Fundamentos de Banco de Dados

SQL Básico

Márcio Silva

marcio.inacio@ufms.br

SQL Básico

- Linguagem SQL
  - Considerada uma das maiores razões para o sucesso comercial dos bancos de dados relacionais.
- SQL
  - **Structured Query Language**
  - Comandos para definição de dados, consultas e atualizações (DDL e DML).
  - Padrões SQL mais recentes (a partir do SQL:1999) possuem:
    - Especificação núcleo (core specification).
    - Extensões especializadas.

Definição de dados e de tipos de dados

- Terminologia:
  - Tabela, linha e coluna usados para os termos relação, tupla e atributo do modelo relacional.
  - Comando CREATE
    - Principal comando SQL para definição de dados.
    - Pode ser usado para criar esquemas, tabelas (relações) e domínios (bem como outras construções como views, assertions e triggers).

Esquema

- Esquema SQL
  - Objetivo: agrupar tabelas e outras construções que pertencem à mesma aplicação de banco de dados.
  - Identificado por um nome de esquema.
  - Inclui um identificador de autorização e descritores para cada elemento.
- Elementos de esquema incluem:
  - Tabelas, restrições, visões, domínios e outras construções como concessões (*grants*) de autorização.
- Cada comando SQL termina com um ponto e vírgula.

- Comando CREATE SCHEMA:
  - CREATE SCHEMA EMPRESA AUTHORIZATION 'Jsilva';

Catálogo

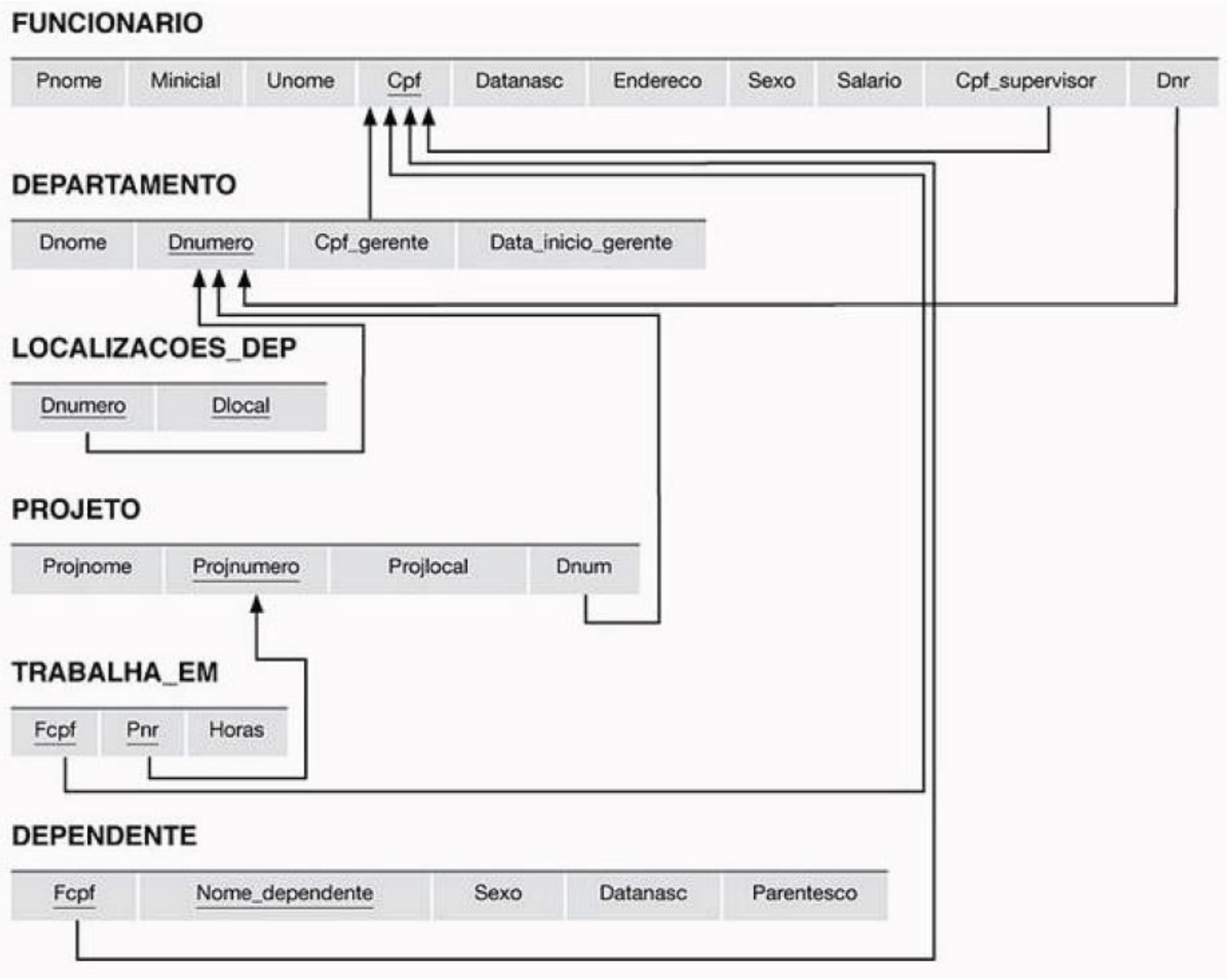
- Coleção nomeada de esquemas em um ambiente SQL.
- Ambiente SQL
  - Instalação de um SGBDR compatível com SQL em um sistema de computador.

