Banco de Dados II

Curso de Análise e Desenvolvimento de Sistemas

Prof. MSc. Oscar Jr.

Linguagem Structured Query Language (SQL)

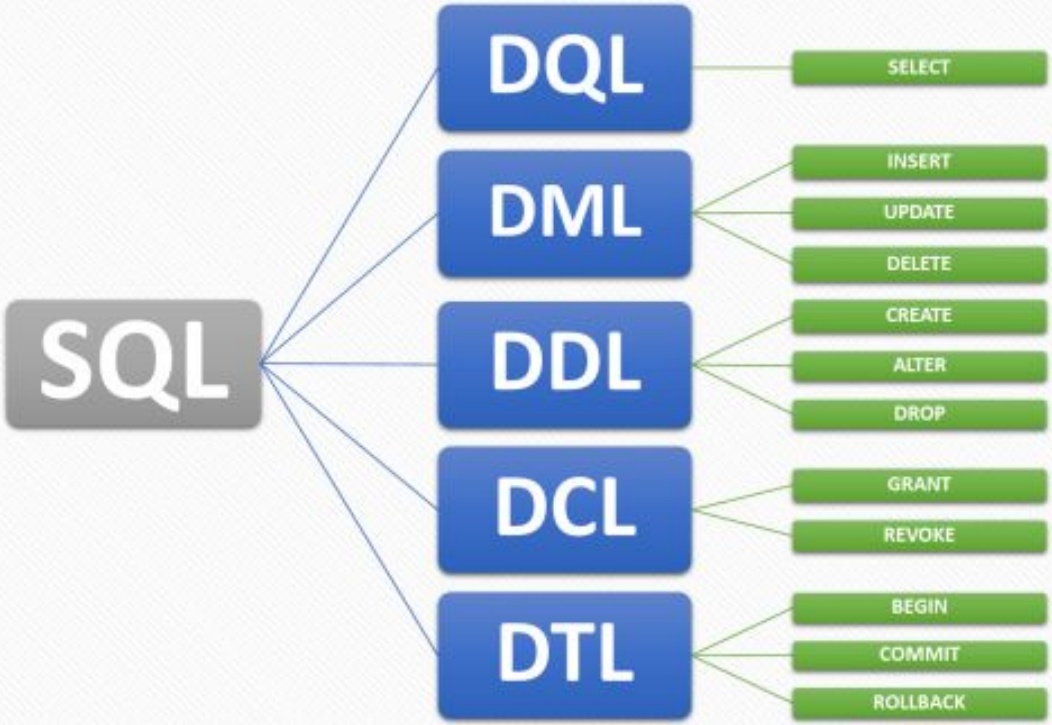
Através do SQL é possível **criar tabelas, colunas, índices, atribuir permissões a usuários**, assim como **realizar consultas a dados**.

O SQL é a **linguagem-padrão para os SGBDs relacionais** comerciais.

O SQL trata de uma **linguagem declarativa**, onde a preocupação é mais com o que deve ser feito ao invés de como será feito.

Os **comandos da linguagem** SQL são **divididos em conjuntos** e essa separação é feita de acordo com o que cada comando faz.

Linguagem Structured Query Language (SQL)



Data Query Language (DQL)

Trata de uma **linguagem de consulta de dados**, conta com o **conjunto da instrução utilizada para a recuperação e obtenção de registros dos bancos de dados**

- **SELECT**: utilizada para **recuperar informações de uma tabela em um banco de dados**

```
1   SELECT Nome, Telefone
2       FROM Cliente
3       WHERE Nome = 'Ramon';
```

- **ORDER BY**: definir uma **ordenação especifica dos dados recuperados**

```
1   SELECT Codigo, Nome FROM Cliente
2       ORDER BY Nome;
3   SELECT Codigo, Nome FROM Cliente
4       ORDER BY UF, Nome;
```

Data Query Language (DQL)

- **JOIN**: junção de duas ou mais tabelas

```
1  SELECT C.Nome, C.Telefone, P.Codigo, P.Data_Pedido
2      FROM Cliente C
3      JOIN Pedido P
4      ON C.ID = P.ID_Cliente
5      WHERE C.ID = 5;
```

