EnterpriseDB™

Introduction to PostgreSQL Administration

Instructor: Raghavendra

Dates: 18th July to 23rd July

EnterpriseDB™

Module-1

Introduction & Architecture

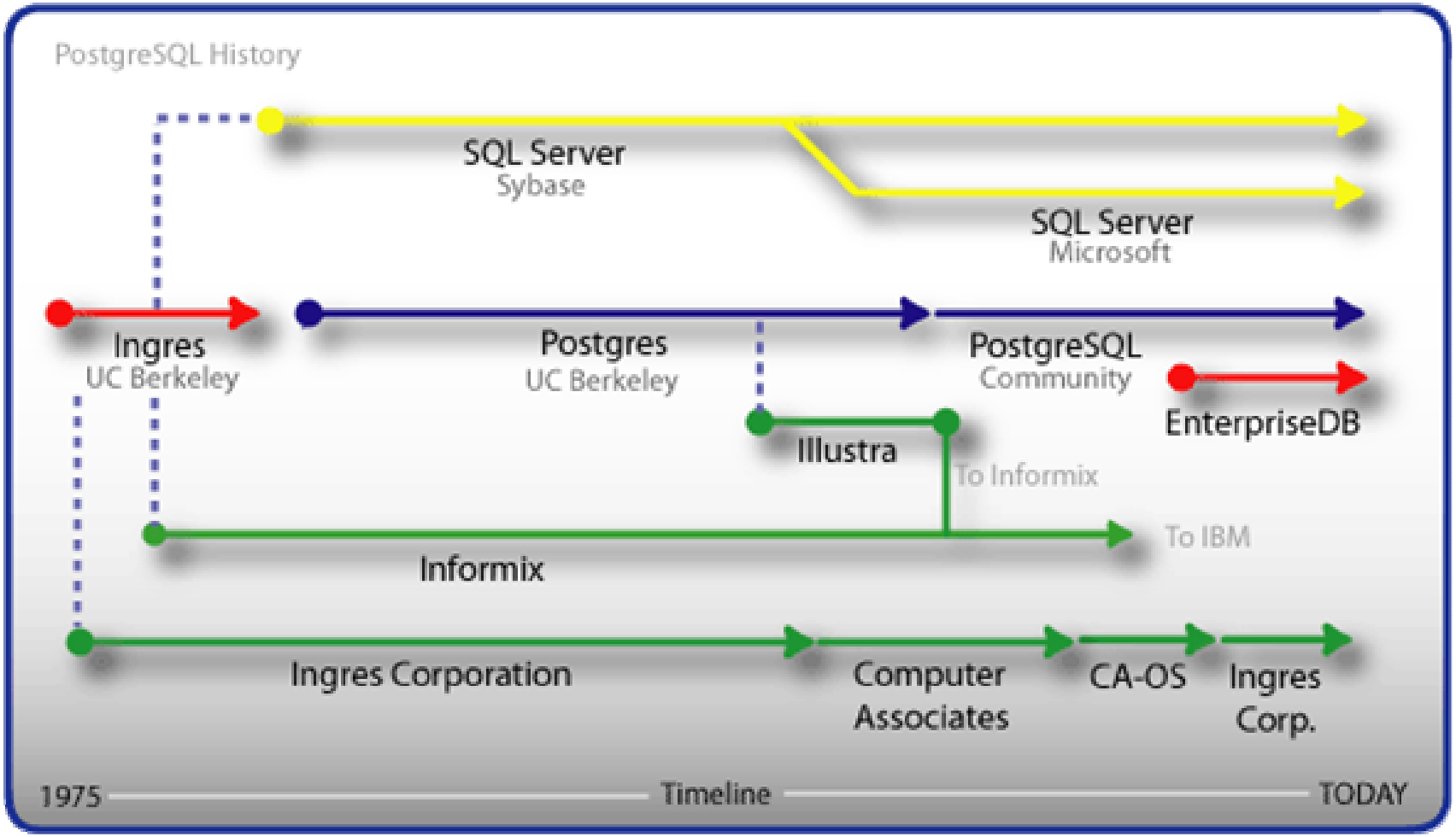
Objectives

- In this module you will learn:
  - History of PostgreSQL
  - Major Features
  - Multi-Version Concurrency Control (MVCC)
  - Write-Ahead Logging
  - Architectural Overview
  - Limits
  - Postgres Terminology

History of PostgreSQL

- The world's most advanced open source database
- Designed for extensibility and customization
- ANSI/ISO compliant SQL support
- Actively developed for more than 20 years
  - University POSTGRES (1986-1993)
  - Postgres95 (1994-1995)
  - PostgreSQL (1996-2009)
- Active global support community
  - Support Mailing Lists
    - http://www.postgresql.org/community/lists/
  - Support Forums
    - http://www.enterprisedb.com/products/postgres_plus.do

Postgres Lineage



EnterpriseDB™

Major Features

- Portable
  - Written in ANSI C
  - Supports Windows, Linux, Mac and major UNIX platforms
- Reliable
  - ACID Compliant
  - Supports Transactions
  - Supports Savepoints
  - Uses Write Ahead Logging
- Scalable
  - Uses Multiversion Concurrency Control
  - Uses Row-Level Locking
  - Supports Table Partitioning
  - Supports Tablespaces

Major Features (continued)

- Secure
  - Employs Host-Based Access Control
  - Provides Object-Level Permissions
  - Supports Logging
  - SSL
- Available
  - Replication Support
  - Support for High Availability
- Advanced
  - Full Text Search
  - Supports Triggers & Functions
  - Supports Custom Procedural Languages
    - PL/pgSQL, PL/Perl, PL/TCL, PL/PHP, …
  - Supports Hot-Backup and Point-in-Time Recovery
  - Warm Standby

EnterpriseDB™

Multi-Version Concurrency Control (MVCC)

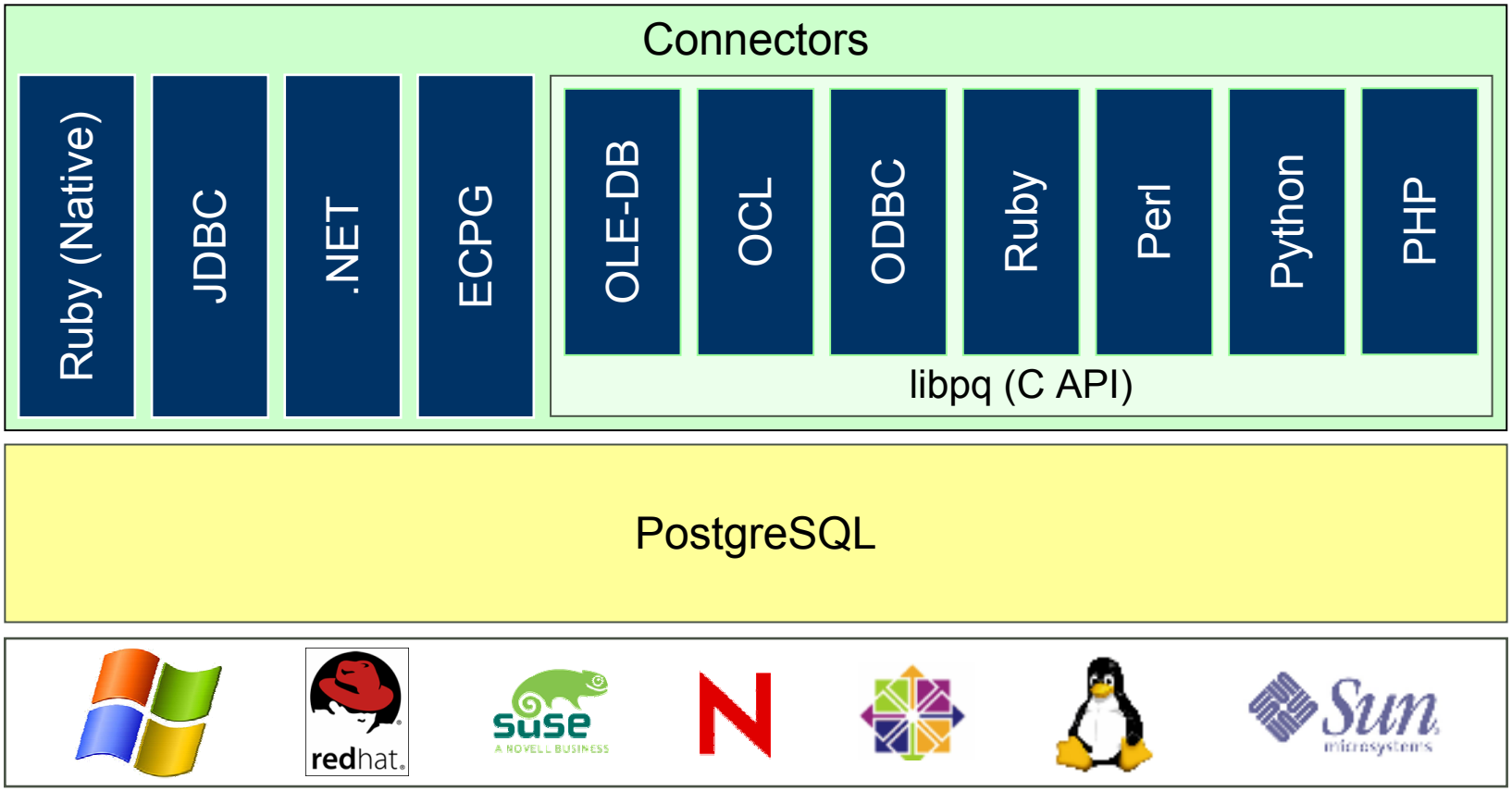
Maintain data consistency internally

- While querying a database each transaction sees a snapshot of data (a database version) as it was some time ago
- Prevent transaction from viewing inconsistent data
- Provides transaction isolation in concurrent transactions
- Readers does not block writers and writers does not block readers

