Aula 17

Prof: Henrique Augusto Maltauro

Implementar Banco de Dados Para WEB

SQL

- **DQL**

Depois de criado as nossas tabelas com os comandos de **DDL** (`CREATE`, `ALTER`, `DROP`), e inserido, atualizado ou removido os registros das tabelas com os comandos de **DML** (`INSERT`, `UPDATE`, `DELETE`), nós podemos consultar os nossos registros com os comandos de **DQL**.

Mais precisamente, o **DQL** se resume em **um único comando**, e **vários subcomandos** que são dependentes do primeiro comando.

SQL

- **DQL: SELECT**

O principal comando **DQL** é o **SELECT**, que vai consultar os registros em uma tabela.

Seguido desse comando, é necessário **informar as colunas dessa tabela** que nós queremos visualizar, ou, podemos usar um asterisco `*` para dizer que queremos ver todas as colunas daquela tabela.

Depois de informado as colunas, usamos o comando **FROM** seguido do nome da tabela que está sendo consultada.

SQL

- DQL: SELECT

```
SELECT                                                    *
FROM NOME_DA_TABELA
```

SQL

- **DQL: SELECT**

Para escolher separadamente quais colunas serão apresentadas na consulta, é preciso informar elas pelo **nome das colunas**, separadas por **vírgula**.

SQL

- DQL: SELECT

```sql
SELECT
    NOME_DA_COLUNA1,
    NOME_DA_COLUNA2
FROM NOME_DA_TABELA
```

SQL

- **DQL: SELECT**

Caso nós quisermos filtrar o resultado da consulta, precisamos informar as cláusulas de condição com o comando **WHERE**, da mesma forma que fazemos no **UPDATE** e **DELETE**.

SQL

- DQL: SELECT

```
SELECT
    NOME_DA_COLUNA1,
    NOME_DA_COLUNA2
FROM                            NOME_DA_TABELA
WHERE NOME_DA_COLUNA = VALOR
```

SQL

- **DQL: SELECT (AS)**

No **SELECT**, podem fazer uso de um **alias**, usando o comando **AS**.

O **alias** é um **apelido**, um **nome temporário**, que pode tanto ser atribuído a uma coluna, quanto a uma tabela.

O **alias** é extremamente útil, ainda mais para quando formos juntar os registros das tabelas.

SQL

- **DQL: SELECT (AS)**

Para usar o **alias**, basta informar o comando **AS** logo após o nome da coluna, ou da tabela, e em seguida informar o apelido.

SQL

- DQL: SELECT (AS)

```
SELECT
    ID              AS          ID_PRODUTO,
    NOME             AS        NOME_PRODUTO
FROM PRODUTO AS P
```

SQL

- **DQL: SELECT (JOIN)**

O **SELECT** é talvez o comando **SQL** mais usado de todos, porém raramente a consulta dos registros de uma única tabela é suficiente.

Na sua maioria de uso, o **SELECT** consulta os registros de mais de uma tabela ao mesmo tempo, e junta esses registros.

E essa junção é feita com os subcomandos de **JOIN**.

SQL

- **DQL: SELECT (JOIN)**

Temos 4 subcomandos de **JOIN**:

➔ **INNER JOIN**

➔ **LEFT JOIN**

➔ **RIGHT JOIN**

➔ **FULL OUTER JOIN**

SQL

- **DQL: `SELECT` (`JOIN`)**

Hoje, vamos nos concentrar apenas no `INNER JOIN`, que é o mais básico e mais simples de compreender.

A ideia agora é que você se acostumem com o uso do `SELECT` e das **junções simples** de tabelas, para daí passarmos para junções mais complexas.

SQL

- **DQL: `SELECT` (`INNER JOIN`)**

Então, o `INNER JOIN` vai juntar os registros de **duas ou mais tabelas**.

Para usar ele, logo após informar a tabela principal, fazemos uso do comando `INNER JOIN` seguido do nome da tabela a qual queremos juntar os registros.

SQL

- DQL: SELECT (INNER JOIN)

```
SELECT                                                    *
FROM PRODUTO
INNER JOIN CATEGORIA
```

SQL

- **DQL: SELECT (INNER JOIN)**

Depois disso, precisamos informar o comando **ON**, seguido de uma comparação entre um campo da primeira tabela e um campo da segunda tabela.

E essa comparação tem por objetivo determinar qual registro da segunda tabela, está relacionado com o registro da segunda tabela.

SQL

- **DQL:** `SELECT` (`INNER JOIN`)

Em geral, aqui é feita uma comparação de **FK** e **PK**, e deixa muito mais claro a importância do uso das **FKs**.

Não é obrigatório fazer o uso de **FKs** nessa comparação, contudo, se a comparação for feita com o uso de **FKs** a consulta é realizada de forma mais performática.

SQL

- DQL: SELECT (INNER JOIN)

```
SELECT *
FROM PRODUTO
INNER JOIN CATEGORIA ON PRODUTO.ID_CATEGORIA = CATEGORIA.ID
```

SQL

- **DQL: `SELECT` (`INNER JOIN`)**

Aqui, também fica muito mais claro, o quanto o uso de **alias** facilita a criação de scripts **SQL**.

SQL

- DQL: SELECT (INNER JOIN)

```
SELECT *
FROM PRODUTO AS P
INNER JOIN CATEGORIA AS C ON P.ID_CATEGORIA = C.ID
```

SQL

- **DQL: SELECT (INNER JOIN)**

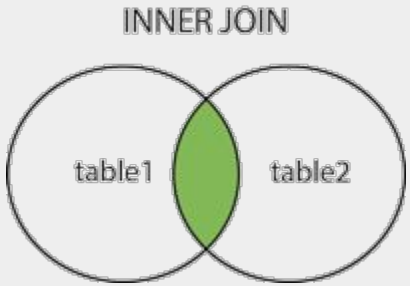
É muito importante lembrar que, o uso do **INNER JOIN** resulta em uma consulta que apenas retorna **registros que tenham uma vinculação válida dos registros**.

Ou seja, registros da primeira tabela que não tem vínculo com a segunda, ou registros da segunda tabela que não tem vínculo com a primeira, **não serão apresentado no resultado da consulta**.

SQL

- **DQL: SELECT (INNER JOIN)**

É muito comum, se fazer uso desta imagem para representar o funcionamento do `INNER JOIN`.



SQL

- **DQL: `SELECT` (`INNER JOIN`)**

Quando se usa um `JOIN`, e quer escolher separadamente quais colunas serão apresentas, é preciso especificar qual é tabela que possui aquela coluna, para **evitar conflitos de duplicidade**.

SQL

- **DQL: SELECT (INNER JOIN)**

Por exemplo, o código abaixo, não vai ser executado, por causa de **conflito de duplicidade de coluna**.

```
SELECT
    ID,
    NOME
FROM PRODUTO AS P
INNER JOIN CATEGORIA AS C ON P.ID_CATEGORIA = C.ID
```

SQL

- **DQL:** `SELECT` (`INNER JOIN`)

Novamente, fica mais claro, o quanto o uso de **alias** facilita a criação de scripts **SQL**.

SQL

- DQL: SELECT (INNER JOIN)

```
SELECT
 P.ID,
P.NOME
        FROM          PRODUTO          AS          P
INNER JOIN CATEGORIA AS C ON P.ID_CATEGORIA = C.ID
```

Exercício

Exercício

gg.gg/SenacBD17

github.com/hmaltaurodev/slides