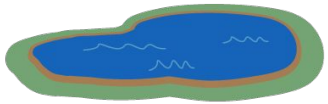
Ashland Daily Photo
http://travelJapanblog.com/ashland/2012/08/portland-oregon-from-the-air/

Data Repository

All data as we receive it stored in one place.



Databases

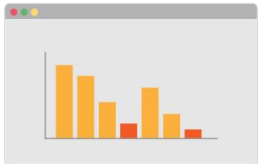
Structured views of the data we receive.

APIs

Web accessible ways for developers to interface with our databases

Web

Public knowledge accessible for all



Learn PostGres the Hack Oregon Way

PostgreSQL

High-level takeaways

- Features of PostGres
- CRUD & Loading Data
- Performance
- Other Tools
- Security
- PostGIS
- PostGres at HackOregon

Postgres or PostGreSQL?

- Originally named POSTGRES for (Post Ingres)
- Project and product named PostgreSQL
- Pronounced *“post gress queue ell”*
- SQL can be pronounced *“ess queue ell”* or *“sequel”*

Industry Trends

- Backend frameworks (Apache/PHP, Ruby on Rails, Django)
- Work with any database (MySQL/MariaDB, SQLite, PostgreSQL)
- Use the database for CRUD, application logic is all in PHP / Ruby / Python / JavaScript code

Features of Postgres

- Industrial-strength open source permissive-licensed CRUD engine
- Fully ACID compliant
- Scales to huge installations
- Replication / failover / high availability as standard equipment
- Full-text search
- Stored procedures in Python, Perl, Ruby, R, Tcl and Lua
- JSON document stores (jsonb)
- Key-value stores (hstore)
- Foreign data wrappers. Access to external data sources from within the Postgres database. (Note: We don't use this on AWS RDS)

Hack Oregon Page in Github

https://github.com/hackoregon/disaster-resilience/wiki/Tools-For-Hack-Oregon

Installation - Windows or Mac

- Download from here: https://www.PostgreSQL.org/download/
- Select the most recent version (11.3)
- Select your operating system (Windows or Mac)

Installation - *NIX

- Linux: use the PGDG repositories
- RHEL / CentOS / Fedora: https://www.PostgreSQL.org/download/linux/redhat/
- Linux Mint 18 / Ubuntu 16.04 LTS ("Xenial"): https://github.com/hackoregon/data-science-petcontainers/blob/master/docs/Mint18-Xenial/README.md
- Debian: https://www.PostgreSQL.org/download/linux/debian/

pgAdmin

- A free and open-source graphical user interface (GUI) administration tool for PostgreSQL
- Installation https://www.pgadmin.org/download/

Connecting with pgAdmin – Windows

- Connecting with pgAdmin - (R. Obe and Hsu 2017b, chap. 4)
  - Right-click on "Servers"
  - Create a server
  - Give it any name you want
  - Fill in the connection tab
- EnterpriseDB connection
  - Host: localhost
  - Port: 5432
  - Maintenance database and user are Postgres
  - Password is what you set when you installed

Connecting with pgAdmin - Linux

Linux connection (at least on Mint / Ubuntu)

Host: /var/run/PostgreSQL

Port: 5432

Maintenance database and user are the same as your Linux ID

Password is empty

Follow the steps here for setup:
https://github.com/hackoregon/data-science-pet-containers/tree/master/old-docs/Mint18-Xenial

CRUD - Create / Read / Update / Delete

- SQL is set-based
- Relational database model
- Most implementations, everything is data
    - Schema
    - Procedures
    - Users
    - Extensions
    - etc.

Create

- Creating
  - Database (pgAdmin)
  - View
  - Column
  - Functions
  - Events
  - Triggers
- Create table
- Creating tables - Data Definition Language (DDL)
- Working with tables, you need to do two things:
  - Define the names and data types of every column in the table.
  - Load the data into the table.

