



Laboratório de Bases de Dados

Prof. José Fernando Rodrigues Júnior

Aula 3 - Revisão SQL/DML

Material original editado: Profa. Elaine Parros Machado de Sousa



DML - Introdução

- **Comandos da DML:**

- INSERT

- UPDATE

- DELETE



- SELECT



Comandos DML

- **SELECT** – comando de consulta
 - retorno \Rightarrow tabela resultado (**multiconjunto – potencialmente um conjunto com repetições**)

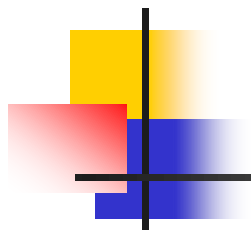
```
SELECT [DISTINCT|ALL] <lista de atributos>
FROM <lista de tabelas>
[WHERE <condições>]
[GROUP BY atributo]
[HAVING <condições>]
[ORDER BY atributo [ASC|DESC]]
```



SELECT

- **SELECT** → **O QUE** se deseja na tabela resultado
 - *<lista de atributos>* ou
 - * (para todos os atributos)
 - ALL – resultado pode conter tuplas duplicadas (*default*)
 - DISTINCT – resultado contém somente tuplas distintas
- **FROM** → **DE ONDE** retirar os dados necessários
- **WHERE** → **CONDIÇÕES** (predicado) da consulta
 - expressão condicional booleana
 - condições de seleção
 - condições de junção, ...

Exemplo:



Aluno = {Nome, Nusp, Idade, DataNasc, CidadeOrigem }

Professor = {Nome, NFunc, Idade, Titulação}

Disciplina = {Sigla, Nome, NCred, Professor, Livro}

Turma = {Sigla, Numero, NAlunos}

Matrícula = {Sigla, Numero, Aluno, Ano, Nota, Freq}

- Selecionar os alunos (NUSP) que cursam a disciplina SCC541 ou a SCC 241;

```
SELECT Aluno FROM Matricula
WHERE Sigla IN ('SCC541', 'SCC241');
```

- Selecionar os alunos (NUSP) que cursam alguma disciplina do SCC no ano de 2010;

```
SELECT Distinct Aluno FROM Matricula
WHERE Sigla LIKE 'SCC%' and Ano = 2010;
```

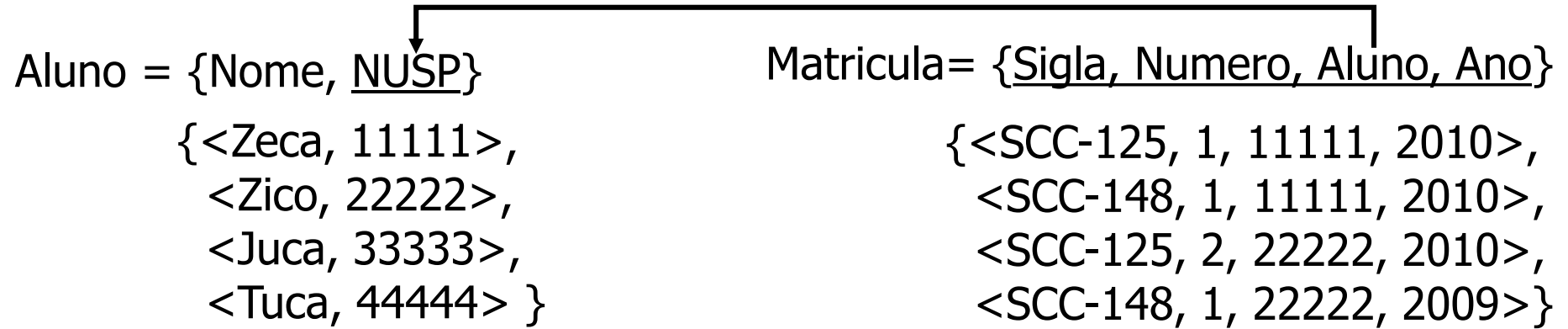


SELECT

- Cláusula **FROM** com mais de uma tabela
 - **Junção interna** (*Inner Join*)
 - **WHERE** \Rightarrow condição de junção
 - em geral: atributos com relacionamento PK - FK

```
SELECT [DISTINCT|ALL] <atributos>
FROM tabela1, tabela2
WHERE tabela1.atributo1 =
      tabela2.atributo2
```