Data Query Language (DQL)

- **SUBQUERY**: uma instrução SELECT dentro de outra instrução SQL

```
1  SELECT * FROM tabela1 AS T1 WHERE coluna1 IN (
2      SELECT coluna2 FROM tabela2 AS T2 WHERE T1.ID = T2.ID);
```

Data Manipulation Language (DML)

Conjunto de comandos que lidam com a **manipulação dos dados**

A **manipulação** envolve **inserir, recuperar, excluir e atualizar dados em tabelas** de banco de dados

- **INSERT**: **adiciona novas informações no banco de dados**

```
1   INSERT INTO Cliente (Nome, Telefone)
2      VALUES ('Suzana','99999-9999');
```

- **UPDATE**: **altera informações armazenadas no banco de dados**

```
1   UPDATE Cliente SET Telefone = '99999-9999' WHERE ID = 3;
2   UPDATE Cliente SET Telefone = '99999-9999';
```

- **DELETE**: **deleta informações do banco de dados**

```
1   DELETE FROM Cliente WHERE ID = 2;
2   DELETE FROM Cliente;
```

Data Definition Language (DDL)

Conjunto de comandos que **lidam com os objetos**, **criando bancos de dados, esquemas, tabelas, campos**

Permite que os usuários **especifiquem um esquema de banco de dados através de um conjunto de definições**

Um **esquema** é o **projeto geral de um banco de dados** e dificilmente é modificado

- **CREATE**: utilizado para **criar banco de dados, tabelas, store procedures**, entre outros

```
1  CREATE TABLE Contato (
2      ID int,
3      Nome varchar (255),
4      Telefone varchar(11)
5  );
```

Data Definition Language (DDL)

- **ALTER**: faz **modificações em objetos criados com o CREATE**, como **inserir ou remover uma nova coluna** em uma tabela, **alterar o tipo de dado das colunas**

```
1  ALTER TABLE Contato
2      ADD email varchar(255);
```

- **DROP**: **remove o que foi criado com o CREATE**

```
1  DROP TABLE Contato;
```

Data Control Language (DCL)

Conjunto das instruções usadas para **controlar o acesso e gerenciar permissões de usuários no banco de dados**

Permite que os usuários possam **especificar e controlar o acesso e gerenciamento de permissões para os demais usuários** em um banco de dados

- **GRANT**: **atribuir privilégios de acesso de um usuário a objetos em um banco de dados**

```
1  GRANT SELECT ON Tabela1 TO Usuario1;
```

- **REVOKE**: **remover privilégios de acesso aos objetos em um banco de dados**

```
1  REVOKE SELECT ON Tabela1 TO Usuario1;
```

- **DENY**: **impedir explicitamente que um usuário receba uma permissão específica**

```
1  DENY SELECT ON Tabela1 TO Usuario1;
```

Data Transaction Language (DTL)

Conjunto de instruções usadas para **gerenciar as transações que ocorrem dentro do banco de dados**

Permite que os usuários possam **gerenciar as mudanças feitas pelas instruções DML**

- **BEGIN**: usado para **especificar onde a transação inicia**
- **COMMIT**: usado para **salvar permanentemente qualquer transação no banco de dados**
- **ROLLBACK**: usado para **restaurar o banco de dados para o último estado commited**

```
1  CREATE TABLE Cliente (ID int);
2  BEGIN TRANSACTION;
3          INSERT INTO Cliente VALUES(1);
4          INSERT INTO Cliente VALUES(2);
5  COMMIT;
```

Tipos de Dados de Atributos no SQL

- **Numéricos**: englobam os **números inteiros de vários tamanhos** (**INT** e **SMALLINT**) e os **números de ponto flutuante de várias precisões** (**FLOAT**)

- **Cadeia de Caracteres**: possuem **tamanho fixo ou tamanho variável**. **CHAR(n)** onde n é o número de caracteres. **VARCHAR(n)** onde n é o número máximo de caracteres aceitos

