My partner told me they would leave me if I don't stop making Microsoft puns, and I need some advice

I immediately left my Office and tried explaining myself.

Sure, on the Surface I do it often, but I think it Works.

It's not just about Word play, either; my Outlook on life helps me Excel.

We have such a great Team Foundation, I Azure you.

I wanted to Exchange my thoughts with them, so we could work with OneDrive.

I looked at my partner right in the Windows of their soul, to Access the deepest parts of their heart, and told them I loved them.

Completely on Edge, I awaited their answer…

PowerPoint of the story is: does anyone know of a good divorce lawyer?

Module 2-1

Introduction to Databases and SQL

Objectives

- Understand what a database is
- Understand what SQL is
- Basic proficiency with tools specific to PostgreSQL
- Be able to write simple queries that retrieve data

He's making a database.
He's sorting it twice.
SELECT * from contacts WHERE
behavior = 'nice'
SQL Clause is coming to town!

Databases

- A database is an electronically stored organized collection of data.
- A **relational database** is one in which the data is organized around columns and tables:
  - A table is designed to store an **entity**, a data representation of a real world object.
  - Each row of a table represents one instance of the entity.
  - The columns represent attributes the entity might have.

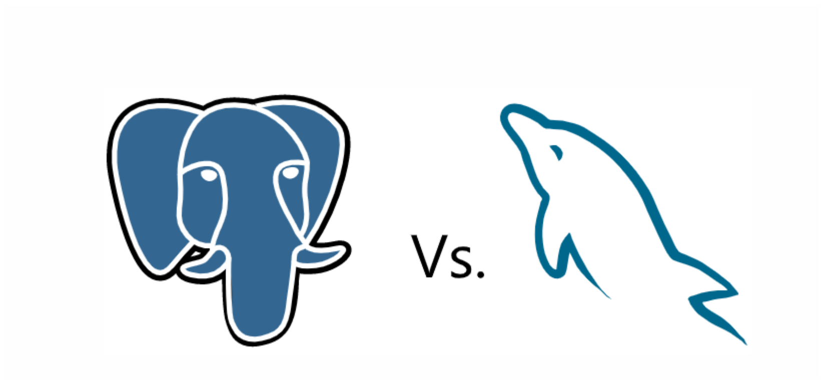
Databases

- Relational (SQL Server, Oracle, MySQL, PostgreSQL)
- NoSQL (MongoDB, CouchDB)
- Cloud (Microsoft Azure, Amazon Relational DB Service)
- Columnar (Google BigQuery, MariaDB, Azure SQL Data Warehouse)
- Wide Column (BigTable, Apache Cassandra)
- Object-oriented (Wakanda, ObjectStore)
- Key-value (Amazon DynamoDB, Redis)
- Hierarchical (IMS, Windows Registry)

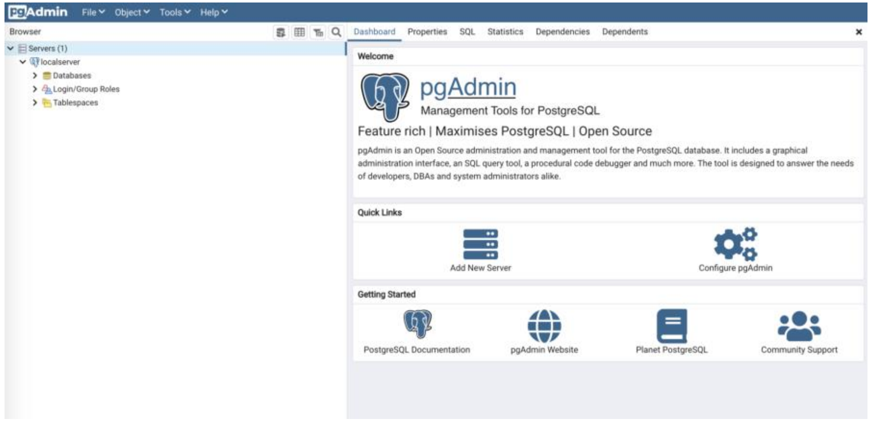
and more!

https://www.matillion.com/resources/blog/the-types-of-databases-with-examples

Database Mangement System (DBMS)



RDBMS



Welcome

pgAdmin

Management Tools for PostgreSQL

Feature rich | Maximises PostgreSQL | Open Source

pgAdmin is an Open Source administration and management tool for the PostgreSQL database. It includes a graphical administration interface, an SQL query tool, a procedural code debugger and much more. The tool is designed to answer the needs of developers, DBAs and system administrators alike.

