Banco de Dados

Bancos de Dados Orientados a Objetos

DCC-UFLA

Prof. Denilson Alves Pereira

denilsonpereira@dcc.ufla.br

http://lattes.cnpq.br/4120230814124499

CIÊNCIA DA COMPUTAÇÃO
UFLA - LAVRAS - MG

UFLA
UNIVERSIDADE FEDERAL DE LAVRAS

versão dos slides: jul/2017

Introdução

- Bancos de dados orientados a objetos são conhecidos agora como **Bancos de Dados de Objetos** (**BDO**)
- E seus SGBDs, **Sistemas de Gerenciamento de Dados de Objeto** (**SGDO**)
- Foram propostos para atender a necessidades de aplicações mais complexas
  - Estruturas mais complexas para objetos armazenados
  - Novos tipos de dados para armazenar imagens, vídeos ou textos grandes
  - Transações de maior duração
  - Operações específicas da aplicação

Introdução

- Nos BDOs, é possível especificar:
  - A estrutura dos objetos complexos
  - As operações que podem ser aplicadas a esses objetos
- BDOs podem ser integrados diretamente à sistemas desenvolvidos usando linguagens de programação orientadas a objetos
  - Evitam o problema de divergência de impedância

Introdução

- Duas vertentes de desenvolvimento
  - Banco de dados de objeto “puro”
    - Vários protótipos experimentais e sistemas comercial foram desenvolvidos
      - Orion, Ode, GemStone, ONTOS, Versant, ObjectStore, Ardent (O2)
    - Padrão ODMG (Object Data Management Group)
  - Banco de dados objeto-relacional
    - Incorporação de recursos propostos para BDO em SGBD relacional
    - Recursos incluídos no padrão SQL
- O foco dos slides será em banco de dados objeto-relacional

Conceitos de Banco de Dados de Objeto

- Um objeto possui dois componentes:
  - Estado (valor)
  - Comportamento (operações)
- Uma operação é definida em duas partes:
  - Assinatura (ou interface)
    - especifica o seu nome e seus parâmetros
  - Método (ou corpo)
    - especifica sua implementação
    - escrita em alguma linguagem de programação

Conceitos de Banco de Dados de Objeto

- Herança
  - Permite a especificação de novos tipos ou classes que herdam grande parte de sua estrutura e/ou operações de tipos ou classes previamente definidas
  - Permitem a reutilização
- Sobrecarga de operador
  - Capacidade de uma operação de ser aplicada a diferentes tipos de objetos
  - Um nome de operação pode se referir a várias implementações distintas
  - Recurso conhecido como polimorfismo de operador

Identidade de Objetos *versus* Literais

- Identificador de objeto (OID)
  - Cada objeto armazenado tem uma identificação única
  - Imutável
  - Não visível ao usuário externo
  - Usado para referenciar um objeto
- Literal (ou valor)
  - Não tem OID
  - Armazenado dentro do objeto
  - Não pode ser referenciado de outros objetos

Estruturas de Tipos Complexas

- Objetos e literais podem ter uma estrutura de tipo de complexidade arbitrária
  - Contém todas as informações necessárias para descrever o objeto ou literal
  - Diferente dos sistemas de bancos de dados tradicionais, onde a informação pode ser espalhada por muitas relações ou registros
- Construtores de tipos
  - Um tipo complexo pode ser construído com base em outros tipos por meio do aninhamento de construtores de tipos

Estruturas de Tipos Complexas

- Os três construtores de tipos mais básicos são:
  - Átomo
    - Tipos de dados embutidos básicos: inteiros, cadeias de caracteres, reais, booleanos, tipos enumerados etc.
    - Valor atômico
  - Struct (ou tupla)
    - Tipo estruturado padrão, para criar tuplas ou registros
    - Composto de vários componentes (tipo composto)
  - Coleção (ou multivalorado)
    - Inclui uma coleção de outros objetos ou valores
    - Todos os elementos do mesmo tipo
    - Principais construtores: set, list, bag, array e dictionary

Estruturas de Tipos Complexas

- Linguagem de Definição de Objetos (ODL)
  - Incorpora os construtores de tipos
  - Usada para definir os tipos de objeto
- Os construtores de tipos podem ser usados pra definir as estruturas de dados para um esquema de banco de dados OO

