PostgreSQL Architecture

Ágnes Kovács

Budapest, 2015-01-20

Agenda

- Introduction
- Overview of architecture
- Process structure
- Shared memory
- Concurrency handling
- The Optimizer

Introduction

What is PostgreSQL?

- open source
- object-relational database system
- runs on Linux, UNIX (AIX, BSD, HP-UX, SGI IRIX, Mac OS X, Solaris, Tru64), and Windows
- ACID compliant
- supported data types: INTEGER, NUMERIC, BOOLEAN, CHAR, VARCHAR, DATE, INTERVAL, TIMESTAMP and binary large objects
- native programming interfaces for C/C++, Java, .Net, Perl, Python, Ruby, Tcl, ODBC

PostgreSQL in numbers

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

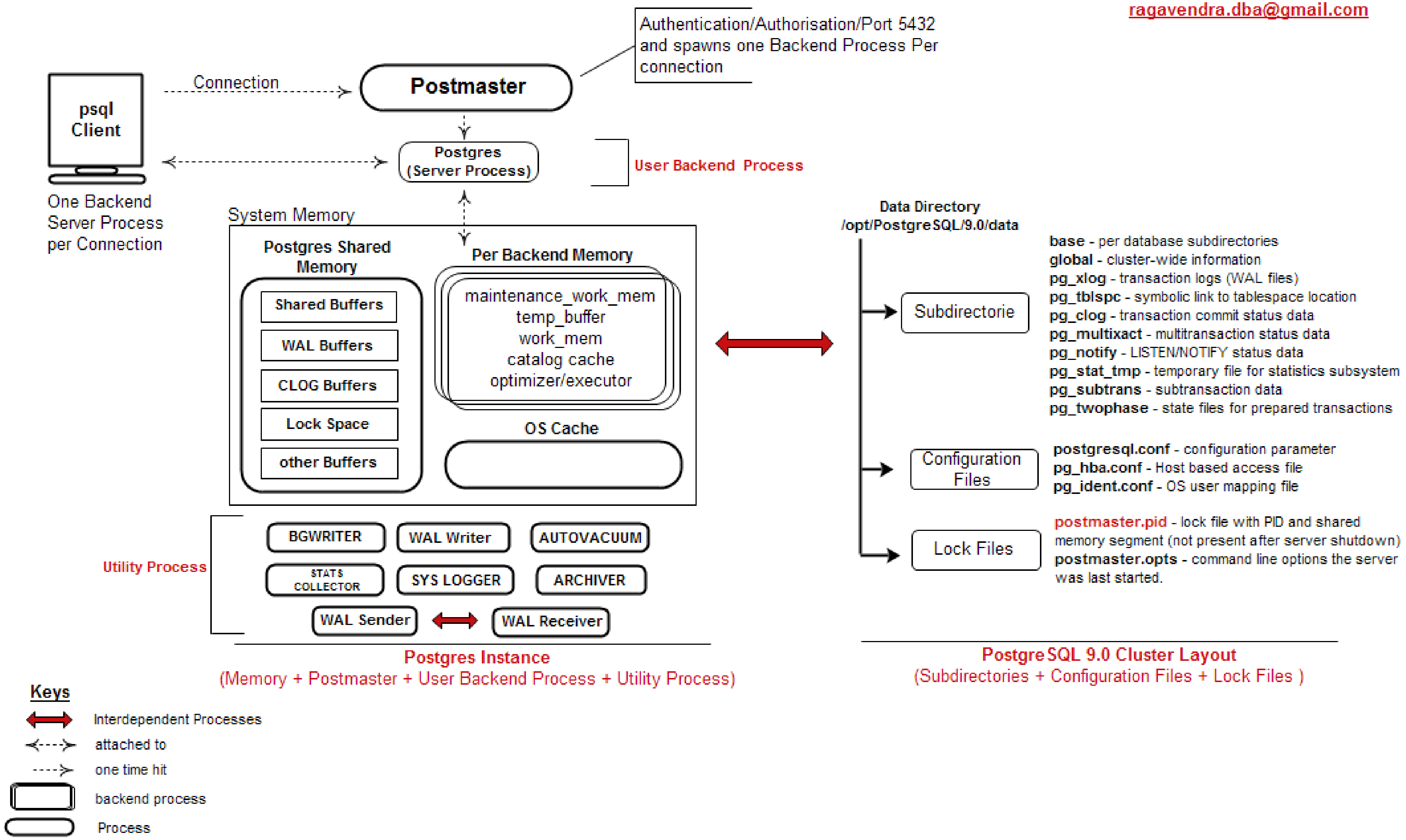
What PostgreSQL can do?

- Multi-Version Concurrency Control (MVCC)
- Point in time recovery
- Tablespaces
- Asynchronous replication
- Nested transactions (savepoints)
- Online/hot backups
- Query planner/optimizer
- Write Ahead Logging
- International character sets, multibyte character encodings, Unicode, and it is locale-aware for sorting, case-sensitivity, and formatting

Overview of architecture

PostgreSQL 9.0 Architecture

Raghavendra
www.raghavt.blogspot.com
ragavendra.dba@gmail.com



Authentication/Authorisation/Port 5432 and spawns one Backend Process Per connection

psql Client

Connection

Postmaster

Postgres (Server Process)

User Backend Process

One Backend Server Process per Connection

System Memory

Postgres Shared Memory

Shared Buffers

WAL Buffers

CLOG Buffers

Lock Space

other Buffers

Per Backend Memory

maintenance_work_mem
temp_buffer
work_mem
catalog cache
optimizer/executor

OS Cache

Utility Process

BGWRITER

WAL Writer

AUTOVACUUM

STATS COLLECTOR

SYS LOGGER

ARCHIVER

WAL Sender

WAL Receiver

Postgres Instance
(Memory + Postmaster + User Backend Process + Utility Process)

Data Directory
/opt/PostgreSQL/9.0/data

Subdirectorie

base - per database subdirectories
global - cluster-wide information
pg_xlog - transaction logs (WAL files)
pg_tblspc - symbolic link to tablespace location
pg_clog - transaction commit status data
pg_multixact - multitransaction status data
pg_notify - LISTEN/NOTIFY status data
pg_stat_tmp - temporary file for statistics subsystem
pg_subtrans - subtransaction data
pg_twophase - state files for prepared transactions

Configuration Files

postgresql.conf - configuration parameter
pg_hba.conf - Host based access file
pg_ident.conf - OS user mapping file

Lock Files

postmaster.pid - lock file with PID and shared memory segment (not present after server shutdown)
postmaster.opts - command line options the server was last started.

