2
Auto-sintonia de Bancos de Dados e Agentes de Software

A uso da abordagem de agentes de *software*[1] pode trazer benefícios a áreas de aplicação em que é necessário construir sistemas autônomos, ou seja, capazes de agir sem a intervenção de humanos ou de outros sistemas [66]. A propriedade de autonomia é o ponto-chave para a construção de sistemas de bancos de dados capazes de auto-sintonia e auto-administração. Investigamos, em nosso grupo de pesquisa, como agentes podem ser utilizados no contexto de SGBDs para tornar os sistemas menos necessitados da intervenção de profissionais especializados.

Neste capítulo, iremos revisar alguns trabalhos existentes sobre auto-sintonia de bancos de dados na seção 2.1 e, em seguida, sobre agentes na seção 2.2. Apresentamos alguns comentários sobre estes trabalhos na seção 2.3. Nosso objetivo é apresentar os conceitos necessários para contextualizar nossa proposta, desenvolvida no Capítulo 3, de uso de agentes para automatizar a escolha de índices em sistemas de bancos de dados relacionais.

2.1
Auto-sintonia de Bancos de Dados

A auto-sintonia de sistemas de bancos de dados refere-se à execução automática, por parte do sistema, das atividades de sintonia regularmente realizadas por um DBA. A necessidade de SGBDs capazes de auto-sintonia vêm se tornando mais presente, uma vez que novas aplicações demandam a adoção em maior escala deste tipo de sistema [33, 2]. Idealmente, deveríamos conseguir colher os benefícios do uso de SGBDs incorrendo na menor medida possível nos custos relativos à sua administração e ao gerenciamento de seu desempenho.

Diversos trabalhos procuram encontrar soluções automáticas para aspectos envolvidos na sintonia de SGBDs. Um estudo detalhado de trabalhos

[1] Nesta dissertação, iremos tratar agentes de *software* simplesmente pelo nome de agentes.

de auto-sintonia de bancos de dados presentes na literatura e em sistemas comerciais foi realizado dentro do contexto desta dissertação e documentado em [36]. Basicamente, é possível separar os trabalhos anteriores de pesquisa em auto-sintonia em dois grandes grupos, de acordo com o foco dado na abordagem do problema.

No primeiro grupo estão os trabalhos de auto-sintonia local, que procuram estudar problemas específicos de sintonia existentes nos sistemas atuais. Destes problemas podemos citar o projeto físico de bancos de dados (em especial, a seleção de índices para o sistema), a alocação de dados entre diferentes elementos de armazenamento, o controle de carga, o refino de estatísticas utilizadas pelo otimizador, a substituição de páginas e o ajuste de áreas de memória. Cada trabalho de auto-sintonia local enfoca um destes problemas, analisando vantagens e desvantagens de uma solução que reduz ou elimina a intervenção humana.

No segundo grupo estão os trabalhos de auto-sintonia global, que procuram estudar como é possível tomar ações de sintonia que tragam benefícios de desempenho para o sistema como um todo. Este tipo de abordagem procura encontrar um equilíbrio entre as diversas considerações locais de sintonia de forma a alcançar um desempenho global que seja melhor do que o que seria alcançado caso cada componente do sistema tomasse decisões de sintonia isoladamente. Um exemplo seria o de considerar ações de criação de índices em tabelas pequenas por motivos de concorrência. Ainda há poucos trabalhos nesta linha de pesquisa e, de fato, bem poucos trazem algum resultado experimental, mesmo quando consideramos trabalhos que lidam com sistemas que não são SGBDs. Isto pode estar relacionado ao fato das inter-relações entre os diversos componentes de um SGBD não serem bem conhecidas [64]. O trabalho de Milanés [42], de nosso grupo de pesquisa, procura explorar o uso de agentes para realizar auto-sintonia global.

Esta dissertação desenvolve pesquisa referente ao primeiro grupo. Apresentamos um trabalho de auto-sintonia local que procura investigar como agentes podem ser utilizados para automatizar a escolha de índices em um sistema de banco de dados. A seguir, abordaremos a metodologia empregada por outras propostas da literatura para implementar ferramentas de seleção de índices, uma vez que este é o foco de estudo de nosso trabalho. Referimos o leitor a [36] para uma melhor visão de outros trabalhos correlatos que procuram tornar a sintonia de SGBDs um processo mais independente de intervenção humana.

