Integração de um Banco de Dados e um *Data Warehouse* sobre um Sistema de Arquivos Paralelos

José Craveiro da Costa Neto[1], Hélio Crestana Guardia[2], Liria Matsumoto Sato[3],
[1] Departamento de Computação e Estatística, Universidade Federal de Mato Grosso do Sul
Campo Grande–MS, Brasil
craveiro@dct.ufms.br
[2] Departamento de Computação, Universidade Federal de São Carlos
São Carlos-SP, Brasil
helio@dc.ufscar.br
[3] Departamento de Engenharia de Computação e Sistemas Digitais, Escola Politécnica da USP
São Paulo-SP, Brasil
liria@pcs.usp.br

Abstract—

Nowadays, there is an increasing demand for data warehousing systems, which are computational systems to support decision making based on information stored in a data warehouse. In order to provide such systems with high performance, a concurrent database system, called CDBS, has been proposed, CDBS integrates parallel database and warehousing technologies, and contains a data definition and manipulation function library that makes it possible to create, modify, and search on parallel databases and warehouses. In order to guarantee high performance, CDBS uses a parallel file system, called NPFS, which organizes data in such a way that its data manipulation functions can be executed rapidly. In this paper, we describe CDBS and present a case study in which the use of parallelism in databases and warehousing systems is required in order to achieve high performance. Besides, we discuss future research opportunities related to the specification, design and implementation of new CDBS library functions, as well as the main contributions of this work.

Keywords— **bancos de dados paralelos, *data warehousing*, sistemas de arquivos paralelos**

I. Introdução

O processamento paralelo permite um aumento do desempenho das aplicações, uma vez que as operações a serem realizadas podem ser distribuídas entre diversos processadores, que as executam simultaneamente. Entretanto, as plataformas para processamento paralelo ainda são pouco difundidas e, em geral, caras. Este fato tem feito com que recentes esforços de pesquisa se concentrem na obtenção de plataformas de hardware e sistemas de software mais acessíveis aos usuários [Yu98, Gar00].

Particularmente, uso do paralelismo é fundamental para obter bom desempenho no acesso a bancos de dados extensos. Contribuições importantes têm sido feitas na área de bancos de dados paralelos tanto no contexto de projeto do sistema como em processamento de consulta [Kim95, Yu98].

No uso de um sistema de banco de dados, observam-se picos de processamento em determinados horários e momentos de ociosidade em outros. A crescente necessidade pelo uso da informação, causando *overhead* nos momentos de pico. Além disso, os profissionais dos altos escalões das organizações estão exigindo mais informações que os apóiem na tomada de decisões. Desta forma, para que tais respostas sejam dadas com rapidez, é necessária a criação de informação redundante, que é armazenada nos *data warehouses* [Kim95, Kim96, Gar99].

Neste trabalho, registra-se o esforço de pesquisa desenvolvido para integrar o processamento paralelo e o uso de sistemas de arquivos paralelos em sistemas de bancos de dados e *data warehousing* paralelos para oferecer às organizações soluções nesta área, que se convertam em benefício para o cotidiano das pessoas. Um dos objetivos é, apresentar a proposta do sistema CDBS (*Concurrent Database System*), que é constituído por uma biblioteca de funções para criação, consulta e manutenção de um banco de dados paralelo e um *data warehouse*, reunidos num ambiente de dados paralelo e projetados para trabalhar sobre um sistema de arquivos paralelos, que está distribuído ao longo de uma rede de estações de trabalho.

Esta primeira seção apresentou o problema, a motivação e os objetivos pretendidos no presente trabalho. A seção 2 apresentará uma revisão de conceitos de bancos de dados, *data warehousing* e tipos de processamento sobre bancos de dados. A terceira seção desenvolverá conceitos de bancos de dados distribuídos e paralelos, procurando enfatizar a necessidade de desempenho sobre determinadas aplicações de bancos de dados. A seção 4 descreve sistemas de arquivos para aplicações que exigem alto desempenho, enfatizando o sistema NPFS [Gua99]. A quinta seção tratará do detalhamento da especificação e projeto da

biblioteca de funções que comporá o CDBS e sua implementação. A seção 6 ilustra o trabalho desenvolvido na seção anterior descrevendo a implementação do estudo de caso utilizado ao longo do texto. Na sétima seção são apresentadas as conclusões e as contribuições oferecidas pelo trabalho e uma relação de trabalhos futuros que deverão ser conduzidos.

II. BANCOS DE DADOS E *DATA WAREHOUSING*

Os bancos de dados têm sido muito úteis para permitir a execução de uma gama enorme de aplicações. Eles podem ser definidos como coleções de dados que precisam estar armazenados de forma duradoura, para a solução de um problema do mundo real, apresentando significado associado ao problema [Elm00].

