CAPÍTULO 1
Arquitetura do MySQL

A arquitetura do MySQL é muito diferente da dos outros servidores de banco de dados e é útil para uma grande variedade de objetivos. MySQL não é perfeito, mas é flexível o suficiente para trabalhar bem em ambientes muito exigentes, como aplicações web. Ao mesmo tempo, MySQL pode potencializar aplicações embutidas, depósitos de dados, indexação de conteúdo e software de distribuições, sistemas redundantes altamente disponíveis, processamento de transação on-line (OLTP), e muito mais.

Para obter o máximo do MySQL, você precisa entender seu design para que você possa trabalhar com ele, não contra ele. MySQL é flexível de diversas maneiras. Por exemplo, você pode configurá-lo para rodar bem em ampla variedade de hardware e suportar uma variedade de tipos de dados. Porém, a característica mais importante e incomum do MySQL é sua arquitetura de ferramenta de armazenamento, cujo design separa o processamento da consulta e outras tarefas do servidor do armazenamento de dados e recuperação. No MySQL 5.1, você pode até carregar ferramentas de armazenamento como plugins em tempo de execução. Esta separação de interesses permite que você escolha, em uma base por tabela, como seus dados são armazenados e qual desempenho, atributos e outras características você quer.

Este capítulo oferece uma visão geral de alto nível da arquitetura do servidor MySQL, as maiores diferenças entre as ferramentas de armazenamento, e por que estas diferenças são importantes. Nós tentamos explicar o MySQL simplificando os detalhes e mostrando exemplos. Esta discussão será útil para aqueles novos em servidores de banco de dados, assim como para os leitores que são especialistas com outros servidores de banco de dados.

Arquitetura Lógica do MySQL

Uma boa imagem mental de como os componentes do MySQL trabalham juntos vai te ajudar a entender o servidor. A Figura 1-1 mostra uma visão lógica da arquitetura do MySQL.

A camada superior contém os serviços que não são exclusivos do MySQL. Eles são serviços que a maioria das ferramentas cliente/servidor baseadas em rede ou servidores precisa: gerenciamento de conexão, autenticação, segurança e assim por diante.

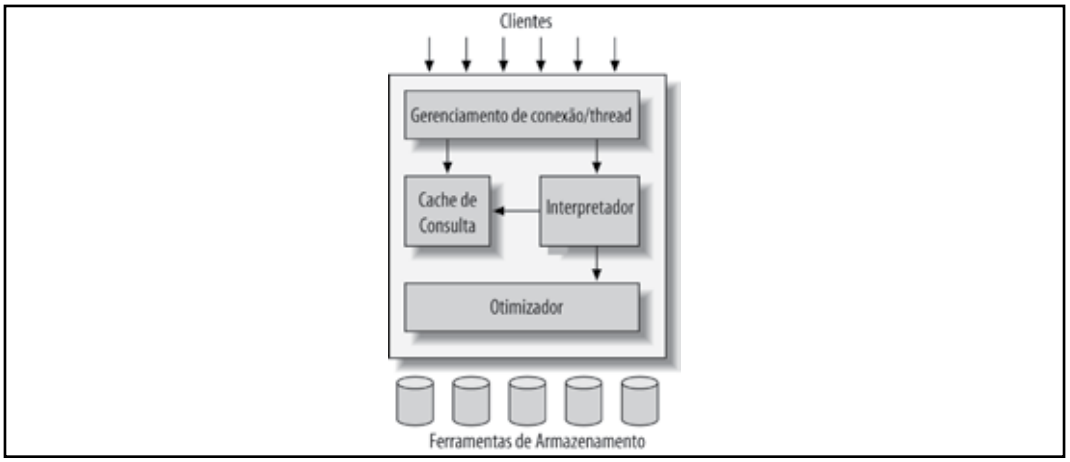

Figura 1-1. Uma visão lógica da arquitetura do servidor MySQL.

A segunda camada é onde as coisas ficam interessantes. A maior parte do cérebro do MySQL está aqui, incluindo o código para interpretar consulta, análise, otimização, cache e todas as funções embutidas (por exemplo, datas, horas, matemática e codificação). Toda funcionalidade oferecida através das ferramentas de armazenamento fica neste nível: procedimentos armazenados, triggers e visualizações, por exemplo.

A terceira coluna contém as ferramentas de armazenamento. Elas são responsáveis por armazenar e recuperar todos os dados "no" MySQL. Como os diversos sistemas de arquivo disponíveis para GNU/Linux, cada ferramenta de armazenamento possui seus próprios benefícios e desvantagens. O servidor comunica-se com elas através da API da ferramenta de armazenamento. Esta interface esconde diferenças entre ferramentas de armazenamento e as tornam amplamente transparentes na camada de consulta. A API contém várias funções de baixo nível que realiza operações como "iniciar uma transação" ou "trazer a linha que contém a linha primária". As ferramentas de armazenamento não interpretam SQL* nem se comunicam umas com as outras; elas simplesmente respondem às requisições do servidor.

Gerenciamento de Conexão e Segurança

Cada conexão de cliente obtém sua própria thread dentro do processo do servidor. As consultas da conexão dentro daquela única thread, que consequentemente habita em uma central ou CPU. O servidor faz cache de threads, para que elas não precisem ser criadas e destruídas para cada nova conexão**.

Quando clientes (aplicações) conectam-se ao servidor MySQL, o servidor precisa autenticá-las. Autenticação é baseada no nome do usuário, hospedeiro de origem e senha. Certificados X.509 também podem ser usados através de uma conexão Secure Sockets Layer (SSL – Camadas de Tomadas Seguras). Depois de um cliente conectar-se, o servidor verifica se o cliente tem privilégios para cada consulta que ele executa (exemplo: se o cliente tem permissão para executar uma expressão SELECT que acessa a tabela Country no banco de dados world). Nós abordamos estes tópicos em detalhes, no Capítulo 12.

* Uma exceção é InnoDB, que interpreta definições de chave estrangeira, porque o servidor MySQL ainda não as implementa.

** MySQL AB planeja separar conexões das threads em uma futura versão do servidor.

Otimização e Execução

MySQL interpreta consultas para criar uma estrutura interna (a árvore de interpretação), e, então, aplica uma variedade de otimizações. Estas podem incluir reescrever a consulta, determinar a ordem na qual ela vai ler as tabelas, escolher quais índices usar e assim por diante. Você pode passar dicas ao otimizador por meio de palavras-chave especiais na consulta, afetando seu processo de tomada de decisão. Você também pode pedir ao servidor para explicar vários aspectos da otimização. Isso permite que você saiba quais decisões o servidor está tomando e dá a você um ponto de referência para repetir consultas, esquemas e configurações para fazer tudo rodar o mais eficientemente possível. Nós discutimos o otimizador com mais detalhes no Capítulo 4.

O otimizador realmente não se importa por qual ferramenta de armazenamento uma tabela particular usa, mas a ferramenta de armazenamento realmente afeta como o servidor otimiza a consulta. O otimizador pergunta à ferramenta de armazenamento sobre algumas das suas capacidades e o custo de certas operações, e pede por estatísticas sobre os dados da tabela. Por exemplo, algumas ferramentas de armazenamento suportam tipos de índice que podem ser úteis para certas consultas. Você pode ler mais sobre indexação e otimização de esquema no Capítulo 3.

Antes mesmo de interpretar a consulta, porém, o servidor consulta o cache de consulta, que pode armazenar somente expressões SELECT, junto com seus conjuntos de resultado. Se alguém executar uma consulta que é idêntica àquela que já está no cache, o servidor não precisa interpretar, otimizar ou executar a consulta – ele pode simplesmente passar de volta o conjunto de resultado armazenado! Nós discutimos o cache de consulta ao longo de "O Cache de Consulta do MySQL", na página 171.

Controle de Concorrência

Toda vez que mais de uma consulta precisar alterar dados ao mesmo tempo, o problema de controle de concorrência surge. Para nossos propósitos neste capítulo, MySQL tem de fazer isso em dois níveis: no nível do servidor e no nível da ferramenta de armazenamento. Controle de concorrência é um grande tópico para o qual um grande corpo da literatura teórica é devotado, mas este livro não é sobre teoria ou, até mesmo, sobre as internalidades do MySQL. Assim, nós vamos dar a você somente uma visão geral simplificada de como MySQL lida com leitores e escritores concorrentes, então você terá o contexto necessário para o resto deste capítulo.

Nós usaremos uma caixa de e-mail em um sistema Unix como um exemplo. O clássico formato de arquivo mbox é muito simples. Todas as mensagens na caixa de e-mail mbox são concatenadas juntas, uma depois da outra. Isso torna muito fácil ler e interpretar mensagens de e-mail. Também torna a entrega do e-mail mais fácil: apenas anexe uma nova mensagem no final do arquivo.

Mas o que acontece quando dois processos tentam entregar mensagens ao mesmo tempo na mesma caixa de e-mail? Claramente isso pode corromper a caixa de e-mail, deixando duas mensagens vazias no final do arquivo da caixa de e-mail. Sistemas de entrega de e-mail bem comportados usam bloqueio para evitar corrupção. Se um cliente tentar uma entrega secundária enquanto a caixa de e-mail estiver bloqueada, ele deve esperar para adquirir o próprio bloqueio antes de entregar sua mensagem.

Este esquema funciona razoavelmente bem na prática, mas não dá nenhum suporte para concorrência. Porque somente um único processo pode modificar a caixa de e-mail a qualquer dado horário, esta abordagem torna-se problemática com uma caixa de entrada de alto volume.

Bloqueios de Leitura/Escrita

Ler a partir da caixa de e-mail não é um problema. Não há nada de errado em múltiplos clientes lerem a mesma caixa de e-mail simultaneamente; porque eles não estão fazendo alterações, não há probabilidade de algo dar errado. Mas o que acontece se alguém tentar deletar a mensagem número 25 enquanto programas estão lendo a caixa de e-mail? Depende, mas um leitor pode ter uma visualização corrompida ou

inconsistente da caixa de entrada. Então, por segurança, até mesmo ler a partir de uma caixa de e-mail requer cuidado especial.

