Uma Arquitetura de Armazenamento de Informações de Contexto para Aplicações Ubíquas

Vinícius Maran[1], Iara Augustin[2]

[1] DCEEng – Universidade Regional do Noroeste do Estado do Rio Grande do Sul (UNIJUI) - Rod RS 344 – Santa Rosa – RS – Brasil

[2] Centro de Tecnologia – Universidade Federal de Santa Maria (UFSM) – Av. Roraima, 1000 – Santa Maria – RS - Brasil

vinicius.maran@unijui.edu.br, august@inf.ufsm.br

Abstract. *Ontologies are used in computer systems to the definition of domains. Although there are rules concerning the representations and operations on ontologies, their use in computer systems are not yet complete, because there is not a tool, that from the structure defined in an ontology, is able to accept it in a database data in a distributed fashion, to facilitate recovery and use these data in multiple nodes in a network. This paper presents a new approach to the persistence of ontologies for the representation of context data, which, together with the database CouchDB, aims to offer a software architecture for data recovery and persistence context for ubiquitous systems.*

Resumo. *Ontologias são utilizadas em sistemas computacionais para a definição de domínios. Apesar de existirem normas relativas às representações e operações em ontologias, a sua utilização em sistemas computacionais ainda não é completa, pois ainda não existe uma ferramenta, que a partir da estrutura definida em uma ontologia, seja capaz de aceitá-la em um banco de dados de modo distribuído, para facilitar a recuperação e utilização destes dados em vários nós de uma rede. Apresentamos uma nova abordagem para a persistência de ontologias para a representação de dados de contexto, que, juntamente com o banco de dados CouchDB, visa a oferecer uma arquitetura de software para recuperação e persistência de dados de contexto para sistemas ubíquos.*

1. Introdução

A terceira onda da evolução da computação, também conhecida como computação ubíqua [1], tem como proposta a definição de um espaço onde usuários e dispositivos dos mais diversos tipos estão integrados. Onde a computação é transparente ao usuário, ou seja, dispositivos e sistemas auxiliam o usuário em suas tarefas diárias mesmo que estes usuários não consigam perceber o auxílio da computação envolvida nesses processos. Assim, a principal característica de sistemas ubíquos é que estes realizam tarefas centradas no usuário final e em suas atividades cotidianas de acordo com as necessidades destes usuários e com o contexto onde ele está inserido [2]. O contexto captado por sistemas ubíquos pode ser representado computacionalmente de diversas formas. Uma das formas mais utilizadas é conhecida como ontologia, a qual permite a

representação de um conjunto de conceitos e termos que representam um domínio de conhecimento [3].

Para que os sistemas ubíquos consigam realizar a adaptação ao contexto durante os vários momentos e situações de sua execução (*context-aware adaptation*), é necessário que exista uma arquitetura de persistência de dados de contexto, que permita o acesso e a modificação destes dados em qualquer lugar e a qualquer hora. Para que isto aconteça é necessária uma ferramenta que permita a utilização e modificação de ontologias de uma forma simplificada e distribuída [4].

É apresentada uma proposta de arquitetura para a persistência de ontologias, proporcionando o acesso e modificação de forma distribuída, contemplando características desejáveis de pervasividade dos sistemas ubíquos na utilização de informações de contexto. Na sequência, é apresentada uma breve introdução ao problema do acesso pervasivo contextualizado (seção 2), em seguida apresentamos uma análise dos trabalhos relacionados (seção 3), a proposta de uma arquitetura de persistência distribuída de ontologias (seção 4), a descrição de um cenário de uso da arquitetura proposta (seção 5). A seção 6 descreve as conclusões parciais deste trabalho.

2. Informações de Contexto

Contexto pode ter diversas definições, pois se trata de um termo amplo e que abrange diversas áreas. Neste trabalho, utilizamos a definição feita por Dey & Abowd [8], que define contexto como uma variedade de informações que podem ser utilizadas com o objetivo de caracterizar a situação de um grupo de entidades.

