1
Introdução

1.1.
Motivação

A adoção de Sistemas de Gerência de Banco de Dados (SGBDs) Relacionais na gerência de banco de dados é uma realidade na indústria. O sucesso desses sistemas se deu, em parte, pela eficiência no processamento de consultas que tem sido largamente estudado pela comunidade de banco de dados (Yu&Meng, 1998; Kossmann, 2000). As etapas do processamento de consultas compreendem: Análise (tradução e validação), Otimização e Execução de consultas submetidas pela aplicação. A Figura 1 mostra essas etapas, desde a submissão da consulta pela aplicação até a sua execução, produzindo os resultados.

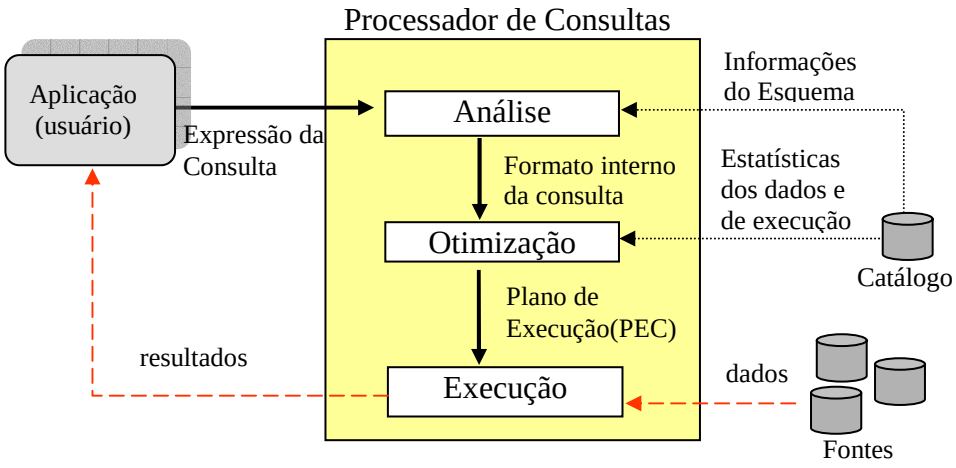

Figura 1 - Processamento de Consultas em SGBDs

A eficiência no processamento de consultas deve-se aos dois principais componentes do processador de consultas que são:

- Otimizador de Consultas, que produz Planos de Execução de Consultas (PECs) otimizados, e
- Máquina de Execução de Consultas (MEC) que recebe e executa um PEC segundo um modelo de execução e de dados

Em particular, este trabalho investiga as funcionalidades de Máquinas de Execução de Consultas (MECs), descritas a seguir.

Uma MEC é responsável pelo processamento físico dos dados requisitados pela consulta. Durante a execução de uma consulta, tuplas[1] de dados fluem através dos operadores de um Plano de Execução de Consultas (PEC), da seguinte forma: as tuplas são criadas pelos operadores mais próximos das fontes de dados e são processadas pelos demais operadores do PEC, até chegarem ao operador mais próximo do resultado da consulta, podendo ainda serem descartadas por um desses operadores. O fluxo de dados segue o modelo produtor-consumidor, ou seja, os dados fluem de um operador produtor para um operador consumidor. Este fluxo deve estar em sintonia com as técnicas de execução próprias do tipo de aplicação na qual se insere. Estas técnicas visam controlar o fluxo de dados que se deseja adotar num PEC (ou em parte dele). Exemplos de técnicas tradicionais encontradas na literatura incluem: *Exchange* (Graefe, 1996) e *Send-Receive* (Kossmann, 2000).

Um "modelo de execução de consultas" é visto aqui como uma combinação de técnicas de execução em um PEC para produzir um fluxo de dados que atenda aos requisitos de uma classe de aplicações de consultas que generalizamos como um "cenário de execução de consultas". Por exemplo, o cenário tradicional é representado pelas consultas SQL, dos SGBDs comerciais, e o modelo de execução deste cenário utiliza técnicas tradicionais para distribuição e paralelismo (Garcia-Molina et al., 2000).

