

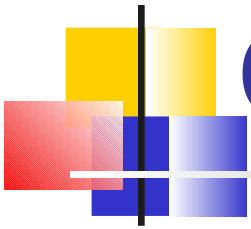


Laboratório de Bases de Dados

Prof. José Fernando Rodrigues Júnior

Aula 3 – Gerenciamento de Usuários Revisão SQL/DML (Parte 2)

Material: Profa. Elaine Parros Machado de Sousa



Conteúdo

- Gerenciamento de Usuários no Oracle
 - usuários
 - papéis (atribuições)
 - privilégios
- Revisão SQL/DML

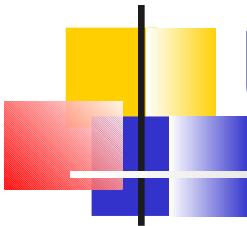


Usuários

- Criação:

```
CREATE USER nome_usuario
  IDENTIFIED {BY senha | EXTERNALLY}
  DEFAULT TABLESPACE nome_tablespace
  TEMPORARY TABLESPACE nome_tablespace
  QUOTA {integer [K|M] | UNLIMITED} ON
                                     nome_tablespace

  PROFILE nome_profile
  PASSWORD EXPIRE
  ACCOUNT {LOCK | UNLOCK}
```



Usuários

- Exemplo:

```
CREATE USER labbd  
  IDENTIFIED BY labbd  
  DEFAULT TABLESPACE users  
  TEMPORARY TABLESPACE users  
  QUOTA 10M ON users  
  ACCOUNT UNLOCK
```



Usuários

- Exclusão:

```
DROP USER nome_usuario [CASCADE]
```



Privilégios

- **Privilégio** ⇒ autorização para que o usuário acesse e manipule objetos do BD.
- Tipos:
 - Sistema: permissão de executar ações no BD
 - + de 100 tipos de privilégios distintos
 - ex: `create table, create session, create tablespace, create user, alter user, ...`
 - Objeto: permissão para acessar e manipular um objeto específico
 - em torno de 11 possíveis privilégios
 - ex: `select on Aluno, insert on Aluno, update (nusp) on Aluno ...`



Privilégios

- **Privilégios de sistema** – criação, alteração e remoção de objetos de esquemas
 - esquema do usuário
 - ex: `create table, create view, create procedure, ...`
 - usuário é **dono** dos objetos que cria ⇒ possui permissão de alterar, remover, consultar, inserir, atualizar, executar, conceder privilégios sobre seus objetos, etc...
 - qualquer esquema
 - ex: `create any table, alter any table, drop any table, select any table, insert any table, execute any procedure ...`

Papéis (atribuições)

