Gerenciamento de Dados e Informação

Fernando Fonseca
Ana Carolina
Robson Fidalgo

Centro de Informática
Universidade Federal de Pernambuco
CIn.ufpe.br

SQL

- 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

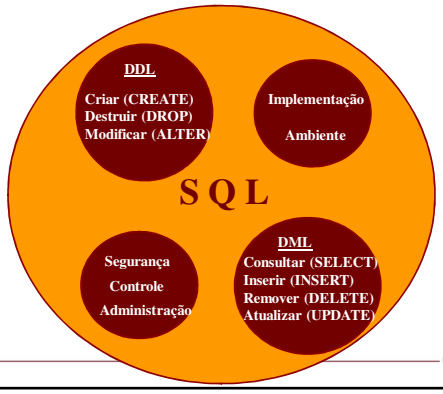
Enfoques de SQL

- 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

Enfoques de SQL

- 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



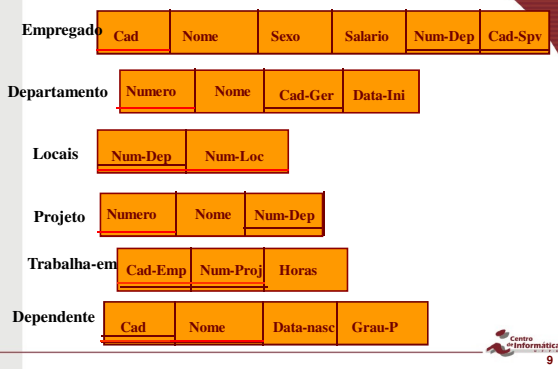
SQL - Vantagens

- 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



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

```
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_float, binary_double, date, timestamp, blob, clob, nclob

Criação de Tabelas

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

```
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

Criação de Tabelas

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

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

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

13

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 com o comando
      - <nome>.NEXTVAL
- Lista das chaves estrangeiras na forma

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

14

Criação de Tabelas

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

```
CONSTRAINT nometabela_check
CHECK (salario >= 450)
```

  - Só admite valor único

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

15

Criação de Tabelas

- Exemplo 1

Empregado(Cad, Nome, Sexo, Salario, Num-Dep, Cad-Spv)

```
CREATE TABLE Empregado
 (Cad number,
  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));
```

Criação de Tabelas

- Exemplo 2

Trabalha-em(Cad-Emp, Num-Proj, Horas)

```
CREATE TABLE Trabalha_em
  (Cad_emp number,
   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> (<atributo1>[, <atributo2>...]);
```

18

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

19

Alteração de Tabelas

- Exemplos
  - Acrescentar coluna na tabela Empregado

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

  - Remover coluna na tabela Empregado

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

20

Remoção de Tabelas

- Eliminar uma tabela que foi previamente criada → DROP TABLE

DROP TABLE <tabela>;

- Exemplo

DROP TABLE Empregado;

- Observação
  - Os dados são também excluídos

21

Extração de Dados de uma Tabela (Consulta)

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

SELECT <lista de atributos> FROM <tabela>;

  - Exemplo: Listar nome e salário de todos os empregados

SELECT Nome, Salario FROM Empregado;

22

Extração de Dados de uma Tabela (Consulta)

- Selecionando todos os atributos

SELECT * FROM <tabela>;

- Exemplo

SELECT * FROM Empregado;

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

23

Extração de Dados de uma Tabela (Consulta)

- Selecionando tuplas da tabela → cláusula WHERE

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

24

Extração de Dados de uma Tabela (Consulta)

- Exemplos
  - Listar nome e sexo dos empregados do departamento 15

```
SELECT Nome, Sexo FROM Empregado
 WHERE Num_Dep = 15;
```

  - Listar nome e sexo dos empregados do departamento 15 com salário > R$ 1.000,00

```
SELECT Nome, Sexo FROM Empregado
 WHERE Num_Dep = 15 AND Salario > 1000;
```

