Uma Avaliação Experimental sobre Técnicas de Indexação em Bancos de Dados Orientados a Objetos.

Conference Paper · January 1999

Source: DBLP

CITATION
1

READS
182

2 authors:

Eduardo Ogasawara
Federal Center for Technological Education Celso Suckow da Fonseca
262 PUBLICATIONS **2,283** CITATIONS

SEE PROFILE

Marta Mattoso
Federal University of Rio de Janeiro
357 PUBLICATIONS **4,641** CITATIONS

SEE PROFILE

Uma Avaliação Experimental sobre Técnicas de Indexação em Bancos de Dados Orientados a Objetos[♦]

Eduardo Soares Ogasawara
senna@cos.ufrj.br

Marta Lima de Queirós Mattoso
marta@cos.ufrj.br

Programa de Engenharia de Sistemas e Computação - COPPE / UFRJ
Caixa Postal 68511, Rio de Janeiro, RJ, 21945-970

Resumo

As técnicas de indexação para SGBDOO propostas na literatura podem ser classificadas em três grupos, a saber: (i) índices que apoiam consultas sobre hierarquia de classes, (ii) índices que apoiam expressões de caminho e (iii) índices híbridos que apoiam consultas envolvendo expressões de caminho sobre hierarquia de classes. Este trabalho apresenta uma análise de técnicas de indexação estrutural para hierarquia de classes que foi realizada no sentido de avaliar os índices mais adequados ao Sistema de Gerência de Objetos GOA++. A partir dos resultados desta análise, optou-se por validar experimentalmente alguns índices representativos da categoria de índices hierárquicos existentes na literatura. Desta forma, são apresentados resultados experimentais para a implementação dos índices SCI (Índice para Única Classe) e CHI (Índice para Hierarquia de Classe) sobre o GOA++, contendo diversas situações representativas da sensibilidade do índice em relação ao predicado de seleção. Os resultados apresentados no presente trabalho não foram observados em outros trabalhos existentes na literatura em relação ao comportamento dos índices para consultas que buscam faixas de valores do atributo indexado na hierarquia.

Palavras chaves: métodos de acesso, indexação, bancos de dados orientados a objetos.

Abstract

Existing OODBMS indexing techniques in the literature can be classified in three groups: (i) class hierarchy indexes; (ii) path expressions indexes; and (iii) hybrid indexes that support path expressions in a class hierarchy. This work presents an analysis of structural indexing techniques to support class hierarchy in order to identify which indexes are more adequate to the GOA++ Object Management System. Experimental results for SCI (Single Class Index) and CHI (Class Hierarchy Index) implemented in GOA++ are shown, including significant details of index sensibility in a query selection predicate. These results were not previously reported in the related work of the literature when it comes to index behavior in range queries over an indexed attribute.

***Keywords**: access methods, indexing, object-oriented databases.*

[♦] Trabalho parcialmente financiado pela CAPES e CNPq.

1. Introdução

Assim como no modelo relacional, os índices são componentes essenciais nos sistemas de banco de dados orientados a objetos para melhor apoiar o processamento de consultas. Em particular, os índices devem acelerar a varredura de objetos que satisfaçam um determinado predicado de seleção. Independentemente do SGBD ser orientado a objetos ou relacional-objeto, surge a necessidade de novas organizações que acelerem a implementação da representação de hierarquias e relacionamentos complexos. Bertino e Foscoli [1] classificam as técnicas de indexação para os bancos de dados orientados a objetos como comportamentais e estruturais.

As técnicas de indexação comportamentais objetivam prover uma execução eficiente de consultas que contenham invocação de métodos. Elas são baseadas num pré-cálculo ou num "cache" de método, armazenando os resultados num índice. O maior problema desta abordagem consiste em detectar as mudanças nos objetos e invalidar o resultado de um método.

Já os índices estruturais se baseiam em valores de atributos de objetos. Estes índices são importantes visto que linguagens de consulta como a OQL (*Object Query Language*) do padrão ODMG 2.0 [4], permitem usar predicados que envolvam condições de acesso a objetos ao longo de uma ligação de referência entre eles (expressões de caminho), permitindo assim a navegação sobre o grafo de composição de objetos [15]. As técnicas de indexação estrutural podem ser classificadas em três grupos, a saber: (i) técnicas que apoiam consultas sobre hierarquia de classes [8], (ii) técnicas que apoiam expressões de caminho [3, 7] e (iii) técnicas híbridas que apoiam consultas envolvendo expressões de caminho sobre hierarquia de classes [1, 2]. Uma análise cuidadosa e extensa envolvendo índices das categorias (i) e (iii) foi apresentada por Stehling e Nascimento [16] evidenciando os aspectos positivos e negativos dos principais índices dessas categorias.

Fizemos uma análise qualitativa detalhada em Ogasawara e Mattoso [14] comparando os três grupos de técnicas de indexação estrutural no sentido de avaliar as técnicas mais adequadas ao Sistema de Gerência de Objetos GOA++ [10, 11, 12]. Após esta análise, optamos por validar experimentalmente alguns índices representativos das três categorias existentes. Em particular, implementamos índices derivados da estrutura de árvore B+. Embora os índices com origem em estruturas espaciais [16, 14] possuam diversos aspectos positivos, como melhoria de desempenho para leitura em relação aos baseados em árvore B+, esses índices possuem um custo muito alto para modificação de valores, além de complexidade maior para a implementação no SGBD.

Neste trabalho apresentamos os resultados de desempenho obtidos com a avaliação de dois índices da categoria de hierarquia de classes sobre o GOA++, a saber, os índices SCI (*single class index*) e CHI (*class hierarchy index*). Nosso trabalho se assemelha ao de Kim et al. [8] pois ambos se preocupam em avaliar a relação custo/benefício entre os índices SCI e CHI. Entretanto, enquanto que em Kim et al. [8] os resultados se baseiam em simulações, neste trabalho as avaliações foram realizadas num sistema real, envolvendo o armazenamento de objetos persistentes. Nesse sentido, observamos que o uso de um sistema de paginação de banco de dados evidenciou o impacto das diferenças estruturais dos nós folha de SCI e CHI. Tal situação muitas vezes fica encoberta devido à fixação do parâmetro “número de páginas do índice acessadas”, no modelo de simulação, como fator determinante de desempenho.