Os bancos de dados procuram oferecer um volume cada vez maior de informações. Assim, um maior número de pessoas pode ter acesso a elas, fazendo crescer a funcionalidade requerida pelas aplicações. Algumas destas funções interessam-se por resumos retirados de um bancos de dados. A produção destes resumos ocorre de maneira compatível com o *data warehousing*, que se ocupa com a produção de tabelas de sumário ou visões materializadas e com o processamento de consultas de análise, que fazem uso destas tabelas ou de outras tabelas do banco de dados, executando o processamento analítico, que é um tipo de processamento que busca oferecer informações em forma de resumos, para análise por parte das gerências e diretorias das organizações.

Têm sido desenvolvidos sistemas de apoio à decisão, que produzem informações derivadas dos bancos de dados de produção, executando um tipo especial de processamento denominado processamento analítico, que se ocupa da geração de informações para os executivos das gerências e diretorias das empresas, procurando oferecer-lhes resumos, tendências e outros dados para que possam tomar decisões mais acertadas.

O banco de dados usado extensivamente no dia-a-dia é também chamado banco de dados de produção porque serve para o registro dos dados gerados pela execução das rotinas da organização, normalmente executadas pelo pessoal do nível operacional. A partir do banco de dados de produção são gerados novos conjuntos de dados, dando origem a um banco de dados específico, que pode ser chamado banco de dados para análise ou *data warehouse*.

Em geral, as aplicações de *data warehousing* diferem das aplicações tradicionais de bancos de dados. Enquanto os bancos de dados tradicionais são ajustados para transações de atualização rápida, um *data warehouse* é ajustado para consultas de longa duração. Ainda mais, bancos de dados tradicionais contêm, usualmente, apenas o estado atual do banco de dados; *data warehouses* mantêm estados anteriores, acrescentam informações temporais, como carimbos de tempo (*timestamps*), e contêm sumários pré-calculados dos dados [Gar99].

A arquitetura usada para criar e consultar um *data warehouse* está mostrada na Figura 1. As fontes contêm os dados brutos a serem integrados; podem ser bancos de dados relacionais, orientados a objetos, hierárquicos ou de rede ou bancos de dados de legado ou arquivos de texto. Para cada fonte, um extrator de dados específico recupera os dados desejados e o converte num formato adequado para o *data warehousing*. Assim, o comportamento do extrator de dados tem uma dependência forte da fonte da qual faz a extração dos dados.

Num *data warehouse*, as visões são computadas e armazenadas no banco de dados para permitir consulta e análise eficiente dos dados. Estas visões armazenadas no *data warehouse* são conhecidas como visões materializadas ou tabelas de sumário. A materialização de visões requer suporte de vários modos. A visão pode ser recomposta a partir do zero ou mantida incrementalmente por propagação de mudanças de dados do banco de dados para a visão [Gup95, Rao99, Gar00].

O *refresh*, usado para se promover o povoamento das tabelas de sumário, faz disparar a rotina de povoamento da visão materializada. É completo (*refresh complete*) quando atua sobre a tabela inteira, para cada tabela participante da consulta.

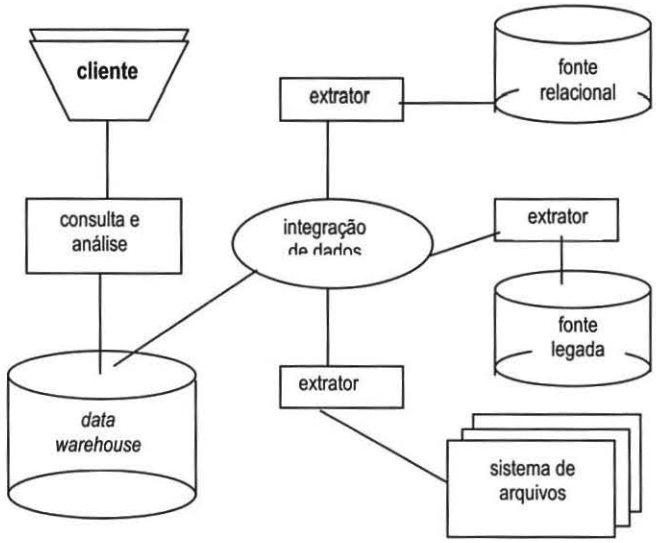

Fig 1. Arquitetura de um *Data Warehouse*

O *refresh* é incremental (*refresh incremental*) quando atua somente sobre os registros da tabela que não foram visitados até o *refresh* anterior. Assim, será necessário definir para tabelas de sumário o tipo de *refresh* e o período gasto entre o *refresh* anterior e o atual. No caso do *refresh refresh* será necessário definir uma estrutura que determine

os registros dos arquivos de produção usados na consulta que já sofreram *refresh* [Kim95, Cos01].

O exame completo de uma tabela, mesmo periódico, é custoso; uma alternativa é a avaliação incremental: as estimativas são mantidas e atualizadas toda vez que o banco de dados é modificado. Para tanto, é necessário modificar a implementação dos comandos SQL INSERT, DELETE e UPDATE para registrar as modificações ocorridas [Gar00].

