# Uma experiência na avaliação de Sistemas Gerenciadores de Bancos de Dados Orientados para Objetos com o *Benchmark* 007.

Geraldo Zimbrão da Silva.

COPPE - UFRJ

Agosto - 1994

# Resumo

Este trabalho apresenta resultados do desempenho de diferentes estratégias na avaliação de consultas em sistemas de gerência de bases de dados orientadas a objetos (SGBDOO). Os algoritmos avaliados foram implementados no servidor de objetos GOA que possui capacidade para gerência de armazenamento de objetos persistentes e para processamento de predicados de consulta. São analisados os efeitos das estratégias experimentadas face às técnicas de agrupamento utilizadas, o aproveitamento do *buffer* de páginas e demais características relevantes no projeto de um servidor de objetos.

# Abstract

This work presents performance results for different strategies on object oriented database management systems query evaluation. The analysed algorithms were implemented in the GOA object server which provides the management of persistent objects as well as the evaluation of query predicates. The effects of the evaluated strategies are analysed in light of the clustering tecniques employed, the page buffer effectiveness and several relevant design decisions for builders of object servers.

# Sumário

1) Introdução. ..... 1

2) Trabalhos relacionados. ..... 3

3) O Gerenciador de Objetos Armazenados (GOA) ..... 4

3.a) Descrição sumária. ..... 4

3.b) O Avaliador de Consultas. ..... 5

3.c) Limitações atuais ..... 6

4) Descrição do *benchmark* 007. ..... 8

4.a) Diagrama de Objetos. ..... 10

4.b) Descrição da base. ..... 11

5) Implementação do *benchmark* 007. ..... 12

5.a) Geração da base. ..... 12

5.b) Implementação das consultas (*queries*). ..... 13

5.c) Implementação das travessias (*traversals*). ..... 14

5.d) Implementação das modificações estruturais. ..... 14

6) Conclusões. ..... 15

7) Referências ..... 16

Anexo: Listagem dos programas fonte em C. ..... 17

## 1) Introdução.

Este relatório apresenta as atividades desenvolvidas pelo autor do mesmo durante a realização do primeiro exame de qualificação para a obtenção do título de Doutor em Ciências na COPPE-UFRJ, Programa de Sistemas, linha Banco de Dados.

A pesquisa de técnicas para serem utilizadas em Sistemas Gerenciadores de Objetos, apesar de ter sido muito intensa nos últimos anos, ainda não pôde oferecer soluções plenamente satisfatórias para muitas das questões que surgiram. Podemos observar um grande número de propostas surgirem, principalmente na área de processamento de consultas, agrupamento de objetos, *buffers* de páginas, conversão de referências à objetos em disco para ponteiros em memória (*swizzling*) e cache de objetos.

A pesquisa na área de processamento de consultas para sistemas orientados a objetos tem crescido muito ultimamente. De acordo com Graefe [Grae93], a maioria das propostas utiliza algum tipo de álgebra para o processamento de consultas. Essa álgebra se assemelha à álgebra relacional no sentido em que manipulam conjuntos de dados, mas são genéricas a ponto de manipular arranjos, listas, conjuntos heterogêneos, métodos e herança. Operações associativas constituem uma parte importante nas álgebras orientadas a objetos, pois permitem a redução de grandes volumes de dados para um subconjunto relevante para operações subseqüentes. Desta forma, as técnicas para se percorrer uma relação e para se verificar um predicado sobre uma relação também são encontradas nos Sistemas Gerenciadores de Bancos de Dados Orientados para Obejtos (SGBDOOs). O processamento de consultas em Sistemas Gerenciadores de Objetos (SGOs) ainda não é um problema solucionado, embora a grande maioria dos produtos ofereça uma ferramenta de consulta. Recentemente Carey et al [Care93] analisando o desempenho de SGOs com o *benchmark* OO7 comentam que as linguagens de consulta dos SGOs analisados não foram utilizadas na codificação da maior parte consultas do *benchmark* já que a maioria não fazia uso de otimização, o que tornaria o seu uso irreal.

Diferentes esquemas de agrupamento de objetos têm sido propostos, e tem-se observado que a escolha do tipo de agrupamento tem um impacto muito forte no desempenho de determinadas operações [Catt91]. Os agrupamentos propostos permitem escolher entre agrupar por tipo, por índice, por extensão da classe, por composição etc; ou ainda várias formas combinadas e até mesmo nenhum agrupamento. Em particular, as consultas, inserções e remoções de objetos têm o seu desempenho fortemente afetado pelo agrupamento escolhido.