Create

We'll be working with a sample file of Oregon highway mileposts

https://github.com/hackoregon/data-science-pet-containers/blob/master/examples/mileposts/Mileposts_2014

- Tip: If you don't know a field's data types, you can always just set them all to text and re-cast them to the correct type with SQL later!
- In this case it's mostly obvious which columns are numeric or timestamps, and we can use text for the rest.
- For the mileposts, we'll use double precision for the latitude and longitude and text for the others.

Create

- The DDL

```
CREATE TABLE mileposts_2014 (
  hwyname text,
  hwynumb text,
  st_hwy_sfx text,
  rdwy_id text,
  mlge_typ text,
  ovlp_cd text,
  mp text,
  mp_desc text,
  mp_disp text,
  lrs_key text,
  lrm_key text,
  lat double precision,
  longtd double precision,
  hrz_col_m text,
  crd_rf_dtm text,
  effectv_dt text,
  gis_prc_dt text);
```

Naming in Postgres

A tip on naming:

PostgreSQL requires special care in coding SQL queries when column names have anything besides lower-case letters, numbers or underscores.

If the table, view or role has a hyphen or capital letter it needs to have double-quotes.

It’s a real hassle, so snake_case rules!

Loading Data

Loading the data into the table:

psql \copy

psql is the command-line PostgreSQL client.

Fully scriptable - you can write programs in it

Has string substitution.

You can mix SQL statements and psql commands.

It's great for reproducibility But it's another language to learn. Most folks may prefer to script in a language they know, like Python, which has its own PostgreSQL client libraries.

Data Types

char(10) - fixed length

varchar(255) - variable-length

text - variable-length

integer - 4-byte integer that has a range from -2,147,483,648 to 2,147,483,647.

float - floating point precision

numeric - real number

date - date

timestamp - date & time

json - json

boolean - true / false

And more...

Read (query)

SELECT, TABLE, WITH -- retrieve rows from a table or view

Select statement

Where

Join

Order by

;

Read Example

```
select *
from public.mileposts_2014
where hwyname = 'KINGS VALLEY'
order by mp_disp;
```

Update

ALTER TABLE -- change the definition of a table

UPDATE -- update rows of a table

Update Examples

```
alter table public.mileposts_2014 add column source_file varchar(255);
```

```
update public.mileposts_2014 set source_file = 'Mileposts_2014.csv';
```

Delete

DELETE -- delete rows of a table

DROP TABLE -- remove a table

TRUNCATE -- empty a table or set of tables

Delete Examples

```
delete from
  public.mileposts_2014
where
  hwyname = 'CLACKAMAS';


alter table public.mileposts_2014 drop column source_file;
```

Aggregates

AVG([DISTINCT | ALL] expression)

COUNT(*)

COUNT([DISTINCT | ALL] expression)

MAX([DISTINCT | ALL] expression)

MIN([DISTINCT | ALL] expression)

SUM([DISTINCT | ALL] expression)

Example:

```
select count(*)
from public.mileposts_2014
where hwyname = 'KINGS VALLEY'
```

Views

Join

Subqueries

Returning Aggregates

Performance - Optimization

Rule of Optimization:

- Make it work
- Make it right
- Make it fast enough

Performance - Indexes

Primary keys are indexes

In large tables, consider adding indexes to any fields that will be searched, joined or sorted.

An index which covers multiple columns is generally better than creating multiple indexes for a single column.

PostgreSQL will consider using a multi-column index even if some of the columns in the index are not even used in the query.

For example if we had the following index:

create index order_dates on orders(creation_date, completion_date);

It can be used for queries using any combination of the creation_date and completion_date column. The benefit of having fewer indexes is that they might live in the shared memory, and caches, for a longer time.

Indexes (usually) have a minimal effect on the size of the db.

They can slow down inserts.