- **Papel:** grupo de privilégios

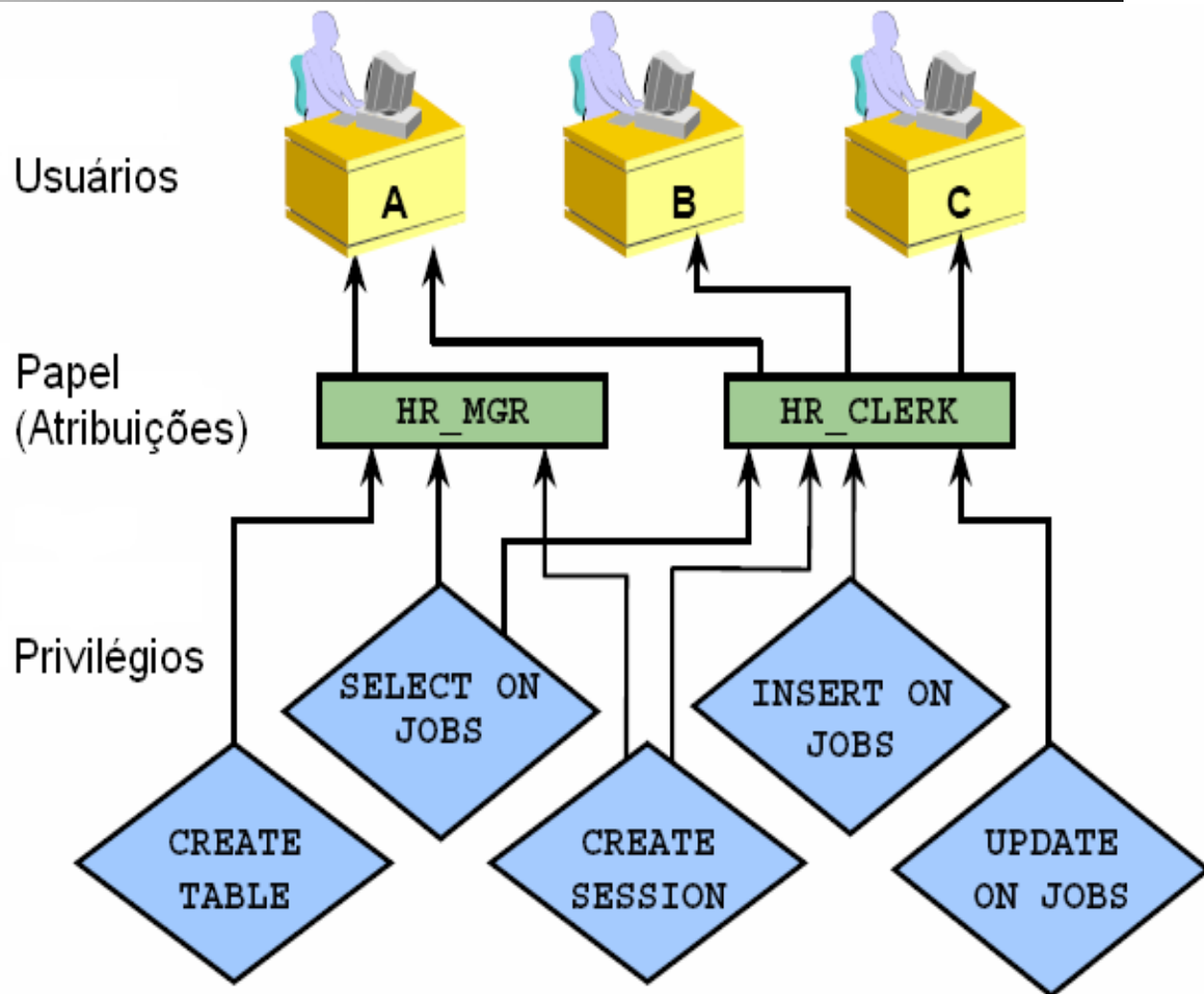
- simplifica a administração dos usuários

- Criar papel:

`CREATE ROLE papel`

- Excluir papel:

`DROP ROLE papel`





Privilégios, Papéis e Usuários

- Concessão de privilégio e/ou papel:

- **de sistema**

```
GRANT privilegio [,privilegio,...] | papel
```

```
TO usuario [,usuario,...] | papel | PUBLIC
```

```
[WITH ADMIN OPTION]
```

- **de objeto**

```
GRANT privilegio [,privilegio,...] | papel
```

```
ON objeto
```

```
TO usuario [,usuario,...] | papel | PUBLIC
```

```
[WITH GRANT OPTION]
```

- alterar (ALTER)
- excluir (DELETE)
- indexar (INDEX)
- inserir (INSERT)
- referenciar (REFERENCES)
- selecionar (SELECT)
- alterar (UPDATE)
- atualizar (ON COMMIT REFRESH)
- ler (READ)
- executar (EXECUTE) procedures, functions, packages, ...
- escrever (WRITE) os objetos (tabela, visão, seqüência, rotina, visão materializada e diretório).

