

---

# SQL - Structured Query Language

## Unidade 2 – Manipulação Básica de Dados – Parte 1

Prof. Daniel Callegari  
Escola Politécnica – PUCRS

---

### 1. Introdução

Nessa unidade, praticaremos a manipulação de bancos de dados, explorando consultas e atualizações de baixa complexidade. Iniciaremos o estudo da linguagem SQL e veremos as vantagens e desvantagens dos ambientes visuais de manipulação, quando comparados aos ambientes declarativos. Em especial, exploraremos: consultas simples e com junção de tabelas, seleção de registros, projeção de colunas, junção externa de dados, atualizações simples e SQL embutido.

### 2. Modelo Relacional

É um Modelo Lógico de Dados, originalmente proposto por E. F. Codd, caracterizado por:

- Independência de Dados: mudanças no esquema interno não afetam o esquema conceitual e mudanças no esquema conceitual não afetam as aplicações;
- Linguagem Estruturada de Consulta (SQL);
- Redução da Redundância;
- Simplicidade na representação dos dados;
- Simplicidade na representação dos resultados das consultas sobre os dados.

#### 2.1. Modelo Relacional – Conceitos Formais

- **Domínio:** dado uma característica de um objeto da realidade, por exemplo, o número de matrícula de um aluno, o domínio deste elemento de dado é o conjunto de todos os números de matrícula possíveis;
- **Relação:** um objeto da realidade pode ser representado, de forma abstrata, por um conjunto de elementos de dados definidos, cada um, sobre um domínio.

Exemplo: Suponha uma relação chamada LIVROS, composta pelos seguintes elementos de dados, ou ATRIBUTOS, definidos por domínios específicos:

<i>Código do livro</i>	Numérico, 5 posições, obrigatório
<i>Título</i>	Texto, tam.200, obrigatório
<i>Ano de lançamento</i>	Data, obrigatória
<i>Importado?</i>	Booleano, S/N, obrigatório
<i>Preço</i>	Numérico, 10 posições, 2 decimais
<i>Prazo de entrega</i>	Numérico, 3 posições, obrigatório

Assim, cada relação possui:

- Um Cabeçalho: composto pelo conjunto de ATRIBUTOS que a compõe
- Um Corpo: composto por um conjunto de TUPLAS, cada qual correspondendo à representação abstrata de um objeto da realidade

Adicionalmente:

- Uma TABELA armazena os objetos de uma classe da realidade;
- Uma tabela é composta por COLUNAS, as quais armazenam determinado atributo dos objetos da classe;
- Cada instância de uma classe (objeto) é armazenada como uma LINHA da tabela;

### 3. A Linguagem SQL

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

A Linguagem SQL (Structured Query Language) é padrão para os SGBDs Relacionais que seguem o padrão ANSI (American National Standards Institute).

A Linguagem SQL pode ser dividida em cinco conjuntos de comandos:

<i>Recuperação de dados</i>	Comando SELECT
<i>Linguagem de definição de dados (DDL - Data Definition Language):</i>	Comandos para criação e manutenção de objetos do banco de dados: CREATE, ALTER, DROP, RENAME e TRUNCATE
<i>Linguagem de manipulação de dados (DML - Data Manipulation Language):</i>	Comandos para inserções (INSERT), atualizações (UPDATE) e exclusões (DELETE)
<i>Linguagem para controle de transações</i>	Comandos COMMIT, ROLLBACK e SAVEPOINT
<i>Linguagem para controle de acesso a dados</i>	Comandos GRANT e REVOKE

Observações gerais sobre os comandos SQL:

- Não importa se são escritos em maiúsculas ou minúsculas;
- Podem ser escritos em uma só linha ou com quebras após qualquer palavra;
- Usualmente coloca-se cada cláusula de um comando em uma linha separada;
- Sugere-se usar tabulações e espaços para melhorar a clareza.

### 3.1. Criando a sua primeira tabela no banco de dados

Para criar uma tabela no banco de dados, usamos o comando CREATE TABLE:

```
CREATE TABLE nome_da_tabela
(
    nome_da_coluna1 tipo_de_dado [NULL | NOT NULL],
    nome_da_coluna2 tipo_de_dado [NULL | NOT NULL],
    nome_da_coluna3 tipo_de_dado [NULL | NOT NULL],
    ...
    restrições ...
);
```

Observação: os elementos entre colchetes são opcionais. Neste caso servem para dizer, para cada coluna, se o seu dado é obrigatório (NOT NULL) ou opcional (NULL). Esse conceito, assim como as restrições, será explorado mais adiante.

Para exemplificar, vamos criar uma tabela para armazenar informações sobre alguns veículos. Experimente o seguinte comando no SGBD:

```
CREATE TABLE VEICULOS
(
    placa CHAR(8),
    ano NUMBER(4),
    km NUMBER(6),
    marca VARCHAR(50),
    modelo VARCHAR(50)
);
```

← repare que não há virgula aqui.

