ULBRA
CENTRO UNIVERSITÁRIO LUTERANO DE PALMAS
COMUNIDADE EVANGÉLICA LUTERANA "SÃO PAULO"
Recredenciado pela Portaria Ministerial nº 3.607 - D.O.U. nº 202 de 20/10/2005

Marcos Paulo Honorato da Silva

Criação e Implantação de Bancos de Dados Distribuído para a Fábrica de Software do CEULP/ULBRA

Palmas
2012

ULBRA CENTRO UNIVERSITÁRIO LUTERANO DE PALMAS
COMUNIDADE EVANGÉLICA LUTERANA "SÃO PAULO"
Recredenciado pela Portaria Ministerial nº 3.607 - D.O.U. nº 202 de 20/10/2005

Marcos Paulo Honorato da Silva

Criação e Implantação de Bancos de Dados Distribuído para a Fábrica de Software do CEULP/ULBRA

Projeto apresentado como requisito parcial da disciplina de Trabalho de Conclusão de Curso II (TCC II) do curso de Sistemas de Informação, orientado pelo Professor Mestre Jackson Gomes de Souza.

Palmas
2012

ULBRA CENTRO UNIVERSITÁRIO LUTERANO DE PALMAS
COMUNIDADE EVANGÉLICA LUTERANA "SÃO PAULO"
Recredenciado pela Portaria Ministerial nº 3.607 - D.O.U. nº 202 de 20/10/2005

Marcos Paulo Honorato da Silva

Criação e Implantação de Bancos de Dados Distribuído para a Fábrica de Software do CEULP/ULBRA

Projeto apresentado como requisito parcial da disciplina de Trabalho de Conclusão de Curso II (TCC II) do curso de Sistemas de Informação, orientado pelo Professor Mestre Jackson Gomes de Souza.

Aprovada em novembro de 2012.

BANCA EXAMINADORA

_____

Prof. MSc. Jackson Gomes de Souza
Centro Universitário Luterano de Palmas

_____

Prof. MSc Fernando Luiz de Oliveira
Centro Universitário Luterano de Palmas

_____

Prof. M.Sc Madianita Bogo Marioti
Centro Universitário Luterano de Palmas

Palmas
2012

RESUMO

Cada vez mais estudos estão sendo realizados na área de Banco de Dados devido à sua importância para o bom funcionamento de sistemas de informação. Este trabalho aborda conceitos de bancos de dados distribuídos (características, fragmentação e replicação de dados), bem como apresenta uma proposta de banco de dados distribuído para a Fábrica de Software do CEULP/ULBRA. São mostrados os procedimentos necessários (utilização e configuração) para disponibilizar parte do banco de dados de forma distribuída usando SQL Server Management Studio.

Palavras-chave: banco de dados distribuído, SQL Server, banco de dados fsw-conteudo.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

1 INTRODUÇÃO

Como o acesso à internet está se tornando mais fácil, a necessidade de recuperar dados de maneira rápida e segura em sistemas tem sido um assunto bastante relevante para as empresas. Tradicionalmente, os bancos de dados dessas empresas se encontram em um único espaço físico, ou seja, possui uma arquitetura centralizada. Nesta arquitetura, vários problemas estão mais dispostos a ocorrer, tal como a falha do servidor do Banco de Dados (BD) ocasionando a inoperabilidade de todo o sistema. Esta indisponibilidade do sistema traz enormes consequências financeiras como, por exemplo, as vendas que não foram efetivadas (BARRAZA, 2002, p. 4).

A fim de sanar problemas como a indisponibilidade de BD ou o tempo de recuperação das informações (performance), novas estratégias de organização e distribuição dos dados de um sistema estão sendo estudadas. Estas estratégias envolvem tecnologias de banco de dados, redes e comunicação de dados, e compreendem a área de Banco de Dados Distribuído (BDD).

Atualmente, as empresas estão cada vez mais dependentes do sistema de banco de dados devido ao fato que as informações armazenadas são essenciais para o funcionamento das mesmas. Desta maneira, o banco de dados é considerado como uma peça chave no sistema. Os avanços da tecnologia levaram a novas aplicações dos sistemas de banco de dados o que tornou possível armazenar também galerias de fotos, vídeos e áudios e não apenas informações textuais e numéricas. Portanto, o volume de informações em sistemas de banco de dados tem aumentado significativamente.

Além disso, o acesso a essas informações tem crescido por causa da diversidade de dispositivos móveis conectados à Internet como, por exemplo, *tablets*, *smatphones* e PDAs (*personal digital assistent*). Sabe-se que há um alto índice de acessos ao banco de dados da Fábrica de Software, do Centro Universitário Luterano de Palmas. Desta forma, foi feito um levantamento estatístico, utilizando a ferramenta *Google Analytics*, sobre a quantidade de acessos dos

usuários ao seu sistema nos últimos quatro anos. Para um melhor entendimento do que ocorreu nesse período pode-se observar o Gráfico 1.

Gráfico 1: Quantidade de Acessos ao Sistema da Fábrica de Software (Fonte: Fábrica de Software).

Através deste gráfico, pode-se observar que está tendo uma evolução do número de acessos ao sistema com o passar dos anos, sendo em torno de 300 mil em 2008 e, em 2011 (até 24/11/2011), este número já ultrapassou 1 milhão.

Um segundo parâmetro coletado foi a média do número de publicações junto ao sistema da Fábrica de Software. No Gráfico 2, pode-se conferir a média de publicações mensal no Portal no período de 2002 a 2011.

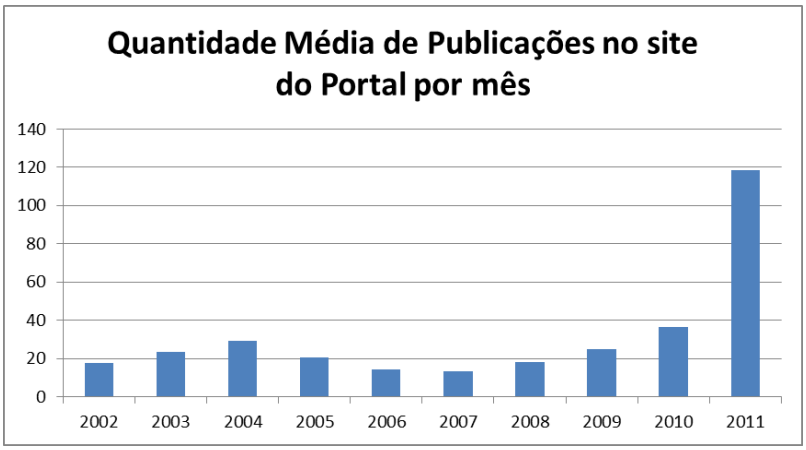


Gráfico 2: Número de Publicações ao Sistema da Fábrica de Software (Fonte: Fábrica de Software).

É possível observar que é o número de publicações no ano de 2011 aumentou significativamente. Uma das justificativas é a oferta de novos cursos na Instituição e a divulgação mais frequente das atividades dos cursos ofertados.

Um terceiro parâmetro coletado foi o tempo de acesso à página inicial do Portal. Em média, é gasto 5 segundos para carregar a página inicial do Portal. Já para o processamento das consultas é gasto 0,712095242 segundos. Portanto, 15% do tempo gasto para o carregamento da página inicial do Portal é com a comunicação do banco de dados.

A Fábrica de Software tem um grande número de acessos por parte dos professores, alunos e comunidade em geral. Devido a isto, o acesso torna mais difícil durante os períodos de provas, vestibular e matrícula. Além disso, o banco de dados funciona de forma tradicional, com acesso aos dados centralizados e controlados. Dessa maneira, qualquer falha no servidor de banco de dados fará com que o sistema fique inoperante. Diante deste contexto, justifica-se o estudo de tecnologias de banco de dados distribuídos a fim de separar os dados em vários computadores, possibilitando uma maior agilidade na busca das informações solicitadas pelos usuários, visto que atualmente o banco de dados se encontra de forma centralizada.

Este trabalho apresenta um estudo de aplicação de BDD para um dos bancos da Fábrica de Software. Assim, são apresentados conceitos de BDD e também as configurações necessárias para um modelo distribuído usando a ferramenta SQL Server Management Studio, apresentado os componentes necessários para a comunicação de forma distribuída. Para alcançar este objetivo, foi preciso:

- descrever e entender os principais conceitos de banco de dados distribuído;
- entender o modelo de banco de dados da Fábrica de Software;
- estudar ferramentas para criação de banco de dados distribuídos para plataforma Windows e SQL Server;
- elaborar e implantar a proposta de banco de dados distribuídos;
- definir critérios nos testes de desempenho e viabilidade tanto da situação atual do banco de dados da Fábrica de Software quanto da proposta;
- realizar os testes de desempenho na proposta de BDD;
- avaliar e analisar os resultados dos testes.

A hipótese do trabalho foi que o desempenho de acesso às informações da Fábrica de Software melhore com a distribuição (fragmentação e replicação) do banco de dados em partes menores, pois haverá uma separação das consultas às tabelas do banco. Desta forma, o usuário do sistema terá um tempo de resposta menor às suas solicitações e, assim, melhorando a performance do sistema.

Este trabalho está organizado da seguinte forma: no capítulo 2 será apresentado o referencial teórico referente à área de Banco de Dados Distribuído; no capítulo 3 serão apresentados os materiais e métodos utilizados para a elaboração deste trabalho; no capítulo 4 são mostrados os resultados e discussões; no capítulo 5 serão apresentadas as considerações finais deste trabalho e trabalhos futuros e, por fim, são listadas as referências bibliográficas.

2 REFERENCIAL TEÓRICO

Sistemas de banco de dados são elementos essências em uma sociedade moderna. De alguma maneira é possível se deparar com atividades que envolvam um banco de dados como, por exemplo, transferência bancária via *home banking* ou compras online (DATE, 1991, p. 20). Até mesmo uma compra de uma passagem aérea, a atualização do banco de dados é automática em relação ao controle do número de poltronas disponíveis. Com isso, observa-se que banco de dados desempenha uma função primordial nas áreas que utilizam recursos computacionais como: medicina, engenharia, *e-commerce* e economia.

A tecnologia de banco de dados distribuídos tem como propósito incorporar tanto à eficiência e transparência a tecnologia de banco de dados bem como a de processamento distribuído. A distribuição do processamento e de grande valia para a tecnologia de banco de dados de forma em que a descentralização das aplicações, tarefas e armazenamento, seja de maneira transparência e a integração.

Um Banco de Dados Distribuído (BDD) pode ser visto com uma coleção de múltiplos bancos de dados logicamente inter-relacionados, distribuídos por uma rede de computadores, e um sistema de gerenciamento de banco de dados distribuídos (ELMASRI e NAVATHE, 2011, p.592). Dentre as várias vantagens do BDD, destacam-se:

- **maior facilidade e flexibilidade de desenvolvimento** - o desenvolvimento e a manutenção de aplicações em nós geograficamente distribuídos são facilitados devido à transparência da distribuição e controle de dados;
- **maior confiabilidade e disponibilidade** - o isolamento de falhas na origem fará com que os outros bancos de dados não sejam afetados. Assim, quando um nó vier a falhar os outros continuarão funcionando. Desta forma, aumenta tanto a confiabilidade quanto a disponibilidade;
- **maior desempenho** - a distribuição de um banco de dados grande em vários nós possibilita um melhor desempenho, pois as consultas e transações são feitas em banco de dados menores;
- **facilidade de expansão** – maior possibilidade de incluir nós no sistema de BDDs sem afetar a estrutura atual. Desta forma, um novo nó com dados já armazenados em outro(s) nó(s) pode ser adicionado ao sistema facilmente.

As seções a seguir apresentam as características fundamentais de Sistemas de Banco de Dados Distribuídos (SBDDs), que são transparência, confiabilidade e disponibilidade. Em seguida, são apresentados os componentes que compõem uma arquitetura de SBDDs e, ainda, os aspectos relacionados a um projeto de SBDDs.

2.1 Características de SBDDs

2.1.1 Transparência

De acordo com Özsu e Valduriez, em um SBDD, os dados e software são distribuídos entre nós interligados através de uma rede de computadores. Por ser distribuído, surge à questão da transparência para o usuário que é definido como a separação entre a semântica de alto nível de um sistema e os seus detalhes de implementação (ÖZSU e VALDURIEZ, 2002, p.7). Dessa forma, o sistema transparente "esconde" o detalhe da implementação do usuário, assim, a visão para os usuários se torna integrada em relação ao banco de dados, mesmo esse sendo fisicamente distribuído.

Algumas formas de transparência do SBDD são apresentadas por (ELMASRI e NAVATHE, 2011, p.591), que são:

- **de rede:** refere-se a liberdade do usuário de detalhes operacionais da rede e do posicionamento dos dados no sistema distribuído, podendo ser local (a tarefa que o usuário vai realizar independe do local dos dados e local do nó onde o comando foi emitido) ou nomes (o acesso a um objeto nomeado é feito de maneira que não exista ambiguidade de nomes);
- **de replicação:** para garantir a disponibilidade, o desempenho e a confiabilidade, são feitas cópias dos mesmos objetos de dados e armazenadas em vários nós, sem o conhecimento do usuário;
- **de fragmentação:** refere-se a divisão da relação em subconjuntos, sendo que a horizontal distribui a relação em subconjuntos de tuplas e a vertical distribui a relação em subconjuntos de atributos.

Além destas formas de transparência, outras podem ser aplicadas em um BDD como transparência de projeto (como o BDD foi construído) e também transparência de execução (onde uma transação é executada).

2.1.2 Confiabilidade e Disponibilidade

Conforme ELMASRI e NAVATHE (2011, p. 592), a confiabilidade pode ser definida em termos gerais como a probabilidade de um sistema estar funcionando (não parado) em certo ponto no tempo. Já a disponibilidade é a probabilidade de que o sistema esteja continuamente disponível durante um intervalo de tempo.

Cabe ressaltar que as características de confiabilidade e de disponibilidade estão diretamente relacionadas com defeitos, erros e falhas do BD. Assim, uma vez que os dados forem replicados, há uma eliminação de um único ponto de falha. Portanto, a falha de um nó não será capaz de parar um sistema inteiro, pois os outros continuam funcionando. Assim, somente os dados e software que estão no nó inoperante não poderão ser acessados. Já em um sistema centralizado, a falha em um único nó deixa todos os usuários sem acesso ao sistema.

2.1.3 Autonomia

A autonomia determina o grau de independência dos nós individuais em um SBDD. Assim, um nó com elevado grau de autonomia proporciona uma maior flexibilidade e manutenção no sistema. Esta autonomia pode ser aplicada em três aspectos, que são: projeto (independência do modelo de dados), comunicação (independência do compartilhamento de informações com outros nós) e execução (independência de execução das transações).

2.2 Arquitetura de SBDDs

A arquitetura de um BDD define sua estrutura, mostrando os seus componentes e as suas atribuições. Além disso, apresenta como ocorre o relacionamento entre os componentes. Para um maior entendimento de um BDD, a Figura 1 apresenta os componentes que fazem parte de sua arquitetura.

