

Introdução à SQL

Comandos Select-From-Where

Sub-consultas

Agregação e agrupamento

Prof. Dr. Luis Mariano del Val Cura

SQL

- SQL : *Structured Query Language*
- SQL é uma linguagem de alto nível.
 - Um comando nela indica “o que fazer” em vez de “como fazer”
 - Evita muitos detalhes de manipulação de dados necessários em linguagens procedurais como C, Pascal, Java.
 - Baseada fundamentalmente em Álgebra Relacional .

SQL

- Estabelecida como linguagem padrão para consulta e manipulação em SGBD Relacionais.
- Sistemas de Gerenciamento de Bancos de Dados (SGBD) interpretam qual a melhor maneira de executar consultas.
 - Otimização de consultas

Componentes de SQL

- DML (*Data Manipulation Language*)
 - Especificação de Consultas
 - Inserção, remoção e atualização de dados
- DDL – (*Data Definition Language*)
 - Definição, Remoção e modificação de Esquemas Relações (Esquemas Relacionais)
 - Especificação de restrições de integridade
 - Definição de visões
 - Especificação de direitos de acesso e proteção dos dados

Comando Select-From-Where

SELECT atributos desejados

FROM uma ou mais tabelas

WHERE condição das tuplas das tabelas

Especificação de Consultas

Forma geral:

```
SELECT  A1, A2, . . . , An  
FROM    r1, r2, . . . , rm  
WHERE    P
```

Onde

A_i : atributos r_j : relações P : predicado

Equivalente à expressão da Algebra Relacional

$$\Pi_{\langle A_1, A_2, \dots, A_n \rangle} (\sigma_{\langle P \rangle} (r_1 \times r_2 \times \dots \times r_m))$$

Exemplo para as aulas

- Esquema de Banco de dados utilizado nos exemplos

- Sublinhado indica chave primária.

Cervejas(nome, fabr)

Bares(nome, endr, alvara)

Pessoas(nome, endr, fone)

Gostam(pessoa, cerveja)

Vendem(bar, cerveja, preço)

Frequenta(pessoa, bar)

Exemplo

□ Usando `Cervejas(nome, fabr)`,

Quais cervejas são feitas pela Ambev?

```
SELECT nome
```

```
FROM Cervejas
```

```
WHERE fabr = 'Ambev';
```

SQL utiliza apóstrofes para strings.

SQL não é sensível a maiúsculas, exceto dentro dos strings.

Resultado da consulta

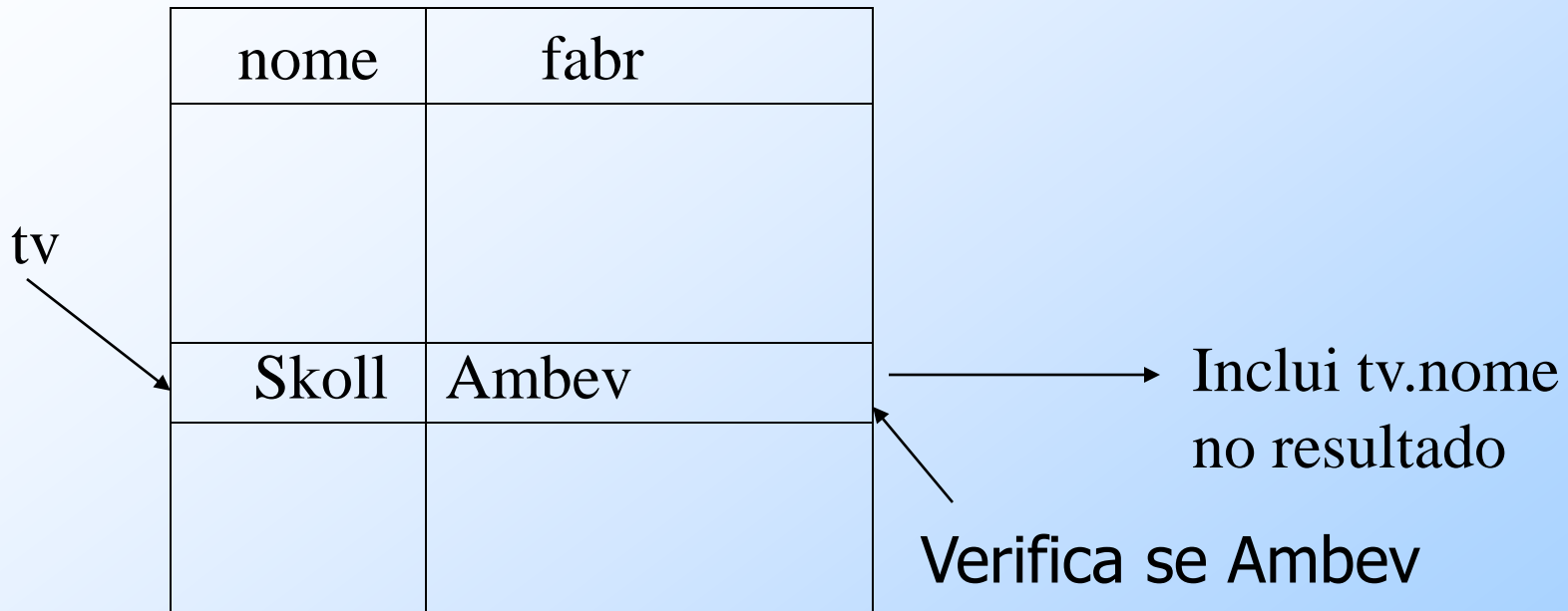
nome
Skoll
Antartica
Brahma
. . .