Segundo Kim et al., em geral, o índice CHI tem um desempenho melhor que o SCI. Entretanto, nossos resultados apresentam consultas típicas com faixas de valores, envolvendo

de 5 a 10% do domínio da chave, onde o índice SCI possui um desempenho superior ao CHI. De fato, alguns resultados de comparações [16, 7] indicam que o CHI não é eficiente para consultas a valores de intervalo que não envolvam a hierarquia completa. Porém, evidenciamos experimentalmente que essa observação depende da taxa de repetição dos valores da chave dentre os objetos indexados. Verificamos que para consultas de intervalo, quando há muita repetição, em geral o CHI se comporta melhor, sendo eficiente mesmo para consultas que não envolvem a hierarquia completa. Quando não há muita repetição, em geral o SCI apresenta melhor desempenho.

Este trabalho contribui analisando a relação custo/benefício entre o uso dos índices SCI e CHI, apresentando resultados não observados em trabalhos da literatura. Esses desempenhos são relevantes não só para o otimizador de consultas do SGBD, como também para o projeto físico de uma aplicação orientada a objetos. O trabalho está organizado em cinco seções. A segunda seção apresenta de forma introdutória as organizações de índices utilizadas neste trabalho. Uma apresentação mais didática contendo exemplos de uso e características principais das três categorias de índices estruturais pode ser vista em Ogasawara e Mattoso [14]. A terceira seção apresenta as ferramentas utilizadas para a implementação dos índices, ou seja, o sistema GOA++, o benchmark OO7 e as características principais das classes que implementam os serviços de indexação. A quarta seção discute e compara os resultados obtidos com a implementação. A última seção apresenta a conclusão, indicando trabalhos futuros.

2. Índices Estruturais para Hierarquia de Classes

Os índices apresentados nesta seção têm em comum a estrutura básica de armazenamento em forma de árvore B+. O que diferencia um índice de outro são os nós folha. Os nós intermediários (Figura 1) apresentam *f* tuplas, onde cada *tupla* é da forma *[chave de indexação, ponteiro]*. Cada *chave de indexação* tem a forma *[tamanho da chave, valor da chave]*. O tamanho da chave representa o número de bytes ocupados pelo valor da chave. Cada nó intermediário da árvore B apresenta *f* ponteiros de saída (*f* entre *d* e *2d*, onde *d* é a ordem da árvore B). O número de ponteiros que saem da raiz da árvore é de *2* a *2d*. O ponteiro de cada tupla contém o endereço físico do próximo nível do nó de indexação da árvore. Se for necessário inserir uma tupla num nó que contenha *2d* tuplas, o nó é partido e as *2d + 1* tuplas são distribuídas em dois nós.

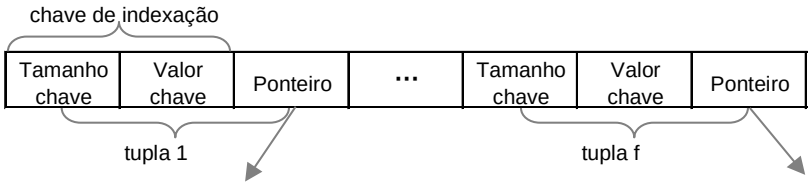

Figura 1 - Representação de um nó intermediário

Uma das diferenças entre sistemas relacionais e sistemas orientados a objetos é a possibilidade de uso da abstração de herança para representar relacionamentos de generalização com semântica de especialização, classificação, enumeração [4], entre outros, ou seja, classes podem ser especializadas em subclasses. Em termos genéricos, uma classe pode ter um número qualquer de subclasses e superclasses. A maioria dos sistemas

apresentam uma classe raiz, denominada *objeto,* de onde todas as outras classes são hierarquicamente descendentes.

A partir deste conceito de generalização, os esquemas orientados a objetos capturam a semântica do relacionamento "é um" entre um par de classes. Isto influencia a semântica de instanciação de objetos. Um dos impactos desta influência está no escopo de uma consulta sobre uma determinada classe, que pode ser apenas sobre instâncias daquela classe ou incluir todas as instâncias de classes descendentes. Um outro grande impacto está no domínio *D* de um atributo de uma classe *C*, que pode englobar apenas a classe *C* ou todas as subclasses. Esta semântica de instanciação de objetos força mudanças significativas no modo com que os sistemas de banco de dados usam índices [8].

Diversos índices têm sido propostos como alternativa aos índices SCI e CHI, como por exemplo, árvore-H [17, 6], árvore-χ [5], multi-chave[13], entre outros [16]. Entretanto, optamos por analisar apenas o SCI e CHI devido ao alto custo e/ou complexidade de atualizações, comum a todas as demais técnicas. Futuramente pretende-se estender a árvore-R disponível no GOA++[9] para avaliar a árvore-χ.

2.1. Índice SCI

O índice para uma única classe **SCI** (*Single Class Index*) corresponde ao índice mais tradicional para classes, porém não apresenta nenhum mecanismo específico para tratar a herança. A estrutura SCI faz com que o banco de dados mantenha um índice sobre um determinado atributo para uma única classe da hierarquia. Isto significa dizer que para apoiar a avaliação de uma consulta cujo escopo de acesso esteja na raiz ou no meio de uma hierarquia de classes, o sistema deve manter um índice para cada classe na hierarquia e fazer as combinações correspondentes.

Como vemos na Figura 2, os nós folha do SCI são constituídos pelo tamanho do registro, tamanho da chave, valor da chave, ponteiro para página de sobrecarga, número de elementos que apresentam aquela chave e a lista de identificadores de objetos (OID).

Tamanho Registro	Tamanho chave	Valor chave	Ponteiro para Página de Sobrecarga	N° OIDs = n	$oid_1 \ldots oid_n$

Figura 2 - Nó folha do índice para cada classe (SCI) [8]

Um nó folha pode ser pequeno (menor que a página de indexação) ou grande (maior que a página de indexação). Um registro pequeno pode crescer para um registro grande ou simplesmente transpor os limites do nó folha. Existem diferentes formas para lidar com este tipo de situação. Kim et al. [8] sugerem uma forma consistente para resolver o problema. Se um registro pequeno crescer a ponto de transpor os limites da página de índice, mas continuar um registro pequeno, então a página de índice é dividida. Se, por outro lado, um registro de índice se tornar um registro grande, um nó folha é totalmente atribuído a ele, sendo que as partes que não couberem nesta página são armazenadas em páginas de sobrecarga. É para este propósito que existe a área de ponteiros para páginas de sobrecarga. Se o valor deste campo for zero, pode-se assumir que o registro de índice cabe totalmente na página corrente.

