INSTITUTO FEDERAL
RIO GRANDE DO SUL

Consultas SQL

TIAGO G MORAES

Roteiro

- Revisão DML
  - Insert, delete update e select
- Consultas SQL
  - Eliminando resultados duplicados
  - Renomeando atributos e relações
  - Utilizando WHERE
  - Ordenando resultados
  - Agrupando dados

Revisão DML

- DML: Data Manipulation Language
  - Manipulação dados: inserir, alterar, deletar e consultar dados
- Inserindo dados
  - INSERT:

```
INSERT INTO <nome_tabela> [(<col1>,<col2>,...)]
VALUES (<val1>, <val2>,...)
```

- Variantes:
  - Mais linhas

```
INSERT INTO pessoa VALUES (1, 'João'), (2, 'Maria')
```

  - A partir de uma tabela

```
INSERT INTO aluno (cod, nome) (SELECT * FROM pessoa)
```

→[] = opcionalidade →<> = preencher com algum valor →{} = escolha um dos itens

Revisão DML

- DML: Data Manipulation Language
  - Manipulação dados: inserir, alterar, deletar e consultar dados
- Excluindo dados
  - DELETE

```
DELETE FROM <tabela> [WHERE <condição>]
```

  - Cuidado para não deletar todas linhas de uma tabela. Para isso use o where

→[] = opcionalidade →<> = preencher com algum valor →{} = escolha um dos itens

Revisão DML

- Atualizando dados
  - Update:

```
UPDATE <nome_tabela> SET <col1>=<val1>,...
<colN>=<valN> [WHERE <CONDIÇÃO>]
```

- Buscando dados
  - Select:

```
SELECT <colunas> FROM <tabela> [WHERE <condição>]
```

→[] = opcionalidade →<> = preencher com algum valor →{} = escolha um dos itens

  - Porém esse formato de consulta é limitado....

Resultados duplicados

- A relação resposta a uma *SELECT* pode conter linhas iguais (duplicadas)
  - Eliminar duplicadas → "distinct"
    - Exemplo:

```
SELECT distinct cargo
FROM empregado
```

cargo
Analista
Programador
Diretor

codEmp	nome	Salario	cargo
001	Anderson	5000	Analista
002	Rui	1500	Programador
003	Carlos	7000	Diretor
004	João	3000	Analista

Resultados duplicados

- A relação resposta a uma *SELECT* pode conter linhas iguais (duplicadas)
  - Eliminar duplicadas → "distinct"
    - Exemplo:

```
SELECT distinct cargo
FROM empregado
```

cargo
Analista
Programador
Diretor

  - Manter duplicadas → "all"
    - Exemplo:

```
select all cargo from
empregado
```

cargo
Analista
Programador
Diretor
Analista

Resultados duplicados

- A relação resposta a uma *SELECT* pode conter linhas iguais (duplicadas)
  - Eliminar duplicadas → "distinct"
    - Exemplo:

```
SELECT distinct cargo
FROM empregado
```

cargo
Analista
Programador
Diretor

  - Manter duplicadas → "all"
    - Exemplo:

```
select all cargo from
empregado
```

cargo
Analista
Programador
Diretor
Analista

OBS: "**all**" é o padrão, por isso não necessita ser utilizado

Renomeação

- A renomeação é feita pelo com a cláusula "as"

nome_antigo ***AS*** *nome_novo*

- Pode ser renomeado:
  - Atributos
    - `SELECT nome AS funcionario FROM empregado`
  - Relações
    - `SELECT e.nome FROM empregado AS e`
    - `empregado AS e(cod,nom,sal,car)`
    - No POSTGRES o uso do "AS" é opcional → `empregado e`

- ✓ Evita ambiguidades
- ✓ Melhora a apresentação da resposta
- ✓ Facilita a construção da consulta

Condição do *where*

- A condição cláusula WHERE pode envolver diversos operadores e funções:
  - Operadores relacionais
    - =, <> (!=), <, <=, > e >=
  - Operadores lógicos
    - **and**, **or**, **not**

Condição do *where*

- A condição cláusula WHERE pode envolver diversos operadores e funções:
  - Operadores relacionais
    - =, <> (!=), <, <=, > e >=
  - Operadores lógicos
    - **and**, **or**, **not**
  - Operador "*between*" (define um intervalo)
    - `"idade between 10 and 20"` equivale a `"idade >= 10 and idade <= 20"`

