BANCO DE DADOS

LINGUAGEM SQL, DEFINIÇÃO E MANIPULAÇÃO DE DADOS

Me. Otacílio José Pereira

INICIAR

introdução

Introdução

Com um servidor de dados disponível e com a estruturação de tabelas e outros elementos obtidos na modelagem, um banco de dados pode ser manipulado. Para esta manipulação, existe uma linguagem própria que chamamos de SQL (*Structured Query Language*), ou Linguagem de Consulta Estruturada.

O foco desta unidade são os princípios da SQL. No início, veremos os fundamentos com os quais ela foi concebida, a Álgebra Relacional. Depois, começaremos com um comando muito importante, o *Select* , para recuperação de dados. Na sequência, veremos os comandos para inserção, alteração e exclusão de dados e, por fim, os comandos para criação de tabelas serão nosso foco.

Enfim, é hora de “brincar” com os dados, e este assunto tem uma visão bem prática.

Aproveite!

Introdução, SQL e Álgebra Relacional

Durante a implementação de um banco de dados, de início, a modelagem é feita e, a partir daí, é necessário criar o banco de dados em si em um sistema gerenciador de banco de dados (SGBD). A forma como as tabelas foram estruturadas e os diversos outros elementos identificados na modelagem, chaves primárias, chaves estrangeiras, tipos de dados e outros, tudo isso deve ser instalado para que os usuários possam enfim usar o banco de dados para abrigar seus registros.

A SQL (*Structured Query Language*) é uma linguagem própria para o trato com os dados e pode ser empregada de forma bem variada ao operar com os componentes em um banco de dados. Ela é dividida em grupos ou subconjuntos de comandos, entre os quais dois deles serão focados neste nosso estudo, a DDL e a DML. A DDL, que no português seria linguagem para definição de dados (no inglês, *Data Definition Language*), contém comandos para operações como criação, alteração e exclusão de tabelas, alterações em campos, manipulação em índices, chaves e outros. Enfim serve para indicar como os dados estão estruturados no banco de dados. A DML, linguagem de manipulação de dados (do inglês, *Data Manipulation Language*), contém os comandos para operar com os registros das tabelas, por exemplo, para recuperar, inserir, alterar ou excluir dados em uma tabela de clientes no banco de uma loja virtual. Para concretizarmos melhor, o Quadro 3.1 apresenta alguns exemplos de comandos de DML, o primeiro para inserção de um registro na tabela de CLIENTES e o segundo recuperar NOME e CONTATO da tabela de CLIENTES.

Exemplos de comando de DML

- Comando para inserção de um novo registro em uma tabela:

```
INSERT INTO CLIENTES (CODIGO, NOME, CONTATO)
VALUES (10, 'José Silva', '71 977123412')
```

- Comando para consulta ou recuperação de registros em uma tabela:

```
SELECT NOME, CONTATO
FROM CLIENTES
```

Quadro 3.1 – Exemplos de comando DML
Fonte: Própria.

Operações sobre uma Tabela

Para termos uma compreensão mais interessante do que os comandos de SQL fazem, vamos pensar um pouco antes. Imagine que você está diante da necessidade de organizar os registros de clientes e para isso você usa uma tabela, um elemento chave em um banco de dados. Esta tabela teria um formato análogo ao mostrado na Figura 3.1 a seguir. Quais operações você imagina que poderia fazer sobre estes dados? Um determinado usuário poderia recuperar os clientes com idade inferior a 40, daí apenas as linhas em amarelo seriam recuperadas. Caso outro usuário queira focar apenas em nome e cidade dos clientes, apenas as colunas em verde seriam recuperadas. Enfim, em uma visão bem simplificada, as operações sobre os dados terão de dar conta de como tratar as colunas e as linhas de uma tabela, considerando-se que uma tabela é um dos elementos centrais de um banco de dados.

Cod	Nome	Sexo	Idade	Cidade
2	Ademar	M	22	São Paulo
1	Fabrício	M	41	Salvador
5	Joelma	F	60	Rio de Janeiro
4	Carlos	M	21	São Paulo
3	Adriana	F	30	Curitiba

Quadro 3.2 – Operações típicas em um exemplo de tabela de CLIENTES
Fonte:Elaborado pelo autor.

Álgebra Relacional

Um banco de dados relacional tem este nome, pois, conforme Elmasri (2011), **relação** é o bloco básico na qual este tipo de banco de dados funciona. A álgebra relacional, conforme Puga (2013), é uma linguagem formal de alto nível para expressar as operações sobre tabelas, suas linhas e colunas. Ela engloba operações de conjuntos por exemplo: união, interseção, produto cartesiano e diferença de conjuntos; e operações chamadas relacionais de projeção, seleção, junção e divisão.

A Álgebra Relacional é a fundamentação que permite entender como a linguagem SQL foi concebida, o que nos dá maior propriedade para compreendermos os comandos de SQL. Veremos, dessa maneira, que envolverá bastante do que discutimos anteriormente, sobre o trato com linhas e colunas de tabelas.

reflita

Reflita

A Álgebra Relacional é um conceito matemático com boa fundamentação teórica. A partir daí, podemos fazer uma reflexão, por que é importante termos uma boa base teórica, ou de conceitos, conciliada com as habilidades práticas? Você acha a teoria ou o aprendizado de conceitos importante? Bom, vale você fazer sua reflexão. Mas permita-nos comentar que muitas das inovações disruptivas ocorrem quando as mudanças acontecem nos conceitos. Na área de Banco de Dados, a *International Business Machines Corporation* (IBM), a partir do artigo de Codd (1970), propiciou a criação de toda uma teoria e base para tecnologia de bancos de dados relacionais e que tem sido pilar do Sistema de Informação. Passados 50 anos, estamos estudando essa tecnologia que se baseou nesta fundamentação teórica. Claro que é melhor já aprendermos a aplicação dela nos bancos de dados e na linguagem SQL, mas vale ficarmos atentos a quais teorias estão sendo criadas hoje para acompanharmos o "trem das inovações". Faça suas reflexões!

Fonte: Codd (1970).

Operação de SELEÇÃO (σ) para filtrar linhas

Vamos então compreender os comandos de álgebra relacional. O primeiro deles é o de Seleção, que visa filtrar linhas da tabela conforme uma condição. Lembrando nosso exemplo anterior de selecionar os clientes com idade inferior a 40, a escrita em álgebra relacional está exposta a seguir. A letra sigma (σ) é o símbolo que representa a operação de seleção. O resultado desta

operação são as linhas amarelas já apresentadas na Figura 1.1.

```
σ idade<40  (CLIENTE)
```

Quando falamos de comandos no mundo da computação, é comum estabelecermos a sintaxe, isto é, a regra com que um comando deve ser escrito. Para este comando de seleção em álgebra relacional, a sintaxe é a exposta a seguir e indica que o comando é escrito com o símbolo σ seguido da condição e depois da relação a qual a operação se aplica.

```
σ <condição>  (<Relação>)
```

Essa operação de seleção pode ter condições combinadas com operadores lógicos como os operadores E e OU. Por exemplo, a sentença abaixo fará a seleção de linhas de pessoas com idade acima de 40 anos e que moram no Morumbi. O operador E escreve-se com "^" e o operador ou se escreve com "ˇ " ("circunflexo invertido").

```
σ ((idade > 40) ^ (Cidade= 'Salvador' )) (CLIENTE)
```

Operação de Projeção (π) para escolher colunas

Assim como é possível escolher as linhas, a operação de projeção permite escolher as colunas de uma determinada relação ou tabela e é representada pela letra grega Pi (π) . Também com base em nosso exemplo, para escolher as colunas de Nome e Cidade na tabela de clientes, o comando está exposto a seguir. O resultado serão as colunas destacadas em verde na tabela de CLIENTE.

```
π NOME, CIDADE (CLIENTE)
```

Enquanto comando, a sintaxe da projeção é:

```
π <lista de atributos>  (<Relação>)
```

Combinação de operações

A álgebra relacional permite a combinação de operações, isto é, o resultado para uma determinada consulta é obtido pela execução de operações combinadas por meio de parênteses. No exemplo a seguir, primeiro é feita a operação de seleção (σ) para filtrar as linhas

de CLIENTE. Depois é realizada a de projeção (π) para escolher as colunas que devem ser apresentadas.

```
π NOME, CIDADE ( σ idade<40  (CLIENTES))
```

Ao aplicar estas operações sobre a tabela mostrada em nosso exemplo CLIENTES, o resultado será o mostrado no Quadro 3.3 .