Estruturas de Tipos Complexas

```
define type  FUNCIONARIO
tuple (
    Pnome:              string;
    Minicial:           char;
    Unome:              string;
    Cpf:                string;
    Data_nascimento:    DATE;
    Endereco:           string;
    Sexo:               char;
    Salario:            float;
    Supervisor:         FUNCIONARIO;
    Dep:                DEPARTAMENTO;

define type  DATA
tuple (
    Ano:                integer;
    Mes:                integer;
    Dia:                integer; );

define type  DEPARTAMENTO
tuple (
    Dnome:              string;
    Dnumero:            integer;
    Ger:                tuple (
                        Gerente: FUNCIONARIO;
                        Data_inicio: DATE; );
    Localizacoes:       set(string);
    Funcionarios:       set(FUNCIONARIO);
    Projetos:           set(PROJETO); );
```

Encapsulamento de Operações

- Relacionado aos conceitos de tipos abstratos de dados e ocultação de informação
- Define o comportamento de um tipo de objeto com base nas operações que podem ser aplicadas externamente
- Usuários externos só conhecem a interface das operações
  - Os detalhes de implementação são escondidos
  - Interface – chamada de assinatura
  - Implementação – chamada de método
- A estrutura de um objeto é dividida em atributos visíveis e ocultos
- O termo **classe** se refere a uma definição de tipo, junto com as definições das operações para esse tipo

Operações Típicas

- Construtor de objeto
  - Usado para criar um objeto
- Construtor de destruição
  - Usado para destruir (excluir) um objeto
- Operações modificadoras
  - Usadas para modificar os estados (valores) de atributos de um objeto
- Operações para recuperar informações sobre um objeto

Operações Típicas

```
define class FUNCIONARIO
 type tuple (
      Pnome:              string;
      Minicial:           char;
      Unome:              string;
      Cpf:                string;
      Data_nascimento:    DATE;
      Endereco:           string;
      Sexo:               char;
      Salario:            float;
      Supervisor:         FUNCIONARIO;
      Dep:                DEPARTAMENTO;   );
 operations
      idade:              integer;
      criar_func:         FUNCIONARIO;
      destroi_func:       boolean;
end FUNCIONARIO;

define class DEPARTAMENTO
 type tuple (
      Dnome:              string;
      Dnumero:            integer;
      Ger:                tuple (
                          Gerente: FUNCIONARIO;
                          Data_inicio: DATE; );
      Localizacoes:       set (string);
      Funcionarios:       set (FUNCIONARIO);
      Projetos            set(PROJETO); );
 operations
      nr_funcs:           integer;
      criar_dep:          DEPARTAMENTO;
      destroi_dep:        boolean;
      aloca_func          (e: FUNCIONARIO): boolean;

      (* acrescenta um funcionário ao departamento *)
      remove_func(e: FUNCIONARIO):
      boolean;
      (* remove um funcionário do departamento *)
end DEPARTAMENTO;
```

Persistência de Objetos

- Objetos transientes
  - Existem no programa em execução
  - Desaparecem quando o programa termina
- Objetos persistentes
  - Armazenados no banco de dados e persistem após o término do programa
- Mecanismos para tornar um objeto persistente
  - Nomeação
  - Acessibilidade

Hierarquias de Tipo e Herança

- Herança
  - Permite a definição de novos tipos com base em outros tipos predefinidos
  - Leva a uma hierarquia de tipos
- Um tipo (ou classe) possui um nome, uma lista de atributos (variáveis de instância) e operações (métodos)
- Subtipo
  - Herda todos os atributos e operações do tipo predefinido
  - Exemplo:

    PESSOA: cpf, nome, endereço, dataNascimento

    FUNCIONARIO **subtype-of** PESSOA: salário, dataContratação

Herança Múltipla e Herança Seletiva

- Herança Múltipla
  - Ocorre quando um subtipo T é um subtipo de dois (ou mais) tipos
  - T herda as funções (atributos e métodos) dos dois supertipos
- Herança Seletiva
  - Ocorre quando um subtipo herda apenas algumas das funções de um supertipo

Recursos Objeto-Relacional: Extensões do Banco de Dados de Objeto para SQL

Recursos Objeto-Relacional

- Modelo Objeto-Relacional
  - Modelo relacional com adição de conceitos de banco de dados de objeto
- A linguagem SQL incorporou recursos de BDO em seu padrão
  - Construtores de tipos
  - Identidade de objeto
  - Encapsulamento de operações
  - Herança

Tipos Definidos pelo Usuário (UDT)

- Permitem a criação de objetos estruturados complexos
- Sintaxe:

  CREATE TYPE nome-tipo AS

  (<declarações de componentes>);

- ROW type
  - Cria diretamente um atributo estruturado usando a palavra-chave ROW

```
CREATE TYPE TIPO_END_BRASIL AS (
   END_RUA  ROW (NUMERO      VARCHAR(5),
                 NOME_RUA    VARCHAR(25),
                 NR_APTO     VARCHAR(5),
                 NR_BLOCO    VARCHAR(5)),
   CIDADE   VARCHAR(25),
   CEP      VARCHAR(8)
);
```

