Fecomércio RS | Senac

Técnico em Informática
Módulo 4
Aula 10 - Introdução Tuning
BANCO DE DADOS II

Senac

Introdução Tuning

Disciplina de Banco de Dados II

Agenda 27/09/2016

- Introdução Tuning
- SQL Tuning
- Otimização de consultas SQL

Introdução Tuning

Disciplina de Banco de Dados II

Minimizar o tempo de resposta e recuperação de dados!

- Minimizar tempo de concorrência de acesso aos dados;
  - Muitos usuários
- Otimizar taxa de transferência de dados;
  - retornar apenas dados necessários
  - consulta menos dados
  - estruturas menores
- Otimizar a capacidade de carga do banco de dados;
  - retornar menos dados = menos trabalho do BD

Introdução Tuning

Disciplina de Banco de Dados II

Problemas de performance
(Michael r. Ault. https://cdsiusa.com/)

Introdução Tuning

Disciplina de Banco de Dados II

60% problemas de performance são decorrentes de SQL ruins

20% problemas de performance são decorrentes do Projeto BD

17% problemas de performance são decorrentes do Banco

3 % problemas de performance são decorrentes do SO

Introdução Tuning

Disciplina de Banco de Dados II

60% problemas de performance são decorrentes no BD!!!!

- Sql mal escritos
- Consultas
- Instruções precisam ser otimizadas

20% problemas de performance são decorrentes do Projeto BD

- tabelas
- índices
- visões
- modelagem ruim

Introdução Tuning

Disciplina de Banco de Dados II

17% problemas de performance são decorrentes do Banco

- Alocação de memória (ORACLE SGA)
- Estrutura do banco
- Parâmentros
- configurações ruins
- configurações que precisam ser otimizadas
- tuning de instância de dados (DBA)

3 % problemas de performance são decorrentes do SO

- Linux: I/O dados, largura de banda, alocação de espaço em disco, SWAP, parâmetros.
- parâmentros sistema operacional

Introdução Tuning

Disciplina de Banco de Dados II

60% SLQ ruim + 20% modelagem ruim = 80% dos problemas de performance!!!

Problemas de Escrita!!!!

Introdução Tuning

Disciplina de Banco de Dados II

Como melhorar o desempenho do meu BD a partir de uma escrita melhor??!!

Introdução Tuning

Disciplina de Banco de Dados II

Melhorando estrutura do BD
Otimizando Consultas
Ferramentas de desempenho
Criando Índices

Introdução Tuning

Disciplina de Banco de Dados II

Melhorando estrutura do BD

Introdução Tuning

Disciplina de Banco de Dados II

Criar base de dados cadastro;

Introdução Tuning

Disciplina de Banco de Dados II

Usando constraints (parâmentros)
Usando collation (definição de caracteres)

```
CREATE DATABASE cadastro
default character set utf8
default collate utf8_general_ci;
```

Introdução Tuning

Disciplina de Banco de Dados II

Criar tabela pessoas:

nome, idade, sexo, peso, altura, nacionalidade

Introdução Tuning

Disciplina de Banco de Dados II

Otimizar estrutura e alocação em disco
Utilizar constraints em tabelas
Economia de bytes

Introdução Tuning

Disciplina de Banco de Dados II

```
CREATE TABLE pessoas (
id int NOT NULL AUTO_INCREMENT,
nome varchar (30) NOT NULL,
nascimento date,
sexo enum ('M','F'),
peso decimal (5,2),
altura decimal (3,2),
nacionliade varchar (20) DEFAULT 'BRASIL',
PRIMARY KEY (id),
) DEFAULT CHARSET = utf8;
```

Introdução Tuning

Disciplina de Banco de Dados II

Otimizando Consultas

Introdução Tuning

Disciplina de Banco de Dados II

- **SORT e GROUP-BY:**

OBS.: Entender tempo de resposta x tempo total

Requerem que toda a sua entrada seja usada antes de retornar resultados para suas operações pai. Elas são conhecidas como operações que requerem materialização. As consultas implementadas com essas operações geralmente sofrem um atraso inicial devido à materialização. Após esse atraso inicial, em geral elas retornam registros com bastante rapidez.

