a web server that accesses the database to display web pages, or a specialized database maintenance tool.

Architecture of PostgreSQL

- The client and the server can be on different hosts
- The PostgreSQL server can handle multiple concurrent connections from clients
- it starts a new process for each connection
- the client and the new server process communicate without intervention by the original postmaster process
- the postmaster is always running, waiting for client connections, whereas client and associated server processes come and go.




PostgreSQL Basic Architecture
User
User
User
Postmaster
Shared Memory
Shared Buffers
WAL buffers
Backend Process
CLOG buffers
Temp Buffers
Other Buffers
Writer
WAL Writer
Checkpointer
Archiver Process
Logging Collector
Stats Collector
Autovacuum Launcher
Utility Processes
Data Files
WAL files
Log files
Archive Files
Physical Files

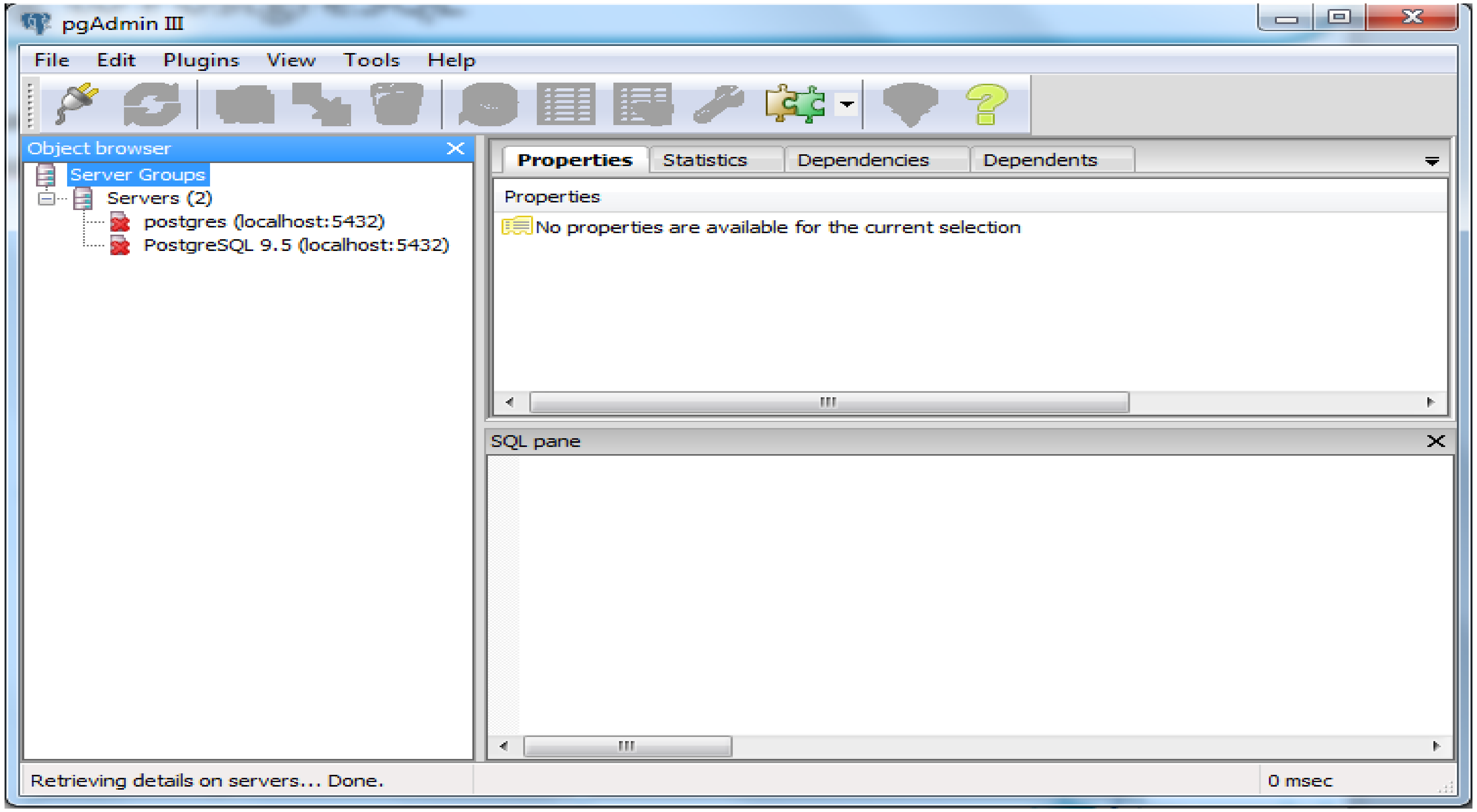
Installing PostgreSQL

- Requirements for installation of PostgreSQL:
  - *64bit CPU*
  - *64bit Operating System*
  - *2 Gigabytes of memory*
  - *Dual CPU/Core*
  - *RAID 1*

  - *postgresql-9.5.3-1-windows*

Connecting to PostgreSQL

- After installing PostgreSQL 9.5
- Open pgAdmin III



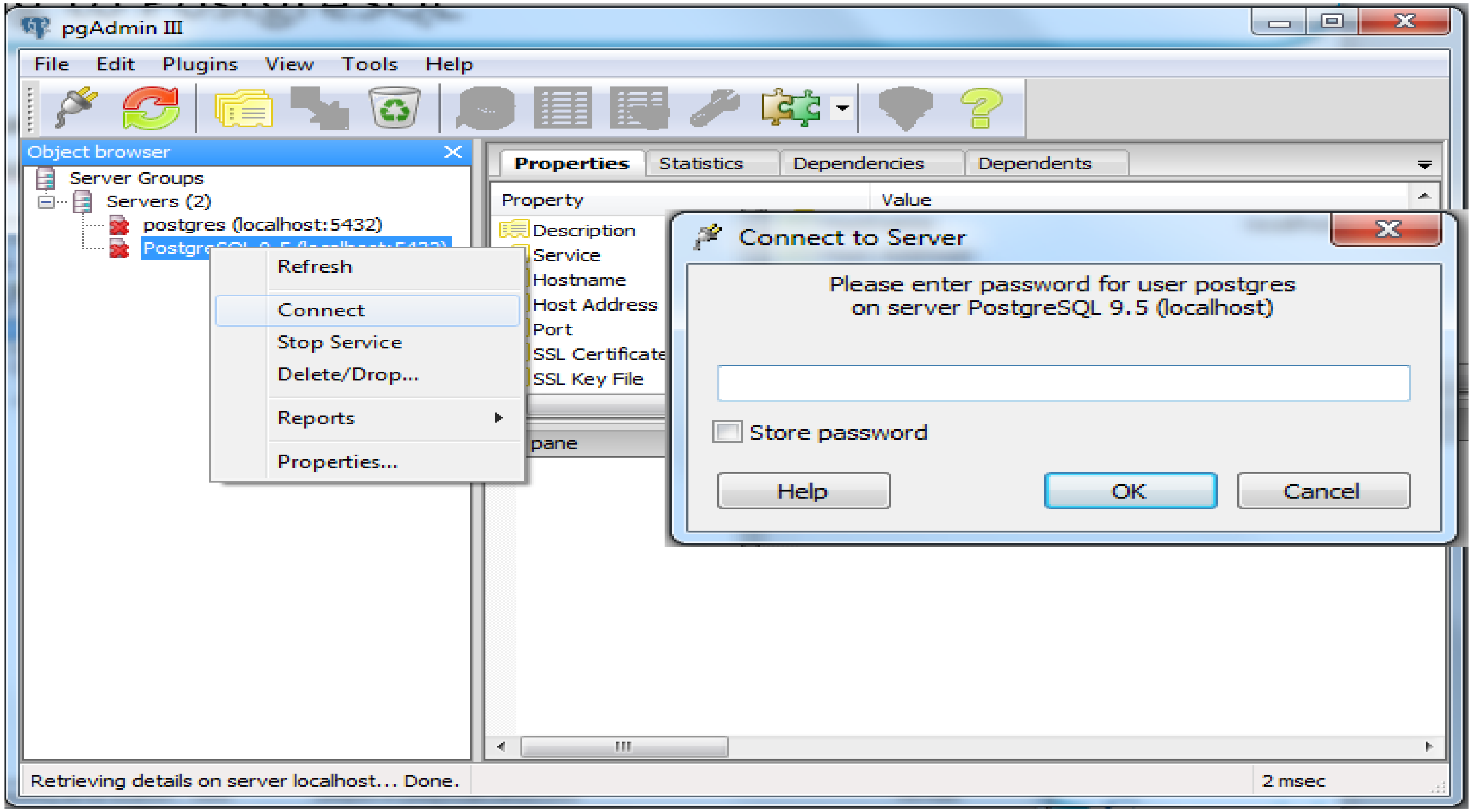
No properties are available for the current selection

Retrieving details on servers... Done.