PostgreSQL 9.0 Cluster Layout
(Subdirectories + Configuration Files + Lock Files)

Keys

⟷	Interdependent Processes
<--->	attached to
--->	one time hit
▭	backend process
⬭	Process

Database file layout

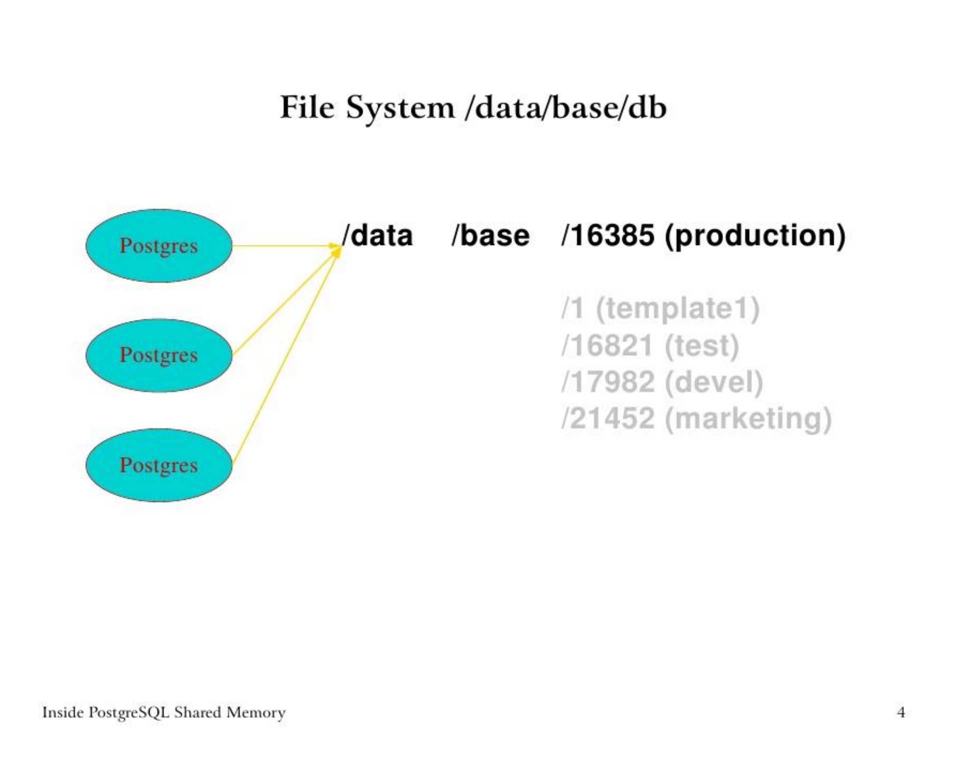
- PGDATA – base directory for the Database Server: traditionally it contains configuration and data files + data directory
- example location: /var/lib/pgsql/data
- Multiple clusters, managed by different server instances, can exist on the same machine
- configuration files and pid file location can be configured any where, it can reside under PGDATA also

Subdirectories within PGDATA

Item	Description
PG_VERSION	A file containing the major version number of PostgreSQL
base	Subdirectory containing per-database subdirectories
global	Subdirectory containing cluster-wide tables, such as pg_database
pg_clog	Subdirectory containing transaction commit status data
pg_dynshmem	Subdirectory containing files used by the dynamic shared memory subsystem
pg_logical	Subdirectory containing status data for logical decoding
pg_multixact	Subdirectory containing multitransaction status data (used for shared row locks)
pg_notify	Subdirectory containing LISTEN/NOTIFY status data
pg_replslot	Subdirectory containing replication slot data
pg_serial	Subdirectory containing information about committed serializable transactions
pg_snapshots	Subdirectory containing exported snapshots
pg_stat	Subdirectory containing permanent files for the statistics subsystem
pg_stat_tmp	Subdirectory containing temporary files for the statistics subsystem
pg_subtrans	Subdirectory containing subtransaction status data
pg_tblspc	Subdirectory containing symbolic links to tablespaces
pg_twophase	Subdirectory containing state files for prepared transactions
pg_xlog	Subdirectory containing WAL (Write Ahead Log) files