Introdução Tuning

Disciplina de Banco de Dados II

- SELECT *

Não use o * em suas consultas. Um SELECT * cria uma sobrecarga maior sobre a tabela, entrada/saída e largura de banda na rede do servidor.

Exemplo:
SELECT * FROM clientes ORDER BY nome ASC;
SELECT nome FROM clientes ORDER BY nome ASC;

Introdução Tuning

Disciplina de Banco de Dados II

- JOIN

Reescrever uma subconsulta para usar JOIN e obter melhor desempenho. A vantagem de criar uma subconsulta JOIN é poder avaliar as tabelas em uma ordem diferente daquela definida pela consulta.

Introdução Tuning

Disciplina de Banco de Dados II

- **JOIN**

Por exemplo, para determinar todos os pedidos que tenham pelo menos um item com um desconto de 25 por cento ou mais, você pode usar a seguinte subconsulta EXISTS:

```
SELECT "Order ID" FROM Orders O
WHERE EXISTS (SELECT "Order ID"
FROM "Order Details" OD
WHERE O."Order ID" = OD."Order ID"
AND Discount >= 0.25)
```

Introdução Tuning

Disciplina de Banco de Dados II

- **JOIN**

É possível reescrever a subconsulta usando JOIN:

```
SELECT DISTINCT O."Order ID" FROM Orders O
INNER JOIN "Order Details"
OD ON O."Order ID" = OD."Order ID" WHERE
Discount >= 0.25
```

Introdução Tuning

Disciplina de Banco de Dados II

- Stored Procedure

Em consultas muito frequentes, verifique se não pode ser criado um procedimento armazenado (STORED PROCEDURE).

Normalmente a performance melhora ainda mais!

Isso se deve à otimizações no plano de execução dos procedimentos armazenados.

Introdução Tuning

Disciplina de Banco de Dados II

- **Top ou LIMIT**

Sempre restrinja a quantidade de linhas e colunas retornadas em suas consultas: assim você economiza o disco, memória e rede do servidor, além de ter resultados mais concisos e rápidos! Ou seja, sempre tente filtrar corretamente na cláusula WHERE ou restrinja o resultado usando outras formas, como a cláusula TOP.

Introdução Tuning

Disciplina de Banco de Dados II

- **Top ou LIMIT**

SELECT TOP 3 * FROM jogadores ORDER BY pontos DESC

Results

ID	NOME	PONTOS
1	JOÃO	10
2	MARIA	20
3	JOSÉ	30
4	FRANCISCA	40
5	PEDRO	50
6	ANTÔNIA	60
7	CARLOS	70

(7 row(s) affected)

Introdução Tuning

Disciplina de Banco de Dados II

- LOWER ou UPPER

O SQL Server é case insensitive: ou seja, ele não liga para maiúsculas e minúsculas. Poupe trabalho e não use funções como LOWER ou UPPER na hora de comparar VARCHARs.

Introdução Tuning

Disciplina de Banco de Dados II

- Operadores

A ordem decrescente de performance dos operadores em consultas é:

=

>, >=, <, <=

LIKE

<>

Introdução Tuning

Disciplina de Banco de Dados II

- **BETWEEN é melhor**

Usar consultas sempre que possível com BETWEEN.

(a) SELECT titulo FROM eventos WHERE '2014-02-01' >= date(inicio) AND '2014-02-01' <= date(fim)

(b) SELECT titulo FROM eventos WHERE '2014-02-01' BETWEEN date(inicio) AND date(fim)

Introdução Tuning

Disciplina de Banco de Dados II

- Wildcards %

Quando usar o operador LIKE, tente colocar os wildcards mais próximos do final dos VARCHARs.

