5 – Introdução

- Existem diversos sistemas gerenciadores de banco de dados no mercado, sendo que cada um deles possui suas características específicas, que os diferenciam dos demais.

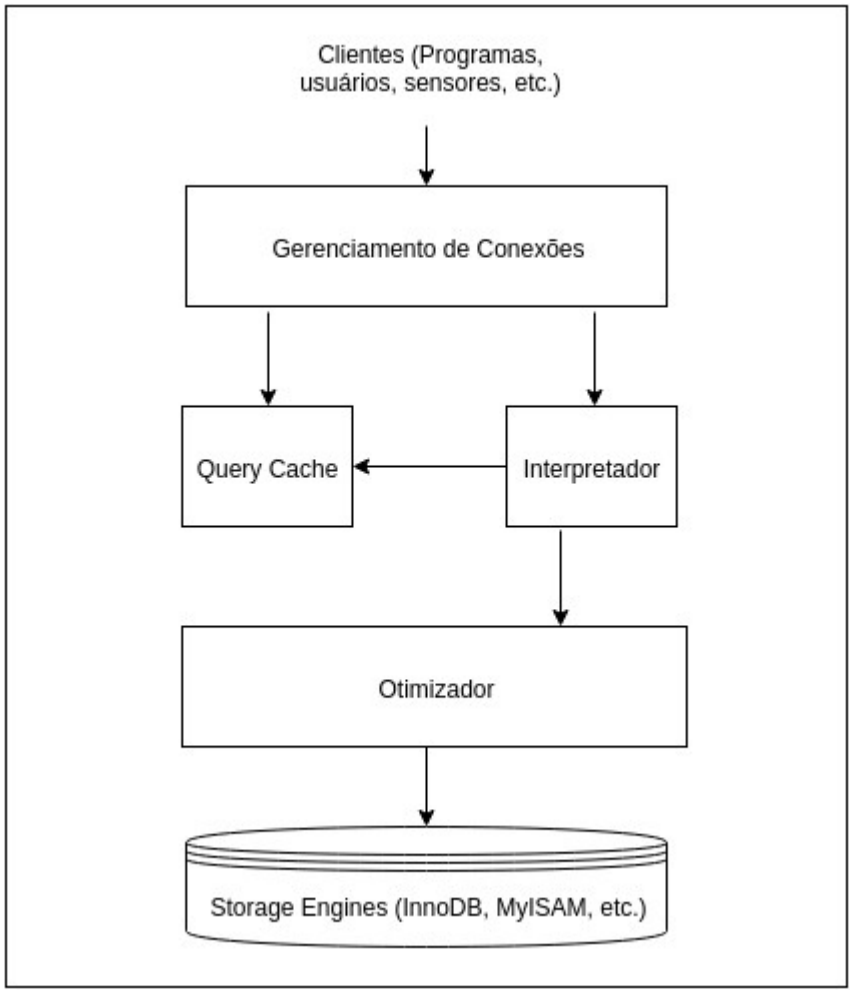
- O MySQL, assim como qualquer outro SGBD, possui as suas limitações, mas é um banco de dados bem flexível e estável, podendo trabalhar em qualquer tipo de ambiente. Atualmente é o SGBD mais usado no mundo.

- Para se utilizar ao máximo o potencial de um SGBD, é importante entender sua arquitetura e configurações, com o objetivo de customizá-lo da melhor forma para cada tipo de aplicação.

- A seguir serão mostrados os principais conceitos e características do sistema gerenciador de banco de dados MySQL.

5.1 – Arquitetura Lógica do MySQL

- A figura abaixo mostra como é formada a arquitetura do MySQL:



5.1 – Arquitetura Lógica do MySQL

- Os elementos que formam o MySQL são:

1. **Gerenciamento de Conexões**: Responsável por autenticar o usuário por meio de seu login, senha e host. Após autenticar, verifica se o usuário possui privilégio para executar as consultas desejadas.

2. **Interpretador**: Responsável por interpretar todos os comandos SQL, sempre verificando se o Query Cache possui os dados necessários armazenados.

