Estudo e Elaboração de um Conjunto de Regras para Sintonia de Performance em Aplicações de Banco de Dados

Simone R. Vicari e Cirano Iochpe
Instituto de Informática
Universidade Federal do Rio Grande do Sul
Bento Gonçalves, 9500 - Bloco IV
91501-970 Porto Alegre - RS
Fax: (051) 336-5576
e-mail: {simonerv| iochpe}@inf.ufrgs.br

Comentario [SRV1]:

Resumo

Este artigo tem por objetivo apresentar parte do estudo realizado sobre sintonia de performance de banco de dados que consiste em levantar e propor um conjunto de regras que melhore o desempenho de aplicações de banco de dados. São apresentadas algumas considerações sobre sintonia de performance e os aspectos de implementação de sistema de banco de dados que causam impacto no desempenho da aplicação. As regras estudadas referem-se à arquitetura do sistema gerenciador de banco de dados ORACLE - utilizado como estudo de caso - e uma delas é analisada para ilustrar o problema. Além disso, é definido um conjunto de cenários que representam contextos específicos de aplicações reais e que serão utilizados para validar a aplicabilidade das regras propostas. O conjunto de regras será testado para o conjunto de cenários através de um protótipo de ferramenta de tuning baseada em conhecimento. Tal ferramenta deverá auxiliar o administrador de banco de dados no diagnóstico de problemas de performance.

Palavras-chave: *bancos de dados, tuning, sintonia de performance, sistema baseado em regras*

Abstract

The purpose of this article is to present a part of a study on database performance which consists of a survey of tuning situations and propose of a set of rules to improve performance of database applications. Some considerations about tuning will be done and the aspects of database system that cause impact on application performance will be presented. The studied rules rely on the ORACLE database management system architecture that has been used as a case study. Furthermore, one of rules is detailed in order to ilustrate the problem. In addition, a set of database production scenarios is defined which represents specific applications contexts that will be used to validate the proposed rules. The development of a prototype relying on a knowledge based diagnosys tool is under way. It shall be able to help database administrators in the identification of performance problems.

Key words: *database systems, performance tuning, rule-based systems*

1 Introdução

Aplicações corretas que apresentam bom desempenho representam o objeto de trabalho de analistas e programadores especializados. No entanto, o desenvolvimento das tecnologias de computador como, por exemplo, a ampla utilização de redes de computadores e sistemas distribuídos, tem contribuído para aumentar a complexidade dos sistemas, na medida que mais dados precisam ser gerenciados e consultas mais complexas precisam ser processadas. Sendo assim, por melhor que seja o projeto, problemas de performance tendem a ocorrer mais cedo ou mais tarde.

Para medir o desempenho de uma aplicação, são utilizadas duas métricas: tempo de resposta (tempo médio necessário para que uma tarefa seja concluída) e *throughput* (volume de tarefas executadas por unidade de tempo).

Normalmente, os usuários da aplicação são os primeiros a perceberem a degradação da performance do sistema. No entanto, identificar a causa da mesma e resolver o problema é uma tarefa bem mais difícil. A atividade de fazer com que uma aplicação de banco de dados apresente um melhor desempenho é chamada de *tuning* de banco de dados ou sintonia de performance [SHA 92]. Para melhorar a performance de uma aplicação é necessário analisar toda a aplicação desde o projeto do banco de dados, passando pelo código dos programas, até a execução das consultas, pois estes aspectos podem causar um forte impacto na performance das aplicações gerando os chamados "gargalos".

Alguns sistemas gerenciadores de banco de dados (SGBD), disponíveis no mercado, apresentam na estrutura do seu dicionário de dados, tabelas que armazenam estatísticas de performance da aplicação [SHA 92, ORA 92a]. Essas estatísticas, normalmente, são utilizadas pelos otimizadores de consultas e por ferramentas oferecidas pelo próprio SGBD para auxiliar no diagnóstico de problemas de performance. Os resultados oferecidos por estas ferramentas, porém, se encontram em um estado bruto, ou seja, devem ser analisados e interpretados pelo administrador de banco de dados (DBA) para que este, então, identifique os problemas no sistema e proponha alternativas para a solução dos mesmos.

Complementarmente, a maioria dos SGBDs comerciais apresentam um conjunto de parâmetros de configuração que permite configurar características do sistema como, por exemplo, tamanho dos *buffers* de dados, granularidade de bloqueios (*locks*), freqüência de *checkpoints.* Esses parâmetros são indicados ao sistema pelo DBA e, se configurados adequadamente, podem contribuir para um melhor desempenho da aplicação [ORA 92a, ORA 92c].