Se você considera a caixa de e-mail como uma tabela de banco de dados e cada mensagem de e-mail com uma linha, é fácil ver que o problema é o mesmo neste contexto. De muitas maneiras, uma caixa de e-mail é, realmente, uma simples tabela de banco de dados. Modificar linhas em uma tabela de banco de dados é muito similar a remover ou modificar o conteúdo das mensagens no arquivo de caixa de e-mail.

A solução para este problema clássico de controle de concorrência é muito simples. Sistemas que lidam com acesso de leitura/escrita concorrente tipicamente implementam um sistema de bloqueio que consiste em dois tipos de bloqueio. Estes são geralmente conhecidos como bloqueios compartilhados e bloqueios exclusivos, ou bloqueios de leitura e bloqueios de escrita.

Sem nos preocuparmos sobre a atual tecnologia de bloqueio, podemos descrever o conceito como segue. Bloqueios de leitura em um recurso são compartilhados, ou mutuamente não-bloqueáveis: muitos clientes podem ler a partir de um recurso ao mesmo tempo e não interferir uns com os outros. Bloqueios de escrita, por outro lado, são exclusivos – por exemplo, eles bloqueiam leitura e outros bloqueiam escritas – porque a única política segura é ter um único cliente escrevendo no recurso em um dado tempo e evitar todas as leituras quando um cliente estiver escrevendo.

No mundo do banco de dados, bloqueio ocorre o tempo todo: MySQL tem de impedir que um cliente leia uma porção de dados enquanto outro esteja modificando-a. Ele executa este gerenciamento de bloqueio internamente de uma maneira que seja transparente a maior parte do tempo.

Granularidade de Bloqueio

Uma maneira de melhorar a concorrência de um recurso compartilhado é ser mais seletivo sobre o que você bloqueia. Em vez de bloquear todo o recurso, bloqueie somente a parte que contém os dados que você necessita alterar. Melhor ainda, bloqueie somente a porção exata dos dados que planeja alterar. Minimizar a quantidade de dados que você bloqueia de uma vez permite que alterações a um dado recurso ocorram simultaneamente, desde que elas não se conflitem com as outras.

O problema é que bloqueios consomem recursos. Toda operação de bloqueio – obter um bloqueio, checar para ver se um bloqueio está liberado, liberar um bloqueio e assim por diante – causa problemas. Se o sistema gastar muito tempo gerenciando bloqueios em vez de armazenar e restaurar dados, o desempenho pode sofrer.

Uma estratégia de bloqueio é um acordo entre problemas de bloqueio e segurança de dados, e esse acordo afeta desempenho. A maioria dos servidores de banco de dados comerciais não dá a você muita escolha: você obtém o que é conhecido como bloqueio a nível de linha nas suas tabelas, com uma variedade de maneiras normalmente complexas para dar bom desempenho com muitos bloqueios.

MySQL, por outro lado, oferece escolhas. Sua ferramenta de armazenamento pode implementar suas próprias políticas de bloqueio e granularidades de bloqueio. Gerenciamento de bloqueio é uma decisão muito importante no design de ferramenta de armazenamento; fixar a granularidade em certo nível pode gerar melhor desempenho para certos usos, embora torne esta ferramenta menos adequada para outros propósitos. Por MySQL oferecer múltiplas ferramentas de armazenamento, ele não exige uma única solução de objetivo geral. Vamos dar uma olhada nas duas mais importantes estratégias de bloqueio.

Bloqueios de tabela

A estratégia mais básica de bloqueio disponível no MySQL, e a com menor risco de problemas, é bloqueios de tabela. Um bloqueio de tabela é similar aos bloqueios de caixa de e-mail descritas anteriormente: ele bloqueia a tabela inteira. Quando um cliente deseja escrever em uma tabela (inserir, deletar, atualizar etc), ele adquire um bloqueio de escrita. Isso coloca todas as outras operações de leitura e escrita

em dificuldades. Quando ninguém está escrevendo, leitores podem obter bloqueios de leitura, que não entram em conflito com outros bloqueios de leitura.

Bloqueios de tabela possuem variações para bom desempenho em situações específicas. Por exemplo, os bloqueios de tabela READ LOCAL permitem alguns tipos de operações de escrita concorrentes. Bloqueios de escrita também têm uma prioridade maior do que bloqueios de leitura, então uma solicitação de um bloqueio de escrita vai avançar para a frente da fila de bloqueio mesmo se os leitores já estiverem na fila (bloqueios de escrita podem passar bloqueios de escrita na fila, mas bloqueios de escrita não podem passar adiante dos bloqueios de escrita).

Embora ferramentas de armazenamento possam gerenciar seus próprios bloqueios, MySQL usa também uma variedade de bloqueios que estão efetivamente em nível de tabela para vários propósitos. Por exemplo, o servidor usa um bloqueio em nível de tabela para expressões como ALTER TABLE, independente da ferramenta de armazenamento.

Bloqueios de linha

O estilo de bloqueio que oferece a maior concorrência (e carrega o maior risco de problemas) é o uso de bloqueios de linha. Bloqueio em nível de linha, como esta estratégia é conhecida, está disponível nas ferramentas de armazenamento InnoDB e Falcon, entre outros. Bloqueios de linha são implementadas na ferramenta de armazenamento, não no servidor (referencie-se de volta ao diagrama de arquitetura lógica se você precisar). O servidor é completo inconsciente de bloqueios implementados nas ferramentas e, como você verá mais tarde neste capítulo e ao longo do livro, as ferramentas de armazenamento todas implementam bloqueio nas suas próprias maneiras.

Transações

Você não pode examinar as características mais avançadas de um sistema de banco de dados por muito tempo antes de transações se incorporarem. Uma transação é um grupo de consultas SQL que são tratadas atomicamente, como uma unidade única de trabalho. Se a ferramenta de banco de dados puder aplicar o grupo inteiro de consultas a um banco de dados, ela vai fazer, mas, se alguma delas não puder ser feita por causa de um travamento ou outra razão, nenhuma delas é aplicada. É tudo ou nada.

Um pouco desta seção é específica ao MySQL. Se você já for familiar com transações ACID, sinta-se à vontade para pular para "Transações no MySQL", na página 9, adiante neste capítulo.

Uma aplicação bancária é o exemplo clássico de porque transações são necessárias. Imagine o banco de dados de um banco com duas tabelas: checking e savings. Para mover $200 da conta de Jane para sua conta poupança, você precisa executar pelo menos três passos:

1. Tenha certeza de que o saldo da sua conta seja maior do que $200.
2. Subtraia $200 do saldo da sua conta.
3. Adicione $200 ao saldo da sua conta poupança.

Toda a operação deve ocorrer em uma transação para que se algum dos passos falhar, qualquer passo completado possa ser executado novamente.

Você inicia uma transação com a expressão START TRANSACTION e, então, torna suas alterações permanentes com COMMIT ou descarta as alterações com ROLLBACK. Então, o SQL para nosso exemplo de transação deve parecer assim:

```
1 START TRANSACTION;
2 SELECT balance FROM checking WHERE customer_id = 10233276;
3 UPDATE checking SET balance = balance - 200.00 WHERE customer_id = 10233276;
4 UPDATE savings SET balance = balance + 200.00 WHERE customer_id = 10233276;
5 COMMIT;
```

Mas transações sozinhas não são a história toda. O que acontece se o servidor de banco de dados travar enquanto executa a linha 4? O cliente provavelmente acabou de perder $200. E se outro processo vier entre as linhas 3 e 4 e remover todo o saldo da conta? O banco deu ao cliente um crédito de $200 sem nem mesmo saber.

Transações não são suficientes a menos que o sistema passe no teste ACID. ACID significa Atomicidade, Consistência, Isolamento e Durabilidade. Estes são os critérios muito relacionados que um sistema de processamento de transação bem comportado precisa se adaptar:

Atomicidade

Uma transação deve funcionar como uma única unidade indivisível de trabalho para que a transação inteira seja aplicada ou executada de volta. Quando transações são atômicas, não existe transação parcialmente completada: é tudo ou nada.

Consistência

O banco de dados deve sempre mover de um estado consistente para o próximo. No nosso exemplo, consistência garante que um travamento entre as linhas 3 e 4 não resulte em $200 desaparecendo da conta. Porque a transação nunca é realizada, nenhuma das alterações da transação é refletida no banco de dados.

Isolamento

Os resultados de uma transação são, geralmente, invisíveis para outras transações até que a transação esteja completa. Isso garante que se o resumo de uma conta bancária executar depois da linha 3, mas antes da linha 4 no nosso exemplo, ele ainda vai ver os $200 na conta. Quando discutimos níveis de isolamento, você entenderá por que dissemos geralmente invisíveis.

Durabilidade

Depois de realizadas, as alterações de uma transação são permanentes. Isso significa que as alterações precisam ser registradas para que estes dados não fiquem perdidos em um travamento do sistema. Durabilidade é um conceito um pouco confuso, porém, porque há, geralmente, muitos níveis. Algumas estratégias de durabilidade oferecem uma garantia de segurança mais forte do que outras, e nada é 100% durável. Nós discutimos o que durabilidade realmente significa no MySQL em capítulos futuros, especialmente em "Ajustes de E/S do InnoDB", na página 236.

Transações ACID garantem que bancos não percam seu dinheiro. É, geralmente, extremamente difícil ou impossível fazer isso com esta lógica de aplicação. Um servidor de banco de dados compatível a ACID tem de fazer todos os tipos de coisas complicadas que você pode não perceber para oferecer garantias ACID.

Assim como granularidade de bloqueio aumentada, a desvantagem desta segurança extra é que o servidor de banco de dados tem de fazer mais trabalho. Um servidor de banco de dados com transações ACID também geralmente requer mais potência de CPU, memória e espaço de disco do que um sem elas. Como dissemos diversas vezes, aqui é onde a arquitetura da ferramenta de armazenamento do MySQL trabalha para sua vantagem. Você pode decidir se sua aplicação precisa de transações. Se você realmente não precisar delas, você pode ser capaz de obter desempenho melhor com uma ferramenta de armazenamento não-transacional para alguns tipos de consultas. Você pode ser capaz de usar `LOCK TABLES` para dar o nível de proteção que você precisa sem transações. Tudo depende de você.