Nome	Cidade
Ademar	São Paulo
Carlos	São Paulo
Adriana	Curitiba

Quadro 3.3 – Resultado da combinação de operações de álgebra relacional
Fonte: Própria.

Alguns livros, como em Puga (2013), a notação está registrada conforme o que foi escrito até aqui, mas poderá ser encontrada também a escrita de comandos em álgebra relacional nos quais as condições ou campos das diversas operações são escritas em subscrito como a seguir.

```
π NOME, CIDADE ( σ idade<40  (CLIENTES))
```

Outras operações

Além dessas operações básicas, a Álgebra Relacional dispõe de diversas outras com variadas finalidades no trato com as relações. Puga (2013) e Elmasri (2011) apresentam detalhes destas operações. A Figura 3.2a seguir é uma adaptação de Puga (2013) com algumas outras operações relevantes de álgebra relacional presentes. Perceba que há muitos aspectos em comum com conceitos matemáticos, de teoria de conjuntos e relações. Esta é a base na qual os comandos de SQL foram concebidos.

Símbolo	Significado	Finalidade	Sintaxe
σ (sigma)	Seleção, restrição ou select	Extrair linhas que atendam a uma determinada condição	σ <condição de seleção> (R)
π (pi)	Projeção ou project	Extrair colunas específicas.	π <lista de atributos> (R)
x	Produto cartesiano ou cartesian product	Combinar cada linha de uma tabela, com cada linha de outra tabela.	(R) x (R)
⋈	Junção natural ou natural join	Extrair linhas de duas ou mais tabelas com nome e tipo de dados iguais.	(R) ⋈ (R)
∪	União ou union	Extrair e combinar todas as linhas de uma ou mais consultas compatíveis	(R) ∪ (R)
∩	Intersecção ou intersect	Extrair apenas as linhas presentes em duas tabelas.	(R) ∩ (R)
-	Diferença ou minus	Extrair todas as linhas da primeira tabela, que não estão presentes na segunda tabela.	(R) - (R)

Quadro 3.2 – Operações típicas em um exemplo de tabela de CLIENTES
Fonte:Elaborado pelo autor.

saiba mais

Saiba mais

A compreensão de Álgebra Relacional, neste texto, permite uma introdução a sua forma de tratar relações ou tabelas e nos permite perceber quais os fundamentos base da concepção da SQL. Na prática cotidiana com banco de dados, uma boa competência com SQL já é interessante, mas caso queira saber mais sobre Álgebra Relacional, uma leitura mais aprofundada pode ser encontrada em livros mais densos sobre bancos de dados, como o capítulo 3 de Puga (2013) e o capítulo 6 de

Elmasri (2011), que se encontram disponíveis no link.

Fonte: Puga (2013, p. 53); Elmasri (2011, p. 96).

ACESSAR

praticar

Vamos Praticar

A álgebra relacional permite realizar operações como seleção, projeção e outras aplicadas às relações, que são equivalentes às tabelas. Este assunto é importante para a área de banco de dados, pois serve como fundamento para a manipulação de dados. Imagine que uma relação apresente os seguintes valores mostrados no quadro a seguir e que foi aplicada a seguinte operação de álgebra relacional:

```
π NOME, CIDADE (σ (sexo='F')^(bairro='Barra')(CLIENTES))
```

CODIGO	NOME	CPF	CONTATO	DATA_NASC	SEXO	BAIRRO
1	Adriana Araújo	01120389921	71 982213455	1987-02-03	F	Barra
2	Renato Nogueira	98220379931	11 933321999	1977-07-09	M	Morumbi
3	Viviane Sales	43517386521	11 987712022	1995-11-02	F	Vila Madalena
4	Marcela Campos	99220377221	71 973514498	1980-01-19	F	Barra
5	Rodrigo Gonçalves	72177589991	21 986121942	1992-05-10	M	Centro
6	Jorge Marinho	83144619976	11 995439812	1990-06-07	M	Morumbi
7	Rodrigo Vieira	73520421929	71 972318872	1985-08-27	M	Centro
8	Vanessa Aquino	76211038739	21 933211346	1972-02-15	F	Centro
9	Fabiana Moreira	31160319971	11 933216758	1985-04-13	F	Vila Madalena
10	Maria Conceição	82410399921	21 954419257	1999-07-10	F	Centro

Quadro 3.2 – Operações típicas em um exemplo de tabela de CLIENTES
Fonte:Elaborado pelo autor.

O resultado da operação aplicada a esse conjunto de dados terá

- **a)** Três linhas e duas colunas.
- **b)** Três linhas e três colunas.
- **c)** Quatro linhas e duas colunas.
- **d)** Duas linhas e duas colunas.
- **e)** Duas linhas e três colunas.

Data Definition Language (DDL)

Aprenderemos agora sobre os grupos de comandos divididos em três ênfases: os comandos de definição de dados (DDL) como o *CREATE TABLE* , a recuperação via *SELECT* e, por fim, inserção, alteração e exclusão via comandos como *INSERT* por exemplo.

Existem duas sequências que podem ser empregadas. Uma das abordagens envolve seguir a ordem natural com que os comandos acontecem no ciclo de vida de um banco que envolve modelagem, criação, inserção dos registros e depois recuperação e outras manipulações. Seguindo esta linha, o início ocorre com os comandos de DDL para criar a estrutura do banco, depois as manipulações, como inserção e outros, e somente no fim a operação via *SELECT* .

Outra forma de abordagem consiste em começarmos pelos comandos mais simples, irmos nos ambientando com o jeito e a lógica de pensar os comandos e daí ir incorporando os outros comandos. Nesta linha, o início seria realizado com o comando mais simples, o *SELECT* , para recuperação dos dados. Depois pode-se então tratar as manipulações e como as tabelas foram criadas.

Nesta seção, seguiremos uma trajetória que envolve a criação das tabelas, depois o comando *SELECT* e, por fim, as manipulações. É uma sequência comum de se encontrar em planos de ensino, mas que pode ser alterada conforme a escolha do leitor; fique à vontade.

Preparando o ambiente MySQL e MySQL Workbench

No decorrer do curso, utilizaremos como servidor de banco de dados o MySQL, um servidor “ *open source* ” e popular, o que torna possível sua instalação e o acesso a um vasto material de suporte. A operação com o banco de dados pode ser feita via linha de comandos (*prompt*) ou via ferramentas administrativas; o MySQL provê as duas formas.

