Banco de Dados I

5 – Linguagens de Consulta

Grinaldo Lopes de Oliveira (grinaldo@gmail.com)
Curso Superior de Tecnologia em
Análise e Desenvolvimento de Sistemas

*** Material com créditos de colaboração dos professores Márcio Soussa, Leandro Coelho e Pablo Florentino**

Agenda

- **Aprendendo**
  - **Álgebra Relacional**
  - **SQL**

Álgebra Relacional

Modelo Relacional

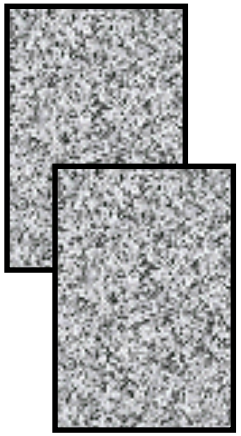
Álgebra Relacional

- Conjunto de operadores que possuem relações como operandos e retorna relações como resultado;
- Consiste de oito operadores, dois grupos de quatro:
  - Operações de conjunto tradicionais: União, Interseção, Diferença e Produto Cartesiano (Union, Intersect, Difference, Product);
  - Operações relacionais especiais: selecionar, projetar, junção e divisão (Select, Project, Join, Divide).

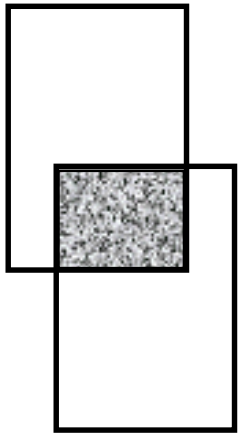
Modelo Relacional

Álgebra Relacional

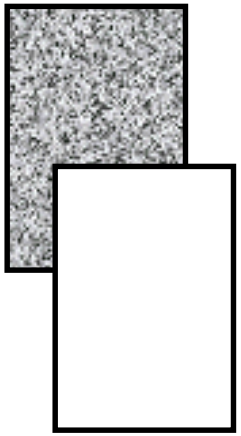
União



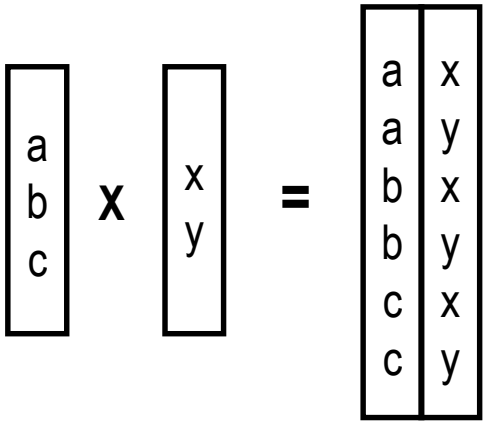
Interseção



Diferença

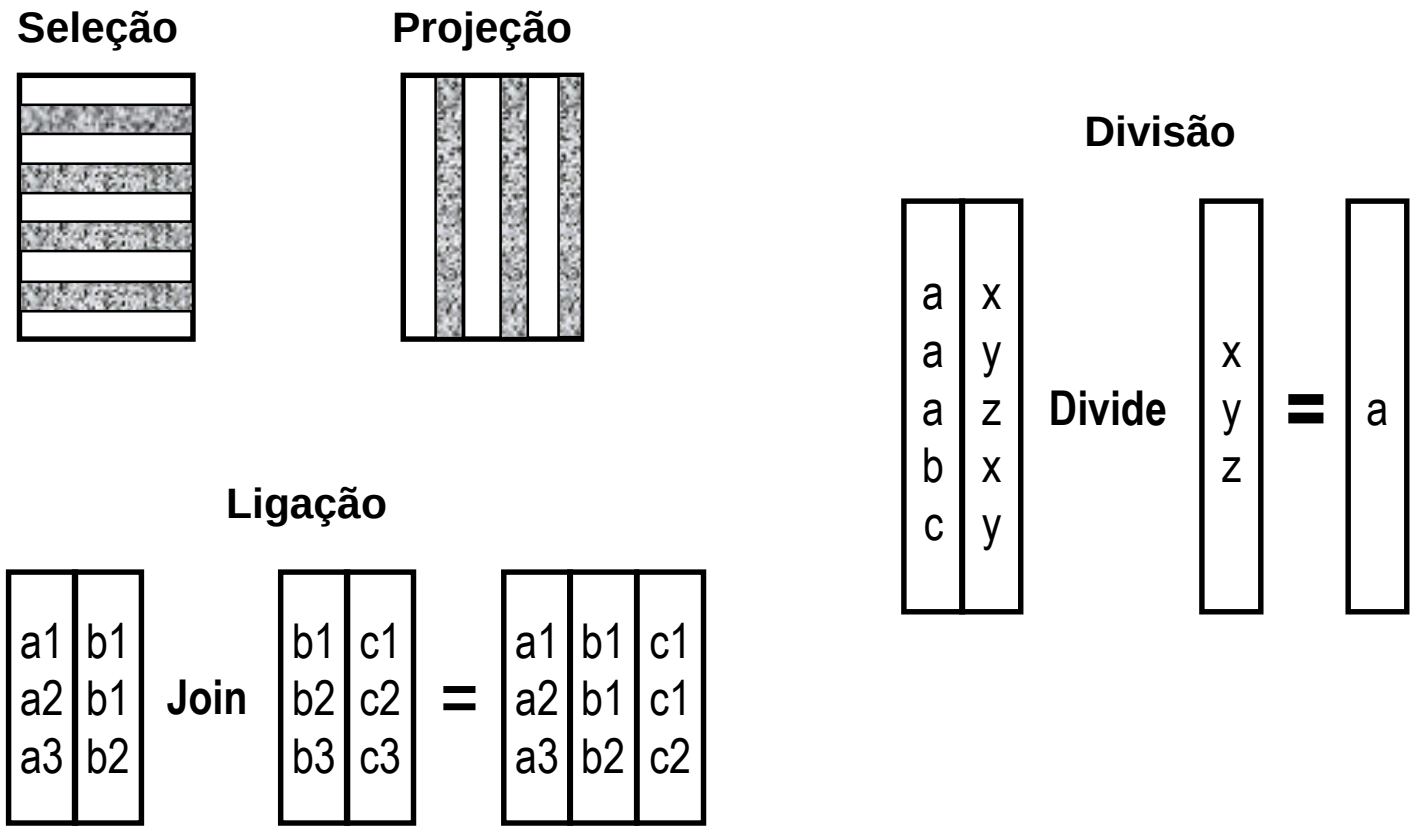


Produto Cartesiano



Modelo Relacional

Álgebra Relacional



Modelo Relacional

Álgebra Relacional

Seja **A** o conjunto de fornecedores de Londres e **B** o conjunto de fornecedores da peça P1.

- União
  - Constrói uma relação consistindo em todas as tuplas que aparecem em uma ou em ambas as relações;
  - **A Union B** = Conjunto com os fornecedores de Londres ou fornecedores da peça P1.
- Interseção
  - Constrói uma relação consistindo em todas as tuplas que aparecem em ambas as relações;
  - **A Intersect B** = Conjunto com os fornecedores de Londres e que fornecem a peça P1.

Álgebra Relacional

- Diferença
  - Constrói uma relação contendo as tuplas que aparecem na primeira relação, mas não aparecem na segunda;
  - **A MINUS B** = Conjunto com os fornecedores de Londres e que não fornecem a peça P1.
- Produto
  - Constrói uma relação a partir de duas relações, consistindo em todas as possibilidades de pares de tuplas, uma para cada duas relações;
  - **A TIMES B** = Conjunto com cada fornecedor de Londres associado a cada fornecedor da peça P1.

