Intro To SQL

Devpoint Labs - Jake Sorce / Dave Jungst

What is SQL and what does it do?

- SQL or Structured Query Language is the programming language we use to talk to databases with

- What is a Database?
  - Databases collect and organize data to allow for easy retrieval

Types of SQL Databases

There are many popular SQL databases including SQLite, MySQL, Postgres, Oracle and Microsoft SQL Server. All of them support the common SQL language standard. Each implementation can differ in the additional features and storage types it supports.

The SQL Database we will use

- We will be using postgreSQL
  - http://www.postgresql.org

- Reasons why we are using postgreSQL
  - Familiar tools: pgadmin3
    - http://www.pgadmin.org
  - Plays nice with Rails
  - Plays nice with Heroku (Our hosting platform)

How do I interact with my postgreSQL Database?

- Through postgres.app
  - When the postgres.app is running you should have a elephant at the top of your screen. Click it and click on the button that says psql

- Through pgadmin3
  - http://www.pgadmin.org
  - Preferred and easiest way for beginners

Using Pgadmin3 - Configuration / Setup

- Open the program
  - Click on the plug icon in the top left to create a new database server profile
  - Set the name to whatever you want, I'll call mine 'Local Postgres Server'
  - Set the host to 'localhost'
  - Set the port to '5432'
  - Set the username to 'your computer's username'
  - Click 'Ok' to save that connection profile
  - Open the postgress app terminal by clicking the elephant and clicking on open psql
  - type CREATE DATABASE testdb;

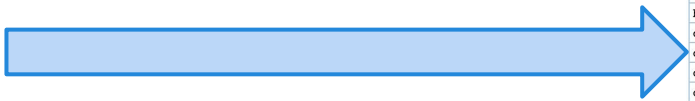
Using Pgadmin3

Once connected to your database server

- Click to expand:
  - Server Groups
  - Servers
  - Local Postgres
  - Databases
  - Find your testdb database in that list and click on it
  - Click on the magnifying glass with the words SQL inside of it to start making SQL queries

PostgreSQL Database Data Types

All data types available



Most common data types:

- integer
- varchar
- boolean
- date
- text
- time
- timestamp

Name	Aliases	Description
bigint	int8	signed eight-byte integer
bigserial	serial8	autoincrementing eight-byte integer
bit [(*n*)]		fixed-length bit string
bit varying [(*n*)]	varbit	variable-length bit string
boolean	bool	logical Boolean (true/false)
box		rectangular box on a plane
bytea		binary data ("byte array")
character [(*n*)]	char [(*n*)]	fixed-length character string
character varying [(*n*)]	varchar [(*n*)]	variable-length character string
cidr		IPv4 or IPv6 network address
circle		circle on a plane
date		calendar date (year, month, day)
double precision	float8	double precision floating-point number (8 bytes)
inet		IPv4 or IPv6 host address
integer	int, int4	signed four-byte integer
interval [*fields*] [(*p*)]		time span
json		textual JSON data
jsonb		binary JSON data, decomposed
line		infinite line on a plane
lseg		line segment on a plane
macaddr		MAC (Media Access Control) address
money		currency amount
numeric [(*p*, *s*)]	decimal [(*p*, *s*)]	exact numeric of selectable precision
path		geometric path on a plane
pg_lsn		PostgreSQL Log Sequence Number
point		geometric point on a plane
polygon		closed geometric path on a plane
real	float4	single precision floating-point number (4 bytes)
smallint	int2	signed two-byte integer
smallserial	serial2	autoincrementing two-byte integer
serial	serial4	autoincrementing four-byte integer
text		variable-length character string
time [(*p*)] [without time zone]		time of day (no time zone)
time [(*p*)] with time zone	timetz	time of day, including time zone
timestamp [(*p*)] [without time zone]		date and time (no time zone)
timestamp [(*p*)] with time zone	timestamptz	date and time, including time zone
tsquery		text search query
tsvector		text search document
txid_snapshot		user-level transaction ID snapshot
uuid		universally unique identifier
xml		XML data

