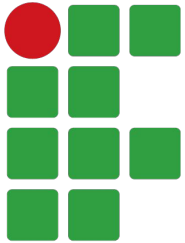


INSTITUTO FEDERAL
Goiano
Campus Morrinhos

BANCO DE DADOS
DATA QUERY LANGUAGE - DQL

PROFESSOR MARCEL MELO
MARCEL.MELO@IFGOIANO.EDU.BR

INSTITUTO FEDERAL
Goiano

REVISÃO SQL

Comando para criação do modelo de dados - DDL

- Create Database
- Create Table
- Alter Table
- Drop Table

Comando para manipulação de dados - DML

- Insert Into
- Update
- Delete From

INSTITUTO FEDERAL
Goiano

COMANDO SELECT

O comando SELECT recupera os dados de uma ou mais tabelas, sendo um dos comandos mais simples e, ao mesmo tempo, mais extenso da SQL devido as suas funções, operandos, comandos, subcomandos e cláusulas não obrigatórias.

```
SELECT <lista-colunas-seleção>
FROM <lista-tabelas>
WHERE <condição>
```

INSTITUTO FEDERAL
Goiano

COMANDO SELECT

```
SELECT <lista-colunas-seleção>
FROM <lista-tabelas>
WHERE <condição>
```

- O **SELECT** especifica as colunas que serão apresentadas no resultado da consulta
- Na cláusula **FROM** são especificados todas as tabelas que serão utilizadas para construção da consulta. O SQL realiza um produto cartesiano nas tabelas especificadas no FROM.
- A cláusula opcional **WHERE** especifica as condições de seleção dos registros resultados das tabelas mencionadas na cláusula FROM;

