FACULDADE DE ADMINISTRAÇÃO E NEGÓCIOS DE SERGIPE – FANESE

MBA EM ADMINISTRAÇÃO DE BANCO DE DADOS

JOÃO INÁCIO DE MELLO FILHO

MODELO DE DADOS: comparação do modelo de banco de dados orientado a objetos em relação ao modelo objeto relacional

Aracaju – SE
2016.1

João Inácio de Mello Filho*

MODELO DE DADOS: comparação do modelo de banco de dados orientado a objetos em relação ao modelo objeto relacional

Trabalho de Conclusão de Curso apresentado ao Núcleo de Pós-Graduação e Extensão – NPGE, da Faculdade de Administração de Negócios de Sergipe – FANESE, como requisito para a obtenção do título de Especialista em Banco de Dados.

_____

Nome completo do Avaliador

_____

Nome completo do Coordenador de Curso

_____

Nome completo do Aluno

Aprovado (a) com média: _____

Aracaju (SE), ____ de _____ de 2016.

* Desenvolvedor de Sistemas, Graduado em Sistemas para Internet, Faculdade de Administração de Negócios de Sergipe – FANESE

E-mail: joaoinaciomello@gmail.com

RESUMO

A comparação dos modelos de dados orientados a objetos em relação aos objetos relacionais é apresentada neste artigo. Esta análise possibilita evidenciar as vantagens e desvantagens na utilização de cada um dos modelos. Serão apresentados diversos aspectos do banco de dados orientados a objetos, sua origem, identidade de objeto, estruturas e construtores de tipos, encapsulamento de operações, métodos e persistência, comportamento do objeto, persistência de objeto na base de dados, hierarquias de classe, tipo, herança múltipla e herança seletiva, entre outros. Assim como do banco de dados objeto relacional, origem, evolução a partir Sistemas Gerenciadores de Banco de dados, características, gerenciamento de objetos longos e armazenamento adicionais, implementação e aspectos relacionados a sistemas de tipos estendidos, modelo relacional aninhado. Dados apresentação das diferenças entre os bancos de dados, a escolha da opção que melhor atenda suas necessidades específicas é facilitada.

Palavras chaves: Modelo.Arquitetura.Objetos.

1 INTRODUÇÃO

A comparação dos modelos de banco de dados orientado a objetos em relação aos objetos relacionais resulta nas diferenças, características, funções, vantagens e desvantagens de cada banco de dados, permitindo ao usuário optar pelo mais prático e que atenda suas necessidades.

Atualmente é cada vez maior a demanda por um SGBD (Sistema Gerenciador de Banco de Dados) que seja confiável e que possa ser o mais transparente possível para o paradigma de Orientação a Objeto.

Os fabricantes de SGBDs na tentativa de atender aos sistemas orientados a objetos, adicionaram aos Bancos de Dados Relacionais recursos de Orientação a Objeto, surgindo o Banco de Dados Objeto Relacional, ou Relacional Estendido, a exemplo do Oracle 9i e PostgreSQL 8.0.

2 BANCO DE DADOS ORIENTADOS A OBJETOS

> O modelo de banco de dados orientado a objetos foi originado para atender as aplicações mais complexas, que necessitam de requisitos e características como: estruturas complexas para objetos, transações de longa duração, novos tipos de dados para armazenamento de imagens ou textos, e definir operações não convencionais específicas das aplicações.
> Uma característica chave dos bancos de dados orientado a objetos é o poder dado ao projetista para especificar tanto a estrutura de objetos complexos quanto as operações que podem ser aplicadas a esses objetos. Outra razão para a criação de banco de dados orientado a objetos é o uso crescente de linguagens de programação orientadas a objetos no desenvolver de aplicações de software.
> O termo orientado a objetos, abreviatura como OO ou O.O, tem sua origem nas linguagens de programação OO, ou LPOOs. Atualmente, os conceitos Orientados a Objetos são aplicáveis nas áreas de bancos de dados, engenharia de software, bases de conhecimento, inteligência artificial e sistemas computacionais em geral. Um dos objetivos dos bancos de dados Orientados a Objetos é manter a correspondência direta entre objetos do mundo real e do banco de dados, assim esses objetos não perdem sua integridade e identidade e podem ser facilmente identificados e acessados. Então, o banco de dados Orientado a Objetos fornece um identificador do objeto (OID), único gerado pelo sistema para cada objeto.(ELMASRIR; NAVATHE, 2005, p.724)

2.1-Identidade de objeto, estrutura de objeto e construtores de tipos.