Condição do *where*

- Comparações com texto
  - Operador "="
    - Exemplo → `nome = 'Silva'`

OBS: Função **upper()** e **lower()** para maiúsculas e minúsculas

Condição do *where*

- Comparações com texto
  - Operador "="
    - Exemplo → `nome = 'Silva'`
  - Operador "like"
    - para testar semelhança de *strings*
    - Caracteres especiais:
    - '_' qualquer carácter
    - '%' qualquer *substring*
    - Exemplo → nome like '_a%' → Cabeça de teta, Carlos, Fabiane Silva

Condição do *where*

- Comparações com texto
  - Operador "="
    - Exemplo → `nome = 'Silva'`
  - Operador "like"
    - para testar semelhança de *strings*
    - Caracteres especiais:
    - '_' qualquer carácter
    - '%' qualquer *substring*
    - Exemplo → nome like '_a%' → Cabeça de teta, Carlos, Fabiane Silva

ilike → like que desconsiderar maiúsculas e minúsculas

Condição do *where*

- Outras funções texto:
  - Concatenar: ||
    - `SELECT 'ola' || ' mundo'` → `'ola mundo'`
  - Tamanho string: char_length(string)
    - `SELECT char_length('mundo')` → 5
  - Pegar parte de string: substring(string, offset, limit)
    - `SELECT substring('mundo',2,2)` → 'ndo'

Condição do *where*

- Exemplo: empregados com **salário entre 2000 e 4000** ou **maior que 6000** e com **"Silva"** no **nome**

```
Select * from empregado where
salario between 2000 and 4000
or salario>6000
and nome like '%Silva%'
```

codEmp	nome	Salario	cargo
001	Anderson Silva	5000	Analista
002	Rui Silva	1500	Programador
003	Carlos Silva	7000	Diretor
004	João Mora	3000	Analista

Condição do *where*

- Exemplo: empregados com **salário entre 2000 e 4000** ou **maior que 6000** e com **"Silva"** no **nome**

```
Select * from empregado where
salario between 2000 and 4000
or salario>6000
and nome like '%Silva%'
```

codEmp	nome	Salario	cargo
001	Anderson Silva	5000	Analista
002	Rui Silva	1500	Programador
003	Carlos Silva	7000	Diretor
004	João Mora	3000	Analista

Condição do *where*

- Exemplo: empregados com **salário entre 2000 e 4000** ou **maior que 6000** e com **"Silva"** no **nome**

```
Select * from empregado where
salario between 2000 and 4000
or salario>6000
and nome like '%Silva%'
```

codEmp	nome	Salario	cargo
001	Anderson Silva	5000	Analista
002	Rui Silva	1500	Programador
003	Carlos Silva	7000	Diretor
004	João Mora	3000	Analista

Condição do *where*

- Exemplo: empregados com **salário entre 2000 e 4000** ou **maior que 6000** e com **"Silva"** no **nome**

```
Select * from empregado where
salario between 2000 and 4000
or salario>6000
and nome like '%Silva%'
```

codEmp	nome	Salario	cargo
001	Anderson Silva	5000	Analista
002	Rui Silva	1500	Programador
003	Carlos Silva	7000	Diretor
004	João Mora	3000	Analista

Condição do *where*

- Exemplo: empregados com **salário entre 2000 e 4000** ou **maior que 6000** e com **"Silva"** no **nome**

```
Select * from empregado where
salario between 2000 and 4000
or salario>6000
and nome like '%Silva%'
```

Cuidado com valores nulos
null>1000?
- **Is null**
- **Is not null**

Ordenando resultados

- Order by → serve para ordenar por um ou mais campos
  - Utilizado ao final da *select*
  - A ordenação é feita pelo primeiro campo passado
    - Em caso de empate é ordenado pelo segundo campo e assim sucessivamente