Percebe-se, a partir do que foi apresentado, até aqui, que o DBA possui um papel importante na sintonia de performance, pois é ele quem analisa e interpreta as estatísticas geradas pelo SGBD, bem como, determina alternativas para solução dos gargalos. Para isso o DBA segue uma seqüência de ações, baseado em seus conhecimentos e experiência, para identificar e tentar resolver os problemas de performance presentes na aplicação. Estas ações podem, então, ser definidas sob a forma de regras e, a partir delas, pode-se monitorar a aplicação, identificando gargalos e avaliando alternativas para tratamento de uma forma mais controlada.

Este artigo apresenta parte de um trabalho de pesquisa que se propõe a identificar um conjunto de regras para sintonia de *performance* capaz de auxiliar o trabalho do DBA. Para isso, está sendo desenvolvido um estudo de caso com base no SGBD ORACLE e na bibliografia disponível sobre sintonia de performance em sistemas de banco de dados. Os exemplos apresentados neste artigo são todos relativos àquele SGBD.

Para melhor avaliar em que situação as regras devem ser aplicadas, está sendo definido, também um conjunto de cenários equivalentes a contextos específicos de aplicações reais.

Cada regra deverá ser testada em diferentes cenários para avaliar sua eficácia em cada situação distinta.

Ao final desta pesquisa pretende-se estar em condições de construir uma ferramenta baseada em regras que seja capaz de identificar, no ambiente de produção do banco de dados, um determinado cenário de execução e, a partir daí, aplicar um conjunto de regras a ele associadas para diagnosticar possíveis gargalos e propor alternativas de configuração do SGBD ao DBA.

O restante do artigo está organizado como segue. A seção 2 apresenta os principais aspectos da arquitetura de banco de dados que tem impacto na performance das aplicações. A seção 3 apresenta um exemplo de regra e sua avaliação. Na seção 4 são apresentadas as características dos cenários construídos para validação das regras. A seção 5 apresenta a arquitetura básica do protótipo da ferramenta implementada. Finalmente a seção 6 conclui o artigo.

2 Aspectos de Implementação que Influenciam na Performance

A forma como são implementadas algumas características dos sistemas de banco de dados pode influenciar a performance das aplicações. Tais características devem ser conhecidas pelo DBA e pelos desenvolvedores de aplicações.

Um exemplo dessa situação pode ser observado no processamento de consultas. Dependendo de como é construída uma consulta, o SGBD pode dispor de um ou mais caminhos de acesso aos dados solicitados, cada um deles oferecendo desempenho maior ou menor. A figura 1 representa esta situação:

consulta 1:	*consulta 2:*
SELECT *	SELECT *
FROM <nome_da_relação>	FROM <nome_da_relação>
WHERE <atributo> = <valor>	WHERE <atributo + 10> = <valor>

Figura 1: Exemplo de consultas SQL

Na primeira consulta, se existir um índice definido para <atributo> então este será utilizado; na segunda consulta a presença de um índice será ignorada porque uma expressão é utilizada na cláusula WHERE, fazendo com que o caminho de acesso que utiliza o índice não seja disponibilizado. Pode-se observar que ambas as consultas retornam o mesmo conjunto de dados com a diferença de que a primeira consulta executará mais rapidamente que a segunda devido à utilização do índice.

Em [VIC 95] foi realizado um estudo a respeito dos aspectos de implementação de banco de dados que influenciam a performance das aplicações e foi apresentado os principais problemas relacionados a cada um desses aspectos. Segundo este trabalho os aspectos de implementação podem ser classificados em:

- gerenciamento do meio de armazenamento principal;
- gerenciamento do meio de armazenamento secundário;
- gerenciamento do sistema de controle de concorrência;
- gerenciamento do sistema de recuperação após falhas (*recovery*);
- utilização de índices;
- otimização de consultas.

As seções seguintes apresentam uma breve descrição dessa classificação.

2.1 Gerenciamento do Meio de Armazenamento Principal

A velocidade de acesso à memória principal é maior que aquela de acesso aos discos, por exemplo. No entanto sua capacidade de armazenamento é, normalmente, menor, sendo praticamente impossível manter todo o banco de dados em memória principal.