Níveis de Isolamento

Isolamento é mais complexo do que parece. O padrão SQL define quatro níveis de isolamento, que regras específicas para quais alterações são e não são visíveis dentro e fora de uma transação. Níveis mais baixos de isolamento tipicamente permitem maior concorrência e possuem menos riscos de problemas.

Cada ferramenta de armazenamento implementa níveis de isolamento de uma maneira um pouco diferente, e elas não necessariamente se adaptam ao que você pode esperar se você estiver acostumado com outro produto de banco de dados (assim, não entraremos em exaustivos detalhes nesta seção). Você deve ler os manuais para qualquer ferramenta de armazenamento que você decidir usar.

Vamos dar uma rápida olhada nos quatro níveis de isolamento:

`READ UNCOMMITTED`

No nível de isolamento `READ UNCOMMITTED`, transações podem visualizar os resultados de transações não comitadas. Neste nível, muitos problemas podem ocorrer, a menos que você realmente saiba o que está fazendo e tenha uma boa razão para fazê-lo. Este nível é raramente usado na prática, porque seu desempenho não é muito melhor do que os outros níveis, que possuem muitas vantagens. Ler dados não comitados é também conhecido como leitura suja.

`READ COMMITTED`

O nível de isolamento padrão para a maioria dos sistemas de banco de dados (mas não o MySQL!) é `READ COMMITTED`. Ele satisfaz a simples definição de isolamento usada anteriormente: uma transação verá somente aquelas alterações feitas por transações que já eram comitadas quando ela iniciou, e suas alterações são serão visíveis para outras até que ela seja comitada. Este nível ainda permite o que é conhecido como leitura não repetível. Isso significa que você pode executar a mesma expressão duas vezes e ver dados diferentes.

`REPEATABLE READ`

REPEATABLE READ resolve os problemas que `READ UNCOMMITTED` permite. Ele garante que qualquer linha que uma transação ler vai "parecer a mesma" em leituras subsequentes dentro da mesma transação, mas na teoria ainda permite outro problema chato: leituras fantasmas. Simplificando, uma leitura fantasma pode acontecer quando você seleciona alguns intervalos de linhas, outra transação insere uma nova linha no intervalo e, então, você seleciona o mesmo intervalo novamente; você verá então a nova linha "fantasma". InnoDB e Falcon resolvem o problema de leitura fantasma com controle de concorrência de versões múltiplas, que explicaremos mais tarde neste capítulo.

`REPEATABLE READ` é o nível de isolamento de transação padrão do MySQL. As ferramentas de armazenamento InnoDB e Falcon respeitam esta configuração, a qual você aprenderá a modificar, no Capítulo 6. Algumas outras ferramentas de armazenamento também fazem isso, mas a escolha depende da ferramenta.

`SERIALIZABLE`

O mais alto nível de isolamento, `SERIALIZABLE`, resolve o problema da leitura fantasma forçando as transações a serem ordenadas para que eles não possam possivelmente se conflitar. Em poucas palavras, `SERIALIZABLE` coloca um bloqueio em cada linha que lê. Neste nível, muitos intervalos e contenção de bloqueio podem ocorrer. Nós raramente vemos pessoas usarem este nível de isolamento, mas as necessidades da sua aplicação podem te forçar a aceitar a baixa concorrência em favor da estabilidade dos dados em que resulta.

A Tabela 1-1 resume os vários níveis de isolamento e as desvantagens associadas com cada um.

Tabela 1-1. Níveis de isolamento ANSI SQL

Nível de Isolamento	Possíveis leitura sujas	Possíveis leituras não repetíveis	Phantom reads Possible	Leituras de bloqueio
READ ANCOMMITED	Sim	Sim	Sim	Não
READ COMMITED	Não	Sim	Sim	Não
REPEATABLE READ	Não	Não	Sim	Não
SERIALIZABLE	Não	Não	Não	Sim

Deadlocks

Um deadlock é quando duas ou mais transações estão mutuamente mantendo e solicitando bloqueios nos mesmos recursos, criando um ciclo de dependências. Deadlocks ocorrem quando transações tentam bloquear recursos em uma ordem diferente. Eles podem acontecer sempre que múltiplas transações bloqueiam os mesmos recursos. Por exemplo, considere estas duas transações sendo executadas na tabela StockPrice:

Transação nº.1

```
START TRANSACTION;
UPDATE StockPrice SET close = 45.50 WHERE stock_id = 4 and date = '2002-05-01';
UPDATE StockPrice SET close = 19.80 WHERE stock_id = 3 and date = '2002-05-02';
COMMIT;
```

Transação nº.2

```
START TRANSACTION;
UPDATE StockPrice SET high = 20.12 WHERE stock_id = 3 and date = '2002-05-02';
UPDATE StockPrice SET high = 47.20 WHERE stock_id = 4 and date = '2002-05-01';
COMMIT;
```

Se você não tiver sorte, cada transação vai executar sua primeira consulta e atualizar uma linha de dados, bloqueando-a no processo. Cada transação vai, então, tentar atualizar sua segunda linha, apenas para descobrir que já está bloqueada. As duas transações vão esperar eternamente para que cada uma complete, a menos que algo interfira para quebrar o deadlock.

Para combater este problema, sistemas de banco de dados implementam várias formas de detecção de deadlock e intervalos. Os sistemas mais sofisticados, como a ferramenta de armazenamento InnoDB, vai perceber dependências circulares e retornar um erro instantaneamente. Isso é, na verdade, algo muito bom – do contrário, deadlocks iriam se manifestar como consultas muito lentas. Outros vão desistir depois da consulta exceder o tempo limite de espera de um bloqueio, o que não é muito bom. A maneira que o InnoDB atualmente lida com deadlocks é desfazer a transação que tem o menor número de bloqueios de linha exclusivos (uma métrica aproximada para a qual será mais fácil desfazer).

Ordem e comportamento do bloqueio são específicos à ferramenta, então algumas ferramentas podem deadlock em certa sequência de expressões mesmo que outras não. Deadlocks possuem uma natureza dupla: alguns são inevitáveis por causa dos conflitos de dados verdadeiros, e alguns são causados por como uma ferramenta de armazenamento trabalha.

Deadlocks não podem ser quebrados sem desfazer uma das transações, seja parcialmente ou totalmente. Eles são fatos reais em sistemas transacionais, e suas aplicações devem ser desenvolvidas para lidar com eles. Muitas aplicações podem, simplesmente, testar novamente suas transações desde o começo.

Log de Transações

Log de transações ajudam a tornar as transações mais eficientes. Em vez de atualizar as tabelas no disco toda vez que uma alteração ocorrer, a ferramenta de armazenamento pode modificar sua cópia em memória dos dados. Isso é muito rápido. A ferramenta de armazenamento pode então escrever um registro da alteração no log de transação, que está no disco e dessa forma é durável. Esta também é uma relação relativamente rápida, porque anexar eventos de log envolve sequencial E/S em uma pequena área do disco em vez de E/S aleatória em muitos lugares. Então, em algum momento no futuro, um processo pode atualizar a tabela no disco. Dessa maneira, a maioria das ferramentas de armazenamento que usa esta técnica (conhecida como write-ahead logging) acaba escrevendo as mudanças no disco duas vezes*.

Se houver um travamento depois da atualização ser escrita no log de transação, mas antes das alterações serem feitas nos próprios dados, a ferramenta de armazenamento ainda pode recuperar as alterações ao reiniciar. O método de recuperação vária entre as ferramentas de armazenamento.

Transações no MySQL

MySQL AB oferece três ferramentas de armazenamento transacionais: InnoDB, NDB Cluster e Falcon. Diversas ferramentas de terceiros também estão disponíveis; as ferramentas mais conhecidas atualmente são solidDB e PBXT. Nós discutimos algumas propriedades específicas de cada ferramenta na próxima seção.

AUTOCOMMIT

MySQL opera no modo AUTOCOMMIT por padrão. Isso significa que a menos que você explicitamente tenha começado uma transação, ele automaticamente executa cada consulta em uma transação separada. Você pode ativar ou desativar AUTOCOMMIT para a atual conexão configurando uma variável:

```
mysql> SHOW VARIABLES LIKE 'AUTOCOMMIT';
+---------------+-------+
| Variable_name | Value |
+---------------+-------+
| autocommit    | ON    |
+---------------+-------+
1 row in set (0.00 sec)
mysql> SET AUTOCOMMIT = 1;
```

Os valores 1 e ON são equivalentes, como 0 e OFF. Quando você executa com AUTOCOMMIT=0, você sempre está em uma transação, até que execute um COMMIT ou ROLLBACK. MySQL então inicia uma nova transação imediatamente. Alterar o valor de AUTOCOMMIT não tem efeito em tabelas não transacionais, como MyISAM ou Tabelas em Memória, que essencialmente sempre operam em modo AUTOCOMMIT.

Certos comandos, quando executados durantes uma transação aberta, fazem com que MySQL conclua a transação antes de eles executarem. Eles são tipicamente comandos Data Definition Language (DDL – Linguagem de Definição de Dados) que fazem alterações significativas, como ALTER TABLE, mas LOCK TABLES e algumas outras expressões também possuem este efeito. Cheque a documentação da sua versão para a lista completa de comandos que automaticamente comitam uma transação.

MySQL permite que você configure o nível de isolamento usando o comando SET TRANSACTION ISOLATION LEVEL, que surte efeito quando a próxima transação inicia. Você pode configurar o nível de isolamento para todo o servidor no arquivo de configuração (veja Capítulo 6), ou somente para a sua sessão:

```
mysql> SET SESSION TRANSACTION ISOLATION LEVEL READ COMMITTED;
```

* A ferramenta de armazenamento PBXT inteligentemente evita alguns write-ahead logging.

MySQL reconhece todos os quatros níveis padrões de isolamento ANSI, e InnoDB suporta todos eles. Outras ferramentas de armazenamento possuem diversificado suporte para diferentes níveis de isolamento.

Misturando ferramentas de armazenamento em transações

MySQL não gerencia transações no nível do servidor. Em vez disso, as fundamentais ferramentas de armazenamento implementam as transações. Isso significa que você não pode confiavelmente misturar ferramentas diferentes em uma única transação. MySQL AB está trabalhando para adicionar um serviço de gerenciamento de transação de alto nível ao servidor, o que vai tornar seguro misturar e combinar tabelas transacionais em uma transação. Até então, tenha cuidado.

