3
Descrição da implementação: o sistema Clairvoyant

Existem várias maneiras de implementar um repositório de medição de acordo com os requisitos expostos na seção de visão e escopo deste trabalho. O objetivo desse capítulo é descrever uma dessas possibilidades, no intuito de demonstrar a viabilidade desse conjunto de requisitos que escolhemos previamente para o repositório e inspirar outras idéias para implementá-los. Batizamos essa implementação de sistema *Clairvoyant*.

Neste capítulo, explicaremos detalhadamente essa implementação de repositório de medição. Já que optamos por utilizar um meta-modelo de medição como maneira de dispor de capacidade de evolução do modelo de dados, começamos por explicá-lo. Em seguida, detalhamos o modelo de consulta, que nos permite fazer consultas ao repositório sem ter que saber detalhes do meta-modelo. Depois disso, descreveremos, em detalhes, como ocorre a operacionalização das funcionalidades oferecidas pelo repositório no contexto dos modelos de consulta e medição utilizados neste trabalho. Por fim, comentaremos sobre alguns aspectos de arquitetura e implementação do sistema que julgamos relevantes.

3.1.
Estrutura do sistema

A estrutura do modelo é composta pelo meta-modelo de medição e o modelo de consultas, que representam, respectivamente, a maneira que os dados são armazenados no repositório e a maneira que podemos resgatar as informações nele contidas. As interações entre esses modelos, que ocorrem para disponibilizar as funcionalidades do repositório, são inúmeras. Por isso, a interface entre eles tem de ser bem definida. Nota-se essa interação principalmente ao lidarmos com as funcionalidades relacionadas ao macro-processo de consulta, pois sua operacionalização é fortemente influenciada pela estrutura do modelo de consulta. Já a estrutura do meta-modelo de medição afeta diretamente a operacionalização

dos macro-processos de manutenção dos dados de medição e de importação de dados. Ele pode afetar os outros indiretamente através do modelo de consulta.

A grosso modo podemos dizer que o modelo de medição está situado num nível de abstração mais baixo que o modelo de consulta. Isso se deve ao fato de que o modelo de consulta referencia o modelo de medição, mas o modelo de medição não referencia o modelo de consulta. Além disso, o modelo de consulta encapsula o uso do meta-modelo de medição para o macro-processo de consulta: uma modificação no meta-modelo de medição não influencia o funcionamento do sistema se a interface entre os modelos não for modificada.

3.1.1. Meta-modelo de medição

Como já foi previamente mencionado, dados de medição de software são armazenados em repositórios seguindo modelos de medição. O modelo de medição adotado por um repositório pode ser, na verdade, um meta-modelo de medição - um modelo de modelos de medição. É justamente esse o caso do repositório Clairvoyant, onde o modelo de medição em uso num dado momento é representado como uma instância do meta-modelo empregado pelo sistema. Mais precisamente, a idéia principal do sistema Clairvoyant é implementar a capacidade de evolução por meio do uso da indireção e de meta-dados em seu modelo de medição.

Isso é feito atribuindo a cada entidade existente no repositório uma instância de uma estrutura correspondente a uma entidade. Analogamente, a cada atributo também é atribuída uma estrutura correspondente. Já a associação de entidades e atributos se daria por instâncias de uma estrutura de associação. Como conseqüência dessa abordagem para modelar os dados de medição, o modelo subjacente teria que atravessar vários níveis de indireção para poder apresentar os dados de uma entidade. Isso implica uma penalidade no desempenho do sistema e, mais importante, um aumento considerável na complexidade da composição de consultas aos dados de medição do sistema.

Podemos contrastar esse modo de operação com aquele que provavelmente é a maneira mais simples e mais intuitiva para representar um modelo de medição: representar entidades como tabelas ou classes, e atributos colunas ou campos de

classes. Nesse contexto, a evolução do modelo de medição se torna mais trabalhosa e propensa a erros, por implicar uma freqüência maior de mudança no esquema do banco de dados subjacente. Além disso, essa abordagem tem outras desvantagens como, por exemplo, não possibilitar uma visão histórica do modelo de medição, nem versionamento dos modelos, pois nesse caso o cessar da coleta de uma medida implicaria a perda das informações históricas associadas a ela.

3.1.1.1. Representação conceitual

O meta-modelo de medição se encontra implementado no sistema Clairvoyant em dois níveis: sua representação conceitual, equivalente ao modelo de classes que representa o meta-modelo no código-fonte, e a sua representação relacional, que corresponde aos dados armazenados no mecanismo de persistência atualmente em uso pelo sistema: um banco de dados relacional. A representação conceitual do meta-modelo de medição é representada pela figura 2, e seus elementos são:

- **Entidade (*Entity*):** representa o conceito de *entidade* ou *ente medido* – ou seja, um possível objeto de medição. Seu nome deve ser único no repositório de medição (como meio de identificação inequívoca), assim como a sua descrição. Exemplos típicos de entidades em repositórios de medição incluem produtos, projetos, pessoas, entre outros.
- **Relação (*Relation*):** reflete o conceito de relação entre entidades. Referencia duas entidades, uma de origem e outra de destino (que pode ser a mesma, denotando uma relação reflexiva), cada uma com uma cardinalidade, que pode ser *única* ou *múltipla*. Assim como a entidade, ela tem um nome que deve identificá-la inequivocamente no repositório, assim como uma descrição.
- **Tipo de atributo (*AttributeType*):** reflete o conceito de tipo de atributo, armazenando informações de nome, descrição e tipo de escala do tipo de um atributo.

