SISTEMA GERENCIADOR DE BANCO DE DADOS

AULA 5

Prof. Leonel da Rocha

CONVERSA INICIAL

Nesta etapa, vamos tratar de assuntos importantes para o planejamento e a criação de um banco de dados capaz de impactar positivamente na performance de utilização. Em relação ao planejamento, que é muito importante para uma boa performance, vamos estudar a normalização de dados, que trata da organização de um banco de dados para reduzir a redundância de dados e garantir um aumento de integridade. Na sequência, vamos tratar do controle de concorrência, que organiza a utilização simultânea por usuários em um mesmo objeto, prevenindo dessa maneira inconsistências nos dados. A otimização de consultas será outro dos nossos temas. Vamos avaliar de que forma o acesso aos dados deve ser implementado, privilegiando a rapidez e a qualidade dos resultados obtidos. Para ajudar essa otimização, vamos estudar as estatísticas de consultas, uma ferramenta que permite verificar como a consulta se comporta em relação à performance. Outro assunto pertinente à performance de um banco de dados é a utilização de índices, itens importantes na classificação de dados e na velocidade de acesso a eles.

TEMA 1 – NORMALIZAÇÃO DE DADOS

A normalização de banco de dados é um conjunto de regras que tem por objetivo organizar um projeto de banco de dados para reduzir a redundância e aumentar a integridade e o desempenho. Para normalizar um banco de dados, é preciso examinar os atributos de uma entidade e as relações entre as entidades, com o objetivo de evitar anomalias observadas em termos de inclusão, exclusão e alteração de dados.

Para aplicar a normalização a um projeto de banco de dados, é necessário avaliar as entidades e os seus atributos, tendo como base cinco regras de normalização, que recebem o nome de *formas normais*. Pensando na transformação das entidades e de seus atributos em tabelas e colunas, a normalização corresponde a um conjunto de regras de simplificação e adequação dessas tabelas. A tabela de um banco de dados relacional está em uma certa forma normal quando satisfaz determinadas condições.

O trabalho original de Edgar F. Codd, definiu três dessas formas, mas existem outras formas normais aceitas para os projetos atuais. Cada forma normal que veremos neste material representa uma condição mais forte das que

a precedem na lista. Para a maioria dos projetos implementados, podemos considerar que as bases de dados estão normalizadas se aderem à terceira forma normal.

Para dar início ao processo, precisamos definir todos os atributos relacionados a uma entidade principal, sendo que um deles deve ser um atributo identificador. Depois dessa parte inicial, partimos para a análise do documento de acordo com as formas normais descritas a seguir.

Na Primeira Forma Normal, ou 1FN, os atributos precisam ser atômicos, ou seja, as tabelas não podem ter valores repetidos e nem atributos com mais de um valor. Por exemplo, vamos analisar a tabela funcionário e um de seus atributos, que é o telefone: FUNCIONARIO = {IDENTIFICADOR + ENDERECO + TELEFONES}. Nesse caso, um funcionário pode ter mais de um número de telefone. Sendo assim, o atributo "TELEFONES" é multivalorado. Para normalizar, será preciso identificar a chave primária e a coluna que apresenta dados repetidos, nesse caso "TELEFONES", e removê-los da entidade FUNCIONARIO. Devemos então criar uma outra entidade com o atributo em questão, lembrando de estabelecer uma relação entre as duas entidades: FUNCIONARIO = {IDENTIFICADOR + ENDEREÇO} e TELEFONE (nova entidade) = {FUNCIONARIO_IDENTIFICADOR (que será uma chave estrangeira) + TELEFONE}.

Passagem à 1FN:

- Determinar a chave primária da entidade;
- Descobrir quais são os atributos da entidade que apresentam dados repetidos, para que sejam removidos;
- Criar uma entidade para os atributos repetidos, com a chave primária da anterior;
- Por fim, relacionar a nova entidade e a principal.

Vejamos agora a Segunda Forma Normal, ou 2FN. Lembrando que, para estar na 2FN, é preciso estar também na 1FN. 2FN define que os atributos normais, ou seja, os que não são identificadores, devem depender unicamente do atributo identificador da entidade. Assim, os atributos da entidade, que não são dependentes dessa chave, devem ser removidos da entidade principal, com a criação de uma nova entidade utilizando esses atributos.

Exemplo: FUNCIONARIO_CARGO = {ID_FUNC + NOME_FUNC + ID_CARGO + SALARIO + DESCRICAO_CARGO}. Como podemos observar, o atributo "DESCRICAO_CARGO" não depende unicamente da chave primária "ID_FUNC", mas somente da chave "ID_CARGO". Para normalizar, é necessário identificar os dados não dependentes da chave primária (nesse exemplo "DESCRICAO_CARGO") e removê-los, criando uma nova entidade com os dados em questão: FUNCIONARIO_CARGO = {ID_FUNC + ID_CARGO + SALARIO} e CARGO (nova entidade) = {ID_CARGO + DESCRICAO_CARGO}.

