Banco de Dados I

DDL

Repositório de dados

Quando falarmos sobre um repositório de dados, estaremos falando sobre armazenar dados. Para a criação de um repositório de dados em um Sistema Gerenciador de Banco de Dados (SGBD), veremos:

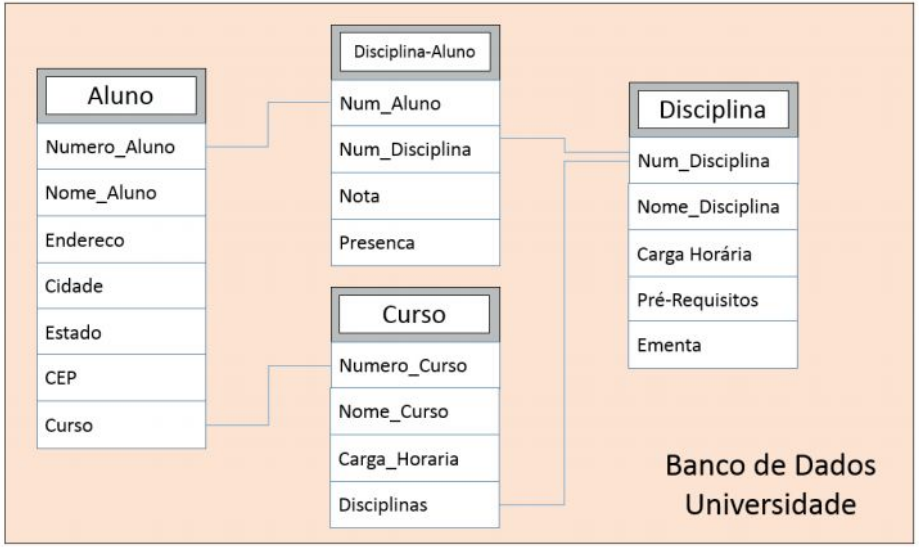
- Linguagem de consulta estruturada (SQL)
- Criação de tabelas
- Estrutura de banco de dados

Linguagem de consulta estruturada

Ao citarmos Programação em Bancos de Dados temos dois fatores muito importantes:

- **Programação** – Conjunto de técnicas para criar procedimentos estruturados que permitem que os computadores realizem tarefas desejadas.

- **Bancos de Dados Relacionais** – Conjunto de dados estruturados e correlacionados/ correlacionáveis, organizados de forma a permitir eficiência em sua manipulação por meio de uma linguagem formal (no caso, o SQL).



Linguagem de consulta estruturada

Banco de dados “Universidade”. Possui quatro tabelas:

- **Aluno** – Tabela com uma chave primária para o número do aluno, com informações gerais sobre o aluno e nome do curso em que ele está matriculado na instituição. Em outra tabela temos o detalhamento do curso.

- **Curso** – Tabela que contém informações acerca dos vários cursos oferecidos pela universidade (exemplos: Ciência da Computação, Engenharia da Computação, Engenharia Elétrica, Matemática, etc.). Cada curso tem informações de carga horária e um conjunto de disciplinas. Cada curso terá várias disciplinas componentes e cada disciplina pode ser oferecida em mais de um curso, criando uma relação N:N entre essas tabelas.

- **Disciplina-Aluno** – conecta duas tabelas (Disciplina e Aluno), contém informações sobre as disciplinas que um aluno cursa em determinado semestre.

Linguagem de consulta estruturada

SQL

A linguagem SQL se estabeleceu como a linguagem padrão dos SGBD (Sistema de Gerenciamento de Banco de Dados).

É uma linguagem de consulta estruturada (SQL, do inglês Structured Query Language) contendo , além de instruções de consultas ao banco de dados, instruções para:

- Definir esquemas de relacionamento, excluir relações e modificar estruturas.
- Criar restrições em relacionamentos, garantindo condições específicas de integridade e proibindo qualquer violação.
- Criar visões específicas sobre determinados dados.
- Realizar consultas interativas, baseadas em álgebra relacional, podendo, até mesmo, incluir, atualizar e excluir dados.
- Determinar a segurança do ambiente com todo o controle de acesso ao banco de dados, tabelas ou campos específicos.
- Determinar todo o controle de transações, garantindo a persistência e integridade dos dados.
- Permitir utilização autônoma ou como parte de outras aplicações.

