FLORIDA POLYTECHNIC UNIVERSITY

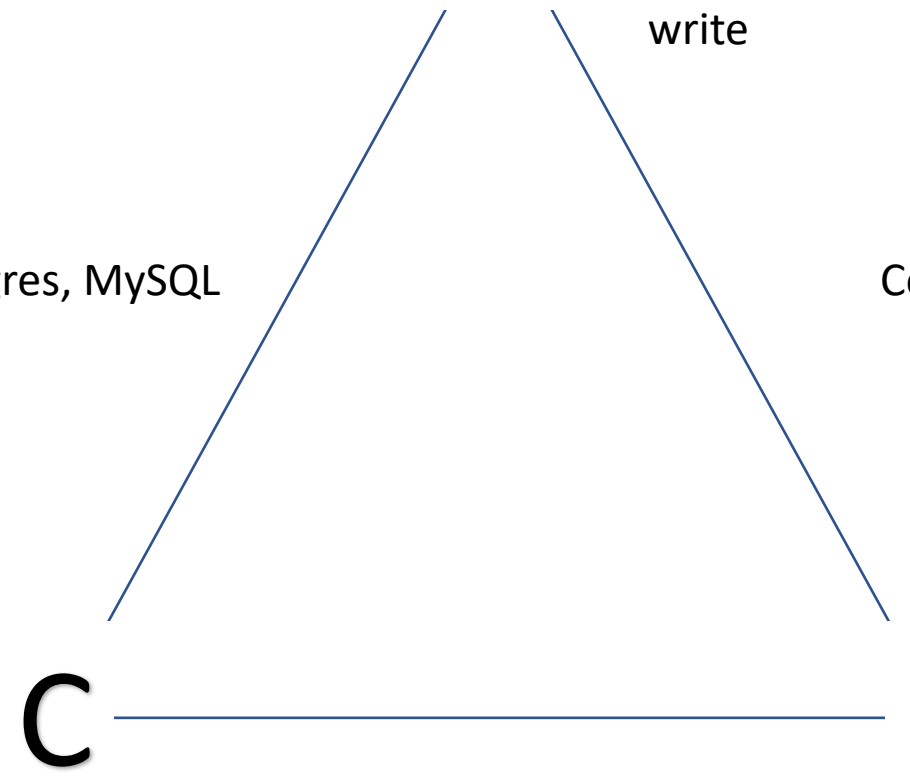
Postgresql

Overview

General Design

- PostgreSQL is an objected oriented architecture broken up into three large subsystems. These subsystems are:
  - Client Server (also known as the Front End)
  - Server Processes
  - Database Control

