Banco de Dados Avançados

Aula 8
Consultas avançadas em Banco de Dados

Profa. Ma. Fabiana A. Rodrigues

EDUCAÇÃO METODISTA

Linguagem SQL

A linguagem SQL é dividida em duas: Linguagem de Definição de Dados (DDL) e Linguagem de Manipulação de Dados (DML). As DDLs são utilizadas para montar o banco de dados e suas tabelas, enquanto que as DMLs são utilizadas para manipular os dados armazenados no banco. Nesta Unidade de Aprendizagem, vamos tratar das principais instruções da linguagem SQL, assim como da finalidade de cada uma e sua aplicação no contexto de banco de dados.

Linguagem SQL

- Por meio da linguagem SQL (Structured Query Language), utilizada para interagir com banco de dados, é possível **inserir, buscar, remover e alterar valores e estruturas do banco de dados.**

- A **seleção (busca)** de dados em banco de dados compreende uma das ações mais importantes, uma vez que de nada adiantaria existirem valores armazenados se não houvesse sistemas que permitissem consultá-los.

JOINS em SQL

O que é um JOIN?

- **JOIN** é um comando utilizado para combinar registros de duas ou mais tabelas, criando um conjunto de resultados baseado em uma condição de relacionamento entre elas.

- As tabelas devem estar **relacionadas** por **chaves primárias** e **chaves estrangeiras** para que o JOIN possa funcionar corretamente.

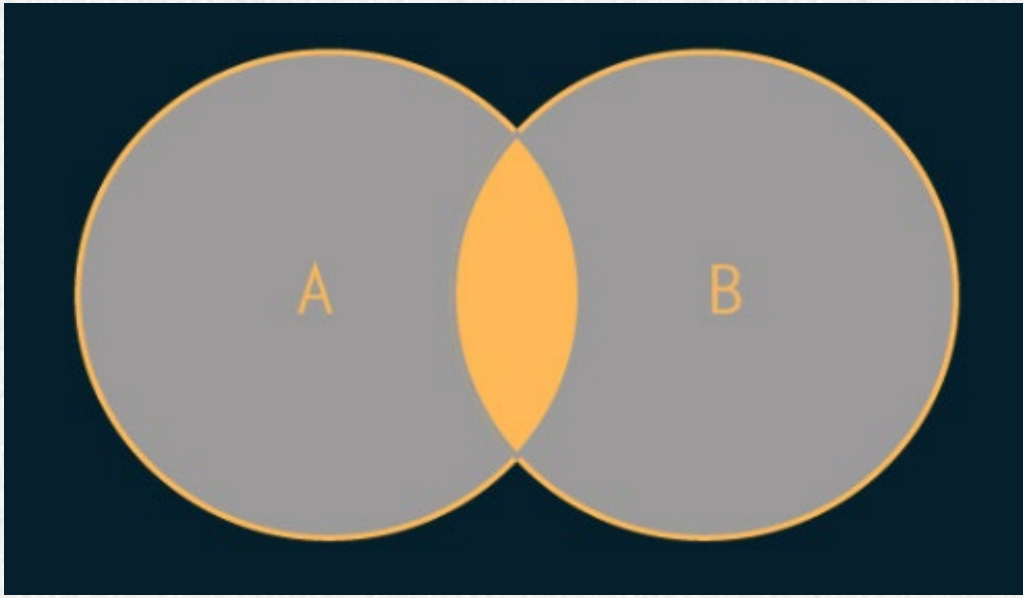
Utilizando JOINS

Temos três tipos de JOINS:

- INNER JOIN
- LEFT JOIN
- RIGHT JOIN

INNER JOIN

INNER JOIN:
Retorna apenas os registros que têm correspondências em ambas as tabelas.
•**Exemplo**: Mostra clientes que têm contas a pagar.



EDUCAÇÃO METODISTA

INNER JOIN

Retorna apenas as linhas das tabelas que sejam comuns entre si, ou seja, as linhas em ambas as tabelas que possuam o campo de relacionamento com o mesmo valor.

