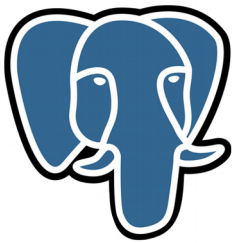
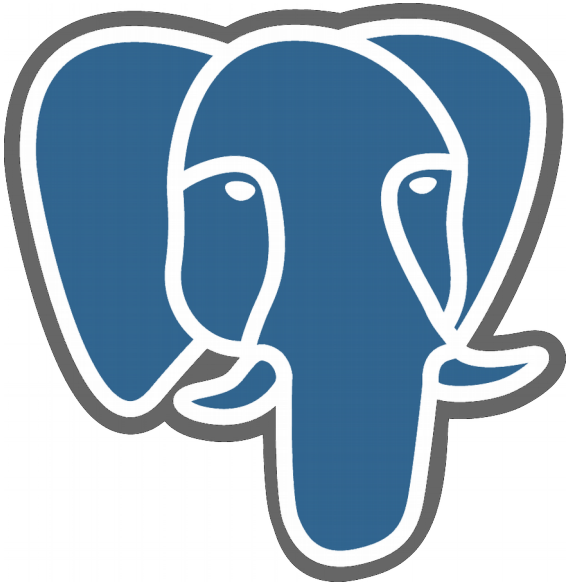


TodoPostgreSQL

Replicación en PostgreSQL

INSTRUCTOR:

José Segovia <info@todopostgresql.com>



Conceptos de replicación

- La replicación es la **transmisión de información** derivada de las operaciones DML, de una BD a otra.

- Todas las operaciones que modifiquen el estado de la BD se transmiten (transformadas o no) a otra BD que "replica" las operaciones, de forma que **ambas BD tengan la misma información.**

Conceptos de replicación

- La replicación permite alcanzar objetivos como:
  - **Alta disponibilidad** (caída del maestro).
  - **Backups "caliente"** (backup con poca o cero recuperación necesaria).
  - Disponer de una **copia en otro lugar geografico**.

Tipos de replicación

Basada en Triggers

- Cada operación DML ejecuta un **trigger** (acción en respuesta a un evento de la BD) que genera una representación del cambio (SQL, JSON, formato binario, etc) y la envía a la BD remota.

- Utiliza una cola para almacenar los cambios, a un ritmo potencialmente diferente del de envío a la BD remota (modelo productor-consumidor): **asincronía.**

Basada en Triggers

- Dado que los triggers se instalan por el software en las tablas, se puede seleccionar un **subconjunto arbitrario de la(s) tabla(s) de la base(s) de datos** que se quieren replicar.

- Slony (Maestro – Esclavo)
- Bucardo (Multi Maestro)

Basada en WAL

- PostgreSQL dispone de un formato **nativo** de representación de los cambios en la BD: los WAL (Write-Ahead Log).

- No supone más overhead, ya que se generan para garantizar la **durabilidad** de la BD.

Basada en WAL

- La implementación es muy **sencilla**, puesto que PostgreSQL sabe cómo interpretar estos registros (proceso de recuperación, wal writer, checkpoint).

- Los registros WAL pueden enviarse a la réplica por ficheros (WAL shipping) o por la red (streaming)

Estudio de los distintos tipos de replicación en cada caso

Trigger

- **Impacto en el maestro (overhead):**
  - Alto (10% – 15%).
- **Latencia de la replicación:**
  - Baja / media / alta.
- **Posibilidad de seleccionar subconjunto de bases de datos y/o tablas (secuencias) a replicar:**
  - Sí.

Trigger

- **Adecuación a entornos MAN / alta latencia / canales ruidosos:**
  - ✔ Funciona mucho mejor.
- **Integración en el core de PostgreSQL:**
  - ✗ No.

WAL

- **Impacto en el maestro (overhead):**
  - ✔ Bajo (1% – 3%).
- **Latencia de la replicación:**
  - ✔ Baja / muy baja.
- **Posibilidad de seleccionar subconjunto de bases de datos y/o tablas (secuencias) a replicar:**
  - ✗ No.

WAL

- **Adecuación a entornos MAN / alta latencia / canales ruidosos:**
  - ✗ Canales TCP/IP permanentemente abiertos y es muy sensible.
- **Integración en el core de PostgreSQL:**
  - ✓ Sí.

WAL Shipping

- Desde PostgreSQL 8.2, soporta **WAL Shipping**, una técnica para enviar registros de WAL de un servidor maestro a otro(s) esclavo(s).
  - La replica está continuamente en modo recuperación.

