DataBase Administrator PostgreSQL

Ernesto Quiñones A.
ernesto@eqsoft.net

Sesión 1
Instalación y Configuración Básica de PostgreSQL

1: Un poco de historia de PostgreSQL

2: Arquitectura de PostgreSQL

3. Instalando PostgreSQL

1. Un poco de historia de PostgreSQL

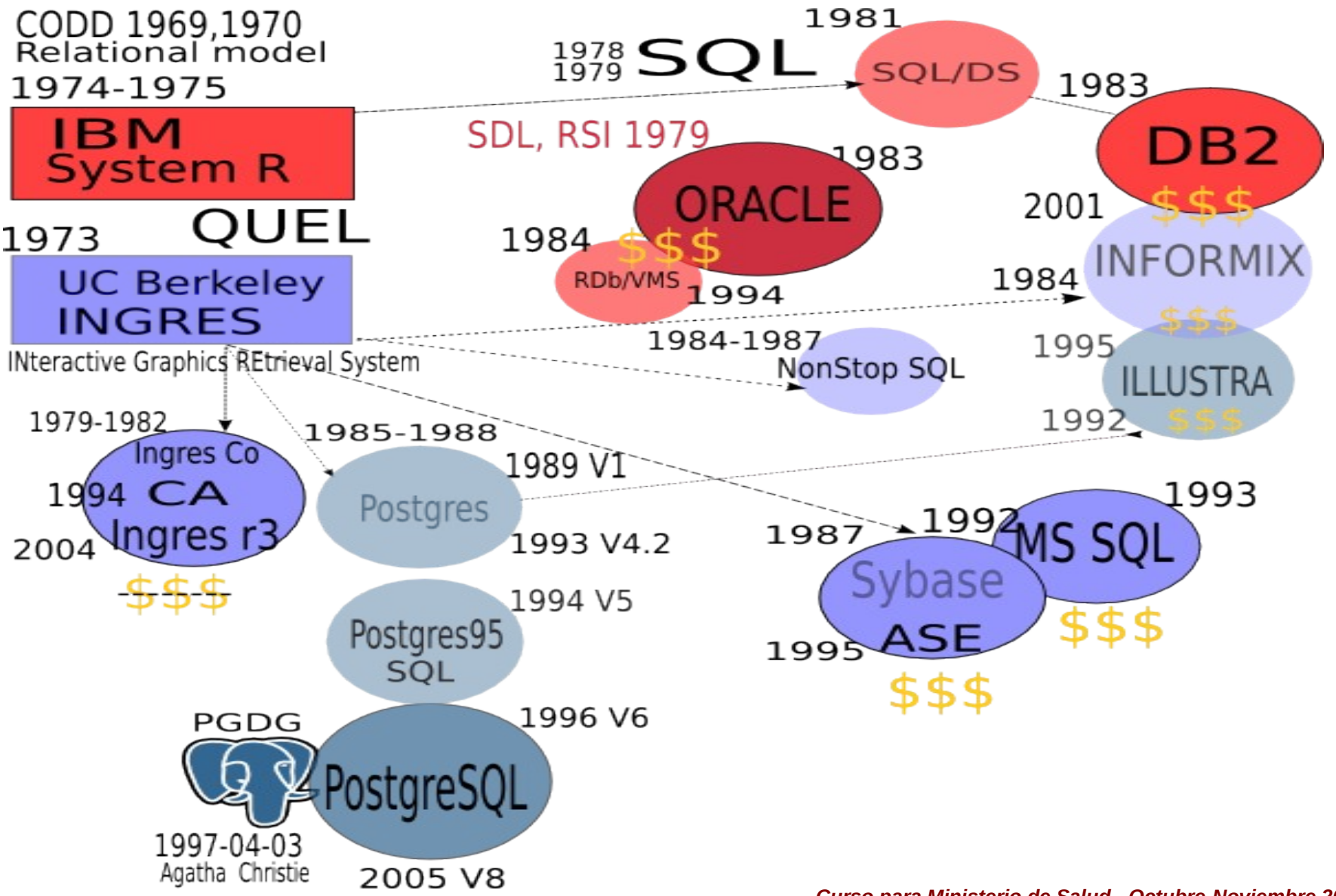
- **PostgreSQL es una base de datos con una trayectoria mucho mas antigua de la que aparenta, pionera en la implementación de varios nuevas características en las RDBMS.**

