

Banco de Dados I

Bancos de Dados Orientados a Objetos

Modelo Relacional

- Definido para aplicações convencionais (administração, contabilidade, cadastro,...)
 - folha pçto., controle estoque, automação bancária,...
 - Características destas aplicações
 - dados com formato fixo (clientes, contas, produtos,...)
 - tipos de dados simples (inteiros, reais, strings,...)
 - consultas geralmente simples (baseadas em chaves)
 - transações de curta duração (p.ex.: transações bancárias)
- Adequadas com representações tabulares

Modelo Relacional - Limitações

- Relações devem estar na 1 FN (valores atômicos)
 - o que não "casa" bem deve ser decomposto
exemplo: pessoa com filhos
- Homogeneidade Horizontal (todas as tuplas com os mesmos atributos)
 - força padronização (ex. roupas de tipos diferentes)
 - soluções desagradáveis:
 - * todos os atributos possíveis

Modelo Relacional - Limitações

- Homogeneidade Vertical (um atributo com o mesmo tipo de informação em todas as tuplas)
 - c.ex.: tamanho de roupa, veículos locados para pessoas ou empresas
 - soluções desagradáveis:
 - * atributos extras para todos os tipos possíveis
 - * tipos genéricos (pode exigir conversão)

Aplicações Não Convencionais

- Avanço tecnológico de HW viabiliza o desenvolvimento de novas aplicações de porte
- Caracterizam-se por presença de muitos dados e operações complexas
- Exemplos: projeto (CAD/CAM), ambientes de desenvolvimento de SW (CASE), multimídia (hipertexto, SIG), SAD

Aplicações Não Convencionais - Aspectos

- Projeto / Desenvolvimento SW:
 - Entidades com estrutura complexa, que podem ser decompostas em entidades menores
 - Podem ter várias representações e versões;
- Multimídia:
 - Tipos de dados não usuais (imagem, áudio, vídeo,...)
 - Armazenados como seqüências de bytes ou convertidos para outros formatos estruturais
- Sistemas de Apoio à Decisão:
 - Armazenam dados e regras de dedução
 - Não "casam" bem com BDs relacionais

Conceitos de OO em BD

Paradigma de Orientação a Objetos

- Empregado em várias áreas da computação
 - LPs (pioneira), IU, ES, BD, ...
- Permite o desenvolvimento de sistemas de forma fácil, natural e modular
 - Semântica vinculada ao conceito de objeto
- Baseia-se em 2 conceitos:
 - Encapsulamento
 - Extensibilidade

Paradigma de Orientação a Objetos

- Encapsulamento
 - *Objeto*: entidade da realidade que encapsula estrutura (dados) e comportamento (programas ou métodos)
 - Objeto recebe e responde a mensagens através de uma interface pública (manipulação do objeto)
 - Exemplo: objeto **Empregado**
 - dados: Nome, DataNasc., Endereço, Filhos, Salário, Depto., ...
 - programas: ObterIdade, TotalFilhos, AumentaSalário, MudaDepto., AlteraEndereço, Admite, Demite, ... (interface)
 - Vantagens: aplicação não programa a definição e manipulação de dados; desenvolvimento facilitado

Paradigma de Orientação a Objetos

- Extensibilidade
 - Habilidade de estender um sistema já existente de forma transparente
 - Preocupação com a evolução do sistema
 - Formas de extensibilidade:
 - Extensão da definição (inclusão de novos dados e métodos)
 - Instanciação (objetos gerados com a mesma estrutura e comportamento)
 - Herança (definição de um objeto é aproveitada na definição de objetos mais especializados)

Banco de Dados Orientado a Objetos

- **Objetivo**: modelar objetos do mundo real de forma mais direta
- **Definição**:
 - É um sistema com as funcionalidades e características de uma LPOO e de um SGBD
- **O projeto de um BDOO implica a integração de tecnologia de banco de dados com tecnologia orientada a objetos**