Para alocar mais memória que o disponível, o sistema operacional executa operações de *swap* e paginação, que fazem com que partes da memória sejam copiadas (movidas) para o meio secundário e vice-versa (memória virtual). Estas operações de leitura e gravação, no entanto, são bastante lentas e podem degradar a performance do sistema no caso de ocorrerem freqüentemente.

Para o *tuning*, devem ser observados o tamanho da memória virtual, a quantidade de memória alocada para cada usuário e para o SGBD e o tamanho dos *buffers*. Se estas áreas tiverem um tamanho adequado para as características da aplicação o número de paginações e *swaps* necessários pode ser minimizado, contribuindo para o melhor desempenho do sistema.

2.2 Gerenciamento do Meio de Armazenamento Secundário

Toda vez que um dado não presente na memória principal é requisitado, um acesso à memória não volátil (ex.: disco) é feito. A diferença entre o tempo de acesso à memória principal e aquele referente à memória secundária pode ser da ordem de um milhão de vezes. Esta diferença de tempo acaba se refletindo no tempo de resposta da aplicação. Mais ainda, o aumento não controlado dos acessos à memória secundária pode provocar o fenômeno conhecido como contenção de disco, quando o *throughput* é prejudicado devido ao aumento no tempo de espera, por acesso a disco, de cada programa de aplicação.

A alternativa neste caso, consiste em distribuir as operações de entrada e saída na memória secundária (I/O) em uma ou mais unidades de armazenamento, colocando, por exemplo, tabelas, índices e arquivos de *log,* em discos separados.

Outra técnica que pode ser utilizada é a de dividir uma tabela de dados em pequenas porções e armazená-las em arquivos físicos independentes e em discos diferentes (*stripping table)*. Esta técnica é interessante principalmente para grandes tabelas, com grande volume de acessos concorrentes. Como a tabela está dividida em porções, cada transação pode acessar uma destas partes independentemente, evitando assim contenção de dados.

2.3. Gerenciamento do Sistema de Controle de Concorrência

O controle de concorrência é a atividade de coordenar a ação de processos que operam em paralelo, acessam dados compartilhados, e portanto, interferem uns com os outros [BER 87].

O mecanismo utilizado para realizar este controle, baseado geralmente na técnica de *two-phase-locking* (bloqueio em duas fases), são encontrados em, praticamente, todos os SGBDs transacionais do mercado. Dependendo da forma como são administradas e como são aproveitadas algumas facilidades, pode-se melhorar consideravelmente a performance da aplicação.

Alguns cuidados no projeto da aplicação contribuem significativamente para um melhor aproveitamento do controle de concorrência. Quando todas as transações são de leitura ou quando uma única transação roda sobre o banco de dados, a cada instante, o sistema de *locking* pode ser completamente desativado, evitando, com isso, que seja feito o controle de *locks* sem necessidade. O ganho de performance neste caso não é muito grande, mas deve ser explorado.

Em [SHA 92], são relacionados os seguintes pontos a serem observados:

a) tamanho das transações: transações longas e que acessam muitos dados tendem a requerer muitos bloqueios e a segurá-los por muito tempo, fazendo com as outras transações tenham que esperar pela liberação dos mesmos, diminuindo assim o grau de concorrência;

b) enfraquecimento dos níveis de consistência: refere-se ao fato de se utilizar níveis de isolamento que permitem que dados de transações ainda não concluídas sejam utilizados por outras transações;

c) burlar *hot spots*: refere-se a tentar retardar o bloqueio de dados muito acessados dentro de uma transação;

d) granularidade dos bloqueios (*locks*): refere-se à porção de dados que é bloqueada por uma transação, por exemplo, bloqueios por tabela, por página e por tupla. Diminuindo a granularidade aumenta-se o paralelismo. Aumentando-se a granularidade dos bloqueios, diminui-se o tempo de acesso à tabela de *locks*;

e) freqüência de testes de deadlocks: intervalo de tempo entre dois testes de *deadlock* entre transações. Quanto maior a freqüência, menos tempo para liberar recursos (dados) bloqueados por transações que estão em *deadlock*. Quanto menor a freqüência, maior a probabilidade de desperdiçar recursos de máquina para identificar a inexistência de *deadlock*.

2.4. Sistema de Recuperação após Falhas (Recovery)

Sistemas de computação, estão sujeitos a falhas. Estas podem ser de vários tipos como, por exemplo, falha de transação, falha do sistema operacional ou do SGBD, falha dos meios de armazenamento não voláteis. Em caso de falhas é necessário que se tenha algum tipo de redundância, de modo a restaurar o estado do banco de dados para aquele anterior à falha. A parte do sistema de banco de dados, responsável pela restauração do estado consistente do banco de dados após uma falha, é conhecida como componente de *recovery*.

