Conceptual Architecture of PostgreSQL

PopSQL
Andrew Heard, Daniel Basilio,
Eril Berkok, Julia Canella,
Mark Fischer, Misiu Godfrey

Principles and Key Mechanisms for Research

- Through extensive internet research, a conceptual architecture was slowly pieced together
- The most helpful items found were
  1. The PostgreSQL Developer's Handbook
  2. How PostgreSQL Processes a Query by Bruce Momjian
  3. PostgreSQL Documentation

- After the research was collected it was analyzed to determine which architecture PostgreSQL employed.
- Research also showed interrelationships with components in the system.
- Source code was also found and will be used to recover the concrete architecture.

A General Overview

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

- Within these subsystems, other architectures such as a hybrid pipe and filter (in the Postgres Server process), implicit invocation (in the Postmaster), client-server (with the Postmaster as the server), and object oriented (in the database control) .

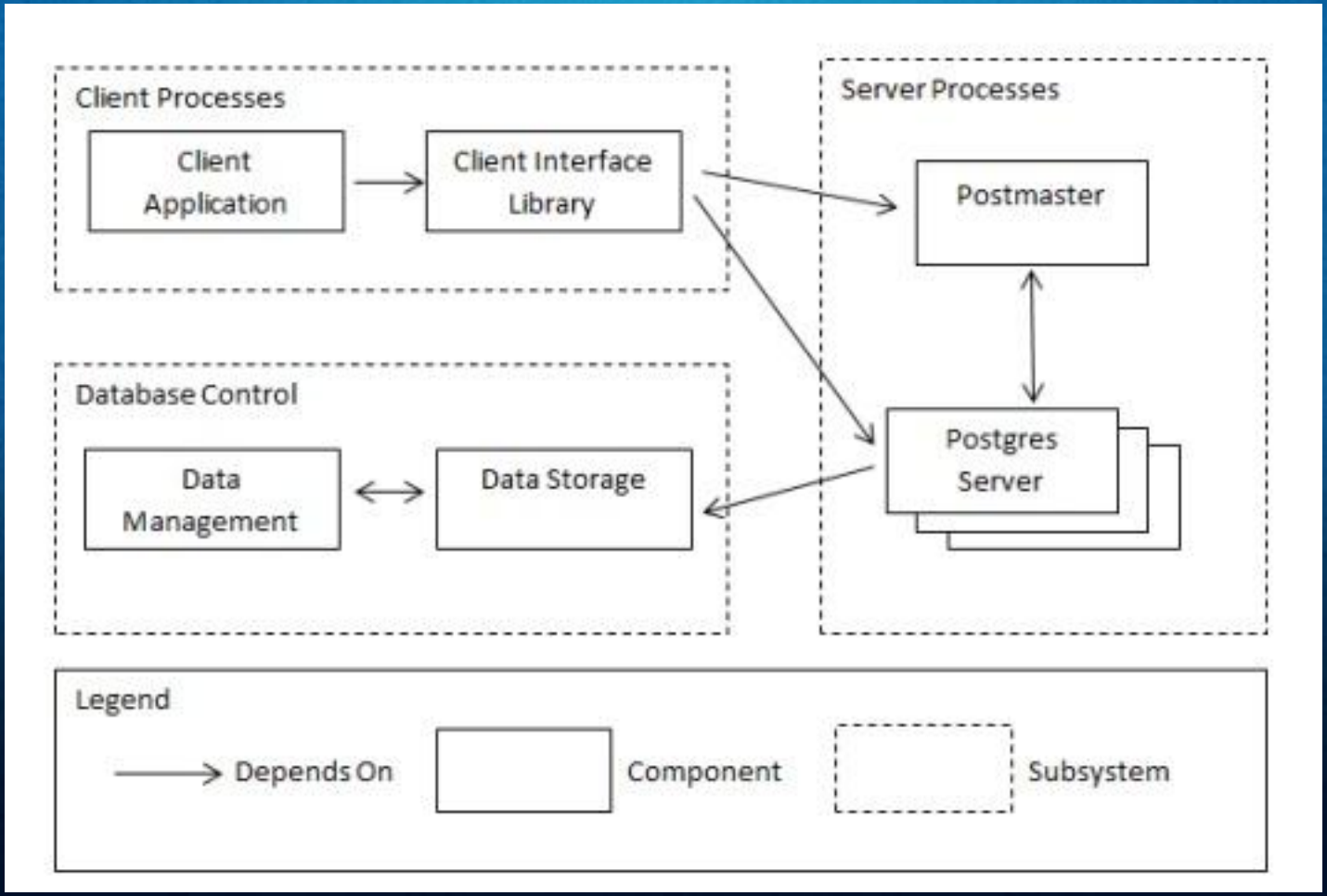

Diagram 1: Overall Conceptual Architecture of PostgreSQL

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.
- This maximizes cohesion by the server not having to parse different languages, but only understand SQL queries, which makes the whole system faster.

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.
- Once the process is created, it links the client and postgres process so that they no longer have to communicate through the postmaster.

General Architecture of Postgres Server

- Hybrid pipe and filter architecture.

- Each component references a shared repository of catalogs, rules and tables.
- Postgres server is passed an SQL query and it is incrementally transformed into result data.

