SISTEMA GERENCIADOR DE BANCO DE DADOS

AULA 4

Prof. Leonel da Rocha

CONVERSA INICIAL

Aqui, veremos alguns cuidados que devemos ter com um sistema gerenciador de banco de dados e o conteúdo por ele gerenciado. Veremos como a manutenção de um banco é importante para manter os dados em boas condições. Outro ponto importante que abordaremos será a importação e exportação de dados, tanto para outros bancos de dados de um mesmo SGBD quanto para outros gerenciadores e sistemas. Veremos a replicação de dados, que é uma ferramenta importante no gerenciamento de dados. Trataremos da migração de dados que possibilita fazer a transferência de base de dados para versões atualizadas de um SGBD ou de outros sistemas. Por último, analisaremos rotinas de limpeza e ocupação de disco que precisam de uma atenção especial, pois é nesse ambiente que o SGBD funciona.

TEMA 1 – MANUTENÇÃO DE UM SGBD

Para tratarmos da manutenção de um banco de dados, é importante saber que o estado dos dados afeta sua disponibilidade e eficiência, sendo necessário mantê-los em boas condições. É sob essa ótica que a manutenção de dados deve ser tratada. Uma das responsabilidades de um DBA é cuidar e proteger os dados contra acesso não autorizado. Quando essa função não é executada, usuários externos podem obter acesso rápido aos dados.

A manutenção de um banco de dados é um conjunto de atividades que visa a garantir seu bom funcionamento. As soluções de gerenciamento de um banco de dados exigem atualizações regulares para acomodar novas tecnologias e evitar riscos de segurança. Os assuntos de que estamos tratando, como backups, exportação/importação de dados, revisões de índice, exclusão de objetos, expurgo de dados antigos, são alguns exemplos de manutenção de banco de dados.

Os sistemas de gerenciamento de bancos de dados são utilizados para manter um repositório de dados valiosos. Para ter acesso rápido a eles, são necessários índices bem organizados, que poderão acelerar a extração dos dados. Os administradores podem optar por usar um sistema alternativo para indexação e outros propósitos. As modificações no sistema gerenciador de banco de dados podem fazer que aconteçam falhas com o passar do tempo. É para isso que a manutenção é importante, pois auxiliará a manter o sistema funcionando perfeitamente, evitando perda dos dados, por exemplo.

A manutenção do banco de dados deve ser realizada por especialistas, que precisam conhecer os recursos e funções específicas para a atividade. Ele deve conhecer, por exemplo, um dos componentes mais importantes da manutenção do banco de dados, que é o backup dos dados. Se algum desastre acontecer, os dados estarão seguros e poderão ser restaurados.

Figura 1 – Manutenção de banco de dados

Créditos: dny3d / Shutterstock.

A manutenção de dados é fundamental para toda a organização e seus negócios. Ter boas rotinas de manutenção de dados tem muitas vantagens. Isso demonstra por qual motivo as organizações líderes não abrem mão da implementação de um bom programa de manutenção.

Para que um banco de dados tenha um aumento na sua eficiência, é importante que sua manutenção esteja em dia. Para isso será necessário, entre outras ações, remover dados repetidos ou que não serão utilizados. Essa redução de dados poderá diminuir o tempo de acesso a eles. No gerenciamento de um banco de dados, é necessário disponibilizar o acesso aos dados com rapidez e

que eles reflitam o ambiente operacional sem causar atrasos nem, por consequência, aumento nos custos.

A manutenção do banco de dados bem planejada prevê backups realizados e arquivados em locais de fácil acesso quando precisam ser manipulados para o processo de recuperação de dados. A manutenção regular do banco de dados pode auxiliar a reduzir o tempo de inatividade.

Para otimizar os resultados da manutenção do banco de dados, é necessário planejá-la e executá-la com eficiência. Senão, é possível que alguns detalhes importantes sejam ignorados. A precisão é fundamental para a implementação da manutenção. Lembrando que, sem um gerenciamento de banco de dados eficaz, a precisão dos dados será prejudicada, por isso a importância da manutenção.

Para a manutenção de um banco de dados, as seguintes recomendações devem ser observadas: ao armazenar os dados em um único arquivo, a agregação de dados, em alguns casos, pode dificultar o gerenciamento de banco de dados. Salvar todos os dados em um único SGBD ou database poderá evitar que se tenha de procurar os dados quando for preciso atualizar os arquivos de dados e, até mesmo, recuperá-los.

Para ajudar na manutenção dos dados, é importante que eles sejam identificados de forma clara. No projeto de um banco de dados, quando é possível implementar as identificações, é recomendável que os rótulos que identificam os objetos, tabelas e colunas nos permitam identificá-los. Por exemplo, que uma tabela em que serão armazenados os dados dos funcionários seja nomeada como *funcionario* e suas colunas identifiquem seus dados, como nome, endereço, documentos e datas. Outra situação importante é que todas as alterações pertinentes aos dados do sistema sejam realizadas por meio dele, sem criar controles externos que poderão dificultar a manutenção.

Além disso, mostra-se importante na manutenção dos dados a atualização dos bancos, sendo necessário o seu acompanhamento, pois, quando um sistema é utilizado, as inserções e atualizações geram páginas de dados onde estes são armazenados. Porém, quando o acompanhamento não é feito, com o passar do tempo, essas páginas tornam-se um gargalo durante as pesquisas de acesso aos dados.

É importante elaborar um plano de manutenção do banco de dados. Um cronograma deve ser elaborado e precisa ser rigorosamente seguido para melhorar a eficiência. Um plano de manutenção

é uma estratégia de segurança que deve implementada em todos os bancos de dados para garantir a prevenção de riscos e o desempenho ideal.

A seguir, algumas ações que devem ser implementadas em todos os planos de manutenção de um banco de dados:

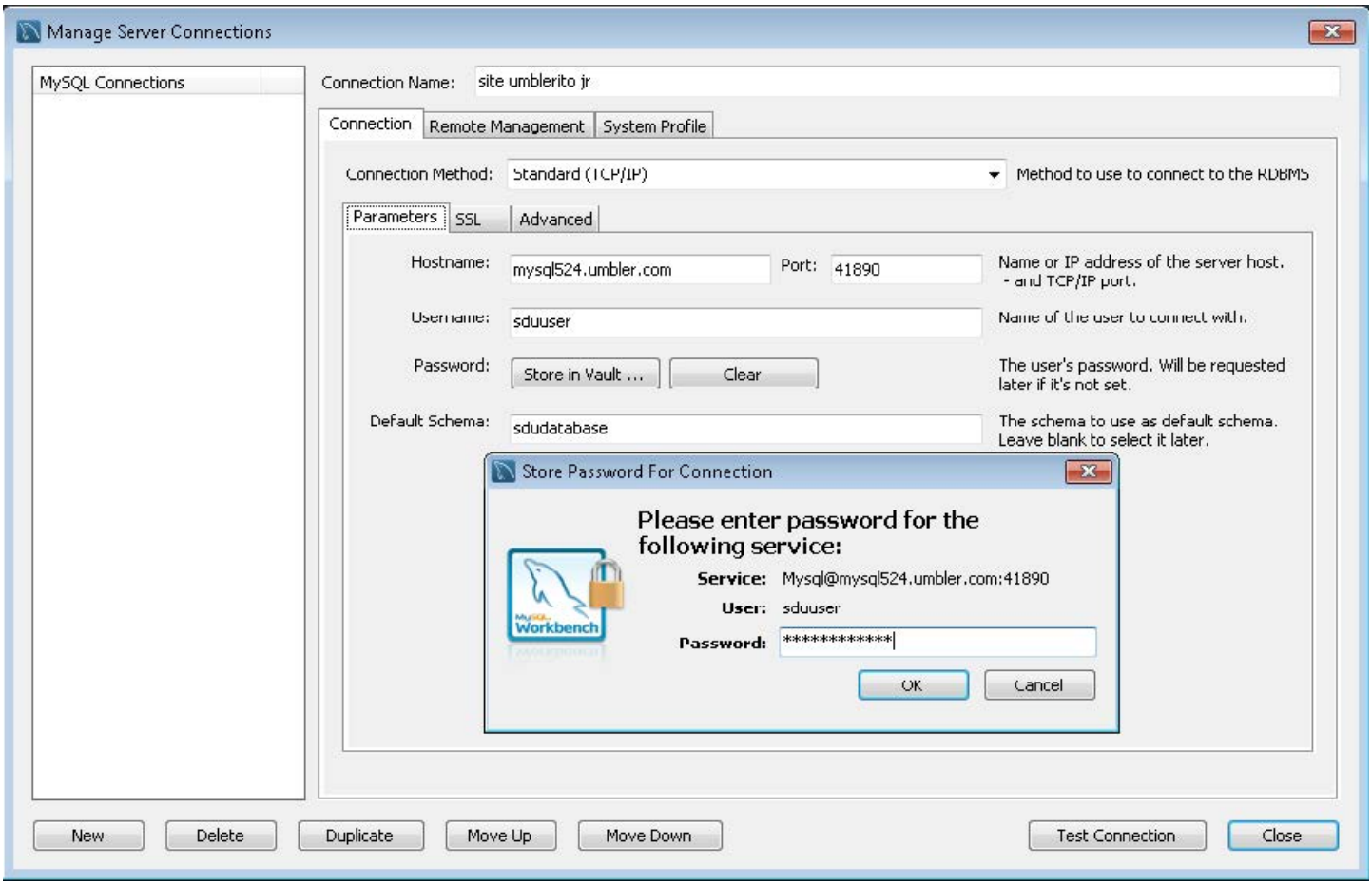
- reorganização e redução do tamanho de tabelas (OPTIMIZE);
- rotinas de desfragmentação dos índices;
- coleta de estatísticas de desempenho;
- backup do MySQL;
- utilizar o mysqlcheck para verificar, reparar, otimizar e analisar tabelas; e
- usar o comando CHECK TABLE para realizar a verificação de dados.

TEMA 2 – IMPORTAR/EXPORTAR DADOS

Para a realização da exportação e importação dos arquivos de banco de dados MySQL, é recomendável a utilização do MySQL Workbench, que poderá ser acessado com os dados de acesso externo.

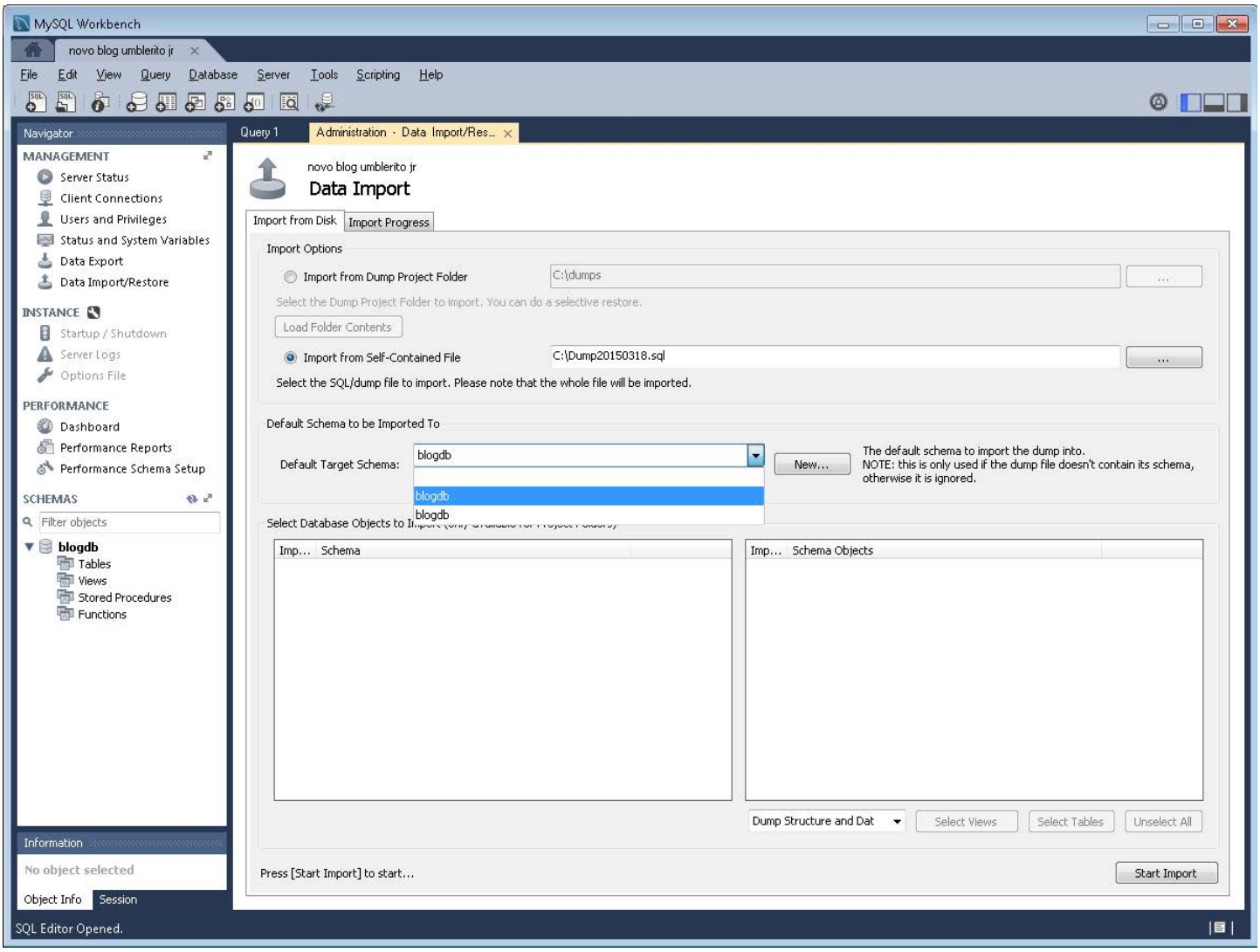
Para importar um arquivo, vamos abrir o Workbench e clicar em + ao lado de MySQL Connections. Os campos de usuário e senha devem ser preenchidos para fazer uma nova conexão.

Figura 2 – MySQL Connections



Ao acessar o banco a ser importado, selecionar Data Import/Restauration. A opção Import from Self-Contained File deve ser selecionada, e o arquivo desejado precisa ser informado para a realização da importação.