Exemplo: Junção



```
select A.nome, A.nusp, M.Sigla  
from Aluno A, Matricula M  
where A.nusp = M.aluno
```

{Nome, NUSP, Sigla}
{<Zeca, 11111, SCC-125>, <Zeca, 11111, SCC-148>, <Zico, 22222, SCC-125>, <Zico, 22222, SCC-148 >}

Junção Interna – operador

JOIN

```
SELECT [DISTINCT|ALL] <atributos>
      FROM tabela1 T1
      [INNER] JOIN tabela2 T2
      ON T1.atributo1 = T2.atributo2
```




Junção Interna

```
SELECT <atributos>  
    FROM tabela1 T1 , tabela2 T2  
    WHERE T1.atributo1 = T2.atributo2
```



```
SELECT <atributos>  
    FROM tabela1 T1 JOIN tabela2 T2  
    ON T1.atributo1 = T2.atributo2
```



Junções Externas

```
SELECT [DISTINCT|ALL] <atributos>  
      FROM tabela1 T1  
      [LEFT | RIGHT | FULL] JOIN tabela2 T2  
      ON T1.atributo1 = T2.atributo2
```

```
SELECT [DISTINCT|ALL] <atributos>  
      FROM tabela1 T1, tabela2 T2  
      WHERE T1.atributo1[(+)] = T2.atributo2[(+)]
```



Junções Externas

- ***LEFT JOIN COM (+)***

```
SELECT [DISTINCT|ALL] <atributos>  
      FROM tabela1 T1, tabela2 T2  
      WHERE T1.atributo1 = T2.atributo2 (+)
```

- ***RIGHT JOIN COM (+)***

```
SELECT [DISTINCT|ALL] <atributos>  
      FROM tabela1 T1, tabela2 T2  
      WHERE T1.atributo1 (+) = T2.atributo2
```



Junções Externas

- **LEFT**

SELECT

FROM

WHERE

- **RIGHT**

SELECT

FROM

O (+) não é interpretado de acordo com qual lado ele está (left ou right); ele apenas indica de qual tabela serão aceitos valores null.

(+)

WHERE *T1.atributo1 (+) = T2.atributo2*

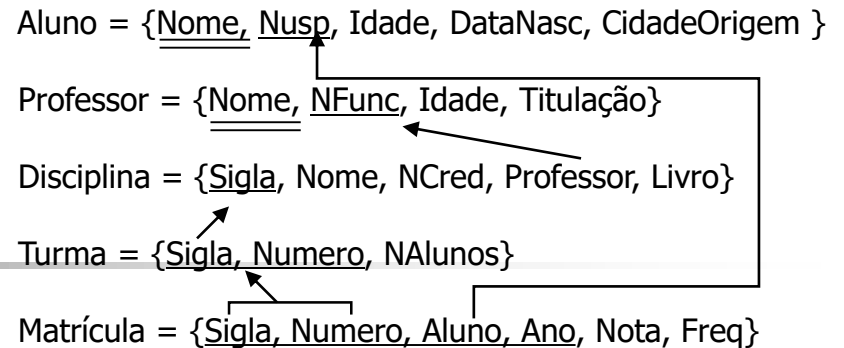
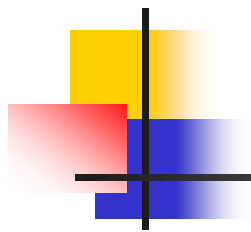
Exemplo: Junção Externa

