FACULDADE LOURENÇO FILHO
CIÊNCIA DA COMPUTAÇÃO

Marluce Nunes Albuquerque

UM ESTUDO SOBRE O FUNCIONAMENTO DO PROCESSAMENTO DE CONSULTAS EM BANCO DE DADOS ORIENTADO A OBJETOS, XML E RELACIONAL

FORTALEZA, 2010

Marluce Nunes Albuquerque

UM ESTUDO SOBRE O FUNCIONALMENTO DO PROCESSAMENTO DE CONSULTAS EM BANCO DE DADOS ORIENTADO A OBJETOS, XML E RELACIONAL

Monografia apresentada ao curso de Ciência da Computação da Faculdade Lourenço Filho como requisito parcial para obtenção do grau de Bacharel em Ciência da Computação.

Orientador: Prof. MSc. Fernando Siqueira

FORTALEZA, 2010

Marluce Nunes Albuquerque

UM ESTUDO SOBRE O FUNCIONALMENTO DO PROCESSAMENTO DE CONSULTAS EM BANCO DE DADOS ORIENTADO A OBJETOS, XML E RELACIONAL

Monografia apresentada ao curso de Bacharel em Ciência da Computação da Faculdade Lourenço Filho, como parte dos requisitos necessários à obtenção do grau de Bacharel em Ciência da Computação.

Monografia apresentada em: 16 /03/ 2010

Composição da Banca Examinadora:

_____

Prof. MSc. Fernando Soares de Siqueira (Orientador)

_____

Prof. MSc./Dr. Carlos Alberto Manso (Coordenador)

_____

Prof. MSc. Ériko Joaquim Rogério Moreira (Professor)

AGRADECIMENTOS

Agradeço ao meu orientador Fernando Siqueira, pelo apoio desde o começo do trabalho. Aos meus amigos e familiares que acreditaram em mim desde o princípio e a Deus principalmente por ter me dado forças para continuar a lutar, apesar das dificuldades enfrentadas.

Agradeço pelos meus colegas por terem estado comigo nos momentos bons e difíceis, oferecendo apoio e atenção.

Também agradeço àqueles que de alguma maneira me ajudaram no desenvolvimento deste trabalho.

RESUMO

Os Bancos de Dados (BDs) são projetados para gerir grandes volumes de informações, e o gerenciamento dessas informações implica também na definição de mecanismos de consultas adequados para recuperação de informações de um Banco de Dados (BD) de forma eficiente. A eficiência de uma consulta está relacionada com o tratamento da complexidade interna de suas estruturas de representação de dados ou como a consulta submetida é processada. Contudo, não é esperado que o usuário sempre escreva suas consultas de uma maneira eficiente, até porque é uma atividade complexa e envolvem inúmeras situações que precisam ser tratadas e é impraticável ficar sobre a responsabilidade do usuário. Para isso, o Sistema Gerenciador de Banco de Dados (SGBD) possui um componente que realiza o seu processamento que não trata somente da extração de informações de um banco de dados, mas também das atividades que resultem na construção de um plano de execução que minimize o custo e maximize o desempenho das consultas. Componente esse denominado Processador de Consultas. Considerando no cenário atual a existência de diferentes plataformas de SGBDs, o que resulta em processadores de consutas específicos para cada plataforma, este trabalho vem apresentar um estudo do funcionamento do processamento de consultas em banco de dados da plataforma Orientado a Objetos, XML e Relacional, através da descrição do funcionamento de cada passo das etapas do processamento de uma consulta de um SGBD, e destacar a importância da otimização de consulta nesses SGBDS disponíveis no mercado através da descricao dos mecanismos utilizados nessa etapa. Para tal, serão descritas as etapas realizadas no processamento de consultas dos SGBDs MySQL, Tamino e em um modelo geral de um SGBD Orientado a Objeto, propiciando um entendimento mais detalhado do funcionamento dos componentes de seus respectivos Processadores de Consulta.

Palavras-chave: Banco de Dados; Processamento de Consultas; Otimização.

ABSTRACT

Databases (DBs) are designed to manage large volumes of information, and the management of this information also includes the establishment of suitable query mechanisms for retrieving information from a database (DB) efficiently. The efficiency of a query is related to the treatment of complex internal structures of data representation or how the submitted query is processed. However, it is not expected that the user would always write his queries in an efficient manner, even because it is a complex activity and involve many other situations that need to be treated and it is impractical to be on the user's responsibility. For this, the Management System Database (DBMS) has a component that performs its processing in a way that will not only tract the information extraction from a database, but also the activities resulting from the building of an execution plan that minimizes the cost and maximize the performance of queries. This component is called the Query Processor. Considering the existence of different DBMS platforms in the current scenario, which results in processors platform-specific queries, this work presents a study of the query processing operation in Object Oriented, XML and Relational database platform by describing the operation of each step of the stages by processing a query from a DBMS, and highlight the importance of query optimization in these DBMSs available in the market through the description of the mechanisms used in this step. This will describe the steps used in query processing of MySQL DBMS, Tamino and in a Object Oriented DBMS general model, providing a more detailed understanding of the functioning components of their respective processors Consultation.

Keywords: Functioning of the Query Processing, Database, Optimization, DBMS

SUMÁRIO

LISTA DE QUADROS E FIGURAS

LISTA DE TABELAS

1. INTRODUÇÃO

Os Bancos de Dados (BDs) são projetados para gerir grandes volumes de informações, e o gerenciamento dessas informações implica a definição de estruturas de armazenamento e de mecanismos de consultas adequados para recuperação de informações de um Banco de Dados (BD) de forma eficiente.

Muitas vezes esta eficiência está relacionada com a complexidade interna das estruturas de representação dos dados ou com as consultas submetidas para processamento. Não é esperado que o usuário sempre escreva suas consultas de uma maneira eficiente, logo, é responsabilidade do Sistema Gerenciador de Banco de Dados (SGBD) construir um plano de execução que minimize o custo e maximize o desempenho das consultas. Contudo, a complexidade destas atividades deve ser transparente para o usuário final.

O processamento de consultas em um sistema de banco de dados é realizado quando o usuário submete uma consulta a um SGBD, interativamente ou através de uma aplicação, utilizando uma linguagem de consulta de alto nível (SILBERSCHATZ, 2006). Este componente não trata somente da extração de informações de um banco de dados, mas também atividades com o objetivo de otimizar o processamento das consultas submetidas ao SGBD.

Considerando a importância do processamento de consulta na recuperação das informações, este trabalho descreve a estrutura de processamento e os passos de otimização de consultas realizados por alguns SGBDs encontrados no mercado.

1.1. Motivação

Um Sistema Gerenciador de Banco de Dados (SGBD) é o conjunto de programas responsável pelo gerenciamento de uma base de dados que retira da aplicação cliente a responsabilidade de gerenciar o acesso, manipulação e organização dos dados. O SGBD disponibiliza uma interface para que os seus clientes possam manipular os dados de uma base de dados.

O principal objetivo de um SGBD é proporcionar um ambiente tanto conveniente quanto eficiente para a recuperação e armazenamento das informações do banco de dados

(SILBERSCHATZ, 2006). Duas das funcionalidades mais importantes estão relacionadas com o processamento e a otimização de consultas.

A literatura apresenta várias estratégias disponíveis para processar uma determinada consulta em um SGBD e a cada uma delas é associada um custo. Portanto, o custo de avaliação da consulta pode ser medido em termos de vários recursos diferentes, onde são incluídos os acessos ao disco e o tempo de CPU para executar uma consulta. Vários estudos já comprovaram que vale a pena para o SGBD gastar uma quantia significativa de tempo na seleção de uma boa estratégia para processar uma consulta, até mesmo se a consulta for executada somente uma vez (SILBERSCHATZ, 2006). Todos os SGBDs geram um plano de execução para um comando submetido. Esse plano informa qual a estratégia de acesso aos dados ou como realizar as tarefas ligadas ao BDs.

O desempenho dos SBDs e o processamento otimizado de consultas são assuntos discutidos em artigos, tais como, Paul DuBois (DUBOIS, 25ª ed), Glenn Goodrum (GOODRUM, 18ª ed) e outros. DUBOIS aborda a otimização de consultas para o SGBD MySQL. Já Glenn Goodrum fala que na prática o plano de consulta selecionado nem sempre é o mais rápido e pode, até mesmo, não ser próximo do mais rápido. Quando isso acontece, pode-se considerar o uso de "dicas" para adquirir o uso de um plano melhor.

Diante dos pontos descritos, esta monografia, além de apresentar a teoria existente na literatura sobre processamento e otimização de consultas, analisará as principais técnicas e "dicas" práticas relacionadas com o processamento e otimização de consultas, servindo como fonte para auxiliar os projetistas e administradores de banco de dados no entendimento e uso destes conceitos.

1.2. Objetivos

Este trabalho pretende apresentar como funciona cada passo das etapas do processamento de uma consulta de um SGBD, destacando a importância da otimização de consulta nos principais SGBDS disponíveis no mercado.

Outro ponto ressaltado é a atividade de otimização de consulta através da descrição do processo de montagem e seleção dos planos de avaliação de consulta. Uma vez que cada plano de avaliação pode resultar em custos diferentes, e o otimizador escolhe o plano que resultar na a estimativa de menor custo.

1.3. Estrutura da Monografia

Esta monografia está organizada em cinco capítulos, sendo que no capítulo 1 foram mostrados a motivação, os objetivos e a estrutura do trabalho.

No capítulo 2 são apresentados os conceitos do Sistema de Banco de dados, incluindo suas características, arquitetura, componentes e os modelos de dados.

No capítulo 3 são descritos os conceitos gerais dos SGBDs Relacional, Orientado a Objetos (OO) e XML, sendo que no SGBD Relacional discorre-se sobre a importância da estrutura dos Bancos de Dados Relacionais, as restrições de integridade, as linguagens de consulta e a álgebra Relacional; no Sistema Gerenciador de Banco de Dados Orientado a Objetos (SGBDOO) são destacados os conceitos de Objetos Complexos, Identidade de Objetos, Herança única ou múltipla, Persistência de Objetos, as características e as restrições de Integridade. Por fim, descreve-se o SGBD XML, como uma breve introdução da Linguagem XML, suas características, a modelagem conceitual e física e o XML Query.

No capítulo 4 são apresentados os principais passos envolvidos no processamento de consultas, destacando o funcionamento do processamento de consultas nos SGBDs Relacionais, XML e OO, juntamente com o processamento de consultas dos bancos de dados MySQL e Tamino. São destacados a definição, a visão geral da arquitetura e o detalhamento dos componentes relacionados ao processamento de consultas desses SGBDs.

No capitulo 5 conclui-se o estudo realizado neste trabalho e indicam-se também as conclusões e trabalhos futuros.

2. SISTEMAS DE BANCO DE DADOS

Neste capítulo são abordados alguns conceitos de sistemas de banco de dados, destacando suas características operacionais mais importantes, os tipos de arquiteturas nos quais os Bancos de Dados são executados. E também apresenta a especificação de seus componentes que atendem às funcionalidades do Sistema de Banco de Dados, seus conceitos, suas principais funções e a importância dos modelos de dados utilizados.

2.1 Introdução

Atualmente, o Banco de Dados é a tecnologia mais utilizada para armazenar dados e gerir informações para posterior recuperação ou atualização dessas informações por um usuário, através de uma aplicação ou acessando um sistema de banco de dados.

Para Chiavenato (1985, p. 336):

> Assim como os dados não constituem informação, a informação, isoladamente, não é significativa. Se os dados exigem processamento (classificação, armazenamento e relacionamento) para que possam realmente informar, a informação também exige processamento para que possa adquirir significado.

Na necessidade de se trabalhar com o BD, o processamento trouxe à Ciência da Computação uma solução para manipulação, reutilização e otimização dos sistemas. Conseqüentemente, os dados exigem processamento para que possam ser muito bem especificados com ênfase no sistema onde se armazenam os dados para posterior utilização.

Um Banco de Dados possui os dados classificados numa ordem pré-determinada conforme um projeto de sistema, sempre para um propósito definido. Logo, um BD é uma fonte de onde se podem extrair informações derivadas, que possui um patamar de interação com eventos como a globalização e tecnologia que representa (HENRY, 2006).

2.2 Características

Os bancos de dados e a sua tecnologia vêm gerando um impacto maior na utilização de computadores, em qualquer setor em que estes podem ser atuados. Essa tecnologia é

aplicada aos métodos de armazenamento de informações, os chamados Sistema de Banco de Dados (SBDs), Tal difusão é motivada pelas novas características que possuem frente aos sistemas de armazenamentos antecessores. Tais características são destacadas logo a seguir (NAVATHE**,** 2005):

- **Controle de Redundâncias** - A redundância consiste no armazenamento de uma mesma informação, inúmeras vezes em localidades distintas, para atender a diversas aplicações, provocando inconsistência. Na verdade, em um Banco de Dados as informações somente se encontram armazenadas em um único local não existindo duplicação descontrolada dos dados. No entanto, há replicações dos dados, decorrentes do processo de armazenagem do ambiente Cliente-Servidor, totalmente sob controle do Banco de Dados.
- **Compartilhamento dos Dados** - O SGBD deve incluir software de controle de concorrência ao acesso aos mesmos dados, garantindo em qualquer tipo de situação a escrita/leitura de dados sem falha.
- **Controle de Acesso** - O SGDB deve dispor de recursos como o controle de segurança e autorização, sendo que o controle de autorização são recursos que possibilitam selecionar a autoridade de cada cliente, sendo que um usuário poderá realizar qualquer modalidade de acesso, e atualizar outros e ainda poderá somente acessar um conjunto restrito de dados para escrita e leitura. O controle de segurança abrange conceitos tais como: procedimentos de validação e controle, garantia de integridade e controle de acesso, que visam resguardar o banco de dados de uma possível perda ou destruição de dados, seja por falha de programa ou por falha de equipamento.
- **Interface** - Um Banco de Dados disponibiliza formas de acesso gráfico, em linguagem natural, em SQL, ou via menus de acesso, não sendo uma "caixa-preta" somente sendo possível de ser acessada por aplicações.
- **Esquematização** - Um Banco de Dados deverá fornecer mecanismos que possibilitem a compreensão dos relacionamentos existentes entre as tabelas e de sua manutenção.
- **Controle de Integridade –** A maior parte dos SGBDs provê restrições de integridade, que precisam ser aplicadas aos dados. Com isso, o SGBD possui a necessidade de ter mecanismos para permitir a definição das restrições e assegurar a respeito a estas. Um Banco de Dados deverá impedir que aplicações pelas interfaces possam comprometer a integridade dos dados.

- **Backups** - O SGBD deverá apresentar facilidades para recuperar falhas de hardware ou software. Estes mecanismos evitam que cada aplicação tenha que projetar e desenvolver seu próprio controle contra a perda de dados. Porém têm-se aplicações que são comprometidas por falhas de hardware, que a modalidade de falha não causa a perda de dados.

Portanto, com base em Silberschatz (2006) um banco de dados é:

- Uma coleção lógica coerente de dados com um significado próprio. E, não uma disposição desordenada dos dados;
- Projetado com dados para um propósito específico;
- Um conjunto que possui pré-definido de clientes e aplicações; e,
- Uma representação do aspecto do mundo real.

2.3 **Arquitetura**

O modelo de sistema de computador nos quais os bancos de dados são executados é composto por quatro categorias ou plataformas: as arquiteturas de banco de dados Centralizados, Cliente/Servidor, Distribuído e Paralelo. Os quatro se distinguem, principalmente, no local onde realmente sucede o processamento dos dados.

A arquitetura do próprio SGBD não define, basicamente, o tipo de sistema de computador no qual o banco de dados precisa rodar; contudo, certas arquiteturas são mais adequadas para determinadas plataformas do que para outras.

A seguir, apresenta-se uma breve descrição do funcionamento da arquitetura de BDs, com base em Silberschatz (2006).

2.3.1 **Bancos de Dados Centralizados**

Os Bancos de Dados Centralizados são aqueles que são executados sobre um único sistema computacional que não interagem com outros sistemas. Nessa abordagem, o modo como os computadores são utilizados são caracterizados em dois tipos: por sistema de um único usuário e sistemas multiusuários. Os Sistemas de Banco de Dados planejados para serem monousuário, como computadores pessoais, não costumam apresentar muitos recursos

comuns aos bancos de dados multiusuários. Logo, eles não oferecem suporte ao controle de concorrência, o que não é necessário quando um único usuário pode gerar atualizações.

Os sistemas multiusuários trazem como principal vantagem a confiabilidade em permitir que múltiplos usuários manipulem um grande volume de dados.

2.3.2 **Banco de Dados Cliente-Servidor**

Nesta arquitetura, o termo cliente-servidor é classificado em duas categorias, *front-end* e *back-end*. Onde o termo cliente são os programas de aplicações e as interfaces de usuário, que acessam o banco de dados e que são processados no módulo Cliente.