0 msec

Connecting to PostgreSQL

- To connect to the database - click on connect and provide password



Please enter password for user postgres
on server PostgreSQL 9.5 (localhost)

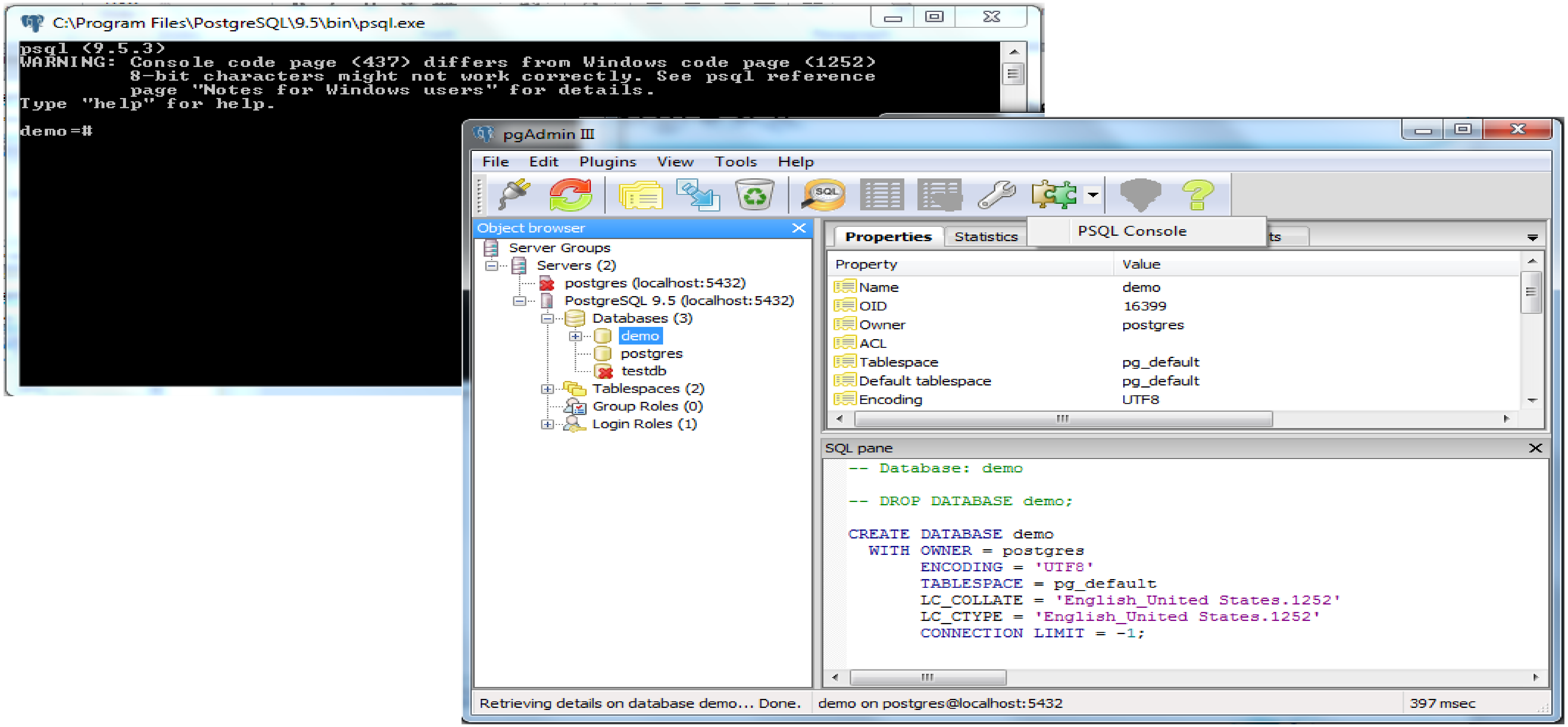
□ Store password

Retrieving details on server localhost... Done.