Quick Links

Add New Server

Configure pgAdmin

Getting Started

PostgreSQL Documentation

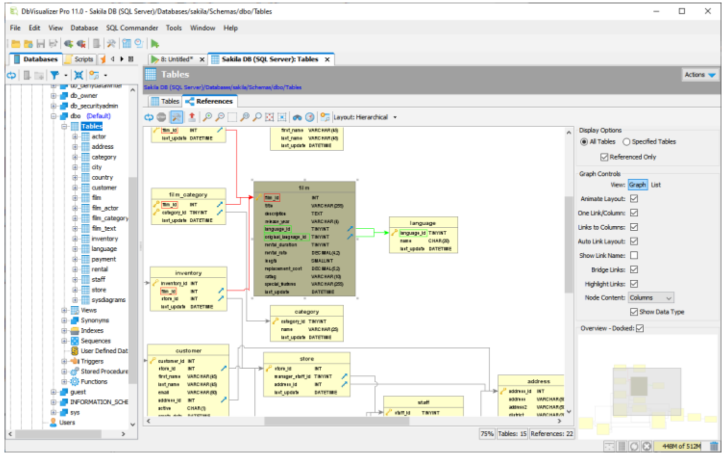
pgAdmin Website

Planet PostgreSQL

Community Support

DbVisualizer
The Universal Database Tool

eSoftner



```
winpty createdb -U postgres UnitedStates
```

winpty – so we can use a bash window instead of cmd

createdb – create the db in psql (PostgreSQL)

-U – for user

postgres

Database named UnitedStates

Let’s get setup!

Relational Database: Example

Suppose we are interested in storing data about cars. We can model a car entity into its own table:

This table has 4 attributes: CarName, Manufacturer, NumberOfDoors, FuelEconomy

This table has 3 rows.

CarName	Manufacturer	NumberOfDoors	FuelEconomy
Explorer	Ford	4	23
C-Class	Mercedes Benz	4	28
Jeep Wrangler	Fiat Chrysler	2	20

Relational Database: Attribute Data Types

There is a large variety of data types in Postgresql, to name a few:

- **varchar**: holds text containing letters and numbers (somewhat like a String in Java).
- **char**: fixed length field containing letters and numbers.
- Various numeric data types:
- When referring to a non-numeric “text” field (i.e. varchar or char) we must surround them in single quotes (i.e. country=**’USA’**).
- Numeric literals do not need single quotes (numberOfDoors = **4**).

https://www.postgresql.org/docs/9.3/datatype.html

Relational Database: SQL

- **SQL** is an acronym for **Structured Query Language**
- SQL is the language used to interact with relational database management systems.
- The exact implementation of SQL varies slightly depending on the database system involved, i.e. there will be minor differences in the language between PostgreSQL and MS SQL Server.
- This class will be using PostgreSQL.

3 types of commands

- DML
  - Database Manipulation Language
    - INSERT, SELECT, DELETE, etc.
- DDL
  - Data Definition Language
    - Commands for creating tables, defining relationships, etc.
- DCL
  - Data Control Language
    - Commands that control permissions on the data and access rights

Postgres!

