

NExT – Nova Experiência de Trabalho

Módulo – Banco de Dados

Gabrielle K. Canalle
gkc@cesar.school

Natacha Targino
ntrs@cesar.school



Linguagem SQL

Structured Query
Language



DQL

Data Query Language (DQL)

Select

- Permite realizar consultas no banco de dados
 - A cláusula **from** indica de quais tabelas as informações serão recuperadas.
 - A cláusula **where** filtra as informações através da condição

Sintaxe

Selecionando todos os atributos

Select * from <nomeTabela>;

Select * from aluno;

codigo	nome	idade	telefone
001	Maria	18	9999
002	José	23	4444
003	João	45	2222
004	Pedro	13	6454
005	Luiza	37	2333

Data Query Language (DQL)

Select

Selecione nome e idade dos alunos

```
Select nome, idade  
from aluno;
```

nome	idade
Maria	18
José	23
João	45
Pedro	13
Luiza	37

Data Query Language (DQL)

Select - select-from-where

Selecionando tuplas com condição

Select <nomeColunas>

from <nomeTabelas>

where **<condicao>**;

<nomeAtributo> <operador> <valor>

- constante
- variável
- consulta aninhada

Lógicos	
AND	e
OR	ou
NOT	não

Relacionais			
<> ou !=	Diferente	=	Igual a
>	Maior que	>=	Maior ou igual a
<	Menor que	<=	Menor ou igual a

Data Query Language (DQL)

Select

Selecione a idade de João

```
Select idade  
from aluno  
where nome =  
"João";
```

codigo	nome	idade	telefone
001	Maria	18	9999
002	José	23	4444
003	João	45	2222
004	Pedro	13	6454
005	Luiza	37	2333

idade
45

Data Query Language (DQL)

Select

Selecione todos os alunos maiores de idade

```
Select *  
from aluno  
where idade > 18;
```

codigo	nome	idade	telefone
001	Maria	18	9999
002	José	23	4444
003	João	45	2222
005	Luiza	37	2333


Data Query Language (DQL)

Renomeando atributos

- Utiliza-se a palavra chave **as**

Select nome as
"NomeAluno", idade
from aluno;

codigo	nome	idade	telefone
001	Maria	18	9999
002	José	23	4444
003	João	45	2222
005	Luiza	37	2333



NomeAluno	idade
Maria	18
José	23
João	45
Luiza	37

Data Query Language (DQL) - Operadores

BETWEEN e NOT BETWEEN

- Selecionar valores em um determinado intervalo

Sintaxe

where nome_coluna **between/not between** <valor1> **and** <valor2>;

```
Select * from aluno  
where idade between 10 and 20;
```

codigo	nome	idade	telefone
001	Maria	18	9999
004	Pedro	13	6454

codigo	nome	idade	telefone
001	Maria	18	9999
002	José	23	4444
003	João	45	2222
004	Pedro	13	6454
005	Luiza	37	2333

Data Query Language (DQL) – Operadores

LIKE e NOT LIKE

- Esses operadores só trabalham sobre colunas do tipo carácter.
- Podem fazer uso de curingas:
 - % -> substitui um texto (um, nenhum ou vários caracteres)
 - _ -> substitui um carácter
- **Exemplo:**
 - 'LAPIS%' = LAPIS PRETO, LAPIS BORRACHA
 - 'T_M' = Tim, Tom

Sintaxe


where nome_coluna **like/not like** <padrão>;

Data Query Language (DQL) - Operadores

LIKE e NOT LIKE

Selecione os alunos cujo nome iniciam com a letra "J"

```
Select *  
from aluno  
where nome like 'J%';
```



codigo	nome	idade	telefone
002	José	23	4444
003	João	45	2222

codigo	nome	idade	telefone
001	Maria	18	9999
002	José	23	4444
003	João	45	2222
004	Pedro	13	6454
005	Luiza	37	2333

Data Query Language (DQL) - Operadores

IN e NOT IN

- Selecionar os dados que estão contidos em um conjunto de valores

Sintaxe

where nome_coluna **in** (<valores>;

```
Select telefone  
from aluno  
where nome in  
('Maria', 'Luiza');
```



telefone
9999
2333

codigo	nome	idade	telefone
001	Maria	18	9999
002	José	23	4444
003	João	45	2222
004	Pedro	13	6454
005	Luiza	37	2333


Data Query Language (DQL) - Operadores

IS NULL e IS NOT NULL

Sintaxe

where nome_coluna **is null**;