2.2. Índice CHI

O índice para hierarquia de classes **CHI** (*Class Hierarchy Index*) representa um índice para toda uma hierarquia de classes. A avaliação de uma consulta sobre a hierarquia ou qualquer sub-hierarquia é feita apenas sobre um único índice. Os nós folha (Figura 3) de um

índice CHI são constituídos pelo tamanho do registro, tamanho da chave, valor da chave, ponteiro para página de sobrecarga, diretório de classes (para cada classe na hierarquia que apresente aquela chave) e a lista de OIDs (IDentificadores de Objetos) fragmentados pelas classes presentes no diretório. O diretório de classes é decomposto em pares de identificador de classe e deslocamento no registro (indicando o começo da lista de OIDs daquela classe específica).

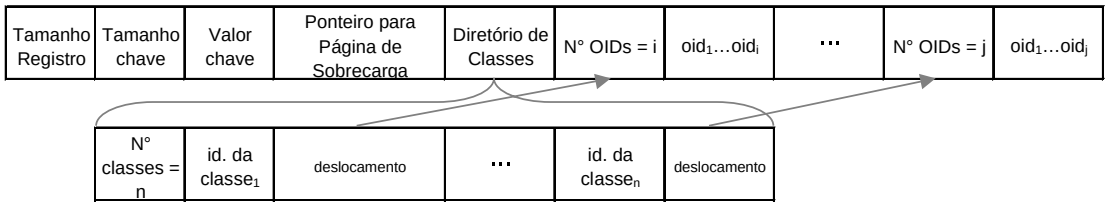

Figura 3 - Nó folha do Índice para Hierarquia de Classes (CHI) [8]

Nesta organização, um índice para hierarquia de classes é mantido sobre um atributo numa hierarquia de classe que contenha *n* classes enraizadas a partir de uma classe *C*. Note que se o registro folha do índice fosse organizado simplesmente como um SCI, não tendo um diretório de classes, seria necessário uma busca exaustiva sobre a lista de OIDs do resultado final da consulta para extrair os OIDs que não pertencessem às classes relevantes à consulta. Além disto, se uma classe fosse removida, a lista de instâncias da classe teria que ser removida do índice de hierarquia de classes. O CHI facilita a remoção de qualquer classe da hierarquia.

3. Ambiente de Avaliação Experimental

Para fazer a avaliação experimental dos índices, optamos por utilizar o sistema GOA++ e a base de dados OO7 [18] desenvolvida para avaliar desempenho de SGBDOOs. Os testes foram realizados num Pentium II 266MHz com 128Mb de memória RAM usando o servidor GOA++, executando no Microsoft Windows NT 4.0 Server.

3.1. O GOA++

O GOA++ é um gerente de objetos armazenáveis, com modelo de dados compatível com ODMG 2.0, utilizando portanto a ODL como linguagem de definição de dados e OQL como linguagem de consulta. Apesar de não ser um sistema gerenciador de banco de dados completo (não existência de controle de transações, por exemplo), vários recursos estão nele implementados, como por exemplo, capacidade de processamento de consulta, gerência de esquema e cache, além de recursos de distribuição e paralelismo. O GOA++ possui APIs para serem utilizadas diretamente com C++ e JAVA.

A Figura 4 ilustra a arquitetura em que o servidor de objetos GOA++ é utilizado. Esta arquitetura está baseada no paradigma cliente-servidor, utilizando como base para comunicação o protocolo TCP/IP, escolhido por ser o protocolo mais apoiado numa vasta gama de arquiteturas via Internet.

No núcleo do sistema GOA++ fica toda a parte de gerência dos objetos, incluindo serviços de persistência de objetos com gerência de cache, manipulação de atributos, integridade dos relacionamentos, gerência de esquema e processamento de consultas em OQL. Sendo assim, as estruturas de índice (representada pela caixa IDX) foram implementadas no núcleo do GOA++ fazendo uso dos serviços de armazenamento de objetos, gerência de cache e sendo usado pelo processador de consultas. Para essa implementação foram criadas classes

específicas, dentre as quais destacam-se GoaArvoreB, GoaIndice e GoaChave (Figura 5), descritas a seguir.