Embora praticamente todos os ambientes em que são executados os SGOs possuam um cache de disco a nível de sistema operacional, esse cache não é plenamente equivalente à um *buffer* de páginas gerenciado pelo SGO nem pode substituí-lo oferecendo as mesmas vantagens. Entre tais vantagenss podemos encontrar [Catt91]: as páginas se encontram no espaço de endereçamento da aplicação, evita-se a comunicação entre processos e a própria cópia (duplicação) da página em memória, além do que a aplicação pode adotar uma política de troca de páginas mais adequada do que a política genérica adotada pelo sistema operacional.

Em [Catt91], são identificadas várias estratégias de *swizzling*, bem como suas implicações. Vale notar que este é um tema bastante relacionado com o cache de objetos, e que também possui um impacto muito grande em determinadas operações.

Para estudar os efeitos das diferentes estratégias de avaliação de consultas, agrupamento de objetos e buffers de páginas, foi implementado o *benchmark* 007 para o servidor GOA. Algumas adaptacões foram feitas, e em particular objetos muito grandes foram eliminados do esquema.

O uso de um *benchmark*, no caso o 007, foi adotado por oferecer a possibilidade de se testar: operações de criação e inserção de objetos, diferentes consultas, recuperação e atualização de um grande número de objetos seguindo suas referências. Dessa forma, foi possível analisar os efeitos de diferentes estratégias de agrupamento, estratégias de avaliação de consultas e tamanhos de *buffers* de páginas.

Assim, na Seção 2 temos enumerados os artigos derivados do presente trabalho, na Seção 3 é apresentado um pequeno resumo do que é o GOA, na Seção 4 temos a descrição do *benchmark* 007, na Seção 5 a forma como foi implementado, as adaptações feitas e a descrição das operações avaliadas, e finalmente na Seção 6 são relatadas as conclusões. Na Seção 7 temos as referências bibliográficas, e no Anexo a listagem dos programas fontes em C.

## 2) Trabalhos relacionados.

As atividades desenvolvidas ao longo do presente exame de qualificação deram origem a dois artigos:

A) "Explorando o Processamento de Consultas no Servidor de Objetos GOA", [Matt94] - Este trabalho apresenta resultados do desempenho de diferentes estratégias na avaliação de consultas em sistemas de gerência de bases de dados orientadas a objetos.

B) "Evaluating Implementation Strategies for the GOA Object Server", [Zimb94] - Este trabalho apresenta resultados do desempenho de diferentes estratégias na implementação das operações básicas em servidores de objetos para sistemas de gerência de base de dados orientadas a objetos.

Os algoritmos avaliados foram implementados no servidor de objetos GOA que possui capacidade para gerência de armazenamento de objetos persistentes e para processamento de predicados de consulta. Em ambos, foram realizadas medições de desempenho através da geração do *benchmark* OO7 para o GOA, e que é parte fundamental das atividades desenvolvidas. Com os resultados obtidos são analisados os efeitos das estratégias experimentadas face às técnicas de agrupamento utilizadas, o aproveitamento do *buffer* de páginas e demais características relevantes no projeto de um servidor de objetos. Os resultados estão detalhadamente descritos nos dois artigos acima.

## 3) O Gerenciador de Objetos Armazenados (GOA).

O GOA possui uma arquitetura cliente/servidor de objetos. Este trabalho avalia o desempenho no âmbito do servidor GOA. As operações do servidor GOA e relevantes para o presente trabalho são: criação, recuperação, alteração de objetos e acesso associativo a coleções de objetos. Os algoritmos implementados pelo GOA permitem que sejam exploradas diferentes estratégias de agrupamento e consulta a coleções.

## 3.a) Descrição sumária.

O Servidor do GOA é o módulo responsável pela implementação das funções de gerência de objetos armazenados no servidor da arquitetura do SGBDOO GEOTABA [Matt93a]. Num servidor de objetos a unidade de transferência entre o servidor e o cliente é o objeto. O servidor conhece o conceito de objeto e é capaz de executar consultas e métodos sobre os objetos.