Tipos de dados básicos em SQL:

<b>CHAR (tamanho)</b>	Sequência de caracteres de tamanho fixo
<b>VARCHAR (tamanho)</b>	Sequência de caracteres de tamanho variável
<b>NUMERIC / DECIMAL / NUMBER (total, decimais)</b>	Armazena um valor, com ou sem decimais
<b>DATE / DATETIME</b>	Armazena uma data (pode incluir também a hora).

Existem outros tipos de dados, até mesmo dependentes do SGBD utilizado, porém vamos nos ater aos mais elementares aqui.

### 3.2. Inserindo alguns dados na tabela VEICULOS

Agora vamos inserir alguns veículos no banco de dados. Para isso, usaremos o comando INSERT INTO.

```
INSERT INTO nome_da_tabela [(colunas)]  
VALUES (valores);
```

Exemplos:

```
INSERT INTO VEICULOS  
VALUES ('IJK-1212', 2012, 0, 'Chevrolet', 'Vectra');  
  
INSERT INTO VEICULOS (placa, ano, km, marca, modelo)  
VALUES ('IJM-1556', 2001, 72045, 'Volkswagen', 'Gol');  
  
-- Isso é um comentário em SQL.  
-- Repare o uso de aspas simples para textos.  
-- Repare também na especificação das colunas  
--   no segundo caso.  
-- >>> Insira pelo menos mais cinco veículos no BD.
```

### 3.3. Consultando dados a partir da tabela VEICULOS

Para consultar os veículos armazenados no banco de dados, usamos o comando SELECT:


```
SELECT coluna1, coluna2, ...  
FROM nome_da_tabela  
[WHERE condicao];
```

Para consultar todos os dados de uma tabela usamos o comando a seguir:

```
SELECT *  
FROM nome_da_tabela;
```

Podemos também aplicar um filtro nas linhas e nas colunas retornadas pelo comando SELECT. Vamos consultar a placa e o ano dos veículos novos (zero km). Para isso, usamos a cláusula WHERE:

```
SELECT placa, ano  
FROM VEICULOS  
WHERE km = 0;
```

 Escreva o comando para selecionar a placa, o ano e o modelo dos veículos anteriores ao ano 2000. Experimente também variar a ordem das colunas.

### 3.4. Alterando dados na tabela VEICULOS

Para alterar uma informação no banco de dados, usamos o comando UPDATE:

```
UPDATE nome_da_tabela  
SET campo1 = valor1;
```

Por exemplo, se quiséssemos alterar a quilometragem dos veículos para 0 (zero), poderíamos usar o comando:

```
UPDATE VEICULOS  
SET km = 0;
```

Cuidado! O problema de usar o comando acima é que ele se aplica a toda a tabela, ou seja, a todos os registros armazenados. Como consequência, todos os veículos teriam o seu atributo km igual a 0!

Para que o comando UPDATE seja aplicado a somente um registro ou a alguns dos registros, também usamos a cláusula WHERE. Por exemplo, poderíamos alterar o modelo do veículo cuja placa é IJK-1212 para “Vectra Elite 2.0”:

```
UPDATE VEICULOS  
SET modelo = 'Vectra Elite 2.0'  
WHERE placa = 'IJK-1212';
```

Verifique como ficou:

```
SELECT *  
FROM VEICULOS;
```

Também podemos alterar diversos registros de uma só vez. Vamos, por exemplo, estabelecer que todos os veículos com menos de 10 quilômetros rodados devam apresentar a sua quilometragem igual a zero:


```
UPDATE VEICULOS  
SET km = 0  
WHERE km < 10;
```

Verifique como ficou:

```
SELECT *  
FROM VEICULOS;
```

Podemos usar também expressões aritméticas e lógicas (V/F). Exemplos:

- SELECT com `preco * 2`
- SELECT e WHERE com `idade > 18`
- SELECT e WHERE com `idade > 18 AND peso < 80`
- SELECT e WHERE com `idade <> 18`
- UPDATE com `x = x + 10`
- UPDATE com `preco = preco * 1.1`
- *(escreva outras expressões aritméticas)*
- *(escreva outras expressões lógicas. Use AND, OR, NOT)*

 Experimente escrever o comando UPDATE para somar 100 quilômetros a todos os veículos cujos anos estão entre 1998 e 2001 (inclusive). Dica: Depois de resolver, pergunte ao professor uma forma alternativa de restringir os intervalos numéricos.

### 3.5. Excluindo registros da tabela VEICULOS

Para excluir um ou mais registros de uma tabela, usamos o comando DELETE.

```
DELETE FROM nome_da_tabela  
WHERE condição;
```


Repare novamente na cláusula WHERE, que especifica quais registros deverão ser excluídos. Cuidado: se você omitir a cláusula WHERE, todos os registros da tabela serão excluídos! (Note: Isso não irá excluir a tabela em si. A estrutura da tabela permanecerá intacta. Você inclusive poderá inserir outros registros posteriormente).