Se você misturar tabelas transacionais e não transacionais (por exemplo, tabelas InnoDB e MyISAM) em uma transação, a transação vai funcionar apropriadamente se tudo correr bem. Porém, se um rollback for solicitado, as alterações na tabela não transacional não podem ser desfeitas. Isso deixa o banco de dados em um estado inconsistente do qual pode ser difícil de se recuperar e renderiza todo o ponto de transações. É por isso que é muito importante pegar a ferramenta de armazenamento correta para cada tabela.

MySQL geralmente não vai te prevenir ou levantar erros se você fizer operações transacionais em uma tabela não transacional. Às vezes, desfazer uma transação vai gerar o aviso "Algumas tabelas alteradas não transacionais não puderem ser desfeitas", mas, na maior parte do tempo, você não terá indicação de que está trabalhando com tabelas não transacionais.

Bloqueio implícito e explícito

InnoDB usa um protocolo de bloqueio de duas fases. Ele pode obter bloqueios a qualquer momento durante uma transação, mas não os libera até um COMMIT ou ROLLBACK. Ele libera todos os bloqueios ao mesmo tempo. Os mecanismos de bloqueio descritos anteriormente são todos implícitos. InnoDB lida com bloqueios automaticamente, de acordo com seu nível de isolamento.

Porém, InnoDB também suporta bloqueio explícito, que o padrão SQL não menciona absolutamente:

- `SELECT ... LOCK IN SHARE MODE`
- `SELECT … FOR UPDATE`

MySQL também suporta os comandos LOCK TABLES e UNLOCK TABLES, que são implementados no servidor, não nas ferramentas de armazenamento. Estes têm seus usos, mas não são um substituto para transações. Se você precisa de transações, use uma ferramenta de armazenamento transacional.

Geralmente, nós vemos aplicações que foram convertidos de MyISAM para InnoDB, mas que ainda estão usado LOCK TABLES. Isso não é mais necessário por causa do bloqueio em nível de linha e pode causar severos problemas de desempenho.

A interação entre LOCK TABLES e transações é complexa, e há comportamentos inesperados em algumas versões de servidor. Então, nós recomendamos que você nunca use LOCK TABLES a menos que esteja em uma transação e AUTOCOMMIT esteja desativado, não importa qual ferramenta de armazenamento você está usando.

Controle de Concorrência de Versão Múltipla

A maioria das ferramentas de armazenamento transacionais do MySQL, como InnoDB, Falcon e PBXT, não usa um simples mecanismo de bloqueio de linha. Em seu lugar, elas usam um bloqueio em nível de linha em conjunto com uma técnica para aumentar concorrência conhecida como controle de concorrência de versão múltipla (MVCC). MVCC não é exclusivo ao MySQL: Oracle, PostgreSQL e alguns outros sistemas de banco de dados usam também.

Você pode classificar MVCC como em bloqueio em nível de linha; ele evita a necessidade de bloquear em muitos casos e tem risco de problemas muito menor. Dependendo de como ele é implementado, permite leituras não bloqueadas, enquanto bloqueia somente os registros necessários durante operações de escrita.

MVCC funciona mantendo um snapshot dos dados como eles existiam em algum ponto no tempo. Isso significa que transações podem enxergar uma visualização consistente dos dados, não importa quanto tempo elas executem. Também significa que diferentes transações podem ver dados diferentes nas mesmas tabelas ao mesmo tempo! Você nunca tentou isso antes, pode ser confuso, mas vai se tornar mais fácil de entender conforme você se familiariza.

Cada ferramenta de armazenamento implementa MVCC de maneira diferente. Algumas das variações incluem controle de concorrência otimista e pessimista. Nós iremos ilustrar uma maneira que o MVCC trabalha explicando uma versão simplificada do comportamento do InnoDB.

InnoDB implementa MVCC armazenando em cada linha dois valores adicionais, escondidos, que registram quando a linha foi criada e quando ela expirou (ou foi deletada). Em vez de armazenar os horários reais que estes eventos ocorreram, a linha armazena o número da versão do sistema na hora em que o evento ocorreu. Este é um número que incrementa toda vez que uma transação começa. Cada transação mantém seu próprio registro da versão do sistema atual, na data em que ele iniciou. Toda consulta tem de checar os números de versão de cada linha em relação à versão da transação. Vamos ver como isso se aplica a operações particulares quando o nível de isolamento da transação é configurado em REPEATABLE READ:

SELECT

InnoDB deve examinar cada linha para garantir que ela se adapte a dois critérios:

- InnoDB deve encontrar uma versão da linha que é no mínimo tão velha quanto a transação (por exemplo, sua versão deve ser menor ou igual à versão da transação). Isso garante que a linha existia antes da transação começar, ou que a transação criou ou alterou a linha.
- A versão de apagamento da linha deve ser indefinida ou maior do que a versão da transação. Isso garante que a linha não foi deletada antes da transação iniciar.

Linhas que passam nos dois testes podem ser retornadas como o resultado da consulta.

INSERT

InnoDB registra o número da versão do sistema atual com a nova linha.

DELETE

InnoDB registra o número da versão do sistema atual como o ID de apagamento da linha.

UPDATE

InnoDB escreve uma nova cópia da linha, usando o número da versão do sistema para a versão da nova linha. Ele também escreve o número da versão do sistema com a versão de apagamento da velha linha.

O resultado de manter todos estes registros extras é que a maioria das consultas de leitura nunca adquire bloqueios. Elas simplesmente leem os dados o mais rápido que podem, garantindo a seleção de somente as linhas que atendem os critérios. As desvantagens são que a ferramenta de armazenamento tem de armazenar mais dados com cada linha, trabalhar mais ao examinar linhas e lidar com algumas adicionais operações de manutenção.

MVCC funciona somente com os níveis de isolamento REPEATABLE READ e READ COMMITTED. READ UNCOMMITTED não é compatível a MVCC porque consultas não leem a versão da linha que é apropriada para a sua versão de transação; elas lêem a versão mais nova, não importa qual seja. SERIALIZABLE não é compatível a MVCC porque leituras bloqueiam todas as linhas que elas retornam.

A Tabela 1-2 resume os vários modelos de bloqueio e níveis de consistência no MySQL.

Tabela 1-2. Modelos de bloqueio de concorrência no MySQL usando o nível de isolamento padrão.

Estratégia de bloqueio	Concorrência	Risco de problemas	Ferramentas
Em nível de tabela	Mais baixa	Mais baixo	MyISAM, Merge, Memory
Em nível de linha	Alta	Alto	NDB Cluster
Em nível de linha com MVCC	Mais alto	Mais alto	InnoDB, Falcon, PBXT, solidDB

Ferramentas de Armazenamento do MySQL

Esta seção dá a você uma visão geral das ferramentas de armazenamento do MySQL. Não entraremos em muitos detalhes aqui, porque discutimos ferramentas de armazenamento e seus comportamentos particulares ao longo d livro. No entanto, até mesmo este livro não é uma fonte completa de documentação; você deve ler os manuais do MySQL para as ferramentas de armazenamento que você decidir usar. MySQL também tem fóruns dedicados para cada ferramenta de armazenamento, geralmente com links para informação adicional e maneiras interessantes de usá-la.

Se você apenas quer comparar as ferramentas em um nível alto, pode pular para a Tabela 1-3.

MySQL armazena cada banco de dados (também chamado de esquema) como um subdiretório do seu diretório de dados no sistema de arquivo fundamental. Quando você cria uma tabela, MySQL armazena a definição da tabela em um arquivo .frm com o mesmo nome da tabela. Assim, quando você cria uma tabela chamada MyTable, MySQL armazena a definição da tabela em MyTable.frm. Por o MySQL usar o sistema de arquivo para armazenar nomes de banco de dados e definições de tabela, distinção entre letras maiúsculas e minúsculas depende da plataforma. Em uma instância MySQL Windows, nomes de tabela e de banco de dados não possuem distinção entre maiúsculas e minúsculas; em sistemas como Unix, eles possuem a distinção. Cada ferramenta de armazenamento armazena os dados da tabela e índices de maneira diferente, mas o próprio servidor lida com a definição da tabela.

Para determinar qual ferramenta de armazenamento uma tabela particular usa, use o comando SHOW TABLE STATUS. Por exemplo, para examinar a tabela user no banco de dados mysql, execute o seguinte:

```
mysql> SHOW TABLE STATUS LIKE 'user' \G
*************************** 1. row ***************************
           Name: user
         Engine: MyISAM
        Row_format: Dynamic
           Rows: 6
    Avg_row_length: 59
       Data_length: 356
   Max_data_length: 4294967295
      Index_length: 2048
         Data_free: 0
    Auto_increment: NULL
       Create_time: 2002-01-24 18:07:17
       Update_time: 2002-01-24 21:56:29
        Check_time: NULL
         Collation: utf8_bin
          Checksum: NULL
    Create_options:
          Comment: Users and global privileges
   1 row in set (0.00 sec)
```

O resultado mostra que esta é uma tabela MyISAM. Você também pode perceber muitas outras informações e estatísticas no resultado. Vamos rapidamente dar uma olhada no que cada linha significa:

Name

O nome da tabela.

Engine

A ferramenta de armazenamento da tabela. Em versões antigas do MySQL, esta coluna era chamada Type, não Engine.

Row_format

O formato intocado. Para uma tabela MyISAM, isso pode ser Dinâmico, Fixo ou Compactado. Linhas dinâmicas variam em comprimento porque contêm campos de comprimento variável como VARCHAR ou BLOB. Linhas fixas, que são sempre do mesmo tamanho, são constituídas de campos que não variam em comprimento, como CHAR e INTEGER. Linhas compactadas existem somente em tabelas compactadas; veja "Tabelas MyISAM compactadas", na página 15.

Rows

O número de linhas na tabela. Para tabelas não transacionais, este número é sempre exato. Para tabelas transacionais é, geralmente, uma estimativa.

Avg_row_length

Quantos bytes a linha comum contém.

Data_length

Quantos dados (em bytes) a tabela inteira contém.

Max_data_length

A quantidade máxima de dados que esta tabela pode conter. Veja "Armazenamento", na página 14, para mais detalhes.

Index_length

Quanto espaço de disco os dados de índice consomem.