Com o surgimento do modelo computacional baseado na *Internet* e do modelo de dados semi-estruturados (Abiteboul et al., 1999), novos cenários de execução de consultas foram produzidos, tais como:

Cenário 1: Consultas que precisam lidar com variações nas condições de execução, tais como: tempo de acesso às fontes de dados, seletividade e falta de memória (Amsaleg et al., 1996; Ives et al., 1999; Avnur&Hellerstein, 2000; Bouganim et al., 2000, 2001);

1 O termo "tupla" será utilizado neste trabalho como uma unidade de dados genérica para MECs de diferentes modelos de dados.

Cenário 2: Consultas do tipo *publish/subscribe* que precisam ser reavaliadas de tempos em tempos, de acordo com a necessidade do usuário (Chen et al., 2000; Chandrasekaran et al.,2003);

Cenário 3: Consultas sobre *streams* de dados, nas quais os dados precisam ser processados à medida que chegam (Dobra et al., 2002; Madden et al., 2002; Josifovski et al., 2002);

Cenário 4: Consultas sobre dados XML a partir de SGBDs XML nativos ou SGBDs relacionais (Scheffner&Freytag, 2002; Chamberlin et al., 2003);

Cenário 5: Consultas de integração de dados entre fontes de dados relacional e XML (Fernandez et al., 2002; LeSelect, 2003; Manolescu et al., 2000).

Estes cenários suscitaram novos modelos de dados (por exemplo: o modelo XML) e novos modelos de execução (por exemplo, modelo adaptável e modelo contínuo) ambos não suportados pelas MECs tradicionais. Além disso, esses cenários sugerem algumas das particularidades que novas aplicações requererão de um sistema de processamento de consultas necessitando-se de novas adaptações das MECs existentes.

A comunidade acadêmica de banco de dados está fortemente direcionada na procura por alternativas de soluções para estes cenários. As soluções encontradas apresentam implementações específicas, ou seja, foram projetadas para um modelo de dados e um modelo de execução específicos. Dessa forma, os PECs submetidos às MECs são sempre executados segundo um único modelo de execução e de dados, implícito na própria arquitetura da MEC, não permitindo a sua extensibilidade para outros modelos. Além disso, não encontramos na literatura uma arquitetura padronizada nem uma metodologia para o desenvolvimento de MECs que suportem estes e novos modelos. Diante disso, considera-se as seguintes abordagens para construção de uma MEC:

- A adaptação da MEC tradicional
- A completa construção de uma nova MEC.

Considera-se como um obstáculo para a primeira abordagem, o fato dos SGBD tradicionais apresentarem arquiteturas monolíticas reduzindo sua

capacidade de adaptação aos novos cenários de processamento de consultas. Além disso, a extensão das MECs tradicionais pode requerer um esforço considerável de desenvolvimento, como o visto recentemente nos SGBDs comerciais para o suporte ao modelo de dados XML (Shoning, 2001). A segunda abordagem traz algumas desvantagens, tais como a falta de uma estrutura inicial, o que implica em custos, além da dificuldade de se estabelecer um novo produto no mercado.

Tudo isso nos leva a crer que o esforço de construção de MECs aumentará à medida que surjam novos modelos de execução e de dados, resultando na necessidade de se projetar MECs de fácil reuso e adaptação frente aos novos cenários.

1.2. Objetivo da Tese

O objetivo desta tese consiste no desenvolvimento de uma MEC extensível que suporte diferentes modelos de execução e de dados, de forma ortogonal, e que permita o reuso de uma infra-estrutura básica de uma MEC. Como base para isto, a MEC proposta é fundamentada em abstrações sobre o PEC tradicional que refletem o modelo de execução a ser instanciado.

Neste contexto, surgem algumas questões importantes no projeto da MEC extensível, tais como:

1. Como a MEC suportará diferentes modelos de execução e de dados?
2. Qual a relação entre modelo de dados e modelo de execução?
3. Que arquitetura proverá extensibilidade à MEC?

Visando responder estas questões, este trabalho tem como objetivos específicos:

- investigar as arquiteturas das MECs e dos modelos de execução e de dados que surgem a partir dos cenários.
- investigar alguns tópicos de pesquisa relevantes para este trabalho, tais como: SGBDs distribuídos e paralelos, sistemas de integração de dados, SGBDs XML, SGBDs extensíveis e *frameworks* de *software*.

