



AS33C

Banco de Dados 2

Professor: Eduardo Cotrin Teixeira



cotrin@utfpr.edu.br



SQL – União, Intersecção e Diferença

- **União, intersecção e diferença** de tabelas são expressas pelas seguintes formas, todas envolvendo subconsultas:

(<subconsulta>) **UNION** (<subconsulta>)

(<subconsulta>) **INTERSECT** (<subconsulta>)

(<subconsulta>) **EXCEPT** (<subconsulta>)

*** IMPORTANTE:** as subconsultas têm que ter conteúdo do **mesmo tipo** !

SQL – União, Intersecção e Diferença

- O **UNION** une os resultados das subconsultas.

Obtenha os códigos das peças com preço menor que \$10 **ou** que foram fornecidas ao projeto P4.

```
SELECT PeNro  
FROM Peca  
WHERE PePreco < 10
```

UNION

```
SELECT PeNro  
FROM Fornece_para  
WHERE PNro = 'P4';
```

PeNro
PE4

PeNro
PE1
PE3

PeNro
PE4
PE1
PE3

SQL – União, Intersecção e Diferença

- O **INTERSECT** reúne os resultados repetidos nas subconsultas.

Obtenha os códigos das peças com preço maior que \$15 e que foram fornecidas ao projeto P4.

```
SELECT PeNro
FROM Peca
WHERE PePreco > 15
INTERSECT
SELECT PeNro
FROM Fornece_para
WHERE PNro = 'P4';
```

PeNro
PE1
PE2
PE5

PeNro
PE1

PeNro
PE1
PE3

*** Não funciona no MySQL.**

SQL – União, Intersecção e Diferença

- O **EXCEPT** exclui do resultado da primeira subconsulta o resultado da segunda subconsulta.

Obtenha os códigos das peças com preço maior que \$15 que **NÃO** foram fornecidas ao projeto P4.

```
SELECT PeNro  
FROM Peca  
WHERE PePreco > 15  
EXCEPT  
SELECT PeNro  
FROM Fornece_para  
WHERE PNro = 'P4';
```

PeNro
PE1
PE2
PE5

PeNro
PE1
PE3

PeNro
PE2
PE5

*** Não funciona no MySQL.**

SQL – União, Intersecção e Diferença

- Comandos **SELECT** usam **multiconjuntos**, ou seja, registros duplicados são **mantidos**.

Obtenha as cores das peças com preço maior que \$15.

```
SELECT PeCor  
FROM Peca  
WHERE PePreco > 15;
```

PeCor
Azul
Vermelho
Vermelho

SQL – União, Intersecção e Diferença

- União, intersecção e diferença usam **conjuntos**, ou seja, registros duplicados são **eliminados**.

Obtenha os códigos das peças com preço menor que \$10 ou que foram fornecidas ao projeto P5.

```
SELECT PeNro  
FROM Peca  
WHERE PePreco < 10  
UNION
```

PeNro
PE4

PeNro
PE4

```
SELECT PeNro  
FROM Fornece_para  
WHERE PNro = 'P5';
```

PeNro
PE4
PE4



SQL – União, Intersecção e Diferença

- Para forçar que o resultado de uma consulta seja tratado como um conjunto, usamos a cláusula **DISTINCT**:

SELECT DISTINCT...

- E, inversamente, para forçar que o resultado seja tratado como um multiconjunto, usamos a cláusula **ALL** (com UNION, INTERSECT ou EXCEPT):

... UNION ALL ...



