BANCO DE DADOS ORIENTADO A OBJETOS

- Introdução
- Conceito de BDOO
- Características OO
- Linguagens OO
- Linguagens de Programação Persistentes
- Persistência de Objetos
- Identidade de Objetos
- Armazenamento e Acesso a Objetos Persistentes
- Linguagem de Definição de Objeto
- Linguagem de Manipulação de Objeto

1

Introdução

- Novas aplicações para sistemas de BD estão sendo limitadas por restrições do modelo relacional
- Novos modelos de dados estão sendo propostos, entre eles o orientado a objeto
- A abordagem orientada a objetos oferece flexibilidade para manipular requisitos das novas aplicações sem ser limitada a tipos de dados e às linguagens de consulta disponíveis em SGBDs tradicionais.

2

Introdução

Características das aplicações "antigas":

- **Uniformidade:** dados estruturados com o mesmo tamanho
- **Orientação a Registro**: itens são registros de comprimento fixo
- **Itens de dados pequenos**: registros curtos, poucas centenas de bytes
- **Campos Atômicos**: campos curtos e de comprimento fixo

3

Introdução

Características das novas aplicações:

- Estruturas complexas para objetos
- Transações com mais longa duração
- Novos tipos de dados para armazenar imagens ou grandes itens textuais
- Necessidade de definir operações de aplicação específica não-padrão

4

Conceito de BDOO

- É um sistema em que a unidade de armazenamento é o objeto, com o mesmo conceito das linguagens de programação orientadas a objetos
- Diferença fundamental é a persistência dos objetos: continuam a existir mesmo após o encerramento do programa
- Combina os benefícios e conceitos da orientação a objetos com a funcionalidade dos bancos de dados

5

Características de OO - Objeto

- Um objeto corresponde a uma entidade no modelo ER
- Todas as interações entre um objeto e o sistema se dão via mensagens
- A interface entre um objeto e o resto do sistema é definida por um conjunto permitido de mensagens

6

Características de OO - Objeto

São associados a um objeto:

- Conjunto de variáveis que contém os dados (Correspondem aos atributos em ER)
- Conjunto de mensagens ao qual o objeto responde (pode ter múltiplos parâmetros)
- Um conjunto de métodos, cada qual sendo um corpo de código p/ implementar a mensagem (um método retorna um valor em resposta a mensagem)

7

Características de OO - Objeto

- *Mensagem* em OO não significa uma mensagem física pela rede, e sim passagem de pedidos entre objetos, ocultando os detalhes de implementação
- Todos objetos respondem as mensagens, mas de modos diferentes, mas pelo encapsulamento todos apresentam a mesma interface
- A única interface externa apresentada é o conjunto de mensagens as quais o objeto responde
- Os métodos podem ser classificados como *somente leitura* ou *atualização*, assim como as mensagens

8

Características de OO – Classes de Objeto

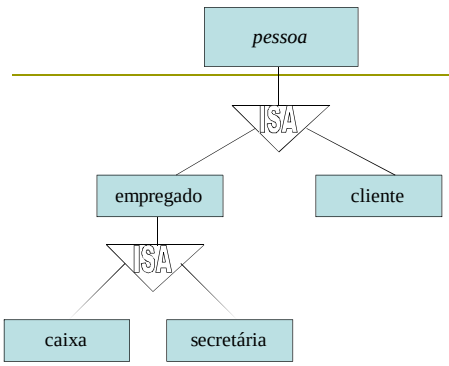
- Existem muitos objetos similares em um BD
- Similares: mesmos métodos, variáveis de mesmo nome e tipo, respondem as mesmas mensagens.
- Agrupamento desses objetos forma a classe (noção de classe se assemelha a um conjunto de entidades no ER)