O MySQL Workbench é a ferramenta administrativa disponibilizada pelo fornecedor e sua visão típica está representada pela Figura 3.3. Na parte superior, estão o menu e a barra de tarefas ou comandos. À esquerda está a parte “Navegador”, interessante por permitir visualizar os diversos objetos presentes no servidor, sobretudo os bancos de dados para o nosso caso. Na parte central, o usuário pode realizar as tarefas e, no nosso caso, estamos com consultas abertas para digitar os comandos. Os resultados da execução dos comandos, por sua vez, podem ser visualizados na parte inferior.

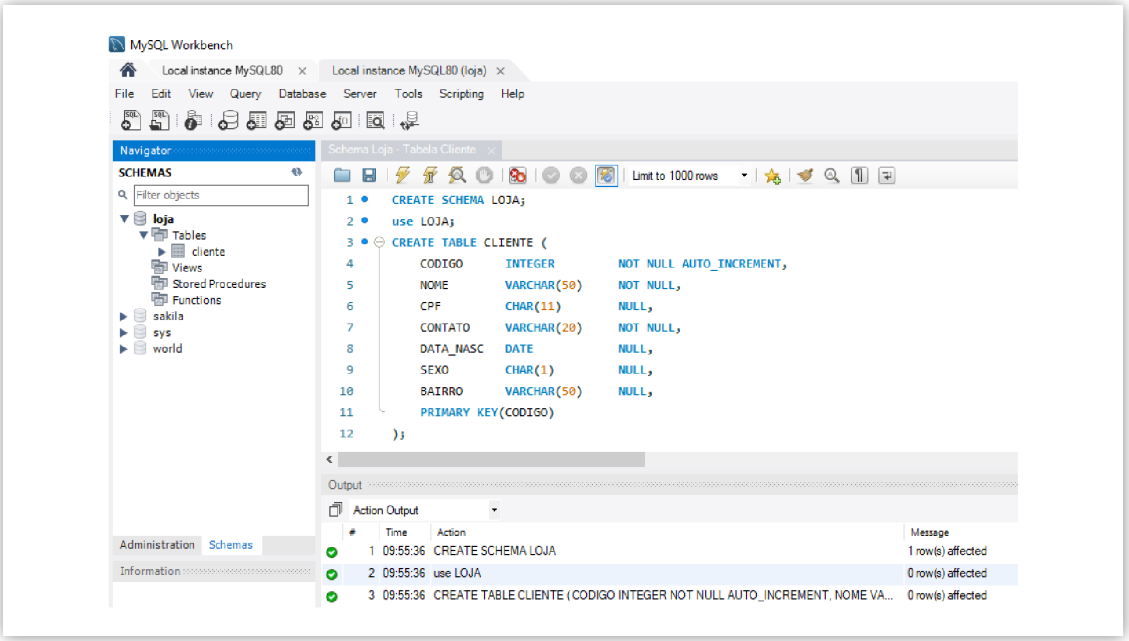


Figura 3.1 – Ferramenta administrativa MySQL Workbench
Fonte: Elaborada pelo autor.

Apesar de ter usado o WorkBench, uma ferramenta específica de um servidor, vale destacar que esta é uma visão geral comum em diversas ferramentas administrativas de outros bancos. Elas em geral apresenta uma parte superior para disparar tarefas por menu e barra de comandos, uma parte lateral esquerda para navegação pelos objetos e a parte central para a operação com o servidor em si.

saiba mais

Saiba mais

Ferramentas administrativas para bancos de dados, como o Workbench para o MySQL, em geral apresentam funcionalidades comuns: visualizar os objetos no banco, criar consultas, ver resultados. Caso você queira comprovar, analise alguma outra de algum outro banco de sua preferência. Por exemplo, no SQL Server existe o SQL Server Management Studio. Experimente observar (não precisa instalar, apenas ler algo sobre a ferramenta), ler um tutorial e veja se de fato apresentam funções análogas. Claro que os bancos serão diferentes, existirão diferenças, mas perceba que algumas tarefas são

equivalentes. A propósito, se não se mobilizou ainda, é uma boa hora para instalar o servidor e o Workbench. Para saber mais, acesse o *link* disponível.

ACESSAR

A visão da ferramenta na Figura 3.3 mostra também informações dos elementos que serão usados como ambiente para nosso aprendizado. O script, conjunto de comandos no centro, cria um *schema* "Loja" e, na sequência, a tabela CLIENTE, que será explicada nos tópicos a frente. Perceba que o *schema* está expandido na parte de navegação com a tabela CLIENTE em questão.

Este é um bom momento para entendermos *schemas* . No decorrer da unidade, sempre falamos de bancos de dados como um componente abrigado em um servidor onde estão as tabelas e outros elementos. Podemos dizer que este conceito de banco de dados corresponde ao de *schema* em nosso ambiente, é nele que estarão "pendurados" os elementos de banco de dados do nosso sistema "Loja".

Uma vez com o ambiente de trabalho compreendido, podemos então entender a sua operação via comandos.

Criação de tabelas (*CREATE TABLE*)

Durante a modelagem, as tabelas foram definidas com os seus atributos como visto na Figura 3.4 para a tabela CLIENTE. A propósito, esta será a tabela que exploraremos como exemplo base para nosso aprendizado; é simples e didática. Entretanto fica o convite, sempre que um novo comando ou operação for explicado, que o aluno, por analogia, experimente fazer em um exemplo próprio, imaginando outras tabelas.

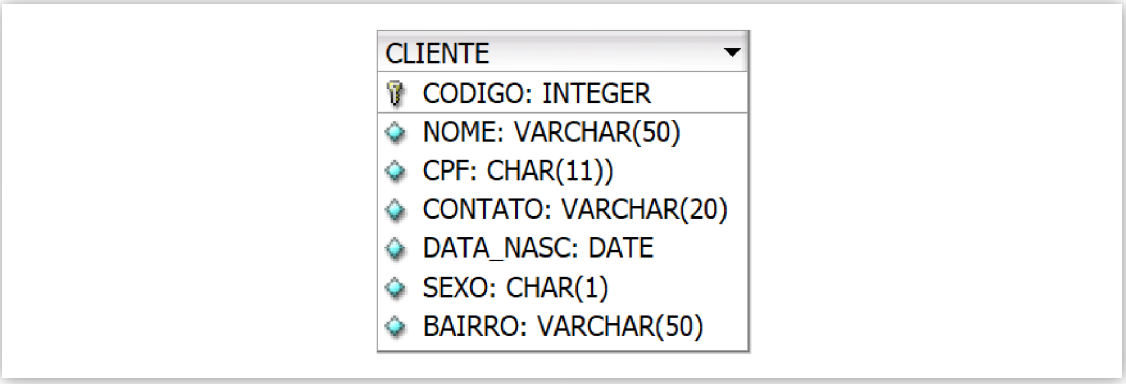


Figura 3.2 – Tabela (CLIENTE) exemplo para aprendizado do SQL
Fonte: Própria.

Como você aprende? Para as diferentes disciplinas, qual a melhor forma de aprender? E para este aspecto de modelagem e de programar consultas em SQL, como melhor usufruir das sessões de aprendizagem?

Este autoconhecimento sobre como você domina os assuntos que estuda é importante. Para este assunto em específico, de forte apelo prático, pois envolve programação, é interessante você instalar as ferramentas e pensar um projeto próprio em paralelo. E daí, verifique e reflita se isso de fato ajuda a aprender melhor este conteúdo. Se quiser saber mais, existem algumas metodologias que tratam disso, uma delas é a aprendizagem baseada em projetos. A fonte a seguir comenta sobre essa metodologia.