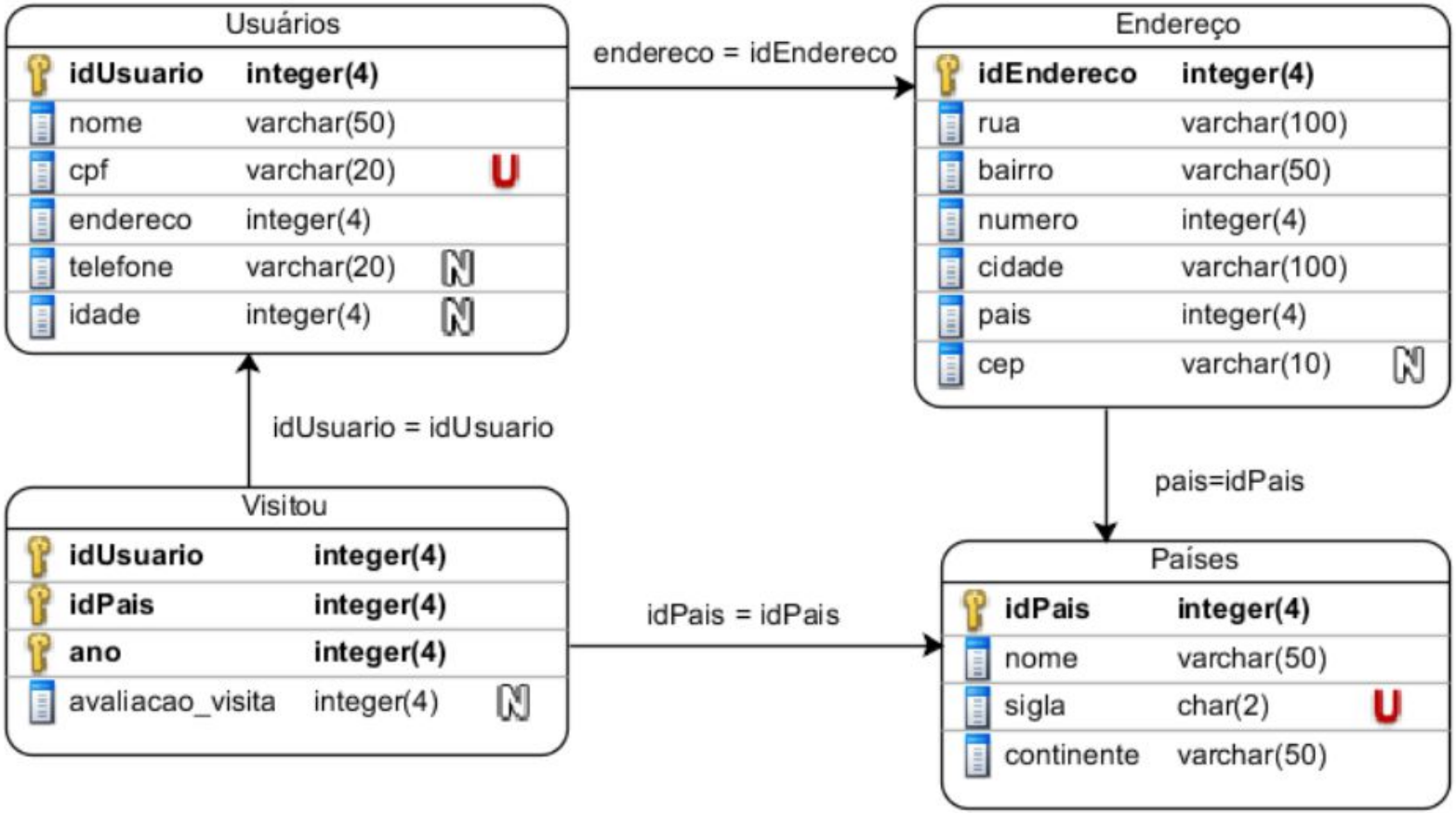
INSTITUTO FEDERAL
Goiano

SINTAXE COMPLETA SQL SELECT

```
SELECT [ ALL | DISTINCT [ ON ( expression [, ...] ) ] ]
    * | expression [ AS output_name ] [, ...]
    [ FROM from_item [, ...] ]
    [ WHERE condition ]
    [ GROUP BY expression [, ...] ]
    [ HAVING condition [, ...] ]
    [ { UNION | INTERSECT | EXCEPT } [ ALL ] select ]
    [ ORDER BY expression [ ASC | DESC | USING operator ] [, ...] ]
    [ LIMIT { count | ALL } ]
    [ OFFSET start ]
    [ FOR { UPDATE | SHARE } [ OF table_name [, ...] ] [ NOWAIT ] [...] ]
```

INSTITUTO FEDERAL
Goiano

EXEMPLO



SELECT SIMPLES - RECUPERAR TODOS USUÁRIOS

O resultado de uma consulta é um conjunto de linhas (registros) das tabelas definidas no FROM, sendo cada linha um conjunto de colunas (definidos na cláusula SELECT), que obedecem a condição imposta na cláusula WHERE.

A consulta possível é a recuperação de todos registros de uma única tabela, recuperando todas as colunas da tabela.

```
SELECT *
FROM Usuarios
```

SELECT SIMPLES - DEFININDO COLUNAS

Recupere todos o nome e a idade de todos os usuários.

Neste caso o resultado será composto apenas das colunas nome e idade de todos os usuários da tabela Usuarios.

```
SELECT nome, idade
FROM Usuarios
```

UTILIZANDO A CLÁUSULA WHERE

Selecione todos os usuários que possuem idade maior ou igual a 30 anos.

Neste caso teremos que fazer uma condição lógica utilizando a cláusula WHERE. As linhas da consulta inicial serão filtradas para que atendam a condição definida nesta cláusula.

```
SELECT nome, cpf, idade
FROM Usuarios as u
WHERE u.idade >= 30
```

ENTENDENDO O EXEMPLO ANTERIOR

Dentro da cláusula **SELECT** listamos todas as colunas que desejamos exibir no resultado ou utilizamos um asterisco (*) que indica que retornará todas as colunas.

A palavra reservada **AS** utilizada na clausula **FROM** é utilizada para criar uma “variável” da tabela ou conjunto de tabelas. (Usuarios as u)

A cláusula **WHERE** foi utilizada para filtrar o resultado e exibir apenas os usuários com idade superior a 30 anos (u.idade > 30).

JUNÇÃO BÁSICA DE TABELAS

JUNÇÃO BÁSICA DE TABELAS

A Junção básica de tabelas utiliza de uma condição realizada na cláusula WHERE e utiliza-se das chaves estrangeiras e primárias das tabelas envolvidas.

Exemplo: Encontre todos os usuários que visitaram o país com sigla BR.

```
SELECT u.nome, u.cpf
FROM Usuarios as u, Visitou as v, Paises as p
WHERE u.idUsuario = v.idUsuario AND
v.idPais = p.idPais AND
p.sigla = 'BR'
```

JUNÇÃO BÁSICA DE TABELAS

A expressão **u.idUsuario = v.idUsuario** de certa forma faz uma junção entre as tabelas Usuario e Visitou, pois serão filtradas somente os registros que terão o mesmo idUsuario dentro de ambas as tabelas.

