Replicação de Banco de Dados PostgreSQL

Arthur Delai, Miguel Capitanio e Paulo Lazzarotto

Departamento de Engenharias e Ciência da Computação Universidade
Regional Integrada do Alto Uruguai e das Missões (URI)
Erechim - RS - Brasil - 2024
099495@aluno.uricer.edu.br 091689@aluno.uricer.edu.br
099497@aluno.uricer.edu.br

1. Introdução

A replicação de bancos de dados é um aspecto indispensável para qualquer aplicação web que se utiliza de BDs. É uma forma de assegurar a proteção e disponibilidade de dados através de backups realizados de forma regular de um BD para outro. “A replicação de dados é um processo de backup automatizado no qual seus dados são repetidamente copiados de seu banco de dados principal para outro local remoto para serem mantidos em segurança (Salman Ravoof).”

Através de métodos de balanceamento de carga e escalabilidade, é garantida a disponibilidade para uma grande quantidade de usuários, proporcionando maior segurança em casos de ataques de DDOS. Além disso, a tolerância a falhas também é um as. “Se o servidor primário falhar, o servidor standby pode agir como um servidor porque os dados contidos tanto para o servidor primário quanto para o standby são os mesmos (Salman Ravoof)."

Há várias maneiras de aplicar a replicação de bancos de dados, dependendo das necessidades de quem for implementá-la, como por exemplo: master-master; master-slave; updates síncronos e assíncronos; entre diversos outros.

2. Modos de Replicação

A replicação de banco de dados pode ser realizada de diferentes formas, cada uma com suas vantagens, desvantagens e casos de uso próprios.

2.1 Replicação Síncrona

Na replicação síncrona, os dados são escritos em ambos os bancos ao mesmo tempo, sendo consideradas “completas” quando ambas forem finalizadas. Sendo consideravelmente mais “custosa”, com problemas de performance, pois a escrita de dados está sendo realizada em vários nós de execução diferentes, sequencialmente, trafegar essa carga de dados via uma rede pode causar lentidão ou até mesmo parada do funcionamento do sistema.

Para remediar estes problemas de lentidão, Böszörmenyi e Schönig sugerem, entre outras coisas:

- **Utilizar COMMITS**: Dentro das configurações do PostgreSQL existe a configuração *synchronous_commit*, que permite confirmar a escrita correta dos dados em ambos os bancos de dados, garantindo a proteção dos dados a serem salvos;

- **Usar transações maiores**: Operando sob as regras dos COMMITS nos bancos de dados, utilizando e enviando pacotes maiores pode reduzir dramaticamente a comunicação dentro da rede;

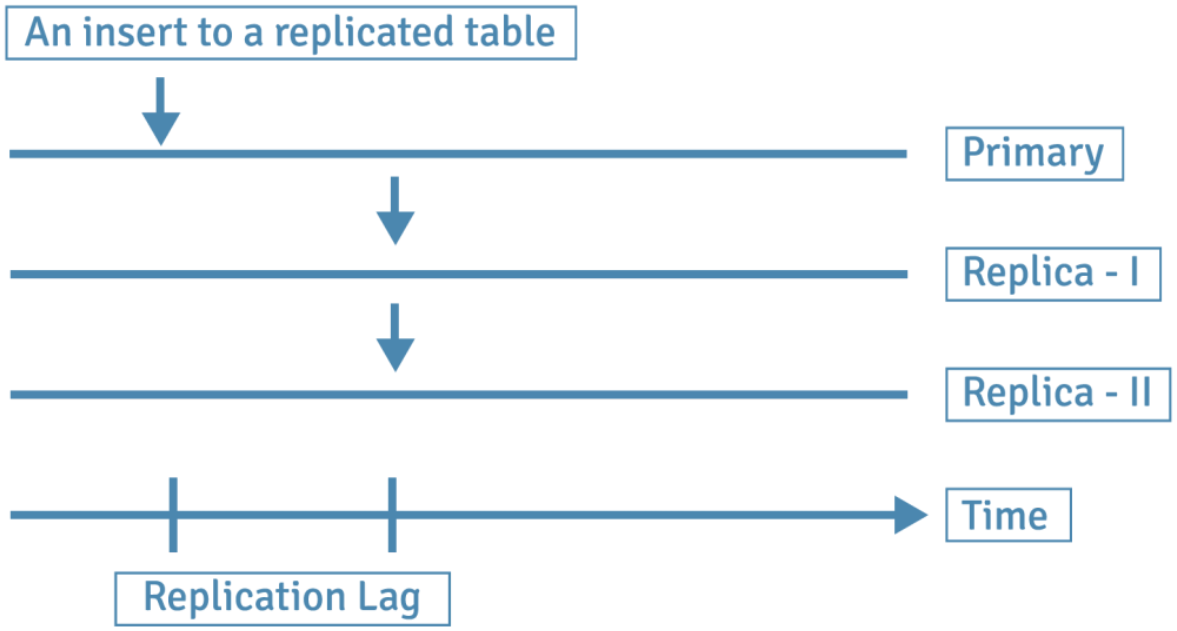
- **Execução concorrente**: Caso existam 2 ou mais operações acontecendo ao mesmo tempo, é sábio executá-las ao mesmo tempo, garantindo que várias transações ocorram e sejam confirmadas conjuntamente.

