Estratégia Concursos

Aula 00 - Prof. Thiago Cavalcanti

Câmara de Macapá-AP (Técnico em Segurança da Informação) SGBDs - 2024 (Pós-Edital)

Autor:
Thiago Rodrigues Cavalcanti

15 de Agosto de 2024

Índice

CONCEITOS DE BANCO DE DADOS

APRESENTAÇÃO DO PROFESSOR

Olá,

Sejam bem-vindos a mais um curso de Tecnologia da Informação (TI)! Hoje apresentamos o mais completo curso no que se refere a Banco de Dados para concursos. Gosto sempre de dizer que é um prazer imenso fazer parte desta equipe de professores do Estratégia Concursos e ter a oportunidade de apresentar um pouco do meu conhecimento e experiência em concursos públicos.

Antes de começar de fato o conteúdo teórico desta aula, vou me apresentar de forma rápida. Meu nome é Thiago, sou casado, pernambucano, tenho três filhos, Vinícius (13 anos), Lucas (*in memoriam*) e Júlia (3 anos). Torço pelo Sport Clube do Recife. Sou cristão. Me formei em Ciência da Computação pela UFPE. Tenho mestrado em engenharia de software na mesma instituição. Também tenho doutorado em economia na UnB.

Frequento academia para manter a forma, mas meu hobby mesmo é pedalar! Decidi vender o carro e viver num desafio intermodal de transporte. Ia para o trabalho de *bicicleta* sempre que possível! Ultimamente, com o teletrabalho, a bicicleta é usada apenas nos finais de semana! Agora, uma pergunta: onde eu trabalho? No Banco Central do Brasil!

Fruto de uma trajetória de dois anos de estudos diários. Aposentei as canetas em 2010. Hoje estou lotado no Banco Central em Recife trabalhando com fiscalização das empresas que compõem a infraestrutura do mercado financeiro (IMFs). Hoje, minhas tarefas envolvem verificações e sugestões de TI para melhoria das empresas e do ecossistema financeiro.

Minha experiência com gestão de dados é parte de uma estratégia profissional de alinhar meu trabalho diário como servidor público com minha carreira paralela de professor e consultor de Banco de Dados (BD) e *Business Intelligence* (BI). A ideia é conseguir me especializar cada vez mais no tema desta carreira dentro da TI, que o mercado chama de **cientista dos dados (*Data scientist*)**.

Entrei neste universo de professor de concurso há alguns anos. Desde 2012, tenho me dedicado especificamente ao conteúdo de BD e BI. Minhas experiências em cursos presenciais em Brasília e em diversas partes do Brasil, bem como na gravação sistemática de aulas on-line me ajudaram a desenvolver um conteúdo exclusivo para os alunos do Estratégia Concursos.

A ideia é desenvolver um material completo, recheado de questões e com diversas dicas para ajudar você no seu objetivo: **ser aprovado e nomeado**!

Agora gostaria, humildemente, de fazer um pedido, não deixe de seguir meu perfil no Instagram® (@profthiagocavalcanti), onde eu posto, sistematicamente, questões comentadas e dicas semanais.

Para facilitar sua vida, você pode usar o QR code acima para acessar meu perfil no Instagram. Se precisar falar comigo por e-mail, mande mensagem para:

rcthiago@gmail.com

Por fim, e talvez a dica mais importante relacionada a redes sociais, gostaria de apresentar a vocês o meu canal no Telegram®(https://t.me/profthiagocavalcanti) ... neste canal procuro condensar todas as dicas que apresento nas minhas redes sociais. Na minha opinião, é a melhor forma de acompanhar todas as minhas publicações sem precisar ficar procurando nas redes sociais, lá eu ainda tiro dúvidas (no chat do canal) e interajo diretamente com os alunos. Ou seja, é uma forma de otimizar seus estudos!

Agora que você já me conhece! Vamos seguir em frente com o nosso curso!

PARE TUDO! E PRESTE ATENÇÃO!!

Hoje eu faço parte de uma equipe **SENSACIONAL** de professores: **OS CANETAS PRETAS**! Depois de muita luta conseguimos reunir **um time** de profissionais extremamente **QUALIFICADO** e sobretudo **COMPROMETIDO** em fazer o melhor pelos alunos. Para tal, criamos um conjunto de ações para nos aproximarmos dos alunos, entendermos suas necessidades e evoluirmos nosso material para um patamar ainda mais diferenciado. São 3 as novidades que gostaria de convidá-lo a conhecer:

//estratégia tech ESTRATEGIA CONCURSOS	Nosso podcast alternativo ... livre, descontraído e com dicas rápidas que todo **CANETA PRETA** raiz gosta de ouvir. Já temos alguns episódios disponíveis e vários outros serão gravados... acompanhe em: http://anchor.fm/estrategia-tech
Telegram a new era of messaging	Nosso grupo do Telegram! É um local onde ouvimos os alunos e trocamos ideias. Está crescendo a cada dia. A regra do grupo é: só vale falar sobre concursos. Lá divulgamos nossas aulas ao vivo e falamos sobre os concursos abertos, expectativas de novos concursos, revisões de véspera ... **https://t.me/canetaspretaschat**
Instagram	Criamos um perfil no Instagram ... e qual o objetivo? Fazer com que os alunos percam tempo nas redes sociais? Claro que não!! Estamos consolidando diversos posts dos professores! São dicas especiais, um patrimônio que deve ser explorado por todos os concurseiros de TI ... https://www.instagram.com/canetas.pretas/

MOTIVAÇÃO PARA O CURSO

Preparar esse curso é um desafio! Consolidar de forma amigável o conhecimento de banco de dados, análise de informações ou business Intelligence para concursos não é uma tarefa fácil. Calibrar o nível do teórico associado a uma didática eficiente tem sido minha meta nos últimos anos. Separamos o conteúdo de forma a segmentar e impulsionar seu aprendizado. Para que você entre na primeira aula com um pouco mais de segurança, vou aproveitar para fazer uma rápida apresentação sobre o assunto.

Você já ouviu falar sobre **Data Science ou Ciência dos Dados**? É um conceito relativamente recente que agrupa diversas atividades executadas sobre um conjunto de dados, em especial, sobre grandes conjuntos de dados. Para analisar os dados eles precisam estar **armazenados e organizados** de maneira **conveniente** para os cientistas dos dados. Essa base de dados facilita o trabalho e o entendimento do conteúdo armazenado.

Cientistas de dados são uma nova geração de especialistas em análise que têm habilidades técnicas para resolver problemas complexos e a curiosidade de explorar quais são os problemas que precisam ser resolvidos. A solução desses problemas passa por analisar os dados presentes em um banco de dados. Neste curso veremos o passo-a-passo para a construção de um banco de dados.

Nossa primeira aula deve inserir você no universo dos bancos de dados. Um banco pode ser visto como uma estrutura que armazena algo, por exemplo, um banco de leite guarda leite materno para que possa ser reutilizado de forma adequada em momentos posteriores. Um banco de dados guarda dados que devem ser controlados de forma adequada. É nesse momento que surge um sistema para “cuidar” do acesso consistente aos dados.

Os sistemas de gerenciamento de banco de dados (SGBDs) contribuem para a disponibilidade de um conjunto de informações para diferentes usuários simultaneamente. É preciso decidir quais dados armazenar, estruturar e manter na base de dados. Para controlar esse sistema e todo o desenvolvimento do projeto e da infraestrutura associada ao sistema de banco de dados várias tarefas têm que ser feitas.

Veremos que existem profissionais dedicados a tarefas específicas. Veremos ainda que a construção de um banco de dados, em especial um banco de dados relacional, passa por algumas etapas bem definidas. Essas etapas criam modelos de dados ou esquemas que permitem um melhor entendimento da estrutura de dados da organização ao tentar abstrair a complexidade presente no armazenamento físico dos dados.

Todos esses conceitos serão vistos em detalhes nas próximas páginas. Ao final, teremos nossa tradicional lista de exercícios. Espero conseguir contribuir para a sua aprovação. Vamos em frente?!

Teremos muito trabalho! Por isso, montamos um **curso teórico em PDF,** baseado nas mais diversas bancas, em especial da banca do seu concurso[1], apresentando o conteúdo observando as variadas formas de cobrança do mesmo pelas bancas examinadoras.

Teremos ainda videoaulas que apresentam o conteúdo teórico de forma detalhada para todo o conteúdo deste curso. Caso você não esteja visualizando os vídeos, peço que entre em contato comigo, o mais rápido possível, para que eu possa associá-los às respectivas aulas.

Ao final deste curso, nosso objetivo é garantir que você tenha capacidade e conhecimento para ser aprovado.

Observação importante: este curso é protegido por direitos autorais (copyright), nos termos da Lei 9.610/98, que altera, atualiza e consolida a legislação sobre direitos autorais e dá outras providências.

Grupos de rateio e pirataria são clandestinos, violam a lei e prejudicam os professores que elaboram os cursos. Valorize o trabalho de nossa equipe adquirindo os cursos honestamente através do site Estratégia Concursos ;-)

Observação importante II: todo o conteúdo deste curso encontra-se completo em nossos textos escritos. As videoaulas visam reforçar o aprendizado, especialmente para aqueles que possuem maior facilidade de aprendizado com vídeos e/ou querem ter mais uma opção didática.

Motivação I: Para se inspirar, aqui estão algumas frases motivacionais que podem ajudar a manter o foco e a determinação nos estudos:

“As raízes do estudo são amargas, mas seus frutos são doces.” - Aristóteles
“Não deixe seu futuro nas mãos da sorte, comece hoje mesmo a estudar e lutar pelo seu sucesso.” - Provérbio Chinês
“O lucro dos nossos estudos é tornarmo-nos melhores e mais sábios.” - Michel de Montaigne
“Investir em conhecimento rende sempre os melhores juros.” - Benjamin Franklin
“Estudar é crescer em silêncio!”

Agora vamos voltar para a nossa aula. Vamos juntos? Se você tiver alguma dúvida, por favor, não hesite em perguntar.

[1] Sempre que possível a lista de questões comentadas virá com diversas questões da sua banca. Entretanto, é possível que, por questões de didática ou carência de questões, existam questões de outras bancas nas aulas do seu curso.

INTRODUÇÃO DO MYSQL

Queremos aqui contextualizar MySQL para que você possa se situar dentro do assunto.

O QUE É MYSQL?

O banco de dados open source mais popular do mundo. Essa é a frase de marketing presente no site ***mysql.com***. Hoje de propriedade da Oracle, o MySQL continua em evolução, sendo sempre considerado como uma opção para o armazenamento de dados empresariais. Mas nem sempre foi assim.

Tudo começou em 1995, com uma empresa fundada na Suécia, cujo nome era **MySQL AB**. Essa empresa viria a desenvolver a primeira versão do SGBD. Essa versão do sistema foi entregue em 23 de maio do mesmo ano. Em 2008, a Sun Microsystems comprou a MySQL AB por, aproximadamente, um bilhão de dólares. A Sun seria comprada pela Oracle alguns anos depois, em 2010.

Enquanto software livre, o MySQL teve seu crescimento estimulado pelo pacote LAMP (Linux, Apache, MySQL e PHP). Hoje é possível usar a versão ***MySQL Community Edition*** que continua sobre a guarda da licença GPL. A Oracle também fornece versões pagas (***Standard, Enterprise e Cluster***) do mesmo sistema, agregando outra licença que permite o uso comercial, algumas ferramentas e, principalmente, suporte técnico.

Um exemplo de ferramenta que traz vantagem à edição MySQL Enterprise sobre edição comunitária é o *MySQL Thread Pool*. Ela fornece um modelo altamente escalável, com manipulação baseada em filas e foi projetada para reduzir a sobrecarga no gerenciamento de conexões de clientes e de threads de execução das instruções. Outras extensões proprietárias também podem ser instaladas como plug-ins aos servidores.

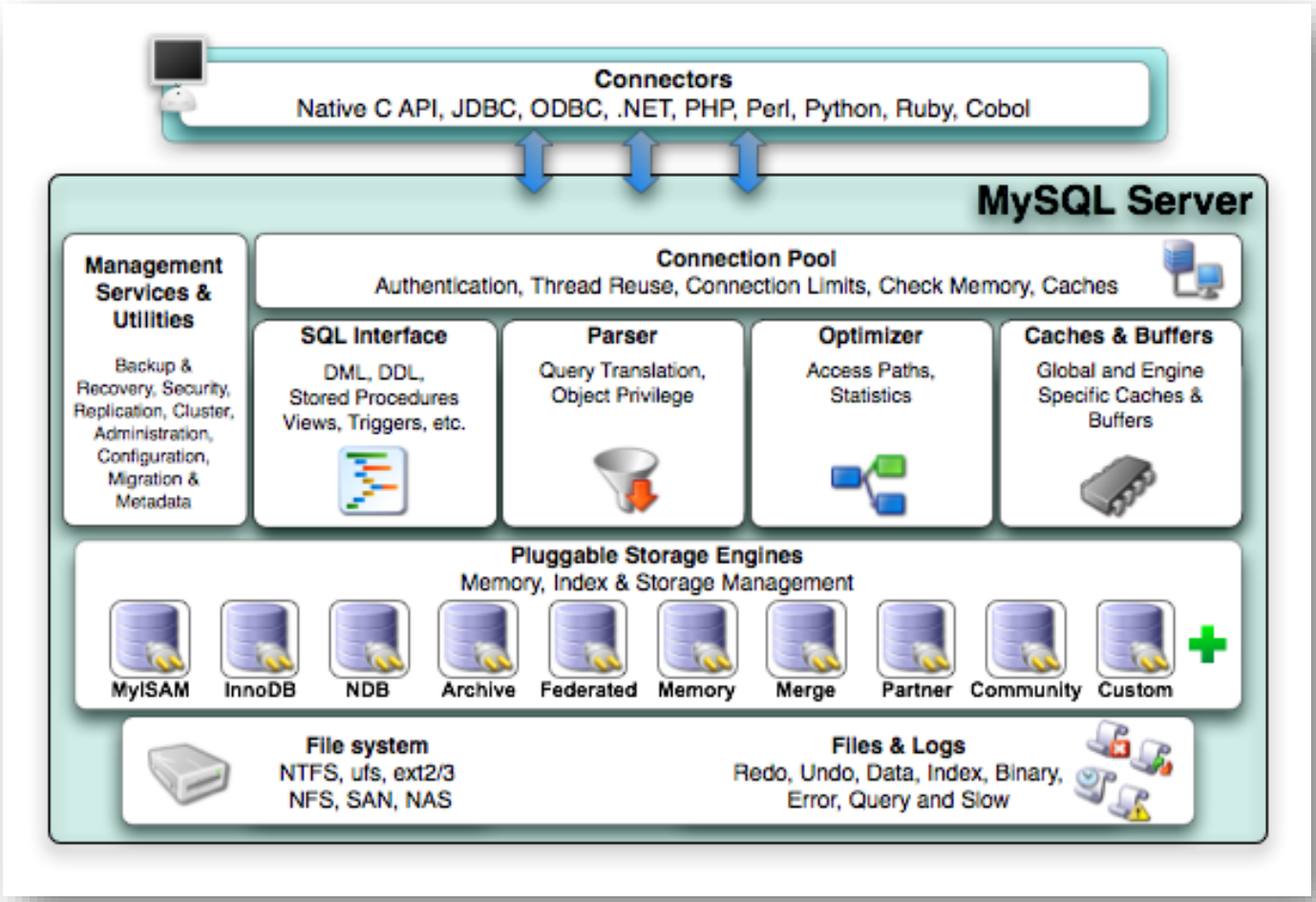
ARQUITETURA

Quando pensamos em Sistemas de Gerenciamento de Banco de Dados (SGBDs), sabemos que ele deve ser composto por um conjunto de funcionalidades que dão suporte à adequada manutenção dos dados. As transações devem ser gerenciadas para atingir os objetivos de **atomicidade**, **consistência**, **isolamento** e **durabilidade**. É preciso também coordenar a concorrência entre as transações, evitando ***deadlock***, ***livelock*** e ***starvation***.

Os conceitos de **backup e recuperação** devem ser construídos e constituídos de forma que, após uma falha, o SGBD consiga trazer de volta sua base de dados para um estado válido, que seja mais

próximo possível do momento da falha. Temos ainda necessidades de **segurança das informações** e **otimização de consulta**. A figura a seguir mostra como todos esses e outros elementos estão estruturados dentro da arquitetura do *MySQL Server*. Vamos passar por cada um deles, tentando expor suas principais características.

A arquitetura do mecanismo de armazenamento do MySQL o torna único no mundo do banco de dados, especialmente, pela interessante camada de mecanismos de armazenamento plugáveis (*pluggable storage engines layer*).



Conectores (*Connectors*): O MySQL fornece drivers baseados em diversos padrões para JDBC, ODBC e .Net, que permitem aos desenvolvedores criarem aplicativos de banco de dados na linguagem de sua preferência. Além disso, uma biblioteca nativa em C (*Native C API*) permite aos desenvolvedores incorporar instruções de *MySQL* diretamente em suas aplicações.

Pool de conexões (Connection Pool): Um pool de conexão no servidor fornece autenticação e gerencia threads, conexões, memória e caches.

Management Services & Utilities: Esse componente da arquitetura é responsável por uma gama enorme de atividades. Para não nos alongarmos muito, falaremos do gerenciamento de armazenamento, de transações e da recuperação. Aproveitaremos também para listar alguns utilitários disponíveis ao servidor.

- **Gerenciamento de armazenamento** – A interface que gerencia o armazenamento junto ao sistema operacional visa gravar dados no disco de forma eficiente. As funções de armazenamento residem em um subsistema separado ao da *engine MySQL*. A interface opera em um nível de abstração acima do sistema operacional. O gerenciador de armazenamento em disco escreve todos os dados do usuário, índices e logs, bem como os dados do sistema interno nas tabelas.
- **Gerenciamento de transação** - A função do gerenciador de transações é facilitar a concorrência no acesso aos dados. Este subsistema fornece um mecanismo de bloqueio para garantir que vários usuários simultâneos acessem os dados de forma consistente, sem corromper ou danificar os dados. O controle de transação é realizado através do subcomponente gerenciador de bloqueio, o qual retém e libera os *locks* sobre os vários objetos que estão sendo usados em transação.
- **Gerenciamento de recuperação** - O trabalho do gerenciador de recuperação é manter cópias de dados para recuperação posterior, em caso de perda de dados. Ele também registra comandos que modificam os dados e outros eventos significativos dentro do banco de dados.

Utilitários - Os utilitários podem ser divididos em algumas categorias que incluem: Utilitários de administração (*Clone, Copy, Compare, Diff, Export, Import, User Management*), Utilitários de replicação (*Setup, Configuration, Verification*) e Utilitários de uso geral (*Disk Usage, Redundant Indexes, Manage Meta & Audit Data*).

Vamos, agora, antes de continuarmos nossa viagem pelos componentes da arquitetura, fazermos uma questão de prova!

1. CESPE - Auditor Municipal de Controle Interno (CGM João Pessoa)/Tecnologia da Informação/ Desenvolvimento de Sistemas/2018]

A respeito de bancos de dados, julgue o item a seguir.

O MySQL Utilities é um pacote de utilitários voltados à manutenção e à administração de servidores MySQL. Cada um desses utilitários encapsula comandos primitivos e os agrupa para que possam ser utilizados no cumprimento de operações compostas a partir da execução de um único comando.

Certo

Errado

Comentários: A questão está correta. Realmente o MySQL Utilities é um pacote de utilitários que são usados para manutenção e administração de servidores MySQL. Os utilitários podem ser instalados com o MySQL Workbench ou como um pacote independente. Eles são um conjunto de utilitários de linha de comando e uma biblioteca Python para facilitar a realização das tarefas comuns. A biblioteca é escrita inteiramente em Python, o que significa que não é necessário ter outras ferramentas ou bibliotecas instaladas para fazê-lo funcionar. Atualmente, ele foi projetado para funcionar com o Python v2.6 ou posterior e não há suporte para o Python

v3.1. Os utilitários estão disponíveis sob a licença GPLv2 e são extensíveis usando a biblioteca fornecida.

Gabarito: C

Continuaremos nosso estudo sobre os elementos presentes na arquitetura descrita na figura anterior.

Interface SQL: Fornece os mecanismos para receber comandos e transmitir resultados para o usuário. Foi construída para atender o padrão SQL/ANSI e aceita as mesmas instruções SQL básicas que a maioria dos servidores de banco de dados compatíveis com o padrão, embora vários comandos SQL suportados em MySQL tenham opções que não são definidas pelo padrão. Como todo banco de dados relacional, SQL apresenta extensões na linguagem SQL.

As conexões com o servidor de banco de dados são recebidas pela rede e uma *thread* é criada para cada conexão. O processo de criação de threads é o coração da execução no servidor MySQL. O MySQL é construído como uma verdadeira aplicação com vários processos. Cada processo é executado de forma independente dos outros (com exceção de certos encadeamentos auxiliares). O comando SQL é recebido pela interface e é armazenado em uma estrutura de dados temporária. Os resultados são, então, transmitidos para o cliente, pelos protocolos de comunicação da rede. Uma vez que uma *thread* foi criada, o servidor MySQL tenta analisar o comando SQL e armazenar as partes da solicitação na estrutura de dados interna.

Parser: Quando um cliente emite uma consulta, uma nova *thread* é criada e a instrução SQL é encaminhada para o ***parser*** para validação sintática (ou rejeição devido a erros). O analisador MySQL é implementado usando um script **Lex-YACC** que é compilado com ***Bison***. O *parser* constrói uma estrutura de consulta usada para representar a instrução de consulta SQL na memória como uma estrutura de árvore (também conhecida como **árvore sintática abstrata**), que pode ser usada durante a execução da consulta.

Otimizador de consulta: O otimizador utiliza uma estratégia SELECT-PROJECT-JOIN que tenta, primeiramente, reestruturar a consulta, executando as restrições (SELECT) para reduzir o número de tuplas. Em seguida, executa as projeções para reduzir o número de atributos (campos) nas tuplas, e, finalmente, avalia quaisquer condições de junção. Apesar de não ser considerado um membro da categoria de otimizadores de consulta baseado em custos, o método SELECT-PROJECT-JOIN se enquadra na categoria de otimizadores heurísticos. Neste caso, as heurísticas (regras) são simplesmente:

- Eliminar horizontalmente dados em excesso, avaliando as expressões na cláusula WHERE (ou HAVING).
- Eliminar verticalmente os dados extras ao limitar as colunas aos atributos especificados. A exceção é o armazenamento dos atributos utilizados na cláusula de junção que não podem eliminados, pois são usados na parte final da consulta.
- Avaliar as junções.

Isso resulta em uma estratégia que assegura ao método de acesso boas condições para recuperar dados de um modo eficiente. Apesar das críticas, a estratégia SELECT-PROJECT-JOIN tem se mostrado eficaz em executar as consultas típicas encontradas no processamento de transações.

É possível também melhorar a consulta por meio de sua análise através do comando **EXPLAIN**. O MySQL ***explain plan*** é uma ferramenta de grande utilidade para ajudar os desenvolvedores e administradores de banco de dados a melhorar o desempenho das consultas específicas que acontecem em um banco de dados. Vejam um exemplo do comando abaixo:

EXPLAIN select name, created from recipes where created > '2011-11-01 00:00:00' **\G**

É possível perceber elementos no início e no final da linha:

EXPLAIN, no início da linha, significa que teremos como resultado informações sobre esta consulta em vez dos resultados retornados pela mesma. No final da linha, o ponto e vírgula (;) é substituído por **\G**. Este pode ser utilizado com qualquer instrução SQL e faz com que uma lista de nomes de colunas e seus respetivos valores sejam apresentados verticalmente em vez de horizontalmente. Isso é útil porque a saída de EXPLAIN pode ser difícil de ler horizontalmente.

No MySQL 5.7, é permitido o uso da instrução EXPLAIN para os comandos SELECT, DELETE, INSERT, REPLACE e UPDATE.

```
{EXPLAIN | DESCRIBE | DESC}
    tbl_name [col_name | wild]

{EXPLAIN | DESCRIBE | DESC}
    [explain_type]
    {explainable_stmt | FOR CONNECTION connection_id}

explain_type: {
    EXTENDED
  | PARTITIONS
  | FORMAT = format_name
}

format_name: {
    TRADITIONAL
  | JSON
}
```

Veja a figura acima. Existem duas informações relevantes, começando de baixo para cima. Veja que o ***format_name*** pode receber os valores **TRADITIONAL** e **JSON**. Basicamente, esse parâmetro serve para definir o formato de saída das informações geradas durante a execução do comando. Outro parâmetro que devemos levar em consideração é o ***explain_type***. Nele apresentamos as opções de incluir os dados adicionais sobre o plano de execução, obtidos por meio do parâmetro EXTENDED ou, ainda, utilizarmos o PARTITIONS para descrever quais as partições são utilizadas nas consultas.

Caches & Buffer: Esse subsistema é responsável por garantir que os dados mais frequentemente usados (ou estruturas) estejam disponíveis da forma mais eficiente possível. Em outras palavras, os dados devem estar prontos para serem lidos em todos os momentos. O uso de caches melhora drasticamente o tempo de resposta aos pedidos porque os dados estão na memória e, portanto, não precisamos de acesso adicional ao disco para recuperá-los. O subsistema de cache foi criado para encapsular todo o mecanismo cache e buffer em um conjunto de biblioteca de funções de baixo

acoplamento. Embora você possa encontrar os caches implementados em vários arquivos de código-fonte diferentes, eles são considerados parte do mesmo subsistema.

Uma variedade de diferentes caches é implementada neste subsistema. A maioria dos mecanismos de cache usa os mesmos conceitos de armazenamento de dados em estruturas como uma lista ligada. Os caches são implementados em diferentes partes do código para adaptar a aplicação ao tipo de dados que está sendo armazenado em cache. Vamos olhar para cada um dos possíveis caches utilizados no MySQL.

Table Cache - Foi criado para minimizar a sobrecarga na abertura, leitura e fechamento de tabelas (arquivos **.FRM** em disco). Por esta razão, o cache de tabela é projetado para armazenar metadados sobre as tabelas em memória. Isso torna muito mais rápido para um processo ler o esquema da tabela sem ter de reabrir o arquivo. Cada thread tem sua própria lista de estruturas de cache de tabela.

Record Cache - Foi criado para melhorar leituras sequenciais dos mecanismos de armazenamento. Assim, o cache de registro normalmente só é usado durante as varreduras de tabela. Ele funciona como um *read-ahead (*leitura antecipada*) buffer*, recuperando um bloco de dados a cada leitura, resultando assim em menos acessos ao disco durante a varredura. Menos acessos a disco geralmente equivale a um melhor desempenho. Este comportamento sequencial, chamado localidade de referência, é a principal razão pela qual o cache de registro é mais frequentemente usado com o mecanismo de armazenamento MyISAM, embora não seja limitado ao mesmo.

Key Cache - O cache de chaves é um buffer para dados de índice usados com frequência. Neste caso, é um bloco de dados para o arquivo de índice (árvore-B) e é usado exclusivamente para tabelas MyISAM (os arquivos **.MYI** em disco). Os próprios índices são armazenados como listas ligadas dentro da estrutura de chave de cache. O cache de chaves é criado quando uma tabela MyISAM é aberta pela primeira vez. E é acessado em cada leitura do índice. Se um índice for encontrado no cache, ele é lido a partir daí. Caso contrário, um novo bloco de índice deve ser lido a partir do disco e colocado em cache. No entanto, o cache tem um tamanho limitado e é ajustável mudando a variável de configuração **key_cache_block_size**.

Privilege Cache - O cache de privilégios é usado para armazenar **dados de permissões** para uma conta de usuário. Estes dados são guardados da mesma maneira que uma lista de controle de acesso (ACL). A lista com todos os privilégios de um usuário torna-se um objeto no sistema. Os dados são recolhidos para o cache quando as tabelas de permissões são lidas durante a autenticação de usuário e inicialização. É importante armazenar estes dados em memória, pois se economiza muito tempo que seria gasto lendo as tabelas de permissões.

Hostname Cache - O cache de *hostname* é outro caches auxiliar, como o cache de privilégio. Ele também é implementado com uma estrutura de pilha. Ele contém os nomes de hosts de todas as conexões para o servidor. Pode parecer surpreendente, mas estes dados são frequentemente solicitados e, portanto, candidatos para um cache dedicado.

Pluggable Storage Engines: Os mecanismos de armazenamento do MySQL são componentes que lidam com as operações SQL para diferentes tipos de tabela. ***InnoDB*** é o padrão de **motor de armazenamento** de propósito geral, a Oracle recomenda usá-lo para as tabelas, exceto para os casos

de aplicações especializadas. O comando CREATE TABLE desde o MySQL 5.5 cria, por padrão, tabelas InnoDB.

O MySQL Server usa uma arquitetura de mecanismo de armazenamento *plugável* que permite os mecanismos serem carregados e descarregados dentro de um servidor MySQL em execução. Para determinar quais os motores de armazenamento seu servidor suporta, use o comando **SHOW ENGINES**. O valor na coluna ***Support*** indica se o motor pode ser usado. O valor de YES, NO, ou DEFAULT indica se um motor está disponível.

Observe a figura a seguir com um exemplo do comando:

```
mysql> SHOW ENGINES\G
*************************** 1. row ***************************
      Engine: PERFORMANCE_SCHEMA
     Support: YES
     Comment: Performance Schema
Transactions: NO
          XA: NO
  Savepoints: NO
*************************** 2. row ***************************
      Engine: InnoDB
     Support: DEFAULT
     Comment: Supports transactions, row-level locking, and foreign keys
Transactions: YES
          XA: YES
  Savepoints: YES
```

Vamos agora fazer uma questão e mostrar que o entendimento do assunto até aqui já nos ajuda a acertar algumas questões:

2. BANCA: FGV ANO: 2013 ÓRGÃO: AL-MT PROVA: ANALISTA DE SISTEMAS - BANCO DE DADOS

O engine padrão utilizado pelo MySQL Storage a partir da versão 5.5 é o

A MyISAM.

B MyVSAM.

C MyInnoDB

D InnoDB.

E ISAM.

Comentário: São várias as possibilidades de ***engines*** que tem compatibilidade com a arquitetura do MySQL. Apenas para listar alguns exemplos temos: **MyISAM, InnoDB, NDB, Archive, Federated, Memory, Merge, Partner, Community, Custom, CSV, Blackhole e Example.** Vamos então apresentar algumas características das principais engines disponíveis. Os mecanismos de armazenamento do MySQL 5.7 incluem tanto aqueles que lidam com tabelas transacionais quanto aqueles que tratam com estruturas de dados não transacionais.

InnoDB: O mecanismo de armazenamento **padrão a partir do MySQL 5.5**. É um mecanismo de armazenamento seguro para transações. Possui os comandos de *commit* e *rollback*, e a capacidade de recuperação após falha para proteger os dados do usuário. Ele segue, portanto, as propriedades do ACID. *InnoDB* possui bloqueio em nível de linha e um estilo Oracle de **leitura**

nonlocking, consistente com o aumento de concorrência multiusuário e com a melhoria de desempenho. Ele armazena dados do usuário em índices agrupados (*clustered*) para reduzir I/O em consultas comuns com base nas chaves primárias. Para manter a integridade dos dados, o *InnoDB* também suporta **restrições de integridade referenciais** por meio de FOREIGN KEY.

MyISAM: Seu nível de bloqueio por tabela limita o desempenho em leitura/escrita das cargas de trabalho (*workload*), por isso é frequentemente usado em *workloads* somente de leitura ou preponderantemente de leitura, por exemplo, em algumas aplicações Web e armazém de dados (*Data Warehouse*).

Memory: Armazena todos os dados em memória RAM, para acesso rápido em ambientes que exigem pesquisas rápidas de dados não críticos. Este motor era anteriormente conhecido como **HEAP**. Seus casos de uso estão diminuindo. O *InnoDB* com a sua área de memória para *buffer pool* fornece uma maneira de uso geral e durável para manter a maioria ou todos os dados na memória. E o **NDBCLUSTER** fornece pesquisas de chaves-valor rápidas para grandes conjuntos de dados distribuídos.

CSV: Suas tabelas são realmente arquivos de texto com valores separados por vírgulas. Tabelas CSV permitem importar ou descarregar dados em formato CSV para trocar dados com scripts e aplicativos que leem e escrevem no mesmo formato. Como as tabelas CSV não são indexadas, você costuma manter os dados em tabelas *InnoDB* durante a operação normal e só usar tabelas CSV durante a fase de importação ou exportação.

Archive: Estas tabelas compactas e não indexadas são destinadas a armazenar e a recuperar grandes quantidades de informações que raramente são referenciadas. Como exemplo, nós podemos citar arquivos de histórico ou arquivos de auditoria de segurança.

Blackhole: O mecanismo de armazenamento *Blackhole* aceita, mas não armazena dados, similar ao Unix dispositivo */dev/null*. Consultas sempre retornam um conjunto vazio. Estes podem ser usados em configurações de replicação, em que comandos DML são enviados para servidores escravos, mas o servidor mestre não mantém sua própria cópia dos dados.

Merge: Permite um DBA MySQL ou desenvolvedor agrupar logicamente uma série de tabelas *MyISAM* idênticas e referenciá-las como um objeto. Bom para ambientes VLDB (*Very Large Database*), tais como o armazenamento de grandes conjuntos de dados.

Federated: Oferece a capacidade de vincular servidores MySQL separados para criar um banco de dados lógico de muitos servidores físicos. Muito bom para ambientes distribuídos ou *datamart*.

Example: Este motor serve como um exemplo no código fonte do MySQL que ilustra como começar a escrever novos mecanismos de armazenamento. É bastante interessante aos desenvolvedores. O mecanismo de armazenamento é um "*stub*" que não faz nada. É possível criar tabelas com este motor, mas os dados não podem ser armazenados ou recuperados a partir deles.

Você não está restrito a utilizar o mesmo motor de armazenamento para um servidor ou um esquema inteiro. Você pode especificar o mecanismo de armazenamento para qualquer tabela. Por exemplo, uma aplicação pode usar tabelas *InnoDB* para a maioria dos dados, com

uma tabela CSV para exportar dados para uma folha de cálculo e algumas tabelas de memória para espaços de trabalho temporários.

Gabarito: D

VERSÕES

Hoje as versões que continuam com o desenvolvimento ativo são MySQL Server 5.5, MySQL Server 5.6, MySQL Server 5.7 e MySQL Server 8.0. As demais foram descontinuadas, embora continuem existindo como SGBDs para diversas aplicações. Veremos abaixo quais as principais melhorias que foram incorporadas em cada uma dessas versões.

COMO SE DEU A EVOLUÇÃO DAS FUNCIONALIDADES

Abaixo segue a primeira lista de novas funcionalidade que foram incluídas na versão 5.5 do MySQL.

1. **Storage engine padrão** - foi modificada para o ***InnoDB***, que suporta transação e restrição de integridade referencial. O **subsistema de I/O** do *InnoDB* também foi modificado para melhor uso da capacidade de I/O.
2. **MySQL Enterprise Thread Pool** – Já tratamos do Thread pool no início da aula. Só relembrando: ele fornece um modelo altamente escalável, com manipulação baseada em filas e foi projetado para reduzir a sobrecarga no gerenciamento de conexões de clientes e threads na execução de instruções.
3. **MySQL Enterprise Audit** - utiliza a API de auditoria do MySQL para permitir o monitoramento padrão com base em políticas e registros de conexão e na atividade de consultas executadas sobre servidores MySQL específicos.
4. **Pluggable authentication -** Autenticação MySQL suporta dois novos recursos: **autenticação conectáveis** e **proxy de usuários**. Com **autenticação conectável**, o servidor pode usar plug-ins para autenticar conexões do cliente e os clientes podem carregar um plug-in de autenticação que interage adequadamente com o plug-in do servidor correspondente. Esta capacidade permite aos clientes se conectarem ao servidor MySQL com credenciais apropriadas em vez de usar outros métodos de autenticação nativos do MySQL baseado em senhas armazenadas na tabela mysql.user. Por exemplo, plug-ins podem ser criados para usar métodos de autenticação externos, tais como **LDAP, Kerberos, PAM ou Windows**. O **proxy de usuário** permite que um cliente se conecte e autentique como um usuário. Este usuário será tratado, para fins de controle de acesso, enquanto conectado, como outro usuário com seus respectivos privilégios. Sendo assim, um usuário se faz passar por outro.
5. **Multi-core scalability -** A tendência no desenvolvimento de hardware é em direção ao aumento de núcleos em vez de aumentos contínuos na velocidade do *clock* da CPU. Em outras palavras, esperar até chegarem CPUs mais rápidas no mercado não é um meio viável para melhorar o desempenho do banco de dados. Em vez disso, é necessário fazer um melhor uso de múltiplos

núcleos para explorar ao máximo os ciclos de processamento que eles dispõem. A ideia é usar um cluster de dispositivos de hardware mais simples e fazer uso do processamento em paralelo.

6. **Diagnostic improvements** – Foram melhorados o acesso às informações sobre a execução de comandos e o desempenho do SGBD. Podem ser listados como avanços no diagnóstico o esquema de desempenho (um recurso para execução em baixo nível no servidor MySQL utilizado para monitoramento), os testes de *DTrace*, a expansão da saída do comando SHOW INNODB STATUS, o Debug Sync, e uma nova variável de status.
7. **MySQL Cluster** - MySQL Cluster foi lançado como um produto separado, com o desenvolvimento da versão 7.2 do motor de armazenamento do NDB sendo baseado em MySQL 5.5.
8. **Semisynchronous replication** - Replicação semi síncrona é implementada através de um componente de plug-in opcional.
9. **Partitioning** – Novas opções que podem ser definidas pelo usuário. O particionamento RANGE COLUMNS como extensão do RANGE e o LIST COLUMNS como extensão para o LIST. LIST e RANGE são duas formas de particionamento que já existiam.
10. **SIGNAL e RESIGNAL** – Suporta os comandos SINGNAL e RESIGNAL do padrão SQL. SIGNAL é uma forma de retornar um erro. RESIGNAL repassa as informações sobre uma condição de erro que está disponível durante a execução de um comando. Ele funciona como um tratamento de erro, que recebe a informação sobre o erro e repassa, possivelmente, modificando-a.
11. **Unicode** – Suporte para conjuntos de caracteres (character sets) suplementares para Unicode: utf16, utf32 e utf8mb4.

Mais informações sobre as funcionalidades apresentadas nesta versão podem ser visualizadas em: https://dev.mysql.com/doc/refman/5.5/en/mysql-nutshell.html.

Agora apresentamos as novidades do **MySQL 5.6** que foram anunciadas junto ao seu lançamento em fevereiro de 2013.

1. Melhoria de desempenho do otimizador de consulta
2. Aumento do *throughput* para transações que utilizam o *InnoDB*
3. Novas APIs para utilização dos conceitos de NoSQL com cache de memória
4. Melhorias nas partições usadas para consulta e gerenciamento de tabelas muito grandes (*very large tables*)
5. O tipo *TIMESTAMP* em uma coluna armazena corretamente os milissegundos.
6. Melhorias no processo de replicação
7. Um monitoramento de desempenho melhorou pela expansão na quantidade de dados disponíveis no PERFORMANCE_SCHEMA
8. O *InnoDB* agora suporta busca em **full-text** e ganhou um avanço na performance do *commit* em grupos
9. Melhoria nas partições de tabelas

A seguir apresentamos uma lista das principais novas funcionalidades do **MySQL 5.6**, agrupadas de forma lógica:

Agora apresentamos uma parte do rol de novidades da versão do **MySQL 5.7**:

1. **Melhorias na segurança –** Um conjunto de novas funcionalidades de segurança foi apresentado. Por exemplo, (1) uma política de controle para o tempo de vida das senhas, o DBA pode definir o tempo que uma senha expira; (2) servidores MySQL instalados utilizando mysql_install_db agora estão seguros por padrão. O processo de instalação cria apenas uma única conta root e não mais contas anônimas.
2. **SQL mode changes -** Modo SQL estrito para mecanismos de armazenamento transacionais (STRICT_TRANS_TABLES) agora está ativado por padrão.
3. **Online ALTER TABLE** – O comando ALTER TABLE agora suporta uma cláusula RENAME INDEX que renomeia um índice. A alteração é feita “no lugar”, sem necessidade de uma operação de cópia da tabela.
4. **Ngram e MeCab full-text parser plug-ins** - A partir do MySQL 5.7.6, o MySQL fornece nativamente um plug-in full-text Ngram parser que suporta chinês, japonês e coreano (CJK), e o MeCab, um plug-in instalável de full-text parser para japonês. Agora você não precisa mais se preocupar quando quiser demonstrar seu conhecimento nas línguas asiáticas. ☺
5. **Melhorias no InnoDB** – Algumas das melhorias que podemos listar para o InnoDB nesta versão do MySQL: (1) A tabela temporária do InnoDB para metadados não é mais armazenada nas tabelas do sistema do InnoDB; (2) o desempenho de comandos DDL para as tabelas temporárias do InnoDB foi melhorado por meio da otimização do CREATE TABLE, DROP TABLE, TRUNCATE TABLE e ALTER TABLE; (3) InnoDB agora suporta MySQL com tipos de dados espaciais.
6. **Suporte a JSON** - A partir do MySQL 5.7.8, o MySQL suporta o tipo JSON nativo. Valores de variáveis JSON não são armazenados como strings, ele usa um formato binário interno que permite acesso rápido de leitura aos elementos do documento.