O mecanismo de *recovery* gera trabalho extra ao SGBD (*overhead*) em dois momentos distintos, onerando o tempo de resposta de cada transação e competindo para reduzir o *throughput*. Por um lado, este mecanismo coleta, durante o tempo de execução normal do sistema, uma série de informações sobre o processamento das transações. Isso já representa um *overhead* significativo. Por outro lado, o *recovery*, praticamente monopoliza os recursos da máquina em tempo de falha do sistema de banco de dados.

Aspectos importantes para a performance do SGBD, associados ao *recovery,* são: a localização do arquivo de *log*, a estratégia de gerenciamento de *buffers*, freqüência de *checkpoints* e *database dumps,* e o tamanho das transações (em número de operações de leitura e gravação de dados).

2.5 Utilização de Índices

Um índice é uma estrutura de dados auxiliar, que permite acesso aleatório mais rápido aos registros existentes em um arquivo. Cada índice está associado a uma chave de pesquisa particular, que corresponde a um atributo ou conjunto de atributos que identificam o registro dentro da tabela.

O uso de índices pode melhorar o tempo de resposta para muitas consultas, no entanto, apresenta um custo adicional para o sistema. Índices requerem espaço no meio de armazenamento. Requerem, também, manutenção, ou seja, cada vez que um registro é incluído ou excluído de uma tabela, ou é alterado o valor do atributo chave, os índices devem ser atualizados. Isso significa que é necessário processamento e acessos a disco adicionais para executar essas operações.

A maior parte dos SGBD oferece diferentes tipos de índices que podem ser utilizados e, dependendo do tipo de consultas que serão processadas, um tipo pode ser mais adequado que outro. Por exemplo para a consulta: SELECT nome FROM Empregados WHERE código = 8478, é mais indicado a utilização de um índice baseado em *hash,* uma vez que a consulta pode ser atendida com um único acesso a disco caso não exista a lista de colisões.

A utilização de índices é essencial para melhorar o tempo de resposta de uma aplicação, porém, a sua má utilização, pode também degradá-lo consideravelmente. Deve-se, portanto, observar atentamente, durante a especificação do projeto, os índices que serão definidos e sobre que atributos serão definidos. Com isso, pode-se evitar a criação de índices que serão mantidos, mas nunca utilizados.

2.6 Otimização de Consultas

A otimização de consultas, nesse contexto, refere-se principalmente a forma como podem ser construídas as consultas SQL e se obter uma melhor performance. A maior parte dos problemas reside em consultas que não fazem uso de índices e no uso inadequado de cláusulas SELECT, DISTINCT do SQL, por exemplo.

Para se identificar se uma determinada consulta está com problemas de acesso e classificação de tuplas, devido às causas acima, têm-se dois caminhos:

- monitorar o número de acessos a disco para identificar se o código executável da consulta está varrendo tabelas seqüencialmente ao invés de utilizar índices;
- analisar o plano de execução (*query plan),* gerado para a consulta, onde pode-se visualizar como a consulta será processada podendo-se verificar a utilização ou não de índices e uma possível ordenação onerosa de tuplas para eliminar duplicatas.

3 O Conjunto de Regras

Para sintonizar uma determinada aplicação o DBA deve monitorá-la e, uma vez identificados os gargalos do sistema, propor alternativas de configuração do SGBD e/ou codificação de consultas e atualizações.

Para monitorar a aplicação o DBA deve coletar estatísticas do uso do banco através da execução de ferramentas de diagnose do SGBD. Contudo, tanto as estatísticas geradas pelo banco quanto os resultados oferecidos pelas ferramentas representam dados brutos, não interpretados, que precisam ser analisados pelo próprio DBA. A eficácia deste processo depende, portanto, do conhecimento e da experiência do DBA.

Percebe-se que o DBA segue uma seqüência de passos, a partir dos quais ele consegue coletar as informações, interpretá-las, identificar os gargalos e propor soluções para o mesmo e testá-las. Essa seqüência pode ser vista como aplicação de uma ou mais regras que oferecem meios para que o DBA possa avaliar o sistema e resolver os problemas encontrados.

