Replicação

A replicação permite que os dados de um servidor (o mestre) sejam copiados a um ou mais servidores (os escravos). A replicação é assíncrona por definição: os escravos não precisam estar conectados ao mestre. Dependendo da configuração pode-se replicar todos os BDs, alguns DBs ou mesmo algumas tabelas dentro de um DB. As vantagens:

- Escalabilidade: ao distribuir a carga por diversos servidores, pode-se dar vazão a maior demanda. Aqui, escritas e atualizações devem ser feitas apenas no mestre, mas as leituras podem ser espalhadas. Assim o mestre pode ser dedicado a atualizações, enquanto as leituras podem crescer à vontade.
- Segurança: backups podem ser feitos num servidor escravo sem interromper nada.
- Distribuição geográfica: escravos podem ser locais de um mestre remoto.
- Balanceamento de carga: cada servidor pode ter um limite de carga a atender antes que comecem a se formar filas. Ocorrendo isto, mais servidores podem ser criados.

Como não existe almoço grátis, depois da lista de vantagens vem a correspondente lista de desvantagens.

- Aumento no tráfego de rede: consequência imediata, já que cada atualização no servidor precisa ser replicada aos escravos.
- Atraso na informação: Pode-se escolher replicação *just in time* ou apenas de tempos em tempos, mas nos dois casos, algum atraso precisa ser esperado.
- Risco potencial de perda de integridade: a execução em paralelo de bases distribuídas pode levar a situações em que o banco se perca. Por exemplo, suponha uma informação numérica correta valendo 11. Na mesma hora um servidor incrementa este valor e outro o decrementa. Qual o valor correto ? Este fato sugere cuidados na replicação (e na aplicação, por certo)
- Crescimento da complexidade do ambiente e de todas as suas operações. Nem por ser óbvio, este fator deve ser desconsiderado.

A replicação tradicional exige consulta aos arquivos de log do servidor para manter a sincronia dos arquivos. Um novo método baseado em GTI (*global transactions identifiers* que usa o conceito de transação atômica e transfere a responsabilidade da replicação para outra camada do software. É idéia bastante moderna.

Tipo Mestre-Escravo

Talvez o tipo mais comum, esta replicação conta com um mestre que é o único que pode fazer alterações e um ou mais servidores escravos que disponibilizam apenas informações para leitura. O balanceamento é óbvio. Ao computador com mais recursos reserva-se o papel de cuidar do banco de dados, enquanto todas as leituras são desviadas. Uma vantagem adicional aqui, é que havendo um defeito no servidor de atualização, esta tarefa pode facilmente ser assumida por qualquer dos escravos. Cada operação de inserção, alteração, ou exclusão feita no mestre é armazenada numa lista de sincronização para que seja repassada a todos os escravos. A sincronização pode ser dar na hora da atualização ou de tempos em tempos (a cada hora, por exemplo). Esta estratégia é indicada quando o volume de alterações é baixo, enquanto o volume de consultas é alto. Um exemplo: um portal de notícias na Internet, ou um servidor de vídeos em tempo real.

Tipo Mestre-Mestre

Aqui a divisão entre servidores se dá por características do negócio (por exemplo, um servidor para pessoal e outro para contabilidade). Todos os servidores podem atualizar e eles atendem tanto a consultas de atualização quando de leitura. A mecânica é a mesma do caso anterior: em momentos certos, cada servidor informa aos demais quais as alterações que todos eles deverão efetuar.

Tipo Mestre-Mestre e Mestre-Escravo

Este caso mescla os dois anteriores e é o estado da arte em replicação (com toda a complexidade associada, é claro). Há vários mestres que se sincronizam entre sí e ligado a cada mestre há os escravos *(read only)* que disponibilizam cópias dos arquivos para consulta. O MySQL até a versão 5.0 apenas a versão mestre-escravo era aceita. Na versão 5.1 já é possível fazer a replicação mestre-mestre.