Metodologia para Seleção de Índices

Os vários trabalhos da literatura que tratam de seleção de índices [36] procuram construir ferramentas de apoio ao DBA na escolha dos índices para uma determinada carga de trabalho $W = \{(Q_i, f_i), i = 1, \ldots, n\}$. Q_i representa um comando SQL e f_i sua freqüência de submissão. Estas ferramentas implementam algoritmos que objetivam minimizar o custo total de processamento de W pelo sistema respeitando um limite de espaço S disponível para a criação de índices. O custo total de processamento é definido pela soma dos custos de acesso e atualização dos dados e dos custos de manutenção dos índices. Já foi demonstrado que uma versão restrita do problema de seleção de índices é NP-difícil [15][2].

Nesses mesmos trabalhos podemos perceber uma metodologia comum para a escolha de índices que se adequam a uma carga de trabalho. Esta metodologia é mostrada na Figura 2.1.

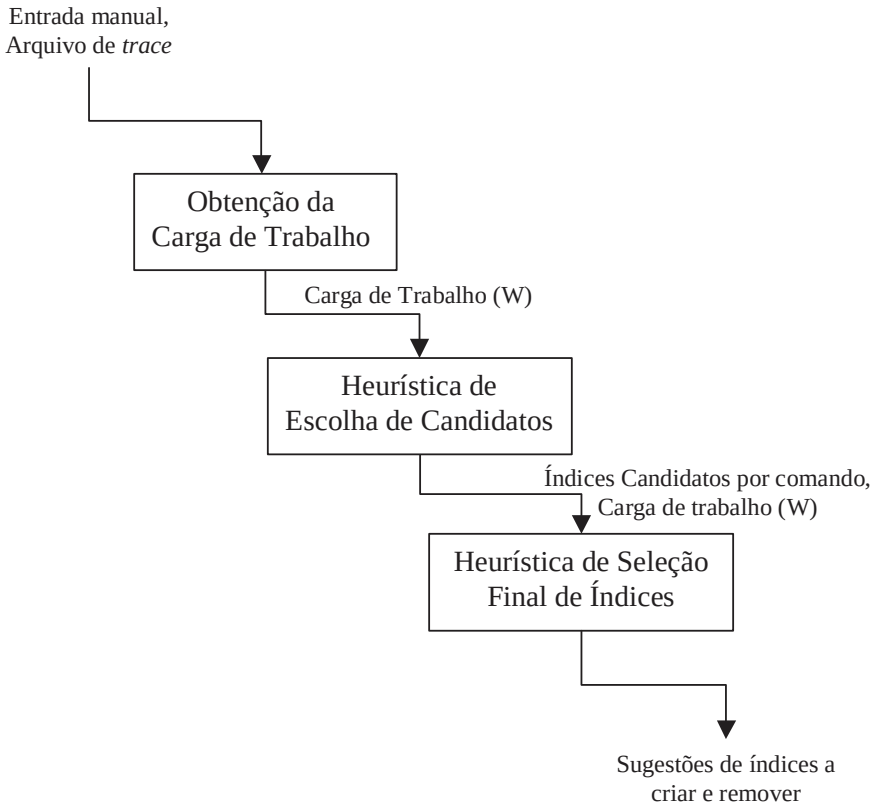

Figura 2.1: Metodologia para seleção de índices

O primeiro passo para a seleção de índices é obter a carga de trabalho W sobre a qual a ferramenta irá operar. Normalmente, as ferramentas

[2]Considera-se apenas uma relação e o problema de encontrar um índice único com o menor número possível de colunas.

possuem duas formas de aquisição dos comandos SQL e suas freqüências de execução: a entrada manual e a utilização de arquivos de *trace*. Na entrada manual, o DBA insere na ferramenta uma lista de comandos SQL e suas freqüências estimadas. Já na coleta de *trace*, o DBA usa um utilitário do sistema que permite o registro, em um arquivo, de estatísticas sobre todos os comandos SQL submetidos pelos usuários. Esse arquivo é processado pela ferramenta para detectar quantas vezes os comandos SQL foram executados durante o período de coleta.