(a) SELECT * FROM cidades WHERE nome LIKE 'ber%';

(b) SELECT * FROM cidade WHERE nome LIKE '%ber%';

Introdução Tuning

Disciplina de Banco de Dados II

- TRUNCATE

A pagar todas as linhas de uma tabela, prefira o comando TRUNCATE TABLE ao invés de DELETE.

TRUNCATE TABLE é mais rápido e utiliza menos recursos de sistema e log de transações.

```
TRUNCATE TABLE tbl_teste_incremento;
```

Introdução Tuning

Disciplina de Banco de Dados II

- **TRIGGERS**

Evite usar gatilhos, ou TRIGGERs. Use-os apenas como último recurso. Se for uma lógica muito extensa, prefira STORED PROCEDUREs.

Introdução Tuning

Disciplina de Banco de Dados II

Ferramentas de desempenho

Introdução Tuning

Disciplina de Banco de Dados II

- **MySQLTuner-perl:**

MySQLTuner é um script escrito em Perl que permite rever uma instalação MySQL rapidamente e fazer ajustes para aumentar o desempenho e estabilidade. As variáveis de configuração e dados de status atual é recuperada e apresentada em um formato breve, juntamente com algumas sugestões básicas de desempenho.

Introdução Tuning

Disciplina de Banco de Dados II

- **Query Store:**

Ferramenta que foi criada com intuito de facilitar a vida do DBA na identificação e solução de problemas de desempenho relacionados às consultas no banco de dados.

Introdução Tuning

Disciplina de Banco de Dados II

- Query Cache

O query cache salva resultados de SELECT's já executados e que seus dados brutos não tenham sido alterados tornando assim o tempo de resposta da query muito mais otimizada, pois ele vai buscar da memória e não do disco.

Introdução Tuning

Disciplina de Banco de Dados II

- Query Cache

Ele pode trabalhar de 3 maneiras diferentes, no nosso **my.ini** dentro do grupo **[mysqld]**:

query_cache_type = 0
Desligado
query_cache_type = 1
Ligado para todas as query's
query_cache_type = 2
Ligado sobre demanda

Introdução Tuning

Disciplina de Banco de Dados II

- Query Cache

Tendo em vista que toda a vez que você altera dados em alguma tabela, ele invalida o cache da mesma, o mais indicado é você fazer um mapeamento da frequência de atualização das tabelas e utilizar o cache para as que não tenham uma frequência grande a atualização/inserção, para que assim, o cache lhe dê um ganho de performance, e não fique sobrecarregando o servidor tendo que invalidar muitas querys.

Introdução Tuning

Disciplina de Banco de Dados II

- Query Cache

Por exemplo, a tabela de registros de log do sistema, a cada ação do usuário ele irá inserir um novo dado nesta tabela, logo ela não é uma boa candidata a ser cacheada, já a tabela de noticias do site, recebe atualizações 2 vezes ao dia, tá aí, uma boa candidata a ser cacheada,

Introdução Tuning

Disciplina de Banco de Dados II

- Query Cache

Estudo de Caso:

- Setado para ON DEMAND: query_cache_type = 2
- Tabela com 100 milhões de registros,
- Para a opção ON DEMAND temos que **especificar no select** que queremos que ele utilize o cache com SQL_CACHE:

Introdução Tuning

Disciplina de Banco de Dados II

- Query Cache

Rodando a consulta pela primeira vez:

```
mysql> SELECT SQL_CACHE * FROM noticia WHERE
conteudo LIKE "8555556%";
. . .
20 rows in set (2 min 15.20 sec)
```

Introdução Tuning

Disciplina de Banco de Dados II

- Query Cache

Colocamos o cache com a nossa consulta, ele demorou o tempo padrão, porém colocou-a em cache, veja o resultado da mesma query, rodando pela segunda vez:

```
mysql> SELECT SQL_CACHE * FROM noticia WHERE
conteudo LIKE "8555556%";
...
20 rows in set (0.00 sec)
```

Introdução Tuning

Disciplina de Banco de Dados II

- Query Cache

Notas:

Opção query_cache_size, valores em torno de 16M são recomendados, dependendo da capacidade do seu servidor.