Passagem à 2FN:

- Geração de novas entidades com Dependências Funcionais (DF) completas.
- Análise de dependências funcionais: descrição cargo depende de id_cargo; nome e salário dependem de id_func; data_início depende de toda a chave.

Quadro 1 – Exemplo 1

ID_FUNC	NOME_FUNC	ID_CARGO	VALOR_SALARIO	DESCRICAO_CARGO	DATA_INICIO
1	João	10	5.000,00	Analista	11/01/2000
2	Maria	20	8.000,00	Desenvolvedor	25/02/2005
3	Ana	8	7.000,00	DBA	30/04/2010

Fonte: Rocha, 2023.

Agora, veremos a Terceira Forma Normal, ou 3FN. Se para estar na 2FN é preciso estar na 1FN, para estar na 3FN é preciso estar na 2FN. A 3FN define que todos os atributos dessa entidade devem ser funcionalmente independentes uns dos outros. Ao mesmo tempo, devem ser dependentes exclusivamente da chave primária da tabela. A 3FN foi projetada para melhorar o desempenho de processamento dos bancos de dados e minimizar os custos de armazenamento.

Exemplo: FUNCIONARIO = {ID_FUNC + NOME + VALOR_SALARIO + VALOR_FGTS}. O valor do FGTS é proporcional ao salário, logo o atributo normal "VALOR_FGTS" depende também do atributo normal "VALOR_SALARIO". Para normalizar, é preciso identificar os atributos normais dependentes de outros atributos normais. Nesse exemplo, "VALOR_FGTS" é dependente do atributo normal “VALOR_SALARIO”. Depois, vamos removê-los

da tabela. Esses atributos, como são itens calculados, devem ser definitivamente excluídos, deixando para a camada de negócio a responsabilidade pelo seu cálculo. Também devem ser movidos para uma nova tabela, referenciando a principal.

Passagem à 3FN:

- Para estar na 3FN, precisa estar na 2FN;
- Geração de novas tabelas com DF diretas;
- Análise de dependências funcionais entre atributos não-chave;
- Verificar a dependência exclusiva da chave primária;
- Entidades na 3FN também não podem conter atributos que sejam resultados de algum cálculo de outro atributo.

Quadro 2 – Exemplo 2

ID_FUNC	NOME_FUNC	VALOR_SALARIO	VALOR_FGTS
1	João	5.000,00	400,00
2	Maria	8.000,00	640,00
3	Ana	7.000,00	560,00

Fonte: Rocha, 2023.

A Quarta Forma Normal, ou 4FN, requer que não exista nenhuma dependência multivalorada não trivial de conjuntos de atributo em algo mais do que um superconjunto de uma chave candidata.

Passagem à 4FN:

- Geração de novas tabelas, eliminando dependências multivaloradas;
- Análise de dependências multivaloradas entre atributos:
- Funcionário, telefone → dependência multivalorada de id_func.

Quadro 3 – Exemplo 3

ID_FUNC	NOME_FUNC	TEL_FUNC
1	João	(41) 89999.0000 (41) 3333.0000 (11) 70000.4545
2	Maria	(41) 79999.0000 (41) 5555.0000
3	Ana	(41) 68888.0001 (41) 5000.6000 (21) 454545.4545

Fonte: Rocha, 2023.

A Quinta Forma Normal (ou 5FN ou PJ/NF) requer que não exista dependências de joins (associações) não triviais que não venham de restrições-chave. Aqui, podemos ter atributos compostos que poderiam ser subdivididos em vários atributos. Por exemplo, podemos citar um atributo ENDEREÇO com as informações bairro, estado e cidade em uma mesma célula.

Quadro 4 – Exemplo 4

ID_FUNC	NOME_FUNC	ENDERECO
1	João	Rua XV, 35 - CENTRO - CURITIBA - PR
2	Maria	Rua da Liberdade, 350 - REBOUÇAS - SÃO PAULO - PR
3	Ana	Av. do Estado, 111 - BOTAFOGO - SALVADOR - BA

Fonte: Rocha, 2023.

TEMA 2 – CONTROLE DE CONCORRÊNCIA

Um banco de dados, quando é utilizado por mais de um usuário, terá de administrar a concorrência entre as informações que estão sendo acessadas pelos usuários. Essa administração é conhecida como controle de concorrência. Ela ocorre quando, em um banco de dados, os usuários tentam acessar a mesma informação simultaneamente, sendo necessário então um controle desses acessos. Para solucionar esse tipo de problema, existem diversas técnicas de controle de concorrência, que são usadas como forma de assegurar a propriedade de não interferência entre uma operação e outra, ou ainda o isolamento dessas transações executadas ao mesmo tempo. Algumas dessas técnicas implementam a serialização, que é a execução das transações de forma serial. Para entender melhor esse tema, é preciso entender que transações são todas as operações executadas entre o início e o fim de uma operação de manutenção de dados. Para gerenciar as transações. é necessário conhecer as propriedades comumente chamadas de ACID (acrônimo de Atomicidade, Consistência, Isolamento e Durabilidade), que devem ser usadas pelos métodos de controle de concorrência e recuperação do SGBD:

- Atomicidade é o princípio segundo o qual uma transação é uma unidade de processamento atômica, ou seja, a transação deve ser realizada por completo ou não deve ser realizada de forma alguma. Caso haja alguma falha durante a transação, os efeitos parciais dessa transação no banco devem ser desfeitos. Para que essa transação, onde houve falha, seja

desfeita, é necessário que o banco de dados emita o comando que desfaz a transação, garantindo assim a integridade do banco.