Banco de Dados Orientado a Objetos

- **Modelo de dados OO**
 - Falta de consenso sobre um padrão a seguir (uso de conceitos, quais conceitos aplicar,...)
 - Falta de uma base formal (atividade experimental voltada às necessidades das aplicações)
 - Propostas: extensões do modelo relacional, extensões de LPOOs, "começar do zero", ...
- **Padrão UML**
- **ODMG**

Banco de Dados Orientado a Objetos

- **Diversos sistemas experimentais e comerciais**
 - GemStone
 - ORION
 - O2
 - ONTOS
 - ENCORE
 - Object Store
 - IRIS
 - Objectivity
 - Jasmine

Conceitos de OO em BD

1) Objeto e Identidade do Objeto (OID)

- **Objeto**: tratamento uniforme dado a qualquer entidade do mundo real
- **Identidade do objeto (OID)**
 - Identificação interna única, gerada pelo SGBD
 - Identificação independente de valor (inalterável)
 - Permite compartilhamento de objetos (através de referências)

Identidade do Objeto (OID)

- Desvantagens da identificação por valor (chave)
 - Uma chave é definida pelo valor de um ou mais atributos e pode sofrer modificações
 - Tipos de dados das chaves não são uniformes
 - Programadores/Administradores do BD devem preocupar-se com:
 - selecionar o subconjunto apropriado de atributos para usar como chave
 - fornecer valores únicos para as chaves
 - Chaves não são únicas no sistema - (relação)
 - OIDs são únicos no sistema

Identidade do Objeto (OID)

- Desvantagens da identificação por valor (chave)
 - Objetos relacionados não estão presentes dentro do objeto (menos natural, força junções)

Relacional	OO
<pre>select empregados.nome, deptos.nome from empregados, deptos where empregados.codd = deptos. codd</pre>	<pre>select empregados.nome, empregados.depto.nome from empregados</pre>

Identidade do Objeto (OID)

- Igualdade de identidade (= =)
 - Um objeto referido por mais de um objeto é o mesmo?
 - Igualdade de identidade \neq Igualdade de valor (=)
 - Exemplo: pessoa com nome, idade e filhos (visão OO)
(Pedro Silva,42,{João Silva,15,{}}) e
(Maria Silva, 38,{João Silva, 15,{}})
 - João Silva é filho de Pedro Silva e Maria Silva?
OID(João Silva em Pedro Silva)
= =
OID(João Silva em Maria Silva) ?

Identidade do Objeto (OID)

- Identidade (= =)
 - se dois objetos possuem o mesmo OID
- Igualdade (=)
 - se dois objetos possuem recursivamente o mesmo valor
 - estrutura deve ser analisada

<table><tr><td>Livro[1]</td></tr><tr><td>Título: Banco de Dados</td></tr><tr><td>Autores: Autor[1]</td></tr></table>	Livro[1]	Título: Banco de Dados	Autores: Autor[1]	<table><tr><td>Livro[2]</td></tr><tr><td>Título: Banco de Dados</td></tr><tr><td>Autores: Autor[2]</td></tr></table>	Livro[2]	Título: Banco de Dados	Autores: Autor[2]
Livro[1]							
Título: Banco de Dados							
Autores: Autor[1]							
Livro[2]							
Título: Banco de Dados							
Autores: Autor[2]							
<table><tr><td>Autor[1]</td></tr><tr><td>Henry Korth</td></tr><tr><td>Abraham Silberchatz</td></tr></table>	Autor[1]	Henry Korth	Abraham Silberchatz	<table><tr><td>Autor[2]</td></tr><tr><td>C. Date</td></tr></table>	Autor[2]	C. Date	
Autor[1]							
Henry Korth							
Abraham Silberchatz							
Autor[2]							
C. Date							

Livro[1] \neq Livro[2]

Identidade do Objeto (OID)

- Formas de determinação do OID
 - OID = ID-Classe + ID-Objeto
 - OID como info. única (seqüencial, p.ex.)
 - OID = RID (identificação física)

Conceitos de OO em BD

2) Classes

- **Classificação:** Agrupa objetos com a mesma estrutura e comportamento

- **Instanciação**

→ **Reusabilidade:** classe guarda a definição das instâncias; instâncias compartilham código/interface dos métodos

Classes - Questões em Aberto

- Atributos/métodos da classe ? (classe objeto)
Válidos para a classe e não para as instâncias
Capturam informações do conjunto de instâncias
(total pessoas, peso médio, mais idosa, ...)