III. Bancos de Dados Distribuídos e Paralelos

Enquanto há várias direções nas quais sistemas de bancos de dados modernos estão evoluindo, novas aplicações exigem grandes volumes de processamento e pesquisas complexas no banco de dados, devendo oferecer tempo de resposta satisfatório. Para várias famílias de aplicações, é importante que a informação esteja facilmente acessível, favorecendo a sua disponibilidade. Para permitir que esta situação se tornasse realidade, foram criados os bancos de dados distribuídos.

Em *data warehousing,* verifica-se o armazenamento redundante de informações, havendo necessidade de manutenção do controle desta redundância. A redundância produz um aumento do volume dos dados armazenados. Um grande número de acessos requer que as funções do sistema apresentem alto desempenho. São oferecidos os bancos de dados paralelos e, pelas características que apresentam, os bancos de dados paralelos mostram-se adequados para as atividades *data warehousing* [Gar00].

No decorrer desta evolução, sentiu-se a necessidade de bancos de dados que, mesmo com o armazenamento de grandes volumes de informação permitissem um processamento eficiente sobre eles, sem a necessidade de aquisição de poderosas arquiteturas de hardware. Desta forma, disseminou-se no mercado a tecnologia dos bancos de dados paralelos.

A. Bancos de Dados Distribuídos

Um sistema de banco de dados distribuído consiste em uma coleção de bancos de dados logicamente inter-relacionados distribuídos em diversos nós conectados por uma rede de computadores, tal que cada nó apresenta capacidade de processamento autônomo e participa na execução de consultas que requerem acesso a dados através de múltiplos nós [Kim95, Yu98, Gar00].

Quando um SGBD recebe uma consulta, ele constrói um plano de processamento da consulta e executa o plano para produzir a resposta à consulta. Um plano de consulta deve procurar maximizar o paralelismo e minimizar a transferência de dados pela rede [Kim95, Gar00].

B. Bancos de Dados Paralelos

Um sistema de banco de dados paralelo implica na distribuição dos dados do banco de dados pelos vários nós do sistema de uma maneira que permita transparência plena. Num sistema de banco de dados paralelo existem três grandes tipos de recursos: processadores, memórias e dispositivos de armazenamento. A forma como esses recursos são combinados define as arquiteturas de bancos de dados de memória compartilhada, de disco compartilhado ou de memória distribuída [DeW92, Val93]. Um estudo interessante das três arquiteturas baseado em experimentos é relatado em [Bhi88].

Dentre as arquiteturas com vários processadores, as arquiteturas distribuídas têm recebido uma maior atenção. Neste tipo de arquitetura, cada processador tem acesso direto só à sua memória e às suas unidades de disco. Sobre essa arquitetura, o *data warehouse* em si pode ser um banco de dados paralelo. O armazenamento dos dados no *data warehouse* pode ser um processo demorado e o ideal é que o paralelismo de software e hardware seja utilizado. Cuidados especiais devem ser tomados com registros que não foram carregados por problemas de integridade referencial. A preocupação com o desempenho deve ser uma constante, procurando explorar ao máximo o paralelismo na consulta ao banco de dados e o desenvolvimento de técnicas de distribuição dos dados do banco de dados paralelo [Gar99].

Em aplicações de bancos de dados, o paralelismo pode ser obtido em diferentes níveis. No paralelismo interconsulta, consultas concorrentes podem ser executadas em paralelo por diferentes processadores. No paralelismo interoperação, operações diferentes de uma mesma consulta podem ser executadas em paralelo por processadores diferentes, que é um tipo de paralelismo intraconsulta. No paralelismo de instrução, a mesma operação pode ser executada em paralelo por vários processadores. Os dois últimos tipos de paralelismo são mais interessantes, por oferecerem maior desempenho. Numa outra dimensão, o processamento de bancos de dados pode usar três formas de paralelismo: o paralelismo independente, o paralelismo em *pipeline* e o paralelismo particionado.

Um meio eficaz para se usar vários nós de processamento, cada um com unidade de disco, é particionar as tabelas que fazem parte da operação em vários fragmentos e distribuí-los por diferentes nós. Uma tabela R deve ser particionada em N fragmentos R_1, ..., R_N tal que todos os fragmentos tenham aproximadamente o mesmo tamanho e possam suportar satisfatoriamente as operações esperadas sobre a tabela. São utilizadas a distribuição circular (*round robin*), a distribuição por faixa de valores e a distribuição *hash* [Yu98].

Seleções e projeções na mesma tabela são, usualmente, executadas simultaneamente. Daí, elas podem ser consideradas como uma operação de redução simples que retorna um conjunto de subtuplas da tabela que satisfazem uma condição especificada. A redução de um fragmento num processador é feita da mesma forma como num

sistema regular de banco de dados, seqüencialmente ou usando um caminho de acesso. Assim, reduções em todos os fragmentos podem ser executadas em paralelo, sem problemas [Gra93].

Como a junção é uma operação muito usada em bancos de dados, apesar do seu alto custo, é necessário o desenvolvimento de técnicas para sua execução de forma a não comprometer o desempenho do banco de dados[Gra93].

