EDB Postgres Advanced Server Database Administration Training

EnterpriseDB Korea

2016-02-15

Course Agenda

Day I - Basic

- Introduction
- System Architecture
- Installation
- Lunch
- User Tools for Management
- Creating and Managing Databases
- Configuration
- Security

Day II - Administration

- MVCC, Vacumm & Transaction
- Lunch
- Extension
- Routine Maintenance
- Extension
- Monitoring
- Moving Data
- Scheduler
- Tablespace

Course Agenda

Day III - High Availability

- Backup & Recovery Overview
- Physical Backup & Recovery
- Lunch
- Logical Backup & Recovery
- Streaming Replication
- pgpool-II
- EDB Failover Manager

Day IV - Developer

- Procedural Language
- Extension Development
- Lunch
- Connectors
- JSON & JSONB
- Foreign Data Wrapper

Course Agenda

Day V - Oracle Compatibility

- **Oracle Migration**
- **Lunch**
- **EDB Postgres vs Oracle**
- **9.5 New Feature**
- **Q&A**
- **Wrap-up**
- **Test**

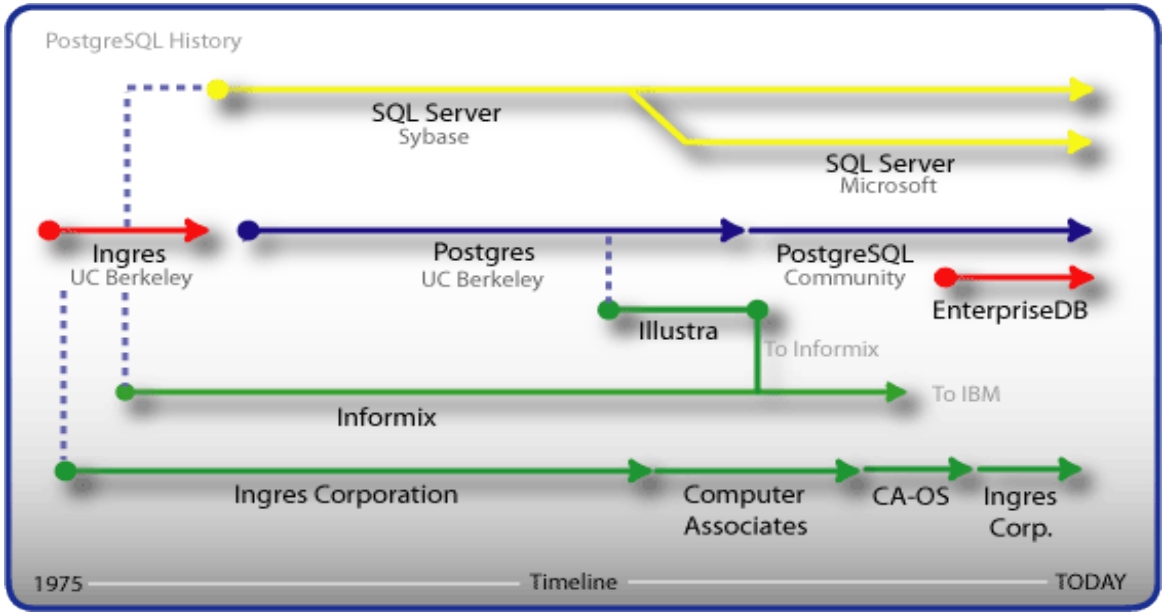
Day I - Basic

- **Introduction**
- **System Architecture**
- **Installation**
- **Lunch**
- **User Tools for Management**
- **Creating and Managing Databases**
- **Configuration**
- **Security**

EDB &
EDB Postgres Introduction

Postgres: A Proven Track Record

- Most mature open source DBMS technology
- Enterprise-class features (built like Oracle, DB2, SQL Server)
- Strong, independent community driving rapid innovation

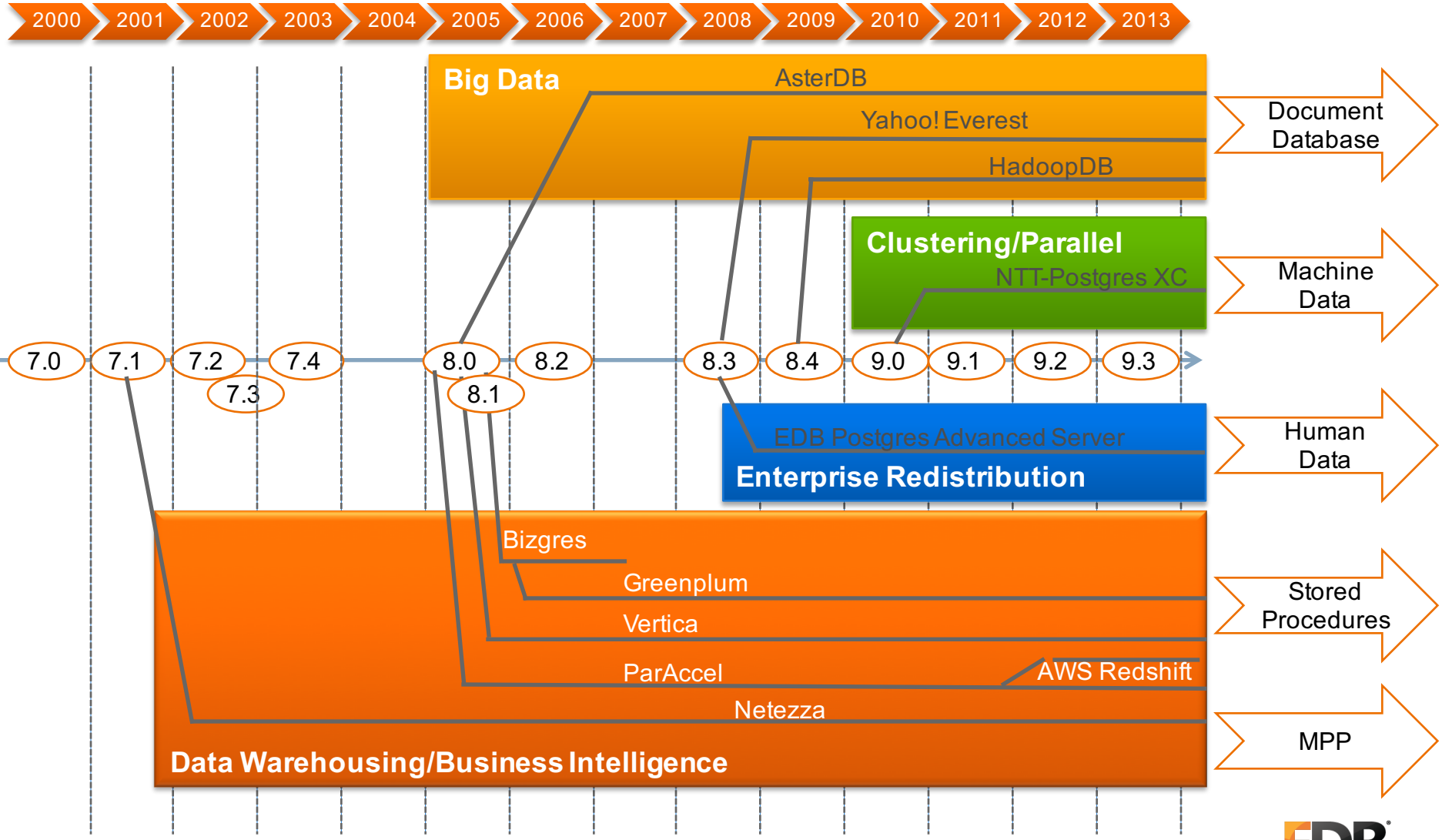


PostgreSQL

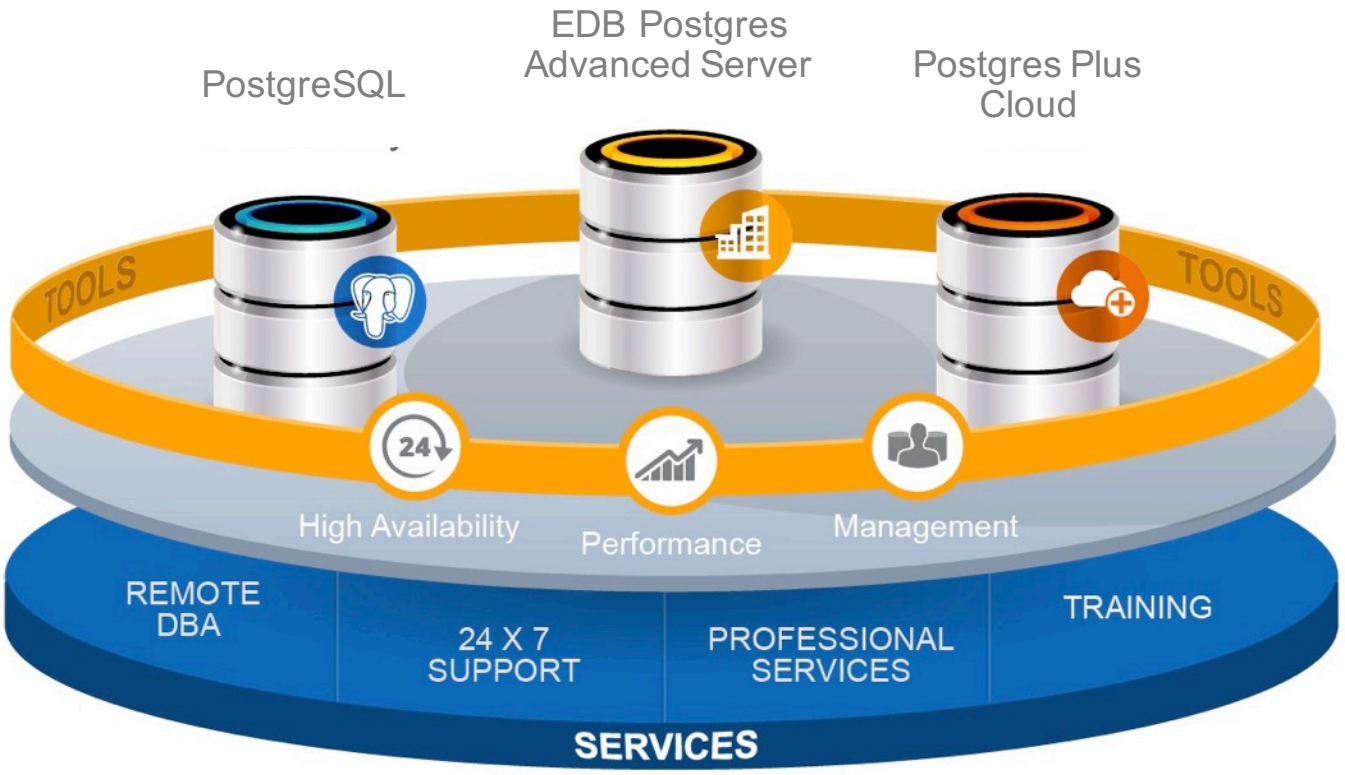
Fully ACID Compliant
MVCC
Point in Time Recovery (PITR)
Data and Index Partitioning
Bitmap Indexes
ANSI Constraints
Triggers & Stored Functions
Views & Data Types
Nested Transactions
Online Backup
Online Reorganization
Foreign Keys
Streaming Replication
Multi-Core Support
JSON Support
HStore

EDB ENTERPRISEDB

PostgreSQL : Foundation Technology



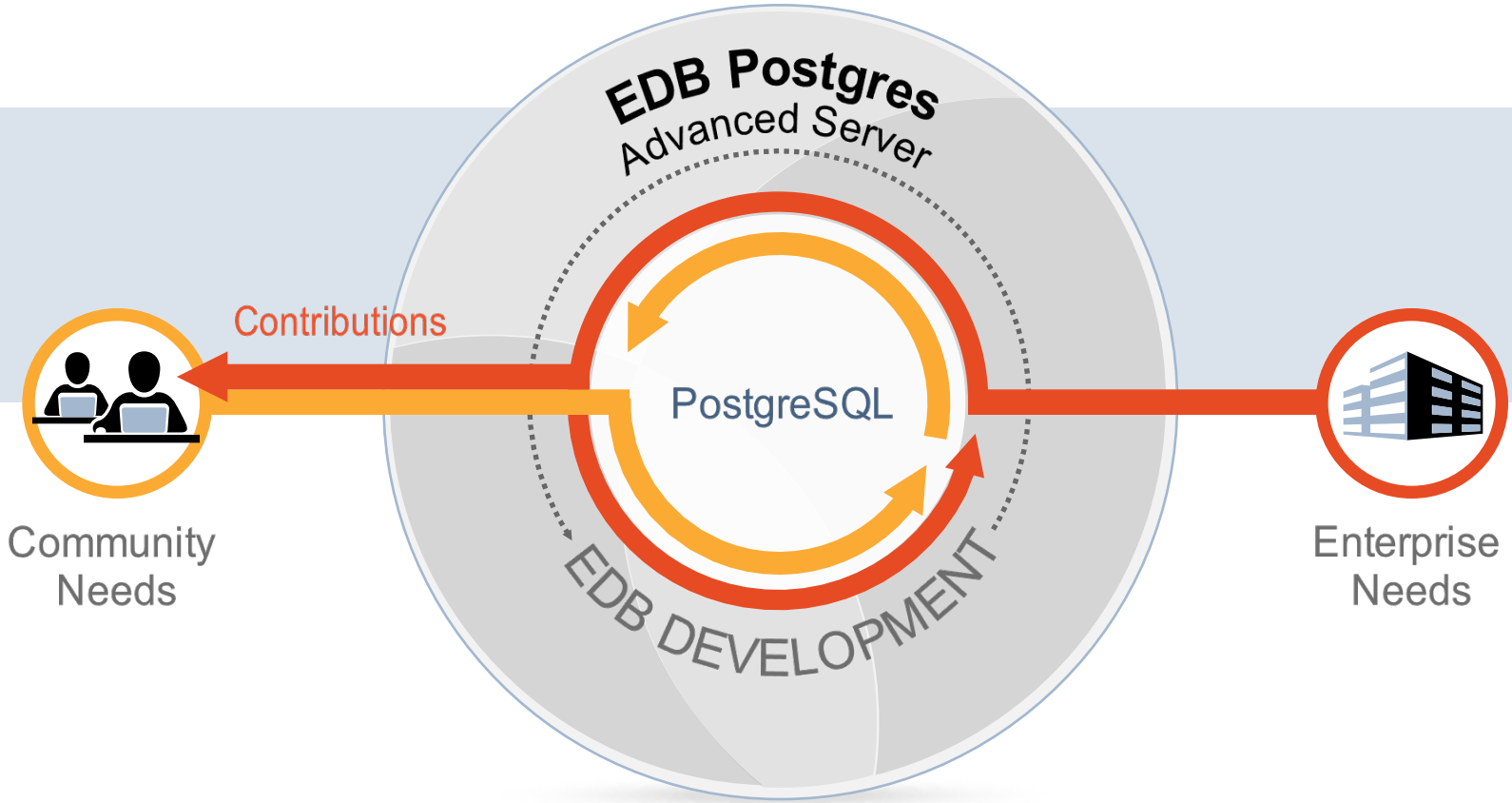
EDB Serves
All Your Postgres Needs



EDB POSTGRES

ADVANCED SERVER

Continuously synchronized with PostgreSQL for a super-set of community PLUS enterprise features



EDB ENTERPRISEDB

EnterpriseDB is a Leader

The Gartner report, *Magic Quadrant for Operational Database Management Systems*, by Donald Feinberg, Merv Adrian, Nick Heudecker, Adam Ronthal, and Terilyn Palanca was published October 12, 2015.

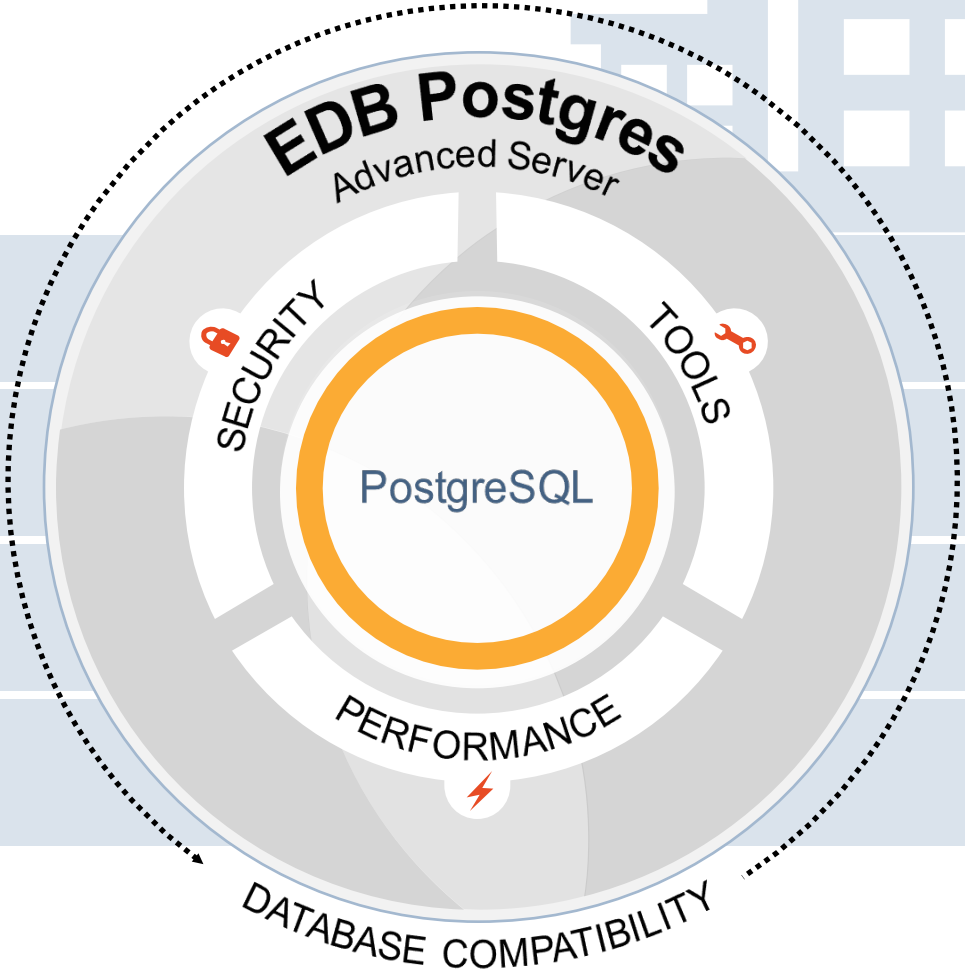
EDB Postgres Advanced Server Key Feature Development

from PostgreSQL core	EDB contributions to PostgreSQL core	*from EDB Development*
• Block Range Indexes (BRIN) • UPSERT • Row Level Security • Schema creation for FDW • Grouping Sets, ROLLUP and CUBE • More JSON functions	**V9.5** **• Performance: sorting, in-memory hash, concurrency locking** **• Parallelism Infrastructure**	• Password profiles: User account and password management • XA two-phase commit protocol across multiple relational databases • Improved compatibility with Oracle 12c • Better user scalability
• Logical Decoding for Scalability • JSONB Data Type • JSONB Indexing • Expanded JSON functions • Delayed Application of Replication • 3x Faster GIN indexes • Support for Linux Huge Pages	**v9.4** **• pg_prewarm** **• ALTER SYSTEM** **• Concurrently updatable Materialized Views** **• Mongo FDW & MySQL FDW**	• CPU & I/O Resource Management • SQL Aggregation with CUBE, ROLLUP and GROUPING SETS • Comprehensive UTL_HTTP Package • Hash Partitioned Tables • Connect_By_Root Operator for hierarchical queries • SQL/Protect Logging to DB Table • EDB*Loader Improved Error handling
• 64 bit LOBs up to 4TB in size • Custom background workers • Writable Foreign Data Wrappers	**v9.3** **• Materialized Views**	• Partition Read Improvements over 75x • Support for 1000s of Partitions • Partition write improvements over 400x
• Cascaded streaming replication • JSON support, Range Types	**v9.2** **• MySQL Foreign Data Wrappers for SQL/MED**	• Table() function support for nested tables • INSERT APPEND hint • EDB Postgres Multi-master Replication • Expanded Object Type support
• Synchronous replication • Serializable Snapshot Isolation • In-memory (unlogged) tables • Writeable Common Table Expressions (WITH)	**v9.1** **• Index-only scans (covering indexes)** **• Linear read scalability to 64 cores**	• Row Level Security • Declarative Partitioning syntax