2.2 Replicação Assíncrona

Na replicação assíncrona, as informações são replicadas para o banco de dados primário e depois para o secundário mais tarde, sem a necessidade de todos os componentes estarem disponíveis a todo tempo, como na replicação síncrona.

Com esse modo de funcionamento, mira não causar sobrecarga, podendo ter realização agendada, ocorrendo de hora em hora ou uma vez por dia.

Uma preocupação sobre este método de replicação é o “Lag de Replicação”, que é o tempo de demora entre escrita no banco de dados principal e nos bancos secundários.



Lag de Replicação

A IBM cita 2 exemplos de modelos que podem ser utilizados na replicação assíncrona:

- **Replicação por Alvo Primário (Primary-target Replication)**: Onde o fluxo de dados é unidirecional, originando no banco de dados primário e sendo replicado para os demais;

- **Replicação Atualize-Tudo (Update-Anywhere)**: Onde as mudanças que ocorrem em um banco de dados são replicadas para todas as outras, este modelo permite a utilização de vários bancos de dados de escrita e consulta, funcionando autonomamente.

3. Estratégias de Replicação

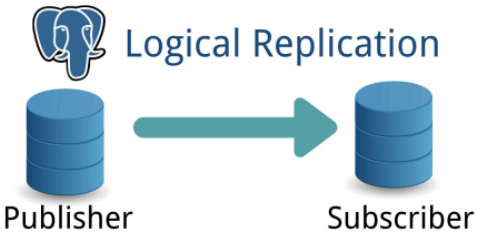
Para realizar e aplicar as diferentes formas de replicação, podem ser abordadas diferentes estratégias, novamente, cada uma dessas estratégias apresenta características e casos de usos diferentes, dependendo da necessidade de utilização.

3.1 Replicação Transacional/Lógica

Segundo a documentação oficial do PostgreSQL, replicação lógica é um método de replicação baseado na identidade de replicação (normalmente chave primária), utilizando-se de publishers e subscribers. O termo remete à replicação física, onde os blocos exatos de endereçamento são utilizados para replicação byte-a-byte.

Publications (Publicações) são conjuntos de mudanças geradas por uma ou mais tabelas, um nó onde as publicações são definidas é chamada de *Publisher*. Publicações não alteram o esquema original de uma tabela, se diferenciando de schemas, uma tabela pode aparecer em uma ou mais publicações.

Subscriptions (Inscrições) é o lado lógico da replicação, um nó onde uma inscrição é definida é chamada de *Subscriber* (Inscrito). Uma inscrição define a conexão para outro banco de dados e conjunto de publicações a serem realizadas, cada inscrito pode receber uma ou mais inscrições.



