Introdução ao Banco de Dados PostGreSQL

Diogo Cezar Teixeira Batista

diogocezar@utfpr.edu.br

Universidade Tecnológica Federal do Paraná
Campus Cornélio Procópio
UTFPR-CP

Cornélio Procópio - 2008

Agenda I

Agenda II

- Entidades e conjuntos de entidades
- Atributos
  - Atributos Compostos e Multivalorados
- Identificadores
  - Super Chave
  - Chave Candidata
  - Chave Primária
- Relacionamentos
- Grau de Relacionamento
  - Cardinalidade do Mapeamento: 1:1
  - Cardinalidade do Mapeamento: 1:N
  - Cardinalidade do Mapeamento: N:N

3 Postgres

- ACID
- Alguns tipos de dados

Agenda III

- Funcionalidades Avançadas
- Linguagem de Definição de Dados(DDL)
  - Create Table
  - Alter Table
  - Drop Table
  - Comment
  - Create Index
- Linguagem de Modificação de Dados (DML)
  - Insert
  - Update
  - Delete
  - Truncate

4 Subconsultas e Junções

- Exists
- In e Not In

Agenda IV

- Junções
  - Inner Join
  - Left Join
  - Right Join

Questionário inicial

- O que você entende por banco de dados?
- Qual a diferença entre dado e informação?
- Quais os bancos de dados conhecidos?
- Qual a vantagem de se utilizar um banco de dados?

Realidade das empresas que não utilizam BD

- Cada setor possui seus aplicativos e seus dados:
  - Planilhas de texto, documentos textuais, arquivos.
- O acesso aos dados é gerenciado por cada aplicação:
  - Editores de texto, aplicações descentralizaras.
- Não há compartilhamento;
- Dados dependem dos programas, sendo manipulados apenas por estes.

Problemas

- Dados duplicados;
- Dificuldade de manutenção;
- Atualização? envio de relatórios, ofícios ou memorandos?
- Falta de padronização.

Processamento de arquivos: por que não é viável?

- *Inconsistência ou redundância*:
- *Dificuldade de acesso aos dados*: necessidade de novas aplicações;
- *Isolamento de dados*: dados dispersos em vários arquivos (difícil recuperação);
- *Problemas de integridade*: novas restrições da aplicação, exemplo: mudança da maioridade, novas taxas, impostos;
- *Problemas de atomicidade*: quando ocorrer alguma falha na escrita, deve-se voltar ao estado anterior. Exemplo: transação bancária;
- *Acesso concorrente*: dois usuários alteram o valor de uma mesma variável;
- *Problemas de segurança*: autorização a usuários.

Surgimento dos BD's

- Necessidade de armazenar e recuperar grande quantidade de informação de maneira organizada;
- Cria-se um modelo da realidade através de dados representativos e suas relações (Diagrama de Entidades e Relacionamentos).

O que é um Bando de Dados?

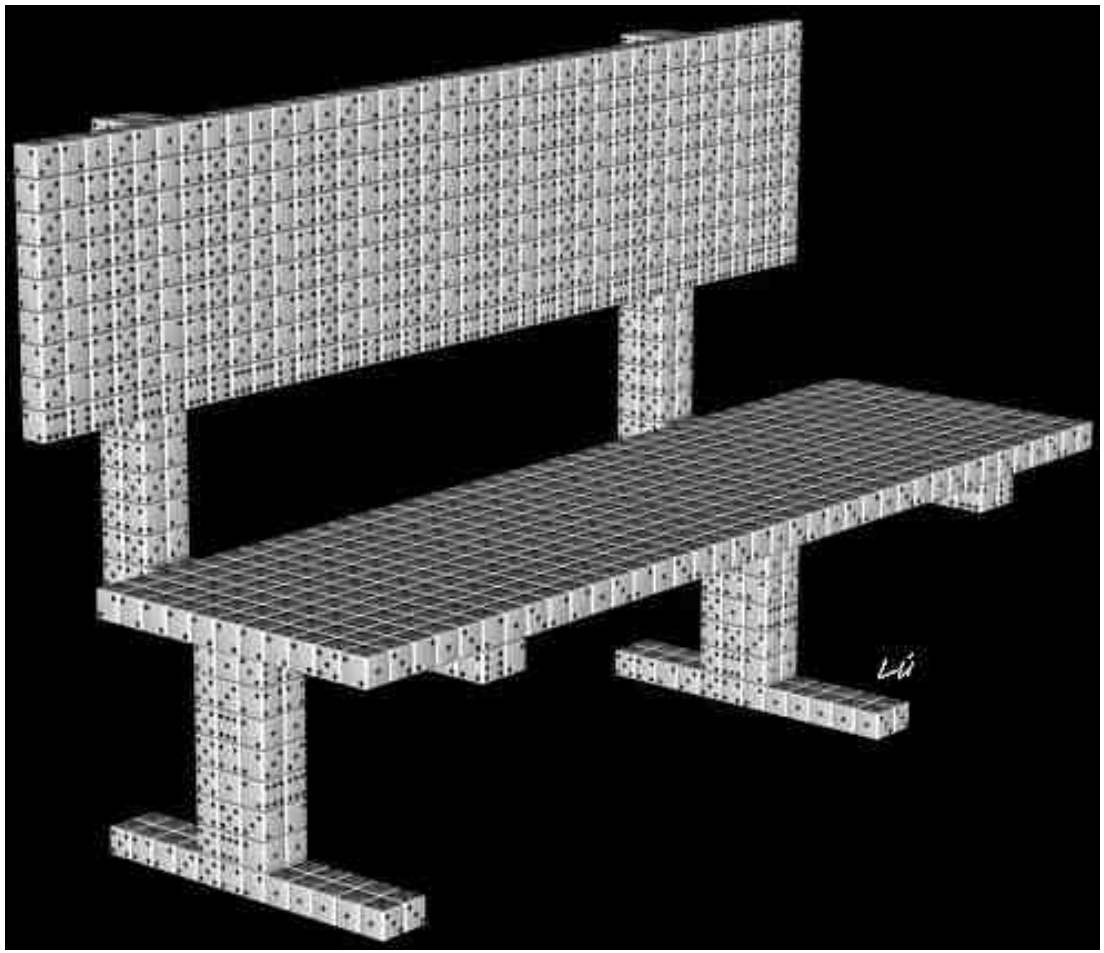


Figura: Um sólido banco de dados

O que é um Bando de Dados?

