Gerenciamento de Dados e Informação

SQL

Fernando Fonseca
Ana Carolina
Robson Fidalgo

Introdução

- SQL - Structured Query Language
  Linguagem de Consulta Estruturada
  - Apesar do QUERY no nome, não é apenas de consulta, permitindo definição (DDL) e manipulação (DML) de dados
- Fundamentada no modelo relacional (álgebra relacional)
  - Cada implementação de SQL pode possuir algumas adaptações para resolver certas particularidades do SGBD alvo

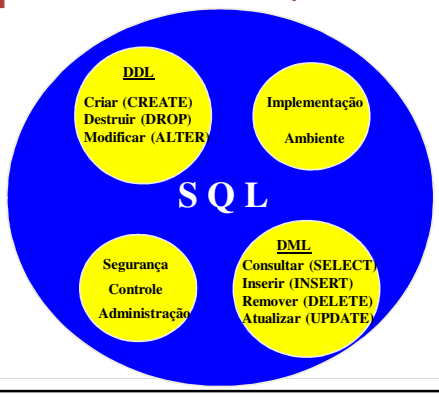
SQL - Origem/Histórico

- Primeira versão: SEQUEL, definida por Chamberlain em 1974 na IBM
- Em 1975 foi implementado o primeiro protótipo
- Revisada e ampliada entre 1976 e 1977 e teve seu nome alterado para SQL por razões jurídicas
- Em 1982, o American National Standard Institute tornou SQL padrão oficial de linguagem em ambiente relacional
- Utilizada tanto de forma interativa como incluída em linguagens hospedeiras

Enfoques de SQL

- Linguagem interativa de consulta (ad-hoc): usuários podem definir consultas independente de programas
- Linguagem de programação para acesso a banco de dados: comandos SQL embutidos em programas de aplicação
- Linguagem de administração de dados: o DBA pode utilizar SQL para realizar suas tarefas
- Linguagem cliente/servidor: os programas clientes usam comandos SQL para se comunicarem e compartilharem dados com o servidor
- Linguagem para banco de dados distribuídos: auxilia na distribuição de dados por vários nós e na comunicação de dados com outros sistemas
- Caminho de acesso a outros bancos de dados em diferentes máquinas: auxilia na conversão entre diferentes produtos em diferentes máquinas

Usos de SQL



SQL - Vantagens

- Independência de fabricante
- Portabilidade entre sistemas
- Redução de custos com treinamento
- Comandos em inglês
- Consulta interativa
- Múltiplas visões de dados
- Manipulação dinâmica dos dados

SQL - Desvantagens

- A padronização inibe a criatividade
- Está longe de ser uma linguagem relacional ideal
  - Algumas críticas
    - falta de ortogonalidade nas expressões
    - discordância com as linguagens hospedeiras
    - não dá suporte a alguns aspectos do modelo relacional

Esquema Relacional dos Exemplos

Empregado (Cad, Nome, Sexo, Salario, Num_Dep, Cad_Spv)
Num_Dep referencia Departamento (Numero),
Cad_Spv referencia Empregado(Cad)

Departamento (Numero, Nome, Cad_Ger, Data_Ini)
Cad_Ger referencia Empregado(Cad)

Locais (Num_dep, Nome_Loc)
Num_Dep referencia Departamento (Numero)

Projeto (Numero, Nome, Num_Dep, Local)
Num_Dep referencia Departamento (Numero)

Trabalha_em (Cad_Emp, Num_Pro, Horas)
Cad_Emp referencia Empregado(Cad),
Num_Pro referencia Projeto (Numero)

Dependente (Cad_emp, Nome, Data_nasc, Grau_P)
Cad_emp referencia Empregado(Cad)

Comandos SQL (Padrão ANSI)

- Criação, alteração e destruição de tabelas
- Inserção, modificação e remoção de dados
- Extração de dados de uma tabela (Consultas)
- Definição de visões
- Definição de privilégios de acesso

Criação de Tabelas

- Definição de nova tabela → CREATE TABLE

SQL

```
CREATE TABLE <nome da tabela>
            (<descrição dos atributos>
            <descrição das chaves>
            <descrição das restrições);
```

- Descrição dos atributos → <nome> <tipo>
- Tipos de dados (Oracle): varchar2, char, nvarchar2, nchar, number, number(n), number(m,n), binary_integer, binary_float, binary_double, date, timestamp, blob, clob, nclob

Criação de Tabelas

- Descrição das Chaves
  - A chave primária deve ser declarada como

SQL

```
CONSTRAINT nometabela_pkey
PRIMARY KEY (<atributos>)
```

  - Chave primária definida por auto-numeração
    - Chave inteira cujo valor é atribuído pelo sistema, sendo incrementado de 1 a cada nova inserção de uma tupla

Criação de Tabelas

- Chave primária por auto-numeração (Cont.)
  - No Oracle
    - Define-se uma sequência e esta será utilizada para gerar as chaves primárias

SQL

```
CREATE SEQUENCE <nome_seq>
INCREMENT BY 1 START WITH 1;
```

    - O tipo do atributo que será a chave primária deve ser INTEGER

Criação de Tabelas

- Chave primária por auto-numeração
  - No Oracle (Cont.)
    - Ao inserir dados na tabela, deve-se solicitar a criação do valor ao sistema no atributo chave utilizando o comando
      - <nome_seq>.NEXTVAL
  - Lista das chaves estrangeiras na forma

SQL

```
CONSTRAINT nometabela_fkey
FOREIGN KEY (<atributo>)
REFERENCES <outra_tabela> (<chave primária>)
```


Criação de Tabelas

- Descrição de Restrições
  - Salário não pode ser inferior ao mínimo

SQL

```
CONSTRAINT nometabela_check
CHECK (salario >= 880.00)
```

- Cada valor do atributo é único na relação, mesmo não sendo o atributo chave primária

SQL

```
CONSTRAINT nometabela_const
UNIQUE (nome)
```


Criação de Tabelas

- Exemplo 1: A entidade Departamento, considerando
  - Chave primária com auto numeração

Departamento (Numero, Nome, Cad_Ger, Data_Ini)
Cad_Ger referencia Empregado(Cad)

SQL

```
CREATE SEQUENCE Numero
INCREMENT BY 1 START WITH 1;
```


Criação de Tabelas

- Exemplo 1 (Cont.)

SQL

```
CREATE TABLE Departamento
  (Numero integer,
   Nome varchar2(15),
   Cad_Ger integer,
   Data_Ini date,
   CONSTRAINT Departamento_pkey
      PRIMARY KEY (Numero));
```

A chave estrangeira de Empregado (Cad_Ger) só pode ser criada depois que a relação Empregado for criada. Para tanto, posteriormente, altera-se a definição da relação Departamento.

Criação de Tabelas

- Exemplo 2: A entidade Empregado, considerando
  - O atributo Sexo deve ter os valores 'M' ou 'F'
  - O atributo Salario deve respeitar o mínimo nacional

Empregado (Cad, Nome, Sexo, Salario, Num_Dep, Cad_Spv)
Num_Dep referencia Departamento (Numero),
Cad_Spv referencia Empregado(Cad)

Criação de Tabelas

- Exemplo 2 (Cont.)