A resposta é uma relação com um atributo simples, *nome* e tuplas com o nome de cada cerveja produzida pela Ambev

Uma consulta a tabela simples

- Inicia com a relação na cláusula FROM.
- Aplica a seleção indicada na cláusula WHERE.
- Aplica a projeção estendida definida pela cláusula SELECT.

Semântica operacional



Semântica operacional

- Para implementar este algoritmo pense em uma *variável de tupla* percorrendo cada tupla da relação da cláusula FROM.
- Verifica se a tupla “atual” satisfaz a condição da cláusula WHERE.
- Sim, calcula os atributos ou expressões na cláusula SELECT usando os componentes da tupla.

* em cláusulas SELECT

- Quando temos uma relação na cláusula FROM, o * na cláusula SELECT significa “todos os atributos desta relação.”
- Exemplo com Cervejas(nome, fabr)

```
SELECT *  
FROM Cervejas  
WHERE fabr = 'Ambev';
```

Resultado da consulta:

nome	fabr
Skoll	Ambev
Antártica	Ambev
Brahma	Ambev
.

O resultado contem todos os atributos de Cervejas.

Renomear atributos

- Se deseja o resultado com outro nome de atributo, usar "AS <novo nome>".
- Exemplo em **Cervejas(nome, fabr):**

```
SELECT nome AS cerveja, fabr  
FROM Cervejas  
WHERE fabr = 'Ambev'
```

Resultado da consulta:

cerveja	fabr
Skoll	Ambev
Antartica	Ambev
Brahma	Ambev
.

Expressões na cláusula SELECT

- Qualquer expressão que tenha sentido pode aparecer como elemento da cláusula SELECT.

- Exemple: de `Vendem(bar, cerveja, preço)`

```
SELECT bar, cerveja,  
       preço / 1.80 AS preço_dolar  
FROM Vendem;
```

Resultado da consulta:

bar	cerveja	preço_dolar
Vitoria	Antartica	1.50
João	Skoll	1.64
...

Outro exemplo: Expressões constantes

□ Usando `Gostam(pessoa, cerveja)`

```
SELECT pessoa,  
'gosta de Skoll' AS QuemGostaSkoll  
FROM Gostam  
WHERE cerveja = 'Skoll';
```

Resultado da consulta:

Pessoa	QuemGostaSkoll
Pedro	gosta de Skoll
Ana	gosta de Skoll
...	...

Condições complexas na cláusula WHERE

- Usando `Vendem(bar, cerveja, preço)` , achar o preço que vendem a Skoll no Vitoria:

```
SELECT preço
FROM Vendem
WHERE bar = 'Vitoria' AND
       cerveja = 'Skoll';
```

Padrões

- Cláusulas WHERE podem ter condições de comparação de padrões de *strings*.
- Forma geral :
 <Atributo> LIKE <padrão> ou
 <Atributo> NOT LIKE <padrão>
- Padrão está identificado por
- % = "Qualquer string";
- _ = "Qualquer carater."

Exemplo

- ❑ Usando `Pessoas(nome, endr, fone)`
busque as pessoas com fone de Campinas

```
SELECT nome
FROM Pessoas
WHERE
fone LIKE '%019-_____' ;
```

Valores Nulos

- Tuplas em relações SQL podem ter valores NULL para uma ou mais componentes.
- Interpretação depende do contexto. Dois casos comuns:
 - *Valor desconhecido* : ex., sabemos que o João tem uma *idade*, mas não sabemos qual é.
 - *Valor não definido*: ex., o valor do atributo *esposa* de uma pessoa solteira.
 - *Valor não aplicável* : ex: CNH de uma pessoa de 4 anos

Comparando NULLs com valores

- A lógica das condições em SQL realmente tem três valores : TRUE, FALSE, UNKNOWN.
- Quando um valor é comparado com NULL, o valor resultante é UNKNOWN.
- Mas uma consulta coloca uma tupla na resposta se o valor para a cláusula WHERE é TRUE (não FALSE ou UNKNOWN).

