1
Introdução

Nos últimos anos, tem-se observado um crescente interesse em aplicações Web semânticas, que usam o modelo de dados RDF[1] [Manola e Miller, 2004] como camada de informação. Além disso, há cada vez mais referências entre aplicações espalhadas por toda WWW, estimuladas, especialmente, pela iniciativa *Linking Open Data* [LOD Project].

O surgimento de grandes repositórios de dados RDF (em inglês, *triple stores*) apresenta-se como uma excelente oportunidade para investigação de como usar, de forma adequada, este novo modelo de dados em larga escala.

Assim como aplicações Web convencionais, aplicações Web semânticas também apresentam forte comportamento transacional, ou seja, dados RDF compartilhados são lidos ou alterados por sessões concorrentes, estando, portanto, sujeitos a condições de corridas, caso estes acessos não sejam apropriadamente isolados. Um mecanismo de controle de concorrência para isolamento de transações que consultam e alteram triplas RDF é de vital importância.

Neste novo cenário, naturalmente aparecem algumas indagações: o modelo de bloqueio tradicional, utilizado em banco de dados relacionais, pode ser reutilizado? Novos modelos de bloqueio são necessários? Estamos diante de uma oportunidade de aprimorar os modelos de bloqueio existentes? A abordagem otimista é a única solução viável? Ou existe uma opção pessimista interessante para o modelo RDF?

Até o presente momento, a grande maioria das iniciativas aborda apenas consultas em repositórios RDF ou, quando oferecem suporte a transações de escrita, proveem isolamento ou baseado no protocolo otimista ou baseado em *snapshots*.

Como sabemos, na abordagem otimista não existe emprego de bloqueios. O que se faz é uma verificação de versão, ao final da transação, para saber se houve conflito, ou seja, se o dado manipulado foi modificado por uma segunda transação concorrente. Caso ocorra um conflito, em geral, a transação é

[1] RDF - *Resource Description Framework*

abortada e o trabalho é perdido. Trata-se de um protocolo de *detecção* de conflitos. Há situações onde esta perda do trabalho é inadmissível. Por exemplo, em uma loja comercial é intoleravelmente frustrante para o usuário levar horas editando um pedido de venda e ao final, ao confirmar suas edições, receber um aviso que o pedido foi alterado por uma sessão concorrente e que terá que editar tudo novamente. Nestas situações, onde as consequências de retrabalho são severas ou quando a probabilidade de conflito é alta, torna-se necessário uma abordagem pessimista, de *prevenção* de conflitos, que garanta que o usuário esteja trabalhando com a última versão do dado e que nenhuma outra transação concorrente possa usar o dado.

A principal motivação para este trabalho é justamente a falta de um modelo pessimista de bloqueio adequado para o modelo RDF. Um modelo que, onde possível, explore a estrutura de triplas do RDF para melhorar o nível de multiprogramação em comparação com os modelos de bloqueio tradicionais. E, obviamente, garanta a prevenção dos quatro problemas clássicos de concorrência, a saber: *lost updates, dirty reads, non-repeatable reads e phantoms read*s.

Este trabalho propõe um modelo pessimista de bloqueio, multigranular, para RDF, oferecendo quatro grânulos "hierarquicamente" relacionados, seis novos tipos de bloqueio e, por fim, um protocolo de aquisição destes bloqueios que é uma adaptação do protocolo multigranular convencional [Gray et al., 1976].

1.1. Objetivos

O objetivo principal desta pesquisa é definir uma modelo pessimista de bloqueio para o modelo de dados RDF.

Adicionalmente, visa-se prover a infraestrutura para o uso efetivo do modelo de bloqueio: uma pequena ontologia que facilite a aquisição dos bloqueios pelas transações e uma discussão de como poder-se-ia aplicar a abordagem otimista, de forma complementar ao modelo pessimista proposto.

Finalmente, apresenta-se também a definição de uma DSL[2] interna [Fowler e Parsons, 2011] para especificação (executável) de *workflow* de transações Web em geral (escrita na linguagem Ruby [Flanagan e Matsumoto, 2008] e

[2] DSL - *Domain Specific Language*

baseada em modelo de transições de estado). Uma transação Web nada mais é do que um "caso de uso" transacional em uma aplicação Web, onde a interação do usuário é pré-definida por um *workflow*, até atingir um objetivo. Ao longo de uma transação Web ocorre uma sequencia de leituras de dados e uma única atualização ao final (espécie de *commit*) que garante que, em caso de sucesso, todos os efeitos da transação Web sejam persistidos de uma só vez (atomicidade).