SQL

```
CREATE TABLE Empregado
  (Cad integer,
   Nome varchar2 (20),
   Sexo char,
   Salario number (10,2),
   Num_Dep number(1),
   Cad_Spv number,
   CONSTRAINT empregado_pkey PRIMARY KEY (Cad),
   CONSTRAINT empregado_fkey1 FOREIGN KEY
        (Num_Dep)   REFERENCES Departamento (Numero),
   CONSTRAINT empregado_fkey2 FOREIGN KEY
        (Cad_Spv)   REFERENCES Empregado (Cad),
CONSTRAINT Empregado_checkSal CHECK (salario >= 880.00),
CONSTRAINT Empregado_checkSex CHECK (sexo = 'M' OR
sexo = 'F') );
```

Criação de Tabelas

- Exemplo 3 (Considerando que a relação Projeto já foi criada)

Trabalha_em_ (Cad_Emp, Num_Proj, Horas)
Cad_Emp referencia Empregado(Cad),
Num_Proj referencia Projeto (Numero)

SQL

```
CREATE TABLE Trabalha_em
  (Cad_Emp integer,
   Num_Proj integer,
   Horas number (3,1) ,
   CONSTRAINT trabalha_em_pkey
      PRIMARY KEY (Cad_emp, Num_proj),
   CONSTRAINT trabalha_em_fkey1
      FOREIGN KEY (Cad_Emp) REFERENCES Empregado
       (Cad),
   CONSTRAINT trabalha_em_fkey2
      FOREIGN KEY (Num_Proj )REFERENCES Projeto
       (Numero));
```


Criação de Tabelas

- Criação de índices em uma tabela existente → CREATE INDEX
  - São estruturas que permitem agilizar a busca e ordenação de dados em tabelas

CREATE [UNIQUE] INDEX *<nome> ON <tabela> (<atributo$_1$>[, <atributo$_2$>…]);*

- Exemplo 3: Criar um índice sobre o atributo salario de Empregado

CREATE INDEX *indice_sal ON Empregado (salario);*

É usado automaticamente pelo sistema nas consultas realizadas por salario

Alteração de Tabelas

- Alterar definições de tabelas existentes → ALTER TABLE
  - Permite inserir/eliminar/modificar elementos da definição de uma tabela

ALTER TABLE <ação>;

Análoga ao Create / Drop

Alteração de Tabelas

- Exemplo 4: Acrescentar o atributo Diploma na tabela Empregado

SQL

```
ALTER TABLE EMPREGADO
  ADD (Diploma varchar2(20));
```

- Exemplo 5: Remover o atributo Diploma da tabela Empregado

SQL

```
ALTER TABLE EMPREGADO DROP (Diploma);
```


Alteração de Tabelas

- Exemplos 6: Acrescentar chave estrangeira de Empregado como gerente na tabela Departamento

SQL

```
ALTER TABLE DEPARTAMENTO
ADD ( CONSTRAINT  Departamento_fkey
  FOREIGN KEY (Cad_Ger) REFERENCES Empregado
       (Cad));
```

Necessário para a segunda relação, sempre que duas relações têm chave estrangeira uma da outra.

Remoção de Tabelas

- Eliminar uma tabela que foi previamente criada → DROP TABLE

DROP TABLE <tabela>;

- Exemplo 7: Remover a tabela Trabalha_em

SQL **DROP TABLE Trabalha_em;**

- Observações
  - Os dados são também excluídos
  - No caso do Oracle, para preservar as definições da relação (só os dados são eliminados) utilizar em vez de DROP

SQL **TRUNCATE TABLE Trabalha_em;**

Extração de Dados de uma Tabela (Consulta)

- Consultar dados em uma tabela → SELECT
  - Selecionando atributos (Projeção)

$\pi_{\text{<lista de atributos>}}$ (Nome-da-relação) *Álgebra Relacional*

SQL

SELECT <lista de atributos> FROM <tabela>;

- Exemplo 8: Listar nome e salário de todos os empregados

Extração de Dados de uma Tabela (Consulta)

Empregado

Cad	Nome	Sexo	Salario	Num_Dep	Cad_Spv
1	José	M	5000.00	1	2
2	Maria	F	800.00	2	NULL
3	João	M	3000.00	1	2

$\pi_{\text{nome, salario}}$ (Empregado) *Álgebra Relacional*

SQL

SELECT Nome, Salario FROM Empregado;

Nome	Salario
José	5000.00
Maria	800.00
João	3000.00

Extração de Dados de uma Tabela (Consulta)

- Selecionando todos os atributos

SQL

SELECT * FROM <tabela>;

- Exemplo 9: Selecionar todos os atributos dos empregados

Empregado

Cad	Nome	Sexo	Salario	Num_Dep	Cad_Spv
1	José	M	5000.00	1	2
2	Maria	F	800.00	2	NULL
3	João	M	3000.00	1	2

$\pi_{\text{Cad,Nome, Sexo, Salario, Num_Dep, Cad_Spv}}$ (Empregado)

Álgebra Relacional

Extração de Dados de uma Tabela (Consulta)

SQL

SELECT * FROM Empregado;

Cad	Nome	Sexo	Salario	Num_Dep	Cad_Spv
1	José	M	5000.00	1	2
2	Maria	F	800.00	2	NULL
3	João	M	3000.00	1	2

- Observação
  - Deve ser usado com cautela pois pode comprometer o desempenho do sistema

Extração de Dados de uma Tabela (Consulta)

- Selecionando tuplas da tabela → cláusula WHERE

SQL

SELECT <lista de atributos> FROM <tabela> WHERE <condição>;

- Onde condição

<nome atributo> <operador> <valor>

Relacionais			
<> ou !=	Diferente	=	Igual a
>	Maior que	>=	Maior ou igual a
<	Menor que	<=	Menor ou igual a

Lógicos	
AND	E
OR	Ou
NOT	Não

Uma constante, variável ou consulta aninhada

Extração de Dados de uma Tabela (Consulta)

- Exemplo 10: Listar todos os dados dos empregados do departamento 1

Empregado

Cad	Nome	Sexo	Salario	Num_Dep	Cad_Spv
1	José	M	5000.00	1	2
2	Maria	F	800.00	2	NULL
3	João	M	3000.00	1	2

$\sigma_{\text{Num_Dep = 1}}$ (Empregado) *Álgebra Relacional*

Extração de Dados de uma Tabela (Consulta)

SQL

```
SELECT * FROM Empregado WHERE Num_Dep = 1;
```

Cad	Nome	Sexo	Salario	Num_Dep	Cad_Spv
1	José	M	5000.00	1	2
3	João	M	3000.00	1	2

Extração de Dados de uma Tabela (Consulta)

- Exemplo 11: Listar nome e sexo dos empregados do departamento 1 com salário > R$ 4.000,00

Empregado

Cad	Nome	Sexo	Salario	Num_Dep	Cad_Spv
1	José	M	5000.00	1	2
2	Maria	F	800.00	2	NULL
3	João	M	3000.00	1	2

Álgebra Relacional

$\pi_{nome, sexo}$ ($\sigma_{Num\_Dep = 1 \text{ AND } salario > 4000.00}$ (Empregado))

Extração de Dados de uma Tabela (Consulta)

SQL

```
SELECT Nome, Sexo FROM Empregado
 WHERE Num_Dep = 1 AND Salario > 4000.00;
```

Nome	Sexo
José	M

- Consulta para o usuário fornecer valores para o SELECT só na hora da execução
  - Colocar parâmetro na forma &<variável>