Comando CREATE TABLE

- Especifica uma nova relação.
  - Fornece um nome.
  - Especifica atributos e restrições iniciais.
- Pode opcionalmente especificar esquema:
  - CREATE TABLE EMPRESA.FUNCIONARIO ...

  ou
  - CREATE TABLE FUNCIONARIO
- Tabelas da base (relações da base)
  - Relações e suas tuplas são realmente criadas e armazenadas como um arquivo pelo SGBD.
- Relações virtuais
  - Criadas através da instrução CREATE VIEW.

Comando CREATE TABLE



Comando CREATE TABLE

```
CREATE TABLE FUNCIONARIO
        (Pnome          VARCHAR(15)         NOT NULL,
        Minicial        CHAR,
        Unome           VARCHAR(15)         NOT NULL,
        Cpf             CHAR(11),           NOT NULL,
        Datanasc        DATE,
        Endereço        VARCHAR(30),
        Sexo            CHAR,
        Salario         DECIMAL(10,2),
        Cpf_supervisor  CHAR(11),           NOT NULL,
        Dnr             INT
    PRIMARY KEY (Cpf),
    FOREIGN KEY (Cpf_supervisor) REFERENCES FUNCIONARIO(Cpf),
    FOREIGN KEY (Dnr) REFERENCES DEPARTAMENTO(Dnumero) );
```

Comando CREATE TABLE

```
CREATE TABLE DEPARTAMENTO
        ( Dnome                 VARCHAR(15)             NOT NULL,
          Dnumero               INT                     NOT NULL,
          Cpf_gerente           CHAR(11),               NOT NULL,
          Data_inicio_gerente   DATE,
        PRIMARY KEY (Dnumero),
        UNIQUE (Dnome),
        FOREIGN KEY (Cpf_gerente) REFERENCES FUNCIONARIO(Cpf) );
CREATE TABLE LOCALIZACAO_DEP
        ( Dnumero               INT                     NOT NULL,
          Dlocal                VARCHAR(15)             NOT NULL,
        PRIMARY KEY (Dnumero, Dlocal),
        FOREIGN KEY (Dnumero) REFERENCES DEPARTAMENTO(Dnumero) );
```

Comando CREATE TABLE

```
CREATE TABLE PROJETO
        ( Projnome          VARCHAR(15)          NOT NULL,
        Projnumero          INT                  NOT NULL,
        Projlocal           VARCHAR(15),
        Dnum                INT                  NOT NULL,
      PRIMARY KEY (Projnumero),
      UNIQUE (Projnome),
      FOREIGN KEY (Dnum) REFERENCES DEPARTAMENTO(Dnumero) );
CREATE TABLE TRABALHA_EM
        ( Fcpf              CHAR(9)              NOT NULL,
        Pnr                 INT                  NOT NULL,
        Horas               DECIMAL(3,1)         NOT NULL,
      PRIMARY KEY (Fcpf, Pnr),
      FOREIGN KEY (Fcpf) REFERENCES FUNCIONARIO(Cpf),
      FOREIGN KEY (Pnr) REFERENCES PROJETO(Projnumero) );
```

Comando CREATE TABLE

```
CREATE TABLE DEPENDENTE
        ( Fcpf                  CHAR(11),               NOT NULL,
        Nome_dependente  VARCHAR(15)             NOT NULL,
        Sexo                    CHAR,
        Datanasc                DATE,
        Parentesco              VARCHAR(8),
    PRIMARY KEY (Fcpf, Nome_dependente),
    FOREIGN KEY (Fcpf) REFERENCES FUNCIONARIO(Cpf) );
```

Comando CREATE TABLE

- Algumas chaves estrangeiras podem causar erros.
  - Especificadas por referências circulares.
  - Ou porque dizem respeito a uma tabela que ainda não foi criada.
  - Exemplo:
    - A chave estrangeira Cpf_supervisor na tabela FUNCIONARIO é uma referência circular, pois se refere à própria tabela.
    - A chave estrangeira Dnr na tabela FUNCIONARIO se refere à tabela DEPARTAMENTO, que ainda não foi criada.
  - Solução: omitir essas restrições no comando CREATE TABLE, e depois acrescentar usando a instrução ALTER TABLE.

