ACH2025

Laboratório de Bases de Dados

Aula 14

Bancos de Dados Objeto-Relacional

Professora:

- **Fátima L. S. Nunes**

Conceitos

- ✓ Problema do modelo relacional: tipos limitados de dados e incompatibilidade com linguagens OO
- ✓ Aplicações complexas exigem dados complexos:
  - estruturas aninhadas
  - atributos multivalorados
  - herança de atributos

Conceitos

- ✓ Sistemas de BD Orientados a Objetos:
  - estendem modelo relacional, incluindo tipos de dados complexos e orientação a objeto
  - permitem acesso direto aos dados de uma linguagem de programação usando sistema de tipo nativo da linguagem
- ✓ SQL também precisa ser estendida para lidar com tipos complexos de dados
  - extensão da SQL-99 e SQL-2003
  - SGBDs: geralmente não incluem todos recursos das extensões

Tipos de dados complexos

- ✓ Não seguem necessariamente Primeira Forma Normal
- ✓ **Tipos estruturados**: permitem que atributos do DER sejam representados diretamente no BD
- ✓ Exemplo 1:

```
create type Nome as
 (prenome varchar (20),
  sobrenome varchar (20))
 final
```

Tipos de dados complexos

✓ Exemplo 2:

```
create type Endereço as
 (rua varchar (20),
  cidade varchar (20),
  cep varchar (8))
 not final
```

Tipos de dados complexos

✓ Exemplo 2:

```
create type Endereço as
 (rua varchar (20),
  cidade varchar (20),
  cep varchar (8))
 not final
```

Tipos definidos pelo usuário

Relacionados à subtipagem:

final: não pode criar subtipo

Tipos de dados complexos

✓ Podemos usar tipos para criar as tabelas:

```
create table Cliente(
        nome Nome,
        endereço Endereço,
        dataDeNascimento date)
```

```
create type Nome as
        (prenome varchar (20),
         sobrenome varchar (20))
 final

create type Endereço as
        (rua varchar (20),
         cidade varchar (20),
         cep varchar (8))
 not final
```

Tipos de dados complexos

- ✓ Tabela também pode ser criada a partir de tipo que representa uma tupla:

```
create type TipoCliente as(
  nome Nome,
  endereço Endereço,
  dataDeNascimento date)
not final

create table cliente of TipoCliente
```

Tipos de dados complexos

- ✓ Acesso aos componentes de atributo composto usa notação de “ponto”:

```
nome.prenome

endereço.rua

endereço.cep
```

```
create type Nome as
        (prenome varchar (20),
         sobrenome varchar (20))
 final

create type Endereço as
        (rua varchar (20),
         cidade varchar (20),
         cep varchar (8))
 not final
```

Tipos de dados complexos

✓ Acesso aos componentes do atributo composto.

```
select nome.sobrenome,endereço.cidade
from cliente;
```

Tipos de dados complexos

✓ Tipo estruturado pode ter métodos definidos.

```
create type TipoCliente as(
  nome Nome,
  endereço Endereço,
  dataDeNascimento date)
not final
method idadeNaData(naData date)
  return interval year
```

Tipos de dados complexos

- ✓ Método pode ser definido separadamente:

```
create instance method idadeNaData(naData date)
      return interval year
      for TipoCliente
    begin
      return naData - self.dataDeNascimento;
    end
```

Tipos de dados complexos

✓ Método pode ser definido separadamente:

```
create instance method idadeNaData(naData date)
      return interval y
      for TipoCliente
    begin
      return naD  - self.dataDe
    end
```

Indica que método é executado em uma instância de TipoCliente

Indica para que tipo é este método

Tipos de dados complexos

✓ Método pode ser definido separadamente:

```
create instance method idadeNaData(naData date)
      return interval year
      for TipoCliente
    begin
      return naData - self.dataDeNascimento;
    end
```

Refere-se à instância Cliente em que o método é chamado

Tipos de dados complexos

- ✓ Método pode ser chamado em instância de um tipo:

```
select nome.sobrenome, idadeNaData(current_date)
from cliente;
```

Tipos de dados complexos

- ✓ Função construtora: criar valores de tipos estruturados.
- ✓ Função com mesmo nome de um tipo estruturado é função construtora para o tipo estruturado.

```
create function Nome (prenome varchar(20),
   sobrenome varchar(20))
   return Nome
   begin
      set self.prenome = prenome;
      set self.sobrenome = sobrenome;
   end
```

```
create type Nome as
         (prenome varchar (20),
          sobrenome varchar (20))
 final
```