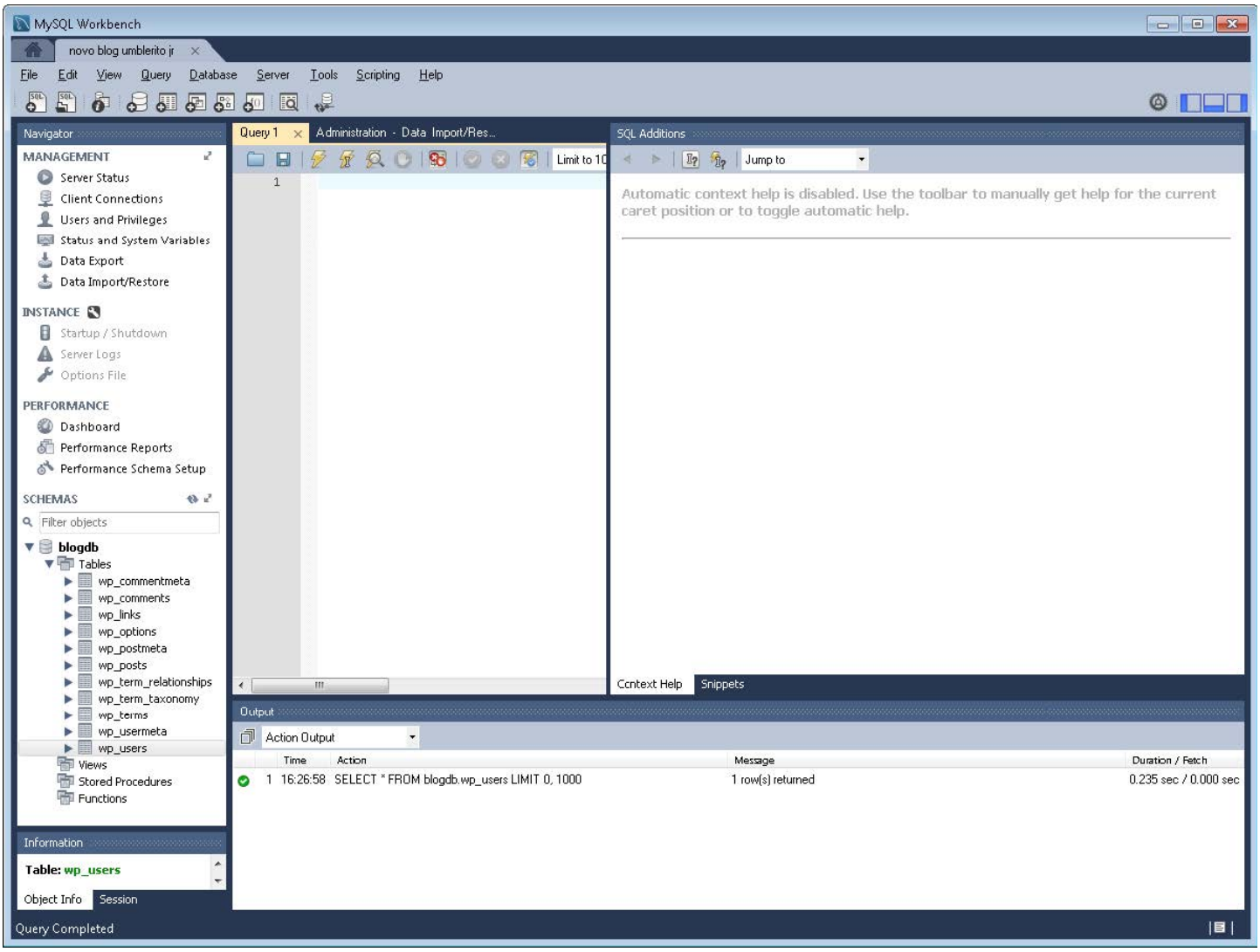
Figura 3 – Data Import



Após os itens serem selecionados, é preciso escolher para qual banco de dados este arquivo será importado em Default Schema to be Imported To. Após a escolha do banco de dados de destino, clicar em Start Import.

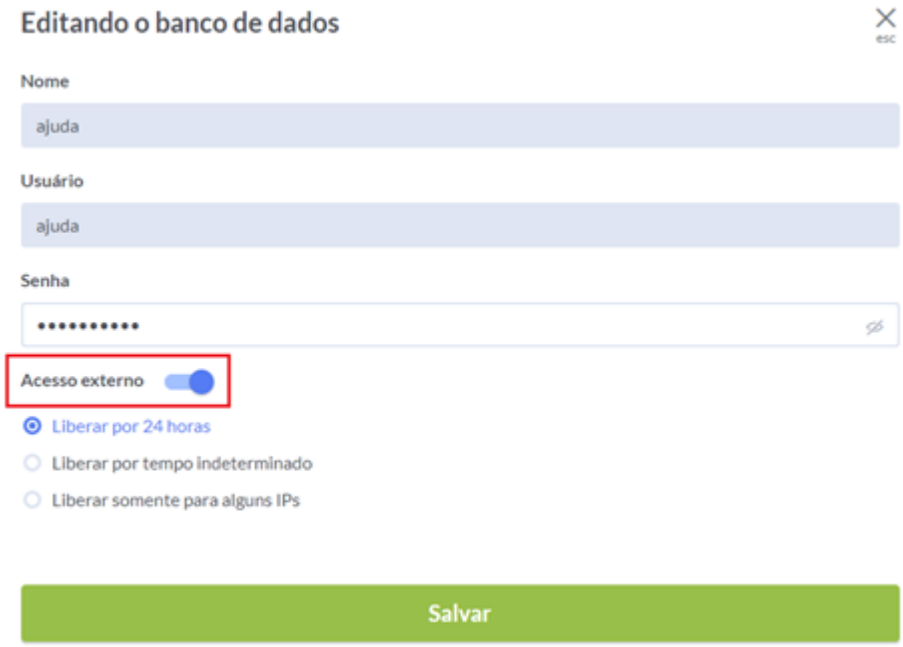
Depois de a importação ser finalizada, será possível ver as tabelas disponíveis clicando em cima do nome do banco de dados.

Figura 4 – Tabelas



Para que a conexão seja realizada com sucesso, é muito importante que o acesso externo ao banco esteja habilitado, conforme pode ser observado na tela a seguir.

