**Trabalho 5 de Banco de dados 2**

**Alunos: Guido Margonar Moreira.**

**Caio Vinicius Oliveira Brito.**

**Guilherme Yamamoto Kato.**

* **O que é replicação PostgreSQL?**
  + A replicação do PostgreSQL envolve a cópia de dados de um servidor primário para um servidor de réplica. O modelo de replicação do PostgreSQL é simples, onde todas as escritas são feitas no node primário e as mudanças são transmitidas para o node secundário para serem aplicadas.
* **Benefícios de usar a replicação PostgreSQL**
  + Migração de dados: A replicação do PostgreSQL facilita a migração eficiente de dados, seja para atualizar o hardware do servidor ou implantar um novo sistema, garantindo a consistência dos dados durante o processo.
  + Tolerância a falhas: A replicação do PostgreSQL oferece uma solução robusta para lidar com falhas. Em caso de falha do servidor primário, o servidor standby assume o papel de principal, garantindo alta disponibilidade e evitando interrupções no serviço.
  + Melhoria no desempenho do OLTP: A replicação do PostgreSQL melhora o desempenho do processamento transacional online (OLTP), reduzindo o tempo de processamento de transações e consultas, ao remover a carga de consultas de relatórios do servidor primário, resultando em um sistema OLTP mais ágil e responsivo.
  + Teste em paralelo do sistema: A replicação do PostgreSQL permite a criação de uma cópia do banco de dados de produção para testes em paralelo, garantindo que um novo sistema funcione corretamente com os dados existentes antes da implantação completa, sem afetar o ambiente de produção.
* **Elementos da replicação do PostgreSQL**
  + **Replicação física e lógica:**

A replicação PostgreSQL pode ser realizada de duas maneiras: replicação lógica e replicação física.

A replicação física lida com arquivos e diretórios, sendo usada para manter uma cópia completa dos dados em outro servidor. Ela opera no nível do sistema de arquivos ou disco, usando endereços de bloco exatos.

Já a replicação lógica reproduz entidades de dados e suas modificações com base em sua identidade de replicação. Ela lida com bancos de dados, tabelas e operações de DML, e é realizada no nível do cluster de bancos de dados. A replicação lógica utiliza um modelo de publicação e assinatura, onde os assinantes recebem dados das publicações e podem republicá-los para permitir configurações mais complexas.

A replicação lógica é útil para enviar mudanças incrementais, compartilhar subconjuntos de dados, consolidar bancos de dados e fornecer acesso a dados replicados para diferentes grupos de usuários. O PostgreSQL suporta ambos os mecanismos simultaneamente, permitindo um controle fino sobre a replicação de dados e segurança.

* + **Síncrono e assíncrono:**

A transação no servidor principal só é considerada confirmada quando os servidores secundários confirmam que a receberam e a aplicaram com sucesso. Em outras palavras, o servidor principal espera pela confirmação dos servidores secundários antes de finalizar a transação. Esse tipo de replicação garante que os servidores secundários estejam sempre atualizados e em sincronia com o servidor principal.

A replicação síncrona é considerada mais segura, pois oferece consistência máxima entre os servidores. No entanto, ela pode ter um impacto maior no desempenho, já que as transações podem levar mais tempo para serem concluídas, uma vez que dependem da confirmação dos servidores secundários.

* **Como funciona a replicação PostgreSQL**
  + **Streaming:**

A replicação do PostgreSQL, também conhecida como replicação WAL, envolve a configuração dos servidores primário e secundário após a instalação. O processo consiste em mover os arquivos WAL (Write-Ahead Logs) do banco de dados primário para o banco de dados secundário, garantindo uma cópia idêntica. O servidor primário lida com operações de leitura/gravação, enquanto o servidor secundário é apenas para leitura. A configuração envolve a configuração de ambos os servidores e será detalhada nas seções a seguir.

* + **Dispositivo de bloco replicado:**

O método de dispositivo de bloco replicado envolve o espelhamento síncrono de volumes persistentes para escrever mudanças em outro volume. Essa abordagem é compatível com diversos bancos de dados relacionais, incluindo PostgreSQL, MySQL e SQL Server, e é adequada para ambientes em nuvem. No entanto, requer a replicação tanto dos logs WAL quanto dos dados da tabela, com o objetivo de garantir a durabilidade dos dados. Esse método é comumente usado com serviços como o Azure PostgreSQL e o Amazon RDS.

* + **Replicação baseada em log (WAL - Write-Ahead Log):**

Replicação baseada em log (log shipping): Nesse método, as alterações no banco de dados são gravadas em um arquivo de log (WAL - Write-Ahead Log). Os arquivos de log são copiados e transferidos para outros servidores, onde são aplicados para atualizar os bancos de dados replicados. Essa abordagem é assíncrona, o que significa que pode haver um pequeno atraso entre a atualização no banco de dados principal e a replicação nos servidores secundários.