Adicionando uma chave estrangeira após a criação da tabela

```
alter table FUNCIONARIO
  ADD FOREIGN KEY(Dnr) REFERENCES DEPARTAMENTO(Dnumero);
```

PostgreSQL - Instalação

- Linux / Ubuntu
  - sudo apt-get install postgresql pgadmin3

SGBD – Cliente PostgreSQL – Resumo

- Alterar a senha de um usuário pelo console:
  - Após acessar o psql, usar o comando:
    - \password <nome_do_usuario>
    - Exemplo: \password postgres
- Estabelecer uma conexão pelo console:
  - psql -U postgres -W -h localhost
    - -U ☐ usuário / -W ☐ pedir password (senha) / -h ☐ host
- Criar um banco de dados
  - create database nome_bd;
- Usar (conectar) um banco de dados como padrão
  - \c nome_bd;
- Apagar um banco de dados
  - drop database nome_bd;

SGBD – Cliente PostgreSQL – Resumo

- Obtendo informações sobre o BD...
  - Mostrar todos os bancos de dados
    - \l
  - Mostrar todas as tabelas de um BD
    - \d;
  - Mostrar detalhes de uma tabela
    - \d nome_tabela
  - Mostrar esquemas
    - \dn
- Ajuda
  - \?
    - Ajuda sobre comandos do cliente (terminal)
  - \h
    - Ajuda sobre comandos SQL
- Sair (quit) do cliente (terminal)
  - \q

Alguns tipos de dados (SQL 92)

- char(n)
  - string de caracteres de tamanho fixo n
- varchar(n)
  - string de caracteres de tamanho variável (máximo n)
- Integer (int), smallint, tinyint, bigint
- decimal(p,d), numeric (p,d)
  - numérico com p dígitos
  - Dos p dígitos, d dígitos representam casas decimais após a vírgula
- Real, float - numérico ponto flutuante
- date – data
- time – hora
- datetime, smalldatetime, timestamp – data e hora

Tipos de dados de atributos e domínios

- Tipos de dados básicos
  - **Numeric**
    - Números inteiros: INTEGER, INT, and SMALLINT
    - Números de ponto-flutuante (reais): FLOAT or REAL, and DOUBLE PRECISION
  - **Cadeia de caracteres / String**
    - Tamanho fixo: CHAR(n), CHARACTER(n)
    - Tamanho variável: VARCHAR(n), CHAR VARYING(n), CHARACTER VARYING(n)

Tipos de dados de atributos e domínios

- Tipos de dados básicos (continuação)
  - **Cadeia de bits**
    - Tamanho fixo: BIT(*n*)
    - Tamanho variável: BIT VARYING(*n*)
  - **Boolean**
    - Valores TRUE ou FALSE ou NULL
  - **DATE**
    - Dez posições
    - Componentes são YEAR, MONTH e DAY na forma YYYY-MM-DD

Tipos de dados de atributos e domínios

- Tipos de dados adicionais
  - **Timestamp**
    - Inclui os campos DATE e TIME
    - Além de um mínimo de 6 posições para frações decimais de segundos
    - Qualificador WITH TIME ZONE opcional
  - **INTERVAL**
    - Especifica um valor relativo que pode ser usado para incrementar ou decrementar um valor absoluto de uma data, hora ou timestamp.

Tipos de dados de atributos e domínios

- Domínio
  - Um domínio pode ser declarado e seu nome, usado com a especificação de atributo.
  - Torna mais fácil mudar o tipo de dado para um domínio que é usado por diversos atributos em um esquema.
  - Melhora a legilibilidade do esquema.
  - Domínios podem não estar disponíveis em algumas especificações de SQL.
  - Exemplo:
    - CREATE DOMAIN TIPO_CPF AS CHAR(11);

Tipos de dados de atributos e domínios

- Domínio

```
CREATE DOMAIN cep AS TEXT
CHECK(
   VALUE ~ '^\d{5}$'
OR VALUE ~ '^\d{5}-\d{3}$'
);
```

```
create domain positive_int as
int check (value > 0);
```

Restrições (Constraints)

- Restrições básicas
  - Restrições de chave e de integridade referencial.
  - Restrições sobre domínios de atributos e NULLs.
  - Restrições sobre tuplas individuais de uma relação.