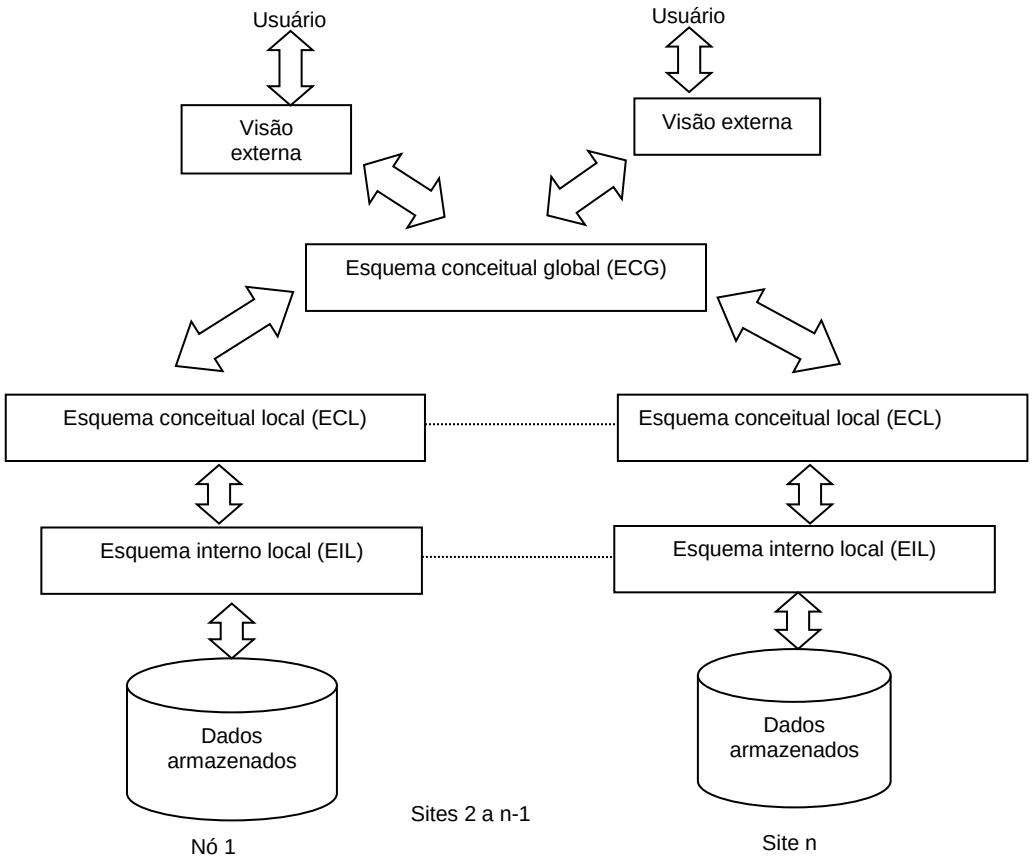


Figura 1: Arquitetura de um BDD (ELMASRI e NAVATHE, 2011, p. 598).

O componente Esquema Conceitual Global (ECG) é responsável pela transparência de rede para os usuários, fazendo o direcionamento para os vários componentes de Esquema Conceitual Local (ECL). Cada ECL descreve o banco de dados local e este define o Esquema Interno Local (EIL) para cada nó. Portanto, cada nó tem os modelos lógico e físico de cada banco armazenado.

Deste modo, percebe-se que mesmo com os dados do banco fracionados em vários nós de uma rede de computadores, o usuário ainda tem a impressão de que os mesmos se encontram centralizados. Além disso, um diferencial nesta arquitetura é que parte do banco (um nó da rede, por exemplo) pode estar separada para manutenção sem deixar indisponível o restante do banco.

2.3 Projetos de SBDDs

Nesta seção será abordada técnica de fragmentação de dados que é utilizada para particionar um banco em unidades lógicas e alocar seus fragmentos entre os

nós do sistema. Também será abordada a utilização da replicação de dados, que tem como objetivo o armazenamento de dados de um nó em outros nós. Essas técnicas farão parte do processo de projeto de banco de dados distribuídos.

2.3.1 Fragmentação de Dados

Em um projeto de BDD, a fragmentação dos dados é uma etapa em que ocorre a definição de qual nó deve ser utilizado para armazenar certas partes do banco de dados. Isso favorece que dados que não são importantes para determinadas aplicações não sejam acessados, com isso, reduzindo os recursos de entrada e saída e aumentando o desempenho (KARLAPALEM et al, 1994, p. 186).

Quando uma relação ***r*** for fragmentada, ***r*** será subdividida em vários fragmentos como r_1, r_2, r_3, ..., r_n. Logo, esses fragmentos possuem informações necessárias para a reconstrução da relação ***r*** (SILBERSCHATZ et al., 2006, p.591). De modo geral, existem dois tipos de técnica de fragmentação de dados, que são: horizontal e vertical.

2.3.1.1 Fragmentação Horizontal

Em uma fragmentação horizontal, cada tupla da relação ***r*** pertence a um ou mais fragmentos. Assim, a reconstrução de ***r*** pode ser vista como: $r = r_1 \cup r_2 \cup r_3 ... \cup r_n$.

Para facilitar o entendimento da fragmentação horizontal, considere a relação ***r*** como a relação Setor, apresentada na Tabela 1.

Tabela 1: Relação Setor.

Nome_setor	Nome_maquina	Nome_usuario
Cac	Rf220	José
Satec	Rf398	Fernando
Cac	Rf765	Julio
Satec	Rf827	Lara
Cac	Rf221	Maria

Sabendo que existem dois setores, a relação Setor será particionada em dois fragmentos chamados de Setor 1 e Setor 2, conforme pode ser observado nas Tabelas 2 e 3, respectivamente.

Tabela 2: Fragmento Setor 1.

Nome_setor	Nome_maquina	Nome_usuario
Cac	Rf220	José
Cac	Rf221	Maria
Cac	Rf765	Julio

Tabela 3: Fragmento Setor 2.

Nome_setor	Nome_maquina	Nome_usuario
Satec	Rf398	Fernando
Satec	Rf827	Lara

Neste caso, uma tupla se apresenta em apenas um fragmento. No entanto, como dito anteriormente, uma tupla pode ser alocada em mais de um fragmento, sendo que isso é feito através da replicação de dados, que será visto adiante.

2.3.1.2 Fragmentação Vertical

Já a fragmentação vertical está relacionada com a partição dos atributos de uma relação ***r*** em vários fragmentos. Desta maneira, os fragmentos armazenam parte dos atributos R_i de uma relação. Assim, a reconstrução de ***r*** pode ser vista como: $r = r_1 |X| r_2 |X| r_3 |X| \ldots |X| r_n$. Nesse tipo de fragmentação para se garantir que a reconstrução de ***r*** seja efetuada é preciso que seja acrescido uma chave primária a relação ***r***, e essa é disponibilizada em cada fragmento.

Considere agora a relação Controle com o acréscimo do campo tupla_id que serve para identificar cada tupla na relação, como pode ser visto na Tabela 4.

Tabela 4: Relação Controle com tupla_id.

Nome_setor	Nome_maquina	Nome_usuario	tupla_id
Cac	Rf220	José	1
Satec	Rf398	Fernando	2
Cac	Rf765	Julio	3
Satec	Rf827	Lara	4
Cac	Rf221	Maria	5

As Tabelas 5 e 6 são resultados de uma fragmentação vertical da relação Controle, vista anteriormente. Neste caso, foram agrupados os atributos Nome_maquina, Nome_setor e tupla_id para o fragmento Controle 1, e Nome_maquina, Nome_usuario e tupla_id para o fragmento Controle 2. O fragmento Controle 1 foi destinado para se obter informações do setor em que máquinas se encontram e o Controle 2 direcionado para guardar informações dos usuários das máquinas.

Tabela 5: Fragmento Controle 1.

Nome_maquina	Nome_setor	tupla_id
Rf220	Cac	1
Rf398	Satec	2
Rf765	Cac	3
Rf827	Satec	4
Rf221	Cac	5

Tabela 6: Fragmento Controle 2.

Nome_maquina	Nome_usuario	tupla_id
Rf220	José	1
Rf398	Fernando	2
Rf765	Julio	3
Rf827	Lara	4
Rf221	Maria	5

Logo, por meio do atributo tupla_id é possível reconstruir toda a relação Controle.

2.3.1.3 Fragmentação Híbrida

Na maioria dos projetos de banco de dados distribuídos, a utilização de uma única técnica de fragmentação já satisfaz as necessidades, no entanto, há situações em que a combinação das duas técnicas apresentadas é necessária, sendo assim, conhecida como híbrida ou mista. Desta forma, pode-se fazer uma fragmentação vertical primeiro e, depois, uma horizontal ou vice-versa.

Para exemplificar será utilizada a Tabela 5, fragmento Controle 1, que já corresponde a uma fragmentação vertical. A partir dela foi aplicada uma fragmentação horizontal, em que foi dividida pelo atributo Nome_setor. Neste caso, foram criados dois fragmentos chamados de Fragmento Controle 1.1 e Fragmento Controle 1.2, que mantêm informações do setor Cac e Satec, respectivamente. Esses dois novos fragmentos podem ser observados nas Tabelas 7 e 8.

Tabela 7: Fragmento Controle 1.1.

Nome_maquina	Nome_setor	tupla_id
Rf220	Cac	1
Rf765	Cac	3
Rf221	Cac	5

Tabela 8: Fragmento Controle 1.2.

Nome_maquina	Nome_setor	tupla_id
Rf398	Satec	2
Rf827	Satec	4

Em relação a esses fragmentos a possibilidade de pertencerem a nós diferentes.

2.3.1.4 Princípios de Fragmentação

Em um processo de fragmentação, três princípios devem ser levados em consideração para garantir a semântica do banco de dados, que são (ÖZSU; VALDURIEZ, 2001, p. 79):

- **completude**: os dados de uma relação global são mapeados a partir dos fragmentos, garantindo que não ocorra perda de dados. Na fragmentação horizontal, a completude atua com a reunião as tuplas e na fragmentação vertical com os atributos.
- **reconstrução**: na fragmentação horizontal a reconstrução ocorre através da operação de união entre os fragmentos de uma relação R, e na vertical pela operação de junção dos fragmentos.
- **disjunção**: caso uma relação R for fragmentada horizontalmente em N fragmentos, os dados do fragmento F1 não poderão estar no fragmento F2. Com isso, garante que a relação R esteja totalmente disjunta. Já caso da fragmentação vertical, a disjunção é definida sobre os atributos que não pertencem à chave primária, pois cada fragmento deve conter todas as chaves primárias da relação R.

2.3.2 Replicação e Alocação de Dados

Normalmente, em um projeto de distribuição de dados, a etapa de alocação de dados é realizada após a fase de fragmentação, indicando onde cada fragmento vai ficar. Uma das grandes dificuldades na alocação dos dados é achar a distribuição “ótima” dos dados nos nós da rede, fazendo com que o custo das aplicações sobre esses dados seja minimizado. Em sistemas de BDDs a alocação é considerada um fato critico, pois, uma estratégia de alocação ineficiente dos dados pode ocasionar um aumento no custo de acesso ao SBDD.

A replicação dos dados, basicamente, consiste em duplicar dados em vários nós do sistema. Esta duplicação de dados pode ser parcial ou total. A replicação parcial dos dados ocorre quando parte do banco de dados está armazenado em outro(s) nó(s). No caso da replicação total, todo o banco de dados está duplicado em outros nós.

O grau de replicação dos dados está associado com as necessidades de cada sistema em relação ao desempenho e disponibilidade. Assim, quanto maior for a disponibilidade exigida, maior deve ser o grau de replicação dos dados. Portanto, quando se exige uma alta disponibilidade das transações, deve-se optar pela replicação total, pois as transações podem ser efetuadas em qualquer nó do sistema.

Já o desempenho é afetado devido às atualizações serem feitas em vários nós. Se houver um alto número de atualizações nos outros nós, o desempenho será prejudicado. Portanto, encontrar uma solução ideal para alocação de dados distribuídos não é uma tarefa fácil. Isto porque cada sistema tem seu contexto compreendendo, por exemplo, a capacidade de comunicação entre nós, de armazenamento, entre outros.

Quando ocorre uma alteração no banco de dados, a replicação desses dados pode ocorrer de duas formas, de acordo com o tempo de latência, que são:

- **replicação síncrona**: o tempo de latência para esse tipo de sincronização é zero, pois se ocorrer alguma modificação na cópia do banco de dados, essa será aplicada as demais réplicas automaticamente. A utilização desse tipo de replicação é mais utilizada em aplicações em que necessitam de um nível de atualização precisa entre os servidores.
- **replicação assíncrona**: ao contrário da replicação síncrona apresentada anteriormente, nesta os dados serão atualizados nas réplicas em um determinado momento. A modificação nas outras réplicas ocorrerá de forma programada, podendo ser a cada um minuto ou a cada hora. Assim o tempo de latência é maior que zero.

A Figura 2 mostra um exemplo de replicação de dados em um BDD.

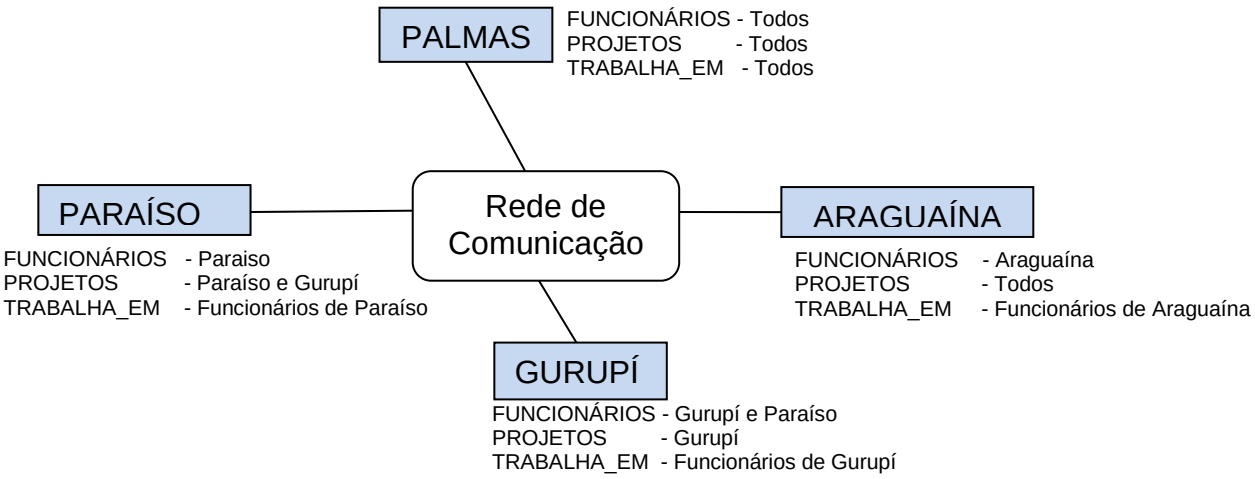


Figura 2: Replicação de dados em vários nós (Adaptado de ELMASRI e NAVATHE, 2011).

2.3.3 Tipos de Estratégia

Basicamente, existem dois tipos de estratégia para a elaboração de um projeto de BDDs, que são: ascendente e descendente.

2.3.3.1 Ascendente (*bottom-up*)

A estratégia ascendente é direcionada quando múltiplas bases de dados já existem. Assim, a meta é integrá-las em apenas uma base de dados. Esta estratégia analisa os esquemas conceituais locais, de cada base de dados, para formar o esquema conceitual global.

2.3.3.2 Descendente (*top-down*)

A estratégia descendente, ao contrário da ascendente, é voltada para a análise do esquema conceitual global e, a partir da necessidade do sistema de BD, constrói os esquemas conceituais locais. Cada esquema conceitual local está associado a uma parte da base de dados original, criados a partir de técnicas de fragmentação e alocação.

2.4 Processamento de Consultas em BDDs

Um dos objetivos de um projeto de banco de dados distribuído é garantir um maior poder de processamento das consultas, fazendo com que as consultas mais relevantes sejam feitas em paralelo. Por isso, uma das finalidades do processador de consultas distribuídas é fazer o mapeamento das consultas de alto nível para um conjunto operações de banco de dados em fragmentos da relação.

O processamento de consulta em ambientes distribuídos tende a ser mais complexo se comparado aos centralizados, pois um número maior de variáveis pode afetar o desempenho da consulta distribuída (ÖZSU e VALDURIEZ, 2011, p.205). Isto porque as relações podem estar fragmentas e/ou replicadas tendo, assim, um custo de comunicação (*overhead*). Para Elmasri e Navathe, além deste custo, é preciso observar os custos com relação à concorrência de consultas, a decomposição em subconsultas e, por fim, a combinação do resultado das subconsultas para gerar o resultado final (ELMASRI E NAVATHE, 2011, p.609).

2.4.1 Estágios do Processamento de Consulta

O processamento de uma consulta em um BDDs ocorre em 4 (quatro) estágios, que são (ELMASRI E NAVATHE, 2011, p. 605):

2.4.1.1 Mapeamento de Consultas

Este estágio envolve o mapeamento de uma consulta na linguagem de consulta (SQL) para a linguagem algébrica (álgebra relacional), sem considerar a distribuição e replicação dos dados. As informações necessárias para este mapeamento são encontradas no esquema conceitual global, que descrevem as relações globais, de forma semelhante à realizada nos SGBDs centralizados (ELMASRI E NAVATHE, 2011, p. 605).