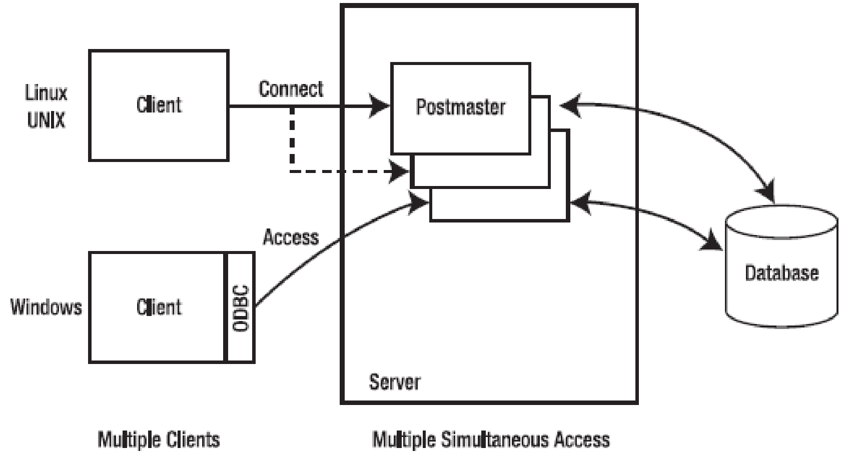


Figure 1-3. *PostgreSQL architecture*

SQL: SELECT

- The most basic SQL statement is a SELECT query, and it follows the following format:

SELECT **[column]**, **[column-n]** FROM **[table]**;

- **[column]** and **[column-n]** are stand ins for the attributes or columns that you want returned from your query.
- **[table]** refers to the name of the table you are querying.
- You can create column Aliases using the “**AS**” keyword followed by the alias.

SQL: SELECT Example

Let’s take the Vehicle table we just saw as an example:

- We could write the following SELECT statement:

  SELECT CarName, NumberOfDoors AS doors FROM Vehicle;

  The output of this would be:

CarName	doors
Explorer	4
C-Class	4
Jeep Wrangler	2

Note how the alias affects the column name in the output.

- Instead of listing specific columns we could use the wildcard * to indicate that all columns should be returned: *SELECT * FROM Vehicle;*

SQL: SELECT with WHERE clause

- We can include a WHERE clause in our select statements to limit the data returned by specifying a condition.
- The WHERE statement relies on comparison operators.
  - **Greater Than**: **>**
  - **Greater Than or Equal To**: **>=**
  - **Less Than**: **<**
  - **Less Than or Equal To**: **<=**
  - **Equal**: **=**
  - **Not Equal To**: **<> !=**
- There is a special comparison operator called **LIKE** which is often used in conjunction with a wildcard (**%**) operator.

SQL: SELECT with WHERE clause Example 1

Let’s take the Vehicle table we just saw as an example:

- We could write the following SELECT statement:

  SELECT * FROM Vehicle WHERE Manufacturer = ‘Ford’;

- Only 1 row matches this criteria, and thus the results of the query will be:

CarName	Manufacturer	NumberOfDoors	FuelEconomy
Explorer	Ford	4	23

SQL: SELECT with WHERE clause Example 2

Here is an example of the WHERE clause using the LIKE / Wildcard.

- We could write the following SELECT statement:

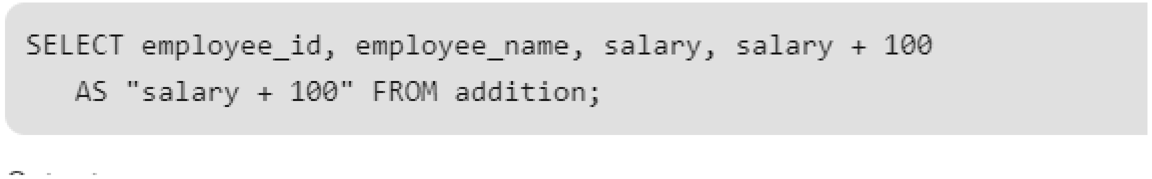
  SELECT * FROM Vehicle WHERE CarName LIKE ‘Ex%’;

- Only 1 row matches this criteria, and thus the results of the query will be:

CarName	Manufacturer	NumberOfDoors	FuelEconomy
Explorer	Ford	4	23

Derived Columns with Math Operations

- A custom field containing math operations can be included in the SELECT.
- The basic math operators are present: **+**, **-**, *****, **/**, **%**



```
SELECT employee_id, employee_name, salary, salary + 100
   AS "salary + 100" FROM addition;
```

Derived Columns Example

- Consider the following example:

SELECT CarName, FuelEconomy * 0.425144 AS kpl FROM Vehicle;

CarName	kpl
Explorer	9.778312
C-Class	9.778312
Jeep Wrangler	8.50288

SQL: AND / OR on WHERE statements

- Within the WHERE statement, various filter conditions can be combined using the AND / OR statement.
- Consider the following example:
  SELECT * FROM Vehicle WHERE Manufacturer = ‘Ford’ OR NumberOfDoors = 4;
- Two rows are returned:

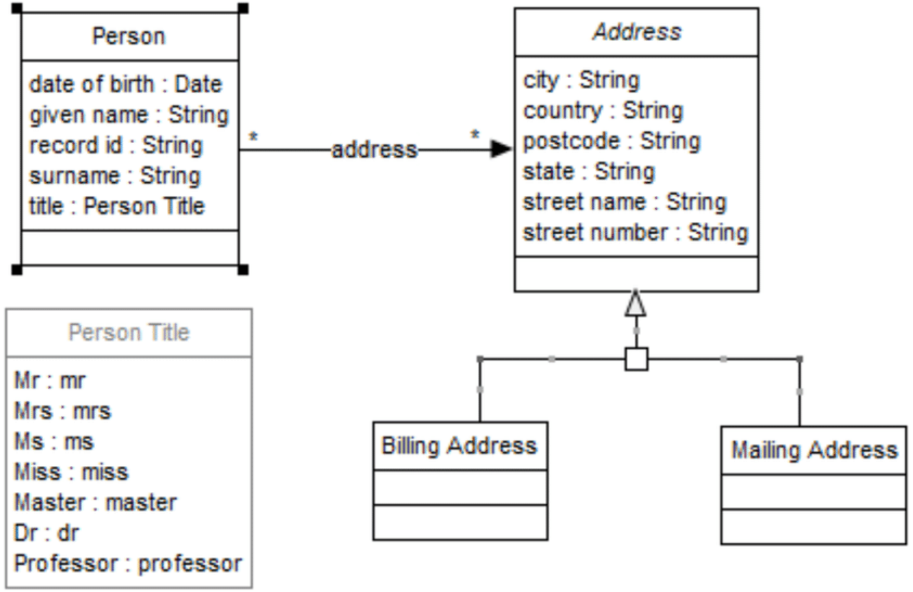
CarName	Manufacturer	NumberOfDoors	FuelEconomy
Explorer	Ford	4	23
C-Class	Mercedes Benz	4	28

Let’s write some SELECT statements!