Dados de contexto podem ser representados de diversas formas. Strang & Popien [9] realizaram uma comparação entre formas de representação de contexto e os requisitos que elas atendem. Esta comparação foi baseada em seis fatores principais, nomeados na tabela comparativa como: composição distribuída (*dc*); validação parcial (*pv*); qualidade da informação (*qua*); incompleto e ambíguo (*inc*); nível de formalidade (*for*) e aplicabilidade em ambientes existentes (*app*). Assim, constatou-se na comparação que a modelagem de contexto baseada em ontologias atende aos principais requisitos para a representação completa de contextos. A Tabela 1 apresenta o resultado desta comparação.

Tabela 1. Comparativo entre formas de representação de contexto [9]

Abordagens Requisitadas	***dc***	***pv***	***qua***	***inc***	***for***	***app***
Chave Valor	-	-	-	-	-	X
Esquema de Marcação	X	X	-	-	X	X
Orientado a Objetos	X	X	X	X	X	X
Baseado em Lógica	X	-	-	-	X	-
Gráfico	-	-	X	-	X	X
Baseado em Ontologias	X	X	X	X	X	X

Portanto, a próxima etapa do trabalho corresponde a busca por soluções para a persistência de ontologias de uma forma distribuída, a fim de atender aos requisitos de sistemas ubíquos.

3. Trabalhos Relacionados

Batzios & Mitkas [5] realizaram uma análise completa de ferramentas para o armazenamento de ontologias, são elas: 3store [6], Sesame [6], Jena2 [6] e DB4Owl [Batzios; Mitkas, 2009].

Analisando seus resultados, observamos que, dentre os principais formatos utilizados, estão os bancos de dados relacionais – utilizados principalmente pela utilização de padrões consolidados e por possuírem bom desempenho; e bancos de dados baseados em triplas (*triple store databases*) – utilizados para o armazenamento de documentos RDF [7]. O projeto 3store é baseado na persistência de documentos RDF, utilizando banco de dados de modelo relacional como sistema de persistência. Para a realização de consultas, o 3store utiliza a linguagem RDQL, voltada para consultas na linguagem RDF.

A realização de inferências e consultas é feita através de conversões para a linguagem SQL (utilizada em bancos de dados relacionais), fato que gera muitos dados extras, comprometendo o desempenho em inferências realizadas em grandes bases de dados [5] [6]. Já os projetos Jena2 e Sesame consistem de APIs que utilizam sistemas de banco de dados relacionais como sistemas para persistência de arquivos RDF e OWL-Lite [6].

Para amenizar o volume de dados extras criados na conversão de ontologias para bancos de dados relacionais, o projeto DB4Owl [9] propôs a utilização do banco de dados orientado a objetos DB4o para persistir um conjunto de classes que representam as entidades de uma ontologia. Para a realização de consultas, o projeto utiliza uma linguagem própria, baseada em XML e a converte para a linguagem de consulta utilizada pelo banco de dados DB4o. Um resumo comparativo entre as ferramentas de persistência de ontologias e o trabalho apresentado neste artigo é apresentado na Tabela 2.

Tabela 2. Comparativo de soluções para a persistência de ontologias

	3store	**Sesame**	**Jena2**	**db4owl**	**Arquitetura Proposta Neste Trabalho**
Linguagem Consulta	RDQL	SeRQL	SPARQL	XML	SWRL, SQWRL
Suporte Nativo a OWL	Não	Não	OWL-Lite	Sim	Sim
Facilmente Escalável	Não	Sim	Não	Não	Sim
Modelo de BD	RD	RD	RD	OO	JSON Docs
Baixo Overhead de dados	Não	Não	Não	Sim	Sim

A partir dos resultados dessa comparação, detectou-se que as ferramentas disponíveis atualmente não oferecem alguns requisitos importantes para uma arquitetura pervasiva de persistência de ontologias, são eles:

Suporte a linguagens de consulta e inferência padronizadas para OWL: As ferramentas analisadas suportam ou criam linguagens distintas para a consulta ou inferência, mas poderiam utilizar linguagens já padronizadas e amplamente utilizadas para a manipulação de dados semânticos no formado OWL, dentre estas linguagens destacamos as linguagens SWRL e SQWRL, por se tratarem de padrões consolidados e permitirem a inferência e consulta de dados na linguagem OWL.