Diagram 2: Conceptual Architecture of Back-end

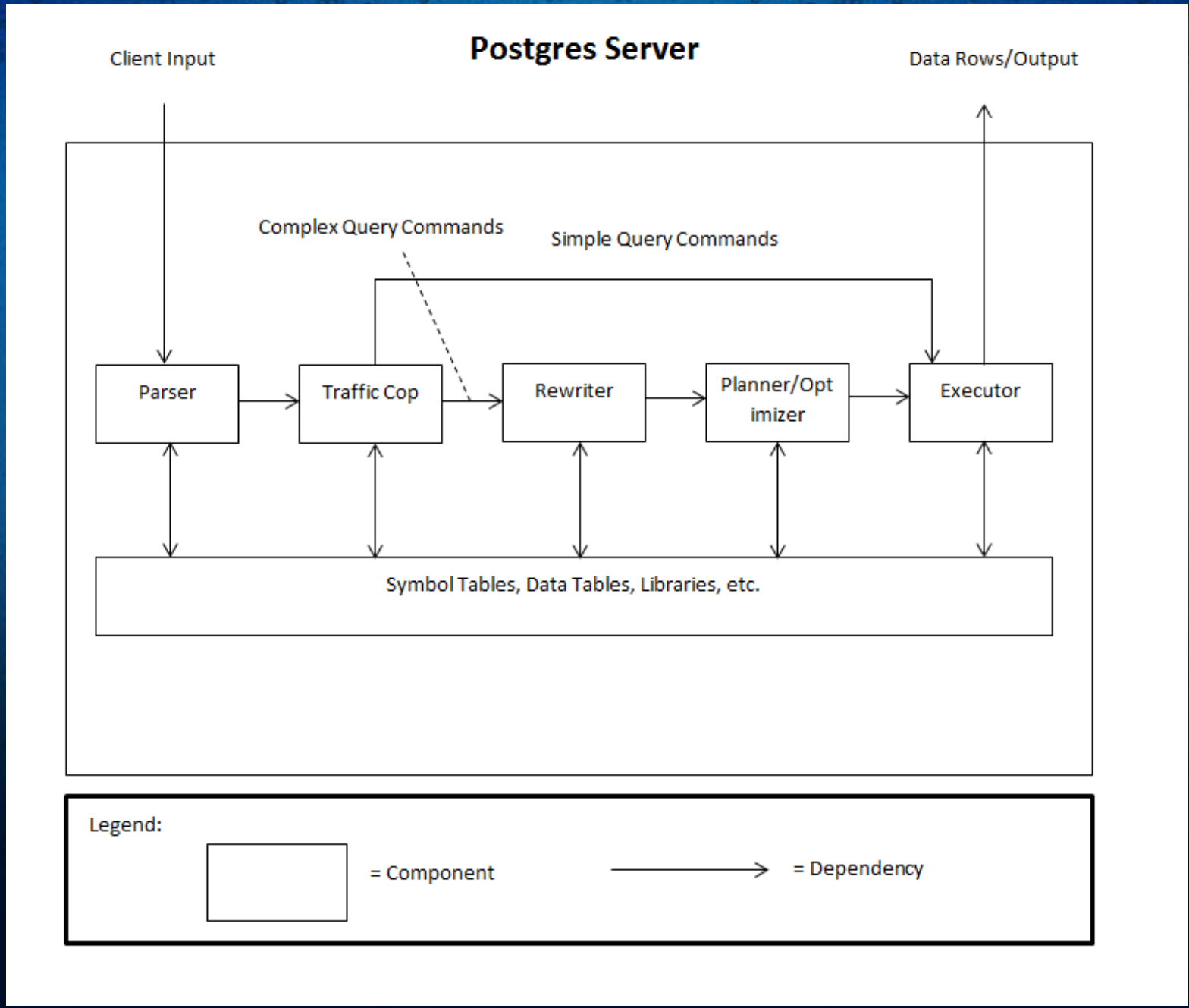

Diagram 2: Conceptual Architecture of Postgres server

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.
- Traffic cop sends simple commands to the executor and complex ones are sent to the planner / optimizer.

Planner / Optimizer

- 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.

Data Storage

- Data storage is handled through an Access and Storage subsystem.
- A Bootstrap subsystem is required the create the initial template database

Diagram 3: Database Control Dependencies

Data Management

- The database is maintained by several independent (and sometimes optional) subsystems initiated by the Postmaster upon construction including:
- The Statistics Collector
- The Auto-Vacuum
- The Background Writer
- The Memory Management System