9

Características de OO – Herança

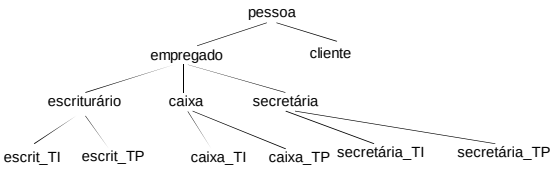
- Um esquema de BD orientado a objeto exige um grande número de classes, muitas são similares entre si
- Para representar essa similaridade, colocamos as classes em uma hierarquia de especialização
- Motivação: reusabilidade de código
- Exemplo: em um sistema bancário há clientes e funcionários. Ambos possuem atributos semelhantes, então ambos são definidos como PESSOA, após isso separam-se suas características individuais

10



11

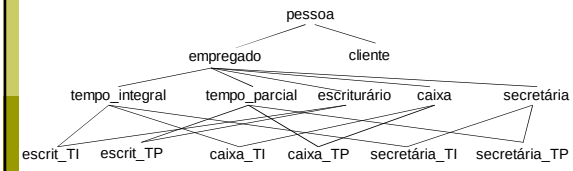
Características de OO – Herança



12

Características de OO – Herança

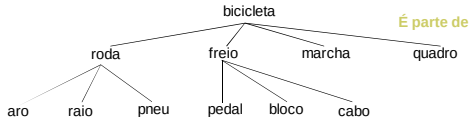
- Herança Múltipla pode gerar ambiguidade se a mesma variável ou método for herdada por mais de uma superclasse



13

Características de OO – Objetos Compostos

Os links entre classes são interpretados como é-parte-de em vez de é-um.



14

Características de OO – Identidade de Objeto

- **Valor**: Um valor de dados é usado para identidade. Essa forma de identidade é utilizada em sistemas relacionais. Por exemplo, o valor da chave primária de uma tupla identifica a tupla.
- **Nome**: Um nome fornecido pelo usuário é usado para identidade. Essa forma de identidade é utilizada em sistemas de arquivos convencionais. Por exemplo autoexec.bat
- **Embutido**: Uma noção de identidade está embutida no modelo de dados ou linguagem de programação e nenhum identificador fornecido pelo usuario é necessário. Usado em sistemas orientados a objeto.

15

Características de OO – Construtores de Tipo

- Uma Linguagem de Definição de Objeto (ODL) que incorpora os construtores de tipos pode ser usada para definir os tipos de objetos para uma aplicação de banco de dados particular.
- Os construtores de tipo podem ser usados para definir as estruturas de dados para um esquema de banco de dados OO.

16

Características de OO – Construtores de Tipo

```
Define type Employee:
   tuple (   fname:        string;
             minit:        char;
             lname:        string;
             ssn:          string;
             birthdate:    Date;
             address:      string;
             sex:          char;
             salary:       float;
             supervisor:   Employee;
             dept:         Department;  );
```

17

Características de OO – Construtores de Tipo

```
Define type Date:
   tuple (   year:         integer;
             month:        integer;
             day:          integer;

Define type Department:
   tuple (   dname:        string;
             dnumber:      integer;
             mgr:          tuple ( manager: Employee;
                                   startdate: Date;    );
             locations:    set(string);
             employees:    set(Employee);
             projects:     set(Project);  );
```

18

Linguagens OO

- Existem 2 maneiras de se usar conceitos de OO em banco de dados
  - Usados como ferramenta de projeto mas depois codificado em um banco de dados relacional
  - Incorporados os conceitos de OO em uma linguagem que é usada para manipular no BD

19

Linguagens de Programação Persistentes

- Linguagens de BD x Linguagens de Programação
- Linguagens de BD manipulam dados persistentes
- O acesso ao BD é somente um componente de qualquer aplicação
- Linguagem de manipulação de dados: SQL
- Linguagens de programação: C++, Smalltalk
- Uma *linguagem de programação persistente* é uma linguagem de programação estendida com estruturas para tratar dados persistentes.