The /base subdirectory

- contains the user database files
- subdirectory names are the database OIDs



File System /data/base/db

Postgres

Postgres

Postgres

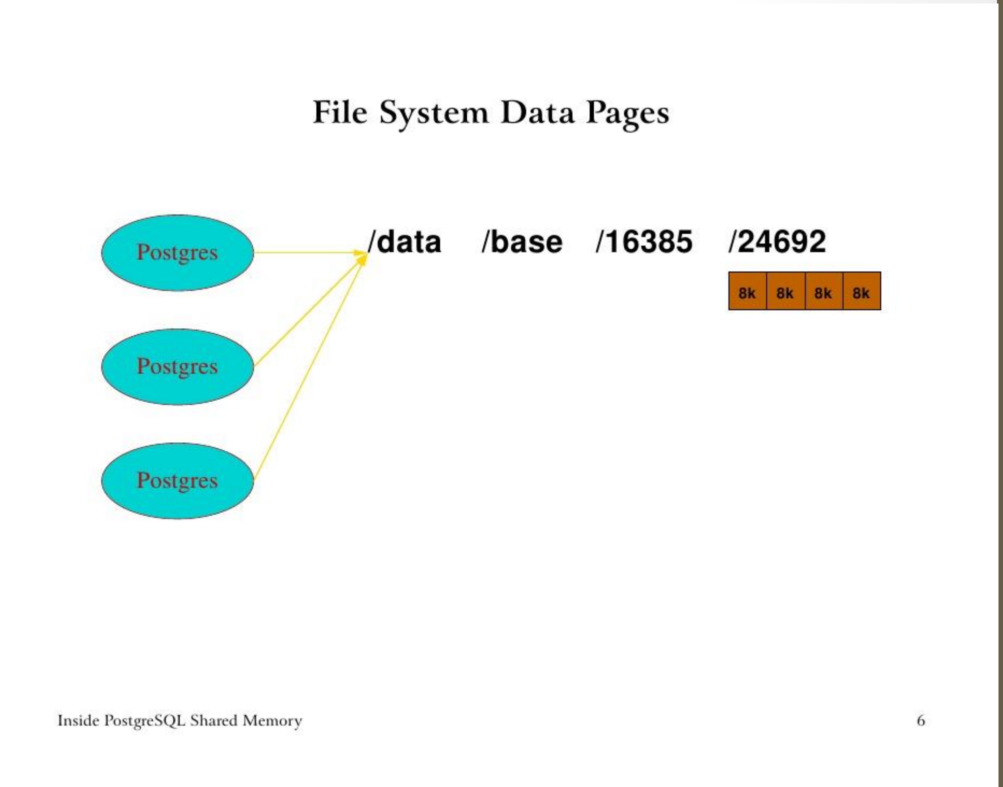
/data /base /16385 (production)

/1 (template1)
/16821 (test)
/17982 (devel)
/21452 (marketing)

Inside PostgreSQL Shared Memory 4

Data Pages

- pages are located under the database subdirectories
- page default size: 8k
    - additional sizes:4k and 16k but needs compilation of postgresql
- for general purpose 8k is best practice



File System Data Pages

Inside PostgreSQL Shared Memory

6

Important user accessible files

- PGVERSION: major version number of installation
- **postgresql.conf**: main configuration file for PostgreSQL installation
- **pg_hba.conf**: configures the client authentication method
- pg_ident.conf: configures OS and PostgreSQL authentication name mapping
- postmaster.opts: default command line options for the postmaster
- postmaster.pid: PID of the postmaster and identification the main directory

Write Ahead Logs I.

- located under /pg_xlog
- REDO logs of PostgreSQL database Server
- per default it is used during crash recovery
- related main parameters are
  - *wal_level*
  - *archive_mode, archive_command*

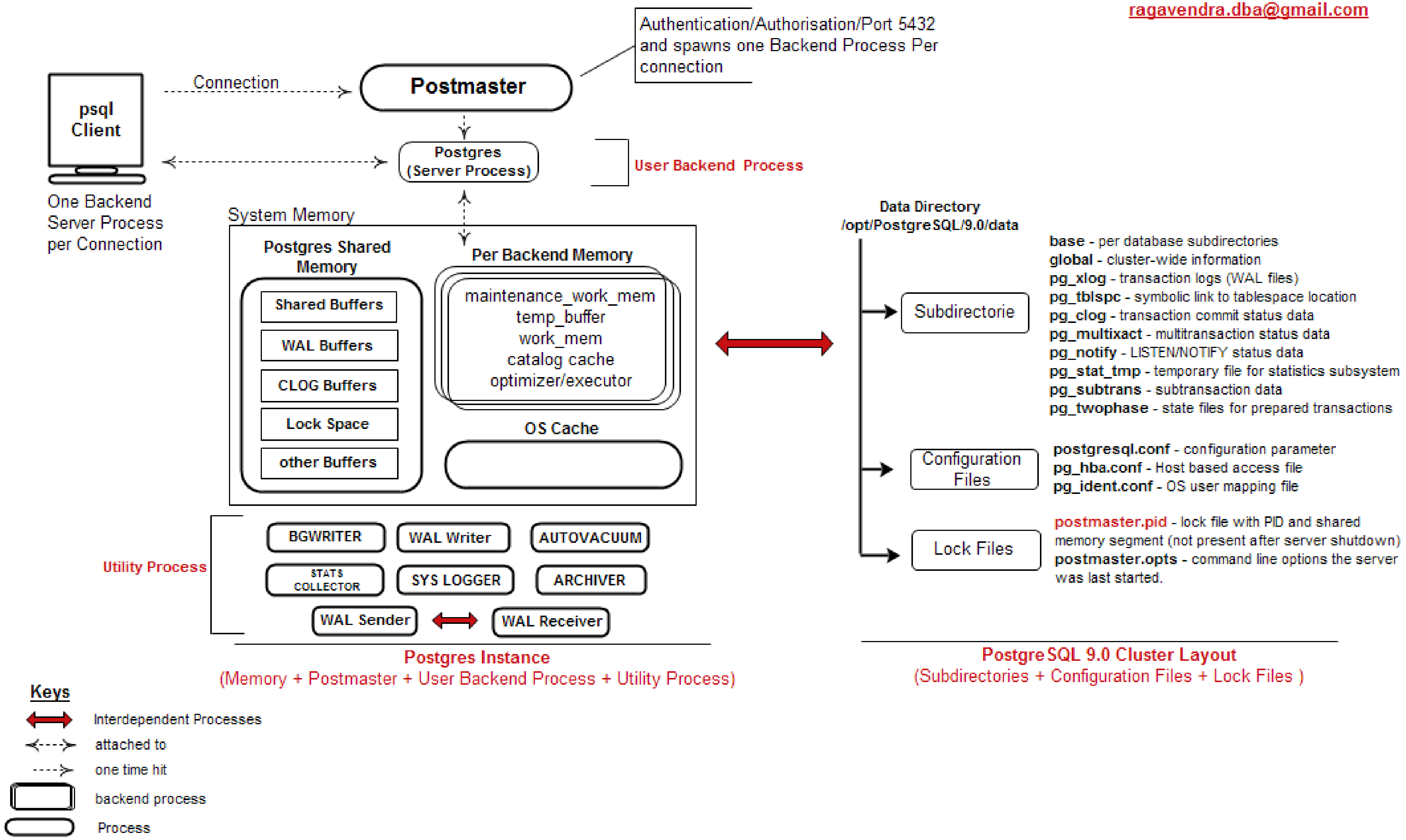
Write Ahead Logs II.