Diagram 3: Database Control Dependencies

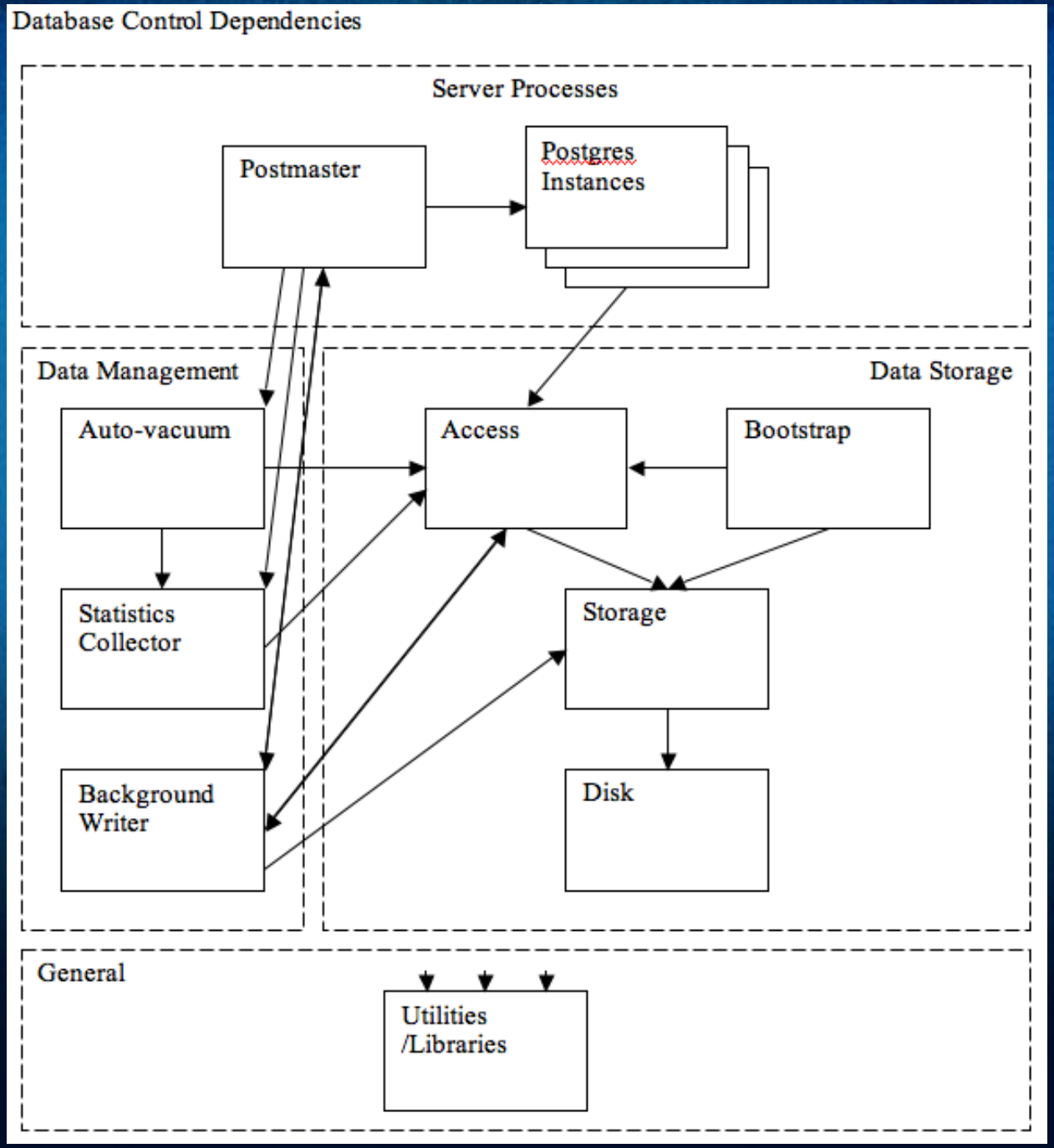

Diagram 3: Database Control Dependencies

Statistics Collector

- Records table/index accesses of the database, usage of user-defined functions, and commands being executed by server processes.
- Information is passed from the collector via temporary files to requesting processes.
- Processes submit relevant information to the collector periodically to keep it up to date.

Auto-Vacuum

- The Auto-Vaccum is a collection of processes that scan tables in the database(s) in order to release unused memory, update the statistics, and prevent loss of data.
- The Auto-Vacuum relies on data received from the Statistics Collector for proper table analysis.

Background Writer

- The Background Writer keeps the logs and backup information up to date.
- The Writer maintains Write Ahead Logs, which record all changes to the database since its last backup so that all data are secure.
- The standard output from every subsystem is passed to the Background Writer to maintain these logs.

Access

- The Access Subsystem is in charge of:
  - indexing
  - scanning
  - searching
  - compiling and returning data
- PostgrSQL server processes retrieve data using the Access Subsystem.
- Access subsytem can use a variety of indexing methods.

Storage

- Stored data is accessed through the Storage subsystem.
- In charge of maintaining a series of shared buffers.
- Allows for multiple accesses to the same tables using a multiversion concurency controle model (MVCC).
- Maintains table locks and insures data concurrency.

Bootstrap

- The Bootstrap subsystem allows users to start the database in bootstrap mode.
- Bootstrap mode does not allow for SQL queries.
- Bootstrap allows system catalogs to be created and filled from scratch, whereas ordinary SQL commands require the catalogs to exist already.
- Bootstrap is used by the installer to create the initial template database.