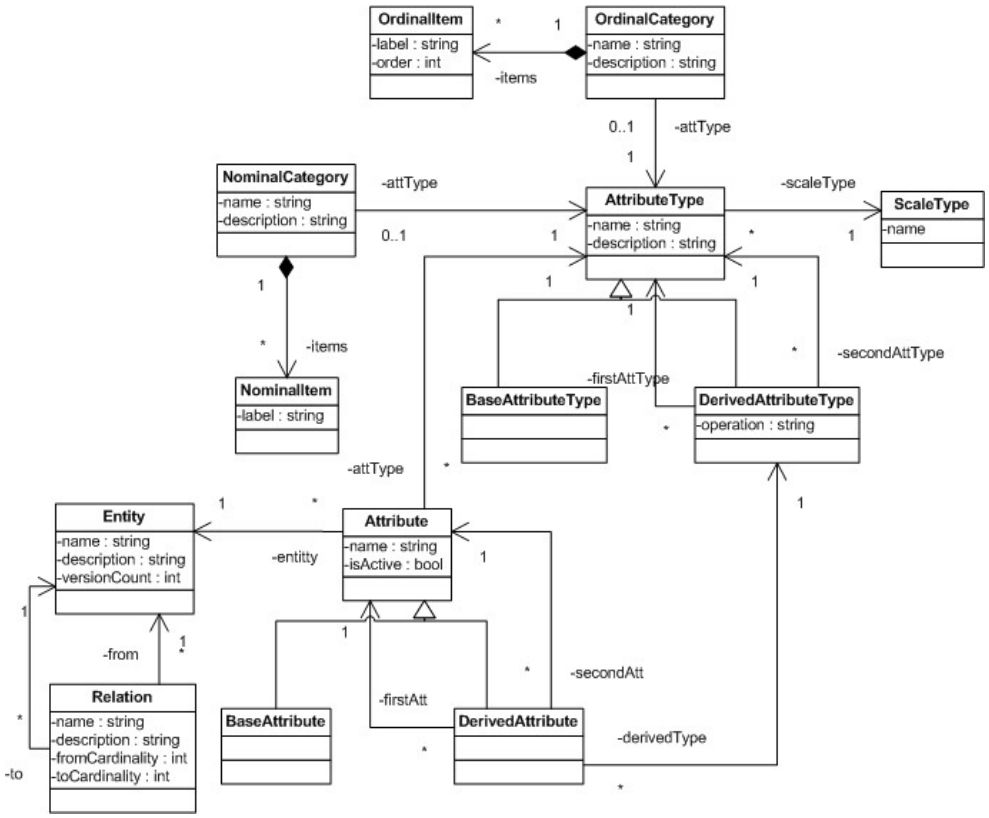

Figura 2 – Representação conceitual do meta-modelo de medição do sistema Clairvoyant, inspirada por (Harrison, 2004) e (Palza et al., 2003)

- **Tipo de atributo-base (*BaseAttributeType*):** reflete o conceito de tipo de atributo-base, que é um tipo de atributo correspondente a dados de medição cujo valor está armazenado diretamente no repositório, não sendo função de outros valores.
- **Tipo de atributo derivado (*DerivedAttributeType*):** reflete o conceito de tipo de atributo-base, que é uma especialização de tipo de atributo. Indica quais tipos de atributo e qual operação de derivação são utilizados para gerar o seu valor derivado.
- **Atributo (*Attribute*):** reflete o conceito de atributo conforme explicado previamente. Seu nome deve ser único no repositório de medição (como meio de identificação inequívoca do atributo), assim como a sua descrição e qual o seu tipo de escala.
- **Atributo-base (*BaseAttribute*):** reflete o conceito de atributo-base, uma especialização do atributo. Seu valor está armazenado diretamente no repositório.

- **Atributo derivado (*DerivedAttribute*):** reflete o conceito de atributo derivado, uma especialização do atributo. Seu valor é obtido através de um cálculo que referencia valores de outros atributos associados à mesma entidade. Estes podem, por sua vez, ser atributos-base ou derivados. Podemos citar como exemplo de atributo derivado a densidade de defeitos de um produto como a divisão do número de defeitos desse produto pela quantidade de linhas de código do mesmo.
- **Categoria nominal (*NominalCategory*):** representa uma categoria nominal, é referenciada por atributos de tipo de escala nominal.
- **Ítem de categoria nominal (*NominalItem)*:** representa os possíveis itens associado a uma categoria nominal, referenciando a categoria à qual ele pertence e o identificador do item, denominado *rótulo*.
- **Categoria ordinal (*OrdinalCategory*):** representa uma categoria ordinal, é referenciada por atributos de tipo de escala ordinal
- **Item de categoria ordinal (*OrdinalItem*):** analogamente ao item nominal, representa os possíveis itens associado a uma categoria ordinal. Além da referência à sua categoria associada e o seu rótulo, o item ordinal guarda a informação de sua ordem na categoria.
- **Tipo de escala de atributo (*ScaleType)*:** é uma extensão dos tipos de escala de medição apresentados previamente. Essa extensão será explicada em detalhe na próxima seção.

3.1.1.2. Tipos de escala

Como foi visto anteriormente, as medições de software se encaixam em certos tipos de escala, cada uma com um conjunto semântica particular, denotando que tipos de comparações podem ser feitas entre elas e quais operações matemáticas são definidas para elas. Como o sistema Clairvoyant utiliza esse tipo de informação, é natural que essa informação esteja representada em seu meta-modelo de medição. Mais precisamente, o repositório representa os tipos de atributos que correspondem aos tipos de escala de medição encontrados na literatura de teoria da medição (Zuse, 1997), com alguns elementos acrescidos

para aumentar as possibilidades semânticas das escalas dos dados de medição. Os tipos de escala utilizados no sistema Clairvoyant são:

- **Nominal (*Nominal*):** representa o tipo de escala nominal encontrado na literatura de tipos de escala de medição
- **Ordinal (*Ordinal*):** representa o tipo de escala ordinal encontrado na literatura.
- **Intervalo (*Interval*):** representa o tipo de escala intervalo encontrado na literatura.
- **Razão (*Ratio*):** representa o tipo se escala razão encontrado na literatura.
- **Informativo (*Informative*):** representa informações que não servem para processamento, mas que podem ter alguma utilidade para um usuário humano (por exemplo, a descrição de um projeto).
- **Identificador (*Identifier*):** é um caso especial de atributo informativo, para podermos identificar uma entidade inequivocamente a partir do valor de um atributo dessa mesma entidade – já que o usuário do repositório não tem acesso ao identificador interno de uma entidade (propositalmente, para ocultar esse detalhe de implementação do usuário).
- **Temporal (*Temporal*):** é um caso especial do tipo intervalo utilizado para dotar as variáveis temporais de uma semântica própria, visando à implementação futura de consultas com operadores temporais.
- **Absoluto (*Absolute*):** é um caso especial do tipo de escala *razão*. Corresponde a um número inteiro, e é usado para dotar dessa semântica específica atributos que são necessariamente números inteiros por definição (por exemplo, contagens de quantidade). É importante observar que em algumas referências na literatura da teoria de medição o tipo de escala absoluto também é considerado um tipo de escala de medição. (Zuse, 1997).