O GOA permite a criação de um objeto gerando um identificador único para o mesmo, através do envio do objeto para o servidor. O servidor armazena o objeto na base de dados e retorna para o cliente o identificador do objeto (IDO). A partir de então, pode-se recuperar, remover e alterar o objeto (inclusive o seu tamanho) através de referências a este IDO.

O GOA possui o conceito de coleções de objetos. Em uma coleção, todos os objetos são mantidos juntos, e é possível acessar um objeto da mesma tanto pelo seu IDO como pela sua posição relativa dentro da coleção. Ao se criar um objeto, pode-se impor ou não condições sobre a sua localização. Assim, é possível deixar a cargo do sistema a escolha do local mais apropriado (em geral, após o último objeto criado), ou estabelecer uma coleção para o mesmo residir, como é ilustrado na Figura 1. Essa flexibilidade, como veremos depois, é aproveitada como mais um parâmetro dos testes realizados.

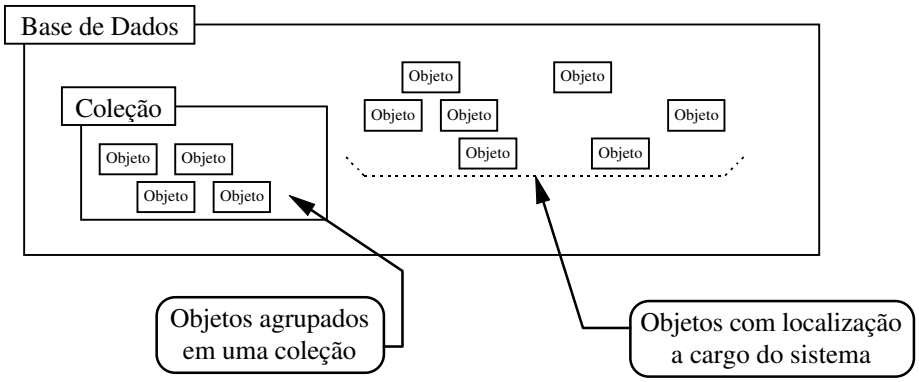


**Figura 1**

**3.b) O Avaliador de Consultas.**

Poucos sistemas apresentam, formalmente ou não, o modelo de consultas adotado no SGBDOO. Em geral, são apresentadas algumas características da linguagem, a sintaxe e exemplos. Poucos autores comentam sobre o processamento da consulta em si. Kim [Kim 90a] apresenta uma proposta de modelo genérico para consultas no Orion baseado em grafos. Uma estrutura de grafo é utilizada para representar as classes, subclasses e as classes dos domínios dos atributos envolvidos na consulta.

No caso do GOA, o modelo de consultas é ligado ao modelo de objetos através da classe *coleção* e a linguagem de consulta é um subconjunto da linguagem global de operação do GEOTABA. Cabe ao GOA, no servidor de objetos, o suporte ao processamento de consultas através da avaliação de predicados. Desta forma, o servidor recebe como uma de suas operações, um código intermediário para realizar o processamento do predicado enviado pelo Gerente de Objetos do Cliente. Este predicado foi previamente analisado pelo processador da linguagem de consulta e posteriormente analisado pelo otimizador de consultas do Gerente de Objetos do cliente.

Existe uma pseudo-sintaxe de cláusulas de qualificação de consultas [Matt93c]. Desta forma, pode ser especificado um código intermediário das operações de avaliação de predicados implementadas na atual versão do servidor de objetos GOA.

Assim como nos SGBDOOs Orion (comercialmente denominado Itasca) [Kim90a] e $O_2$ [Banc92], a linguagem de consulta do GOA permite que os objetos de uma classe sejam percorridos e acessados. Entretanto, o $O_2$ [$O_2$Te93] e o GOA restringem o escopo da classe alvo da consulta para uma coleção, ou seja, o alvo de uma consulta no comando 'select' não pode ser uma classe qualquer. No caso do $O_2$, o alvo da consulta é sobre uma coleção nomeada ('named set') persistente. Esta coleção pode ser associada a uma determinada classe do esquema (ou subclasses da mesma), e que é definida no momento da criação da coleção. Já no caso do GOA o alvo da consulta tem que ser sobre uma classe do tipo *coleção*, definida já na modelagem da aplicação. Contudo, em ambos os casos, em qualquer momento da vida útil do objeto pode-se optar por incluí-lo ou removê-lo das coleções existentes.