SQL

```
SELECT <lista de atributos> FROM <tabela>
WHERE <atributo> = &<variável>;
```


Extração de Dados de uma Tabela (Consulta)

- Exemplo 12: Listar nome e salário dos empregados do departamento com um dado código

Empregado

Cad	Nome	Sexo	Salario	Num_Dep	Cad_Spv
1	José	M	5000.00	1	2
2	Maria	F	800.00	2	NULL
3	João	M	3000.00	1	2

$\pi_{nome, salario}$ ($\sigma_{Num\_Dep = ?}$ (Empregado))

Álgebra Relacional

Extração de Dados de uma Tabela (Consulta)

SQL

```
SELECT Nome, Salario FROM Empregado
 WHERE Num_Dep = &cod_dep;
```

- Para utilizar, após digitar o comando acima, o sistema apresenta a mensagem

Informe o valor para cod_dep:

- Caso o usuário digitasse **1**

Nome	Salario
José	5000.00
João	3000.00

Operadores SQL

- BETWEEN e NOT BETWEEN: substituem o uso dos operadores <= e >=

SQL

```
... WHERE <nome atributo> BETWEEN
    <valor1>   AND <valor2>;
```

- Exemplo 13: Listar os nomes dos empregados com salário entre R$ 4.000,00 e R$ 10.000,00

Operadores SQL

Empregado

Cad	Nome	Sexo	Salario	Num_Dep	Cad_Spv
1	José	M	5000.00	1	2
2	Maria	F	800.00	2	NULL
3	João	M	3000.00	1	2

π_{nome} ($\sigma_{salario\ between\ 4000.00\ and\ 10000.00}$ (Empregado))

Álgebra Relacional

SQL

```
SELECT Nome FROM Empregado
WHERE Salario BETWEEN 4000 AND 10000;
```

Nome
José

Operadores SQL

- LIKE e NOT LIKE: só se aplicam sobre atributos do tipo char
  - Operam como = e < >, respectivamente
  - O uso do símbolo % permite que a posição na cadeia de caracteres seja substituída por qualquer cadeia de caracteres
  - O uso do símbolo _ permite que a posição na cadeia de caracteres seja substituída por qualquer caractere

Operadores SQL

- LIKE e NOT LIKE (Cont.)

SQL

```
...WHERE <nome atributo> LIKE <valor1>;
```

- Exemplo 14: Listar os nomes dos empregados que iniciam com Jo

Empregado

Cad	Nome	Sexo	Salario	Num_Dep	Cad_Spv
1	José	M	5000.00	1	2
2	Maria	F	800.00	2	NULL
3	João	M	3000.00	1	2

π_{nome} ($\sigma_{nome\ like\ 'Jo\%'}$ (Empregado))

Álgebra Relacional

Operadores SQL

SQL

```
SELECT Nome FROM Empregado
WHERE Nome LIKE 'Jo%';
```

Nome
José
João

Operadores SQL

- IN e NOT IN: procuram dados que estão ou não contidos em um dado conjunto de valores

SQL

```
... WHERE <nome atributo> IN <valores>;
```

- Exemplo 15: Listar o nome e data de nascimento dos dependentes com grau de parentesco 'M' ou 'P'
  - Considerando criada a tabela Dependente

Operadores SQL

Dependente

Cad_emp	Nome	Data_nasc	Grau_P
1	Bruno	01/02/2000	P
2	Gina	05/10/2002	M
1	Telma	04/03/2010	D

$\pi_{nome,\ data\_nasc}$ ($\sigma_{grau\_p\ in\ ('M', 'P')}$ (Dependente))

Álgebra Relacional

SQL

```
SELECT Nome, Data_Nasc FROM Dependente
WHERE Grau_P IN ('M', 'P');
```

Nome	Data_nasc
Bruno	01/02/2000
Gina	05/10/2002

Operadores SQL

- IS NULL e IS NOT NULL: identificam se o atributo tem valor nulo (não informado) ou não

SQL

```
... WHERE <nome atributo> IS NULL;
```

- Exemplo 16: Listar número e nome dos projetos que não tenham local definido

Projeto

Numero	Nome	Num_Dep	Local
1	Desenvolvimento	1	Recife
2	Análise	1	Olinda
3	Testes	2	NULL

$\pi_{numero, nome}$ ($\sigma_{local\ is\ NULL}$ (Projeto))

Álgebra Relacional

Operadores SQL

SQL

```
SELECT Numero, Nome FROM Projeto
WHERE Local IS NULL;
```

Numero	Nome
3	Testes

Ordenando os Dados Selecionados

- Cláusula ORDER BY

SQL

```
SELECT <lista atributos> FROM <tabela>
      [WHERE <condição>]
ORDER BY <Nome atributo> {ASC | DESC};
```

Crescente (*default*)

Ordenando os Dados Selecionados

- Exemplo 17: Listar todos os dados dos empregados ordenados ascendentemente por nome

Empregado

Cad	Nome	Sexo	Salario	Num_Dep	Cad_Spv
1	José	M	5000.00	1	2
2	Maria	F	800.00	2	NULL
3	João	M	3000.00	1	2

$\pi_{cad, nome, sexo, salario, num\_dep, cad\_spv}$ ($\sigma_{nome\ ASC}$ (Empregado))

Álgebra Relacional

Ordenando os Dados Selecionados

SQL

```
SELECT * FROM Empregado ORDER BY Nome;
```

Cad	Nome	Sexo	Salario	Num_Dep	Cad_Spv
3	João	M	3000.00	1	2
1	José	M	5000.00	1	2
2	Maria	F	800.00	2	NULL

Ordenando os Dados Selecionados

- Exemplo 18: Listar todos os dados dos empregados ordenados descendentemente por salário

Empregado

Cad	Nome	Sexo	Salario	Num_Dep	Cad_Spv
1	José	M	5000.00	1	2
2	Maria	F	800.00	2	NULL
3	João	M	3000.00	1	2

$\pi_{cad, nome, sexo, salario, num\_dep, cad\_spv}$ ($\sigma_{salario\ DESC}$ (Empregado))

Álgebra Relacional

Ordenando os Dados Selecionados

SQL

```
SELECT * FROM Empregado ORDER BY
Salario DESC;
```

Cad	Nome	Sexo	Salario	Num_Dep	Cad_Spv
1	José	M	5000.00	1	2
3	João	M	3000.00	1	2
2	Maria	F	800.00	2	NULL

Realizando Cálculo com Informação Selecionada

- Pode-se criar um campo para a resposta da consulta que não pertença à tabela a partir de cálculos sobre atributos da tabela
  - Uso de operadores aritméticos

Oper. Aritméticos	
+	Adição
-	subtração
*	Multiplicação
/	Divisão

Realizando Cálculo com Informação Selecionada

- Exemplo 19: Mostrar nome e o novo salário dos empregados, calculado com base no reajuste de 60% para os que ganham abaixo de R$ 4.000,00

Empregado *1.6

Cad	Nome	Sexo	Salario	Num_Dep	Cad_Spv
1	José	M	5000.00	1	2
2	Maria	F	800.00	2	NULL
3	João	M	3000.00	1	2

*1.6

$\pi_{\text{nome, salario * 1.6}}(\sigma_{\text{salario < 4000.00}}(\text{Empregado}))$

