IMPLEMENTAÇÃO E AVALIAÇÃO DE ÍNDICES PARA ORIENTAÇÃO A OBJETOS

Eduardo Soares Ogasawara

TESE SUBMETIDA AO CORPO DOCENTE DA COORDENAÇÃO DOS PROGRAMAS DE PÓS-GRADUAÇÃO DE ENGENHARIA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIAS EM ENGENHARIA DE SISTEMAS E COMPUTAÇÃO.

Aprovada por:

_____
Prof. Marta Lima de Queirós Mattoso, D.Sc.

_____
Prof. Claudia Bauzer Medeiros, Ph.D.

_____
Prof. Geraldo Bonorino Xexéo, D.Sc.

RIO DE JANEIRO, RJ - BRASIL

ABRIL DE 2000

OGASAWARA, EDUARDO SOARES
Implementação e avaliação de índices para orientação a objetos [Rio de Janeiro] 2000
IX, 87 p. 29,7 cm (COPPE/UFRJ, M.Sc., Engenharia de Sistemas e Computação, 2000)
Tese - Universidade Federal do Rio de Janeiro, COPPE
1. Índices para Sistemas de Banco de Dados Orientados a Objetos
I. COPPE/UFRJ II. Título (série)

*A meus pais,*

*por tudo o que representam em minha vida.*

# Agradecimentos

À Marta Mattoso, por sua cuidadosa orientação ao longo deste trabalho;

À Ingrid Nogueira Manhães, pelas repetidas e incansáveis correções do texto;

Ao Lúcio Leão Fialho, pelas orientações nas otimizações de algoritmos;

Ao Geraldo Zimbrão, por fornecer a biblioteca de gerência de memória;

Ao Renato Mauro, pela ajuda no entendimento do GOA++;

À Fernanda Baião e Angélica Ogasawara, pela revisão do *abstract*;

Ao Fred Ferreira dos Santos e Patrícia Leal, pela ajuda nas tarefas cotidianas;

À minha família, pelo apoio e compreensão da minha ausência neste período;

À CAPES, pela concessão da bolsa que me permitiu realizar o trabalho,

Agradeço.

Resumo da Tese apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Mestre em Ciências (M.Sc.)

# IMPLEMENTAÇÃO E AVALIAÇÃO DE ÍNDICES PARA ORIENTAÇÃO A OBJETOS

Eduardo Soares Ogasawara

Abril/2000

Orientadora: Marta Lima de Queirós Mattoso

Programa: Engenharia de Sistemas e Computação

Técnicas de indexação são recursos tipicamente utilizados por SGBDs que permitem definir o armazenamento físico das informações existentes na base de dados a fim de melhorar o desempenho das consultas a serem executadas. No modelo relacional este já é um problema resolvido na medida em que diversas técnicas de indexação já foram propostas, avaliadas e implementadas em sistemas comerciais. No entanto, com o advento do modelo orientado a objetos, características como a existência de atributos de coleção, atributos de referência e relacionamentos de herança entre classes incluem uma série de novas dificuldades que devem ser endereçadas a fim de se propor novas técnicas de indexação ou extensão das técnicas já existentes. Este trabalho apresenta uma análise qualitativa e quantitativa das principais técnicas de indexação para hierarquia de classes e para expressões de caminho na orientação a objetos. Para tanto, foi elaborado um ambiente propício para a avaliação destes índices estruturais. Através da utilização de benchmarks, pode-se analisar melhor os índices para expressões de caminho e para hierarquia de classes, correlacionando os resultados obtidos comumente encontrados na literatura por meio de simulação com os resultados alcançados experimentalmente nesta dissertação. De fato, foram verificadas as situações nas quais os resultados experimentais convergiram com os resultados obtidos através de simulações, assim como se detectou situações nas quais havia diferenças entre os dois métodos de análise. Esta dissertação contribui para o projeto físico de bases OO apresentando heurísticas sobre o uso destes índices.

Abstract of Thesis presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Master of Science (M.Sc.)

IMPLEMENTATION AND MEASUREMENTS OF OBJECT ORIENTED INDEXES

Eduardo Soares Ogasawara

April/2000

Advisor: Marta Lima de Queirós Mattoso

Department: Systems and Computer Engineering

Indexing techniques are commonly used by DBMS to define how the information in a database must be physically stored in the disk, in order to improve query execution performance. There has been a lot of research addressing indexes on relational DBMS. Many indexing techniques were proposed, evaluated and some of them are already implemented in commercial systems. However, when it comes to the object data model, new issues such as set attributes, reference attributes, and inheritance relationships must be addressed either by proposing new indexing techniques or by extending the existing ones. This work presents a qualitative and quantitative analysis of the main indexing techniques supported in OODBMS, including class hierarchy indexes and path expression indexes. In order to conduct this analysis process, we have implemented an evaluation framework to better understand the behavior of these structural indexes in different situations. Therefore, we present experimental results obtained through a benchmark implementation, and compare these results with the ones obtained through simulation in the literature. We identify situations where both results converge, and also explore the basic reasons for other situations where they do not, thus providing a better understanding of important issues and their influence on index structure on Object Oriented Systems.

Índice:

Índice de figuras:

Índice de tabelas:

## 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. 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 para 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 muitas linguagens de consulta, inclusive a OQL [2] (Object Query Language), 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 [3]. As técnicas de indexação estrutural podem ser classificadas em técnicas que apóiam expressões de caminho [4][5] e técnicas que apóiam consultas sobre hierarquia de classes [6]. Esta dissertação de mestrado concentra seu esforço na análise de índices estruturais.

Devido à riqueza do modelo Orientado a Objetos existem índices adequados às diversas estruturas semânticas do modelo como, por exemplo, a herança e os relacionamentos de composição e associação de objetos. Surgem na literatura diversas propostas de índices com análises qualitativas e quantitativas através de simulações. Entretanto, esses índices ainda são novidades nos sistemas Orientados a Objetos e Relacionais-Objetos e não são encontradas análises experimentais que avaliem o real impacto dessas estruturas em sistemas de cache e de entrada e saída. Assim surge a necessidade de analisar experimentalmente estes índices para que o analista do projeto físico do SGBDOO possa fazer a escolha mais adequada.

Existem diversas propostas para índices estruturais que apóiam consultas sobre hierarquia de classes. Os índices mais conhecidos na literatura são os índices apresentados por KIM e DALE [6]: o índice para cada classe (SCI) e o Índice para hierarquia de classes (CHI). Os demais e mais recentes índices existentes para hierarquia de classes visam explorar as limitações conceituais destes dois índices.

Apesar das diversas publicações sobre possíveis alternativas para estes índices, sente-se falta de meios para compará-los. Além disto, a própria comparação dos índices SCI e CHI presentes em [6] apresenta controvérsia como, por exemplo, no artigo de KILGER e MOERKOTTE [7] no qual os resultados experimentais obtidos eram praticamente o oposto dos resultados apresentados por KIM e DALE.

No que tange às expressões de caminho, sabe-se que ainda não existe muito conhecimento quanto à otimização de seus processamentos. Apesar do surgimento na literatura de diferentes algoritmos para a execução de consultas envolvendo expressões de caminho [8], não há ainda um consenso ou conhecimento com profundidade suficiente para definir a melhor forma de varredura numa expressão de caminho. Em função disto, a escolha do melhor índice para expressão de caminho torna-se complexa. São diversos os fatores que influenciam no desempenho do processamento da expressão de caminho. A combinação completa desses fatores torna essa tarefa exaustiva. Assim, o estudo das expressões de caminho traz uma motivação adicional, visto que os resultados apresentados até então não aparentam ser tão conclusivos ou definitivos quanto no caso da hierarquia de classes.

Este trabalho apresenta uma análise qualitativa e quantitativa experimental das principais técnicas de indexação estrutural para hierarquia de classes e para expressões de caminho.

A análise qualitativa auxiliou o processo de identificação dos índices mais adequados para um sistema de gerência de objetos. A partir dos resultados desta análise, optou-se por validar experimentalmente alguns índices representativos das categorias de índices hierárquicos e de expressões de caminho existentes na literatura.

Para tanto foi necessário elaborar um ambiente propício para a comparação conjunta das principais estruturas de indexação. Este ambiente tornou-se, portanto, um ponto chave para um melhor entendimento dos índices de expressões de caminho e hierarquia de classes, propiciando rápidas variações de parâmetros de indexação e uma progressiva e contínua análise dos resultados obtidos.

Desta forma, são apresentados resultados experimentais - sobre o GOA++ (Gerente de Objetos Armazenados da COPPE/UFRJ) - tanto para a implementação dos índices SCI (índice para cada classe) e CHI (índice para hierarquia de classe), referentes aos índices para hierarquia de classes, quanto para a implementação dos índices MX (multi-

índice), NX (índice aninhado) e PX (índice de caminho), relativos aos índices para expressões de caminho.

Além dos trabalhos de KIM e DALE [6] e BERTINO e KIM [5], trabalhos representativos para hierarquia de classes e expressões de caminho respectivamente, existem diversos trabalhos propondo o uso de outros mecanismos de indexação para índices estruturais. Apesar disto, a maioria destes trabalhos tinha a utilização de modelos de simulação como principal critério de avaliação dos índices. Nesta dissertação, as avaliações foram realizadas num sistema real, envolvendo o armazenamento de objetos persistentes. Neste sentido, foi meta deste trabalho examinar o desempenho desses índices frente à existência de um sistema de paginação de banco de dados, gerência de esquema e demais conjuntos existentes nos SGBDOOs. Assim, pode-se verificar as situações nas quais os resultados experimentais convergiram com os resultados obtidos através de simulações, bem como explorar as situações nas quais havia diferenças entre os dois métodos de análise.

Vários resultados da simulação foram observados nas experimentações, porém surgiram novas situações que revelaram sensibilidade dos índices quanto a elevações das taxas de repetição de chaves, bem como quanto à necessidade do uso de mecanismos para divisão de um índice em mais de uma página física.

Esta dissertação é organizada em quatro capítulos básicos, além desta introdução. O capítulo dois apresenta, de modo bastante didático, os principais índices estruturais conhecidos na literatura, procurando enfatizar as principais caraterísticas desses índices. Os índices são classificados e agrupados segundo suas aplicabilidades (hierarquia de classes, expressões de caminho e expressões de caminho com hierarquia de classes).

O capítulo três descreve o ambiente desenvolvido para avaliação experimental dos índices estruturais. O ambiente experimental é formado a partir de uma tríade que se constitui pelos algoritmos, pelas estruturas de dados e pelo mecanismo para medições (os denominados *benchmarks*). São apresentadas também neste capítulo otimizações realizadas na implementação deste ambiente.

O capítulo quatro expõe os resultados experimentais obtidos dos índices para hierarquia de classes e expressões de caminho no GOA++. Na apresentação deste capítulo há uma preocupação em correlacionar os resultados experimentais com os resultados obtidos a partir de simulações existentes na literatura, bem como apresentar

informações que resumam a aplicabilidade dos índices mediante variações de alguns parâmetros que influenciem os índices estruturais.

A título de conclusão, o capítulo cinco estabelece uma consolidação geral dos índices para hierarquia de classes e expressões de caminho, como também faz uma análise da utilização de um ambiente experimental para a avaliação dos índices estruturais.

## 2. Índices estruturais para Orientação a objetos

Os índices *estruturais* são os índices que se baseiam em valores de atributos de objetos. As técnicas de indexação estrutural podem ser classificadas em técnicas que apóiam expressões de caminho [4][5] e técnicas que apóiam consultas sobre hierarquia de classes [6]. Este capítulo concentra seu esforço na análise destes índices. São feitos um levantamento e uma avaliação qualitativa das principais propostas de estruturas de indexação para o modelo orientado a objetos. Para tanto, o capítulo é dividido em cinco seções básicas. A seção 2.1 apresenta as definições básicas para as seções seguintes. A seção 2.2 trata dos índices para hierarquia de classes e faz uma análise comparativa entre eles. Uma exposição de índices para expressão de caminho é vista na seção 2.3. Esta seção também avalia o comportamento destes índices face a diversas situações. Na seção 2.4, são apresentados dois índices híbridos - que combinam as características de índices estruturais com as de índices para expressão de caminho - juntamente com um resumo comparativo entre os índices abordados ao longo do texto. Finalmente, a seção 2.5 apresenta as considerações finais sobre a adequação dos índices face a diversas situações de uma consulta OO.

### 2.1. Estrutura Básica de Índices

Os índices apresentados neste trabalho têm uma estrutura básica de armazenamento em forma de árvore. Eventualmente alguns deles podem ser criados em forma de *hash*, mas pouca ênfase é realizada neste sentido. Portanto, a não ser que dito explicitamente ao longo do texto, os índices são armazenados na forma de árvore B+.

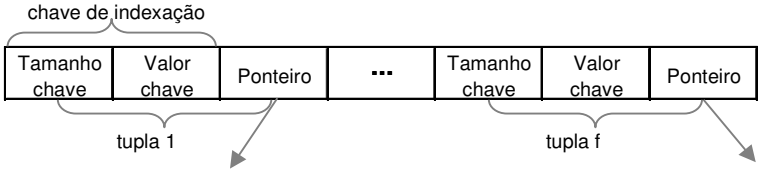



Figura 1 - Representação de um nó-intermediário

Quando armazenados na forma de árvore B+, o que diferencia um índice do outro são os nós-folhas. Os nós-intermediários 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.

### *2.1.1 Aplicação para Estudo de Caso*

Para facilitar o entendimento dos índices e auxiliar a comparação entre eles, considere a modelagem da Figura 2, que adota a representação gráfica de UML.

A aplicação tem quatro classes básicas: **Acervos, Livros, Periódicos** e **Bibliotecas**, sendo que **Livros** e **Periódicos** são especializações de **Acervos**. Neste modelo, **Acervos** representa uma classe abstrata, ou seja, não permite a existência de nenhuma instância da própria classe; há apenas a existência de membros provenientes de alguma de suas especializações como, por exemplo, **Livros**. Há, também, uma expressão de caminho entre **Bibliotecas** e **Acervos** representada por: **Biblioteca.ListaAcervos**.

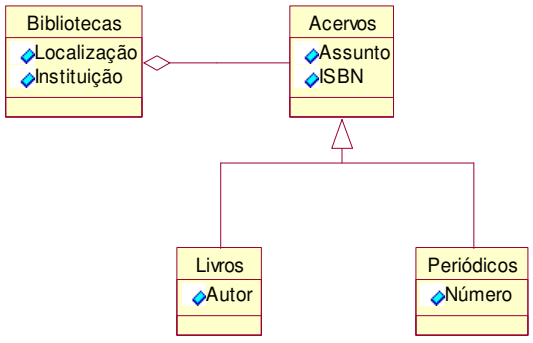



Figura 2 - Aplicação Básica

A partir deste modelo pode-se também definir um conjunto básico de instâncias, mostradas na Figura 3, e ainda diversas consultas, como as apresentadas a seguir:

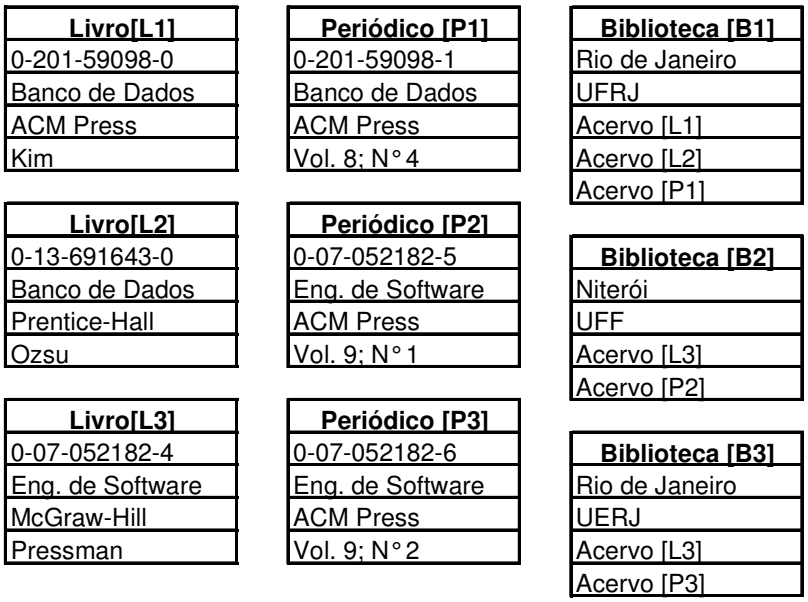



Figura 3 - Instâncias da Aplicação

[a] Selecionar todos os livros de "Banco de Dados".

Representa uma consulta envolvendo uma hierarquia de classes.

[b] Selecionar as bibliotecas do Rio de Janeiro que tenham acervos de "Banco de Dados".

Representa uma navegação sob uma expressão de caminho.

## 2.2. Índices para Hierarquia de Classes

Uma das diferenças entre sistemas relacionais e sistemas orientados a objetos é a possibilidade de abstração de generalização / especialização, 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 apresenta uma classe raiz, denominada *objeto*, de onde todas as outras classes são hierarquicamente descendentes. A partir deste conceito de especialização, 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 ser apenas a classe $D$ ou todas as subclasses. Esta semântica de instanciação de objetos força mudanças significativas no modo no qual os sistemas de banco de dados

usam índices [6]. A seguir serão expostos cinco índices diferentes para o tratamento de hierarquia de classes.

Os exemplos apresentados nesta seção envolvem a definição de um índice hierárquico sobre o atributo **Assunto** para toda subárvore formada a partir de **Acervos**.

Os índices propostos objetivam satisfazer as duas consultas abaixo:

**Consulta 1.** Selecionar todos os acervos cujo assunto seja "Banco de Dados".

A primeira consulta busca todos os membros de **Acervos** que satisfazem o predicado **Assunto = 'Banco de Dados'**.

**Consulta 2.** Selecionar todos os livros cujo assunto seja "Banco de Dados".

A segunda apenas busca as instâncias de **Livros** que satisfazem o predicado **Assunto = 'Banco de Dados'**.

### *2.2.1 Índice para cada Classe - SCI*

O Índice para cada Classe (*Single Class Index - SCI*) corresponde ao índice mais tradicional para hierarquia de classes. Esta estrutura 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 classe, o sistema deve manter um índice para cada classe na hierarquia e fazer as combinações correspondentes.

Como vemos na Figura 4, os nós-folhas do SCI são constituídos pelo tamanho do registro, tamanho da chave, valor da chave, ponteiro para página de sobrecarga (overflow), número de elementos que apresentam aquela chave e a lista de identificadores de objetos (OID).

Um registro no 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. [6] 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 serão armazenadas em páginas de sobrecarga. É para este

propósito que existe 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.

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

Figura 4 - Nó-folha do índice para cada classe (SCI) [6]

Assim, para se indexar o atributo **Assunto** para hierarquia formada por **Acervo** são necessários dois índices SCI: um para a classe **Livros** e o outro para a classe **Periódicos**. Não é necessário ter um terceiro índice SCI para **Acervos**, visto que trata-se de uma classe abstrata. A Figura 5 mostra um exemplo de SCI para a classe **Livros**, com valor de chave igual a **'Banco de Dados'**.

| Tam Reg | Tam chave | Chave | Sobrecarga | N° OIDs | Lista de OIDs |
|---|---|---|---|---|---|
| 74 | 50 | "Banco de Dados" | 0 | 2 | [L1][L2] |

Figura 5 -Exemplo de nó-folha do SCI para a classe Livros

Note que para satisfazer a Consulta 1 é necessário percorrer os dois índices para buscar os OIDs que satisfaçam o predicado **Assunto = 'Banco de Dados'**. Entretanto, a Consulta 2 é facilitada, visto que é necessário apenas buscar diretamente no índice específico da classe **Livros**.

### *2.2.2 Índice para Hierarquia de Classes - CHI*

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

Como mostrado na Figura 6, os nós-folhas de um índice para hierarquia de classes 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 particionados pelas classes presentes no diretório. O diretório de classes é decomposto em pares de identificador de classe por deslocamento no registro (indicando o começo da lista de OIDs daquela classe específica).

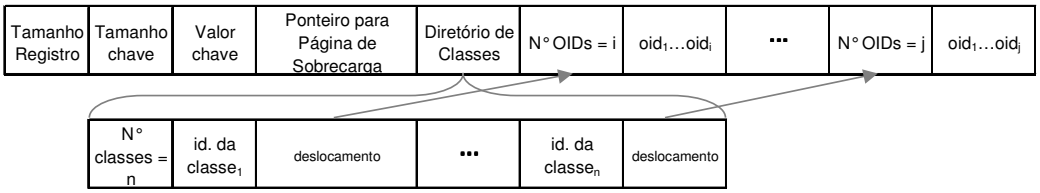



Figura 6 - Nó-folha do Índice para Hierarquia de Classes (CHI) [6]

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 $C$. No modelo da Figura 2 este índice pode ser usado para consultas que sejam diretas a membros de Acervos. A Figura 7 mostra um nó-folha para um índice CHI para o atributo **Assunto** da classe **Acervos**, com valor de chave igual a **'Banco de Dados'**.

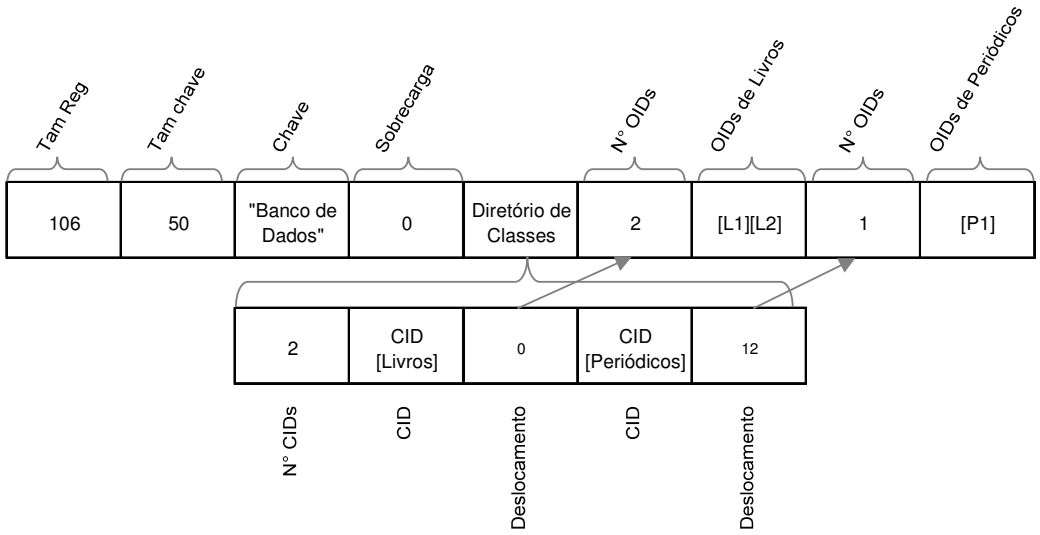



Figura 7 - Exemplo de nó-folha do CHI

Assim, para satisfazer a Consulta 1 é necessário percorrer apenas o CHI e buscar os OIDs que satisfaçam o predicado **Assunto = 'Banco de Dados'**. Para realizar a Consulta 2 é necessário também examinar o diretório de classes, identificando a entrada

que tenha o identificador da classe (CID) igual à da classe **Livros** e buscar apenas os OIDs específicos da classe.

Note que se o registro folha do índice fosse organizado simplesmente como um SCI, não tendo um diretório de classes, seria necessária uma busca exaustiva sobre a lista de OIDs do resultado da busca 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.

### *2.2.3 Árvore H Aninhada - Árvore H*

KEMPER e MOERKOTTE [4] apresentam um esquema de indexação sofisticado - desenvolvido por LOW et al., denominado Árvore H Aninhada (Árvore H) [9] - para combinar hierarquia de tipos num índice único. A idéia básica consiste em aninhar árvores B+, ou seja, aninhar árvores de subtipos dentro de uma árvore de supertipo.

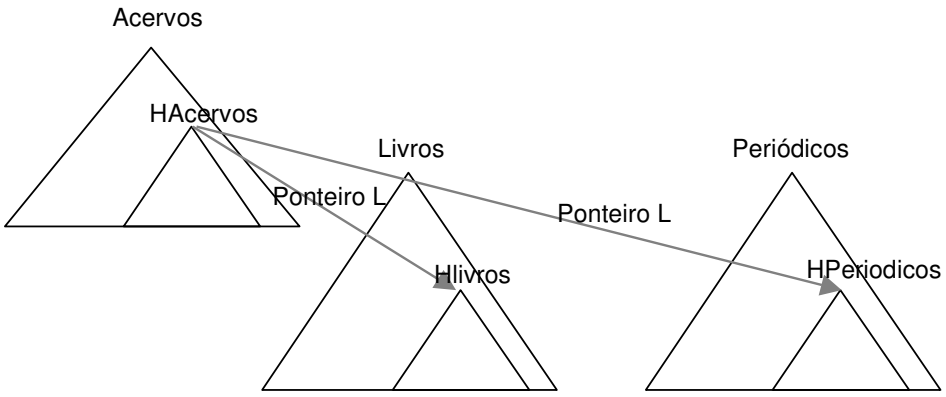



Figura 8 –Árvore H Aninhada [4]

Uma varredura ao longo de uma única classe pode ser feita diretamente a partir da **Árvore H** correspondente, simplesmente ignorando-se os ponteiros $L$ presentes. Uma consulta sobre toda uma hierarquia deve ser feita a partir da raiz da hierarquia (classe na qual a consulta foi iniciada), atravessando os ponteiros $L$ para as subclasses. Assim, para se realizar a Consulta 1, é necessário começar a busca a partir da árvore da classe ancestral **Acervos**, encontrar o valor de chave igual a **'Banco de Dados'** e percorrer todos os ponteiros **L** para buscar objetos de especializações. Entretanto, para se

realizar a Consulta 2, basta ir direto para a árvore da classe **Livros** e procurar a chave correspondente, não fazendo uso dos seus ponteiros $L$.

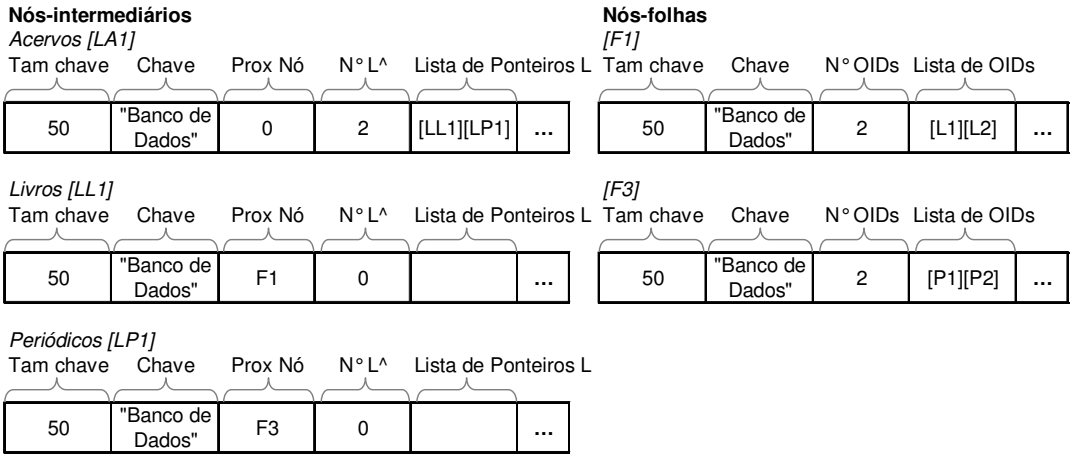



Figura 9 –Exemplo de Árvore H Aninhada

A manutenção de **Árvore H** impõe uma sobrecarga significativa em operações de atualização. Segundo KEMPER e MOERKOTTE [4], uma avaliação da penalidade para o uso desta estrutura ainda não foi investigada detalhadamente.