3.1.1.3. Representação relacional

Como as informações do modelo de medição apresentada pelo modelo conceitual são, na verdade, armazenadas num banco de dados segundo um modelo relacional, é importante apresentarmos também esse modelo, representado pela figura 3 e constituído pelos seguintes elementos:

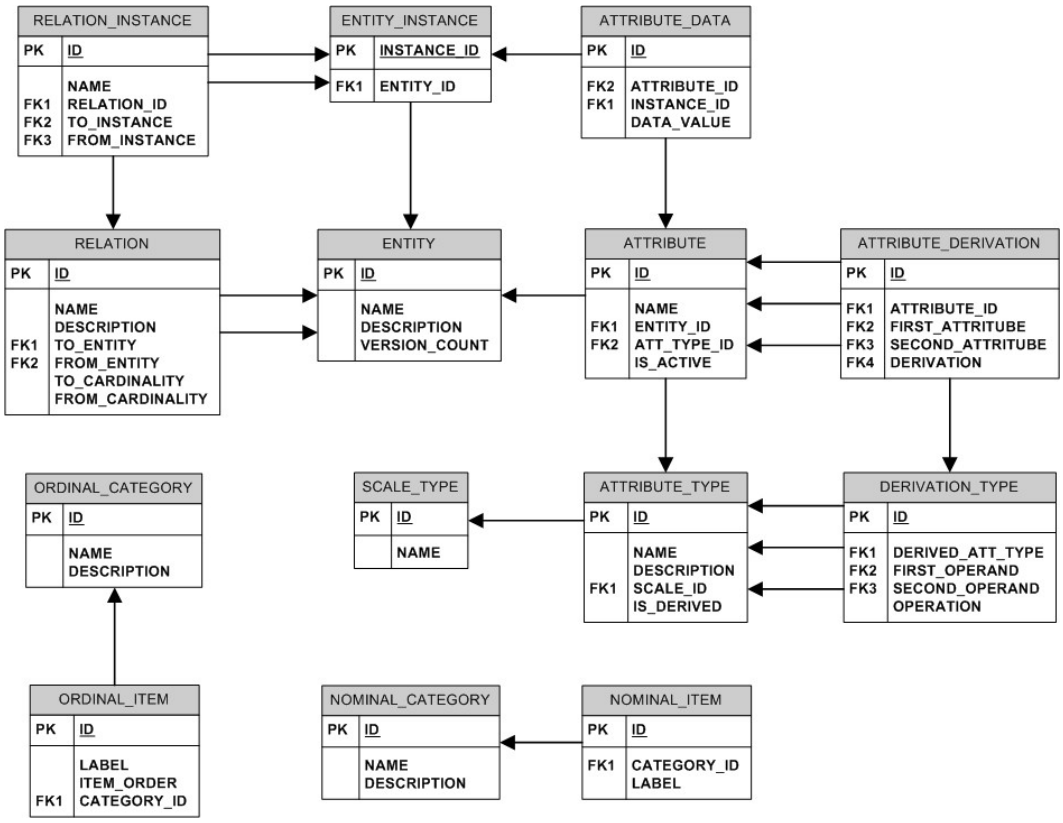

Figura 3 – representação relacional do meta-modelo de medição do sistema Clairvoyant

- **ENTITY:** tabela que lista as entidades do modelo de medição, analogamente ao modelo conceitual.
- **ENTITY_INSTANCES:** tabela que um mantém um registro de cada instância de entidade e seu tipo de entidade associada.
- **INSTANCES_<nome da entidade>:** visão que representa o registro das instâncias de entidade de um dado tipo. É formada pela projeção de ***ENTITY_INSTANCES*** para um tipo de entidade.

- **RELATION:** tabela que representa tipos de relação que ligam duas entidades do modelo de medição, analogamente ao modelo conceitual.
- **RELATION_INSTANCE:** tabela que representa instâncias de um tipo de relação. Cada registro referencia um tipo de relação e as instâncias de entidade envolvidas.
- **LINK_<nome da relação>:** visão que representa as instâncias de um tipo de ligação de entidade, por projeção da tabela ***RELATION_INSTANCE*** para um tipo de relação.
- **ATTRIBUTE_TYPE:** tabela que representa tipos de atributos, analogamente ao modelo conceitual. Referencia o tipo de escala do tipo de atributo e indica se o tipo corresponde a um atributo-base ou derivado.
- **DERIVATION_TYPE:** tabela que representa as informações necessárias para que tipos de atributos exponham as suas formas de derivação. Ela referencia o atributo derivado, referencia os atributos-fonte dessa derivação e guarda informação sobre a operação de derivação.
- **ATTRIBUTE:** tabela que representa atributos associados a entidades, analogamente ao modelo conceitual. Referencia o tipo de atributo correspondente e guarda um indicador de atividade, que é ativado e desativado segundo se estabelece ou rompe a ligação do atributo em questão à entidade.
- **ATTRIBUTE_DERIVATION:** tabela que representa as informações necessárias para que atributos derivados gerem o seu valor derivado. Referencia um tipo de derivação do atributo (que nos indica a operação de derivação) e os atributos-fonte usados na derivação, que devem estar associados à mesma entidade em questão.
- **ATTRIBUTE_DATA:** tabela onde os dados de medição são efetivamente armazenados. Eles são como armazenados como cadeias de caracteres para poder armazenar qualquer tipo de valor,

tendo o tipo inferido pela semântica da escala do tipo associado ao atributo.