Utilizar qualquer uma das funções abaixo inutiliza o cache na consulta:

```
BENCHMARK() CONNECTION_ID() CONVERT_TZ()
CURDATE() CURRENT_DATE() CURRENT_TIME()
CURRENT_TIMESTAMP() CURTIME() DATABASE()
ENCRYPT() with one parameter FOUND_ROWS() GET_LOCK()
LAST_INSERT_ID() LOAD_FILE() MASTER_POS_WAIT()
NOW() RAND() RELEASE_LOCK()
SLEEP() SYSDATE() UNIX_TIMESTAMP(
USER() UUID() UUID_SHORT()
```

Introdução Tuning

Disciplina de Banco de Dados II

- Slow log

O MySQL possui um log chamado “slow log”, onde são armazenadas todas as consultas cujo tempo de execução seja maior que o parâmetro long-query-time, que por padrão é 10 segundos. Além disto, pode-se configurar este log para armazenar também as consultas que não utilizam índices ou que realizam um SELECT *

Introdução Tuning

Disciplina de Banco de Dados II

Slow log

Por padrão este log vem desabilitado, e pode 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, contado desde um o início da execução do MySQL

Introdução Tuning

Disciplina de Banco de Dados II

Slow log

Caso o slow log esteja ativo, o que é recomendado, estas consultas serão gravadas neste arquivo e possibilitarão a **identificação dos comandos que são os gargalos do seu sistema.**

Introdução Tuning

Disciplina de Banco de Dados II

Explain

Avaliar de consultas lentas e como o MySQL está executando estes comandos.

Para isto faz-se uso do comando EXPLAIN, que deve ser colocado antes do comando SELECT a ser estudado. Este comando irá exibir o plano de execução escolhido pelo otimizador.

Introdução Tuning

Disciplina de Banco de Dados II

Explain

Existem diversas informações apresentadas pelo EXPLAIN:
SELECT_TYPE que mostra o tipo de consulta que está sendo processada. Estas podem ser consultas simples ou sem sub-consultas (SIMPLE) e SUB_QUERY ou UNION para comando que possuem consultas aninhadas.
A coluna POSSIBLE_KEYS fornece quais os índices estão disponíveis para a execução do comando, e o índice que ele está utilizando para a leitura do dados aparece na coluna KEY (NULL, caso não esteja fazendo uso de índices).

Introdução Tuning

Disciplina de Banco de Dados II

Explain

A coluna ROWS fornece o número de linhas lidas pelo MySQL para buscar o resultado, idealmente este número deve ser igual ao número de linhas retornadas pelo comando.

A coluna REF indica a coluna utilizada para referenciar tabelas em JOIN.

EXTRA fornece informações adicionais sobre a execução, tais como, o uso de tabelas temporárias, ordenação, dentre outros.

A coluna TYPE exibe o algoritmo de busca utilizado para a leitura dos dados.

Introdução Tuning

Disciplina de Banco de Dados II

Explain

O link apresenta os valores possíveis para esta coluna, indo do melhor para o pior tipo.

Introdução Tuning

Disciplina de Banco de Dados II

Explain

Exemplo:
EXPLAIN SELECT * FROM categorias

Introdução Tuning

Disciplina de Banco de Dados II

Explain

O EXPLAIN é muito útil para entender os problemas que a consulta possui e criar índices mais específicos e performáticos.

Introdução Tuning

Disciplina de Banco de Dados II

OPTIMIZE

No caso de tabelas que sofrem muita alteração nos dados, é recomendável executar com frequência o comando OPTIMIZE para que o MySQL organize os dados na tabela, melhorando também o seu desempenho.