Linguagem de consulta estruturada

SQL

Na linguagem SQL, destacam-se cinco subconjuntos de instruções que estão representadas conforme a Figura:

Linguagem de consulta estruturada

SQL

Na linguagem SQL, destacam-se cinco subconjuntos de instruções que estão representadas conforme a Figura:

DDL (Data Definition Language) - Linguagem de Definição de Dados.

Conjunto de instruções SQL para definição dos dados e sua estrutura.

- **CREATE** – cria banco de dados, tabelas, colunas.
- **DROP** – exclui banco de dados, tabelas, colunas.
- **ALTER** – altera banco de dados, tabelas, colunas.
- **TRUNCATE** – esvazia toda a tabela.

Linguagem de consulta estruturada

SQL

Na linguagem SQL, destacam-se cinco subconjuntos de instruções que estão representadas conforme a Figura:

DML (Data Manipulation Language) - Linguagem de Manipulação dos Dados.

Conjunto de instruções SQL para inserção e manutenção dos dados.

- **INSERT** – insere dados em uma tabela.
- **UPDATE** – atualiza os dados existentes em uma tabela.
- **DELETE** – exclui registros de uma tabela.

Linguagem de consulta estruturada

SQL

Na linguagem SQL, destacam-se cinco subconjuntos de instruções que estão representadas conforme a Figura:

DQL (Data Query Language) - Linguagem de Consulta a Dados.

Conjunto de instruções SQL para consulta de todos os dados armazenados e suas relações, e ajuda para comandos de sintaxe.

- **SELECT** – principal instrução de consulta do SQL.
- **SHOW** – exibe todas as informações além dos dados (metadata).
- **HELP** – exibe informações do manual de referência do MySQL.

Linguagem de consulta estruturada

SQL

Na linguagem SQL, destacam-se cinco subconjuntos de instruções que estão representadas conforme a Figura:

DCL (Data Control Language) - Linguagem de Controle de Dados.

Conjunto de instruções SQL para controle de autorizações de acesso e seus níveis de segurança.

- **GRANT** – essa instrução concede privilégios às contas de usuário.
- **REVOKE** – essa instrução permite revogar os privilégios da conta de usuário.

SQL

Na linguagem SQL, destacam-se cinco subconjuntos de instruções que estão representadas conforme a Figura:

DTL (Data Transaction Language) - Linguagem de Transação de Dados

Conjunto de instruções para o controle de transações lógicas que são agrupadas e executadas pela DML.

- **START TRANSACTION** – inicia uma nova transação.
- **SAVEPOINT** – identifica um determinado ponto em uma transação.
- **COMMIT** – é uma instrução de entrega ao SGBD, fazendo com que todas as alterações sejam permanentes.
- **ROLLBACK [TO SAVEPOINT]** – é uma instrução ao SGBD para reverter toda a transação, cancelando todas as alterações ou até determinado ponto da transação.
- **RELEASE SAVEPOINT** – instrução para remoção de um SAVEPOINT.

Linguagem de consulta estruturada

Estrutura básica das Consultas SQL

Em um repositório de dados, temos a definição de tabelas de dados baseada nas seguintes premissas:

- Cada **tabela** é uma **relação**.
- O **registro**, ou seja, o **conjunto de uma linha e colunas**, é chamado de **tupla**.
- Cada **campo** ou **coluna** dessa tabela tem um nome único representando um **domínio** distinto, denominado **atributo**.
- A ordem dos registros é irrelevante.
- **Não pode ocorrer dois registros iguais**.
- A ordem dos campos é irrelevante.
- **Cada tabela deve ser identificada de maneira única**, distinta de qualquer outra tabela do banco de dados, utilizando-se um nome próprio.

Linguagem de definição de dados - DDL

a criarmos um banco de dados chamado "Universidade", a instrução a
a seguinte:

```
CREATE DATABASE Universidade;
```

isso, o banco de dados foi criado e, utilizando a instrução:

```
SHOW DATABASES
```

emos visualizá-lo.

Se executarmos a instrução de criação de banco de dados e o mesmo já existir, teremos uma mensagem de erro.

Para isso não ocorrer, utilizamos a instrução da seguinte maneira:

```
CREATE DATABASE IF NOT EXISTS mundo;
```

Agora não será apresentado um erro, mas um alerta de que o banco já existe.