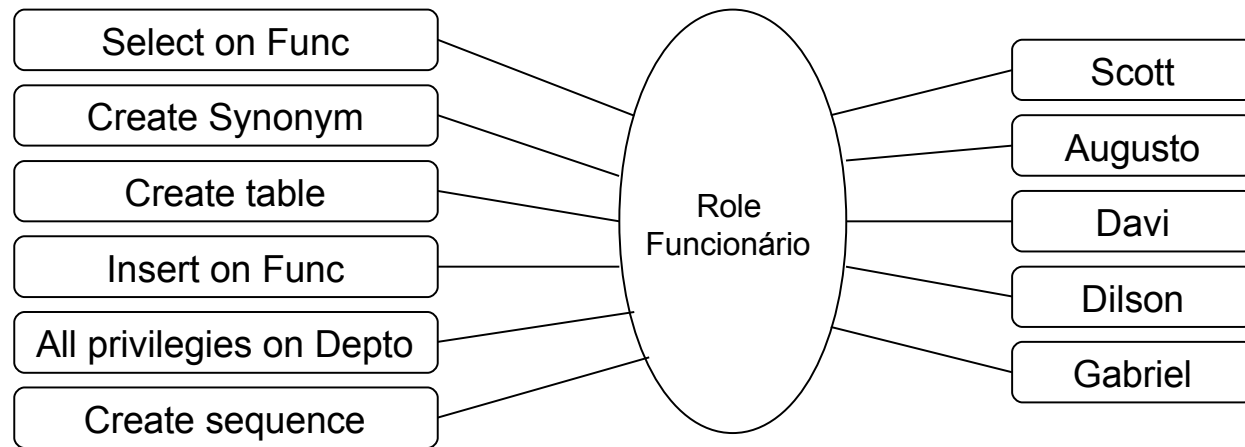


Privilégios, Papéis e Usuários

- Revogar privilégio e/ou papel:

```
REVOKE privilegio [,privilegio,...] | papel  
      [ON objeto]  
FROM usuario [,usuario,...] | papel | PUBLIC
```

Exemplo



- Criar papel:

```
CREATE ROLE FUNCIONARIO;
```

- Atribuir alguns privilégios ao papel:

```
GRANT CREATE TABLE, CREATE SEQUENCE TO FUNCIONARIO;
```

```
GRANT SELECT, INSERT ON TABELA_FUNC TO FUNCIONARIO;
```

- Atribuir o papel ao usuário Scott:

```
GRANT FUNCIONARIO TO SCOTT.
```

--criação da role

create role usuariolabbd;

--permissão dos privilégios

grant create session, create table, create view,
create materialized view, create sequence, create
synonym, create procedure, create trigger, create
type to usuariolabbd;

-- create user ALUNO identified by ALUNO

default tablespace USERS;

grant unlimited tablespace to ALUNO;

grant usuariolabbd to ALUNO;



Privilégios, Papéis e Usuários

■ WITH ADMIN OPTION

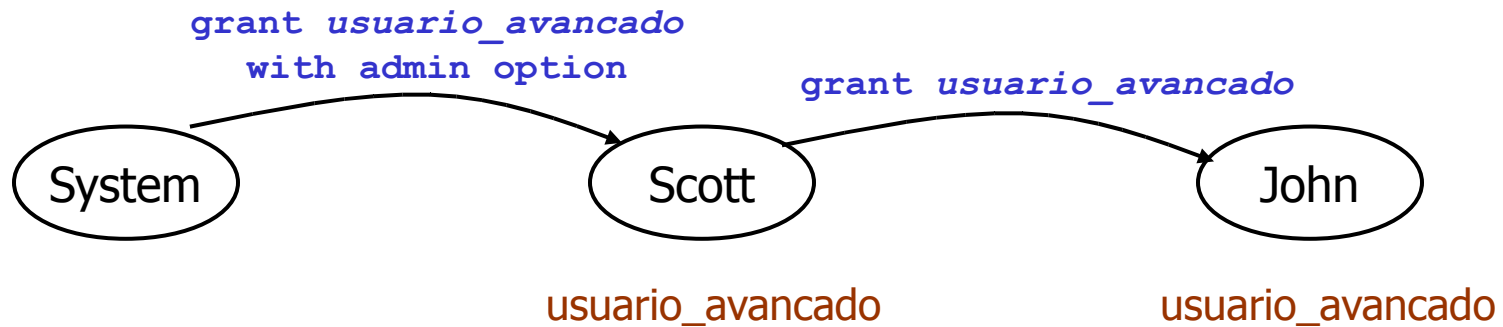
- opção para **privilégios de sistema**
- para usuários ou papéis
- permite ao usuário
 - conceder/revogar o privilégio/papel para/de qualquer usuário ou papel
 - alterar ou remover o papel concedido
- ex:

```
-- conectado como usuário SYSTEM
```

```
create role usuario_avancado;  
grant create table to usuario_avancado;  
grant usuario_avancado to Scott WITH ADMIN OPTION;
```

```
-- conectado como Scott
```

```
grant usuario_avancado to John;
```