7. **Variáveis de sistema e status** - Informações de sistema e variáveis de status estão agora disponíveis em tabelas no PERFORMANCE_SCHEMA, ao invés de usar as tabelas do INFORMATION_SCHEMA para obter essas variáveis.

Por fim, apresentamos as principais novidades da versão do **MySQL 8.0**:

1. **Data dictionary** - O MySQL agora incorpora um dicionário de dados transacionais que armazena informações sobre objetos de banco de dados.
2. **Atomic Data Definition Statements (Atomic DDL)** - Uma instrução DDL atômica combina atualizações de dicionário de dados, operações do mecanismo de armazenamento e gravações de log binário associadas a uma operação DDL em uma única transação atômica.
3. **Security and account management** - aprimoramentos foram adicionados para melhorar a segurança e permitir maior flexibilidade de DBA no gerenciamento de contas.
4. **Resource management** - o MySQL agora suporta a criação e o gerenciamento de grupos de recursos, e permite a atribuição de encadeamentos em execução no servidor a grupos específicos para que os encadeamentos sejam executados de acordo com os recursos disponíveis para o grupo.
5. Aprimoramentos de InnoDB e JSON foram adicionados
6. **Character set support** - o conjunto de caracteres padrão foi alterado de latin1 para utf8mb4. O conjunto de caracteres utf8mb4 possui vários novos agrupamentos, incluindo utf8mb4_ja_0900_as_cs, o primeiro agrupamento específico de idioma japonês disponível para Unicode no MySQL.
7. **Data type support** - o MySQL agora suporta o uso de expressões como valores padrão em especificação de tipo de dados. Isso inclui o uso de expressões como valores padrão para BLOB, TEXT, GEOMETRY e tipos de dados JSON, que anteriormente não podiam ser atribuídos a valores padrão.
8. **Aprimoramentos no Optimizer** - o MySQL agora suporta índices invisíveis, descendentes e, também, a criação de partes de chave de índice funcional que indexam valores de expressão em vez de valores de coluna.
9. **Replication** - o MySQL Replication agora suporta log binário de atualizações parciais para documentos JSON usando um formato binário compacto, economizando espaço nos documentos JSON completos de log.
10. **Connection management** – o MySQL Server agora permite que uma porta TCP / IP seja configurada especificamente para conexões administrativas.
11. **Plugins** - anteriormente, os plugins do MySQL podiam ser escritos em C ou C ++. Arquivos de cabeçalho do MySQL usados por plugins agora contêm código C ++, o que significa que plugins devem ser escritos em C ++, não C.

Vamos agora fazer mais uma questão sobre o assunto!

3. CESPE - Analista Judiciário (STM)/Apoio Especializado/Análise de Sistemas/2018

Julgue o item que se segue, a respeito do processamento de transações e otimização de desempenho do SGBD e de consultas SQL.

No MySQL 5.6, o banco de dados information_schema guarda dados estatísticos e eventos para serem utilizados caso se queira encontrar problemas de velocidade de acesso aos dados e(ou) problemas de integridades no SGBD.

Certo

Errado

Comentários: No MySQL, o INFORMATION_SCHEMA fornece acesso a metadados de banco de dados, informações sobre o servidor MySQL, como o nome de um banco de dados ou tabela, o tipo de dados de uma coluna ou os privilégios de acesso. Outros termos que às vezes são usados para esta informação são o dicionário de dados e o catálogo do sistema.

O INFORMATION_SCHEMA é um banco de dados dentro de cada instância do MySQL. Trata-se do local que armazena informações sobre todos os outros bancos de dados que o servidor MySQL mantém. O banco de dados INFORMATION_SCHEMA contém várias tabelas somente leitura. Elas são, na verdade, visualizações, não tabelas de base, portanto, não há arquivos associados a elas, e você não pode definir gatilhos ou triggers. Além disso, não há um diretório de banco de dados com esse nome. As tabelas são construídas temporariamente, enquanto a instância do SGBD estiver executando.

Existe outro esquema padrão no MySQL denominado PERFORMANCE_SCHEMA, que é usado para inspecionar a execução de consultas e eventos em tempo real. Esse esquema destina-se a fornecer acesso a informações úteis sobre a execução do servidor, tendo um impacto mínimo no desempenho. O PERFORMANCE_SCHEMA centra-se principalmente nos dados de desempenho. Isso difere de INFORMATION_SCHEMA, que serve para inspeção dos metadados. Sendo assim, a afirmação mistura os dois conceitos. Logo, está incorreta.

Gabarito: E

4. BANCA: FGV ANO: 2013 ÓRGÃO: AL-MT PROVA: ANALISTA DE SISTEMAS - BANCO DE DADOS

A avaliação do desempenho no SGBD MySQL, versão 5.5 ou superior, pode ser obtida através de consultas executadas no

A performance_schema.

B monitoring_schema.

C evaluation_schema.

D log_schema.

E metadata_schema.

Comentário: O MySQL performance_schema é um recurso para monitorar a execução do servidor MySQL em baixo nível. Seguem algumas das suas características:

Ele fornece uma maneira de inspecionar a execução interna do servidor em tempo de execução. É implementado utilizando o mecanismo de armazenamento **PERFORMANCE_SCHEMA** e o banco de dados **performance_schema**. Ele se concentra principalmente em dados de desempenho. Isso difere do INFORMATION_SCHEMA, que serve para a inspeção dos metadados.

O **PERFORMANCE_SCHEMA** monitora eventos do servidor. Um "evento" é qualquer ação que o servidor faz que dura algum tempo, e foi instrumentado de modo que a informação sobre o tempo de execução possa ser coletada. Em geral, um evento pode ser: uma chamada de função, uma espera pelo sistema operacional, uma etapa de execução de uma instrução SQL, como *parsing* ou ordenação, ou uma declaração inteira ou um grupo de instruções. Atualmente, a coleta de eventos fornece acesso a informações sobre as chamadas de sincronização de arquivo e tabelas de I/O, de Lock, e assim por diante para o servidor e para os vários mecanismos de armazenamento.

Eventos armazenados pelo esquema de desempenho são distintos dos eventos gravados no log binário do servidor (**que descrevem as modificações nos dados**) e dos eventos do *Event Scheduler* (que é um tipo de programa armazenado executado em um determinado instante).

Gabarito: A

Instalação MySQL 8.0 Community Edition

O MySQL Community Edition é a versão de download gratuito do banco de dados de código aberto mais popular do mundo. Está disponível sob a licença GPL e é suportado por uma comunidade enorme e ativa de desenvolvedores de código aberto.

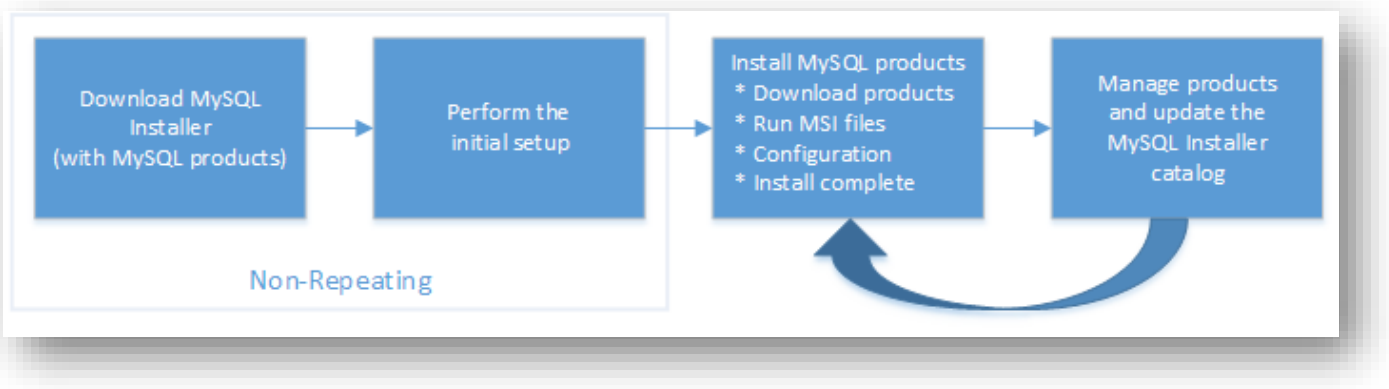
Os arquivos para download estão disponíveis em:

https://www.mysql.com/downloads/

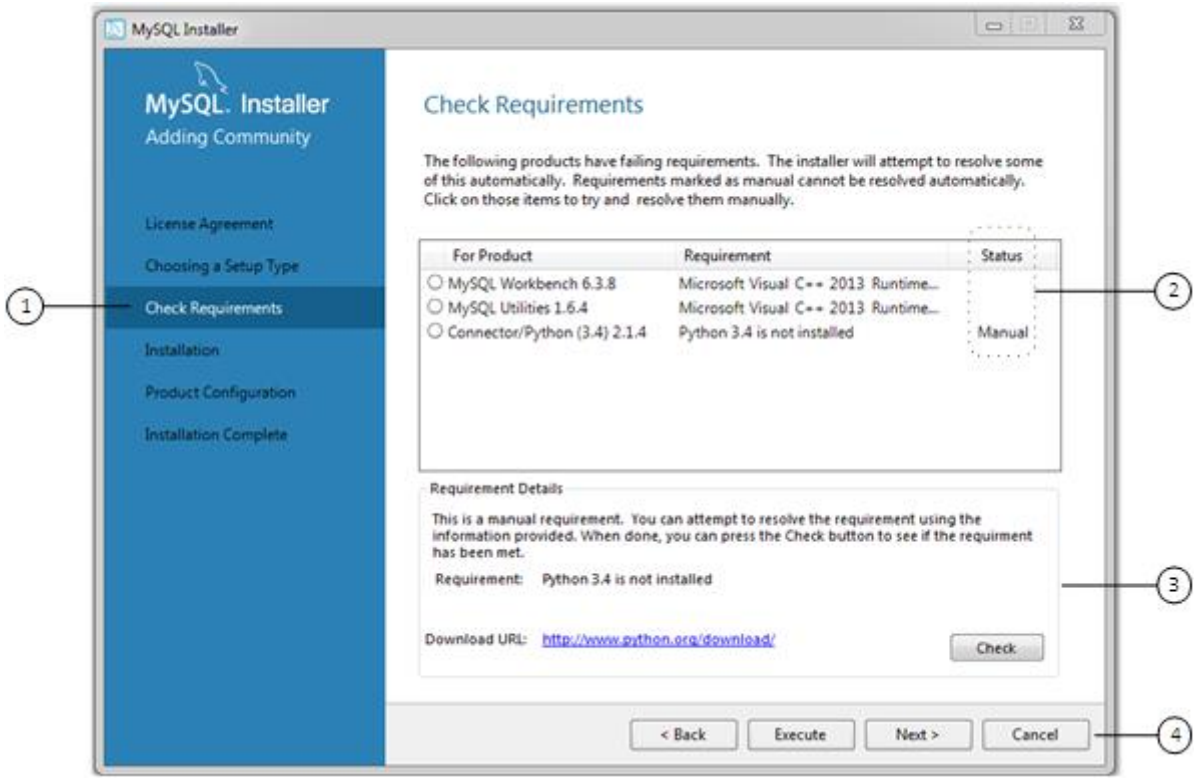
Para instalação, baixe o MySQL Installer do Community Edition (GPL).

O MySQL Installer fornece uma experiência de instalação fácil de usar, baseada em assistente, para todas as suas necessidades de software MySQL. Incluído no produto estão as versões mais recentes de: MySQL Server, MySQL Connectors, MySQL Workbench and sample models, Sample Databases, MySQL for Excel, MySQL Notifier, MySQL for Visual Studio, Documentation.

Confira a Visão Geral do Processo do MySQL Installer:



Veja a tela principal do MySQL Installer:



Segue a descrição dos elementos de verificação de requisitos:

1 - Mostra o passo atual na configuração inicial. As etapas nesta lista podem mudar um pouco dependendo dos produtos já instalados no host, da disponibilidade do software de pré-requisito e dos produtos a serem instalados no host.

2 - Lista todos os requisitos de instalação pendentes por produto e indica o seu status.

3 - Descreve o requisito em detalhes para ajudá-lo em cada resolução manual. Quando possível, um URL de download é fornecido. Depois de baixar e instalar o software necessário, clique em Verificar para verificar se o requisito foi atendido.

4 - Fornece as seguintes operações para continuar a instalação:

Voltar - Retornar ao passo anterior. Esta ação permite selecionar um tipo de configuração diferente.

Executar - O MySQL Installer tenta baixar e instalar o software necessário para todos os itens sem um status manual. Os requisitos manuais são resolvidos por você e verificados clicando em Verificar.

Próximo - Não execute a solicitação para aplicar os requisitos automaticamente e prossiga para a instalação sem incluir os produtos que falharam na etapa de verificação de requisitos.

Cancelar - Pare a instalação de produtos MySQL. Como o MySQL Installer já está instalado, a configuração inicial começa novamente quando você abre o MySQL Installer a partir do menu Iniciar e clica em Adicionar no painel.

ARQUIVOS DE CONFIGURAÇÃO

Arquivos de Configuração no Windows

File Name	Purpose
%WINDIR%\my.ini, %WINDIR%\my.cnf	Global options
C:\my.ini, C:\my.cnf	Global options
BASEDIR\my.ini, *BASEDIR*\my.cnf	Global options
defaults-extra-file	The file specified with --defaults-extra-file, if any
%APPDATA%\MySQL\.mylogin.cnf	Login path options (clients only)
DATADIR\mysqld-auto.cnf	System variables persisted with SET PERSIST or SET PERSIST_ONLY (server only)

Arquivos de Configuração no Unix (Unix-Like)

File Name	Purpose
/etc/my.cnf	Global options
/etc/mysql/my.cnf	Global options
SYSCONFDIR/my.cnf	Global options
$MYSQL_HOME/my.cnf	Server-specific options (server only)
defaults-extra-file	The file specified with --defaults-extra-file, if any
~/.my.cnf	User-specific options
~/.mylogin.cnf	User-specific login path options (clients only)
DATADIR/mysqld-auto.cnf	System variables persisted with SET PERSIST or SE PERSIST_ONLY (server only)

MY.CNF (MY.INI nos Sistemas Operacionais Windows)

Localize a seção `[mysqld]` no arquivo e adicione ou modifique os seguintes parâmetros:

- Especifique a definição padrão de caracteres como UTF-8:

```
[mysqld]
...
character-set-server=utf8
collation-server=utf8_bin
...
```

- Configure o mecanismo de armazenamento padrão para InnoDB:

```
[mysqld]
...
default-storage-engine=INNODB
...
```

- Especifique o valor de `max_allowed_packet` como no mínimo 256M:

```
[mysqld]
...
max_allowed_packet=256M
...
```

- Especifique o valor de `innodb_log_file_size` como no mínimo 2 GB:

```
[mysqld]
...
innodb_log_file_size=2GB
...
```

- Certifique-se de que o parâmetro sql_mode não especifique NO_AUTO_VALUE_ON_ZERO

```
// remove this if it exists
sql_mode = NO_AUTO_VALUE_ON_ZERO
```

EXECUTANDO MYSQL PELA LINHA DE COMANDO

São várias as possibilidades de usar linha de comando para fazer alguma ação sobre o banco de dados MySQL. Apresentaremos abaixo alguns programas cliente que permitem a execução de comandos. Eles são determinantes para uma boa utilização e gerenciamento de banco de dados MySQL. Passaremos por quatro deles: **mysql, mysqldump, mysqladmin e mysqlshow**.

PROGRAMA CLIENTE MYSQL

O **mysql** é uma shell simples para SQL com capacidade de edição de linhas de entrada. Suporta uso interativo e não interativo. Quando usado interativamente, os resultados da consulta são apresentados em um formato de tabela ASCII. Quando utilizado não interativamente (por exemplo, como um filtro), o resultado é apresentado em formato separado por tab. O formato de saída pode ser alterado utilizando opções de comandos.

Se você tiver problemas devido à memória insuficiente para grandes conjuntos de resultados, use a opção –quick ou --q. Este parâmetro mysql força a recuperação dos resultados uma linha de cada vez, em vez de recuperar todo o resultado e colocar em buffer na memória antes de exibi-lo.

Usando a API C descrita pelo MySQL, é possível executar chamada a algumas funções. Uma função da API C na biblioteca cliente/servidor que deve ser usada quando estamos trabalhando com o parâmetro –quick é o mysql_use_result().

Depois de invocar mysql_query() ou mysql_real_query(), você deve chamar mysql_store_result () ou mysql_use_result () para cada declaração que produz um conjunto de resultados (SELECT, SHOW, DESCRIBE, EXPLAIN, CHECK TABLE e assim por diante). Você também deve chamar mysql_free_result() após você ter terminado de trabalhar com o conjunto de resultados.

O mysql_use_result() inicia uma recuperação do resultado definido, mas não lê realmente o conjunto de resultados para o cliente como mysql_store_result() faz. Em vez disso, cada linha deve ser recuperada individualmente fazendo chamadas a mysql_fetch_row(). Ele lê o resultado de uma consulta diretamente do servidor sem armazená-lo em uma tabela temporária ou em um buffer local. Logo, é um pouco mais rápido e usa menos memória que mysql_store_result (). O cliente atribui memória somente para a linha atual e um buffer de comunicação, que pode crescer até o parâmetro **max_allowed_packet** definido em bytes.

Utilizar mysql é muito fácil. Chame a partir do prompt do seu interpretador de comandos da seguinte forma:

shell> mysql nome_bd

A lista a seguir resume as opções que podem ser usados para controlar como os programas cliente se conectam ao servidor:

--host = host_name, -h host_name - O host onde o servidor está executando. O valor padrão é **localhost**.

--password [= pass_val], -P [pass_val] - A senha da conta MySQL. O valor da senha é opcional, mas se for dada, não deve haver nenhum espaço entre -p ou --password = e a senha que se lhe segue. O padrão é não enviar nenhuma senha.

--port = port_num, -P port_num - O número da porta a ser usado para a conexão. Para conexões feitas usando TCP / IP, o número da porta **padrão é 3306**.

--protocol = {TCP | SOCKET | TUBOS | MEMORY} - Esta opção especifica explicitamente um protocolo a ser usado para conexão com o servidor. É útil quando os outros parâmetros de conexão normalmente levariam a um protocolo a ser usado que não seja o que você quer. Por exemplo,

conexões em Unix para localhost são feitas utilizando um arquivo socket Unix por padrão. Para forçar uma conexão TCP/IP, deve ser usado a opção --protocol:

shell> mysql --host = localhost --protocol = TCP

--ssl * - Opções que começam com --ssl são usados para estabelecer uma conexão segura ao servidor usando SSL, se o servidor está configurado com suporte SSL.

--user = user_name, -u nome_usuário - O nome de usuário da conta do MySQL que você deseja usar. O nome de usuário padrão é ODBC no Windows ou seu nome de login no Unix.

MYSQLDUMP, MYSQLADMIN, OR MYSQLSHOW

O utilitário cliente **mysqldump** executa backups lógicos, produzindo um conjunto de instruções SQL que podem ser executados para reproduzir as definições de objeto do banco de dados original e dos dados da tabela. Ele despeja (dump) um ou mais bancos de dados MySQL para backup ou para transferência para outro servidor SQL. O comando mysqldump também pode gerar sua saída em formato CSV, em forma de outro texto delimitado ou em formato XML.

Para executar o **mysqldump**, é requerido pelo menos o privilégio SELECT para as tabelas de dumping, o de SHOW VIEW para as visões de dumping, o de TRIGGER para gatilhos de dumping e o de LOCK TABLES, se a opção de transação --single-transaction não é usada. Algumas opções podem exigir outros privilégios.

Para recarregar um arquivo de despejo, você deve ter os privilégios necessários para executar as instruções que ele contém. A ideia é possuir os privilégios apropriados para os objetos criados por essas declarações ou esses comandos.

A saída do mysqldump pode incluir instruções ALTER DATABASE que alterara a *collations* ou o formato de codificação do banco de dados. Estes podem ser utilizados quando o dumping de programas armazenados necessita preservar suas codificações de caracteres. Para recarregar um arquivo de despejo que contenha tais declarações, é necessário o privilégio ALTER para o banco de dados afetado.

(Ministério da Economia – Infraestrutura - 2020) A respeito de MySQL, julgue o item subsequente.

101 O comando a seguir permite a execução de um backup de todas as bases de dados, que serão armazenados em um arquivo de nome backup.sql.

mysqldump --all-databases > backup.sql

_____

Comentários:

Utilizando a ferramenta ***mysqldump***, você poderá realizar o backup de uma base de dados local e restaurá-la para uma base de dados remota com as mesmas características, tudo isso utilizando um simples comando.

Em geral, existem três maneiras de usar o mysqldump - para despejar um conjunto de uma ou mais tabelas (1), um conjunto de um ou mais bancos de dados completos (2) ou um servidor MySQL inteiro (3) - conforme mostrado aqui:

(1) shell> mysqldump [options] db_name [tbl_name ...]
(2) shell> mysqldump [options] --databases db_name ...
(3) shell> mysqldump [options] --all-databases

Logo, temos uma alternativa correta.

Gabarito Certo.

3. CESPE - Analista de Tecnologia da Informação (FUB)/2015

Com relação ao banco de dados MySQL, julgue o item subsequente.

O utilitário mysqldump permite extrair dados online, isto é, sem parar o funcionamento do banco de dados, a fim de que sirvam de becape.

Certo

Errado

Comentários: Para tabelas InnoDB, o mysqldump fornece uma maneira de fazer um backup online:

shell> mysqldump --all-databases --master-data --single-transaction > all_databases.sql

Este backup adquire um bloqueio de leitura global em todas as tabelas (usando FLUSH TABLES WITH READ LOCK) no início do dump. Assim que esse bloqueio tiver sido adquirido, as coordenadas do log binário serão lidas e o bloqueio será liberado. Se longas instruções de atualização estiverem em execução quando a instrução FLUSH for emitida, o servidor MySQL pode ficar paralisado até que essas instruções sejam concluídas. Depois disso, o dump se torna livre de bloqueio e não atrapalha as leituras e as gravações nas tabelas. Se as instruções de atualização que o servidor MySQL recebe são curtas (em termos de tempo de execução), o período de bloqueio inicial não deve ser perceptível, mesmo com muitas atualizações.

Gabarito: C

---

4. BANCA: CESPE ANO: 2013 ÓRGÃO: FUB PROVA: ANALISTA - TECNOLOGIA DA INFORMAÇÃO

A respeito de banco de dados, julgue os próximos itens.

[1] O comando mysqldump permite o backup de bancos de dados MySQL, possibilitando que os dados possam ser migrados para outras versões de banco de dados MySQL. O uso da opção –q junto a esse comando descarrega os dados diretamente para a saída especificada, sem utilizar buffer de memória.

Comentário: Da mesma forma que no mysql, o mysqldump também tem a opção de usar o parâmetro --quick ou -q.

Essa opção é útil para fazer o dump de grandes tabelas. Obriga o mysqldump a recuperar linhas de uma tabela a partir do servidor, uma linha de cada vez, em vez de recuperar todas as linhas e colocar em buffer na memória antes de escrevê-lo para fora do servidor na saída especificada.

Gabarito: C

O cliente **mysqlshow** pode ser usado para ver rapidamente quais bancos de dados existem, suas tabelas, suas colunas ou índices de uma tabela. O mysqlshow fornece uma interface de linha de comando. Para invocar o mysqlshow, podemos executar o comando:

shell> mysqlshow [options] [db_name [tbl_name [col_name]]]

Se nenhum banco de dados é passado como parâmetro, é mostrada uma lista de nomes de banco de dados. Se nenhuma tabela é fornecida, todas as tabelas encontradas no banco de dados são mostradas. Se nenhuma coluna for fornecida, todas as colunas e tipos de colunas da tabela são mostrados.

A saída exibe **apenas os nomes** dessas bases de dados, tabelas ou colunas que você tem alguns **privilégios de acesso**.

O **mysqladmin** é um cliente para realizar operações administrativas. Você pode usá-lo para verificar a configuração atual e o status do servidor, para criar e eliminar bancos de dados e muito mais. Invocar o mysqladmin pode ser feito da seguinte forma:

shell> mysqladmin [options] command [command-arg] [command [command-arg]]

O mysqladmin suporta vários comandos. Alguns dos comandos aceitam um argumento após o nome do comando. Vejamos alguns exemplos:

CREATE db_name - Criar um banco de dados chamado db_name.

DEBUG - Diz ao servidor para escrever informações de depuração no log de erro. O formato e o conteúdo dessas informações estão sujeitos a alterações.

DROP db_name - Excluir o banco de dados chamado db_name e todas as suas tabelas.

EXTENDED-STATUS – Usado para visualizar as variáveis de estado de servidor e seus valores.

flush-hosts - Captura todas as informações no cache do host.

5. BANCA: FGV ANO: 2013 ÓRGÃO: AL-MT PROVA: ANALISTA DE SISTEMAS - BANCO DE DADOS

Leia o fragmento a seguir.

"O programa _____ e o script _____ podem ser utilizados para fazer cópias de segurança no SGBD MySQL. O primeiro é mais geral porque pode ser utilizado em todos os tipos de tabela, o segundo somente funciona em alguns tipos de storage engines, como por exemplo o _____."

Assinale a alternativa cujos itens completam corretamente as lacunas do fragmento acima.

A mysqlhotcopy – mysqldump – InnoDB.

B mysqldump – mysqlhotcopy – MyISAM.

C mysqldump – mysqlcopy – MyISAM.

D mysqlbkp – mysqlrestore – InnoDB.

E mysqlbkp – mysqlhotcopy – HEAP/Memory.

Comentário: A questão trata, na primeira lacuna, de um programa usado para fazer backup. Sabemos que a resposta que se adequa perfeitamente ao contexto seria o uso do programa cliente **mysqldump**. A segunda lacuna trata de um script. Neste caso, precisamos abrir um parêntese para falar do **mysqlhotcopy**.

O **mysqlhotcopy** é um script Perl que foi originalmente escrito por Tim Bunce. Ele usa FLUSH TABLES, LOCK TABLES, e **cp** ou **scp** para fazer um backup de banco de dados. É uma maneira rápida para fazer um backup do banco de dados completo ou de apenas algumas tabelas, mas pode ser executado somente na mesma máquina onde os diretórios de banco de dados estão localizados. O mysqlhotcopy funciona apenas para fazer backup de tabelas **MyISAM** e **ARCHIVE**. Ele pode ser executado em Unix e NetWare.

Para utilizar mysqlhotcopy, você deve ter acesso de leitura aos arquivos para as tabelas que você está fazendo backup. Deve ter o privilégio SELECT para essas tabelas, o privilégio RELOAD (para ser capaz de executar FLUSH TABLES) e o privilégio LOCK TABLES (para ser capaz para bloquear as tabelas). Veja a sintaxe usada para execução do script:

shell> mysqlhotcopy db_name [/path /to /new_directory]

shell> mysqlhotcopy db_name_1 ... db_name_n /path /to /new_directory

É possível ainda fazer backup de tabelas no banco de dados que correspondem a uma expressão regular determinada:

shell> mysqlhotcopy db_name ./ regex /

Pelo exposto acima, já temos condições de marcar a alternativa correta. Sabemos que o script seria o mysqlhotcopy e que o mesmo só consegue fazer backup de tabelas MyISAM e ARCHIVE. A resposta está na alternativa B.

Gabarito: B

Dica do professor

Após executar o comando de mysqld para iniciar o servidor MySQL, você pode usar alguns comandamos para interagir com a base. Veja a lista abaixo:

USE DatabaseName: Será usado para selecionar um determinado banco de dados MySQL na área de trabalho.

SHOW DATABASES: Lista os bancos de dados que podem ser acessados pelo SGBD MySQL.

SHOW TABLES: Mostra as tabelas no banco de dados uma vez que o banco de dados foi selecionado com o comando USE.

SHOW COLUMNS FROM tablename: Mostra os atributos, os tipos de atributos, a informação chave, se null é permitido, os padrões e outras informações para uma tabela.
SHOW INDEX FROM tablename: Apresenta os detalhes de todos os índices da tabela, incluindo a chave primária.
SHOW TABLE STATUS LIKE tablename \G: Mostra detalhes do desempenho e estatísticas do SGBD MySQL.

STORAGE INNODB E MYISAM

Os dois **motores de armazenamentos** mais comumente usados na maioria dos servidores MySQL são InnoDB e MyISAM. Esta parte da aula é dedicada a cobrir brevemente os dois tipos e identificar quais os mais recomendados e em que circunstâncias. Vejamos, portanto, as suas principais diferenças.

QUAIS AS PRINCIPAIS DIFERENÇAS

MyISAM	InnoDB
Não compatível com *ACID e não transacional	Compatível com *ACID e, portanto, transação integral com ROLLBACK e COMMIT; e suporta Foreign Key
Motor padrão MySQL 5.0	Mecanismo padrão da nuvem rackspace
Oferece compressão	Oferece compressão
Requer reparo / reconstrução completa de índices / tabelas	Auto recuperação de cash via replay de logs
Alterações de páginas DB gravadas no disco instantaneamente	Páginas sujas convertidas de aleatório para sequencial antes de COMMIT e liberação para o disco
Sem pedido no armazenamento de dados	Dados de linha armazenados em páginas em ordem PK
Bloqueio de nível de tabela	Bloqueio de nível de linha

Podemos observar na tabela acima as principais diferenças entre as duas storage engines. Vejam que MyISAM se caracteriza por uma abordagem não transacional, ou seja, não segue as características de atomicidade, consistência, isolamento e durabilidade (ACID). Ele era padrão até a versão 5.0 do MySQL.

O InnoDB, por sua vez, é muito mais robusto que o MyISAM. Admite ROLLBACK e COMMIT, e suporta chaves estrangeiras. Segue, portanto, os princípios definidos pelo ACID. Atualmente, é o padrão de storage do MySQL. Seu *lock* é feito em nível de linhas.

De uma forma prática, você deve usar MyISAM quando sua aplicação necessitar de busca em texto completo (full text search) ou quando tenha a grande maioria das operações feitas sobre a base do tipo SELECT.

Em relação ao InnoDB, ele é mais indicado quando requer o uso de transações ou as operações realizadas com maior frequência na base sejam de modificação (INSERT, UPDATE ou DELETE). Deve ser considerado também quando o modelo relacional estiver sendo utilizado e, ainda, quando for necessário o uso de *lock* em nível de linha.

ÍNDICES FULL-TEXT (CONFIGURAÇÃO E MÉTODOS DE PESQUISA)

A palavra full-text apareceu diversas vezes na aula até o momento. Durante a evolução das funcionalidades do SGBD MySQL, a capacidade de busca e indexação sobre tipos que representam conjuntos de caracteres foi introduzida. O suporte deve ser feito por meio da criação de um índice do tipo FULLTEXT. Para entendermos melhor o funcionamento deste comando, vamos partir de um exemplo.

Primeiramente, vamos definir uma tabela chamada de articles que possui os atributos title e body, respectivamente, do tipo VARCHAR (200) e TEXT. Em seguida, definimos um índice utilizando a palavra-chave FULLTEXT () e passando como parâmetro os atributos que orientam a criação dos índices. Por fim, estabelecemos a ENGINE para InnoBD. Lembrando que a utilização deste tipo de índice só pode ser feita com InnoDB e MyISAM.

Criada a tabela, vamos inserir algumas tuplas dentro dela para que possamos fazer testes. No exemplo, seis linhas são inseridas. Em seguida, construímos o comando SELECT que vai fazer uso do índice, com a cláusula MATCH (coluna1, coluna2, ...) seguida pela sintaxe AGAINST (expressão [método de pesquisa]). No exemplo, utilizamos as colunas title e body e faremos a busca pela expressão ‘database’. Utilizamos o método de pesquisa de linguagem natural (IN NATURAL LANGUAGE MODE). Observe o exemplo e tente entender a estrutura básica para a utilização do índice FULLTEXT.

```
mysql> CREATE TABLE articles (
       id INT UNSIGNED AUTO_INCREMENT NOT NULL PRIMARY KEY,
       title VARCHAR(200),
       body TEXT,
       FULLTEXT (title,body)
     ) ENGINE=InnoDB;
Query OK, 0 rows affected (0.00 sec)

mysql> INSERT INTO articles (title,body) VALUES
     ('MySQL Tutorial','DBMS stands for DataBase ...'),
     ('How To Use MySQL Well','After you went through a ...'),
     ('Optimizing MySQL','In this tutorial we will show ...'),
     ('1001 MySQL Tricks','1. Never run mysqld as root. 2. ...'),
     ('MySQL vs. YourSQL','In the following database comparison ...'),
     ('MySQL Security','When configured properly, MySQL ...');
Query OK, 6 rows affected (0.00 sec)
Records: 6  Duplicates: 0  Warnings: 0

mysql> SELECT * FROM articles
     WHERE MATCH (title,body)
     AGAINST ('database' IN NATURAL LANGUAGE MODE);
+----+-------------------+------------------------------------------+
| id | title             | body                                     |
+----+-------------------+------------------------------------------+
|  1 | MySQL Tutorial    | DBMS stands for DataBase ...             |
|  5 | MySQL vs. YourSQL | In the following database comparison ... |
+----+-------------------+------------------------------------------+
2 rows in set (0.00 sec)
```

Agora já sabemos que o MySQL suporta a indexação e busca do tipo full-text ou textual. Um índice textual pode ser criado sobre colunas dos tipos CHAR, VARCHAR ou TEXT. Comentamos anteriormente que o parser ngram dá suporte aos caracteres do chinês, japonês e coreano (CJK), e o parser MeCab é um plugin instalável que dá suporte aos caracteres japoneses.

A definição de um índice do tipo FULLTEXT pode ser feita na criação da tabela por meio do comando CREATE TABLE ou adicionada posteriormente utilizando o comando ALTER TABLE ou CREATE INDEX. Em grandes bancos de dados, é mais rápido carregar os dados dentro de uma tabela que não possui o índice definido e, em seguida, executar o comando para criação do índice.

Como vimos acima, no nosso exemplo, a execução de uma busca é feita usando a sintaxe: MATCH (...) AGAINST. MACTH estabelece uma lista de colunas separadas por vírgula sobre as quais será executada a busca. AGAINST recebe a string que estamos procurando e um modificador que define qual o método de busca será utilizado. Essa string é um valor constante durante a execução da consulta. Por padrão, o resultado da busca via FULLTEXT é ordenado conforme um ranking próprio de relevância.

Existem três tipos ou métodos de busca full-text no MySQL.

1. **Pesquisa de linguagem natural** interpreta a sequência de pesquisa como uma frase em linguagem humana natural (frase em texto livre). Não há operadores especiais. Uma lista de palavras de parada (stopword) pode ser aplicada. Esse tipo de pesquisa é utilizado se o modificador **IN NATURAL LANGUAGE MODE** for definido ou nenhum modificador for estabelecido.
2. **Pesquisa booleana** interpreta a cadeia de pesquisa utilizando as regras de uma linguagem de consulta especial. A cadeia de caracteres contém as palavras utilizadas na pesquisa. Ela também pode conter operadores que especificam os requisitos da forma com que uma palavra deve estar presente ou ausente nas linhas correspondentes ou se deve ter alguma ponderação maior ou menor do que o habitual. Certas palavras (palavras de parada) são omitidas do índice de pesquisa e não são consideradas se presente na sequência de pesquisa. O modificador **IN BOOLEAN MODE** especifica uma pesquisa booleana.
3. **Pesquisa de expansão de consulta** é uma modificação de uma pesquisa de linguagem natural. A cadeia de caracteres procurada é usada para executar uma pesquisa de linguagem natural. Em seguida, as palavras das linhas mais relevantes retornadas pela pesquisa são adicionadas à cadeia de pesquisa e a pesquisa é feita novamente. A consulta retorna as linhas da segunda pesquisa. Os modificadores IN NATURAL LANGUAGE MODE WITH QUERY EXPANSION ou WITH QUERY EXPANSION especificam uma pesquisa de expansão de consulta.

6. CESPE - Analista Judiciário (STM)/Apoio Especializado/Análise de Sistemas/2018

Julgue o item que se segue, a respeito do processamento de transações e otimização de desempenho do SGBD e de consultas SQL.

No MySQL 5.6, os índices são usados para, entre outras operações, desconsiderar linhas a serem pesquisadas e(ou) encontrar linhas abrangidas pelo WHERE mais rapidamente.

Certo

Errado

Comentários: O MySQL usa índices de várias maneiras:

Os índices são usados para acelerar pesquisas de linhas que combinam termos de uma cláusula WHERE ou linhas que combinam tuplas de outras tabelas quando executam junções. Por isso, a alternativa está **correta**.

Para consultas que usam as funções MIN() ou MAX(), o MySQL pode encontrar o menor ou maior valor em uma coluna indexada rapidamente sem examinar cada linha.

O MySQL em geral pode usar índices para executar eficientemente operações de classificação e agrupamento para cláusulas ORDER BY e GROUP BY.

Gabarito: C

USANDO CRIPTOGRAFIA EM CONEXÕES

Com uma conexão não criptografada entre o cliente MySQL e o servidor, alguém com acesso à rede poderia observar todo o seu tráfego e inspecionar os dados sendo enviados ou recebidos entre o cliente e o servidor.

Quando você precisa mover informações em uma rede de maneira segura, uma conexão não criptografada é inaceitável. Para tornar qualquer tipo de dado ilegível, é necessário usar criptografia. Os algoritmos de criptografia devem incluir elementos de segurança para resistir a muitos tipos de ataques conhecidos.

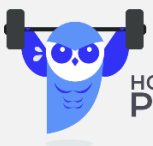
O MySQL oferece suporte a conexões criptografadas entre clientes e o servidor usando o **protocolo TLS (Transport Layer Security).** O TLS usa algoritmos de criptografia para garantir que os dados recebidos em uma rede pública sejam confiáveis. Ele possui mecanismos para detectar alteração, perda ou reprodução de dados. O TLS também incorpora algoritmos que fornecem verificação de identidade usando o padrão X.509.

O X.509 possibilita identificar alguém na Internet. Em termos básicos, deve haver alguma entidade chamada "Autoridade Certificadora" (ou CA) que atribui certificados eletrônicos a qualquer pessoa que deles precise. Os certificados contam com algoritmos de criptografia assimétricos que possuem duas chaves de criptografia (uma chave pública e uma chave secreta).

O proprietário de um certificado pode apresentar o certificado a outra parte como prova de identidade. Um certificado consiste de várias informações incluindo a chave pública de seu proprietário. Todos os dados criptografados com essa chave pública podem ser descriptografados com a chave secreta correspondente, que é mantida pelo proprietário do certificado.

O MySQL executa a criptografia por conexão, e o uso da criptografia para um determinado usuário pode ser opcional ou obrigatório. Isso permite que você escolha uma conexão criptografada ou não criptografada de acordo com os requisitos de aplicativos individuais. Para exigir que os clientes se conectem usando conexões criptografadas, habilite a variável do sistema ***require_secure_transport***.

No lado do servidor, a opção ***--ssl*** especifica que o servidor permite, mas não requer conexões criptografadas. Esta opção é ativada por padrão, portanto, não precisa ser especificada explicitamente.