Where does it fit in CAP Theorem



Availability:
Each client can always read and write

RDBMS, Postgres, MySQL

Couch DB, Cassandra, Dynamo

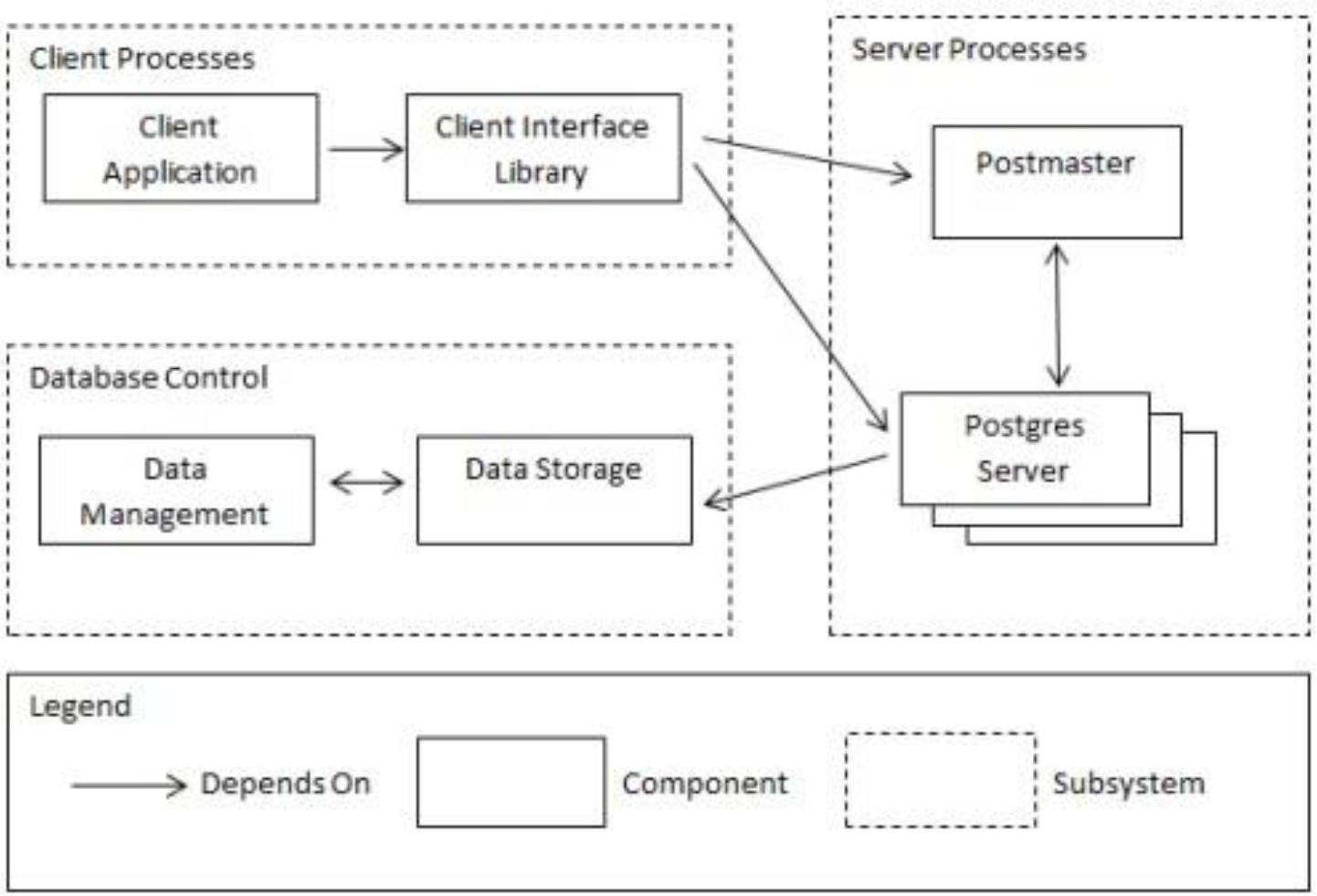
Consistency: All clients always have the same view of data