Tipos de dados complexos

- ✓ Padrão: todo tipo estruturado tem uma função construtora sem argumentos
- ✓ Pode haver mais de uma função construtora para mesmo tipo: distintas por número e tipo dos argumentos

```
create function Nome (prenome varchar(20))
   return Nome
   begin
      set self.prenome = prenome;
      set self.sobrenome = 'SILVA';
   end
```

```
create type Nome as
        (prenome varchar (20),
         sobrenome varchar (20))
 final
```

Tipos de dados complexos

```
create function Endereço(rua varchar(20), cidade
  varchar(20), cep varchar(8))
  return Endereço
  begin
    set self.rua = rua;
    set self.cidade = cidade;
    set self.cep = cep;
  end
```

```
create type Endereço as
        (rua varchar (20),
         cidade varchar (20),
         cep varchar (8))
 not final
```

Tipos de dados complexos

- ✓ Função construtora: podemos criar nova tupla usando ***new***

```
insert into Cliente values
   (new Nome ('João', 'Silva'),
    new Endereço ('Rua ABC','São Paulo',01234000),
   date '1963-8-24')
```

```
create type Nome as
        (prenome varchar (20),
         sobrenome varchar (20))
 final

create type Endereço as
        (rua varchar (20),
         cidade varchar (20),
         cep varchar (8))
 not final
```

Herança de tipo

```
create type Pessoa
   (nome varchar (20),
    endereço varchar (20))
```

- ✓ Armazenar informações extras para estudantes e professores no BD:

```
create type Estudante
   under Pessoa
   (grau varchar (20),
    curso varchar (20))
```

```
create type Professor
   under Pessoa
   (salario integer(10),
    departamento varchar(20))
```

Herança de tipo

```
create type Pessoa
   (nome varchar (20),
    endereço varchar (20))
```

- ✓ Armazenar informações extras para estudantes e professores no BD:

```
create type Estudante
   under Pessoa
   (grau varchar (20),
    curso varchar (20))
```

```
create type Professor
   under Pessoa
   (salario integer(10),
    departamento varchar(20))
```

Estudante e Professor herdam os atributos de Pessoa
Estudante e Professor são chamados de subtipos de Pessoa

Herança de tipo

- ✓ Métodos também são herdados pelos subtipos
- ✓ Subtipo pode redefinir método declarando-o novamente, usando ***overriding method*** no lugar de ***method***
- ✓ Termo ***final*** no final do tipo indica que não podem ser criados subtipos daquele tipo
- ✓ Termo ***not final*** indica que podem ser criados subtipos daquele tipo

Herança de tipo

- ✓ Herança múltipla: tipo é subtipo de múltiplos tipos ***(não previsto até o padrão SQL 2003).***
- ✓ Pode renomear atributos que têm nomes repetidos no tipo pai:

```
create type ProfessorAssistente
   under Estudante,Professor
```

Herança de tipo

- ✓ Herança múltipla: tipo é subtipo de múltiplos tipos ***(não previsto até o padrão SQL 2003).***
- ✓ Pode renomear atributos que têm nomes repetidos no tipo pai:

```
create type Estudante
   under Pessoa
   (grau varchar (20),
    curso varchar (20))
```

```
create type Professor
   under Pessoa
   (salario integer(10),
    curso varchar(20))
```

```
create type ProfessorAssistente
   under Estudante with (curso as curso_estudante)
         Professor with (curso as curso_professor)
```

Herança de tabela

- ✓ Subtabelas na SQL correspondem à especialização/ generalização no DER.
- ✓ Tipos das subtabelas precisam ser subtipos do tipo da tabela pai.

```
create table pessoas of Pessoa
```

```
create table estudantes of
   Estudante under pessoas

create table professores of
   Professor under pessoas
```

- ✓ Todos atributos presentes em pessoas estão presentes nas subtabelas

Herança de tabela

- ✓ Cada tupla presente em **estudantes** ou **professores** também estará implicitamente presente em **pessoas**.
- ✓ Uma consulta feita à tabela **pessoas**, incluirá as tuplas diretamente incluída nesta tabela e mais as tuplas de **estudantes** e **professores** (*mas somente atributos de pessoas podem ser acessados*)

```
create table pessoas of Pessoa
```

```
create table estudantes of
   Estudante under pessoas

create table professores of
   Professor under pessoas
```