Data_free

Para uma tabela MyISAM, a quantidade de espaço que é alocado, mas atualmente inutilizado. Este espaço contém as linhas previamente deletadas e podem ser exigidas por futuras expressões INSERT.

Auto_increment

O próximo valor AUTO_INCREMENT.

Create_time

Quando a tabela foi criada pela primeira vez.

Update_time

Quando os dados na tabela foram modificados pela última vez.

Check_time

Quando a tabela foi checada pela última vez usando CHECK TABLE ou myisamchk.

Collation

O conjunto de caractere padrão e classificação para colunas de caractere nesta tabela. Veja "Conjunto de Caractere e Classificação", na página 197, para mais sobre estas características.

Checksum

Uma soma de verificação dos conteúdos de toda a tabela se ativado.

Create_options

Qualquer outra opção que foi especificada quando a tabela foi criada.

Comment

Este campo contém uma variedade de informações extras. Para uma tabela MyISAM, contém os comentários, se houver, que foram configurados quando a tabela foi criada. Se a tabela usar a ferramenta de armazenamento InnoDB, a quantidade de espaço disponível na tablespace InnoDB aparece aqui. Se a tabela for uma visualização, o comentário contém o texto "VIEW".

A Ferramenta MyISAM

Como ferramenta de armazenamento padrão do MySQL, MyISAM oferece um bom acordo entre desempenho e características úteis, como indexação de texto completo, compactação e funções espaciais (GIS). MyISAM não suporta transações ou bloqueios em nível de linha.

Armazenamento

MyISAM tipicamente armazena cada tabela em dois arquivos: um arquivo de dados e um arquivo de índice. Os dois arquivos sustentam extensões *.MYD* e *.MYI*, respectivamente. O formato MyISAM é de plataforma neutra, significando que você pode copiar os arquivos de dados e índice a partir de um servidor baseado em Intel para um Powerpc ou Sun SPARC sem nenhum problema.

Tabelas MyISAM podem conter linhas dinâmicas ou estáticas (comprimento fixo). MySQL decide qual formato usar baseado na definição da tabela. O número de linhas que uma tabela MyISAM pode conter é limitado primariamente pelo espaço em disco disponível no seu servidor de banco de dados e o maior arquivo que seu sistema operacional vai permitir que você crie.

Tabelas MyISAM criadas no MySQL 5.0 com linhas de comprimento variável são configuradas por padrão para lidar com 256 TB de dados, usando ponteiros de 6 bytes aos registros de dados. Versões anteriores do MySQL padronizavam ponteiros de 4 bytes, para até 4 GB de dados. Todas as versões do MySQL podem lidar com um tamanho de ponteiro de até 8 bytes. Para modificar o tamanho do ponteiro em uma tabela MyISAM (seja para mais ou para menos), você deve especificar os valores para as opções MAX_ROWS e AVG_ROW_LENGTH, que representam figuras aproximadas para a quantidade de espaço que você precisa:

```
CREATE TABLE mytable (
a INTEGER NOT NULL PRIMARY KEY,
b CHAR(18) NOT NULL
) MAX_ROWS = 1000000000 AVG_ROW_LENGTH = 32;
```

Neste exemplo, nós dissemos ao MySQL para ficar preparado para armazenar pelo menos 32 GB de dados na tabela. Para descobrir o que o MySQL decidiu fazer, simplesmente peça o status da tabela:

```
mysql> SHOW TABLE STATUS LIKE 'mytable' \G
*************************** 1. row ***************************
           Name: mytable
         Engine: MyISAM
     Row_format: Fixed
           Rows: 0
 Avg_row_length: 0
    Data_length: 0
Max_data_length: 98784247807
   Index_length: 1024
```

```
     Data_free: 0
Auto_increment: NULL
   Create_time: 2002-02-24 17:36:57
   Update_time: 2002-02-24 17:36:57
    Check_time: NULL
Create_options: max_rows=1000000000 avg_row_length=32
       Comment:
1 row in set (0.05 sec)
```

Como você pode ver, MySQL lembra das opções de criação exatamente como especificado e escolhe uma representação capaz de conter 91 GB de dados! Você pode alterar o tamanho do ponteiro mais tarde com a expressão ALTER TABLE, mas isso vai fazer com que toda a tabela e todos os seus índices sejam reescritos, o que pode levar muito tempo.

Características do MyISAM

Como uma das mais velhas ferramentas de armazenamento incluída no MySQL, MyISAM possui muitas características que foram desenvolvidas ao longo de muitos anos de uso para preencher necessidades do mercado:

Bloqueio e concorrência

MyISAM bloqueia tabelas inteiras, não linhas. Leitores obtêm bloqueios (de leitura) compartilhados em todas as tabelas que eles precisam ler. Escritores obtêm bloqueios (de leitura) exclusivos. Porém, você pode inserir novas linhas na tabela enquanto consultas selecionadas estão rodando nela (inserções concorrentes). Esta é uma característica muito importante e útil.

Reparação automática

MySQL suporta checagem e reparação automática de tabelas MyISAM. Veja "Ajuste de E/S do MyISAM", na página 234, para mais informações.

Reparação manual

Você pode usar os comandos CHECK TABLE mytable e REPAIR TABLE mytable para checar os erros de uma tabela e consertá-los. Você também pode usar a ferramenta de linha de comando myisamchk para checar e reparar tabelas quando o servidor estiver off-line.

Características de índice

Você pode criar índices nos primeiros 500 caracteres das colunas BLOB e TEXT nas tabelas MyISAM. MyISAM suporta índices de textos completos, o que indexa palavras individuais para operações complexas de busca. Para mais informações sobre indexação, veja o Capítulo 3.

Escritas de chave atrasadas

Tabelas MyISAM marcadas com a opção de criação DELAY_KEY_WRITE não escrevem dados de índice alterados no disco no final de uma consulta. Em vez disso, MyISAM faz buffer das alterações no buffer chave na memória. Ele envia blocos de índice ao disco quando ele corta o buffer ou fecha a tabela. Isso pode melhorar o desempenho em tabelas pesadamente usadas que mudam frequentemente. Porém, depois de um servidor ou sistema travar, os índices estarão definitivamente corrompidos e precisarão de reparo. Você deve cuidar disso com um script que execute myisamchk antes de reiniciar o servidor, ou usando as opções de recuperação automática mesmo se você não usar DELAY_KEY_WRITE, estes salva-vidas ainda podem ser uma excelente ideia). Você pode configurar escritas de chave atrasadas globalmente, assim como para tabelas individuais.

Tabelas MyISAM compactadas

Algumas tabelas – por exemplo, em aplicações baseadas em CD-ROM ou DVD-ROM e alguns ambientes embutidos – nunca mudam depois de serem criadas e preenchidas com os dados. Estas podem ser bem adaptadas a tabelas MyISAM compactadas.

Você pode compactar (ou "empacotar") tabelas com a utilidade myisampack. Você não pode modificar tabelas compactadas (embora você possa descompactar, modificar e recompactar tabelas se precisar), mas elas geralmente usam menos espaço em disco. Como resultado, elas oferecem desempenho mais rápido, porque seu tamanho menor exige buscas ao disco para encontrar registros. Tabelas MyISAM compactadas podem ter índices, mas eles são somente leitura.

O problema de descompactar dados para lê-lo é insignificante para a maioria das aplicações em hardwares modernos, em que o ganho real está em reduzir a E/S. As linhas são compactadas individualmente, então MySQL não precisa descompactar uma tabela inteira (ou até mesmo uma página) apenas para trazer uma única linha.

A Ferramenta Merge MyISAM

A ferramenta Merge é uma variação do MyISAM. Uma tabela Merge é a combinação de diversas tabelas MyISAM idênticas em uma tabela virtual. Isso é particularmente útil quando você usa MySQL em aplicações de log e depósito de dados. Veja "Tabelas Merge e Particionamento", na página 211, para uma discussão detalhada sobre tabelas Merge.

A Ferramenta InnoDB

InnoDB foi desenvolvido para processamento de transação – especialmente, processamento de muita transações de vida curta que geralmente completam ao invés de serem desfeitas. Ele continua a ferramenta de armazenamento mais popular para armazenamento transacional. Seu desempenho e recuperação de travamento automática também o tornam popular para necessidades de armazenamento não-transacional.

InnoDB armazena seus dados em uma série de um ou mais arquivos de dados que são coletivamente conhecidos como uma tablespace. Uma tablespace é essencialmente uma caixa preta que InnoDB gerencia sozinho. No MySQL 4.1 e em versões mais novas, InnoDB pode armazenar os índices e dados de cada tabela em arquivos separados. InnoDB também pode usar partições de discos intocados para construir sua tablespace. Veja "O tablespace InnoDB", na página 241, para mais informações.

InnoDB usa MVCC para atingir alta concorrência e implementa todos os quatros níveis de isolamento padrão do SQL. Ele padroniza no nível de isolamento REPEATABLE READ e possui uma estratégia next-key locking (bloqueio da próxima linha) que previne leituras fantasmas neste nível de isolamento: em vez de bloquear somente as linhas que você tocou em uma consulta, InnoDB bloqueia lacunas na estrutura de índice também, evitando que fantasmas sejam reiniciados.

Tabelas InnoDB são construídas em um índice agrupado, que iremos abordar em detalhes no Capítulo 3. As estruturas de índice do InnoDB são muito diferentes das estruturas da maioria das outras ferramentas de armazenamento do MySQL. Como um resultado, ele oferece buscas de chave primária muito rápidas. Porém, índices secundários (índices que não são a chave primária) contêm as colunas da chave primária, então se sua chave primária for grande, outros índices também serão grandes. Você deve tentar uma pequena chave primária se você tiver muitos índices em uma tabela. InnoDB não compacta seus índices.

Na época desta escrita, InnoDB não pode construir índices por classificação, o que MyISAM pode fazer. Assim, InnoDB carrega dados e cria índices mais lentamente do que MyISAM. Qualquer operação que modifica a estrutura da tabela InnoDB vai reconstruir a tabela inteira, incluindo todos os índices.

InnoDB foi desenvolvido quando a maioria dos servidores tinha discos lentos, uma única CPU, e memória limitada. Hoje, como servidores de múltiplas centrais com grandes quantidades de memória e discos rápidos estão se tornando menos caros, InnoDB está experimentando alguns problemas de escalonamento.