2.1.1- Identidade de Objeto

Um sistema de banco de dados orientado a objetos fornece uma identidade única para cada objeto independente armazenado no banco de dados. Essa identidade única é geralmente implementada por meio de um identificador de objeto único gerado pelo sistema ou OID. O valor de um OID não é visível para um usuário externo, mas é utilizado internamente pelo sistema para identificar univocamente cada objeto e para criar e gerenciar referências Inter objetos.

A principal propriedade exigida de um OID é que ele seja imutável, isto é, o valor do OID para um objeto particular não deve ser modificado. Isso preserva a identidade do objeto do mundo real que está sendo representado. Desse modo, um sistema de banco de dados orientado a objetos deve possuir algum mecanismo para gerar os OIDs e garantir a propriedade da imutabilidade. Também é desejável que cada OID seja utilizado apenas uma vez, ou seja, mesmo que um objeto seja removido do banco de dados seu OID não deve ser atribuído a outro objeto.

2.1.2-Estrutura de Objeto

O banco de dados orientado a objeto possui o estado (valor corrente) de um objeto complexo que pode ser construído a partir de outros objetos ou outros valores, utilizando alguns construtores de tipos. Um modo formal de representar tais objetos é visualizar cada objeto como uma tripla (i, c, v), na qual i é o identificador único do objeto (o OID), c é um construtor de tipo (ou seja, uma indicação de como o estado do objeto é construído) e v é o estado do objeto (ou valor corrente). O modelo de dados normalmente inclui vários construtores de tipos. Os três construtores mais básicos são atom (atômico), tuple (tupla) e set (conjunto). Outros construtores geralmente utilizados são list, bag e array. O construtor atom é utilizado para representar todos os valores atômicos básicos, como inteiros, números reais, cadeias de caracteres, booleanos e quaisquer outros tipos básicos que o sistema suporte diretamente. Os construtores de tipos set, list, array e bag são chamados tipos coleção (ou tipos empilhados) para distingui-los dos tipos básicos e dos tipos tuplas. A principal característica de um tipo coleção é que o estado de um objeto será uma coleção de objetos que podem ser não ordenados (como um do tipo set

ou do tipo bag) ou ordenados (como uma lista ou um array). O construtor de tipo tupla é frequentemente chamado tipo estruturado, uma vez que corresponde à construção struct nas linguagens de programação C e C++.

2.1.3-Construtores de Tipos

Os construtores de tipos podem ser utilizados para definir as estruturas de dados para um esquema de banco de dados orientado a objetos.

2.2 Encapsulamento de Operações, Métodos e Persistência

"Uma das principais características das linguagens e dos sistemas orientados a objetos, é o encapsulamento. Ele está relacionado também com os conceitos de tipos abstratos de dados e ocultar informação nas linguagens de programação."(1)

2.3 Especificando o Comportamento do Objeto Através de Operações de Classe

Os conceitos para ocultar a informação e encapsulamento podem ser aplicados a objetos de banco de dados. A ideia principal é definir o comportamento de um tipo de objeto com base nas operações que podem ser externamente aplicadas aos objetos daquele tipo. A estrutura interna do objeto é escondida e o objeto é acessível por meio de um número de operações predefinidas.

Os usuários externos de um objeto conhecem apenas a interface do tipo do objeto, ao qual define o nome e os argumentos (parâmetros) de cada operação. A implementação é oculta para os usuários externos, o que pode incluir a definição das estruturas internas de dados do objeto e a implementação das operações que acessam essas estruturas. Na terminologia orientação a objetos, a parte de interface de cada operação é chamada assinatura, enquanto a implementação da operação é chamada método. Normalmente um método é invocado pela passagem de uma mensagem ao objeto para que seja executado o método correspondente.

Para aplicações de banco de dados, o requisito de que todos os objetos sejam completamente encapsulados é muito rígido. Uma maneira de flexibilizar esse

requisito é dividir a estrutura de um objeto em atributos (variáveis de instância) visíveis e ocultos.

