

Relational Database Management Systems

- What is an RDBMS?
- RDBMSs in COMP3311
- PostgreSQL Architecture
- SQLite Architecture
- Using PostgreSQL in CSE
- Managing Databases
- Managing Tables
- Managing Tuples
- Table Definition Example
- Exercise: Creating/Populating Databases
- Managing Other DB Objects

❖ What is an RDBMS?

A relational database management system (RDBMS) is

- software designed to support large-scale data-intensive applications
- allowing high-level description of data (tables, constraints)
- with high-level access to the data (relational model, SQL)
- providing efficient storage and retrieval (disk/memory management)
- supporting multiple simultaneous users (privilege, protection)
- doing multiple simultaneous operations (transactions, concurrency)
- maintaining reliable access to the stored data (backup, recovery)

Note: databases provide **persistent** storage of information

❖ RDBMSs in COMP3311

PostgreSQL

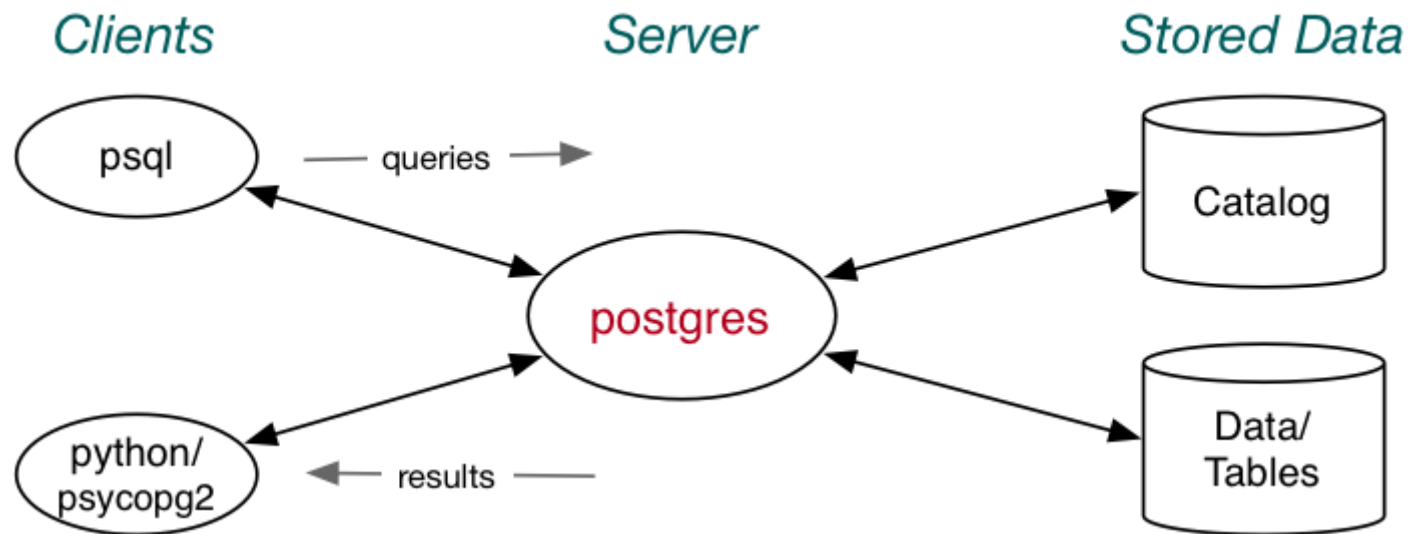
- full-featured, client-server DBMS, resource intensive
- applications communicate via server to DB
- can run distributed and replicated
- follows SQL standard closely, but not totally
- extra data types (e.g. JSON), multiple procedural languages

SQLite

- full-featured, serverless DBMS, light user of resources
- intended to be embedded in applications
- follows SQL standard closely, but not totally
- no stored procedures, add functions by embedding in apps