SQL – União, Intersecção e Diferença

Obtenha os códigos de todas as peças fornecidas aos projetos.

```
SELECT PeNro  
FROM Fornece_para;
```

```
SELECT DISTINCT PeNro  
FROM Fornece_para;
```

PeNro
PE1
PE2
PE2
PE3
PE4
PE4
PE5

PeNro
PE1
PE2
PE3
PE4
PE5



SQL – União, Intersecção e Diferença

Obtenha os códigos de cada peça com preço menor que \$10 ou que foram fornecidas ao projeto P5.

```
SELECT PeNro
FROM Peca
WHERE PePreco < 10
UNION ALL
SELECT PeNro
FROM Fornece_para
WHERE PNro = 'P5';
```

PeNro
PE4

PeNro
PE4
PE4

PeNro
PE4
PE4
PE4



SQL – Operadores de Agregação

- Operadores (ou Funções) de Agregação se aplicam a colunas inteiras de uma tabela, e produzem um único resultado como retorno.
- Os principais são:

SUM, AVG, COUNT, MIN e MAX.

- Operadores podem ser aplicados a uma coluna na cláusula SELECT para produzir a agregação da coluna. A sintaxe geral é:

SELECT ..., OPERADOR(COLUNA), ...



SQL – Operadores de Agregação

■ Uso:

- **SUM(*coluna*)**: obtém a soma dos valores em *coluna*.
- **AVG(*coluna*)**: calcula a média dos valores em *coluna*.
- **COUNT(*coluna*)**: retorna o número de valores em *coluna*.
- **COUNT(*)**: retorna o número de registros em uma consulta.
- **MIN(*coluna*)**: recupera o valor mínimo em *coluna*.
- **MAX(*coluna*)**: recupera o valor máximo em *coluna*.

*Obs.: MAX, MIN e COUNT também **funcionam com texto**.

SQL – Operadores de Agregação

Encontre quantas peças estão cadastradas, o preço médio das peças, e o maior e o menor preço.

```
SELECT Count (PeNro) as QuantPecas,  
       Avg (PePreco) as PrecoMedio,  
       Max (PePreco) as MaiorPreco,  
       Min (PePreco) as MenorPreco  
FROM Peca;
```

QuantPecas	PrecoMedio	MaiorPreco	MenorPreco
5	21.2	43	9

Tabela Peça

PeNro	PeNome	PePreco	PeCor
PE1	Cinto	22	Azul
PE2	Volante	18	Vermelho
PE3	Lanterna	14	Preto
PE4	Limpador	09	Amarelo
PE5	Painel	43	Vermelho

SQL – Operadores de Agregação

Encontre o preço médio das peças utilizadas no projeto P4.

```
SELECT Avg(PePreco) as PrecoMedioP4
FROM Peca NATURAL JOIN Fornece_para
WHERE PNro = 'P4';
```

PrecoMedioP4
18

Encontre quantas peças diferentes foram usadas nos projetos.

```
SELECT Count(DISTINCT PeNro) as QuantPecas
FROM Fornece_para;
```

QuantPecas
5

Tabela Fornece_para

PeNro	FNro	PNro	Quant
PE1	F5	P4	5
PE2	F2	P2	1
PE3	F3	P4	2
PE4	F4	P5	3
PE5	F1	P1	1
PE2	F2	P3	1
PE4	F3	P5	2

Tabela Peça

PeNro	PeNome	PePreco	PeCor
PE1	Cinto	22	Azul
PE2	Volante	18	Vermelho
PE3	Lanterna	14	Preto
PE4	Limpador	09	Amarelo
PE5	Painel	43	Vermelho

SQL – Operadores de Agregação

- Operadores **só podem ser usados no SELECT**, portanto para consulta a seguir:

Mostre o nome da peça mais cara (NÃO FUNCIONA !).

```
SELECT PeNome  
FROM Peca  
WHERE PePreco = max(PePreco) ;
```

O correto seria:

```
SELECT PeNome  
FROM Peca  
WHERE PePreco = SELECT max(PePreco) from Peca ;
```

PeNome
Painel

*** Mas vale lembrar que o mais apropriado seria o uso do ALL !**



SQL – Operador de Agrupamento

- Depois de uma expressão SELECT-FROM-WHERE podemos adicionar **GROUP BY** e uma lista de colunas.
- O resultado do SELECT vai ser mostrado conforme os **grupos formados pelos valores nas colunas**.
- Qualquer agregação é aplicada dentro de cada grupo.