Álgebra Relacional

Realizando Cálculo com Informação Selecionada

Renomear

SQL

```
SELECT Nome, (Salario * 1.60) AS Novo_salario
FROM Empregado WHERE Salario < 4000.00;
```

Nome	Novo_Salario
Maria	1280.00
João	4800.00

Funções Agregadas

- Utilização de funções sobre conjuntos
  - Disparadas a partir do SELECT

Funções de Agregação	
AVG	Média
MIN	Mínimo
MAX	Máximo
COUNT	Contar
SUM	Somar

Funções Agregadas

- Exemplo 20: Mostrar o valor do maior salário dos empregados e o nome do empregado que o recebe

Empregado

Cad	Nome	Sexo	Salario	Num_Dep	Cad_Spv
1	José	M	5000.00	1	2
2	Maria	F	800.00	2	NULL
3	João	M	3000.00	1	2

$\pi_{\text{nome, salario}}(\sigma_{\text{max(salario)}}(\text{Empregado}))$

Álgebra Relacional

Funções Agregadas

- A solução é buscar o nome e o salário do empregado que tem o maior salário

Consulta aninhada

SQL

```
SELECT Nome, Salario FROM Empregado
 WHERE Salario IN (  SELECT MAX (Salario)
FROM EMPREGADO  );
```

Nome	Salario
José	5000.00

Funções Agregadas

- Exemplo 21: Mostrar qual o salário médio dos empregados

Empregado

Cad	Nome	Sexo	Salario	Num_Dep	Cad_Spv
1	José	M	5000.00	1	2
2	Maria	F	800.00	2	NULL
3	João	M	3000.00	1	2

média

$\pi_{salario}$ ($\sigma_{avg\ (salario)}$ (Empregado))

Álgebra Relacional

Funções Agregadas

SQL

```
SELECT  AVG (Salario) FROM Empregado;
```

AVG(Salario)
2933.33

Funções Agregadas

- Exemplo 22: Quantos empregados ganham mais de R$4.000,00?

Empregado

Cad	Nome	Sexo	Salario	Num_Dep	Cad_Spv
1	José	M	5000.00	1	2
2	Maria	F	800.00	2	NULL
3	João	M	3000.00	1	2

contar

$\sigma_{count\ (salario > 4000.00)}$ (Empregado)

Álgebra Relacional

Funções Agregadas

SQL

```
SELECT  COUNT (*) FROM Empregado
WHERE Salario > 4000.00;
```

Count(*)
1

Cláusula DISTINCT

- Elimina tuplas duplicadas do resultado de uma consulta
  - Exemplo 23: Quais os diferentes códigos dos supervisores dos empregados?

Empregado

Cad	Nome	Sexo	Salario	Num_Dep	Cad_Spv
1	José	M	5000.00	1	2
2	Maria	F	800.00	2	NULL
3	João	M	3000.00	1	2

Distinto

$\sigma_{distinct\ (cad\_spv)}$ (Empregado)

Álgebra Relacional

Cláusula DISTINCT

SQL

```
SELECT DISTINCT Cad_spv
FROM Empregado;
```

Cad_Spv
-
2

Cláusula GROUP BY

- Organiza a seleção de dados em grupos
  - Exemplo 24: Listar os quantitativos de empregados de cada sexo

Empregado

Cad	Nome	Sexo	Salario	Num_Dep	Cad_Spv
1	José	M	5000.00	1	2
2	Maria	F	800.00	2	NULL
3	João	M	3000.00	1	2

Contar Contar Agrupar

$\pi_{sexo, count(*)} (\sigma_{group\ by(sexo)} (Empregado))$

Álgebra Relacional

Cláusula GROUP BY

Exceção: Funções agregadas

Todos os atributos do SELECT devem aparecer no GROUP BY

SQL

```
SELECT Sexo, Count(*) FROM Empregado
GROUP BY Sexo;
```

Sexo	Count(*)
M	2
F	1

Cláusula HAVING

- Agrupando Informações de forma condicional
  - Vem depois do GROUP BY e antes do ORDER BY
  - Exemplo 25: Listar o número total de empregados que recebem salários superiores a R$1.000,00 em cada departamento com mais de 1 empregado

Cláusula HAVING

Empregado

Cad	Nome	Sexo	Salario	Num_Dep	Cad_Spv
1	José	M	5000.00	1	2
2	Maria	F	800.00	2	NULL
3	João	M	3000.00	1	2

Contar > Agrupar

$\pi_{num\_dep, count(*)} (\sigma_{salario > 1000.00,\ group\ by(num\_dep),\ count(*) > 1} (Empregado))$

Álgebra Relacional

Cláusula HAVING

SQL

```
SELECT Num_Dep, COUNT (*) FROM Empregado
 WHERE Salario > 1000 GROUP BY Num_Dep
HAVING COUNT(*) > 1;
```

Num_Dep	Count(*)
1	2

Uso de "Alias"

- Para substituir nomes de tabelas em comandos SQL
  - São definidos na cláusula FROM

SQL

```
SELECT A.nome FROM Departamento A
WHERE A.Numero = 15;
```

Alias

Consultando Dados de Várias Tabelas - Junção

- Junção de Tabelas (JOIN)
  - Citar as tabelas envolvidas na cláusula FROM
  - Qualificadores de nomes - utilizados para evitar ambiguidades
    - Referenciar os nomes de Empregado e de Departamento

```
Empregado.Nome
Departamento.Nome
```

Melhor ainda usar alias

Junção de Tabelas

- Exemplo 26: Listar o nome de cada empregado e o nome do departamento no qual está alocado

Empregado

Cad	Nome	Sexo	Salario	Num_Dep	Cad_Spv
1	José	M	5000.00	1	2
2	Maria	F	800.00	2	NULL
3	João	M	3000.00	1	2

Departamento

Numero	Nome	Cad_Ger	Data_Ini
1	RH	3	12/03/2010
2	Contabilidade	2	15/01/2009

$\pi_{E.nome,\ D.nome}$ (σ (Empregado $\bowtie_{Num\_Dep = Numero}$ Departamento))

Álgebra Relacional

Junção de Tabelas

SQL

```
SELECT E.Nome, D.Nome
 FROM Empregado E, Departamento D
WHERE E.Num_Dep = D.Numero;
```

E.NOME	D.NOME
José	RH
João	RH
Maria	Contabilidade

Junção de Tabelas

- Exemplo 27: Listar os nomes dos departamentos que têm projetos

Departamento

Numero	Nome	Cad_Ger	Data_Ini
1	RH	3	12/03/2010
2	Contabilidade	2	15/01/2009

Projeto

Numero	Nome	Num_Dep	Local
1	Desenvolvimento	1	Recife
2	Análise	1	Olinda
3	Testes	1	NULL

$\pi_{D.nome}$ (σ (Departamento $\bowtie_{Numero = Num\_Dep}$ Projeto))

Álgebra Relacional

Junção de Tabelas

SQL

```
SELECT D.Nome
FROM Departamento D, Projeto P
WHERE P.Num_Dep = D.Numero;
```

Nome
RH

Junção de Tabelas

- Pode-se utilizar as cláusulas (NOT) LIKE, (NOT) IN, IS (NOT) NULL misturadas aos operadores AND, OR e NOT nas equações de junção (cláusula WHERE)
  - Exemplo 28: Listar os nomes dos departamentos que têm projetos com número (identificação) inferior a 3 e estão localizados em Olinda ou Recife, ordenados por nome de departamento