IV. Um Sistema de Arquivos Paralelos para um Banco de Dados Paralelo

No sentido de construir um banco de dados paralelo para permitir a garantia de alto desempenho, foi pesquisada a adoção de um sistema de arquivos que atendesse a este requisito. Existem na literatura vários sistemas de arquivos que se propõem a resolver o problema. Dentre os sistemas estudados, optou-se pelo sistema NPFS (*Network Parallel File System*), que é um sistema de arquivos paralelos comprometido com o alto desempenho, o qual está descrito em [Gua99].

Procurou-se explorar o paralelismo através do particionamento dos dados, que são atribuídos aos processos ou processadores disponíveis, envolvendo dois aspectos principais:

a decomposição de um programa em partes, que devem ser distribuídas entre os processos de uma aplicação paralela, e a decomposição física dos dados, que são distribuídos entre múltiplas unidades de armazenamento [Kim95, Old98].

O NPFS é um sistema de arquivos paralelos destinado a um ambiente computacional distribuído. A arquitetura alvo considerada para sua utilização consiste em um conjunto de estações de trabalho interligadas em rede, cada uma com seu próprio disco local, que pode ser compartilhado. A estratégia adotada no sistema NPFS consiste em utilizar o modelo cliente/servidor para possibilitar o acesso das aplicações aos arquivos paralelos distribuídos entre os discos das estações de trabalho da rede. Nesse sentido, servidores fornecem suporte para o acesso aos dados armazenados em seus sistemas de arquivos locais. Implementado no nível de programa de usuário, o sistema NPFS é composto de três tipos de processos: Mestre, Servidor e Cliente [Gua99].

Com um particionamento apropriado dos dados, é possível que os processos manipulem segmentos distintos, com redução das operações de posicionamento (*seek*) no acesso seqüencial, e que explorem a localidade no acesso aos arquivos, reduzindo a transmissão de dados através da rede de comunicação.

Foi criado um conjunto de primitivas com o intuito de tirar proveito da estrutura paralela dos arquivos por parte das aplicações paralelas. As primitivas construídas tratam da manipulação de servidores e de arquivos paralelos e da integração do paralelismo de dados e processos.

Para a implementação do CDBS, novas primitivas para leitura e escrita foram introduzidas no NPFS. As primitivas r_read() e r_write() são usadas para fazer, respectivamente, leitura e escrita replicada, quando solicitado pelo CDBS. Já as primitivas x_read() e x_write() foram criadas para atender a solicitação do CDBS para, respectivamente, ler e gravar dados num segmento específico, nos casos em que se está trabalhando com distribuição por faixa de valores ou por função de *hashing* [Cos01].

V. CDBS: Um Ambiente de Dados Paralelo para *Data Warehousing*

O CDBS oferece primitivas para a manutenção automática de visões materializadas do *data warehouse*, sendo capaz de produzir informações mesmo que não ocorram mudanças nas tabelas fontes. Para favorecer o desempenho e a consistência, as tabelas do banco de dados de produção e as visões materializadas do *data warehouse* estão integradas num sistema único, aqui denominado de ambiente de dados.

A. Criação de um Ambiente de Dados Paralelo

A criação de ambientes para *data warehousing* requer a ligação a bancos de dados que permitam o tráfego de dados entre o banco de dados de produção e o *data warehouse* de uma forma integrada. Este tráfego é realizado com sucesso quando se faz um projeto cuidadoso do ambiente. No presente trabalho busca-se um ambiente que se preocupe com o desempenho das operações de *data warehousing* e, para tanto, esta preocupação manifesta-se já na etapa de projeto do ambiente.

A Figura 2 apresenta a arquitetura do CDBS, representando este importante objetivo de integração. O banco de dados será formado por um conjunto de arquivos tradicionais aptos para apoiar o processamento de transações. Por outro lado, o ambiente de dados deve apoiar também o processamento analítico

Na criação do ambiente de dados define-se a área de disco necessária para a descrição dos objetos e para armazenamento dos dados do ambiente de dados. Ao criar a estrutura do ambiente de dados, armazena-se informações sobre as tabelas do banco de dados e do *data warehouse*. Ao criar um ambiente de dados distribuído entre os vários nós de um cluster, cada máquina do cluster possuirá um fragmento deste ambiente.

A primitiva d_create() é usada para criar a estrutura do ambiente de dados que ficará armazenada no catálogo e criar os objetos do ambiente de dados. Isto é feito conforme especificação contida no catálogo, a qual faz a chamada de outras primitivas que serão responsáveis por atividades específicas durante a criação do ambiente. Deste modo, a