2.4 Especificando persistência de objeto através de Nomeação e Alcançabilidade

"Um SGBDOO normalmente possui um acoplamento forte com uma Linguagem de Programação Orientada a Objetos, que é utilizada para especificar a implementação dos métodos, assim como outros códigos da aplicação."(1)

Normalmente um objeto é criado por algum programa de aplicação em execução, pela chamada da operação do construtor do objeto. Nem todos os objetos criados são armazenados permanentemente no banco de dados. Os objetos transientes existem na execução do programa e desaparecem quando o programa termina. Os objetos persistentes são armazenados no banco de dados e persistem após o término do programa. Os mecanismos usuais para tornar um objeto persistente são a nomeação e a alcançabilidade.

O mecanismo de nomeação consiste em dar a um objeto um nome persistente único pelo qual ele possa ser recuperado pelo programa em execução e outros programas.

Alcançabilidade é um mecanismo que torna o objeto alcançável a partir de algum objeto persistente. Um objeto B é dito alcançável a partir de um objeto A se uma sequência de referências no grafo conduz, a partir do objeto A, ao objeto B.

2.5 Hierarquias de Classe e Tipo, e Herança

Outra característica essencial em sistemas de banco de dados Orientados a Objetos é que eles permitem hierarquias de tipo e herança. As hierarquias de tipo em bancos de dados normalmente acarretam uma restrição nas extensões correspondentes aos tipos na hierarquia.

2.6 Hierarquias de Tipo e Herança

Na maioria das aplicações de bancos de dados há uma grande quantidade de objetos do mesmo tipo ou classe. Assim os bancos de dados

Orientados a Objetos devem ser capazes de classificar objetos com base em seus tipos, como fazem outros sistemas de bancos de dados. Mas em bancos de dados Orientados a Objetos um requisito adicional é que o sistema permita a definição de novos tipos baseados em outros tipos predefinidos, formando uma hierarquia de tipos (ou de classes).

Normalmente um tipo é definido pela atribuição de um nome e então se definem seus atributos (variáveis de instância) e operações (métodos) para esse tipo.

2.7 Restrições em extensões correspondentes a uma hierarquia de tipos

Na maioria dos bancos de dados Orientado a Objetos, a coleção de objetos em uma extensão possui o mesmo tipo ou classe.

É comum em aplicações de bancos de dados que cada tipo ou subtipo possua uma extensão associada, que mantenha a coleção de todos os objetos persistentes daquele tipo ou subtipo.

2.8 Objetos Complexos

As principais motivações que levaram ao desenvolvimento de sistemas Orientados a Objetos foi o desejo de representar objetos complexos. Existem dois tipos principais de objetos complexos: estruturados e não estruturados. Um objeto complexo estruturado é constituído de componentes e é definido pela aplicação de construtores de tipos disponíveis, de modo recursivo, em vários níveis. Um objeto complexo não estruturado, em geral, é um tipo de dado que requer um grande volume de armazenamento, tal como um tipo de dado que representa uma imagem ou um objeto de texto longo.

2.9 Objetos complexos não estruturados e extensibilidade de tipo

Uma vantagem de lidar com objetos complexos não estruturados oferecidos por um SGBD é que eles permitem o armazenamento e a recuperação

de grandes objetos que são necessários à aplicação do banco de dados. Exemplos típicos desses objetos são imagem bitmap e string (cadeias de caracteres) de texto longo (como documentos); também são conhecidos como binary large objects (objetos binários extensos) ou BLOBs. Cadeias de caracteres também são conhecidas como objetos de textos extensos (character largte objects), ou CLOBs. Esses objetos não são estruturados, o que significa que o SBGD não conhece suas estruturas, somente a aplicação que os utiliza pode interpretar seu significado. Por exemplo, a aplicação pode possuir funções para exibir uma imagem ou procurar algumas palavras-chave numa string de texto longo. Os objetos são considerados complexos porque exigem uma grande área de armazenamento e não são parte dos tipos de dados padrões fornecidos pelos SGBDs convencionais. Uma vez que o objeto é muito grande, o SGBD pode recuperar uma parte do objeto e fornecê-lo ao programa de aplicação antes que o objeto seja recuperado por inteiro. O SGBD também pode utilizar técnicas de buffering e caching para buscar previamente partes do objeto antes que o programa de aplicação precise acessá-los.

