Aninhamento de Consultas

Nomes dos pilotos que reservaram o avião nº 103:

```
SELECT  P.nome_p
FROM    Pilotos P
WHERE   P.num_p IN (SELECT R.num_p
                    FROM   Reservas R
                    WHERE  R.num_a = 103)
```

- A cláusula WHERE pode conter uma sub-consulta! (Cláusulas FROM e HAVING também podem.)
- Para achar os pilotos que não reservaram o avião nº 103, use NOT IN.
- Semântica: Para cada tupla de Pilotos, verifique se a condição é satisfeita computando a sub-consulta.

Aninhamento de Consultas Correlacionadas

Nomes dos pilotos que reservaram o avião nº 103:

```
SELECT  P.nome_p
FROM Pilotos P
WHERE  EXISTS (SELECT *
               FROM Reservas R
               WHERE R.num_p = P.num_p
                  AND R.num_a = 103)
```

- O operador EXISTS verifica se um conjunto de tuplas é não-vazio.
- Este exemplo ilustra por que, em geral, a sub-consulta precisa ser recomputada para cada tupla de Pilotos.

Mais Operações Com Conjuntos

- Já vimos IN e EXISTS. Há também NOT IN e NOT EXISTS.
- Também disponíveis: *op* ANY e *op* ALL (onde *op* pode ser >, <, =, >=, <=, ou <>).
- Pilotos mais velhos que algum piloto chamado Zé:

```
SELECT  *
FROM    Pilotos P
WHERE   P.idade > ANY (SELECT P2.idade
                       FROM   Pilotos P2
                       WHERE  P2.nome_p = 'Zé')
```


Usando IN em vez de INTERSECT

Números dos pilotos que reservaram um Cessna <u>e</u> um Learjet:

```
SELECT P.num_p
FROM   Pilotos P, Avioes A, Reservas R
WHERE P.num_p = R.num_p
    AND A.num_a = R.num_a AND A.tipo = 'Cessna'
    AND P.num_p IN (SELECT P2.num_p
                    FROM   Pilotos P2, Avioes A2, Reservas R2
                    WHERE P2.num_p = R2.num_p
                        AND A2.num_a = R2.num_a
                        AND A2.tipo = 'Learjet')
```

- De modo análogo, consultas com EXCEPT podem ser reescritas usando NOT IN.
- Para achar os nomes (em vez dos números) desses pilotos, basta trocar P.num_p por P.nome_p no primeiro SELECT.
  - Como fazer essa mudança na consulta com INTERSECT?

Divisão em SQL

Nomes dos pilotos que reservaram todos os aviões:

❶ Usando EXCEPT ⇨

```
SELECT P.nome_p
FROM    Pilotos P
WHERE NOT EXISTS
          ( (SELECT A.num_a
             FROM    Avioes A)
            EXCEPT
            (SELECT R.num_a
             FROM Reservas R
             WHERE R.num_p = P.num_p))
```

❷ Sem EXCEPT

⇩

```
SELECT P.nome_p
FROM    Pilotos P
WHERE NOT EXISTS (SELECT A.num_a
                  FROM    Avioes A
                  WHERE NOT EXISTS (SELECT R.num_a
                                    FROM    Reservas R
                                    WHERE R.num_a = A.num_a
                                    AND R.num_p = P.num_p))
```

Pilotos P tais que…

não há avião A…

sem uma reserva de A para P

Funções de Agregação

- Extensão significativa da álgebra relacional.

COUNT (*)
COUNT ([DISTINCT] *atrib*)
SUM ([DISTINCT] *atrib*)
AVG ([DISTINCT] *atrib*)
MAX (*atrib*)
MIN (*atrib*)

```
SELECT COUNT(DISTINCT P.nivel)
FROM Pilotos P
WHERE P.nome_p = 'Jorge'
```

```
SELECT AVG (P.idade)
FROM Pilotos P
WHERE P.nivel = 10
```

```
SELECT AVG (DISTINCT P.idade)
FROM Pilotos P
WHERE P.nivel = 10
```

```
SELECT COUNT (*)
FROM Pilotos P
```

```
SELECT P.nome
FROM Pilotos P
WHERE P.nivel = (SELECT MAX (P2.nivel)
                 FROM Pilotos P2)
```

Nome e idade do(s) piloto(s) mais velho(s)

- A primeira consulta é inválida! (Veremos a razão daqui a pouco, quando discutirmos GROUP BY.)
- A terceira consulta é equivalente à segunda e é permitida pelo padrão SQL-92, mas não é suportada por alguns sistemas.

```
SELECT P.nome_p, MAX (P.idade)
FROM Pilotos P

SELECT P.nome_p, P.idade
FROM Pilotos P
WHERE P.idade =
          (SELECT MAX (P2.idade)
           FROM Pilotos P2)

SELECT P.nome_p, P.idade
FROM Pilotos P
WHERE (SELECT MAX (P2.idade)
           FROM Pilotos P2)
          = P.idade
```