20

Linguagens de Programação Persistentes

- Linguagens com SQL embutida
  - O programador é responsável por qualquer conversão de tipos
  - O código para conversão entre objetos e tuplas opera fora do paradigma orientado a objeto
  - Esta conversão exige uma quantidade substancial de código
  - O programador é responsável por escrever o código explícito para mover dados a partir do BD para dentro da memória
  - O código para as atualizações no BD deve ser explícito

21

Linguagens de Programação Persistentes

- Linguagens de Programação Persistentes
  - A linguagem de manipulação de dados e a linguagem host compartilham o mesmo sistema
  - Objetos podem ser criados e armazenados no BD sem qualquer tipo explícito ou mudança de formato
  - Quaisquer mudanças de formato exigidas são executadas de maneira transparente
  - O código para manipulação dos dados é implícito

22

Persistência de Objetos

- Formas de tornar os objetos persistentes:
  - **Persistência por classe**
    - Abordagem simples e menos conveniente
    - Declarar se uma classe é persistente ou transiente
  - **Persistência por criação**
    - Um objeto é persistente ou transiente conforme foi criado

23

Persistência de Objetos

- **Persistência por marcação**
  - Todos os objetos são criados como transientes
  - Objeto é marcado como persistente após sua criação
- **Persistência por referência**
  - Um ou mais objetos são explicitamente declarados como objetos persistentes (raiz)
  - Todos os outros objetos são persistentes se forem referidos diretamente, ou indiretamente, a partir da raiz

24

Identidade de Objetos

- Quando um objeto é criado recebe um identificador de objeto persistente
- Se a linguagem não suportar persistência, recebe um identificador de objeto transiente
- Em muitas linguagens OO, um identificador de objeto é um ponteiro de memória

25

Identidade de Objetos

- Graus de permanência de identidade:
  - **Intraprocedimento**
    - Ex: variáveis locais de procedimento
  - **Intraprograma**
    - Ex: variáveis globais
  - **Interprograma**
    - Persiste de uma execução de programa para outra
  - **Persistência**
    - A identidade persiste entre reorganizações estruturais de dados. É a forma exigida para sistemas orientados a objetos.

26

Armazenamento e Acesso a Objetos Persistentes

- Parte de dados de um objeto tem de ser armazenada individualmente para cada objeto
- Pode-se dar nomes ao objeto ou alocar ponteiros para os objetos ou armazenar conjuntos de objetos
- Um caso especial de um conjunto é uma classe extensão

27

Linguagem de Definição de Objeto

- Grupo de Gerenciamento de Banco de Dados Objeto (ODMG)
  - Padronizar extensões de linguagem C++ e Smalltalk a fim de aceitar persistência e definir bibliotecas de classe para aceitar persistência
- Há duas partes para a extensão C++ ODMG
  - Linguagem de Definição de Objeto C++ (C++ ODL)
  - Linguagem de Manipulação de Objeto (C++ OML)

28

Linguagem de Definição de Objeto

```
class Pessoa : public Objeto_Persistente{
Public:
   String nome;
   String endereco;
};

class Cliente : public Pessoa{
public:
   Date membro_desde;
   int cliente_id;
   Ref<Agência>origem_agência;
   Set<Ref<Conta>>contas inverse Conta::Proprietários;
};
```

29

Linguagem de Definição de Objeto

```
class Agência : public Objeto_Persistente{
public
   String nome;
   String endereço;
   int ativos;
};

class Conta : public Objeto_Persistente{
private:
   int saldo;
Public:
   int numero;
   Set<Ref<Cliente>>proprietários inverse Cliente::contas;
   int encontra_saldo();
   int atualiza_saldo(int delta);
};
```

30

Linguagem de Manipulação de Objeto

```
int cria_proprietário_conta(string nome, String endereco){
Database * bank_db;
bank_db = Database::open(“Bank-DB”);
Transaction Trans;
Trans.begin;

Ref<Conta> account = new(bank-db) Conta;
Ref<Client> clie = new (bank-db) Cliente;
clie->nome = nome;
clie->endereco = endereco;
clie->conta.inserir_elemento(conta);
Clie->proprietario.inserir_elemento(clie);
... Código para inicializar cliente_id, numero conta, etc
Trans.commit();
```

31

Bibliografia

Silberschatz, Abraham; Korth, Henry F. e Sudarshan, S. Sistema de Banco de Dados. São Paulo: Markon Books, 1999.

Elmasri, Ramez; Navathe, Shamkant B. Fundamentals of Database Systems. Addison-Wesley, 2000.

32