Aluno = {Nome, <u>NUSP</u> }	Matricula= { <u>Sigla</u> , <u>Numero</u> , <u>Aluno</u> , <u>Ano</u> }
{<Zeca, 11111>, <Zico, 22222>, <Juca, 33333>, <Tuca, 44444> }	{<SCC-125, 1, 11111, 2010>, <SCC-148, 1, 11111, 2010>, <SCC-125, 2, 22222, 2010>, <SCC-148, 1, 22222, 2009>}

```
select A.nome, A.nusp, M.Sigla  
from Aluno A left join Matricula M  
where A.nusp = M.aluno
```

```
{Nome, NUSP, Sigla}  
{<Zeca, 11111, SCC-125>,  
<Zeca, 11111, SCC-148>,  
<Zico, 22222, SCC-125>,  
<Zico, 22222, SCC-148>,  
<Juca, 33333, NULL >,  
<Tuca, 44444, NULL>}
```

Exemplo:



- Selecionar nome e nro funcional dos professores DOUTORES que ministram ou não ministram disciplinas.

```
select P.Nome, P.NFunc, D.Sigla
  from Disciplina D right join Professor P
                    on D.Professor = P.NFunc
 where UPPER(P.Titulacao) = UPPER('doutor')
```



SELECT

- Funções Agregadas

- entrada \Rightarrow conjunto de valores

- saída \Rightarrow valor

- Exemplos:

- `AVG(atributo)` \rightarrow calcula a média da coluna *atributo*

- `COUNT()`

- `count(*)` – retorna o número de tuplas de uma consulta

- `count(atributo)` – retorna o nro de valores (com repetição) da coluna *atributo* que não tem valores null



SELECT

- Funções Agregadas

- Exemplos

- `MAX(atributo)` → recupera o valor máximo da coluna *atributo*
 - `MIN(atributo)` → recupera o valor mínimo da coluna *atributo*
 - `SUM(atributo)` → obtém a soma de valores da coluna *atributo*
 - `AVG (média)`, `STDDEV (desvio padrão)`, e `VARIANCE (variância)`

http://docs.oracle.com/cd/E11882_01/server.112/e10592/functions003.htm



SELECT

Funções Agregadas

- Exemplo: **GROUP BY**, ou agrupamento, assume a presença de valores repetidos → portanto, apesar de aceito, não faz sentido a realização de agrupamentos sobre os atributos chave
- *SUM(atributo)* → obtém a soma de valores da coluna *atributo*
- *AVG (média)*, *STDDEV (desvio padrão)*, e *VARIANCE (variância)*

http://docs.oracle.com/cd/E11882_01/server.112/e10592/functions003.htm



SELECT

- **GROUP BY** → agrupamento de tuplas
 - para a aplicação de funções agregadas
- **HAVING** → condições aplicadas **a grupos já formados** por **GROUP BY**
- **ORDER BY** → estabelece a ordenação lógica da tabela de resultados
 - **ASC** (*default*)
 - **DESC**

Exemplo:

Aluno = {Nome, NUSP}

{<Zeca, 11111>,
<Zico, 22222>,
<Juca, 33333>,
<Tuca, 44444> }

Matricula= {Sigla, Numero, Aluno, Ano, Nota}

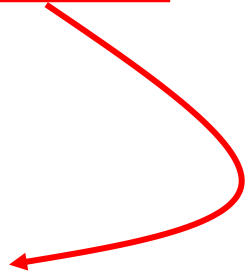
{<SCC-125, 1, 11111, 2010, 5.0>,
<SCC-148, 1, 11111, 2010, 7.0>,
<SCC-125, 2, 22222, 2010, 5.0>,
<SCC-148, 1, 22222, 2009, 4.0>}

- Selecionar, para cada aluno, seu nome e a média das notas das disciplinas em que foi aprovado (nota ≥ 5). Ordenar por nome de aluno

