



```
elif operation == "MIRROR_Y":  
    mirror_mod.use_x = False  
    mirror_mod.use_y = True  
    mirror_mod.use_z = False  
elif operation == "MIRROR_Z":  
    mirror_mod.use_x = False  
    mirror_mod.use_y = False  
    mirror_mod.use_z = True  
  
#selection at the end -add back the deselected mirror modifier  
mirror_ob.select= 1  
modifier_ob.select=1  
bpy.context.scene.objects.active = modifier_ob  
print("Selected" + str(modifier_ob)) # modifier ob is the active  
#selected object  
#selected object = 0  
#selected object = 1
```

Gerenciamento e Desenvolvimento em Banco de Dados

Gerenciamento e Desenvolvimento em Banco de Dados

Plínio Tavares Florentino

© 2018 por Editora e Distribuidora Educacional S.A.

Todos os direitos reservados. Nenhuma parte desta publicação poderá ser reproduzida ou transmitida de qualquer modo ou por qualquer outro meio, eletrônico ou mecânico, incluindo fotocópia, gravação ou qualquer outro tipo de sistema de armazenamento e transmissão de informação, sem prévia autorização, por escrito, da Editora e Distribuidora Educacional S.A.

Presidente

Rodrigo Galindo

Vice-Presidente Acadêmico de Graduação e de Educação Básica

Mário Ghio Júnior

Conselho Acadêmico

Ana Lucia Jankovic Barduchi

Camila Cardoso Rotella

Danielly Nunes Andrade Noé

Grasiele Aparecida Lourenço

Isabel Cristina Chagas Barbin

Lidiane Cristina Vivaldini Olo

Thatiane Cristina dos Santos de Carvalho Ribeiro

Revisão Técnica

Marcio Aparecido Artero

Ruy Flávio de Oliveira

Editorial

Camila Cardoso Rotella (Diretora)

Lidiane Cristina Vivaldini Olo (Gerente)

Elmir Carvalho da Silva (Coordenador)

Leticia Bento Pieroni (Coordenadora)

Renata Jéssica Galdino (Coordenadora)

Dados Internacionais de Catalogação na Publicação (CIP)

F633g Florentino, Plínio Tavares
Gerenciamento e desenvolvimento em banco de dados /
Plínio Tavares Florentino. – Londrina : Editora e Distribuidora
Educacional S.A., 2018.
192 p.

ISBN 978-85-522-1132-7

1. Análise de requisitos. 2. Modelagem de dados. 3.
Integração e implementação de banco de dados. 4. SQL.
5. Oracle. 6. MySQL. I. Florentino, Plínio Tavares. II. Título.

CDD 005.74

Thamiris Mantovani CRB-8/9491

2018

Editora e Distribuidora Educacional S.A.

Avenida Paris, 675 – Parque Residencial João Piza

CEP: 86041-100 – Londrina – PR

e-mail: editora.educacional@kroton.com.br

Homepage: <http://www.kroton.com.br/>

Sumário

Unidade 1 Administração de banco de dados	7
Seção 1.1 - Introdução à administração de banco de dados	9
Seção 1.2 - Introdução às operações em banco de dados	22
Seção 1.3 - Desempenho do banco de dados	37
Unidade 2 Segurança de banco de dados	53
Seção 2.1 - Introdução à segurança em banco de dados	55
Seção 2.2 - Privilégios de acesso	67
Seção 2.3 - Gerenciamento de privilégios	80
Unidade 3 Recursos avançados em banco de dados	95
Seção 3.1 - Introdução a múltiplas instâncias	97
Seção 3.2 - Recuperação de dados	108
Seção 3.3 - Banco de dados na nuvem	123
Unidade 4 Fundamentos de banco de dados não-convencionais	141
Seção 4.1 - Introdução a bancos de dados geográficos	143
Seção 4.2 - A XML e seu armazenamento em bancos de dados	158
Seção 4.3 - Introdução ao NoSQL	174

Palavras do autor

Caro aluno, o objetivo desta obra é que você conheça, compreenda e saiba como gerenciar com segurança bancos de dados convencionais e não-convencionais.

Os bancos de dados são componentes essenciais da sociedade moderna. Há bancos de dados envolvidos em inúmeras atividades em nosso dia-a-dia, por exemplo, quando vamos depositar ou retirar dinheiro, quando compramos um produto em um site ou até mesmo quando somos registrados ao nascermos.

Neste livro, vamos aprender como instalar um banco de dados e testar suas funcionalidades, compreendendo o conceito e a estrutura do banco de dados e sabendo monitorar e verificar a performance. Vamos criar uma estrutura de banco de dados e entender suas regras de criação e manipulação, tanto de arquivos quanto estruturas, bem como criar *logins*, sabendo da importância de criar um usuário e manipular os privilégios de acesso. Entenderemos como realizar um backup e um *restore* do banco de dados e sua importância, e como funciona um banco de dados na nuvem. Abordaremos um pouco do XML e os benefícios de sua utilização, e, por último, vamos entender como funciona um banco de dados não-relacional.

Na Unidade 1, você vai aprender o conceito do banco de dados, além entender como ele funciona, sua instalação, seus testes de estrutura e o monitoramento de sua performance. Também vai aprender como manipular informações dentro do banco de dados, para usá-lo de forma adequada.

Na Unidade 2, você verá assuntos relacionados à segurança do banco de dados, tais como criar privilégios de acesso e gerenciar os mesmos.

Na Unidade 3, você conhecerá alguns recursos avançados do banco de dados, como gerenciamento de múltiplas instâncias, backup, *restore* e replicação do banco de dados, e entenderá o banco de dados na nuvem.

Na Unidade 4, você aprenderá sobre banco de dados não relacionais e entenderá como funciona um banco de dados geográfico e o XML.

É importante que você passe por todas as unidades para fixar o conteúdo e compreender suas dificuldades, mas lembre-se sempre: o aprendizado requer prática.

Bons estudos!

Administração de banco de dados

Convite ao estudo

Aqui iniciamos a primeira unidade da nossa disciplina e você aprenderá o que é um sistema gerenciador de banco de dados (SGBD) e diversos conceitos, como a instalação e a configuração de um SGBD.

Você pode achar que o estudo de SGBD requer o conhecimento de inúmeros conceitos, mas não se preocupe, porque são de fácil entendimento.

Nesta unidade de ensino, vamos dialogar sobre um CEO de uma renomada empresa de *fast food* que está insatisfeito com seu sistema desktop, pois não consegue visualizar todas as informações necessárias para tomar decisões, nem cruzar as informações com as das outras filiais por que o sistema não acessa a internet. Você foi contratado para criar um sistema que permita o acesso por meio de *apps* e da web e tenha acesso às informações das demais filiais, de forma a possibilitar o cruzamento de informações.

Atualmente, esse é um cenário muito comum nas empresas e precisamos estar preparados para esse tipo de situação. Com a facilidade da internet, cada vez mais as empresas buscam ferramentas que possam facilitar seu trabalho e um SGBD muito bem elaborado pode ajudar muito uma empresa em seu cotidiano.

Na Seção 1.1, você aprenderá o conceito do banco de dados, onde ele surgiu e qual é sua importância nas empresas e na nossa vida. Além disso, você criará um banco de dados e realizará testes de um SGBD

Na Seção 1.2, você verá como criar o arquivo de dados de um SGBD, suas regras de identificadores e como manipular e inserir dados.

Na Seção 1.3, verá como é realizado um monitoramento no banco de dados, verificando sua performance constantemente, e entenderá como gerenciar o serviço de banco de dados em uma plataforma Windows.

Em resumo, com o seu empenho nos estudos e sua participação em aula, esse módulo vai ser bastante produtivo e abrirá muitas portas no âmbito profissional.

Pronto para esse desafio?

Então, mãos à obra!

Seção 1.1

Introdução à administração de banco de dados

Diálogo aberto

Caro aluno, vamos dar início a essa unidade curricular? Esperamos que possamos aprender juntos sobre como administrar um banco de dados. Nesta seção, você irá compreender os conceitos de um banco de dados relacional e sua estrutura.

Em nosso contexto de aprendizagem, você foi contratado para criar um novo sistema que tenha acesso às informações de todas as filiais da rede de *fast food*. Neste primeiro momento você precisa apresentar para o CEO, por meio de informações detalhadas em um relatório, testes do sistema gerenciador de banco de dados e logs de monitoramento de performance.

Com essas informações, ele decidirá qual banco de dados será utilizado, mas ele é uma pessoa extremamente econômica em relação a custos. Como você convenceria ele a usar este banco, caso necessite adquirir uma licença de uso ou trocar o servidor por um melhor para que o banco de dados funcione normalmente? Caso o banco de dados precise de uma licença de uso, quais vantagens ele apresenta em relação a um banco de dados de uso livre? É muito importante lembrar que o sistema será acessado por qualquer filial por meio da web e *apps*.

Para auxiliá-lo nesse desafio, vamos estudar e conhecer um banco de dados, seus conceitos e suas características. Vamos dar o primeiro passo no entendimento para que você possa resolver esta situação problema.

Bons estudos!

Não pode faltar

Introdução ao sistema gerenciador de banco de dados relacional

Os sistemas de gerenciamento de banco de dados surgiram no início da década de 1970 com o objetivo de facilitar a programação

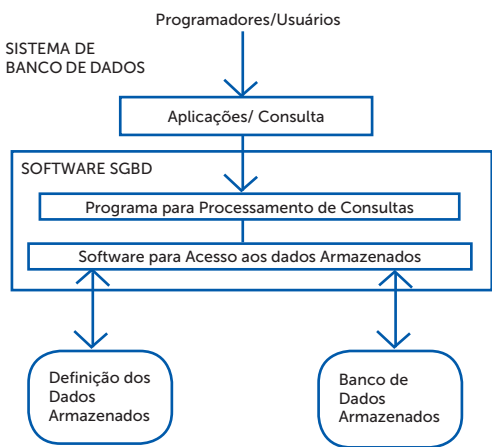
de aplicações de banco de dados. Os primeiros sistemas eram caros e difíceis de usar, requerendo especialistas para usar o sistema gerenciador de banco de dados específico (HEUSER, 2009, p. 8).

Um banco de dados é uma coleção de dados relacionados. Os dados são fatos que podem ser gravados e que possuem um significado implícito. Por exemplo, considere nomes, número telefônicos e endereços de pessoas; esses dados podem ter sido salvos em uma agenda de telefone ou armazenados em algum computador por meio de um software.

Um banco de dados representa alguns aspectos do mundo real, sendo chamado, às vezes, de minimundo ou de universo de discurso. As mudanças no minimundo são refletidas em um banco de dados. É uma coleção lógica e coerente de dados com algum significado inerente, como mostra a Figura 1.1, em que podemos entender como é o funcionamento de uma configuração de um sistema de banco de dados.

Em outras palavras, um banco de dados possui algumas fontes das quais os dados são derivados, bem como alguns níveis de interação com eventos do mundo real interessados em seus conteúdos, e pode ser de qualquer tamanho ou de complexidade variável, podendo possuir algumas ou centenas de registros, cada um com uma estrutura simples (ELMASRI, 2005).

Figura 1.1 | Configuração de um sistema de banco de dados simplificado



Fonte: elaborada pelo autor.

Para definirmos um banco de dados, devemos especificar a estrutura de cada registro em cada arquivo, considerando os diferentes tipos de elementos de dados a serem armazenados em cada registro.

Um banco de dados precisa ser eficiente, ou seja, possibilitar o acesso às informações corretas no tempo adequado, de preferência imediatamente ou dentro de poucos segundos. Para ser eficiente, o banco de dados deve observar os seguintes princípios (MEDEIROS, 2013, p. 15):

- Redundância – se dá quando os dados são salvos de forma duplicada. Por exemplo, quando uma empresa com dois departamentos armazenando as mesmas informações sobre uma atividade separadamente, gerando custo de armazenamento.
- Inconsistência – ocorre quando os dados não estão sendo atualizados e, quando um usuário tenta acessar as informações, só aparecem informações antigas ou não aparece nada.
- Integração – os dados da empresa são compartilhados entre várias pessoas ou departamentos, portanto há uma necessidade de eles serem integrados para que todos tenham acesso às informações.



Pesquise mais

Para entender um pouco mais sobre os conceitos fundamentais de banco de dados de acesso, comece pelo artigo disponível em:

<<https://www.devmedia.com.br/conceitos-fundamentais-de-banco-de-dados/1649>> . Acesso em: 10 de mar. 2018.

Um número significativo de características distingue a abordagem que utiliza o banco de dados da tradicional, que usa a programação de arquivos.

Uma característica fundamental de um banco de dados é que seu o possui não apenas o banco de dados, mas uma complexa definição da sua estrutura e de suas restrições. Essa definição está armazenada no catálogo do SGBD, que contém informações com a estrutura de cada arquivo, o tipo e o formato de armazenamento de cada item.

Um banco de dados tipicamente tem muitos usuários, e cada qual pode solicitar diferentes perspectivas ou visões do banco. Uma visão pode ser um subconjunto de um banco de dados ou conter uma visão virtual dos dados derivados dos arquivos do banco de dados, mas não explicitamente armazenados (ELMASRI, 2005, p. 7).



Assimile

As principais propriedades de um banco de dados são:

- Representar algum aspecto do mundo real, o que chamamos de minimundo (ou universo de discurso).
- Ser uma coleção logicamente coerente de dados com algum significado inerente, e não uma variedade aleatória de dados.
- Ter uma finalidade específica.
- Possuir um grupo definido de usuários.
- Possuir aplicações previamente concebidas, de interesse dos usuários.

Funções do sistema gerenciador de banco de dados

Um SGBD é composto por uma coleção de programas que permitem que o usuário crie ou altere um banco de dados. Em geral ele facilita o processo de definição, construção, manipulação e compartilhamento entre usuários e aplicações.

Os quatro itens citados (definição, construção, manipulação e compartilhamento) tem significados específicos para o um sistema gerenciador de banco de dados. São eles (VICCI, 2009):

- Definição: a definição de um banco de dados envolve especificar os tipos, as estruturas e as restrições dos dados a serem armazenados. A definição ou informação descritiva também é armazenada pelo sistema gerenciador de banco de dados na forma de catálogo ou dicionário, e são chamadas de metadados.
- Construção: é o processo de armazenamento dos dados em algum meio controlado pelo SGBD.

- Manipulação: esse processo inclui funções como consulta ao banco de dados para recuperar dados específicos, atualização para refletir mudanças e geração de relatórios com base nos dados.
- Compartilhamento: é o processo que permite que o banco de dados seja acessado simultaneamente por diversos usuários e programas.

Outras funções de um sistema de banco de dados são: armazenar e restaurar dados, administrar os metadados, controlar e limitar dados redundantes em múltiplos sistemas, garantir a aplicação das regras de negócio, fornecer serviços de autorização de segurança e de backup de dados, e proporcionar atomicidade de transação. Um SGBD fornece diversas funções que os programadores teriam que desenvolver com seus próprios recursos, aumentando muito a produtividade do programador. Por exemplo, o programador não terá que se preocupar com indexação de dados para recuperação rápida, ou em executar validações nos dados adicionados ao banco de dados. As regras dentro do banco de dados executarão a checagem necessária e retornarão um código de erro.

Como o SGBD permite aos usuários recuperar dados facilmente, sem qualquer programação externa, as aplicações de negócios não têm que antecipar cada uso possível dos dados que elas coletam. Os programadores desenvolvem a maioria dos relatórios úteis diretamente na interface de usuário do aplicativo, mas os administradores podem analisar os dados de muitas outras formas, usando diretamente o SGBD e, como resultado, os programas são menores e mais fáceis de administrar.



Refleta

O que aconteceria se um banco não usasse um banco de dados para cadastrar seus clientes? Provavelmente, haveria muitos arquivos locais nos bancos só para registro de cliente e, dificilmente, seria possível ter informações de um mesmo cliente em uma agência de outra região.

Outras funções importantes de um sistema gerenciador de banco de dados são:

- Em alguns tipos de sistemas de banco de dados, como orientado a objeto e objeto-relacional, os usuários podem estabelecer as operações sobre os dados como parte das definições de dados (ELMASRI, 2005).
- Uma operação ou método é especificada em duas partes: a interface ou assinatura de uma operação, incluindo o nome da operação e os tipos de dados de seus argumentos (ou parâmetros), e a implementação (ou método) de uma operação que é definida separadamente e pode ser alterada sem afetar a interface. Os programas de usuários da aplicação podem operar nos dados, invocando essas operações por meio de seus nomes e argumentos, sem considerar como essas operações são implementadas (CLAUDIA, 2009).



Pesquise mais

Para conhecer melhor a instalação de um SGBD, use o MySQL, que é um dos bancos de dados *Open Source* mais usados. Existe a versão *Community* do MySQL, própria para desenvolvedores, com a qual você conseguirá progredir em seus estudos.

O link a seguir explica de uma forma simples como instalar o MySQL.

COMPUTER STUDY. **How to instal MySQL Workbench.** 4 mar. 2018. Disponível em: <<https://www.youtube.com/watch?v=TNoCNSDRg1k>>. Acesso em: 3 de abr. 2018.

Teste do sistema gerenciador de banco de dados

Um dos maiores problemas que enfrentamos em um banco de dados se refere ao seu desempenho: a manipulação de informações ou pesquisas feitas a partir de um sistema legado; a implantação de um outro sistema; ou o aumento inesperado de usuários acessando simultaneamente um banco de dados são fatores que podem comprometer o desempenho de um SGBD. Não é raro se deparar com situações em que uma aplicação para de responder após uma movimentação no banco de dados. Em geral, esse problema

ocorre quando os testes feitos não consideram o uso intensivo por vários usuários simultaneamente. Em relação à inclusão de um novo sistema, é normal que o administrador de banco de dados seja questionado quanto à capacidade do banco de dados suportar novas aplicações. Em alguns casos, o problema não é aparente, até que o volume inesperado de demanda ocorra.

Atualmente, é possível encontrar alguns softwares que de alguma forma, auxiliam o administrador de dados em testes de capacitação. Alguns exemplos são: Quest Software, BMC Software, Unicener NewMics, Capacity Planner Option, OpenSTA e Apache. Também é bom ressaltar que as ferramentas citadas não têm as mesmas funcionalidades e, entre as empresas de softwares citadas acima, vamos falar sobre o JMeter da Apache para uso de teste de serviço de banco de dados.

O JMeter é um software *Open Source* desenvolvido na linguagem Java, projetado para realizar teste de cargas de aplicações Cliente/Servidor. Inicialmente, foi projetado para realizar testes em aplicações web, mas logo foi expandido para realizar outros tipos de testes. Com ele, é possível assegurar que um SGBD esteja capacitado para suportar um grande volume de acessos simultâneos, bem como seu comportamento, no que tange à escalabilidade. Entende-se a escalabilidade como o tempo de resposta do sistema em relação à demanda de recursos exigidos dele. Aumentar a escalabilidade de um banco de dados consiste em expandir sua capacidade de processamento e armazenamento de registros, seja melhorando a capacidade do hardware ou otimizando recursos do sistema (DEVMEDIA, 2018).

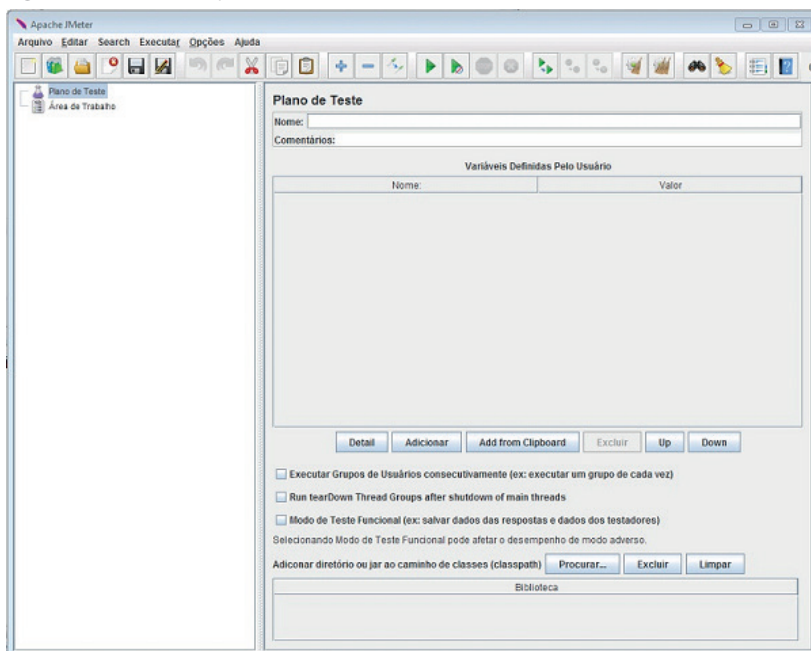
Para realizar testes de carga, o JMeters disponibiliza diversos tipos de asserções e requisições, além de controladores lógicos, como loops e controles para serem usados na construção de planos de teste.

A Figura 1.2 apresenta a tela inicial do JMeter, em que são realizados os testes de banco de dados. Como mostrado, ela tem uma interface limpa e bem intuitiva, facilitando o desenvolvimento de testes.

Para usarmos qualquer plano de teste, é necessário criar um plano de teste, que inclui os elementos do teste. Esses elementos podem ser:

- *Thread Group* – é o principal elemento. Todos os outros elementos do plano de teste devem estar sob este. Ele controla os *threads* que serão executados.
- *Listeners* – são elementos que fornecem o acesso as informações obtidas durante o teste.
- *Controllers* – são divididos em dois grupos: os *Samplers*, que são controladores pré-definidos para requisições específicas, podendo ser customizados com inclusão de configurações e de asserções; e os *Logic Controllers*, que são mais genéricos que o *Samplers*, podendo ser customizados com a inserção de outros controladores, configurações e asserções.
- *Timers* – por padrão, as requisições não apresentam pausa entre elas, então são utilizados *timers* para incluir pausa entre elas.
- *Asserções* – são usadas para validar uma resposta obtida na requisição e resposta a esperada. Podem ser usadas expressões regulares na comparação.
- *Configuration Elements* – embora o banco de dados não faça requisições, esses elementos podem criar ou alterar requisições.
- *Pre-Processor Elements* – são responsáveis por executar uma ação antes de fazer uma requisição, e são usados antes das configurações de requisições.
- *Post-Processor Elements* – são responsáveis por executar uma ação depois de fazer uma requisição e usados para processar retorno de requisição.

Figura 1.2 | Tela Principal do JMeter



Fonte: captura de tela do JMeter.

Com essa ferramenta podemos realizar diversos testes no nosso servidor de dados.

Assista ao vídeo *JMeter – Teste de performance em banco de dados* (LIMA, 2017), no *YouTube*, e veja como instalar o JMeter e usá-lo em um banco de dados MySQL de forma adequada.



Exemplificando

Vamos supor que um banco abra uma nova filial em sua cidade e essa filial terá acesso ao mesmo banco de dados, porém, mais usuários se conectarão a ele. Nesse caso, é muito útil realizar um teste de carga para verificar quantos acessos simultâneos o banco de dados aceita.

Sem medo de errar

Conforme abordamos no início desta seção, nosso desafio é convencer o CEO de uma rede de *fast food* a usar um determinado banco de dados, que será acessado por todas as filiais.

Uma boa dica inicial é usarmos um banco de dados com licença *Open Source*, por causa do custo benefício. Além disso, conforme dito na situação problema, o CEO precisa de um banco com acesso por meio da Web e de Apps, então a segunda dica é buscar um banco que atenda a esta demanda e aos seguinte itens:

- Integridade do banco de dados.
- Segurança.
- Compartilhamento de informações.
- Redundância.

Após a escolha do banco, você precisa avaliar a estrutura do servidor. Cada banco tem uma configuração mínima e um sistema operacional adequado para a instalação. Também é preciso verificar qual é o tamanho mínimo do banco instalado, sem informações, para que se tenha noção do tamanho que o banco pode atingir futuramente. Com isso, você pode realizar a instalação no servidor e depois realizar o teste de performance.

Realize o teste de performance com um número de acessos acima do normal esperado para as filiais, para que o banco de dados não use 100% do servidor a todo momento.

Também realize um backup e, depois, tente realizar um *restore* no banco para validar se as informações foram salvas corretamente.

Após essas etapas, apresente ao CEO o relatório de performance do banco de dados e todas as informações levantadas a cima.

Divisão de pessoas envolvidas no banco de dados

Descrição da situação-problema

Em uma empresa, pessoas envolvidas em um sistema de banco de dados podem ser divididas em dois grupos: administradores e usuários. Essas pessoas estão envolvidas no projeto, uso e manutenção do banco de dados e você, que faz parte do grupo de administradores, precisa realizar manutenção preventiva no SGBD.

Resolução da situação-problema

Para realizar a manutenção preventiva, é preciso primeiramente realizar o backup do banco diariamente. Algumas empresas optam por realizar o backup por período, assim a chance de perder informações é reduzida.

Depois de verificar se os backups estão sendo realizados corretamente, é preciso verificar no log do banco de dados se existe algum erro. Caso nada seja apontado, você só precisa monitorar o acesso, verificando se o uso do servidor está saudável ou não. Também é recomendável que possa realizar testes de *restore* no banco de dados, ao menos mensalmente, para verificar se as informações salvas estão seguras. Não é necessário, nem recomendado, realizar o *restore* no mesmo banco. Crie um banco de testes para que possa realizar o *restore*.

Faça valer a pena

1. Garantir a confidencialidade, segurança e integridade de um banco de dados é uma tarefa constante, e é responsabilidade do administrador de dados impor políticas de segurança. Ele é responsável pela autorização de acesso ao banco, bem como pela coordenação e monitoração dos recursos do banco de dados.

Vamos verificar as seguintes afirmações:

- I. A integridade e segurança de um banco de dados não depende somente do seu administrador.
- II. As políticas de segurança devem ser aprovadas por pessoas que conheçam os processos da organização.

Com base nessas afirmações, assinale a opção correta:

- a) Ambas as afirmações são verdadeiras, mas não se complementam.
- b) A afirmação I é uma proposição falsa e a II é uma proposição verdadeira.
- c) A afirmação I é uma proposição verdadeira e a II é uma proposição falsa.
- d) Ambas as afirmações são verdadeiras e se completam.
- e) Ambas as afirmações são falsas.

2. Quando houver uma suspeita de qualquer tipo de manipulação indevida no banco de dados, é solicitada uma auditoria do banco de dados, que consiste em rever o log e examinar os acessos e operações aplicadas no banco de dados durante determinado período.

Analisemos as seguintes afirmativas:

- I. Auditorias no banco de dados são práticas recomendadas que servem para analisar a integridade dos dados, caso tenha ocorrido alguma manipulação indevida.
- II. Geralmente, as auditorias em banco de dados são realizadas por meio de logs gerados pelo próprio banco de dados.

Com base nas afirmações, assinale a alternativa correta:

- a) Ambas as afirmações são verdadeiras, mas não se complementam.
- b) Ambas as afirmações são verdadeiras e se complementam.
- c) A afirmação I é uma proposição falsa e a II é uma proposição verdadeira.
- d) A afirmação I é uma proposição verdadeira e a II é uma proposição falsa.
- e) Ambas as afirmações são falsas.

3. Para a segurança de um banco de dados, é necessário implementar diversos tipos de mecanismos de segurança, buscando proteger a sua confidencialidade, disponibilidade e integridade, além de realizar testes de estrutura diariamente, verificando backup do banco e monitorando os acessos.

Considerando o contexto acima, analise as afirmações abaixo:

- I. No controle de transações, é avaliado se o local do dado tem os mesmos níveis de confiança que o local de origem.
- II. A criptografia é usada para garantir a segurança dos dados originais, protegendo-os de pessoas não autorizadas.
- III. O controle do acesso físico também tem de ser implementado para garantir disponibilidade.

Considerando o apresentado assinale a sequência das afirmativas corretas:

- a) A alternativa I está incorreta.
- b) A alternativa III está incorreta.
- c) Somente a alternativa II está correta.
- d) Estão corretas as afirmativas I e II.
- e) Todas as alternativas estão corretas.

Seção 1.2

Introdução às operações em banco de dados

Diálogo aberto

Caro aluno, depois de aprendermos o conceito do banco de dados e como realizar uma instalação e configuração de um banco de dados, estaremos prontos para criar um banco de dados e manipulá-lo. Mas, é muito importante lembrar que a manipulação de um banco de dados requer muita atenção e cuidado, pois estamos mexendo com informações importantes, algumas até sigilosas.

Você está achando que é complicado manipular informações? Ou que não será capaz de dar manutenção a um grande banco de dados? Após essa introdução da situação problema, você começará a ver que isso não é um bicho de sete cabeças. Então, vamos lá!

Depois de negociar com seu CEO qual banco usar, você precisa levantar as informações necessárias para a instalação do banco de dados e apresentá-las a ele em um relatório. Quanto espaço é necessário para usar o banco de dados? Qual é a memória mínima necessária para a instalação e o bom funcionamento do banco de dados sem que ele perca performance? Você precisa levar em consideração que o banco de dados precisa ser online e será acessado por vários dispositivos ao mesmo tempo. Então, o que seria melhor: um banco de dados em uma das filiais para ser compartilhado entre as demais, ou um banco de dados na nuvem?

Com o que aprendemos na Seção 1.1, conseguiríamos resolver uma parte dessa situação problema. Para resolvermos o restante, vamos estudar e conhecer mais profundamente como criar e alterar um arquivo de dados no SGBD e as regras de identificadores e comandos de manipulação de dados.

Boa Leitura!

Criação do arquivo de dados no SGBD

Um banco de dados nada mais é que uma coleção de dados relacionados, e essa tecnologia está cada vez mais presente em nosso dia a dia. É viável afirmar que eles representam um papel crítico em quase todas as áreas em que os computadores são usados, incluindo negócios, comércio eletrônico, engenharia, medicina, direito e educação (ELMASRI, 2004, p. 22).

Para podermos manipular um banco de dados, precisamos trabalhar com a linguagem padrão de acesso a bancos de dados, mais conhecida como SQL (*Structure English Query Language*). Essa linguagem, que existe desde a década de 70, foi desenvolvida dentro dos laboratórios da IBM com o intuito de criar uma linguagem simples e limpa de acesso ao banco de dados.



Pesquise mais

Alguns fabricantes de banco de dados começaram a utilizar a mesma sintaxe de acesso ao banco de dados, porém com suas próprias particularidades. Saiba mais sobre esses padrões na Unidade 2 desta obra (NISHIMURA, 2009).

Dentro do banco de dados existem objetos que agrupam informações, chamados de arquivos de dados ou tabelas. Para manipularmos essas tabelas, utilizamos a linguagem de definição de dados (DDL), que são comandos específicos para a estrutura própria do banco de dados.

Create: Comando usado para construir um novo objeto, seja ele banco de dados, tabela, índice ou visão.

Exemplos de Criação de objeto:

Create Database: é usado para criar fisicamente um novo banco de dados onde esteja instalado o SGBD. A sintaxe base do comando para criar o banco de dados é modificada em cada SGBD para adaptar-se às características de cada um.

Create database <nome do banco>

Create Table: é usado para criar uma nova tabela dentro de um SGBD, especificando seus atributos e restrições iniciais. Os atributos são especificados primeiro, e cada um deles recebe um nome, um tipo de dados para especificar seu domínio de valores e outras restrições de atributos, como o *Not Null*. As restrições de chave, integridade de entidade e integridade referencial podem ser especificadas na instrução, após os atributos serem declarados. É possível, também, acrescentá-las depois, usando o comando *Alter Table*. Ao criar uma coluna, você precisa, primeiramente, pensar em qual tipo de atributo usar para ela. Os tipos de dados básicos disponíveis para atributos são: numérico, cadeia de caracteres, cadeia de bits, booleano, date e time.

```
Create table Clientes (  
    Codigo Int,  
    Nome Varchar(100),  
    Idade Int  
    Primary Key (Codigo)  
);  
  
Create table Filmes (  
    Codigo Int,  
    NomeFilme Varchar(100),  
    Cliente Int,  
    DataAluguel DateTime,  
    Primary Key (Codigo),  
    Foreign Key (Cliente)  
References Clientes(Codigo)  
);
```

Create Index: é usado para criar um índice novo para a tabela existente.

```
Create index Idx_Clientes On Clientes (Nome Asc);
```

Create view: cria uma visão lógica de uma ou mais tabelas existentes dentro do banco de dados. Uma visão não é uma tabela física, é uma organização lógica de uma ou mais tabelas.

```
Create view V_Clientes as (Select Codigo, Idade  
From Clientes Order by Codigo Desc)
```

Create procedure: são programas com uma lógica específica que executam determinados comandos e estão armazenados diretamente no banco de dados.

```
Create procedure P_Clientes
```

Create function: é parecido com a *procedure*, porém as *procedures* podem passar e retornar valores, já a função pode passar valores, mas retorna apenas um valor no final da execução.

```
Create function F_Clientes
```



Reflita

Como você resolveria um problema de um *insert* que foi executado com informações incorretas? Caso o erro seja somente de algumas colunas, é possível resolver com um script de *update*, mas se o erro foi em uma linha inteira incluída de forma errada na tabela, então a maneira correta de corrigi-lo é rodar o script de delete somente na linha e depois inseri-la de forma correta.



Pesquise mais

Identificadores são extremamente importantes em nosso cotidiano com o banco de dados, que, por vezes, não entendemos bem seu funcionamento e suas regras. Quando escrevemos requisições SQL, usamos nomes para nos referirmos aos bancos de dados e a outros objetos que têm, em seus componentes, nomes próprios, por exemplo, tabelas em suas colunas e índices. É possível, ainda, criar apelidos que atuam como sinônimos para nomes de tabelas ou colunas. Saiba mais sobre estes identificadores e suas regras na obra *Bancos de Dados II* (NISHIMURA, 2009, p. 94)

Manipulando um banco de dados

Usando a DDL, além de criar um objeto, podemos alterar ou remover o mesmo objeto criado (VICCI, 2009, p. 26):

Alter – Comando usado para modificar um objeto já criado, tal como banco de dados, tabela, índice ou visão.

Alter database – Modifica um banco de dados.

Alter table – Modifica uma tabela.

Alter index – Modifica uma índice.

Alter view – Modifica uma visão.

Drop – Serve para remover fisicamente um objeto criado, tal como banco de dados, uma tabela, um índice ou uma visão.

Drop database – Exclui o banco de dados.

Drop table – Exclui a tabela.

Drop index – Exclui o índice.

Drop view – Exclui a visão.

Drop function – Exclui a função.

Drop procedure – Exclui o procedimento.



Assimile

Em empresas onde existem tarefas bem definidas, os comandos DDL são restritos aos administradores de banco de dados (DBAs) por questões de segurança.

Procedimentos e funções não podem ser modificados, nesse caso. Para alterá-las, é preciso removê-los e depois inclui-los novamente, com as alterações.

Conceitos de manipulação e modelagem de banco de dados

Os comandos de manipulação de dados são conhecidos como *Data Manipulation Language* (DML). Diferente do DDL, que trabalha com o objeto, o DML trabalha com a informação dos dados, usando linguagem padrão dos SGBD, o SQL (VICCI, 2009).

Os principais comandos do DML são:

Select – Seleciona dados dentro de uma ou mais tabelas ou visões.

Insert – Insere novos dados dentro das tabelas.

Update – Atualiza novos dados dentro das tabelas.

Delete – Apaga dados existentes na tabela.

Exemplos de uso do DML, usando uma tabela com nome clientes:

Clientes		
Codigo	Nome	Idade
1	João	10
2	Zeca	20
3	Alex	16

-- Retornar somente o resultado da coluna código e nome:

Select c.Codigo, c.Nome **From** Clientes c;

Codigo	Nome
1	João
2	Zeca
3	Alex

– Insere um novo registro na tabela:

Insert into Clientes(Nome, Idade) **Values** ("Ronaldo",30);

Clientes		
Codigo	Nome	Idade
1	João	10
2	Zeca	20
3	Alex	16
4	Ronaldo	30

– Atualiza os registros da tabela:

Update Clientes **Set** Idade = 30;

Clientes		
Codigo	Nome	Idade
1	João	30
2	Zeca	30
3	Alex	30
4	Ronaldo	30

– Apaga os registros da tabela:

Delete from Clientes;

Clientes
Sem Registros

Dentro do DML, existem algumas subclasses, tais como a *Data Control Language* (DCL), a *Data Transaction Language* (DTL) e a *Data Query Language* (DQL).

A DCL é a linguagem de controle de acesso de dados, usada para conceder permissões, privilégios de acesso ou manipulação em uma tabela específica por usuário.

Connect – Privilégio de conexão ao banco de dados.

Select – Privilégio de executar o comando select na tabela.

Insert – Privilégio de executar o comando insert na tabela.

Update – Privilégio de executar o comando update na tabela.

Delete – Privilégio de executar o comando delete na tabela.

Execute – Privilégio de executar o comando execute na tabela.

Usage – Privilégio de executar o comando usage.

Esses comandos normalmente são restritos aos DBAs.

A DTL é a linguagem de controle de transações de dados.

Begin Transaction – Inicia uma transação do banco de dados.

Commit – Confirma todas as operações realizadas no início da transação até este momento.

Rollback – Descarta todas as operações realizadas desde o início da transação corrente, voltando todos os dados envolvidos aos seus respectivos valores anteriores.

A DQL é a mais usado no dia a dia para realizar consultas ou filtros mais elaborados, como retorno de uma ou mais tabelas simultaneamente.