Verificação de valor NULL

Utilizamos

✓ **A IS NULL** (resultado TRUE ou FALSE)

x **A = NULL** (resultado UNKNOWN)

```
SELECT bar  
FROM Vendem  
WHERE cerveja IS NULL
```

Consultas com múltiplas relações

- A maioria das consultas combinam dados de mais de uma relação.
- Podemos referenciar mais de uma relação em uma consulta incluindo-as na clausula FROM.
- Atributos do mesmo nome podem ser identificados "`<relação>.<atributo>`"

Exemplo

- Usando as relações **Gostam(pessoa, cerveja)** e **Frequenta(pessoa, bar)**, encontre as cervejas das que gosta pelo menos uma pessoa que frequenta o bar Vitoria.

```
SELECT cerveja
FROM Gostam, Frequenta
WHERE bar = 'Vitoria' AND
Frequenta.pessoa = Gostam.pessoa;
```

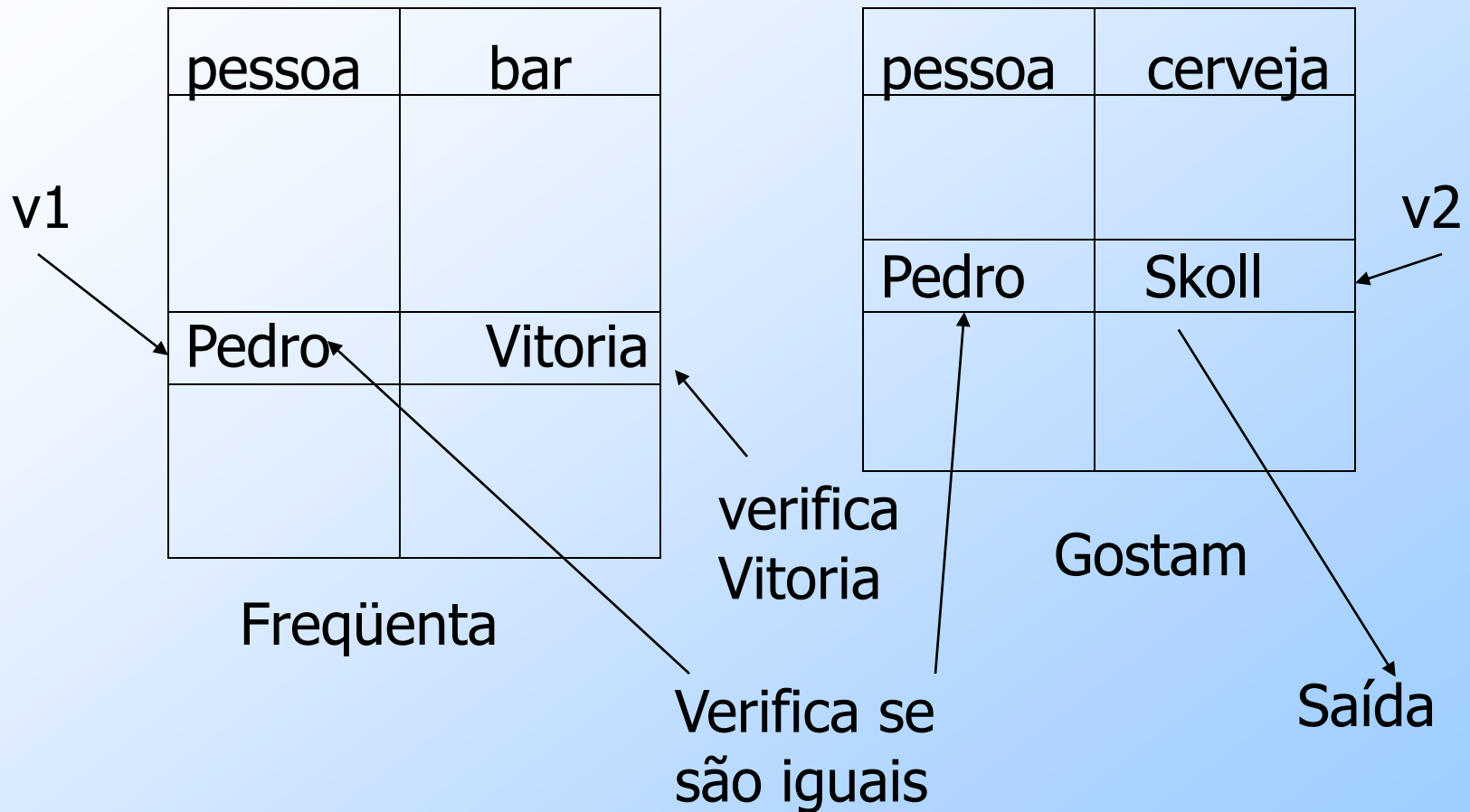
Semântica Formal

- Quase a mesma que uma consulta de simples relação:
 1. Começa com o produto cartesiano de todas as relações na cláusula FROM.
 2. Aplicar a condição de seleção da cláusula WHERE.
 3. Projetar a lista de atributos e expressões da cláusula SELECT.

Semântica Operacional

- Imagine uma variável de tupla para cada relação na cláusula FROM.
 - As variáveis visitam cada combinação de tuplas, uma por cada relação.
- Se as tuplas apontadas pelas variáveis satisfazem a condição da cláusula WHERE enviar essas tuplas para posterior projeção dos atributos na cláusula SELECT

Exemplo



Variáveis de tupla explícitas

- As vezes, uma consulta precisa usar duas cópias da mesma relação.
- Diferenciamos cada cópia pelo nome da variável de tupla, na cláusula FROM.
- Sempre é uma opção re-nomear as relações mesmo quando não seja uma necessidade.

Exemplo

- ❑ Usando **Cervejas(nome, fabr)**
- ❑ Busque todos os pares de cervejas que são do mesmo fabricante.
 - ❑ Não pode criar (Skoll, Skoll).
 - ❑ Gerar os pares em ordem alfabética, ex. (Brahma, Skoll), não (Skoll, Brahma).

```
SELECT b1.nome, b2.nome  
FROM Cervejas b1, Cervejas b2  
