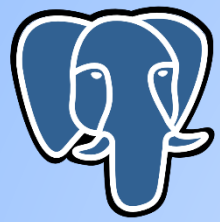
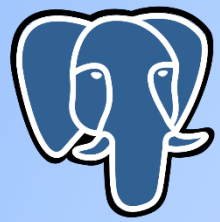


PostgreSQL

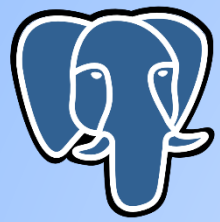
Module 4

Exploring the Databases



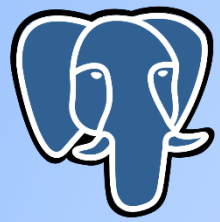
psql - The PostgreSQL Interactive Terminal

- psql is a command-line interface to PostgreSQL
- Query tool supplied as part of the core distribution of PostgreSQL
- It is available and works similarly in all environments
- Provides features for use as both an interactive query tool and as a scripting tool
- Is a sufficient command to allow you access to the PostgreSQL server



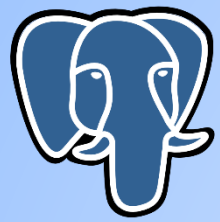
psql

- PostgreSQL interactive terminal – psql
- Syntax - psql [option...] [dbname [username]]
- Provides a number of meta-commands
- Options
  - -d dbname or – dbname=dbname
  - -f filename or --file=filename
  - -h hostname or --host=hostname
  - -l or –list
  - -p port or --port=port
  - -U username or --username=username
  - -V or --version



psql

- In psql, you can enter two types of commands:
  - psql meta-commands
    - meta-command is a command for the psql client
  - SQL
    - If it isn't a meta-command, then it's SQL. We keep reading SQL until we find a semicolon
- There are two types of help commands, which are as follows:
  - ?: This provides help on psql meta-commands
  - h: This provides help on specific SQL commands



Checking the connection

- Once connected with the server, confirm the connection
  - **postgres=# SELECT current_database();**
- The following command shows the current_user ID:
  - **postgres=# SELECT current_user;**
- To get IP address and port of the current connection:
  - **postgres=# SELECT inet_server_addr(), inet_server_port();**
- You can also use psql's new meta-command, conninfo:
  - **postgres=# \conninfo**

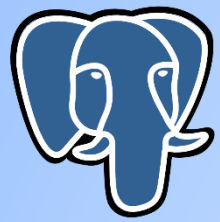
Server Information

- Version by directly querying the database server.
  - **postgres # SELECT version();**
  - **bash # psql - -version**
- Server uptime, how long since the server is started.
  - **postgres=# SELECT pg_postmaster_start_time();**