O módulo cliente é denominado de *front-end* e o termo Servidor trata de armazenamento de dados, acessos, pesquisas e outras funções. O módulo Servidor é o próprio SGBD denominado de *back-end*. Portanto, o termo *back-end* controla as estruturas de acesso, desenvolvimento e otimização de consultas, controle de concorrência e recuperação dos dados e o termo *front-end* corresponde às ferramentas, como a interface do usuário da SQL, interfaces de formulários, ferramentas de geração de relatórios. Logo, a interface entre o *front-end* e o *back-end* é feita por meio da linguagem SQL ou de um programa de aplicação. Nessa arquitetura, o cliente (*front-end*) executa as tarefas do aplicativo, ou seja, fornece a interface do usuário (tela e processamento de entrada e saída). Em seguida, o servidor (*back-end*) executa as consultas no DBMS e retorna os resultados ao cliente.

Apesar dessa arquitetura bastante popular, são necessárias soluções sofisticadas de software que aprovem: o tratamento de transações, as transações *commits e rollbacks*, linguagens de consultas (*stored procedures*) e gatilhos (*triggers*).

Com a evolução dos sistemas centralizados, estes agem atualmente como sistemas servidores que atendem à solicitação de sistemas clientes.

Os sistemas servidores são classificados como servidores de transações e servidores de dados. Os de transações são caracterizados por sistemas servidores de consultas, que proporcionam uma interface, a qual os clientes podem enviar pedidos para uma determinada ação e, em resposta, eles executam a ação e retornam os resultados aos clientes. Usuários podem enviar pedidos por SQL ou por meio de um programa de aplicação, usando um mecanismo de chamada de procedimento remoto.

Já os sistemas servidores de dados permitem que os servidores interajam com clientes que fazem solicitações de leitura e atualização de dados em unidades como arquivos ou

páginas. Por exemplo, servidores de arquivos proporcionam uma interface de sistema de arquivo no qual os clientes podem criar, atualizar, ler e remover arquivos.

2.3.3 **Banco de Dados Paralelos**

A ação principal por trás dos sistemas de banco de dados paralelos é a demanda de aplicações que precisam consultar banco de dados extremamente grandes ou que tenham de processar um volume enorme de transações por segundo. Os sistemas de banco de dados centralizados e cliente-servidor não são poderosos o suficiente para tratar desse tipo de aplicação.

No processamento paralelo, várias operações são realizadas simultaneamente, ao contrário do processamento serial em que as etapas do processamento são sucessivas. Os sistemas paralelos melhoram as velocidades de processamento e E/S por meio do uso paralelo de diversas CPUs e discos. Portanto, o paralelismo é usado para oferecer ganho de velocidade, e as consultas são executadas mais rapidamente porque são oferecidos mais recursos, assim como os processadores e discos. O paralelismo de E/S é referente à redução de tempo exigida para apanhar relações do disco, particionando as relações sobre vários discos. No banco de dados paralelos, a forma mais comum de particionamento de dados é o particionamento horizontal, em que as tuplas de uma relação são divididas entre vários discos.

2.3.4 **Bancos de Dados Distribuídos**

Nesta arquitetura, a informação está distribuída em diversos servidores. Sua característica básica é a existência de diversos programas aplicativos consultando a rede para acessar os dados necessários. Sendo que nos sistemas de banco de dados distribuídos, os dados são compartilhados entre os vários computadores através de atualizações enviadas pelas conexões diretas (na mesma rede). Além do mais, nos banco de dados distribuídos, os computadores não compartilham memória principal ou discos.

As distinções entre os bancos de dados paralelos sem compartilhamento e os bancos de dados distribuídos são que os bancos de dados distribuídos normalmente estão separados geograficamente, são administrados separadamente e possui uma interconexão mais lenta,

outra diferença é que, nos sistemas distribuídos, são distinguidas transações locais de transações globais. A transação local acessa um único site, justamente no qual ela se inicia. Uma transação global, por outro lado, é aquela que acessa diferentes sites, ou a outro site além daquele em que se inicia.

No compartilhamento dos dados, a principal vantagem de um sistema de banco de dados distribuído é criar um ambiente no qual os usuários de um site podem ter acesso a dados disponíveis em outros sites. Por exemplo, em um esquema bancário, os usuários de uma agência podem ter acesso aos dados de outra agência.

Em um banco de dados distribuído, no caso da falha de um site, os sites restantes são capazes de continuar operando. Além do mais, se itens de dados forem replicados em vários locais, e uma transação precisar de um item de dados em particular, ela poderá encontrar esse item de dados entre os vários sites. Neste caso, ao ocorrer à falha de um site, esta não envolve necessariamente o desligamento do sistema.

2.4 Componentes

Segundo Silberschatz (2006), um sistema de banco de dados está dividido em componentes, que atendem às funcionalidades do sistema. Algumas das funções do SBDs podem ser fornecidas pelo Sistema Operacional (SO), portanto o projeto do BD deve considerar a interface entre o SBD e o SO.

Os componentes funcionais do sistema de banco de dados são divididos em componentes do Processador de Consultas e em componentes do Gerenciador de Memória. A Figura 1 mostra os componentes de uma arquitetura geral do SBD.

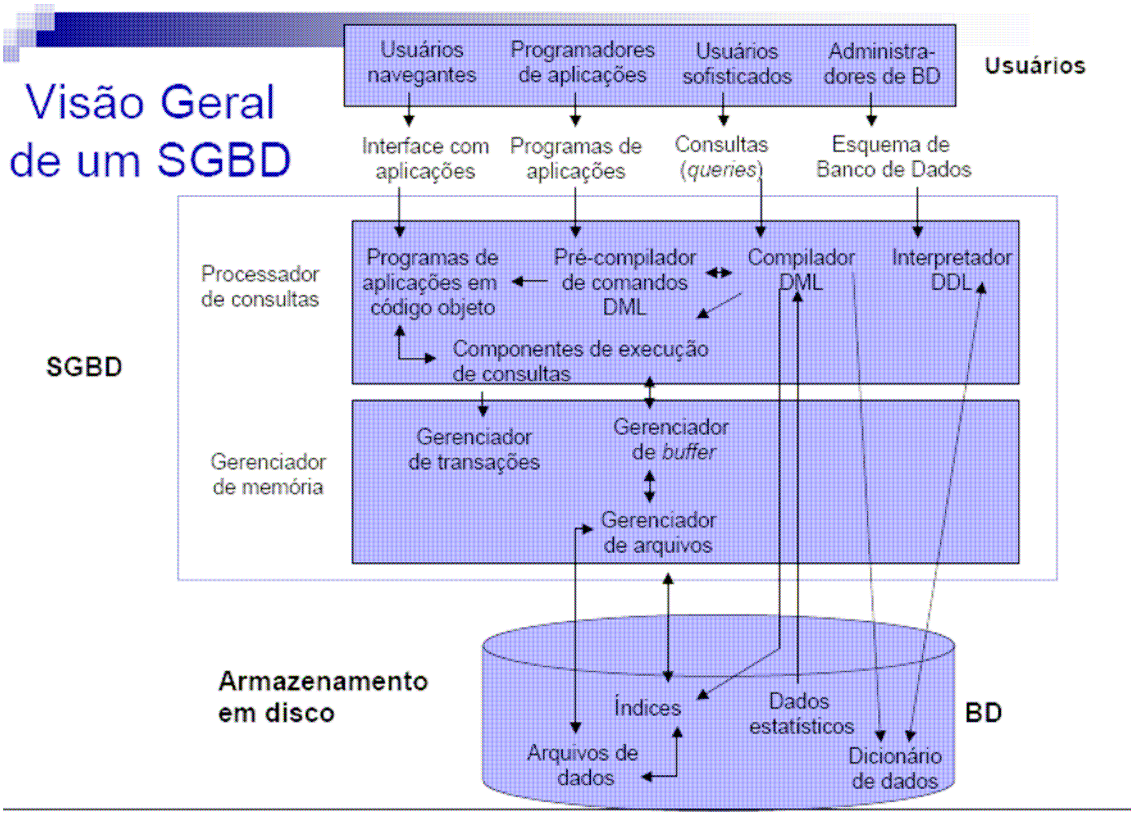

Figura 1: Visão Geral do SGBD
Fonte: Silberschartz (2006)

Os componentes do Processador de consultas são responsáveis por traduzir os comandos em uma linguagem de consulta para instruções de baixo nível em que o gerenciador do banco de dados pode interpretar. Além disso, o processador de consultas tenta transformar uma requisição do usuário em uma maneira compatível e mais eficiente com respeito ao banco de dados, encontrando uma boa estratégia para executar a consulta (SANCHES, 2005). Para tal, este é composto pelos seguintes elementos: Compilador DML, Pré-compilador para comandos DML, Interpretador DDL, Componentes de avaliação de consultas. Esses elementos são detalhados a seguir:

- **Compilador DML:** traduz os comandos DML da linguagem de consulta em instruções de baixo nível acessíveis ao componente de execução de consultas. O compilador DML transforma a solicitação do usuário em uma solicitação equivalente, selecionando a melhor estratégia para execução da consulta ou o melhor plano de execução que seja mais eficiente para o processamento da consulta.

- **Pré-compilador para comandos DML:** inseridos em programas de aplicação, que convertem comandos DML em chamadas de procedimentos normais da

linguagem hospedeira. O pré-compilador precisa interagir com o compilador DML de modo a gerar o código apropriado.

- **Interpretador DDL:** Interpreta os comandos DDL e registra-os em um conjunto de tabelas que contêm metadados, armazenados no dicionário de dados.

- **Componentes de avaliação de consultas:** Executam instruções de baixo nível geradas pelo compilador DML.

Os componentes do Gerenciador de memória são responsáveis por fornecer a interface entre os dados de baixo nível armazenados no disco e os programas aplicativos e de consulta submetidos ao sistema. O gerenciador de memória fornece a interação com o gerenciador de arquivos. Os dados brutos são armazenados no disco usando um sistema de arquivos, que normalmente é fornecido por um sistema operacional convencional. O gerenciador de memória traduz as várias instruções DML em comandos de sistema de arquivos de baixo nível. Logo, o mesmo é responsável por armazenar, recuperar e atualizar dados no banco de dados (SILBERSCHATZ, 2006). Para tal, é composto pelos seguintes componentes: Gerenciador de transações, Gerenciador de arquivos, Gerenciador de buffer. Esses componentes são detalhados a seguir:

- **Gerenciador de transações:** garante que o banco de dados permanecerá em um estado consistente (correto) a despeito de falhas no sistema e que execuções de transações concorrentes sejam executadas sem conflitos em seus procedimentos.
- **Gerenciador de arquivos:** controla a alocação de espaço no armazenamento em disco e as estruturas de dados usadas para representar estas informações armazenadas em disco.
- **Gerenciador de buffer:** responsável pela intermediação de dados do disco para a memória principal e pela decisão de quais dados colocarem em memória cachê. O gerenciador de buffer permite que o banco de dados manipule tamanhos de dados que sejam muito maiores do que o tamanho da memória principal.

Além disso, algumas estruturas de dados são exigidas como parte da implementação física do sistema, estas estruturas de dados são apresentadas a seguir:

- **Arquivo de dados:** armazena o próprio banco de dados.
- **Dicionário de dados:** armazena os metadados relativos à estrutura do banco de dados. O mesmo contém o esquema do Banco de Dados, suas tabelas, índices, forma de acesso e relacionamentos existentes.
- **Índices:** proporcionam acesso rápido aos itens de dados que são associados a valores determinados.
- **Estatísticas de dados:** armazenam informações estatísticas relativas aos dados contidos no banco de dados. Essas informações são usadas pelo processador de consultas para seleção de meios eficientes para execução de uma consulta.

2.5 Modelo de Dados

Um modelo de dados é caracterizado como um grupo de ferramentas conceituais que são utilizadas para descrever os dados; o relacionamento entre esses dados, e as normas de consistência. A sua função principal é buscar a organização de dados de estratégia de processamento que otimizem a performance de acesso e métodos de acesso que sejam independentes na ação (SILBERSCHATZ, 2006).

Nas seções seguintes são mostrados os variados modelos de dados desenvolvidos. De acordo com a classificação de Silberschatz (2006), os modelos de dados podem ser divididos em três categorias: modelo relacional, modelo de dados baseado em objeto, modelo de dados semi-estruturado.

2.5.1 Modelo de dados Relacional

O modelo Relacional é o modelo mais usado atualmente e a maioria dos SGBDs é baseado nele. É um sistema que utiliza um modelo baseado em relações e tuplas. No modelo relacional, o banco de dados é estruturado em tuplas de tamanho fixo de vários tipos. Cada tupla define um número fixo de campos, ou atributos e as relações são usadas para poder representar os dados e o relacionamento entre eles. De uma maneira informal, cada tupla de uma relação representa uma linha da tabela que corresponde a uma coleção de valores de dados relacionados.

2.5.2 Modelo de dados baseado em objeto

O modelo orientado a objetos possui várias características, que são importantes em um SGBDOO, nessas características são incluídas noções de encapsulamento, métodos (funções) e identidade de objetos, tendo por base um conjunto de objetos (entidades). Sendo que um objeto contém valores armazenados em várias instâncias dentro do objeto. Um objeto também contém conjunto de códigos que operam esses objetos. Estes conjuntos de códigos são chamados de métodos. Os objetos que contém os mesmos tipos de valores e os mesmos métodos são agrupados em classes, onde uma classe pode ser vista como uma definição de tipo para objetos.

2.5.3 Modelo de dados semi-estruturado

O modelo permite que os esquemas em dados semi-estruturados possam ser definidos depois dos dados, ou seja, não há nenhuma exigência de um esquema predefinido para o qual os objetos de dados precisam estar em conformidade. Na verdade, é assegurado que dados semi-estruturados são dados nos quais os esquemas de representação está presente (de forma explicita ou implícita) juntamente com o dado, ou seja, a informação de esquema, como nome de atributos, relacionamentos e classes (tipos de objetos) estão misturadas com os objetos e seus valores de dados na mesma estrutura. Alem disso, cada objeto de dados pode ter atributos diferentes que não são conhecidos com antecedência, esse tipo de dado é conhecido como dados autodescritivo.

As características de dados semi-estruturados são (RIBEIRO, 2005):

Definição à posteriore: esquemas para dados semi-estruturados são definidos após a existência dos dados, com foco em uma investigação de suas estruturas particulares e da análise de similaridade e diferenças. Isto não significa que sempre existe um esquema associado a um dado semi-estruturado.

Estrutura irregular: coleções extensas de dados semanticamente similares estão organizados de formas distintas, podendo algumas ocorrências conter informações incompletas ou adicionais em relação a outras e não há um esquema padrão para esses dados.

Estrutura parcial: apenas parte dos dados disponíveis pode ter alguma estrutura, seja implícita ou explícita. Como conseqüência, um esquema para estes dados nem sempre é

completo do ponto de vista semântico e nem sempre todas as informações esperadas estão presentes;

Estrutura extensa: a ordem de proporção de uma estrutura para estes dados são elevados, uma vez que os mesmos são muito heterogêneos.

Estrutura descritiva e não prescritiva: dado à natureza irregular e evolucionária dos dados semi-estruturados, as estruturas de representação implícitas ou explícitas se restringem a descrever o estado corrente de poucas ocorrências de dados similares.

Estrutura evolucionária: a estrutura dos dados modifica-se tão freqüentemente quanto os seus valores. Dados Web apresentam este comportamento, uma vez que existe o interesse em manter dados sempre atualizados;

No capítulo seguinte será apresentada uma visão geral dos Sistemas Gerenciadores de Banco de Dados no que se refere aos modelos Relacional, Orientado a Objetos e ao XML.

3. SISTEMAS GERENCIADORES DE BANCO DE DADOS

Neste capítulo são abordados alguns conceitos de SGBDs Relacional, Orientado a objetos e XML, mostrando os seus objetivos, o funcionamento de cada um, suas características mais importantes, e a especificação dos componentes as quais atendem às funcionalidades do sistema.

3.1 SGBD Relacional

3.1.1 Modelo Relacional

O modelo relacional foi o primeiro modelo de dados utilizado para aplicações comerciais e tem sido implementado em vários sistemas. O sistema relacional é um gerenciador de banco de dados focado no modelo de relações entre tabelas, que recebe pedido de acesso de determinados usuários, e analisa pedidos, dentro de todas as restrições acionadas nesse banco de dados.

Importante afirmar que o modelo relacional foi lançado devido à necessidade de elevar a independência dos dados com relação às aplicações dos clientes e prover um conjunto de funções proporcionadas por intermédio da utilização da álgebra relacional para armazenamento e recuperação de dados (NAVATHE, 2005). Além disso, um SGDB relacional é um software com recursos específicos para facilitar a manipulação das informações de um banco de dados e o desenvolvimento de programas. Como por exemplo, *Oracle, Paradox, MySQL, Intarbase, Sybase*.

Para Navathe (2005. p. 4):

> Um banco de dados pode ser criado e mantido por um conjunto de aplicações desenvolvidas especialmente para esta tarefa ou por um Sistema Gerenciador de Banco de Dados (SGBD). Um SGBD permite aos usuários criarem e manipularem banco de dados de propósito geral. O conjunto formado por um banco de dados

> mais as aplicações que manipulam o mesmo é chamado de “Sistema de banco de dados”.

A seção a seguir aborda os principais conceitos envolvidos na estrutura dos bancos de dados relacionais (NAVATHE**,** 2005).

