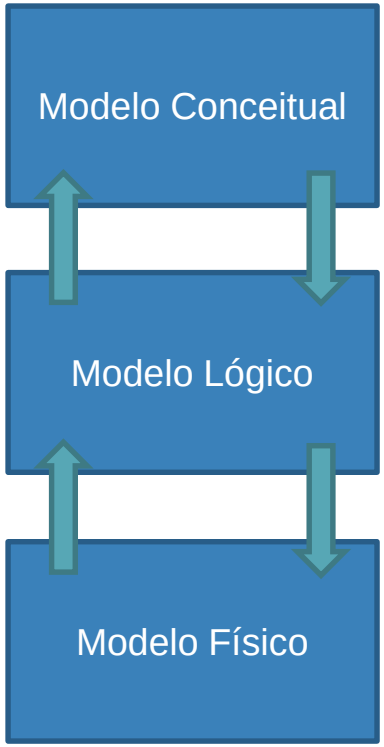
Banco de Dados II

Rebeca Barros

Minimundo



Modelo Entidade-Relacionamento (ER)

Modelo Relacional

Script SQL referente a algum SGBD específico

Projeto de Banco de Dados

SQL

- O Modelo Relacional foi definido por E.F.Codd em 1970 e pode ser definido como **uma coleção de uma ou mais relações, onde cada relação é uma tabela com linhas e colunas.**
- Baseada no trabalho de Codd, a IBM concebeu e desenvolveu uma linguagem para manipular os dados em tabelas relacionais.
- Inicialmente chamada de **SEQUEL**, a primeira versão comercial da **SQL** (Structured Query Language) foi lançada em 1979.

SQL

- A SQL se tornou uma padrão a partir de 86, quando a ANSI (American National Standars Institute) lançou oficialmente o primeiro padrão da linguagem. De lá pra cá, novas versões do SQL foram lançadas sempre com refinamentos da linguagem e com a adição de novas funcionalidades. A mais atual é a **SQL:2016.**

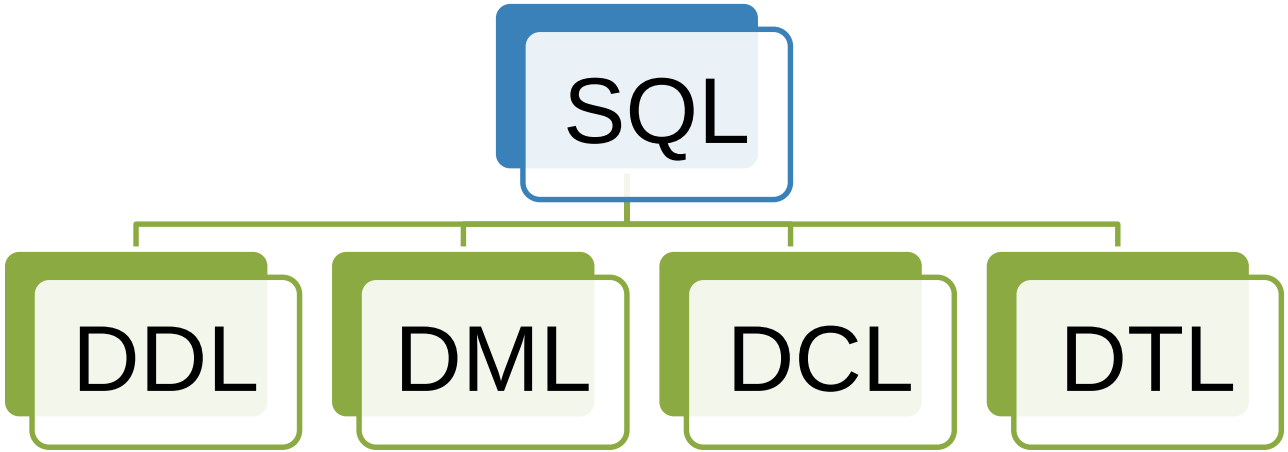
SQL

- SQL pode ser considerada um dos principais motivos pelo sucesso dos bancos relacionais. Como ela se tornou um padrão, os usuários ficaram menos preocupados com a migração de uma aplicação de outro sistema de banco de dados pra um relacional.
- Mesmo que um usuário esteja insatisfeito com um SGBD especifico, a conversão pra outro SGBD não é tão custosa, pois ambos seguem os mesmos padrões de linguagem.