Persistir apenas dados relevantes: A constante conversão de formatos e linguagens de consulta faz com que ocorra *overhead* constantemente nas ferramentas atuais, o que acarreta em perda de desempenho considerável dependendo do tamanho da base de dados e do número de consultas realizadas. Além disso, o espaço ocupado por estes dados vai aumentando consideravelmente com o uso destas bases de dados [5].

Utilizar apenas parte da ontologia em memória: O processamento de inferências e consultas deve ser feito apenas com dados relevantes, principalmente porque o desempenho e consumo de memória aumenta consideravelmente com o aumento de indivíduos na ontologia. Por isso, é importante que as modificações sejam processadas em uma parte limitada de dados (consultada na base de dados) [5].

A partir dos resultados encontrados na comparação entre as soluções de bancos de dados pesquisadas, criou-se uma proposta de arquitetura para a persistência e recuperação de ontologias de uma forma distribuída, descrita a seguir.

4. Arquitetura de Armazenamento de Contexto

A arquitetura proposta tem como principal objetivo oferecer uma ferramenta que, em conjunto com o banco de dados CouchDB [10], permita: a (i) persistência de ontologias em uma base de dados escalável e de simples acesso, e a (ii) utilização de linguagens já consolidadas para a realização de consultas e inferências, neste caso as linguagens SWRL e SQWRL [11]. A escolha do CouchDB se deve ao fato de atender aos seguintes critérios analisados:

Facilmente Escalável: O banco de dados CouchDB oferece mecanismos para facilitar a distribuição de dados entre diversos nós em uma rede, oferecendo escalabilidade horizontal. Entre estes mecanismos, destacamos a fácil replicação de documentos entre nós e o controle de versões de documentos, permitindo o controle de concorrência [10];

Modelo de Dados Baseado em Documentos: O banco de dados utiliza o modelo de persistência baseado em documentos JSON [12], são documentos com estrutura facilmente convertida para XML e com *frameworks* de auxílio para a sua utilização por *webservices* [10];

Compatibilidade com Diversas Linguagens de Programação: O banco de dados CouchDB é utilizado independentemente de linguagem de programação, devido a interface de consulta *RESTful* API [10];

A arquitetura proposta trabalha como uma camada superior à camada de controle do banco de dados, e (i) realiza a conversão de arquivos OWL [13] para o formato utilizado no banco de dados CouchDB, (ii) fornece suporte a um conjunto de classes definidas para a representação simplificada de ontologias na linguagem Java, (iii) oferece suporte à realização de consultas e inferências nas ontologias persistidas no banco de dados CouchDB. O diagrama simplificado da arquitetura é apresentado na

Figura 1. Para oferecer compatibilidade com documentos OWL, a arquitetura utiliza um *parser* definido na Protégé OWL API [14], para a conversão de documentos OWL, e um *parser* próprio para a conversão de documentos JSON, persistidos posteriormente no banco de dados CouchDB.

A persistência de dados utilizados pela arquitetura é feita de duas formas: (a) diretamente pela RESTful API [15], que é utilizada diretamente pelo banco de dados CouchDB ou (b) através da API Ektorp [16], que oferece métodos para conversão e consulta de objetos da linguagem Java no banco de dados CouchDB.

