Unidade VIII

Bancos de Dados Orientados a Objeto

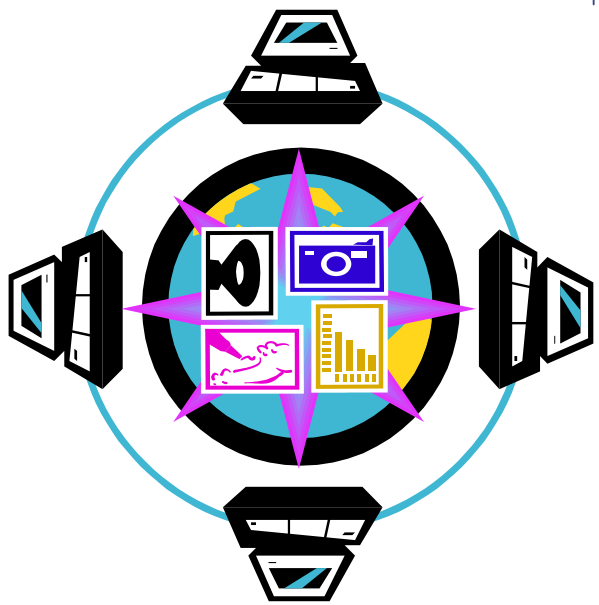
- Justificativa
- Descrição do Modelo
- Linguagens

Justificativa

- BDs relacionais possuem limitações
  - Poucos tipos de dados
  - Dado geralmente pequeno, com tamanho predefinido
  - Valor de um dado é atômico (1ª forma normal)
  - São inapropriados para armazenamento e busca de novos tipos de dados como imagens, áudio e vídeo
- Novos modelos vêm sendo propostos para solucionar as limitações dos BDs relacionais
  - Modelo Orientado a Objeto
  - Modelo Relacional-Objeto
  - Modelo Dedutivo
  - ...

Justificativa

- BDs não-convencionais são úteis em uma série de novas aplicações
  - Hipertexto
  - Multimídia
  - Ensino à distância
  - Medicina
  - CRM
  - CAD
  - CASE
  - ...