-- conectado como usuário SYSTEM
revoke usuario_avancado from Scott;





Privilégios, Papéis e Usuários

■ WITH GRANT OPTION

- opção para **privilégios de objetos**
- somente para usuários
- permite ao usuário
 - conceder o privilégio/papel para qualquer usuário ou papel
 - a opção WITH GRANT OPTION não é propagada automaticamente, ela deve ser especificada

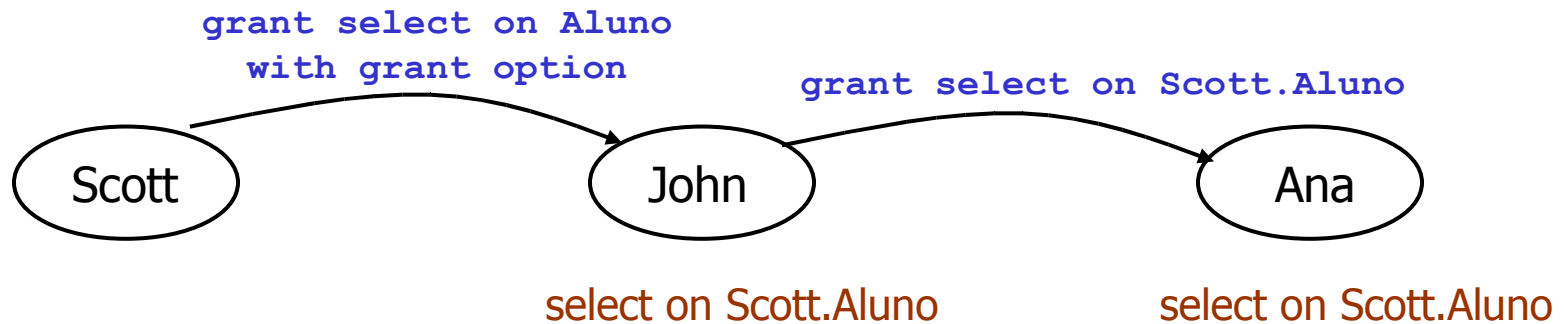
■ ex:

-- conectado como *Scott* (dono da tabela *Aluno*)

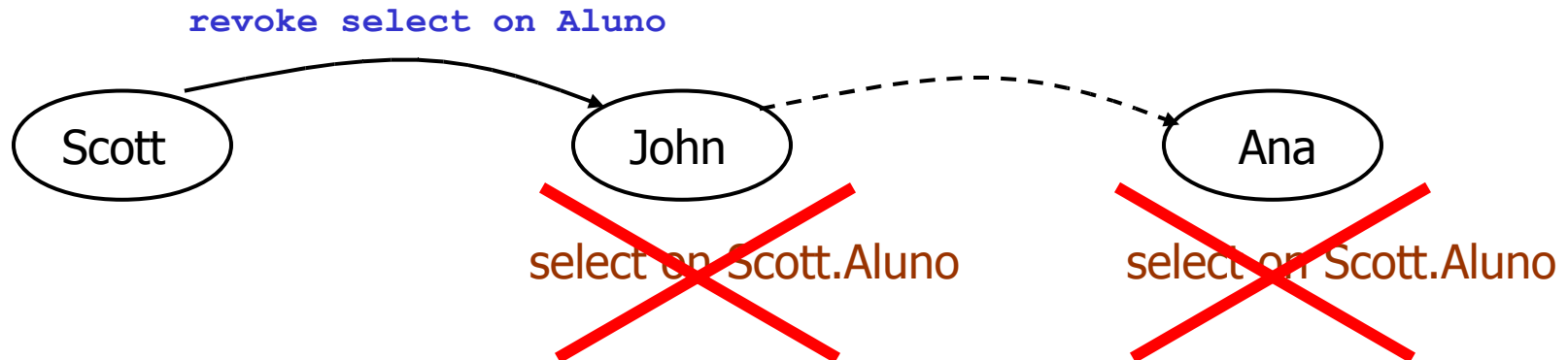
```
grant select on Aluno to John WITH GRANT OPTION;
```

