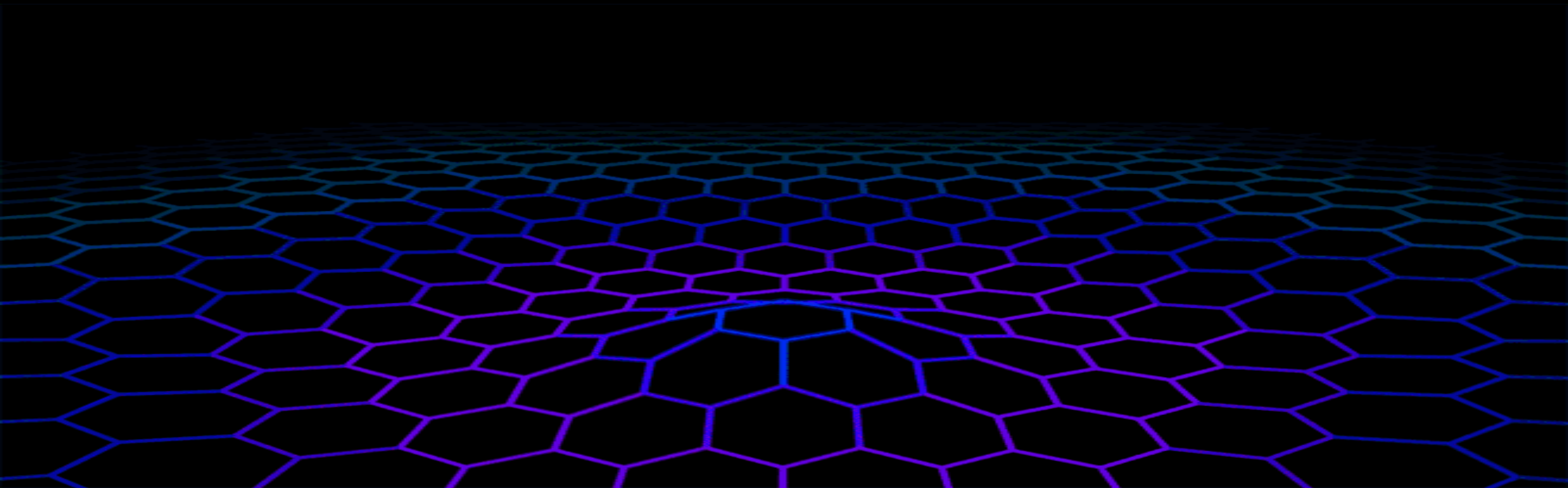


Banco de Dados 2



INDEXAÇÃO

INDEX

Para que serve um índice?

Auxiliam o MySQL a encontrar os registros de interesse mais rapidamente nas buscas

Podem ser criados junto com a tabela, ou posteriormente

Indexam uma ou mais colunas de interesse

Serve para organizar os dados e agilizar a pesquisa/consulta dos dados armazenado nas tabelas.

Para que serve um índice?

Utilizados para encontrar registros com um valor específico de uma coluna rapidamente.

Sem um índice o MySQL tem de iniciar com o primeiro registro e depois ler através de toda a tabela até que ele encontre os registros relevantes.

O que indexar?

Colunas (campos):

- Que sofrem muitas buscas (cláusula WHERE)
- Que tem uma gama de valores bastante variada

Quem define a indexação necessária é a aplicação que consome o banco de dados

Atenção

Índices ocupam espaço em disco, e em memória, ao serem utilizados

Não criar índices em colunas que não necessitem

Se um índice ficar muito grande, o MySQL pode escolher não utilizá-lo, pois o peso de carregá-lo na memória poderá ser maior do que varrer a tabela inteira

Cada inserção, remoção ou atualização da tabela gera a necessidade de atualizar os índices

Index

Para colunas **CHAR** e **VARCHAR**, índices que utilizam apenas parte da coluna podem ser criados, usando a sintaxe **nome_coluna(length)** para indexar os primeiros **LENGTH()** bytes de cada valor da coluna.

Exemplo de criação de Index

Pode ser usado tanto no SQL SERVER, MySQL ou Oracle:

Índice na tabela tab_cliente, sendo que o campo de pesquisa é o CPF (numero do CPF)

```
CREATE INDEX nome_do_indice ON tab_cliente(cpf);
```

Index

Exemplo de index para colunas do tipo char e varchar:

```
CREATE INDEX indice1 ON empregado (nome(10));
```

Como a maioria dos nomes normalmente diferem nos primeiros 10 caracteres, este índice não deve ser muito menor que um índice criado com toda a coluna `nome_empregado`.