3. **Query Cache**: Mantém armazenado o resultado das últimas consultas, com o objetivo de agilizar novas solicitações dos mesmos dados.

5.1 – Arquitetura Lógica do MySQL

4. **Otimizador**: É um dos elementos mais complexos do MySQL, responsável por otimizar os processos de busca e escrita de dados. Desta forma, o otimizador muitas vezes modifica automaticamente o código SQL para acelerar o processo.

5. **Storage Engine**: É o local onde os dados são armazenados. Cada uma das storage engine disponível no MySQL possui sua forma de armazenamento específica.

5.2 – Otimização

- O MySQL, por meio de seu otimizador, aplica automaticamente uma série de otimizações nas consultas SQL executadas pelos usuários.

- Estas otimizações incluem reescrever a consulta, determinar a ordem na qual irá ler a tabela, escolher o índice que será usado, etc.

- Antes mesmo de interpretar a consulta SQL feita pelo usuário, o MySQL verifica se alguma consulta idêntica está armazenada no Query Cache (que armazena somente consultas). Caso encontre, não será necessário nem interpretar o comando SELECT.

5.3 – Controle de Concorrência

- Toda vez que mais de uma operação de escrita ocorre ao mesmo tempo, o problema de controle de concorrência surge.

- Como exemplo, pode-se imaginar um usuário atualizando um registro de uma tabela ao mesmo tempo que outro usuário está tentando excluir este mesmo registro. Caso o SGBD não trate esta situação, problemas de consistência podem surgir.

- Os SGBD's que trabalham com operações de leitura/escrita concorrentes, geralmente implementam um sistema de bloqueio. Basicamente, existem dois tipos de bloqueio:

  1. **Bloqueio de Leitura**: Diversos usuários podem acessar um determinado registro ao mesmo tempo, mas nenhum usuário pode alterar este registro durante o bloqueio.

  2. **Bloqueio de Escrita**: São exclusivos, ou seja, quando um registro está sofrendo alterações, qualquer leitura ou outras solicitações de escrita neste registro ficam bloqueadas.

5.4 – Granularidade de Bloqueio

- A melhor forma de garantir o compartilhamento de dados com um bom desempenho é ser mais seletivo sobre o que será bloqueado. A ideia consiste em bloquear somente parte dos recursos que se deseja alterar, e não ele como um todo.

- Por exemplo, ao invés de bloquear uma tabela inteira, seria melhor bloquear somente a linha ou a coluna que está sendo alterada. Permitindo que mais alterações possam ser realizadas ao mesmo tempo.

- O problema é que gerenciar os bloqueios consome recursos do banco de dados. Todo o processo de checar, liberar e bloquear um recurso requer uma fatia considerável de recursos do SGBD.

5.4 – Granularidade de Bloqueio

- As duas principais estratégias de bloqueio são:

1. **Bloqueio de Tabela**: Estratégia básica e mais segura de bloqueio, pois quando um registro de uma determinada tabela vai ser alterado, toda a tabela é bloqueada para escrita. Isso pode comprometer seriamente o desempenho do banco de dados.

2. **Bloqueios de Linha**: Esse bloqueio oferece maior concorrência, mas também carrega um maior risco de inconsistências, pois somente a linha que será alterada será bloqueada.

5.5 – Transações

- As transações são formadas por um grupo de comandos SQL que são tratados atomicamente, como uma unidade única de trabalho.

- Caso o SGBD possa executar todos os comandos da transação, ela será finalizada com sucesso, mas se por algum motivo um dos comandos não puder ser executado, todas as operações são desfeitas.

- Uma aplicação bancária é um exemplo clássico no uso de transações. Vamos imaginar que um cliente deseja transferir o dinheiro de sua conta corrente para sua conta poupança.

5.5 – Transações

- Para realizar essa operação de transferência, três passos devem ser seguidos:

1. Verificar se o saldo da conta corrente é suficiente.
2. Subtrair o valor a ser transferido do saldo da conta corrente.
3. Adicionar o valor transferido no saldo da poupança.