- **Metaclasses:** classe da classe?

Classes pré-definidas, em geral (raízes na hierarquia)

Conceitos de OO em BD

3) Objetos Complexos

- Objetos Atômicos

true, false, 25, “este é um objeto atômico”

- Dois tipos de objetos complexos: não-estruturados e estruturados

- Objetos complexos não-estruturados

- Objetos grandes “não interpretados”, como imagens, voz, etc.

- SGBD não “entende” ou “interpreta” estes objetos

- deve ser feito pelo programa de aplicação

Objetos Complexos

- Objetos complexos estruturados

Obtidos com os construtores de objetos
SGBD conhece a estrutura do objeto

- Construtores de Objetos

tupla [Nome: Luís, Sobrenome: Silva]

conjunto {João, Ana, Maria}

Lista [25, 20, 117]

- Vantagens de um modelo que suporte objetos complexos

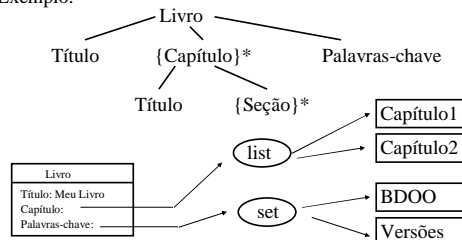
- representação direta dos objetos

- sem necessidade de decomposição em unidades

- recuperação de dados e navegação rápidas

Objetos Complexos

Exemplo:



Conceitos de OO em BD

4) Métodos

- Conjunto de procedimentos associados a um objeto (*ações que caracterizam o comportamento*)

- Dois Componentes

→ **Assinatura** (*nome método, argumentos, resultado*)

→ **Implementação** (*em geral, mesma LP para aplicações*)

- **Mensagens:** ativam métodos via interface pública

- Em geral, atributos acessíveis só via métodos

→ **Vantagem:** *encapsulamento*

→ **Desvantagem:** *limita o acesso*

Conceitos de OO em BD

5) Hierarquia de Classes e Herança

- Classes (*subclasses*) podem ser definidas a partir de outras classes (*superclasses*)
- Construção de uma hierarquia com herança de propriedades (*atributos e métodos*) pelas subclasses; estas podem definir propriedades adicionais
- Reusabilidade: evita redefinição de propriedades

Hierarquia de Classes e Herança

- Propriedades herdadas podem ser redefinidas
- *Questões*:
 - * Métodos que fazem acesso a atributos redefinidos? (*exige redefinição do método.*)
 - * Métodos redefinidos ?
(*inicia-se a busca do método sempre pela própria classe do objeto, continuando em direção às superclasses caso o mesmo não seja encontrado*)

Conceitos de OO em BD

6) Late Binding, Overriding e Overloading

- Overloading (sobrecarga): uso de um mesmo nome para mais de um método em classes \neq
- Overriding (redefinição): redefinição da implementação do método para cada classe
- Exemplo: operações como *new*, *remove*
- Late Binding (ligação tardia): seleciona o código para a execução do método somente em tempo de execução

Late Binding, Overriding e Overloading

- Late Binding
→ *Conforme os argumentos da mensagem, o método de uma dada classe é selecionado*
- Sem estes conceitos: vários nomes e assinaturas