- "Uma coleção de dados relacionados" [ELMASRI & NAVATHE];
- "Conjunto de dados integrados que tem por objetivo atender a uma comunidade específica" [HEUSER];
- "Um conjunto de dados" [SILBERSCHATZ];

Componentes de um Sistema de Banco de Dados

- *Dados*: diferente de informação;
- *Hardware*: parte física (armazenamento);
- *Software*: SGBD (Sistema Gerenciador de Banco de Dados);
- *Usuários*: programadores de aplicação, DBA, usuário final;

Vantagens

- *Controle centralizados dos dados*: visão geral, relacionamento das informações, entre outros.
- Que relação é essa?
  - Exemplo: Agenda(nome, telefone, endereço e aniversário); Fiado(nome, saldo devedor);
    Situação: Enviar cartões de aniversário para os meus clientes exceto para os inadimplentes.

Características

- *Eliminação de redundância*: duplicidade, perda de espaço;
- *Controle de consistência*: armazenamento em um único local;
- *Compartilhamento*: questão de concorrência;
- *Restrições de segurança*: nível de acesso por usuário;
- *Integridade*: precisão dos dados, validação.

Abstração

- *Nível físico*: armazenamento;
- *Nível lógico*: quais dados são armazenados e qual a relação entre eles?

DDL e DML

- *DDL*: Linguagem de definição de dados;
  - Define a estrutura (esquema) de um BD;
  - Compilação de definições: dicionário de dados (dados sobre dados: metadados).
- *DML*: Linguagem de modelagem de dados;
  - Permite o acesso ou manipulação dos dados no BD;
  - Recuperação, inserção, exclusão e modificação dos dados.

Suporte a Sistemas Operacionais

	Windows	Mac OS X	Linux	BSD	Unix
DB2	Sim	Não	Sim	Não	Sim
Firebird	Sim	Sim	Sim	Sim	Sim
HSQLDB	Sim	Sim	Sim	Sim	Sim
Informix	Sim	Sim	Sim	Sim	Sim
Ingres	Sim	?	Sim	?	Sim
InterBase	Sim	Não	Sim	Não	Sim
Adabas	?	?	?	?	?
MaxDB	Sim	Não	Sim	Não	Sim
Microsoft SQL Server	Sim	Não	Não	Não	Não
MySQL	Sim	Sim	Sim	Sim	Sim
Oracle	Sim	Sim	Sim	Não	Sim
PostgreSQL	Sim	Sim	Sim	Sim	Sim
SQLite	Sim	Sim	Sim	Sim	Sim

Características principais

	ACID*	Integridade Referencial	Transação	Unicode
DB2	Sim	Sim	Sim	Sim
Firebird	Sim	Sim	Sim	Sim
HSQLDB	Sim	Sim	Sim	?
Informix	Sim	Sim	Sim	Sim
Ingres	Sim	Sim	Sim	Sim
InterBase	Sim	Sim	Sim	Sim
MaxDB	Sim	Sim	Sim	Sim
Microsoft SQL Server	Sim	Sim	Sim	Sim
MySQL	Sim	Sim	Sim	Sim
Oracle	Sim	Sim	Sim	Sim
PostgreSQL	Sim	Sim	Sim	Sim
SQLite	Sim	Não	Básico	Sim

* ACID é a abreviação de Atomicidade, Concistência, Isolação e Durabilidade

Figura: Características principais

Modelos de Banco de Dados

- Modelo Lógico:
  - Modelo de dados abstrato;
  - Independente do SGBD;
  - Abordagem ER (Entidades e Relacionamentos) através de um DER (Diagrama de Entidades e Relacionamentos);
- Modo Físico:
  - Descrição do BD no nível de abstração visto pelo usuário;
  - Dependente do tipo;

Entidades e conjuntos de entidades

Caracterização

- Entidade: objeto que existe e que é distinguível de outros objetos $\Rightarrow$ Pode ser concreta ou abstrata.
- Conjunto de entidades: Conjunto de todas as entidades de um mesmo tipo.

Funcionários	Datas	Livros	Departamentos

Entidades e conjuntos de entidades

Caracterização

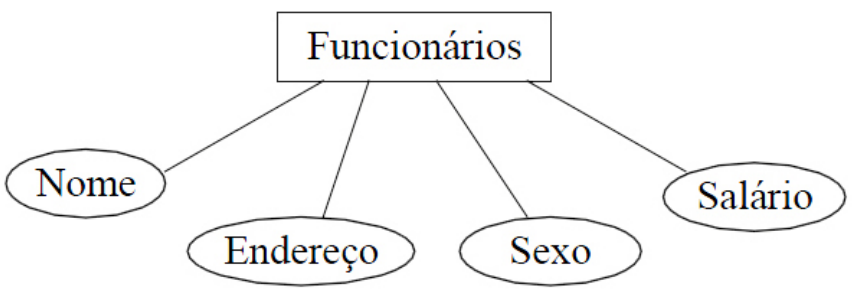
- Entidade: objeto que existe e que é distinguível de outros objetos $\Rightarrow$ Pode ser concreta ou abstrata.
- Conjunto de entidades: Conjunto de todas as entidades de um mesmo tipo.