Replicação Transacional/Lógica

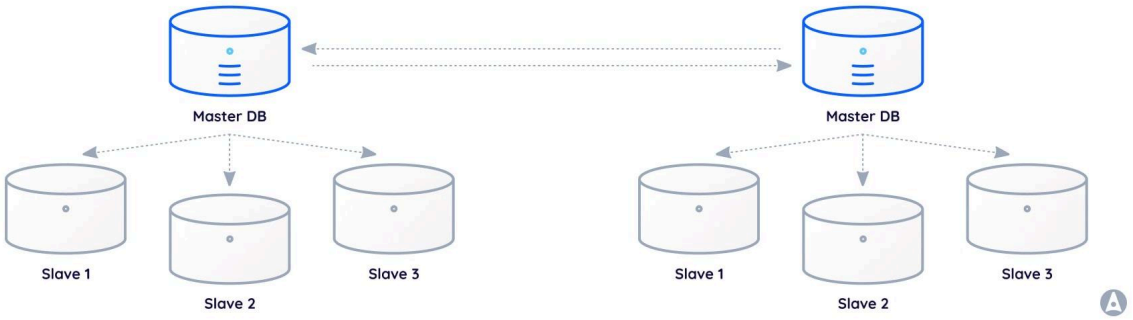
3.2 Replicação Master-Master

Composta por bancos primários (com leitura e gravação de dados), replicação Master-Master (ou Multi-Master) é uma forma de replicação em que cada banco de dados pode passar as informações para os outros, sendo capazes tanto de escrita como de leitura.

Entretanto, alguns problemas são gerados por este modelo, sendo um deles o conflito de informações. Caso dois bancos estejam sendo atualizados ao mesmo tempo pode ocorrer inconsistência, pois ambos possuem diferenças. Isto pode ser resolvido através de algoritmos de consenso ou levando em conta a hora em que os

updates foram feitos, mantendo aquele que foi feito mais recentemente e descartando o outro.

Uma das vantagens do modelo Master-Master é a tolerância a falhas. Pelo motivo de existirem diversos servidores primários, caso um deles caia ainda será possível realizar leitura e escrita. Porém ainda existe o problema da inconsistência de dados.



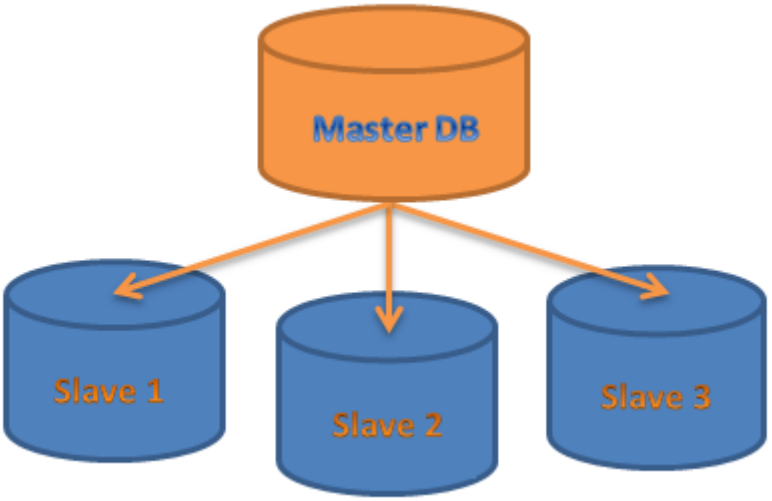
Replicação Master-Master/Multi-Master

3.3 Replicação Master-Slave

Dados de um banco de dados primário (master) que são repassados como backup para servidores secundários (slaves), onde funcionam somente operações de leitura, escrita sendo reservadas para o banco de dados primário (LESMANA). Esta estratégia pode ser síncrona ou assíncrona.

Utilizada principalmente para reduzir a carga de execução nos bancos de dados, caso existam muitos registros ou a ação de leitura de dados é executada constantemente, delegar a leitura para ser feita somente no servidor slave enquanto a escrita pode ser reservada para o master. Assim, garante que ambas as partes realizem o “seu peso” do processamento separadamente.

Outra utilização seria para backups, mantendo um ou mais servidores slaves em *standby*, para oferecer redundância dos dados, com backups diários, semanais ou até mesmo mensais, garantindo um meio de recuperação caso ocorram desastres.



Replicação Master-Slave

4. PostgreSQL

PostgreSQL é um programa de criação de bancos de dados relacionais que utiliza a linguagem SQL. Ele apresenta diversas ferramentas e funcionalidades, entre elas a possibilidade de realizar replicações dos bancos criados, havendo muitas formas diferentes de realizar essa replicação.

4.1 Replicação em PostgreSQL

A replicação em bancos de dados PostgreSQL se dá na cópia de dados e informações de um banco para outro, geralmente envolvendo um banco de dados primário e um ou mais que funcionam como réplica. A replicação é de suma importância para que haja disponibilidade e segurança dos dados armazenados. “A configuração da replicação entre duas bases de dados oferece tolerância a falhas contra contratempos inesperados” (Salman Ravoof).