<i>case classe of</i>	<i>Print(obj,...);</i>
<i>documento: PrintDoc(obj,...);</i>	<i>(ex.: LP Pascal e a função</i>
<i>imagem: PrintImage(obj,...);</i>	<i>Write p/ vídeo e arquivo)</i>
<i>pessoa: PrintPessoa(obj,...);</i>	
<i>end;</i>	
- Caracteriza o polimorfismo (*função aplicada a uma variedade de objetos com comportamentos diferentes*)

3. Linguagens - Interface Básica

- Abordagens a seguir:
→ *LBD embutida x LPOO estendida*
- Interface com um SGBD: DDL, DML, DCL
→ *flexibilidade, consistência c/ paradigma mensagens*
- Envio de mensagens (ORION)
→ *sintaxe básica: (seletor receptor [arg1 arg2 ... arg n])*
seletor: nome da mensagem - receptor: objeto
- Exemplos *SGBDOO*

Linguagens - Interface Básica

1) DDL ORION

- Sintaxe básica:
(*make-class NomeClasse [:superclasses ListaSuper]*
[:attributes ListaAtributos]
[:methods ListaEspMétodos])
- Superclasses: *controle de herança*
- Atributos: *nome, domínio, herança*
- Métodos: *nome, herança*

Linguagens - Interface Básica

O2: Exemplo

```
add class Projeto
  type tuple ( NomeProjeto: string,
              Artigos: set(Document),
              Tarefas: set(Tarefa),
              Gerente: Empregado)
  method adiciona_tarefa(tarefa:Tarefa): set(Tarefa);

method body adiciona_tarefa (tarefa:Tarefa): set(Tarefa)
in class Projeto
{ ... corpo do método...}
```

Linguagens - Interface Básica

2) DML

ORION

- Operações de manipulação de instâncias
→ envio de mensagens à classe do objeto
- I: (make NomeClasse :Atributo1 valor1 ...
:Atributo n valor n)
- C: (select NomeClasse ExpressãoConsulta)
- E: (delete NomeClasse ExpressãoConsulta)
(delete-object Objeto)
- A: (change NomeClasse [ExpressãoConsulta]
NomeAtributo NovoValor)

Linguagens - Interface Básica

O2

- Extensões da classe são armazenadas como conjuntos, gerenciados pelo usuário
- Objetos com nome são persistentes
- Exemplo:
add name Projetos: set(Projeto)
/* Cria um conjunto de projetos */

Projetos = set(); /* inicializa o conjunto de projetos*/
tmp = new(Projeto); /* cria um novo projeto */
Projetos += set(tmp); /*adiciona um novo projeto ao conjunto*/

Linguagens - Interface Básica

O2

```
SELECT tuple (NomeProjeto: x.NomeProjeto,
              NomeGerente: x.Gerente.Nome)
FROM x in Projetos
WHERE x.Gerente.Salário > 3000
```

Processamento de Consultas

- Muitas semelhanças com o processamento em BD relacionais
- Seleção, projeção, junção
- Otimização de operações
- Computar possíveis combinações de classes, eliminando os produtos cartesianos
- Geração de um plano de execução: consideração de índices, método de concatenação de instâncias
- Estimar custo: estimativas do BD

Processamento de Consultas

- Formas de concatenação de instâncias (supondo classe C com atributo a que faz referência à classe D):
- 1) Ordem de avaliação: $C \rightarrow D$
 - a) busca-se uma instância de C ;
 - b) o atributo a é extraído;
 - c) busca-se uma instância de D com OID idêntico;
- 2) Ordem de avaliação: $D \rightarrow C$
 - a) busca-se uma instância de D ;
 - b) o seu OID é extraído;
 - c) busca-se uma instância de C com este OID no atributo a ;

Processamento de Consultas

- Hierarquias de classes
 - *Estatísticas adicionais são úteis: nro. instâncias e valores distintos de atributos em cada classe, páginas de disco que as mantém,...*
 - *Índices de hierarquia de classes: muitas vezes melhor do que índices separados por classe.*