Fonte: Porvir (2019).

Para criar esta tabela em um banco de dados, usa-se o comando *CREATE TABLE* e, para o nosso caso, ele será operado da seguinte forma (Quadro 2.1).

Exemplo de comando: CREATE TABLE

```
CREATE TABLE CLIENTE (
CODIGO      INTEGER       NOT NULL AUTO_INCREMENT,
NOME        VARCHAR(50)   NOT NULL,
CPF         CHAR(11)      NULL,
CONTATO     VARCHAR(20)   NOT NULL,
DATA_NASC   DATE          NULL,
SEXO        CHAR(1)       NULL,
BAIRRO      VARCHAR(50)   NULL,
PRIMARY KEY(CODIGO)
)
```

Quadro 3.4 – Exemplos do comando CREATE TABLE
Fonte: Própria.

Vale percebermos que cada campo está associado ao seu tipo de dados, isto é, quais valores que são possíveis de serem atribuídos aos campos. Por exemplo, o campo CODIGO só pode ter valores inteiros. Nome e CPF podem armazenar caracteres. No caso de Nome, o tipo VARCHAR permite que o espaço de valores que não usarem todos os 50 caracteres seja melhor gerenciado internamente. Já para o caso CPF, sempre 11 caracteres serão reservados para o dado. E assim, outros campos possuem seus domínios de valores.

Cada campo também possui o indicativo de *NULL / NOT NULL* , que indica se o campo deve ter obrigatoriamente um valor ou não, isto é, *NULL* (sem valor). Vemos que para cadastrar um cliente é necessário ter pelo menos os campos Código, Nome e Contato.

Em algumas situações, pode ser conveniente que um determinado campo tenha seus valores obtidos incrementados de forma automática; o banco de dados gerencia esta geração automática. Neste caso, usa-se a sentença " *CODIGO INTEGER NOT NULL AUTO_INCREMENT* " com a palavra " *AUTO_INCREMENT* ".

Ao final do comando *CREATE TABLE* , percebe-se que foi criado um comando para adicionar a chave primária que é o campo CODIGO. Isso é interessante, pois permite compreender que uma tabela não é apenas um conjunto de campos, ela possui um conjunto de restrições sobre as tabelas, dentre as principais, as chaves primárias e estrangeiras.

Neste momento, vale sinalizar que a sintaxe que será usada no texto será baseada no banco MySQL, usado como exemplo no decorrer da unidade. Cada banco de dados apresenta algumas variações em torno de um padrão, o chamado padrão ANSI SQL. Por isso, dependendo do SGBD que você usar, MySQL, ORACLE, SQL Server, além de outros, vale conhecer o seu guia de referência dos comandos para tratar de eventuais adaptações.

reflita

Reflita

Pare um pouco e pense. Você gosta de seguir padrões? Por que você acha que o SQL teve sucesso em sua jornada? Definição de padrões é uma estratégia importante em tecnologia e em outras áreas também, porque permite melhor integração de diversos fabricantes e tecnologia. Por exemplo, existem vários servidores de bancos de dados, entretanto, se um profissional domina o SQL, o esforço dele para se adaptar a algum servidor específico é menor, uma vez que o servidor "obedece àquele padrão" que um conjunto ou consórcio de empresas estabeleceu.

A propósito, para o SQL existe o ANSI SQL, padrão gerenciado pela *American National Standards Institute* (ANSI). Os servidores possuem algumas peculiaridades, mas mantêm boa compatibilidade com o padrão. Portanto, vale refletir, qual o papel dos padrões em Tecnologia de Informação? Quais vantagens e desvantagens em usá-los?

Todo comando tem uma sintaxe formal, isto é, uma especificação mais formal sobre como escrever o comando. Apesar de ser mais fácil aprendermos pelo exemplo, a cada comando maior, apresentaremos uma visão breve da sintaxe. No caso do CREATE TABLE, uma breve sintaxe está representada no Quadro 3.5. Perceba que para criar uma tabela é necessário ter o Nome_Tabela e a especificação das colunas e das restrições. Para uma coluna, é necessário especificar o *Nome da Coluna* , o *Tipo de Dados* e a obrigatoriedade (*NULL, NOT NULL*).

Sintaxe de comando: CREATE TABLE

CREATE TABLE *Nome_Tabela* (
< *Lista_Colunas*> ,

< *Restrições*>
)

Coluna : < *Nome_Coluna Tipo_dado* [*NULL/NOT NULL*] >

Restrição : < *Tipo_Restrição* > < *Especificação* >

Quadro 3.5 – Sintaxe de comando CREATE TABLE
Fonte: Elaborado pelo autor.

Alteração e Exclusão de tabelas

Assim como uma tabela pode ser criada, ela pode ser modificada e excluída. Na verdade, estas operações de criar, alterar e excluir são típicas em componentes de banco de dados. O Quadro 3.6 mostra os exemplos destes comandos. No primeiro, de alteração da tabela, será adicionado um campo COD_CIDADE do tipo inteiro. O segundo exclui uma coluna DATA_NASC da tabela e, por último, o terceiro comando exclui a tabela. Importante alertar que estes dois últimos devem ser usados com muito cuidado, pois podem ocasionar perda de dados.

Exemplos de Comando: ALTER TABLE / DROP TABLE

ALTER TABLE CLIENTE
ADD COLUMN COD_CIDADE INTEGER NULL

ALTER TABLE CLIENTE DROP COLUMN DATA_NASC

DROP TABLE CLIENTE

Quadro 3.6 – Exemplos para alteração e exclusão de tabela
Fonte: Elaborado pelo autor.

Restrições de Chave Primária e Chave Estrangeira

Uma tabela, além dos campos, possui as restrições que estabelecem algumas regras para os valores que são atribuídos aos campos. As chaves primárias e chaves estrangeiras são dois

exemplos destas restrições. As chaves primárias identificam um certo registro e as chaves estrangeiras são campos que só podem ser preenchidos com valores que existem em uma outra tabela referenciada.

Assim como no caso de adicionar colunas, para restrições é também usado o comando *ALTER TABLE* com a especificação das restrições. O Quadro 3.5 mostra exemplos do comando para manipular as chaves primárias. Vale lembrar que a chave primária pode ser criada dentro do próprio comando *CREATE TABLE* como já foi visto, além de existir diversas outras formas. É necessário indicar qual tabela está sendo alterada, o tipo de restrição, que é a *PRIMARY KEY* a ser adicionada, e a especificação da restrição, que, no caso, é o campo. No primeiro comando, dá-se um nome para esta restrição, o PK_CLIENTE, e 'PK', de *primary key* , chave primária em inglês. No segundo comando, usa-se também um formato por meio do *CONSTRAINT* , indicando que é uma restrição; os dois comandos são equivalentes, são só formas diferentes de escrever. E o último comando é um exemplo para a exclusão da chave.

Exemplos de Comando: Chave Primária

```
ALTER TABLE       CLIENTE
ADD PRIMARY KEY PK_CLIENTE (CODIGO)

ALTER TABLE              CLIENTE
ADD CONSTRAINT       PK_CLIENTE
PRIMARY KEY              (CODIGO)

ALTER TABLE CLIENTE DROP PRIMARY KEY
```

Quadro 3.7 – Exemplos de comandos para chaves primárias
Fonte: Elaborado pelo autor.