1.2. Contribuições

Esta tese apresenta as seguintes contribuições para o gerenciamento de transações em RDF:

- Protocolo de bloqueio pessimista multigranular para RDF
  - Quatro grânulos "hierarquicamente" relacionados;
  - Novos tipos de bloqueio criados, aproveitando as particularidades do grafo RDF, para aumentar o grau de multiprogramação;
  - *Lock Manager*, implementado em Ruby e avaliado por simulação, oferecendo dois modos de operação: multigranular (MGL) ou monogranular (NO_MGL);
  - Ontologia e uma proposta de uso do SPARQL Construct [Prud'hommeaux e Seaborne, 2008] para facilitar a aquisição de bloqueios por parte das transações.

- Discussão sobre como empregar a abordagem otimista, de forma complementar ao protocolo pessimista proposto, para RDF
  - Apresentação de uma possível solução por meio do uso de CBD - *Concise Bounded Description* [Sticker, 2005], que é um subgrafo RDF representativo de um recurso.

- Padrão arquitetural para transações Web
  - Proposta de reificação de transações Web, ou seja, representar a transação Web como um objeto de primeira ordem, que se comporta e pode ser tratado como qualquer outro objeto de domínio;

- DSL interna escrita na linguagem Ruby, baseada em modelo de transições de estado, para especificação (executável) de *workflow* de transações Web. Especificação executável significa que a DSL, ao ser interpretada pelo Ruby, gera, via metaprogramação [Perrota, 2010], em tempo de execução, a máquina de estados do *workflow* da transação. Esta máquina de estados corresponde aos estados pelos quais o objeto que representa a transação reificada pode passar.

1.3.Trabalhos Relacionados

Como este trabalho versa sobre controle de concorrência no modelo RDF, bem como propõe uma DSL para especificação de transações Web, serão apontados trabalhos relacionados a ambos os temas. Para cada um deles será apresentado um breve resumo do seu propósito.

1.3.1. Transações em RDF

O advento de RDF trouxe consigo a necessidade de investigar como usar este modelo de dados em larga escala. Há vários projetos voltados para desempenho em consultas (*queries*) complexas, propondo melhorias de indexação e otimização. Dentres eles, podemos citar:

- **Scalabe semantic Web data management using vertical partitioning**
  Propõe particionamento vertical dos dados RDF para melhorar consultas RDF. Mostra como estender banco de dados orientados a colunas para implementar esta *abordagem* [Abadi et. al, 2007].

- **DOGMA: A Disk-Oriented Graph Matching Algorithm for RDF Databases**
  Propõe um índice, residente em disco, para grafos RDF e um algoritmo básico para responder a consultas sobre este índice [Bröcheler, 2009].

- **Hexastore: sextuple indexing for semantic Web data management**

  Propõe um esquema que melhora a idea de particionamento vertical dos dados RDF. Os dados RDF são indexados em seis formas possíveis, uma para cada ordenação possível dos três elementos RDF [Weiss, 2008].

- **xRDF3X: Fast Querying, High Update Rates, and Consistency for RDF Databases**

  Propõe um método estendido de *deferred-indexing* que dá suporte a versionamento e altas taxas de atualização, quer seja com isolamento *snapshot* ou *serializable* [Neumann e Weikum, 2010]. O primeiro isolamento garante uma visão consistente dos dados, mas está sujeito a inconsistências quando consideramos atualizações.

1.3.2. Operações RDF

Um metamodelo de operações em RDF com pré/pós-condições, parâmetros de entrada e saída e exceções foi proposto em [Santos, 2010]. Operações são ações atômicas e são a primitiva mais básica de execução. Em um *workflow* de uma transação Web, em algum momento uma operação será invocada para modificar o estado dos objetos de domínio.

Este modelo foi aprimorado para ser incorporado à ferramenta de autoria Synth, criado em [Bomfim, 2011].

1.3.3. Transações em Aplicações Web

Diante da crescente necessidade do emprego de transações em aplicações *Web*, métodos bem conhecidos de design de aplicações *Web*, criados inicialmente para projetar aplicações puramente navegacionais, têm evoluído para fornecer primitivas, conceitos e modelos especificamente destinados para a concepção de transações *Web*. Alguns deles serão descritos a seguir.

1.3.3.1. Object Oriented Hypermedia Design Method (OOHDM)

OOHDM [Rossi, 1996] é um método orientado a objetos para o design de aplicações Web. Em [Rossi et al., 2003], o meta-modelo do OOHDM foi estendido adicionando transações Web (denominadas processos no método) e atividades tanto no modelo conceitual (*domain model*) quanto no modelo navegacional.