Exemplos de uso do DQL, usando uma tabela com nome clientes:

Clientes		
Codigo	Nome	Idade
1	João	10
2	Zeca	20
3	Alex	16

Filmes			
Codigo	NomeFilme	Cliente	DataAluguel
1	Star Wars	2	01/02/2018
2	Avatar	2	02/01/2017
3	Vingadores	1	03/02/2018

– Retornar clientes acima de 16 anos:

Select * From Clientes c **Where** c.Idade >= 16 **Order by** c.Idade;

Codigo	Nome	Idade
3	Alex	16
2	Zeca	20

– Retornar clientes que alugaram filmes no mês de janeiro:

Select c.Nome, c.NomeFilme **From** Clientes c, Filmes f **Where** c.codigo = f.Cliente **and** f.DataAluguel **between** '01-01-2018' **and** '31-01-2018';

Nome	NomeFilme
Zeca	Avatar

– Retornar os clientes acima de 16 anos e o código do cliente ser menor que 3:

Select * From Clientes c **Where** c.Idade >= 16 **and** c.Codigo < 3;

Codigo	Nome	Idade
2	Zeca	20

– Apagar somente o primeiro registro:

Delete from Clientes **c Where** c.codigo = 1;

Codigo	Nome	Idade
2	Zeca	20
3	Alex	16

– Atualiza o cliente a idade do cliente para 20 quando o codigo for 3 e idade acima de 16:

Update Clientes **Set** Idade = 20 **Where** c.Codigo = 3 **and** c.Idade = 16;

Clientes		
Codigo	Nome	Idade
1	João	10
2	Zeca	20
3	Alex	20



Exemplificando

Neste SQL vamos apresentar uma estrutura mais completa usando duas tabelas:

```
Select c.Nome,
        f.NomeFilme,
        f.DataAluguel
```



```
From Clientes c, Filme f  
  
Where c.Codigo = f.Cliente  
  
Order by c.Codigo
```

A tabela Filme tem ligação com a tabela Clientes pela coluna Cliente. Na instrução SQL acima, foi realizado um *join* na cláusula **Where**, fazendo a ligação entre as tabelas.

Essas instruções SQL são padrão na maioria dos bancos de dados. Com base no que aprendemos, você será capaz de realizar movimentações e consultas em diversos tipos de banco de dados.

Sem medo de errar

Conforme abordamos no início desta seção, o desafio era apresentar informações necessárias para a instalação do banco de dados e apresentar através de um relatório. Para levantar informações usaremos o software JMeter, que foi apresentado em nossa primeira seção. Nele vamos conseguir realizar testes de performance, acessos e espaço disponível para o nosso banco de dados.

Após os testes de performance, crie uma estrutura no banco de dados para que o CEO possa também verificar a velocidade de consulta, e suas informações se estão todas corretas.

Vamos criar um exemplo usando três tabelas, cliente, produto e venda:

```
Create table cliente (  
    Codigo Int,  
    Nome Varchar(100) ,  
    Telefone Varchar(20) ,  
    Email Varchar(50) ,
```

```

        Primary Key (Codigo)
    );

Create table produto (
    Codigo Int,
    Descricao Varchar(100),
    Preco Numeric,
    Primary Key (Codigo)
);

Create table venda (
    Codigo Int,
    Cod_cliente int,
    Cod_produto int,
    Quantidade int,
    Primary Key (Codigo),
    Foreign Key (Cod_cliente)
References cliente (Codigo),
    Foreign Key (Cod_produto)
References produto (Codigo)
);

```

Após criar as tabelas vamos inserir as informações nas tabelas de cliente e produto.

```

Insert into cliente (Codigo, Nome, Telefone,
Email) Values (1,"João", "(19)98756.4321", "joao@
google.com.br" );

```

```

Insert into cliente (Codigo, Nome, Telefone,
Email) Values (2,"Maria", "(11)92213.0123", "maria@
google.com.br" );

```

```

Insert into cliente (Codigo, Nome, Telefone,
Email) Values (3,"José", "(21)90123.2222", "jose@
google.com.br" );

```

```

Insert into produto (Codigo, Descricao, Preco)
Values (1,"Tênis", "99.90" );

```

```
Insert into produto (Codigo, Descricao, Preco)  
Values (2,"Óculos", "200.00");
```

```
Insert into produto (Codigo, Descricao, Preco)  
Values (3,"Sapato", "110.00");
```

Depois de inseridos registros nas tabelas de cliente e produto, podemos incluir registros na tabela de venda lembrando que esta tabela tem referência nas tabelas de cliente e produto, ou seja, não será possível incluir registros que não existam nestas duas tabelas.

```
Insert into venda (Codigo, Cod_cliente, Cod_  
produto, Quantidade) Values (1, 1, 2, 3);
```

```
Insert into venda (Codigo, Cod_cliente, Cod_  
produto, Quantidade) Values (2, 2, 1, 1);
```

```
Insert into venda (Codigo, Cod_cliente, Cod_  
produto, Quantidade) Values (3, 3, 3, 5);
```

Pronto, já temos o relatório com as informações de testes de base de dados, agora só falta apresentar para o CEO como seria uma consulta em um banco de dados mostrando seus resultados.

Vamos ao select para que possa exibir as informações das 3 tabelas.

```
Select v.Codigo,  
        c.Nome,  
        c.Telefone,  
        c.Email,  
        p.Descricao,  
        v.Quantidade,  
        p.Preco  
  
From cliente c, produto p, venda v  
Where v.Cod_cliente = c.Codigo  
        And v.Cod_produto = p.Codigo  
  
Order by v.Codigo
```

Após realizar a consulta SQL, apresente para o seu CEO, os resultados obtidos.

Consulta entre três tabelas

Descrição da situação-problema

Uma empresa do ramo de calçados precisa saber quais marcas e quais tamanhos desta marca foram vendidos neste mês, para que possa analisar junto com o pessoal do departamento de compras a rotatividade do produto no mercado. Porém no sistema deles não existe algum relatório que mostre estas informações e você foi solicitado para que ajude nesta situação. Para facilitar eles enviaram a estrutura das tabelas.

Venda
Codigo
Preco
CodigoProduto
Qtde
DataVenda

Produto
Codigo
Descricao
Tamanho
Saldo
CodigoMarca

Marca
Codigo
Descricao

Resolução da situação-problema

Para resolvermos essa situação, devemos nos atentar as referências das tabelas. A Tabela Venda tem o campo CodigoProduto que faz referência a tabela Produto que tem a coluna tamanho e CodigoMarca que faz referência com a tabela Marca. Logo teremos o seguinte SQL.

```
Select m.Descricao, p.Tamanho, v.DataVenda
From Venda v, Produto p, Marca m
Where v.CodigoProduto = p.Codigo
and p.CodigoMarca = m.Codigo
and v.DataVenda between '01-03-2018' and '31-03-2018';
```

Nesse SQL, estamos usando um *Join* que nada mais é que uma junção entre as tabelas, Venda, Produto e Marca.

Para referenciar a junção entre estas tabelas, teremos que usar dentro da cláusula **Where** como a tabela de Venda.CodigoProduto com a tabela Produto.Codigo e a tabela de Produto.CodigoMarca com a tabela Marca.Codigo

Para verificar as vendas no mês usamos, incluímos na cláusula o filtro da data onde usamos a função **between** que é uma função nativa do banco de dados e é usada para verificar período entre datas. Também poderíamos usar uma cláusula diferente na data, em vez de usar o **between** poderíamos usar `v.DataVenda > '01-03-2018' and v.DataVenda < '31-03-2018'` e o resultado seria o mesmo, a diferença seria somente na escrita do SQL.

Faça valer a pena

1. No SGBD existem objetos que agrupam informações, chamados de arquivo de dados. Para manipularmos esses arquivos, utilizamos a DDL, que são comandos específicos para a estrutura própria do banco de dados.

De acordo com o texto-base, qual é o comando usado para construir um novo objeto no banco de dados?

- a) SQL.
- b) *Insert*.
- c) *Create*.
- d) *Update*.
- e) *Alter Table*.

2. Ao criar uma coluna você precisa primeiramente pensar em qual tipo de atributo usar para ela. Os tipos de dados disponíveis para atributos são: numérico, cadeia de caracteres, cadeia de bits, booleano, date e time.

Temos a seguinte estrutura de tabela:

```
Create table Fornecedores (  
    Codigo Int,  
    RazaoSocial Varchar(255) ,  
    Telefone Varchar(50) ,  
    Endereco Varchar(255) ,  
    Contato Varchar(50) ,
```

```
DataCadastro DateTime,  
Primary Key (Codigo),  
);
```

De acordo com a estrutura da tabela acima qual seria o comando para incluir uma coluna e-mail?

- a) Alter Table Add email varchar(200);
- b) Update Table Add email varchar(200)
- c) Insert Table Add email text
- d) Execute Table Add email text varchar(200);
- e) Alter Table email varchar(200);

3. Quando você compra um ingresso pela internet, para assistir a um show, um filme ou uma partida de futebol, o banco de dados do site que vendeu o ingresso é modificado para que o outro usuário não consiga comprar o mesmo ingresso.

Qual é o comando realizado no banco de dados para alterar o status do ingresso vendido?

- a) Insert into IngressosVendidos (CodigoIngresso, Cliente, DataVenda) Values (002018, 'João', '18-09-2017');
- b) Insert IngressosVendidos (CodigoIngresso, Cliente, DataVenda) Values (002018, 'João', '18-09-2017')
- c) Insert into IngressosVendidos Values (002018, 'João', '18-09-2017')
- d) Insert into IngressosVendidos (CodigoIngresso, DataVenda) Values (002018, '18-09-2017')
- e) Update IngressosVendidos (CodigoIngresso, Cliente, DataVenda) Values (002018, 'João', '18-09-2017')

Seção 1.3

Desempenho do banco de dados

Diálogo aberto

Caro aluno, com o conteúdo das seções anteriores, entendemos o conceito de um banco de dados, como realizar uma instalação de um banco de dados e como realizar a manipulação de dados. Aprendemos, também, sua importância na nossa vida pessoal e profissional e onde são usados, como bancos, celulares, cartórios, etc. Nesta seção, vamos aprender como criar um banco de dados na plataforma Windows™, gerenciar serviços de banco de dados e como monitorar sua performance. Para isso, daremos continuidade à nossa consultoria na empresa de fast-food, pois o CEO está insatisfeito com seu sistema desktop e precisa visualizar informações das lojas para tomar as devidas decisões.

Após o levantamento dos requisitos, a segunda parte seria a instalação do banco de dados no servidor e, depois de instalado e configurado, você precisa criar uma estrutura dentro do banco de dados para que você possa apresentar ao CEO, por meio de uma aplicação simples, que o banco de dados está funcionando corretamente, exibindo um relatório das informações inseridas no banco. Quais testes você faria para comprovar um funcionamento saudável do banco de dados? Onde ficaria salvo o arquivo de dados? Quais comandos você utilizaria para mostrar, no SGBD, que as informações salvas estão sendo armazenadas lá? Lembre-se de que serão várias filiais acessando o mesmo banco de dados ao mesmo tempo.

Para auxiliá-lo nesse desafio, você aprenderá a gerenciar serviços de banco de dados na plataforma Windows e a monitorar sua performance através de ferramentas específicas que ajudarão você na sua vida profissional.

Bom estudo!

Seja bem-vindo à nossa última seção da primeira unidade! Nesta última etapa da unidade, você irá aprender como funciona e as vantagens de usar um banco de dados na plataforma Windows. Também aprenderá como gerenciar serviços de banco de dados e monitorá-los, sabendo quanto está consumindo um banco de dados, o número de acessos e a memória disponível.

Banco de dados na plataforma Windows

Quando usamos um banco de dados na plataforma Windows, significa que já escolhemos a estrutura em que iremos trabalhar, pensando em um melhor aproveitamento do banco de dados (ORACLE, 2018).

Ao instalarmos o banco de dados Oracle® MySQL na plataforma Windows, nos deparamos com a facilidade de uso, para os administradores de banco de dados, pois as são ferramentas mais intuitivas, e para o usuário, pois já há familiaridade no uso da plataforma (ORACLE, 2018) e há suporte e atualizações constantes de produtos.

Usando o MySQL na plataforma Windows, temos um conjunto de ferramentas para desenvolver e gerenciar aplicativos. Estas ferramentas são (ORACLE, 2018):

- MySQL para Excel: um componente que permite navegar por esquemas, tabelas, visualizações e procedimentos do MySQL a partir do Excel. Com ele, podemos importar dados do MySQL para o Excel, modificar dados diretamente dentro do Excel, e exportar dados do Excel para o MySQL com uma nova tabela ou acrescentar dados em uma nova existente.
- MySQL para Visual Studio: um componente que fornece acesso aos dados MySQL dentro do Visual Studio. Ele também é encontrado como um pacote no Visual Studio. Esse componente se integra diretamente ao Server Explorer, oferecendo a capacidade de criar conexões com o banco de dados MySQL. Alguns exemplos de suas funcionalidades são: desenvolvimento SQL, designer de consulta e depuração de rotina armazenada.

- MySQL Workbench: uma ferramenta gráfica para trabalhar com servidores de banco de dados MySQL. O MySQL Workbench comporta todas as versões antigas a partir do MySQL 5.x, devido a tabelas do sistema alteradas. Algumas de suas principais funcionalidades são: desenvolvimento SQL, *data modeling* (design), administração do servidor e migração de dados.
- MySQL conector/.Net: esse componente permite que você desenvolva aplicativos .NET que requerem conectividade de dados segura e de alto desempenho com o MySQL. Ele implementa as interfaces necessárias do ADO.NET e integra-se às ferramentas. O desenvolvedor pode criar aplicativos totalmente gerenciados, escritos em C#.
- MySQL conector do MySQL/ODBC: esse componente oferece aos desenvolvedores o acesso a um banco de dados usando a API *Open Database Connectivity* (ODBC). O ODBC fornece interfaces nativas baseadas em gerenciador de driver para o banco de dados MySQL, com total funcionalidade, incluindo procedimentos de armazenamento e transações.



Assimile

Para poder escolher o tipo de ambiente em que será instalado o banco de dados, é necessário realizar o levantamento de estrutura do mesmo, se o banco precisará usar replicação de informações, alocação de dados. Saiba mais na unidade 7 da obra (ELMASRI, 2005).

Gerenciando serviços de banco de dados no Windows

ELMASRI (2005), afirma que, em geral, os SGBDs usam os próprios gerenciadores para garantir o acesso mútuo e exclusivo aos recursos compartilhados. Esses gerenciadores são implementados no espaço do usuário no nível dos softwares de aplicação. Um exemplo de uso são várias bases de dados realizando uma conexão ao mesmo tempo: esse gerenciador será o responsável por enfileirar as transações de forma que não prejudique o desempenho do SGBD.

O Windows só não tem conhecimento a respeito deles, logo, se o Windows desativar um processo do SGBD, outros processos do SGBD que esperam esse recurso de bloqueio serão enfileirados. Essa situação pode causar uma séria degradação de desempenho, tal como consultas não finalizadas e conexões perdidas do banco de dados. Para resolvermos esses processos, às vezes é necessário reiniciar o serviço do banco de dados ou parar o serviço até que se normalize o servidor. Abaixo, vamos entender melhor como realizar estes tipos de comandos que ajudariam no desempenho.

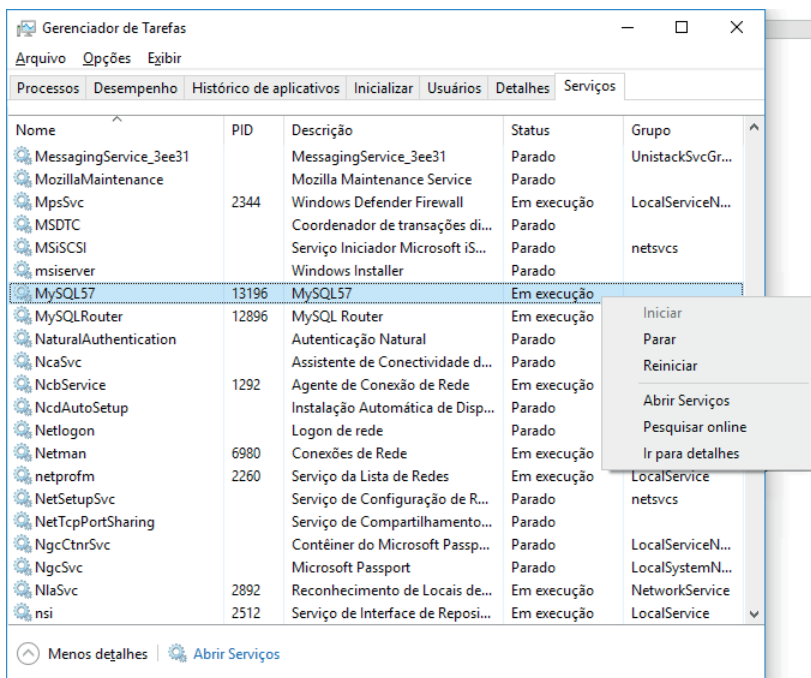
Para executar manualmente um processo no servidor MySQL no Windows a partir de linha de comando, é necessário navegar até a pasta bin, que por padrão, fica localizada em *C:\Program Files\MySQL\MySQL Server 5.x*, abaixo do diretório de onde foi instalado o MySQL, e parar o prompt de comando. Com ele já é direcionado à pasta bin, é possível realizar diferentes comandos disponíveis, tais como:

```
Shell> net start MYSQLTESTE
```

```
Shell> net stop MYSQLTESTE
```

Nesse caso, o **MYSQLTESTE** é o nome do nosso serviço. Para podermos controlar o serviço **MYSQLTESTE**, utilize a ferramenta gráfica de gerenciador de serviços, em que será exibida uma janela (Figura 1.3) com a listagem de todos os serviços, com opções para parar e iniciar.

Figura 1.3 | Tela do Gerenciador de Tarefas Aba Serviços

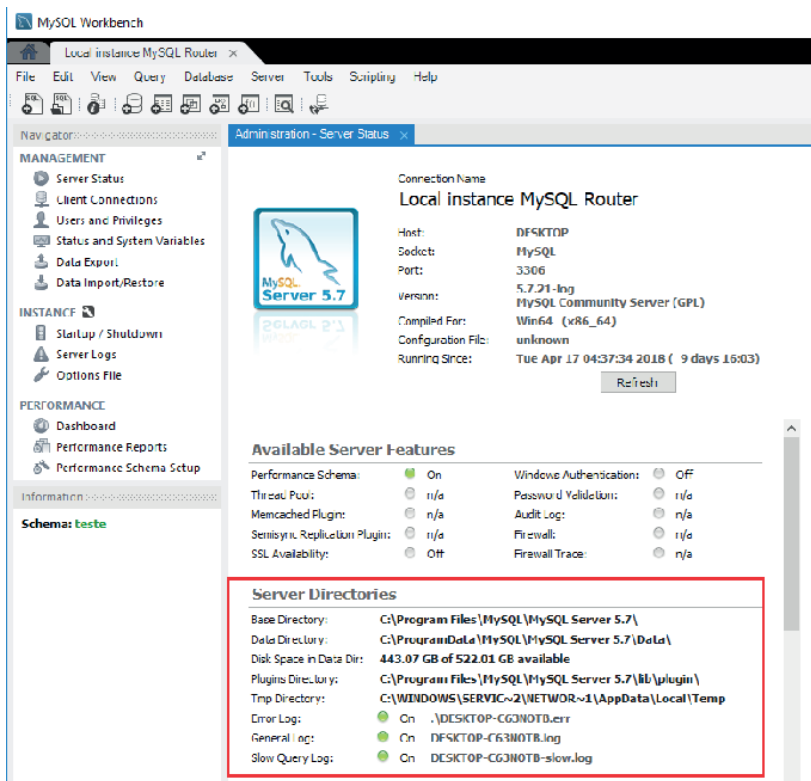


Fonte: captura de tela do Gerenciador de Tarefas do Windows, adaptada pelo autor.

Também é possível parar um serviço manualmente, a partir da linha de comando **mysqladmin shutdown**.

Caso o serviço não inicie quando for executado, verifique o log de arquivo de erros, no diretório de dados. A Figura 1.4 mostra onde ficam localizados os diretórios. Outra opção é executar o processo no servidor manualmente, usando a opção **--console**.

Figura 1.4 | Tela do Mysql Workbench (Server Status)



Fonte: captura de tela do MySQL, elaborada pelo autor.

Para remover serviço **MYSQLTESTE**, primeiramente pare o serviço se ele estiver sendo executado e, em seguida, execute o comando **Shell>mysql -remove**.

Os comandos abaixo são baseados no serviço nomeado como **MYSQLTESTE**. Para executar com um nome diferente, utilize comandos que especifiquem o nome explicitamente:

```
Shell>mysql --install my_service_new
```

```
Shell>mysql --install my_service_new --defaults-file=c:\server-opts
```

```
Shell>mysqld --remove my_service_new
```

```
Shell>net start my_service_new
```

```
Shell>net stop my_service_new
```

Caso você instale o processo no servidor, em forma de serviço, com outro nome que não seja **MYSQLETESTE**, e não for especificado o parâmetro **-defaults-file**, o processo do servidor irá ler as todas as opções do arquivo padrão no grupo **[my_service_new]**, em vez das opções do grupo padrão **[mysqld]**.



Exemplificando

Alguns servidores trabalham com mais de uma versão do mesmo banco de dados, seja por uma das estruturas ser antiga e não comportar a versão mais nova, seja por erro de administração no banco de dados. Quando ocorre algum tipo de problema que requer parar o serviço e subir o banco de dados novamente, é preciso ter muito cuidado, pois, nesses casos, haverá mais de um serviço.

Para entender melhor como configurar um serviço do MySQL durante a instalação, acesse a documentação no próprio site do MySQL, pelo link:

<<http://download.nust.na/pub6/mysql/doc/administrator/pt/mysql-administrator-service-control-configure-service.html>>. Acesso em: 21 jun. 2018.

Monitoramento de performance

O que faz o log de erros do SGBD estar com tamanho excessivo? O que acontece quando o número de conexões no banco de dados excede? O que pode acontecer quando o nível do buffer ultrapassar 90%? E quando o nível da CPU estiver acima do normal?

Ramakrishnan (2011) afirma que as aplicações complexas são construídas sobre vários gerenciadores de recursos, tais como os sistemas de gerenciamento de banco de dados, os sistemas operacionais, as interfaces com o usuário e os softwares de troca de mensagens. O monitoramento de performance congrega os serviços de vários gerenciadores de recursos e fornece aos programadores de aplicativo uma interface uniforme para o desenvolvimento de suas transações. Por fim, o monitor exibe os

indicadores de consumo da aplicação, como memória utilizada, usuários conectados e consultas e transações em andamento.

Um SGBD fornece muitas das funções suportadas a um monitor de performance, além de processar consultas e atualizar o banco de dados com eficiência. Com o monitoramento, é possível analisar e otimizar continuamente a performance do SGBD.

Ao monitorar transações, deve-se notar que elas devem ser executadas dentro de um prazo especificado pelo usuário ou desenvolvedor. Um prazo limite fixo significa que o valor da transação é zero após o prazo-limite. Por exemplo, em um SGBD projetado para registrar apostas em corridas de cavalo, uma transação que faz uma aposta é inútil quando a corrida começa. Tal transação não deve ser executada e a aposta não deve ser feita. Um prazo provisório significa que o valor da transação diminui após o prazo, eventualmente chegando a zero. Em um SGBD projetado para monitorar alguma atividade (por exemplo, um reator complexo), uma transação que pesquisa a leitura corrente de um sensor deve ser executada em um tempo curto, digamos, um segundo. Quanto mais tempo se leva para executar a transação, menos útil a leitura se torna. Em um SGBD de tempo real, o objetivo é maximizar o valor das transações executadas, e o SGBD deve priorizar as transações levando em conta seus prazos.



Refleta

O que fazer quando um sistema web não termina de carregar as informações na tela e, conseqüentemente, impossibilita outros acessos, prejudicando o desempenho do sistema? Pode ser que o problema esteja na construção da instrução SQL ou em haver múltiplos acessos à mesma tela. Para descobrir o que houve, verifique no monitor o que está ocorrendo durante o processo de execução da tela e tente reiniciar o serviço do banco de dados. Porém, essa não é a solução efetiva. O correto seria validar o trecho do código que chama a tela e verificar se a consulta ou a transação está sendo executada ou construída de forma correta.

Monitoramento de performance no Windows

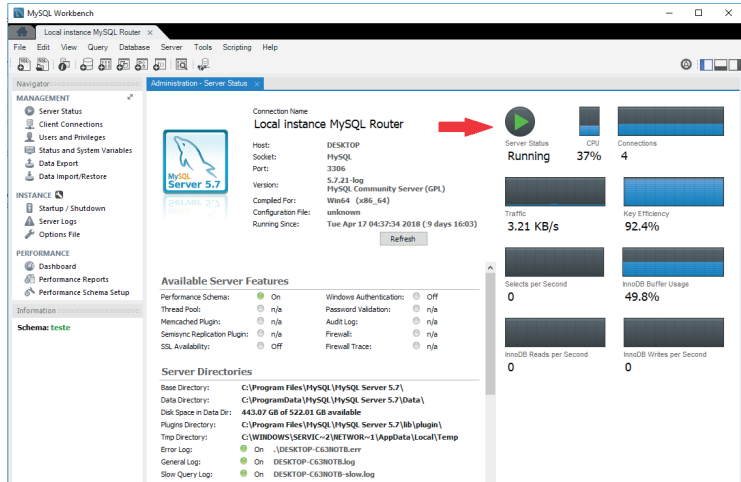
Para monitorarmos performance na plataforma Windows, precisamos de ferramentas que sejam úteis e adequadas a essa plataforma. Um exemplo de ferramenta para verificarmos performance é o MySQL Workbench. Suas principais funcionalidades abrangem:

- **Desenvolvimento SQL:** ele possibilita gerenciar e criar conexões com servidores de banco de dados, além de permitir que você configure parâmetros de conexão.
- **Administração do Servidor:** permite que você administre instâncias do servidor MySQL, administrando usuários, executando backup e recuperação, inspecionando dados de auditoria, visualizando a integridade do banco de dados e monitorando o desempenho do servidor MySQL.
- **Data Modeling (Design):** permite criar modelos do esquema do banco de dados de forma gráfica, reverter e encaminhar o engenheiro entre um esquema e um banco de dados ativo, editar todos os aspectos do banco de dados usando o Editor de Tabelas, além de fornecer recursos fáceis para editar Tabelas, Colunas, Índices, Disparadores, Particionamento, Opções, Inserções e Privilégios, Rotinas e Visualizações.

Agora que já entendemos o Workbench, abra-o com uma conexão local e verifique o Status do Servidor (*Server Status*). Na Figura 1.5, note que se o seu servidor estiver iniciado, ele irá indicar que o servidor está rodando, e seu painel de performance estará ativo, mostrando o funcionamento do servidor.

Nele você conseguirá visualizar se o servidor está a ativo, quanto da CPU está sendo usado no computador, o tráfego da rede onde o servidor está conectado, quantas seleções estão rodando por segundo, o percentual de uso do buffer e a quantidade de leituras e escritas no InnoDB.

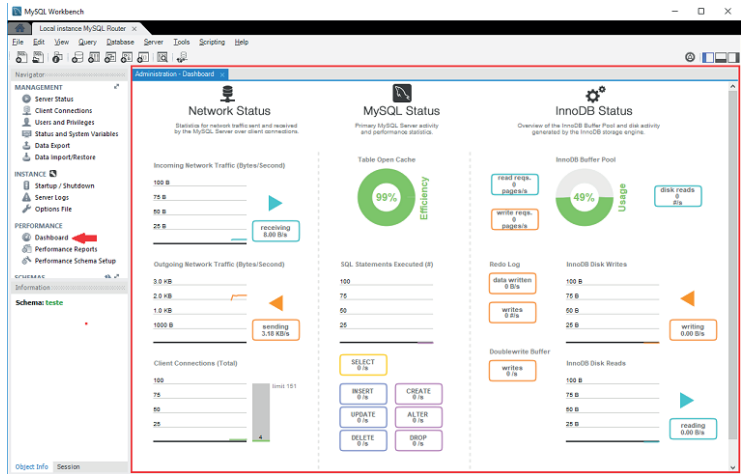
Figura 1.5 | Tela de Status do Servidor (Server Status) com serviço Iniciado



Fonte: captura de tela do MySQL Workbench, adaptado pelo autor.

Para verificar mais a fundo quantas seleções estão sendo rodadas, quantos processos estão sendo executados ao mesmo tempo, como o banco está se comportando diante do tráfego de entrada e saída, usaremos a ferramenta *Dashboard*, como mostra a Figura 1.6. Nela, é possível analisarmos mais profundamente como que está a performance do banco de dados, se o servidor precisa de mais memória, se a rede está lenta ou se existem processos que impedem que o tráfego continue normalmente.

Figura 1.6 | Tela do *Dashboard* exibindo a performance do servidor



Fonte: captura de tela do MySQL Workbench, adaptado pelo autor.

O *Dashboard* é muito útil para verificar processos lentos de sistemas, pesquisas lentas ou não concluídas por informações, processos de backup que podem consumir também o tráfego da rede, consequentemente afetando aos usuários, e requisições em excesso.



Pesquise mais

Usamos aqui o Workbench como exemplo de monitoramento de performance, mas existem muitos outros, tanto para o banco de dados MySQL como para outros SGBDs no mercado. Outros exemplos de monitores de performance são o Mytop e o Innotop, que, diferente do Workbench, têm uma interface gráfica, visualizada por meio de console.

Para entender mais sobre visualizar e monitorar a performance do banco de dados no Windows acesse:

BÓSON TREINAMENTOS. **Instalando MySQL Server e MySQL Workbench no Windows 10**. 7 jun. 2016. Disponível em: <<https://www.youtube.com/watch?v=TbzfByXwCzk>>. Acesso em: 20 jun. 2018.

Com base no que aprendeu até agora, você com certeza tem conceitos claros em relação ao banco de dados, a como usá-lo no cotidiano profissional e a qual sua importância nas nossas vidas. Sabe, também, como podemos gerenciar e monitorar as atividades dentro do banco de dados e manipular informações de dados. Mas isso não é tudo, então continue estudando, leia bastante e busque conhecimentos em várias fontes. Certamente, logo você se destacará na administração de bancos de dados.

Sem medo de errar

Conforme proposto no início dessa seção, nosso desafio é criar uma estrutura dentro do banco de dados, para que você possa apresentar ao seu CEO, por meio de uma aplicação simples, que o banco de dados está funcionando corretamente. Quais testes devem ser feitos para comprovar um funcionamento saudável do banco de dados? Onde deve ser salvo o arquivo de

dados? Quais comandos deve-se utilizar para mostrar, dentro do SGBD, que as informações salvas estão sendo armazenadas nele?

Essa situação-problema é bem interessante, pois usaremos também os conteúdos das seções passadas. Primeiramente, precisamos criar informações dentro do banco de dados e, para isso, usaremos o comando **Create**:

```
Create table Clientes (  
  
    Codigo Int,  
  
    Nome Varchar(100) ,  
  
    Idade Int  
  
    Primary Key (Codigo)  
  
);
```

Com esse comando, criamos a tabela cliente. Depois de criada, vamos inserir nela informações para que, futuramente, possamos visualizar estas informações. Para inserir as informações, vamos usar o comando **Insert**:

```
-- Insere um novo registro na tabela  
  
Insert into Clientes (Nome, Idade) Values  
("Ronaldo", 30);
```

Com esse comando, inserimos os dados na tabela **Clientes**.

Como estudamos o monitoramento por meio do Workbench, vamos mantê-lo como exemplo da apresentação. Para a apresentação, abra o Workbench e execute o comando **Select** para que o CEO visualize os dados salvos no banco.

```
-- Retornar somente o resultado da coluna código  
e nome  
  
Select c.Codigo, c.Nome From Clientes c;
```

Codigo	Nome
1	Rolando

Feito isso, no MySQL Workbench, abra o *Dashboard* e apresente para o CEO como funcionaria a performance do banco de dados quando executados esses comandos.

Para mostrar onde ficaria salvo o arquivo de dados, você tem que acessar o Status do Servidor, que fica no centro do painel, e apresentar os Diretórios do Servidor (*Server Directories*), em que são exibidos os diretórios usados no banco de dados conforme visto na Figura 1.4.

Avançando na prática

Monitorando a performance do banco de dados durante a Black Friday

Descrição da situação-problema

Black Friday é o dia que inaugura a temporada de compras para o natal, com aumento de até 16% nas vendas em um único dia. Nessa ocasião, muitos empresários de *e-commerce* se preocupam com a estrutura de seu servidor e se o banco de dados conseguirá aguentar a imensa carga de informações de uma só vez. Com um *e-commerce* em mãos, você precisa analisar sua estrutura e informar o CEO sobre quais atitudes tomar para que o *e-commerce* não fique off-line nessa data.

Resolução da situação-problema

Abra o MySQL Workbench e acesse o *Dashboard*. Com base nas informações apresentadas no painel, teremos que tomar medidas diferentes, por exemplo, número de conexões excedendo o limite de acesso ou uso de memória e de CPU acima do normal. Nesse caso, é preciso limitar o número de acessos ao banco e verificar se o mesmo usuário está tentando se conectar diversas vezes, ao

mesmo tempo. A solução, portanto, se daria com o uso de um software, em que o desenvolvedor criaria uma trava para validar a conexão. Se o problema persistisse, então o certo seria parar o serviço até que o uso da CPU voltasse ao normal para que assim volte a iniciar o serviço.

Caso o processo esteja em um banco específico como um SQL lento, é recomendado primeira mente finalizar aquele processo SQL que serviria de correção emergencial. Após finalizar este processo, é preciso analisar junto ao desenvolver que tipo de SQL está rodando e validar se o mesmo precisa de alguma melhoria no código para que não consuma memória no banco de dados.

Faça valer a pena

1. Alguns servidores de banco de dados realizam automaticamente atualizações que, por sua vez, requerem o reinício do sistema para que funcionem de forma correta. Porém, pode ocorrer que, ao reiniciar a plataforma, o serviço do banco de dados não seja inicializado.

Com base na afirmação acima, qual seria o comando para iniciar o serviço do banco de dados?

- a) `start MYSQL`
- b) `net stop MYSQL`
- c) `net restart MYSQL`
- d) `net MYSQL`
- e) `net start MYSQL`

2. Conforme estudamos, aprendemos a visualizar a performance de um banco de dados usando a ferramenta do MySQL Workbench. Com ela podemos analisar e verificar o status do servidor, além de executar comandos de criação ou manipulação de informações.

Com base no estudo da ferramenta Workbench, assinale onde podemos monitorar as informações de performance do banco de dados.

- a) *Dashboard*
- b) *Server Logs*
- c) *Startup/Shutdown*
- d) *Client Connections*
- e) *Status and System Variables*

3. Com o monitor de performance, conseguimos analisar e validar informações em tempo real e, com base nelas, podemos tomar decisões sobre como realizar uma manutenção preventiva no servidor. Porém, todo cuidado é pouco e devemos sempre manter o banco de dados ativo.

De acordo com o enunciado, como saber que o banco de dados não está ativo?

- a) Ao abrir o gerenciador de serviços e constar que o serviço do banco de dados está parado.
- b) Por meio do *Dashboard*, quando o banco de dados não apresentar um número mínimo de consultas.
- c) Por meio de informações de logs do banco de dados.
- d) Por meio do *Dashboard*, quando o banco de dados não apresentar o número acessos.
- e) Por meio do *Dashboard*, quando o banco de dados estiver com 100% de uso de memória.

Referências

DEVMEDIA. **Artigo da SQL Magazine 33** – Teste de Capacitação do BD com o JMeter. Disponível em: <<https://www.devmedia.com.br/artigo-da-sql-magazine-33-teste-de-capacitacao-do-bd-com-o-jmeter/6696>>. Acesso em: 2 jun. 2018.

ELMASRI, R. **Sistemas de banco de dados**. 6. ed. São Paulo: Pearson Education, 2005.

GUILHERME LIMA. **JMeter** – Teste de performance em banco de dados. 9 abr. 2017. Disponível em: <<https://www.youtube.com/watch?v=Jp3jwBCE4-s>>. Acesso em: 30 abr. 2018.

HEUSER, C.A. **Projeto de banco de dados**. 6. ed., Porto Alegre: Bookman, 2009.

MEDEIROS, L.F. **Banco de dados** – princípios e práticas. São Paulo: Intersaberes, 2013.

NISHIMURA, R. **Banco de dados II**. São Paulo: Pearson Education, 2009.

ORACLE. **Configurando o serviço**. [S.d.; s.p.]. Disponível em: <<http://download.nust.na/pub6/mysql/doc/administrator/pt/mysql-administrator-service-control-configure-service.html>>. Acesso em: 15 jun. 2018.

ORACLE. **Manual de Referência do MySQL**. Disponível em: <<https://dev.mysql.com/doc/>>. Acesso em: 1 maio 2018.

ORACLE. **MySQL Connector/Net Developer Guide**. 11 jun. 2018. Disponível em: <<https://dev.mysql.com/doc/connector-net/en/>>. Acesso em: 15 jun. 2018.

ORACLE. **MySQL Connector/ODBC Developer Guide**. 11 jun. 2018. Disponível em: <<https://dev.mysql.com/doc/connector-odbc/en/>>. Acesso em: 15 jun. 2018.

ORACLE. **MySQL for Excel**. 11 jun. 2018. Disponível em: <<https://dev.mysql.com/doc/mysql-for-excel/en/>>. Acesso em: 15 jun. 2018.

ORACLE. **MySQL for Visual Studio**. 11 jun. 2018. Disponível em: <<https://dev.mysql.com/doc/visual-studio/en/>>. Acesso em: 15 jun. 2018.

ORACLE. **MySQL Workbench**. 14 jun. 2018. Disponível em: <<https://dev.mysql.com/doc/workbench/en/>>. Acesso em: 15 jun. 2018.

Ramarkrishnan, R. **Sistemas de gerenciamento de banco de dados**. 3. ed. Nova Iorque: McGraw Hill, 2011.

VICCI, C. **Banco de dados**. São Paulo: Pearson Education, 2009.

Segurança de banco de dados

Convite ao estudo

Olá, aluno! Você já reparou que é comum ouvir falar em jornais ou notícias na televisão que empresas têm informações acessadas por pessoas de fora, ou que modificam estas informações para benefício próprio? Pois bem, esse tipo de situação pode estar mais perto de você do que imagina. Algumas pessoas acabam se beneficiando dos privilégios concedidos a elas, passando as informações para frente ou adulterando dados.