Ordenando resultados

- Order by → serve para ordenar por um ou mais campos
  - Utilizado ao final da *select*
  - A ordenação é feita pelo primeiro campo passado
    - Em caso de empate é ordenado pelo segundo campo e assim sucessivamente
  - Crescente → "asc"
  - Decrescente → "desc"

```
select * from empregado
order by cargo, salario asc
```

codEmp	nome	Salario	cargo
004	João	3000	Analista
001	Anderson	5000	Analista
003	Carlos	7000	Diretor
002	Rui	1500	Programador

empate

Limitando dados

- Limit → limita o número de linhas na resposta
  - Elimina os últimos
- Offset → pula linhas superiores da resposta

```
select * from empregado
limit 2 offset 1
```

codEmp	nome	Salario	cargo
004	João	3000	Analista
001	Anderson	5000	Analista
003	Carlos	7000	Diretor
002	Rui	1500	Programador

Operações com grupos

- União → A∪B

- Intersecção → A∩B

- Menos → A–B

- Menos → B–A

Operações com grupos

- União → A∪B
- Intersecção → A∩B

- Menos → A–B

- Menos → B–A

Operações de conjuntos só podem ser feitas entre relações do mesmo tipo:
Exemplo: rel1(3 colunas) e rel2(3colunas)

Operações com grupos

- Union

```
SELECT nome FROM empregado
UNION
SELECT nome FROM cliente;
```

Union all
X
Union distinct (padrão)

- Intersect

```
SELECT nome FROM empregado
INTERSECT
SELECT nome FROM cliente;
```

- Except (Minus)

```
SELECT nome FROM empregado
EXCEPT
SELECT nome FROM cliente;
```


Agrupando dados

- Funções agregadas
  - Tomam uma coleção de valores e retornam um único valor
    - Média (*average*) → `avg()`
    - Mínimo → `min()`
    - Máximo → `max()`
    - Somatório → `sum()`
    - Contagem (Número de linhas) → `count()`

Agrupando dados

- Funções agregadas
  - Tomam uma coleção de valores e retornam um único valor
    - Média (*average*) → `avg()`
    - Mínimo → `min()`
    - Máximo → `max()`
    - Somatório → `sum()`
    - Contagem (Número de linhas) → `count()`
  - Exemplos:

```
SELECT count(*) FROM empregado
SELECT avg(salario) AS media FROM
empregado
```


Agrupando dados

- Cláusula GROUP BY
  - Utilizado para dividir uma relação em subgrupos
  - Os subgrupos são divididos com base em atributos (colunas)

Agrupando dados

- Cláusula GROUP BY
  - Utilizado para dividir uma relação em subgrupos
  - Os subgrupos são divididos com base em atributos (colunas)
  - As funções de agregação atuam separadamente em cada subgrupo
  - Exemplo: qual a média dos salários de cada cargo

```
SELECT cargo, avg(salario) AS media
FROM empregado GROUP BY cargo
```

Exemplo GROUP BY

- Exemplo: qual a média dos salários de cada cargo?

```
SELECT cargo, avg(salario) AS media
FROM empregado GROUP BY cargo
```

codEmp	Nome	salario	cargo
001	Tiago	5000	Analista
002	João	3000	Analista
003	Rui	1500	Programador
004	Julio	2000	Programador
005	Ana	2500	Programador
006	Maria	2000	Programador
007	Carlos	7000	Diretor

Exemplo GROUP BY

- Exemplo: qual a média dos salários de cada cargo?

```
SELECT cargo, avg(salario) AS media
FROM empregado GROUP BY cargo
```

codEmp	Nome	salario	cargo
001	Tiago	5000	Analista
002	João	3000	Analista
003	Rui	1500	Programador
004	Julio	2000	Programador
005	Ana	2500	Programador
006	Maria	2000	Programador
007	Carlos	7000	Diretor

Cargo	Media
Analista	4000
Programador	2000
Diretor	7000

Agrupando dados

- Cláusula HAVING
  - Define uma condição para a seleção de cada subgrupo
    - Deve ser usado com o **GROUP BY**
  - Funcionamento semelhante ao **WHERE**
    - **WHERE** analisa a condição por **linha**
    - **HAVING** analisa a condição por **subgrupo**

