Linguagem SQL básica

Apresentação

A *Structured Query Language* (SQL) é uma linguagem de consulta estruturada, pela qual é possível realizar diferentes tipos e formas de consultas, utilizadas para recuperar informações em um banco de dados. Também pode ser aplicada para a definição, a manipulação e as atualizações de dados.

Nesta Unidade de Aprendizagem, você estudará algumas das funcionalidades da SQL, como as categorias de instruções da linguagem, esquema e catálogo de banco de dados em SQL e, ainda, sobre tipos de restrições.

Bons estudos.

Ao final desta Unidade de Aprendizagem, você deve apresentar os seguintes aprendizados:

- Identificar as categorias de instruções da linguagem SQL (DDL, DML, TCL e DCL).
- Demonstrar como criar e alterar o esquema e o catálogo em banco de dados SQL.
- Descrever os tipos de restrições e seus exemplos.

Infográfico

Um grande número de restrições (*constraints*) é criado sobre os valores a serem inseridos e armazenados em um estado do banco de dados. Essas restrições podem ser divididas em três grupos principais: as que são inerentes ao modelo de dados, as que podem ser definidas diretamente nos esquemas do modelo de dados e as que não podem ser definidas diretamente nos esquemas do modelo de dados e, por isso, são impostas pelos programas de aplicação. As restrições podem ser especificadas quando a tabela é criada, com a instrução CREATE TABLE, ou após a tabela ser criada, com a instrução ALTER TABLE.

Neste Infográfico, você vai ver uma representação gráfica das restrições utilizadas em SQL.

RESTRIÇÕES UTILIZADAS EM SQL

DEFAULT

Especifica o valor de uma coluna quando não for informado via INSERT.

Exemplo: neste exemplo, é criada uma tabela "Carros", em que o valor padrão da coluna modelo é "Fusca":

```
CREATE TABLE Carros(
Cod_placa CHAR(20) PRIMARY KEY,
nm_modelo VARCHAR(30) DEFAULT
'Fusca'
);
```

CHECK

Especifica quais valores podem ser aceitos em uma coluna.

Exemplo: este exemplo cria uma tabela "Alunos", checando se o campo idade é maior que 10:

```
CREATE TABLE Alunos(
id_aluno CHAR(20) NOT NULL,
nome CHAR(30) NOT NULL,
idade INTEGER NOT NULL CHECK
(idade > 10)
);
```

REFERENCES

Especifica quais valores podem ser aceitos em um UPDATE, com base nos valores de uma coluna de outra tabela.

Exemplo: aqui, é criada a tabela "Consulta", com uma chave primária e outra externa da tabela referência, que é "Cliente":

```
CREATE TABLE Consulta(
Cod_consulta CHAR(20) PRIMARY KEY,
Cpf_cliente VARCHAR(11) NOT NULL,
Cpf_esp VARCHAR(11) NOT NULL,
FOREIGN KEY (Cpf_cliente)
REFERENCES cliente
);
```

PRIMARY KEY

Identifica a linha da tabela de forma única, para que os usuários não insiram valores duplicados. Valores nulos não são permitidos.

Exemplo: este exemplo é da criação da tabela "Estudantes", com a definição de "id_aluno" como chave primária:

```
CREATE TABLE Estudantes(
id_aluno CHAR(20) PRIMARY KEY,
nome CHAR(30) NOT NULL
);
```

UNIQUE

Impede a duplicação de chaves candidatas e garante que um índice seja criado para melhora do desempenho. Valores nulos são permitidos.

Exemplo: novamente em um exemplo da tabela "Alunos", agora definindo que "id_alunos" não pode ter valores repetidos:

```
CREATE TABLE Alunos(
id_aluno CHAR(20) UNIQUE,
nome CHAR(30) NOT NULL
);
```

FOREIGN KEY

Define uma coluna ou combinação de colunas, das quais os valores correspondem a uma chave primária, na mesma ou em outra tabela.

Exemplo: agora, um exemplo em que fica clara a chave externa vinda da tabela "Cliente" para a tabela "Consulta":

```
CREATE TABLE Consulta(
Cod_consulta CHAR(20) PRIMARY KEY,
Cpf_cliente VARCHAR(11) NOT NULL,
FOREIGN KEY (Cpf_cliente)
REFERENCES cliente
);
```

Aponte a câmera para o código e acesse o link do conteúdo ou clique no código para acessar.

Conteúdo do Livro

Os bancos de dados relacionais utilizam sistemas de gerenciadores de banco de dados (SGDB), a fim de facilitar a sua manipulação. Esses SGBD, por sua vez, utilizam uma linguagem de consulta estruturada, a SQL. Por meio da SQL, é possível criar tabelas, gerenciar relações, incluir atributos e também restringir dados que não podem ou não devem ser inseridos nas tabelas, entre outras funções. Por este motivo, a SQL é considerada a linguagem padrão de manipulação de banco de dados relacionais.