- **Como proyecto Open Source lleva una trayectoria de más de 20 años, versión actual 10.0 (Oct-2017)**

- **Ofrece soporte a una gran gama de sistemas operativos, como por ejemplo: Linux, UNIX (AIX, sistemas BSD, HP-UX, SGI IRIX, Mac OS X, Solaris, Tru64) y Windows.**

- **Es desarrollada por una comunidad de desarrolladores muy activa (programada en C), no puede ser adquirida por una empresa, usualmente se libera un nuevo release cada año.**

1. Un poco de historia de PostgreSQL



PostgreSQL tiene una arquitectura que involucra muchos estilos, en su nivel mas alto es un esquema clásico cliente-servidor, mientras que el acceso a la data es un esquema en capas.

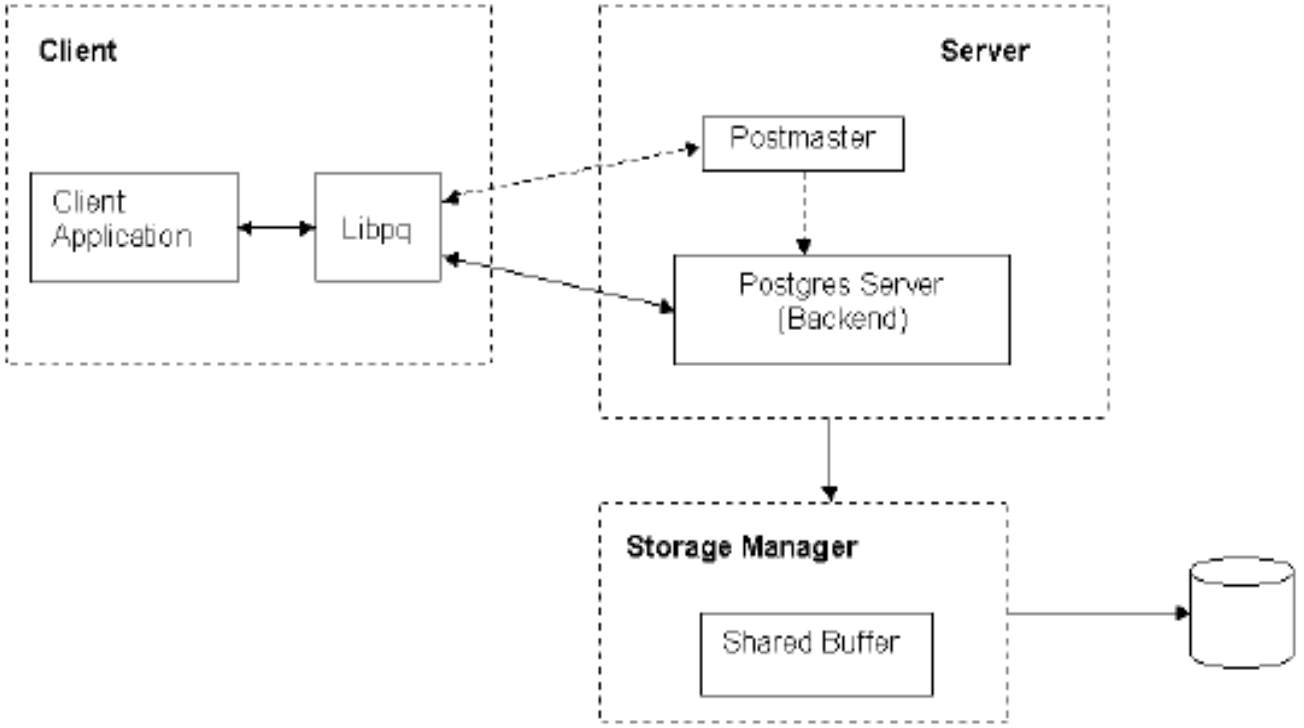


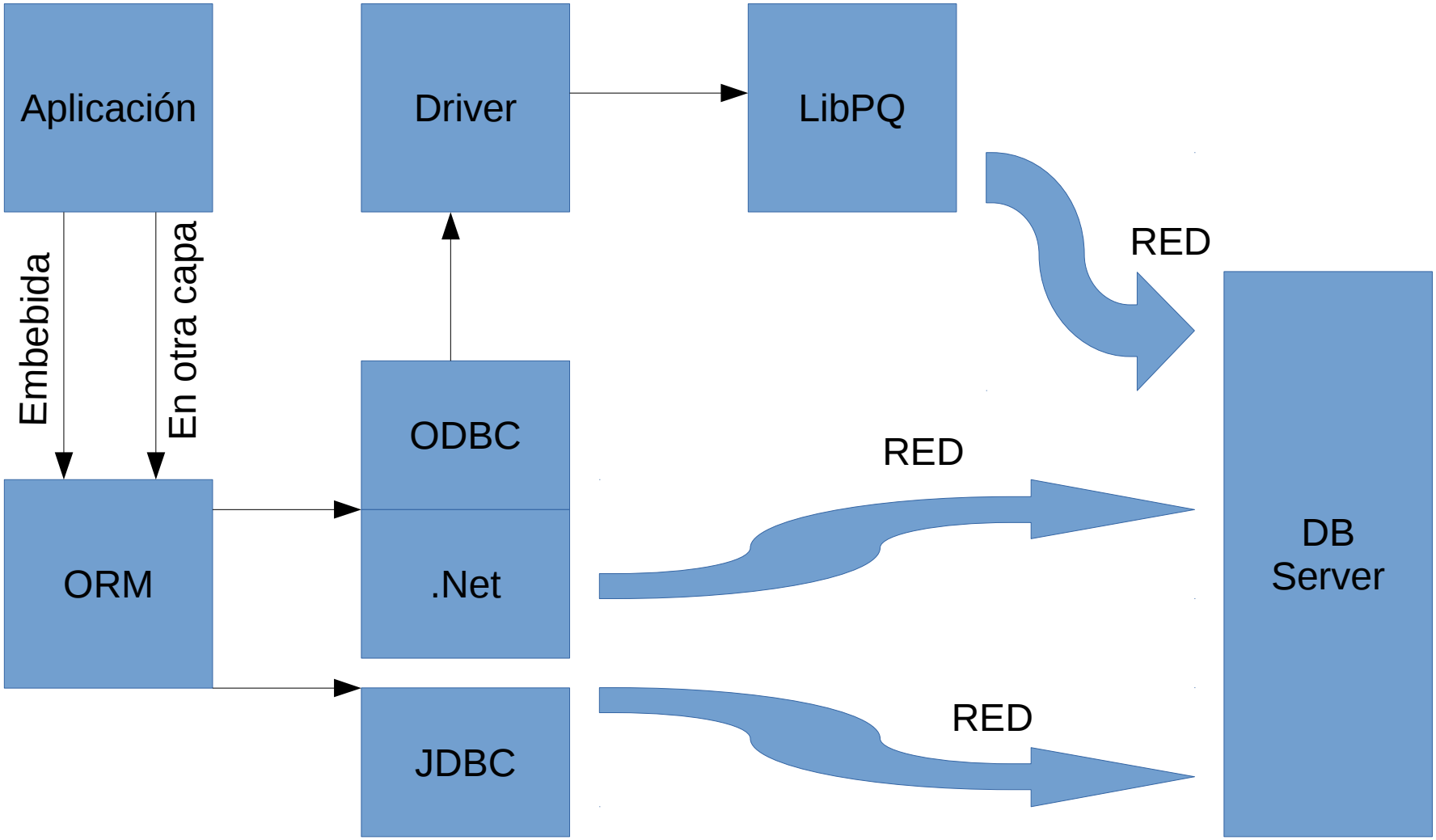
Figure 1 PostgreSQL System Concept Architecture

- El Libpq es el responsable de manipular las comunicaciones entre la aplicación cliente y el postmaster (servicio del PostgreSQL en el servidor).