3.1.2 Estrutura dos bancos de dados relacionais

O SGBD pode ser considerado um sistema de software de propósito geral, que facilita os processos de definição, construção, manipulação e compartilhamento da base de dados entre os vários usuários e aplicações. Sendo que, o processo de definição de um banco de dados envolve em especificar os tipos de dados, as estruturas e as restrições para os dados a serem armazenados.

A construção de uma base de dados é o processo de armazenar os dados em alguma mídia apropriada controlada pelo SGBD. A manipulação inclui algumas funções, como pesquisas em um banco de dados para a recuperação dos dados, atualização do banco de dados para refletir as mudanças no mini-mundo e gerar os relatórios dos dados e por ultimo compartilhamento da base de dados que permite aos múltiplos usuários e programas acessar, de forma concorrente, o banco de dados.

Um banco de dados relacional é constituído de uma coleção de relações (tabelas). Em uma relação, cada tupla é constituída por uma coleção de valores de dados relacionados e os valores são denominados instâncias de uma entidade. Dá-se o nome de instância de uma relação ao conjunto de tuplas de uma relação sendo importante a coleção de esquema de relações (NAVATHE,2005).

As tabelas são estruturas lógicas administradas pelo sistema gerenciador de banco de dados para armazenagem. A estrutura implica na criação de campos que devem ser procedidos conforme as informações que estão contidas nesses campos. As tabelas possuem chaves de recuperação e identificação de uma tupla dentro da tabela.

As chaves de identificação podem ser classificadas como chaves primárias e secundárias, sendo que a chave primária define que cada tupla seja única dentro da tabela. Desta forma, com a chave primária cria-se uma identificação única, o que dá total segurança para que aplicações possam acessar alterar e excluir dados sem correr o risco de apagar ou alterar dois campos da tabela ao mesmo tempo.

As chaves secundárias se distinguem da chave-primária por não identificar unicamente uma tupla e pode ser utilizada para buscas simultâneas de várias chaves**.** Desta forma, a recuperação das informações em um SGBD relacional se faz em conjuntos de tuplas que comporão uma nova relação.

A seguir é apresentado o exemplo da relação de cada Disciplina, onde cada tupla da relação representa os valores dos atributos da relação DISCIPLINA. Os nomes dos atributos são: Identificador de Disciplina, Número do Curso, Semestre, Ano e Instrutor. Os valores dos atributos especificam como interpretar cada tupla, com base na coluna em que cada valor se encontra.

ID_Disciplina	Num_Curso	Semestre	Ano	Instrutor
85	MAT2410	Segundo Semestre	98	King
92	CC1310	Segundo Semestre	98	Anderson
102	CC3320	Primeiro Semestre	99	Knuth
112	MAT2410	Segundo Semestre	99	Chang
119	CC1310	Segundo Semestre	99	Anderson
135	CC3380	Segundo Semestre	99	Stone

Quadro 1: Quadro da Relação Disciplina
Fonte: Navathe (2005)

Segundo Navathe (2005), no modelo relacional formal, uma linha é destacada como tupla, um cabeçalho de coluna é conhecido como atributo e para cada atributo, existe um conjunto de valores permitidos, chamado domínio do atributo em questão. Importante que, para o atributo ID_Disciplina, o domínio é o conjunto de todos os IDs da disciplina. O tipo de dado que descreve os tipos de valores que podem aparecer em cada coluna é representado pelo domínio de valores possíveis.

A seção a seguir apresenta as restrições definidas para o modelo relacional com o objetivo de manter a integridade e consistência dos dados armazenados.

3.1.3 Restrições de Integridade

Um dos objetivos especiais de um SGBD é a integridade de dados, e dizer que os dados de um banco são íntegros, quer dizer que eles refletem corretamente a realidade verificada pelo banco de dados e são consistentes entre si.

Uma restrição de integridade é uma regra de consistência de dados mantida pelo próprio SGBD. As restrições de integridade fornecem meios para assegurar que transformações realizadas no banco de dados não resultem na perda da consistência sobre os estes dados.

No conceito relacional, as restrições de integridade são divididas da seguinte forma:

- **Integridade de Domínio:** Cada atributo tem um conjunto possível de valores chamado domínio. Um valor denominado para um campo deve estar contido no domínio previsto para aquele campo e o princípio é garantir a veracidade dos valores.

- **Integridade de Vazio:** Por meio desta restrição de integridade é possível determinar se um campo pode conter valores nulos. Os campos que compõem a chave primária devem ser diferentes de vazio.

- **Integridade de entidade:** A chave principal de qualquer relação não pode ter valor nulo em nenhuma tupla da relação.

- **Integridade de Chave:** Restrição que determina que os valores de chaves primárias e alternativas devem ser únicos. Não poderá haver duas entidades com o mesmo valor de chave primária;

- **Restrição de unicidade de chave:** Uma chave principal não pode ter o mesmo valor em duas tuplas distintas da mesma relação.

- **Integridade Referencial:** A definição de integridade referencial envolve basicamente duas relações e o conceito de chave estrangeira, sendo utilizada para manter a consistência entre tuplas de duas relações. A restrição de integridade referencial especifica que valores dos atributos que são denotados em uma chave estrangeira devem estar presentes na coluna da chave primária da tabela referenciada, onde esta referência é realizada por meio da chave estrangeira.

Na verdade, o conceito de chave primária é importante para o entendimento de como funciona um banco de dados baseado no modelo relacional. Ao definir um campo como sendo uma chave primária, informa-se ao banco de dados que não há conjunto de entidades com o mesmo valor do atributo definido pela chave primária.

3.1.4 Linguagens de Consulta

Outra forma de consulta em um banco de dados são as linguagens de consulta utilizadas pelo cliente para solicitar as informações contidas na base de dados. Essas linguagens formam uma linguagem de nível mais alto que as linguagens de programação tradicionais.

Os sistemas de banco de dados comerciais oferecem uma linguagem de consultas que incorpora elementos de dois enfoques: procedurais e não-procedurais. Em uma linguagem procedural, o cliente deve "ensinar" ao sistema como deve ser a realização de uma seqüência de operações no banco de dados para obter o resultado desejado. Em uma linguagem não-procedural, o cliente descreve a informação desejada sem fornecer um procedimento específico para a obtenção dessas informações.

Na seção a seguir detalham-se as principais operações usadas na álgebra relacional (SILBERSCHATZ, 2006).

3.1.5 Álgebra Relacional

A álgebra relacional é uma linguagem de consulta formal, porém procedural. A forma de trabalho está baseada em selecionar relações como entrada de dados e produzir uma nova relação como resultada das operações. A álgebra é utilizada como base para otimizar as consultas em sistemas gerenciadores de banco de dados relacional (SGBDRs).

As operações básicas utilizadas na álgebra relacional são: seleção, projeção, produto cartesiano e junção. As operações de seleção e projeção são chamadas de operações unárias, pois operam sobre uma única relação. As operações de produto cartesiano e junção operam sobre um par de relações e são chamadas operações binárias.

A seguir são definidas as operações principais da álgebra relacional. Considere R, R1 e R2 relações.

- **Seleção:** produz uma nova relação que contém todas as tuplas que satisfazem à condição de seleção em uma relação R;

- **Projeção:** produz uma nova relação com apenas alguns dos atributos de R e remove as tuplas repetidas;

- **Produto Cartesiano:** produz uma nova relação que possui os atributos de R1 e R2 e inclui, como tuplas, todas as possíveis combinações de tuplas de R1 e R2;

- **Junção:** constrói uma relação a partir de duas relações consistindo em todas as possibilidades de pares de tuplas concatenadas, uma de cada uma das duas relações, de forma que em cada par as duas tuplas satisfaçam uma condição única;

3.2 SGBD Orientado a Objetos

O SGBD Orientado Objeto é mais adequado para o tratamento de objetos complexos. Esses objetos são classificados como estruturados e não estruturados. Sendo que um objeto complexo não estruturado possui um tipo de dado que requer um grande volume de armazenamento, por adquirir novos tipos de dados para armazenamento de imagens ou textos longos. E os estruturados são definidos pela aplicação de determinados construtores de tipos, como, conjunto (coleções), tupla, lista ou array (ordem) (NAVATHE, 2005).

Conforme Setzer (2005**),** o desenvolvimento do referido SGBD teve combinação de idéias dos modelos de dados tradicionais e das linguagens de programação orientada a objetos. E, o contexto rico desses objetos é verificado no nível lógico e possui características não encontradas nas linguagens de programações tradicionais, como operadores de manipulação de estruturas, gerenciamento de armazenamento e outros.

Na verdade, para que um sistema de banco de dados seja considerado orientado a objetos, é importante a presença de Identificadores de Objetos (OIDs), mecanismo de herança (única ou múltipla), objetos complexos e persistência de objetos.

A seguir são apresentados alguns conceitos importantes para a orientação a objetos que foram recomendados para adicionar as funcionalidades de banco de dados às linguagens de programação orientada a objetos.

3.2.1 Objetos Complexos

Quando nos referimos a um objeto, o mesmo possui, caracteristicamente, dois componentes: estado (valores) e comportamento (operações). Assim, é semelhante a uma

variável de programa em uma linguagem de programação, exceto que geralmente terá uma estrutura de dados complexa.

Um conceito mais simples relacionado a um objeto é que o mesmo é uma entidade lógica que contém dado e código com o objetivo de manipular esses dados. Os dados são destinados como sendo atributos do objeto, e o código que o manipula é denominado de método. Sendo que um método é uma função que manipula a estrutura de dados do objeto.

Objetos complexos são formados por construtores (conjuntos, listas, tuplas, registros, coleções, arrays) aplicados a objetos simples (inteiros, booleanos, strings). A distinção do modelo OO em relação ao modelo relacional, é que no modelo OO qualquer construtor pode ser aplicado a qualquer objeto, já no modelo relacional este não é o caso, visto que só é possível aplicar o construtor de conjuntos nas tuplas e o construtor de registros nos valores atômicos (ARBEGAUS, 2003). Para tal, existem praticamente dois tipos de objetos complexos em um SGBDOO:

- **Objetos embutidos:** são caracterizados como "objetos filhos", que constituem os atributos, que só podem ser acessados pelo seu "objeto-pai". São resultados da estrutura de agregação ou "todo-parte". Objetos embutidos não possuem OID próprio e são, em geral, armazenados na mesma estrutura física de seu "objeto pai". Os mesmos possuem como exemplo os casos dos relacionamentos dependentes, como os "itens de um pedido de venda". Nesse exemplo, os "itens" são objetos embutidos dos "pedidos de venda".
- **Objetos referenciados:** são caracterizados como objetos originários das regras de integridade referencial. Qualquer relacionamento, como por exemplo, de uma "Cidade" com o seu respectivo "Estado" corresponde à criação de um objeto composto do tipo referenciado. Os objetos referenciados possuem OID próprio e podem ser acessados diretamente ou através de seus objetos relacionados.

3.2.2 Identificador de Objetos

No SGBDOO, o OID é dito persistente, ou seja, a identidade do objeto persiste não só entre execuções de programas, mas também durante reorganizações estruturais de dados. A identidade do objeto é geralmente gerada pelo sistema quando o objeto é criado. No entanto, segundo Silva (2001) um OID em SGBDOO não pode ser:

a) Um endereço (como um nome de variável ou referencia de memória de uma linguagem de programação) porque não é externo ao objeto;
b) Uma chave (como chave primária de um BDR) porque não é um valor de dados mutável;
c) Um *register* ID porque não é uma coluna lógica.

Para Silva (2001) a identidade de objetos tende a eliminar anomalias de atualização e de integridade referencial, uma vez que a atualização de um objeto será automaticamente refletida nos objetos que o referenciam e que o identificador de um objeto não tem seu valor alterado.

Para Navathe (2005), a identidade única de cada objeto armazenado na base de dados é geralmente implementada por meio de um identificador de objeto (OID). Sendo que o valor de um OID não é visível para um usuário externo, mas é utilizado internamente pelo sistema, com a finalidade de identificar univocamente cada objeto.

3.2.3 Herança única ou múltipla

Um conceito essencial no banco de dados OO é que a herança é um tipo de relacionamento entre classes, onde uma classe compartilha (herda) estrutura de dados (atributos) e métodos (operações) de outras classes.

Logo, nesse banco de dados há dois tipos de herança, que são classificados como herança simples e múltipla. Na herança simples, uma classe é subclasse de apenas uma superclasse. Na herança múltipla, uma classe pode ser subclasse de várias superclasses, herdando variáveis e métodos de várias classes (GONÇALVES**,** 2008).

As “Classes de Coleção” ou “Classes de Sistema” são exemplos principais das características de herança em um SGBDOO. A maior parte desses SGBDs possuem uma coleção própria de classes de objetos, onde estas classes são organizadas numa hierarquia. Na verdade, uma determinada aplicação pode desenvolver sua própria classe e herdar atributos ou métodos de alguma classe do sistema. Logo, as classes de sistema dos SGBDOO implementam os métodos de armazenamento, manipulação, controle de concorrência, entre outros (ARBEGAUS, 2003).

Para Arbegaus (2003), os SGBDOO armazenam as declarações das classes, confeccionando-as como parte do esquema do banco de dados. Logo, uma hierarquia de classes proporciona muito mais flexibilidade para modificar a estrutura de um banco de dados

(incluindo novos atributos ou métodos nos objetos) possibilitando a evolução do esquema do banco de dados por meio da adição de novas classes na hierarquia.

3.2.4 Persistência de objetos

Nas linguagens de programação OO, durante a execução do programa, ao se eliminar um objeto que foi instanciado, ou criado, ele passa a deixar de existir. Esses objetos são denominados como transientes, pois só existem durante a execução do programa e desaparecem quando o programa termina (SETZER, 2005).

Para garantir que um objeto continue persistindo após o término do programa é preciso que o banco de dados OO estenda a existência desses objetos de uma maneira que sejam armazenados de forma permanente na base de dados. Para isso acontecer, deve ser associado a um mecanismo de nomeação, que consiste em dar a um objeto um nome persistente único pelo qual ele possa ser recuperado pelo programa em execução e outros programas. Este nome de objeto persistente pode ser atribuído por meio de um comando específico ou uma operação no programa. Todos os nomes atribuídos aos objetos devem ser únicos dentro de um determinado banco de dados. Portanto, os objetos persistentes nomeados são utilizados como pontos de entrada pelos quais os usuários e as aplicações podem iniciar o acesso ao banco de dados. Com isso, um objeto estará realmente armazenado, e todos os seus componentes também serão tornados persistentes (NAVATHE, 2005).

Na visão de Setzer (2005. p. 280):

> A solução mais simples adotada foi a de se estabelecer que a persistência seja uma característica de todos os objetos criados em um processamento. Uma outra foi a de se especificar quais objetos devem ser persistentes, isto é, quando eliminados durante ou no fim do processamento, os valores de seus atributos não são perdidos. Essa é a solução mais flexível, porém mais complexa.

3.2.5 Características

A seguir são apresentadas algumas características que são importantes em um SGBDOO.

- **Tipos abstratos de dados (Classes):** É definido como um conjunto de objetos que compartilham a mesma estrutura e o mesmo comportamento (operações). Sendo que um objeto é uma instância de uma classe, ou seja, suas características são definidas na classe a qual foi instanciado (GONÇALVES, 2008).

- **Encapsulamento:** Essa definição se refere à independência dos dados de um objeto em relação a outros objetos, isto é, nenhum método de um objeto pode obter ou alterar diretamente os dados de outro. Desse modo, o objetivo do encapsulamento é de ocultar detalhes da implementação dos métodos e da estrutura dos atributos (SETZER, 2005).

- **Identificador de objeto:** Os objetos possuem identificadores únicos que são independentes de seus valores de atributos.

- **Polimorfismo ou Sobrecarga de Operador:** No conceito de orientação a objetos, dentro de uma classe, operações podem ser sobrecarregadas para serem aplicadas a diferentes tipos de objetos com diferentes implementações (NAVATHE, 2005).

Para Setzer (2005. p. 277)**:**

> Uma classe da OO contém um conjunto de dados, denominados de variáveis de classes ou atributos, e um conjunto de procedimentos, denominados de métodos, formando um todo. Os dados de uma classe são globais a todos os métodos da mesma, isto é, estão disponíveis em todos estes. Os dados de uma classe só podem ser alterados pelos métodos da classe (e de subclasses, no mecanismo de herança).

3.2.6 Restrições de integridade

A maior parte das restrições de integridade existentes nos bancos de dados tradicionais se aplica também aos SGBDOO. No entanto, devido às construções e ao poder de modelagem do modelo de dados orientado a objetos, algumas restrições deixam de ser relevantes, outros adquirem formas diferentes, logo os BDOO acrescentaram outros tipos de restrições e especificações de integridade (ARBEGAUS, 2003). Para tal, as restrições de integridade são classificadas como Restrições de Integridade de Chave, Restrições de Integridade Referencial, Restrição Existencial, Restrição NOT NULL, Restrições de Pré-Condições e Pós-Condições de Métodos e Restrições de Cobertura.

3.2.6.1 Restrições de integridade de chave

Nos Bancos de Dados Relacionais é freqüente a especificação de chaves para as relações, com o objetivo de identificar exclusivamente as tuplas de uma relação tanto em nível

físico quanto em nível lógico, com o intuito de otimizar os processos de consulta. No entanto, nos BDOOs é função do OID identificar exclusivamente em nível físico as instâncias de uma classe e, para identificação em nível lógico; os BDOO permitem que sejam definidas chaves para uma determinada classe. Nos BDOO as chaves também otimizam os processos de consulta.