EDB POSTGRES

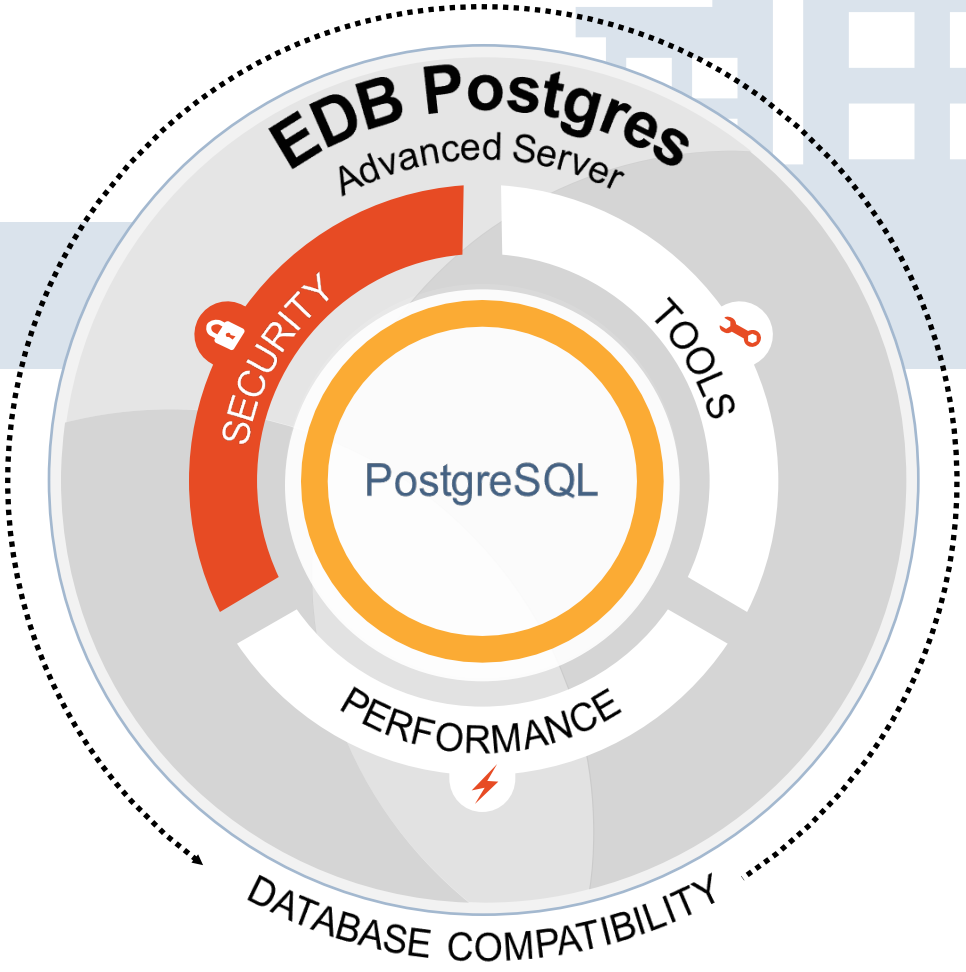
ADVANCED SERVER

- Security
- Tools
- Performance
- Compatibility



EDB POSTGRES

ADVANCED SERVER



Security

- User account / password policy management NEW
- Enhanced Auditing IMPROVED
- Row Level Security (VPD)
- SQL Injection attack guard
- Server-side code protection
- Multiple US Gov't certifications including EAL2

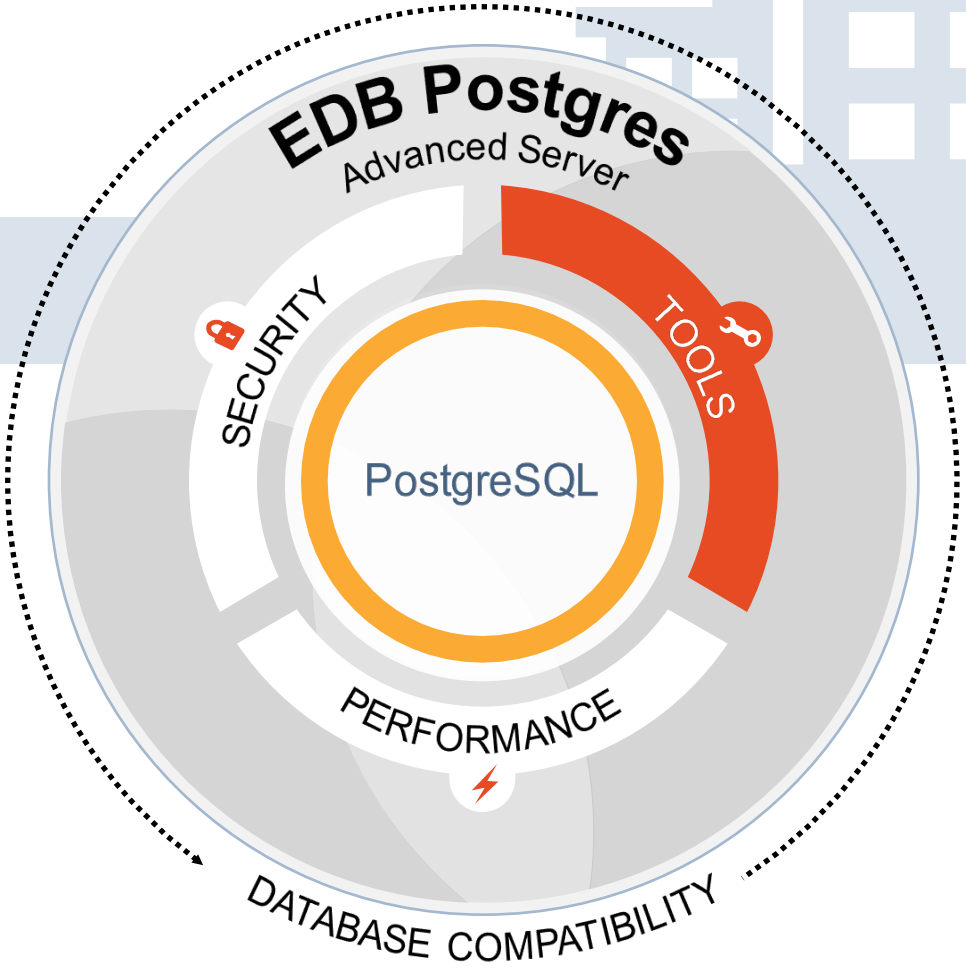
EDB ENTERPRISEDB

EDB POSTGRES

ADVANCED SERVER

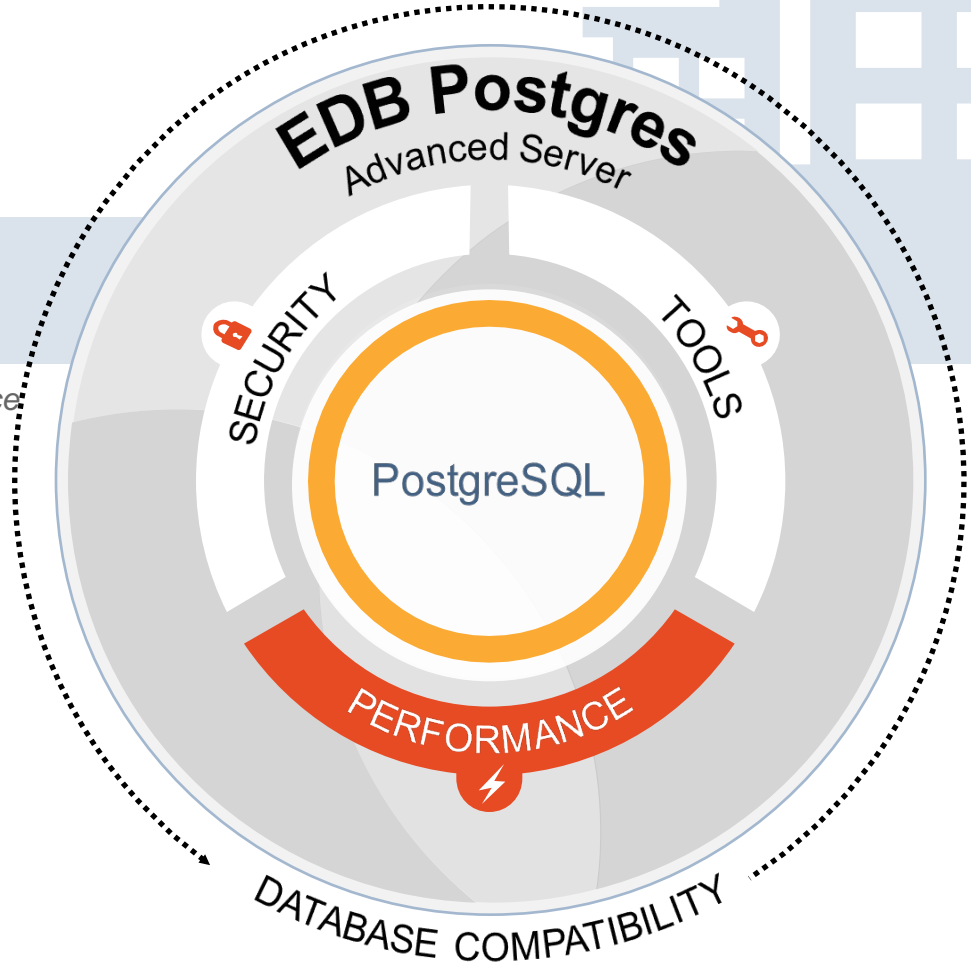
Bundled Tools

- Oracle, SQL Server & PostgreSQL to EDB Postgres replication
- Enterprise management. monitoring, and tuning IMPROVED
- Multi-master replication
- HA failover protection
- Oracle, SQL Server & MySQL to EDB Postgres migration
- Update monitoring



EDB POSTGRES

ADVANCED SERVER



Performance

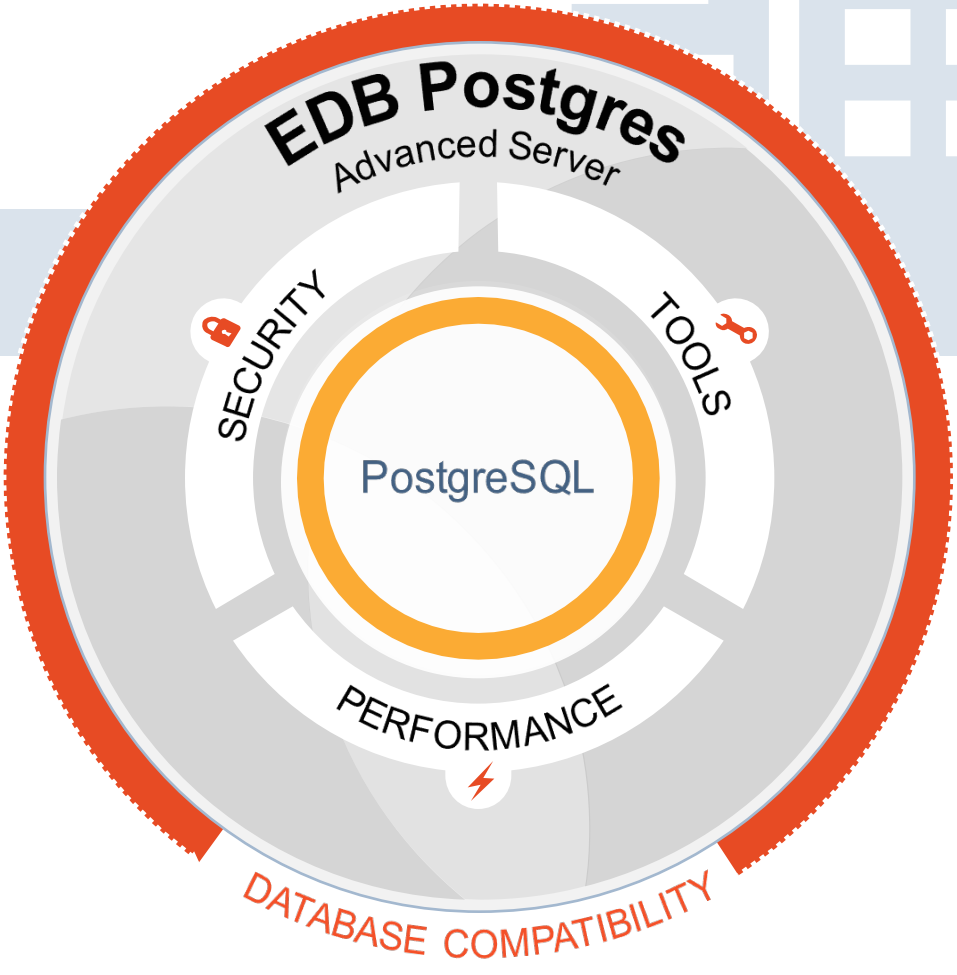
- Improved Scalability – *better performance under high concurrency* IMPROVED
- Resource Manager – *adjust CPU & I/O resources on mixed workloads*
- Faster Partitioning - *400x faster writes & 76x faster selects*
- SQL Profiler – *fix slow workloads*
- Bulk Data Loader - *2x faster* IMPROVED
- Index Advisor - *speeds up inquiries*
- DynaTune - *memory upgrades*
- Dynamic runtime statistics - *reveals SQL wait bottlenecks*

EDB POSTGRES

ADVANCED SERVER

Database Compatibility for Oracle®

- Faster, easier migrations
- PL/SQL, OCI support
- Oracle SQL extensions IMPROVED
- User defined objects
- Function packages NEW
- Database links
- Oracle-like tools:
  *EDB*Loader, EDB*Plus, EDB*Wrap*



EDB Postgres Advanced Server System Architecture

PostgreSQL Terminology

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

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

PostgreSQL Limits

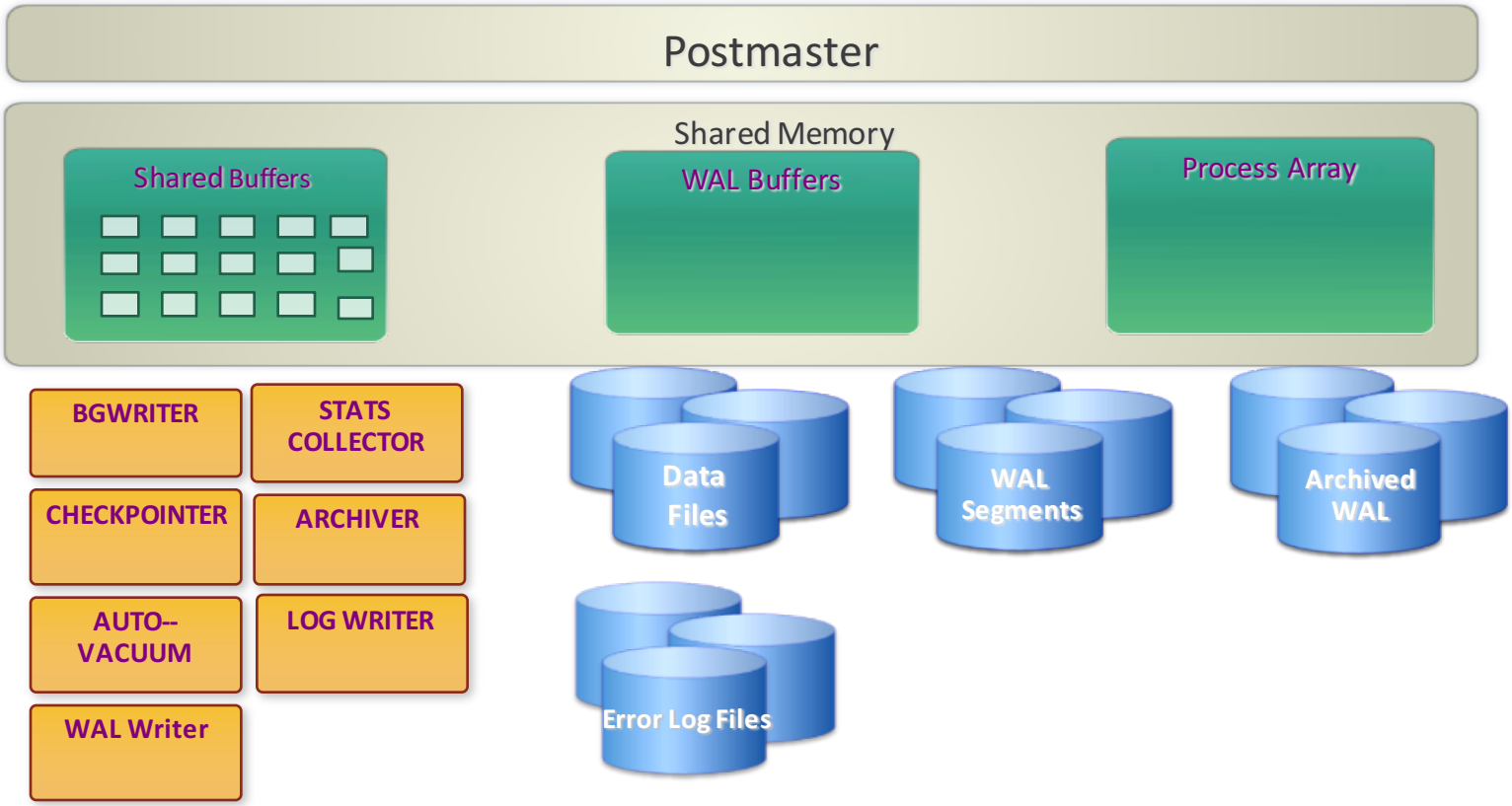
- Limitations are generally defined by:
  - Operating System Limits
  - Compile-Time Parameters
  - Data Type Usage
- General database limitations:

Limit	Value
Maximum Database Size	Unlimited
Maximum Table Size	32 TB
Maximum Row Size	1.6 TB
Maximum Field Size	1 GB
Maximum Rows per Table	Unlimited
Maximum Columns per Table	250-1600 (Depending on Column types)
Maximum Indexes per Table	Unlimited

Architecture

- EDB Postgres Advanced Server Architecture is based on PostgreSQL architecture
- Lets discuss following three parts of the architecture in detail:
  1. Process
  2. Memory
  3. Storage

Process & Memory Architecture



Utility Processes

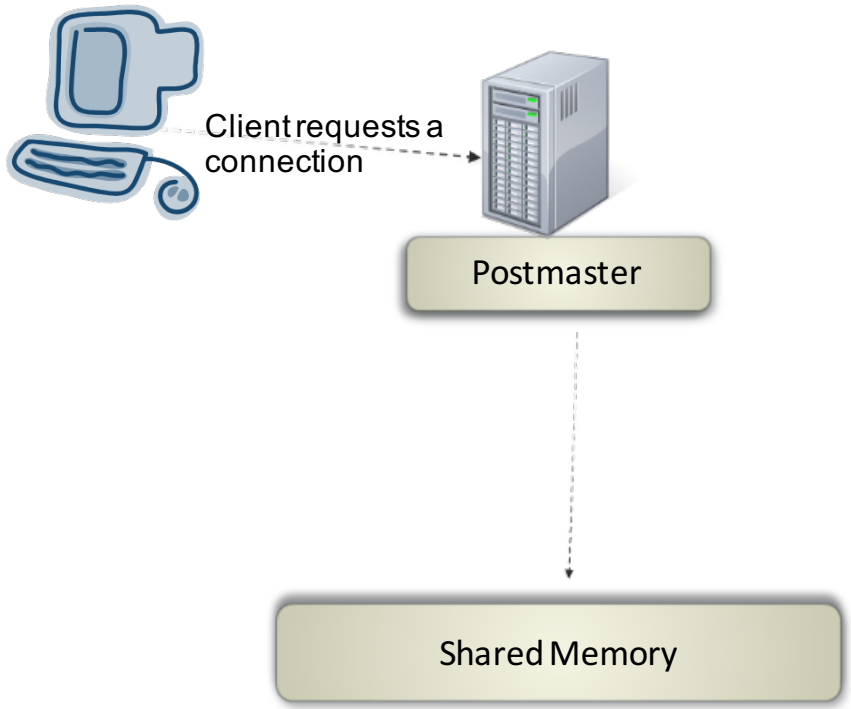
- Background writer
  - Writes dirty data blocks to disk
- WAL writer
  - Flushes write-ahead log to disk
- Checkpointer process
  - Automatically performs a checkpoint based on config parameters
- Autovacuum launcher
  - Starts Autovacuum workers as needed
- Autovacuum workers
  - Recover free space for reuse

