Instituto Federal de Educação, Ciência e Tecnologia da Bahia - IFBA
Departamento de Informática
Análise e Desenvolvimento de Sistemas / Licenciatura em Computação

Comandos SQL - PARTE 1

André L. R. Madureira <andre.madureira@ifba.edu.br>
Doutorando em Ciência da Computação (UFBA)
Mestre em Ciência da Computação (UFBA)
Engenheiro da Computação (UFBA)

Linguagem SQL

- Composta por:
  - **Linguagem de definição de dados (DDL)**
  - **Linguagem de manipulação de dados (DML)**
  - **Restrições de integridade**
  - **Definição de view**
  - **Controle de transação**
  - **Controle de autorização**

Tipos de Dados de Atributos em SQL

- Definem o conjunto de dados que um atributo pode assumir (**domínio do atributo**)
- Cada sistema de gerenciamento de banco de dados (SGBD) fornece suporte a tipos de dados diferentes
  - **INT, SMALLINT, BIGINT**
  - **FLOAT, DOUBLE, NUMERIC(P, D)**
  - **BOOLEAN**
  - **DATE, DATETIME, TIME**
  - **VARCHAR(N), TEXT**

INT ou INTEGER

- Inteiro de tamanho normal (4 bytes),
  - Um inteiro **com** sinal pertence ao intervalo:
    - $[-2^{31} , +2^{31}-1] = [-2147483648, +2147483647]$
  - Um inteiro **sem** sinal pertence ao intervalo:
    - $[0 , 2^{32}] = [0, 4294967295]$

INT ou INTEGER

- Inteiro de tamanho normal (4 bytes)
- Podemos ter números inteiros de tamanho maior ou menor que 4 bytes, de acordo com o que for necessário para a aplicação
  - **TINYINT** (1B), **SMALLINT** (2B), **INT** (4B), **BIGINT** (8B)
- **Como saber qual deles eu preciso?**

INT ou INTEGER

- Inteiro de tamanho normal (4 bytes)
- Podemos ter números inteiros de tamanho maior ou menor que 4 bytes, de acordo com o que for necessário para a aplicação
  - **TINYINT** (1B), **SMALLINT** (2B), **INT** (4B), **BIGINT** (8B)
- **Como saber qual deles eu preciso?**
  - Precisamos calcular o intervalo de números que podemos representar com cada um deles

INT ou INTEGER

- Cálculo do intervalo de valores de um inteiro de tamanho N bytes:
  - Se o inteiro possui sinal (+/-):
    - $[-2^{N*8}/2, +2^{N*8}/2 - 1]$
  - Se o inteiro **não** possui sinal (+/-):
    - $[0, +2^{N*8}]$

FLOAT, DOUBLE e NUMERIC(p,d)

- Permitem representar números fracionários usando uma precisão previamente definida
  - **Objetivo**: representar esses números com o menor erro de aproximação possível

FLOAT, DOUBLE e NUMERIC(p,d)

- Permitem representar números fracionários usando uma precisão previamente definida
  - **Objetivo**: representar esses números com o menor erro de aproximação possível
  - **Erro de aproximação**: Erro causado em uma operação com números fracionários dentro de um computador
    - Quanto maior a precisão do número fracionário menor o erro

FLOAT, DOUBLE e NUMERIC(p,d)

- **Problema**: Números com maior precisão ocupam muito espaço da memória do computador
  - **Solução**: ajustar a precisão necessária de acordo com a necessidade do nosso sistema
  - **Ex**:
    - **FLOAT** (precisão simples)
    - **DOUBLE** (precisão dupla)
    - **NUMERIC** (precisão variável)

FLOAT, DOUBLE e NUMERIC(p,d)

- **Float**: Número decimal de precisão simples (32 bits)
- **Double**: Número de decimal de precisão dupla (64 bits)
- **Numeric(p,d):** número decimal de ponto fixo com **p** dígitos (incluindo a parte inteira e fracionária) e **d** casas decimais (algarismos da parte fracionária)
  - **Ex**: numeric(3,1) permite armazenar os números:
    - 44,2
    - 1,8

BOOLEAN

- **BOOLEAN**: Valores booleanos que podem ser representado por false/true ou 0/1
  - **Ex**: Aprovação ou reprovação em uma disciplina

```
create table disciplina (
        nome_aluno varchar(20),
        nota double,
        aprovado_reprovado boolean
);
```