Linguagem de definição de dados - DDL

É necessário muito cuidado com a próxima instrução, pois ela **apagará** o banco de dados:

```
DROP DATABASE IF EXISTS Universidade;
```

A cláusula **IF EXISTS** previne que não seja gerado um erro se a base não existir.

Todas as instruções SQL poderão estar em um arquivo texto que chamamos de Script (roteiros de instruções SQL), o qual é bastante utilizado para criação de bancos de dados.

Em um Script, poderemos adicionar quaisquer comandos SQL, porém eles devem estar em uma ordem lógica de execução.

Modelo de dados

A instrução para a criação de tabelas e sua estrutura é a **CREATE TABLE**.

Em sua sintaxe, há vários parâmetros, mas aqui veremos as principais:

```
CREATE TABLE [IF NOT EXISTS] nome_tabela (
        Lista_atributos
);
```

Se a tabela já existir e a cláusula **IF NOT EXISTS** for utilizada, não ocasionará um erro, apenas um alerta.

Cria a tabela com o nome especificado.

Modelo de dados

Na lista de campos, a sintaxe é:

nome_atributo tipo_atributo[tamanho][**NOT NULL|NULL**][DEFAULT valor][AUTO_INCREMENT][PRIMARY KEY]

Nessa sintaxe, a cláusula **NOT NULL|NULL** indica se o campo aceita valores nulos ou não, **DEFAULT** especifica o valor padrão do campo e **AUTO_INCREMENT** identifica que o valor do campo é incrementado automaticamente quando um novo registro é inserido na tabela.

Cada tabela poderá ter apenas um campo AUTO_INCREMENT.

Modelo de dados

Tipos numéricos

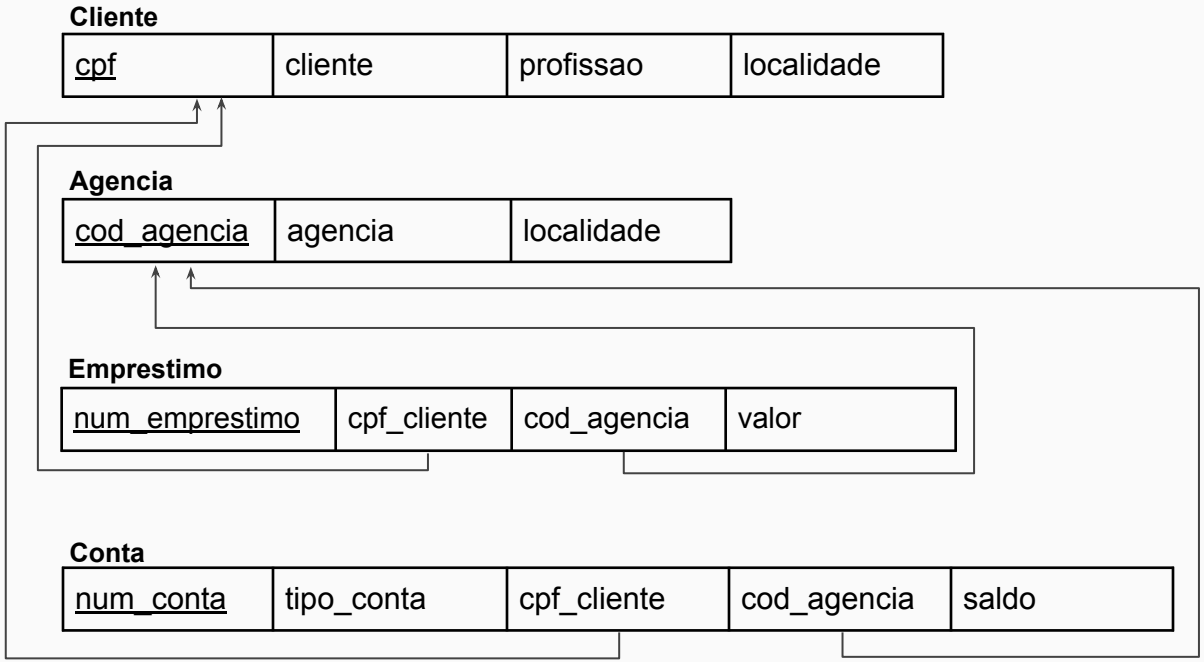
Numérico	**Com sinal**		**Sem sinal**	
SMALLINT	-32768	32767	0	65535
MEDIUMINT	-8388608	8388607	0	16777215
INT	-2147483648	2147483647	0	4294967295
BIGINT	-9223372036854775808	9223372036854775807	0	18446744073709551615
FLOAT	-3,402823466E+38	-1,175494351E-38	1,175494351E-38	3,402823466E+38
DOUBLE	-1,7976931348623157E+308	-2,2250738585072014E-308	2,2250738585072014E308	1,7976931348623157E+308
Data e hora		**Texto**		
DATE	YYYY-MM-DD	CHAR	0	255
TIME	HH:MM:SS	VARCHAR	0	65535
YEAR	YYYY	ENUM(LIST)		
TIMESTAMP	YYYY-MM-DD UTC	TEXT		
DATETIME	YYYY-MM-DD HH:MM:SS			