Creating our first Table

```
CREATE TABLE users (
    id          integer PRIMARY KEY,
    first_name  varchar(40) NOT NULL,
    last_name   varchar(40) NOT NULL,
    age         integer,
    gender      varchar(15) NOT NULL
);
```

new table with the name of users

id as integer and has a primary key constraint

first_name as a varying character

last_name as a varying character up to 40 characters

age as an integer

gender as a varying character up to 15 characters

Basic Database terminology

- Database
  - The term when describing the whole database server
- Table
  - A container that is stored inside of the database server
  - A Table can have many rows
- Row
  - The actual data that is stored in the table
  - Often referred to as a record as well
- Column
  - Is a value that is stored in a Row

Basic Database terminology

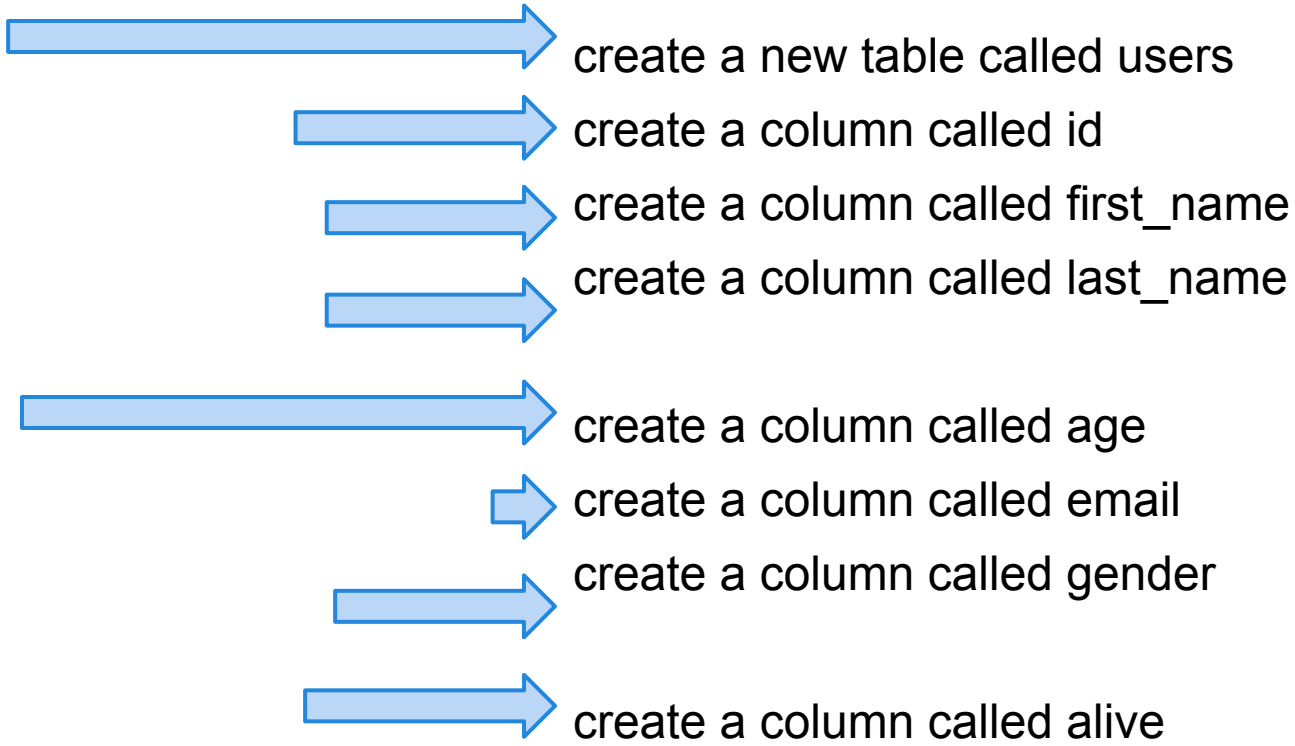
Some of The Most Important SQL Commands

- SELECT - extracts data from a database
- UPDATE - updates data in a database
- DELETE - deletes data from a database
- INSERT INTO - inserts new data into a database
- CREATE DATABASE - creates a new database
- ALTER DATABASE - modifies a database
- CREATE TABLE - creates a new table
- ALTER TABLE - modifies a table
- DROP TABLE - deletes a table
- CREATE INDEX - creates an index (search key)
- DROP INDEX - deletes an index

