Curso de Operador de Tecnologia da Informação e Comunicações

Banco de dados

Linguagem de Consulta

Objetivo de Aprendizado

Conhecer os principais conceitos relacionados ao SQL e os principais comandos.

A linguagem SQL serve para a comunicação com o banco de dados. Todas as linguagens de programação se integra ao banco de dados utilizando esse tipo de comunicação porque ela é universal e serve em qualquer gerenciador de banco de dados.

Definição

SQL é sigla inglesa de “Structured Query Language” que significa, em Português, Linguagem de Consulta Estruturada, uma linguagem padrão de gerenciamento de dados que interage com os principais bancos de dados baseados no modelo relacional.

Alguns dos principais sistemas que utilizam SQL são: MySQL, Oracle, Firebird, Microsoft Access, PostgreSQL (código aberto), HSQLDB (código aberto e escrito em Java).

A linguagem SQL surgiu em 1974 e foi desenvolvida nos laboratórios da IBM como interface para o Sistema Gerenciador de Banco de Dados Relacional (SGBDR) denominado SYSTEM R. Esse sistema foi criado com base em um artigo de 1970 escrito por Edgar F. Codd.

Outras linguagens do gênero surgiram, mas a SQL tornou-se a mais utilizada. A criação de um padrão para a SQL foi realizada em 1986 pelo American National Standard Institute (ANSI) e em 1987 pela International Organization for Standards (ISO).

SQL é uma linguagem essencialmente declarativa. Isso significa que o programador necessita apenas indicar qual o objetivo pretendido para que seja executado pelo SGBDR.

Alguns dos principais comandos SQL para manipulação de dados são: INSERT (inserção), SELECT (consulta), UPDATE (atualização), DELETE (exclusão). SQL possibilita ainda a criação de relações entre tabelas e o controle do acesso aos dados.

A linguagem **SQL é dividida em 4 agrupamentos** de acordo com o tipo de operação a ser executada no banco de dados:

- DML (Data Manipulation Language, ou Linguagem de Manipulação de Dados e português);
- DDL (Data Definition Language, ou Linguagem de Definição de Dados em português);
- DCL (Data Control Language, ou Linguagem de Controle de Dados em português); e
- DTL (Data Transaction Language, ou Linguagem de Transação de Dados em português).

Alguns autores classificam também uma divisão da linguagem para consultas, a DQL (Data Query Language, Linguagem de Consulta de Dados), que tem apenas um comando (SELECT), porém é mais comum encontrar este comando como integrante da DML, juntamente com os comandos INSERT, UPDATE e DELETE. Vejamos os comandos SQL de cada agrupamento.

DQL – DATA QUERY LENGUAGE

Define o comando utilizado para que possamos consultar (SELECT) os dados armazenados no banco

SELECT: O principal comando da SQL, o comando select é utilizado para efetuar consultas no banco de dados.
Exemplo: SELECT ID, NOME FROM CLIENTE;

DML - DATA MANIPULATION LANGUAGE

DML (Linguagem de Manipulação de Dados) é o subconjunto mais utilizado da linguagem SQL, pois é através da DML que **operamos** sobre os dados dos bancos de dados com instruções de **inserção, atualização e exclusão**. Os comandos SQL desse subconjunto são:

INSERT: utilizado para inserir registros (tuplas), em uma tabela.
Exemplo: INSERT into CLIENTE(ID, NOME) values(1,'José');

UPDATE: utilizado para alterar valores de uma ou mais linhas (tuplas) de uma tabela.
Exemplo: UPDATE CLIENTE set NOME = 'João' WHERE ID = 1;

DELETE: utilizado para excluir um ou mais registros (tupla) de uma tabela.
Exemplo: DELETE FROM CLIENTE WHERE ID = 1;

Nota: Registro, Linha e Tupla são palavras sinônimas para referenciar a uma linha da tabela.