- *wal_level*: defines what to log into the WAL files
  - minimal (default): transaction logging is skiped for bulk operations like `CREATE TABLE AS`, `CREATE INDEX`, `CLUSTER`, `COPY` etc.
    - enough to recover after a crash or immediate shutdown
  - archive: needed for archiving WAL files
  - hot_standby: enable to open standby read only
  - logical: supports logical decoding
- *archive_mode*: default is false, if true WAL files are archived with the command given at *archive_command*

Process structure

PostgreSQL 9.0 Architecture

Raghavendra
www.raghavt.blogspot.com
ragavendra.dba@gmail.com



Authentication/Authorisation/Port 5432 and spawns one Backend Process Per connection

psql Client

Connection

Postmaster

Postgres (Server Process)

User Backend Process

One Backend Server Process per Connection

System Memory

Postgres Shared Memory

- Shared Buffers
- WAL Buffers
- CLOG Buffers
- Lock Space
- other Buffers

Per Backend Memory

- maintenance_work_mem
- temp_buffer
- work_mem
- catalog cache
- optimizer/executor

OS Cache

Utility Process

BGWRITER, WAL Writer, AUTOVACUUM, STATS COLLECTOR, SYS LOGGER, ARCHIVER, WAL Sender, WAL Receiver

Postgres Instance
(Memory + Postmaster + User Backend Process + Utility Process)

Data Directory
/opt/PostgreSQL/9.0/data

Subdirectorie

base - per database subdirectories
global - cluster-wide information
pg_xlog - transaction logs (WAL files)
pg_tblspc - symbolic link to tablespace location
pg_clog - transaction commit status data
pg_multixact - multitransaction status data
pg_notify - LISTEN/NOTIFY status data
pg_stat_tmp - temporary file for statistics subsystem
pg_subtrans - subtransaction data
pg_twophase - state files for prepared transactions

Configuration Files

postgresql.conf - configuration parameter
pg_hba.conf - Host based access file
pg_ident.conf - OS user mapping file

Lock Files

postmaster.pid - lock file with PID and shared memory segment (not present after server shutdown)
postmaster.opts - command line options the server was last started.

PostgreSQL 9.0 Cluster Layout
(Subdirectories + Configuration Files + Lock Files)

Keys

- Interdependent Processes
- attached to
- one time hit
- backend process
- Process

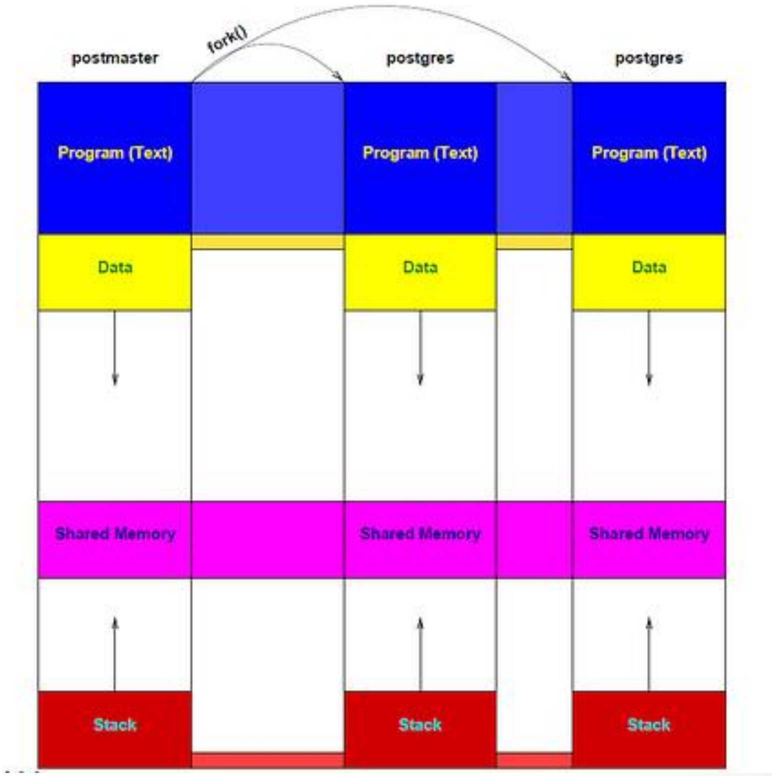
Postmaster

- main PostgreSQL program

```
postgres@agideb:~$ pg_ctl status
pg_ctl: server is running (PID: 2971)
/usr/lib/postgresql/9.1/bin/postgres "-D"
"/var/lib/postgresql/9.1/main" "-c"
"config_file=/etc/postgresql/9.1/main/postgresql.conf"
```

- postmaster is listening and if user connection comes in it forks postgres server processes
- postgres server process is the copy of postmaster

Shared Memory Creation



Additional important background processes I.

