

# SQL

Bancos de Dados I

Altigran Soares da Silva

IComp/UFAM

Adaptado do Material do Professor Jeffrey Ullman

## Por que SQL?

- SQL é uma linguagem declarativa
  - Diz “o que fazer” em vez de “como fazer”
  - Evita que seja necessário conhecer de detalhes de como manipular os dados.
- SGBD procura a melhor maneira de executar a consulta.
  - “Otimização de Consulta.”

## Introdução

- Originalmente proposta para o System R desenvolvido nos laboratórios da IBM na década de 70
  - SEQUEL (Structured English QUery Language)
- Objeto de um esforço de padronização coordenado pelo ANSI/ISO:
  - SQL1 (SQL-86)
  - SQL2 (SQL-92)
  - SQL3 (SQL:1999)

## Setenças Select-From-Where

**SELECT** atributos desejados

**FROM** uma ou mais tabelas

**WHERE** condição sobre as tuplas da(s) tabela(s)

## Esquema de Exemplo

- Todas nossas consultas SQL serão baseadas no seguinte Esquema.
  - Sublinhado indica chave.

Cervejas(nome, fabr)

Bares(nome, ender, cnpj)

Cientes(nome, ender, fone)

Gosta(cliente, cerveja)

Vendas(bar, cerveja, preco)

Frequenta(cliente, bar)

5

## Resultado da Consulta

nome
Bud
Bud Lite
Michelob
...

A resposta é uma Relação com um atributo simples: **nome** e tuplas com o "**nome**" de cada cerveja (**cerveja**) do **Anheuser-Busch**, tal como Bud.

7

## Exemplo

- Usando Cervejas(nome, fabr), quais cervejas são frabriacas por **Anheuser-Busch**?

```
SELECT nome
FROM Cervejas
WHERE fabr = 'Anheuser-Busch';
```

6

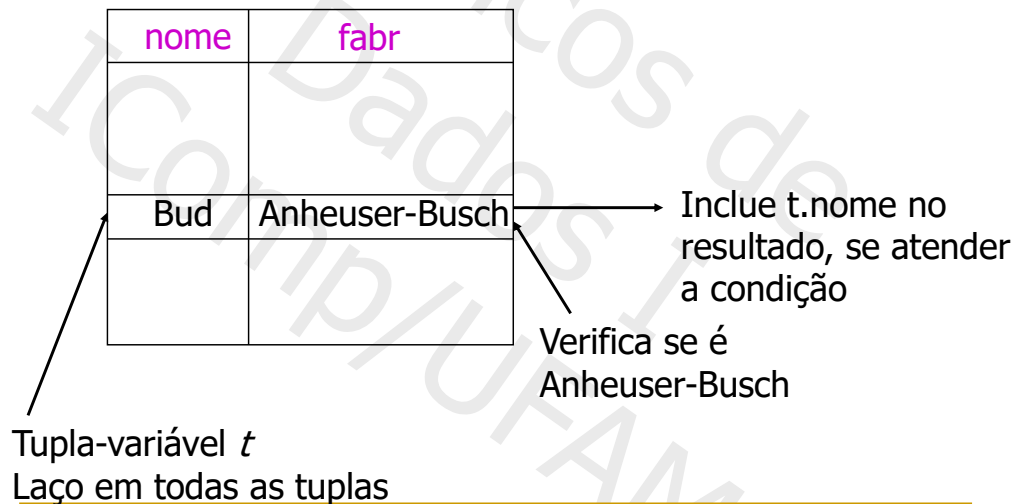
## Definindo uma consulta usando uma única relação

- Inicie com a Relação na cláusula **FROM**.
- Aplique a seleção indicada pela cláusula **WHERE**.
- Aplique a projeção estendida indicada pela cláusula **SELECT**.

```
SELECT nome
FROM Cervejas
WHERE fabr =
'Anheuser-Busch';
```

8

## Semântica Operacional



9

### \* Nas cláusula SELECT

- Quando um “\*” for utilizado na cláusula SELECT para uma relação definida na cláusula FROM, significa que “**todos**” atributos desta relação serão utilizados.