pg_postmaster_start_time

2016-03-27 14:31:51.3829.66+00

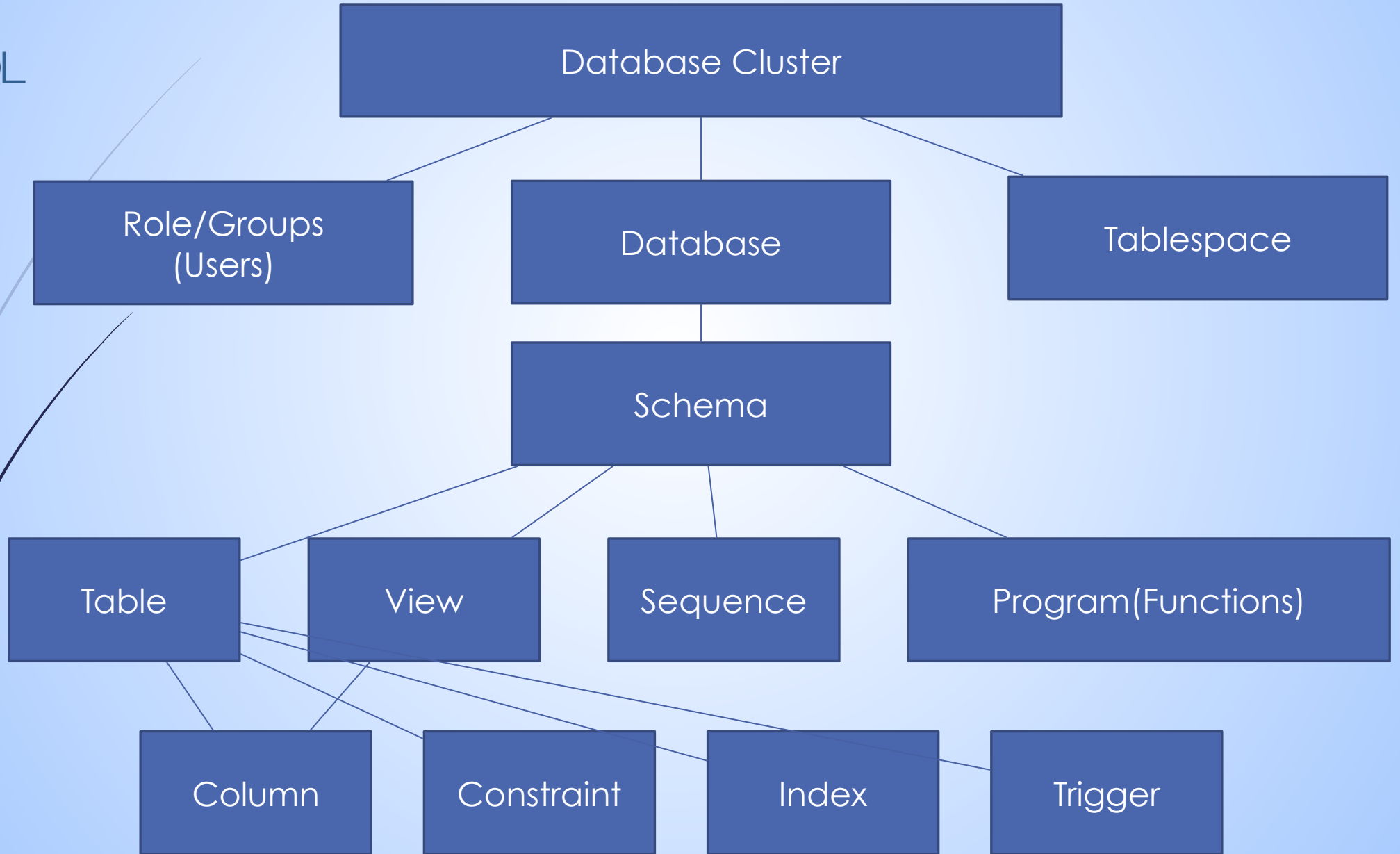
  - **postgres=# SELECT current_timestamp - pg_postmaster_start_time();**
  - **Postgres=# SELECT date_trunc('second', current_timestamp - pg_postmaster_start_time()) as uptime;**

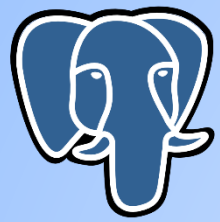


PostgreSQL Understanding Of Objects

- A PostgreSQL database cluster contains one or more named databases
- Any given client connection to the server can access only the data in a single database, the one specified in the connection request
- Database is collection of database objects such as schema, which contains table, view, sequence, functions and other database objects
- A database cluster is a collection of databases that is stored at a common file system location (the "data area")
- **pg_database**
  - A database is a collection of schemas and the schemas contain the tables functions, etc.

Object Hierarchy



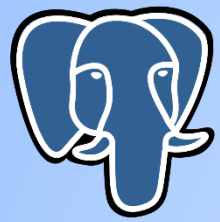


Choosing Good Names for Database Objects

- Ensure that all the objects have a meaningful name
- Points you should consider when naming your database objects:
  - The name clearly describes the role or table contents
  - For major tables, use short, powerful names
  - For associative or linked tables, use all of the names of the major tables to which they relate
  - Make sure that the name is clearly distinct from other similar names
  - Use underscores. Case is not preserved by default, so don't use CamelCase names
  - Use suffixes to identify the type of an object
- The standard names for indexes in PostgreSQL are as follows:
  - **{tablename}_{columnname(s)}_{suffix} –> accounts_col1_idx**