Restrições (Constraints)

- NOT NULL
  - O atributo deve ser obrigatoriamente preenchido.
  - NULL não é permitido para um determinado atributo.
- Valor padrão (DEFAULT)
  - Atribui um valor padrão ao atributo, caso não seja especificado um valor.
  - DEFAULT <value>
- CHECK
  - Verifica se o valor inserido é permitido para o atributo.
  - `Dnumero INT NOT NULL CHECK (Dnumero > 0 AND Dnumero < 21);`

Restrições (Constraints)

- PRIMARY KEY (PK)
  - Especifica um ou mais atributos para fazer parte da chave primária da relação.
  - Dnumero INT PRIMARY KEY;

- UNIQUE
  - Especifica chaves alternativas (secundárias).
  - Garante que o atributo não terá valores repetidos na tabela.
  - Dnome VARCHAR(15) UNIQUE;

Restrições (Constraints)

- FOREIGN KEY (FK)
  - Implementa o conceito de chave estrangeira e garante a integridade referencial.
  - Deve referenciar um campo que possua chave primária ou uma restrição UNIQUE.
  - Operação padrão: rejeitar a atualização na violação.
  - Conecta uma cláusula de ação de disparo referencial.
    - Opções incluem SET NULL, CASCADE e SET DEFAULT
    - Ação tomada pelo SGBD para SET NULL ou SET DEFAULT é a mesma para ON DELETE e ON UPDATE
    - Opção CASCADE adequada para:
      - Relações de “relacionamento”, como TRABALHA_EM.
      - Relações que representam atributos multivalorados, como LOCALIZACAO_DEP.
      - Relações que representam entidades fracas, como DEPENDENTE.

Dando nomes a restrições

- Palavra-chave CONSTRAINT
  - Nomeia uma restrição.
  - Útil para posterior alteração.

Restrições (Constraints)

```
CREATE TABLE FUNCIONARIO
    ( . . . .
    Dnr                 INT             NOT NULL      DEFAULT 1,
    CONSTRAINT CHPFUNC
        PRIMARY KEY (Cpf),
    CONSTRAINT CHESUPERFUNC
        FOREIGN KEY (Cpf_supervisor) REFERENCES FUNCIONARIO(Cpf)
        ON DELETE SET NULL     ON UPDATE CASCADE,
    CONSTRAINT CHEDEPFUNC
    FOREIGN KEY(Dnr) REFERENCES DEPARTAMENTO(Dnumero)
            ON DELETE SET DEFAULT ON UPDATE CASCADE);
```

Restrições (Constraints)

```
CREATE TABLE DEPARTAMENTO
    ( . . . ,
    Cpf_gerente     CHAR(11)     NOT NULL     DEFAULT '88866555576',
    . . . ,
    CONSTRAINT CHPDEP
        PRIMARY KEY(Dnumero),
    CONSTRAINT CHSDEP
        UNIQUE (Dnome),
    CONSTRAINT CHEGERDEP
    FOREIGN KEY (Cpf_gerente) REFERENCES FUNCIONARIO(Cpf)
        ON DELETE SET DEFAULT ON UPDATE CASCADE);
```

Restrições (Constraints)

```
CREATE TABLE LOCALIZACAO_DEP
    ( . . . .
    PRIMARY KEY (Dnumero, Dlocal),
    FOREIGN KEY (Dnumero) REFERENCES DEPARTAMENTO(Dnumero)
        ON DELETE CASCADE    ON UPDATE CASCADE);
```

Especificando restrições em tuplas usando CHECK

- Cláusulas CHECK no final de uma instrução CREATE TABLE.
  - Aplicada a cada tupla individualmente.
  - `CHECK (Dep_data_criacao <= Data_inicio_gerente);`

Remoção de tabelas

- DROP TABLE nome-tabela [CASCADE | RESTRICT]
- Remove as tuplas da tabela e sua definição do catálogo
  - CASCADE remove as restrições do tipo *foreign key* tabelas que referenciam a tabela removida