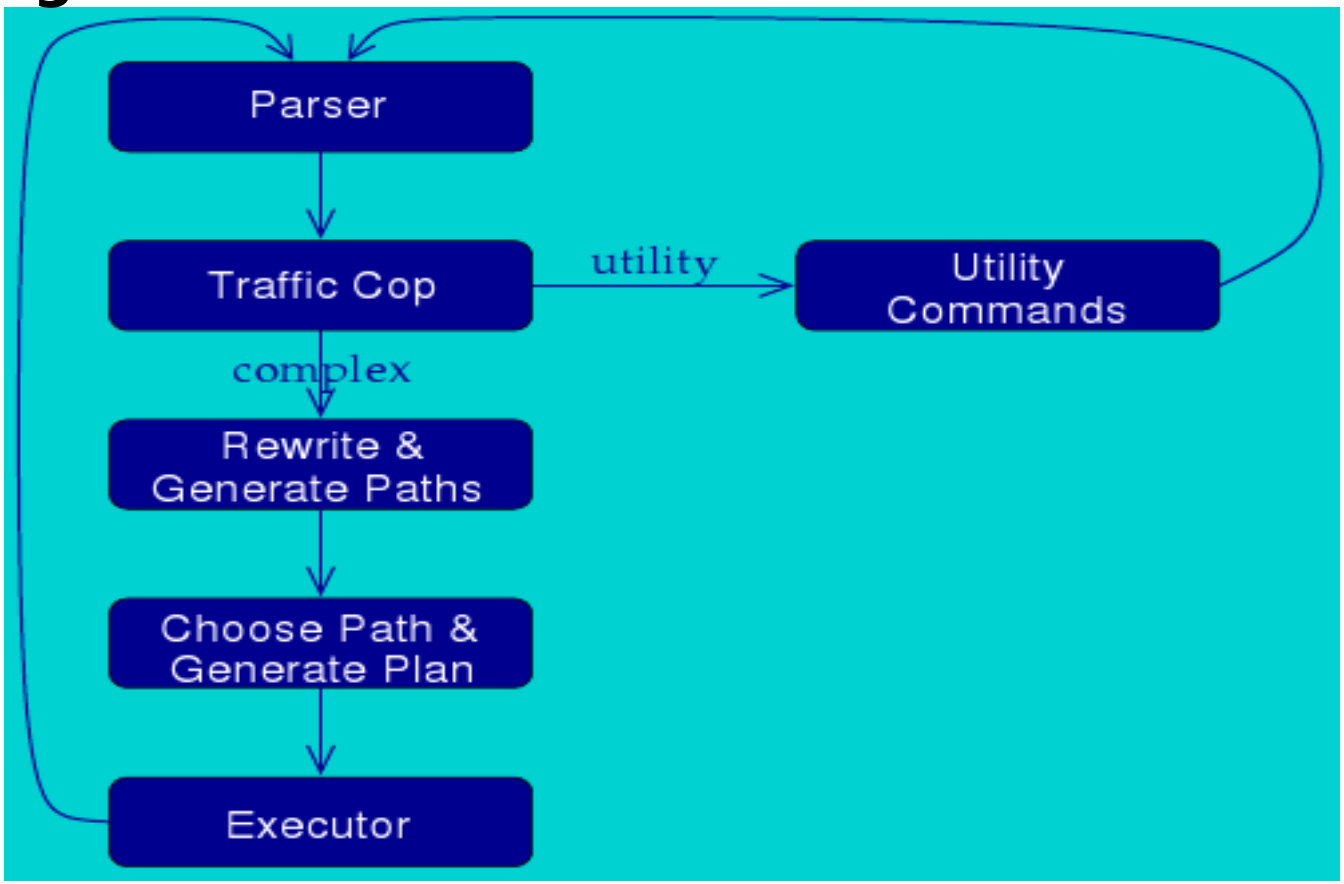
- Se usa el concepto de “Un proceso por cliente”, como no es posible saber cuanto tiempo esta activo este proceso existe un proceso principal llamado “postgres” que escucha los puertos Tcp/Ip para crear los procesos apenas se registren solicitudes de conexión a la base de datos.

- Los procesos se comunican entre si utilizando memoria compartida y semáforos (esto influye en la cantidad de conexiones que soporta un servidor).

- **Que pasa cuando tenemos varias capas en la comunicación:**



El PostgreSQL Server es el cerebro central del motor de la base de datos, es el responsable del procesamiento de los queries, su arquitectura es como sigue:



- **Parser, se encarga de la revisión sintáctica de los queries.**

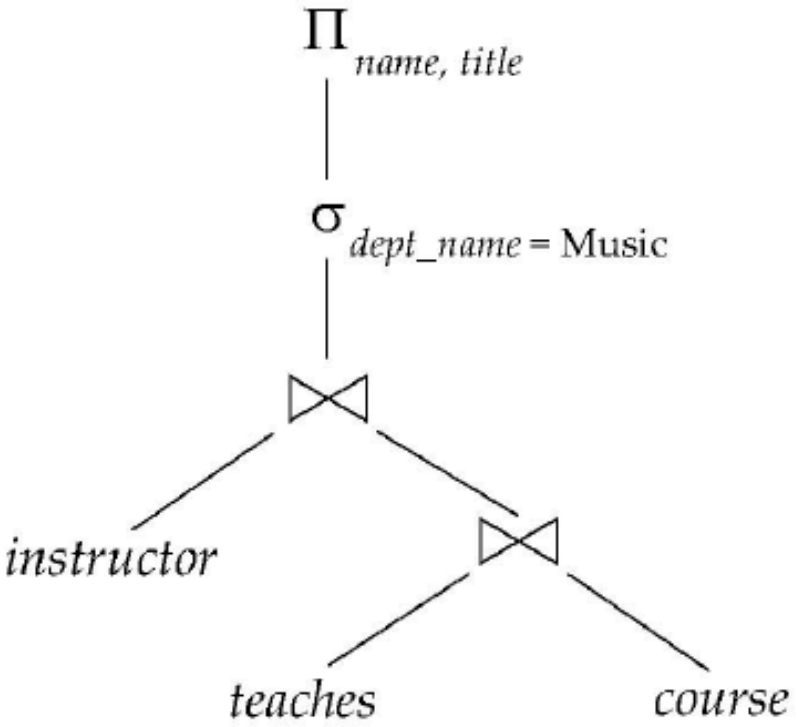
- **Traffic Cop, revisa si lo recibido es un DML para pasarlo al Rewrite o si es un DML para pasarlo al Utility Commands**
  - **El Utility Commands se encarga de procesar comandos que principalmente son utilidades de administración de la db, por ejemplo el Create, Alter o Drop se ejecutan en este componente.**

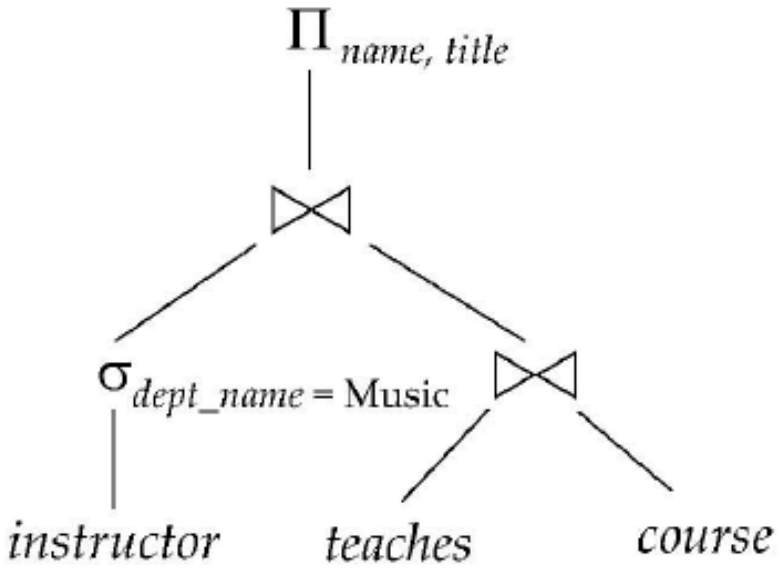
- **El Rewrite se encarga de las reescribir los Queries para transformarlos a un lenguaje que entiende el motor de la base de datos.**

- **El Choose & Generation Path, es el encargado (en base a funciones matemáticas aplicadas a las “estadísticas”) de seleccionar cual es la mejor ruta de extracción de los datos solicitados, se obtienen varias rutas para una misma consulta.**

- **El Executor, es el componente que se comunica con el Storage Manager para la extracción física de los datos.**

- **Esto es un ejemplo de un plan de ejecución.**





El Storage Manager es responsable de la administración general de almacenamiento de los datos, controla todos los trabajos del back-end incluido la administración del buffer, archivos, bloqueos y control de la consistencia de la información.

Los módulos que componen el Storage Manager son:

- **Transaction System**
- **Relational Storage**
- **Time Management**
- **Concurrency Control y Timestamp Management**
- **Record Access**

TOMAR EN CUENTA

PostgreSQL siempre esta añadiendo data, la data modificada o borrada realmente no se modifica o se borra, las páginas donde ellas están almacenadas se marca como “no visible” y se inserta un nuevo registro completo con un clon de toda la data, esto se conoce como WORM (write once read many).