Desenvolvedores InnoDB estão resolvendo esses assuntos, mas na época desta escrita, diversos deles continuavam problemáticos. Veja "Ajuste de Concorrência do InnoDB", na página 246, para mais informações sobre como alcançar alta concorrência com InnoDB.

Além das suas capacidades de alta concorrência, a próxima característica mais popular do InnoDB é a restrição de chave estrangeira, que o próprio servidor MySQL ainda não oferece. InnoDB também oferece buscas extremamente rápidas para consultas que usam uma chave primária.

InnoDB tem uma variedade de otimizações internas. Estas incluem previsão de leitura para trazer dados antecipadamente a partir do disco, um índice de hash adaptável que automaticamente constrói índices de hash na memória para buscas muito rápidas, e um buffer de inserção para acelerar inserções. Nós abordamos estas extensivamente mais tarde neste livro.

O comportamento do InnoDB é muito confuso, e nós altamente recomendamos ler a seção "Modelo de Transação InnoDB e bloqueio" do manual MySQL se você estiver usando InnoDB. Há muitas surpresas e exceções que você deve estar ciente antes de construir uma aplicação com InnoDB.

A Ferramenta Memory

Tabelas Memory (anteriormente chamadas de tabelas HEAP) são úteis quando você precisa de rápido acesso aos dados que nunca mudam ou que não precisam persistir depois de uma reinicialização. Tabelas Memory são, geralmente, em ordem de magnitude mais rápidas do que tabelas MyISAM. Todos os seus dados são armazenados na memória, então consultas não têm de esperar por E/S de disco. A estrutura da tabela de uma tabela Memory persiste após a reinicialização de um servidor, mas nenhum dado sobrevive.

Aqui estão alguns bons usos para tabelas Memory:

- Para "buscar" ou "mapear" tabelas, como uma tabela que mapeia códigos postais para nomes de estados
- Para fazer cache de resultados dos dados periodicamente agregados
- Para resultados intermediários ao analisar dados

Tabelas Memory suportam índices HASH, que são muito rápidos para buscar consultas. Veja "Índices hash", na página 84, para mais informações sobre índices HASH.

Apesar das tabelas Memory serem muito rápidas, elas geralmente não funcionam bem como um substituto de objetivo geral para tabelas baseadas em disco. Elas usam bloqueio em nível de tabela, o que dá uma concorrência de escrita muito baixa, e elas não suportam os tipos de coluna TEXT ou BLOB. Elas também suportam somente linhas de tamanho fixo, então elas realmente armazenam VARCHARs como CHARs, o que pode desperdiçar memória.

MySQL usa a ferramenta Memory internamente enquanto processa consultas que exigem uma tabela temporária para manter resultados imediatos. Se o resultado intermediário tornar-se muito grande para uma tabela Memory, ou tiver as colunas TEXT ou BLOB, MySQL vai convertê-lo em uma tabela MyISAM no disco. Falaremos mais sobre isso em capítulos futuros.

As pessoas geralmente confundem tabelas Memory com tabelas temporárias, que são tabelas efêmeras criadas com CREATE TEMPORARY TABLE. Tabelas temporárias podem usar qualquer ferramenta de armazenamento; elas não são a mesma coisa que tabelas que usam a ferramenta de armazenamento Memory. Tabelas temporárias são visíveis somente à única conexão e desaparecem completamente quando a conexão fecha.

A Ferramenta Archive

A ferramenta Archive suporta somente consultas INSERT e SELECT, e não suporta índices. Ela causa muito menos E/S de disco do que MyISAM, porque faz buffer de escritas de dados e compacta cada linha com zlib conforme ela é inserida. Também, cada consulta SELECT exige um scan de tabela completo. Tabelas Archive são, dessa maneira, ideais para aquisição de log e de dados, em que a análise tende a escanear uma tabela inteira, ou onde você quer consultas INSERT rápidas em um mestre de replicação. Escravos de replicação usam uma

ferramenta de armazenamento diferente para a mesma tabela, o que significa que a tabela no escravo possui índices para desempenho mais rápido na análise veja o Capítulo 8 para mais sobre replicação).

Archive suporta bloqueio em nível de linha e um sistema especial de buffer para inserções de alta concorrência. Ele dá leituras consistentes parando um SELECT depois de ter recuperado o número de linhas que existiam na tabela quando a consulta começou. Ele também faz várias inserções invisíveis até que elas estejam completas. Estas características emulam alguns aspectos de comportamentos transacionais e de MVCC, mas Archive não é uma ferramenta de armazenamento transacional. É, simplesmente, uma ferramenta de armazenamento que é otimizada para inserção em alta velocidade e armazenamento compactado.

A Ferramenta CSV

A ferramenta CSV pode tratar arquivos de valores separados por vírgulas (CSV) como tabelas, mas não suporta índices neles. Esta ferramenta permite que você copie arquivos dentro e fora do banco de dados enquanto o servidor está rodando. Se você exportar um arquivo CSV de uma planilha eletrônica e salvá-lo no diretório de dados do servidor MySQL, o servidor pode lê-lo imediatamente. Similarmente, se você escrever dados em uma tabela CSV, um programa externo pode lê-los imediatamente. Tabelas CSV são especialmente úteis como um formato de troca de dados e para certos tipos de log.

A Ferramenta Federated

A ferramenta Federated não armazena dados localmente. Cada tabela Federated refere-se a uma tabela em um servidor MySQL remoto, então realmente ela se conecta a um servidor remoto para todas as operações. Às vezes é usada para ativar "hacks", assim como truques com replicação.

Há muitas esquisitices e limitações na atual implementação desta ferramenta. Por causa da maneira como a ferramenta Federated funciona, nós achamos que ela é mais útil para buscas de linha única por chave primária, ou para consultas INSERT que você quer para afetar um servidor remoto. Ela não executa bem em consultas agregadas, joins ou outras operações básicas.

A Ferramenta Blackhole

A ferramenta não possui nenhum mecanismo de armazenamento. Ela descarta todo INSERT ao invés de armazená-lo. Porém, o servidor escreve consultas nas tabelas Blackhole para seu log habitual, para que elas possam ser replicadas aos escravos ou, simplesmente, mantidas no log. Isso torna a ferramenta Blackhole útil para configurações de replicação sofisticadas e auditar log.

A Ferramenta NDB Cluster

MySQL AB adquiriu a ferramenta NDB Cluster da Sony Ericsson em 2003. Foi originalmente desenvolvida para alta velocidade (requisitos de desempenho em tempo real), com redundância e capacidade de equilíbrio de carga. Embora tenha sido logada no disco, ela manteve todos os seus dados na memória e foi otimizada para buscas de chave primária. Desde então MySQL adicionou outros métodos de indexação e muitas otimizações, e MySQL 5.1 permite que algumas colunas sejam armazenadas no disco.

A arquitetura NDB é única: um NDB Cluster é completamente diferente, por exemplo, de um cluster Oracle. A infraestrutura do NDB é baseada em um conceito de nenhum compartilhamento. Não há uma rede de área de armazenamento ou outra grande solução de armazenamento centralizada, nos quais outros tipos de clusters baseiam-se. Um banco de dados NDB consiste em nós de dados, nós de gerenciamento e nós SQL (instâncias MySQL). Cada nó de dado contém um segmento ("fragmento") dos dados do cluster. Os fragmentos são duplicados, então o sistema tem múltiplas cópias dos mesmos dados em diferentes nós. Um servidor físico é, geralmente, dedicado a cada nó por redundância e alta disponibilidade. Neste sentido, NDB é similar a RAID no nível do servidor.

Os nós de gerenciamento são usados para recuperar a configuração centralizada e para monitorar e controlar os nós de cluster. Todos os nós de dados comunicam-se uns com os outros, e todos os servidores MySQL conectam-se a todos os nós de dados. Baixa latência de rede é crucialmente importante para NDB Cluster.

Um aviso: NDB Cluster é uma tecnologia "muito legal" e definitivamente vale a pena alguma exploração para satisfazer sua curiosidade, mas muitas pessoas técnicas tendem a procurar desculpas para usá-la e tentam aplicá-la para necessidades as quais não é favorável. Na nossa experiência, mesmo depois de estudá-la cuidadosamente, muitas pessoas ainda não aprendem para que serve esta ferramenta e como ela funciona até instalarem e usarem por um tempo. Isso comumente resulta em muito desperdício de tempo, porque simplesmente não é desenvolvido como uma ferramenta de armazenamento de objetivo geral.

Um choque comum é que NDB atualmente realiza joins no nível do servidor MySQL, não na camada da ferramenta de armazenamento. Porque todos os dados para NDB devem ser recuperados através da rede, joins complexas são extremamente lentas. Por outro lado, buscas de tabela única podem ser muito rápidas, porque múltiplos nós de dados oferecem parte do resultado. Este é apenas um dos muitos aspectos que você terá de considerar e entender totalmente ao procurar no NDB Cluster para uma aplicação em particular.

NDB Cluster é tão grande e complexo que não discutiremos com detalhes neste livro. Você deve procurar um livro dedicado a este tópico se você tiver interesse. Porém, diremos que, geralmente, ele não é o que você pensa que é, e para a maioria das aplicações tradicionais, ele não é a resposta.

A Ferramenta Falcon

Jim Starkey, um pioneiro em banco de dados cujas mais recentes invenções incluem Interbase, MVCC e o tipo de coluna BLOB, desenvolveu a ferramenta Falcon. MySQL AB adquiriu a tecnologia Falcon em 2006, e Jim atualmente trabalha para MySQL AB.

Falcon é desenvolvido para hardwares atuais – especificamente, para servidores com múltiplos processadores de 64 bits e muita memória – mas também pode operar em ambientes mais modestos. Falcon usa MVCC e tenta manter as transações executando inteiramente na memória. Isso faz rollbacks e operações de recuperação extremamente rápidas.

Falcon está finalizado na época desta escrita (por exemplo, ele ainda não sincroniza seus commits com o log binário), então não podemos escrever sobre ele com muita autoridade. Até avaliações iniciais de desempenho que fizemos com ele provavelmente estarão desatualizadas quando estiver pronto para uso geral. Ele parece ter um bom potencial para muitas aplicações on-line, mas saberemos mais sobre ele conforme o tempo for passando.