Nesta unidade você aprenderá como evitar estes tipos de situações em um banco de dados e, como resultado da aprendizagem, você estará apto a analisar e aplicar o gerenciamento de segurança em um banco de dados.

Otávio é um empresário no ramo de transportes, no interior de São Paulo, onde trabalha com entrega de produtos de diversos tipos e tamanhos, enviando produtos para todo o Brasil. A gestão de entregas é feita por um sistema que ele comprou de terceiros: um aplicativo fechado, sem atualizações e sem fonte do sistema, tendo somente o acesso ao banco de dados do sistema, e ele é o único que tem as informações de acesso. Porém, agora, com as novas leis fiscais, o sistema não está atendendo mais às necessidades do dia a dia de sua transportadora e, com medo de ser fiscalizado, ele contratou desenvolvedores para que fizessem um aplicativo que pudesse acessar as informações do banco de dados e conseguisse exibir ou atualizar informações fiscais obrigatórias.

Para tratar esta situação com eficiência você irá estudar:

Na Seção 2.1, o conceito e a criação de logins e usuários, sabendo quais são suas vantagens e porque devemos ficar atentos à criação desses itens.

Na Seção 2.2, o conceito e criação de privilégios, sabendo para que servem e como usá-los no dia a dia, bem como quais são os níveis de privilégios em um SGBD.

Na Seção 2.3, como funciona o gerenciamento de privilégios, como conceder ou revogar um privilégio já dado para um usuário e como limitar o acesso de um usuário ao banco de dados.

Em resumo, com seu empenho nos estudos e sua participação durante as aulas, esta unidade será bem produtiva. É importante que você faça os exercícios para fixar o conteúdo e compreender suas dificuldades. Então, vamos seguir nossa leitura.

Bom estudo!

Seção 2.1

Introdução à segurança em banco de dados

Diálogo aberto

Olá aluno, vamos iniciar a nossa segunda unidade. A partir dela, vamos entender sobre a segurança do banco de dados. Agora que você já tem noção do conceito do que é um banco de dados e de como manipulá-lo, nada mais sábio do que aprimorarmos nossos estudos no quesito segurança. Entenderemos o que deve ser protegido no banco de dados e quais são suas principais fragilidades, bem como quais são os objetivos principais para um banco de dados seguro.

Com nossos esforços, vamos poder contextualizar o uso de login de usuários para o Otávio, um empresário no ramo de transportes no interior de São Paulo que busca um novo sistema de acesso às informações do banco de dados, que consiga exibir e atualizar as informações fiscais e obrigatórias. Ele precisa, neste primeiro momento, entender a importância de criar um usuário por desenvolvedor ou por grupo de desenvolvedores de acordo com o que precisarem, como, por exemplo, somente consultar dados ou manipular dados. É necessário apresentar ao Otávio, em um relatório, as vantagens de criar usuários para cada desenvolvedor e como isso seria vantajoso para a segurança de seu banco de dados.

Para auxiliá-lo neste desafio, vamos conhecer o que é a segurança de um banco de dados e como podemos trabalhar com ela. Com foco e estudo, vamos conseguir dar um passo a diante e resolver esta situação problema.

Bom estudo!

Não pode faltar

Cada vez mais, vemos aspectos de nossas vidas armazenados em banco de dados, e informações valiosíssimas a nosso respeito espalhadas pela internet. Consequentemente, um administrador de

dados precisa sempre estar atento à segurança dos dados para que não comprometa a integridade do banco de dados.

Segundo Ramakrishnan (2011), existem três objetivos de segurança ao se projetar um sistema de banco de dados seguro:

1. **Sigilo:** as informações não devem ser reveladas a usuários não autorizados. Por exemplo, um aluno não pode examinar as notas de outros alunos.
2. **Integridade:** apenas usuários autorizados podem modificar dados. Por exemplo, os alunos devem ver suas notas, embora (obviamente) não possam modificá-las.
3. **Disponibilidade:** os usuários autorizados não devem ter acesso negado. Por exemplo, um professor que queira alterar uma nota deve poder fazer isso quando desejar.

Para atingir esses objetivos, deve-se desenvolver uma política de segurança clara e consistente que descreva quais medidas de segurança devem ser impostas.

A política de segurança de banco de dados consistiria em:

- Monitorar acessos inadequados ou excessivos ao banco de dados.
- Avaliar direitos de permissões de acesso de usuários com privilégio de administrador de dados e usuários de aplicação.
- Auditar mudanças e configurações dos próprios SGBDs.
- Gerenciar privilégios de usuários para impedir o acesso a campos de dados sensíveis.
- Avaliar a vulnerabilidade e conformidade.
- Identificar dados sensíveis.

Em particular, devemos determinar quais informações devem ser protegidas e quais usuários podem acessar estas informações, em seguida, os mecanismos de segurança do banco de dados e do sistema operacional, assim como mecanismos externos, como a segurança no acesso aos prédios, devem ser utilizados para impor a política. Enfatizamos que medidas de segurança devem ser tomadas em vários níveis.

Brechas de segurança no sistema operacional ou em conexões de rede podem inutilizar os mecanismos de segurança do banco de dados, por exemplo:

- Permitir que um intruso se conecte como administrador no banco de dados, com todos os consequentes direitos de acesso ao SGBD.

- Fatores humanos também podem influenciar nessas brechas de segurança. Um exemplo seria um usuário escolher uma senha fácil de adivinhar ou um usuário autorizado a visualizar os dados sigilosos usar tal autorização de modo impróprio. O fator humano é responsável por uma grande porcentagem das brechas de segurança.

Isso mostra como é importante autenticar um usuário no sistema de banco de dados, afinal, podemos impor uma política de segura que permita ao usuário João ler uma tabela **Salario**, e ao usuário Pedro gravar nessa mesma tabela, evitando que o usuário João acesse a tabela com o usuário Pedro e tendo as permissões para gravar na tabela. Isso garante que o usuário está se comunicando com um sistema legítimo e não um aplicativo espúrio, destinado a roubar informações sigilosas, como um número de cartão de crédito.



Assimile

Ao realizar uma análise de criação de usuários ou de login, é preciso ter muito cuidado com a que o usuário terá acesso ou o que ele poderá fazer com o banco de dados.

Sempre que possível, verifique o log de acessos para validar se o usuário não está tentando realizar uma ação indevida.

A questão da segurança em banco de dados não se restringe apenas ao que é possível gerenciar via SGBD. Diferentes precauções devem ser tomadas no SGBD, na especificação das consultas, na programação das aplicações, no hardware e na rede relacionada ao uso do SGBD.

Questões éticas e legais, assim como políticas de uso da informação estabelecidas pelas organizações, também precisam ser consideradas.

As principais preocupações relacionadas à segurança na área de banco de dados estão relacionadas a evitar:

- Perda de integridade.

- Perda de disponibilidade.
- Perda de confidencialidade.

Todas essas perdas são danosas para um usuário de banco de dados e precisam ser observadas no momento de estabelecer as políticas de segurança.



Assimile

Questões éticas envolvem conceitos relacionados a sistemas e administradores. Estes conceitos são:

- Escolhas éticas - são escolhas individuais, decisões tomadas por indivíduos responsáveis pelos seus atos.
- Responsabilidade - é aceitar os deveres e obrigações decorrentes das decisões tomadas.
- Obrigação de Indenizar - amplia o conceito de responsabilidade para as leis, permitindo que indivíduos recuperem danos causados a eles por pessoas, ou sistemas
- Prestação de contas - característica dos sistemas que identificam quem realizou uma ação.
- Devido processo legal - caracteriza a sociedade governada por leis conhecidas e entendidas, com a possibilidade de apelar as autoridades superiores para serem cumpridas.

Perda de integridade

A integridade do banco de dados diz respeito à necessidade de proteger as informações contra modificações inapropriadas: criação, modificação ou exclusão de dados.

Essa perda pode ser decorrente de ações intencionais ou acidentais que podem permitir fraude ou tomada de decisão incorreta. Segundo Elmasri (2005), para garantir a integridade dos dados ou dos valores que estão armazenados no banco de dados, precisamos lembrar um pouco do que aprendemos na primeira unidade e precisamos ter um projeto de banco de dados com qualidade para podermos garantir a integridade das informações.

Perda de disponibilidade

A indisponibilidade se refere à ocorrência de problemas para o uso do sistema e de seus recursos (funcionalidade e dados). Sofrem com a indisponibilidade tanto os usuários humanos quanto outros sistemas (programas) que acessam o banco de dados (ELMASRI, 2005). Um exemplo seria um *e-commerce* que, durante uma promoção, muitos usuários tentam realizar suas compras, porém o banco de dados está limitado a múltiplas conexões e, conseqüentemente, alguns usuários sofrerão lentidão ou até mesmo indisponibilidade do site.

Perda de confidencialidade

Segundo Elmasri (2005), essa perda está relacionada com o risco de disponibilização de dados ou de informação de forma não autorizada, e pode incorrer em violação de direitos legais. Um exemplo seria o *e-commerce* que precisa validar informações do cartão de crédito para concretizar a venda e, nesse caminho, as informações estarem acessíveis para usuários não autorizados, podendo copiar os dados dos compradores e usando-os para benefício próprio.

Papel do administrador do banco de dados (DBA – *Data Base Administrator*)

Segundo Elmasri (2005), o DBA possui uma conta com direitos de super-usuário que proporciona a responsabilidade de zelar pela segurança no SGBD e nos bancos de dados associados. Algumas ações básicas do DBA:

- Criar contas: criar uma nova conta e senha para um usuário ou um grupo de usuários para habilitar o acesso ao banco de dados.
- Concessão de privilégios: permite que o DBA conceda certos privilégios a determinadas contas.

- **Revogação de privilégios:** permite que o DBA revogue (cancele) certos privilégios que foram concedidos anteriormente a determinadas contas.
- **Atribuição de nível de segurança:** consiste em atribuir as contas de usuários de nível de classificação de segurança adequado.

É o papel do DBA garantir que não ocorram as perdas mencionadas acima, e também tomar medidas de segurança remediando futuros problemas.

Algumas das medidas principais de segurança são:

- **Controle de acesso:** é o controle que impõe restrições e regras, por meio de contas de perfis de usuários, quando acessamos o banco de dados.
- **Controle de inferência:** no banco de dados existe um mecanismo de segurança que atua protegendo informações estatísticas de um grupo.
- **Controle de fluxo:** o controle de fluxo é responsável por regular a distribuição de informações entre objetos acessíveis. É um mecanismo que evita que estas informações fluam por canais que violem a política de segurança.
- **Criptografia:** medida de controle final, usada para proteger dados sigilosos que são trafegados por algum tipo de comunicação. Também é usada para fornecer uma proteção extra para o banco de dados, evitando acessos por usuários não autorizados.



Exemplificando

No início de 2017, hackers invadiram o banco de dados do Snapchat e divulgaram cerca de 4,5 milhões de nomes e números parciais de telefones e usuários. Nenhuma imagem compartilhada pelo serviço foi liberada, a intenção dos hackers era pressionar os desenvolvedores do Snapchat a resolver os problemas de segurança.

Sempre que uma pessoa ou um grupo de pessoas precisa acessar um sistema de banco de dados, o indivíduo ou o grupo deve primeiro requisitar uma conta de usuário, então o DBA criará

um novo número de conta e senha para o usuário, se houver uma necessidade legítima de acessar o banco de dados. O usuário deve realizar a conexão (log in) no banco de dados, entrando com o número de conta e senha sempre que o acesso ao banco de dados for necessário. O SGBD verifica se o número de conta e senha são validados, validando se o usuário tem permissão de usar e acessar o banco de dados.

O sistema de banco de dados também deve manter as informações de todas as operações que são aplicadas por um usuário durante cada sessão de conexão (*login session*), que consiste em fazer uma sequência de interações que um usuário realiza desde o momento de conexão (login) até o momento de desconexão (logout). Quando um usuário se conecta, o SGBD registra o número da sua conta e o associa ao terminal a partir do qual ele se conectou.



Refleta

Vale a pena criar um usuário ou login para cada desenvolvedor ou para vários grupos? Imaginemos que uma empresa de desenvolvimento de softwares tenha diversos desenvolvedores e que cada um é responsável por uma determinada parte do sistema. Uns desenvolvem relatórios, outros a parte visual e outros trabalham com movimentações. Para cada um deles, é compreensível criar um tipo de perfil: o desenvolvedor que cuida de relatórios não precisa ter privilégios para criar ou deletar informações, e o responsável pela parte visual teria somente acesso de inclusão parcial em algumas tabelas específicas.

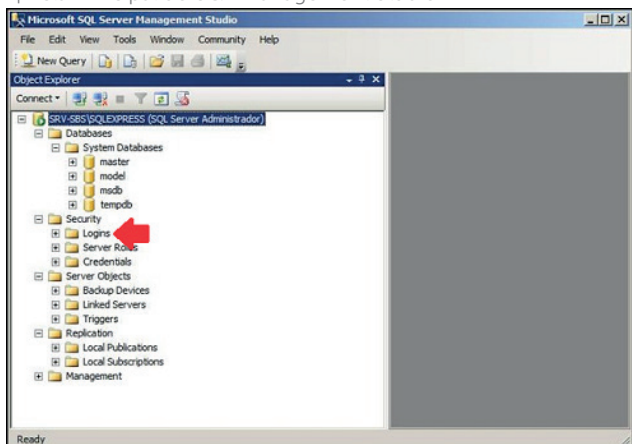
Logins

Um login é uma entidade de segurança que pode ser autenticada por um sistema seguro. Você pode criar um login com base em uma entidade de segurança do Windows, que seria o próprio usuário já cadastrado para acessar o Windows, ou então criar um login exclusivo do SQL Server.

Para criar um login do SQL Server usando o Management Studio

1. No pesquisador de objetos (Figura 2.1), expanda a pasta da instância de servidor no qual deseja criar o novo login.

Figura 2.1 | Tela Principal do SQL Management Studio

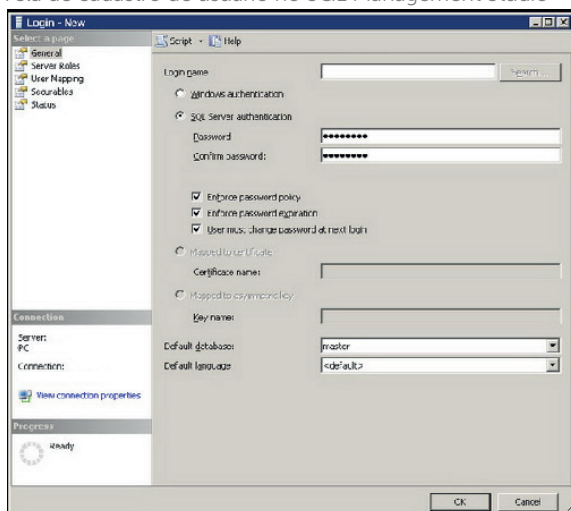


Fonte: captura de tela do Microsoft SQL Server Management Studio.

2. Clique com o botão direito do mouse e, na pasta segurança, vá até novo e selecione login.

3. Na caixa de diálogo login- novo (Figura 2.2), na página geral, insira o nome de um usuário na caixa nome de login (*Login name*). Como alternativa, clique em pesquisar e, para abrir a caixa de diálogo, selecione usuário ou grupo.

Figura 2.4 | Tela de cadastro de usuário no SQL Management Studio



Fonte: captura de tela do Microsoft SQL Server Management Studio.

Para criar um login utilizando a Autenticação do Windows

1. No pesquisador de objeto, conecte a uma nova instância do mecanismo de banco de dados.
2. Na barra padrão do sistema, selecione nova consulta.
3. Execute o seguinte comando:

```
Create      login      [<domainName>\<loginName>]  
From Windows;  
GO
```

Com o comando acima, estamos executando uma instrução SQL de criação de login no Windows. O <domainName> seria o nome do computador ou servidor, e o <loginName> seria o nome do usuário usado para conectar no Windows.

Aprendemos aqui vários pontos importantes sobre a segurança de banco de dados, entre eles:

- Objetivos de segurança ao projetar um sistema de dados seguro.
- A política de segurança de banco de dados.
- Como evitar brechas de segurança, tais como perda de integridade, disponibilidade ou confidencialidade.
- O papel do administrador de banco de dados.
- Como criar logins usando o SQL Management Studio.

Sem medo de errar

Conforme abordamos no início dessa seção, nosso desafio era contextualizar o uso de login de usuários para o Otávio, um empresário no ramo de transportes, do interior de São Paulo, que busca um novo sistema para acessar as informações do banco de dados e conseguir exibir e atualizar as informações obrigatórias. Ele precisa entender a importância de criar um usuário por desenvolvedor, ou por um grupo de desenvolvedores, de acordo com que precisarem. Para isso, você deve apresentar para o Otávio

um relatório com as vantagens de se criar um usuário para cada desenvolvedor para a segurança de seu banco de dados.

Para que Otávio entenda as vantagens de criar um login para cada usuário, teremos que exemplificar com uma situação cotidiana. Imaginemos que uma empresa de automóveis está prestes a comercializar um novo carro no mercado, porém não pode divulgar as informações do veículo antes do lançamento. Essas informações são privilegiadas e somente supervisores e gerência têm acesso a elas, por meio de um usuário único. Um desses supervisores resolve deixar a empresa para trabalhar na concorrência e leva com ele as informações do novo carro que será lançado.

Nessa situação, a concorrência terá as informações relevantes do novo carro e a empresa fabricante nem sequer saberá como vazaram as informações, porque só há um usuário para os servidores, e de nada adiantaria visualizar o log de acessos. Esse exemplo mostra um problema que poderia ter sido resolvido no início, mas não notaram essa brecha, que acabou causando esta dor de cabeça para empresa.

A partir desse exemplo, fica claro que Otávio precisa criar um usuário para cada desenvolvedor, assim ele evitará que informações de seus sistemas acabem sendo acessadas por pessoas mal intencionadas. Também é recomendado apresentar um relatório para o Otávio dos desenvolvedores que só precisam consultar o banco de dados e dos que precisam manipular as informações, assim, cada um teria seu perfil próprio de acesso, dando uma segurança a mais ao banco de dados e mantendo a integridade de suas informações.

Avançando na prática

Informações fantasmas

Descrição da situação-problema

Alexandre, um dos 50 funcionários de uma empresa de consultoria de seguros, está tendo problemas com seus clientes por

que alguns boletos de seguros não estão chegando a eles, ou estão sendo enviados com informações faltantes, como, por exemplo, o valor do seguro.

Após avisar a empresa, foi constatado que alguns funcionários estão com o mesmo problema, e que isso ocorreu após uma integração com um sistema de um fornecedor. Após relatar o problema, foi afirmado que o erro de informações era de uma tabela no banco de dados com o mesmo nome da tabela do fornecedor.

Nesse caso, que tipo de solução teria que ser tomada para que o sistema da consultoria de seguros possa voltar a funcionar corretamente?

Resolução da situação-problema

Primeiramente, é preciso analisar o perfil de acesso do cliente no sistema de banco de dados, se ele precisa modificar informações ou somente visualizar informações. Feita esta análise, se um cliente deveria ter acesso somente para visualizar, mas está com acesso para modificar, o correto seria revogar seu acesso e alterá-lo, para que ele somente visualize as informações, do contrário, seria manter o mesmo acesso. E, quando o cliente estiver com acesso somente para visualizar e precisa modificar, é preciso revogar o acesso também, pois podem existir casos de clientes que não podem visualizar informações de outras tabelas, mas podem acessar somente as que eles mesmos tem acesso.

Faça valer a pena

1. A questão da segurança em banco de dados não se restringe apenas ao que é possível gerenciar por meio do SGBD. Diferentes precauções devem ser tomadas no SGBD, na especificação das consultas, na programação das aplicações, no hardware e na rede relacionada ao uso do SGBD. Questões éticas e legais, assim como políticas de uso da informação estabelecidas pelas organizações, também precisam ser consideradas.

Quais são as principais preocupações relacionadas à segurança do banco de dados?

- a) Integridade, disponibilidade e confidencialidade.
- b) Integridade, disponibilidade e segurança.
- c) Segurança, disponibilidade e confidencialidade.
- d) Segurança, criptografia e confidencialidade.
- e) Disponibilidade e confidencialidade.

2. O DBA (Administrador de banco de dados) possui uma conta com direitos de super usuário, a qual atribui a ele a responsabilidade de zelar pela segurança no SGBD e dos bancos de dados associados, assim, evitando a perda de informações em relação à segurança do banco de dados.

Selecione quais ações pertencem ao DBA:

- a) Gerenciar contas, desenvolver sistema e inserir informações no banco de dados.
- b) Controlar o fluxo de dados e desenvolver novas estruturas do sistema
- c) Somente monitorar acesso.
- d) Gerenciar e realizar testes de softwares e fluxo de dados.
- e) Criar contas, conceder e revogar privilégios, e atribuir nível de segurança.

3. Para acessar um banco de dados, é preciso que o usuário tenha um login no banco de dados ou um usuário, e tanto o login como a senha precisam ter um acesso mínimo para logar no banco de dados e, consequentemente, acessar as devidas informações necessárias.

Com base no texto, qual seria a diferença fundamental entre login e usuário de acesso?

- a) Logins são criados no nível da instancia do banco de dados do servidor, enquanto usuário é criado no nível do banco de dados.
- b) Logins são criados no nível do banco de dados, e o usuário cria uma instância do banco de dados do servidor.
- c) Logins dependem de usuário criado para acessar banco de dados e usuário não depende de login para acessar.
- d) Logins não acessam o banco de dados e usuários acessam.
- e) Usuários não acessam o banco de dados e logins acessam.

Seção 2.2

Privilégios de acesso

Diálogo aberto

Olá, aluno! Vamos a mais uma etapa de nosso aprendizado!

Na seção anterior falamos sobre o conceito e o uso de logins, diferenciando-os e suas formas de uso adequadas no nosso dia a dia, aprendendo também como gerenciá-los de maneira segura.

Imagine uma empresa de grande porte, em que haja diversos setores usando um ou mais sistemas distribuídos, interligados por softwares ou pela própria rede. Independentemente da quantidade de usuários existente na empresa, teremos sempre alguns com mais ou menos perfis de acessos que outros. Alguns com perfis de administradores, outros com perfis de usuários internos e outros usuários finais.

Mas, afinal, qual é a diferença entre esses tipos de usuários? A resposta é bem simples: o que diferencia são seus privilégios de acesso. Vamos supor que um desses usuários consiga ter acesso a um nível de administrador, ele poderia limpar tabelas ou acessar informações que não cabem a ele. Isso seria um problema para os DBAs.

Otávio está empenhado em terminar logo o sistema para não correr o risco de ser fiscalizado e para que tudo ocorra de maneira mais breve possível, sem que prejudique a segurança de seu banco de dados. Ele precisa decidir, após ter criado acessos aos seus desenvolvedores, quais tipos de acessos eles podem ter, se precisam acessar somente informações ou manipular dados. Sua tarefa é direcioná-lo e apresentar um relatório claro de como ele deve definir quais tipos de privilégios cada desenvolvedor deve ter para acessar seu banco de dados. Pensando no que seria melhor, deveria passar o um privilégio para um único desenvolvedor ou para um grupo de desenvolvedores? Como proceder de forma a garantir que a integridade do banco de dados continue intacta?

Nesta seção, você vai estudar quais tipos de privilégios um usuário pode ter, saber definir seus níveis e entender o conceito

de níveis de privilégio. O objetivo é entender e aplicar a análise de gerenciamento de acordo com o perfil do usuário.

Boa Leitura!

Não pode faltar

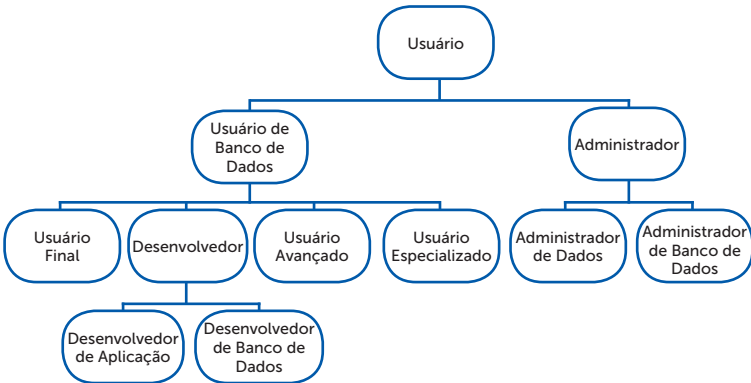
Um banco de dados de uma empresa contém muitas informações e normalmente, vários grupos de usuários (ELMASRI, 2005). A maioria dos usuários só precisa de uma pequena parte do banco de dados para realizar suas tarefas. Permitir aos usuários acesso irrestrito a todos os dados pode ser indesejável, e um SGBD deve fornecer mecanismos para controlar o acesso aos dados.

Pires (2009) afirma que, para entendermos um pouco mais sobre os grupos de usuários, precisamos dividi-los em dois grupos básicos, quais sejam:

- **Administradores de banco de dados:** é aquele que interage diretamente com o banco de dados, responsável pela segurança e organização.
- **Usuários de banco de dados:** interage direta ou indiretamente com o banco de dados, de acordo com os privilégios concedidos pelos Administradores de banco de dados.

Com base nessas duas categorias (PIRES, 2009) estabeleceu um diagrama que organiza e interage com o banco de dados, como mostra na Figura 2.3.

Figura 2.3 | Tipos de usuários



Fonte: Pires (2009, p. 12).



Se uma empresa concede praticamente todos os privilégios de administrador a todos os usuários, faz sentido haver perfis diferentes? Isso seria uma prática segura?

Elmasri (2005) afirma que podemos trabalhar com dois tipos de privilégios: privilégio por controle de acesso discricionário, baseado na concessão e na revogação de privilégios, e o privilégio por controle de acesso obrigatório, baseado em papéis para segurança multinível.

Vamos analisar estes dois tipos de privilégios:

- Privilégio por controle de acesso discricionário, baseado na concessão e na revogação de privilégios (ELMASRI, 2005): a principal ideia é incluir sentenças na linguagem de consulta que permitam ao DBA e aos usuários selecionados concederem e revogarem privilégios. O privilégio discricionário é usado, a grosso modo, para referenciar uma conta de usuário ou grupo de usuários.

Informalmente existem dois níveis para atribuição de privilégios para o uso do sistema de banco de dado. O primeiro é o nível de conta, que se aplica às capacidades providas para a própria conta e pode incluir:

- O privilégio **Create Table** ou **Create Schema**, para criar uma relação base.
- O privilégio **Create Alter**, que serve para aplicar mudanças no esquema, como adicionar ou remover atributos das relações.
- O privilégio **Drop**, usado para excluir relações ou visões.
- O privilégio **Modify**, que serve para incluir, excluir ou atualizar tabelas.
- E o privilégio **Select**, que serve para recuperar e visualizar informações do banco de dados.

O segundo é o nível da relação (Tabela), que se aplica a relações base ou relações virtuais (visões) (ELMASRI, 2005). Em SQL, os seguintes tipos de privilégio podem ser concedidos para cada relação R individual:

- O privilégio **Select** (*retrieval* ou *read*) sobre R proporciona à conta o privilégio de recuperação. Em SQL, dá à conta o privilégio de usar a sentença **Select** para recuperar tabelas a partir de R.

- O privilégio **Modify** sobre R proporciona a capacidade de modificar tabelas de R. Em SQL, esse privilégio é mais avançada, dividido entre os privilégios **Insert**, **Update** e **Delete** para aplicar em R os comandos SQL correspondentes.

- O Privilégio de **References** sobre R proporciona à conta a capacidade de fazer referência à relação R quando estiver especificando restrições de integridade. Esse privilégio também pode ser restrito a atributos específicos de R.

- Privilégio por controle de acesso obrigatório baseado em papéis para segurança multinível: é um método tudo ou nada. Ou seja, ou um usuário possui um certo privilégio ou não o possui (ELMASRI, 2005). Essa abordagem, conhecida por controle de acesso obrigatório, tipicamente deveria ser combinada com os mecanismos de controle de acesso discricionários. É importante observar que, atualmente, a maioria dos SGBDs comerciais oferece mecanismos somente para controle de acesso discricionário.



O Privilégio baseado em Papeis é uma tecnologia comprovada para o gerenciamento da imposição de segurança em um sistema de grande escala que abrangem toda a organização. Sua noção básica é que as permissões são associadas a papéis, e aos usuários são atribuídos papéis adequais. Os papeis podem ser criados por meio do uso dos comandos **Create Role** e **Destroy Role**. Então, os comandos **Grant** e **Revoke** podem ser utilizados para atribuir e revogar privilégios dos papéis (ELMASRI, 2005, p. 536).

Ele é uma alternativa viável para os tradicionais controles de acesso discricionários e obrigatórios, assegurando que apenas usuários autorizados tenham acesso a certos dados ou recursos.



Atualmente, os principais bancos de dados oferecem mecanismos somente para controle de acesso discricionário. Outro tipo de controle de acesso seria por mecanismo de segurança obrigatório, que é usado para conceder acesso ao usuário por meio da classificação dos usuários baseada nas funções que eles desempenham no banco de dados.

Os privilégios garantem a segurança e a integridade dos dados, bem como a responsabilidade de cada usuário sobre seus dados específicos (MARCHADO, 2014).

O Comando **Grant** (Garantir) é usado quando uma tabela ou view é criada. O nome do usuário que a criou é anexado internamente ao nome da tabela. Por exemplo, se a tabela **Cliente** foi criada pelo usuário Pedro, então, internamente, ela será conhecida como **Pedro.Cliente**. O criador da tabela ou view tem total privilégio sobre ela, podendo disponibilizar qualquer privilégio para outros usuários pelo comando **Grant**, como pode ser visto no Quadro 2.1.

Quadro 2.1 | Elementos para permissão e suas funcionalidades

Elementos e suas funcionalidades	
Grant	Instrução padrão para concessão de privilégios.
ALL Privilégios	Privilégios que serão concedidos.
On	Objeto que receberá privilégios.
To	Usuário que receberá privilégios
With Admin Option	Receptor do privilégio de sistema que pode conceder o privilégio recebido.
With Grant Option	Receptor do privilégio de objeto que pode conceder o privilégio recebido

Fonte: elaborado pelo autor.

Sintaxe Básica do comando

Grant all

On Clientes To Aluno with Grant Option;

De acordo com essa sintaxe, a palavra reservada ALL é usada quando qualquer privilégio é aplicável ao objeto, senão, especificamos o privilégio que está sendo dado (**Select**, **Insert**, **Update** etc.). A cláusula ON especifica a tabela ou a view, bem como suas colunas, para as quais está sendo dado o privilégio. Esse comando só pode ser utilizado em uma tabela ou view por vez.

Para continuar nosso estudo, suponha que o a conta A1 tenha privilégios nas duas tabelas listadas na Figura 2.4:

Figura 2.4 | Esquemas para as duas relações Funcionário e Departamento

FUNCIONARIO

<u>CODIGO</u>	NOME	DATANASC	ENDereco	SEXO	IDADE	SALARIO
---------------	------	----------	----------	------	-------	---------

DEPARTAMENTO

<u>CODIGODEPTO</u>	NOMEDEPTO	NUMFUNCIONARIOS
--------------------	-----------	-----------------

Fonte: elaborada pelo autor.

A seguir, suponha que a conta A1 queira conceder à conta A2 o privilégio somente para incluir ou excluir tabelas em ambas as relações. Entretanto, A1 não quer que A2 seja capaz de propagar esses privilégios para contas adicionais. Para essa situação, teríamos o seguinte comando:

```
Grant Insert, delete
```

```
On Funcionarios, Departamento To A2;
```

Observe que, nesse exemplo, a conta A1 vem de uma relação automática, possuindo a **Grant Option**, permitindo-lhe conceder privilégios sobre a relação para outras contas. Entretanto, à conta A2 não foi dado o **Grant Option**, assim, essa conta não pode conceder privilégios para outras contas.

Para que uma conta passe seus privilégios de acesso de uma tabela para outra a sintaxe seria a seguinte:

```
Grant Select
```

```
On Funcionarios, Departamento To A2
```

```
With Grant Option;
```

Da mesma forma que o criador da tabela pode garantir (**Grant**) privilégios de acesso aos outros usuários, com o comando **Revoke**, pode revogar esses privilégios por meio do comando.

Suponhamos que a conta A1 não queira que a conta A2 delete mais informações da tabela Funcionário. Para essa situação, teríamos o seguinte comando:

```
Revoke delete
On Funcionarios To A2;
```

O Quadro 2.2, a seguir, apresenta o comando Revoke.

Quadro 2.2 | Elementos para revogar e suas funcionalidades

Elementos e suas funcionalidades	
Revoke	Instrução padrão para revogar privilégios recebidos.
Privilégios	Privilégios que serão revogados.
On	Objeto que terá privilégios revogados.
From	Usuário que terá privilégios revogados.

Fonte: elaborado pelo autor.

Continuando o exemplo, o usuário da conta A1 verificou que o usuário da conta A2 também não precisa modificar informações da tabela Funcionários. Para essa situação, teríamos dois exemplos:

```
Revoke delete
On Funcionarios To A2;
```

```
Revoke update
On Funcionarios To A2;
```

No exemplo acima, foram executados dois comandos que, no final funcional, funcionarão de uma única forma: removendo permissões de atualizar e deletar para a conta A2. Agora, temos também uma forma de otimizarmos esses dois comandos, transformando-os em um só, da seguinte forma:

```
Revoke delete, update
On Funcionarios To A2;
```



Para saber a extensão do que é possível fazer em termos de atribuição de privilégio, recomendamos estudar a fundo os próprios comandos do SQL que são passíveis de privilégios.

DUARTE, E. **Gerenciamento de usuários e controle de acessos do MySQL**. [S.l.]: DEVMEDIA, 2006. Disponível em: <<https://www.devmedia.com.br/gerenciamento-de-usuarios-e-controle-de-acessos-do-mysql/1898>>. Acesso em: 3 maio 2018.

Quando o comando **Grant** é usado, é necessário informar quais privilégios ele irá conceder, se serão todos (ALL) ou específicos. O quadro abaixo mostra os privilégios que podem ser especificados:

Quadro 2.3 | Tabela de Tipos de permissões a serem concedidas ou revogadas

Select	O direito de acessar (ler) todas as colunas da tabela especificada como objeto, incluindo colunas adicionadas posteriormente por meio do comando Alter Table .
Insert	O direito de inserir linhas com valores (não-nulos ou não-padrão) na mesma coluna nomeada da tabela nomeada como objeto. Se esse direito precisar ser concedido com relação a todas as colunas, incluindo colunas que possam ser adicionadas posteriormente, podemos usar simplesmente Insert .
Update	Os privilégios Update são semelhantes.
Delete	O direito de excluir linhas da tabela nomeada como objeto.
References	O direito de definir chaves estrangeiras (em outras tabelas) que se refiram à coluna especificada da tabela objeto, References sem um nome de coluna especificado denota esse direito com relação a todas as colunas, incluindo as que forem adicionadas posteriormente.

Fonte: elaborado pelo autor.



O site Monster.com é o maior banco de dados de empregos da internet. Em agosto de 2007, o site passou por vazamentos de dados, sendo que 94 milhões de clientes tiveram seus dados vazados e algumas contas eram contas internacionais. Isso levou ao site a adotar uma nova política de segurança. Uma delas é que o cliente coloque somente as informações pedidas no cadastro e que fiquem atentos aos e-mails enviados pela empresa, evitando, desta forma, novos ataques.

ARRUDA, F. Os 9 maiores roubos de dados da internet. **Tecmundo**. 11 jul. 2012. Disponível em: <<https://www.tecmundo.com.br/seguranca/26476-os-9-maiores-roubos-de-dados-da-internet.htm>>. Acesso em : 27 maio 2018.

Em conjunto com os comandos **Grant** e **Revoke**, as visões são componentes importantes dos mecanismos de segurança fornecidos por um banco dados. Definindo visões nas tabelas base, podemos apresentar as informações necessárias para alguns usuários, enquanto ocultamos outras informações a que eles não devem ter acesso (RAMARKRISHNAN, 2011). Por exemplo, considere a seguinte definição da visão:

```
Create View Comissao_Vendas (Vendedor, comissao)  
AS Select f.Funcionario AS vendedor,  
        SUM(V.Comissao) AS comissao  
From Vendas v, Funcionarios f  
Where v.cod_funcionario = f.codigo  
And v.Data = CURRENT_DATE
```

Nessa situação, o usuário pode acessar **Comissao_Vendas** mas não a tabela vendas ou funcionários.

Com base no que aprendemos, você será capaz de criar e gerenciar privilégios de acordo com o perfil de cada usuário, sabendo qual tipo de permissão ele precisa ter ou qual permissão ele tem e o que precisa ser revogado.

Sem medo de errar

Conforme abordamos no início desta seção, o desafio era apresentar para o Otávio um relatório claro sobre como ele deve definir quais tipos de privilégios os desenvolvedores da empresa dele devem ter para garantir a integridade do banco de dados. Primeiramente, vamos dividir a equipe de desenvolvedores em grupo A, composto pelos desenvolvedores que trabalham somente com relatórios e consultas, e grupo B, de desenvolvedores que trabalham com desenvolvimento dos aplicativos.

O grupo A não precisa de permissões para alterar, incluir ou excluir registros, mas somente da permissão para visualizar. Com essa definição, usaremos o seguinte comando como exemplo para uma tabela de Produto:

```
Grant Select
```

```
On Produto To A;
```

Nessa situação, não incluímos o comando **With Grant Option** pois essa conta não precisa ter acesso para distribuir outros acessos.

O grupo B precisa de mais privilégios que o grupo A. Neste caso, o comando de exemplo seria o seguinte, usando a mesma tabela Produto:

```
Grant Select, Insert, Delete, Update
```

```
On Produto To B;
```

Nesse caso, também é possível usar o comando **ALL**, já que precisamos de todos os privilégios.

```
Grant ALL
```

```
On Produto To B;
```

Depois de executar estes comandos de permissão, apresente ao Otávio como funcionaria esta nova divisão, mostrando que agora os Desenvolvedores do grupo A podem somente visualizar a tabela Produto, e o Grupo B tem total acesso na tabela Produto.