Em alguns sistemas também é possível obter os comandos recentemente executados no sistema através da varredura do *cache* reservado para operações SQL [45, 37]. É importante notar que, seja qual for o método empregado, há uma suposição de que a obtenção da carga de trabalho depende fundamentalmente de uma interação humana com a ferramenta, uma vez que essa é uma entrada do problema de seleção de índices.

Em seguida, para cada comando da carga de trabalho, é aplicada uma heurística de escolha de índices candidatos, que irá determinar os melhores índices para cada comando SQL da carga de trabalho. Na literatura existem diversas abordagens para a construção de heurísticas de escolha de candidatos. Podemos citar as estratégias baseadas em conhecimento (ou regras) [7, 49] e as estratégias baseadas em custos do otimizador [25, 10, 37].

No primeiro tipo de abordagem, a escolha de quais índices são interessantes para um comando SQL é baseada no uso de regras derivadas a partir do conhecimento de especialistas. Um exemplo de regra seria sempre criar um índice para consultar uma tabela com mais do que 50 blocos e filtrada por um predicado que obtém menos do que 5% das tuplas.

Já na segunda abordagem, o próprio otimizador de consultas do sistema é utilizado para classificar quais índices poderão trazer maior benefício para o comando. Esta abordagem apresenta como principal vantagem a característica de evitar que a ferramenta crie um modelo de custos separado do modelo usado pelo otimizador. Isto permite garantir que os índices criados serão realmente considerados pelo sistema. Também auxilia na manutenção da ferramenta de seleção, uma vez que não é necessário realizar a evolução de um modelo separado. Por outro lado, o conhecimento de especialistas, em algumas situações, pode ser superior ao embutido no otimizador, como é comprovado pela existência de mecanismos de *dicas* (*hints*) em sistemas comerciais [41, 45]. Estes mecanismos poderiam ser utilizados em conjunto com a ferramenta de seleção de índices para forçar o uso de índices que não fossem escolhidos pelo otimizador.

Conforme veremos no Capítulo 3, nosso trabalho está mais relacio-

nado às técnicas baseadas em custos do otimizador, tanto pelas vantagens apresentadas acima como pelo fato de já serem aplicadas em SGBDs existentes. Por exemplo, Chaudhuri e Narasayya [10] utilizaram o SGBD Microsoft SQL Server e Lohman et al. [37] trabalharam sobre o IBM DB2 UDB. Outra característica relevante é que estas técnicas tendem a exigir menor intervenção humana para uso dos índices escolhidos, uma vez que estes foram determinados utilizando o próprio otimizador do sistema.

Por fim, após a escolha dos índices candidatos, é aplicada uma heurística final de seleção de índices que procura determinar quais índices oferecem a melhor relação custo-benefício para a carga de trabalho como um todo. Este tipo de heurística pode levar em conta o compromisso existente entre os custos de manutenção dos índices e as melhorias trazidas pelos mesmos em consultas. Em diversos trabalhos, como é o caso de Lohman et al. em [37], também é considerada a restrição de espaço máximo disponível para a criação de índices (S). Nestes trabalhos, o problema de escolher quais índices candidatos, com benefício positivo para a carga de trabalho, devem ser criados, dado um espaço máximo, é comumente modelado como uma variante do clássico Problema da Mochila [27].

Após a aplicação da heurística final de seleção de índices, as ferramentas recomendam criações ou remoções de índices para o DBA, que irá decidir se, e quando, estas recomendações devem ser efetivadas. É importante reparar que, nas ferramentas de seleção de índices existentes, nem todas as etapas necessárias para a auto-sintonia de índices são automatizadas, como é o caso da obtenção da carga de trabalho e da efetiva criação ou remoção dos índices escolhidos.

2.2 Agentes

Em princípio, SGBDs são sistemas passivos, que somente executam ações em resposta às requisições de seus usuários. O conceito de SGBD Ativo foi desenvolvido como uma opção para aumentar o grau de reatividade dos sistemas. Os SGBDs Ativos utilizam o paradigma de processamento de regras do tipo evento, condição e ação. Estas regras são compiladas e mantidas como objetos do SGBD.

Em [3] são discutidas as semelhanças e diferenças entre SGBDs Ativos e agentes. As regras ativas provêem um mecanismo para que aplicações reajam a eventos ocorridos no SGBD. Elas não permitem, entretanto, que componentes internos do SGBD, como o gerente de *buffers*, sejam

acessados e reconfigurados. Assim, não podem ser em geral utilizadas para a auto-sintonia do sistema. O paradigma de agentes traz a característica de reatividade associada à possibilidade de ação sobre componentes que estão fora do contexto oferecido pelo SGBD a seus usuários.