Essa restrição não impede que classes que não estejam associadas a coleções sejam referenciadas ao longo do predicado da consulta. A restrição ocorre em respeito ao modelo de objetos adotado, onde uma classe não caracteriza um conjunto ou uma coleção de objetos. Desta forma, o acesso associativo a uma classe só faz sentido se essa classe possui um tratamento de conjunto ou de coleção para seus objetos, segundo a modelagem do usuário. Entretanto, instâncias de subclasses da hierarquia da classe correspondente à coleção alvo também estarão envolvidas no acesso associativo, desde que essas instâncias pertençam à coleção alvo.

As consultas são do tipo acíclicas, ou seja, não envolvem ciclos nem recursividade. Além disso, não foram implementadas cláusulas de predicados que

restringem o seu escopo para subclasses especificadas dentro da hierarquia da classe da coleção alvo. O servidor retorna como resultado da avaliação do predicado para o avaliador da consulta no cliente a lista dos IDOs da classe alvo que qualificam o predicado.

A linguagem de consulta projetada permite a utilização de navegação entre classes e a quantificação de atributos multi-valorados através de operadores tipo IS_IN, HAS_SUBSET e FOR_ALL.

### 3.c) Limitações atuais

No tocante ao gerenciamento de objetos, ao nível do GOA., não foi definido ainda nenhum esquema de contadores de referências, fundamentais para a eliminação automática de objetos que não podem mais ser acessados, e para impedir o surgimento de referências inválidas quando se remover um objeto que é referenciado em outros objetos. Dessa forma, quando um objeto é eliminado, os seus atributos que são também objetos (listas, por exemplo) não são automaticamente eliminados do sistema. Por outro lado, nenhuma garantia é dada de que o objeto eliminado não é referenciado por outros objetos.

Uma outra limitação se refere ao comportamento das coleções. Uma coleção é composta de duas listas: uma lista de IDOs dos objetos inseridos ou criados na coleção, e uma lista de segmentos reservados para os objetos criados na coleção.

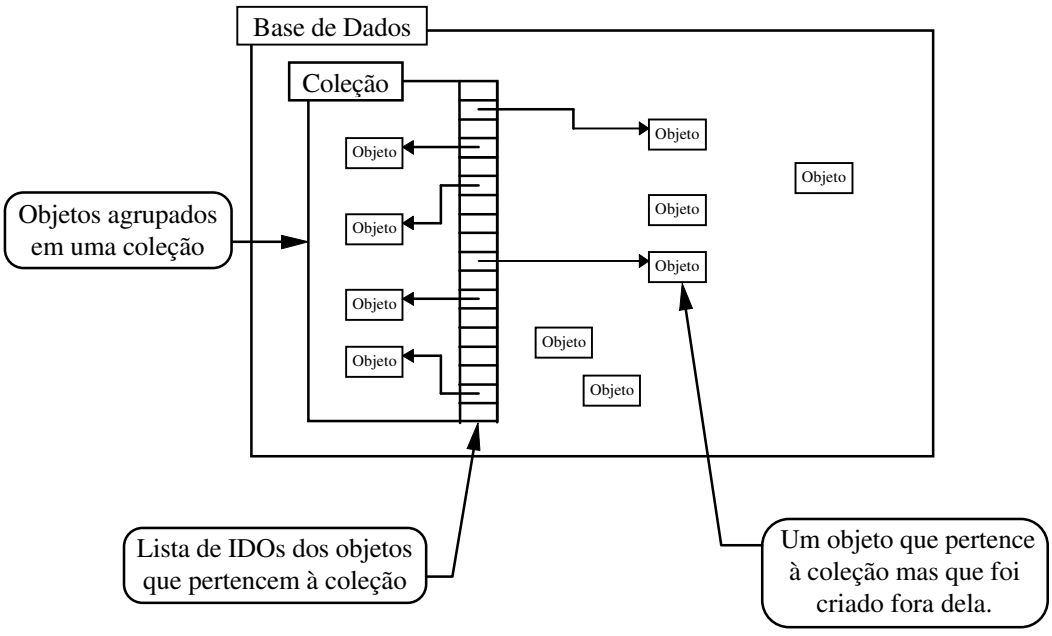


**Figura 2**

Como podemos observar pela Figura 2, uma coleção pode conter objetos que foram criados agrupados dentro da área reservada para a coleção, ou objetos que