O mapeamento de consultas, basicamente, envolve 4 passos, que são:

- **normalização**: modifica a consulta de alto nível em uma consulta normalizada para facilitar o processamento utilizando operadores lógicos AND e OR. A modificação mais importante é na cláusula WHERE de instruções SQL. Basicamente, ocorre a modificação da instrução SQL para uma das duas formas normais, que são: forma normal conjuntiva e a forma normal disjuntiva.

  A forma normal conjuntiva é uma conjunção ($\wedge$) de disjunções ($\vee$) que pode ser representada como:

  $$(p_1 \vee p_2 \vee p_3 \vee \ldots \vee p_n) \wedge (q_1 \vee q_2 \vee q_3 \vee \ldots \vee q_n) \wedge \ldots \wedge (z_1 \vee z_2 \vee z_3 \vee \ldots \vee z_n)$$

  Já a forma normal disjuntiva é uma disjunção ($\vee$) de conjunção ($\wedge$) que pode ser representada como:

  $$(p_1 \wedge p_2 \wedge p_3 \wedge \ldots \wedge p_n) \vee (q_1 \wedge q_2 \wedge q_3 \wedge \ldots \wedge q_n) \vee \ldots \vee (z_1 \wedge z_2 \wedge z_3 \wedge \ldots \wedge z_n)$$

  No caso da forma normal disjuntiva, a consulta pode ser executada como várias subconsultas conjuntivas independentes, ligadas pela união (disjunção). Por exemplo:

Consulta original:

```
select  codigo, nome, nomesetor
from   Pessoa, Setor
where Setor.idsetor = Pessoa.idsetor
and salario > 1000
and ((cidade = 'Palmas') or  (cidade = 'Paraiso'))
```

Consulta na forma normal disjuntiva:

```
((setor.idsetor = Pessoa.idsetor) ∧ (salario > 1000) ∧
(cidade = 'Palmas'))
∨
((setor.idsetor = Pessoa.idsetor) ∧ (salario > 1000) ∧
(cidade = 'Paraiso'))
```

- **análise**: ocorre uma verificação das consultas a fim de encontrar consultas incorretas semanticamente e, assim, rejeitá-las. Por exemplo:

  Consulta incorreta:

  ```
  select  nome
  from   Pessoa
  where nome > 10
  ```

  Neste exemplo, a consulta encontra-se incorreta porque o atributo `nome` é do tipo VARCHAR e 10 é um inteiro.

- **simplificação**: elimina as eventuais redundâncias e simplifica a cláusula WHERE. Por exemplo:

  Consulta original:

  ```
  select  codigo, nome
  from   Pessoa
  where codigo > 10
  ```

```
and codigo > 50
and ((nome = 'Paulo') or (nome <> 'Paulo'))
```

Consulta simplificada:

```
select  codigo, nome
from    Pessoa
where codigo > 50
```

- **reescrita**: reescreve a consulta SQL em álgebra relacional. Por exemplo:

Consulta em SQL:

```
select  codigo, nome
from    Pessoa
where codigo > 50
```

Consulta em álgebra relacional:

$\pi_{\text{codigo, nome}}(\sigma_{\text{codigo > 50}}\ (\text{Pessoa}))$

A decomposição de consultas visa desmembrar as consultas solicitadas em partes menores (chamadas de subconsultas) que são efetuadas nos nós locais. Além disso, a decomposição de consultas é responsável por reorganizar os resultados obtidos de cada nó local para, em seguida, formar o resultado final.

2.4.1.2 Localização dos Dados

Este estágio visa localizar os dados da consulta utilizando as informações de distribuição e replicação de dados dos fragmentos locais. Assim, determina quais fragmentos estão envolvidos na consulta e transforma a consulta distribuída em uma consulta nos fragmentos.

2.4.1.3 Otimização Global da Consulta

Este estágio consiste em obter uma estratégia de execução ótima para uma consulta distribuída. Para isso, dentre as opções de consulta disponíveis, verifica-se o custo de cada execução da consulta levando em consideração tempo de CPU

(tempo de processamento), E/S (tempo de acesso a disco) e de comunicação (tempo de envio de dados entre nós). O custo de comunicação é o mais importante porque os BDDs normalmente estão conectados em rede.

2.4.1.4 Execução da Consulta Local

Este estágio visa a execução de uma consulta em todos os nós que possuem fragmentos envolvidos. Cada uma dessas consultas é vista como um “consulta local”, que utiliza o esquema local. Esta consulta é similar a execução de uma consulta no banco de dados centralizado.

A Figura 3 mostra o fluxo de execução dos estágios do processamento de consultas.

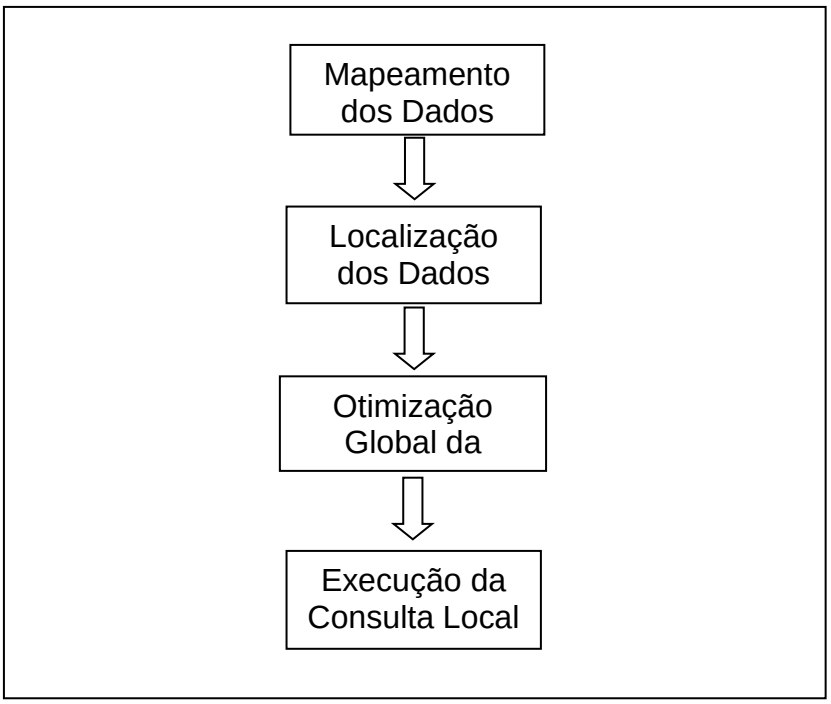


Figura 3: Estágios do Processamento de Consultas.

Portanto, ao receber uma consulta, o nó principal a interpreta e analisa, verifica onde os dados necessários se encontram, gera um plano de execução global e repassa para os nós locais efetuarem a consulta.

2.4.2 Estratégia de Execução da Consulta

Um aspecto importante do processamento de consulta em ambiente distribuído é a otimização, pois pretende encontrar, dentre as estratégias de execução, a mais próxima do ideal, ou seja, com o menor custo. Özsu e Valduriez apresentam uma função para o cálculo do custo da estratégia de execução distribuída (ÖZSU e VALDURIEZ, 2011, p.250). Esta função pode ser vista a seguir:

$$Custo = T_{CPU} * N_{instr} + T_{E/S} * N_{E/S} + T_{msg} * N_{msg} + T_{dados} * N_{bytes}$$

sendo:

T_{CPU} é o tempo médio de execução de uma instrução na máquina;

N_{instr} é o número de instruções executadas pelo processador;

$T_{E/S}$ é o tempo para carregar uma página em memória (uma unidade de E/S);

$N_{E/S}$ é o número de operações de E/S;

T_{msg} é um tempo fixo de inicialização e recebimento de uma mensagem;

N_{msg} é o número de mensagens trocadas entre nós;

T_{dados} é o tempo gasto para transmitir uma unidade de dados (p. ex., pacote);

N_{bytes} é o número de bytes transmitidos na mensagem.

Os dois primeiros componentes desta função estão relacionados com o tempo de processamento local, enquanto os dois últimos estão associados ao tempo de comunicação em rede.

Dentre os custos avaliados para a execução de uma consulta distribuída, o custo associado à transferência de dados pela rede é o mais relevante. É importante ressaltar que caso os nós do sistema estejam em uma rede local, o custo de transferência tende a ser baixo. No entanto, caso os nós estejam em uma rede remota (WAN – *Wide Area Network*), este custo tende a ser mais alto.

2.4.3 Exemplo de Estratégia de Execução da Consulta

Para exemplificar a escolha da melhor estratégia de execução de uma consulta distribuída, considere as relações SETOR e FUNCIONARIO, apresentadas a seguir.

SETOR(Numsetor, Nomesetor)
FUNCIONARIO(Cpf, Nome, Sobrenome, Nsetor)

A relação SETOR possui 4 e 10 bytes para os campos Numsetor e Nomesetor, respectivamente, e 20 registros armazenados no nó 1. Já a relação FUNCIONARIO possui 11, 15, 15 e 4 bytes para os campos Cpf, Nome, Sobrenome e Numsetor, respectivamente, e ainda 2.000 registros armazenados no nó 2.

A consulta para identificar o nome, o sobrenome e o nome do setor para o qual os funcionários trabalham seria necessária a seguinte instrução: SELECT nome, sobrenome, nomesetor FROM FUNCIONARIO INNER JOIN SETOR ON Nsetor = Numsetor. Traduzindo esta consulta para a álgebra relacional, tem-se: $\pi_{nome,\ sobrenome,\ nomesetor}$ (FUNCIONARIO |x| $_{Nsetor\ =\ Numsetor}$ SETOR)

Considerando que esta consulta tenha sido submetida ao nó 3 (nó de resultado), então existem 3 estratégias para executar a consulta distribuída que são:

- **opção 1**: transferir os registros das relações SETOR (nó 1) e FUNCIONARIO (nó 2) para o nó 3. Neste caso, serão transferidos um total de **90.280 bytes**, sendo:
  - do nó 1 para o nó 3 –> 14 bytes cada registro * 20 = 280 bytes
  - do nó 2 para o nó 3 -> 45 bytes cada registro * 2.000 = 90.000 bytes
- **opção 2**: transferir os registros da relação SETOR (nó 1) para o nó 2, executar a junção no nó 2 e, depois, enviar o resultado para o nó 3. Neste caso, são transferidos um total de **80.280 bytes**, sendo:
  - do nó 1 para o nó 2 -> 14 bytes cada registro * 20 = 280 bytes
  - do nó 2 para o nó 3 -> 40 bytes (Nome do funcionário (15) + sobrenome do funcionário (15) + nome do setor (10)) cada registro da consulta * 2.000 = 80.000 bytes;
- **opção 3**: transferir os registros da relação FUNCIONARIO (nó 2) para o nó 1, executar a junção no nó 1 e, depois, enviar o resultado para o nó 3. Neste caso, são transferidos um total de **170.000 bytes**, sendo:
  - do nó 2 para o nó 1 -> 45 bytes cada registro * 2.000 = 90.000 bytes
  - do nó 1 para o nó 3 -> 40 bytes (Nome do funcionário (15) + sobrenome do funcionário (15) + nome do setor (10)) cada registro da consulta * 2.000 = 80.000 bytes.

Caso o critério de escolha da estratégia de execução de uma consulta distribuída seja a quantidade de dados transferidos pela rede, então a opção 2 é a melhor e a que deve ser adotada. Comparando as opções 2 (melhor caso) e 3 (pior caso), verifica-se que caso a opção 3 fosse adotada, uma quantidade maior de bytes precisaria ser transferida entre os nós do sistema.

2.5 Comparações entre BD Centralizado e Distribuído

Algumas características que devem ser levado em consideração para um comparativo entre a tecnologia de BDD e centralizado são:

- **disponibilidade**: em um BDD há uma maior disponibilidade do banco visto que dados podem estar replicados em vários nós, ao contrário de um ambiente centralizado. Assim, se um nó estiver inoperante, outro pode estar apto para executar uma consulta;
- **complexidade**: em um projeto de BDD existe uma maior complexidade porque envolve aspectos que devem ser observados como fragmentação, alocação e replicação dos dados, ao contrário de um ambiente centralizado;
- **custo**: em um BDD observa-se que o custo tende a ser maior, visto que é preciso a aquisição de recursos de armazenamento (hardware) para os diversos nós. Além disso, há um custo para o processamento de consultas de forma distribuída (CPU, E/S e comunicação de rede);
- **controle de atualização**: em um BDD o controle de atualizações dos nós é mais dispendioso, pois os dados precisam estar consistentes nos vários nós, ao contrário de um ambiente centralizado que ocorre em um apenas um local.

A seguir serão apresentados os materiais e métodos utilizados para a realização deste trabalho.

3 MATERIAIS E MÉTODOS

Este capítulo apresenta os materiais utilizados neste trabalho, bem como a metodologia empregada para alcançar o objetivo proposto.

3.1 Local e Período

A elaboração deste trabalho ocorreu no Complexo de Informática do Curso de Sistemas de Informação do Centro Universitário de Palmas.

3.2 Materiais

Para a realização deste trabalho foram utilizadas várias ferramentas com objetivo de desenvolver a solução de BBD. O *Microsoft SQL Server Management Studio* foi utilizado para o gerenciamento do *SQL Server*. O *SQL Server Profiler* foi utilizado para rastrear execuções de consultas e tempos de execução. O Monitor de Desempenho foi utilizado para identificar o consumo de recursos no servidor, como CPU e memória. A seguir, cada ferramenta é apresentada com mais detalhes e também o banco de dados da Fábrica de Software do CEULP/ULBRA.

3.2.1 *Microsoft SQL Server Management Studio*

O *Microsoft SQL Server Management Studio* agrupa ferramentas gráficas para acesso ao SQL Server, direcionado para administradores e desenvolvedores de bancos de dados. Dentre as ferramentas disponíveis, a utilizada neste trabalho foi a SQL Server Profiler.

3.2.2 SQL Server Profiler

A ferramenta *SQL Server Profiler* está disponível no *Microsoft SQL Server Management Studio* e é direcionada para capturar informações relevantes para análise de desempenho como, por exemplo, quantas consultas o banco de dados está executando, quanto tempo estão demandando estas consultas e qual banco de dados está executando qual consulta. O *SQL Server Profiler* fornece 9 *templates* (modelos pré-definidos para a captura de classes de evento) para a captura de eventos de acordo com o propósito (MICROSOFT, 2012 a). A Tabela 9 mostra uma breve descrição dos *templates* e as principais classes de evento capturadas.

Tabela 9: Templates do SQL Profiler.

Nome do *Template*	Descrição	Principais Classes de Evento
SP_Counts	Captura comportamento de execução de *stored procedure* no tempo.	SP:Starting
Standard	Captura os *stored procedures* e lotes Transact-SQL que são executados.	Audit Login Audit Logout ExistingConnection RPC:Completed
TSQL	Captura todas as instruções Transact-SQL enviadas ao SQL Server por clientes e a hora em que foram emitidas.	Audit Login Audit Logout ExistingConnection
TSQL_Duration	Captura todas as instruções Transact-SQL enviadas ao SQL Server por clientes e seus tempos de execução, agrupando-as segundo a duração.	RPC:Completed SQL:BatchCompleted
TSQL_Grouped	Captura todas as instruções Transact-SQL enviadas ao SQL Server e a hora em que foram emitidas. Agrupa as informações pelo usuário que enviou a instrução.	ExistingConnection RPC:Starting SQL:BatchStarting
TSQL_Locks	Captura todas as instruções Transact-SQL enviadas ao SQL Server por intermédio de clientes juntamente com eventos de bloqueio excepcionais.	SP:StmtCompleted SP:StmtStarting SQL:StmtCompleted
TSQL_Replay	Captura informações detalhadas sobre as instruções Transact-SQL que serão necessárias se o rastreamento for reproduzido.	CursorOpen CursorPrepare CursorUnprepare
TSQL_SPs	Captura informações detalhadas sobre todos os procedimentos armazenados em execução.	Audit Login Audit Logout ExistingConnection
Tuning	Captura informações sobre procedimentos armazenados e execução Transact-SQL em lote.	RPC:Completed SP:StmtCompleted SQL:BatchCompleted

Além destes *templates*, é possível customizar os eventos a serem rastreados de acordo com a necessidade.

