Linguagem SQL

SENAI

Criação e manipulação de banco de dados

O Modelo Relacional prevê, desde sua concepção, a existência de uma linguagem baseada em caracteres que suporte a definição do esquema físico (tabelas, restrições, etc.), e sua manipulação (inserção, consulta, atualização e remoção)

A Linguagem SQL pode ser dividida em 5 conjuntos de comandos

Linguagem de definição de dados (DDL - Data Definition Language)

- Comandos para criação e manutenção de objetos do banco de dados: CREATE, ALTER, DROP, RENAME e TRUNCATE

Linguagem de manipulação de dados (DML - Data Manipulation Language)

- Comandos para inserções (INSERT), atualizações (UPDATE) e exclusões (DELETE)

Recuperação de dados (DQL – Data Query Language) (Atualmente faz parte do DML)

- Comando SELECT

Linguagem para controle de transações (DTL – Data Transaction Language)

- COMMIT, ROLLBACK e SAVEPOINT

Linguagem para controle de acesso a dados (DCL – Data Control Language)

- GRANT e REVOKE

DDL
Data Definition Language

Criando uma database

Sintaxe

CREATE DATABASE *nome_do_banco;*

Exemplo

CREATE DATABASE *escola;*

Selecionando uma database

Antes de começar a criar suas tabelas ou realizar consultas, é necessário selecionar a database que será utilizada.

Sintaxe

```
USE nome_do_banco;
```

Exemplo

```
USE escola;
```

Criando tabelas

Sintaxe

```
CREATE TABLE nome_tabela
(primeiro_campo  tipo_do_campo(tamanho),
 segundo_campo tipo_do_campo(tamanho)
);
```

Exemplo

```
CREATE TABLE alunos
(matricula          int,
 nome varchar(45),
 turma_idturma int
);
```

Criando tabelas com CHAVE PRIMARIA

Sintaxe

```
CREATE TABLE nome_tabela
(primeiro_campo  tipo_do_campo(tamanho),
 segundo_campo tipo_do_campo(tamanho),
 PRIMARY KEY (`primeiro_campo`)
);
```

Exemplo

```
CREATE TABLE alunos
(matricula          INT,
 nome VARCHAR(45),
 turma_idturma int,
 PRIMARY KEY (`matricula`)
);
```

Criando tabelas com CHAVE ESTRANGEIRA

Sintaxe

```
CREATE TABLE nome_tabela
(primeiro_campo     tipo_do_campo(tamanho),
 segundo_campo tipo_do_campo(tamanho),
 PRIMARY KEY (primeiro_campo),
 CONSTRAINT nome_da_regra FOREIGN KEY (campo_desta_tabela)    REFERENCES
nome_da_segunda_tabela(chave_primaria_da_segunda_tabela)
);
```

Exemplo

```
CREATE TABLE alunos
(matricula     INT,
 nome VARCHAR(45),
 turma_idturma int,
 PRIMARY KEY (`matricula`),
 CONSTRAINT fk_alunos_turma FOREIGN KEY (turma_idturma) REFERENCES turma(idturma)
);
```

Incluindo CHAVE ESTRANGEIRA depois da tabela criada

Sintaxe

ALTER TABLE *nome_tabela*
ADD CONSTRAINT *nome_da_regra* **FOREIGN KEY (***campo_desta_tabela***) REFERENCES** *nome_da_segunda_tabela***(***chave_primaria_da_segunda_tabela***);**

Exemplo

ALTER TABLE *nome_tabela*
ADD CONSTRAINT *fk_alunos_turma* **FOREIGN KEY (***turma_idturma***) REFERENCES** *turma***(***idturma***);**

Incluindo novos campos na tabela

Sintaxe

```
ALTER TABLE nome_tabela
ADD novo_campo_1 integer,
ADD novo_campo_2 varchar(45);
```

Exemplo

```
ALTER TABLE alunos
ADD idade integer,
ADD tamanho_uniforme varchar(3);
```

Criando uma VIEW

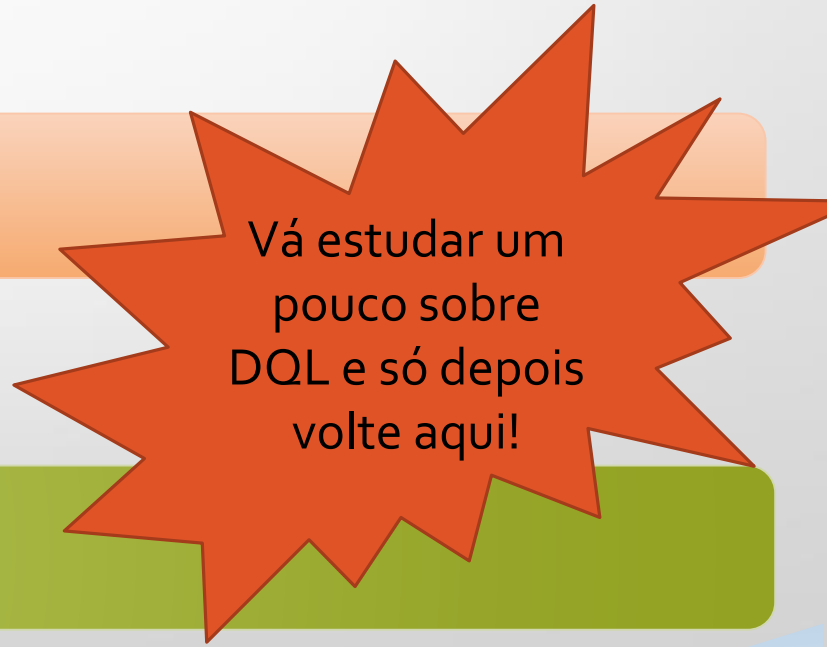
Uma view pode ser definida como uma tabela virtual criada a partir de uma consulta (query) pré-definida.
Ela facilita a visualização das informações do banco de dados