Write Ahead Logs (WAL)

- Write Ahead Logging
  - Makes a record of each insert/update/delete before it actually takes place
  - System does not consider data 'safe' until log is written to disk
  - Provides recovery in case of system crash or failure
- Characteristics
  - Similar to Oracle REDO logs (no separate undo)
    - *Database grows with undo information*
    - *Logs never fill up on long transactions*
  - Fast, single-phase Recovery
  - Log currently needs to log whole blocks after a checkpoint

Architectural Overview



EnterpriseDB™

Database Limitations

- Limitations are generally defined by
  - Operating System Limits
  - Compile-Time Parameters
  - Data Type Usage

- General Database Limitations

Limit	Value
Maximum Database Size	Unlimited
Maximum Table Size	32 TB
Maximum Row Size	1.6 TB
Maximum Field Size	1 GB
Maximum Rows / Table	Unlimited
Maximum Columns / Table	250-1600
Maximum Indexes / Table	Unlimited

PostgreSQL Terminology

- PostgreSQL was designed in academia
  - Objects are defined in academic terms
    - Terminology based on relational calculus / algebra
- Common Database Object Names

Industry Term	Postgres Term
Table or Index	Relation
Row	Tuple
Column	Attribute

- Storage Object Names

Industry Term	Postgres Term
Data Block	Page (when block is on disk)
Page	Buffer (when block is in memory)

Summary

- In this module you learned:
  - History of PostgreSQL
  - Major Features
  - Multi-Version Concurrency Control (MVCC)
  - Write-Ahead Logging
  - Architectural Overview
  - Limits
  - PostgreSQL Terminology

EnterpriseDB™

Module 2

PostgreSQL System Architecture

Objectives

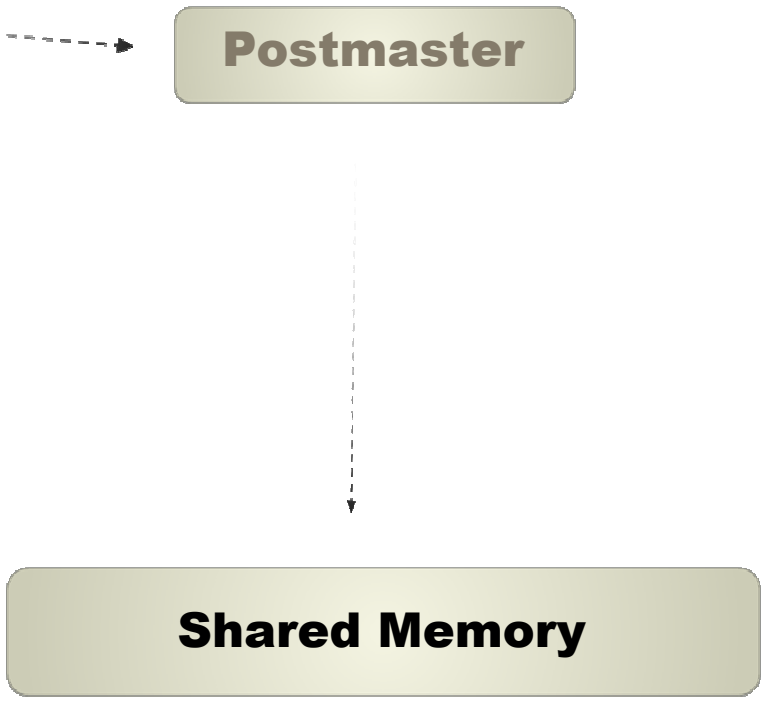
- In this session you will learn:
  - Architectural Summary
  - Process Architecture
  - Connect Request
  - Backend Spawning
  - Respond to Client
  - Full Process Architecture
  - Disk Read Buffering
  - Writing Buffers
  - Shared Buffer & Write-Ahead Log
  - Physical Database Architecture
  - Data File Architecture
  - Page Layout
  - Transaction Logging (WAL) and Archiving

Architectural Summary

- PostgreSQL architecture is process-per-user
  - Each user connection has its own OS process
- Distinct types of processes
  - The postmaster
  - Utility processes
  - User backend processes

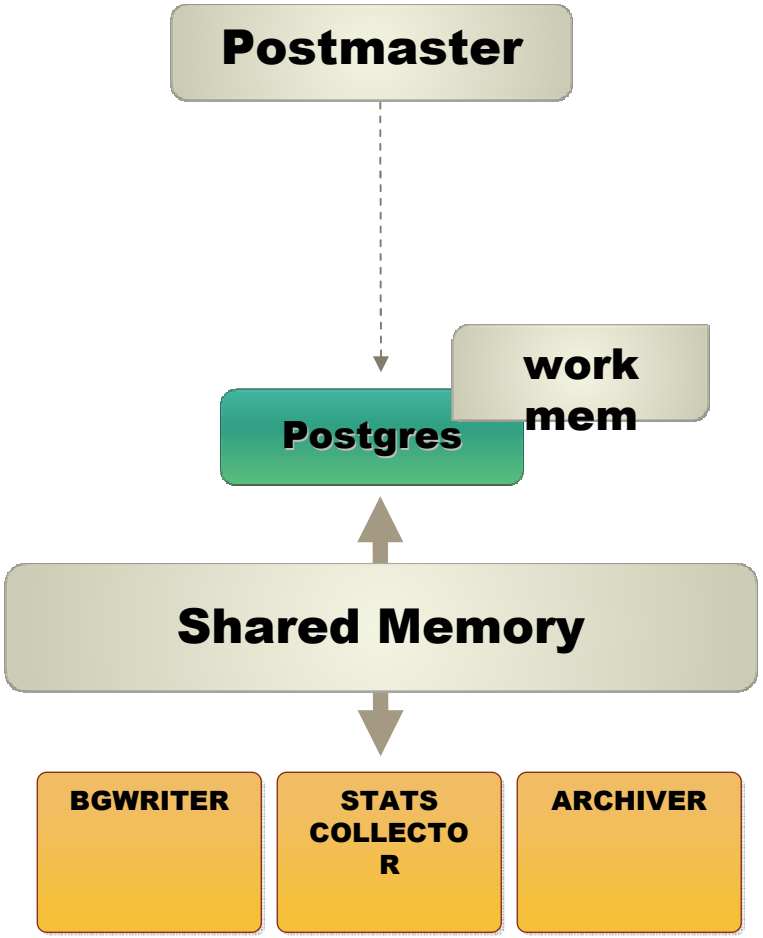
Connect Request

- postmaster
  - listens on 1-and-only-1 well known tcp port address, typically 5432
  - receives client connection request