primitiva tem uma grande importância na definição do

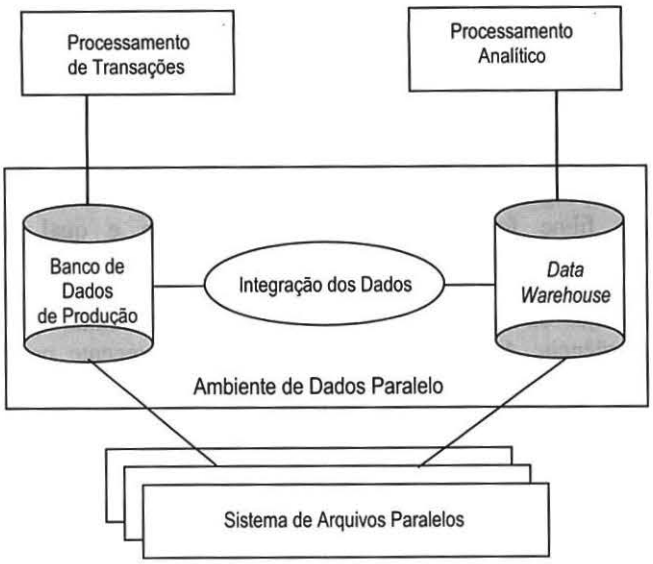

Fig. 2. Arquitetura do Sistema CDBS

ambiente de dados.

A primitiva t_create() foi implantada para criar a estrutura de uma tabela do ambiente de dados, que fica armazenada no catálogo. Para permitir o processamento analítico, são criadas as visões materializadas. A primitiva v_create() foi implantada para criar a estrutura de uma visão materializada.

B. Manipulação de um Ambiente de Dados Paralelo

As tabelas do banco de dados usadas para o processamento de transações são povoadas usando a funcionalidade de um SGBD. Neste ponto, é importante que o projetista crie funções e tabelas que garantam a administração da consistência entre as tabelas do banco de dados e as tabelas de sumário. Como as tabelas de sumário não serão alimentadas via funcionalidade do SGBD, então, serão construídas novas funções para responsabilizarem-se pelo povoamento automático destas, funções estas que serão executadas pelo sistema nos instantes definidos pela política de atualização, tendo atenção à complexidade manifestada pelo paralelismo dos dados.

No CDBS, para executar a inserção de registros numa tabela está disponível a primitiva t_insert(), que obtém os dados e os armazena no arquivo paralelo. Quando se tratar do povoamento de uma tabela do banco de dados de produção, a obtenção dos dados é feita a partir de um dispositivo de entrada. Já, no povoamento de uma tabela de sumário, a obtenção dos dados é feita a partir de uma consulta a tabelas do banco de dados ou tabelas de sumário, de acordo com a necessidade.

Para a remoção de registros existe a primitiva t_delete(), que localiza os registros a serem removidos e coloca neles uma marca de remoção, por questões de desempenho.

Uma vez criado e povoado o ambiente de dados, deve-se programar funções para garantir a execução de consultas sobre o mesmo, tanto as consultas tradicionais como as consultas analíticas.

Determina-se o resultado da consulta, concentrando-o num único nó ou segmentando-o por vários nós da arquitetura paralela. Por fim, faz-se a reunião global, usando vários nós ou um só nó, gerando o resultado final. Para executar esta seqüência de atividades, foi criada a primitiva t_select(). Na execução do comando t_select(), principalmente quando se tratar do povoamento das tabelas de sumário, verifica-se a presença de campos calculados, onde são executadas contagens, somatórias e outras funções agregadas.

O povoamento automático das tabelas de sumário é feito derivando-se novos dados a partir das tabelas do banco de dados de produção, com a execução das primitivas de manipulação do ambiente. Uma outra preocupação é definir como deve ser feito o povoamento e o período que se deve esperar para introduzir novas informações, correspondendo à cláusula *refresh*. Percebe-se que a atividade de atualização do *data warehouse* vai exigir um processamento extra em determinados períodos, momentos estes em que o processamento de transações deve ser diminuído, ou mesmo, cessado

C. Aspectos de Implementação do CDBS

A execução da primitiva d_create() cria o ambiente de dados paralelo em todos os nós do cluster usados para o seu armazenamento, criando, inclusive, a estrutura das tabelas deste ambiente. É feita a criação de um diretório temporário contendo os arquivos de catálogo especificados. Definem-se os nós do cluster, com a execução de cat_sites(), e o diretório <amb_name> em cada um dos nós onde vai ser armazenado o ambiente de dados, retirando as informações do diretório temporário. Os arquivos de catálogo devem ser replicados em todos os nós definidos durante a execução de cat_sites(). Para cada um dos registros dos arquivos cat_tables e cat_views, gerar um arquivo paralelo ou não paralelo correspondendo a uma tabela ou visão materializada do ambiente de dados. A estrutura de cada um desses arquivos é construída buscando-se as características dos campos nos arquivo cat_t_fields e cat_v_fields, respectivamente.

A seleção de arquivos e registros cujos campos satisfazem condições definidas pelo operador constitui uma operação freqüente e com grande demanda computacional em bancos de dados e *data warehouses*, sendo beneficiados com a sua integração por meio de um sistema de arquivos paralelos.