3.2.3 Monitor de Desempenho

O Monitor de Desempenho é uma ferramenta para a coleta de informações geradas pelo Windows examinando os programas que são executados em tempo real e como esses afetam o desempenho da máquina. Esta ferramenta utiliza de contadores de desempenho (medidas do estado ou atividade do sistema), dados de rastreamento do evento (dados coletados dos provedores do rastreamento) e informações de configuração (dados a serem armazenados no log), que podem ser relacionados nos conjuntos de coletores de dados.

3.2.4 Banco de Dados da Fábrica de Software do CEULP/ULBRA

A Fábrica de Software do CEULP/ULBRA desenvolve software para esta instituição de ensino e, consequentemente, necessita de armazenamento de dados, o que em grande parte do tempo é realizado em bancos de dados relacionais, sobre a plataforma do Microsoft SQL Server. Os bancos de dados, atualmente, estão centralizados em um único servidor (FENIX), o qual está em duas redes: a primeira rede é a internet, que vem de conexão direta do CPD do CEULP/ULBRA e a segunda rede é interna , realizando conexão com outros computadores.

Alguns dos bancos de dados mais importantes são apresentados a seguir:

- **Portal_Core**, **Portal_Ensino** e **Portal_RecursosHumanos**: estes três bancos de dados são utilizados para armazenar informações acadêmicas e gerenciais do CEULP/ULBRA. Estes bancos de dados são utilizados no sistema CGC, para apresentar informações acadêmicas no Portal Acadêmico do CEULP/ULBRA e pelo Sistema Ensino, que gerencia informações acadêmicas, como disciplinas, turmas, cursos e professores. Além disso, o banco Portal_Ensino é utilizado no sistema Intranet, que fornece, dentre outras, funcionalidades para o sistema de Atividades Semipresenciais;
- **Fsw-Conteudo**: é o banco de dados que armazena conteúdos publicados nos sites, como notícias, textos e imagens. Este banco de dados é utilizado pelo sistema CGC (Central de Gerenciamento de Conteúdo), que gerencia conteúdo de sites e gera a parte pública (de visualização) dos mesmos;

- **Fabrica_Geral**: é o banco de dados utilizado no Sistema Eventos, que gerencia inscrições em eventos acadêmicos realizados no CEULP/ULBRA.

O banco de dados Fsw-Conteudo, por representar a maior parcela de acessos ao Microsoft SQL Server, conforme informações obtidas juntamente à Fábrica de Software do CEULP/ULBRA, foi escolhido para o contexto deste trabalho e foram os dados reais. Além deste banco de dados, o consumo de recursos no servidor para gerar a página inicial do Portal Acadêmico do CEULP/ULBRA é rastreado para identificar as necessidades de otimização desta infraestrutura. Os detalhes desta metodologia são apresentados na próxima seção.

A Figura 4 apresenta o esquema do banco de dados Fsw-Conteudo.

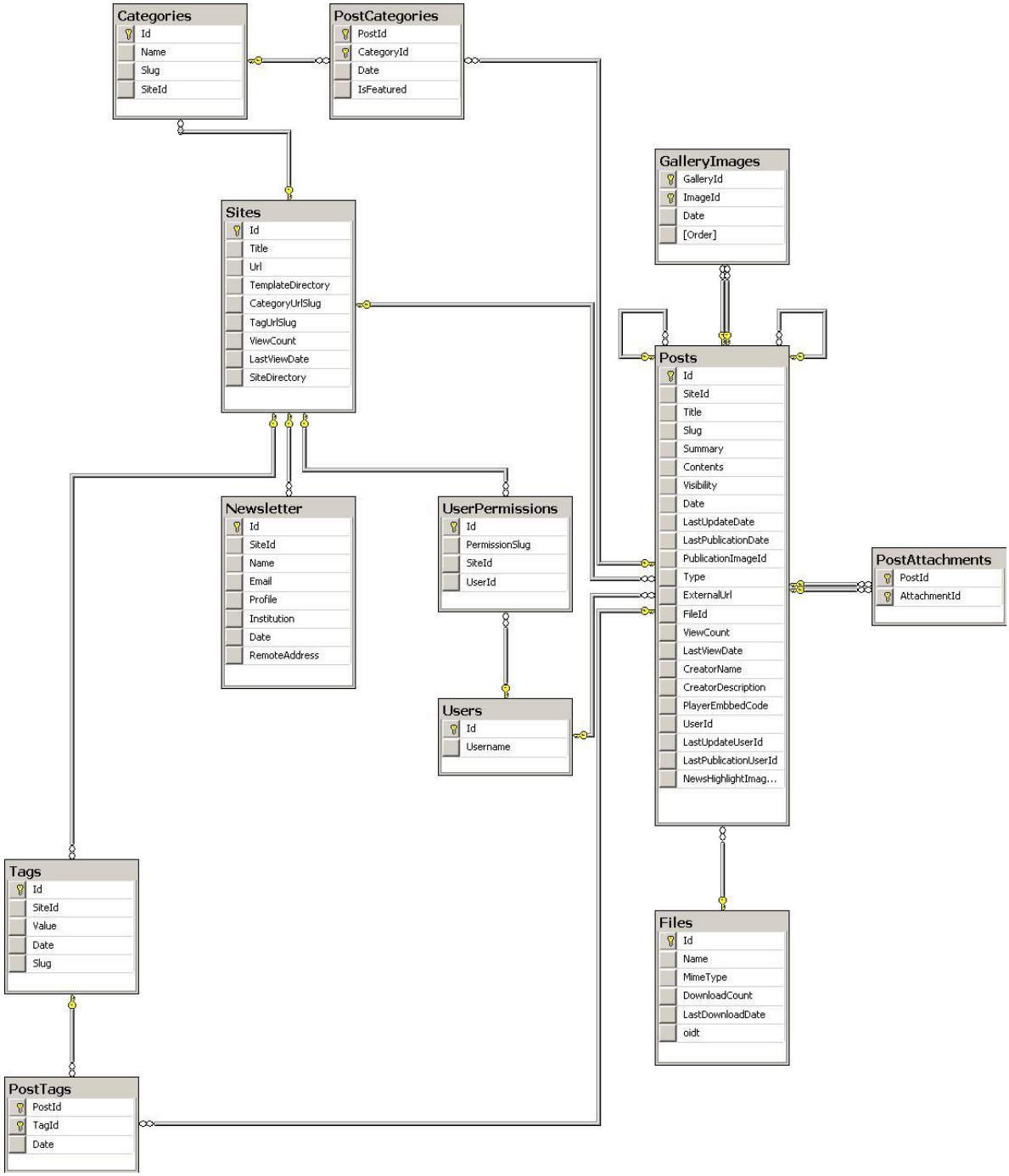


Figura 4: Esquema do Banco de Dados fsw-conteudo.

Como demonstra a Figura 4, é possível perceber os relacionamentos entre as tabelas do banco de dados Fsw-Conteudo e, também, que as tabelas “Posts” e “Sites” são fundamentais no contexto deste banco de dados, principalmente, pela quantidade de relacionamentos nos quais fornecem chaves. A tabela Posts tem uma função importante na estratégia do banco Fsw-conteudo, pois tem a função de

armazenamento do conteúdo como, por exemplo: notícias, textos, imagens e galerias.

A Tabela 10 apresenta uma breve descrição e a quantidade de registros das 5 principais tabelas do banco fsw-conteudo.

Tabela 10: Principais Tabelas do Banco fsw-conteudo.

Tabela	Descrição	N° de Registros (até 11/2012)
Posts	armazena conteúdos, que podem ser notícias, textos, imagens e galerias.	8.933
Files	armazena informações de arquivos (como os que representam as imagens).	4.815
PostCategories	representa as categorias dos conteúdos.	2.272
GalleryImages	armazena as imagens das galerias de fotos.	1.802
PostTags	armazena as tags (palavras-chave) dos conteúdos.	805

Para ilustrar a utilização da tabela “Posts”, a Figura 5 apresenta a página inicial do Portal Acadêmico do CEULP/ULBRA que contém diversas notícias institucionais, avisos e notícias do curso.

Figura 5: Página Inicial do Portal do CEULP/ULBRA.

Como pode ser visto por meio da Tabela 10 e da Figura 5, as notícias carregadas na página do Portal Acadêmico do CEULP/ULBRA são armazenados na tabela “Posts”. A quantidade de notícias armazenadas vem aumentando a cada ano e sua evolução pode ser vista no Gráfico 3.

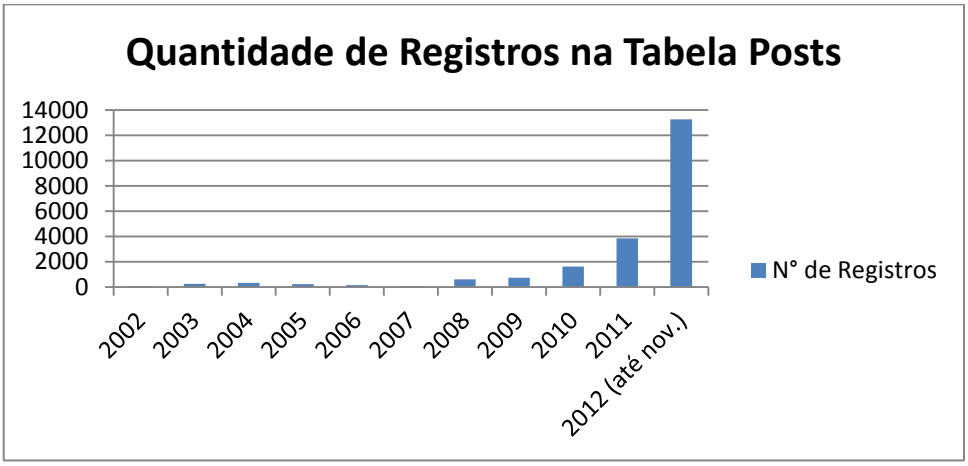


Gráfico 3: Quantidade de Registros na Tabela Posts por Ano.

A evolução da quantidade de conteúdos na tabela “Posts” reforça a importância desta tabela para a página do Portal Acadêmico do CEULP/ULBRA, pois é o local onde muitas informações são disponibilizadas para professores, funcionários, alunos e para a comunidade em geral.

3.2.4 Hardware

Para o desenvolvimento deste trabalho foram utilizados:

- parte teórica: 1 notebook HP com processador Intel i5, 2.53GHz, 4GB de RAM e disco rígido de 600 GB;
- parte prática: 2 computadores com as seguintes configurações:
  - Fenix: processador Intel Xeon 2.20 GHz (2 processadores), 8 GB de RAM e disco rígido de 408 GB;
  - Dragão: processador Intel Xeon 3.0 GHz, 4 GB RAM e disco rígido de 408 GB.

3.3 Métodos

A seção anterior demonstrou os materiais (ferramentais) utilizados na realização deste trabalho. O objetivo da seção atual é relacionar tais ferramentas com o modo como foram utilizadas, ou seja, apresentar a metodologia do trabalho. Desta forma, as etapas realizadas para este trabalho foram:

1. **Revisão de literatura**: foi feito um estudo teórico sobre as características, arquitetura, técnicas de replicação e fragmentação de dados, bem como sobre projetos de Sistemas de Banco de Dados Distribuídos;
2. **Compreensão do banco de dados fsw-conteudo**: o banco de dados fsw-conteudo foi analisado a fim de se obterem maiores informações sobre a sua estrutura, bem como a identificação de possíveis “gargalos”, ou seja, identificar os pontos de uso deste banco de dados que poderiam estar causando maior consumo de recursos do servidor;
3. **Rastreamento dos eventos SQL Server no servidor FENIX**: foi utilizada a ferramenta *SQL Server Profiler* que capturar eventos do banco fsw-conteudo na instância do SQL Server instalada no servidor FENIX;
4. **Monitoração do desempenho do servidor FENIX**: foi executado o Monitor de Desempenho do Windows para examinar como os programas executados afetam o desempenho do sistema através de contadores (exemplo: tempo de processamento, transações SQL, e número de transações);
5. **Correlação do rastreamento e monitor de desempenho**: foram realizados correlacionamentos entre os dados do rastreamento e do monitor de desempenho para verificar os períodos de consumo excessivo de recursos do servidor e, assim, encontrar as instruções SQL que estão causando este problema;
6. **Proposta de BDD para o Banco fsw-conteudo da Fábrica de Software**: foi elaborada uma proposta de BDD para banco de dados fsw-conteudo da Fábrica de Software envolvendo a inclusão de um segundo servidor, possibilitando a implantação de técnicas de replicação de dados;
7. **Implantação dos componentes do BDD proposto**: foi usada a ferramenta SQL Server Management Studio para a configuração dos componentes do BDD (publicador, distribuidor, assinante, publicação e artigo, detalhados no capítulo a seguir) para a disponibilização dos dados de modo distribuído;

8. **Testes no ambiente distribuído**: foi executado o Monitor de Desempenho do Windows para examinar o comportamento dos servidores envolvidos no ambiente distribuído;
9. **Análise comparativa dos ambientes centralizado e distribuído**: foi realizada uma análise dos ambientes centralizado e distribuído a partir dos resultados obtidos dos comportamentos dos servidores; e
10. **Elaboração das considerações finais**: foi realizado um levantamento das considerações finais a partir das atividades realizadas.

Este capítulo apresentou detalhes das ferramentas de software utilizadas neste trabalho e sobre como elas foram utilizadas, dentro de uma metodologia que tinha como objetivo final definir etapas para sistematizar o presente trabalho. No próximo capítulo serão apresentados os resultados obtidos na realização deste trabalho.

4 RESULTADOS E DISCUSSÃO

Este capítulo tem como objetivo descrever como se trabalhar com BDD no SQL Server e apresentar os resultados obtidos para as etapas definidas na seção 3.3 (exceto a revisão de literatura que se encontra no capítulo 2 e a compreensão do banco de dados fsw-conteudo que se encontra na seção 3.2.4) utilizando os materiais citados no capítulo 3.

4.1 Trabalhando com BDD no SQL Server

No Microsoft SQL Server, os bancos de dados contêm três tipos de arquivos, que são de:

- **dados primários** - todo banco tem um arquivo de dados primários que serve para armazenar dados e direcionar para outros arquivos do BD. Por convenção, estes arquivos têm a extensão .mdf;
- **dados secundários** - contém arquivos de dados, exceto os que estão no primário. Nem todo banco de dados contém arquivos de dados secundários. Por convenção, estes arquivos têm a extensão .ndf;
- **log** - contém informações de registro, muito utilizado para a recuperação de banco de dados. Por convenção, estes arquivos têm a extensão .ldf.

4.1.1 Componentes de Replicação

O SQL Server utiliza alguns componentes para implementar a replicação de dados em BDD, que são: publicador (*publisher*), distribuidor (*distribuitor*), assinante (*subscriber*), publicação (*publication*), artigo (*article*) e agente (MICROSOFT, 2012 b).

O publicador é uma instância do BD que disponibiliza dados para serem replicados em outros servidores. Cada publicador pode ter uma ou mais publicações que são um conjunto de dados, ou seja, uma unidade de replicação. Após definida uma publicação, o publicador envia este conjunto de dados para um ou mais distribuidores. O distribuidor é uma instância do banco de dados que age como um armazém para a replicação de dados associados a um ou mais publicadores. No distribuidor, cada publicador tem um banco de dados, conhecido como *distribution*, que armazena informações sobre o *status* de replicação e metadados, por exemplo.

É muito comum um servidor de BD atuar como publicador e distribuidor (MICROSOFT, 2012 b).

O assinante é um servidor SQL Server que mantém dados replicados, podendo receber dados de vários publicadores. Dependendo do tipo de replicação definido, um assinante pode ainda encaminhar aos publicadores as alterações de dados. Uma publicação é uma coleção de artigos, que podem ser tabelas, visões e *stored procedures*.