DDL - DATA DEFINITION LANGUAGE

DDL (Linguagem de Definição de Dados) é o subconjunto da SQL utilizado para gerenciar a estrutura do banco de dados. Com a DDL podemos **criar**, **alterar** e **remover** objetos (tabelas, visões, funções, etc.) no banco de dados. Os comandos deste subconjunto são:

CREATE: utilizado para criar objetos no banco de dados.
Exemplo (criar uma tabela): CREATE TABLE CLIENTE (ID INT PRIMARY KEY, NOME VARCHAR(50));

ALTER: utilizado para alterar a estrutura de um objeto.
Exemplo (adicionar uma coluna em uma tabela existente): ALTER TABLE CLIENTE ADD SEXO CHAR(1);

DROP: utilizado para remover um objeto do banco de dados.
Exemplo (remover uma tabela): DROP TABLE CLIENTE;

DCL - DATA CONTROL LANGUAGE

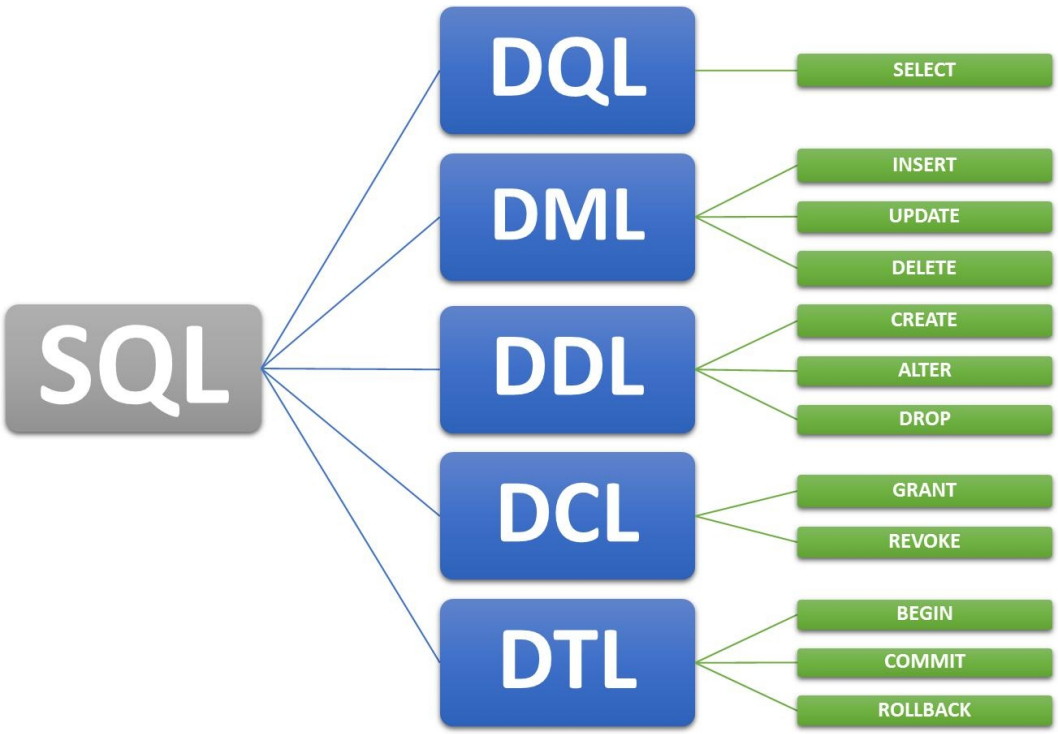
DCL (Linguagem de Controle de Dados) é o subconjunto da SQL utilizado para **controlar o acesso** aos dados, basicamente com dois comandos que permite ou bloqueia o acesso de usuários a dados. Vejamos estes comandos:

GRANT: Autoriza um usuário a executar alguma operação.
Exemplo (dar permissão de consulta na tabela cliente para o usuário carlos): GRANT select ON cliente TO carlos;

REVOKE: Restringe ou remove a permissão de um usuário executar alguma operação. Exemplo (não permitir que o usuário carlos crie tabelas no banco de dados): REVOKE CREATE TABLE FROM carlos;

DTL - DATA TRANSACTION LANGUAGE

DTL (Linguagem de controle de transações) é o subconjunto da SQL que fornece mecanismos para **controlar transações** no banco de dados. São 3 comandos: iniciar uma transação (BEGIN TRANSACTION), efetivar as alterações no banco de dados (COMMIT) e cancelar as alterações (ROLLBACK).