PostgreSQL Organizes Data

- See all databases in a cluster using the following query:
  - SELECT datname, oid FROM pg_database;
- Find storage for these databases by looking in the $PGDATA directory
  - cd /var/lib/pgsql/12/data/
  - ls -l base/
- The name of each subdirectory corresponds to the oid of one entry in the pg_database table
  - /usr/pgsql-12/bin/oid2name
- Inside database directory, there are lot of files with numeric filenames
  - Find filenames and table oids
    - psql -q -d template1
    - SELECT relname, oid FROM pg_class;
    - SELECT relname, oid, relpages, reltuples FROM pg_class ORDER BY oid;



Create Database

- In order to create database , your server must be up and running
  - **Create database [database-name];**
- Sometimes you want to create a database for someone else
  - **Create database [database-name] owner [rolename];**
- or you can do that from the command line using:
  - **bash $ createdb demo**

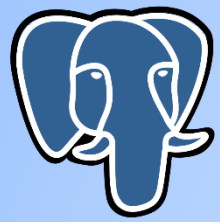
Listing Databases In Databases Server

- When we connect to PostgreSQL, we always connect to just one specific database on any database server
- In PostgreSQL, a database server is potentially split into multiple individual databases
  - **postgres # \l** – to list the databases

```
                                  List of databases
   Name    | Owner   | Encoding |   Collate   |    Ctype    | Access privileges
-----------+---------+----------+-------------+-------------+-------------------
postgres   | sriggs  | UTF8     | en_GB.UTF-8 | en_GB.UTF-8 |
template0  | sriggs  | UTF8     | en_GB.UTF-8 | en_GB.UTF-8 | =c/sriggs        +
           |         |          |             |             | sriggs=CTc/sriggs
template1  | sriggs  | UTF8     | en_GB.UTF-8 | en_GB.UTF-8 | =c/sriggs        +
           |         |          |             |             | sriggs=CTc/sriggs
(3 rows)
```

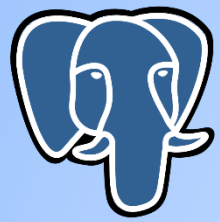
Table Inheritance

- Inheritance is an object-oriented concept when a subclass inherits the properties of its parent class
- For example, the capitals table inherits from cities table. (It inherits -- all data fields from cities.)
  - CREATE TABLE cities (name text, population float8, altitude int);
  - CREATE TABLE capitals (state char(2)) INHERITS (cities);
- Now, let's populate the tables
  - INSERT INTO cities VALUES ('San Francisco', 7.24E+5, 63);
  - INSERT INTO cities VALUES ('Las Vegas', 2.583E+5, 2174);
  - INSERT INTO cities VALUES ('Mariposa', 1200, 1953);
  - INSERT INTO capitals VALUES ('Sacramento', 3.694E+5, 30, 'CA');
  - INSERT INTO capitals VALUES ('Madison', 1.913E+5, 845, 'WI');



Enforcing Data Integrity

- Data integrity ensures the consistency and correctness of data stored in a database.
- It is broadly classified into the following four categories:
  - Entity integrity
  - Domain integrity
  - Referential integrity
  - User-defined integrity
- Entity Integrity
  - Ensures that each row can be uniquely identified by an attribute called the primary key



Enforcing Data Integrity

- Domain Integrity
  - Ensures that only a valid range of values is allowed to be stored in a column
- Referential Integrity
  - Ensures that the values of the foreign key match with the value of the corresponding primary key
- User-Defined Integrity
  - Refers to a set of rules specified by a user, which do not belong to the entity, domain, and referential integrity categories

Understanding Object Dependencies

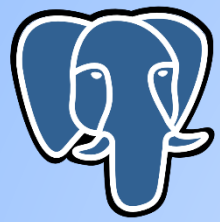
- In most databases, there will be dependencies between objects in the database
- There are two tables, as follows:
  - **CREATE TABLE orders (orderid integer PRIMARY KEY);**
  - **CREATE TABLE orderlines (orderid integer,lineid smallint,PRIMARY KEY (orderid,lineid));**
- Now, we add a link between them to enforce what is known as **Referential Integrity**, as follows:
  - **ALTER TABLE orderlines ADD FOREIGN KEY (orderid) REFERENCES orders (orderid);**
- Drop the referenced table