Exercício

- Dado o modelo relacional a seguir, defina o esquema físico do banco de dados usando SQL.
- clientes(id: int, nome: varchar(50), cpf: char(11), data_cadastro:date, cidade: varchar(50), uf: char(2))
- categorias(id: int, nome: varchar(20))
- classes (id:int, nome: varchar(20), preco: decimal(10,2))
- distribuidores (id: int, nome: varchar(50))
- filmes (id: int, titulo: varchar(50), id_distribuidor: int, ano_lancamento: int(4), id_categoria: int, id_classe: int)
  - id_distribuidor referencia distribuidores
  - id_categoria referencia categorias
  - id_classe referencia classes
- locacoes (id: int, id_cliente: int, id_filme: int, dt_locacao: date, dt_devolucao_prevista: date, dt_devolucao:date, valor: decimal(10,2)
  - id_cliente referencia clientes
  - id_filme referencia filmes

Criando a tabela clientes

```
CREATE TABLE clientes (
    id SERIAL,
    cpf char(11),
    nome varchar(50),
    data_cadastro date,
    cidade varchar(40),
    uf char(2) DEFAULT 'CE',
    PRIMARY KEY (id),
    UNIQUE (cpf)
);
```

Criando a tabela clientes

```
CREATE TABLE clientes (
    id SERIAL PRIMARY KEY,
    cpf char(11) UNIQUE,
    nome varchar(50),
    data_cadastro date,
    cidade varchar(40),
    uf char(2) DEFAULT 'CE'
);
```

Serial implicitamente usa uma sequence do PostgreSQL

```
CREATE SEQUENCE clientes_id_seq;

CREATE TABLE clientes (
     id int DEFAULT nextval('clientes_id_seq'),
     cpf char(11),
     nome varchar(50),
     data_cadastro date,
     cidade varchar(40),
     uf char(2) DEFAULT 'CE',
     PRIMARY KEY (id),
     UNIQUE (cpf)
);
```

Inserindo dados na tabela clientes

- `insert into clientes values (default, '123', 'Regis', '2012-04-11', 'Fortaleza', 'CE');`
- `insert into clientes (cpf, nome, data_cadastro, cidade) values ('124', 'João', '2012-04-11', 'Quixadá');`
- `insert into clientes (uf, nome, cidade) values ('CE', ’João', 'Quixadá'), ('CE', 'Maria', 'Fortaleza'), ('CE', 'José', 'Jucas');`

Alteração de tabelas

- ALTER TABLE nome-tabela
  [ADD nome-coluna tipo de dados]
  [DROP nome-coluna]
  [ADD CONSTRAINT nome-restrição]
  [DROP CONSTRAINT nome-restrição]
  [DROP PRIMARY KEY]
  [repetir ADD ou DROP em qualquer ordem]

Consultas básicas em SQL

- Instrução SELECT
  - Instrução básica para recuperar informações de um BD.
- SQL permite que uma tabela tenha 2 ou mais tuplas idênticas em todos os seus valores de atributos.
  - Isso não é permitido no modelo relacional.
  - Comportamento multiconjunto (bag) de tuplas.

Consultas básicas em SQL

- Forma básica da instrução SELECT:

```
SELECT   <lista atributos>
FROM     <lista tabelas>
WHERE    <condição>;
```

- <lista atributos> é uma lista de nomes de atributo cujos valores devem ser recuperados pela consulta.
- <lista tabelas> é uma lista dos nomes de relação exigidos para processar a consulta.
- <condição> é uma expressão condicional (booleana) que identifica as tuplas a serem recuperadas pela consulta.

Consultas básicas em SQL

- Operadores de comparação lógica
  - =, <, <=, >, >=, <>
- Atributos de Projeção
  - Atributos cujos valores serão recuperados.
- Condição de seleção
  - Condição booleana que deve ser verdadeira para qualquer tupla recuperada.

Consultas básicas em SQL

Consulta 0. Recuperar a data de nascimento e o endereço do(s) funcionário(s) cujo nome seja 'João B. Silva'.

Consultas básicas em SQL

Consulta 1. Recuperar o nome e endereço de todos os funcionários que trabalham para o departamento 'Pesquisa'.

```
SELECT pnome, endereco
FROM funcionario f, departamento d
WHERE f.dnr = d.dnumero
    AND dnome='Pesquisa';
```

Consultas básicas em SQL

Consulta 2. Para cada projeto localizado em 'Mauá', liste o número do projeto, o número do departamento que o controla e o sobrenome, endereço e data de nascimento do gerente do departamento.