foram criados fora da coleção e depois inseridos na mesma. Em ambos os casos, o IDO do objeto estará na lista de IDOs dos objetos que pertencem à coleção. Isso significa que, por exemplo, em uma composição todo-parte, as partes do objeto-todo somente poderão ser agrupadas com o mesmo em uma coleção se cada um dos objetos-parte for criado na coleção. Porém, se os objetos-parte forem criados na coleção, seus respectivos IDOs farão parte da lista de IDOs dos objetos que pertencem à coleção. Dessa forma, não será possível ter uma coleção que contenha apenas IDOs de objetos-todo com os objetos-parte agrupados na mesma, o que dificulta em muito a realização de consultas, visto que os objetos-parte também serão visitados como se fossem objetos-todo.

Em outras palavras, o esquema atual de agrupamento não dá flexibilidade para um agrupamento híbrido por fragmentação e composição. Isto é, objetos que pertencem a uma mesma coleção são agrupados no disco, porém os objetos referenciados pelos objetos da coleção não podem ser armazenados junto aos objetos da coleção. Ou eles estão espalhados pelo disco, ou esses objetos referenciados pertencem a uma outra coleção e fazem parte de outro agrupamento. É importante que o GOA seja modificado para permitir o esquema de agrupamento híbrido.

Por último, nem todos os operadores de seleção foram completamente implementados ainda, o que limitou bastante as alternativas para a construção de consultas. Com pequenas modificações dos algoritmos já implementados, vários operadores poderiam estar disponíveis, o que permitiria um estudo muito mais diversificado.

## 4) Descrição do *benchmark* 007.

O *Benchmark* 007 descreve um amplo teste de operações para avaliação de desempenho em SGBDOOs. Foi desenvolvido no Departamento de Ciência da Computação da Universidade de Wisconsin-Madison, como um primeiro passo em direção ao estabelecimento de um perfil de desempenho para SGBDOOs.

Os fatores determinantes para a escolha do *Benchmark* 007 neste trabalho foram o seu amplo espectro de características testadas e o seu aparecimento como quase um padrão. Entre as características de desempenho avaliadas pelo 007 estão (para uma descrição completa ver [Care93]):

- A velocidade de várias formas diferentes de se percorrer ponteiros, incluindo percorrer ponteiros em dados no cache e no disco, e percorrer também de forma esparsa e densa;
- A eficiência de vários tipos de atualizações, incluindo atualizações em atributos de objeto indexados e não indexados, atualizações repetitivas, esparsas, densas, de dados no cache, e a criação e remoção de objetos;
- O desempenho do processador de consultas em vários tipos diferentes de consultas.

A equipe que desenvolveu o *Benchmark* 007 permite o acesso livre aos fontes, via ftp. Assim, torna-se muito mais fácil adaptá-lo para outras plataformas, e também alterar algumas de suas características em função da aplicação alvo. Além disso, existem três tamanhos de base: pequena, média e grande.

Outro aspecto positivo, e que embora não seja o objetivo desse trabalho surge como subproduto do mesmo, é a possibilidade de comparação de tempo de processamento com outros SGBDOOs, como por exemplo o EXODUS, ONTOS, ObjectStore, $O_2$ etc.

Pela sua constituição, o *Benchmark* 007 produz uma série de números ao invés de apenas um número. Um número apenas possui a vantagem de ser fácil de se usar (e abusar) para comparações entre sistemas. Contudo, um *benchmark* que retorna um conjunto de números produz uma quantidade maior de informações a respeito de um sistema. Um *benchmark* de um número apenas só é realmente útil quando o próprio *benchmark* espelha a aplicação alvo.

Apesar de testar várias características importantes, não há (ainda) testes relativos aos aspectos de sistemas multiusuários. Assim, o *Benchmark* 007 não pode fornecer informações sobre o comportamento do sistema quando mais de uma pessoa o estiver usando. Na prática, siginifica que ele não é adequado para uma vasta gama de aplicações que se utilizam de SGBDOOs. Além dessa deficiência, são avaliadas apenas operações isoladas, não existe o conceito de desempenho de transação. Tal fator não impede que isso seja analisado, porém não produz resultados válidos do ponto de vista

comparativo. Entretanto, os autores prometem englobar essas características na próxima versão do *Benchmark* 007.