Listing Tables In Database

- The number of tables in a relational database is a good measure of the complexity of a database
  - **postgres # \d** – to list the tables in public schema

```
        List of relations
 Schema |   Name   | Type  |  Owner
--------+----------+-------+----------
 public | accounts | table | postgres
 public | branches | table | postgres
```

  - **SELECT count(*) FROM information_schema.tables WHERE table_schema NOT IN ('information_schema', 'pg_catalog');**



Disk Space Used By Database

- For planning or space monitoring, we often need to know how big the database is.
  - The easiest way is to just ask the database a simple query, like this:
    - **SELECT pg_database_size(current_database());**
  - If you want to know the size of all the databases together, then you'll need a query such as the following:
    - **SELECT sum(pg_database_size(datname)) from pg_database;**
  - The database server knows which tables it has loaded. It also knows how to calculate the size of each table, so the pg_database_size() function just goes and looks at the file sizes

Disk Space Used By Tables

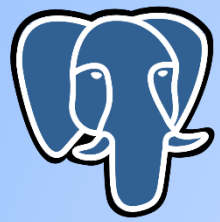
- We can see the size of a table using this command:
  - **postgres=# select pg_relation_size('table1');**

```
pg_relation_size
------------------
          13582336
(1 row)
```

- We can also see the total size of a table including indexes as follows:
  - **postgres=# select pg_total_relation_size('table2');**

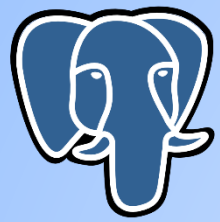
```
pg_total_relation_size
-----------------------
              15425536
(1 row)
```

- We can also use a psql command, like this:
  - **postgres=# \dt+ pg_types**



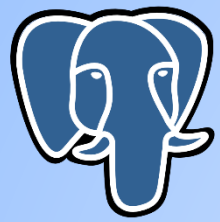
Biggest Tables In Database

- Rather than an absolute value for a specific table, let's look at the relative sizes
  - The following basic query will tell us the 10 biggest tables:
    - **SELECT table_name,pg_relation_size(table_schema || '.' || table_name) as size FROM information_schema.tables WHERE table_schema NOT IN ('information_schema','pg_catalog') ORDER BY size DESC LIMIT 10;**
  - The tables are shown in descending order of size, with at most 10 rows displayed. In this case, we look at all tables in all schemas, apart from tables in the information_schema or in pg_catalog,



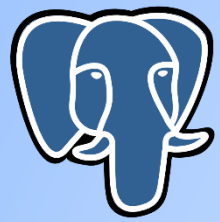
Create Schema

- Container or a namespace within a database
- Any object that we create in a database gets created under a schema
- Objects get created under a default schema called public
- Create new schema:
  - CREATE SCHEMA mynewschema;
  - CREATE TABLE mynewschema.emp (id integer, first_name text);
  - INSERT INTO emp VALUES(1,'Abc');
  - SELECT current_schema;
  - SHOW search_path;
  - SET search_path="$user",public,mynewschema;
  - SELECT schemaname, tablename FROM pg_tables WHERE tablename = 'emp';



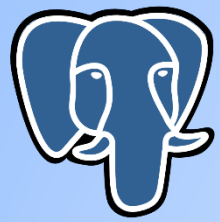
Create Schema

- Container or a namespace within a database
- Any object that we create in a database gets created under a schema
- Objects get created under a default schema called public
- SELECT schemaname, tablename FROM pg_tables WHERE tablename IN('emp','dept');
- Create new schema:
  - CREATE SCHEMA mynewschema;
  - CREATE TABLE mynewschema.emp (id integer, first_name text);
  - INSERT INTO emp1(id,first_name) VALUES(1,'Abc');
  - SELECT current_schema;
  - SHOW search_path;
  - SET search_path="$user",public,mynewschema;