Versões compatíveis

Você só pode adicionar um índice em uma coluna que pode ter valores apenas se você estiver usando o MySQL Versão 3.23.2 ou mais novo e estiver usando os tipos de tabelas MyISAM, InnoDB, ou BDB.

Index

Normalmente para um melhor resultado de performance, o ideal para criar índices são campos:

- o Que sejam chaves;
- o Campos que façam JOIN com outras tabelas.
- o Campos que sejam números (tipo: INTEGER, NUMERIC)

Drop Index

```
DROP INDEX nome_indice ON nome_tabela;
```

DROP INDEX apaga o índice chamado nome_indice da tabela nome_tabela;

DROP INDEX não funciona em versões do MySQL anteriores a 3.22.


Criando um índice

Criação de index segue o padrão ANSI (o mesmo comando para criar index em um determinado SGBD, pode ser usado em qualquer outro).

Criando um índice

Normalmente você cria todos os índices em uma tabela ao mesmo tempo em que a própria tabela é criada com `CREATE TABLE`.

Teste




```
DROP DATABASE IF EXISTS aula11;  
CREATE DATABASE IF NOT EXISTS aula11;  
USE aula11;
```

```
DROP TABLE IF EXISTS cliente;
```

```
CREATE TABLE cliente (  
  id      MEDIUMINT(8) UNSIGNED NOT NULL AUTO_INCREMENT,  
  nome    VARCHAR(255) NOT NULL,  
  email   VARCHAR(255) DEFAULT NULL,  
  cidade  VARCHAR(255),  
  PRIMARY KEY (id)  
) ENGINE=MyISAM AUTO_INCREMENT=1;
```


Criando um índice

Exemplo de índice para os 10 primeiros caracteres do campo nome:



```
ALTER TABLE cliente ADD INDEX(nome(10));
```

Obs.: **Dúvida cruel.**

Indexar um campo inteiro geraria um índice muito grande

- Quanto mais caracteres indexados, maior o índice, e mais memória necessária para carregá-lo

Indexar poucos caracteres não ajuda muito na busca

- Pouca variação de valores

Busca com e sem Índice



```
SELECT SQL_NO_CACHE * FROM cliente WHERE email LIKE "%_it.e%" AND cidade LIKE "%rg%";
```

id	nome	email	cidade
2405	Ruth Boyd	elit.erat@aol.org	Sarpsborg
2426	Nayda T. Baker	adipiscing.elit.etiam@google.net	Orenburg

```
2 rows in set, 1 warning (0.04 sec)
```

```
ALTER TABLE cliente ADD INDEX(email(50));
```

```
Query OK, 3545 rows affected (0.10 sec)
```

```
Records: 3545 Duplicates: 0 Warnings: 0
```

```
SELECT SQL_NO_CACHE * FROM cliente WHERE email LIKE "%_it.e%" AND cidade LIKE "%rg%";
```

id	nome	email	cidade
2405	Ruth Boyd	elit.erat@aol.org	Sarpsborg
2426	Nayda T. Baker	adipiscing.elit.etiam@google.net	Orenburg

```
2 rows in set, 1 warning (0.03 sec)
```

MySQL Performance

Diferenças entre InnoDB e MyISAM

InnoDB

Engine Default do MySQL a partir da versão 5.5

Suporta transações ACID - Atomicidade, Consistência, Isolamento e Durabilidade

Suporta FOREIGN KEYS e integridade referencial (Constraints SET NULL, SET DEFAULT , RESTRICT e CASCADE)

Alto desempenho com grandes volumes de dados e um número elevado de concorrência entre leitura e escrita

Implementa *lock* de registro (melhor gerenciamento da concorrência, como Oracle, DB2...)

Facilidade de implementação dos níveis de isolamento

MyISAM

Derivado de um outro, mais antigo, chamado ISAM — Indexed Sequential Access Method, originalmente desenvolvido pela IBM, para ser usado em mainframes. Ainda dentro da IBM

Não tem suporte a transações ou a restrições de chaves estrangeiras.

Dúvida cruel

InnoDB X MyISAM

InnoDB x MyISAM

InnoDB funciona mais rápido que MyISAM quando há modificações constantes nos dados

MyISAM é melhor para tabelas fixas como cidades, países, ...