More Utility Process

- Logging collector
  - Routes log messages to syslog, eventlog, or log files
- Stats collector
  - Collects usage statistics by relation and block
- Archiver
  - Archives write-ahead log files

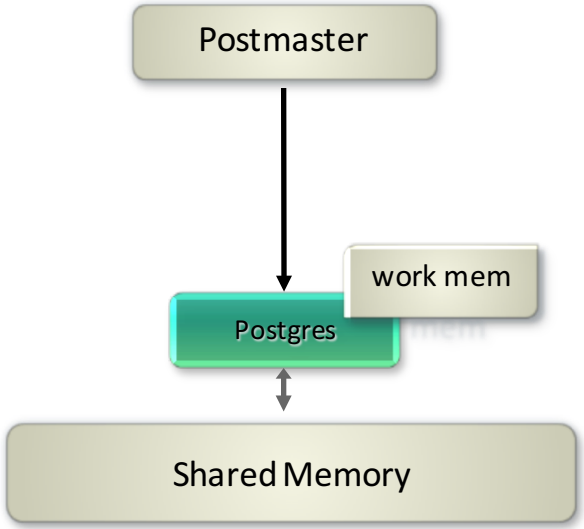
Postmaster as Listener

- Postmaster is master process called postgres
- Listens on 1, and only 1, tcp port
- Receives client connection requests



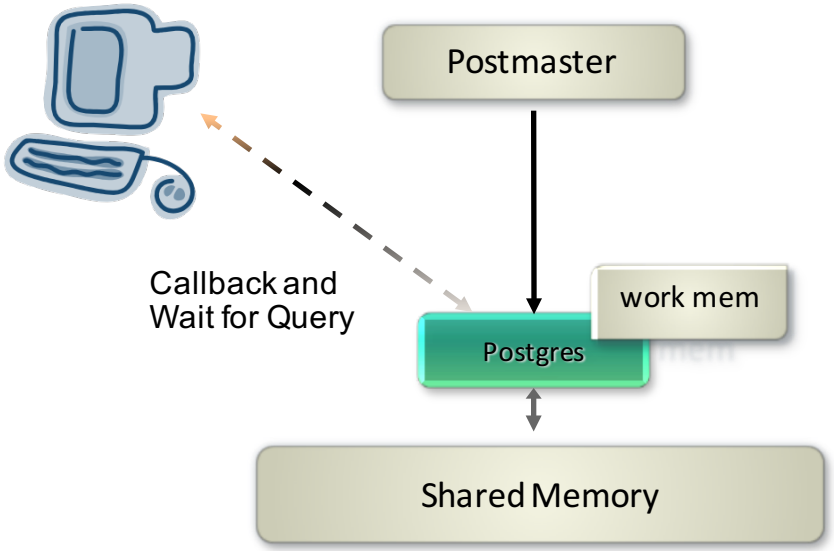
Backend Spawning

- Master process postgres spawns a new server process for each connection request detected
- Communication is done using semaphores and shared memory
- Authentication: IP, user and password
- Authorization: Verify Permissions



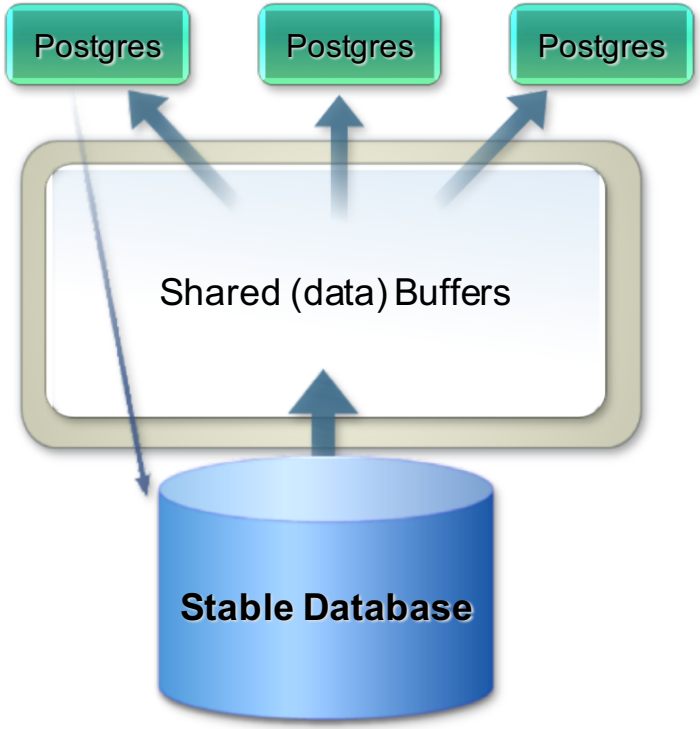
Respond to Client

- User backend process called postgres
- Callback to client
- Waits for SQL
- Query is transmitted using plain text



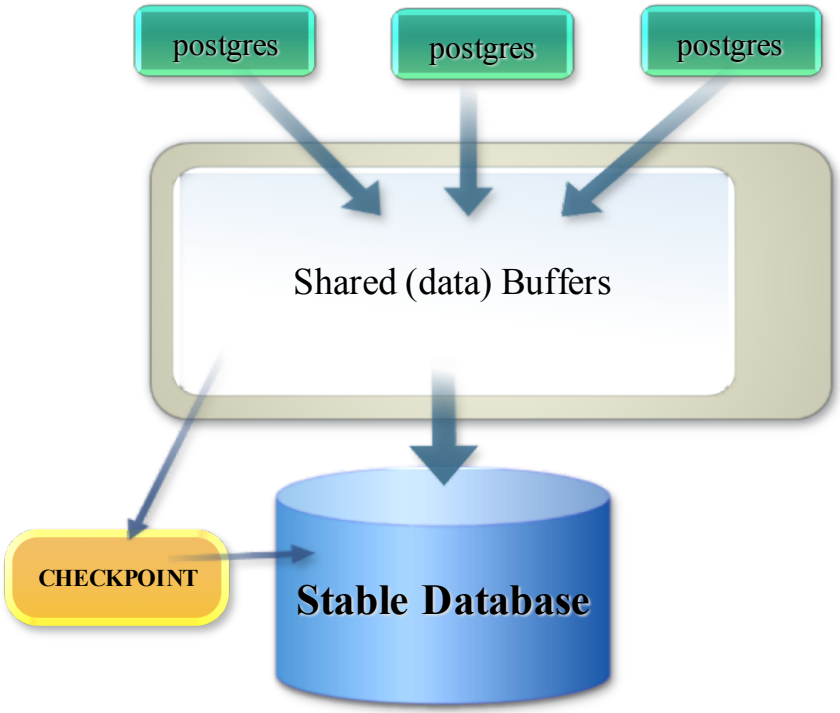
Disk Read Buffering

- PostgreSQL buffer cache (shared_buffers) reduces OS reads.
- Read the block once, then examine it many times in cache.



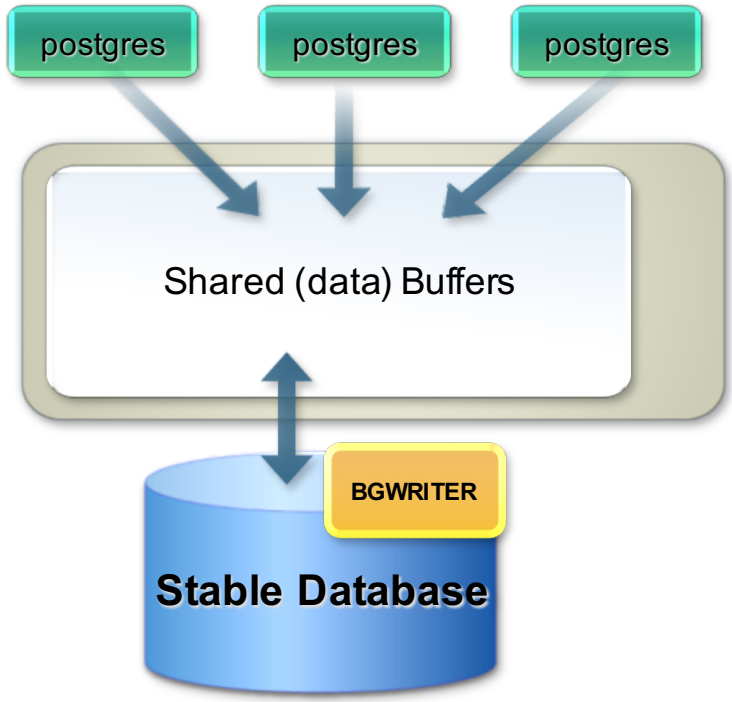
Disk Write Buffering

- Blocks are written to disk only when needed:
  - To make room for new blocks
  - At checkpoint time



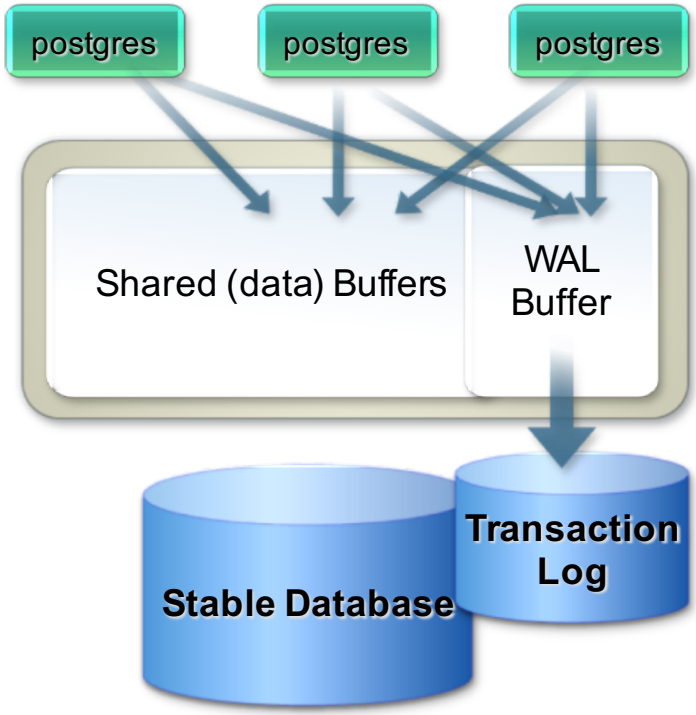
Background Writer Cleaning Scan

- Background writer scan attempts to ensure an adequate supply of clean buffers.
- Backend write dirty buffers at need.



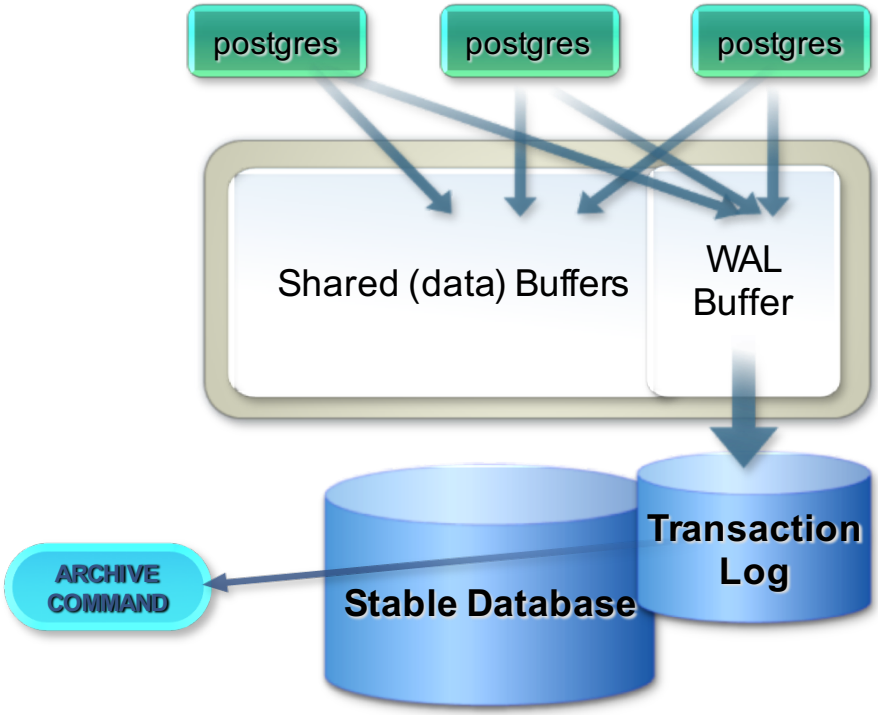
Write Ahead Logging (WAL)

- Backend write data to WAL buffers
- Flush WAL buffers periodically (WAL writer), on commit, or when buffers are full
- Group commit



Transaction Log Archiving

- Archiver spawns a task to copy away pg_xlog log files when full.



Commit & Checkpoint

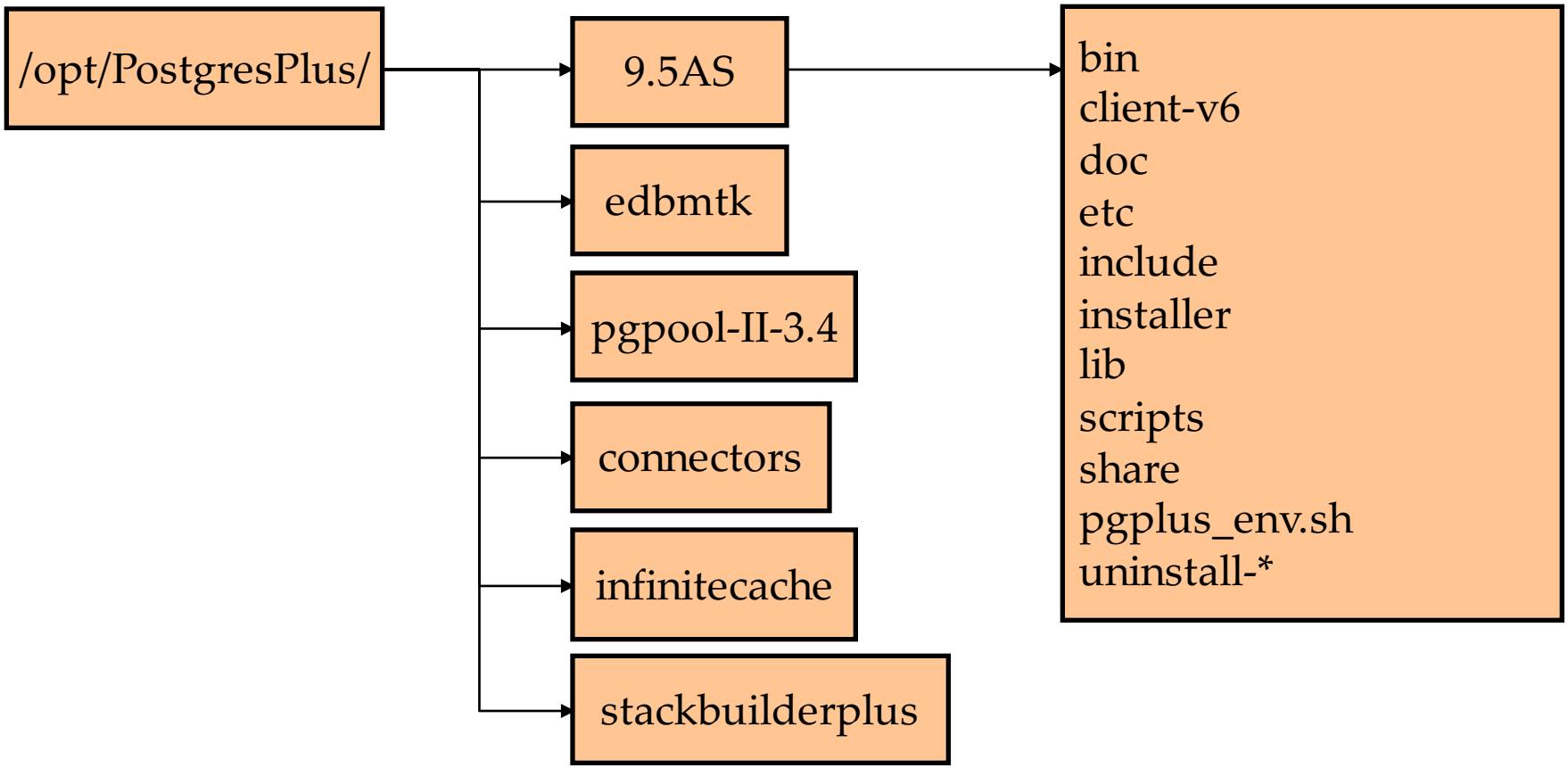
- Before commit
  - Uncommitted updates are in memory
- After commit
  - Committed updates written from shared memory to disk (write-ahead log file)
- After checkpoint
  - Modified data pages are written from shared memory to the data files

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”