COMANDO INSERT

Modelo de sintaxe do comando INSERT

INSERT INTO nome_tabela (lista-de-campos) VALUES (lista_dados)

ou

INSERT INTO nome_tabela VALUES (lista_dados)

Onde:

- **Nome_tabela**: nome da tabela no qual será inserido os dados.
- **Lista-de-campos**: nome das colunas que receberão os valores.
- **Lista-dados**: valores que serão inseridos na tabela. Estes campos devem estar na mesma ordem descrita em lista-de-campos, todos separados por vírgula. Se for utilizado um comando SELECT o mesmo deve retornar a mesma quantidade de colunas com os mesmos tipos de dados especificados em lista-de-campos.

EXEMPLOS

```
INSERT INTO EMPREGADOS(CODIGO, NOME, SALARIO, SECAO)
VALUES(1, "HELBERT CARVALHO", 1.500, 1)
ou
INSERT INTO EMPREGADOS VALUES(1,"HELBERT CARVALHO",1500,1)
```

Na segunda opção foi omitida a declaração dos campos. Essa sintaxe funciona somente se for repassado valores para **todas as colunas**. Podemos também passar valores através de um comando SELECT, conforme abaixo:

(Insert com valores provenientes de um select)

```
INSERT INTO EMPREGADOS(CODIGO,NOME, SALARIO, SECAO)
    SELECT CODIGO,NOME,SALARIO, SECAO
    FROM EMPREGADOS_FILIAL
    WHERE DEPARTAMENTO = 2
```

Neste comando todos os empregados da tabela EMPREGADOS_FILIAL foram cadastrados na tabela EMPREGADOS. Se o nome dos campos não for citado no comando INSERT, o SELECT deverá retornar valores compatíveis para todos os campos disponíveis na tabela de destino. Nesse caso específico, cadastramos somente os registro que tinham o campo “DEPARTAMENTO=2”, sendo estes registros cadastrados na tabela EMPREGADOS

COMANDO UPDATE

Modelo de sintaxe do comando UPDATE

```
UPDATE nome_tabela
SET CAMPO = "novo_valor"
WHERE CONDIÇÃO
```

Onde:

- **Nome_tabela**: nome da tabela que será modificada
- **Campo**: campo que terá seu valor alterado
- **Novo_valor**: valor que substituirá o antigo dado cadastrado em campo
- **Where**: Se não for informado, a tabela intera será atualizada
- **Condição**: regra que impõe condição para execução do comando

EXEMPLOS

```
UPDATE DEPARTAMENTO
SET SALARIO = 1000
WHERE CODIGODEP = 1
```

No trecho acima, todos os colaboradores que fazem parte do departamento 1 terá o salário alterado para 1000.

Update em mais de um campo

```
UPDATE DEPARTAMENTO
SET NOME = "HELBERT CARVALHO", SALARIO = 1000
WHERE CODIGO = 1
```

Neste exemplo alteramos mais de um campo de uma vez. Podemos combinar o comando SELECT com UPDATE.

Outro exemplo de uso do comando update

```
UPDATE EMPREGADOS
SET SALARIO = salario * 1.1
WHERE SALARIO = (SELECT MIN(salario) FROM EMPREGADOS)
```

No exemplo acima, os funcionários de menor salário receberão aumento de 10%.