2.10 Objetos complexos estruturados

Um objeto complexo estruturado diferencia-se de um objeto complexo não estruturado no sentido de que a estrutura do objeto é definida pela aplicação recursiva dos construtores de tipo oferecidos pelo SGBDOO. Assim, a estrutura do objeto é definida e conhecida pelo SGBDOO.

2.11 Polimorfismo (Sobrecarga de Operador)

Os sistemas Orientados a Objetos possuem características que dão suporte ao polimorfismo de operações, que também é conhecido como sobrecarga de operador. Este conceito permite que o mesmo nome de operador ou símbolo seja associado a duas ou mais implementações diferentes para o operador, dependendo do tipo de objeto ao qual o operador é aplicado.

2.12 Herança Múltipla e Herança Seletiva

Em uma hierarquia de tipos, a herança múltipla ocorre quando um subtipo T é subtipo de dois (ou mais) tipos e, assim herda as funções (atributos e métodos) de ambos os supertipos. Há várias técnicas para lidar com ambiguidade em herança múltipla. Uma solução é executar uma verificação do sistema para ver se há ambiguidade quando o subtipo for criado, deixando o usuário escolher, explicitamente, qual função deve ser herdada naquele momento. Outra solução é utilizar algum default do sistema.

Uma terceira solução é desabilitar herança múltipla se ocorrer ambiguidade de nome, forçando o usuário a mudar o nome de uma das funções em um dos supertipos. Alguns sistemas Orientado a Objetos não permitem herança múltipla de modo nenhum.

A herança seletiva ocorre quando um subtipo herda somente algumas das funções de um supertipo, não herdando outras. Nesse caso, uma cláusula EXCEPT pode ser utilizada para listar as funções de um supertipo que não devem ser herdadas pelo subtipo. Normalmente, o mecanismo de herança seletiva não é disponibilizado em sistemas de bancos de dados Orientado a Objetos, mas é utilizado mais frequentemente em aplicações de inteligência artificial.

3 BANCOS DE DADOS OBJETO RELACIONAL

O modelo de banco de dados Objeto Relacional surgiu como uma forma de estender as funcionalidades dos SGBDs relacionais com algumas das características presentes nos SGBDs de objetos (SGBDOs). O principal impulso para o desenvolvimento de SGBDORs estendidos surgiu da não aderência dos SGBDs legados e do modelo relacional básico, assim como dos primeiros SBGDRs para atender às demandas das novas aplicações que estão basicamente em áreas que envolvem uma variedade de tipos de dados, por exemplo: texto em publicação eletrônica, imagens em fotos de satélite ou em previsão do tempo, dados complexos não convencionais em projetos de engenharia, em informações do genoma biológico ou em projetos arquiteturais, em série de dados históricos de transações de

mercado de ações ou histórico de vendas e dados espaciais e geográficos em mapas, dados sobre poluição da água ou do ar, ou informações sobre tráfego.

Assim, há uma clara necessidade de se projetar bancos de dados que possam desenvolver, manipular e manter objetos complexos que surgiram com as novas aplicações. Além disso, tornou-se necessário manipular informações digitalizadas que armazenem sequências de dados de áudio e vídeo (particionadas em quadros individuais) exigindo dos SGBDs o armazenamento de BLOBs(Binary Long Objects – Objetos Binários Longos). A popularidade do modelo relacional foi favorecida por uma infraestrutura muito robusta em termos de SGBDs comerciais, que foram projetados para suportá-lo. Porém, o modelo relacional básico e as primeiras versões de linguagem SQL mostraram-se inadequadas para atender aos novos desafios. Os modelos de dados legados, como o modelo de rede, possuem a facilidade de modelar relacionamento explicitamente, mas falham pelo uso intenso de ponteiros na implementação e não possuem conceitos como identidade de objetos, herança, encapsulamento e suporte a diversos tipos de dados e objetos complexos.

O modelo hierárquico é bem adequado a algumas ocorrências típicas de hierarquias no mundo real e nas organizações, mas é muito limitado e rígido em termos de caminhos hierárquicos internos aos dados. Assim, iniciou-se uma tendência de se combinar as melhores características do modelo de dados e das linguagens de objetos com o modelo de dados relacional, de modo que ele possa ser estendido para lidar com os desafios das aplicações de hoje.