### *2.2.4 Índice de Árvore para Hierarquia de Classes - χTree*

CHAN, GOH, OOI [10] propuseram o Índice de Árvore χ para Hierarquia de Classes (χTree). A χTree combina as vantagens do CHI com o SCI, provendo um desempenho satisfatório tanto para as consultas voltadas para atributos como também para as consultas voltadas para classes. Para conseguir tal resultado, o índice considera tanto a dimensão das classes como também a dimensão dos dados para formar uma fragmentação do espaço de indexação. Para adicionar a grandeza de classes na dimensão de indexação é necessário definir uma linearização de tipos, ou seja, definir uma ordenação total entre as classes. Isto pode ser obtido através de uma busca pré-ordem sobre a hierarquia de classes a partir da classe ancestral (raiz da hierarquia). Como há a restrição de ordenação total, este índice não apóia sistemas que possuam herança múltipla.

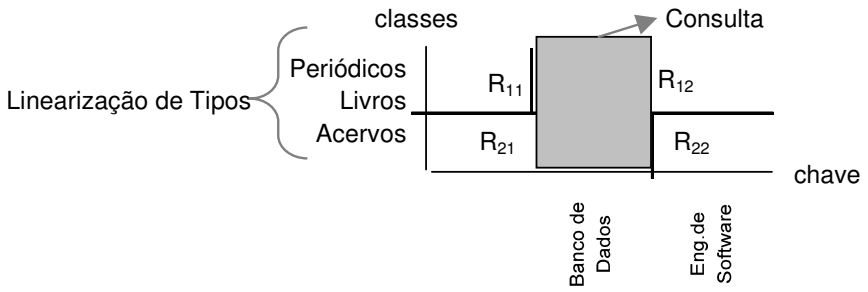



Figura 10 - χTree - [10]

A χTree é uma estrutura de indexação balanceada muito similar à R-Tree usada em buscas espaciais. Os registros armazenados são da forma (x, y) (equivalente a pontos num espaço bi-dimensional) onde x representa a classe do objeto e y representa o atributo indexado. Os nós-folhas são da forma *[CID, OID, chave]*, onde *CID* representa o identificador da classe do objeto, *OID* é o identificador do objeto e *chave* é o valor da chave de indexação. Os nós-intermediários são da forma *[R, P]*, onde *R* define uma caixa de contorno minimal e *P* é o ponteiro para o próximo nó.

Uma χTree contém um fator de utilização $f$ ($0 < f \leq 0.5$) e define $m$ como o número máximo de entradas que podem ficar contidas num nó-intermediário ou num nó-folha. Assim, uma χTree tem as seguintes propriedades:

1. Cada nó tem entre $f \bullet m$ e $m$ entradas, a não ser que seja um nó-folha.
2. A raiz da árvore tem pelo menos duas entradas, a não ser que também seja um nó-folha.
3. Se $X$ for um nó-intermediário com entrada $(R, P)$ e $Y$ um nó apontado por $P$, então:
   - Se $Y$ for um nó-folha, então $R$ é o menor retângulo, tal que contenha todos os pontos correspondentes aos objetos em $X$; caso contrário,
   - Se $Y$ for um nó-intermediário, então $R$ é o menor retângulo, tal que cubra todos os retângulos implicados pelas entradas em $Y$.
4. Todos os nós-folhas estão no mesmo nível.

A estratégia para realizar a divisão de nós-folhas sobrecarregados é medir a qualidade de divisão de um nó. Esta medição é definida como a proximidade de acesso referente a uma determinada divisão. Suponha $S_x$ o conjunto de registros de um nó X onde tenha ocorrido uma sobrecarga. Uma divisão, representada por [X|Y], distribui os registros de $S_x$ entre X e um novo nó Y, particionando o conjunto $S_x$ em dois subconjuntos: $S'_x$ e $S_y$. A proximidade de acesso de uma divisão [X|Y] é definida como a probabilidade de que

ambos os nós resultantes da divisão X e Y sejam acessados num mesmo acesso. Dadas duas possíveis divisões, a melhor escolha é aquela em que a probalidade de acesso da divisão é menor. Divisões podem ser horizontais (na dimensão de classes) ou verticais (na dimensão de atributos). O algoritmo para divisão utiliza lemas para determinar as divisões horizontais ótimas de acordo com a métrica de proximidade de acesso. Após determinadas também as divisões verticais ótimas, a proximidade de acesso é calculada para cada uma das divisões selecionadas. A divisão que resultar em menor proximidade de acesso é escolhida [11]. O critério para esta divisão pode ser melhor visto em [10].

Em termos das consultas 1 e 2 usadas como exemplo, o procedimento do algoritmo é o mesmo. A diferença básica ocorre durante a execução das consultas. Neste caso, a consulta 2 gera sempre um número menor de divisões de busca pela árvore do que a consulta 1. Isto se deve à linearização das classes.

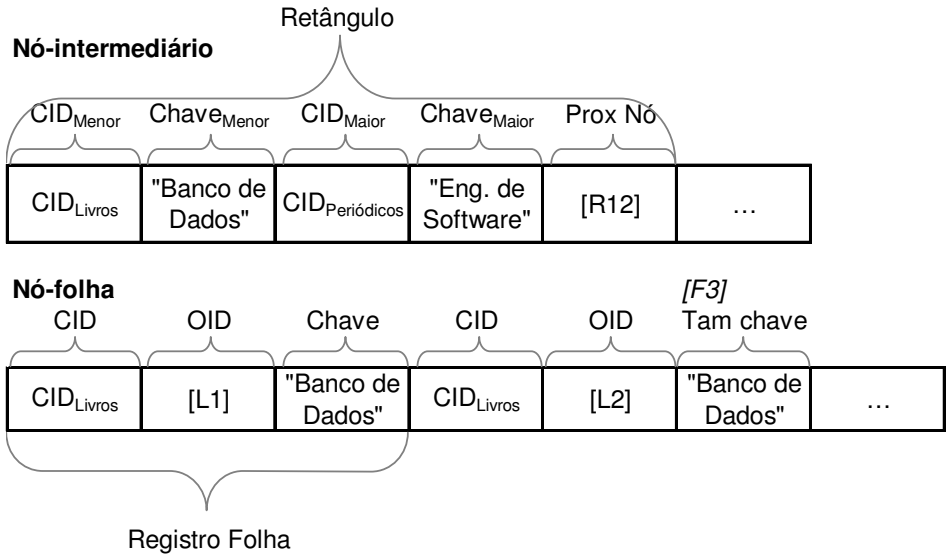



Figura 11 - Exemplo de χTree

### *2.2.5 Índice de Tipo Multichave - MT*

O Índice de Tipo Multichave (MT) [12] é um índice composto e representa uma estrutura alternativa para os tradicionais índices orientados em árvore B+. É um índice baseado numa *linearização ótima* de uma hierarquia de tipos. Uma linearização (ordenação total)

de uma hierarquia de tipos é ótima se e somente se cada subhierarquia corresponder a um intervalo do domínio da ordenação total resultante.

Conceitualmente, assim como o χTree, o índice MT é uma técnica intermediária, pois mescla características de indexação por chave (representado pelo CHI) com as de indexação por classes (representado por uma Árvore H).

A idéia central em torno do MT é incorporar a hierarquia de classes como atributo numa estrutura para busca de multi-atributos. Assim, a estrutura de classes é mapeada num dos domínios do índice, denominado domínio de classes. O resultado é que o índice passa a ser composto por $k + 1$ chaves, onde $k$ representa as $k$ propriedades do objeto a indexar de forma composta. Esta estrutura é montada numa hB-tree (cada nó-intermediário de uma hB-Tree é organizado como uma k-d tree). Esta estrutura para busca em multi-atributo pode ser interpretada como um conjunto no qual cada $k+1$-tupla é um elemento de um espaço geométrico de dimensão $k+1$. De acordo com esta interpretação, todas as tuplas são armazenadas num hiper-retângulo de dimensão $k+1$. Cada hiper-retângulo é fragmentado em hiper-retângulos cada vez menores, de acordo com o aumento do número de tuplas indexadas. Além disto, a hB-tree é uma estrutura que preserva a taxa de ocupação e a complexidade de estrutura de indexação para qualquer distribuição de dados. Os nós-folhas representam uma lista de OIDs, sendo que esta lista pode ser precedida ou não por diretório. Se $k$ for um valor alto, a ausência de um diretório praticamente não é percebida, devido à própria fragmentação imposta pela estrutura.

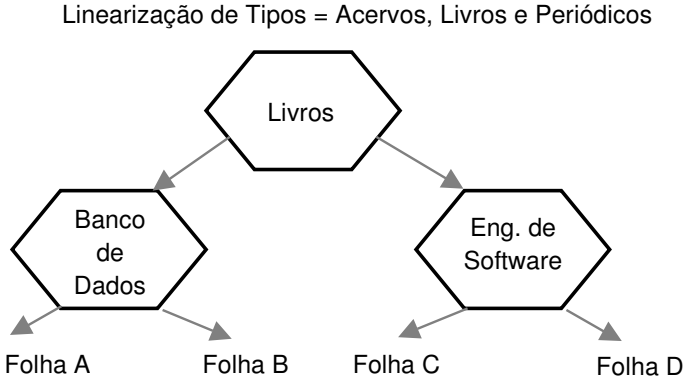



Figura 12 - Índice MT usando hB-Tree

Assim como na χTree, a Consulta 1 e a Consulta 2 usadas como exemplo são processadas da mesma forma. Novamente a diferença básica só é percebida durante a execução das consultas. Neste caso, a Consulta 2 gera sempre um número menor de divisões de busca pela árvore que a Consulta 1 devido a linearização das classes.

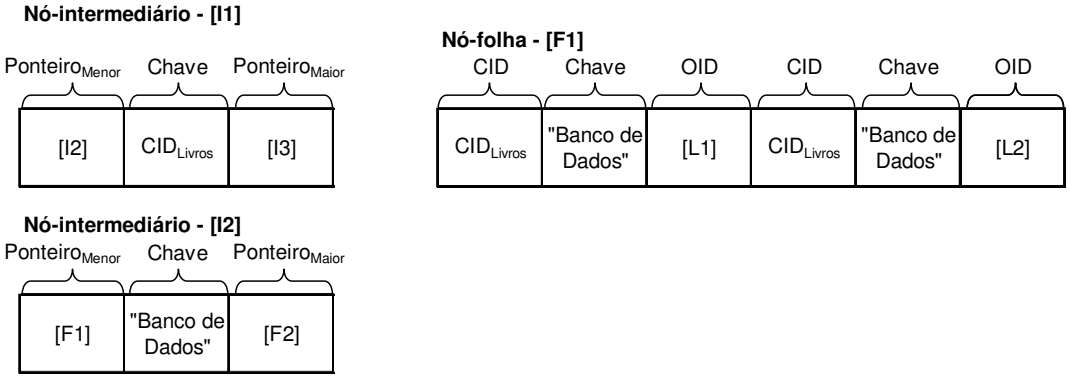



Figura 13 - Exemplo de Índice MT usando hB-Tree

### *2.2.6 Comparação*

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. [6], 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 pode ser menos eficiente que o SCI sobre uma classe C. Entretanto, numa distribuição conjunta, atravessar um CHI pode ser muito mais eficiente que atravessar os SCI referentes a cada hierarquia de classe [6]. Já as árvores H, χTree e MT não são sensíveis a esta distribuição de chaves.

Analisando a sensibilidade de armazenamento dos índices em função da distribuição de chaves, verifica-se que o SCI consome pouco espaço em distribuições disjuntas mas consome muito espaço em distribuições conjuntas, pois ele redunda muita informação dos nós-intermediários. Da mesma forma, as Árvore H também são sensíveis à distribuição de chaves, redundando muita informação nas distribuições conjuntas. Já o

CHI, χTree e MT não são sensíveis, em termos de armazenamento, à distribuição de chaves.

O SCI é o índice que menos as apóiam consultas sobre atributos que pertencem a diversas classes. Os demais índices, por razão de construção, tratam-nas bem. Já para consultas sobre classes específicas, o CHI é o que menos as apóiam. A Árvore H é uma estrutura voltada para realizar bem tanto as consultas orientadas a classes quanto a atributos. Já χTree e MT têm uma estrutura equilibrada, satisfazendo igualmente bem as consultas voltadas a classes e as voltadas a atributos. As consultas sobre faixa de domínio são pouco apoiadas pelo SCI e CHI. Já a χTree e MT as apóiam bem, devido à sua distribuição espacial.

O algoritmo para indexação de objetos em SCI e CHI é predominantemente o de uma árvore B+, tendo, assim, uma complexidade bem baixa. O MT também é simples por ter um comportamento de uma KD-Tree, embora restrito a uma linearização de tipos. A Árvore H é um pouco mais complexa, pois requer a manutenção da integridade dos ponteiros L que ligam as subárvores de hierarquia descendente. Já a χTree, além de requerer a linearização de tipos, tem também um algoritmo complexo para a realização da divisão de retângulos de sua R-Tree, que faz uso do conceito de admissibilidade e proximidade de acesso.

Em termos de atualizações e remoções é que as estruturas espaciais começam a deixar a desejar. Enquanto o SCI e CHI apóiam trivialmente estas operações, a Árvore H tem que constantemente rever a integridade dos ponteiros L, o que torna estas operações custosas. Já a χTree e MT não apóiam estas operações, o que impossibilita fazer uso destes índices na maioria das aplicações, pois mudanças em atributos de objetos são operações bastante comuns. Assim, uma mudança como a remoção de uma classe numa SCI e CHI quase não é sentida, mas numa Árvore H, χTree e MT esta operação requer uma reindexação total.

| **Índice** / **Critério** | **SCI** | **CHI** | **Árvore H** | **χTree** | **MT** |
|---|---|---|---|---|---|
| **Sensibilidade a distribuição de chaves - desempenho** | Sim | Sim | Não | Não | Não |
| **Sensibilidade a distribuição de chaves - armazenamento** | Sim | Não | Sim | Não | Não |
| **Apoio a consultas sobre atributo** | Baixo | Alto | Alto | Médio | Médio |
| **Apoio a consultas sobre classes** | Alto | Baixo | Alto | Médio | Médio |

| | | | | | |
|---|---|---|---|---|---|
| **Apoio a consultas Sobre faixa de domínio** | Baixo | Baixo | Médio | Alto | Alto |
| **Complexidade em Inserções** | Simples | Simples | Média | Alta | Simples |
| **Complexidade em Atualizações** | Simples | Simples | Complexa | Não apóia | Não apóia |
| **Complexidade em Remoções** | Simples | Simples | Complexa | Não apóia | Não apóia |
| **Apoio a Mudanças na Hierarquia de Classes** | Sim | Sim | Não | Não | Não |
| **Apoio a Estruturação Em Hash** | Sim | Sim | Não | Não | Não |

Tabela 1 - Tabela Comparativa entre os índices hierárquicos

## 2.3. Índices para Expressões de Caminho

As expressões de caminho são recursos poderosos na formulação de consultas. Representam navegações (travessias ou percursos) entre objetos seguindo relacionamentos de forma direta evitando o processamento de junções. A maior parte das técnicas de indexação que apóiam eficientemente a avaliação de expressões de caminho é baseada numa pré-computação de ligações funcionais entre objetos. Algumas destas técnicas requerem uma simples busca num único índice para avaliar uma expressão de caminho. Entretanto, os custos de atualizações podem ser bastante altos [1][4]. Outros métodos [5] requerem percorrer um número de índices igual ao número de classes a serem avaliadas numa expressão de caminho. Entretanto, os custos de atualizações destes métodos não são tão altos como no caso anterior.

Nesta parte do trabalho são expostos quatro dos principais índices existentes para expressões de caminho. Porém, para apresentá-los, faz-se uso de algumas definições básicas. Uma formalização mais rigorosa de tais definições pode ser encontrada em [1][4][5][13].

Dada uma expressão de caminho $P = C_1.A_1.A_2....A_n$, de tamanho $n$, uma instanciação sobre $P$ é uma seqüência $o_i$ de objetos $(o_1o_2...o_{n+1})$, tais que $o_1$ é um membro da classe $C_1$ e $o_i$ é o valor do atributo $A_{i-1}$ do objeto $o_{i-1}$, para $1 < i \leq n+1$. Uma instanciação é dita parcial se $o_1$ começar a partir de um dos objetos em $A_1.A_2....A_{n-1}$, ou seja, $(o_1o_2...o_j)$, onde $j < n+1$. Dada uma instanciação parcial $p = o_1o_2...o_j$ sobre $P$, $p$ é não redundante se não existir nenhuma instanciação $p' = o_1o_2...o_k$, $n+1 \geq k > j$, tal que $o_i = o_{k-j+i}$, $1 \leq i \leq j$ [5].

Uma travessia normal sobre uma expressão de caminho $C_1.A_1.A_2....A_n$ é percorrer uma instanciação $o_i...o_j$, para $1 \leq i < j \leq n+1$, seguindo as referências naturais dos objetos no

caminho. Uma travessia inversa sobre o caminho é percorrer esta mesma instanciação de $j$ para $i$ [5].

Uma expressão de caminho que não contenha nenhum atributo multivalorado é denominada linear. Se existir algum atributo multivalorado, então o caminho é orientado a conjunto [4].

O exemplo básico apresentado nesta seção envolve a definição de um índice para expressão de caminho sobre **Biblioteca.ListaAcervos.Assunto** da Figura 2. Os índices propostos objetivam satisfazer a seguinte consulta:

**Consulta 3.** Selecionar as bibliotecas que tenham acervos de Banco de Dados.

Esta consulta busca todas as **Bibliotecas** que tenham algum membro de **Acervos**, em **ListaAcervos**, com predicado **Assunto = "Banco de Dados"**.

A Consulta 3 pode ser classificada como uma travessia com instanciação total numa expressão de caminho orientada a conjunto.

### *2.3.1 Índice Aninhado - NX*

Dada uma expressão de caminho $P = C_1.A_1.A_2....A_n$, um Índice Aninhado (NX) sobre $P$ é definido como um conjunto de tuplas $[chave, oid_1...oid_i]$, onde $chave$ representa a chave de indexação, valor do atributo $A_n$ e $oid_1...oid_i$ é o conjunto de $i$ objetos alcançados numa travessia inversa a partir de $A_n$ [5]. Assim, um NX provê uma associação direta entre o objeto final e o objeto inicial ao longo de uma expressão.

Este tipo de índice não pode ser adotado em sistemas que não tenham referências inversas, a não ser que eles não apóiem atualizações ao longo da expressão.

Como vemos na Figura 14, os nós-folhas do NX são constituídos pelo tamanho do registro, tamanho da chave, valor da chave, número de instanciações inversas não redundantes, cujo objeto terminal tenha aquela chave, e pela lista ordenada de OIDs propriamente dita. Se, eventualmente, o tamanho do registro for maior do que a página de armazenamento, então um diretório é mantido em seu começo. O número de entradas no diretório é igual ao número de páginas necessárias para armazenar o registro. Cada entrada do diretório contém o endereço de uma das páginas onde é armazenado o registro e o maior OID armazenado naquela página, permitindo a fácil adição e remoção de objetos no registro.

| Tamanho Registro | Tamanho chave | Valor chave | N° OIDs = n | $oid_1 \ldots oid_n$ |
|---|---|---|---|---|

Figura 14 - Nó-folha de um Índice Aninhado - NX [5]

Nesta organização, pode-se definir um índice NX para a expressão de caminho **Biblioteca.ListaAcervos.Assunto**, como no modelo da Figura 2. A Figura 15 mostra um NX de tamanho dois para a chave **Assunto** com valor igual a **'Banco de Dados'**.

| Tam Reg | Tam chave | Chave | N° OIDs | Lista de OIDs |
|---|---|---|---|---|
| 70 | 50 | "Banco de Dados" | 2 | [B1][B2] |

Figura 15 - Exemplo de NX com valor de chave igual a "Banco de Dados"

Dada uma expressão de caminho $P = C_1.A_1.A_2....A_n$, tem-se que o NX gera uma associação direta entre o atributo aninhado $A_n$ e a classe $C_1$, requerendo apenas uma varredura num único índice. Conseqüentemente, o custo de avaliação de um predicado aninhado é o mesmo que se $A_n$ fosse atributo direto da classe $C_1$.

O custo para atualizações num NX é alto. Seja $o_i$ $(1 \leq i \leq n)$ um objeto pertencente a uma instanciação numa expressão de caminho $P = C_1.A_1.A_2....A_n$; $o_i$ tem $o_{i+1}$ como atributo na expressão. Suponha que $o_i$ atualize seu atributo $A_i$ de $o_{i+1}$ para $o'_{i+1}$. Para atualizar o índice, deve-se determinar $S^+$ (conjunto dos OIDs alcançados a partir de $o_{i+1}$ numa travessia normal[1]) e $S^-$ (conjunto de OIDs alcançados por $o'_{i+1}$ numa travessia normal). Se $S^+ = S^-$, então não é necessário nenhuma modificação no índice. Caso contrário, deve-se determinar o conjunto de objetos $R$ (objetos alcançados a partir de $o_i$ numa travessia inversa) que devem ser modificados no índice. Para cada chave existente em $S^+ - S^-$, deve-se remover os OIDs existentes em $R$ e, para cada chave existente em $S^- - S^+$, deve-se adicionar entradas respectivas para os OIDs existentes em $R$ [5]. A única exceção a esta regra ocorre quando há alterações no último atributo da expressão de

[1] Se a subexpressão formada a partir de $o_{i+1}$ for linear, então este conjunto só tem um elemento.

caminho. Neste caso, não são necessárias travessias diretas nem inversas, bastando uma simples mudança nos valores das chaves do índice.

### *2.3.2 Multi-Índice - MX*

Dado um caminho de tamanho $n$, uma indexação via Multi-Índice (MX) sobre o caminho é o conjunto de $n$ índices $MX$, da forma $MX_1..MX_n$, onde cada MX é um índice NX para caminhos de tamanho 1 [5]. A estrutura básica do MX é idêntica ao do NX, como representado na Figura 16.

| Tamanho Registro | Tamanho chave | Valor chave | N° OIDs = n | $oid_1 \ldots oid_n$ |
|---|---|---|---|---|

Figura 16 - Nó-folha de um Multi-Índice - MX [5]

Para satisfazer a expressão de caminho **Biblioteca.ListaAcervos.Assunto** deve-se definir dois índices MX. O primeiro é para satisfazer a subexpressão **Acervo.Assunto**. O segundo é para satisfazer a subexpressão **Biblioteca.ListaAcervos**. A Figura 17 mostra dois MXs: um com valor de chave igual a **'Banco de Dados'** para auxiliar a primeira subexpressão, e o outro com o valor de chave **L1** para a segunda subexpressão.

**MX para expressão Acervos.Assunto**

| Tam Reg | Tam chave | Chave | N° OIDs | Lista de OIDs |
|---|---|---|---|---|
| 78 | 50 | "Banco de Dados" | 4 | [L1][L2] [P1][P2] |

**MX para expressão Bibliotecas.ListaAcervos**

| Tam Reg | Tam chave | Chave | N° OIDs | Lista de OIDs |
|---|---|---|---|---|
| 20 | 4 | [L1] | 1 | [B1] |

Figura 17 - Exemplo de MX para expressão de caminho Biblioteca.ListaAcervos.Assunto

Dada uma expressão de caminho $P = C_1.A_1.A_2....A_n$, o MX requer a varredura de $(n-i+1)$ índices para avaliar um predicado sobre um atributo aninhado $A_n$ da classe $C_i$ $(1 \leq i \leq n)$. O índice $MX_n$ é varrido primeiramente e uma lista de OIDs é obtida. A seguir, é feita uma junção entre a lista de OIDs e o índice $MX_{n-1}$ e assim por diante, até que o $MX_i$ venha a ser acessado. Em particular, para avaliar um predicado que se refere à classe $C_1$, $n$ índices devem ser percorridos.

Uma das grandes vantagens do MX é permitir atualizações eficientes, visto que não é necessário nenhuma varredura direta ou inversa. Supondo que um objeto $o_i$, uma ínstância da classe $C_i$ sobre o caminho $P$, onde $1 \leq i \leq n$, seja atualizado trocando o valor do seu atributo $A_i$ de $o_{i+1}$ para $o'_{i+1}$, então, para atualizar o índice $MX_i$ deve-se remover o objeto $o_i$ do conjunto de OIDs associados com $o_{i+1}$ e adicionar este mesmo conjunto de OIDs associados a $o'_{i+1}$.

### *2.3.3 Índice de Caminho - PX*

Dada uma chave de indexação, um índice de caminho PX armazena todas as $i$ instanciações parciais não redundantes que terminam com aquela chave [5]. Assim, um PX pode ser representado por $[chave, (oid_{11}..oid_{1n}, oid_{i1}..oid_{in})]$, onde $chave$ novamente representa a chave de indexação, ou seja, o valor do atributo $A_n$ numa expressão de caminho $P = C_1.A_1.A_2....A_n$ e $(oid_{11}..oid_{1n}, oid_{i1}..oid_{in})$ representa um conjunto das i instanciações parciais de tamanho, no máximo, igual a $n$.

Como vemos na Figura 18, os nós-folhas do PX são constituídos pelo tamanho do registro, tamanho da chave, valor da chave, número de instanciações, cujo objeto terminal tenha aquela chave, e pela a lista de OIDs propriamente dita.

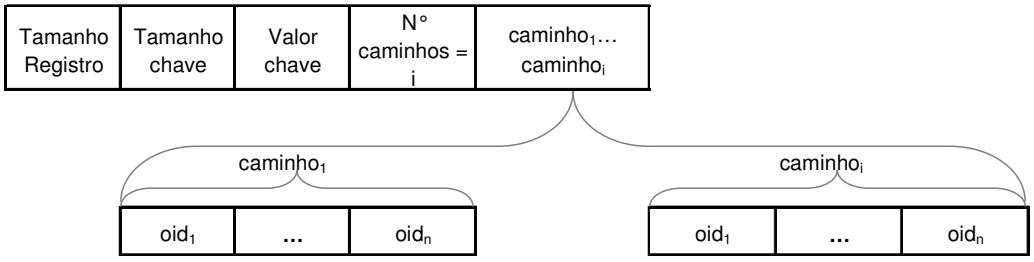



Figura 18 - Nó-folha de um Índice de Caminho - PX [5]

Note que um PX pode ser usado para avaliar qualquer predicado com uma expressão aninhada ao longo do caminho. Se um PX e um NX forem usados para expressões de

caminho de tamanho 1, eles apresentam a mesma estrutura, tornando-se predominantemente um índice igual aos usados em sistemas relacionais [5].

Nesta organização, pode-se definir um índice PX para a expressão de caminho **Biblioteca.ListaAcervos.Assunto**, como no modelo da Figura 2. A Figura 19 mostra um PX para expressões de caminho de tamanho dois para a chave **Assunto** com valor igual a **'Banco de Dados'**.

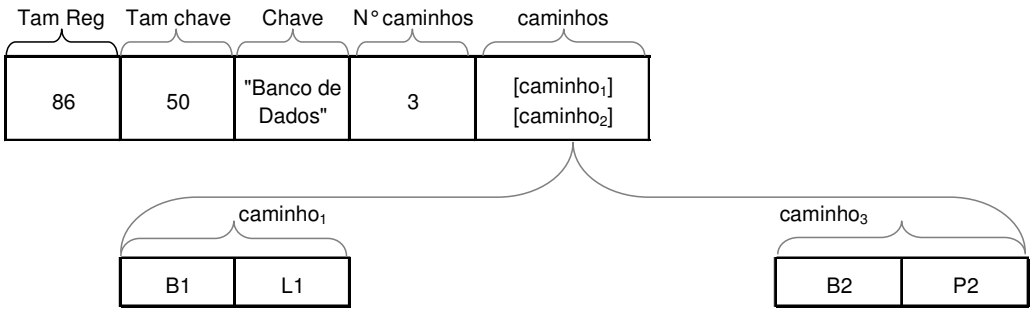



Figura 19 - Exemplo de um PX para valor de chave igual a "Banco de Dados"

O custo para atualizações de um PX também é alto. Dado o caminho $P = C_1.A_1.A_2....A_n$ e um objeto $o_i$ $(1 \leq i \leq n)$ que pertença a uma instanciação, tendo $o_{i+1}$ como atributo na expressão, se $o_i$ atualizar seu atributo $A_i$ de $o_{i+1}$ para $o'_{i+1}$, para atualizar o índice, deve-se determinar $S^+$ e $S^-$ (ver na página nº 19). Para cada chave existente em $S^-$ deve-se obter as expressões de caminho que passam por $o_i$ e $o_{i+1}$. Nestas expressões, antes de se remover estas entradas do índice, deve-se manter as partes esquerdas do caminho de $o_1$ a $o_i$ para que se possa compor novos registros no PX, concatenando-as com as subexpressões $o_{i+1}...o_n$ existentes em cada uma das entradas de $S^+$. Uma explicação bem detalhada com exemplos ilustrativos sobre esta operação pode ser vista em [5].