- Nesse caso, para a transferência ser realizada, obrigatoriamente os três passos devem ser realizados. Se um deles falhar, a operação não pode ser realizada, ou irá gerar uma inconsistência.

- Desta forma, estas três operações devem ser realizadas dentro de uma transação, para que o banco garanta a execução das 3 instruções ou não realize nenhuma.

5.5 – Transações

- O código para realização de uma possível transferência de R$200,00 entre a conta corrente e a poupança é mostrado no quadro abaixo:

```
START TRANSACTION;
SELECT saldo FROM conta_corrente WHERE id_cliente='2';
UPDATE conta_corrente SET saldo=saldo-200 WHERE id_cliente='2';
UPDATE poupanca SET saldo=saldo+200 WHERE id_cliente='2';
COMMIT;
```

5.5 – Transações

- O SGBD precisa gerenciar diversas transações de diversos usuários ao mesmo tempo. E para que tudo funciona adequadamente, o servidor deve passar no teste ACID:

1. **Atomicidade**: A transação deve funcionar como uma unidade única e indivisível, de forma que ela seja executada por inteira, ou não seja executada.

2. **Consistência**: A transação deve mover de um estado consistente para outro estado consistente, não corrompendo as informações.

3. **Isolamento**: Enquanto uma transação é executada, seus dados devem ser invisíveis para outras transações.

4. **Durabilidade**: Depois de finalizadas, as alterações realizadas pelas operações da transação devem ser permanentes.

5.6 – Níveis de Isolamento

- O padrão SQL define quatro níveis de isolamento para transações, com o intuito de evitar três tipos de problemas em transações simultâneas. Esse problemas são:

1. **Leitura Suja (dirty read)**: Supondo que uma transação “A” tenha modificado um determinado campo de uma tabela, porém ainda não tenha commitado. Se uma transação “B” efetua um select neste campo e consegue visualizar o valor não commitado pela transação “A”, essa é uma leitura suja.

A leitura suja é um problema em ambientes de tomada de decisão, relatórios e consultas, pois caso a transação “A” sofra um Rollback, a transação “B” não terá conhecimento disto, além de já ter informado o valor desatualizado.

5.6 – Níveis de Isolamento

2. **Leitura não repetitiva (Nonrepeatable Read)**: Ocorre quando duas consultas idênticas são executadas dentro da mesma transação e reproduzem resultados diferentes para o mesmo campo.

Exemplo: A transação “A” faz uma leitura do valor de um campo. Em seguida uma transação “B” atualiza este valor e faz o commit. Caso a transação “A” volte a consultar o mesmo campo, ela irá visualizar o valor commitado pela transação “B”.

Desta forma, dentro da transação “A” existirão dois valores diferentes para o mesmo campo, acontecendo uma leitura não repetitiva.

5.6 – Níveis de Isolamento

3. **Leitura Fantasma (Phantom Read)**: Determinada transação “A” faz a leitura de um conjunto de linhas de uma tabela com base em alguma condição where.

Supondo que uma transação “B” insira uma nova linha que também satisfaça a mesma clausula where na tabela utilizada pela transação “A”.

Se a transação “A” for repetida ela verá um fantasma, ou seja, uma linha que não existia na primeira leitura utilizando a mesma clausula where.

5.6 – Níveis de Isolamento

- Para evitar estes problemas, existem os quatro níveis de isolamento para transações que são propostos pelo SQL, são eles:

1. **Read Uncommitted**: Transações podem visualizar os resultados de outras transações que finalizaram suas operações mas ainda não fizeram commit. Ocasiona leitura suja.

2. **Read Committed**: Uma transação visualiza apenas as alterações feitas por outras transações que já foram comitadas. Gera problema de leitura não repetitiva.

3. **Repeatable Read**: Uma transação com esse nível de isolamento também visualiza apenas dados comitados. A diferença do nível anterior com o este é que o Repeatable Read visualiza apenas os dados que já foram comitados até o seu inicio, garantindo que duas consultas iguais sempre tenham o mesmo resultado. Resolve a leitura não repetitiva, mas não a leitura fantasma.