Big Table, Hbase, MongoDB, Redis

Partition Tolerance:
Works well despite physical network partitions

FLORIDA POLYTECHNIC UNIVERSITY

System Architecture



FLORIDA POLYTECHNIC UNIVERSITY

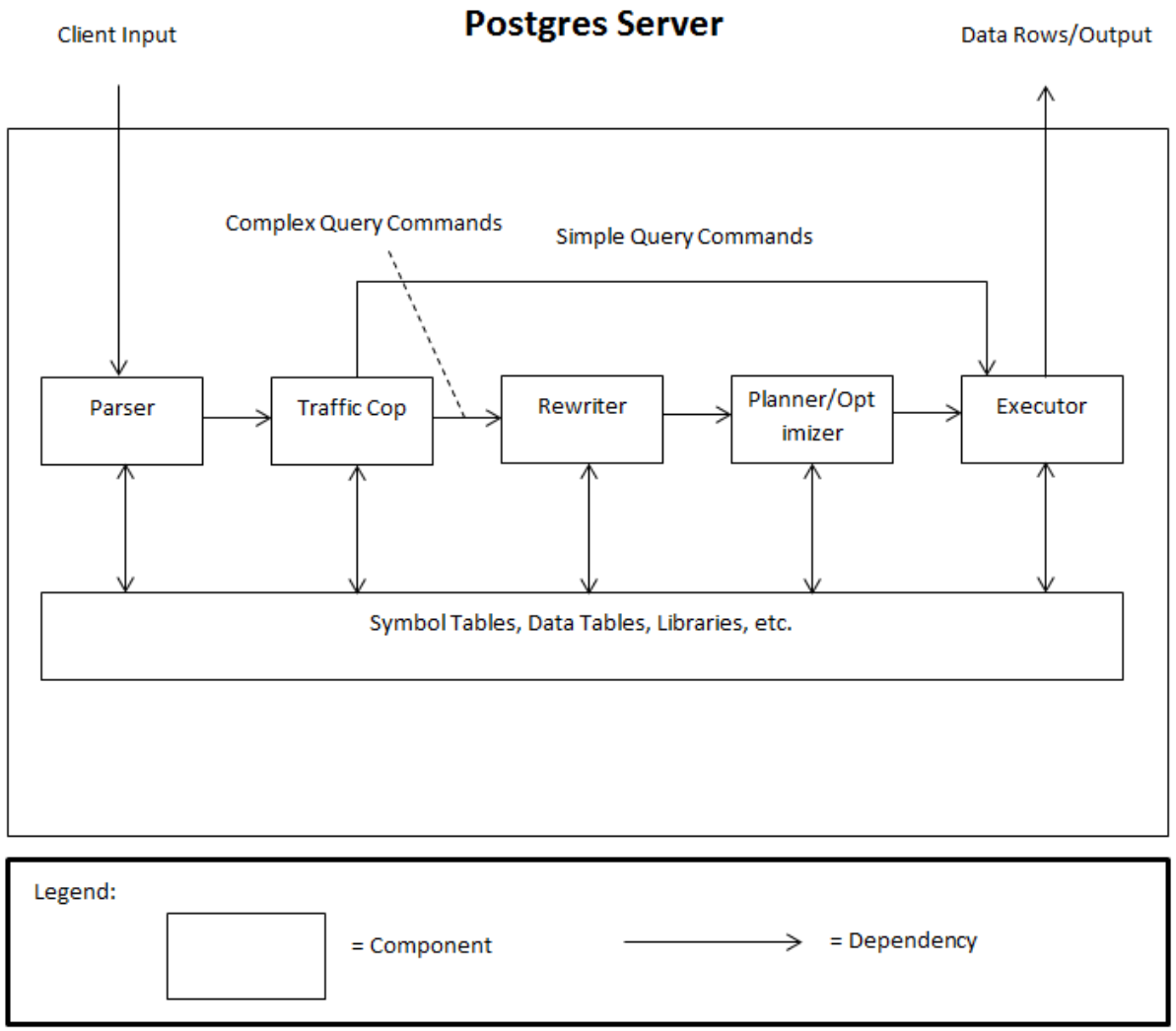
Client Server

- Comprised of two main parts: the client application and the client interface library.
- Many different client applications, many of which run on different OS's, some include: Mergeant, PGInhaler, SQirreL and more.
- Client interface library is the way that each of those applications can talk to the Server because the client interface library will convert to the proper SQL queries that the server can understand and parse.

Postmaster

- Is a daemon thread that runs constantly.
- Uses an implicit invocation architecture to listen for any and all calls to the database.
- When it receives a call from a client, it creates a back-end process (postgres server) to match it, using 1-1 correspondence.

Postgres Server Architecture



FLORIDA POLYTECHNIC UNIVERSITY

Parser

- Accepts an SQL query in the form of ASCII text.
- The *lexer* does pattern matching on the query to recognize identifiers and keywords.
- The *parser* then assembles these into a parse tree.
- The *parser* checks that the SQL query has valid syntax but does not understand the semantics.