Tipos Definidos pelo Usuário (UDT)

- Tipos de coleção
  - ARRAY, MULTISET, LIST e SET
- Função CARDINALITY
  - Retorna o número atual de elementos em uma coleção

Tipos Definidos pelo Usuário (UDT)

```
(a) CREATE TYPE TIPO_END_RUA AS (
        NUMERO              VARCHAR (5),
        NOME_RUA            VARCHAR (25),
        NR_APTO             VARCHAR (5),
        NR_BLOCO            VARCHAR (5)
    );
    CREATE TYPE TIPO_END_BRASIL AS (
        END_RUA             TIPO_END_RUA,
        CIDADE              VARCHAR (25),
        CEP                 VARCHAR (10)
    );
    CREATE TYPE TIPO_TELEFONE_BRASIL AS (
        TIPO_TELEFONE       VARCHAR (5),
        CODIGO_AREA         CHAR (3),
        NUM_TELEFONE        CHAR (7)
    );

(b) CREATE TYPE TIPO_PESSOA AS (
        NOME                VARCHAR (35),
        SEXO                CHAR,
        DATA_NASCIMENTO     DATE,
        TELEFONES           TIPO_TELEFONE_BRASIL ARRAY [4],
        END                 TIPO_END_BRASIL
    INSTANTIABLE
    NOT FINAL
    REF IS SYSTEM GENERATED
    INSTANCE METHOD IDADE( ) RETURNS INTEGER;
    CREATE INSTANCE METHOD IDADE( ) RETURNS INTEGER
        FOR TIPO_PESSOA
        BEGIN
            RETURN /* CÓDIGO PARA CALCULAR A IDADE DE UMA PESSOA COM BASE NA
                      DATA DE HOJE E SUA DATA_NASCIMENTO */
        END;
    );

(c) CREATE TYPE TIPO_NOTA AS (
        DISCIPLINA          CHAR (8),
        SEMESTRE            VARCHAR (8),
        ANO                 CHAR (4),
        NOTA                CHAR
    );
    CREATE TYPE TIPO_ALUNO UNDER TIPO_PESSOA AS (
        CODIGO_CURSO        CHAR (4),
        COD_ALUNO           CHAR (12),
        GRAU                VARCHAR (5),
        HISTORICO_ESCOLAR   TIPO_NOTA ARRAY [100]
```

(continua)

Tipos Definidos pelo Usuário (UDT)

```
    INSTANTIABLE
    NOT FINAL
    INSTANCE METHOD COEFICIENTE( ) RETURNS FLOAT;
    CREATE INSTANCE METHOD COEFICIENTE( ) RETURNS FLOAT
        FOR TIPO_ALUNO
        BEGIN
            RETURN /* CÓDIGO PARA CALCULAR COEFICIENTE MEDIO DE UM ALUNO COM BASE EM
                      SEU HISTORICO ESCOLAR */
        END;
    );
    CREATE TYPE TIPO_FUNCIONARIO UNDER TIPO_PESSOA AS (
        CODIGO_EMPREGO      CHAR (4),
        SALARIO             FLOAT,
        CPF                 CHAR (11)
    INSTANTIABLE
    NOT FINAL
    );
    CREATE TYPE TIPO_GERENTE UNDER TIPO_FUNCIONARIO AS (
        DEP_GERENCIADO      CHAR (20)
    INSTANTIABLE
    );

(d) CREATE TABLE PESSOA OF TIPO_PESSOA
        REF IS ID_PESSOA SYSTEM GENERATED;
    CREATE TABLE FUNCIONARIO OF TIPO_FUNCIONARIO
        UNDER PESSOA;
    CREATE TABLE GERENTE OF TIPO_GERENTE
        UNDER FUNCIONARIO;
    CREATE TABLE ALUNO OF TIPO_ALUNO
        UNDER PESSOA;

(e) CREATE TYPE TIPO_EMPRESA AS (
        NOME_EMP            VARCHAR (20),
        LOCALIZACAO         VARCHAR (20));
    CREATE TYPE TIPO_EMPREGO AS (
        Funcionario REF (TIPO_FUNCIONARIO) SCOPE (FUNCIONARIO),
        Empresa REF (TIPO_EMPRESA) SCOPE (EMPRESA) );
    CREATE TABLE EMPRESA OF TIPO_EMPRESA (
        REF IS COD_EMP SYSTEM GENERATED,
        PRIMARY KEY (NOME_EMP) );
    CREATE TABLE EMPREGO OF TIPO_EMPREGO;
```