Herança de tabela

- ✓ Para encontrar tuplas que estão em pessoas, mas não em suas subtabelas: usar "**only** pessoas" no lugar de "pessoas"
- ✓ "**only**" também pode ser usado para **delete** e **update**.
- ✓ Os conceitos de herança múltipla aplicados a tipos também se aplicam a tabelas (lembrando que SQL até 2003 não os implementam)

Herança de tabela

✓ Requisitos de consistências para subtabelas:

– tuplas **correspondentes** entre tabela pai e subtabela: mesmos valores para atributos herdados

– requisitos de consistência:

1) cada tupla da supertabela pode corresponder a, no máximo, uma tupla em cada uma das subtabelas
   – Ex: não pode ter duas tuplas em **estudantes** que correspondam à mesma tupla em **pessoas**

2) restrição adicional da SQL: tuplas correspondentes precisam ser derivadas de uma tupla (inserida na tabela)
   – Ex: não pode ter uma tupla em **pessoas** correspondente tanto a uma tupla em **estudantes** quanto a uma tupla em **professores** (a menos que seja tupla de **professores_assistentes**)

Tipos array e multiconjunto na SQL

- ✓ SQL aceita dois tipos de coleção: arrays (SQL 1999) e multiconjunto (SQL 2003)
- ✓ Multiconjunto: coleção desordenada – um elemento pode ocorrer mais de uma vez

```
create type Editora as
   (nome varchar(20),
    divisão varchar(20))
create type Livro as
   (título varchar(20),
    array_autor varchar(20) array [10],
    data_pub date,
    editora Editora,
    conjunto_palavras_chave varchar(20) multiset)
create table livros of Livro
```

Tipos array e multiconjunto na SQL

```
create type Editora as
   (nome varchar(20),
    divisão varchar(20))
create type Livro as
   (título varchar(20),
    array_autor varchar(20) array [10],
    data_pub date,
    editora Editora,
    conjunto_palavras_chave varchar(20) multiset)
create table livros of Livro
```

até 10 nomes de autores – array: ordenação é importante

multset: ordenação não é importante

Tipos array e multiconjunto na SQL

- ✓ Criando e acessando valores de coleção
- ✓ Criação de array:

```
array ['Joao Silva', 'Maria Duarte', 'Luis
   Martins']
```

- ✓ Criação de multiset:

```
multiset['banco de dados', 'linguagem SQL',
   'entidade', 'SQL']
```

Tipos array e multiconjunto na SQL

- ✓ Criando e acessando valores de coleção
- ✓ Inserção em tabela:

```
insert into livros values
('Sistemas de Bancos de Dados',
  array['Korth','Silberschatz'],
   new Editora('McGraw-Hill','New York'),
  multiset('banco de dados', 'SQL'))
```

```
create type Livro as
    (título varchar(20),
     array_autor varchar(20) array [10],
     editora Editora,
     conjunto_palavras_chave varchar(20)
        multiset)
```

Tipos array e multiconjunto na SQL

- ✓ Consultando valores de coleção
- ✓ Todos os livros que têm ‘bancos de dados’ como palavra-chave:

```
select título
from livros
where ´bancos de dados´ in unnest(conjunto-
   palavras-chaves)
```

```
create type Livro as
    (título varchar(20),
     array_autor varchar(20) array [10],
     data_pub date,
     editora Editora,
     conjunto_palavras_chave varchar(20)
        multiset)
```

Tipos array e multiconjunto na SQL

- ✓ Consultando valores de coleção
- ✓ Todos os livros que têm ‘bancos de dados’ como palavra-chave:

```
select título
from livros
where ´bancos de dados´ in unnest(conjunto-
   palavras-chaves)
```

utilizado no lugar de “select-from-where”

```
array_autor varchar(20) array [10],
data_pub date,
editora Editora,
conjunto_palavras_chave varchar(20)
   multiset)
```

Tipos array e multiconjunto na SQL

- ✓ Consultando valores de coleção
- ✓ Considerando que um livro tem 3 autores:

```
select array_autor[1],array_autor[2],array_autor[3]
from livros
where título = ´Sistemas de Bancos de Dados’;
```

```
create type Livro as
    (título varchar(20),
     array_autor varchar(20) array [10],
     data_pub date,
     editora Editora,
     conjunto_palavras_chave varchar(20)
        multiset)
```