- A preservação da consistência permite que uma transação seja executada do início ao fim, sem a interferência de outras transações durante a sua execução – ou seja, implica a execução de uma transação isolada, levando o banco de um estado consistente para outro.
- Isolamento por sua vez, garante que uma transação seja isolada de outras transações, mesmo com várias transações executadas simultaneamente. O sistema vai garantir que, para cada conjunto de transações, uma transação seja encerrada antes do início da outra. Ou seja, a execução de uma determinada transação não sofrerá a interferência de nenhuma outra transação que esteja acontecendo simultaneamente.
- Durabilidade, ou permanência, é a garantia de que as mudanças que ocorrem no banco de dados, ao término de uma transação finalizada com sucesso, persistam no banco. Ou seja, depois que uma operação é encerrada com êxito, os dados gravados devem consistir no banco de dados, mesmo em caso de falhas no sistema.

Para a implementação do controle de concorrência, podemos utilizar a técnica de bloqueio em duas fases, que se baseia no bloqueio de itens de dados. Chamamos de bloqueio uma variável que fica atrelada ao item de dados. Esse bloqueio é binário, ou seja, apresenta somente dois estados: o item de dados está bloqueado ou não está bloqueado. Esse tipo de procedimento permite que o item de dado só esteja acessível para uma transação se a variável não estiver bloqueada. São utilizadas duas operações para o bloqueio binário: lock(1), bloqueado; e unlock(0), desbloqueado. Quando o item de dados está sendo acessado, o estado da variável é lock. Assim que a transação for encerrada, o status da operação é alterado para unlock. A partir daí, o item já está disponível para outra transação.

Existem duas formas de bloquear (lock) os dados:

- Bloqueio compartilhado: quando uma transação apresente o bloqueio e a instrução é de leitura, mais de uma transação poderá acessar o mesmo dado. Entretanto, se a instrução for de gravação, ela não poderá participar

de um bloqueio compartilhado, pois nesse caso os dados serão alterados por apenas uma transação.

- Bloqueio exclusivo: quando uma transação recebe esse tipo de bloqueio, ela fica exclusivamente reservada para a instrução que compõe a transação, não permitindo que outra transação faça uso do dado que está sendo utilizado. Logo, apenas uma transação mantém o bloqueio no item.

Uma transação precisa manter o bloqueio de um item de dado durante todo o tempo de acesso, mais um curto período após o término. Afinal, nem sempre é interessante fazer o desbloqueio imediato após o acesso final, pois isso pode comprometer a serialização em alguns casos. Sempre existe a necessidade de bloqueio e desbloqueio dos itens de dados, mas existem algumas situações que podem acarretar um problema no banco dados. Por exemplo, quando não é feito o desbloqueio antes da solicitação de um novo bloqueio a outro, situação em que pode ocorrer um deadlock, ou seja, travamento das transações do banco de dados. Considerando que, para a situação de deadlock, existem duas transações, a primeira espera por algum item que está bloqueado na segunda transação. Ao mesmo tempo, está acontecendo o contrário, ou seja, a segunda transação espera por algum item que está sendo bloqueado pela primeira transação. Assim, ambas ficam na fila de espera, aguardando a liberação do bloqueio do item que está com a outra transação. Como isso não vai acontecer, as duas transações acabam nunca sendo concluídas, o que gera muitos problemas para o banco de dados, inclusive um travamento total.

Existe ainda um problema conhecido como *starverd*, que pode ser exemplificado por três transações: a primeira faz a requisição de um bloqueio compartilhado de um item de dado; em seguida, a segunda transação faz uma requisição de bloqueio exclusivo do mesmo item de dado. Como a primeira transação está usando o bloqueio compartilhado, a segunda transação não terá a permissão de bloqueio exclusivo, de modo que ela fica na fila de espera, até a liberação. Enquanto a segunda transação está na fila, chega a terceira transação com pedido de bloqueio compartilhado do mesmo item. Como é possível fazer o bloqueio compartilhado, a terceira transação passa na frente da segunda e consegue o compartilhamento do item de dado. Quando a primeira transação finaliza a sua transação, o item de dado continuará bloqueado pela terceira transação. Enquanto isso, a segunda transação continua aguardando a liberação

total para que possa fazer o bloqueio exclusivo. Caso as próximas transações sejam sempre de acesso compartilhado desse mesmo item, a segunda transação nunca conseguirá fazer a sua transação, de modo que será considerada travada.

TEMA 3 – OTIMIZAÇÃO DE CONSULTAS

Para a otimização de um SGBD, é necessário identificar e eliminar os possíveis problemas de desempenho existentes em todos os níveis do sistema, entre eles as consultas lentas que são submetidas ao banco. Também é necessário melhorar as configurações do servidor de banco de dados, do sistema operacional e também do hardware que suporta o ambiente do sistema. Alguns aspectos da otimização não se aplicam apenas ao MySQL. Por isso, algumas metodologias mostradas aqui podem ser aplicadas em outros SGBDs disponíveis no mercado.

