ADMINISTRAÇÃO DE BANCO DE DADOS

Claudia Abreu Paes

SAGAH SOLUÇÕES EDUCACIONAIS INTEGRADAS

Gerenciamento de espaço de banco de dados

Objetivos de aprendizagem

Ao final deste texto, você deve apresentar os seguintes aprendizados:

- Conceituar *tablespaces* (espaços da base de dados).
- Explicar a relação entre *tablespaces* e *datafiles*.
- Descrever as etapas do gerenciamento de *tablespaces* (espaços da base de dados).

Introdução

A utilização de um banco de dados requer a definição lógica dos dados a serem armazenados. Ela também exige a definição física do local em que os dados serão armazenados. A definição lógica está relacionada à construção dos modelos de dados em que são definidas as estruturas, os relacionamentos e as regras de integridade. A definição física torna-se necessária na medida em que os dados e as suas estruturas serão armazenados em meios tecnológicos.

A definição física estabelece espaços em disco para o armazenamento, definindo os limites de delimitação lógica, os *tablespaces*, que conterão a delimitação física, e os *datafiles*. Os *tablespaces* são formados por segmentos, compostos por blocos onde os dados ficam efetivamente armazenados. Com o uso, o espaço definido pode se tornar pequeno ou fragmentado. Nesse caso, é necessária a gestão de espaço, que conduz ações no sentido de redimensionar a área lógica e física de armazenamento.

Neste capítulo, você vai conhecer os espaços lógicos e físicos delimitados para o armazenamento dos dados. Além disso, vai conhecer as ações necessárias para o gerenciamento do espaço, como criação e redimensionamento.

Tablespaces e *datafiles*

O gerenciamento de espaço de um banco de dados se baseia em uma estrutura própria para o armazenamento dos dados. Tal estrutura é composta por *tablespaces* e *datafiles*. Os *tablespaces* representam o espaço lógico. Dentro do espaço lógico delimitado (*tablespaces*), são criados os *datafiles*, que se referem ao espaço físico que irá conter os objetos. Há algumas regras de criação, como:

- um banco de dados pode conter vários *tablespaces*;
- um *tablespace* pode conter vários *datafiles*;
- um *datafile* contém vários objetos;
- os objetos podem ocupar mais de um *datafile*, mas sempre no mesmo *tablespace*.

Na Figura 1, você pode ver a composição de um *tablespace*.

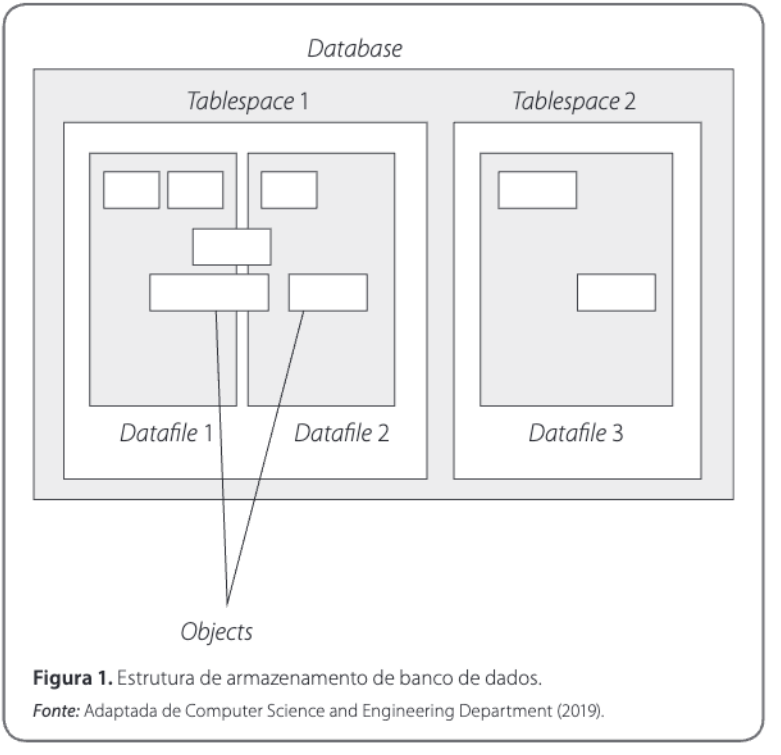

Figura 1. Estrutura de armazenamento de banco de dados.
Fonte: Adaptada de Computer Science and Engineering Department (2019).