No capítulo Linguagem SQL básica, da obra *Banco de dados*, base teórica desta Unidade de Aprendizagem, você vai conhecer os conceitos principais das instruções SQL, como criação e atualização de tabelas e definição de restrições na alimentação de um banco de dados.

Boa leitura.

BANCO DE DADOS

Roni Francisco Pichetti

SAGAH
SOLUÇÕES EDUCACIONAIS INTEGRADAS

Linguagem SQL básica

Objetivos de aprendizagem

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

- Identificar as categorias de instruções da linguagem SQL (DDL, DML, TCL e DCL).
- Demonstrar como criar e alterar o esquema e o catálogo em banco de dados SQL.
- Descrever os tipos de restrições (*constraints*) e seus exemplos.

Introdução

Os bancos de dados são empregados para armazenar dados que serão utilizados novamente por sistemas de informação ou aplicativos para as mais diversas funções. Para facilitar a manipulação desses aglomerados de dados, foram criados os sistemas de gerenciamento de banco de dados (SGBDs), que são utilizados em bancos de dados relacionais. Os pacotes de SGBD relacionais comerciais possuem características únicas, mas têm em comum a utilização da linguagem de consulta estruturada (SQL, do inglês *structured query language*), que se tornou um padrão para esse tipo de banco de dados.

Neste capítulo, você vai estudar as categorias de instruções da SQL e vai compreender como criar e alterar esquemas de dados e catálogos com SQL. Além disso, você vai conferir exemplos de tipos de restrições utilizadas na SQL.

1 Categorias de instruções em SQL

A **SQL** é uma linguagem abrangente, pois possui instruções para definição, consulta, atualização e eliminação de dados. Por esse motivo, ela pode ser considerada uma linguagem de definição de dados (DDL, do inglês *data definition language*), uma linguagem de manipulação de dados (DML, do inglês *data manipulation language*), uma linguagem de controle de dados (DCL, do inglês *data control language*) ou uma linguagem de controle de transações (TCL, do inglês *transactional control language*). A SQL possibilita definir visões em um banco de dados, a fim de especificar aspectos de segurança e autorização, definir restrições de integridade e especificar o controle de transação de dados. A SQL também possui regras para embutir suas instruções em uma linguagem de programação, como Java (ELMASRI; NAVATHE, 2011).

De acordo com Ramarkrishnan e Gehrke (2011), o subconjunto da SQL chamado **DDL** suporta criação, exclusão e modificação de definições das tabelas e visões, por meio de comandos como `ALTER`, `CREATE`, `DROP`, `RENAME`, `DISABLE TRIGGER`, `ENABLE TRIGGER`, `COMMENT` e `TRUNCATE TABLE`. Já o subconjunto **DML** permite que os usuários formulem consultas e insiram, excluam e modifiquem tuplas. Para isso, utilizam-se comandos como `SELECT`, `INSERT`, `UPDATE` e `DELETE`. Já a **DCL** é utilizada para controlar o acesso e gerenciar permissões de usuários em bancos de dados. Alguns exemplos de comandos utilizados nesse caso são: `GRANT`, `REVOKE` e `DENY`. Por fim, o subconjunto **TCL** é utilizado para gerenciar mudanças feitas por instruções em DML, o que possibilita agrupar declarações em transações lógicas. Para isso, pode-se usar `COMMIT` (salva permanentemente uma transação no banco de dados) e `ROLLBACK` (restaura o banco de dados para o último estado do comando `COMMIT`).

A linguagem SQL padrão utiliza a palavra **tabela** para identificar uma relação, enquanto as **linhas** são as tuplas ou os registros das tabelas de dados, e as **colunas** são os campos ou os seus atributos. Na DDL, a instrução `CREATE TABLE` é empregada para criar uma nova tabela, enquanto as linhas são inseridas por meio do comando `INSERT`; com a cláusula `INTO`, são apontadas as colunas nas quais os valores serão inseridos, conforme lecionam Ramarkrishnan e Gehrke (2011). Para compreender melhor essas cláusulas, verifique os exemplos a seguir.