Para os propósitos do nosso trabalho, consideramos que um agente é um elemento autônomo, adaptativo e interativo do domínio, cujas instâncias são representadas através de conceitos mentais como crenças, metas, planos e ações [56]. Um agente atua como um processador de planos de ações que são executados utilizando suas crenças de modo a alcançar uma meta. Uma meta é um objetivo que o sistema deve alcançar; uma crença é algum conhecimento sobre o sistema.

Os agentes são bastante usados em domínios de aplicação em que é necessário que os sistemas possuam um alto grau de reatividade. Como exemplos, podemos citar controle de tráfego aéreo, controle de processos industriais e simuladores [66]. O uso de agentes permite que tais sistemas percebam o ambiente em que estão inseridos e tomem ações de acordo com as mudanças ocorridas neste ambiente, sem intervenção humana.

Para a auto-sintonia de sistemas de bancos de dados, os fatores de autonomia e reatividade são de grande importância, uma vez que o sistema deve se adaptar de forma automática e dinâmica à carga de trabalho que lhe é submetida. Assim, o uso de agentes pode se revelar interessante neste contexto. Nas próximas subseções, descrevemos alguns trabalhos correlatos que procuram explorar este relacionamento e mostramos arquiteturas que permitem o uso de agentes em SGBDs.

Uso de Agentes para Realizar Auto-sintonia

Agentes foram utilizados para a auto-sintonia de sistemas computacionais por Bigus et al. em [4, 19], que estudaram a sintonia do servidor de *e-mail* Lotus Notes e a do servidor *web* Apache, respectivamente. Nestes trabalhos, é utilizado o *framework* de agentes AutoTune, que permite controlar a alocação de recursos dos sistemas objetivo através da alteração de parâmetros de sintonia pautada por políticas definidas pelo administrador do sistema. No AutoTune, não é necessário o conhecimento prévio do sistema objetivo. O modelo do sistema é explicitamente adicionado ao *framework* através de um agente de modelagem. Nas implementações realizadas, este agente de modelagem foi construído pelo treinamento de uma rede neural com respostas geradas pelos sistemas objetivo quando são feitas alterações nos seus parâmetros de sintonia. No presente momento, estudos

da aplicação deste *framework* a SGBDs, que são sistemas de maior grau de complexidade do que servidores *web* e de *e-mail*, estão sendo conduzidos [18].

Quando tratamos de auto-sintonia de sistemas de bancos de dados, nosso objetivo central é construir sistemas mais autônomos, capazes de tomar decisões de sintonia de forma pró-ativa e automática. Em nosso grupo de pesquisa, investigamos como aplicar a abordagem de agentes a SGBDs de forma a torná-los mais extensíveis e capazes de auto-configuração e de auto-sintonia. Os trabalhos anteriores de [8], [42] e [40] seguem esta linha de pesquisa.

Nosso trabalho é um desenvolvimento das idéias lançadas por Costa e Lifschitz em [8], onde se discute como agentes poderiam ser utilizados para incluir a propriedade de auto-sintonia em sistemas de bancos de dados. São apresentados modelos para uso de agentes na construção de componentes de auto-sintonia local e global. Em especial, discute-se uma heurística interessante para inclusão em um agente para sintonia de índices.

O trabalho de Milanés em [42] propõe arquiteturas de agentes para auto-sintonia global de sistemas de bancos de dados. Agentes com escopo local são utilizados para realizar a sintonia de componentes individuais, como o gerente de bloqueios ou o de memória. Além destes, é definido um agente coordenador e um agente de histórico. O agente coordenador recebe informações de todos os agentes de escopo local e interage com o agente de histórico para consultar uma base de decisões de sintonia anteriormente tomadas. As ações de sintonia locais somente são executadas quando ocorre a aprovação do agente coordenador, o que visa garantir que ações conflitantes, redutoras do desempenho global, não sejam executadas. Os agentes voltados para a sintonia de índices, propostos no Capítulo 3, podem ser incorporados na arquitetura global discutida em [42].