Removendo privilégios desnecessários

Descrição da situação-problema

Você foi contratado por uma empresa de manutenção de impressoras em uma pequena cidade, onde existem somente duas empresas que trabalham com manutenção de impressoras. Recentemente, uma notícia dos vendedores vem tirando o sono do departamento de TI. Após um funcionário ter sido desligado da empresa, o mesmo passou informações sigilosas de acessos para a concorrente, que agora está finalizando chamados de vendas e transferindo os dados dos clientes. Você sabe que cada funcionário tem seu acesso ao banco de dados e que esse funcionário insatisfeito passou seu usuário para a empresa concorrente. Nesse caso, qual seria a solução para resolver a situação, garantindo que não ocorra novamente dentro da empresa.

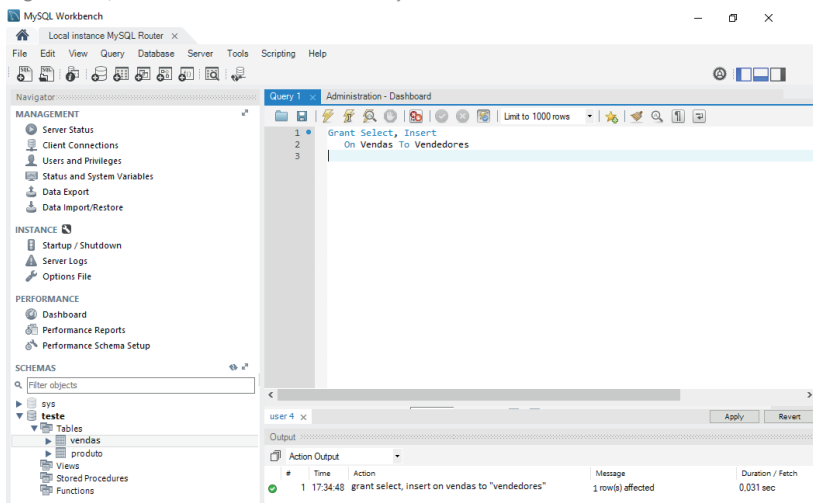
Resolução da situação-problema

Primeiramente, para resolver esse problema, é preciso saber qual usuário foi passado para a concorrência. Após descobrir, o ideal seria remover tal usuário do banco de dados, usando o comando **Drop User**. Porém, como você é um profissional que trabalha com a segurança do banco de dados, tem sempre que estar prevenido, então o ideal é validar todos os funcionários que têm acesso ao banco de dados, e se os privilégios que eles têm são mais do que necessários.

Por exemplo, um vendedor precisaria ter acesso a visualização e a inclusão, mas não de alteração e exclusão, que seriam permissões de um gerente.

Para deixarmos os vendedores com esses privilégios limitados, vamos usar o seguinte comando:

Figura 2.5 | Tela Workbench com execução do comando Grant



Fonte: captura de tela do MySQL Workbench, elaborada pelo autor.

Assim evitamos que outros usuários possam ter acesso a manipulação de informações.

Faça valer a pena

1. O uso dos comandos Grant e Revoke permite que o administrador de banco de dados administre com precisão quais recursos o usuário final do banco de dados pode acessar. O usuário final só deve receber privilégios para executar as tarefas específicas de sua função, aumentando, com isso, a segurança dos dados.

Com base na afirmação acima, qual seria a sintaxe correta para conceder todos os privilégios a tabela Produto?

- a) Grant ALL On Produto
- b) Grant Insert On Produto
- c) Grant * From Produto
- d) Grant With Select Produto
- e) Grant Select, Insert On Produto

2. Um banco de dados de uma empresa contém muitas informações e, normalmente, vários grupos de usuários. A maioria dos usuários só precisa de uma pequena parte do banco de dados para realizar suas

tarefas. Permitir aos usuários acesso irrestrito a todos os dados pode ser indesejável e um SGBD deve fornecer mecanismos para controlar o acesso aos dados (PIRES, 2009).

Qual seria o comando correto para permitir somente o acesso de consulta em uma tabela específica?

- a) Grant all clientes from aluno
- b) Grant Insert On clientes from aluno
- c) Grant Select On clientes from aluno
- d) Grant table On clientes from aluno
- e) Grant column On clientes from aluno

3. Da mesma forma que o criador da tabela pode garantir (**Grant**) privilégios de acesso aos outros usuários, ele pode revogar esses privilégios por meio do comando **Revoke**. Esse comando é útil para permissões de usuários que precisam ser reavaliadas.

Com base no conceito sobre o comando Revoke, qual seria sintaxe correta para remover todos os privilégios concedidos a todos os usuários da tabela cliente.

- a) Revoke ALL From cliente
- b) Revoke On cliente From public
- c) Revoke Grant On Cliente
- d) Revoke Select From cliente
- e) Revoke Insert, Select, Update, Delete On cliente

Seção 2.3

Gerenciamento de privilégios

Diálogo aberto

Olá, mais uma vez!

Estamos chegando ao final de mais uma unidade de estudo, em que contextualizamos a segurança de banco de dados. Agora que você já conhece um pouco mais acerca dos conceitos que alicerçam esse assunto, está na hora de entendermos como funciona o gerenciamento de acessos e privilégios e como podemos trabalhar com concessão e revogação de recursos por usuário.

O gerenciamento de privilégios, sem dúvidas, é fundamental nos dias de hoje, principalmente porque quase todos nossos dados estão sendo acessados por mais de uma pessoa.

Sem o gerenciamento, como poderíamos evitar que pessoas com más intenções possam acessar dados que são privados, que somente alguns usuários podem acessar? Um caso desse está acontecendo na empresa de Otavio, um empresário no ramo de transportes. Alguns desenvolvedores estão realizando alterações no banco de dados que estão afetando informações do sistema de Otavio, e ele está tendo problema com a saída de caminhões para entregas por falta de informações. Você, como consultor, deve ajudar Otavio a gerenciar o acesso de cada desenvolvedor e a definir se cada um deles tem os acessos necessários ou não. Apresente para ele scripts de como criar ou revogar acessos desses desenvolvedores para que não ocorra mais problemas com seus dados.

Para auxiliá-lo nesse desafio, você vai entender o que são e para que servem os privilégios, irá aprender, também, como gerenciá-los de maneira adequada e como criar e revogar privilégios por meio de comandos.

Vamos em frente!

Segundo Date (2000), a segurança de um banco de dados se refere à proteção de dados contra revelação, modificação ou destruição não autorizada.

Quando falamos em segurança em banco de dados, hoje, existem dois tipos de abordagens, conhecidas como controle discricionário e controle obrigatório (DATE, 2000). Estes aspectos são abordados da seguinte forma:

Controle discricionário

Segundo Date (2000), ocorre quando um usuário tem acessos diferentes aos dados, incluindo a capacidade de acessar campos ou tabelas (ler e modificar).

A SQL suporta controle de acesso discricionário por meio dos comandos **Grant** e **Revoke**. O comando **Grant** concede privilégios de usuário às tabelas base e às visões. A sintaxe desse comando é a seguinte:

Grant "Privilégios" ON objeto TO usuários [With Grant Option]

Para nossos propósitos, objeto é uma tabela base ou visão. Vários privilégios podem ser especificados, incluindo os seguintes:

- **Select:** o direito de acessar (visualizar) todas as colunas da tabela especificada como **objeto**, incluindo colunas adicionadas posteriormente por meio do comando de inclusão **Alter Table**.
- **Insert (Nome da coluna):** o direito de inserir linhas com valores na coluna nomeada da tabela. Se esse direito precisar ser concedido com relação a todas as colunas, incluindo colunas que possam ser adicionadas posteriormente, podemos usar simplesmente o **Insert**. Os privilégios **Update (Nome da coluna)** são semelhantes.
- **Delete:** o direito de excluir linhas da tabela nomeada como **objeto**.
- **References (Nome da coluna):** O direito de definir chaves estrangeiras que se refiram à coluna específica da tabela.



Em escolas com acesso às notas por meio da Internet, os alunos podem visualizar suas notas e médias a partir do portal da própria escola, usando seu próprio login e senha. As notas são inseridas pelos professores no mesmo portal, com acessos e permissões diferentes. Nessa situação, se o aluno tivesse permissão igual à de um professor, ele poderia mudar suas notas ou até a frequência nas aulas, causando problemas à escola e ao próprio aluno que acessou o portal de forma indevida.

Nesse cenário, todos os alunos teriam acesso somente de consulta e os professores de permissão para movimentações.

A estrutura de criação de privilégios do aluno seria:

Grant select on alunos to notas;

Já a de professores seria:

Grant all on professores to notas;

O problema pode acontecer se um aluno tiver permissão além da de consulta, podendo realizar movimentações indevidas na tabela. Uma estrutura de criação errada seria a seguinte:

Grant insert on alunos to notas;

Esse caso pode ocorrer caso façam inclusão de outro usuário de aluno em vez de um de professor.

Se um usuário tem um privilégio com **Grant Option**, ele pode passá-lo para outro usuário (com ou sem **Grant Option**) usando o comando Grant. Por exemplo:

- Um usuário que cria uma tabela base tem, automaticamente, todos os privilégios aplicáveis a ela junto com o direito de conceder esses privilégios a outros usuários.
- Um usuário que cria uma visão tem sobre ela precisamente os privilégios que possui em todas as visões ou tabelas base usadas para definir a visão. É claro que o usuário que está criando a visão deve ter o privilégio **Select** em cada tabela subjacente e, assim, é sempre garantido o privilégio **Select** na visão. O criador da visão tem o privilégio **Select** com **Grant Option** somente se tiver o

privilegio **Select** com **Grant Option** em cada tabela subjacente. Além disso, se a visão admite atualização e o usuário tem privilégios **Insert**, **Delete** ou **Update** (com ou sem **Grant Option**) sobre a (única) tabela subjacente, ele automaticamente recebe os mesmos privilégios sobre a visão.

Um exemplo de uso do **Grant Option** seria:

```
Grant select, update on empregados to Joaquin  
with Grant Option;
```

Diferente do uso do comando Grant básico (**Grant select on empregados to Joaquin;**), esse comando não dá permissão de inclusão de privilégios para outro usuário.

Apenas o proprietário de um esquema pode executar as instruções de definição de dados **Create**, **Alter** e **Drop** nesse esquema. O direito de executar essas instruções não pode ser concedido nem revogado. Em conjunto com os comandos **Grant** e **Revoke**, as visões são componentes importantes dos mecanismos de segurança fornecidos por um banco de dados. Definindo visões nas tabelas base, podemos apresentar as informações necessárias para um usuário, enquanto ocultamos outras informações a que ele não deve ter acesso. Por exemplo, considere a seguinte definição de visão:

```
1 Create View FuncionariosAtivos (nome,  
idade, setor)  
2 AS Select f.nome, f.idade, s.setor  
3 From funcionarios f, setor s  
4 Where f.codsetor = s.codigo  
5 And f.ativo = true
```

Na primeira linha, passamos o nome da visão que será criada e os retornos dela (**nome**, **idade**, **setor**) e, a partir da segunda linha, temos a construção da estrutura do retorno daquela visão.

Nesse caso, o usuário pode acessar **FuncionariosAtivos**, mas não **funcionarios** e **setor**. Sabe quais funcionários estão ativos, mas não sabe quais estão inativos, nem muito menos qual é o código de cada funcionário.

O exemplo dessa consulta seria:

Select * from FuncionariosAtivos.

Retornando somente os campos **nome**, **idade** e **setor**.



Pesquise mais

Os SGBD usam mecanismos que permitem armazenar informações dos seus usuários, principalmente o MySQL, que armazena em 4 tabelas: **db**, **user**, **tables_priv** e **columns_priv**.

Para entender um pouco mais, leia o artigo a seguir:

DUARTE, E. Gerenciamento de Usuários e Controle de Acessos do MySQL. **DEV MEDIA**. 2006. Disponível em: <<https://www.devmedia.com.br/gerenciamento-de-usuarios-e-controle-de-acessos-do-mysql/1898>>. Acesso em: 14 maio 2018.

Os privilégios mantidos pelo criador de uma visão (com relação à visão) mudam com o passar do tempo, à medida que ele ganha ou perde privilégios nas tabelas subjacentes. Se o criador perde um privilégio mantido com **Grant Option**, os usuários que receberam esse privilégio na visão também o perdem. Existem alguns aspectos sutis nos comandos **Grant** e **Revoke** quando eles envolvem visões ou restrições de integridade. Consideraremos alguns exemplos que destacam os seguintes pontos importantes:

1. Uma visão pode ser eliminada porque um privilégio **Select** é revogado para o usuário que criou a visão.
2. Se o criador de uma visão ganha mais privilégios nas tabelas subjacentes, ele automaticamente ganha privilégios adicionais na visão.
3. A distinção entre os privilégios **References** e **Select** é importante.

Os mecanismos de controle de acesso discricionário (que são baseados na concessão e revogação de privilégios **Grant** e **Revoke**), embora geralmente sejam eficazes, têm certas deficiências. Em particular, eles são suscetíveis a esquemas tipo cavalo de Troia, por meio dos quais um usuário não autorizado e desonesto pode enganar um usuário autorizado, fazendo-o revelar dados sigilosos. Por exemplo, suponha que o aluno Joaquim Furtado queira violar as tabelas de notas do professor Justino Justo. Furtado faz o seguinte:

- Ele cria uma tabela chamada `TabelaTudoMeu` e fornece privilégios `INSERT` nessa tabela a Justino (que não percebe toda essa atenção, é claro).

```
Create table TabelaTudoMeu (id int,  
nome varchar,  
nota double);  
Grant insert on TabelaTudoMeu to Justino;
```

- Ele modifica o código de algum aplicativo de SGBD que Justino usa frequentemente para fazer mais duas coisas: primeiro, ler a tabela `Notas` e, em seguida, gravar o resultado em `TabelaTudoMeu`.

Então, ele se senta e espera que as notas sejam copiadas em `TabelaTudoMeu` e depois desfaz as modificações no aplicativo, para garantir que Justino não descubra de algum modo que foi tapeado. Assim, apesar do banco de dados impor todos os controles de acesso discricionário, em que apenas o código autorizado de Justino podia acessar notas, dados sigilosos foram expostos a um intruso. O fato de que Furtado poderia modificar o código de Justino clandestinamente está fora do alcance do mecanismo de controle de acesso do SGBD.

Controle Obrigatório

Os mecanismos de controle de acesso obrigatório destinam-se a tratar dessas brechas no controle de acesso discricionário. O modelo popular de controle de acesso obrigatório, chamado modelo de Bell-LaPadula, é descrito em termos de objetos (por exemplo, tabelas, visões, linhas, colunas), sujeitos (por exemplo, usuários, programas), classes de segurança e liberações (RAMARKRISHNAN, 2011). A cada objeto do banco de dados, é atribuída uma classe de segurança e cada sujeito recebe uma liberação para uma classe de segurança; denotamos a classe de um objeto ou sujeito *A* como classe (*A*). As classes de segurança de um sistema são organizadas de acordo com uma ordem parcial, com uma classe mais segura e uma classe menos segura.

Por simplicidade, supomos que existem quatro classes: ultrassecreta (*TS - Top secret*), secreta (*S - Secret*), confidencial (*C - Confidencial*) e não classificada (*U - unclassified*). Nesse sistema, $TS > S > C > U$, em que $A > B$ significa que os dados da classe A são mais sigilosos do que os dados da classe B (RAMARKRISHNAN, 2011).



Assimile

O modelo de Bell-LaPadula impõe duas restrições em todas as leituras e gravações de objetos do banco de dados:

1. Propriedade de segurança simples - o sujeito S pode ler o objeto O somente se classe (S) \geq classe (O). Por exemplo, um usuário com liberação TS pode ler uma tabela com liberação C, mas um usuário com liberação C não pode ler uma tabela com classificação TS.
2. Propriedade-* - o sujeito S pode gravar o objeto O somente se classe (S) \leq classe (O). Por exemplo, um usuário com liberação S pode gravar apenas objetos com classificação S ou TS.

Se controles de acesso discricionário também forem especificados, essas regras representarão restrições adicionais. Portanto, para ler ou gravar um objeto do banco de dados, um usuário deve ter os privilégios necessários (obtidos por meio de comandos **Grant**) e as classes de segurança do usuário e do objeto devem satisfazer as restrições anteriores.

Vamos considerar como um mecanismo de controle obrigatório poderia ter frustrado Joaquim Furtado. A tabela Notas poderia ser classificada como S, Justino poderia receber liberação para S e Joaquim Furtado poderia receber uma liberação mais baixa (C). Furtado só poderia criar objetos de classificação C ou menor; portanto, a tabela TabelaTudoMeu poderia ter no máximo a classificação C. Quando o programa aplicativo executado em nome de Justino (e, portanto, com liberação S) tentasse copiar Notas em TabelaTudoMeu, não poderia fazer isso, pois classe (TabelaTudoMeu), classe (aplicativo) e a Propriedade seria violada.

Nesse caso, todos os usuários teriam permissão somente de leitura em vez de modificações na tabela TabelaTudoMeu.


```
Grant select on TabelaTudoMeu to Justino;
```

```
Grant select on TabelaTudoMeu to Joaquin;
```

E, na tabela Notas, somente Justino teria acesso à tabela, e Joaquin não teria acesso nenhum a tabela.

```
Grant all on notas to Joaquin;
```

Limitando recursos por Usuário

Elmasri (2005) afirma que técnicas para limitar a propagação de privilégios têm sido desenvolvida, embora elas ainda não tenham sido implementadas na maioria dos bancos de dados e não façam parte da SQL.

Para explicarmos como limitar um recurso do usuário, vamos usar o MySQL como exemplo que possui um mecanismo que permite limitar acessos de usuários a nível de banco, tabela e até mesmo coluna, além de poder limitar em dias o tempo que o usuário pode se conectar no banco de dados.

Para realizar as limitações de recursos por usuários, teremos que usar o controle discricionário, em que teremos duas divisões:

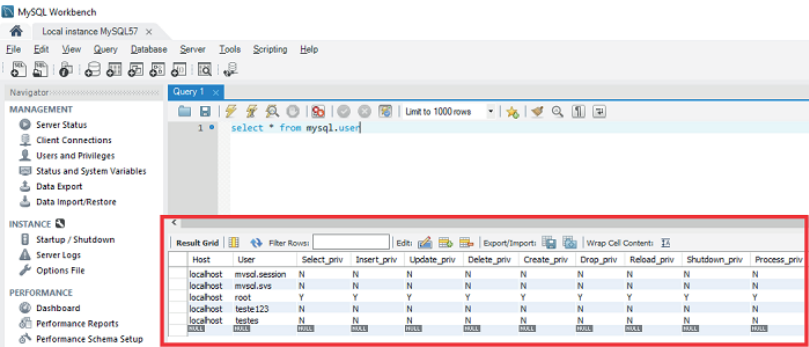
- Nível de conta: permissões para o usuário realizar comandos de definição de dados ou também chamado de DDL. Exemplos: CREATE, ALTER e DROP.
- Nível de relação: permissões para o usuário realizar comandos de manipulação de dados, também chamado de DML. Exemplos: SELECT, INSERT e DELETE.

Primeiramente, precisamos saber quais são os usuários cadastrados no meu SGBD. Abra o MySQL Workbench e conecte-se como o usuário root. Com esse usuário, você tem permissão completa. Conectado, execute o seguinte comando:

```
Select * from mysql.user;
```

mysql é a base de dados e **user** é a tabela. O resultado vai exibir todos os usuários do SGBD e suas permissões, como mostrado na Figura 2.6.

Figura 2.6 | Usuários e suas permissões no MySQL Workbench



Fonte: captura de tela do MySQL Workbench 6.3 CE, elaborada pelo autor.

Essa tabela nos traz também as permissões que o usuário tem no nível de conta (DDL), exibindo permissões em todo o SGBD e não em bancos de dados específicos.

Outra maneira de visualizar informações de permissões é por meio dos comandos a seguir:

-- Mostra as permissões que o usuário tem no banco conectado

Show Grants;

-- Mostra as permissões que um usuário específico tem não especificamente no mesmo banco de dados

Show Grants for '[usuario]'@'[local]';

Com base nesses comandos, é possível tomar decisões para verificar se um usuário necessita ou não de uma determinada permissão. É preciso muita cautela ao realizar um comando para revogar uma permissão, pois caso, por um descuido, for realizar um comando para remover uma permissão do mesmo usuário em que está conectado, o mesmo privilégio será revogado e, dependendo da permissão, somente outro usuário com nível administrativo poderá realizar a alteração novamente.

Esse usuário específico não pode executar muitas conexões simultâneas, ou modificações em um espaço de tempo muito curto? Nessa situação, entraremos mais a fundo no comando

Grant. Com ele podemos gerenciar o nível de usuário através de três recursos específicos:

- Número de consultas (SELECT) por hora: todos os comandos que podem ser executados somente por um usuário.
- Número de atualizações por hora: todos os comandos que modifiquem tabelas ou banco de dados.
- Número de conexões por hora: conexões abertas por hora.

Por padrão, ao criar um usuário ele não é limitado pelos recursos acima, a não ser que estes limites sejam garantidos a ele. A sintaxe deste comando seria a seguinte:

```
Grant ... With Max_Queries_per_hour N1  
Max_Updates_per_hour N2  
Max_Connections_per_hour N3;
```

Se o usuário alcançar o número limite de consultas ou modificações durante uma hora, o SGBD não aceitará mais consultas ou modificações até a próxima hora. O mesmo ocorrerá se usuário atingir o limite de conexões dentro de uma hora, caso em que o SGBD não aceitará mais conexões até a próxima hora.



Refleta

Quem pode dar mais “dor de cabeça” em um SGBD: um usuário que inadvertidamente obtenha privilégios de uma tabela ou um usuário que indevidamente obtenha privilégios de sistema?

Com base no que aprendemos, você será capaz de gerenciar e limitar privilégios de acordo com o perfil dos usuários.

Sem medo de errar

Conforme abordamos no início dessa seção, precisamos ajudar Otavio, que está tendo problemas com o seu departamento de desenvolvimento que está realizando alterações no banco de dados de forma indevida, afetando informações já inseridas no sistema. Por causa dessas alterações, ele está tendo problemas

com a saída de caminhos para entregas por falta de informações ou informações erradas.

Nessa situação, primeiramente, é preciso saber quais usuários estão afetando o sistema de Otávio para que possamos tomar as medidas certas e não prejudicar outros desenvolvedores. Após verificar quais usuários estão afetando o sistema, é preciso realizar o seguinte comando:

```
Revoke      Insert,      update,      delete      ON  
'[desenvolvedor]'@'localhost' From "User"
```

Isso também pode ser feito removendo todos os privilégios e, depois, inserindo somente a permissão para consultar. Veja:

```
Revoke all ON '[desenvolvedor]'@'localhost'  
From "User"
```

```
Grant Select ON '[desenvolvedor]'@'localhost'  
From "User"
```

Na primeira linha do comando, **Revoke all** serve para remover todas as permissões de todos os usuários "desenvolvedores". Ou seja, um desenvolvedor que tenha acesso a **Insert, update** e **delete** acaba perdendo todas essas permissões. Outros usuários desse banco de dados terão suas permissões intactas.

Depois de remover as permissões, a segunda linha inclui somente a opção de visualizar para os usuários "desenvolvedores". Com essa permissão de consulta, os usuários "desenvolvedores" somente conseguiram realizar comandos simples de consulta, como:

```
Select Codigo, Cliente, Endereco, Telefone  
from cadastroCliente
```

Em vez de realizar comandos de movimentações, como **insert, delete** ou **update**.

Nos dois exemplos, os usuários desenvolvedores somente conseguirão visualizar as informações, o que é o suficiente para que Otávio resolva seus problemas com erros de informações no seu banco de dados. Os dois exemplos são bastante usados, dependendo da situação de cada empresa, por exemplo, um comando **Revoke**:

```
Revoke all ON '[desenvolvedor]'@'localhost'  
From "User"
```

Esse comando poderia ser usado também para “limpar” usuários que estão inativos ou que não estão mais trabalhando na empresa. Depois, teria que ser criada uma regra para os desenvolvedores ativos somente da permissão de acesso. Assim, evitar-se-ia que um usuário ativo, com menos privilégios que um usuário inativo, utilizasse tal usuário inativo na empresa.

Avançando na prática

Removendo privilégios desnecessários

Descrição da situação-problema

Rodolfo é dono de um restaurante muito conhecido na cidade onde mora há anos. O negócio cresceu e Rodolfo precisou adquirir um sistema para facilitar seu dia a dia.

Depois de adquirir seu sistema, Rodolfo cadastrou quatro funcionários, todos com privilégios para consultar o banco de dados.

Depois de alguns anos, Rodolfo consultou seu banco de dados e observou a situação de acessos do restaurante. Ele percebeu que dois de seus ex-funcionários ainda constam no banco de dados. Porém, outros funcionários usam esses mesmos usuários para realizar consultas no banco de dados, sendo que os funcionários anteriores precisavam de privilégios para modificar informações como o preço de um produto.

Quais providencias que Rodolfo precisa tomar para que os funcionários não usem usuários ‘fantasmas’ do banco de dados, com privilégios a mais do que precisam. E, caso precisem usar estes usuários ‘fantasmas’, o que Rodolfo precisa fazer para diminuir os privilégios destes usuários?

Resolução da situação-problema

Primeiramente, o ideal é que Rodolfo remova os ex-funcionários do banco de dados, usando o seguinte comando:

```
Drop user 'exfuncionario' ;
```

Nesse comando, Rodolfo precisa especificar quais funcionários saíram da empresa para removê-los do banco de dados.

Porém, caso tal funcionário use um usuário de um novo funcionário, é preciso removê-lo usando o comando **Drop user**, ou limitar o acesso dele somente para permissão de consulta.

Para realizar uma permissão somente de consulta, é preciso, primeiramente, remover todas as permissões do funcionário para depois incluir a permissão de visualização:

```
Revoke all ON '[funcionario]' '@'localhost'  
From "User";
```

Só depois de executar o comando para remover a permissão que poderá ser executado o novo comando para dar permissão somente para visualizar informações.

```
Grant Select ON '[funcionario]' '@'localhost'  
From "User"
```

É preciso muita atenção na ordem desses comandos, porque caso você os inverte, primeiramente será concedido o privilégio de visualizar e depois todos os tipos de privilégio serão perdidos. Essa execução inversa será parecida com o comando de **Drop user**, porém o registro do funcionário continuará no banco inativo por causa das permissões.

Faça valer a pena

1. No banco de dados, para podermos realizar as limitações de recursos de um usuário, teremos que usar o controle discricionário. Com ele, podemos dividir em dois tipos de níveis de permissões para que o usuário realize seus comandos.

Quais são esses dois níveis de permissões?

- a) Nível de usuário e conta.
- b) Nível de conta e login.
- c) Nível de relação e conta.
- d) Nível de usuário e login.
- e) Nível de servidor e conta.

2. Em uma análise mais rigorosa de classificação dos comandos do módulo funcional DCL, não é difícil enxergar que a divisão em básicos e avançados é arbitrária. Isso se deve ao fato de que toda a DCL é baseada em dois comandos.

Quais comandos efetivamente fazem parte do DCL?

- a) GRANT e REVOKE.
- b) GRANT e INSERT.
- c) GRANT e SELECT.
- d) REVOKE e CREATE.
- e) CREATE e ALTER .

3. Se o criador perde um privilégio mantido com Grant Option, os usuários que receberam esse privilégio na visão também o perdem. Existem alguns aspectos sutis nos comandos Grant e Revoke quando envolvem visões ou restrições de integridade.

Consideraremos alguns exemplos que destacam os seguintes pontos importantes:

- 1. Uma visão pode ser eliminada porque um privilégio **Select** é revogado do usuário que criou a visão.
- 2. Se o criador de uma visão ganha mais privilégios nas tabelas subjacentes, ele automaticamente ganha privilégios adicionais na visão.
- 3. A distinção entre os privilégios **References** e **Select** é importante.

Referente à visão, qual seria a semelhança entre uma visão e um privilégio de acesso?

- a) Tanto uma Visão quanto um Grant podem limitar o usuário a consultar uma determinada tabela a coluna
- b) Visões e Grant são usados para monitorar acesso.
- c) Na visão, você pode acessar tabelas sem ter permissões, já no Grant não.
- d) No Grant, você consegue ter acesso a mais de um tipo de manipulação, como uma consulta e modificação, na visão também.
- e) Na visão você consegue dar permissões nas tabelas, igual no Grant.

Referências

DATE, C. J. **Introdução a Sistemas de Banco de Dados**. 7. ed. Rio de Janeiro: Campus, 2000.

ELMASRI, R. **Sistemas de banco de dados**. 4. ed. São Paulo: Pearson Education, 2005.

MACHADO, F. N. R. **Projeto e implementação de banco de dados**. 3. ed., São Paulo: Érica, 2014

PIRES, C. E. **Conhecendo os usuários de um sistema de banco de dados**. Universidade Federal de Campo Grande, 9 dez. 2009. Disponível em: <http://www.dsc.ufcg.edu.br/~cesp/publications/Palestra_Grupo_PET_DSC_UFCG.pdf>. Acesso em: 04 maio 2018.

RAMAKRISHNAN, R. **Sistemas de gerenciamento de banco de dados**. 3. ed. Nova Iorque: McGraw Hill, 2011.

Recursos avançados em banco de dados

Convite ao estudo

Olá, aluno! Você já deve ter ouvido falar de empresas que tiveram seus dados sequestrados ou criptografados por hackers e, para liberá-los, precisam pagar uma quantia enorme para que eles liberem a criptografia. Hoje em dia, isso é mais comum do que parece, pois essas empresas não pensam em um plano B para o caso de acontecer esse tipo de incidente, não só na parte de sequestro, mas na melhoria de seu banco de dados, para que não perca performance e o usuário continue acessando as informações normalmente.

O Sr. Joaquin tem receio de que no próximo feriado prolongado o número de acessos de seu servidor aumente e um de seus sistemas web fique off-line em decorrência do volume de acesso. Em um primeiro momento, você precisa apresentar uma maneira de conciliar os acessos e tráfegos de consultas das áreas de produção e de engenharia de produtos importados. Apresente um relatório sobre como ele poderia resolver essa questão de acessos e de tráfego usando instâncias, criando uma para cada área. Apresente também qual seria a vantagem de separar cada área em uma nova instância e como isso poderia ajudá-lo a evitar problemas futuros de acessos ou de consultas em seu sistema.

Para tratar essa situação com eficiência você irá estudar os seguintes conteúdos:

- Na Seção 3.1, você vai aprender o conceito de múltiplas instâncias e como gerenciá-las.
- Na Seção 3.2, você irá compreender o conceito de backups e *restores*, entendendo suas principais

vantagens, além de saber como funciona uma replicação de dados em um sistema de gerenciamento de banco de dados (SGBD).

- Na Seção 3.3, você entendera como funciona um banco de dados na nuvem, conhecendo seus aspectos e estruturas, além de compreender suas principais vantagens de utilizá-lo.

Com seu empenho nos estudos e sua participação durante as aulas, esta unidade será bastante produtiva. É importante que você faça os exercícios para fixar o conteúdo e compreender suas dificuldades. Então, vamos seguir nossa leitura.

Bom estudo!

Seção 3.1

Introdução a múltiplas instâncias

Diálogo aberto

Vamos iniciar a nossa terceira unidade, a partir da qual vamos entender alguns recursos avançados do banco de dados. Depois de aprendermos a administrar um banco de dados e a manter sua segurança, agora vamos aprender sobre novos recursos que são úteis na vida de um DBA, tanto para a administração quanto para segurança das informações. Nesta primeira seção, falaremos sobre instâncias: o que são, onde usar e como trabalhar com múltiplas delas. Iremos analisá-las e entendê-las, com exemplos do dia a dia profissional, para fixar melhor os estudos.

Com nossos esforços, vamos poder contextualizar o uso de instâncias e como elas seriam úteis na empresa do Sr. Joaquin, que está com receio de que no próximo feriado prolongado o número de acessos de seu servidor aumente e um de seus sistemas web fique off-line por causa do número alto de acessos em uma das suas áreas. Neste primeiro momento, você precisa apresentar uma maneira de conciliar os acessos e tráfegos de consultas das áreas de produção e de engenharia de produtos importados.

Apresente um relatório sobre como ele poderia resolver essa questão de acessos e de tráfego usando instâncias, criando uma para cada área, e qual seria a vantagem de separar cada área em uma nova instância e como isso o ajudaria a evitar problemas de acessos ou de consultas em seu sistema. Para auxiliá-lo nesse desafio, vamos conhecer o que é uma instância, quais seriam suas vantagens, além de poder gerenciar múltiplas delas. Com foco e estudo, vamos conseguir dar um passo a diante e resolver essa situação problema.

Bom estudo!

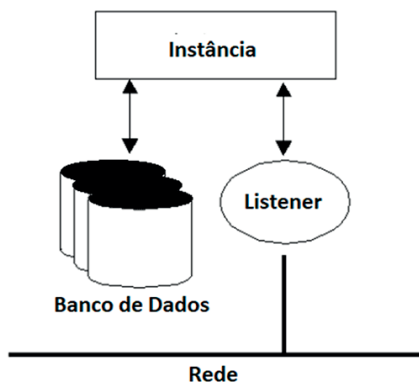
Não pode faltar

Com o aumento da quantidade de informações geradas pelas aplicações, os administradores de banco de dados se deparam com complexidades no gerenciamento dos ambientes de um banco de dados. Esse aumento, combinado com o baixo custo dos meios de armazenamentos nos servidores, iniciou um cenário no qual um administrador de banco de dados, além de administrar algumas dezenas de banco de dados, passou a administrar centenas de banco de dados.

Garantir a integridade, melhorar o desempenho e a estabilidade do banco de dados são responsabilidades de um administrador de banco de dados. Além de garantir de forma proativa e automatizada banco de dados com um número maiores de instâncias (NISHIMURA, 2009). Mas o que é uma instância?

Para Nishimura (2009), uma instância é um componente típico de um servidor, em que uma ou mais CPUs podem alocar espaços em discos e na memória. Uma instância é importante no banco de dados por que serve para acessar as informações sem qualquer restrição (NISHIMURA, 2009). Você pode criar sua instância com o objetivo de montá-la especificamente para um banco de dados para melhorar seu desempenho. A Figura 3.1 mostra um exemplo da estrutura de uma instância. É possível configurar múltiplas instâncias para poder acessar o mesmo grupo de arquivos ou banco de dados. Múltiplas instâncias em vários servidores acessando um banco de dados central permitem uma escalabilidade e alta disponibilidade de desempenho.

Figura 3.1 | Estrutura do funcionamento de uma instância no banco de dados.



Fonte: elaborada pelo autor.

Na Figura 3.1, o Listener é o processo ouvidor, em que uma aplicação cliente faz uma conexão com servidor de BD através do listener(ouvidor), que é um processo que “ouve” todas as requisições de conexão como representante de um banco de dados ou mais. As principais características do Listener são:

- Ele pode “ouvir” um ou mais banco de dados.
- Diversos listeners podem “ouvir” um mesmo banco de dados afim de promover o balanceamento de carga.
- Ele pode “ouvir” através de vários protocolos.
- O nome padrão do listener em uma rede é LISTENER;

Enquanto um armazenamento do banco de dados é em um servidor, a memória fica responsável por alocar a instancia. A instancia é composta por um bloco de memória que fica alocado em uma área chamada *System Global Area* (SGA), que fica juntamente com alguns processos em segundo plano e que interagem com arquivos de banco de dados (NISHIMURA, 2009).



Pesquise mais

Para entender um pouco mais sobre o mecanismo de instâncias, comece pelo artigo disponível em:

INSTÂNCIAS do mecanismo de banco de dados. 12 jun. 2017. Disponível em: <[https://msdn.microsoft.com/pt-br/library/hh231298\(v=sql.120\).aspx](https://msdn.microsoft.com/pt-br/library/hh231298(v=sql.120).aspx)>. Acesso em: 1 ago. 2018.

Em um banco de dados (Oracle® MySQL, por exemplo), quando criado automaticamente, ele já tem uma instancia própria. Caso precise de duas ou mais instancias em um servidor, é preciso analisar se a situação se encaixa em um desses dois casos:

1. São realmente necessárias várias conexões simultâneas, com uma somente para consulta e a outra para gravação no mesmo banco de dados? É preciso entender bem o que se está fazendo, ser capaz de desenvolver aplicações de leitura a partir de uma instância somente leitura e escrever em uma instância feita somente para gravação.

2. É preciso ter múltiplas versões de desenvolvimento do mesmo banco de dados no servidor, com o mesmo nome e, talvez (não recomendado, mas ainda possível), uma versão de produção também. Por exemplo, uma empresa de software que tem um banco de dados na versão 1.0 para o sistema local e outro banco de dados na versão 2.0 para o sistema web.

No MySQL existe uma ferramenta que facilita a execução de múltiplas instâncias, chamada *mysqld_multi*. Ele foi desenvolvido para gerenciar múltiplos processos que trabalham com conexões diferentes, podendo parar ou iniciar servidor ou então reportar seu status atual.

Configurando múltiplas instancias no MySQL

No banco de dados MySQL, o componente *mysqld_multi* vem sendo bastante usado nos servidores de produção. Ele é disponibilizado em forma de pacote do servidor e usa o mesmo arquivo de configurações do MySQL (*my.ini*). Dentro desse arquivo, você encontrará uma estrutura organizada por grupos, em que seções são definidas por opções de configuração e são específicas para um determinado propósito. Um exemplo básico de um arquivo de configuração é dividido em 3 grupos, como mostra a Figura 3.2

Figura 3.2 | Estrutura básica de um arquivo de configuração do MySQL

```
[client]
port          = 3306
socket        = /var/run/mysqld/mysqld.sock
user          = aluno
password      = senha12345

[mysqld]
user          = mysql
pid-file      = /var/run/mysqld/mysqld.pid
socket        = /var/run/mysqld/mysqld.sock
port          = 3306
datadir       = /var/lib/mysql

[xtrabackup]
target_dir    = /backups/mysql/
```

Fonte: elaborada pelo autor.