Processamento de Consultas

- Métodos em consultas
 - *Influenciam na avaliação da consulta (seleção de objetos que satisfazem os predicados)*
 - *Difícil otimização → método é um programa, como estimar o seu custo?*
 - *Estratégia mais comum:*
 - * *Avaliação tardia de predicados com métodos*

Gerência de Transações

- Transação: seqüência de R/W sobre o BD
- Propriedades: (ACID)
 - *Atomicidade (tudo ou nada)*
 - *Consistência (preservar integridade)*
 - *Isolamento (serializabilidade)*
 - *Durabilidade (persistir a futuras falhas)*
- Concorrência → *bloqueio*
- Recovery → *logging (undo e redo)*

Gerência de Transações

- BDOO
 - *Hierarquia de classes*
 - * *bloqueio da hierarquia, classe e instâncias?*
 - *Aplicações de projeto:*
 - * *Transações de longa duração (atomicidade?)*
 - * *Cooperação entre projetistas*
 - * *Versões*

Transações

- Transações Longas
 - *Típicas de aplicações não convencionais (ex.:CAD)*
 - *Transação dividida em transações filhas, podendo existir vários níveis de aninhamento*
 - *Transação respeita ACID dentro da transação longa*
 - *Transação longa respeita ACID em relação a outras transações longas*
 - *Quando existe versionamento de objetos, não existem longas esperas: cópias são feitas e novas versões serão geradas*

Os treze compromissos de um OODBMS

- M. Atkinson, F. Banchilhon, D. DeWitt, K. Dittrick, D. Maier, and S. Zdonik
- “Object-Oriented Database System Manifesto”
- 1989
- First International Conference in Deductive and Object-Oriented Databases
- Kyoto, Japan

- Manifesto
 - 13 características obrigatórias
 - algumas características opcionais
- Características Obrigatórias
 - 8 regras relacionadas com OO
 - 5 regras relacionadas com DBMSs

Regras Obrigatórias

- Regra 1
 - suporte a objetos complexos
- Regra 2
 - suporte a identificação de objeto (OID)
- Regra 3
 - os objetos devem ser encapsulados.
 - Objetos tem uma interface pública, mas a implementação deve ser privada

Regras Obrigatórias

- Regra 4
 - suporte a classes
- Regra 5
 - Suporte a herança
- Regra 6
 - *Late Biding*

Regras Obrigatórias

- Regra 7
 - o sistema deve ser computacionalmente completo
 - todos os tipos de operações sobre um banco de dados devem poder ser expressas através da sua linguagem de programação

Regras Obrigatórias

- Regra 8
 - suporte a extensibilidade
- Regra 9
 - armazenar dados permanentemente
 - dados persistentes
- Regra 10
 - suporte a VLDB (*Very Large Databases*)

Regras Obrigatórias

- Regra 11
 - Suporte a acesso concorrente
- Regra 12
 - *recovery*
- Regra 13
 - Linguagem de consulta deve ser simples

Regras opcionais

- Suporte à herança múltipla
- Suporte à distribuição
- Suporte a versões

Vantagens e Desvantagens

- OODBMS possuem diversas vantagens e desvantagens sobre sistemas convencionais
- A maioria destas características está baseada nos conteúdos vistos anteriormente

Vantagens

- permitem a inclusão de mais informações semânticas no banco de dados, permitindo uma melhor representação da realidade
- Suporte a objetos complexos, o que facilita a sua utilização em aplicações específicas
- extensibilidade
- suporte a versões
- reusabilidade

Vantagens

- menor tempo de desenvolvimento
 - utilizando de forma apropriada a herança
 - utilizando um metodologia de projeto OO
- Capacidade de abstração
 - integração com sistemas já existentes

Desvantagens

- resistência por parte dos usuários
- falta de formalização
- pointers
- falta de uma linguagem de consulta
- Falta de ferramentas para projetar e avaliar
- tempo de aprendizado
- falta de profissionais qualificados