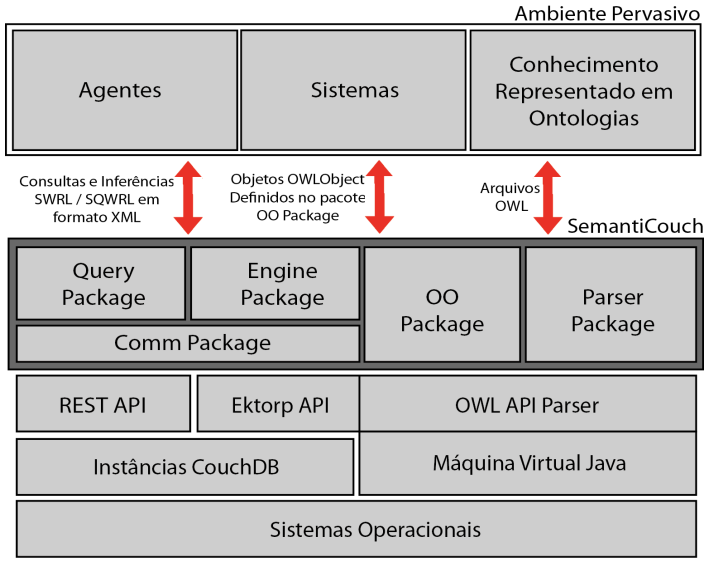

Figura 1. Representação básica dos módulos da arquitetura proposta

A arquitetura é utilizada pelos desenvolvedores e sistemas através de uma API, a qual é estruturada em pacotes que desempenham funções específicas. A próxima seção do artigo detalha a função dos pacotes utilizados na API.

4.1. Organização Lógica da Arquitetura

A estrutura da arquitetura foi separada em pacotes, que são responsáveis por funções distintas no processo de persistência e consulta de ontologias. Os pacotes e suas respectivas funções são descritos a seguir.

Pacote *Comm*: Pacote responsável pela comunicação do pacote Engine com as APIs de comunicação (Ektorp API e *RESTful* API) com o banco de dados CouchDB. Controla os erros de comunicação com o banco de dados e o versionamento de documentos JSON; **Pacote *OO***: Pacote responsável pela definição de classes que representam os componentes de ontologias persistidas no banco de dados. Essas classes podem ser utilizadas pelos desenvolvedores para a comunicação direta de classes Java de seus sistemas com os métodos de persistência e consulta na arquitetura de persistência, tendo como classe *OOWLModel* como classe principal para a modelagem de uma ontologia persistida no sistema. A Figura 2 apresenta um diagrama das classes que compõe o pacote *OO* para a integração de programas Java com a arquitetura de persistência; **Pacote *Engine***: Pacote responsável pela ligação e controle dos pacotes utilizados pela API. Fornece a interface necessária para a utilização da API por sistemas computacionais; **Pacote *Parser***: Pacote responsável pela conversão de formatos para a

exportação ou importação de documentos no formato OWL ou para a conversão interna de formatos utilizados nas consultas, inferências e demais operações no banco de dados.

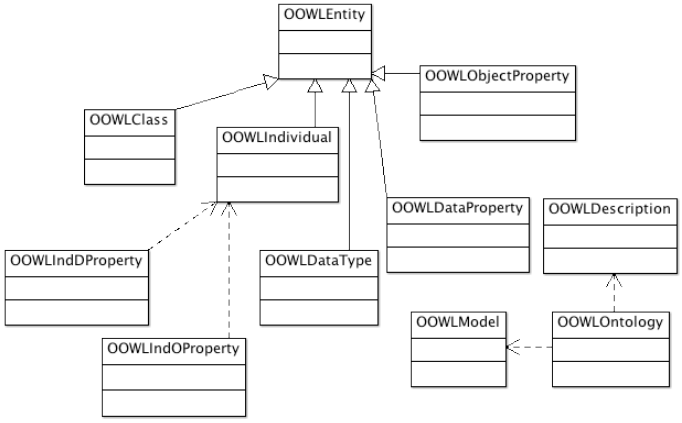

Figura 2. Diagrama de classes do pacote *OO*

Pacote *Query*: Pacote responsável pelo controle de consultas e inferências na base de dados. Oferece o suporte às linguagens SWRL e SQWRL, fazendo a conversão destas para as linguagens utilizadas para a realização de consultas no banco de dados CouchDB.