- Identificar aspectos comuns encontrados no suporte à execução dos cenários pesquisados de modo a abstrair requisitos extensíveis de uma MEC.
- Propor uma MEC extensível com características gerais que permitam a sua adaptação para diferentes modelos de execução e de dados.

1.3. Visão Geral da Solução Proposta

A solução proposta utiliza a técnica de *framework* de *software* (Fayad, 1999) para a construção da MEC extensível. Como conseqüência, desenvolveu-se o QEEF *(Query Execution Engine Framework)* (Ayres et al., 2003) que pode ser instanciado, de forma ortogonal, para um determinado modelo de execução e de dados, bem como para mais de um modelo de dados.

A escolha da técnica de *framework* deve-se principalmente ao fato dela ter sido bastante utilizada na construção de sistemas flexíveis e que podem ser adaptados com maior rapidez e facilidade para atender aos requisitos de novas aplicações. Espera-se com isso que a flexibilidade oferecida pelo QEEF reduza o esforço de construção de novas MECs.

O desenvolvimento do QEEF considera o balanceamento dos seguintes itens:

- Facilidade de implementação: oferecer uma infra-estrutura básica (*building blocks*) para construção de MECs;
- Desempenho: implementar eficientemente as técnicas de execução;
- Extensibilidade: permitir adição de novas funcionalidades;
- Portabilidade: permitir a adaptação em ambientes de execução distintos.

Quando instanciado, o QEEF se comporta como um componente de *software* conectado a outros componentes, tais como: um processador de consultas de um SGBDs ou uma aplicação *ad hoc*. Para permitir uma maior interoperabilidade, os PECs, submetidos ao QEEF, devem ser especificados na linguagem XML e são denominados neste trabalho de PEC-XML.

A seguir, descreve-se com mais detalhes a solução proposta baseando-se nas questões postas na seção anterior (Seção 1.2) como guias para esta descrição.

1.3.1.
Suporte a Diferentes Modelos de Execução

O primeiro passo em direção à solução proposta foi o de reunir um conjunto de conceitos relativos à execução de consultas em uma MEC os quais são descritos a seguir.

Uma "Máquina de Execução de Consultas" (MEC) executa "Planos de Execução de Consultas" (PEC) contendo um conjunto de operadores algébricos e de controle relacionados na forma produtor-consumidor. As "tuplas de dados" são unidades de dados processadas pelos operadores do PEC e possuem uma estrutura de dados baseada no modelo de dados subjacente. Os "operadores algébricos" implementam a semântica expressa na consulta baseados numa álgebra e os "operadores de controle" implementam a comunicação entre os operadores algébricos, sem a manipulação de dados, exercendo o papel de meta-operador.

Neste trabalho, denomina-se de "característica de execução", um tipo de técnica de controle do fluxo de dados que se deseja adotar num PEC (ou em parte dele). A Tabela 1 reúne as características de execução encontradas a partir da análise de alguns cenários apresentados no início deste capítulo, incluindo o cenário tradicional, onde cada característica pode apresentar diferentes estados. Algumas dessas características foram inicialmente introduzidas por Goetz Graefe em (Graefe, 1993).

Características de Execução	**Estados**
Sincronismo	*wait*
	wait-all
	no-wait
Distribuição	*remote*
	local
Paralelismo	*Inter-operator*
	Intra-opeerator
Fluxo de Dados	*fixed*
	adaptive
Fluxo de Controle	*demand-driven*
	data-driven
Tempo de Resposta	*first-tuple*
	last-tuple

Tabela 1 - Características de Execução de Consultas e seus Estados

Considera-se que cada estado de uma característica de execução é implementado através de um componente da MEC denominado de "módulo de

execução de consultas", o qual utiliza uma determinada estratégia de implementação para inserir operadores de controle entre os operadores algébricos de um PEC e para expressar a característica desejada.. Exemplos de módulos encontrados na literatura incluem: *Exchange* (Graefe, 1996) e *Send-Receive* (Kossmann, 2000) e *Eddies* (Avnur&Hellerstein, 2000) que implementam, respectivamente, os estados: *intra-operator*, *remote* e *adaptive*.

Finalmente, denomina-se de "modelo de execução de consultas" uma combinação de módulos de execução para produzir um fluxo de dados que atende aos requisitos de uma classe de aplicações de consultas que generalizamos como um "cenário de execução de consultas". Por exemplo, no cenário tradicional, representado pelos SGBDs comerciais, o modelo de execução tradicional das MECs, combina os seguintes módulos: *Demand-driven*, *Inter-operator* e *Remote*.