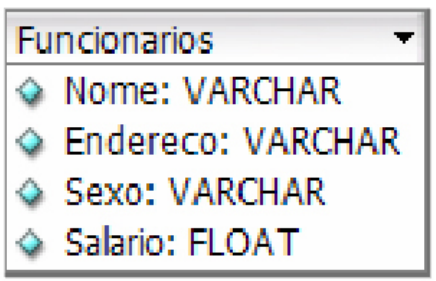
Funcionários	Datas	Livros	Departamentos

Atributos

Caracterização

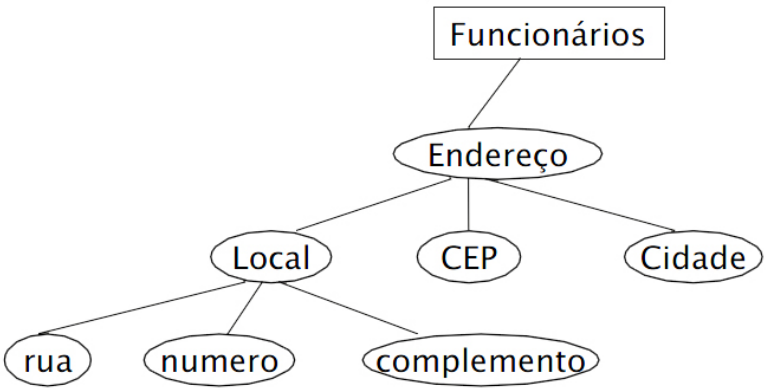
Permitem a associação de informações pertinentes aos "objetos" do mundo real.



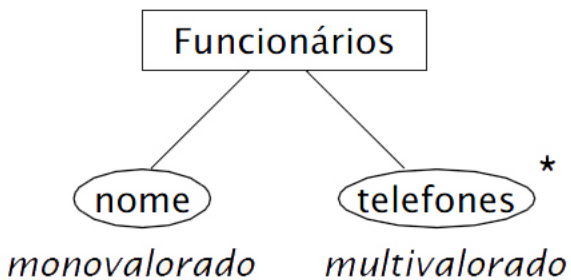


Atributos Compostos e Multivalorados

Atributos compostos:



Atributos multivalorados:



Identificadores

Caracterização

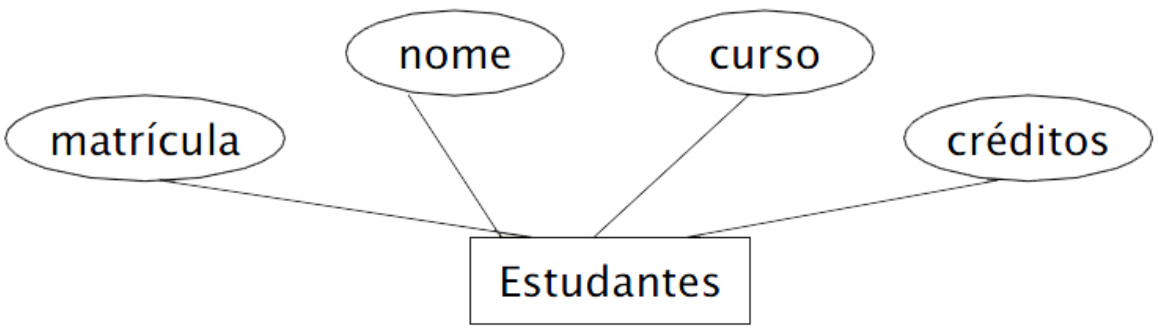
Atributos que fornecem um nome ou identificam as instâncias da entidade.

- Único. Ex: num_matrícula_funcionario;
- Não único. Ex: nome_funcionario;
- Composto. Ex: nome, num_matricula;

Superchave

Caracterização

Uma superchave é um conjunto de um ou mais atributos que identifica unicamente uma entidade.

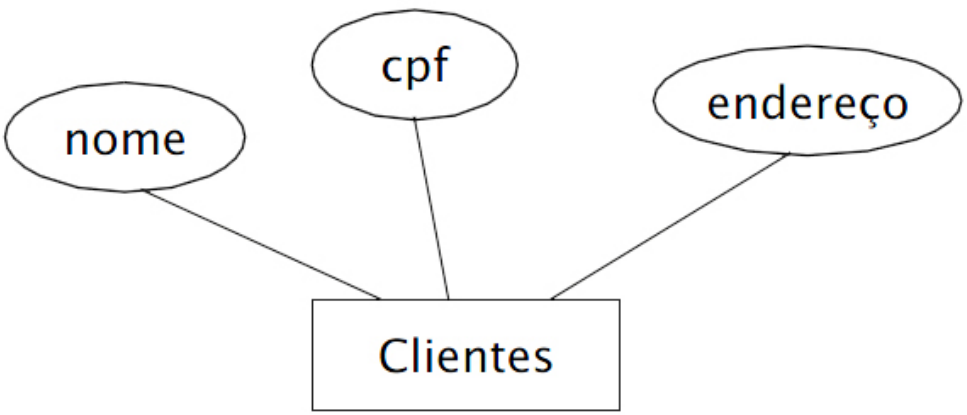


Superchaves:
{matrícula}
{matrícula, nome}
...