SQL

- O resultado de uma consulta SQL é uma tabela (também chamada, nesse contexto, de conjunto-resultado ou, *result set*). Assim, uma nova tabela permanente no BD relacional pode ser criada armazenando o conjunto-resultado de uma consulta. De forma semelhante, uma consulta pode usar tanto tabelas permanentes quanto conjuntos-resultados de outras consultas.

SQL Componentes (Sublinguagens)



DDL

- **Data Definition Language (DDL)** consiste nos comandos SQL que podem ser usados para definir o esquema do banco de dados. Lida com a descrição do esquema e é usado para criar e modificar a estrutura do banco.
- **Principais comandos:**
  - **CREATE** – usado para criar o banco de dados ou objetos (como tabelas, índices, visões, etc..)
  - **DROP** – usado para deletar objetos do banco de dados;
  - **ALTER** – usado para alterar a estrutura do banco de dados.

DML

- **Data Manipulation Language (DML)** consiste nos comandos SQL que lidam com a manipulação dos dados presentes em um banco de dados.
- **Principais comandos:**
  - **SELECT** – usado para recuperar dados do BD
  - **INSERT** – usado para inserir dados em uma tabela;
  - **UPDATE** – usado para atualizar dados existentes em uma tabela
  - **DELETE** – usado para apagar dados de uma tabela.

DCL

- **Data Control Language (DCL)** consiste nos comandos SQL que lidam com acesso e permissões e outras opções de controle em um banco de dados.
- **Principais comandos:**
  - **GRANT** – concede a um usuário privilégios ao banco de dados;
  - **REVOKE** – revoca o direito de acesso de um usuário

