SISTEMAS GERENCIADORES DE BANCOS DE DADOS ORIENTADOS A OBJETOS PARA AMBIENTES DE DESENVOLVIMENTO DE SOFTWARE

OBJECT ORIENTED DATA BASE MANEGEMENT SYSTEMS FOR THE SUPPORT OF SOFTWARE DEVELOPMENT ENVIRONMENTS

Carlos Miguel Tobar TOLEDO*

ABSTRACT

Data Base Management Systems (DBMSs) for the support of Software Development Environments (SDEs) constitute a new generation of DBMSs; some of those DBMSs, called Object Oriented Data Base Management Systems (OODBMSs), present characteristics of the object oriented approach and are considered to be an answer for the support of SDEs. This paper presents a review of concepts, characteristics, and established requirements for an SDE in order to characterisize the functionalities of an OODBMS for the support of SDEs.

KEY-WORDS: Data Base Management Systems, DBMS, Object Oriented Data Base Management Systems, OODBMS, Software Development Environments, SDE, Software Engineering, SE

RESUMO

Sistemas Gerenciadores de Bancos de Dados (SGBDs) para ambientes de desenvolvimento de software (ADSs) compõem uma nova geração de SGBDs, alguns dos quais apresentam características do paradigma da orientação a objetos, os Sistemas Gerenciadores de Bancos de Dados Orientados a Objetos (SGBDOOs), que surgem como uma resposta ao suporte de ADSs. Este trabalho apresenta uma revisão de conceitos, características e requisitos estabelecidos para ADSs, com o objetivo de caracterizar as funcionalidades de um SGBDOO suporte para ADSs.

PALAVRAS-CHAVE: Sistemas Gerenciadores de Bancos e Dados, SGBDs, Sistemas Gerenciadores de Bancos de Dados Orientados a Objetos, SGBDOOs, Ambientes de Desenvolvimento de Software, ADSs, Engenharia de Software, ES

1. INTRODUÇÃO

Sistemas Gerenciadores de Bancos de Dados (SGBDs) para ambientes de desenvolvimento de software (ADSs) compõem uma nova geração de SGBDs, destinada a suportar o que se convencionou chamar novas aplicações. Como exemplo das novas aplicações, além daquelas abrangidas pela engenharia de software, podem ser citadas:

- CAD/CAM, projeto auxiliado por computador,
- AI, bases de conhecimento/sistemas especialistas,
- GIS, sistemas de informação cartográficos/geográficos,
- multimidia e hipertexto,
- redes de telefonia inteligentes e
- sistemas de entretenimento.

Para atender às novas aplicações, o papel do SGBD deixa de ser de um repositório central contendo dados estáticos, para transformar-se em um sistema distribuído e inteligente para armazenamento de objetos ativos.

O suporte aos ADSs continua sendo um desafio para os pesquisadores da área, apesar de já existirem dezenas de SGBDs comercializados e centenas de pro-

(*) Mestre em Ciência da Computação pelo IMECC-UNICAMP, atualmente trabalha como pesquisador junto à Fundação Centro Tecnológico para Informática e como professor junto ao Instituto de Informática da PUCCAMP.

jetos e protótipos ditos orientados às novas aplicações.

Considera-se que um SGBD para engenharia de software, além de ter que suportar as atividades de desenvolvimento de software e respectiva gerência, deve considerar uma miríade de ferramentas e necessidades advindas dos diversos domínios da aplicação que possam existir [Zicari92]. Precisa, também, ser adptável, flexível e dinâmico na mesma proporção do empreendimento, projetos e pessoas que deve suportar [Chen92].

O restante deste trabalho está dividido como se segue. A seção 2 apresenta as necessidades e restrições que um ADS pode impor a um SGBD. A seção 3 define o que vem a ser um SGBD Orientado a Objetos (SGBDOO). Na seção 4 são tecidas considerações com relação às características desejáveis em um SGBDOO para suporte à engenharia de software e, finalmente, são tecidas conclusões.

2. ADS COM SGBD

O termo ADS se refere a um conjunto de ferramentas de hardware e software necessário ao desenvolvedor de sistemas de software (aplicações).