-- conectado como *John*

```
grant select on Scott.Aluno to Ana;
```

-- conectado como usuário Scott
revoke select on Aluno from *John*;





Privilégios, Papéis e Usuários

- Quando grant e revoke têm efeito?
 - privilégios (de sistema e de objetos) para usuários e papéis
 - efeito imediato para sessões correntes e sessões posteriores
 - papéis para usuários e papéis
 - sessões posteriores ⇒ efeito imediato
 - sessões correntes ⇒ comando **SET ROLE role**, para reabilitar o papel



Conteúdo

- Gerenciamento de Usuários no Oracle
 - usuários
 - papéis (atribuições)
 - privilégios
- **Revisão SQL/DML**



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**)

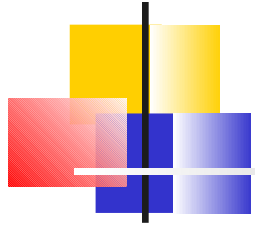
```
SELECT [DISTINCT|ALL] <lista de atributos>
FROM <lista de tabelas>
[WHERE <condições>]
[GROUP BY atributo]
[HAVING <condições>]
[ORDER BY atributo [ASC|DESC]]
```



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;
```

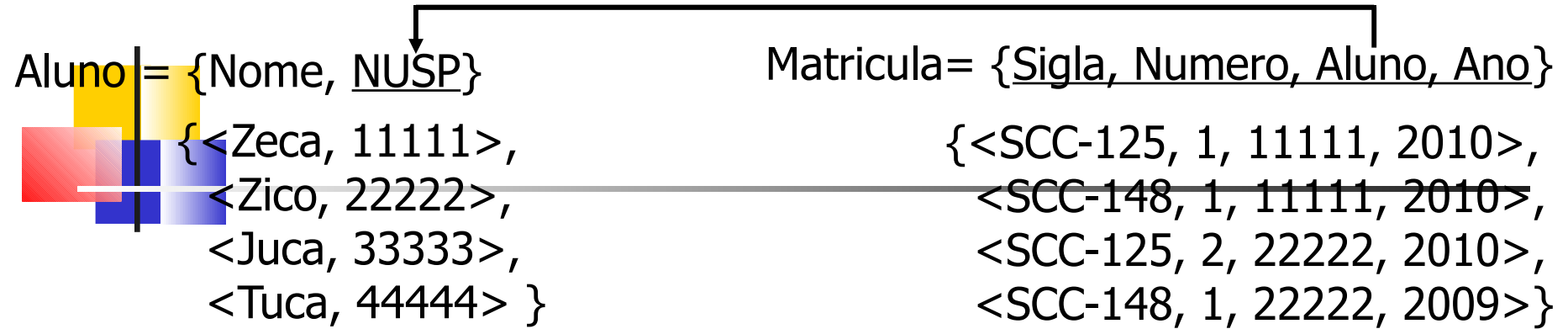


SELECT

- 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
```



