

PostgreSQL Overview

- PostgreSQL
- PostgreSQL Online
- User View of PostgreSQL
- PostgreSQL Functionality
- PostgreSQL Architecture

❖ PostgreSQL

PostgreSQL is a full-featured open-source (O)RDBMS.

- provides a relational engine with:
 - efficient implementation of relational operations
 - transaction processing (concurrent access)
 - backup/recovery (from application/system failure)
 - novel query optimisation (based on genetic algorithm)
 - replication, JSON, extensible indexing, etc. etc.
- already supports several non-standard data types
- allows users to define their own data types
- supports most of the SQL3 standard

❖ PostgreSQL Online

Web site: www.postgresql.org

Key developers: Tom Lane, Andres Freund, Bruce Momjian, ...

Full list of developers: postgresql.org/contributors/

Source code: ~cs9315/21T1/postgresql/src.tar.bz2

Documentation: postgresql.org/docs/12/index.html

❖ User View of PostgreSQL

Users interact via SQL in a **client** process, e.g.

```
$ psql webcms
```

```
psql (12.5)
```

```
Type "help" for help.
```

```
webcms2=# select * from calendar;
```

id	course	evdate	event
1	4	2001-08-09	Project Proposals due
10	3	2001-08-01	Tute/Lab Enrolments Close
12	3	2001-09-07	Assignment #1 Due (10pm)
...			

or

```
$dbconn = pg_connect("dbname=webcms");
$result = pg_query($dbconn, "select * from calendar");
while ($tuple = pg_fetch_array($result))
{ ... $tuple["event"] ... }
```


❖ PostgreSQL Functionality

PostgreSQL systems deal with various kinds of entities:

- **users** ... who can access the system
- **groups** ... groups of users, for role-based privileges
- **databases** ... collections of schemas/tables/views/...
- **namespaces** ... to uniquely identify objects (schema.table.attr)
- **tables** ... collection of tuples (standard relational notion)
- **views** ... "virtual" tables (can be made updatable)
- **functions** ... operations on values from/in tables
- **triggers** ... operations invoked in response to events
- **operators** ... functions with infix syntax
- **aggregates** ... operations over whole table columns
- **types** ... user-defined data types (with own operations)
- **rules** ... for query rewriting (used e.g. to implement views)

- **access methods** ... efficient access to tuples in tables

❖ PostgreSQL Functionality (cont)

PostgreSQL's dialect of SQL is mostly standard (but with extensions).

- attributes containing arrays of atomic values

```
create table R ( id integer, values integer[] );  
insert into R values ( 123, '{5,4,3,2,1}' );
```

- table-valued functions

```
create function f(integer) returns setof TupleType;
```

- multiple languages available for functions
 - PLpgSQL, Python, Perl, Java, R, Tcl, ...
 - function bodies are strings in whatever language

❖ PostgreSQL Functionality (cont)

Other variations in PostgreSQL's **CREATE TABLE**

- **TEMPORARY** tables
- **PARTITION**'d tables
- **GENERATED** attribute values (derived attributes)
- **FOREIGN TABLE** (data stored outside PostgreSQL)
- table type inheritance

```
create table R ( a integer, b text);  
create table S ( x float, y float);  
create table T inherits ( R, S );
```

❖ PostgreSQL Functionality (cont)

PostgreSQL stored procedures differ from SQL standard:

- only provides functions, not procedures
(but functions can return **void**, effectively a procedure)
- allows function overloading
(same function name, different argument types)
- defined at different "lexical level" to SQL
- provides own PL/SQL-like language for functions

```
create function ( Args ) returns ResultType
as $$
... body of function definition ...
$$ language FunctionBodyLanguage;
```

- where each **Arg** has a *Name* and *Type*

❖ PostgreSQL Functionality (cont)

Example:

```
create or replace function
    barsIn(suburb text) returns setof Bars
as $$
declare
    r record;
begin
    for r in
        select * from Bars where location = suburb
    loop
        return next r;
    end loop;
end;
$$ language plpgsql;
used as e.g.
select * from barsIn('Randwick');
```

❖ PostgreSQL Functionality (cont)

Uses **multi-version concurrency control** (MVCC)

- multiple "versions" of the database exist together
- a transaction sees the version that was valid at its start-time
- readers don't block writers; writers don't block readers
- this significantly reduces the need for locking

Disadvantages of this approach:

- extra storage for old versions of tuples (until **vacuum**'d)
- need to check "visibility" of every tuple fetched

PostgreSQL also provides locking to enforce critical concurrency.