De acordo com a Figura 3.2, as opções definidas dentro do grupo [cliente] são usadas pela ferramenta de comando do MySQL. Caso não especifique nenhuma outra opção durante a execução do MySQL, ele tentará se conectar com o servidor local usando o caminho do soquete com as credenciais preenchidas nesse mesmo grupo. Da mesma forma, o MySQL irá procurar opções definidas na sua seção de inicialização. Os parâmetros definidos pelos grupos acima também podem ser declarados por linha de comando.

Para podermos criar várias instâncias do MySQL, temos que substituir o grupo [mysqld] (Figura 3.3) do arquivo de configuração pela quantidade de grupos necessários. O padrão para criar esses grupos seria [mysqldN], em que N seria um número inteiro que serve para identificar cada instância criada, então muito cuidado para não criar um nome de grupo igual, pois o MySQL não conseguirá reconhecer esse grupo duplicado. Além do nome do grupo ter que ser distinto, as mesmas opções são válidas para cada grupo criado. Diferente do [mysqld] os demais grupos precisam estar com todas informações preenchidas para que a instância apareça.

Figura 3.3 | Exemplo da estrutura de configuração do mysqld

```
[mysqld]
user      = aluno
pid-file  = /var/run/mysqld/mysqld.pid
socket    = /var/run/mysqld/mysqld.sock
port      = 3306
datadir   = /var/lib/mysql

[mysqld1]
user      = aluno1
pid-file  = /var/run/mysqld/mysqld1.pid
socket    = /var/run/mysqld/mysqld1.sock
port      = 3307
datadir   = /data/mysql/mysql1

[mysqld9]
user      = aluno9
pid-file  = /var/run/mysqld/mysqld9.pid
socket    = /var/run/mysqld/mysqld9.sock
port      = 3308
datadir   = /data/mysql/mysql9
```

Fonte: elaborada pelo autor.

Dentre as opções do mysqld as principais são (ELMASRI, 2005):

- Pid-file – indica o nome do caminho do arquivo em que o servidor grava seu ID de processo.
- Socket – é responsável por controlar o caminho do arquivo soquete ou o nome do pipe nomeado no Windows. No

Windows, é necessário detalhar os nomes de canal distintos apenas para os servidores configurados, para permitir conexões de pipe nomeado.

- Port – responsável por controlar o número da porta de conexões TCP/IP.
- Datadir – diretório onde fica a configuração da instância.

Analise com cautela a Figura 3.3 e note que os caminhos de pid-file, soquete, porta e datadir são distintos. Isso se dá por que uma instância não pode acessar informações de outra instância.



Assimile

No `mysqld_multi` existem dois comandos uteis para usar de exemplo na hora de verificar ou criar uma instância:

```
$ mysqld_multi --example (fornece um exemplo de um arquivo my.ini configurado com várias instancias).
```

```
$ my_print_defaults --defaults-file=/etc/my.cnf mysqld9 (mostra como um determinado grupo "mysqld9" foi definido no my.ini).
```

Agora que entendemos como funciona a configuração de várias instâncias, vamos aprender como gerenciá-la. O `mysql_multi` permite que você inicie, pare, reinicie ou reporte o status atual de uma determinada instância ou de todas as instâncias. Uma observação é que, para a ação "stop", ela é gerenciada por meio do `mysqldadmin`, e, internamente, isso acontece individualmente, com uma chamada de "stop" por instância, mesmo que tenha o `mysqld_multi` para todas elas.

Para funcionar de maneira correta, você precisa configurar uma conta MySQL com o privilégio de **SHUTDOWN** e definir o mesmo nome de usuário e senha para todas as instâncias configuradas. Isso fará com que funcione imediatamente se você executar o `mysqld_multi` como root e em um servidor recém-instalado.

O Exemplo do comando seria este:


```
GRANT SHUTDOWN ON *.* TO 'multi_admin'@'localhost'  
IDENTIFIED BY 'multipass';
```

```
FLUSH PRIVILEGES;
```

Para replicar o datadir do servidor principal nas outras instâncias, é possível criar uma conta antes de fazer as cópias. Caso contrário, basta conectar-se a cada instância e criar outra conta semelhante.



Exemplificando

As empresas que utilizam software legado normalmente não podem atualizar a versão de seus bancos de dados, e, obviamente, caso necessitem utilizar um software novo. Porém, com a versão do banco de dados mais atualizada, é necessário criar uma nova instância para não interromper o funcionamento do sistema legado.

Se você usar várias instalações do MySQL e em locais diferentes, poderá especificar o diretório base para cada instalação como opção. Consequentemente, isso faz com que cada instância use automaticamente um diretório de dados, arquivos de log e arquivos PID diferentes, pois o padrão de cada um desses parâmetros é relativo ao diretório base.



Refleta

Seria vantajoso, atualmente, usar uma única instância para monitorar diversos bancos de dados? Uma das vantagens de usarmos múltiplas instâncias é o isolamento do processo, ou seja, cada banco de dados teria seu diretório dedicado de persistência e monitoramento, aumentando a confiabilidade e segurança.

Outros tipos de instancias em banco de dados, como no Oracle, são o SGA e o PGA.

O *System Global Area* (SGA) é uma área onde a memória é compartilhada e disponibilizada pelo banco de dados do Oracle.

Essa memória é dinâmica e alocada quando se inicia a instância, porém, exclusivamente para cada processo do Oracle (no caso, o processo é uma conexão). Essa memória pode ser utilizada por todos os processos derivados do Oracle e tem como objetivo prover:

- Cache de buffer do banco de dados.
- *Shared Pool*.
- *Buffer de Redolog*.
- *Large Pool*.
- *Java Pool*.
- *Streams Pool*.

O *Program Global Area* (PGA) é uma área onde a memória fica dedicada exclusivamente a cada processo do Oracle. Nessa área da memória, encontramos:

- Área de SQL, em que binds, queries e os demais atributos de *queries* executadas permanecem;
- Memória de sessão, em que os valores dos processos ou variáveis residem.

Em geral, cada vez que você conecta a um banco de dados da Oracle, uma área de memória no servidor PGA será dada exclusivamente para você. Porém, tudo que você executar no servidor usará a memória dedicada à instância (SGA).

Aprendemos aqui algumas vantagens de usar múltiplas instancias e como gerenciá-las, otimizando a performance, disponibilidade e integridade do banco de dados.

Sem medo de errar

O Sr. Joaquin está com receio de que no próximo feriado prolongado o número de acessos de seu servidor aumente e um de seus sistemas web fique off-line por causa do número de acessos em uma das suas áreas. Neste primeiro momento, você precisa apresentar uma maneira de conciliar os acessos e tráfegos de consulta das áreas de produção e de engenharia de produtos importados. Apresente um relatório sobre como ele poderia resolver essa questão usando instâncias, criando uma para cada

área (produção e engenharia de produtos importados), qual seria a vantagem de separar cada área em uma nova instância e como elas o ajudariam a evitar problemas futuros de acessos ou de consultas em seu sistema.

Para resolver o problema do Sr. Joaquin, é preciso criar uma instância para cada área (produção e engenharia de produtos importados). Mas, antes disso, é preciso validar se realmente é necessário instalar duas instâncias no servidor. Para analisar a necessidade, vamos voltar no que aprendemos no nosso livro de estudo:

1. São realmente necessárias várias conexões simultâneas, em que uma seria somente para consulta e a outra para gravação no mesmo banco de dados? Primeiramente é preciso saber que está fazendo e ser capaz de desenvolver aplicações de leitura a partir de uma instância de somente leitura e escrever em uma instancia feita somente para gravação.

2. É preciso ter várias versões de desenvolvimento de um mesmo banco de dados no servidor, com o mesmo nome, e, talvez uma versão de produção também. Por exemplo, uma empresa de software que tem um banco de dados na versão 1.0 para o sistema local e outro banco de dados na versão 2.0 para o sistema web.

Ao analisarmos esses itens, podemos observar que, na empresa do Senhor Joaquin, é realmente necessário criar duas instâncias. O uso de múltiplas instâncias no banco de dados do Sr. Joaquin permitirá reduzir custos com infraestrutura, realizar a otimização de recursos físicos, eficiência na depuração de falhas sistêmicas encontradas na produção e agilidade na atualização de ambientes de teste, homologação e produção.

Avançando na prática

Sistemas com o mesmo núcleo

Descrição da situação-problema

Uma software house recebeu uma demanda para construir uma plataforma web com vários aplicativos distintos usando o mesmo

núcleo. O tempo de desenvolvimento era curto para atender a demanda, assim, foi preciso separar módulos entre a equipe de desenvolvedores, o que criou um grande desafio para integrar a demanda ao mesmo núcleo, visto que esse também estaria em construção. Nessa situação, quais medidas a software house precisaria para poder realizar a entrega da plataforma com qualidade?

Resolução da situação-problema

Para poder entregar a plataforma web com todos os módulos e com qualidade, o administrador de dados precisa, primeiramente, analisar se é possível dividir os módulos sem atrapalhar o desenvolvimento. Caso seja possível, o ideal é dividir uma instância para o desenvolvimento, uma para o pessoal de testes e outra para uma versão estável já no cliente.

A versão do desenvolvimento será a instância com o banco de dados mais atual, pois ele tem as modificações que estão sendo desenvolvidas. Em seguida, tem-se o banco de testes, em que é usado um banco feito por um backup do cliente, porém com atualizações que saíram do desenvolvimento. Depois, teríamos a instância do banco de dados do cliente, em que teríamos o banco de dados final usado para o cliente.

Com instâncias divididas, os departamentos de desenvolvimento e testes não terão problemas com atualizações de seus bancos ou scripts, por que cada um poderá trabalhar com o seu banco e, consequentemente, aumentando a velocidade da entrega.

Faça valer a pena

1. Para podermos criar várias instâncias do banco de dados, temos que substituir o grupo `[mysqld]` do arquivo de configuração pela quantidade de grupos necessários. O padrão para criar esses grupos seria `[mysql"X"]`. Esse nome tem que ser distinto das mesmas opções que são válidas para cada grupo criado.

No que concerne o grupo `[mysql"X"]`, o que seria o caractere "X"?

- a) "X" seria um número inteiro, que identifica cada instância criada.
- b) "X" seria um qualquer valor, responsável por identificar cada grupo.
- c) "X" seria uma constante.
- d) "X" seria o nome do usuário.
- e) "X" seria o a chave do grupo.

2. O `mysql_multi` permite que você inicie, pare, reinicie ou reporte o status atual de uma determinada instância ou de todas as instâncias. Para a ação de "stop", ela é gerenciada por meio do `mysqladmin` e, internamente, isso acontece individualmente, com uma chamada de "stop" por instância, mesmo que tenha o `mysql_multi` para todas elas.

Para gerenciar, é preciso ter um certo privilégio na conta MySQL, qual seria este privilégio?

- a) INSERT
- b) UPDATE
- c) DELETE
- d) SHUTDOWN
- e) RESTART

3. Para obtermos um melhor desempenho dentro das instâncias, é possível especificar uma opção dentro da configuração da instância para distribuir a carga entre vários discos físicos, assim, cada instância usa seu próprio diretório de dados.

Qual seria o parâmetro para esta opção?

- a) Port
- b) Mkdir
- c) Tmpdir
- d) Socket
- e) Datadir

Seção 3.2

Recuperação de dados

Diálogo aberto

Caro aluno, depois de aprender o conceito de instância, que nada mais é que um componente típico de um servidor, em que uma ou mais CPUs, podem alocar espaços em discos e memória. Você aprenderá o conceito sobre backups e *restore*, entenderá como funcionam e qual a importância deles. Perder informações de um banco de dados é um sentimento difícil de descrever, ainda mais se as informações forem confidenciais. Essas informações são anos de memória e de dados jogados no lixo por uma falha qualquer, sem a possibilidade de recuperá-los.

Esse problema pode ocorrer de diversas formas, algumas até acidentais. Uma pesquisa feita pela empresa EaseUS mostrou que 44% das perdas de dados ocorreram por ações sem intenção, como apagar arquivos ou partições erradas, má colocação de cartões de memória ou ataque por vírus. Já 32% ocorrem por ações intencionais, como a formatação do disco rígido, e 21% por falhas (no software, HD ou energia, além de corrupção dos sistemas de arquivo do banco de dados) (ACTIVE SOLUTION, 2018).

Para evitar que isso ocorra, a solução é bem simples: fazer o backup dos dados da sua empresa. Um backup nada mais é do que uma cópia de segurança de todas as informações virtuais, feita para evitar a perda de arquivos importantes ou mesmo para poder recuperar arquivos que tenham sido alterados de maneira não autorizada. Após entender o funcionamento de backup e *restore*, você aprenderá como funciona a replicação de dados, entendendo desde seu conceito até como utilizá-los de forma adequada.

Antes da chegada da véspera do feriado prolongado, o Sr. Joaquin precisa se preocupar com outro item, que é a segurança de suas informações. Mesmo separando as duas áreas do sistema (produção e engenharia de produtos importados), ainda pode ocorrer uma queda no desempenho com o aumento de acessos em uma das instâncias, e, para que este tipo de situação não

ocorra, o Sr. Joaquin disponibilizou outro servidor com outro link de internet. Porém, ele ainda tem dúvidas sobre como fazer isso, portanto, você precisa apresentar para ele, por meio da ferramenta Workbench, como ele poderia resolver esta situação, evitando que uma das instâncias fique off-line por muito tempo.

Com seu esforço nos estudos e nas aulas, você será capaz de realizar criação e restauração de backup de banco de dados, além de saber fazer replicação de dados, aumentando o seu nível de segurança na estrutura.

Vamos em frente!

Não pode faltar

O banco de dados armazena e gerência os bens mais valiosos de uma empresa. Isso acontece por que o mercado está cada vez mais competitivo e acelerado, exigindo das empresas respostas rápidas e assertivas, além de estratégias bem planejadas e executadas. Como dissemos acima, o banco de dados armazena informações e, nessa batalha de competitividade, informação é poder (ALVES, 2014).

Para mantermos a segurança de nossos dados, precisamos guardar essas informações em outros locais, seja outra unidade de disco, outro servidor, um pendrive ou até mesmo backups em nuvem, para que possamos garantir a preservação e recuperação de informações ou do banco de dados inteiro. Isso significa que o administrador deve fazer uso de recursos oferecidos pelo próprio servidor de banco de dados ou de outro meio que possibilite ter cópias do banco de dados. A maneira mais óbvia é fazer backups periódicos. Alguns sistemas operacionais e servidores oferecem ferramentas para essa tarefa, mas no caso de não estarem disponíveis, pode ser utilizado um dos vários aplicativos utilitários disponíveis no mercado, alguns até gratuitos. A maioria dos SGBDs relacionais padrão SQL oferece utilitários para cópia e restauração (backup/restore) das bases de dados (ALVES, 2014).



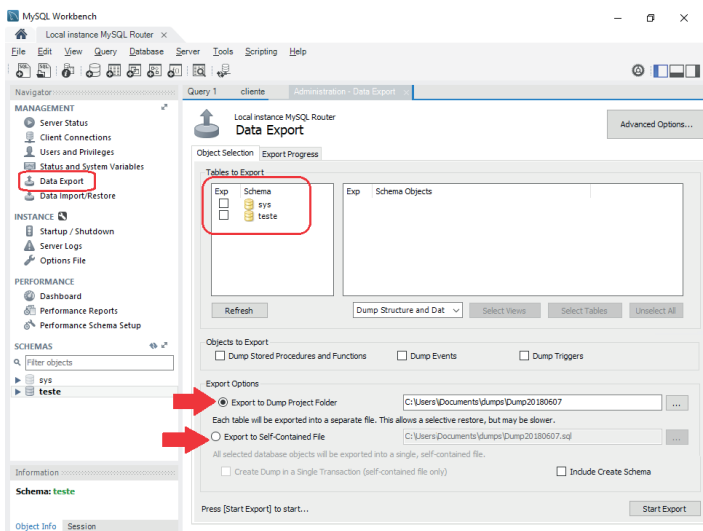
Vamos supor que não existisse a opção de backup nem de restore nos bancos de dados. Como poderíamos garantir a segurança das informações, caso houvesse uma perda de informações?

Uma ferramenta para realizar um backup no banco de dados MySQL é o Workbench, um gerenciador de banco de dados disponibilizado junto com a instalação do MySQL. Com ele é possível realizar consultas e modificações de dados, inclusão de usuários ou permissões, e até mesmo gerenciar um ou mais banco de dados. Com o Workbench conectado em uma base na qual deseja realizar o backup, acesse o menu à esquerda e selecione a opção Data Export (Figura 3.4), em que você encontrará duas opções de exportação de dados para o backup:

- *Export to Dump Project Folder*: selecionando esta opção, será gerado o backup no diretório selecionado. Para que o conteúdo que está em cada tabela na base de dados seja salvo, a opção "*Skip table data (no-data)*" deve ser desmarcada, do contrário, somente a estrutura das tabelas será salva. Esse recurso é ideal para quem deseja criar um banco de dados sem informações, usando somente a estrutura já criada (INFOLINK, 2018).
- *Export to Self-Contained File*: Selecionando esta opção será criado um código da base em somente um único arquivo .sql. Lembrando que para que as informações que estão na base de dados sejam salvos, a opção "*Skip table data (no-data)*" deve ser desmarcada, se não somente a estrutura das tabelas será salva (INFOLINK, 2018).

Ao escolher a opção desejada para realizar o backup, selecione *Start Export* para inicializar a construção do backup e salvar no diretório selecionado.

Figura 3.4 | Tela Principal Worckbench - Data Export



Fonte: captura de tela do MySQL Workbench 6.3, elaborada pelo autor.

Embora não seja uma regra que deva ser seguida impreterivelmente, é aconselhável que as operações de backup e *restore* sejam executadas quando não houver usuários acessando o banco de dados.

Deve-se ter em mente, ainda, que não é possível deixar a critério de uma única pessoa a execução da cópia de segurança, mas utilizar o método de cópia automática quando for oferecido. Nunca é demais ter duas ou três cópias, por exemplo, uma da imagem atual do banco de dados e duas anteriores. Isso facilita no caso de haver necessidade de recuperação de dados.

Outra forma um pouco mais dispendiosa, porém mais segura e que não depende de terceiros, é a utilização de discos rígidos espelhados, como na arquitetura RAID (*Redundant Array Inexpensive/Independent* - Conjunto Redundante de Discos Econômicos/Independentes), apresentada a seguir. Em ambientes cliente/servidor que rodam aplicações de missão crítica, é possível encontrar máquinas inteiras espelhadas, cuja arquitetura (hardware e software) permite que, se uma delas falhar, a outra entre em substituição automaticamente, sem que os usuários percebam qualquer anomalia (ALVES, 2014).



O Workbench pode ser usado também para restaurar um backup de um banco de dados, podendo ser restaurado em uma base nova ou substituindo uma existente.

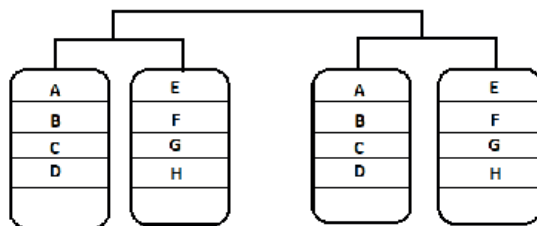
Você pode estudar como funciona a restauração assistindo ao vídeo a seguir:

LOCAWEB. Workbench: Backup e Restore de Banco de Dados. 16 mar. 2017. Disponível em: <https://www.youtube.com/watch?v=_gv2Vm0zlW4>. Acesso em: 2 ago. 2018.

RAID compreende um agrupamento de discos rígidos que funcionam de forma concomitante ou paralela, com o objetivo de reduzir os riscos de danos causados a arquivos e aumentar o desempenho no acesso aos dados. Os discos podem trabalhar independentemente ou de maneira sincronizada, com os dados espalhados entre eles. O RAID pode ser implementado por software ou hardware. Existem vários tipos de configurações disponíveis, denominadas de níveis. Abaixo teremos alguns exemplos de RAID, em que as letras representam os discos (ALVES, 2014):

- O RAID 0 + 1 exige pelo menos quatro discos na implementação, com espelhamento de cada par de disco e os pares representando o RAID nível 0 (Figura 3.5). Sua vantagem é que os dados sempre estarão protegidos e você ganha no desempenho, já a desvantagem é que necessita de mais discos, ou seja, um custo maior.

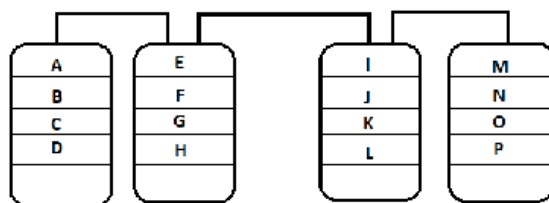
Figura 3.5 | Demonstração do espelhamento do RAID 0+1



Fonte: elaborada pelo autor.

- No RAID 0, são necessários ao menos dois discos. Nele, os dados são fragmentados em segmentos consecutivos, gravados sequencialmente em diferentes discos do conjunto. Os segmentos possuem um tamanho fixo, definido por blocos (Figura 3.6). O funcionamento do RAID 0 é considerado um dos mais simples, em que cada byte de informação é composto por um conjunto de unidades binárias (bits). Quando o sistema envia a instrução de gravar um byte num arranjo de discos em RAID 0, a informação é "fatiada" e distribuída, e cada uma das partes é gravada em um dos HDs do arranjo. Como todos os HDs trabalham simultaneamente, existe um ganho de velocidade durante os processos de leitura e a gravação de dados, e quanto mais discos, menos congestionamentos no processo.
- A penalidade na utilização desse nível de RAID é a total ausência de redundância da informação armazenada, ou seja, caso um dos discos utilizados no RAID tenha uma falha, todo o grupo de discos ficará indisponível (ALVES, 2014).

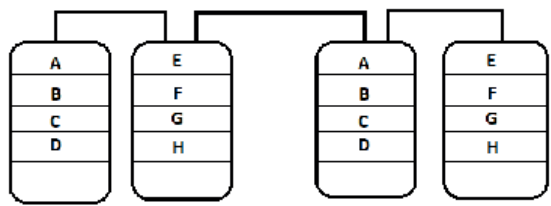
Figura 3.6 | Demonstração do espelhamento do RAID 0



Fonte: elaborada pelo autor.

- O RAID nível 1 também distribui os dados entre os discos, mas, nesse caso, além do espelhamento, há a duplicidade de discos, como ilustrado na Figura 3.7. Para implementação são necessários pelo menos dois discos. Sua vantagem é a redundância. Se tiver algum problema nos discos, o sistema continua funcionando, porém, você vai precisar de no mínimo dois HDs, reduzindo seu desempenho na escrita, pois o mesmo dado é gravado nos discos que estiverem em RAID nível 1 (ALVES, 2014).

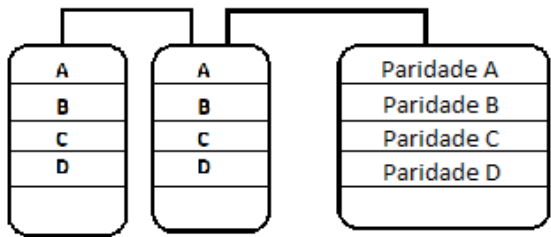
Figura 3.7 | Demonstração do espelhamento do RAID 1.



Fonte: elaborada pelo autor.

- O RAID nível 2 possui semelhança com o RAID 4, apresentado a seguir, mas exige um disco extra, em que são gravadas informações de controle de erros (ECC - *Error Correcting Code*). Esse tipo ficou obsoleto em virtude de os novos drives de disco rígido já possuírem esse controle no próprio circuito (Figura 3.8) (ALVES, 2014).

Figura 3.8 | Demonstração do espelhamento do RAID 2

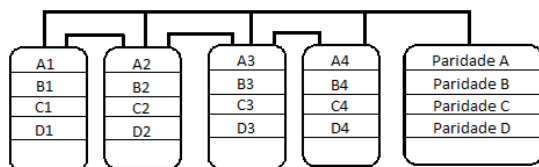


Fonte: elaborada pelo autor.

- O RAID nível 3 possui como característica principal a gravação paralela com paridade. O controle dos discos é bastante complexo, uma vez que se utiliza o menor tamanho possível para os segmentos de dados. Desta forma, é necessário que todos os discos tenham seus eixos perfeitamente sincronizados, a fim de evitar o atraso na transferência dos dados (ALVES, 2014).
- O RAID nível 4 exige um conjunto de discos iguais, com um mínimo de três unidades (Figura 3.9). Um dos discos é reservado para gravação das informações de paridade dos dados, como no RAID 3. Os discos de gravação de dados são configurados para armazenar segmentos grandes o

suficiente para conter um registro inteiro, o que permite a leitura independente dos dados. (ALVES, 2014).

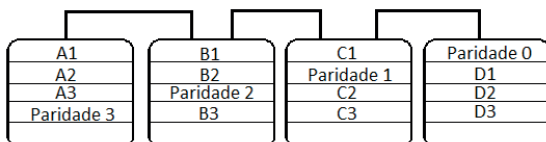
Figura 3.9 | Demonstração do espelhamento do RAID 4



Fonte: elaborada pelo autor.

- O RAID nível 5 tem funcionamento similar ao RAID 4, mas em vez de gravar as informações de paridade num disco extra, elas são distribuídas pelos discos do conjunto, gravando um ou mais bits em cada disco (Figura 3.10). Sua vantagem é a rapidez com que trata a leitura (porém a escrita não é tão rápida), e sua desvantagem é a sua montagem, pois existe um sistema complexo de controle de HDs (ALVES, 2014).

Figura 3.10 | Demonstração do espelhamento RAID 5.



Fonte: elaborada pelo autor.

Replicação de dados

A replicação de dados é usada para realizar cópias idênticas de um mesmo dado em vários servidores de banco de dados. Suas principais vantagens são a redundância, possibilitando um balanceamento de carga do sistema, já que o acesso do banco de dados pode ser distribuído entre suas réplicas, e ter um backup de dados, já que seus dados estão sendo sincronizados (ELMASRI,2005).

Uma replicação de dados também é conhecida como *Master-Slave*, em que temos um servidor atuando como *máster* e outros atuando como *slave*. Desta forma, todas as alterações ocorridas no servidor master são imediatamente replicadas para outros servidores.



O master é o responsável por registrar movimentações de seus dados em seu log binário (cada registro de evento é chamado de log binário). A cada transação de dados, o master registra as alterações em seu log binário e, em seguida, diz às ferramentas de armazenamento para salvarem as transações (ELMASRI, 2005).

O *slave* é responsável por copiar eventos do log binário do master em seu *relay log* (log de vigilância). O *slave*, por meio de um thread, abre uma nova conexão com o master e, em seguida, inicia o processo de esvaziamento de binlog. O binlog consiste na leitura do evento a partir do master (log binário) (ELMASRI, 2005).

A replicação de um banco de dados funciona da seguinte maneira:

- O servidor master registra movimentações do banco de dados em seu log binário.
- O *slave* copia os eventos registrados pelo master no log binário.
- O *slave* repete os mesmos eventos do log binário, um thread lê e repete todos os eventos, atualizando, assim, os dados do *slave* para ficarem idênticos aos do master.

Backups na Nuvem

Diferente do modelo tradicional de backup, em que os armazenamentos de informações eram feitos em mídias físicas como (CDs, HDs externos, pendrives, etc.) e alojados na própria empresa, o backup na nuvem é feito de forma externa, ou seja, é salvo em servidores remotos, por meio da internet, sendo protegido por empresas especializadas em manutenção e segurança de dados.

Um exemplo seria uma empresa ter o sistema local e o banco de dados na mesma rede do sistema. Em vez de realizar um backup manualmente na máquina, o próprio banco de dados tem um gerenciador que, em determinado momento do dia (uma vez ou mais), realiza um backup e envia para um servidor online, garantindo a integridade do banco de dados e podendo ser restaurado de qualquer outro lugar.

Vantagens do backup na nuvem

- **Segurança:** diminui a margem de risco de terceiros acessarem o banco de dados, por estar servidores mais robustos e gerenciados por especialistas em proteção de dados, além de ter um isolamento de conteúdos e protocolos de validação de acesso.
- **Flexibilidade:** pode ser realizado automaticamente, diminuindo a necessidade de armazenar em arquivos locais.
- **Economia de espaço:** apresenta a vantagem de liberar espaços que antes eram usados nos hardwares físicos das empresas.
- **Acessibilidade e mobilidade:** acesso a partir de qualquer local, bastando uma conexão da internet.

O MySQL Workbench disponibiliza um backup online, em que é possível gerenciar o processo do backup como o seu início de execução, suas informações sobre o estado do backup e todas operações realizadas no servidor.



Exemplificando

Uma empresa que tem diversas filiais espalhadas pelo Brasil, precisa acessar informações da matriz de maneira rápida. Como por exemplo consultar um estoque da matriz, se a informação não for atualizada, um vendedor pode vender um produto sem ao menos existir no estoque, podendo dar prejuízo na empresa. Com a replicação de dados, não haveria esse tipo de problema.

Com base no que aprendeu até agora, com certeza você tem conceitos claros em relação a estruturas de backups e recuperação de dados, a como usá-las no cotidiano profissional e qual é sua importância nas nossas vidas. Mas, isso não é tudo, então continue estudando, leia bastante e busque conhecimentos em várias fontes. Certamente, logo você se destacará na administração de bancos de dados.

Sem medo de errar

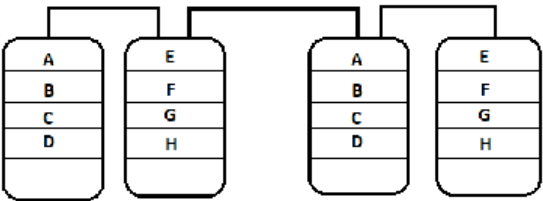
Caro aluno, conforme abordamos no início desta seção, nosso desafio é que você apresente para o Sr. Joaquin uma forma de melhorar a disponibilidade de suas informações e ajudá-lo a evitar que seu sistema fique off-line por muito tempo. Mesmo separando as duas áreas do sistema (produção e engenharia de produtos importados), ainda pode ocorrer uma queda no desempenho do sistema com o aumento dos acessos em uma das instâncias.

Para poder ajudá-lo, precisamos entender que, se uma das instâncias cair, ele precisa ter acesso a essas informações em outro servidor quase que instantaneamente. Para resolver mais este desafio, podemos utilizar a arquitetura de discos *Redundant Arrays of Inexpensive Disks* (RAID).

Com o uso do RAID, é possível combinar mais de um disco rígido, aumentando, assim, a performance do processo executado do servidor de banco de dados. Se um dos discos sofrer algum tipo de problema, outro começará a funcionar já no lugar com as informações do banco de dados. Com exceção do RAID 0, que não é recomendado para estruturas de banco de dados devido à falta de redundância, os demais são altamente recomendados.

No nosso exemplo, vamos utilizar o RAID 1 (Figura 3.11), já que ele apresenta redundância para as informações de dados e é recomendado para uso de banco de dados por causa de sua estrutura. Caso um disco do servidor falhe, os dados continuarão acessíveis. Ele também é excelente no espelhamento de disco, o que significa que as informações podem ser duplicadas de um disco para outro.

Figura 3.11 | Demonstração do espelhamento do RAID 1

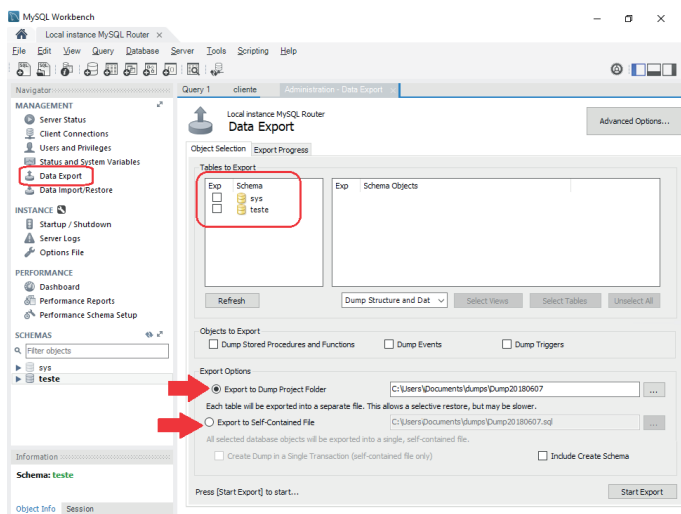


Fonte: elaborada pelo autor.

Utilizando uma estrutura RAID 1, o servidor do Sr. Joaquin não terá problemas de acessos ou de estruturas, pois, caso uma das estruturas das instâncias falhe, o RAID 1 irá imediatamente alocar outra espelhada, seja no mesmo servidor ou em outro da rede.

Mesmo assim, por segurança é recomendado que ele faça um backup das bases, caso uma delas corrompa e não replique informações erradas. Neste caso, vamos usar a ferramenta do MySQL Workbench, que possibilita gerar um backup de uma ou mais bases salvas no mesmo servidor. Para acessá-lo, é preciso estar conectado no banco de dados e acessar o menu à direita, “Data Export” (Figura 3.12), que apresenta as opções para gerar o backup dos dados. Em “export options”, é possível alterar o caminho do backup do banco de dados, assim a exportação pode ser salva diretamente em uma mídia externa ou driver virtual.

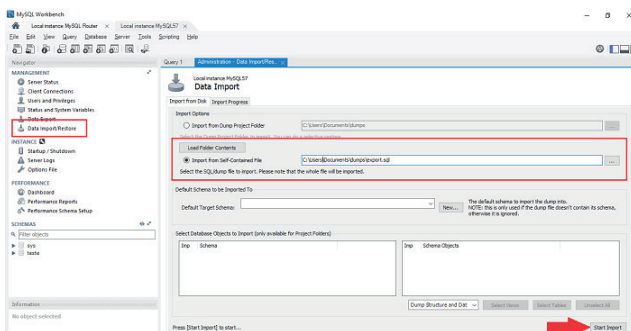
Figura 3.12 | Tela Principal do Workbench - Data Export



Fonte: captura de tela do MySQL Workbench 6.3, elaborada pelo autor.

Agora, caso ocorra de perder alguma informação ou o servidor fique fora por um período longo é preciso restaurar essas informações em outro servidor até normalizar o servidor principal. Neste caso, você usará o “Data Import” (Figura 3.12), que permite realizar o *restore* de um banco de dados usando o arquivo do backup criado anteriormente.

Figura 3.13 | Tela Principal Workbench - Data Import



Fonte: captura de tela do MySQL Workbench 6.3, elaborada pelo autor.

Avançando na prática

Distribuindo dados entre lojas

Descrição da situação-problema

Um empresário alugou duas salas para montar suas lojas em um shopping na região de São Paulo, para vendas de celulares e notebooks. Essas duas lojas estão em andares diferentes, estrategicamente pensando na visualização mais ampla de seus produtos, porém o estoque precisa ficar somente em uma delas, por que o espaço de uma é menor.

Pensando na situação desta loja, como seria realizada a manutenção do estoque desta loja. Como é possível que a loja sem estoque saiba que a loja com estoque tenha o produto disponível ou não para que não venda um produto inexistente? Como isso pode ocorrer, sendo que essas duas lojas usam o mesmo sistema, porém os bancos de dados são físicos em cada loja para caso de uma queda da internet. Qual seria a estrutura recomendada para que as duas tenham sempre um backup atualizado de suas informações.

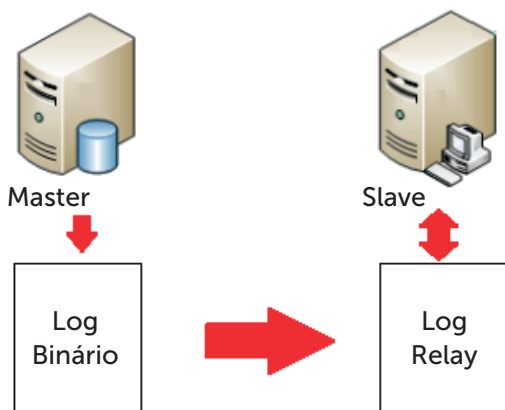
Resolução da situação-problema

Para resolvermos essa situação, o ideal é identificarmos qual seria o servidor principal, neste caso, o servidor da loja com estoque.

Com o servidor principal, é preciso construir uma arquitetura para replicar informações entre as lojas. Uma replicação de dados recomendada e também conhecida é o *Master-Slave*, em que temos um servidor atuando como master (no caso, a loja com estoque) e outro atuando como *slave* (loja sem estoque). Assim, todas as alterações ocorridas no servidor master são imediatamente replicadas para outro servidor. Essa replicação de um banco de dados funcionaria da seguinte maneira:

1. O servidor master registra movimentações do banco de dados em seu log binário.
2. O *slave* copia os eventos registrados pelo master no log binário.
3. O *slave* repete os mesmos eventos do log binário, um thread lê e repete todos os eventos, atualizando os dados do *slave* para ficarem idênticos aos do master, conforme a Figura 3.14.

Figura 3.14 | Demonstração da replicação Master-Slave



Fonte: elaborada pelo autor.

Faça valer a pena

1. Para mantermos a segurança do banco de dados, precisamos guardar essas informações em outros locais, seja outra unidade de disco, outro servidor, um pendrive ou em nuvens, para que possamos garantir a preservação e recuperação de informações ou do banco de dados inteiro.

Qual seria a frequência recomendada para realizar um backup?

- a) Pelo menos uma vez por mês.
- b) Duas vezes ao mês.
- c) Uma vez por semana.
- d) A cada dois dias.
- e) Todos os dias.

2. A replicação de dados serve para gerar cópias idênticas dos mesmos dados para vários servidores de banco de dados, em que o master seria o servidor principal e o slave seria um ou mais servidores que irão receber informações do servidor master.

