



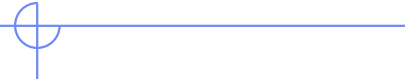
SQL

Linguagem de Manipulação de Dados

Prof. Humberto Razente

Bloco B - sala 1B144

SELECT



```
SELECT <lista de atributos e funções>  
FROM <lista de tabelas>  
[ WHERE predicado ]  
[ GROUP BY <atributos de agrupamento> ]  
[ HAVING <condição para agrupamento> ]  
[ ORDER BY <lista de atributos> ]
```

Junção Natural

◆ SQL1 ou SQL86

- não tem uma representação para a operação de junção

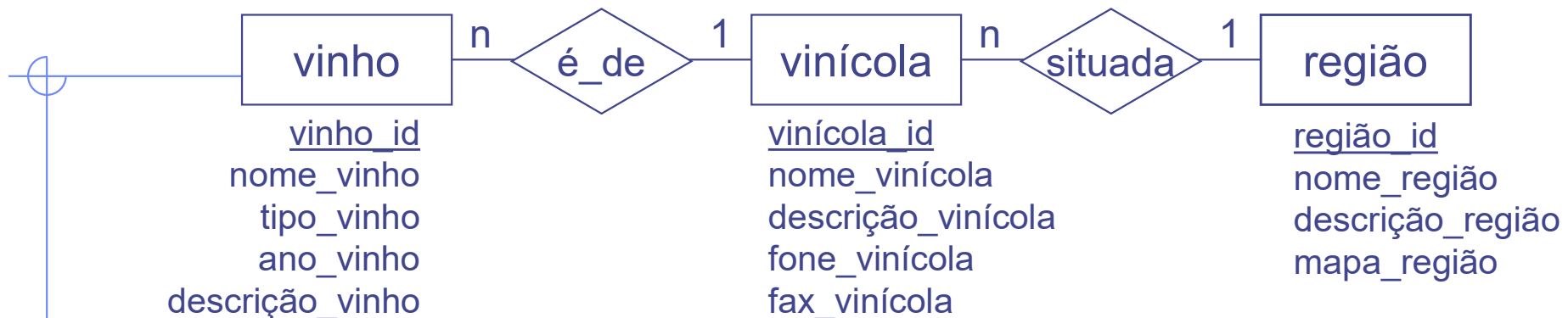
◆ Definida em termos de

- um produto cartesiano
- uma seleção
- uma projeção

Junção

- ◆ Não é representada explicitamente
- ◆ Cláusulas SELECT e WHERE
 - especificam atributos com mesmo nome usando o nome da tabela e o nome do atributo (nome_tabela.nome_atributo)
- ◆ Cláusula FROM
 - possui mais do que uma tabela
- ◆ Cláusula WHERE
 - inclui as condições de junção

Exemplos



- ◆ `SELECT nome_vinícola, nome_região`
`FROM vinícola, região`
`WHERE vinícola.região_id = região.região_id;`
- ◆ `SELECT nome_vinícola, nome_região, nome_vinho`
`FROM vinícola, região, vinho`
`WHERE vinícola.região_id = região.região_id AND`
`vinho.vinícola_id = vinícola.vinícola_id;`

Junções

◆ SQL2 (SQL92)

- CROSS JOIN → produto cartesiano
- INNER JOIN → junção interna
- LEFT OUTER JOIN → junção externa a esquerda
- RIGHT OUTER JOIN → junção externa a direita
- FULL OUTER JOIN → junção externa completa

Junção: exemplos

```
SELECT *  
  FROM clima CROSS JOIN cidade;
```

```
SELECT *  
  FROM clima INNER JOIN cidade  
         ON (clima.cidade = cidade.nome);
```

```
SELECT *  
  FROM clima LEFT OUTER JOIN cidade  
         ON (clima.cidade = cidade.nome);
```

Cláusula ORDER BY

- ◆ Ordena as tuplas que aparecem no resultado de uma consulta
 - asc (padrão): ordem ascendente
 - desc: ordem descendente

- ◆ Ordenação pode ser especificada em vários atributos
 - a ordenação referente ao primeiro atributo é prioritária. Se houver valores repetidos, então é utilizada a ordenação referente ao segundo atributo, e assim por diante

Exemplo

- ◆ Liste os dados da relação vinícola. Ordene o resultado pelo nome da vinícola em ordem descendente e pela região da vinícola em ordem ascendente.

```
SELECT *  
  FROM vinícola, região  
 WHERE vinícola.região_id = região.região_id  
 ORDER BY nome_vinícola desc, nome_região asc;
```

```
SELECT *  
  FROM vinícola INNER JOIN região  
           ON (vinícola.região_id = região.região_id)  
 ORDER BY nome_vinícola desc, nome_região asc;
```

SELECT DISTINCT / ALL

◆ Observação

- DISTINCT: não considera tuplas duplicados
- ALL: inclui todas as tuplas
 - ◆ declaração é opcional
- Exemplos:
 - ◆ `SELECT distinct ano_vinho FROM vinho`
 - ◆ `SELECT ano_vinho FROM vinho` → ALL omitido

Funções de Agregação

◆ Funções

- Média → AVG()
- Mínimo → MIN()
- Máximo → MAX()
- Soma → SUM()
- Contagem → COUNT()