- Statistics Collector
- Background Writer
  - writes dirty pages to disk
  - runs repeatedly (time is defined by multiple parameters and actual statistics)
- WAL Writer

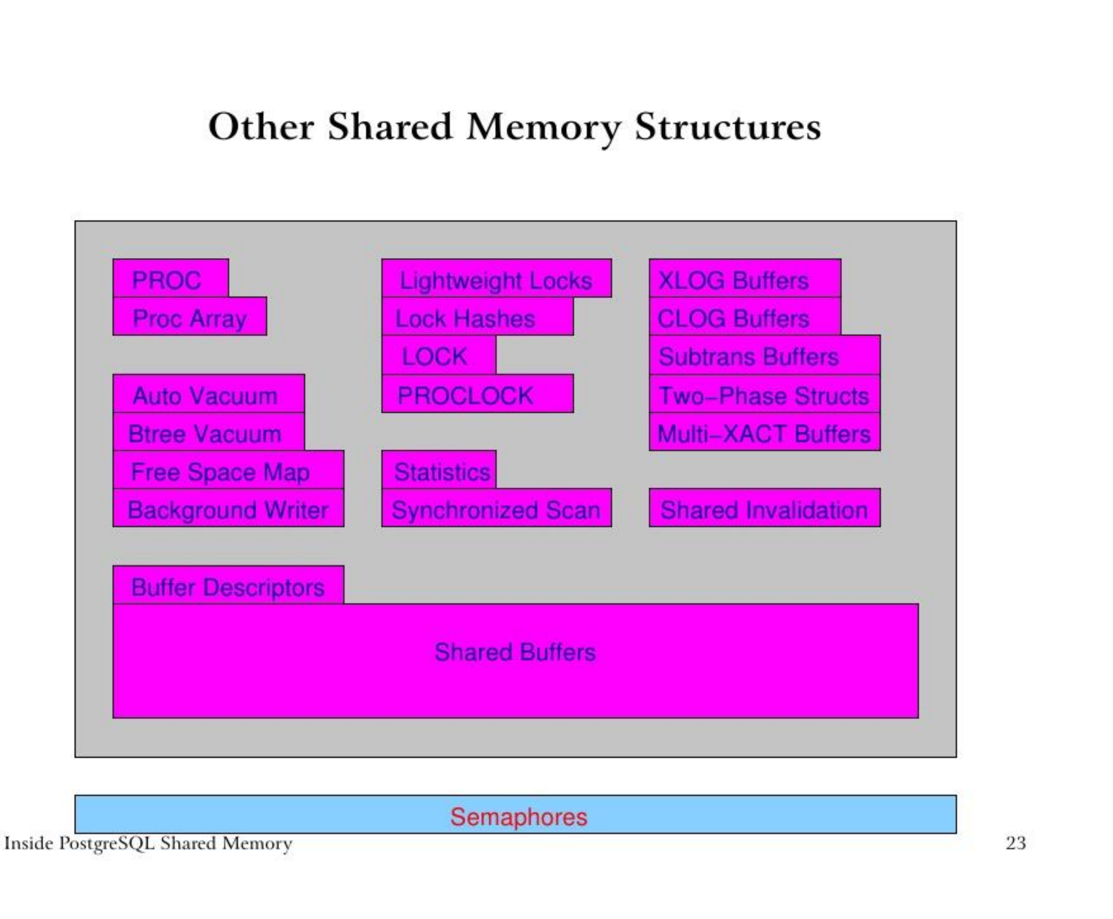
Additional important background processes II.

- Auto Vacuum daemon
  - optional but highly recommended
  - automates `VACUUM` and `ANALYZE` commands
  - multiple processes:
    a. auto vacuum launcher: persistent
    b. auto vacuum workers: started by launcher for all databases
- What is `VACUUM`?
  - recover or reuse disk space occupied by deleted or updated rows
  - update data statistics
  - update visibility maps – speeds up index only scans
  - protects against transactional ID Wraparound

Shared Memory

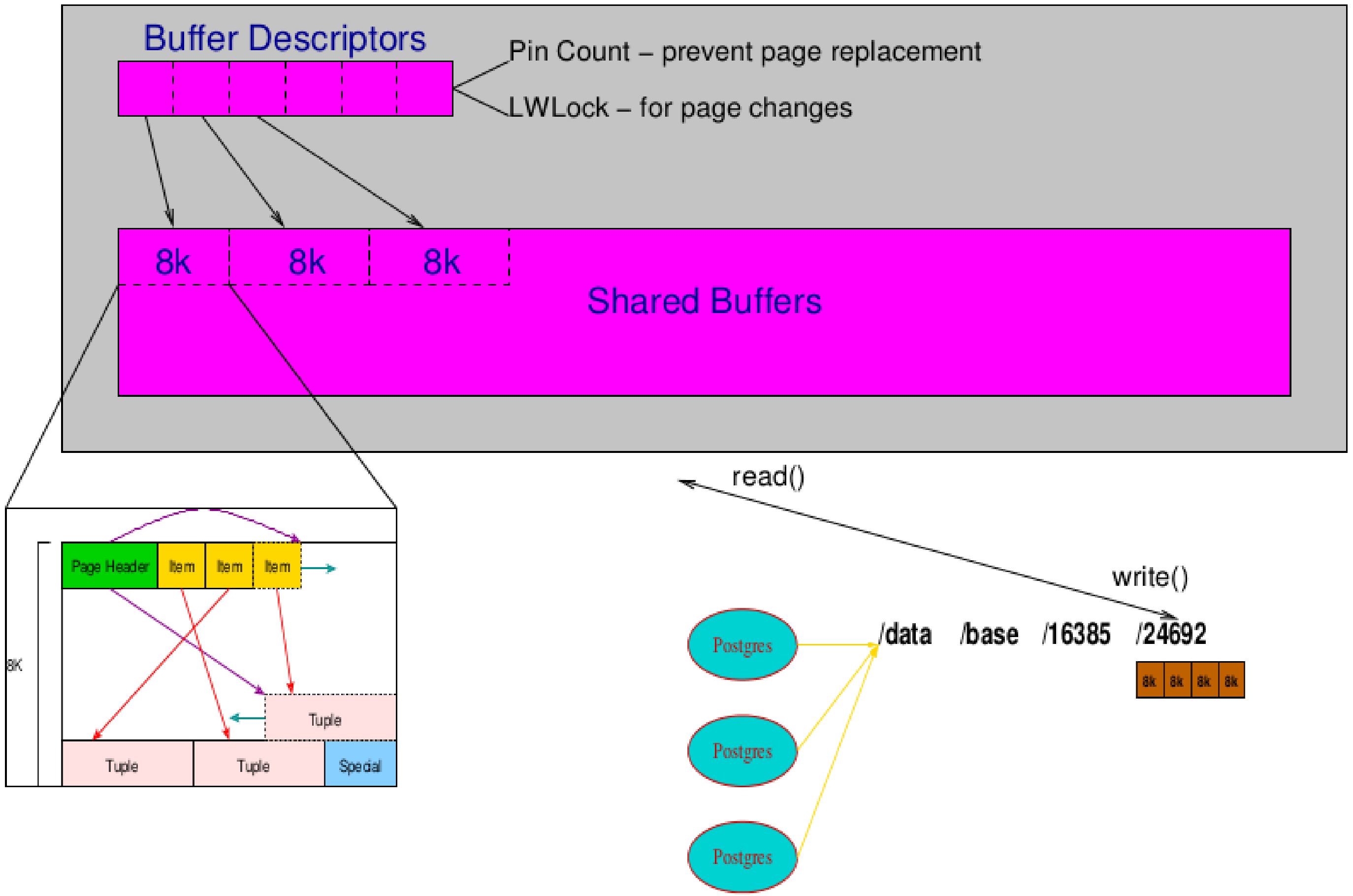
Why do we need shared memory in PostgreSQL?