But first…

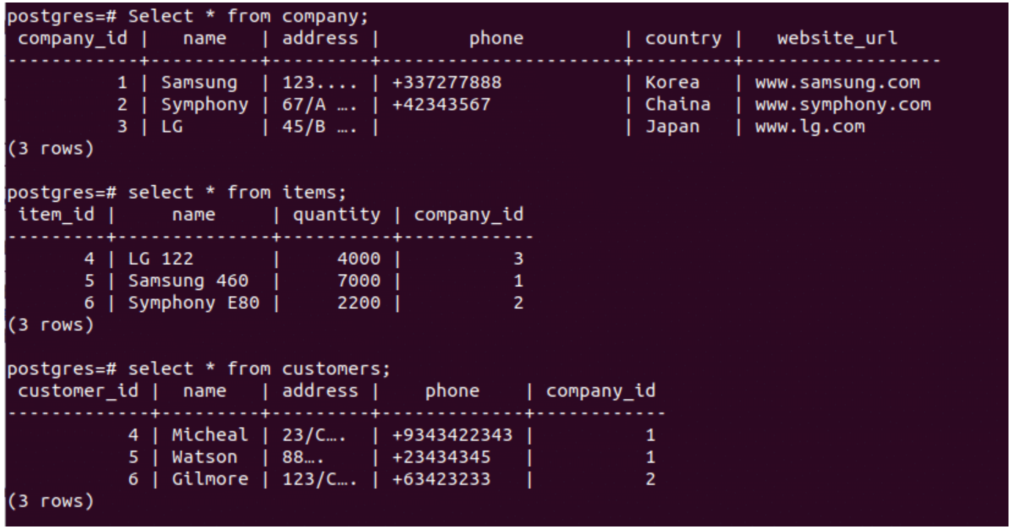
Objectives

- Understand what a database is



Objectives

- Understand what a database is
- Understand what SQL is



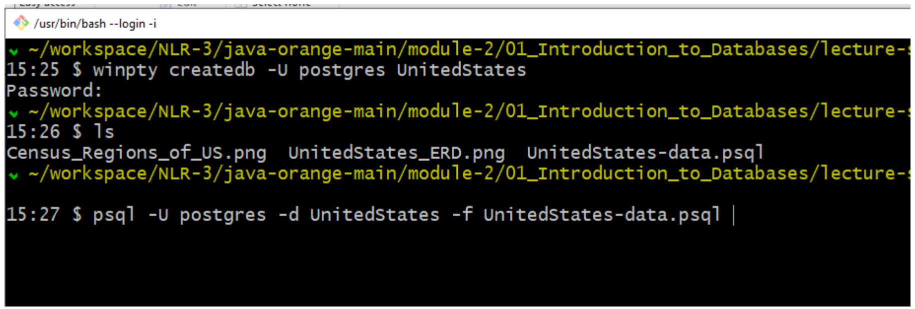
```
postgres=# Select * from company;
 company_id |   name   | address |        phone         | country |   website_url
------------+----------+---------+----------------------+---------+------------------
          1 | Samsung  | 123.... | +337277888           | Korea   | www.samsung.com
          2 | Symphony | 67/A …. | +42343567            | Chaina  | www.symphony.com
          3 | LG       | 45/B …. |                      | Japan   | www.lg.com
(3 rows)

postgres=# select * from items;
 item_id |     name     | quantity | company_id
---------+--------------+----------+------------
       4 | LG 122       |     4000 |          3
       5 | Samsung 460  |     7000 |          1
       6 | Symphony E80 |     2200 |          2
(3 rows)

postgres=# select * from customers;
 customer_id |  name   | address |    phone    | company_id
-------------+---------+---------+-------------+------------
           4 | Micheal | 23/C….  | +9343422343 |          1
           5 | Watson  | 88….    | +23434345   |          1
           6 | Gilmore | 123/C…. | +63423233   |          2
(3 rows)
```

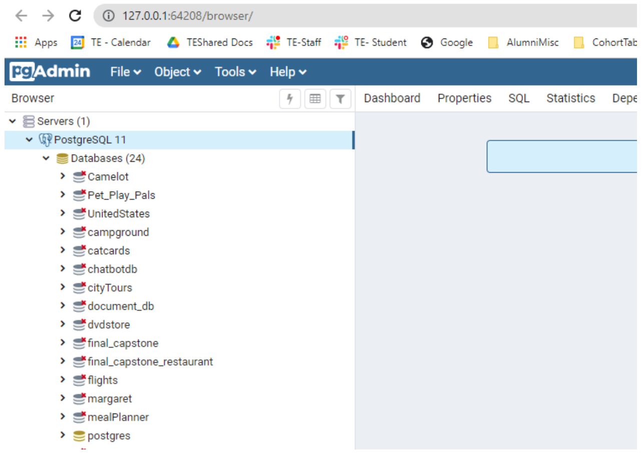
Objectives

- Understand what a database is
- Understand what SQL is
- Basic proficiency with tools specific to PostgreSQL

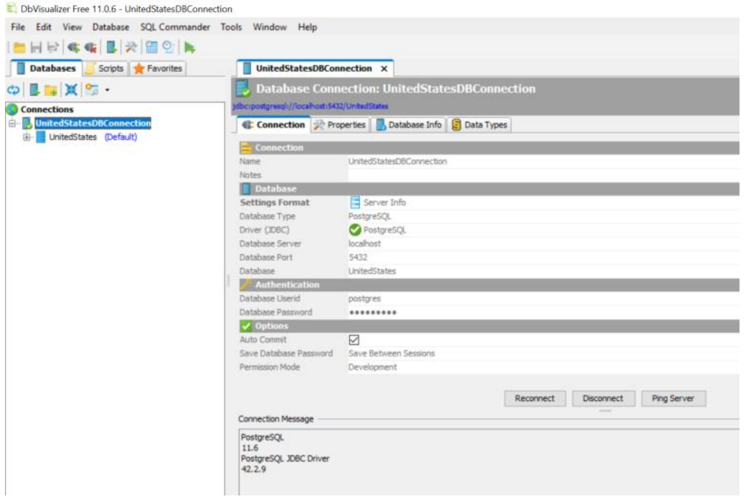


```
/usr/bin/bash --login -i
~/workspace/NLR-3/java-orange-main/module-2/01_Introduction_to_Databases/lecture-
15:25 $ winpty createdb -U postgres UnitedStates
Password:
~/workspace/NLR-3/java-orange-main/module-2/01_Introduction_to_Databases/lecture-
15:26 $ ls
Census_Regions_of_US.png  UnitedStates_ERD.png  UnitedStates-data.psql
~/workspace/NLR-3/java-orange-main/module-2/01_Introduction_to_Databases/lecture-

15:27 $ psql -U postgres -d UnitedStates -f UnitedStates-data.psql
```