Select codigo, nome
from aluno
where telefone is null;



codigo	nome
006	Carmem

codigo	nome	idade	telefone
001	Maria	18	9999
002	José	23	4444
003	João	45	2222
004	Pedro	13	6454
005	Luiza	37	2333
006	Carmem	20	

Data Query Language (DQL) - Ordenação

ORDER BY

- Utilizado quando há necessidade de ordenar os dados recuperados é utilizada

Sintaxe

select <nome(s) da(s) coluna(s)>

from <tabela ou lista de tabelas>


where <condições>

order by nome_coluna ou número da coluna ASC ou DESC;

Data Query Language (DQL) - Ordenação

ORDER BY

```
Select nome  
from aluno  
order by nome;
```



nome
João
José
Luiza
Maria
Pedro


codigo	nome	idade	telefone
001	Maria	18	9999
002	José	23	4444
003	João	45	2222
004	Pedro	13	6454
005	Luiza	37	2333

Data Query Language (DQL) - Eliminando repetição

DISTINCT

Selecione os códigos de departamento que existem funcionários alocados

```
Select distinct cod_dep  
from funcionario;
```



cod_dep
001
002
003


codigo	nome	cod_dep
001	Maria	001
002	José	001
003	João	002
004	Pedro	002
005	Luiza	002
005	Luiza	003

Data Query Language (DQL) - Alias

ALIAS

- Utilizado para "apelidar" uma tabela, campo ou expressão
 - melhora a legibilidade do comando

```
Select a.nome  
from aluno a;
```



nome
Maria
José
João
Pedro
Luiza

codigo	nome	idade	telefone
001	Maria	18	9999
002	José	23	4444
003	João	45	2222
004	Pedro	13	6454
005	Luiza	37	2333

Data Query Language (DQL) - Atributos calculados

- Pode-se utilizar expressões aritméticas na cláusula select

Mostrar o nome e a idade dos alunos daqui a dois anos.

```
select nome, idade + 2  
as "Idade em 2024"  
from aluno;
```

nome	Idade em 2024
Maria	20
José	25
João	47
Pedro	15
Luiza	39

codigo	nome	idade	telefone
001	Maria	18	9999
002	José	23	4444
003	João	45	2222
004	Pedro	13	6454
005	Luiza	37	2333


as é utilizado para nomear/renomear uma
coluna contida no select

Data Query Language (DQL) - Funções

MAX, MIN, SUM, AVG, COUNT

- São funções embutidas do SQL, criadas (funções armazenadas) ou pré-definidas do SGBD
 - Computam um valor único a partir de um conjunto de valores de um atributo

```
select MIN(idade), MAX(idade)  
from aluno;
```



MIN(idade)	MAX(idade)
13	45

codigo	nome	idade	telefone
001	Maria	18	9999
002	José	23	4444
003	João	45	2222
004	Pedro	13	6454
005	Luiza	37	2333

Data Query Language (DQL) - Funções

MAX, MIN, SUM, AVG, COUNT

```
select AVG(idade) from aluno;
```

AVG(idade)

27

```
select COUNT(*) from aluno  
where telefone = 2222;
```

COUNT(*)

1

```
select SUM(idade) from aluno;
```

SUM(idade)

136

codigo	nome	idade	telefone
001	Maria	18	9999
002	José	23	4444
003	João	45	2222
004	Pedro	13	6454
005	Luiza	37	2333

Data Query Language (DQL) - Agrupamento

Agrupamentos

- Utilizado quando é preciso definir subgrupos para obter o resultado das funções agregadas
 - São definidos com base em algum atributo

GROUP BY


- A cláusula GROUP BY tem a finalidade de agrupar tuplas especificando os atributos de agrupamento
- Esses atributos também devem aparecer na cláusula SELECT.
- Somente as funções de agregação podem aparecer no SELECT e não aparecer no GROUP BY

Data Query Language (DQL) - Agrupamento

GROUP BY

Selecione a quantidade de funcionários por departamento

```
select cod_dep, COUNT(*)  
from funcionario  
group by cod_dep order by  
cod_dep;
```



cod_dep	COUNT(*)
001	2
002	3
003	1

codigo	nome	cod_dep
001	Maria	001
002	José	001
003	João	002
004	Pedro	002
005	Luiza	002
005	Luiza	003

Data Query Language (DQL) - Agrupamento

HAVING

- Utilizado para aplicar condições sobre cada grupo, ou seja, recuperar valores das funções de agregação e do agrupamento somente para grupos que satisfazem certas condições
 - A cláusula HAVING deve ser utilizada em conjunto com o GROUP BY
 - Somente os grupos que satisfazem a condição do HAVING são recuperados