Os agentes têm o papel de copiar e distribuir dados entre distribuidores e assinantes. De modo geral, existem duas formas de configurar a atualização os dados entre publicador e assinante, que são: *pull* e *push*. A configuração é do tipo *pull* quando o agente é executado no assinante e este busca os dados do publicador. Já a configuração do tipo *push* é quando o agente é executado no distribuidor e este “empurra” as alterações para os assinantes.

4.1.2 Tipos de Replicação

O SQL Server fornece três tipos de replicação, que são: *snapshots*, *transaction* e *merge*. Na replicação *snapshots*, os dados são atualizados a cada intervalo de tempo e não há monitoração da atualização dos dados nos assinantes. Este tipo de replicação é destinado quando há uma baixa frequência de atualizações dos dados como, por exemplo, o preço de um produto que só é reajustado uma vez por ano. Na replicação *transaction*, quando ocorre uma modificação em um ou mais artigos de uma publicação, esta é capturada pelo *log* de transações e, logo após, replicada para o distribuidor. Este tipo de replicação é mais direcionado quando há necessidade de todos os servidores manterem as mesmas informações. Por fim, na replicação *merge*, quando ocorrem alterações em um dos assinantes, estas são repassadas diretamente para os demais assinantes, convergindo para uma única base de dados (MICROSOFT, 2012 c).

A partir das próximas seções, são apresentados os resultados das etapas enumeradas na seção 3.3, exceto para as etapas de revisão de literatura e compreensão do banco de dados fsw-conteudo.

4.2 Rastreamento dos Eventos SQL no Servidor FENIX

A Figura 6 mostra as propriedades selecionadas de um rastreamento criado na ferramenta SQL Server Profiler. Neste caso, foi criado um *template* em que foram capturados eventos da classe *RPC-Completed* e *SP:StmtCompleted*. A classe *RPC:Completed* indica que uma chamada de procedimento remoto foi concluída. Já a classe *SP:StmtCompleted* indica que uma instrução Transact-SQL em um *stored procedure* foi concluída.

Neste caso, os eventos capturados teriam, por exemplo, os atributos: *Duration* (tempo usado pelo evento), SPID (ID do processo do servidor atribuído pelo SQL Server ao processo associado ao cliente), *TextData* (valor de texto dependente da classe de evento capturada no rastreamento), *StartTime* (horário de início da instrução) e *EndTime* (horário de término da instrução).

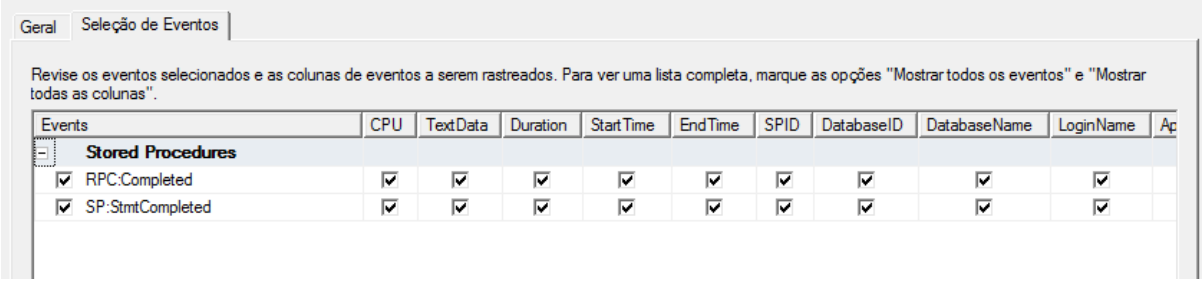


Figura 6: Configuração do Rastreamento na Ferramenta SQL Server Profiler.

Os rastreamentos foram realizados entre os dias 17 a 30 de outubro de 2012. Cada rastreamento foi salvo em um arquivo com extensão .trc. A Figura 7 mostra um dos resultados do rastreamento feito no banco fsw-conteudo.

De modo geral, pode-se observar que neste rastreamento foram capturados os eventos direcionados ao fsw-conteudo no dia 24 de outubro de 2012 (os demais resultados também usarão este dia como exemplo). Foram capturadas 157034 linhas que representam eventos das classes *RPC:Completed* e *SP:StmtCompleted*.

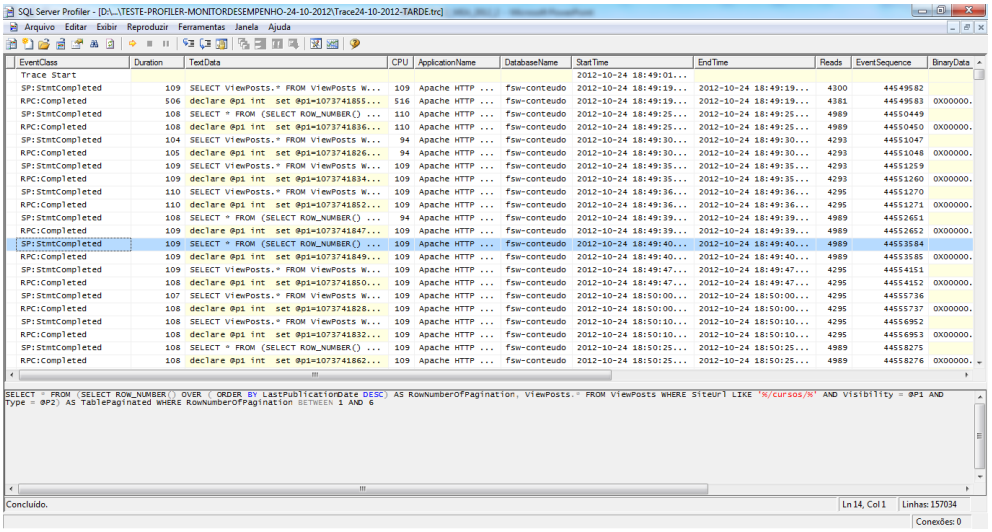


Figura 7: Resultado do Rastreamento usando o SQL Server Profiler.

Apresentado como foram feitos os rastreamentos de instruções SQL, a próxima etapa foi a monitoração do desempenho no servidor FENIX, que será descrita a seguir.

4.3 Monitoração do Desempenho no Servidor FENIX

Para verificar como os programas executados afetam o desempenho do sistema foi preciso definir os contadores desejados. Neste trabalho foram especificados os coletores tempo de processamento, transações SQL, número de transações, entre outros. Esta coleta ocorreu no mesmo período dos rastreamentos de execução de instrução SQL, vistos na seção anterior.

A Figura 8 mostra um resultado do monitor de desempenho com todos os coletores.

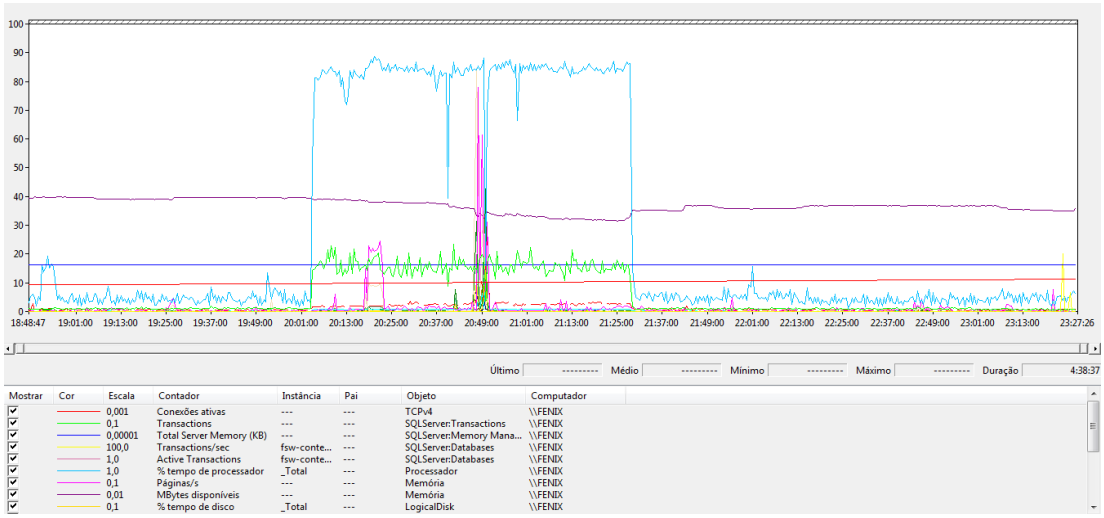


Figura 8: Resultado do Monitor de Desempenho com Todos Coletores.

Na Figura 8, pode-se verificar que alguns coletores tiveram um comportamento constante como, por exemplo, memória total do servidor do SQL (em azul), memória de disco (em roxo), transações por segundo (em amarelo) e conexões ativas (em vermelho). Por outro lado, outros coletores sofreram uma maior oscilação, tais como tempo de processador e número de transações SQL.

Inicialmente, foi feita uma análise individual dos contadores verificando o seu comportamento, ou seja, observando os períodos em que cada coletor teve seu recurso maximizado. Depois, foi analisado quais contadores estavam influenciando o tempo do processador. Verificou-se que, dentre os contadores analisados, o que se mostrou mais expressivo foi o número de transações SQL realizado. A Figura 9 mostra, mais claramente, a relação entre o tempo de processador e o número de transações SQL.

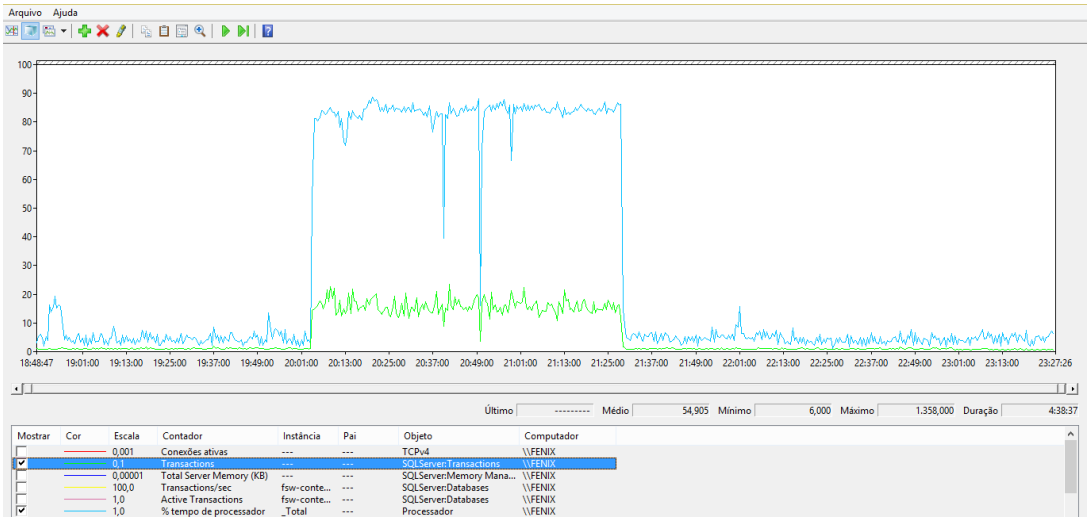


Figura 9: Tempo de Processador x Número de Transações SQL.

Como pode ser visto na Figura 9, no geral, o tempo do processador aumenta em função do número de transações SQL, principalmente, a partir das 20h (horário de “pico”). Embora este resultado seja do dia 24 de outubro, este comportamento tem se repetido outras vezes. Isto se comprova através de análise dos resultados obtidos nos outros dias de monitoramento. Cada monitoramento feito foi salvo em um arquivo com extensão .blg.

De posse dos rastreamentos das instruções SQL e dos monitoramentos de desempenho do servidor FENIX num certo período de tempo, a próxima etapa foi o relacionamento destes dados, que será apresentado na seção a seguir.

4.4 Correlação do Rastreamento e Monitor de Desempenho

A correlação dos dados do rastreamento e do monitor de desempenho serve para compreender o comportamento do servidor de acordo com as transações SQL que estavam sendo executadas, ou seja, verificar os períodos de excesso de uso do recurso do servidor e, assim, encontrar as instruções SQL que estão causando este problema. Isto é importante porque fornece indicativos de quais dados devem ser migrados de um servidor quando este se encontra sobrecarregado.

Para fazer os correlacionamentos foi preciso abrir, na ferramenta SQL Profiler, um arquivo .trc (rastreamentos capturados pelo SQL Profiler) e um arquivo .blg (dados dos coletores obtidos pelo Monitor de Desempenho) executados no mesmo período. Os correlacionamentos são compostos de 3 partes, que são: as instruções SQL capturadas, o comportamento do servidor (coletores) e a instrução executada num certo instante a partir de um seleção. A Figura 10 mostra um desses correlacionamentos e suas partes são sinalizadas pelas letras A, B e C, respectivamente.

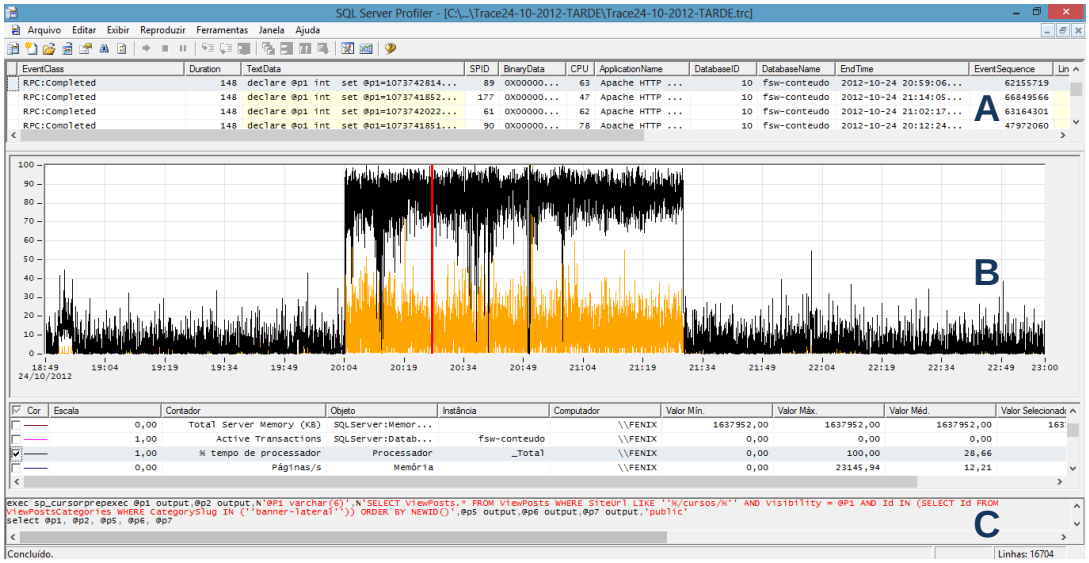


Figura 10: Correlação entre rastreamento e coletores.

Como demonstra a Figura 10, foi possível verificar na parte B que há um período “longo” (aproximadamente 90 minutos) em que os recursos (tempo de processador e número de transações) do servidor estão elevados (período de “pico”). Para identificar quais instruções estão sendo executadas neste período, o primeiro passo foi efetuar um *zoom* no gráfico referente a este período a fim de apresentar os dados mais legíveis. A Figura 11 mostra o *zoom* deste período.