Para as chaves estrangeiras, os exemplos estão mostrados no Quadro 3.7 . Perceba que basta indicar qual o campo é chave estrangeira e qual chave primária de outra tabela ela referencia.

Exemplos de Comando: Chave Estrangeira

```
ALTER TABLE              CLIENTE
ADD CONSTRAINT       FK_CLIENTE_CIDADE
FOREIGN KEY              (COD_CIDADE) REFERENCES CIDADE(COD_CIDADE)

ALTER TABLE              CLIENTE
DROP FOREIGN KEY     FK_CLIENTE_CIDADE
```

Quadro 3.8 – Exemplos de comandos para manipular chave estrangeira
Fonte: Elaborado pelo autor.

praticar

Vamos Praticar

Os comandos de DDL (*Data Definition Language*) permitem estabelecer como os dados serão estruturados em um banco de dados e, além disso, permite também definir algumas regras sobre como estes dados são mantidos. Um determinado comando foi usado em um banco de dados para criar uma tabela de ALUNO.

```
CREATE TABLE ALUNO (
CODIGO        INTEGER          NOT NULL,
NOME          VARCHAR(50)      NOT NULL,
MATRICULA     CHAR(11)         NOT NULL,
CONTATO       VARCHAR(20)      NULL,
PRIMARY KEY(CODIGO));
```

A respeito dos dados que podem ser mantidos nesta tabela de ALUNO, é CORRETO o que se afirma em:

- **a)** Cada campo NOME ocupará 50 caracteres no banco de dados.
- **b)** Um aluno pode ser cadastrado sem telefone (CONTATO) e depois o seu telefone ser atualizado.
- **c)** Dois alunos podem ter o mesmo código.
- **d)** O campo CODIGO pode ser formado por uma letra e números.
- **e)** Um aluno pode não ter matrícula.

Manipulação de Dados (DML) - Comando SELECT

Um comando muito importante na manipulação de dados é o comando *Select* , pois com ele conseguimos recuperar as informações que estão contidas nas tabelas e com muitos recursos para especificar e sofisiticar como querecemos esta recuperação. Podem ser usados os filtros de diversas formas, ordenação, colunas podem ser escolhidas ou abstraídas, consultas podem conter subconsultas, enfim, isso é só para comentar que o *SELECT* é um comando muito versátil.

Neste momento, veremos as formas mais básicas de realizar as consultas com o Select. Uma vez com as tabelas criadas no banco de dados, o foco agora é consultar seus dados.

Dados de exemplos

Para que possamos compreender o comando, vamos considerar a tabela CLIENTE que apresenta os seguintes registros do Quadro 3.9.

CODIGO	NOME	CPF	CONTATO	DATA_NASC	SEXO	BAIRRO
1	Adriana Araújo	01120389921	71 982213455	1987-02-03	F	Barra
2	Renato Nogueira	98220379931	11 933321999	1977-07-09	M	Morumbi
3	Viviane Sales	43517386521	11 987712022	1995-11-02	F	Vila Madalena
4	Marcela Campos	99220377221	71 973514498	1980-01-19	F	Barra
5	Rodrigo Gonçalves	72177589991	21 986121942	1992-05-10	M	Centro
6	Jorge Marinho	83144619976	11 995439812	1990-06-07	M	Morumbi
7	Rodrigo Vieira	73520421929	71 972318872	1985-08-27	M	Centro
8	Vanessa Aquino	76211038739	21 933211346	1972-02-15	F	Centro
9	Fabiana Moreira	31160319971	11 933216758	1985-04-13	F	Vila Madalena
10	Maria Conceição	82410399921	21 954419257	1999-07-10	F	Centro

Quadro 3.9 – Dados de exemplos para os comandos
Fonte: Elaborado pelo autor.

Vale comentar que são dados de uma tabela simples, talvez em uma situação real os dados podem ser bem mais complexos do que estes e com uma estrutura diferente; talvez, em vez de tratar “Bairro” como um VARCHAR, ele seria uma chave estrangeira para uma tabela de bairros. O propósito, neste momento, é apenas ter alguns dados para compreender as variações de comandos *Select* que vamos estudar.

Primeiro exemplo: *SELECT* simples para colunas e

linhas

Para começar, consideremos um formato bem simples de *SELECT* em que especificamos algumas colunas que nos interessam e estabelecemos um filtro para algumas linhas. A necessidade é de recuperar “os clientes, especificamente nome, contato e data de nascimento dos clientes do sexo feminino”, e o comando está exposto no Quadro 3.10. A coluna mostra a forma, a sintaxe de escrever o comando.

Exemplo de Comando: SELECT simples	**Sintaxe**
SELECT NOME, CONTATO, DATA_NASC FROM CLIENTE WHERE SEXO = 'F'	SELECT < *colunas* > FROM < *tabela* > WHERE < *filtros* >

Quadro 3.10 – Exemplo de um comando simples de SELECT
Fonte: Própria.

O resultado da consulta acima deve ser o da Figura 3.6. Perceba que, ao observar os nossos registros de exemplo (Figura 3.11), apenas as linhas com os códigos 1, 3, 4, 8, 9 e 10 satisfazem a condição de “SEXO = 'F'”, isto é, clientes do sexo feminino. E quanto às colunas, apenas NOME, CONTATO e DATA_NASC foram mostradas.

NOME	**CONTATO**	**DATA_NASC**
Adriana Araújo	71 982213455	1987-02-03
Viviane Sales	11 987712022	1995-11-02
Marcela Campos	71 973514498	1980-01-19
Vanessa Aquino	21 933211346	1972-02-15
Fabiana Moreira	11 933216758	1985-04-13
Maria Conceição	21 954419257	1999-07-10

Figura 3.11 – Resultado da consulta inicial com Select
Fonte: Própria.

Operadores para refinar os filtros das linhas

A partir deste exemplo simples, é possível refinar a forma de expressão das sentenças que

representam nossa necessidade sobre os dados em condições. Essas condições compõem os filtros (cláusula *WHERE*) da consulta e apenas as linhas que satisfazem as condições são apresentadas no conjunto de resultados.

Por exemplo, como recuperar os Clientes que nasceram entre 2000 e 2005? Como selecionar os Clientes do sexo masculino que moram no Bairro Barra? E como capturar um cliente em específico que se chama “Viviane”? Enfim, neste momento de consultas básicas, o foco é desenvolver esta habilidade de traduzir uma necessidade sobre os dados e os seus filtros correspondentes.

O Quadro 3.12 a seguir apresenta algumas operações que podem ser realizadas na edição dos filtros. Os filtros envolvem operações de comparação para valores como números e datas. Existem outras operações comuns e também os operadores lógicos que permitem combinar condições compondo filtros maiores.

Operador	**Comentário**
Igual: = Maior e menor: > < Com igual: >= <= Diferente: <> !=	Permitem a comparação de valores. O operador igual (‘=’) é muito usado em diversos tipos de campos. Os outros são mais comuns em campos com valores numéricos, de data em que a relação “maior” e “menor” faz mais sentido Exemplo: CODIGO >= 10
like	Usado para reconhecer padrões ou subpalavras em campos textos. - Para ver se o nome possui a palavra “Viviane”, - o % indica “qualquer caractere” Exemplo: NOME LIKE ‘%Viviane%’
betweeen	Verifica se os valores estão entre um limite inferior e um limite superior Exemplo: CODIGO BETWEEN 3 AND 20

Quadro 3.12 – Operadores para compor condições em comandos SQL
Fonte: Elaborado pelo autor.