- **DATA_<nome da entidade>_<nome do atributo>:** visão que projeta os dados de medição provenientes da tabela ***ATTRIBUTE_DATA*** para um atributo específico associado a uma entidade específica.
- **NOMINAL_CATEGORY:** tabela que representa categorias de variáveis de tipo nominal, analogamente ao modelo conceitual.
- **NOMINAL_ITEM:** tabela que representa itens de categorias nominais, analogamente ao modelo conceitual.
- **ORDINAL_CATEGORY:** tabela que representa categorias de variáveis de tipo nominal, analogamente ao modelo conceitual.
- **ORDINAL _ITEM:** tabela que representa itens de categorias nominais, analogamente ao modelo conceitual.
- **BASE_<nome da entidade>:** visão que contém a parte não-derivada da entidade, ou seja, uma projeção de todos os seus atributos-base. Ela é construída como uma junção de todos os atributos-base associados à entidade (as visões ***DATA_<nome da entidade>_<nome do atributo>*** correspondentes à entidade), tomando como eixo principal da junção a visão ***INSTANCES_<nome da entidade>***.
- **DERIVED_<nome da entidade>:** visão que contém a parte derivada da entidade, ou seja, uma projeção de todos os seus atributos derivados. Ela é construída pela formulação de cada atributo derivado, desdobrando-o em operações de derivação até que consigamos expressar todos os atributos derivados em termos de atributos-base (representados pelas visões ***DATA_<nome da entidade>_<nome do atributo>***), fazendo uma junção destes tomando como eixo principal da junção a visão ***INSTANCES_<nome da entidade>***.
- **<nome da entidade>:** visão que representa a entidade completa – a combinação das visões base e derivadas da entidade. Ela é construída como uma junção das visões ***BASE_<nome da entidade>***

e ***DERIVED_<nome da entidade>***, tomando como eixo principal da junção a visão ***INSTANCES_<nome da entidade>***.

Convém notar que existem algumas visões que devem sempre ser atualizadas ao atualizarmos uma entidade, devido à sua associação ou dissociação a um atributo. Essas visões são:

- ***VERS_ENT_BASE_<nome da entidade>_<versão da entidade>***
- ***VERS_ENT_DERIVED_<nome da entidade>_<versão da entidade>***
- ***VERS_ENT_ <nome da entidade>_<versão da entidade>***

Elas são, respectivamente, as variantes versionadas de ***BASE_<nome da entidade>***, ***DERIVED_<nome da entidade>*** e ***<nome da entidade>***. Toda vez que se altera uma entidade pelo ligamento ou desligamento de uma associação com um atributo, é gerada uma nova versão da entidade, e a variante não versionada (que representa o estado atual da entidade no modelo) é feita apontar para essa nova versão. Isso quer dizer que a versão mais recente (ou seja, com maior número de versão) dessas visões deve ser igual à sua variante não versionada.

3.1.2. Modelo de consulta

O modelo de consulta é a representação das diferentes possibilidades de consulta no repositório aos dados de medição, sobre o qual se baseiam todas as funcionalidades do repositório relacionadas à consulta de dados de medição. Como o meta-modelo de medição – nossa solução para a evolução – passa pelo uso de vários níveis de indireção, a construção de consultas diretamente sobre ele seria extremamente complexa, tornando essa tarefa demorada e propensa a erros. O modelo de consulta aumenta a usabilidade do repositório ajudando a ocultar essa complexidade. Ele o faz traduzindo uma consulta de um modelo simplificado que apresentaremos a seguir numa consulta para execução de acordo com o modelo na qual os dados de medição estão efetivamente armazenados, com vários níveis de indireção.

Como será visto em breve, a inspiração principal do modelo é, sem dúvida, o modelo relacional para bancos de dados, pois o modelo é fortemente influenciado pela álgebra relacional e pela linguagem SQL. Também foram incorporadas algumas extensões para facilitar a análise estatística, notadamente a possibilidade de contagem de freqüência em agregações de dados. Imagina-se poder incorporar em breve ao modelo de consultas, funcionalidades do modelo dimensional de bancos de dados (Inmon, 2005) e funcionalidades de consultas em bancos de dados temporais (Snodgrass, 1995).

3.1.2.1. Tipos de consulta

O modelo de consultas do sistema Clairvoyant possibilita a execução de três tipos de consulta: consulta de seleção, de contagem e de distribuição de freqüência.

A consulta de seleção é uma consulta onde o objetivo é gerar um conjunto de resultados de uma parte do universo obedecendo a um conjunto de característica, e mostrar uma parte dessa parte. Nota-se que ela se caracteriza pela ausência de agregações de dados.

A consulta de contagem é uma consulta onde evidenciamos informações sobre quantidades de partes do universo que existem ou obedecem a um conjunto de características. Para isso, necessitamos definir qual o nível de agregação que será aplicado sobre essas partes para contá-las.

A consulta de distribuição de freqüência é uma consulta muito parecida com a contagem comum, mas se na anterior são representadas quantidades, neste tipo mostramos distribuições de freqüência. É interessante notar que essas funcionalidades são muito importantes em análises de dados, especialmente em análises estatísticas (por exemplo, para comparação com uma função de densidade de distribuição probabilística). Apesar disso, os bancos de dados relacionais e suas linguagens de consultas (Silberschatz et al., 1999) não prevêem esse tipo de consulta.

Separamos as consultas em três tipos por não ter certeza se seria possível juntar numa mesma consulta projeções sem agregação, contagens comuns e contagens de freqüência, dadas as especificidades de cada tipo de projeção.

Pretendemos em trabalhos futuros investigar a possibilidade de juntar todos esses tipos de projeção e agregação num único tipo de consulta.

3.1.2.2. Etapas e partes da consulta

O modelo de consulta (figura 4) está dividido em partes comuns a todos os tipos de consulta – que contém a maioria dos elementos do modelo, e algumas partes específicas a certos tipos de consulta. Veremos mais à frente que o particionamento dos diferentes elementos do modelo de consultas influencia fortemente o particionamento das etapas do macro-processo de consulta aos dados de medição.