O Read Committed atualiza os dados a leitura, e caso esses dados sejam alterados e comitados por outra transação, irá gerar leitura não repetitiva.

5.6 – Níveis de Isolamento

4. **Serializable**: É o mais alto nível de isolamento, resolve o problema da leitura fantasma forçando as transações a serem ordenadas para que eles não possam se conflitar. Embora segura, compromete a concorrência.

O MySQL utiliza o nível Repeatable Read como padrão, e resolve o problema da leitura fantasma com uma técnica conhecida como controle de concorrência de versões múltiplas (MVCC).

Nível de Isolamento	Leitura Suja	Leitura não Repetitiva	Leitura Fantasma
Read Uncommited	Sim	Sim	Sim
Read Commited	Não	Sim	Sim
Repeatable Read	Não	Não	Sim
Serializable	Não	Não	Não

5.6.1 – Modo AUTOCOMMIT

- O MySQL opera no modo autocommit por padrão, fazendo com que cada comando SQL executado pelo usuário seja alocado em uma transação separada.

- Desta forma, quando um delete é executado por exemplo, ele será automaticamente comitado, sem que o usuário precise executar o comando commit.

- Entretanto, é possível desativar o modo autocommit do MySQL. Para verificar em qual modo ele está operando, basta usar o comando abaixo:

```
SHOW VARIABLES LIKE 'AUTOCOMMIT';
```

```
+---------------+-------+
| Variable_name | Value |
+---------------+-------+
| autocommit    | ON    |
+---------------+-------+
```

5.6.1 – Modo AUTOCOMMIT

- Existem duas formas de desabilitar o autocommit, são eles:

```
//Desabilitar o autocommit temporariamente
SET AUTOCOMMIT=0
//Habilitar o autocommit temporariamente
SET AUTOCOMMIT=1
```

```
//Para desabilitar/habilitar o autocommit de forma permanente, deve
configurar o my.cnf

1. sudo gedit /etc/mysql/mysql.conf.d/mysqld.cnf
2. Insere no arquivo para desabilitar: autocommit=0;
3. Salva e reinicia o MySQL: /etc/init.d/mysql restart;

Obs: Dependendo da versão o arquivo de configuração está em
/etc/mysql/my.cnf;
```

5.6.2 – Trocar Nível de Isolamento

- O MySQL possui como nível de isolamento padrão o nível de isolamento Repeatable Read. Entretanto, é possível trocar este nível de isolamento de acordo com a necessidade do usuário. O quadro abaixo mostra como verificar e trocar o nível de isolamento da seção de um usuário:

```
//Verificar o nível de isolamento
SHOW VARIABLES LIKE '%isolation%';
```

```
+---------------+-----------------+
| Variable_name | Value           |
+---------------+-----------------+
| tx_isolation  | REPEATABLE-READ |
+---------------+-----------------+
```

```
//Trocar o nível de isolamento temporariamente
SET SESSION TRANSACTION ISOLATION LEVEL READ UNCOMMITTED;


//Trocar o nível de isolamento permanente no my.cnf
1. sudo gedit /etc/mysql/mysql.conf.d/mysqld.cnf
2. insere o comando: transaction-isolation = 'READ UNCOMMITED'
```

5.7 – Bloqueio Implícito e Explicito

- Quando uma transação do MySQL precisa efetuar uma uma leitura ou escrita, o próprio SGBD efetua automaticamente os bloqueios necessários. Este bloqueio é implícito.