Tipos array e multiconjunto na SQL

- ✓ Consulta mostrando título e nome de autor para cada livro e cada autor do livro

```
select B.título, A.autor
from livros as B, unnest(B.array_autor) as
   A(autor);
```

```
create type Livro as
    (título varchar(20),
     array_autor varchar(20) array [10],
     data_pub date,
     editora Editora,
     conjunto_palavras_chave varchar(20)
        multiset)
```

Tipos array e multiconjunto na SQL

✓ Consulta mostrando título e nome de autor para cada livro e cada autor do livro

```
select B.título, A.autor
from livros as B, unnest(B.array_autor) as
   A(autor);
```

utilizado no lugar de "select-from-where"

```
ivro as
(título varchar(20),
 array_autor varchar(20) array [10],
 data_pub date,
 editora Editora,
 conjunto_palavras_chave varchar(20)
    multiset)
```

Tipos array e multiconjunto na SQL

- ✓ Aninhamento e desaninhamento
- ✓ Desaninhamento: transformação de uma relação aninhada em uma forma com menos atributos avaliados para relação
- ✓ Ex: converter **livros** em relação simples:

```
select titulo, A.autor, editora.nome as nome_editora,
   editora.divisão as divisao_editora, K.palavra-chave
from livros as B, unnest(B.array_autor) as
   A(autor),unnest(B.conjunto_palavras_chave) as
   K(palavra_chave)
```

Tipos array e multiconjunto na SQL

- ✓ Aninhamento e desaninhamento
- ✓ Desaninhamento: transformação de uma relação aninha em uma forma com menos atributos avaliados para
- ✓ Ex:

alias para autores de array_autor

alias para conjunto_palavras_chave

```
select titulo, A.autor, editora.nome as nome_editora,
   editora.divisão as divisao_editora, K.palavra-chave
from livros as B, unnest(B.array_autor) as
   A(autor),unnest(B.conjunto_palavras_chave) as
   K(palavra_chav
```

alias para livros

Tipos array e multiconjunto na SQL

✓ Tabela com aninhamento

título	array_autor	editora (nome,divisão)	conjunto_palavras_chave
Bancos de Dados	{Joao,Marcos,Maria}	(McGraw,New York)	{bancos de dados, SQL}
Sistemas de Bancos de Dados	{Joao,Daniel}	(McGraw,New York)	{bancos de dados, SQL}
Introdução à Computação	{Carlos,Luís}	(ABC, São Paulo)	{programação, algoritmos}

Tipos array e multiconjunto na SQL

✓ Tabela com aninhamento

título	array_autor	editora (nome,divisão)	conjunto_palavras_chave
Bancos de Dados	{Joao,Marcos,Maria}	(McGraw,New York)	{bancos de dados, SQL}
Sistemas de Bancos de Dados	{Joao,Daniel}	(McGraw,New York)	{bancos de dados, SQL}
Introdução à Computação	{Carlos,Luís}	(ABC, São Paulo)	{programação, algoritmos}

✓ Tabela sem aninhamento

título	autor	nome_editora	divisao_editora	palavra_chave
Bancos de Dados	Joao	McGraw	New York	bancos de dados
Bancos de Dados	Marcos	McGraw	New York	bancos de dados
Bancos de Dados	Maria	McGraw	New York	bancos de dados
Bancos de Dados	Joao	McGraw	New York	SQL
Bancos de Dados	Marcos	McGraw	New York	SQL
Bancos de Dados	Maria	McGraw	New York	SQL

Tipos array e multiconjunto na SQL

✓ Tabela com aninhamento

título	array_autor	editora (nome,divisão)	conjunto_palavras_chave
Bancos de Dados	{Joao,Marcos,Maria}	(McGraw,New York)	{bancos de dados, SQL}
Sistemas de Bancos de Dados	{Joao,Daniel}	(McGraw,New York)	{bancos de dados, SQL}
Introdução à Computação	{Carlos,Luís}	(ABC, São Paulo)	{programação, algoritmos}

✓ Tabela sem aninhamento

título	autor	n		palavra_chave
Bancos de Dados	Joao	M		bancos de dados
Bancos de Dados	Marcos	M		bancos de dados
Bancos de Dados	Maria	M		bancos de dados
Bancos de Dados	Joao	McGraw	New York	SQL
Bancos de Dados	Marcos	McGraw	New York	SQL
Bancos de Dados	Maria	McGraw	New York	SQL

Esta tabela está na Primeira Forma Normal

Tipos array e multiconjunto na SQL

✓ Processo inverso (desaninhado para aninhado): função **collect**

```
select titulo, autor,
   Editora(nome_editora,divisao_editora) as editora,
   collect  (palavra-chave) as conjunto_palavras_chave
   from livros_simples
group by título, autor, editora
```