4.2. Organização de Arquivos

Além da estrutura de programação, a arquitetura mantém uma estrutura de armazenamento definida no banco de dados, reduzindo a necessidade de criação de dados extras, além dos já informados pela ontologia, aumentando o desempenho na realização de consultas e facilitando a replicação de documentos pelo CouchDB. A estrutura de armazenamento utilizada pela arquitetura proposta é apresentada na Figura 3.

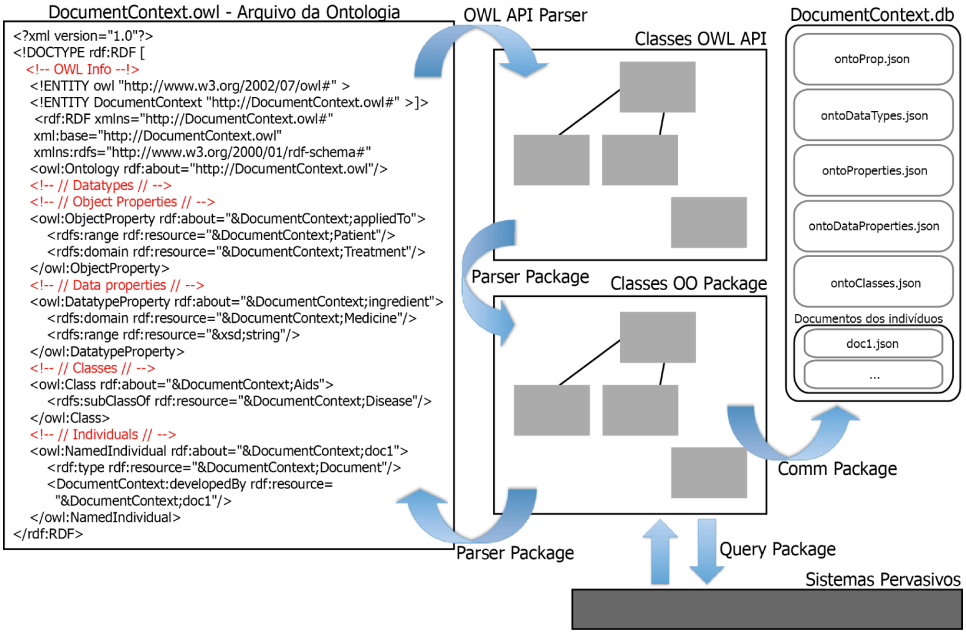

Figura 3. Estrutura de representação de documentos na arquitetura proposta

A divisão da estrutura do arquivo OWL original de uma ontologia permite que o sistema faça consultas carregando apenas os dados necessários para o mesmo. Por exemplo, caso uma entidade externa queira consultar apenas os dados referentes à estrutura de classes da ontologia, o sistema irá fazer a pesquisa somente no arquivo correspondente, sem precisar consultar outros arquivos, que representam os indivíduos da ontologia.

Cada entidade é representada no banco de dados como um documento único, que contém informações de acordo com cada tipo de entidade inserido. Além disso, cada documento possui um campo *_id* e um campo *_rev* únicos, o campo *_id* se refere ao nome único da entidade na estrutura do banco de dados, tendo como base uma *flag* determinando o tipo da entidade e o seu nome, baseado na URI (*Uniform Resource Identifier*) da entidade. Já o campo *_rev* é gerado e utilizado automaticamente pelo banco de dados CouchDB para o controle de concorrência e a replicação de documentos entre diversos nós em uma rede. Para demonstrar as funcionalidades desenvolvidas até o momento, descreveu-se um possível cenário de uso da arquitetura, a seguir.