- **Booleano**: valores tradicionais de **TRUE (verdadeiro)** ou **FALSE (falso)**. Também pode assumir o valor de **UNKNOWN (desconhecido)**, em virtude da presença dos valores NULL, temos esse terceiro valor lógico

Tipos de Dados de Atributos no SQL

- **Data e Hora**: um tipo de dado **DATE** contém **dez posições** e seus componentes são **YEAR (ano), MONTH (mês) e DAY (dia) no formato YYYY-MM-DD**. O tipo de dado **TIME** tem, no mínimo, **oito posições**, com os componentes **HOUR (hora), MINUTE (minuto) e SECOND (segundo) no formato HH:MM:SS**

- **NOT NULL**: o **SQL permite valores NULL para os atributos**, é possível **especificar uma restrição NOT NULL, não permitindo valores NULL para um atributo específico**. Implicitamente, todo atributo aceita valores NULL, com **exceção de um atributo que seja chave primária**

- **PRIMARY KEY**: usado para **especificar um ou mais atributos como chave primária**

- **FOREIGN KEY**: usado para **especificar um ou mais atributos como chave estrangeira, garantindo a integridade referencial entre os relacionamentos**

INTRODUÇÃO AO SQL

Componentes comuns de uma Query

`SELECT`	Quais **CAMPOS** você quer retornar com sua query	**SEMPRE**
`FROM`	Quais **TABLES** (tabelas) você quer usar?	**SEMPRE**
`WHERE`	Quais **FILTROS** você deseja aplicar	**FREQUENTE**
`GROUP BY`	Quais **CAMPOS** você deseja agrupar	**FREQUENTE**
`ORDER BY`	Quais **CAMPOS** você quer usar para ordenar	**ÀS VEZES**
`JOIN`	Junta ou agrega 2 ou mais tabelas	**ÀS VEZES**
`CASE`	Condicional	**ÀS VEZES**
`SUBSELECT`	Aplicar subtabelas	**ÀS VEZES**
`LIMIT`	**QUANTAS LINHAS** você deseja retornar?	**ÀS VEZES**

INTRODUÇÃO AO SQL

SQL é utilizado para comunicar questões à database

Os 3 comandos principais em SQL são SELECT, FROM e WHERE.

- SELECT
  - Permite que você selecione determinadas colunas de uma tabela.
  - Determina quais colunas de informações são baixadas.
- FROM
  - Especifica as tabelas das quais a consulta extrai dados.
- WHERE
  - DEFINE FILTROS
  - Determina quais linhas são selecionadas das tabelas.

INTRODUÇÃO AO SQL

Query Structure

Toda SQL query terá a mesma estrutura básica

SELECT ← Os campos que você quer retornar

FROM ← A tabela onde estão os dados

Employee

emplID	firstName	lastName	jobCode	salary
0001	Montgomery	Burns	AAA	1000000
0002	Homer	Simpson	SFI	15000
0003	Lenny	None	SFI	60000

INTRODUÇÃO AO SQL

Query Structure

- Quando selecionamos as 100 primeiras linhas de uma tabela, como produtos, na verdade, executamos uma instrução SQL em segundo plano:

```
SELECT * FROM vendas LIMIT 100;
```

* significa "TUDO" ou "todas"

- Você pode dizer que esta é uma consulta SELECT porque começa com SELECT.

INTRODUÇÃO AO SQL

Query Structure

Toda SQL query terá a mesma estrutura básica

```
SELECT
    codigo

FROM
    produtos;
```

Use um ; no final da query

Employee

name	gender	house_id	person_id	birthyear
****	******	******	******	******
****	******	******	******	******
****	******	******	******	******

INTRODUÇÃO AO SQL

Data Types

INTEGER: Valores possíveis: -2147483648 a 2147483647.
BOOLEAN: Valores possíveis: TRUE e FALSE.
DATE Armazena valores de ano, mês e dia
TIME Armazena valores de hora, minuto e segundo
TIMESTAMP Armazena ano, mês, dia, hora, minuto e segundo valores
DECIMAL: Exato numérico. exemplo: decimal (5,2) é um número que tem 3 dígitos antes do decimal e 2 dígitos após o decimal
CHARACTER (n) Cadeia de caracteres.