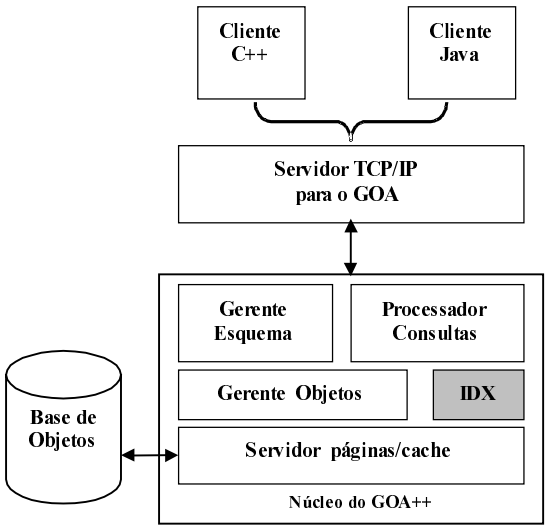

Figura 4 - Arquitetura GOA++

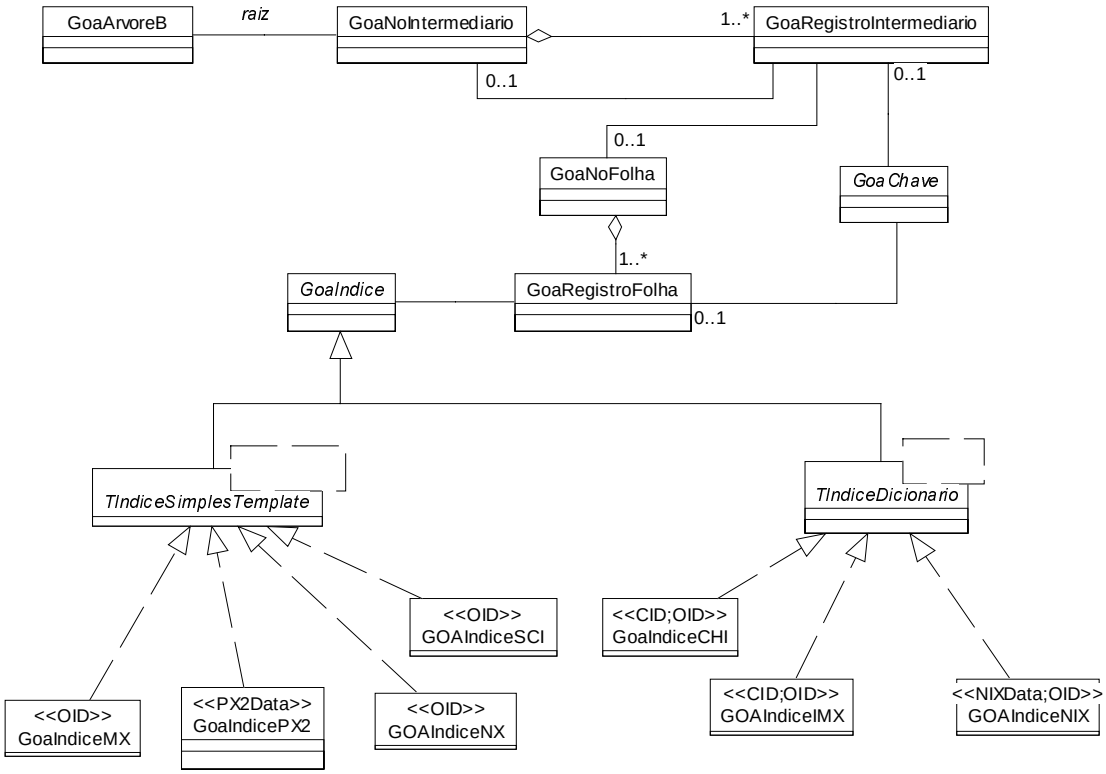

Figura 5 - Modelo UML dos Índices no GOA++

3.1.1. Classe GoaArvoreB

A classe GoaArvoreB encapsula as funcionalidades de uma árvore B+ - inclusão, atualização, exclusão e consulta de registros - de acordo com a configuração do índice a ser

usado. Em função de sua adaptabilidade aos diferentes tipos de índices, toda árvore B+ também é um objeto persistente GOA, mantendo seus metadados armazenados na própria base de dados.

3.1.2. Classe GoaIndice

A classe GoaIndice é uma classe abstrata que encapsula a funcionalidade básica dos índices propriamente ditos, ou seja, os aspectos existentes nos registros dos nós folha que diferenciam um índice do outro. As diferenças entre os índices são vistas nas especializações de cada índice apresentado no modelo como GoaIndiceSCI, GoaIndiceCHI, etc. Nas especializações são incorporadas a semântica de integridade de cada índice, incluindo a forma de acesso e persistência das informações.

3.1.3. Classe GoaChave

A classe GoaChave é uma classe abstrata que detém as principais operações de uma chave de indexação numa árvore B+. Ela é basicamente utilizada para formar os registros intermediários e os registros folhas, associando as chaves aos índices. A classe GoaChave também pode ser usada nos critérios para realização de consultas na árvore.

3.2. O benchmark

A base de dados do benchmark OO7 (Figura 6) foi adotada para realizar a análise de desempenho dos índices apoiados pelo GOA++. Uma das características importantes para a sua adoção é a riqueza do seu modelo, em função da existência tanto de hierarquia de classes como também de expressões de caminho. Assim, embora este artigo se detenha em comparações dos índices hierárquicos, num âmbito maior o modelo OO7 pode ainda ser usado para avaliar os índices para expressão de caminho, como também os índices híbridos, permitindo inclusive preparar uma bateria de testes que compare o uso dos índices híbridos versus possíveis combinações de índices hierárquicos com índices de expressões de caminho. Além disto, o OO7 apresenta largo uso em análise de desempenho no modelo OO e, particularmente, no GOA++.

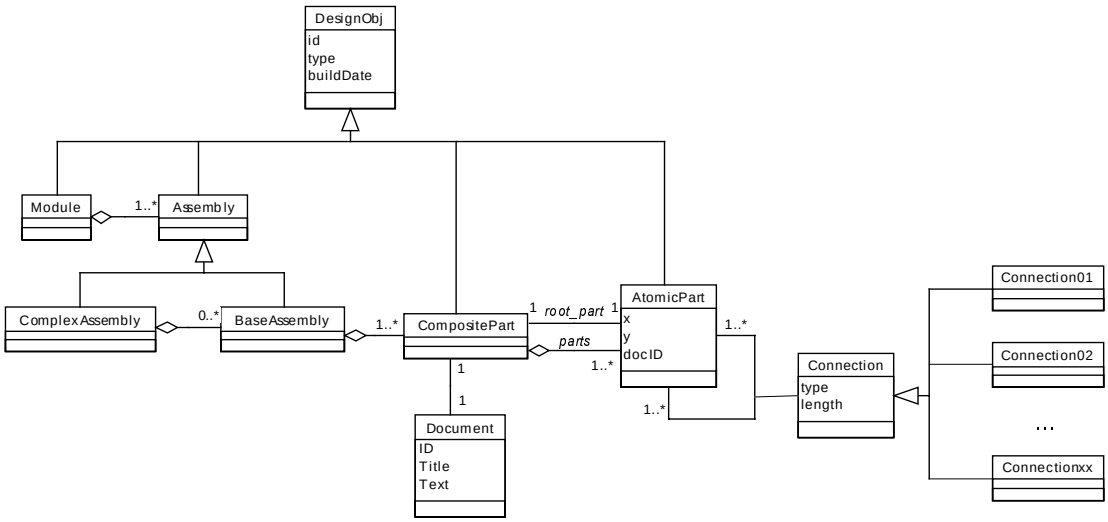

Figura 6 - Modelo UML do Benchmark OO7

Uma única e importante modificação realizada no modelo OO7 foi a inclusão de especializações na classe *Connection*. Basicamente o atributo tipo foi trocado por um número

constante de especializações, de modo a realizar mais testes sobre hierarquia de classes numa distribuição uniforme de objetos.