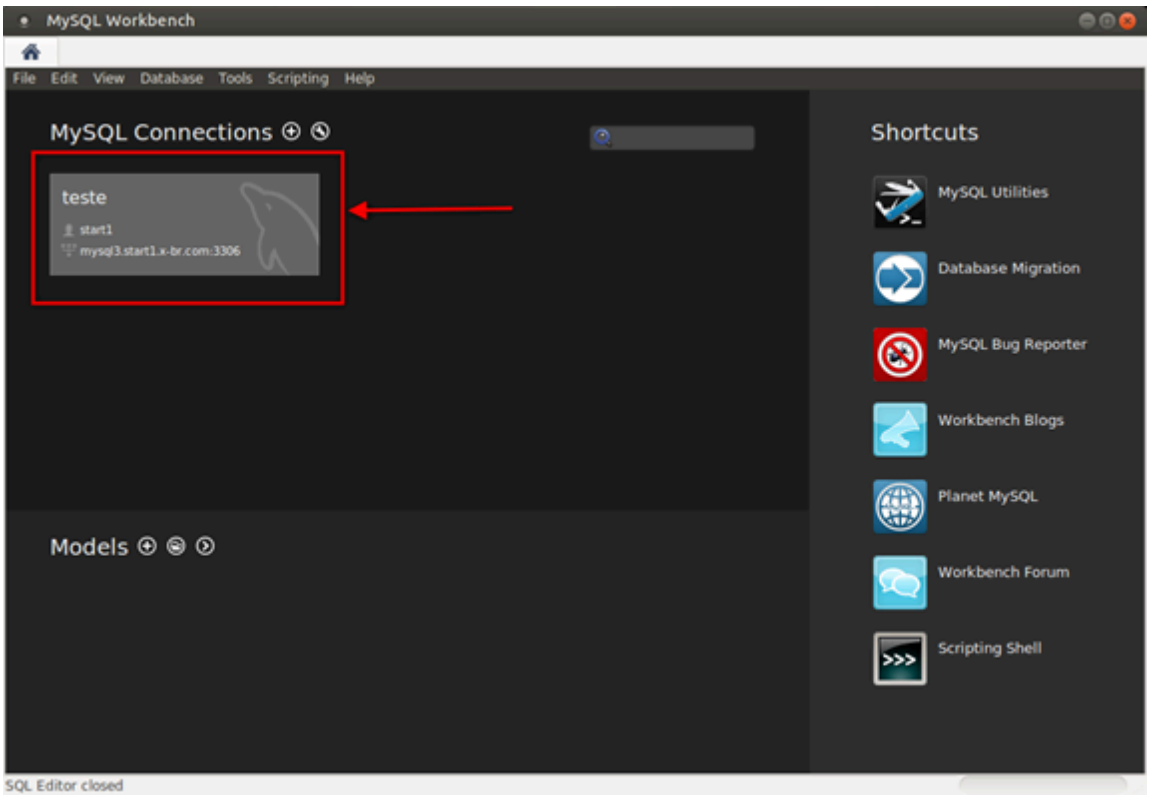
Figura 5 – Liberação de acesso externo



A exportação de dados no MySQL WorkBench serve para fazer um backup do banco de dados, de maneira a garantir a segurança dos dados.

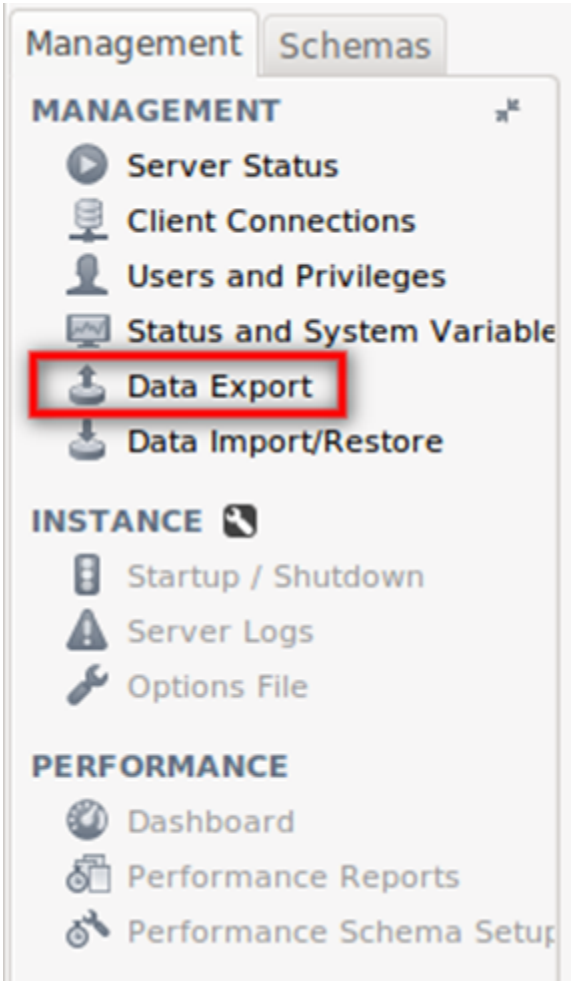
Em primeiro lugar, é preciso acessar o MySQL WorkBench informando usuário e senha e iniciar a conexão com o Banco de Dados.

Figura 6 – Acesso ao MySQL WorkBench



Ao acessar o MySQL WorkBench, abrir a aba Management, do lado esquerdo da tela, e clicar na opção Data Export, conforme pode ser observado na figura a seguir.

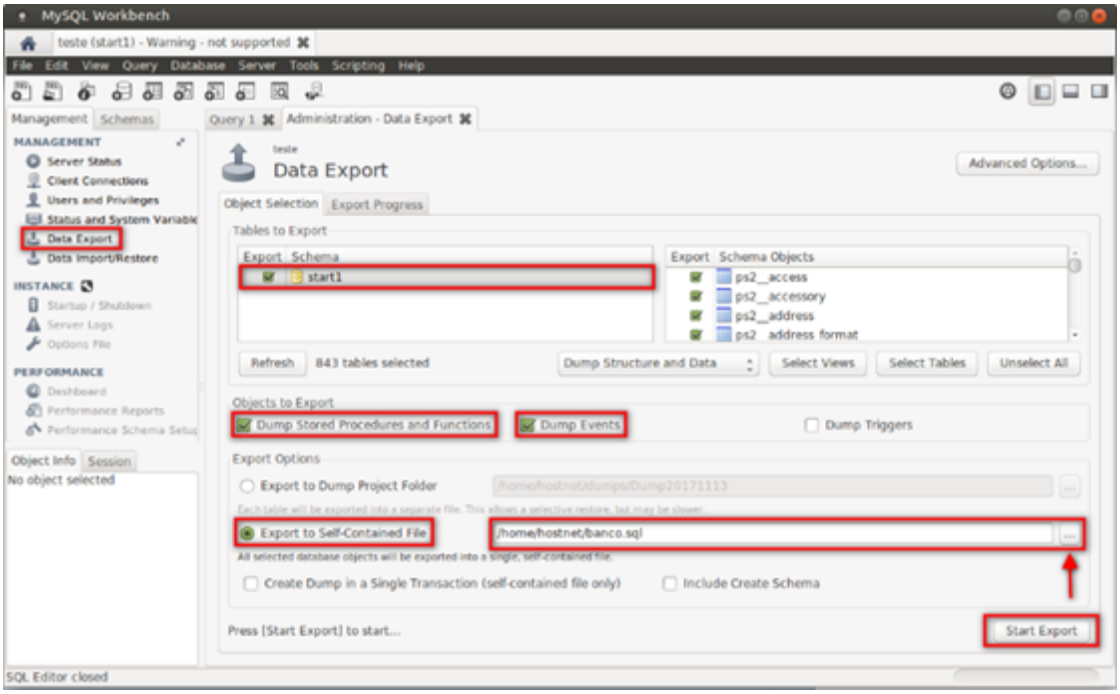
Figura 7 – Exportação de dados



Em seguida, realizar os seguintes procedimentos, que estarão visíveis na próxima tela:

- selecionar o Banco de Dados que deseja exportar;
- marcar a opção Dump Stored Procedures and Functions;
- marcar a opção Dump Events;
- marcar a opção Export to Self-Contained File e, em seguida, selecionar o local em que se deseja salvar o banco de dados clicando no botão dos três pontinhos (um nome para o arquivo do Banco deve ser atribuído nesse momento. Verificar se o caminho e o nome estão corretos no campo da localidade); e
- após selecionar todas as opções citadas, clicar em Start Export.