Modelo Relacional

Álgebra Relacional

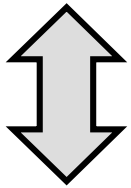
- Seleção
  - Extrai tuplas específicas de uma dada relação;
  - **R** WHERE **R.X** theta **R.Y**
  - Exemplos:
    - **S** WHERE CITY = “London”
    - **P** WHERE WIGHT < 14
- Projeção
  - Extrai atributos específicos de uma dada relação;
  - **R** [X,Y,...,Z]
  - Exemplos:
    - **S** [CITY]
    - **P** [P#, PNAME]

Álgebra Relacional

- Junção
  - Constrói uma relação consistindo em todas as possibilidades de pares de tuplas concatenadas, de forma que, em cada par, as duas tuplas satisfaçam a uma condição específica.

(A TIMES B) WHERE A.X = B.X

X é um atributo comum a A e B



(A JOIN B)

Álgebra Relacional

- Divisão
  - Toma duas relações, uma binária e uma unária, e constrói uma relação consistindo em todos os valores de um atributo da relação binária com equivalência a todos os valores da relação unária.
  - (A **DIVIDE BY** B)
  - Exemplo:
    - ((SP [S#, P#] **DIVIDE BY** P [P#]) **JOIN** S) [SNAME] =

      Nome dos fornecedores que fornecem todas as peças.

Operadores Adicionais

- Funções de agregação e agrupamento
  - SUM (Soma)
  - AVERAGE (Média)
  - MAX (Máximo)
  - MIN (Mínimo)
  - COUNT (Contador)
- Símbolo : ζ
- Ex: curso ζ count (aluno)

Modelo Relacional

Álgebra Relacional

- Exemplos:
  - Nomes dos fornecedores que fornecem a peça P2:

    ((S JOIN SP) WHERE P# = “P2”) [SNAME]

  - Nomes dos fornecedores que fornecem pelo menos uma peça vermelha:

    (((P WHERE COLOR = “RED” [P#] JOIN SP) [S#] JOIN S) [SNAME]

  - Pares de números de fornecedores de forma que cada par seja localizado na mesma cidade:

    DEFINE ALIAS S1 FOR S
    DEFINE ALIAS S2 FOR S

    ((S1 TIMES S2) WHERE S1.CITY = S2.FIRST
    AND S1.S# < S2.S#) [S1.S#, S2.S#]

Debate em Sala de Aula

- Em sua opinião, para que serviria, na prática, uma operação de diferença utilizando SQL ?

SQL

SQL

- ***S****tructured* ***Q****uery* ***L****anguage;*
- Linguagem de alto nível para SGBD relacionais;
- Originalmente desenvolvida pela IBM;
- Linguagem de consultas oficial padronizada pelo ANSI (*American National Standars Institute*);

SQL - Histórico

- Versão original IBM, chamava-se Sequel
  - implementada em 1970.
  - Evoluiu e passou a chamar-se SQL (Structured Query Language).
  - Existem vários produtos no mercado que suportam SQL (Oracle, SQL Server, Sybase).

SQL - Histórico

- 1986: American National Standards Institute (ANSI) e a International Standards Organization (ISO) publicaram padrões para SQL (SQL-86)
- 1987: IBM publica seus padrões - Systems Application Architecture Database Interface (SAA-SQL)
- Versão em uso: padrão ANSI/ISO SQL (SQL-92)

SQL

- Permite armazenamento, alteração e recuperação de dados de um banco de dados, criação de bancos de dados e dos objetos nele contidos;
- Especifica **O QUE** e não **COMO;**
- Dividida em 3 categorias
  - Data Definition Language (DDL);
  - Data Manipulation Language (DML);
  - Data Control Language (DCL).

SQL

- ***D**ata **D**efinition **L**anguage* (DDL)
- Utilizada para definir bancos de dados e os objetos nele contidos;
- Exemplos de comandos DDL:
  - CREATE TABLE;
  - ALTER TABLE;
  - DROP TABLE;
  - CREATE INDEX;
  - DROP INDEX;
  - CREATE PROCEDURE.