Uma outra característica que restringe a utilização do *Benchmark* 007 é que apesar de não modelar nenhuma aplicação específica, ele é orientado para aplicações CAD/CAM/CASE - que são as aplicações mais típicas de um SGBDOO. Dessa forma, os resultados obtidos com a utilização do *Benchmark* 007 devem ser tomados como bastante significativos apenas para estes tipos de aplicações. Para as aplicações que não se assemelham aos padrões de operações das aplicações CAD/CAM/CASE, seus resultados podem ter apenas um valor secundário.

Foram feitas algumas adaptações no esquema de dados no sentido de adequar a base às limitações atuais do poder de representação de objetos no GOA. Assim, foram eliminados do esquema original os atributos do tipo texto grande, e a classe Manual.

**4.a) Diagrama de Objetos.**

A Figura 3 representa o diagrama de objetos da base do 007 segundo a notação Coad & Yourdon [COAD91]

**Figura 3**

## 4.b) Descrição da base.

A seguir, uma descrição em uma pseudo-linguagem semelhante a C++ das classes:

```
class Document {
        OID             oid;
        char            title[40];
        int             date;
        CompositePart   part    <-> CompositePart::document;
} with extent Documents;

class Connection {
        OID             oid;
        char            type[10];
        int             length;
        AtomicPart*     from    <-> AtomicPart::to;
        AtomicPart*     to      <-> AtomicPart::from;
} with extent Connections;

class DesignObject {
        OID     oid;
        char    type[10];
        int     buildDate;
};

class Module: public DesignObject {
        Set(Assembly*)  assemblies <-> Assembly::module;
        OID             design_root;
} with extent Modules;

classe Assembly: public DesignObject {
        AssmType assm_type;
        OID             super_assembly  <-> ComplexAssembly::sub_assemblies;
        OID             module          <-> Module::assemblies;
};

class ComplexAssembly: public Assembly {
        Set(Assembly*)  sub_assemblies <-> Assembly::super_assembly;
} with extent ComplexAssemblies;

class BaseAssembly: public Assembly {
        Bag(CompositePart*)     componentsPriv  <-> CompositePart::usedInPriv;
        Bag(CompositePart*)     componentsShar  <-> CompositePart::usedInShar;
} with extent BaseAssemblies;

class CompositePart: public DesignObject {
        Document*               document        <-> Document::part;
        Bag(BaseAssembly*)      usedInPriv      <-> BaseAssembly::componentsPriv;
        Bag(BaseAssembly*)      usedInShar <-> BaseAssembly::componentsShar;
        Set(AtomicPart*)        parts           <-> AtomicPart::partOf;
        AtomicPart*             rootPart;
} with extent CompositeParts;

class  AtomicPart: public DesignObject {
        int             x;
        int             y;
        OID             doc_oid;
        Set(Connection*)        to      <-> Connection::from;
        Set(Connection*)        from    <-> Connection::to;
        CompositePart*          partOf  <-> CompositePart::parts;
} with extent AtomicParts;
```

## 5) Implementação do *benchmark* 007.

Foram criados os seguintes arquivos fonte em C para implementar a versão do *benchmark* 007 para o GOA (a listagem completa está no Anexo, e os arquivos estão disponíveis em meio magnético - contactar o autor ou verificar no diretório BD/zimbrao/coperel):

| Arquivo | | Descrição |
|---|---|---|
| assembly.c | : | Classe Assembly |
| atomicpart.c | : | Classe AtomicPart |
| baseassembly.c | : | Classe BaseAssembly |
| classes.c | : | Classe DesignObject e algumas rotinas de inicialização |
| complexassembly.c | : | Classe ComplexAssembly |
| compositepart.c | : | Classe CompositePart |
| connection.c | : | Classe Connection |
| criabase.c | : | Criação do volume BDTRABA para conter a Base |
| document.c | : | Classe Document |
| gerabd.c | : | Geração da Base |
| minicollection.c | : | Listas para os relacionamentos |
| module.c | : | Classe Module |
| rotinas.c | : | Rotinas para manipular as extensões das classes |
| objeto.c | : | Rotinas de suporte para a representação dos objetos |
| queries.c | : | Consultas |
| insert.c | : | Inserções |
| traverse.c | : | Travessias |
| classes.h | : | Declaração das Classes do modelo |
| objeto.h | : | Declaração das estruturas para a representação dos objetos |

## 5.a) Geração da base.

A base de dados é gerada em duas etapas principais. Na primeira etapa, são criados todos os objetos da classe CompositePart, Document, AtomicPart e Connection, enquanto na segunda etapa são criados todos os objetos da classe Module, BaseAssembly e ComplexAssembly.