Porém, antes de estudar os itens anteriores, é preciso salientar que a otimização não é um processo simples, pois será preciso medir todos os aspectos do sistema para o entendimento de seu funcionamento. Ciente dessa característica, podemos determinar o ajuste mais adequado – por exemplo, o ajuste do SGBD para aplicações de leitura é diferente das aplicações de escrita. Lembrando que as medições de desempenho de sistema são imprescindíveis após a implementação de ajustes, pois os dados de desempenho obtidos vão servir de referência para determinar se um ajuste realizado no SGBD teve os efeitos esperados ou não.

Para a otimização das consultas SQL, durante o processo de projeto da base de dados, é preciso conhecer os tipos de consultas que serão mais comuns no sistema, para criar a base de dados buscando otimização do processo de extração de informações. As consultas devem ser desenvolvidas de forma que sejam executadas no menor tempo possível. Para isso, é preciso ter conhecimento das técnicas de desenvolvimento de consultas. Além disso, é preciso monitorar as consultas já implementadas, para descobrir se existe lentidão em alguma delas. Às vezes, será preciso tratá-las, seja pela reescrita da consulta ou através de alteração da sua aplicação, para garantir um acesso mais eficiente ao banco. Trata-se de uma situação que requer averiguação constante, já que em um ambiente onde existe um número grande de consultas, com um tempo de processamento elevado, prejudica o desempenho de forma

considerável. Para a otimização de consultas, é preciso entender a forma como elas são processadas pelo MySQL. Assim, é importante tentar atuar em cada etapa, buscando a redução do tempo de processamento e, como consequência, um melhor desempenho.

Uma vez eliminados os problemas relativos às consultas SQL, será possível modificar as configurações do MySQL para garantir um uso mais apropriado dos recursos disponíveis no sistema operacional e no hardware, melhorando assim o desempenho do banco. Para esse processo, é preciso conhecer como o MySQL funciona em termos de utilização de memória e disco, além de seus principais parâmetros de configuração relacionados ao desempenho. O MySQL apresenta um conjunto de ferramentas para o monitoramento e acompanhamento do servidor, para descobrir quais são os gargalos do sistema, para que possam ser tratados.

Para finalizar, precisamos monitorar o desempenho do Sistema Operacional onde o MySQL está instalado. No sistema operacional, podemos utilizar recursos mais apropriados para o banco, como sistema de arquivos mais eficientes, processos e threads nativas, além da própria escolha de um sistema operacional mais apropriado ao MySQL. Essa escolha pode gerar ganhos de desempenho em torno de 50%. Outro fator importante para o desempenho é a escolha do hardware onde o MySQL será instalado. Utilizando, por exemplo, um processador de 64 bits, é possível fazer o gerenciamento de bases maiores, o que também permite a alocação de uma quantidade maior de memória, com a configuração de buffers de memória maiores para o MySQL, com melhoria considerável do desempenho.

O MySQL apresenta um comando chamado “slow log”, que registra todas as consultas nas quais o tempo de execução é maior do que o parâmetro long-query-time, que por padrão é 10 segundos. Também é possível configurar esse log para registrar as consultas que não utilizam índices ou que realizam um SELECT *. O slow log vem desabilitado por padrão, podendo ser ativado através do parâmetro log-slow-queries. Ao executar o comando STATUS, o MySQL irá exibir, dentre outras informações, o SLOW QUERIES, que é o número de consultas lentas recebidas pelo servidor, desde o início do serviço do MySQL, ou desde o último FLUSH STATUS. Se o slow log estiver ativo, o que é recomendado, as consultas serão gravadas no arquivo, permitindo a identificação das consultas que são responsáveis pela lentidão do sistema.

Uma vez identificadas as consultas lentas, é preciso avaliar como o MySQL executa esses comandos. Para isso, devemos utilizar o comando EXPLAIN, que deve ser colocado antes do comando SELECT que vai ser analisado. Esse comando irá exibir o plano de execução escolhido pelo otimizador. A seguir, um exemplo da utilização do comando EXPLAIN para uma consulta aleatória:

```
EXPLAIN SELECT nomeA as 'Aluno', nomeD as 'Disciplina'
FROM aluno inner join cursa ON aluno.matricula=cursa.matricula
inner join disciplina ON disciplina.codigo = cursa.codigo
WHERE cursa.ano=2023
```

O quadro a seguir apresenta o resultado do comando EXPLAIN para a consulta anterior.

Quadro 5 – Resultado

id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	SIMPLE	a		ALL	PRIMARY				200	11,11	Using where
1	SIMPLE	fa		ref	PRIMARY, ix_fk_codigo	PRIMARY	2	db.leo.matricula	27	100	Using index
1	SIMPLE	f		eq_ref	PRIMARY	PRIMARY	2	db.leo.matricula	1	100	

Fonte: Rocha, 2023.