Para demonstrar como os SGBDORs surgiram, começamos focalizando o modo de classificar as aplicações de SGBDs de acordo com duas dimensões ou eixos: (1) complexidade dos dados – a dimensão X, e (2) a complexidade das consultas – a dimensão Y. Podemos dispor esses eixos em um espaço elementar 0-1 formado por quatro quadrantes:

Quadrante 1 (X=0, Y=0): Dados simples, consultas simples

Quadrante 2 (X=0, Y=1): Dados simples, consultas complexas

Quadrante 3 (X=1, Y=0): Dados complexos, consultas simples

Quadrante 4 (X=1, Y=1): Dados complexos, consultas complexas

Os SGBDRs tradicionais pertencem ao Quadrante 2. Apesar de suportarem consultas e atualização ad-hoc complexas (assim como processamento de transações), eles lidam apenas com dados simples que podem ser modelados como um conjunto de linhas em uma tabela. A maioria dos bancos de dados de objetos (SGBDOs) cabe no Quadrante 3, uma vez que este concentra-se no gerenciamento de dados complexos, mas possuem recursos limitados para consultas, baseadas em navegação. Com o objetivo de ocupar o quarto quadrante para dar suporte tanto a dados como as consultas complexas, os SGBDRs passaram a agregar mais objetos de dados complexos, enquanto os SGBDOs agregam mais suporte para consultas complexas (por exemplo, a linguagem de consultas de alto nível OQL. O quadrante 4 por possui extensões em seu modelo relacional básico, incorpora uma gama de características que o tornam objeto-relacional.

3.1 SGBDROS que evoluíram a partir dos SGBDRS

SGBDROSs atuais que evoluíram a partir de SGBDRs incluem o Oracle da Oracle Corporation, Universal DB (UDB) da IBM, o Odapter da Hewlett Packard(HP) (que estende o SBGD da Oracle) e o Open ODB da HP (que estende o Allbase/SQL, produto próprio da HP). Os produtos com maior sucesso parecem ser aqueles que mantêm a opção de trabalhar como um SGBDR enquanto acrescentam funcionalidades adicionais.

3.1.2 Características objeto-relacionais do Oracle 8

Como um SGBDOR, o Oracle 8 continua fornecendo a capacidade de um SGBDR e adicionalmente oferece suporte a conceitos orientados a objetos. O gerenciador fornece altos níveis de abstração de modo que os desenvolvedores de aplicações podem manipular objetos de aplicação em vez de construir os objetos a partir de dados relacionados. A informação complexa sobre um objeto pode estar oculta, mas as propriedades (atributos, relacionamentos) e os métodos (operações) do objeto podem ser identificados no modelo de dados. Além disso, as declarações

de tipos de objetos podem ser reutilizadas através de herança, reduzindo desse modo, o esforço e o tempo de desenvolvimento da aplicação. Para facilitar a modelagem de objetos, o Oracle apresenta as características descritas a seguir:

- Representação de atributos multivalorados utilizando VARRAY. Alguns atributos de um objeto/entidade podem ser multivalorados;

- Utilizar tabelas aninhadas para representar objetos complexos. Na modelagem de objetos, alguns atributos de um objeto poderiam ser também objetos.

- Visões de objetos. As visões de objetos podem ser utilizadas para construir objetos virtuais a partir de dados relacionais, possibilitando assim que programadores utilizem esquemas existentes para suportar objetos.

3.1.3 Gerenciamento de objetos longos e características de armazenamento adicionais

O Oracle atualmente permite armazenar grandes objetos como vídeo, áudio e textos de documentos. Foram introduzidos novos tipos de dados para essa finalidade. Dentre esses tipos, incluem-se:

- BLOB (binary large object – Objeto binário longo).
- CLOB (character large object – Objeto de texto longo).
- BFILE (arquivo binário armazenado à parte do banco de dados).
- NCLOB (CLOB de tamanho fixo)

Com exceção de BFILE, que é armazenado externamente ao banco de dados, os demais tipos são armazenados internamente junto com outros dados. Para um BFILE, somente o nome do diretório é gravado no banco de dados.

