2
Conceitos Básicos e Trabalhos Relacionados

Com o aumento da utilização de sistemas com bases de dados cada vez maiores e com a alta complexidade das aplicações atuais, tornou-se necessário para os Sistemas Gerenciadores de Bancos de Dados (SGBDs) incorporar a funcionalidade de auto-sintonia (*self-tuning*) [Horn01]. No manifesto de [Horn01] é feito um chamado à construção de sistemas integrados, autônomos, capazes de se adaptar automaticamente às variações do ambiente. Dessa forma, diversos trabalhos buscaram encontrar soluções automáticas em diferentes aspectos do SGBD com o intuito de melhorar o desempenho.

Este capítulo tem por objetivo revisar trabalhos da área de sintonia automática do projeto físico do banco de dados focados na sintonia de visões materializadas e expor conceitos relacionados com a gerência de visões materializadas. Os trabalhos, aqui apresentados, procuram reduzir ou eliminar a intervenção humana na hora de decidir quais visões materializadas devem ser criadas para diminuir o tempo de resposta. Nosso objetivo é contextualizar nosso trabalho de automatizar a escolha de visões materializadas em sistemas de banco de dados relacionais.

2.1. Índices e Visões Materializadas

Uma atividade comumente realizada por administradores de sistemas de bancos de dados para acelerar o desempenho de consultas submetidas a um SGBD relacional é a seleção de índices sobre as tabelas presentes na base de dados [Salles04]. Índices são estruturas de dados que organizam registros de dados em disco para otimizar operações de recuperação [Ramakrishnan08], desta forma índices melhoram o tempo de resposta a consultas no banco de dados. Visões materializadas também podem ser usadas para melhorar o tempo de resposta a consultas.

Uma visão é definida por uma consulta cuja tabela resultado não é armazenada explicitamente no banco de dados. As tuplas são calculadas conforme requisitado pelo sistema [Ramakrishnan08]. Portanto, uma visão é uma definição

de consulta sobre um conjunto de tabelas do banco de dados que é recalculada cada vez que a visão é referenciada.

A visão materializada é, como seu nome indica, uma visão cujo resultado da consulta é fisicamente armazenado na base de dados [Silberschatz06]. São objetos persistentes como as tabelas, porém não fazem parte do conjunto original de dados que define o banco utilizado e têm existência dependente das tabelas sobre as quais são definidas.

O uso de visões materializadas pode ser mais vantajoso do que índices em consultas com agregações e agrupamentos, pois elas já armazenam o resultado das agregações não precisando ser calculado quando for referenciada. Para compreendermos melhor a diferença do uso de visões materializadas e índices analisaremos um exemplo. Tomemos a consulta A e a visão V sobre a tabela Venda conforme figura 1:

Consulta A	Visão V
select idPedido from Venda group by idPedido having sum(qtdItem) > 3	select idPedido, sum(qtdItem) from Venda group by idPedido

Figura 1 - Consulta A e Visão V para a consulta A

Considerando a existência de um índice sobre o campo *idPedido* da tabela Venda, ele seria usado para agrupar a consulta A, porém ainda precisaria ser feito a agregação "*sum*" do *having*. Considerando a existência da visão V, seria necessário apenas percorrer a visão materializada V para obter o mesmo resultado da consulta A, sem ser necessário fazer agrupamento e agregação.

2.2.Auto-Sintonia de Banco de Dados

Considerando um sistema de banco de dados em operação, seu projeto físico pode ser reavaliado por meio da coleta de estatísticas reais sobre os padrões de uso. O monitoramento dos recursos disponíveis pode revelar gargalos de desempenho [Monteiro08]. Portanto, é necessário monitorar e revisar o projeto físico do banco de dados constantemente, atividade conhecida como sintonia do projeto físico do banco de dados [Navathe05].

Auto-Sintonia é o processo de buscar uma configuração ótima através da automatização das operações do SGBD visando o aumento de desempenho das

aplicações e a redução da interação do DBA (*database administrator*) com o sistema.

Existem diversos trabalhos que procuram encontrar soluções automáticas para sintonizar SGBDs em algum aspecto. Em [Milanés et al. 04] é feito um estudo sobre trabalhos de auto-sintonia de banco de dados presentes na literatura e em sistemas comerciais. Foi possível classificar os trabalhos anteriores de pesquisa de auto-sintonia em auto-sintonia local e auto-sintonia global. Outros trabalhos de auto-sintonia posteriores se enquadram no grupo de auto-sintonia local ou global como pode ser visto em [Karde10, Ashedevi09, Baril03 e Agrawal et al. 00].