Através de um estudo bibliográfico [COR 95, ORA 92a, ORA 92c] e de entrevistas com DBAs foram identificadas algumas ações, geralmente, executadas durante a sintonia de performance e, a partir disso, estas foram definidas sob a forma de regras. Foi observado, ainda, que as regras estão muito relacionadas aos aspectos de implementação do sistema apresentados anteriormente.

O SGBD ORACLE foi escolhido para estudo de caso, pois é um produto bem difundido no mercado além de estar disponível nos laboratórios do Instituto de Informática e do CPGCC da UFRGS, o que facilitaria o andamento desta pesquisa. Em vista disso, as regras levantadas estão baseadas na arquitetura deste SGBD. Embora não se possa afirmar

ainda, acredita-se que várias das regras levantadas possam ser utilizadas em outros SGBDs desde que estes apresentem uma arquitetura semelhante àquela do ORACLE.

Para ilustrar a estrutura das regras, é apresentado um exemplo referente ao gerenciamento da memória principal alocada, pelo sistema operacional, ao SGBD.

3.1 Exemplo de Regra

Conforme apresentado nas seções 2.1 e 2.2, no momento de serem acessados para consulta ou atualização, os dados podem estar armazenados tanto na memória principal quanto na memória secundária. Porém, quanto mais dados solicitados pelas aplicações puderem ser encontrados diretamente na memória principal, menos acessos ao armazenamento secundário serão necessários, evitando-se com isso um aumento do número de operações de I/O (*overhead)* que são as principais causas de degradação de desempenho do SGBD.

No ORACLE, existem algumas estruturas internas que são manipuladas pelo sistema, entre elas estruturas de memória, que podem ter seu tamanho definido através de parâmetros de configuração específicos. Uma dessas áreas de memória é a *shared pool,* responsável por armazenar informações sobre o dicionário de dados e manter áreas reservadas para a execução de comandos SQL. Esta área é dividida em outras áreas menores chamadas *library cache* e *data dictionary cache* que armazenam, respectivamente, informações sobre comandos SQL e sobre o dicionário de dados como, por exemplo, nomes de tabelas e visões do banco de dados, tipo de dados utilizados nos atributos das tabelas e privilégios dos usuários.

As informações estatísticas a respeito da utilização dessas áreas estão armazenadas em duas tabelas: V$LIBRARYCACHE e V$ROWCACHE. Estas tabelas são conhecidas como tabelas dinâmicas de performance e são assim chamadas por armazenarem estatísticas relacionadas a performance do banco de dados e por serem criadas toda a vez que o banco é inicializado (*database startup*). O ORACLE mantém em seu dicionário de dados, basicamente, uma tabela para cada estrutura manipulada.

Tomando por exemplo a tabela V$LIBRARYCACHE, dois atributos devem ser observados: PINS e RELOADS. O primeiro atributo refere-se ao número de vezes que comandos SQL são executados, o segundo atributo indica o número de vezes que estas execuções resultaram em uma falta (*miss*) fazendo com que o ORACLE tenha que, implicitamente, analisar (*parse*) novamente o comando ou tenha que recarregar a definição de um objeto (por estar desatualizada ou não estar presente na *library cache*).

A razão entre a soma total de RELOADS por PINS deve ser menor que 1% (um por cento). Caso seja maior, uma das alternativas é aumentar o tamanho da *shared pool* através do parâmetro de configuração SHARED_POOL_SIZE [ORA 92, COR 95].

Traduzindo esta situação para uma regra temos:

SE *soma(RELOADS) / soma(PINS) > 0,01*

ENTÃO *deve-se aumentar o tamanho da shared pool*

configurar parâmetro SHARED_POOL_SIZE

Surge então uma questão: esta regra é válida para qualquer tipo de aplicação? A partir das estatísticas coletadas pode-se inferir que para que a razão entre RELOADS e PINS seja inferior a 1%, o valor de RELOADS deve tender a zero, ou seja, a aplicação deve executar os mesmos comandos sobre o mesmo conjunto de dados, com isso as definições já estarão na *library cache* evitando-se com isso que as definições precisem ser recarregadas e/ou os comandos analisados novamente.

Supondo, então, uma situação onde os comandos executados são diferentes e acessam dados diferentes. Neste caso, não haveria como evitar que os comandos fossem re-analisados, nem que as definições fossem recarregadas e, dificilmente, seria possível definir uma quantidade de memória adequada para esta situação.