Identificadores de Objeto

- Indica que o identificador de objeto será gerado pelo sistema

  REF IS SYSTEM GENERATED

- É também possível usar as chaves tradicionais do modelo relacional básico
- Cria identificador

  REF IS <nome-identificador> [SYSTEM GENERATED | DERIVED]

Criando Tabelas baseadas nos UDTs

- UDT instanciável
  - Especificado pela palavra-chave INSTANTIABLE
  - Tabelas podem ser criadas com base no UDT

    CREATE TABLE PESSOA **OF** TIPO_PESSOA
- Um UDT não instanciável só pode ser usado como tipo para atributos

Encapsulamento de Operações

- Especificação UDT

  CREATE TYPE <nome-tipo> (

  <lista-de-atributos-e-seus-tipos>

  <declaração-de-funções(métodos)>

  );

- Funções embutidas
  - Construtora (TYPE_T): retorna um novo objeto desse tipo
  - Observadora (GET): retorna o valor do atributo
  - Alteradora (SET): define um novo valor para o atributo
  - Privilégio EXECUTE é necessário para se ter acesso a essas funções
  - As funções podem ser bloqueadas contra uso público

Encapsulamento de Operações

- Sintaxe de definição de função:

  INSTANCE METHOD <nome> (<lista-de-argumentos>)
  RETURNS <tipo-de-retorno>

- Dois tipos de função:
  - Função interna: escrita na linguagem PSM estendida de SQL
  - Função externa: escrita em uma linguagem hospedeira
- Categoria de atributos e funções:
  - PUBLIC: visíveis na interface do UDT
  - PRIVATE: não visíveis na interface do UDT
  - PROTECTED: visíveis apenas aos subtipos

Herança e Sobrecarga de Funções

- Regras de herança:
  - Todos os atributos são herdados
  - A ordem dos supertipos na cláusula UNDER determina a hierarquia de herança
  - Uma instância de um subtipo pode ser usada em cada contexto em que uma instância do supertipo é utilizada
  - Um subtipo pode redefinir qualquer função que é definida em seu supertipo, mantendo a mesma assinatura
  - Quando uma função é chamada, a melhor combinação é selecionada com base nos tipos de todos os argumentos
  - Para a ligação dinâmica, a execução dos tipos de parâmetros são considerados

Herança e Sobrecarga de Funções

- A palavra-chave NOT FINAL indica que subtipos podem ser criados
- Herança de tabela
  - Quando um novo registro é inserido em uma subtabela, também é inserido em suas supertabelas
  - Operações INSERT, UPDATE e DELETE são propagadas corretamente

Relacionamentos por Referência

- Um atributo componente de uma tupla pode ser uma referência a uma tupla de outra tabela (ou da mesma)
  - Especificado pela palavra-chave REF
  - A palavra-chave SCOPE especifica o nome da tabela cujas tuplas podem ser referenciadas
  - Semelhante à noção de chave estrangeira
- Expressões de caminho usam a notação de ponto
  - Atributos do tipo REF usam o símbolo de desreferência –>
  - Ex: recuperar os funcionários que trabalham na empresa de nome ‘ABC’

    SELECT E.Funcionario–>NOME

    FROM EMPREGO **AS** E

    WHERE E.Empresa–>NOME_EMP = ‘ABC’;

Diferenças de Projeto de Banco de Dados de Objetos (BDO) e de Banco de Dados Relacional (BDR)

Projeto BDO x BDR

- Relacionamentos
  - BDO
    - Referências de OID aos objetos relacionados
    - Relacionamentos binários podem ser declarados em uma direção ou nas duas direções
      - Relacionamentos M:N podem ser implementados diretamente usando coleções dos dois lados
      - Se o relacionamento contiver atributos, normalmente cria-se uma classe separada para representar o relacionamento
        - Técnica também usada para relacionamentos de grau n > 2
  - BDR
    - Referências de valor especificadas por meio chaves estrangeiras
    - Relacionamentos binários podem ser declarados somente em uma direção
      - Relacionamentos M:N necessitam de uma tabela separada

Projeto: BDO x BDR

- Herança
  - BDO
    - Estrutura embutida no modelo
  - BDR
    - Não possui construção embutida para herança
    - Várias opções de mapeamento EER para relacional
- Operações
  - BDO
    - Fazem parte da especificação das classes
  - BDR
    - Não são exigidas antes da fase de implementação

Bibliografia Básica

ELMASRI, Ramez; NAVATHE, Shamkant B. Sistemas de Bancos de Dados. Pearson Education, 6a edição, 2011. ISBN-978-85-7936-085-5

Capítulo 11