SQL

- **D**ata **M**anipulation **L**anguage (DML);
- Utilizada para manipular os objetos contidos no banco de dados;
- Exemplos de comandos DDL:
  - SELECT;
  - INSERT;
  - UPDATE;
  - DELETE.

SQL

- **D**ata **C**ontrol **L**anguage (DCL);
- Utilizada para permitir (ou negar) o acesso aos objetos contidos no banco de dados;
- Exemplos de comandos DCL:
  - GRANT;
  - REVOKE.

Data Definition Language (DDL)

SQL - DDL

- Alguns domínios aceitos por SQL-92:
  - char (n) varchar(n)
  - Int smallint
  - Numeric real, double precision, float(n)
  - date
  - time
- Domínios criados por usuário:
  - Create domain nome_pessoa char(20)

Bancos de Dados Relacionais

SQL - DDL

```
CREATE TABLE table_name
     (  definicao_coluna [, definicao_coluna...]  [constraints] )

DROP TABLE table_name
```

- definicao_coluna:
  col_name tipo_dado [NULL| NOT NULL]

- constraints:

```
 PRIMARY KEY (chave primária)
      [CONSTRAINT  constraint_name]  PRIMARY KEY
      [CLUSTERED|NONCLUSTERED] (col_name [,col_name2[,...]])
 UNIQUE (unicidade)
      [CONSTRAINT  constraint_name]  UNIQUE
      [CLUSTERED|NONCLUSTERED] (col_name [,col_name2[,...]])
 FOREIGN KEY (chave estrangeira)
      [CONSTRAINT  constraint_name]  FOREIGN KEY
      (col_name [,col_name2[,...]]) REFERENCES ref_table
                    (ref_col_name [,ref_col_name2[,...]])
```

SQL - DDL

```
CREATE TABLE aluno (
    num_matricula                   int             not null,
    nome_empregado                  char(30)        not null,
    endereco                        char(40)null,
    telefone                        char(15)null,
    codigo_curso        int                 not null,

        CONSTRAINT    pk_aluno
        PRIMARY KEY   NONCLUSTERED (num_matricula),

        CONSTRAINT    fk_aluno_curso
                FOREIGN KEY   (codigo_curso)
                REFERENCES curso (cod_curso)
)
```

SQL - DDL

```
ALTER TABLE nome_tabela
    [ADD ( { col_name  column_properties        [column_constraint] [[,]
    table_constraint]}  [,]  next_col_name } ... )
    | [DROP [CONSTRAINT]
                constraint_name [, constraint_name2]...]
```

- EXEMPLOS:

```
ALTER TABLE aluno ADD data_nascimento datetime null

ALTER TABLE aluno
CONSTRAINT          fk_aluno_curso
                FOREIGN KEY      (codigo_curso)
                REFERENCES curso (cod_curso)

ALTER TABLE aluno DROP CONSTRAINT fk_aluno_curso
```

Bancos de Dados Relacionais
SQL - DDL

CREATE [UNIQUE] [CLUSTERED|NONCLUESTERED]
INDEX index_name ON table_name
(column_name [, column_name] ...)

- EXEMPLO:
  CREATE INDEX ix_aluno02 ON aluno (codigo_curso)

DROP INDEX table_name. index_name

- EXEMPLO:
  DROP INDEX aluno.ix_aluno02

Data Manipulation Language (DML)

Bancos de Dados Relacionais

SQL - DML

- SELECT - Recuperação/ busca de dados
- INSERT - Inclusão de tuplas/registros
- UPDATE - Alteração do conteúdo dos dados
- DELETE - Exclusão de tuplas/registros

```
SELECT [DISTINCT] lista_colunas
[INTO [nome_nova_tabela]]
FROM {nome_tabela}
        [[,{nome_tabela}] ...]
[WHERE condição]
[GROUP BY clause]
[HAVING clause]
[ORDER BY clause]
```

Bancos de Dados Relacionais

SQL - DML

- Seleção simples

```
SELECT  num_matricula, nome
FROM aluno

SELECT * FROM aluno
```

- Seleção com filtro

```
SELECT   nome,  endereco
FROM     aluno
WHERE    codigo_curso = 1
AND      num_matricula > 100
```

SQL - DML

- Junções (Joins)
  - CROSS JOIN
    - Produto Cartesiano;
  - INNER JOIN
    - Igualdade entre os lados da comparação;
  - OUTER JOIN
    - Igualdade parcial entre os lados da comparação;
    - Pode ser left, right ou full outer join;
    - Quando não há igualdade, valores nulos são assumidos.