Chave Candidata

Caracterização

Uma chave candidata é uma superchave que não contem nenhum subconjunto próprio que seja por si só uma superchave.



Chaves candidatas:

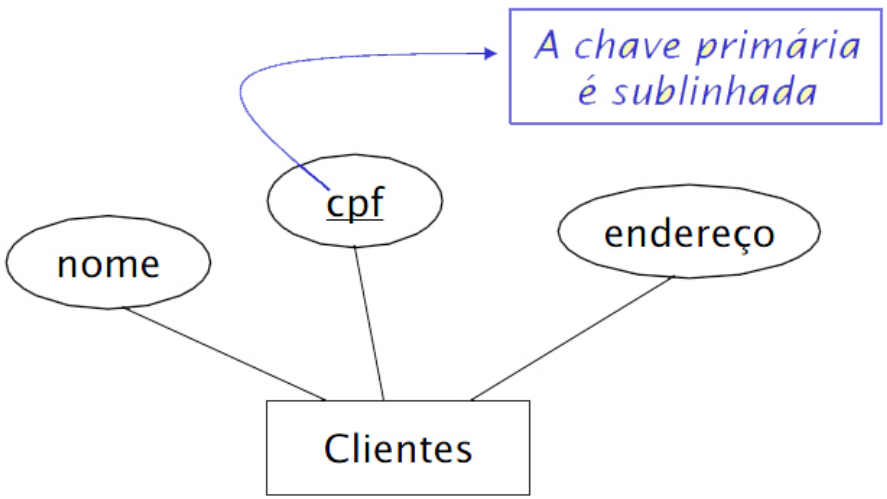
{ cpf}

{nome, endereço}

Chave Primária

Caracterização

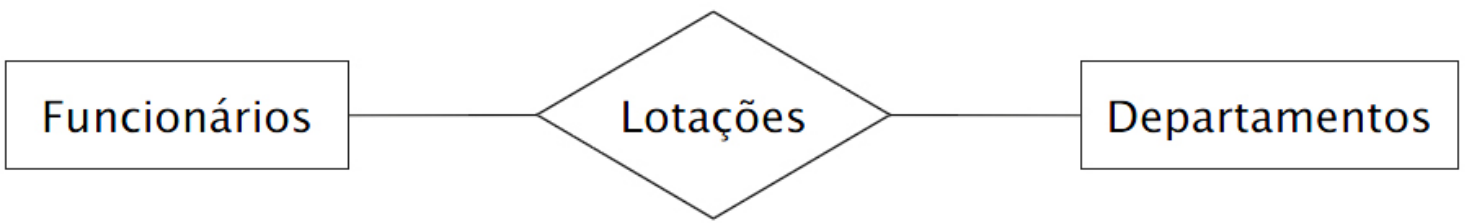
Chave candidata escolhida pelo projetista do BD como principal meio de identificação de cada uma das entidades de um conjunto de entidades.



Relacionamentos

Caracterização

Associação entre várias entidades.



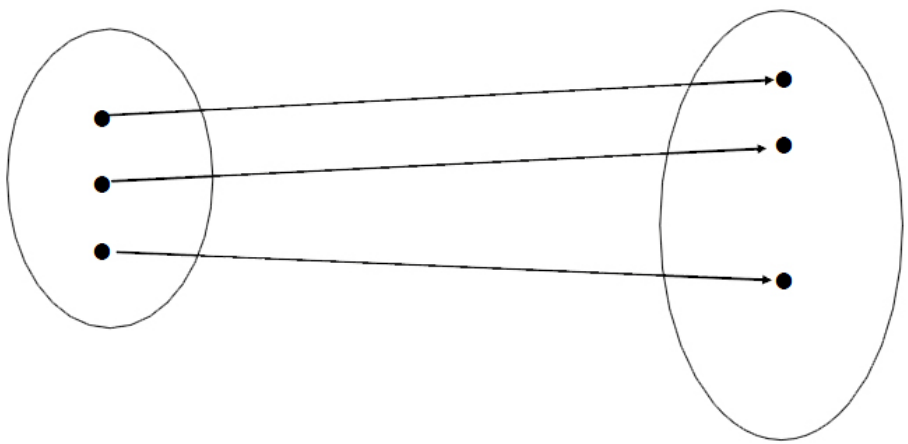
Cardinalidade do Mapeamento

- Um-para-um (1:1);
- Um-para-muitos (1:n);
- Muitos-para-muitos (n:n);