Nesse sentido, a existência de um sistema de arquivos paralelo e distribuído, em que diversos nós de um cluster são usados para o armazenamento dos dados fornece uma excelente oportunidade para a otimização do desempenho de um banco de dados. O paralelismo dos dados e dos processos pode, por exemplo, ser usado para exploração da localidade dos acessos. São determinados os segmentos dos arquivos que serão lidos e localizados os campos desses arquivos que serão recuperados. Enquanto se faz a leitura, são separados os registros que satisfazem as condições de seleção para, logo em seguida, fazer a junção desses registros segundo as condições de junção presentes na consulta.

Como se trabalha com a preservação da ordem histórica dos registros dos arquivos, define-se um vetor v1, para armazenar a execução parcial da consulta correspondente ao grupo de registros relativo ao período que se está processando quando se faz o processamento relativo às cláusulas SELECT, FROM e WHERE do comando SELECT de SQL. Havendo agrupamento ou ordenação, v1 é esvaziado para um segundo vetor, v2, que será percorrido pela rotina de agrupamento ou ordenação ou transferido para o arquivo de resultado. Ao mesmo tempo em que se faz agrupamento e ordenação do grupo de registros que foi processado para v2, executa-se a parte inicial da consulta relativa ao próximo grupo de registros.

Ao executar agrupamento ou ordenação, correspondendo às cláusulas GROUP BY, HAVING e ORDER BY, faz-se o processamento do vetor v2, gerando um vetor v3, que estará ordenado pelos campos de agrupamento ou de ordenação. Ao esvaziar v2, este é liberado para a recepção do próximo grupo de registros gerado em v1, e v3 é esvaziado para o resultado da consulta, sendo disponibilizado para agrupamento e ordenação do próximo grupo de registros que forem colocados em v2. Grande parte das consultas de análise usa, no agrupamento, a dimensão tempo, proporcionando paralelismo independente em todos os nós [Cos01].

VI. Um Estudo de Caso com o CDBS

Para ilustrar a sua viabilidade, faz-se, neste seção, a apresentação de um estudo de caso com comentários justificando a importância do sistema projetado. O problema escolhido relaciona-se ao sistema de uma rede de locadoras de filmes para vídeo-cassete, distribuída por vários bairros da cidade de São Paulo, de onde são escolhidas algumas tabelas do banco de dados de produção e algumas visões materializadas. O domínio a ser explorado representa uma situação bastante real no âmbito do comércio de produtos e serviços. Várias organizações trabalham com múltiplas filiais, distribuídas geograficamente, que precisam dispor informações às suas coordenações e direções centrais.

A. O Ambiente de Dados da Rede de Locadoras

As lojas da rede alugam cópias de filmes para seus clientes e recebem uma remuneração por este serviço. Além da rotina cotidiana das lojas da rede, a diretoria e as gerências das lojas necessitam retirar do sistema uma série de informações que possam servir de auxílio na tomada de decisões.

Como ilustração, dentre as várias solicitações que são feitas, algumas compreendem: 1) o número de vezes que um filme foi alugado num dado período e qual o faturamento da locação do filme no período; 2) clientes que dão maior retorno a uma loja; 3) clientes que freqüentavam as lojas da locadora e, atualmente, vêm diminuindo esta freqüência; 4) renda por fita num período recente para apoiar a política de aquisição de fitas, inclusive da forma mais localizada possível; 5) lojas que estão apresentando melhor desempenho de locações, com pesquisa sobre o motivo deste desempenho, e 6) o período de melhor desempenho por loja, ao longo de todo o período.

É importante reforçar que a quantidade de locações feitas pelas lojas da rede é muito grande, considerado o tempo de existência da rede, por volta de dez anos. O banco de dados possui as onde se verifica a presença das seguintes tabelas:

Loja (idloja, nomeloja, classe, rua, num_rua, bairro, CEP, zona, fone), Cliente (idcli, nomecli, idloja, dtnasc, responsavel, rua, num_rua, bairro, CEP, fone), Filme (idfilme, nomefilme, genero, censura, produtora, numfitasloja, numfitaslocdia, valorlocdia) e Locacao (idloc, idcliloc, idlojaloc, idfilmeloc, dialoc, mesloc, anoloc, valorlocfita).

As projeções a seguir revelam uma amostra reduzida do volume de dados tendo em vista a capacidade de máquinas e de processamento do cluster existente no LASB/PCS (Laboratório de Arquitetura e Software Básico do Departamento de Engenharia da Computação e Sistemas Digitais). Considera-se a experiência realizada num cluster com 8 ou 16 máquinas. Trabalha-se com uma rede de locadoras com apenas 16 lojas, com as médias de 2 lojas por bairro e de 2 bairros por zona.

B. Execução do Estudo de Caso

São consideradas as seguintes tabelas do banco de dados de produção: Loja, Cliente, Filme e Locacao. As três primeiras disponibilizam informações à tabela Locacao, que ocupa posição privilegiada no sentido de oferecer soluções para suporte à tomada de decisões, uma vez que, a partir dela, podem ser construídas várias tabelas de sumário. As dependências entre estas tabelas estão representadas na Figura 3.