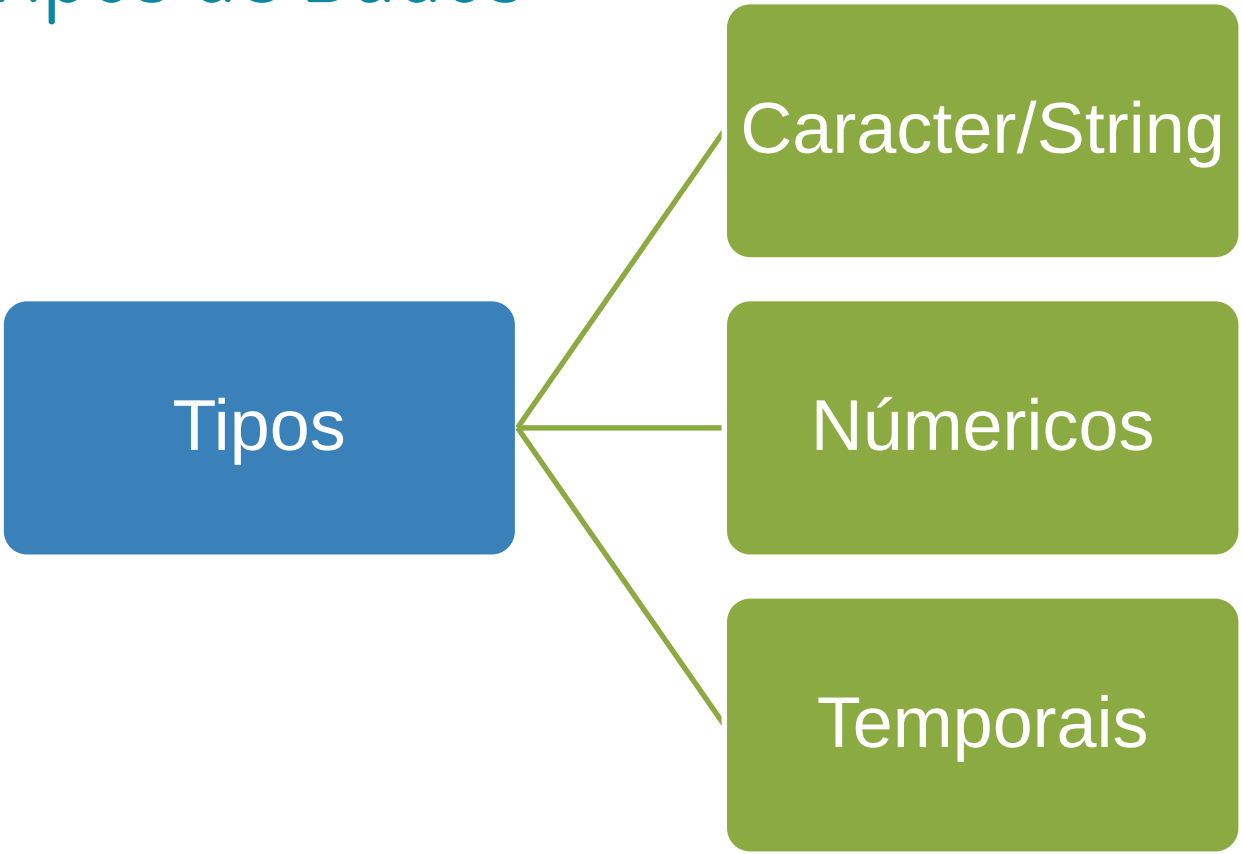
TCL

- **Transaction Control Language (TCL)** consiste nos comandos SQL que lidam as transações dentro de um banco de dados.
- **Principais comandos:**
  - **COMMIT** – confirma uma transação;
  - **ROLLBACK** – cancela uma transação no caso de algum erro;
  - **SAVEPOINT** – define um ponto salvo dentro da transação;
  - **SET TRANSACTION** – especifica as características para uma transação.

TIPOS DE DADOS

MySQL

Tipos de Dados



Tipos de Dados – Caractere

- Permite armazenar cadeias de caracteres (letras, símbolos e números).

Tipo	Descrição
CHAR(tamanho)	Armazena uma string de tamanho fixo, especificado no parênteses. Pode armazenar até 255 caracteres.
VARCHAR(tamanho)	Armazena uma string de tamanho variável, o tamanho máximo é especificado no parênteses. Pode armazenar até 255 caracteres.
TINYTEXT	Armazena uma string com tamanho máximo de 255 caracteres.
TEXT	Armazena uma string com o máximo de 65,535 bytes

Tipos de Dados – Caractere

Tipo	Descrição
BLOB	Para BLOBs (Binary Large Objects). Suporta até 65,535 bytes.
MEDIUMTEXT	Armazena string com tamanho máximo de 16,777,215 caracteres.
MEDIUMBLOB	Para BLOBs (Binary Large Objects). Suporta até 16,777,215 bytes.
LONGTEXT	Armazena string com tamanho máximo de 4,294,967,295 caracteres.
LONGBLOB	Para BLOBs (Binary Large Objects). Suporta até 4,294,967,295 bytes.

Tipos de Dados – Caractere

Tipo	Descrição
ENUM(x,y,z,etc.)	Permite a entrada de valores pré-estabelecidos. É possível armazenar até 65535 valores em uma lista ENUM. Se um valor é inserido que não está na lista, um valor vazio será inserido. Nota: Os valores são organizados na ordem em que são definidos. ENUM('M','F')
SET	Similar ao ENUM mas o SET pode conter até 64 itens e pode armazenar mais de uma escolha.

Tipos de Dados – Conjuntos de Caracteres

- Para línguas que usam o alfabeto latino, como inglês e português, há um número suficiente pequeno de caracteres usados de forma a se necessitar de apenas um byte para armazenar cada um deles;
- Outras línguas como japonês, possuem um número maior de caracteres, precisando de mais bytes para armazenamento.

Tipos de Dados – Conjuntos de Caracteres

- O MySQL pode armazenar dados de vários conjuntos de caracteres e permite que você defina o conjunto de caracteres específico para o seu banco de dados e/ou para uma coluna da sua tabela.
- O comando: **SHOW CARACTER SET;**
  lista todos os conjuntos disponíveis no MySQL.

Tipos de Dados – Numéricos

- Permite armazenar números.

Tipo	Escopo com sinal	Escopo sem sinal (unsigned)
TINYINT(tamanho)	-128 a 127	0 a 255
SMALLINT(tamanho)	-32768 a 32767	0 a 65535
MEDIUMINT(tamanho)	-8388608 a 8388607	0 a 16777215
INT(tamanho)	-2147483648 a 2147483647	0 a 4294967295
BIGINT(tamanho)	-9223372036854775808 a 9223372036854775807	0 a 18446744073709551615

Tipos de Dados – Numéricos

Tipo	Descrição
FLOAT(tamanho,d)	Um número pequeno com ponto decimal. O número máximo de dígitos pode ser especificado no parâmetro **tamanho**. O número máximo de dígitos após o ponto decimal pode ser especificado no parâmetro **d**.
DOUBLE(tamanho,d)	Um número grande com ponto decimal. O número máximo de dígitos pode ser especificado no parâmetro **tamanho**. O número máximo de dígitos após o ponto decimal pode ser especificado no parâmetro **d**.
DECIMAL(tamanho,d)	Um DOUBLE armazenado como uma string, permitindo um ponto decimal fixo. O número máximo de dígitos pode ser especificado no parâmetro **tamanho**. O número máximo de dígitos após o ponto decimal pode ser especificado no parâmetro **d**.