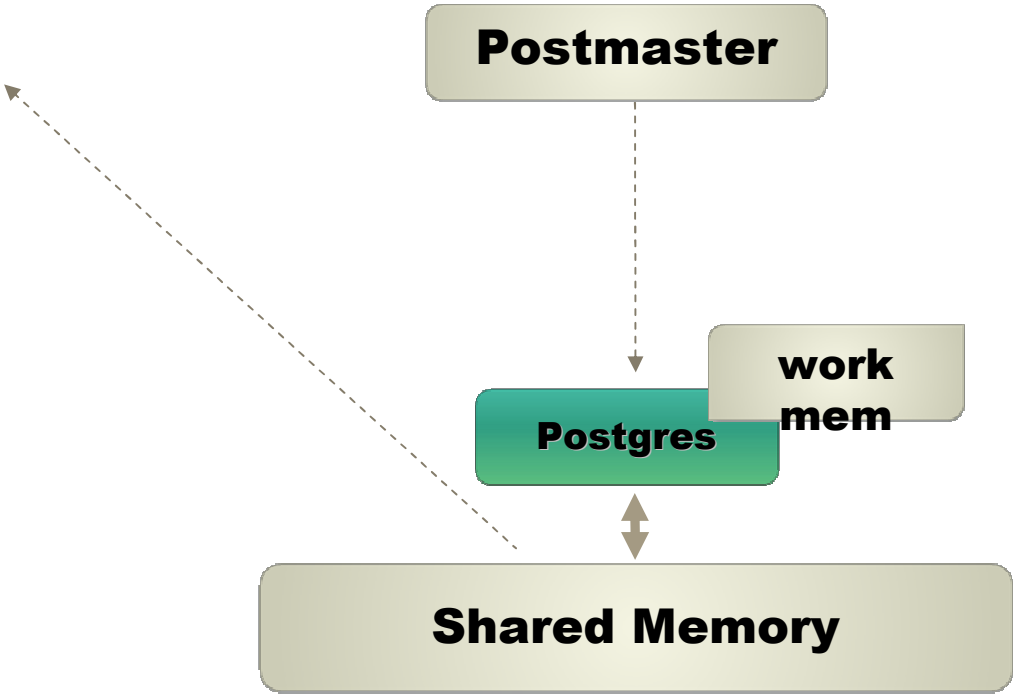
Backend Spawning

- postmaster
  - postgres backend
    - authentication

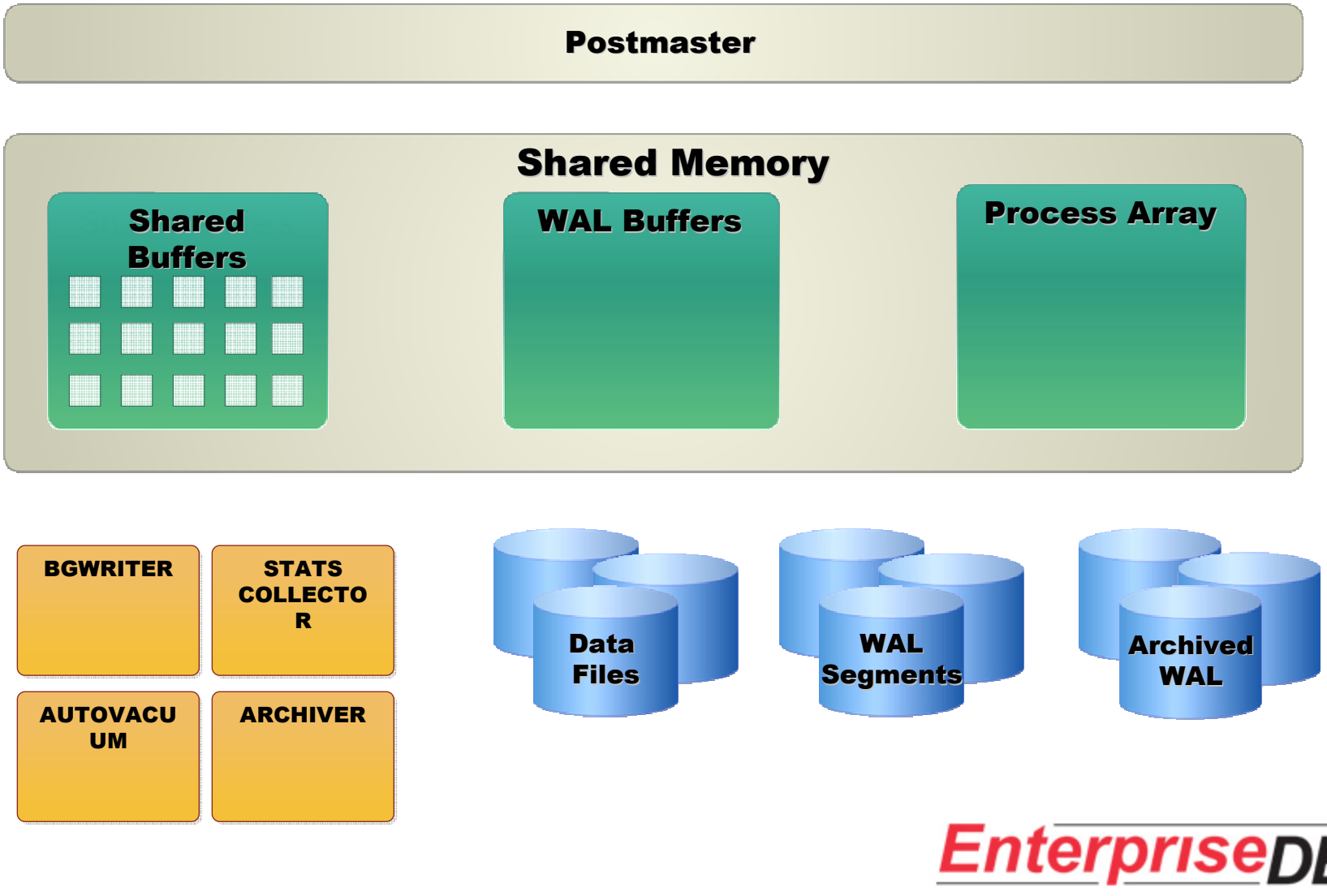


Respond to Client

- postmaster
  - postgres backend
    - callback to client
    - waits for SQL



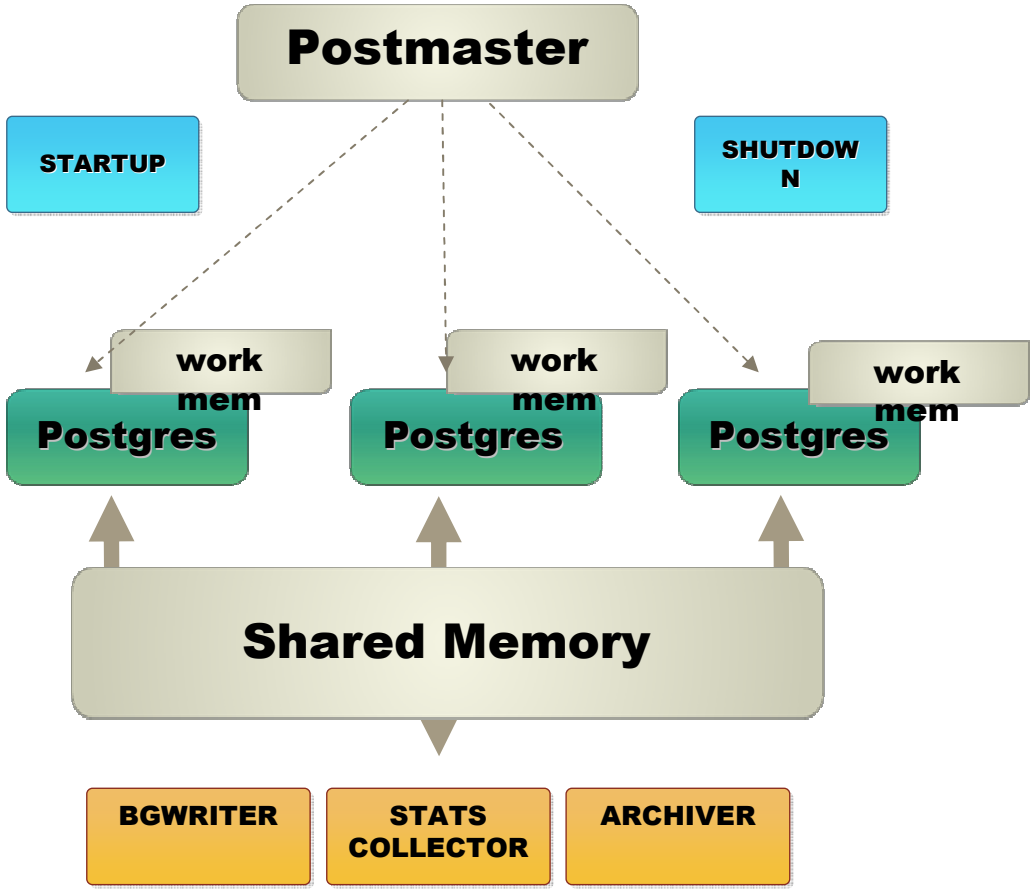
Process Architecture



EnterpriseDB™

Full Process Architecture

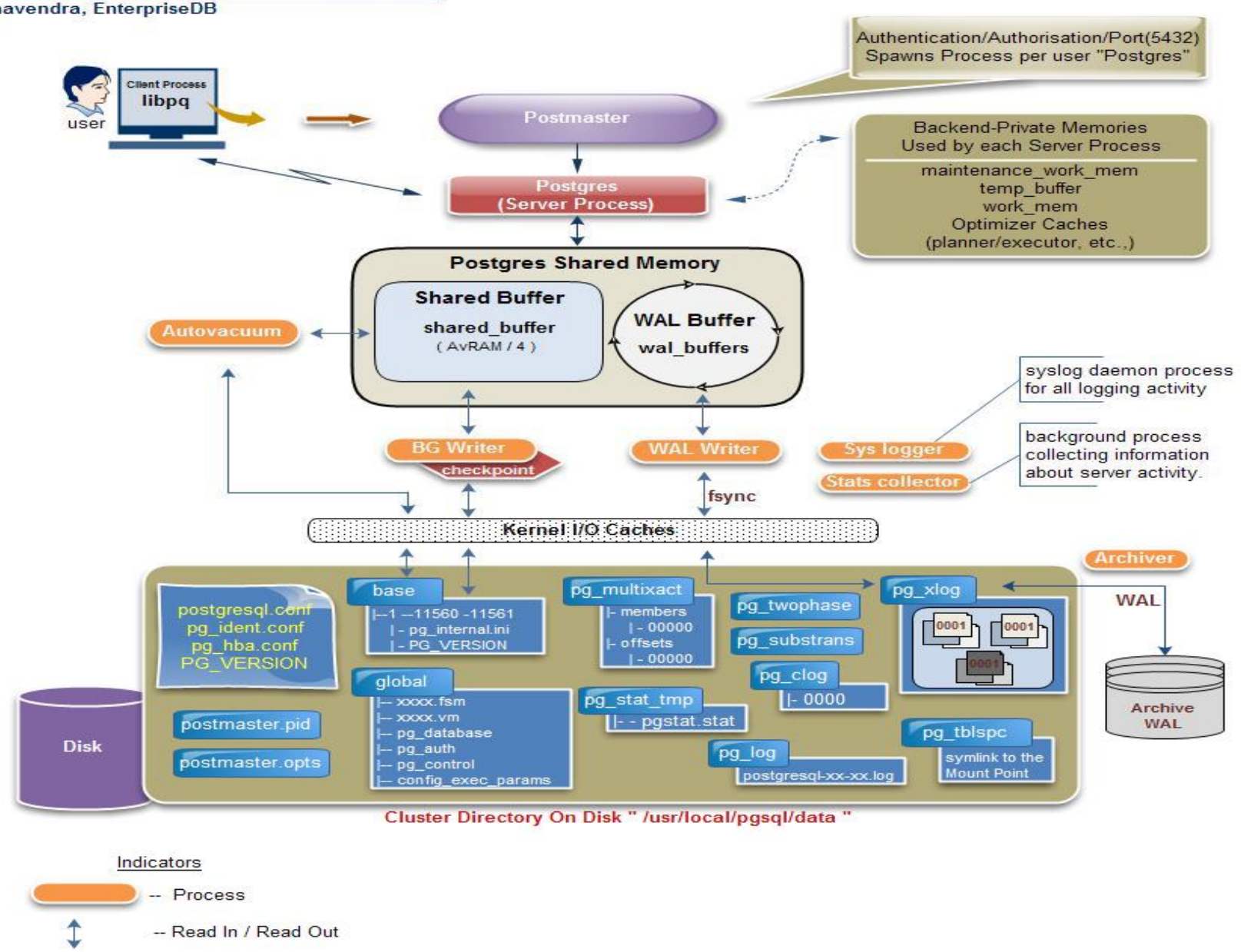
- postmaster
- startup, shutdown
- postgres backends
  - additional connections spawn new backends
- bgwriter
- archiver
- stats collector