A auto-sintonia local procura estudar problemas específicos de sintonia fina. Entre os problemas específicos podemos citar o problema de automatização do projeto físico de banco de dados (como índices e visões materializadas), da alocação de dados, que consiste em como alocar e realocar fragmentos de arquivo, controle de carga, que consiste em controlar o processo de múltiplas transações, entre outros.

A auto-sintonia local é o esforço na automatização dos ajustes do sistema para resolver um entre os problemas seguintes [Milanés et al. 04]:

• Projeto físico: consiste em escolher a melhor organização física para um dado esquema e uma carga de trabalho específica para que a busca pelos dados seja feita mais rápida.

• Alocação de dados: consiste em determinar como alocar e realocar dinamicamente fragmentos de arquivos ou relações nestes elementos de tal forma que o equilíbrio seja o melhor possível.

• Controle de carga: consiste em controlar o processo de múltiplas transações de forma concorrente e que se utiliza de bloqueios para garantir as propriedades ACID.

• Substituição de páginas: consiste em manter em memória o conjunto de páginas mais populares do banco de dados.

• Ajuste de buffers: consiste em estudar quantos buffers distintos devem ser configurados no banco de dados e de que forma devem ser distribuídos os objetos para que o desempenho aumente.

• Refino de estatísticas: consistem em escolher quais estatísticas devem ser criadas no banco de dados e refiná-las de modo de não haver necessidade de executar comandos específicos para coleta no sistema.

Já a auto-sintonia global procura estudar como é possível ajustar o SGBD de forma que traga benefícios de desempenho para o sistema como um todo. Nesta abordagem procura-se encontrar um equilíbrio entre os diversos aspectos de sintonia local de forma a alcançar um desempenho global melhor do que seria alcançado por sintonizar cada aspecto isoladamente. Porém as inter-relações entre os diversos componentes de um SGBD não são bem conhecidas [Weikum et al 02].

2.3.Gerência de Visões Materializadas

Uma visão materializada pode ser vista como uma cópia dos dados necessários para responder uma consulta e que pode ser acessada rapidamente [Monteiro08]. Assim, trata-se de uma estrutura que oferece acesso rápido aos dados.

A presença de certas visões materializadas pode melhorar significativamente o desempenho, particularmente para aplicações de suporte à decisão [Agrawal et al. 00]. E são úteis quando uma consulta pode buscar dados nelas ao invés de ter que calcular o resultado da consulta no momento da execução.

Para uma melhor compreensão desse processo, consideremos as tabelas *r, s* e *t*, e as seguintes definições:

- Uma visão ***v*** = r ⋈ s
- Uma consulta ***a*** = r ⋈ s ⋈ t
- Uma consulta ***b*** = v ⋈ t

Suponha que a visão ***v*** esteja disponível e materializada e um usuário submeta uma consulta ***a***. A reescrita da consulta ***a*** utilizando a visão, resultando na consulta ***b***, pode oferecer um plano de consulta mais eficiente do que a otimização da consulta ***a*** [Silberschatz06].

A desvantagem das visões materializadas é que elas precisam ser mantidas atualizadas quando os dados, resultado da execução da definição da visão, mudarem [Silberschatz06]. Ou seja, há um custo associado à atualização da visão materializada toda vez que alterações forem realizadas nos dados das tabelas base. Desta forma, deve ser analisado se a visão materializada está trazendo mais

benefícios, sendo bastante usada para auxiliar na busca de dados de consultas, ou se está causando mais malefício por conta das frequentes atualizações nos dados referenciados na definição da visão, causando um custo adicional ao ter que atualizar também a visão materializada.

Neste trabalho, delimitamos nosso escopo para carga de trabalho com característica OLAP (Online analytical processing), apenas com consultas, sem atualizações, por ser uma carga compatível com utilização de visões materializadas por evitar a atualização delas.