Note que na tabela Visitou a coluna idUsuario é uma chave estrangeira que referencia a chave primária da tabela Usuario (idUsuario). **Assim temos garantia que o valor adiciona nesta coluna idUsuario da tabela Visitou realmente existe na tabela Usuário.**

ORDEM DE EXECUÇÃO

Esta consulta é computada na seguinte forma:

1. Compute o produto cartesiano das tabelas contidas no FROM;
2. Exclua as linhas no produto cartesiano que não satisfazem as condições de qualificação do WHERE;
3. Exclua as colunas que não aparecem no SELECT;

CONDIÇÕES DO WHERE

INSTITUTO FEDERAL
Goiano

CONDIÇÃO DE COMPARAÇÃO

Operador comparativo	Descrição
=	Igualdade
!=	Diferença
>	Maior
<	Menor
>=	Maior ou Igual
<=	Menor ou Igual

CONDIÇÃO LÓGICAS E COMPOSTAS

```
SELECT <lista-colunas-seleção>
FROM <lista-tabelas>
WHERE condição1 AND codição2 AND ...
```

```
SELECT <lista-colunas-seleção>
FROM <lista-tabelas>
WHERE condição1 OR codição2 OR ...
```

INSTITUTO FEDERAL
Goiano

CONDIÇÕES NOT

A Condição NOT é utilizada para inverter o resultado de uma expressão lógica, negando o resultado da condição. Caso a condição seja verdadeira, será retornado falso e vice-versa.

```
SELECT nome, cpf
FROM Usuarios as u
WHERE NOT(u.idade >= 30)
```

CONDIÇÃO IS NULL

Sabe-se que todas as colunas que não têm valor inicializado são nulas em banco de dados SQL. Logo, esse comando é utilizado para saber se o conteúdo da coluna foi ou não inicializado.

```
SELECT nome, cpf
FROM Usuarios as u
WHERE u.telefone IS NULL
```

Recupera todos usuários que **não tem** um telefone definido.

CONDIÇÃO IS NOT NULL

Esse comando inverte a condição anterior, logo é utilizado para saber se o conteúdo da coluna está preenchido.

```
SELECT nome, cpf
FROM Usuarios as u
WHERE u.telefone IS NOT NULL
```

Recupera todos usuários que **tem** um telefone definido.

INSTITUTO FEDERAL
Goiano

CONDIÇÃO IN

A condição IN permite comparar o valor de uma coluna com um conjunto de valores. Normalmente utiliza-se o IN para substituir uma série de comparações seguidas da cláusula **OR** para uma mesma coluna.

```
SELECT nome
FROM Paises as p
WHERE p.sigla IN ('BR', 'AR', 'CL')
```

INSTITUTO FEDERAL
Goiano

BETWEEN

Esse operador é utilizado para determinar um intervalo de busca. Assim, sempre que se fazer necessário realizar buscas que indiquem um intervalo de números e/ou datas pode-se utilizar o comando **between** para simplificar a consulta.

- Normalmente este comando simplifica uma consulta mais extensa utilizando o operador AND

```
SELECT *
FROM Visitou as v
WHERE v.avaliacao_visita BETWEEN 7 AND 10
```

INSTITUTO FEDERAL
Goiano

NOT BETWEEN

Recupera todos as visitas que não estão no intervalo determinado pelo operador BETWEEN.

```
SELECT *
FROM Visitou as v
WHERE v.avaliacao_visita NOT BETWEEN 6 AND 8
```

OPERADOR LIKE

O operador LIKE permite comprarar cadeias de caracteres utilizando padrões de comparação (*wildcards*) para um ou mais caracteres.

- Normalmente o caracter **%** substitui zero, um ou mais caracteres e **_** substitui um caracter.

Utilizando a combinação desses caracteres especiais com o que se deve localizar, pode-se conseguir uma variedade de expressões, tornando a pesquisa mais precisa.

INSTITUTO FEDERAL
Goiano

OPERADOR LIKE

Expressão	Descrição
LIKE 'A%'	Todas as palavras que iniciem com a letra A
LIKE '%A'	Todas as palavras que terminam com a letra A
LIKE '%A%'	Todas que tenham a letra A em qualquer posição.
LIKE 'A_'	String de dois caracteres que tenham a primeira letra A e o segundo caracter qualquer outro
LIKE '_A'	String de dois caracteres que tenham a primeira letra qualquer um e o segundo caracter deve ser a letra A
LIKE '_A_'	String com três caracteres e o segundo caractere é a letra A.
LIKE '%A_'	Todos que tenham a letra A na penúltima posição e a última seja qualquer outro caracter
LIKE '_A%'	Todos que tenham a letra A na segunda posição e o primeiro caracter seja qualquer um.

INSTITUTO FEDERAL
Goiano

OPERADOR LIKE

```
SELECT nome
FROM Usuarios as u
WHERE u.nome LIKE 'MA%'
```

```
SELECT nome
FROM Usuarios as u
WHERE u.nome LIKE '%AR%'
```