5. Cenário de Uso

O cenário de uso realizado neste artigo parte da inserção de uma ontologia no banco de dados e a recuperação de um indivíduo descrito nesta ontologia, com o objetivo de avaliar a arquitetura de persistëncia através da interface de operação e das APIs prototipadas até o momento. A Figura 4 apresenta a interface de operação da arquitetura, utilizada para a realização de operações na arquitetura.

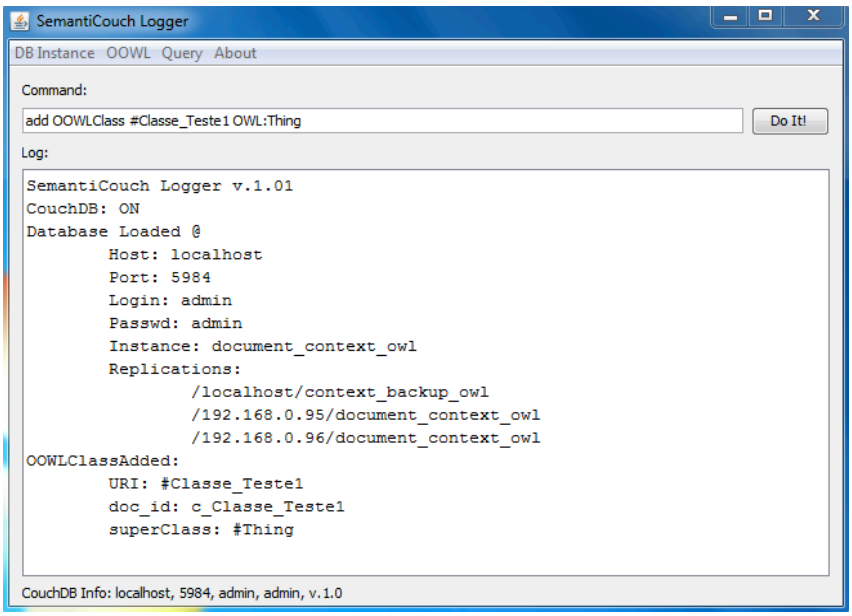

Figura 4. Interface de edição de ontologias na arquitetura

Considera-se a seguinte situação: um sistema de saúde ubíquo baseado em ontologias é utilizado em um hospital. Este sistema possui uma modelagem ontológica de (i) pacientes e seus exames clínicos e (ii) médicos que são vinculados a este hospital e seus respectivos dispositivos de acesso a informações. Em uma determinada tarefa, um médico deseja consultar os exames clínicos de um determinado paciente durante a realização do atendimento a este paciente. Apesar de a ontologia descrever o paciente e

o relacionar com todos os seus exames médicos, o médico somente necessita (i) visualizar os exames relacionados à sua especialidade médica e (ii) a visualizar as informações de forma adaptada à tela do dispositivo que ele está utilizando no momento.

5.1. Organização de Arquivos

Para este cenário de uso, modelou-se uma ontologia simples (Figura 5), que representa as classes e propriedades que podem interferir na adaptação do conteúdo clínico no cenário de uso.