Para que a visão materializada possa ser usada, ela tem que se manter atualizada com os dados da tabela base. Para tanto é necessário recriar a visão a partir dos dados da origem ou atualizá-la de forma incremental. Este é um problema de pesquisa antigo e vários trabalhos descrevem algoritmos que permitem a manutenção incremental das visões materializadas com diferentes domínios de aplicação [Ceri91, Harrison92, Gupta93]. Outro aspecto importante diz respeito ao momento em que as atualizações são efetuadas uma vez que as tabelas base são alteradas, ora imediatamente, ora postergadas, de tal forma que todas as atualizações ocorridas sejam reunidas a fim de serem aplicadas juntas em processos *batch*.

Para uma consulta da carga de trabalho, podem-se ser enumeradas várias visões partindo-se do princípio que uma visão materializada pode ser proposta para qualquer subconjunto de tabelas em uma consulta. Desta forma, selecionar todas as visões materializadas sintaticamente relevantes não é escalável [Agrawal et al. 00]. Existem diversos trabalhos que tratam do tema da seleção de visões materializadas relevantes para consultas [Karde10, Ashedevi09, Monteiro06, Baril03].

Além do custo de manutenção de visões materializadas, deve-se analisar também o momento em que a visão materializada deve ser criada para não prejudicar o desempenho do sistema. Se o servidor de banco de dados está sendo muito requisitado para atualizações, pode não ser indicado materializar visões. Porém, se o servidor está sendo muito requisitado para consultas dispendiosas e a visão a ser materializada pode melhorar o custo dessas consultas, cabe considerar a materialização da visão de imediato de forma a não prejudicar ainda mais o desempenho do sistema como um todo.

Definir o melhor conjunto de visões a serem materializadas é um problema não trivial. A carga de trabalho pode mudar com o tempo e uma visão materializada pode não mais ser útil ou até mesmo estar prejudicando o processo por exigir muita atualização e não mais ser usada em consultas [Silberschatz06].

2.4.Seleção de Visão Materializada

Vários trabalhos discutem como selecionar visões materializadas [Karde10, Ashedevi09, Baril03 e Agrawal et al. 00]. Em [karde10] são selecionadas visões a serem materializadas voltadas para SGBDs distribuídos. Em [Ashedevi09 e Baril03] estuda-se o problema de seleção de visões que devem ser materializadas em relação à freqüência com que as consultas são executadas no banco de dados.

Em [Agrawal et al. 00] são descritos algoritmos para seleção de visões baseados na frequência das consultas executadas e no tamanho das tabelas referenciadas na consulta. Isso é importante, pois criar uma visão materializada sobre um subconjunto de tabelas relativamente pequenas pode não trazer um benefício satisfatório que compense os custos de criação e manutenção da visão materializada.

[Agrawal et al. 00] apresenta uma arquitetura de sistema e um conjunto de algoritmos para resolver o problema de seleção de visões materializadas de acordo com uma carga de trabalho. O primeiro passo deste trabalho é identificar subconjunto de tabelas (TS), que correspondem a qualquer subconjunto de tabelas formado por tabelas referenciadas em uma consulta que podem ser interessantes dentre todos os possíveis subconjuntos de tabelas para uma determinada carga de trabalho.

Nesta dissertação consideramos a frequência com que as consultas são executadas e o tamanho das tabelas referenciadas na consulta. A arquitetura utilizada nesta dissertação difere da arquitetura proposta por [Agrawal et al. 00] pois é on-the-fly. Ou seja, a carga de trabalho é capturada à medida que ela é submetida. Para a arquitetura de [Agrawal et al. 00] é necessário um script com a carga de trabalho toda que será analisada, ou seja, não considera capturar a carga de trabalho no momento da execução do sistema.

Podemos considerar um TS interessante se este tiver o potencial de reduzir significativamente o custo de execução de consultas sobre determinada carga de trabalho. Considere que as tabelas pertencentes a um TS são muito volumosas e

que uma grande quantidade de consultas submetidas ao SGBD utiliza TS. Logo, criar uma visão materializada sobre este TS pode trazer mais benefícios no tempo de resposta do que sobre um TS que não é muito utilizado e que envolva tabelas pequenas.

As TS são consideradas interessantes a partir de métricas que determinam o custo de todas as consultas onde uma TS ocorre e o tamanho das tabelas. A lista de TS interessantes indica quais as visões candidatas para materialização.