3.2.6.2 Restrições de integridade referencial

Os objetos de um banco de dados normalmente estão relacionados com outros objetos no banco de dados. Da existência das relações vem à necessidade da integridade referencial, que tem por objetivo assegurar que objetos não contenham relacionamentos com objetos que não existam mais no banco de dados. As especificações de restrição de integridade referencial garantem que não haja referências "pendentes" para objetos de uma classe.

Referente ao conceito desta restrição, o que distingue o modelo relacional do modelo OO é que, no modelo relacional, o mecanismo mais utilizado para referenciar objetos entre si é o da chave estrangeira, que relacionam os objetos pelos valores de seus atributos. Logo, nos SGBDOO que permitem o conceito de OID do objeto, não há necessidade de restrições de integridade do tipo "chave estrangeira", visto que o OID permite referenciar objetos diretamente.

3.2.6.3 Restrição de integridade existencial

As restrições existenciais suportam somente sistemas que implementam o conceito de OID, a mesma tem a finalidade de assegurar que um objeto compartilhado referencialmente possui um domínio "ativo" (um grupo específico de objetos que no momento existem em função de determinada condição) no qual ele deve existir. Por exemplo: se um escritório de contabilidade possui um sistema de folha de pagamento multi-empresa e no cadastro de empresas do sistema existe um campo denominado "Telefone", logo, consiste numa restrição existencial certificar, que todo o telefone informado no campo "Telefone Comercial" do cadastro de funcionários é um telefone existente no domínio ativo do cadastro de empresas (telefones cadastrados nas empresas).

3.2.6.4 restrição *not null*

Para um SGBDOO suportar restrições *NOT NULL*, ele será obrigado a suportar valores *NULL*. Um valor referente à NULL define que este valor não existe ou está indisponível. Quando uma restrição *NOT NULL* é definida para um atributo, a faixa de valores que esse atributo pode receber corresponde à faixa especifica para o tipo definido para

o atributo. Portanto, os valores *NOT NULL* não são aceitos, como um valor válido, porque são permitidos apenas valores correspondentes à faixa de valores determinados para o atributo.

3.2.6.5 restrições de pré-condições e pós-condições de métodos

As restrições de pré-condições permitem a definição de certas restrições nas variáveis de instância que devem ser satisfeitas, para que um determinado método possa ser executado. Já as restrições de pós-condições permitem a definição de outras restrições que devem ser cumpridas depois da execução do método.

3.2.6.6 restrições de cobertura

As restrições de cobertura determinam que uma superclasse não pode possuir instâncias que não sejam elementos de determinado grupo de suas subclasses. Uma maneira de obter tal restrição consiste em definir a superclasse como sendo classe abstrata. Desta forma não seria possível criar uma instância da superclasse, somente de suas subclasses. Porém, podem existir situações onde seja necessário instanciar uma superclasse, mas mesmo assim manter a restrição de cobertura, sendo que o sistema iria levantar uma exceção somente se esta instância criada fosse salva como um elemento da superclasse e não como um elemento de uma das subclasses de cobertura.

3.3 SGBD XML

No cenário atual, diante da necessidade requerida pelas aplicações de trocar informações com outros sistemas, a linguagem XML (eXtend Markup Language) tem dado o suporte necessário, não só para a descrição de dados estruturados, mas também para os dados não estruturados.

Segundo Graves (2003), o SGBD que fornece manipulação direta aos documentos expressos em XML, onde as informações desses documentos, com seus respectivos relacionamentos hierárquicos, estão devidamente armazenadas em um repositório voltado para essa representação dos dados envolvidos, esse repositório é denominado de SGBD XML.

A tecnologia XML é similar à tecnologia SGBD, pois possui um formato de documento e linguagens de consulta, permitindo conceitos de interfaces (*APIs*) para o acesso a dados. Mas na tecnologia XML não existem soluções consolidadas para todos os aspectos de gerenciamento de dados, como controle de integridade, gerenciamento de transações e

visões. Além do mais, o formato de um dado XML é altamente irregular e muitas vezes combina texto em linguagem natural com informações estruturadas (LIMA, 2008).

A seguir são apresentados dois tipos de modelos:

Um centrado nos dados - documentos que são caracterizados por possuírem uma estrutura regular e normalmente a ordem pela qual os vários elementos com o mesmo nível na estrutura são representados são irrelevantes.

Centrado no documento- caracterizam-se por possuir uma estrutura irregular, ou seja, as unidades mais elementares de informação podem estar ao nível de elementos com conteúdo misto e a ordem com que os elementos são representados é normalmente relevante.

Na verdade estes modelos influenciam a forma como o XML é armazenado de forma persistente em Banco de Dados.

Contudo, antes de detalhar de forma geral o SGBD XML, serão apresentados alguns conceitos da linguagem XML.

3.3.1 Introdução à Linguagem XML

A princípio, a expressão XML significa *Extensible Markup Language*. A linguagem XML foi inicialmente projetada com o objetivo de incluir informações de marcação em documentos de texto, e tem se tornado importante para a necessidade das aplicações na troca de informações (SILBERSCHATZ, 2006).

Dois conceitos principais de estruturação que são usados para construir um documento XML são os elementos e atributos (NAVATHE, 2005). No entanto, o termo atributos em uma linguagem XML, não é usado da mesma maneira como é habitual na terminologia de banco de dados, mas da maneira como é usado em linguagens de descrição de documentos. Portanto, uma string de texto, em uma linguagem XML é usada para representar os dados, pois a mesma contém uma "marcação" intercalada com o intuito de descrever as propriedades dos dados. A utilização dessa marcação possibilita que o texto seja intercalado por informações relacionadas ao seu conteúdo ou forma.

Para Almeida (2002), a linguagem XML possui um componente básico denominado elemento. Um elemento é um pedaço de texto intercalado pelos sinais "<" e ">", como, por exemplo, "<pessoa>" e "</pessoa>", onde uma expressão <pessoa> é chamada marcação inicial, e a expressão </pessoa> é chamada marcação final, e a estrutura entre as marcações é denominada de conteúdo. Para Graves (2003), como uma string em uma linguagem XML, um documento é composto de tags e dados formados por caracteres. E então, estas tags têm a

forma de “<palavras_chave > intercaladas </palavras_chave>” no texto que está sendo marcado.

Muitas dessas informações encontram-se organizadas em documentos XML e provêm de várias fontes. O gerenciamento de conteúdo envolvendo documentos XML heterogêneos carece de mecanismos que orientem o processo de armazenamento de forma a facilitar sua posterior recuperação.

De acordo com Almeida (2002), na linguagem XML é permitido associar “atributos” aos elementos. O termo “atributo” é utilizado no contexto do XML para especificar propriedades ou características do elemento. Para tal, assim como nas marcações, o usuário pode definir também os atributos, como os atributos “língua”, “moeda” e “formato” que são especificados no exemplo do quadro 2. Nesse exemplo são apresentados elementos repetitivos com a mesma marcação para representar coleções de dados e também alguns atributos que são utilizados para explicitar detalhes de um elemento.

A seguir é apresentado um exemplo de documento XML.

Exemplo 1.1

```
<mestrado>
      <descrição> Pessoas que estudam na UFMG </descrição>
      <turma>
            <pessoa>
                  <nome> João</nome>
                  <idade> 30 </idade>
                  <email>joão@ufmg.br </email>
            </pessoa>
            <pessoa>
                  <nome> Cláudia </nome>
                  <idade> 25 </idade>
                  <email> claudia@ufmg.br <email>
            </pessoa>
            <pessoa>
                  <nome> José </nome>
                  <idade> 25 </idade>
                  <email> jose@ufmg.br </email>
            </pessoa>
      </turma>
</mestrado>
...

<produto>
       <nome língua= “inglês”> book </nome>
       <preço moeda= “dolar”> 45,00 </preço>
       <fornecedor formato= “XLB56” língua= “inglês”>
             <rua> Penbridge Square </rua>
             <número> 30 </número>
             <cep> 92310 </cep>
             <país> United Kingdom </país>
       </fornecedor>
</produto>
```

Quadro2: Fragmento de uma página XML

e Fragmento de uma página XML com atributos.
Fonte: Almeida (2002)

De forma genérica, esses trabalhos propõem um modelo de descrição de documentos XML normalmente através de um esquema de numeração, o qual pode ser usado para carregar o documento em tabelas do SGBD.

3.3.2 Características do SGBD XML

Na XML, as aplicações podem ser atualizadas mais rapidamente e também permitem múltiplas formas de visualização dos dados estruturados. Segue abaixo uma lista das principais aplicações onde a XML é usada:

- Um simples documento;
- Um registro estruturado tal como uma ordem de compra de produtos;
- Um objeto com métodos e dados como objetos Java ou controles ActiveX;
- Um registro de dados. Ex: o resultado de uma consulta a bancos de dados;
- Apresentação gráfica, como interface de aplicações de usuário;
- Entidades e tipos de esquema padrões;
- Todos os links entre informações e pessoas na web.

Um modelo de marcação acontece no exemplo apresentado a seguir, onde a marcação representa uma representação XML, onde é criada uma hierarquia de marcação para as anotações. O exemplo anterior e o exemplo a seguir, ilustram dois recursos da XML. No primeiro recurso um documento pode ser marcado e também compreendido; onde adicionar a marcação XML preserva as informações do documento. O segundo recurso da XML é que ela pode representar informações hierárquicas (aninhadas). Portanto, a marcação XML pode adicionar informações a um documento sobre seu conteúdo e estrutura.

A seguir é apresentado um exemplo de marcação XML.

Exemplo 1.2

```
<? xml  version= “1.0”?>
   <sentence language= “English”>
   <subject0fBook><noum>XML</noum></subject0fBook><verb type=“be”>is</verb> a <annotation>
       <text>language</text>
    <note>
        actually an artificial language rather than a
    <annotation>
        <text>natural</text>
    <note>
        by natural language I mean a language developed over time
        By humans for communicating with each other rather than an
    <annotation>
        <text>artificial</text>
    <note>
        by artificial I do not mean that it is not real,
        but that it is an
    <definition>
        <text>artifact</text>
    <def>
        an artifact is any object creat or shaped by humans
    </def>
    </definition>
    </note>
    </annotation>
        language, such as a programming language
        </note>
        </annotation>
        language
       </note>
       </annotation>
<prepPhrase>for <colloquial>marking up</colloquial>text</prepPhrase>
```

Quadro 3: Marcação XML.
Fonte: Graves (2003)

Conforme Graves (2003. p.6) “a XML foi projetada como uma linguagem para transferência de informações entre aplicativos. De fato, a XML pode ser essencial na transferência de dados e por si só não tem por finalidade a apresentação ao cliente final, mas quando empregados junto com as folhas de estilo os documentos XML podem ser visualizados. Os documentos XML provêem uma forma de capturar os dados, e as folhas de estilo como a eXtensible Stylesheet Language (XSL) ou as Cascading Style Sheets (CSS) disponibilizam mecanismos de conversão da XML em HTML (ou outras conversões).”

Essencial, já que a XML pode representar dados e textos com estilo livre, e também tratar de ambos no mesmo documento. Comentários longos podem comentar um único trecho de dados em um conjunto de informações complexa e grande, e os documentos científicos podem ter conjuntos de dados hierárquicos, complexos e extensos incorporados a eles.

Essas razões são aplicáveis não só à XML em geral, mas também simplificam o uso dos bancos de dados XML em aplicativos Web. Também é importante usar XML, porque ela pode ser empregada em banco de dados. Eles podem armazenar os dados como XML e/ou interagir com outros aplicativos que a utilizem.

3.3.3 O Banco de Dados XML

Os documentos XML tendem a ser orientados ao processamento de documentos ou de dados. Os mesmos orientados ao processamento de documentos são aqueles em que a XML é usada por sua habilidade em capturar linguagens naturais (humanas) como as dos manuais de usuários, das páginas Web e de folhetos de propaganda. Eles são caracterizados por estruturas complexas ou irregulares e conteúdo misto, sendo estrutura física importante.

Os documentos orientados ao processamento de dados são aqueles em que a XML é usada principalmente para a transferência de dados. Aí se incluem os pedidos de compra, os registros de pacientes e dados científicos.

Conforme Silberschatz (2006. p.272), "em particular, as ferramentas para a consulta e transformação dos dados XML são essenciais para extrair informações de grandes campos de dados XML e converter os dados entre diferentes representações (esquemas) em XML. Assim como a saída de uma consulta relacional é uma relação, a saída de uma consulta XML pode ser um documento XML. Como resultado, consulta e transformações podem ser combinadas em uma única ferramenta".

A diferença entre os documentos XML orientada ao processamento de dados e os orientados ao processamento de documentos é muito sutil. Entre as operações esperadas em um documento orientado ao processamento de documentos está a recuperação do documento completo, a busca de uma palavra, encontrar palavras antecedentes ou posteriores, alterar uma seção ou reorganizá-la.

Em um documento orientado ao processamento de dados, as operações desejadas incluem a recuperação de um trecho especificado do documento, a busca de uma combinação particular de elementos e dados, a alteração ou a exclusão de um único elemento ou trecho de dados, ou a inserção de um novo elemento no documento.

O SGBD XML fornece um acesso direto aos documentos XML e a trechos deles, e a possibilidade de consultá-los. Usar um SGBD XML é muito importante para a captura de dados de uma área com relacionamentos hierárquicos complexos, como nos banco de dados científicos e em sistemas de produção (GRAVES, 2003).

Para Graves (2003), desenvolver um SGBD XML sobre outro SGBD é mais rápido e fornece muitos benefícios do SGBD subjacente usado para armazenamento. Além disso, pode ser mais simples integrá-los aos dados legados e aplicativos existentes. Portanto, desenvolver um SGBD XML, sobre outro SGBD como, por exemplo, o SGBD relacional existe muitas

vantagens. Pois os dados são protegidos de maneira tão robusta quanto qualquer dado de um SGBD relacional, ele pode ser acessado por todos os aplicativos existentes para banco de dados relacionais, incluindo ferramentas de consulta, navegadores, programas de pesquisa, e em relação às consultas e visões pode ser integrados aos dados legados existentes.

Os SGBDs XML fornecem mecanismos para armazenar, alterar, consultar e excluir elementos e documentos XML contidos em um banco de dados. Um documento pode ser armazenado, alterado, consultado com base em seu conteúdo e excluído baseado em sua entidade, como também podem ser modificados pela inserção, alteração ou exclusão de elementos e o próprio fornece um mecanismo conveniente para organizar os elementos, mas ele não é estritamente necessário na perspectiva de um banco de dados (GRAVES, 2003).

A primeira etapa no projeto de um esquema relacional para XML é criar um esquema conceitual. O assunto pode ser abordado com o uso das definições a seguir extraídas da especificação XML:

- Um documento contém um elemento (raiz).
- Um elemento contém um nome de tipo de elemento, um conjunto de atributos, elementos e dados de caracteres, em que a ordem dos elementos e dos dados formados por caracteres é mantida.
- Um atributo é composto de um nome e um valor.
- Os dados de caracteres são compostos por strings do documento que não sejam tags ou outra marcação.

Para manter simples a apresentação do projeto do banco de dados, a maioria das informações da especificação, exceto as tags e os dados formados por caracteres, é ignorada nesse estágio, ou seja, os comentários, instruções de processamento, referências e declarações. Só as informações básicas do documento são capturadas.

3.3.4 Modelo conceitual XML

A XML pode ser usada como uma linguagem de modelagem conceitual com o objetivo de capturar as informações agregadas de um assunto estruturado hierarquicamente sem o emprego da herança. Um modelo bastante útil pode ser desenvolvido com algumas convenções apropriadas, como as apresentadas a seguir:

- Um esquema conceitual XML é composto de um conjunto de elementos XML.
- A inexistência de valores em atributos significa que eles não apresentam restrições.

A seguir é apresentado um exemplo de um esquema conceitual simples das partes de um livro representado em XML.

Exemplo 1.3

```
<titulo livro= “  ”  autor= “  ”>
        <sumario / >
        <titulo capitulo= “  ”>
              <seção>
                    <Subseção / >
              </secao>
        </capitulo>
        <índice/>
</livro>
```

Quadro4: Esquema Conceitual.
Fonte: Graves(2003)

Para verificação, um esquema conceitual XML descreve os relacionamentos de funcionários, que serão apresentados no exemplo a seguir e o esquema conceitual XML pode ser convertido em um esquema relacional da seguinte forma:

a. Cada elemento se torna uma tabela (relacionamento).

b. Cada atributo XML se torna uma coluna (atributo relacional).

c. Cada subelemento de um elemento se torna uma coluna na tabela, com uma restrição de chave externa para a tabela especificada pelo subelemento.

d. Todos os elementos com o mesmo nome se referem ao mesmo relacionamento (e todos os relacionamentos de subelementos ou atributos são mesclados).