Esto hace que la base de datos ocupe MUCHO ESPACIO y afecta el “tiempo de acceso” a la data.

- **Transaction System, responsable de la administración de las tareas en operaciones indivisibles llamadas transacciones.**

- **Relational Storage, es el encargado de almacenar las “filas de información”, manteniendo la congruencia relacional en los datos.**

- **Time Management, asegura la validez de la data que es devuelta a un usuario en un determinado lapso de tiempo (milisegundos o menos habitualmente), esto debido al tiempo de ejecución de la operación realizada y la concurrencia efectuada sobre esta.**

- **Concurrency Control y Timestamp Management, mantiene el control de los tiempos mínimo y máximo de “commiteo” de las transacciones. Para ello PostgreSQL maneja una tabla principal de bloqueos de memoria.**

- **Record Access, es el responsable de acceder a los datos físicamente, para ello utiliza punteros de acceso a las páginas de datos que se almacenan dentro de ellos mismos.**

3. Instalando PostgreSQL

Los sistemas operativos más usuales para instalar PostgreSQL son Gnu/Linux y Unix, en menor medida Windows.

2 formas de instalación son las más comunes:

- **Instalación vía paquetes instaladores (Principalmente en Gnu/Linux, deb ó rpm)**
  - **Ventajas: mantenimiento prácticamente 0, excepto en cambios de versión superior.**
  - **Desventajas: se trabaja con ramas congeladas o sin actualizar por mucho tiempo (especialmente en “Linux Enterprises”), básicamente solo se dan actualizaciones de seguridad.**
    - **Para superar esta desventaja se recurre a instaladores no oficiales.**

3. Instalando PostgreSQL

- **Compilación desde el código fuente (Principalmente en Unix)**
  - **Ventajas: Rápida ejecución, menor consumo de recursos, uso de últimas versiones disponibles, posibilidad de añadir/activar nuevas funcionalidades extendidas.**
  - **Desventajas: Mantenimiento manual, no es sencillo actualizar (salvo en ports/portage)**
    - **Totalmente desaconcejado en distribuciones "Enterprise".**

3. Instalando PostgreSQL

Requerimientos de hardware:

- **Procesador : Un punto importante aquí es que PostgreSQL usa un core por conexión a la base de datos (esto incluyen procesos como Vacuum), una regla no escrita es: “PostgreSQL requiere tantos cores como conexiones concurrentes quiera soportar entre cuatro”.**
  - **Alta concurrencia de usuarios transaccionales simples, procesadores de muchos cores de regular potencia sería recomendable.**
  - **Baja concurrencia de procesos pesados, como un BI, procesadores de regular cantidad de cores pero de mucha potencia.**
  - **Alta concurrencia de procesos pesados, procesadores de alto rendimiento por core y muchos cores.**

3. Instalando PostgreSQL

- **Memoria : mientras más ram se pueda tener es mejor, especialmente si se van a ejecutar Querys que van a mover mucha data, mientras mayor capacidad tenga el cache mejor será el rendimiento de las consultas, pero debe afinarse la configuración del servidor.**

  - **OjO: La ram debe ser acorde a la capacidad de disco implementada, discos duros más grandes requerirán más ram para mapear todas las direcciones del filesystem.**

3. Instalando PostgreSQL

Requerimientos de hardware:

- **Disco : la cantidad de espacio dependerá del volumen de datos, recomendaciones:**
  - **SAS/SCSI, bases de datos relativamente medianas donde se requiere un alto acceso a los datos, entorno muy propenso a caídas de servidor.**
  - **SATA, bases de datos muy grandes pero con no muy altos requerimientos de acceso a los mismos.**
  - **SSD, para cache de la DB, no usar para almacenamiento de datos “real” debido a su “fragilidad” en entornos propensos a caídas.**

- **SAN/NAS, permite una gran cantidad de almacenamiento de datos y asegura la disponibilidad de accesos a los datos (gracias a la gran cantidad de discos en la que redunda la misma), pero se debe tomar en cuenta lo siguiente:**
  - **Existe un problema de latencia por el canal de comunicación.**
  - **Muchos discos redundando datos añaden tiempo a los procesos transaccionales a menos que se invierta en “cachés” amplios.**

3. Instalando PostgreSQL

Configuración:

- **Para instalar:**

```
usuario@server:  yum install postgresql
```

Se debe ejecutar como Administrador del Sistema. Instalará la última versión disponible en el repositorio oficial.