DATE e TIME

- **DATE:** Uma data. A faixa suportada é entre '1000-01-01' e '9999-12-31'
  - MySQL mostra valores DATE no formato ‘AAAA-MM-DD’
  - **Ex**: armazenar data de nascimento de uma pessoa
- **TIME:** A faixa é entre '-838:59:59' e '838:59:59'.
  - MySQL mostra valores TIME no formato 'HH:MM:SS'
  - **Ex**: armazenar o horário de chegada e saída no IFBA

DATETIME

- **DATETIME:** Um combinação de hora e data. A faixa suportada é entre '1000-01-01 00:00:00' e '9999-12-31 23:59:59'
    - MySQL mostra valores DATETIME no formato 'AAAA-MM-DD HH:MM:SS
    - **Ex:** armazenar o dia e horário que um funcionário chegou para trabalhar

VARCHAR e TEXT

- **VARCHAR (M):** Uma string de tamanho variável **M** caracteres
  - M = [1 , 255]
  - Se o valor M especificado for maior que 255, o tipo do atributo é convertido para **TEXT**
- **TEXT:** String com tamanho máximo de 65535 caracteres
  - Cada valor do atributo consome 2^{16}-1 bytes
- **LONGTEXT:** máximo de 4,294,967,295 caracteres
  - Cada valor do atributo consome 2^{32}-1 bytes

Linguagem SQL

- A linguagem SQL é **não é orientada a indentação**
  - Ao contrário do Python, o SQL não se importa com espaços vazios
  - A indentação e quebras de linhas no SQL devem ser usadas para facilitar a leitura da consulta SQL

```
create table disciplina (
        nome_aluno varchar(20),
        nota double,
        aprovado_reprovado boolean
);
```

=

```
create table disciplina ( nome_aluno
varchar(20), nota double,
aprovado_reprovado boolean );
```

Linguagem SQL

- A linguagem SQL é ***case-insentive***
  - Tanto faz se um comando, tipo de dados, etc for escrito em maiusculo ou minusculo
  - O SQL irá interpretar o comando, tipo de dados, etc do mesmo jeito
  - **Ex**: **CREATE TABLE** e **create table** possuem o mesmo efeito
  - **Ex**: **DOUBLE** ou **double**

Linguagem SQL

- A linguagem SQL é ***case-insentive***
  - Não podemos usar **palavras reservadas** (ex: CREATE, TABLE, INT, DOUBLE, etc) do SQL para nomear atributos, ou para outros fins além daqueles para os quais elas foram projetadas (comandos SQL, tipos de dados, etc)
    - **Ex**: criar uma tabela chamada “*table*”
- **Convenção:** Palavras reservadas do SQL são escritas em MAIÚSCULO e **NEGRITO**

Linguagem SQL

- Todo comando da linguagem SQL precisa ser terminado com **;** (ponto e vírgula)
  - Não incluir ; no final é uma falha grave
    - Pode fazer comando SQL não ser executado
      - Alguns DBMS exigem o ; (ponto e vírgula)
    - Pode permitir que um atacante consiga executar comandos SQL indesejados no DBMS
      - **Ex**: *SQL Injection*

Linguagem SQL

- Cada sistema DBMS possui uma sintaxe, tipos de dados e recursos do SQL suportados
  - **Ex**: **INTEGER** (SQLite), **INT** (MySQL)
  - Em caso de dúvida, sempre consulte a documentação do seu DBMS !
- Nesta disciplina iremos adotar o padrão SQL do **MySQL**

Linguagem SQL e Comandos

- Nesta disciplina iremos adotar o padrão SQL do **MySQL**
  - MySQL é um dos maiores DBMSs em número de usuários
  - A sintaxe SQL do MySQL funciona na maioria dos outros DBMSs também
  - MySQL é um projeto open source (código aberto)

Comando **CREATE DATABASE**

- Cria um Banco de Dados
  - Cada banco de dados pode ser visto como uma coleção de tabelas relacionais, que estão associadas através de suas chaves (primárias e estrangeiras)
- **Sintaxe**
  - **CREATE DATABASE** <nome_do_banco_de_dados> ;
- **Ex**: **CREATE DATABASE** curso ;
- **Ex**: **CREATE DATABASE** projeto_bd ;