Tipos de Dados – Temporais

- Permite armazenar datas e/ou horários.

Tipo	Formato Padrão	Valores Permitidos
DATE()	YYYY-MM-DD	'1000-01-01' até '9999-12-31'
DATETIME()	YYYY-MM-DD HH:MI:SS	'1000-01-01 00:00:00' até '9999-12-31 23:59:59'
TIMESTAMP()	YYYY-MM-DD HH:MI:SS	'1970-01-01 00:00:01' até '2038-01-09 03:14:07'
TIME()	HH:MI:SS	'-838:59:59' até '838:59:59'
YEAR()	YYYY	1901 to 2155

Criando um Banco de Dados

MySQL

Criar Banco de Dados

```
CREATE DATABASE [IF NOT EXISTS] nome_banco
[CHARACTER SET nome_charset]
[COLLATE nome_collation]
```

Excluir um Banco de Dados

```
DROP DATABASE [IF EXISTS] nome_banco;
```

Criando Tabelas

```
CREATE TABLE [IF NOT EXISTS] nome_tabela(
    nome_coluna1 tipo_dado(tamanho)
    [restrições-coluna1],
    nome_coluna2 tipo_dado(tamanho)
    [restrições-coluna2],
    [restrições da tabela]
)
```

Criando Tabelas - Restrições

- **DEFAULT:** define um valor padrão para a coluna. Sempre que uma nova linha for inserida, caso o valor não seja informado, ela receberá o valor definido em DEFAULT.

```
nome_coluna1 tipo_dado(tamanho) DEFAULT valor-padrao
```

Criando Tabelas - Restrições

- **NOT NULL:** indica que o conteúdo da coluna não pode ser nulo. Todas as colunas que pertencem a chave primária da tabela devem possuir essa restrição.

```
nome_coluna1 tipo_dado(tamanho) NOT NULL
```

Criando Tabelas - Restrições

- **UNIQUE:** indica que não pode existir repetição no conteúdo da coluna.

```
nome_coluna1 tipo_dado(tamanho) UNIQUE
```

Criando Tabelas - Restrições

- **CHECK:** cria um domínio para uma coluna com os possíveis valores que ela pode assumir.

```
nome_coluna1 tipo_dado(tamanho) CHECK (condicao)

sexo CHAR(1) CHECK (sexo IN ('M', 'F'))
```

Criando Tabelas – Restrições

- **Chave Primária**

```
CONSTRAINT pk_nome PRIMARY KEY (lista-colunas) ou

PRIMARY KEY (lista-colunas)
```