2.5. Arquitetura não-intrusiva para manutenção do projeto físico do banco de dados

[Monteiro08] apresenta uma abordagem não intrusiva para a manutenção automática e *on-the-fly* do projeto físico de banco de dados. Uma vantagem de não ser intrusiva, é ser totalmente desacoplada do código do SGBD, podendo assim ser utilizada com diferentes SGBDs. Outra vantagem é que como não altera o código do SGBD, não altera o seu desempenho, sendo necessário realizar apenas consultas rápidas a metabase para obter informações necessárias para arquitetura.

A estratégia é fazer consultas periódicas à metabase do SGBD utilizado, capturando a carga de trabalho para obter as expressões SQL que foram submetidas, juntamente com seus respectivos planos de execução e estimativas de custo. Essas informações são armazenadas em uma metabase local. Cada expressão capturada é analisada em busca de identificar estruturas de acesso mais adequadas e seus respectivos benefícios.

Para acompanhamento dessas estruturas de acesso que compõem o projeto físico de um banco de dados, a estratégia usada baseia-se em heurísticas que executam continuamente e que atribuem benefícios às estruturas de acesso que contribuam (estruturas reais) ou possam contribuir (estruturas hipotéticas) de forma positiva, diminuindo o custo de execução nos comandos submetidos ao SGBD.

Entretanto, é notável que operações de atualização sobre uma determinada tabela t também necessitam ser analisadas, pois podem implicar em atualização das estruturas de acesso definidas sobre t (como no caso de índices). Desta forma o custo de um comando de atualização pode ser considerado um malefício. Nesta

dissertação consideramos apenas consultas, não avaliando o malefício de uma atualização.

A arquitetura proposta neste trabalho é baseada em componentes onde cada componente executa uma tarefa específica. Os componentes apresentados nessa arquitetura são, basicamente, aqueles com as seguintes funções na arquitetura: (i) obtenção da carga de trabalho; (ii) obtenção de estatísticas do banco alvo; (iii) manutenção do projeto físico (índices, visões materializadas, clusterização, partição de tabela) e (iv) executor.

A tarefa do componente de obtenção da carga de trabalho é capturar periodicamente a carga de trabalho submetida ao SGBD alvo. Para tanto o componente consulta a metabase do SGBD e captura as cláusulas SQL que foram executadas, juntamente com seu plano de execução e estimativas de custo.

O componente para obtenção de estatísticas acessa o SGBD alvo e recupera informações tais como o número de páginas ocupadas por uma tabela, a quantidade de tuplas de uma tabela, a altura de um índice, dentre outros.

O componente para manutenção do projeto físico consulta periodicamente a metabase onde são armazenadas as consultas capturadas para verificar se novas consultas foram adicionadas (a frequência que é capturada as consultas é configurável). Em caso afirmativo, para cada nova consulta capturada, o agente aplica heurísticas para seleção e acompanhamento de determinada estrutura física. Esta heurística determinará os melhores conjuntos de estruturas físicas (índices, visões materializadas, clusters) para cada consulta e seus respectivos benefícios.

Já o componente executor é responsável pelas atualizações no esquema do SGBD alvo, isto é, pela criação, remoção ou reorganização das estruturas que compõe o projeto físico do banco de dados..

2.6. Seleção automática de Visão Materializada

Como mostrado na seção 2.4 existem trabalhos relacionados à seleção de visões materializadas, porém poucos relacionados à seleção de forma automática como [Monteiro06]. Pontos importantes devem ser considerados para seleção automática de visões materializadas como:

1. Definir quais consultas serão utilizadas para definir a visão. Precisamos definir visões sobre as consultas executadas no banco de dados. Os números de visões possíveis para cada consulta tende a crescer à medida que a consulta abrange mais campos projetados e

tabelas. Conforme visto na seção 2.4, este é um dos aspectos mais estudados da literatura.

2. Frequência dos comandos executados. Para melhor aproveitamento da visão materializada, devemos considerar quantas vezes ela vai ser usada para saber se sua criação será realmente compensada.
3. Variações das cargas de trabalho. Algumas cargas de trabalho tendem a possuir variações, desta forma, pode ser necessário que novas visões atendam a nova demanda.
4. Obtenção da carga. A carga de trabalho é um insumo básico para determinar se uma visão materializada será usada, porém muitos trabalhos deixam esse trabalho na mão dos administradores de banco de dados (DBA).
5. Determinar o momento de materialização da visão. A materialização da visão implica em transações que podem ser dispendiosas podendo atrasar respostas de comandos da aplicação. Para tanto, é necessário materializar a visão apenas com a garantia que ela será útil.