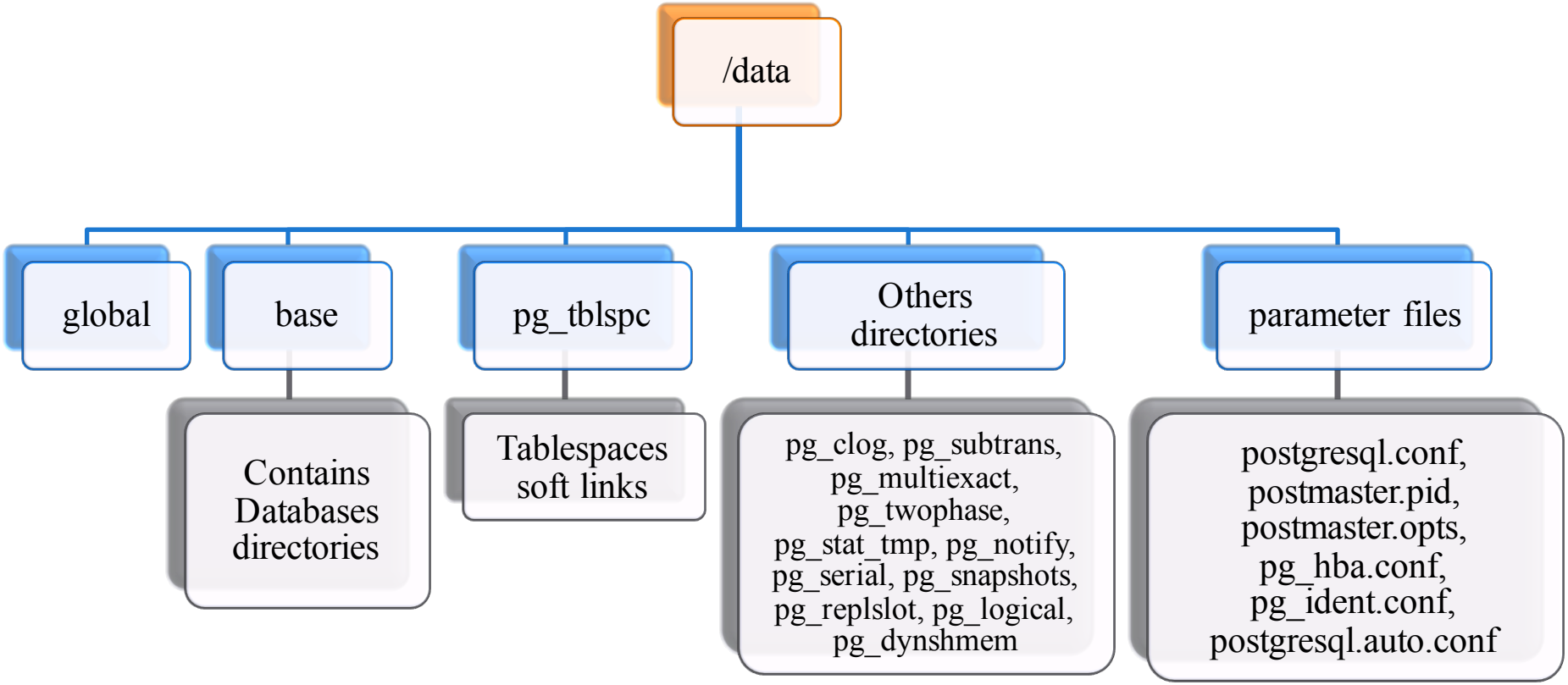
Physical Database Architecture

Installation Directory Structure



Physical Database Architecture

Cluster Data Directory Structure



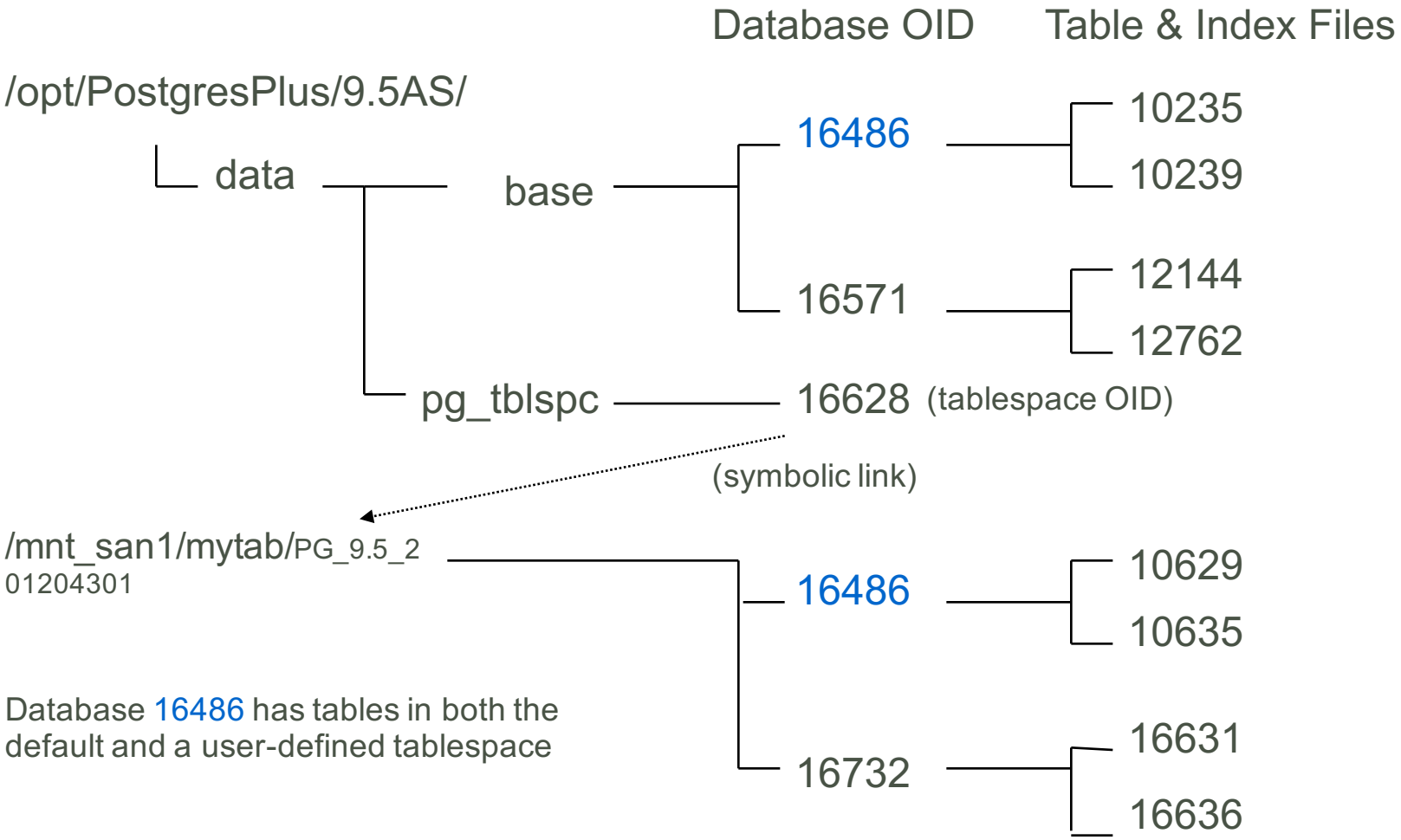
Physical Database Architecture

- 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
  - Advanced Server manages links to other filesystem directories
  - By default, indexes stored in same tablespace

Physical Database Architecture

- Each Relation has a free space map
  - Stores information about free space available in the relation
  - File named as filenode number plus the suffix _fsm
- Tables also have a visibility map
  - Track which pages are known to have no dead tuples.
  - Stored in a fork with the suffix _vm

Database Structure



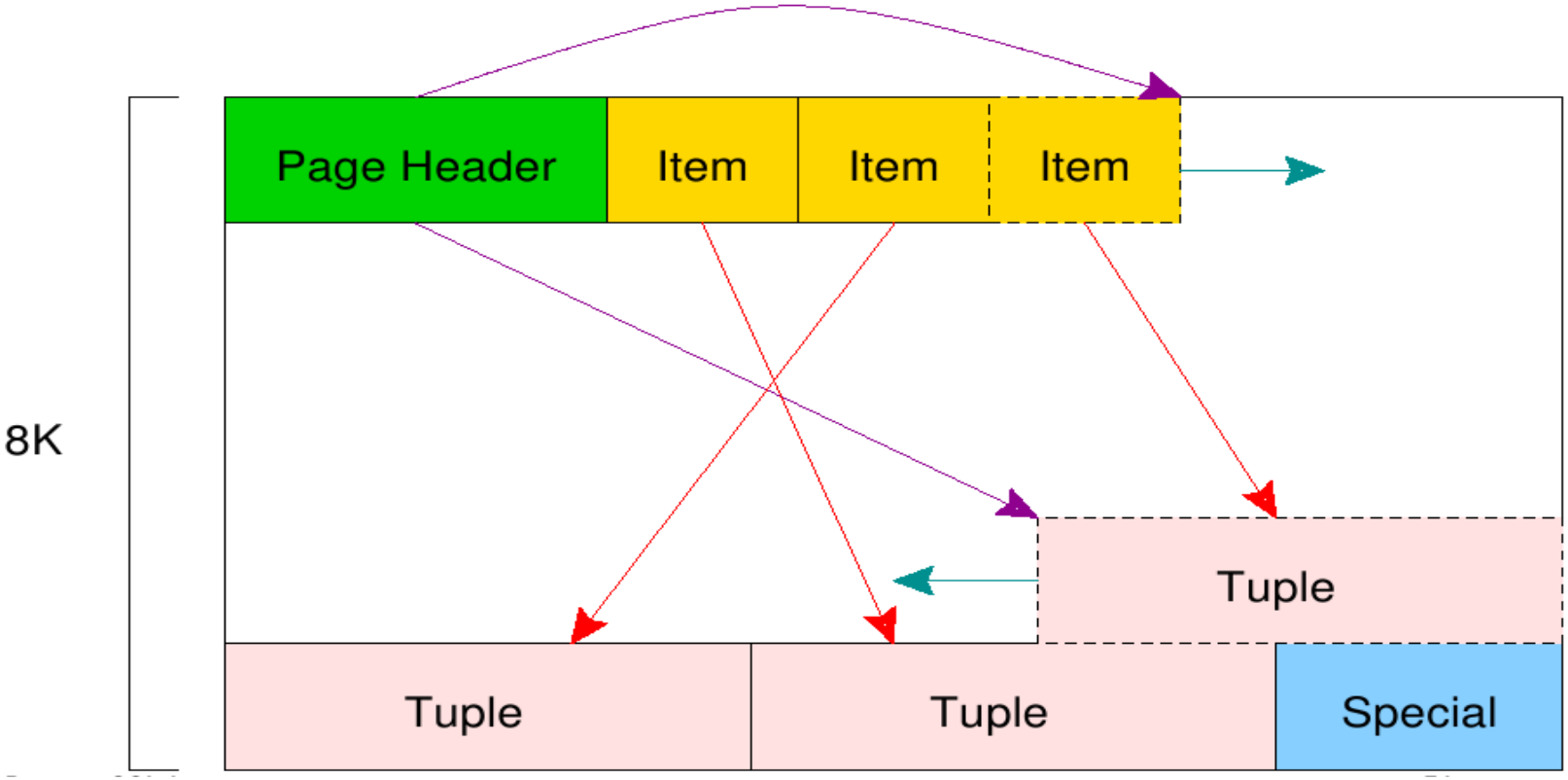
Data Files

- 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
  - Use pg_relation_filepath() function to view relation's data file location
- 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
  - 24 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



Installation

OS User & Permissions

- EDB Postgres Advanced Server runs as a daemon (Unix / Linux) or service (Windows)
- Installation requires superuser/admin access
- All EDB Postgres Advanced Server processes and data files must be owned by a user in the OS
  - OS user is un-related to database user accounts
  - For security reasons, the OS user must not be root or an administrative account
- SELinux Permissions
  - Before installing Advanced Server on a system that is running SELinux, you must set
  - SELinux to permissive mode.

Supported Platforms

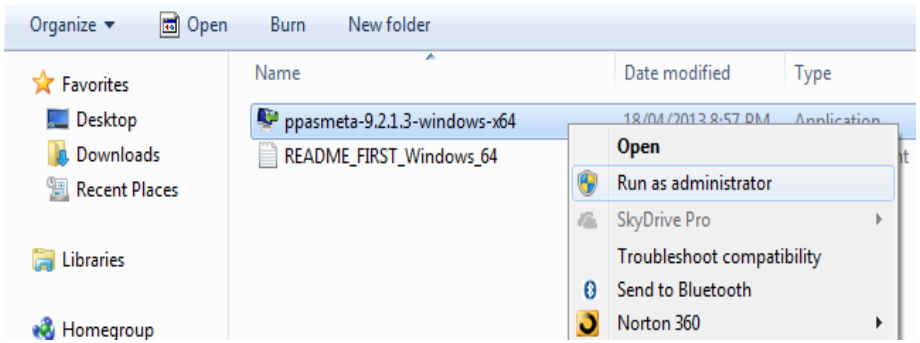
- EDB Postgres Advanced Server 9.5 is available for Windows, Linux.
- 64 bit Windows:
  - Windows 2012 R2 (64 bit)
  - Windows Server 2008 R2 (64 bit)
- 64 bit Linux platforms:
  - CentOS 6.x and 7.x
  - Red Hat Enterprise Linux 6.x and 7.x
  - OEL 6.x and 7.x
  - SLES 11.x and 12.x
  - Ubuntu 14.04

Supported Locales

- Advanced Server 9.x has been tested and certified for the following locales:
  - en_US United States English
  - zh_HK Traditional Chinese with Hong Kong SCS
  - zh_TW Traditional Chinese for Taiwan
  - zh_CN Simplified Chinese
  - ja_JP Japanese
  - ko_KR Korean

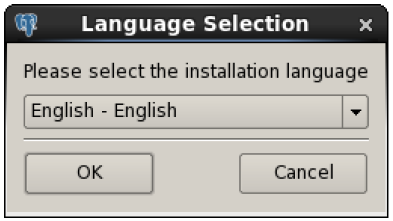
Windows Installation

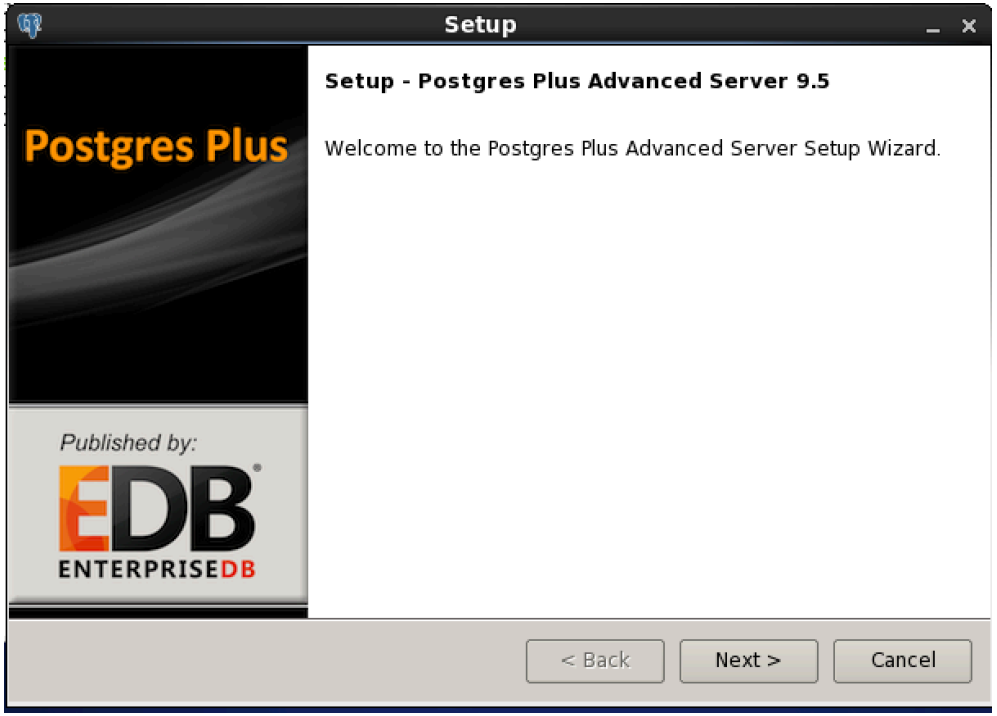
- Download the windows installer from http://www.enterprisedb.com/downloads/postgres-postgresql-downloads
- Extract ppasmeta-9.5.x.x-windows.zip
- Run ppasmeta-9.5.x.x-windows.exe as administrator



Windows Installation (cont)

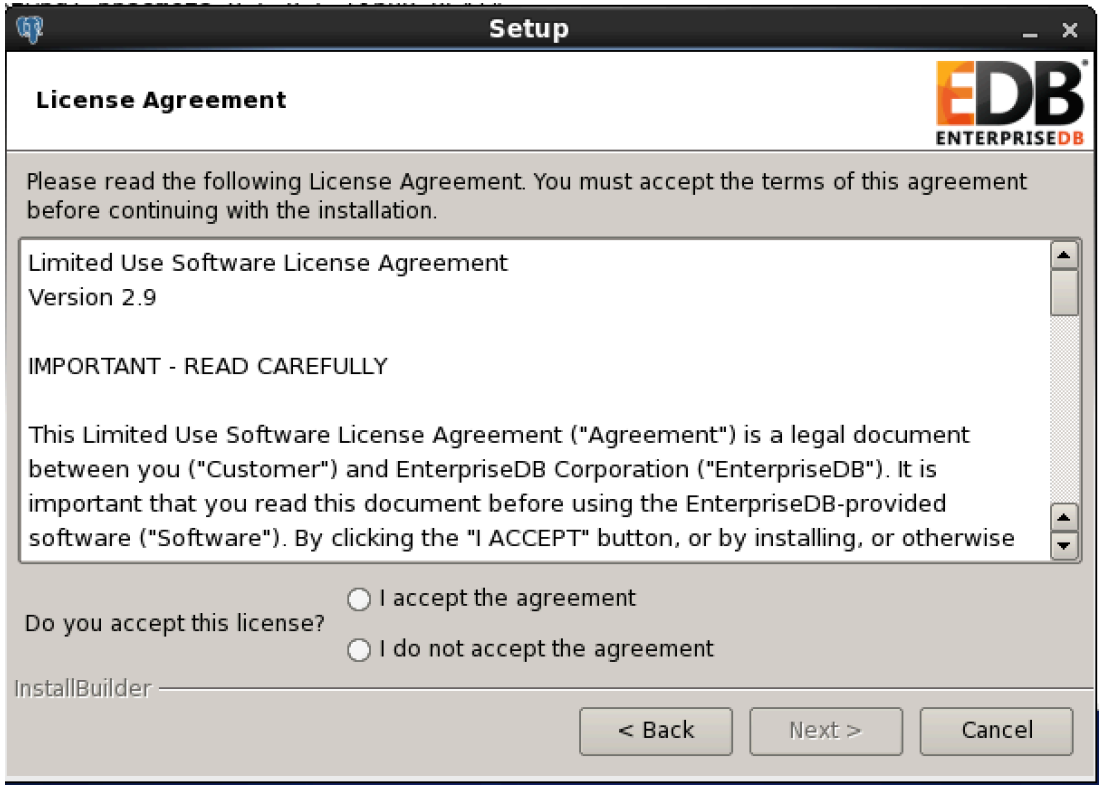
- Select the Language.
- Setup: Welcome message from the packaged installer.





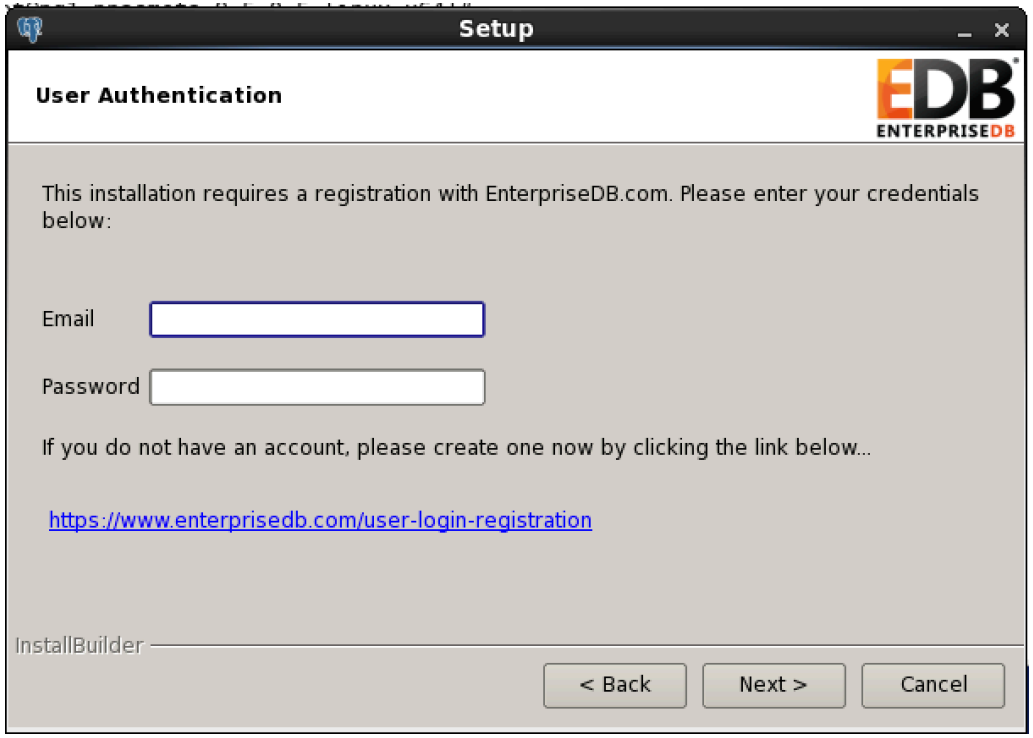
Windows Installation (cont)

- Accept the License Terms:



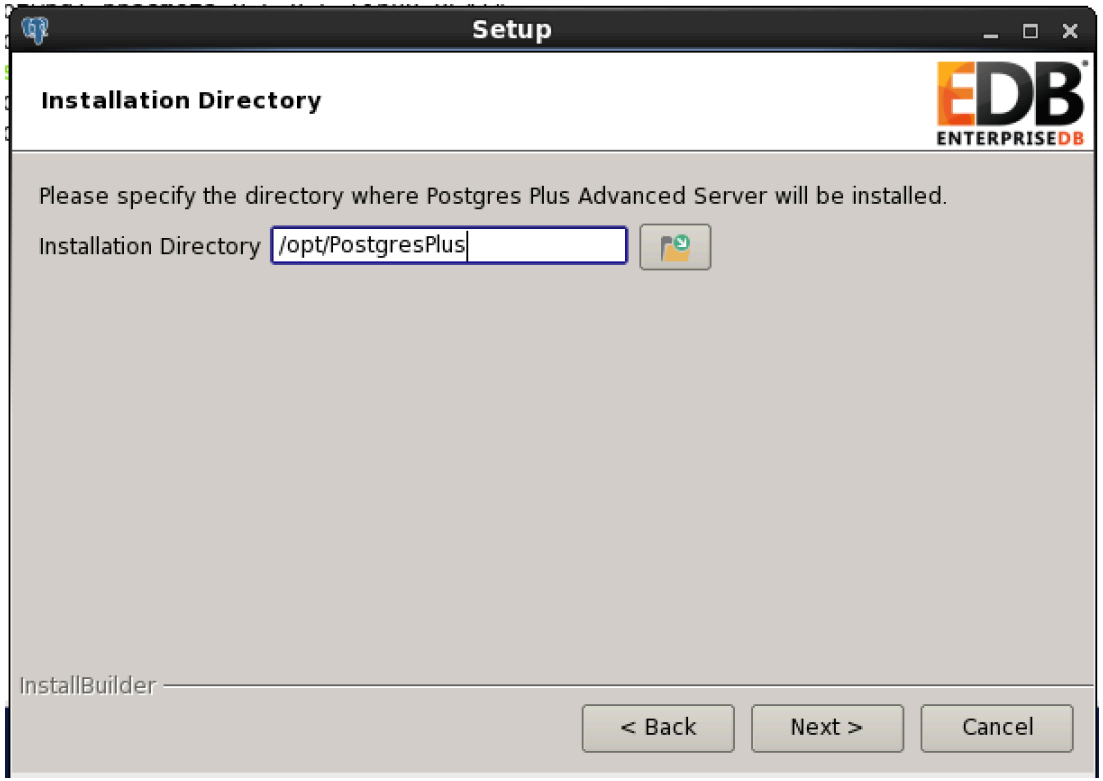
User Authentication

- You will need to register on www.enterprisedb.com to complete following step:



Windows Installation (cont)

- Installation Diretory: Choose the location where you want to install

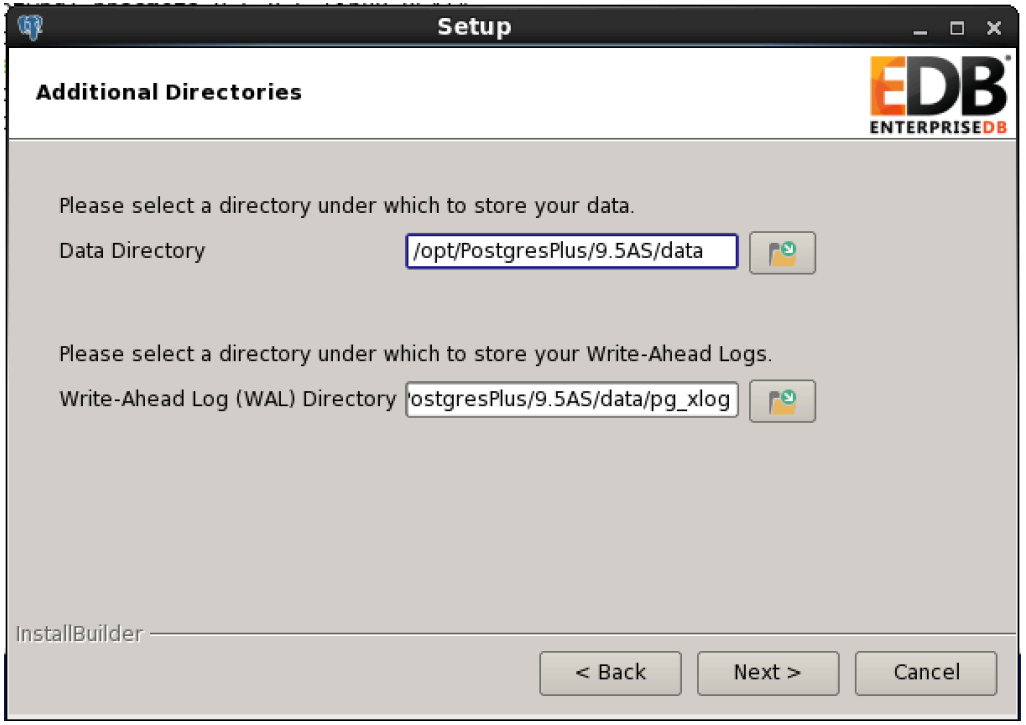


Windows Installation (cont)

- Select the Components:

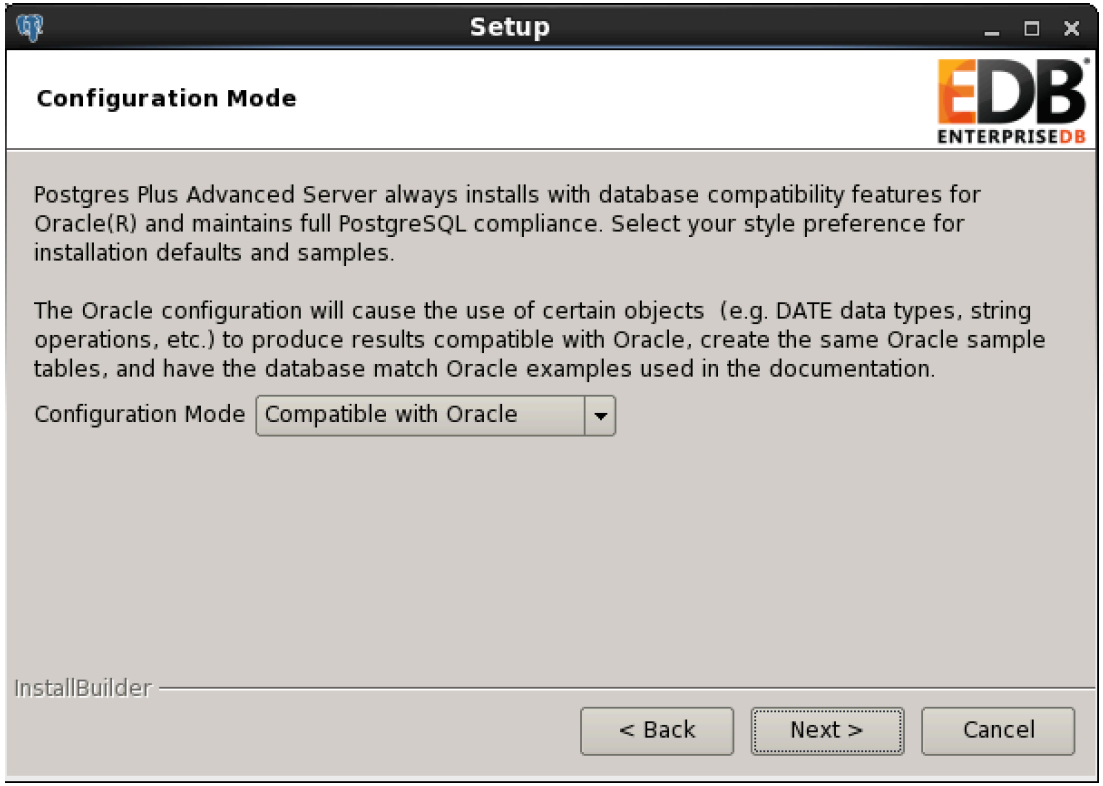
Windows Installation (cont)

- Data Directory: Choose the location where you want to install default database cluster
- Transaction Logs: Choose the location for xlogs



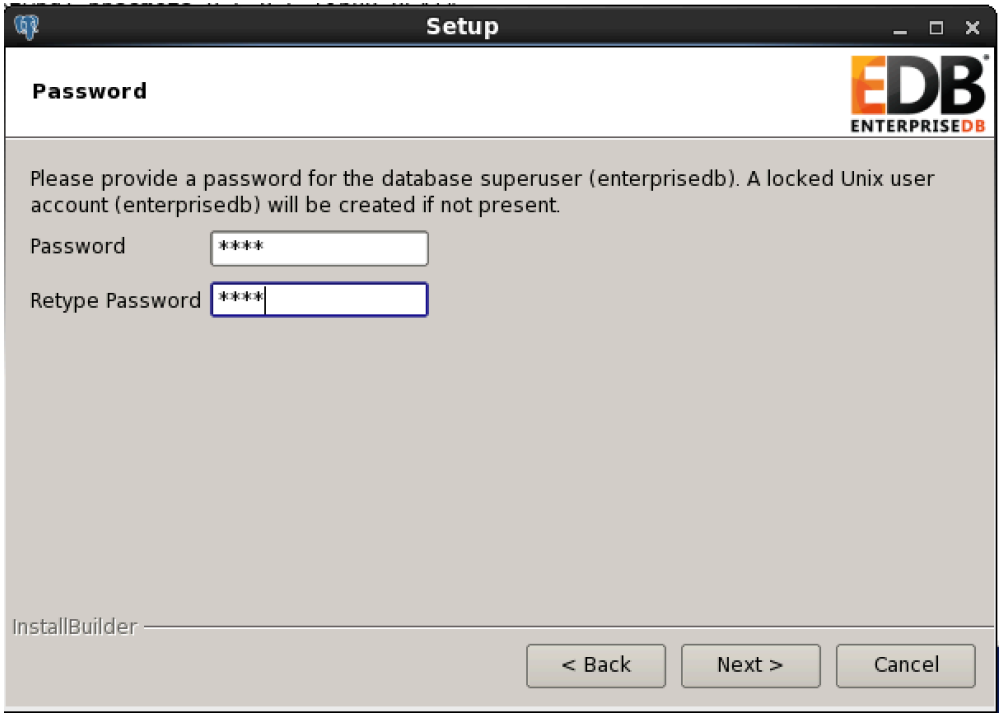
Windows Installation (cont)

- Choose the Configuration Mode:



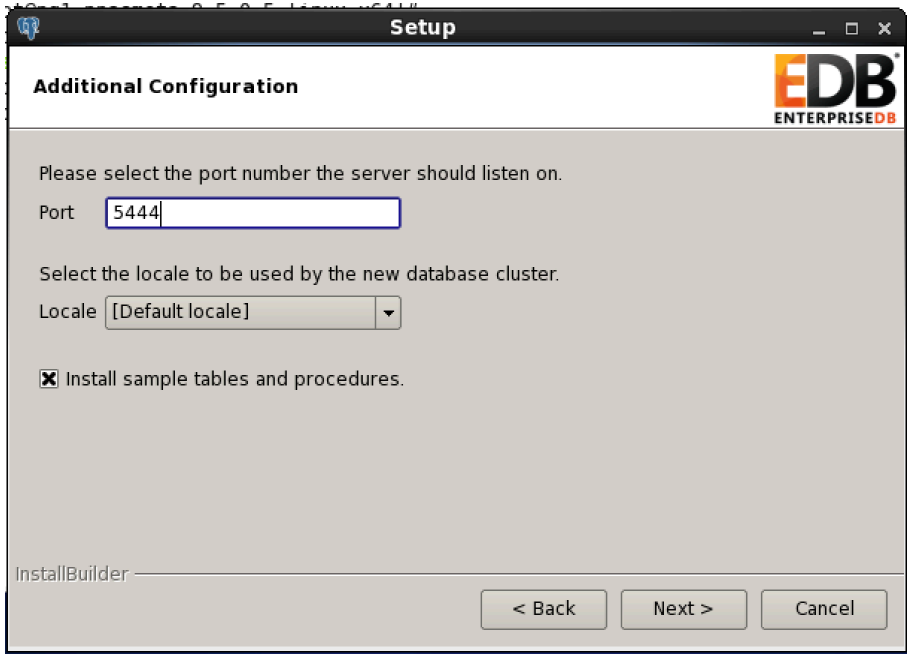
Windows Installation (cont)

- Provide the database Superuser and installation user password.
- Read the instruction given in snapshot.



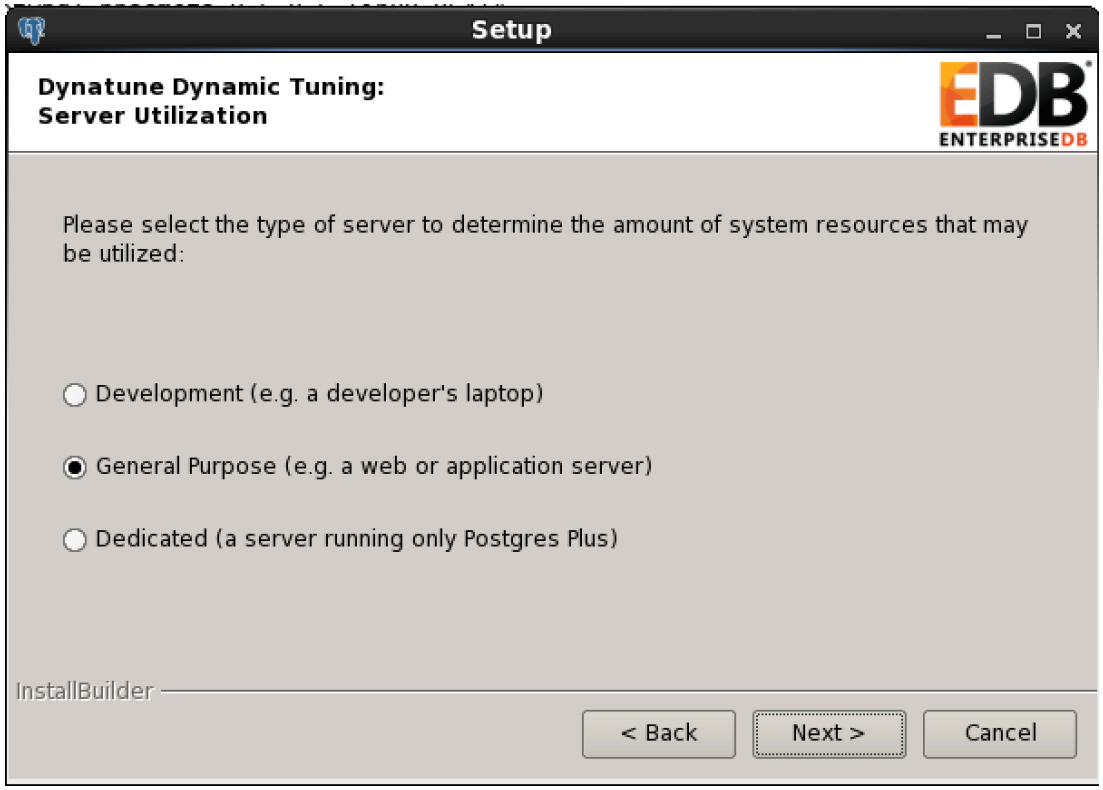
Windows Installation (cont)

- On the next step of wizard you can specify the port number on which your default cluster will run.
- Default port is 5444.
- Then you can choose the Locale and click next.



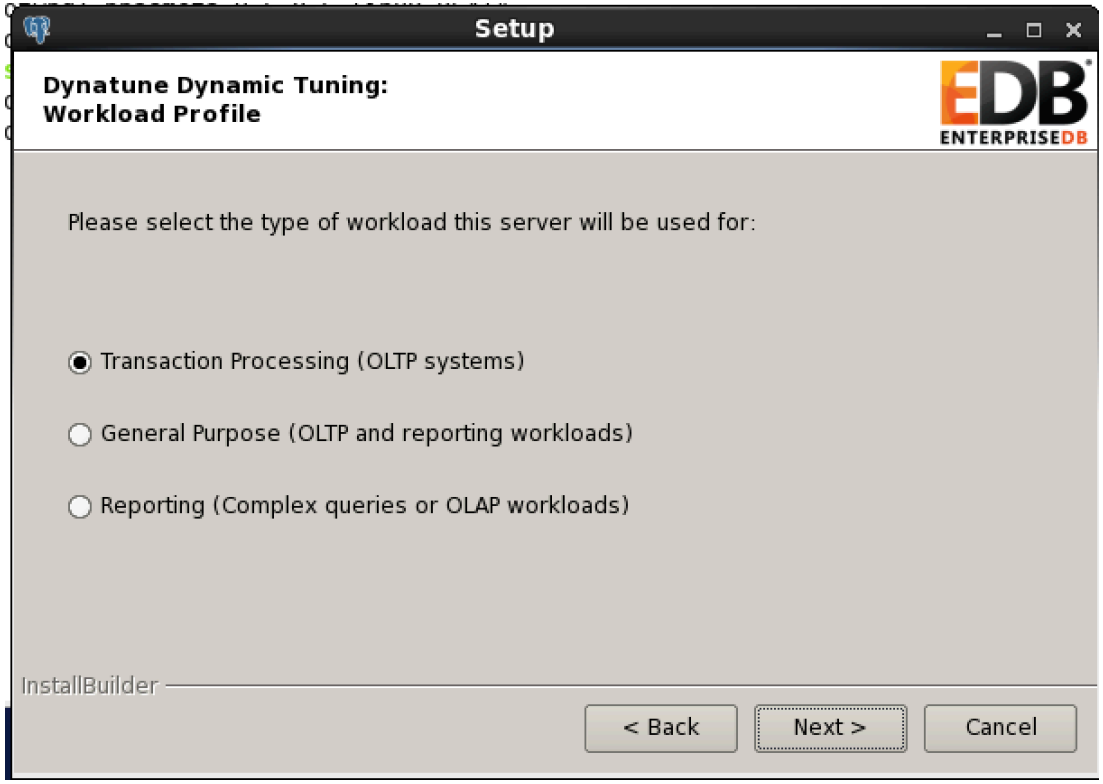
Windows Installation (cont)

- Choose the dynatune settings:



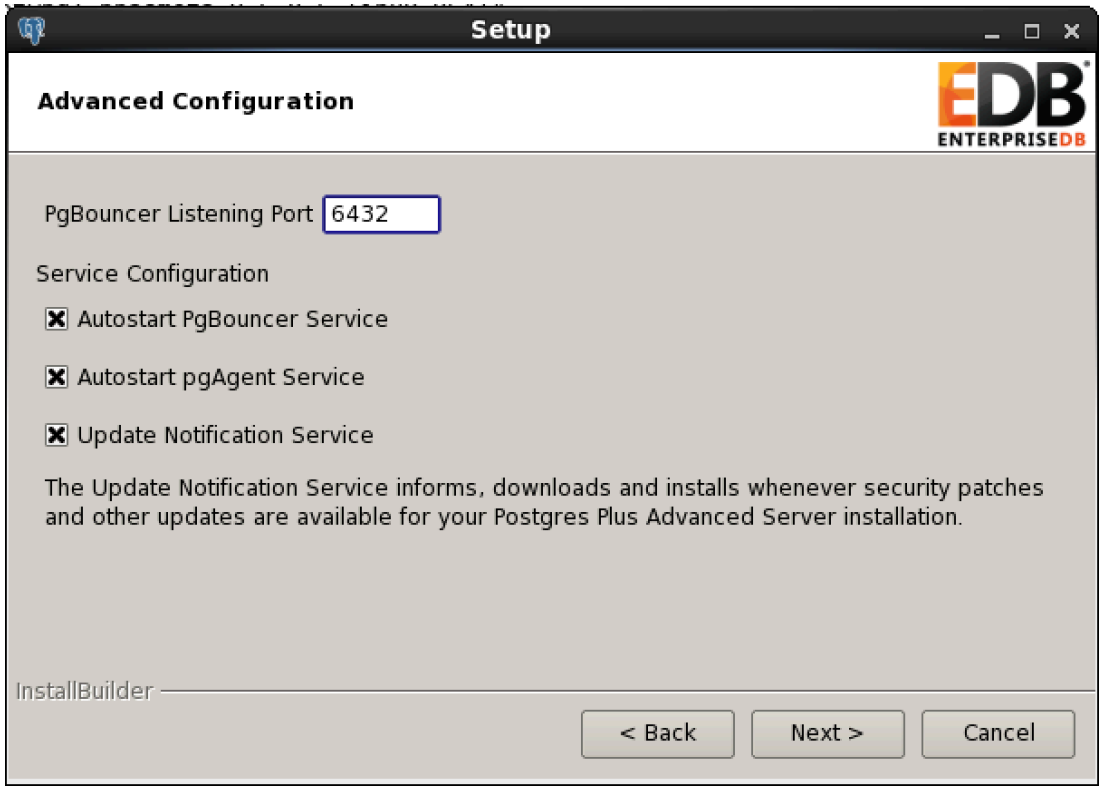
Windows Installation (cont)

- Choose the dynatune settings:



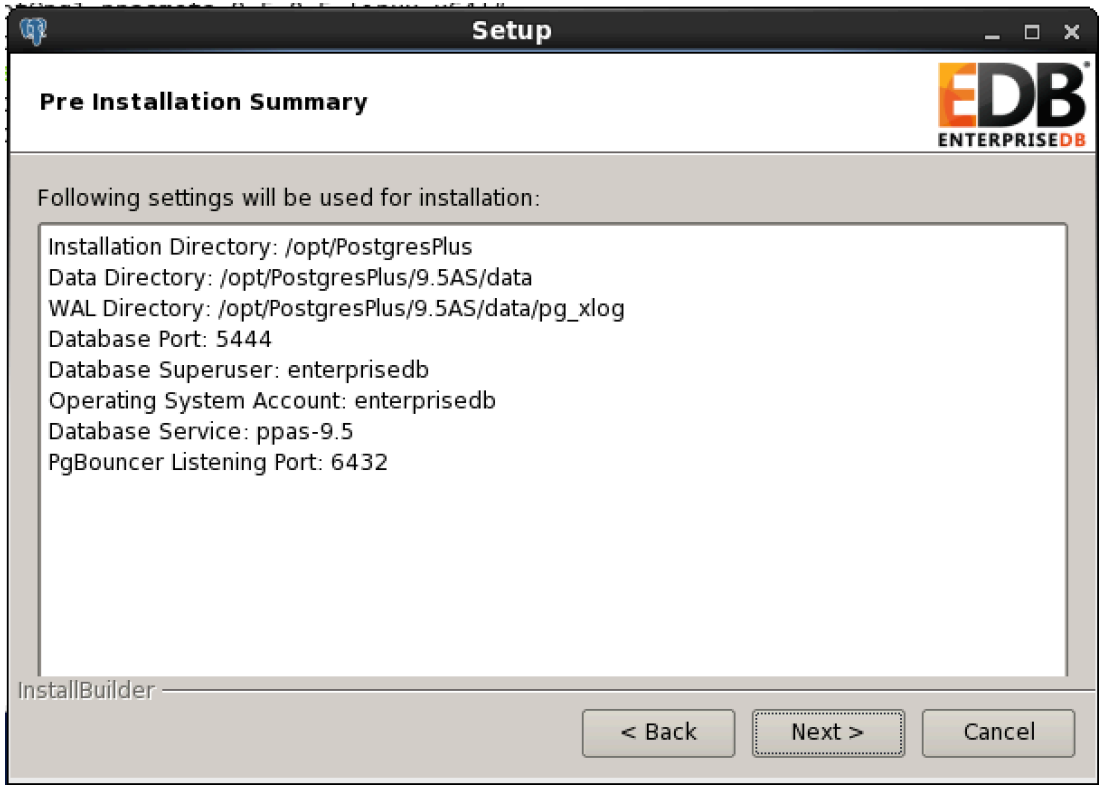
Windows Installation (cont)