- Exemplo:** Usando Cervejas(nome, fabr):

```
SELECT *  
FROM Cervejas  
WHERE fabr = 'Anheuser-Busch';
```

11

## Semântica Operacional (Geral)

- Imagine uma **variável tupla** visitando cada tupla da **Relação** mencionada em **FROM**.
- Verifica se a tupla “corrente” satisfaz a cláusula **WHERE**.
- Se sim, computa os atributos ou expressões da cláusula SELECT usando os componentes da tupla.

```
SELECT nome  
FROM Cervejas  
WHERE fabr = 'Anheuser-Busch';
```

10

### Resultado da Consulta:

```
SELECT *  
FROM Cervejas  
WHERE fabr = 'Anheuser-Busch';
```

nome	fabr
Bud	Anheuser-Busch
Bud Lite	Anheuser-Busch
Michelob	Anheuser-Busch
...	...

Agora, o resultado tem todos os atributos da Relação 'Cervejas'.

12

## Renomeando Atributos

- Caso deseje um resultado com nome de atributos diferentes, use “AS <novo nome>” para renomear um atributo.

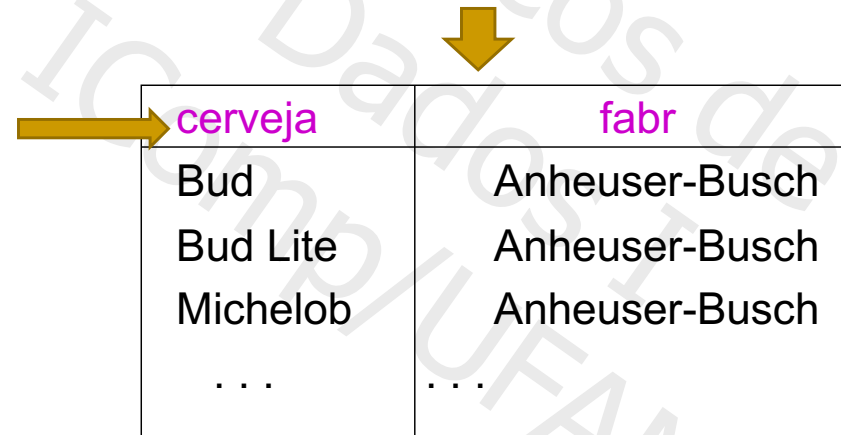
- **Exemplo:** Usando **Cervejas(nome, fabr):**

```
SELECT nome AS cerveja, fabr
FROM Cervejas
WHERE fabr = 'Anheuser-Busch'
```

13

## Resultado da Consulta:

```
SELECT nome AS cerveja, fabr
FROM Cervejas
WHERE fabr = 'Anheuser-Busch'
```



cerveja	fabr
Bud	Anheuser-Busch
Bud Lite	Anheuser-Busch
Michelob	Anheuser-Busch
...	...

14

## Expressões em cláusula SELECT

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

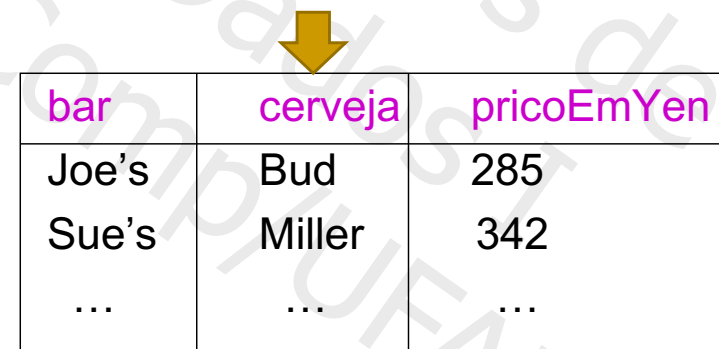
- **Exemplo:** Usando **Vendas(bar, cerveja, preco):**

```
SELECT bar, cerveja,
       preco*114 AS precoEmYen
FROM Vendas;
```

15

## Resultado da Consulta

```
SELECT bar, cerveja, preco*114 AS precoEmYen
FROM Vendas;
```



bar	cerveja	pricoEmYen
Joe's	Bud	285
Sue's	Miller	342
...	...	...