1º Passo: seleção e junção

```
SELECT ...  
  FROM Aluno A JOIN Matricula M  
           ON M.Aluno = A.NUSP  
 WHERE M.Nota BETWEEN 5.0 AND 10.0
```

{Nome, NUSP, Sigla, Nota}
{<Zeca, 11111, SCC-125, 5.0>,
<Zeca, 11111, SCC-148, 7.0>,
<Zico, 22222, SCC-125, 5.0>}



Exemplo: (continuação)

2º Passo: agrupamento e agregação

```
SELECT A.Nome, AVG(M.Nota) as Media
  FROM Aluno A JOIN Matricula M
             ON M.Aluno = A.NUSP
 WHERE M.Nota BETWEEN 5.0 AND 10.0
 GROUP BY A.Nome
 ORDER BY A.Nome;
```

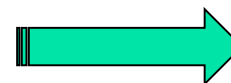
Grupo Zeca

<SCC125, 5.0>
<SCC148, 7.0>

Grupo Zico

<SCC125, 5.0>

Função **AVG** aplicada sobre cada grupo



{Nome, Media}
{<Zeca, 6.0>,
 <Zico, 5.0>}

Exemplo:

Aluno = {Nome, NUSP}

{<Zeca, 11111>,
<Zico, 22222>,
<Juca, 33333>,
<Tuca, 44444> }

Matricula= {Sigla, Numero, Aluno, Ano, Nota}

{<SCC-541, 1, 11111, 2009, 3.0>,
<SCC-541, 1, 11111, 2010, 7.0>,
<SCC-240, 1, 11111, 2010, 5.0>,
<SCC-240, 1, 22222, 2009, 4.0>}

Disciplina = {Sigla, Nome}

{<SCC-541, LabBD>,
<SCC-240, BD>}

- Selecionar os nomes dos alunos que fizeram uma mesma disciplina mais de uma vez. Listar também o nome da disciplina, o nro de vezes que cursou e a nota máxima que o aluno obteve (considerando todas as vezes que cursou).

1º Passo: junção

```
select ....  
  from Aluno A join Matricula M  
           on A.NUSP = M.Aluno  
  join Disciplina D  
  on D.Sigla = M.Sigla
```

Exemplo: (continuação)

2º Passo: agrupamento e agregação

```
select A.Nome, D.Nome, count(*), max(M.Nota)
  from Aluno A join Matricula M
              on A.NUSP = M.Aluno
  join Disciplina D
  on D.Sigla = M.Sigla
 group by A.Nome, D.Nome
```

Grupo Zeca

sub-grupo LabBD

<LabBD, 3.0>

<LabBD, 7.0>

sub-grupo BD

<BD, 5.0>

Grupo Zico

sub-grupo BD

<BD, 5.0>

Funções **COUNT** e **MAX** aplicadas sobre cada sub-grupo

Exemplo: (continuação)

3º Passo: condição having

```
select A.Nome, D.Nome, count(*), max(M.Nota)
  from Aluno A join Matricula M
           on A.NUSP = M.Aluno
      join Disciplina D
           on D.Sigla = M.Sigla
 group by A.Nome, D.Nome
 having count(*) > 1
```

Grupo Zeca

sub-grupo LabBD

<LabBD, 3.0>

<LabBD, 7.0>

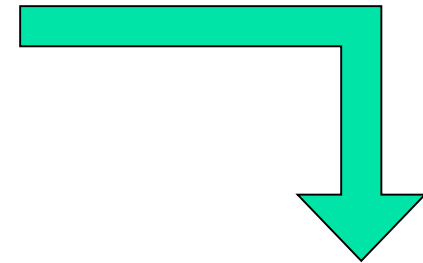
~~sub-grupo BD~~

~~<BD, 5.0>~~

Grupo Zico

~~sub-grupo BD~~

~~<BD, 5.0>~~



{A.Nome, D.Nome, count, max}
{< Zeca, LabBD, 2, 7.0>}



Consultas Aninhadas (Nested Queries)