Performance - Query Tools

Explain — show the execution plan of a statement

https://www.PostgreSQL.org/docs/current/sql-explain.html

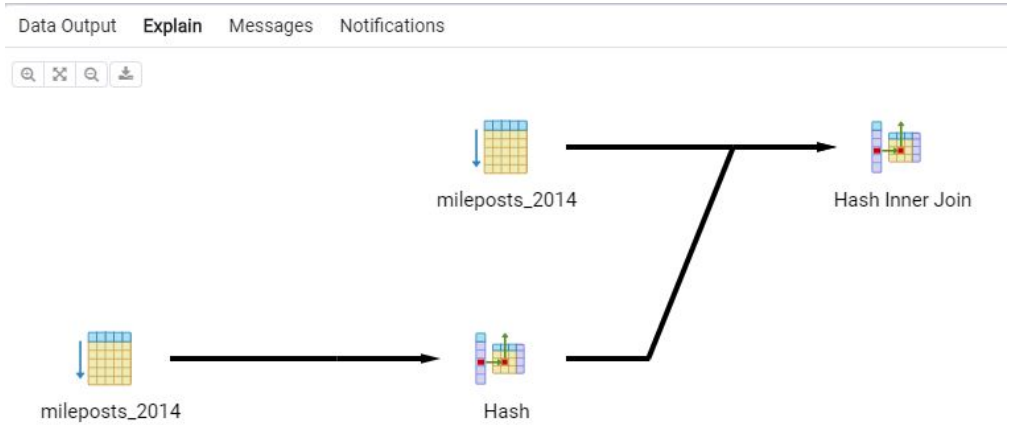
You can do this in pgAdmin

Analyze - collect statistics about a database

https://www.PostgreSQL.org/docs/current/sql-analyze.html

Explain - Example Output

```
Select
  A.hwyname, B.hwyname, A.mp
From public.mileposts_2014 as A
  join public.mileposts_2014 as B on B.mp = A.mp
Where A.hwyname = 'KINGS VALLEY'
```



Performance - Materialized Views

Materialized views

Queries returning aggregate, summary, and computed data are frequently used in application development, but sometimes these queries are not fast enough. The simplest way to improve performance is to use a materialized view.

A materialized view is a snapshot of a query saved into a table.

CREATE MATERIALIZED VIEW

https://www.PostgreSQL.org/docs/9.3/sql-creatematerializedview.html

Because it's a table, you can create indexes and use those efficiently.

Warning: this can increase the size of the database.

More about materialized views: https://hashrocket.com/blog/posts/materialized-view-strategies-using-PostgreSQL

Performance - Considerations

Do more in the database. - More efficient than pulling over many rows and then working in the front end.

Production is different than on the desktop, so test production.

Ref: This page has a lot of suggestions for perf. Tips
https://dev.to/elmuerte/improving-PostgreSQL-queries-4pc1

Psql

Psql - Psql is the interactive terminal for working with Postgres.
http://Postgresguide.com/utilities/psql.html

psql --version

sudo -u Postgres psql - Log into PostgreSQL on your computer. Login as default user "Postgres" with password password_of_your_ubuntu_system:

sudo -u role_name psql db_name - Log in directly into a database as a user from the terminal (with password as password_of_your_ubuntu_system).

Psql - Key Commands

- \l - List the available databases
- \du - List all users in PostgreSQL
- \c database_name - Connect to a db via the default user
- \c database_name user_name - connect to db with a specific user
- \q - Exit out of psql
- \d - List all tables in psql
- \dt - listing the available tables in the current database
- \d <table name> - Describe schema of a table
- SELECT version(); - show version of Postgres
- \g - show command history
- \? - listing all the available commands
- \h <command> - help on a specific command
- \timing - turn on timing for inspecting query performance

Backup & Restore - Backup

use pg_dump to make a backup of a database (pgAdmin live)

Q: What's the best way to dump the schema definition?

A: pgdump can be used for dumping everything, or selectively dump.