A complexidade advinda da aplicação repercute na quantidade de tempo e de recursos (hardware, software e pessoal) necessária ao seu desenvolvimento, além, é claro, nos requisitos de gerência impostos por eles.

A integração dos recursos de hardware e de software a ferramentas que auxiliem o desenvolvimento e a sua gerência é fator fundamental para a obtenção de qualidade nos resultados e diminuição dos custos a eles associados.

Existem cinco tipos de integração de ferramentas: dados (das ferramentas e seus inter-relacionamentos), controle (notificação de eventos entre ferramentas), apresentação (interface como o usuário), processos (controle das atividades de desenvolvimento e seus encadeamentos) e plataforma (inter-operabilidade das ferramentas) [Clow89] [Wasserman89].

A combinação dos tipos de integração e mesmo a sua existência varia de ADS para ADS. Para efeitos deste trabalho, considera-se a existência da integração de dados nos ADSs.

O papel de um SGBD em um ADS é o de integrador de dados. Como tal, o SGBD deve prover serviços básicos que incluem: armazenamento e gerência de objetos, entidades, relações e ligações; controle de versões e configuração; serviço e identificação ("naming"); segurança e controle de transações [Chen92].

Podemos classificar ASDs de diversas maneiras: de acordo com a cobertura de um ciclo de vida, de acordo com os tipos de ferramentas por eles providos, de acordo com a dimensão da instalação e complexidade da aplicação, etc.

A abordagem adotada para análise dos ADSs foi a da escolha de uma perspectiva centrada na existência do SGBD e, a partir daí, explorar os aspectos decorrentes de seu estudo.

Na análise promovida em [Lockeman90], a respeito do futuro da tecnologia de SGBDs, são estabelecidas três forças que direcionam os esforços atuais de pesquisa na área:

- aplicações da tecnologia de SGBDs,
- tecnologia básica de hardware e software existente, sobre a qual a tecnologia de SGBDs se apoia ou pode vir a se apoiar, e
- padrões que, por um lado, estabelecem restrições sobre os desenvolvimentos, mas, por outro, promovem a integração de produtos e tecnologias heterogêneos.

Tendo estabelecido que a aplicação da tecnologia de SGBDs é a engenharia de software, obtêm-se três dimensões para a análise dos requisitos que um ADS impõe ao seu componente integrador de dados:

a) das aplicações a serem desenvolvidas no ADS, resultante das necessidades da organização, denominada visão da organização;

b) das políticas para desenvolvimento, advindas do exercício da engenharia de software, denominada visão da engenharia de software; e

c) das tecnologias e padrões existentes no ADS e requeridos nas aplicações a serem desenvolvidas, denominada visão tecnológica.

Nota-se que as duas primeiras dimensões resultam da primeira força direcionadora de [Lockeman90]. Esta mudança de perspectiva se origina na existência de duas dimensões quando se estabelece a engenharia de software como aplicação da tecnologia de SGBDs (a própria engenharia de software e a aplicação desta).

A terceira dimensão é uma junção das duas últimas forças direcionadoras, por considerarmos que os padrões se aplicam aos componentes da tecnologia, podendo dessa maneira, serem analisados em conjunto.

2.1. A Visão da Organização

Esta dimensão representa o porquê do uso do ADS. Nela podem-se distinguir três aspectos relativos à organização onde o ADS se encontra instalado: o perfil da instalação responsável pelos desenvolvimentos, o domínio das aplicações e os tipos de soluções desenvolvidas.

Termos como "programming in small" e "programming in the large" foram cunhados para sintetizar aspectos desta dimensão.

O perfil da instalação representa todas as componentes encontradas no ambiente de desenvolvimento, principalmente com relação aos recursos humanos: tamanho da equipe, funções, relacionamentos, etc.

O domínio das aplicações representa a área de atuação da organização e, portanto, o tipo de problemas que requerem solução.

Os tipos de solução representam os modelos tecnológicos selecionados como base para a solução dos problemas da organização: tempo real, tempo compartilhado ou "batch"; centralizado ou distribuído; multi usuário ou mono usuário; etc. (este aspecto é discutido mais adiante na dimensão tecnológica, porém, nesta dimensão se aplica aos produtos do ADS e lá ao próprio ADS).