- Choose the Advanced Configuration:



Windows Installation (cont)

- Review the installation settings and click next to install:



Linux Installation: Binary

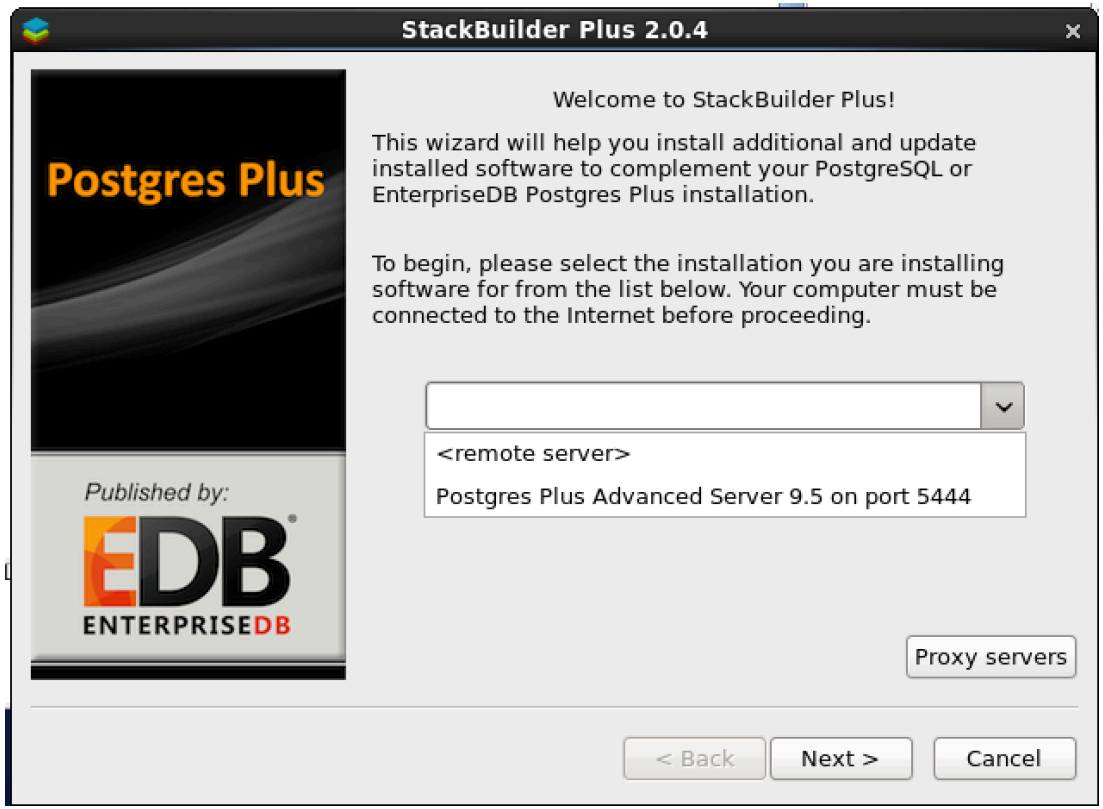
- Download Linux binary from: http://www.enterprisedb.com/downloads/postgres-postgresql-downloads
- Extract the installers from the binary.
- Run the binary and go through the setup wizard.
- The wizard steps will be same as windows setup wizard.

StackBuilder Plus

- Simplifies the process of downloading and installing modules
- StackBuilder Plus requires Internet access
- Run stackbuilder using the StackBuilder Plus menu option from the EDB Postgres Advanced Server 9.x menu

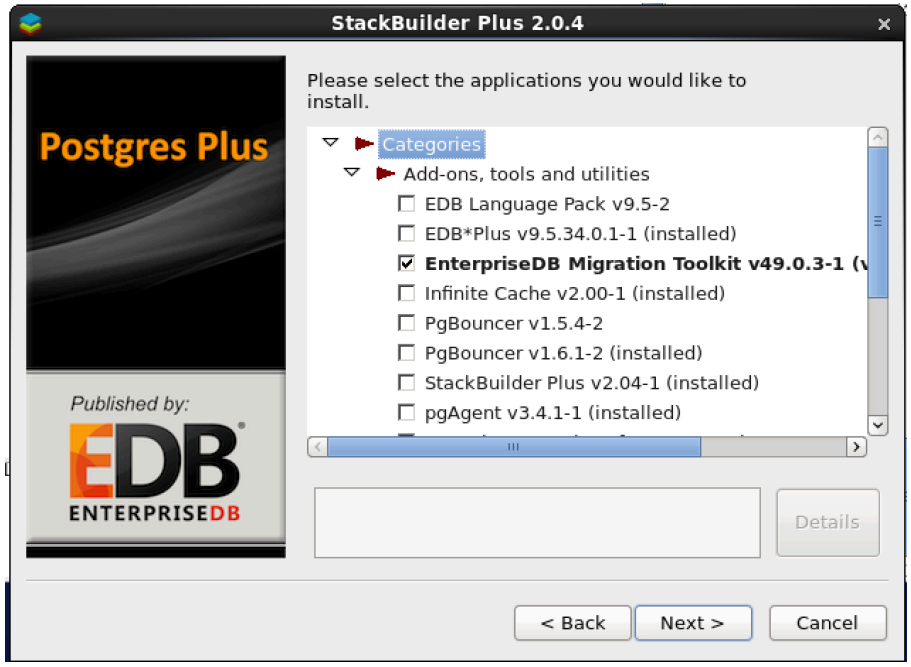
StackBuilder Plus

- Use the drop-down listbox on the welcome window to select your Advanced Server installation



StackBuilder Plus

- Each entry within the tree control is the name of a module
- Lists installed modules also
- Select the modules you want to install



Setting environmental variables

- Setting Environment Variables is very important for trouble free startup/shutdown of the database server.
  - PATH –should point correct bin directory.
  - PGDATA –should point correct data cluster directory.
  - PGPORT –should point correct port on which database cluster is running.
  - Edit .bash_profile to set the variables
  - In Windows set these variables using my computer properties page.

Clusters

- Each instance of Advanced Server is referred to as a "cluster"
- Comprised of a data directory that contains all data and configuration files
- Referred to in two ways
  - Location of the data directory
  - Port number
- A single server can have many installations and you can create multiple clusters using initdb.

Creating a Database Cluster

- Use initdb to create a database cluster. Must be run as the OS user that the instance will run as.

  initdb –D <data directory>

  - -D <data directory> - Database cluster directory
  - -U <super user> - Select the database super user name
  - -E <encoding> - Specify the database encoding
  - -n – No clean (do not clean up files in case of failure)

- After creating a new database cluster, modify postgresql.conf and pg_hba.conf, be sure to assign a unique port # to the cluster in postgresql.conf

Starting and Stopping the Server

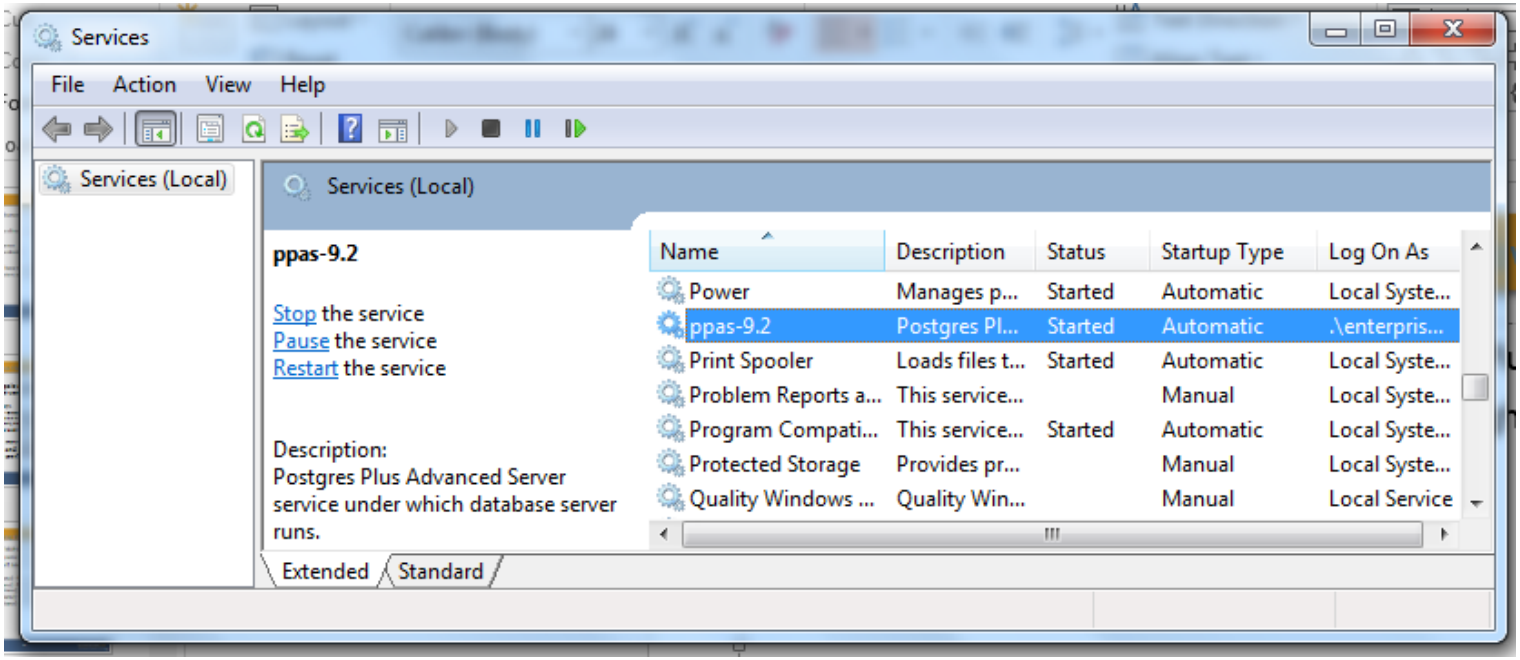
- Controlling a Service on Linux
  - service service_name action
  - /etc/init.d/service_name action
- Use pg_ctl to start and stop the server from the command line
  - pg_ctl start [<options>] – Start the server
  - pg_ctl stop [<options>] – Stop the server
  - pg_ctl restart [<options>] – Stop then start the server
  - pg_ctl status [<options>] – Display server status
  - pg_ctl reload [<options>] – Reload the configuration file

Starting and Stopping the Server (pg_ctl)

- Useful pg_ctl Parameters
  - -D <data directory> - Database cluster directory
  - -l <log file> - Write log output to the specified file
  - -m smart – Wait for clients to disconnect (default)
  - -m fast – Active tx rolled back and clients disconnected
  - -m immediate – Abort processes without clean shutdown

Starting and Stopping the Server(windows)

- Use Services Utility to start/stop EDB Postgres server
- The Services utility can be accessed through the Windows Control Panel



User Tools for Management

Command Line Tools

- EDB Postgres Advanced Server comes with variety of management tools and interfaces
- Command Line Tools:
  - EDB*Plus
  - EDB-PSQL

Lets discuss them in detail

EDB*Plus

- EDB*Plus is a utility program that provides a command line user interface to the EDB Postgres Advanced Server.
- EDB*Plus accepts SQL commands, SPL anonymous blocks, and EDB*Plus commands.
- EDB*Plus commands are compatible with Oracle SQL*Plus commands

Starting EDB*Plus

- EDB*Plus can be started by selecting it from the application menu
- EDB*Plus program is invoked by running *edbplus* from the edbplus subdirectory
  - **edbplus** [-S[ILENT]] [*login* | /NOLOG] [*@scriptfile*[*.ext*]]
  - **Login**: *username*[/*password*][@{*connectstring* | *variable* }]
  - **Connection string**: *host*[:*port*][/*dbname*]]
  - **CONNECT** username[/password][@connect_identifier] where connect_identifier syntax is as follows: host[:port][/db_name]

EDB*Plus SET Command

- Sets a session level variable to control certain aspects of EDB*Plus behavior as follows:

SET AUTO[COMMIT] {ON | OFF | IMMEDIATE | statement_count}

- Note that EDB*Plus always autocommits ddl statements. IMMEDIATE has the same effect as ON. statement_count cause EDB*Plus to issue a commit after N successful SQL statements.

SET ECHO {ON | OFF}

- Determines whether SQL & EDB*Plus script statements are shown to the screen as they are executed. The default is OFF.

EDB*Plus Commands

- Edb*Plus support variety of Oracle Compatible commands like:
  - ACCEPT
  - APPEND
  - DEFINE
  - LIST
  - CHANGE
  - CLEAR
  - COLUMN
  - DEL
  - DESCRIBE
  - DISCONNECT
  - EDIT
  - EXIT
  - GET
  - HELP
  - HOST
  - INPUT

EDB*Plus Commands Example

```
SQL> LIST
1 SELECT empno, ename, job, sal, comm
2 FROM emp
3 WHERE deptno = 20
4* ORDER BY empno
SQL> 3
3* WHERE deptno = 20
SQL> CHANGE /20/30/
3* WHERE deptno = 30
SQL> LIST
1 SELECT empno, ename, job, sal, comm
2 FROM emp
3 WHERE deptno = 30
4* ORDER BY empno
```

EDB-PSQL

- edb-psql is the name of EnterpriseDB PSQL's executable.
- EnterpriseDB PSQL provides a terminal-based front-end to Advanced Server.
- It enables you to type in queries interactively, issue them to EnterpriseDB, and see the query results.
  - *edb-psql [option...] [dbname [username]]*

Connecting To A Database

- In order to connect to a database you need to know the name of your target database, the host name and port number of the server and what user name you want to connect as.
- Command line options, namely -d, -h, -p, and -U respectively.
- Environment variables PGDATABASE, PGHOST, PGPORT and PGUSER to appropriate values.

Conventions

- psql has it's own set of commands, all of which start with a backslash (\). These are in no way related to SQL commands, which operate in the server. Psql commands only affect psql.
- Some commands accept a pattern. This pattern is a modified regex. Key points:
  - * and ? are wildcards
  - Double-quotes are used to specify an exact name, ignoring all special characters and preserving case

On startup...

- psql will execute commands from $HOME/.psqlrc, unless option -X is specified.
- -f FILENAME will execute the commands in FILENAME, then exit
- -c COMMAND will execute COMMAND (SQL or internal) and then exit
- --help will display all the startup options, then exit
- --version will display version info and then exit

Entering commands

- psql uses the command line editing capabilities that are available in the native OS. Generally, this means
- Up and Down arrows cycle through command history
  - on UNIX, there is tab completion for various things, such as SQL commands and to a more limited degree, table and field names
  - disabled with -n
- \s will show the command history
- \s FILENAME will save the command history
- \e will edit the query buffer and then execute it
- \e FILENAME will edit FILENAME and then execute it
- \w FILENAME will save the query buffer to FILENAME

Output

- There are numerous ways to control output.
- The most important are:
  - -o FILENAME or \o FILENAME will send query output (excluding STDERR) to FILENAME (which may be a pipe)
  - \g FILENAME executes the query buffer, sending output to FILENAME (may be a pipe)
  - -q runs quietly.

Information Commands

- \d(i, s, t, v, S)[+] [pattern]
  - List information about indexes, sequences, tables, views or System objects. Any combination of letters may be used in any order, e.g.: \dvs
  - + displays comments
- \d[+] [pattern]
  - For each relation describe/display the relation structure details
  - + displays any comments associated with the columns of the table, and if the table has an OID column
  - without a pattern, \d[+] is equivalent to \dtvs[+]

Information Commands (Cont'd)

- \l[ist][+]
  - List the names, owners,
  - and character set encodings of all the databases in the server.
  - If + is appended to the command name,
  - database descriptions are also displayed.
- \dn+ [pattern]
  - Lists schemas (namespaces)
  - + adds permissions and description to output
- \df[+] [pattern]
  - Lists functions
  - + adds owner, language, source code and description to output

Other common psql commands

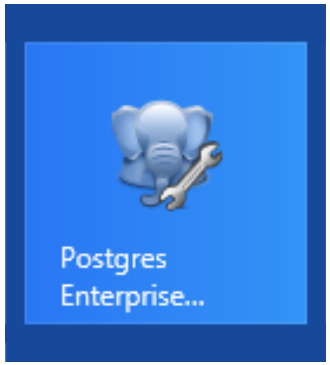
- \q or ^d
  - Quits the psql program.
- \cd [directory]
  - Change current working directory
  - Tip: To print your current working directory, use \! pwd.
- \! [command]
  - Executes the specified command
  - If no command is specified, escapes to a separate Unix shell (CMD.EXE in Windows)

Help

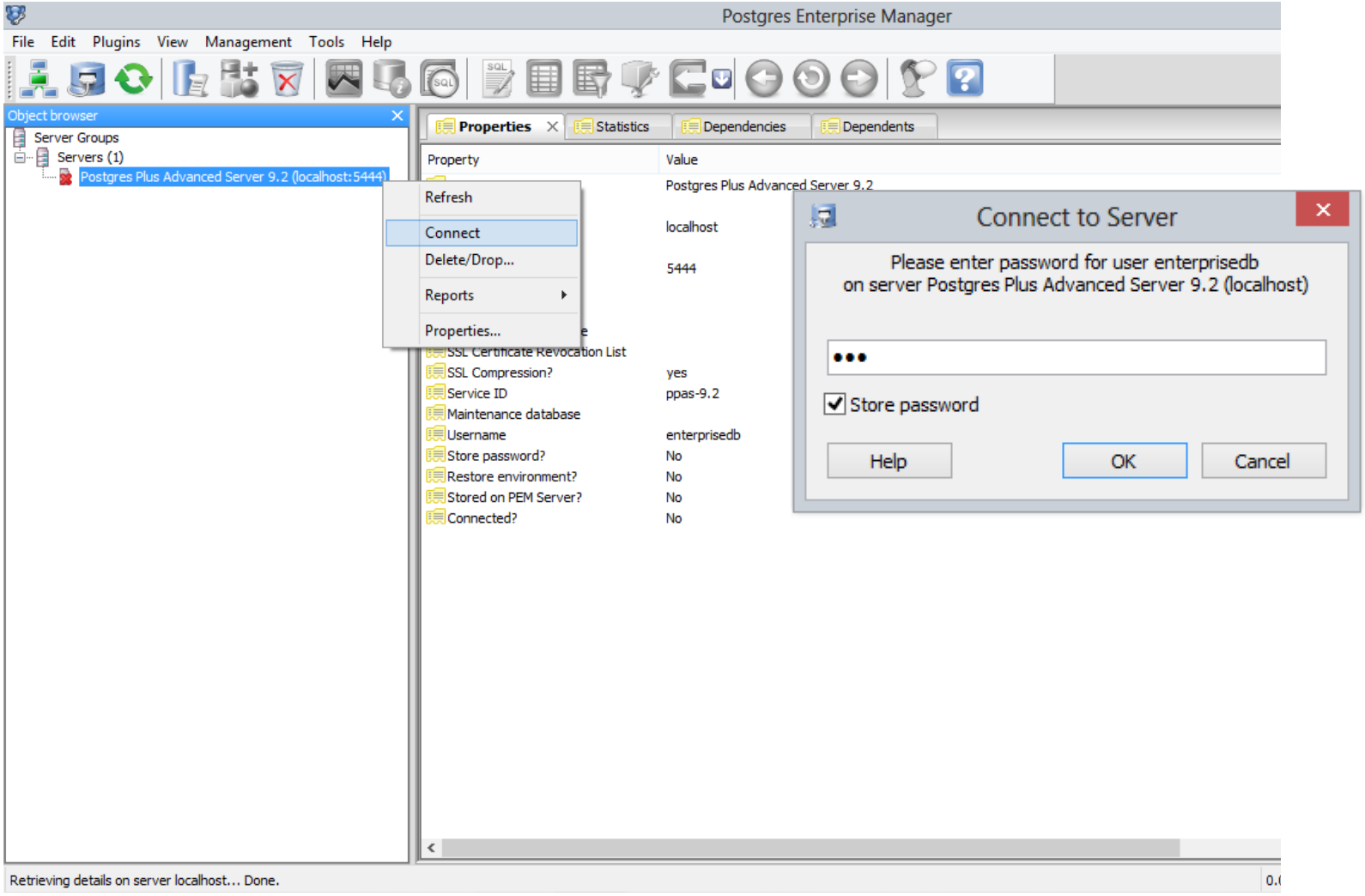
- \?
  - Shows help information about psql commands
- \h [command]
  - Shows information about SQL commands
  - If command isn't specified, lists all SQL commands
- psql --help
  - Lists command line options for psql