A replicação é realizada a partir dos logs binários mantidos no servidor mestre e constantemente acessados pelos escravos que buscam e executam as operações. Para configurar a replicação há que se habilitar o log binário no servidor mestre e também migrar a estrutura e a imagem do banco de dados antes do início da replicação.

Antes de continuar uma lembrança: Recomenda-se fortemente que todos os servidores envolvidos em uma replicação sejam da mesma versão do software (e tão parecidos quanto possível) para evitar as desconhecidas, mas esperadas, incompatibilidades.

O Processo: 1. Configuração do servidor

O arquivo `my.cnf` (linux) ou o `my.ini` (windows) devem ter no bloco `[mysqld]` as seguintes linhas

```
[mysql]
log-bin=mysqlmaster
server-id=1
```

A primeira especificação diz que o MySQL deverá armazenar todas as operações em um log chamado `mysqlmaster`. (ou o nome que você quiser). A segunda especificação dá a identificação do servidor que deve ser única. Depois de fazer estas alterações, reinicie o MySQL. Para ver seu funcionamento, o comando pode ser `SHOW MASTER STATUS`.

2. Criar um usuário que acesse a replicação

Deve-se criar uma conta administrativa apenas para ler a replicação. o comando é

```
GRANT SUPER, RELOAD, SELECT,
   REPLICATION SLAVE ON *.*
   TO usuario@dominio IDENTIFIED BY senha
```

O domínio é opcional, mas fortemente indicado, para limitar os poderes de copiar tudo o que acontece no seu sistema. Os direitos de SUPER, RELOAD e SELECT são para permitir a emissão dos comandos LOAD TABLE FROM MASTER ou LOAD DATA FROM MASTER para carregar a situação inicial do servidor.

3. Configurando o escravo

Editar o arquivo `my.ini` colocando nele

```
server-id=2
```

Depois de reiniciar o escravo, dando a ele uma identificação única, deve-se aplicar o comando

```
CHANGE MASTER TO
MASTER_HOST = endereço ip ou url do master
MASTER_PORT = porta onde o servidor está config.
MASTER_USER = usuário com acesso ao log binário
MASTER_PASSWORD = senha
MASTER_LOG_FILE = vazio ou um arquivo
MASTER_LOG_POS = zero ou linha de início
```

As últimas 2 entradas servem para uma replicação a partir de um determinado ponto e não desde o início. Pode-se obter informações sobre o escravo através do comando `SHOW SLAVE STATUS`.

4. Migrando a imagem

Primeiro, embora não obrigatório é muito recomendável que esta atividade seja feita enquanto o servidor mYSQL estiver desabilitado de fazer alterações. Uma maneira possível é exportar os dados do servidor e importá-los no escravo. Lento, mas funcional. Outra é através dos comandos LOAD TABLE FROM MASTER e LOAD DATA FROM MASTER. Na versão 5.0 este recurso só admitia tabelas do tipo MyISAM. Deve-se evitar migrar as tabelas do sistema, pois cada servidor (computador) possui as suas idiossincrasias nelas registradas.

5. Iniciando a replicação

Depois de terminar a configuração do escravo por meio do comando CHANGE MASTER TO, deve-se executar o comando `START SLAVE;` que dirá ao servidor em questão para iniciar os procedimentos de replicação. Para interromper o processo, o comando é `STOP SLAVE;`. Obviamente, após o início da replicação, deve-se testar o ambiente, realizando uma alteração no mestre e consultando-a logo depois no escravo. Não havendo coincidência deve-se examinar o LOG do escravo para determinar o ou os erros acontecidos.

A definição de quais bancos ou tabelas deverão ser replicadas é feita no mestre, configurando a variável `binlog-do-db = banco1, banco2, ....` Neste caso todos os bancos não citados aqui não serão gravados no arquivo de log. Também pode-se escolher quais tabelas serão replicadas no servidor através da variável `replicate-do-table = banco.tabela1, ...` Estas definições podem ser emitidas também no escravo, o que implica severo estudo e simulação antes de implementar o esquema final. Para configurar uma replicação mestre-mestre, basta repetir o processo acima, agora invertendo os papéis entre mestre e escravo.