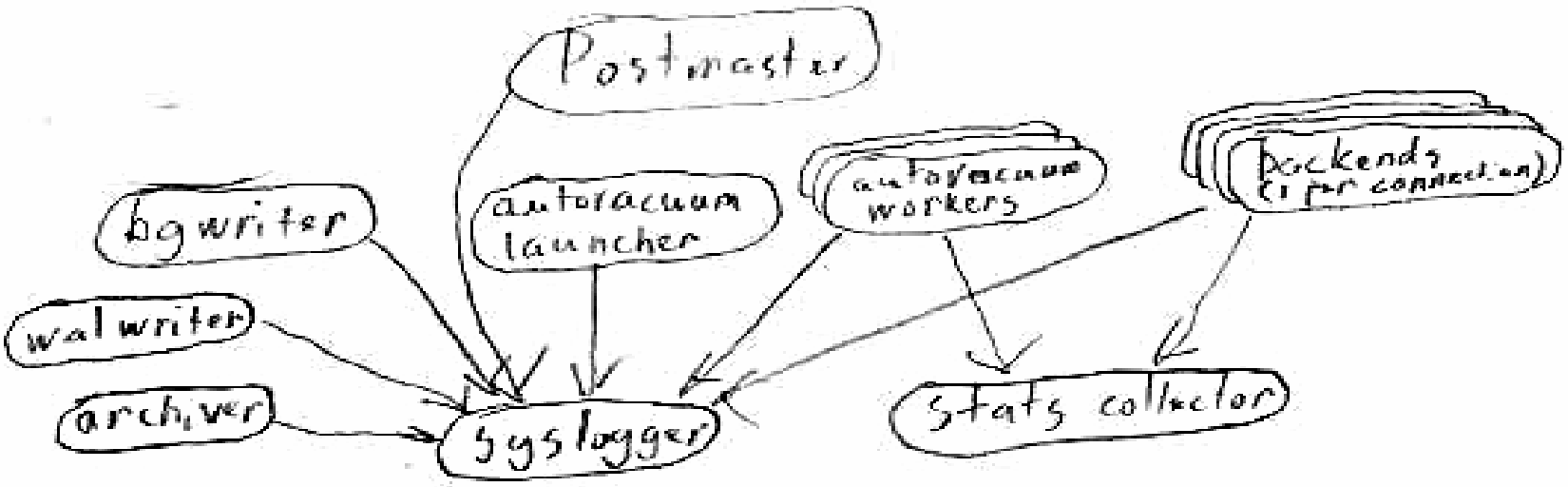
EnterpriseDB™

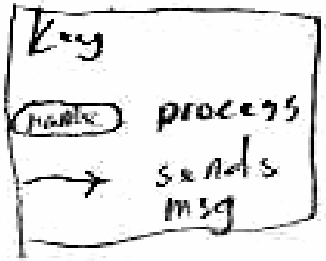
PostgreSQL 8.4 Architecture

Raghavendra, EnterpriseDB



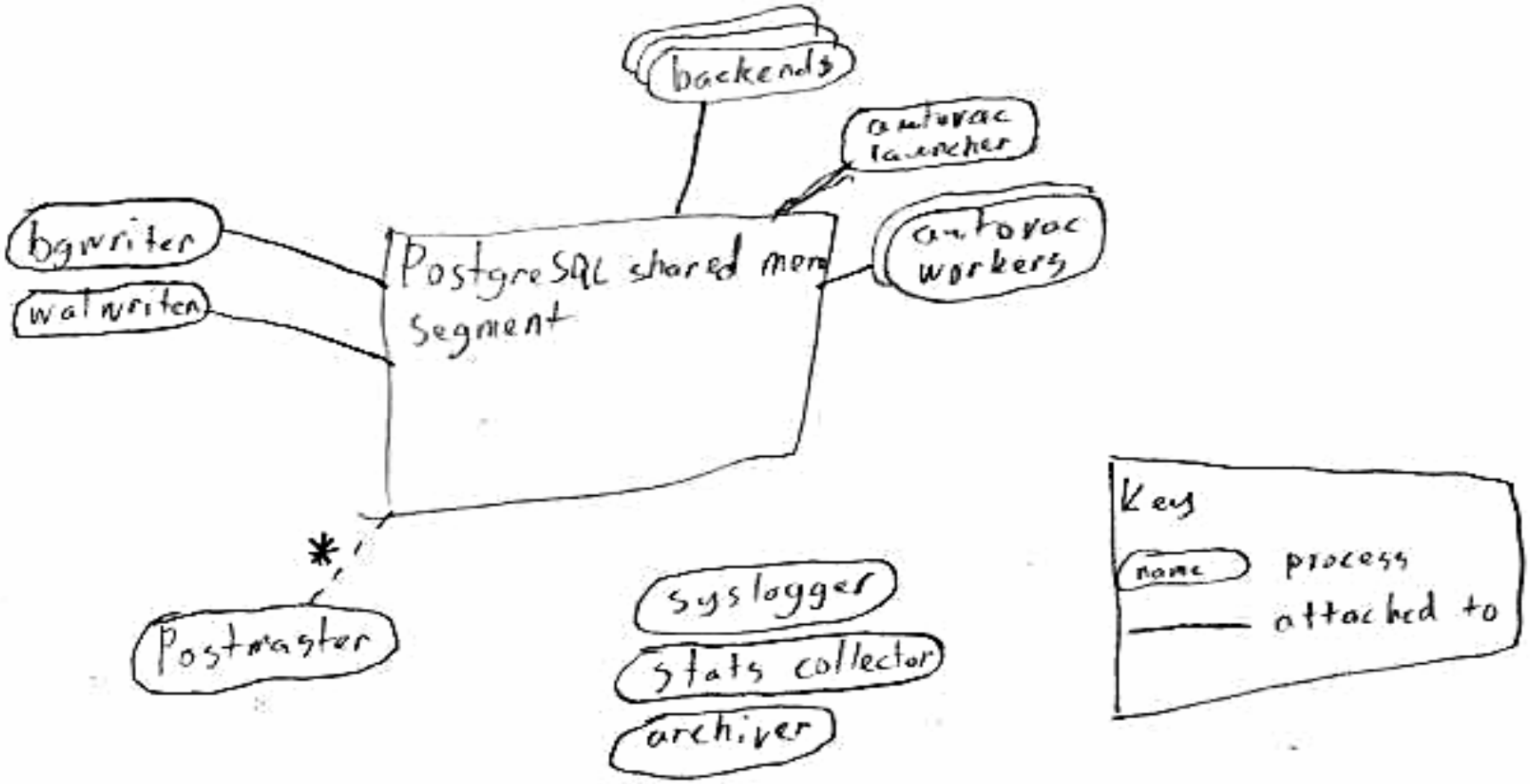
Process view I





EnterpriseDB™

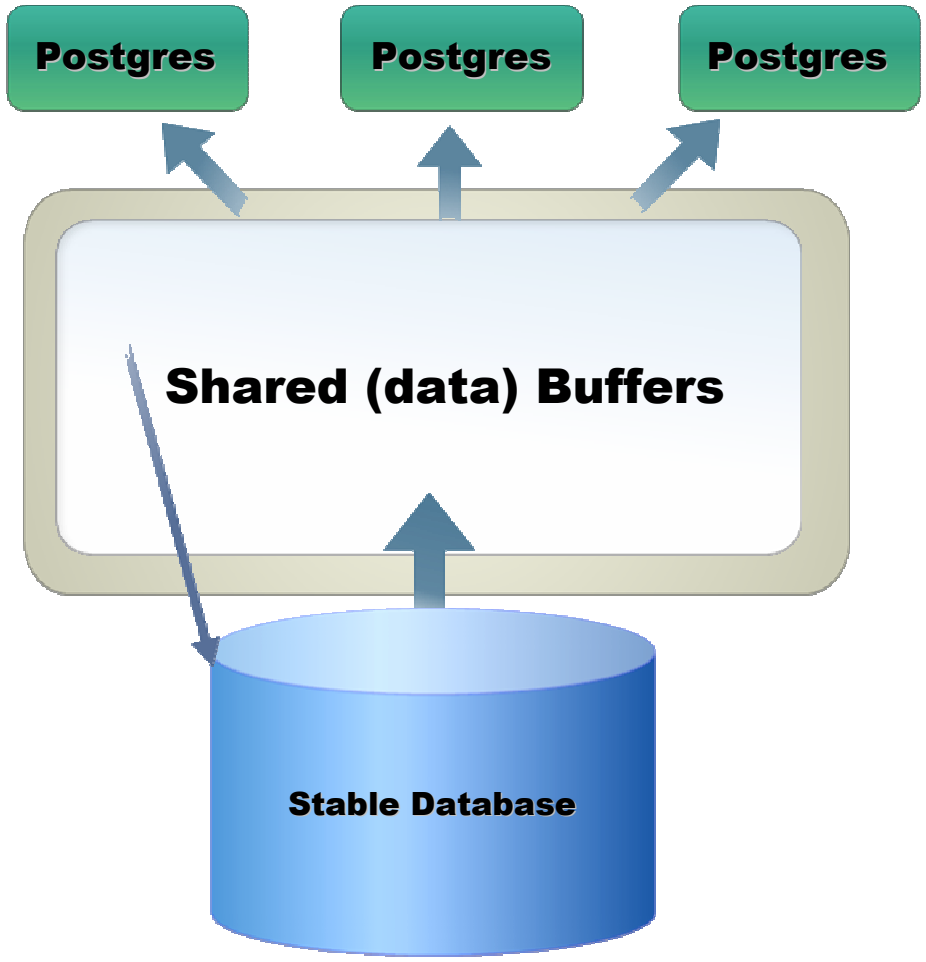
Process view 1+



*Postmaster is attached to shmem segment, but refrains from accessing it