3.2 Implementação e aspectos relacionados a sistemas de tipos estendidos

Os SGBDORs devem associar dinamicamente uma função definida pelo usuário em seu espaço de endereçamento apenas quando esta é chamada. Como observamos nos casos dos dois SGBDORs, várias funções são requeridas para operar sobre dados espaciais em duas ou três dimensões, imagens, textos, e assim por diante. Os aspectos cliente-servidor lidam com a substituição e ativação de

funções. Se o servidor precisar executar uma função, é preferível fazê-lo no espaço de endereçamento do SGBD em vez de remotamente, em razão do alto volume de sobrecarga. Deve ser possível efetuar consultas dentro de funções. Uma função deve operar do mesmo modo quando é utilizada a partir de uma aplicação por meio de uma API (Aplication Program Interface) ou quando é chamada pelo SGBD como parte de uma execução SQL com a função embutida em um comando SQL. Os sistemas devem suportar um aninhamento desses callbacks.

3.3 O modelo relacional aninhado

O modelo relacional aninhado elimina a restrição da primeira forma normal (1NF) do modelo relacional básico, sendo também conhecido como a Não-1NF ou NFNF(Non-First Normal Form) ou modelo relacional NF2. No modelo relacional básico, também denominado modelo relacional plano (flat relational model), é exigido que os atributos sejam monovalorados e que possuam domínios atômicos. O modelo relacional aninhado permite atributos compostos e multivalorados, gerando assim tuplas complexas com estruturas hierárquicas.

4 DIFERENÇAS DE BANCO DE DADOS ORIENTADOS A OBJETOS EM RELAÇÃO AOS TRADICIONAIS RELACIONAIS E RELACIONAIS ESTENDIDOS.

Deve ser observada a diferença entre os modelos de banco de dados convencionais e bancos de dados Orientados a Objetos. Em modelos de banco de dados convencionais, tais como o modelo relacional ou o modelo EER (Entidade Relacionamento Estendido), assume-se que todos os objetos são persistentes.

Assim, quando um tipo de entidade ou classe como EMPREGADO é definido no modelo EER, ele representa tanto a declaração de tipo para EMPREGADO quanto um conjunto persistente de todos os objetos EMPREGADO. Na abordagem Orientada a Objetos, uma declaração de classe para EMPREGADO especifica somente o tipo e as operações para uma classe de objetos. O usuário deve definir separadamente um objeto persistente do tipo set (EMPREGADO) ou list (EMPREGADO), cujo valor é a coleção de referências para todos os objetos

persistentes EMPREGADO, se for desejado. Isso permite que objetos transientes e persistentes sigam o mesmo tipo e declarações de classe da ODL e da LPOO, respectivamente. Em geral, é possível definir várias coleções persistentes para a mesma definição de classe, se desejável.

5 CONSIDERAÇÕES FINAIS

Os bancos objetos relacionais, a exemplo o Oracle 10 g, estão estruturados para sistemas relacionais, bem como, orientado a objetos. Na comparação dos bancos, o objeto relacional por oferecer as duas opções, permitindo a flexibilidade de acordo com a necessidade do cliente. Portanto é a melhor opção de banco de dados.

6 ABSTRACT

The comparison of data models object oriented with respect to relational objects is presented in this article. This analysis permits to show the advantages and disadvantages of using each of the models. They will be presented various aspects of the database object-oriented, its origin, object identity, structures and builders types, encapsulation operations, methods and persistence, object behavior, object persistence in the database, class hierarchies, type, multiple inheritance and selective inheritance, among others. As the object-relational database, origin, evolution from Systems Managers Database, characteristics, management of long objects and additional storage, and implementation aspects related to extended type systems, nested relational model. Given presentation of the differences between the databases, choosing the option that best meets your specific needs is easy.

Keywords: Database post-relational, database model object-oriented, model object-relational database, analysis of data models and object-oriented relational objects.

7 REFERÊNCIAS

ELMASRIR; NAVATHE, S.B. Sistema de banco de dados, São Paulo: Pearson Addison Wesley, 2005, 724 pg.

DATE, C.J. Introdução a sistemas de banco de dados, Edit Elsevier, 2003, 865 pg.

BERZ, E.L.; WAGNER, R.; FIGUEIRA, T.G. Sistemas Gerenciadores de Banco de Dados Orientado a Objeto e Objeto-Relacional. Faculdade de Informática de Taquara (FIT/FACCAT).

Eugênio A. Nassu , Valdemar W. Setzer ,Bancos de Dados Orientados a Objetos,1999.

Khoshafian, S. Banco de Dados Orientado a Objeto. Livraria e Editora Infobook S.A., 1994.