Sobre replicação de dados, quais seriam suas principais vantagens?

- a) Equilíbrio de carga, distribuição de dados, backup, alta disponibilidade.
- b) O aumento de *checkpoints* pode melhorar o desempenho do serviço replicado.
- c) O cliente pode retransmitir requisição entre o ultimo *update* e a falha do primário.
- d) O mecanismo de log armazena em disco todas as requisições.
- e) Pouca complexidade.

3. RAID é sigla de *Redundant Array Inexpensive/Independent Disk* (Conjunto Redundante de Discos Econômicos/Independentes) e compreende um agrupamento de discos rígidos que funcionam de forma concomitante ou paralela, com o objetivo de reduzir os riscos de danos causados a arquivos e aumentar o desempenho de acesso aos dados.

Com base no conceito do RAID, por que o RAID nível 2 não é mais usado?

- a) Esse tipo ficou obsoleto em virtude de os novos drives de disco rígido já possuírem esse controle no próprio circuito.
- b) O RAID nível 3 é melhor, assim, não é mais usado o RAID nível 2.
- c) Ele exige um conjunto de discos iguais, aumentando seu custo no servidor.
- d) A gravação de dados nele é lenta.
- e) O seu controle dos discos é bastante complexo, uma vez que se utiliza o menor tamanho possível para os segmentos de dados.

Seção 3.3

Banco de dados na nuvem

Diálogo aberto

Olá, aluno! Vamos a mais uma etapa de nosso aprendizado.

Embora o termo “nuvem” seja relativamente novo, o conceito surgiu em 1961, nos Estados Unidos, por McCarthy. Na realidade, ele gostaria que a internet fosse oferecida como um serviço público, que você pagaria pelo seu consumo, assim como faz com os serviços de água, luz e telefonia (IMASTERS, 2018). Hoje em dia, existem inúmeras vantagens de usar o armazenamento na nuvem em comparação ao armazenamento de um servidor local, entre elas: a possibilidade de acesso remoto ao banco de dados; atualização automática de softwares ou upgrade de hardware quase imediato, sem precisar reiniciar o servidor; segurança das informações; e, o mais chamativo para os empresários, redução de custos, pois a empresa tira de suas costas a responsabilidade da segurança e custo com servidores (energia elétrica, internet e aplicativos pagos para segurança como por exemplo antivírus).

Atualmente, a computação na nuvem é muito importante, tanto profissionalmente como no âmbito pessoal. Um exemplo são os smartphones, que, apesar de potentes, ainda não têm muita capacidade de armazenamento, fazendo com que arquivos, aplicações e dados sempre estejam salvos na nuvem. Essas mesmas informações podem ser resgatadas a partir do mesmo smartphone ou de outro computador que tenha acesso a internet. Resultados de exames, acesso a informações de carro, compra de passagem de avião, rastreamento de entrega de mercadoria dos correios. Todos estão envolvidos com servidores na nuvem.

Estamos chegando ao final de mais uma unidade de estudo, em que fizemos uma contextualização dos recursos avançados em banco de dados. E, agora que você já conhece um pouco mais acerca dos conceitos que alicerçam esse assunto, está na hora de entendermos como funciona o gerenciamento de banco de dados na nuvem, desde seu conceito até questões relacionadas à

segurança, e quando devemos usar um banco de dados na nuvem em vez de um servidor convencional dentro de uma corporação.

Com base no que iremos aprender, você poderá ajudar o Sr. Joaquin que, passado o feriado e os sistemas terem aguentado o número de acessos, continua preocupado com seu servidor, pois, caso aconteça algum problema de performance no servidor ou haja algum dia propício a aumento de acessos, ele demoraria um certo tempo para adequar as configurações para outro servidor e subir os dados atualizados. Esse tempo pode custar caro, pois seria um tempo em que o servidor ficaria off-line. Ele está disposto a comprar um servidor de última geração para resolver problema de performance, mas a internet ainda continua sendo um obstáculo, caso ela caia durante o acesso. Esse servidor de última geração pode resolver boa parte dos problemas nos dias de maior movimentação no sistema, mas, no geral, ele seria desnecessário, já que seu seria custo muito alto em relação ao uso em determinadas datas. Apresente ao Sr. Joaquin um relatório com informações sobre como ele poderia resolver essa situação sem ter que comprar um novo servidor, bem como o problema com a queda da internet. Exemplifique as situações em que um servidor na nuvem pode ajudar no seu dia a dia no lugar de um servidor local, e quais seriam suas vantagens. Crie um servidor na nuvem e, nele, um novo banco de dados, apresentando para o Sr. Joaquin suas facilidades.

Para auxiliá-lo nesse desafio, você vai entender o que é e para que serve um banco de dados na nuvem, suas vantagens e como gerenciá-los de maneira adequada. Com seu esforço nos estudos e sua participação nas aulas, você estará apto a passar por mais um desafio. Vamos lá!

Não pode faltar

Na era dos grandes volumes de dados em que vivemos, usuários comuns agora são fontes de informação e empresas armazenam incontáveis informações de clientes, milhões de sensores monitoram o mundo real, criando e trocando informações em um mundo virtual.

Um banco de dados na nuvem é caracterizado pela existência de diversas máquinas servidoras interligadas pela internet, com o intuito de garantir uma alta disponibilidade de dados, com segurança e um bom nível de desempenho, além de garantir backups automáticos.

Cada vez mais bancos de dados nas nuvens estão sendo usados por diversos tipos de usuários, de diferentes setores, desde empresas menores que desejam reduzir custos com infraestrutura, sistema e licenças, até as maiores empresas, que contratam soluções prontas para gerenciar diversos computadores, consequentemente permitindo um aumento inesperado no tráfego (ABADI, 2009).

Com a evolução da Internet, novos volumes vêm sendo gerados, porém, os bancos de dados atuais ainda não se demonstram preparados para manipularem da melhor maneira uma quantidade de dados de grande magnitude (ABADI, 2009).



Exemplificando

O Microsoft Azure e o Amazon Web Services (AWS) são as plataformas de serviços em nuvem mais populares da atualidade. Elas garantem redundância, segurança e velocidade de dados, equiparando-se a uma rede local. Esses tipos de serviços são considerados como servidores dedicados, pois não compartilham informações de hardware nem de software com outros servidores. Existem outras empresas que também trabalham com servidores dedicados, porém, com poucas ferramentas de gerenciamento e, consequentemente, baixo custo. Nesses servidores, o banco de dados é único e pode ser instalado mais de um tipo de banco de dados sem que haja interferência nos demais servidores.

Outros tipos de servidores que são de baixo custo são os servidores compartilhados, onde um mesmo servidor armazena diversos sistemas operacionais. Consequentemente não é possível alterar configurações desses servidores e o servidor do banco de dados acaba sendo um só para todos, ou seja, você consegue visualizar todos os usuários daquele mesmo servidor.

No geral, podemos definir a migração de dados na nuvem como um grupo de informações armazenadas na web, que pode ser manipulado e gerenciado usando gerenciador de banco de dados

na nuvem (SGDN). Esses bancos de dados são criados para fazer o processamento paralelo diante de diversas tarefas simultâneas e utilizar em recursos distribuídos (ABADI, 2009).

Nos dias de hoje, as velocidades de internet favorecem que as empresas migrem para um banco de dados na nuvem.

Um estudo realizado pelo site DB-Engines mostra que os principais bancos de dados vêm aumentando consideravelmente o número de acessos em relação ao ano anterior (Figura 3.15) (DB-ENGINES, 2018). Os cinco primeiros bancos de dados listados estão em alta por causa de seus acessos em servidores na nuvem, conseguindo manter a alta performance mesmo quando acessados pela internet. Esses bancos de dados (Oracle, MySQL, Microsoft SQL Server, MongoDB e PostgreSQL) são responsáveis pela maioria das migrações dos bancos de dados locais para os bancos de dados na nuvem.

Figura 3.15 | Ranking de banco de dados mais usados em 2017 segundo db-engines.com

Rank			DBMS	Database Model	Score		
Jan 2017	Dec 2016	Jan 2016			Jan 2017	Dec 2016	Jan 2016
1.	1.	1.	Oracle 🟡	Relational DBMS	1416.72	+12.32	-79.36
2.	2.	2.	MySQL 🟡	Relational DBMS	1366.29	-8.12	+67.03
3.	3.	3.	Microsoft SQL Server	Relational DBMS	1220.95	-5.70	+76.89
4.	📈 5.	4.	MongoDB 🟡	Document store	331.90	+3.22	+25.88
5.	📉 4.	5.	PostgreSQL	Relational DBMS	330.37	+0.35	+47.97
6.	6.	6.	DB2	Relational DBMS	182.49	-1.85	-13.88
7.	7.	📈 8.	Cassandra 🟡	Wide column store	136.44	+2.16	+5.49
8.	8.	📉 7.	Microsoft Access	Relational DBMS	127.45	+2.75	-6.59
9.	9.	📈 10.	Redis 🟡	Key-value store	118.70	-1.20	+17.54
10.	10.	📉 9.	SQLite	Relational DBMS	112.38	+1.54	+8.64
11.	11.	📈 12.	Elasticsearch 🟡	Search engine	106.17	+2.90	+28.96
12.	12.	📈 14.	Teradata	Relational DBMS	74.17	+0.79	-0.78
13.	13.	📉 11.	SAP Adaptive Server	Relational DBMS	69.10	-1.32	-14.08
14.	14.	📉 13.	Solr	Search engine	68.08	-0.92	-7.32
15.	15.	📈 16.	HBase	Wide column store	59.14	+0.51	+5.77

Fonte: captura de tela do DB-Engines.com.

Os bancos de dados na nuvem possuem tecnologias com nível elevado de escalabilidade, o que faz com que os aplicativos evoluam de acordo com sua utilização. De fato, podemos dizer que, ao usarmos um banco de dados na nuvem, teremos algumas vantagens, se comparado com o banco de dados convencional. Entre elas, podemos destacar a escalabilidade/flexibilidade, a tecnologia utilizada, o acesso às informações em qualquer lugar (as informações são armazenadas de forma distribuída) e o menor custo do servidor.



Um banco de dados na nuvem pode lhe proporcionar maior agilidade e flexibilidade, significando que você terá menos tempo gasto com rotinas e administração.

O gerenciamento de informações do banco de dados é um fator importantíssimo dentro do contexto de computação em nuvem, uma vez que sua segurança é fator crucial em ambientes na nuvem e tem que ser manipulado com certo cuidado e atenção, já que estamos lidando com informações confidenciais que não podem ser acessadas por qualquer usuário. Um outro item importante é que os Sistemas de Gerenciamento de Banco de Dados (SGBD) não possuem escalabilidade quando um banco de dados tem um volume elevado de dados armazenados (ABADI, 2009).

O armazenamento de informações, tempo de carregamento de consultas e manipulação de dados têm sido flexibilizado por algumas abordagens para garantir a eficiência do banco de dados, entretanto, ainda não existe uma solução que combine esses aspectos para melhorar o processamento sem perder a consistência das informações (ABADI, 2009).

Em um servidor na nuvem, a escalabilidade ocorre quando o mesmo tem a capacidade de aumentar conforme o fluxo de dados. Já sua disponibilidade de serviço possibilita aos usuários o acesso às informações em qualquer momento ou lugar, onde quiserem ou precisarem, e, na maioria dos Sistemas de Gerenciamento de Dados na nuvem, ele tem que ser de alta disponibilidade, visto que o tipo de comunicação entre o usuário e o aplicativo é a internet e que pode ocorrer algum tipo de atraso ou indisponibilidade do sistema usado.

Um outro item que também é relevante no gerenciamento de dados em nuvem é a consistência das informações, o que significa que todos os usuários devem ter a mesma visão das informações ao acessá-las, por exemplo, se uma determinada informação do banco de dados for atualizada, então ele tem que ser atualizado para todos os sistemas, assim mantendo as informações do sistema íntegras, consistentes e coerentes.

Também existem outras ferramentas que nos ajudam a gerenciar dados na nuvem, algumas das mais relevantes serão apresentadas a seguir.

Sistema de Gerenciamento de Dados na Nuvem (SGDN)

De acordo com Sousa (2010), uma infraestrutura de um Sistema de Gerenciamento de Dados em Nuvem (SGDN) possui várias vantagens para o usuário, tais como:

- Baixo custo.
- Previsibilidade referente às cargas de trabalho reais e à qualidade de serviço.
- Complexidade técnica diminuída, graças à interface facilitada de acessos unificados a uma administração de Sistema de Gerenciamento de Dados.
- Delegação de *tuning* com escalabilidade e elasticidade, proporcionando uma percepção de recursos quase ilimitados.

O Banco de dados na nuvem nos proporciona diversas vantagens, porém, existem diversos desafios que precisam ser superados nesse ambiente, entre eles podemos destacar a segurança da informação, a consistência, a escalabilidade e a qualidade do tráfego de dados pela internet.

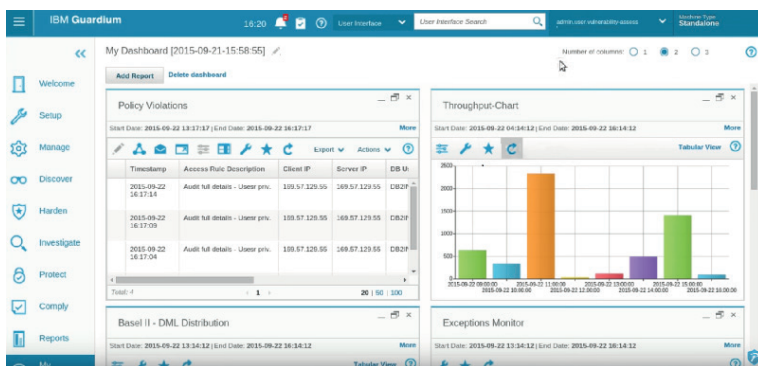
Segurança

Quando falamos de banco de dados na nuvem, pensamos em diversos benefícios, como backup, acessibilidade, flexibilidade e custo. Contudo, basearmos uma estrutura em uma nuvem remota, a empresa acaba liberando dados e informações privadas, coisas que podem ser confidenciais. Essas informações acabam sendo passadas para o gerenciador de nuvens que fica responsável em proteger e cuidar dos dados, dando uma sensação de dependência do serviço comprado. Algumas empresas acabam não adquirindo um banco de dados na nuvem com receio de perder informações, outras adquirem serviços mais simples de servidor na nuvem e acabam tendo dificuldades técnicas diárias, como lentidão de acesso ou, até mesmo, interrupções de conexão, correndo o risco de corromper o banco de dados.

Um problema grave no banco de dados na nuvem, é a tentativa de acessos de pessoas não autorizadas. Hackers tentam acessar banco de dados através de portas liberadas no servidor, buscando uma forma de conseguir uma permissão de acesso ao banco de dados. Para podermos acessar um banco de dados fora do servidor, é preciso liberar uma porta de acesso, e é com essa porta que os hackers tentam acessar o banco de dados. Para impedir esse tipo de ataque, é recomendado sempre liberar a porta do servidor para um IP fixo, assim evita-se que outros IPS tentem acessar. Outra forma é localizar o IP que está acessando e bloqueá-lo através do sistema operacional ou de outros softwares.

O IBM Guardium (Figura 3.16) é um exemplo de como combater esse tipo de acesso por meio de um monitoramento das atividades suspeitas no banco de dados, como um número de acessos anormal, caso em que é enviado um e-mail para o DBA informando o ocorrido.

Figura 3.16 | IBM Guardium 10 Dashboard



Fonte: captura de tela do IBM Guardium, adaptada pelo autor.

Com o IBM Guardium, é possível analisar informações, em tempo real, sobre quais bancos têm mais acessos ou quais comandos sendo executados estão demorando para serem finalizados. Esses recursos podem ser visualizados dentro do *Throughput-char* e podem ser filtrados também por períodos, sabendo em quais dias um banco de dados tem mais acessos, assim os administradores podem analisar e montar uma estratégia para melhorar performance do servidor nesses períodos.

Em Policy Violation, são exibidos os acessos ou tentativas de acesso no banco de dados. É muito útil para visualizar usuários que estão tentando se conectar sem permissão. Quando isso ocorre, o DBA tem que bloquear o acesso, por meio da porta no servidor, ou o IP que está tentando acessar.

Escalabilidade e consistência dos dados

Os bancos de dados na nuvem são muito focados em escalabilidade, oferecendo uma fraca consistência de seus dados armazenados. Eles devem ser escaláveis, pois não há uma quantidade definida para ser armazenada, e sua escalabilidade proporcionará o crescimento do sistema de acordo com a demanda, mas isso não garante a consistência das suas informações (CASTRO, 2011).

Grande parte dos bancos de dados na nuvem tem uma consistência fraca, portanto, a informação do banco de dados se torna importante, devendo ser analisada e tratada ao se projetar novas aplicações ou soluções de banco de dados (CASTRO, 2011).



Reflita

O que acontece quando um servidor na nuvem para de funcionar? Os acessos são interrompidos? A maioria dos servidores da nuvem possui uma estrutura de espelhamento, ou seja, se um servidor ficar inacessível, outro será acessado imediatamente, de modo que o usuário não perceba que está acessando outro servidor.

Qualidade do serviço de dados

A qualidade dos serviços proporciona aos usuários algumas garantias de desempenho e disponibilidade. A disponibilidade permite aos usuários acessarem a nuvem quando e onde desejarem. Outra vantagem é a alta disponibilidade das aplicações devido ao fornecedor de serviço possuir uma estrutura em que um servidor assume automática e imediatamente, caso outro serviço pare de funcionar.

A elasticidade também é contada com uma vantagem, por que, em determinados períodos de maior atividade, o usuário pode

reduzir ou aumentar a quantidade de recursos contratados do seu servidor. Esses processos são simples, rápidos e não requerem a reinicialização do sistema operacional.



Pesquise mais

Para entender mais sobre um dos servidores de banco de dados mais conhecidos no mercado, o Microsoft Azure, acesse o link abaixo:

ARAUJO, L. **Artigo SQL Magazine 72** - Bancos de dados em nuvem com AQL Azure. 2010. Disponível em: <<https://www.devmedia.com.br/artigo-sql-magazine-72-bancos-de-dados-em-nuvem-com-sql-azure/15644>>. Acesso em: 6 ago. 2018.

Nele é apresentado um paralelo entre a computação em nuvem e os serviços de banco de dados na internet, bem como algumas das características suportadas pela ferramenta, de forma prática.

Criando um banco de dados em um servidor na nuvem

Antes de tudo, precisamos pesquisar quais servidores na nuvem existem no mercado e o que eles têm a oferecer de melhor para o nosso uso cotidiano. A escolha é essencial porque nele vamos armazenar informações que poderão ser sigilosas, então dependemos muito da sua segurança.

O servidor também precisa ter as ferramentas necessárias para o nosso trabalho no dia a dia, para não dependermos de ferramentas de terceiros ou precisarmos futuramente contratar outro servidor só para atender nossas necessidades. Um exemplo é contratar um servidor que apenas tenha acesso a um banco de dados MySQL e, futuramente, precisar incluir outro servidor, só que SQL server, e esse servidor não atender ao requisito. Mas, como criar um banco de dados na nuvem?

Primeiramente vamos usar um servidor online chamado 000webhost, que pode ser acessado através do site <www.000webhost.com/>. Esse servidor possui um serviço gratuito e oferece um banco de dados MySQL ilimitado, mas há a limitação de dois bancos por conta. Para criar uma conta,

é necessário acessar o site e clicar em **free sign up** (<https://www.000webhost.com/free-website-sign-up>), que redirecionará para a tela de registro (Figura 3.17), em que será pedido um e-mail, uma senha e um nome de website (o website não é obrigatório no nosso caso), e o site irá gerar uma url aleatória com o subdomínio .webhost.com. Use um endereço de e-mail válido, pois será enviado um e-mail de validação e será necessário clicar no link para poder abrir o painel de controle do servidor.

Figura 3.17 | Tela de login do site webhost.com

The screenshot shows a registration form titled "Start FREE Sign UP!". Below the title, it says "It's 100% FREE". The form contains three input fields: an email field with the placeholder "teste@teste.com.br", a password field with six dots, and a "Website Name" field. Below the email field, there is a red error message: "The email field is required." Below the password field, there is a red error message: "The password field is required." Below the website name field, there is a link that says "Like us on Facebook". At the bottom of the form, there is a large red button with the text "GET FREE HOSTING" and a checkmark icon. Below the button, there is a small text line: "By proceeding you agree to our Terms of Service".

Fonte: captura de tela do site <www.000webhost.com>.

Depois de realizado o cadastro e validado o e-mail, você será direcionado à tela de painel do seu servidor de testes (Figura 3.18). Nela, clique em **Manage Website** para acessar o painel do nosso servidor.

Figura 3.18 | Tela de acesso para o gerenciamento do servidor

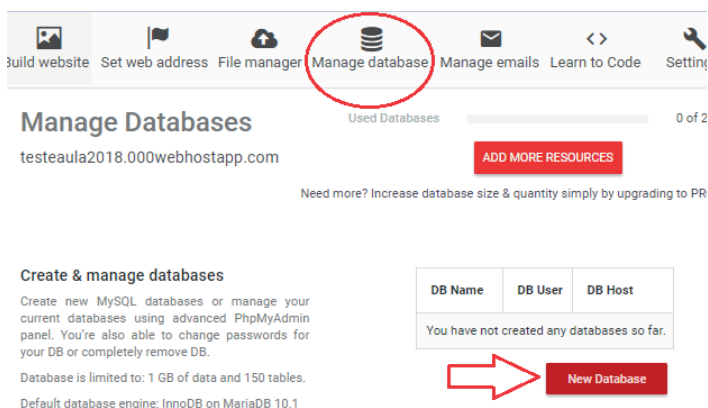
The screenshot shows a dashboard for a website named "reservable". Below the name, it says "Status: Running" with a green dot. At the bottom, there are two buttons: "Manage Website" (in red) and "Details" (in grey). Below the buttons, there is a URL: "https://reservable.000webhostapp...".

Fonte: captura de tela do site <www.000webhost.com>.

Na tela do painel de gerenciamento do servidor, realize os seguintes passos para criar um banco de dados em nuvem:

1. Acesse o *Manage database*, em que criaremos nosso primeiro banco de dados em um servidor da nuvem.
2. Ao acessá-lo, clique em *New Database* (Figura 3.19) para criarmos um novo banco de dados.
3. Para criar o banco, ele pedirá um nome do banco, nome do usuário e uma senha de acesso.
4. Criado o novo banco, ele aparecerá no painel, no centro da tela, com o seu nome e usuário, junto com um ID criado pelo servidor, que é concatenado com o usuário e o nome que você criou.

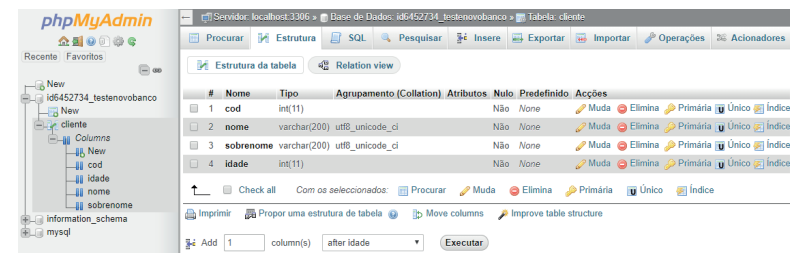
Figura 3.19 | Painel de gerenciamento do servidor



Fonte: captura de tela do site <www.000webhost.com>.

Para acessar o banco de dados, basta clicar em **Manage** e, depois, em **phpMyAdmin**. Nele, é necessário fornecer novamente o seu usuário e senha (lembrando que o usuário precisa ter o ID concatenado que foi gerado pelo servidor). Dentro do phpMyAdmin, você poderá administrar seu banco de dados, criando tabelas ou até mesmo importando uma base já existente. Com isso, temos o nosso banco de dados na nuvem, que podemos acessar de qualquer lugar.

Figura 3.20 | Painel do phpMyAdmin



Fonte: captura de tela do site <www.000webhost.com>.

Com base no que aprendeu até agora, você com certeza tem conceitos claros em relação ao banco de dados na nuvem, a como usá-lo no cotidiano profissional e a qual sua importância nas nossas vidas. Mas isso não é tudo, então continue estudando, leia bastante e busque conhecimentos em várias fontes.

Sem medo de errar

Depois de ter passado o feriado de os sistemas terem aguentado o número de acessos, o Sr. Joaquin ainda continua preocupado com seu servidor, por que, caso ocorra algum problema de performance no servidor ou haja algum dia propício ao aumento de acessos, ele demoraria um certo tempo para adequar as configurações para outro servidor e subir os dados atualizados. Esse tempo pode custar caro, pois seria um tempo em que o servidor ficaria off-line. Ele está disposto a comprar um servidor de última geração para resolver problema de desempenho, mas a internet ainda continua sendo um obstáculo, caso ela caia durante o acesso. Esse servidor de última geração pode resolver boa parte dos problemas nos dias de maior movimentação no sistema, mas, no geral, seria desnecessário, já que o custo é alto para usar somente em determinadas datas. Apresente ao Sr. Joaquin um relatório com informações sobre como ele poderia resolver essa situação sem ter que comprar um novo servidor, mas com possíveis soluções para a queda da internet, exemplificando situações em que um servidor na nuvem pode ajudar no seu dia a dia em vez de usar um servidor, bem como suas

vantagens. Crie um servidor na nuvem e, nele, um novo banco de dados, apresentando ao seu cliente suas facilidades.

Para podermos ajudar o Sr. Joaquin, é preciso explicar que hospedar um banco de dados na nuvem significa que suas informações serão gerenciadas por uma empresa especializada no assunto. Mostre que, de acordo com o PwC, 48% das empresas já utilizam a nuvem para armazenamento de informações.

Além de ser gerenciada por uma empresa especialista, o custo é até 40% menor, como é informado no link indicado. Essa diminuição de custo se dá por causa da infraestrutura, energia elétrica e a manutenção no próprio servidor. Com o servidor na nuvem, a preocupação seria somente com tipo de plano adquirido pela empresa. Uma outra vantagem é a elasticidade do servidor, sendo possível aumentar ou diminuir recursos quase que instantaneamente, sem ter que reiniciar um servidor. Essas alterações podem ser solicitadas em alguns servidores na nuvem, com o próprio usuário fazendo alterações por meio de um painel específico, como no caso da Microsoft Azure.

Para podermos apresentar para o cliente o funcionamento de um banco de dados na nuvem, ele primeiramente precisa ter um servidor na nuvem. Para que ele entenda melhor sua funcionalidade, crie uma conta, conforme aprendemos no site www.000webhost.com.

Para criar uma conta no servidor 000webhost, você precisa acessar o site e clicar em **free sign up** (<https://www.000webhost.com/free-website-sign-up>) para ser redirecionado para a tela de registro, em que será pedido um e-mail, uma senha e um nome de um website, mas o website não é obrigatório no nosso caso. Desta forma, o site irá gerar uma url aleatória com o subdomínio ".webhost.com". Use um e-mail válido, pois será enviado um e-mail de validação e será necessário clicar no link para poder abrir o painel de controle do servidor.

Na tela do painel de gerenciamento do servidor, realize os seguintes passos para criar um banco de dados em nuvem:

1. Acesse *Manage database*, em que vamos criar nosso primeiro banco de dados em um servidor da nuvem.

2. Ao acessá-lo, clique em *New Database* para criarmos um novo banco de dados.
3. Será solicitado um nome do banco, nome do usuário e uma senha de acesso.
4. Criado o novo banco, ele aparecerá no painel no centro da tela com o seu nome e usuário, junto com um ID criado pelo servidor, concatenado com o usuário e o nome que você criou.

Agora, para acessar o banco de dados, basta em **Manage** e depois em **phpMyAdmin**, e você será redirecionado para o phpMyAdmin, em que é necessário fornecer novamente o seu usuário e senha (lembrando que o usuário precisa ter o ID concatenado que foi gerado pelo servidor).

Avançando na prática

Mudança de servidor

Descrição da situação-problema

Tiago, dono de uma empresa de vendas de carros online, tem um sistema web com o servidor de banco de dados MySQL na própria loja física. Certo dia, ele precisou mudar de cidade para uma loja maior, pois as vendas de carros estavam indo bem e, consequentemente, ele precisou transferir o seu servidor de local. Só que, para a surpresa de Tiago, a fiação da internet ainda não tinha chegado na sua nova loja e ele corria o risco de ficar sem a internet. Apresente para o Tiago uma maneira de continuar com o seu sistema web independentemente de o servidor ser local ou de ter uma internet na empresa.

Resolução da situação-problema

Para podermos ajudar o Tiago, ele primeiramente precisa ter um servidor na nuvem. Para que ele entenda melhor sua funcionalidade, crie uma conta conforme aprendemos no site www.000webhost.com.

Para criar uma conta no servidor 000webhost, você precisa acessar o site e clicar em **free sign up** (<https://www.000webhost.com>).

com/free-website-sign-up) para ser redirecionado para a tela de registro, em que será pedido um e-mail, uma senha e um nome de um website, mas o website não é obrigatório no nosso caso. Desta forma, o site irá gerar uma url aleatória com o subdomínio “.webhost.com”. Use um e-mail válido, pois será enviado um e-mail de validação e será necessário clicar no link para poder abrir o painel de controle do servidor.

Na tela do painel de gerenciamento do servidor, realize os seguintes passos para criar um banco de dados em nuvem:

1. Acesse *Manage database*, em que vamos criar nosso primeiro banco de dados em um servidor da nuvem.
2. Ao acessá-lo, clique em *New Database* para criarmos um novo banco de dados.
3. Será solicitado um nome do banco, nome do usuário e uma senha de acesso.
4. Criado o novo banco, ele aparecerá no painel no centro da tela com o seu nome e usuário, junto com um ID criado pelo servidor, concatenado com o usuário e o nome que você criou.

Para acessar o banco de dados, basta clicar em **Manage** e depois em **phpMyAdmin**, para ser redirecionado para o phpMyAdmin, em que é necessário fornecer novamente o seu usuário e senha (lembrando que o usuário precisa ter o ID concatenado que foi gerado pelo servidor). Dentro do phpMyAdmin, é preciso somente importar o banco de dados para dentro dele, assim todas as informações serão salvas na nuvem.

Uma observação é que, com esse servidor dado como exemplo, Tiago não teria custo algum, se o banco de dados dele for relativamente pequeno e ele não precisar de nenhuma modificação para melhorar a performance do seu servidor. Mas, caso ele precise de um servidor escalável, é necessário adquirir um servidor dedicado, que ele mesmo possa administrar, incluindo ou removendo recursos do servidor.

Faça valer a pena

1. A qualidade dos serviços proporciona aos usuários algumas garantias de desempenho e disponibilidade. A disponibilidade permite aos usuários acessarem a nuvem quando e onde desejarem. Outra vantagem é a alta disponibilidade das aplicações, devido ao fornecedor de serviço possuir uma estrutura que, caso um serviço pare de funcionar, o outro assume imediatamente, de forma automática.

Além da garantia de desempenho e da disponibilidade, qual seria outra vantagem no quesito qualidade no banco de dados?

- a) A elasticidade.
- b) O acesso.
- c) Estrutura do servidor estática, não sendo possível modificar nada.
- d) Acesso à rede limitado.
- e) Banco de dados estático com tamanho fixo.

2. Hoje em dia, a migração do gerenciamento de dados dos servidores físicos e locais para a nuvem é uma tendência que, a cada ano, mais se desenvolve nas empresas. A maioria das empresas realiza essa migração pensando estrategicamente no futuro.

De acordo com o enunciado, quais seriam as três principais vantagens que os empresários teriam ao migrar seus servidores físicos para a nuvem?

- a) Flexibilidade, custo e acessibilidade.
- b) Backup e facilidade nas consultas.
- c) Velocidade nas consultas, e facilidade no acesso ao banco.
- d) Atualizações automáticas de backup.
- e) Servidor mais seguro e com mais espaço.

3. Um item relevante no gerenciamento de dados em nuvem é a consistência das informações, que significa que todos devemos ter a mesma visão das informações ao acessá-las, sendo, assim, todas as informações armazenadas as mesmas para todos no sistema. Por exemplo, se uma determinada informação do banco de dados for atualizada, então ele tem que ser atualizado para todos os sistemas, mantendo as informações do sistema íntegras, consistentes e coerentes.

Além da consistência de informações, eventualmente, uma empresa precisa realizar upgrades em seu servidor para que possa manter um sistema com qualidade. Qual seria o maior problema, que diferenciaria o servidor local e o servidor na nuvem, quando é necessário realizar uma atualização?

- a) Disponibilidade.
- b) Velocidade da Internet.
- c) Backup instantâneo.
- d) Controle de usuário.
- e) Base de dados estática.

Referências

ABADI, D. J. **Data management in the cloud: Limitations and opportunities**. Universidade de Yale. 2009. Disponível em: <<http://www.cs.yale.edu/homes/dna/papers/abadi-cloud-ieee09.pdf>>. Acesso em: 7 ago. 2018.

ACTIVE SOLUTION. **A importância de fazer backup de dados em sua empresa**. [S.d.]. Disponível em: <<http://www.activesolutions.com.br/blog/a-importancia-de-fazer-backup-de-dados-em-sua-empresa/>>. Acesso em: 7 ago. 2018.

ALVES, W. P. **Banco de dados**. 1. ed. São Paulo: Érica, 2014.

MICROSOFT. **Microsoft Azure**. [S.d.]. Disponível em: <<https://azure.microsoft.com/pt-br/>>. Acesso em: 14 jun. 2018.

APACHE. **Cassandra**. [S.d.]. Disponível em: <<http://cassandra.apache.org/>>. Acesso em: 7 ago. 2018.

CASTRO, R. C. C; PIMENTEL, V. L. **Segurança em Cloud Computing: Governança e Gerenciamento de Riscos de Segurança**. Disponível em: <http://www.academia.edu/7520311/Seguran%C3%A7a_em_Cloud_Computing_Governan%C3%A7a_e_Gerenciamento_de_Riscos_de_Seguran%C3%A7a>. Acesso em: 9 ago. 2018.

COUCHDB. **The CouchDB Project**. [S.d.]. Disponível em: <<http://couchdb.apache.org>>. Acesso em: 14 de jun. 2018.

DB-ENGINES. **Ranking de acesso de banco de dados**. [S.d.]. Disponível em: <<https://db-engines.com/en/ranking>>. Acesso em: 7 ago. 2018.

DILLON, T.; CHANG, C. **Cloud Computing: Issues and Challenges**. 24th IEEE International Conference on Advanced Information Networking and Applications, 2010.

ELMASRI, R. **Sistemas de banco de dados**. 4. ed. São Paulo: Pearson Education, 2005.

IMASTERS. **Uma breve história da cloud computing**. 20 jan. 2017. Disponível em: <<https://imasters.com.br/cloud/uma-breve-historia-da-cloud-computing>>. Acesso em: 7 ago. 2018.

INFOLINK. **Como efetuar um backup do banco de dados Mysql**. [S.d.]. Disponível em: <https://wiki.infolink.com.br/Como_efetuar_um_Backup_do_banco_MySQL_com_o_Mysql_Workbench>. Acesso em: 7 ago. 2018.

NISHIMURA, R. Y. **Banco de dados I**. São Paulo: Pearson Education, 2009.

Fundamentos de banco de dados não-convencionais

Convite ao estudo

Chegamos à última unidade da nossa disciplina, em que você, aluno, aprenderá sobre fundamentos de banco de dados não convencionais.

Vamos entender como é estrutura de um banco de dados geográfico, conhecer suas vantagens e ver onde é usado. Além disso, vamos compreender o uso do XML em banco de dados, a sua estrutura, seus benefícios e como utilizá-lo de maneira adequada. Por último, vamos aprender sobre banco de dados não-relacionais, por que existem e quais suas diferenças em relação ao banco de dados relacional.

Um jovem formado em computação pretende abrir seu próprio negócio. Sua ideia, a princípio, é criar um sistema de localização em que o usuário pesquisa o que deseja localizar perto dele e o sistema informa a localização mais próxima do que ele pesquisou, por exemplo, a localização da padaria mais próxima de onde estou ou o supermercado mais próximo, facilitando assim a vida do usuário, que poderá se locomover para os lugares com maior facilidade. Porém, ele não entende nada de estrutura de banco de dados ou de sistemas, e você foi contratado para auxiliá-lo a desenvolver sua ideia.

Com seu esforço nos estudos e nas aulas, você poderá ajudá-lo a desenvolver sua ideia e, ao mesmo tempo, aprender como gerenciar um banco de dados não convencional, dando um passo a mais em sua carreira. Na seção 4.1, você vai aprender o conceito do banco de dados geográficos, o que são e qual sua importância para as empresas, como podem ser

administrados e qual sua estrutura. Na seção 4.2, vai aprender como criar arquivo de XML em um banco de dados, entender suas vantagens, e como utilizá-lo de maneira adequada. Na seção 4.3, você conhecerá como é um banco de dados não-relacional, ou NoSQL, o que é como é sua estrutura, além de algumas funções de utilização e suas principais operações.

Em resumo, com o seu empenho nos estudos e sua participação, esta unidade vai ser bastante produtiva e vai abrir muitas portas no quesito profissional. Pronto para mais este desafio? Então, mãos à obra!

Seção 4.1

Introdução a bancos de dados geográficos

Diálogo aberto

Caro aluno, a partir dos estudos desta seção, vamos entender sobre fundamentos de banco de dados não-convencionais. Depois de aprendermos a administrar um banco de dados, manter sua segurança e recuperar informações, agora vamos aprender novos conceitos de dados não-convencionais que são úteis na vida de um DBA.

Nesta primeira seção, falaremos sobre banco de dados geográficos, que é, onde usar e como trabalhar com ele.