(Ministério da Economia – Infraestrutura - 2020) A respeito de MySQL, julgue o item subsequente.

100 Para o MySQL suportar conexões seguras com TLS (transport layer security) especificamente e somente na versão 1.2 da TLS, deve-se adicionar, no arquivo my.cnf, a entrada tls_connections=TLSv1.2 e o servidor mysql deve ser reiniciado

Comentários:

.

O MySQL oferece suporte a conexões criptografadas entre clientes e o servidor usando o protocolo TLS (Transport Layer Security). O MySQL oferece suporte a conexões criptografadas usando os protocolos TLSv1, TLSv1.1 e TLSv1.2, listados em ordem do menos seguro para o mais seguro.

No lado do servidor, a opção ***--ssl*** especifica que o servidor permite, mas não requer conexões criptografadas. Esta opção é ativada por padrão, portanto, não precisa ser especificada explicitamente.

Para exigir que os clientes se conectem usando conexões criptografadas, habilite a variável do sistema ***require_secure_transport***.

Gabarito Errado.

Com isso, terminamos o conteúdo teórico relacionado à **administração de banco de dados MySQL**.

CONCEITOS DE DESENVOLVIMENTO EM BANCOS DE DADOS MYSQL

CONSTRUINDO E ACESSANDO UM BANCO DE DADOS

Antes de começar a desenvolver nosso raciocínio sobre os comandos, precisamos entender os tipos de dados e a suas restrições. Vamos então falar um pouco sobre cada um deles!

TIPO DE DADOS

O MySQL suporta uma variedade de tipos de dados que podem ser divididos em algumas categorias: tipos numéricos, tipos de data e tempo, tipo string (pode ser de caracteres ou de bytes) e tipos espacial. Apresentaremos a seguir uma breve descrição de cada um dos tipos disponíveis. Se você estiver interessado em saber um pouco mais de detalhes sobre o assunto, sugiro olhar o capítulo de Data Type da documentação oficial do MySQL.

Dica do professor:

Existem algumas convenções utilizadas pelo MySQL que vale a pena conhecer. Elas estão relacionadas às descrições dos tipos de dados. Vejam abaixo:

- **M** indica a quantidade máxima de dígitos de um **tipo inteiro** que é exibida. Para tipos de ponto flutuante ou de ponto fixo, **M** significa o número total de dígitos que podem ser armazenados. Quando se trata do tipo string, **M** é o comprimento máximo (*length*). Ou seja, o valor máximo permitido para **M** varia de acordo com o tipo de dados.
- **D** é aplicado aos tipos de ponto fixo e flutuante e indica o número de dígitos que compõe as casas decimais (conhecidas com escala). O valor máximo permitido é 30, mas não deve ser maior que **M – 2.**
- **fsp** é aplicado aos tipos TIME, DATETIME e TIMESTAMP e representa a precisão fracionária dos segundos (*fractional seconds precision*). Pode ser visto como o número de casas decimais usadas para representar a fração dos segundos. Deve ser um inteiro entre 0 (nenhuma parte fracionária) e 6. Em caso de omissão será considerado o zero.

Abre e fecha colchetes (**"["** e **"]"**) indica uma parte **opcional** da definição.

TIPOS DE DADOS NUMÉRICOS

Vamos começar tratando dos tipos numéricos disponíveis em MySQL. Abaixo segue uma lista com a sintaxe de cada um deles. Já falamos do significado do parâmetro **M**. Outro conceito importante é o **ZEROFILL** que, quando utilizado, adiciona automaticamente o atributo **UNSIGNED** para determinada coluna.

- **BIT [(M)]** – Um tipo de dados com campos para bits. **M** indica o número de bits, varia entre 1 a 64. O valor default é um, se o valor de **M** for omitido. Lembre-se do nosso comentário, todos os parâmetros entre colchetes são opcionais.
- **TINYINT [(M)] [ZEROFILL] [UNSIGNED]** - Tipo inteiro muito pequeno. O seu range pode variar de -128 a 127, para o caso da variável ou coluna ser definida como **SIGNED**, ou de 0 a 255, ser for inicializada como **UNSIGNED.**
- **BOOL** ou **BOOLEAN -** Este tipo pode ser visto como um sinônimo para TINYINT[1]. O valor zero (0) é considerado **FALSE**. E qualquer valor diferente de zero é considerado **TRUE.** Observe, entretanto, que o valor TRUE e FALSE são meramente alias para um e zero, respectivamente.
- **SMALLINT[(M)] [UNSIGNED] [ZEROFILL]** – Um inteiro com range pequeno. O range SIGNED é de -32768 a 32767. E o UNSIGNED varia de 0 a 65535.
- **MEDIUMINT[(M)] [UNSIGNED] [ZEROFILL] -** Um inteiro com range médio. O range SIGNED é de -8388608 a 8388607. E o UNSIGNED varia de 0 a 16777215.
- **INT[(M)] [UNSIGNED] [ZEROFILL] -** Um inteiro com tamanho padrão (normal). O range SIGNED é de -2147483648 a 2147483647. E o UNSIGNED varia de 0 a 4294967295. Um sinônimo para o tipo INT é o INTERGER, que pode ser usado com os mesmos parâmetros. Se observarmos o parâmetro **SERIAL DEFAULT VALUE** na definição de uma coluna inteira, podemos entender com um alias para **NOT NULL AUTO_INCREMENT UNIQUE.**
- **BIGINT[(M)] [UNSIGNED] [ZEROFILL]** – Um inteiro grande. O parâmetro **SERIAL** na definição de uma coluna é um alias para **BIGINT UNSIGNED NOT NULL AUTO_INCREMENT UNIQUE**. Algumas coisas devem ser consideradas quando estamos usando BIGINT:

1. Todas as operações aritméticas feitas pelo banco são feitas usando valores **SIGNED BIGINT** ou **DOUBLE**. Desta forma, você não deve usar valores para uma variável BIGINT UNSIGNED maiores do que 9223372036854775807 (63 bits) ao menos que você esteja usando funções que operam bit-a-bit.
2. MySQL pode trabalhar com variáveis BIGINT de diferentes formas:

- Quando você quer armazenar valores inteiros grandes, pode usar o UNSIGNED em uma coluna BIGINT.
- Quando você usa as funções MIN(col_name) e MAX(col_name), onde col_name refere-se a uma coluna numérica BIGINT.
- Quando você usa os operadores (+, -, *, e outros) e os dois operandos são **inteiros**.

3. Você sempre pode armazenar valores inteiro exatos em um campo BIGINT, armazenando-o usando uma string. Neste caso, o MySQL realiza a conversão de string-para-número que não envolve representação intermediária utilizando precisão dupla (*double precision)*.

Por fim, abaixo, segue uma lista dos valores máximo e mínimo para cada um dos tipos numéricos tratados acima e a quantidade de bytes utilizados para o armazenamento:

Type	Storage (Bytes)	Minimum Value (Signed/Unsigned)	Maximum Value (Signed/Unsigned)
TINYINT	1	-128	127
		0	255
SMALLINT	2	-32768	32767
		0	65535
MEDIUMINT	3	-8388608	8388607
		0	16777215
INT	4	-2147483648	2147483647
		0	4294967295
BIGINT	8	-9223372036854775808	9223372036854775807
		0	18446744073709551615

Vamos continuar a apresentação dos demais tipos numéricos do MySQL. A partir de agora, os tipos apresentam casas decimais. Vejam que o parâmetro **D**, que definimos anteriormente, começa a fazer parte do contexto. Vamos juntos!

DECIMAL[(M[,D])] [UNSIGNED] [ZEROFILL]

Pode ser visto como um **pacote "exato"** para variáveis de ponto fixo. **M** é o número total de dígitos (**precisão**) e D é o número de dígitos após a vírgula ou ponto decimal (**escala**). O ponto decimal e, para números negativos, o sinal "-" **não são contados em M**. Se D é zero, os valores não têm nenhuma parte decimal ou parte fracionária. O número máximo de dígitos **(M)** para DECIMAL é 65. O número máximo de casas decimais suportadas **(D)** é 30. Se D for omitido, o padrão é zero. Se M for omitido, o padrão é 10. Já sabemos também que, se o UNSIGNED for especificado, ele não permite valores negativos. Por fim, todos os cálculos básicos (+, -, *, /) com colunas decimais são feitas com uma precisão de 65 dígitos.

DEC[(M[,D])] [UNSIGNED] [ZEROFILL]
NUMERIC[(M[,D])] [UNSIGNED] [ZEROFILL]
FIXED[(M[,D])] [UNSIGNED] [ZEROFILL]

Estes tipos são sinônimos para DECIMAL. O tipo **FIXED** está disponível para compatibilidade com outros sistemas de banco de dados.

FLOAT[(M,D)] [UNSIGNED] [ZEROFILL]

É número de ponto flutuante considerado pequeno de precisão simples. Os valores permitidos são de -3.402823466E+38 até -1.175494351E-38, zero, e de 1.175494351E-38 até 3.402823466E+38.

Estes são os limites teóricos, com base na norma IEEE. O intervalo real pode ser ligeiramente menor, dependendo do seu hardware ou sistema operacional.

Analisando agora seus parâmetros, M é o número total de dígitos e D é o número de dígitos após o ponto decimal. Se M e D são omitidos, os valores são armazenados com os limites permitidos pelo hardware. Um número de ponto flutuante de **precisão simples** é preciso para aproximadamente **sete** casas decimais.

Usar FLOAT pode gerar alguns problemas inesperados, porque todos os cálculos em MySQL são feitos com precisão dupla (DOUBLE PRECISION).

DOUBLE[(M,D)] [UNSIGNED] [ZEROFILL]

Considerado o tamanho normal para número de ponto flutuante, usa uma precisão dupla, por isso chamado de DOUBLE. Os valores permitidos são de -1.7976931348623157E+308 até -2.2250738585072014E-308, zero, e de 2.2250738585072014E-308 até 1.7976931348623157E + 308. Mais uma vez estes são os limites teóricos. O intervalo real pode ser ligeiramente menor, dependendo do seu hardware ou sistema operacional.

M é o número total de dígitos e D é o número de dígitos após o ponto decimal. Se os valores M e D forem omitidos, os valores são armazenados com os limites permitidos pelo hardware. Um número de ponto flutuante de precisão dupla tem uma precisão de, aproximadamente, **15** casas decimais.

DOUBLE PRECISION[(M,D)] [UNSIGNED] [ZEROFILL]
REAL[(M,D)] [UNSIGNED] [ZEROFILL]

Estes tipos são sinônimos para DOUBLE. **Exceção:** Se o modo de **REAL_AS_FLOAT** de SQL estiver habilitado, REAL será utilizado como um sinônimo para FLOAT em vez de DOUBLE.

FLOAT(p) [UNSIGNED] [ZEROFILL]

Apresenta outro número de ponto flutuante. O p representa a precisão em bits, mas o MySQL usa esse valor apenas para determinar se deve usar float ou double para o tipo de dados resultante. Se p é de 0 a 24, o tipo de dados torna-se um float sem valores de M ou D. Se p for 25-53, o tipo de dados torna-se DUPLO sem valores M ou D. O range da coluna resultante é o mesmo que o range para os tipos de dados DOUBLE e FLOAT descritos anteriormente. A sintaxe **FLOAT (p)** é fornecida para **compatibilidade ODBC.**

Tipos de Dado de Data e hora

A versão MySQL 5.6.4 ou superior permite frações de segundos para valores de TIME, DATETIME e TIMESTAMP, com microssegundos de até 6 dígitos de precisão. Para definir uma coluna que inclui uma parte da fração de segundos, use a sintaxe **type_name (fsp)**, onde type_name pode ser **TIME**, **DATETIME** ou **TIMESTAMP**, e **fsp** é a precisão fracionária de segundo. Vejam o exemplo abaixo:

```
CRETATE TABLE tempo (t TIME (3), dt DATETIME (6));
```

O valor do fsp, se for determinado, deve estar no intervalo de 0 a 6. Um valor de zero indica que não há nenhuma parte fraccionada dos segundos. Se omitido, a precisão padrão é zero. (Isso difere do padrão SQL, cujo valor default é seis para manter a compatibilidade com versões anteriores do MySQL)

O MySQL 5.6.5 introduz a expansão automática da inicialização e atualização dos tipos temporais. Qualquer coluna TIMESTAMP de uma tabela pode ter estas propriedades, ao invés de no máximo uma coluna por tabela. Além disso, essas propriedades estão agora disponíveis para as colunas DATETIME.

Mas o que seria a expansão automática da inicialização? Para qualquer coluna TIMESTAMP ou DATETIME em uma tabela, você pode atribuir a hora atual como o valor padrão, ou o valor de atualização automática, ou ainda ambos. Uma coluna inicializada automaticamente define o *timestamp* atual para linhas inseridas que não especificam valor para a coluna. Uma coluna auto atualizada é automaticamente atualizada para a hora atual, quando o valor de qualquer outra coluna na linha é alterado.

Falaremos também do tipo de dados YEAR(2), que possui certas questões que você deve considerar antes de optar por usá-lo. A partir do MySQL 5.6.6, YEAR (2) é obsoleto. Colunas YEAR(2) em tabelas existentes são tratados como antes, mas YEAR (2) em tabelas novas ou alteradas são convertidos em YEAR (4). Agora vamos tratar rapidamente de cada um dos tipos de data.

DATE

Representa, basicamente, uma data. O range de valores suportados varia de '1000-01-01' até '9999-12-31'. O MySQL exibe um valor de DATE no formato 'YYYY-MM-DD', contudo ele permite atribuições para valores de colunas DATE feitas por meio de strings ou números.

DATETIME[(fsp)]

Combinação de uma data com hora. Suporta valores que variam de '1000-01-01 00:00:00.000000' até '9999-12-31 23:59:59.999999'. O MySQL exibe valores DATETIME no formato 'YYYY-MM-DD HH:MM:SS[.fraction]', porém permite atribuir valores ao campo DATETIME usando tanto strings quanto números.

Desde a versão MySQL 5.6.4, o parâmetro opcional fsp pode ser atribuído. Seu valor, como já vimos, varia de 0 a 6 e especifica a fração de segundos que são armazenados. O valor zero especifica que não existe parte fracionária. Se for omitido, o valor default é 0. A versão MySQL 5.6.5 trouxe a inicialização automática e a atualização do valor para o tempo corrente em uma variável DATETIME. Ela pode ser especificada usando os comandos DEFAULT e ON UPDATE durante a definição das colunas.

TIMESTAMP[(fsp)]

Um timestamp ou selo de tempo. Seu intervalo é de '1970-01-01 00: 00: 01,000000' UTC até '2038/01/19 03:14:07,999999 UTC'. TIMESTAMP não pode representar o valor '1970-01-01 00:00:00', porque isso é equivalente a zero segundos, do ponto de vista histórico, e o valor zero (0)

é reservado para representar '0000-00-00 00:00:00', o valor "zero" TIMESTAMP. No MySQL 5.6.4, um valor fsp opcional variando de 0 a 6 pode ser usado para especificar a precisão fracionária de segundos. Já falamos sobre fsp acima.

A forma como o servidor processa definições TIMESTAMP depende do valor da variável de sistema **explicit_defaults_for_timestamp**. Por padrão, ela está desativada e o servidor lida com TIMESTAMP como segue:

Salvo disposição em contrário, a primeira coluna TIMESTAMP de uma tabela é descrita para ser automaticamente definida ou atualizada para a data e hora da modificação mais recente, se não explicitamente atribuído um valor. Isso faz com que TIMESTAMP seja útil para gravar a hora de uma operação INSERT ou UPDATE. Você também pode definir colunas TIMESTAMP com a data e hora atual ao atribuir-lhes o valor NULL, a menos que na definição do atributo seja permitido valores nulos (NULL).

A inicialização automática e a atualização para a data e a hora atual podem ser especificadas usando as cláusulas DEFAULT CURRENT_TIMESTAMP e ON UPDATE CURRENT_TIMESTAMP na definição de coluna.

TIME[(fsp)]

Hora, minuto, segundos e milissegundos, um tempo. Seu intervalo é de '-838:59:59.000000' até '838:59:59.000000'. O MySQL mostra valores TIME no formato 'HH:MM:SS[.fraction]', mas permite atribuição de valores para as colunas TIME usando strings ou números. Vemos mais de uma vez a introdução do valor fsp opcional no MySQL 5.6.4.

YEAR[(2|4)]

Representa um ano no formato de dois dígitos ou de quatro dígitos. O padrão é o formato de quatro dígitos. YEAR(2) ou YEAR(4) diferem em formato de exibição, mas têm o mesmo intervalo de valores. Em formato de quatro dígitos, os valores de exibição são de 1901 a 2155 e 0000. Em formato de dois dígitos, os valores de exibição são de 70 a 69, representando anos de 1970 a 2069. O MySQL mostra valores ano em formato AAAA ou YY, mas permite a atribuição de valores para as colunas ano usando strings ou números.

Tipos de Dados de String

O MySQL interpreta especificações de comprimento para colunas de caracteres em unidades de caracteres. Isso se aplica a CHAR, a VARCHAR, e aos tipos TEXT. Definições de coluna para muitos tipos de dados de strings podem incluir atributos que especificam o conjunto de caracteres (character set) ou COLLATION da coluna. Esses atributos se aplicam à CHAR, à VARCHAR, e aos tipos TEXT, ENUM e SET.

O atributo CHARACTER SET especifica o conjunto de caracteres, e o atributo COLLATE especifica um agrupamento para o conjunto de caracteres. Por exemplo:

```
CREATE TABLE t
(
    c1 VARCHAR(20) CHARACTER SET utf8,
    c2 TEXT CHARACTER SET latin1 COLLATE latin1_general_cs
);
```

Esta definição de tabela cria uma coluna c1, que tem um conjunto de caracteres utf8 com o agrupamento (*collation*) padrão para esse conjunto de caracteres. Cria também uma coluna c2, que tem um conjunto de caracteres latin1 e um agrupamento que diferencia letras maiúsculas de minúsculas (case-sensitive). CHARSET é um sinônimo para CHARACTER SET.

Especificando o atributo conjunto de caracteres para binário, você faz com que a coluna a ser criada use o tipo de dados binário correspondente: CHAR torna-se BINARY, VARCHAR torna-se VARBINARY, e TEXT torna-se BLOB. Para tipos de dados ENUM e SET, isso não ocorre; eles são criados da mesma forma que declarado no comando.

Vamos fazer essa distinção entre CHARACTER SET e COLLATION para ficar mais claro para vocês a diferença entre os dois. Suponha que nós temos um alfabeto com 4 letras: “A”, “B”, “a”, “b”. Nós agora associamos a cada letra a um valor numérico: “A” = 0, “B” = 1, “a” = 2, “b” = 3. A letra “A” é o símbolo, e o numeral 0 é a codificação para “A”. A combinação de todas as 4 letras e suas respectivas codificações é o CHARACTER SET.

Suponha que queremos comparar dois valores de cadeia, "A" e "B". A maneira mais simples de fazer isso é olhar para as codificações: 0 para "A" e 1 para "B". Como zero é menor do que 1, dizemos que "A" é menor do que "B". O que temos feito é apenas aplicar uma *collation* ao nosso conjunto de caracteres. Esse conjunto de regras é conhecido como COLLATION (apenas uma regra neste caso) e é usado para comparar codificações. Chamamos as collations mais simples possíveis de collation binária.

É possível observar uma relação entre o *character set* e *collation*, que é comprovada pela tabela abaixo:

```
mysql> SHOW CHARACTER SET;
+----------+-----------------------------+---------------------+--------+
| Charset  | Description                 | Default collation   | Maxlen |
+----------+-----------------------------+---------------------+--------+
| big5     | Big5 Traditional Chinese    | big5_chinese_ci     |      2 |
| dec8     | DEC West European           | dec8_swedish_ci     |      1 |
| cp850    | DOS West European           | cp850_general_ci    |      1 |
| hp8      | HP West European            | hp8_english_ci      |      1 |
| koi8r    | KOI8-R Relcom Russian       | koi8r_general_ci    |      1 |
| latin1   | cp1252 West European        | latin1_swedish_ci   |      1 |
| latin2   | ISO 8859-2 Central European | latin2_general_ci   |      1 |
| swe7     | 7bit Swedish                | swe7_swedish_ci     |      1 |
| ascii    | US ASCII                    | ascii_general_ci    |      1 |
| ujis     | EUC-JP Japanese             | ujis_japanese_ci    |      3 |
| sjis     | Shift-JIS Japanese          | sjis_japanese_ci    |      2 |
| hebrew   | ISO 8859-8 Hebrew           | hebrew_general_ci   |      1 |
| tis620   | TIS620 Thai                 | tis620_thai_ci      |      1 |
| euckr    | EUC-KR Korean               | euckr_korean_ci     |      2 |
| koi8u    | KOI8-U Ukrainian            | koi8u_general_ci    |      1 |
| gb2312   | GB2312 Simplified Chinese   | gb2312_chinese_ci   |      2 |
| greek    | ISO 8859-7 Greek            | greek_general_ci    |      1 |
```

Um resumo dos tipos de dados para cadeia de caracteres vem em seguida.

[NATIONAL] CHAR[(M)] [CHARACTER SET charset_name] [COLLATE collation_name]

Uma sequência de caracteres de comprimento fixo, que é sempre preenchida à direita com espaços até o tamanho especificado quando armazenado. **M** representa o comprimento da coluna de caracteres. Os valores possíveis para M são de 0 a 255. Se M for omitido, o comprimento é um (1). Espaços extras são removidos quando valores CHAR são recuperados, a menos que o modo de **PAD_CHAR_TO_FULL_LENGTH** está ativado.

CHAR é uma abreviação para CHARACTER. NATIONAL CHAR (ou em sua forma reduzida NCHAR) é o modo padrão de SQL para definir que uma coluna CHAR deve usar alguns conjuntos de caracteres predefinidos. Nas versões superiores ao MySQL 4.1, usa-se utf8 como conjunto de caracteres predefinido.

O tipo de dados CHAR BYTE é um alias para o tipo de dados binários (BINARY). Este é um recurso para garantir compatibilidade.

O MySQL permite que você crie uma coluna do tipo CHAR(0). Isso é útil principalmente quando você tem que ser compatível com aplicativos antigos que dependem da existência de uma coluna, mas que na realidade não usam o seu valor. CHAR(0) também é muito bom quando você precisa de uma coluna que pode ter apenas dois valores. Assim, uma coluna definida como **CHAR(0) NULL** ocupa apenas um bit e pode assumir somente os valores nulos e “” (string vazia).

[NATIONAL] VARCHAR(M) [CHARACTER SET charset_name] [COLLATE collation_name]

Uma sequência de caracteres de comprimento variável. M representa o comprimento máximo da coluna. Os valores de M variam de 0 a 65.535. O comprimento máximo eficaz de um VARCHAR está sujeito ao tamanho máximo da linha (65.535 bytes, que é compartilhado entre todas as colunas) e o conjunto de caracteres usados. Por exemplo, caracteres utf8 podem exigir até três bytes por caractere, portanto, uma coluna VARCHAR que usa o conjunto de caracteres utf8 pode ser declarada no máximo com 21.844 caracteres.

O MySQL armazena valores VARCHAR como um prefixo com comprimento de 1 byte ou 2 bytes. O comprimento do prefixo indica o número de bytes no valor. Uma coluna VARCHAR usa um byte, se os valores não forem maiores do que 255 bytes. E usa dois bytes, se o valor de linha requerer mais de 255 bytes.

MySQL 5.6 segue a especificação SQL padrão e não remove espaços de valores VARCHAR.

VARCHAR é uma abreviação para CHARACTER VARYING. NACIONAL VARCHAR é a maneira padrão SQL para definir que uma coluna VARCHAR deve usar alguns conjuntos de caracteres predefinidos. Versões superiores ao MySQL 4.1 usam utf8 como conjunto de caracteres predefinido. NVARCHAR é uma abreviação para NATIONAL VARCHAR.

BINARY(M)

O tipo binário é semelhante ao tipo CHAR, mas armazena cadeias de bytes binários, em vez de sequências de caracteres não binários. M representa o comprimento da coluna em bytes.

VARBINARY(M)

O tipo VARBINARY é semelhante ao tipo VARCHAR, mas armazena cadeias de bytes binários, em vez de sequências de caracteres não binários. M representa o comprimento máximo de coluna em bytes.

TINYBLOB

Uma coluna BLOB com um comprimento máximo de 255 (2^8-1) bytes. Cada valor armazenado em uma variável TINYBLOB usa um byte de prefixo que indica o número de bytes do valor armazenado.

TINYTEXT [CHARACTER SET charset_name] [COLLATE collation_name]

Uma coluna de texto com um comprimento máximo de 255 (2^8-1) caracteres. O comprimento máximo é menos eficaz se o valor contém caracteres representados em vários bytes. Cada valor é armazenado em TINYTEXT usa um byte de prefixo que indica o número de bytes do valor armazenado.

BLOB[(M)]

Uma coluna BLOB tem o comprimento máximo de 65.535 (2^{16} - 1) bytes. Cada valor é armazenado do tipo BLOB usa um prefixo com comprimento de 2 bytes que indica o número de bytes no valor armazenado.

Um parâmetro M opcional pode ser dado para este tipo. Se isso for feito, o MySQL cria a coluna com o tipo BLOB menor, mas suficientemente grande para conter M bytes de valor.

TEXT[(M)] [CHARACTER SET charset_name] [COLLATE collation_name]

Uma coluna de texto com o comprimento máximo de 65.535 (216 - 1) caracteres. O comprimento máximo é menos eficaz se o valor contém um CHARACTER SET cujos elementos são descritos em mais de um byte. Cada valor de texto é armazenado usando um prefixo comprimento de dois bytes que indica o número de bytes efetivamente gravado.

O comprimento M é opcional e pode ser dado para este tipo. Se isso for feito, o MySQL cria a coluna como o tipo de texto menor, mas grande o suficiente para armazenar valores de texto com comprimento de até M.

MEDIUMTEXT [CHARACTER SET charset_name] [COLLATE collation_name]

A coluna de texto com comprimento máximo de 16.777.215 (2^{24} - 1) caracteres. Cada valor armazenado por um MEDIUMTEXT utiliza um prefixo com três bytes de comprimento que indica o número de bytes armazenados no valor.

LONGBLOB

Uma coluna BLOB com um comprimento máximo de 4294967295 ou 4GB (2^{32} - 1) bytes. O comprimento máximo efetivo de colunas LONGBLOB depende do tamanho máximo do pacote configurado no protocolo de cliente/servidor e da memória disponível. Cada valor armazenado em uma variável do tipo LONGBLOB possui um prefixo que utiliza 4 bytes que indica o número de bytes armazenados no valor.

LONGTEXT [CHARACTER SET charset_name] [COLLATE collation_name]

A coluna de texto com um comprimento máximo de 4,294,967,295 ou 4GB (2^{32}-1) caracteres. Mais uma vez, o comprimento máximo é influenciado pela quantidade de bytes utilizados pelos valores do character set. O comprimento máximo de colunas LONGTEXT também depende do tamanho máximo do pacote configurado no protocolo de cliente/servidor e da memória disponível. Cada valor armazenado em LONGTEXT usa um prefixo de quatro bytes que indica o número de bytes no valor.

ENUM('value1','value2',...) [CHARACTER SET charset_name] [COLLATE collation_name]

Uma enumeração. Um objeto string que só pode ter um valor que é selecionado entre uma lista de valores possíveis: 'valor1', 'valor2', ..., NULL ou um valor de erro especial. Valores ENUM são representados internamente como números inteiros.

Uma coluna ENUM pode ter um máximo de 65.535 elementos distintos. (O limite prático é inferior a 3000). Uma tabela não pode ter mais de 255 definições de ENUM e SET entre suas colunas.

SET('value1','value2',...) [CHARACTER SET charset_name] [COLLATEcollation_name]

Um conjunto. Um objeto string que pode ter zero ou mais valores, cada um dos quais deve ser escolhido a partir da lista de valores 'valor1', 'valor2', ... Os valores de um SET são representados internamente como números inteiros.

A coluna SET pode ter um máximo de 64 valores distintos. Conforme já falamos, uma tabela não pode ter mais de 255 definições de ENUM e SET entre suas colunas.

EXTENSÕES PARA DADOS ESPECIAIS

O Open Geospatial Consortium (OGC) é um consórcio internacional de mais de 250 empresas, agências e universidades que participam no desenvolvimento de soluções conceituais disponíveis publicamente. Essas soluções podem ser úteis para todos os tipos de aplicações que gerenciam dados espaciais.

O Open Geospatial Consortium publica a Norma OpenGIS® Implementation Standard for Geographic - Simple feature access - Part 2: SQL option, um documento que propõe diversas formas conceituais para estender um SGBD relacional SQL para suportar dados espaciais. Esta especificação está disponível no site da OGC em http://www.opengeospatial.org/standards/sfs.

Após a especificação OGC, o MySQL implementou extensões espaciais, como um subconjunto do SQL adaptando o ambiente para tipos geométricos (SQL with Geometry Types). Este termo refere-se a um ambiente SQL que foi estendido com um conjunto de tipos geométricos. A especificação descreve um conjunto de tipos geométricos SQL, bem como funções para manipular este tipo, visando criar e analisar valores geométricos. As extensões espaciais permitem a geração, o armazenamento e a análise de características geográficas. Possuem algumas características:

- Tipos de dados para representar valores espaciais
- Funções para manipular valores espaciais
- Indexação espacial para melhorar os tempos de acesso às colunas espaciais

Os tipos de dados e funções estão disponíveis para a storage engine MyISAM, tabelas InnoDB, NDB e arquivo. Para indexar colunas espaciais, MyISAM suporta índices espaciais e não-espaciais. Os outros mecanismos de armazenamento suportam índices não espaciais.

Aqui terminamos a apresentação dos tipos do MySQL. Apenas selecionamos os conceitos básicos, para que você possa ter um entendimento holístico da estrutura dos tipos de dados do MySQL. Agora já podemos dar continuidade ao nosso estudo analisando os comandos DDL.

Data Definition Language (DDL)

Quando analisamos a divisão do conteúdo descrito na parte de comandos para definições de estruturas, observamos uma gama gigantesca de comandos. Apenas para termos uma ideia, são 33 tópicos, cada um analisando um comando distinto e suas características.

Foge totalmente ao nosso escopo fazer qualquer análise detalhada sobre eles. Decidimos, porém agrupá-los em três grandes grupos: ALTER, CREATE e DROP.

ALTER

É possível usar o comando de ALTER para fazer modificações em uma das seguintes estruturas: DATABASE, EVENT, LOGFILE GROUP, FUNCTION, PROCEDURE, SERVER, TABLE, TABLESPACE e VIEW. Vejamos cada uma das possíveis formas de utilização.

ALTER DATABASE permite ao DBA, ou administrador de banco de dados, fazer modificações nas características do seu banco de dados. Essas características são gravadas no arquivo db.opt no diretório do banco. Para executar o comando, é preciso ter permissão de ALTER sobre a base de dados. O comando ALTER SCHEMA é um sinônimo para o ALTER DATABASE. Vejamos abaixo uma figura com a sintaxe do comando:

```
ALTER {DATABASE | SCHEMA} [db_name]
    alter_specification ...
ALTER {DATABASE | SCHEMA} db_name
    UPGRADE DATA DIRECTORY NAME

alter_specification:
    [DEFAULT] CHARACTER SET [=] charset_name
  | [DEFAULT] COLLATE [=] collation_name
```

Veja que, na figura acima, temos duas especificações que podem ser modificadas por meio do comando. A primeira refere-se ao CHARACTER SET, um conjunto de símbolos e codificações. A outra, COLLATE, é um conjunto de regras para comparar caracteres em um CHARACTER SET. Já falamos sobre eles quando tratamos das definições sobre tipos de dados.

O segundo comando da lista é **ALTER EVENT**. Ele altera a característica de um evento, sem precisar excluir e recriar. O usuário pode alterar um evento desde que ele tenha o privilégio de EVENT sobre este evento. Ao fazer a modificação, você se torna o definidor do evento. Vejamos a sintaxe do comando abaixo:

```
ALTER
    [DEFINER = { user | CURRENT_USER }]
    EVENT event_name
    [ON SCHEDULE schedule]
    [ON COMPLETION [NOT] PRESERVE]
    [RENAME TO new_event_name]
    [ENABLE | DISABLE | DISABLE ON SLAVE]
    [COMMENT 'comment']
    [DO event_body]
```

Vejam algumas características que podem ser redefinidas. Perceba que, para fazermos um RENAME no objeto EVENT, utilizamos o comando ALTER EVENT com a clausula “RENAME TO”. É possível, ainda, utilizar o RENAME para mover o evento para um banco de dados diferente. Observe o comando abaixo:

```
ALTER EVENT olddb.myevent
    RENAME TO newdb.myevent;
```

Vamos agora conhecer o **ALTER LOGFILE GROUP**. Este comando permite adicionar um arquivo de UNDO com um determinado nome a um grupo de arquivos de log. Vejam a sintaxe do comando abaixo:

```
ALTER LOGFILE GROUP logfile_group
    ADD UNDOFILE 'file_name'
    [INITIAL_SIZE [=] size]
    [WAIT]
    ENGINE [=] engine_name
```

O parâmetro INITIAL_SIZE define o tamanho inicial do arquivo em bytes. Caso não seja definido, o valor default é 128 MB. Veja ainda que a ENGINE é um atributo obrigatório e determina a storage engine que será utilizada.

Continuamos agora com o **ALTER FUNCTION**. Este comando pode ser usado para alterar as características de uma função armazenada. Mais de uma mudança pode ser especificada em uma instrução ALTER FUNCTION. No entanto, você não pode alterar os parâmetros ou corpo de uma função armazenada usando esta declaração. Para fazer tais mudanças, você deve apagar e recriar a função, usando os comandos DROP FUNCTION e CREATE FUNCTION.

Você deve ter o privilégio ALTER ROUTINE para a função (o privilégio é concedido automaticamente para o criador função). Se o log binário estiver habilitado, a instrução ALTER FUNCTION também pode requerer o privilégio SUPER. Veja abaixo a sintaxe do comando:

```
ALTER FUNCTION func_name [characteristic ...]

characteristic:
    COMMENT 'string'
  | LANGUAGE SQL
  | { CONTAINS SQL | NO SQL | READS SQL DATA | MODIFIES SQL DATA }
  | SQL SECURITY { DEFINER | INVOKER }
```

O próximo comando é o **ALTER PROCEDURE**. Ele pode ser usado para alterar as características de um procedimento armazenado. Mais do que uma mudança pode ser especificada em uma instrução ALTER PROCEDURE. No entanto, você não pode alterar os parâmetros ou corpo de um procedimento

armazenado usando esta declaração. Para fazer tais mudanças, você deve excluir e recriar o procedimento utilizando DROP PROCEDURE e CREATE PROCEDURE.

Você deve ter o privilégio ALTER ROUTINE para o procedimento. Por padrão, esse privilégio é concedido automaticamente para o criador do procedimento. Este comportamento pode ser alterado, desativando a variável de sistema **automatic_sp_privileges**.

Outro comando não muito conhecido é o **ALTER SERVER**. Este comando altera as informações do servidor para nome_do_servidor passado como parâmetro, ajustando qualquer uma das opções permitidas na instrução CREATE SERVER. Os campos correspondentes na tabela de mysql.servers são atualizados. Esta declaração exige o privilégio SUPER.

O comando mais importante desta sessão talvez seja o **ALTER TABLE**. Ele altera a estrutura de uma tabela. Por exemplo, você pode adicionar ou excluir colunas, criar ou destruir índices, alterar o tipo de colunas existentes, ou renomear as colunas ou a própria tabela. Você também pode alterar características, tais como o mecanismo de armazenamento utilizado pela tabela ou o comentário da tabela.

Após descrever o nome da tabela, você dever especificar as alterações a serem feitas. Se nenhuma informação for passada, o comando ALTER TABLE não faz nada. A sintaxe para muitas das alterações admissíveis é semelhante às cláusulas da instrução CREATE TABLE. Seguem algumas Informações úteis ainda sobre o comando ALTER TABLE.

Quando você especificar uma cláusula ENGINE, o ALTER TABLE reconstrói a tabela. Isso acontece mesmo se a tabela já tem o motor de armazenamento especificado. Para utilizar o comando ALTER TABLE, você precisa dos privilégios de ALTER, CREATE e INSERT para a tabela. Mudar o nome de uma tabela requer os privilégios de ALTER e DROP na tabela antiga, e ALTER, CREATE e INSERT na nova tabela.

Para evitar a perda acidental de dados, ALTER TABLE não pode ser usado para mudar o mecanismo de armazenamento de uma tabela para MERGE ou BLACKHOLE.

Você pode emitir um comando com múltiplas modificações ADD, ALTER, DROP e CHANGE em uma única instrução ALTER TABLE, separados por vírgulas. Esta é uma extensão do MySQL ao SQL padrão, que só permite uma cláusula de modificação por instrução ALTER TABLE. Por exemplo, para dropar várias colunas em uma única instrução, utilize o seguinte:

ALTER TABLE DROP COLUMN t2 c, DROP COLUMN d;

CHANGE nome_campo, DROP nome_campo e DROP INDEX são extensões do MySQL ao SQL padrão. A palavra COLUMN é opcional e pode ser omitida. Vejam alguns exemplos do ALTER TABLE abaixo:

Para alterar uma coluna de inteiro para TINYINT NOT NULL (deixando o nome da mesma), e para alterar coluna b de CHAR (10) para CHAR (20), bem como mudar o nome de b para c:

ALTER TABLE t2 MODIFY a TINYINT NOT NULL, CHANGE b c CHAR(20);

Para adicionar uma nova coluna TIMESTAMP d:

ALTER TABLE t2 ADD d TIMESTAMP;

Para adicionar um índice a uma coluna d e um UNIQUE índice para a coluna a:

ALTER TABLE t2 ADD INDEX (d), ADD UNIQUE (a);

ALTER TABLESPACE é nosso penúltimo comando. Este comando pode ser usado tanto para adicionar um novo arquivo de dados, quanto para dropar um arquivo de dados de um espaço de tabela. Uma variação do comando pode usar o **ADD DATAFILE** que permite especificar um tamanho inicial usando uma cláusula **INITIAL_SIZE**. O tamanho é medido em bytes, sendo o valor padrão de 134217728 (**128 MB**).

ALTER VIEW é o último comando de alteração desta nossa aula, lembrando que **estamos falando de DDL**. Esta declaração altera a definição de uma visão, que já deve existir no banco de dados. A sintaxe é semelhante ao comando **CREATE VIEW** e o efeito é o mesmo do **CREATE OR REPLACE VIEW**. Para executar esse comando, é necessário ter os privilégios de CREATE VIEW e DROP para a visão específica, além do privilégio leitura para cada coluna referida na instrução SELECT. ALTER VIEW é autorizada apenas para o definidor ou para os usuários com o privilégio SUPER sobre a visão.

CREATE

Passaremos agora para os comandos que iniciam com o identificador CREATE. A lista de comandos passa pelos mesmos modificares do ALTER com a adição do CREATE TRIGGER. Veja, portanto, que não existe ALTER TRIGGER na sintaxe do MySQL. As outras opções do CREATE são DATABASE, EVENT, INDEX, LOGFILE GROUP, FUNCTION, PROCEDURE, SERVER, TABLE, TABLESPACE e VIEW.

Dado que nosso curso está voltado especificamente para concursos, nosso objetivo é descrever apenas os assuntos com maior probabilidade de cair em provas de concurso. Passaremos agora para analisarmos os comandos DDL de CREATE INDEX, CREATE PROCEDURE, CREATE FUNCTION e CREATE TABLE. Os demais comandos não serão abordados em detalhes.

Vejamos a seguir a sintaxe do CREATE INDEX:

```
CREATE [ONLINE|OFFLINE] [UNIQUE|FULLTEXT|SPATIAL] INDEX index_name
    [index_type]
    ON tbl_name (index_col_name,...)
    [index_option]
    [algorithm_option | lock_option] ...

index_col_name:
    col_name [(length)] [ASC | DESC]

index_type:
    USING {BTREE | HASH}

index_option:
    KEY_BLOCK_SIZE [=] value
  | index_type
  | WITH PARSER parser_name
  | COMMENT 'string'

algorithm_option:
    ALGORITHM [=] {DEFAULT|INPLACE|COPY}

lock_option:
    LOCK [=] {DEFAULT|NONE|SHARED|EXCLUSIVE}
```

CREATE INDEX é mapeado para uma instrução ALTER TABLE para criar índices. CREATE INDEX não pode ser usado para criar uma chave primária; utilize ALTER TABLE para esta operação. Normalmente, você cria todos os índices de uma tabela no momento da criação da mesma com CREATE TABLE. Esta orientação é especialmente importante para tabelas que usam InnoDB, em que a chave primária determina o layout físico de linhas no arquivo de dados.