Foram realizados testes e estudos com um número variável (4, 8, 16) de classes sob a hierarquia de *Connection*. Como os resultados foram bastante similares, as análises ficaram restritas à configuração para 16 classes na hierarquia, no sentido de se tentar representar modelos orientados a objetos com uma hierarquia de classes rica e, ao mesmo tempo, acentuar as características de sensibilidade dos índices quanto ao número de classes. Os testes foram realizados na configuração média do OO7 (Tabela 1).

Tabela 1 - Configuração Média do OO7

Classes	Qtd.	Classes	Qtd.
Assemblies	166	AtomicParts	10,000
CompositeParts	500	Connections	60,000
Documents	500		

4. Resultados Experimentais

A distribuição dos valores de chave sobre a hierarquia de classes pode ter impacto na comparação entre os índices. De acordo com Kim et al.[8], denomina-se distribuição disjunta para o atributo indexado sobre as classes numa hierarquia a ocorrência de valores do atributo confinados a instâncias de uma única classe na hierarquia. Se os valores do atributo estão distribuídos ao longo de todas as classes na hierarquia, tem-se uma distribuição conjunta para o atributo indexado. Vale ressaltar que o conceito de distribuição conjunta/disjunta de valores é independente do conceito de distribuição uniforme/não uniforme de valores para o atributo indexado.

Assim, numa distribuição disjunta de valores de atributos sobre as classes o CHI é menos eficiente que o SCI, já que o SCI pode varrer apenas as classes relevantes à consulta. Além disso, analisando a sensibilidade de armazenamento dos índices em função da distribuição de chaves, verifica-se que o SCI consome pouco espaço em distribuições disjuntas. Em nossos experimentos, não apresentamos resultados para as consultas com distribuição disjunta, pois analiticamente é possível verificar que o SCI é sempre melhor. Sendo assim, os resultados apresentados nesta seção sempre dizem respeito à distribuição conjunta de valores tanto para o SCI quanto para o CHI.

Dividimos nossos experimentos em dois grupos de consultas. No primeiro grupo avaliamos consultas envolvendo o operador de igualdade (consulta pontual), enquanto que no segundo grupo avaliamos consultas por faixas de valores (consulta por intervalo). Para os dois grupos utilizamos a mesma base e fizemos variações no número de classes envolvidas e na taxa de repetição dos valores da chave, medindo o tempo de resposta.

Para cada consulta avaliada, foram realizadas trinta medições e reportada a média para cada execuções com partida quente. As consultas foram executadas alternadamente para evitar a influência do sistema de paginação do GOA++.

4.1. Consultas pontuais

As consultas pontuais refletem a rapidez do índice em responder a buscas por valores específicos. Conforme observado em Kemper e Moerkotte [6], métodos que agrupam objetos por valor, como é o caso do CHI, são imunes ao número de classes consultadas. O mesmo não ocorre para o SCI, pois vários índices têm que ser percorridos; logo, o desempenho degrada proporcionalmente ao número de classes. Esses resultados foram de fato confirmados nos

experimentos realizados. O índice CHI obteve comportamento linear para todas as variações do número de classes, sendo sempre muito mais eficiente que o SCI. Também se mostrou imune à distribuição de valores nas classes e à taxa de repetição desses valores. Desta forma, optamos por apresentar apenas um gráfico (Figura 7) representativo deste grupo de consultas pontuais.

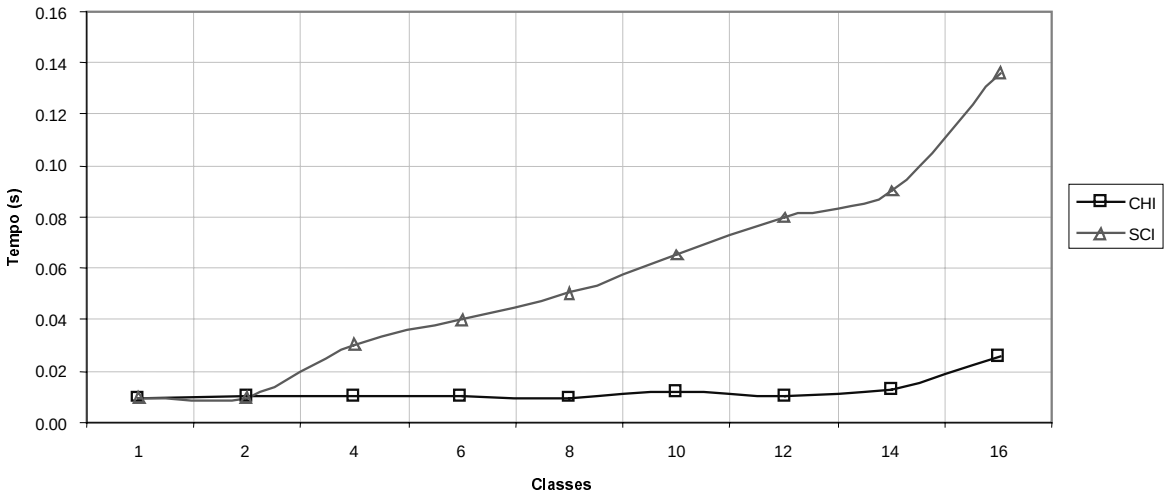

Figura 7 - Consulta pontual

4.2. Consultas por intervalo

As consultas por intervalo indicam a robustez da estrutura do índice na busca por uma faixa de valores. Foi nesse grupo de consultas que obtivemos resultados não previstos em análises qualitativas e quantitativas nos trabalhos relacionados. Nas simulações de Kim et al. [8] para consultas por intervalo, o índice CHI possui sempre um desempenho superior ao SCI. Já nas análises de Kilger e Moerkotte [7] o resultado é praticamente o oposto, ou seja, o desempenho do CHI é linear e bem inferior ao do SCI, até que o número de classes aumenta muito e ambos se tornam ineficientes. Esses resultados são a princípio contraditórios, entretanto, observamos experimentalmente que essas duas situações ocorrem de fato. Na realidade, esses trabalhos fizeram uma generalização de comportamento para consultas sobre faixas de valores que não se aplica.