Using Multiple Schemas

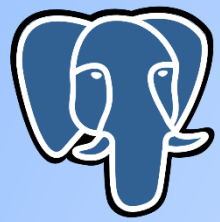
- We can separate groups of tables into their own "namespaces", referred to as "schemas".
  - Schemas can be easily created using the following commands:
    - **CREATE SCHEMA finance;**
    - **CREATE SCHEMA sales;**
  - We can then create objects directly within those schemas:
    - **CREATE TABLE finance.month_end_snapshot (.....)**
  - When we access database objects, we use the user-settable search_path parameter to identify the schemas to search.
  - If we want to let only a specific user look at certain sets of tables, we can modify their search_path parameter.
    - **ALTER ROLE fiona SET search_path = 'finance';**
    - **ALTER ROLE sally SET search_path = 'sales';**



Create Tablespace

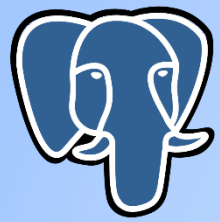
- A tablespace is a link to a location in the filesystem, that is, a directory
- It's a container to hold all other objects, such as tables, indexes, etc.
- Storing data in a location other than the default location, could be useful
- Two default tablespaces got created, one was a tablespace called **pg_default,** and the other one**: pg_global**
- At shell prompt, change the directory with the following command:
  - **cd /pgdata/12/pg_tblspc**
- Make sure postgres is the owner of /pgdata by executing
  - **chown postgres /pgdata**

- Then, as postgres user, execute:
  - mkdir /pgdata/tbl1
- Then, log in to psql and execute the SQL statements, as shown here:
  - CREATE TABLESPACE mytablespace LOCATION '/pgdata/tbl1';
- We can create a table and specify its tablespace
  - DROP TABLE IF EXISTS emp;
  - DROP TABLE IF EXISTS dept;
  - CREATE TABLE emp(id int, first_name text);
  - CREATE TABLE dept (id int, dept_name text) tablespace mytablespace;
  - \! oid2name -d test
- Select pg_relation_filepath('emp')
- Select pg_relation_filepath('dept')



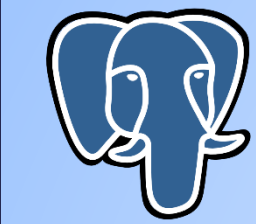
Adding An External Module To Postgresql

- Another strength of PostgreSQL is its extensibility.
  - There are many kinds of additional module offerings, such as the following:
    - Additional functions
    - Additional data types
    - Additional operators
    - Additional indexes
- The procedure that makes a module usable is actually a two-step process.
  - First, you install the module's files on your system so that they become available to the database server
  - Next, you connect to the database (or databases) where you want to use the module, and create the required objects.



There's more…

- we use the words "extension" and "module" as synonyms.
  - **CREATE EXTENSION myext;**
  - **ALTER EXTENSION myext UPDATE;**
- Installing from a manually downloaded package
- PostgreSQL functions and objects can reference code in these libraries, allowing extensions to be bound tightly to the running server process



Using An Installed Module

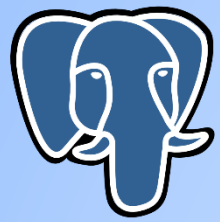
- How to enable an installed module so that it can be used in a particular database.
- Extension support is a great step towards an automated package management system for PostgreSQL.
- Using the extension infrastructure
  - **CREATE EXTENSION myextname;**
  - This will automatically create all the required objects inside the current database.
  - For security reasons, you need to do so as a database superuser. For instance, if you want to install the dblink extension, type this:
  - **CREATE EXTENSION dblink;**

Managing Installed Extensions

- We list all available extensions:
  - **Postgres-# SELECT * FROM pg_available_extensions ORDER BY name;**

```
name               | dblink
default_version    | 1.0
installed_version  | 1.0
comment            | connect to other PostgreSQL databases from within a database
```

  - we can list all the objects in the dblink extension, as follow:
    - **postgres=# \dx+ dblink**



Limitations

Limit	Value
Maximum Database Size	Unlimited
Maximum Table Size	32 TB
Maximum Row Size	1.6 TB
Maximum Field Size	1 GB
Maximum Rows per Table	Unlimited
Maximum Columns per Table	250 - 1600 depending on column types
Maximum Indexes per Table	Unlimited