Disk Read Buffering

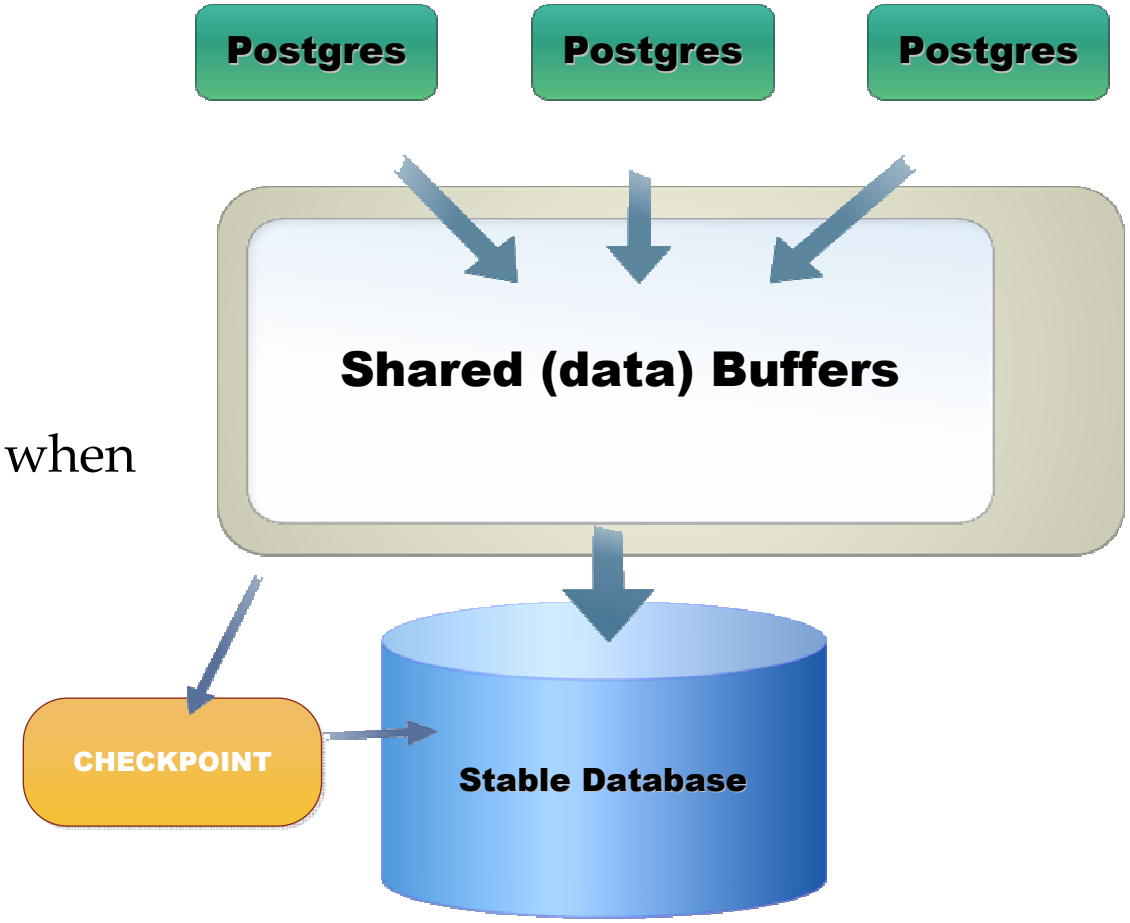
- postmaster
  - postgres backends
    - one read I/O
    - many logical reads from buffer cache
    - cache management improved significantly since 8.0



EnterpriseDB™

Writing Buffers

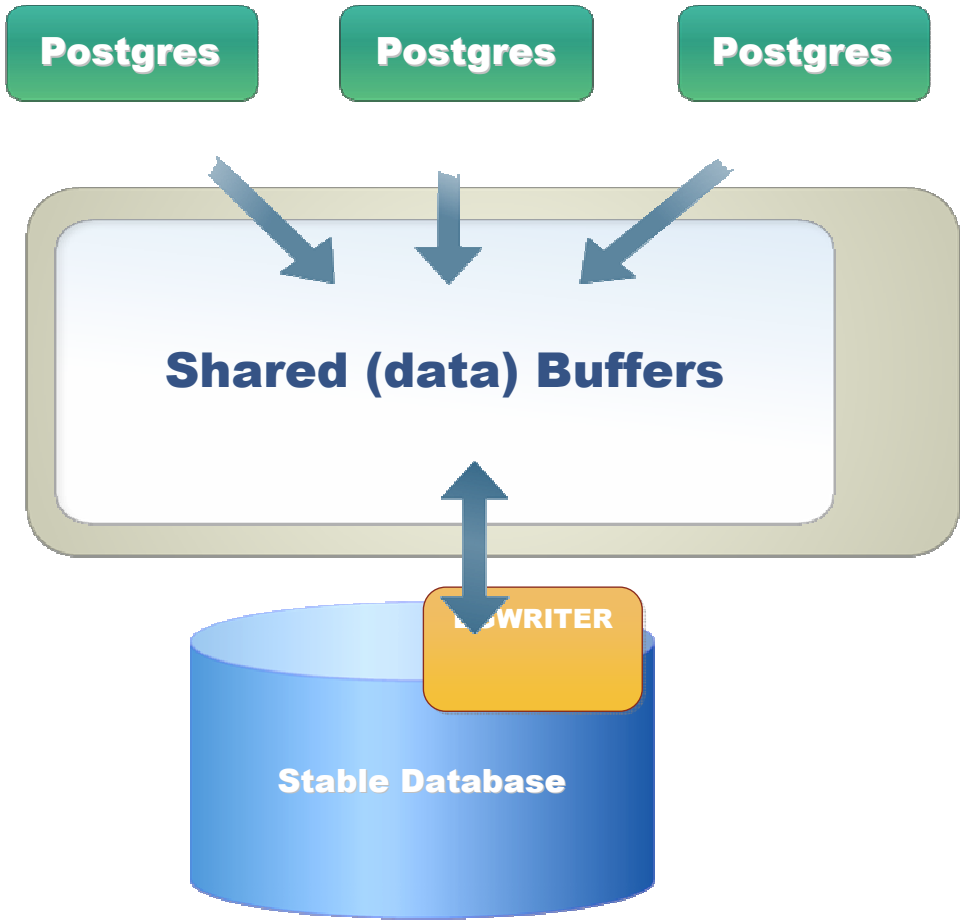
- postmaster
  - checkpoint
  - postgres backends
    - write to disk, when required