Percebe-se com isso que é necessário analisar outros fatores além dos aspectos de implementação do sistema de banco de dados. O contexto da aplicação, isto é, suas características tem influência na aplicação das regras. Como o contexto da aplicação dificilmente pode ser modificado resta adequar os aspectos de implementação a estes contextos.

No exemplo acima, seria interessante, como medida organizacional, que programas de aplicação com funções similares executassem em bloco, para minimizar esforços de *parsing* do SGBD. Por outro lado, isso poderia causar retensão de dados devido ao controle de concorrência do sistema. O importante é definir, para cada aplicação, qual dos dois problemas é o mais grave: *parsing* freqüente ou bloqueios por causa de *locking*.

Para avaliar a aplicabilidade destas regras foi construído um conjunto de cenários que representam contextos específicos de aplicações reais. Sua definição é abordada na seção seguinte.

4 A Construção dos Cenários

Como dito anteriormente, as características da aplicação podem influenciar na performance e na configuração do sistema. As características da aplicação podem ser definidas pelo tipo de transações executadas pela aplicação e definem uma carga (*workload)* ou perfil de transação. Pode-se dizer, então, que a característica da carga associada às características de implementação do componente do sistema de banco de dados, mais a configuração do SGBD podem ocasionar gargalos no sistema.

As características da carga das transações podem ser definidas por vários parâmetros como, por exemplo:

- número de transações: volume de transações que são executadas pela aplicação na unidade de tempo;
- tamanho das transações: número de comandos presente nas transações, pode ser pequeno, médio ou grande;
- proporção de transações longas e transações curtas: número de transações de um determinado tipo (longa ou curta) dividido pelo total de transações;
- volume de operações de escrita e leitura das transações;
- tamanho do banco de dados: tamanho do banco de dados acessado pelas

transações;

- localidade de acesso: freqüência com que transações acessam um mesmo conjunto de dados.

A partir destes parâmetros pode-se caracterizar a carga executada e avaliar a sua influência nos diversos componentes do sistema de banco de dados. Como considerar todos esses parâmetros levaria a um número muito grande de combinações de cargas diferentes, selecionou-se dois aspectos: a localidade de acesso e o volume de operações de leitura/escrita que as transações executam. Estes parâmetros foram escolhidos porque foi possível identificar que vários aspectos de implementação do sistema de banco de dados se comportam de forma diferente quando esses parâmetros são modificados.

A escolha destes parâmetros levou à construção de quatro cenários básicos que podem ser identificados na figura 2.

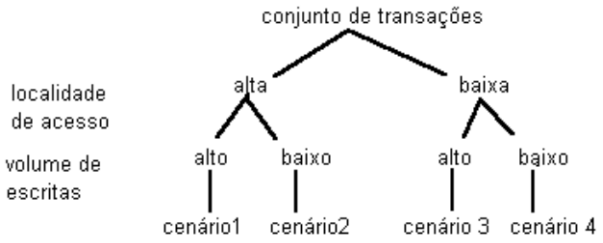

Figura 2: Cenários básicos

Os cenários básicos, considerando-se um volume médio de transações, são:

- cenário 1: alta localidade de acesso e alto volume de operações de escrita;
- cenário 2: alta localidade de acesso e baixo volume de operações de escrita;
- cenário 3: baixa localidade de acesso e alto volume de operações de escrita;
- cenário 2: baixa localidade de acesso e baixo volume de operações de escrita.

Para cada um destes cenários será observado a adequação das regras, levando-se em consideração os seguintes aspectos do sistema de banco de dados: gerenciamento de *buffers*, controle de concorrência e uso de *checkpoints*. Estes aspectos foram escolhidos por se ter disponível estatísticas a eles relacionadas e por razões de simplicidade, em vista do tempo e esforço necessários para a construção e o teste de todos os cenários que seriam gerados.

Com a validação das regras será possível tornar o processo de sintonia mais automatizado, além de se estruturar o conhecimento dos DBAs através das regras, mesmo que estas se encontrem, neste primeiro momento, ainda simplificadas.

5 Arquitetura da Ferramenta para Análise de Performance

Em vista das características das regras estudadas o processo de diagnose de performance pode ser dividido em quatro fases:

a) análise das estatísticas geradas pelo SGBD;
b) configuração dos parâmetros do SGBD a partir da aplicação das regras;
c) realimentação;
d) calibragem do sistema (*feedback*).

A fase de análise de estatísticas consiste em retirar do dicionário de dados do SGBD, as

estatísticas de performance geradas, a partir da execução de um conjunto de transações, e aplicar as regras existentes (base de regras). A partir disso é gerada uma nova configuração para o sistema.