Vamos excluir o veículo Gol, de 2001, cuja placa é IJM-1556:

```
DELETE FROM VEICULOS  
WHERE placa = 'IJM-1556';
```

Verifique como ficou:

```
SELECT *  
FROM VEICULOS;
```

 Experimente agora excluir todos os veículos da marca Chevrolet que possuem mais de 90.000 KM.

```
DELETE FROM VEICULOS  
WHERE (marca = 'Chevrolet') AND (km > 90000);
```

O uso dos parênteses não é obrigatório, mas facilita a leitura do comando.


Nota: Pode ser que seu comando não afete nenhuma linha porque talvez não existam veículos com tais características no banco de dados. Naturalmente, isso também poderá ocorrer com os comandos SELECT e UPDATE vistos anteriormente.

### 3.6. Ordenando os dados

A ordem dos registros de um comando SELECT não é garantida (pergunte para o seu professor o motivo). No entanto, se quisermos colocá-los em alguma ordem específica, podemos usar a cláusula ORDER BY no comando de seleção:

```
SELECT coluna1, coluna2, ...  
FROM nome_da_tabela  
[WHERE condição]  
ORDER BY coluna1 [ASC | DESC], coluna2 [ASC | DESC]...
```

Usamos ASC para ordenar do menor para o maior (ascendente) ou DESC para ordenar do maior para o menor (descendente). O padrão é ASC.

 Experimente os seguintes comandos:

```
SELECT placa, km  
FROM VEICULOS  
ORDER BY km;  
  
SELECT placa, km  
FROM VEICULOS  
WHERE ano > 2000  
ORDER BY km DESC;
```

É possível também ordenar por mais de uma coluna:

```
SELECT marca, modelo  
FROM VEICULOS  
ORDER BY marca DESC, modelo ASC;
```

### 3.7. Contando registros

Embora seja uma palavra-chave que pertence a outra categoria na linguagem SQL, é bastante conveniente conhecer nesse momento a “função de agregação” denominada COUNT.


Em rápidas palavras, uma função de agregação é uma função que opera sobre um conjunto de linhas de uma tabela. A função COUNT retorna o número de linhas envolvidas em uma consulta.

Para saber quantas linhas no total existem na tabela VEICULOS, podemos usar:

```
SELECT COUNT(*)  
FROM VEICULOS;
```

Para saber quantos veículos zero-KM temos registrados, podemos usar:

```
SELECT COUNT(*)  
FROM VEICULOS  
WHERE KM = 0;
```

 Agora tente descobrir quantos veículos da marca Ford estão cadastrados no banco de dados.

### 3.8. Evitando duplicatas


Depois de inserir diversos registros na tabela de veículos, experimente executar o seguinte comando:

```
SELECT marca  
FROM VEICULOS;
```

Se existir mais de um veículo da mesma marca, a listagem desse comando exibe o nome daquela marca repetidamente – afinal de contas, estamos solicitando a coluna que contém a marca de cada um dos registros armazenados.

Se, no entanto, quisermos saber o nome de cada uma das marcas do nosso estoque de veículos, mas sem repeti-los (ou seja, eliminando as ocorrências duplicadas), devemos usar o seguinte comando:

```
SELECT DISTINCT marca  
FROM VEICULOS;
```

 Para pensar: o que significa solicitar DISTINCT para a coluna “ano” sobre todos os registros da tabela VEICULOS?