A tabela Loja está replicada pelos vários nós do cluster; Cliente tem os registros distribuídos pelos nós, e Filme tem alguns campos replicados pelos nós e outros campos armazenando apenas os valores locais, correspondentes ao nó do cluster a que se referem. Deste modo, permite-se

examinar várias situações que não seriam possíveis quando do estudo do problema considerando uma única tabela.

A tabela Locacao será segmentada pelos vários nós do cluster de acordo com as locações que são feitas pela loja que tem sua movimentação controlada pelo nó. Trata-se de uma tabela bastante grande quando se considera o registro de todas as locações da rede de lojas ao longo de sua existência, chegando a um volume de dados da ordem de 2 *terabytes*.

São criadas as tabelas do banco de dados de produção a partir das informações introduzidas sobre as tabelas e seus campos, alimentando as estruturas das tabelas. Neste ponto, são completadas as definições das tabelas, atendendo os requisitos de segmentação e replicação, com sua implementação nos nós que foram definidos no projeto. Além disto, a ordem de implantação das tabelas no ambiente deve ser feita de forma a garantir a dependência existente entre elas.

Depois de criadas as tabelas ou, à medida que se criam as tabelas, faz-se o seu povoamento, segundo a rotina de povoamento das tabelas que foi escrita, usando a geração randômica dos dados para vários campos destas tabelas, para simular a rotina das lojas, e geração seqüencial das chaves. Este povoamento de registros nas tabelas obedece à seqüência estabelecida pelas dependências existentes entre elas.

Na implantação do sistema de *data warehousing*, o trabalho inicial consistiu na criação das visões materializadas, segundo a hierarquia de dependência verificada entre as visões materializadas e as tabelas do banco de dados de produção e aquela verificada entre uma visão materializada e outra. Como foi feito com a criação de tabelas, na criação das visões materializadas, observou-se a mesma seqüência de passos. As dependências entre as tabelas do banco de dados de produção e estas tabelas de sumário estão representadas na Figura 3.

À medida que se criam as tabelas de sumário do *data warehouse*, providencia-se o seu povoamento inicial, com a

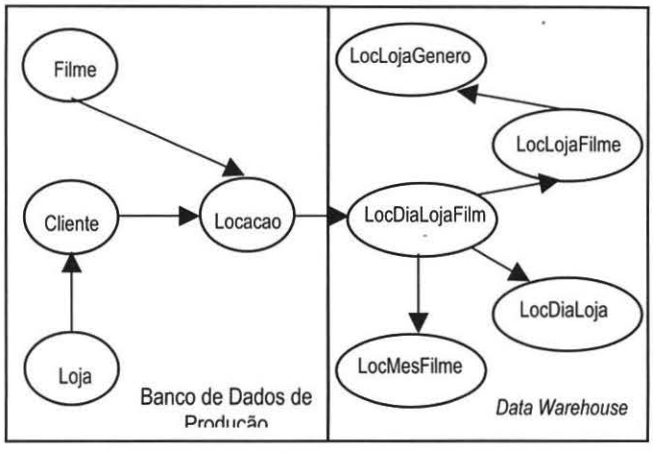

Fig 3. Dependências de Dados nos Arquivos do Ambiente

cópia dos valores dos campos de agrupamento e geração dos valores calculados.

Seja a Consulta 1: Obter as locações anuais por bairro (quantidade de fitas locadas e valor total em locações) para filmes do gênero drama, efetuadas a partir de 1995 relativas às regiões da Zona Sul da cidade. Em SQL, a sua expressão será manifestada assim:

```
SELECT bairro, anoloc, COUNT(*), SUM (valorlocfita)
FROM Filme, Locacao, Loja
WHERE idfilme = idfilmeloc AND idlojaloc= idloja
AND genero = 'drama' AND anoloc >= 1995 AND zona = 'Sul'
GROUP BY bairro, anoloc;
```

A execução desta consulta compreenderá na sua decomposição em uma série de operações intermediárias, tendo em vista a obtenção de melhor desempenho. Neste caso, a consulta está tendo acesso à tabela Locacao, que está sujeita continuamente ao processamento de transações de atualização. O plano para execução da consulta no ambiente contempla a execução de 4 processos, conforme se pode ver em [Cos01].

No caso da execução da Consulta 1 usando a tabela de sumário LocDiaLojaFilme no lugar da tabela Locacao, o plano de execução da consulta é, praticamente, o mesmo. Verifica-se, porém, que o tamanho da tabela de sumário é menor que o tamanho da tabela Locacao, está organizada pelos campos de agrupamento e o processamento da consulta não influencia de forma negativa o processamento OLTP.

VII. Conclusões e Trabalhos Futuros

O CDBS foi criado para permitir a integração de um banco de dados de produção com as tabelas de sumário de um *data warehouse*, servindo-se do paralelismo, compondo um ambiente denominado ambiente de dados paralelo. Foram desenvolvidas primitivas para garantir maior velocidade nas consultas ao banco de dados de produção e ao *data warehouse*, oferecendo maior desempenho com uso efetivo do paralelismo e de uma implementação eficiente de suas funções.