Pode parecer que não, mas os bancos de dados geográficos são bastante utilizados no dia a dia, por exemplo, por empresas de fast-food que realizam entregas por aplicativo ou por aplicativos que ajudam o usuário a conseguir uma carona ou um meio de transporte mais barato. Esses aplicativos utilizam a geolocalização para informar o trajeto do serviço contratado até você ou o seu trajeto até seu destino.

Com nossos esforços, vamos poder contextualizar o uso de banco de dados geográficos e sua utilidade para o um jovem formado em computação que pretende abrir seu negócio. Sua ideia, a princípio, é criar um sistema de localização em que o usuário pesquisa o que deseja localizar perto dele e o sistema informa a localização mais próxima, por exemplo, a localização da padaria mais próxima de onde ele está.

Sua função é auxiliar o jovem formado a criar um relatório sobre como funciona um banco de dados geográfico, apresentando, por meio de um sistema de informação geográfico (GIS), como seria o seu funcionamento e como poderia ajudá-lo.

Para auxiliá-lo em mais um desafio, vamos conhecer um banco de dados geográfico, suas vantagens e sua estrutura. Vamos entender também o que é o GIS e o Geoprocessamento. Boa sorte e bom estudo!

Não pode faltar

Os bancos de dados geográficos (também conhecidos como bancos de dados espaciais) podem suportar feições geométricas, permitindo análise espacial, ou seja, é possível calcular áreas e distâncias (ELMASRI, 2005).



Assimile

Usamos o termo dados espaciais em um sentido amplo, abrangendo pontos multidimensionais, linhas, retângulos, polígonos, cubos e outros objetos geométricos. Um objeto de dados espaciais ocupa determinada região do espaço, chamada de extensão espacial, que é caracterizada por sua localização e por seu limite (ELMASRI, 2005).

Do ponto de vista de um SGBD, podemos classificar os dados espaciais como dados de ponto ou dados de região, tais como terrenos, ou localização em mapas (ELMASRI, 2005). A saber:

- **Dados de ponto:** um ponto tem uma extensão espacial caracterizada completamente por sua localização. Intuitivamente, ele não ocupa nenhum espaço e não tem nenhuma área nem volume associados. Os dados de ponto consistem em uma coleção de pontos em um espaço multidimensional. Os dados de ponto armazenados em um banco de dados podem ser baseados em medidas diretas ou gerados pela transformação de dados obtidos por meio de medidas para facilidade de armazenamento e consulta. Os dados de varredura são um exemplo de dados de ponto medidos diretamente, e incluem mapas de bits ou mapas de pixels, como as imagens de satélite. Cada pixel armazena um valor medido (por exemplo, temperatura ou cor) para uma localização correspondente no espaço. Outro exemplo são as imagens médicas, como as varreduras do cérebro, feitas por ressonância magnética tridimensional. Os vetores de características extraídos de imagens, textos ou sinais, como as séries temporais, são exemplos de dados de pontos obtidos pela transformação de um objeto de dados. Conforme veremos, frequentemente, é mais fácil usar tal

representação dos dados em vez da imagem ou sinal real para responder às consultas (ELMASRI, 2005).

- **Dados de região:** uma região tem uma extensão espacial com uma localização e um limite. A localização pode ser considerada como a posição de um “ponto de âncora” fixo para a região, tal como seu centroide. Em duas dimensões, o limite pode ser visualizado como uma linha (para regiões finitas, um loop fechado) e em três dimensões ele é uma superfície. Os dados de região consistem em uma coleção de regiões. Os dados de região armazenados em um banco de dados normalmente são uma simples aproximação geométrica de um objeto de dados real. “Dados vetoriais” é o termo usado para descrever tais aproximações geométricas, construídas usando pontos, segmentos de linha, polígonos, esferas, cubos etc. Muitos exemplos de dados de região surgem nas aplicações geográficas. Por exemplo, as estradas e rios podem ser representados como uma coleção de segmentos de linha, e os países, estados e lagos podem ser representados como polígonos. Outros exemplos surgem nas aplicações de projeto auxiliado por computador. Por exemplo, a asa de um avião pode ser modelada como um modelo de arame, usando uma coleção de polígonos intuitivamente dispostos de maneira a construir a superfície do modelo de arame, aproximando-se do formato da asa, e um objeto tubular pode ser modelado como a diferença entre dois cilindros concêntricos (ELMASRI, 2005).



Pesquise mais

O QGIS é um sistema de informação geográfica livre, com que é possível visualizar e publicar informações geoespaciais usando informações de bancos de dados, como Postgres e MySQL. No caso do Postgres, é preciso de uma extensão chamada PostGIS instalada na máquina.

Para conhecer um pouco mais sobre a ferramenta QGIS acesse o site oficial: <https://www.qgis.org/pt_BR/site/> (acesso em: 6 set. 2018).

Os GIS tratam extensivamente dados espaciais, incluindo pontos, linhas e regiões bi ou tridimensionais. Por exemplo, um mapa

contém localizações de pequenos objetos (pontos), rios e estradas (linhas), cidades e lagos (regiões), conforme mostra a Figura 4.1. Um sistema GIS precisa gerenciar eficientemente conjuntos de dados bidimensionais e tridimensionais. Todas as classes de consultas espaciais que descrevemos surgem naturalmente, e tanto dados de ponto como dados de região precisam ser manipulados. Os sistemas GIS comerciais, como o ArcInfo, estão em pleno uso atualmente, e os sistemas de banco de dados de objetos tem como objetivo suportar também as aplicações de GIS (NISHIMURA, 2009).

Figura 4.1 | Mapa com distritos de São Paulo com suas fronteiras



Fonte: Queiroz e Ferreira (2018, p.12).

De acordo com a Figura 4.1, uma consulta de localizações do distrito de São Paulo, ficaria da seguinte maneira:

```
SELECT bairro, AsText(spatial_data) geom  
  
FROM distritoss  
  
WHERE denominação = 'GRAJAU'
```



Refleta

Um banco de dados geográfico tem sua própria estrutura e tabelas específicas para o armazenamento de geolocalizações. É possível instalar um banco de dados relacional comum e usar informações geográficas?

Os bancos de dados multimídia, que contém objetos multimídia, como imagens, texto e vários tipos de dados de séries temporal (por exemplo, áudio), também exigem gerenciamento de dados espaciais. Em particular, encontrar objetos semelhantes a determinado objeto é uma consulta comum em um sistema multimídia, e uma estratégia popular para responder as consultas por similaridade envolve primeiro mapear os dados multimídia em uma coleção de pontos, chamados de vetores de características. Então, uma consulta por similaridade é convertida no problema de encontrar os vizinhos mais próximos do ponto que representa o objeto da consulta (NISHIMURA, 2009).

No PostgreSQL, esses tipos de informações salvas são classificados como TinyBlob, MediumBlob e LongBlob.

- O Tiny Blob é um campo de armazenamento de até 255 caracteres, que equivale a 8 bits.
- O Blob é o mais comum entre os bancos de dados, com um armazenamento de um pouco mais 16535, que equivale a 16 bits.
- O longBlob tem capacidade de armazenamento de até 4294967295 caracteres, que equivale a 32 bits.



Exemplificando

Ao usarmos comandos SQL para consultas de banco de dados, precisamos sempre converter as informações de dados geográficos para nossa realidade, para que possamos interpretá-las.

No exemplo abaixo, é preciso converter o campo `spatial_data` em texto para podermos trabalhar com ele em outras tabelas.

```
SELECT bairro, AsText(spatial_data) geom  
  
FROM distritoss
```

É possível dividir os GIS em três categorias:

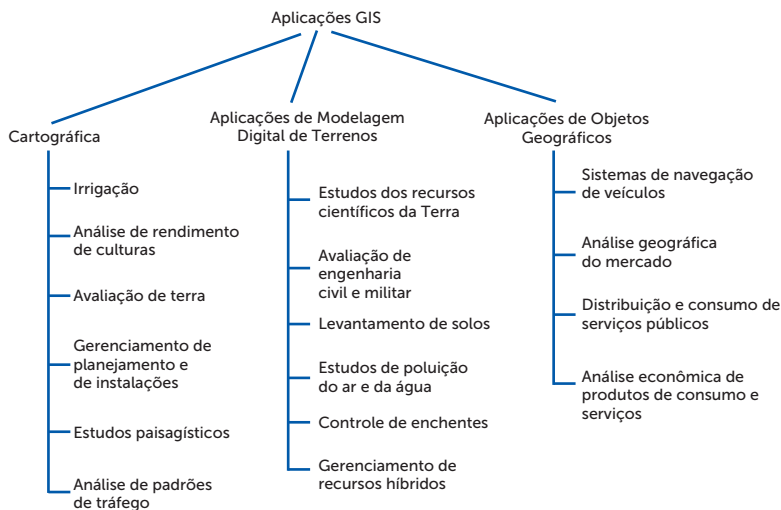
- 1- Aplicações cartográficas;

2- Aplicações de modelagem digital de terrenos;

3- Aplicações de objetos geográficos;

A Figura 4.2 resume bem essas categorias:

Figura 4.2 | Categoria GIS



Fonte: Elmasri (2005, p.669).

Os vetores de características que representam objetos multimídia normalmente são pontos em um espaço de dimensão alta. Por exemplo, podemos obter vetores de características a partir de um objeto de texto usando uma lista de palavras-chave (ou conceitos) e observar quais palavras-chave estão presentes; assim, obtemos um vetor de valores 1 (a palavra-chave correspondente está presente) e valores 0 (a palavra-chave correspondente está ausente no objeto texto), cujo comprimento é igual ao número de palavras-chave presentes em nossa lista. Listas de várias centenas de palavras são comumente usadas. Podemos obter vetores de características a partir de uma imagem, examinando sua distribuição de cores (os níveis de vermelho, verde e azul de cada pixel) ou usando os primeiros coeficientes de uma função matemática (por exemplo, a transformação de Hough) que mais se aproximam das formas da imagem. Em geral, dado um sinal arbitrário, podemos representá-lo usando uma função matemática que tenha uma série padrão de

termos e aproximá-lo, armazenando os coeficientes dos termos mais significativos (ELMASRI, 2005).

Ao se fazer o mapeamento de dados multimídia em uma coleção de pontos, é importante garantir que haja uma medida da distância entre dois pontos que capture a noção de semelhança entre os objetos multimídia correspondentes. Assim, duas imagens mapeadas em dois pontos próximos devem ser mais semelhantes do que duas imagens mapeadas em dois pontos distantes entre si. Uma vez que os objetos sejam mapeados em um espaço de coordenadas conveniente, a atividade de encontrar imagens semelhantes, documentos semelhantes ou séries temporais semelhantes pode ser modelada como a atividade de encontrar pontos que estão próximos entre si: mapeamos o objeto da consulta em um ponto e procuramos seus vizinhos mais próximos. O tipo mais comum de dados espaciais em aplicações multimídia são os dados de ponto, e a consulta mais comum é a por vizinho mais próximo. Em contraste com os sistemas GIS e CAD/CAM, os dados são de dimensão alta (normalmente 10 ou mais dimensões) (ELMASRI, 2005).

Vejam os um exemplo:

```
INSERT INTO cidade (denominacao, sigla,
spatial_data)
VALUES ('AV. PAULISTA', 'SP', GeometryFromText('POLY
GON((335589.530575 7356020.721956, 335773.784959
7355873.470174))', -1));

SELECT d2.denominacao
FROM cidade c1,
cidade c2
WHERE touches(c1.spatial_data, c2.spatial_data)
AND c2.denominacao <> 'AV. PAULISTA'
AND c1.denominacao = 'AV. PAULISTA'
```

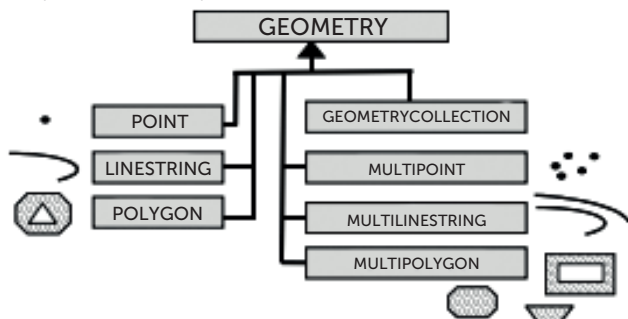
No exemplo, foi incluído na tabela cidade um polígono mostrando os pontos em volta da Av. Paulista, circulando sua

região. Com ela, podemos realizar o SQL, para encontrar regiões ou ruas próximas à Avenida Paulista. Em modelagem cartográfica e de terreno, capturam-se as variações em atributos espaciais, tais como características do solo, densidade de cultura e qualidade do ar. Em objetos geográficos, os objetos de interesse são identificados a partir de um domínio físico, como usinas elétricas, distritos eleitorais, lotes de propriedades, regiões de distribuição de produto e pontos de referência em cidades. Esses objetos estão relacionados a dados de aplicação relevantes, que podem ser, para este exemplo específico, consumo de energia, padrões de voto, volumes de vendas em lotes, volume de vendas de produtos e densidade de tráfego (ELMASRI, 2005).

As primeiras duas categorias de aplicações GIS requerem uma representação baseada em campos, enquanto a terceira categoria requer uma baseada em objetos. A abordagem cartográfica envolve funções especiais, que podem incluir a sobreposição de chamadas de mapas para combinar dados de atributos que permitirão, por exemplo, a medição de distância no espaço tridimensional e a reclassificação dos campos dados no mapa. A modelagem digital de terrenos requer uma representação digital de partes da superfície terrestre utilizando as elevações da Terra em pontos de amostra que são conectados para produzir um modelo de superfície como uma rede tridimensional (linhas em 3-D conectadas), mostrando a superfície do terreno. Isso requer funções de interpolação entre os pontos observados, bem como a visualização. Em aplicações geográficas baseadas em objeto, são necessárias funções espaciais adicionais para lidar com dados relacionados a estradas, dutos físicos, cabos de comunicação, linhas de transmissão de energia, entre outros. Por exemplo, para uma determinada região, mapas comparativos podem ser usados para realizar comparação entre vários momentos, a fim de mostrar mudanças em certos dados, tais como localizações de estradas, cabos, prédios e fluxos (ELMASRI, 2005).

Um dos pontos fortes do PostgreSQL é seu potencial de extensibilidade, que ajudou no desenvolvimento de uma extensão geográfica completa, o PostGIS, por uma empresa canadense que mantém a equipe de desenvolvimento. Os tipos de dados espaciais fornecidos por esta extensão podem ser vistos na Figura 4.3.

Figura 4.3 | Tipos de dados espaciais do PostGIS



Fonte: Gilberto e Karine (2018, p.75).

Vejamos os tipos de dados espaciais do PostGIS da Figura 4.3:

- O **Point** é herdado simples como um objeto geométrico.
- Os **Multipoints** são simples se nenhuma de duas coordenadas (**Points**) forem iguais (ou tiverem o valor de coordenadas idênticas).
- O **Linestring** é simples se não passar pelo mesmo **Point** duas vezes (exceto para ponto finais, que em cada caso são referidos como um anel linear e considerados fechados).
- O **Multilinestring** é como se todos seus elementos fossem uma simples e única interseção entre qualquer um dos dois elementos que ocorrem em **Points** que estejam nos limites dos dois elementos.
- O **Polygon** é válido se nenhum dos dois anéis (feitos de um exterior e um interior) cruzar no limite. O limite de um **Polygon** pode intersectar em um **Point**, mas só como uma tangente (ex.: não em uma linha). Um **Polygon** pode não ter linhas cortadas ou extremidades, e os anéis interiores devem estar inteiramente contidos dentro dos anéis exteriores.
- O **Multipolygon** é válido somente se todos seus elementos forem válidos e os interiores de nenhum dos dois elementos intersectarem. Os limites de quaisquer dois elementos podem se tocar, mas somente em um número finito de **Points**.

A criação de uma tabela com esses tipos de dados deve ser realizada em duas etapas, como mostrado no exemplo abaixo, em que é criada uma tabela para armazenar os distritos da cidade de São Paulo:

```
Create Table distritosp
```

```
( cod SERIAL,  
  
  sigla VARCHAR(10) ,  
  
  denominação VARCHAR(50) ,  
  
  PRIMARY KEY (COD) ) ;
```

```
SELECT                                AddGeometryColumn( 'terralibdb' ,  
'distritosp', 'spatial_data', -1, 'POLYGON', 2) ;
```

Para a instrução SQL acima, usamos a função `AddGeometryColumn` para adicionar a coluna com o tipo espacial. Com esta função do PostGIS especificada no OpenGIS, é realizado todo o preenchimento da tabela de metadado "geometry_columns". Os parâmetros dessa função são:

- Nome do banco de dados conectado;
- Nome da tabela que irá conter a coluna espacial;
- Nome da coluna espacial;
- Sistema de coordenadas em que se encontram as geometrias da tabela;
- Tipo da coluna espacial, que serve para criar uma restrição que verifica o tipo do objeto sendo inserido na tabela;
- Dimensão em que se encontram as coordenadas dos dados.

Após criar as tabelas de dados, podemos inserir as informações usando o comando SQL `INSERT`. Para isso, podemos usar a representação textual das geometrias em conjunto com a função `GeometryFromText`, que recebe a representação textual e o sistema de coordenadas em que se encontra a geometria:

```
INSERT INTO distritosp(denominacao, sigla, spatial_
data)

VALUES ('GRAJAU', 'MAR', GeometryFromText('POLYG
ON((335589.530575 7356020.721956, 335773.784959
7355873.470174))', -1));
```

Abaixo, mostramos um exemplo de consulta espacial envolvendo o operador *touches* para descobrir os distritos vizinhos a Grajau:

```
SELECT d2.denominacao

FROM distritosp d1,

      Distritosp d2

WHERE touches(d1.spatial_data, d2.spatial_data)

AND d2.denominacao <> 'GRAJAU'

AND d1.denominacao = 'GRAJAU'
```

Esse é um exemplo de junção espacial entre duas relações (no nosso caso, a mesma relação foi empregada duas vezes). Com base no que aprendeu até agora, você com certeza tem conceitos claros em relação ao banco de dados geométricos e a como usá-lo no cotidiano profissional. Mas isso não é tudo, então continue estudando, leia bastante e busque conhecimentos em várias fontes. Certamente, logo você se destacará no mercado profissional.

Sem medo de errar

Chegou o momento de ajudar o jovem formado em computação a criar um sistema de localização em que o usuário pesquise o que deseja localizar perto dele e o sistema informe a localização mais próxima. Primeiramente, é preciso pensar na estrutura inicial, como seria o sistema, qual sua linguagem e qual seu banco de dados.

Como o sistema irá trabalhar com localização, nada melhor do que um banco de dados geográficos.

Com base no que você aprendeu, é recomendado indicar o PostgreSQL por ter uma extensão geográfica completa, o

PostGIS. Com a criação de tabela, veja o exemplo da Av. Paulista, em São Paulo.

Create Table cidade

```
( cod SERIAL,  
  
  sigla VARCHAR(10) ,  
  
  denominacao VARCHAR(50) ,  
  
  PRIMARY KEY (COD)  
  
);
```

Depois de criada a tabela, é preciso adicionar a coluna com o tipo espacial, com o seguinte comando:

```
SELECT      AddGeometryColumn('bancodb' ,    'cidadesp',  
'spatial_data', -1, 'POLYGON', 2);
```

Com esta função do AddGeometryColumn, você incluiu a coluna com o tipo espacial.

Agora, para inserir informações nesta tabela com o tipo espacial, seria assim:

```
INSERT INTO cidade (denominacao, sigla, spatial_  
data)  
  
VALUES ('AV. PAULISTA' , ' SP' ,GeometryFromText('POLY  
GON((335589.530575  7356020.721956,  335773.784959  
7355873.470174))' , -1));
```

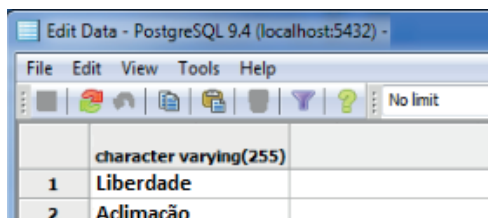
Assim podemos recuperar as informações usando o SQL e identificar os bairros perto da Av. Paulista.

```
SELECT d2.denominacao  
  
FROM cidade c1,  
  
      cidade c2  
  
WHERE touches(c1.spatial_data, c2.spatial_data)
```

```
AND c2.denominacao <> 'AV. PAULISTA'
```

```
AND c1.denominacao = 'AV. PAULISTA'
```

Figura 4.4 | Resultado da Consulta PostGIS



The screenshot shows the 'Edit Data' window for PostgreSQL 9.4 (localhost:5432). The window has a menu bar (File, Edit, View, Tools, Help) and a toolbar with icons for various database operations. Below the toolbar is a table with the following data:

	character varying(255)
1	Liberdade
2	Aclimação

Fonte: captura da tela do PostgreSQL, elaborada pelo Autor.

Avançando na prática

Localizar uma padaria próxima

Descrição da situação-problema

João acabou de se mudar para o interior de São Paulo e precisa localizar os principais comércios próximos a sua residência, como a padaria mais próxima para tomar o café da manhã, mas não encontra ninguém na rua para perguntar. Usando a estrutura de um banco de dados PostgreSQL, desenvolva uma solução para essa consulta.

Resolução da situação-problema

Para o aplicativo conseguir realizar a busca, primeiramente ele precisa saber real localização do usuário. Vamos supor que ele esteja na rua 13 de maio. A estrutura para inserir no banco de dados antes de realizar o cálculo seria:

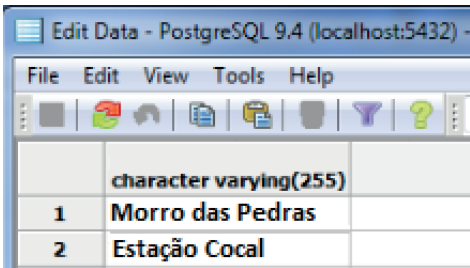
```
INSERT INTO cidade (denominacao, sigla, spatial_  
data)
```

```
VALUES('13 de maio', 'SP', GeometryFromText('POLY  
GON((335589.530575 7356020.721956, 335773.784959  
7355873.470174))', -1));
```

Depois de executar o comando para inserir a localização atual no banco de dados, será executado um comando SQL para mostrar os pontos de padaria mais próximos ao usuário. Veja tal comando abaixo:

```
SELECT d2.denominacao
FROM cidade c1,
      cidade c2
WHERE touches(c1.spatial_data, c2.spatial_data)
AND c2.denominacao <> '13 de maio'
AND c1.denominacao = '13 de maio'
```

Figura 4.5 | Resultado da Consulta PostGIS



Fonte: captura da tela do PostgreSQL, elaborada pelo autor.

Faça valer a pena

1. É um sistema com capacidade de aquisição, armazenamento, processamento, análise e exibição de informações digitais georreferenciadas, topologicamente estruturadas, associadas ou não a um banco de dados alfanumérico. Trata extensivamente os dados espaciais, incluindo pontos, linhas e regiões bi ou tridimensionais.

Qual é o nome deste sistema?

- a) Sistema CAM
- b) Sistema CAD
- c) Sistema AM/FM
- d) Sistema AM
- e) Sistema GIS

2. Os erros cometidos frequentemente no processo de desenvolvimento de mapas em um SIG podem estar relacionados a diferentes etapas de processamento de dados do sistema. Com base nas fontes de erro listadas a seguir, assinale as opções corretas.

Todas as classes de consultas espaciais surgem naturalmente e tanto dados de ponto como dados de região precisam ser manipulados.

Quanto às afirmações abaixo, quais estão associadas a erros dos dados originais?

- I. Projeção da superfície tridimensional da Terra sobre um mapa bidimensional.
- II. Conversão de dados analógicos em digitais.
- III. Técnicas de classificação não-supervisionada de imagens.
- IV. Cruzamento de dados digitais.
- V. Escala do mapa.
- VI. Densidade de observações.
- VII. Idade dos dados.

- a) I e V estão corretas
- b) I e VI estão corretas
- c) III está correta
- d) II está correta
- e) VII está correta

3. Um dos pontos fortes do PostgreSQL é seu potencial de extensibilidade, o que ajudou no desenvolvimento de uma extensão geográfica completa, o PostGIS, por uma empresa canadense, que mantém a equipe de desenvolvimento.

Sabemos qual função do `AddGeometryColumn` você incluiu a coluna com o tipo espacial.

O que seria a função *touches*, usada nas condições de consultas?

- a) O *touches* é responsável por validar se as geometrias. Uma junção espacial entre duas relações.
- b) O *touches* é uma condição no GIS.
- c) O *touches* é uma cláusula no SQL para validar a string.
- d) O *touches* serve para realizar a busca por índice no banco de dados.
- e) O *touches* é usado para inserir informações.

Seção 4.2

A XML e seu armazenamento em bancos de dados

Diálogo aberto

Caro aluno, depois de aprender o conceito de banco de dados geográficos e como são úteis no nosso dia a dia, você aprenderá o conceito XML, onde é usado e quais seus benefícios. Esta seção se concentrará na criação do modelo de dados XML, em suas linguagens associadas e em como os dados extraídos dos bancos de dados relacionais podem ser formados como documentos XML para serem trocados pela Web.

Hoje em dia, diversos tipos de aplicações, como editores, navegadores e aplicativos de e-mail, usam o XML para marcações de conteúdos, introduzindo novas possibilidades e uma melhor integração entre dados e usuários. Com uma estrutura do banco de dados criada, algumas localizações cadastradas e clientes inseridos no banco de dados, o Sr. Paulo pretende avançar para a próxima etapa da sua ideia.

Depois de ter criado uma estrutura do banco de dados, ele precisa inserir informações de futuros clientes que serão cadastrados em seu banco de dados para que possam aparecer na localização de seu sistema. Porém, estas informações chegam a ele por meio de um sistema terceiro, e a única forma de copiar estas informações se dá por meio de um arquivo .xml gerado por esse sistema. É preciso auxiliá-lo, neste caso, em como ele pode importar essas informações para seu banco de dados, de maneira que as informações exportadas fiquem em suas respectivas tabelas. Apresente para ele, por meio de um relatório, um exemplo de como é feita uma estrutura do XML e como pode ser usada junto com o banco de dados.

Com seu esforço nos estudos e nas aulas, você será capaz de realizar criação de estruturas de XML em backup de banco de dados, além de saber como interpretar a sua estrutura.

Mãos à obra!

Não pode faltar

O que é um XML?

XML significa *Extensible Markup Language*. É uma linguagem usada para demonstrar dados como uma *string*, que inclui uma “marcação” intercalada para descrever as propriedades de seus dados. A marcação permite que o texto seja intercalado por dados que são relacionadas a seu conteúdo ou forma (ELMASRI, 2011).

A marcação ocorre predominantemente na forma de *tags*, que se distinguem dos dados formados por caracteres por enfeixarem esses com os colchetes angulares “<” e “>” como “<exemplo>”. Portanto, como uma *string*, um documento é composto de *tags* e dados formatados por caracteres que são combinados para formar elementos. O elemento sempre começa com uma *tag* de abertura e termina com uma de fechamento. A *tag* de fechamento possui os colchetes angulares e uma barra “/”, que a distingue de uma *tag* de abertura, por exemplo “</exemplo>”.

As *tags* têm forma de “<texto-da-tag>palavra</ texto-da-tag>” no texto que está sendo marcado, e a região entre a tag de abertura e a de fechamento é um elemento, assim, a frase apresentada como exemplo, acima, é um elemento conforme apresentado no Quadro 4.1 (ELMASRI, 2011).

Quadro 4.1 | Exemplo de estrutura de XML

```
<texto-da-tag>
  palavra
</texto-da-tag>
```

Fonte: elaborado pelo autor.

Especificamente, a marcação disponibiliza um mecanismo para adicionar conteúdo e dados sobre a estrutura de documentos. Essas *tags* proporcionam um meio de inserir os dados formados por caracteres em um elemento, e eles podem ter outros elementos aninhados em seu interior, chamados de **subelementos**. Um documento é composto de somente um elemento externo (de nível superior), que contém outros elementos e/ou dados de

caracteres, podendo cada subelemento conter outros subelementos intercalados por dados de caracteres, e assim por diante. Veja uma demonstração simples de um documento XML no exemplo abaixo (Quadro 4.2). Este documento é composto da instrução de processamento "<?xml version='1.0'>" seguida do elemento "<historia>...</historia>". O elemento "historia" possui um subelemento "para".

Quadro 4.2 | Exemplo de uma estrutura simples de XML

```
<?xml version="1.0">
<historia>
  <para>Ronaldo sempre demonstrou habilidades com
a bola desde criança. Seus pais sempre o apoiavam
e incentivavam em seus treinos, acompanhando de
perto o filho. </para>
  <para>Quando fez 18 anos Ronaldo realizou seu
sonho e foi jogar futebol profissional na Europa,
mostrando habilidades que nenhum jogador possuía.
</para>
</historia>
```

Fonte: elaborado pelo autor.



Reflita

O que acontece se um XML tem duas *tags* com o mesmo nome? O XML não será interpretado?

Os elementos do XML aninhados também podem ser usados para descrever parte de outro elemento. Sua apresentação em XML é apresentada abaixo, no Quadro 4.3, com um exemplo de um XML usado para locadores, em que a *tag* principal é a locadora e nela há duas tags <filme>, diferenciadas pelo elemento "gênero" dentro delas.

Quadro 4.3 | Exemplo de uma estrutura de XML aninhados

```
<?xml version="1.0">
<locadora tipo="filmes">
  <filme genero="acao">
```

```

        <nome classificacao="livre">Vingadores</
nome>
        <nome classificacao="16anos">Velozes e
furiosos</nome>
        <nome classificacao="18anos">Logan</nome>
    </filme>
    <filme genero="comedia">
        <nome classificacao="livre">O Mascara</nome>
        <nome classificacao="livre">Debie e Loide</
nome>
        <nome classificacao="livre">12 de mais</
nome>
    </filme>
    <filme genero="terror">
        <nome classificacao="18anos">Premonição</
nome>
        <nome classificacao="18anos">Atividade
paranormal</nome>
        <nome classificacao="18anos">Hora do
Pesadelo</nome>
    </filme>
</locadora>

```

Fonte: elaborado pelo autor.

Um conjunto de elementos com as mesmas propriedades é chamado de **tipo de elemento**. Os tipos de elementos são diferenciados pela atribuição de um nome exclusivo à *tag* desses elementos. O tipo da XML é usado da mesma maneira que as linguagens de programação, em que (1, 2, 3) são membros de um tipo de dado chamado de "inteiro" (ELMASRI, 2011).

Algumas linguagens de programação possuem tipos de dados extensíveis, como as orientadas a objetos, por exemplo, o Java. Se você criar uma classe de objeto chamada "cor", ela poderá ter instâncias com nomes "vermelho", "azul", "amarelo", "verde".

No XML, os tipos de elementos são extensíveis (Quadro 4.4), portanto é possível criar um novo tipo de elemento chamado “cor”, “palavra-chave” ou “textoexemplo”. Esses tipos de elementos possuem atributos, que são representados como parte da *tag*, por exemplo `<cor rgb="110">` ou `<linguagem historia="Portugues" palavra-chave="cachorro">`.

Um atributo é composto de um par de nome e valor.



Pesquise mais

Veja no link abaixo a importância do XML no desenvolvimento Android.

ANDROIDPRO. A importância do XML. 2 ago. 2016. Disponível em <https://www.youtube.com/watch?v=zTBhLLNaEyM>. Acesso em: 11 set. 2018

Quadro 4.4 | Exemplo de uma estrutura de XML com tipos de elementos

```
<?xml version="1.0">
<tinta>
  <cor rgb="110" />
  <cor rgb="210" />
</tinta>
```

Fonte: elaborado pelo autor.

O XML foi originalmente projetado para vencer os desafios da editoração eletrônica em larga escala, isolando o conteúdo da formatação. Separar o conteúdo de um documento de como ele deve ser formatado simplifica o desenvolvimento e a manutenção. Equipes diferentes com experiências distintas podem trabalhar independentemente nas informações capturadas em documentos e no formato, estilo e estética (FIGUEIREDO; HIGASI; MURAOKA JUNIOR, 2003).



Assimile

O XML foi projetado para ser distribuído pela Web. Os dados são representados com o uso da estrutura de um documento XML.

Os elementos e atributos fornecem metadados úteis e estruturam os dados de uma maneira que podem ser transferidos pela Web, manipulados por aplicativos diferentes e alterados para atender a necessidades diversas.

O XML também tem se mostrado útil na transferência de dados. Ele pode ser usado para transferir dados na Web, entre aplicativos ou entre outros usuários. O XML sozinho não tem a finalidade de apresentação para o usuário final, mas quando empregados junto com as folhas de estilo (que manipulam a apresentação), os documentos XML podem ser facilmente visualizados. Os documentos XML fornecem uma maneira de capturar os dados e as folhas de estilo, como *eXtensible Stylesheet Language* (XSL) ou *Cascading Style Sheets* (CSS), disponibilizam mecanismos de conversão da XML em HTML (FIGUEIREDO; HIGASI; MURAOKA JUNIOR, 2003).

O XML foi projetado para comunicar conteúdo em uma representação flexível e extensível, portanto, as descrições dos dados podem ser repetidamente revisadas quando houver alterações na área abordada. Isso torna o XML particularmente útil para áreas cujas informações tenham uma organização complexa e passem por revisões frequentes. Entre algumas das primeiras a adotar a tecnologia XML estão as áreas científicas, como a biologia, a química e a matemática. Nessas áreas, existem padrões científicos com base em XML, como o BioML, o CML, e o MathML, respectivamente. Portanto, áreas bastante complexas podem ser representadas consideravelmente bem com XML (ELMASRI, 2011).

As duas principais vantagens do XML sobre outras linguagens de transferência de dados é que ele é muito expressivo e extensível. Na verdade, você pode dizer o que quiser sobre o tópico que escolher. Ele também possui uma sintaxe consistente, que o torna fácil de analisar. Portanto:

- O XML pode capturar o tipo de informação que é transferida entre aplicativos.
- Os documentos XML podem ser personalizados para se ajustarem a necessidades muito específicas.

O XML também pode ser usado para representar o conteúdo de páginas Web com folhas de estilo empregadas para definir o processamento. As páginas Web podem ser estáticas ou dinâmicas, como os catálogos online. Mesmo em páginas Web estáticas, a separação entre conteúdo e apresentação pode simplificar o desenvolvimento e reduzir seu tamanho. As páginas Web dinâmicas são muito mais fáceis de tratar, por que o conteúdo pode ser alterado quando necessário enquanto as folhas de estilo permanecem estáticas. É possível ter apresentações muito diferentes usando a mesma folha de estilo quando apropriadamente condicionada ao conteúdo. Por exemplo, os dados podem variar de uma tabela extensa a uma única instância, e a apresentação ser personalizada para atender a formatações diferentes. Os documentos XML podem ser publicados em vários formatos: HTML, PDF, post-script, entre outros.



Exemplificando

Uma empresa possui um sistema de cadastro de clientes usado para envios de e-mail marketing de promoções. A estrutura de envio desses e-mails é XML, com informações do banco de dados que são montadas via programação e enviadas, cliente a cliente, de acordo com o XML montado.

```
<?xml version="1.0">
<cliente>
  <nome>João</nome>
  <telefone>(11) 4879.1234</telefone>
  <email>joao@email.com</email>
</cliente>
<cliente>
  <nome>Marcos</nome>
  <telefone>> (21) 3214.1234</telefone>
  <email>marcos@email.com </email>
</cliente>
```

Quando a estrutura dos dados não for simples, o XML pode ser necessário para armazenar relacionamentos complexos. Muitos deles podem ser capturados em XML. Se forem muito complexos ou se detalhes não forem completamente compreendidos, pode ser necessário um esforço significativo para modelar esses

relacionamentos antes de se tomar a decisão sobre a melhor estrutura a ser usada na representação XML. Ele também é útil quando é preciso associar informações de banco de dados ao conteúdo de uma publicação. Você pode desenvolver representações separadas para os dados e as publicações e, em seguida, combiná-las em um documento XML.

Também existem razões comerciais para se usar o XML. Ele tem o suporte dos navegadores Web, que, por já estarem difundidos, reduzem o custo da distribuição. A XML, como uma recomendação do World Wide Web Consortium (W3C), possui um nível de garantia para o desenvolvimento de sistemas grandes por que as maiores empresas de software, como a Microsoft, a IBM, a Sun e a Oracle, dão suporte a essa tecnologia. Essas razões são ainda mais importantes para os bancos de dados, já que os sistemas grandes podem ser caros e uma parcela substancial da empresa pode depender deles.

O banco de dados XML é um conjunto de documentos XML que persistem e podem ser manipulados. Historicamente, os documentos foram desenvolvidos para a comunicação entre pessoas, mas com o advento dos computadores, eles também puderam ser usados para a comunicação entre computadores, entre uma pessoa e um computador ou entre um computador e uma pessoa (FIGUEIREDO; HIGASI; MURAOKA JUNIOR, 2003).

Os documentos XML tendem a ser orientados ao processamento de documentos ou de dados. Os documentos orientados ao processamento de documentos são aqueles em que o XML é usado por sua habilidade de capturar linguagens naturais (humanas), como as dos manuais de usuários, das páginas Web e de folhetos de propagandas. Eles são caracterizados por estruturas complexas ou irregulares e conteúdo misto, sendo sua estrutura física importante.

A diferença entre os documentos XML orientados ao processamento de dados e os orientados ao processamento de documentos é muito sutil, e alguns documentos, como as páginas Web dinâmicas com dados e textos descritivos, poderiam ser visualizados em qualquer das duas formas. No entanto, as operações desejadas nos documentos variarão. O controle do processo pode ser representado como XML ou definido independentemente.

O XML não possui um mecanismo predefinido para estabelecer os procedimentos, mas há várias linguagens para códigos procedimentais que foram desenvolvidas a partir dele, como o SOAP.