InnoDB usa a proteção por registros (*row locking*)

MyISAM usa proteção por tabelas (*table locking*)

MyISAM

FULLTEXT – MATCH - AGAINST



```
SELECT nome  
FROM alunos  
WHERE MATCH(nome, obs) AGAINST ('Jordan Sanders feugiat');
```

```
SELECT nome, MATCH(nome, obs) AGAINST ('Jordan Sanders feugiat')  
FROM alunos  
ORDER BY MATCH(nome, obs) AGAINST ('Jordan Sanders feugiat') DESC;
```

Crie as tabelas



```
CREATE TABLE usuario(  
    id          INT(11) AUTO_INCREMENT,  
    nome        VARCHAR(255),  
    email       VARCHAR(255),  
    senha       VARCHAR(255),  
    PRIMARY KEY (id)  
)ENGINE=MyISAM;
```

```
CREATE TABLE atividade(  
    id INT(11) AUTO_INCREMENT,  
    idUsuario INT(11),  
    dataHora  TIMESTAMP,  
    PRIMARY KEY (id)  
)ENGINE=MyISAM;
```

Usuários Randômicos



```
DELIMITER $$
CREATE PROCEDURE insereUsuario(qtd INT)
BEGIN
    DECLARE v_limite INT DEFAULT qtd;
    DECLARE v_contador INT DEFAULT 0;
    START TRANSACTION;
    WHILE v_contador < v_limite DO
        INSERT INTO usuario (nome, email, senha) VALUES (
            CONCAT ('nomedocara', FLOOR((RAND() * 900000))),
            CONCAT ('emaildocara', '@' , FLOOR((RAND() * 900000)), '.com.br'),
            CONCAT ('senhadocara', MD5(RAND())));
        SET v_contador = v_contador + 1;
    END WHILE;
    COMMIT;
END $$
DELIMITER ;
```

Atividades Randômicas



```
DELIMITER $$
CREATE PROCEDURE insereAtividade(qtd INT)
BEGIN
    DECLARE v_limite INT DEFAULT qtd;
    DECLARE v_contador INT DEFAULT 0;
    SELECT MAX(id) INTO @Vlimite FROM usuario;
    START TRANSACTION;
    WHILE v_contador < v_limite DO
        INSERT INTO atividade (idUserario) VALUES (1+RAND()* v_limite);
        SET v_contador = v_contador + 1;
    END WHILE;
    COMMIT;
END $$
DELIMITER ;
```

INSERINDO “ÀS GANHA”!



```
CALL insereUsuario(50000000); -- Duvidei hahahah
```

```
CALL insereUsuario(5000); -- Ok. Vamos na humildade ;)
```

```
CALL insereAtividade(1000000);
```

Análise e otimização de queries no MySQL

Análise de queries
Comando **EXPLAIN**

Comando EXPLAIN

Mostra como o MySQL planeja executar uma query

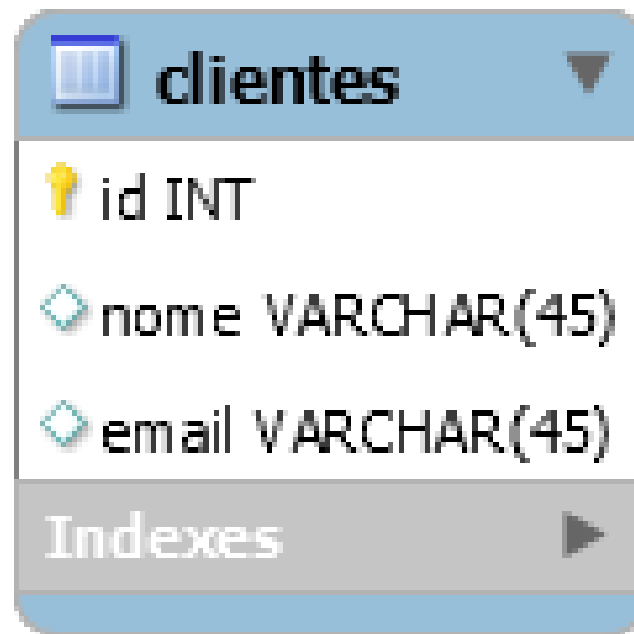
Funciona com comandos SELECT, DELETE, INSERT, REPLACE e UPDATE.