(FYI, pgdump can be called from pgAdmin: right-click on database and choose "Backup")

Q: What if I want to export code to create the tables in a new database?

A: pgdump --format=p

eg:

\pg_dump.exe --host localhost --port 5432 --username "Postgres" --no-password --format plain --verbose --file "C:\temp\export-me2.backup" "Postgres"

Backup & Restore - Restore

Restoring a database: pg_restore (pgAdmin live)

Tip: When restoring a dump, restore it without the owner

Security - General

Default Postgres port is 5432 and username Postgres.

By default, the owner of any database you create is Postgres.

If you want your database to belong to a specific user, switch to that user and then create the database.

Security - Users & Roles

Change the default port & user name on a server (or even on a desktop) to avoid hackers cracking your password.

Consider restricting listen_address to localhost on a desktop.

Logins / roles are set at the database server level.

Grant permissions:

grant SELECT on public.mileposts_2014 to "ima-readonly-user";

Security - AWS

On a desktop, you’re usually the database superuser. Everything works, and your machine’s behind a firewall.

On a server, there’s a DBA and a SysAdmin that hand out privileges only with justification. You don’t get it unless you need it, and everything you do is logged and monitored. Bottom line: don’t be surprised if you get told, "No! Find another way."

PostGIS

- PostGIS cool features
    - Geographic Information Systems (GIS)
    - Read and write GIS data files
    - Process geometric, geographic and topology GIS data types
    - Both vector and raster data
    - Geocoding, reverse geocoding, address standardization
- Open Source route planning libraries (OSRM and pgRouting)
    - Shortest / fastest / lowest cost routes from point A to point B
    - Traveling salesperson problem
    - Turn-by-turn directions for cars, bikes and pedestrians!