Basic Database Table terminology

- PRIMARY KEY
  - All database tables should have a primary key, this should be a unique key that is used as a safe way to look up a row in your database
- NOT NULL
  - Means that specific value in the row needs to be there or the database won't let you save that record
- UNIQUE
  - Means that the value of data in that row cannot be in any other row
- DEFAULT
  - This gives a default value to the column of a row

Database terminology examples



Inserting data into a Table

```
INSERT INTO users (first_name, last_name, age, email, gender, alive)
    VALUES ('Jake', 'Sorce', 25, 'jake@devpointlabs.com', 'male', true);
```

This will insert a new row or record into our database with the values specified.

Inserting data into a Table (Defaults)

```
INSERT INTO users (first_name, last_name, age, email, gender, alive)
    VALUES ('Jake', 'Sorce', 25, 'jakesorce@gmail.com', 'male', DEFAULT);
```

This will insert a new row or record into our database with the values specified using the default value with the DEFAULT syntax.

Inserting data into a Table (Multiples)

```
INSERT INTO users (first_name, last_name, age, email, gender, alive) VALUES
('Dave', 'Jungst', 31, 'dave@tracktrain.com', 'male', DEFAULT),
('Chris', 'Memmott', 32, 'chris@tracktrain.com', 'male', DEFAULT),
('Clive', 'Savacool', 40, 'clive@tracktrain.com', 'male', DEFAULT),
('Rosie', 'Thomas', 29, 'rosiethomas@tracktrain.com', 'female', DEFAULT);
```

This will insert 4 new records into the users table all with different values.

Common basic query syntax

- SELECT - extracts data from the database
- LIMIT - only return the limit that is specified
- WHERE - condition clause
- ORDER - orders by the column/s specified
- GROUP BY - groups by the column/s specified
- DISTINCT - only return unique rows
- COUNT - return the count of the rows that would come back

Selecting data from a Table

This will return all rows with all values from the users table. *(warning: this can be very slow and could bring your database to its knees if you have enough data)*

Selecting data from a Table (specific values)

```
select first_name, last_name from users;
```

This will only return the first and last names of all the records in the users table. Since this is selecting only certain data, this is considered better than a select *.

Selecting data from a Table (using limit)

```
select first_name, last_name from users limit(2);
```

This will only return the first 2 rows selected.

Filtering selected data from a Table (using where)

```
select first_name, last_name from users where users.first_name = 'Jake';
```

This will only return rows that have the first_name equal to ‘Jake’.

Ordering selected data from a Table (using order)

```
select first_name, last_name, age from users where users.first_name = 'Jake' order
by(users.age);
```

```
select first_name, last_name, age from users where users.first_name = 'Jake' order
by(users.age) desc;
```

This will only return rows that have the first_name equal to ‘Jake’ and the data will be ordered first to last by age. By default the order is done ascending.

Grouping selected data from a Table (using group by)

```
select first_name, last_name, age from users group by first_name, last_name, age;
```

This returns the columns first_name, last_name, and age from all rows in the user table. These rows will be grouped together based on the group specified

Selecting only unique data from a Table (using distinct)

```
select distinct on(first_name) first_name, last_name, age from users;
```

If there are 2 rows that contain the same name, only 1 of these rows will be returned back in the result of the query.

Counting selected data from a Table (using count)

```
select count(*) AS "number of young users"
from users
where users.age < 30;
```

This will return a column named “Number of young users” with the value of how many rows in the database have an age of 30 or less.

Updating data in a Table

```
update users set(first_name, age) = ('Jacob', 30)
where users.id = 1
```

This will update the first_name and age of the user record with the id of 1.

Deleting data from a Table.

```
delete from users
where users.id = 1;


delete from users
where users.first_name = 'Jake';
```

Dropping a Table

Dropping a Database

```
drop database testdb;
```

Databases from PSQL

Open psql command line

connect to database

\connect database_name

list tables

\d

list columns in table

\d table_name

More learning resources

- SQLBolt - Interactive SQL Lessons
  - http://sqlbolt.com
- Code School Try SQL
  - http://campus.codeschool.com/courses/try-sql
- W3 Schools - Great Resource For Many Technology Related Things
  - http://www.w3schools.com/sql