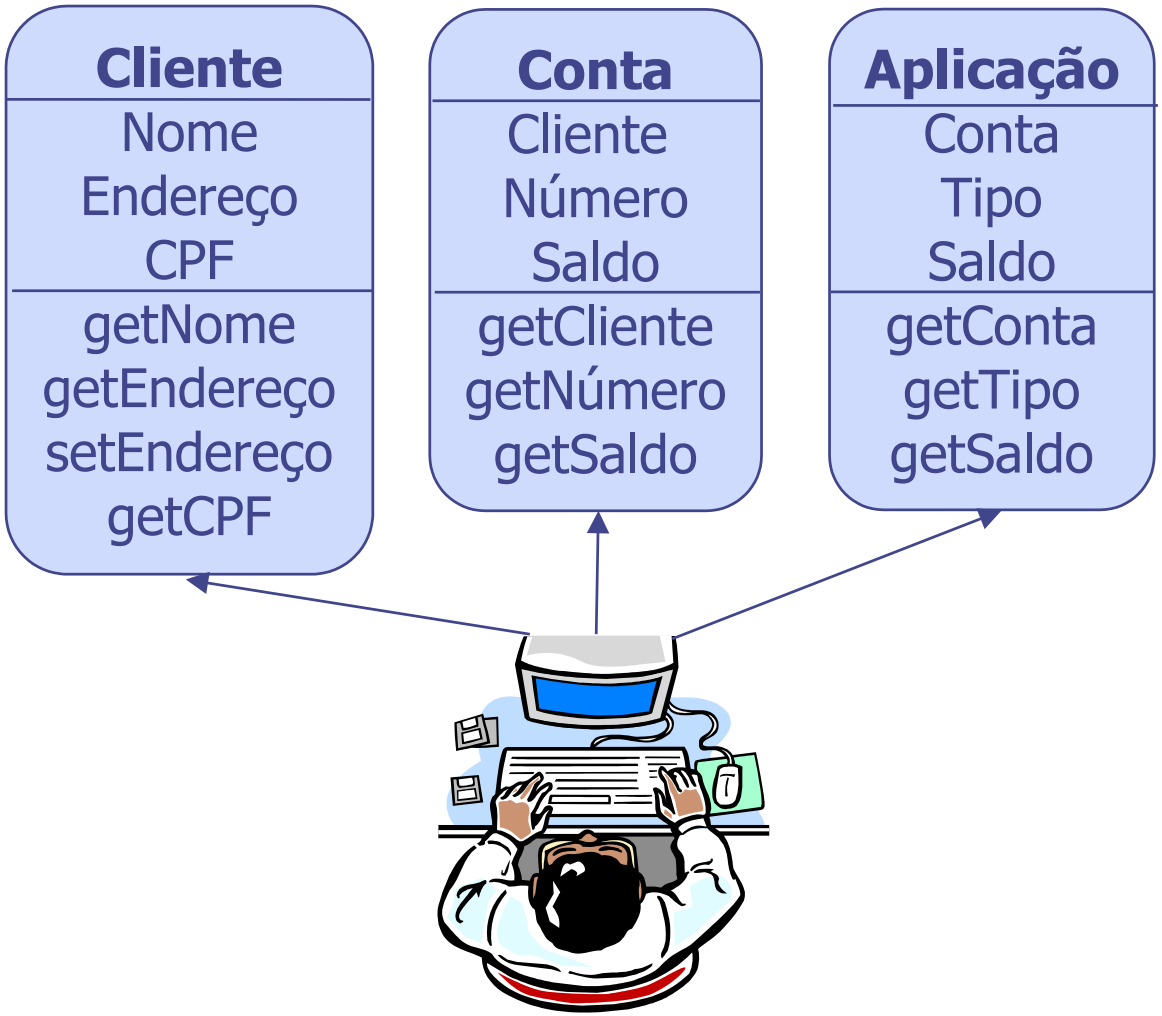


- BDs relacionais não conseguem modelar os dados destas aplicações adequadamente

Descrição do Modelo

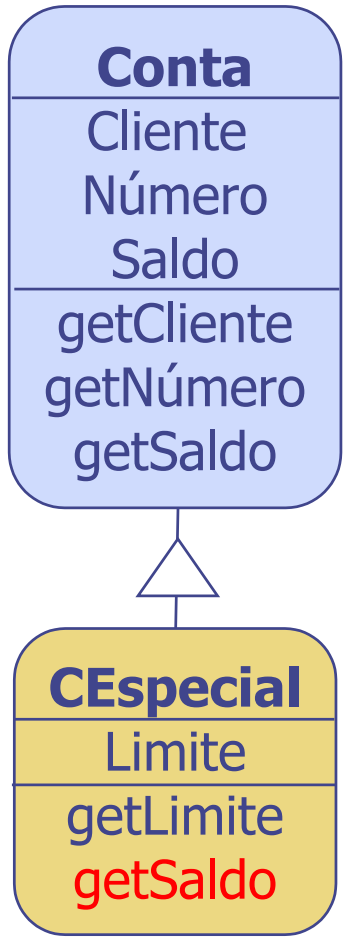
- Dados são modelados como objetos
  - Um Objeto equivale a uma entidade do modelo E-R
  - Objetos encapsulam um conjunto de Variáveis que contém os dados do objeto
  - Métodos são usados para acessar e alterar os dados
  - Mensagens são trocadas entre objetos para ativar seus métodos
    - Carregam os Parâmetros de chamada do método
    - Geram uma Resposta com o retorno do método
    - O envio de mensagens não implica na transmissão de dados pela rede (pode ser implementado como uma chamada de método local)

Descrição do Modelo



Descrição do Modelo

- A classe de um objeto define a sua interface externa
  - Variáveis, métodos e mensagens aceitas
- Podem existir relações de herança entre classes
  - Herança de interface e implementação
  - Ex: Conta → Conta Especial (com limite)
- Métodos podem ter implementações diferentes nas subclasses
  - Ex: getSaldo() da conta especial pode incluir o limite de crédito no cálculo do saldo disponível



Descrição do Modelo

- Identidade
  - Objetos são identificados univocamente no sistema, sem usar para tal os valores dos dados
  - Já em BDs relacionais, a identidade de uma tupla é dada pelo valor de um dado (ex.: chave primária)
- Referências
  - Objeto podem ter referências para outros objetos
    - Ou seja, uma variável de um objeto podem ter como valor a identidade de um outro objeto
    - No exemplo anterior, Conta → Cliente
  - Em BDs relacionais, uma tupla precisa conter um valor chave de outra tupla e usá-lo para localizá-la

Linguagens

- Diferentes abordagens podem ser usadas para acessar BDs orientados a objeto:
  - Usar orientação a objeto somente para projeto e acessá-lo como um BD relacional
  - Acessar o BD usando uma linguagem que entenda o conceito de objeto
    - Linguagem relacional-objeto: linguagem relacional (geralmente SQL) com extensões para manipular objetos e construir novos tipos de dados
    - Linguagem de programação persistente: LPOO (C++, Java, ...) com extensões para uso em BDs