16

## Exemplo: Constantes como Expressões

- Usando **Gosta(cliente, cerveja)**:

```
SELECT cliente,  
       'curte Bud' AS QuemGostadeBud  
FROM Gosta  
WHERE cerveja = 'Bud';
```

17

## Resultado da Consulta

```
SELECT cliente, 'curte Bud' AS QuemGostadeBud  
FROM Gosta  
WHERE cerveja = 'Bud';
```



cliente	QuemGostadeBud
Sally	curte Bud
Fred	curte Bud
...	...

18

## Condições Complexa em cláusula WHERE

- Operadores Booleanos AND, OR, NOT.
- Comparações =, <>, <, >, <=, >=.
- E muitos outros operadores que produzem um resultado booleano.

19

## Exemplo: Condição Complexa

- Usando **Vendas(bar, cerveja, preco)**, procure o preço da cerveja Bud em Joe's Bar:

```
SELECT preco  
FROM Vendas  
WHERE bar = 'Joe's Bar' AND  
       cerveja = 'Bud';
```

20

## Padrões

- Uma condição pode comparar uma string à um padrão:
  - <Atributo> LIKE <padrão>
  - <Atributo> NOT LIKE <padrão>
- **Padrão**
  - é uma string anotada com caracteres coringa
    - % = “qualquer string”;
    - \_ = “qualquer character.”

21

## Exemplo: LIKE

- Usando **Clientes(nome, ender, fone)** procure os *clientes* com fones com 555:

```
SELECT nome
FROM Clientes
WHERE fone LIKE '%555- _ _ _ _';
```

22

## Valores NULL

- Tuplas em relações podem ter valores como NULL para um ou mais atributos
- Dois casos comuns:
  - **Faltando valor** : e.x: Nós sabemos que o Zeca's Bar tem um endereço, mas não sabemos qual.
  - **Não aplicável**: e.x: O valor do atributo **cônjuge** para uma pessoa que não é casada.

23

## Comparando NULL com Valores

- A lógica das condições em SQL aceita 3 valores lógicos: TRUE, FALSE, UNKNOWN.
- Comparando qualquer valor (incluindo o próprio NULL) com NULL retorna UNKNOWN.
- Se a cláusula WHERE é verdadeira (not FALSE ou UNKNOWN), a resposta é composta de tupla(s).

24

## Consultas em Múltiplas Relações

- Consultas interessantes frequentemente combinam dados de mais de uma relação.
- Podemos usar diversas relações em uma consulta, listando-as na cláusula FROM.
- Atributos distintos com mesmo nome usando “<relação>.<atributo>” .

25

## Exemplo: Junção de Duas Relações

- Usando as relações **Gosta(cliente, cerveja)** e **Frequenta(cliente, bar)**, quais os apreciadores de cerveja que frequentam o Zeca's Bar.

```
SELECT Gosta.cliente
FROM Gosta, Frequenta
WHERE bar = 'Zeca's Bar' AND
Frequenta.cliente=Gosta.cliente;
```

26

## Semântica Formal

- Parecido com consultas por uma relação simples:
  1. Inicia com o produto cartesiano de todas as relações na cláusula **FROM**.
  2. Aplica a condição de seleção descritas na cláusula **WHERE**
  3. Projeta na lista de atributos e expressões da cláusula **SELECT**.