Aluno

cod_aluno	nome_aluno	cod_curso
1	Mario	52
2	Ana	51
3	Paula	52

Curso

cod_curso	nome_curso
51	Biologia
52	Computação
53	Direito

Cross Join

cod_aluno	nome_aluno	cod_curso	cod_curso	nome_curso
1	Mario	52	51	Biologia
1	Mario	52	52	Computação
1	Mario	52	53	Direito
2	Ana	51	51	Biologia
2	Ana	51	52	Computação
2	Ana	51	53	Direito
3	Paula	52	51	Biologia
3	Paula	52	52	Computação
3	Paula	52	53	Direito

Aluno

cod_aluno	nome_aluno	cod_curso
1	Mario	52
2	Ana	51
3	Paula	52

Curso

cod_curso	nome_curso
51	Biologia
52	Computação
53	Direito

Inner Join

cod_aluno	nome_aluno	cod_curso	cod_curso	nome_curso
1	Mario	52	52	Computação
2	Ana	51	51	Biologia
3	Paula	52	52	Computação
4	Carlos	52	52	Computação

Aluno

cod_aluno	nome_aluno	cod_curso
1	Mario	52
2	Ana	51
3	Paula	52

Curso

cod_curso	nome_curso
51	Biologia
52	Computação
53	Direito

Outer Join (right)

cod_aluno	nome_aluno	cod_curso	cod_curso	nome_curso
1	Mario	52	52	Computação
2	Ana	51	51	Biologia
3	Paula	52	52	Computação
4	Carlos	52	52	Computação
null	null	null	53	Direito

SQL - DML

- Junções (Joins)
  - CROSS JOIN

    select * from aluno, curso

    select * from aluno **cross join** curso

  - INNER JOIN

    select * from aluno A, curso C

    where A.cod_curso = C.cod_curso

    select * from aluno A **inner join** curso C

    on A.cod_curso = C.cod_curso

  - OUTER JOIN

    select * from aluno A **right outer join** curso C

    on A.cod_curso = C.cod_curso

SQL - DML

- Cláusula INTO
  - Cria uma nova tabela no banco de dados a partir do resultado do SELECT:

```
select   num_matricula, nome, endereco
INTO     nova_tabela_aluno
from     aluno
where    num_matricula > 100
```

- Cláusula DISTINCT
  - Especifica que apenas linhas distintas devem ser retornadas na consulta:

```
select   DISTINCT C.codigo_curso, C.nome_curso
from     aluno A, curso C
where    A.codigo_curso = C.codigo_curso
```

SQL - DML

- Funções de Agregação
  - SUM(expressao)
    - Somatório dos valores da expressão;
  - AVG(expressao)
    - Média aritmética dos valores da expressão;
  - COUNT(expressao)
    - Quantidade de valores da expressão;
  - COUNT(*)
    - Número de linhas selecionadas;
  - MAX(expressao)
    - Maior valor da expressão;
  - MIN(expressao)
    - Menor valor da expressão.
  - Observação
    - SUM, AVG, COUNT, MAX, e MIN ignoram valores nulos;
    - COUNT(*) não ignora linhas nulas.