Linguagens

- SQL 1999
  - Padrão de linguagem relacional-objeto
  - Ainda não é suportada por muitos BDs comerciais
  - Permite a construção de tipos com **create type**
    Ex.: **create type** Pessoa
    (Nome **varchar**, CPF **integer**)
    Obs.: o nome é referenciado como Pessoa.Nome
  - Suporta herança simples, arrays, objetos binários (**Blobs**) e de caracteres (**Clobs**)
    Ex.: **create type** Aluno **under** Pessoa
    (Matrícula **integer**, Curso **varchar**
    Disciplinas **varchar array[10]**,
    Foto3x4 **Blob**(32Kb), Histórico **Clob**(16Kb))
    Obs.: elementos do array são acessados com [n]

Linguagens

- SQL 1999 (cont.)
  - Podemos criar tabelas e sub-tabelas
    Ex.: **create table** Alunos **of** Aluno
    create table Professores **of** Pessoa
    create table Bolsistas **of** Aluno **under** Alunos
  - Arrays são inseridos usando **array**[…]
    Ex.: **insert into** Alunos **values**
    (('João', 12345678900, 01136400,'SIN'),
    20021, **array**['INE5612','INE5613'],
    '10110…', 'Histórico…')
  - Referências apontam para objetos
    Ex.: **create table** Disciplina (Nome **varchar**,
    Professor **ref**(Pessoa) **scope** Professores,
    Turma **ref**(Aluno) **array**[50] **scope** Alunos)

Linguagens

- SQL 1999 (cont.)
  - Referências são adicionadas explicitamente à tabela
    Ex.: **create table** Alunos **of** Aluno
    ref is oid **system generated**
  - Referências são obtidas como atributos da tabela
    Ex.: **select** oid **from** Alunos **where** Nome = `João'
    Obs.: Oracle e alguns outros BDs usam **ref**(obj)
  - Métodos manipulam os dados
    Ex.: **create method** TransfInterna (matric **integer**,
    curso varchar) **for** Aluno
    begin
    set self.Matrícula = matric;
    set self.Curso = curso;
    end

Linguagens

- Linguagens Persistentes ODMG
  - ODL (Object Definition Language)
    - Permite que sejam definidos dados e objetos persistentes em uma LPOO
  - OML (Object Manipulation Language)
    - Fornece mecanismos para que LPOOs manipularem bancos de dados persistentes criados em ODL
  - OQL (Object Query Language)
    - Permite que LPOOs consultem bancos de dados persistentes criados em ODL, com a ajuda da OML
  - São mapeadas para C++, Java e Smalltalk

Linguagens

- ODMG ODL C++
  - Possui tipos de dados persistentes
    - **d_Short**, **d_UShort** - 16 bits
    - **d_Long**, **d_ULong** - 32 bits
    - **d_Float** - 32 bits, formato IEEE
    - **d_Double** - 64 bits, formato IEEE
    - **d_Char**, **d_String** - 8 bit, formato ASCII
    - **d_Octet** - 8 bit, sem formato predefinido
    - **d_Boolean** – 1 bit, 0 (falso) ou 1 (verdadeiro)
  - Permite definir classes de objetos

```
Ex.: class Pessoa : public d_Object {
        public:   d_String Nome;
        private: d_Long CPF;
       };
```

Linguagens

- ODMG ODL C++ (cont.)
  - Permite criar conjuntos de objetos
    Ex.: **class** CorpoDocente : **public d_Object** {
    public: **d_Set**<Pessoa> Professores;
    };
  - Podem existir relações de herança entre objetos
    Ex.: **class** Aluno : **public** Pessoa {
    public: **d_Long** Matrícula;
    };
  - Referências apontam para objetos
    Ex.: **class** Turma {
    public: **d_String** Disciplina;
    d_Ref<Pessoa> Professor;
    d_Set<**d_Ref**<Aluno>> Alunos;
    };

Linguagens

- Exemplo de código em ODMG OML C++

```
void novo_aluno (d_String Nome, d_Long CPF,
  d_Long Matrícula) {
  d_Database db;
  d_Transaction trans;
  db.open("AlunosDB");
  trans.begin();
  d_Ref<Aluno> NovoAluno = new(db) Aluno;
  NovoAluno→Nome = Nome;
  NovoAluno→CPF = CPF;
  NovoAluno→Matrícula = Matrícula ;
  trans.commit();
  db.close();
}
```

Linguagens

- Exemplo de código em ODMG OQL C++

```
void busca_prof (d_String nome) {
  d_Database db;
  db.open("CorpoDocenteDB");
  d_OQL_Query query("select Professores from
     CorpoDocente where Professores.Nome like :nome")
  d_Set<d_Ref<Pessoa>> profs;
  d_OQL_Execute(query, profs);
  d_Iterator<d_Ref<Pessoa>> iter =
    profs.create_iterator();
  while(iter.next(profs)) {
    cout << "Nome: " << profs.Nome
            << "   CPF: " << profs.CPF << endl;
  }
  db.close();
}
```