Consultas básicas em SQL

FUNCIONARIO

Pnome	Minicial	Unome	Cpf	Datanasc	Endereco	Sexo	Salario	Cpf_supervisor	Dnr
João	B	Silva	12345678966	09-01-1965	Rua das Flores, 751, São Paulo, SP	M	30.000	33344555587	5
Fernando	T	Wong	33344555587	08-12-1955	Rua da Lapa, 34, São Paulo, SP	M	40.000	88866555576	5
Alice	J	Zelaya	99988777767	19-01-1968	Rua Souza Lima, 35, Curitiba, PR	F	25.000	98765432168	4
Jennifer	S	Souza	98765432168	20-06-1941	Av. Arthur de Lima, 54, Santo André, SP	F	43.000	88866555576	4
Ronaldo	K	Lima	66688444476	15-09-1962	Rua Rebouças, 65, Piracicaba, SP	M	38.000	33344555587	5
Joice	A	Leite	45345345376	31-07-1972	Av. Lucas Obes, 74, São Paulo, SP	F	25.000	33344555587	5
André	V	Pereira	98798798733	29-03-1969	Rua Timbira, 35, São Paulo, SP	M	25.000	98765432168	4
Jorge	E	Brito	88866555576	10-11-1937	Rua do Horto, 35, São Paulo, SP	M	55.000	NULL	1

DEPARTAMENTO

Dnome	Dnumero	Cpf_gerente	Data_inicio_gerente
Pesquisa	5	33344555587	22-05-1988
Administração	4	98765432168	01-01-1995
Matriz	1	88866555576	19-06-1981

LOCALIZACAO_DEP

Dnumero	Dlocal
1	São Paulo
4	Mauá
5	Santo André
5	Itu
5	São Paulo

Consultas básicas em SQL

TRABALHA_EM

Fcpf	Pnr	Horas
12345678966	1	32,5
12345678966	2	7,5
66688444476	3	40,0
45345345376	1	20,0
45345345376	2	20,0
33344555587	2	10,0
33344555587	3	10,0
33344555587	10	10,0
33344555587	20	10,0
99988777767	30	30,0
99988777767	10	10,0
98798798733	10	35,0
98798798733	30	5,0
98765432168	30	20,0
98765432168	20	15,0
88866555576	20	NULL

PROJETO

Projnome	Projnumero	Projlocal	Dnum
ProdutoX	1	Santo André	5
ProdutoY	2	Itu	5
ProdutoZ	3	São Paulo	5
Informatização	10	Mauá	4
Reorganização	20	São Paulo	1
Novosbeneficios	30	Mauá	4

DEPENDENTE

Fcpf	Nome_dependente	Sexo	Datanasc	Parentesco
33344555587	Alicia	F	05-04-1986	Filha
33344555587	Tiago	M	25-10-1983	Filho
33344555587	Janaína	F	03-05-1958	Esposa
98765432168	Antonio	M	28-02-1942	Marido
12345678966	Michael	M	04-01-1988	Filho
12345678966	Alicia	F	30-12-1988	Filha
12345678966	Elizabeth	F	05-05-1967	Esposa

Nomes de atributos ambíguos

- O mesmo nome pode ser usado para dois ou mais atributos.
  - Desde que os atributos estejam em relações diferentes.
  - Deve-se qualificar o nome do atributo com o nome da relação para evitar ambiguidade.

```
C1': SELECT FUNCIONARIO.Pnome, FUNCIONARIO.
        UNome FUNCIONARIO.Endereco,
     FROM   FUNCIONARIO, DEPARTAMENTO
     WHERE  DEPARTAMENTO.DNome='Pesquisa' AND
        DEPARTAMENTO.Dnumero=
        FUNCIONARIO.Dnr;
```

Apelido, renomeação e variáveis de tupla

- Apelidos ou variáveis de tupla
  - Declare nomes de relação alternativos

FUNCIONARIO **AS** F(Pn, Mi, Un, Cpf, Dn, End, Sexo, Sal, Scpf, Dnr)

Cláusula WHERE não especificada

- Ausência da cláusula WHERE
  - Indica que não há condição sobre a seleção de tuplas.
- Produto cartesiano
  - Todas as combinações de tuplas possíveis.

Consultas 9 e 10. Selecionar todos os Cpfs de FUNCIONARIO (C9) e todas as combinações de Cpf de FUNCIONARIO e Dnome de DEPARTAMENTO (C10) no banco de dados.

Uso do asterisco (*)

- Recupera todos os valores de atributos das tuplas selecionadas.

```
C1C:  SELECT  *
      FROM    FUNCIONARIO
      WHERE   Dnr=5;
C1D:  SELECT  *
      FROM    FUNCIONARIO, DEPARTAMENTO
      WHERE   Dnome='Pesquisa' AND Dnr=Dnumero;
C10A: SELECT  *
      FROM    FUNCIONARIO, DEPARTAMENTO;
```

Tabelas como conjuntos em SQL

- SQL não elimina automaticamente tuplas duplicadas em resultados de consultas.
- Use a palavra-chave DISTINCT na cláusula SELECT.
  - Somente tuplas distintas deverão permanecer no resultado.