A maioria dos trabalhos presentes na literatura, como [Karde10, Ashedevi09, Baril03 e Agrawal et al. 00], abordam os primeiros dois pontos acima. Eles apoiam o DBA na tarefa de seleção de visões materializadas, mas os trabalhos não eliminam completamente a intervenção humana da tarefa de sintonia de visões materializadas.

Em [Monteiro06] é tratada uma abordagem de seleção automática de visões materializadas, sem a necessidade da interferência humana, baseada no trabalho de [Agrawal et al. 00]. É considerado que uma vez tendo escolhido as visões materializadas candidatas, precisa-se pesquisar dentre estas estruturas quais podem ser efetivamente usadas pelo otimizador para melhorar o desempenho ao processar a consulta. A idéia é que um módulo de seleção final de visões materializadas seja usado para este trabalho. O algoritmo usado nesse módulo é baseado na idéia da heurística de benefícios utilizado em [Salles04, Morelli06a].

A idéia básica da heurística de benefícios, com foco em estruturas de índices, abordada em [Salles04, Morelli06a] atribui um determinado valor a um índice, seja ele hipotético ou real. Índice hipotético é um índice que não existe fisicamente no banco de dados. Esse valor é a diferença entre o custo obtido através do plano de execução de uma determinada consulta da carga de trabalho com ou sem a presença do índice em consideração. Esse custo pode ser positivo, caso o uso do índice traga benefício na execução de uma consulta, ou negativo, em caso de malefício.

O índice é criado caso seu benefício acumulado atinja um determinado valor. Esse valor foi especificado como o custo de criação do índice. Caso o benefício acumulado seja negativo e o módulo do benefício acumulado seja igual ao custo de criação, o índice deve ser removido ou reindexado.

O trabalho de [Monteiro06] sobre seleção final de visões materializadas define alguns fatores para serem utilizados pelas heurísticas de avaliação de consultas e atualização que são apresentados a seguir:

(1) CR: o melhor plano de custo (gerado pelo otimizador) diante de configurações de visões materializadas reais.
(2) CH: o melhor plano de custo (gerado pelo otimizador) diante de configurações de visões materializadas reais e também hipotéticas.
(3) CN: o melhor plano de custo (gerado pelo otimizador) na ausência de visões materializadas hipotéticas ou reais.
(4) CA: o custo estimado para atualizar uma visão materializada durante o processamento de uma cláusula de atualização (insert/update/delete) em uma relação base.
(5) B_K: Benefício de uma visão materializada k para uma cláusula SQL corrente.
(6) BAC_K: O benefício acumulado da visão materializada k para o conjunto de todas as cláusulas SQL executadas.
(7) CC_K: O custo de estimado de criação de uma visão materializada não calculado pelo SGBD.

Baseado nas heurísticas de benefícios sobre índices de [Salles04, Morelli06a] e da seleção de visão materializada de [Agrawal et al. 00], em [Monteiro06] são utilizados esses fatores apresentados acima para definir a heurística de avaliação de consultas (ver figura 2).

```
1.  Para cada VM candidata CVM (usada no melhor plano de acesso [com VM candidata e real])
2.      B_CVM = CR -CH
3.      BAC_CVM = BAC_CVM + B_CVM
4.      Se BAC_CVM > CC_CVM Então
5.          Cria visão materializada CVM
6.          BAC_CVM = 0
7.      Fim se
8.  Fim para
```

Figura 2 - Heurística para avaliação de consultas

O algoritmo apresentado na figura 2 atribui benefício a uma visão candidata através da diferença entre o melhor plano de custo real (CR) e o melhor plano de custo diante da visão hipotética (CH) como pode ser visto no passo 2. Esse

benefício é adicionado ao benefício acumulado da visão candidata no passo 3. É verificado se o benefício acumulado da visão é maior do que o custo de criação como pode ser visto no passo 4. Se for maior, então a visão candidata é materializada no passo 5.

2.7. Consultas e Visões Materializadas

Nesta dissertação analisamos quando uma consulta utiliza uma visão materializada. Para tanto se realizou testes com algumas visões e consultas que serão mostradas nesta seção.