WAL Shipping

- Desde PostgreSQL 8.2, soporta **WAL Shipping**,
  - Para ello hay que activar el **archivado continuo**.

[postgresql.conf]

```
wal_level = archive
archive_mode = on
archive_command = 'test ! -f /path/dir/archivado/%f && cp %p /path/dir/archivado/%f'
```

WAL Shipping

- Tiene dos inconvenientes:
  - Hasta que un **WAL no rota, no se copiará y procesará al esclavo**, por lo que hay una **ventana de pérdida de datos**.
  - Requiere un **proceso externo** para la copia de los ficheros (síncrono o asíncrono).

- El **tiempo de pérdida de datos** es la suma de el tiempo de **rotación del fichero WAL** y el tiempo de **transmisión / copia** del fichero rotado y archivado a la base de datos esclava.

Streaming Replication

- Para resolver los inconvenientes de WAL Shipping, en PostgreSQL 9.0 se introdujo **Streaming Replication (SR)**, que copia los registros de WAL a las BD esclavas mediante la red (streaming) **según se produzcan**, no cuando rote el fichero.

- Los esclavos se comportan como una conexión más a la base de datos para obtener los WAL, el **overhead es muy bajo**.

Streaming Replication

- La latencia de replicación (determina la máxima pérdida de datos) es **muy baja** (inferior al segundo).

- Por defecto es **Asíncrono** (el COMMIT en el maestro no espera a que los esclavos hayan recibido los registros WAL).

Streaming Replication

- Desde la 9.1 se soporta también el modo **Síncrono** (el COMMIT en el maestro sólo se produce cuando todos los esclavos han recibido los registros WAL). **No hay nunca pérdida de datos**.

- Desde la 9.3 se **soporta replicación en cascada**, que permite que un esclavo pueda enviar los registro WAL como si fuese un maestro.

Streaming Replication

- Con el **parámetro synchronous_commit**, se configura que la replicación sea **síncrona (on)** o **asíncrona (off o local)**. Este parámetro se puede cambiar por cada tx:

```
SET synchronous_commit TO local;
```

Streaming Replication

- Si la sincronización maestro / esclavo(s) se desfasa mucho, puede suceder que el **esclavo se "desconecte"** (si los segmentos de WAL que se deben enviar por la red al esclavo, éste no los ha consumido, y en el maestro se reciclan, ya no se le podrán enviar). En este caso es necesario **comenzar de nuevo** (backup base).

Streaming Replication

- Ya que **SR es compatible con WAL Shipping** (configuración recomendada), se **resuelve el inconveniente de comenzar de nuevo** en el caso de que se reciclen los segmentos en el maestro o si la red cae. Postgres puede aplicar los registros de WAL independientemente de donde vengan, **permite seguir aplicando los ficheros archivados sin que se desconecte el esclavo**.

Hot Standby

Hot Standby

- Es el término para describir la **capacidad de conectarse al servidor y ejecutar consultas de sólo lectura** mientras el servidor está en recuperación de archivos o en el modo de espera. Esto es útil para fines de replicación y para la restauración de una copia de seguridad a un estado deseado con gran precisión.

Hot Standby

- Se puede hacer escalabilidad horizontal de las consultas de lectura en un esquema de replicación.
- Permite tener esclavos retrasados para solucionar fallos humanos. (SR en 9.4)
- No es trivial, por lo tanto, ¿qué sucede si mientras una query larga que se ejecuta en un esclavo, llega un DML de DROP TABLE? Se puede retrasar la aplicación de los registros WAL o cancelar la consulta.

Hot Standby

- A partid de PostgresSQL 8.4, se permite que una BD postgres en modo recuperación acepte consultas de **sólo lectura** durante dicho proceso de recuperación.

  - Puede ser utilizado con WAL shipping o con streaming replication.

Hot Standby

- Configuración

  [postgresql.conf maestro]

  wal_level = **hot standby**

  [postgresql.conf esclavo]

  hot_standby = **on**

  max_standby_archive_delay = **30s**

Configuración
WAL Shipping + Hot Standby

Configuración WAL Shipping

- El primer paso es **activar archivado continuo**:

[postgresql.conf maestro]

```
wal_level = replica
archive_mode = on
archive_command = 'test ! -f /path/archivado/%f
  && cp %p /path/archivado/%f'
```

[requiere reiniciar el cluster]

Configuración WAL Shipping

- El directorio debe estar compartido en local o por NFS para que los ficheros estén disponible en la máquina del esclavo.