- Entretanto, o próprio usuário pode bloquear uma ou mais tabelas de forma explícita, por meio dos comandos do quadro abaixo:

```
//Bloqueio de Escrita – tabela não pode ser consultada ou alterada
LOCK TABLE cliente WRITE;

//Bloqueio de Leitura – tabela não pode ser alterada mas pode ser consultada por diversas transações simultaneamente
LOCK TABLE cliente READ;

//Desbloqueia todas as tabelas bloqueadas
UNLOCK TABLES;
```

5.8 – MVCC

- Além do bloqueio em nível de linha, o MySQL utiliza uma técnica para aumentar a concorrência conhecida como **controle de concorrência de versão múltipla (MVCC)**. Esta técnica não é exclusiva do MySQL, sendo que PostgreSQL, Oracle e outros SGBD's também utilizam.

- O MVCC consiste em armazenar versões anteriores das linhas das tabelas, desta forma não é necessário nenhum tipo de bloqueio de leitura, somente de escrita.

- Quando uma transação precisa efetuar uma leitura, mesmo que a tabela esteja bloqueada para escrita, ela poderá ler uma versão anterior da linha, aumentando de forma expressiva a concorrência.

- A desvantagem é que duas transações que fazem a mesma consulta ao mesmo tempo, poderão ter resultados diferentes. Mas trata-se de uma técnica muito segura e de alto desempenho.

5.9 – Storage Engine

- As ferramentas de armazenamento ou storage engine funcionam como plugins no MySQL, e são responsáveis diretamente pelo gerenciamento dos arquivos no disco. Geralmente, as principais engines já vem instaladas no MySQL, mas outras podem ser inseridas.

- Cada engine possui características específicas, sendo indicadas para determinados tipos de aplicações. Atualmente, a engine padrão é a InnoDB, mas a MyISAM, Memory, entre outras também são de grande utilidade.

- Independente da engine utilizada, o MySQL armazena os dados armazenados em uma pasta, onde para cada database haverá uma subpasta com o mesmo nome. O quadro abaixo mostra o endereço do datadir do MySQL:

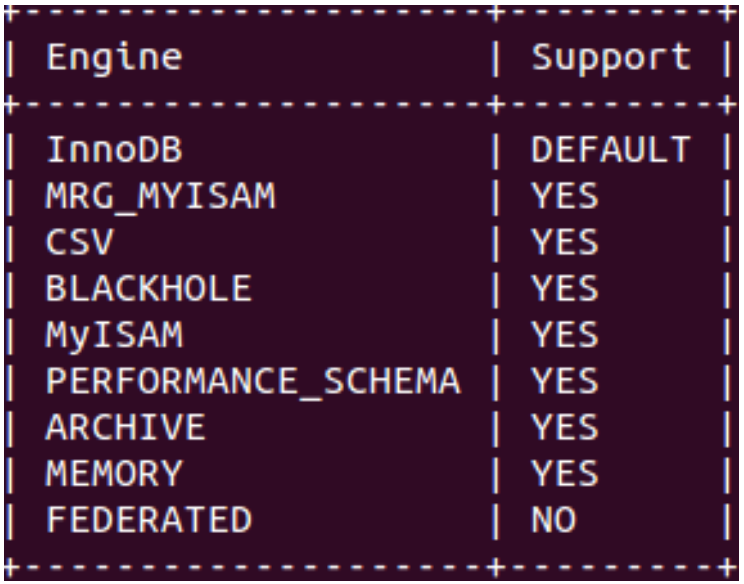
Linux: /var/lib/mysql/

Windows: C:\ProgramData\MySQL\MySQL Server 5.7\

5.9 – Storage Engine

- Por meio do comando SHOW ENGINES, é possível verificar todas as ferramentas de armazenamento que estão ativas no MySQL, como mostra a figura abaixo:

```
SHOW ENGINES;
```