A Ferramenta solidDB

A ferramenta solidDB, desenvolvida por Solid Information Technology (http://www.soliddb.com), é uma ferramenta transacional que usa MVCC. Ela suporta controle de concorrência pessimista e otimista, o que nenhuma outra ferramenta atualmente faz. solidDB para MySQL inclui suporte completo de chave estrangeira. É similar ao InnoDB em muitas maneiras, como seu uso de índices agrupados. solidDB para MySQL inclui uma capacidade de backup on-line sem nenhum custo.

O solidDB para produto MySQL é um pacote completo que consiste na ferramenta de armazenamento solidDB, a ferramenta de armazenamento MyISAM e o servidor MySQL. A "cola" entre a ferramenta de armazenamento solidDB e o servidor MySQL foi introduzida em 2006. Porém, a tecnologia fundamental e código amadureceram durantes os 15 anos de história da empresa. solid certifica e suporta o produto todo. É licenciado sob GPL e oferecido comercialmente sob um modelo de licenciamento duplo que é idêntico ao do servidor MySQL.

A Ferramenta PBXT (Primebase XT)

A ferramenta PBXT, desenvolvida por Paul McCullagh da SNAP Innovation GmbH em Hamburgo, Alemanha (http://www.primebase.com), é uma ferramenta de armazenamento transacional com um design único. Uma das suas características diferenciadas é como ela usa seus logs transacionais e arquivos de dados para evitar log write-ahead, o que reduz muitos problemas da execução da transação. Esta arquitetura dá ao PBXT o potencial para lidar com concorrência de escrita muito alta, e testes já mostraram que ele pode ser mais rápido do InnoDB para certas operações. PBXT usa MVCC e suporta restrições de chave estrangeira, mas não usa índices agrupados.

PBXT é uma ferramenta muito nova, e vai precisar se provar superior em ambientes de produção. Por exemplo, sua implementação de transações verdadeiramente duráveis só foi completada recentemente, enquanto escrevíamos este livro.

Como uma extensão a PBXT, SNAP Innovation está trabalhando em uma estrutura "blob streaming" escalonável (*http://www.blobstreaming.org*). É desenvolvida para armazenar e recuperar grandes quantidades de dados binários de maneira eficiente.

A Ferramenta de Armazenamento Maria

Maria é uma nova ferramenta de armazenamento sendo desenvolvida por alguns engenheiros superiores do MySQL incluindo Michael Widenius, que criou MySQL. O release inicial 1.0 inclui somente algumas das suas características planejadas.

O objetivo é usar Maria como um substituto para MyISAM, que é atualmente a ferramenta de armazenamento padrão do MySQL, e a qual o servidor usa internamente para tarefas como tabelas de privilégios e tabelas temporárias criadas ao executar consultas. Aqui estão alguns destaques:

- A opção de armazenamento transacional ou não-transacional, em uma base por tabela
- Recuperação de travamento, mesmo quando a tabela estiver executando em modo não-transacional
- Bloqueio em nível de linha e MVCC
- Melhor gerenciamento de BLOB

Outras Ferramentas de Armazenamento

Vários terceiros oferecem outras (às vezes do proprietário) ferramentas, e há uma variedade de ferramentas experimentais e de objetivo geral por aí (por exemplo, uma ferramenta para consultar serviços web). Algumas destas ferramentas são desenvolvidas informalmente, talvez por apenas um ou dois engenheiros. Isso é porque é relativamente fácil criar uma ferramenta de armazenamento para MySQL. Porém, a maioria destas ferramentas não é amplamente publicada, em parte por causa da sua aplicabilidade limitada. Deixaremos que você explore estas ofertas por sua conta.

Selecionando a Ferramenta Correta

Ao desenvolver aplicações baseadas no MySQL, você deve decidir qual ferramenta de armazenamento usar para armazenar seus dados. Se você não pensar nisso durante a fase de desenvolvimento, você provavelmente vai se deparar com complicações mais tarde no processo. Você pode achar que a ferramenta padrão não oferece uma característica que você precisa, como transações, ou, talvez, a mistura de consultas de escrita e leitura que a sua aplicação gera vão exigir bloqueio mais granular do que os bloqueios de tabela do MyISAM.

Por você poder escolher ferramentas de armazenamento em uma base de tabela-por-tabela, você precisará de uma ideia clara de como cada tabela será usada e os dados que ela vai armazenar. Também ajuda ter um bom entendimento da aplicação por completo e seu potencial para crescimento. Munidos com estas informações, você pode começar a fazer boas escolhas sobre quais ferramentas de armazenamento podem fazer o trabalho.

Não é necessariamente uma boa ideia usar diferentes ferramentas de armazenamento para tabelas diferentes. Se você puder se virar com ele, geralmente vai tornar sua vida muito mais fácil se você escolher uma ferramenta de armazenamento para todas as suas tabelas.

Considerações

Embora muitos fatores possam afetar sua decisão sobre qual ferramenta(s) de armazenamento usar, normalmente chegamos a algumas considerações primárias. Aqui estão os elementos principais que você deve levar em consideração:

Transações

Se sua aplicação exigir transações, InnoDB é a escolha mais comprovada, estável e bem integrada na época desta escrita. Porém, esperamos ver as ferramentas transacionais como fortes candidatos com o passar do tempo.

MyISAM é uma boa escolha se uma tarefa não exigir transações e realizar primariamente consultas SELECT ou INSERT. Às vezes, componentes específicos de uma aplicação (como log) caem nesta categoria.

Concorrência

Como melhor satisfazer seus requisitos de concorrência depende da sua carga de trabalho. Se você somente precisar inserir e ler concorrentemente, acredite ou não, MyISAM é uma boa escolha! Se você precisar permitir que uma mistura de operações rode concorrentemente sem interferir umas com as outras, uma das ferramentas com bloqueio em nível de linha deve funcionar bem.

Backups

A necessidade de realizar backups regulares também pode influenciar suas escolhas de tabela. Se seu servidor puder ser encerrado em intervalos regulares para backups, as ferramentas de armazenamento são igualmente fáceis de lidar. Porém, se você precisar executar backups on-line de uma forma ou de outra, as escolhas se tornam menos claras. O Capítulo 11 trata deste tópico com mais detalhes.

Também tenha em mente que usar múltiplas ferramentas de armazenamento aumenta a complexidade de backups e ajuste de servidor.

Recuperação de travamento

Se você tiver muitos dados, deve seriamente considerar quanto tempo ele vai levar para recuperar-se de um travamento. Tabelas MyISAM geralmente se corrompem mais facilmente e demoram mais para recuperar-se do que tabelas InnoDB, por exemplo. De fato, esta é uma das razões mais importantes porque as pessoas usam InnoDB quando não precisam de transações.

Características especiais

Finalmente, às vezes, você pode perceber que uma aplicação baseia-se em características particulares ou otimizações que somente algumas das ferramentas de armazenamento do MySQL oferecem. Por exemplo, muitas aplicações se baseiam em otimizações de índice agrupado. Neste momento, isso limita você a InnoDB e solidDB. Por outro lado, somente MyISAM suporta busca de texto completo dentro do MySQL. Se uma ferramenta de armazenamento adequar-se a um ou mais dos requisitos importantes, mas não a outros, você precisa chegar a um termo comum ou encontrar uma inteligente solução de design. Você pode freqüentemente obter o que precisa de uma ferramenta de armazenamento que aparentemente não suporta seus requisitos.

Você não precisa decidir imediatamente. Há muito material sobre os pontos fortes e fracos de cada ferramenta de armazenamento no resto do livro e, também, muitas dicas de arquitetura e design. Em geral, há, provavelmente, mais opções do que você imagina, e pode ajudar a voltar a esta questão depois de ler mais.

Exemplos Práticos

Estes assuntos podem parecer muito abstratos sem algum tipo de contexto do mundo real, então vamos considerar algumas comuns aplicações de banco de dados. Vamos examinar uma variedade de tabelas e determinar qual ferramenta melhor se adapta às necessidades de cada tabela. Damos um resumo das opções na próxima seção.

Log

Suponha que você queira usar MySQL para logar um registro de cada ligação telefônica de um interruptor de telefone central em tempo real. Ou talvez você instalou `mod_log_sql` para Apache, então você pode logar todas as visitas ao seu site diretamente em uma tabela. Em tal aplicação, velocidade é, provavelmente, o objetivo mais importante; você não quer o banco de dados seja um obstáculo. As ferramentas de armazenamento MyISAM e Archive funcionam muito bem porque possuem baixíssimo risco de problemas e podem inserir milhares de registros por segundo. A ferramenta de armazenamento PBXT é provável que também seja particularmente adequada para objetivos de log.

Porém, as coisas vão ficar interessantes se você decidir que é hora de começar a executar relatórios para resumir os dados que você logou. Dependendo das consultas que você usa, há uma boa chance de que reunir dados para o relatório diminua significativamente o processo de inserção de registros. O que você pode fazer?

Uma solução é usar a característica de replicação embutida do MySQL para clonar os dados em um segundo (escravo) servidor e, então, executar suas consultas consumidoras de tempo e CPU sobre os dados no escravo. Isso deixa o mestre livre para inserir registros e permite que você execute qualquer consulta que quiser no escravo sem se preocupar como ela pode afetar o log de tempo real.

Você também pode executar consultas em momentos de baixa carga, mas confie que esta estratégia vai continuar a funcionar conforme sua aplicação cresce.

Outra opção é usar uma tabela Merge. Ao invés de sempre logar na mesma tabela, ajusta a aplicação para logar a uma tabela que contenha o ano e nome ou número do mês no seu nome, como `web_logs_2008_01` ou `web_logs_2008_jan`. Então, defina uma tabela Merge que contenha os dados que você quer resumir e use-a nas suas consultas. Se você precisa resumir dados diariamente ou semanalmente, a mesma estratégia funciona; você somente precisa criar tabelas com nomes mais específicos, como `web_logs_2008_01_01`. Enquanto você estiver ocupado executando consultas nas tabelas que não estão mais sendo escritas, sua aplicação pode logar registros à sua atual tabela ininterrupta.