Data Query Language (DQL) - Agrupamento

HAVING

Selecione os departamentos cuja quantidade de funcionários é maior que 2;

```
select cod_dep, COUNT(*)  
from funcionario  
group by cod_dep HAVING COUNT(*) >= 2;
```

codigo	nome	cod_dep
001	Maria	001
002	José	001
003	João	002
004	Pedro	002
005	Luiza	002
005	Luiza	003



cod_dep	COUNT(*)
001	2
002	3

Data Query Language (DQL) - Agrupamento

HAVING

O predicado de **having** só deve envolver as funções de agregação ou os campos agregados.

```
select cod_dep, COUNT(*)  
from funcionario  
group by cod_dep HAVING idade >= 20;
```



ERRO!

codigo	nome	idade	cod_dep
001	Maria	23	001
002	José	34	001
003	João	67	002
004	Pedro	54	002
005	Luiza	32	002
005	Luiza	28	003

Data Query Language (DQL) - Agrupamento

Ordem de processamento

1. As linhas que satisfazem o **where** são selecionadas
2. Então os grupos são formados de acordo com a cláusula **group by** e as **funções de agregação** são aplicadas para cada grupo
3. Os grupos que não satisfazem a condição do **having** são descartados
4. Mostra-se os valores para as colunas e agregações listadas na cláusula **select**

Data Query Language (DQL) – Junção de Tabelas

- Permite juntar duas ou mais tabelas, ligando-as através da chave Primária de uma e a chave Estrangeira de outra tabela
 - necessidade de acessar mais de uma tabela na consulta
- Uma junção pode ser **implícita** ou **explícita**

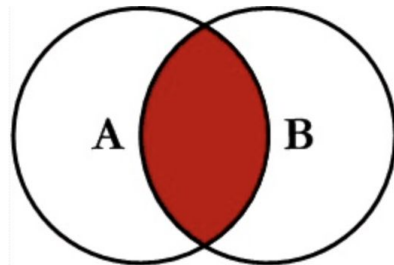
Data Query Language (DQL) - Junção de Tabelas

IMPLÍCITA - Ocorre quando especificamos o relacionamento entre as tabelas na cláusula WHERE

```
select <lista de atributos>  
from tabela1, tabela2  
where tabela1.campoPK = tabela2.campoFK;
```

EXPLÍCITA - Ocorre quando utilizamos a cláusula **JOIN/INNER JOIN**

```
select <lista de atributos>  
from tabela1 JOIN tabela2 ON  
tabela1.campoPK = tabela2.campoFK;
```



Data Query Language (DQL) - Junção de Tabelas

Apresente o nome dos alunos e a descrição e carga horária do curso de cada um

select a.nome, c.descricao, c.carga_hr

from aluno a, curso c

where a.cod_curso = c.cod_curso;

ou

select a.nome, c.descricao, c.carga_hr

from aluno a **JOIN** curso c **ON**

a.cod_curso = c.cod_curso;

CURSO

codigo	descricao	carga_hr
001	Inglês	100
002	Alemão	120
003	Francês	90
004	Italiano	100

ALUNO

matricula	nome	telefone	cod_curso
001	Maria	9999	003
002	José	4444	003
003	João	2222	004
004	Pedro	6454	001

Data Query Language (DQL) – Junção de Tabelas

Para os alunos cujo nome inicia com P, apresente o telefone e a descrição do curso

select a.nome, a.telefone, c.descricao

from aluno a, curso c

where a.cod_curso = c.cod_curso

and a.nome like 'P%';

ou

select a.nome, c.nome, c.carga_hr

from aluno a **JOIN** curso c **ON**

a.cod_curso = c.cod_curso

where a.nome like 'P%';

CURSO

codigo	descricao	carga_hr
001	Inglês	100
002	Alemão	120
003	Francês	90
004	Italiano	100

ALUNO

matricula	nome	telefone	cod_curso
001	Maria	9999	003
002	José	4444	003
003	João	2222	004
004	Pedro	6454	001

Data Query Language (DQL) – Subconsultas

- Uma sub-consulta ou consulta aninhada é um comando **SELECT** que foi incluído em outro comando SELECT, UPDATE, INSERT, DELETE ou dentro de outra subconsulta de forma encadeada e contida no mesmo comando SQL
- Utilizada quando é necessário que valores do BD sejam obtidos e então usados numa condição ou requerem um conjunto de linhas