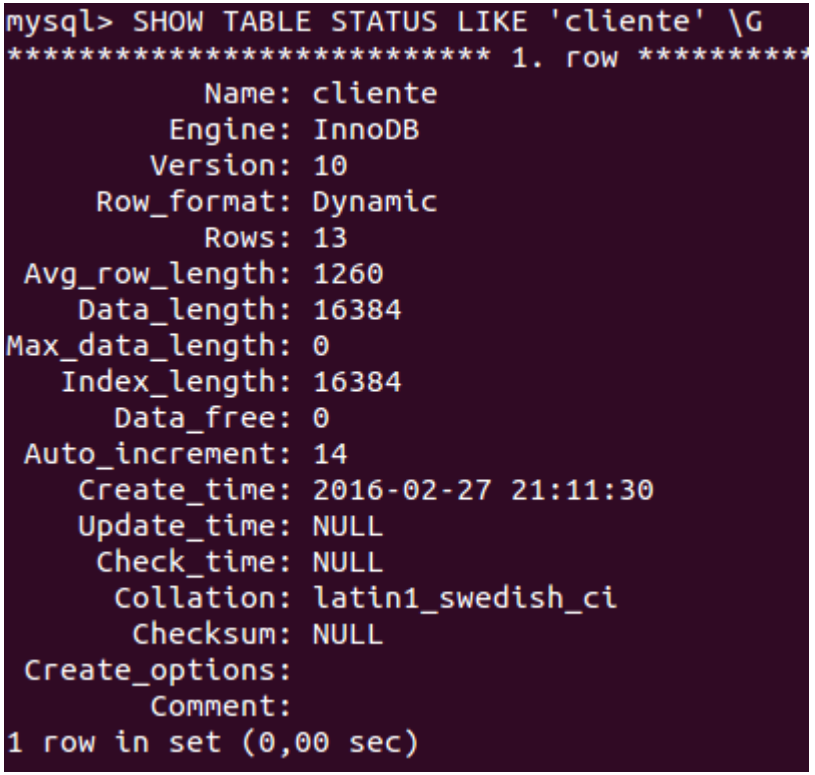


Engine	Support
InnoDB	DEFAULT
MRG_MYISAM	YES
CSV	YES
BLACKHOLE	YES
MyISAM	YES
PERFORMANCE_SCHEMA	YES
ARCHIVE	YES
MEMORY	YES
FEDERATED	NO

5.9.1 – Consultar Características das Tabelas

- Para verificar uma série de características das tabelas, pode-se utilizar o comando do quadro abaixo:

```
SHOW TABLE STATUS LIKE 'cliente' \G
```



5.9.2 – Definir Engine da Tabela

- O MySQL possui uma engine padrão, que pode ser mudada no arquivo de configuração. Mas caso o usuário queira uma engine diferente, pode especificar no momento de criação da tabela por meio do comando "**engine**", como mostra o quadro abaixo:

```
CREATE TABLE `cliente` (
    `id_cliente` int(11) NOT NULL AUTO_INCREMENT,
    `nome` varchar(50) NOT NULL,
    `endereco` varchar(50) DEFAULT NULL,
    `id_cidade` int(11) DEFAULT NULL,
    `telefone` varchar(11) DEFAULT NULL,
    `email` varchar(50) DEFAULT NULL,
     PRIMARY KEY (`id_cliente`)
) ENGINE=InnoDB;
```

5.9.2 – Definir Engine da Tabela

- Para especificar a engine que será padrão no MySQL, deve-se entrar no arquivo de configuração e informar. Inicialmente, a engine padrão é a InnoDB. O quadro abaixo mostra como definir a engine padrão do MySQL como CSV:

```
//Entra no arquivo de configuração do MySQL
gedit /etc/mysql/mysql.conf.d/mysqld.cnf


//Insere o parâmetro com a engine desejada
default-storage-engine='csv'


//Reinicia o MySQL
/etc/init.d/mysql restart
```

5.9.3 – MyISAM

- O MyISAM foi o engine padrão do MySQL até a versão 5.4, perdendo o posto para a InnoDB a partir da versão 5.5.

- É recomendado utilizar o MyISAM para tabelas que possuem um alto número de operações de consultas e uma baixa quantidade de operações de escrita. Como exemplo, pode-se pensar em uma tabela de cidade ou estado por exemplo.

- Porém, não é recomendado utilizar o MyISAM para uma tabela com alta concorrência de leitura e escrita, pois está engine utiliza bloqueio de tabela, reduzindo a concorrência e consequentemente o desempenho do SGBD.