Figura 8 – Seleção de banco de dados



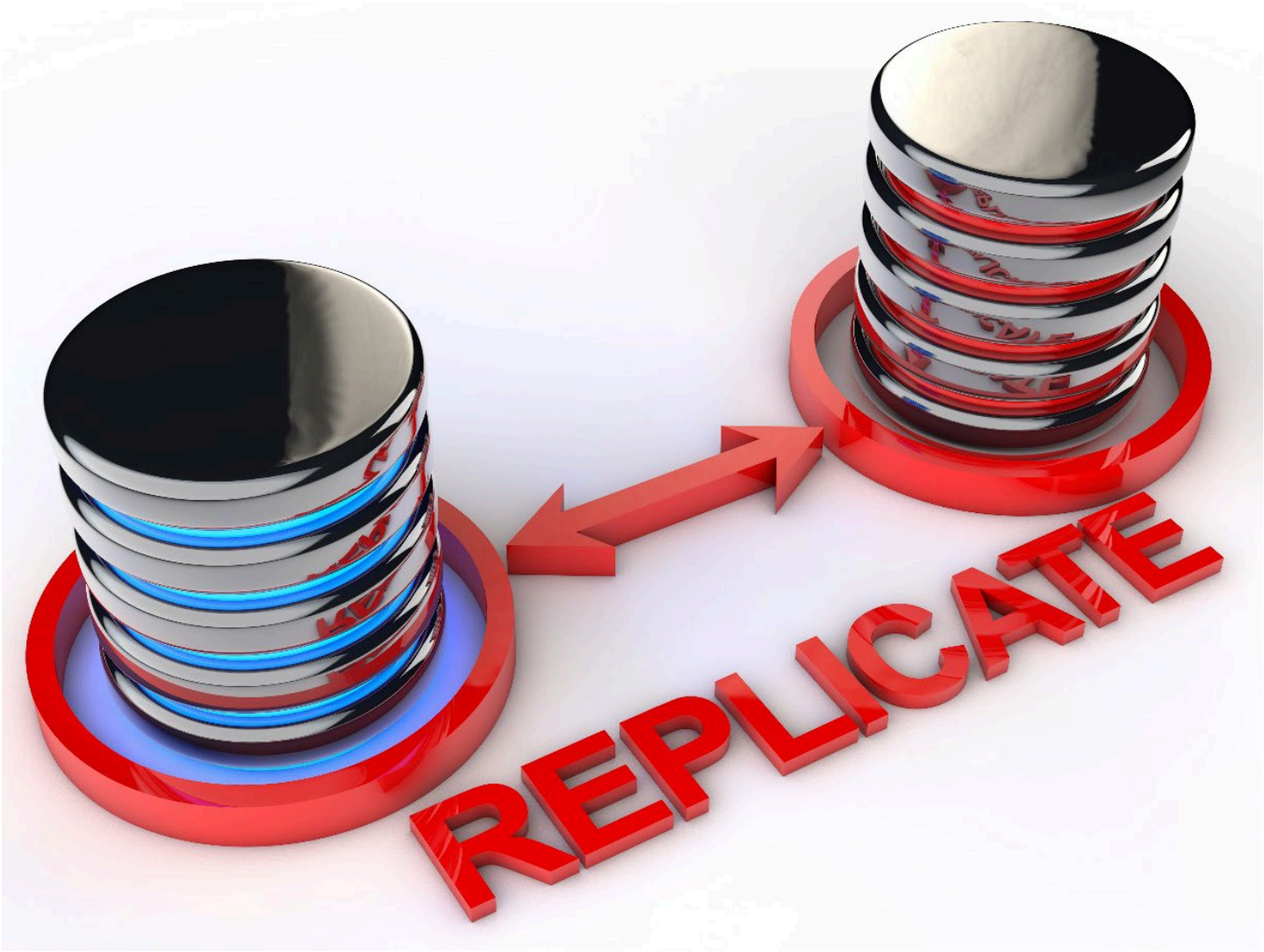
Quando o processo de exportação for finalizado, o arquivo com o nome informado estará salvo no local especificado.

TEMA 3 – REPLICAÇÃO DE DADOS

A replicação de dados tem como principal objetivo transferir para um ou mais bancos de dados o conteúdo administrado por um sistema. Pode apresentar vários benefícios, como ter dados redundantes, que possibilitam alternativas de tolerância a falhas, desde que o sistema alternativo possa ser acessado após algum tipo de desastre com o sistema principal. Dependendo de como a cópia replicada é montada, podemos ter um balanceamento entre os sistemas disponíveis por um balanceamento de acesso. Poderemos ter também um backup dos dados originais, embora essa não seja uma opção muito válida para cópias de segurança. É possível, mas não deve ser a única opção de backup, podendo ser utilizada somente em casos extremos.

O sistema gerenciador de banco de dados MySQL permite a realização de replicação chamada de Master-Slave, em que dois servidores são utilizados para a replicação, sendo um o replicador, identificado como master, e o outro servidor, que recebe os dados, é identificado como slave. Nesse processo, o master gera logs de alteração dos dados que, posteriormente, são replicados para o servidor slave.

Figura 9 – Representação de replicação de dados

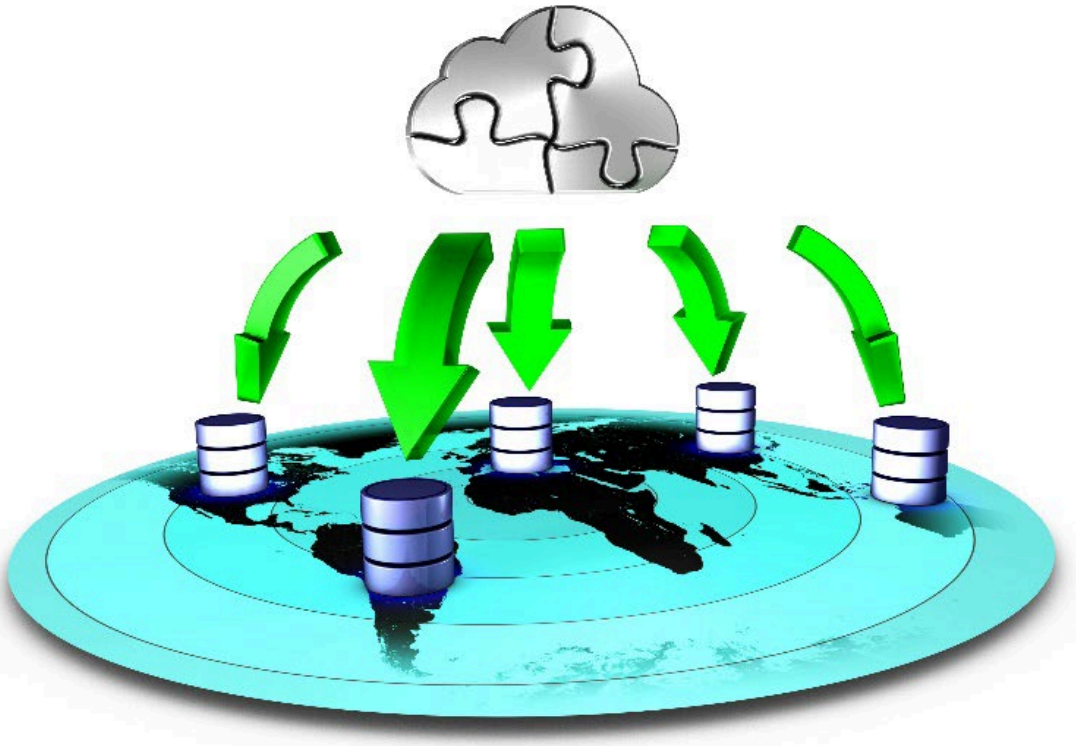


Créditos: Novelo/Shutterstock.

A replicação de uma versão atual do MySQL é compatível com versões anteriores, isto é, um servidor com versão mais nova pode normalmente ser um escravo de um servidor com versão mais antiga. O contrário, porém, não funciona. As versões mais antigas são incapazes de servir como slaves de versões mais atuais, pois as versões mais antigas não conseguem entender novas características ou novas funcionalidades que o servidor mais novo possui, podendo haver diferenças no formato dos arquivos que a replicação utiliza. Por exemplo, não é possível replicar os dados de um master MySQL 5.0 para um slave MySQL 4.0.