CREATE INDEX permite adicionar índices a tabelas existentes. A lista é formada por uma ou mais colunas (col1, col2, ...) e cria uma coluna de índice sobre essas colunas. Valores de chave de índice são formados concatenando os valores de colunas dadas.

Os índices podem ser criados utilizando apenas a parte inicial dos valores de coluna, usando a sintaxe nome_coluna(length) para especificar um comprimento do prefixo de índice. A instrução mostrada abaixo cria um índice usando os primeiros 10 caracteres da coluna name:

```
CREATE INDEX part_of_name ON customer (name(10));
```

Vejamos agora o que significa algumas das palavras chaves que compõem o índice. As palavras-chave ONLINE e OFFLINE estão disponíveis apenas para o MySQL Cluster; a tentativa de usar essas palavras-chave no padrão MySQL 5.6 exibe um erro de sintaxe.

Um índice **UNIQUE** cria uma restrição de tal forma que todos os valores do índice devem ser distintos. Um erro ocorre se você tentar adicionar uma nova linha com um valor de chave que corresponda a uma linha existente. Para todos os motores, um índice UNIQUE permite vários valores NULL para colunas que podem conter NULL. Se você especificar um valor de índice UNIQUE como prefixo de uma coluna, os valores da coluna devem ser exclusivos dentro do prefixo.

Índices **FULLTEXT** são suportados apenas para InnoDB e MyISAM. Estes índices podem incluir apenas colunas **CHAR**, **VARCHAR** e **TEXT**. A indexação sempre acontece sobre a coluna inteira. A indexação baseada em prefixo de coluna não é suportada e qualquer comprimento prefixo é ignorado, se especificado.

Os mecanismos de armazenamento MyISAM, InnoDB, NDB, e ARCHIVE suportam colunas espaciais como POINT e GEOMETRY. No entanto, o suporte para a indexação de colunas espaciais varia entre os motores. Os índices espaciais (criados usando o SPATIAL INDEX) têm as seguintes características:

- Disponível somente para tabelas MyISAM. Especificando SPATIAL INDEX para outros mecanismos de armazenamento resulta em um erro.
- Colunas cadastradas devem ser NOT NULL.
- No MySQL 5.6, prefixo de comprimentos de coluna são proibidos. A largura total de cada coluna é indexada.

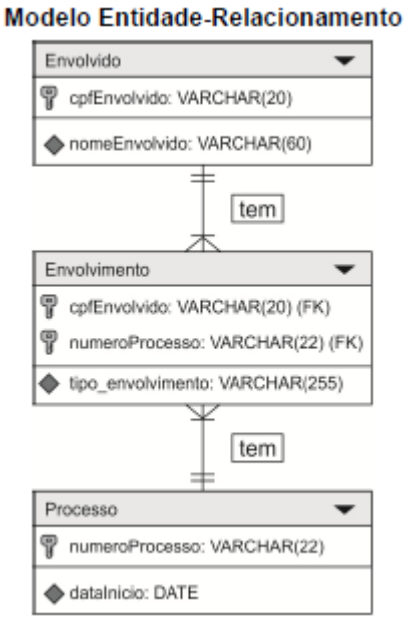
Outro parâmetro importante é o index_type. Alguns mecanismos de armazenamento permitem que você especifique um tipo de índice ao criá-lo. Os valores de tipo de índice suportados por diferentes mecanismos de armazenamento são mostrados na tabela a seguir. Se vários tipos de índices forem listados, o primeiro é o padrão caso nenhum tipo de índice seja especificado.

Storage Engine	Permissible Index Types
InnoDB	BTREE
MyISAM	BTREE
MEMORY/HEAP	HASH, BTREE
NDB	HASH, BTREE

Vamos agora fazer a questão abaixo antes de passarmos para o comando CREATE PROCEDUDE.

3. FCC - Analista Ministerial (MPE PB)/Analista de Sistemas/Administrador de Banco de Dados/2015

Atenção: Considere as informações a seguir para responder à questão.



Dados cadastrados na tabela:

cpfEnvolvido	nomeEnvolvido
121.134.045-01	Marcos Paulo
128.249.039-14	Maria de Fátima
131.091.431-09	André Luiz
158.245.067-12	Pedro nda Silva
160.234.074-11	João da Silva

Observação: O erro no nome Pedro nda Silva é proposital.

Na tabela Envolvido, deseja-se incluir um campo dataNascEnvolvido cujo preenchimento será obrigatório, imediatamente após o campo cpfEnvolvido. Considerando que o banco de dados e as tabelas foram criados no MySQL, deve-se utilizar, para isso, a instrução

a) ALTER TABLE Envolvido ADD COLUMN dataNascEnvolvido DATE NOT NULL AFTER cpfEnvolvido;

b) ADD COLUMN dataNascEnvolvido DATE NOT NULL FROM Envolvido AFTER cpfEnvolvido;

c) UPDATE TABLE Envolvido ADD COLUMN dataNascEnvolvido DATE NOT NULL AFTER cpfEnvolvido;

d) ALTER TABLE Envolvido ADD COLUMN dataNascEnvolvido DATE NOT NULL AFTER (cpfEnvolvido);

e) INSERT COLUMN dataNascEnvolvido DATE NOT NULL AFTER cpfEnvolvido FROM Envolvido;

Comentários: ALTER TABLE altera a estrutura de uma tabela. Por exemplo, você pode adicionar ou excluir colunas, criar ou destruir índices, alterar o tipo de colunas existentes ou renomear colunas ou a própria tabela. Você também pode alterar características, como o mecanismo de armazenamento usado para a tabela ou o comentário da tabela.

Para usar ALTER TABLE, você precisa de privilégios ALTER, CREATE e INSERT para a tabela. Renomear uma tabela requer ALTER e DROP na tabela antiga, ALTER, CREATE e INSERT na nova tabela.

Seguindo o nome da tabela, especifique as alterações a serem feitas. Se nenhum for dado, ALTER TABLE não faz nada.

Múltiplas cláusulas ADD, ALTER, DROP e CHANGE são permitidas em uma única instrução ALTER TABLE, separadas por vírgulas. Esta é uma extensão do MySQL para SQL padrão, que permite apenas uma de cada cláusula por instrução ALTER TABLE. Por exemplo, para adicionar várias colunas em uma única instrução, faça o seguinte:

ALTER TABLE t2 ADD COLUMN c, ADD COLUMN d;

Segue a sintaxe, considerando o enunciado:

```
ALTER TABLE tbl_name
    [alter_specification [, alter_specification] ...]
    [partition_options]
alter_specification:
    table_options
  | ADD [COLUMN] col_name column_definition
        [FIRST | AFTER col_name]
```

Temos como gabarito da questão a letra a):

ALTER TABLE Envolvido ADD COLUMN dataNascEnvolvido DATE NOT NULL AFTER cpfEnvolvido

Gabarito: A

Vamos tratar agora do CREATE PROCEDURE. Primeiramente, observe a sintaxe do comando abaixo. Apresentamos também a sintaxe do CREATE FUNCTION.

```
CREATE
    [DEFINER = { user | CURRENT_USER }]
    PROCEDURE sp_name ([proc_parameter[,...]])
    [characteristic ...] routine_body

CREATE
    [DEFINER = { user | CURRENT_USER }]
    FUNCTION sp_name ([func_parameter[,...]])
    RETURNS type
    [characteristic ...] routine_body

proc_parameter:
    [ IN | OUT | INOUT ] param_name type

func_parameter:
    param_name type

type:
    Any valid MySQL data type

characteristic:
    COMMENT 'string'
  | LANGUAGE SQL
  | [NOT] DETERMINISTIC
  | { CONTAINS SQL | NO SQL | READS SQL DATA | MODIFIES SQL DATA }
  | SQL SECURITY { DEFINER | INVOKER }

routine_body:
    Valid SQL routine statement
```

Estas declarações acima criam rotinas armazenadas. Por padrão, a rotina está associada ao banco de dados padrão. Para associar a rotina explicitamente a um determinado banco de dados, especifique o nome da procedure como db_name.sp_name ao criá-la.

Para invocar um procedimento armazenado, utilize a instrução CALL. Para chamar uma função armazenada, você pode referenciá-la em uma expressão. A função retorna um valor durante a avaliação da expressão.

CREATE PROCEDURE e CREATE FUNCTION exigem o privilégio CREATE ROUTINE. Eles também podem requerer o privilégio SUPER, dependendo do valor DEFINER. Se o log binário estiver habilitado, CREATE FUNCTION pode requerer o privilégio SUPER. Por padrão, MySQL concede automaticamente os privilégios de ALTER ROUTINE e EXECUTE para o criador da rotina. Este comportamento pode ser alterado, desativando a variável de sistema **automatic_sp_privileges**.

As cláusulas DEFINER e SQL SECURITY especificam o contexto de segurança utilizado para verificar os privilégios de acesso em tempo de execução da rotina.

Se o nome da rotina é o mesmo que o nome de uma função interna do SQL, um erro de sintaxe ocorre ao menos que você use um espaço após o nome e a definição da rotina ou invocá-la mais tarde. Por esta razão, evite usar os nomes de funções SQL existentes para suas próprias rotinas armazenadas. O modo IGNORE_SPACE SQL se aplica às funções internas, não para rotinas armazenadas. É sempre permitido ter espaços depois de um nome de rotina armazenada, independentemente do IGNORE_SPACE estar ativado.

A lista de parâmetros entre parênteses deve estar sempre presente. Se não houver nenhum parâmetro, deve ser usada uma lista de parâmetros vazia (). Os nomes de parâmetros não são *case sensitive*.

Cada parâmetro é definido como IN por padrão. Para especificar outro tipo de parâmetro, use a palavra-chave OUT ou INOUT antes do nome do parâmetro. A especificação de um parâmetro como IN, OUT ou INOUT só é válida para uma PROCEDURE. Para função, os parâmetros são sempre categorizados como IN.

Agora falaremos do último comando, o **CREATE TABLE**. As possíveis variações sintáticas do comando ocupam quase duas páginas no manual de uso. Optamos por apresentar abaixo apenas a parte principal do comando:

```
CREATE [TEMPORARY] TABLE [IF NOT EXISTS] tbl_name
    { LIKE old_tbl_name | (LIKE old_tbl_name) }

create_definition:
    col_name column_definition
  | [CONSTRAINT [symbol]] PRIMARY KEY [index_type] (index_col_name,...)
      [index_option] ...
  | {INDEX|KEY} [index_name] [index_type] (index_col_name,...)
      [index_option] ...
  | [CONSTRAINT [symbol]] UNIQUE [INDEX|KEY]
      [index_name] [index_type] (index_col_name,...)
      [index_option] ...
  | {FULLTEXT|SPATIAL} [INDEX|KEY] [index_name] (index_col_name,...)
      [index_option] ...
  | [CONSTRAINT [symbol]] FOREIGN KEY
      [index_name] (index_col_name,...) reference_definition
  | CHECK (expr)

column_definition:
    data_type [NOT NULL | NULL] [DEFAULT default_value]
      [AUTO_INCREMENT] [UNIQUE [KEY] | [PRIMARY] KEY]
      [COMMENT 'string']
      [COLUMN_FORMAT {FIXED|DYNAMIC|DEFAULT}]
      [STORAGE {DISK|MEMORY|DEFAULT}]
      [reference_definition]
```

Vejam que você pode usar a palavra-chave TEMPORARY ao criar uma tabela. Uma tabela temporária é visível apenas para a sessão atual. Ela é descartada automaticamente quando a sessão é encerrada. Isso significa que duas sessões diferentes podem usar o mesmo nome de tabela temporária, não gerando conflito uma com a outra ou com tabelas não temporárias existentes com o mesmo nome.

Observem que é possível definir ainda as restrições de integridade e alguns parâmetros para os tipos de dados. Esses tipos de dados podem ser qualquer um daqueles que tratamos anteriormente no nosso curso. As restrições podem ser de chave primária, estrangeira, not null e unique. Podemos definir os tipos ainda como auto incrementado. Tente analisar a sintaxe do comando acima e procure as palavras-chaves para as especificidades que acabamos de comentar.

DROP

O DROP opera sobre a mesma lista de objetos do CREATE, nada mais lógico! Tudo que é criado dentro do banco de dados pode ser apagado. A lista inclui todas as opções que vimos: DATABASE,

EVENT, FUNCTION, INDEX, LOGFILE GROUP, PROCEDURE, SERVER, TABLE, TRIGGER, TABLESPACE e VIEW.

DROP DATABASE apaga todas as tabelas no banco de dados e exclui o banco de dados. Tenha muito cuidado com esta afirmação! Para usar DROP DATABASE, é necessário o privilégio DROP no banco de dados. DROP SCHEMA é um sinônimo para DROP DATABASE. IF EXISTS é utilizado para prevenir a ocorrência de erros, se o banco de dados a ser apagado não existe. Se o banco de dados padrão é descartado, o banco de dados padrão fica definido como unset (a função DATABASE() retorna NULL). Veja a sintaxe do DROP DATABASE abaixo:

```
DROP {DATABASE | SCHEMA} [IF EXISTS] db_name
```

A declaração **DROP EVENT** exclui o evento nomeado (event_name). O evento imediatamente deixa de ser ativo e é excluído completamente do servidor.

O comando **DROP FUNCTION** é usado para apagar funções armazenadas e funções definidas pelo usuário, conhecidas como user-defined functions (UDFs).

DROP INDEX apaga um índice da tabela. Esta declaração é mapeada para uma instrução ALTER TABLE para fazer o DROP do índice. Para excluir o índice de uma chave primária, em que o nome do índice é sempre descrito como PRIMARY, deve ser especificado como um identificador entre aspas porque PRIMARY é uma palavra reservada: DROP INDEX `PRIMARY` ON t; .

A declaração **DROP LOGFILE GROUP** apaga o grupo de arquivo de log. O grupo de arquivos de log já deve existir ou ocorrerá um erro.

O comando **DROP PROCEDURE** é usado para apagar um procedimento armazenado ou função. Ou seja, a rotina especificada é removida do servidor. Você deve ter o privilégio ALTER ROUTINE para a rotina. Se a variável de sistema **automatic_sp_privileges** está habilitada, o privilégio de executar e apagar são concedidos automaticamente ao criador da rotina.

DROP SERVER apaga a definição de um servidor. A linha correspondente na tabela de mysql.servers é excluída. Esta declaração exige o privilégio SUPER.

DROP TABLE remove uma ou mais tabelas. Você deve ter o privilégio DROP para cada tabela. Todos os dados da tabela e a definição da tabela são removidos, por isso tenha cuidado com este comando! Se qualquer uma das tabelas da lista de argumentos não existe, o MySQL retorna um erro indicando que essas tabelas não foram apagadas, mas dropa todas as tabelas na lista que existem.

Para uma tabela de partição, DROP TABLE remove permanentemente a definição da tabela, todas as suas partições e todos os dados que foram armazenados nessas partições. Ele também remove o arquivo de definição de particionamento (.par) associado com a tabela descartada.

Você pode utilizar o IF EXISTS para prevenir a ocorrência de um erro para tabelas que não existem. Uma nota é gerada para cada tabela inexistente quando se utiliza o IF EXISTS. RESTRICT e CASCADE são permitidos para fazer uma portabilidade mais fácil. Veja a sintaxe do comando abaixo:

```
DROP [TEMPORARY] TABLE [IF EXISTS]
    tbl_name [, tbl_name] ...
    [RESTRICT | CASCADE]
```

DROP TABLESPACE apaga um tablespace que foi anteriormente definida pelo comando CREATE TABLESPACE.

A declaração **DROP TRIGGER** apaga um gatilho. O banco de dados ou nome do esquema é opcional. Se o esquema for omitido, o gatilho será removido do esquema padrão. DROP TRIGGER requer o privilégio TRIGGER para a tabela associada com o gatilho. Veja a sintaxe abaixo:

```
DROP TRIGGER [IF EXISTS] [schema_name.]trigger_name
```

DROP VIEW remove uma ou mais VIEWs. Você deve ter o privilégio DROP sobre cada visão. Se qualquer uma das visões na lista de argumentos não existir, o MySQL retorna um erro indicando pelo nome quais não existem e não conseguiu apagar, mas também deleta todas as VIEWS que existem na lista.

RENAME TABLE

```
RENAME TABLE tbl_name TO new_tbl_name
[, tbl_name2 TO new_tbl_name2]...
```

Esta declaração renomeia uma ou mais tabelas. A operação de *rename* é feita atomicamente, o que significa que nenhuma outra sessão pode acessar qualquer uma das tabelas enquanto a ação está sendo executada. Por exemplo, uma tabela chamada tabela_antiga pode ser renomeada para nova_tabela como mostrado aqui:

```
RENAME TABLE tabela_antiga TO nova_tabela;
```

Esta declaração é equivalente à seguinte instrução ALTER TABLE:

```
ALTER TABLE RENAME new_table OLD_TABLE;
```

Se a declaração renomeia mais de uma tabela, as operações de mudança de nome são feitas da esquerda para a direita. Se você quiser trocar os nomes de tabela, você pode fazê-lo como a seguir (supondo que tmp_table ainda não existe):

```
RENAME TABLE tabela_antiga TO tmp_table,
new_table TO tabela_antiga,
tmp_table TO nova_tabela;
```

O MySQL verifica o nome da tabela de destino antes de verificar se a tabela de origem existe. Por exemplo, se nova_tabela já existe e a tabela_antiga não, a seguinte declaração de falha é mostrada:

```
mysql> SHOW TABLES;
+-----------------+
| Tables_in_mydb |
+-----------------+
| table_a         |
+-----------------+
1 row in set (0.00 sec)
mysql> RENAME TABLE table_b TO table_a;
ERROR 1050 (42S01): Table 'table_a' already exists
```

TRUNCATE

TRUNCATE TABLE esvazia uma tabela completamente. Ela exige o privilégio de DROP. Logicamente, TRUNCATE TABLE é semelhante a uma instrução de DELETE que exclui todas as linhas, ou a uma sequência de DROP TABLE e CREATE TABLE. Para atingir um alto desempenho, o comando ignora o método DML de exclusão de dados. Assim, o comando não pode ser revertido, não aciona gatilhos de ON DELETE e não pode ser realizado em tabelas InnoDB com relações de chave estrangeira entre pais e filhos.

Embora TRUNCATE TABLE seja semelhante ao DELETE, ele é classificado como uma instrução DDL em vez de uma instrução DML.

DATA DEFINITION LANGUAGE (DDL)

Quando analisamos a divisão do conteúdo descrito na parte de comandos para definições de estruturas, observamos uma gama gigantesca de comandos. Apenas para termos uma ideia, são 33 tópicos, cada um analisando um comando distinto e suas características.

Foge totalmente ao nosso escopo fazer qualquer análise detalhada sobre eles. Decidimos, porém agrupá-los em três grandes grupos: ALTER, CREATE e DROP.

ALTER

É possível usar o comando de ALTER para fazer modificações em uma das seguintes estruturas: DATABASE, EVENT, LOGFILE GROUP, FUNCTION, PROCEDURE, SERVER, TABLE, TABLESPACE e VIEW. Vejamos cada uma das possíveis formas de utilização.

ALTER DATABASE permite ao DBA, ou administrador de banco de dados, fazer modificações nas características do seu banco de dados. Essas características são gravadas no arquivo db.opt no diretório do banco. Para executar o comando, é preciso ter permissão de ALTER sobre a base de dados. O comando ALTER SCHEMA é um sinônimo para o ALTER DATABASE. Vejamos abaixo uma figura com a sintaxe do comando:

```
ALTER {DATABASE | SCHEMA} [db_name]
    alter_specification ...
ALTER {DATABASE | SCHEMA} db_name
    UPGRADE DATA DIRECTORY NAME

alter_specification:
    [DEFAULT] CHARACTER SET [=] charset_name
  | [DEFAULT] COLLATE [=] collation_name
```

Veja que, na figura acima, temos duas especificações que podem ser modificadas por meio do comando. A primeira refere-se ao CHARACTER SET, um conjunto de símbolos e codificações. A outra, COLLATE, é um conjunto de regras para comparar caracteres em um CHARACTER SET. Já falamos sobre eles quando tratamos das definições sobre tipos de dados.

O segundo comando da lista é **ALTER EVENT**. Ele altera a característica de um evento, sem precisar excluir e recriar. O usuário pode alterar um evento desde que ele tenha o privilégio de EVENT sobre este evento. Ao fazer a modificação, você se torna o definidor do evento. Vejamos a sintaxe do comando abaixo:

```
ALTER
    [DEFINER = { user | CURRENT_USER }]
    EVENT event_name
    [ON SCHEDULE schedule]
    [ON COMPLETION [NOT] PRESERVE]
    [RENAME TO new_event_name]
    [ENABLE | DISABLE | DISABLE ON SLAVE]
    [COMMENT 'comment']
    [DO event_body]
```

Vejam algumas características que podem ser redefinidas. Perceba que, para fazermos um RENAME no objeto EVENT, utilizamos o comando ALTER EVENT com a clausula “RENAME TO”. É possível, ainda, utilizar o RENAME para mover o evento para um banco de dados diferente. Observe o comando abaixo:

```
ALTER EVENT olddb.myevent
    RENAME TO newdb.myevent;
```

Vamos agora conhecer o **ALTER LOGFILE GROUP**. Este comando permite adicionar um arquivo de UNDO com um determinado nome a um grupo de arquivos de log. Vejam a sintaxe do comando abaixo:

```
ALTER LOGFILE GROUP logfile_group
    ADD UNDOFILE 'file_name'
    [INITIAL_SIZE [=] size]
    [WAIT]
    ENGINE [=] engine_name
```

O parâmetro INITIAL_SIZE define o tamanho inicial do arquivo em bytes. Caso não seja definido, o valor default é 128 MB. Veja ainda que a ENGINE é um atributo obrigatório e determina a storage engine que será utilizada.

Continuamos agora com o **ALTER FUNCTION**. Este comando pode ser usado para alterar as características de uma função armazenada. Mais de uma mudança pode ser especificada em uma instrução ALTER FUNCTION. No entanto, você não pode alterar os parâmetros ou corpo de uma função armazenada usando esta declaração. Para fazer tais mudanças, você deve apagar e recriar a função, usando os comandos DROP FUNCTION e CREATE FUNCTION.

Você deve ter o privilégio ALTER ROUTINE para a função (o privilégio é concedido automaticamente para o criador função). Se o log binário estiver habilitado, a instrução ALTER FUNCTION também pode requerer o privilégio SUPER. Veja abaixo a sintaxe do comando:

```
ALTER FUNCTION func_name [characteristic ...]

characteristic:
    COMMENT 'string'
  | LANGUAGE SQL
  | { CONTAINS SQL | NO SQL | READS SQL DATA | MODIFIES SQL DATA }
  | SQL SECURITY { DEFINER | INVOKER }
```

O próximo comando é o **ALTER PROCEDURE**. Ele pode ser usado para alterar as características de um procedimento armazenado. Mais do que uma mudança pode ser especificada em uma instrução ALTER PROCEDURE. No entanto, você não pode alterar os parâmetros ou corpo de um procedimento

armazenado usando esta declaração. Para fazer tais mudanças, você deve excluir e recriar o procedimento utilizando DROP PROCEDURE e CREATE PROCEDURE.

Você deve ter o privilégio ALTER ROUTINE para o procedimento. Por padrão, esse privilégio é concedido automaticamente para o criador do procedimento. Este comportamento pode ser alterado, desativando a variável de sistema **automatic_sp_privileges**.

Outro comando não muito conhecido é o **ALTER SERVER**. Este comando altera as informações do servidor para nome_do_servidor passado como parâmetro, ajustando qualquer uma das opções permitidas na instrução CREATE SERVER. Os campos correspondentes na tabela de mysql.servers são atualizados. Esta declaração exige o privilégio SUPER.

O comando mais importante desta sessão talvez seja o **ALTER TABLE**. Ele altera a estrutura de uma tabela. Por exemplo, você pode adicionar ou excluir colunas, criar ou destruir índices, alterar o tipo de colunas existentes, ou renomear as colunas ou a própria tabela. Você também pode alterar características, tais como o mecanismo de armazenamento utilizado pela tabela ou o comentário da tabela.

Após descrever o nome da tabela, você dever especificar as alterações a serem feitas. Se nenhuma informação for passada, o comando ALTER TABLE não faz nada. A sintaxe para muitas das alterações admissíveis é semelhante às cláusulas da instrução CREATE TABLE. Seguem algumas Informações úteis ainda sobre o comando ALTER TABLE.

Quando você especificar uma cláusula ENGINE, o ALTER TABLE reconstrói a tabela. Isso acontece mesmo se a tabela já tem o motor de armazenamento especificado. Para utilizar o comando ALTER TABLE, você precisa dos privilégios de ALTER, CREATE e INSERT para a tabela. Mudar o nome de uma tabela requer os privilégios de ALTER e DROP na tabela antiga, e ALTER, CREATE e INSERT na nova tabela.

Para evitar a perda acidental de dados, ALTER TABLE não pode ser usado para mudar o mecanismo de armazenamento de uma tabela para MERGE ou BLACKHOLE.

Você pode emitir um comando com múltiplas modificações ADD, ALTER, DROP e CHANGE em uma única instrução ALTER TABLE, separados por vírgulas. Esta é uma extensão do MySQL ao SQL padrão, que só permite uma cláusula de modificação por instrução ALTER TABLE. Por exemplo, para dropar várias colunas em uma única instrução, utilize o seguinte:

ALTER TABLE DROP COLUMN t2 c, DROP COLUMN d;

CHANGE nome_campo, DROP nome_campo e DROP INDEX são extensões do MySQL ao SQL padrão. A palavra COLUMN é opcional e pode ser omitida. Vejam alguns exemplos do ALTER TABLE abaixo:

Para alterar uma coluna de inteiro para TINYINT NOT NULL (deixando o nome da mesma), e para alterar coluna b de CHAR (10) para CHAR (20), bem como mudar o nome de b para c:

ALTER TABLE t2 MODIFY a TINYINT NOT NULL, CHANGE b c CHAR(20);

Para adicionar uma nova coluna TIMESTAMP d:

ALTER TABLE t2 ADD d TIMESTAMP;

Para adicionar um índice a uma coluna d e um UNIQUE índice para a coluna a:

ALTER TABLE t2 ADD INDEX (d), ADD UNIQUE (a);

ALTER TABLESPACE é nosso penúltimo comando. Este comando pode ser usado tanto para adicionar um novo arquivo de dados, quanto para dropar um arquivo de dados de um espaço de tabela. Uma variação do comando pode usar o **ADD DATAFILE** que permite especificar um tamanho inicial usando uma cláusula **INITIAL_SIZE**. O tamanho é medido em bytes, sendo o valor padrão de 134217728 (**128 MB**).

ALTER VIEW é o último comando de alteração desta nossa aula, lembrando que **estamos falando de DDL**. Esta declaração altera a definição de uma visão, que já deve existir no banco de dados. A sintaxe é semelhante ao comando **CREATE VIEW** e o efeito é o mesmo do **CREATE OR REPLACE VIEW**. Para executar esse comando, é necessário ter os privilégios de CREATE VIEW e DROP para a visão específica, além do privilégio leitura para cada coluna referida na instrução SELECT. ALTER VIEW é autorizada apenas para o definidor ou para os usuários com o privilégio SUPER sobre a visão.

CREATE

Passaremos agora para os comandos que iniciam com o identificador CREATE. A lista de comandos passa pelos mesmos modificares do ALTER com a adição do CREATE TRIGGER. Veja, portanto, que não existe ALTER TRIGGER na sintaxe do MySQL. As outras opções do CREATE são DATABASE, EVENT, INDEX, LOGFILE GROUP, FUNCTION, PROCEDURE, SERVER, TABLE, TABLESPACE e VIEW.

Dado que nosso curso está voltado especificamente para concursos, nosso objetivo é descrever apenas os assuntos com maior probabilidade de cair em provas de concurso. Passaremos agora para analisarmos os comandos DDL de CREATE INDEX, CREATE PROCEDURE, CREATE FUNCTION e CREATE TABLE. Os demais comandos não serão abordados em detalhes.

Vejamos a seguir a sintaxe do CREATE INDEX:

```
CREATE [ONLINE|OFFLINE] [UNIQUE|FULLTEXT|SPATIAL] INDEX index_name
    [index_type]
    ON tbl_name (index_col_name,...)
    [index_option]
    [algorithm_option | lock_option] ...

index_col_name:
    col_name [(length)] [ASC | DESC]

index_type:
    USING {BTREE | HASH}

index_option:
    KEY_BLOCK_SIZE [=] value
  | index_type
  | WITH PARSER parser_name
  | COMMENT 'string'

algorithm_option:
    ALGORITHM [=] {DEFAULT|INPLACE|COPY}

lock_option:
    LOCK [=] {DEFAULT|NONE|SHARED|EXCLUSIVE}
```

CREATE INDEX é mapeado para uma instrução ALTER TABLE para criar índices. CREATE INDEX não pode ser usado para criar uma chave primária; utilize ALTER TABLE para esta operação. Normalmente, você cria todos os índices de uma tabela no momento da criação da mesma com CREATE TABLE. Esta orientação é especialmente importante para tabelas que usam InnoDB, em que a chave primária determina o layout físico de linhas no arquivo de dados.

CREATE INDEX permite adicionar índices a tabelas existentes. A lista é formada por uma ou mais colunas (col1, col2, ...) e cria uma coluna de índice sobre essas colunas. Valores de chave de índice são formados concatenando os valores de colunas dadas.

Os índices podem ser criados utilizando apenas a parte inicial dos valores de coluna, usando a sintaxe nome_coluna(length) para especificar um comprimento do prefixo de índice. A instrução mostrada abaixo cria um índice usando os primeiros 10 caracteres da coluna name:

```
CREATE INDEX part_of_name ON customer (name(10));
```

Vejamos agora o que significa algumas das palavras chaves que compõem o índice. As palavras-chave ONLINE e OFFLINE estão disponíveis apenas para o MySQL Cluster; a tentativa de usar essas palavras-chave no padrão MySQL 5.6 exibe um erro de sintaxe.

Um índice **UNIQUE** cria uma restrição de tal forma que todos os valores do índice devem ser distintos. Um erro ocorre se você tentar adicionar uma nova linha com um valor de chave que corresponda a uma linha existente. Para todos os motores, um índice UNIQUE permite vários valores NULL para colunas que podem conter NULL. Se você especificar um valor de índice UNIQUE como prefixo de uma coluna, os valores da coluna devem ser exclusivos dentro do prefixo.

Índices **FULLTEXT** são suportados apenas para InnoDB e MyISAM. Estes índices podem incluir apenas colunas **CHAR**, **VARCHAR** e **TEXT**. A indexação sempre acontece sobre a coluna inteira. A indexação baseada em prefixo de coluna não é suportada e qualquer comprimento prefixo é ignorado, se especificado.

Os mecanismos de armazenamento MyISAM, InnoDB, NDB, e ARCHIVE suportam colunas espaciais como POINT e GEOMETRY. No entanto, o suporte para a indexação de colunas espaciais varia entre os motores. Os índices espaciais (criados usando o SPATIAL INDEX) têm as seguintes características:

- Disponível somente para tabelas MyISAM. Especificando SPATIAL INDEX para outros mecanismos de armazenamento resulta em um erro.
- Colunas cadastradas devem ser NOT NULL.
- No MySQL 5.6, prefixo de comprimentos de coluna são proibidos. A largura total de cada coluna é indexada.

Outro parâmetro importante é o index_type. Alguns mecanismos de armazenamento permitem que você especifique um tipo de índice ao criá-lo. Os valores de tipo de índice suportados por diferentes mecanismos de armazenamento são mostrados na tabela a seguir. Se vários tipos de índices forem listados, o primeiro é o padrão caso nenhum tipo de índice seja especificado.

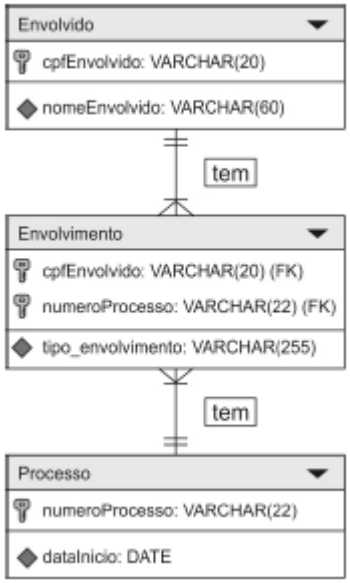
Storage Engine	Permissible Index Types
InnoDB	BTREE
MyISAM	BTREE
MEMORY/HEAP	HASH, BTREE
NDB	HASH, BTREE

Vamos agora fazer a questão abaixo antes de passarmos para o comando CREATE PROCEDUDE.

3. FCC - Analista Ministerial (MPE PB)/Analista de Sistemas/Administrador de Banco de Dados/2015

Atenção: Considere as informações a seguir para responder à questão.

Modelo Entidade-Relacionamento



Dados cadastrados na tabela:

cpfEnvolvido	nomeEnvolvido
121.134.045-01	Marcos Paulo
128.249.039-14	Maria de Fátima
131.091.431-09	André Luiz
158.245.067-12	Pedro nda Silva
160.234.074-11	João da Silva

Observação: O erro no nome Pedro nda Silva é proposital.

Na tabela Envolvido, deseja-se incluir um campo dataNascEnvolvido cujo preenchimento será obrigatório, imediatamente após o campo cpfEnvolvido. Considerando que o banco de dados e as tabelas foram criados no MySQL, deve-se utilizar, para isso, a instrução

a) ALTER TABLE Envolvido ADD COLUMN dataNascEnvolvido DATE NOT NULL AFTER cpfEnvolvido;

b) ADD COLUMN dataNascEnvolvido DATE NOT NULL FROM Envolvido AFTER cpfEnvolvido;

c) UPDATE TABLE Envolvido ADD COLUMN dataNascEnvolvido DATE NOT NULL AFTER cpfEnvolvido;

d) ALTER TABLE Envolvido ADD COLUMN dataNascEnvolvido DATE NOT NULL AFTER (cpfEnvolvido);

e) INSERT COLUMN dataNascEnvolvido DATE NOT NULL AFTER cpfEnvolvido FROM Envolvido;

Comentários: ALTER TABLE altera a estrutura de uma tabela. Por exemplo, você pode adicionar ou excluir colunas, criar ou destruir índices, alterar o tipo de colunas existentes ou renomear colunas ou a própria tabela. Você também pode alterar características, como o mecanismo de armazenamento usado para a tabela ou o comentário da tabela.

Para usar ALTER TABLE, você precisa de privilégios ALTER, CREATE e INSERT para a tabela. Renomear uma tabela requer ALTER e DROP na tabela antiga, ALTER, CREATE e INSERT na nova tabela.

Seguindo o nome da tabela, especifique as alterações a serem feitas. Se nenhum for dado, ALTER TABLE não faz nada.

Múltiplas cláusulas ADD, ALTER, DROP e CHANGE são permitidas em uma única instrução ALTER TABLE, separadas por vírgulas. Esta é uma extensão do MySQL para SQL padrão, que permite apenas uma de cada cláusula por instrução ALTER TABLE. Por exemplo, para adicionar várias colunas em uma única instrução, faça o seguinte:

ALTER TABLE t2 ADD COLUMN c, ADD COLUMN d;

Segue a sintaxe, considerando o enunciado:

```
ALTER TABLE tbl_name
    [alter_specification [, alter_specification] ...]
    [partition_options]
alter_specification:
    table_options
  | ADD [COLUMN] col_name column_definition
        [FIRST | AFTER col_name]
```

Temos como gabarito da questão a letra a):

ALTER TABLE Envolvido ADD COLUMN dataNascEnvolvido DATE NOT NULL AFTER cpfEnvolvido

Gabarito: A

Vamos tratar agora do CREATE PROCEDURE. Primeiramente, observe a sintaxe do comando abaixo. Apresentamos também a sintaxe do CREATE FUNCTION.

```
CREATE
    [DEFINER = { user | CURRENT_USER }]
    PROCEDURE sp_name ([proc_parameter[,...]])
    [characteristic ...] routine_body

CREATE
    [DEFINER = { user | CURRENT_USER }]
    FUNCTION sp_name ([func_parameter[,...]])
    RETURNS type
    [characteristic ...] routine_body

proc_parameter:
    [ IN | OUT | INOUT ] param_name type

func_parameter:
    param_name type

type:
    Any valid MySQL data type

characteristic:
    COMMENT 'string'
  | LANGUAGE SQL
  | [NOT] DETERMINISTIC
  | { CONTAINS SQL | NO SQL | READS SQL DATA | MODIFIES SQL DATA }
  | SQL SECURITY { DEFINER | INVOKER }

routine_body:
    Valid SQL routine statement
```

Estas declarações acima criam rotinas armazenadas. Por padrão, a rotina está associada ao banco de dados padrão. Para associar a rotina explicitamente a um determinado banco de dados, especifique o nome da procedure como db_name.sp_name ao criá-la.

Para invocar um procedimento armazenado, utilize a instrução CALL. Para chamar uma função armazenada, você pode referenciá-la em uma expressão. A função retorna um valor durante a avaliação da expressão.

CREATE PROCEDURE e CREATE FUNCTION exigem o privilégio CREATE ROUTINE. Eles também podem requerer o privilégio SUPER, dependendo do valor DEFINER. Se o log binário estiver habilitado, CREATE FUNCTION pode requerer o privilégio SUPER. Por padrão, MySQL concede automaticamente os privilégios de ALTER ROUTINE e EXECUTE para o criador da rotina. Este comportamento pode ser alterado, desativando a variável de sistema **automatic_sp_privileges**.

As cláusulas DEFINER e SQL SECURITY especificam o contexto de segurança utilizado para verificar os privilégios de acesso em tempo de execução da rotina.

Se o nome da rotina é o mesmo que o nome de uma função interna do SQL, um erro de sintaxe ocorre ao menos que você use um espaço após o nome e a definição da rotina ou invocá-la mais tarde. Por esta razão, evite usar os nomes de funções SQL existentes para suas próprias rotinas armazenadas. O modo IGNORE_SPACE SQL se aplica às funções internas, não para rotinas armazenadas. É sempre permitido ter espaços depois de um nome de rotina armazenada, independentemente do IGNORE_SPACE estar ativado.

A lista de parâmetros entre parênteses deve estar sempre presente. Se não houver nenhum parâmetro, deve ser usada uma lista de parâmetros vazia (). Os nomes de parâmetros não são *case sensitive*.

Cada parâmetro é definido como IN por padrão. Para especificar outro tipo de parâmetro, use a palavra-chave OUT ou INOUT antes do nome do parâmetro. A especificação de um parâmetro como IN, OUT ou INOUT só é válida para uma PROCEDURE. Para função, os parâmetros são sempre categorizados como IN.

Agora falaremos do último comando, o **CREATE TABLE**. As possíveis variações sintáticas do comando ocupam quase duas páginas no manual de uso. Optamos por apresentar abaixo apenas a parte principal do comando:

```
CREATE [TEMPORARY] TABLE [IF NOT EXISTS] tbl_name
    { LIKE old_tbl_name | (LIKE old_tbl_name) }

create_definition:
    col_name column_definition
  | [CONSTRAINT [symbol]] PRIMARY KEY [index_type] (index_col_name,...)
      [index_option] ...
  | {INDEX|KEY} [index_name] [index_type] (index_col_name,...)
      [index_option] ...
  | [CONSTRAINT [symbol]] UNIQUE [INDEX|KEY]
      [index_name] [index_type] (index_col_name,...)
      [index_option] ...
  | {FULLTEXT|SPATIAL} [INDEX|KEY] [index_name] (index_col_name,...)
      [index_option] ...
  | [CONSTRAINT [symbol]] FOREIGN KEY
      [index_name] (index_col_name,...) reference_definition
  | CHECK (expr)

column_definition:
    data_type [NOT NULL | NULL] [DEFAULT default_value]
      [AUTO_INCREMENT] [UNIQUE [KEY] | [PRIMARY] KEY]
      [COMMENT 'string']
      [COLUMN_FORMAT {FIXED|DYNAMIC|DEFAULT}]
      [STORAGE {DISK|MEMORY|DEFAULT}]
      [reference_definition]
```

Vejam que você pode usar a palavra-chave TEMPORARY ao criar uma tabela. Uma tabela temporária é visível apenas para a sessão atual. Ela é descartada automaticamente quando a sessão é encerrada. Isso significa que duas sessões diferentes podem usar o mesmo nome de tabela temporária, não gerando conflito uma com a outra ou com tabelas não temporárias existentes com o mesmo nome.