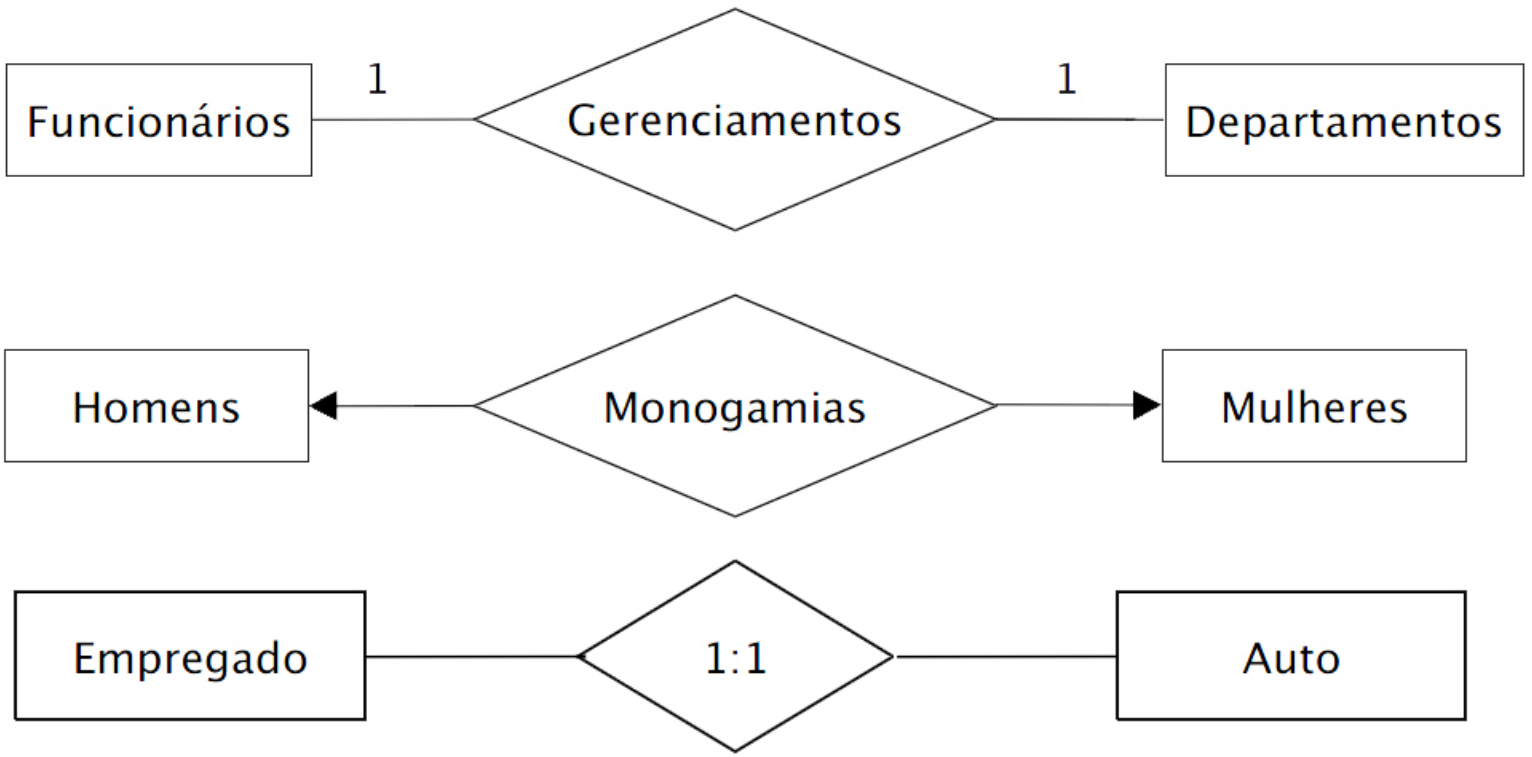
Cardinalidade do Mapeamento: 1:1

Caracterização

Uma entidade em A é associada com no máximo uma entidade em B e vice-versa.