27

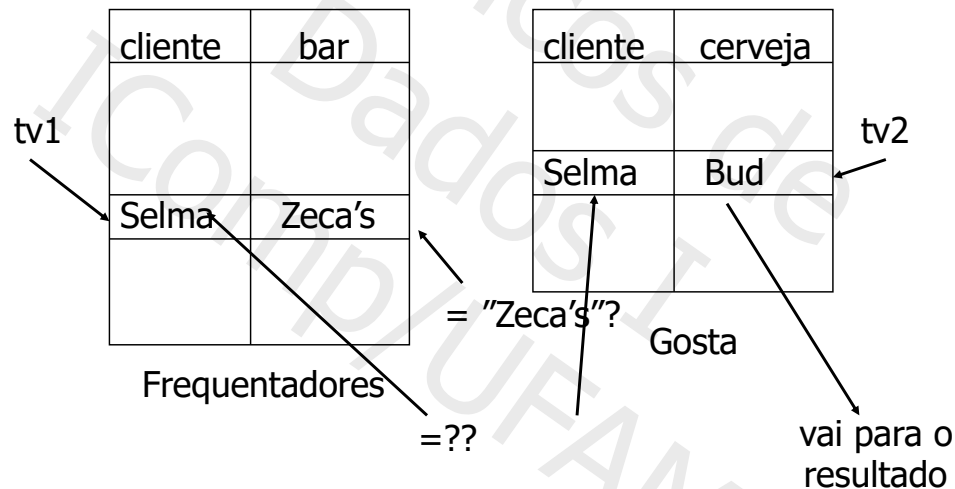
## Semântica Operacional

- Uma tupla-variável para cada relação na cláusula FROM .
  - Estas tuplas-variáveis consultam cada combinação de tuplas, uma de cada relação.
- Se a tupla-variável satisfaz a cláusula WHERE , então envie esta tuplas para a cláusula SELECT .

28



## Exemplo



29

## Tupla-Variáveis Explícitas

- Algumas vezes, uma consulta precisa usar duas cópias da mesma relação.
- Cópias distintas seguem o nome da relação para o nome de uma tupla-variável, na cláusula FROM.
- Isto é uma opção para renomear relações, até mesmo quando não é essencial.

30

## Exemplo: Self-Join

- Da relação **Cervejas(nome, fabr)**, procure os pares de cervejas do mesmo fabricante.
  - Não liste pares como (Bud, Bud).
  - List pares em ordem alfabetica, e.g. (Bud, Miller) não (Miller, Bud).

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

31

## Sub-Consultas

- Colocada entre parênteses uma sentença SELECT-FROM-WHERE (**sub-consulta**) pode ser usado como valor em vários pontos, incluindo em cláusulas FROM e WHERE.
- **Exemplo:**
  - No lugar de uma relação na cláusula FROM, podemos usar uma sub-consulta e verificar seu resultado.
  - Deve-se usar uma tupla-variável para o nome das tuplas do resultado.

32



## Exemplo: Sub-consulta em FROM

- Procure as cervejas curtidas por pelo menos uma pessoa, que frequenta o Zeca's Bar.

```
SELECT cerveja
FROM Gosta, (SELECT cliente
             FROM Frequenta
             WHERE bar='Zeca's Bar') CZ
WHERE Gosta.cliente = CZ.cliente;
```

Clientes que frequentam o Zeca's Bar

33

## Sub-consultas que retornam uma tupla

- Se uma sub-consulta garantidamente produz uma única tupla, então pode ser usada como um valor.
  - Normalmente, a tupla tem um só atributo.
  - Um erro em tempo de execução ocorre, caso não exista tupla ou mais de uma tupla for gerada

34

## Exemplo: Sub-consulta com Tuplas simples

- Usando **Vendas(bar, cerveja, preco)**, procure bares que servem **Miller** pelo mesmo preço que o Zeca's vende **Bud**.
- Duas consultas:
  - Procure o preço cobrado no Zeca's para Bud.
  - Procure os bares que servem Miller com este preço

35

## Consulta+ Subconsulta Solução

```
SELECT bar
FROM Vendas
WHERE cerveja = 'Miller' AND
preco = (SELECT preco
        FROM Vendas
        WHERE bar = 'Zeca's Bar'
        AND cerveja = 'Bud');
```

36

## O operador IN

- <tupla> IN (<subconsulta>) é verdadeira, se e somente se a tupla é um membro da relação produzida por uma subconsulta.
  - Oposto: <tupla> NOT IN (<subconsulta>).
- Expressões IN podem aparecer em cláusulas WHERE .

37

## Exemplo: IN

- Usando Cervejas(nome, fabr) e Gosta(cliente, cerveja), procure o nome e fabricante de cada cerveja que Fred gosta.

```
SELECT *  
FROM Cervejas  
WHERE nome IN (SELECT cerveja  
FROM Gosta  
WHERE cliente = 'Fred');
```

Conjunto de cervejas  
que Fred gosta

38

## Join vs. IN

```
SELECT a  
FROM R, S  
WHERE R.b = S.b;
```

```
SELECT a  
FROM R  
WHERE b IN (SELECT b FROM S);
```

39

## IN é um predicado sobre tuplas de R

```
SELECT a  
FROM R  
WHERE b IN (SELECT b FROM S);
```

Duas Vezes

Um laço, sobre as tuplas da relação R

a	b
1	2
3	4

R

b	c
2	5
2	6

S

(1,2) satisfaz a condição;