Exemplo

Criação da tabela `Estudantes` com definição de atributos:

```
CREATE TABLE Estudantes(
  id-aluno CHAR(20),
  nome VARCHAR(30),
  login VARCHAR (20),
  idade INTEGER
  );
```

Inserção de dados na tabela `Estudantes`:

```
INSERT
INTO Estudantes (id-aluno, nome, login, idade),
VALUES (53456, 'Beto', 'beto@email.com', 19);
```

Perceba que os atributos do exemplo são criados com a definição do tipo de dado, bem como de seu tamanho. Ao se inserir valores na tabela Estudantes, eles são organizados na mesma ordem em que foram criados, a fim de que a tabela seja alimentada corretamente. Geralmente, o esquema SQL no qual as relações são declaradas é especificado de forma implícita na criação das tabelas. Porém, é possível conectar de forma explícita o nome do esquema com o nome da relação, separando-os com um ponto, conforme o exemplo a seguir:

```
CREATE TABLE Escola.Estudantes...
```

em vez de:

```
CREATE TABLE Estudantes...
```

O CREATE é o principal comando SQL para a criação de tabelas, mas também pode ser utilizado para a criação de esquemas e domínios. Dessa forma, o comando CREATE TABLE é empregado para especificar uma nova relação. Esta precisa ter um nome definido, bem como especificados os seus atributos e as possíveis restrições. Primeiramente, especificam-se os atributos, com a definição de um nome para cada um deles e um tipo de dados, conforme pode ser visto no primeiro exemplo da criação da tabela 'Estudantes'. Na mesma cláusula CREATE TABLE, pode ser definido o domínio de valores e restrições dos atributos, como o fato de serem obrigatórios e não poderem ficar em branco (NOT NULL). As restrições de chave e de integridade, tanto de entidade quanto referencial, também podem ser incluídas nessa cláusula. Caso a tabela criada necessite ser alterada posteriormente, pode ser utilizado o comando ALTER TABLE (ELMASRI; NAVATHE, 2011).

Entre as principais instruções DML em SQL, estão SELECT, INSERT, UPDATE e DELETE. A instrução SELECT seleciona registros e campos específicos em um banco de dados. Já a instrução INSERT é utilizada para a inclusão de novos valores a atributos do banco de dados. Por meio da instrução UPDATE, os valores de atributos são atualizados, e utilizando-se DELETE, são apagados do banco de dados (ELMASRI; NAVATHE, 2011). Em toda consulta, é necessário que se tenha uma cláusula SELECT com a especificação das colunas que devem ser mantidas no resultado, bem como uma cláusula FROM para indicar o produto cartesiano das tabelas. Nesse caso, a cláusula WHERE pode ser opcional, pois é utilizada para refinar a consulta.

Para Ramarkrishnan e Gerhrke (2011), a forma básica de uma consulta SQL é a seguinte:

```
SELECT [DISTINCT] <lista-seleção>
FROM <lista-from>
WHERE <qualificação>
```

Assim, a cláusula SELECT consulta os nomes de colunas das tabelas presentes na lista-from, e a cláusula FROM busca uma lista com os nomes de tabelas. Já a qualificação realizada na cláusula WHERE é uma expressão que se utiliza dos conectivos ou operadores lógicos NULL, AND, OR e NOT ("nulo", "e", "ou" e "não", respectivamente) para realizar uma seleção dos dados presentes na tabela consultada (RAMARKRISHNAN; GEHRKE, 2011).

De acordo com Pires (2017), a DCL controla aspectos de autorização de dados, bem como licenças de usuários, a fim de controlar quem pode ter acesso para ver ou manipular os dados de um banco de dados. Dessa forma, são os comandos que possibilitam gerenciar a segurança dos dados armazenados, por meio de níveis de acesso e de privilégios para os usuários do banco de dados. Os comandos DCL mais comuns nos SGBDs são:

- `GRANT`, que autoriza privilégios aos usuários, para executarem comandos, consultas e alterações nos dados armazenados; e
- `REVOKE`, que basicamente revoga os privilégios concedidos ao usuário pelo comando anterior.

A seguir, são apresentados dois exemplos, um autorizando acesso ao banco de dados e outro revogando o acesso (PIRES, 2017).

Exemplo

Exemplo de `GRANT`:

```
GRANT create _ table
TO usuario _ teste;
```

Exemplo de `REVOKE`:

```
REVOKE create _ table
FROM usuário _ teste;
```

No primeiro exemplo, é autorizado que `usuario_teste` tenha acesso ao comando de criar tabelas. Já no segundo exemplo, o privilégio da criação de tabelas é retirado.