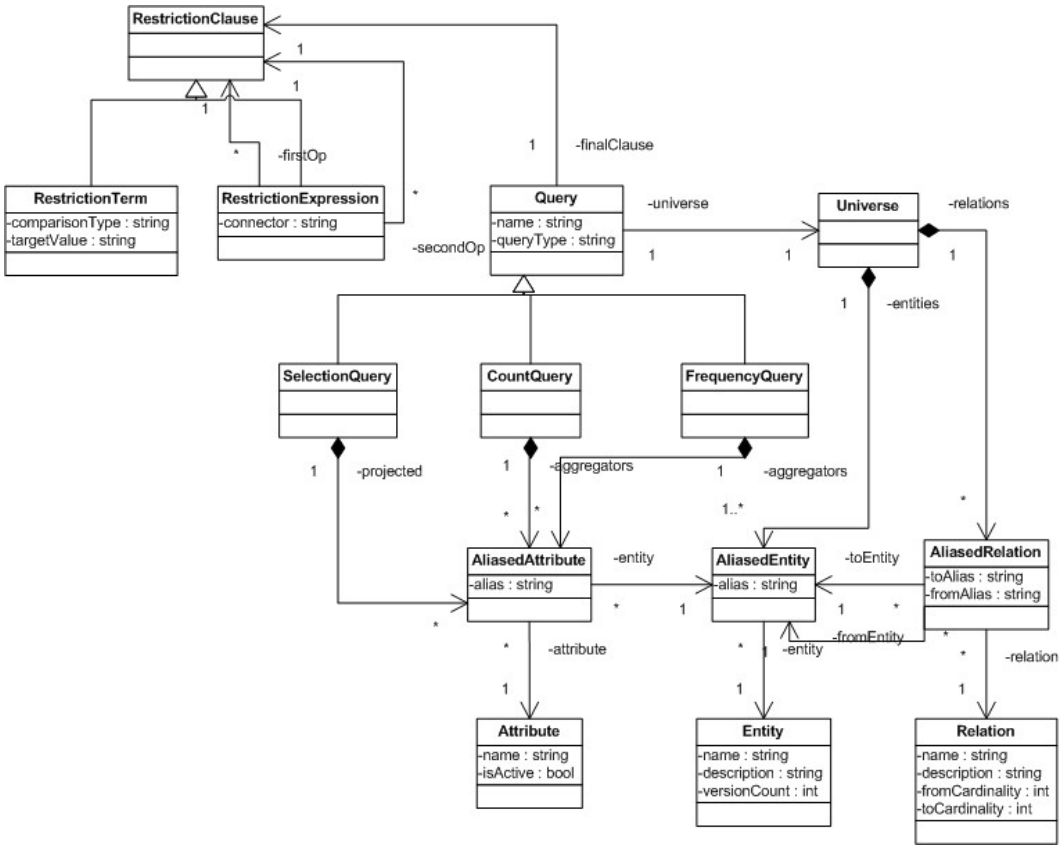

Figura 4 – Modelo de consulta do sistema Clairvoyant

Uma consulta é representada pela classe ***Query***, que é representada por um universo de discurso, um conjunto de restrições e um conjunto de atributos projetados (se for consulta de seleção – ***SelectionQuery***) ou conjunto de atributos

agregadores (se for consulta de contagem – ***CountQuery*** – ou consulta de contagem de freqüência – ***FrequencyQuery***).

Chamamos de *universo de discurso* ou simplesmente *universo* da consulta o conjunto de entidades relacionadas às quais podemos nos referir na consulta. Ele é representado no modelo de consulta pelo campo ***universe*** da consulta, que é um objeto do tipo ***Universe***.

Os membros do universo são um conjunto de entidades que podemos referenciar na consulta e o conjunto de relações que as une. Elas agregam elementos dos respectivos tipos:

- **Entidade contextualizada (*AliasedEntity*):** corresponde a uma entidade do modelo de medição e a um cognome (*alias*) o identificando no contexto da consulta. O cognome se faz necessário para identificar a entidade em cada um de seus possíveis papéis numa consulta, pois é possível que uma entidade assuma mais de um papel numa consulta Como exemplo, podemos imaginar uma consulta onde queiramos mostrar informações sobre um produto e seus subprodutos, tendo a mesma entidade "produto" desempenhando os papéis "produto-pai" e "produto-filho".
- **Relação contextualizada (*AliasedRelation*):** corresponde a uma relação entre duas entidades contextualizados de um universo de consulta.

Chamamos de *conjunto de restrições* de uma consulta o critério que delimita qual parte do universo será analisada na consulta. Ele envolve os conceitos de *cláusula de restrição, termo de restrição* e *expressão de restrição*:

- **Cláusulas de restrição (*RestrictionClause*):** é tudo que pode ser avaliado quanto à veracidade – tanto uma expressão quanto um termo de restrição.
- **Termo de restrição (*RestrictionTerm*):** é uma restrição atômica, composta por um atributo contextualizado que queiramos avaliar, um comparativo (e.g., igual a, diferente, maior) e um valor-alvo de avaliação. Um exemplo de restrição seria "mais de 15.000 linhas de código". Nesse caso, o atributo seria linhas de código, a restrição seria "maior que" e o valor-alvo seria 15.000.

- **Expressão de restrição (*RestrictionExpression*):** é uma conjunção ou disjunção (expressa por conectivos *e* ou *ou*) de dois termos que podem ser termos de restrição ou restrições de expressão.

Por fim, devemos notar que, segundo o modelo de consulta, o critério de restrição da mesma é representado por uma cláusula final de restrição (***finalClause***), que é um objeto do tipo cláusula de restrição (***RestrictionClause***).

No caso de uma consulta de seleção, a *projeção* de uma consulta é o conjunto de atributos contextualizados cujos valores são expostos por uma consulta. Ele é representado no modelo de consulta pelo campo ***projectedAttributes*** da consulta de seleção, que é um conjunto de atributos contextualizados (***AliasedAttribute***).

- **Atributo contextualizado (*AliasedAttribute*):** é um associado a uma entidade contextualizada.

No caso das consultas de agregação (contagem comum ou contagem de freqüência), alguns atributos contextualizados são usados para determinar a granularidade do agrupamento das contagens. Chamamos a esses atributos de *agregadores*, e eles são representados no modelo de consulta pelo campo ***aggregators***, que também são representados por um conjunto de atributos contextualizados.

3.2. Operacionalização dos macro-processos

Ao desempenhar as funcionalidades a ele atribuídas, o repositório de medição tem que fazê-lo no contexto dos modelos de medição e consultas por ele utilizados. Assim sendo, é importante demonstrar como ele manipula as estruturas para desempenhar essas atribuições.

3.2.1. Manutenção do modelo de medição

As funcionalidades agrupadas sob o macro-processo de manutenção do modelo de medição do repositório têm como objetivo a definição e subseqüentes

modificações ao modelo de medição em uso pelo mesmo. As funcionalidades de manutenção do modelo são:

Funcionalidade: Criar nova entidade (***AddEntity***)
Entrada: nome e descrição da entidade
Desdobramento:

1. Cria-se novo registro na tabela ***ENTITY***;
2. Cria-se nova uma nova visão de instâncias dessa entidade com o nome ***INSTANCES_<nome da entidade>***, como projeção da tabela ***ENTITY_INSTANCE*** para essa entidade.