Para melhorar a compreensão, o interessante é observar alguns exemplos. Os quadros a seguir

mostram alguns e comentam outros recursos comuns na escrita de consultas.

Quadro 3.13 mostra um exemplo de comandos de comparação e alguns comentários sobre o uso de campos do tipo data. Um recurso muito útil e rápido com consultas é o uso do asterisco (‘*’) no lugar das colunas. Ele indica que deseja recuperar todas as colunas.

Exemplos de Comandos: SELECT com operadores de comparação

- Para recuperar os clientes que nasceram a partir de 1990

SELECT *
FROM CLIENTE
WHERE **DATA_NASC >= '1990-01-01'**

- Comentários:
- O campo data por padrão é tratado nesta forma ‘YYYY-MM-DD’
- Existem funções para data que podem ajudar neste tipo de camp
- O uso do ‘ * ‘ (asterisco) no lugar das colunas traz todas as colunas

Resultado

CODIGO	NOME	CPF	CONTATO	DATA_NASC	SEXO	BAIRRO
3	Viviane Sales	43517386521	11 987712022	1995-11-02	F	Vila Madalena
5	Rodrigo Gonçalves	72177589991	21 986121942	1992-05-10	M	Centro
6	Jorge Marinho	83144619976	11 995439812	1990-06-07	M	Morumbi
10	Maria Conceição	82410399921	21 954419257	1999-07-10	F	Centro

Quadro 3.13 – Operadores para compor condições em comandos SQL
Fonte: Elaborado pelo autor.

O Quadro 3.14 mostra um exemplo de comandos com o uso do *LIKE* .

Exemplos de Comandos: SELECT com operador LIKE

- Para recuperar os clientes que apresentam Viviane no nome

```
SELECT    NOME, CONTATO
  FROM    CLIENTE
 WHERE    NOME LIKE '%Viviane%'
```

- Comentários:
- O ‘%’ indica “qualquer caractere
- Com apenas 'Viviane%', seriam consideradas as linhas que começam com o nome Viviane

Resultado

NOME	CONTATO
Viviane Sales	11 987712022

Quadro 3.14
Fonte: Elaborado pelo autor.

No exemplo do Quadro 3.15, os clientes nascidos em determinado período de tempo são processados com o operador *BETWEEN* .

Exemplos de Comandos: SELECT com operador BETWEEN

- Para recuperar os clientes que nasceram entre 1990 e 1995

```
SELECT NOME, CONTATO, DATA_NASC
FROM CLIENTE
WHERE DATA_NASC BETWEEN '1990-01-01' AND '1995-12-31 23:59'
```

- Comentários:
- Perceba que a data final foi tratada com hora para incluir os clientes nascidos naquele dia de 31/12/1995

Resultado

NOME	CONTATO	DATA_NASC
Viviane Sales	11 987712022	1995-11-02
Rodrigo Gonçalves	21 986121942	1992-05-10
Jorge Marinho	11 995439812	1990-06-07

Quadro 3.15
Fonte: Elaborado pelo autor.

Operadores lógicos *AND* , *OR* e *NOT*

Além dos operadores de comparação, os operadores lógicos permitem expressar filtros com recursos mais avançados, combinando condições com operação de E, OU e NÃO. Um operador lógico permite combinar situações, por exemplo, ao se consultar clientes do sexo feminino E que moram na Barra; o conectivo “E” faz com que apenas clientes que satisfaçam às duas condições sejam considerados no resultado. Para o OU, se apenas uma das condições forem satisfeitas, a linha é considerada. Por exemplo, para clientes que nasceram depois de 1995 OU que sejam do sexo feminino, linhas que satisfizerem pelo menos uma das condições serão consideradas.

O Quadro 3.16 apresenta como expressar as consultas com o operador lógico E (AND).

Exemplos de Comandos: SELECT com operador lógico E

- Para recuperar os clientes do sexo feminino e que moram na Barra

```
SELECT      NOME, SEXO, BAIRRO
    FROM    CLIENTE
WHERE       SEXO = 'F' AND BAIRRO = 'Barra'
```

- Comentários:
- Perceba que no resultado a seguir os dois campos apresentam valores que satisfazem a condição.

Resultado

NOME	SEXO	BAIRRO
Adriana Araújo	F	Barra
Marcela Campos	F	Barra

Quadro 3.16
Fonte: Elaborado pelo autor.

O Quadro 3.16 apresenta exemplo de consulta com o operador lógico OU (*OR*).

Exemplos de Comandos: SELECT com operador lógico OU

- Para recuperar os clientes que nasceram até 1980 ou que sejam do sexo masculino

```
SELECT    NOME, DATA_NASC, SEXO
FROM      CLIENTE
WHERE     DATA_NASC < '1980-01-01' OR SEXO = 'M'
```

- Comentários:
- Perceba que no resultado a última linha (sexo = ‘F’) só foi considerada porque a data de nascimento é inferior a 1980, uma das condições foi satisfeita.
- Por outro lado, perceba que Jorge Marinho nasceu em 1990 porém foi considerado por conta de ser do sexo masculino, satisfez uma das condições.
- Já Renato foi considerado porque satisfez as duas condições.

Resultado

NOME	DATA_NASC	SEXO
Renato Nogueira	1977-07-09	M
Rodrigo Gonçalves	1992-05-10	M
Jorge Marinho	1990-06-07	M
Rodrigo Vieira	1985-08-27	M
Vanessa Aquino	1972-02-15	F

Quadro 3.17
Fonte: Elaborado pelo autor.

Ainda existe o operador *NOT* que recupera uma linha quando ela não satisfaz certa condição, por exemplo, imagine que queiramos os clientes que não moram na Barra. O Quadro 3.18 exemplifica este caso.

Exemplos de Comandos: SELECT com operador lógico NOT

- Para recuperar os clientes que não moram na Barra

```
SELECT    NOME, SEXO, BAIRRO
  FROM    CLIENTE
WHERE     NOT (BAIRRO = 'Barra')
```

- Comentários:
- Perceba que nenhum dos registros apresentam o Bairro igual a Barra
- O filtro é equivalente a escrever WHERE BAIRRO <> 'Barra'
- Isto é, considerar linhas cujo bairro seja diferente (NOT igual) a Barra

Resultado

NOME	SEXO	BAIRRO
Renato Nogueira	M	Morumbi
Viviane Sale	F	Vila Madalena
Rodrigo Gonçalves	M	Centro
Jorge Marinho	M	Morumbi
Rodrigo Vieira	M	Centro
Vanessa Aquino	F	Centro
Fabiana	F	Vila Madalena
Maria Conceição	F	Centro

Quadro 3.18
Fonte: Elaborado pelo autor.

O comando *SELECT* é bastante versátil e útil para recuperar os dados de tabelas combinando as mais variadas demandas por informações em sentenças e condições sobre os campos de uma tabela.

Imagine que uma equipe de marketing deseja fazer uma campanha sobre os clientes de nosso exemplo sobre Loja. Para esta situação, é necessário recuperar os clientes do sexo Masculino que moram no Morumbi ou Vila Madalena.

Qual comando *select* a seguir atenderia a esta demanda?