40

## Join: emparelha tuplas de R, S

```
SELECT a
FROM R, S
WHERE R.b = S.b;
```

Duplo laço, sobre as  
tuplas de R e S

a	b
1	2
3	4

R

b	c
2	5
2	6

S

(1,2) com (2,5)  
(1,2) com (2,6)  
satisfazem a  
condição

41

## O operador Exists

- EXISTS(<subconsulta>) é verdadeiro se e somente se o resultado da subconsulta não é vazio.
- Exemplo: Usando Cervejas(nome, fabr) , procure as cervejas que são únicas por fabricante.

42

## Exemplo: EXISTS

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

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

```
);
```

Cervejas diferentes  
com mesmo  
fabricante que em b1

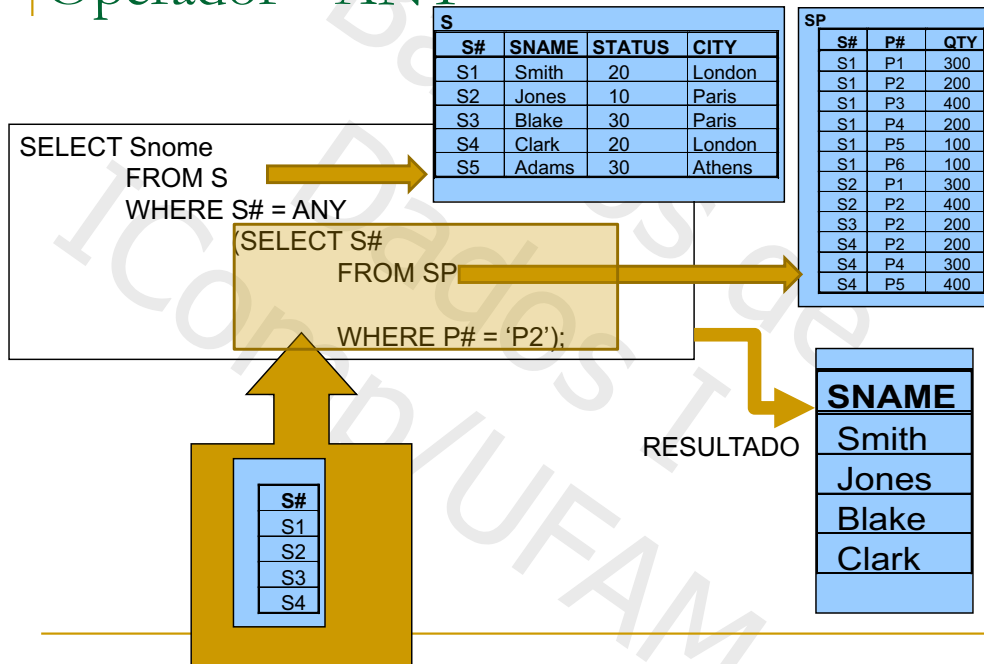
43

## Operador ANY

- <ANY: encontrar tuplas que tenham valores menores que o maior valor retornado por uma sub-consulta.
- >ANY: encontrar tuplas que tenham valores maiores que o menor valor retornado de uma sub-consulta
- =ANY: funciona da mesma maneira que o operador IN.

44

## Operador =ANY



45

## Operador <ANY

SELECT S#  
FROM S  
WHERE STATUS < ANY  
(SELECT STATUS  
FROM S);

S#	SNAME	STATUS	CITY
S1	Smith	20	London
S2	Jones	10	Paris
S3	Blake	30	Paris
S4	Clark	20	London
S5	Adams	30	Athens

S#
S1
S2
S4

46

## Operador ALL

- $x \neq \text{ALL}(\text{sub-consulta})$  é verdadeiro se e somente se para cada tupla  $t$  na relação,  $x$  não é igual em  $t$ .
- Isto é,  $x$  não está no resultado da sub-consulta.
- Ao invés de  $\neq$  pode ser qualquer operador de comparação.
- **Exemplo:**
  - $x \geq \text{ALL}(\text{subconsulta})$
  - Nenhuma tupla é maior que  $x$  no resultado da subconsulta.