A criação do *tablespace* deve ser realizada depois da criação do banco de dados. Afinal, só quando o espaço lógico estiver delimitado será possível a inserção de qualquer objeto, seja de estrutura, dicionário de dados ou dados. Com o espaço lógico definido, é preciso criar os espaços físicos, os *datafiles*, que são criados com o *tablespace*.

O armazenamento dos objetos no *datafile* é efetivado de forma aleatória. Não se garante a sequência dos dados, pois a gravação é realizada de acordo com os espaços vazios. Além disso, mesmo que os espaços sejam contínuos, no caso de banco de dados vazio ou com novas áreas criadas, ou ainda após a reorganização dos dados, a ordem de gravação não dará uma sequência lógica aos dados, simplesmente haverá uma ordem de criação deles. Dessa forma, não se pode determinar a ordem. Sempre que se desejar visualizar os dados ordenados, será preciso utilizar o comando de consulta. É possível definir mais de um *tablespace* no banco de dados, como você pode ver na Figura 2.

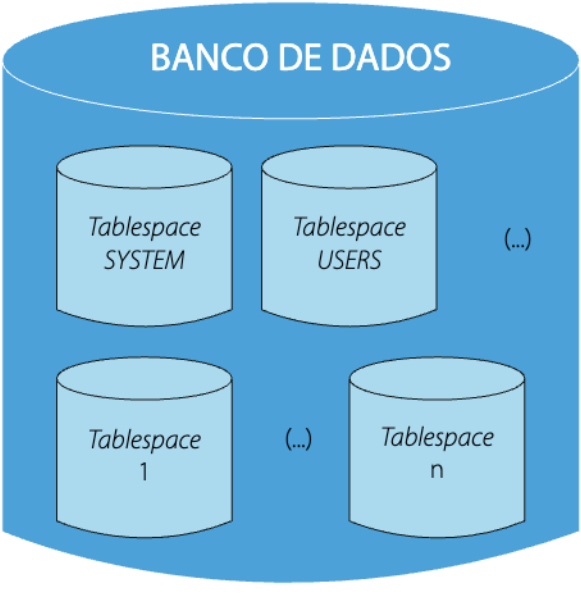

Figura 2. Formação de banco de dados.

A criação de mais de um *tablespace* apresenta algumas vantagens. Veja a seguir.

- Dá flexibilidade à execução de procedimentos, na medida em que se pode determinar a localização em função da frequência de utilização e do volume. Por exemplo: dados utilizados para consultas no nível estratégico. Normalmente, essas são consultas que requerem volume de dados. Dessa forma, é possível acomodar as tabelas envolvidas nessa necessidade em um *tablespace* específico.
- Separa o dicionário de dados dos dados de usuário.
- Utiliza discos diferentes para armazenar os *datafiles*, pois assim há a distribuição do esforço de processamento e o compartilhamento de recursos, como memória e acesso I/O.
- Facilita e organiza melhor a realização dos *backups* por *tablespace*.

Saiba mais

Backup é o procedimento de fazer cópias de segurança. A realização de *backups* é imprescindível para a operação de um ambiente que utiliza bancos de dados, pois representa a segurança dos dados em caso de qualquer ocorrência de erro, seja no disco físico ou devido à má utilização dos dados. Para ter êxito na realização de um *backup*, você deve planejar a execução e a restauração dos dados. O planejamento pode englobar procedimentos de *backup* incremental ou completo.

O *backup* incremental realiza somente cópias dos dados atualizados desde a última cópia, durante um período normalmente de sete dias. Ao final do período, é realizada uma cópia completa para iniciar um novo período. A vantagem do *backup* incremental é o tempo que se gasta para realizá-lo. Como o volume de dados é muito menor, o tempo de realização do *backup* também se torna menor. Em compensação, o processo de restauração, dependendo de quantos volumes de *backup* incremental são utilizados, torna-se mais lento. Por sua vez, o *backup* completo realiza a cópia de todos os dados, o que aumenta o período em que o banco de dados fica fora de atividade. A escolha do procedimento de *backup* varia em função das condições e necessidades do ambiente.