- Outra característica é que o MyISAM não oferece suporte a transações ou a chaves estrangeiras. Este fato põe em risco a integridade de alguns dados, mas em contrapartida garante agilidade.

5.9.3 – MyISAM

- O MyISAM suporta até 256TB de dados por tabela. Não suporta MVCC, transações e nem chav estrangeira.

- A engine MyISAM é otimizada para buscas, possuindo diversas formas de indexação que agilizam as consultas SQL.

- Quando uma tabela MyISAM é criada, também são criados no datadir três arquivos com extensões diferentes, cada um tendo sua funcionalidade. Estas extensões são:

1. **.frm**: Armazena o formato da tabela
2. **.MYD**: Armazena os dados da tabela
3. **.MYI**: Armazena os índices da tabela

5.9.4 – InnoDB

- O InnoDB foi implementado como engine padrão no MySQL a partir da versão 5.5, deixando de ser um plugin como nas versões anteriores.

- InnoDB dá suporte a transações e é totalmente adequado as propriedades ACID. O MVCC é utilizado para otimizar o grau de concorrência;

- Em caso de falhas, o InnoDB possui um sistema de auto_recovery. Além disso, dispões de mecanismos de integridade referencial e de chave estrangeira.

- O InnoDB é ideal para situações que exigem um grande número de escritas e de consultas, diferentemente do MyISAM, que possui desempenho ruim com escritas.

5.9.4 – InnoDB

- Na engine MyISAM, cada tabela possuía três arquivos no datadir. Já o InnoDB funciona de forma diferente.

- Em algumas versões mais antigas do MySQL, por padrão, cada tabela InnoDB possui apenas um arquivo .frm com sua estrutura, mas os dados e índices de todas as tabelas ficam armazenadas em um único arquivo (chamado tablespace).

- É possível configurar este tablespace único para todas as tabelas, definindo um tamanho máximo para ele, ou configurando para que aumente seu tamanho automaticamente caso seja necessário. Para isso, no arquivo de configuração deve ser colocado o parâmetro:

```
//Define que o tablespace tenha máximo de 10M, mas aumente automaticamente caso necessário
innodb_data_file_path=ibdata1:10M:autoextend
```

5.9.4 – InnoDB

- Entretanto, gerenciar um único tablespace para todas as tabelas pode ser um pouco complexo. Desta forma, é possível mudar a configuração para que cada tabela tenha o seu próprio tablespace. As versões mais novas do MySQL já vem com este padrão (versão 5.6.6 e superior).

- Quando se define como padrão um tablespace por tabela, cada tabela criada gera um arquivo “.frm” (estrutura) e “.ibd” (dados e índices) na pasta de dados (datadir).

```
//Este parâmetro define que cada tabela tenha seu próprio tablespace
innodb_file_per_table
```

5.9.4 – InnoDB

- O InnoDB reserva um buffer na memória RAM com o objetivo de manter armazenados os dados das tabelas acessadas recentemente, além dos índices. Isso evita muitos acessos ao disco, aumentando o desempenho do SGBD.

- O parâmetro que define o tamanho da reserva de buffer é o **innodb_buffer_pool_size,** que por padrão é 134217728 (128MB). Pode-se alterar este valor no arquivo de configuração.

```
//Mostra o buffer atual do MySQL
SHOW VARIABLES LIKE 'innodb_buffer_pool_size';


//Altera a configuração do buffer do MySQL para 256MB
1. sudo gedit /etc/mysql/mysql.conf.d/mysqld.cnf
2. innodb_buffer_pool_size=268435456


//Altera a configuração do buffer do MySQL para 256MB temporariamente
SET GLOBAL innodb_buffer_pool_size=268435456;
```

5.9.5 – MEMORY

- Uma tabela com a engine do tipo MEMORY irá armazenar seus dados em memória RAM, o que lhe dará grande desempenho. Entretanto, cada vez que o MySQL for reiniciado, todos os dados serão perdidos, ficando apenas a estrutura da tabela.