Observem que é possível definir ainda as restrições de integridade e alguns parâmetros para os tipos de dados. Esses tipos de dados podem ser qualquer um daqueles que tratamos anteriormente no nosso curso. As restrições podem ser de chave primária, estrangeira, not null e unique. Podemos definir os tipos ainda como auto incrementado. Tente analisar a sintaxe do comando acima e procure as palavras-chaves para as especificidades que acabamos de comentar.

DROP

O DROP opera sobre a mesma lista de objetos do CREATE, nada mais lógico! Tudo que é criado dentro do banco de dados pode ser apagado. A lista inclui todas as opções que vimos: DATABASE,

EVENT, FUNCTION, INDEX, LOGFILE GROUP, PROCEDURE, SERVER, TABLE, TRIGGER, TABLESPACE e VIEW.

DROP DATABASE apaga todas as tabelas no banco de dados e exclui o banco de dados. Tenha muito cuidado com esta afirmação! Para usar DROP DATABASE, é necessário o privilégio DROP no banco de dados. DROP SCHEMA é um sinônimo para DROP DATABASE. IF EXISTS é utilizado para prevenir a ocorrência de erros, se o banco de dados a ser apagado não existe. Se o banco de dados padrão é descartado, o banco de dados padrão fica definido como unset (a função DATABASE() retorna NULL). Veja a sintaxe do DROP DATABASE abaixo:

```
DROP {DATABASE | SCHEMA} [IF EXISTS] db_name
```

A declaração **DROP EVENT** exclui o evento nomeado (event_name). O evento imediatamente deixa de ser ativo e é excluído completamente do servidor.

O comando **DROP FUNCTION** é usado para apagar funções armazenadas e funções definidas pelo usuário, conhecidas como user-defined functions (UDFs).

DROP INDEX apaga um índice da tabela. Esta declaração é mapeada para uma instrução ALTER TABLE para fazer o DROP do índice. Para excluir o índice de uma chave primária, em que o nome do índice é sempre descrito como PRIMARY, deve ser especificado como um identificador entre aspas porque PRIMARY é uma palavra reservada: DROP INDEX `PRIMARY` ON t; .

A declaração **DROP LOGFILE GROUP** apaga o grupo de arquivo de log. O grupo de arquivos de log já deve existir ou ocorrerá um erro.

O comando **DROP PROCEDURE** é usado para apagar um procedimento armazenado ou função. Ou seja, a rotina especificada é removida do servidor. Você deve ter o privilégio ALTER ROUTINE para a rotina. Se a variável de sistema **automatic_sp_privileges** está habilitada, o privilégio de executar e apagar são concedidos automaticamente ao criador da rotina.

DROP SERVER apaga a definição de um servidor. A linha correspondente na tabela de mysql.servers é excluída. Esta declaração exige o privilégio SUPER.

DROP TABLE remove uma ou mais tabelas. Você deve ter o privilégio DROP para cada tabela. Todos os dados da tabela e a definição da tabela são removidos, por isso tenha cuidado com este comando! Se qualquer uma das tabelas da lista de argumentos não existe, o MySQL retorna um erro indicando que essas tabelas não foram apagadas, mas dropa todas as tabelas na lista que existem.

Para uma tabela de partição, DROP TABLE remove permanentemente a definição da tabela, todas as suas partições e todos os dados que foram armazenados nessas partições. Ele também remove o arquivo de definição de particionamento (.par) associado com a tabela descartada.

Você pode utilizar o IF EXISTS para prevenir a ocorrência de um erro para tabelas que não existem. Uma nota é gerada para cada tabela inexistente quando se utiliza o IF EXISTS. RESTRICT e CASCADE são permitidos para fazer uma portabilidade mais fácil. Veja a sintaxe do comando abaixo:

```
DROP [TEMPORARY] TABLE [IF EXISTS]
    tbl_name [, tbl_name] ...
    [RESTRICT | CASCADE]
```

DROP TABLESPACE apaga um tablespace que foi anteriormente definida pelo comando CREATE TABLESPACE.

A declaração **DROP TRIGGER** apaga um gatilho. O banco de dados ou nome do esquema é opcional. Se o esquema for omitido, o gatilho será removido do esquema padrão. DROP TRIGGER requer o privilégio TRIGGER para a tabela associada com o gatilho. Veja a sintaxe abaixo:

```
DROP TRIGGER [IF EXISTS] [schema_name.]trigger_name
```

DROP VIEW remove uma ou mais VIEWs. Você deve ter o privilégio DROP sobre cada visão. Se qualquer uma das visões na lista de argumentos não existir, o MySQL retorna um erro indicando pelo nome quais não existem e não conseguiu apagar, mas também deleta todas as VIEWS que existem na lista.

RENAME TABLE

```
RENAME TABLE tbl_name TO new_tbl_name
[, tbl_name2 TO new_tbl_name2]...
```

Esta declaração renomeia uma ou mais tabelas. A operação de *rename* é feita atomicamente, o que significa que nenhuma outra sessão pode acessar qualquer uma das tabelas enquanto a ação está sendo executada. Por exemplo, uma tabela chamada tabela_antiga pode ser renomeada para nova_tabela como mostrado aqui:

```
RENAME TABLE tabela_antiga TO nova_tabela;
```

Esta declaração é equivalente à seguinte instrução ALTER TABLE:

```
ALTER TABLE RENAME new_table OLD_TABLE;
```

Se a declaração renomeia mais de uma tabela, as operações de mudança de nome são feitas da esquerda para a direita. Se você quiser trocar os nomes de tabela, você pode fazê-lo como a seguir (supondo que tmp_table ainda não existe):

```
RENAME TABLE tabela_antiga TO tmp_table,
new_table TO tabela_antiga,
tmp_table TO nova_tabela;
```

O MySQL verifica o nome da tabela de destino antes de verificar se a tabela de origem existe. Por exemplo, se nova_tabela já existe e a tabela_antiga não, a seguinte declaração de falha é mostrada:

```
mysql> SHOW TABLES;
+-----------------+
| Tables_in_mydb |
+-----------------+
| table_a         |
+-----------------+
1 row in set (0.00 sec)
mysql> RENAME TABLE table_b TO table_a;
ERROR 1050 (42S01): Table 'table_a' already exists
```

TRUNCATE

TRUNCATE TABLE esvazia uma tabela completamente. Ela exige o privilégio de DROP. Logicamente, TRUNCATE TABLE é semelhante a uma instrução de DELETE que exclui todas as linhas, ou a uma sequência de DROP TABLE e CREATE TABLE. Para atingir um alto desempenho, o comando ignora o método DML de exclusão de dados. Assim, o comando não pode ser revertido, não aciona gatilhos de ON DELETE e não pode ser realizado em tabelas InnoDB com relações de chave estrangeira entre pais e filhos.

Embora TRUNCATE TABLE seja semelhante ao DELETE, ele é classificado como uma instrução DDL em vez de uma instrução DML.

DATA MANIPULATION LANGUAGE (DML)

Após passarmos pelos comandos DDL, agora voltaremos nossas atenções para a parte da linguagem que manipula os dados. São nove comandos sobre os quais apontaremos nossas baterias, sempre focando nos aspectos chave de cada um deles.

CALL

A instrução CALL chama um procedimento armazenado que foi definido anteriormente com o CREATE PROCEDURE. Procedimentos armazenados que não têm argumentos podem ser chamados sem parênteses, isto é, CALL p e CALL p() são equivalentes.

CALL pode passar valores de retorno para quem o chamou utilizando os parâmetros que são declarados como parâmetros OUT ou INOUT. Quando o procedimento retorna, um programa cliente também pode obter o número de linhas afetadas pela declaração final executada dentro da rotina. No nível de SQL, é possível chamar a função ROW_COUNT(); a partir da API C, você deve utilizar a função mysql_affected_rows ().

Para obter um valor de um procedimento usando um parâmetro OUT ou INOUT, podemos passar o parâmetro por meio de uma variável de usuário e, em seguida, verificar o valor da variável após o retorno do procedimento. Se você está chamando o procedimento de dentro de outro procedimento armazenado ou função, você também pode passar um parâmetro de rotina ou variável rotina local como um parâmetro IN ou INOUT.

Para um parâmetro INOUT, é preciso inicializar o seu valor antes de passá-lo para o procedimento. O procedimento a seguir tem um parâmetro OUT, que o procedimento define para a versão atual do servidor, e um valor de INOUT, que representa os incrementos feitos por um procedimento em seu valor atual:

```
CREATE PROCEDURE p (OUT ver_param VARCHAR(25), INOUT incr_param INT)
BEGIN
  # Set value of OUT parameter
  SELECT VERSION() INTO ver_param;
  # Increment value of INOUT parameter
  SET incr_param = incr_param + 1;
END;
```

Antes de chamar o procedimento, inicialize a variável a ser passada como parâmetro. Depois de chamar o procedimento, os valores das duas variáveis terão sido determinados ou alterados:

```
mysql> SET @increment = 10;
mysql> CALL p(@version, @increment);
mysql> SELECT @version, @increment;
+--------------+------------+
| @version     | @increment |
+--------------+------------+
| 5.5.3-m3-log |         11 |
+--------------+------------+
```

DELETE

A instrução DELETE exclui linhas de uma tabela e retorna o número de linhas excluídas. Para verificar o número de linhas excluídas, chame a função ROW_COUNT (). Veja a sintaxe do comando DELETE para uma única tabela:

```
DELETE [LOW_PRIORITY] [QUICK] [IGNORE] FROM tbl_name
    [PARTITION (partition_name,...)]
    [WHERE where_condition]
    [ORDER BY ...]
    [LIMIT row_count]
```

As condições na cláusula WHERE opcional identifica quais linhas eliminar. Sem a cláusula WHERE, todas as linhas são excluídas. **Where_condition** é uma expressão que avalia para *true* ou *false* para cada linha a ser apagada.

Se a cláusula ORDER BY é especificada, as linhas serão excluídas na ordem especificada. A cláusula LIMIT coloca um limite para o número de linhas que podem ser excluídas. Estas cláusulas aplicam-se a exclusões de uma única tabela, mas não a exclusões em múltiplas tabelas.

Você precisa do privilégio DELETE em uma tabela para excluir linhas da mesma. Também é preciso do privilégio de SELECT para todas as colunas que são lidas, tais como os citados na cláusula WHERE.

DO

DO executa as expressões, mas não retornam nenhum resultado. Na maioria dos aspectos, DO é uma abreviação para SELECT expr, ..., mas tem a vantagem de ser um pouco mais rápido quando você não se preocupa com o resultado.

DO é útil principalmente com funções que têm efeitos colaterais, tais como RELEASE_LOCK(). Vejam um exemplo. Esta instrução SELECT faz uma pausa, mas também produz um conjunto de resultados:

```
mysql> SELECT SLEEP(5);
+----------+
| SLEEP(5) |
+----------+
|        0 |
+----------+
1 row in set (5.02 sec)
```

DO, por outro lado, faz uma pausa sem produzir um conjunto de resultados:

```
mysql> DO SLEEP(5);
Query OK, 0 rows affected (4.99 sec)
```

Isto pode ser útil, por exemplo, em uma função armazenada ou gatilho, que proíbe declarações que produzem conjuntos de resultados. DO somente executa as expressões. Ele não pode ser usado em todos os casos em que SELECT pode ser usado. Por exemplo, DO id FROM t1 é inválido, porque faz referência a uma tabela (id).

HANDLER

A instrução HANDLER fornece acesso direto à interface do mecanismo de armazenamento de tabela. HANDLER está disponível para as tabelas InnoDB e MyISAM. Vejam a figura a seguir:

```
HANDLER tbl_name OPEN [ [AS] alias]

HANDLER tbl_name READ index_name { = | <= | >= | < | > } (value1,value2,...)
    [ WHERE where_condition ] [LIMIT ... ]
HANDLER tbl_name READ index_name { FIRST | NEXT | PREV | LAST }
    [ WHERE where_condition ] [LIMIT ... ]
HANDLER tbl_name READ { FIRST | NEXT }
    [ WHERE where_condition ] [LIMIT ... ]

HANDLER tbl_name CLOSE
```

A instrução HANDLER ...OPEN abre uma tabela, tornando-a acessível usando HANDLER...READ. Este objeto da tabela não é compartilhado por outras sessões e não é fechado até que a sessão chame o HANDLER...CLOSE ou a sessão termine. Se você abrir a tabela usando um alias, outras referências para a tabela aberta feitas pelo HANDLER devem usar o alias em vez do nome da tabela.

INSERT

O INSERT insere novos registros em uma tabela existente. Os INSERT ... VALUES e INSERT ... SET são baseados na inserção de linhas com base em valores especificados explicitamente. A forma INSERT ... SELECT insere linhas selecionadas de outra tabela ou tabelas. Vejam as duas primeiras formas abaixo:

```
INSERT [LOW_PRIORITY | DELAYED | HIGH_PRIORITY] [IGNORE]
    [INTO] tbl_name
    [PARTITION (partition_name,...)]
    [(col_name,...)]
    {VALUES | VALUE} ({expr | DEFAULT},...),(...),...
    [ ON DUPLICATE KEY UPDATE
      col_name=expr
        [, col_name=expr] ... ]

INSERT [LOW_PRIORITY | DELAYED | HIGH_PRIORITY] [IGNORE]
    [INTO] tbl_name
    [PARTITION (partition_name,...)]
    SET col_name={expr | DEFAULT}, ...
    [ ON DUPLICATE KEY UPDATE
      col_name=expr
```

No MySQL 5.6.2 ou posterior, ao inserir em uma tabela de partição, você pode controlar quais partições e subpartições aceitam novas linhas. A opção de PARTITION leva uma lista dos nomes de uma ou mais partições ou subpartições (ou ambos) da tabela, separados por vírgulas. Se qualquer uma das linhas a serem inseridas por uma dada instrução INSERT não corresponder a uma das

partições listadas, a instrução INSERT falha com o erro: **found a row not matching the given partition set**.

Você pode usar REPLACE em vez de INSERT para sobrescrever linhas antigas. REPLACE funciona exatamente como o INSERT, exceto que, se uma linha antiga na tabela tem o mesmo valor que uma nova linha para uma chave primária ou um índice exclusivo, a velha linha é excluída antes da nova linha ser inserida.

Se você usar a palavra-chave DELAYED, o servidor coloca a linha ou linhas a serem inseridas em um buffer, e o cliente que emite a instrução INSERT DELAYED pode então continuar imediatamente. Se a tabela está em utilização, o servidor mantém as linhas. Quando a tabela for liberada, o servidor começa a inserir linhas, verificando periodicamente para ver se existem quaisquer novos pedidos de leitura para a tabela. Se houver, a fila de linhas atrasadas é suspensa até que a tabela fique livre de novo.

Outra opção é usar a palavra-chave LOW_PRIORITY, em que a execução do INSERT é atrasada até que nenhum outro cliente esteja lendo da tabela. Isto inclui outros clientes que começaram a ler enquanto os clientes existentes já estão lendo e a instrução INSERT LOW_PRIORITY está esperando. É possível, portanto, para um cliente que emite uma instrução INSERT LOW_PRIORITY esperar por um tempo muito longo (ou mesmo para sempre) em um ambiente de leitura pesada. Vamos agora fazer a questão abaixo.

3. CESPE - Oficial Técnico de Inteligência/ Área 9/2018

Julgue o próximo item, a respeito de conceitos e comandos PostgreSQL e MySQL.

No MySQL, a instrução CALL é usada para chamar os procedimentos armazenados.

Certo

Errado

Comentários: A instrução CALL invoca um procedimento armazenado que foi definido anteriormente com CREATE PROCEDURE.

Procedimentos armazenados que não recebem argumentos podem ser chamados sem parênteses. Ou seja, CALL p () e CALL p são equivalentes.

CALL pode retornar valores ao seu chamador usando parâmetros que são declarados como parâmetros OUT ou INOUT. Quando o procedimento retorna, um programa cliente também pode obter o número de linhas afetadas para a instrução final executada dentro da rotina: No nível SQL, chame a função ROW_COUNT (); da API C, chame a função mysql_affected_rows ().

Veja a sintaxe do comando CALL:

CALL sp_name([parameter[,...]])

CALL sp_name[()]

Gabarito: C

LOAD DATA INFILE

O LOAD DATA INFILE lê linhas de um arquivo de texto em uma tabela numa velocidade muito alta. Esta função é adequada para trabalhos que envolvam grande volume de dados. LOAD DATA INFILE é um complemento do SELECT ... INTO OUTFILE. Para **gravar** dados de uma tabela para um arquivo, use SELECT ... INTO OUTFILE. Para **ler** o arquivo de volta em uma tabela, use LOAD DATA INFILE. A sintaxe das cláusulas FIELDS e LINES é a mesma para ambas as declarações. Ambas as cláusulas são opcionais, mas FIELDS deve preceder LINES, se ambos são especificados. Veja a sintaxe do comando:

```
LOAD DATA [LOW_PRIORITY | CONCURRENT] [LOCAL] INFILE 'file_name'
    [REPLACE | IGNORE]
    INTO TABLE tbl_name
    [PARTITION (partition_name,...)]
    [CHARACTER SET charset_name]
    [{FIELDS | COLUMNS}
        [TERMINATED BY 'string']
        [[OPTIONALLY] ENCLOSED BY 'char']
        [ESCAPED BY 'char']
    ]
    [LINES
        [STARTING BY 'string']
        [TERMINATED BY 'string']
    ]
    [IGNORE number {LINES | ROWS}]
    [(col_name_or_user_var,...)]
    [SET col_name = expr,...]
```

Se o volume de dados for significativamente grande e você estiver preocupado em manter a disponibilidade do servidor a vários outros clientes, use a diretiva ***LOW_PRIORITY*** na declaração — o seu efeito é "segurar" a execução até que não haja outros clientes lendo dados na tabela (isto só afeta *storage engines*, como MyISAM, MEMORY e MERGE).

Você pode usar diretiva **CONCURRENT** em tabelas MyISAM, que permite que outros clientes façam requisições à mesma tabela em que o LOAD DATA está trabalhando. Esta opção afeta a performance da tarefa de importação de dados, mesmo que nenhuma outra *thread* esteja fazendo uso da tabela simultaneamente.

Você também pode carregar arquivos de dados usando o utilitário **mysqlimport**. Ele opera enviando uma declaração LOAD DATA INFILE para o servidor. A opção --local faz com que *mysqlimport* leia arquivos de dados a partir do host cliente. Você pode especificar a opção --compress para conseguir um melhor desempenho sobre redes lentas, se o cliente e o servidor suportam o protocolo de compressão.

LOAD XML

A instrução LOAD XML lê dados de um arquivo XML em uma tabela. O nome do arquivo deve ser passado como uma string literal. A opção tagname da cláusula ROWS IDENTIFIED BY também deve ser passada como uma string literal e deve ser em tags, ou seja, (< e >). Veja a sintaxe do comando:

```
LOAD XML [LOW_PRIORITY | CONCURRENT] [LOCAL] INFILE 'file_name'
    [REPLACE | IGNORE]
    INTO TABLE [db_name.]tbl_name
    [CHARACTER SET charset_name]
    [ROWS IDENTIFIED BY '<tagname>']
    [IGNORE number {LINES | ROWS}]
    [(field_name_or_user_var,...)]
    [SET col_name = expr,...]
```

LOAD XML age como o complemento para o cliente executar o mysql com o modo de saída XML, ou seja, iniciar o cliente com a opção --xml. Para gravar dados de uma tabela para um arquivo XML, você pode invocar o cliente mysql com as opções --xml e -e do shell do sistema, como mostrado abaixo:

```
shell> mysql --xml -e 'SELECT * FROM mydb.mytable' > file.xml
```

SELECT

SELECT é usado para recuperar as linhas específicas a partir de uma ou mais tabelas e pode incluir declarações de UNION e subconsultas. As mais comumente cláusulas usadas em instruções SELECT são tratadas abaixo. Antes, porém, vejamos a sintaxe do comando:

```
SELECT
    [ALL | DISTINCT | DISTINCTROW ]
      [HIGH_PRIORITY]
      [STRAIGHT_JOIN]
      [SQL_SMALL_RESULT] [SQL_BIG_RESULT] [SQL_BUFFER_RESULT]
      [SQL_CACHE | SQL_NO_CACHE] [SQL_CALC_FOUND_ROWS]
    select_expr [, select_expr ...]
    [FROM table_references
      [PARTITION partition_list]
    [WHERE where_condition]
    [GROUP BY {col_name | expr | position}
      [ASC | DESC], ... [WITH ROLLUP]]
    [HAVING where_condition]
    [ORDER BY {col_name | expr | position}
      [ASC | DESC], ...]
    [LIMIT {[offset,] row_count | row_count OFFSET offset}]
    [PROCEDURE procedure_name(argument_list)]
    [INTO OUTFILE 'file_name'
        [CHARACTER SET charset_name]
        export_options
      | INTO DUMPFILE 'file_name'
      | INTO var_name [, var_name]]
    [FOR UPDATE | LOCK IN SHARE MODE]]
```

- Cada expressão de seleção (select_expr) indica as colunas que você deseja recuperar. Deve haver pelo menos um select_expr.
- Tabelas_ref indica a tabela ou tabelas das quais vamos recuperar linhas.

> • A partir do MySQL 5.6.2, o SELECT oferece suporte para seleção de partição explícita usando a palavra-chave PARTITION com uma lista de partições ou subpartições, seguida pelo nome da tabela. Neste caso, as linhas são selecionadas apenas das partições listadas, e quaisquer outras partições da tabela são ignoradas.

No MySQL 5.6.6 e posterior, o comando SELECT ... PARTITION a partir de tabelas usando mecanismos de armazenamento, tais como MyISAM, que realizam bloqueios no nível de tabela (e, portanto, bloqueio de partição), podem bloquear somente as partições ou subpartições nomeadas pelas opções do PARTITION.

> • A cláusula WHERE, se for preenchida, indica a condição ou condições que os registros devem satisfazer para serem selecionados. where_condition é uma expressão que avalia para como verdadeira (true) cada linha a ser selecionado. A declaração seleciona todas as linhas se não houver nenhuma cláusula WHERE.

Na expressão WHERE, você pode usar qualquer uma das funções e operadores que o MySQL suporta, exceto para funções de agregação.

SELECT também pode ser utilizado para recuperar linhas calculadas sem referência a qualquer tabela.

Por exemplo:

```
mysql> SELECT 1 + 1;
-> 2
```

Você tem permissão para especificar DUAL como um nome tabela fictícia em situações em que não há tabelas referenciadas:

```
mysql> SELECT 1 + 1 FROM DUAL;
-> 2
```

Em geral, as cláusulas usadas devem ser fornecidas exatamente na ordem mostrada na descrição da sintaxe. Por exemplo, uma cláusula HAVING deve vir depois de qualquer cláusula GROUP BY e antes de qualquer cláusula ORDER BY. A exceção é a cláusula INTO, que pode aparecer como mostrado na descrição da sintaxe ou imediatamente após a lista de atributos definidas em *select_expr*. Vamos agora fazer uma questão sobre o assunto.

4. CESPE - Auditor Municipal de Controle Interno (CGM João Pessoa)/Tecnologia da Informação/Desenvolvimento de Sistemas/2018

```
SELECT c.Nome
FROM CONTRIBUINTE as c, IMOVEL as i ORDER BY
  i.Valor_IPTU DESC LIMIT 10
```

WHERE c.CPF_CNPJ=i.CPF_CNPJ_Proprietario

Com base no trecho de código apresentado para execução pelo SGBD MySQL, julgue o item a seguir.

Quando executado, o código retornará os nomes dos dez contribuintes com maior valor atribuído de IPTU, considerando a soma dos valores de IPTU de todos os imóveis registrados nos nomes desses contribuintes.

Certo

Errado

Comentários: SELECT é usado para recuperar linhas selecionadas de uma ou mais tabelas e pode incluir instruções e subconsultas UNION.

Segue a sintaxe:

```
SELECT
    [ALL | DISTINCT | DISTINCTROW ]
      [HIGH_PRIORITY]
      [STRAIGHT_JOIN]
      [SQL_SMALL_RESULT] [SQL_BIG_RESULT] [SQL_BUFFER_RESULT]
      SQL_NO_CACHE [SQL_CALC_FOUND_ROWS]
    select_expr [, select_expr ...]
    [FROM table_references
      [PARTITION partition_list]
    [WHERE where_condition]
    [GROUP BY {col_name | expr | position}, ... [WITH ROLLUP]]
    [HAVING where_condition]
    [WINDOW window_name AS (window_spec)
        [, window_name AS (window_spec)] ...]
[ORDER BY {col_name | expr | position}
      [ASC | DESC], ... [WITH ROLLUP]]
    [LIMIT {[offset,] row_count | row_count OFFSET offset}]
    [INTO OUTFILE 'file_name'
        [CHARACTER SET charset_name]
        export_options
      | INTO DUMPFILE 'file_name'
      | INTO var_name [, var_name]]
```

[FOR {UPDATE | SHARE} [OF tbl_name [, tbl_name] ...] [NOWAIT | SKIP LOCKED]

| LOCK IN SHARE MODE]]

A questão está errada, pois o resultado não será a soma dos valores de IPTU de todos os imóveis registrados em nome desses contribuintes. O resultado será os nomes dos dez contribuintes com maior valor atribuído de IPTU, considerando valores individuais dos imóveis.

A questão estaria correta se o comando SELECT tivesse também uma função agregada SUM para somar todos os valores de IPTU por contribuinte e, em seguida, ordenar os dez contribuintes com maior valor atribuído de IPTU.

Gabarito: E

UPDATE

O comando UPDATE pode ser executado sobre uma ou mais tabelas. Segue abaixo a sintaxe do comando, quando executado apenas sobre uma tabela:

```
UPDATE [LOW_PRIORITY] [IGNORE] table_reference
    SET col_name1={expr1|DEFAULT} [, col_name2={expr2|DEFAULT}] ...
    [WHERE where_condition]
    [ORDER BY ...]
    [LIMIT row_count]
```

Para a sintaxe acima, o UPDATE atualiza uma coluna das linhas existentes na tabela com novos valores. A cláusula SET indica quais colunas modificar e os valores que devem ser dados. Cada valor pode ser passado como uma expressão, ou utilizando a palavra-chave DEFAULT para redefinir uma coluna explicitamente com seu valor padrão. A cláusula WHERE, se utilizada, especifica as condições que identificam quais linhas devemos atualizar. Sem cláusula WHERE, todas as linhas são atualizadas. Se a cláusula ORDER BY for especificada, as linhas serão atualizadas na ordem especificada. A cláusula LIMIT coloca um limite para o número de linhas que podem ser atualizadas.

Para a sintaxe de múltiplas tabelas, o UPDATE atualiza linhas em cada tabela de tabelas_ref que satisfaça as condições. Cada linha correspondente é atualizada uma vez, mesmo que ela corresponda às condições múltiplas vezes. Para obter a sintaxe de múltiplas tabelas, ORDER BY e LIMIT não podem ser usados. Segue a sintaxe do comando:

```
UPDATE [LOW_PRIORITY] [IGNORE] table_references
    SET col_name1={expr1|DEFAULT} [, col_name2={expr2|DEFAULT}] ...
    [WHERE where_condition]
```

Para tabelas particionadas, ambas as formas, uma única ou múltiplas tabelas podem ser utilizadas para apoiar o uso do UPDATE. Esta opção recebe uma lista de uma ou mais partições ou subpartições (ou ambos). Somente as partições (ou subpartições) listadas são verificadas. Uma linha que não faça parte de alguma dessas partições ou subpartições não é atualizada, mesmo que corresponda à opção definida na where_condition. Hora da questão!

5. BANCA: FGV ANO: 2013 ÓRGÃO: AL-MT PROVA: ANALISTA DE SISTEMAS - BANCO DE DADOS

As alternativas a seguir apresentam declarações DML da linguagem SQL utilizadas pelo MySQL versão 5.1 ou superior, à exceção de uma. Assinale-a.

A DO e HANDLER.

B TRUNCATE e DO.

C REPLACE e INSERT.

D CALL e DO.

E LOAD DATA e CALL.

Comentário: Conforme comentamos quando estudamos o comando TRUNCATE do MySQL, vimos que ele é considerado um comando DDL. Logo, encontramos nossa resposta na alternativa B. Vejam que as demais opções estão todas na lista de nove comandos DML que acabamos de tratar do ponto de vista teórico.

Gabarito: B.

COMANDOS PARA TRANSAÇÕES

O MySQL suporta transações locais (dentro de uma determinada sessão do cliente) através dos comandos **SET autocommit**, **START TRANSACTION**, **COMMIT** e **ROLLBACK**. Estas declarações fornecem controle sobre o uso de transações:

- **START TRANSACTION** ou **BEGIN** inicia uma nova transação.
- **COMMIT** efetiva a transação corrente, fazendo suas mudanças permanentes.
- **ROLLBACK** reverte a transação atual, cancelando suas alterações.
- **SET autocommit** desabilita ou habilita o modo de confirmação automático padrão para a sessão atual.

Por padrão, MySQL é executado com o modo *autocommit* habilitado. Isto significa que assim que você executar uma instrução que atualiza ou modifica uma tabela, o MySQL armazena a atualização no disco para torná-la permanente. Neste momento, a alteração não pode mais ser revertida.

Segue abaixo a sintaxe dos comandos:

```
START TRANSACTION
    [transaction_characteristic [, transaction_characteristic] ...]

transaction_characteristic:
    WITH CONSISTENT SNAPSHOT
  | READ WRITE
  | READ ONLY

BEGIN [WORK]
COMMIT [WORK] [AND [NO] CHAIN] [[NO] RELEASE]
ROLLBACK [WORK] [AND [NO] CHAIN] [[NO] RELEASE]
SET autocommit = {0 | 1}
```

O InnoDB ainda suporta as instruções SQL de SAVEPOINT, ROLLBACK TO SAVEPOINT, RELEASE SAVEPOINT e a palavra-chave opcional WORK para ROLLBACK. A declaração SAVEPOINT estabelece um *savepoint* para a transação com um nome de identificação. Se a transação atual tem um *savepoint* com o mesmo nome, o velho *savepoint* é excluído e um novo é definido.

O ROLLBACK TO SAVEPOINT reverte uma transação para o *savepoint* chamado sem terminar a transação. As modificações que a transação corrente fez após o *savepoint* ser criado são desfeitas na reversão, mas o InnoDB não libera os bloqueios de linha na memória que foram armazenados depois do *savepoint*. Os *savepoints* que foram definidos em um momento posterior são excluídos.

Se o ROLLBACK TO SAVEPOINT retorna o seguinte erro: SAVEPOINT identifier does not exist, significa que o nome do savepoint solicitado não foi especificado.

3. CESPE - Analista Judiciário (STM)/Apoio Especializado/Análise de Sistemas/2018

Julgue o item que se segue, a respeito do processamento de transações e otimização de desempenho do SGBD e de consultas SQL.

No MySQL 5.6, o modo padrão de execução das transações é autocommit, o qual faz que as mudanças realizadas se tornem permanentes após a execução bem-sucedida desse comando; entretanto, esse modo será desabilitado implicitamente, se uma série de instruções for iniciada por meio do comando START TRANSACTION.

Certo

Errado

Comentários: No InnoDB, toda a atividade do usuário ocorre dentro de uma transação. Se o modo de confirmação automática estiver ativado, cada instrução SQL formará uma única transação por conta própria. Por padrão, o MySQL inicia a sessão para cada nova conexão com o autocommit habilitado, então o MySQL faz uma confirmação após cada instrução SQL se essa instrução não retornar um erro. Se uma instrução retornar um erro, o comportamento de confirmação ou reversão dependerá do erro.

Uma sessão que tenha permissão automática habilitada pode executar uma transação de várias instruções iniciando-a com uma instrução START TRANSACTION ou BEGIN explícita e terminando com uma instrução COMMIT ou ROLLBACK.

Para desabilitar o modo de autocommit implicitamente para uma única série de instruções, use a instrução START TRANSACTION. Com START TRANSACTION, o recurso de autocommit permanece desativado até que você finalize a transação com COMMIT ou ROLLBACK. O modo de confirmação automática reverte para o estado anterior.

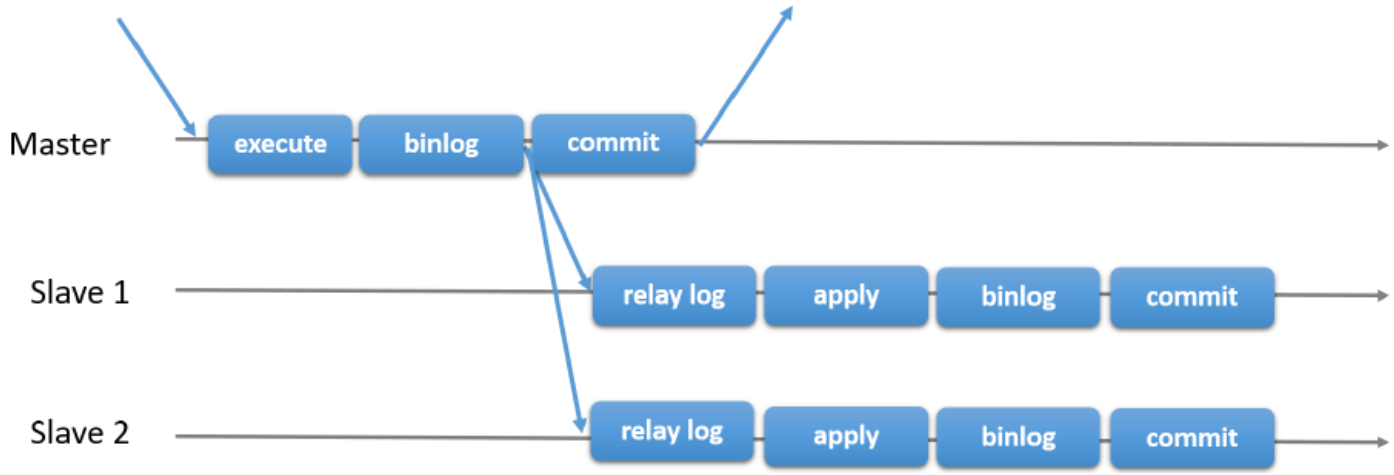
Gabarito: C

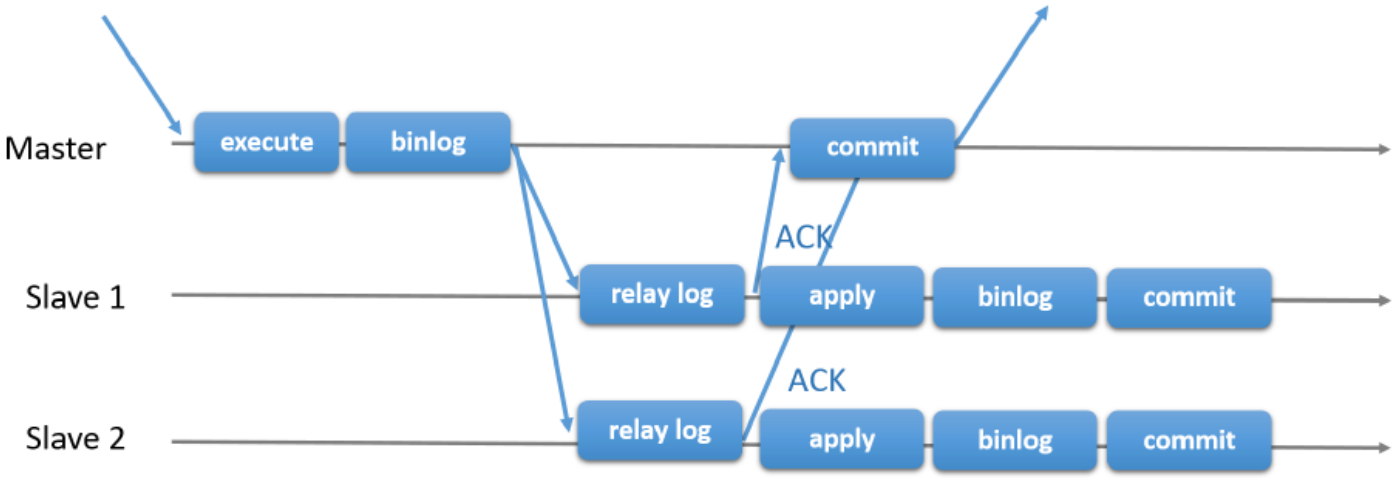
REPLICAÇÃO

A replicação pode ser controlada através da interface do SQL usando as instruções descritas nesta sessão. Um grupo de instruções controla servidores mestres, e outro grupo controla os servidores escravos.

Replicação permite que os dados de um servidor de banco de dados MySQL (o mestre) sejam replicados para um ou mais servidores de banco de dados MySQL (os escravos). Replicação é assíncrona por padrão, portanto, os escravos não precisam ser conectados permanentemente para receber atualizações do mestre. Isto significa que as atualizações podem ocorrer em conexões de longa distância e, até mesmo, através de conexões temporárias ou intermitentes, como um serviço dial-up. Dependendo da configuração, você pode replicar todos os bancos de dados, bancos de dados selecionados ou, até mesmo, tabelas selecionadas dentro de um banco de dados.

A replicação entre servidores MySQL baseia-se no mecanismo de log binário. A instância MySQL funcionando como o mestre (a fonte das mudanças de banco de dados) escreve atualizações e mudanças como "eventos" para o log binário. As informações contidas no log binário estão armazenadas em diferentes formatos de registro, de acordo com as alterações de banco de dados que estão sendo gravadas. Escravos são configurados para ler o log binário do mestre e executar os eventos no log binário no banco de dados local do escravo.





UTILITY STATEMENTS

A seguir apresentaremos quatro comandos utilitários, que ajudarão no acesso a informações sobre as estruturas e o funcionamento do MySQL.

DESCRIBE

Os comandos DESCRIBE e EXPLAIN são sinônimos. Na prática, a palavra-chave DESCRIBE é mais usada para obter informações sobre a estrutura da tabela, enquanto o EXPLAIN é usado para obter um plano de execução da consulta (ou seja, uma explicação de como o MySQL vai executar uma consulta). A discussão a seguir usa as palavras-chave DESCRIBE e EXPLAIN em conformidade com esses usos, mas o analisador MySQL trata os dois como completamente sinônimos.

```
{EXPLAIN | DESCRIBE | DESC}
    tbl_name [col_name | wild]

{EXPLAIN | DESCRIBE | DESC}
    [explain_type]
    explainable_stmt

explain_type: {
    EXTENDED
  | PARTITIONS
  | FORMAT = format_name
}

format_name: {
    TRADITIONAL
  | JSON
}

explainable_stmt: {
    SELECT statement
  | DELETE statement
  | INSERT statement
  | REPLACE statement
  | UPDATE statement
}
```

DRECRIBE é um atalho para SHOW COLUMNS. O comando também exibe informações sobre VIEWs. A descrição do comando SHOW COLUMNS fornece mais informações em suas colunas de saída. Por padrão, DESCRIBE exibe informações sobre todas as colunas da tabela. Col_name, se for passado, deve ser o nome de uma coluna na tabela. Neste caso, só exibe a informação se as colunas existirem. Wild, se for dada, é um pattern string. O comando pode conter os caracteres curinga SQL "%" e "_". Neste caso, a declaração mostra na saída apenas para as colunas com nomes correspondentes à sequência. Não há necessidade de colocar a cadeia de caracteres entre aspas, ao menos que ele contenha espaços ou outros caracteres especiais.

A instrução DESCRIBE é fornecida para compatibilidade com Oracle.

O SHOW CREATE TABLE, SHOW TABLE STATUS e declarações SHOW INDEX também fornecem informações sobre as tabelas.

