# DATABASES – DATABASES – DATABASES

**See Classwork Week\_5 Day\_1**

CRUD - Create, Read, Update, Delete

## PSQL – PSQL – PSQL – PSQL

Seee week\_5 day\_1 for intro, then link with ruby/webdev on day\_3

Homework w5 d3 videos\_tube

Can create a database from terminal: createdb name

Command line: psql

**\*\*\*\* All PSQL commands are run with a backslash \ \*\*\***

**PSQL is to create / access a database**

**When working on database, it is pure SQL -> no \**

**\*\*\*\* All commands and end with “;” \*\*\*\***

\h for list of commands

\q to quit

Create database

create database people\_app;

Or after building a file on sublime, database: campus: build databse campus based on the .sql file

psql -d campus -f campus.sql

Delete database

drop database people\_app;

List all databases

\l list of databases

Connect to a database

\connect people\_app

\d to get list of relations

***In the sql file, pretty cool to do drop table xyz before create table, so that if you make changes to the table, you just need to re-run the file and the table will be dropped and recreated***

Create table, and create columns for headings

create table people ( // people is name of table name varchar(255), // heading name: type char, max 255 characters age int2, sex char(1) );

Add data into the heading -> create row with ‘bob’ under ‘name’

INSERT INTO PEOPLE (name) VALUES ('bob')

INSERT INTO PEOPLE (age) VALUES (52); *// this is on a different row!*

INSERT INTO PEOPLE (name, age, sex) VALUES ('sue', 52, 'f' );

Add data and return what you just added

"insert into items (item, done) values ('#{item}', 'false') returning \*"

Read data: Select

SELECT \* FROM PEOPLE; // \* means everything

name | age | sex

------+-----+-----

bob | |

| 52 |

sue | 52 | f

(3 rows

Insert data into an existing row

UPDATE people SET sex = 'm' WHERE name = 'bob';

Delete data from a row *// will delete ALL matching rows*

DELETE FROM PEOPLE WHERE age = 52;

Adding IDs as a heading to the table items makes it a lot easier to access data

serial8 means id# will be incremented each time

people\_app=# create table weather ( id serial8, city varchar(255), low int, high int, high\_recorded\_on date, low\_recorded\_on date );

insert into weather (city, low, high, high\_recorded\_on) VALUES ('London', -5, 3, '1 Jan 2015'); (do it 3 times)

id | city | low | high | high\_recorded\_on | low\_recorded\_on

----+--------+-----+------+------------------+-----------------

1 | London | -5 | 3 | 2015-01-01 |

2 | London | -5 | 3 | 2015-01-01 |

3 | London | -5 | 3 | 2015-01-01 |

(3 rows)

Refine lookup

select \* from people where name = ‘bob’;

=> will give you the whole row

select name from people where age = ‘52’

* will return only the name column

select age, sex from people

select first, last from students where id in (2,3) *// “in” like an array*

JOIN is the best

SELECT \* FROM students JOIN classes ON schedules.class\_id = classes.id WHERE classes.name = ''

* get everything from students

## Integrate PSQL with Ruby: PG Gem

***GREAT EXO w5 day 3: todo\_app : link sql, Sinatra, http methods***

Require ‘pg’ on top of ruby file (+ require ‘pry’ )

(pg is gem on command line for postgressql)

require 'pg'

require 'pry'

# connect to this database on our machine (localhost)

db = PG.connect(dbname: 'facebook', host: 'localhost')

# execute a command to the databse

sql = insert into people (name) values (‘gui’)

db.exec(sql)

### # Pass on commands to the database, from the Ruby file

To ensure connection to database before every method we run:

KEY TO CLOSE THE DB AFTER USE

before do

@db = PG.connect( dbname: 'movies\_app', host: 'localhost')

end

after do

@db.close

end

If passing database instructions in a block inside a GET or Function

KEY TO CLOSE THE DB AFTER USE

def display\_rows(sql) (here sql would take ("select \* from people)

db = PG.connect(dbname: ‘name’, host: ‘localhost’)

begin

db.exec (sql) do |result|

result.each {|row| puts "#{row['first']} #{row['last']}" }

end

ensure

db.close

*# simply ensure database is closed after we run our line*

end

end

***# result is the PG Object representing the people table;***

***# PG objects is made of rows, kind of like an array***

***# we iterate over the rows in the table: row[0], row[1], etc***

***# !! but this is NOT an array !!***

***# result.first is the Hash with all data for first person***

***=> {"id"=>"1", "first"=>"jeremy", "last"=>"marer", "dob"=>nil, "relationship"=>nil, "friends"=>"0", "city"=>nil}***

### Write Ruby code to add a new person to the table people

begin

puts "Full name: "

name = gets.chomp.split()

puts "date of birth: "

dob = gets.chomp

puts = "relationship"

relationship = gets.chomp

puts "City: "

city = gets.chomp

print "#{name} was born in #{dob} and lives in #{city}"

sql = "INSERT INTO people (first, last, dob, relationship, city) VALUES ( '#{name[0]}', '#{name[1..-1].join(' ')}', '#{dob}', '#{relationship}', '#{city}' )"

***!!! sql values have to be put between ‘quotes’ unless we want to write an integer !!!***

db.exec(sql)

ensure

db.close

end

order in, order by

# Mongo DB

Non-relational document driven database

Mongo is very fast, great with apps running on Node

Uses mongoose as ORM

Objects are known as Documents

PSQL is relational database. Was is great for integration with a rails project

Uses Active record as ORM

ORM, object relational mapper. It links objects with rows in a database

If doing a Node project

Npm init (to create a package.json file)

START SERVER: run ‘mongod’ in terminal

npm install --save mongoose

remember save to put it as a dependency on package.json

Once database is seeded (SEE w8\_d3\_to eatly mustache mongo )

Run ‘mongo’ on terminal to enter the db (same as good ol psql)

‘Show databases’

Then do ‘use foodsdatabase’ to connect to the db foods

‘Show collections’