A medida em que é criado cada um dos objetos da classe CompositePart, é criado também um objeto da classe Document e estabelecido o relacionamento entre ambos. Essa criação simultânea é intencional, e assegura uma certa localidade dos dados. Além dessa instância de Document, são criados os objetos da classe AtomicPart e Connections, e os respectivos relacionamentos.

Os objetos da classe ComplexAssembly são criados recursivamente, de forma semelhante à uma busca em profundidade: uma instância somente está concluída

quando todas as suas sub-assemblies também estiverem concluídas. Quando a recursão atinge o nível dos objetos da classe BaseAssembly são finalmente escolhidos randomicamente os objetos da classe CompositePart que farão parte dos relacionamentos ComponentsPriv e ComponentsShared. Nesse momento, a localidade dos dados existe apenas na hierarquia de objetos das classes ComplexAssembly e BaseAssembly.

Segundo os autores do *Benchmark* 007, a base é gerada dessa forma para simular o que normalmente ocorre em aplicações típicas: uma grande localidade dos objetos menores (CompositeParts, Documents, AtomicParts e Connections; ComplexAssemblies e BaseAssemblies), porém quase nenhuma localidade dos objetos mais complexos em relação aos seus componentes (BaseAssemblies e CompositeParts).

Nesse momento, vale ressaltar um recurso oferecido pelo GOA: pode-se escolher o agrupamento utilizado no momento de criar um objeto. Assim, existe a opção de se criar um objeto próximo ao último objeto criado, ou na extensão da classe próximo ao último objeto criado da mesma classe (entende-se por extensão de uma classe o conjunto de objetos que pertence à mesma).

Para tirar proveito desse recurso, foi definido um parâmetro de agrupamento que permite escolher entre dois esquemas básicos: agrupamento por ordem cronológica de criação, ou agrupamento pela extensão das classes. Esse parâmetro deve ser passado quando da execução do programa que gera a base (gerabd.c).

Note-se que outras formas híbridas de agrupamentos podem ser definidas combinando-se ora o agrupamento pela extensão da classe, ora o agrupamento pela ordem cronológica de criação. Em particular, essa flexibilidade do GOA permite que se obtenha um bom proveito da localidade dos dados.

### 5.b) Implementação das consultas (*queries*).

Devido às limitações do processador de consultas do GOA, nem todas as consultas especificadas na descrição do benchmark 007 foram implementadas. A seguir, é feita a descrição de cada uma das consultas implementadas.

a) Consulta 2: “Select ap from AtomicParts where ap.date < 1011”. Essa consulta pretende retornar aproximandamente 1% dos objetos da classe AtomicPart.

b) Consulta 3: “Select ap from AtomicParts where ap.date < 1101”. Essa consulta pretende retornar aproximandamente 10% dos objetos da classe AtomicPart.

c) Consulta 4: “Select doc.part from Documents where doc.title isin Titles”. “Titles” é um conjunto de títulos aleatoriamente gerados.

d) Consulta 5: “Select ap from AtomicParts where ap.part_of.document.date < 1101”. Essa consulta é implementada de duas formas diferentes. A primeira forma é percorrendo a extensão AtomicParts e realizando a seleção. A segunda forma se divide em três etapas: percorrer primeiro a extensão Documents, selecionando os objetos da classe Document que satisfazem a condição de seleção; percorrer extensão CompositeParts selecionando os objetos da classe CompositePart que apontam para os objetos selecionados no passo anterior; e finalmente percorrer a extensão AtomicParts selecionando os objetos da classe AtomicPart que apontam para os objetos selecionados no passo anterior.

**5.c) Implementação das travessias (*traversals*).**

Novamente devido às limitações atuais do GOA, não foram implementadas todas as travessias descritas no *benchmark* 007. A seguir, é feita a descrição de cada uma das travessias implementadas.

a) Travessia 1: “percorrer a hierarquia de objetos da classe Assembly; para cada objeto da classe BaseAssembly visitado, visitar cada um dos objetos da classe CompositePart referenciados por componentsPriv; para cada objeto da classe CompositePart visitado, realizar uma busca em profundidade no seu grafo de objetos da classe AtomicPart; retornar o número de objetos da classe AtomicPart visitados”.

b) Travessia 2: “Repetir a travessia 1, porém alterando os objetos da classe AtomicPart a medida em que são visitados”.

c) Travessia 6: “percorrer a hierarquia de objetos da classe Assembly; para cada objeto da classe BaseAssembly visitado, visitar cada um dos objetos da classe CompositePart referenciados por componentsPriv; para cada objeto da classe CompositePart visitado, visitar o objeto da classe AtomicPart referenciado por rootPart; retornar o número de objetos da classe AtomicPart visitados”.