EXPLAIN

A instrução EXPLAIN fornece informações sobre como o MySQL executa instruções:

- A partir do MySQL 5.6.3, foi permitido o uso do EXPLAIN para as instruções de SELECT, DELETE, INSERT, REPLACE e UPDATE. Antes do MySQL 5.6.3, SELECT era a única instrução explicável.
- Quando EXPLAIN é usado como uma instrução explicável, o MySQL exibe informações do otimizador sobre o plano de execução da instrução. Ou seja, o MySQL explicará como ele deve processar a instrução, incluindo informação sobre como as tabelas estão unidas por meio dos joins e em qual ordem.
- EXPLAIN EXTENDED pode ser usado para obter informações adicionais do plano de execução.
- EXPLAIN PARTITIONS é útil para examinar consultas envolvendo tabelas particionadas.
- A partir do MySQL 5.6.5, a opção de FORMAT pode ser usada para selecionar o formato de saída. TRADICIONAL apresenta a saída em formato tabular. Este é o padrão, se nenhuma opção de formato estiver presente no comando. Com FORMAT = JSON, a saída inclui informações estendidas e partição, e as informações em formato JSON.

Com a ajuda do EXPLAIN, você pode ver onde deve adicionar índices às tabelas. Assim, a instrução pode ser executada mais rapidamente, usando índices para encontrar linhas. Você também pode usar o EXPLAIN para verificar se o otimizador une as tabelas na melhor ordem. Para dar uma dica para o otimizador usar a uma ordem de associação correspondente à ordem na qual as tabelas são descritas na instrução SELECT, você pode começar a declaração com SELECT STRAIGHT_JOIN ao invés de apenas SELECT.

Se você tiver um problema com índices que não estão sendo usados quando você acredita que eles deveriam ser, execute o ANALYZE TABLE para atualizar estatísticas da tabela, tais como a cardinalidade das chaves, que podem afetar as escolhas que o otimizador faz.

HELP

```
HELP 'search_string'
```

A declaração HELP retorna informações on-line do Manual de Referência do MySQL. O seu funcionamento adequado exige que as tabelas de ajuda do banco de dados mysql sejam inicializadas com informações do tópico de ajuda.

A declaração HELP procura nas tabelas de ajuda para a sequência de pesquisa dada e mostra o resultado da pesquisa. A sequência de pesquisa não diferencia maiúsculas de minúsculas. A cadeia de pesquisa pode conter os caracteres curinga "%" e "_". Estes têm o mesmo significado como para as operações de correspondência de padrões realizada com o operador LIKE. Por exemplo, HELP "rep% ' retorna uma lista de tópicos que começam com rep.

A declaração HELP compreende vários tipos de sequências de pesquisa:

No nível mais geral, use *contents* para recuperar uma lista das categorias de nível superior da Ajuda:

HELP 'contents'

Para obter uma lista de tópicos de uma determinada categoria de ajuda, tais como tipos de dados, use o nome da categoria:

HELP 'data type'

Para obter ajuda sobre um tópico de ajuda específico, como a função ASCII() ou a instrução CREATE TABLE, use a palavra-chave ou palavras-chave associadas:

HELP 'ascii'
HELP 'create table'

Em outras palavras, a cadeia de procura corresponde a uma categoria, muitos temas ou um único tópico. Você não pode dizer antecipadamente se uma determinada sequência de pesquisa irá retornar uma lista de itens ou informações para um único tópico de ajuda. No entanto, você pode verificar o tipo de resposta de ajuda retornado, examinando o número de linhas e colunas no conjunto de resultados 'search_string'.

USE

A instrução USE nome_bd diz ao MySQL para usar o banco de dados nome_bd como o banco de dados padrão (atual) para as instruções subsequentes. O banco de dados continua a ser o padrão até o final da sessão ou se outra instrução USE for emitida. Veja um exemplo abaixo:

```
USE db1;
SELECT COUNT (*) FROM mytable; # Seleciona de db1.mytable
USE db2;
SELECT COUNT (*) FROM mytable; # Seleciona de db2.mytable
```

Fazer um banco de dados específico padrão, por meio da instrução USE, não impede você de acessar tabelas em outras bases de dados. O exemplo a seguir acessa a tabela 'author' do banco de dados db1 e a tabela 'editor' do banco de dados db2:

```
USE db1;
SELECT author_name,editor_name FROM author,db2.editor
WHERE author.editor_id = db2.editor.editor_id;
```

Agora que já sabemos algo sobre os comandos utilitários, vamos fazer a questão abaixo para fixar nosso conhecimento.

4. CESPE - Auditor de Controle Externo (TCE-PA)/Informática/Analista de Suporte/2016

Acerca da configuração e administração dos bancos de dados SQL Server 2008 R2 e MySQL 5.7, julgue o item subsequente.

No MySQL 5.7, ao se digitar o comando mysqld --help, será exibida uma lista completa e detalhada de todos os comandos utilizados pelo SGBD.

Certo

Errado

Comentários: A sintaxe correta para o comando Help é:

HELP 'search_string'

A instrução HELP retorna informações on-line do manual de referência do MySQL. Sua operação apropriada requer que as tabelas de ajuda no banco de dados mysql sejam inicializadas com informações sobre o tópico da ajuda.

A instrução HELP pesquisa as tabelas de ajuda para a sequência de pesquisa fornecida e exibe o resultado da pesquisa. A cadeia de pesquisa não faz distinção entre maiúsculas e minúsculas.

A cadeia de pesquisa pode conter os caracteres curinga% e _. Estes têm o mesmo significado que para operações de correspondência de padrões realizadas com o operador LIKE. Por exemplo, HELP 'rep%' retorna uma lista de tópicos que começam com rep.

Assim, a questão está errada.

Gabarito: E

MARIA DB

Vamos abrir um parêntese antes do final da aula para apresentar o Maria DB. Se você usa Linux e está familiarizado com as ferramentas de Software Livre, deve saber que o Maria DB foi um fork do MySQL. O Maria DB teve início quando o MySQL foi adquirido pela Sun Microsystems em 2008. Como falamos durante a aula, a Sun foi adquirida pala Oracle posteriormente.

Na maioria das distribuições Linux, o Maria DB é o pacote default que provê um servidor de banco de dados relacional compatível com MySQL. Uma coisa interessante é que ele apresenta novas funcionalidades, melhorias nos testes e na performance, e algumas correções de bugs que não estão presentes no MySQL.

Um dos principais avanços foi a incorporação de um novo **mecanismo de armazenamento** conhecido como **Xtradb**, que é um fork do InnoDB. Essa ramificação XtraDB tinha um objetivo: ser uma substituição simples para o mecanismo de armazenamento InnoDB, de modo que os usuários poderiam simplesmente alternar seu mecanismo de armazenamento sem ter que alterar o código do aplicativo subjacente.

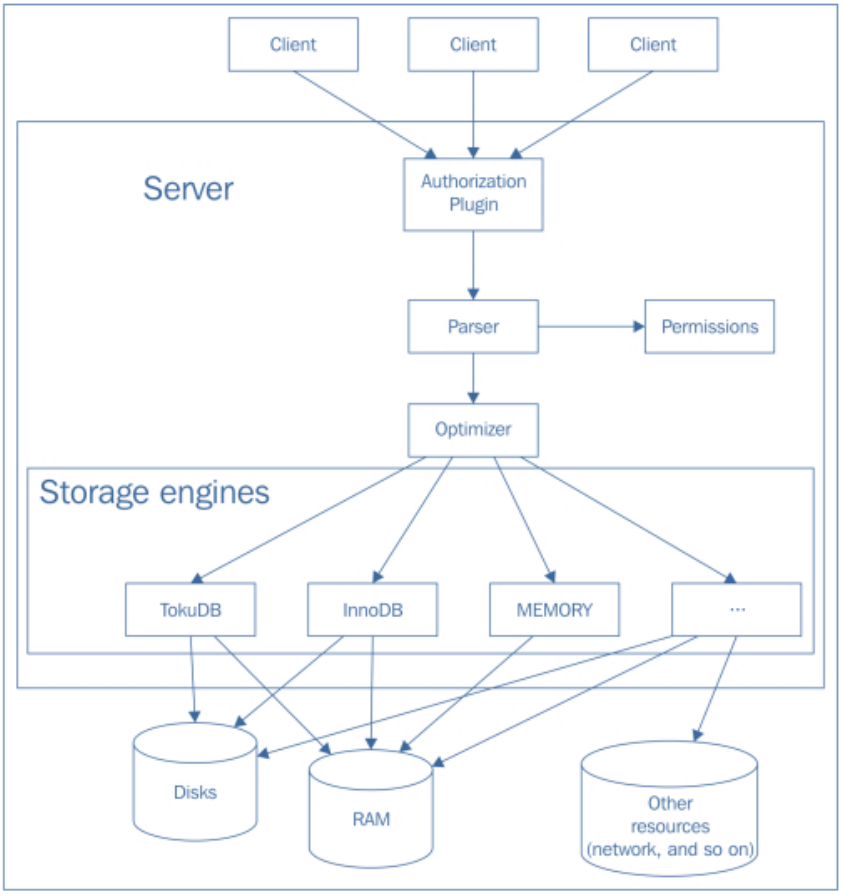
XtraDB, portanto, tinha que ter compatibilidade retroativa com o InnoDB, além de fornecer todos os novos recursos e melhorias que eles queriam incluir. XtraDB foi projetado para lidar com websites de alta disponibilidade, executando em servidores modernos com uma dúzia ou mais de núcleos e muita memória RAM (32 GB ou mais). Para incorporar esse mecanismo ao MariaDB, devemos utilizar o produto Percona Server[1].

O protocolo ou API e a maioria das instruções SQL que rodam no MySQL também funcionam plenamente com o MariaDB. Os *plugins* que são escritos para o MySQL também funcionam com o MariaDB. Graças a estas características, a maioria das aplicações desenvolvidas para rodar no MySQL funcionam no MariaDB, sem quaisquer modificações necessárias.

[1] Percona Server é um produto, lançado pela Percona, empresa líder em consultoria de MySQL. É um produto de banco de dados independente que oferece aos usuários a capacidade de modificar sua instalação do MySQL e colocar o produto Percona Server, e assim obter a vantagem do mecanismo de armazenamento XtraDB.

Ao mesmo tempo, ao mudar para MariaDB você pode usar outros recursos interessantes que não estão disponíveis com o MySQL. Desta forma, se sua organização fizer uso destes recursos, você deve ter o MariaDB instalado. Se o desenvolvedor de software ignorar esses recursos, a aplicação pode usar ambos os bancos MariaDB e MySQL.

A seguir, apresentamos a arquitetura do MariaDB:



Basicamente, do ponto de vista de um usuário, o MariaDB recebe algumas consultas ou instruções SQL, as elabora e retorna um conjunto de resultados. Vamos ver este processo e os componentes envolvidos em mais detalhes.

Quando um cliente se conecta ao MariaDB, uma **autenticação** é executada com base no nome do **host do cliente**, **nome de usuário** e **senha**. A autenticação pode, opcionalmente, ser delegada a um ***plugin***. Se o ***login*** for bem-sucedido, o cliente poderá enviar uma consulta SQL para o servidor.

O ***parser*** compreende a sequência de comandos SQL. Então, o servidor verifica se o cliente tem as permissões necessárias para a ação solicitada. Se a consulta estiver armazenada no **cache de consulta**, os resultados são imediatamente devolvidos ao cliente.

O **otimizador** tentará encontrar a estratégia de execução mais rápida ou o **plano de consulta**. Isso significa que o otimizador decide a ordem na qual as tabelas serão lidas. Ele também decide quais índices serão acessados e se uma tabela interna temporária será usada. Uma boa estratégia pode reduzir muito o acesso aos discos e a complexidade das operações em algumas ordens de grandeza.

Os **mecanismos de armazenamento** leem e gravam os arquivos de dados, os arquivos de índice e as informações de cache. Estas últimas podem ser usadas para acelerar as operações. Alguns recursos importantes, como transações e chaves estrangeiras, são implementados no nível do mecanismo de armazenamento.

O **MariaDB** e os **motores de armazenamento** mantêm um conjunto de logs para guardar uma faixa das instruções recebidas, erros ocorridos, alterações nos dados e assim por diante. A maioria dos logs é opcional, no entanto, alguns logs são necessários para algumas tarefas administrativas. Por exemplo, o **log binário** permite **backups ou replicação**.

O MariaDB tem várias opções ou variáveis que afetam o comportamento do servidor. Muitas delas são dinâmicas, o que significa que podem ser alterados em tempo de execução. Outras são estáticas, o que significa que o valor atribuído durante a inicialização do servidor não pode ser alterado. A maioria deles existe em diferentes níveis: de sessão e global.

Quando uma variável é definida no nível de sessão, significa que qualquer usuário individual pode alterar o valor por meio da sua conexão atual. Já o nível global se aplica a todos os usuários que não definiram um valor de sessão. Uma opção pode ser definida de várias maneiras, como **parâmetros de linha de comando do servidor**, em **arquivos de configuração** ou, se for dinâmica, por meio de uma instrução SQL.

O MariaDB lê um conjunto de arquivos de configuração em uma determinada ordem. A localização exata e a ordem de leitura dependem do sistema operacional. Normalmente, apenas uma instância do MariaDB é executada em uma máquina, portanto, apenas um arquivo de configuração é necessário. Normalmente, o arquivo está disponível em **/etc/my.cnf**, no **Linux**, e em **my.ini**, no diretório de instalação do MariaDB no **Windows**, por exemplo, C:\MariaDB 10.0\My.ini.

No entanto, este sistema de configuração modular é útil se vários servidores MariaDB (e talvez MySQL) estiverem instalados na mesma máquina. É provável que algumas configurações sejam válidas para todos os servidores, mas cada servidor pode especificar mais opções ou substituir os valores genéricos.

Um arquivo também pode ser colocado no diretório home do usuário, de modo que só será lido se MariaDB for executado com essa identidade de usuário (o parâmetro de inicialização --user). Iniciar um servidor com parâmetros de linha de comando substitui as configurações do arquivo. Essas técnicas são úteis ao testar o comportamento de diferentes versões do servidor ou servidores com configurações diferentes.

O serviço do MariaDB é inicializado por meio do arquivo **mysqld**. No Linux, é possível executar o servidor diretamente, mas geralmente ele é invocado por outro script. O script **mysqld_safe** inicia o servidor e também o reinicia, nos casos em que ele é encerrado anormalmente. Isso é muito mais seguro nos ambientes de produção.

O script **mysql.server** também está disponível para os sistemas do tipo **System V**, em que o nível de execução existe. Este script é distribuído com outro nome por muitas distribuições Linux. Quando várias instalações estão presentes na mesma máquina, é possível gerenciá-las usando o **mysql_multi**.

Essa foi uma apresentação rápida do MariaDB, sua arquitetura e seus comandos básicos de inicialização. Lembro que uma das poucas questões de MariaDB apareceu na prova do Banco Central e questionava ao candidato se ele era ou não um SGBD de código aberto. Sabemos que, pelo exposto, o MariaDB é um exemplo de software livre.

Vamos, então, seguir em frente na nossa aula e resolver algumas questões de provas anteriores.

QUESTÕES COMENTADAS

1. FUNDATEC - ANC (PROCERGS)/PROCERGS/Suporte/Bancos de Dados/2023

Analise as assertivas abaixo sobre o sistema de banco de dados MySQL versão 8 e assinale a alternativa correta.

I. O sistema suporta múltiplos mecanismos de armazenamento.

II. O tipo de dado JSON é suportado, permitindo acesso a campos internos de um valor.

III. Tabelas diferentes podem ser transacionais ou não-transacionais no mesmo banco de dados.

a) Todas estão corretas.

b) Todas estão incorretas.

c) Apenas I e II estão corretas.

d) Apenas I e III estão corretas.

e) Apenas II e III estão corretas.

Comentário: A alternativa correta é: a) Todas estão corretas.

I. O MySQL versão 8 suporta múltiplos mecanismos de armazenamento, incluindo InnoDB, MyISAM, MEMORY, NDB (Cluster), entre outros. Cada mecanismo tem suas próprias características e é adequado para diferentes tipos de aplicação.

II. O tipo de dado JSON é suportado no MySQL 8, permitindo armazenar e consultar dados no formato JSON diretamente no banco de dados. Isso oferece flexibilidade para trabalhar com dados semiestruturados e simplifica operações de leitura e escrita de dados JSON.

III. No MySQL 8, tabelas diferentes podem ser transacionais (usando o mecanismo InnoDB, por exemplo) ou não-transacionais (usando o mecanismo MyISAM, por exemplo) no mesmo

banco de dados. Isso permite adaptar o tipo de tabela conforme as necessidades específicas de cada conjunto de dados.

Gabarito: A

2. FUNDATEC - ANC (PROCERGS)/PROCERGS/Suporte/Bancos de Dados/2023

Analise as assertivas abaixo sobre o sistema de banco de dados MySQL versão 8 e assinale a alternativa correta.

I. A replicação do banco de dados melhora a performance de leituras, mas piora a performance de escritas.

II. Replicação pode ser usada para criar uma cópia local dos dados de um nodo remoto do sistema distribuído.

III. Análise dos dados precisa ser realizada na fonte, já que réplicas podem ter dados desatualizados.

a) Todas estão corretas.

b) Todas estão incorretas.

c) Apenas I e II estão corretas.

d) Apenas I e III estão corretas.

e) Apenas II e III estão corretas.

Comentário: A alternativa correta é: e) Apenas II e III estão corretas.

I. A replicação do banco de dados geralmente melhora a performance de leitura, pois distribui a carga de consultas entre os servidores de réplica. No entanto, a performance de escrita pode ser afetada, especialmente em configurações síncronas, onde as transações precisam ser confirmadas em todas as réplicas antes de serem consideradas concluídas.

II. Replicação pode ser usada para criar uma cópia local dos dados de um nodo remoto do sistema distribuído. Essa é uma das vantagens da replicação, permitindo que os dados sejam espalhados geograficamente para melhorar a disponibilidade e a redundância.

III. Análise dos dados precisa ser realizada na fonte, já que réplicas podem ter dados desatualizados. Embora as réplicas sejam uma cópia dos dados originais, pode haver um pequeno atraso entre as atualizações no banco de dados principal e a propagação dessas alterações para as réplicas. Portanto, a análise de dados em réplicas pode não refletir o estado mais atualizado dos dados.

Gabarito: E

3. FUNDATEC - ANC (PROCERGS)/PROCERGS/Suporte/Bancos de Dados/2023

Assinale a alternativa que NÃO corresponde à técnica que pode ser usada para otimização de instruções do tipo SELECT no MySQL 8.

a) Adicionar um índice.

b) Diminuir o número de leituras de tabelas inteiras.

c) Usar o comando ANALYZE_TABLE periodicamente.

d) Criptografar o banco de dados.

e) Ajustar o tamanho dos espaços de memória usados para cache.

Comentário: A opção correta é: d) Criptografar o banco de dados.

As técnicas a, b, c e e são comumente usadas para otimizar instruções SELECT no MySQL 8:

a) Adicionar um índice: Isso pode acelerar a busca de registros em uma tabela, especialmente quando a coluna indexada é usada em uma cláusula WHERE.

b) Diminuir o número de leituras de tabelas inteiras: Isso pode ser feito usando índices ou otimizando a consulta para buscar apenas os dados necessários.

c) Usar o comando ANALYZE TABLE periodicamente: Isso pode ajudar o otimizador de consultas a tomar decisões mais precisas sobre como acessar os dados.

e) Ajustar o tamanho dos espaços de memória usados para cache: Isso pode melhorar o desempenho geral do banco de dados, permitindo que mais dados sejam armazenados em memória para acesso rápido.

Gabarito: D

4. FUNDATEC - ANC (PROCERGS)/PROCERGS/Suporte/Bancos de Dados/2023

É um comando MySQL 8 que, quando executado, remove a permissão do usuário Usuario de inserir na tabela Tabela?

a) REMOVE INSERT ON Tabela FROM 'Usuario';

b) DELETE INSERT ON Tabela FROM 'Usuario';

c) REVOKE INSERT ON Tabela FROM 'Usuario';

d) REVOKE GRANT OPTION ON Tabela FROM 'Usuario';

e) DENY INSERT ON Tabela FROM 'Usuario';

Comentário: A opção correta é: c) REVOKE INSERT ON Tabela FROM 'Usuario';

O comando REVOKE é usado para remover permissões anteriormente concedidas a um usuário ou papel em um banco de dados. Neste caso, REVOKE INSERT ON Tabela FROM 'Usuario' revogaria a permissão de inserção na tabela Tabela do usuário 'Usuario'.

Gabarito: C

5. FCC - AM (MPE PB)/MPE PB/Analista de Sistemas/Administrador de Banco de Dados/2023

O banco de dados de um órgão do Judiciário foi modelado conforme imagem abaixo, utilizando o Modelo Entidade-Relacionamento (MER).

Foi criado um banco de dados chamado MPEPB123 com as tabelas referentes ao modelo e os dados abaixo foram cadastrados. Considere para todas as questões que o banco de dados está aberto e em condições ideais.

Tabela Processo

numeroProc	orgaoProc	tribunalProc	origemProc
00018426 72017	5	01	0246
00456139 12014	8	19	0004
00567124 32022	6	14	0023
00023476 52022	8	02	0341

Tabela Advogado

numeroOABAdv	nomeAdv
28H418	Marcos Vieira Dias
34.443	Fabiana Duque Zanon

Tabela Advogado_Processo

numeroOABAdv	numeroProc	papel
28H418	00018426 72017	Defesa

34.443	00456139 12014	Def esa
28H418	00567124 32022	Acu saçã o
28H418	00456139 12014	Acu saçã o
34.443	00567124 32022	Acu saçã o
34.443	00018426 72017	Acu saçã o

Considere a Stored Procedure abaixo, criada no banco de dados MySQL.

```
delimiter //
___I
BEGIN
SELECT nomeAdv INTO nome FROM Advogado
WHERE numeroOABAdv = oab;
END//
delimiter;
CALL obterNome('28H418', @nome);
SELECT @nome AS NomeAdvogado;
```

Para que, ao executar esta sequência de comandos, seja exibido corretamente o nome do advogado Marcos Vieira Dias, cujo número OAB é 28H418, a lacuna I deve ser preenchida com

a) CREATE STORED PROCEDURE obterNome(CHAR(6) oab, CHAR(45) nome OUT)

b) CREATE PROCEDURE obterNome(IN oab VARCHAR(6), OUT nome VARCHAR(45))

c) CREATE PROCEDURE obterNome(VARCHAR(6) oab, VARCHAR(45) nome)

d) CREATE STORED PROCEDURE obterNome(IN oab CHAR(6), OUT nome CHAR(45))

e) CREATE PROCEDURE obterNome(String oab, String nome)

Gabarito: B

Comentário: Para que a Stored Procedure seja executada corretamente e exiba o nome do advogado Marcos Vieira Dias, cujo número OAB é 28H418, a lacuna I deve ser preenchida com a opção:

b) CREATE PROCEDURE obterNome(IN oab VARCHAR(6), OUT nome VARCHAR(45))

- A declaração da Stored Procedure começa com CREATE PROCEDURE.
- Os parâmetros de entrada (IN) e saída (OUT) devem ser declarados corretamente.
- Neste caso, o número OAB (oab) é um parâmetro de entrada, e o nome do advogado (nome) é um parâmetro de saída.
- A opção b) tem a sintaxe correta para a declaração de uma Stored Procedure no MySQL.

Portanto, a opção correta é a letra b) "CREATE PROCEDURE obterNome(IN oab VARCHAR(6), OUT nome VARCHAR(45))".

6. FCC - AM (MPE PB)/MPE PB/Analista de Sistemas/Administrador de Banco de Dados/2023

Um analista está usando uma ferramenta de gerenciamento de migrações de banco de dados que ajuda a manter a consistência dos esquemas em várias instâncias. Nessa ferramenta, em condições ideais, ele digitou o comando abaixo.

___I___-user=mppb -password=justice -
url=jdbc:mysql://localhost:3306/mppbdb -
locations=filesystem:/mppb/bd/data ___II

Para aplicar todas as migrações disponíveis que ainda não foram aplicadas ao banco de dados MySQL mppbdb usando o nome de usuário mppb e a senha justice, as lacunas I e II devem ser corretamente preenchidas por

a) expdp e commit.

b) flyway e migrate.

c) flyway e commit.

d) swagger e migrate.

e) deploy e and commit.

Gabarito: B

Comentário: Para aplicar todas as migrações disponíveis que ainda não foram aplicadas ao banco de dados MySQL mppbdb usando o nome de usuário mppb e a senha justice com a ferramenta Flyway, as lacunas I e II devem ser corretamente preenchidas por:

I -user=mppb -password=justice -

url=jdbc:mysql://localhost:3306/mppbdb -

locations=filesystem:/mppb/bd/data

II migrate

Portanto, a opção correta é a letra b): flyway e migrate. O comando flyway é utilizado para configurar a ferramenta Flyway, e migrate é o comando para aplicar as migrações ao banco de dados. Note que esses comandos são específicos do Flyway, uma ferramenta popular para migração de banco de dados.

7. (VUNESP - Tec (Jaguariúna)/Pref Jaguariúna/Tecnologia da Informação/2023)

Considere o seguinte comando emitido por meio do sistema gerenciador de banco de dados MySQL (v. 8.0):

```
SELECT '\'teste';
```

A execução desse comando terá como saída:

a) \teste

b) \'teste

c) teste

d) '\'teste

e) ‘teste

Gabarito: E

Comentário: No MySQL, quando você usa uma barra invertida (\) dentro de uma string, ela é considerada um caractere de escape. Neste caso específico:

sql

Copy code

SELECT '\'teste';

A barra invertida antes da aspa simples (') é um caractere de escape, indicando que a aspa simples faz parte da string, e não está sendo usada para delimitar o início ou o fim da string.

A saída do comando será: e) 'teste. Portanto, a resposta correta é a letra e).

8. (VUNESP - MID (Pref Sto André)/Pref Santo André/2023)

Considere o seguinte comando SQL emitido a partir do Sistema Gerenciador de Banco de Dados MySQL:

SELECT 'Teste\nde\nImpressão'

Esse comando terá como resultado a exibição de:

a) Teste

de

Impressão

b) Teste de Impressão

c) TestedeImpressão

d) Impressão

de

Teste

e) Impressão de Teste

Gabarito: A

Comentário:

O comando SQL fornecido utiliza a sequência de escape \n para representar quebras de linha. O resultado da execução será uma saída na forma:

a) Teste

de

Impressão

9. (VUNESP - TTI (TJ RS)/TJ RS/Programador/2023)

Acerca de bancos de dados relacionais, considere a seguinte situação:

Em um SGBD MySQL versão 5.6 há a tabela “Processo” contendo as seguintes colunas: "ID" , do tipo INT e "Data_Julgamento", do tipo DATE.

Assinale a alternativa que apresenta o comando SQL que retorna a quantidade total de julgamentos realizados no ano de 2022.

a) SELECT COUNT(ID) FROM Processo WHERE

Data_Julgamento IN ('2022-01-01', '2022-31-12')

b) SELECT COUNT(ID) FROM Processo WHERE

MONTH(Data_Julgamento) = 2022

c) SELECT COUNT(ID) FROM Processo WHERE

Data_Julgamento LIKE '2022-%'

d) SELECT COUNT(ID) FROM Processo WHERE

Data_Julgamento = '2022'

e) SELECT COUNT(ID) FROM Processo WHERE

YEAR(Data_Julgamento) BETWEEN '2022' AND

'2023'

Gabarito: C

Comentário:

A questão apresenta uma situação em que é necessário criar um comando SQL para obter a quantidade total de julgamentos realizados no ano de 2022 a partir de uma tabela chamada "Processo" em um Sistema de Gerenciamento de Banco de Dados (SGBD) MySQL versão

5.6. A tabela "Processo" possui duas colunas relevantes para a questão: "ID" (do tipo INT) e "Data_Julgamento" (do tipo DATE).

A alternativa correta para essa tarefa seria:

SELECT COUNT(ID) FROM Processo WHERE Data_Julgamento LIKE '2022-%';

A função COUNT(ID) é usada para contar o número de registros que atendem à condição especificada na cláusula WHERE. Nesse caso, a condição é que a "Data_Julgamento" deve começar com "2022-", garantindo que apenas julgamentos do ano de 2022 sejam contados.

Dessa forma, a alternativa correta é a letra c) SELECT COUNT(ID) FROM Processo WHERE Data_Julgamento LIKE '2022-%';.

10. (VUNESP - TTI (TJ RS)/TJ RS/Programador/2023)

Considere um banco de dados relacional MySQL versão 5.7 contendo uma tabela "Processo", que apresenta as colunas "ID" do tipo INT, "ID_Cartorio", também do tipo INT e que faz referência ao cartório no qual tramitou o processo, e "Data_Julgamento" do tipo DATE. Não há valores nulos nessa tabela.

Foi solicitado a um técnico do TJ-RS fazer uma busca dos cartórios que possuem mais do que dois processos.

Assinale a alternativa que apresenta o comando SQL usado para a obtenção da resposta a essa solicitação.

a) SELECT ID_Cartorio FROM Processo WHERE

COUNT(*) > 2;

b) SELECT ID_Cartorio FROM Processo HAVING

Processo.ID_Cartorio > 2;

c) SELECT ID_Cartorio FROM Processo GROUP BY

Processo.ID_Cartorio HAVING Processo.ID_

Cartorio > 2;

d) SELECT ID_Cartorio FROM Processo GROUP BY

Processo.ID_Cartorio HAVING COUNT(*) > 2;

e) SELECT ID_Cartorio FROM Processo WHERE

COUNT(Processo.ID_Cartorio) > 2;

Gabarito: D

Comentário: A opção correta para obter a resposta à solicitação de buscar os cartórios que possuem mais do que dois processos em um banco de dados MySQL versão 5.7 é a alternativa:

d): SELECT ID_Cartorio FROM Processo GROUP BY Processo.ID_Cartorio HAVING COUNT(*) > 2;

Nesta consulta SQL:

GROUP BY Processo.ID_Cartorio agrupa os resultados com base no ID do cartório.

HAVING COUNT(*) > 2 filtra os grupos (cartórios) para incluir apenas aqueles que têm mais de dois processos.

Portanto, a resposta correta é a opção d) SELECT ID_Cartorio FROM Processo GROUP BY Processo.ID_Cartorio HAVING COUNT(*) > 2.

11. (VUNESP - ACE (TCM SP)/TCM SP/Tecnologia da Informação/2023)

Considerando o sistema gerenciador de bancos de dados MySQL (versão 8.0), uma informação importante é saber quais são as bases de dados existentes no servidor, bem como obter o nome da base de dados selecionada no momento da execução do comando. Os dois comandos que respondem a essas duas questões são, respectivamente,

a) RETURN DATABASES; e GET DATABASE();

b) SHOW DATABASES; e SELECT DATABASE();

c) RELEASE DATABASES; e SET DATABASE();

d) REFERENCE DATABASES; e LOAD DATABASE();

e) PROPT DATABASES; e RELOAD DATABASE();

Gabarito: B

Comentário: No MySQL (versão 8.0), os comandos que podem ser usados para listar as bases de dados existentes no servidor e para obter o nome da base de dados selecionada são: a) SHOW DATABASES; e SELECT DATABASE();

Vamos comentar um pouco sobre os dois comandos:

SHOW DATABASES;: Este comando é utilizado para listar todas as bases de dados disponíveis no servidor MySQL. Quando executado, ele retorna uma lista de nomes de bancos de dados presentes no sistema. Aqui está um exemplo de uso:

SHOW DATABASES;

Esse comando exibirá algo semelhante a:

+--------------------+

| Database |

+--------------------+

| information_schema |

| mysql |

| performance_schema |

| sys |

| your_database |

+--------------------+

Isso mostra os bancos de dados disponíveis no servidor.

SELECT DATABASE();: Este comando é usado para obter o nome da base de dados atualmente selecionada durante a sessão do usuário. Se você não tiver selecionado explicitamente um banco de dados, ele retornará NULL. Aqui está um exemplo:

SELECT DATABASE();

A resposta seria algo como:

+------------------+

| DATABASE() |

```
+------------------+
| your_database    |
+------------------+
```

Isso mostra o nome do banco de dados atualmente selecionado. Se nenhum banco de dados foi selecionado, a resposta será NULL.

12. (VUNESP - Tec (Jaguariúna)/Pref Jaguariúna/Tecnologia da Informação/2023)

O sistema gerenciador de banco de dados MySQL (v. 8.0) dispõe de alguns dispositivos de armazenamento (*storage engines*), dentre os quais é correto citar:

a) *Commit* e *Compile*.

b) *InnoDB* e *MyISAM*.

c) *Transaction* e Federated.

d) *Memory* e *Hash*.

e) *Locked* e *Flushing*.

Gabarito: B

Comentário: A resposta correta é: b) InnoDB e MyISAM. Os dispositivos de armazenamento no MySQL referem-se às engines de armazenamento ou mecanismos de armazenamento que o sistema oferece. Cada engine tem suas próprias características, comportamentos e otimizações específicas. Duas das engines mais comuns no MySQL são o InnoDB e o MyISAM.

InnoDB:

- Tipo: Transacional.
- Características Principais:
  - Suporte a transações (ACID compliance), garantindo consistência e confiabilidade.
  - Suporte a chave estrangeira (foreign key), o que é fundamental para integridade referencial.
  - Implementação de bloqueios a nível de linha, proporcionando maior concorrência em ambientes com várias operações simultâneas.
  - Oferece rollback e commit para transações.
  - Eficiente para leitura e escrita intensiva.

MyISAM:

- Tipo: Não transacional.
- Características Principais:
    - Não oferece suporte a transações ou chave estrangeira, o que pode ser uma limitação em certos contextos.
    - Implementação de bloqueios a nível de tabela, o que pode resultar em menor concorrência em ambientes com muitas operações simultâneas.
    - Mais eficiente para operações de leitura, sendo adequado para situações em que as leituras são predominantes.
    - Estrutura simples e fácil de usar.

Ao escolher entre InnoDB e MyISAM, é importante considerar os requisitos específicos do projeto. Se a integridade referencial, suporte a transações e concorrência são cruciais, o InnoDB é geralmente a escolha preferida. Se a simplicidade e a eficiência em leituras são mais importantes, o MyISAM pode ser adequado. Além dessas, existem outras engines no MySQL, cada uma com suas características específicas, como MEMORY (HEAP), NDB (Cluster), etc. A escolha dependerá das necessidades e requisitos do projeto.

13. (VUNESP - Ana (Pref Marília)/Pref Marília/Programador de Sistemas/2023)

Considerando o sistema gerenciador de banco de dados MySQL (8.0), quando da escrita de algum comando, deve-se atentar para as regras de precedência de operadores, sendo correto que o operador

a) & tem menor precedência do que o operador +.

b) MOD tem menor precedência do que o operador &&.

c) NOT tem maior precedência do que o operador *.

d) IS tem maior precedência do que o operador DIV.

e) XOR tem maior precedência do que o operador ~.

Gabarito: A

Comentário: No MySQL, as regras de precedência de operadores determinam a ordem em que diferentes operadores são avaliados em uma expressão. Vamos analisar cada opção fornecida:

a) & tem menor precedência do que o operador +.

Correto. O operador "&" tem menor precedência em relação ao operador "+".

b) MOD tem menor precedência do que o operador &&.

Incorreto. "MOD" e "&&" não são operadores diretos no MySQL. O operador de módulo é representado por "%" e "&&" é geralmente usado como AND lógico.

c) *NOT tem maior precedência do que o operador .

Incorreto. O operador "*" (multiplicação) tem maior precedência do que o operador "NOT".

d) IS tem maior precedência do que o operador DIV.

Incorreto. "IS" não é um operador aritmético e, portanto, não possui precedência em relação ao operador "DIV" (divisão inteira).

e) XOR tem maior precedência do que o operador ~.

Incorreto. O operador "~" (bitwise NOT) tem maior precedência do que o operador XOR.

Portanto, a opção correta é a letra (a).

14. (VUNESP - Ana (Pref Marília)/Pref Marília/Programador de Sistemas/2023)

Considere os dois comandos abaixo colocados, emitidos a partir do sistema gerenciador de banco de dados MySQL (8.0).

a: SELECT STRCMP (‘fase’, ‘fase2’);
b: SELECT STRCMP (‘tecla’, ‘tecla’);

O resultado da execução desses comandos no MySQL (8.0) será, respectivamente

a) a: 0, b: –1.

b) a: –1, b: 1.

c) a: –1, b: 0.

d) a: 1, b: 0.

e) a: 1, b: –1.

Gabarito: C

Comentário:

O comando STRCMP no MySQL retorna 0 se as duas strings são iguais, -1 se a primeira string é menor que a segunda e 1 se a primeira string é maior que a segunda.

Vamos analisar cada comando:

a: SELECT STRCMP('fase', 'fase2');

As duas strings são diferentes. A primeira string ('fase') é menor que a segunda ('fase2'). Portanto, o resultado será -1.

b: SELECT STRCMP('tecla', 'tecla');

As duas strings são idênticas. Portanto, o resultado será 0.

Portanto, o resultado da execução desses comandos será: a) a: -1, b: 0.

15. (VUNESP - Tec (Pref Sorocaba)/Pref Sorocaba/Informática/2023)

Ao utilizar o sistema gerenciador de banco de dados MySQL (v. 8), é possível, no arquivo denominado my.cnf fazer restrições quanto ao uso de *passwords*. Nesse caso, a linha ou comando a ser inserido nesse arquivo de forma a proibir a utilização dos últimos 3 *passwords* utilizados por um usuário é:

a) password_history=3

b) last_passwords=3

c) 3 passwords history

d) last 3 passwords_history

e) history of last 3 passwords

Gabarito: A

Comentário: No MySQL (v. 8), para fazer restrições quanto ao uso de passwords e proibir a utilização dos últimos 3 passwords, você pode usar a opção password-reuse-interval. Esta opção define o intervalo (em dias) durante o qual uma senha anterior não pode ser reutilizada. Portanto, o comando a ser inserido no arquivo my.cnf é:

password-reuse-interval = 3

Portanto, a opção correspondente à resposta correta seria uma versão modificada da opção password-reuse-interval. Nenhuma das opções fornecidas corresponde exatamente à sintaxe correta. No entanto, a opção mais próxima seria: a) password_history=3

16. (VUNESP - PTIC (UNICAMP)/UNICAMP/Desenhista de Páginas da Internet (Web Designer)/2023)

O sistema gerenciador de banco de dados MYSQL v.8.0 possui diversos programas clientes que se conectam ao servidor MySQL. Dentre tais programas clientes encontram- se

a) mysqldump e mysqlinit.

b) mysqlimport e mysqlstart.

c) mysqlpump e mysqldouble.

d) mysqladmin e mysqlcheck.

e) mysqlsh e mysqlpart.

Gabarito: D

Comentário: No MySQL v.8.0, alguns dos programas clientes que se conectam ao servidor MySQL são: d) mysqladmin e mysqlcheck

1. mysqladmin: O `mysqladmin` é uma ferramenta de linha de comando que permite aos administradores executar várias tarefas administrativas no servidor MySQL. Algumas das operações comuns incluem reinicialização do servidor, criação e remoção de bancos de dados, exibição de status do servidor, entre outras. Pode ser útil para automação de tarefas administrativas e monitoramento do servidor.

2. mysqlcheck: O `mysqlcheck` é uma ferramenta de linha de comando usada para verificar, reparar e otimizar tabelas MySQL. Ele pode ser utilizado para analisar e corrigir problemas de tabelas que podem ocorrer devido a falhas, erros ou operações não concluídas adequadamente. É uma ferramenta útil para a manutenção e o cuidado geral com a integridade das tabelas no banco de dados MySQL.

Essas ferramentas são parte do conjunto de utilitários do MySQL e são valiosas para administradores de banco de dados ao lidar com tarefas de manutenção, monitoramento e diagnóstico no ambiente MySQL.