Uma classificação de ADSs considerando esta dimensão, seria a encontrada em [Perry88]:

- modelo individual - também conhecido como ambiente de programação, representa os ambientes que provêm o conjunto mínimo de ferramentas de implementação necessárias para a construção de software;
- modelo familiar - representa os ambientes que, além de oferecerem o que o modelo individual supre, oferece facilidades para o suporte da interação de um pequeno grupo de programadores. Basicamente, estende o modelo anterior com uma combinação de gerência de configuração, integração, distribuição (lan) ou gerência de projeto;
- modelo metropolitano - representa os ambientes que oferecem o mesmo que o modelo familiar e se destinam a um grupo de mais de 20 pessoas, orientado por uma metodologia (métodos e técnicas de desenvolvimento e gerência);
- modelo estadual - que apresenta um modelo genérico para desenvolvimento de software, o qual é instanciado para cada projeto e que permite a gerência das diferenças entre as várias instâncias.

2.2. A visão da engenharia de software

Esta dimensão representa o que o ADS oferece, em termos de políticas e estratégias, para facilitar o desenvolvimento. Nela podem-se distinguir três aspectos existentes no desenvolvimento e que, de alguma maneira, receberam atenção da engenharia de software. Esta perspectiva é encontrada em [Pocock91]. São eles:

- o processo, onde se concentram as atividades de planejamento, definição e gerência do desenvolvimento e dos recursos necessários; as ferramentas de suporte a estas atividades são as de gerência de projeto;
- o produto, onde se concentram as atividades de geração e validação de produtos e sub produtos; as ferramentas de suporte são as denominadas CASE ou, mais precisamente, "lower case" (codificação e teste) ou "upper case" (especificação e projeto);
- a construção, onde se concentram as atividades de coordenação dos produtos e sub produtos gerados em paralelo por atividades inter dependentes; as ferramentas são as denominadas gerência de configuração ou controle de mudanças e integração.

Uma classificação de ADSs considerando esta dimensão seria a encontrada em [Wasserman89]:

- ferramentas horizontais, usadas durante todo o processo de desenvolvimento de software,
- ferramentas verticais, usadas para a especificação de uma determinada fase do processo.

[Gibson91] estende a classificação das ferramentas verticais em:

- "upper", utilizadas para a definição da perspectiva estratégica da organização (missão, objetivos, planos estratégicos, planos operacionais, recursos, etc.);
- "middle", utilizadas para o planejamento (análise e projeto; definição do nível lógico do modelo estratégico);
- "lower", utilizado para a construção (nível físico do modelo estratégico).

2.3. A visão tecnológica

Esta dimensão representa o como a ADS suporta o ferramental oferecido aos seus usuários. Os aspectos relevantes nesta dimensão dizem respeito ao conjunto de soluções utilizadas para a implementação e funcionamento do próprio ADS, são elas:

- a arquitetura, diz respeito à tecnologia usada para tratar os problemas de capacidade de processamento, capacidade de armazenamento e tempo de resposta; basicamente, indica se a solução é centralizada ou distribuída, podendo apresentar uma combinação das duas;
- a plataforma, diz respeito à tecnologia existente nos produtos escolhidos para comporem o hardware e o software básico; aquí se encontra uma gama enorme de fornecedores e opções; por exemplo, para o hardware, "main frames", estações de trabalho, etc., para o software, basicamente, o sistema operacional (pode-se pensar nos componentes de comunicação, armazenamento primário e secundário, interface, etc.);
- o modelo computacional, diz respeito às tecnologias escolhidas para comporem as facilidades oferecidas, i.e., a variedade de suporte às possíveis soluções ou necessidades dos usuários; basicamente incorporam as características funcionais do ambiente (concorrência, multi-usuário, multi-linguagens, paradigmas de programação, etc.);
- filosofia de integração, diz respeito à preocupação para que os componentes do ambiente (presentes e futuros) sejam portáveis, possam interoperar entre si e interajam com o usuário de maneira similar, basicamente indica o grau de "abertura" do ambiente.

Estes aspectos em conjunto indicam a existência ou não de: facilidades para a inclusão de novas tecnologias, transparência de localização, trabalho em grupo, portabilidade, evolução, etc.