GROUP BY e HAVING

- Até agora aplicamos funções de agregação a todas as tuplas que satisfazem a condição no WHERE. Algumas vezes é desejável aplicá-las a vários grupos de tuplas.
- Exemplo: *Para cada nível, achar a idade do piloto mais jovem nesse nível.*
  - No caso geral, podemos não saber quantos níveis existem e quais os seus valores!
  - Supondo que existem 10 níveis numerados de 1 a 10, podemos escrever 10 consultas assim (!):

Para *i* = 1, 2, …, 10:

```
SELECT MIN (P.idade)
FROM Pilotos P
WHERE P.nivel = i
```

Consultas Com GROUP BY e HAVING

SELECT [DISTINCT] *lista-de-atribs-ou-agregações*
FROM *lista-de relações*
WHERE *condição*
GROUP BY *lista-de-atribs-para-agrupamento*
HAVING *condição-para-grupos*

A *lista-de-atribs-ou-agregações* contém:

❶ Nomes de atributos

❷ Termos com funções de agregação (ex: MIN (P.idade)).

Os nomes de atributos (❶) precisam estar todos contidos na *lista-de-atribs-para-agrupamento*. Intuitivamente, cada tupla do resultado corresponde a um grupo, e esses atributos precisam ter o mesmo valor sobre todo o grupo.

Avaliação Conceitual

- Efetue o produto cartesiano da *lista-de-relações*, elimine as tuplas que não satisfazem à *condição*, remova as colunas "desnecessárias" e particione as tuplas restantes em grupos, de acordo com a *lista-de-atribs-para-agrupamento*. (Um grupo é um conjunto de tuplas que coincidem sobre todos os atributos dessa lista.)
- Aplique a *condição-para-grupos* e elimine os grupos que não atenderem esse requisito. Toda expressão que aparecer nessa condição deverá ter um valor único por grupo!
  - Precisa estar na *lista-de-atribs-para-agrupamento* todo atributo que aparecer na *condição-para-grupos* e não for argumento para uma função de agregação. (SQL não explora a semântica de chaves aqui!)
- Para cada grupo "sobrevivente", gere uma tupla resultado.

Ache a idade do mais jovem dos pilotos com idade ≥ 18, para cada nível com pelo menos dois desses pilotos

```
SELECT P.nivel, MIN (P.idade)
FROM Pilotos P
WHERE P.idade >= 18
GROUP BY P.nivel
HAVING COUNT (*) > 1
```

num_p	nome_p	nível	idade
22	Joca	7	45
31	Laerte	8	55
71	Zildo	10	16
64	Henry	7	35
29	Bronco	1	33
58	Rafa	10	35

nível	idade
1	33
7	45
7	35
8	55
10	35

Nível	
7	35

Tabela resultado

- Só P.nivel e P.idade aparecem nas cláusulas SELECT, GROUP BY ou HAVING; os demais atributos são 'desnecessários'.
- A segunda coluna do resultado não tem nome. (Use AS para nomeá-la.)

Quantas reservas existem para cada avião tipo Learjet?

```
SELECT A.num_a, COUNT (*) AS reservas
FROM Pilotos P, Avioes A, Reservas R
WHERE P.num_p = R.num_p
    AND R.num_a = A.num_a AND A.tipo = 'Learjet'
GROUP BY A.num_a
```

- Formando grupos sobre a junção de tres relações.
- O que obtemos retirando A.tipo = 'Learjet' da cláusula WHERE e adicionando uma cláusula HAVING com essa condição?
- E se retirarmos Pilotos P e a condição sobre P.num_p?

Ache a idade do mais jovem dos pilotos com idade ≥ 18, para cada nível com pelo menos 2 pilotos (de ∀ idade)

```
SELECT P.nivel, MIN (P.idade)
FROM Pilotos P
WHERE P.idade >= 18
GROUP BY P.nivel
HAVING 1 < (SELECT COUNT (*)
            FROM Pilotos P2
            WHERE P.nivel = P2.nivel)
```

- Mostra uma cláusula HAVING contendo uma sub-consulta.
- Compare com a consulta que levava em conta somente níveis com pelo menos dois pilotos maiores de 18!
- E se a cláusula HAVING for trocada por:
  HAVING COUNT (*) > 1

Ache os níveis para os quais a média das idades dos pilotos é a mais baixa

- Funções de agregação não podem ser aninhadas! Solução <u>errada</u>:

```
SELECT P.nivel
FROM Pilotos P
WHERE P.idade = (SELECT MIN (AVG( P2.idade) ) )
                 FROM Pilotos P2)
```

- Solução correta (em SQL-92):

```
SELECT Temp.nivel, Temp.idade_media
FROM (SELECT P.nivel, AVG (P.idade) AS idade_media
      FROM Pilotos P
      GROUP BY P.nivel) AS Temp
WHERE Temp.idade_media = (SELECT MIN (Temp.idade_media)
                          FROM Temp)
```