**5.d) Implementação das modificações estruturais.**

Apenas a seguinte inserção foi implementada: “Criar cinco novos objetos da classe CompositePart, bem como o número correspondente de objetos associados (Document, AtomicPart e Connection), e inserir esses cinco novos objetos na base escolhendo aleatoriamente objetos da classe BaseAssembly”.

As remoções não foram implementadas pois, como já foi mencionado antes, não existe nenhum esquema de contadores de referência nem de eliminação automática de objetos que não podem ser referenciados. Isso dificultaria sobremaneira a implementação das remoções, além de mascarar completamente o seu resultado.

## 6) Conclusões.

A realização dos testes permitiu, além das conclusões relatadas nos artigos relacionados a esse trabalho, [Matt94] e [Zimb94], a correção de pequenos erros de digitação encontrados nos módulos que implementam o GOA. Os erros foram prontamente corrigidos e registrados.

Além disso, tornou-se evidente que o pequeno dimensionamento do buffer de páginas do GOA possui uma grande influência no desempenho dos testes. Um buffer maior se tornou necessário, e com isso algumas decisões de projeto tiveram de ser revistas: em particular, a modificação do esquema de busca das páginas dentro do buffer, que deverá deixar de ser busca sequencial para ser busca binária. As modificações decorrentes já estão sendo realizadas, e espera-se concluí-las a tempo de incluir os novos resultados no segundo artigo.

Futuramente, espera-se também que seja resolvida a deficiência relatada em 3.c, relativa à impossibilidade de se criar objetos-parte dentro do espaço de armazenamento de objetos-todo, independente das coleções criadas. Em termos de implementação são necessárias poucas modificações. Uma sugestão seria que a rotina de criação de objetos em uma coleção possuísse um parâmetro (do tipo lógico) que indicasse se o IDO desse objeto deve ser ou não incluído na lista de IDOs dos objetos que pertencem à coleção.

## 7) Referências.

[Banc92] Bancilhon, F. Delobel, C. Kanellakis, P. (editores) "Building an Object-Oriented Database System: The Story of $O_2$", Morgan Kaufmann Publishers, 1992.

[Care93] Carey, M.J. DeWitt, D. Naughton, J. "The OO7 Benchmark" Proc ACM SIGMOD International Conference on Management of Data, junho 1993, pp.12-21.

[Catt91] Cattell, R. G. G. "Object Data Management" Addison-Wesley ,1991.

[Coad91] Coad, Peter e Yourdon, Edward "Object Oriented Analysis" Prentice Hall, 1991.

[Grae93] Graefe, G. "Query evaluation techniques for large databases" ACM Computing Surveys, v.25(2), junho 1993, pp.73-172.

[Kim 90] Kim, W. "Introduction to Object-Oriented Databases" The MIT Press, 1990.

[Matt93a] Mattoso, M. L. Q. "Documentação do Servidor de Objetos - GOA", Relatório Técnico do Projeto TABA, RT-01/93, março 1993.

[Matt93b] Mattoso, M. L. Q. "Aspectos de Paralelismo na Gerência de Dados e Objetos do GEOTABA", Dissertação de Tese de Doutorado, Programa de Engenharia de Sistemas e Computação, COPPE/UFRJ, abril 1993.

[Matt94] Mattoso, M. L. Q., G. Zimbrão da Silva "Explorando o Processamento de Consultas no Servidor de Objetos GOA", 9º Simpósio Brasileiro de Bancos de Dados, Setembro de 1994.

[$O_2$Te93] $O_2$ Technology, "The $O_2$ User's Manual" Version 4.35, julho 1993.

[Zimb94] Zimbrão da Silva, Geraldo, Mattoso, M. L. Q. "Evaluating Implementation Strategies for the GOA Object Server", *pre-print*.

# Anexo: Listagem dos programas fonte em C.