### *2.3.4 Relações de Apoio a Acesso - ASR*

Dada uma expressão de caminho $P = C_1.A_1.A_2....A_n$, tem-se que uma Relação de Apoio a Acesso (ASR) faz uso dos $n+1$ atributos $c.o_1o_2...o_n$ para compor sua estrutura. O ASR tem uma formação muito semelhante ao PX que, segundo KEMPER e MOERKOTTE [4], representa um caso particular de um ASR.

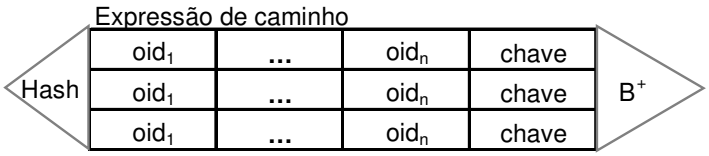



Figura 20 - Estrutura Básica de ASR - [4]

Cada ASR é redundantemente armazenado em duas estruturas de índices básicas. A primeira tem como chave o atributo mais à esquerda da expressão de caminho, enquanto a segunda se refere ao atributo mais à direita. Estas estruturas podem ser tanto Hash quanto Árvore $B^+$. A estrutura da esquerda apóia consultas de travessia para frente. A estrutura direita apóia travessias inversas sobre a expressão de caminho. O nó-folha de um ASR é idêntico ao de um PX, como mostrado na Figura 21.

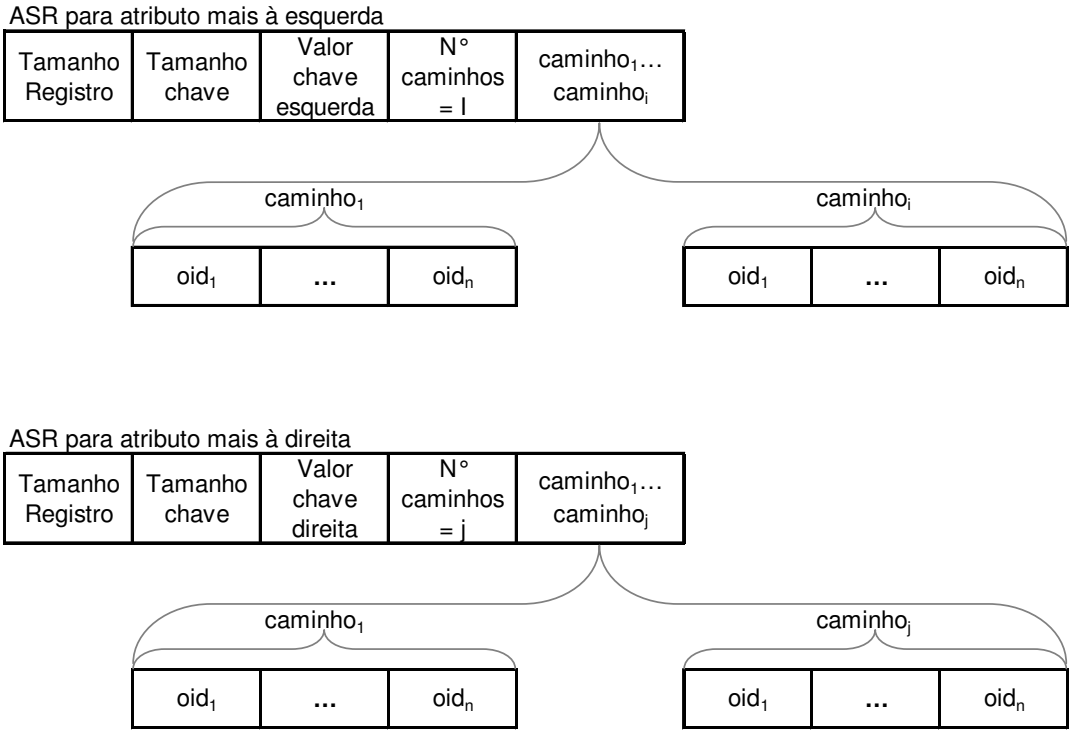



Figura 21 - Nó-folha de ASR

Este esquema de índice é bem estruturado para travessias sobre caminhos tanto da esquerda para a direita quanto da direita para a esquerda, mesmo se essas se dividirem sobre diversas estruturas ASR. Assim, pode-se escolher qual dos índices que compõem

o ASR deve ser usado para a consulta sobre **Biblioteca.ListaAcervos.Assunto**. A Figura 22 mostra um ASR para a expressão de caminho anterior.

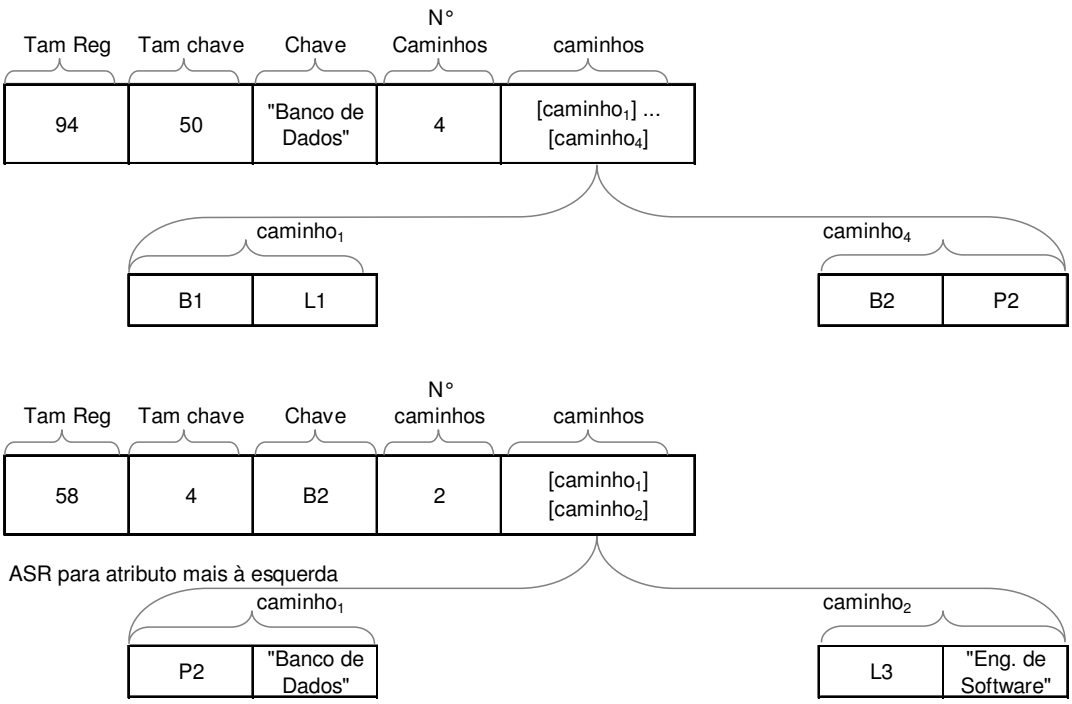



Figura 22 - Exemplo de um ASR com valor de chave esquerda igual a B2 e valor de chave direita igual a "Banco de Dados"

### *2.3.5 Comparação*

BERTINO e KIM [5] desenvolveram um modelo de custo para realizar comparações envolvendo tamanho de índices, custo de busca e custo de atualizações apresentando resultados com os índices NX, PX e MX. Este estudo foi posteriormente aprofundado por BERTINO e FOSCOLI [1].

Em termos de *tamanho de índices*, o NX - por praticamente criar uma associação direta entre o objeto final e o objeto inicial ao longo da expressão - é o que consome menos espaço. O ASR, por ser quase equivalente a dois PX, é o índice mais custoso em termos de tamanho. A comparação entre PX e MX é feita avaliando-se o grau de compartilhamento entre objetos. Algumas instanciações no caminho podem compartilhar algumas referências. Neste caso, os objetos referenciados são replicados em cada uma das matrizes que implementam a instanciação. Quando não há compartilhamento de

objetos, o PX tem menor custo do que o MX. Entretanto, quando o grau de compartilhamento vai crescendo, a mesma informação vai sendo replicada pelo PX, aumentando o custo de armazenamento em relação ao MX que, por sua vez, vai se aproximando do custo do NX.

A avaliação do *custo de buscas* é feita através da comparação do número de páginas trazidas à memória durante as consultas. As consultas são categorizadas em buscas por valor e buscas por faixa de domínio. Dois parâmetros muito importantes para esta comparação são $k_i$ (número médio de instâncias da classe $C_i$ que contêm o mesmo valor para seu atributo $A_i$) e $k_i \bullet k_{i+1,}$ que representa o fator de ligação ou grau de compartilhamento entre as classes *i* e *i+1*. Para o NX e o PX, o produto dos $k_i$ tem uma representação direta no *tamanho de cada registro* nos nós-folhas que, quando passa a ocupar mais de uma folha, tem um impacto grande em seu desempenho nas buscas. Já para o MX, o produto dos $k_i$ representa o número de OIDs que devem ser avaliados durante a varredura do i-ésimo índice $MX_i$. O custo do MX é independente do tamanho do registro e cresce linearmente com o produto do grau de compartilhamento dos objetos.

Nas *consultas sobre faixa de domínio*, o MX leva muita desvantagem, pois há muitas junções provindas de OIDs intermediários dos índices $MX_i$ parciais que não fazem parte do resultado final da consulta. O PX e ASR têm um resultado melhor que o MX, mas inferior ao NX. Isto se deve ao maior tamanho dos nós-folhas do PX e ASR quando comparados ao NX.

Em relação a *consultas em subexpressões* de caminho, o NX não pode ser usado eficientemente. O PX só pode ser usado em subexpressões do tipo $A_i....A_n$, $1 < i < n$. O ASR pode ser usado em subexpressões da forma $C_1....A_i$ ou $A_i....A_n$, onde $1 < i < n$. O MX pode ser usado em qualquer subexpressão de caminho. Dependendo do tamanho da subexpressão o MX pode ter um desempenho superior ao PX e ao ASR; especificamente quando a subexpressão for de tamanho um, a superioridade do MX é indiscutível devido ao tamanho menor dos registros nos nós-folhas e a não redundância de informações.

As atualizações dependem muito dos parâmetros $k_i$. Como no NX, no PX e no ASR o tamanho do registro é muito influenciado pelo produto dos $k_i$, a atualização pode ficar comprometida quando o tamanho do registro ultrapassar o tamanho da página. Além disto, no NX este parâmetro fornece uma indicação muita boa de quantos objetos

devem ser avaliados numa travessia inversa. O custo de atualização do ASR é praticamente o dobro do PX. A atualização no MX é bem simples. Neste caso, se o número de atualizações for intenso, então o uso do MX é recomendado.

O MX é o índice de melhor desempenho nas atualizações. Para expressões de caminho de tamanho dois o NX é apenas um pouco melhor do que o PX. Para caminhos de tamanho três, o desempenho depende de como são distribuídas as atualizações. Se as atualizações forem predominantemente na primeira e segunda classes, então o NX é melhor; caso contrário, o PX tem um custo menor do que o NX.

A Tabela 2 apresenta uma comparação do custo envolvendo as diversas situações de consulta além do impacto de algumas características físicas dos índices.

| **Índice** / **Critério** | **NX** | **MX** | **PX** | **ASR** |
|---|---|---|---|---|
| **Sensibilidade ao grau de compartilhamento – armazenamento** | Médio | Baixo | Alto | Muito Alto |
| **Sensibilidade ao grau de compartilhamento – desempenho** | Baixo | Alto | Médio | Médio |
| **Sensibilidade ao grau de tamanho do registro - desempenho** | Médio | Baixo | Alto | Alto |
| **Apoio a consultas Singulares** | Alto | Médio/ Baixo | Alto/Médio | Alto/Médio |
| **Apoio a consultas Por faixa** | Alto | Baixo | Médio | Médio |
| **Apoio a consultas sobre subexpressões** | Ausente | Alto | Médio/ Baixo | Médio |
| **Apoio a atualizações** | Baixo | Alto | Médio | Médio/ Baixo |

Tabela 2 - Tabela Comparativa da complexidade entre os índices para expressões de caminho

## 2.4. Índices Híbridos

Embora a existência de índices específicos para expressões de caminho e para hierarquia de classes tenha uma grande aplicabilidade quanto à otimização de consultas, algumas consultas envolvem tanto hierarquia de classes quanto expressões de caminho. Nestes tipos de consultas, o uso separado de índices para expressões de caminho e para hierarquia de tipos pode não ser satisfatório e nem sempre é possível aplicar seqüencialmente os dois tipos de índices. Pode-se tomar como exemplo a seguinte consulta baseada no modelo da Figura 2:

**Consulta 4.** Selecionar as bibliotecas do Rio de Janeiro que tenham livros de Banco de Dados.

Esta consulta busca todas as **Bibliotecas** cujo atributo **Localização** seja "Rio de Janeiro" e que tenham algum membro de **Livros**, em **ListaAcervos**, com predicado **Assunto = "Banco de Dados"**.

Suponha que exista um índice hierárquico sobre o atributo **Assunto** para toda subárvore formada a partir de Acervos e um índice para expressão de caminho sobre **Biblioteca.ListaAcervos.Assunto**. Note que esta consulta pode fazer uso dos índices não convencionais sob, pelo menos, três formas distintas:

1. Usando apenas o índice para hierarquia de classes, pode-se selecionar todos os livros de banco de dados. A seguir, se o sistema possuir referências inversas, deve-se selecionar todas as bibliotecas localizadas no Rio de Janeiro.

   Esta forma de consulta é vantajosa quando o fator de seletividade de livros de banco de dados for baixo, mas requer referências inversas e pode ser inadequada em expressões de caminho longas.

2. A segunda forma faz uso dos dois índices. Primeiro seleciona-se todos os livros de banco de dados via índice para hierarquia de classes e, aproveitando a lista de OIDs obtida, aplica-se o índice de expressão de caminho, caso o índice permita armazenar OIDs intermediários na expressão (como o PX e o MX), para selecionar as bibliotecas que tenham livros de banco de dados e verificar se elas apresentam localização no Rio de Janeiro.

   Esta forma de consulta é vantajosa quando o fator de seletividade de livros de banco de dados for muito menor do que os acervos de banco de dados e quando a cardinalidade de acervos banco de dados for superior a de bibliotecas, mas não é qualquer índice para expressão de caminho que pode ser usado. Particularmente o NX, que é o índice de mais rápido acesso em expressões de caminho, não pode ser adotado.

3. A terceira forma faz uso apenas do índice para expressão de caminho. Deve-se, a partir de todos os acervos de banco de dados, testar se eles são membros de livros, caso o índice permitir armazenar OIDs intermediários na expressão de caminho (como o PX e MX) e obter os OIDs das bibliotecas, verificando se elas apresentam localização no Rio de Janeiro.

   Esta forma é vantajosa quando o fator de seletividade de acervos de banco de dados é baixo, permitindo uma rápida checagem de classes dos OIDs. A cardinalidade de bibliotecas pode ser muito maior do que a de acervos. Novamente o NX não pode ser adotado nesta consulta.

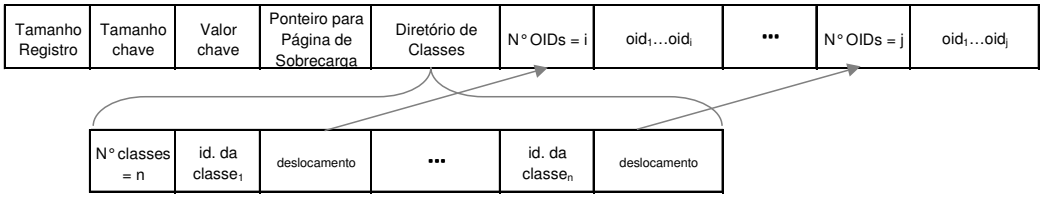
BERTINO e FOSCOLI [1] contribuem muito para o estudo de organização de índices para orientação a objetos ao apresentar duas técnicas básicas para melhor apoiar consultas que envolvam tanto expressões de caminho quanto hierarquia de classes. Para apresentá-las faz-se uso de algumas definições adicionais. Uma formalização mais rigorosa de tais definições pode ser encontrada em [1] e [13].

Seja $C$ uma classe; pode–se representar $C^*$ como o conjunto de classes descendentes de $C$. Dada uma expressão de caminho $P = C_1.A_1.A_2....A_n$, de tamanho $n$, tem-se que $classe(P) = C \cup \{C_i \mid C_i$ é domínio do atributo $A_{i-1}$ da classe $C_{i-1}$, $1 < i \leq n\}$ e $escopo(P) =$

$\bigcup_{C_i \in Classe(P)} C_i^*$. Dada uma classe $C'$ no *escopo(P)*, a posição de $C'$ é o número inteiro $i$, de modo que $C'$ pertença à hierarquia enraizada na classe $C_i$, onde $C_i$ pertence à *classe(P)*.

### *2.4.1 Multi-Índice de Herança - IMX*

O Multi-Índice de Herança (IMX) é uma organização que representa um melhoramento do Multi-Índice (MX) e consiste em usar um índice para hierarquia de classe (CHI) [6] para cada posição ao longo da expressão de caminho. Dada uma expressão de caminho $P = C_1.A_1.A_2....A_n$, existe um índice em cada classe $C_i$ em *classe(P)*. O índice em $C_i$ associa OIDs de membros de $C_i$ aos valores do atributo $A_i$ [1].




A estrutura básica do IMX é idêntica ao do CHI, como representado na Figura 23.

Figura 23 - Nó-folha de um IMX

Para satisfazer a expressão de caminho **Biblioteca.ListaAcervos.Assunto**, deve-se definir dois índices IMX. O primeiro é para satisfazer a subexpressão **Acervo.Assunto**. O segundo é para satisfazer a subexpressão **Biblioteca.ListaAcervos**. A Figura 24 mostra dois IMXs: um com valor de chave igual a **'Banco de Dados'** para auxiliar a primeira subexpressão, e o outro com a chave **L1** para a segunda subexpressão.

Salvo o melhor apoio a consultas que envolvam tanto hierarquia de classes quanto expressões de caminho, em termos de complexidade, o comportamento do IMX é muito semelhante ao MX, tendo as mesmas limitações básicas. O custo de busca num IMX é muito alto e requer a varredura dos diversos índices que o compõem. O custo de atualizações e remoções é baixo, por não requerer nenhuma travessia direta ou inversa.

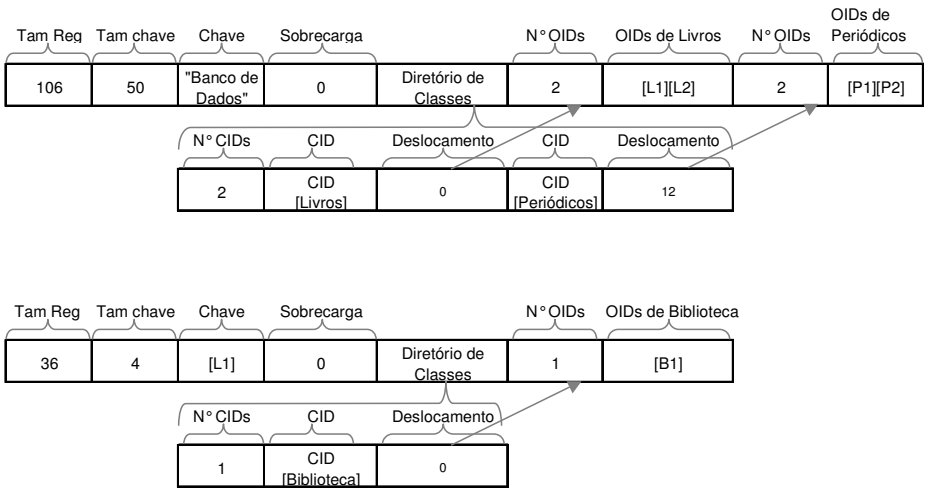



Figura 24 - Exemplo de IMX para expressão de caminho Biblioteca.ListaAcervos.Assunto

### *2.4.2 Índice Aninhado de Herança - NIX*

Em essência, o Índice Aninhado de Herança (NIX) permite que consultas contendo predicados sobre expressões de caminho e classes possam ser resolvidas numa varredura num único índice. Dada uma expressão de caminho $P = C_1.A_1.A_2....A_n$, um NIX associa ao valor $v$ do atributo $A_n$ OIDs de instâncias de classes no escopo de $P$, tendo $v$ como atributo aninhado [1].

Assim como o NX, o PX e o ASR, o NIX apóia a rápida varredura em expressões de caminho. Entretanto, o NIX, ao contrário das organizações anteriores, não requer travessias diretas ou inversas para atualizações, visto que algumas informações adicionais são armazenadas no índice. O nó-folha do NIX é denominado registro primário. Ele contém a forma exibida na Figura 25.

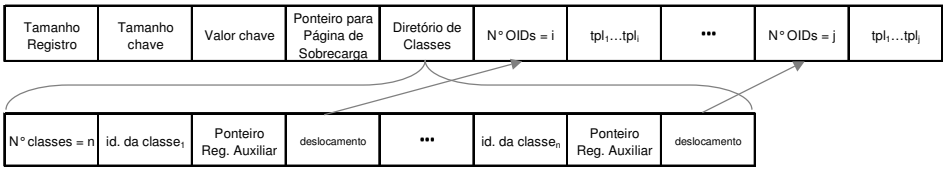



Figura 25 - Registro Primário de um NIX

Assim, o registro primário de um NIX é constituído pelo seu tamanho, tamanho da chave, valor da chave, ponteiro para página de sobrecarga, diretório de classes e, para

cada classe na hierarquia que apresente aquela chave, o número de elementos que tenham aquela chave e uma lista de tuplas *tpl*. Esta lista de tuplas é definida da seguinte forma:

1. Se $A_i$ é um atributo simples ou $i = n$, a tupla *tpl* é apenas formada por (*OID*), onde *OID* representa o OID de uma instância daquela classe que possui o atributo aninhado $A_n$.
2. Se $A_i$ é um atributo multivalorado, a tupla é formada pelos pares (*OID, N° Filhos*) onde *OID* representa o OID de uma instância daquela classe que possui o atributo aninhado $A_n$ e *N° Filhos* é o número de filhos daquele OID que possui o mesmo atributo aninhado $A_n$.

O diretório de classes contém um número de entradas igual ao número de classes tendo instâncias com aquele valor de chave como atributo indexado. Para cada classe C tem-se uma entrada no diretório que contém:

- Identificador de classe.
- Deslocamento a partir do registro primário onde a lista de tuplas *tpl* está armazenada.
- Ponteiro para registro auxiliar onde a lista de pais é armazenada para cada instância de $C_i$. Um registro auxiliar é definido para cada classe, exceto para a classe raiz do caminho e seus descendentes. Um registro auxiliar consiste numa seqüência de quádruplas da seguinte forma: OID, ponteiros para registros primários, número de pais do OID e a lista de pais propriamente dita.

Existem tantas quádruplas quanto o número de instâncias de $C_i$. Para cada objeto $o$, instância de $C_i$, a tupla contém o identificador do objeto $o$, o ponteiro para os $k$ registros primários, onde $k$ é o número de chaves no atributo aninhado $A_n$ de $o$, o número de pais de $o$ e a lista de OIDs dos pais de $o$.

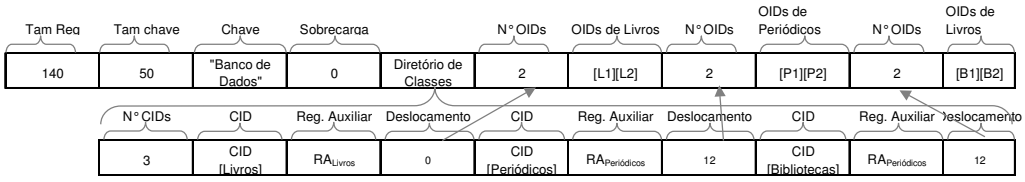



Figura 26 - Exemplo de registro primário num NIX para Biblioteca.ListaAcervos.Assunto

O registro auxiliar é indexado por uma árvore B+ baseado no OID do primeiro elemento de cada quádrupla. Conseqüentemente, o NIX tem uma organização que possui dois índices básicos. O primeiro índice, denominado índice primário, associa ao valor $v$ do atributo $A_n$ o conjunto de OIDs de instâncias de todas as classes que tenham $v$ como

atributo aninhado $A_n$. O segundo índice, denominado índice auxiliar, tem OIDs como chave indexada. Ele associa o OID do objeto $o$ à lista de OIDs dos pais de $o$. Os nós-folhas do índice primário contêm ponteiros para os nós-folhas do registro secundário e vice-versa. Conseqüentemente, o índice primário é invertido em respeito ao atributo $A_n$ e é usado em operações de busca. O índice secundário é invertido em respeito a OIDs de instâncias de todas as classes do escopo do caminho, exceto nas classes de primeira posição na expressão. Basicamente o índice secundário é usado para determinar todos os registros primários onde OIDs de uma determinada instância são armazenados para eficientemente realizar operações de remoção e inserção.

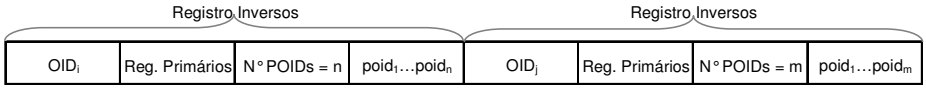



Figura 27 - Registro Auxiliar de um NIX

Assim, numa expressão de caminho $P = C_1.A_1.A_2....A_n$, tem-se que o NIX, assim como o NX, gera uma associação direita entre o atributo aninhado $A_n$ e a classe $C_1$, requerendo apenas uma única varredura num índice. Conseqüentemente, o custo de avaliação de um predicado aninhado é o mesmo que se $A_n$ fosse atributo direto da classe $C_1$, podendo-se ainda restringir a classe alvo da consulta.

Seja uma expressão de caminho $P = C_1.A_1.A_2....A_n$, e uma classe $C'$, pertencendo ao escopo de $P$, tendo uma posição $i$. Suponha que um objeto $o$, instância de $C'$ seja removido. O efeito geral da atualização do índice, é que o OID do objeto $o$ deve ser eliminado de todo registro primário que lhe faça referência, assim como todas as instâncias que o referenciam devem ser eliminadas dos registros primários, caso elas não possuam nenhum outro filho. A manutenção do índice secundário requer que as tuplas que fazem referências a $o$ (exceto para instâncias de classes raízes e suas subclasses) devem ser atualizadas, onde as referências a $o$ devem ser eliminadas da lista de pais, para cada um dos filhos de $o$, enquanto a tupla de $o$ deve ser removida do registro auxiliar relativo à classe $C'$.

Dada uma expressão de caminho $P = C_1.A_1.A_2....A_n$ e uma classe $C'$, pertencendo ao escopo de $P$, tendo uma posição $i$, suponha que um objeto $o$, instância de $C'$, deva ser adicionado. Para atualizar o índice deve-se determinar um conjunto SCH de atributos $A_i$ de $o$. O índice auxiliar deve ser varrido, usando como entrada cada OID em SCH, para inclusão de $o$ como pai dos elementos em SCH e obtenção de um conjunto de ponteiros

$S$ para os registros primários. O registro primário é varrido para cada OID em $S$. Uma busca no diretório de classes é avaliada para a classe $C'$ e o OID de $o$ é adicionado à lista de OIDs da classe. Se $A_i$ for multivalorado, o número de filhos é inicializado com o seu valor adequado. Uma nova tupla é adicionada ao registro auxiliar, com o OID de $o$ como chave.