GUI Tool: PEM

- EDB Postgres Advanced server comes with Postgres Enterprise Manager Client tool for managing and monitoring
- PEM provides comprehensive database administration, query, and maintenance consoles
- Run PEM Client from start menu

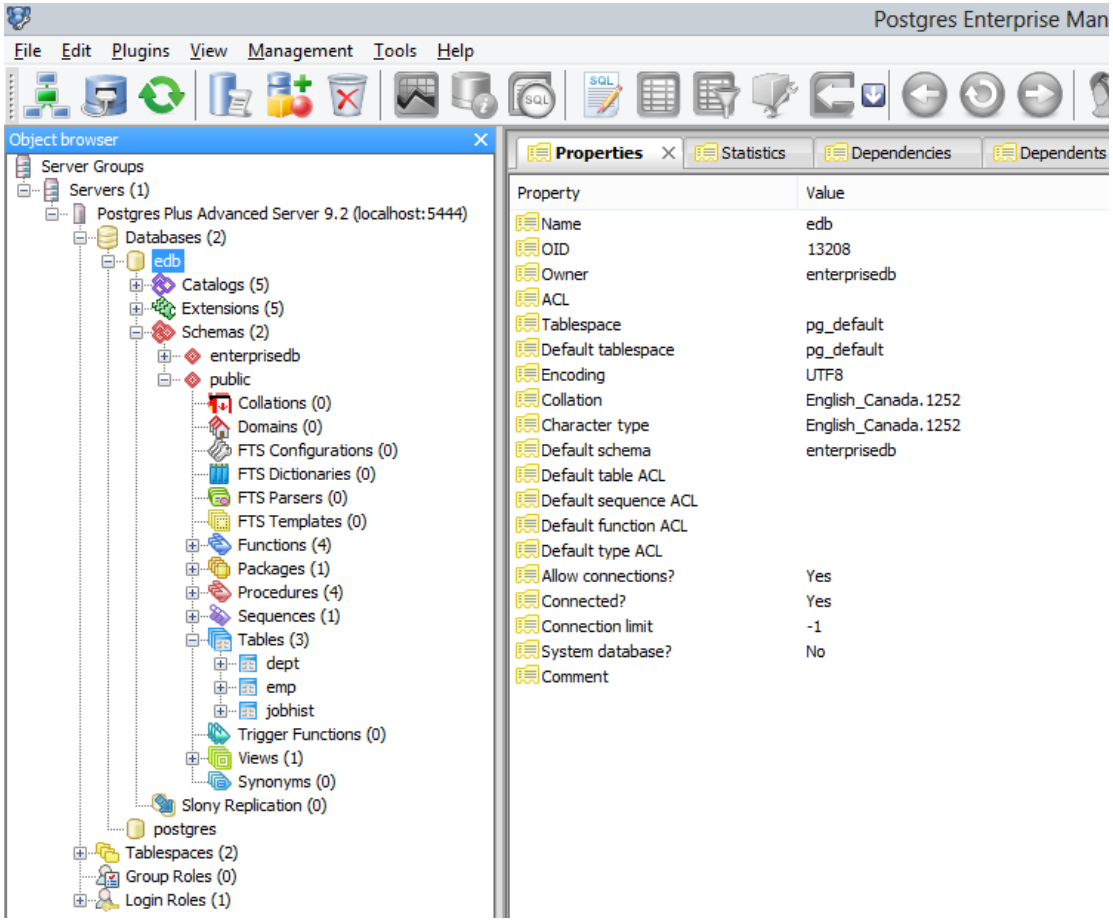


PEM



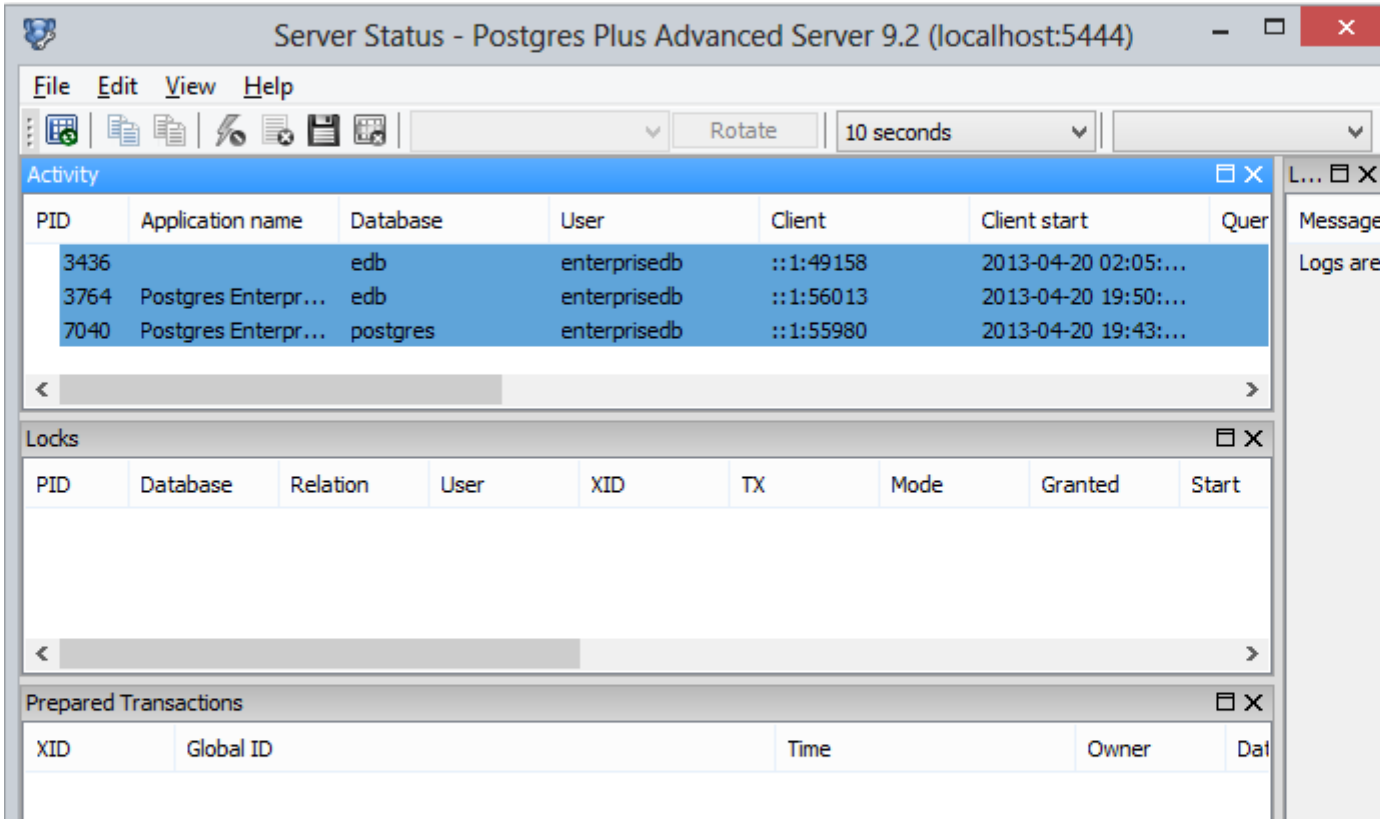
PEM

- Using the Object browser you can browse through the database cluster



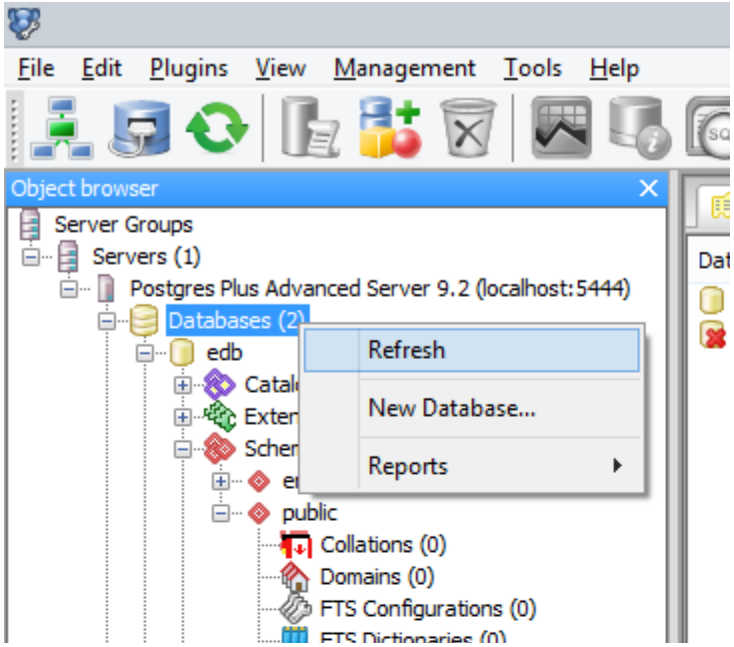
PEM

- Go to tools and click server status



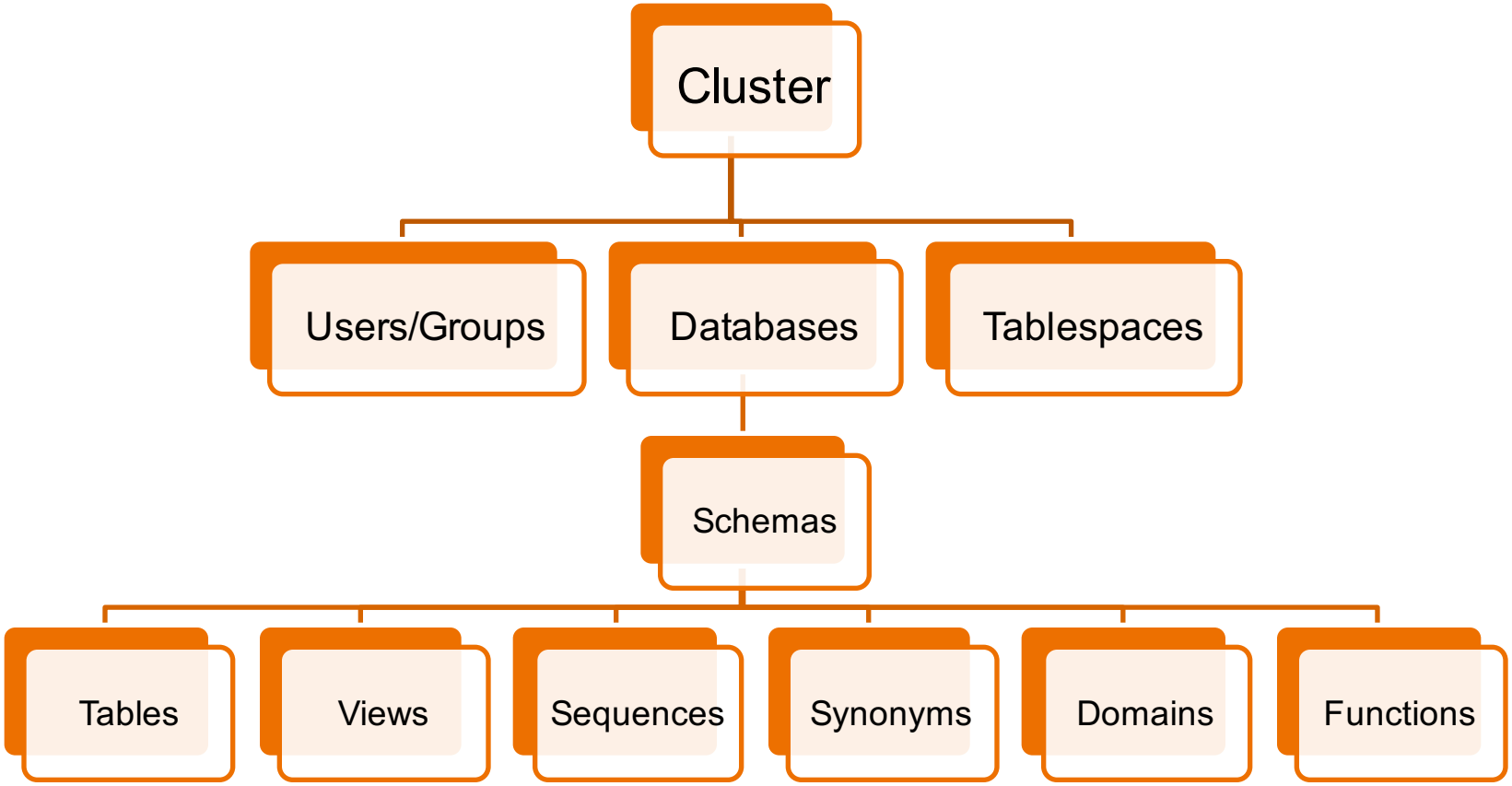
PEM

- Create database, database objects, users/roles and tablespaces by a simple right click in the object browser
- E.g. Right click on databases and create new databases



Creating and Managing Advanced Server Databases

Object Hierarchy



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”

Database

- A running EDB Postgres server can manage many databases
- A Database is a named collection of SQL objects. It is a collection of schemas and the schemas contain the tables, functions, etc
- Databases are created with CREATE DATABASE command
- Databases are destroyed with DROP DATABASE command
- To determine the set of existing databases:
  - SQL: SELECT datname FROM pg_database;
  - EDB-PSQL META COMMAND: \l (backslash lowercase L)

Creating Databases

- There is a program that you can execute from the shell to create new databases, createdb.
  - createdb dbname
- Create Database command can be used to create a database in a cluster.
  - Syntax:

```
CREATE DATABASE name
  [ [ WITH ] [ OWNER [=] dbowner ]
     [ TEMPLATE [=] template ]
     [ ENCODING [=] encoding ]
     [ TABLESPACE [=] tablespace ]
     [ CONNECTION LIMIT [=] connlimit ] ]
```

Accessing a Database

- edb-psql tool allows you to interactively enter, edit, and execute SQL commands.
- PEM GUI tool can also be used to access a database.
- Lets use psql to access a database:
  - Open Command prompt or terminal.
  - If PATH is not set you can execute next command from the bin directory location of EDB Postgres installation
  - $*edb-psql –U enterprisedb edbstore*
  - edbstore=#

Creating Schemas

- Schemas:
  - A database contains one or more named schemas, which in turn contain tables. Schemas also contain other kinds of named objects, including data types, functions, and operators.
  - There are several reasons why one might want to use schemas:
    - To allow many users to use one database without interfering with each other.
    - To organize database objects into logical groups to make them more manageable.
    - Third-party applications can be put into separate schemas so they cannot collide with the names of other objects.

Creating Schemas (cont)

- To create a schema, use the CREATE SCHEMA command. Give the schema a name of your choice.
- Syntax:
  - CREATE SCHEMA schemaname [AUTHORIZATION username] [schema_element [...]]
  - CREATE SCHEMA AUTHORIZATION username [schema_element [...]]
- For example:
  - CREATE SCHEMA edb AUTHORIZATION goldy_dba;
  - CREATE SCHEMA pgplus;

Schema Search Path

- Qualified names are tedious to write, so we use table names directly in queries.
- If no schema name is given, the schema search path determines which schemas are searched for matching table names
  - E.g.:

    SELECT * FROM employee
    - This statement will find the first employee table the schemas listed in the search path

Schema Search Path

- The first schema named in the search path is called the current schema if that named schema exist.
- Aside from being the first schema searched, it is also the schema in which new tables will be created if the CREATE TABLE command does not specify a schema name.
- To show the current search path, use the following command:
  - SHOW search_path;
- To put our new schema in the path, we use:
  - SET search_path TO myschema, public;

Database Objects

- Database Schemas can contain different types of objects including:
  - Tables
  - Sequences
  - Views
  - Synonyms
  - Domains
  - Packages
  - Functions
  - Procedures

Configuration

Setting Advanced Server Parameters

- There are many configuration parameters that effect the behavior of the database system.
- All parameter names are case-insensitive.
- Every parameter takes a value of one of five types:
  - boolean
  - integer
  - floating point
  - string
  - enum
- One way to set these parameters is to edit the file postgresql.conf, which is normally kept in the data directory.

Setting Advanced Server Parameters (cont.)

- Some parameters can be changed per session using SET command
- Some parameters can be changed at user level using ALTER USER
- Some parameters can be changed at the database level using ALTER DATABASE
- SHOW command can be used to see settings
- pg_settings catalog table lists settings information

Connection Settings

- Following Parameters in postgresql.conf file control the connection settings:
  - **listen_addresses (string)**
    - Specifies the TCP/IP address(es) on which the server is to listen for connections
  - **port (integer)**
    - The TCP port the server listens on; 5432 by default.
  - **max_connections (integer)**
    - Determines the maximum number of concurrent connections to the database server
  - **superuser_reserved_connections (integer)**
    - Determines the number of connection "slots" that are reserved for connections by Advanced Server superusers..

Security and Authentication Settings

- Following Parameters in postgresql.conf file control the Security and Authentication Settings
  - **authentication_timeout (default: 1 minute):** Maximum time to complete client authentication, in seconds.
  - **ssl (default: off).** Enables SSL connections.
  - **ssl_ca_file:** Specifies the name of the file containing the SSL server certificate authority (CA).
  - **ssl_cert_file:** Specifies the name of the file containing the SSL server certificate.
  - **ssl_ciphers:** List of SSL ciphers that may be used for secure connections.
  - **ssl_renegotiation_limit (default 512MB):** Specifies how much data can flow over an SSL-encrypted connection before renegotiation of the session keys will take place.

Memory Settings

- Following Parameters in postgresql.conf file control the Memory settings:
  - **shared_buffers (integer)**
    - Sets the amount of memory the database server uses for shared memory buffers.
  - **temp_buffers (integer)**
    - Sets the maximum number of temporary buffers used by each database session.
  - **max_prepared_transactions (integer)**
    - Sets the maximum number of transactions that can be in the "prepared" state simultaneously
  - **temp_file_limit (default -1):**
    - amount of disk space that a session can use for temporary files. A transaction attempting to exceed this limit will be cancelled. Default is unlimited.
  - **work_mem (integer)**
    - Specifies the amount of memory to be used by internal sort operations and hash tables before switching to temporary disk files.
  - **maintenance_work_mem (integer)**
    - Specifies the maximum amount of memory to be used in maintenance operations, such as VACUUM, CREATE INDEX, and ALTER TABLE ADD FOREIGN KEY.

Query Planner Settings

- **random_page_cost** (default 4.0): Estimated cost of a random page fetch, in abstract cost units. May need to be reduced to account for caching effects.
- **seq_page_cost** (default 1.0): Estimated cost of a sequential page fetch, in abstract cost units. May need to be reduced to account for caching effects. Must always set random_page_cost >= seq_page_cost.
- **effective_cache_size** (default 128M): Used to estimate the cost of an index scan. Rule of thumb is 75% of system memory.
- There are plenty of enable_* parameters which influence the planner in choosing an optimal plan.
- **enable_indexonlyscan** available in 9.2, enables or disables the query planner's use of index-only-scan plan types

Write Ahead Logs Settings

- **wal_level** (default: minimal). Determines how much information is written to the WAL. Change this to enable replication. Other values are archive and hot_standby.
- **fsync** (default on): Turn this off to make your database much faster – and silently cause arbitrary corruption in case of a system crash.
- **wal_buffers** (default: -1, autotune): The amount of memory used in shared memory for WAL data. The default setting of -1 selects a size equal to 1/32nd (about 3%) of shared_buffers
- **checkpoint_segments** (default 3): Maximum number of 16MB WAL file segments between checkpoints. Default is too small!
- **checkpoint_timeout** (default 5 minutes): Maximum time between checkpoints.