No *link* a seguir, veja a importância da realização dos *backups* para a segurança dos dados.

https://qrgo.page.link/EftJ

A estrutura do *tablespace* ainda utiliza dois outros objetos: **segmento**, que representa o objeto armazenado no *tablespace*, e **extensão**, que é o espaço ocupado pelo segmento. A extensão é formada por blocos para armazenamento. Segundo Tavares (2006), blocos são as menores unidades de armazenamento de um banco de dados. O tamanho é medido em *bytes*. Os segmentos são formados por uma ou mais extensão. Há três tipos de segmentos: dados, índice e temporário. Veja a seguir.

- O segmento de dados representa os dados de uma tabela.
- O segmento de índice representa os índices não particionados. Há também segmentos de índice que armazenam seus dados.
- O segmento temporário é utilizado durante as operações de classificação, por exemplo.

Na Figura 3, você pode ver a estrutura do *tablespace*.

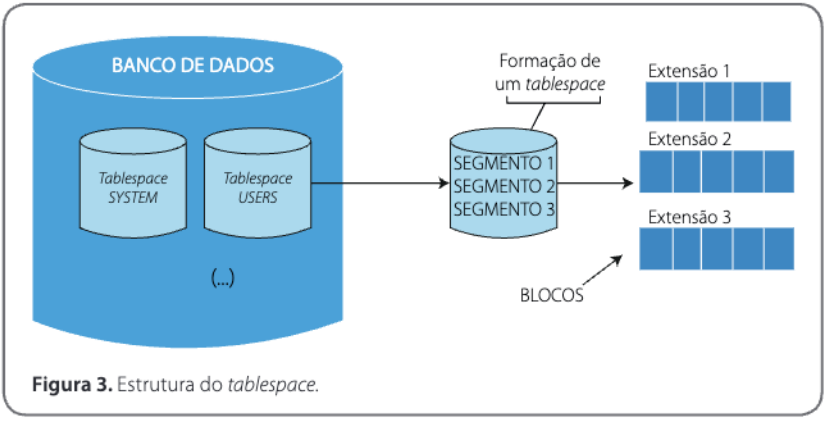

Figura 3. Estrutura do *tablespace*.

Pode haver mais de um *tablespace* no banco de dados. Portanto, em alguns ambientes, os *tablespaces* são denominados de acordo com a necessidade do ambiente e o padrão definido. Pode-se considerar um *tablespace* para cada área da organização, dividindo assim os dados. Nesse caso, haverá um *tablespace* chamado "vendas", outro "compras", outro "*marketing*" e assim por diante.

Os SGBDs definem dois *tablespaces* padrão, SYSTEM e USERS. O *tablespace* SYSTEM possui as seguintes características:

- armazena os dados do dicionário de dados para o seu gerenciamento;
- é criado na criação do banco de dados;
- está sempre aberto se o banco de dados estiver aberto, pois caso contrário o banco de dados parará.

O *tablespace* USERS possui as seguintes características:

- é considerado um *tablespace* padrão;
- armazena os dados de objetos e tabelas automaticamente quando o *tablespace* não for especificado.

No SGBD MySQL, há ainda o *tablespace* TEMPORÁRIA e o *tablespace* UNDO. O *tablespace* TEMPORÁRIA é considerado o menos crítico, por armazenar dados temporários. Em caso de falha, basta que ele seja apagado e recriado; não é necessário restaurar. Sua utilização consiste em armazenar:

- informações temporárias;
- objetos transitórios durante a execução de consultas que utilizam as cláusulas ORDER BY e GROUP BY.
- dados de sessão das tabelas temporárias globais (*global temporary tables*).

Segundo o *Manual de Referência do SGBD MySQL* (2019), é reservado um espaço de tabela temporário da sessão. Esse espaço é recriado sempre que o servidor é iniciado, sendo disponibilizado para uso.

O *tablespace* UNDO armazena informações utilizadas para recuperação, ou seja, guarda a informação anterior à alteração. Dessa forma, se for necessário retornar ao valor anterior, será utilizado o valor armazenado no *tablespace* UNDO. É possível criar vários *tablespaces* UNDO, mas somente um pode estar ativo.

São objetivos dos segmentos UNDO:

- prover a informação para a operação de *rollback*;
- desfazer as alterações não gravadas no banco de dados no processo de recuperação, na abertura do banco de dados e quando o banco de dados é interrompido inesperadamente;

- exercer a consistência de leitura, cumprindo a característica de isolamento, ou seja, as transações em andamento não deverão estar visíveis a outros usuários enquanto não forem finalizadas.