- **a)** SELECT NOME, CONTATO FROM CLIENTE
  WHERE SEXO = 'M'
  OR (BAIRRO = 'Morumbi' AND BAIRRO = 'Vila Madalena')
- **b)** SELECT * FROM CLIENTE
  WHERE SEXO = 'M'
  AND (BAIRRO = 'Morumbi')
- **c)** SELECT * FROM CLIENTE
  WHERE (BAIRRO = 'Morumbi' AND BAIRRO = 'Vila Madalena')
- **d)** SELECT NOME, CONTATO FROM CLIENTE
  WHERE SEXO = 'M'
  AND (BAIRRO = 'Morumbi' OR BAIRRO = 'Vila Madalena')
- **e)** SELECT * FROM CLIENTE
  WHERE SEXO = 'F'
  AND (BAIRRO = 'Morumbi' OR BAIRRO = 'Vila Madalena')

Inserção, Alteração e Exclusão de Dados

Parte dos grupos de comandos de DML realiza alterações nas tabelas de dados. Por exemplo: um novo registro pode ser criado (*INSERT*), um registro existente pode ter dados alterados (*UPDATE*) e linhas podem ser excluídas (*DELETE*). Juntamente com o *SELECT* , esses comandos formam as operações que são comuns de serem feitas por usuário em sistemas. Imagine, por exemplo, um carrinho de compras, em uma loja virtual, no qual itens podem ser adicionados, retirados e um item, já no carrinho, pode ter a quantidade alterada. Além disso, para visualizar os dados do carrinho, uma consulta com *SELECT* deve ter sido feita antes.

Inserção (*INSERT*)

No comando *INSERT* , para cada um dos campos da tabela, um determinado valor deve ser estabelecido. O Quadro 3.19 mostra alguns destes comandos que originaram as duas primeiras linhas de nosso conjunto de dados na tabela de CLIENTE. Perceba que cada campo é listado logo após o nome da tabela e, na cláusula *VALUES* , cada valor é especificado obedecendo à ordem dos campos.

Exemplos de Comandos: INSERT

- Caso inserir em uma suposta tabela CIDADE

```
INSERT INTO CIDADE (COD_CIDADE, DCR_CIDADE)
VALUES (1, 'São Paulo')
```

- Exemplo de duas linhas usadas para inserção de nossos dados

```
INSERT INTO CLIENTE
( CODIGO, NOME, CPF, CONTATO,
DATA_NASC, SEXO, BAIRRO)
VALUES
( 1, 'Adriana Araújo', '01120389921', '71 982213455',
'1987-02-03', 'F', 'Barra')

INSERT INTO CLIENTE
( CODIGO, NOME, CPF, CONTATO,
DATA_NASC, SEXO, BAIRRO)
VALUES
( 2, 'Renato Nogueira', '98220379931', '11 933321999',
'1977-07-09', 'M', 'Morumbi')
```

- Comentários:
- Perceba que nenhum dos registros apresentam o Bairro igual a Barra
- O filtro é equivalente a escrever WHERE BAIRRO <> 'Barra'
- Isto é, considerar linhas cujo bairro seja diferente (NOT igual) a Barra

Resultado ao consultar a tabela depois das inserções de Clientes exemplificadas

1	Adriana Araújo	01120389921	71 982213455	1987-02-03	F	Barra
2	Renato Nogueira	98220379931	11 933321999	1977-07-09	M	Morumbi

Quadro 3.19 – Comando INSERT para inserção de dados
Fonte: Elaborado pela autora.

Dos exemplos, podemos perceber que a sintaxe do comando *INSERT* pode ser resumida conforme o Quadro 3.20.

Sintaxe do comando INSERT

INSERT INTO < *Nome da tabela* > (< *Lista de campos*>)
VALUES (< Lista de valores, na mesma ordem dos campos>)

Quadro 3.20 – Sintaxe do comando INSERT
Fonte: Elaborado pelo autor.

Alteração (*UPDATE*)

Para a alteração dos registros, naturalmente eles já devem estar contidos nas tabelas. O comando precisa portanto de indicar quais campos receber novos valores e em geral é especificado um filtro (semelhante ao que foi visto no *SELECT*) para selecionar quais linhas devem sofrer as alterações. O Quadro 3.21 mostra dois exemplos de comandos *UPDATE* . Vale ressaltar que estes comandos de *UPDATE* , assim como o de DELETE devem ser usados com cuidado pois podem ocasionar perda de dados.

Exemplos de Comandos: UPDATE

- Para modificar o nome do bairro ‘Barra’ para ‘Nova Barra’

```
UPDATE   CLIENTE
SET      BAIRRO = 'Nova Barra'
WHERE    BAIRRO = 'Barra'
```

- Para modificar o bairro do cliente de código 10 para 'Nova Barra'

```
UPDATE   CLIENTE
SET      BAIRRO = 'Nova Barra'
WHERE    CODIGO = 10
```

Comentários

- Perceba que a coluna Bairro na linha 10 está com valor Nova Barra.
- Apenas o último exemplo foi processado para exemplificar os resultados a seguir.

8	Vanessa Aquino	76211038739	21 933211346	1972-02-15	F	Centro
9	Fabiana Moreira	76211038739	11 933216758	1985-04-13	F	Vila Madalena
10	Maria Conceição	82 410399921	21 954419257	1999-07-10	F	Nova Barra

Quadro 3.21 – Comando UPDATE
Fonte: Elaborado pelo autor.

Dos exemplos, podemos perceber que a sintaxe do comando de *UPDATE* pode ser resumida conforme o Quadro 3.22.

Sintaxe do comando UPDATE

UPDATE < *Nome da tabela* >
SET < *Nome do campo* > = < *novo valor* > ,
< *Nome do campo* > = < *novo valor* > , ...
WHERE < Filtros>

Quadro 3.22 – Sintaxe do comando UPDATE
Fonte: Elaborado pelo autor.

Exclusão (DELETE)

A exclusão dos dados ocorre com o comando DELETE e sua sintaxe também deve apresentar um filtro que indica quais linhas devem ser afetadas pela exclusão. O Quadro 3.23 apresenta alguns exemplos de comandos de exclusão de registros.

Exemplos de Comandos: DELETE

- Para modificar o nome do bairro ‘Barra’ para ‘Nova Barra’

```
DELETE FROM   CLIENTE
WHERE         BAIRRO = 'Barra'
```

- Para excluir o cliente de código 7

```
DELETE FROM   CLIENTE
WHERE         CODIGO = 7
```

Comentários

- Perceba que na lista de clientes não está mais o registro com Código = 7
- Apenas o último exemplo foi processado para exemplificar os resultados a seguir, por isso o conjunto ainda possuem registros com bairro ‘Barra’.

CODIGO	NOME	CPF	CONTATO	DATA_NASC	SEXO	BAIRRO
1	Adriana Araújo	01120389921	71 9820212355	1987-02-03	F	Barra
2	Renato Nogueira	98220379931	11 933321999	1977-07-09	M	Morumbi
3	Viviane Sales	43517386521	11 987712022	1995-11-02	F	Vila Madalena
4	Marcela Campos	99220377221	71 973514498	1980-01-19	F	Barra
5	Rodrigo Gonçalves	72177589991	21 986121942	1992-05-10	M	Centro
6	Jorge Marinho	83144619976	11 995439812	1990-06-07	M	Morumbi
8	Vanessa Aquino	76211038739	21 933211346	1972-02-15	F	Centro

9	Fabiana Moreira	31160319971	11 933216758	1985-04-13	F	Vila Madalena
10	Maria Conceição	82410399921	21 954419257	1999-07-10	F	Nova Barra

Quadro 3.23 – Comando DELETE
Fonte: Elaborado pelo autor.

Dos exemplos, podemos perceber que a sintaxe do comando *UPDATE* pode ser resumida conforme o Quadro 3.24.