Tabelas somente leitura ou principalmente leitura

Tabelas que contêm dados usados para construir um catálogo ou algum tipo de listagem (trabalhos, leilões, bens imobiliários etc.) são, geralmente, mais lidas do que escritas. Isso faz delas boas candidatas para MyISAM – se você não se importa com o que acontece quando MyISAM trava. Não subestime a importância disso; muitos usuários realmente não entendem como é arriscado usar uma ferramenta de armazenamento que nem ao menos se esforça para ter seus dados escritos no disco.

É uma excelente ideia executar uma realista simulação de carga em um servidor de teste e, então, literalmente retirar o plug de energia. A experiência de recuperar-se de um travamento é impagável. Poupa surpresas chatas no futuro.

Não acredite somente na comum sabedoria popular: "MyISAM é mais rápido do que InnoDB". Não é categoricamente verdade. Nós podemos nomear dezenas de situações em que InnoDB deixa o MyISAM na poeira, especialmente para aplicações em que índices agrupados são úteis ou onde os dados cabem na memória. Conforme lê o restante deste livro, você terá uma compreensão de quais fatores influenciam o desempenho de uma ferramenta de armazenamento (tamanho dos dados, número de operações de E/S exigidas, chave primária versus índices secundários etc.) e quais delas têm importância para sua aplicação.

Processamento de pedido

Quando você lida com qualquer tipo de processamento de pedido, transações são tudo, menos exigidas. Pedidos pela metade não vão fazer os clientes gostarem do seu serviço. Outra consideração importante é se a ferramenta precisa suportar restrições de chave estrangeira. É provável que InnoDB seja sua melhor aposta para aplicações de processamento de pedido, apesar de que qualquer ferramenta de armazenamento transacional seja boa candidata.

Preços de Ações ou Títulos

Se você estiver coletando preços de ações ou títulos para sua própria análise, MyISAM funciona muito bem, com as interrupções normais. Porém, se você estiver executando um serviço web de alto tráfego que tem uma alimentação de cotações em tempo real e milhares de usuários, uma consulta nunca deve ter de esperar. Muitos clientes podem estar tentando ler e escrever na tabela simultaneamente, então um bloqueio em nível de linha ou um design que minimiza atualizações é o caminho certo.

Quadro de avisos e fóruns de discussão concorrentes

Discussões concorrentes são um problema interessante para usuários MySQL. Há centenas de sistemas baseados em Perl e PHP gratuitamente disponíveis que oferecem discussões concorrentes. Muitos deles não são escritos com eficiência de banco de dados em mente, então eles tendem a executar muitas consultas para cada requisição que eles servem. Alguns foram escritos para serem independentes do banco de dados, então suas consultas não tiram vantagem das características de nenhum sistema de banco de dados. Eles também tendem a atualizar contadores e compilar estatísticas de uso sobre as várias discussões. Muitos dos sistemas também usam algumas tabelas monolíticas para armazenar todos os seus dados. Como um resultado, algumas tabelas centrais tornam-se o foco de pesada atividade de leitura e de escrita, e os bloqueios exigidos para forçar consistência tornam-se uma substancial fonte de contenção.

Apesar dos seus defeitos de design, a maioria dos sistemas funciona bem para cargas pequenas e médias. Porém, se um site crescer bastante e gerar tráfego significativo, ele pode se tornar muito lento. A solução óbvia é trocar para uma diferente ferramenta de armazenamento que possa lidar com pesado volume de leitura/escrita, mas usuários que tentam isso às vezes são surpreendidos ao descobrir que os sistemas executam ainda mais vagarosamente do que antes!

O que estes usuários não percebem é que o sistema está usando uma consulta particular, normalmente algo assim:

```
mysql> SELECT COUNT(*) FROM table;
```

O problema é que nem todas as ferramentas podem executar esta consulta rapidamente: MyISAM pode, mas outras ferramentas não. Há exemplos similares para toda ferramenta. O Capítulo 2 vai te ajudar a evitar que esta situação te pegue de surpresa e vai mostrar a você como encontrar e consertar os problemas.

Aplicações em CD-ROM

Se um dia você precisar distribuir uma aplicação baseada em CD-ROM ou DVD-ROM que usa arquivos de dados MySQL, considere usar MyISAM ou tabelas MyISAM compactadas, que podem facilmente ser isoladas e copiadas em outra mídia. Tabelas MyISAM compactadas usam muito menos espaço do que as descom-

pactadas, mas são somente leitura. Isso pode ser problemático em certas aplicações, mas como os dados vão ser mídias de somente leitura, há pouca razão em não usar tabelas compactadas para esta tarefa particular.

Resumo de Ferramenta de Armazenamento

A Tabela 1-3 resume as características relacionadas à transação e bloqueio das ferramentas de armazenamento mais populares do MySQL. A coluna da versão do MySQL mostra a versão mínima do MySQL que você vai precisar para usar a ferramenta, embora para algumas ferramentas e versões de MySQL você pode ter de compilar seu próprio servidor. A palavra "Todas" nesta coluna indica todas as versões desde MySQL 3.23.

Tabela 1-3. Resumo de ferramenta de armazenamento do MySQL.

Ferramenta de armazenamento	Versão MySQL	Transações	Granularidade de bloqueio	Aplicações chave	Indicações do contador
MyISAM	Todas	Não	Tabela com inserções concorrentes	SELECT, INSERT, carga em massa	Carga de trabalho leitura/escrita mista
MyISAM Merger	Todas	Não	Tabela com inserções concorrentes	Arquivamento segmentado, depósito de dados	Muitas buscas globais
Memory (HEAP)	Todas	Não	Tabela	Cálculos intermediários, dados de busca estatísticos	Grandes datasets, armazenamento persistente
InnoDB	Todas	Sim	Nível de linha com MVCC	Processamento transacional	Nenhuma
Falcon	6.0	Sim	Nível de linha com MVCC	Processamento transacional	Nenhuma
Archive	4.1	Sim	Nível de linha com MVCC	Log, análise agregada	Necessidades de acesso, atualizações e apagamentos aleatórios
CSV	4.1	Não	Tabela	Log, carga em massa de dados externos	Necessidades de acesso aleatório, indexação
Blackhole	4.1	Sim	Nível de linha com MVCC	Arquivamento logado ou replicado	Tudo menos o uso intencionado
Federated	5.0	Não disponível	N/D	Fontes de dados distribuídas	Tudo menos o uso intencionado
NDB Cluster	5.0	Sim	Nível de linha	Alta disponibilidade	Usos mais típicos
PBXT	5.0	Sim	Nível de linha com MVCC	Processamento transacional, log	Necessidade índices agrupados
solidDB	5.0	Sim	Nível de linha com MVCC	Processamento transacional	Nenhuma
Maria (planejado)	6.x	Sim	Nível de linha com MVCC	Substituição de MyISAM	Nenhuma

Conversões de Tabela

Há diversas maneiras de converter uma tabela de uma ferramenta de armazenamento para outra, cada uma com vantagens e desvantagens. Nas seções seguintes, nós abordamos três das formas mais comuns.

ALTER TABLE

A maneira mais fácil de mover uma tabela de uma ferramenta para outra é com a expressão ALTER TABLE. O seguinte comando converte mytable para Falcon:

```
mysql> ALTER TABLE mytable ENGINE = Falcon;
```

Esta sintaxe funciona para todas as ferramentas de armazenamento, mas há uma desvantagem: pode levar muito tempo. MySQL vai executar uma cópia linha por linha da sua tabela antiga na sua tabela nova. Durante este tempo, você provavelmente estará usando toda a capacidade de E/S de disco do servidor, e a tabela original será bloqueada para leitura enquanto a conversão acontece. Então, tenha cuidado antes de tentar esta técnica em uma tabela ocupada. Ao invés disso, você pode usar um dos métodos discutidos a seguir, que envolvem fazer uma cópia da tabela primeiro.

Quando você converte uma ferramenta de armazenamento para outro, qualquer característica específica à ferramenta de armazenamento é perdida. Por exemplo, se você converter uma tabela InnoDB para MyISAM e voltar novamente, você perderá qualquer chave estrangeira originalmente definida na tabela InnoDB.

Esvaziar e importar

Para ganhar mais controle sobre o processo de conversão, você pode escolher primeiro esvaziar a tabela a um arquivo de texto usando a utilidade mysqldump. Depois de você ter esvaziado a tabela, você pode, simplesmente, editar o arquivo vazio para ajustar a expressão CREATE TABLE que ele contém. Tenha certeza de modificar o nome da tabela, assim como seu tipo, porque você não pode ter duas tabelas com o mesmo nome no mesmo banco de dados, mesmo se elas forem de tipos diferentes – e mysqldump padroniza em escrever o comando DROP TABLE antes de CREATE TABLE, então você pode perder seus dados se não for cuidadoso!

Veja Capítulo 11 para mais conselhos sobre como apagar e recarregar dados eficientemente.

CREATE e SELECT

A terceira técnica de conversão é um termo comum entre a velocidade do primeiro mecanismo e a segurança do segundo. Ao invés de apagar a tabela inteira ou convertê-la toda de uma vez, crie a nova tabela e usa a sintaxe INSERT ... SELECT do MySQL para populá-la, como segue:

```
mysql> CREATE TABLE innodb_table LIKE myisam_table;
mysql> ALTER TABLE innodb_table ENGINE=InnoDB;
mysql> INSERT INTO innodb_table SELECT * FROM myisam_table;
```

Isso funciona bem se você não tiver muitos dados, mas se você tiver, geralmente é mais eficiente popular a tabela progressivamente, comitando a transação entre cada bloco para que os logs de cancelamento não cresçam muito. Supondo que id é a chave primária, execute esta consulta repetidamente (usando valores maiores de x e y cada vez) até que você tenha copiado todos os dados na tabela nova:

```
mysql> START TRANSACTION;
mysql> INSERT INTO innodb_table SELECT * FROM myisam_table
    -> WHERE id BETWEEN x AND y;
mysql> COMMIT;
```

Depois de fazer isso, será deixada a você a tabela original, que você pode limpar quando tiver terminado o trabalho com ela, e a tabela nova, que não está totalmente populada. Tenha o cuidado de bloquear a tabela original, se necessário, para evitar obter uma cópia inconsistente dos dados!