No entanto, na transferência de dados, pode ser importante rastrear ou registrar dados que estão sendo enviados, principalmente quando os aplicativos e/ou usuários estiverem em empresas diferentes. Os dados que forem transferidos podem precisar ser retidos em um arquivo e ocasionalmente pesquisados (ELMASRI, 2011). Os repositórios XML podem atender a outras necessidades de retenção, alteração e pesquisa em documentos XML, como ser usados por documentos XML interconectados a um servidor Web, reter conteúdos importantes em um sistema seguro ou capturar informações que sejam representadas mais claramente na estrutura dos documentos XML.

Uma das vantagens do XML é que sua estrutura é mais expressiva que os relacionamentos usados em bancos de dados relacionais. O relacionamento é um conjunto desordenado de registros, em que cada um possui um conjunto fixo de características (ou atributos). A estrutura de um elemento XML é mais expressiva que a do relacionamento por que ela pode conter outros elementos, variar a ordem e a quantidade de atributos e ter vários elementos do mesmo tipo.

Mas, como usar o XML no banco de dados? Eu tenho uma base de dados MySQL e preciso manusear informações dos dados e do XML ao mesmo tempo. No Quadro 4.5 vamos demonstrar como é realizada uma consulta de uma estrutura simples de XML dentro do banco de dados MySQL. Na consulta, vamos retornar valores que estão entre as *tags*, retornando os resultados logo em seguida (Quadro 4.5).

Quadro 4.5 | Exemplo de consulta de uma estrutura XML no banco de dados MySQL

```
<carro>
  <marca>Honda</marca >
  <modelo>Civic</modelo>
  <ano>2018 </ano>
  <combustivel>flex</combustivel>
```

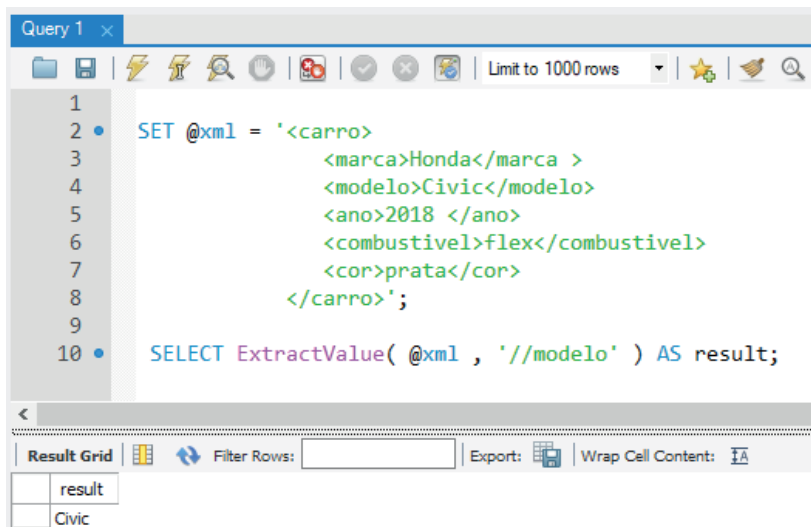


```
<cor>prata</cor>
</carro>
```

Fonte: elaborado pelo autor.

A partir do código do Quadro 4.5, vamos supor que você precise somente do “modelo” que vem do XML para carregar no banco de dados. Usaremos a estrutura da Figura 4.6 para montá-lo.

Figura 4.6 | Exemplo de consulta de uma estrutura XML no banco de dados MySQL



Fonte: captura de tela do MySQL Workbench 6.3, elaborada pelo autor.

Na Figura 4.6, o comando SET é usado para declarar uma variável, no caso o @xml, que irá receber a *string* do XML.

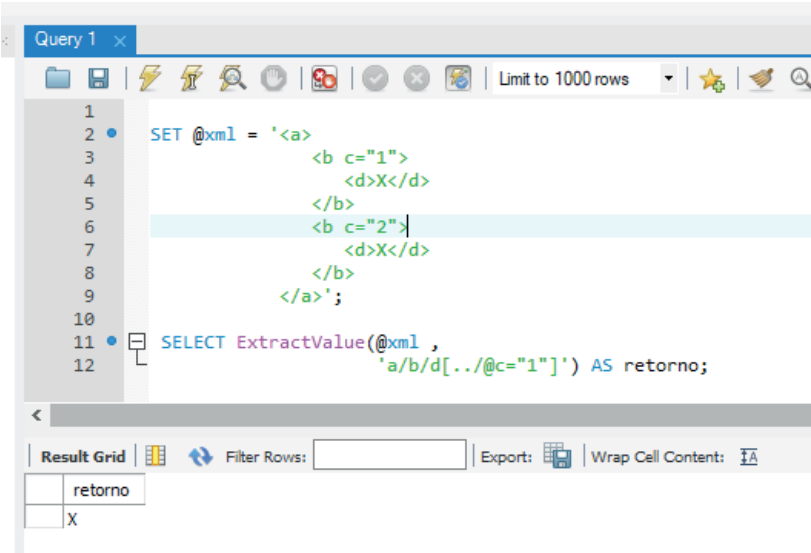
Com essa *string* salva na variável, podemos manipular vários modos, como salvar estrutura inteira em uma tabela, ou então escolher as *tags* a serem usadas no banco, ou até mesmo comparar esse XML com outro salvo no banco de dados.

Para realizar uma consulta no banco de dados usando a estrutura XML, precisamos usar “funções” que são nativas de cada banco de dados, no nosso caso, o MySQL tem a função ExtractValue, que serve para extrair os valores necessários e que estão dentro do XML. Dentro dessa função, os parâmetros dela são a *string* do XML e a *tag* na qual se irá pesquisar para retorno.

Como nesse exemplo vamos buscar o modelo somente do XML, então usaremos a *tag* modelo, antecedida por duas barras *"/*", assim, ele irá exibir o que há dentro daquela *tag*, sendo possível incluir após este resultado em uma tabela. Além de buscar somente pela *tag*, podemos supor que foi enviado um XML de que não sabemos sua estrutura, somente a *tag* principal `<carro/>`. Neste caso, em vez de buscarmos diretamente pela *tag*, podemos usar o comando **child::***, que retornará todas as *tags* dentro do XML, possibilitando filtrar quais *tags* realmente seriam necessárias para o banco de dados.

Na Figura 4.7 temos um exemplo de uma consulta no MySQL pelo tipo do elemento.

Figura 4.7 | Exemplo de consulta de uma estrutura XML no banco de dados MySQL



Fonte: captura de tela do MySQL Workbench 6.3, elaborada pelo autor.

A função **ExtractValue** serve para retornar o valor de determinada posição ou *tag* do XML. Com base no que aprendeu até agora, você tem conceitos claros em relação ao XML entendendo o funcionamento da sua estrutura, como criar e como vincular suas informações junto com um banco de dados relacional.

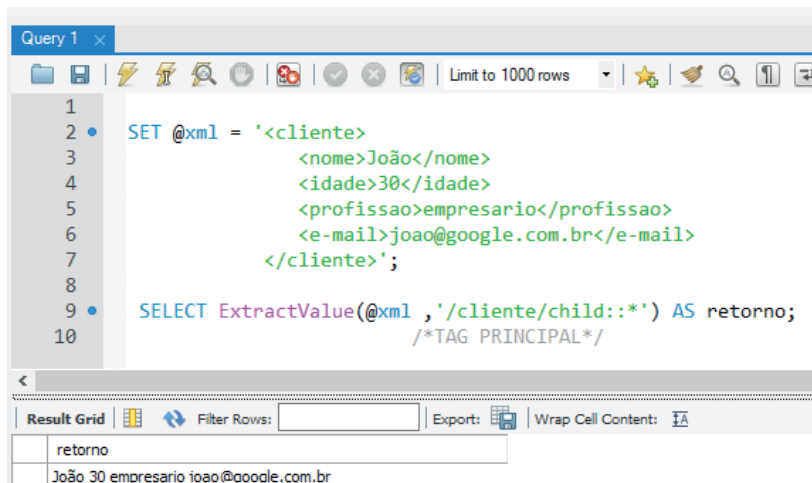
Sem medo de errar

Caro aluno, depois de termos criado uma estrutura do banco de dados, nosso desafio é apresentar para o Sr. Paulo, em um relatório, uma forma de interligar as informações vindas do cliente, por meio do XML, com o seu banco de dados. Vamos supor que o XML que foi entregue ao Sr. Paulo tenha esta estrutura:

```
<cliente>
  <nome>João</nome>
  <idade>30</idade>
  <profissao>empresario</profissao>
  <e-mail>jaoao@google.com.br</e-mail>
</cliente>
```

Com o MySQL, podemos observar que podemos nos preocupar somente com a *tag* principal, no caso `<cliente>`. Dentro da função que estudamos para consulta de XML (`ExtractValue`) existe um parâmetro chamado **child::***, em que podemos observar que é possível resgatar todos os subelementos de dentro da *tag* `<cliente>` e, conseqüentemente, manipular estas informações dentro do banco de dados. Na Figura 4.8 é apresentada a estrutura da consulta e como seria o resultado.

Figura 4.8 | Exemplo de consulta de uma estrutura XML no banco de dados MySQL



The screenshot shows the MySQL Workbench interface. The top toolbar includes icons for file operations, execution, and settings. The query editor displays the following SQL code:

```
1
2 • SET @xml = '<cliente>
3           <nome>João</nome>
4           <idade>30</idade>
5           <profissao>empresario</profissao>
6           <e-mail>jaoao@google.com.br</e-mail>
7           </cliente>';
8
9 • SELECT ExtractValue(@xml, '/cliente/child::*') AS retorno;
10      /*TAG PRINCIPAL*/
```

The bottom panel shows the 'Result Grid' with the following data:

retorno
João 30 empresario ioao@google.com.br

Fonte: captura de tela do MySQL Workbench 6.3, elaborada pelo autor.

Com essa consulta, o Sr. Paulo não precisa se preocupar com a estrutura do XML, somente com a manipulação dela no banco de dados. Com base no que aprendemos, você, aluno, será capaz de compreender e manipular informações de XML dentro de um banco de dados.

Avançando na prática

Desvendando problema do XML

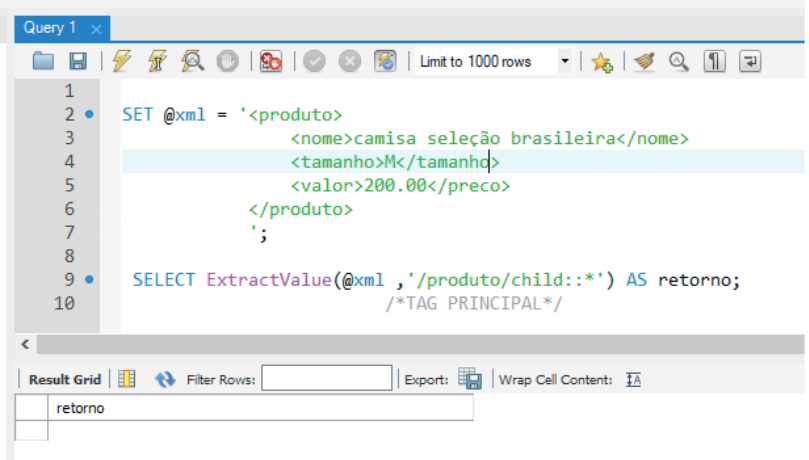
Descrição da situação-problema

Rogério, um jovem desenvolvedor, está com problemas para visualizar o XML no seu banco de dados. Ele já pediu diversas vezes o XML para o cliente e sempre ocorre o mesmo erro, e ele não consegue realizar a leitura do arquivo. A estrutura do XML seria:

```
<produto>
  <nome>camisa seleção brasileira</nome>
  <tamanho>M</ tamanho >
  <valor>200.00</preco>
</produto>
```

O resultado da consulta é sempre o mesmo, mostrado na Figura 4.9

Figura 4.9 | Exemplo de consulta de uma estrutura XML no banco de dados MySQL



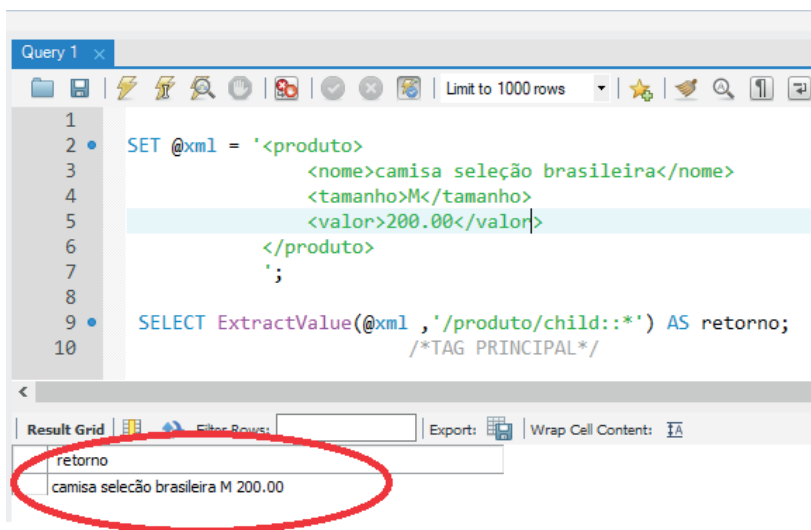
Fonte: captura de tela do MySQL Workbench 6.3, elaborada pelo autor.

Como podemos ajudar o Rogério a resolver este problema? É bom refletirmos sobre o que aprendemos até agora, compreendendo a estrutura e como são formadas as *tags* para resolver esta situação.

Resolução da situação-problema

Analisando o resultado da Figura 4.9, a consulta não retorna erro de construção do SQL, então o erro pode estar dentro do XML. Observe que a *tag* `<valor>` está sendo fechada com outro tipo de *tag*, no caso `</preco>`, o que não é permitido. Nesse caso, a leitura poderia sair totalmente do padrão estruturado do XML. Corrigindo essa *tag*, o resultado irá aparecer na tela corretamente, conforme a Figura 4.10.

Figura 4.10 | Exemplo de consulta de uma estrutura XML no banco de dados MySQL



Fonte: captura de tela do MySQL Workbench 6.3, elaborada pelo autor.

Como resultado, será apresentado o retorno da estrutura na tela com as informações contidas nas *tags*.

Faça valer a pena

1. É uma linguagem usada para demonstrar dados como uma string que inclui uma “marcação” intercalada, a fim de descrever as propriedades de

seus dados. A marcação permite que o texto seja intercalado por dados relacionados a seu conteúdo ou forma.

Qual o significado da palavra XML?

- a) *eXtensible Markup Language*.
- b) *eXtreme Markup Language*.
- c) *eXtreme Maximum Language*.
- d) *eXtensible Maximum Language*.
- e) *Xtreme Markup Language*.

2. Especificamente, a marcação disponibiliza um mecanismo para adicionar conteúdo e dados sobre a estrutura de documentos. Essas *tags* proporcionam um meio de inserir as informações formadas por caracteres em um elemento ou mais, permitindo que o texto seja intercalado por informações que são relacionadas ao conteúdo, por exemplo:

```
<cliente "nome=Joao">
```

O que seria `<cliente>` e o que seria `"nome=Joao"`, respectivamente?

- a) *Tag* e Tabela.
- b) Chave e tabela.
- c) Tabela e Campo.
- d) Chave e atributo.
- e) *Tag* e elemento.

3. A marcação disponibiliza um mecanismo para adicionar conteúdo e dados à estrutura de documentos. Essas *tags* proporcionam um meio de inserir os dados formados por caracteres em um elemento. Eles podem conter outros elementos aninhados em seu interior, chamados de subelementos.

De acordo com o XML abaixo:

```
<aluno>
  <idade>18</idade>
  <curso>Gerenciamento em banco de dados</curso>
  <periodo>diurno</periodo>
  <dados-pessoais>
    <telefone>2232-4789</telefone>
    <celular>2232-4789</celular>
    <e-mail>aluno@estudo.edu</e-mail>
```

```
    </dados-pessoais>  
</aluno>
```

O que seria a *tag* `<dados-pessoais>`?

- a) Elemento principal.
- b) Um subelemento da *tag* `aluno`.
- c) Um subelemento da *tag* `telefone`.
- d) Um elemento secundário.
- e) Uma *tag* secundária.

Seção 4.3

Introdução ao NoSQL

Diálogo aberto

Caro aluno, depois de ter aprendido sobre banco de dados geográficos e XML em banco de dados, entendendo seus conceitos e como manipulá-los, você irá aprender sobre banco de dados não relacionais, ou NoSQL. Um case real que podemos usar como base de um banco de dados não relacional é o Netflix, que utiliza bancos de dados NoSQL para gerir suas atividades. Esse é um exemplo de como o banco de dados não relacional está se mostrando cada vez mais importante no nosso mercado.

Uma vez que o sistema de localização já está disponível e fazendo muito sucesso, o objetivo agora é que o sistema se torne ainda mais conhecido, o que pode ser obtido por meio da publicidade.

A intenção, nesse caso, é exemplificar como a solução facilita a nossa vida. Para salvar as informações do site, não podemos usar o mesmo banco de dados, por causa da segurança e do tamanho do banco devido às tabelas geográficas. Apresente a estrutura de um banco de dados NoSQL para consulta de produtos, mostrando a criação e a visualização de dados usando o MongoDB.

Com seu esforço, você será capaz de entender como é o funcionamento de um banco de dados não relacional, sabendo criar suas estruturas, consultar suas informações e até manipulá-las. Vamos em frente!

Não pode faltar

Certamente, um dos meios de armazenamento de dados mais tradicionais e conhecidos atualmente são os bancos de dados relacionais baseados nas famosas propriedades ACID (Atomicidade, Consistência, Isolamento, Durabilidade), em que podemos encaixar ferramentas conhecidas, como o Oracle, o MySQL, o SQL Server e o PostgreSQL, por exemplo. Por muito tempo e, na verdade, até hoje,

a grande parte dos bancos de dados estão fundamentados sobre o modelo relacional.

A arquitetura das aplicações evolui a cada ano que passa. Hoje em dia, é relativamente comum que os desenvolvedores se preocupem com itens que não eram levados tanto em consideração no passado, principalmente quando falamos de escalabilidade (capacidade de manipular uma porção crescente de trabalho de forma uniforme, ou estar preparado para crescer) e replicação (permitir a manutenção de várias cópias idênticas de um mesmo dado em vários servidores de bancos de dados). A preocupação com esses itens é ainda mais intensa atualmente por causa da popularização da *cloud computing*, em que replicação e escalabilidade, por exemplo, são itens essenciais (AMAZON, 2018). As propriedades ACID são as seguintes:

- **Atomicidade:** ou todas as operações são persistidas ou todas são desfeitas;
- **Consistência:** se uma transação é realizada no banco de dados, as informações envolvidas devem ser armazenadas corretamente na estrutura do banco de dados;
- **Isolamento:** as transações podem ser realizadas de maneira concorrente e isolada umas das outras. Uma transação não pode impactar o resultado de outra;
- **Durabilidade:** diz respeito à capacidade de um banco de dados conseguir retornar a seu último estado válido após uma falha (uma queda de energia, por exemplo).

Aí entra um grande problema: bases de dados relacionais podem apresentar dificuldades ao tentarmos aplicar técnicas como a replicação e a escalabilidade. Temos ainda outra situação em que os bancos de dados relacionais podem acabar apresentando problemas: o armazenamento de dados estruturados. Com os conceitos modernos de *big data*, a forma como os dados são persistidos também mudou. Novas estruturas de dados mais complexas começaram a ser mais comuns, como sub-colunas e tabelas pivotadas (tabelas dinâmicas).

O NoSQL surgiu como alternativa aos bancos relacionais. Essa necessidade surgiu por causa da quantidade de informações que hoje são gerenciadas por aplicações, da necessidade de um suporte melhor a formatos estruturados de maneira diferente e, principalmente, de questões de escalabilidade e replicação (EDLICH, 2018).

Perceba que não estamos afirmando que bancos de dados relacionais não podem ser replicados ou, principalmente, escalados. Bancos de dados relacionais podem ser replicados e escalados, mas a forma como isso ocorre é muito custosa em relação aos bancos de dados relacionais, principalmente em relação à escalabilidade.

Vamos supor que você precise expandir e melhorar a performance de um banco de dados relacional. Imaginando que ele está em um servidor específico, provavelmente você precisará adicionar mais memória, mais poder de armazenamento (o que se traduz em discos rígidos maiores) e mais poder de processamento (o que se traduz em processadores mais "potentes"). Quanto mais performance e necessidade de replicação forem necessárias, maior o disco rígido em que você irá trabalhar, com quantidades maiores de memória e processadores cada vez mais potentes. Quando temos esse tipo de situação, costumamos dizer que há escalabilidade vertical (quando adicionamos recursos a um sistema existente). A performance e a capacidade de resposta de banco de dados relacionais, de maneira geral, está diretamente relacionada não somente à quantidade de servidores, mas também à capacidade computacional. Quanto mais performance for necessária, mais servidores com maior capacidade computacional serão necessários.

Bancos de dados NoSQL, por sua arquitetura interna, não trabalham com escalabilidade vertical. A escalabilidade deles ocorre de maneira horizontal, isso quer dizer que, se você precisar escalar e replicar um banco de dados NoSQL, você precisará de mais computadores, mas não necessariamente esses computadores precisarão ter um grande poder de processamento. Com máquinas bem mais modestas, é possível escalar (aumentar) a performance e melhorar a replicação quando lidamos com bancos de dados NoSQL (EDLICH, 2018).

Ainda podemos tratar de outras diferenças que existem entre bancos de dados relacionais e NoSQL. Veja no Quadro 4.6, que foi baseado em um *whitepaper* da Amazon, as principais diferenças entre esses dois tipos de bancos de dados (SQL e NoSQL) (AMAZON, 2018).

Tabela 4.6 | Diferenças entre banco de dados SQL e NoSQL

Item analisado	SQL	NoSQL
Modelo de dados	Baseado em tabelas que são agrupadas em <i>schemas</i> (que constituem uma metadescrição das estruturas do banco de dados – tabelas, índices, chaves, etc.)	Não possuem um tipo de estrutura definido, variando de banco para banco. Porém, geralmente possuem uma estrutura para recuperação de registros, chamada <i>chave de partição</i> .
Propriedades ACID	Seguem as propriedades ACID fielmente.	Geralmente abrem mão de algumas propriedades em troca de escalabilidade horizontal.
Desempenho	Diretamente relacionado à capacidade computacional (disco, memória e processador)	É muito mais relacionado a propriedades de hardware adjacentes (como a infraestrutura de rede) e à maneira como a aplicação acessa o banco de dados NoSQL
Escalabilidade	Escalabilidade vertical	Escalabilidade horizontal
APIs (consulta e manipulação de dados)	Feita por meio da linguagem SQL, que é gerenciada por subsistemas de controle (os chamados RDBMS – <i>sistemas de gerenciamento de bancos de dados relacionais</i> em português).	Baseada em objetos, que podem ser armazenados facilmente em memória, contribuindo com a performance.

Fonte: elaborado pelo autor.

Mesmo com todas essas características interessantes dos bancos de dados NoSQL, isso não quer dizer que devemos simplesmente abolir o modelo relacional. Na verdade, cada um deles é adequado a situações específicas, o que faz com que a situação ideal seja a combinação de ambas as arquiteturas. Em pontos em que há a necessidade de persistência mais “segura” (ou seja, a integridade dos dados é essencial), prefira bancos de dados relacionais, pois eles implementam as propriedades ACID em sua integridade. Agora, se seu objetivo é armazenar dados não normalizados, como em um *Data Warehouse*, um banco de dados NoSQL será mais interessante,

já que bancos de dados NoSQL não se preocupam tanto com uma estrutura rígida como são as tabelas.

Se você quer, na verdade, montar uma camada de cache de dados para sua aplicação, também prefira bancos de dados NoSQL, já que eles armazenam as informações em objetos e também utilizam de maneira mais eficiente a memória, acelerando a velocidade de resposta. Agora, se você precisa de portabilidade de dados (aplicações que devem funcionar para diferentes bancos de dados em diferentes situações), prefira bancos de dados relacionais. Como bancos de dados NoSQL não possuem uma estrutura rígida, cada um deles implementa sua estrutura interna de maneira completamente diferente, o que atrapalha processos de migração de dados, por exemplo. O mesmo não ocorre em bancos de dados relacionais, pois todos eles seguem estruturas extremamente semelhantes.



Assimile

É importante não se esquecer que os bancos de dados NoSQL foram criados para serem um complemento aos bancos de dados relacionais, não para substituí-los. Isso pode ser comprovado pela própria sigla, que significa Not Only SQL. O criador do conceito, Carlo Strozzi, até mesmo defende que o nome dado não reflete muito bem a intenção desse modelo. Para ele, deveria se chamar NoREL, ou seja, não-relacional.

O MongoDB é um banco de dados de alta performance, orientado a documentos, é poderoso, flexível e escalável (MONGODB, 2018). Ele não usa o conceito de tabelas, e sim um conjunto de documentos em formato JSON (JavaScript Object Notation), o que nos permite modelar dados de forma mais natural. Isso significa que em vez de ligar uma linha de uma tabela à linha de outra tabela, podemos colocar nesta linha outro documento ou um *array* (MONGODB, 2018).

Não há estruturas definidas, e vários documentos de uma mesma coleção podem ter formatos diferentes, o que não ocorre no mundo relacional, em que temos que ter os dados de uma mesma tabela seguindo um formato pré-definido (MONGODB, 2018).

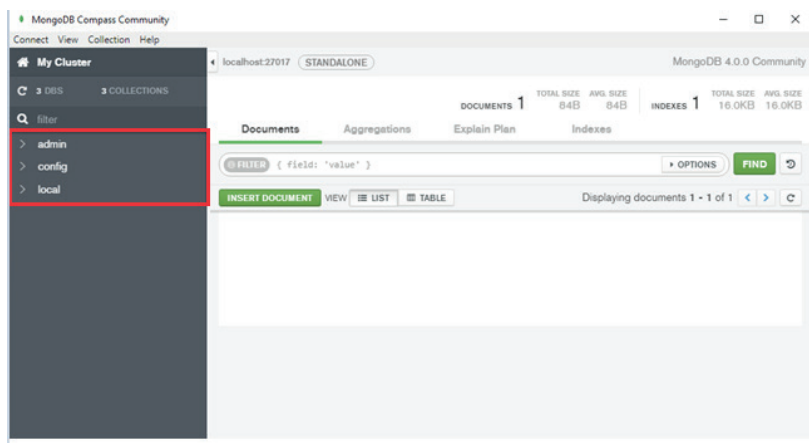


Existem bancos de dados não relacionais específicos para cada situação, como geolocalização, *Big Data* ou até documentos?

Para usar o MongoDB, acesse a página de download oficial e baixe o instalador (<<https://www.mongodb.com/download-center>>). Em seguida, execute o instalador, sem precisar realizar alterações durante a instalação. Após a instalação, a tela do MongoDB Compass Community será aberta. Como será sua primeira conexão, é possível clicar em conectar para acessar nosso banco de dados do MongoDB.

Na Figura 4.11 você irá visualizar três bancos, que são criados durante a instalação: admin, config e local.

Figura 4.11 | Tela Inicial do Compass Community



Fonte: captura de tela do MongoDB Compass Community, elaborada pelo autor.

Recomenda-se criar um novo para que não haja conflito com nomes dos outros bancos de dados na hora de pesquisar pelo sistema ou tentar conectar. Depois de criado o banco de dados, você pode trabalhar com sua manipulação dentro dele, criando novos documentos e inserindo informações nesses documentos. Quando se conectar ao seu banco de dados (Figura 4.12), você pode criar suas *collections* e, dentro dele, seus documentos em formato BSON (*Binary JSON*).

Figura 4.12 | Tela *Databases* e suas informações

localhost:27017

3 DBS 3 COLLECTIONS

filter

admin

local

DATABASES

PERFORMANCE

CREATE DATABASE

Database Name	Storage Size	Collections	Indexes
admin	16.0KB	0	2
local	36.0KB	1	1
test	1.2MB	2	2

Fonte: captura de tela do MongoDB Compass Community, elaborada pelo autor.

Diferente do JSON, é basicamente um objeto JavaScript, que possui uma chave que define o nome do campo, seguida de seu respectivo valor.

Abaixo, podemos ver uma representação de um objeto no formato JSON.

```
{
  "nome" : "João",
  "idade" : 15
}
```

Alguns bancos, como o MongoDB, usam o formato JSON para guardar dados, assim como os bancos relacionais utilizam tabelas. Já o BSON é uma representação binária do formato JSON. O MongoDB representa o JSON em binário, estendendo o seu modelo para oferecer mais tipos de dados, além dos que fazem parte da especificação do JSON (MONGODB, 2018). Alguns tipos de dados que são disponibilizados pelo BSON são (MONGODB, 2018):

- *String*;
- *Integer*;
- *Double*;
- *Date*;
- *Byte array*;
- *Boolean*;

- *Null*;
- *BSON object*;
- *BSON array*;
- Código JavaScript;
- *MD5 Binary Data*;



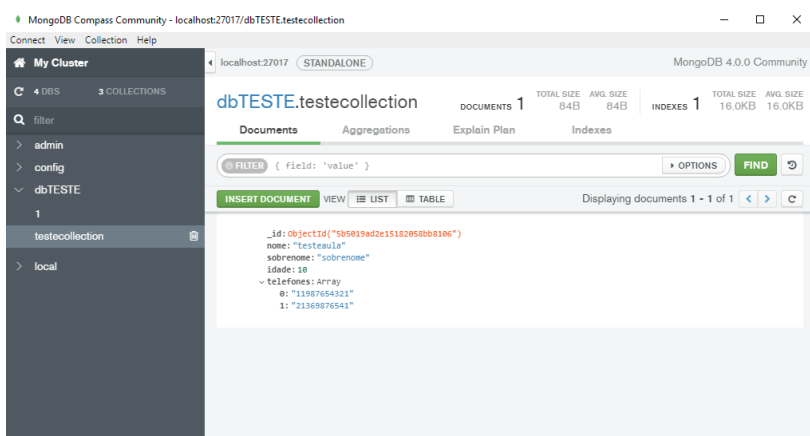
Exemplificando

A estrutura do MongoDB é idêntica ao retorno de um JSON, o que facilita muito o seu desenvolvimento.

```
{
  Nome : "João",
  Idade : 20,
  Endereco: "Rua 13 , número 5"
}
```

Na Figura 4.13 é apresentada a estrutura de um documento na visão de lista, com exemplo de um campo em *array*. Nesta tela, é possível criar documentos, além de montar consultas de resultados específicos.

Figura 4.13 | Tela de inclusão de documentos no MongoDB



Fonte: captura de tela do MongoDB Compass Community, elaborada pelo autor.

Ainda na Figura 4.13, é possível ver que, nessa tela, podemos aplicar filtros e editar ou agrupar valores dos documentos.



Pesquise mais

Apesar de o MongoDB Compass Community ser uma ferramenta robusta, ele não possui interface para que o usuário edite comandos de pesquisa ou movimentações.

Mesmo não existindo essa interface, o MongoDB aceita comandos de *update*, *insert* e busca, mas por meio através do *Shell* (ou CMD).

Para entender melhor o funcionamento desses comandos e quais estão disponíveis, leia o artigo a seguir:

MEDEIROS, H. Introdução ao MongoDB. **DevMedia**, 6 jun. 2014. <<https://www.devmedia.com.br/introducao-ao-mongodb/30792>>. Acesso em: 12 set. 2018.

Com base no que você aprendeu, você será capaz de administrar e gerenciar um banco de dados não relacional como um banco de dados orientado a documentos, como o MongoDB.

Sem medo de errar

Com base no que aprendemos durante a aula, agora é preciso apresentar uma estrutura de um banco de dados NoSQL para consulta de produtos, mostrando a criação e a visualização de dados usando o MongoDB. Vamos apresentar esse caso usando um cadastro de loja de calçados, inserindo informações e, depois, exibindo-as na tela.

No cadastro, teremos os campos “marca”, “tamanho” e “preço”. A estrutura em JSON seria:

```
{ "Marca", "tamanho", "preco" }
```

Primeiramente, vamos inserir no banco de dados as informações do novo produto. Para isso, usaremos o comando *insert*. A sintaxe seria:

```
db.produtos.insert();
```


Logo, para inserir, a sintaxe seria:

```
db.produtos.insert ({ "Marca": "Nike", "tamanho":  
"38", preco: 100 })
```

Ao inserir os dados no documento, é preciso apresentá-los, mostrando que foram realmente inseridos no banco. Para apresentá-los, usaremos outra sintaxe para exibir os produtos, que é o `find()`.

```
db.produtos.find ({ "Marca": "Nike" })
```

Com esse resultado, serão apresentados todos produtos com a marca "Nike".

Avançando na prática

Movimentando informações no banco de dados

Descrição da situação-problema

No dia das mães, uma loja de roupas acabou vendendo todos seus vestidos de cor branca, porém, no sistema ainda aparecem todos os vestidos de cor branca, mesmo sem estoque. Apresente uma solução para que as informações desses vestidos não sejam mais exibidas e mostre, por meio de comando, que os vestidos não serão mais exibidos no sistema. Apresente como ficariam os passos para a consulta e como exibir na tela. A tabela em que se encontram os documentos é a "vestido", que tem os campos "cor", "tamanho" e "preço".

Resolução da situação-problema

Primeiramente, é preciso verificar se os vestidos estão mesmo aparecendo no sistema. Então, primeiro, consulte os resultados.

```
db.vestidos.find ({ "cor": "branca" })
```

Feita a consulta, vamos remover as informações desses vestidos que não estão mais em estoque. Para remover essas informações,

vamos usar o comando `remove()`. Para remover as informações, a sintaxe seria:

```
db.vestidos.remove({"cor": "branca"})
```

Deve-se ter muito cuidado ao rodar esse comando, porque, se não for colocada a condição, tudo que estiver dentro da tabela "vestidos" será apagado. Depois de remover os itens que não estão em estoque, é preciso mostrar que eles foram realmente removidos. Apresente novamente, executando a mesma instrução que foi solicitada inicialmente.

```
db.vestidos.find({"cor": "branca"})
```

Essa consulta irá mostrar que os itens sem estoque não existem mais na tabela.

Faça valer a pena

1. Se você precisar escalar e replicar um banco de dados NoSQL, você precisará de mais computadores, mas não necessariamente esses computadores precisarão ter um grande poder de processamento.

Em relação à escalabilidade, qual é a diferença entre os bancos de dados relacionais e os NoSQL?

- a) Os bancos de dados relacionais possuem escalabilidade horizontal, enquanto os banco de dados NoSQL possuem escalabilidade vertical.
- b) Os bancos de dados relacionais não permitem a realização de escalabilidade, enquanto os banco de dados NoSQL possuem escalabilidade horizontal.
- c) Os bancos de dados relacionais não permitem a realização de escalabilidade, enquanto os banco de dados NoSQL possuem escalabilidade vertical.
- d) Os bancos de dados relacionais possuem escalabilidade vertical, enquanto os banco de dados NoSQL não permitem a realização de escalabilidade.
- e) Os bancos de dados relacionais possuem escalabilidade vertical, enquanto os banco de dados NoSQL possuem escalabilidade horizontal.

2. Diferente do JSON (*JavaScript Object Notation*), é basicamente um objeto JavaScript, que possui uma chave (que define o nome do campo) seguida de seu respectivo valor. O BSON (*Binary JSON*) é uma representação binária do formato JSON. O MongoDB representa o JSON

em binário, estendendo o seu modelo para oferecer mais tipos de dados, além dos que fazem parte da especificação do JSON.

Quais dos tipos de dados abaixo são disponibilizados pelo BSON?

- a) *Time, double, int, string, date.*
- b) *Double, int, string, date.*
- c) *Time, double, int, varchar, datetime.*
- d) *Time, numeric, int, string, date.*
- e) *Time, blob, int, string, date.*

3. O MongoDB é um banco de dados de alta performance, orientado a documentos, poderoso, flexível e escalável. Ele não usa o conceito de tabelas, e sim um conjunto de documentos em formato JSON. Com isso, podemos modelar dados de forma mais natural.

Dentre as alternativas abaixo, quais pertencem a uma estrutura de banco de dados NoSQL?

- a) *Graphs, Documents, Key/Value, Wide Columns.*
- b) *Documents, Key/Value, Column Suport.*
- c) *Column Suport, Graphs, Key.*
- d) *Column Suport, Graphs, Value.*
- e) *Column Suport, Documents, Wide Columns.*

Referências

AMAZON. **Comparação entre banco de dados SQL e NoSQL**. Disponível em: <<https://aws.amazon.com/pt/nosql/>>. Acesso em: 12 set. 2018

EDLICH, S. **List of NoSQL Databases**. [S.d.]. Disponível em <<http://nosql-database.org/>>. Acesso em: 12 set. 2018.

ELMASRI, R. **Sistemas de banco de dados**. 6. ed. São Paulo: Pearson Education, 2011.

FIGUEIREDO, F. J. V.; HIGASI, H.; MURAOKA JUNIOR, L. M. **Xml e banco de dados**. 2003. Disponível em: <<http://meusite.mackenzie.com.br/rogerio/tgi/2003XMLXindex.PDF>>. Acesso em: 12 set. 2018.

GILBERTO R.Q; KARINE R. F. **Tutorial sobre Bancos de Dados Geográficos**. Instituto Nacional de Pesquisas Espaciais. Disponível em: <http://www.dpi.inpe.br/TutorialBdGeo_GeoBrasil2006.pdf>. Acesso em: 12 set. 2018.

MARCELO, G. B; RODRIGO, O. S. Xml e seu armazenamento em banco de dados. **DevMedia**, 19 ago. 2013. Disponível em: <<https://www.devmedia.com.br/xml-e-seu-armazenamento-em-bancos-de-dados/28744>>. Acesso em: 12 set. 2018.

MONGODB. **Documentação do MongoDB**. [S.d.]. Disponível em: <<https://docs.mongodb.com/>>. Acesso em: 12 set. 2018.

NISHIMURA, R. **Banco de dados II**. São Paulo: Pearson Education, 2009.

STONEBRAKER, M. **Object-relational DBMSs: the next great wave**. São Francisco: Morgan Kaufmann, 1996.

ISBN 978-85-522-1132-7



9 788552 211327 >