Tipos array e multiconjunto na SQL

título	autor	nome_editora	divisao_editora	palavra_chave
Bancos de Dados	Joao	McGraw	New York	bancos de dados
Bancos de Dados	Marcos	McGraw	New York	bancos de dados
Bancos de Dados	Maria	McGraw	New York	bancos de dados
Bancos de Dados	Joao	McGraw	New York	SQL
Bancos de Dados	Marcos	McGraw	New York	SQL
Bancos de Dados	Maria	McGraw	New York	SQL

título	autor	editora (nome,divisão)	conjunto_palavras_chave
Bancos de Dados	Joao	McGraw.New York	{bancos de dados, SQL}
Bancos de Dados	Marcos	McGraw,New York	{bancos de dados, SQL}
Bancos de Dados	Maria	McGraw,New York	{bancos de dados, SQL}

```
select titulo, autor, Editora(nome_editora,divisao_editora) as
    editora,
    collect  (palavra-chave) as conjunto_palavras_chave
    from livros_simples
group by título, autor, editora
```

Tipos array e multiconjunto na SQL

livros_quase_objeto:

título	autor	editora (nome,divisão)	conjunto_palavras_chave
Bancos de Dados	Joao	McGraw.New York	{bancos de dados, SQL}
Bancos de Dados	Marcos	McGraw,New York	{bancos de dados, SQL}
Bancos de Dados	Maria	McGraw,New York	{bancos de dados, SQL}

Como transformar o atributo autor em um array de autor?

Tipos array e multiconjunto na SQL

livros_quase_objeto:

título	autor	editora (nome,divisão)	conjunto_palavras_chave
Bancos de Dados	Joao	McGraw.New York	{bancos de dados, SQL}
Bancos de Dados	Marcos	McGraw,New York	{bancos de dados, SQL}
Bancos de Dados	Maria	McGraw,New York	{bancos de dados, SQL}

Como transformar o atributo autor em um array de autor?

```
select titulo, collect (autor) as array_autor, editora,
    conjunto_palavras_chave
    from livros_quase_objeto
group by título, editora
```

Tipos array e multiconjunto na SQL

- ✓ Outra forma de criar relações aninhadas: *subqueries*
- ✓ Vantagem: *order by* pode ser usada opcionalmente na *subquery* para gerar resultados na ordem desejada
  - – resultado pode ser usado para criar um array

```
select título,
   array (select autor
          from autores as A
          where A.título = B.título
          order by A.position) as array_autor,
   Editora(nome_editora,divisão_editora) as editora,
   multiset (select palavra_chave
             from palavras_chave as K
             where K.título = B.título) as
   conjunto_palavras_chave,
from livros_dados as B
```

Tipos array e multiconjunto na SQL

```
select título,
   array (select autor
          from autores as A
          where A.título = B.título
          order by A.position) as arr    _autor,
   Editora(nome_editora,divisão_edit    ra) as editora,
   multiset (select palavra_chave
             from palavras_chave as K
             where K.título = B.título) as
   conjunto_palavras_chave,
from livros_dados as B
```

executa subconsultas aninhadas para cada tupla gerada pelas cláusulas from e where da consulta externa

Tipos array e multiconjunto na SQL

```
select título,
   array (select autor
          from autores as A
          where A.título = B.título
          order by A.position) as array_autor,
   Editora(nome_editora,divisão_editor   as editora,
   multiset (select palavra_chave
             from palavras_chave as K
             where K.título = B.título) as
   conjunto_palavras_chave,
from livros_dados as B
```

garante somente tuplas válidas (títulos coincidem)

Tipos array e multiconjunto na SQL

✓ Operadores de multiconjunto adicionais da SQL-2003:

- função set (M): retorna uma versão sem duplicatas de um multiconjunto M
- operação intersection: intersecção de todos os multiconjuntos em um grupo
- operação fusion: união de todos os multiconjuntos em um grupo
- predicado submultiset: verifica se um multiconjunto está contido em outro multiconjunto

Tipos array e multiconjunto na SQL

✓ Atualização de multiconjunto:

- tem que atribuir novo valor ao atributo
- para excluir o valor ***v*** de um multiset ***A*** deve usar ***A except all multiset[v]***

```
update livro
set título = título,
set palavras_chave = palavras_chave except all multiset
   [´bancos de dados´]
where título = ´Sistemas bancos de dados'
```