- to mainly enable communication between postmaster and postgres server processes
  - child processes cannot propagate information
  - shared memory is available to share the same information to all processes
- to cache pages

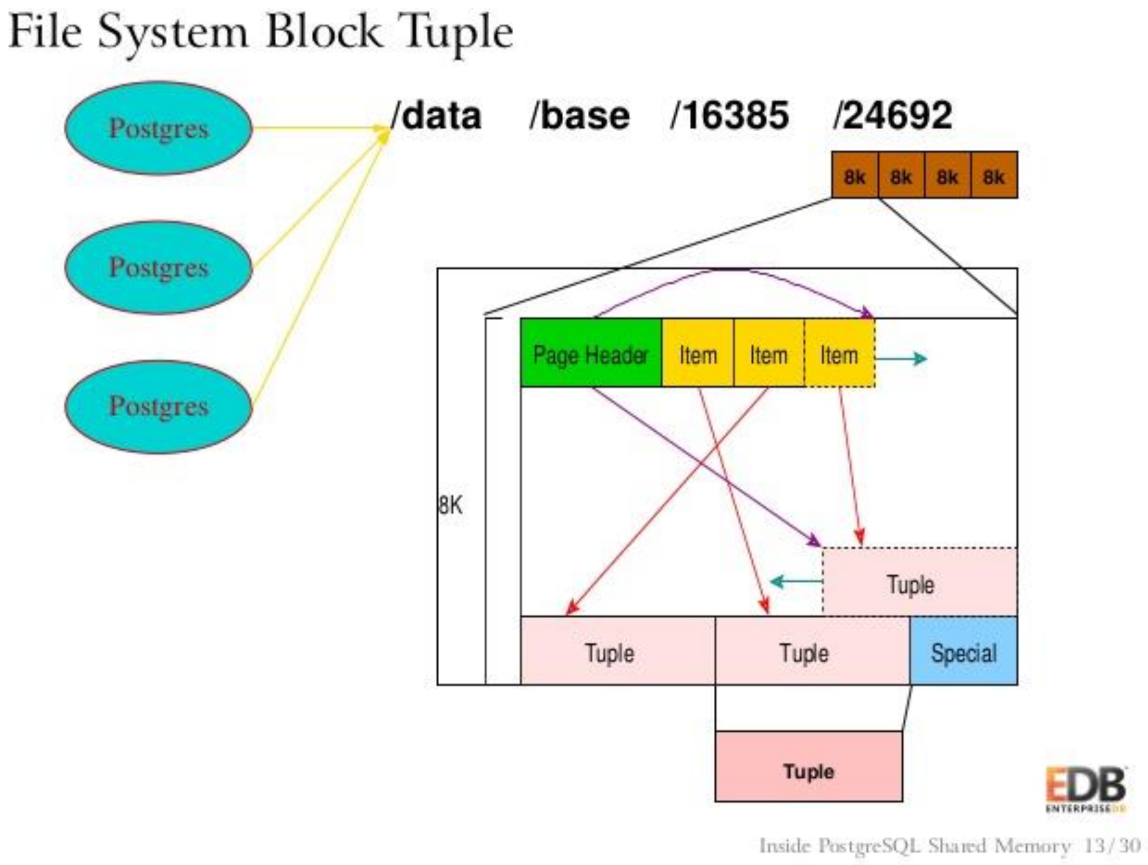


Other Shared Memory Structures

Inside PostgreSQL Shared Memory 23


Buffer Descriptors
Pin Count – prevent page replacement
LWLock – for page changes
8k
8k
8k
Shared Buffers
read()
write()
Page Header
Item
Item
Item
8K
Tuple
Tuple
Tuple
Special
Postgres
Postgres
Postgres
/data /base /16385 /24692
8k
8k
8k
8k