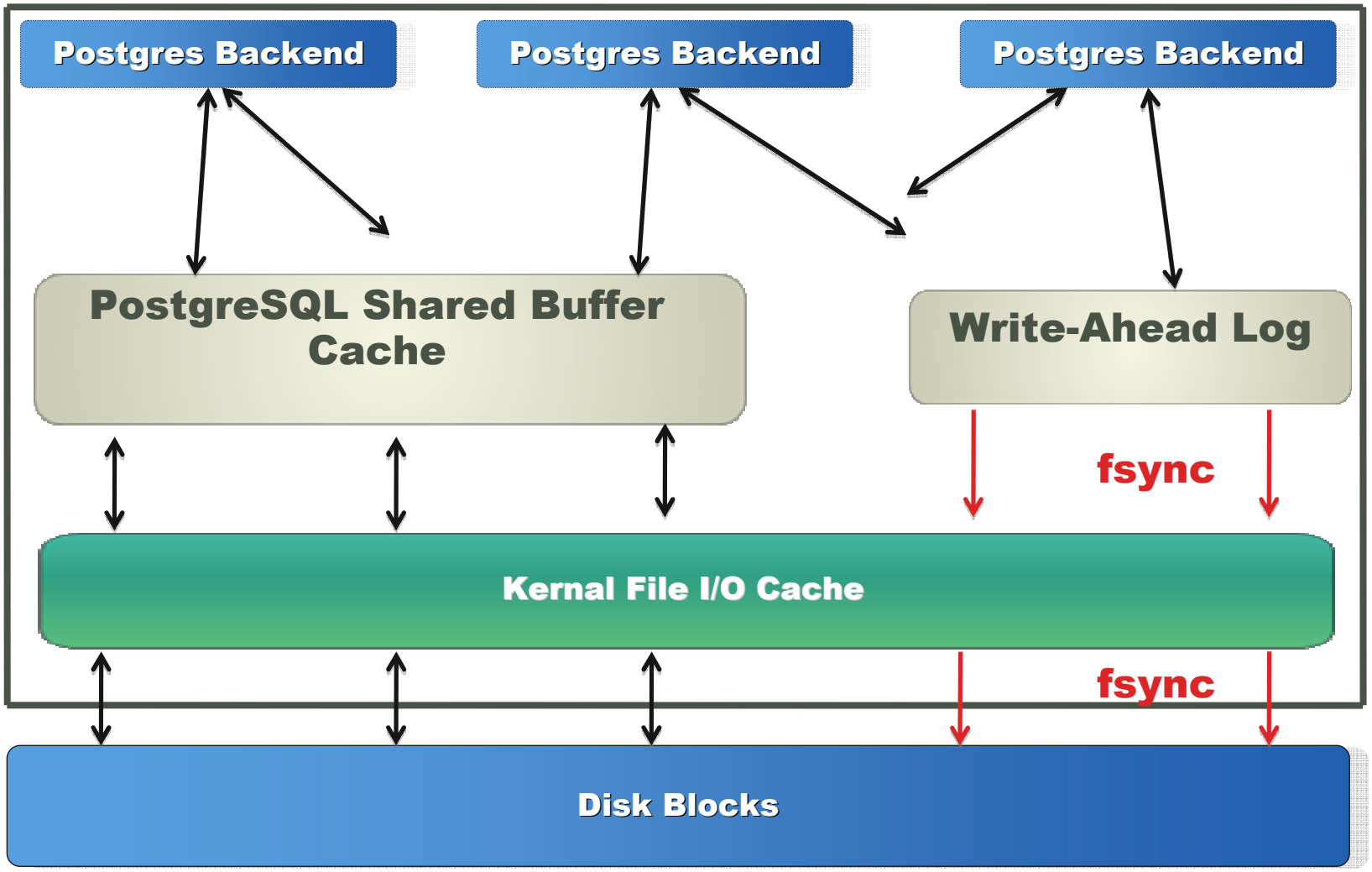
EnterpriseDB™

Writing Buffers

- postmaster
  - bgwriter
    - dirty block cleanout
    - checkpoint



Shared Buffer & Write-Ahead Log



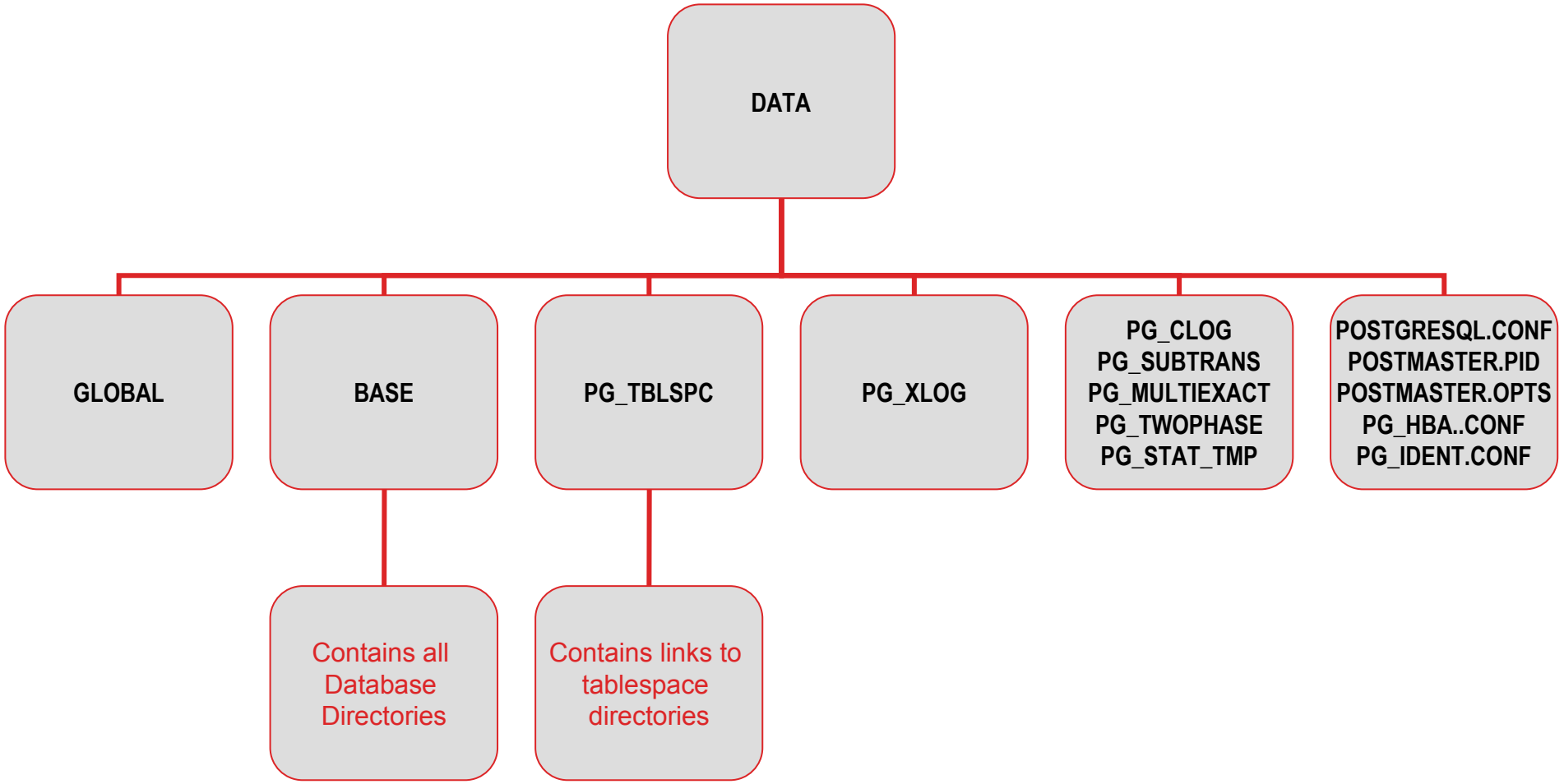
EnterpriseDB™

PostgreSQL Physical Database Architecture

- CLUSTER
  - A database cluster is a collection of databases that are managed by a single server instance
  - Creating a database cluster consists of the following:
    - Creating the directories in which the database will live
    - Generating the shared catalog tables
  - One postmaster and port per cluster
  - Accessed from a single “Data Directory”