Pudemos detectar em nossos experimentos a influência de dois parâmetros nesse grupo de consultas que interferem diretamente nos resultados. O primeiro parâmetro diz respeito à taxa de repetição de valores no índice. O segundo parâmetro que observamos está relacionado ao percentual da faixa de valores envolvido no intervalo da consulta. Situações extremas para a taxa de repetição de valores do índice explicam o conflito entre os resultados obtidos nos trabalhos [7, 8]. Enquanto que em Kim et al. [8] essa taxa era muito alta, da ordem de 200 e 600 repetições por valor, em Kilger e Moerkotte [7] essa taxa era igual a 1, ou seja, sem repetições. Do ponto de vista do percentual da faixa de valores, em Kim et al. [8] essa taxa não é explicitada nem são avaliadas variações, enquanto que em Kilger e Moerkotte [7] era fixa em dez.

Apresentamos então os resultados obtidos em consultas por intervalo sem repetições nos valores indexados (Figuras 8 e 9) e para intervalos com uma taxa de repetição igual a dez (Figuras 10 a 12).

4.2.1. Consultas por intervalo sem repetição de valores

Observa-se nesse grupo de consultas que quando a faixa de valores é muito baixa (0.1%) o índice CHI possui um desempenho superior ao SCI (Figura 8), já que esse comportamento é semelhante à consulta por igualdade.

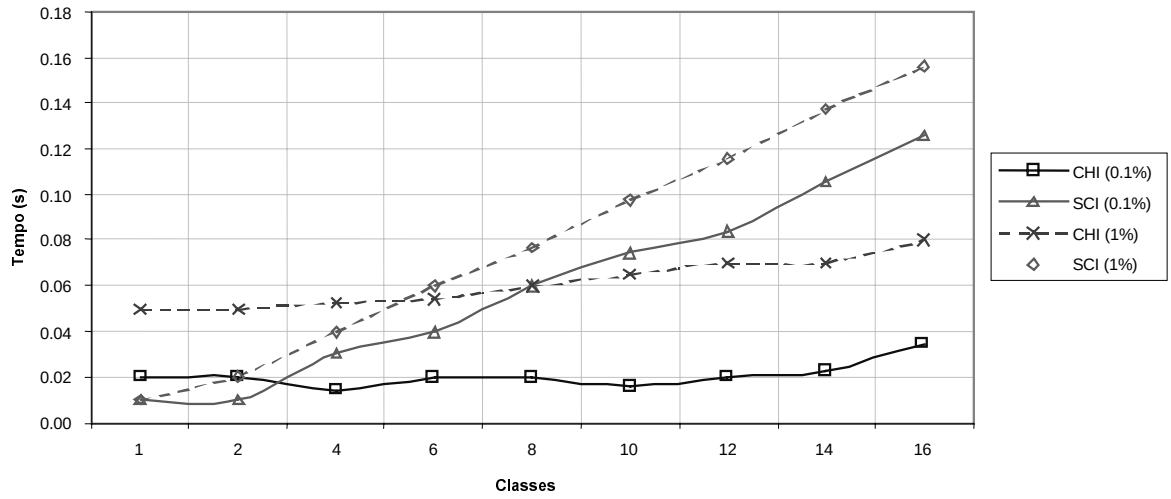

Figura 8 - Faixa pequena de valores; distribuição uniforme; taxa de repetição = 1

Entretanto, quando a faixa de valores aumenta um pouco (1%), para um número de classes menor ou igual a 6, o índice SCI supera o CHI. À medida que o número de classes vai aumentando, o SCI apresenta sua sensibilidade ao número de classes envolvido e degrada seu desempenho. Quando a faixa de valores sobe para 10% (Figura 9), há uma inversão de comportamento em relação à Figura 8. A situação fica totalmente desfavorável ao CHI. Isto decorre da sobrecarga causada pelo diretório de classes que requer uma ampla filtragem de objetos. Esta é a situação reportada em Kilger e Moerkotte [7].

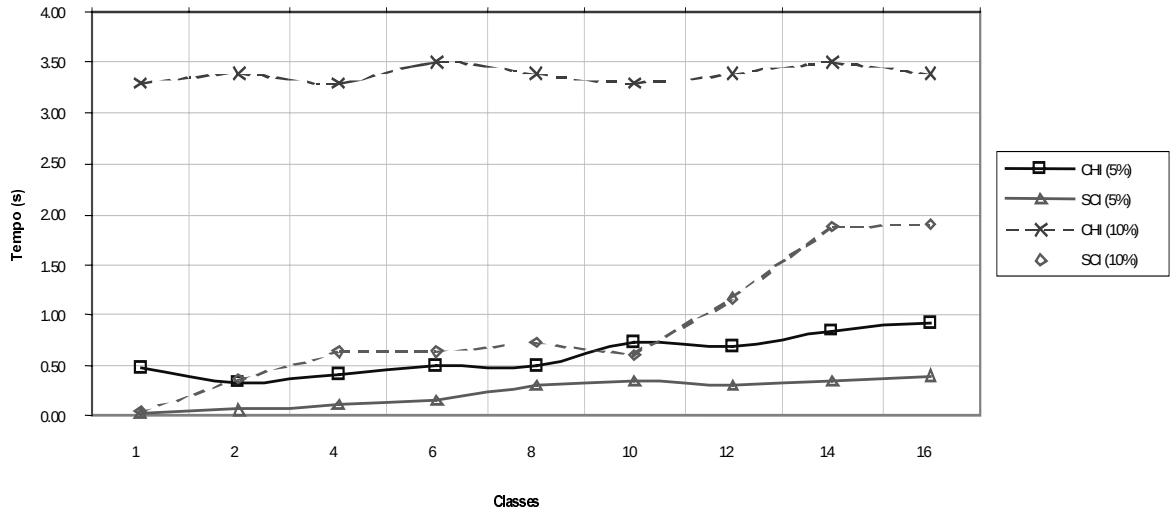

Figura 9 - Faixa média de valores; distribuição uniforme; taxa de repetição = 1

4.2.2. Consultas por intervalo com repetição de valores

Observa-se nesse grupo de consultas que quando a faixa de valores é muito baixa (0.1%) o índice CHI possui um desempenho superior ao SCI (Figura 10), já que esse comportamento

é semelhante à consulta por igualdade. Essa situação corresponde à analise de Kim et al. [8] para o desempenho de consultas à faixa de valores.