Para entendermos os testes vamos definir alguns conceitos. Considere que uma consulta básica é uma consulta que não possui uma consulta interna, ou seja, é uma consulta contendo um *select* e um *from*, podendo conter *where*, *group by*, *having* e *order by*. E considere que uma consulta aninhada é aquela que tem outra consulta embutida dentro dela; a consulta embutida é chamada de subconsulta [Ramakrishnan08].

Considere uma consulta básica A, obtida de uma subconsulta do Benchmark TPC-H (descrito melhor no apêndice B), e uma visão materializada VM_1 conforme figura 3:

Consulta A	Visão VM_1
select l_orderkey from lineitem group by l_orderkey having sum(l_quantity) > 3	select l_orderkey, sum(l_ quantity) from lineitem group by l_orderkey

Figura 3 - Consulta A do Benchmark TPC-H e Visão VM_1

A visão VM_1 possui todos os campos necessários para a consulta A e, ao realizar o primeiro teste no mesmo ambiente apresentado no capítulo 4, executando a consulta A na presença da visão materializada VM_1, o otimizador do banco utilizou a visão VM_1 para retornar o resultado da consulta A.

Em um segundo teste utilizou outra consulta para avaliar se a visão VM_1 seria utilizada. A consulta utilizada foi a consulta B, obtida a partir de uma consulta do Benchmark TPC-H, exibida na figura 4:

```
    Consulta B
select  o_orderkey,  sum(l_quantity)
from orders, lineitem
where o_orderkey = l_orderkey
group by o_orderkey
```

Figura 4 - Consulta B obtida do Benchmark TPC-H

A visão materializada VM_1 possui todos os campos a serem retornados pela consulta B e todos os campos de condição da consulta B que são utilizados para junção com a tabela orders, porém a visão não possui a junção com a tabela orders. Ao se executar a consulta B na presença da visão materializada VM_1, o otimizador não utilizou a visão VM_1 para retornar o resultado da consulta B. Através desse teste, notou-se que não é garantido que o otimizador utilizará a visão materializada se ela não possuir todos os campos e tabelas da consulta.

Para o terceiro teste utilizamos a consulta C, obtida também do Benchmark TPC-H, e a visão VM_2 exibidas nas figuras a seguir:

```
          Consulta C
select  l_returnflag, l_linestatus,
        sum(l_quantity) , sum(l_extendedprice) ,
        sum(l_extendedprice * (1 - l_discount)) ,
        sum(l_extendedprice * (1 - l_discount) * (1 + l_tax)) ,
        avg(l_quantity), avg(l_extendedprice) ,
        avg(l_discount) , count(*)
from lineitem
where l_shipdate <= DATEADD(DAY, -10, '1998/12/01')
group by l_returnflag, l_linestatus
```

Figura 5 - Consulta C obtida no Benchmark TPC-H

```
     Visão VM_2

select  l_returnflag, l_linestatus, l_shipdate,
        sum(l_quantity) , sum(l_extendedprice) ,
        sum(l_extendedprice * (1 - l_discount)) ,
        sum(l_extendedprice * (1 - l_discount) * (1 + l_tax)) ,
        sum(l_discount) , count(*) as cont
from lineitem
where l_shipdate <= DATEADD(DAY, -10, '1998/12/01')
group by l_returnflag, l_linestatus, l_shipdate
```

Figura 6 - Visão VM_2

A visão materializada VM_2 possui todos os campos a serem retornados pela consulta C, considerando que os campos de média podem ser obtidos através

dos campos de soma correspondentes divido pelo count. Ao se executar a consulta C na presença da visão materializada VM_2, o otimizador utilizou a visão VM_2 para retornar o resultado da consulta C. Através desse teste, notou-se que o otimizador utilizou a visão materializada pois ela possuía todos os campos da consulta.

Outras consultas e visões que foram usadas para avaliar se o otimizador utilizará a visão materializada (se a visão cobrir apenas parte da consulta) estão apresentados no apêndice A.

2.8. Resumo do capítulo

Este capítulo tratou dos conceitos básicos e relacionou alguns trabalhos na área de sintonia fina da estruturas física de banco de dados para contextualizar o objetivo desta dissertação. Esta dissertação envolve pesquisa no contexto da auto-sintonia local do projeto físico de banco de dados, onde seleciona automaticamente visões materializadas usando componentes de forma não intrusiva ao código do SGBD.