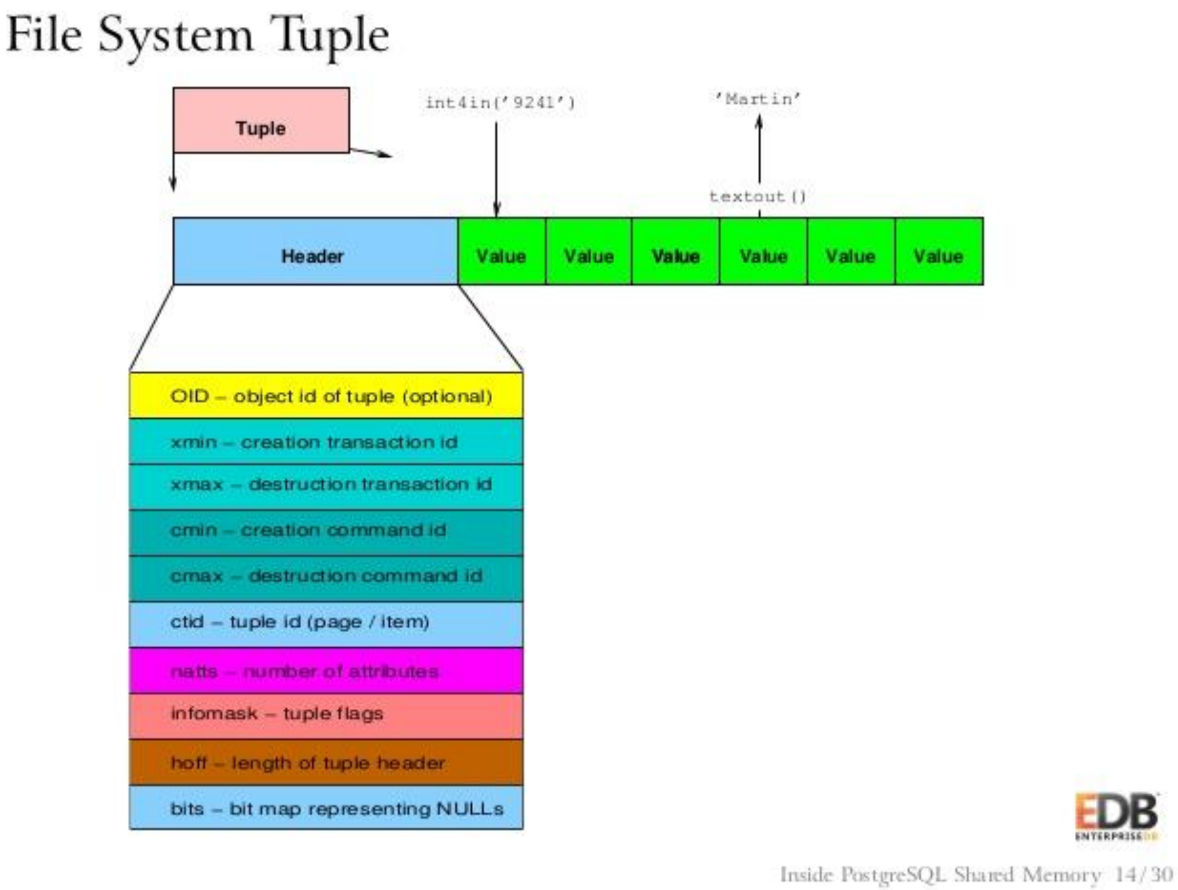
Structure of a block tuple



File System Block Tuple

Inside PostgreSQL Shared Memory 13 / 30

Structure of a single file system tuple



File System Tuple

Tuple

int4in('9241')

'Martin'

textout()

Header

Value | Value | Value | Value | Value | Value

OID – object id of tuple (optional)

xmin – creation transaction id

xmax – destruction transaction id

cmin – creation command id

cmax – destruction command id

ctid – tuple id (page / item)

natts – number of attributes

infomask – tuple flags

hoff – length of tuple header

bits – bit map representing NULLs

Inside PostgreSQL Shared Memory 14/30

Handling concurrency

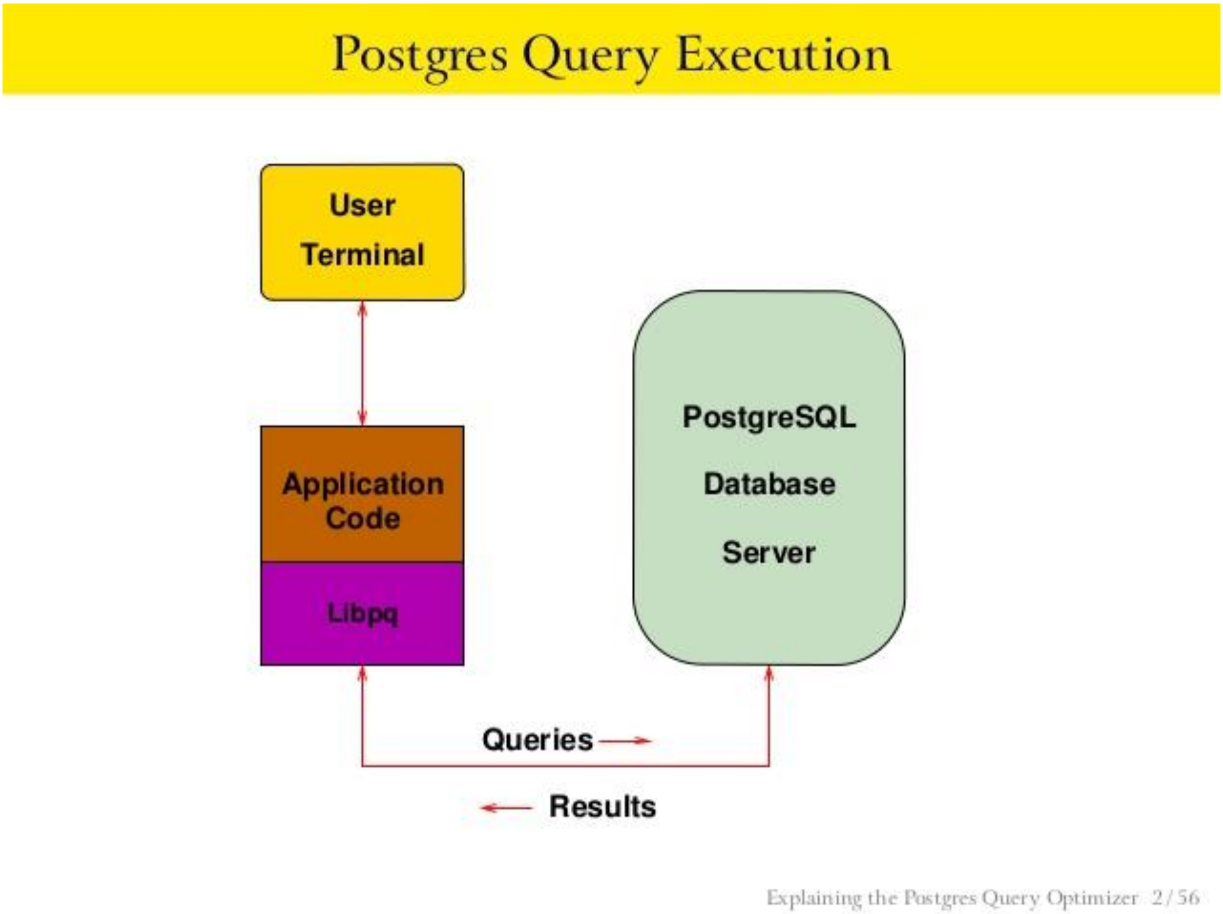
Multi-Version Concurrency Control (MVCC)