Mostra possíveis gargalos na execução

Comando EXPLAIN

Tabela **clientes**

- 50 milhões de registros
- Sem índices



clientes	
id	INT
nome	VARCHAR(45)
email	VARCHAR(45)
Indexes	

Comando EXPLAIN

Selecionar todos os clientes de nome comece por 'Br'



```
EXPLAIN SELECT * FROM clientes WHERE nome LIKE 'Br%';
```

id	select_type	table	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	SIMPLE	clientes	ALL	nome	NULL	NULL	NULL	45232432	11.11	Using where

Comando EXPLAIN

possible_keys mostra quais são os índices que podem ser utilizados para acelerar a query. Se o valor retornado é NULL, não há índices possíveis, e o MySQL terá que buscar na tabela inteira.

key mostra qual das possible_keys foi escolhida para acelerar a query. Se é NULL, não há índice possível para otimizar esta query.

rows é o número de registros que o MySQL acredita que terá de analisar para encontrar o resultado. Quanto maior, pior.

Extra são diversas informações adicionais.

Comando EXPLAIN

Selecionar todos os clientes de nome comece por 'Br'



```
SELECT * FROM clientes WHERE nome LIKE 'Br%';
```

NOME	EMAIL
Brandon S. Kelly	Integer.id@infaucibusorci.co.uk
Breanna H. Rasmussen	dictum.eu.eleifend@Donec.edu
Bryar Hays	augue.porttitor.interdum@liberonec.org
Brody Howard	luctus.et.ultrices@apurus.com
Brendan W. Jennings	libero@feugiatLorem.edu

5 rows in set (19.36 sec)

Teste

Selecionar todos os clientes de nome comece por 'Br'



```
EXPLAIN SELECT * FROM clientes WHERE nome LIKE 'Br%';
```

Sem índice:

id	select_type	table	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	SIMPLE	clientes	ALL	nome	NULL	NULL	NULL	45232432	11.11	Using where

Com índice:

id	select_type	table	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	SIMPLE	clientes	range	nome	nome	13	NULL	3	100.00	Using where



mysqlslap

mysqlslap

Pequena ferramenta de diagnóstico que acompanha o MySQL

Testar a carga de servidores de banco de dados

Pode emular um grande número de conexões de cliente que chegam ao servidor de banco de dados simultaneamente.

BATENDO FORTE NO BANCO com mysqlslap



mysqlslap

```
--user=root  
--password=  
--auto-generate-sql  
--concurrency=100  
--iterations=10  
--number-char-cols=10  
--number-int-cols=5  
--engine=innodb
```

100 conexões concorrentes, com comandos auto gerados (você pode também testar seus próprios comandos SQL), com 10 execuções, em uma tabela com 5 colunas INT, e com 10 coluna CHAR, e apenas com o Engine InnoDB

BATENDO FORTE NO BANCO com mysqlslap



```
mysqlslap --user=root --password --host=localhost --concurrency=88 --iterations=100 --create-schema=aula12 --query="SELECT * FROM alunos;"
```

```
mysqlslap --concurrency=8 --iterations=100 --number-int-cols=2 --number-char-cols=3 --auto-generate-sql
```

```
mysqlslap --delimiter=";" --create="CREATE TABLE a (b int);INSERT INTO a VALUES (23)" --query="SELECT * FROM a" --concurrency=50 --iterations=200
```

BATENDO FORTE NO BANCO com mysqlslap

```
D:\wamp64\bin\mysql\mysql8.0.18\bin>mysqlslap --concurrency=8 --iterations=100 --number-int-cols=2 --number-char-cols=3 --auto-generate-sql  
Benchmark
```

```
Average number of seconds to run all queries: 0.080 seconds
```

```
Minimum number of seconds to run all queries: 0.031 seconds
```

```
Maximum number of seconds to run all queries: 1.047 seconds
```

```
Number of clients running queries: 8
```

```
Average number of queries per client: 0
```

Storage Engines / motores - Mecanismos de Armazenamento

Principais características, recursos ou funcionalidades inerentes aos storage engines:

- Capacidade Transacional
- Meio de armazenamento
- Índices
- Integridade Referencial
- Tipo de travamento
- BOL – Backup On-Line
- Auto Recovery

Storage Engines / motores - Mecanismos de Armazenamento

Capacidade Transacional

Capacidade da tabela aceitar múltiplos acessos (de múltiplos usuários / aplicações), com colisão e travamento mínimos, sem que um usuário interfira com a operação do outro.

É poder executar comandos em blocos (transação), ao invés de executar um comando SQL por vez.

É estar de acordo com o modelo ACID (Atômico, Consistente, Isolado e Durável).