Funcionalidade: Ligar entidades através de relação (***LinkEntities***)
Entrada: nome e descrição da relação, cardinalidade nas extremidades de origem e de destino da relação (*única* ou *múltipla*), nome das entidades nas extremidades de origem e destino da relação.
Desdobramento:

1. Cria-se novo registro correspondente na tabela ***RELATION***;
2. Cria-se nova visão de instâncias dessa relação com o nome ***LINK_<nome_da_relação>***, como projeção da tabela ***RELATION_INSTANCE*** para essa relação.

Funcionalidade: Criar novo tipo de atributo-base (***AddBaseAttributeType***)
Entrada: nome e descrição do atributo, tipo de escala do mesmo.
Desdobramento:

1. Cria-se novo registro correspondente na tabela ***ATTRIBUTE_TYPE***.

Observações: Só é usada para a criação de tipos de atributo, não-categóricos, para os categóricos nominais e ordinais são usadas as primitivas ***AddNominalCategory*** e ***AddOrdinalCategory***, respectivamente.

Funcionalidade: Criar novo tipo de atributo derivado (***AddDerivedAttributeType***)
Entrada: nome e descrição do atributo, nome dos atributos-fonte da derivação (que podem ser atributos-base ou derivados) e a operação de derivação (aplicada sobre os atributos-fonte para gerar o atributo derivado)

Desdobramento:

1. Insere nova entrada na tabela ***ATTRIBUTE_TYPE***;
2. Insere nova entrada na tabela ***DERIVATION_TYPE***.

Observações: Atualmente as operações de derivação se encontram limitadas às operações aritméticas. Pela limitação às operações aritméticas, os atributos derivados só podem se basear em atributos de tipo razão ou intervalo e são eles mesmos do tipo razão ou intervalo.

Funcionalidade: Adicionar atributo-base a entidade (***LinkEntityAttribute***)

Entrada: nome da entidade, nome do tipo do atributo e nome do atributo para esta entidade.

Desdobramento:

1. Cria-se um registro referente ao atributo na tabela ***ATTRIBUTE***, se ele não existir;
2. Declara-se a situação do atributo perante a entidade como *ativa*;
3. Cria-se a visão ***DATA_<nome da entidade>_<nome do atributo>***, que é uma projeção da tabela ***ATTRIBUTE_DATA*** que irá mostrar os dados referentes a este atributo;
4. Aumenta-se a contagem de versões da entidade;
5. Reconstrói-se a visão correspondente à entidade com o novo atributo.

Funcionalidade: Adicionar atributo derivado a entidade (***AddDerivedAttribute***)

Entrada: nome da entidade, nome do tipo do atributo, nome do atributo para esta entidade, nome dos atributos-fonte da derivação.

Desdobramento:

1. Cria-se um registro referente ao atributo na tabela ***ATTRIBUTE***, se ele não existir;
2. Declara-se a situação do atributo perante a entidade como *ativa*;
3. Cria-se um registro referente ao atributo na tabela ***ATTRIBUTE_DERIVATION***, caso não exista;
4. Aumenta-se a contagem de versões da entidade;
5. Reconstrói-se a visão correspondente à entidade com o novo atributo.

Funcionalidade: Criar nova categoria nominal (***AddNominalCategory***)

Entrada: nome e descrição da categoria.

Desdobramento:

1. É adicionado um registro correspondente na tabela ***NOMINAL_CATEGORY***;
2. É adicionado um registro correspondente na tabela ***ATTRIBUTE***, com tipo de escala configurado como escala nominal.

Funcionalidade: Criar novo item de categoria nominal (***AddNominalCategoryItem***)

Entrada: nome da categoria e rótulo do item.

Desdobramento:

1. Nova tupla na tabela ***NOMINAL_ITEM***.

Funcionalidade: Criar nova categoria ordinal (***AddOrdinalCategory***)

Entrada: nome e descrição da categoria.

Desdobramento:

1. É adicionado um registro correspondente na tabela ***ORDINAL_CATEGORY***;
2. É adicionado um registro correspondente na tabela ***ATTRIBUTE***, com tipo de escala configurado como escala ordinal.

Funcionalidade: Criar novo item de categoria ordinal (***AddOrdinalCategoryItem***)

Entrada: nome da categoria, ordem do item na categoria e rótulo do item.

Desdobramento:

1. É adicionado um registro correspondente à tabela ***ORDINAL _ITEM***.

Funcionalidade: Desligar atributo de entidade (***UnlinkEntityAttribute***)

Entrada: nomes de entidade e do atributo na entidade

Desdobramento:

1. Declara a situação do atributo perante a entidade como inativa.
2. Aumenta a contagem de versões da entidade.
3. Reconstrói a visão correspondente à entidade.

3.2.2. Importação de dados de medição

As funcionalidades agrupadas sob o macro-processo de importação de dados de medição no repositório têm como objetivo o registro de dados de medição provenientes do ambiente de engenharia de software que o modelo do repositório representa. As funcionalidades de importação de dados de medição são:

Funcionalidade: Instanciar entidade (***InstantiateEntity***)
Entrada: nome da entidade
Desdobramento:

1. Um registro é inserido na tabela ***ENTITY_INSTANCE***, que, conseqüentemente, aparece como um registro adicional na visão ***INSTANCES_<nome da entidade>***.

Funcionalidade: Ligar instâncias de entidades (***LinkEntityInstances***)
Entrada: nome da relação, instância de entidade de origem, instância de entidade de destino.
Desdobramento:

1. Insere mais uma tupla na tabela ***RELATION_INSTANCE*** que, conseqüentemente, aparece como um registro adicional na visão ***LINK_<nome da relação>***.

Funcionalidade: Registrar valor de atributo para entidade (***SetAttributeData***)
Entrada: nome da entidade, instância de entidade, nome do atributo, valor do atributo.
Desdobramento:

1. Insere tupla na tabela ***ATTRIBUTE_DATA***, consequentemente aparecendo também na visão ***DATA_<nome da entidade>_<nome do atributo>***.

3.2.3. Consulta

O macro-processo de consulta tem como objetivo construir requisições de resgate de dados do repositório para inspeção humana. Devemos notar que, assim como há elementos do modelo de consulta específicos a certos tipos de consulta, há primitivas também específicas a alguns tipos de consulta. A delimitação do universo e sua conseqüente restrição passam-se de maneira absolutamente igual para todos os tipos de consulta. Já a projeção e agregação têm caminhos diferentes nas consultas de seleção, contagem comum ou contagem de freqüência. As funcionalidades do macro-processo de consulta aos dados de medição são:

Funcionalidade: Criar nova consulta para construção (***CreateNewQuery***)
Entrada: nome que se quer atribuir à consulta e seu tipo
Desdobramento:

1. O sistema passa a aceitar que os outros macro-processos de consulta aos dados sejam acionados para essa consulta.

Observações: O nome dado será usado para identificar a consulta nas outras primitivas.