Na construção deste ambiente foi importante o estudo e a utilização de sistemas de arquivos paralelos para armazenamento dos dados do ambiente. A escolha do sistema NPFS [Gua99] foi muito apropriada pela funcionalidade oferecida pelo mesmo e pela disponibilidade de informações sobre o mesmo.

O conhecimento obtido a partir da pesquisa desenvolvida sobre sistemas de arquivos paralelos ofereceu a possibilidade de definir estratégias de como as organizações podem usufruir desta tecnologia para sua aplicação na construção de bancos de dados paralelos e na aplicação do paralelismo em *data warehousing*.

Como conseqüência natural deste trabalho, foi proposta a arquitetura e a especificação da funcionalidade das

primitivas do sistema de software CDBS e a correspondente implementação de suas funções, para permitir a realização das operações de criação, consulta e a manutenção de um banco dados que requeira alto desempenho e do correspondente *data warehouse* relacionado com o banco de dados.

Além desta forma de integração, foi pesquisada, também, a integração entre banco de dados de produção e *data warehouse*, permitindo mantê-los num ambiente único, tal que se possa executar uma gama variada de consultas e de transações em geral.

Como trabalhos futuros, prevê-se a automatização gradual de uma série de atividades a serem desenvolvidas manualmente pelo projetista, tais como, a política de fragmentação e replicação dos dados pelos nós da rede; a definição dos métodos de acesso ao ambiente de dados mais apropriados, para que garantam processamento satisfatório de consultas e transações e a construção de uma interface mais amigável para o CDBS.

Deve ser feito um estudo mais detalhado do comportamento das funções agregadas, com propostas de soluções para otimização de consultas analíticas e de transações para um ambiente de dados paralelo. Concluindo, o desenvolvimento de novos estudos nesta área trará aos pesquisadores oportunidades de oferecerem novos resultados à comunidade, uma vez que é uma área de estudos ampla e promissora que passou a ser explorada mais efetivamente nos últimos anos, com a criação de vários grupos de pesquisa no Brasil e no exterior para investigar o assunto.

REFERÊNCIAS

[Cos01] Costa Neto, J. C. Considerações sobre a Integração de um Banco de Dados e um *Data Warehouse* sobre um Sistema de Arquivos Paralelos. Tese de doutorado, Escola Politécnica da Universidade de São Paulo, 2001.

[DeW92] DeWitt, D.; Gray, J. Parallel Database Systems: the Future of High Performance Database Systems, CACM, 35(6), junho/1992, 85-98.

[Elm00] Elmasri, R.; Navathe, S. R. Fundamentals of Database Systems. Addison Wesley, 30 ed., 2000.

[Gar99] Garcia-Molina, H.; Labio, W. J.; Wiener, J. L.; Zhuge, Y. Distributed and Parallel Computing Issues in Data Warehousing, 1999.

[Gar00] Garcia-Molina, H.; Ullman, J. D.; Widom, J. Database System Implementation. Prentice Hall, 2000.

[Gra93] Graefe, G. Query Evaluation Techniques for Large Databases, ACM Computing Surveys, 25(2), junho/1993, 73-170.

[Gua99] Guardia, H. C. Considerações sobre as Estratégias de um Sistema de Arquivos Paralelos Integrado ao Processamento Distribuído. Tese de doutorado, Escola Politécnica da Universidade de São Paulo, 1999.

[Gup95] Gupta, A.; Harinarayan, V.; Quass, D. Aggregate-Query Processing in Data Ware housing Enviroments. in Proceedings of the 21st VLBD Conference, Zurique, Suíça, 1995.

[Kim95] Kim, W. (ed.) Modern Database Systems: The Object Model, Interoperability, and Beyond. Reading, MA, Addison Wesley/ACM Press, 1995.

[Kim96] Kimball, R. The Data Warehouse Toolkit : Practical Techniques for Building Dimensional Data Warehouses. John Wiley & Sons, 1996.

[Old98] Oldfield, R.; Kotz, D. Applications of Parallel I/O. Technical Report PCS-TR98-337 (suplemento ao PCS-TR96-297), Computer Science, Dartmouth College, agosto/1998.

[Qua97] Quass, D.; Widom, J. On-line Warehouse View Maintenance for Batches Upadates. In Proceedings of the ACM SIGMOD International Conference on Management of Data, Tucson, Arizona, maio/97.

[Rao99] Rao, J.; Ross, K. A. Cache Conscious Indexing for Decison-Suport in Main Memory, Columbia University, in Proceedings of the 25th VLDB Conference, Edimburgo, Escócia, 1999.

[Val93] Valduriez, P. Parallel Database Systems: Open Problems and New Issues, Distributed and Parallel Databases, 1(2), abril/1993, 137-165.

[Wid95] Widom, J. Research Problems in Data Warehousing. In Conference on Information and Knowledge Management, 1995. Ver: <http://db.stanford.edu/pub/widom/1995/> warehouse-research.ps.

[Yu98] Yu, C. T.; Meng, W. Principles of Database Query Processing for Advanced Applications.Morgan Kaufmann Publishers, Inc., 1998.