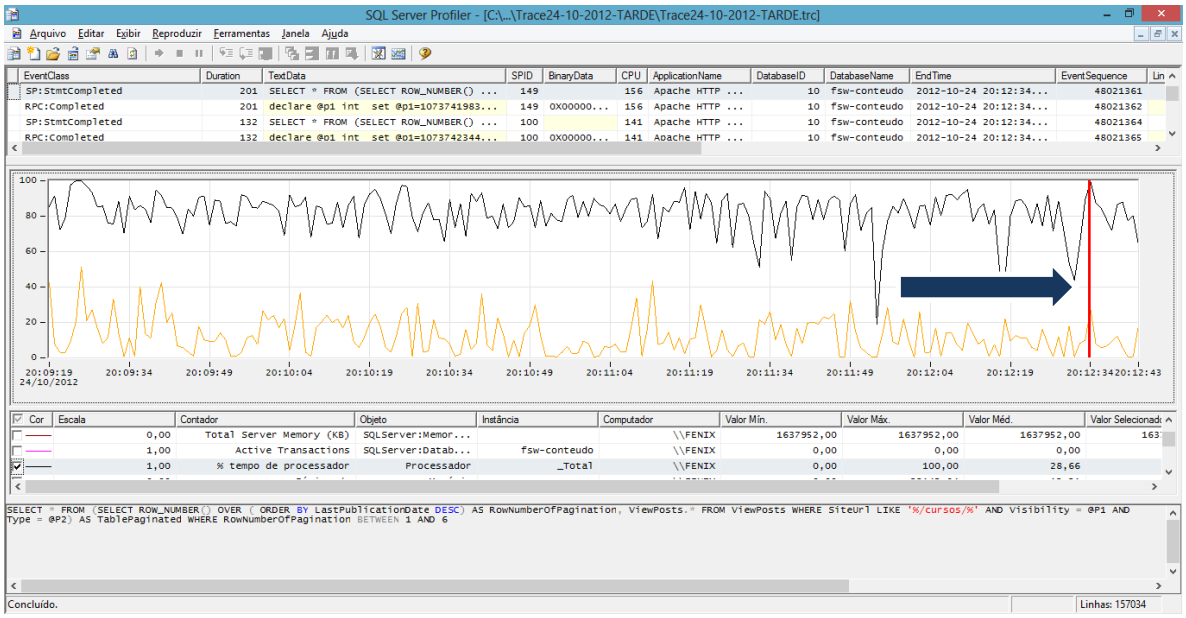


Figura 11: *Zoom* no Período de “Pico”.

Na Figura 11 a barra vertical mostrada na parte B ilustra uma situação em que 100% do recurso de tempo de processador está sendo utilizada. Nesta situação, a instrução SQL executada é apresentada na parte C, e a mesma se refere à busca das 6 primeiras notícias da página inicial do Portal Acadêmico do CEULP/ULBRA.

Com o objetivo de identificar a quantidade de instruções que estavam sendo executadas no horário de “pico” (das 20h04min até ás 21h28min), foi executada a consulta mostrada na Figura 12.

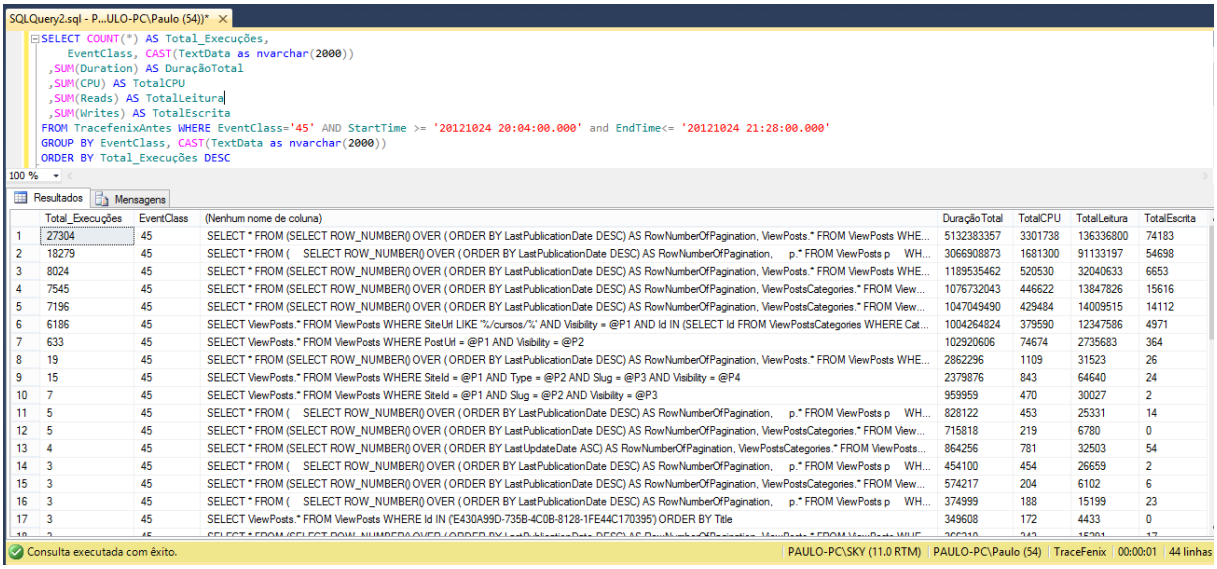


Figura 12: Consulta das Instruções Executadas no Horário de “Pico”.

De acordo com o resultado desta consulta, foram analisadas as 3 consultas mais executadas no servidor FENIX. Na Tabela 11 são apresentadas por completo estas 3 consultas, bem como a sua finalidade.

Tabela 11: Descrição das 3 Consultas Mais Executadas.

N°.	Qtd. de execuções	Consulta	Finalidade
1	27.304	`SELECT * FROM (SELECT ROW_NUMBER() OVER (ORDER BY LastPublicationDate DESC)AS RowNumberOfPagination,ViewPosts.* FROM ViewPosts WHERE SiteUrl LIKE '%/cursos/%'AND Visibility = @P1 AND Type = @P2)AS TablePaginated WHERE RowNumberOfPagination BETWEEN 1 AND 6`	Carregamento das notícias do curso da página inicial do Portal.
2	18279	`SELECT * FROM (SELECT ROW_NUMBER() OVER (ORDER BY LastPublicationDate DESC) AS RowNumberOfPagination, p.* FROM ViewPosts p`	Carregamento das notícias destaque da página inicial do

		`WHERE p.SiteId = '7CE4F71D-AD1F-45B1-979F-19668BF81797' AND Type = 'news' AND Visibility = 'public' AND p.Id NOT IN (select pc.postid from PostCategories pc inner join Categories c on pc.CategoryId = c.Id where p.Id = pc.PostId and c.Slug IN ('destaque-noticia'))) AS TablePaginated WHERE RowNumberOfPagination BETWEEN 1 AND 6`	Portal.
3	8024	`SELECT * FROM (SELECT ROW_NUMBER() OVER (ORDER BY LastPublicationDate DESC) AS RowNumberOfPagination, ViewPosts.* FROM ViewPosts WHERE SiteId = @P1 AND Visibility = @P2 AND Id IN (SELECT Id FROM ViewPostsCategories WHERE CategorySlug IN ('aviso'))) AS TablePaginated WHERE RowNumberOfPagination BETWEEN 1 AND 1`	Carregamento do aviso da página inicial do Portal.

A Figura 13 mostra a página inicial do Portal, destacando a localização do resultado das três consultas mais frequente. A primeira é relacionada com as notícias do curso, a segunda com as notícias destaque e a terceira com os avisos.

Figura 13: Página Inicial do Portal do CEULP/ULBRA.

Até este momento buscou-se informações sobre que tipo de instrução SQL tem afetado o desempenho do servidor FENIX e como o banco de dados fsw-conteudo está sendo utilizado no ambiente centralizado, pois são de grande valia

para a elaboração da proposta de BDD para a Fábrica de Software, que será apresentada a seguir.

4.5 Proposta de BDD para o Banco fsw-conteudo da Fábrica de Software

A partir das demandas da Fábrica de Software e das especificidades da página inicial do Portal, foi elaborada uma proposta de BDD para o banco fsw-conteudo. Nesta proposta buscou-se atender os seguintes aspectos:

- **diminuir o processamento elevado do servidor FENIX** - conforme mostrado na seção 3.2.4, constatou-se que o servidor FENIX é responsável por todo o processamento tanto de banco de dados quanto de aplicação. Assim, é preciso diminuir esta carga de processamento e uma alternativa seria a inclusão de outro servidor para realizar parte do processamento;
- **disponibilizar os dados mais acessados em um servidor dedicado** - como a página inicial do Portal contém os dados mais acessados (o que gera o maior fluxo), é conveniente disponibilizá-los em um servidor dedicado e, assim, reduziria o processamento realizado no FENIX;
- **tornar o sistema escalável** – considerando que o cenário atual é centralizado e que há um crescimento elevado dos dados (considerando a tabela Posts os valores podem ser vistos no Gráfico 3), é importante criar um sistema que tenha condições de assimilar um carga crescente.

Desta forma, o primeiro item da proposta tem relação com a infraestrutura de servidores disponíveis. Sugere-se a inclusão de um segundo servidor, já disponível na Fábrica de Software, identificado por “DRAGÃO”. Em resumo, um computador atua como Publicador/Distribuidor (FENIX) e, outro, como Assinante (DRAGÃO), conforme mostra a Figura 14.

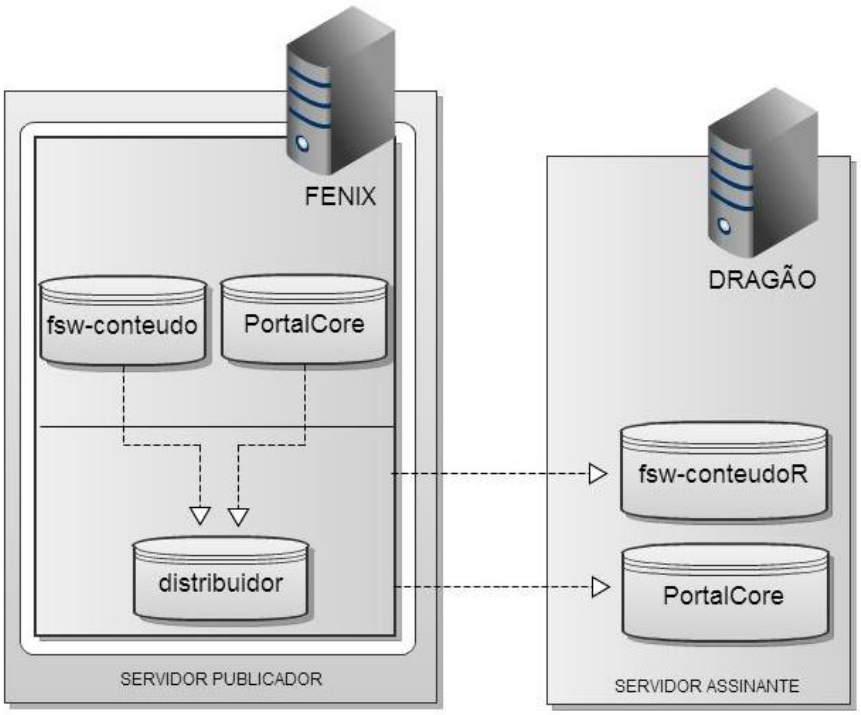


Figura 14: Proposta de BDD para o Banco fsw-conteudo da Fábrica de Software.

Nesta proposta, o servidor publicador atua também como distribuidor, dispensando, assim, necessidade de inclusão de outro servidor. Este servidor mantém a cópia original dos dados dos bancos fsw-conteudo e PortalCore e envia o conjunto de dados a ser replicado para o distribuidor, que faz o envio dos dados para o assinante. Assim, servidor assinante possui os bancos fsw-conteudoR (armazena uma réplica dos dados do fsw-conteudo) e PortalCore (armazena uma réplica dos dados do PortalCore). O banco de dados PortalCore também precisa ser replicado (tabelas Usuario e Pessoa) porque possuem informações acadêmicas e gerenciais do CEULP/ULBRA.

O servidor assinante recebe ainda atualizações periódicas do distribuidor (replicação assíncrona). Desta forma, as consultas à página inicial do portal serão carregadas com as réplicas dos dados, localizadas no servidor DRAGÃO. No entanto, quando o usuário for cadastrar (CRUD) uma notícia ou buscar por notícias antigas, este fará uso do banco fsw-conteudo no servidor FENIX.

Esta proposta utiliza a estratégia descendente, pois o esquema conceitual local, DRAGÃO, está associado à parte da base de dados original, criados a partir de técnicas de fragmentação e alocação. Assim, parte do banco fsw-conteudo e

PortalCore ficariam replicadas no servidor DRAGÃO, sendo estes dados os mais acessados que compõem a página inicial do Portal (fragmentação horizontal).

4.6 Implantação dos Componentes do BDD Proposto

É importante ressaltar que antes de qualquer configuração de um ambiente de banco de dados distribuído, no SQL Server, foi necessário confirmar se as configurações de rede do SQL Server protocolo TCP/IP estão habilitadas. Este procedimento deve ser realizado através do *SQL Server Configuration Manager*.

A criação e configuração dos componentes envolvidos nesta proposta são apresentadas a seguir.

4.6.1 Criação de uma Publicação

A partir do recurso de replicação, disponível no *Object Explorer* no *Management Studio*, foi possível criar uma nova publicação. Para que o publicador (FENIX), para o qual se pretende criar uma publicação, atuasse também como distribuidor, foi preciso especificar esta função. Em seguida, foi especificado o banco de dados que se pretende replicar (fsw-conteudo), conforme mostra a Figura 15.

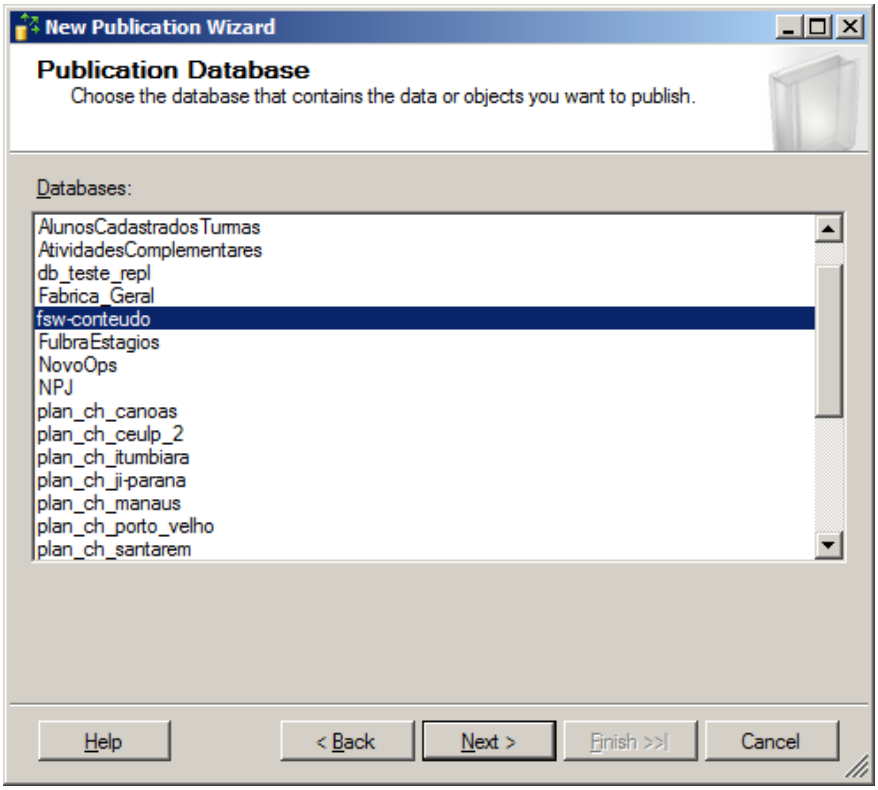


Figura 15: Seleção do Banco de Dados fsw-conteudo.

Definido o banco de dados, foi preciso especificar a forma de replicação dos dados. Com relação ao tipo de replicação, o tipo *transaction* foi escolhido e o tempo de sincronização ficou estipulado em 5 minutos. A escolha do tipo de replicação pode ser vista na Figura 16.