Uma classificação de ADSs considerando esta dimensão seria a encontrada em [Dart87] e estendida por [Strelich89]:

- centrados em linguagens (Interlisp, Smalltalk ou Cedar);
- orientados por estrutura (Gandalf ou Mentor);
- "tool-kits" (sistemas Unix);
- baseados em métodos (Excelerator ou SREM); e
- "environment framework" (Arcádia ou ISM).

3. SGBDOO

O paradigma de programação orientada a objetos, incorporado ao mundo dos SGBDs, resultou no aparecimento de bancos de dados não convencionais, denominados SGBDs Orientados a Objetos (SGBDOOs).

O termo "orientado a objetos" é melhor interpretado quando se refere a qualquer abordagem que explore encapsulamento, no processo de projeto e desenvolvimento de software [Nierstrasz88].

O termo SGBDOO é utilizado da mesma maneira que [Cattell91], para quem as abordagens mais populares para SGBDs de próxima geração são chamadas de abordagens para SGBDOOs ou SGBDs relacionais estendidos.

Existem, basicamente, duas escolas de pesquisa trabalhando com SGBDOO: a primeira advoga a abordagem de que um SGBDOO é o resultado da tecnologia dos SGBDs relacionais e estendidos semanticamente com os aspectos da orientação a objetos [CADBMSF90]; a segunda advoga os princípios existentes nas linguagens orientadas a objetos como base para o desenvolvimento de uma nova tecnologia de SGBDs [Atkinson89].

Exemplos de SGBDOOs da primeira escola são Postgres [Stonebraker91] e Starburst [Lohman91], enquanto que, da segunda escola são Object Store [Lamb91], Gemstone [Butterworth91] e O [Deux91].

As características essenciais para um SGBDOO defendidas por cada escola são apresentadas nas tabelas 1 e 2. Na tabela 3 as características de cada escola são apresentadas lado a lado. Note-se que, além da coincidência no número (treze), algumas características são iguais ou podem ser vistas como uma composição das defendidas pela outra escola. É o caso da característica número 1 em [CADBMSF90] que é uma combinação das características 1, 4 e 8 em [Atkinson89].

Tabela 1 - Características de um SGBDOO em [Atkinson89]

1. Complex Objects
   A capacidade de permitir definição de objetos (mais complexos) a partir de objetos já definidos (mais simples), através de operadores estruturais utilizados arbitrariamente. Além disso, a capacidade de gerenciar as hierarquias de composição resultantes.
2. Object Identity
   O mecanismo de se poderem distinguir dois objetos, independentemente dos valores de seus atributos.
3. Encapsulation
   A separação clara entre a semântica visível e a implementação dos objetos.
4. Types and classes
   A capacidade de organização de objetos similares e suas implementações em uma abstração.
5. Inheritance
   A capacidade de criação de novas classes a partir das existentes.
6. Polymorphism and Dynamic Binding
   A capacidade de associação de mensagens similares a diferentes métodos, dependendo do tipo do objeto passado como parâmetro; os métodos podem ser associados a tempo de execução (dynamic) ou de compilação.
7. Computation completness
   A capacidade de permitir ao usuário expressar qualquer tipo de função computável.
8. Extensibility
   A capacidade de permitir a extensão do conjunto de tipos de um SGBD; não permitindo a ocorrência de distinção de uso entre os tipos do sistema e os definidos pelo usuário.
9. Persistence
   A capacidade de permitir a existência de objetos além do tempo de vida dos processos que os criaram.
10. 2ndary Storage Management
    Mecanismos de acesso às informações que sejam eficientes; por exemplo "clustering", índices, "buffers" e otimizações de busca.
11. Concurrency
    A capacidade de permitir o acesso concorrente às informações através de atomicidade, compartilhamento controlado por "locks" e "serializability".
12. Recovery
    A capacidade de permitir a recuperação das informações do SGBD a um estado anterior íntegro, em caso de falha de software ou hardware.
13. Ad hoc query facility
    Mecanismos que permitam o acesso eficiente aos objetos, através de declarações de alto nível, além de navegação programável.