Exemplo GROUP BY c/ HAVING

- Exemplo: qual a média dos salários de cada cargo com mais de um empregado?

```
SELECT cargo, avg(salario) AS media
FROM empregado GROUP BY cargo
     HAVING count(*)>1
```

codEmp	Nome	salario	cargo
001	Tiago	5000	Analista
002	João	3000	Analista
003	Rui	1500	Programador
004	Julio	2000	Programador
005	Ana	2500	Programador
006	Maria	2000	Programador
007	Carlos	7000	Diretor

Exemplo GROUP BY c/ HAVING

- Exemplo: qual a média dos salários de cada cargo com mais de um empregado?

```
SELECT cargo, avg(salario) AS media
FROM empregado GROUP BY cargo
     HAVING count(*)>1
```

codEmp	Nome	salario	cargo
001	Tiago	5000	Analista
002	João	3000	Analista
003	Rui	1500	Programador
004	Julio	2000	Programador
005	Ana	2500	Programador
006	Maria	2000	Programador
007	Carlos	7000	Diretor

Cargo	Media
Analista	4000
Programador	2000

Não satisfaz a condição do HAVING

Selects aninhadas

- O resultado de uma *select* é uma relação
  - Pode ser usada como tal
- Select no **where**
  - Operador **in** e **not in**:

```
select * from Empregado
 where NOME in
(Select Nome from cliente)
```

Selects aninhadas

- O resultado de uma *select* é uma relação
  - Pode ser usada como tal
- Select no **where**
  - Operador **in** e **not in**:

```
select * from Empregado
 where NOME in
(Select Nome from cliente)
```

Resulta em uma lista (grupo) de nomes

Se a resposta à **subselect** fosse uma relacao com duas colunas teríamos problemas

Selects aninhadas

- O resultado de uma *select* é uma relação
  - Pode ser usada como tal
- Select no **where**
  - Operador **in** e **not in**:

```
select * from Empregado
 where NOME in
(Select Nome from cliente)
```

- Select no **from**

```
select S.CARGO, S.QTDE, SM from
(select CARGO,count(CARGO) as QTDE, avg(SALARIO) as SM
from empregado group by CARGO) S
where avg(SALARIO) > 2000
```


Selects aninhadas

- O resultado de uma *select* é uma relação
  - Pode ser usada como tal
- Select no **where**
  - Operador **in** e **not in**:

```
select * from Empregado
 where NOME in
(Select Nome from cliente)
```

- Select no **from**

```
select S.CARGO, S.QTDE, SM from
(select CARGO,count(CARGO) as QTDE, avg(SALARIO) as SM
from empregado group by CARGO) S
where avg(SALARIO) > 2000
```

Trata-se a resposta da *subselect* como uma **tabela(relação)** de nome **S**

Selects aninhadas

- O comando ***With***
  - Guarda a resposta de uma consulta (tabela) em uma variável
  - É como se fosse criada uma View temporária
    - fica em memória apenas durante a execução da consulta
  - Ajuda na clareza de consultas complexas **(dividir para conquistar!)**
- Exemplo

```
with consulta as
(SELECT cargo, count(cargo) as qtde, avg(salario) as sm
FROM empregado GROUP BY cargo)

SELECT * FROM consulta
WHERE sm>2000
```


Junções

- É possível executar junções entre tabelas. **(ou o produto cartesiano)**

```
relacao1 JOIN relacao2
```

- As tabelas devem ter um campo para serem vinculadas.
- Útil para vincular tabelas que se relacionam por chave estrangeira
- Tipos de junções
  - Interna
  - Externa
  - Externa à esquerda
  - Externa à direita
  - Natural

Junções – Produto Cartesiano

- Produto cartesiano **Empregado X Depto**

```
SELECT * FROM empregado, depto
```

CodEmp	Nome	CodDepto
004	João	null
002	Anderson	040

X

CodDepto	Nome
040	TI
001	Direcao

CodEmp	Nome	CodDepto	CodDepto	Nome
004	João	null	040	TI
004	João	null	001	Direcao
002	Anderson	040	040	TI
002	Anderson	040	001	Direcao

Junções – Produto Cartesiano

- Produto cartesiano **Empregado X Depto**

```
SELECT * FROM empregado, depto
```

CodEmp	Nome	CodDepto
004	João	null
002	Anderson	040

CodDepto	Nome
040	TI
001	Direcao

CodEmp	Nome	CodDepto	CodDepto	Nome
004	João	null	040	TI
004	João	null	001	Direcao