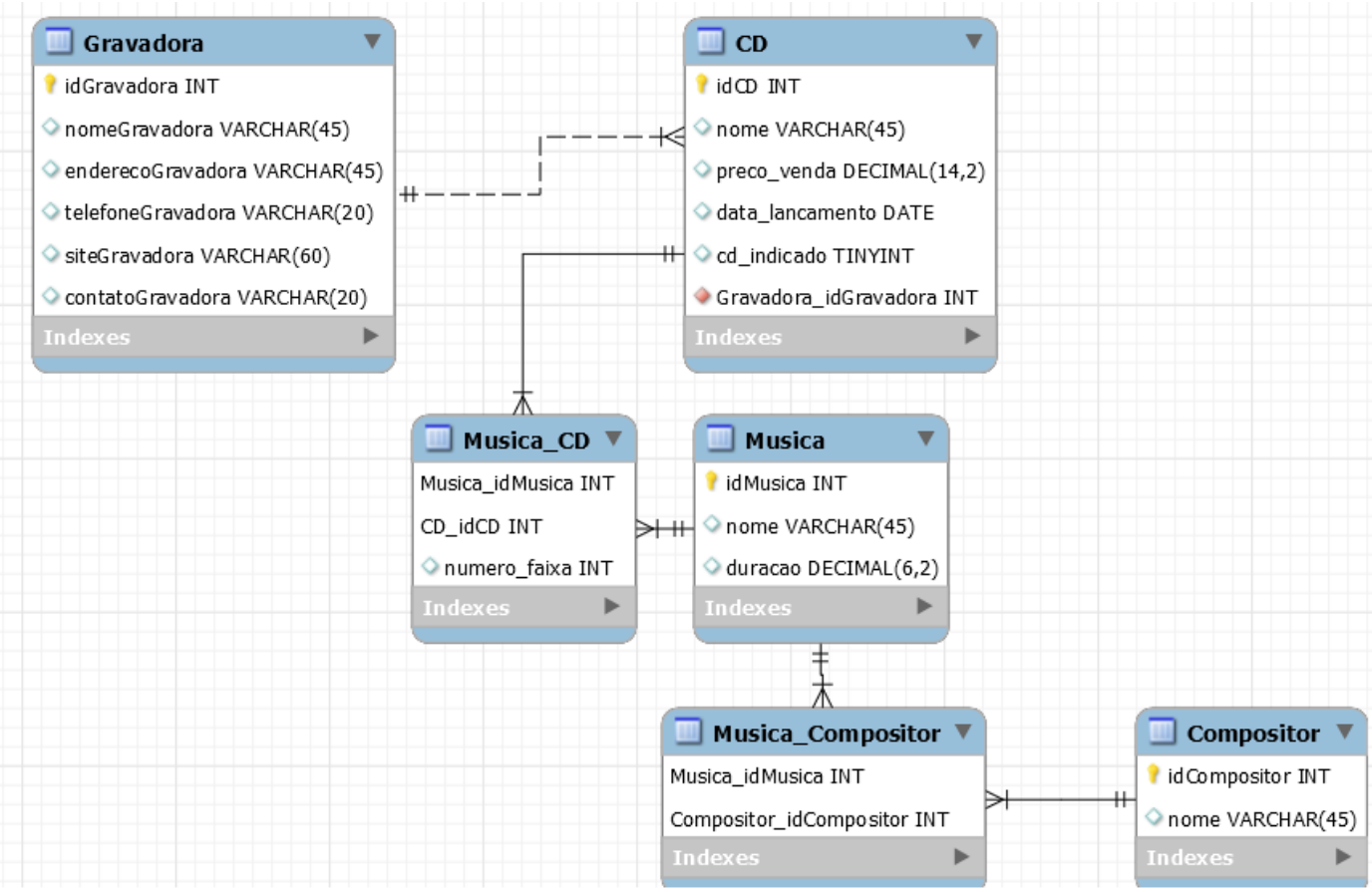
Criando Tabelas – Restrições

- **Chave Estrangeira**

```
CONSTRAINT fk_nome FOREIGN KEY (lista-colunas)
    REFERENCES nome-tabela (lista-colunas)
    [ON UPDATE ação]
    [ON DELETE ação]
ou
FOREIGN KEY (lista-colunas)
REFERENCES nome-tabela (lista-colunas)
```

Criando Tabelas - Restrições

- **Chave Estrangeira**
  - **Ação** determina qual ação o banco irá tomar quando for excluída ou alterada uma linha da tabela que contém referência a essa chave.
    RESTRICT ou NO ACTION – não permite a exclusão da chave primária
    - **CASCADE -** exclui ou altera os registros que se relacionam a ele
    - **SET NULL** – altera o contéudo da coluna para null
    - **SET DEFAULT** – altera o contéudo da coluna para o valor padrão



Considere o seguinte esquema relacional e crie o código SQL para criação física do banco de dados **bd_pratica**;

Exemplo – Tabela Gravadora

```
CREATE DATABASE IF NOT EXISTS bd_pratica;

#comando para conectar ao banco bd_pratica
use bd_pratica;

CREATE TABLE IF NOT EXISTS gravadora (
    idGravadora INTEGER NOT NULL,
    nomeGravadora VARCHAR(45),
    enderecoGravadora VARCHAR(45),
    telefoneGravadora VARCHAR(20),
    siteGravadora VARCHAR(60),
    contatoGravadora VARCHAR(60),

    PRIMARY KEY (idGravadora)
);
```

Exemplo – Tabela CD

```
CREATE TABLE IF NOT EXISTS cd (
    idCd INTEGER NOT NULL,
    nome VARCHAR(45),
    preco_venda DECIMAL (5,2),
    data_lancamento DATE,
    cd_indicado BOOL,
    idGravadora INTEGER,

    PRIMARY KEY (idCd),
    FOREIGN KEY (idGravadora) REFERENCES gravadora (idGravadora)

);
```

- Siga os exemplos e complete o código para criação física desse BD;