O mesmo conjunto de transações deve ser, então, reexecutado utilizando-se a nova configuração, a fim de gerar novas estatísticas. Estas serão, então, comparadas com as estatísticas anteriores para se avaliar a confiabilidade das regras aplicadas.

Segue abaixo uma breve descrição sobre os módulos da arquitetura da ferramenta baseada em regras(Figura 3):

a) conjunto de transações executadas: corresponde ao conjunto de transações que se quer melhorar o desempenho. No caso deste protótipo é um dos cenários gerados;
b) extrator/gerador de estatísticas*:* tem por objetivo extrair estatísticas geradas pelas ferramentas de diagnose de performance do ORACLE, bem como gerar novas estatísticas a partir de dados brutos;
c) dicionário de dados: corresponde ao dicionário de dados do SGBD, onde estão armazenadas as estatísticas geradas durante o processamento das transações;
d) analisador de estatísticas: tem por objetivo analisar as informações obtidas no item (b), através da aplicação de regras armazenadas na base de regras (e). As estatísticas analisadas geram o modelo de configuração de performance (i) sugerido ao DBA para melhorar a performance da aplicação. Este modelo de configuração deve ser armazenado em uma base de configurações (f);
e) base de regras: é constituída por regras de avaliação/ação de performance*,* definidas para cada aspecto de implementação. É baseada no conhecimento e experiência do DBA e em informações existentes na bibliografia como, por exemplo, características técnicas do SGBD e do sistema operacional;
f) base de configurações: armazena os modelos de configuração gerados que serão utilizados em fase posterior para identificar se houve melhoria de performance;
g) analisador de acertos: recupera as estatísticas geradas anteriormente e as compara com as estatísticas geradas pelo sistema reconfigurado. A partir da comparação dos resultados é verificada a aplicabilidade das regras para esse conjunto de transações. Pode-se constatar que: (1) o conjunto de regras é aplicável, ou seja, o sistema teve sua performance melhorada; (2) o conjunto de regras não é aplicável, pois não houve melhoria de performance ou ocorreu degradação;
h) calibrador das regras: quando um conjunto de regras for considerado não aplicável, as regras desse conjunto deverão ser reavaliadas pelo módulo calibrador de regras e as alterações feitas serão transferidas para a base de regras. Ao final de sucessivas execuções sobre um determinado conjunto de transações, tem-se um conjunto de regras que são aplicáveis a este contexto e que garantem uma melhor performance para o sistema;
i) modelo de configuração: apresenta os aspectos que devem ser modificados na aplicação para melhorar a o desempenho das mesmas. Consiste em alterações de parâmetros do SGBD, do sistema operacional e da estrutura do dicionário de dados. Pode ser apresentado na forma de arquivo de configuração a ser utilizado pelo SGBD ou na forma de relatório.

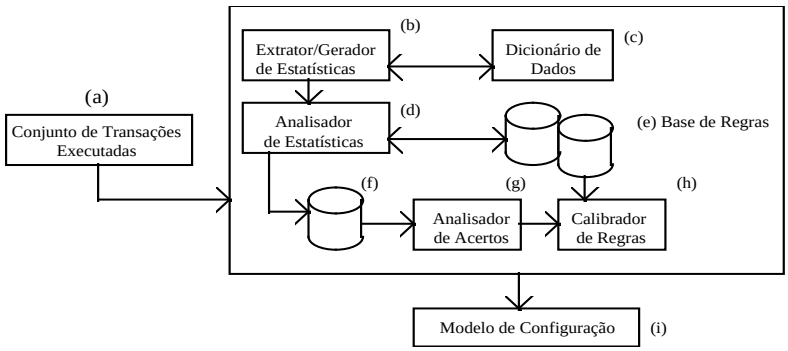

Figura 3. Arquitetura da ferramenta para análise de performance

6 Conclusões e Trabalhos Futuros

Neste artigo, foi apresentado algumas considerações sobre sintonia de performance e os aspectos de implementação de sistemas de banco de dados que causam impacto no desempenho das aplicações. Foi ressaltada a importância do papel do administrador de banco de dados na interpretação das estatísticas fornecidas pelo sistema para a identificação e solução dos gargalos. Foi apresentado, também, que as ações normalmente executadas pelos DBAs podem ser definidas sob a forma de regras e que estas podem auxiliar o trabalho de identificação e solução de gargalos do sistema. Um exemplo de regra foi apresentado para ilustrar que tanto os aspectos de implementação quanto a característica da carga das transações podem influenciar a aplicação da regra. Ao fim do artigo, foi definido as características dos cenários a serem construídos. O objetivo desses cenários é representar contextos específicos de aplicações para que se possa validar a aplicabilidade das regras.