- Readers do not block writers, writers do not block readers
- PostgreSQL guarentees this even with the strictest isolation level

Standard SQL Transaction Isolation Levels available in PostgreSQL

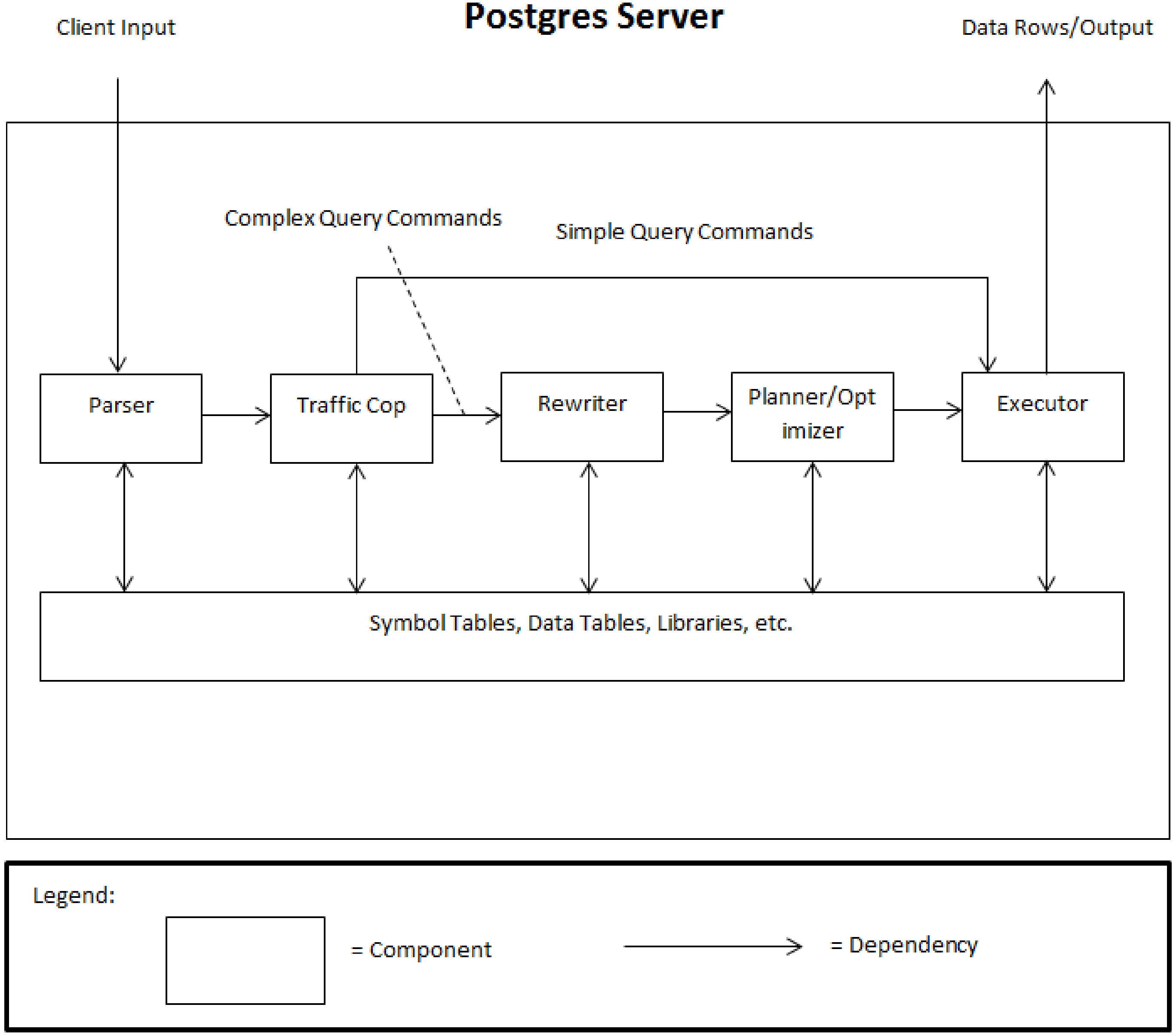
Isolation Level	Dirty Read	Nonrepeatable Read	Phantom Read
Read uncommitted	Possible	Possible	Possible
Read committed	Not possible	Possible	Possible
Repeatable read	Not possible	Not possible	Possible
Serializable	Not possible	Not possible	Not possible

The Optimizer



Postgres Query Execution

Postgres Server



Decisions taken by the optimizer

- Scan Method
  - Sequential Scan
  - Bitmap Index Scan
  - Index Scan
- Join Method
  - Nested Loop
  - Hash Join
  - Merge Join
- Join Order

Optimizer statistics

- distribution of data:
  - 100 most common values
  - histograms with 100 buckets
  - granularity can be changed to have more data to calculate distribution
  - `ALTER TABLE`
- statistic collection cannot be turned off
- statistics cannot be backed up individually for an object

Sources

- http://momjian.us/main/presentations/overview.html
- http://www.postgresql.org/
- http://raghavt.blogspot.hu/2011/04/postgresql-90-architecture.html