A replicação também pode ser realizada entre bases de dados localizadas na nuvem.

Figura 10 – Representação da nuvem



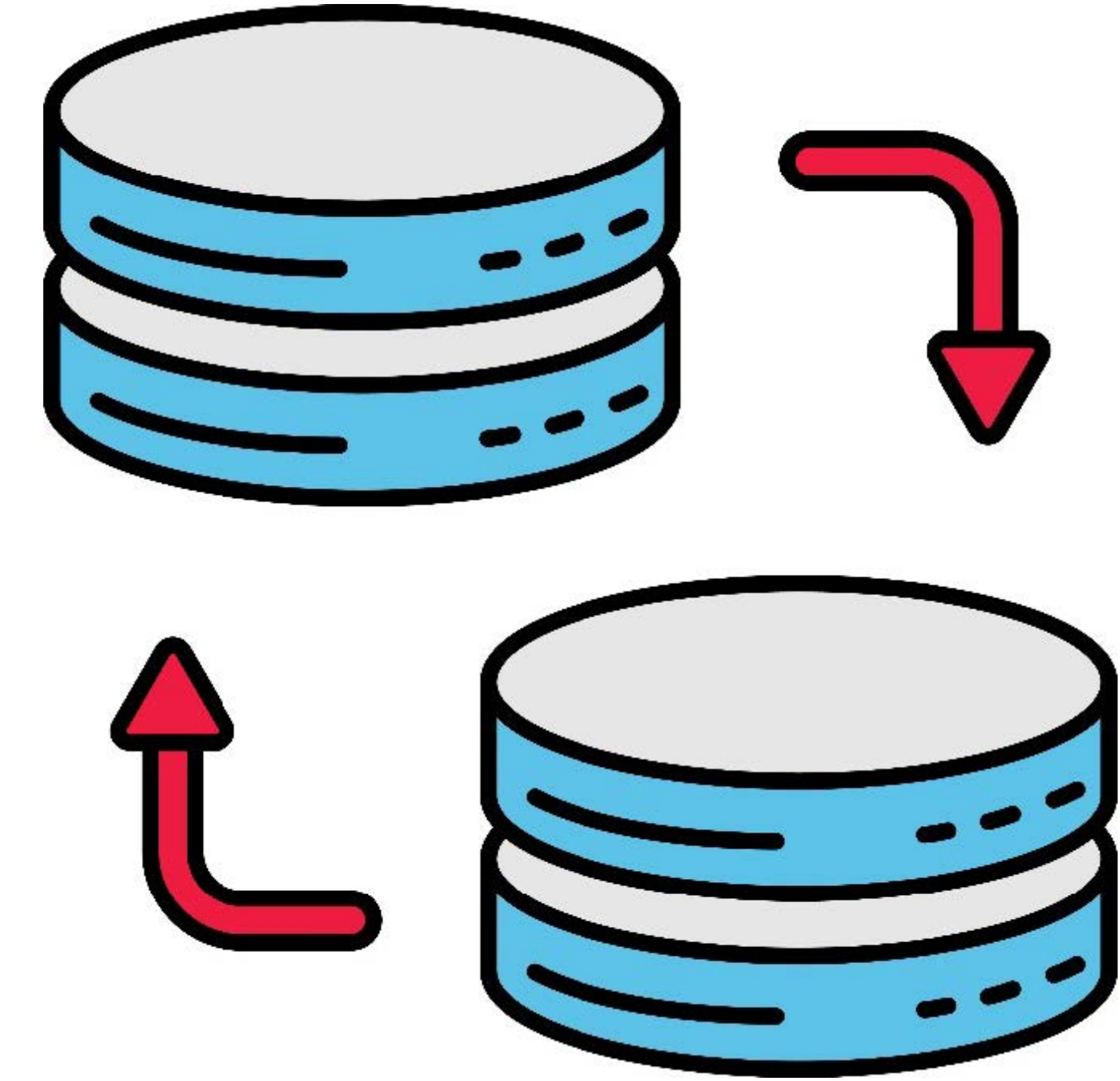
Créditos: Yabresse/Shutterstock.

A replicação apresenta uma série de vantagens, tais como o equilíbrio de carga, pois a replicação não precisa de uma largura de banda muito larga, sendo possível ainda iniciar e parar conforme a necessidade do administrador, além de ser possível para os servidores trabalharem distantes um do outro.

A replicação permite distribuir os dados de um sistema, possibilitando, dessa maneira, que eles sejam acessados nos servidores onde foram distribuídos. Isso é possível porque, quando é feita a replicação, o mesmo dado pode ser encontrado nos bancos de dados que participam do processo.

A replicação de dados pode ser entendida de uma forma bem simples como uma cópia dos dados, total ou parcial:

Figura 11 – Representação de replicação de dados



Créditos: shmai / Shutterstock.

Outra vantagem da replicação é que ela pode ser uma técnica de backup, lembrando, porém, que um slave não é exatamente um backup, e não deve ser um substituto para os backups tradicionais.

Podemos citar também como vantagem da replicação a alta disponibilidade dos dados, já que se um servidor falhar teremos outro para suprir sua função.

O processo de replicação do MySQL é realizado em fases. A primeira registra as alterações realizadas na base de dados em um arquivo de log. Na fase seguinte, os dados são registrados no servidor master. Para finalizar o processo, as transações são efetivadas pelas ferramentas de armazenamento.

Dando continuidade ao processo de replicação, o servidor slave copiará os eventos de log realizados pelo servidor master. Esse processo de gravação é realizado no relay log, conhecido como log de retransmissão. Nesse processo, é realizada uma conexão entre o servidor master e o slave, e o conteúdo do arquivo de log é esvaziado no master e escrito no log de retransmissão do servidor slave, para efetivar a replicação.

Esse processo é repetido até que os dados fiquem atualizados entre os dois servidores que participam da replicação, master e slave.

Figura 12 – MySQL

Créditos: icon99/Shutterstock.

TEMA 4 – MIGRAÇÃO DE DADOS

O processo da migração de dados pode ser muito complexo e, como existem muitas bases de dados diferentes, não existe uma solução que funcione em todas as situações. Além da dificuldade de transferir os dados entre os dois SGBD, também contribuirá muito na complexidade da migração o tipo de dados das tabelas envolvidas no processo: datas e booleanos podem dar problemas ao passar de um SGBD para outro porque podem ser tratados de maneiras diferentes ou, no caso dos números com casas decimais, ocorrem problemas com uma precisão diferente.

A seguir, algumas recomendações para a migração entre SGBDs diferentes:

Se a nossa base de dados origem estiver em Access, a migração será fácil pois o MySQL dispõe de um driver ODBC para sistemas Windows, que permite conectar Access com o próprio MySQL e realizar de maneira fácil a migração.

Devemos informar que queremos fazer uma exportação de um Access local para um MySQL remoto, podendo haver problemas nesse caso, pois não são todos os servidores que permitem conexões remotas com a base de dados. Se não tivermos uma conexão remota com o nosso servidor de bases de dados, vamos ter que mudar de estratégia, instalando MySQL localmente e realizando a migração do Access local para o MySQL local e, depois, fazendo um backup da base de dados local e o restore no MySQL remoto.