2 msec

Connecting to PostgreSQL

- Select database and open PSQL console



```
psql (9.5.3)
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.

demo=#
```

```
Server Groups
  Servers (2)
    postgres (localhost:5432)
    PostgreSQL 9.5 (localhost:5432)
      Databases (3)
        demo
        postgres
        testdb
      Tablespaces (2)
      Group Roles (0)
      Login Roles (1)
```

Property	Value
Name	demo
OID	16399
Owner	postgres
ACL	
Tablespace	pg_default
Default tablespace	pg_default
Encoding	UTF8

```sql
-- Database: demo

-- DROP DATABASE demo;

CREATE DATABASE demo
  WITH OWNER = postgres
       ENCODING = 'UTF8'
       TABLESPACE = pg_default
       LC_COLLATE = 'English_United States.1252'
       LC_CTYPE = 'English_United States.1252'
       CONNECTION LIMIT = -1;
```

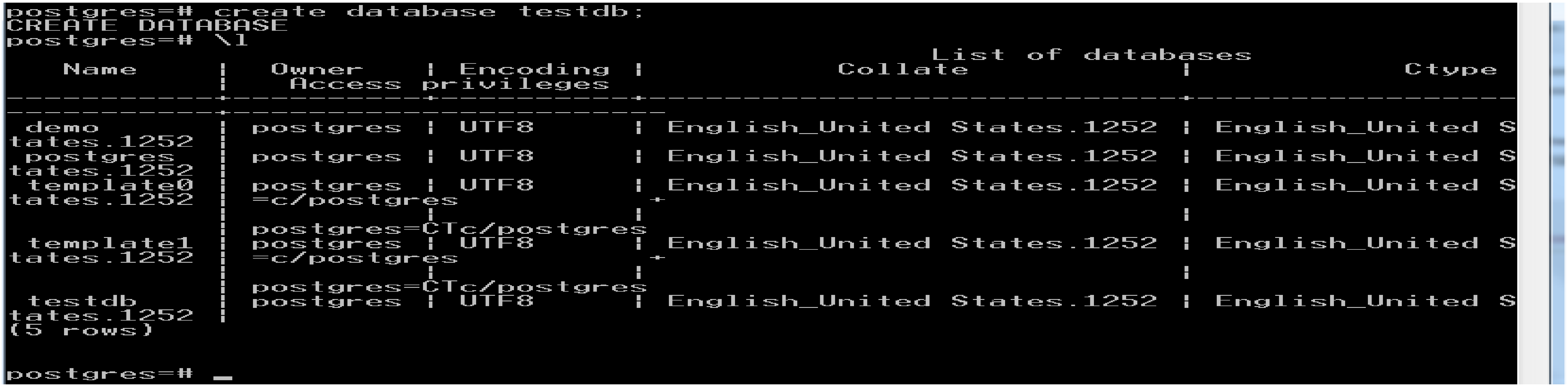
Retrieving details on database demo... Done. | demo on postgres@localhost:5432 | 397 msec

Create Database

- To create a database from PostgreSQL shell prompt:

```
postgres=# create database testdb;
To view all databases existing:
postgres=# \l
```

- By default, the new database will be created by cloning the standard system database *template1*.