Tabela 2 - Características de um SGBDOO em [CADBMSF90]

1. Rich Type System
   Existência de um sistema de tipos semanticamente rico e extensível.
2. Inheritance
   Capacidade de herança entre tipos.
3. Encapsulation
   Capacidade de isolar, através de encapsulamento, a chamada das funções (incluindo os procedimentos do SGBD) de suas implementações e das estruturas de dados.
4. Unique Identifiers
   Capacidade de associação de identificadores únicos aos registros em caso de não existir uma chave primária.
5. Rules (triggers, constraints)
   Capacidade de associação de regras (restrições ou gatilhos) a uma função ou conjunto de informações específico (coleção).
6. Non-procedural HL Access Lang.
   Mecanismo programável (único) de acesso às informações do SGBD, cuja "interface" é uma linguagem de acesso de alto nível não "procedural".
7. 2Ways to specify collections
   Existência de ao menos duas formas para a especificação de coleções: uma usando a enumeração dos membros e a outra usando a linguagem de recuperação para a especificação de pertinência.
8. Updatable Views
   Mecanismos atualizáveis que permitam diferentes visões das informações no SGBD.
9. No performance indicators
   Indicadores de "performance" pouco têm a ver com modelos de dados e não devem aparecer neles.
10. Multiple HLLs
    Existência de múltiplas linguagens de alto nível para o acesso às informações do SGBD.
11. Persistence
    Capacidade de permitir que um entre vários objetos permaneça armazenado após o término do processo que o gerou. Essa capacidade deve ser suportada através de extensões em compiladores e sistemas de execução (mais ou menos) complexos.
12. SQL
    Suporte à linguagem SQL, visto que a mesma se tornou um padrão.
13. Queries for communication
    Mecanismo onde consultas e suas respostas resultem no nível mais baixo de comunicação entre clientes e servidor.

Tabela 3 - Comparação de Características para SGBDOOs

[Atkinson89]	[CADBMSF90]
01. Complex Objects	01. Rich Type System
02. Object Identity	02. Inheritance
03. Encapsulation	03. Encapsulation
04. Types and classes	04. Unique Identifiers
05. Inheritance	05. Rules (triggers, constraints)
06. Dynamic Binding and Polymorphism	06. Non-procedural HL Access Lang.
07. Computation completness	07. 2 ways to specify collections
08. Extensibility	08. Updatable Views
09. Persistence	09. No performance indicators
10. 2ndary Storage Management	10. Multiple HLLs
11. Concurrency	11. Persistence
12. Recovery	12. SQL
13. Ad hoc query facility	13. Queries for communication

Pode-se verificar que existe concordância entre as escolas quanto a determinado subconjunto de características que um SGBDOO deve oferecer:

a. sistema de tipos e classes extensível que permita a definição de objetos complexos;

b. identificação de objetos;

c. encapsulamento e a existência de polimorfismo;

d. hereditariedade e a existência de "overriding"; e

e. persistência.

Dessas apenas a identificação de objetos e a persistência não são encontradas, normalmente, nas linguagens orientadas a objetos.

Estas cinco características correspondem a oito das características essenciais em [Atkinson89] e a cinco em [CADBMSF90].

De qualquer forma, pode-se definir um SGBDOO como o resultado da conjunção de idéias provenientes das linguagens de programação orientadas a objetos e dos SGBDs convencionais. Em [Schlageter89] encontramos a seguinte definição para SGBDOO: um SGBDOO deve ser um SGBD, deve ser um sistema orientado a objetos e, além disso, suportar as necessidades advindas das novas aplicações.

4. SGBDOO PARA ENGENHARIA DE SOFTWARE

No restante deste trabalho assume-se a existência de um SGBDOO como integrador de dados nos ADSs, isto porque os SGBDOOs são considerados ideais para o suporte de ADSs [Zicari92] [Cattell91] [Blair90] [Atkinson89] [Peterson87], embora ainda não exista uma concordância quanto às características e funcionalidades necessárias para atendimento dessa nova aplicação.

[Zicari92] propõe dezessete características essenciais em um SGBD para suporte de ADSs. Estas são apresentadas na Tabela 4, relacionadas aos seguintes macro aspectos:

- suporte multi-usuário e perfis de usuários,
- tipo de informação e sua transformação/evolução,
- armazenamento distribuído, íntegro, eficiente e confiável,
- incorporação de tecnologia proveniente de outras áreas.