Em conjunto às regras, está sendo desenvolvido protótipo de uma ferramenta baseada em conhecimento que utilize tais regras para oferecer um nível maior de automatização do processo de sintonia, configurando adequadamente os parâmetros do SGBD. A validação destas regras bem como a implementação deste protótipo está em curso e espera-se ter um protótipo executando no fim do segundo semestre de 1997. O protótipo auxiliará na identificação das regras que melhorem o desempenho de contextos específicos de aplicações, além de servir como base de consulta para problemas de performance já identificados.

Como trabalhos futuros está previsto ampliar o escopo das regras, bem como, avaliar a validade das mesmas para outros SGBDs além do ORACLE. Além disso, pretende-se acrescentar características de aprendizado à ferramenta de forma que o conhecimento e a experiência do DBA possam ser refletidas sobre a base de conhecimentos da ferramenta, à medida que novos conjuntos de transações são analisados.

7 Referências

[AGR 87] AGRAWAL, Rakesh, CAREY, Michael and LIVNY, Miron. Concurrency Control Performance Modeling: Alternatives and Implications. ACM Transactions on Database Systems, vol. 12, Ni. 4, Dec 1987, pp 609-654.

[ATR 80] ATRE, S. Database structured techniques for design, performance and management. USA, John Wiley & Sons, Inc. 1980.

[BER 87] BERNSTEIN, Philip A. et al. **Concurrency control and recovery in database systems**. Addison-Wesley Publishing Company, USA, 1987.

[BUC 84] BUCHANAN, B. E SHORTLIFFE, E. Rule-based expert systems: the MYCIN experiments. Reading, Addison Wesley, 1984.

[CES 92] CESARINI, Francesca, MISSIKOFF, Michele, SODA, Giovani. An expert system approach for database application tuning. Data Knowledge Engineering. Vol 8. No. 1, Apr 1992, 35-55.

[COR 93] CORRIGAN, P. e GURRY, M. **Oracle performance tuning**. O'Really & Associates Inc., USA, 1993.

[COR 95] COREY, Michael J., ABBEY, MICHAEL e DECHICHIO, Dan J. Jr. **Tuning ORACLE**. Osborne McGraw-Hill, USA, 1995.

[EFF 84] EFFELSBERG, Wolfgang. Principles of database buffer management. ACM Transactions on Database Systems, vol 9, No 4, Dec 1984, pages 560-595.

[ELM 89] ELMARSI, Ramez e Navathe, Shankant B. Fundamentals of database systems. The Benjamim Cummings Publishing Company Inc, California, 1989.

[GRA 78] GRAY, J. **Notes on database operating systems**. Lecture Notes on Computer Science, vol 60, Eds. Springer-Verlag, New York, 1978.

[HAE 83] HAERDER, Theo e REUTER, Andreas. **Principles of transaction-oriented database recovery**. Computing Surveys, vol. 15, no. 4, December 1983, pp. 287-317.

[HAR 85] HARMON, Paul; KING, David. Expert Systems. John Wiley & Sons, Inc., USA, 1985.

[HAY 85] HAYES-ROTH, Frederick. Rule-based systems. Communications of the ACM, vol. 28, num. 9, September 1985, p. 921-932.

[ORA 92a] **Oracle 7 Server Administrator´s Guide.** Oracle Corporation, USA, 1992.

[ORA 92b] **Oracle 7 Server Application Developer´s Guide.** Oracle Corporation, USA, 1992.

[ORA 92c] **Oracle 7 Server Basic Concepts.** Oracle Corporation, USA, 1992.

[REU 84] REUTER, Andreas. **Performance analysis of recovery techniques**. ACM Transactions on Database Systems, vol. 9, no. 4, December 1984, pp. 526-559.

[SHA 92] SHASHA, D.E. **Database tuning: a principled approach.** Prentice-Hall, New Jersey, 1992.

[VIC 95] VICARI, Simone Rosa. Database tuning: aspectos de implementação de banco de dados que influenciam a performance de aplicações. Trabalho Individual I. CPGCC/UFRGS, 1995.