Sintaxe

CREATE VIEW *nome_da_view* **AS**
código_da_consulta;

Exemplo

CREATE VIEW aluno_completo **AS**



DML
Data Manipulation Language

Inserindo registros

Sintaxe

```
INSERT INTO nome_tabela (campo_1, campo_2, campo_3)
 VALUES(valor_1, valor_2, valor_3);
```

Exemplo

```
INSERT INTO alunos (matricula, nome, turma_idturma)
 VALUES(101, "Godofredo", 8);
```

ou

```
INSERT INTO alunos
 VALUES(101, "Godofredo", 8);
```

Alterando Registros

CUIDADO!!!
Se você utilizar o UPDATE sem especificar o WHERE,
você irá alterar todos os registros da tabela.

Sintaxe

```
UPDATE nome_tabela
SET campo_1=valor_1, campo_2=valor_2
WHERE chave_primaria = registro_que_será_alterado;
```

Exemplo

```
UPDATE alunos
SET nome="Diogo Fernando"
WHERE matricula = 101;
```

Excluindo Registros

CUIDADO!!!
Se você utilizar o DELETE sem especificar o WHERE,
você irá apagar todos os registros da tabela.

Sintaxe

DELETE FROM *nome_tabela*
WHERE *chave_primaria = registro_que_será_alterado*;

Exemplo

DELETE FROM *alunos*
WHERE *matricula = 104*;

Comando "SELECT"

Exibindo os dados

Sintaxe

SELECT * **FROM** *nome_tabela*;

SELECT *campo_1, campo_2* **FROM** *nome_tabela*;

Exemplo

SELECT * **FROM** *alunos*;

SELECT *matricula, nome* **FROM** *alunos*;

Filtrando os dados

Sintaxe

SELECT *campo_1, campo_2* **FROM** *nome_tabela*
WHERE *campo_1* **=** *filtro***;**

Exemplo

SELECT *matricula, nome* **FROM** *alunos*
WHERE *matricula* **=** *101***;**

Recuperando informações de 2 ou mais tabelas
INNER JOIN

Sintaxe

```
SELECT tabela1.campo_1, tabela1.campo_2, tabela2.campo_1
FROM nome_tabela_1
INNER JOIN nome_tabela_2
ON tabela1.chave_estrangeira = tabela2.chave_primaria;
```

Exemplo

```
SELECT alunos.matricula, alunos.nome, turma.periodo
FROM alunos
INNER JOIN turma
ON tabela1.chave_estrangeira = tabela2.chave_primaria;
```

Recuperando informações de 2 ou mais tabelas WHERE

Sintaxe

SELECT *tabela1.campo_1, tabela1.campo_2, tabela2.campo_1* **FROM** *nome_tabela_1, nome_tabela_2*
WHERE *tabela1.chave_estrangeira = tabela2.chave_primaria***;**

Exemplo

SELECT *alunos.matricula, alunos.nome, turma.periodo*
FROM *alunos, turma*
WHERE *alunos.turma_idturma= turma.idturma***;**

Usando filtro

Sintaxe

```
SELECT tabela1.campo_1, tabela1.campo_2, tabela2.campo_1
FROM nome_tabela_1, nome_tabela_2
WHERE tabela1.chave_estrangeira = tabela2.chave_primaria
AND tabela2.campo_1 = filtro;
```

```
SELECT tabela1.campo_1, tabela1.campo_2, tabela2.campo_1
FROM nome_tabela_1
INNER JOIN nome_tabela_2
ON tabela1.chave_estrangeira = tabela2.chave_primaria
WHERE tabela2.campo_1 = filtro;
```

Exemplo

```
SELECT alunos.matricula, alunos.nome, turma.periodo
FROM alunos, turma
WHERE alunos.turma_idturma= turma.idturma
AND turma.idturma = 8;
```

Ordenando

Comandos

ASC – Ordem crescente
DESC – Ordem decrecente

Sintaxe

SELECT *tabela1.campo_1, tabela1.campo_2, tabela2.campo_1* **FROM** *nome_tabela_1, nome_tabela_2*
WHERE *tabela1.chave_estrangeira = tabela2.chave_primaria*;
ORDER BY *tabela2.campo1, tabela1.campo_1* **DESC**

Exemplo

SELECT *alunos.matricula, alunos.nome, turma.periodo*
FROM *alunos, turma*
WHERE *alunos.turma_idturma= turma.idturma*
ORDER BY *turma.periodo* **DESC,** *alunos.matricula*;

Agrupando informações

Você pode somar, contar ou realizar a média de valores, mas para isso, você precisa selecionar algum campo que será agrupado

Sintaxe

```
SELECT tabela1.campo_1, SUM(tabela1.campo_2), COUNT(tabela2.campo_1)
FROM nome_tabela_1, nome_tabela_2
WHERE tabela1.chave_estrangeira = tabela2.chave_primaria
GROUP BY tabela1.campo_1;
```

Exemplo

```
SELECT turma.periodo, count(alunos.matricula)
FROM alunos, turma
WHERE alunos.turma_idturma = turma.idturma
GROUP BY turma.periodo;
```

Comandos

SUM() – Soma os valores
COUNT() – Conta os valores
AVG() – Média dos valores