Configuración WAL Shipping

- Realizar un **backup base** del maestro al esclavo:

```
SELECT pg_start_backup('backup');

rsync -a –exclude backup_label –exclude
  postmaster.pid –exclude postmaster.opts
  /path/master/ /path/slave/

SELECT pg_stop_backup();
```

Configuración WAL Shipping

- Crear **recovery.conf** en el esclavo:

```
[recovery.conf]
restore_command = 'cp /path/archivado/%f %p'
standby_mode = on
```

Configuración Hot Standby

- Es posible **configurar hot standby** para que el esclavo acepte consultas de sólo lectura, ya sea con WAL shipping o con Streaming replication.

[postgresql.conf esclavo]

```
hot_standby = on
max_standby_archive_delay = 30s
```

- El parámetro wal_level es ignorado en el esclavo (no genera WALs)

WAL Shipping + Hot Standby

- **Iniciar el esclavo**. Comienza primero a recuperar la base de datos a partid de backup, y posteriormente entra en un bucle continuo de esperar a que aparezcan nuevos ficheros WAL y aplicarlos.

Configuración WAL Shipping

```
postgresql96@ubuntu:~$ pg_ctl -D /curso/postgresql/slave/ start
server starting
postgresql96@ubuntu:~$ LOG:  database system was interrupted; last known up at 2017-07-18 13:23:53 CEST
LOG:  entering standby mode
LOG:  database system was not properly shut down; automatic recovery in progress
LOG:  redo starts at 0/5000108
LOG:  invalid record length at 0/50001E8: wanted 24, got 0
LOG:  consistent recovery state reached at 0/50001E8
LOG:  database system is ready to accept read only connections
LOG:  restored log file "000000010000000000000005" from archive
cp: cannot stat '/tmp/archivado/000000010000000000000006': No such file or directory
LOG:  unexpected pageaddr 0/4000000 in log segment 000000010000000000000006, offset 0
cp: cannot stat '/tmp/archivado/000000010000000000000006': No such file or directory
cp: cannot stat '/tmp/archivado/000000010000000000000006': No such file or directory
cp: cannot stat '/tmp/archivado/000000010000000000000006': No such file or directory
```

```
cp: cannot stat '/tmp/archivado/000000010000000000000007': No such file or directory
cp: cannot stat '/tmp/archivado/000000010000000000000007': No such file or directory
LOG:  restored log file "000000010000000000000007" from archive
cp: cannot stat '/tmp/archivado/000000010000000000000008': No such file or directory
LOG:  WAL file is from different database system: WAL file database system identifier is 6442206381223572402, pg_control database system
identifier is 6442206326001623973
cp: cannot stat '/tmp/archivado/000000010000000000000008': No such file or directory
cp: cannot stat '/tmp/archivado/000000010000000000000008': No such file or directory
```

```
cp: cannot stat '/tmp/archivado/000000010000000000000008': No such file or directory
LOG:  restored log file "000000010000000000000008" from archive
LOG:  restored log file "000000010000000000000009" from archive
LOG:  restored log file "00000001000000000000000A" from archive
cp: cannot stat '/tmp/archivado/00000001000000000000000B': No such file or directory
```

Configuración
SR + Hot Standby

Configuración SR asíncrono

- Opcionalmente, configurar **archivado continuo**

- Configuración del maestro

[postgresql.conf maestro]

```
max_wal_senders = x          #número max esclavo
wal_keep_segments = y        #número de segmentos
                             WAL a conservar
```

[requiere reiniciar el cluster]

Configuración SR asíncrono

- Realizar un **backup base** del maestro al esclavo.

- Configuración del esclavo

[postgresql.conf esclavo]

```
hot_standby = on
max_standby_streaming_delay = 30s
hot_standby_feedback = on      #previene conflictos
```

Configuración SR asíncrono

- La replicación se realiza mediante conexiones a la BBDD de los esclavos al maestro que requieren permisos especiales.

- Para ello hace falta una entrada específica en **pg_hba.conf master**, con conexiones a la base de datos llamada **replication**

[pg_hba.conf maestro]

```
host    replication    replicador    127.0.0.1/32    md5
```

[requiere reload]

Configuración SR asíncrono

- Crear el **usuario con privilegios de replication**

```
CREATE USER replicador WITH replication
  PASSWORD 'replicador';
```

Configuración SR asíncrono

- Crar el **archivo recovery.conf** en el esclavo.

[recovery.conf esclavo]

```
primary_conninfo = 'host=127.0.0.1 port=5432
  user=replicador password=replicador'
standby_mode = on
```

Configuración SR asíncrono

- Iniciar el servidor esclavo.