Update passando select como valor

```
UPDATE EMPREGADOS
SET SALARIO = (SELECT MAX(salario) FROM EMPREGADOS)
WHERE DEPARTAMENTO = 5
```

O comando SELECT também pode ser utilizado na atribuição de valor ao campo. No exemplo anterior, para que fosse passado o parâmetro do maior salário, foi passado um SELECT que pudesse obter essa informação do banco de dados.

COMANDO DELETE

Modelo de sintaxe do comando DELETE

DELETE FROM nome_tabela

WHERE condição

Onde:

- **Nome_tabela**: nome da tabela que será modificada
- **Where**: cláusula que impõe uma condição sobre a execução do comando

Exemplo de uso do comando delete

DELETE FROM EMPREGADOS

WHERE CODIGO = 125

COMANDO SELECT

Modelo de sintaxe do comando SELECT

SELECT * FROM nome_tabela;

Onde:

- nome_tabela: é o nome propriamente dito de uma determinada tabela;

Selecionando colunas específicas

SELECT numnota, dtsaida, vltotal FROM pcnfsaid;

Definindo apelido para campos selecionados

A atribuição de apelidos aos campos selecionados tem como objetivo facilitar o entendimento das informações exibidas nos resultados das pesquisas.

Existem 4 maneiras diferentes de se atribuir apelidos aos campos selecionados, são elas:

- select codprod as codigo from pcprodut;
- select codprod codigo from pcprodut;
- select codprod as “Codigo do Produto” from pcprodut;
- select codprod “Codigo do Produto” from pcprodut;

Observe que a única diferença entre os exemplos 1 e 2 é a utilização ou não do termo “as”. A utilização desse termo é opcional, e tem como objetivo melhorar o entendimento das seleções efetuadas. Esta diferença também pode ser observda entre os exemplos 3 e 4.

Observe que enquanto nos exemplos 1 e 2 os apelidos são escritos sem a utilização de aspas(“), os exemplos 3 e 4 fizeram a utilização desse caracter. A utilização de aspas(“) é necessária quando o apelido a ser atribuído contém espaços em branco.

Concatenar

O comando concatenar consiste em unificar as informações de dois campos distintos da tabela utilizada, de forma que sejam exibidas em apenas um campo. Também podem ser utilizadas strings que não sejam informações existentes nas tabelas.

O código abaixo seleciona o campo especie, concatenando com o campo seria, seleciona também o campo numnota, de todos os registros da tabela pcnfsaid.

Exemplo de concatenação

```
SELECT especie || serie, numnota FROM pcnfsaid;
```

O código abaixo mostra como concatenar a string “Emitido em:” com o valor do campo dtemissao, com a string “ com vencimento em:”, com o valor do campo dtvenc, onde os campos são da tabela pcprest, utilizando todos os registros da tabela.

Outro exemplo de concatenação

```
SELECT “Emitido em:” || “dtemissao:” || “ com
vencimento em:” || dtvenc FROM pcprest;
```

Distinct

O comando “distinct” consiste em exibir as informações da tabela utilizada de maneira distinta, de forma que informações não sejam exibidas de maneira redundante.

Usando Distinct

```
SELECT distinct codcli FROM pcnfsaid;
```

O código acima mostra como selecionar o campo codcli da tabela pcnfsaid de maneira distinta, ou seja, cada informação do campo codcli será exibido apenas uma vez, não importando quantas outras vezes essa informação esteja presente em outros registros da tabela utilizada. Serve para descobrirmos quantos valores distintos existem em um determinado campo.

RESTRINGINDO DADOS COM (WHERE)

O comando de restrição “where” é o comando de restrição mais utilizado. O comando “where” quer dizer “onde”, ou seja, com a utilização do comando “where”, serão exibidas as informações solicitadas, onde as mesmas obedecerem as restrições informadas.

Condições de Comparação (=, >, >=, <, <=, <>, in, like, between, is null)

“=“ é igual

O código abaixo seleciona todos os campos dos registros da tabela pcnfsaid onde o valor do campo numnota for igual a 50.