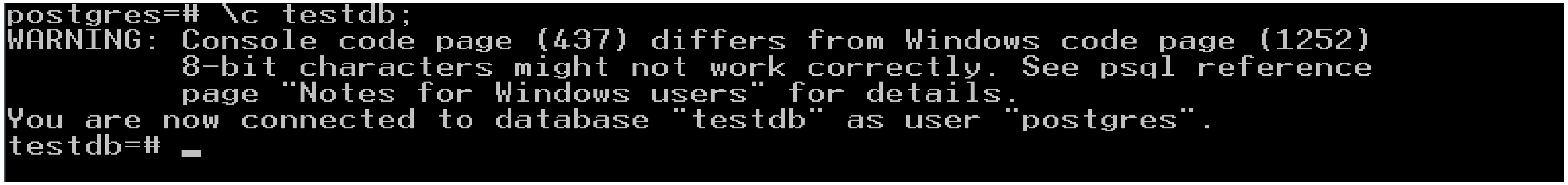
```
postgres=# create database testdb;
CREATE DATABASE
postgres=# \l
                                                        List of databases
    Name      |  Owner   | Encoding |              Collate               |           Ctype
              |   Access privileges
--------------+----------+----------+------------------------------------+------------------
--------------+-----------------------
 demo         | postgres | UTF8     | English_United States.1252 | English_United S
tates.1252    |
 postgres     | postgres | UTF8     | English_United States.1252 | English_United S
tates.1252    |
 template0    | postgres | UTF8     | English_United States.1252 | English_United S
tates.1252    | =c/postgres          +
              |          |          |                            |
              | postgres=CTc/postgres
 template1    | postgres | UTF8     | English_United States.1252 | English_United S
tates.1252    | =c/postgres          +
              |          |          |                            |
              | postgres=CTc/postgres
 testdb       | postgres | UTF8     | English_United States.1252 | English_United S
tates.1252    |
(5 rows)

postgres=# _
```

Create Database

- To connect to testdb database :

```
postgres=# \c testdb;
```



```
postgres=# \c testdb;
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.
You are now connected to database "testdb" as user "postgres".
testdb=# _
```

- To drop a database, we can use drop

```
postgres=# drop database testdb;
```

- It removes the catalog entries for the database and deletes the directory containing the data
- This command cannot be executed while you or anyone else is connected to the target database

Postgres datatypes

- Data types : help in specifying the type of data to be stored in the table columns
- It also provides some benefits:
  - Consistency – operations against columns with same datatype are consistent and fast
  - Validation – Proper use of data types implies format validation of data and rejection of data outside
  - Compactness – A column can store a single type of value, in a compact way
  - Performance – Proper use of data types gives the most efficient storage of data for quick processing

Numeric

Name	Storage Size	Description	Range
smallint	2 bytes	small-range integer	-32768 to +32767
integer	4 bytes	typical choice for integer	-2147483648 to +2147483647
bigint	8 bytes	large-range integer	-9223372036854775808 to 9223372036854775807
decimal	variable	user-specified precision, exact	up to 131072 digits before the decimal point; up to 16383 digits after the decimal point
numeric	variable	user-specified precision, exact	up to 131072 digits before the decimal point; up to 16383 digits after the decimal point
real	4 bytes	variable-precision	6 decimal digits precision
double precision	8 bytes	variable-precision	15 decimal digits precision
smallserial	2 bytes	small autoincrementing integer	1 to 32767
serial	4 bytes	autoincrementing integer	1 to 2147483647
bigserial	8 bytes	large autoincrementing integer	1 to 9223372036854775807

Monetary, Character and Binary

Name	Storage Size	Description	Range
money	8 bytes	currency amount	-92233720368547758.08 to +92233720368547758.07

Character Types

Name	Description
character varying(n), varchar(n)	variable-length with limit
character(n), char(n)	fixed-length, blank padded
text	variable unlimited length

Binary Data Types

Name	Storage Size	Description
bytea	1 or 4 bytes plus the actual binary string	variable-length binary string

Date/Time, Boolean

Name	Storage Size	Description	Low Value	High Value
timestamp [(p)] [without time zone]	8 bytes	both date and time (no time zone)	4713 BC	294276 AD
timestamp [(p)] with time zone	8 bytes	both date and time, with time zone	4713 BC	294276 AD
date	4 bytes	date (no time of day)	4713 BC	5874897 AD
time [(p)] [without time zone]	8 bytes	time of day (no date)	00:00:00	24:00:00
time [(p)] with time zone	12 bytes	times of day only, with time zone	00:00:00+1459	24:00:00-1459
interval [fields] [(p)]	12 bytes	time interval	-178000000 years	178000000 years