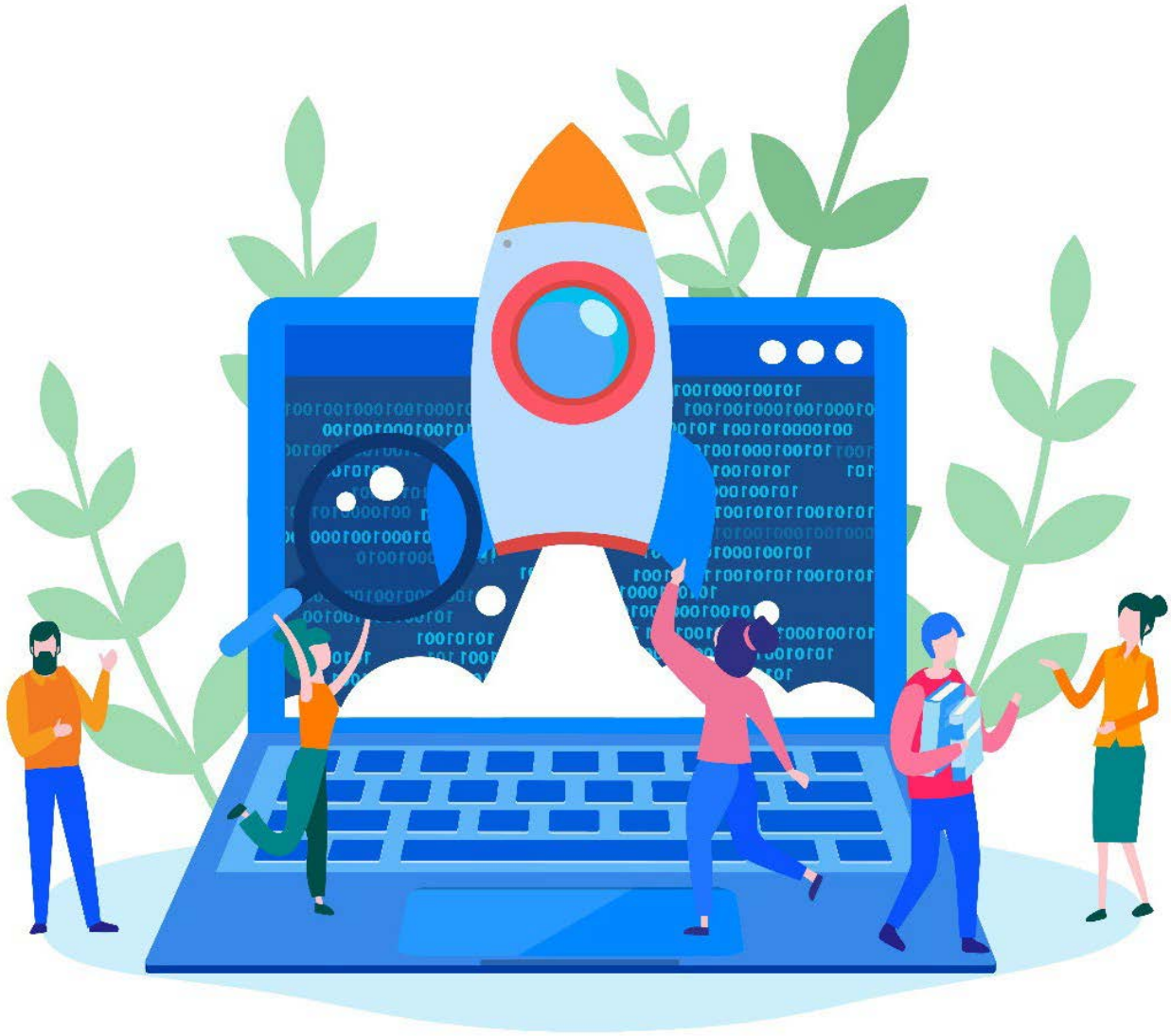
Para migrar dados do SQL Server para o MySQL, podemos utilizar o Access como ponte entre esses dois SGBDs. O Access permite selecionar uma base de dados do SQL Server e, como vimos anteriormente que ele se conecta com o MySQL, será possível migrar os dados entre o SQL Server e o MySQL.

Para outros SGBDs que possuam conexão via driver ODBC, não haverá maiores problemas para conectá-la com o Access, de maneira similar a como se conecta com o MySQL. Podemos, então, utilizar o Access para exportar os dados, por exemplo, para o MySQL.

Se não temos Access, ou a base de dados original não tem driver ODBC, ou se o processo de migração não funcionou corretamente, outra possibilidade é exportar os dados para um arquivo de texto, separados por vírgulas. Muitas bases de dados têm ferramentas para exportar os dados das tabelas para arquivos de texto, os quais podem ser inseridos no MySQL com o auxílio de alguma ferramenta, como phpMyAdmin por exemplo.

No processo de migração de dados, muitas vezes é necessário realizar a criação de scripts que implementem mudanças nos dados que serão migrados, para que os mesmos possam ser adequados aos tipos de dados do servidor de destino. Os dados booleanos, que em alguns SGBDs são armazenados como true e false, devem ser transformados para valores inteiros, 0 e 1, para serem adequados ao MySQL. As datas devem ter uma atenção especial porque podem ter armazenamentos diferentes nos SGBDs. Em alguns, podem aparecer no formato “dd/mm/aaaa” e, no MySQL, são armazenados no formato “aaaa-mm-dd”, portanto, os formatos precisarão ser adequados para que a migração ocorra de maneira correta entre os SGBDs.

Figura 13 – Compatibilização de dados



Créditos: Kachka / Shutterstock.

TEMA 5 – VERIFICAÇÃO DE OCUPAÇÃO DE ESPAÇO EM DISCO

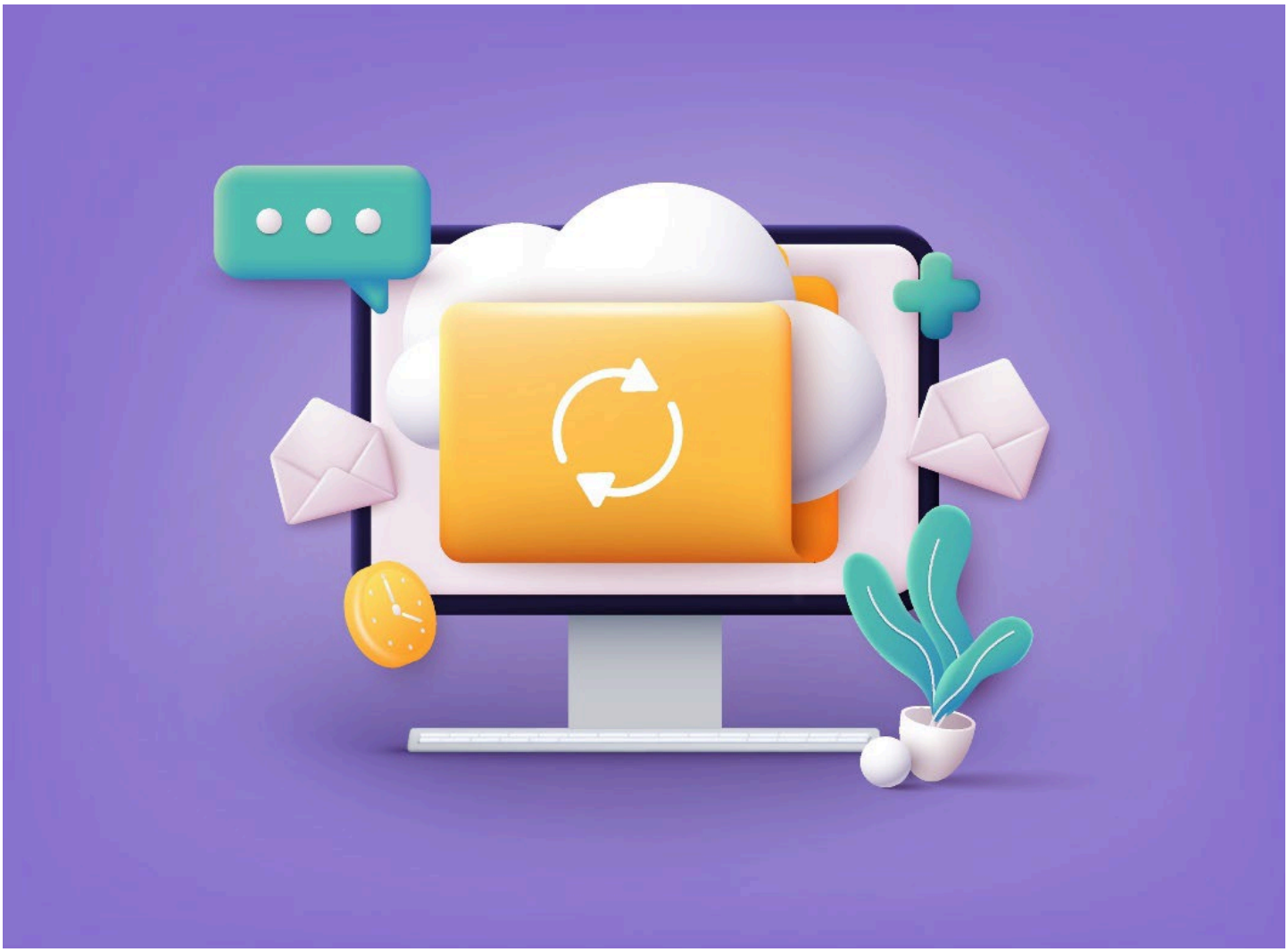
O bom funcionamento de um SGBD depende de muitos fatores. Entre eles, o espaço em disco é um item muito importante. Veremos aqui como o MySQL se comporta com discos sem espaço.