### *2.4.3 Comparação*

Os parâmetros que mais influenciam o desempenho dos índices híbridos são os valores médios de cardinalidade de ligação entre classes para cada uma das posições na expressão de caminho. Em termos de armazenamento, na maioria dos casos, o NIX tem uma taxa de ocupação muito maior do que a do IMX. Entretanto, este quesito pode ser pormenorizado em função do barateamento dos dispositivos de armazenamento. O NIX tem um desempenho de consulta superior ao IMX, embora não tenha a mesma versatilidade para uso em subconsultas na expressão de caminho indexada. Já nas atualizações e remoções, o NIX apresenta custos altos, sendo influenciado diretamente pela topologia de referência entre os objetos existentes na expressão de caminho e pela classe alvo da consulta, tendo assim, um desempenho inferior ao do IMX, superando-o apenas em algumas situações particulares.

Em função das características de cada família de índices, seja ela pertencente aos índices hierárquicos ou aos índices para expressão de caminho, pode-se correlacionar para cada uma das estruturas de indexação seu respectivo adequado índice híbrido substituto. A Tabela 3 exibe esta correlação.

| Correlação / Índice | Família | IMX | NIX |
|---|---|---|---|
| **SCI** | Hierárquico | **X** | |
| **CHI** | Hierárquico | **X** | |
| **NX** | Expressão de Caminho | | **X** |
| **PX** | Expressão de Caminho | | **X** |
| **MX** | Expressão de Caminho | **X** | |

Tabela 3 - Tabela de substituição por similaridade entre os índices

## 2.5. Considerações Finais

As técnicas para indexação estrutural - índices que se baseiam em valores de atributos de objetos - podem ser classificadas em técnicas que apóiam expressões de caminho e técnicas que apóiam consultas sobre hierarquia de classes. Os índices para hierarquia de classes possibilitam o emprego da técnica de abstração de generalização / especialização, permitindo que objetos capturem a semântica do relacionamento "é um"

entre um par de classes. Os índices de hierarquia de classes podem ser orientados a valores de atributos (CHI), ou seja, um único índice para toda a hierarquia de classes, como também podem ser orientados a classes, um índice para cada uma das classes existentes na hierarquia (SCI) e ainda podem ser mistos, ou seja, tanto orientados a classes quanto orientados a valores de atributo (Árvore H, $\chi$-Tree, MT). Os índices espaciais são os que têm desempenho de consulta mais estável, mas quando a ocorrência de atualizações e remoções de objetos passa a ser uma variável relevante, a escolha do melhor índice passa a ficar restrita ao SCI e CHI.

A maior parte das técnicas de indexação que apóiam eficientemente a avaliação de expressões de caminho é baseada numa pré-computação de ligações funcionais entre objetos. Algumas destas técnicas requerem uma simples busca num índice para avaliar um predicado aninhado. Entretanto, os custos de atualizações podem ser bastante altos, como no NX e no PX. Outros métodos, como o MX, requerem percorrer um número de índices igual ao número de classes a serem avaliadas numa expressão de caminho. Entretanto, os custos de atualizações destes métodos não são tão altos como no caso anterior.

Para tentar tirar um melhor proveito das duas famílias básicas de índices estruturais foram desenvolvidos índices híbridos, como MIX e NIX, que melhor apóiam consultas que envolvam tanto expressões de caminho quanto hierarquia de classes. Comparando as demais famílias de índices com os índices híbridos, pode-se perceber que dependendo das características do SGBDOO, em geral, é sempre bom adotar algum índice híbrido, visto que pode-se tirar proveito de suas vantagens sem um aumento significativo na manutenção destes índices.

## 3. Ambiente de avaliação experimental de índices

A comparação analítica de índices é um mecanismo muito importante para a sua compreensão. Entretanto, esse tipo de comparação nem sempre engloba todos os índices como também nem sempre traz uma classificação completa, deixando alguns quesitos em aberto. Assim, torna-se necessário realizar comparações experimentais para validar e completar a comparação analítica dos índices. Para tanto, deve-se criar um ambiente de avaliação experimental de índices.

Além da elaboração de um projeto adequado para avaliação dos índices, deve-se definir um meio comum para comparação, que inclua um conjunto de parâmetros para análise,

bem como possibilitar a comparação não tendenciosa destes índices, explorando a variação dos parâmetros de análise.

## 3.1. Projeto de ambiente de avaliação experimental

No capítulo anterior foram apresentados os índices para hierarquia de classes e expressões de caminho. Alguns índices apresentavam uma estrutura de armazenamento individualizada, voltada para satisfazer as suas características intrínsecas.

Estrategicamente, deve-se fazer uma distinção entre o método de acesso e a estrutura de indexação adotada. Na seção 2.1 foi apresentado que a maioria dos índices utiliza a árvore B+ como método de acesso, entretanto, alguns deles poderiam ter sido implementados também via “hash”, enquanto os demais índices utilizam outros métodos de acesso. O mais importante de tudo é destacar que as páginas que armazenam o índice propriamente dito quase não têm relação com o método de acesso utilizado. Na verdade elas precisam apenas notificar para o método de acesso a inclusão de novas páginas provindas de divisão da página de armazenamento dos índices. Portanto, saber distinguir as páginas que armazenam os índices das páginas de método de acesso é uma decisão estratégica de projeto, que permite a criação de outros métodos de acesso às estruturas de indexação de modo transparente. É claro que alguns índices requerem métodos de acesso próprios, mas isto em nada invalida esta separação. Além disto esta separação é importante também para os índices que, durante a consulta, necessitam de mais de um método de acesso para sua realização visto que, além da simplificação da implementação, fornece a possibilidade de uso de diferentes métodos de acesso para o processamento da consulta como um todo.

Basicamente, os índices para serem manipulados numa inclusão, atualização ou exclusão devem ser alcançados por um método de acesso. O método de acesso aos índices consiste na exploração de suas páginas internas, os nós-intermediários. Estas páginas internas devem ser percorridas até que se alcance as páginas que contêm a estrutura de indexação propriamente dita, os chamados nós-folhas.

Para percorrer individualmente os nós-intermediários utiliza-se uma coleção de registros-intermediários. Estes registros-intermediários fornecem uma camada que associa as chaves usadas para indexação aos próximos nós-intermediários ou nós-folhas.

Os índices são manipulados pelos nós-folhas através da coleção de registros-folhas. Estrategicamente, estes registros são, na verdade, montados e desmontados a partir da classe auxiliar de modo a fornecer uma camada transparente para manipulação de registros longos, ou seja, registros que não cabem totalmente nos nós-folhas e que requerem nós-de-sobrecarga.

Os registros-folhas ligam as chaves aos índices propriamente ditos, encapsulando as chaves e os índices, permitindo a elaboração do algoritmo de divisão de nós e do algoritmo de sobrecarga de páginas totalmente isolados dos índices e chaves.

A Figura 28 apresenta o modelo UML [14] para a arquitetura de indexação implementada.

O modelo de objetos deve apoiar tanto índices para expressões de caminho como também índices para hierarquia de classes. Estes tipos de índices requerem tratamento diferenciado. Para tanto foram criadas classes para suas manipulações. Estas classes foram elaboradas ainda para uniformizar uma interface padrão para manipulação dos dados, independentemente do índice usado, tornando a seleção dos índices bastante simplificada.

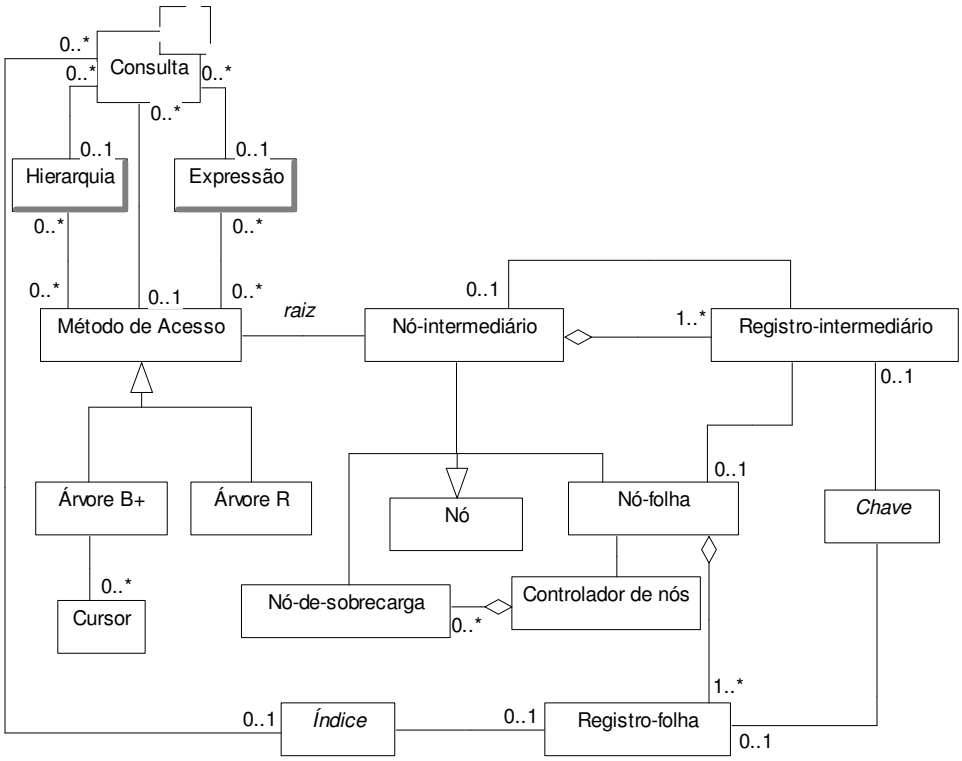



Figura 28 - Modelo de objetos dos índices

As consultas são requisitadas às classes de manipulação. Estas classes se encarregam da decomposição da consulta em subconsultas que são processadas segundo alguns métodos de acesso. A seguir o resultado das subconsultas é processado de acordo com o índice utilizado e posteriormente montado de modo a retornar uma consulta padronizada para o requisitante.

A classe índice é especializada em duas classes parametrizadas [14][15]: *Índice Simples* e *Índice Diretório* (Figura 29). Novamente, a decisão de parametrizar estas classes foi estratégica, visto que a partir delas pode-se, além de evitar redundância de código, enfatizar as características de que um *índice simples* é uma lista invertida em relação ao atributo chave, enquanto um *Índice Diretório* contém a semântica de um diretório, ou seja, para cada item $T_1$ no diretório existe uma lista do tipo $T_2$. Assim, os índices SCI, MX, PX2, PX3 e NX são especializações do *Índice Simples*, enquanto o CHI, IMX e NIX são especializações do *Índice Diretório*.

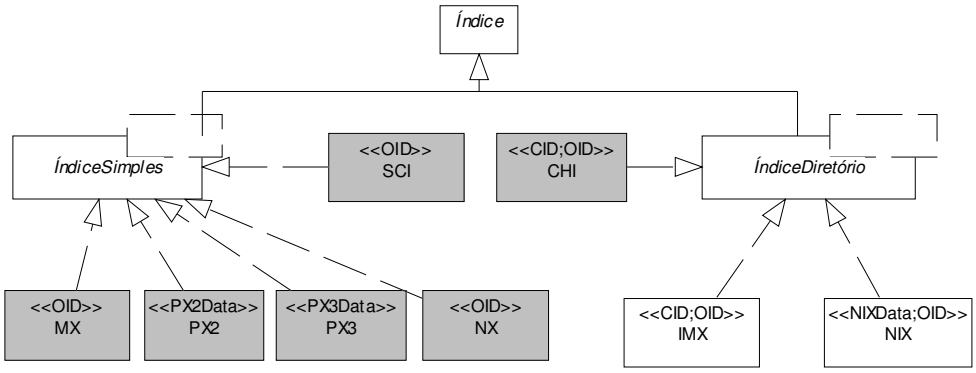



Figura 29 - Visão dos índices[2]

Usufruindo de todas as potencialidades do polimorfismo, as chaves são especializadas em todos os tipos primitivos, como alfanumérico, real, inteiro; complexos (OID) e até mesmo nebulosos (Fuzzy) [16]. Estas especializações (Figura 30) trazem toda a semântica necessária para a indexação dos dados, sem, no entanto, comprometer ou requerer qualquer modificação no sistema de indexação.

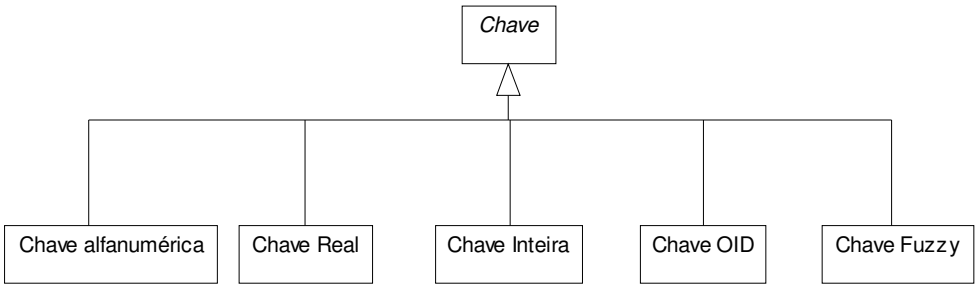



Figura 30 - Visão das chaves

A visão de consultas (Figura 31) permite uma maior compreensão do mecanismo de consulta. Cada consulta, seja ela hierárquica ou de expressão de caminho, pode ser quebrada em 1 a n subconsultas aos índices envolvidos no escopo da consulta. Pode-se tomar como exemplo uma requisição à hierarquia de classe. Independentemente do usuário ter solicitado uma consulta usando o SCI ou CHI, o resultado final da consulta deve ser tuplas da forma <chave, CID, OID>. Se o usuário tiver solicitado o processamento da consulta via SCI a classe de hierarquia de classes deve realizar as

[2] Os índices hachurados foram implementados e se constituíram enquanto objeto de estudo para resultados experimentais, conforme apresentado no próximo capítulo.

consultas em cada índice SCI envolvido no escopo da consulta para a seguir retornar uma tupla <chave, CID, OID>. Da mesma forma, se for requisitada uma consulta via CHI o sistema deve realizar uma única consulta ao índice CHI e uniformizar o resultado final da consulta para retornar <chave, CID, OID> de modo transparente.

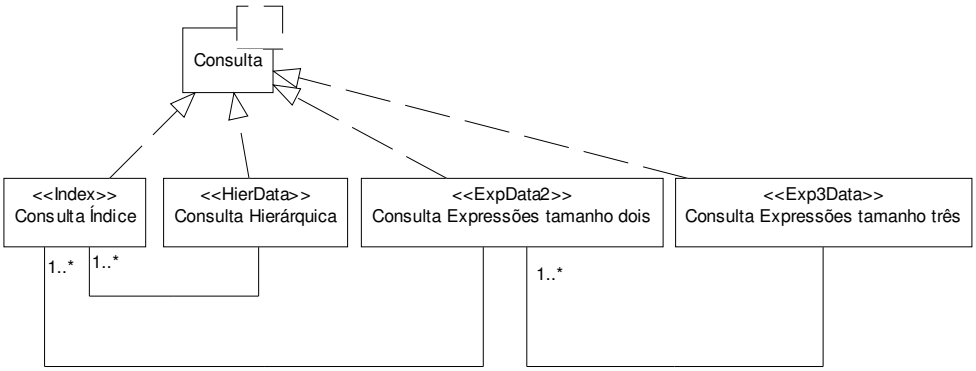



Figura 31 - Visão das consultas

## 3.2. Implementação do ambiente de indexação

A implementação do ambiente de indexação foi realizada no sistema GOA++[17]. 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 a OQL como linguagem de consulta. Apesar de não ser um sistema gerenciador de banco de dados completo (devido à 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 32 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 (representadas pela caixa IDX) foram

implementadas no núcleo do GOA++ fazendo uso dos serviços de armazenamento de objetos e gerência de *cache* e sendo usadas pelo processador de consultas.

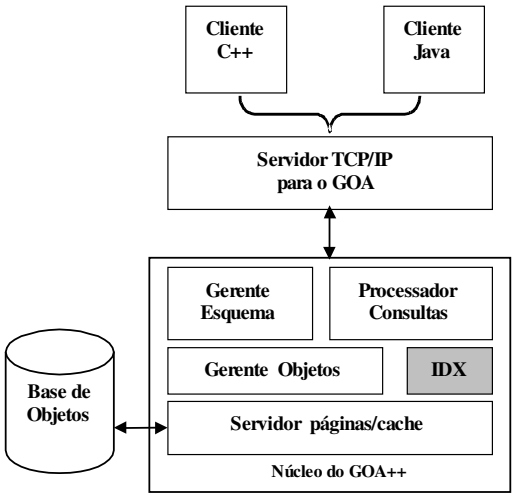



Figura 32 - Arquitetura GOA++

O sistema de indexação é um módulo acoplável, ou seja, possui uma estrutura bem encapsulada e detém uma interface bem definida que possibilita sua utilização em outros sistemas de armazenamento de objetos que possuam gerência de *cache* e sistema de consulta.

Uma completa descrição do GOA++ está fora do escopo deste trabalho e pode ser obtida em [17] ou em http://www.cos.ufrj.br/~bd/goa.

## 3.3. Otimizações de Implementação

As otimizações podem ser divididas em três tipos básicos: processamento em disco, consumo de memória principal e processamento em memória.

No processamento em disco, além do sistema de paginação de dados - recurso já existente no GOA++ e abordado em [17] - escolher a melhor forma de representação das informações, de modo que as estruturas de indexação não sobrecarreguem a própria informação armazenada, é fundamental. No caso do GOA++, o uso das estruturas de armazenamento não vinculadas ao próprio esquema foi de vital importância para diminuir a sobrecarga da própria estrutura de indexação e aumentar o seu desempenho.

O consumo de memória deve ser avaliado com o devido cuidado [18]. Falhas neste quesito anulam a vantagem técnica do uso de qualquer estrutura de acesso a dados em disco, visto que a ausência de memória física disponível numa máquina força o uso indevido de memória virtual [19]. Este uso indevido, além de prejudicar a execução do programa, invalida qualquer medição de desempenho na dimensão de tempo. Mesmo tendo memória física suficiente para a execução de consultas, deve-se eliminar todas as possíveis perdas de memória [18][20][21][22] de um sistema de indexação. Uma perda de memória consiste basicamente em alocar memória dinamicamente, mas não liberá-la após sua utilização. Esse erro, na maioria das vezes, aparece quando o programa é usado por muito tempo, onde permanece consumindo muita memória, até apresentar falha de execução pela sua falta. Deve-se sempre lembrar de desalocar toda memória que não seja mais útil, permitindo que outras funções e objetos possam usufruí-la.

Foi observado que o processamento em memória, embora não tenha sido crítico nos índices para hierarquia de classes, foi decisivo nos índices para expressões de caminho. Basicamente, todas as agregações [14][15] apresentadas no modelo de objetos foram implementadas através de uma classe de vetores. Inicialmente esta classe de vetores foi implementada através de inserções ordenadas sem fator de blocagem, ou seja, a cada inclusão de um novo registro alocava-se um novo bloco de tamanho n+1; todos os objetos eram então copiados e a inserção ordenada era executada. Esta implementação trouxe um impacto negativo muito grande para o desempenho de todos os índices para expressões de caminho. A inclusão de um fator de blocagem na alocação de memória e a troca do algoritmo de inserção ordenada por um de inserção não-ordenada, seguida de quick-sort [23] foi de vital importância para elevação do desempenho de todos os índices.

## 3.4. Benchmark OO7

O modelo de dados do benchmark OO7 (Figura 33) [24] foi adotado para realizar a análise de desempenho dos índices apoiados pelo GOA++. Uma das características importantes para a sua adoção é a existência tanto de hierarquia de classes como também de expressões de caminho. Como o modelo OO7 é semanticamente rico, ele permite explorar diferentes tipos de consultas de hierarquia de classes e de expressões de caminho. Além disto, ele pode ser usado para avaliar os índices híbridos, possibilitando 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 do modelo de dados, o benchmark OO7 define um conjunto de consultas e travessias. DeWitt et al [24] define como travessia a navegação ao longo de objetos através de métodos. A ênfase do benchmark OO7 foi na especificação de travessias, visto que no momento da elaboração do benchmark poucos sistemas orientados a objetos apresentavam uma linguagem de consulta declarativa como a OQL [2]. Assim, apesar da riqueza do modelo de objetos do OO7, poucas são as consultas padronizadas para exploração de expressões de caminho.

Uma outra alternativa para medições seria a execução de Bucky Object-Relational Benchmark [25]. O modelo de dados do Bucky Benchmark detém o seu foco nas consultas que envolvem hierarquia de classes, referências entre objetos, métodos e atributos e métodos de ADT [26]. Entretanto, o modelo Bucky, embora rico, também define poucas consultas que exploram as expressões de caminho. Seja qualquer um dos modelos adotados, deve-se adicionar consultas para melhor explorar as expressões de caminho. Assim, como o OO7 apresenta um largo uso em análise de desempenho no modelo OO [27][28][29][30][31] e, particularmente, no GOA++[32][33][34], optou-se por utilizá-lo nas medições.

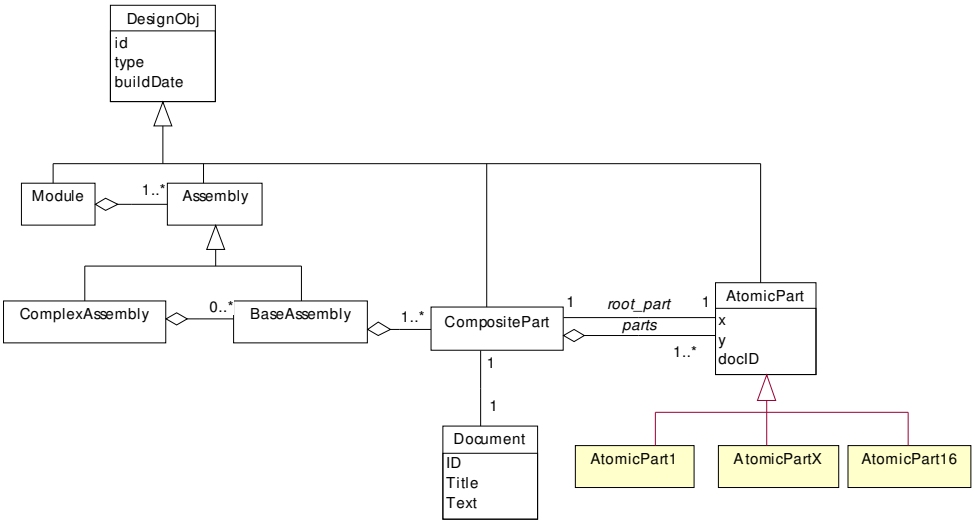



Figura 33 - Modelo UML do Benchmark OO7

Uma importante modificação realizada no modelo OO7 foi a inclusão de especializações na classe *Atomic Parts*. 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.

## 3.5. Configuração do Benchmark

Para fazer a avaliação experimental dos índices, optou-se por utilizar o sistema GOA++ e a base de dados OO7 em sua configuração média [24] desenvolvida para avaliar desempenho de SGBDOOs. Os testes foram realizados num AMD K6 III com 128MB de memória RAM usando o servidor GOA++, executando no Microsoft Windows NT 4.0 Server.

Tabela 4 - Configuração Média do OO7

| Classes | Cardinalidade |
|---|---|
| Assemblies | 3000 |
| CompositeParts | 9000 |
| Documents | 9000 |
| Atomic Parts | 180000 |

A base de dados gerada para o benchmark ocupou cerca de 300MB. Ao todo foram construídos duzentos índices ocupando 200MB, sendo sessenta e cinco índices para hierarquia de classes, cem índices para expressões de caminho de tamanho dois e trinta e cinco índices para expressões de caminho de tamanho três.

Os cem índices para expressões de caminho foram definidos de modo a propiciar diversas formas de navegação em diferentes configurações de cardinalidade e grau de compartilhamento de objetos. Já no que tange aos índices para hierarquia de classes, foram realizados testes com dezesseis especializações de classes sob a hierarquia de *Atomic Parts*, visando 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.

# 4. Resultados Experimentais

As técnicas de indexação estrutural podem ser classificadas em três grupos, a saber: (i) técnicas que apóiam consultas sobre hierarquia de classes [6], (ii) técnicas que apóiam expressões de caminho [5][7] e (iii) técnicas híbridas que apóiam consultas envolvendo expressões de caminho sobre hierarquia de classes [1][13]. Uma análise cuidadosa e extensa envolvendo índices das categorias (i) e (iii) foi apresentada por Stehling e

Nascimento [11] evidenciando os aspectos positivos e negativos dos principais índices dessas categorias.

Foi feita uma análise qualitativa detalhada em Ogasawara e Mattoso [35] comparando os três grupos de técnicas de indexação estrutural no sentido de avaliar as técnicas mais adequadas ao Sistema de Gerência de Objetos GOA++ [17][36][37]. Após esta análise, optou-se por validar experimentalmente alguns índices representativos das três categorias existentes. Em particular, foram implementados índices derivados da estrutura de árvore B+ (CHI e SCI para hierarquia de classes e MX, PX e NX para expressões de caminho). Embora os índices com origem em estruturas espaciais [11][35] 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.

A partir de observações gerais obtidas durante os resultados das medições experimentais, este capítulo analisa os índices para hierarquia de classes e também os índices para expressões de caminho de tamanhos dois e três.

Este trabalho contribui analisando a relação custo/benefício, a partir do benchmark OO7, entre o uso dos índices SCI e CHI para hierarquia de classes e MX, PX e NX para expressões de caminho, apresentando inclusive resultados não observados em trabalhos da literatura. Esses estudos de desempenho são relevantes não só para o otimizador de consultas do SGBD, como também para o projeto físico de uma aplicação orientada a objetos.

## 4.1. Observações gerais sobre a análise experimental

São explicitados a seguir os parâmetros medidos experimentalmente e utilizados na análise dos índices de hierarquia de classes e de expressões de caminho.

### *4.1.1 Taxa de acertos do* cache

Durante a comparação dos índices, em geral, deve ser feita uma ressalva na taxa de acertos do *cache*. A taxa de acertos do *cache* fornece dois indicadores importantes. O primeiro é o grau de confiabilidade das medições no SGBD. Basicamente, numa medição não tendenciosa, os índices não podem apresentar páginas de índice no *cache* do GOA++ para não comprometer os resultados de tempo durante as medições.

O segundo indicador é observado nas consultas por faixa de valores. No caso das expressões de caminho, o índice MX requer a execução de várias junções intermediárias. Estas junções forçam repetidas varreduras em nós-intermediários do MX. Neste caso, a taxa de acertos do *cache* fornece o grau de eficiência do algoritmo de junções intermediárias.

Excetuando-se o índice MX, foi observado que durante as medições realizadas a taxa de acertos do *cache* de todos os índices foi igual a zero. Nas consultas pontuais e nas subconsultas sob a expressão de caminho, a taxa de acertos do *cache* do MX foi baixa, revalidando a confiabilidade das medições (primeiro indicador). Já nas consultas por faixa de valores na expressão de caminho como um todo, a taxa de acertos do *cache* foi alta, próxima a 80% em consultas por faixa de valores igual a 5%, indicando um eficiente processamento de junções.

### *4.1.2 Páginas de sobrecarga e taxa de ocupação de páginas*

Outra característica geral foi observada na taxa de repetição de chaves. Variando apenas o atributo final de uma mesma expressão de caminho, os índices cujo atributo indexado apresentavam baixa taxa de repetição tinham desempenho superior em consultas pontuais. Entretanto, ao se realizar uma consulta por faixa de valores o resultado se invertia.

Basicamente as taxas de repetição elevadas forçam a existência de páginas de sobrecarga. Isto influencia a taxa de ocupação dos nós-folhas como um todo. Um nó-folha simples de uma árvore B+ tem, em média, taxa de ocupação de 75%. Se o nó-folha necessitar de n páginas de sobrecarga, tem-se as seguintes propriedades:

- O nó-folha simples e os primeiros n-1 nós-de-sobrecarga estão totalmente ocupados.

- O último nó-de-sobrecarga tem, em média, taxa de ocupação igual a 50%.