Exemplo de comando **CREATE TABLE**

- Crie uma tabela *disciplina* para armazenar os IDs dos alunos, notas e status (aprovado ou reprovado)

```
CREATE TABLE disciplina (
        id_aluno INT NOT NULL,
        nota DOUBLE DEFAULT 10,
        aprovado_reprovado BOOLEAN
);
```

Comando **CREATE TABLE**

- Cria uma tabela dentro de um banco de dados
- **Sintaxe**
  - **CREATE TABLE** <nome_tabela> (
    <nome_atributo1> <tipo> [**NOT NULL**] [**DEFAULT**] [**CHECK**],
    <nome_atributo2> <tipo> [**NOT NULL**] [**DEFAULT**] [**CHECK**],
    ...
) ;

Nome da tabela

Nome do atributo

Tipo de dados do atributo

Parâmetros opcionais
(*NOT NULL* = o atributo não pode ser nulo
DEFAULT = valor padrão para o atributo)

Exemplo do Comando **AUTO_INCREMENT**

- Crie uma tabela disciplina, e crie o ID de uma disciplina automaticamente, quando ela for criada

```
CREATE TABLE disciplina (
        id INT NOT NULL AUTO_INCREMENT,
        nome VARCHAR(20),
        id_turma INT NOT NULL,
        id_professor INT NOT NULL
);
```

Exemplo do Comando **AUTO_INCREMENT**

- Crie uma tabela disciplina, e crie o ID de uma disciplina automaticamente, quando ela for criada

```
CREATE TABLE disciplina (
        id INT NOT NULL AUTO_INCREMENT,
        nome VARCHAR(20)
        id_turma INT NOT NULL,
        id_professor INT NOT NULL
);
```

```
INSERT INTO disciplina VALUES (null, “Prog. WEB”, 741, 15);
```

Comando AUTO_INCREMENT

- Se não fornecermos um valor para o atributo, o DBMS irá pegar o último valor mais alto e incrementar automaticamente
- **Sintaxe (que funciona no MySQL)**
  - **CREATE TABLE** <nome_tabela> (
    <nome_atributo1> <tipo> [**AUTO_INCREMENT**],
    <nome_atributo2> <tipo> [**AUTO_INCREMENT**],
    ...
) ;

Comando AUTO_INCREMENT

- **Ex**: Criar um nova disciplina, e criar o ID dela automaticamente usando o **AUTO_INCREMENT**

INSERT INTO disciplina **VALUES** (null, “Prog. WEB”, 741, 15);

Disciplina			
id	**nome**	**id_turma**	**id_professor**
1	Banco de Dados	631	8
2	Modelagem	954	20

Disciplina			
id	**nome**	**id_turma**	**id_professor**
1	Banco de Dados	631	8
2	Modelagem	954	20
3	Prog. WEB	741	15

Inserido automaticamente pelo **AUTO_INCREMENT**

Comando **DROP**

- Remove a definição de uma tabela ou banco de dados, bem como remove os seus dados e referências.
- **Sintaxe**:
  - **DROP TABLE** <nome_tabela>;
  - **DROP DATABASE** <nome_database>;
- **Ex**: **DROP TABLE** Aluno;
- **Ex**: **DROP DATABASE** projeto01;

Comando **ALTER**

- Altera o nome da tabela ou adiciona / remove atributos na tabela
- **Sintaxe**:
  - **ALTER TABLE** <nome_antigo> **RENAME** <novo_nome>;
  - **ALTER TABLE** <nome_tabela> **ADD** <nome_atributo1> <tipo> [NOT NULL] ;
  - **ALTER TABLE** <nome_tabela> **DROP** <nome_atributo>;
- **Ex**: **ALTER TABLE** Aluno **RENAME** Discente ;
- **Ex**: **ALTER TABLE** Disciplinas **DROP** descricao ;

Comando **ALTER**

- **Ex**: **ALTER TABLE** Empregado **ADD** rg **VARCHAR(10) NOT NULL**, **ADD PRIMARY KEY**(rg);

Ao adicionar um novo atributo, os registros existentes são atualizados como **NULL** no atributo adicionado

Empregado		
id	**nome**	**cpf**
80	Juan	111.222.333-44
81	Hebert	555.666.777-88
82	Claudia	123.456.789-00