Funções de Agregação

◆ Características

- recebem uma coleção de valores como entrada
- retornam um valor

◆ Entrada

- `sum()` e `avg()`: conjunto de números
- demais funções: tipos de dados numéricos e não-numéricos

Funções de Agregação

vinho (vinho_id, nome_vinho, tipo_vinho, preço, vinícola_id)

vinho_id	nome_vinho	tipo_vinho	preço	vinícola_id
10	Amanda	tinto	100,00	1
09	Belinha	branco	200,00	1
05	Camila	rosê	300,00	1
15	Daniela	branco	250,00	2
27	Eduarda	branco	150,00	2
48	Fernanda	tinto	7,00	2
13	Gabriela	tinto	397,00	3
12	Helena	branco	333,00	3

Exemplos

vinho_id	nome_vinho	tipo_vinho	preço	vinícola_id
10	Amanda	tinto	100,00	1
09	Belinha	branco	200,00	1
05	Camila	rosê	300,00	1

◆ Qual a *média* dos preços?

```
SELECT AVG (preço)
FROM vinho
```

217,125

◆ Qual a *soma* dos preços?

```
SELECT SUM (preço)
FROM vinho
```

1737,00

◆ Qual o preço mais *baixo*?

```
SELECT MIN (preço)
FROM vinho
```

7,00

◆ Qual o preço mais *alto*?

```
SELECT MAX (preço)
FROM vinho
```

397,00

Exemplos

vinho_id	nome_vinho	tipo_vinho	preço	vinícola_id
10	Amanda	tinto	100,00	1
09	Belinha	branco	200,00	1
05	Camila	rosê	300,00	1

◆ *Quantos* vinhos existem na relação vinho?

```
SELECT COUNT (vinho_id)
```

```
FROM vinho
```

8

◆ Quantos tipos de vinho *diferentes* existem na relação vinho?

```
SELECT COUNT (DISTINCT tipo_vinho)
```

```
FROM vinho
```

3

Cláusula GROUP BY

◆ Funcionalidade

- permite aplicar uma função de agregação não somente a um conjunto de tuplas, mas também a um grupo de conjunto de tuplas

◆ Grupo de conjunto de tuplas

- conjunto de tuplas que possuem o mesmo valor para os atributos de agrupamento

◆ Semântica da respostas

- atributos de agrupamento no GROUP BY também devem aparecer no SELECT

Exemplos

vinho_id	nome_vinho	tipo_vinho	preço	vinícola_id
10	Amanda	tinto	100,00	1
09	Belinha	branco	200,00	1
05	Camila	rosê	300,00	1

◆ Qual o preço mais alto e a *média* dos preços *por tipo de vinho*?

```
SELECT tipo_vinho, MAX (preço) AS PMAX, AVG (preço) AS PMED  
FROM vinho  
GROUP BY tipo_vinho
```

tipo_vinho	PMAX	PMED
branco	333	233,25
rosê	300	300
tinto	397	168

Cláusula HAVING

◆ Funcionalidade

- permite especificar uma condição de seleção para grupos

◆ Resposta

- recupera os valores para as funções somente para aqueles grupos que satisfazem à condição imposta na cláusula HAVING

Exemplos

vinho_id	nome_vinho	tipo_vinho	preço	vinícola_id
10	Amanda	tinto	100,00	1
09	Belinha	branco	200,00	1
05	Camila	rosê	300,00	1

- Qual o preço mais alto e a *média* dos preços *por tipo de vinho*, para médias de preços superiores a R\$200,00

```
SELECT tipo_vinho, MAX (preço) AS PMAX,  
        AVG (preço) AS PMED
```

```
FROM vinho
```

```
GROUP BY tipo_vinho
```

```
HAVING AVG (preço) > 200
```

tipo_vinho	PMAX	PMED
branco	333	233,25
rosê	300	300

Operações de Conjuntos

SQL	Álgebra Relacional
UNION	União (\cup)
INTERSECT	Intersecção (\cap)
EXCEPT *	Diferença ($-$)

- Observações
 - as relações participantes das operações precisam ser *compatíveis*
 - operações oferecidas dependem do SGBD

Exemplo

- ◆ Liste os anos de fabricação dos vinhos para vinhos tintos e brancos

```
SELECT ano_vinho
FROM vinho
WHERE tipo_vinho = 'tinto'
UNION ALL
SELECT ano_vinho
FROM vinho
WHERE tipo_vinho = 'branco';
```

ALL → não
elimina valores
duplicados

Exercício

Paciente (CPF, Nome, DataNascimento, Endereco)

Consulta (CPF, CodigoConvenio, CRM, UF, Data, Horario, Valor)

Médico (CRM, UF, Nome, CPF, Endereco, telefone)

Responda as seguintes consultas em SQL:

- 1) Buscar os nomes dos médicos e dos pacientes das consultas agendadas para 20/01/15 ordenados pelo nome do médico e pelo horário da consulta
- 2) Buscar o número de consultas que cada médico realizou em cada data, bem como o valor total recebido em cada data. Considere que consultas realizadas são aquelas que tem valor diferente de nulo. Ordenar por médico e data.

Bibliografia

- ◆ Elmasri, Ramez; Navathe, Shamkant B. **Sistemas de banco de dados.** 6ª ed. São Paulo: Addison Wesley, 2011
 - capítulo 4: SQL Básica