Sintaxe INNER JOIN:

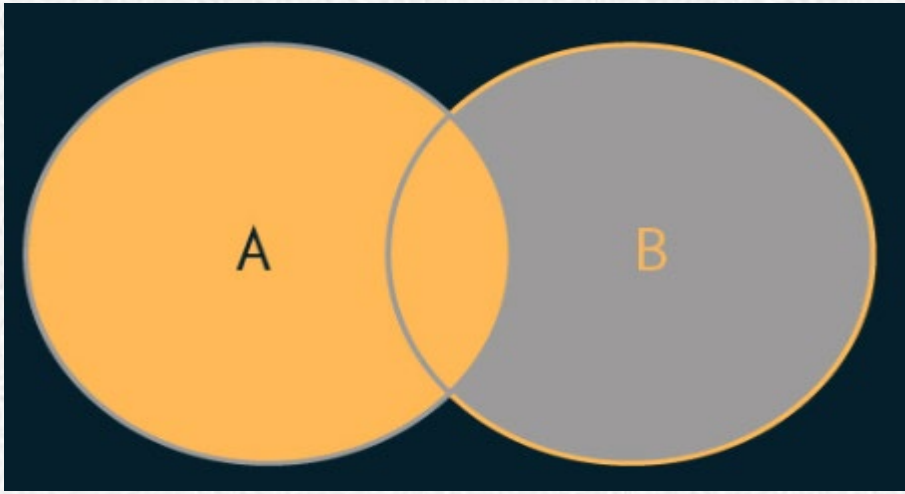
```
SELECT column_name(s)
FROM table1
INNER JOIN table2
ON table1.column_name = table2.column_name;
```

EDUCAÇÃO METODISTA

LEFT JOIN (ou LEFT OUTER JOIN)

LEFT JOIN (ou LEFT OUTER JOIN): Retorna todos os registros da tabela à esquerda, e os correspondentes da tabela à direita. Se não houver correspondência, os resultados da tabela da direita serão **NULL**.

•**Exemplo**: Mostra todos os clientes, mesmo aqueles que não têm contas a pagar.



EDUCAÇÃO METODISTA

LEFT JOIN

Irá listar todas as linhas da primeira tabela relacionada no JOIN, logo após a cláusula FROM.

Quando a linha listada não possuir equivalência na tabela destino , as colunas da tabela destino aparecerão com valores nulos.

Sintaxe de LEFT JOIN

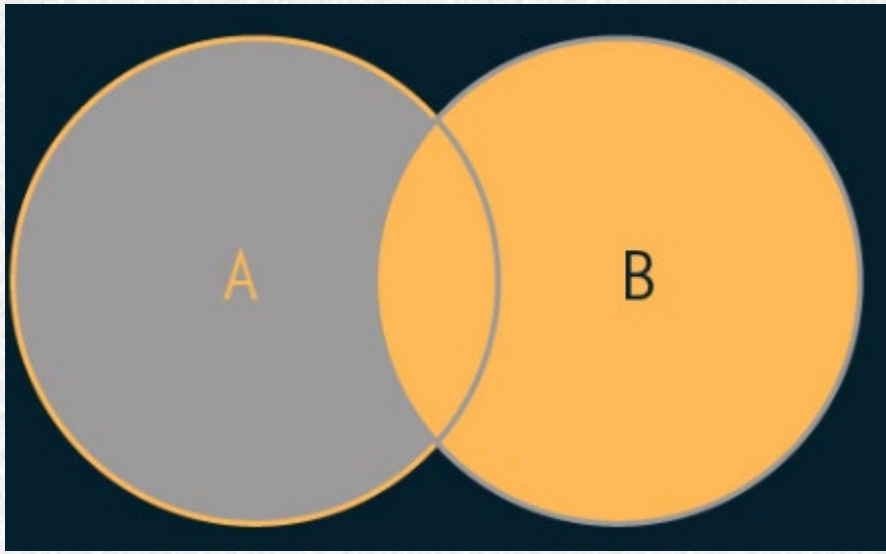
```
SELECT column_name(s)
FROM table1
LEFT JOIN table2
ON table1.column_name =
table2.column_name;
```

EDUCAÇÃO METODISTA

RIGHT JOIN (ou RIGHT OUTER JOIN)

Retorna todos os registros da tabela à direita, e os correspondentes da tabela à esquerda. Se não houver correspondência, os resultados da tabela à esquerda serão **NULL**.