25

Extração de Dados de uma Tabela (Consulta)

- Consulta para o usuário fornecer valores para o SELECT só na hora da execução
  - Colocar parâmetro na forma &<variável>
- Exemplo
  - Listar nome e salário dos empregados do departamento com um dado código

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

26

Operadores SQL

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

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

  - Exemplo: Listar os nomes dos empregados com salário entre R$ 1.000,00 e R$ 2.000,00

```
SELECT  Nome  FROM  Empregado
WHERE Salario BETWEEN 1000 AND 2000;
```

27

Operadores SQL

- LIKE e NOT LIKE: só se aplicam sobre atributos do tipo char. Operam como = e < >, utilizando os símbolos % (substitui uma palavra) e _ (substitui um caractere)

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

  - Exemplo: Listar os empregados que têm como primeiro nome José

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

28

Operadores SQL

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

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

Exemplo: Listar o nome e data de nascimento dos dependentes com grau de parentesco 'M' ou 'P'

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

29

Operadores SQL

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

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

  - Exemplo: Listar os dados dos projetos que não tenham local definido

```
SELECT * FROM Projeto
WHERE Local IS NULL;
```

30

Ordenando os Dados Selecionados

- Cláusula ORDER BY

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


Ordenando os Dados Selecionados

- Exemplos
  - Listar todos os dados dos empregados ordenados ascendentemente por nome

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

  - Listar todos os dados dos empregados ordenados descendentemente por salário

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


Realizando Cálculo com Informação Selecionada

- Pode-se criar um campo 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: Mostrar o novo salário dos empregados calculado com base no reajuste de 60% para os que ganham abaixo de R$ 1.000,00

Renomear

```
SELECT Nome, (Salario * 1.60) AS Novo_salario
FROM Empregado WHERE Salario < 1000;
```


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

- Exemplos
  - Mostrar o valor do maior salário dos empregados e o nome do empregado que o recebe

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

Consulta aninhada

Funções Agregadas

- Exemplos
  - Mostrar qual o salário médio dos empregados

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

  - Quantos empregados ganham mais de R$1.000,00?

```
SELECT COUNT (*) FROM Empregado
WHERE Salario > 1000;
```

37

Cláusula DISTINCT

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

```
SELECT DISTINCT Salario
FROM Empregado;
```

38

Cláusula GROUP BY

- Organiza a seleção de dados em grupos
  - Exemplo: Listar os empregados agrupados por sexo, informando as quantidades

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

Atributos do GROUP BY devem aparecer no SELECT

Exceção: Funções agregadas

39

Cláusula HAVING

- Agrupando Informações de forma condicional
  - Vem depois do GROUP BY e antes do ORDER BY
  - Exemplo: Listar o número total de empregados que recebem salários superior a R$1.000,00, agrupados por departamento mas só daqueles com mais de 5 empregados

```
SELECT Num_Dep, COUNT (*) FROM Empregado
 WHERE Salario > 1000
GROUP BY Num_Dep HAVING COUNT(*) > 5;
```

40

Uso de "Alias"

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

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

Alias

41

Consultando Dados de Várias Tabelas - Junção

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

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

42

Junção de Tabelas

- Exemplos
  - Listar o nome do empregado e do departamento no qual está alocado

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

  - Listar os nomes dos departamentos que têm projetos

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


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: Listar os nomes dos departamentos que têm projetos com número superior a 99 e localizados em RJ ou SP, ordenados por nome de departamento

Junção de Tabelas

```
SELECT D.Nome
FROM  Departamento D, Projeto P
WHERE  P.Local IN ('RJ', 'SP')
AND P.Numero > 99
AND P.Num_Dep =  D.Numero
ORDER BY  D.Nome;
```


Junção de Tabelas

- Classificando uma Junção
  - Exemplo: 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

```
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;
```


Junção de Tabelas

- Agrupando através de mais de um atributo em uma Junção
  - Exemplo: Encontre o total de projetos de cada funcionário por departamento, informando o cadastro do empregado.

```
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;
```