2 Criação e alteração de esquema e catálogo em banco de dados SQL

Um **esquema** em SQL possui um nome de esquema e um identificador de autorização, para indicar seu usuário ou proprietário, além de descritores de cada elemento. Entre esses elementos estão tabelas, restrições, visões (*views*), domínios, entre outras informações que descrevem o esquema. O esquema é criado pelo comando `CREATE SCHEMA`, que pode incluir todas as definições dos elementos do esquema (ELMASRI; NAVATHE, 2011). Para Watson (2010), um esquema é criado quando uma conta de usuário do SQL é criada; por isso, ele consiste em objetos que pertencem à conta. No exemplo a seguir, é apresentada a instrução SQL de criação de um esquema chamado `ESCOLA`, que pertence ao usuário `'Amenezes'` (ELMASRI; NAVATHE, 2011).

```
CREATE SCHEMA ESCOLA
AUTHORIZATION 'Amenezes';
```

De forma geral, nem todos os usuários do banco de dados são autorizados a criar esquemas. O privilégio para criar esquemas, tabelas e outras construções do banco de dados deve ser concedido pelo seu administrador (ELMASRI; NAVATHE, 2011).

A SQL disponibiliza comandos de evolução de esquema, utilizados para alterar um esquema, o que significa acrescentar ou remover tabelas, atributos, restrições, entre outros elementos. Essas alterações podem ser realizadas enquanto o banco de dados está sendo operado, e não é necessária a recompilação do esquema. Um dos comandos úteis nesse caso é o `DROP`, que pode ser empregado para remover elementos nomeados do esquema, como tabelas, domínios ou restrições. Por meio dele, também é possível remover o próprio esquema (ELMASRI; NAVATHE, 2011).

O `DROP` pode ser utilizado de duas formas: `CASCADE` ou `RESTRICT`. Por exemplo, para remover o esquema de um banco de dados `ESCOLA` e todas as suas tabelas e demais elementos, a opção `CASCADE` deve ser utilizada. Nesse mesmo exemplo, se a opção `RESTRICT` for utilizada, o esquema somente será removido se não possuir elementos. Caso tenha elementos, o comando não será executado. Por isso, para usar a opção `DROP RESTRICT`, o usuário precisa remover previamente cada elemento do esquema, para somente depois poder remover o próprio esquema (ELMASRI; NAVATHE, 2011). Veja as demonstrações do uso do comando `DROP` a seguir (ELMASRI; NAVATHE, 2011).

```
DROP TABLE Alunos CASCADE;
DROP TABLE Alunos RESTRICT;
```

O comando DROP TABLE exclui todos os registros na tabela, assim como remove a definição de tabela do catálogo. Caso seja necessário excluir apenas os registros, mantendo a definição de tabela para outra forma de uso, o comando DELETE deve ser utilizado, e não DROP TABLE. O comando DROP pode ser utilizado também para descartar outros tipos de elementos de esquema nomeados, como domínios e restrições (ELMASRI; NAVATHE, 2011). Segundo Ramarkrishnan e Gehrke (2011), caso seja necessário excluir apenas uma visão do banco de dados, o comando DROP VIEW pode ser aplicado.

Outro comando utilizado para alteração de esquema em SQL é o ALTER. Por meio dele, a definição de uma tabela da base ou de outros elementos de esquema nomeados pode ser modificada. Entre as modificações possíveis estão acrescentar ou remover uma coluna, alterar uma definição de coluna ou, ainda, acrescentar ou remover restrições de uma tabela. Abaixo são apresentados dois exemplos de aplicação do comando ALTER (ELMASRI; NAVATHE, 2011).

Exemplo

Incluir um atributo nas tarefas de um colaborador de uma empresa:

```
ALTER TABLE EMPRESA.COLABORADOR
ADD COLUMN Tarefa VARCHAR(15);
```

Remover o endereço do cadastro do colaborador:

```
ALTER TABLE EMPRESA.COLABORADOR
DROP COLUMN Endereco CASCADE;
```

A SQL também utiliza o conceito de **catálogo**, que se trata de uma coleção nomeada de esquemas em um ambiente SQL. O ambiente SQL é, de forma resumida, uma instalação de um SGBD relacional compatível com SQL em um sistema operacional de um computador. O catálogo contém um esquema especial, chamado de `INFORMATION_SCHEMA`, que possui informações sobre todos os esquemas do catálogo, assim como todas as descrições de elementos. Restrições de integridade podem ser definidas entre as relações somente se estiverem nos esquemas dentro do mesmo catálogo (ELMASRI; NAVATHE, 2011).

Segundo Ramarkrishnan e Gehrke (2011), quando são necessárias informações sobre uma tabela, elas são obtidas por meio do catálogo do sistema. Assim, no nível de implementação, sempre que um SGBD necessitar descobrir o esquema de uma tabela de catálogo, o código responsável por recuperar essas informações precisará ser manipulado de maneira especial. Portanto, as tabelas de catálogo podem ser consultadas da mesma maneira que qualquer outra tabela do banco de dados, utilizando-se a linguagem de consulta do SGBD. Da mesma forma, todas as técnicas disponíveis para a implementação e o gerenciamento de tabelas do banco de dados são aplicadas também às tabelas de catálogos.