Seguindo um modelo simples, a escrita é controlada por um nó primário, sendo, em seguida, transferida para os nós secundários, podendo haver, também, diversos nós primários.

4.2 Replicação por Streaming

Mover arquivos Write-Ahead-Logging (WAL) do banco primário para os demais. Estes arquivos armazenam as informações das operações realizadas no

banco de dados, permitindo a replicação das mesmas no banco de dados secundário.

“O servidor primário opera em modo de arquivamento contínuo, enquanto cada servidor em espera opera em modo de recuperação contínua, lendo os arquivos WAL do primário” (POSTGRESQL).

4.3 Replicação por Log

Utilizando um servidor intermediário, é possível utilizar registros WAL para replicar continuamente registros de um banco, utilizando Log Sequence Record (LSN), que aponta para um registro WAL, garantindo a veracidade do registro.

O PostgreSQL realiza envio de logs com base em arquivos ao transferir registros WAL um de cada vez, os quais são enviados de forma fácil e barata de qualquer distância.

“Deve-se observar que o envio de logs é assíncrono, ou seja, os registros WAL são enviados após o commit da transação. Como resultado, há uma janela para perda de dados caso o servidor primário sofra uma falha catastrófica” (POSTGRESQL).

5. Vantagens e Desvantagens da Replicação

Como visto anteriormente, a replicação de banco de dados oferece diferentes modelos e métodos para serem aplicados, com isso, é interessante investigar alguns de seus pontos positivos e negativos.

5.1 Vantagens

Migração de dados: A replicação pode ser utilizada para migrar dados de um servidor para outro sem sacrificar a operacionalidade do servidor atual;

Tolerância a falhas: Se o servidor primário falhar, os servidores em standby irão fazer verificações sobre a integridade dos dados e logo após podem assumir a posição do primário;

Escalabilidade: Permite a escalabilidade no acesso aos dados, balanceando a carga conforme a demanda;

Redundância de Dados: Garante a atomicidade de dados via a comparação dos mesmos e operações como commits, que confirmam a execução de transações ou seu cancelamento (JIMENEZ-PERIS, KEMME, PATIÑO-MARTÍNEZ).

5.2 Desvantagens

Complexidade: Configuração e manutenção dos servidores apresenta desafios em relação à infraestrutura e armazenamento;

Custo: Manter vários bancos de dados/servidores para garantir a atomicidade dos dados, implica em maiores custos em manutenção e integridade dos mesmos;

Inconsistência: Devido ao tempo ocupado entre a replicação e a atualização do banco de dados primário, podem ser geradas inconsistências, as quais requerem medidas complexas para serem remediadas.

6. Conclusão

A replicação de banco de dados serve como pilar para o desenvolvimento e manutenção de diversos serviços e sistemas distribuídos, a capacidade de garantir a atomicidade e veracidade dos dados é de extrema importância para o funcionamento correto dos mesmos.

Entendendo suas diferentes formas, modelos, estratégias, características, vantagens e desvantagens, aplicados em soluções como PostgreSQL, foi possível escolher quais as melhores opções que agregam para o conhecimento de tal tecnologia.

7. Referencial

AIRBYTE. MySQL Master-Slave Replication: 6 Easy Steps. 2024. Disponível em: <https://airbyte.com/data-engineering-resources/master-slave-replication>. Acesso em: Setembro de 2024.

BÖSZÖRMENYI, Zoltan; SCHÖNIG, Hans-Jürgen. PostgreSQL Replication. 2013.

IBM. Asynchronous Data Replication. 2024. Disponível em: <https://www.ibm.com/docs/pt/informix-servers/14.10?topic=overview-asynchronous-data-replication>. Acesso em Setembro de 2024.

JIMENEZ-PERIS, Marta; KEMME, Bettina; PATIÑO-MARTÍNEZ, Ricardo. Database Replication. 2010.

POSTGRESQL. PostgreSQL 16.0 Documentation. 2024. Disponível em: <https://www.postgresql.org/docs/current/>. Acesso em Setembro de 2024.

RAVOOF. Salman. Replicação PostgreSQL: Um Guia Detalhado, 2023 Disponível em <https://kinsta.com/pt/blog/replicacao-postgresql/>. Acesso em Setembro de 2024.