47

## Exemplo: ALL

- Usando **Vendas(bar, cerveja, preco)**, procure as cervejas vendidas com maior preço.

```
SELECT cerveja
FROM Vendas
WHERE preco >= ALL( SELECT preco
                     FROM Vendas
                     )
```

48

## União, Interseção e Diferença

- União, interseção e diferença de relações são expressas das seguintes formas, cada uma envolve subconsultas:

- ❑ (<subconsulta>) UNION (<subconsulta>)
- ❑ (<subconsulta>) INTERSECT (<subconsulta>)
- ❑ (<subconsulta>) EXCEPT (<subconsulta>)

49

## Exemplo: Interseção

- Usando **Gosta(cliente, cerveja)**, **Vendas(bar, cerveja, preco)**, e **Frequenta(cliente, bar)**, procure os consumidores e cervejas tal que:
  - ❑ Gostem de cerveja,
  - ❑ Frequentem pelo menos um bar que venda cerveja.

50

## Solução

(**SELECT \* FROM Gosta**)

Consumidores que frequentam um bar que venda cerveja.

INTERSECT

(**SELECT cliente, cerveja  
FROM Vendas, Frequenta  
WHERE Frequents.bar = Vendas.bar**);

51

## RELAÇÕES COMO BAGS

## Semântica de Bags em SQL

- Setenças SELECT-FROM-WHERE usa a semântica de bags
  - Seleção:
    - preserva o número de ocorrências
  - Projeção:
    - preserva o nr. de ocorrências (não elimina duplicados)
  - Produto Cartesiano, Junção:
    - não elimina duplicados

## Motivação: Eficiência

- Quando fazemos projeções em Álgebra Relacional, é mais fácil evitar a eliminação de tuplas duplicadas.
- Para interseção ou diferença, isto é mais eficiente para ordenação de relações.

## Tratamento de Tuplas Duplicadas

- SELECT DISTINCT ...
  - Força com que o resultado seja um conjunto
- ... UNION ALL ...
  - Força que o resultado seja um Bag (não elimina duplicados) de todas as tuplas

## Exemplo: DISTINCT

- Usando **Vendas(bar, cerveja, preco)**, procure todos os preços diferentes cobrados por cervejas:  

```
SELECT DISTINCT preco
FROM Vendas;
```
- Note que sem DISTINCT, cada preço poderia ser listado várias vezes, conforme o par bar/cerveja e seus respectivos preços.

## Junções

- A SQL provê diversas versões de junções.
- Estas expressões podem ser consultas independentes ou usadas no lugar das relações em uma cláusula FROM
- Semântica de Bags!

57

## Produto e Junção Natural

- Junção Natural:  
R NATURAL JOIN S;
- Produto:  
R CROSS JOIN S;
- Exemplo:  
Gosta NATURAL JOIN Vendas;

58

## Junção Theta

- R JOIN S ON <condição>
- Exemplo: usando Clientes(nome, ender) e Frequenta(cliente, bar):  
Clientes JOIN Frequenta ON  
nome = cliente;
- Dá todas as quadruplas  $(n, e, c, b)$  tal que cliente  $n$  mora no endereço  $e$  e frequenta o bar  $b$ .

59

## Outerjoins

- R OUTER JOIN S é o núcleo de uma expressão outerjoin. Podendo utilizar outros como:
  - Opcional NATURAL em frente do OUTER.
  - Opcional ON <condição> depois do JOIN.
  - Opcional LEFT, RIGTH, FULL
- Mutuamente exclusivas!

60



## Agregação

- SUM, AVG, COUNT, MIN, e MAX pode ser aplicada em uma coluna de uma cláusula SELECT para produzir a agregação de uma coluna.
- Também, COUNT(\*) conta o número de tuplas.

61

## Exemplo: Agregação

- Usando **Vendas(bar, cerveja, preco)**, procure a média de preço da cerveja *Bud*:

```
SELECT AVG(preco)
FROM Vendas
WHERE cerveja = 'Bud';
```

62

## Eliminando duplicados em agregações

- Usar DISTINCT dentro de uma agregação.
- **Exemplo:** Procure o número de diferentes preços cobrados pela cerveja Bud:

```
SELECT COUNT(DISTINCT preco)
FROM Vendas
WHERE cerveja = 'Bud';
```

63

## NULL é ignorado em agregações

- NULL nunca contribue para soma, média ou contagem, e nunca pode ser o mínimo ou máximo de uma coluna.
- Porém, se não existe nenhum valor non-NULL em uma coluna, então o resultado da agregação é NULL.
  - ▣ **Exceção:** COUNT de um conjunto vazio é 0.