3 Restrições em SQL

Em um banco de dados relacional, existem muitas relações, que podem ser organizadas de diferentes maneiras. Dessa forma, o estado do banco inteiro depende dos estados de todas as suas relações. No geral, um grande número de **restrições** (*constraints*) é criado sobre os valores em um estado do banco de dados. Essas restrições podem ser divididas em três grupos principais (ELMASRI; NAVATHE, 2011):

- as que são inerentes ao modelo de dados e, por esse motivo, são chamadas de **restrições inerentes baseadas no modelo** ou **restrições implícitas**;
- as que podem ser definidas diretamente nos esquemas do modelo de dados, chamadas de **restrições baseadas em esquema** ou **restrições explícitas**; e
- as que não podem ser definidas diretamente nos esquemas do modelo de dados e, por isso, são impostas pelos programas de aplicação, sendo chamadas de **restrições baseadas na aplicação** ou **restrições semânticas**.

A restrição de que uma relação não pode ter linhas duplicadas é uma restrição implícita, por exemplo. Já as restrições que podem ser definidas no esquema do modelo relacional na DDL são explícitas. Por sua vez, as restrições semânticas são mais gerais, relacionadas ao significado e ao comportamento dos atributos. Essas restrições dificilmente são expressas e impostas no modelo de dados; geralmente são verificadas nos programas de aplicação que executam as atualizações no banco de dados. As restrições explícitas podem ser restrições de domínio, de chave, sobre valores nulos, de integridade e de integridade referencial (ELMASRI; NAVATHE, 2011). Veja a seguir exemplos de restrições na criação de tabelas.

Exemplo

Veja a seguir a criação da tabela `Estudantes`, em que o `id_aluno` deve ser único e não nulo, o nome e o *login* também devem ser não nulos, e a idade, além de ser não nula, é checada para confirmar se é maior do que 10 anos.

```
CREATE TABLE Estudantes(
  id _ aluno CHAR(20) UNIQUE NOT NULL,

  nome CHAR(30) NOT NULL,
  login CHAR (20) NOT NULL,
  idade INTEGER NOT NULL CHECK (idade > 10)
  );
```

A criação da tabela `Estudantes` também pode ser como apresentada a seguir, visto que definir a chave primária em `id_aluno` é o mesmo que dizer que o seu valor deve ser único e não pode ser nulo.

```
CREATE TABLE Estudantes(
  id _ aluno CHAR(20) PRIMARY KEY,

  nome CHAR(30) NOT NULL,
  login CHAR (20) NOT NULL,
  idade INTEGER NOT NULL CHECK (idade > 10)
  );
```

Conforme Ramarkrishnan e Gehrke (2011), as restrições de domínio, de chave primária e de chave estrangeira são parte fundamental do modelo de dados relacional. Por esse motivo, recebem atenção especial da maioria dos sistemas comerciais. Entretanto, às vezes, é preciso definir restrições mais gerais. Por exemplo, no exemplo anterior, pode-se exigir que, em um banco de dados com a tabela `Estudantes`, a idade dos estudantes esteja dentro de determinado intervalo; assim, o SGBD não permitirá a inserção ou a atualização de informações que violarem essa restrição. Esse tipo de restrição é útil para que se evite erro na entrada dos dados. Ainda no exemplo dos estudantes, se a restrição disser que os estudantes precisam ter mais de 10 anos, ela pode ser considerada uma restrição de domínio estendida, pois não está sendo definido apenas o tipo de dados (como `integer`), mas também o conjunto de valores permitidos para a idade.

De forma resumida, as **restrições de domínio** especificam a forma que o valor de cada atributo deve assumir, como o tipo de dados ou um intervalo de valores possíveis. Já as **restrições de chave** garantem que duas tuplas, em qualquer estado da relação, não tenham valores idênticos para todos os atributos; para isso, define-se que uma das colunas de cada tabela seja a chave nas relações com as demais tabelas do banco de dados. Por exemplo, em um cadastro de colaboradores de uma empresa, o `CPF` pode ser definido como a chave da tabela `COLABORADORES`, pois dois colaboradores não podem ter o mesmo número de `CPF`. Já a restrição sobre valores nulos é utilizada para definir se esses valores serão permitidos ou não, o que é considerada uma **restrição de integridade**. Pode ser aplicada no exemplo da tabela `COLABORADORES`, para definir que o atributo `Nome` precisa ter um valor válido, diferente de `NULL`; portanto, esse atributo é restrito a ser `NOT NULL` (ELMASRI; NAVATHE, 2011).