Referência a objetos em SQL

- ✓ Linguagens OO têm capacidade de referenciar objetos (atributo de determinado tipo pode ser referência a objeto daquele tipo)
- ✓ Em SQL também podemos fazer referências a tipos

```
create type Departamento
   ( nome varchar (20),
     chefe ref(Pessoa) scope pessoas)

create table departamentos of Departamento
```

Referência a objetos em SQL

- ✓ Linguagens OO têm capacidade de referenciar objetos (atributo de determinado tipo pode ser referência a objeto daquele tipo)
- ✓ Em SQL também podemos fazer referências a tipos

```
create type Departamento
   ( nome varchar (20),
     chefe ref(Pessoa) scope pessoas)

create table departamentos of Departamento
```

referência fica restrita às tuplas da relação *pessoas*

Referência a objetos em SQL

- ✓ **Scope** pode ser retirado da definição de tipo e incluído na criação da tabela.

```
create type Departamento
   ( nome varchar (20),
     chefe ref(Pessoa))

create table departamentos of Departamento
   (chefe with options scope pessoas)
```

Referência a objetos em SQL

- ✓ **Scope** pode ser retirado da definição de tipo e incluído na criação da tabela.

```
create type Departamento
   ( nome varchar (20),
     chefe ref(Pessoa))

create table departamentos of Departamento
   (chefe with options scope pessoas)
```

tabela referenciada (pessoas) deve ter atributo que armazena o identificador da tupla: atributo denominado autorreferencial
(acrescenta-se cláusula ref is na criação da tabela)

Referência a objetos em SQL

- ✓ **Scope** pode ser retirado da definição de tipo e incluído na criação da tal

```
create type Departamento
    ( nome varchar (20),
      chefe ref(Pessoa))

create table departamentos of Dep    ame
    (chefe with options scope pessoas)
```

```
create table pessoas of Pessoa
    ref is id_pessoa system generated
```

tabela referenciada (pessoas) tem que ter atributo que armazena o identificador da tupla: atributo denominado autorreferencial (acrescenta-se cláusula ref is na criação da tabela)

Referência a objetos em SQL

- ✓ **Scope** pode ser retirado da definição de tipo e incluído na criação da tabela.

```
create type Departamento
   ( nome varchar (20),
     chefe ref(Pessoa))

create table departamentos of Departamento
   (chefe with options scope pessoas)
```

```
create table pessoas of Pessoa
   ref is id_pessoa system generated
```

id_pessoa é um atributo
create table estabelece que o identificador é gerado automaticamente pelo BD

Referência a objetos em SQL

- ✓ **Para inicializar atributo de referência: obter identificador da tupla que deve ser referenciada ➔ consulta**
  - **Então, para criar tupla com valor de referência:**
    - **primeiro, criar tupla com referência nula**
    - **depois criar referência separadamente**

```
insert into departamentos
   values (´SI´, null)

update departamentos
   set chefe = (select p.id_pessoa
                from pessoas as p
                where nome = ´John´)
   where nome = ´SI´
```

Referência a objetos em SQL

- ✓ **Alternativa 1: permitir que usuários gerem identificadores**
  - **tipo do atributo autorreferencial precisa ser especificado na definição de tipo da tabela referenciada**
  - **definição de tabela precisa especificar que referência é gerada pelo usuário**

```
create type Pessoa
   (nome varchar (20),
    endereço varchar (40))
    ref using varchar (20)

create table pessoas of Pessoa
   ref is id_pessoa user generated
```

Referência a objetos em SQL

- ✓ **Quando insere tupla em *pessoas*, deve fornecer valor para identificador**

```
insert into pessoas (id_pessoa, nome, endereço) values
   (´2408´,'Maria', ´Av Rio Branco´)
```

- ✓ **Nenhuma outra tupla de pessoa ou de suas supertabelas ou subtabelas pode ter o mesmo identificador**
- ✓ **Podemos usar este identificador para inserir tupla em *departamentos***

```
insert into departamentos values (´CS´,´2408´)
```

```
create type Pessoa
    (nome varchar (20),
     endereço varchar (40))
     ref using varchar (20)

create table pessoas of Pessoa
   ref is id_pessoa user generated

create type Departamento
    ( nome varchar (20),
      chefe ref(Pessoa))
```