PostGIS - Installation

PostGIS will be included in the container deployed on AWS in production

To install the extensions in your database

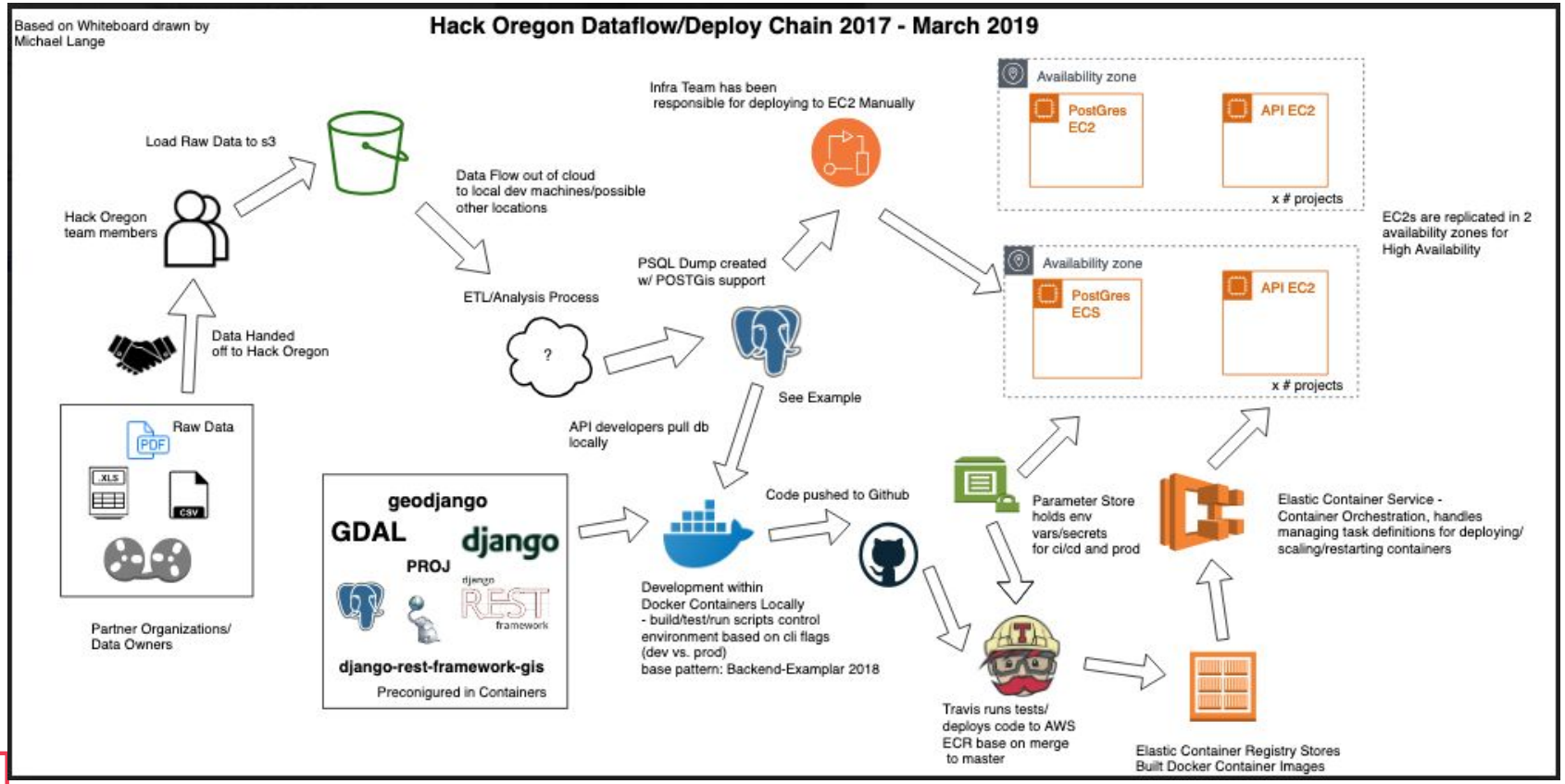
Postgres in Docker Containers

Usually Debian stable - similar to Ubuntu

Sometimes Alpine - avoid this unless you want to do a lot of research

Ed has a full PostgreSQL / PostGIS / pgRouting stack in a container at https://github.com/hackoregon/data-science-pet-containers

Postgres at Hack Oregon



Postgres at Hack Oregon

Open Source

We host in AWS RDS which provides a managed environment

Deploying databases from S3 through jumpbox to RDS

Postgres at Hack Oregon

Best practices

- Track schema changes in code under revision in git.
- Use Django migrations to alter schema
- Include small datasets in the migrations
    - eg: lookup tables
    - Data that helps with processing
- Load large datasets from source data (that is in a shared repository)
- Data transformations?
    - Use views to simplify & aggregate data
    - Or, preprocess from static database and then load only the refined data

Postgres at Hack Oregon

The (data) cycle of life: (proposed)

- Make code/schema changes in the models
- Migrate model changes to the local database
- Load data into local database
    - Or, use Fixtures to do some data migrations
- Backup to a dump file
- Restore dump file to production
- Deploy code changes. Database schema changes have already been deployed

Thanks to Ed

Ed Borasky's Postgres orientation for Hack oregon 2018 season

"Getting Started with PostgreSQL M. Edward (Ed) Borasky 2019-04-30"

https://github.com/hackoregon/data-science-pet-containers/blob/master/old-docs/getting-started-with-PostgreSQL.pdf

More Information

Postgres documentation
https://www.postgresql.org/docs/

Postgres free books, cheap books:
https://www.packtpub.com/packt/offers/free-learning

PgRouting: A Practical Guide. Locate Press LLC.
https://locatepress.com/pgrouting

PostgreSQL: Up and Running, 3rd Edition. O'Reilly Media, Inc.
http://shop.oreilly.com/product/0636920052715.do

PostGIS in Action, Second Edition. Manning Publications Co.
https://www.manning.com/books/postgis-in-action-second-edition

Thank you!