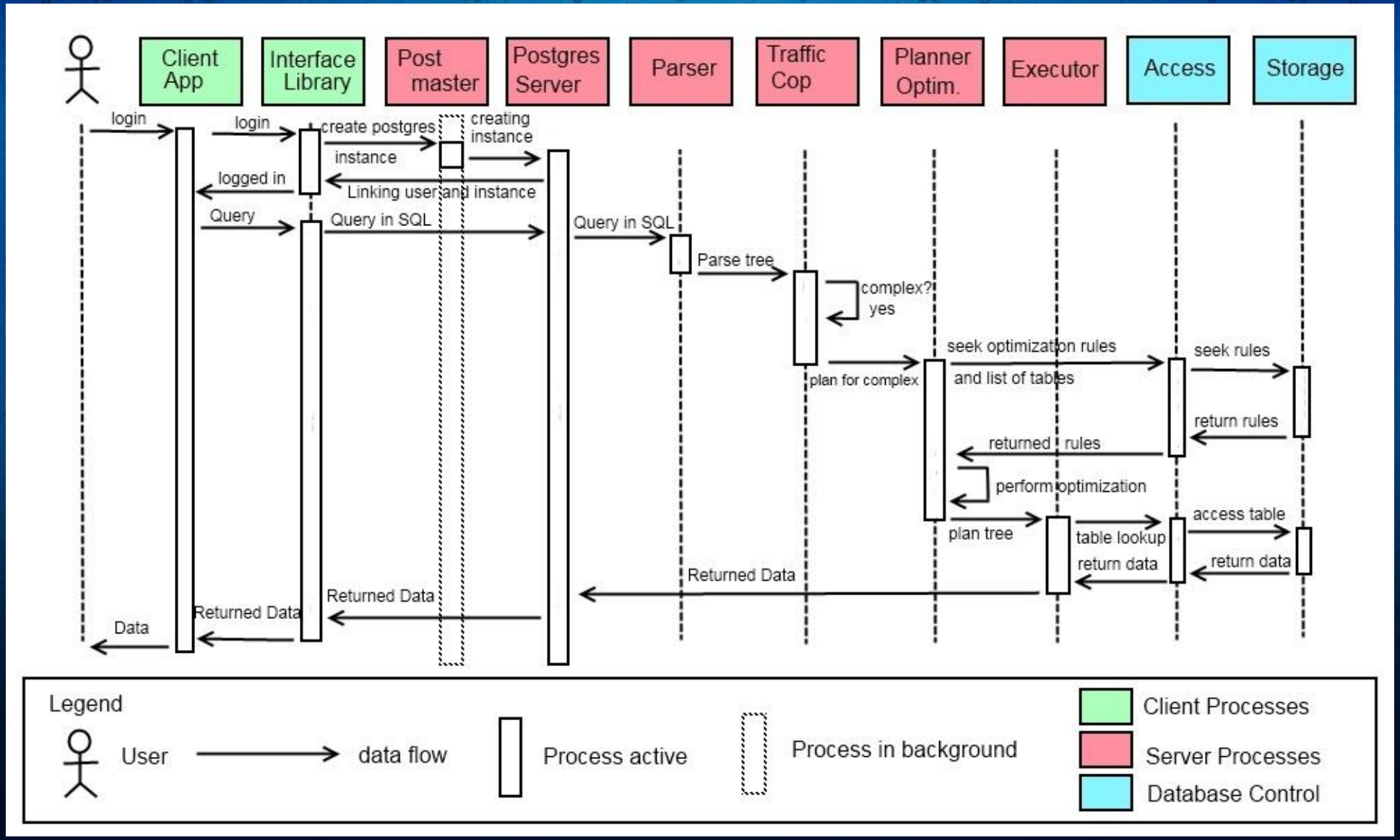

Diagram 4: Use case of login and complex query

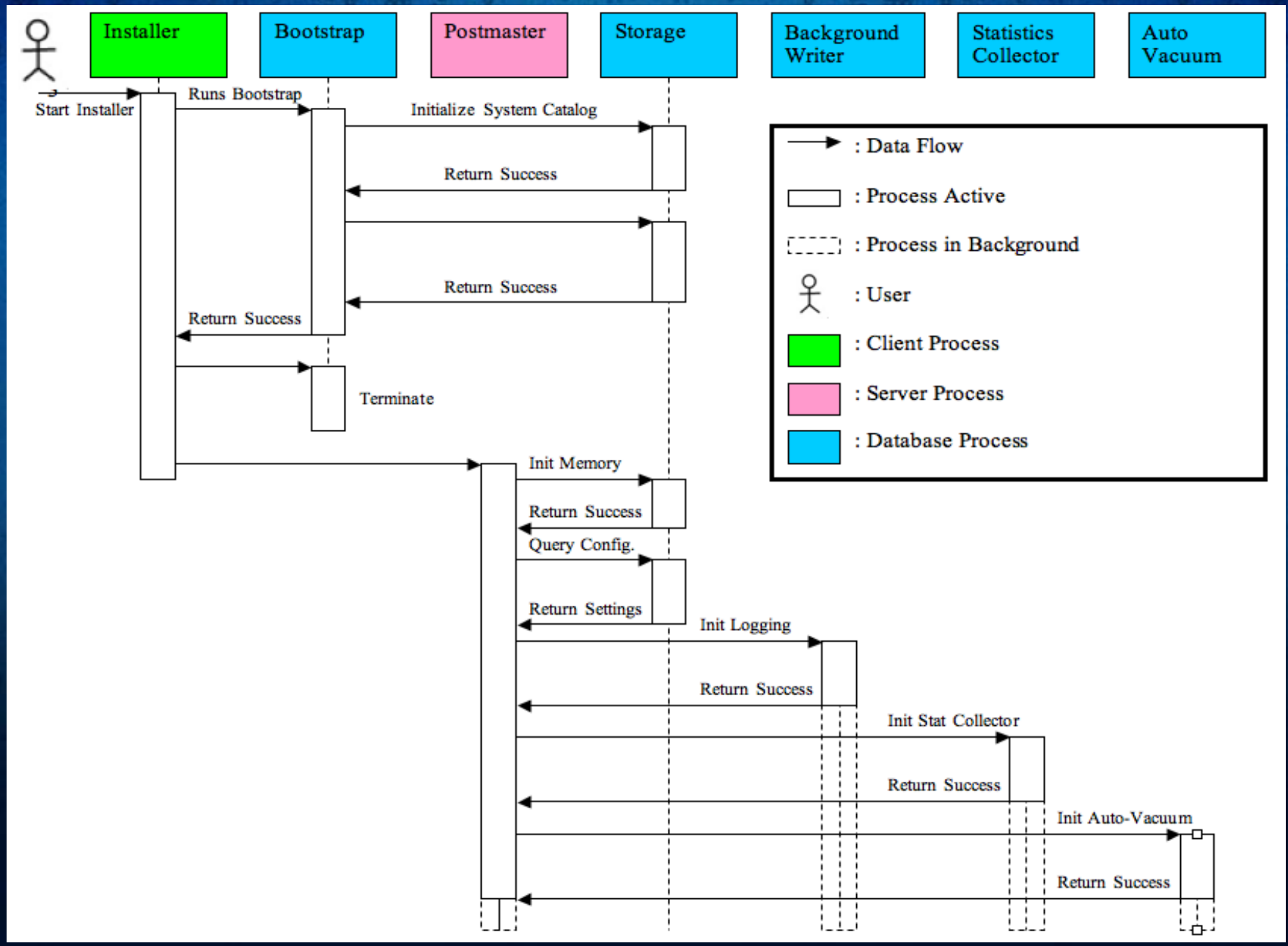

Diagram 5: Use case of creating a new database

Conclusion

- Overall architecture of PostgreSQL is object oriented and repository.
- Front-end is a client-server architecture from the client library to the Postgres server process and the postmaster uses implicit invocation.
- Postgres server employs a hybrid pipe and filter/repository architecture.
- The database control uses an object oriented architecture.
- In the future we will be able to study the source code to determine the code-level dependencies, allowing us to form a concrete architecture.