Data Query Language (DQL) - Subconsultas

- **ESCALAR**

- Retornam um único valor

- **ÚNICA LINHA**

- Retornam várias colunas, mas apenas uma única linha é obtida

- **TABELA**

- Retornam uma ou mais colunas e múltiplas linhas

Data Query Language (DQL) - Subconsultas

Única linha

```
select codigo, nome  
from aluno  
where codigo =
```

```
(select codigo  
from aluno  
where nome = '%Maria');
```

= 001

codigo	nome	idade	cod_dep
001	Ana Maria	23	001
002	José	34	001
003	João	67	002

codigo	nome
001	Ana Maria

Data Query Language (DQL) - Subconsultas

Escalar

```
select nome, idade  
from aluno  
where idade >
```

(select avg(idade)
from aluno);

= 41

codigo	nome	idade	cod_dep
001	Ana Maria	23	001
002	José	34	001
003	João	67	002

nome	idade
João	67

Data Query Language (DQL) - Subconsultas

IN/NOT IN

- Permite especificar múltiplos valores na cláusula WHERE
- A cláusula é utilizada para comparar um valor a uma lista ou subconsulta
- Procura um valor em um subconjunto.

Data Query Language (DQL) - Subconsultas

IN/NOT IN

```
select nome  
from funcionario
```

```
WHERE cod_cargo IN (select codigo  
                     from cargo  
                     where nome = 'Programador');
```

CARGO

codigo	nome
001	Programador
002	Secretaria
003	Ux Designer
004	Designer

FUNCIONÁRIO

codigo	nome	cod_cargo
001	Maria	002
002	João	001
003	José	004
004	Pedro	001

Data Query Language (DQL) - Subconsultas

ANY/SOME

- São sinônimos
- Retornam *true* se o valor de um atributo for igual a *algum valor* do conjunto retornado pela subconsulta

Data Query Language (DQL) - Subconsultas

ANY/SOME

CARGO

codigo	nome
001	Programador
002	Secretaria
003	Ux Designer
004	Designer

FUNCIONÁRIO

codigo	nome	idade	cod_cargo
001	Maria	23	002
002	João	34	001
003	José	67	004
004	Pedro	41	001

select nome
from funcionario

WHERE idade != **some** (select idade
from funcionario
where cod_cargo = 001);

vai retornar o nome dos
funcionários cuja idade seja
diferente de 34 ou 41

= 34, 41

nome
Maria
José

Data Query Language (DQL) - Subconsultas

ALL

- A condição de comparação ALL é usada justamente para comparar um valor à uma lista ou subconsulta

select nome, idade

from funcionario

WHERE idade > **all**(select idade
from funcionario
where cod_cargo = 001);

codigo	nome	idade	cod_cargo
001	Maria	23	002
002	João	34	001
003	José	67	004
004	Pedro	41	001

nome	idade
José	67

Data Query Language (DQL) - Subconsultas correlacionadas

Subconsultas correlacionadas

- Sempre que uma condição na cláusula WHERE de uma consulta aninhada referencia algum atributo de uma relação em uma consulta externa, dizemos que essas duas consultas estão **correlacionadas**
- Utilizada quando você tem interesse em saber se *ALGUMA* linha é retornada, mas não se importa com a *QUANTIDADE* retornada
- É executada uma vez para cada linha na consulta externa
 - <> das não correlacionadas que é executada uma vez antes da execução da consulta externa

Data Query Language (DQL) - Subconsultas

EXISTS/NOT EXISTS

- Foram projetadas para uso apenas com subconsultas
- Usada para verificar se o resultado de uma consulta aninhada é vazia ou não
- O resultado é um valor booleano:
 - True se o resultado da consulta aninhada tiver pelo menos uma tupla
 - False caso contrário

Data Query Language (DQL) - Subconsultas correlacionadas

EXISTS/NOT EXISTS

Quais os cursos que não tem nenhum aluno matriculado?

```
select c.cod_curso, c.nome  
from curso c
```

```
where not exists (select * from aluno a  
                  where  
                  c.cod_curso = a.cod_curso);
```

codigo	descricao
002	Alemão

CURSO		
codigo	descricao	carga_hr
001	Inglês	100
002	Alemão	120
003	Francês	90
004	Italiano	100

ALUNO

matricula	nome	telefone	cod_curso
001	Maria	9999	003
002	José	4444	003
003	João	2222	004
004	Pedro	6454	001

Vamos praticar?

Vamos para as rooms?

- Arquivo com as atividades de linguagem SQL disponível no classroom
- Ficaremos passando nas salas;
- Arquivo com as correções será disponibilizado ao final

Dúvidas?

Gabrielle K. Canalle
gkc@cesar.school

Natacha Targino
ntrsb@cesar.school