Já Macêdo, em [40], estuda arquiteturas para integrar agentes e SGBDs. É realizada a implementação de um agente para balanceamento de carga de junções paralelas utilizando uma das arquiteturas de integração propostas. Iremos descrever em mais detalhe estas arquiteturas de integração a seguir.

Integração de SGBDs com Agentes

O trabalho de Macêdo [40] discute como a abordagem de agentes pode ser aplicada ao contexto de sistemas de bancos de dados. Em especial, são

apresentadas três arquiteturas para a integração de agentes e SGBDs. Estas arquiteturas são mostradas na Figura 2.2.

Figura 2.2: Integração de Agentes com SGBDs

A arquitetura *em camadas* permite a implementação de agentes sobre SGBDs já existentes com pouca ou nenhuma necessidade de modificação do código do sistema. Apesar desta relativa facilidade de integração, a capacidade de implementação de componentes de auto-sintonia utilizando agentes desta forma é limitada pelas interfaces oferecidas pelo SGBD para modificação de seus parâmetros de sintonia e para acesso a informações de seus componentes internos. A profundidade dos ajustes possíveis e a qualidade das informações que são expostas variam entre diferentes SGBDs. Alguns fornecedores de sistemas comerciais para monitoramento de desempenho de bancos de dados utilizam a abstração de agente em uma arquitetura em camadas para construir seus monitores [36]. As ações de sintonia, caso necessárias, normalmente precisam ser tomadas por um DBA. Assim, estes sistemas não podem ser caracterizados realmente como sistemas que oferecem auto-sintonia.

A arquitetura *integrada* preconiza que o SGBD seja construído como um sistema multi-agentes [65]. Os componentes do SGBD – todos ou em sua maioria – são substituídos por agentes, que se tornam responsáveis por todas as tarefas de gerência dos dados e de auto-sintonia. Este tipo de arquitetura segue a noção de construir um sistema que possui auto-sintonia global por projeto [36]. Até o momento, esta abordagem não foi implementada, justamente pela complexidade de construir um novo SGBD.

Por fim, a arquitetura *embutida* representa um ponto médio entre os dois estilos de arquitetura descritos anteriormente. Nesta abordagem, o sistema de agentes possui acesso direto aos componentes do SGBD e pode interferir no seu comportamento dinamicamente. Esta capacidade é fundamental para a construção de componentes de auto-sintonia. Por outro lado, a implementação deste tipo de arquitetura exige que o código do SGBD esteja disponível e que o desenvolvedor obtenha o conhecimento sobre o

funcionamento interno dos componentes com que o sistema de agentes irá interagir. Este estilo de integração foi utilizado na implementação de [40] com o SGBD Minibase [50] e é o utilizado em nossa implementação com o SGBD de código aberto PostgreSQL (veja o Capítulo 4).

Implementação de Agentes

Existem várias arquiteturas possíveis para a implementação de agentes: baseada em lógica, reativa, crença-desejo-intenção e em camadas de *software* [66]. Além disto, é comum que o desenvolvedor de um sistema de agentes se utilize de algum *framework*, concreto ou abstrato, para agilizar a tarefa de construção [40].

Em nosso trabalho, seguiremos a abordagem de implementação de Macêdo [40], que utiliza uma arquitetura em camadas de *software* baseada no *framework* abstrato proposto por Kendall et al. [35]. A arquitetura empregada em [40] já foi utilizada para construir agentes integrados com SGBDs. Apresentamos a sua estrutura na Figura 2.3.