Introdução Tuning

Disciplina de Banco de Dados II

OPTIMIZE

Nas tabelas MyISAM o OPTIMIZE TABLE faz o seguinte:

- **Se a tabela tem linhas deletadas ou divididas, repara a tabela.**
- **Se as páginas de índice não estão organizadas, as organiza.**
- **Se as estatísticas da tabela não estão atualizadas (e a reparação não pode ser completada organizando o índice), as atualiza.**

Introdução Tuning

Disciplina de Banco de Dados II

OPTIMIZE

- Rode o seguinte comando SQL para verificar quais as tabelas mais fragmentadas:

```
SHOW TABLE STATUS;
```

Qualquer valor no campo 'DATA_FREE' das tabelas MyIsam é indício de fragmentação.

Introdução Tuning

Disciplina de Banco de Dados II

Optimize

Desfragmentado as tabelas MyIsam:

Para desfragmentar uma tabela MyIsam basta rodar o seguinte comando:

```
optimize table NOMEDATABELA;
```

Para reindexar as tabelas InnoDB:

```
ALTER TABLE `NOMEDATABELA` ENGINE=InnoDB;
```

Introdução Tuning

Disciplina de Banco de Dados II

INDEX
(ÍNDICES)

Introdução Tuning

Disciplina de Banco de Dados II

Agilizar pesquisas de seleção de dados nas tabelas.

Introdução Tuning

Disciplina de Banco de Dados II

Permitir que as aplicações de banco de dados encontrem os dados mais rapidamente, sem ter que ler a tabela toda.

Introdução Tuning

Disciplina de Banco de Dados II

O uso de índices pode trazer grandes melhorias para o desempenho do banco de dados. Pensando nisso, devemos então, primeiramente, **entender como funciona o mecanismo de pesquisa que está trabalhando nos bastidores**.

Introdução Tuning

Disciplina de Banco de Dados II

Exame de tabela (FULL TABLE):

examina todas as páginas de dados das tabelas, começando do início da tabela passando por todos os registros, página a página e extraindo aqueles que satisfazem aos critérios da consulta.

Introdução Tuning

Disciplina de Banco de Dados II

Exame de índices (INDEX):

Percorrendo a estrutura da árvore do índice para localizar os registros, por comparação, extraindo somente aqueles registros necessários para satisfazerem os critérios passados pela consulta.

Introdução Tuning

Disciplina de Banco de Dados II

Por que criar índices?
Os índices aceleram a recuperação dos dados!

Por exemplo, imagine que você compre um livro de 800 páginas para suas pesquisas científicas e este não apresente em seu conteúdo um índice reportando o seu conteúdo.

Se você precisar de várias pesquisas, seria muito desagradável ficar horas procurando o conteúdo que deseja estudar.

Introdução Tuning

Disciplina de Banco de Dados II

Por que criar índices?

Por outro lado, um livro que apresente um índice de suas abordagens, se faz muito mais fácil e torna as pesquisas até prazerosas, pois teremos condição de irmos direto ao ponto que queremos, rapidamente!!!

Introdução Tuning

Disciplina de Banco de Dados II

Índices são sempre bem vindos em **colunas de grande seletividade**, como por exemplo, além da chave primária, que muitas vezes pode circular como identificador único da entidade na sua aplicação, você pode ter também um índice para **colunas que** poderão lhe auxiliar em consultas em que estas **contarão com a cláusula WHERE**, precisando ou não usar os operadores AND, OR ou *NOT, que muitas vezes, em casos específicos, alteram a performance da consulta.

Nota: *O operador NOT sempre deixará sua consulta mais lenta que o normal.

Introdução Tuning

Disciplina de Banco de Dados II

SHOW INDEX FROM tabela

Introdução Tuning

Disciplina de Banco de Dados II

SINTAX:

CREATE INDEX nome_índice ON tabela (nome_coluna);

EXEMPLO:

CREATE idx_nome ON tb_cliente (nome);

Revisão Banco de Dados

Disciplina de Banco de Dados II

 alan.correa.sul@gmail.com

 alanalvescorrea

 github.com/alanalvescorrea/