•**Exemplo**: Mostra todas as contas, mesmo que não estejam associadas a um cliente.



EDUCAÇÃO METODISTA

RIGHT JOIN

Irá listar todas as linhas referentes à segunda tabela relacionada no JOIN.

Neste caso também , quando a linha listada não possuir equivalência na tabela destino , as colunas da tabela destino aparecerão com valores nulos.

Sintaxe de RIGHT JOIN

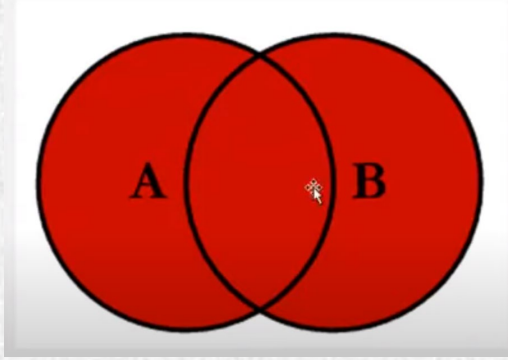
```
SELECT column_name(s)
FROM table1
RIGHT JOIN table2
ON table1.column_name =
table2.column_name;
```

EDUCAÇÃO METODISTA

FULL JOIN (ou FULL OUTER JOIN)

Retorna todos os registros quando há uma correspondência em qualquer uma das tabelas. Se não houver correspondência, o resultado conterá **NULLs** em relação à tabela faltante.

- **Exemplo**: Mostra todos os clientes e todas as contas, independentemente de haver correspondência entre eles.




```
SELECT column_name(s)
FROM table1
FULL JOIN table2
ON table1.column_name = table2.column_name;
```

EDUCAÇÃO METODISTA

CROSS JOIN em SQL

O que é um CROSS JOIN?

- **CROSS JOIN (ou produto cartesiano)** combina **todas as linhas** de duas ou mais tabelas. O resultado final será todas as combinações possíveis entre as linhas das tabelas envolvidas.
- Ao contrário de outros tipos de JOIN, o CROSS JOIN não precisa de uma condição de relacionamento entre as tabelas (não usa chaves primárias ou estrangeiras).

Características:

- Resultado: O número total de linhas no resultado será o produto do número de linhas de cada tabela.
- Por exemplo, se a tabela A tem 3 linhas e a tabela B tem 4 linhas, o resultado do CROSS JOIN será 3 * 4 = 12 linhas.
- Uso comum: O CROSS JOIN pode ser útil para gerar todas as combinações possíveis de dois conjuntos de dados.

```
SELECT column_name(s)
FROM table1
CROSS JOIN table2;
```

EDUCAÇÃO METODISTA

Aula 9

EDUCAÇÃO METODISTA

Views

Criando Tabelas Virtuais (Visões)

Uma **View**, também conhecida como **Exibição** ou **Visão**, é uma tabela virtual que se baseia no conjunto de resultados de uma consulta SQL. Ela é criada a partir de um conjunto de tabelas (ou outras **views**) existentes no banco de dados e funciona como se fosse uma tabela em si.

Views

Características principais e utilização

Estrutura de Dados: A View é uma estrutura de dados no banco de dados.

Similar a Tabelas Reais: Ela se assemelha a uma tabela real, contendo linhas e colunas.

Manipulação de Dados: Assim como uma tabela normal, você pode executar comandos como declarações JOIN, WHERE e usar funções sobre os dados armazenados na View.

Armazenamento: A View é armazenada no banco de dados, facilitando o acesso a informações complexas.

As **Views** são valiosas para simplificar consultas complexas e fornecer uma camada adicional de abstração para os usuários do banco de dados. Elas ajudam a tornar o acesso e a manipulação de dados mais eficientes e organizados.