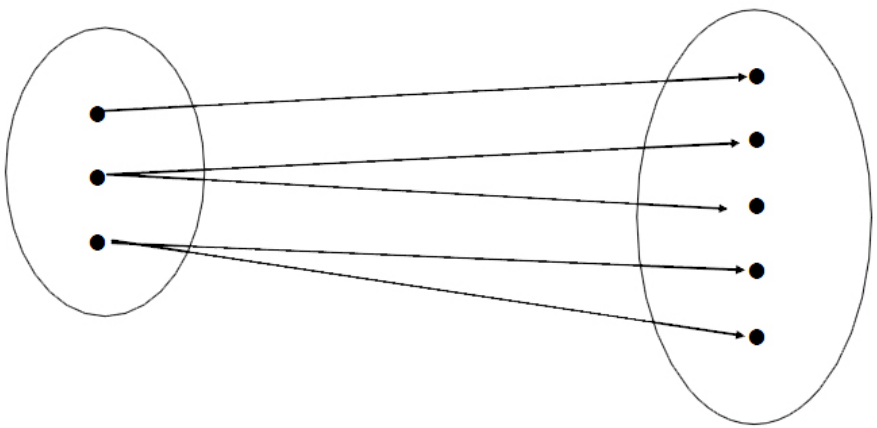
Diagramas de Exemplo



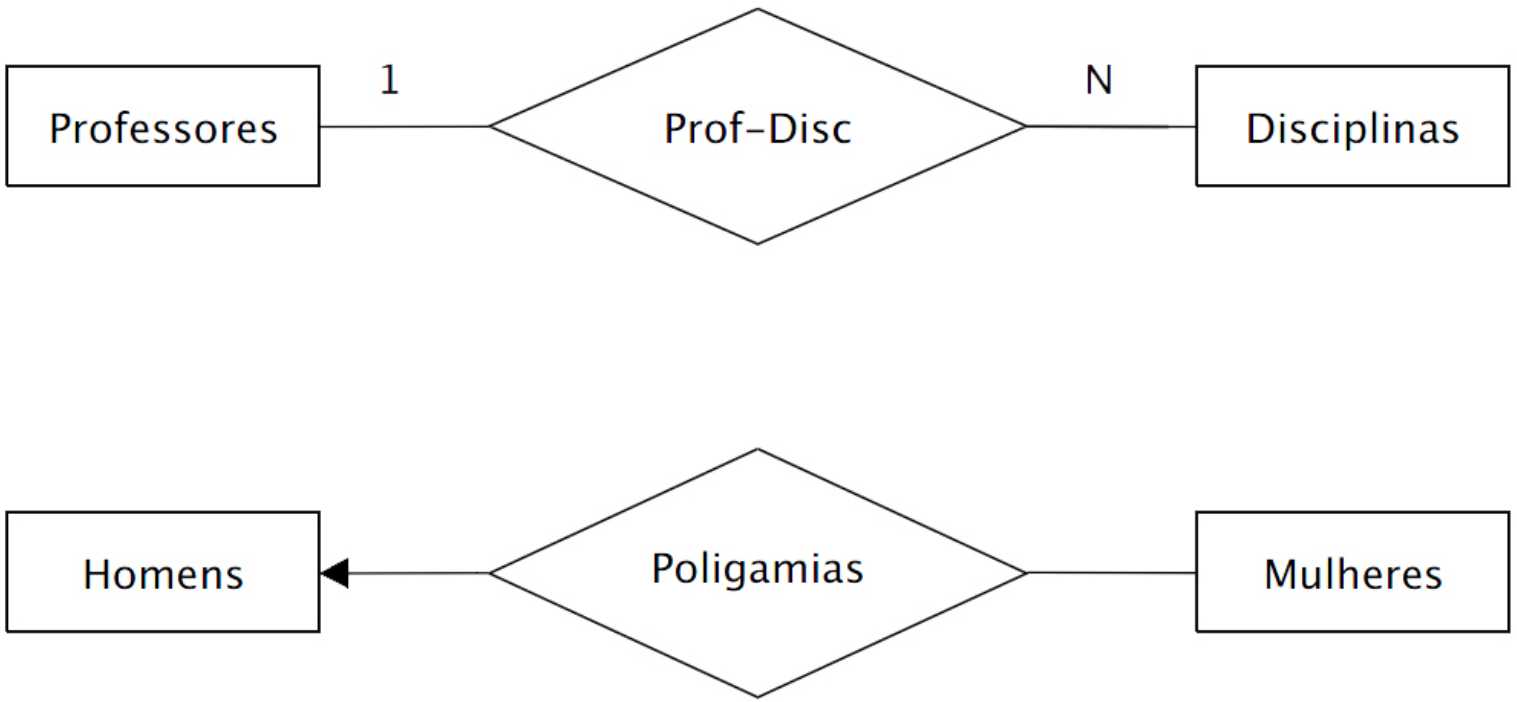
Cardinalidade do Mapeamento: 1:N

Caracterização

Uma entidade de A é associada a qualquer número de entidades de B, uma entidade de B, por outro lado, pode estar associada somente a uma entidade de A.



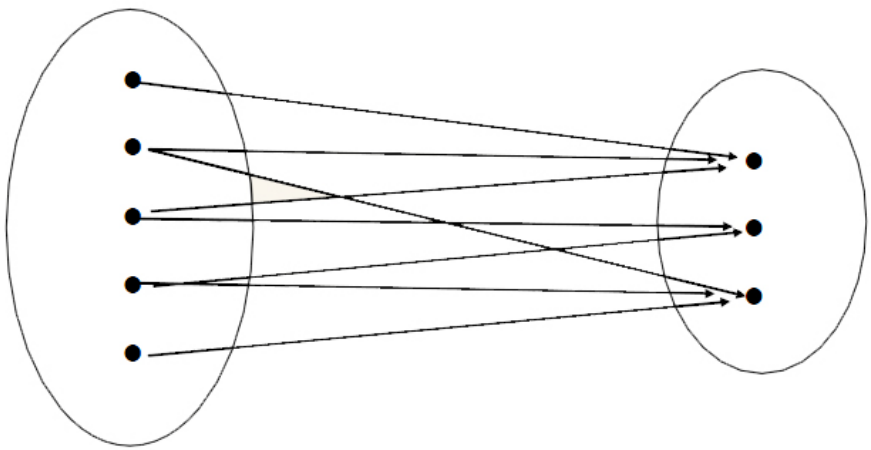
Diagramas de Exemplo



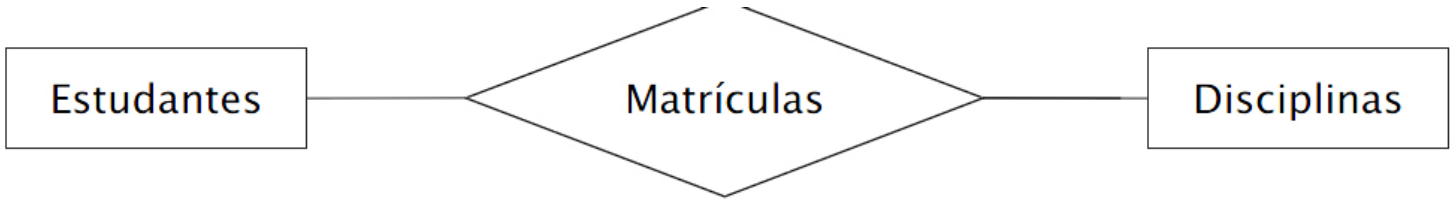
Cardinalidade do Mapeamento: N:N

Caracterização

Uma entidade de A pode estar associada a qualquer número de entidades de B e vice-versa.



Diagramas de Exemplo



Uma breve história do PostGreSQL

- Banco de dados objeto-relacional;
- Derivado do pacote POSTGRES (Universidade da Califórnia em Berkeley);
  - 1986 início do projeto;
  - 1987 primeira versão do Postgres;
  - 1989 liberação para usuários restritos da versão 1;
  - 1991 versão 3 com as principais funcionalidades atuais;
  - 1993 versão 4.2, última lançada pela Berkeley;
  - 1994 Andrew Yu e Jolly Chen criaram a versão conhecida como Postgre95 com interpretador para a linguagem SQL;
  - 1997 Nome do projeto muda para PostgreSQL, a versão 6 é lançada;
  - 2000 versão 7 lançada com suporte a Foreign Key;
  - 2005 versão 8 lançada com versão nativa (sem uso do CYGWIN) para Windows, TABLESPACES, SAVEPOINTS, POINTINTIMERECOVERY. etc.