- **El sistema configurará el servidor con valores por defecto, probablemente lo más importante es el "locale" del servidor.**

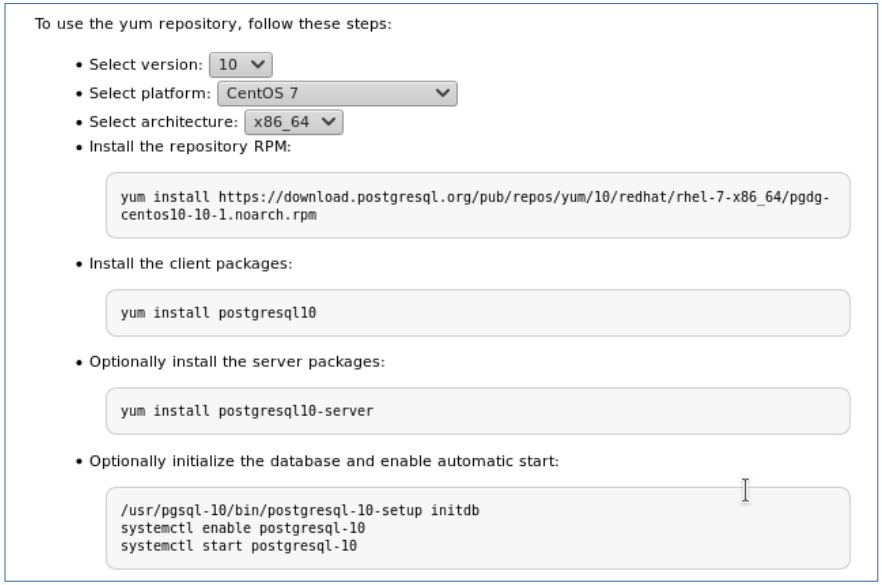
- **El "locale" es la configuración regional de nuestro servidor, entre los valores más importantes van el mapa de caracteres, la moneda, etc., evite crear diversas configuraciones de "locales" en su servidor si no es necesario.**

3. Instalando PostgreSQL

Configuración:

Consideraciones para CentOS/RedHat

- Las versiones estables disponibles son bastante antiguas, por ejem. En CentOS 7.4 se distribuye PostgreSQL 9.2
- Para descargar versiones más recientes dirijase a esta dirección: https://www.postgresql.org/download/linux/redhat/



3. Instalando PostgreSQL

Configuración:
La mayoría de las distribuciones actuales de GNU/Linux auto-generan los directorios necesarios para correr el cluster de la DB.

En Centos se creará /var/lib/pgsql/10

- **Se crearán 2 directorios principales:**

  /var/lib/pgsql/10/data/ <-- donde 10 es la versión utilizada.

  Este directorio almacena los datos y la configuración del sistema.

- **Otros directorios importantes:**
  /usr/pgsql-10 <-- donde encontrarán librerías y binarios.

- **El usuario administrador más común creado por los instaladores es “postgres”.**

Configuración:

- Dentro de **/var/lib/pgsql/10/data/** encontrará los directorios de trabajo del cluster de la base de datos:
  - **base** <-- directorio de data, “tablespace” por defecto.
  - **global** <-- directorio de data global del sistema, por ejemplo: los roles, los catálogos de objetos de la db, etc.
  - **log** <-- log de ocurrencias del dbms.
  - **pg_commit_ts** <-- historial “temporal” de commits en los datos.
  - **pg_dynshmem** <-- archivos de control de la asignación dinámica de memoria compartida
  - **pg_logical** <-- data de la persistencia de los datos, para no tener que bajar al estado físico de los mismos.
  - **pg_multixact** <-- data producida por “locks” del sistema.
  - **pg_notify** <-- almacena los datos de la funcionalidad de LISTEN/NOTIFY en las transacciones.
  - **pg_repslot** <-- contiene vistas de los datos en replicación.
  - **pg_serial** <-- contiene datos de transacciones commiteadas serializadas (finalizadas).
  - **pg_snapshots** <-- almacena los datos de transacciones que pueden ser usadas por otras transacciones.
  - **pg_stat** <-- almacena una copia de las estadísticas cuando se baja el servidor por el proceso convencional.
  - **pg_stat_tmp** <-- archivos temporales de las estadísticas de las dbs

3. Instalando PostgreSQL

Configuración:

- **pg_subtrans** <-- usado en control de sub-transacciones
- **pg_tblspc** <-- enlaces simbólicos a los tablespaces
- **pg_twophase** <-- usado en control de commits de “dos fases” (gestiona los commit de las transacciones que afectan al mismo tiempo la misma data)
- **pg_wal** <-- directorio de WAL (historia de movimientos de data, es la que asegura que la dbms sea ACID) (pg_xlog)
- **pg_xact** <-- directorio de “commit” de las transacciones, de este directorio es que se lee la data (pg_clog).