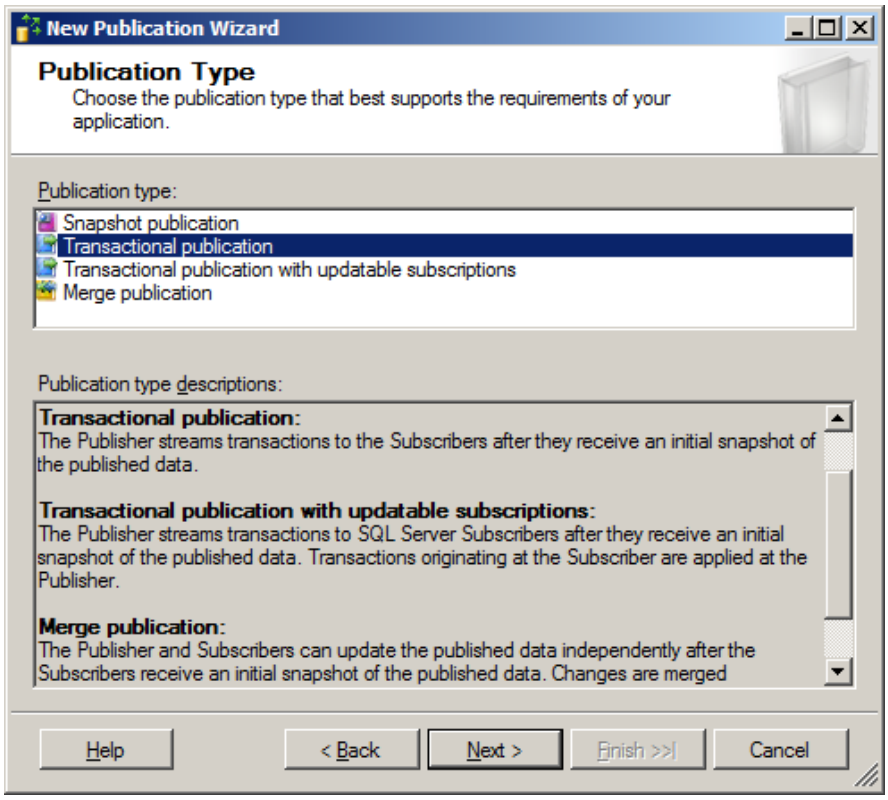


Figura 16: Seleção do Tipo de Replicação Transacional.

Na sequência, foi preciso especificar o(s) artigo(s) para esta publicação. Neste caso, foram selecionadas 13 tabelas com todos os seus atributos, conforme pode ser visto na Figura 17.

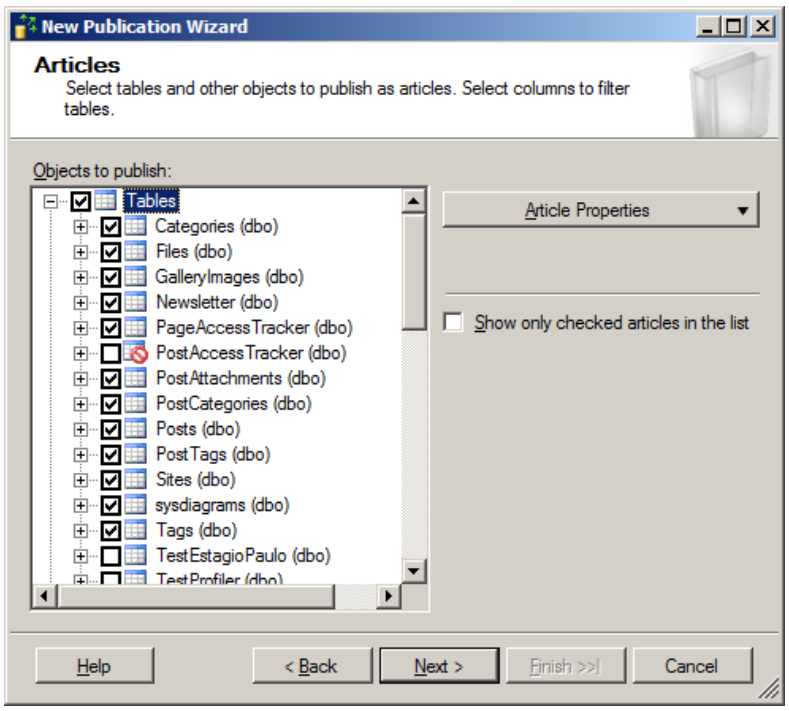


Figura 17: Tabelas Selecionadas do fsw-conteudo para Publicação.

O próximo passo foi adicionar um filtro na tabela Posts com propósito de particionar os dados horizontalmente, replicando assim somente os dados referentes ao semestre corrente, uma vez que esta tabela possui dados desde 2002 (conforme mostrado Tabela 9). A Figura 18 mostra a configuração deste filtro em que foi atribuído o critério "`WHERE [LastPublicationDate] > (GETDATE() - 183)`".

Figura 18: Definição de Filtro para a Tabela Posts.

Realizada a configuração desta publicação, os mesmos passos foram executados para o banco de dados PortalCore, exceto o passo da aplicação do filtro porque este banco contém apenas dados de autenticação de usuário. A replicação foi parcial, pois somente as tabelas Usuario e Pessoa foram replicadas. O próximo passo foi configurar um assinante.

4.6.2 Criação de um Assinante

De forma semelhante, a partir do recurso de replicação, disponível no *Object Explorer*, foi possível criar um novo assinante. A Figura 19 mostra a seleção da publicação Publicador_fsw-conteudo do publicador localizado no servidor FENIX.

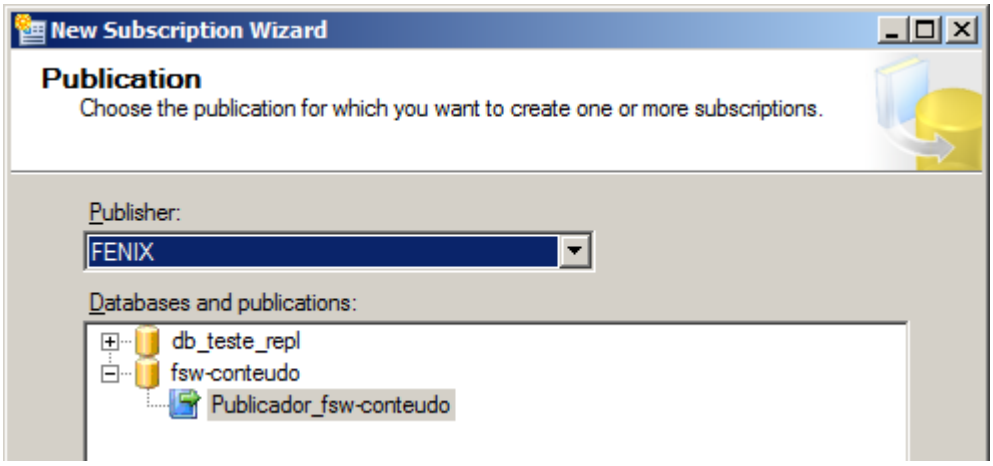


Figura 19: Seleção de Publicador_fsw-conteudo para o Assinante.

O próximo passo foi a escolha de onde será executado o agente de distribuição. A Figura 20 mostra que o agente será executado no servidor FENIX, tendo assim a forma de atualização das alterações do tipo *push*, ou seja, o distribuidor fica encarregado de enviar ao servidor DRAGÃO as alterações ocorridas.

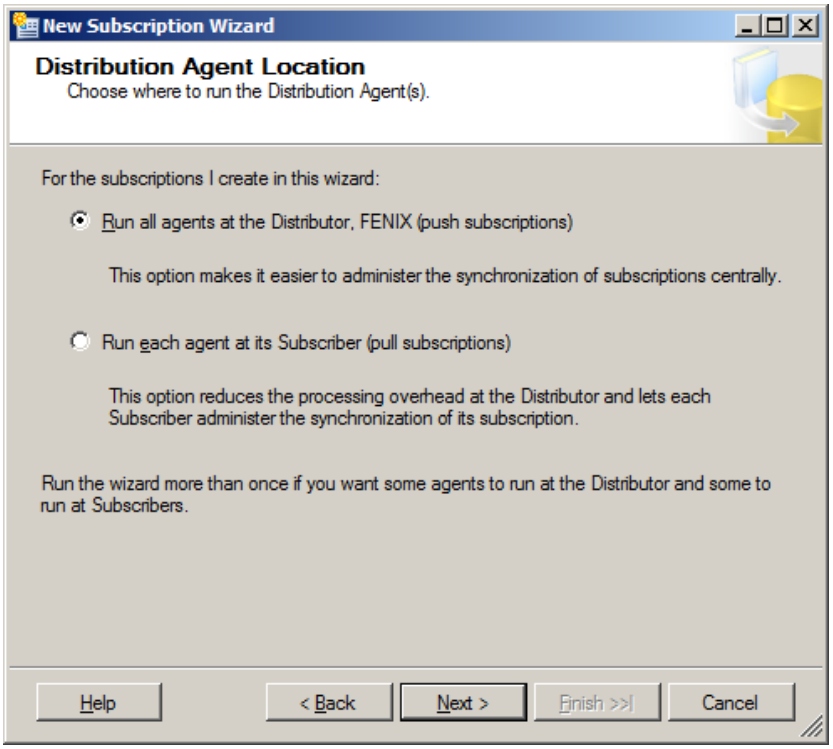


Figura 20: Especificação do Local do Agente de Distribuição.

Este tipo de configuração foi considerado mais adequado ao cenário em questão, pois o servidor FENIX tem o maior poder de processamento se comparado ao servidor DRAGÃO. Além disso, deixando esta tarefa no servidor FENIX, outros assinantes podem ser inseridos na arquitetura sem a necessidade de alterar a configuração pré-definida.

A Figura 21 apresenta a escolha do servidor assinante (DRAGÃO), bem como o banco de dados (fsw-conteudoR) que receberá as tabelas selecionadas na Figura 17 para replicação.

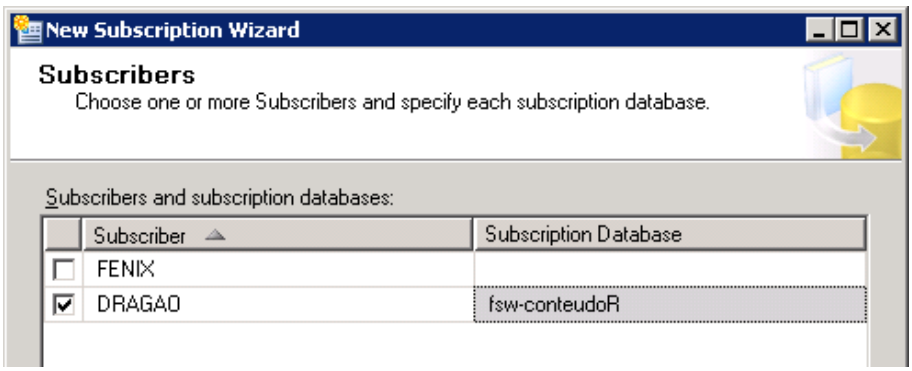


Figura 21: Definição do Assinante e Banco de Dados.

Em seguida, foi configurado o cronograma de sincronização do agente executado no servidor FENIX (distribuidor), conforme mostra a Figura 22. A sincronização foi estipulada a cada 5 minutos porque o sistema suporta este tempo de inconsistência.

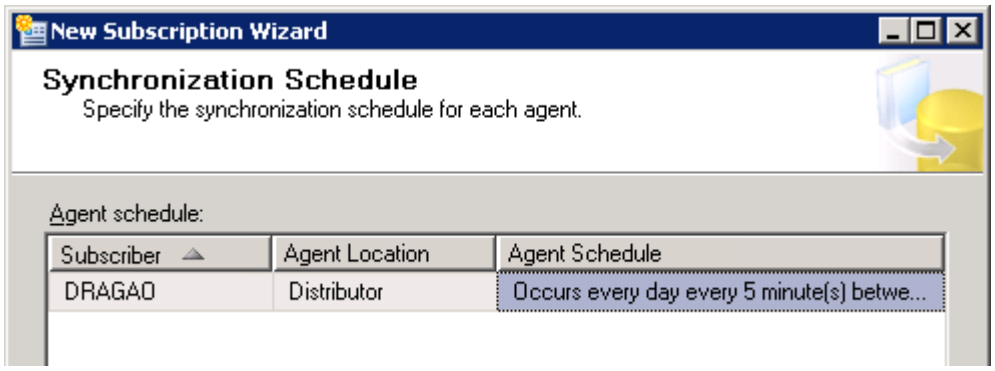


Figura 22: Configuração do Cronograma do Agente.

Por fim, a Figura 23 mostra o resumo da configuração feita no distribuidor para um novo assinante, localizada em DRAGÃO, obtendo êxito como status.

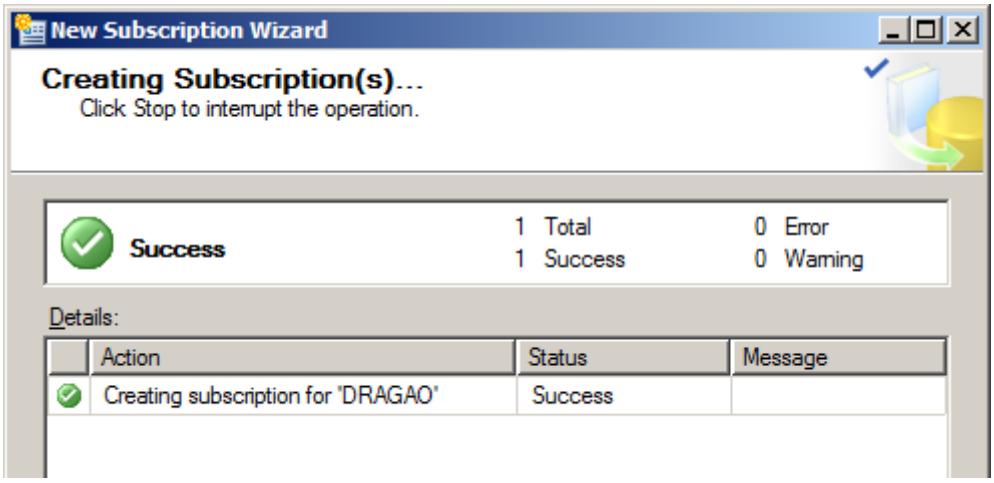


Figura 23: Resumo da Configuração do Novo Assinante.

Após a realização destes passos, os dados já estavam disponíveis em um ambiente distribuído. Para acessar estes dados distribuídos pela página inicial do Portal, foi preciso apenas redirecionar a conexão de string da aplicação para o servidor DRAGÃO. Assim, a partir do dia 10 de novembro de 2012 o carregamento da página inicial do Portal está sendo realizado com os dados replicados no DRAGÂO.

A seção a seguir mostra os testes realizados no ambiente distribuído, tanto para o servidor FENIX quanto para o servidor DRAGÃO.

4.7 Testes no Ambiente Distribuído

A fim de verificar o comportamento do processamento do servidor FENIX após a inclusão da máquina DRAGÃO para o armazenamento dos dados parciais do fsw-conteudo e PortalCore, foram realizados rastreamentos no servidor FENIX, bem como a monitoração do seu desempenho. Os rastreamentos no ambiente foram realizados durante o período de 10 a 20 de novembro deste ano, principalmente, nos horários considerados de “pico”, ou seja, em que os recursos estavam sendo mais demandados.

A Figura 24 mostra um resultado obtido do rastreamento e monitoramento do banco fsw-conteudo no servidor FENIX no dia 16 de novembro de 2012, ressaltando os coletores tempo de processador e número de transações SQL.