Link

Você viu, ao longo deste capítulo, diferentes conceitos de restrições em SQL. Acessando o *link* a seguir, você pode conferir conceitos adicionais sobre restrições de chave primária e estrangeira, que podem ser empregadas em diferentes versões de bancos de dados em SQL.

https://qrgo.page.link/9VXZg

As restrições de integridade são definidas quando uma relação é criada e são verificadas quando uma relação é modificada. Portanto, se um comando de inserção, exclusão ou atualização causa alguma violação nas restrições, ele é rejeitado. Veja a seguir exemplos de inserção que não respeitam restrições (RAMARKRISHNAN; GEHRKE, 2011).

Exemplo

Inserção que viola restrição de chave primária, pois já existe um estudante com `id-estudante 52699`:

```
INSERT
INTO Estudantes (id-estudante, nome, login, idade, media)
VALUES (52699, 'Enzo', 'enzo@email.com', 16, 5,5);
```

Inserção que viola restrição de que a chave primária não pode ser nula:

```
INSERT
INTO Estudantes (id-estudante, nome, login, idade, media)
VALUES (null, 'Enzo', 'enzo@email.com', 16, 5,5);
```

De acordo com Elmasri e Navathe (2011, p. 47):

> A restrição de integridade de entidade afirma que nenhum valor de chave primária pode ser NULL. Isso porque o valor da chave primária é usado para identificar tuplas individuais em uma relação. Ter valores NULL para a chave primária implica que não podemos identificar algumas tuplas. Por exemplo, se duas ou mais tuplas tivesse NULL para suas chaves primárias, não conseguiríamos distingui-las ao tentar referenciá-las por outras relações. As restrições de chave e as restrições de integridade são especificadas sobre relações individuais. A restrição de integridade referencial é especificada entre duas relações e usada para manter a consistência entre tuplas nas duas relações.

As restrições de integridade são incluídas na linguagem de definição de dados, pois ocorrem na maioria das aplicações de banco de dados. Existem outros diversos tipos e exemplos de restrições em SQL, entre eles estão as **restrições de integridade semântica**, que podem ser impostas em um banco de dados relacional. São exemplos desse tipo de restrição: o salário de um colaborador não pode ser maior do que o salário do seu supervisor; o número máximo de horas que um colaborador pode trabalhar sem intervalo para refeição é seis horas. Esse tipo de restrição pode ser definido e imposto em programas de aplicação que atualizam o banco de dados. Em SQL, os comandos `CREATE ASSERTION` (criar afirmações) e `CREATE TRIGGER` (criar gatilhos) podem ser utilizados para esse fim (ELMASRI; NAVATHE, 2011).

Referências

ELMASRI, R.; NAVATHE, S. B. *Sistemas de banco de dados*. 6. ed. São Paulo: Pearson, 2011.

PIRES, C. E. M. *Fundamento de SQL com ênfase em Postgres*. Brasília, DF: AgBook, 2017.

RAMARKRISHNAN, R.; GEHRKE, J. *Sistemas de gerenciamento de banco de dados*. 3. ed. Porto Alegre: AMGH, 2011.

WATSON, J. *OCA Oracle Database 11g*: Administração I (Guia do Exame 1Z0-052): preparação completa para o exame. Porto Alegre: Bookman, 2010.

Fique atento

Os *links* para *sites* da *web* fornecidos neste capítulo foram todos testados, e seu funcionamento foi comprovado no momento da publicação do material. No entanto, a rede é extremamente dinâmica; suas páginas estão constantemente mudando de local e conteúdo. Assim, os editores declaram não ter qualquer responsabilidade sobre qualidade, precisão ou integralidade das informações referidas em tais *links*.

Encerra aqui o trecho do livro disponibilizado para esta Unidade de Aprendizagem. Na Biblioteca Virtual da Instituição, você encontra a obra na íntegra.

Conteúdo:

SAGAH
SOLUÇÕES EDUCACIONAIS INTEGRADAS

Dica do Professor

As consultas DML são empregadas na manutenção e nas atualização das informações armazenadas em bancos de dados que utilizam SQL. Por este motivo, é fundamental conhecer as funções básicas da manipulação de dados, como inserir, atualizar e excluir informações.

Nesta Dica do Professor, você vai ver alguns exemplos de aplicação dos recursos de DML em SQL.

Aponte a câmera para o código e acesse o link do conteúdo ou clique no código para acessar.

Exercícios

1) **A SQL é uma linguagem que tem instruções para definição, consulta, atualização e eliminação de dados.**

Entre os subconjuntos presentes na SQL, assinale a alternativa na qual está aquele que é utilizado para definir controles de acesso pelos usuários ao banco de dados.