Funcionalidade: Determinar a entidade inicial do universo (***SetFirstElement***)
Entrada: nome da consulta, nome da entidade e cognome que se quer atribuí-la no contexto da consulta.
Desdobramento:

1. Adiciona-se a entidade em questão ao conjunto de entidades do universo da consulta (***entities***), atribuindo-lhe o cognome em questão no contexto da consulta.

Observações: A determinação da entidade inicial do universo da consulta é tratada separadamente porque, por definição, se ainda não houver entidades no universo também não haverá relações associadas ao universo nesse momento.

Funcionalidade: Expandir universo com entidade relacionada (***ExpandUniverse***)
Entrada: nome da consulta, cognome da entidade de origem (já pertencente ao universo), entidade de destino usada para expandir o universo, cognome que se

quer atribuí-la no contexto da consulta e relação utilizada para se alcançar a entidade de destino

Desdobramento:

1. Adiciona-se a entidade de destino em questão ao conjunto de entidades do universo da consulta (***entities***), atribuindo-lhe o cognome em questão no contexto da consulta;
2. Adiciona-se a relação em questão ao conjunto de relações entre atributos cognominados da consulta (***relations***).

Observações: As entidades que podemos escolher para a expansão do universo são as que estão ligadas ao universo atual por via de algum relacionamento.

Funcionalidade: Adicionar termo de restrição ao conjunto de termos de restrição que dará origem à cláusula de restrição da consulta (***AddRestrictionTerm***)

Entrada: nome da consulta, nome e cognome do atributo sobre o qual se quer impor uma restrição, tipo de comparação que ser quer usar para restringir o atributo e valor-alvo da comparação usada para restrição.

Desdobramento:

1. Um termo de restrição (***RestrictionTerm***) com atributo, tipo de comparação e valor alvo fornecidos é adicionado ao conjunto de cláusulas temporárias da consulta;

Observações: A sistemática de geração de uma cláusula de restrição para a consulta passa primeiro pelo estabelecimento das cláusulas atômicas de restrição (as *expressões de restrição*) por meio desta primitiva, para posteriormente agrupá-las de maneira incremental (duas a duas) por meio de uma primitiva de agrupamento de cláusulas, até que tenhamos uma única cláusula final de restrição para a consulta.

Funcionalidade: Agrupar cláusulas de restrição da consulta (***GroupClauses***)

Entrada: nome da consulta, referências às duas cláusulas que se quer agrupar e conectivo que se quer utilizar para agrupá-las (*e* ou *ou*)

Desdobramento:

1. Adiciona-se a nova cláusula agrupada ao conjunto de cláusulas temporárias da consulta;

2. Removem-se as cláusulas individuais usadas para gerar a cláusula agrupada do conjunto de cláusulas temporárias da consulta;
3. Se o conjunto de cláusulas resultante tiver só um elemento, esse se torna a cláusula final de restrição da consulta (***finalClause***).

Observações: O agrupamento de cláusulas de restrição é feito de duas em duas cláusulas. As cláusulas que agrupamos vêm de um conjunto de cláusulas intermediárias relacionadas ao estado momentâneo do processo de construção da consulta. Ao se escolherem as cláusulas intermediárias e o conectivo, uma nova cláusula intermediária é composta por esses elementos, e as cláusulas intermediárias que deram origem à nova cláusula intermediária são removidas do conjunto de cláusulas intermediárias em questão.

Funcionalidade: Adicionar atributo ao conjunto de agregadores (***AddAggregator***)
Entrada: nome da consulta, nome e cognome do atributo que se quer adicionar ao conjunto de agregadores da consulta.
Desdobramento:

1. Adiciona-se o atributo (***AliasedAttribute***) em questão ao conjunto de atributos utilizados para discriminar a contagem (comum ou de freqüência) da consulta (***aggregators***).

Observações: Esta primitiva só pode ser acionada para consultas de contagem de freqüência ou contagem comum.

Funcionalidade: Agrupar atributo ao conjunto de atributos projetados (***SetProjection***)
Entrada: nome da consulta, nome e cognome do atributo que se quer adicionar ao conjunto de atributos projetados da consulta.
Desdobramento:

1. Adiciona-se o atributo (***AliasedAttribute***) em questão ao conjunto de atributos projetados pela consulta (***ProjectedAttriutes***).

Observações: Esta primitiva só pode ser acionada para consultas de seleção.

Funcionalidade: Persistir consulta construída (***PersistQuery***)
Entrada: nome da consulta que se quer persistir

Desdobramento:

1. É criada uma visão associada ao nome da consulta que é definida pelo conteúdo especificado da consulta.

Observações: Após ser persistida, a consulta poderá ser referenciada no macro-processo de exportação.

3.2.4. Exportação

As funcionalidades agrupadas sob o macro-processo de exportação de dados de medição do repositório têm como objetivo resgatar dados do repositório de medição de forma a torná-los disponíveis para uso em outros sistemas. A primitiva de exportação do sistema é:

Funcionalidade: Exportar dados de medição resultantes de uma consulta (***ExportMeasurementData***)

Entrada: nome da consulta utilizada cujo resultado queremos exportar, tipo formato para dados de exportação (e.g., CSV, XML)

Desdobramento:

1. Executa-se a consulta identificada e o seu conteúdo é resgatado;
2. O sistema executa as transformações especificadas nos resultados da consulta;
3. O seu conteúdo é retornado no formato escolhido.

3.3. Arquitetura do sistema

3.3.1. Sistemas constituintes

O sistema Clairvoyant tem atualmente quatro módulos associados. Um deles desempenha as funções do macro-processos de manutenção do modelo de medição, efetuando as operações pertinentes por meio da leitura de arquivos de texto com comandos de manutenção. Existe também um módulo que desempenha as funções do macro-processo de importação de dados, que funciona analogamente ao módulo responsável pela manutenção (leitura de arquivos). No

mais, existe um módulo responsável pelo macro-processo de consulta implementado como um aplicativo web. Por fim, existe um módulo simples de exportação que é basicamente uma interface de linha de comando para a primitiva de exportação previamente descrita.