Para se obter uma visão geral deste processo de execução de consultas, relacionamos esses conceitos (ver Tabela 2) em níveis de abstração (Físico e Lógico), onde os conceitos físicos implementam os conceitos lógicos.

Conceito Físico	**Conceito Lógico**
MEC	Cenário de aplicações
PEC	Modelo de execução
Modulo	Característica de execução
Operador de controle	Tipo de comunicação
Operador algébrico	Álgebra (modelo de dados)
Tupla	Estrutura de dados (modelo de dados)

Tabela 2 - Conceitos de execução de consultas.

Tradicionalmente, uma MEC não permite a utilização de diferentes módulos de execução num PEC, pois é construída para suportar um modelo de execução de consultas específico, presente num certo cenário. Neste contexto, os PECs submetidos à MEC contêm operadores (algébricos e de controle) e são executados sempre utilizando um único modelo de execução que, em geral, fica implícito e codificado na sua implementação. Por exemplo, a MEC tradicional executa PECs baseando-se no modelo tradicional. A abordagem utilizada nesta tese é utilizar módulos como operadores de mais alto nível que, em geral, insere operadores de controle entre os operadores algébricos de um PEC para atender a uma determinada característica de execução.

O QEEF fundamenta-se nesses conceitos de execução e, em particular, no fato de um PEC combinar módulos de execução, para permitir o suporte a

diferentes modelos de execução. Neste caso, o usuário deve especificar um PEC contendo os módulos de execução associados ao plano algébrico da consulta. Para permitir esta especificação, propomos um Meta-Modelo de Execução de Consultas denominado de QUEM *(QUery Execution Meta-model).* Este meta-modelo definirá as regras de combinação dos módulos existentes no QEEF e de novos módulos que possam surgir. Os PECs especificados segundo o QUEM são denominados de meta-PECs e são pré-processados pela meta-MEC do QEEF resultando em PECs finais que, por sua vez, são executados pela MEC instanciada do QEEF, gerando o resultado da consulta.

1.3.2. Suporte a Diferentes Modelos de Dados

O QEEF pode ser instanciado para diferentes modelos de dados de forma ortogonal ao modelo de execução. Isso possibilita tanto a instanciação de uma MEC baseada num modelo de dados específico quanto uma baseada em mais de um modelo de dados. Para se obter tal flexibilidade, instancia-se para cada modelo de dados, seus operadores algébricos e a estrutura da sua tupla de dados. Além disso, para o suporte a execução de consultas com mais de um modelo de dados deve-se especificar operadores de conversão (inter-algébricos) de dados entre esses modelos. Por exemplo, um cenário de aplicações de integração de dados nos modelos Relacional e XML requer a instanciação do QEEF para ambos os modelos. A MEC instanciada será capaz de executar fragmentos de PECs em ambos os modelos. Para isso, deve-se especificar os operadores nas álgebras Relacional e XML, a estrutura da tupla Relacional e da tupla XML, e os operadores de conversão de dados, do modelo Relacional para o modelo XML e vice-versa.

1.3.3. O *Framework* QEEF

A seguir descreve-se uma visão geral do framework QEEF enfocando a instanciação, a extensibilidade, os estudo de casos e a implementação.

Instanciação

Um *framework* é uma arquitetura semi-completa que deve ser instanciada para um determinado domínio de aplicação (Fayad et al., 1999). O framework QEEF deve ser instanciado para um determinado modelo de execução e de dados, presentes num cenário de execução de consultas, resultando numa MEC específica. Para isso, faz-se necessária a instanciação dos seguintes pontos de adaptação (*hot-spots*):

- **tupla de dados:** instanciada de acordo com o modelo de dados desejado (exemplos: relacional, XML)
- **fonte de dados:** instanciada de acordo com o modelo de dados e com o tipo da fonte (exemplos: arquivo, SGBD, serviço *web*);
- **operador algébrico:** instanciado de acordo com a semântica do modelo de dados;
- **operador inter-algébrico:** instanciado para a conversão entre modelos de dados;
- **módulo de execução:** instanciado para um modelo de execução presente no cenário escolhido;
- **operador de controle:** instanciado de acordo com um módulo de execução (exemplos: leitor, coletor, distribuidor);
- **estrutura de dados:** instanciada a partir das necessidades de armazenamento de tuplas pelos operadores;

Os pontos fixos (*frozen-spots*) do QEEF consistem na gerência e no relacionamento destes pontos de adaptação.