Junção de Tabelas

Departamento

Numero	Nome	Cad_Ger	Data_Ini
1	RH	3	12/03/2010
2	Contabilidade	2	15/01/2009

Projeto

Numero	Nome	Num_Dep	Local
1	Desenvolvimento	1	Recife
2	Análise	1	Olinda
3	Testes	1	NULL

Álgebra Relacional

$\pi_{D.nome}$ ($\sigma_{P.Numero < 3, P.Local\ IN\ ('Olinda', 'Recife')}$ (Departamento $\bowtie_{Numero = Num\_Dep}$ Projeto))

Junção de Tabelas

SQL

```
SELECT D.Nome
FROM  Departamento D, Projeto P
WHERE  P.Local IN ('Olinda', 'Recife')
AND P.Numero < 3
AND P.Num_Dep =  D.Numero
ORDER BY  D.Nome;
```

Nome
RH

Junção de Tabelas

- Classificando uma Junção
  - Exemplo 29: Para cada departamento, liste o nome do departamento, e para cada um deles, listar o número, o nome e o salário de seus empregados, ordenando a resposta em ordem decrescente de salário e em ordem alfabética

Junção de Tabelas

Departamento

Numero	Nome	Cad_Ger	Data_Ini
1	RH	3	12/03/2010
2	Contabilidade	2	15/01/2009

Empregado

Cad	Nome	Sexo	Salario	Num_Dep	Cad_spv
1	José	M	5000.00	1	2
2	Maria	F	800.00	2	NULL
3	João	M	3000.00	1	2

$\pi_{D.Nome\ ,\ E.Cad,\ E.Nome,\ E.Salario,\ order\ by\ E.Salario\ DESC,\ E.Nome}$ (σ (Departamento $\bowtie_{Numero = Num\_Dep}$ Empregado))

Álgebra Relacional

Junção de Tabelas

SQL

```
SELECT  D.Nome, E.Cad, E.Nome, E.Salario
FROM  Departamento D,  Empregado E
WHERE  D.Numero = E.Num_Dep
ORDER BY E.Salario DESC, D.Nome;
```

D.NOME	E.CAD	E.NOME	E.SALARIO
RH	1	José	5000.00
RH	3	João	3000.00
Contabilidade	2	Maria	800.00

Junção de Tabelas

- Agrupando por meio de mais de um atributo em uma Junção
  - Exemplo 30: Encontre o total de projetos de cada empregado por departamento, informando o código do departamento e o cadastro desse empregado

Junção de Tabelas

Empregado

Cad	Nome	Sexo	Salario	Num_Dep	Cad_Spv
1	José	M	5000.00	1	2
2	Maria	F	800.00	2	NULL
3	João	M	3000.00	1	2

Trabalha_em

Cad_Emp	Num_Proj	Horas
1	1	20
1	2	24
2	2	44
3	3	15
3	2	15
3	1	14

$\pi_{E.Num\_Dep\ ,\ E.Cad,\ count(*)}\ (\sigma\ (Empregado \bowtie_{Cad\ =\ Cad\_Emp} Trabalha\_em))$

Álgebra Relacional

Junção de Tabelas

SQL

```
SELECT E.Num_Dep, E.Cad, COUNT(*) AS Total
 FROM Trabalha_em T, Empregado E
 WHERE E.Cad = T.Cad_Emp
 GROUP BY E.Num_Dep, E.Cad
 ORDER BY E.Num_Dep, E.Cad;
```

E.NUM_DEP	E.CAD	TOTAL
1	1	2
1	3	3
2	2	1

Junção de Tabelas

- Realizando Junção com mais de duas tabelas
  - Exemplo 31: Listar o nome dos empregados juntamente com o nome do departamento no qual estão alocados e que trabalhem mais de 20 horas em algum projeto

Junção de Tabelas

Empregado

Cad	Nome	Sexo	Salario	Num_Dep	Cad_Spv
1	José	M	5000.00	1	2
2	Maria	F	800.00	2	NULL
3	João	M	3000.00	1	2

Trabalha_em

Cad_Emp	Num_Proj	Horas
1	1	20
1	2	24
2	2	44
3	3	15
3	2	15
3	1	14

Departamento

Numero	Nome	Cad_Ger	Data_Ini
1	RH	3	12/03/2010
2	Contabilidade	2	15/01/2009

$\pi_{E.Nome\ ,\ D.Nome}\ (\sigma_{T.Horas\ >\ 20}\ (Trabalha\_Em \bowtie_{Cad\_Emp\ =\ Cad} Empregado,\ Empregado \bowtie_{Num\_Dep\ =\ Numero} Departamento))$

Álgebra Relacional

Junção de Tabelas

SQL

```
SELECT E.Nome, D.Nome
FROM Empregado E, Departamento D,
     Trabalha_em T
WHERE T.Horas > 20
  AND T.Cad_Emp = E.Cad
  AND E.Num_Dep = D.Numero;
```

E.NOME	D.NOME
José	RH
Maria	Contabilidade

Junção de Tabelas

- Inner join (às vezes chamada de "junção simples")
  - É uma junção de duas ou mais tabelas que retorna somente as tuplas que satisfazem à condição de junção
  - Equivalente à junção natural

Junção de Tabelas

- Outer join
  - Retorna todas as tuplas de uma tabela e somente as tuplas de uma tabela secundária onde os campos de junção são iguais (condição de junção é encontrada)
  - Para todas as tuplas de uma das tabelas que não tenham tuplas correspondentes na outra, pela condição de junção, é retornado null para todos os campos da lista do select que sejam colunas da outra tabela

Junção de Tabelas

- Outer join (Cont.)
  - Para escrever uma consulta que executa um outer join das tabelas A e B e retorna todas as tuplas de A além das tuplas comuns, utilizar

SQL

```
SELECT <atributos>
FROM <tabela A> LEFT [OUTER] JOIN <tabela B>
ON <condição de junção>;
```


Junção de Tabelas

- Outer join (Cont.)
  - Exemplo 32: Listar os nomes de todos os departamentos da companhia e os nomes e locais dos projetos de que são responsáveis

Junção de Tabelas

Departamento

Numero	Nome	Cad_Ger	Data_Ini
1	RH	3	12/03/2010
2	Contabilidade	2	15/01/2009

Projeto

Numero	Nome	Num_Dep	Local
1	Desenvolvimento	1	Recife
2	Análise	1	Olinda
3	Testes	1	NULL

Álgebra Relacional

$\pi_{D.nome, P.Nome, P.Local}$ (σ(Departamento $\bowtie_{Numero = Num\_Dep}$ Projeto))

Junção de Tabelas

SQL

```
SELECT D.Nome, P.Nome, P.Local
 FROM Departamento D   LEFT OUTER JOIN
Projeto  P
ON D.Numero = P.Num_Dep;
```

D.NOME	P.NOME	P.LOCAL
RH	Desenvolvimento	Recife
RH	Análise	Olinda
RH	Testes	
Contabilidade		

Junção de Tabelas

- Outer join (Cont.)
  - Para escrever uma consulta que executa uma outer join das tabelas A e B e retorna todas as tuplas de B além das tuplas comuns, utilizar

