SQL — A Linguagem de Consulta

Exemplos de Tabelas

- Usaremos estas instâncias das relações “Pilotos” e “Reservas”.
  - Que “regra do negócio” seria diferente se a chave da relação Reservas contivesse só os atributos num_p e num_a?

R1

num_p	num_a	data
22	101	10/10/98
58	103	11/12/98

P1

num_p	nome_p	nível	idade
22	Joca	7	45
31	Laerte	8	55
58	Rafa	10	35

P2

num_p	nome_p	nível	idade
28	Zeca	9	35
31	Laerte	8	55
44	Giba	5	35
58	Rafa	10	35

Consulta SQL Básica

SELECT [DISTINCT] *lista-de-atribs*
FROM *lista-de-relações*
WHERE *condição*

- *lista-de-relações*: Uma lista de nomes de tabelas ou visões (possivelmente com uma variável tupla depois de cada nome).
- *lista-de-atribs*: Uma lista de nomes de colunas das tabelas ou visões na *lista-de-relações*.
- *condição*: comparações (Atrib *op* const ou Atrib1 *op* Atrib2, onde *op* é um de <, >, =, <=, >=, <>) combinadas com AND, OR ou NOT.
- DISTINCT: eliminar linhas repetidas.

Estratégia de Avaliação Conceitual

- A semântica de uma consulta SQL é definida pela seguinte estratégia de avaliação conceitual:
  - Compute o produto cartesiano da *lista-de-relações*.
  - Descarte as tuplas que não satisfazem a *condição*.
  - Remova colunas não listadas em *lista-de-atribs*.
  - Se houver DISTINCT, elimine repetições de linhas.
- Esta estratégia é provavelmente a maneira menos eficiente de se executar uma consulta! Um otimizador encontrará métodos mais eficientes que produzem os mesmos resultados.

Exemplo de Avaliação Conceitual

Nomes dos pilotos que reservaram o avião nº 103:

```
SELECT  P.nome_p
FROM    Pilotos P, Reservas R
WHERE   P.num_p = R.num_p  AND  R.num_a = 103
```

$$\Pi_{\text{Pilotos.nome\_p}}(\sigma_{\substack{\text{Pilotos.num\_p = Reservas.num\_p} \\ \text{AND} \\ \text{num\_a = 103}}}(\text{Pilotos} \times \text{Reservas}))$$

(num_p)	nome_p	nível	idade	(num_p)	num_a	data
22	Joca	7	45	22	101	10/10/98
22	Joca	7	45	58	103	11/12/98
31	Laerte	8	55	22	101	10/10/98
31	Laerte	8	55	58	103	11/12/98
58	Rafa	10	35	22	101	10/10/98
58	Rafa	10	35	58	103	11/12/98

Uma Nota Sobre Variáveis Tupla

- São de fato necessárias só no caso da mesma tabela aparecer mais de uma vez na cláusula FROM. A consulta anterior pode ser escrita assim:

```
SELECT  P.nome_p
FROM    Pilotos P, Reservas R
WHERE   P.num_p = R.num_p  AND  num_a = 103
```

Ou assim:

```
SELECT  nome_p
FROM    Pilotos, Reservas
WHERE   Pilotos.num_p = Reservas.num_p
   AND  num_a = 103
```

É considerado “bom estilo”, no entanto, usar sempre variáveis tupla!

Números dos pilotos que reservaram algum avião
(pelo menos um)

- Vamos supor que existem várias tabelas com informações sobre pilotos (Pilotos, Pilotos2, etc.) e que na tabela Reservas são registradas as reservas feitas por pilotos das diferentes tabelas.
- Só estamos interessados nos pilotos que estão relacionados numa dessas tabelas (a tabela Pilotos).

```
SELECT  P.num_p
FROM    Pilotos P, Reservas R
WHERE   P.num_p = R.num_p
```

- Se estivéssemos interessados nos números de <u>todos</u> os pilotos (das várias tabelas) que reservaram algum avião, a consulta seria mais simples. (Como?)

Números dos pilotos que reservaram algum avião
(pelo menos um)

```
SELECT  P.num_p
FROM    Pilotos P, Reservas R
WHERE   P.num_p = R.num_p
```

- Faria alguma diferença especificar DISTINCT nesta consulta?
- Qual o efeito de se trocar P.num_p por P.nome_p na cláusula SELECT? Faria diferença especificar DISTINCT nesse caso?
- Para obter todos os atributos desses pilotos e de suas reservas, troque a primeira linha por

SELECT * ← todos os atributos

Expressões e Strings

```
SELECT  P.nivel, nivel1 = P.nivel - 5, 2 * P.nivel AS nivel2
FROM    Pilotos P
WHERE   P.nome_p LIKE 'B_%B'
```

- Ilustra o uso de expressões aritméticas e de strings com "curingas": *Encontrar triplas (nível do piloto e mais dois campos definidos por expressões) para pilotos cujos nomes começam e terminam com B e contém pelo menos três caracteres.*
- AS e = são duas maneiras de nomear colunas do resultado.
- LIKE é usado para comparações envolvendo curingas: '_' significa qualquer caracter e '%' significa 0 ou mais caracteres arbitrários.

Números dos pilotos que reservaram um Cessna <u>ou</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' OR A.tipo = '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'
UNION
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 = 'Learjet'
```

- O que obtemos trocando este OR por AND?
- <u>UNION</u>: união de dois conjuntos de tuplas compatíveis para união.

Números dos pilotos que reservaram um Cessna e um Learjet

```
SELECT P.num_p
FROM    Pilotos P, Avioes A1, Reservas R1,
        Avioes A2, Reservas R2
WHERE P.num_p = R1.num_p AND A1.num_a = R1.num_a
   AND P.num_p = R2.num_p AND A2.num_a = R2.num_a
   AND (A1.tipo = 'Cessna' AND A2.tipo = 'Learjet')
```

Mais complicado que o caso do "ou"!

```
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'
INTERSECT
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 = 'Learjet'
```

Chave!

- INTERSECT: intersecção de dois conjuntos de tuplas compatíveis para união.

UNION, INTERSECT, EXCEPT

- Nos exemplos anteriores, contraste a simetria das consultas que usam UNION e INTERSECT com a diferença entre as outras versões das mesmas consultas.
- Também disponível: EXCEPT, que computa a diferença de conjuntos de tuplas.
  - No exemplo anterior, qual o efeito de se trocar INTERSECT por EXCEPT?
- INTERSECT e EXCEPT estão no padrão SQL-92, mas ainda não são suportados por todos os sistemas.
  - Alguns sistemas suportam MINUS em vez de EXCEPT.