Junções – Produto Cartesiano

- Produto cartesiano **Empregado X Depto**

```
SELECT * FROM empregado, depto
```

CodEmp	Nome	CodDepto
004	João	null
002	Anderson	040

CodDepto	Nome
040	TI
001	Direcao

CodEmp	Nome	CodDepto	CodDepto	Nome
004	João	null	040	TI
004	João	null	001	Direcao
002	Anderson	040	040	TI
002	Anderson	040	001	Direcao

Junções – Produto Cartesiano

- Produto cartesiano **Empregado X Depto**

```
SELECT * FROM empregado, depto
```

CodEmp	Nome	CodDepto
004	João	null
002	Anderson	040

X

CodDepto	Nome
040	TI
001	Direcao

CodEmp	Nome	CodDepto	CodDepto	Nome	
004	João	null	040	TI	X
004	João	null	001	Direcao	X
002	Anderson	040	040	TI	✓
002	Anderson	040	001	Direcao	X

Correspondência

Junções

- Condições de vinculação entre tabelas
  - On

```
ON empregado.CODDEPTO = depto.CODDEPTO
```

  - Using

```
USING(CODDEPTO)
```

- Exemplo Sintaxe

```
SELECT ... FROM
Empregado <tipo> JOIN depto USING (CODDPTO)
```


Junções

- Condições de vinculação entre tabelas
  - On

```
on empregado.CODDEPTO = depto.CODDEPTO
```

  - Using

```
using(CODDEPTO)
```

- OU

```
on empregado.CODDEPTO =
depto.CODDEPTO
```

- Exemplo Sintaxe

```
Select ... From
Empregado <tipo> JOIN depto USING (CODDPTO)
```

- [INNER]
- FULL [OUTER]
- LEFT [OUTER]
- RIGHT [OUTER]
- NATURAL

Junções – interna

```
SELECT * FROM empregado INNER JOIN depto
USING(CODDEPTO)
```

CodEmp	Nome	CodDepto
004	João	null
002	Anderson	040

X

CodDepto	Nome
040	TI
001	Direcao

CodEmp	Nome	CodDepto	Nome
002	Anderson	040	TI

Junções – interna

```
SELECT * FROM empregado INNER JOIN depto
ON empregado.CODDEPTO = depto.CODDEPTO
```

CodEmp	Nome	CodDepto
004	João	null
002	Anderson	040

X

CodDepto	Nome
040	TI
001	Direcao

CodEmp	Nome	CodDepto	CodDepto	Nome
002	Anderson	040	040	TI

Junções – interna natural

```
SELECT * FROM empregado NATURAL INNER JOIN depto
```

CodEmp	NomeEmp	CodDepto
004	João	null
002	Anderson	040

X

CodDepto	NomeDepto
040	TI
001	Direcao

CodEmp	NomeEmp	CodDepto	CodDepto	NomeDepto
002	Anderson	040	040	TI

Junções – esquerda externa

```
SELECT * FROM empregado LEFT OUTER JOIN depto
ON empregado.CODDEPTO = depto.CODDEPTO
```

CodEmp	Nome	CodDepto
004	João	null
002	Anderson	040

X

CodDepto	Nome
040	TI
001	Direcao

CodEmp	Nome	CodDepto	CodDepto	Nome
004	João	null	-	-
002	Anderson	040	040	TI

Junções – direita externa

```
SELECT * FROM empregado RIGHT OUTER JOIN depto
ON empregado.CODDEPTO = depto.CODDEPTO
```

CodEmp	Nome	CodDepto
004	João	null
002	Anderson	040

X

CodDepto	Nome
040	TI
001	Direcao

CodEmp	Nome	CodDepto	CodDepto	Nome
002	Anderson	040	040	TI
_	-	-	001	Direcao

Junções – externa total

```
SELECT * FROM empregado FULL OUTER JOIN depto
ON empregado.CODDEPTO = depto.CODDEPTO
```

CodEmp	Nome	CodDepto
004	João	null
002	Anderson	040

X

CodDepto	Nome
040	TI
001	Direcao

CodEmp	Nome	CodDepto	CodDepto	Nome
004	João	null	-	-
002	Anderson	040	040	TI
-	-	-	001	Direcao