17. (VUNESP - PTIC (UNICAMP)/UNICAMP/Desenhista de Páginas da Internet (Web Designer)/2023)

O comando do sistema gerenciador de banco de dados MYSQL v.8.0 que exibe os privilégios e papéis do usuário corrente de um banco de dados é:

a) SHOW GRANTS FOR CURRENT USER;

b) DISPLAYS GRANTS FOR CURRENT USER;

c) SET REVOKES FOR CURRENT USER;

d) MAKE GRANTS FOR CURRENT USER;

e) SHOW REVOKES FOR CURRENT USER;

Gabarito: A

Comentário: O comando correto para exibir os privilégios e papéis do usuário corrente em um banco de dados MySQL é:

SHOW GRANTS FOR CURRENT_USER;

Portanto, a opção correta é: a) SHOW GRANTS FOR CURRENT USER;

18. (VUNESP - PTIC (UNICAMP)/UNICAMP/Desenhista de Páginas da Internet (Web Designer)/2023)

Considerando o sistema gerenciador de banco de dados MYSQL v.8.0 é possível determinar uma política global para a expiração do prazo de passwords. Para tanto, no arquivo my.cnf deve-se incluir a seguinte linha (considerando que tal prazo deva ser estipulado em 180 dias):

a) time_for_password=180

b) default_password_lifetime=180

c) lifetime_password_databases=180

d) default_time_password=180

e) make_password_time=180

Gabarito: B

Comentário:

A opção correta para definir uma política global para a expiração do prazo de senhas no MySQL 8.0, no arquivo my.cnf, é: default_password_lifetime=180. Portanto, a alternativa correta é a opção: b) default_password_lifetime=180.

O parâmetro default_password_lifetime no MySQL 8.0 é utilizado para definir a quantidade de dias que uma senha é válida. Ele especifica a quantidade de dias que um usuário pode manter sua senha antes que ela expire. No exemplo fornecido, default_password_lifetime=180, a senha será válida por 180 dias a partir do momento em que é definida ou alterada.

Isso faz parte das políticas de segurança, incentivando os usuários a atualizarem suas senhas regularmente, reduzindo assim o risco associado a senhas que podem ser comprometidas se forem mantidas inalteradas por longos períodos. O administrador do banco de dados pode ajustar esse valor de acordo com os requisitos de segurança da organização.

Ao definir essa política global, ela se aplica a todos os usuários do sistema, a menos que usuários individuais tenham políticas específicas configuradas, o que pode substituir ou complementar a política global.

19. (VUNESP - Ana Sis (CM SBO)/CM SBO/2023)

O Sistema Gerenciador da Banco de Dados MySQL 8.0 suporta cinco tipos de valores inteiros, sendo dois desses tipos

a) MININT e MAXINT.

b) TINYNT e BIGINT.

c) SINGLEINT e PLUSINT.

d) UNIINT e DOUBLEINT.

e) MICROINT e MACROINT.

Gabarito: B

Comentário: O Sistema Gerenciador de Banco de Dados MySQL 8.0 suporta cinco tipos principais de valores inteiros, que são:

- TINYINT
- SMALLINT
- MEDIUMINT
- INT ou INTEGER
- BIGINT

Portanto, a opção correta é a alternativa (b) "TINYINT e BIGINT".

20. (CESGRANRIO - Tec Cien (BASA)/BASA/Tecnologia da Informação/2022)

Em um servidor MySQL, qual log é usado para registrar mensagens de diagnóstico, como erros, avisos e notificações, que ocorrem durante a inicialização e o desligamento do servidor?

a) server log
b) error log
c) general query log
d) binary log
e) relay log

Gabarito: B

Comentário: No MySQL, o log usado para registrar mensagens de diagnóstico, incluindo erros, avisos e notificações, que ocorrem durante a inicialização e o desligamento do servidor, é conhecido como **error log**. Portanto, a opção correta é: b) **error log**.

Vamos comentar as demais opções:

a) **server log**: Não é uma designação específica no MySQL para logs. Pode se referir genericamente a logs relacionados ao servidor, mas não é uma denominação específica no contexto do MySQL.

c) **general query log**: Registra todas as consultas SQL executadas no servidor MySQL. Pode ser útil para fins de depuração, mas geralmente não é a escolha ideal em ambientes de produção devido ao volume significativo de registros que pode gerar.

d) **binary log**: Registra todas as alterações nos dados, como inserções, atualizações e exclusões, em um formato binário. É frequentemente usado para replicação e recuperação de desastres.

e) **relay log**: É específico para servidores MySQL configurados como servidores de réplica (replication). Ele registra as alterações recebidas de um servidor mestre e as aplica no servidor de réplica.

Esses logs desempenham papéis específicos em cenários diferentes, mas para mensagens de diagnóstico durante a inicialização e desligamento do servidor, o **error log** é o mais relevante.

21. Cebraspe – Técnico Judiciário – Tecnologia da Informação (TRT-AP/PA)/2022

Para se realizar um backup lógico, por meio de um conjunto de instruções SQL que podem ser executadas para reproduzir as definições de objetos do banco de dados originais e os dados da tabela no MySql 8 de uma database chamada dbtrt, o comando correto é

a) mysqldump --databases dbtrt > backup.sql.

b) cp database dbtrt > backup.bkp.

c) mysqlbackup -db dbtrt > backup.sql.

d) mysqlbackup dbtrt –logical > backup.bkp.

e) backup -database dbtrt > backup.sql.

Comentário:

O comando correto é:

a) mysqldump --databases dbtrt > backup.sql.

Explicação: O utilitário "mysqldump" é utilizado para criar backups lógicos do MySQL. Na opção "--databases", você especifica o nome do banco de dados "dbtrt" que deseja fazer o backup. O sinal ">" é usado para redirecionar a saída do comando para um arquivo chamado "backup.sql".

As demais alternativas estão incorretas:

b) O comando "cp" é usado para copiar arquivos e diretórios, não é uma opção para realizar backup lógico com a estrutura do MySQL.

c) O comando "mysqlbackup" não tem uma opção "-db", e a opção "-logical" não é usada para especificar um backup lógico.

d) A opção "-logical" existe no utilitário "mysqlbackup", mas o formato da linha de comando está incorreto. O correto seria "mysqlbackup --backup-dir=/caminho/para/diretorio/dbtrt --backup-image=logical".

e) O comando "backup" não é reconhecido como um comando válido para fazer backups lógicos no MySQL.

Gabarito: Letra A

---

22. CESPE - Técnico Judiciário (STM)/Apoio Especializado/Programação de Sistemas/2018

Julgue o item, que dizem respeito aos SGBDs Oracle, MySQL e PostgreSQL.

Nas tabelas do tipo InnoDB do MySQL, o armazenamento dos dados pode ser realizado por um ou mais arquivos separados.

Certo Errado

Comentários: A questão está correta. Nas tabelas do tipo InnoDB, os dados são organizados em tabelas em muitos arquivos pequenos, alguns arquivos maiores ou uma combinação de ambos.

Os arquivos de dados que você define no arquivo de configuração usando a opção de configuração innodb_data_file_path formam o espaço de tabelas do sistema InnoDB. Os arquivos são logicamente concatenados para formar o espaço de tabela do sistema. Não há striping em uso. Você não pode definir onde, dentro do espaço de tabela do sistema, suas tabelas são alocadas. Em um espaço de tabela do sistema recém-criado, o InnoDB aloca espaço a partir do primeiro arquivo de dados.

Para evitar os problemas que acompanham o armazenamento de todas as tabelas e

índices dentro do espaço de tabela do sistema, é possível ativar a opção de configuração innodb_file_per_table (o padrão), que armazena cada tabela recém-criada em um arquivo de espaço de tabela separado (com extensão .ibd). Para tabelas armazenadas dessa maneira, há menos fragmentação no arquivo do disco e, quando a tabela é truncada, o espaço é retornado ao sistema operacional, em vez de ainda ser reservado pelo InnoDB dentro do espaço de tabela do sistema.

Você também pode armazenar tabelas em espaços de tabela gerais. Espaços de tabela gerais são espaços de tabela compartilhados criados usando a sintaxe CREATE TABLESPACE. Eles podem ser criados fora do diretório de dados do MySQL, são capazes de manter várias tabelas e suportar tabelas de todos os formatos de linha.

Gabarito: C

---

23. CESPE - Oficial Técnico de Inteligência/Área 9/2018

Julgue o próximo item, a respeito de conceitos e comandos PostgreSQL e MySQL.

Por se tratar de um sistema gerenciador de banco de dados de código aberto, o MySQL não oferece suporte a conexões criptografadas entre clientes e o servidor.

Certo

Errado

Comentários: A questão está errada, pois existe suporte a conexões criptografadas. O recurso de criptografia de espaço de tabela InnoDB fornece criptografia de dados em repouso para espaços de tabela de arquivos por tabela, espaços de tabela gerais e o espaço de tabela do sistema mysql. O suporte a criptografia para tablespaces gerais foi introduzido no MySQL

8.0.13. O suporte de criptografia para o espaço de tabelas do sistema mysql está disponível a partir do MySQL 8.0.15.

A criptografia de espaço de tabela usa uma arquitetura de chave de criptografia de dois níveis, consistindo de uma chave mestra de criptografia e de chaves de espaço de tabela. Quando um espaço de tabela é criptografado, uma chave de espaço de tabela é criptografada e armazenada no cabeçalho do espaço de tabela. Quando um aplicativo ou usuário autenticado deseja acessar dados de espaço de tabela criptografados, o InnoDB usa uma chave mestra de criptografia para descriptografar a chave do espaço de tabela. A versão descriptografada de uma chave de espaço de tabela nunca muda, mas a chave de criptografia principal pode ser alterada conforme necessário. Esta ação é referida como rotação da chave mestra.

Gabarito: E

---

24. CESPE - Técnico Judiciário (STJ)/Apoio Especializado/Suporte Técnico/2018

A respeito de sistemas gerenciadores de banco de dados (SGBD.), julgue o item.

Em um SGBD MySQL, o número de subqueries é limitado em até dois níveis em cada conjunto de instruções em um SELECT.

Certo

Errado

Comentários: A questão está errada, pois o guia de referência do MySQL não menciona limite de níveis para o número de subqueries.

Gabarito: E

25. FCC - Analista Tecnológico (PRODATER)/Analista de Suporte Técnico/2016 Considere o comando do sistema gerenciador de bancos de dados MySQL (v. 5.6): SELECT FORMAT (3587.9, 2)

O resultado da execução desse comando é:

a) '3587.90'

b) '3587,90'

c) '3.587,90'

d) '3,587.90'

e) '3,587,90'

Comentários: A resposta está na letra d). A função FORMAT (X, D) retorna um número formatado para o número especificado de casas decimais.
A função formata o número X para um formato como '#, ###, ###. ##', arredondado para D casas decimais e retorna o resultado como uma string. Se D for 0, o resultado não terá ponto decimal ou parte fracionária.

Seguem alguns exemplos:

```
mysql> SELECT FORMAT(12332.123456, 4);
    -> '12,332.1235'
mysql> SELECT FORMAT(12332.1,4);
    -> '12,332.1000'
mysql> SELECT FORMAT(12332.2,0);
    -> '12,332'
mysql> SELECT FORMAT(12332.2,2,'de_DE');
    -> '12.332,20'
```

Gabarito: D

26. FCC - Analista de Tecnologia da Informação (CREMESP)/Administração de Banco de

Dados/2016

O sistema de privilégios do MySQL garante que cada usuário possa fazer exatamente as operações as quais possui permissão.

O controle de acesso do MySQL é composto de dois estágios. No estágio

a) 1 o servidor pede uma senha de acesso e verifica se a senha confere usando a tabela passw_users.

b) 2 o servidor verifica se o usuário realizou um drop e, em caso positivo, cancela a ação do usuário pois é um comando proibido.

c) 1 o servidor utiliza apenas as tabelas user e db e no estágio 2 utiliza a tabela host_priv no banco de dados MySQL.

d) 2, se a solicitação envolver tabelas, o servidor pode consultar adicionalmente as tabelas tables_priv e columns_priv.

e) 1 o servidor verifica quais privilégios que o usuário possui antes de permitir que ele realize a primeira operação.

Comentários: Quando você tenta se conectar a um servidor MySQL, o servidor aceita ou rejeita a conexão com base nessas condições (estágio 1):

Sua identidade e se você pode verificar sua identidade fornecendo a senha correta Se sua conta está bloqueada ou desbloqueada

O servidor verifica as credenciais primeiro e, em seguida, o estado de bloqueio da conta. Uma falha em qualquer etapa faz com que o servidor negue acesso total a você. Caso contrário, o servidor aceita a conexão e, em seguida, entra no Estágio 2 e aguarda solicitações.

No estágio 2 do controle de acesso, para cada solicitação emitida por meio dessa conexão, o servidor determina qual operação você deseja executar e, em seguida, verifica se você tem privilégios suficientes para fazer isso. É nesse ponto que as colunas de privilégio nas tabelas de permissões entram em ação. Esses privilégios podem vir de qualquer uma das tabelas user, db, tables_priv, columns_priv ou procs_priv.

As tabelas tables_priv, columns_priv e procs_priv concedem privilégios específicos da tabela, da coluna específica e da rotina.

Temos, então, nossa resposta na letra d).

Gabarito: D

27. FCC - Analista de Tecnologia da Informação (CREMESP)/Administração de Banco de Dados/2016

O MySQL 5.7 possui diversas tabelas do sistema que contêm informações sobre contas de usuário e os privilégios por eles detidos. A tabela que contém os privilégios no nível do Bando de Dados é

a) database_priv.

b) columns_priv.

c) procs_priv.

d) db.

e) proxies_priv.

Comentários: O banco de dados do sistema mysql inclui várias tabelas de permissões que contêm informações sobre contas de usuários e os privilégios mantidos por elas.

Para manipular o conteúdo das tabelas de permissões, modifique-as indiretamente usando instruções de gerenciamento de conta, como CREATE USER, GRANT e REVOKE, para configurar contas e controlar os privilégios disponíveis para cada uma.
Essas tabelas de banco de dados mysql contêm informações de concessão:

user: contas de usuário, privilégios globais e outras colunas sem privilégios
global_grants: Atribuições de privilégios globais dinâmicos aos usuários
db: privilégios no nível de banco de dados
tables_priv: Privilégios no nível da tabela
columns_priv: privilégios no nível da coluna
procs_priv: Procedimento armazenado e privilégios de função
proxies_priv: Privilégios do usuário proxy
default_roles: funções de usuário padrão
role_edges: bordas para subgráficos de funções
password_history: alterações de senha

Temos, então, o gabarito na letra d).

Gabarito: D

---

28. CESPE - Analista de Tecnologia da Informação (FUB)/2015

Com relação ao banco de dados MySQL, julgue o item subsequente.

No MySQL o uso do comando grant permite adicionar permissões a objetos do banco de dados. Certo

Errado

Comentários: A questão está correta. A instrução GRANT permite que os administradores do sistema concedam privilégios e funções, que podem ser concedidos a contas e funções de usuários. Essas restrições de sintaxe se aplicam:

GRANT não pode combinar a concessão de privilégios e funções na mesma instrução. Uma determinada declaração GRANT deve conceder privilégios ou funções.

A cláusula ON distingue se a instrução concede privilégios ou

funções: Com ON, a declaração concede privilégios.

Sem ON, a declaração concede papéis.

É permitido atribuir privilégios e funções a uma conta, mas você deve usar instruções GRANT separadas, cada uma com uma sintaxe apropriada ao que deve ser concedido.

A sintaxe do comando

GRANT é: GRANT

priv_type [(column_list)]

[, priv_type

[(column_list)]] ... ON

[object_type] priv_level

TO user_or_role [,

user_or_role] ... [WITH GRANT

OPTION]

object_type: {

TABLE

| FUNCTION

| PROCEDURE

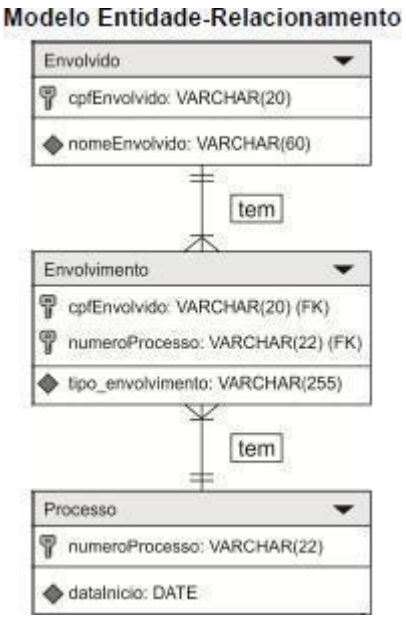
}

Gabarito: C

---

29. FCC - Analista Ministerial (MPE PB)/Analista de Sistemas/Administrador de Banco de Dados/2015

Atenção: Considere as informações a seguir para responder à questão.



Dados cadastrados nas tabelas:

cpfEnvolvido	nomeEnvolvido
121.134.045-01	Marcos Paulo
128.249.039-14	Maria de Fátima
131.091.431-09	André Luiz
158.245.067-12	Pedro nda Silva
160.234.074-11	João da Silva

Observação: O erro no nome Pedro nda Silva é proposital.

cpfEnvolvido	numeroProcesso	tipo_envolvimento
121.134.045-01	12345670020108261234	acusado
128.249.039-14	12345670020108261234	acusador
128.249.039-14	78654310020118150675	acusado
131.091.431-09	16789000020105131234	acusado
158.245.067-12	73982110120111131001	acusado
158.245.067-12	78654310020118150675	acusador
158.245.067-12	98006770120111144571	acusado

numeroProcesso	dataInicio
12345670020108261234	20/01/2015
16789000020105131234	08/08/2010
73982110120111131001	10/12/2014
78654310020118150675	30/01/2015
82745660320095202300	22/06/2014
98006770120111144571	21/02/2015

Considere que todos os números de processo tenham 20 dígitos na tabela Envolvimento. Dos 20 dígitos, 2 caracteres do número dos processos representam o número do Tribunal. Por exemplo, no processo de número 78654310020118150675 o número do Tribunal é 15.

Ao ser executada uma instrução SQL, foram exibidos os seguintes dados:

cpfEnvolvido	Tribunal	tipo_envolvimento
128.249.039-14	26	Acusador
158.245.067-12	15	Acusador

Considere a instrução SQL abaixo.

```
SELECT cpfEnvolvido, numeroProcesso
FROM Envolvimento
WHERE tipo_envolvimento = (
SELECT tipo_envolvimento FROM Envolvimento WHERE tipo_envolvimento ='acusador'
);
```

Ao executá-la no Oracle ou MySQL,

a) serão exibidos o CPF e o número do processo apenas dos registros cujo conteúdo do campo tipo_envolvimento possui o valor acusador.

b) ocorrerá um erro, pois a subconsulta retornará mais de uma linha.

c) para que não ocorra erro, o sinal de igual (=) deverá ser trocado pela palavra ANY.

d) serão exibidos todos os dados dos registros cujo conteúdo do campo tipo_envolvimento possui o valor acusador.

e) ocorrerá um erro, pois não é possível utilizar o resultado de uma instrução SELECT como parâmetro de comparação da cláusula WHERE de outra instrução SELECT.

Comentários: A execução da instrução trará um erro, gabarito letra b).

Uma subconsulta pode retornar um escalar (um valor único), uma única linha, uma única coluna ou uma tabela (uma ou mais linhas de uma ou mais colunas). Eles são chamados de subconsultas escalares, de colunas, de linhas e de tabelas.

Neste caso, temos mais de uma linha. O MySQL trará o seguinte erro:

Error number: 1242; Symbol: ER_SUBQUERY_NO_1_ROW; SQLSTATE: 21000

Message: Subquery returns more than 1 row

Gabarito: B

30. CESPE - Analista Judiciário (TJDFT)/Apoio Especializado/Análise de Sistemas/2015

Um banco de dados permite à aplicação o armazenamento e a recuperação de dados com eficiência, o que garante segurança e integridade das informações. No caso de banco de dados relacional, os dados são armazenados em tabelas e os relacionamentos entre elas as tornam relacionais. A esse respeito, julgue o item que se segue.

Uma das principais desvantagens do banco de dados MySQL é que ele não possui suporte para triggers — gatilhos predefinidos e associados a tabelas — disparados por algum evento específico.

Certo

Errado

Comentários: A questão está errada. O MySQL oferece suporte para os triggers. Um trigger é um objeto de banco de dados nomeado associado a uma tabela e que é ativado quando ocorre um determinado evento para a tabela. Alguns usos para gatilhos são:

Executar verificações de valores a serem inseridos em uma tabela

Executar cálculos nos valores envolvidos em uma atualização.

Um trigger é definido para ativar quando uma instrução insere, atualiza ou exclui linhas na tabela associada. Essas operações de linha são eventos de gatilho. Por exemplo, as linhas podem ser inseridas pelas instruções INSERT ou LOAD DATA e um trigger de inserção é ativado para cada linha inserida. Um gatilho pode ser definido para ativar antes ou depois do evento de gatilho. Por exemplo, você pode ativar um trigger antes de cada linha inserida em uma tabela ou após cada linha atualizada.

Gabarito: E

---

31. CESPE - Técnico Judiciário (TJDFT)/Apoio Especializado/Programação de Sistemas/2015

Acerca do PostgreSQL e do MySQL, julgue o seguinte item.

O comando SELECT FROM CAT é utilizado para que sejam apresentados todos os bancos de dados disponíveis no servidor MySQL.

Certo

Errado

Comentários: O gabarito da questão é errado.

SELECT é usado para recuperar linhas selecionadas de uma ou mais tabelas e pode incluir instruções e subconsultas UNION. Uma instrução SELECT pode começar com uma cláusula WITH para definir expressões de tabelas comuns acessíveis no SELECT. Segue sua sintaxe:

SELECT

[ALL | DISTINCT |

DISTINCTROW]

[HIGH_PRIORITY]

[STRAIGHT_JOIN]

[SQL_SMALL_RESULT] [SQL_BIG_RESULT]

[SQL_BUFFER_RESULT] SQL_NO_CACHE

[SQL_CALC_FOUND_ROWS]

select_expr [, select_expr ...] (não foi definida nenhuma

coluna) [FROM table_references (CAT)

```
[WHERE where_condition]
  [GROUP BY {col_name | expr | position}, ... [WITH
  ROLLUP]] [HAVING where_condition]
  [WINDOW window_name AS
    (window_spec) [, window_name AS
    (window_spec)] ...]
  [ORDER BY {col_name | expr |
   position} [ASC | DESC], ... [WITH
   ROLLUP]]
  [LIMIT {[offset,] row_count | row_count OFFSET
  offset}] [INTO OUTFILE 'file_name'
    [CHARACTER SET
    charset_name]
    export_options
   | INTO DUMPFILE 'file_name
   | INTO var_name [, var_name]]
  [FOR {UPDATE | SHARE} [OF tbl_name [, tbl_name] ...] [NOWAIT | SKIP LOCKED]
   | LOCK IN SHARE MODE]]
```

Cada select_expr indica uma coluna que você deseja recuperar. Deve haver pelo menos um select_expr. *Table_references* indica a tabela ou tabelas das quais recuperar linhas.

A cláusula WHERE, se fornecida, indica a condição ou condições que as linhas devem satisfazer para serem selecionadas. *where_condition* é uma expressão avaliada como verdadeira para cada linha a ser selecionada. A instrução seleciona todas as linhas se não houver uma cláusula WHERE.

Gabarito: E

---

32. Ano: 2018 Banca: CESPE Órgão: STM Cargo: Analista de Sistemas Questão: 51 a 54

```
1. CREATE TABLE IF NOT EXISTS 'software' (
2. 'id' int NOT NULL,
3. 'nome' varchar(70) NOT NULL,
4. PRIMARY KEY ('id')
5. ) DEFAULT CHARSET=utf8;
6. INSERT INTO 'software' ('id', 'nome') VALUES
7. ('1', 'Programa ABC'),
```

```
8. ('2', 'Programa WYZ'),
9. ('3', 'Programa DFG');
10. CREATE TABLE IF NOT EXISTS 'vsoftware' (
11. 'idsoft' int UNSIGNED NOT NULL REFERENCES software(id),
12. 'versao' int(3) NOT NULL,
13. 'descricao' varchar(70) NOT NULL,
14. PRIMARY KEY ('idsoft','versao'));
15. INSERT INTO 'vsoftware'('idsoft', 'versao', 'descricao') VALUES
16.     ('1', '1', 'criacao do programa.'),
17. ('2', '1', '1a versao.'),
18. ('1', '2', 'atualizacao na tela A.'),
19. ('1', '3', 'adicao da tela C.'),
20. ('2', '2', 'adicao da tela D.');
```

Com base nos comandos MySQL 5.6 precedentes, julgue os itens a seguir.

75 Especialmente devido à expressão na linha 11, o comando a seguir, após executado, retornará três registros.

```
SELECT a.idsoft, a.versao, a.descricao
FROM 'vsoftware' a
INNER JOIN (
 SELECT idsoft, MAX(versao) versao
 FROM 'vsoftware'
 GROUP BY idsoft) b ON a.idsoft = b.idsoft
AND a.versao = b.versao;
```

76 A execução do comando

```
SELECT a.idsoft, a.versao, a.descricao
FROM 'vsoftware' a
LEFT OUTER JOIN 'vsoftware' b
 ON a.idsoft = b.idsoft AND a.versao < b.versao
WHERE b.idsoft IS NULL;
```

retornará os seguintes dados.

idsoft	versao	descricao
1	3	adicao da tela C.
2	2	adicao da tela D.

77 A execução do comando

```
SELECT a.id
FROM 'software' a
where a.id not in (
SELECT b.idsoft
FROM 'vsoftware' b);
```

terá resultado idêntico à execução do comando a seguir.

```
SELECT distinct a.id
FROM 'software' a left outer join 'vsoftware' b
on a.id=b.idsoft;
```

78 A tabela vsoftware está na segunda forma normal (2FN), porque contém uma chave estrangeira referenciada à tabela software.

79 O comando ALTER TABLE software add data datetime; no MySQL 5.6 adiciona um novo campo data à tabela software.

80 O comando a seguir no MySQL 5.6 modifica o tipo do campo nome para CHAR na tabela software.

```
ALTER TABLE software MODIFY nome char(100);
```

Comentário: Vamos comentar cada uma das questões acima:

75. Essa alternativa tem uma consulta interna que agrupa por software id. Veja que cada software vai retornar sua última versão na outra coluna. Como temos dois softwares, o resultado intermediário são apenas duas tuplas. Logo em seguida, temos um inner join que vai compara o valor da versão e do id na clausula ON. Tal fato mantém apenas duas tuplas no resultado. Logo, a alternativa está errada.

76. A lógica para resolver essa questão é seguinte: não existirá elementos no lado direito, quando a retrição "a.versao < b.versao" não for satisfeita. Logo, temos as seguintes tuplas abaixo, em que b.idsoft IS NULL. Isso é, de fato, o que consta na sugestão de resposta. Logo, a afirmação está correta.

('1', '3', 'adicao da tela C.'),

('2', '2', 'adicao da tela D.');

77. A primeira retornará os programas que não têm versões ainda publicadas. Neste caso concreto, apenas o id 3. Já a segunda retornará todos os valores de id que tenham ou não versão publicada. Como elas não apresentam o mesmo resultado, temos uma alternativa incorreta.

78. A tabela está de fato na segunda forma normal. Mas isso acontece, pois não existe dependência parcial da chave primária. A justificativa presente na alternativa afirma algo diferente disto. Logo, temos uma alternativa errada.

79. A afirmação usa a sintaxe correta do MySQL para alteração de tabela. Lembrando que o column é opcional. Logo, alternativa correta.

80. Para mudar o tipo de dados ou a definição da coluna, você pode usar o comando

MODIFY, que é uma extensão do MySQL para ser compatível com o SGBD Oracle. É uma forma de mudar a definição sem mexer no nome do atributo. Desta forma, a alternativa está correta.

Gabarito: E C E E C C.

33. Ano: 2018 Banca: CESPE Órgão: STM Cargo: Analista de Sistemas Questão: 71 a 74

Um sistema gerenciador de banco de dados (SGBD) instalado no Linux deve ser configurado de modo a permitir os seguintes requisitos:

I no máximo, 1000 conexões simultâneas;

II somente conexões originadas a partir do servidor de aplicação com IP 10.10.10.2.

Tendo como referência essas informações, julgue os seguintes itens.

73 Caso o SGBD instalado seja o MySQL 5.7, para atendimento dos requisitos I e II, deve-se modificar o arquivo my.cnf, alterando-se os parâmetros max_user_connections para 1000 e connection_source para o IP fornecido; e reiniciar o serviço do SGBD.

Comentário: Neste caso, não existe o connection_source no arquivo my.cnf. É possível restringir os IPs que acessam o banco usando a variável bind-address. Já a variável que limita a quantidade de requisições por usuários é *max_user_connections*. Se você quiser definir a quantidade de requisições globais, use o termo *max_connections*. Logo, temos mais uma alternativa errada.

Gabarito: E.

34. Ano: 2017 Banca: FCC Órgão: ARTESP Cargo: Especialista em Regulação de Transporte - Tecnologia da Informação Questão: 72.

72. O sistema gerenciador de bancos de dados MySQL (versão 5.6 e posteriores) admite o bloqueio e o desbloqueio para acesso às tabelas. Os dois comandos utilizados para essas funções de bloqueio e desbloqueio de tabelas são, respectivamente,

(A) GRANT e REVOKE.

(B) COMMIT e ROLLBACK.

(C) OPEN e CLOSE.

(D) LOCK e UNLOCK.

(E) START e END.

Comentário: Essa questão de bloqueio em banco de dados nos remete ao conceito de LOCK e UNLOCK. O MySQL permite que sessões de cliente adquiram os bloqueios de tabela explicitamente com a finalidade de cooperar com outras sessões no acesso a tabelas ou para evitar que outras sessões modifiquem tabelas durante o período em que a sessão requer acesso exclusivo a elas.

Uma sessão pode adquirir ou liberar bloqueios apenas para si mesmo. Uma sessão não pode

adquirir bloqueios para outra sessão ou liberar bloqueios mantidos por outra sessão. Os bloqueios podem ser usados para emular transações ou para obter uma velocidade na atualização das tabelas.

O LOCK TABLES adquire explicitamente os bloqueios de tabela para a sessão de cliente atual. Os bloqueios de tabela podem ser adquiridos para as tabelas base ou visões. Você deve ter o privilégio LOCK TABLES e o privilégio SELECT para cada objeto a ser bloqueado.

Para bloqueio sobre visões, o LOCK TABLES adiciona todas as tabelas usadas na view ao conjunto de tabelas a ser bloqueado. Se você bloquear uma tabela explicitamente com LOCK TABLES, todas as tabelas usadas nos triggers associados a essa tabela também serão bloqueadas implicitamente.

O UNLOCK TABLES libera explicitamente todos os bloqueios de tabela mantidos pela sessão atual. O LOCK TABLES libera implicitamente quaisquer bloqueios de tabela mantidos pela sessão atual antes de adquirir novos bloqueios.

Outro uso para UNLOCK TABLES é liberar o bloqueio de leitura global adquirido por meio da instrução FLUSH TABLES WITH READ LOCK, que permite bloquear todas as tabelas em todos os bancos de dados.

Um bloqueio de tabela protege somente contra leituras inadequadas ou gravações por outras sessões. Uma sessão segurando um bloqueio de WRITE pode executar operações em nível de tabela, como DROP TABLE ou TRUNCATE TABLE. Para sessões com um READ LOCK, operações DROP TABLE e TRUNCATE TABLE não são permitidas.

Assim, após essa rápida explanação teórica sobre o uso de LOCK e UNLOCK no MySQL, podemos marcar nosso gabarito na alternativa D.

Gabarito: D

35. Ano: 2017 Banca: FCC Órgão: TRE-SP Cargo: Técnico Judiciário de TI – Operação de computadores

58. Resumidamente e, ainda, considerando todas as condições e pré-condições de ambiente já existentes para garantir o funcionamento adequado de um banco de dados, um Técnico, usando MySQL 5.6, escreveu as seguintes expressões e comandos SQL:

```
CREATE TABLE NAMES (Id integer PRIMARY KEY, Name
text); INSERT INTO NAMES VALUES(1,'Tom');
INSERT INTO NAMES
VALUES(2,'Lucy'); INSERT INTO
NAMES VALUES(3,'Frank'); INSERT
INTO NAMES VALUES(4,'Jane');
INSERT INTO NAMES
VALUES(5,'Robert');
SELECT Name FROM NAMES WHERE Id = 3 or Id = 5;
```

Quanto à construção do Schema e quanto à execução do Select, este, presumidamente,

especificado para recuperar Frank e Robert, o resultado foi

(A) erro, porque todos os Insert deveriam ter sido escritos seguindo a sintaxe exemplo INSERT INTO NAMES VALUES OF (1,'Tom').

(B) erro, porque o SQL deveria ter sido executado como SELECT Name FROM NAMES WHERE Id = 3 or = 5.

(C) a recuperação de Frank e Robert.

(D) erro, porque o CREATE deveria ter sido escrito CREATE TABLE NAMES (Id integer PRIMARY KEY, For Name text).

(E) a recuperação de 3 Frank e 5 Robert.

Comentário: Vejam que essa questão, embora tenha como tema MySQL, trata de uma questão de SQL. Você conseguiria responder sem problemas apenas entendendo que as inserções foram feitas com sucesso na tabela e que a consulta vai retornar apenas as informações da coluna Name. Sendo assim, a recuperação de Frank e Robert foi executada com sucesso e nossa resposta está presente na alternativa C.

Gabarito: C.

36. Ano: 2016 Banca: CESPE Órgão: TCE-SC Cargo: Auditor de TI

Acerca dos sistemas gerenciadores de banco de dados MySQL e PostgreSQL, julgue os itens subsequentes.

61 Na administração do MySQL 5.6, ao se executar o becape full lógico por meio do aplicativo mysqldump, recomenda-se verificar a integridade dos dados antes de realizar o becape. Nesse caso, a sintaxe correta é a mostrada a seguir. mysqldump -u usuario -p senha --check-all-db -- all-databases > arquivo.sql

Comentário: Vamos lá comentar a alternativa acima.

[61] O mysqldump de fato executa um becape lógico. Ele produz um conjunto de comandos SQL que possam reconstruir as tabelas e os dados. Contudo, não existe nenhuma recomendação para verificação da integridade dos dados antes do bacape. O comando está com todos os parâmetros corretos, exceto o –check-all-db, que não existe na documentação oficial.

Desabafo: essa questão foge de qualquer contexto de prova de concurso. Acho que até um DBA teria dificuldade de responder essa questão com 100% de certeza.

Gabarito: E.

37. BANCA: FGV ANO: 2013 ÓRGÃO: AL-MT PROVA: ANALISTA DE SISTEMAS - BANCO DE DADOS

No MySQL 5.1 ou mais recentes, os tipos padrão de índices para o storage engine do tipo MEMORY/HEAP são:

A B+TREE e B*TREE.

B BTREE e B+TREE.

C B*TREE e HASH.

D B+TREE e HASH.

E BTREE e HASH.

Comentário: Antes de responder a essa questão, queria apresentar uma tabela disponível na documentação do MySQL, que compara as funcionalidades das diferentes storage engines. A figura apresenta, entre outros aspectos, limites de armazenamento, suporte a transação, granularidade de bloqueios e suporte aos diferentes tipos de índices.

Feature	MyISAM	Memory	InnoDB	Archive	NDB
Storage limits	256TB	RAM	64TB	None	384EB
Transactions	No	No	Yes	No	Yes
Locking granularity	Table	Table	Row	Table	Row
MVCC	No	No	Yes	No	No
Geospatial data type support	Yes	No	Yes	Yes	Yes
Geospatial indexing support	Yes	No	Yes [a]	No	No
B-tree indexes	Yes	Yes	Yes	No	No
T-tree indexes	No	No	No	No	Yes
Hash indexes	No	Yes	No [b]	No	Yes
Full-text search indexes	Yes	No	Yes [c]	No	No
Clustered indexes	No	No	Yes	No	No
Data caches	No	N/A	Yes	No	Yes
Index caches	Yes	N/A	Yes	No	Yes
Compressed data	Yes [d]	No	Yes [e]	Yes	No
Encrypted data [f]	Yes	Yes	Yes	Yes	Yes
Cluster database support	No	No	No	No	Yes
Replication support [g]	Yes	Yes	Yes	Yes	Yes
Foreign key support	No	No	Yes	No	No
Backup / point-in-time recovery [h]	Yes	Yes	Yes	Yes	Yes
Query cache support	Yes	Yes	Yes	Yes	Yes
Update statistics for data dictionary	Yes	Yes	Yes	Yes	Yes

[a] InnoDB support for geospatial indexing is available in MySQL 5.7.5 and higher.

[b] InnoDB utilizes hash indexes internally for its Adaptive Hash Index feature.

[c] InnoDB support for FULLTEXT indexes is available in MySQL 5.6.4 and higher.

[d] Compressed MyISAM tables are supported only when using the compressed row format. Tables using the compressed row format with MyISAM are read only.

Pela figura é possível verificar que a storage engine do tipo MEMORY, que anteriormente era chamada de HEAP, como já vimos, possui duas possibilidades para a estrutura de dados dos arquivos de índices: B-TREE e HASH. Essas opções combinam com a nossa resposta na alternativa E.

Gabarito: E.

38. BANCA: FGV ANO: 2013 ÓRGÃO: AL-MT PROVA: ANALISTA DE SISTEMAS - BANCO DE DADOS

O modo recomendado para iniciar o servidor MySQL no UNIX é através do script

A mysql_up.

B mysql_startup.

C mysqld_safe.

D mysqld.

E mysqld_on.

Comentário: Já vimos que o mysqld, também conhecido como MySQL Server, é o principal programa que faz a maioria do trabalho em uma instalação do MySQL. Ele gerencia o acesso ao diretório de dados MySQL, que contém os bancos de dados e as tabelas. O diretório de dados também é o padrão para outras informações, como arquivos de log e de status.

Quando o servidor MySQL é iniciado, ele fica monitorando as conexões de rede de programas de cliente e gerencia o acesso delas a bases de dados em nome desses clientes.

O programa mysqld tem muitas opções que podem ser especificadas na inicialização. Para obter uma lista completa de opções, você pode usar este comando:

shell> mysqld –verbose --help

Outro programa que existe é o mysqld_safe. Ele fornece uma maneira recomendada para iniciar um servidor mysqld no Unix. O mysqld_safe adiciona alguns recursos de segurança, tais como reiniciar o servidor quando ocorre um erro e escrever informações sobre os erros de *runtime* no arquivo de log de erro.

Distribuições MySQL em Unix incluem um script chamado mysql.server, que inicia o servidor usando mysqld_safe. Ele pode ser usado em sistemas como o Linux e o Solaris, que utilizam System V-Style para executar diretórios visando iniciar e parar serviços do sistema. Ele também é usado pelo item de inicialização OS X para o MySQL.

Para iniciar ou parar o servidor manualmente usando o script mysql.server, basta invocá-lo com argumentos start ou stop:

shell> mysql.server start

shell> mysql.server stop

O último programa de inicialização é o mysqld_multi, projetado para gerenciar vários processos mysqld que escutem ligações em diferentes arquivos socket UNIX e portas TCP/IP. Ele pode iniciar ou parar servidores, ou relatar seu status atual.

O mysqld_multi procura por grupos nomeados [mysqldN] no arquivo my.cnf (ou no arquivo definido na opção --defaults-file). N pode ser qualquer número inteiro positivo. Os números de grupo diferenciam um grupo de opções de outro. Eles são também usados como argumentos para mysqld_multi para especificar quais servidores você deseja iniciar, parar ou obter um relatório de status.

Gabarito: C

39. BANCA: FGV ANO: 2013 ÓRGÃO: AL-MT PROVA: ANALISTA DE SISTEMAS - BANCO DE DADOS

O formato físico utilizado pelas versões 5.1 e para as mais recentes do MySQL, para representar suas tabelas, é o

A .tab.

B .mys.

C .frm.

D .sql.

E .ino.

Comentário: A questão está testando seu conhecimento a respeito da representação física no MySQL que representa cada tabela por um arquivo de formato .frm (definição) no diretório do banco de dados. O mecanismo de armazenamento (engine) para a tabela pode criar outros arquivos.

Para tabelas do InnoDB, o armazenamento de arquivos é controlado pela opção de configuração innodb_file_per_table. Quando esta opção for desativada, todas as tabelas e índices InnoDB são armazenados no espaço de tabela do sistema, representada por um ou mais arquivos .ibd. Para cada tabela InnoDB criada quando essa opção for ativada, os dados da tabela e de todos os índices associados são armazenados em um arquivo .ibd localizado dentro do diretório do banco de dados.