Sintaxe do comando DELETE

DELETE FROM < *Nome da tabela* >
WHERE < *Filtros* >

Quadro 3.24 – Sintaxe do comando DELETE
Fonte: Própria.

praticar

Vamos Praticar

Os comandos *INSERT* , *UPDATE* e *DELETE* permitem a manipulação dos dados (DML) e é importante que eles sejam escritos com atenção. Imagine que uma tabela de CLIENTE possui os seguintes registros:

CODIGO	NOME	CPF	CONTATO	DATA_NASC	SEXO	BAIRRO
1	Adriana Araújo	01120389921	71 982213455	1987-02-03	F	Barra
2	Renato Nogueira	98220379931	11 933321999	1977-07-09	M	Morumbi
3	Viviane Sales	43517386521	11 987712022	1995-11-02	F	Vila Madalena
4	Marcela Campos	99220377221	71 973514498	1980-01-19	F	Barra
5	Rodrigo Gonçalves	72177589991	21 986121942	1992-05-10	M	Centro
6	Jorge Marinho	83144619976	11 995439812	1990-06-07	M	Morumbi
7	Rodrigo Vieira	73520421929	71 972318872	1958-08-27	M	Centro
8	Vanessa Aquino	76211038739	21 933211346	1972-02-15	F	Centro
9	Fabiana Moreira	31160319971	11 933216758	1985-04-13	F	Vila Madalena
10	Maria Conceição	82410399921	21 954419257	1999-07-10	F	Centro

Imagine que o seguinte comando de DELETE foi aplicado ao banco de dados

DELETE FROM CLIENTE

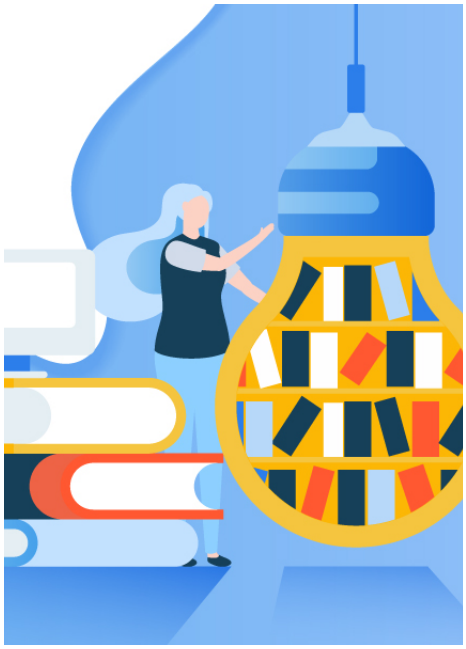
WHERE (Bairro = ‘Barra’) OR (Sexo = ‘F’)

Quais os códigos de cliente das linhas que serão afetadas por este comando?

- **a)** 1 e 4
- **b)** 1, 3, 4, 8, 9, 10
- **c)** 7, 8 e 10
- **d)** 8 e 10
- **e)** 2 e 6

indicações

Material Complementar



LIVRO

Use a cabeça! SQL

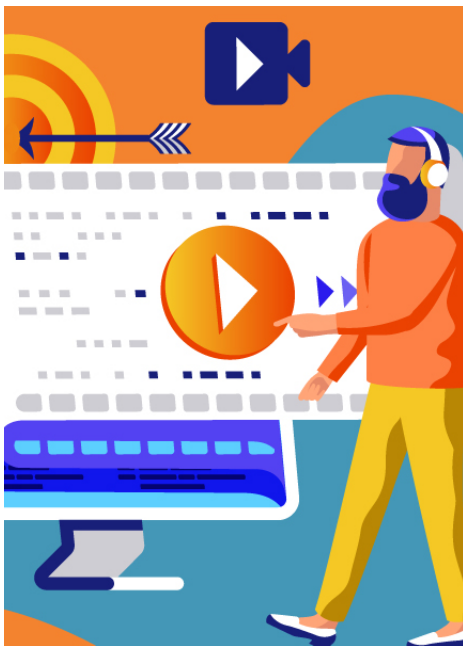
Editora : O'RELLY

Autor : Lynn Beighley

ISBN : 9788576082101

Comentário : Para esta unidade, vale ter à mão um livro de referência sobre os comandos SQL. Ao longo do material, foram apresentadas algumas obras para oferecer uma base, mas existe um universo vasto de possibilidades em que um livro ou mesmo um manual de SQL de algum servidor de dados será bastante útil ou para uma leitura, ou para consultas quando precisar de algum comando em específico.

Além de oferecer esta fonte para consultas, a linguagem deste livro é mais leve, espero que se identifique!



FILME

Estrelas Além do Tempo

Ano : 2017

Comentário : Este filme conta a trajetória de três mulheres negras por volta da década de 60, e que trajetória!

Para este tema de banco de dados, há uma curiosidade que, no filme, é retratada quando a NASA passa a ser equipada com os primeiros computadores IBM.

A IBM sempre foi uma gigante de TI e nela surgiu, por exemplo o artigo de Codd sobre bancos de dados relacionais e que sustentaram os fundamentos de toda esta tecnologia por todos estes anos. Isso não aparece no filme, mas é possível ver um cenário dos primeiros *mainframes* IBM na NASA.

Voltando ao filme, esteja atento à perspicácia de uma das personagens ao perceber esta chegada dos computadores IBM e se mobilizar, junto às outras colegas negras, para estudar e se capacitar nesta tecnologia e na linguagem para programá-los, já que isso seria o futuro. Depois elas conseguem “dominar a cena” justamente por terem se antecipado a esta capacitação.

Ah, e não se esqueça que isso acontece aproximadamente na década de 60 com três mulheres negras precisando transpor todas as barreiras raciais e de gênero que impediam o desenvolvimento delas. Inspirador para não dizer emocionante! É o poder transformador da educação.

É isso, caro aluno, mais um filme para se inspirar, refletir e se divertir! Aperte o play!

Para conhecer mais sobre o filme, acesse o trailer disponível

TRAILER

conclusão

Conclusão

Esta unidade abordou as habilidades necessárias para operar um banco de dados depois de ter sido modelado.

De início, vimos como criar o banco de dados em si, refletindo as definições do modelo, além de como os comandos DDL permitiam criar as tabelas com suas colunas, restrições, características dos campos e outros.

A partir daí, o foco foi em como manipular os registros destas tabelas, se os registros estão presentes, como recuperá-los com o comando *SELECT* . Caso precisem ser modificados, como fazer inserções, alterações e exclusões com os comandos *INSERT* , *UPDATE* e *DELETE* , respectivamente.

E tudo isso tentando mostrar exemplos. Esta parte prática com comandos SQL e seus resultados é interessante ao aprender sobre banco de dados.

Espero que tenha aproveitado!

---

referências

Referências Bibliográficas

CODD, E. F. A relational model of data for large shared data banks. **Information Retrieval** , v.13, n.6, 1970.

ELMASRI, R.; NAVATHE, S. **Sistemas de banco de dados** . São Paulo: Pearson Addison Wesley, 2011.

PORVIR. **Aprendizagem baseada em projetos** . [2019?]. Disponível em: < http://porvir.org/especiais/maonamassa/aprendizagem-baseada-em-projetos >. Acesso em: 28 maio 2019.

PUGA, S. **Banco de dados** : implementação em SQL, PL/SQL e Oracle 11g. São Paulo: Pearson Education do Brasil, 2013.