WHERE b1.fabr = b2.fabr AND  
      b1.nome <> b2.nome;
```

Ordenação

- ORDER BY L_1, L_2, \dots, L_n
 - L_i é uma lista de alguns atributos de relação.
- São ordenadas inicialmente pelo primeiro atributo de L_1 , depois pelo segundo atributo de L_2 e assim sucessivamente.

Exemplo: Ordenação

R = (

A	B
3	2
1	4
5	2

)

```
SELECT      A , B
FROM        R
ORDER BY    B, A
```

(A , B)

1	2
3	2
5	4

Exemplo: Ordenação

Empregados (

<u>RG,</u>	Nome ,	Sexo ,	Salario ,	Codigo)
2232	João	M	700.00	1
4050	Pedro	M	700.00	1
2245	Ana	F	1100.00	2
8960	Roberto	M	1800.00	3
7865	Claudia	F	1100.00	2
0983	Helena	F	600.00	NULL
2348	Jose	M	700.0	3

Primeiro critério
(salário)

Segundo critério
(Nome)

R (

<u>RG,</u>	Nome ,	Sexo ,	Salario ,	Codigo)
0983	Helena	F	600.00	NULL
2232	João	M	700.00	1
2348	Jose	M	700.0	3
4050	Pedro	M	700.00	1
2245	Ana	F	1100.00	2
7865	Claudia	F	1100.00	2
8960	Roberto	M	1800.00	3

SELECT *
FROM
ORDER BY

Empregados
Salario, Nome

União, Interseção e diferença

- União, interseção e diferença de relações são definidas da seguinte forma envolvendo sub-consultas: :
 - (sub-consulta) UNION (sub-consulta)
 - (sub-consulta) INTERSECT(sub-consulta)
 - (sub-consulta) EXCEPT (sub-consulta)

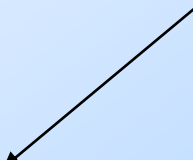
Exemplo

- Usando `Gostam(pessoa, cerveja)`, `Vendem(bar, cerveja, preço)` e `Frequenta(pessoa, bar)`, encontrar pessoas e cervejas tais que:
 1. A pessoa goste da cerveja e,
 2. A pessoa frequente pelo menos um bar que vende essa cerveja.

Solução

(SELECT * FROM Gostam)
INTERSECT

A pessoa frequenta
um bar que vende a
cerveja



```
(SELECT pessoa , cerveja  
FROM Vendem, Frequenta  
WHERE Frequenta.bar = Vendem.bar  
);
```

Semântica de Coleção

- Embora `SELECT-FROM-WHERE` utilize semântica de coleção, o padrão para união, interseção e diferença é semântica de conjuntos.
- Isto é, tuplas duplicadas são eliminadas enquanto a operação é aplicada.