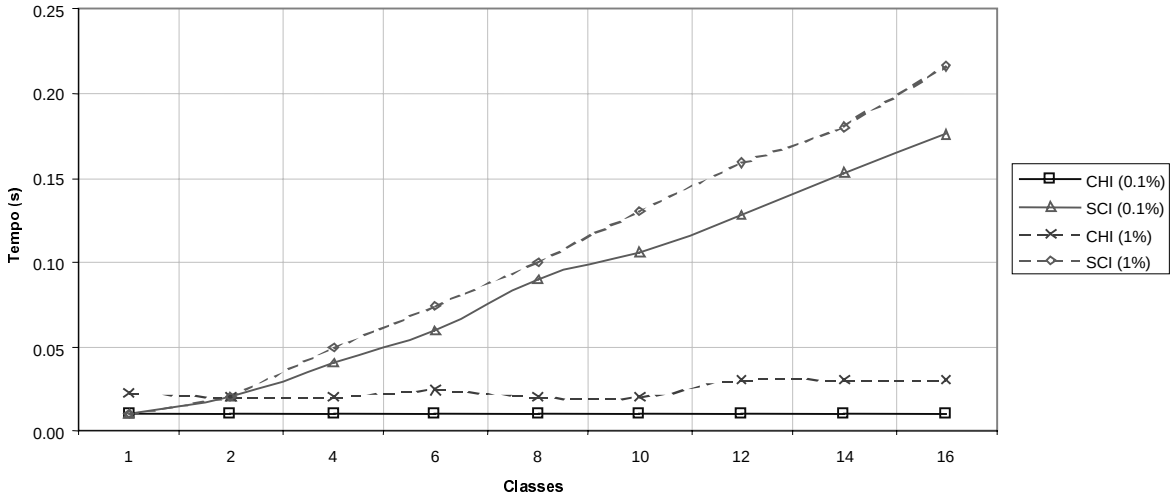

Figura 10 - Faixa pequena de valores; distribuição uniforme; taxa de repetição = 10

Quando a faixa de valores sobe para 10% (Figura 11), o índice CHI não apresenta o mesmo comportamento da consulta sem repetição (Figura 9). Isto decorre da diminuição do número de valores representados no índice devido às repetições. Conseqüentemente, a influência da sobrecarga do diretório de classes é amenizada. Neste caso o CHI apresenta um desempenho superior ao SCI a partir de consultas que envolvem quatro classes da hierarquia.

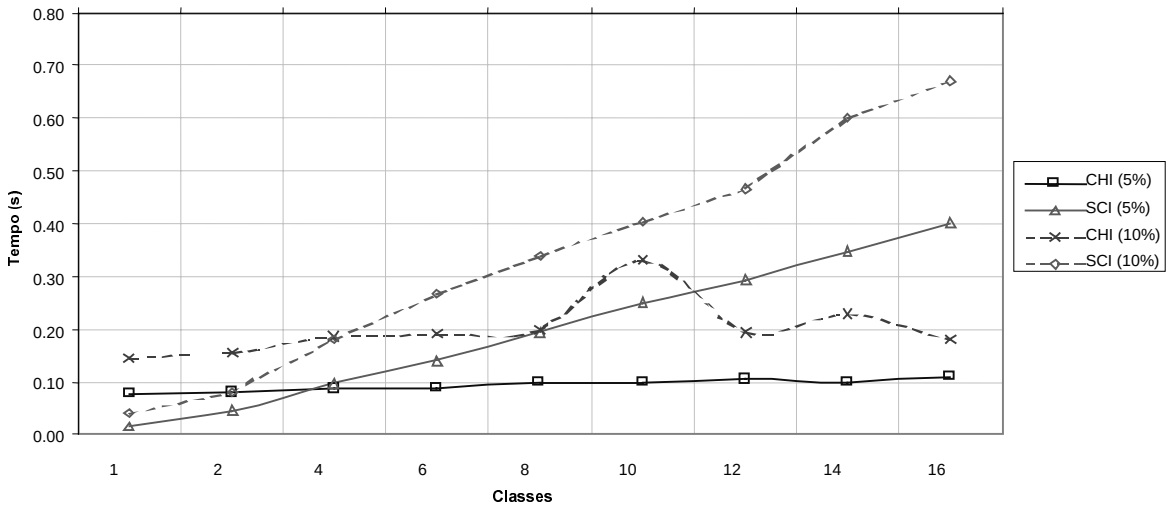

Figura 11 - Faixa média de valores; distribuição uniforme; taxa de repetição = 10

Em todas as situações anteriores de consultas por faixa de valores, o fator de distribuição de valores (uniforme ou não uniforme) não interfere no comportamento das consultas analisadas. Desta forma, não apresentamos os seus respectivos gráficos. Entretanto, quando a faixa de valores sobe para 10% com taxa de repetição igual a 10, o comportamento é sensível à distribuição de valores (Figura 12). O índice CHI mantém seu comportamento em relação à Figura 11. Já o SCI melhora seu desempenho, sendo superior ao CHI para um número de classes menor ou igual a 10. Isso decorre do fato de que a distribuição não uniforme fez com que o índice SCI varresse um número menor de nós folha.

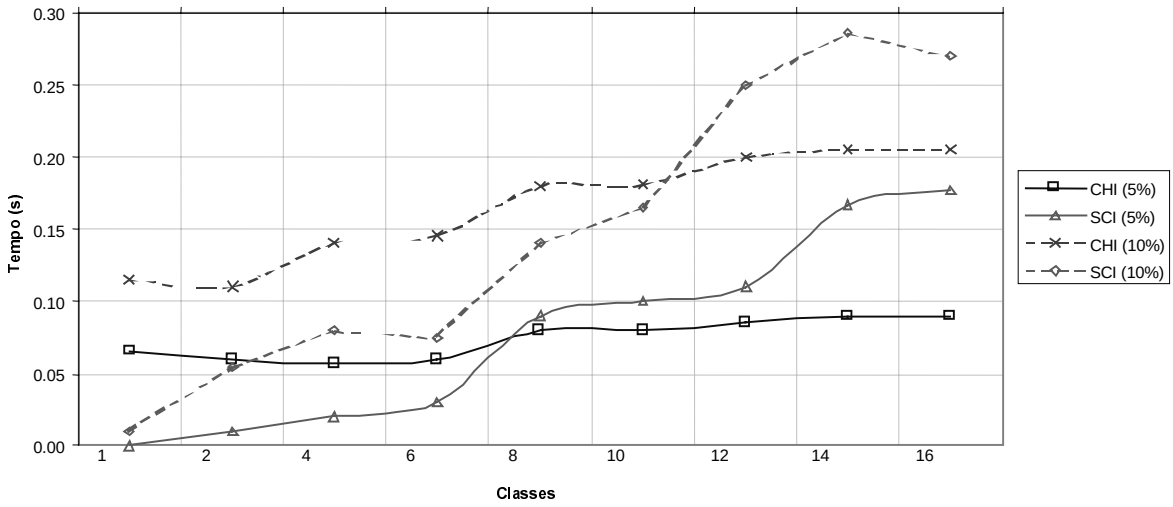

Figura 12 - Faixa média de valores; distribuição não-uniforme; taxa de repetição = 10

5. Conclusões e Trabalhos Futuros

A maioria das situações analisadas nos trabalhos anteriores se confirmaram experimentalmente. Entretanto, alguns comportamentos sugeridos como genéricos nem sempre foram confirmados.