SQL – Operador de Agrupamento

Número de fornecedores por cidade:

```
SELECT FCidade, count(*) as Quant
FROM Fornecedor
GROUP BY FCidade;
```

FCidade	Quant
São Paulo	1
Campinas	2
São Carlos	1
Piracicaba	1

- Os registros da tabela Fornecedor são **separados em grupos** (Cidade) e a função COUNT é **aplicada a cada grupo** separadamente.

- Cada elemento da lista do SELECT **precisa ser** uma **agregação**, ou uma **coluna citada na lista do GROUP BY**.

Tabela Fornecedor

FNro	FNome	FCidade	FCateg
F1	Plastec	Campinas	B
F2	C&M	São Paulo	D
F3	Kirurgic	Campinas	A
F4	Piloto's	Piracicaba	A
F5	Equipment	São Carlos	C

SQL – Operador de Agrupamento

Obtenha a quantidade de cada peça utilizada nos projetos.

```
SELECT PeNro, Sum(Quant)
FROM Fornece_para
GROUP BY PeNro;
```

PeNro	SUM(Quant)
PE2	2
PE4	5
PE1	5
PE3	2
PE5	1

Mostre o nome e o número de cada projeto, com o total de peças que ele usou.

```
SELECT PNro, PNome, Sum(Quant) as TotalPecas
FROM Fornece_para NATURAL JOIN Projeto
GROUP BY PNro, PNome;
```

PNro	PNome	TotalPecas
P3	Alfa	1
P5	Paraíso	5
P1	Detroit	1
P4	Sea	7
P2	Pegasus	1

* Obs.: O agrupamento e as funções são aplicadas após a junção.



SQL – Operador de Ordenação

- O operador **ORDER BY** permite ordenar o resultado de uma consulta por um ou mais atributos.
- O operador deve ser incluído **ao final da consulta**. A ordem padrão é ascendente (**ASC**), para ordem decrescente deve ser usado **DESC**.

Número de fornecedores por cidade, em ordem alfabética.

```
SELECT FCidade, Count(*) as Quant  
FROM Fornecedor  
GROUP BY FCidade  
ORDER BY FCidade;
```

FCidade	Quant
Campinas	2
Piracicaba	1
São Carlos	1
São Paulo	1



SQL – Operador de Ordenação

Obtenha a quantidade de cada peça utilizada nos projetos, em ordem decrescente de quantidade.

```
SELECT PeNro, Sum(Quant) as Total  
FROM Fornece_para  
GROUP BY PeNro  
ORDER BY Total DESC;
```

PeNro	Total
PE4	5
PE1	5
PE2	2
PE3	2
PE5	1

SQL – Cláusula HAVING

- **HAVING <condição>** pode aparecer depois da cláusula GROUP BY, para permitir a **inclusão de condições nos grupos**.
- Se aparecer, a **condição é aplicada sobre cada grupo**. Grupos que não satisfazem a condição são eliminados da resposta da consulta.

Número de fornecedores por cidade, para cidades com 2 ou mais fornecedores.

```
SELECT FCidade, Count(*) as Quant  
FROM Fornecedor  
GROUP BY FCidade  
HAVING Count(*) >= 2;
```

FCidade	Quant
Campinas	2



SQL – Cláusula HAVING

Obtenha o nome e o número de cada projeto que usou 5 ou mais peças, com o total de peças que ele usou.

```
SELECT PNro, PNome, Sum(Quant) as TotalPecas  
FROM Fornece_para NATURAL JOIN Projeto  
GROUP BY PNro, PNome  
HAVING Sum(Quant) >= 5;
```

PNro	PNome	TotalPecas
P5	Paraíso	5
P4	Sea	7