Controle da eliminação de duplicadas

- O resultado pode ser forçado a ser conjunto usando:
SELECT DISTINCT . . .
- O resultado pode ser forçado a ser coleção (i.e., não elimina duplicadas) usando ALL,
. . . UNION ALL . . .

Exemplo: DISTINCT

- Usando `Vendem(bar, cerveja, preço)` busque todos os preços diferentes existentes para as cervejas:

```
SELECT DISTINCT preço  
FROM Vendem;
```

- Note que sem o `DISTINCT`, cada preço vai ser listado tantas vezes como exista um par bar/cerveja com esse preço.

Exemplo: ALL

- Usando as relações **Frequenta(pessoa, bar)** e **Gostam(pessoa, cerveja)**:

```
(SELECT pessoa FROM Frequenta)
```

```
EXCEPT ALL
```

```
(SELECT pessoa FROM Gostam) ;
```

- Fornece as pessoas tais que a quantidade de bares que freqüentam é maior que a quantidade de cervejas das que gostam.
- Tantas tuplas de uma pessoa como a quantidade dessa diferença.

Expressões de Junção

- SQL oferece várias versões de junção (de coleções).
- Estas expressões podem ser consultas independentes ou podem ser utilizadas no lugar de relações na cláusula FROM.

Junção de produto ou natural

- Produto cartesiano:

`R CROSS JOIN S;`

- Junção natural:

`R NATURAL JOIN S;`

- Exemplo:

`Gostam NATURAL JOIN Vendem;`

- Relações podem ser utilizadas entre parênteses também.

Junção Theta

R (INNER) JOIN S ON <condição>

□ Exemplo: usando

Pessoas(nome, endr, fone) e

Frequenta(pessoa, bar)

Pessoas JOIN Frequenta ON

nome = pessoa;

retorna todas as tuplas (n, e, f, p, b) tais que a pessoa n frequenta o bar b .

Junção Theta : Exemplo

Empr (RG , Nome , Sexo , Salario , Cod)

2232	João	M	700.00	1
4050	Pedro	M	700.00	1
2245	Ana	F	1100.00	2
8960	Roberto	M	1800.00	3
7865	Claudia	F	1100.00	2
0983	Helena	F	600.00	NULL
2348	Jose	M	700.00	3

Dep (Cod , Dpto , RGChefe)

1	RH	2232
2	Pesquisa	7865
3	Manutenção	8960

R (RG , Nome , Sexo , Salario , Empr. Cod , Dep. Cod , Dpto , RGChefe)

2232	João	M	700.00	1	1	RH	2232
4050	Pedro	M	700.00	1	1	RH	2232
2245	Ana	F	1100.00	2	2	Pesquisa	7865
8960	Roberto	M	1800.00	3	3	Manutenção	8960
7865	Claudia	F	1100.00	2	2	Pesquisa	7865
2348	Jose	M	700.00	3	3	Manutenção	8960

SELECT *
FROM Empr , Dep
WHERE Empr.Cod = Dep.Cod

OU

SELECT *
FROM Empr **JOIN** Dep **ON**
 Empr.Cod = Dep.Cod

Junções externas (outer joins)

- R OUTER JOIN S é a base de uma expressão de junção externa. Pode ser modificada com:
 1. NATURAL opcional na frente de OUTER.
 2. ON <condição> opcional depois do JOIN.
 3. LEFT, RIGHT, ou FULL opcional antes do OUTER.
 - LEFT = só tuplas não relacionadas de R .
 - RIGHT = só tuplas não relacionadas de S .
 - FULL = ambas.

Exemplo: Junção externa

R =

(A	B)
1	2
4	5

S =

(B	C)
2	3
6	7

(1,2) realiza junção com (2,3), mas as outras duas tuplas não tem como se relacionar

A	B	C
1	2	3
4	5	NULL
NULL	6	7

SELECT *
FROM R **full outer natural join** S

Exemplo: Junção theta esquerda

R = (

A	B
1	2
4	5

S = (

B	C
2	3
6	7

A	R.B	S.B	C
1	2	2	3
4	5	NULL	NULL

SELECT *
FROM R **left outer join** S **on** R.B = S.B

Exemplo: Junção theta (*Outerjoin*)

Empregados (RG , Nome , Sexo , Salario , Cod)

2232	João	M	700.00	1
4050	Pedro	M	700.00	1
2245	Ana	F	1100.00	2
8960	Roberto	M	1800.00	3
7865	Claudia	F	1100.00	2
0983	Helena	F	600.00	NULL
2348	Jose	M	700.00	3

Departamentos (Cod , Dpto , RGChefe)