Name	Storage Size	Description
boolean	1 byte	state of true or false

Date/Time

- Datatypes
  - Date - Date only (2012-04-25)
  - Time - Time only (13:00:00.00)
  - Timestamp - Date and Time (2012-04-25 13:00:00.00)
  - Time with Timezone - Time only (13:00:00.00 PST)
  - Timestamp with Timezone (2012-04-25 13:00:00.00 PST)
  - Interval - A span of time (4 days)
- Note: Interval, is a great utility for when you : need to query against some range of specific time

Date/Time examples

- To get todays date use current_date

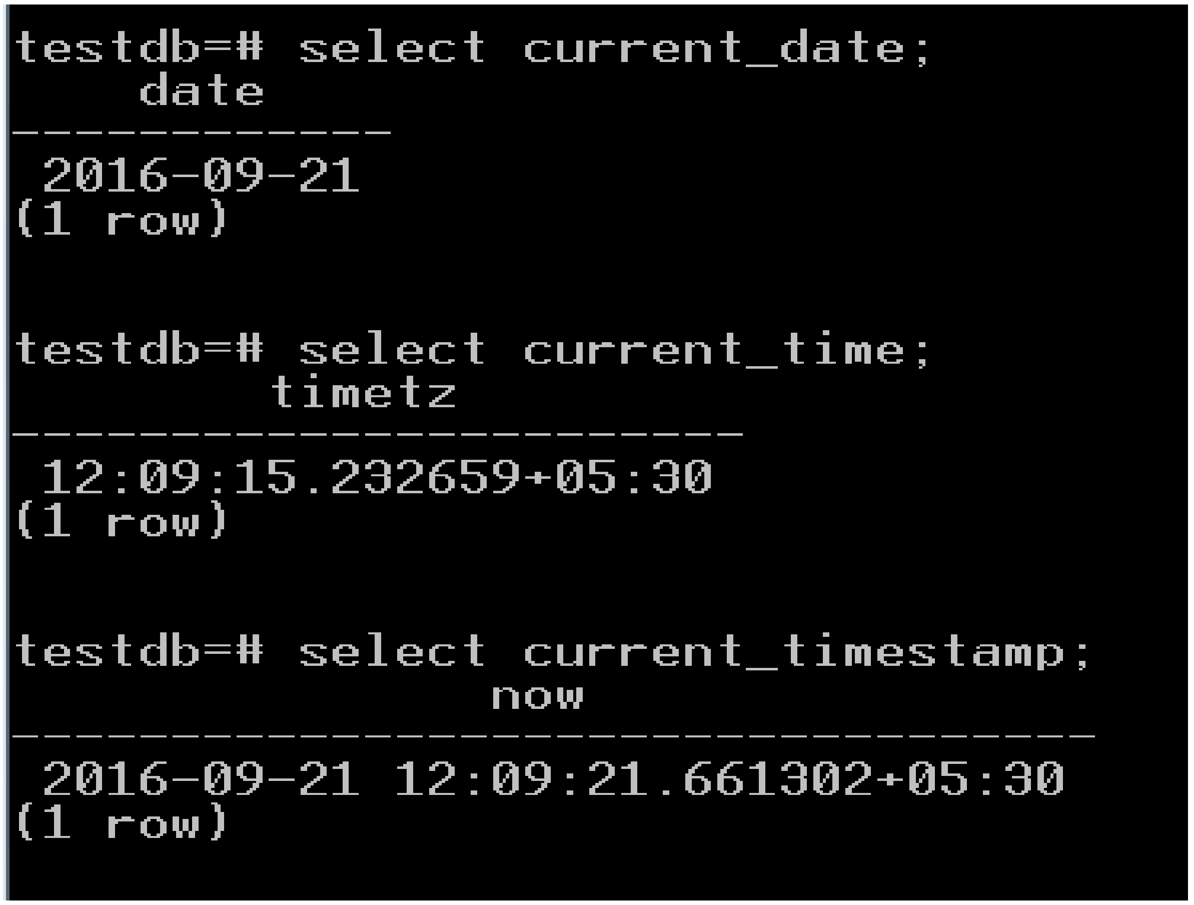
```
select current_date;        //output is in format yyyy-mm-dd -> eg:    2016-09-21
```

- To get time use current_time

```
select current_time;
//output is with timezone -> eg:    12:08:33.871234+05:30
```