ORDER BY

SQL

ORDER BY

O ORDER BY é usado para classificar o conjunto de resultados em ordem crescente ou decrescente.

ORDER BY classifica os registros em ordem crescente por padrão. Para classificar os registros em ordem decrescente, use a palavra-chave DESC.

ORDER BY Sintaxe

```
SELECT coluna1, coluna2, ...
FROM nome_da_tabela
ORDER BY coluna1, coluna2, ... ASC|DESC;
```

INTRODUÇÃO AO SQL

Query Structure

SELECT

SELECIONE quais campos da tabela você deseja incluir em seus resultados

1. A instrução SELECT é necessária
2. Pense na instrução SELECT como escolhendo os campos que queremos
3. Use o * para incluir todos os campos em seus resultados

+

FROM

Em qual tabela esses campos são encontrados

1. A instrução FROM é necessária
2. Pense na instrução FROM como a forma como indicamos a fonte de dados

+

WHERE

WHERE nos permite definir condições para restringir os dados retornados

A instrução WHERE é opcional
Pense na instrução WHERE como um filtro

INTRODUÇÃO AO SQL

Query Structure

```sql
SELECT date, codigo, qty, ship_country
FROM vendas
WHERE date = '2022-06-28'
ORDER BY date DESC
```

10,000+ query results (1.92 seconds) **View log**

Showing first 10,000 rows. Download or open in a third-party app for a full result set.

date	codigo	# qty	ship_country

WHERE

SQL

OPERADORES

- Vamos aprender alguns operadores para usar em suas condições em suas SQL queries.

!=, <>	Diferente
>, >=	Maior, maior ou igual
<, <=	Menor que, menor ou igual
IN	Em um grupo de ítens
BETWEEN	Entre valores...

AGGREGATE FUNCTIONS

Operador	Significado
SUM	Soma de todos os valores
COUNT	A contagem de valores em uma coluna ou expressão
MAX	O maior valor na coluna
MIN	O menor valor na coluna
AVG	A Média de todos os valores
FIRST	O primeiro valor
LAST	O último valor