PostgreSQL Physical Database Architecture

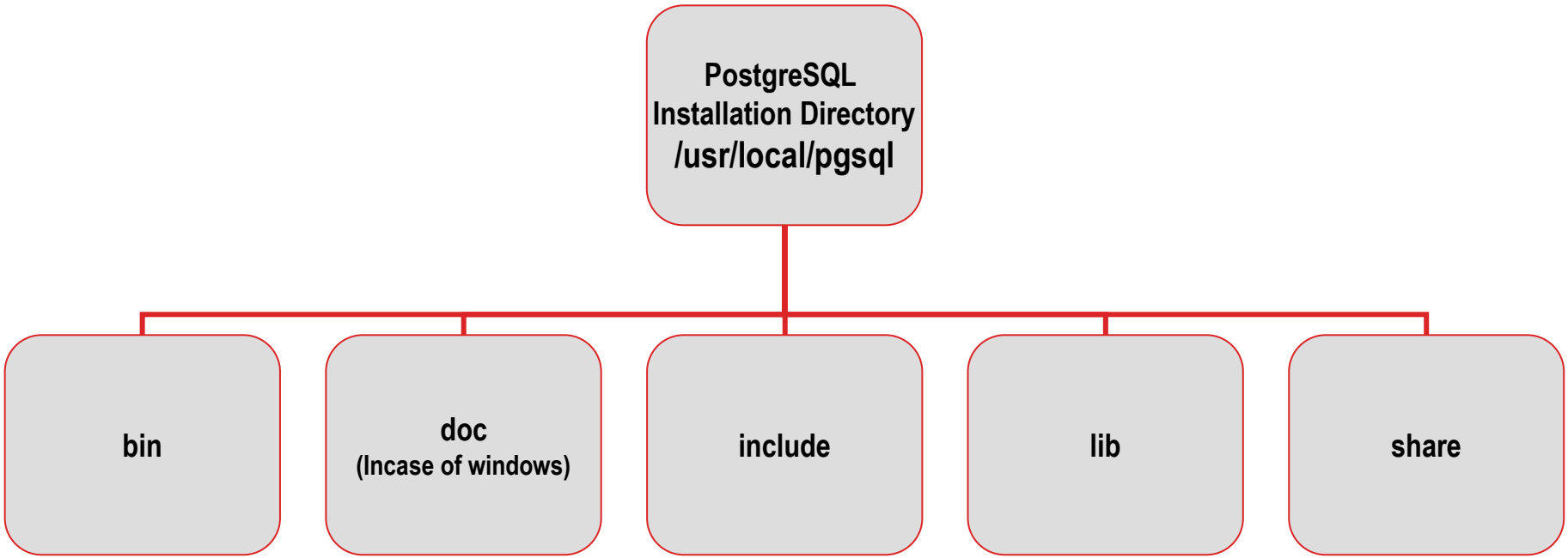
Cluster Data Directory Structure



EnterpriseDB™

PostgreSQL Physical Database Architecture

Installation Directory Structure

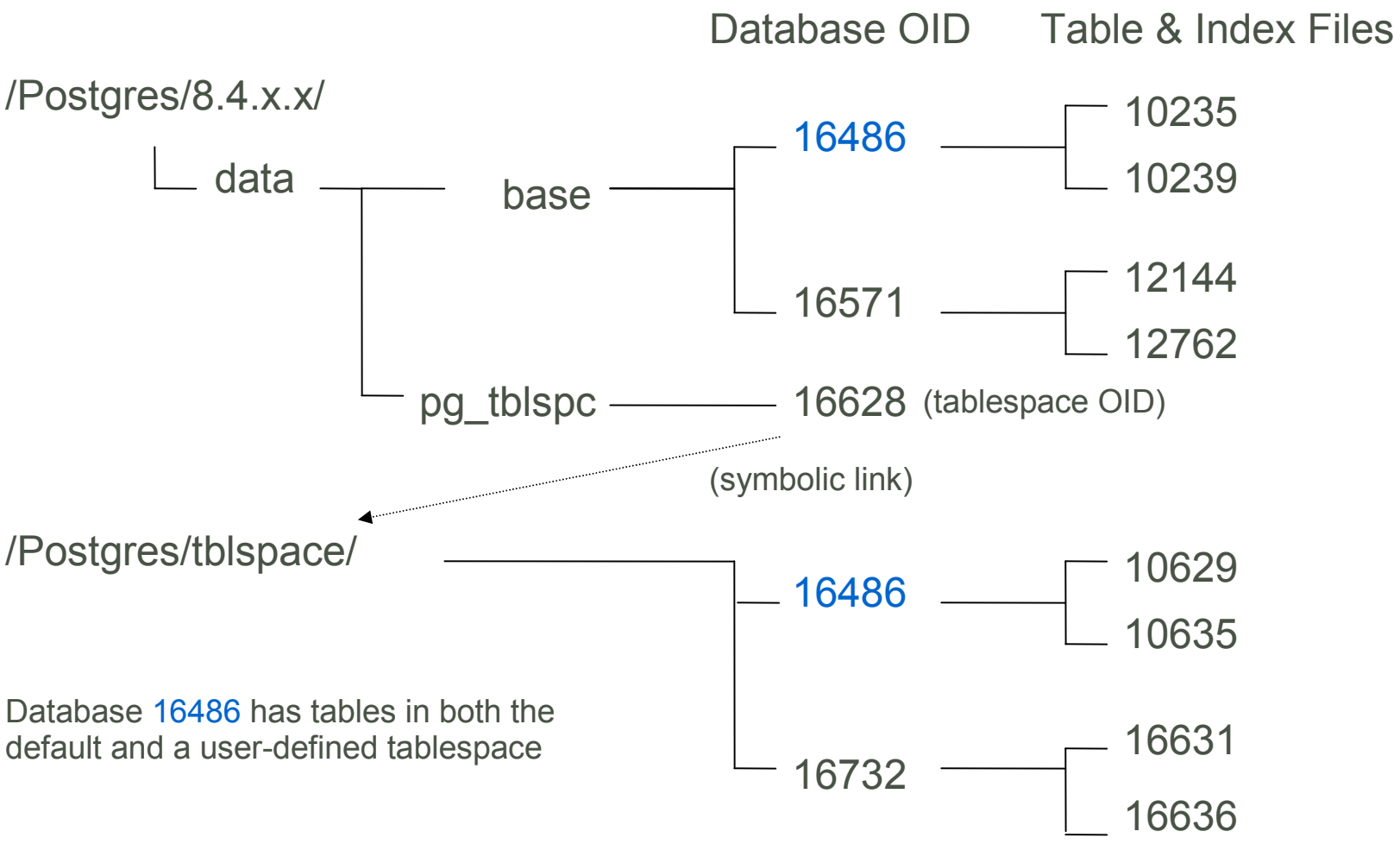


EnterpriseDB™

Physical Database Architecture

- Each Database is one directory
  - with one or more files per Relation
  - ...splits into a new file at max size 1Gb
    - No need to set large file support
- Each Relation is stored in one Tablespace
  - PostgreSQL manages links to other filesystem directories
  - By default, indexes stored in same tablespace

Database Structure



EnterpriseDB™

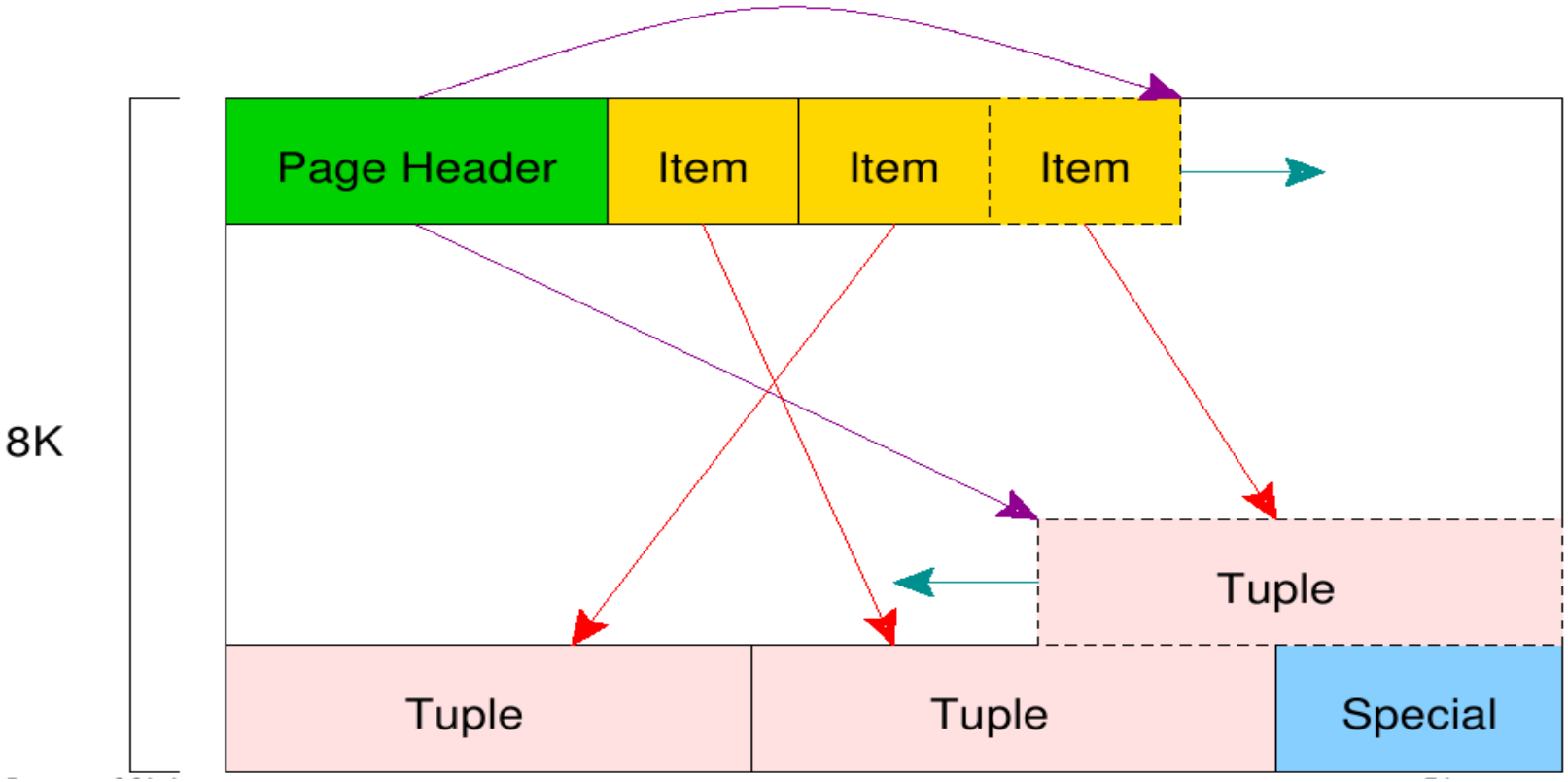
Data Files

- Each table and index is stored in a separate file
  - File name is the table or index's filenode number
  - Filenode number found in pg_class.relfilenode
- Segments
  - Tables or indexes exceeding 1 GB are divided into gigabyte-sized segments
  - First segment is named after the filenode
  - Second and subsequent segments are named filenode.1, filenode.2, etc.