Consulta 11. Recuperar o salário de cada funcionário (C11) e todos os valores de salário distintos (C11A).

```
C11:    SELECT    ALL Salario
        FROM      FUNCIONARIO;
```

```
C11A:   SELECT    DISTINCT Salario
        FROM      FUNCIONARIO;
```

Consultas

- **SELECT [ALL | DISTINCT] {* | expr [[AS] c_alias]**
  {, expr [[AS] c_alias] ... }}
  FROM nome-tabela [[AS] qualificador]
  {, nome-tabela [[AS] qualificador] ...}
  WHERE predicado
- ALL
  - Retorna todas as tuplas, inclusive repetidas (default)
- DISTINCT
  - Retorna apenas tuplas não repetidas
- *
  - Retorna todos os atributos da(s) tabela(s)
- expr
  - Representa um atributo ou
  - Expressão matemática envolvendo atributos das tabelas
    - salario*1.40

Tabelas como conjuntos em SQL

- Operações de conjunto
  - UNION, EXCEPT / MINUS (diferença), INTERSECT.
    - Tuplas duplicadas são eliminadas do resultado.
- Operações multiconjuntos correspondentes
  - UNION ALL, EXCEPT ALL, INTERSECT ALL.
  - Seus resultados são multiconjuntos (duplicatas não são eliminadas).

Tabelas como conjuntos em SQL

Consulta 4. Fazer uma lista de todos os números de projeto para aqueles que envolvam um funcionário cujo último nome é 'Silva', seja como um trabalhador ou como um gerente do departamento que controla o projeto.

```
C4A:  ( SELECT    DISTINCT Projnumero
        FROM      PROJETO, DEPARTAMENTO,
                  FUNCIONARIO
        WHERE     Dnum=Dnumero AND
                  Cpf_gerente=Cpf
                  AND Unome='Silva' )
        UNION
      ( SELECT    DISTINCT Projnumero
        FROM      PROJETO, TRABALHA_EM,
                  FUNCIONARIO
        WHERE     Projnumero=Pnr AND Fcpf=Cpf
                  AND Unome='Silva' );
```

Consultas

- Consultas com o operador de união
  - UNION
    - União de duas relações (consultas)
      - Sem repetições
  - UNION ALL
    - União de duas relações
      - Com repetições
  - Exemplo
    - Considere as seguintes relações
      - Empregado(matr, nome, ender, dt_nasc, cpf, salário, lotação)
      - Dependente(nome_dep, data-nasc, matr_resp)
    - Liste o nome e data de nascimento de todos os funcionários e dependentes existentes na empresa, ordenando-os por nome:

```
select nome,dt_nasc from Empregado
UNION
select nome_dep,data_nasc from Dependente;
```

Consultas

- Consultas com o operador de interseção
  - INTERSECT
    - Interseção entre duas relações (consultas)
      - Sem repetições
  - INTERSECT ALL
    - Interseção entre duas relações
      - Com repetições
- Consultas com o operador de diferença
  - EXCEPT (o Oracle usa o operador MINUS)
    - Diferença entre duas relações (consultas)
      - Sem repetições
  - EXCEPT ALL
    - Diferença entre duas relações (consultas)
      - Com repetições
- O **MySQL** não tem suporte a **INTERSECT** e **EXCEPT**.

Combinação de padrão de subcadeias e operadores aritméticos

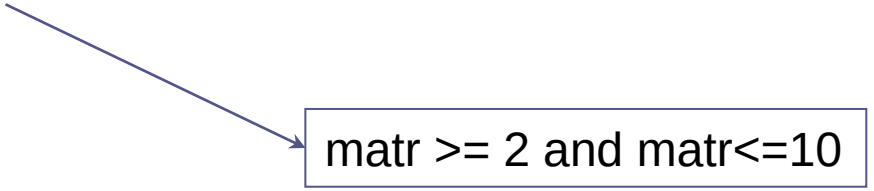
- Operador de comparação LIKE
  - Usado para combinação de padrão de cadeia.
  - % substitui um número arbitrário de zero ou mais caracteres.
  - Underline (_) substitui um único caractere.
- Operador de concatenação: ||
- Operadores aritméticos padrão
  - Adição (+), subtração (-), multiplicação (*) e divisão (/).
- Operador de comparação BETWEEN.
- Operadores para incremento (+) e decremento (-) de data, hora ou timestamp.