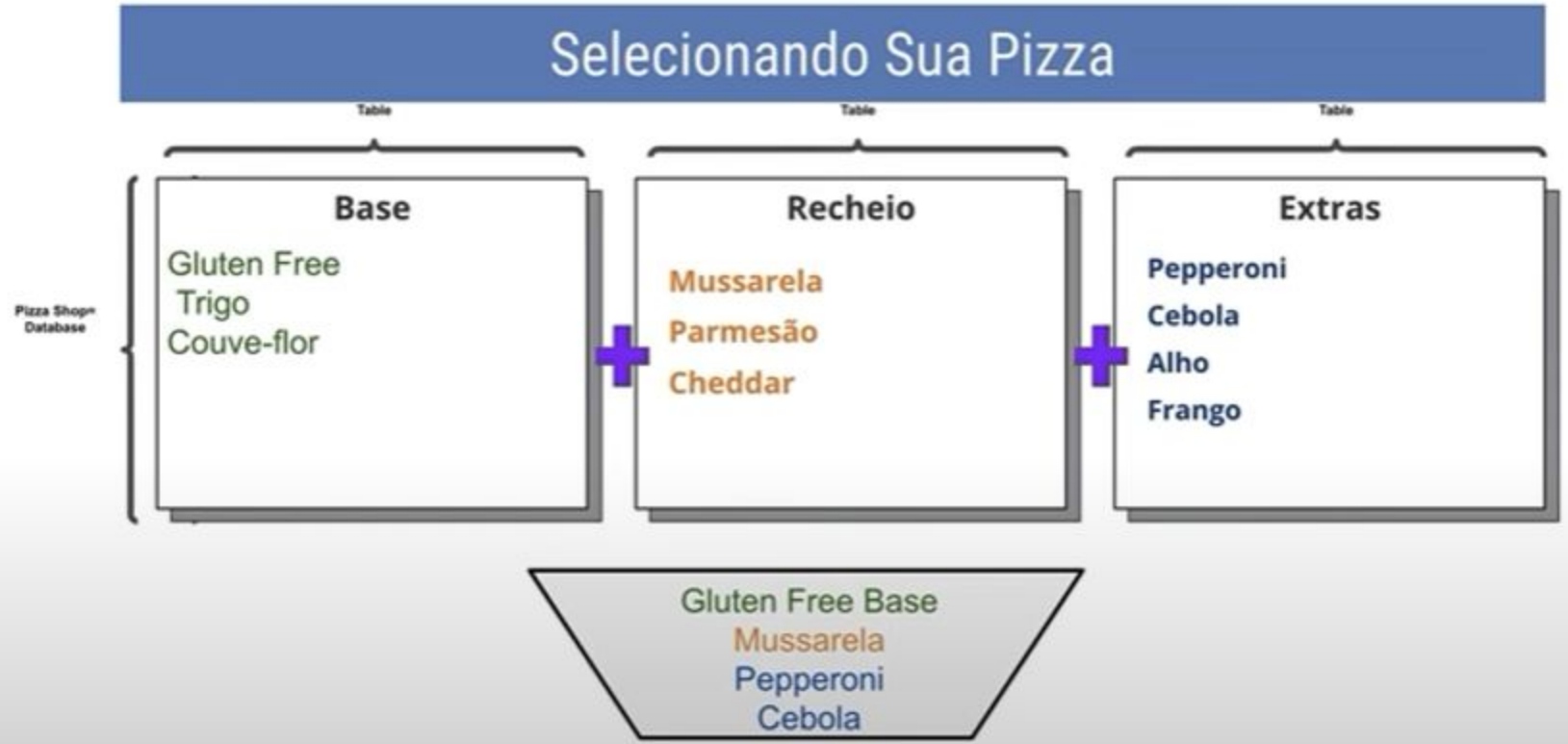
AGGREGATIONS

- Vejamos um exemplo, usando uma tabela chamada "vendas" com os campos "date" e "qty". Você deseja encontrar o total de quantidades vendidas por dia do dia mais atual primeiro

```
SELECT date, SUM(qty)
FROM vendas
GROUP BY date
ORDER BY date DESC
```