O *tablespace* UNDO tem a limitação de não poder ser somente leitura (*ready-only*), pois a gerência de segmentos e a alocação dos *extends* é realizada automaticamente. Os SGBDs disponibilizam a opção de fazer a gestão manualmente, mas isso não é aconselhável, pois os espaços ficam ocupados após o uso. Veja os passos de gerência:

1. no início da transação, é criado um segmento UNDO;
2. o aumento no volume de dados adiciona novos *extends* ao segmento;
3. a diminuição no volume de dados exclui os segmentos.

Há dois modos de UNDO: ativo e expirado. O modo ativo representa os dados que serão retornados caso seja necessário desfazer a transação. Esses dados não podem ser alterados até o final da transação. Já o modo expirado são os dados que estavam no modo ativo após o encerramento da transação. Eles podem ser alterados.

Gerenciamento de espaço em *tablespaces*

Os espaços ocupados em *tablespaces* se dividem em segmentos, representados nas extensões formadas por blocos. Como você já viu, a criação das áreas de armazenamento não garante a alocação em espaços físicos contínuos, pois depende do tamanho físico disponível para a acomodação do banco de dados.

O gerenciamento de espaços não é uma tarefa muito simples. Como toda gestão, o gerenciamento de espaços deve ser planejado na criação do *tablespace*. É preciso dimensionar o tamanho ideal para os *datafiles*, extensões e blocos e acompanhar a execução para avaliar o nível e a frequência de utilização. O administrador do banco de dados deve sempre manter um percentual de espaço livre, em função da taxa de crescimento ideal para o ambiente de trabalho. Sem isso, pode haver baixo desempenho, inconsistência ou, de forma drástica, a paralização do banco de dados por falta de espaço. Desse modo, você deve ter alguns cuidados na criação e no acompanhamento do *tablespace*.

Criação

As estratégias de gerenciamento de espaços devem ser definidas e planejadas desde a criação do *tablespace*, das extensões, dos blocos e dos objetos. Os parâmetros utilizados na criação devem ser muito bem estudados em função:

- do sistema operacional em que o banco de dados será armazenado, devido ao tratamento dos blocos de dados;
- da dimensão física dos discos disponíveis para acomodar o banco de dados;
- do volume de informações;
- da frequência de acesso.

Analisando a natureza do ambiente, você pode pensar em adotar as seguintes práticas (indicadas no mercado) de criação de *tablespaces*, *datafiles*, extensões e blocos:

- *bigfile tablespace*, que depende da configuração do seu banco de dados e consiste na alocação de espaços grandes para as tabelas, o que permite a utilização da capacidade dos sistemas de 64 bits;
- bom planejamento do tamanho máximo do *tablespace*;
- dados e índices em *tablespaces* distintos;
- gerenciamento de segmentos automáticos;
- modo local de gerenciamento;
- poucos *datafiles* por *tablespace* para otimizar a performance;
- *tablespaces* separados por aplicação para facilidade e flexibilidade nas manutenções, configurações de segurança e *backups*.

O *tablespace* tem dois modos de armazenamento: ***local*** e ***dicionary***. No gerenciamento *local*, é possível especificar um tamanho uniforme para todas as extensões (*uniform*), ou especificar apenas o tamanho da extensão inicial e deixar que o banco de dados determine automaticamente o tamanho de todas as extensões subsequentes (*autoallocate*).

No gerenciamento *dicionary*, são utilizados parâmetros de armazenamento como: NEXT, PCTINCREASE, MINEXTENTS, MAXEXTENTS e DEFAULT_STORAGE. A seguir, veja a especificação dos modos de armazenamento *local* e *dicionary*.

Dicionary

O Database Master (2019) define o *tablespace dicionary* como uma forma centralizada de criação. Extensões são gerenciadas pelo dicionário de dados, em que o mapa de armazenamento é registrado. O servidor atualiza as tabelas apropriadas no dicionário de dados sempre que uma extensão é alocada ou desalocada.

O *dicionary* não pode ser utilizado caso o *tablespace system* tenha sido criado com gerenciamento local. Ele demanda um custo de *overhead* de gerenciamento de espaço, pois são necessárias várias gravações nas tabelas do dicionário de dados ou nos segmentos de *rollback*. Além disso, esse tipo de gerenciamento é determinado no comando de criação do *tablespace* por meio da cláusula `EXTENT MANAGEMENT DICIONARY`.