A seguir, o quadro apresenta os itens que o EXPLAIN nos mostra quando executado em uma consulta.

Quadro 6 – Itens

Coluna	Nome JSON	Significado
id	select_id	SELECT identificador
select_type	Nenhum	SELECT tipo
table	table_name	A tabela para a linha de saída
partitions	partitions	As partições correspondentes
type	access_type	O tipo de junção
possible_keys	possible_keys	Os índices possíveis para escolher
key	Key	O índice realmente escolhido
key_len	key_length	O comprimento da chave escolhida

ref	Ref	As colunas comparadas com o índice
rows	rows	Estimativa de linhas a serem examinadas
filtered	filtered	Porcentagem de linhas filtradas pela condição da tabela
Extra	Nenhum	Informações adicionais

Fonte: Rocha, 2023.

Existem diversas informações apresentadas pelo EXPLAIN. A primeira delas é o SELECT_TYPE, que mostra o tipo de consulta que está sendo processada. Elas podem ser consultas simples ou sem sub-consultas (SIMPLE) e SUB_QUERY ou UNION para comando que apresentam consultas aninhadas. Além disto, o EXPLAIN mostra os índices que estão disponíveis para a execução do comando (coluna POSSIBLE_KEYS). O índice que está utilizado para a leitura dos dados aparece na coluna KEY (NULL, caso não esteja fazendo uso de índices).

Vale destacar que será utilizado apenas um índice para cada tabela lida pelo MySQL, por isso a criação do índice deve ser feita com critério – isto é, sempre compondo as colunas que serão empregadas no WHERE.

A coluna ROWS fornece o número de linhas lidas pelo MySQL para buscar o resultado. Idealmente, esse número deve ser igual ao número de linhas retornadas pelo comando. A coluna REF indica a coluna utilizada para referenciar tabelas em JOIN, enquanto o EXTRA fornece informações adicionais sobre a execução, como uso de tabelas temporárias e ordenação.

Vejamos algumas importantes dicas de otimização para um processo de melhoria de planos de execução de consultas:

- Reescreva a consulta para percorrer um menor caminho – por exemplo, utilize as dicas para o otimizador ou campos indexados na cláusula WHERE, evitando SELECT *;
- Altere a ordem de leitura das tabelas, para ler sempre a tabela com menos registros;
- Configure o MySQL para utilizar sempre um índice;
- Indexe novas colunas quando necessário;
- Compare sempre as colunas indexadas com valores constantes e nunca aplique funções ou expressões ao índice.

TEMA 4 – ESTATÍSTICAS DE CONSULTA

Para otimizar o desempenho de um SGBD, é necessário realizar as rotinas de forma eficiente e eficaz. Principalmente em grandes coleções de dados, trata-se de uma questão fundamental. Devemos considerar a eficiência, mediante execução de consultas e alterações realizadas nos dados armazenados. A eficácia refere-se ao bom funcionamento. buscando atender as necessidades do indivíduo que utiliza um SGBD específico, dentre os vários existentes no mercado. Também pode estar relacionada com a qualidade de um produto.

O MySQL se tornou um dos mais populares SGBDs *open source* do mercado porque apresenta consistência, alta performance e confiabilidade, além de ser fácil de usar. Ele tem mais de 70 milhões de instalações, que vão desde instalações em grandes corporações até aplicações embarcadas e muitos sistemas web.

O MySQL se tornou a escolha de uma nova geração de aplicações que utilizam Linux, Apache, MySQL, PHP (modelo LAMP) e WEB 2.0. Ele pode ser utilizado tranquilamente em mais de 20 plataformas, incluindo Linux, Windows, HP UX, AIX, Netware, Solaris e outras, oferecendo flexibilidade e controle.

Para um bom desempenho das consultas de um banco de dados, é importante que o projeto seja bem desenvolvido. Após a sua implementação, é preciso realizar um acompanhamento operacional para averiguar o desempenho. Ajustes devem ser implementados caso necessário, tendo como base dados reais do comportamento de acesso ao banco de dados, em busca do melhor desempenho e da maior estabilidade.

Essa fase é chamada de *database tuning* ou simplesmente *tuning*. As informações coletadas no projeto físico podem ser revisadas por meio da coleta de estatísticas sobre os padrões de uso da corporação. O processo de utilização dos recursos, bem como de processamento interno do SGBD, pode ser monitorado para revelar gargalos, como a disputa pelos mesmos dados por vários usuários ou processos de forma simultânea.

Dessa forma, o volume de atividades e o tamanho da base de dados podem ser estimados com maior precisão. Surge, assim, a necessidade de monitoramento e revisão do projeto físico do banco de dados, constantemente, visto que os objetivos do *tuning* são os seguintes:

- Fazer com que as aplicações sejam executadas da forma mais rápida possível;
- Baixar o tempo de resposta das consultas e transações;
- Otimizar o desempenho geral das transações;
- A revisão das decisões de projeto na fase de *tuning* consiste em um ajuste do projeto; além disso, as informações de entrada para o processo de *tuning* incluem estatísticas relacionadas a diversos fatores.