Dentro de este directorio además encontrará 3 archivos más:

- **postmaster.opts** <-- comando con el que se ejecuto el servicio de la dbms
- **postmaster.pid** <-- “pid” del proceso del sistema, en raras ocasiones la dbms se bloquea al reiniciar el servicio porque no se elimina este archivo, se puede borrar manualmente pero jamas mientras el servicio este iniciado.
- **PG_VERSION** <-- versión de PostgreSQL que estamos usando.

3. Instalando PostgreSQL

En distribuciones de Linux el directorio esta en /etc

Configuración:

- Dentro de **/var/lib/pgsql/10/data/** encontrará los siguientes archivos de configuración:
  - **pg_hba.conf** <-- archivo de control de accesos al sistema.
  - **pg_ident.conf** <-- configuración de usuarios usando método “ident” en pg_hba.conf desde IPs remotas.
  - **postgresql.conf** <-- parámetros de configuración del sistema

En Ubuntu Server encontrará algunos archivos adicionales como:

  - **environment** <-- variables de entorno para el proceso “postmaster”,
  - **pg_ctl.conf** <-- parámetros del cluster ha ser pasados por el binario pg_ctl (no usable en Ubuntu)
  - **start.conf** <-- permite configurar si el servicio se levanta automáticamente al iniciar el server.

Configuración

Por defecto PostgreSQL viene configurado para dar accesos en "localhost", los usuarios comunes de administración son "postgres" (Gnu/Linux) y "pgsql" (Unix).

Vamos a crear un nuevo "Super Usuario" desde consola, como usuario "root":

```
root@host: su – postgres
postgres@host: createuser -s admindb <-- "-s" indica que será un super
                                         usuario, y solo lo puede
                                         hacer otro super usuario.
                                         "admindb" es el usuario a
                                         crear.

postgres@host: psql template1        <-- ingresamos para cambiar el
                                         password de "admindb"
                                         apesol

template1=# alter user admindb  password 'dbadmin'; <-- asignamos el
template1=# alter user postgres password 'dbpgsql';     nuevo
                                                        Password.
```

Configuración

Ahora necesitamos que no cualquier persona con acceso físico al "servidor" pueda ingresar a la base de datos, para ello editamos el archivo "pg_hba.conf" (como usuario "postgres")

```
postgres@host: nano pg_hba.conf
```

DICE:

```
# TYPE DATABASE USER    CIDR-ADDRESS     METHOD
# "local" is for Unix domain socket connections only
local   all         all                              peer
# IPv4 local connections:
host    all         all         127.0.0.1/32         md5
# IPv6 local connections:
host    all         all         ::1/128              md5
```

DEBE DECIR:

```
# TYPE DATABASE USER    CIDR-ADDRESS     METHOD
# "local" is for Unix domain socket connections only
local   all         all                          password
# IPv4 local connections:
host    all         all         127.0.0.1/32  md5
# IPv6 local connections:
#host    all         all         ::1/128      md5 <-- activar solo si
                                                      usamos protocolo IPv6
```

Configuración

Reiniciamos el sistema, para ello tenemos estas alternativas:

postgres@host : /usr/pgsql-10/bin/pg_ctl restart <-- como usuario postgres

root@host : service postgresql-10 restart <-- eso es lo mismo como "root"

Porque hicimos esto:

- **Hemos creado un password para el usuario administrador superior "postgres".**
- **Trabajaremos con un usuario alterno para mayor seguridad.**
- **Evitamos que cualquier persona con acceso al servidor (físico o remoto) entre a la base de datos sin conocer los passwords.**

Debilidad:
Si algún usuario tiene permiso de editar el archivo pg_hba.conf y reiniciar servicios entonces la seguridad es vulnerada.