Rewriter

- Lookup Rules
- Lookup View Definitions

Planner

- SQL queries can be executed in many different orders and produce the same results.
- The planner / optimizer will choose the best path or a reasonably efficient one if too many possibilities exist.
- It will then pass on the path to the executor.

Executor

- Receives plan from planner / optimizer in the form of a tree.
- Extracts the necessary data tables.
- Recursively goes through the plan, and performs the necessary action at each node.
- Pipe and filter, *not* batch processing.
- Returns output to client.

Catalogs

- Postgres is catalog-driven to a much greater extent than most other DBMSes.
- We have the usual sort of system catalogs to describe tables, their columns, their indexes, etc.
- But we also use system catalogs to store information about datatypes, functions, operators, index access methods, and so forth.
- The system can be extended by adding new catalog entries (and writing the underlying code, in the case of adding a function).

Basic system catalogs that describe a table:

- **pg_class**: one row for each table in the database, containing name, owner, access permissions, number of columns, etc.
- **pg_attribute**: one row for each column of each table, containing name, data type, column number, etc.
- **pg_index**: relates a table to its indexes. (Indexes are tables and so have their own entries in pg_class and pg_attribute, too.) One row per index, containing references to pg_class entries of index and underlying table, plus info about which columns of the table the index is computed on and what the index operators are.

Data Types Numbers

Data Type Syntax	Explanation
bit(*size*)	Fixed-length bit string Where ***size*** is the length of the bit string.
varbit(*size*) bit varying(*size*)	Variable-length bit string Where ***size*** is the length of the bit string.
smallint	Equivalent to int2. 2-byte signed integer.
int	Equivalent to int4. 4-byte signed integer.
integer	Equivalent to int4. 4-byte signed integer.
bigint	Big integer value which is equivalent to int8. 8-byte signed integer.
smallserial	Small auto-incrementing integer value which is equivalent to serial2. 2-byte signed integer that is auto-incrementing.
serial	Auto-incrementing integer value which is equivalent to serial4. 4-byte signed integer that is auto-incrementing.
bigserial	Big auto-incrementing integer value which is equivalent to serial8. 8-byte signed integer that is auto-incrementing.
numeric(*m*,*d*)	Where ***m*** is the total digits and ***d*** is the number of digits after the decimal.
double precision	8 byte, double precision, floating-point number
real	4-byte, single precision, floating-point number
money	Currency value.
bool	Logical boolean data type - true or false
boolean	Logical boolean data type - true or false

FLORIDA POLYTECHNIC UNIVERSITY

Data Types String

Data Type Syntax	Explanation
char(*size*)	Where ***size*** is the number of characters to store. Fixed-length strings. Space padded on right to equal ***size*** characters.
character(*size*)	Where ***size*** is the number of characters to store. Fixed-length strings. Space padded on right to equal ***size*** characters.
varchar(*size*)	Where ***size*** is the number of characters to store. Variable-length string.
character varying(*size*)	Where ***size*** is the number of characters to store. Variable-length string.
text	Variable-length string.

Data Type Date Time

Data Type Syntax	Explanation
date	Displayed as 'YYYY-MM-DD'.
timestamp	Displayed as 'YYYY-MM-DD HH:MM:SS'.
timestamp without time zone	Displayed as 'YYYY-MM-DD HH:MM:SS'.
timestamp with time zone	Displayed as 'YYYY-MM-DD HH:MM:SS-TZ'. Equivalent to timestamptz.
time	Displayed as 'HH:MM:SS' with no time zone.
time without time zone	Displayed as 'HH:MM:SS' with no time zone.
time with time zone	Displayed as 'HH:MM:SS-TZ' with time zone. Equivalent to timetz.

Features

- complex queries
- foreign keys
- triggers
- updatable views
- transactional integrity
- multiversion concurrency control

Extensibility

- data types
- functions
- operators
- aggregate functions
- index methods
- procedural languages