1	RH	2232
2	Pesquisa	7865
3	Manutenção	8960
4	Transporte	NULL

SELECT *
FROM Empregados **as** E **outer join** Departamentos **as** D
on E.Cod = D.cod

R (RG , Nome , Sexo , Salario , E.Cod , D.Cod , Dpto , RGChefe)

2232	João	M	700.00	1	1	RH	2232
4050	Pedro	M	700.00	1	1	RH	2232
2245	Ana	F	1100.00	2	1	Pesquisa	7865
8960	Roberto	M	1800.00	3	3	Manutenção	8960
7865	Claudia	F	1100.00	2	2	Pesquisa	7865
2348	Jose	M	700.00	3	3	Manutenção	8960
0983	Helena	F	600.00	NULL	NULL	NULL	NULL
NULL	NULL	NULL	NULL	NULL	4	Transporte	NULL

Exemplo: Junção theta esquerda (*Left Outerjoin*)

Empregados (RG , Nome , Sexo , Salario , Cod)

2232	João	M	700.00	1
4050	Pedro	M	700.00	1
2245	Ana	F	1100.00	2
8960	Roberto	M	1800.00	3
7865	Claudia	F	1100.00	2
0983	Helena	F	600.00	NULL
2348	Jose	M	700.00	3

Departamentos (Cod , Dpto , RGChefe)

1	RH	2232
2	Pesquisa	7865
3	Manutenção	8960
4	Transporte	NULL

SELECT *
FROM Empregados **as** E **left outer join** Departamentos **as** D
on E.Cod = D.cod

R (RG ,	Nome ,	Sexo ,	Salario ,	E.Cod ,	D.Cod	Dpto ,	RGChefe)
2232	João	M	700.00	1	1	RH	2232
4050	Pedro	M	700.00	1	1	RH	2232
2245	Ana	F	1100.00	2	1	Pesquisa	7865
8960	Roberto	M	1800.00	3	3	Manutenção	8960
7865	Claudia	F	1100.00	2	2	Pesquisa	7865
2348	Jose	M	700.00	3	3	Manutenção	8960
0983	Helena	F	600,00	NULL	NULL	NULL	NULL

Agregação

- SUM, AVG, COUNT, MIN, and MAX podem ser aplicados a colunas no SELECT para produzir agregações dessas colunas.
- COUNT(*) conta o número de tuplas.

Exemplo: Agregação

- A partir de `Vendem(bar, cerveja, preço)` calcular o preço médio da Skoll:

```
SELECT AVG(preço)
FROM Vendem
WHERE cerveja = 'Skoll';
```

Eliminar duplicados em uma Agregação

- Utilize DISTINCT como parte da agregação.
- Exemplo: Busque o número de preços diferentes existentes para a Skoll:

```
SELECT COUNT(DISTINCT preço)  
FROM Vendem  
WHERE cerveja = 'Skoll';
```


NULL é ignorado na Agregação

- NULL nunca contribui para uma soma, media ou contagem e nunca pode ser mínimo ou máximo.
- Mas se não existem valores não NULL em uma coluna, então o resultado da agregação é NULL.

Exemplo: Efeito dos NULL's


```
SELECT count(*)  
FROM Vendem  
WHERE cerveja = 'Skoll';
```

Número de bares que vendem Skoll



```
SELECT count(preço)  
FROM Vendem  
WHERE cerveja = 'Skoll';
```

Número de bares que vendem Skoll a um preço definido (not NULL).



Agrupamento

- Podemos associar a uma cláusula SELECT-FROM-WHERE um GROUP BY e uma lista de atributos
- A relação resultante do SELECT-FROM-WHERE é agrupada de acordo com os valores desses atributos e qualquer agregação é aplicada somente sobre cada grupo.

Aplicação de GROUP BY

- Agrupar R de acordo com todos os atributos de agrupamento em L .
 - Isto é: formar um grupo para cada combinação de valores diferentes em R dos atributos em L .
- Em cada grupo, calcular $AGR(A)$ para cada agregação na lista L .
- O resultado tem uma tupla para cada grupo:
 1. Os atributos de agrupamento, e
 2. As agregações dos grupos.