Para tabelas MyISAM, o mecanismo de armazenamento cria arquivos de dados e arquivos de índices. Assim, para cada tabela nome_tabela MyISAM, existem três arquivos de disco.

Arquivo / Finalidade arquivo

nome_tabela.frm / Tabela formato de arquivo (definição)

tbl_name.MYD / Arquivo de dados

tbl_name.MYI /Arquivo de índice

Gabarito C.

As questões abaixo têm o foco maior no desenvolvimento de banco de dados usando MySQL.

40. BANCA: FCC ANO: 2013 ÓRGÃO: MPE-CE PROVA: ANALISTA MINISTERIAL - CIÊNCIAS DA COMPUTAÇÃO

A associação entre o comando do banco de dados MySQL e sua descrição é expressa corretamente em:

A

Mudar de base de dados	Ver os formatos dos campos da tabela	Apagar uma base de dados	Atualizar permissões e privilégios de uma base de dados
change [nome bd];	check [nome tabela];	drop database [nome bd];	UPDATE PRIVILEGES;

B

Mudar de base de dados	Ver os formatos dos campos da tabela	Apagar uma base de dados	Atualizar permissões e privilégios de uma base de dados
use [nome bd];	describe [nome tabela];	drop database [nome bd];	FLUSH PRIVILEGES;

C

Mudar de base de dados	Ver os formatos dos campos da tabela	Apagar uma base de dados	Atualizar permissões e privilégios de uma base de dados
change [nome bd];	show [nome tabela];	delete database [nome bd];	set privileges;

D

Mudar de base de dados	Ver os formatos dos campos da tabela	Apagar uma base de dados	Atualizar permissões e privilégios de uma base de dados
show [nome bd];	list [nome tabela];	drop tables [nome bd];	list privileges;

E

Mudar de base de dados	Ver os formatos dos campos da tabela	Apagar uma base de dados	Atualizar permissões e privilégios de uma base de dados
use [nome bd];	show [nome tabela];	erase database [nome bd];	flush privileges;

Comentário: Vamos analisar qual comando está associado a cada uma das descrições independente da alternativa.

Mudar de base de dados: Criar um banco de dados não seleciona ele automaticamente para o uso. Você deve fazer isso explicitamente. Para fazer seu banco de dado 'concurso' ser o banco corrente, você deve usar o comando: USE concurso; e deve receber como resposta Database Change.

Verificar os formatos dos campos da tabela: Os comandos DESCRIBE e EXPLAIN são sinônimos. Na prática, a palavra-chave DESCRIBE é mais frequentemente usada para obter informações sobre a estrutura da tabela, enquanto que EXPLAIN é usada para obter um plano de execução da consulta (ou seja, uma explicação de como o MySQL vai executar uma consulta).

Apagar uma base de dados: Vimos que todos os objetos criados podem ser apagados, utilizando o comando DROP. Neste caso, portando, devemos usar o DROP DATABASE.

Atualizar permissões e privilégios de uma base de dados: Parte do controle de segurança FLUSH PRIVILEGES recarrega os privilégios das tabelas de permissões no banco de dados mysql. As informações de caches do servidor na memória são resultado de declarações GRANT, CREATE USER, CREATE SERVER, e INSTALL PLUGIN. Esta memória não é liberada pelos comandos REVOKE, DROP USER, DROP SERVER e UNINSTALL PLUGIN correspondentes. Assim, para um servidor que executa muitas instâncias, essas informações acabam ficando em cache. Esta informação da memória cache pode ser liberada com FLUSH PRIVILEGES.

Vejam, portanto, que a nossa resposta está na alternativa B

Gabarito: B

41. BANCA: CESPE ANO: 2014 ÓRGÃO: ANATEL PROVA: ANALISTA ADMINISTRATIVO - SUPORTE E INFRAESTRUTURA DE TI

A respeito de banco de dados, julgue os itens que se seguem.

[1] Considere que haja necessidade de se criar uma tabela T1 com uma coluna do tipo ENUM que armazene uma lista permitida de dados em formato string. Suponha ainda que essa operação deva ser feita no Mysql 5.7. Nesse caso, o comando a ser dado seria o seguinte:

CREATE TABLE T1 (COL1 ENUM('1', '2', '3'));

Comentário: A questão basicamente vai criar uma tabela com uma coluna e segue a sintaxe correta do comando. Abaixo coloquei apenas a descrição da criação de coluna presente na documentação do MySQL. Observem as palavras chaves usadas para implementar as restrições de integridade.

```
column_definition:
    data_type [NOT NULL | NULL] [DEFAULT default_value]
      [AUTO_INCREMENT] [UNIQUE [KEY] | [PRIMARY] KEY]
      [COMMENT 'string']
      [COLUMN_FORMAT {FIXED|DYNAMIC|DEFAULT}]
      [reference_definition]
  | data_type [GENERATED ALWAYS] AS (expression)
      [VIRTUAL | STORED] [UNIQUE [KEY]] [COMMENT comment]
      [NOT NULL | NULL] [[PRIMARY] KEY]
```

Gabarito: C.

42. BANCA: FGV ANO: 2013 ÓRGÃO: AL-MT PROVA: ANALISTA DE SISTEMAS - BANCO DE DADOS

No trecho de código SQL a seguir está representado um conjunto de declarações que foram executadas por um programador no MySQL.

```
CREATE TABLE TAB1 (MAT INT(10) NOT NULL, NOME
VARCHAR(4) NOT NULL, FONE VARCHAR(9), PRIMARY KEY
(MAT));
INSERT INTO TAB1 VALUES (0123456789, 'ABC',
'XYZ');
CREATE VIEW V1 AS SELECT NOME, FONE, FONE+NOME AS
VALUE FROM TAB1;
SELECT * FROM V1;
```

O resultado da declaração SELECT corresponde, respectivamente, a

A

NOME	FONE	VALUE
ABC	XYZ	XYZABC

B

NOME	FONE	VALUE
ABC	XYZ	ABCXYZ

C

NOME	FONE	VALUE
ABC	XYZ	-------

D

NOME	FONE	VALUE
ABC	XYZ	.....

E

NOME	FONE	VALUE
ABC	XYZ	0

Comentário: Essa questão é interessante, pois o MySQL não implementa a operação de concatenação usando o operador '+'. Ele executa essa operação por meio do comando *concat*(), como já falamos anteriormente quanto tratamos de operações com *strings*. Contudo, o MySQL não lança um erro ou interrompe a execução. Quando solicitado para executar FONE+NOME, o SGBD simplesmente atribui o valor zero por não conseguir encontrar parâmetros numéricos para operar consistentemente.

Gabarito: E.

Lista de Questões

1. FUNDATEC - ANC (PROCERGS)/PROCERGS/Suporte/Bancos de Dados/2023

Analise as assertivas abaixo sobre o sistema de banco de dados MySQL versão 8 e assinale a alternativa correta.

I. O sistema suporta múltiplos mecanismos de armazenamento.

II. O tipo de dado JSON é suportado, permitindo acesso a campos internos de um valor.

III. Tabelas diferentes podem ser transacionais ou não-transacionais no mesmo banco de dados.

a) Todas estão corretas.

b) Todas estão incorretas.

c) Apenas I e II estão corretas.

d) Apenas I e III estão corretas.

e) Apenas II e III estão corretas.

2. FUNDATEC - ANC (PROCERGS)/PROCERGS/Suporte/Bancos de Dados/2023

Analise as assertivas abaixo sobre o sistema de banco de dados MySQL versão 8 e assinale a alternativa correta.

I. A replicação do banco de dados melhora a performance de leituras, mas piora a performance de escritas.

II. Replicação pode ser usada para criar uma cópia local dos dados de um nodo remoto do sistema distribuído.

III. Análise dos dados precisa ser realizada na fonte, já que réplicas podem ter dados desatualizados.

a) Todas estão corretas.

b) Todas estão incorretas.

c) Apenas I e II estão corretas.

d) Apenas I e III estão corretas.

e) Apenas II e III estão corretas.

3. FUNDATEC - ANC (PROCERGS)/PROCERGS/Suporte/Bancos de Dados/2023

Assinale a alternativa que NÃO corresponde à técnica que pode ser usada para otimização de instruções do tipo SELECT no MySQL 8.

a) Adicionar um índice.

b) Diminuir o número de leituras de tabelas inteiras.

c) Usar o comando ANALYZE_TABLE periodicamente.

d) Criptografar o banco de dados.

e) Ajustar o tamanho dos espaços de memória usados para cache.

4. FUNDATEC - ANC (PROCERGS)/PROCERGS/Suporte/Bancos de Dados/2023

É um comando MySQL 8 que, quando executado, remove a permissão do usuário Usuario de inserir na tabela Tabela?

a) REMOVE INSERT ON Tabela FROM 'Usuario';

b) DELETE INSERT ON Tabela FROM 'Usuario';

c) REVOKE INSERT ON Tabela FROM 'Usuario';

d) REVOKE GRANT OPTION ON Tabela FROM 'Usuario';

e) DENY INSERT ON Tabela FROM 'Usuario';

5. FCC - AM (MPE PB)/MPE PB/Analista de Sistemas/Administrador de Banco de Dados/2023

O banco de dados de um órgão do Judiciário foi modelado conforme imagem abaixo, utilizando o Modelo Entidade-Relacionamento (MER).

Foi criado um banco de dados chamado MPEPB123 com as tabelas referentes ao modelo e os dados abaixo foram cadastrados. Considere para todas as questões que o banco de dados está aberto e em condições ideais.

Tabela Processo

numeroProc	orgaoProc	tribunalProc	origemProc
00018426 72017	5	01	0246
00456139 12014	8	19	0004
00567124 32022	6	14	0023
00023476 52022	8	02	0341

Tabela Advogado

numeroOABAdv	nomeAdv
28H418	Marcos Vieira Dias

34.443	Fabiana Duque Zanon

Tabela Advogado_Processo

numeroOABAdv	numeroProc	papel
28H418	00018426 72017	Defesa
34.443	00456139 12014	Defesa
28H418	00567124 32022	Acusação
28H418	00456139 12014	Acusação
34.443	00567124 32022	Acusação
34.443	00018426 72017	Acusação

Considere a Stored Procedure abaixo, criada no banco de dados MySQL.

```
delimiter //
___I
BEGIN
SELECT nomeAdv INTO nome FROM Advogado
WHERE numeroOABAdv = oab;
```

```
END//
delimiter;
CALL obterNome('28H418', @nome);
SELECT @nome AS NomeAdvogado;
```

Para que, ao executar esta sequência de comandos, seja exibido corretamente o nome do advogado Marcos Vieira Dias, cujo número OAB é 28H418, a lacuna I deve ser preenchida com

a) CREATE STORED PROCEDURE obterNome(CHAR(6) oab, CHAR(45) nome OUT)

b) CREATE PROCEDURE obterNome(IN oab VARCHAR(6), OUT nome VARCHAR(45))

c) CREATE PROCEDURE obterNome(VARCHAR(6) oab, VARCHAR(45) nome)

d) CREATE STORED PROCEDURE obterNome(IN oab CHAR(6), OUT nome CHAR(45))

e) CREATE PROCEDURE obterNome(String oab, String nome)

6. FCC - AM (MPE PB)/MPE PB/Analista de Sistemas/Administrador de Banco de Dados/2023

Um analista está usando uma ferramenta de gerenciamento de migrações de banco de dados que ajuda a manter a consistência dos esquemas em várias instâncias. Nessa ferramenta, em condições ideais, ele digitou o comando abaixo.

```
___I___-user=mppb -password=justice -
url=jdbc:mysql://localhost:3306/mppbdb -
locations=filesystem:/mppb/bd/data ___II
```

Para aplicar todas as migrações disponíveis que ainda não foram aplicadas ao banco de dados MySQL mppbdb usando o nome de usuário mppb e a senha justice, as lacunas I e II devem ser corretamente preenchidas por

a) expdp e commit.

b) flyway e migrate.

c) flyway e commit.

d) swagger e migrate.

e) deploy e and commit.

7. (VUNESP - Tec (Jaguariúna)/Pref Jaguariúna/Tecnologia da Informação/2023)

Considere o seguinte comando emitido por meio do sistema gerenciador de banco de dados MySQL (v. 8.0):

```
SELECT '\'teste';
```

A execução desse comando terá como saída:

a) \teste

b) \'teste

c) teste

d) '\'teste

e) ‘teste

8. (VUNESP - MID (Pref Sto André)/Pref Santo André/2023)

Considere o seguinte comando SQL emitido a partir do Sistema Gerenciador de Banco de Dados MySQL:

```
SELECT ‘Teste\nde\nImpressão’
```

Esse comando terá como resultado a exibição de:

a) Teste

de

Impressão

b) Teste de Impressão

c) TestedeImpressão

d) Impressão

de

Teste

e) Impressão de Teste

9. (VUNESP - TTI (TJ RS)/TJ RS/Programador/2023)

Acerca de bancos de dados relacionais, considere a seguinte situação:

Em um SGBD MySQL versão 5.6 há a tabela “Processo” contendo as seguintes colunas: "ID" , do tipo INT e "Data_Julgamento", do tipo DATE.

Assinale a alternativa que apresenta o comando SQL que retorna a quantidade total de julgamentos realizados no ano de 2022.

a) SELECT COUNT(ID) FROM Processo WHERE

Data_Julgamento IN ('2022-01-01', '2022-31-12')

b) SELECT COUNT(ID) FROM Processo WHERE

MONTH(Data_Julgamento) = 2022

c) SELECT COUNT(ID) FROM Processo WHERE

Data_Julgamento LIKE '2022-%'

d) SELECT COUNT(ID) FROM Processo WHERE

Data_Julgamento = '2022'

e) SELECT COUNT(ID) FROM Processo WHERE

YEAR(Data_Julgamento) BETWEEN '2022' AND

'2023'

10. (VUNESP - TTI (TJ RS)/TJ RS/Programador/2023)

Considere um banco de dados relacional MySQL versão 5.7 contendo uma tabela "Processo", que apresenta as colunas "ID" do tipo INT, "ID_Cartorio", também do tipo INT e que faz referência ao cartório no qual tramitou o processo, e "Data_Julgamento" do tipo DATE. Não há valores nulos nessa tabela.

Foi solicitado a um técnico do TJ-RS fazer uma busca dos cartórios que possuem mais do que dois processos.

Assinale a alternativa que apresenta o comando SQL usado para a obtenção da resposta a essa solicitação.

a) SELECT ID_Cartorio FROM Processo WHERE

COUNT(*) > 2;

b) SELECT ID_Cartorio FROM Processo HAVING

Processo.ID_Cartorio > 2;

c) SELECT ID_Cartorio FROM Processo GROUP BY

Processo.ID_Cartorio HAVING Processo.ID_

Cartorio > 2;

d) SELECT ID_Cartorio FROM Processo GROUP BY

Processo.ID_Cartorio HAVING COUNT(*) > 2;

e) SELECT ID_Cartorio FROM Processo WHERE

COUNT(Processo.ID_Cartorio) > 2;

11. (VUNESP - ACE (TCM SP)/TCM SP/Tecnologia da Informação/2023)

Considerando o sistema gerenciador de bancos de dados MySQL (versão 8.0), uma informação importante é saber quais são as bases de dados existentes no servidor, bem como obter o nome da base de dados selecionada no momento da execução do comando. Os dois comandos que respondem a essas duas questões são, respectivamente,

a) RETURN DATABASES; e GET DATABASE();

b) SHOW DATABASES; e SELECT DATABASE();

c) RELEASE DATABASES; e SET DATABASE();

d) REFERENCE DATABASES; e LOAD DATABASE();

e) PROPT DATABASES; e RELOAD DATABASE();

12. (VUNESP - Tec (Jaguariúna)/Pref Jaguariúna/Tecnologia da Informação/2023)

O sistema gerenciador de banco de dados MySQL (v. 8.0) dispõe de alguns dispositivos de armazenamento (*storage engines*), dentre os quais é correto citar:

a) *Commit* e *Compile*.

b) *InnoDB* e *MyISAM*.

c) *Transaction* e Federated.

d) *Memory* e *Hash*.

e) *Locked* e *Flushing*.

13. (VUNESP - Ana (Pref Marília)/Pref Marília/Programador de Sistemas/2023)

Considerando o sistema gerenciador de banco de dados MySQL (8.0), quando da escrita de algum comando, deve-se atentar para as regras de precedência de operadores, sendo correto que o operador

a) & tem menor precedência do que o operador +.

b) MOD tem menor precedência do que o operador &&.

c) NOT tem maior precedência do que o operador *.

d) IS tem maior precedência do que o operador DIV.

e) XOR tem maior precedência do que o operador ~.

14. (VUNESP - Ana (Pref Marília)/Pref Marília/Programador de Sistemas/2023)

Considere os dois comandos abaixo colocados, emitidos a partir do sistema gerenciador de banco de dados MySQL (8.0).

a: SELECT STRCMP ('fase', 'fase2');
b: SELECT STRCMP ('tecla', 'tecla');

O resultado da execução desses comandos no MySQL (8.0) será, respectivamente

a) a: 0, b: –1.

b) a: –1, b: 1.

c) a: –1, b: 0.

d) a: 1, b: 0.

e) a: 1, b: –1.

15. (VUNESP - Tec (Pref Sorocaba)/Pref Sorocaba/Informática/2023)

Ao utilizar o sistema gerenciador de banco de dados MySQL (v. 8), é possível, no arquivo denominado my.cnf fazer restrições quanto ao uso de *passwords*. Nesse caso, a linha ou comando a ser inserido nesse arquivo de forma a proibir a utilização dos últimos 3 *passwords* utilizados por um usuário é:

a) password_history=3

b) last_passwords=3

c) 3 passwords history

d) last 3 passwords_history

e) history of last 3 passwords

16. (VUNESP - PTIC (UNICAMP)/UNICAMP/Desenhista de Páginas da Internet (Web Designer)/2023)

O sistema gerenciador de banco de dados MYSQL v.8.0 possui diversos programas clientes que se conectam ao servidor MySQL. Dentre tais programas clientes encontram- se

a) mysqldump e mysqlinit.

b) mysqlimport e mysqlstart.

c) mysqlpump e mysqldouble.

d) mysqladmin e mysqlcheck.

e) mysqlsh e mysqlpart.

17. (VUNESP - PTIC (UNICAMP)/UNICAMP/Desenhista de Páginas da Internet (Web Designer)/2023)

O comando do sistema gerenciador de banco de dados MYSQL v.8.0 que exibe os privilégios e papéis do usuário corrente de um banco de dados é:

a) SHOW GRANTS FOR CURRENT USER;

b) DISPLAYS GRANTS FOR CURRENT USER;

c) SET REVOKES FOR CURRENT USER;

d) MAKE GRANTS FOR CURRENT USER;

e) SHOW REVOKES FOR CURRENT USER;

18. (VUNESP - PTIC (UNICAMP)/UNICAMP/Desenhista de Páginas da Internet (Web Designer)/2023)

Considerando o sistema gerenciador de banco de dados MYSQL v.8.0 é possível determinar uma política global para a expiração do prazo de passwords. Para tanto, no arquivo my.cnf deve-se incluir a seguinte linha (considerando que tal prazo deva ser estipulado em 180 dias):

a) time_for_password=180

b) default_password_lifetime=180

c) lifetime_password_databases=180

d) default_time_password=180

e) make_password_time=180

19. (VUNESP - Ana Sis (CM SBO)/CM SBO/2023)

O Sistema Gerenciador da Banco de Dados MySQL 8.0 suporta cinco tipos de valores inteiros, sendo dois desses tipos

a) MININT e MAXINT.

b) TINYNT e BIGINT.

c) SINGLEINT e PLUSINT.

d) UNIINT e DOUBLEINT.

e) MICROINT e MACROINT.

20. (CESGRANRIO - Tec Cien (BASA)/BASA/Tecnologia da Informação/2022)

Em um servidor MySQL, qual log é usado para registrar mensagens de diagnóstico, como erros, avisos e notificações, que ocorrem durante a inicialização e o desligamento do servidor?

a) server log
b) error log
c) general query log
d) binary log
e) relay log

21. Cebraspe – Técnico Judiciário – Tecnologia da Informação (TRT-AP/PA)/2022

Para se realizar um backup lógico, por meio de um conjunto de instruções SQL que podem ser executadas para reproduzir as definições de objetos do banco de dados originais e os dados da tabela no MySql 8 de uma database chamada dbtrt, o comando correto é

a) mysqldump --databases dbtrt > backup.sql.

b) cp database dbtrt > backup.bkp.

c) mysqlbackup -db dbtrt > backup.sql.

d) mysqlbackup dbtrt –logical > backup.bkp.

e) backup -database dbtrt > backup.sql.

22. CESPE - Técnico Judiciário (STM)/Apoio Especializado/Programação de Sistemas/2018

Julgue o item, que dizem respeito aos SGBDs Oracle, MySQL e PostgreSQL.

Nas tabelas do tipo InnoDB do MySQL, o armazenamento dos dados pode ser realizado por um ou mais arquivos separados.

Certo Errado

23. CESPE - Oficial Técnico de Inteligência/Área 9/2018

Julgue o próximo item, a respeito de conceitos e comandos PostgreSQL e MySQL.

Por se tratar de um sistema gerenciador de banco de dados de código aberto, o MySQL não oferece suporte a conexões criptografadas entre clientes e o servidor.

Certo

Errado

24. CESPE - Técnico Judiciário (STJ)/Apoio Especializado/Suporte Técnico/2018

A respeito de sistemas gerenciadores de banco de dados (SGBD.), julgue o item.

Em um SGBD MySQL, o número de subqueries é limitado em até dois níveis em cada conjunto de instruções em um SELECT.

Certo

Errado

25. FCC - Analista Tecnológico (PRODATER)/Analista de Suporte Técnico/2016 Considere o comando do sistema gerenciador de bancos de dados MySQL (v. 5.6): SELECT FORMAT (3587.9, 2)

O resultado da execução desse comando é:

a) '3587.90'

b) '3587,90'

c) '3.587,90'

d) '3,587.90'

e) '3,587,90'

26. FCC - Analista de Tecnologia da Informação (CREMESP)/Administração de Banco de Dados/2016

O sistema de privilégios do MySQL garante que cada usuário possa fazer exatamente as operações as quais possui permissão.

O controle de acesso do MySQL é composto de dois estágios. No estágio

a) 1 o servidor pede uma senha de acesso e verifica se a senha confere usando a tabela passw_users.

b) 2 o servidor verifica se o usuário realizou um drop e, em caso positivo, cancela a ação do usuário pois é um comando proibido.

c) 1 o servidor utiliza apenas as tabelas user e db e no estágio 2 utiliza a tabela host_priv no banco de dados MySQL.

d) 2, se a solicitação envolver tabelas, o servidor pode consultar adicionalmente as tabelas tables_priv e columns_priv.

e) 1 o servidor verifica quais privilégios que o usuário possui antes de permitir que ele realize a primeira operação.

27. FCC - Analista de Tecnologia da Informação (CREMESP)/Administração de Banco de Dados/2016

O MySQL 5.7 possui diversas tabelas do sistema que contêm informações sobre contas de usuário e os privilégios por eles detidos. A tabela que contém os privilégios no nível

do Bando de Dados é

a) database_priv.

b) columns_priv.

c) procs_priv.

d) db.

e) proxies_priv.

28. CESPE - Analista de Tecnologia da Informação (FUB)/2015

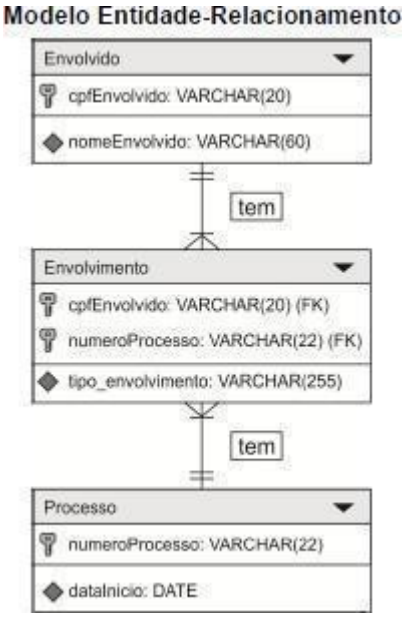
Com relação ao banco de dados MySQL, julgue o item subsequente.

No MySQL o uso do comando grant permite adicionar permissões a objetos do banco de dados. Certo

Errado

29. FCC - Analista Ministerial (MPE PB)/Analista de Sistemas/Administrador de Banco de Dados/2015

Atenção: Considere as informações a seguir para responder à questão.



Dados cadastrados nas tabelas:

cpfEnvolvido	nomeEnvolvido
121.134.045-01	Marcos Paulo
128.249.039-14	Maria de Fátima
131.091.431-09	André Luiz
158.245.067-12	Pedro nda Silva
160.234.074-11	João da Silva

Observação: O erro no nome Pedro nda Silva é proposital.

cpfEnvolvido	numeroProcesso	tipo_envolvimento
121.134.045-01	12345670020108261234	acusado
128.249.039-14	12345670020108261234	acusador
128.249.039-14	78654310020118150675	acusado
131.091.431-09	16789000020105131234	acusado
158.245.067-12	73982110120111131001	acusado
158.245.067-12	78654310020118150675	acusador
158.245.067-12	98006770120111144571	acusado

numeroProcesso	dataInicio
12345670020108261234	20/01/2015
16789000020105131234	08/08/2010
73982110120111131001	10/12/2014
78654310020118150675	30/01/2015
82745660320095202300	22/06/2014
98006770120111144571	21/02/2015

Considere que todos os números de processo tenham 20 dígitos na tabela Envolvimento. Dos 20 dígitos, 2 caracteres do número dos processos representam o número do Tribunal. Por exemplo, no processo de número 78654310020118150675 o número do Tribunal é 15.

Ao ser executada uma instrução SQL, foram exibidos os seguintes dados:

cpfEnvolvido	Tribunal	tipo_envolvimento
128.249.039-14	26	Acusador
158.245.067-12	15	Acusador

Considere a instrução SQL abaixo.

SELECT cpfEnvolvido, numeroProcesso

FROM Envolvimento

```
WHERE tipo_envolvimento = (
SELECT tipo_envolvimento FROM Envolvimento WHERE tipo_envolvimento ='acusador'
);
```

Ao executá-la no Oracle ou MySQL,

a) serão exibidos o CPF e o número do processo apenas dos registros cujo conteúdo do campo tipo_envolvimento possui o valor acusador.

b) ocorrerá um erro, pois a subconsulta retornará mais de uma linha.

c) para que não ocorra erro, o sinal de igual (=) deverá ser trocado pela palavra ANY.

d) serão exibidos todos os dados dos registros cujo conteúdo do campo tipo_envolvimento possui o valor acusador.

e) ocorrerá um erro, pois não é possível utilizar o resultado de uma instrução SELECT como parâmetro de comparação da cláusula WHERE de outra instrução SELECT.

30. CESPE - Analista Judiciário (TJDFT)/Apoio Especializado/Análise de Sistemas/2015

Um banco de dados permite à aplicação o armazenamento e a recuperação de dados com eficiência, o que garante segurança e integridade das informações. No caso de banco de dados relacional, os dados são armazenados em tabelas e os relacionamentos entre elas as tornam relacionais. A esse respeito, julgue o item que se segue.

Uma das principais desvantagens do banco de dados MySQL é que ele não possui suporte para triggers — gatilhos predefinidos e associados a tabelas — disparados por algum evento específico.

Certo

Errado

31. CESPE - Técnico Judiciário (TJDFT)/Apoio Especializado/Programação de Sistemas/2015

Acerca do PostgreSQL e do MySQL, julgue o seguinte item.

O comando SELECT FROM CAT é utilizado para que sejam apresentados todos os bancos de dados disponíveis no servidor MySQL.

Certo

Errado

32. Ano: 2018 Banca: CESPE Órgão: STM Cargo: Analista de Sistemas Questão: 51 a 54

```
1. CREATE TABLE IF NOT EXISTS 'software' (
2. 'id' int NOT NULL,
```

```
3. 'nome' varchar(70) NOT NULL,
4. PRIMARY KEY ('id')
5. ) DEFAULT CHARSET=utf8;
6. INSERT INTO 'software' ('id', 'nome') VALUES
7. ('1', 'Programa ABC'),
8. ('2', 'Programa WYZ'),
9. ('3', 'Programa DFG');
10. CREATE TABLE IF NOT EXISTS 'vsoftware' (
11. 'idsoft' int UNSIGNED NOT NULL REFERENCES software(id),
12. 'versao' int(3) NOT NULL,
13. 'descricao' varchar(70) NOT NULL,
14. PRIMARY KEY ('idsoft','versao'));
15. INSERT INTO 'vsoftware'('idsoft', 'versao', 'descricao') VALUES
16.     ('1', '1', 'criacao do programa.'),
17. ('2', '1', '1a versao.'),
18. ('1', '2', 'atualizacao na tela A.'),
19. ('1', '3', 'adicao da tela C.'),
20. ('2', '2', 'adicao da tela D.');
```

Com base nos comandos MySQL 5.6 precedentes, julgue os itens a seguir.

75 Especialmente devido à expressão na linha 11, o comando a seguir, após executado, retornará três registros.

```
SELECT a.idsoft, a.versao, a.descricao
FROM 'vsoftware' a
INNER JOIN (
 SELECT idsoft, MAX(versao) versao
 FROM 'vsoftware'
 GROUP BY idsoft) b ON a.idsoft = b.idsoft
AND a.versao = b.versao;
```

76 A execução do comando

```
SELECT a.idsoft, a.versao, a.descricao
FROM 'vsoftware' a
LEFT OUTER JOIN 'vsoftware' b
 ON a.idsoft = b.idsoft AND a.versao < b.versao
```

WHERE b.idsoft IS NULL;

retornará os seguintes dados.

idsoft	versao	descricao
1	3	adicao da tela C.
2	2	adicao da tela D.

77 A execução do comando

```
SELECT a.id
FROM 'software' a
where a.id not in (
SELECT b.idsoft
FROM 'vsoftware' b);
```

terá resultado idêntico à execução do comando a seguir.

```
SELECT distinct a.id
FROM 'software' a left outer join 'vsoftware' b
on a.id=b.idsoft;
```

78 A tabela vsoftware está na segunda forma normal (2FN), porque contém uma chave estrangeira referenciada à tabela software.

79 O comando ALTER TABLE software add data datetime; no MySQL 5.6 adiciona um novo campo data à tabela software.

80 O comando a seguir no MySQL 5.6 modifica o tipo do campo nome para CHAR na tabela software.

```
ALTER TABLE software MODIFY nome char(100);
```

33. Ano: 2018 Banca: CESPE Órgão: STM Cargo: Analista de Sistemas Questão: 71 a 74

Um sistema gerenciador de banco de dados (SGBD) instalado no Linux deve ser configurado de modo a permitir os seguintes requisitos:

I no máximo, 1000 conexões simultâneas;

II somente conexões originadas a partir do servidor de aplicação com IP 10.10.10.2.

Tendo como referência essas informações, julgue os seguintes itens.

73 Caso o SGBD instalado seja o MySQL 5.7, para atendimento dos requisitos I e II, deve-se modificar o arquivo my.cnf, alterando-se os parâmetros max_user_connections para 1000 e connection_source para o IP fornecido; e reiniciar o serviço do SGBD.

34. Ano: 2017 Banca: FCC Órgão: ARTESP Cargo: Especialista em Regulação de Transporte - Tecnologia da Informação Questão: 72.

72. O sistema gerenciador de bancos de dados MySQL (versão 5.6 e posteriores) admite o bloqueio e o desbloqueio para acesso às tabelas. Os dois comandos utilizados para essas funções de bloqueio e desbloqueio de tabelas são, respectivamente,

(A) GRANT e REVOKE.

(B) COMMIT e ROLLBACK.

(C) OPEN e CLOSE.

(D) LOCK e UNLOCK.

(E) START e END.

35. Ano: 2017 Banca: FCC Órgão: TRE-SP Cargo: Técnico Judiciário de TI – Operação de computadores

58. Resumidamente e, ainda, considerando todas as condições e pré-condições de ambiente já existentes para garantir o funcionamento adequado de um banco de dados, um Técnico, usando MySQL 5.6, escreveu as seguintes expressões e comandos SQL:

CREATE TABLE NAMES (Id integer PRIMARY KEY, Name

text); INSERT INTO NAMES VALUES(1,'Tom');

INSERT INTO NAMES

VALUES(2,'Lucy'); INSERT INTO

NAMES VALUES(3,'Frank'); INSERT

INTO NAMES VALUES(4,'Jane');

INSERT INTO NAMES

VALUES(5,'Robert');

SELECT Name FROM NAMES WHERE Id = 3 or Id = 5;

Quanto à construção do Schema e quanto à execução do Select, este, presumidamente, especificado para recuperar Frank e Robert, o resultado foi

(A) erro, porque todos os Insert deveriam ter sido escritos seguindo a sintaxe exemplo INSERT INTO NAMES VALUES OF (1,'Tom').

(B) erro, porque o SQL deveria ter sido executado como SELECT Name FROM NAMES WHERE Id = 3 or = 5.

(C) a recuperação de Frank e Robert.

(D) erro, porque o CREATE deveria ter sido escrito CREATE TABLE NAMES (Id integer PRIMARY KEY, For Name text).

(E) a recuperação de 3 Frank e 5 Robert.

36. Ano: 2016 Banca: CESPE Órgão: TCE-SC Cargo: Auditor de TI

Acerca dos sistemas gerenciadores de banco de dados MySQL e PostgreSQL, julgue os itens subsequentes.

61 Na administração do MySQL 5.6, ao se executar o becape full lógico por meio do aplicativo mysqldump, recomenda-se verificar a integridade dos dados antes de realizar o becape. Nesse caso, a sintaxe correta é a mostrada a seguir. mysqldump -u usuario -p senha --check-all-db -- all-databases > arquivo.sql

37. BANCA: FGV ANO: 2013 ÓRGÃO: AL-MT PROVA: ANALISTA DE SISTEMAS - BANCO DE DADOS

No MySQL 5.1 ou mais recentes, os tipos padrão de índices para o storage engine do tipo MEMORY/HEAP são:

A B+TREE e B*TREE.

B BTREE e B+TREE.

C B*TREE e HASH.

D B+TREE e HASH.

E BTREE e HASH.

38. BANCA: FGV ANO: 2013 ÓRGÃO: AL-MT PROVA: ANALISTA DE SISTEMAS - BANCO DE DADOS

O modo recomendado para iniciar o servidor MySQL no UNIX é através do

script

A mysql_up.

B mysql_startup.

C mysqld_safe.

D mysqld.

E mysqld_on.

39. BANCA: FGV ANO: 2013 ÓRGÃO: AL-MT PROVA: ANALISTA DE SISTEMAS - BANCO DE DADOS

O formato físico utilizado pelas versões 5.1 e para as mais recentes do MySQL, para representar suas tabelas, é o

A .tab.

B .mys.

C .frm.

D .sql.

E .ino.

As questões abaixo têm o foco maior no desenvolvimento de banco de dados usando MySQL.

40. BANCA: FCC ANO: 2013 ÓRGÃO: MPE-CE PROVA: ANALISTA MINISTERIAL - CIÊNCIAS DA COMPUTAÇÃO

A associação entre o comando do banco de dados MySQL e sua descrição é expressa corretamente em:

A

Mudar de base de dados	Ver os formatos dos campos da tabela	Apagar uma base de dados	Atualizar permissões e privilégios de uma base de dados
change [nome bd];	check [nome tabela];	drop database [nome bd];	UPDATE PRIVILEGES;

B

Mudar de base de dados	Ver os formatos dos campos da tabela	Apagar uma base de dados	Atualizar permissões e privilégios de uma base de dados
use [nome bd];	describe [nome tabela];	drop database [nome bd];	FLUSH PRIVILEGES;

C

Mudar de base de dados	Ver os formatos dos campos da tabela	Apagar uma base de dados	Atualizar permissões e privilégios de uma base de dados
change [nome bd];	show [nome tabela];	delete database [nome bd];	set privileges;

D

Mudar de base de dados	Ver os formatos dos campos da tabela	Apagar uma base de dados	Atualizar permissões e privilégios de uma base de dados
show [nome bd];	list [nome tabela];	drop tables [nome bd];	list privileges;

E

Mudar de base de dados	Ver os formatos dos campos da tabela	Apagar uma base de dados	Atualizar permissões e privilégios de uma base de dados
use [nome bd];	show [nome tabela];	erase database [nome bd];	flush privileges;

41. BANCA: CESPE ANO: 2014 ÓRGÃO: ANATEL PROVA: ANALISTA ADMINISTRATIVO - SUPORTE E INFRAESTRUTURA DE TI

A respeito de banco de dados, julgue os itens que se seguem.

[1] Considere que haja necessidade de se criar uma tabela T1 com uma coluna do tipo ENUM que armazene uma lista permitida de dados em formato string. Suponha ainda que essa operação deva ser feita no Mysql 5.7. Nesse caso, o comando a ser dado seria o seguinte:

CREATE TABLE T1 (COL1 ENUM('1', '2', '3'));

42. BANCA: FGV ANO: 2013 ÓRGÃO: AL-MT PROVA: ANALISTA DE SISTEMAS - BANCO DE DADOS

No trecho de código SQL a seguir está representado um conjunto de declarações que foram executadas por um programador no MySQL.

```
CREATE TABLE TAB1 (MAT INT(10) NOT NULL, NOME
VARCHAR(4) NOT NULL, FONE VARCHAR(9), PRIMARY KEY
(MAT));
INSERT INTO TAB1 VALUES (0123456789, 'ABC',
'XYZ');
CREATE VIEW V1 AS SELECT NOME, FONE, FONE+NOME AS
VALUE FROM TAB1;
SELECT * FROM V1;
```

O resultado da declaração SELECT corresponde, respectivamente, a

A

NOME	FONE	VALUE
ABC	XYZ	XYZABC

B

NOME	FONE	VALUE
ABC	XYZ	ABCXYZ

C

NOME	FONE	VALUE
ABC	XYZ	-------

D

NOME	FONE	VALUE
ABC	XYZ	.....

E

NOME	FONE	VALUE
ABC	XYZ	0

GABARITO

1. A
2. E
3. D
4. C
5. B
6. B
7. E
8. A
9. C
10. D
11. B
12. B
13. A
14. C
15. A
16. D
17. A
18. B
19. B
20. B
21. A
22. C
23. E
24. E
25. D
26. D
27. D
28. C
29. B
30. E
31. E
32. E C E E C C
33. E
34. D
35. C
36. E
37. E
38. C
39. C
40. B

41. C
42. E

ESSA LEI TODO MUNDO CONHECE:
PIRATARIA É CRIME.

Mas é sempre bom revisar o porquê e como você pode ser prejudicado com essa prática.

1. 

Professor investe seu tempo para elaborar os cursos e o site os coloca à venda.

2. 

Pirata divulga ilicitamente (grupos de rateio), utilizando-se do anonimato, nomes falsos ou laranjas (geralmente o pirata se anuncia como formador de "grupos solidários" de rateio que não visam lucro).

3. 

Pirata cria alunos fake praticando falsidade ideológica, comprando cursos do site em nome de pessoas aleatórias (usando nome, CPF, endereço e telefone de terceiros sem autorização).

4. 

Pirata compra, muitas vezes, clonando cartões de crédito (por vezes o sistema anti-fraude não consegue identificar o golpe a tempo).

5. 

Pirata fere os Termos de Uso, adultera as aulas e retira a identificação dos arquivos PDF (justamente porque a atividade é ilegal e ele não quer que seus fakes sejam identificados).

6. 

Pirata revende as aulas protegidas por direitos autorais, praticando concorrência desleal e em flagrante desrespeito à Lei de Direitos Autorais (Lei 9.610/98).

7. 

Concurseiro(a) desinformado participa de rateio, achando que nada disso está acontecendo e esperando se tornar servidor público para exigir o cumprimento das leis.

8. 

O professor que elaborou o curso não ganha nada, o site não recebe nada, e a pessoa que praticou todos os ilícitos anteriores (pirata) fica com o lucro.

Deixando de lado esse mar de sujeira, aproveitamos para agradecer a todos que adquirem os cursos honestamente e permitem que o site continue existindo.