- To get date and time use current_timestamp

```
select current_timestamp;
//output -> eg: 2016-09-21 12:08:33.871234+05:30
```



```
testdb=# select current_date;
    date
------------
 2016-09-21
(1 row)


testdb=# select current_time;
        timetz
-----------------------
 12:09:15.232659+05:30
(1 row)


testdb=# select current_timestamp;
              now
----------------------------------
 2016-09-21 12:09:21.661302+05:30
(1 row)
```

1.3: Creating Tables

Creating Tables

- CREATE TABLE is the keyword telling the database system to create a new table
- Table should have a unique name or identifier for the table
- Initially table will be empty in the current database and will be owned by the user issuing the command

```
create table employee(
empid int primary key not null,
name text not null,
age int not null,
salary real
);
```

```
create table department(
deptid int primary key not null,
dname char(50) not null,
empid int not null
);
```

- To display all tables in your database use \d command
- To describe each table use \d tablename
- To remove a table use command “drop table table name”

Creating Schema

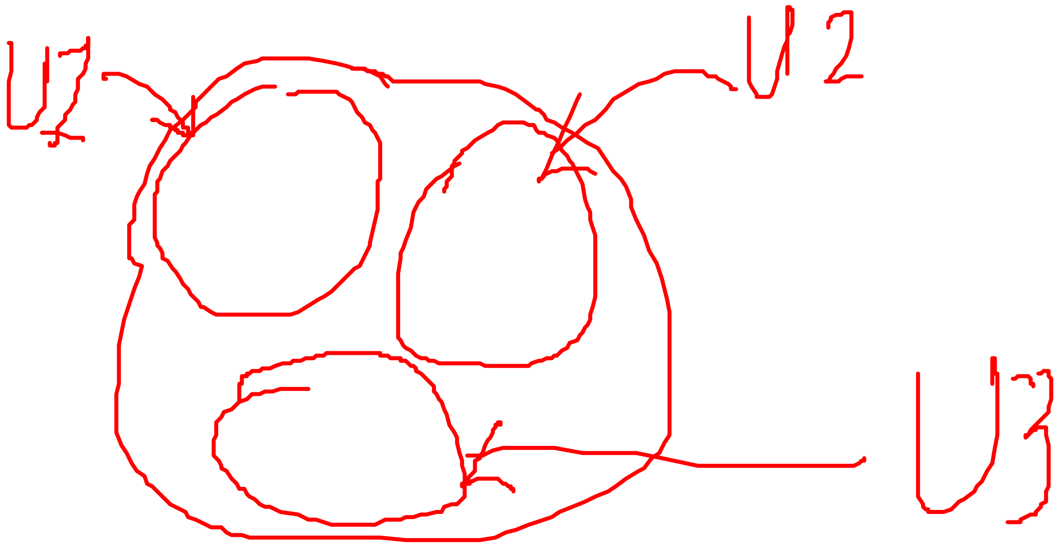
- Schema is a named collection of tables
- Schema can also contain views, indexes, sequences, data types, operators and functions
- Schemas are like directories, but they cannot be nested
- To create a schema:

```
create schema myschema;
```