- **Não correlacionadas** – independentes
 - ex: selecionar nome e nusp dos alunos com a idade mais alta

```
select nome, nusp from aluno
where idade IN
      (select max(idade)
       from aluno)
```




Consultas Aninhadas (Nested Queries)

- **Não correlacionadas** – independentes

- ex: Consultas IN funcionam com a idade trazendo dados de “fora” para “dentro” da consulta principal.

```
select nome, nusp from aluno
where idade IN
    (select max(idade)
     from aluno)
```



Consultas Aninhadas

- **Correlacionadas** – condição na cláusula WHERE da consulta interna referencia algum atributo de tabela da consulta externa

EXEMPLO:

Aluno = {Nome, Nusp, Idade, DataNasc}

Disciplina = {Sigla, Nome, NCred, Professor, Livro, Monitor}

Matrícula = {Sigla, Numero, Aluno, Ano, Nota}

- Selecionar nome e nusp dos alunos que estão matriculados em alguma disciplina e que são monitores de alguma disciplina

```
select nome, nusp from aluno A where
```

```
    EXISTS (select NULL from matricula M
            where M.aluno = A.nusp)
```

and

```
    EXISTS (select NULL from disciplina D
            where D.monitor = A.nusp )
```

EXEMPLO:

Aluno = {Nome, Nusp, Idade, DataNasc}

Disciplina = {Sigla, Nome, NCred, Professor, Livro, Monitor}

Matrícula = {Sigla, Numero, Aluno, Ano, Nota}

- Selecionar n
disciplina e q

select n

Consultas EXISTS funcionam levando dados de “dentro” para “fora” da consulta principal.

A cláusula EXISTS não retorna dados, mas sim um status booleano.

EXISTS

and

EXISTS (**select** NULL from disciplina D
where D.monitor = **A.nusp**)

```
select nome, nusp from aluno A where  
exists (select NULL from matricula M  
where M.aluno = A.nusp)
```

and

```
exists (select NULL from disciplina D  
where D.Monitor = A.nusp )
```



Também pode ser resolvida com junção

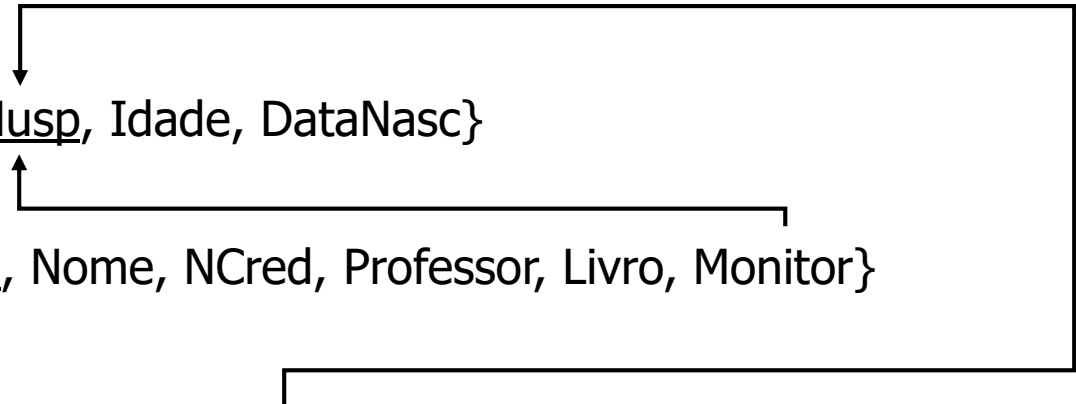
```
select distinct A.nome, A.nusp  
from aluno A, matricula M, disciplina D  
where M.aluno = A.nusp and D.monitor = A.nusp
```

EXEMPLO:

Aluno = {Nome, Nusp, Idade, DataNasc}

Disciplina = {Sigla, Nome, NCred, Professor, Livro, Monitor}

Matrícula = {Sigla, Numero, Aluno, Ano, Nota}



- Selecionar nome e nusp dos alunos que não estão matriculados em nenhuma disciplina

```
select nome, nusp from aluno A where  
NOT EXISTS
```

```
(select NULL from matricula M  
where M.aluno = A.nusp)
```

EXEMPLO:

```
select nome, nusp
from aluno A where
    NOT EXISTS
        (select NULL from matricula M
         where M.aluno = A.nusp)
```

```
select nome, nusp
from aluno A LEFT JOIN matricula M
    on M.aluno = A.nusp
where M.disciplina IS NULL
```



Outros recursos SQL do Oracle



SELECT com CASE

- Usado para gerar novos dados categóricos

```
SELECT TIPO, CATEGORIA,  
       CASE TIPO  
         WHEN 'G' THEN CATEGORIA || ' Graduando '  
         WHEN 'P' THEN CATEGORIA || ' PosGraduando '  
         ELSE 'Indefinido '  
       END  
FROM D01
```



RANK

- **Gera informação de ranking de acordo com a ordem de qualquer atributo**

```
SELECT Nome, NUsp, AnoIngresso,  
        RANK() OVER (ORDER BY AnoIngresso) MAIS_VELHO  
FROM D01
```



LISTAGEM ALEATÓRIA

```
SELECT *  
FROM (  
    SELECT *  
    FROM table  
    ORDER BY DBMS_RANDOM.RANDOM)  
WHERE rownum < 21;
```



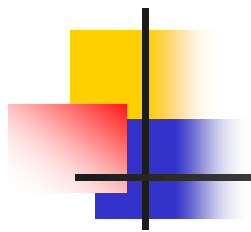
Para testar:

```
select to_char(datanascimento, 'dd-mm-yyyy  
hh24:mi')  
      from ALUNO;
```

```
select sysdate from dual;
```

```
select to_char(sysdate, 'dd-mm-yyyy hh12:mi')  
      from dual;
```

```
select seq_dnro.currval from dual;
```



Lógica de três valores



NULL – Lógica de três valores

- Clausulas WHERE verificam o status de cada tupla com relacao a um predicado – os possiveis status são:
 - true
 - false
 - null: informação indisponível, não se pode afirmar nada
- Cláusulas WHERE retornam apenas tuplas cujo status definido pelo predicado seja true – recusam-se tuplas com status false e NULL (unknown).



NULL – Lógica de três valores

- **IS [NOT] NULL:** retorna true se um dado valor tiver estado NULL;

Na tupla\No predicado	= NULL	= value	IS NULL
valor	NULL ("false")	false/true	false
NULL	NULL ("false")	NULL ("false")	true

Exemplo:

SELECT IDADE
FROM ALUNO
WHERE IDADE = NULL

NULL ("false") para todas
as tuplas

SELECT IDADE
FROM ALUNO
WHERE IDADE = value

Retorna NULL ("false"),
para as tuplas com valor
NULL, true ou false para
as demais

SELECT IDADE
FROM ALUNO
WHERE IDADE IS NULL

Se na tupla houver NULL
retorna TRUE, para as demais
retorna FALSE

* "false": valor NULL que não é retornado pelo SELECT, não se trata de um false de fato



Onde consultar ...

- R. Elmasri, S. Navathe: *Fundamentals of Database Systems* – 4th Edition
 - Capítulo 8
- A. Silberschatz, H. F. Korth, s. Sudarshan: *Sistema de Banco de Dados*
 - Capítulo 4
- Manuais em *list of books* no site da Oracle
 - ***SQL Reference***



Onde consultar ...

- R. Elmasri, S. Navathe: *Fundamentals of Database Systems* – 4th Edition
- A. Silberschatz, H. F. Korth, s. Sudarshan: *Sistema de Banco de Dados*
- Manuais em *list of books* no site da Oracle
 - ***SQL Reference***



PRÁTICA 3