Views

Aplicações das Views

As Views são ferramentas poderosas com várias aplicações úteis:

1. **Simplificação de Acesso a Dados:** As Views simplificam o acesso a dados que estão armazenados em múltiplas tabelas relacionadas, tornando as consultas mais eficientes e fáceis de executar.

1. **Implementação de Segurança de Dados:** É possível criar Views com cláusulas WHERE para limitar o acesso a dados, fornecendo segurança ao restringir as informações que podem ser visualizadas.

1. **Isolamento de Aplicação:** As Views permitem isolar uma aplicação da estrutura específica de tabelas do banco de dados, tornando a manutenção e atualização da aplicação mais flexível.

Essas aplicações das Views tornam-nas uma ferramenta valiosa no gerenciamento de dados e na otimização do acesso aos mesmos.

Views

Funções SQL relacionadas a “VIEWS”

As funções SQL relacionadas a "VIEWS" são usadas para **criar, gerenciar e usar *VIEWS*.**

CREATE VIEW	• Cria uma nova *view*.
ALTER VIEW	• Altera uma *view* existente.
DROP VIEW	• Exclui uma *view* existente.
DESCRIBE VIEW	• Mostra informações sobre uma *view*.
SELECT FROM VIEW	• Consulta uma *view*.

Views

Função CREATE VIEW

A função **CREATE VIEW** é usada para criar uma nova view. A sintaxe da função CREATE VIEW é a seguinte:

```sql
CREATE VIEW view_name AS
SELECT
  column_name(s)
FROM
  table_name
[WHERE condition];
```

EXEMPLO

```sql
CREATE VIEW clientes_ativos AS
SELECT
  nome,
  email,
  telefone
FROM
  clientes
WHERE
  status = 1;
```

Onde:

- **view_name:** É o nome da view que você está criando.
- **column_name(s):** São as colunas que você deseja incluir na view.
- **table_name:** É a tabela que você deseja usar como base para a view.
- **condition:** É uma condição que você pode usar para filtrar os resultados da view.

Views

Função ALTER VIEW

A função **ALTER VIEW** é usada para alterar uma view existente. A sintaxe da função ALTER VIEW é a

```sql
ALTER VIEW view_name
ADD COLUMN column_name data_type;

ALTER VIEW view_name
DROP COLUMN column_name;

ALTER VIEW view_name
MODIFY COLUMN column_name data_type;

ALTER VIEW view_name
RENAME TO new_view_name;
```

EXEMPLO

```sql
ALTER VIEW clientes_ativos
ADD COLUMN idade INT;
```

Onde:

- **view_name:** É o nome da view que você está alterando.
- **column_name:** É o nome da coluna que você deseja adicionar, remover, modificar ou renomear.
- **data_type:** É o tipo de dados da coluna que você está adicionando ou modificando.
- **new_view_name:** É o novo nome da view.

Views

Função DROP VIEW

A função **DROP VIEW** é usada para excluir uma view existente. A sintaxe da função DROP VIEW é a seguinte:

```
SQL

DROP VIEW view_name;
```

EXEMPLO

.

```
SQL

DROP VIEW clientes_ativos;
```

Onde:

- **view_name:** É o nome da view que você deseja excluir.

EDUCAÇÃO METODISTA

Views

Função DESCRIBE VIEW

A função **DESCRIBE VIEW** é usada para mostrar informações sobre uma view. A sintaxe da função DESCRIBE VIEW é a seguinte:

```
SQL

DESCRIBE view_name;
```

EXEMPLO

```
SQL

DESCRIBE clientes_ativos;
```

Onde:

- **view_name:** É o nome da view sobre a qual você deseja obter informações.

EDUCAÇÃO METODISTA

Views

SELECT FROM VIEW

A função **SELECT FROM VIEW** é usada para consultar uma view. A sintaxe da função SELECT FROM VIEW é a seguinte:

```sql
SELECT
  column_name(s)
FROM
  view_name;
```

EXEMPLO

.

```sql
SELECT nome
FROM clientes_ativos;
```

Onde:

- **column_name(s):** São as colunas que você deseja incluir nos resultados da consulta.
- **view_name:** É o nome da view que você deseja consultar.

EDUCAÇÃO METODISTA