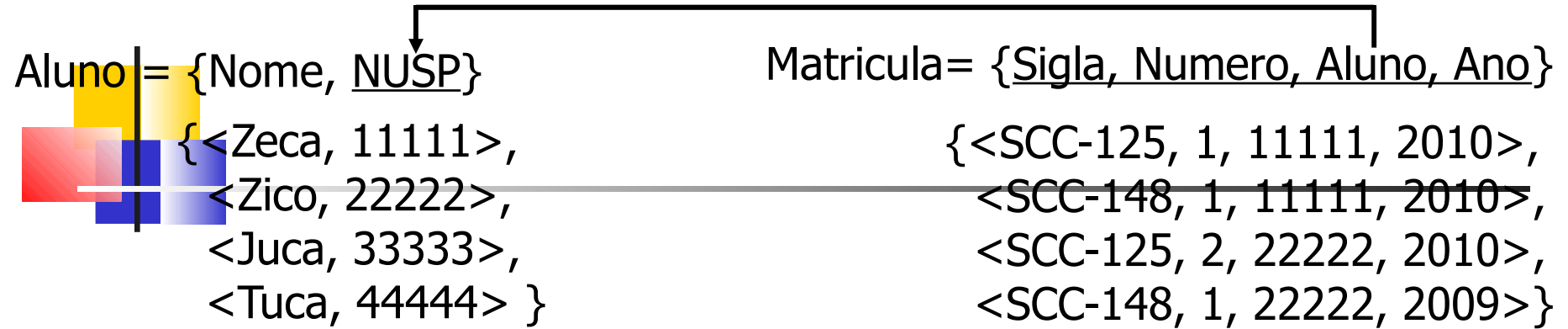
```
SELECT <atributos>  
    FROM tabela1 T1 JOIN tabela2 T2  
    ON T1.atributo1 = T2.atributo2
```



Junções Externas

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

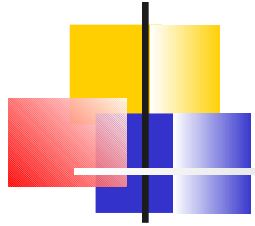
Exemplo: Junção Externa



```
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:



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}

Diagram illustrating relationships between tables:
- An arrow points from NFunc in Professor to Nusp in Aluno.
- An arrow points from Professor to Disciplina.
- An arrow points from Disciplina to Turma.
- An arrow points from Turma to Matrícula.
- A line connects the Sigla attribute in Turma to the Sigla attribute in Matrícula.

- Selecionar nome e nro funcional dos professores doutores e, para aqueles que ministram alguma disciplina, a sigla da disciplina.

```
select  P.Nome, P.NFunc, D.Sigla
      from Professor P left join Disciplina D
                        on P.NFunc = D.Professor
     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*



SELECT

- Função **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
 - `AVG(atributo)` – 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*



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*

- ...



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}

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

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

{<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;
```

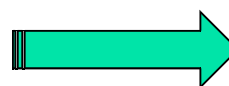
Grupo Zeca

<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>

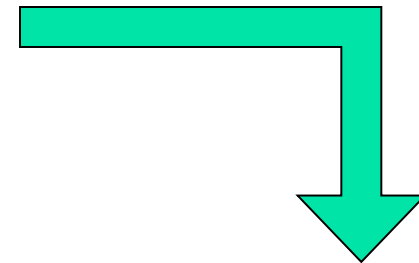
~~sub-grupo BD~~

~~<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: selecionar nome e curso dos alunos com a idade mais alta

Consultas IN funcionam trazendo dados de “fora” para “dentro” da consulta principal.

`select`

`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 qualquer 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}

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

- Selecionar n
disciplina e q

n alguma

select n

EXISTS

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

M

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 )
```

Maneira mais eficiente de fazer a mesma consulta???

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

Por que é mais eficiente???

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
```



Para testar:

```
select data from jogo;
```

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

```
select sysdate from dual;
```

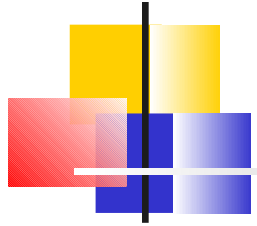
```
select to_char(sysdate, 'dd-mm-yyyy hh12:mi')  
      from dual;
```

```
select seq_nro_usp.currval from dual;
```




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