Empregado			
id	**nome**	**cpf**	**rg**
80	Juan	111.222.333-44	NULL
81	Hebert	555.666.777-88	NULL
82	Claudia	123.456.789-00	NULL

Comando **INSERT**

- Adiciona registros/tuplas em uma tabela
- **Sintaxe**:
  - **INSERT INTO** <nome_tabela> [(atributo1, atributo2,...)] **VALUES** (valor1,valor2,...) ;
  - **INSERT INTO** <nome_tabela> <consulta_SELECT>**;**
- **Ex**: **INSERT INTO** Categoria (nome_categoria, descricao) **VALUES** (“limpeza”,”produtos de limpeza”);
- Atributos **AUTO_INCREMENT, NULL** ou **DEFAULT** podem ser omitidos

Exemplo de Comando **INSERT**

- Sabendo que id é um campo **AUTO_INCREMENT**, crie uma nova disciplina usando o **INSERT INTO**. O id da disciplina deve ser gerado pelo SGBD.

```
INSERT INTO disciplina(nome,id_turma,id_professor)
VALUES (“Prog. WEB”, 741, 15);
```

Disciplina			
id	**nome**	**id_turma**	**id_professor**
1	Banco de Dados	631	8
2	Modelagem	954	20

Disciplina			
id	**nome**	**id_turma**	**id_professor**
1	Banco de Dados	631	8
2	Modelagem	954	20
3	Prog. WEB	741	15

Exemplo de Comando **INSERT**

- Seja a tabela Conta abaixo:
  - **CREATE TABLE** Conta (
    num **INT**,
    ag **INT**,
    saldo **DOUBLE**
    );
- Criar uma nova conta com R$ 200,00 de presente para cada cliente que possui um empréstimo no banco
  - **INSERT INTO** Conta **SELECT** num_emprestimo, nome_agencia, 200 **FROM** Emprestimo;

Comando **UPDATE**

- Modifica os valores dos atributos de um registro
- **Sintaxe**:
  - **UPDATE** <nome_tabela> **SET** <nome_atributo> = <novo_valor> [**WHERE** condicao] **;**
- **Ex**: **UPDATE** Categoria **SET** descricao='limpeza' **WHERE** id=1
  - Altere o atributo "descrição" dos registros da tabela "Categoria" que possuem "id = 1" para que contenham o valor "limpeza"

Comando **UPDATE**

- **Ex**: **UPDATE** Categoria **SET** descricao='limpeza' **WHERE** id=1
  - Altere o atributo "descrição" dos registros da tabela "Categoria" que possuem "id = 1" para que contenham o valor "limpeza"

Categoria		
id	**nome_categoria**	**descricao**
1	Limpeza	Produtos de Limpeza
2	Doces	Produtos comestíveis doces
3	Cosmeticos	Produtos cosméticos

Categoria		
id	**nome_categoria**	**descricao**
1	Limpeza	limpeza
2	Doces	Produtos comestíveis doces
3	Cosmeticos	Produtos cosméticos

Comando **DELETE**

- Remove registros de uma tabela
- **Sintaxe**:
  - **DELETE FROM** <nome_tabela> [**WHERE** condicao] **;**
- **Ex**: **DELETE FROM** Categoria **WHERE** id=1
  - Remova os registros da tabela “Categoria” que possuem “id = 1”

Comando **DELETE**

- **Ex**: **DELETE FROM** Categoria **WHERE** id=1
  - Remova os registros da tabela “Categoria” que possuem “id = 1”

Categoria		
id	**nome_categoria**	**descricao**
1	Limpeza	Produtos de Limpeza
2	Doces	Produtos comestíveis doces
3	Cosmeticos	Produtos cosméticos

Categoria		
id	**nome_categoria**	**descricao**
2	Doces	Produtos comestíveis doces
3	Cosmeticos	Produtos cosméticos

Referencial Bibliográfico

- KORTH, H.; SILBERSCHATZ, A.; SUDARSHAN, S. **Sistemas de bancos de dados**. 5. ed. Rio de Janeiro: Ed. Campus, 2006.

- DATE, C. J. **Introdução a sistemas de bancos de dados**. Rio de Janeiro: Ed. Campus, 2004. Tradução da 8ª edição americana.