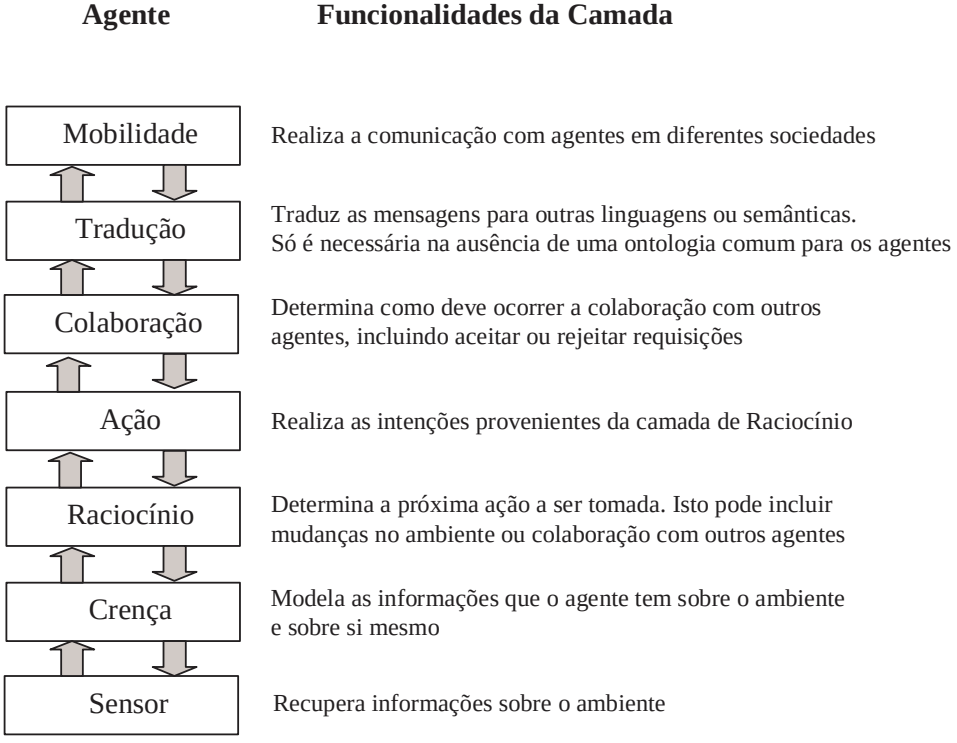

Figura 2.3: *Framework* de construção de agentes de [35]

As camadas são estruturadas de acordo com as funcionalidades necessárias para a operação do agente. O estilo de implementação em cama-

das tem como vantagem o fato de cada camada depender apenas de suas camadas vizinhas, o que facilita o desenvolvimento e a depuração.

Existem dois fluxos de informação no agente: um *bottom-up* e um *top-down*. No fluxo *bottom-up*, o agente obtém informações do ambiente a partir de seus sensores e atualiza as suas crenças. Com base nos novos valores das crenças, o agente avalia qual será o seu plano de ações. As ações escolhidas podem envolver uma atuação sobre o ambiente, implementada por efetuadores, ou a colaboração com outros agentes. No caso de uma colaboração, mensagens são criadas, codificadas e enviadas.

O fluxo *top-down* pode ser ativado quando o agente recebe uma mensagem de outro agente ou quando obtém um retorno direto de alguma ação. No caso de uma mensagem, esta será decodificada. Em ambos os casos, o agente irá considerar se alguma atitude – atualização de crenças, obtenção de informações do ambiente, ou instanciação de uma intenção – deve ocorrer.

O uso do *framework* originalmente proposto em [35] na implementação de agentes para auto-sintonia de índices será descrito nos Capítulos 3 e 4.

2.3 Comentários Finais

Este capítulo revisou alguns trabalhos nas áreas de auto-sintonia de SGBDs e de agentes diretamente relacionados com a dissertação. As propostas de auto-sintonia presentes na literatura podem ser classificadas como de auto-sintonia local ou global. Iremos focar nossa atenção na auto-sintonia local, mais especificamente no problema de determinar e construir automaticamente os índices que melhor se adequam à carga de trabalho submetida ao SGBD.

Para a seleção de índices, os trabalhos anteriormente realizados propõem ferramentas que seguem três grandes etapas: a obtenção da carga de trabalho, a aplicação de uma heurística de escolha de índices candidatos e a aplicação de uma heurística de seleção final de índices. Essas ferramentas supõem que algumas atividades do processo de auto-sintonia de índices serão executadas manualmente pelo DBA, como é o caso da obtenção da carga de trabalho e a criação (ou destruição) final dos índices.

Acreditamos que o uso de agentes pode trazer mais autonomia e reatividade a SGBDs. No próximo capítulo, iremos apresentar nossa proposta de uso de agentes para auto-sintonia de índices em SGBDs relacionais. Iremos utilizar a arquitetura embutida de integração discutida neste capítulo,

bem como o *framework* de [35] para a construção de agentes. Apresentaremos duas arquiteturas em que agentes obtêm informações de execução de comandos SQL do SGBD, interagem com o otimizador do sistema para avaliar índices interessantes para a carga de trabalho e realizam a criação ou remoção de índices automaticamente.