64

## Exemplo: Efeito do NULL

```
SELECT count(*)  
FROM Vendas  
WHERE cerveja = 'Bud';
```

O número de Bares  
Que vendem Bud.

```
SELECT count(preco)  
FROM Vendas  
WHERE cerveja = 'Bud';
```

O número de bares que  
Vendem Bud por um preço  
Conhecido.

65

## Exemplo: Agrupando

- Usando `Vendas(bar, cerveja, preco)`, procure a média de preço para cada cerveja:

```
SELECT cerveja, AVG(preco)  
FROM Vendas  
GROUP BY cerveja;
```

cerveja	AVG(preco)
Bud	2.33
...	...

67

## Agrupamentos

- Adicionar um GROUP BY e uma lista de atributos na expressão SELECT-FROM-WHERE
- A relação resultante é agrupada de acordo com os valores dos atributos do GROUP BY, e qualquer agregação é aplicada somente em cada grupo.

66

## Exemplo: Agrupando

- Usando `Vendas(bar, cerveja, preco)` e `Frequenta(cliente, bar)`, procure por cada consumidor, a média de preço da Bud nos Bares que eles frequentam:

```
SELECT cliente, AVG(preco)  
FROM Frequenta, Vendas  
WHERE cerveja = 'Bud' AND  
Frequents.bar = Vendas.bar  
GROUP BY cliente;
```

Agrupa  
por cliente

68

## Restrição em lista SELECT com Agregação

- Se qualquer agregação é usada, então cada elemento da lista do SELECT deve ser também:
  1. Agregado, ou
  2. Um atributo na lista do GROUP BY .

69

## HAVING Cláusula

- HAVING <condição> pode seguir uma cláusula GROUP BY .
- Se a condição for aplicada para cada grupo, e os grupos que não satisfazem a condição são eliminados.

71

## Exemplo de consulta Ilegal

- Procurar o bar que vende Bud mais barato usando:

```
SELECT bar, MIN(preco)
FROM Vendas
WHERE cerveja = 'Bud';
```
- Esta consulta é ilegal em SQL.

## Exemplo: HAVING

- Usando `Vendas(bar, cerveja, preco)` e `Cervejas(nome, fabr)`, procure a média de preço destas cervejas que também são servidas em pelo menos três bares ou fabricados por Skol.

72

## Solução

```
SELECT cerveja, AVG(preco)
FROM Vendas
GROUP BY cerveja
HAVING COUNT(bar) >= 3 OR
```

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

Cervejas agrupadas por pelo  
Menos 3 non-NULL bares e  
Também grupos de cervejas  
Fabricadas por skoll.

Cervejas  
Fabricadas por  
skoll.

73

## Modificação de Banco de Dados

- Um comando de *modificação* não retorna resultado (como em uma consulta).
- Três tipos de modificações:
  1. *Insert* uma tupla ou tuplas.
  2. *Delete* uma tupla ou tuplas.
  3. *Update* o(s) valor(es) de uma tupla existente ou tuplas.

75

## Requisitos em condições HAVING

- Qualquer coisa vai em uma subconsulta.
- Fora das subconsultas, elas podem referenciar atributos somente se são :
  1. Um atributo que agrupa, ou
  2. Agregado(mesma condição como em cláusulas SELECT com agregação).

74

## Inserção

- Para inserir uma simples tupla:

```
INSERT INTO <relação>
VALUES ( <lista de valores> );
```
- *Exemplo*: adiciona para *Gosta(cliente, cerveja)* o fato que Sally gosta de cerveja Bud.

```
INSERT INTO Gosta
VALUES ('Sally', 'Bud');
```

76

## Especificando atributos em INSERT

- Pode-se usar uma lista de atributos
- Duas razões para fazer isto:
  1. Se esquecemos como os atributos estão ordenados para a relação.
  2. Não temos os valores para todos atributos, e queremos que o sistema preencha os atributos com NULL ou um valor padrão.