Extensibilidade

O QEEF permite o reuso e a extensão para modelos de execução e de dados da seguinte forma:

- O reuso se dá à medida que uma instanciação do QEEF faz uso de uma infra-estrutura básica de uma MEC que inclui operadores de controle, estruturas de dados volumosas, interface dos operadores, interface da

MEC e pré-processamento de meta-PECs, além da gerência dessa infra-estrutura.

- A extensão para novos modelos de execução se dá na especificação de módulos de execução (e dos operadores de controle necessários), no meta-modelo QUEM e nas suas implementações no QEEF.
- A extensão para novos modelos de dados se dá através da instanciação de operadores algébricos, de uma tupla de dados e das fontes de dados.

Para permitir o uso do QEEF como componente de *software* em outros sistemas, os PECs são especificados através da linguagem XML, permitindo maior interoperabilidade do QEEF em relação a outros componentes que este venha a se relacionar.

Estudo de Casos

Para a validação do *framework* QEEF foram instanciadas as seguintes MECs:

1. **RQEE *(Relational Query Execution Engine)*** para suporte ao modelo de execução tradicional e ao modelo de dados relacional. Neste caso, a MEC é um componente do processador de consultas do SGBD TINO (descrito na Seção 1.6) e é usada através de uma aplicação baseada numa consulta de seleção, junção e projeção sobre fontes de dados relacionais..
2. **AQEE *(Adaptive Query Execution Engine)*** para suporte ao modelo de execução adaptável e ao modelo de dados relacional. Neste caso, a MEC é usada de forma *ad hoc* através de uma aplicação baseada numa consulta de integração de dados a *sites web* onde a adaptabilidade está associada à imprevisibilidade do tempo de acesso a estes *sites*. Aplicações como essa apresentam desafios na medida em que o tempo de acesso às fontes de dados registra grandes variações e as informações estatísticas são escassas.
3. **XQEE *(Xml Query Execution Engine)*** para suporte ao modelo de dados XML e do modelo de execução baseado em *streams*. Neste caso, a MEC é usada de forma *ad hoc* através de uma aplicação baseada numa

consulta de seleção e projeção sobre uma fonte de dados XML. Aplicações como essa apresentam desafios na medida em que as fontes de dados XML devem ser acessadas através de um analisador (*parser*) que controlam o fluxo de dados da MEC, resultando num modelo de execução com fluxo de controle baseado em dados, além do fato de algumas fontes não possuírem descrição do seu esquema, caracterizando-se um *streams* de dados. Outro desafio consiste no projeto dos operadores algébricos XML e da estrutura de dados para as tuplas XML processadas por estes operadores, ambos baseados no modelo de dados XML.

Além dessas máquinas, estão sendo instanciadas as máquinas: RXQEE *(Relational-Xml Query Execution Engine)*, para suportar tanto o modelo de dados Relacional quanto o modelo de dados XML; ROSA-QEE e CODIMS-QEE para o suporte aos modelo de dados dos sistemas ROSA e CODIMS, respectivamente (descritos na seção 1.6).

Implementação

O QEEF foi projetado segundo o paradigma de orientação a objetos e implementado na plataforma *Java*. Para testar o protótipo, utilizamos o ambiente computacional distribuído do laboratório TecBD da PUC-Rio (TecBD, 2003).

1.4. Contribuições da Tese

A principal contribuição desta tese está no desenvolvimento de uma MEC extensível (QEEF) que suporta diferentes modelos de execução e de dados de forma ortogonal. Não conhecemos na literatura uma MEC que possa combinar e estender módulos de maneira uniforme e flexível visando este suporte.

As contribuições específicas desta tese são as seguintes:

- A análise dos modelos de execução dos cenários apresentados;
- A criação do meta-modelo QUEM que permite a criação e a combinação de novos módulos de execução resultando na extensibilidade para novos modelos de execução;

- A implementação do QEEF e a sua instanciação, produzindo as MECs RQEE, AQEE, XQEE. Cada instanciação reutiliza uma infra-estrutura básica, sem que o construtor tenha que construí-la do estágio inicial;
- A integração do QEEF aos projetos de pesquisa em andamento descritos na Seção 1.6;
- As seguintes publicações: (Ayres et al. 2002a, b, c, 2003).