Consultas

- Predicados com operações sobre *strings*
  - Identificação de padrão
    - % - Casa com qualquer substring
    - _ - Casa com qualquer caracter
  - Operador
    - **like**
  - Exemplos
    - nome like ‘inf%’
      - Retorna strings que iniciam pelo substring inf
    - nome like ‘%si_’
      - Retorna strings que contenham ‘si’ como substring e terminem com um caracter qualquer após ‘si’
    - Listar todos empregados com sobrenome ‘pires’
      - Select nome from empregados where nome like ‘%pires%’

Consultas

- Predicado Between
  - Sintaxe
    - expr1 [NOT] BETWEEN expr2 and expr3
  - Exemplo
    - select * from empregados
      where matr between 2 and 10



Ordem dos resultados da consulta

- Cláusula ORDER BY
  - Palavra-chave DESC para ver o resultado em uma ordem decrescente de valores.
  - Palavra-chave ASC para especificar explicitamente a ordem crescente.

```
ORDER BY D.Dnome DESC, F.Unome ASC, F.Pnome
ASC
```

Consultas

- FROM
  - Representa o produto cartesiano das tabelas referenciadas
- WHERE
  - Corresponde ao predicado de seleção da álgebra relacional
- **ORDER BY** coluna-resultado [ASC | DESC]
  {, coluna-resultado [ASC | DESC] ...}

Discussão e resumo das consultas de recuperação

```
SELECT      <lista atributos>
FROM        <lista tabelas>
[ WHERE     <condição> ]
[ ORDER BY  <lista atributos> ];
```

Instruções INSERT, DELETE e UPDATE

- Três comandos usados para modificar o banco de dados:
  - INSERT, DELETE e UPDATE.

Comando INSERT

- Especifica o nome da relação e uma lista de valores para a tupla.

```
U1:   INSERT INTO FUNCIONARIO
      VALUES     ( 'Ricardo', 'K', 'Marini',
                 '65329865388', '30-12-
                 1962', 'Rua Itapira, 44,
                 Santos, SP', 'M', 37.000,
                 '65329865388', 4 );
```

```
U1A:  INSERT INTO   FUNCIONARIO (Pnome,
                    Unome, Dnr, Cpf)
      VALUES        ('Ricardo', 'Marini', 4,
                    '65329865388');
```

Comando INSERT

- Inserção do resultado de uma consulta.

```
create table nomes_funcionarios (
   id serial,
   nome_completo varchar(50),
   primary key(id)
);
```

```
INSERT INTO nomes_funcionarios (nome_completo)
   SELECT pnome || ' ' || minicial || ' ' || unome
   FROM funcionario;
```

Comando INSERT

- Inserção do resultado de uma consulta.

```
U3A: CREATE TABLE TRABALHA_EM_INFO
     ( Func_nome      VARCHAR(15),
       Proj_nome      VARCHAR(15),
       Horas_semanal  DECIMAL(3,1) );
U3B: INSERT INTO    TRABALHA_EM_INFO
                    ( Func_nome, Proj_nome,
                    Horas_por_semana )
     SELECT         F.Unome, P.Projnome,
                    T.Horas
     FROM           PROJETO P, TRABALHA_
                    EM T, FUNCIONARIO F
     WHERE          P.Projnumero=T.Pnr AND
                    T.Fcpf=F.Cpf;
```

Comando DELETE

- Remove tuplas de uma relação
  - Inclui uma cláusula WHERE para selecionar as tuplas que serão removidas.

```
U4A:  DELETE FROM   FUNCIONARIO
      WHERE         Unome='Braga';
U4B:  DELETE FROM   FUNCIONARIO
      WHERE         Cpf='12345678966';
U4C:  DELETE FROM   FUNCIONARIO
      WHERE         Dnr=5;
U4D:  DELETE FROM   FUNCIONARIO;
```

Comando UPDATE

- Modifica valores de atributos de uma ou mais tuplas selecionadas.
- Cláusula SET adicional no comando UPDATE
  - Especifica atributos a serem modificados e novos valores.

```
U5:  UPDATE   PROJETO
     SET      Projlocal = 'Santo André', Dnum
              = 5
     WHERE    Projnumero=10;
```

```
U6:  UPDATE   FUNCIONARIO
     SET      Salario = Salario * 1,1
     WHERE    Dnr = 5;
```

Referências

- Elsmari, R., Navathe, Shamkant B. “Sistemas de Banco de Dados”. 6ª Edição, Pearson Brasil, 2011.
- Silberschatz, A., Korth, H., Sudarshan, S. “Sistema de Banco de Dados”. 5ª Edição, Editora Campus, 2006.
- Slides Prof. José Maria (UFC).