SQL - DML

- Funções de Agregação (exemplos)

SELECT **COUNT(*)** FROM aluno

SELECT **MAX(salario), MIN(salario)**
FROM empregado

SELECT **COUNT (DISTINCT city)** FROM S

SELECT **AVG(salario)** FROM empregado
WHERE codigo_depto = 01

SELECT **SUM(QTY)** FROM SP

SQL - DML

- Cláusula GROUP BY
  - Especifica os campos que servirão de critério para definição de grupos de valores sumarizados através de funções de agregação;
  - Quando utilizadas sem uma cláusula GROUP BY, as funções de agregação retornam apenas um valor para a consulta realizada.
  - Exemplo:

```
SELECT      D.cod_depto, AVG(E.salario)
FROM        empregado E, departamento D
WHERE       E.cod_depto = D.cod_depto
GROUP BY    D.cod_depto
```

SQL - DML

- Cláusula HAVING
  - Filtro aplicado sobre o resultado das funções de agregação;
  - Restrição aplicada após a operação de agrupamento (GROUP BY);
  - Pode fazer referência a qualquer coluna que esteja na lista do SELECT;
  - Exemplo:

```
SELECT      D.cod_depto, AVG(E.salario)
FROM        empregado E, departamento D
WHERE       E.cod_depto = D.cod_depto
GROUP BY    D.cod_depto
HAVING      AVG(E.salario) > 1000
```

SQL - DML

- Cláusula ORDER BY
  - Ordena o resultado da consulta por colunas;
  - Permite a especificação de múltiplas colunas;
  - Colunas que não aparecem na lista do SELECT podem ser utilizadas;
  - Exemplo:

```
SELECT      D.cod_depto, AVG(E.salario)
FROM        empregado E, departamento D
WHERE       E.cod_depto = D.cod_depto
GROUP BY    D.cod_depto
HAVING      AVG(E.salario) > 1000
ORDER BY    AVG(E.salario) DESC
```

SQL - DML

- Operador UNION
  - Combina o resultado de duas ou mais consultas em um único resultado consistindo de todas as linhas que pertencem a todas as consultas no UNION;
  - EXEMPLO:

```
SELECT  data_lancamento, valor_lancamento
FROM    lancamento2000
UNION
SELECT  data_lancamento, valor_lancamento
FROM    lancamento2001
ORDER BY        valor_lancamento DESC
```

SQL - DML

- Operador LIKE
  - Caracteres especiais
    - “%” String com qualquer tamanho;
    - “_” Único caracter
  - Exemplo:

```
SELECT num_matricula, nome
FROM empregado WHERE nome LIKE ‘A%’

SELECT nome, endereco
FROM empregado
WHERE endereco LIKE ‘_a%’
```

SQL - DML

- Operador BETWEEN
  - Especifica um intervalo como critério, incluindo os valores determinados;
  - Exemplo:

```
SELECT num_matricula, nome
FROM empregado
WHERE num_matricula BETWEEN  10 AND 100
```

SQL - DML

- Sub-consultas

```
SELECT   e.num_matricula, e.nome
FROM     empregado e
WHERE    e. cod_departamento = 1
AND salario > (      SELECT AVG(salario)
                     FROM    empregado e
                     WHERE e.cod_departamento = 1    )
```

- Operadores IN / NOT IN
  - Comparam um valor com um conjunto válido explicitamente informado ou gerado através de uma consulta;
- Operadores EXISTS / NOT EXISTS
  - Funcionam como um teste de existência;
  - A sub-consulta retorna TRUE ou FALSE.

SQL - DML

- Operadores IN / NOT IN
  - Testa membros de um conjunto. Executa a sub-consulta para depois a consulta principal.

```
SELECT S#, SNAME FROM S
WHERE STATUS IN (10, 15, 20)

SELECT num_matricula, nome
FROM empregado
WHERE cod_departamento IN
        (SELECT cod_departamento FROM departamento
         WHERE nome_departamento = “Informatica”)

SELECT nome_departamento FROM departamento
WHERE cod_departamento NOT IN
        (SELECT cod_departamento FROM empregado)
```