Referência a objetos em SQL

✓ **Alternativa 2:** usar chave primária já existente, incluindo cláusula **ref from** na definição do tipo

```
create type Pessoa
   (nome varchar (20) primary key,
    endereço varchar (40))
    ref from (nome)

create table pessoas of Pessoa
   ref is id_pessoa derived
```

na definição da tabela precisa especificar que referência é derivada e um nome de atributo autorreferencial

Referência a objetos em SQL

✓ **Quando insere tupla em *departamentos:***

```
insert into departamentos values (´CS´,´Maria´)
```

```
create type Pessoa
    (nome varchar (20),
     endereço varchar (40))
     ref using varchar (20)

create table pessoas of Pessoa
   ref is id_pessoa user generated

create type Departamento
    ( nome varchar (20),
      chefe ref(Pessoa))
```

Referência a objetos em SQL

- ✓ **Referências em SQL1999: usa símbolo ->**
- ✓ **Mostrar nome e endereço dos chefes de todos os departamentos**

```
select chefe->nome,chefe->endereço from departamentos
```

- ✓ **expressão do tipo `chefe->nome` é chamada de expressão de caminho**
- ✓ ***chefe* é referência a uma tupla de *pessoas*: atributos *nome* e *endereço* são extraídos de *pessoas***

Referência a objetos em SQL

- ✓ **Referências em SQL1999: usa símbolo ->**
- ✓ **Mostrar nome e endereço dos chefes de todos os departamentos**

```
select chefe->nome,chefe->endereço from departamentos
```

- ✓ **expressão do tipo `chefe->nome` é chamada de expressão de caminho**
- ✓ ***chefe* é referência a uma tupla de *pessoas*: atributos *nome* e *endereço* são extraídos de *pessoas***

Como seria esta consulta se não houvesse objetos?

Referência a objetos em SQL

- ✓ **Referências em SQL1999: usa símbolo ->**
- ✓ **Mostrar nome e endereço dos chefes de todos os departamentos**

```
select chefe->nome,chefe->endereço from departamentos
```

- ✓ **expressão do tipo `chefe->nome` é chamada de expressão de caminho**
- ✓ ***chefe* é referência a uma tupla de *pessoas*: atributos *nome* e *endereço* são extraídos de *pessoas***

Como seria esta consulta se não houvesse objetos?

- ✓ ***chefe* da tabela *departamentos* seria chave estrangeira em relação à tabela *pessoas***
- ✓ **consulta deveria ser feita usando junção explícita**

Referência a objetos em SQL

- ✓ **Uso de referência simplifica consultas**
- ✓ **Operador deref: retorna tupla apontada por uma referência e acessa os atributos desta tupla**

```
select deref(chefe).nome
from departamentos
```

Implementação dos conceitos OR nos SGBDs

- ✓ **SGBDOR: extensão dos SGBDR**
- ✓ **tipos de dados complexos podem ser traduzidos para tipos mais simples dos BDR**
- ✓ **Armazenamentos de subtabelas:**

  Primeira opção:

  - **cada tabela armazena chave primária (PK) (que pode ser herdada da tabela-pai) e os atributos são definidos localmente**
  - **atributos herdados (menos PK) não precisam ser armazenados: podem ser derivados por meio de uma junção com as supertabelas (com base na PK)**

Implementação dos conceitos OR nos SGBDs

- ✓ **SGBDOR: extensão dos SGBDR**
- ✓ **tipos de dados complexos podem ser traduzidos para tipos mais simples dos BDR**
- ✓ **Armazenamentos de subtabelas:**

  Segunda opção:

  - **cada tabela armazena atributos herdados e definidos localmente**
  - **quando tupla é inserida: sua presença é suposta em cada uma das supertabelas ➔ acesso mais rápido (sem junção de tabelas)**

Implementação dos conceitos OR nos SGBDs

- ✓ **Implementações podem representar *arrays* e *multiconjuntos* diretamente ou representá-los de forma normalizada internamente**
  - **normalizadas:**
    - **mais espaço, custo de recuperação mais alto (junções)**
    - **mais fáceis de implementar**
  - **JDBC e ODBC**
    - **estendidas para recuperar e armazenar tipos estruturados**
    - **exemplo: JDBC tem método *getObject()* que retorna objeto do tipo *Struct Java*.**
    - **é possível associar uma classe Java a um tipo estruturado em SQL: JDBC faz conversão entre tipos.**