A seguir é apresentado um exemplo de um esquema conceitual dos funcionários representado em XML:

Exemplo 1.4

```
<numero funcionário= “ ” nome= “  ”  tarefa= “  ” data contratação= “  ” comissão= “  ” salario= “  ”>
      <numero departamento= “  ” nome=  “    ” local= “    ” / >
</ funcionário>
```

Quadro5: Esquema Conceitual.
Fonte: Graves (2003)

O exemplo mostrado pode ser transformado nos relacionamentos a seguir: Funcionário (número, nome, tarefa, datacontratacão, comissão, salário, departamento), Departamento (número, nome, local)

3.3.5 Modelagem física

O projeto físico é o processo de capturar de maneira eficiente o esquema geral do modelo de dados lógico nas estruturas da linguagem de implementação relacionada à utilização do SGBD (GRAVES, 2003).

Em banco de dados XML, o banco precisa conhecer a estrutura de documentos XML a serem armazenados. No qual essa estrutura é definida por um DTD ou um *Schema* XML. Portanto, uma notação DTD pode ser usada para definir um esquema. No entanto, ela pode não fornecer o nível de restrições do tipo que são normalmente desejados para um BD. Ela só permite o uso de um tipo de dado, o PCDATA, que define um conjunto de caracteres. A partir daí veio à necessidade do desenvolvimento do esquema XML, pois ele fornece uma maneira de especificar os valores dos dados e a estrutura que podem ocorrer em um documento (PIALARISSI, 2005).

Para Pialarissi (2005), as vantagens do uso do XML *Schema* em relação ao uso da DTD são que o XML *Schema* permite o uso de restrição de valores através do uso de padrões; atribuição de limite máximo ou mínimo; uso de enumeração para elementos; determina um tamanho para o valor do elemento (máximo mínimo ou exato). Alem disso, partindo do grafo da modelagem lógica para um esquema no projeto físico, existem as seguintes etapas de conversão, nos quais os elementos e atributos do esquema são assim definidos: os nodos não terminais tornam-se elementos compostos no esquema; e nodos terminais são convertidos em elementos #PCDATA ou atributos, além do mais, a ordem de sub-elementos é feita baseada na análise das arestas que partem de um nodo não terminal. Um nodo terminal pode ser representado no modelo físico como um elemento ou como um atributo. O nodo terminal representa-se como elemento quando o conteúdo é extenso. Se for o caso de economizar

espaço no documento XML ou definir restrições de integridade, o nodo terminal é representado como atributo no modelo físico. A listagem a seguir trás a DTD gerada a modelagem física.

```
<!ELEMENT LIVROS (LIVRO+)>
<!ELEMENT LIVRO (TÍTULO, AUTOR+, CAPÍTULO+)>
<!ATTLIST LIVRO ISBN  CDATA >
<!ELEMENT TÍTULO (#PCDATA)>
<!ELEMENT AUTOR (NOME, E-MAIL*)>
<!ELEMENT NOME (#PCDATA)>
<!ELEMENT E-MAIL (#PCDATA)>
<!ELEMENT CAPÍTULO (NOME, REFERÊNCIA*)>
<!ATTLIST CAPÍTULO ORDEM CDATA >
<!ELEMENT REFERÊNCIA (#PCDATA)>
```

Quadro6: Modelagem Física
Fonte: Mello (2008)

Em comparação com o esquema XML é mostrado como é a estrutura do esquema relacional, no qual possui como representação a relação FUNCIONARIO com os respectivos atributos, NOME (uma *string*) e NÚMERO_FUNCIONÁRIO (um número) (GRAVES, 2003).

Portanto, o esquema físico correspondente pode descrever a tabela FUNCIONÁRIO obtendo duas colunas como, por exemplo, NOME (uma *string* exclusiva de comprimento variado com menos de 30 caracteres) e NÚMERO_FUNCIONÁRIO (um número com exatamente 4 dígitos no intervalo entre 1000 e 3999).

3.3.6 XML Query

XML *Query* fornece recursos flexíveis de consulta para extrair dados de documentos já existentes e gerados dinamicamente na Web. O objetivo do XML *Query Working Group* é produzir um modelo de dados para documentos XML, um conjunto de operadores de consulta para esse modelo de dados e uma linguagem de consulta com base nesses operadores (GRAVES, 2003).

O XML *Query Data Model* define um modelo de dados orientado a nós para XML que é especificamente personalizado para atender às necessidades de consulta a documentos XML. As operações do sistema de consultas são definidas pela XML *Query* Algebra. O modelo de dados é definido em termos de funções matemáticas que criam árvores a partir da

aplicação recursiva de uma função de geração de árvores a uma lista de árvores filhas. Ele define rigorosamente uma estrutura matemática a qual desenvolve consultas.

O acesso aos dados, em documentos XML são determinados através das linguagens de consultas *XQuery* ou *XPath*. A linguagem *XPath* foi a primeira recomendação do W3C XML *Query Working Group* para consultar os dados em documentos XML. No entanto, essa linguagem mesmo satisfazendo grande parte das consultas, ela possui limitações, como no caso de retornar apenas fragmentos de um documento XML, sendo incapaz de retornar estruturas diferentes das existentes, além disso, a linguagem *XPath* não realiza junções entre os dados (GARCIA, MENEZES, BOSCARIOLI, 2008).

Para SANTOS (2007), a XPath (*XML Path Language*) utiliza expressões de caminhos para acessar qualquer item de dados no documento XML. Uma expressão *XPath* retorna uma coleção de nós de elementos que satisfazem certos padrões especificados na expressão. Sendo que os nós na expressão *XPath* são os nós da árvore de documentos que são os nomes de *tags* (elemento), ou atributos. Além disso, há dois separadores principais, que são utilizados durante a especificação de um caminho, nos quais são divididos em barra única (/) e barras duplas (//). Portanto, uma barra única antes de uma *tag* especifica que a *tag* deve aparecer como filho direito da *tag* anterior (pai); já as *tags* duplas indicam que a *tag* pode aparecer como um descendente da *tag* anterior de qualquer nível. As barras duplas são utilizadas quando não sabemos o caminho completo.

A linguagem XQUERY (*XML Query Language*) é uma tentativa da W3C de disponibilizar uma linguagem de consulta que fornece a mesma funcionalidade que a linguagem SQL apresentada em bancos de dados relacionais. O *XQuery* permite a especificação de consultas mais genéricas em um ou mais documentos XML. A forma típica de uma consulta *XQuery* é conhecida por expressão FLWR que representa as quatro cláusulas principais do *XQuery*, que são divididas em cláusulas *FOR, LET, WHERE e RETURN*, sendo que a cláusula *FOR* são as variáveis ligadas a nodos individuais; a cláusula *LET* são variáveis ligadas a coleção de nodos (elementos); a cláusula *WHERE* são as condições qualificadoras e a cláusula *RETURN* é a especificação do resultado da consulta (SANTOS, 2007).

A seguir é abordado o capítulo que fala sobre o processamento de consultas nos bancos de dados relacionais, Orientado a Objetos e XML.

4. PROCESSAMENTO DE CONSULTAS

O objetivo deste capítulo é mostrar como funciona cada passo das etapas do processamento de consultas em banco de dados Relacionais, XML e OO. Na seção 4.1 apresenta-se a definição e os passos envolvidos no processamento de consultas de uma forma geral. Nas seções seguintes é destacado o processamento de consultas dos modelos relacional e *MySQL*, XML e *Tamino* e do Sistema Gerenciador de Banco de Dados Orientado a objetos respectivamente, no qual é apresentado uma visão geral dos diferentes componentes da arquitetura de cada SGBD e a descrição dos componentes relacionados ao processamento de consultas.

4.1 Introdução

O Processador de consultas é um componente responsável por efetivar a extração de dados de um banco de dados. Dentre as várias atividades que este componente executa, a tradução de consultas tem um papel de destaque. Uma vez que as consultas expressas em linguagem de alto nível são traduzidas para outras formas de representação, de forma a permitir a avaliação e a otimização das consultas. Por exemplo, nos bancos de dados relacionais as consultas em SQL são traduzidas para expressões da álgebra relacional e, em seguida, são otimizadas para depois serem processadas pelo SGBD (SILBERSCHATZ, 2006).

O processamento de consulta em um SGBD é realizado quando o usuário submete uma consulta a um SGBD, interativamente ou através de uma aplicação, utilizando uma linguagem de consulta de alto nível.

Para Brayner e Lopes (2005), o processamento de uma consulta consiste nas etapas a seguir.

- **Análise (*Parsing*):** Nessa fase, a consulta recebida deve ser escrita em uma linguagem de alto nível, como o SQL. O Parser verifica se as relações e atributos utilizados são válidos para o esquema do banco de dados e também avalia se a consulta foi escrita conforme as regras gramaticais da linguagem adotada.

- **Tradução:** em seguida é construída uma árvore de análise (parse) para a consulta. Esta árvore é comumente traduzida para uma expressão equivalente da álgebra relacional, que é uma forma de representação interna utilizada em banco de dados relacionais.
- **Otimização:** a próxima fase consiste no processo de seleção de uma estratégia de execução que minimize o custo de execução da consulta.
- **Avaliação:** Nessa etapa a consulta é executada com base na estratégia escolhida pelo otimizador.

Para Ayres (2003), uma Máquina de Execução de Consultas (MEC) é responsável pela execução de um Plano de Execução de Consulta (PEC), que é fornecido pelo otimizador de consultas. Para tal, é considerado que uma MEC envolve:

- Um conjunto de algoritmos diferentes para cada operação, os quais o otimizador poderá escolher para gerar um PEC;
- Tuplas de dados, processadas por esses operadores e contendo uma estrutura baseada no modelo de dados;
- Estruturas de dados volumosas necessárias para armazenamento de tuplas de dados durante os seus processamentos;
- Uma interface comum aos operadores, a qual permite a sua extensibilidade;
- Uma estratégia de execução, no qual o otimizador se baseia para produzir PECs.

Na Figura 2, Navathe (2005) ilustra os passos envolvidos no processamento de uma consulta de alto nível.

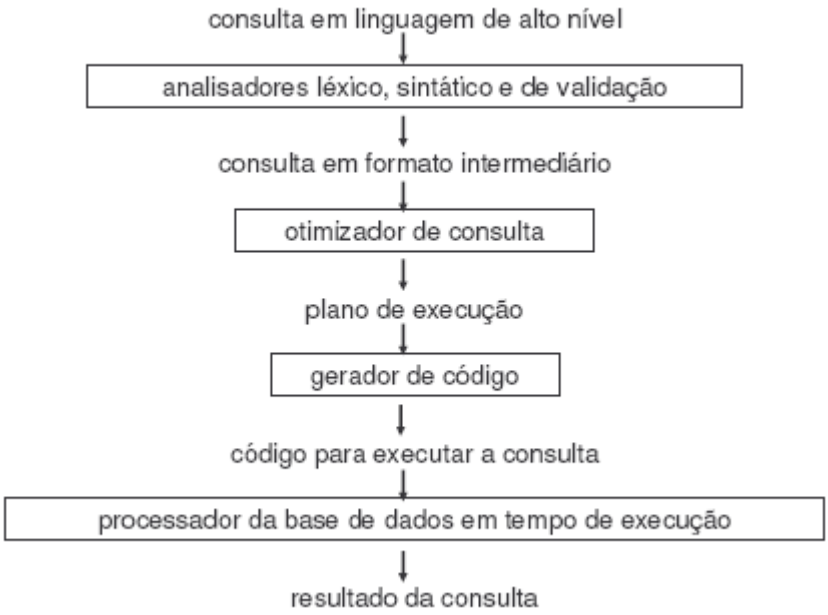

Figura 2:Etapas do processamento de consultas.
Fonte: Navathe (2005).

Quando uma consulta escrita em uma linguagem de alto nível (SQL, por exemplo) é submetida para execução no SGBD, inicialmente, essa consulta passa pela primeira etapa que é análise léxica e a análise sintática. A análise léxica identifica os itens léxicos da linguagem, tais como as palavras-chaves da SQL, nomes de atributos e nomes de relacionamento, no texto da consulta. Enquanto a análise sintática verifica a sintaxe da consulta para determinar se está formulada seguindo as regras da sintaxe (regras gramaticais) da linguagem de consulta. Nessa etapa também é verificado se todos os nomes de atributos e tabelas são válidos e corretos semanticamente.

Em seguida, a consulta é então convertida para uma forma interna. Em geral, as consultas são decompostas em blocos de consultas, que formam as unidades básicas que podem ser traduzidas em operadores algébricos. Para sistemas de bancos de dados relacionais, geralmente essa forma de representação é uma expressão da álgebra relacional (SILBERSCHATZ, 2006)**.**

Importante mencionar que uma consulta geralmente possui várias estratégias de execução e dentre elas deve ser escolhida a melhor estratégia para o processamento da consulta. Sendo que esse processo é conhecido como otimização de consultas. O otimizador de consulta é o componente responsável por essas tarefas.

Sendo assim, o otimizador gera diferentes planos de execução para uma determinada consulta contendo custos diferentes, que normalmente são baseados no número de acessos ao

disco e depois, seleciona o plano de execução que seja mais eficiente para processar a consulta e gera seu código a partir dos arquivos internos do banco de dados (SILBERSCHATZ, 2006).

Por fim, o componente referente ao processador da base de dados em tempo de execução executa o código gerado da consulta. O processo de execução da consulta é realizado no modo interpretado ou no modo compilado, com a finalidade de alcançar o resultado da consulta. Se ocorrer algum erro em tempo de execução, uma mensagem de erro é gerada pelo processador em tempo de execução do banco de dados (NAVATHE, 2005).

Nas seções seguintes serão evidenciados os passos de funcionamento envolvidos nas etapas do processamento de uma consulta, apontando a importância da otimização da consulta nos principais SGBDs disponíveis no mercado, como o Tamino, MySQL e SGBDOO.

4.2 Processamento de consultas Relacional – Banco de Dados *MySQL*

Na seção 4.1 foi descrito em linhas gerais o funcionamento do processamento de consultas do modelo relacional, baseado em uma arquitetura genérica de SGBD. Complementando o que foi apresentado, nesta seção será uma visão geral da arquitetura do SGBD *MySQL* e também o detalhamento dos componentes relacionados ao seu processamento de consultas.

Para Kruckenberg e Pipes (2005), a organização geral da arquitetura do servidor *MySQL* é uma camada, mas não particularmente hierárquica , e sim estrutural, sendo que os subsistemas dessa arquitetura são completamente independentes um do outro.

A figura 3 mostra uma descrição geral dos diferentes componentes da arquitetura do *MySQL*. Sendo que esses componentes são classificados como: *Management Services, Connection Pool, Functional SQL, Parser, Optimizer, Caches & Buffers, Pluggable Storage Engines, File System, Files & Logs*. Sendo que o *Parser, Optimizer e Caches & Buffers* são considerados os componentes principais do processamento de consultas do MySQL.

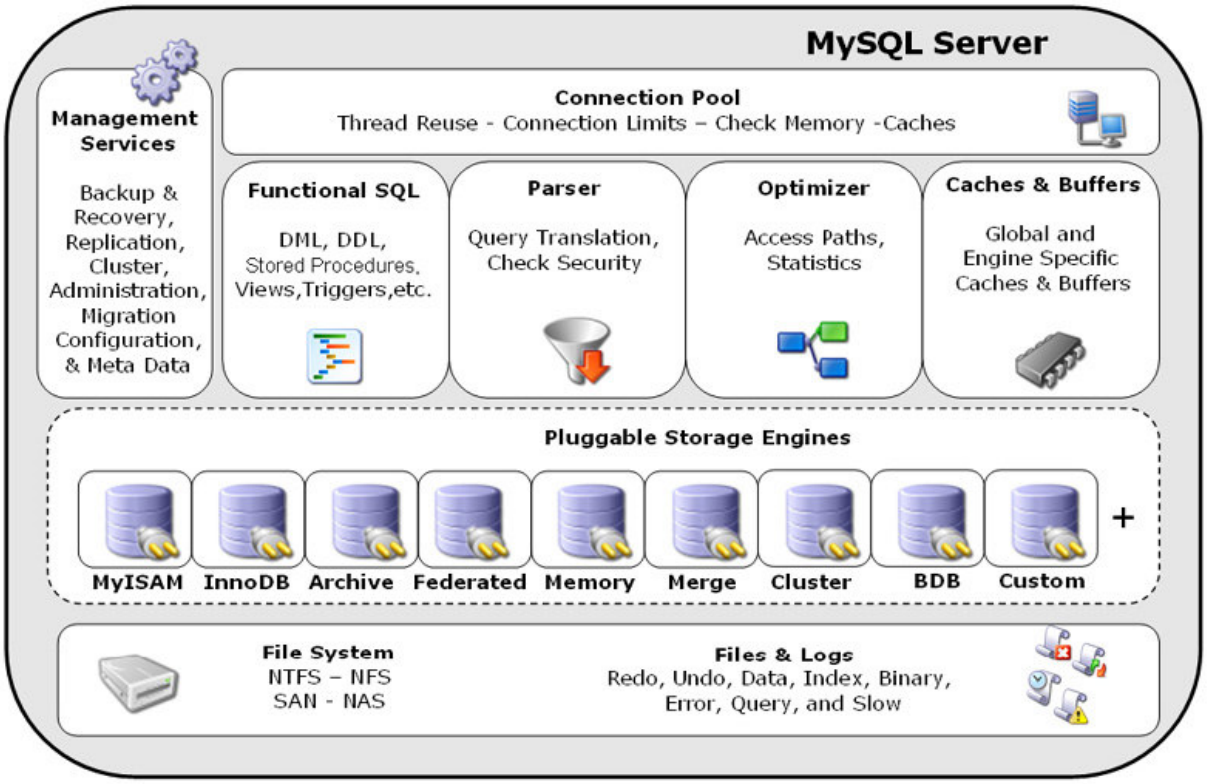

Figura 3: Visão geral da arquitetura do MySQL.
Fonte: Schumacher (2004)

A seguir serão detalhados os componentes principais da arquitetura do MySQL:

- **Serviços de Gerenciamento (*Management Services*):** No Serviço de Gerenciamento de banco de dados existe uma coleção de arquivos inter-relacionados e um conjunto de programas aplicativos para acessar e modificar esses arquivos. O mesmo mantém um controle sobre a redundância de dados, e também garante impor regras definidas pelo usuário para assegurar a integridade dos dados em forma de tabelas; possui um dicionário de dados centralizado para o armazenamento de informações referentes aos campos e manipulação de dados e fornece uma abordagem generalizada para a definição e processamento destes (ANDREW, 2009).
- **Pool de conexão (*Connection Pool*):** O *Pooling* de Conexões permite que o aplicativo da *Web* use uma conexão de um pool ou reservatório de conexões livres que não precisam ser restabelecidas. Depois que a conexão for criada e inserida em um pool, o aplicativo irá reutilizá-la sem que seja preciso executar o processo de conexão. Na prática quando um segmento (*thread*) precisa trabalhar com o *MySQL* por exemplo, ele solicita uma conexão de um pool e quando o segmento (*thread*) é terminado usando a conexão, ele retorna para o *pool*, para que possa ser usado por qualquer outro segmento (MATTHEWS, 2003).