ACID

- *Atomicidade*: transações não podem ficar pela metade (tudo ou nada);
- *Consistência*: transações devem transformar um estado consistente do banco em outro estado consistente;
- *Isolamento*: transações são isoladas umas das outras, elas não "enxergam" dados gravados por transações concorrentes;
- *Durabilidade*: uma vez efetuada a transação os dados devem permanecer no banco.

Alguns tipos de dados

Tipo	Nome	Aliases	Descrição
Binário	bytea		dados binários ("matriz de bytes")
Boleano	boolean	bool	booleano lógico (verdade/falso)
Caracter	text		cadeia de caracteres de comprimento variável
Caracter	character [(n)]	char [(n)]	cadeia de caracteres de comprimento fixo
Caracter	character varying [(n)]	varchar [(n)]	cadeia de caracteres de comprimento variável [c]
Data e hora	timestamp [(p)] with time zone	timestamptz	data e hora, incluindo a zona horária
Data e hora	timestamp [(p)] [without time zone]		data e hora
Data e hora	time [(p)] with time zone	timetz	hora do dia, incluindo a zona horária
Data e hora	time [(p)] [without time zone]		hora do dia
Data e hora	date		data de calendário (ano, mês,dia)

Figura: Alguns tipos de dados

Alguns tipos de dados II

Monetário	money		quantia monetária (não recomendado)
Numérico	serial	serial4	inteiro de quatro bytes com auto-incremento
Numérico	smallint	int2	inteiro de dois bytes com sinal
Numérico	real	float4	número de ponto flutuante de precisão simples
Numérico	numeric [(p, s)]	decimal [(p, s)]	numérico exato com precisão selecionável
Numérico	integer	int, int4	inteiro de quatro bytes com sinal
Numérico	double precision	float8	número de ponto flutuante de precisão dupla [d]
Numérico	bigserial	serial8	inteiro de oito bytes com auto-incremento
Numérico	bigint	int8	inteiro de oito bytes com sinal [a]

Figura: Alguns tipos de dados

No PostgreSQL você pode:

- Criar validações de dados;
- Criar tipos de dados personalizados;
- Disparar *stored procedures* a partir de eventos das tabelas;
- Dividir o banco de dados em esquemas;
- Alocar bases inteiras, esquemas ou tabelas em locais diferentes (disco ou rede);

Componentes do PostgreSQL:

- Esquemas;
- Herança;
- Views;
- Funções;
- Restrições;
- Gatilhos;

Create Table I

Utilizado para criar tabelas.

Código 1: Sintaxe da linguagem DDL para criação de tabelas

```
1  CREATE TABLE nome_da_tabela (
2          { <nome_da_coluna> <tipo_de_dado> [ DEFAULT
               expressão_padrão ] [[ restrição_de_coluna ] | [
               restrição_de_tabela]]}
3  )[WITH OIDS | WITHOUT OIDS ]
4  onde restrição_de_coluna é:
5  [ CONSTRAINT nome_da_restrição ]
6  { NOT NULL | NULL | UNIQUE [ USING INDEX TABLESPACE
       espaço_de_tabelas ] | PRIMARY KEY [ USING INDEX
       TABLESPACE espaço_de_tabelas ] |
7  CHECK (expressão) | REFERENCES tabela_referenciada [ (
       coluna_referenciada ) ] }
8  e restrição_de_tabela é:
9  [ CONSTRAINT nome_da_restrição ]
```

Create Table II

```
10  { UNIQUE ( nome_da_coluna [, ... ] ) [ USING INDEX
        TABLESPACE espaço_de_tabelas ] | PRIMARY KEY (
        nome_da_coluna [, ... ] ) [ USING INDEX TABLESPACE
11  espaço_de_tabelas ] | CHECK ( expressão ) | FOREIGN KEY (
        nome_da_coluna [, ... ] )
12  REFERENCES tabela_referenciada [ ( coluna_referenciada [,
        ... ] ) ] }
```

Usando corretamente o Create Table

Código 2: Boa prática usando Create Table

```
1  CREATE TABLE fatura (
2          id_fatura integer,
3          total numeric(9,2) NOT NULL,
4          desconto numeric(6,2),
5          id_cliente integer NOT NULL,
6          id_vendedor integer NOT NULL,
7          data timestamp(0) DEFAULT current_timestamp(0),
8          CONSTRAINT fatura_pk PRIMARY KEY (id_fatura),
9          CONSTRAINT fatura_cliente_fk FOREIGN KEY id_cliente
               REFERENCES cliente,
10         CONSTRAINT fatura_vendedor_fk FOREIGN KEY
               id_vendedor REFERENCES vendedor
11 ) WHITHOUT OIDS;
```

Usando o Create Table sem convenção

Código 3: Má prática usando Create Table

```
1  create table Faturas
2  (IdFatura integer primary key,
3  Total money not null,
4  Desconto numeric(6,2),
5  IdCliente integer not null references Cliente,
6  IdVendedor integer not null references Vendedor,
7  Data timestamp default now());
```

Alter Table I

Utilizado para alterar tabelas.