Um SGBD em particular pode coletar internamente as seguintes estatísticas:

- Tamanho de tabelas individuais;
- Número de valores distintos em uma coluna;
- Número de vezes em que uma consulta ou transação em particular é submetida/executada em um intervalo de tempo.

Figura 1 – Estatística de desempenho

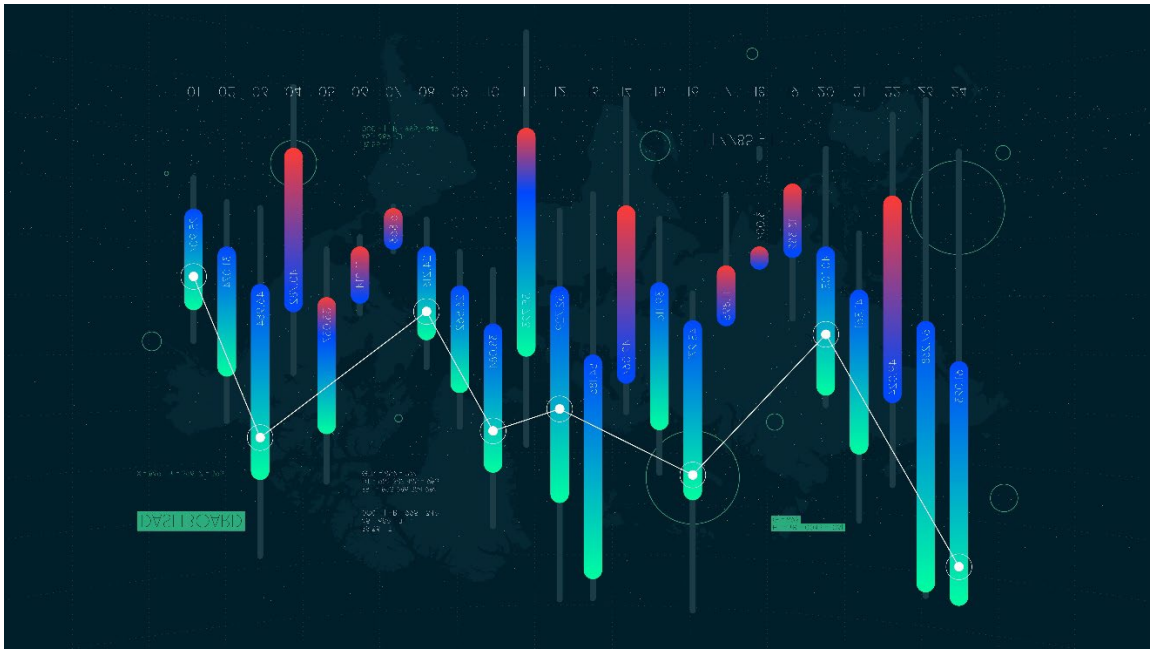
Crédito: Champ008/Shutterstock.

Essas e outras estatísticas criam um perfil do conteúdo e do uso do banco de dados. Podem ser levantadas ainda as seguintes informações, através de estatísticas de monitoramento de atividades do banco de dados:

- Armazenamento: informações sobre a alocação de recursos para armazenamento de tabelas e de índices, além de portas de buffer;
- Desempenho de entrada/saída (I/O): atividade total de leitura/escrita e paginação do disco;
- Processamento de consultas: tempos de execução de consultas, tempos de otimização de consultas;
- Bloqueios e registro de log: taxas de definição de diferentes tipos de bloqueios, taxas de desempenho de transações e registros de log de atividades.

As estatísticas citadas, em sua maioria, referem-se a transações, controle de concorrência e recuperação de dados. Porém, o *tuning* de bancos de dados envolve tratar diretamente as excessivas disputas por bloqueios, a concorrência entre as transações, a sobrecarga de registro de logs, o armazenamento desnecessário de dados, a otimização do tamanho do buffer, o escalonamento de processos e, finalmente, a alocação de recursos como discos, memória e processos para uma utilização mais eficiente.

Figura 2 – Estatística de transações



Crédito: Maxger/Shutterstock.

O *tuning*, que é um processo de evolução de análise e otimização, podendo ser empregado nos SGBDs com o objetivo de melhorar o desempenho de acordo com a aplicação utilizada.

É dada ênfase à configuração dos parâmetros do MySQL para a sua otimização, visto que um ambiente que apresenta inúmeras transações concorrentes, executando consultas simples ou complexas, pode apresentar problemas de gargalos, de modo que uma pessoa sem conhecimento técnico culpará o próprio SGBD por um serviço abaixo das expectativas, no cenário do projeto e da implementação dos objetos de um banco de dados.

Figura 3 – *Tuning* ou ajuste fino

Crédito: jossnat/Shutterstock.

É necessário observar que, ao modificar certos parâmetros do MySQL ou de qualquer outro SGBD, pode haver melhoria de desempenho; porém, liberar para o SGBD, de forma exagerada, recursos do sistema operacional, ou mesmo de hardware, pode não resultar em ganho de desempenho. Às vezes acontece o inverso do desejado: perda de eficiência, tanto do SGBD quanto do sistema que acessa o banco de dados.

