Banco de Dados Relacional

PROFº DR° FRANCISCO DOUGLAS. L ABREU

MSC E PHD EM ENGENHARIA BIOMÉDICA
ESP. EM ANÁLISE DE DADOS

FRANCISCO.ABREU@FATEC.SP.GOV.BR

LINGUAGEM DE CONSULTA DE DADOS (DQL)

Linguagem de Consulta de Dados (DQL)

A **DQL** (*Data Query Language*) é a linguagem que viabiliza o acesso dos dados de forma compatível ao modelo de dados apropriado. Anteriormente essa linguagem estava acopladas na DML (*Data Manipulation Language)*, mas devido a extensão do comando *Select*, houve-se a necessidade de separa-la.

A **DQL** possui dois tipos:

- **DQL Procedural**: exige que o usuário especifique quais dados são necessários, e como obtê-los;
- **DQL Não-Procedural**: exige que o usuário especifique quais dados são necessários, sem especificar como obtê-los. Mais fácil de aprender a usar. Pode gerar códigos não muito eficientes

Selecionar Dados (SELECT)

O comando *SELECT* **consulta uma ou mais tuplas (linhas)** especificas em uma tabela do banco de dados.

SINTAXE

```
SELECT <RETORNO>
FROM <NOME_DO_BANCO>.<NOME_DA_TABELA>
WHERE [CONDIÇÃO];
```

Selecionar Dados (SELECT)

Parâmetros:

<NOME_DA_TABELA>
O nome da tabela onde os dados serão consultados;

<RETORNO>
Campo da tabela que será retornado para visualização. Se eu colocar o * significa que retornará todos os campos da tabela.;

<WHERE CONDIÇÃO>
Condição que fará um filtro nos dados, ou seja, os campos só serão mostrados se atenderem a condição imposta

ATIVIDADE EM AULA

CRIE UMA TABELA:

```
CREATE TABLE ALUNO (
  ID INT,
  NOME VARCHAR (50),
  MEDIA DOUBLE,
  ANO DATE
);
```

INSERE DADOS NA TABELA:

```
INSERT INTO ALUNO VALUES(1, "João", 5, "2020" );
INSERT INTO ALUNO VALUES(2, "Maria",10, "2020" );
INSERT INTO ALUNO VALUES(3, "Aline", 2, "2011" );
```

WHERE

A clausula **WHERE** cria uma condição para que a operação seja efetuada caso o registro atenda a condição imposta.

Ou seja, ela pode ser utilizada para Filtrar os dados de uma consulta (**SELECT**), para impor uma restrição em uma exclusão (**DELETE**) e para impor restrições nos registros atualizados (**UPDATE**).

As condições criadas podem ser combinadas com alguns comandos ou com expressões simples:

OPERADORES

SIMBOLO	SIGNIFICADO
=	IGUAL A (IGUALDADE)
<>	DIFERENTE A (DESIGUALDADE)
<	MENOR QUE
>	MAIOR QUE
>=	MAIOR OU IGUAL A QUE
<=	MENOR OU IGUAL A QUE

AND

Lógica Booleana E

Ex: SELECT * FROM Aluno WHERE ano=2020 AND nome='João';

Resultado Esperado: Apenas uma linha (tupla) será exibida.
Explicativo: Será exibido apenas a coluna que atenda a todas condições impostas (Todos os valores tem que ser verdadeiros/satisfeitas) :

OR

Lógica Booleana OU

Ex: SELECT * FROM Aluno WHERE ano=2020 OR nome='João';

Resultado Esperado: Duas linhas (tupla) serão exibidas.
Explicativo: : A verificação das condições só será verdadeira quando qualquer uma das condições forem satisfeitas.

FUNÇÕES

LIKE BINARY

Comparação se existe o valor no campo testado.

Ex.: SELECT * FROM Aluno WHERE nome like binary 'João%';

Resultado Esperado: Apenas uma linha (tupla) será exibida.
Explicativo: No exemplo utilizamos o % para comparar os registros que comecem com João.. Podemos utilizar o % como um "coringa" onde não sabemos (ou não importa) o inicio ou o fim da palavra a ser localizada. É importante resaltar que o comando like é *case sensitive*, ou seja, faz diferenciação entre letras maiúsculas e minúsculas.

LIKE

Comparação se existe o valor no campo testado.