Código 4: Sintaxe da linguagem DDL para alteração de tabelas

```
1  ALTER TABLE nome ação [, ... ]
2  ALTER TABLE nome RENAME [ COLUMN ] coluna TO
       novo_nome_da_coluna
3  ALTER TABLE nome RENAME TO novo_nome
4  onde ação é uma entre:
5           ADD [ COLUMN ] coluna tipo [ restrição_de_coluna [
                ... ] ]
6           DROP [ COLUMN ] coluna [ CASCADE ]
7           ALTER [ COLUMN ] coluna TYPE tipo [ USING expressão
                ]
8           ALTER [ COLUMN ] coluna SET DEFAULT expressão
9           ALTER [ COLUMN ] coluna DROP DEFAULT
10          ALTER [ COLUMN ] coluna { SET | DROP } NOT NULL
11          ADD restrição_de_tabela
```

Alter Table II

```
12          DROP CONSTRAINT nome_da_restrição [ RESTRICT |
                CASCADE ]
13          OWNER TO novo_dono
14          SET TABLESPACE nome_do_espaço_de_tabelas
```

Drop Table

Remove uma ou mais tabelas.

Código 5: Sintaxe da linguagem DDL para remover tabelas

```
1 DROP TABLE nome [, ...] [ CASCADE ]
```

Comment

Adiciona comentários em uma tabela, coluna, indice, restrição, função, indice, etc.

Código 6: Sintaxe da linguagem DDL para fazer comentários

```
1 COMMENT ON tipo_de_objeto nome_do_objeto IS 'text'
```

Create Index

Cria um índice numa tabela.

Código 7: Sintaxe da linguagem DDL para criar índices

```
1  CREATE [ UNIQUE ] INDEX nome_do_índice ON tabela [ USING
       método ]
2          ( { coluna | ( expressão ) } [ classe_de_operadores
               ] [, ...] )
3          [ TABLESPACE espaço_de_tabelas ]
4          [ WHERE predicado ]
```

Insert

Insere um registro em uma tabela.

Código 8: Sintaxe da linguagem DML para inserir registro

```
1  INSERT INTO tabela [ ( coluna [, ...] ) ]
2          { VALUES ( { expressão | DEFAULT } [, ...] ) |
               consulta }
```

Update

Atualiza um registro em uma tabela.

Código 9: Sintaxe da linguagem DML para atualizar registro

```
1  UPDATE tabela SET coluna = { expressão | DEFAULT } [, ...]
2          [ FROM lista_do_from ]
3          [ WHERE condição ]
```

Delete

Remove um registro em uma tabela.

Código 10: Sintaxe da linguagem DML para remover registro

```
1  DELETE FROM tabela [ WHERE condição ]
```

Truncate

Zera os registros de uma tabela.

Código 11: Sintaxe da linguagem DML para zerar os registros de uma tabela

```
1 TRUNCATE [ TABLE ] nome
```

Exists

Caracterização

Verifica se um valor existe no *resultset* retornado por um *select*.

- Se a subconsulta retornar pelo menos uma linha, o resultado dela é TRUE, caso contrário será FALSE;
- Observe a ligação entre a consulta externa e a interna.

Código 12: Exemplo de utilização do Exists

```
1  SELECT col1 FROM tab1
2  WHERE EXISTS(SELECT 1 FROM tab2 WHERE col2 = tab1.col2);
```

In e Not In

Caracterização

Verifica se um valor está contido em um grupo de valores.

Código 13: Exemplo de utilização do in e not in

```
1  SELECT DISTINCT nome_cliente
2  from devedor
3  Where nome_cliente in (select nome_cliente from depositante)
4
5  SELECT DISTINCT nome_cliente
6  from devedor
7  Where nome_cliente not in ('Smith', 'Jones')
```

Formas Alternativas para In e Not In

Código 14: Exemplo de utilização do in e not in de formas alternativas

```
1  SELECT * FROM produtos
2  WHERE cod_produto = (SELECT cod_produto FROM item_pedido)
3
4  SELECT * FROM Table1
5    WHERE (column1, column2) = (SELECT column1, column2 FROM
         Table2)
```

Inner Join

Caracterização

Somente as linhas/registros que satisfaçam a ligação determinada pelo *JOIN* serão recuperados pelo *SELECT*, sendo assim, os registros que não se enquadram no relacionamento definido pelo *JOIN* não serão recuperados.

Exemplo

Código 15: Exemplo de utilização do Inner Join

```
1  SELECT p.uname, p.nome, a.qtde
2  FROM pessoas p
3  INNER JOIN acessos a
4  ON p.uname = a.pessoa
5  ORDER BY p.uname;
6
7  O mesmo JOIN:
8
9  SELECT p.uname, p.nome, a.qtde
10 FROM pessoas p, acessos a
11 WHERE p.uname = a.pessoa
12 ORDER BY p.uname;
```

Left Join

Caracterização

Através do uso do *LEFT*, todos os registros na tabela à esquerda da *query* serão listados, independente de terem ou não registros relacionados na tabela à direita. Nesse caso, as colunas relacionadas com a tabela da direita voltam nulos (*NULL*).

Exemplo

Código 16: Exemplo de utilização do Left Join

```
1  SELECT p.uname, p.nome, a.pessoa, a.qtde
2  FROM pessoas p LEFT JOIN acessos a
3  ON p.uname=a.pessoa
4  ORDER BY p.uname;
```

Left Join

Caracterização

É o inverso do *Left Join*, ou seja, todos os registros da tabela à direita serão listados, independente de terem ou não registros relacionados na tabela à esquerda.

Exemplo

Código 17: Exemplo de utilização do Right Join

```
1  SELECT p.uname, p.nome, a.pessoa, a.qtde
2  FROM pessoas p
3  RIGHT JOIN acessos a
4  ON p.uname=a.pessoa
5  ORDER BY p.uname;
```