O módulo de consulta do sistema Clairvoyant tem como finalidade construir consultas por meio de uma interface gráfica provida por um sistema Web, para inspeção dos dados por usuários humanos ou para usar consultas construídas como fontes de exportação de dados para outros sistemas. Nele, especificam-se consultas, passando pelas etapas descritas nas primitivas do macro-processo de consulta, até que o resultado da consulta é mostrado na interface do sistema ou a consulta é armazenada. Devemos notar que há uma correspondência quase biunívoca entre as telas do sistema e as etapas do macro-processo de consulta.

3.3.2. Componentes comuns dos sistemas

Existe um núcleo básico do sistema Clairvoyant, formado pelos componentes comuns aos sistemas previamente mencionados. Nesse núcleo, detectamos a separação dos componentes em três grandes grupos: dois associados ao arcabouço teórico do repositório, e outro associado à interface do repositório com o acesso aos dados persistentes.

O primeiro grupo de componentes associado ao arcabouço teórico do repositório é o dos modelos de medição e consulta. Ele é representado pelos seguintes pacotes Java:

- **model.measurement:** implementa o meta-modelo de medição conforme especificado pela representação conceitual contida neste trabalho;
- **model.measurement.scale:** implementa a hierarquia de tipos de escalas apresentada neste trabalho;
- **model.query**. implementa a hierarquia de tipos de escala estendida apresentada neste trabalho.

O segundo grupo de componentes associado ao arcabouço teórico do repositório é o das primitivas de comando do repositório. A idéia por trás do controle do sistema é fazer cada primitiva dos macro-processos de medição

corresponder a uma classe encapsulando um comando equivalente. Ela foi fortemente inspirada nos padrões de projeto *Command* (Gamma et al., 1995) e *Front Controller* (Alur et al., 2003). Esses comandos são representados pelos seguintes pacotes Java:

- **command:** implementa classes abstratas ***AbstractCommander*** (um acionador de comandos da qual cada um dos outros pacotes tem uma implementação concreta) e ***AbstractCommand***, de qual cada comando presente em cada pacote deve herdar;
- c**ommand.maintenance:** contém os comandos associados às primitivas de manutenção do modelo de medição do repositório;
- **command.import:** contém os comandos associados às primitivas de importação de dados de medição no repositório;
- **command.query:** contém os comandos associados às primitivas de consulta aos dados de medição do repositório;
- **command.export:** contém os comandos associados às primitivas de exportação de dados de medição do repositório.

O grupo de componentes referente à interface com o mecanismo de persistência utilizado pelo repositório opera com uma *fachada de acesso a dados* (baseada no padrão de projeto *Façade*) (Gamma et al., 1995). Ela é o ponto único de acesso ao mecanismo de persistência do sistema, ou seja, todas as operações efetuadas pelo sistema que devam interagir com seu mecanismo de persistência devem fazê-los através de chamadas a métodos dessa fachada. Os métodos dessa fachada, por sua vez, acionam métodos de interfaces dependentes do tipo de objeto do meta-modelo de medição: ***AttributeDataAccess***, ***EntityDataAccess***, ***NominalCategoryDataAccess***, ***OrdinalCategoryDataAccess*** e ***QueryDataAccess***. Essas interfaces, por sua vez, deverão ter implementações concretas associadas a um mecanismo de persistência de dados (atualmente, um banco de dados relacional). Os pacotes referentes à persistência são:

- **dataaccess:** implementa a fachada de acesso a dados e define as interfaces de acesso a dados;
- **dataacces.sql:** fornece implementações das interfaces de acesso a dados valendo-se de funcionalidades típicas de bancos de dados relacionais acionadas pela API JDBC.

3.4. Aspectos relevantes da implementação do sistema

O sistema Clairvoyant foi implementado na linguagem Java (versão 5), fazendo uso do banco de dados relacional MySQL para a persistência dos dados. Os acessos ao banco de dados são feitos através da API JDBC. No caso específico do módulo web que implementa a construção de consultas, a plataforma de execução se apóia na tecnologia *Java Server Pages* e no uso de *Servlets*. O sistema foi testado usando o servidor de aplicativos Apache Tomcat.

A principal dificuldade da implementação do sistema foi o mapeamento entre o meta-modelo de medição em Java e sua implementação relacional, que foi desenvolvido manualmente. Não foram usadas soluções de mapeamento objeto / relacional previamente existente (e.g., Hibernate) (JBoss, 2006) por considerarmos que elas não atendiam às particularidades de um meta-modelo de medição, notadamente a criação dinâmica de visões em tempo de execução.

3.5. Conclusão

Ao analisarmos os macro-processos do sistema, podemos dizer que eles permitem a evolução no modelo de medição seguindo os preceitos de usabilidade demandados no capítulo anterior, já que a execução desses macro-processos não demanda ao usuário o conhecimento das junções, indireções e do versionamento sob o qual o sistema se baseia. O meta-modelo de medição nos fornece uma funcionalidade efetiva de medição evolutiva, enquanto o modelo de consulta e as primitivas do repositório ocultam do usuário do repositório a complexidade subjacente envolvida nessa evolução, em especial, o versionamento das visões das entidades.

Para fundamentar a afirmação acima, basta notarmos que na interação do usuário com o sistema, só são referenciados como parâmetros os elementos básicos do meta-modelo de medição: entidade, atributo e relação. Desta maneira, ele encapsula a maior parte dos elementos do meta-modelo de medição. Assim sendo, podemos recomendar o uso conjunto de modelo de medição baseado em meta-dados, indireção e versionamento de esquema como uma alternativa viável

para implementar a evolução transparente de modelos de medição em repositórios de medição.

Entretanto, esse mesmo modelo de medição baseado em meta-dados e indireção é responsável por um dos pontos fracos do sistema: a quantidade significativa de operações de junção que devem ser feitas para se processar uma consulta. Isso pode acarretar problemas de desempenho ao trabalharmos com uma massa de dados de medição grande. Além disso, essa indireção gera um número enorme de visões no banco de dados, elas são utilizadas como instrumento para mapear as possibilidades de indireção no modelo do banco de dados. No mais, convém ressaltar que a estrutura do sistema, com as primitivas dos macro-processos do repositório mapeadas em comandos, pode tornar mais fácil a adaptação do sistema Clairvoyant a outras plataformas.