1.5. Foco da Tese

De uma maneira geral, o foco deste trabalho está direcionado à execução de consultas baseada nos modelos de execução de alguns dos cenários apresentados e nos modelos de dados Relacional e XML. Questões relacionadas ao armazenamento e atualização de fontes de dados, interface com usuário, processamento de transações e otimização estão fora do escopo deste trabalho.

1.6. Contexto da Tese

Esta tese está inserida em alguns projetos de pesquisa de banco de dados, os quais são descritos a seguir:

a) CODIMS *(Configurable Data Integration Middleware System):*

Esse projeto (Barbosa&Porto, 2001) está em andamento no Laboratório de Tecnologia de Banco de Dados do DI/PUC-Rio e visa o desenvolvimento de um *framework* composto por uma família de componentes de *software* (*framelets*), os quais implementam os principais serviços de um sistema de integração de dados na *web*. Dentre estes, o componente QueryProcessing é responsável pelo processamento de consultas e contém o componente QueryEngine para a execução destas. Um dos requisitos do CoDIMS é que o componente QueryProcessing deve ser flexível o suficiente para lidar com várias arquiteturas de integração. Isto requer que a sua MEC possa ser também configurada para atender a esta flexibilidade, ou seja, executar PECs com diferentes modelos de dados e de execução. A integração do QEEF ao CODIMS visa atender a este requisito.

b) Projeto TINO (*Tino Is Not Oracle*)

Este projeto (Tino, 2003) é coordenado pelo Professor Sérgio Lifschitz no Laboratório de Paralelismo do DI/PUC-Rio e visa a construção de um SGBD relacional baseado em componentes para ser utilizado em pesquisas acadêmicas e como recurso didático para ensino de banco de dados. Neste contexto, estamos instanciando, a partir do QEEF, a MEC denominada de RQEE, utilizada como componente do processador de consultas do TINO.

c) Projeto ROSA (*Repository of Objects with Semantic Access*)

Este projeto é coordenado pelos Professores Ana Maria Moura e Fabio Porto no Instituto Militar de Engenharia (IME) e visa a construção de um ambiente para o armazenamento de Objetos de Aprendizados e de processamento de consultas baseadas na álgebra do modelo de dados do ROSA (Porto et al., 2003). Neste contexto, estamos instanciando, a partir do QEEF, a MEC denominada de ROSA-QEE, utilizada para executar consultas na álgebra do ROSA.

1.7. Organização da Tese

Este trabalho está estruturado em sete capítulos descritos a seguir:

O Capítulo 2 descreve uma visão geral dos conceitos relacionados ao processamento de consultas em SGBDs e os cenários de execução de consultas introduzidos neste capítulo.

O Capítulo 3 descreve a solução proposta para o suporte a diferentes modelos de execução e de dados.

O Capítulo 4 introduz o conceito de frameworks e descreve o framework QEEF.

O Capítulo 5 descreve o meta-modelo QUEM que permite a especificação de PECs para vários modelos de execução.

O Capítulo 6 apresenta os estudos de casos com a instanciação do QEEF para diferentes modelos de execução e de dados.

O Capítulo 7 apresenta as conclusões deste trabalho incluindo as considerações finais e trabalhos futuros.

1.8. Síntese do Capítulo

Neste capítulo, foram apresentados a motivação, o objetivo e uma visão geral da solução da tese. A motivação se baseia no fato de que o surgimento de novos cenários de execução de consultas suscitou novos modelos de dados e de execução, ambos não suportados pelas Máquinas de Execução de Consultas (MEC) tradicionais, resultando num esforço considerável de desenvolvimento de MECs. Devido a isto, esta tese tem como objetivo, a construção de MECs para o suporte a diferentes modelos de execução e de dados. Para alcançar este objetivo, desenvolveu-se o QEEF (*Query Execution Engine Framework*) baseado na técnica de *framework* de *software* e que possui um meta-modelo de execução denominado de QUEM *(QUery Execution Meta-model).*