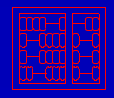
Sinônimo para nome de tabela (*alias*)

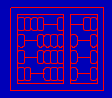


Às vezes precisamos fazer a junção de 3 tabelas: O exemplo abaixo introduz alias para as tabelas a fim de minimizar o texto da consulta.

q19: “Para cada jogador que venceu algum torneio obtenha o seu nome, o nome do torneio e o ano da vitória no torneio”.

```
select j.nome, lt.nomet, t.ano

from jogadores j, lista_torneios lt, torneios t

where j.numj = t.numj and t.numt = lt.numt
```

Consultas aninhadas (subconsultas)



operando de uma expressão booleana na cláusula where:

um outro comando select (“consulta aninhada”ou subconsulta).

Há várias formas como isto pode ser feito:

Subconsultas com o operador *in*

q27: “Obtenha o nome de cada jogador que venceu pelo menos um torneio.”

select nome

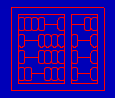
from jogadores

where numj in

(select numj from torneios)

UNICAMP

Consultas aninhadas (cont)



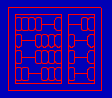
Poderíamos fazer a mesma consulta usando uma junção:

q28:

```
select distinct nome

from jogadores, torneios

where jogadores.numj = torneios.numj
```


UNICAMP

Consultas aninhadas (cont)



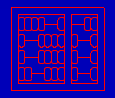
"q29: obter os nomes dos jogadores que venceram uma ou mais vezes o torneio 1":

```
select nome

from jogadores

where numj in

(select numj from torneios

where numt = 1),
```

q30: O exemplo acima poderia também ser feito com uma junção (tente!).

UNICAMP

Consultas aninhadas (cont)

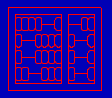


Há casos em que não podemos usar uma consulta aninhada no lugar de uma junção:

q31: “Para cada jogador que venceu torneios, obtenha o seu nome, o número do torneio e o ano”.

```
select nome, numt, ano

from jogadores, torneios

where jogadores.numj = torneios.numj
```

Consultas aninhadas (cont)

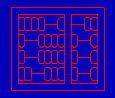


O contrário também é verdadeiro: há casos em que uma consulta aninhada não pode ser substituída por uma junção:

q32: “obtenha os nomes dos jogadores que não venceram torneios”:

```
select nome

from jogadores

where numj not in

(select numj from torneios)
```

Consultas aninhadas (cont)

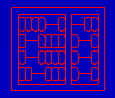


E se intuitivamente usarmos para a consulta: “obtenha os nomes dos jogadores que não venceram torneios”, a seguinte junção ?:

q33:

```
select nome

from jogadores, torneios

where jogadores.numj <> torneios.numj
```

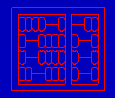
Subconsulta com operador de comparação



q34: “Obtenha o pre-nome e o nome do jogador que ganhou o torneio 2 em 1997”.

```
select pnome, nome

from jogadores

where numj =

(select numj

from Torneios

where numt=2 and ano= 1997
```

Subconsulta com operador de comparação

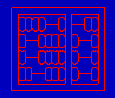


q35: “Obtenha o nome de cada jogador que é mais velho que Kuerten”.

```
select nome

from jogadores

where ano_n <

(select ano_n

from jogadores

where nome = 'Kuerten')
```


UNICAMP

Subconsulta com operador de comparação

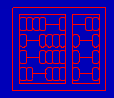


q36: “Obtenha o(s) nome(s) do(s) jogador(es) mais velho(s) que todos os outros jogadores”.

Deve ser óbvio que a solução abaixo é incorreta:

```
select nome

from jogadores

where ano_n <=

(select ano_n from jogadores)
```

Operadores *all e any* em subconsultas



O operador all resolve o problema da solução errônea da consulta anterior:

select nome

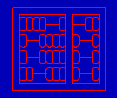
from jogadores

where ano_n $<=$ all

(select ano_n from jogadores),

onde $v\ \theta\ all\ (c_1, c_2, \ldots, c_n)$ significa:
$v\ \theta\ c_1\ and\ v\ \theta\ c_2\ and\ \ldots v\ \theta\ c_n$

Operadores *all e any* em subconsultas



Operador any: q37: “ obter o nome da cada jogador que não pertence ao grupo dos mais velhos”

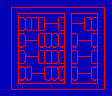
```
select nome
from jogadores
where ano_n > any
(select ano_n from jogadores)
```

$any\ (c_1, c_2, \ldots\ c_n)$ significa: “algum dentre”$c_1, c_2, \ldots\ c_n$, ou seja,
$v\ \theta\ any\ (c_1, c_2, \ldots, c_n)$ significa:
$v\ \theta\ c_1\ or\ v\ \theta\ c_2\ or\ \ldots v\ \theta\ c_n$

Observe que apenas o(s) jogador(es) mais velho(s) dá(dão) um resultado falso para a comparação acima.

UNICAMP

Cláusula *order by*



É possível especificar múltiplas colunas para ordenação:

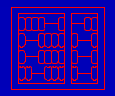
q43:

```
select numj, ano

from torneios

order by numj, ano
```

Nesse caso é feita uma ordenação lexicográfica: por jogador, e para cada jogador pelo ano do(s) torneio(s) que venceu.

UNICAMP

Cláusula *order by*



A ordenação é ascendente por default, mas pode ser descendente usando a palavra reservada *desc*

q44: select nome, cid_res

from jogadores

order by cid_res asc, nome desc

Onde aparecem valores Nulos no exemplo ?:

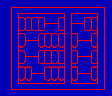
q45: select natp

from jogadores

order by natp

UNICAMP

Cláusula *order by*



dependente da implementação:

SQL Server e Sybase: aparecem antes de qualquer valor, seja para ordem ascendente ou descendente;

Oracle: aparecem em último lugar se ordem ascendente e em primeiro lugar se ordem descendente.

UNICAMP