Storage Engines / motores - Mecanismos de Armazenamento

Meio de armazenamento

No MySQL/MariaDB, dependendo do motor escolhido, pode-se gravar a tabela 100% em memória (nada no disco)

Em vários outros bancos de dados, toda tabela grava e lê os dados de uma única forma padrão.

Pode-se gravar dados em uma TABLESPACE, como no Oracle, por exemplo.

Storage Engines / motores - Mecanismos de Armazenamento

Meio de armazenamento

Pode-se utilizar uma tecnologia dos anos 70, extramente rápida, conhecida como ISAM, para gravar dados e recuperá-los (leitura) de forma muito rápida.

Outro tipo de motor pode gravar os dados de forma compactada, como em um arquivo ZIP, economizando muito espaço em disco.

Storage Engines / motores - Mecanismos de Armazenamento

Meio de armazenamento

Ainda, pode-se gravar em formato CSV que facilita muito a integração com equipamentos de rede e telefonia, por exemplo.

Ao invés de ler e gravar os dados no servidor onde o MySQL/MariaDB está instalado, pode-se espalhar os dados por vários computadores para se criar um cluster de alta disponibilidade e alta performance.

Storage Engines / motores - Mecanismos de Armazenamento

Índices

Dependendo do motor, temos índices do tipo B-TREE, B+TREE, RTREE (spatial), ou FULL TEXT (que indexa palavras ao invés da coluna ou campo inteiro, e permite buscas como fazemos no Google, digitando palavras fora de ordem).

Storage Engines / motores - Mecanismos de Armazenamento

Integridade Referencial

FK (foreign key).

Há motores que usam, e, motores que não usam.

Dependendo da aplicação ou finalidade da tabela isto não é necessário.

É uma funcionalidade que pesa para o banco de dados.

Às vezes, não ter este recurso pode ser uma vantagem em termos de velocidade.

Storage Engines / motores - Mecanismos de Armazenamento

Tipo de travamento

Capacidade de poder travar um único registro (linha), vários registros ou a tabela inteira.

Cada motor tem um ou mais tipo de travamentos à disposição da tabela.

Storage Engines / motores - Mecanismos de Armazenamento

BOL – Backup On-Line

Fazer backup on line (com todo mundo trabalhando, sem precisar parar o banco).

Tem um custo alto.

Se a sua aplicação não roda H24 (24 horas por dia), significa que você tem janela de backup. Talvez seja interessante abrir mão deste recurso para ter mais agilidade nas leituras e escritas.

Storage Engines / motores - Mecanismos de Armazenamento

Auto Recovery

Há motores que caso haja uma corrupção de índice, por exemplo, você se verá obrigado a indisponibilizar o banco para rodar um comando básico de REPAIR TABLE.

Há outros, no entanto, que no máximo você será avisado, através do log de erro, que o MySQL/MariaDB server encontrou uma inconsistência e, já consertou.

Storage Engines / motores - Mecanismos de Armazenamento

Storage engine

Storage engine É uma característica única e exclusiva do MySQL/MariaDB que permite escolher um conjunto de recursos pré-definidos que melhor atenda aos requisitos e às necessidades de sua tabela, respeitando, o hardware disponível.

É muito mais do que, simplesmente, tipo de tabela

Fonte: <http://www.alexandremalmeida.com.br/2011/06/17/o-que-e-storage-engine-parte-1/>

Mecanismos de Armazenamento suportados pelo MySQL

O MySQL possui uma característica um pouco diferente dos outros SGBDs, uma vez que no MySQL é possível escolher o tipo da tabela no momento da criação da mesma.

O formato de armazenamento dos dados, bem como alguns recursos do banco de dados são dependentes do tipo de tabela escolhido.

Mecanismos de Armazenamento suportados pelo MySQL

InnoDB

É um mecanismo de armazenamento transacional seguro (compatível com ACID).

Possuí *rollback* e *commit* de transações, como também recuperação de desastres para proteger os dados do usuário.

Mecanismos de Armazenamento suportados pelo MySQL

InnoDB

Efetua bloqueio em nível de linha (sem escalção de bloqueios de granularidades maiores) e controle de concorrência multi-usuário.

Para manter a integridade referencial dos dados, o InnoDB suporta chaves estrangeiras (Foreign Key) mantendo as restrições referenciadas nos dados armazenados.

É o motor de armazenamento padrão do MySQL a partir da versão 5.5.5

Mecanismos de Armazenamento suportados pelo MySQL

MyISAM

É um dos mecanismos de armazenamento do MySQL mais utilizado na Web e para armazenamento de dados históricos (data warehousing).