Todas as informações de extensão são armazenadas no dicionário de dados, em espaços de tabela gerenciados pelo dicionário. A criação de um *tablespace* no modo *dicionary* se dá a partir do comando SQL `CREATE` na seguinte sintaxe:

```
CREATE TABLESPACE nomeTableSpace
    DATAFILE 'endereço físico dataFile' SIZE 100M REUSE
    DEFAULT STORAGE (
        INITIAL 512K
        NEXT 512K
        PCTINCREASE 0
        MINEXTENTS 4
        MAXEXTENTS 4096)
    BLOCKSIZE 4K
    MINIMUM EXTENT 256K
    LOGGING ONLINE
    PERMANENT
    EXTENT MANAGEMENT DICIONARY;
```

As cláusulas no comando `CREATE TABLESPACE` no modo *dicionary* especificam os itens listados a seguir.

- DATAFILE: define o nome do arquivo de dados a ser criado. Pode-se usar as seguintes cláusulas:
  - SIZE — define o tamanho do espaço a ser alocado para o arquivo de dados.

- REUSE — especifica que o espaço do arquivo de dados deve ser reusado quando já existir, sobrescrevendo os dados. Quando não existir, o SGBD irá criar um novo espaço.

- DEFAULT STORAGE: armazena o padrão na criação de novos objetos. Ele é utilizado quando o tamanho não é especificado na criação do objeto. Nenhum parâmetro é obrigatório.
  - INITIAL: tamanho da primeira extensão do objeto (segmento).
  - NEXT: tamanho das próximas e sucessivas extensões do segmento. A medida padrão de definição é o *byte*. Para definir em KB ou MB, você deve fixar as siglas K ou MB, respectivamente.
  - PCTINCREASE: porcentagem de crescimento entre a terceira extensão e as subsequentes. O valor padrão é 50% e o valor mínimo é 0%, que significa mesmo tamanho.
  - MINEXTENTS: mínimo de extensões alocado na criação para o segmento. O valor padrão e mínimo é 1. É possível especificar como UNLIMITED, não estabelecendo um limite.
  - MAXEXTENTS: máximo de extensões alocado na criação para o segmento. É possível especificar como UNLIMITED, não estabelecendo um limite.
- BLOCKSIZE: tamanho do bloco na extensão do segmento. Quando não definido, utiliza-se o tamanho do bloco definido na criação do banco de dados, com o parâmetro DB_BLOCK_SIZE. O tamanho padrão é utilizado no *tablespace system*. Os tamanhos de bloco não padronizados são: 2 KB, 4 KB, 8 KB, 16 KB e 32 KB. Segundo o Computer Science Department (2019), utilizar tamanhos de blocos variados em tabelas grandes favorece as movimentações.
- MINIMUM EXTENT: os tamanhos de extensão são múltiplos do tamanho especificado, utilizado para controlar fragmentação.
- LOGGING: operações de DDL e INSERT de carregamento direto são gravadas no arquivo de *log* REDO. A cláusula pode ser omitida e ainda especificar NOLOGGING. Nesse caso, o arquivo de *log* não será criado, o que implica não ter possibilidades de restauração de dados em caso de perda.
- ONLINE: disponibiliza o espaço criado imediatamente. O uso de OFFLINE implica a criação e não disponibilização para uso, de modo que o espaço precisa ser liberado posteriormente.

- PERMANENT: disponibiliza espaço para uso de objetos permanentes, como tabelas e índices. PERMANENT é o padrão. Quando for usado TEMPORARY, o uso do *tablespace* será disponibilizado para objetos de classificação, temporários durante o processamento.
- EXTENT MANAGEMENT: define o modo de gerenciamento do espaço (*local*/*dictionary*).

Saiba mais

São consideradas como padrão as seguintes quantidades de blocos nos parâmetros.

- INITIAL: cinco blocos quando não declarados e, na declaração, no mínimo três para *tablespaces* gerenciados localmente (sendo dois blocos para segmento manual e um bloco para cada grupo de lista livre no segmento) e dois para dicionário.
- NEXT: cinco blocos quando não declarados e, na declaração, no mínimo um. A especificação de valores menores implica a utilização do valor mínimo.
- PCTINCREASE: o valor padrão é 50% e o valor mínimo é 0%, que significa o mesmo tamanho.
- MINEXTENTS: o valor padrão e mínimo é um. É possível especificar como UNLIMITED, não estabelecendo um limite.
- MAXEXTENTS: o valor mínimo é um e o valor padrão depende do tamanho do bloco do banco de dados.