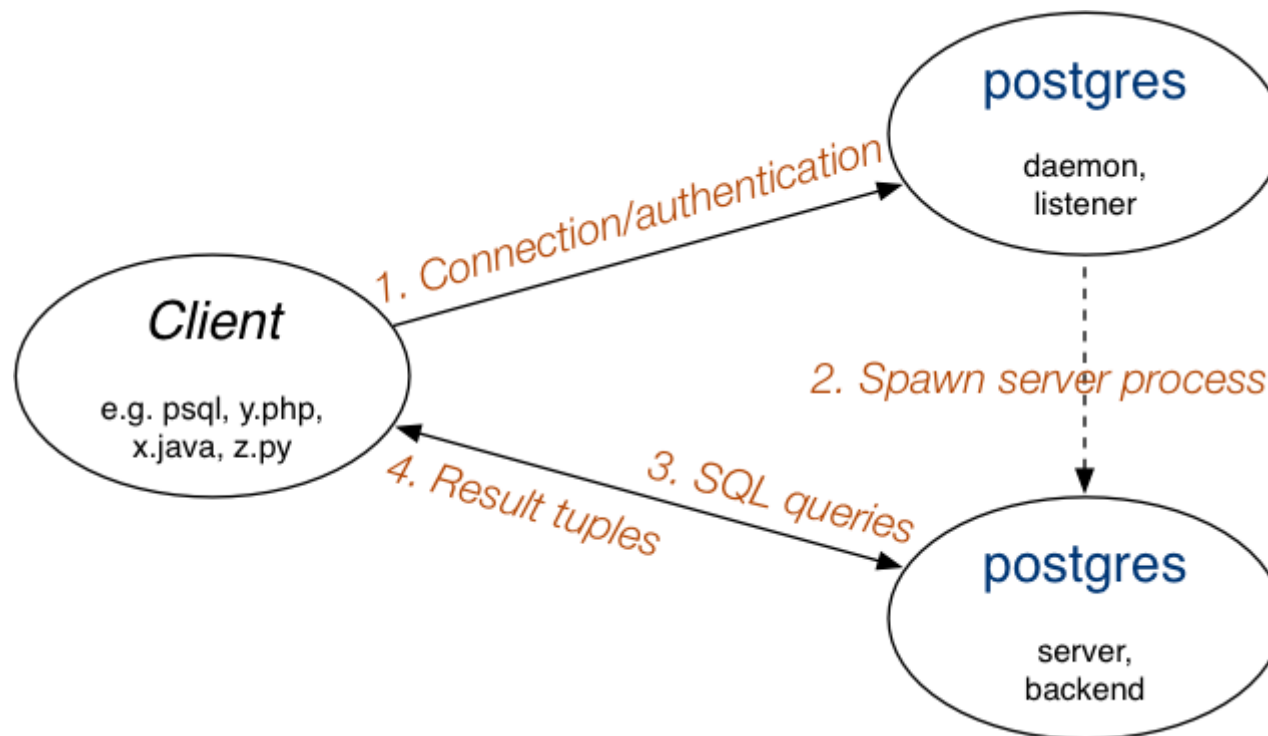
❖ PostgreSQL Functionality (cont)

PostgreSQL has a well-defined and open extensibility model:

- stored procedures are held in database as strings
 - allows a variety of languages to be used
 - language interpreters can be integrated into engine
- can add new data types, operators, aggregates, indexes
 - typically requires code written in C, following defined API
 - for new data types, need to write input/output functions, ...
 - for new indexes, need to implement file structures

❖ PostgreSQL Architecture

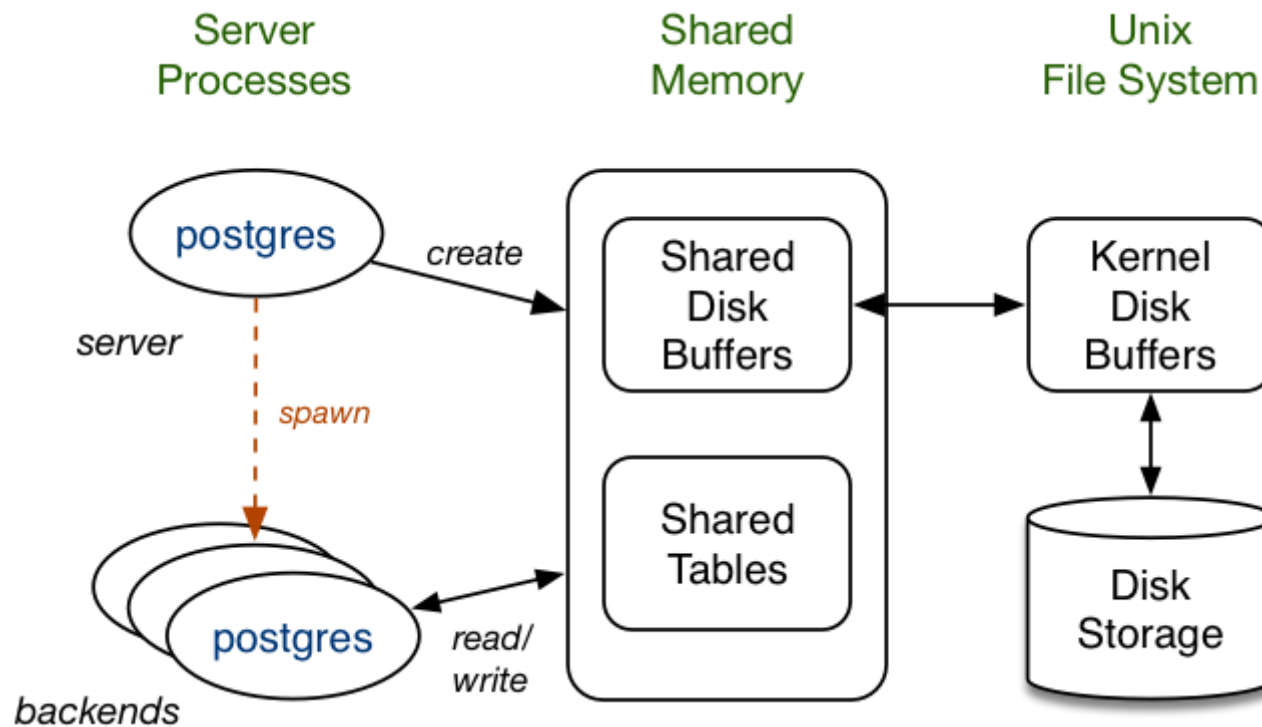
Client/server process architecture:



The listener process is sometimes called **postmaster**

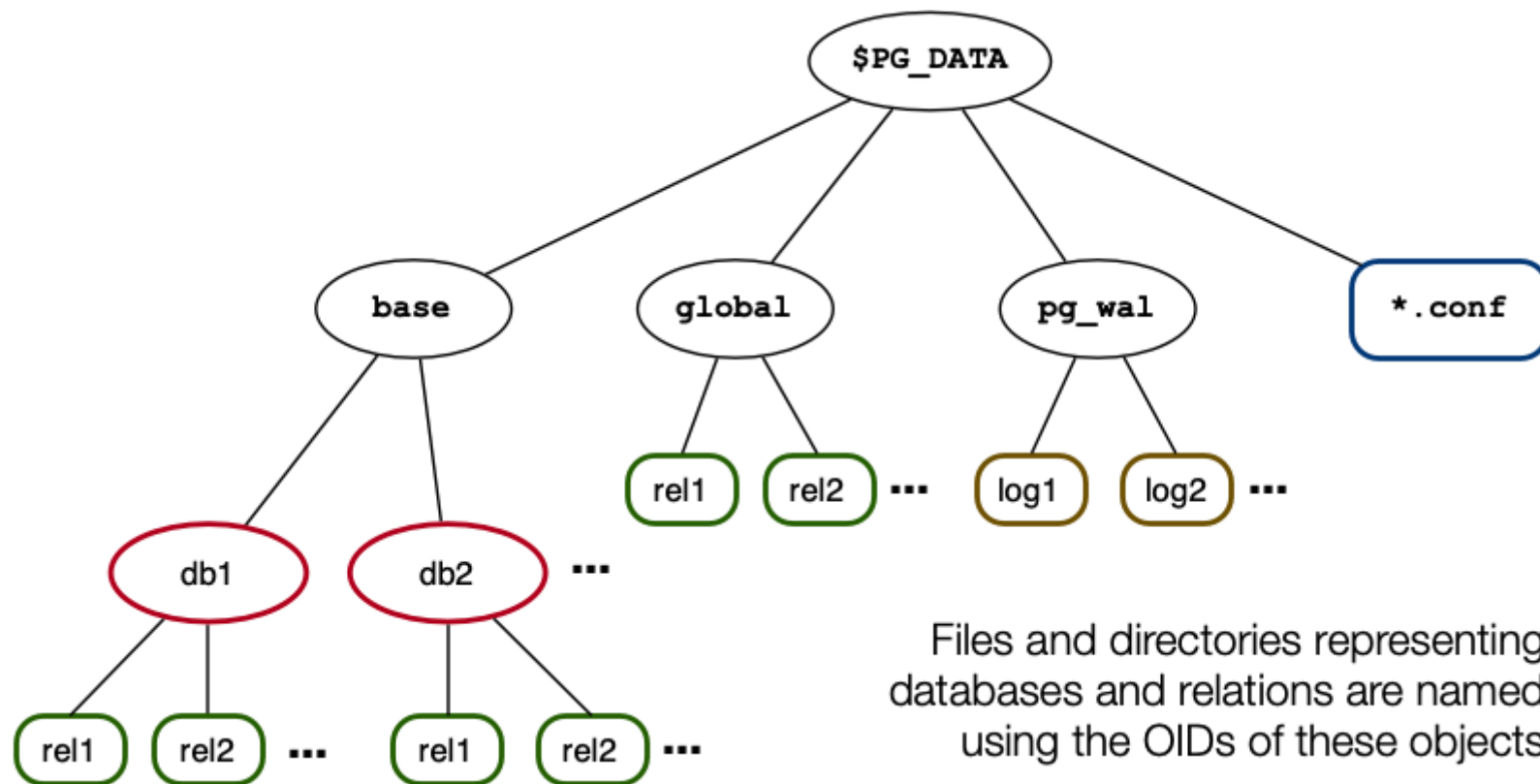
❖ PostgreSQL Architecture (cont)

Memory/storage architecture:



❖ PostgreSQL Architecture (cont)

File-system architecture:



Produced: 15 Feb 2021