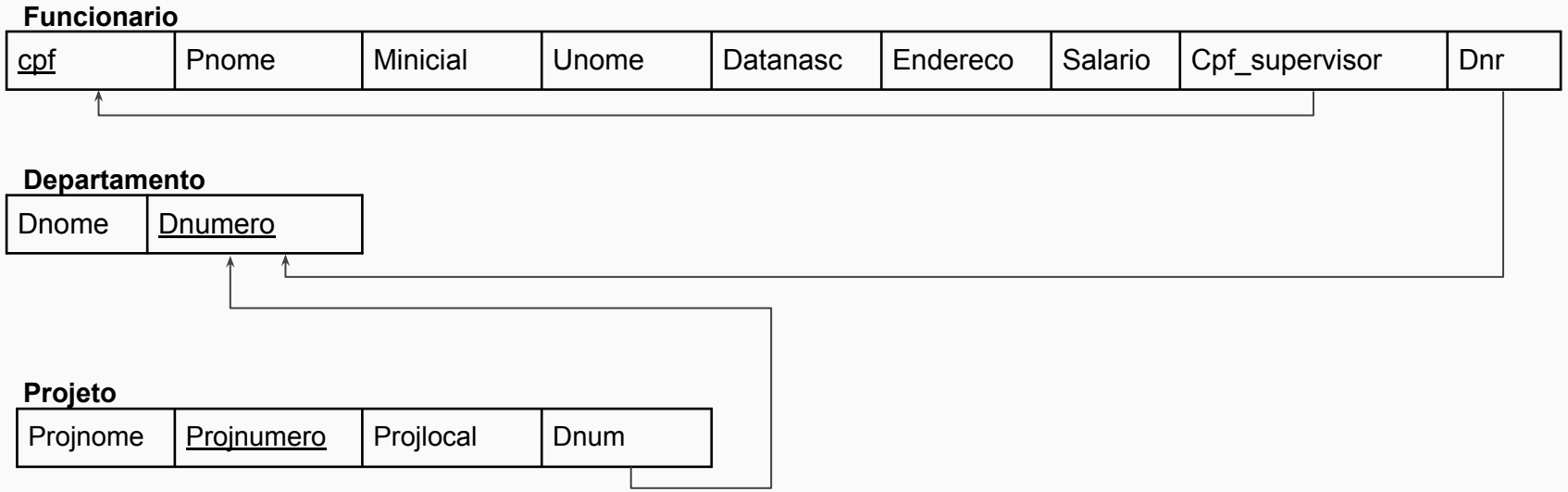
Modelo de dados

Para exemplificar a criação de uma tabela, vamos criar uma chamada de “convidado”.

Sua estrutura será composta por Identificação, Nome, Sobrenome, E-mail, Data de registro e Data de nascimento.

```
CREATE TABLE convidados (
    id INT(6) PRIMARY KEY,
    nome VARCHAR(30) NOT NULL,
    sobrenome VARCHAR(30) NOT NULL,
    email VARCHAR(50),
    data_reg DATETIME,
    nascimento DATE
);
```


Cliente
cpf
cliente
profissao
localidade
Agencia
cod_agencia
agencia
localidade
Emprestimo
num_emprestimo
cpf_cliente
cod_agencia
valor
Conta
num_conta
tipo_conta
cpf_cliente
cod_agencia
saldo


Funcionario
cpf
Pnome
Minicial
Unome
Datanasc
Endereco
Salario
Cpf_supervisor
Dnr
Departamento
Dnome
Dnumero
Projeto
Projnome
Projnumero
Projlocal
Dnum