Podemos concluir que, no momento de escolher um SGBD para uma aplicação, é necessário considerar quais recursos serão consumidos e o que o SGBD disponibiliza, para não prejudicar o desempenho do sistema como um todo. As estatísticas de consulta servem para mapear os recursos disponíveis

de maneira simples e direta, seja em relação ao sistema operacional, ao hardware ou ao próprio SGBD.

TEMA 5 – OTIMIZAÇÃO DE UM DB MYSQL UTILIZANDO ÍNDICES

Os índices em um SGBD são importantíssimos quando o assunto é otimização de desempenho, pois é através dos índices que os dados são arranjados em uma tabela, de modo que a sua recuperação é muito mais rápida.

Para criar um índice, é necessário conhecer o sistema que está acessando o banco de dados e as consultas que ele faz. Sem esse conhecimento, é impossível melhorar o desempenho. Além disso, corre-se o risco de uma piora no acesso aos dados. Para iniciar o processo, a seguir veremos algumas características básicas que devem ser levadas em consideração:

- Os índices devem ser criados levando em consideração a seleção de dados que será feita na tabela ou nas tabelas. Uma dica importante em relação à seleção de dados e a criação de índices: é importante levar em consideração os campos utilizados na cláusula de restrição WHERE. Dessa forma, o primeiro passo é fazer um planejamento e encontrar os campos que estão tomando tempo e recursos do servidor.
- Para a criação propriamente dita, utilizando o phpMyAdmin, localizar e selecionar a opção “Estrutura“.
- Então, desça um pouco a tela. Na tabela de índices, observe a cardinalidade, que é o número de registros ou linhas da sua chave primária ou índice MySQL. Essa informação é importante, pois quando é realizada, nessa tabela, uma consulta (ou *query*), o MySQL irá procurar dados em cada um dos registros, um por um. Imagine o esforço para uma tabela que apresenta, por exemplo, um milhão de registros. Com um índice criado corretamente, a cardinalidade da chave precisa ficar entre 50 e 400 registros. Claro que tudo depende da sua aplicação, do número de registro, entre outros fatores. Quando criamos índices, é possível diminuir a nossa busca para um volume muito menor de dados. Portanto, o desempenho do nosso banco de dados será otimizado significativamente, principalmente se o número de registros for muito grande.

- No phpMyAdmin, ir em “SQL”, para digitar o comando para criar um índice:

CREATE INDEX <nome_do_indice> ON <tabela> (<campo>(<tamanho>));

Em nosso exemplo, utilizamos o seguinte comando:

CREATE INDEX IX_NOME_FUNC ON FUNCIONARIO (NOME(5))

Mas o que significa de fato o “tamanho” dos Índices MySQL? Considere que estamos criando um índice de tamanho (5).

Quadro 7 – Exemplo: nomes

NOME
Maria da Silva
Pedro Amaral
Ana Laurentino
Maria Silveira
Ana da Silva
Pedro Silva
Maria de Lourdes
Pedro Augusto

Fonte: Rocha, 2023.

Sem a criação de um índice, ao realizar uma pesquisa, nosso banco de dados iria passar por todos os 8 nomes acima. Porém, ao criar um índice com 5 posições, o MySQL organizará os dados conforme o quadro a seguir.

Quadro 8 – Exemplo: com índice

Chave	Referências
Ana	Ana da Silva
	Ana Laurentino
Maria	Maria da Silva
	Maria de Lourdes
	Maria Silveira
Pedro	Pedro Amaral
	Pedro Augusto
	Pedro Silva

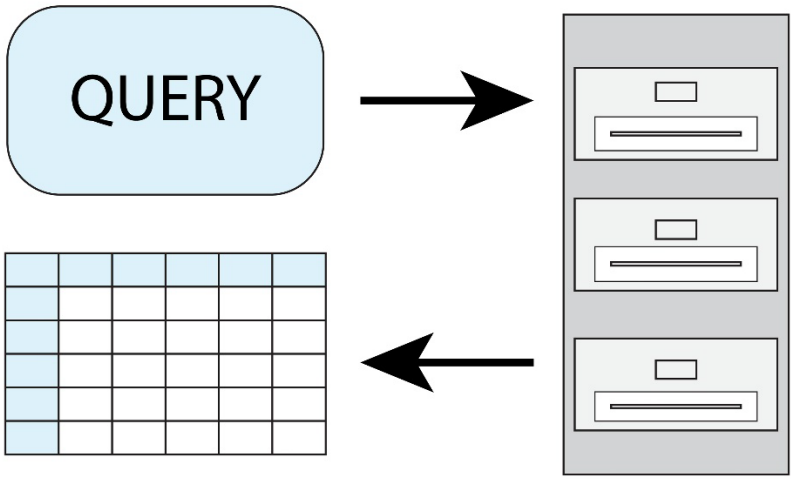
Fonte: Rocha, 2023.