- **Otimizador (*Optimizer*):** O processo de otimização seleciona o plano de avaliação de consulta mais eficiente dentre as muitas estratégias normalmente possíveis para o processamento de determinada consulta, especialmente se ela for complexa, isto significa que a otimização acontece quando o sistema tenta encontrar uma expressão que seja equivalente à expressão dada, porém, cuja execução seja mais eficiente (SILBERSCHATZ, 2006).
- **Analisador (*Parser*):** o processo do *parser* é verificar a sintaxe da consulta do usuário para ver se foi escrito de acordo com as regras gramaticais da linguagem e também verifica se os nomes de relações e atributos são válidos de acordo com o esquema do banco de dados.
- **SQL funcional (*Functional SQL*):** A SQL funcional é classificada em várias funções às quais incluem a DDL da SQL, DML da SQL, A SQL embutida e a dinâmica. A DDL da SQL fornece comandos para definir esquemas de relações, *views*, restrições de integridade, os comandos para excluir relações e modificar esquemas, logo a SQL embutida e dinâmica definem como as instruções SQL podem ser incorporadas dentro das linguagens de programação e a SQL também envolvem comandos para especificar o início e o fim das transações. A DML da SQL inclui uma linguagem de consulta baseada na álgebra relacional e no cálculo relacional de tupla e comandos para inserir, excluir e modificar tuplas na base de dados (SILBERSCHATZ, 2006).
- ***Caches & Buffers*:** A maior parte da memória alocada do MySQL é usada por vários *Buffers* e *Caches* internos. Estes *Buffers* se dividem em dois grandes grupos: Globais e por conexão. Como seu nome sugere, os globais são compartilhados entre todas as conexões (ou threads) no MySQL (WIKI, 2005).
- **Mecanismos de armazenamento *Pluggable* (*Pluggable Storage Engine*):** Esse mecanismo faz com que o programador de aplicação e o DBA interajam com o BD MySQL através do conector API e camadas de serviço que estão acima dos mecanismos de armazenamento. Se mudanças de aplicação trazem requisitos que exigem a mudança do mecanismo de armazenamento subjacente, ou que um ou mais mecanismos de armazenamentos adicionais são adicionados para suportar novas necessidades, nenhuma codificação significativa ou mudanças de processo são obrigados a fazer as coisas funcionarem (SCHUMACHER, 2004).

4.2.1 Análise de consulta, otimização e execução

Nesta seção serão abordados os processos de análise, otimização e execução de consultas, que são um dos focos principais do servidor do banco de dados MySQL. Os mesmos são responsáveis por receber os comandos solicitados pelos usuários e por transformar as declarações solicitadas em uma variedade de estruturas de dados que, em seguida, o servidor de banco de dados utilizará para determinar a melhor estratégia de execução das consultas.

4.2.2 Análise (*Parsing*)

Este processo consiste na análise das consultas, gerando uma árvore de sintaxe abstrata da consulta submetida. O analisador (*Parser*) do MySQL é baseado no programa chamado *Bison*. O programa Bison executa o *Parser* da consulta através de uma ferramenta chamada YACC (*Yet Another Compiler Compiler*). A ferramenta YACC recebe como entrada um fluxo de regras, as quais consistem em uma expressão regular e um trecho de código C projetado para manipular quaisquer correspondências feitas pela expressão regular. YACC então produz um executável a partir de uma *stream* de entrada e particiona-o em expressões regulares. Em seguida, ele executa o código C com cada expressão regular, na ordem correspondente.

Segundo Kruckenberg e Pipes (2005), Bison é um programa complexo que usa o compilador YACC para gerar um *parser* para um conjunto especifico de símbolos, que constituem o léxico da linguagem analisável.

O mecanismo de consulta do MySQL pode usar o analisador Bison gerado para fazer o trabalho difícil de fragmentar o comando de entrada. Este passo de análise não só padroniza a consulta em uma solicitação semelhante a uma árvore para tabelas e junções, mas ele também funciona como uma representação em código de que a solicitação precisa a fim de ser cumprida. Sendo que esta representação em código de uma consulta é denominada de estrutura Lex. Além disso, sua definição está disponível em / sql / sql_lex.h e cada objeto de thread do usuário (THD) tem uma variável de membro Lex, que armazena o estado da análise. Para Specia e Rino (2002), a estrutura Lex consiste em um conjunto de palavras ou expressões da linguagem, chamadas de itens léxicos, as quais são associadas à sua descrição, ou seja, a um conjunto de descrições sintáticas e semânticas, cujos valores fornecem as

informações necessárias para que tais palavras ou expressões sejam processadas pelo sistema. Além disso, cada item, juntamente com sua descrição, é chamado de entrada lexical.

À medida que a analise da consulta começa, a estrutura Lex completa, assim que é executado o processo de análise, a estrutura de Lex é preenchida com uma quantidade crescente de informações sobre os itens utilizados na consulta. A estrutura de Lex contém membros de variáveis para armazenar as listas de tabelas utilizadas pela consulta, campos utilizados na consulta, junções necessárias à consulta e assim por diante. Como o analisador opera sobre as declarações de consulta e determina quais itens é necessária para a consulta, a estrutura de Lex é atualizada para expressar os itens necessários (KRUCKENBERG E PIPES, 2005).

De fato, na conclusão da análise, a estrutura de Lex contém uma espécie de caminho de roteiro para chegar aos dados. Este roteiro inclui os vários objetos de interesse para a consulta. Algumas das variáveis de membro notável de Lex incluem o seguinte:

- **table_list e group_list:** são listas de tabelas usadas, nas cláusulas FROM e GROUP BY.
- **top_join_list:** é uma lista de tabelas para a junção de nível superior (top-level).
- **order_list:** é uma lista de tabelas na cláusula ORDER BY.
- **where and having:** são variáveis do tipo Item, que correspondem as cláusulas WHERE and HAVING.
- **select_limit e offset_limit:** são usados na cláusula LIMIT.

A fim de compreender corretamente o que está armazenado na estrutura de Lex, é necessário investigar as definições de classes e estruturas que estão definidas nos arquivos listados na tabela 1 a seguir.

Importante esses arquivos que representam as unidades de núcleo do mecanismo de execução de consulta SQL.

File	Contents
/sql/field.h and /sql/field.cc	Definition and implementation of the Field class
/sql/item.h and /sql/item.cc	Definition and implementation of the Item class
/sql/item_XXX.h and /sql/item_XXX.cc	Definition and implementation of the specialized Item_ classes used to represent various objects in database; for instance, Item_row and Item_subselect
/sql/sql_class.h and /sql/sql_class.cc	Definition and implementation of the various generic classes and THD

Tabela 1: Núcleo do Mecanismo de Execução de Consulta.
Fonte: Kruckenberg e Pipes (2005)

Os arquivos que são representados pelo o Item_XXX, implementam os vários componentes da linguagem SQL, como por exemplo, seus operadores, expressões, funções, linhas, campos e assim por diante.

Na sua origem, o analisador usa uma tabela de símbolos que correspondem às partes de uma consulta ou comando. Estas tabelas de símbolos podem ser encontradas em /sql/lex.h, /sql/lex_symbol.h, and /sql/lex_hash.h. Os símbolos são realmente apenas as palavras-chaves mantidas pelo o MySQL, incluindo padrão ANSI do SQL e todas as funções estendidas utilizadas em consultas do MySQL. Esses símbolos compõem o léxico do mecanismo de consulta; os símbolos são alfabeto do mecanismo de consulta de classificações.

Os arquivos /sql/lex * contêm as tabelas de símbolos que agem como tokens para o analisador fragmentar a entrada da instrução SQL em estruturas legíveis de máquina, que são então transferidos para os processos de otimização. Pode ser visto o analisador gerado pelo MySQL em / sql /sql _yacc.cc. O processo do analisador começa na linha 11676 do arquivo, onde a variável yyn é verificada e onde começa uma instrução *switch* gigantesco. A variável yyn representa o atual número de símbolo analisados. A seguir segue a lista de alguns arquivos que implementam a funcionalidade de análise na tabela 2**.**

File	Contents
/sql/lex.h	The base symbol table for parsing.
/sql/lex_symbol.h	Some more type definitions for the symbol table.
/sql/lex_hash.h	A mapping of symbols to functions.
/sql/sql_lex.h	The definition of the Lex class and other parsing structs.
/sql/sql_lex.cc	The implementation of the Lex class.
/sql/sql_yacc.h	Definitions used in the parser.
/sql/sql_yacc.cc	The Bison-generated parser implementation
/sql/sql_parse.cc	Ties in all the different pieces and parts of the parser, along with a huge library of functions used in the query parsing and execution stages.

Tabela 2: Análise e arquivos de implementação geral de Lex.
Fonte: Kruckenberg e Pipes,(2005)

A utilização prática de um SGBD se insere mais notadamente em armazenar informações por meio de consultas, sendo importante o sistema em atender às requisições no menor tempo possível, retornando os dados para o cliente que fez a consulta. De fato, em sistemas que geram um volume alto de requisições, se as mesmas não são executadas em curto espaço de tempo, não há possibilidade de se manter o desempenho do sistema.

Realmente para executar uma consulta SQL, o MySQL necessita percorrer várias fases até gerar o resultado para o usuário que a submeteu. A resolução desta consulta depende de vários fatores, como por exemplo, a utilização de índices, as operações *JOINS* e *sub-selects*, ou mesmo a estrutura dos dados.

4.2.3 Otimização

Grande parte da otimização de consultas provém da capacidade do subsistema, por "explicarem" as partes de uma consulta e encontrar a maneira mais eficiente de organizá-la e compreender como e em que ordem separar os conjuntos de dados e saber em que ordem é recuperada, mesclada ou filtrada.

De acordo com Silberschatz, Korth e Sudarshan (2006. p.383), "otimização de consulta é o processo de selecionar o plano de avaliação de consulta mais eficiente dentre as muitas estratégias normalmente possíveis para o processamento de determinada consulta, especialmente se esta for complexa, não esperamos que os usuários escrevam suas consultas de modo que possam ser processadas de forma mais eficiente".

Na tabela 3 é mostrada uma lista dos arquivos principais usados no sistema otimização.

File	Contents
/sql/sql_select.h	Definitions for classes and structs used in the SELECT statements, and thus, classes used in the optimization process
/sql/sql_select.cc	The implementation of the SELECT statement and optimization system
/sql/opt_range.h and /sql/opt_range.cc	The definition and implementation of range query optimization routines
/sql/opt_sum.cc	The implementation of aggregation optimization (MIN/MAX/GROUP BY)

Tabela 3: Arquivos usados no sistema de otimização.
Fonte: Kruckenberg e Pipes (2005)

Na maioria das vezes, a otimização de consultas SQL é necessária somente para instruções SELECT, por isso, é natural que a maioria do trabalho da otimização seja efetuado no /sql/sql_select.cc. Este arquivo usa o *structs* definidos no /sql/sql_select.h, juntamente com o arquivo de cabeçalho que contém as definições para algumas das classes amplamente utilizadas e *structs* no processo de otimização: *JOIN, JOIN_TAB* e *JOIN_CACHE*.

Importante realizar a otimização, já que a maior parte do trabalho de otimização é feita no método de membro *JOIN:: optimize( )*. Este complexo método torna maior o uso da

estrutura Lex disponível no thread do usuário (THD) e o caminho de roteiro é correspondente à solicitação SQL, que ela contém.

JOIN::OPTIMIZE() foca seu esforço em " Otimizar" partes da execução da consulta por redundante eliminação das condições ***WHERE*** e manipulando as listas ***FROM*** e ***JOIN*** das tabelas na ordem mais uniforme possível. Ele executa uma série de sub-rotinas que tentam otimizar todos e cada pedaço das clausulas *WHERE* e condições de junção.

4.2.4 Execução

Por fim, a última atividade é a execução. Uma vez que o caminho para a execução foi otimizado tanto quanto possível, os comandos SQL devem ser executados pela unidade de execução da declaração. A unidade de execução da instrução é a função responsável por manipular a execução do comando SQL apropriado. Por exemplo, a unidade de execução da instrução para os comandos SQL INSERT é *mysql_insert()*, que se encontra na /sql/sql_insert.cc. De maneira similar, a unidade de execução da instrução *SELECT* é mysql_select(), alojados em /sql/sql_select.cc. Estas bases de funções têm um ponteiro para um objeto THD como seu primeiro parâmetro. Este ponteiro é usado para enviar os pacotes de dados do resultado de volta ao cliente.

4.3 Processamento de consultas XML – Banco de Dados *Tamino*

O processamento de consultas é responsável por tornar transparente aos clientes não somente as operações sobre os dados, mas também a sua distribuição. Porém, no contexto XML a complexidade das linguagens de consulta são fatores diferenciais, pois requerem algoritmos mais direcionados e focados no processamento, em contrapartida ao modelo relacional, porque não há uma álgebra para o XML.

Em se tratando da otimização de consultas para esses tipos de SGBDs, os dados XML são armazenados em uma representação interna que preserva o conteúdo dos dados e que inclui informações especiais sobre a ordem dos documentos.

Como mencionado no capitulo anterior, um documento XML é composto de elementos de marcação (*tags*) e texto. O mesmo possui uma estrutura organizada hierarquicamente, aninhada e em forma de árvore. No quadro 7 é exemplificado um documento XML contendo dados sobre livros.

```
<? Xml version= "1.0" encoding="UTF-8
<livros>
<livro isbn="85-241-0590-9">
<titulo> Proj. de Banco de Dados </titulo>
<autor> Carlos A. Heuser </autor>
<editora> Sagra Luzzato </editora>
<edicao> 2 </edicao>
<serie numero="4"> Livros Didáticos </serie>
</livro>
<livro isbn="83-345-0336-2”>
<titulo> Sistemas de Banco de Dados Cliente-Servidor </titulo>
<autor> Rubens Nascimento Melo </autor>
<autor> Asterio Tanaka </autor>
<editora> Campus </editora>
</livro>
...
</livros>
```

Quadro 7: Exemplo de Dados XML.
Fonte: Ayres (2003)

Navathe (2005) considera um documento XML bem-formado quando sintaticamente está correto. Isso permite que seja processado por processadores genéricos que percorrem o documento e criam uma representação de árvore interna.

Ayres (2003) vai além da sintaxe correta. Um documento XML “bem formado” é quando esse obedece às convenções léxicas, como, por exemplo, *tags* iniciais associadas com *tags* finais. E, para verificar se um documento é bem formado faz-se uso de um Analisador (*Parser*) XML. O Analisador é responsável por verificar a integridade dos dados XML

O *Parser* ou analisador pode ser executado de duas formas: sem ou com validação. Um *Parser* sem validação verifica a sintaxe do documento, ou seja, verifica se o documento é bem-formado. No *Parser* com validação, além da verificação da sintaxe, os dados são comparados com uma DTD. Desta maneira, um *Parser* com validação é capaz de verificar se um documento é válido.