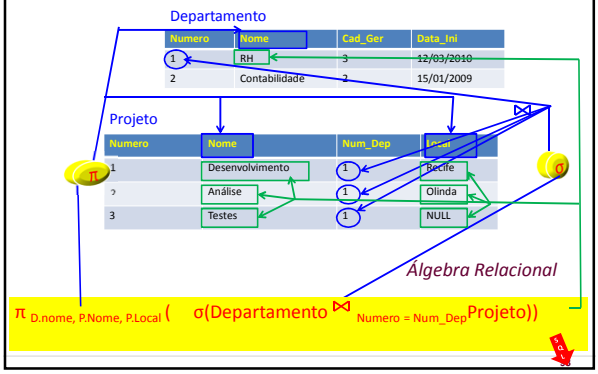
SQL

```
SELECT <atributos>
FROM <tabela A> RIGHT [OUTER] JOIN <tabela B>
ON <condição de junção>;
```


Junção de Tabelas

- Outer join (Cont.)
  - Exemplo 33: Listar os nomes dos departamentos da companhia com os nomes e locais dos projetos de que são responsáveis e os nomes dos demais projetos

Junção de Tabelas



Junção de Tabelas

SQL

```
SELECT D.Nome, P.Nome, P.Local
 FROM Departamento D  RIGHT OUTER JOIN
Projeto  P
ON D.Numero = P.Num_Dep;
```

D.NOME	P.NOME	P.LOCAL
RH	Desenvolvimento	Recife
RH	Análise	Olinda
RH	Testes	

Junção de Tabelas

- Outer join (Cont.)
  - Para escrever uma consulta que executa uma outer join e retorna todas as tuplas de A e B, estendidas com nulls se elas não satisfizerem à condição de junção, utilizar

SQL

```
SELECT <atributos>
FROM <tabela A> FULL [OUTER] JOIN <tabela B>
ON <condição de junção>;
```


Junção de Tabelas

- Outer join (Cont.)
  - Exemplo 34: Listar os nomes de todos os departamentos da companhia, os nomes e locais dos projetos de que sejam responsáveis e os nomes dos demais projetos

Junção de Tabelas

Departamento

Numero	Nome	Cad_Ger	Data_Ini
1	RH	3	12/03/2010
2	Contabilidade	2	15/01/2009

Projeto

Numero	Nome	Num_Dep	Local
1	Desenvolvimento	1	Recife
2	Análise	1	Olinda
3	Testes	1	NULL

Álgebra Relacional

$\pi_{D.nome, P.Nome, P.Local}$ (σ(Departamento $\bowtie_{Numero = Num\_Dep}$ Projeto))

Junção de Tabelas

SQL

```
SELECT D.Nome, P.Nome, P.Local
 FROM Departamento D  FULL OUTER JOIN
Projeto P
ON D.Numero = P.Num_Dep;
```

D.NOME	P.NOME	P.LOCAL
RH	Desenvolvimento	Recife
RH	Análise	Olinda
RH	Testes	
Contabilidade		

Consultas Encadeadas (Aninhadas)

- O resultado de uma consulta é utilizado por outra consulta, de forma encadeada e no mesmo comando SQL
- O resultado do comando SELECT mais interno (*subselect*) é usado por outro SELECT mais externo para obter o resultado final
- O SELECT mais interno (*subconsulta* ou *consulta aninhada*) pode ser usado apenas nas cláusulas WHERE e HAVING do comando mais externo ou em cálculos

Consultas Encadeadas (Aninhadas)

- Subconsultas devem ser escritas entre (e)
- Existem 3 tipos de subconsultas
  - ESCALAR → Retornam um único valor
  - ÚNICA LINHA → Retornam várias colunas, mas apenas uma única linha é obtida
  - TABELA → Retornam uma ou mais colunas e múltiplas linhas

Consultas Encadeadas (Aninhadas)

- Usando uma subconsulta com operador de igualdade
  - Exemplo 35: Listar cadastro, nome e salario dos empregados que trabalham no departamento de Contabilidade

Consultas Encadeadas (Aninhadas)

Empregado

Cad	Nome	Sexo	Salario	Num_Dep	Cad_Spv
1	José	M	5000.00	1	2
2	Maria	F	800.00	2	NULL
3	João	M	3000.00	1	2

Departamento

Numero	Nome	Cad_Ger	Data_Ini
1	RH	3	12/03/2010
2	Contabilidade	2	15/01/2009

$\pi_{E.Cad, E.Nome, E.Salario}$ ($\sigma_{D.Nome = 'Contabilidade'}$ (Empregado))

Álgebra Relacional

Consultas Encadeadas (Aninhadas)

SQL

```
SELECT Cad, Nome, Salario
FROM Empregado
WHERE Num_Dep =
        (SELECT Numero
         FROM Departamento
         WHERE Nome = 'Contabilidade' );
```

Subconsulta escalar

E.CAD	E.NOME	E.SALARIO
2	Maria	800.00

Consultas Encadeadas (Aninhadas)

- Usando uma subconsulta com função agregada
  - Exemplo 36: Listar os empregados cujos salários são maiores do que o salário médio, mostrando a diferença para o salário médio

Consultas Encadeadas (Aninhadas)

Empregado

Cad	Nome	Sexo	Salario	Num_Dep	Cad_Spv
1	José	M	5000.00	1	2
2	Maria	F	800.00	2	NULL
3	João	M	3000.00	1	2

$\pi_{E.Cad,\ E.Nome,\ E.Salario\ -\ Media(Salario)}\ (\sigma_{Salario\ >\ Media(Salario)}\ (Empregado))$

Álgebra Relacional

Consultas Encadeadas (Aninhadas)

SQL

```
SELECT Cad, Nome, Salario –
  (SELECT AVG (Salario) FROM Empregado)
AS DifSal
FROM Empregado
WHERE Salario > ( SELECT AVG ( Salario)
                  FROM Empregado );
```

E.CAD	E.NOME	DifSal
1	José	2066.67

Consultas Encadeadas (Aninhadas)

- Mais de um nível de aninhamento
  - Exemplo 37: Listar os dependentes dos funcionários que trabalham no departamento RH

Consultas Encadeadas (Aninhadas)

Dependente

Cad_Emp	Nome	Data_Nasc	Grau_P
1	Jonas	12/05/2000	Pai
1	Beatriz	05/06/2002	Pai
3	Clara	13/02/2001	Pai

Empregado

Cad	Nome	Sexo	Salario	Num_Dep	Cad_Spv
1	José	M	5000.00	1	2
2	Maria	F	800.00	2	NULL
3	João	M	3000.00	1	2

Departamento

Numero	Nome	Cad_Ger	Data_Ini
1	RH	3	12/03/2010
2	Contabilidade	2	15/01/2009

$\pi_{DEP.Nome,\ DEP.Data\_Nasc,\ DEP.Grau\_P}\ (\sigma_{Cad\_Emp\ =}\ \pi_{Cad}(\sigma_{Num\_Dep\ =}\ \pi_{Numero}\ (\sigma_{Nome\ = 'RH'}(Departamento))\ (Empregado)))\ (Dependente)$

Álgebra Relacional

Consultas Encadeadas (Aninhadas)

SQL

```
SELECT Nome, Data_nasc, Grau_P
FROM Dependente WHERE Cad IN
( SELECT Cad FROM Empregado
                WHERE Num_Dep =
                  ( SELECT Numero
                    FROM Departamento
            WHERE Nome = ' RH' ) );
```