1.3.3.2. Object Oriented Hypermedia Method (OO-H)

OO-H [Cachero e Gomez, 2002] é um método parcialmente orientado a objetos, originalmente criado para aplicações Web voltadas apenas para dados. Posteriormente, o método foi estendido para abranger o design de transações (processos). Os modelos previstos no OO-H são: requisitos, conceitual, processos, navegacional, apresentação, arquitetural.

Conforme pode ser visto em [Koch et al., 2003], o método foi enriquecido com modelagem de processos ou transações, utilizando para tal propósito diagrama de atividades da UML[3] para o fluxo interno de casos de uso não triviais.

1.3.3.3. UML-based Web Engineering (UWE)

UWE [Koch e Kraus, 2002] é um método orientado a objetos para especificação de aplicações *Web*. O meta-modelo UWE é uma especialização do meta-modelo da UML e serve como base para a notação UWE que é definida como um *UML profile* (*tagged values* e estereótipos). Esta extensão adiciona um conjunto de elementos específicos para modelagem de navegação, apresentação, transação (processo) e personalização.

A fase de análise de método UWE transcorre da mesma forma do método OO-H, exatamente com os mesmos modelos (modelo de casos de uso, modelo de processos e modelo conceitual). Os dois métodos divergem na fase de design. No tratamento de processos, UWE difere do método OO-H não traduzindo o modelo de processos em um modelo de navegação, mas

[3] Unified Modeling Language (UML) - http://www.uml.org

introduzindo um modelo separado de processo com uma interconexão bem definida com o modelo navegacional [Koch et al., 2003].

1.3.3.4. Web Site Design Method (WSDM)

O WSDM [Troyer e Casteleyn, 2003] é um método centrado no usuário para projeto de aplicações *Web* que prevê seis fases, a saber: definição da missão, modelagem da audiência, design conceitual (dividido em modelos de tarefa e navegacional) e implementação.

O design conceitual é divido em dois passos: modelagem de tarefa e modelagem navegacional. O objetivo da modelagem de tarefa é capturar os requisitos de informação e funcionais das diferentes classes de audiência, e os processos necessários para atender estes requisitos. Para tal são usados dois *modelos: Task Models* e *Object Chunks*.

1.3.3.5. Ubiquitous Web Applications (UWA)

UWA [UWA Consortium, 2002] é um *framework* que provê um conjunto de meta-modelos e ferramentas para o design de aplicações *Web* ubíquas. Este método é divido em quatro atividades principais: Elicitação de Requisitos; Design Hipermídia e de Operações; Design de Transação; Design de Customização (ubiquidade).

Baseado na Linguagem de Modelagem de Transação Unificada (UTML) [Gioldasis e Christodoulakis, 2002] e em conceitos derivados da teoria de banco de dados, a fase de design de transação possui um meta-modelo fundamentado nos seguintes conceitos: Operação, Atividade e Contratos de Execução.

Basicamente, uma transação é especificada por dois modelos: Modelo de Organização (diagrama de classes, onde as atividades são organizadas em árvore) e Modelo de Execução (diagrama de atividades da UML customizado).

1.3.3.6. UWAT+: uma versão estendida do modelo de transação UWA

Visando resolver algumas limitações do UWA, em [Distante, 2005] e [Distante et al., 2007] é proposta uma extensão do modelo de transação da UWA, chamada UWAT+, cujo metamodelo prevê os seguintes modelos: *organization model* (hierarquia de transações e suas atividades); *execution*

model (modelo de fluxo de atividades); *activity cluster diagram* (determina interação entre uma atividade e navegação); *activity node model* (nó de atividade).

1.3.3.7. Modelos e Meta-Modelos para Transações em Aplicações Web

Em [Jacyntho e Schwabe, 2010a] é proposta uma DSL, definida por um meta-modelo e uma notação gráfica, especializada para modelagem sistemática de transações de negócio e transações Web, onde transações são reificadas e tratadas como objetos de primeira ordem. São fornecidas duas perspectivas: estrutural e comportamental.

1.4. Estrutura do Documento

Este documento se inicia com uma breve explanação sobre os fundamentos de transação e Web semântica, no capítulo 2. O capítulo 3 é o capítulo central desta tese. Neste capítulo é delineado o modelo de bloqueio multigranular proposto, bem como os novos tipos de bloqueio voltados para o modelo RDF. O capítulo 4 apresenta a arquitetura e o design de implementação do *Lock Manager*, bem como a DSL de *workflow* e a ontologia de bloqueio propostas. Também é abordada a simulação usada na avaliação de desempenho do modelo de bloqueio proposto, e os resultados obtidos. Por fim, para auxiliar o projetista de uma aplicação, algumas diretrizes de utilização do modelo de bloqueio proposto são enumeradas. Finalmente, o capítulo 5 conclui o trabalho, reiterando as contribuições e elencando alguns trabalhos futuros.