O aspecto suporte multi-usuário e perfis de usuários compreende características advindas de um ambiente compartilhado por diversos usuários com responsabilidades e interesses distintos. São elas: compartilhamento de informações, concorrência, trabalho cooperativo e gerência de transações não convencionais, além de visões e mecanismos de segurança.

O aspecto tipo de informação e sua transformação/evolução compreende características relacionadas com os diferentes objetos criados e manipulados e seus inter-relacionamentos. São elas: complexidade dos objetos, atualização da estrutura da informação, relacionamento (esquema) e constituição dos objetos, extensibilidade para novos tipos de objetos e manipulação de meta-informação.

O aspecto armazenamento distribuído, íntegro, eficiente e confiável compreende características advindas das facilidades encontradas em SGBDs convencionais, quais sejam: persistência, restrições de integridade, facilidades para consultas (queries) específicas, recuperação de falhas, distribuição em ambientes (hardware e software) não homogêneos e gerência eficiente do armazenamento secundário.

O aspecto incorporação de tecnologia proveniente de outras áreas compreende características existentes em outras áreas de pesquisa e que trazem benefícios. São elas: triggers (gatilhos, responsáveis por permitir a ativação de ações a partir de modificações no estado do SGBD), capacidade dedutiva (encontrada em sistemas especialistas), além de mecanismos gráficos avançados.

Além desses macro-aspectos, existe a necessidade de facilidades para a gerência de configuração, representada pelo tratamento de versões.

A figura 4 apresenta uma síntese da comparação da Tabela 1 [Atkinson89] e da Tabela 2 [CADBMSF90] com as características estabelecidas em [Zicari92].

Nota-se que a abordagem orientada a objetos baseada em linguagens orientadas a objetos, essencialmente, atende a apenas sete dos dezessete requisitos para um SGBD suporte de ADSs. Outros quatro requisitos, considerados opcionais em [Atkinson89], são encontrados em [Zicari92] como essenciais.

A abordagem orientada a objetos baseada na extensão do modelo relacional, essencialmente, atende a apenas cinco dos dezessete requisitos.

Apenas três requisitos recebem concordância de ambas as escolas.

Cada um dos aspectos encontrados nas três dimensões, introduzidas na seção 2, requer algum tipo de suporte do SGBDOO. Por exemplo, o processo na visão da engenharia de software necessita que seja possível: modelar o ciclo de vida do desenvolvimento, representar a estrutura de divisão de trabalho ("work breakdown structure"), monitorar eventos, registrar a história do projeto, representar o estado de cada recurso, de cada atividade e de cada produto ou sub produto, etc. Para isso [Liu88] considera necessário dispor de uma notação temporal, modificação dinâmica de meta-informação e hereditariedade múltipla.

Ao mesmo tempo, a figura 4 apresenta uma distribuição das características em [Zicari92], considerando as três dimensões. Essa distribuição considerou que determinada característica atendia mais fortemente um determinado aspecto, podendo, no entanto, atender outros aspectos das três dimensões.

Tabela 4 - Características essenciais a SGBDs para ADSs em [Zicari92]

	suporte multi-usuário	transf. evolução inform.	tecnol. SGBDs convenc.	novas tecnol.	
1. Complex information modeling		+			(1) (2)
2. Versioning					(3)
3. Integrity constraints and triggers			+	+	(2)
4. Advanced transaction management	+				(3)
5. Schema and object updates		+			
6. Views and authorization mechanisms	+				(2)+
7. Deductive capabilities				+	
8. Advanced graphical facilities				+	
9. Meta-data handling		+			(3)+
10. Data sharing	+				
11. Secondary storage management			+		(1)
12. Ad hoc query facilities			+		(1)
13. Distribution and cooperative work	+		+		(3)
14. Extensibility		+			(1) (2)
15. Persistence			+		(1) (2)
16. Concurrency	+				(1)
17. Recovery			+		(1)

(1) essencial em [Atkinson89]
(2) essencial em [CADBMSF90]
(3) opcional em [Atkinson89]
+ parcialmente

Nota-se uma preocupação maior em considerar os aspectos tecnológicos e da engenharia de software em detrimento dos organizacionais.