Analisando a relação custo/benefício, podemos concluir que o CHI nem sempre apresenta os melhores resultados. Fica evidenciado, no entanto, que para consultas pontuais, independente de qualquer outro parâmetro, o índice CHI é sempre a melhor opção.

No caso de consultas envolvendo pequena parte da hierarquia, o SCI é a melhor escolha. Já para consultas sobre faixa de valores é necessária uma análise mais cuidadosa, pois o desempenho varia de acordo com o grau de repetição dos valores das chaves. Esses resultados não ficaram evidentes nem nas análises qualitativas nem nos experimentos da literatura. Isso ocorreu devido à simplificação quanto à variação dos parâmetros de configuração dos valores das chaves. Observou-se que para esse tipo de consulta os índices são sensíveis ao grau de repetição dos valores das chaves e à faixa de valores especificada na consulta.

Finalmente, sugere-se que os dois índices devem estar disponíveis num SGBDOO sendo o CHI fortemente indicado para chaves com consultas predominantemente pontuais. Nas demais situações sugere-se que seja adotado o SCI para hierarquias em torno de seis classes. Futuramente pretende-se estender a árvore-R disponível no GOA++ para avaliar a árvore-χ indicada para hierarquias com poucas atualizações.

6. Referências Bibliográficas

[1] Bertino, E. e Foscoli, P. 1995. *Index Organizations for Object-Oriented Database Systems*. IEEE Transactions on Knowledge and Data Engeneering , Vol. 7, N° 2, pp.193-209.

[2] Bertino, E. e Foscoli, P. 1997. *On Modeling Cost Functions for Object Oriented Databases*. IEEE Transactions on Knowledge and Data Engeneering , Vol. 9, N° 3, pp.500-508.

[3] Bertino, E. e Kim, W. 1989. *Indexing techniques for queries on nested objects*. IEEE Transactions on Knowledge and Data Engeneering , Vol. 1, N° 2, pp.196-214.

[4] Cattell, R., Barry, D., 1997. *Object Database Standard ODMG 2.0* - Morgan Kaufmann.

[5] Chan, C., Goh, C. e Ooi, B, 1997. *Indexing OODB Instances Based on Access Proximity*. IEEE Int. Conf. Data Engineering, Birminghan, Inglaterra.

[6] Kemper, A. e Moerkotte, G, 1995. *Physical Object Management in Modern Database Systems*. W. Kim, Addison-Wesley, pp.175-202.

[7] Kilger, C. e Moerkotte, G, 1994. *Indexing Multiple Sets. Int. Conf. VLDB'94*, Santiago, Chile, pp.175-202.

[8] Kim, W., Kim, K. e Dale, A, 1989. *Indexing techniques for object-oriented databases in Object Oriented Concepts, Databases, and Applications*, W.Kim e F. Lochovsky, Addison-Wesley.

[9] Lima, A.A. Mattoso, M.L.Q. Esperança, C. 1998. *Extensão de um SGBDOO com dados espaciais*. XXIV Conf. Latino Americana de Informática - CLEI'98 Equador.

[10] R.C.Mauro, G.Zimbrão, T.S.Brügger, F.O.Tavares, M.Duran, A.A.B.Lima, P.F.Pires, E.Bezerra, J.A.Soares, F.A.Baião, M.L.Q.Mattoso, G.Xexéo, 1997, *GOA++: Tecnologia, Implementação e Extensões aos Serviços de Gerência de Objetos*, XII Simpósio Brasileiro de Banco de Dados, Fortaleza, Brasil, pp.272-286.

[11] Mauro, R.C. Mattoso. M.L.Q. 1998. Integração de LPOO e BDOO: *Uma Experiência com JAVA e GOA++*, XIII Simpósio Brasileiro de Banco de Dados, SBC, Maringá, Brasil, pp.169-184.

[12] Mauro, R.C. Mattoso. M.L.Q. 1998. GOA++ e suas Ferramentas, *1ª. Mostra Brasileira de Software Acadêmico e Comercial do XIII Simpósio Brasileiro de Banco de Dados*, SBC, Maringá, Brasil, pp.83-88.

[13] Mueck e Polaschek, 1997. *The Multikey type Index for Persistent Object Sets*. IEEE Int. Conf. Data Engineering, Birminghan, Inglaterra.

[14] Ogasawara, E.S. Mattoso, M.L.Q., 1999. *Uma Análise de Índices em Bancos de Dados Orientados a Objetos*, Relatório Técnico do Programa de Sistemas da COPPE, ES-497/99, URL: http://www.cos.ufrj.br/~senna.

[15] Özsu M. e Blakeley, J.. 1995. *Query Processing in Object-Oriented Database Systems* in Modern Database Systems - The object Model, Interoperability and Beyond, Won Kim (ed.), Addison-Wesley, pp.146-174.

[16] Stehling, R. e Nascimento, M., 1998. *Métodos de Acesso em Bancos de Dados Orientados a Objetos, XIII Simpósio Brasileiro de Banco de Dados*. Maringá, Brasil, pp.369-384.

[17] Low, C. Ooi, B. and Lu, H., 1992. H-Tree - *A Dynamic Associative Search Index for OODB*. Proceedings of the 1992 SIGMOD, p. 134-143.

[18] Carey, M.J. DeWitt, D. Naughton, J. *The OO7 Benchmark*. Proceedings ACM SIGMOD International Conference on Management of Data, June 1993, pp.12-21.