Assim, a taxa de ocupação média do nó-folha como um todo é dada por $Taxa(n) = \frac{1+n-1+0.5}{n+1} = \frac{n+0.5}{n+1}$. Este valor é em média igual a 75% para n = 1 e sempre maior ou igual a 75% para n = 3. Esta característica independe da estrutura usada no índice e deve ser considerada.

### *4.1.3 Tamanho dos índices*

Analisando-se o tamanho dos índices, pode-se verificar que novamente a taxa de repetição das chaves influencia globalmente em seu tamanho. Fixada a cardinalidade dos valores indexados, quanto menor for a taxa de repetição das chaves o índice tenderá a ocupar mais espaço. Isto é justificado pela taxa de ocupação dos nós-folhas e intermediários que, em média, fica em torno de 75%. Quando a taxa de repetição das chaves é suficientemente grande a ponto de exigir que os nós-folhas usem páginas de sobrecarga, tem-se, então, que os nós-folhas passam a ter, em média, taxa de ocupação superior a 75%. Além disto, como há maior concentração das chaves, tem-se uma menor necessidade de divisões de nós-folhas e, conseqüentemente, menor número de nós-intermediários. Nesta configuração, a árvore passa a ter menor altura.

### *4.1.4 Parâmetros para análise de desempenho*

A análise de desempenho dos índices teve enfoque predominante na realização de consultas. Durante cada medição de desempenho foram coletados os seguintes parâmetros:

- **Tempo -** Tempo necessário para realização da consulta como um todo. Não é feita a separação do tempo de processamento em memória versus disco, como também nenhuma distinção entre tempo de processo versus tempo de sistema operacional.

- **Páginas -** Número de páginas manipuladas pelo índice. É um indicador total de acesso a páginas. Não é feita nenhuma distinção quanto à sua origem (nó-intermediário, nó-folha ou nó-de-sobrecarga), e tampouco se estava ou não no *cache* do SGBD.

- **Páginas em *cache* -** Número de páginas que durante a manipulação do índice estavam no *cache* do GOA++. Assim como no parâmetro anterior, não é feita nenhuma distinção quanto à sua origem (nó-intermediário, nó-folha ou nó-de-sobrecarga).

- **Páginas carregadas -** Número total de páginas efetivamente trazidas de disco.

- **Intermediários -** Número de nós-intermediários manipulados pelo índice estando ou não no *cache* do GOA++.

- **Folhas -** Número de nós-folhas manipulados pelo índice estando ou não no *cache* do GOA++.

- **Sobrecarga -** Número de nós-de-sobrecarga manipulados pelo índice estando ou não no *cache* do GOA++.

Além disto, para auxiliar a compreensão dos resultados medidos, foram coletadas informações estruturais dos índices para correlacioná-las com os resultados experimentais como, por exemplo, tamanho do índice (nós-folhas, nós-intermediários, nós-de-sobrecarga), cardinalidade, valores distintos e grau de compartilhamento dos atributos e taxa de repetição de chaves.

A relação completa dos dados coletados na execução do benchmark OO7 pode ser obtida diretamente em http://www.cos.ufrj.br/~senna.

Por questões de comodidade e para facilitar a compreensão dos gráficos foram feitas algumas abreviações para as classes como apresentado na tabela a seguir.

Tabela 5 – Abreviações adotadas para classes do modelo OO7

| Classes | Abreviação |
|---|---|
| Assemblies | Ass |
| CompositeParts | Cmp |
| Documents | Doc |
| Atomic Parts | Atm |

## 4.2. Hierarquia de classes

A medição experimental realizada faz a comparação dos índices SCI (índice para cada classe) e CHI (índice para hierarquia de classes). Neste aspecto, assim como em OGASAWARA e MATTOSO [38], este trabalho se assemelha ao de Kim et al. [6] pois ambos se preocupam em avaliar a relação custo/benefício entre os índices SCI e CHI. Entretanto, enquanto em Kim et al. [6] os resultados se baseiam em simulações, as avaliações aqui apresentadas foram realizadas num sistema real, envolvendo o armazenamento de objetos persistentes. Nesse sentido, foi observado que o uso de um sistema de paginação de banco de dados evidenciou o impacto das diferenças estruturais dos nós-folhas 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 do que o SCI. Entretanto, os resultados apresentaram consultas típicas com faixas de valores, envolvendo de 5 a 20% do domínio da chave, onde o índice SCI possui um desempenho superior ao CHI. De fato, alguns resultados de comparações [11][7] indicam que o CHI não é eficiente para consultas por faixa de valores que não envolvam a hierarquia completa. Porém, foi evidenciado experimentalmente que essa observação depende do número de classes envolvidas na consulta e da taxa de repetição dos valores da chave dentre os objetos indexados.

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.[6], 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 (ver seção 2.2.6). Se os valores do atributo estão distribuídos ao longo de todas as classes na hierarquia, tem-se uma distribuição conjunta 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. Nos experimentos realizados não foram apresentados 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.

Os experimentos foram divididos em dois grupos de consultas. No primeiro grupo foram avaliadas consultas envolvendo o operador de igualdade (consulta pontual), enquanto no segundo grupo foram avaliadas consultas por faixa de valores (consulta por faixa de valores ou de intervalo). Para os dois grupos foi utilizada a mesma base e feitas variações no número de classes envolvidas e na taxa de repetição dos valores da chave, medindo o tempo de resposta. No benchmark foi indexada a hierarquia de classe de *Atomic Parts* (Atm) para os atributos *Date*, *XDate*, *Type* e *XType*. A hierarquia criada possuía trezentas mil *Atomic Parts* divididas em dezesseis classes. A diferença entre os atributos indexados foi a taxa de repetição das chaves que possuía, respectivamente, os seguintes valores: 10, 100, 1.000 e 10.000. Para cada consulta avaliada foram realizadas trinta medições e reportada a média para cada execução com partida quente. As consultas foram executadas alternadamente para evitar a influência do sistema de paginação do GOA++.

### *4.2.1 Consultas pontuais*

As consultas pontuais refletem a rapidez do índice em responder a buscas por valores específicos. Um exemplo de consulta pontual seria: “select Atm where Atm.Date = 1999 and (Atm is Classe *C(*1) or Atm is Classe *C(*2))”. Esta consulta estaria selecionando todas as *Atomic Parts* que fossem fabricadas em 1999 e que fossem especializações de *C(*1) ou *C(*2).

Conforme observado em Kemper e Moerkotte [4], 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 sofre uma degradação proporcional ao número de classes. Esses resultados foram de fato confirmados nos experimentos realizados. O índice CHI obteve comportamento linear em todas as variações do número de classes, mostrando-se também imune à distribuição de valores nas classes. A seguir são apresentados um gráfico de tempo (Figura 34) e outro de número de páginas acessadas (Figura 35) representativos deste grupo de consultas pontuais.

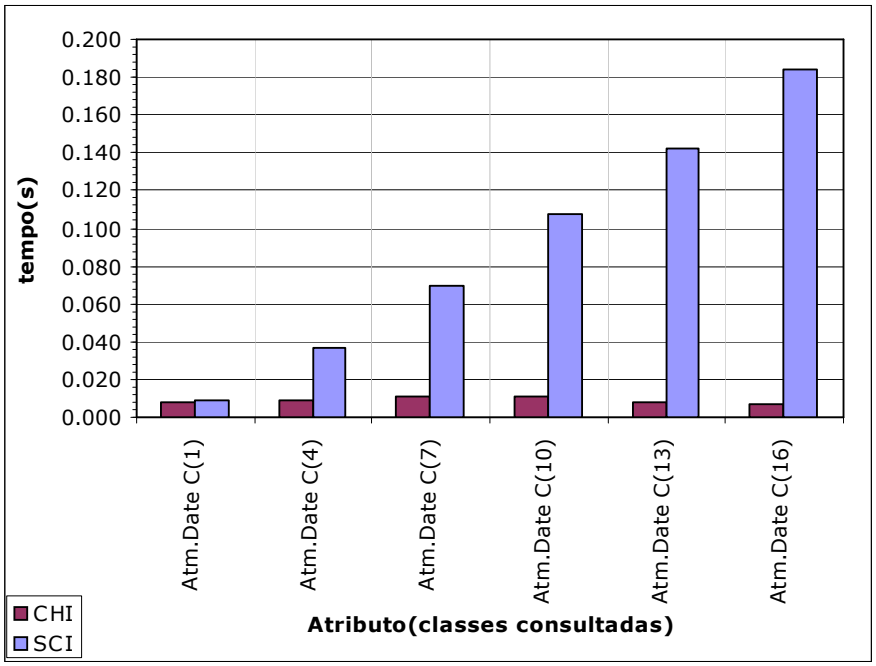



Figura 34 - Consulta pontual, taxa de repetição = 10
(tempo)

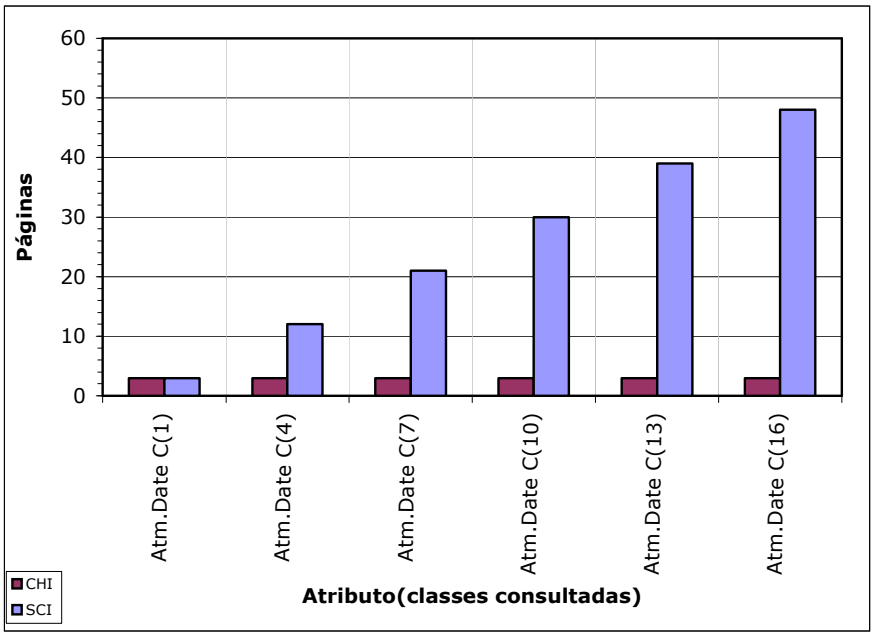



Figura 35 - Consulta pontual, taxa de repetição = 10 (pag. acessadas)

A seguir são apresentados mais dois gráficos de consultas pontuais de tempo (Figura 36) e de número de páginas (Figura 37) com taxa de repetição de chaves muito alta.

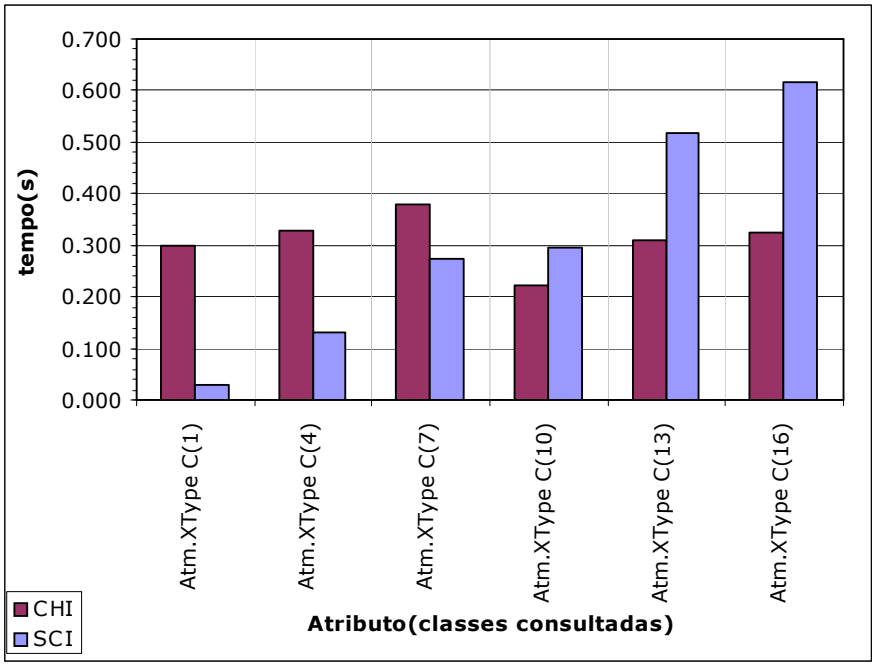


Figura 36 - Consulta pontual, taxa de repetição = 10000
(tempo)

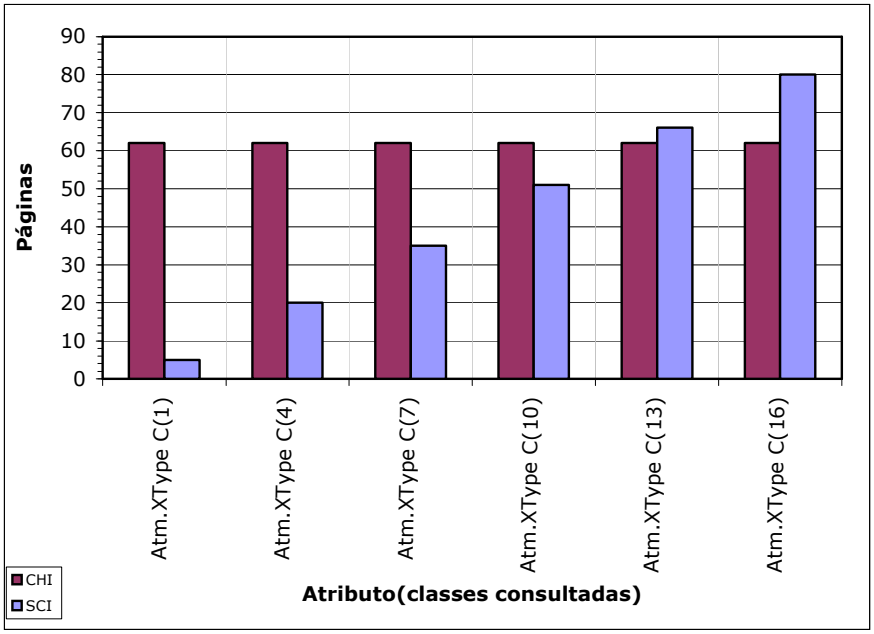



Figura 37 - Consulta pontual, taxa de repetição = 10000
(pag. acessadas)

Nesta configuração, a consulta pontual passa a se comportar como consulta por faixa de valores, diferindo dos resultados clássicos obtidos em [6][7][35]. Nesta configuração o CHI não se apresentou nitidamente superior ao SCI. Na verdade, houve uma alternância de desempenho entre o CHI e o SCI caracterizada pelo número de classes envolvidas na consulta.

### *4.2.2 Consultas por faixa de valores*

As consultas por faixa de valores indicam a robustez da estrutura do índice na busca por uma faixa de valores. Um exemplo de consulta por faixa de valores seria: "*select Atm where 1998 ≤ Atm.Date and Atm.Date ≤ 1999 and (Atm is Classe C(1) or Atm is Classe C(2))*". Esta consulta estaria selecionando todas as *Atomic Parts* que fossem fabricadas entre 1998 e 1999 e que fossem especializações de *C(*1) ou *C(*2).

Nas simulações de Kim et al. [6] para consultas por faixa de valores, 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, foi observado experimentalmente que essas duas situações ocorrem de fato. Na realidade, esses trabalhos fizeram uma generalização de comportamento para consultas por faixa de valores que não se aplica.

Pôde-se detectar experimentalmente a influência de dois parâmetros nesse grupo de consultas que interferiram diretamente nos resultados. O primeiro parâmetro diz respeito à taxa de repetição de valores no índice. O segundo parâmetro observado 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 de [7] e de [6]. Enquanto em Kim et al. essa taxa era muito alta, da ordem de duzentas a seiscentas repetições por valor, em Kilger e Moerkotte [7] essa taxa era igual a um, ou seja, sem repetições. Do ponto de vista do percentual da faixa de valores, em Kim et al. [6] essa taxa não é explicitada nem são avaliadas variações, enquanto que em Kilger e Moerkotte [7] a taxa era fixa em dez.

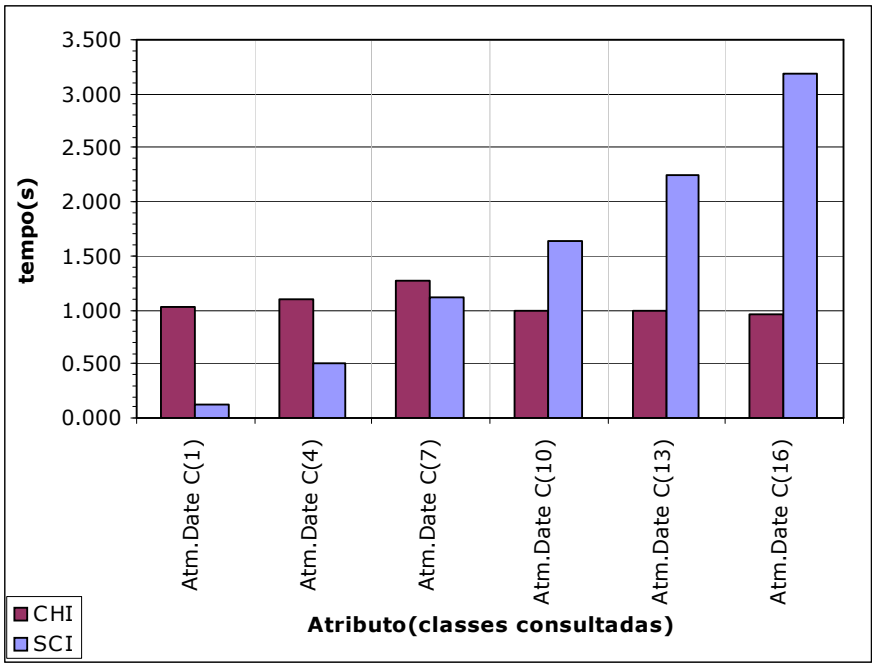



Figura 38 – Faixa de valores = 5%, taxa de repetição = 10 (tempo)

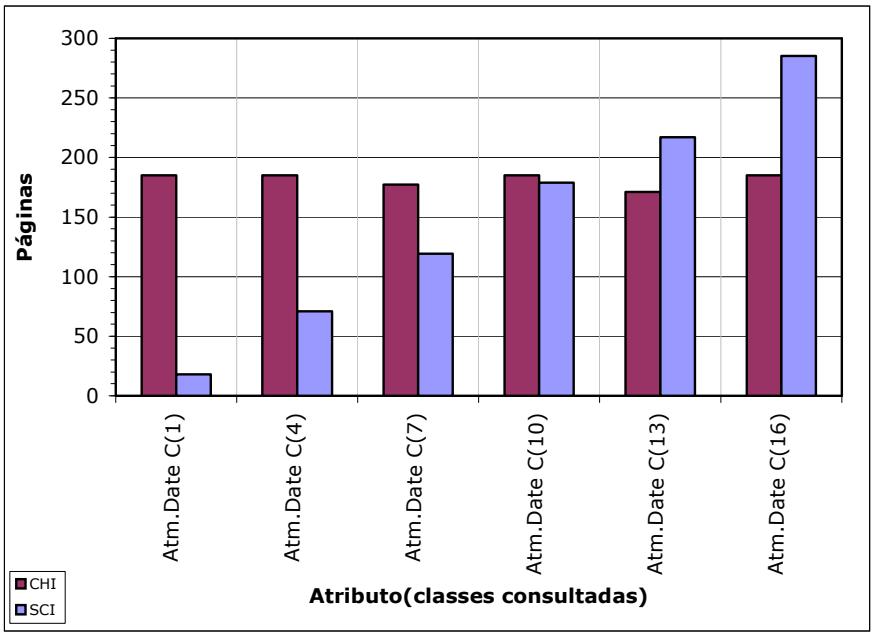



Figura 39 – Faixa de valores = 5%, taxa de repetição = 10 (pag. acessadas)

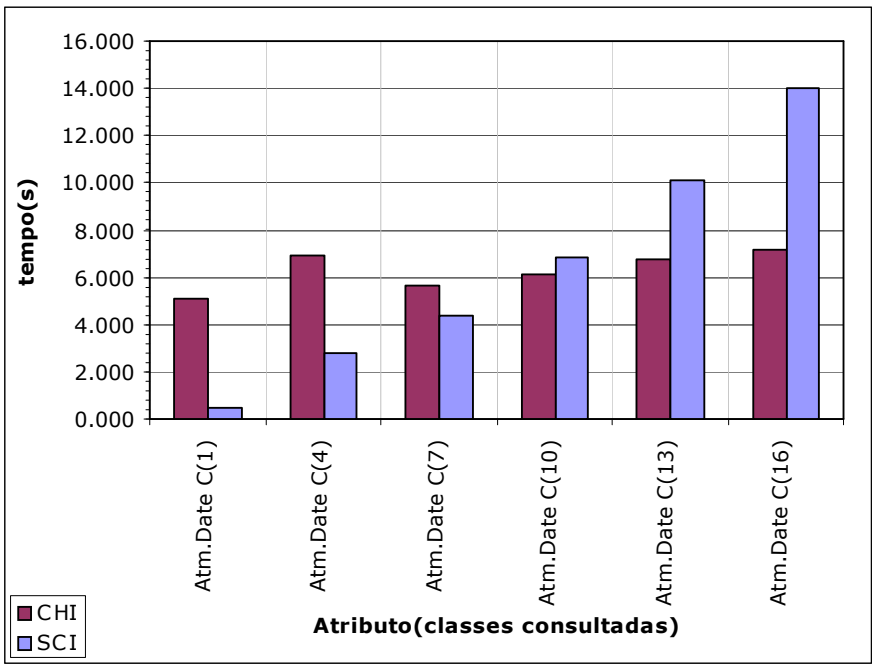



Figura 40 – Faixa de valores = 20%, taxa de repetição = 10 (tempo)

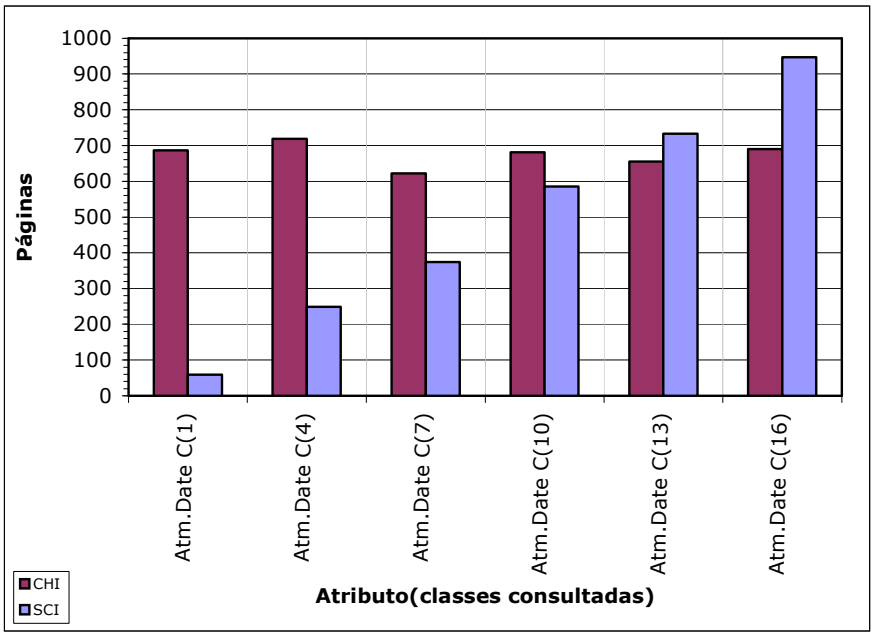



Figura 41 – Faixa de valores = 20%, taxa de repetição = 10 (pag. acessadas)

Nos resultados experimentais, quando a faixa de valores é baixa (1%), o índice CHI possui um desempenho superior ao SCI, já que esse comportamento é semelhante à consulta por valor. Entretanto, quando a faixa de valores aumenta um pouco (5%), para um número de classes menor ou igual a oito, o índice SCI supera o CHI (Figura 38 e Figura 39). À medida em que o número de classes vai aumentando, o SCI apresenta sensibilidade ao número de classes envolvidas e degrada seu desempenho. Quando a faixa de valores sobe para 20% (Figura 40 e Figura 41), há uma inversão de comportamento em relação à faixa de valor igual a 1%. A situação fica na maior parte do tempo 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].

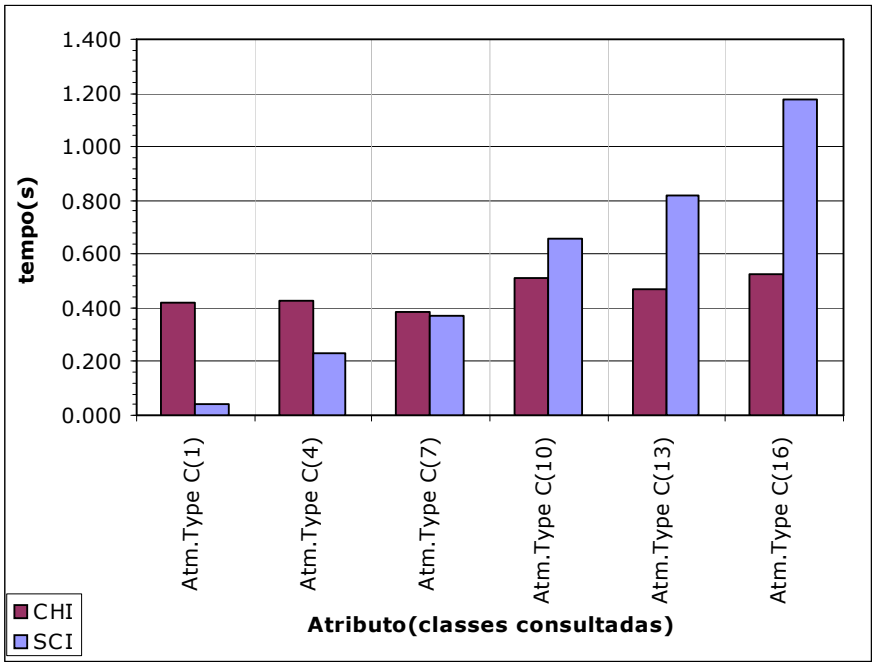



Figura 42 – Faixa de valores = 5%, taxa de repetição = 1000 (tempo)

Nas consultas com taxa de repetição alta - situação que corresponde à analise de Kim et al. [6] – o índice SCI apresentou um desempenho superior ao do CHI até 8 classes. A partir daí o domínio passou a ser do CHI. Entretanto, quanto maior a faixa de valores, o índice CHI apresenta uma maior degradação de desempenho (Figura 42, Figura 43, Figura 44 e Figura 45).