Além das 17 características definidas por [Zicari92], considera-se que as seguintes cinco primeiras características essenciais e seis últimas opcionais deveriam ser consideradas:

- facilidades de recuperação semântica, para suporte dos aspectos domínio da aplicação, o produto, o processo e tipos de solução;
- multi-linguagem (completude computacional), para suporte ao aspecto o produto;
- interface com outros SGBDs, para suporte ao aspecto filosofia de integração;
- disponibilidade, para suporte ao aspecto arquitetura;
- gerência de processamento, para suporte ao aspecto plataforma;
- "traceability', para suporte ao aspecto a construção;
- "branching squeme", para suporte ao aspecto a construção;
- multi-mídia, para suporte ao aspecto modelo computacional;
- adaptabilidade (preservação do hábito, níveis dos usuários), para suporte dos aspectos filosofia de integração e perfil da instalação;
- "customizability", para suporte ao aspecto perfil da instalação;
- facilidades de representação temporal (antes, depois, durante etc.), para suporte dos aspectos domínio da aplicação e o processo.

6. CONCLUSÕES

Neste trabalho, inicialmente, caracteriza-se SGBD suporte de ADS e SGBDOO. Em seguida são tecidas considerações com relação à caracterização de um SGBDOO suporte de ADS.

Existe uma clara necessidade de se estabelecerem características básicas que um SGBDOO deve ter para atender a engenharia de software.

Essa necessidade decorre de dois fatos: primeiro, não existe consenso dos requisitos mínimos para SGBDOOs que atendam outras aplicações, incluindo o suporte a ADSs; segundo, apesar de alguns esforços para padronização, os interesses que estão por trás da comunidade que pesquisa SGBDOOs permitirão que apenas alguns tópicos possam ser padronizados.

Exemplos de esforços para padronização: X3/SPARC/DBSSG/OOBTG, o Object Management Group [Joseph91], o ANSI Resource Dictionary System, o trabalho do IEEE em interconexão de ferramentas (P1175) e o CASE Data Interchange Format da Eletronic Industries Association.

Considerando a caracterização de [Zicari92] como a correta para SGBDOOs, suporte para engenharia de software, e de sua inadequada compatibilidade com as diferentes propostas para SGBDOOs genéricos (ver tabela 4), verificamos que a conceituação de SGBDOOs poderia ser revista.

A revisão poderia dar origem a sistemas especializados e específicos para atender cada nova aplicação, cujas estruturas multi-níveis apresentassem funcionalidades comuns, similares à estrutura do sistema operacional UNIX.

A estrutura multi-níveis associada ao conceito de encapsulamento e especialização, presentes no paradigma de objetos, permitiriam, por exemplo, que um servidor de arquivos servisse como base para um servidor de objetos (que oferecesse a possibilidade de modelar objetos complexos, extensibilidade e persistência, por exemplo). Sobre esse servidor de objetos seriam adicionadas outras características dando origem a um SGBDOO genérico.

Utilizando o SGBDOO genérico e, talvez, outros servidores (comunicação remota, por exemplo) seria possível desenvolver sistemas de gerência de dados específicos para cada nova aplicação. Esse sistema poderia ser visto como um utilitário-aplicação, como uma interface ou como uma versão especializada do SGBDOO.

Não podemos esquecer que não estão contempladas, em qualquer das caracterizações estudadas, as necessidades inerentes ao processo de desenvolvimento de software (ciclos de vida ou metodologias, por exemplo), sua padronização e estruturação nos diversos componentes de software.

Um claro apelo mercantilista é notado nos artigos que descrevem os SGBDOOs existentes, quer porque apresentam um produto comercializado, quer porque defendem um projeto cujas características são contestadas por outros autores (é o caso de Stonebraker e seu Postgres). Este fato pode ser constatado ao compararmos artigos de diferentes datas a respeito do mesmo SGBDOO: ou se dizia que o objetivo não era atender ferramentas CASE e hoje é vendido como um SGBD de última geração, que suporta também ferramentas CASE; ou, inicialmente, se apresentava, sem meias palavras, como extensão ao modelo relacional e hoje tenta estabelecer uma visão tendenciosa do que vem a ser orientação de objetos para um SGBD.