- To create a table in schema:

```
create table myschema.mytable(
);
```

- Database objects can be grouped logically so that they are manageable



1.3: Insert data into table

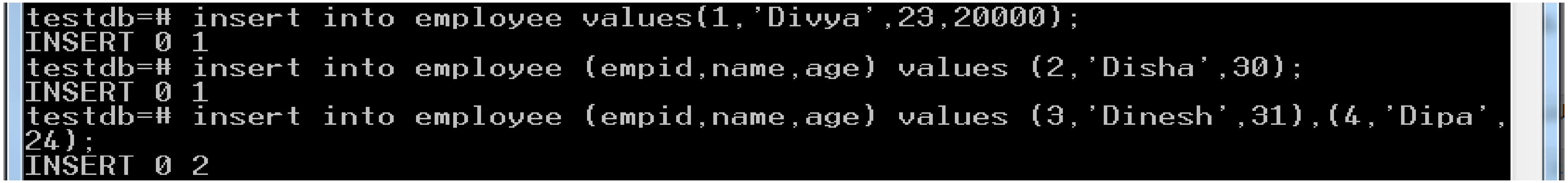
Inserting data into table

- “insert into” statement allows you to insert a row into the table

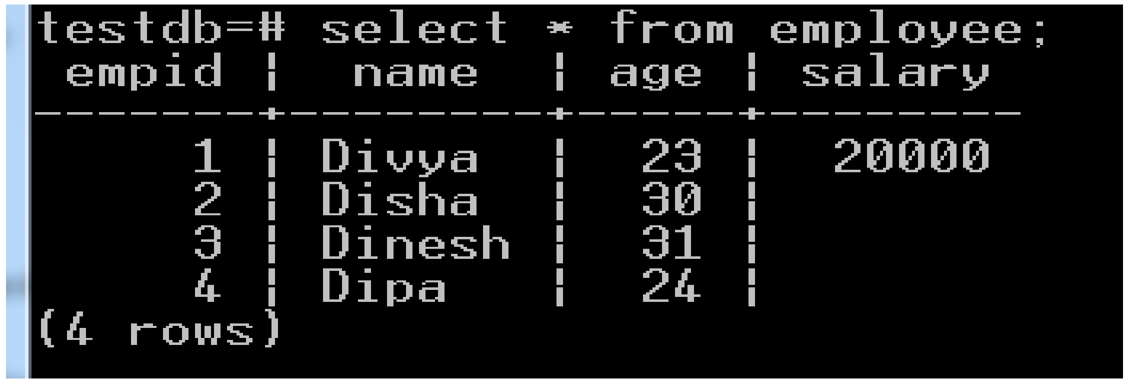
```
INSERT INTO TABLE_NAME VALUES (value1,value2,value3,...valueN);
```

Example:

- Insert without column list
- Insert with column list
- Insert multirows



```
testdb=# insert into employee values(1,'Divya',23,20000);
INSERT 0 1
testdb=# insert into employee (empid,name,age) values (2,'Disha',30);
INSERT 0 1
testdb=# insert into employee (empid,name,age) values (3,'Dinesh',31),(4,'Dipa',
24);
INSERT 0 2
```



```
testdb=# select * from employee;
 empid ¦  name   ¦ age ¦ salary
-------+---------+-----+--------
     1 ¦ Divya   ¦  23 ¦  20000
     2 ¦ Disha   ¦  30 ¦
     3 ¦ Dinesh  ¦  31 ¦
     4 ¦ Dipa    ¦  24 ¦
(4 rows)
```

Updating data into table

- “update” statement allows you to update one or more rows in the table
- This is used for modifying records in the table:

```
update table_name
set column1=value1, column2=value2,..
where condition;
```

- Example: To modify salary for empid =1

```
update employee
set salary = 22000
where empid=1;
```

Deleting data from table

- Delete is used to delete existing records from a table
- Use WHERE clause to restrict deletion to specific rows

```
DELETE TABLE_NAME WHERE condition;
```

- Example: To delete record for empid = 10

```
DELETE employee WHERE empid=10;
```

Demo

- Create database
- Create table
- Create schema
- Insert rows

Lab

Lab 1

Summary

In this lesson, you have learn about:

- PostgreSQL is a general purpose database management system
- It uses client server model
- We can create a database in PostgreSQL

Review Question

Question 1: New database is created by cloning standard database _____________.

Question 2: Which of the following datatype can be used to auto increment values in primary key column?

- Integer
- Numeric
- Real
- Serial

People matter, results count.

About Capgemini

A global leader in consulting, technology services and digital transformation, Capgemini is at the forefront of innovation to address the entire breadth of clients' opportunities in the evolving world of cloud, digital and platforms. Building on its strong 50-year heritage and deep industry-specific expertise, Capgemini enables organizations to realize their business ambitions through an array of services from strategy to operations. Capgemini is driven by the conviction that the business value of technology comes from and through people. It is a multicultural company of 200,000 team members in over 40 countries. The Group reported 2016 global revenues of EUR 12.5 billion.

Visit us at

www.capgemini.com

This message is intended only for the person to whom it is addressed. If you are not the intended recipient, you are not authorized to read, print, retain, copy, disseminate, distribute, or use this message or any part thereof. If you receive this message in error, please notify the sender immediately and delete all copies of this message.