Nome	Data_Nasc	Grau_P
Jonas	12/05/2000	Pai
Beatriz	05/06/2002	Pai

Cláusulas ANY/SOME

- São usadas com subconsultas que produzem uma única coluna de números
  - Exemplo 38: Listar os empregados cujos salários são maiores do que o salário de pelo menos um funcionário do departamento 1

Cláusulas ANY/SOME

Empregado

Cad	Nome	Sexo	Salario	Num_Dep	Cad_Spv
1	José	M	5000.00	1	2
2	Maria	F	800.00	2	NULL
3	João	M	3000.00	1	2

$\pi_{E.Cad,\ E.Nome,\ E.Sexo,\ E.Salario}(\sigma_{salario\ >\ E.salario\ where\ E.Num\_Dep\ =\ 1}(Empregado\ E))$

Álgebra Relacional

Cláusulas ANY/SOME

SQL

```
SELECT Cad, Nome, Sexo, Salario
FROM Empregado
WHERE Salario >
SOME ( SELECT Salario FROM Empregado
        WHERE Num_Dep = 1) ;
```

CAD	NOME	SEXO	SALARIO
1	José	M	5000.00

Cláusula ALL

- É utilizado com subconsultas que produzem uma única coluna de números
  - Exemplo 39: Listar os empregados cujos salários são menores do que o salário de cada funcionário do departamento 1

Cláusula ALL

Empregado

Cad	Nome	Sexo	Salario	Num_Dep	Cad_Spv
1	José	M	5000.00	1	2
2	Maria	F	800.00	2	NULL
3	João	M	3000.00	1	2

$\pi_{E.Cad,\ E.Nome,\ E.Sexo,\ E.Salario}(\sigma_{E.Salario\ <\ \forall\ E.salario\ where\ E.Num\_Dep\ =\ 1}(Empregado\ E))$

Álgebra Relacional

Cláusula ALL

SQL

```
SELECT Cad, Nome, Sexo, Salario
FROM Empregado
WHERE Salario < ALL ( SELECT Salario
                      FROM Empregado
                      WHERE Num_Dep = 1) ;
```

CAD	NOME	SEXO	SALARIO
2	Maria	F	800.00

Cláusulas EXISTS e NOT EXISTS

- Foram projetadas para uso apenas com subconsultas
  - EXISTS
    - Retorna TRUE ⇔ existe pelo menos uma linha produzida pela subconsulta
    - Retorna FALSE ⇔ a subconsulta produz uma tabela resultante vazia

Deve ser utilizado quando for necessário certificar-se que haverá resposta a uma subconsulta

  - Exemplo 40: Liste todos os empregados que trabalham no departamento de RH

Cláusulas EXISTS e NOT EXISTS

Empregado

Cad	Nome	Sexo	Salario	Num_Dep	Cad_Spv
1	José	M	5000.00	1	2
2	Maria	F	800.00	2	NULL
3	João	M	3000.00	1	2

Departamento

Numero	Nome	Cad_Ger	Data_Ini
1	RH	3	12/03/2010
2	Contabilidade	2	15/01/2009

$\pi_{E.Cad,\ E.Nome,\ E.Sexo,\ E.Salario}\ (\exists\sigma_{D.Nome\ =\ 'RH'}\ (Empregado))$

Álgebra Relacional

Cláusulas EXISTS e NOT EXISTS

SQL

```
SELECT Cad, Nome, Sexo, Salario
FROM Empregado E WHERE EXISTS
( SELECT D.Numero FROM Departamento D
  WHERE E.Num_Dep = D.Numero AND
        D.Nome = 'RH') ;
```

CAD	NOME	SEXO	SALARIO
1	José	M	5000.00
3	João	M	3000.00

Regras Genéricas de Subconsultas

- A cláusula ORDER BY não pode ser usada em uma subconsulta
- A lista de atributos especificados no SELECT de uma subconsulta deve conter um único elemento (exceto para EXISTS)
- Nomes de atributos especificados na subconsulta estão associados às tabelas listadas na cláusula FROM da mesma
  - É possível referir-se a uma tabela da cláusula FROM da consulta mais externa utilizando qualificadores de atributos

Regras Genéricas de Subconsultas

- Quando a subconsulta é um dos operandos envolvidos em uma comparação, ela deve aparecer no lado direito da comparação

Operações de Conjunto

- UNION
  - Linhas duplicadas são removidas da tabela resultante
  - Exemplo 41: Construa uma lista de todos os locais onde existe um departamento ou um projeto

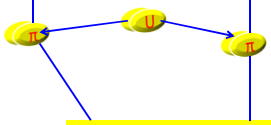
Operações de Conjunto

Projeto

Numero	Nome	Num_Dep	Local
1	Desenvolvimento	1	Recife
2	Análise	1	Olinda
3	Testes	1	NULL

Locais

Num_Dep	Nome_Loc
1	Camaragibe
1	Jaboatão
2	Olinda



$\pi_{Projeto.Local} \cup \pi_{Locais.Nome\_Loc}$

Álgebra Relacional

Operações de Conjunto

SQL

```
( SELECT Local FROM Projeto
  WHERE Local IS NOT NULL )
UNION
  ( SELECT Nome_Loc FROM Locais ) ;
```

Local
Recife
Olinda
Camaragibe
Jaboatão

Operações de Conjunto

- INTERSECT
  - Exemplo 42: Construa uma lista de todos os locais onde existem ambos um departamento e um projeto

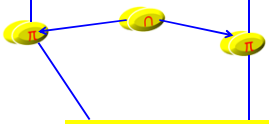
Operações de Conjunto

Projeto

Numero	Nome	Num_Dep	Local
1	Desenvolvimento	1	Recife
2	Análise	1	Olinda
3	Testes	1	NULL

Locais

Num_Dep	Nome_Loc
1	Camaragibe
1	Jaboatão
2	Olinda



$\pi_{Projeto.Local} \cap \pi_{Locais.Nome\_Loc}$

Álgebra Relacional

Operações de Conjunto

SQL

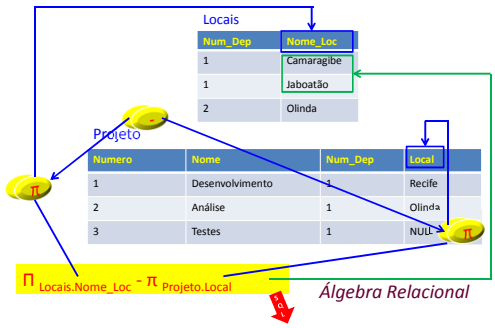
```
( SELECT Local FROM Projeto )
INTERSECT
  ( SELECT Nome_Loc FROM Locais ) ;
```

Local
Olinda

Operações de Conjunto

- MINUS
  - Exemplo 43: Construa uma lista de todos os locais onde existe um departamento mas nenhum projeto

Operações de Conjunto



Operações de Conjunto

SQL

```
( SELECT Nome_Loc FROM Locais )
MINUS
    ( SELECT Nome_Loc FROM Projeto );
```

Nome_Loc
Camaragibe
Jaboatão

Inserção de Dados em Tabelas

- Adicionar uma ou várias tuplas à tabela → INSERT

SQL

```
INSERT INTO <tabela> (<lista de atributos>)
VALUES (<valores>);
```

Inserção de apenas uma Linha

- Caso sejam fornecidos valores para todos os atributos, na ordem em que foram definidos, não é necessário fornecer (**<lista de atributos>**)