Ou seja, ao fazer uma query: SELECT * FROM FUNCIONARIO WHERE nome = “Maria de Lourdes”, ao invés do MySQL percorrer todos os 8 registros

do exemplo de tabela acima, ele irá verificar apenas os 3 índices que criamos. Se você pensar em um grande volume de dados, essa otimização dentro de uma tabela é capaz de economizar muito tempo e processamento!

Dessa forma, com a aplicação de índices em um banco de dados com muitos registros, o tempo de execução sofrerá uma redução considerável. Esse método consegue otimizar muito o desempenho de um banco de dados no MySQL, apenas com por meio de índices.

Figura 4 – Consulta (ou *query*) em um banco de dados e resultado



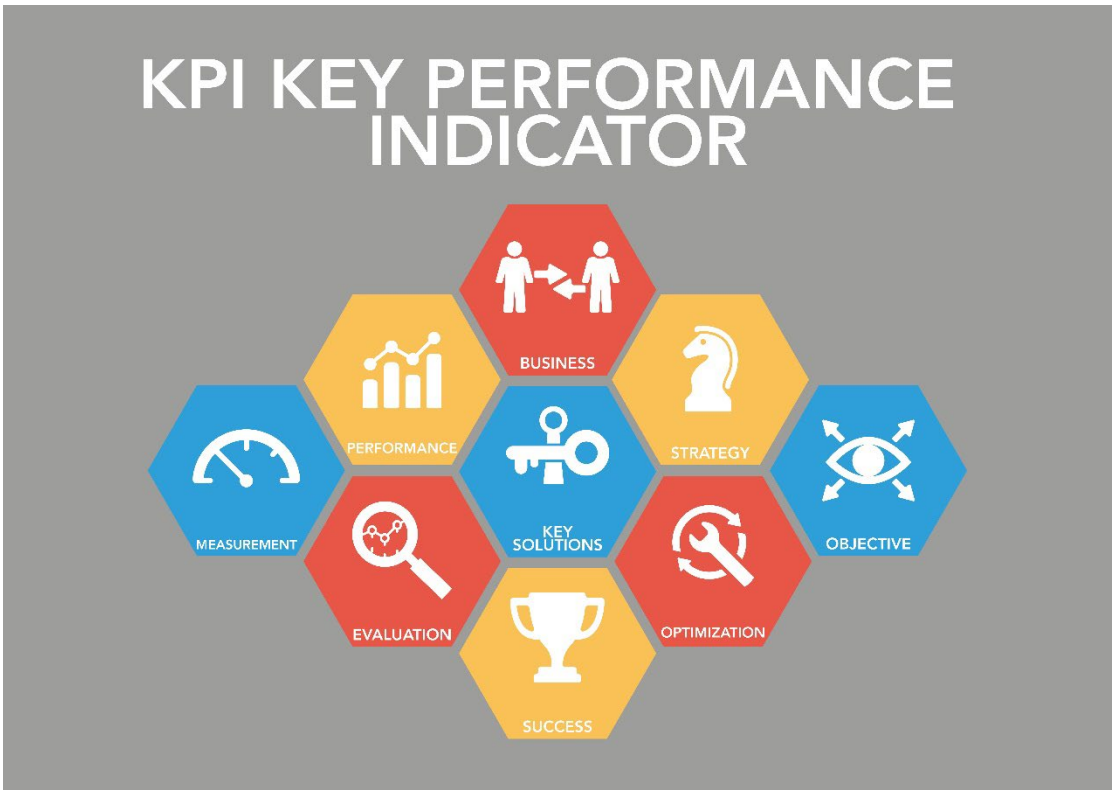
Crédito: Mary-Ang/Shutterstock.

Para descobrir qual campo você precisa indexar, basta executar as suas queries no phpMyAdmin, colocando “desc” antes. Dessa forma, o MySQL vai mostrar passo a passo o caminho que ele está percorrendo para fornecer um resultado. Exemplo:

```
desc SELECT * FROM funcionario WHERE nome='Maria'
```

Esse comando no MySQL mostra em qual coluna o índice deverá ser criado. Como dica, você deverá criar seus índices do MySQL com base nessa pesquisa.

Figura 5 – KPI (Indicador Chave de Desempenho)



Crédito: garagestock/Shutterstock.

FINALIZANDO

Nesta etapa, tratamos de alguns assuntos importantes para o planejamento e a criação de um banco de dados, pois impactam positivamente a performance de utilização. Em relação ao planejamento, que é muito importante para uma boa performance, estudamos a normalização de dados, que trata da organização de um banco de dados, visando reduzir a redundância de dados e aumentar a integridade. Vimos também o tema do controle de concorrência, que organiza a utilização simultânea por usuários em um mesmo objeto, prevenindo, dessa maneira, inconsistências nos dados. Tratamos da otimização de consultas, que nos mostra como o acesso aos dados deve ser implementado, privilegiando a rapidez e a qualidade dos resultados obtidos. Para auxiliar na otimização, estudamos as estatísticas de consultas, uma ferramenta que permite verificar como a consulta está se comportando em relação à performance. Outro assunto pertinente à performance de um banco de dados é a utilização de índices, itens importantes na classificação de dados e na velocidade de acesso a eles.