SQL - DML

- Operadores EXISTS / NOT EXISTS
  - Executa a consulta e depois a sub-consulta (em cada registro da consulta da principal)

```
SELECT num_matricula, nome
FROM  empregado e
WHERE EXISTS
        (SELECT cod_departamento FROM departamento d
         WHERE e.cod_departamento = d.cod_departamento
        AND          nome_departamento = “Informatica”)

SELECT e.num_matricula, e.nome
FROM  empregado e, departamento d
WHERE e.cod_departamento = d.cod_departamento
AND     nome_departamento = “Informatica”

SELECT nome_departamento FROM departamento d
WHERE NOT EXISTS
        (SELECT cod_departamento FROM empregado e
        WHERE e.cod_departamento = d.cod_departamento)
```

SQL - DML

Inserção de registros (INSERT)

INSERT [INTO] {nome_tabela} [(lista_colunas)]
{lista_valores| comando_select}

- lista_colunas: Lista com os campos da tabela que serão incluídos;
- lista_valores: Lista de valores dos dados. Representada pela cláusula VALUES;
- comando_select: Comando SELECT cujo resultado será inserido na tabela em questão;

- Exemplos:

```
INSERT INTO aluno
VALUES (5, 'Paulo Andre', 'Rua A,  100 / 1001',
          '226-4099', 12)

INSERT INTO aluno (num_matricula, nome, cod_curso)
VALUES (5, 'Paulo Andre', 12)
```

SQL - DML

- Alteração de campos em registros existentes (UPDATE)

```
UPDATE  {nome_tabela}
      SET  { nome_coluna = {expressao| NULL}
      [[,] column_name = {expressao | NULL} ]
[FROM {relacao_tabelas}]
[WHERE condicao]
```

  - Exemplos:

```
UPDATE S
SET STATUS = 100
WHERE CITY = 'SSA'

UPDATE empregado
SET salario = salario * 2
WHERE cod_departamento = 1
```

SQL - DML

- Alteração de campos em registros existentes (UPDATE)
  - Exemplos:

```
UPDATE empregado
SET salario = (SELECT AVG(salario) FROM empregado)
WHERE salario IS NULL

UPDATE empregado
SET salario = cargo.salario
FROM cargo
WHERE empregado.cod_cargo = cargo.cod_cargo

UPDATE departamento
SET status = 0
WHERE cod_departamento IN
        ( SELECT cod_departamento FROM empregado)
```

SQL - DML

- Exclusão de registros (DELETE)

```
DELETE  {nome_tabela}
   [FROM {relacao_tabelas}]
   [WHERE condicao]
```

- Exemplos:

```
DELETE S
WHERE CITY = 'SSA'

DELETE empregado
FROM   departamento d
WHERE  d.nome = 'Informatica'
AND empregado.cod_departamento = d.cod_departamento
```