Page Layout

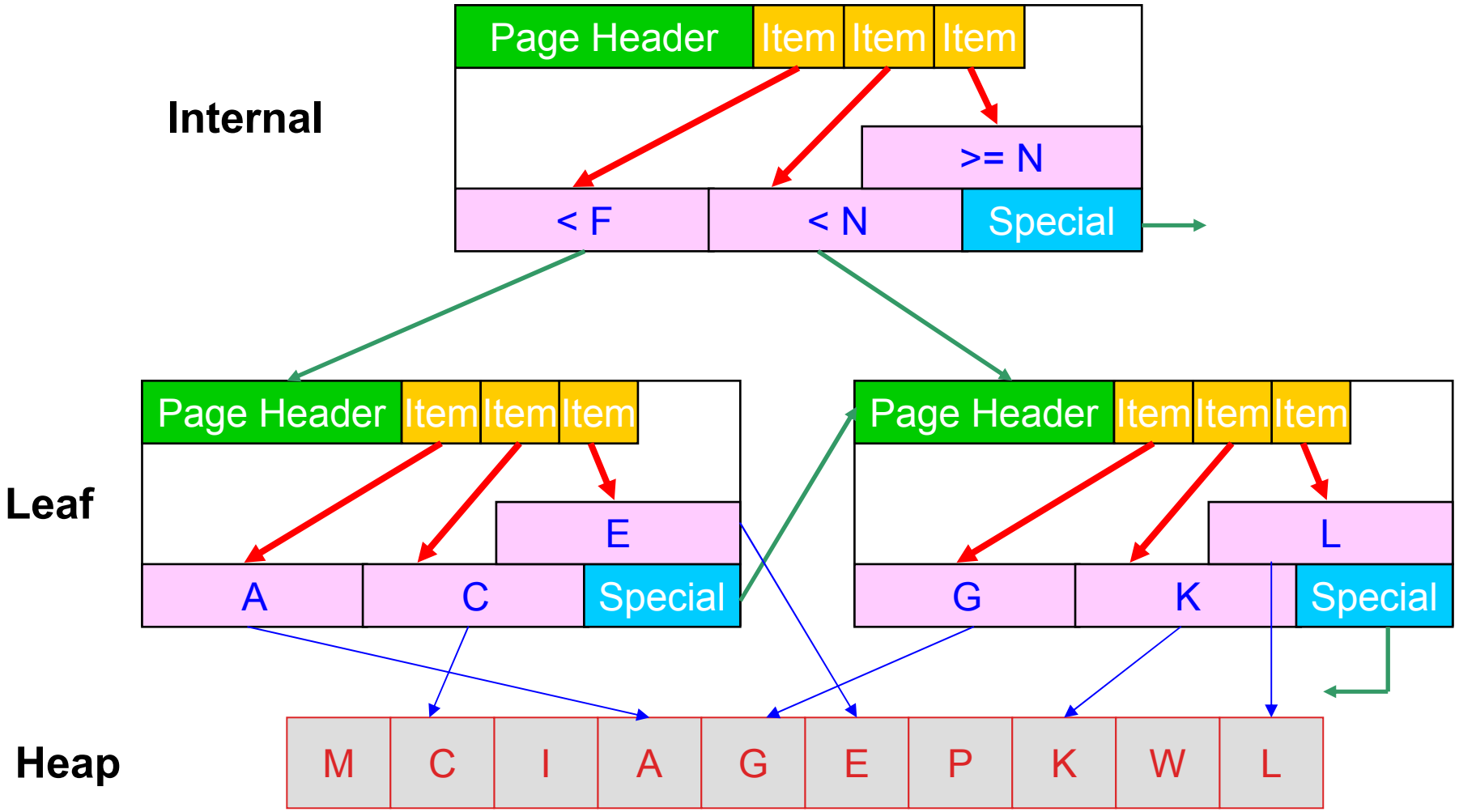
- Page Header
  - General information about the page
  - Pointers to free space
  - 20 bytes long
- Row/Index Pointers
  - Array of offset/length pairs pointing to the actual rows/index entries
  - 4 bytes per item
- Free Space
  - Unallocated space
  - New pointers allocated from the front, new rows/index entries from the rear
- Row/Index Entry
  - The actual row or index entry data
- Special
  - Index access method specific data
  - Empty in ordinary tables

Page Structure



EnterpriseDB™

Index Page Structure



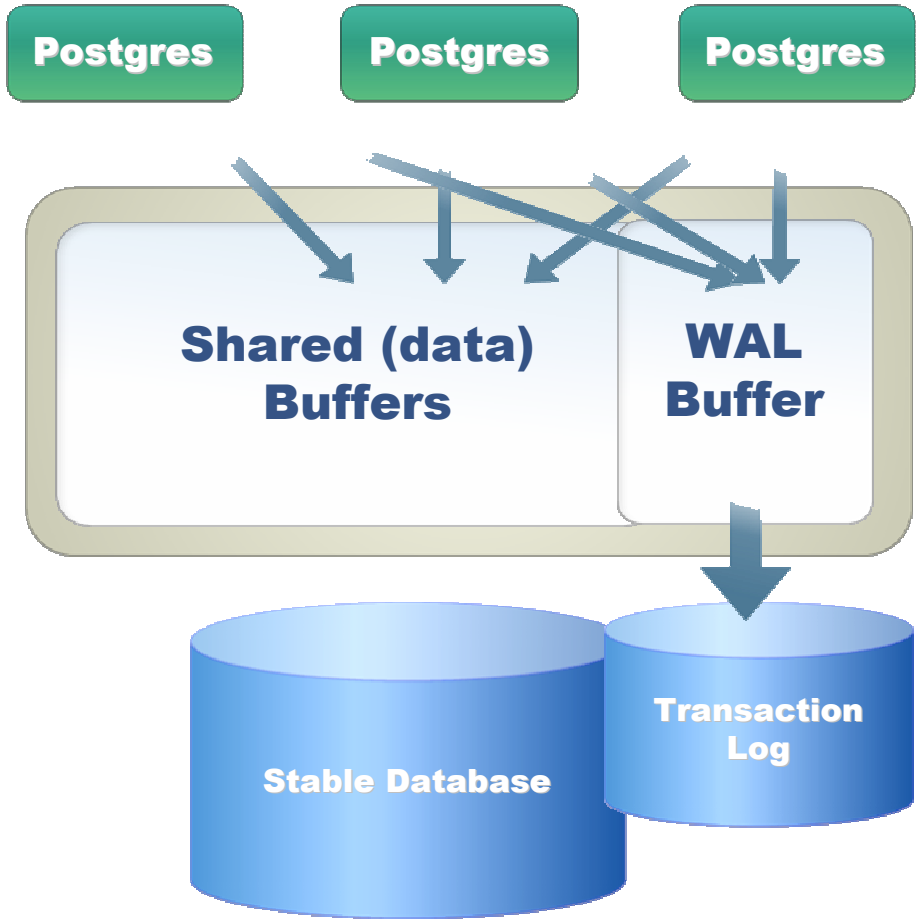
EnterpriseDB™

Commit & Checkpoint

- Before Commit
  - Uncommited updates are in memory
- After Commit
  - Committed updates written from shared memory to disk (write-ahead log file)
- After Checkpoint
  - Modified data pages are written from shared memory to the data files

Write Ahead Logging (WAL)

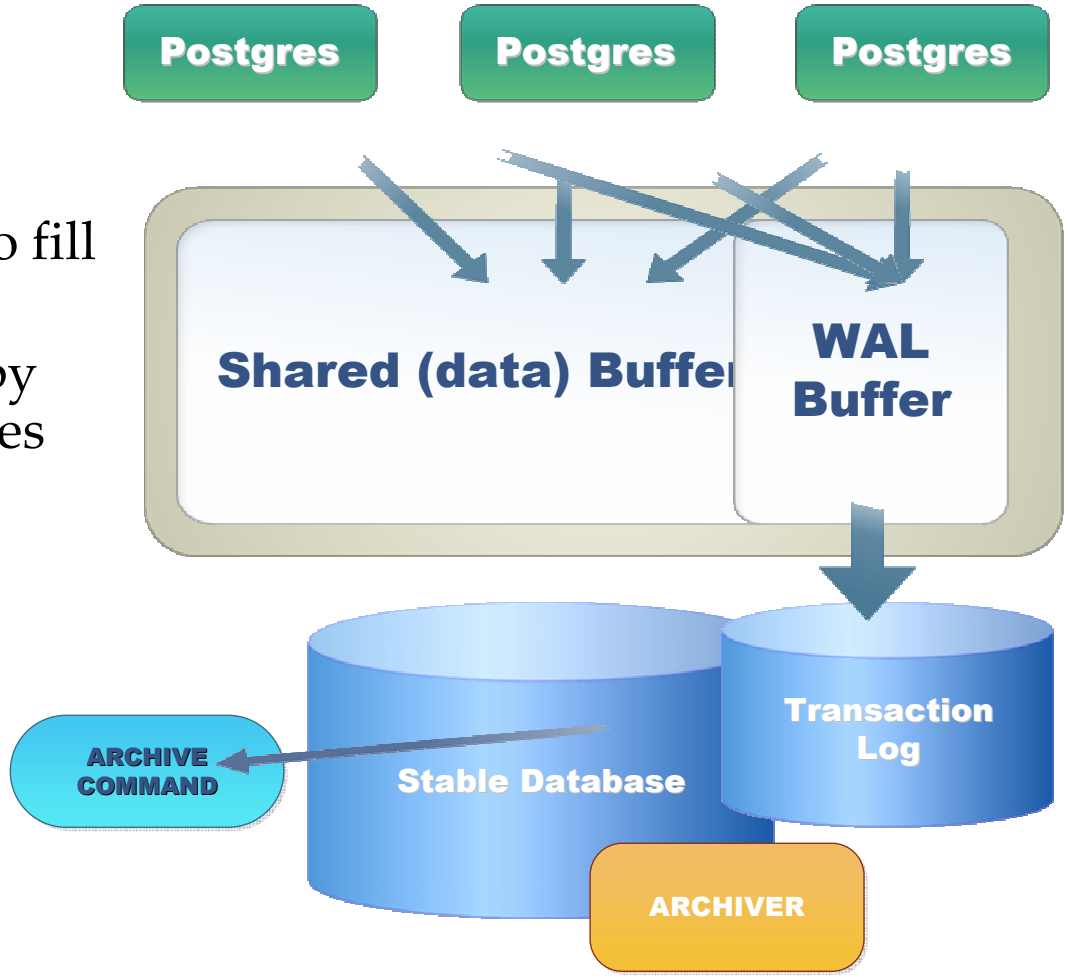
- postmaster
  - postgres backends
    - write data to database buffer
    - write txn log data to wal buffer
    - flush wal buffer on commit (i.e. write to disk in pg_xlog)
      - ...or cache full
      - group commit



EnterpriseDB™

Transaction Log Archiving

- postmaster
  - archiver
    - Waits for xlog files to fill up
    - Spawns a task to copy away pg_xlog log files when full



EnterpriseDB™

Summary

- In this session you learned:
  - Architectural Summary
  - Process Architecture
  - Connect Request
  - Backend Spawning
  - Respond to Client
  - Full Process Architecture
  - Disk Read Buffering
  - Writing Buffers
  - Shared Buffer & Write-Ahead Log
  - Physical Database Architecture
  - Data File Architecture
  - Page Layout
  - Transaction Logging (WAL) and Archiving