Servers (1)
PostgreSQL 11
Databases (24)
Camelot
Pet_Play_Pals
UnitedStates
campground
catcards
chatbotdb
cityTours
document_db
dvdstore
final_capstone
final_capstone_restaurant
flights
margaret
mealPlanner
postgres



Database Connection: UnitedStatesDBConnection
jdbc:postgresql://localhost:5432/UnitedStates

Connection
Name: UnitedStatesDBConnection
Notes:

Database
Settings Format: Server Info
Database Type: PostgreSQL
Driver (JDBC): PostgreSQL
Database Server: localhost
Database Port: 5432
Database: UnitedStates

Authentication
Database Userid: postgres
Database Password: •••••••••

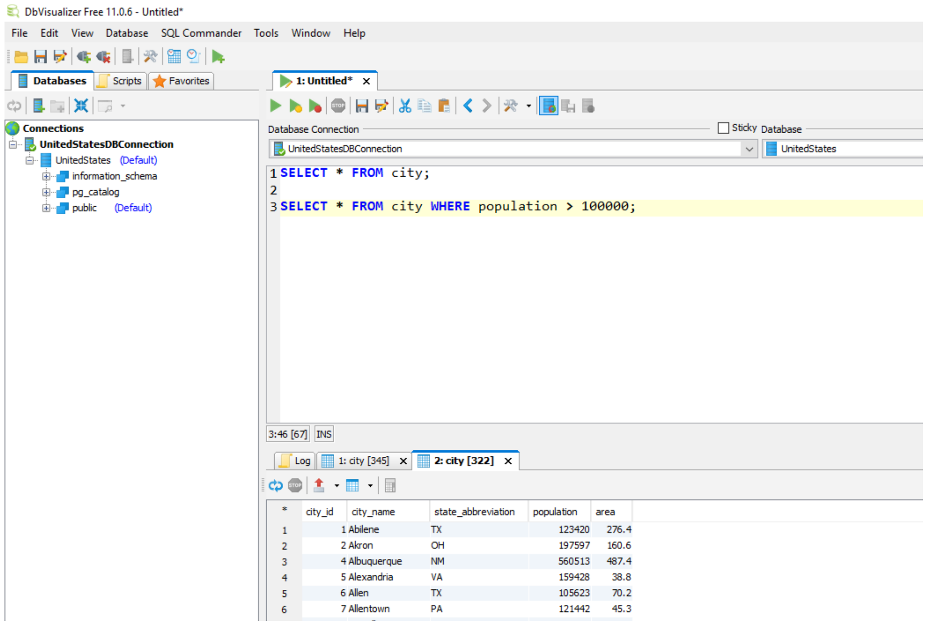
Options
Auto Commit: ☑
Save Database Password: Save Between Sessions
Permission Mode: Development

Connection Message

```
PostgreSQL
11.6
PostgreSQL JDBC Driver
42.2.9
```

Objectives

- Understand what a database is
- Understand what SQL is
- Basic proficiency with tools specific to PostgreSQL
- Be able to write simple queries that retrieve data



```
SELECT * FROM city;

SELECT * FROM city WHERE population > 100000;
```

*	city_id	city_name	state_abbreviation	population	area
1	1	Abilene	TX	123420	276.4
2	2	Akron	OH	197597	160.6
3	4	Albuquerque	NM	560513	487.4
4	5	Alexandria	VA	159428	38.8
5	6	Allen	TX	105623	70.2
6	7	Allentown	PA	121442	45.3