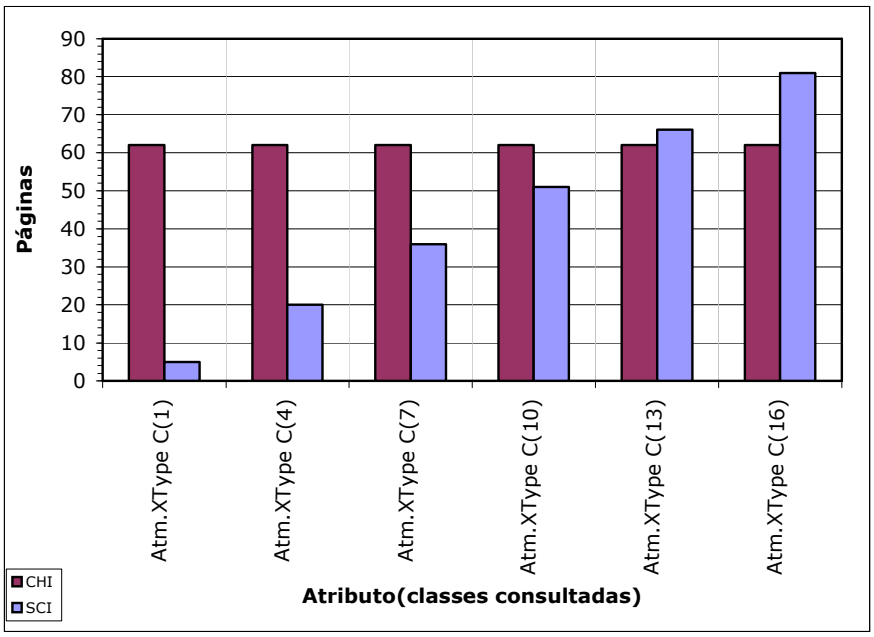



Figura 43 – Faixa de valores = 5%, taxa de repetição = 1000 (páginas)

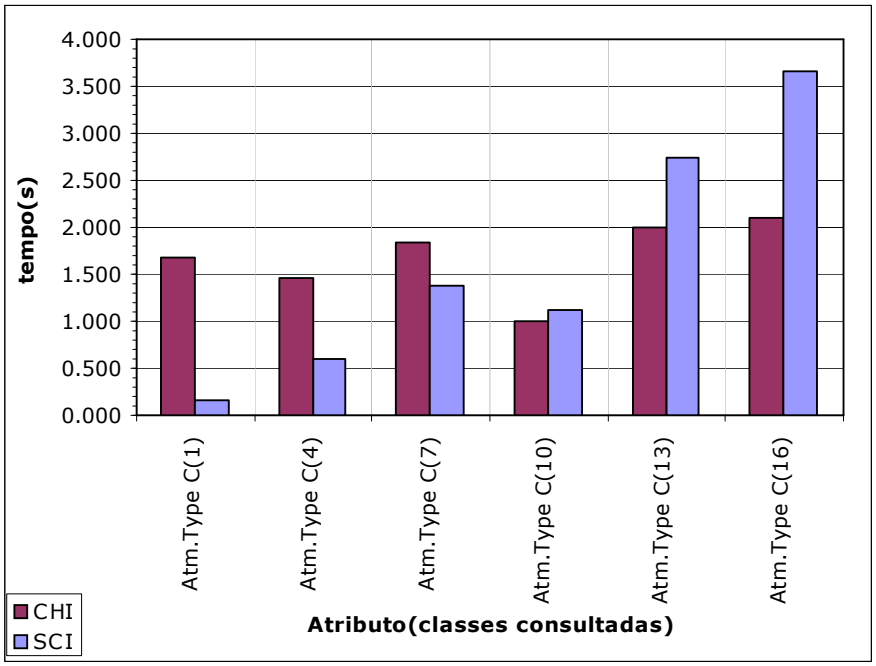



Figura 44 – Faixa de valores = 20%, taxa de repetição = 1000 (tempo)

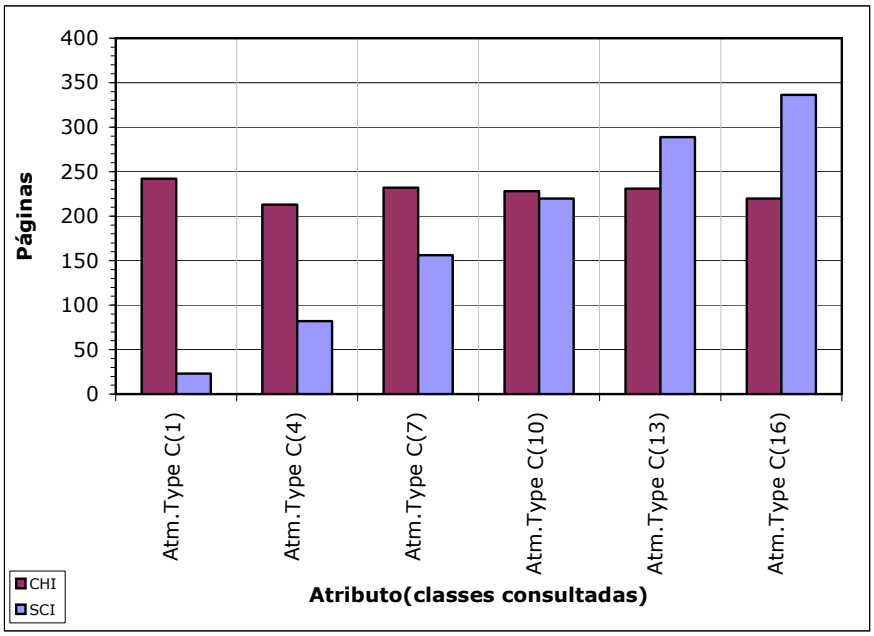



Figura 45 – Faixa de valores = 20%, taxa de repetição = 1000 (páginas)

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.

### *4.2.3 Comparação entre os índices hierárquicos*

Em todas as situações o comportamento do CHI se mantém constante no que diz respeito ao número de classes envolvidas. Isso não ocorre com o SCI. Logo, nas demais análises busca-se identificar em que situações o SCI é melhor do que o CHI. Analisando a relação custo/benefício, pode-se concluir que o CHI nem sempre apresenta os melhores resultados. Fica evidenciado, no entanto, que para consultas pontuais, independentemente 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 por faixa de valores é necessária uma análise mais cuidadosa, pois o desempenho varia de acordo com a taxa de repetição 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 à taxa de repetição das chaves e à faixa de valores especificada na consulta.

Finalmente, os resultados apontam 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 oito classes.

Tabela 6 – Heurísticas para escolha dos índices para hierarquia de classes

| | | Tipo | Repetição | Classes | Índice |
|---|---|---|---|---|---|
| **Distribuição** | Conjunta | Pontual | Baixa/Alta | 0..16 | **CHI** |
| | | Faixa | Baixa | 0..8 | **SCI** |
| | | | | 8..16 | **CHI/SCI** |
| | | | Alta | 0..16 | **SCI** |
| | Disjunta | Faixa / Pontual | Baixa/Alta | 0..16 | **SCI** |

O número oito para as classes envolvidas na consulta representa, na verdade, um ponto de corte e pode sofrer variações dependendo do sistema de banco de dados utilizado.

## 4.3. Expressões de Caminho

Dada uma expressão de caminho $P = C_1.A_1.A_2....A_n$, de tamanho $n$, pode-se definir os seguintes parâmetros:

***$C_i$*:** classe na i-ésima posição da expressão de caminho ($1 \leq i \leq n$).

***$Atr_1$*:** atributo da classe $C_1$.

***$Atr_i$*:** atributo da classe $C_i$, tal que $C_i$ é domínio do atributo $Atr_{i-1}$ da classe $C_{i-1}$ ($1 < i \leq n$).

***Card*(i):** cardinalidade de $C_i$.

***Dist*(i):** o número de valores distintos para o atributo $Atr_i$, tendo significado de número de referências de instâncias de $C_i$ para $C_{i+1}$.

***Comp*(i):** grau de compartilhamento de atributos dado por: $\lceil Card(i)/Dist(i) \rceil$.

***Taxa de repetição de chaves:*** representa a taxa de repetição das chaves indexadas. Seu valor numérico é equivalente ao *Comp(n)*.

Pode-se tomar como exemplo a expressão de caminho *Ass.Cmp.Atm.Date* do modelo OO7 (Figura 33). Nesta expressão de caminho, $C_1$ é representada por *Ass*. $Atr_1$, $Atr_2$ e $Atr_3$ são definidas respectivamente por *Cmp*, *Atm*, *Date*. $Atr_1$ e $Atr_2$ têm como domínio as classes $C_2$ e $C_3$ respectivamente.

Para cada expressão de caminho indexada foram criados índices PX, NX e MX. A medição de desempenho de cada índice foi avaliada tanto para consultas pontuais quanto para consultas por faixa de valores. Além disto, foram observados também os seus desempenhos em consultas que envolvessem uma subexpressão de caminho a partir do atributo indexado.

### *4.3.1 Subconsultas e projeção das consultas*

As consultas foram divididas em consultas a expressões de caminho de tamanhos dois e três. O processamento de consultas a expressões de caminho de tamanho quatro em diante deve ser realizado através da utilização de índices que dividam a expressão de caminho em subexpressões de menor tamanho. O ponto de quebra das subconsultas

influencia o desempenho geral do processamento da consulta como um todo e é objeto de estudo em [5] e [1].

Os índices MX e PX para expressões de caminho de tamanhos dois e três podem ser utilizados para atender às subconsultas a partir do atributo indexado. Assim, numa expressão de caminho $C_1.Atr_1.Atr_2.Atr_3$ indexada, tem-se que o PX e o MX atendem tanto à $C_3.Atr_3$ quanto à $C_2.Atr_2.Atr_3$. Já o NX, por definição (página 18), não pode ser usado.

Outro aspecto importante está nas projeções das consultas. Mesmo que a consulta envolva restrições à expressão de caminho sob o atributo indexado, não necessariamente se tem a escolha direta do índice mais adequado para o seu processamento. É preciso também avaliar a projeção da consulta para se poder realizar a seleção. Esta preocupação em relação à projeção foi observada tanto em [35] quanto em [39].

Pode-se tomar como exemplo a seguinte consulta: "*select* $C_1.*$ *from* $C_1$ *where* $C_1.Atr_1.Atr_2.Atr_3 = valor$". Se o índice NX for o mais rápido para satisfazer a restrição da consulta, ele pode ser usado. Entretanto, modificando a projeção da consulta como, por exemplo: "*select* $C_1.Atr_1.*$ *from* $C_1$ *where* $C_1.Atr_1.Atr_2.Atr_3 = valor$", observa-se que, mesmo o NX sendo o mais rápido para a execução da restrição da consulta, ele não deve ser utilizado pois só retorna a lista de oids da classe $C_1$. Em conseqüência, o usuário ou o gerador do plano de execução terá que incluir uma nova varredura aos OIDs selecionados para montar o resultado. Já o PX ou o MX podem ser adotados para a execução da restrição da consulta, visto que retornam tuplas de oids da forma ($OID_A$, $OID_B$, $OID_C$). Assim, ao satisfazer a restrição da consulta, é possível obter o seu resultado final de modo trivial.

## 4.4. Expressões de caminho de tamanho dois

Foram objeto de análise durante as medições do OO7 as seguintes expressões de caminho: *Cmp.Doc.Date, Cmp.Doc.Type, Doc.Cmp.Date, Doc.Cmp.Type, Ass.Cmp.Date, Ass.Cmp.Type, Cmp.Ass.Date, Cmp.Ass.Type, Atm.Cmp.Date, Atm.Cmp.Type, Cmp.Atm.Date* e *Cmp.Atm.Type*. Ao todo foram indexadas seis expressões de caminho variando o atributo indexado. Esta variação entre os atributos *Date* e *Type* foi feita para explorar a taxa de repetição de chaves. Nos experimentos, o atributo *Date* representa baixa taxa de repetição de chaves, enquanto *Type* está representando alta taxa de repetição de chaves. Os valores absolutos para as taxas de

repetição de chaves são apresentados na Tabela 7. Esta variação é importante, visto que no trabalho da Bertino e Kim [5] este parâmetro foi pouco explorado, nunca passando de cinqüenta. Desta forma o impacto das páginas de sobrecarga pode ficar escondido. Em cardinalidades muito grandes com alta taxa de repetição de chaves, o valor cinqüenta é considerado não representativo.

As expressões de caminho (Tabela 7) foram definidas de modo a propiciar diversas formas de navegação em diferentes configurações de cardinalidade e grau de compartilhamento de objetos. Na Tabela 7, por exemplo, no caso da expressão de caminho *Ass.Cmp.Date*, *Card(1)* representa a cardinalidade de *Base Assembly* e *Card(2)* representa a cardinalidade de *Composite Parts*.

Tabela 7 - Configuração das expressões de caminho de tamanho dois

| Indice | Card(1) | Dist(1) | Comp(1) | Card(2) | Dist(2) | Comp(2) |
|---|---|---|---|---|---|---|
| Ass.Cmp.Date | 2872 | 9000 | 1 | 9000 | 5645 | 2 |
| Ass.Cmp.Type | 2872 | 9000 | 1 | 9000 | 900 | 10 |
| Atm.Cmp.Date | 180000 | 9000 | 20 | 9000 | 5645 | 2 |
| Atm.Cmp.Type | 180000 | 9000 | 20 | 9000 | 900 | 10 |
| Cmp.Ass.Date | 9000 | 3000 | 3 | 3000 | 1902 | 2 |
| Cmp.Ass.Type | 9000 | 3000 | 3 | 3000 | 300 | 10 |
| Cmp.Atm.Date | 9000 | 180000 | 1 | 180000 | 17999 | 11 |
| Cmp.Atm.Type | 9000 | 180000 | 1 | 180000 | 180 | 1000 |
| Cmp.Doc.Date | 5745 | 9000 | 1 | 9000 | 900 | 10 |
| Cmp.Doc.Type | 5745 | 9000 | 1 | 9000 | 90 | 100 |
| Doc.Cmp.Date | 9000 | 9000 | 1 | 9000 | 5645 | 2 |
| Doc.Cmp.Type | 9000 | 9000 | 1 | 9000 | 900 | 10 |

As variações realizadas nas expressões de caminho também foram comuns nas simulações. Um dos objetivos dessas configurações é verificar a sensibilidade dos índices quando a cardinalidade da classe raiz é muito maior que a cardinalidade da classe fim e vice-versa. É sabido [4] que esses fatores influem na ordem em que se percorre a expressão de caminho sem índice, seja descendente (*forward*) ou ascendente (*backward*). Logo, é importante analisar esses comportamentos nos índices.

Tabela 8 - composição dos índices de tamanho dois

| Indice | Tipo | Chaves | Objetos | Inter | Folha | Sobre | Paginas |
|---|---|---|---|---|---|---|---|
| Ass.Cmp.Date | MX12 | 9000 | 9000 | 9 | 449 | 0 | 458 |
| Ass.Cmp.Date | MXa1 | 5645 | 9000 | 3 | 210 | 0 | 213 |
| Ass.Cmp.Date | NX2 | 5645 | 3353 | 4 | 227 | 0 | 231 |
| Ass.Cmp.Date | PX | 5645 | 9000 | 7 | 414 | 0 | 421 |
| Ass.Cmp.Type | MX12 | 9000 | 9000 | 9 | 449 | 0 | 458 |
| Ass.Cmp.Type | MXa1 | 900 | 9000 | 1 | 100 | 0 | 101 |
| Ass.Cmp.Type | NX2 | 900 | 8090 | 3 | 136 | 0 | 139 |
| Ass.Cmp.Type | PX | 900 | 9000 | 5 | 286 | 0 | 291 |
| Atm.Cmp.Date | MX12 | 9000 | 180000 | 33 | 1838 | 0 | 1871 |
| Atm.Cmp.Date | MXa1 | 5645 | 9000 | 3 | 210 | 0 | 213 |
| Atm.Cmp.Date | NX2 | 5645 | 174355 | 48 | 2961 | 0 | 3009 |
| Atm.Cmp.Date | PX | 5645 | 180000 | 82 | 5067 | 79 | 5228 |
| Atm.Cmp.Type | MX12 | 9000 | 180000 | 33 | 1838 | 0 | 1871 |
| Atm.Cmp.Type | MXa1 | 900 | 9000 | 1 | 100 | 0 | 101 |
| Atm.Cmp.Type | NX2 | 900 | 179100 | 15 | 900 | 652 | 1567 |
| Atm.Cmp.Type | PX | 900 | 180000 | 15 | 900 | 1732 | 2647 |
| Cmp.Ass.Date | MX12 | 2872 | 9000 | 3 | 151 | 0 | 154 |
| Cmp.Ass.Date | MXa1 | 1902 | 3000 | 1 | 70 | 0 | 71 |
| Cmp.Ass.Date | NX2 | 1854 | 7146 | 3 | 153 | 0 | 156 |
| Cmp.Ass.Date | PX | 1854 | 9000 | 5 | 304 | 0 | 309 |
| Cmp.Ass.Type | MX12 | 2872 | 9000 | 3 | 151 | 0 | 154 |
| Cmp.Ass.Type | MXa1 | 300 | 3000 | 1 | 34 | 0 | 35 |
| Cmp.Ass.Type | NX2 | 300 | 8700 | 3 | 145 | 0 | 148 |
| Cmp.Ass.Type | PX | 300 | 9000 | 5 | 281 | 0 | 286 |
| Cmp.Atm.Date | MX12 | 180000 | 180000 | 183 | 8999 | 0 | 9182 |
| Cmp.Atm.Date | MXa1 | 17999 | 180000 | 26 | 1987 | 0 | 2013 |
| Cmp.Atm.Date | NX2 | 17999 | 161906 | 45 | 2737 | 0 | 2782 |
| Cmp.Atm.Date | PX | 17999 | 180000 | 93 | 5720 | 0 | 5813 |
| Cmp.Atm.Type | MX12 | 180000 | 180000 | 183 | 8999 | 0 | 9182 |
| Cmp.Atm.Type | MXa1 | 180 | 180000 | 3 | 180 | 1012 | 1195 |
| Cmp.Atm.Type | NX2 | 180 | 170030 | 3 | 180 | 912 | 1095 |
| Cmp.Atm.Type | PX | 180 | 180000 | 3 | 180 | 2088 | 2271 |
| Cmp.Doc.Date | MX12 | 9000 | 9000 | 9 | 449 | 0 | 458 |
| Cmp.Doc.Date | MXa1 | 900 | 9000 | 1 | 104 | 0 | 105 |
| Cmp.Doc.Date | NX2 | 900 | 8095 | 3 | 136 | 0 | 139 |
| Cmp.Doc.Date | PX | 900 | 9000 | 5 | 285 | 0 | 290 |
| Cmp.Doc.Type | MX12 | 9000 | 9000 | 9 | 449 | 0 | 458 |
| Cmp.Doc.Type | MXa1 | 90 | 9000 | 1 | 90 | 0 | 91 |
| Cmp.Doc.Type | NX2 | 90 | 8873 | 1 | 90 | 0 | 91 |
| Cmp.Doc.Type | PX | 90 | 9000 | 1 | 90 | 87 | 178 |
| Doc.Cmp.Date | MX12 | 5745 | 9000 | 5 | 232 | 0 | 237 |
| Doc.Cmp.Date | MXa1 | 5645 | 9000 | 3 | 210 | 0 | 213 |
| Doc.Cmp.Date | NX2 | 4214 | 4786 | 4 | 198 | 0 | 202 |
| Doc.Cmp.Date | PX | 4214 | 9000 | 6 | 369 | 0 | 375 |
| Doc.Cmp.Type | MX12 | 5745 | 9000 | 5 | 232 | 0 | 237 |
| Doc.Cmp.Type | MXa1 | 900 | 9000 | 1 | 100 | 0 | 101 |
| Doc.Cmp.Type | NX2 | 897 | 8103 | 3 | 138 | 0 | 141 |
| Doc.Cmp.Type | PX | 897 | 9000 | 5 | 294 | 0 | 299 |

A Tabela 8 apresenta a composição dos índices armazenados na base de dados do GOA++. São apresentadas as colunas *índice* e *tipo* que, juntas, especificam o índice usado numa expressão de caminho. As colunas *chaves* e *objetos* indicam respectivamente o número de chaves e de objetos armazenados pelo índice. As colunas *inter*, *folha*, *sobre* e *páginas* representam respectivamente o número de nós-intermediários, nós-folhas, nós-de-sobrecarga e total de páginas consumido pelo índice. As colunas de páginas também são indicativos do espaço ocupado em disco, visto que

cada página do sistema ocupa 1kB. No caso dos índices MX, os mesmos são apresentados separadamente por serem na verdade uma composição de índices. Assim, MXa1 representa o índice MX que leva do atributo indexado à primeira classe na instanciação inversa. O índice MX12 liga a primeira classe na instanciação inversa à segunda classe na instanciação inversa e assim por diante. Pode-se tomar como exemplo a expressão de caminho *Ass.Cmp.Date*. Os índices NX e PX para esta expressão ocupam respectivamente 231kB e 421kB, enquanto o índice MX ocupa ao todo 671kB (213kB pelo MXa1 e 548kB pelo MX12).

A Tabela 8 evidencia a importância da avaliação da taxa de repetição de chaves caracterizada pela variação dos atributos *Date* e *Type* num mesmo caminho de uma expressão de caminho, conforme apresentado na Tabela 7. Nas linhas que representam o caminho *Cmp.Atm* nota-se o excesso de páginas de sobrecarga para a expressão de alta taxa de repetição (*Cmp.Atm.Type*). A conseqüência do impacto deste parâmetro é observada pela diferença de tempo entre estas duas expressões de caminho (*Cmp.Atm.Date* versus *Cmp.Atm.Type)* na Figura 46.

### *4.4.1 Tamanho dos índices*

Devido às suas características estruturais, o NX sempre ocupou menos espaço do que o PX. Na maioria das vezes o NX ocupou a metade do espaço ocupado pelo PX, visto que no PX armazena-se o caminho inteiro (dois OIDs), enquanto o NX armazena apenas o objeto alvo (um OID). Da mesma forma o NX predominantemente gastou menos espaço do que o MX. O MX foi mais eficiente que o NX apenas quando a cardinalidade de $C_1$ era muito maior que a cardinalidade de $C_2$ e a taxa de repetição de chaves era baixa.

Já comparando o espaço ocupado pelo PX e pelo MX, observou-se uma alternância no desempenho dos índices. Pode-se comparar o MX com o PX em duas configurações básicas:

- *Card(1)* ≤ *Card(2)*

  Esta situação é observada nas expressões de caminho *Cmp.Doc.Date, Cmp.Doc.Type, Ass.Cmp.Date, Ass.Cmp.Type, Cmp.Atm.Date* e *Cmp.Atm.Type.* Quanto menor for *Card(1)* em relação a *Card(2)*, o grau de compartilhamento de objetos da classe $C_1$ para a classe $C_2$ tenderá a um. Isto faz com que o índice MX12 se comporte como um índice de taxa de repetição muito baixa. O número de nós-intermediários do índice aumenta enquanto a taxa de ocupação dos nós-

folhas diminui. Esta combinação de fatores aumenta o consumo de páginas da árvore. Nesta configuração o PX gasta menos espaço do que o MX.

- *Card(1) > Card(2)*

  Esta situação é observada nas expressões de caminho *Cmp.Ass.Date, Cmp.Ass.Type, Atm.Cmp.Date* e *Atm.Cmp.Type*. Nesta configuração o grau de compartilhamento de objetos da classe $C_1$ para a classe $C_2$ é necessariamente maior do que um. O MX12 passa a não influenciar tanto no gasto geral de páginas do índice MX como um todo, enquanto o índice PX tende a apresentar redundância de informação nos nós-folhas. Nesta configuração o PX gasta mais espaço do que o MX.

Em resumo, se a cardinalidade de $C_1$ for menor ou igual a de $C_2$, o tamanho do índice PX será menor do que o tamanho do índice MX. Se a cardinalidade de $C_1$ for maior do que $C_2$, o tamanho do índice PX será maior do que o do índice MX.

### *4.4.2 Análise de desempenho*

#### *4.4.2.1 Consultas pontuais*

A Figura 46 e a Figura 47 apresentam o desempenho dos índices, respectivamente, nas dimensões de tempo e páginas para a seguinte consulta: "*select $C_1$.* from $C_1$ where $C_1.Atr_1.Atr_2$ = valor*". A Tabela 9 apresenta informações adicionais obtidas nas medições.

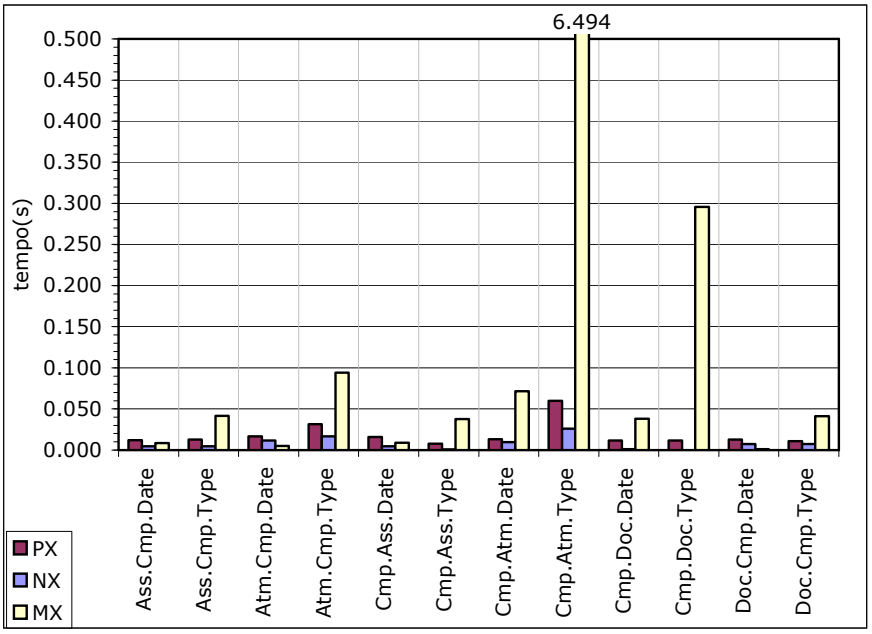


Figura 46 - Consulta pontual em expressões de tamanho dois (tempo)

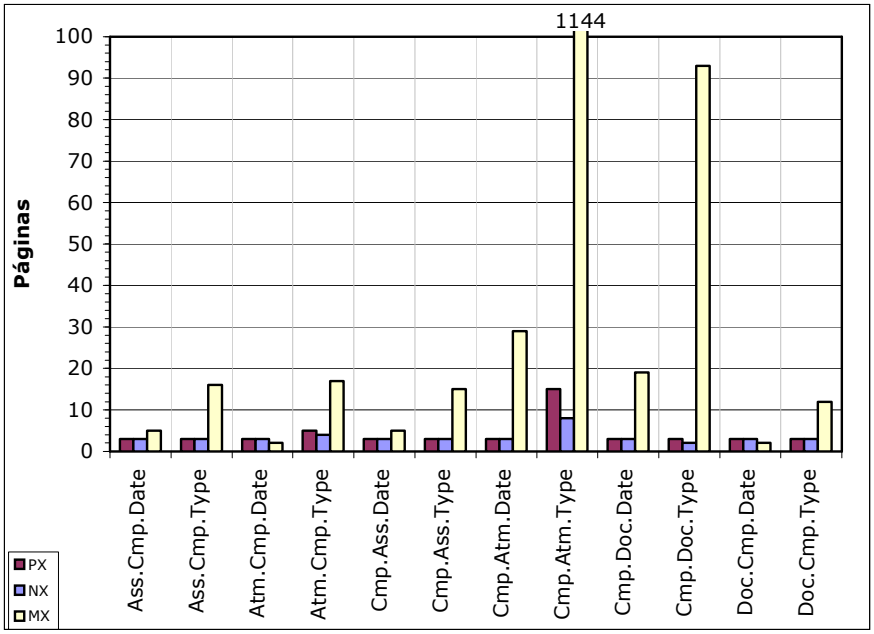



Figura 47 - Consulta pontual em expressões de tamanho dois (pag. carregadas)

Tabela 9 – Consulta pontual em expressões de tamanho dois

| | Intermediárias | | | Folhas | | | Sobrecarga | | | Páginas | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Config** | **PX** | **NX** | **MX** | **PX** | **NX** | **MX** | **PX** | **NX** | **MX** | **PX** | **NX** | **MX** |
| Ass.Cmp.Date | 2 | 2 | 3 | 1 | 1 | 2 | 0 | 0 | 0 | 3 | 3 | 5 |
| Ass.Cmp.Type | 2 | 2 | 19 | 1 | 1 | 10 | 0 | 0 | 0 | 3 | 3 | 29 |
| Atm.Cmp.Date | 2 | 2 | 3 | 1 | 1 | 2 | 0 | 0 | 0 | 3 | 3 | 5 |
| Atm.Cmp.Type | 2 | 2 | 19 | 1 | 1 | 10 | 2 | 1 | 0 | 5 | 4 | 29 |
| Cmp.Ass.Date | 2 | 2 | 4 | 1 | 1 | 2 | 0 | 0 | 0 | 3 | 3 | 6 |
| Cmp.Ass.Type | 2 | 2 | 24 | 1 | 1 | 13 | 0 | 0 | 0 | 3 | 3 | 37 |
| Cmp.Atm.Date | 2 | 2 | 36 | 1 | 1 | 12 | 0 | 0 | 0 | 3 | 3 | 49 |
| Cmp.Atm.Type | 2 | 2 | 3022 | 1 | 1 | 1008 | 12 | 5 | 6 | 15 | 8 | 4036 |
| Cmp.Doc.Date | 2 | 2 | 22 | 1 | 1 | 12 | 0 | 0 | 0 | 3 | 3 | 34 |
| Cmp.Doc.Type | 1 | 1 | 203 | 1 | 1 | 102 | 1 | 0 | 0 | 3 | 2 | 304 |
| Doc.Cmp.Date | 2 | 2 | 3 | 1 | 1 | 2 | 0 | 0 | 0 | 3 | 3 | 5 |
| Doc.Cmp.Type | 2 | 2 | 19 | 1 | 1 | 10 | 0 | 0 | 0 | 3 | 3 | 29 |

Nas consultas pontuais, o índice NX apresentou desempenho superior ao PX. Exceto no caso em que tanto a cardinalidade de $C_1$ é muito maior do que $C_2$ quanto a taxa de

repetição de chaves é muito baixa, o índice NX é sempre superior ao MX. O índice MX teve seu desempenho influenciado predominantemente pela taxa de repetição de chaves. BERTINO e KIM [5] observaram que quanto maior o valor da taxa de repetição de chaves, pior o desempenho do índice MX. Entretanto, eles não avaliaram o quão pior. Nos experimentos realizados, foi verificado que na expressão de *Cmp.Atm.Type* essa diferença na taxa de repetição de chaves pode inviabilizar o uso do MX, já que o volume de páginas manipuladas é muito alto, da ordem de mil páginas em contraposição às cerca de cinqüenta páginas acessadas nas outras situações, conforme mostrado na Tabela 9. Além disto, nas expressões de caminho em que *Card(1)* ≤ *Card(2)* (*Cmp.Doc.Date, Cmp.Doc.Type, Ass.Cmp.Date, Ass.Cmp.Type, Cmp.Atm.Date e Cmp.Atm.Type*), o índice MX apresentou-se desfavorável em relação ao PX e ao NX. Assim, quando a *Card(1)* ≤ *Card(2)* e a taxa de repetição de chaves apresentou-se alta (*Cmp.Doc.Type e Cmp.Atm.Type)*, o índice MX ficou muito aquém do PX e do NX.