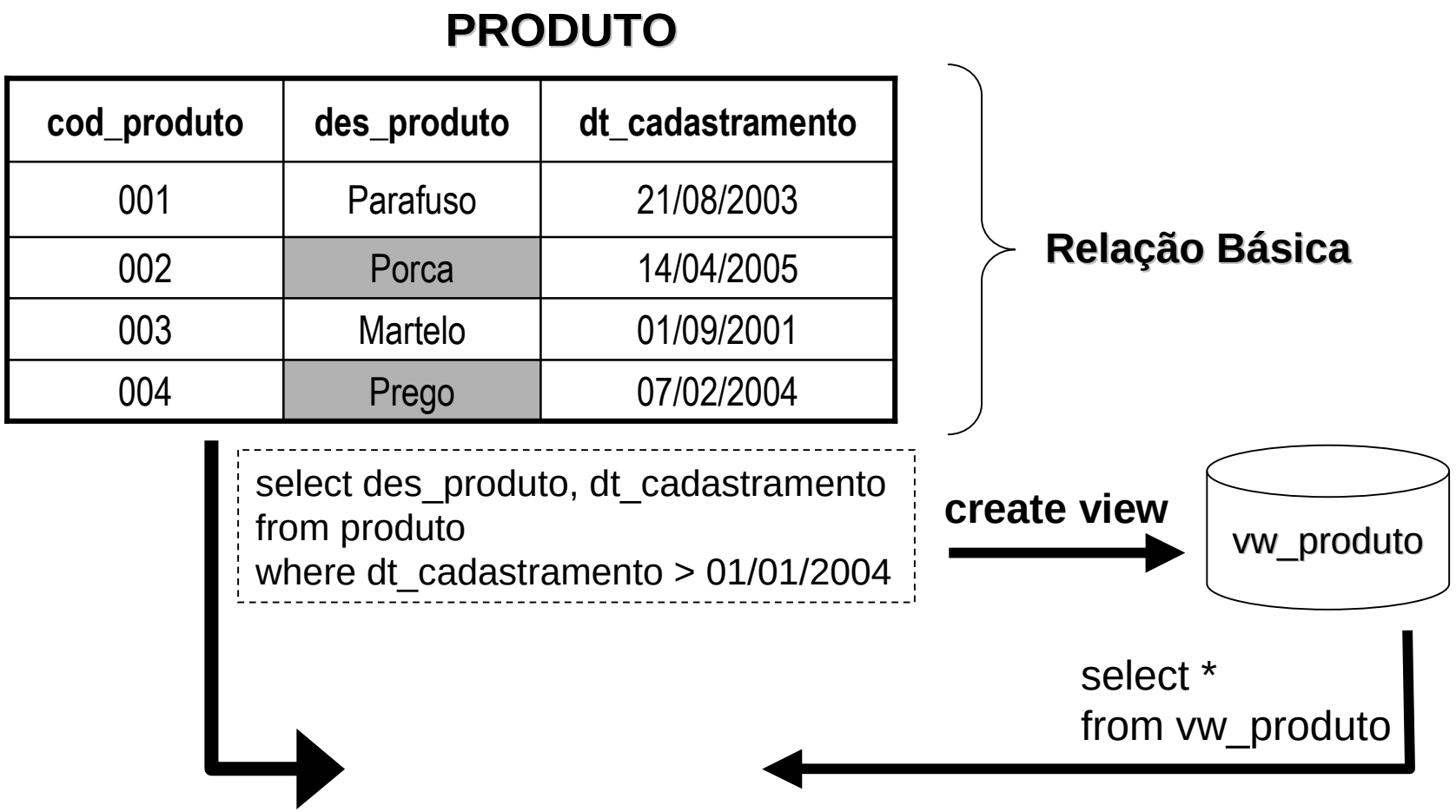
Catálogo do Sistema

- Conjunto de informações detalhadas sobre os objetos do sistema (metadados)
  - Tabelas, índices, usuários, restrições de integridade, etc;
  - Nos bancos de dados relacionais o catálogo é implementado através de tabelas e pode ser consultado da mesma forma que as tabelas de usuário.

Visões

- Relações básicas;
- Relações derivadas
  - Obtida a partir da aplicação de uma expressão relacional sobre uma ou mais relações básicas;
- Visões
  - Relação derivada e nomeada;
  - O valor da visão é fruto da execução de uma expressão relacional em um dado instante;
  - A expressão relacional que define a visão é registrada no momento da criação da visão.

Visões


PRODUTO
cod_produto
des_produto
dt_cadastramento
001
Parafuso
21/08/2003
002
Porca
14/04/2005
003
Martelo
01/09/2001
004
Prego
07/02/2004
Relação Básica
select des_produto, dt_cadastramento
from produto
where dt_cadastramento > 01/01/2004
create view
vw_produto
select *
from vw_produto

Visões

- Sintaxe:
  - Criação

    CREATE VIEW <nome_visao>
    AS <consulta_sql>

  - Exclusão

    DROP VIEW <nome_visao>

  - Exemplos:

    CREATE VIEW vw_funcionario
    AS
    select F.nome_funcionario
    from funcionario F
    where F.cod_departamento = ‘1’

Tarefa Extra-Classe

- Recupere os exercícios de SQL e faça-os todos.

Banco de Dados I

5 – Linguagens de Consulta

Grinaldo Lopes de Oliveira (grinaldo@gmail.com)
Curso Superior de Tecnologia em
Análise e Desenvolvimento de Sistemas