* **Como configurar a replicação do PostgreSQL**
  + **Primeiramente instalação do PostgreSQL no Linux:**
    - A instalação do postgreSQL é obrigatória tanto para os servidores primários quanto para os secundario antes do processo de replicação
  + **Configurando a replicação no servidor primário**
    - Primeiramente faço o login no banco de dados

| **su - postgres** |
| --- |

* + - Crie um usuário replicador com permissões de **REPLICATION**

| **psql -c "CREATEUSER replication REPLICATION LOGIN CONNECTION LIMIT 1 ENCRYPTED PASSWORD'YOUR\_PASSWORD';"** |
| --- |

* + - Edite o arquivo **pg\_hba.cnf**

| **nano /etc/postgresql/{versão postgre}/main/pg\_hba.conf** |
| --- |

* + - Adicione

| **host replication replication {IP Servidor Secundario}/24 md5** |
| --- |

* + - Abra o **postgresql.conf**

| **nano /etc/postgresql/{versão postgre}/main/postgresql.conf** |
| --- |

* + - Edite o **postgresql.conf** colocando a seguinte configuração no servidor primário:

| **listen\_addresses = '\*'**  **wal\_level = replica**  **wal\_keep\_segments = 64**  **max\_wal\_senders = 10** |
| --- |

* + - Agora podemos reiniciar o postgreSQL no servidor principal
  + **Configurando a replicação no servidor secundário:**
    - Edite o arquivo **pg\_hba.conf**

| **nano /etc/postgresql/{versão postgre}/main/pg\_hba.conf** |
| --- |

| **host replication replication {IP Servidor Primario}/24 md5** |
| --- |

* + - Abra e edite o **postgresql.conf,**

| **nano /etc/postgresql/{versão postgre}/main/postgresql.conf**  **listen\_addresses = 'localhost,IPSecundario'**  **wal\_keep\_segments = 64**  **wal\_level = replica**  **hot\_standby = on**  **max\_wal\_senders = 10** |
| --- |

* + - **Remova todas os dados do postgreSQL secundario**

| **cd /var/lib/postgresql/{versão postgre}/main**  **rm -rfv \*** |
| --- |

* + - Copie os dados do servidor primário para o servidor secundário (-R onde automaticamente cria o arquivo **recovery.conf**, que é obrigatório para o servidor secundário)

| **pg\_basebackup -h MasterIP -D /var/lib/postgresql/{versão postgre}/main/ -P -U replication --wal-method=fetch -R** |
| --- |

* + - Adicione os seguintes configurações para o arquivo **recovery.conf** (A opção ***standby\_mode*** sempre tem que ser **ON** para servidores secundários e as configurações do stream WAL data é pela opcão ***primary\_conninfo*** )

| **nano /var/lib/postgresql/{versão postgre}/main/recovery.conf** |
| --- |

| **standby\_mode = 'on'**  **primary\_conninfo = 'host={IP Servidor Primario} port=5432 user=replication password='YOUR\_PASSWORD'**  **trigger\_file = '/tmp/MasterNow'** |
| --- |

* + - **Apenas resta reiniciar o servidor**
* **O que é failover e como ele funciona:**

Failover é um termo usado em sistemas de alta disponibilidade e recuperação de desastres para descrever a transição automática e transparente de um componente ou sistema principal (primário) para um componente ou sistema de backup (secundário) quando ocorre uma falha. O objetivo do failover é minimizar o tempo de inatividade e manter a continuidade do serviço.

Em um contexto de banco de dados, como o PostgreSQL, o failover ocorre quando o servidor principal, que normalmente hospeda o banco de dados em operação, falha ou se torna indisponível. Nesse momento, um servidor secundário ou de backup assume o papel de servidor principal para garantir a continuidade do serviço.

O failover pode ser implementado de diferentes maneiras, dependendo da arquitetura e da configuração do sistema. Alguns métodos comuns de failover incluem:

Replicação síncrona: Se a replicação síncrona estiver em vigor, o servidor secundário está em sincronia constante com o servidor principal. Quando ocorre uma falha no servidor principal, o servidor secundário assume automaticamente o papel de servidor principal. O failover é transparente para os clientes, que são redirecionados automaticamente para o novo servidor principal.

Monitoramento e detecção de falhas: Um sistema de monitoramento contínuo é usado para verificar o status do servidor principal. Se o servidor principal não responder ou apresentar algum problema, um mecanismo de detecção de falhas é ativado. Esse mecanismo pode ser um processo automatizado que identifica a falha e inicia o processo de failover.

Configuração de cluster: Em um ambiente de cluster, vários nós estão configurados para trabalhar juntos como um único sistema. Se um nó falha, os outros nós assumem automaticamente as funções e responsabilidades do nó falho, permitindo que o sistema continue operando sem interrupção.

É importante ressaltar que a configuração de um failover efetivo requer planejamento e implementação cuidadosos. É necessário garantir que os servidores secundários estejam devidamente configurados, sincronizados e prontos para assumir o papel de servidor principal. Além disso, é importante monitorar e testar regularmente o mecanismo de failover para garantir sua eficácia.

O failover desempenha um papel fundamental na garantia da alta disponibilidade e na minimização do tempo de inatividade em sistemas críticos, permitindo que as operações continuem mesmo em caso de falhas.