Quando a cardinalidade de $C_1$ era muito maior do que $C_2$ e a taxa de repetição de chaves apresentava-se baixa (*Cmp.Ass.Date e Atm.Cmp.Date*), o índice MX mostrou um desempenho competitivo ao dos índices PX e NX. Este resultado também foi avaliado por BERTINO e KIM [5]; entretanto, os autores não ressaltaram que nesta configuração o MX é capaz de apresentar desempenho superior nas dimensões de tempo (Figura 46) e de número de páginas (Figura 47).

#### *4.4.2.2 Consultas por faixa de valores*

A Tabela 10 apresenta o desempenho dos índices para a seguinte consulta por faixa de valores: "*select* $C_1$ *from* $C_1$ *where* $C_1.Atr_1.Atr_2 \geq valor1$ *and* $C_1.Atr_1.Atr_2 \leq valor2$". Neste tipo de consulta, a taxa de repetição de chaves deixa de exercer sua influência direta e o parâmetro dominante para a análise passa a ser a relação entre a cardinalidade de $C_1$ e de $C_2$.

Tabela 10 - Consulta por faixa de valores em expressões de tamanho dois

| | Intermediárias | | | Folhas | | | Sobrecarga | | | Páginas | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Config** | **PX** | **NX** | **MX** | **PX** | **NX** | **MX** | **PX** | **NX** | **MX** | **PX** | **NX** | **MX** |
| Ass.Cmp.Date | 2 | 2 | 877 | 21 | 12 | 449 | 0 | 0 | 0 | 24 | 14 | 1327 |
| Ass.Cmp.Type | 2 | 2 | 860 | 14 | 7 | 435 | 0 | 0 | 0 | 16 | 9 | 1295 |
| Atm.Cmp.Date | 6 | 4 | 877 | 246 | 145 | 449 | 3 | 0 | 0 | 255 | 149 | 1327 |
| Atm.Cmp.Type | 2 | 2 | 860 | 43 | 43 | 435 | 83 | 31 | 0 | 128 | 77 | 1295 |
| Cmp.Ass.Date | 2 | 2 | 297 | 15 | 8 | 152 | 0 | 0 | 0 | 18 | 10 | 450 |
| Cmp.Ass.Type | 2 | 2 | 280 | 14 | 7 | 142 | 0 | 0 | 0 | 16 | 9 | 422 |
| Cmp.Atm.Date | 7 | 4 | 26974 | 287 | 137 | 9090 | 0 | 0 | 0 | 293 | 142 | 36064 |
| Cmp.Atm.Type | 2 | 2 | 29012 | 10 | 10 | 9680 | 112 | 49 | 55 | 124 | 61 | 38746 |
| Cmp.Doc.Date | 2 | 2 | 916 | 15 | 8 | 464 | 0 | 0 | 0 | 17 | 10 | 1380 |
| Cmp.Doc.Type | 1 | 1 | 993 | 5 | 5 | 501 | 5 | 0 | 0 | 11 | 6 | 1494 |
| Doc.Cmp.Date | 2 | 2 | 877 | 19 | 10 | 449 | 0 | 0 | 0 | 21 | 13 | 1327 |
| Doc.Cmp.Type | 2 | 2 | 860 | 14 | 7 | 435 | 0 | 0 | 0 | 16 | 9 | 1295 |

Novamente foi observado que o índice NX sempre possuiu desempenho superior ao PX. BERTINO e KIM [5] concluíram que, nas consultas por faixa de valores, o NX é melhor do que o PX e que o PX é melhor do que o MX, mas, sem exibir claramente em que situação a diferença de desempenho entre o PX e o MX diminui.

Nos resultados experimentais, quando a cardinalidade de $C_1$ é menor ou igual a $C_2$ (*Ass.Cmp.Date*, *Ass.Cmp.Type*, *Cmp.Atm.Date* e *Cmp.Atm.Type*) foi verificado que as junções intermediárias são tão intensas para o MX que seu uso se torna impraticável, talvez sendo pior do que realizar a consulta sem índice. A diminuição da diferença de desempenho entre o PX e o MX ocorre quando a cardinalidade de $C_1$ é maior do que $C_2$ (*Cmp.Ass.Date*, *Cmp.Ass.Type*, *Atm.Cmp.Date* e *Atm.Cmp.Type*).

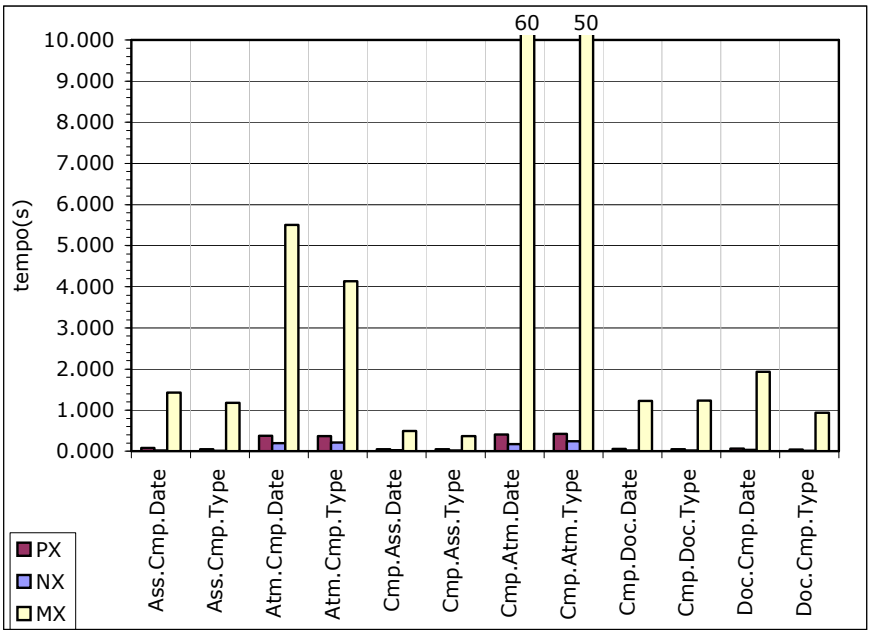



Figura 48 - Consulta por faixa de valores em expressões
de tamanho dois (tempo)

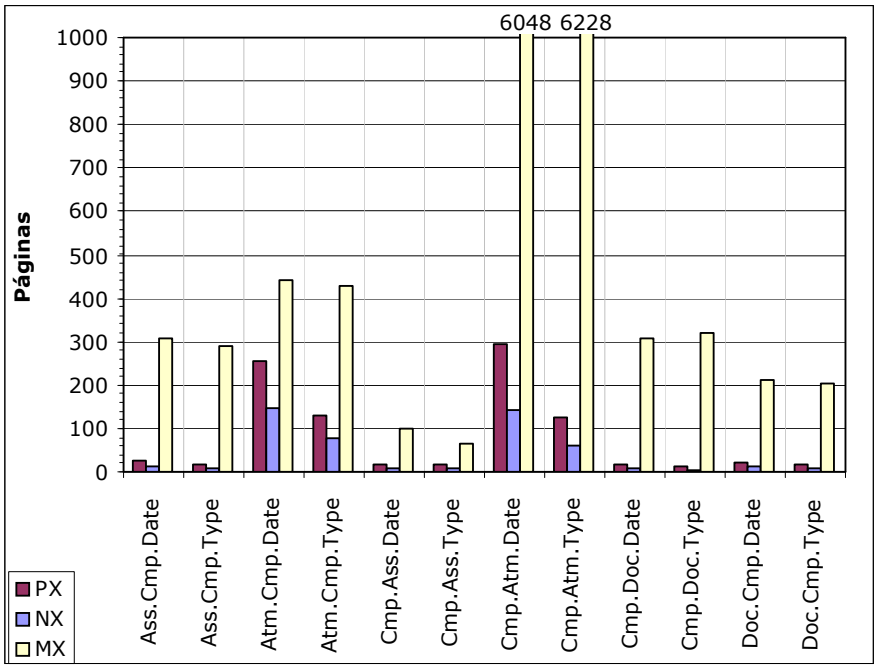



Figura 49 - Consulta por faixa de valores em expressões
de tamanho dois (pag. carregadas)

### 4.4.3 Projeções e subconsultas

Foram identificadas analiticamente e confirmadas experimentalmente as melhores escolhas de índice para diferentes consultas ou subconsultas para expressões de tamanho dois, como mostrado a seguir[3]:

- Projeção final exibindo apenas a classe $C_2$: "*select* $C_1.Atr_1.*$ *from* $C_1$ *where* $C_1.Atr_1.Atr_2 = valor$" ou "*select* $C_1.Atr_1.*$ *from* $C_1$ *where* $C_1.Atr_1.Atr_2 \geq valor1$ *and* $C_1.Atr_1.Atr_2 \leq valor2$"

  Neste caso, o índice MX é a melhor opção, visto que não são necessárias junções intermediárias., basta realizar uma consulta direta no índice MXa1. Já o PX é um índice insensível à projeção final da consulta, sempre podendo ser utilizado. Entretanto, por sua característica de insensibilidade, o tempo gasto para sua execução é sempre o mesmo.

- Projeção final exibindo tanto $C_1$ quanto $C_2$: "*select* $C_1.*$, $C_1.Atr_1.*$ *from* $C_1$ *where* $C_1.Atr_1.Atr_2 = valor$" ou "*select* $C_1.*$, $C_1.Atr_1.*$ *from* $C_1$ *where* $C_1.Atr_1.Atr_2 \geq valor1$ *and A.B.AtrB* $\leq valor2$"

  Salvo nas consultas pontuais em que *Card(*1) > *Card(*2) e o atributo indexado possui baixa taxa de repetição (*Cmp.Ass.Date e Atm.Cmp.Date* - configuração na qual o MX é melhor do que o PX), verifica-se que o índice PX é a melhor opção, enquanto o MX apresenta uma degradação idêntica à mostrada na Tabela 9 e na Tabela 10.

### 4.4.4 Comparações entre os índices para expressões de caminho de tamanho dois

Nas consultas ao longo da expressão de caminho o NX está sempre melhor do que o PX e predominantemente melhor do que o MX. Mesmo nos casos em que o MX apresenta um desempenho superior ao NX, esta melhora não é tão significativa e pode ser desconsiderada. Logo, sempre que não haja a necessidade de subexpressão na projeção da consulta, o índice NX deve ser adotado. A

[3] Não são apresentados resultados referentes ao NX na medida em que este não se aplica, pois é um índice que faz ligação direta entre o atributo indexado e a classe alvo da consulta, não tendo nenhuma informação sobre as classes intermediárias na expressão de caminho.

Tabela 11 apresenta a heurística para a seleção de índices de acordo com a configuração da expressão de caminho.

Comparando-se o PX ao MX, em relação ao tempo, tem-se que o PX é o recomendável. Já no quesito de espaço em disco os resultados não são absolutos, havendo um equilíbrio entre os índices, sendo que o NX tende a consumir menos espaço que os demais.

Em termos conclusivos, considerando-se o destaque da superioridade do NX tanto em relação a tempo quanto em relação a tamanho, sugere-se que um SGBDOO tenha sempre o NX implementado. O índice PX pode ser usado como uma alternativa ao NX para contemplar as subexpressões, tendo como vantagem adicionar um menor custo de manutenção em relação ao NX. Na prática, basta ter um dos dois índices como acelerador no processamento das consultas.

Apesar de tudo o que foi observado, o MX não deve ser descartado, pois é um índice de baixo custo de manutenção. Embora seja inadequado em muitas consultas de expressões de caminho, traz a sua versatilidade em diferentes tipos de consultas. Além disto o MX pode ser aplicado como um mecanismo de controle de esquema e dados como, por exemplo, manutenção de integridade referencial, podendo servir até mesmo como um acelerador para as atualizações dos índices PX e NX.

Tabela 11 - Heurística para escolha de índices para expressões de caminho de tamanho dois

| | | Tipo | Cardinalidade | Repetição | Índice |
|---|---|---|---|---|---|
| **Projeção** | $C_1.*$ | Pontual | $C_1 \leq C_2$ | Baixa/Alta | **NX** |
| | | | $C_1 > C_2$ | Baixa | **NX/MX** |
| | | | | Alta | **NX** |
| | | Faixa | $C_1 \leq C_2$ | Baixa/Alta | **NX** |
| | | | $C_1 > C_2$ | Baixa/Alta | **NX** |
| | $C1.*, C_1.Atr_1.*$ | Pontual | $C_1 \leq C_2$ | Baixa/Alta | **PX** |
| | | | $C_1 > C_2$ | Baixa | **PX/MX** |
| | | | | Alta | **PX** |
| | | Faixa | $C_1 \leq C_2$ | Baixa/Alta | **PX** |
| | | | $C_1 > C_2$ | Baixa/Alta | **PX** |
| | $C_1.Atr_1.*$ | Pontual | $C_1 \leq C_2$ | Baixa/Alta | **MX/PX** |
| | | | $C_1 > C_2$ | Baixa | **MX** |
| | | | | Alta | **MX** |
| | | Faixa | $C_1 \leq C_2$ | Baixa/Alta | **MX** |
| | | | $C_1 > C_2$ | Baixa/Alta | **MX** |

## 4.5. Expressões de caminho de tamanho três

Foram objeto de análise durante as medições do OO7 as seguintes expressões de caminho de tamanho três: *Ass.Cmp.Atm.Date, Ass.Cmp.Atm.Type, Ass.Cmp.Doc.Date, Ass.Cmp.Doc.Type, Atm.Cmp.Ass.Date* e *Atm.Cmp.Ass.Type*.

Conforme realizado na seção anterior, são apresentadas as configurações de cardinalidade e grau de compartilhamento de objetos (Tabela 12) e a composição dos índices para expressões de caminho de tamanho três (Tabela 13).

Tabela 12 - configuração das expressões de caminho de tamanho três

| Índice | N(1) | D(1) | K(1) | N(2) | D(2) | K(2) | N(3) | D(3) | K(3) |
|---|---|---|---|---|---|---|---|---|---|
| Ass.Cmp.Atm.Date | 2872 | 9000 | 1 | 9000 | 180000 | 1 | 180000 | 17999 | 11 |
| Ass.Cmp.Atm.Type | 2872 | 9000 | 1 | 0 | 180000 | 1 | 180000 | 180 | 1000 |
| Ass.Cmp.Doc.Date | 2872 | 5745 | 1 | 5745 | 9000 | 1 | 9000 | 900 | 10 |
| Ass.Cmp.Doc.Type | 2872 | 5745 | 1 | 0 | 9000 | 1 | 9000 | 90 | 100 |
| Atm.Cmp.Ass.Date | 180000 | 9000 | 20 | 9000 | 3000 | 3 | 3000 | 1902 | 2 |
| Atm.Cmp.Ass.Type | 180000 | 9000 | 20 | 0 | 3000 | 3 | 3000 | 300 | 10 |

Tabela 13 - composição dos índices de tamanho três

| Indice | Tipo | Chaves | Objetos | Inter | Folha | Sobre | Paginas |
|---|---|---|---|---|---|---|---|
| Ass.Cmp.Atm.Date | MX12 | 180000 | 180000 | 183 | 8999 | 0 | 9182 |
| Ass.Cmp.Atm.Date | MX23 | 9000 | 9000 | 9 | 449 | 0 | 458 |
| Ass.Cmp.Atm.Date | MXa1 | 17999 | 180000 | 26 | 1987 | 0 | 2013 |
| Ass.Cmp.Atm.Date | NX3 | 17999 | 161663 | 45 | 2731 | 0 | 2776 |
| Ass.Cmp.Atm.Date | PX | 17999 | 180000 | 186 | 11382 | 0 | 11568 |
| Ass.Cmp.Atm.Type | MX12 | 180000 | 180000 | 183 | 8999 | 0 | 9182 |
| Ass.Cmp.Atm.Type | MX23 | 9000 | 9000 | 9 | 449 | 0 | 458 |
| Ass.Cmp.Atm.Type | MXa1 | 180 | 180000 | 3 | 180 | 1012 | 1195 |
| Ass.Cmp.Atm.Type | NX3 | 180 | 146078 | 3 | 180 | 766 | 949 |
| Ass.Cmp.Atm.Type | PX | 180 | 180000 | 3 | 180 | 3899 | 4082 |
| Ass.Cmp.Doc.Date | MX12 | 9000 | 9000 | 9 | 449 | 0 | 458 |
| Ass.Cmp.Doc.Date | MX23 | 9000 | 9000 | 9 | 449 | 0 | 458 |
| Ass.Cmp.Doc.Date | MXa1 | 900 | 9000 | 1 | 104 | 0 | 105 |
| Ass.Cmp.Doc.Date | NX3 | 900 | 8087 | 3 | 135 | 0 | 138 |
| Ass.Cmp.Doc.Date | PX | 900 | 9000 | 10 | 573 | 0 | 583 |
| Ass.Cmp.Doc.Type | MX12 | 9000 | 9000 | 9 | 449 | 0 | 458 |
| Ass.Cmp.Doc.Type | MX23 | 9000 | 9000 | 9 | 449 | 0 | 458 |
| Ass.Cmp.Doc.Type | MXa1 | 90 | 9000 | 1 | 90 | 0 | 91 |
| Ass.Cmp.Doc.Type | NX3 | 90 | 8720 | 1 | 90 | 0 | 91 |
| Ass.Cmp.Doc.Type | PX | 90 | 9000 | 1 | 90 | 169 | 260 |
| Atm.Cmp.Ass.Date | MX12 | 2872 | 9000 | 3 | 151 | 0 | 154 |
| Atm.Cmp.Ass.Date | MX23 | 9000 | 180000 | 33 | 1838 | 0 | 1871 |
| Atm.Cmp.Ass.Date | MXa1 | 1902 | 3000 | 1 | 70 | 0 | 71 |
| Atm.Cmp.Ass.Date | NX3 | 1854 | 178146 | 29 | 1759 | 290 | 2078 |
| Atm.Cmp.Ass.Date | PX | 1854 | 180000 | 30 | 1854 | 3055 | 4939 |
| Atm.Cmp.Ass.Type | MX12 | 2872 | 9000 | 3 | 151 | 0 | 154 |
| Atm.Cmp.Ass.Type | MX23 | 9000 | 180000 | 33 | 1838 | 0 | 1871 |
| Atm.Cmp.Ass.Type | MXa1 | 300 | 3000 | 1 | 34 | 0 | 35 |
| Atm.Cmp.Ass.Type | NX3 | 300 | 179700 | 5 | 300 | 943 | 1248 |
| Atm.Cmp.Ass.Type | PX | 300 | 180000 | 5 | 300 | 3837 | 4142 |

### *4.5.1 Tamanho dos índices*

Novamente, assim como no caso das expressões de caminho de tamanho dois, o NX sempre ocupou menos espaço do que o PX devido às suas características estruturais. Na maioria das vezes o NX ocupou um terço do espaço ocupado pelo PX, visto que no PX armazena-se o caminho inteiro (três OIDs) enquanto o NX armazena apenas o objeto alvo (um OID). Além disto, o NX também gastou menos espaço do que o MX.

Já comparando o espaço ocupado pelo PX e pelo MX, observou-se uma alternância no desempenho dos índices. A configuração das expressões de caminho indexadas no Benchmark OO7 possibilitou realizar as seguintes comparações entre o PX e o MX:

- *Card(1)* ≤ *Card(2)* e *Card(2)* ≤ *Card(3)*

  Esta situação é observada nas expressões de caminho *Ass.Cmp.Doc.Date, Ass.Cmp.Doc.Type, Ass.Cmp.Atm.Date* e *Ass.Cmp.Atm.Type.* Quanto menor for *Card(2)* em relação a *Card(3)*, o grau de compartilhamento de objetos da classe $C_2$ para a classe $C_3$ tenderá a um. Consequentemente, o índice MX12 se comporta como um índice de taxa de repetição muito baixa. O número de nós-

intermediários do índice aumenta; por outro lado, a taxa de ocupação dos nós-folhas diminui. Esta combinação de fatores aumenta o consumo de páginas da árvore. Nesta configuração o PX gasta menos espaço do que o MX.

- *Card(1)* > *Card(2)* e *Card(2)* > *Card(3)*

  Esta situação é observada nas expressões de caminho *Atm.Cmp.Ass.Date* e *Atm.Cmp.Ass.Type*. Nesta configuração tanto o grau de compartilhamento de objetos da classe $C_1$ para a classe $C_2$ quanto o grau de compartilhamento de objetos da classe $C_2$ para a classe $C_3$ são maiores do que um. O MX12 e o MX23 passam a não degradar o gasto geral do índice MX, enquanto o índice PX tende a apresentar redundância de informação nos nós-folhas. Nesta configuração o PX gasta mais espaço do que o MX.

Em resumo, se a cardinalidade de $C_1$ for menor ou igual a de $C_2$ e se a cardinalidade de $C_2$ for menor ou igual a de $C_3$, então o tamanho do índice PX é menor do que o tamanho do índice MX. Se, entretanto, a cardinalidade de $C_1$ for maior do que $C_2$ e se a cardinalidade de $C_2$ for maior do que $C_3$, o tamanho do índice PX é maior do que o do índice MX.

#### *4.5.1.1 Consultas pontuais*

A Figura 50 e a Figura 51 apresentam o desempenho dos índices, respectivamente, nas dimensões de tempo e páginas para a seguinte consulta: "*select $C_1$.* from $C_1$ where $C_1.Atr_1.Atr_2.Atr_3$ = valor*". A Tabela 14 apresenta informações adicionais obtidas nas medições.

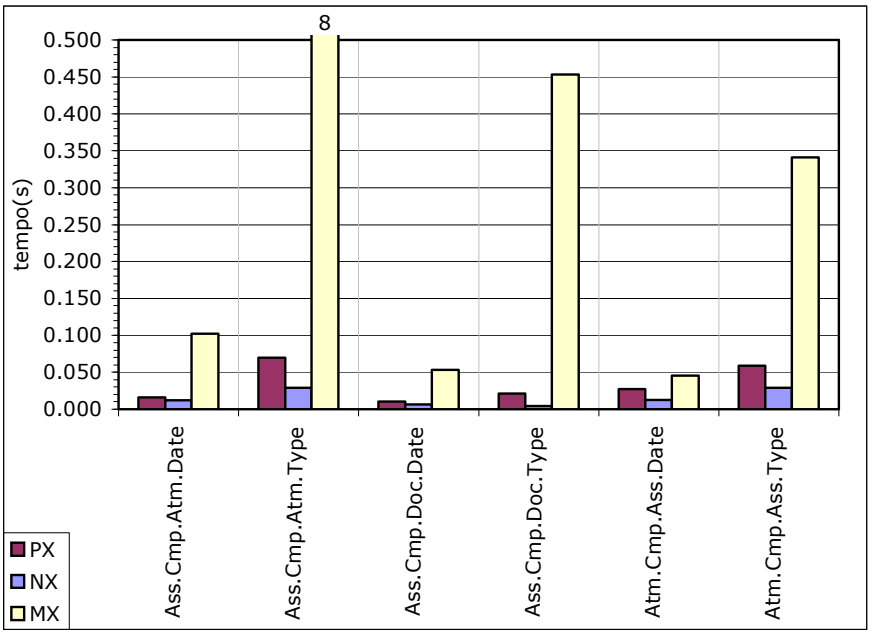



Figura 50 - Consulta pontual em expressões de tamanho três (tempo)

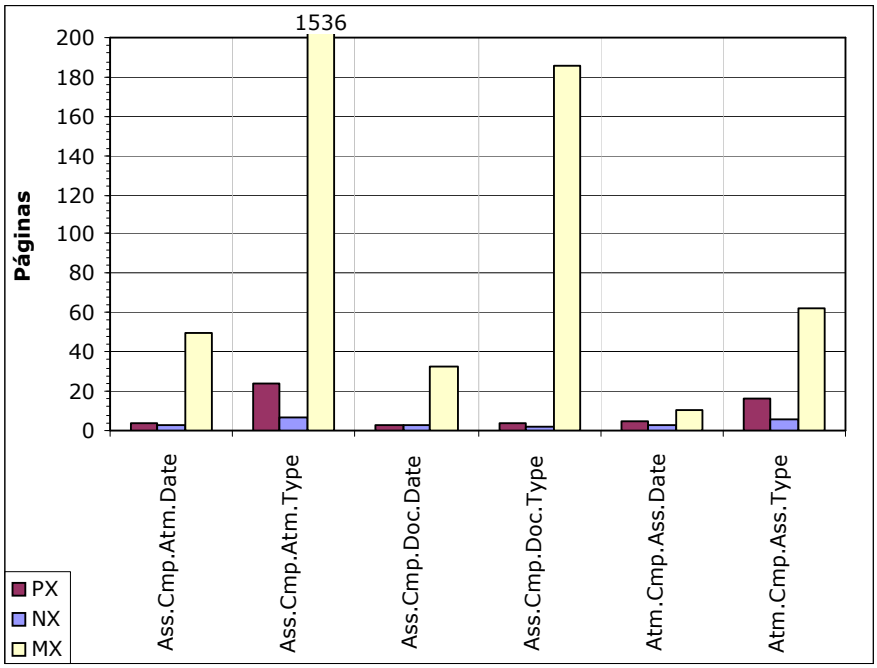



Figura 51 - Consulta pontual em expressões de tamanho três (pag. carregadas)

Tabela 14 - Consulta pontual em expressões de tamanho três

| | Intermediárias | | | Folhas | | | Sobrecarga | | | Páginas | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Config** | **PX** | **NX** | **MX** | **PX** | **NX** | **MX** | **PX** | **NX** | **MX** | **PX** | **NX** | **MX** |
| Ass.Cmp.Atm.Date | 3 | 2 | 62 | 1 | 1 | 25 | 0 | 0 | 0 | 4 | 3 | 87 |
| Ass.Cmp.Atm.Type | 2 | 2 | 4869 | 1 | 1 | 1937 | 21 | 4 | 6 | 24 | 7 | 6812 |
| Ass.Cmp.Doc.Date | 2 | 2 | 38 | 1 | 1 | 20 | 0 | 0 | 0 | 3 | 3 | 58 |
| Ass.Cmp.Doc.Type | 1 | 1 | 384 | 1 | 1 | 193 | 2 | 0 | 0 | 4 | 2 | 577 |
| Atm.Cmp.Ass.Date | 2 | 2 | 10 | 1 | 1 | 5 | 1 | 0 | 0 | 5 | 3 | 15 |
| Atm.Cmp.Ass.Type | 2 | 2 | 83 | 1 | 1 | 42 | 13 | 3 | 0 | 16 | 6 | 124 |

Nas consultas pontuais (Figura 50 e Figura 51), o índice NX apresentou desempenho superior tanto em relação ao PX quanto em relação ao MX. O índice MX teve seu desempenho influenciado predominantemente pela taxa de repetição de chaves. Quanto maior o seu valor, pior o desempenho do índice MX [5]. Entretanto BERTINO e KIM não exploram este parâmetro com valores acima de cinqüenta. Assim, quando o *Card(*1) ≤ *Card(*2), *Card(*2) ≤ *Card(*3) e a taxa de repetição de chaves apresentava-se alta (*Ass.Cmp.Atm.Type* e *Ass.Cmp.Doc.Type)*, o índice MX ficou muito aquém do PX e do NX, assemelhando-se aos resultados observados nas expressões de caminho de tamanho dois.