SQL

```
INSERT INTO <tabela> VALUES (<valores>);
```


Inserção de Dados em Tabelas

- Exemplo 44: Inserir dados de um empregado

Empregado

Cad	Nome	Sexo	Salario	Num_Dep	Cad_Spv
1	José	M	5000.00	1	2
2	Maria	F	800.00	2	NULL
3	João	M	3000.00	1	2

INSERIR

INSERIR $_{4, Clara, F, 7000.00, 1, 2}$ (Empregado)

Inserção de Dados em Tabelas

SQL

```
INSERT INTO Empregado
VALUES (4, 'Clara', 'F', 7000.00, 1, 2);
```

Não é necessário informar nomes dos atributos

Empregado

Cad	Nome	Sexo	Salario	Num_Dep	Cad_Spv
1	José	M	5000.00	1	2
2	Maria	F	800.00	2	NULL
3	João	M	3000.00	1	2
4	Clara	F	7000.00	1	2

Inserção de Dados em Tabelas

- Inserir dados recuperados de uma tabela em outra tabela – uso do SELECT

SQL

```
INSERT INTO <tabela> (<lista de atributos>)
SELECT <lista de atributos> FROM <tabela>
WHERE <condição>;
```

Inserção de Várias Linhas

As duas <lista de atributos> devem ser união compatíveis

- Exemplo 45: Armazenar na tabela Depto_Info (Nome_Depto, Num_Emp, Total_Sal) para cada departamento com mais de 1 empregado, o nome do departamento, o número de empregados e a soma dos salários pagos

Inserção de Dados em Tabelas

Empregado

Cad	Nome	Sexo	Salario	Num_Dep	Cad_Spv
1	José	M	5000.00	1	2
2	Maria	F	800.00	2	NULL
3	João	M	3000.00	1	2
4	Clara	F	7000.00	1	2

>1

Depto_Info

Nome_Depto	...
...	...
...	....

Departamento

Numero	Nome	Cad_Ger	Data_...
1	RH	3	12/03/2010
2	Contabilidade	2	15/01/2009

INSERIR

INSERIR σ (D.Nome, Count(*) > 1, SUM(Salario), Departamento $\bowtie$ Numero = Num_Dep Departamento) (Depto_Info)

Inserção de Dados em Tabelas

SQL

```
INSERT INTO Depto_info (nome_depto,
                        num_emp, total_sal)
SELECT D.nome, COUNT(*), SUM (E.salario)
FROM Departamento D, Empregado E
WHERE D.numero = E.Num_Dep
GROUP BY D.nome HAVING COUNT (*) > 1;
```

Depto_Info

Nome_Depto	Num_Emp	Total_Sal
RH	3	15000.00

Atualização de Dados em Tabelas

- Com base em critérios especificados, alterar valores de campos de uma tabela → UPDATE

SQL

```
UPDATE <nome tabela>
SET <nome atributo> = <valor>
WHERE <condição>;
```

- Exemplo 46: Atualizar salário do empregado 2 para R$9500,00

Atualização de Dados em Tabelas

Empregado

Cad	Nome	Sexo	Salario	Num_Dep	Cad_Spv
1	José	M	5000.00	1	2
2	Maria	F	800.00	2	NULL
3	João	M	3000.00	1	2
4	Clara	F	7000.00	1	2

π ATUALIZAR σ

ATUALIZAR π E.Salario = 9500.00, σ (Cad = 2) (Empregado)

Atualização de Dados em Tabelas

SQL

```
UPDATE Empregado SET Salario = 9500.00
WHERE Cad = 2;
```

Empregado

Cad	Nome	Sexo	Salario	Num_Dep	Cad_Spv
1	José	M	5000.00	1	2
2	Maria	F	9500.00	2	NULL
3	João	M	3000.00	1	2
4	Clara	F	7000.00	1	2

Remoção de Tuplas de Tabela

- Exclusão de dados de uma tabela → DELETE

SQL

DELETE FROM <tabela> WHERE <condição>;

- Exemplo 47: Remover todos os empregados com salário inferior a R$ 5000,00

Remoção de Tuplas de Tabela

Empregado

Cad	Nome	Sexo	Salario	Num_Dep	Cad_Spv
1	José	M	5000.00	1	2
2	Maria	F	9500.00	2	NULL
3	João	M	3000.00	1	2
4	Clara	F	7000.00	1	2

REMOVER

REMOVER $\sigma_{(Salario < 5000.00)}$ (Empregado)

Remoção de Tuplas de Tabela

SQL

DELETE FROM Empregado
WHERE Salario < 5000.00;

Empregado

Cad	Nome	Sexo	Salario	Num_Dep	Cad_Spv
1	José	M	5000.00	1	2
2	Maria	F	9500.00	2	NULL
4	Clara	F	7000.00	1	2

O empregado João foi removido

Utilizando Visões (VIEWS)

- São tabelas virtuais que não ocupam espaço físico no BD para os dados
- Operações
  - Criação e utilização
  - Inserção e modificação (semântica depende da definição/natureza da visão)

SQL

CREATE VIEW <nome da view>
(<lista de atributos>) AS SELECT...;

- Exemplo 48: Criar uma visão dos empregados do departamento 1 que tenham mais de 20 horas de trabalho em algum projeto

Utilizando Visões (VIEWS)

Empregado

Cad	Nome	Sexo	Salario	Num_Dep	Cad_Spv
1	José	M	5000.00	1	2
2	Maria	F	9500.00	2	NULL
4	Clara	F	7000.00	1	2

Trabalha_em

Cad_Emp	Num_Proj	Horas
1	1	20
1	2	24
2	2	44
4	1	34

π σ

CRIAR VISÃO π E.Nome, T.Num_Proj, σ (T.Horas > 20, Num_Dep = 1)
(Empregado $\bowtie$ Cad = Cad_Emp Trabalha_Em)

Utilizando Visões (VIEWS)

SQL

```
CREATE VIEW Dep_1 AS
SELECT E.Nome, T.Num_Proj
FROM Empregado E, Trabalha_em T
WHERE T.Horas > 20
AND T.Cad_Emp = E.Cad
AND E.Num_Dep = 1;
```

Utilizando Visões (VIEWS)

- Para testar, consultar a VIEW

SQL
```
SELECT * FROM Dep_1 ;
```

Dep_1

Nome	Num_Proj
José	2
Clara	1

Garantindo Privilégios de Acesso

- Comando GRANT

SQL
```
GRANT <privilégios> ON <nome tabela/view>
TO <usuário>;
```

- Onde
  - <privilégios>: SELECT, INSERT, DELETE, UPDATE, ALL PRIVILEGES e
  - <usuário>: usuário cadastrado, PUBLIC

Garantindo Privilégios de Acesso

- Exemplo 49: Conceder a permissão de consulta sobre a tabela EMPREGADO à usuária acs

SQL
```
GRANT SELECT ON Empregado TO acs;
```


Removendo Privilégios de Acesso

- Comando REVOKE

SQL
```
REVOKE <privilégios> ON <nome tabela/view>
FROM <usuário>;
```

  - Exemplo 50: Remover a permissão de consulta dada aos demais usuários do banco

SQL
```
REVOKE SELECT ON Projeto FROM PUBLIC;
```