- As tabelas deste tipo possuem bloqueio em nível de tabela, não aceitam campos TEXT e BLOB, funciona o auto_increment, mas não existe MVCC e nem chave estrangeira.

- A tabela MEMORY aceitará quantos dados couberem na memória RAM do servidor, mas esse tamanho pode ser limitado com o parâmetro **max_heap_table_size,** que por padrão é 16MB.

```
//Aumentando o limite de dados das tabelas MEMORY para 256MB

1. sudo gedit /etc/mysql/mysql.conf.d/mysqld.cnf

2. max_heap_table_size=268435456



//Altera a configuração para 256MB temporariamente

SET GLOBAL max_heap_table_size=268435456;
```

5.9.6 – Archive

- O Archive pode ser usado para armazenar grandes volumes de dados em um formato compactado no MySQL, ocupando um espaço muito menor do que as demais engines. Tabelas com esse forma aceitam apenas comandos **INSERT** e **SELECT**.

- Como exemplo de utilização do Archive, pode-se utilizar em tabela de log, histórico, contatos, entre outras tabelas que recebem grande quantidade de dados, mas não são utilizadas rotineiramente no sistema.

- Sua estrutura é representada pelo arquivo .frm, mas seus dados são armazenados em um arquivo .ARZ. Não tem suporte a índices, e não funcionam comandos Delete e Update, apens Insert e Select.

- Possui bloqueio de linha, não possui MVCC, não possui chave estrangeira e nem suporte a transações.

5.9.7 – BlackHole

- A engine BlackHole aceita os dados por meio de Insert, mas não armazena os mesmos, ou seja, como resultado de um select em uma tabele deste tipo, o resultado sempre será vazio.

- Apesar das instruções nesta engine não armazenar os dados, caso os logs binários estejam ativados, todas as transações serão gravadas no log, menos aquelas que são submetidas a rollback.

- Com isso, é possível criar um sistema de filtros entre uma replicação master x slaves, principalmente por meio de Triggers (lembrando que funcionará somente com after ou before insert).

- Po ser utilizado também como uma tabela para operações de benchmark, com o objetivo de testar o desempenho do banco de dados.

5.9.8 – CSV

- Quando uma tabela é criada com a engine CSV, é criado um arquivo .frm com sua estrutura no datadir, mas também é criado um arquivo .CSV com os dados armazenados.

- Esses dados estão no formato CSV, separado por vírgulas, sendo possível abri-los em um software de planilha eletrônica, como Excel por exemplo.

- É ideal para armazenamento de logs ou dados que precisarão ser exportados para outro sistema gerenciador de banco de dados, visto que CSV é um padrão suportado por diferentes arquiteturas de banco de dados.

5.9.9 – Outras Engines

- **Merge**: é um mecanismo de armazenamento que permite que um DBA MySQL ou Desenvolvedor possa agrupar logicamente uma série de tabelas MyISAM idênticas e referencia-las como um objeto. Sendo isso bom bancos de dados com grandes volumes de dados.

- **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. Sendo uma opção boa para implementar em ambientes distribuídos ou data marts.

- **Outras**: Existem diversas engines disponíveis para instalação no MySQL, algumas para uso específico, como armazenamento de dados científicos, etc. Inclusive, o próprio usuário pode desenvolver sua própria engine, para que atenda suas necessidades.

5.9.10 – Alterar Engines de Tabelas

- Existem basicamente três formas de alterar a engine de uma tabela, são elas:

```
//Forma 1 - Método funcional, mas pode demorar, pois a troca é feita linha a linha.
ALTER TABLE Clientes ENGINE = MEMORY;


//Forma 2 – Clonar uma tabela, trocar a engine e importar os dados
CREATE TABLE tabela2 LIKE tabela1;
ALTER TABLE tabela2 ENGINE = InnoDB;
INSERT INTO tabela2 SELECT * FROM tabela1;


//Forma 3 – Exportar a tabela com a ferramenta de backup MySQLDump, e importar
trocando manualmente a engine.
```