77

## Exemplo: Especificando Atributos

- Outra maneira de registrar o fato que Sally gosta da cerveja Bud em **Gosta(cliente, cerveja)**:

```
INSERT INTO Gosta (Cerveja, cliente)
VALUES ('Bud', 'Sally');
```

78

## Adicionando valores Padrões

- Numa sentença CREATE TABLE, podemos definir um atributo por DEFAULT e um valor.
- Quando uma inserção de uma tupla não tem valor para este atributo, o valor DEFAULT será usado.

79

## Exemplo: Valores Padrões (Default)

```
CREATE TABLE Clientes (
    nome CHAR(30) PRIMARY KEY,
    ender CHAR(50)
        DEFAULT 'Djalma Batista 123'
    fone CHAR(16)
);
```

80

## Exemplo: Valores Padrões (Default)

```
INSERT INTO Drinkers(nome)  
VALUES('Sally');
```

Resultando com uma tupla:

nome	ender	fone
Sally	Djalma Batista 123	NULL

81

## Exemplo: Inserindo uma subconsulta

- Usando **Frequenta(cliente, bar)**, como entrada para uma nova relação **Colegas(nome)** todos os potenciais colegas de Sally, clientes que frequentam pelo menos um bar que Sally também frequenta.

83

## Inserindo várias Tuplas

- Podemos inserir o resultado inteiro de uma consulta para uma relação, usando esta forma:

```
INSERT INTO <relação>  
( <subconsulta> );
```

82

## Solução

Outro cliente

```
INSERT INTO PotBuddies  
(SELECT d2.cliente  
FROM Frequenta d1, Frequenta d2  
WHERE d1.cliente = 'Sally' AND  
d2.cliente <> 'Sally' AND  
d1.bar = d2.bar  
);
```

Pares de tuplas cliente onde o Primeiro é Sally, o Segundo é outro diferente de Sally, e que frequentem o mesmo Bar.

84



## Remoção

- Remover tuplas que satisfazem uma condição de uma relação:

```
DELETE FROM <relação>  
WHERE <condição>;
```

85

## Exemplos: Remoção

- Remova de **Gosta(cliente, cerveja)** o fato que Sally gosta de cerveja Bud:

```
DELETE FROM Gosta  
WHERE cliente = 'Sally' AND  
cerveja = 'Bud';
```

86

## Exemplo: Remova todas as tuplas

- Torne a relação Gosta vazia:

```
DELETE FROM Gosta;
```

- Note que nenhuma cláusula WHERE é necessária.

87

## Exemplo: Remova algumas Tuplas

- Remova de **Cervejas(nome, fabr)** todas as cervejas tal que existe outra cerveja do mesmo fabricante.

```
DELETE FROM Cervejas b  
WHERE EXISTS (  
    SELECT nome FROM Cervejas  
    WHERE fabr = b.fabr AND  
           nome <> b.nome);
```

Cervejas com o  
Mesmo fabricante e  
Um nome diferente  
Do nome da cerveja  
Representado pela  
tupla.

88



## Semântica da Remoção --- (1)

- Supondo que Anheuser-Busch fabrica somente Bud e Bud Lite.
- Supondo que iremos consultar a tupla  $b$  para a primeira Bud.
- A consulta não é vazia, por causa da tupla Bud Lite, então nós removemos a Bud.
- Agora, quando  $b$  é a tupla para Bud Lite, Nós removemos esta tupla também?

89

## Semântica de Remoção --- (2)

- **Resposta:** Bud Lite também é removida!
- A razão é que a remoção tem dois estágios:
  1. Marca todas as tuplas para que a condição em WHERE seja satisfeita.
  2. Remove as tuplas marcadas.

90

## Atualização (Updates)

- Para alterar específicos atributos em específicas tuplas de uma relação:  
UPDATE <relação>  
SET <lista de atributos>  
WHERE <condição sobre as tuplas>;

91

## Exemplo: Update

- Altere o número de telefone de Fred para 555-1212:

```
UPDATE Clientes  
SET fone = '555-1212'  
WHERE nome = 'Fred';
```

92

## Exemplo: Atualiza diversas tuplas

- Faça \$4 o máximo preço para cerveja:

```
UPDATE Vendas  
SET preco = 4.00  
WHERE preco > 4.00;
```