Local

Extensões são gerenciadas no *tablespace* por *bitmaps* criados no cabeçalho do arquivo, que corresponde a um bloco ou a um grupo de blocos. Quando ocorre liberação ou alocação de uma extensão, o servidor de banco de dados altera os valores do *bitmap* para mostrar o novo *status* dos blocos.

Esse tipo de gerenciamento é determinado no comando de criação do *tablespace* por meio da cláusula `EXTENT MANAGEMENT LOCAL`. Não se define o tamanho das extensões e blocos. O SGBD gerencia automaticamente usando o padrão `AUTOALLOCATE`.

A criação de um *tablespace* no modo *local* se dá a partir do comando SQL `CREATE` na seguinte sintaxe:

```
CREATE TABLESPACE nome tablespace
DATAFILE 'endereço físico datafile' SIZE 300M
EXTENT MANAGEMENT LOCAL UNIFORM SIZE 512K;
```

A cláusula `UNIFORM SIZE` no comando `CREATE TABLESPACE` no modo *local* define o tamanho de todas as extensões. Para o exemplo, todas as extensões serão criadas com o tamanho de 512 KB.

Acompanhamento

Não é suficiente para um administrador definir o melhor modo e o melhor tamanho de criação para o armazenamento de dados. É preciso acompanhar o uso. Afinal, na medida em que as movimentações de inserção, alteração e exclusão acontecem, os espaços são preenchidos e reaproveitados, esgotando a alocação ou fragmentando o disco com a utilização de espaços não contínuos, o que gera uma sobrecarga na execução, prejudicando o desempenho. Dessa forma, o acompanhamento é imprescindível para identificar necessidades como aumentar o espaço físico e reorganizar os dados nos *datafiles*. O tratamento dessas necessidades é importante para melhorar o desempenho e a confiabilidade.

Há três maneiras de aumentar o espaço físico:

- criar um novo *datafile* no *tablespace*;
- aumentar o tamanho de um *datafile*;
- criar um novo *tablespace*.

Você deve ter cuidado na criação dos espaços para obter, sempre que possível, uma área física contínua. Para reorganizar os dados nos *datafiles*, você pode:

- utilizar ferramentas de administração, como o ANALYSE, que verifica a integridade da estrutura de uma tabela, um índice, um *cluster* ou uma visão;

- utilizar ferramentas de reparação, como o REPAIR, para detectar e reparar pequenos dados corrompidos;
- restaurar os dados a partir de um *backup* que fará com que os dados sejam gravados de forma contínua.

Link

Há vários comandos disponíveis para reparação no banco de dados. Caso você tenha interesse em conhecê-los, acesse o *link* a seguir.

https://qrgo.page.link/dYds

Alteração

Um *tablespace* pode ser alterado para modificar os parâmetros de armazenamento padrão, para ficar *on-line* ou *off-line*, para ter acréscimo de outros *datafiles*, para substituir o nome dos *datafiles* existentes e para a realização do *backup*. O tamanho de um *datafile* pode ser alterado automaticamente por meio da opção AUTOEXTEND, ou manualmente por meio do ALTER DATABASE.

Veja um exemplo de alteração automática:

```
ALTER TABLESPACE USERS
    ADD DATAFILE 'users02' SIZE 10M
    AUTOEXTEND ON NEXT 521K MAXSIZE 250M
```

E para desabilitar:

```
ALTER TABLESPACE
    ADD DATAFILE 'users02'
    AUTOEXTEND OFF
```

Agora veja um exemplo de redimensionamento manual:

```
ALTER DATABASE
    DATAFILE 'users02'
    RESIZE 150M
```

Consultas

As informações dos *tablespaces* são armazenadas em tabelas, no mesmo padrão dos demais objetos no modelo relacional. Dessa forma, para obter qualquer informação, você deve utilizar o comando `SELECT`. No SGBD MySQL, a tabela **files** contém as informações sobre os arquivos nos quais os dados estão armazenados.