Logging

- **log_destination**: Valid values are combinations of stderr, csvlog, syslog, and eventlog, depending on platform.
- **logging_collector**: Enables advanced logging features. csvlog requires logging_collector.
- **log_directory**: Directory where log files are written. Requires logging collector.
- **log_filename**: Format of log file name (e.g. postgresql-%Y-%M-%d.log). Allows regular log rotation. Requires logging collector.
- **log_file_mode**: (default 0600): On Unix systems this parameter sets the permissions for log files when logging_collector is enabled.
- **log_rotation_age**: Automatically rotate logs after this much time. Requires logging_collector.
- **log_rotation_size**: Automatically rotate logs when they get this big. Requires logging_collector.

Logging (Cont'd)

- **client_min_messages** (default NOTICE). Messages of this severity level or above are sent to the client.
- **log_min_messages** (default WARNING). Messages of this severity level or above are sent to the server.
- **log_min_error_statement** (default ERROR). When a message of this severity or higher is written to the server log, the statement that caused it is logged along with it.
- **log_min_duration_statement** (default -1, disabled): When a statement runs for at least this long, it is written to the server log, with its duration.

Logging (Cont'd)

- **log_connections** (default off): Log successful connections to the server log.
- **log_disconnections** (default off): Log some information each time a session disconnects, including the duration of the session.
- **log_error_verbosity** (default "default"): Can also select "terse" or "verbose".
- **log_duration** (default off): Log duration of each statement.
- **log_line_prefix**: Additional details to log with each line.
- **log_statement** (default none): Legal values are none, ddl, mod (DDL and all other data-modifying statements), or all.
- **log_temp_files** (default -1): Log temporary files of this size or larger, in kilobytes.
- **log_checkpoints** (default off): Causes checkpoints and restartpoints to be logged in the server log.

Auditing Settings

- **EDB AUDIT Setting:**
  - **edb_audit**
    - Enables or disables database auditing. The values xml or csv will enable database auditing
  - Other Parameters:
    - edb_audit_directory
    - edb_audit_filename
    - edb_audit_rotation_day
    - edb_audit_rotation_size
    - edb_audit_rotation_seconds
    - edb_audit_disconnect
    - edb_audit_connect
    - edb_audit_statement

Background Writer Settings

- There is a separate server process called the background writer, whose function is to issue writes of "dirty" shared buffers.
- Parameters for Background Writer settings:
  - **bgwriter_delay (integer)**
    - Specifies the delay between activity rounds for the background writer. The default value is 200 milliseconds
  - **bgwriter_lru_maxpages (integer)**
    - In each round, no more than this many buffers will be written by the background writer. Setting this to zero disables background writing (except for checkpoint activity). The default value is 100 buffers.

Statement Behavior

- **search_path:** This parameter specifies the order in which schemas are searched. The default value for this parameter is "$user", public
- **default_tablespace:** Name of the tablespace in which to objects are created by default
- **temp_tablespaces:** Tablespaces name(s) in which temporary objects are created
- **statement_timeout:** Postgres will abort any statement that takes over the specified number of milliseconds A value of zero (the default) turns this off.

Vacuum Cost Settings

- **vacuum_cost_delay** (default 0 ms): The length of time, in milliseconds, that the process will wait when the cost limit is exceeded.
- **vacuum_cost_page_hit** (default 1): The estimated cost of vacuuming a buffer found in the EDB Postgres buffer pool.
- **vacuum_cost_page_miss** (default 10): The estimated cost of vacuuming a buffer that must be read into the buffer pool.
- **vacuum_cost_page_dirty** (default 20): The estimated cost charged when vacuum modifies a buffer that was previously clean.
- **vacuum_cost_limit** (default 200): The accumulated cost that will cause the vacuuming process to sleep.

Vacuum Cost Settings (Cont'd)

- **autovacuum** (default on). Controls whether the autovacuum launcher runs, and starts worker processes to vacuum and analyze tables.
- **log_autovacuum_min_duration** (default -1). Autovacuum tasks running longer than this duration are logged.
- **autovacuum_max_workers** (default 3). Maximum number of autovacuum worker processes which may be running at one time.

Edb-dynatune Settings

- DynaTune
  - Dynamic tuning of the database server to make the optimal usage of the system resources
  - Parameters that control this functionality
    - **edb_dynatune:** determines how much of the host system's resources are to be used by the database server (1-100)
    - **edb_dynatune_profile:** used to control tuning aspects based upon the expected workload profile on the database server. Possible values:
      - oltp
      - reporting
      - mixed

Oracle Compatibility Settings

- EDB Postgres Advanced Server supports the development and execution of PostgreSQL and Oracle applications.
- There are a number of system behaviors that can be altered to act in a more PostgreSQL or in a more Oracle compliant manner.
  - oracle_home: must be set for database links to oracle server
  - edb_redwood_date – Controls whether or not a time component is stored in DATE columns. edb_redwood_strings – Equates null to an empty string for purposes of string concatenation operations.
  - edb_stmt_level_tx – Isolates automatic rollback of an aborted SQL command to statement level rollback only
    - E.g: \set AUTOCOMMIT off

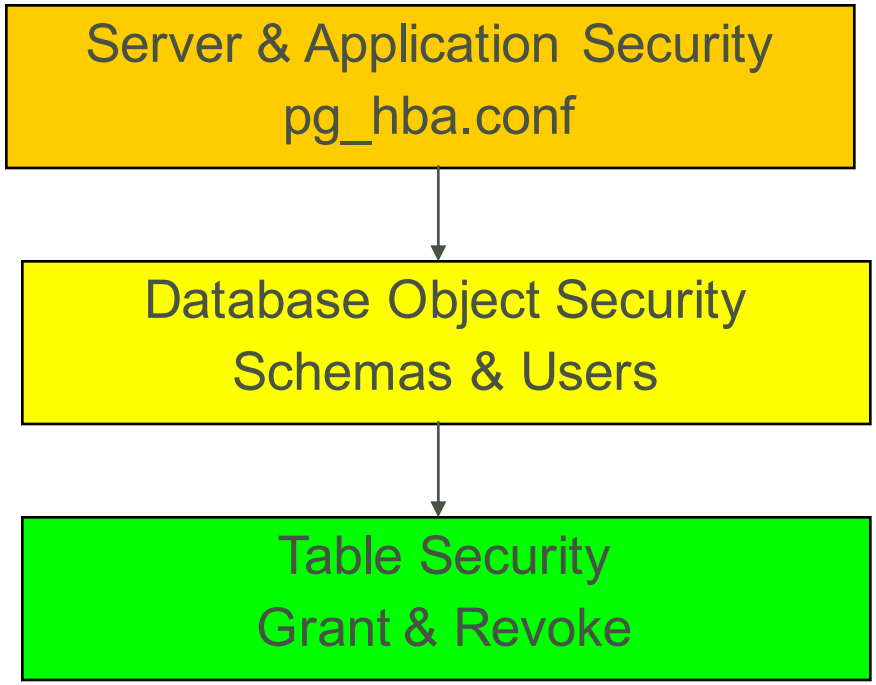
      SET edb_stmt_level_tx TO off;

Security

Authentication and Authorization

- Security is a two step process:
  - Authentication – Ensures a user is who he/she claims to be
  - Authorization - Ensures an authenticated user has access to only the data for which he/she has been granted the appropriate privileges.

Levels of Security



pg_hba.conf – Access control

- Host based access
  - Controls (by ip / subnet) authentication mechanism
  - Adds authorization control by ip / subnet

```
# TYPE   DATABASE     USER         CIDR-ADDRESS            METHOD

# "local" is for Unix domain socket connections only
local    all          all                                  md5

# IPv4 local connections:
host     customer     readonly     62.124.15.9             md5
host     all          all          192.168.12.10/32        md5
host     all          all          127.0.0.1/32            trust

# IPv6 local connections:
host     all          all          ::1/128                 password
```

ROLE, USER, GROUP

- A role can be considered a “user”, a “group”
- The only different is that when the command is spelled CREATE USER, LOGIN is assumed by default
  - CREATE USER A == CREATE ROLE A LOGIN
- CREATE GROUP is now an alias for CREATE ROLE

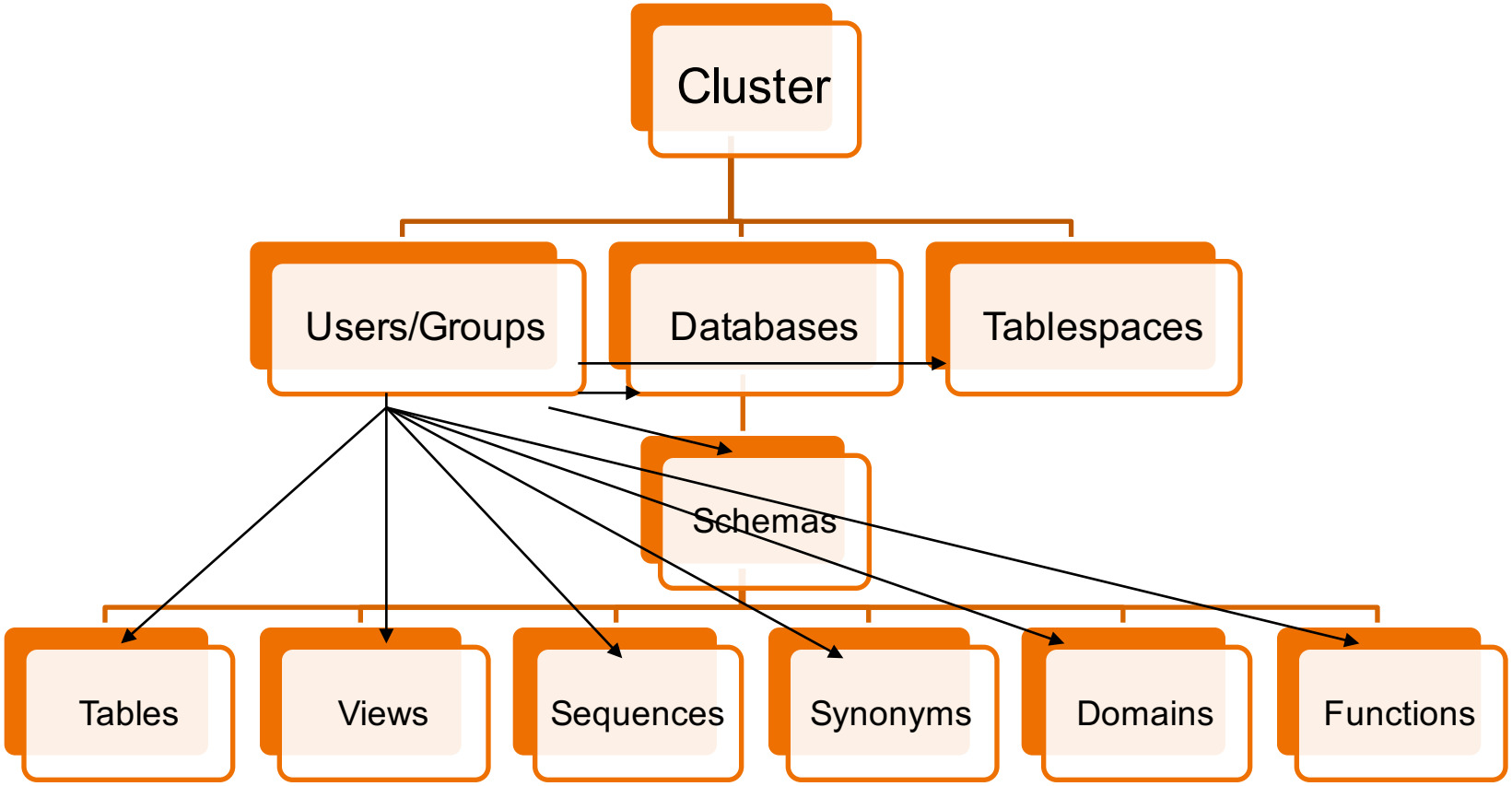
Users

- Database users are different than operating system users
- Users can be created in SQL using “CREATE USER” command or using the “createuser” utility.

SQL Example:

```
CREATE USER scott PASSWORD 'tiger'; --postgres
  compatible syntax
CREATE USER united IDENTIFIED BY air123; --oracle
  compatible syntax
CREATE USER scott_dba CREATEDB CREATEUSER;
CREATE USER scott_temp VALID UNTIL '2006-05-01';
DROP USER scott CASCADE;
```

Object Ownership



Access Controls

- Access to tables is given and taken using the GRANT and REVOKE SQL commands

Examples:

```
GRANT UPDATE DELETE ON emp TO scott_temp;

GRANT ALL ON dept TO GROUP temps;

REVOKE UPDATE DELETE ON emp FROM scott_temp;

GRANT USAGE ON SCHEMA psteinhe TO SCOTT;
```

Access Controls

```
GRANT { { SELECT | INSERT | UPDATE | DELETE | TRUNCATE | REFERENCES | TRIGGER }
    [, ...] | ALL [ PRIVILEGES ] }
    ON { [ TABLE ] table_name [, ...]
         | ALL TABLES IN SCHEMA schema_name [, ...] }
    TO role_specification [, ...] [ WITH GRANT OPTION ]

GRANT { { SELECT | INSERT | UPDATE | REFERENCES } ( column_name [, ...] )
    [, ...] | ALL [ PRIVILEGES ] ( column_name [, ...] ) }
    ON [ TABLE ] table_name [, ...]
    TO role_specification [, ...] [ WITH GRANT OPTION ]

GRANT { { USAGE | SELECT | UPDATE }
    [, ...] | ALL [ PRIVILEGES ] }
    ON { SEQUENCE sequence_name [, ...]
         | ALL SEQUENCES IN SCHEMA schema_name [, ...] }
    TO role_specification [, ...] [ WITH GRANT OPTION ]

GRANT { { CREATE | CONNECT | TEMPORARY | TEMP } [, ...] | ALL [ PRIVILEGES ] }
    ON DATABASE database_name [, ...]
    TO role_specification [, ...] [ WITH GRANT OPTION ]

GRANT { USAGE | ALL [ PRIVILEGES ] }
    ON DOMAIN domain_name [, ...]
    TO role_specification [, ...] [ WITH GRANT OPTION ]

GRANT { USAGE | ALL [ PRIVILEGES ] }
    ON FOREIGN DATA WRAPPER fdw_name [, ...]
    TO role_specification [, ...] [ WITH GRANT OPTION ]
```

```
GRANT { EXECUTE | ALL [ PRIVILEGES ] }
    ON { FUNCTION function_name ( [ [ argmode ] [ arg_name ] arg_type [, ...] ] ) [, ...]
         | ALL FUNCTIONS IN SCHEMA schema_name [, ...] }
    TO role_specification [, ...] [ WITH GRANT OPTION ]

GRANT { USAGE | ALL [ PRIVILEGES ] }
    ON LANGUAGE lang_name [, ...]
    TO role_specification [, ...] [ WITH GRANT OPTION ]

GRANT { { SELECT | UPDATE } [, ...] | ALL [ PRIVILEGES ] }
    ON LARGE OBJECT loid [, ...]
    TO role_specification [, ...] [ WITH GRANT OPTION ]

GRANT { { CREATE | USAGE } [, ...] | ALL [ PRIVILEGES ] }
    ON SCHEMA schema_name [, ...]
    TO role_specification [, ...] [ WITH GRANT OPTION ]

GRANT { CREATE | ALL [ PRIVILEGES ] }
    ON TABLESPACE tablespace_name [, ...]
    TO role_specification [, ...] [ WITH GRANT OPTION ]

GRANT { USAGE | ALL [ PRIVILEGES ] }
    ON TYPE type_name [, ...]
    TO role_specification [, ...] [ WITH GRANT OPTION ]
```

ALTER DEFAULT PRIVILEGES

```
ALTER DEFAULT PRIVILEGES
    [ FOR { ROLE | USER } target_role [, ...] ]
    [ IN SCHEMA schema_name [, ...] ]
    abbreviated_grant_or_revoke
```

```
GRANT { { SELECT | INSERT | UPDATE | DELETE | TRUNCATE | REFERENCES | TRIGGER }
    [, ...] | ALL [ PRIVILEGES ] }
    ON TABLES
    TO { [ GROUP ] role_name | PUBLIC } [, ...] [ WITH GRANT OPTION ]

GRANT { { USAGE | SELECT | UPDATE }
    [, ...] | ALL [ PRIVILEGES ] }
    ON SEQUENCES
    TO { [ GROUP ] role_name | PUBLIC } [, ...] [ WITH GRANT OPTION ]

GRANT { EXECUTE | ALL [ PRIVILEGES ] }
    ON FUNCTIONS
    TO { [ GROUP ] role_name | PUBLIC } [, ...] [ WITH GRANT OPTION ]

GRANT { USAGE | ALL [ PRIVILEGES ] }
    ON TYPES
    TO { [ GROUP ] role_name | PUBLIC } [, ...] [ WITH GRANT OPTION ]
```

```
REVOKE [ GRANT OPTION FOR ]
    { { SELECT | INSERT | UPDATE | DELETE | TRUNCATE | REFERENCES | TRIGGER }
    [, ...] | ALL [ PRIVILEGES ] }
    ON TABLES
    FROM { [ GROUP ] role_name | PUBLIC } [, ...]
    [ CASCADE | RESTRICT ]

REVOKE [ GRANT OPTION FOR ]
    { { USAGE | SELECT | UPDATE }
    [, ...] | ALL [ PRIVILEGES ] }
    ON SEQUENCES
    FROM { [ GROUP ] role_name | PUBLIC } [, ...]
    [ CASCADE | RESTRICT ]
```

Access Controls

- SELECT
  - Allows SELECT from any column, or the specific columns listed, of the specified table, view, or sequence
- INSERT
  - Allows INSERT of a new row into the specified table
- UPDATE
  - Allows UPDATE of any column, or the specific columns listed, of the specified table
- DELETE
  - Allows DELETE of a row from the specified table
- TRUNCATE
  - Allows TRUNCATE on the specified table
- REFERENCES
  - To create a foreign key constraint, it is necessary to have this privilege on both the referencing and referenced columns

Access Controls

- TRIGGER
  - Allows the creation of a trigger on the specified table
- CREATE
  - For databases, allows new schemas to be created within the database
  - For schemas, allows new objects to be created within the schema
  - For tablespaces, allows tables, indexes, and temporary files to be created within the tablespace
- CONNECT
  - Allows the user to connect to the specified database
- TEMPORARY, TEMP
  - Allows temporary tables to be created while using the specified database
- EXECUTE
  - Allows the use of the specified function and the use of any operators that are implemented on top of the function

Access Controls

- USAGE
  - For procedural languages, allows the use of the specified language for the creation of functions in that language
  - For schemas, allows access to objects contained in the specified schema
  - For sequences, this privilege allows the use of the currval and nextval functions
  - For types and domains, this privilege allow the use of the type or domain in the creation of tables, functions, and other schema objects
  - For foreign-data wrappers, this privilege enables the grantee to create new servers using that foreign-data wrapper
  - For servers, this privilege enables the grantee to create foreign tables using the server
- ALL PRIVILEGES
  - Grant all of the available privileges at once

\dp

```
=> \dp mytable
                              Access privileges
 Schema |  Name   | Type  |   Access privileges   | Column access privileges
--------+---------+-------+-----------------------+--------------------------
 public | mytable | table | miriam=arwdDxt/miriam | col1:
                          : =r/miriam             :   miriam_rw=rw/miriam
                          : admin=arw/miriam
(1 row)
```

```
rolename=xxxx -- privileges granted to a role
        =xxxx -- privileges granted to PUBLIC

            r -- SELECT ("read")
            w -- UPDATE ("write")
            a -- INSERT ("append")
            d -- DELETE
            D -- TRUNCATE
            x -- REFERENCES
            t -- TRIGGER
            X -- EXECUTE
            U -- USAGE
            C -- CREATE
            c -- CONNECT
            T -- TEMPORARY
      arwdDxt -- ALL PRIVILEGES (for tables, varies for other objects)
            * -- grant option for preceding privilege

        /yyyy -- role that granted this privilege
```


EDB ENTERPRISEDB

Application Access

- Application access is controlled by settings in both postgresql.conf and pg_hba.conf
- Set the following parameters in postgresql.conf:

  listen_address

  max_connections

  superuser_reserved_connections

  Port