INTRODUÇÃO AO SQL

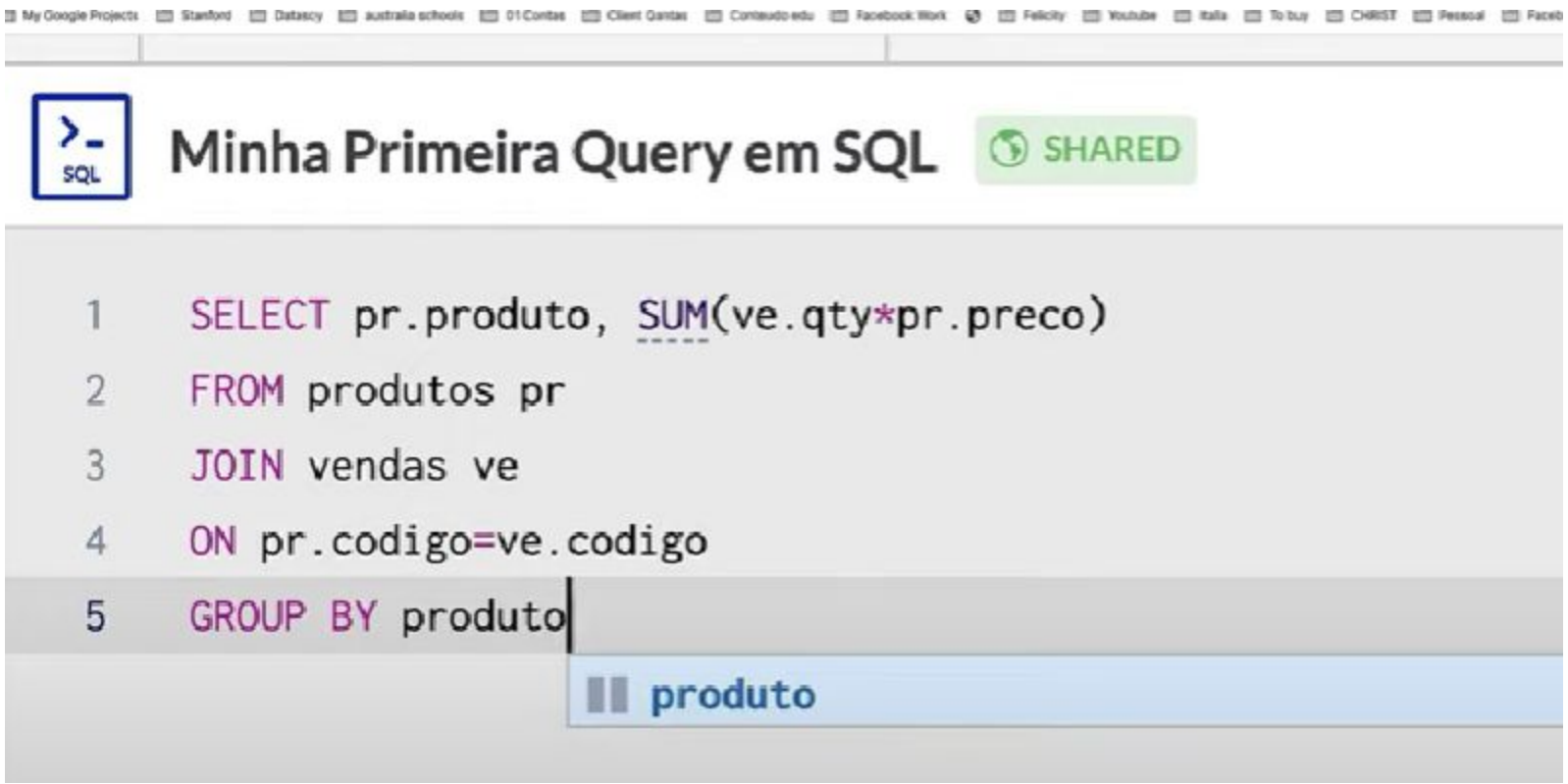
JOIN



JOINS

- Agora nós precisamos descobrir os produtos que tiveram o maior volume financeiro de vendas, ou seja, os produtos que trouxeram mais receita bruta pra nossa loja. Só que agora temos um problema. Os nomes dos produtos e o valor e as unidades vendidas estão em tabelas separadas. Como eu resolvo este problema?

```
SELECT
pr.produto, SUM(pr.preco*vd.qty)
FROM produtos pr
JOIN vendas vd
ON pr.codigo=vd.codigo
GROUP BY produto
```



CENTRO UNIVERSITÁRIO UNIVINTE

- SELECT STATEMENT
- SELECT DISTINCT
- WHERE (Filtros)
- OPERADORES (AND, OR...)
- CASE (CONDICIONAIS)
- HAVING
- JOINS (Inner Join, Left Join, Cross Join, Full Join...)
- Primary Keys (Chaves Primárias)
- INDEX
- Agregações (Group By, SUM, AVG...)
- SUBQUERIES
- Views e Store Procedures (Loops, While, IF THEN ELS
- STRING FUNCTIONS (CONCAT, TRIM, UPPER...)
- FUNÇÕES NUMÉRICAS (FLOOR, ROUND...)
- DATE AND TIME FUNCTIONS (Date, Timestamp, Day, week)
- PARTITIONING (Quando você estiver trabalhando com databases muito grandes)

Exercícios Data World

1. Listar todos os produtos disponíveis
2. Listar todas as vendas realizadas
3. Contar o total de vendas (linhas) registradas
4. Encontrar a quantidade total vendida por produto
5. Listar os produtos mais vendidos (por quantidade)
6. Mostrar o total de pedidos por data
7. Listar todos os países únicos que receberam entregas
8. Contar quantas vendas foram feitas para o Brasil (BR)
9. Listar os produtos que foram cancelados
10. Listar os produtos vendidos com sucesso (não cancelados)
11. Calcular a média de unidades vendidas por pedido
12. Listar os 5 produtos mais caros
13. Contar quantos produtos diferentes foram vendidos
14. Listar a quantidade total vendida por país
15. Mostrar quais produtos nunca foram vendidos