Quando ocorre uma condição de disco sem espaço, o MySQL faz algumas verificações importantes, que veremos a seguir:

- É verificada a cada minuto a existência de espaço suficiente para gravar os dados. Se houver espaço suficiente, ele continua gravando sem interromper o funcionamento;
- De seis em seis minutos, é gravada uma entrada no log de arquivo, avisando sobre a condição de disco cheio;
- Para continuar quando o disco está cheio, é preciso liberar espaço suficiente para inserir todos os registros;
- Para abortar um processo de gravação, é preciso enviar um mysqladmin kill para o procedimento;
- Outro processo pode estar esperando pelas tabelas que provocaram a condição de disco cheio. Se vários processos estão bloqueados, matar o que está esperando pela condição de disco cheio irá permitir que os outros continuem.

Os comandos utilizados para a verificação de espaço em disco geram arquivos de log para registrar os eventos, porém esses arquivos podem ter um tamanho que comprometa o espaço em disco. E se o MySQL, durante a sua operação, ficar sem espaço livre para a execução das operações, o arquivo de log será removido e uma falha acontecerá, com exceção do comando ALTER TABLE, que manterá a estrutura da tabela inalterada.

Figura 14 – Transferência de dados



Créditos: olesia_g/Shutterstock.

Com a utilização do comando TRUNCATE TABLE, é possível recuperar o espaço em disco do sistema operacional. Ao truncar uma InnoDB tabela, a tabela deve ser armazenada em seu próprio arquivo .ibd . Para que uma tabela seja armazenada em seu próprio arquivo .ibd, innodb_file_per_table deverá ser habilitado quando ela for criada. Além da habilitação, não pode haver uma restrição de chave estrangeira entre a tabela que está sendo truncada e outras tabelas, caso contrário, a operação TRUNCATE TABLE não será realizada. Uma restrição de chave estrangeira entre duas colunas na mesma tabela, no entanto, é permitida.

Quando uma tabela é truncada, ela é eliminada e recriada em um novo arquivo .ibd, e o espaço liberado é devolvido ao sistema operacional. Isso contrasta com InnoDB tabelas truncadas que são armazenadas no InnoDB espaço de tabela do sistema, que são tabelas criadas quando innodb_file_per_table = OFF, e tabelas armazenadas em espaços de tabela gerais compartilhados, nos quais somente é possível utilizar o espaço liberado depois que a tabela é truncada no InnoDB.

A capacidade de truncar tabelas e devolver espaço em disco ao sistema operacional também significa que os backups físicos podem ser menores. O truncamento de tabelas armazenadas no tablespace do sistema, que são tabelas criadas quando innodb_file_per_table = OFF, ou em um tablespace geral, deixa blocos de espaço não utilizado no tablespace.

Figura 15 – Representação de gerenciamento de dados

Créditos: FGC / Shutterstock.

Uma das atividades de um DBA é gerenciar a Entrada/Saída (E/S) do disco para impedir que o subsistema de E/S fique saturado, e também gerenciar o espaço em disco para evitar o preenchimento dos dispositivos de armazenamento. O modelo de design ACID requer uma certa quantidade de E/S que pode parecer redundante, mas ajuda a garantir a confiabilidade dos dados. Dentro dessas restrições, InnoDB tenta otimizar o trabalho do banco de dados e a organização dos arquivos do disco para minimizar a quantidade de E/S do disco. Às vezes, a E/S é adiada até que o banco de dados não esteja ocupado ou até que tudo precise ser colocado em um estado consistente, como durante a reinicialização do banco de dados após um desligamento.

A seguir, podemos verificar as principais considerações para E/S e espaço em disco com o tipo padrão de tabelas MySQL, que também são conhecidas como InnoDB tabelas:

- controlar a quantidade de E/S em segundo plano usada para melhorar o desempenho da consulta;
- ativar ou desativar recursos que fornecem durabilidade extra às custas de E/S adicional;
- organizar tabelas em muitos arquivos pequenos, alguns arquivos maiores ou uma combinação de ambos;
- equilibrar o tamanho dos arquivos de redo log em relação à atividade de E/S que ocorre quando os arquivos de log ficam cheios; e
- como reorganizar uma tabela para otimizar o desempenho da consulta.

FINALIZANDO

Tivemos contato com os cuidados que devemos ter com um sistema gerenciador de banco de dados e o conteúdo por ele administrado. Vimos como é importante a manutenção de um banco para manter os dados em boas condições, permitindo dessa maneira um acesso eficaz ao banco de dados. Outro ponto importante que abordamos foi a importação e exportação de dados tanto para outros bancos de dados de um mesmo SGBD quanto para outros gerenciadores e sistemas.

Tratamos da replicação de dados, que é uma ferramenta importante no gerenciamento dos dados de um sistema. Vimos que a migração de dados possibilita realizar a transferência de uma base de dados para versões atualizadas de um SGBD ou de outros sistemas. Para finalizar, conhecemos as rotinas de limpeza e ocupação de disco, que devem receber uma atenção especial, pois é nesse ambiente que o SGBD funciona.

REFERÊNCIAS

BARRIE, H. **Dominando Firebird**: uma referência para desenvolvedores de bancos de dados.Ciência Moderna, 2006.

CARLOS, S. **Comparativo de desempenho de Bancos de dados de Código Aberto**. (UFPE), 2011.

CARNEIRO, A. **Técnicas de otimização de bancos de dados, um estudo comparativo**: MySQL e PostgreSQL. (FURG), 2011.

LEITE, M., **Acessando bancos de dados com ferramentas RAD**. Braspor,2007.

SILBERSCHATZ, A.; KORT; SUDARSHAM. **Sistemas de bancos de dados** . 5 ed. 2006.