Um documento é dito “válido” se ele está de acordo com a sua DTD (*Document Type Definition* - Documento de definição de tipo) ou com o seu esquema XML (*XML Schema*), que define as regras de sintaxe do documento. O exemplo a seguir mostra a DTD que pode ser usada para validar o documento de livros apresentados no exemplo anterior.

```
<?xml version="1.0" encoding="UTF-8"?>
        <!ELEMENT   livros (livro+)>
        <!ELEMENT   livro  (titulo, autor+, editora, edicao?, serie?)>
        <!ATTLIST     livro isbn CDATA #REQUIRED>
                <!ELEMENT   titulo (#PCDATA)>
                <!ELEMENT   autor (#PCDATA)>
                <!ELEMENT   editora(#PCDATA)>
                <!ELEMENT   edicao (#PCDATA)>
                <!ELEMENT   serie (#PCDATA)>
<!ATTLIST     serie numero CDATA #REQUIRED>
```

Quadro 8: DTD para o documento de livros.
Fonte: Ayres (2003)

Para Silberschatz (2006), um *Document Type Definition* (DTD) é considerado como parte opcional de um documento XML. Pois a finalidade principal de um DTD é muito parecida com a de um esquema, que é restringir as informações e os tipos de informações presentes no documento. Além disso, a DTD não restringe os tipos no sentido dos tipos básicos, como inteiro ou string, ela só restringe o surgimento de subelementos e atributos dentro de um elemento. Portanto, a DTD é uma lista de regras que indica qual padrão de subelementos pode aparecer dentro de um elemento.

Na DTD do quadro 8, o operador + que está após o nome do elemento, como por exemplo (livro+), significa que este pode ser repetido uma ou mais vezes no documento, no qual este tipo de elemento é denominado de elemento multivalorado (repetitivo) obrigatório. O operador ? usado após o nome de elemento (edicao?, serie?) quer dizer que este pode ser repetido zero ou mais vezes, logo, esse tipo de elemento é caracterizado como um elemento de valor único (não repetitivo) opcional. Um elemento que aparece sem nenhum símbolo deve aparecer exatamente uma vez no documento, assim, esse tipo de elemento é definido como um elemento de valor único (não repetitivo) obrigatório.

Os atributos podem ser especificados como sendo do tipo CDATA, ID, IDREF ou IDREFS; o tipo CDATA especificado no quadro 8 mostra que o atributo em questão contém dados de caractere. Para Navathe (2005), os atributos precisam ter uma declaração de tipo e uma declaração padrão. A declaração padrão consiste em um valor-padrão para o atributo ou #REQUIRED (significa que um valor precisa ser especificado para o atributo em cada elemento). Neste exemplo, o elemento livro é definido para conter subelementos (titulo, autor+, editora, edicao? serie?). Observando que os elementos: titulo, autor, editora, edicao, serie, são todos declarados como sendo do tipo #PCDATA (indicando dados do tipo texto).

Segundo Ayres (2003) o acesso aos dados armazenados em documentos XML, a partir de uma aplicação, pode ser feito através de uma interface de programação (*API_ Application Programming Interface* – Interface de Programação de Aplicações). Para tal, será apresentado como funciona o processo de análise de documentos XML, através das respectivas interfaces padrão SAX e DOM:

- SAX (*Simple API for XML*): Tem a função de fornecer dados para aplicação através de eventos, disparados durante a análise do documento, e que são processados pela aplicação à medida que são analisados.
- DOM (*Document Object Model_Modelo de Objeto de Documentos*): Tem a função de disponibilizar os dados para a aplicação através de uma árvore, na memória, contendo todos os dados do documento.

Para Santos (2007) O DOM pode ser representado como um modelo para armazenar e manipular em memória documentos com estrutura hierárquica, independente da linguagem de programação. O Analisador DOM analisa um documento XML e constrói uma árvore que pode ser utilizada para percorrer os vários nós.

Navathe (2005) destaca que para a API DOM permitir que programas manipulem a representação de árvore resultante correspondente a um documento XML bem-formado, faz-se necessária a análise sintática do documento inteiro antes de usar a API, o que torna difícil o processamento de documentos grandes.

Neste contexto, a API SAX (*Simple API for XML*) vem para evitar a necessidade de analisar o documento por completo. A SAX realiza o processamento de documentos XML por meio da notificação do programa em execução sempre que uma *tag* de início ou de fim for encontrada. Isto é, permite o processamento dos documentos denominados XML streaming, em que o programa em execução pode processar as *tags* conforme encontrados. Portanto, os arquivos XML são lidos em partes, em que são identificáveis pelo *parser* por meio das *tags* de inicio e fim de cada elemento. E, como resultado, o documento é analisado passo a passo, ou seja, manipulado antes de ter sido percorrido por inteiro. Tornando mais fácil processar documentos grandes.

O processamento de consultas do banco de dados XML *Tamino* é dividido em três partes. A primeira realiza o armazenamento nativo do documento XML; a segunda permite a integração com outros bancos de dados relacionais e a terceira aceita a agregação de funções definidas pelos usuários (DUARTE, 2006).

Para armazenar um documento XML no *Tamino*, o documento deve ter um Esquema XML associado a ele. No momento do armazenamento, este esquema é convertido para o

esquema do *Tamino*, o mesmo utiliza o XML *Schema*, ou seja, se o esquema do documento for DTD, a conversão será feita para o XML *Schema* automaticamente (DUARTE, 2006).

Na Figura 4 é apresentada a arquitetura do *Tamino*, que possui os componentes *Tamino Manager, X-Machine, X-Node, SQL Engine, Extensions*. No *X-Machine* há os componentes da execução de consultas, como o *XML Parser, Object Processor, XML Query Interpreter, Object Composer, Utilities*. E o banco de dados possui os componentes *XML Store, Data Map, SQL Store*. A armazenagem e recuperação dos objetos são apresentadas pelos componentes do mecanismo *X-Machine*.

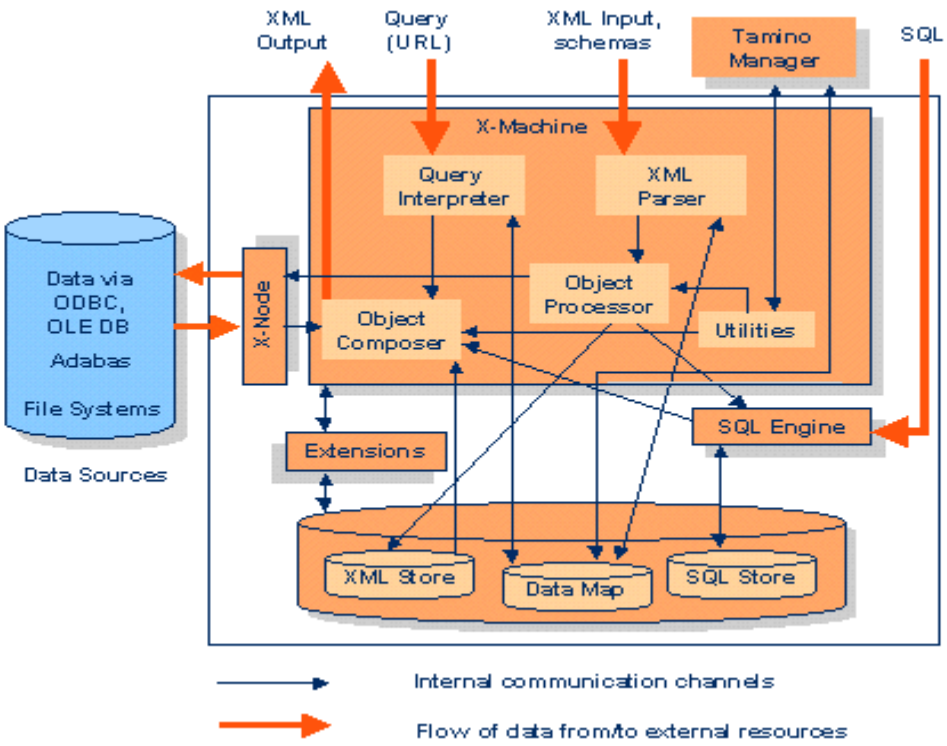

Figura.4: Arquitetura interna do Tamino .
Fonte: Campos (2005)

De acordo com Campos (2005), os componentes de execução da *X-Machine* e o mecanismo XML (*XML Store*) são responsáveis por processar, transformar e manter internamente os documentos XML.

A seguir é apresentado o funcionamento de cada etapa dos componentes mostrando passo a passo o processo de execução de consultas do *Tamino*.

- **Analisador XML (*XML Parser*):** Verifica se o documento XML está correto de acordo com o esquema e se ele é um documento bem formado. Os

documentos XML armazenados no *Tamino* são descritos pelos seus esquemas definidos no *Tamino Data Map*.

- **Processador de Objetos (*Object Processor*):** É usado ao armazenar dados no *Tamino*, onde as instâncias XML são validadas conforme o esquema lógico. O componente *Object Processor* contém as informações requeridas no esquema *Tamino* para armazenar os dados XML ou SQL. O esquema físico define como essas instâncias são armazenadas, como por exemplo, dados SQL são armazenados em tabelas e colunas SQL.
- **Interpretador de Consulta XML (*XML Query Interpreter*):** recebe uma requisição de consulta (*X_QUERY*), interpreta a linguagem e interage com o *object Composer* para que seja retornado um objeto XML de acordo com o esquema armazenado.
- **Compositor de Objetos (*Object Composer*):** É usado quando uma informação deve ser retornada. O mesmo constrói as informações do objeto e retorna um documento XML de acordo com as especificações da consulta; em um caso mais simples irá retornar um objeto armazenado internamente como XML. No entanto, podem ocorrer situações em que deve ser necessário se comunicar com componentes que façam a interface para fontes de dados não XML.
- ***Utilities*:** permite suporte a diretórios e leitura "orientada à árvore" de objetos XML.

A seguir será mostrada a figura que mostra o funcionamento da *X-Machine*:

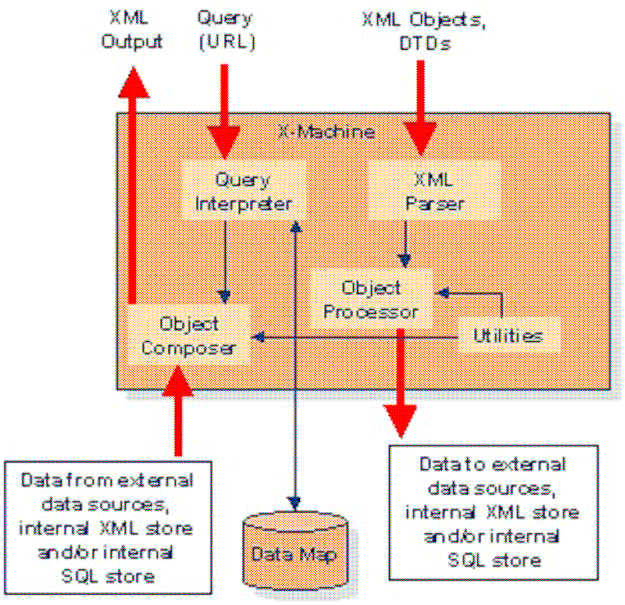

Figura. 5: Funcionamento da X-Machine.
Fonte: Campos (2005).

***X-Node*:** Disponibiliza padrões de interface para aplicações, base de dados ou arquivos externos. O mesmo permite acessar dados de outros tipos de banco de dados e arquivos de texto. O recurso mencionado permite a apresentação de dados corporativos para uma aplicação cliente como se estes dados estivessem armazenados em uma única base de dados.

A seguir é mostrado o esquema do *X-Node*

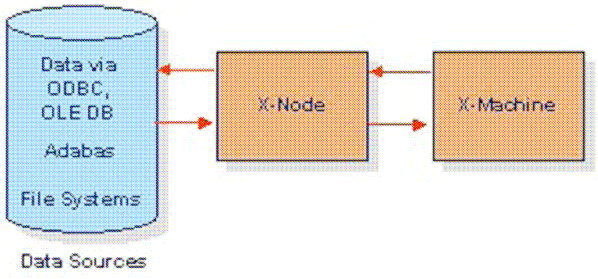

Figura.6: Esquema do X-node.
Fonte: Campos (2005)

Mapa de Dados (*Data-Map*): É o repositório de dados do *Tamino*, que contém os metadados, esquemas XML, mapeamentos para esquemas relacionais, folhas de estilos, etc. O mesmo possui os esquemas que contêm as regras de armazenamento e recuperação dos objetos XML. Também é responsável por toda a parte de indexação de objetos XML, seja dentro do *Tamino*, ou seja, mapeando os dados de estruturas externas.

Mecanismo SQL e armazenamento SQL (*SQL Engine e SQL Store*): São responsáveis por gerenciar o armazenamento interno de dados no modelo relacional permitindo a utilização do SQL. Sendo que o padrão SQL pode ser usado nas seguintes operações: internamente, a partir da *X-Machine* do *Tamino*; a partir de uma aplicação (via SQL embutido, ODBC, JDBC, OLE/DB); a partir do *Tamino Manager*.

4.4 Processamento de consultas em OO

Na visão de Mattoso, Ruberg, Baião e Victor (2009) as consultas em um SGBDOO são expressas em linguagens declarativas que têm como principal objetivo minimizar o conhecimento requerido do usuário sobre aspectos como a implementação interna da estrutura do objeto, existência de índices sobre atributos, e estratégias de processamento e Otimização existentes.

A figura 7 mostra os diferentes passos envolvidos no processamento de consultas do SGBDOO (MATTOSO *et al*, 2009). Para tal, esses passos são classificados como Otimização do Cálculo, Transformação Algébrica, Checagem de Tipos, Otimização Algébrica e Geração do Plano.

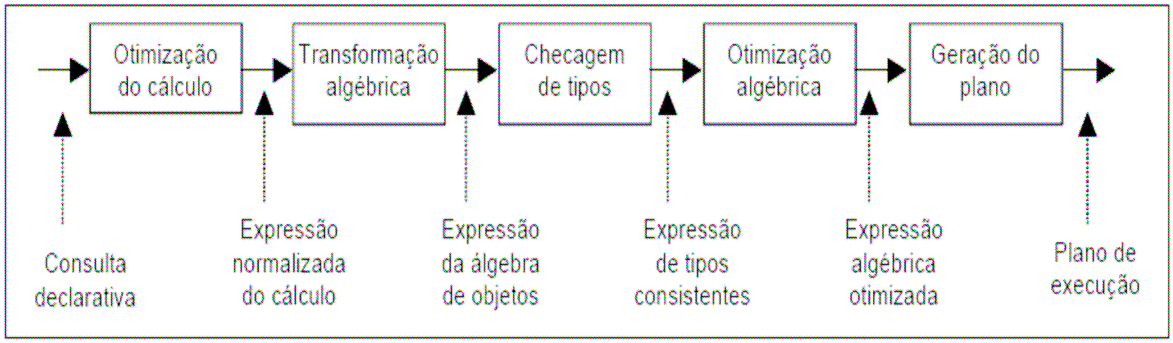

Figura7: Processamento de consultas do SGBDOO.
Fonte: Mattoso, Ruberg, Baião e Victor (2009).

O processamento de consultas no SGBDOO, desde a sua solicitação até o retorno do resultado, envolve vários passos, dentre eles serão detalhados a seguir.

- **Otimização do Cálculo:** Neste processo a expressão da consulta submetida é normalizada para eliminar predicados redundantes ou duplicados.
- **Transformação algébrica:** Em seguida, a expressão normalizada é transformada para uma representação equivalente na álgebra de objetos.
- **Checagem de tipos:** Após a normalização de predicados e da reescrita algébrica da consulta, a expressão algébrica é avaliada quanto à correção de

tipos. A complexidade desta avaliação incide no fato de os resultados intermediários, como conjuntos de objetos, poderem ser de tipos distintos.

- **Otimização algébrica:** O próximo passo é aplicar regras de transformação algébrica que substituem uma expressão algébrica corretamente tipada por outra expressão equivalente com desempenho melhor.
- **Geração do plano:** Depois de obter-se uma expressão equivalente com desempenho melhor, é então, encontrado um plano de execução físico da consulta a partir da expressão algébrica anteriormente otimizada, onde são levados em consideração aspectos físicos que não eram levados em fases anteriores, assim como as cardinalidade das coleções e índices.

Existem muitas maneiras de representar uma consulta OO em sua forma interna, porém a árvore de operadores é a mais utilizada. Desta forma, a consulta pode ser visualizada como uma árvore cujas folhas são expressas como extensões de classes ou coleções de objetos, e os nós internos são operadores, em que os operandos são seus nós filhos, que retornam outras coleções de objetos. Ao ser construída a árvore de representação de uma consulta, o seu processamento acontece a partir dos nós folha em direção à raiz da árvore. Outra maneira de representar uma consulta em sua forma interna é através da sua utilização na forma de grafos ao invés de árvores para representar a consulta.

No entanto, as etapas dos passos de um processamento de consultas no modelo OO que foram mencionadas anteriormente, que constituem a otimização do cálculo, transformação algébrica e checagem de tipos, podem ser vistas como etapas de preparação, ou pré-processamento, da consulta submetida ao processador de consultas. No entanto, os dois últimos passos, que são a otimização algébrica e geração do plano, são responsáveis pelo processamento efetivo da consulta, e podem ser concebidas como as fases mais importantes do processo, as quais serão detalhadas a seguir (MATTOSO *et al*, 2009).

4.4.1 Otimização Algébrica

No que diz respeito ao problema de otimização de consultas, o mesmo pode ser considerado como um problema genérico devido à necessidade da procura por uma solução ótima, dentro de um conjunto de soluções viáveis, conforme um algoritmo de busca. O algoritmo mencionado tem a responsabilidade de analisar as possíveis soluções, e escolher a melhor de acordo com uma função de custo. No entanto, a solução ótima que está sendo

retornada pelo algoritmo de busca é a expressão algébrica de menor custo que representa a consulta.