Ex.: SELECT * FROM Aluno WHERE nome like 'João';

Resultado Esperado: Apenas uma linha (tupla) será exibida.
Explicativo: o comando LIKE é similar operador = (igualdade), porém não é *case sensitive*

FUNÇÕES

IN

Verifica se o conteúdo está dentro de uma lista de valores passados. .

Ex: SELECT * FROM Aluno WHERE nome IN (‘Ana Lucia’, ‘Ana Claudia’, ‘Maria’);

Resultado Esperado: Apenas uma linha (tupla) será exibida.
Explicativo: Exibirá apenas os registros dos alunos que possuíram nome Ana Lucia, Ana Claudia ou Ana Maria.

BETWEEN

Verifica se o conteúdo está entre as duas condições imposta

Ex: SELECT * FROM Aluno WHERE media BETWEEN 3.0 AND 5.0;

Resultado Esperado: Apenas uma linha (tupla) será exibida.
Explicativo: Exibirá os registros de alunos que possuem média maior ou igual a 3 e menor ou igual a 5.

FUNÇÕES AGREGADAS

COUTN()

Retorna o número de linhas

Ex: : SELECT COUNT(id) FROM aluno;

SUM()

Retorna o valor da Soma

Ex: :SELECT SUM(media) FROM aluno;

MAX()

Retorna o maior Valor;

Ex: SELECT MAX(media) FROM aluno;

FUNÇÕES AGREGADAS

AVG()

Retorna o valor Médio

Ex: SELECT AVG(media) FROM aluno;

MIN()

Retorna o menor Valor;

Ex: SELECT MIN(media) FROM aluno;

GROUP BY

O uso da cláusula GROUP BY faz com que os registros sejam retornados na ordem crescente do(s) campo(s) usado(s) para agrupar o resultado da função agregada. Mais um detalhe, todos os campos na cláusula SELECT devem aparecer após a cláusula GROUP BY separados por vírgula, ou você terá um erro de sintaxe na consulta.

Ex: SELECT SUM(media) FROM aluno GROUP BY media;

FUNÇÕES AGREGADAS

HAVING

A cláusula HAVING vem a complementar a cláusula GROUP BY. Quando usamos GROUP BY, os registros retornados serão todos os que satisfizerem ao critério informado após a palavra WHERE. Porém, podemos querer fazer uma segunda filtragem após termos os resultados dos cálculos das funções agregadas. Neste caso, usamos a cláusula HAVING para fazer esta filtragem posterior.

Ex: **SELECT SUM**(media) **FROM** aluno **GROUP BY** media
HAVING SUM(media) < 2;

Outras Funções

DISTINCT

Em uma tabela, algumas das colunas podem conter valores duplicados, para remover os registros duplicados e exibir apenas uma ocorrência de cada valor utiliza-se a cláusula DISTINCT. Ex: Selecione todos os clientes que fizeram compras no ano de 2013.

Ex: SELECT DISTINCT nome FROM aluno WHERE date_part ('year' ,dt_compra)= 2013

NOW()

Retorna a data e hora do sistema

Ex: SELECT NOW();

IS NULL

Retorna campos vazios ou não vazios ao colocar NOT;

Ex : SELECT * FROM aluno WHERE nome IS NULL;

EX: SELECT * FROM aluno WHERE nome IS NOT NULL;.

Outras Funções

LOWER()

Esta função converte o valor de um campo com letras minúsculas

Ex: SELECT LOWER(nome) FROM aluno;

UPPER()

Esta função converte o valor de um campo com letras maiúsculas

Ex: SELECT UPPER(nome) FROM aluno;

ROUND( )

Esta função arredonda um campo numérico para um número de casas decimais especificado.

Ex : SELECT ROUND(media,2) FROM aluno;

.

Outras Funções

SQRT()

Esta função extrair a raiz quadrada do valor especificado

Ex: SELECT SQRT(media) FROM aluno;

TRUNCATE()

Esta função retorna a parte inteira de um valor especificado

Ex: SELECT TRUNC(media,2) FROM aluno;

ORDER BY

Esta função organiza os valores de forma ascendente (ASC) ou descendente (DESC)

Ex: SELECT * FROM aluno ORDER BY nome ASC;

Ex: SELECT * FROM aluno ORDER BY nome DESC;

Alguma Dúvida?