O MyISAM é suportado em todas as versões do MySQL

Era o motor de armazenamento padrão do MySQL até a versão "5.5.5"

Mecanismos de Armazenamento suportados pelo MySQL

Merge

Permite agrupar logicamente uma série de tabelas MyISAM idênticas e referencia-las como um objeto.

Prática usada em bancos de dados com grandes volumes de dados.

Mecanismos de Armazenamento suportados pelo MySQL

Archive

É um mecanismo de armazenamento que fornece a solução ideal para armazenar e recuperar grandes quantidades de dados historicamente referenciados , arquivando informações de auditoria e de segurança.

Mecanismos de Armazenamento suportados pelo MySQL

Federated

É um mecanismo de armazenamento que oferece a capacidade de vincular servidores MySQL separados para criar um banco de dados lógico em vários servidores físicos.

Opção boa para implementar em ambientes distribuídos ou data marts.

Mecanismos de Armazenamento suportados pelo MySQL

CSV

É um mecanismo de armazenamento de dados em arquivos texto utilizando um separador comum para formatar os valores nele armazenado.

Pode-se utilizar o motor CSV facilmente para efetuar troca de dados entre aplicações que exportam e importam dados no formato CSV.

Mecanismos de Armazenamento suportados pelo MySQL

Não se está restrito a utilizar apenas um mecanismo de armazenamento.

Pode-se utilizar um mecanismo por tabela, esquema ou banco de dados.

Mecanismos de Armazenamento suportados pelo MySQL

Exemplo:

```
CREATE TABLE teste1(  
  t1 varchar(10)  
) engine=innodb;
```

```
CREATE TABLE teste2(  
  t2 varchar(10)  
) engine=myisam;
```

```
SHOW TABLES;
```

```
+-----+  
| Tables_in_teste |  
+-----+  
| teste1          |  
| teste2          |  
+-----+
```

Testes com MyISAM, INNODB e índices

Criação banco e tabelas

```
5 CREATE TABLE t1(  
6     id INT NOT NULL AUTO_INCREMENT PRIMARY KEY,  
7     valor INT NOT NULL  
8 ) ENGINE=MyISAM;  
9 CREATE INDEX idx_t1valor ON t1(valor);  
10  
11 create TABLE t2(  
12     id INT NOT NULL AUTO_INCREMENT PRIMARY KEY,  
13     valor INT NOT NULL  
14 ) ENGINE=MyISAM;  
15  
16 CREATE TABLE t3(  
17     id INT NOT NULL AUTO_INCREMENT PRIMARY KEY,  
18     valor INT NOT NULL  
19 ) ENGINE=INNODB;  
20 CREATE INDEX idx_t3valor ON t3(valor);  
21  
22 create TABLE t4(  
23     id INT NOT NULL AUTO_INCREMENT PRIMARY KEY,  
24     valor INT NOT NULL  
25 ) ENGINE=INNODB;
```

Criação procedure e chamada procedure

```
27 DELIMITER $$
28 CREATE PROCEDURE gerandodados(IN qtdLinhas INT, IN minVal INT, IN maxVal INT)
29 BEGIN
30     DECLARE i INT;
31     DECLARE tmp INT;
32     SET i = 1;
33     START TRANSACTION;
34     WHILE i <= qtdLinhas DO
35         SET tmp = minVal + CEIL(RAND() * (maxVal - minVal));
36         -- CEIL ou CEILING -> arredonda para cima
37         INSERT INTO t1 (valor) VALUES (tmp);
38         INSERT INTO t2 (valor) VALUES (tmp);
39         INSERT INTO t3 (valor) VALUES (tmp);
40         INSERT INTO t4 (valor) VALUES (tmp);
41         SET i = i + 1;
42     END WHILE;
43     COMMIT;
44 END$$
45 DELIMITER ;
46
47 CALL gerandodados(1000000, 10, 999999);
```

Criação procedure e chamada procedure

```
49  SELECT SQL_NO_CACHE *
50  FROM t1
51  WHERE valor = 312471;
52
53  SELECT SQL_NO_CACHE *
54  FROM t2
55  WHERE valor = 312471;
56
57  SELECT SQL_NO_CACHE *
58  FROM t3
59  WHERE valor = 312471;
60
61  SELECT SQL_NO_CACHE *
62  FROM t4
63  WHERE valor = 312471;
```



MUITO
OBRIGADO!

BAZINGA!