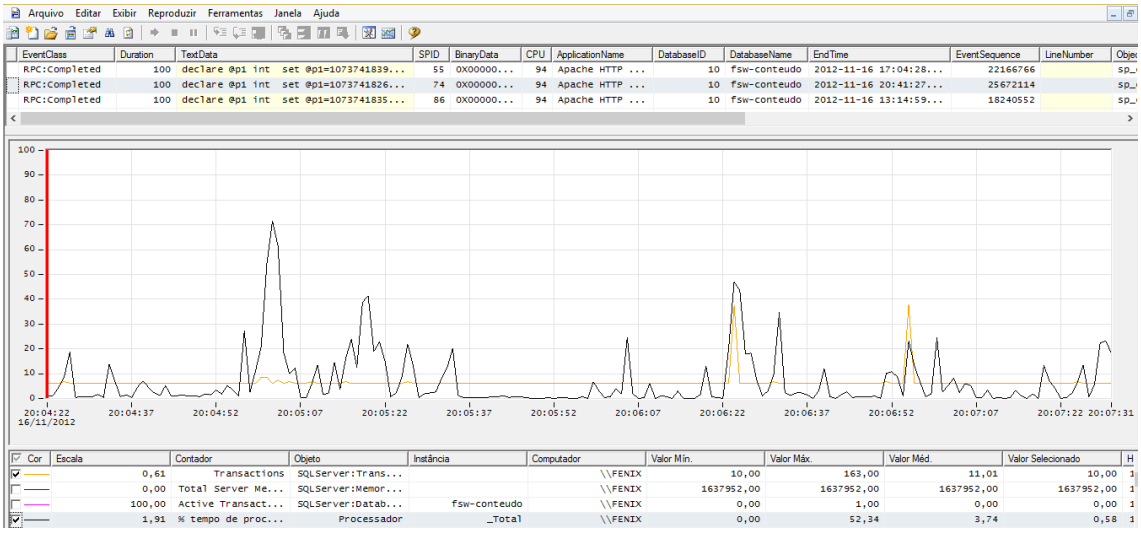


Figura 24: Tempo de Processador x N° de Transações no FENIX – Pós-replicação.

Como indica a Figura 24, observando o impacto do número de transações SQL sobre o tempo de processador, foi verificado que houve uma diminuição tanto das transações SQL quanto do tempo de processador, diminuindo, assim, o consumo de servidor FENIX. Na média, o tempo de processador deste servidor ficou em torno de 30%, pouco ocupado.

Com o intuito de conhecer as instruções que mais estavam sendo processadas no servidor FENIX, foi executada a consulta mostrada na Figura 25.

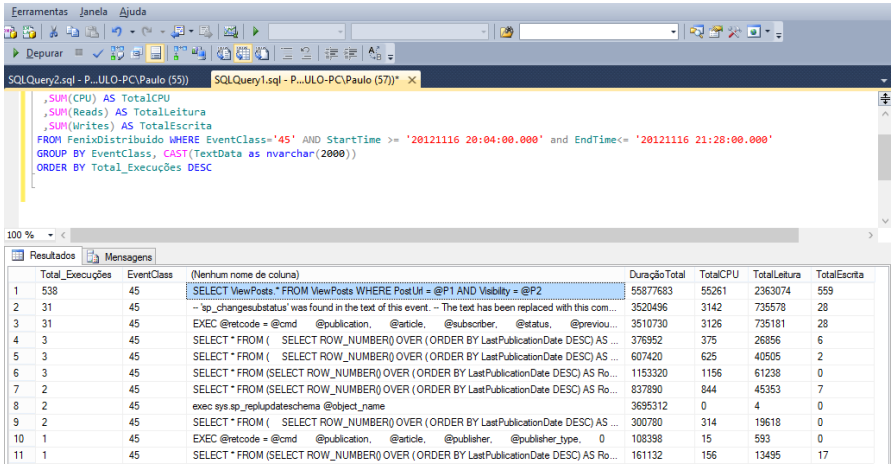


Figura 25: Consulta das Instruções no Horário de “Pico” no FENIX – Pós-replicação.

Pode-se observar que a consulta mais processada, 538 vezes, era relacionada a uma busca de uma publicação especifica pela url.

De forma similar, foi realizado um rastreamento e monitoramento no servidor assinante (DRAGÃO). A Figura 26 mostra um resultado ampliado do desempenho do servidor DRAGÂO no que diz respeito ao tempo de processador e número de transações SQL.

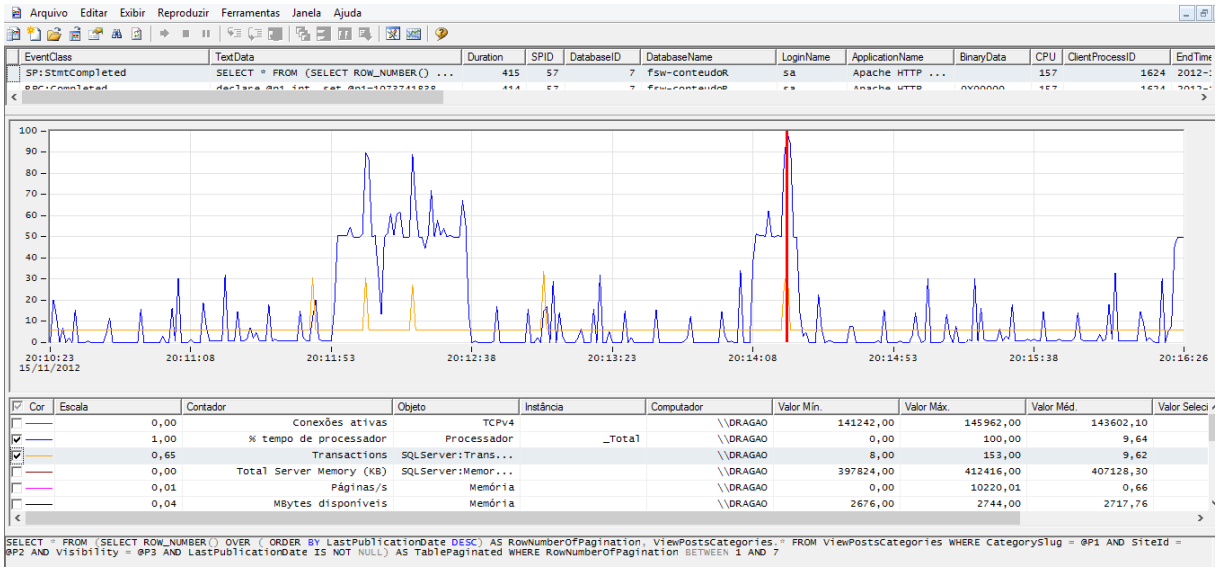
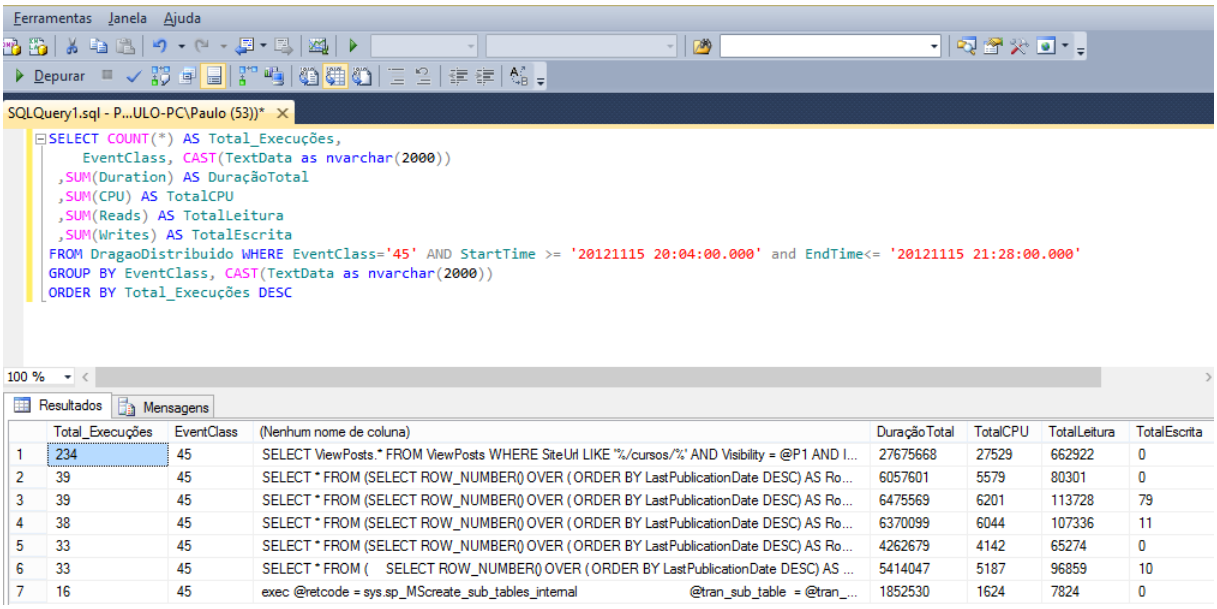


Figura 26: Tempo de Processador x N° de Transações no DRAGÃO – Pós-replicação.

Neste resultado, foi detectado que o recurso de tempo de processador oscilou bastante, havendo vários momentos em que o recurso foi muito consumido. Mas, na

média, este recurso foi consumido em torno de 40%. A Figura 27 mostra a consulta realizada para a identificação das consultas mais processadas no servidor DRAGÃO após a distribuição dos dados.



	Total_Execuções	EventClass	(Nenhum nome de coluna)	DuraçãoTotal	TotalCPU	TotalLeitura	TotalEscrita
1	234	45	SELECT ViewPosts.* FROM ViewPosts WHERE SiteUrl LIKE '%/cursos/%' AND Visibility = @P1 AND I...	27675668	27529	662922	0
2	39	45	SELECT * FROM (SELECT ROW_NUMBER() OVER (ORDER BY LastPublicationDate DESC) AS Ro...	6057601	5579	80301	0
3	39	45	SELECT * FROM (SELECT ROW_NUMBER() OVER (ORDER BY LastPublicationDate DESC) AS Ro...	6475569	6201	113728	79
4	38	45	SELECT * FROM (SELECT ROW_NUMBER() OVER (ORDER BY LastPublicationDate DESC) AS Ro...	6370099	6044	107336	11
5	33	45	SELECT * FROM (SELECT ROW_NUMBER() OVER (ORDER BY LastPublicationDate DESC) AS Ro...	4262679	4142	65274	0
6	33	45	SELECT * FROM (SELECT ROW_NUMBER() OVER (ORDER BY LastPublicationDate DESC) AS ...	5414047	5187	96859	10
7	16	45	exec @retcode = sys.sp_MScreate_sub_tables_internal @tran_sub_table = @tran_...	1852530	1624	7824	0

Figura 27: Consulta das Instruções no Horário de “Pico” no DRAGÃO – Pós-replicação.

Como somente os dados mais recentes do banco fsw-conteudo estão no servidor DRAGÃO para o carregamento da página inicial do Portal, as consultas mais processadas estão relacionadas com este tipo de dado. Neste resultado, houve 39 consultas processadas para carregar as notícias do curso na página inicial do Portal (item 3 do resultado da consulta).

A seguir será apresentada uma análise comparativa entre os ambientes centralizado e distribuído para o banco de dados fsw-conteudo.

4.8 Análise Comparativa dos Ambientes Centralizado e Distribuído

A partir dos resultados obtidos das transações SQL ao banco de dados fsw-conteudo no cenário centralizado e, depois, da implantação da proposta do BBD para o banco de dados fsw-conteudo, foi possível observar que:

- no ambiente centralizado – o banco de dados fsw-conteudo é menos complexo devido todos os dados estarem armazenados em um único local e sem a necessidade de atualizar outros dados remotamente. Por outro lado,

este ambiente mostra uma fragilidade, pois se o servidor FENIX ficar indisponível, os dados do banco fsw-conteudo ficarão sem acesso. Além disso, como existem vários bancos de dados armazenados neste servidor, este fica sobrecarregado para processamento de consultas; e

- no ambiente distribuído – há uma maior disponibilidade dos dados visto que servidor DRAGÃO possui uma réplica dos dados armazenados no FENIX. Além disso, outros assinantes podem ser inseridos no sistema de BDD sem afetar a estrutura criada, isto é, escalável. Assim, se o DRAGÃO estiver inoperante, o carregamento da página inicial do Portal poderá ser redirecionado para outro assinante ou o próprio servidor FENIX. Com relação ao custo, o ambiente distribuído necessitou de apenas outro servidor. Para o controle de atualização, este ambiente se mostrou eficiente, pois há necessidade de atualizar apenas um servidor a cada 5 minutos, não demandando alto custo computacional. Esse ambiente pode ser considerado favorável à consultas distribuídas, visto que este pode disponibilizar dados que sejam relevantes para outras aplicações.

Portanto, com o ambiente distribuído, o benefício é notório para o banco de dados da Fábrica de Software. Tendo como base o dia 15/11/2012, em que ocorreram 1049 execuções no servidor, se o ambiente fosse centralizado, somente a máquina FENIX teria que executar todo o processamento. Com o ambiente distribuído, o número de execuções da máquina FENIX seria de 617 e a máquina DRAGÃO 432. Neste caso, há um balanceamento das execuções.

5 CONSIDERAÇÕES FINAIS

Foi observado que a tecnologia de banco de dados distribuído não é algo trivial, pois envolve a definição de vários parâmetros como a fragmentação, alocação e replicação de dados. Além disso, há um custo para a inserção desta tecnologia como, por exemplo, a aquisição de equipamentos e a contratação de profissionais capacitados no monitoramento em ambientes distribuídos. No entanto, essa tecnologia tem se mostrado ser uma boa solução para duplicação dos dados, deixando os mesmos disponíveis em vários servidores ao mesmo tempo (maior disponibilidade no caso de instabilidade ou pane em algum servidor que faz parte do conjunto de distribuição).

No caso da Fábrica de Software, constatou-se, por meio de rastreamentos de instruções SQL e monitoramentos do desempenho do servidor FENIX, que os dados do carregamento de notícias da página inicial do Portal Acadêmico do CEULP/ULBRA, armazenados no banco fsw-conteudo, são muito consultados. Isto tem afetado o desempenho deste servidor, consumindo em diversos momentos muitos recursos, principalmente, tempo de processador. No cenário centralizado, embora o carregamento da página inicial do Portal estivesse funcionando, havia uma fragilidade quanto à disponibilidade dos dados no caso do servidor FENIX ficar indisponível.

A proposta do BDD para a Fábrica de Software apresenta várias vantagens em relação ao centralizado, como por exemplo a replicação de dados em mais de um nó, proporcionando assim mais segurança e disponibilidade das informações. Além disso, a proposta suporte que novos nós sejam adicionados sem afetar o sistema (escalável).

A estrutura de distribuição implantada para a Fábrica de Software está funcionando bem, conforme a proposta definida. Com o uso da ferramenta *SQL Server Management Studio*, foi possível configurar um ambiente distribuído (2 computadores) através de componentes de publicador, distribuidor, assinante, publicação e artigo. O servidor FENIX ainda fica responsável pelo armazenamento de todos os dados, mas os dados do carregamento mais acessados encontram-se armazenados no servidor DRAGÃO.

Como trabalhos futuros, sugere-se:

- análise de dados mais acessados dos bancos armazenados no servidor Fenix para replicação e fragmentação;
- análise de desempenho dos assinantes em relação à replicação; e
- estudo estatístico da necessidade de ampliação da estrutura de BDD implantada.

REFERÊNCIAS BIBLIOGRÁFICAS

BARRAZA, Omar. Achieving 99.9998+% Storage Uptime and Availability. Dot Hill Systems Corp. Disponível em <http://www.dothill.com/assets/pdfs/5-9s_wp.pdf>. Acesso em 21/09/2011.

DATE, C. J. **Introdução a Sistemas de Bancos de Dados**. Rio de Janeiro: Campus, 1991.

ELMASRI, R.; NAVATHE, S. B. **Sistemas de Banco de Dados**. 6a ed., Pearson-Addison-Wesley, 2011.

KARLAPALEM, K., NAVATHE, S., MORSI, M., "Issues in Distribution Design of Object Oriented Databases", In: Distributed Object Management, Morgan Kaufman Publishers, pp. 148-164. 1994.

MICROSOFT. Referência do *SQL Server Profiler* 2012a Disponível em < http://msdn.microsoft.com/pt-br/library/ms173757(v=sql.105).aspx> Acesso em 11/06/2012.

MICROSOFT. Replication Publishing Model Overview 2012b Disponível em < http://msdn.microsoft.com/en-us/library/ms152567.aspx> Acesso em 14/06/2012.

MICROSOFT. Types of Replication 2012c Disponível em < http://msdn.microsoft.com/en-us/library/ms152531.aspx> Acesso em 14/06/2012.

ÖZSU, M.T. VALDURIEZ, P. **Principles of Distributed Database Systems.** 3a ed., New York, Springer, 2011.

SILBERSCHATZ, A.; KORTH, H. F.; SUDARSHAN, S. **Sistema de Banco de Dados** 5ª ed., Editora Campus, 2006.