### Dicas finais desta aula

- É provável que você tenha cometido algum erro de digitação nos comandos anteriores e o SGBD tenha emitido uma mensagem de erro. Acostume-se a ler e a identificar as diferentes mensagens de erro e tentar solucionar o problema sozinho. Preste especial atenção aos pares de parênteses, às vírgulas e às aspas simples.
- Não é obrigatório usar ponto e vírgula no final de cada comando SQL. No entanto, se você deseja executar diversos comandos juntos em sequência, eles deverão estar separados por pontos e vírgulas. Nota: O separador poderá variar conforme o SGBD.
- Tudo o que você faz de modificações no banco de dados fica armazenado entre as diferentes sessões (assumindo a opção *autocommit=on*; pergunte ao professor o que isto significa!). Se você fechar a conexão com o banco de dados, desligar a máquina e voltar no dia seguinte, seus dados estarão exatamente como você os deixou desde a última modificação.
- Frequentemente nos referimos aos sistemas gerenciadores de bancos de dados (SGBDs) – por exemplo Oracle, Microsoft SQL Server ou IBM DB2 – apenas como “bancos de dados”. Informalmente isso não traz consequências graves. Entenda, contudo, que SGBD é software, enquanto que um banco de dados não necessariamente está armazenado em meio digital (a sua ficha física no dentista faz parte de um banco de dados não digital na gaveta do consultório dele!). Na realidade, uma única instalação de um SGBD pode conter dezenas de bancos de dados armazenados em cada instância.
- É preciso que você diferencie o sistema gerenciador de banco de dados (SGBD) da interface cliente que usamos para interagir com ele. O software Management Studio (no caso da Microsoft) ou SQL Developer (no caso da Oracle) é apenas uma dentre diversas outras interfaces cliente para o SGBD. Note, portanto, que instalar apenas o Management Studio ou o SQL Developer (ou qualquer outra ferramenta similar) não instala o SGBD propriamente dito. Geralmente instalamos o SGBD em um servidor e usamos máquinas clientes para abrir conexões com o banco de dados.
- Há diversas versões de SGBDs gratuitos que você pode utilizar. A maior parte dos comandos deste curso segue o padrão SQL e, portanto, deve funcionar exatamente da mesma forma nos diversos SGBDs. Experimente os seguintes: MySQL Community Edition, Oracle Database Express Edition, Microsoft SQL Server Express Edition.

## Desafios

Parabéns! Você aprendeu um pouco sobre sistemas gerenciadores de bancos de dados relacionais e sobre a linguagem SQL. Mas tem muito mais pela frente! Aqui vão alguns desafios e perguntas para instigar a sua curiosidade:

1. Faz sentido usar a cláusula DISTINCT para a coluna “placa” da tabela VEICULOS?
2. O que acontece se for inserido um valor negativo para a coluna KM?
3. Até o momento temos apenas uma tabela no banco de dados. Crie uma tabela de PESSOAS, contendo CPF, Nome, Idade e Sexo. Insira alguns registros e faça algumas consultas.
4. Sabendo que uma pessoa pode possuir mais de um veículo, mas um veículo somente pode pertencer a uma única pessoa, como você criaria uma relação entre essas duas tabelas? (apenas para pensar, não executar)

-X-

## 4. Prática

### 1. Exercícios sobre conceitos gerais

- a) O que você entende por banco de dados relacional?
- b) Associe os comandos à esquerda com a parte correspondente da linguagem SQL:

<b>Comando</b>	<b>Parte da linguagem SQL</b>
( ) DELETE	A – Recuperação de Dados B – DML C – DDL D – Controle de Transações E – Controle de Acesso
( ) SELECT	
( ) CREATE	
( ) GRANT	
( ) UPDATE	
( ) DROP	
( ) INSERT	
( ) ALTER	
( ) COMMIT	
( ) REVOKE	

- c) Quais os principais tipos de dados da linguagem SQL? Dê um exemplo de uso de cada um.

### 2. Criação de Tabelas

- 2.1. Crie uma tabela para armazenar dados sobre alunos da universidade. Defina os atributos e seus tipos. Use o comando CREATE TABLE.
- 2.2. Crie uma tabela para armazenar dados sobre planetas. Defina os atributos e seus tipos. Use o comando CREATE TABLE.

### 3. Inserção de dados

- 3.1. Usando o comando INSERT, insira pelo menos 10 registros em cada uma das tabelas criadas no exercício anterior.

#### 4. Execute o SCRIPT de Filmes e resolva as questões:

4.1. Escreva comandos **SELECT** para os itens abaixo:

- a) o título, o ano e o diretor de todos os filmes.
- b) os filmes de horror de 2010.
- c) o título e o ano dos filmes com duração maior do que 2 horas.
- d) o título e a duração das comédias lançadas na década de 1990 com pelo menos 1 hora e 20 minutos de duração, dos diretores cujos nomes começam pela letra 'J'. Pesquise sobre o operador LIKE.
- e) o título, o gênero e o valor do ingresso dos filmes a partir de 2006, mostrando os valores inflacionados em 8,63%.
- f) a quantidade de filmes de ação com ingressos que custam mais do que R\$ 20,00.
- g) os nomes de todos os diretores cadastrados, sem repetir, e em ordem alfabética.

4.2. Escreva comandos **UPDATE** para os itens abaixo:

- a) aumentar em 10 minutos a duração dos filmes em que participa a atriz Angelina Jolie.
- b) dar um desconto de 10% para os filmes de ação do ano 2011.
- c) acrescentar um asterisco (\*) no final dos títulos dos filmes com duração menor ou igual a 30 minutos. Pesquise qual o operador utilizado para concatenar strings.

4.3. Escreva comandos **DELETE** para os itens abaixo:

- a) excluir os filmes com valor de ingresso superior a R\$ 60,00
- b) excluir os filmes em cujo título aparece a palavra "assombrado" ou cujo sobrenome do diretor é "Johnson". Use o operador LIKE para realizar essa questão.

-X-