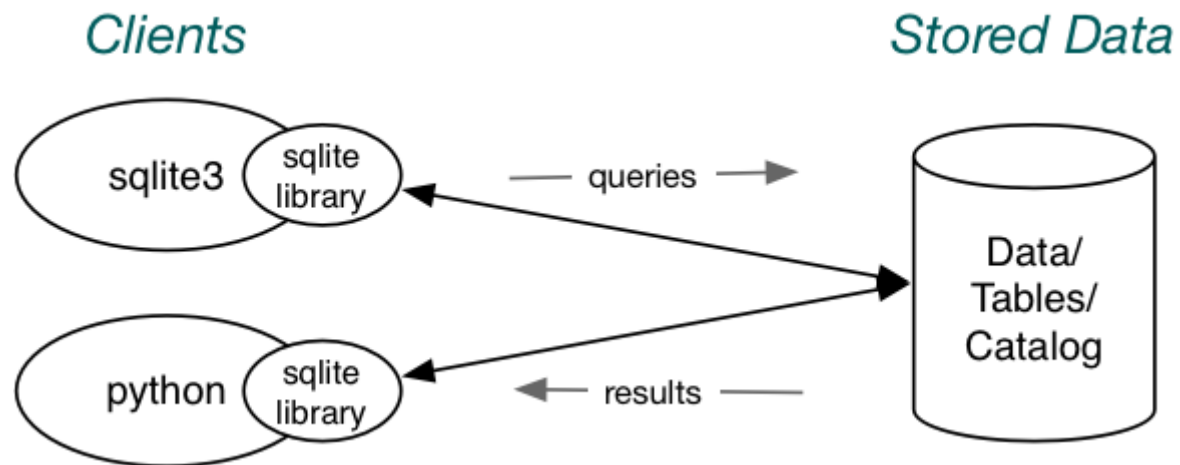
❖ PostgreSQL Architecture

PostgreSQL's client-server architecture:



❖ SQLite Architecture

SQLite's serverless architecture:



❖ Using PostgreSQL in CSE

Using your PostgreSQL server in CSE (once installed):

- login to grieg, set up environment, start server
- use psql, etc. to manipulate databases
- stop server, log off grieg

```
wagner$ ssh YOU@grieg
grieg$ source /srvr/YOU/env
grieg$ pg start
grieg$ psql myDatabase
... do stuff with your database ...
grieg$ pg stop
grieg$ exit
```

Need to run the command **priv srvr** once before the above will work

❖ Using PostgreSQL in CSE (cont)

PostgreSQL files (helps to understand state of server)

- PostgreSQL environment settings ... **/srvr/YOU/env**
- PostgreSQL home directory ... **/srvr/YOU/pgsql/**
- under the home directory ...
 - **postgresql.conf** ... main configuration file
 - **base/** ... subdirectories containing database files
 - **postmaster.pid** ... process ID of server process
 - **.s.PGSQL.5432** ... socket for clients to connect to server
 - **.s.PGSQL.5432.lock** ... lock file for socket
 - **Log** ... log file to monitor server errors, etc.

❖ Managing Databases

Shell commands to create/remove databases:

- **createdb** *dbname* ... create a new totally empty database
- **dropdb** *dbname* ... remove *all* data associated with a DB

(If no *dbname* supplied, assumes a database called *YOU*)

Shell commands to dump/restore database contents:

- **pg_dump** *dbname* > *dumpfile*
- **psql** *dbname* -f *dumpfile*

(Database *dbname* is typically created just before restore)

Main SQL statements in *dumpfile*: **CREATE TABLE**, **ALTER TABLE**, **COPY**

❖ Managing Tables

SQL statements:

- **CREATE TABLE** *table* (*Attributes+Constraints*)
- **ALTER TABLE** *table* *TableSchemaChanges*
- **DROP TABLE** *table(s)* [**CASCADE**]
- **TRUNCATE TABLE** *table(s)* [**CASCADE**]

(All conform to SQL standard, but all also have extensions)

DROP . . CASCADE also drops objects which depend on the table

- objects could be tuples or views, but *not* whole tables

TRUNCATE . . CASCADE truncates tables which refer to the table

❖ Managing Tuples

SQL statements:

- **INSERT INTO** *table* (*Attrs*) **VALUES** *Tuple(s)*
- **DELETE FROM** *table* **WHERE** *condition*
- **UPDATE** *table* **SET** *AttrValueChanges* **WHERE** *condition*

Attrs = (*attr*₁, ..., *attr*_{*n*}) *Tuple* = (*val*₁, ..., *val*_{*n*})

AttrValueChanges is a comma-separated list of:

- *attrname* = *expression*

Each list element assigns a new value to a given attribute.

❖ Table Definition Example

Make a table to hold student data:

```
CREATE TABLE Student (  
    zid      serial,  
    family   varchar(40),  
    given    varchar(40) NOT null,  
    d_o_b    date NOT NULL,  
    gender   char(1) check (gender in ('M', 'F')),  
    degree   integer,  
    PRIMARY KEY (zid),  
    FOREIGN KEY (degree) REFERENCES Degrees(did)  
);
```

serial is a special type which automatically generates unique integer values

❖ Exercise: Creating/Populating Databases

Do the following:

- create a database called **ex1**
- create a table **T** with two integer fields **x** and **y**
- examine the catalog definition of table **T**
- use **insert** statements to load some tuples
- use **pg_dump** to make a copy of the database contents
- remove the **ex1** database, then restore it from the dump

❖ Managing Other DB Objects

Databases contain objects other than tables and tuples:

- views, functions, sequences, types, indexes, roles, ...

Most have SQL statements for:

- **CREATE** *ObjectType name* ...
- **DROP** *ObjectType name* ...

Views and functions also have available:

- **CREATE OR REPLACE** *ObjectType name* ...

See PostgreSQL documentation Section VI, Chapter I for SQL statement details.

Produced: 10 Feb 2021