A) *Data Definition Language – DDL.*

B) *Data Control Language – DCL.*

C) *Data Manipulation Language – DML.*

D) *Restrições Inerentes Baseadas no Modelo – RIBM.*

E) *Transactional Control Language – TCL.*

2) **A instrução CREATE TABLE é empregada para criar uma nova tabela em um banco de dados SQL.**

Entre as opções a seguir, assinale a que utiliza corretamente a instrução CREATE TABLE para criar uma tabela de produtos com a definição de uma chave primária.

A) CREATE VIEW Produto(

id_produto INT PRIMARY KEY,

nome_produto VARCHAR(35),

valor_produto NUMERIC (16,2),

);

B) CREATE Produto(

id_produto INT KEY,

nome_produto VARCHAR(35),

valor_produto NUMERIC (16,2),

);

C) CREATE TABLE Produto(

id_produto INT PRIMARY KEY,

nome_produto VARCHAR(35),

valor_produto NUMERIC (16,2),

);

D) ALTER TABLE Produto(

id_produto INT,

nome_produto VARCHAR(35) PRIMARY KEY,

valor_produto INT,

);

E) CREATE UNIQUE Produto(

id_produto INT PRIMARY KEY,

nome_produto VARCHAR(35),

valor_produto NUMERIC (16,2),

);

3) A SQL tem comandos para restrição de privilégios de acesso ao banco de dados. Veja a seguinte instrução:

GRANT create_table
TO usuario_pedro;

A instrução acima, que utilizou o comando GRANT, foi criada para:

A) autorizar o privilégio de criação de tabelas para 'usuario_pedro'.

B) revogar o privilégio de criação de tabelas para 'usuario_pedro'.

C) revogar o privilégio de criação de domínios para 'usuario_pedro'.

D) autorizar o privilégio de criação de consultas para 'usuario_pedro'.

E) revogar o privilégio de criação de visões para 'usuario_pedro'.

4) A SQL disponibiliza comandos de evolução de esquema, utilizados para alterar um esquema, o que significa acrescentar ou remover tabelas, atributos, restrições, entre outros elementos.

Um dos comandos úteis neste caso é o DROP. Sobre o uso do comando DROP, é correto afirmar que:

A) o comando DROP com a opção CASCADE serve para remover todos os registros de uma tabela de banco de dados, mantendo a definição da tabela.

B) o comando DROP TABLE serve para remover todos os registros de uma tabela de banco de dados, mantendo a definição da tabela.

C) o comando DROP TABLE serve para remover o esquema de um banco de dados e todas as suas tabelas e demais elementos.

D) o comando DROP com a opção CASCADE serve para remover o esquema de um banco de dados e todas as suas tabelas.

E) o comando DROP com a opção RESTRICT serve para remover o esquema de um banco de dados e todas as suas tabelas com seus registros.

5) **O comando ALTER é empregado para alterar a definição de uma tabela da base de dados ou de outros elementos de esquema.**

Analise as seguintes instruções em SQL e assinale a opção na qual o comando ALTER é utilizado corretamente para incluir um atributo na tabela 'PRODUTOS':

A) ALTER TABLE PRODUTOS

ALTER ATRIBUTO fornecedor VARCHAR(25);

B) ALTER TABLE PRODUTOS

DROP COLUMN fornecedor VARCHAR(25);

C) ALTER TABLE PRODUTOS

DROP COLUMN fornecedor CASCADE;

D) ALTER TABLE PRODUTOS

DROP COLUMN fornecedor RESTRICT;

E) ALTER TABLE PRODUTOS

ADD COLUMN fornecedor VARCHAR(25);

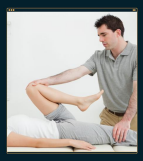
Na prática

O cuidado com a construção das tabelas de um banco de dados SQL reflete em sua eficiência e usabilidade quando estiver pronto. Por este motivo, é importante conhecer as possibilidades de definições e restrições de atributos que podem ser utilizadas em SQL na criação e na alteração das tabelas.

Nesta Na Prática, você vai conhecer o caso de uma clínica de fisioterapia que utilizou SQL para criar as tabelas do seu banco de dados, por meio de restrições.

RESTRIÇÕES EM SQL NA KHORPUS FISIOTERAPIA

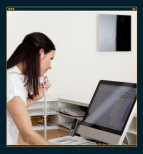
A SQL possibilita a criação de tabelas de acordo com a necessidade do usuário, assim como permite que dados previamente cadastrados sejam alterados ou excluídos. Neste Na Prática, você vai conhecer o caso de uma clínica de fisioterapia e as funções de restrição em suas informações sobre clientes e agendamento de atendimentos.