Portanto, o algoritmo de busca tem a função de definir a estratégia de varredura do espaço de busca, determinando a ordem de geração de novas soluções desde a descrição das regras de transformação.

4.4.2 Geração do Plano de Execução

Ao submeter uma consulta a um SGBD, a execução desta consulta deve ser realizada de forma eficiente pelo processador de consultas. O procedimento da otimização de consultas consiste em gerar o melhor plano de execução para uma dada consulta, o qual deve minimizar uma determinada função de custo.

É importante mencionar a distinção entre os SGBDOOs e os SGBDs relacionais. Nos SGBDs relacionais ocorrem os benefícios entre as correspondências diretas dos operadores da álgebra e as primitivas de acesso e armazenamento dos dados. No entanto, a seleção do melhor plano de execução está na escolha dos algoritmos mais eficientes que implementam os operadores relacionais e suas combinações (árvore de operadores). Na verdade, esta escolha é baseada nas estatísticas armazenadas no sistema. Entretanto, este mesmo benefício não pode ser obtido em SGBDOOs devido à diferença semântica entre a interface dos objetos (normalmente via métodos) e a estrutura física dos objetos, que está escondida do usuário através do encapsulamento.

Portanto, o encapsulamento e o armazenamento de métodos com objetos são um dos fatores de dificuldade no processamento de consulta. Por esse motivo, o acesso aos objetos é deixado por conta do gerenciador de objetos, que recebe solicitações do processador de consulta e devolve os objetos que satisfazem os predicados de consulta. Por outro lado, algumas implementações permitem que o otimizador acesse diretamente os objetos armazenados e o próprio otimizador tenha conhecimento da estrutura de armazenamento dos objetos, violando o encapsulamento. Esta é uma decisão de projeto do módulo de geração do plano (também denominado módulo de execução de consultas) que afeta a construção de todo o sistema.

Na verdade, o módulo de execução de consultas requer alguns algoritmos relacionados a coleções de objetos, como, *scan, indexed scan e collection matching*. O *scan* é uma classe de algoritmos simples que acessam seqüencialmente todos os objetos presentes em uma coleção. *Indexed scan* é uma classe de algoritmos que usam alguma estrutura de índice para

acessarem eficientemente os objetos de uma coleção. Deve ser possível definir índices sobre atributos simples ou até mesmo sobre expressões de caminho. O algoritmo *collection matching* funciona sobre múltiplas coleções de entrada para produzir uma coleção de saída. Vários algoritmos de junções relacionais são modificados para funcionarem sobre coleções de objetos e se enquadram nessa categoria.

4.4.3 Representação de Planos de Execução

A operação de junção é utilizada em consultas que expressam relacionamentos entre os dados armazenados, e é então, o mais custoso operador de avaliação para um processador de consulta. Portanto, o desempenho de consultas no modelo OO com expressões de caminho é influenciado pela escolha da estratégia e dos algoritmos utilizados para processamento das consultas. A próxima seção analisa os aspectos mais importantes sobre estratégias e algoritmos utilizados para o processamento de consultas que utilizam expressões de caminho.

4.4.4 Algoritmos de Consulta

Existem duas dimensões que são consideráveis, em relação ao processamento de expressões de caminho. Pois a direção em que a expressão de caminho é percorrida é relacionada a uma delas. Na verdade existem duas estratégias possíveis para esta dimensão, que são descendentes e ascendentes. A estratégia descendente indica que a expressão de caminho deve ser percorrida na ordem em que foi escrita pelo usuário, ou seja, da classe raiz para a classe folha, enquanto a estratégia ascendente indica que a expressão deve ser percorrida na ordem inversa (da classe folha para a classe raiz).

O operador e os algoritmos usados em cada uma dessas estratégias são considerados como mais uma das dimensões importantes referentes ao processamento de expressões de caminho. Na verdade, há duas classes de algoritmos para os operadores, que são classificados como algoritmos baseados em operadores n-arios e baseados em junções (operadores binários). No entanto, o *map* e a junção são os operadores da álgebra de objetos que são implementados por esses algoritmos, na qual o *map* materializa a referência ao objeto do relacionamento na expressão de caminho. O "*naive pointer chasing*" é o algoritmo mais conhecido para o operador n-ario. Portanto a junção resolve o relacionamento através de uma operação similar à junção do modelo relacional, só que é realizado sobre referências (ou

identificadores de objetos _OIDs) e não sobre valores (chaves). O *map* lida basicamente com referencias a instâncias individuais e a junção, no entanto opera sobre coleções de referências.

4.5 Conclusão

Este capítulo alcançou o seu objetivo, visto que foi possível apresentar um estudo sobre como funciona o processamento de consultas, realizados por alguns SGBDS disponíveis no mercado, mostrando sua importância na recuperação das informações.

O assunto desse capítulo teve grande aproveitamento, pois foi possível utilizá-lo para esclarecer certos aspectos como, por exemplo, o caso de certas distinções do processamento de consultas do modelo relacional em comparação aos outros SGBDS e outro aspecto importante foi mostrar a distinção do processamento de consultas do XML em relação ao modelo relacional, visto que no caso do XML não há uma álgebra específica para ele e foi mencionado que por esse motivo foram exigidos algoritmos mais direcionados e focados no seu processamento.

No caso do processamento de consultas do BD Tamino é permitido a conexão com BDs relacionais e as etapas do processamento de consulta do modelo OO foram definidas que a Otimização do cálculo, a transformação algébrica e a checagem de tipos podem ser vistas como etapas de preparação ou pré-processamento da consulta e os últimos passos que abordam a otimização algébrica e a geração do plano foram definidos como responsáveis pelo processamento efetivo da consulta e podem ser vistas como as fases mais importantes do processo.

5 CONCLUSÕES E TRABALHOS FUTUROS

O trabalho mostrou a importância do estudo do processamento de consultas, bem como o entendimento dos processos de manipulação, reutilização e otimização das consultas na base de dados. No capítulo 3 foi apresentado que para o processamento de uma consulta começar, o primeiro procedimento a ser tomado pelo sistema é traduzir a consulta para uma forma de representação interna; é importante destacar que uma linguagem de consulta de alto nível como, por exemplo, a SQL, não é apropriada para ser uma representação interna ao sistema. Sendo que uma representação interna mais adequada pode ser aquela baseada em uma expressão da álgebra relacional estendida, para bancos relacionais. Desta forma, é destacado que não se pode esperar que o usuário sempre escreva suas consultas de uma forma mais eficiente, por isso é responsabilidade do SGBD construir um plano de execução que minimize o custo e maximize o desempenho das consultas.

No capítulo 4 foi fornecido um melhor detalhamento do funcionamento do processador de consulta, através da apresentação das etapas envolvidas no processamento de consulta e ilustradas com a apresentação do funcionamento nos SGBDs da plataforma Relacional, XML e Orientado a Objetos.

No processamento de consultas do *MySQL* foram destacados os processos de análise, otimização e execução das consultas, e esses processos têm a responsabilidade de receber os comandos solicitados pelos usuários e também transformar as declarações solicitadas em uma variedade de estruturas de dados que posteriormente, o servidor de banco de dados utilizará para determinar a melhor estratégia de execução das consultas.

No processamento da consulta do XML, verifica-se que quando um documento XML é considerado bem-formado é porque ele está sintaticamente correto. Isto significa que o documento obedece às convenções léxicas como, por exemplo, *tags* iniciais são casadas com as *tags* finais, verificação essa realizada por analisador XML que ao percorrer o documento é criada uma representação de árvore interna para ser depois analisada.

Referente ao processamento de consultas do XML, através do SGBD *Tamino*, o mesmo é dividido em três partes, onde a primeira parte se refere ao uso do armazenamento nativo do documento XML; a segunda parte se refere à permissão de integração com outros bancos de dados relacionais; e, na terceira etapa é utilizada a agregação de funções definidas pelos usuários.

Por fim, no processamento de consultas do SGBDOO mostrou-se que desde a solicitação da consulta até o retorno do resultado são envolvidos vários passos, que são definidos como: otimização do cálculo, transformação Algébrica, checagem de tipos, otimização algébrica e geração do plano da consulta. Sendo que, os três primeiros passos são considerados como etapas de preparação ou pré-processamento e os dois últimos passos são considerados como as etapas mais importantes do processo de execução de consultas.

5.1 Sugestões de Trabalhos Futuros

A partir do estudo realizado neste trabalho, sugerem-se como possíveis trabalhos futuros:

- Realizar simulações de consultas nos SGBDs apresentados
- Fazer um comparativo de desempenho dos processadores de consultas
- Detalhar funcionamento do processamento de consultas nos bancos de dados não convencionais (Banco de dados multimídia, geográficos, etc.)

6. REFERÊNCIAS BIBLIOGRÁFICAS

ALMEIDA, M. Barcellos. *Uma introdução ao XML, sua utilização na internet e alguns conceitos complementares*, 2002. Disponível em: http://revista.ibict.br/index.php/ciinf/article/viewFile/140/120. Acesso em 29 de novembro de 2009

ANDREW, MySQLPoint. *Database Management Services*, 2009. Disponível em: http://translate.google.com.br/translate?hl=pt-BR&sl=en&u=http://mysqlpoint.com/database-management-services/&ei=8nohS9TKMs3JlAeMpICJCg&sa=X&oi=translate&ct=result&resnum=6&ved=0CCcQ7gEwBTgK&prev=/search%3Fq%3DManagement%2BServices%2BMySQL%26h1%3Dpt-BR%26sa%3DN%26start%3D10. Acesso em: 28 de novembro de 2009

ARBEGAUS, Aloisio. *Estudo do SGBD "Caché" com uma aplicação na reserva de vagas em eventos acadêmicos via web.* Blumenau, 2003. Disponível em: http://campeche.inf.furb.br/tccs/2003-II/2003-2aloisioarbegausvf.pdf. Acesso em: 20 de novembro de 2009

AYRES, Fausto V.M. *Cenários de Execução de Consultas*. Riode Janeiro, 2003. Disponível em: http://www2.dbd.puc-rio.br/pergamum/tesesabertas/9824821_03_cap_02.pdf. Acesso em: 23 de abril de 2009

BRAYNER, Ângelo; LOPES Aretusa M. A. *Operadores de Junção baseados em mecanismos de Hash para o processamento de consultas em banco de dados.* Fortaleza, 2005. Disponível em: http://www.unifor.br/notitia/file/1055.PDF. Acesso em: 12 de outubro 2008

CAMPOS, L. *Servidor XML Nativo, Gerenciando Dados Semi-estruturados,* 2005. Disponível em: http://www.lbasantarem.com.br/lba/semanadeinformatica2005/downloads/palestras/servidores_xml.pdf. Acesso em: 15 de outubro de 2009

CHIAVENATO, Idalberto. *Administração: teoria, processo e prática.* São Paulo: McGraw-Hill do Brasil, 1985

DUARTE, W. Silva. *Estado da arte em armazenamento de dados XML em banco de dados.* Jaguariúna, 2006. Disponível em: http://bibdig.poliseducacional.com.br/document/?view=102. Acesso em: 20 de setembro de 2009

DOBOIS, Paul. *Otimização de Consultas no MySQL em SQL magazine _Clustering no SQL Server,* p. 40, 25ª Ed Ano 3.

ELMASRI, Ramez; NAVATHE, Shamkant B. *Sistema de Banco de Dados*. Revisor técnico Luiz Ricardo de Figueiredo, 4ª ed. São Paulo: Pearson Addison Wesley, 2005.

GARCIA, M. Salvador; MENEZES, Anderson Luiz; BOSCARIOLI, Clovis. *Um Estudo sobre Gerenciamento de Dados XML em SGBDs Objeto-Relacionais e em SGBDs com suporte nativo a XML*, 2008. Disponível em: http://www.conged.pr.gov.br/arquivos/File/CONGED_Artigos/S2A1.pdf. Acesso em: 29 de novembro de 2009

GONÇALVES, Glaura Inácio; PESENTE, Graziela Rolim; LIMA, Rafael. *Banco de dados Orientado a Objetos*, 2008. Disponível em: http://www.inf.pucrs.br/~arruda/Downloads/Artigos/artigo05/index.htm. Acesso em: 27 de novembro de 2008.

GOODRUM, Glen. *Influenciando o Otimizador de consultas Oracle baseado em custos em SQL Magazine_Otimização_melhore o desempenho de suas aplicações Oracle e SQL Server,* p. 30, 18ª Ed, Ano 2.

GRAVES, Mark. *Projeto de Banco de Dados com XML;* tradução Aldair José Coelho Corrêa da Silva; revisão técnica Marcos Jorge. ___1. Ed. São Paulo: Pearson Education do Brasil, 2003.

KORTH, Henry F; SILBERSCHATZ**,** Abraham. *Sistema de banco de dados.* Tradução de Daniel Vieira, 5ª ed. - 2ª reimpressão. RJ: Elsevier, 2006.

KRUCKENBERG Michael; PIPES Jay. *Pro MySQL*, 2005. Disponível em: http://books.google.com.br/books?id=FAbmW1WdUWkC&dq=Pro+MySQL&printsec=frontcover&source=bn&hl=pt-BR&ei=92MlS5d8xtKUB8a7yP8J&sa=X&oi=book_result&ct=result&resnum=4&ved=0CB0Q6AEwAw#v=onepage&q=&f=false. Acesso em: 15 maio de 2008

LIMA, Cláudio. *Uma Ferramenta para Conversão de Esquemas Conceituais EER para Esquemas Lógicos XML.* Florianópolis_SC, 2008. Disponível em: http://projetos.inf.ufsc.br/arquivos_projetos/projeto_737/MONOGRAFIA_CLAUDIO_FINAL.pdf. Acesso em: 25 de novembro de 2009

MATTHEWS, M. *Connection pooling with MySQL Connector / J, 2003.* Disponível em: http://dev.mysql.com/tech-resources/articles/connection_pooling_with_connectorj.html. Acesso em: 24 de novembro de 2009

MATTOSO, Marta; RUBERG, Grabriela; BAIÃO, Fernanda; VICTOR André. Coordenação dos programas de pós-graduação de engenharia. *Processamento de Consultas Orientada a Objetos.* Universidade Federal do RJ, 2009. Disponível em: http://www.cos.ufrj.br/~gruberg/es54701.pdf. Acesso em: 11 de setembro de 2009

MELLO, Ronaldo dos Santos. *Bancosde dados XML*, 2008. Disponível em: http://www.inf.ufsc.br/~ronaldo/bdnc/10-bdXML.ppt#418,25. Modelagem Física – Revisão. Acesso em: 27 de novembro de 2009

PIALARISSI, Rafael, Londrina. *Estudo Comparativo de SGBDS XML,* 2005. Disponível em: http://www2.dc.uel.br/nourau/document/?view=221. Acesso em: 25 de novembro de 2009

RIBEIRO, Fernanda Potasso. *Um Gerador Automático de Páginas na Internet para professores da Universidade Federal de Lavras usando XML.* Lavras Minas Gerais, 2005, Disponível em: http://www.bcc.ufla.br/monografias/2005/Um_gerador_automatico_de_paginas_na_internet_para_professores_da_Universidade_Federal_de_Lavras_usando_XML.pdf . Acesso em: 12 de outubro de 2009

SANCHES, Andre Rodrigo. *Disciplina: Fundamentos de Armazenamento e Manipulação de Dados.* Faculdade de Sumaré, 2005. Disponível em: http://www.ime.usp.br/~andrers/aulas/bd2005-1/aula5.html. Acesso em: 23 de novembro de 2008

SANTOS, Miriam Oliveira. *Armazenamento e Recuperação de documentos XML Heterogêneos: Aplicando técnicas de KDD para apoiar o projeto físico em SGBDs XML Nativos.* Rio de Janeiro, 2007. Disponível em: http://recreio.de9.ime.eb.br/dissertacoes/2007-Miriam_Santos.pdf. Acesso em: 20 de setembro de 2009

SCHUMACHER, R. *MySQL 5.0's Pluggable Storage Engine Architecture, Parte 1: Overview*. MySQL AB, 2004. Disponível em: http://translate.google.com.br/translate?hl=pt-BR&sl=en&tl=pt&u=http%3A%2F%2Fdev.mysql.com%2Ftech-resources%2Farticles%2Fmysql_5.0_psea1.html. Acesso em: 25 de novembro de 2009

SETZER, V.W. *Bancos de Dados: Aprenda o que são, melhore seus conhecimentos, construa os seus*. 1ª Ed. SP: Edgard Blücher, 2005.

SILVA, Fernando. *Protótipo de um Sistema Gerenciador de Banco de Dados Orientado a objetos.* Universidade Regional de Blumenau, 2001. Disponível em: http://campeche.inf.furb.br/tccs/2001-I/2001-1fernandodasilvavf.pdf. Acesso em: 22 de novembro de 2009

SPECIA, Lucia; RINO, Lucia H. Machado. *O desenvolvimento de um léxico para a geração de estruturas conceituais UNL*, 2002. Disponível em:

http://www2.dc.ufscar.br/~lucia/TechRep/NILC-TR-0214-SpeciaRino.pdf. Acesso em: 30 de novembro de 2009

WIKI. *MySQL Tuning Tips (MySQL 5.0)*, 2005. Disponível em: http://www.performancewiki.com/mysql-tuning.html. Acesso em: 20 de novembro de 2009