Concluímos que há muito a ser feito, antes de podermos considerar a existência de um modelo apropriado para a engenharia de software, análogo ao existente modelo relacional para as aplicações comerciais e de gerência da informação.

BIBLIOGRAFIA

[Atkinson89] ATKINSON, M. et alli The Object-Oriented Database System Manifesto Proc. DOOD 89, Dec/89

[Blair90] BLAIR, G. S. et alli A synthesis of object-oriented and functional ideas in the design of a distributed software engineering environment Software Engineering Journal, May/90, p. 193-204

[Butterworth91] BUTTERWORTH, P., Otis, A. & Stein, J. The Gemstone Object Database Management System CACM, 34(10), Oct/91

[CADBMSF90] The Committee for Advanced DBMS Function Third-Generation Database System Manifesto SIGMOD Record, 19(3), Sept/90

[Cattell91] CATTELL, R. G. G. What are next-generation Database Systems? CACM, 34(10), Oct/91

[Chen92] CHEN, M. & Norman, R. J. A Framework for Integrated CASE IEEE Software, 9(2), Mar/92

[Dart87] DART, S. A. et alli Software Development Environments IEEE Computer, Nov/87, p. 18-28

[Deux91] DEUX, O. et alli The 02 System CACM, 34(10), Oct/91

[Gibson91] GIBSON, M. & Snyder, C. Computer Aided Software Engineering: Facilitating the Path for True Software and Knowledge Engineering International Journal of Soft. Eng'g and Knowledge Eng'g, 1(1), 1991, p. 99-114

[Joseph91] JOSEPH, J. V. et alli Object-Oriented Databases: Design and Implementation Proc. of the IEEE, 79(1), Jan/91

[Lamb91] LAMB, C. et alli The Objectstore Database System CACM, 34(10), Oct/91

[Liu88] LIU, L. & Horowitz, E. Object Database Support for a Software Project Management Environment Sigsoft Soft. Eng'g Notes. 13(5), Nov/88

[Lockeman90] LOCKEMAN, P., Kemper, A. & Moerkotte, G. Future Database Technology: Driving Forces and Directions Data Base Systems of the 90's Proc. Int. Symposium. Nov/90 Lecture Notes in Computer Science

[Lohman91] LOHMAN, G. M. et alli Extensions to Starburst: Objects, Types, Functions and Rules CACM, 34(10), Oct/91

[Nierstrasz88] NIERSTRASZ, O. M. A Survey of Object-Oriented Concepts in Object-Oriented Concepts and Databases, ed. Kim and Lochovsky, Addison-Wesley, 1988

[Perry88] PERRY, D. E. & Kaiser, G. E. Models of Software Development Environments Proc, loth Int. Conf. on Soft. Eng'g 1988, p. 60-68

[Peterson87] PETERSON, R. Object Oriented Data Base Design AI Expert, 2(3), March/87

[Pocock91] POCOCK, J. N. VSF and its Relationship to Open Systems and Standard Repositories Soft. Devel. Env. and CASE Technology Proc. European Symposium, Jul/91, p. 55-64 Lect. Notes in Computer Science

[Schilageter89] SCHLAGETER, R. Object-Oriented Database Systems: Concepts and Perspectives Data Base Systems Proc. Int. Symposium, Oct/89, p. 154-197 Lecture Notes in Computer Science

[Stonebraker91] STONEBRAKER, M. & Kemnitz, G.The Postgres Next-Gerenation Database Management System CACM, 34(10), Oct/91

[Strelich89] STRELICH, T. The Software Life Cycle Support Environment: A computer based framework for developing software systems Proc. ACM SigSoft/ Siggplan Soft. Eng'g Symposium on Practical Soft. Develp. Environ. Sig Plan Notices, 24(2), a989, p. 35-44

[Wasserman89] WASSERMAN, A. I.Tool Integration in Soft. Eng'g Environments Soft. Eng'g Environments Proc. Int. Workshop on Environments, Sept/89, p. 137-149 Lect. Notes in Computer Science

[Zicari92] ZICARI, R. & Medeiros, C. B. Databases for Software Engineering