Exemplo: Agrupamento / Agregação

Empregados (

<u>RG,</u>	Nome ,	Sexo ,	Salario ,	Codigo
2232	João	M	700.00	1
4050	Pedro	M	700.00	1
2245	Ana	F	1100.00	2
8960	Roberto	M	1800.00	3
7865	Claudia	F	1100.00	2
0983	Helena	F	600.00	NULL
2348	Jose	M	700.0	3

)



SELECT Sexo, COUNT(*), MAX(SALARIO)
FROM Empregados
GROUP BY Sexo

R (

Sexo ,	Count () ,	MAX(Salario)
M	4	1800,00
F	3	1100,00

)

Exemplo: Agrupamento / Agregação

Empregados (<u>RG,</u>	Nome ,	Sexo ,	Salario ,	Codigo)
	2232	João	M	700.00	1	
	4050	Pedro	M	700.00	1	
	2245	Ana	F	1100.00	2	
	8960	Roberto	M	1800.00	3	
	7865	Claudia	F	1100.00	2	
	0983	Helena	F	600.00	NULL	
	2348	Jose	M	700.0	3	

← M 700
 ← F 1100
 ← M 1800
 ← F 600

SELECT Sexo, Salario, COUNT(*)
FROM Empregados
GROUP BY Sexo, Salario

R (Sexo ,	Salario ,	Count ())
	M	700,00	3	
	M	1800,00	1	
	F	1100,00	2	
	F	600,00	1	

Exemplo: Agrupamento

- Usando `Vendem(bar, cerveja, preço)`
- Determinar o preço médio de cada cerveja.

```
SELECT cerveja, AVG(preço)
FROM Vendem
GROUP BY cerveja;
```

Exemplo: Agrupamento

- A partir de `Vendem(bar, cerveja, preço)` e `Frequenta(pessoa, bar)`, determinar para cada pessoa o preço médio da Skoll dos bares que ele frequenta:

```
SELECT pessoa, AVG(preço)
```

```
FROM Frequenta join Vendem on  
    Frequenta.bar = Vendem.bar  
WHERE cerveja = 'Skoll'  
GROUP BY pessoa;
```

Primeiro calcule o preço pessoa-bar para Skoll. Depois agrupar por pessoa.

Restrições na lista do SELECT com as agregações

- Se alguma agregação é usada, cada elemento da cláusula SELECT deve ser:
 1. Agregação, ou
 2. Um atributo na lista GROUP BY.

Exemplo de consulta ilegal

- Pode-se pensar que o nome do bar que vende a Skoll mais barata é determinada:

```
SELECT bar, MIN(preço)
FROM Vendem
WHERE cerveja = 'Skoll';
```

- Mas esta consulta é ilegal.

Cláusula HAVING

- HAVING <condição> pode ser usada junto com uma cláusula GROUP BY.
- A condição é aplicada a cada grupo, e os grupos que não satisfazem a condição são eliminados.


Exemplo: HAVING

- A partir de `Vendem(bar, cerveja, preço)` e `Cervejas(nome, fabr)` busque o preço médio das cervejas que são servidas em pelo menos três bares e que são fabricadas pela Ambev.

Solução

```
SELECT cerveja, AVG(preço)  
FROM Vendem  
GROUP BY cerveja
```


Grupos de cervejas com
pelo menos 3 bares not-
null e agrupadas pelo
fabricante Ambev



```
HAVING COUNT(bar) >= 3 AND
```

```
cerveja IN (SELECT nome  
FROM Cervejas  
WHERE fabr = 'Ambev');
```

Cervejas
fabricadas
pela Ambev.



Requisitos nas condições HAVING

- As condições devem referenciar qualquer relação ou variável de tupla na cláusula FROM.
- Elas podem referenciar atributos daquelas relações que fazem sentido dentro do grupo criado
 1. Um atributo de agrupamento, ou
 2. Agregado.

Sub-consultas

- Um comando `SELECT-FROM-WHERE` entre parênteses (*sub-consulta*) pode ser usado como valor em vários lugares, incluindo as cláusulas `FROM` e `WHERE`.
- Exemplo: no lugar de uma relação na cláusula `FROM`, podemos colocar outra consulta e consultar seu resultado.
 - Neste caso, o melhor é usar variáveis de tupla para nomear as tuplas do resultado.

Sub-consultas que retornam uma tupla

- Se uma sub-consulta com certeza produz uma única tupla, então pode ser utilizada como valor.
 - Geralmente a tupla tem um componente.
 - Gera erro de execução se não existe a tupla ou existe mais de uma tupla.

Exemplo

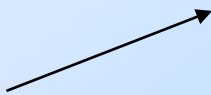
- Usando `Vendem(bar, cerveja, preço)` determine os bares que servem Skoll pelo mesmo preço que o Vitoria vende a Brahma.
- Duas consultas precisam ser realizadas:
 1. Buscar o preço da Brahma no Vitoria.
 2. Buscar os bares que servem Skoll com aquele preço.