Linguagens de Programação persistentes

- ✓ **Linguagens de Programação (LP) estendida com construções para manipular dados persistentes**
- ✓ **Diferentes das linguagens com SQL incorporada:**
  - **Linguagem incorporada: sistema de tipo da LP pode diferir do sistema de tipo da linguagem de manipulação de dados (DDL)**
    - **programador tem que fazer conversão.**
  - **LP persistente: linguagem de consulta está integrada com a LP: compartilham mesmo sistema de tipo.**
    - **objetos podem ser criados e armazenados no BD sem conversões. Eventuais mudanças de formato são transparentes.**
    - **complexidade da LP pode dificultar otimização (como redução de E/S)**

Linguagens de Programação persistentes

- ✓ **LPOO já têm estruturas para trabalhar com objetos, mas estes são transientes**
- ✓ **Vários métodos propostos para tornar uma LPOO em LP persistente:**

1) **Persistência por classe:**
   - **todos objetos da classe são persistentes**
   - **método não é flexível: geralmente é útil ter objetos transientes e persistentes em uma mesma classe**

2) **Persistência por criação:**
   - **nova sintaxe é introduzida para criar objetos persistentes e transientes**
   - **mais flexível**
   - **adotado por vários sistemas de BD**

Linguagens de Programação persistentes

- ✓ **LPOO já têm estruturas para trabalhar com objetos, mas estes são transientes**
- ✓ **Vários métodos propostos para tornar uma LPOO em LP persistente:**

3) Persistência por marcação:

- **variação da persistência por criação**
- **marca objetos como persistentes após terem sido criados**
- **todos objetos inicialmente são criados como transientes**

4) Persistência por acessibilidade:

- **objetos são declarados explicitamente como persistentes (objeto raiz)**
- **outros objetos são persistentes somente se forem acessíveis do objeto raiz de/por uma sequência de referências**
- **maior custo para SGBD: responsável por identificar objetos persistentes**

Sistemas Java persistentes

- ✓ padrão de persistência começou liderado pelo consórcio ODMG (Object Database Management Group)
- ✓ transferido para a iniciativa Java Database Objects (JDO), coordenada pela Sun Microsystems
- ✓ Recursos:
  - Persistência por acessibilidade:
    - objetos não são explicitamente criados em um BD
    - método para tornar objeto persistente: `makePersistent()` da classe `PersistenceManager`
    - objetos acessíveis de um objeto persistente se tornam também persistentes

Sistemas Java persistentes

- ✓ **Recursos:**
  - **Melhoria do *bytecode*:**
    - em vez de declarar classe persistente no código Java, classes cujos objetos podem se tornar persistentes são especificadas em arquivo de configuração (sufixo `.jdo`)
    - programa específico da implementação lê o arquivo de configuração para:
      - criar estruturas em um BD para armazenar objetos da classe
      - modificar o *bytecode* para executar tarefas relacionadas com persistência
      - Exemplo de tarefa: qualquer código que acessa o objeto é modificado para primeiro verificar se objeto está na memória; se não estiver, busca do BD.

Sistemas Java persistentes

- ✓ **Recursos:**
  - **Mapeamento do BD:**
    - **JDO não define como objetos são armazenados no BD final**
    - **Alguns objetos podem ser mapeados para armazenamento relacional e outros para armazenamento serializado (objeto binário no BD)**
  - **Extensões de classe:**
    - **extensões criadas e mantidas automaticamente para cada classe declarada como persistente**
  - **Tipos de referência único:**
    - **não há diferença entre referência a objeto persistente e referência a objeto transiente**

Sistemas Java persistentes

- ✓ **Informações sobre Oracle:**

  Silberschatz, A.; Korth, H.F.; Sudarshan, S. “Sistema de Banco de Dados”, 5a. edição, Makron Books, 2006 (capítulo 27)

- ✓ **Informações sobre PostGreSQL:**

  Silberschatz, A.; Korth, H.F.; Sudarshan, S. “Sistema de Banco de Dados”, 5a. edição, Makron Books, 2006 (capítulo 26)

Bibliografia

- ✓ Silberschatz, A.; Korth, H.F.; Sudarshan, S. “Sistema de Banco de Dados”, 5a. edição, Makron Books, 2006 (capítulo 9)

Exercícios

- ✓ Silberschatz, A.; Korth, H.F.; Sudarshan, S. "Sistema de Banco de Dados", 5a. edição, Makron Books, 2006 (capítulo 9) – exercícios 1 a 12

ACH2025

Laboratório de Bases de Dados

Aula 14

Bancos de Dados Objeto-Relacional

Professora:

- **Fátima L. S. Nunes**