Usando where com =

```
SELECT * FROM pcnfsaid WHERE numnota = 50;
```

“>” é maior

O código abaixo seleciona todos os campos dos registros da tabela pcnfsaid onde o valor do campo numnota for maior que 45.

Usando where com >

```
SELECT * FROM pcnfsaid WHERE numnota > 45;
```

“>=“ é maior ou igual

O código abaixo seleciona todos os campos dos registros da tabela pcnfsaid onde o valor do campo numnota for maior ou igual a 50.

Usando where com >

```
select * from pcnfsaid where numnota >=50;
```

“<” é menor

O código abaixo seleciona todos os campos dos registros da tabela pcnfsaid onde o valor do campo numnota for menor que 10.

Listagem 10: Usando Where com <

select * from pcnfsaid where numnota < 10;

“<=“ é menor ou igual

O código abaixo seleciona todos os campos dos registros da tabela pcnfsaid onde o valor do campo numnota for menor ou igual a 5.

Listagem 11: Usando where com <=

select * from pcnfsaid where numnota <= 5;

“<>” é diferente

O código abaixo seleciona todos os campos dos registros da tabela pcnfsaid onde o valor do campo numnota for diferente de 2.

Listagem 12: Usando where com diferente(<>)

select * from pcnfsaid where numnota <> 2;

“in” é na lista

O código abaixo seleciona todos os campos dos registros da tabela pcnfsaid onde o valor do campo numnota pertencer à lista 1,3,5,6,8.

Listagem 13: Usando where com In

select * from pcnfsaid where numnota in

(1,3,5,6,8); “like” é como, parecido

O código abaixo seleciona todos os campos dos registros da tabela pcnfsaid, onde o valor do campo numnota começar com o número 1, não importando quantos e quais números venham na sequência.

Listagem 14: Usando Like

select * from pcnfsaid where numnota like “1%”;

O código abaixo seleciona todos os campos dos registros da tabela pcnfsaid, onde o valor do campo numnota começar com o número 1, e que tenham apenas um número na sequência, com qualquer valor.

Listagem 15: Outro exemplo de like

select * from pcnfsaid where numnota like “1_”;

O código abaixo seleciona todos os campos dos registros da tabela pcnfsaid, onde o valor do campo numnota começar com o número 1, e que tenham apenas dois números na sequência, com qualquer valor.

Listagem 16: Mais um exemplo de like

select * from pcnfsaid where numnota like “1__”;

O código abaixo seleciona todos os campos dos registros da tabela pcnfsaid, onde o primeiro número do campo numnota tiver qualquer valor, o segundo número for 1, não importando quantos e quais números venham na sequência.

Listagem 17: Outra opção de uso do like

select * from pcnfsaid where numnota like “_1%”; “between” é

a cláusula que coloca a busca entre dois valores

O código abaixo seleciona todos os campos dos registros da tabela pcnfsaid, onde o valor do campo numnota estiver entre os valores 1 e 10, incluindo os valores 1 e 10.

Listagem 18: Cláusula between

select * from pcnfsaid where numnota between 1 and 10;

Obs.: na condição “between” os valores informados como limites são considerados nos resultados.

“is null” quer dizer é nulo

O código abaixo seleciona todos os campos dos registros da tabela pcnfsaid, onde o valor do campo comissao for nulo.

Listagem 19: Usando is null

select * from pcnfsaid where comissao is null;

Obs.: Campo nulo é o campo que não possui nenhum valor atribuído. Campos com um caractere de espaço ou 0(zero) tem valor atribuído. Campo nulo é campo vazio.

https://www.devmedia.com.br/sql-basico/28877

Referência

https://www.significados.com.br/sql/

https://dicasdeprogramacao.com.br/o-que-e-sql/

https://www.devmedia.com.br/comandos-basicos-em-sql-insert-update-delete-e-select/37170 https://www.devmedia.com.br/guia/guia-completo-de-sql/38314