- Para consultar cuál es el último registro de WAL creado, enviado, recibido, se puede consultar mediante funciones o desde una vista (9.2).

Configuración SR asíncrono

Mediante funciones

[BD maestro]

```
SELECT pg_current_xlog_location();
```

[BD esclavo]

```
SELECT pg_last_xlog_receive_location();
SELECT pg_last_xlog_replay_location();
```

Configuración SR asíncrono

A partid de **PostgreSQL 9.2 view**

[BD maestro]

```
SELECT * FROM pg_stat_replication;
```

```
postgres=# SELECT * FROM pg_stat_replication;
 pid  | usesysid |  usename   | application_name | client_addr
 | client_hostname | client_port |         backend_start
   | backend_xmin |
   state   | sent_location | write_location | flush_location |
 replay_location | sync_priority | sync_state
------+----------+------------+------------------+------------
-+-----------------+-------------+--------------------------
---+--------------+
----------+--------------+----------------+----------------+
-----------------+---------------+-----------
 6762 |    16427 | replicador | walreceiver      |
 |                 |          -1 | 2017-07-24 10:58:36.859146+
02 |           606 |
 streaming | 0/60B9198     | 0/60B9198      | 0/60B9198      |
 0/60B9198       |             0 | async
(1 row)
```

Configuración SR síncrono

- Hay que pasar **explícitamente los nombres de las aplicaciones** que se conectarán como esclavos al maestro en replicación síncrona. El "nombre de aplicación" es un parámetro estándar del conninfo.

Configuración SR síncrono

[postgresql.conf maestro]

```
synchronous_standby_names = 'esclavo1,esclavo2'

synchronous_standby_names = '*'


synchronous_commit = on
```

[requiere reload]

Configuración SR síncrono

- Hay que **modificar el archivo recovery.conf**

[recovey.conf esclavo]

```
primary_conninfo = 'host=127.0.0.1 port=5432
  user=replicador password=replicador
  application_name=esclavo1'
standby_mode = on
```

Configuración SR síncrono

- Si un esclavo no está disponible, el maestro se quedará esperando hasta que no conteste (se deberán observar timeouts de queries a nivel de aplicación).

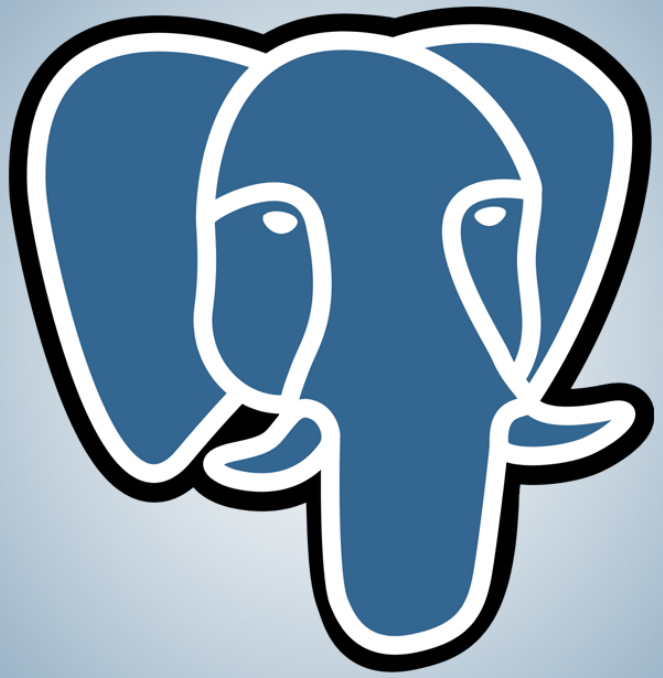
Configuración SR síncrono

- Comprobamos que ahora la replicación sea síncrona y cuál es el último registro WAL

[BD maestro]

```
SELECT * FROM pg_stat_replication;
```

```
postgres=# SELECT * FROM pg_stat_replication;
 pid  | usesysid |  usename   | application_name | client_addr
 | client_hostname | client_port |         backend_start
   | backend_xmin |   state   | sent_location | write_location
 | flush_location | replay_location | sync_priority | sync_sta
te
------+----------+------------+------------------+------------
-+-----------------+-------------+-------------------------------
---+--------------+-----------+---------------+---------------
-+-----------------+-----------------+---------------+---------
---
 7682 |    16427 | replicador | esclavo1         |
 |                 |          -1 | 2017-07-24 13:31:23.556179+
02 |           606 | streaming | 0/70000D0     | 0/70000D0
 | 0/70000D0       | 0/70000D0       |               1 | sync
(1 row)
```



todopostgresql.com