Junção de Tabelas

- Juntando mais de duas tabelas
- Exemplos
  - Listar o nome dos empregados, com seu respectivo departamento que trabalhem mais de 20 horas em algum projeto

Junção de Tabelas

```
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;
```


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 tenha nenhuma tupla correspondente 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 uma outer join das tabelas A e B e retorna todas as tuplas de A além das tuplas comuns, utilizar

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


Junção de Tabelas

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

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


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

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

Junção de Tabelas

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

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

55

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

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

56

Junção de Tabelas

- Outer join (Cont.)
  - Exemplo: 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

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

57

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

58

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

59

Consultas Encadeadas (Aninhadas)

- Exemplos
  - Usando uma subconsulta com operador de igualdade: Listar os empregados que trabalham no departamento de Informática

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

Subconsulta escalar

60

Consultas Encadeadas (Aninhadas)

- Usando uma subconsulta com função agregada: Listar os empregados cujos salários são maiores do que o salário médio, mostrando o quanto são maiores

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

61

Consultas Encadeadas (Aninhadas)

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

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

62

Cláusulas ANY/SOME

- São usadas com subconsultas que produzem uma única coluna de números
  - Exemplo: Listar os empregados cujos salários são maiores do que o salário de pelo menos um funcionário do departamento 20

```
SELECT Cad, Nome, Sexo, Salario
FROM Empregado
WHERE Salario >
SOME ( SELECT Salario FROM Empregado
       WHERE Num_Dep = 20) ;
```

63

Cláusula ALL

- É utilizado com subconsultas que produzem uma única coluna de números
  - Exemplo: Listar os empregados cujos salários são maiores do que o salário de cada funcionário do departamento 15

```
SELECT Cad, Nome, Sexo, Salario
FROM Empregado
WHERE Salario > ALL ( SELECT Salario
                      FROM Empregado
                WHERE Num_Dep = 15) ;
```

64

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

65

Cláusulas EXISTS e NOT EXISTS

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

```
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 = 'Informática') ;
```

66

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: Construa uma lista de todos os locais onde existe um departamento ou um projeto

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


Operações de Conjunto

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

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


Operações de Conjunto

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

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


Inserção de Dados em Tabelas

- Adicionar uma ou várias tuplas à tabela → INSERT

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

Uma Linha

- Exemplo: Inserir dados de um empregado

```
INSERT INTO Empregado(Cad, Nome, Sexo,
Salario, Num_Dep, Cad_Supv)
VALUES (015, 'José da Silva', 'M',
        1000.00, 1, 020);
```

Inserção de Dados em Tabelas

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

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

- Exemplo: Armazenar em uma tabela para cada departamento com mais de 50 empregados, o número de empregados e a soma dos salários pagos

Inserção de Dados em Tabelas

```
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 (*) > 50;
```


Atualização de Dados em Tabelas

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

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

- Exemplo: Atualizar salário do empregado 15 para R$1500,00

```
UPDATE Empregado SET Salario = 1500.00
WHERE Cad = 15;
```


Remoção de Tuplas de Tabela

- Exclusão de dados de uma tabela → DELETE

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

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

```
DELETE FROM Empregado
WHERE Salario > 5000.00;
```


Utilizando Visões (VIEWS)

- São tabelas virtuais que não ocupam espaço físico
- Operações
  - Criação e utilização
  - Inserção e modificação (semântica depende da definição/natureza da visão)

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


Utilizando Visões (VIEWS)

- Exemplo: Criar uma visão dos empregados do departamento 10 que tenham mais de 20 horas de trabalho em projetos

```
CREATE VIEW Dep_10 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 = 10;
```

Garantindo Privilégios de Acesso

- Comando GRANT

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

GRANT SELECT ON Empregado TO acs;

Removendo Privilégios de Acesso

- Comando REVOKE

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

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

REVOKE SELECT ON Projeto FROM PUBLIC;