Quando o *Card(*1) > *Card(*2), *Card(*2) > *Card(*3) e *Comp(*3) apresentava-se baixa (*Atm.Cmp.Ass.Date)*, o desempenho do índice MX se aproximava ao do PX e do NX. Este resultado também é avaliado por BERTINO e KIM.

### *4.5.1.2 Consultas por faixa de valores*

A Tabela 15 apresenta o desempenho dos índices para a seguinte consulta por faixa de valores: "*select* $C_1$*.* from* $C_1$ *where* $C_1.Atr_1.Atr_2.Atr_3 \geq valor1$ *and* $C_1.Atr_1.Atr_2.Atr_3 \leq valor2$". Neste tipo de consulta, a taxa de repetição de chaves deixa de exercer sua influência direta e o parâmetro dominante para a análise passa a ser a relação entre as cardinalidades de $C_1$, $C_2$ e $C_3$.

Tabela 15 - Consulta por faixa de valores em expressões de tamanho três

| | Intermediárias | | | Folhas | | | Sobrecarga | | | Páginas | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Config** | **PX** | **NX** | **MX** | **PX** | **NX** | **MX** | **PX** | **NX** | **MX** | **PX** | **NX** | **MX** |
| Ass.Cmp.Atm.Date | 12 | 4 | 37556 | 557 | 134 | 14477 | 0 | 0 | 0 | 568 | 138 | 52033 |
| Ass.Cmp.Atm.Type | 2 | 2 | 41934 | 10 | 10 | 15993 | 216 | 42 | 56 | 228 | 54 | 57983 |
| Ass.Cmp.Doc.Date | 2 | 2 | 1832 | 30 | 8 | 922 | 0 | 0 | 0 | 32 | 10 | 2755 |
| Ass.Cmp.Doc.Type | 1 | 1 | 1962 | 5 | 5 | 986 | 9 | 0 | 0 | 15 | 6 | 2948 |
| Atm.Cmp.Ass.Date | 3 | 3 | 1233 | 94 | 90 | 621 | 158 | 15 | 0 | 256 | 108 | 1854 |
| Atm.Cmp.Ass.Type | 2 | 2 | 1269 | 16 | 16 | 637 | 202 | 49 | 0 | 220 | 68 | 1905 |

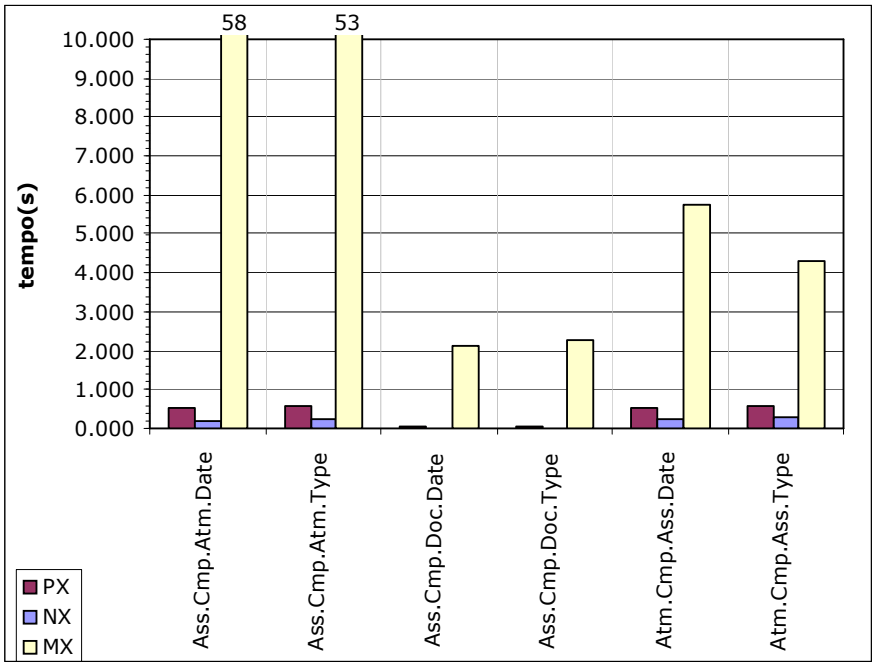



Figura 52 - Consulta por faixa de valores em expressões
de tamanho três (tempo)

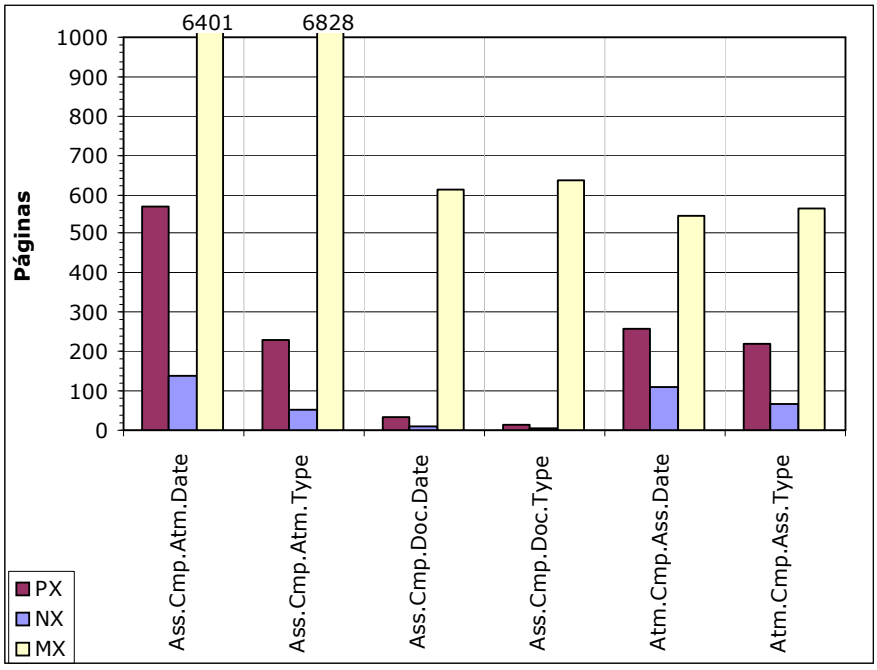



Figura 53 - Consulta por faixa de valores em expressões
de tamanho três (pag. carregadas)

Novamente foi observado que o índice NX sempre possuiu desempenho superior ao PX e MX. BERTINO e KIM concluíram que nas consultas por faixa de valores o NX é melhor do que o PX e que o PX é melhor do que o MX. Estes resultados também foram obtidos experimentalmente.

Nos resultados obtidos através do Benchmark OO7, quando a cardinalidade de $C_1$ é menor ou igual a de $C_2$ e a cardinalidade de $C_2$ é menor ou igual a de $C_3$, verifica-se que as junções intermediárias são tão intensas que inviabilizam o uso do MX nesta configuração. A diminuição da diferença do PX em relação ao MX ocorre quando a cardinalidade de $C_1$ é maior do que $C_2$ e a cardinalidade de $C_2$ é maior do que $C_3$ (*Atm.Cmp.Ass.Date* e *Atm.Cmp.Ass.Type*).

### *4.5.2 Projeções e subconsultas*

Os resultados de projeções e subconsultas obtidos para expressões de caminho de tamanho três foram semelhantes aos encontrados nas expressões de caminho de tamanho dois, como mostrado a seguir:

- Projeção final exibindo apenas a classe $C_3$: "*select* $C_1.Atr_1.Atr_2.*$ *from* $C_1$ *where* $C_1.Atr_1.Atr_2.Atr_3 = valor$" ou "*select* $C_1.Atr_1.Atr_2.*$ *from* $C_1$ *where* $C_1.Atr_1.Atr_2.Atr_3 \geq valor1$ *and* $C_1.Atr_1.Atr_2.Atr_3 \leq valor2$"

  Neste caso, o índice MX é a melhor opção pois não são necessárias junções intermediárias, bastando realizar uma consulta direta no índice MXa1.

- Projeção final exibindo as classes $C_2$: "*select* $C_1.Atr_1.*$ *from* $C_1$ *where* $C_1.Atr_1.Atr_2.Atr_3 = valor$" ou "*select* $C_1.Atr_1.*$ *from* $C_1$ *where* $C_1.Atr_1.Atr_2.Atr_3 \geq valor1$ *and* $C_1.Atr_1.Atr_2.Atr_3 \leq valor2$"

  Neste caso, há uma alternância no desempenho. Quando $Card(2) \leq Card(3)$ (*Ass.Cmp.Doc.Date, Ass.Cmp.Doc.Type, Ass.Cmp.Atm.Date* e *Ass.Cmp.Atm.Type*), o PX é o índice mais adequado; caso contrário, o índice MX é a melhor opção, visto que não são necessárias tantas junções intermediárias.

- Projeção final exibindo as classes $C_1$, $C_2$ e/ou $C_3$: "*select* $C_1.*$, $C_1.Atr_1.*$ *from* $C_1$ *where* $C_1.Atr_1.Atr_2.Atr_3 = valor$" ou "*select* $C_1.*$, $C_1.Atr_1.*$ *from* $C_1$ *where* $C_1.Atr_1.Atr_2.Atr_3 \geq valor1$ *and* $C_1.Atr_1.Atr_2.Atr_3 \leq valor2$"

  Foi verificado que, em geral, o índice PX é a melhor opção.

### *4.5.3 Comparações entre os índices para expressões de caminho de tamanho três*

A comparação entre os índices de expressões de caminho de tamanho três confirma os principais resultados obtidos nas expressões de caminho de tamanho dois. O índice NX é sempre melhor do que o PX e do que o MX. Logo, não havendo a necessidade de subexpressão na projeção da consulta, o índice NX deve ser adotado. A Tabela 16 apresenta heurísticas para a seleção de índices de acordo com a configuração da expressão de caminho.

O PX também apresenta desempenho superior em relação ao MX. Novamente, no quesito de espaço em disco, uma vez que os resultados não são absolutos, há um equilíbrio entre os índices; porém o NX tende a consumir menos espaço em relação aos demais.

Em termos conclusivos, face à superioridade do NX tanto em relação a tempo quanto em relação a tamanho, reforça-se a idéia de que um SGBDOO deve sempre ter o NX implementado. O índice PX confirmou-se como uma alternativa ao NX para contemplar as subexpressões, tendo como vantagem adicionar um menor custo de manutenção em relação ao NX.

Devido às observações da seção 4.4.4, o MX não deve ser descartado devendo ser um dos índices implementados no SBGDOO como um índice de alta versatilidade.

Tabela 16 - Heurística para escolha de índices para expressões de caminho de tamanho três

| | | Tipo | Cardinalidade | Repetição | Índice |
|---|---|---|---|---|---|
| **Projeção** | $C_1.*$ | Pontual / Faixa | $C_1 \leq C_2$ e $C_2 \leq C_3$ | Baixa/Alta | **NX** |
| | | | $C_1 > C_2$ e $C_2 > C_3$ | Baixa/Alta | **NX** |
| | $C1.*$, $C_1.Atr_1.*$ | Pontual / Faixa | $C_1 \leq C_2$ e $C_2 \leq C_3$ | Baixa/Alta | **PX** |
| | | | $C_1 > C_2$ e $C_2 > C_3$ | Baixa/Alta | **PX** |
| | $C_1.Atr_1.*$ | Pontual / Faixa | $C_2 \leq C_3$ | Baixa/Alta | **PX** |
| | | | $C_2 > C_3$ | Baixa/Alta | **MX** |

## 5. Conclusões

Índices desempenham um papel fundamental no acesso a grandes volumes de dados. São estruturas de acesso rápido indispensáveis em SGBDs. Com a incorporação das estruturas de representação do modelo OO tanto nos SGBDOO quantos nos SGBDOR, novos índices precisam ser desenvolvidos.

Embora vários índices tenham sido propostos ao longo da década de 90 para atender aos novos requisitos de representação semântica, poucos trabalhos descrevem seu desempenho mediante diferentes situações de uma base de dados. Os trabalhos que analisam o desempenho o fazem através de modelos de simulação utilizando funções de custo. Nesta dissertação, procurou-se analisar experimentalmente o comportamento destes índices, já que modelos de custo são muitas vezes simplificados.

Foram implementados e avaliados diferentes índices para hierarquia de classes e expressões de caminho. As principais contribuições desta dissertação foram: (i) desenvolvimento de um ambiente para medições experimentais de índices; (ii) apresentação de resultados experimentais que comprovam alguns resultados analíticos

e (iii) apresentação de alguns comportamentos dos índices até então desconhecidos nas análises das simulações.

Através do *benchmark* OO7 foram realizadas medições experimentais sobre o GOA++ (Gerente de Objetos Armazenados da COPPE/UFRJ) tanto para os índices SCI (índice para cada classe) e CHI (índice para hierarquia de classe), referentes à hierarquia de classes, quanto para os índices MX (multi-índice), NX (índice aninhado) e PX (índice de caminho), relativos às expressões de caminho.

Na comparação entre os índices para hierarquia de classes observou-se que o CHI nem sempre apresentava os melhores resultados. Ficou evidenciado, no entanto, que para consultas pontuais, independentemente de qualquer outro parâmetro, o índice CHI é sempre a melhor opção.

No caso de consultas que envolvam pequena parte da hierarquia, foi confirmado o melhor desempenho do SCI. Já para consultas por faixa de valores observou-se a necessidade de uma análise mais cuidadosa, visto que o desempenho dos índices variam de acordo com a taxa de repetição das chaves. Esses resultados não se apresentavam 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. Entretanto, neste tipo de consulta os índices são sensíveis à taxa de repetição das chaves e à faixa de valores especificada na consulta.

Finalmente, os resultados apontaram 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 oito classes.

Na comparação dos índices para expressões de caminho observou-se a predominante superioridade do NX face aos demais índices. Logo, sempre que não haja a necessidade da avaliação de uma subexpressão na projeção final de uma consulta, o índice NX deve ser adotado. Comparando-se o PX ao MX em relação ao tempo, tem-se que o PX é o recomendável. No tocante ao quesito de espaço em disco, observa-se um equilíbrio entre os dois índices na medida em que os resultados não são absolutos.

Considerando-se o destaque da superioridade do NX nas dimensões tempo e tamanho, sugere-se que um SGBDOO tenha sempre o NX implementado. Devido à sua vantagem referente a um menor custo de manutenção em relação ao NX, o índice PX pode ser

usado como uma alternativa para contemplar as subexpressões. Na prática, basta ter um dos dois índices como acelerador no processamento das consultas.

Apesar de tudo o que foi observado, o MX não deve ser descartado, pois é um índice de baixo custo de manutenção. Embora seja inadequado em muitas consultas de expressões de caminho, traz a sua versatilidade em diferentes tipos de consultas. Além disto o MX pode ser aplicado como um mecanismo de gerência de esquema e integridade de dados como, por exemplo, manutenção de integridade referencial, podendo servir, ainda, como um acelerador para as atualizações dos índices PX e NX.

Mais importante do que apresentar os resultados experimentais de hierarquia de classes e expressões de caminho, até certo ponto conhecidos pela literatura, o trabalho apresentado tem como resultado maior a medição, de fato, de todos estes índices baseados em árvore B+. Isto somente foi obtido mediante a criação de um ambiente para medições experimentais de índices. Este ambiente experimental não apenas possibilita as medições como também é suficientemente robusto para o desenvolvimento de novas estruturas de indexação.

Esta robustez do ambiente somente foi alcançada mediante a modelagem do sistema de indexação, que soube separar bem o método de acesso, a estrutura de indexação e as regras de integridade dos índices. Qualquer novo índice a ser implementado pode agora ser comparado diretamente aos índices “básicos” da orientação a objetos, como também podem ser identificados focos críticos para exploração de melhoria de desempenho, seja ela estrutural, no que tange às estruturas de indexação, ou relacionada aos algoritmos utilizados para implementação dos índices.

Assim, o ambiente proposto fornece subsídios suficientes para: (i) explorar novas estruturas para hierarquia de classes, como as estruturas espaciais, tendo uma diretriz básica de desempenho definida pelo SCI e CHI que estas novas estruturas devem superar; (ii) explorar novas estruturas para expressões de caminho, no sentido de ampliar o desempenho dos índices, bem como diminuir o custo de suas atualizações; (iii) realizar medições de composições de índices para hierarquia de classes com expressões de caminho versus a utilização única de índices híbridos.

O trabalho apresentado nesta dissertação pode se constituir como infra-estrutura básica para o desenvolvimento de outras pesquisas. A seguir são sugeridas áreas de pesquisa que merecem ser exploradas:

**Extensões Espaciais**: Devido à separação dos métodos de acesso, das estruturas de indexação e das regras de integridade dos índices, pode-se com simplicidade incorporar ao ambiente as estruturas de dados espaciais como, por exemplo, a Árvore-R [40]. Assim pode-se estabelecer um paralelo entre desempenho das estruturas espaciais versus o desempenho das estruturas baseadas em árvore B+. De fato, a árvore-R já vem sendo implementada por Victor Almeida (COPPE/UFRJ) a partir do ambiente apresentado neste trabalho, o que reforça sua generalidade e flexibilidade.

**Disponibilização do ambiente para uso geral**: Disponibilização do ambiente experimental para avaliação de índices como ferramenta didática. A idéia é separar o ambiente experimental do sistema GOA++ através da criação de uma interface, de tal modo que o pesquisador possa criar novas ou refinar as estruturas existentes ao comparar o desempenho destas com uma base já gerada, ou ainda variar parâmetros, como o tamanho do cache, o tamanho das páginas, dentre outros.

**Extensões Paralelas**: Estender o ambiente de modo que ele funcione em bases de dados fragmentadas como ocorre no Par GOA. Através deste mecanismo, pode-se avaliar o grau de paralelismo dos algoritmos utilizados nas estruturas de indexação [41] tanto para hierarquia de classes quanto para expressões de caminho.

## 6. Referências Bibliográficas


[1] BERTINO, E., FOSCOLI, P. “Index Organizations for Object-Oriented Database Systems”, *IEEE Transactions on Knowledge and Data Engineering*, v. 7, n. 2, pp. 193-209, Abr. 1995.

[2] CATTELL, R., BARRY, D. *Object Database Standard ODMG 2.0*. California, USA, Morgan Kaufmann Publisher Inc., 1997.

[3] ÖZSU, M., BLAKELEY, J. “Query Processing in Object-Oriented Database Systems“. In: Kim, W. (ed), *Modern Database Systems: The Object Model, Interoperability and Beyond*, chapter 8, New York, USA, Addison-Wesley Publishing Company, 1995.

[4] KEMPER, A., MOERKOTTE, G. “Physical Object Management”. In: Kim, W. (ed), *Modern Database Systems: The Object Model, Interoperability, and Beyond*, chapter 9, New York, USA, Addison-Wesley Publishing Company, 1995.

[5] BERTINO, E., KIM, W. “Indexing techniques for queries on nested objects”, *IEEE Transactions on Knowledge and Data Engineering*, v. 1, n. 2. pp.196-214, 1989.

[6] KIM, W., KIM, K., DALE, A. “Indexing techniques for object-oriented databases“. In: *Object Oriented Concepts, Databases and Applications*, chapter 15, New York, USA, Addison-Wesley Publishing Company, 1989.

[7] KILGER, C., MOERKOTTE, G. “Indexing Multiple Sets”. In: Proceedings of the 20th *VLDB* Conference, pp.175-202, Santiago, Chile, 1994.

[8] BRAUMANDI, R., CLAUSSEN J., KEMPER A. “Evaluating Functional Joins Along Nested Reference Sets in Object-Oriented and Object-Relational Databases”, In: Proceedings of the 24th *VLDB* Conference, pp.110-121, EUA, 1998.

[9] LOW, C., OOI, B., LU, H. “H-Trees: A Dynamic Associative Search Index for OODB“. In: *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pp. 134-143, California, USA, 1992.

[10] CHAN, C., GOH, C., OOI, B. “Indexing OODB Instances Based on Access Proximity”. In: *IEEE International Conference on Data Engineering*, pp. 14-21, Birmingham, England, Abr. 1997.

[11] STEHLING, R., NASCIMENTO, M. “Métodos de Acesso em Bancos de Dados Orientados a Objetos“. *XIII Simpósio Brasileiro de Banco de Dados*, Maringá, Brasil, Out. 1998.

[12] MUECK, POLASCHEK. “The Multikey type Index for Persistent Object Sets“. In: *IEEE International Conference on Data Engineering*, pp. 22-31, Birmingham, England, Abr. 1997.

[13] BERTINO, E., FOSCOLI, P. “On Modeling Cost Functions for Object Oriented Databases”, *IEEE Transactions on Knowledge and Data Engineering*, v. 9, n. 3, pp. 500-508, Mai/Jun. 1997.

[14] Rational Software Corporation. *OMG Unified Modeling Language Specification*, versão 1.3, http://www.rational.com, Jun. 1999.

[15] BOOCH, G. *Object-Oriented Analysis and Design with Applications,* 2 ed. California, USA, The Benjamin/Cummings Publishing Company Inc., 1994.

[16] BOULLOSA, J.R., XEXÉO, G.X. “Incerteza em Bancos de Dados: Tipo de Dados Nebulosos no GOA++”. In: *Anais do XIV Simpósio Brasileiro de Banco de Dados*, pp. 205-220, Florianópolis, Out. 1999.

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

[18] OUALLINE, S. “Heap Checking”, *Dr. Dobb's Journal*, Nov.1993.

[19] TANENBAUM, A.S. *Modern Operation System*. New Jersey, USA, Prentice-Hall International, 1992.

[20] KERNIGHAN, B., RITCHIE, D. “Pointers and Arrays”. In: *The C Programming Language*, chapter 5, New Jersey, USA, Prentice-Hall International, 1988.

[21] Borland Corporation. “Memory Management”. In: *Borland C++ Programmer's Guide.* Version 2.0, California, USA,. Borland Press, 1991.

[22] NELSON, T. “Finding Run-Time Memory Errors”, *Dr. Dobb's Journal*, Nov. 1993.

[23] CORMEN, T. H., LEISERSON, C. E., RIVEST, R. L. *Introduction to Algorithms*. New York, USA, McGraw-Hill Book Company/ London, England, *MIT Press*, 1989.

[24] CAREY, M.J., DEWITT, D., NAUGHTON, J. “The OO7 Benchmark”. In: *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pp.12-21, Washington, USA, May. 1993.

[25] CAREY, M. J., DEWITT, D. J., NAUGHTON, J., et al. “The BUCKY Object-Relational Benchmark”, In: *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pp. 135-146, Arizona, USA, May. 1997.

[26] CATTELL, R.G.G. *Object data management: object-oriented and extended relational database systems*. California, USA, Addison-Wesley Publishing Company, 1994.

[27] Shrufi, A. “Performance clustering policies in object bases”. In: *Proceedings of the 3rd Conf on Information and Knowledge management*, novembro, EUA, pp.80-87, 1994.

[28] SU, S. Y. W., Ranka, S., HE, X. “Performance Analysis of Parallel Query processing Algorithms for Object-Oriented Databases”. In: *Technical Report Database Systems Research and Development Center*, Univ. of Florida, EUA, 1999.

[29] Gardarin, G., Gruser, J., Tang. Z, “Cost Based Selection of Path Expression Processing Algorithms in Object Oriented Databases”. In: *Proc. 22 VLDB* Bombay, India, pp.390-401, 1996.

[30] Gardarin, G., Sha, F., Tang, Z., “Calibrating the Query Optimizer Cost Model of IRO-DB an Object Oriented Federated Database System”. In: *Proc. 22 VLDB*, Bombay, India, pp.378-389, 1996.

[31] Ozsu, M. T., Voruganti, K., Unrau, R. C. “An Asynchronous Avoidance-Based Cache Consistency Algorithm for Client Caching DBMSs”. In: *Proc. 22 VLDB* Bombay, India, pp.440-451, 1999.

[32] MAURO, R.C., ZIMBRÃO, G., BRÜGGER, T.S., et al. “GOA++: Tecnologia, Implementação e Extensões aos Serviços de Gerência de Objetos“. In: *Anais do XII Simpósio Brasileiro de Banco de Dados*, pp.272-286, Fortaleza, Out. 1997.

[33] MEYER, L.A.V.C., MATTOSO, M.L.Q. "Parallel query processing in a shared-nothing object database server" In: *Anais do 3rd International Meeting on Vector and Parallel Processing (VECPAR’98)*, pp.1007-1020, Porto, Portugal, jun 1998.

[34] LIMA, F., MATTOSO, M.L.Q. "Performance Evaluation of Distribution in OODBMS: a Case Study with O2". In: *Anais da IX International Conference on Parallel & Distributed Computing Systems,* ISCA/IEEE, pp.720-726, Dijon, França, set 1996.

[35] OGASAWARA, E.S., MATTOSO, M.L.Q. *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, 1999.

[36] BAIÃO, F.A., MATTOSO, M.L.Q., ZAVERUCHA, G. "Horizontal Fragmentation In ODMSs: Some Issues and Performance Evaluation" to be published in the *19th IEEE International Performance, Computing, and Communications Conference - IPCCC 2000*, IEEE CS Press, Phoenix, Arizona, feb. 2000.

[37] MAURO, R.C., MATTOSO, M.L.Q. “Integração de LPOO e BDOO: Uma Experiência com JAVA e GOA++“. In: *Anais do XIII Simpósio Brasileiro de Banco de Dados*, SBC, pp.169-184, Maringá, Out. 1998b.

[38] OGASAWARA, E.S., MATTOSO, M.L.Q. “Uma avaliação Experimental sobre Técnicas de Indexação em Bancos de Dados Orientados a Objetos“. In: *Anais do XIV Simpósio Brasileiro de Banco de Dados*, pp. 285-298, Florianópolis, Out. 1999b.

[39] LEE, W.C., LEE, D.L.. “Path Dictionary: A New Access Method for Query Processing in Object-Oriented Databases“, *IEEE Transactions on Knowledge and Data Engineering*, v. 10, n. 3, pp. 371-388, 1998.

[40] LIMA, A.A., MATTOSO, M.L.Q., ESPERANÇA, C. “Extensão de um SGBDOO com dados espaciais“. In: *Anais da XXIV Conferência Latino Americana de Informática*, Equador, 1998.

[41] DEWITT, D. J., NAUGHTON, J. F., SHAFER, J. C., VENKATARAMAN S. “Parallelizing OODBMS traversals: a performance evaluation”. In: *VLDB Journal*, v.5, n.1, pp.3-18, 1996.