Solução Consulta + Sub-consulta

```
SELECT bar  
FROM Vendem  
WHERE cerveja = 'Skoll' AND
```

```
    preço = (SELECT preço  
             FROM Vendem  
             WHERE bar = 'Vitoria'  
             AND cerveja = 'Brahma');
```

O preço da
Brahma no
Vitoria
Retorna uma
tupla



Sub-consultas que retornam conjuntos de tuplas

- Uma sub-consulta pode interpretada como um conjunto.
- Sobre este conjunto podem ser aplicados operadores específicos
- $\langle \text{tupla} \rangle \text{ IN } \langle \text{relação} \rangle$ é verdadeiro se e somente se $\langle \text{tupla} \rangle$ faz parte da $\langle \text{relação} \rangle$
- $\langle \text{tupla} \rangle \text{ NOT IN } \langle \text{relação} \rangle$ significa o contrario.

Exemplo Operador IN

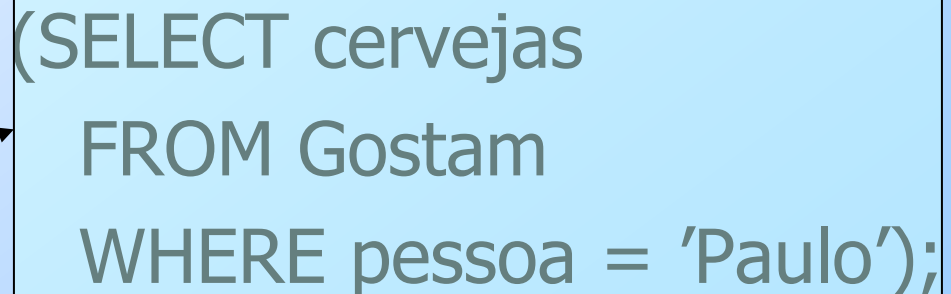
- A partir de `Cervejas(nome, fabr)` e `Gostam(pessoa, cerveja)`, buscar o nome e o fabricante de cada cerveja que o Paulo gosta.

`SELECT *`

`FROM Cervejas`

`WHERE nome IN`

Conjunto dos
nomes das
Cervejas das
que Paulo
gosta



```
(SELECT cervejas  
FROM Gostam  
WHERE pessoa = 'Paulo');
```

Outro Exemplo

A partir de **Bares(nome, endr, alvara)**
e **Vendem(bar, cerveja, preço)** buscar os endereços dos
bares nos quais nenhuma cerveja é vendida por mais
de R\$3.0

SELECT endr

FROM Bares


WHERE nome NOT IN (

SELECT bar

FROM Vendem

WHERE preço > 3.0;

Conjunto dos
nomes dos
Bares que vendem alguma
cerveja com preço maior que 3.0



Operador EXISTS

- EXISTS(<relação>) é verdadeiro se e somente se <relação> é não vazio.
- Exemplo: Usando Cervejas(nome, fabr) , buscar as cervejas que são as únicas feitas pelo seu fabricante.

Exemplo com EXISTS

```
SELECT nome  
FROM Cervejas b1  
WHERE NOT EXISTS (
```

Regra de alcance: fabr refere-se ao FROM mais próximo que tenha uma relação com esse atributo

Conjunto de cervejas do mesmo fabricante que b1, mas diferentes de b1

```
SELECT *  
FROM Cervejas  
WHERE fabr = b1.fabr AND  
       nome <> b1.nome )
```

Operador ANY

- $x = \text{ANY}(\text{ <relação> })$ é verdadeira se x é igual a pelo menos uma tupla em <relação>
- De forma similar, $=$ pode ser substituído por qualquer operador de comparação
- Exemplo: $x \geq \text{ANY}(\text{ <relação> })$ significa que x não é a menor tupla em <relação> .
 - Note que as tuplas em relação devem ter uma única componente.

Operador ALL

- De forma similar, $x <> \text{ALL} (\langle \text{relação} \rangle)$ é verdadeira se e somente se para toda tupla t em $\langle \text{relação} \rangle$, x não é igual a t .
 - Isto é, x não pertence a $\langle \text{relação} \rangle$.
- O operador $<>$ pode ser substituído por qualquer operador de comparação.
- Exemplo: $x \geq \text{ALL} (\langle \text{relação} \rangle)$ significa que não existe tupla maior que x em $\langle \text{relação} \rangle$.

Exemplo

- A partir de **Vendem(bar, cerveja, preço)** buscar a cerveja(s) vendida pelo maior preço.

```
SELECT cerveja  
FROM Vendem
```

```
WHERE preço >= ALL(  
    SELECT preço  
    FROM Vendem);
```

preço do Vendem
externo não pode ser
menor que qualquer
preço.