Link

A tabela **files** disponibiliza também informações sobre as dimensões definidas para alocação de espaço, extensões, tamanho de bloco, etc. Para consultar as informações que essa tabela armazena, acesse o *link* a seguir.

https://qrgo.page.link/sqiV

Por exemplo, para saber qual *tablespace* foi utilizada para criar determinada tabela, utilize a seguinte instrução:

```
select file_name,tablespace_name
  from FILES
 where file_name='nome tabela';
```

Considere a criação da tabela **alunos**:

```
Create table alunos
(cod_aluno        int(11)          primary key,
Nome_aluno        varchar(100)        not null,
Endereco_aluno    varchar(200)        not null,
Telefone_aluno    int (8),
Data_inclusão     date                not null,
Resp_inclusão     varchar(20)         not null,
Data_alteração    date,
Resp_alteração    varchar(20))
Tablespace users;
```

Para visualizar a lista de atributos criados, execute a instrução:

```
SHOW COLUMNS FROM nomeTabela FROM nomeBanco;
SHOW COLUMNS FROM nomeBanco. nomeTabela;
```

Para o exemplo:

```
SHOW COLUMNS FROM ALUNOS;
```

O resultado seria:

Atributo	Tipo	Null?	Chave?	Default	Extra
cod_aluno	Int (11)	no	yes		
Nome_aluno	varchar (100)	no			
Endereco_aluno	varchar (200)	no			
Telefone_aluno	Int (8)				
Data_inclusão	date	no			
Resp_inclusão	varchar (20)	no			
Data_alteração	date				
Resp_alteração	varchar (20)				

Em qual *tablespace* a tabela **alunos** foi criada?

```
select file_name,tablespace_name
  from  files
  where table_name='ALUNOS';
```

A resposta será:

TABLE_NAME	TABLESPACE_NAME
ALUNOS	USERS

Fique atento

Para consultar a forma de gerenciamento de um *tablespace*, utilize o comando:

```
select extent_management
 from table_spaces
 where tablespace_name='SYSTEM';
```

Você vai obter como resposta, por exemplo:

```
EXTENT_MANAGEMENT
----------------
LOCAL
```

Referências

COMPUTER SCIENCE AND ENGINEERING DEPARTMENT. Practive, Managing Tablespace and Data File. *In*: *UPB, Computer Science Department*, 2019. Disponível em: http://andrei.clubcisco.ro/cursuri/5master/abd/5._Managing_Tablespaces_and_Data_Files.pdf. Acesso em: 22 abr. 2019.

FERNANDES, R. R.; FERNANDES, A. P. L. M. Gestão na segurança de dados adotado no Instituto de Tecnologia em Informática e Informação do Estado de Alagoas — ITEC. *In*: SIMPÓSIO DE EXCELÊNCIA EM GESTÃO E TECNOLOGIA, 8., 2011, Resende. *Anais*...

Disponível em: https://www.aedb.br/seget/arquivos/artigos11/32614346.pdf. Acesso em: 22 abr. 2019.

MYSQL. *REPAIR TABLE Syntax*. 2019. Disponível em: https://dev.mysql.com/doc/refman/8.0/en/repair-table.html. Acesso em: 22 abr. 2019.

MYSQL *Temporary Tablespaces*. 2019. Disponível em: https://dev.mysql.com/doc/refman/8.0/en/innodb-temporary-tablespace.html. Acesso em: 22 abr. 2019.

MYSQL. *The INFORMATION_ SCHEMA FILES Table*. 2019. Disponível em: https://dev.mysql.com/doc/refman/5.7/en/files-table.html. Acesso em: 22 abr. 2019.

MYSQL. *Undo Tablespaces*. 2019. Disponível em: https://dev.mysql.com/doc/refman/8.0/en/innodb-undo-tablespaces.html. Acesso em: 22 abr. 2019.

Leituras recomendadas

PUGA, S.; FRANÇA, E.; GOYA, M. *Banco de dados*: implementação em SQL, PL/SQL e Oracle 11g. São Paulo: Pearson, 2013.

RAMAKRISHNAN, R.; GEHRKE, J. *Sistemas de gerenciamento de bancos de dados*. 3. ed. Porto Alegre: McGraw-Hill, 2008.

SILBERSCHATZ, A.; KORTH, H. F.; SUNDARSHAN, S. *Sistema de banco de dados*. Rio de Janeiro: Elsevier, 2012.

Conteúdo:

SAGAH
SOLUÇÕES EDUCACIONAIS INTEGRADAS