A clínica Khorpus Fisioterapia atua há pouco tempo no mercado. Ela oferece os seguintes serviços: *pilates*, reeducação postural global (RPG) e fisioterapia. A clínica utiliza um sistema de informação para agendamento dos atendimentos e avaliações, assim como um banco de dados em SQL.

A Khorpus conta com cinco funcionários, destes, um recepcionista, responsável pelo atendimento prévio aos clientes, um profissional de Educação Física e quatro fisioterapeutas.

Semanalmente, a clínica organiza os horários de atendimento de acordo com a demanda. Quando um novo cliente agenda um atendimento, é necessário que ele preencha um cadastro prévio, com seus dados pessoais e do seu convênio ou plano de saúde, se for o caso. Para evitar erros nos atendimentos, foi necessário criar as tabelas do banco de dados com algumas restrições.



Entre as tabelas do banco de dados da clínica estão:

Cliente
CPF_cliente
Nome_cliente
Data_nascimento
Sexo
Convenio
Telefone

Especialista
CPF_esp
Nome_esp
Especialidade
Horario_trabalho
Telefone

Consulta
Consulta
Data
Hora
CPF_cliente
CPF_esp

A instrução para criação da TABELA DE CLIENTES foi:

Cpf_cliente é a Chave Primária

```
CREATE TABLE Cliente(
Cpf_cliente VARCHAR(11) PRIMARY KEY,
Nome_cliente VARCHAR(100) NOT NULL,
Data_nascimento DATE NOT NULL,
Sexo CHAR(1) NOT NULL,
Convenio VARCHAR(20) CHECK (convenio = "Sempre Viva"),
Telefone VARCHAR(10) NOT NULL,
);
```

Nome_cliente, Data_nascimento, Sexo e Telefone não podem ser nulos.

Convênio pode ser nulo, mas se for preenchido, é checado, pois de acordo com o convênio existe uma porcentagem de desconto.

A instrução para criação da TABELA DOS ESPECIALISTAS (fisioterapeutas e educador físico) foi a seguinte:

Cpf_esp é a Chave Primária

```
CREATE TABLE Especialista(
Cpf_esp VARCHAR(11) PRIMARY KEY,
Data_nascimento DATE NOT NULL,
Especialidade VARCHAR(25) NOT NULL,
Horario_trabalho INT NOT NULL,
Telefone VARCHAR(10) NOT NULL,
);
```

Nome_esp, Especialidade, Horário e Telefone não podem ser nulos.

A instrução para criação da TABELA DE CONSULTAS foi a seguinte:

Cod_consulta é a Chave Primária e seu preenchimento é de auto incremento

```
CREATE TABLE Consulta(
Cod_consulta INT IDENTITY (1,1) PRIMARY KEY,
Data DATE NOT NULL,
Hora TIME NOT NULL,
Especialidade VARCHAR(25) NOT NULL,
Cpf_cliente VARCHAR(11) NOT NULL,
Cpf_esp VARCHAR(11) NOT NULL,
FOREIGN KEY (Cpf_cliente) REFERENCES cliente,
FOREIGN KEY (Cpf_esp) REFERENCES especialista
);
```

Cpf_cliente e Cpf_esp são as Chaves Externas

Aponte a câmera para o código e acesse o link do conteúdo ou clique no código para acessar.

Saiba mais

Para ampliar o seu conhecimento a respeito desse assunto, veja abaixo as sugestões do professor:

Exemplos e conceitos SQL

Neste *link*, você vai ver a documentação oficial do MySQL sobre restrições CHECK, com exemplos do seu uso em banco de dados.

Aponte a câmera para o código e acesse o link do conteúdo ou clique no código para acessar.

Restrições exclusivas e restrições de verificação

Para aprofundar seus conhecimentos sobre a criação de tabelas e o uso de restrições em SQL, acesse o *link* a seguir.

Aponte a câmera para o código e acesse o link do conteúdo ou clique no código para acessar.

Criação de tabelas e consultas com SQL

Para verificar mais exemplos sobre criação de tabelas e consultas em SQL, consulte o seguinte livro. Dê atenção especial aos conceitos presentes entre as páginas 81 e 89, no **Capítulo 4**, que tratam sobre as instruções CREATE TABLE e SELECT.

Conteúdo interativo disponível na plataforma de ensino!

Esquemas e catálogos de banco de dados

O seguinte livro traz mais conceitos sobre esquemas e catálogos de banco de dados. Dê atenção especial aos conceitos presentes entre as páginas 10 e 12.

Conteúdo interativo disponível na plataforma de ensino!