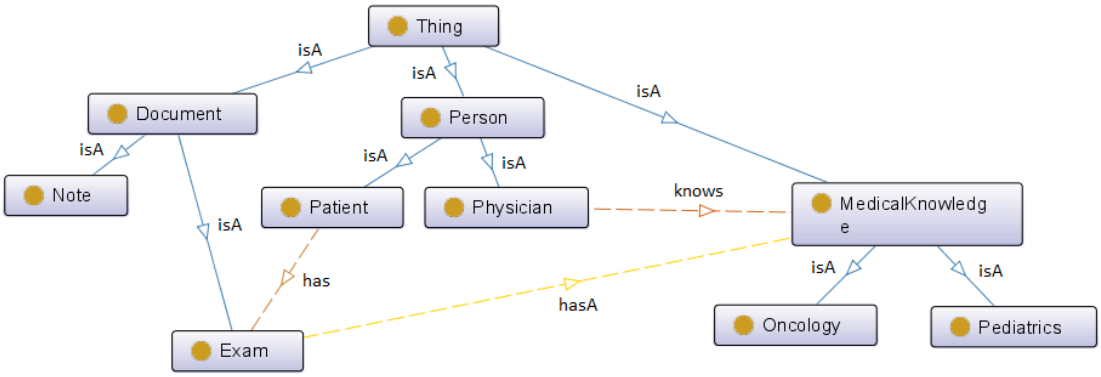

Figura 5. Modelagem ontológica do cenário de uso

A partir da modelagem inicial da ontologia, inseriu-se esta ontologia na arquitetura pela interface *desktop*, através da indicação do arquivo OWL correspondente.

A inserção deste arquivo no sistema implica nas operações: (i) conversão do arquivo OWL para um conjunto de classes da OWL API, onde a ontologia fica representada pela classe *OWLOntology*, descrita na OWL API, (ii) varredura das classes OWL API que representam a ontologia e conversão das mesmas para o conjunto de classes definidas no pacote OO (estas classes possuem mecanismos de conversão para JSON, facilitando a inserção no banco de dados), (iii) a inserção das informações das classes no banco de dados, utilizando a API *Ektorp*.

Após esta primeira etapa, o sistema torna-se capaz de realizar consultas e outras operações nesta ontologia. A arquitetura do banco de dados CouchDB permite que os dados da ontologia sejam replicados em outros nós, e em caso de falha de um destes, a ontologia pode ser acessada e modificada nos outros nós, com a replicação feita de forma automática pelo banco de dados. Realizamos testes variando a quantidade de indivíduos e classes representadas em um arquivo OWL e posteriormente carregando este arquivo na arquitetura de persistência, os resultados são apresentados a seguir (Figura 6).

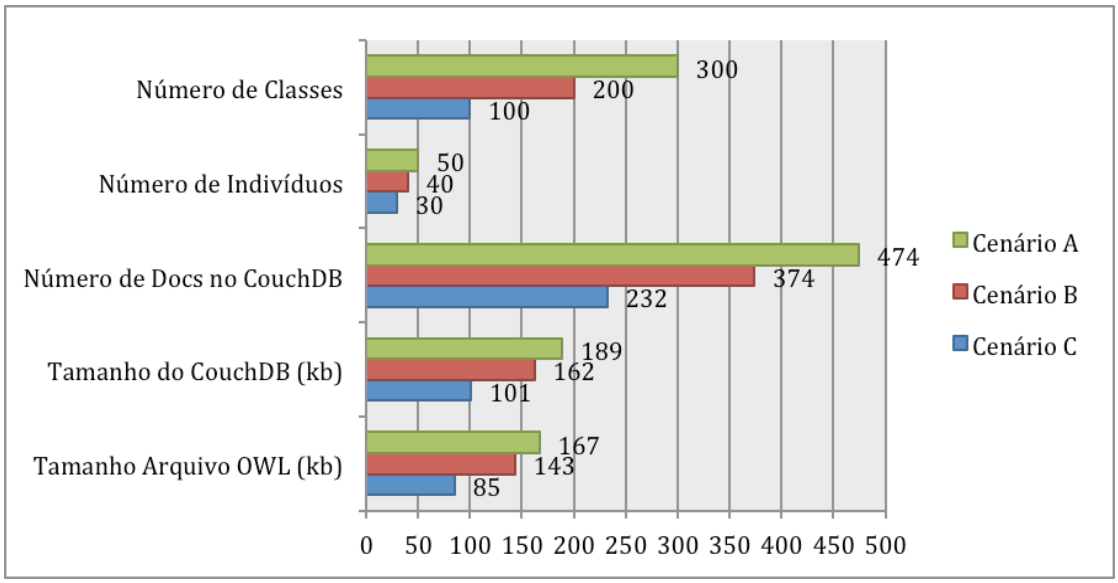

Figura 6. Testes de Inserção de ontologias no sistema

Como podemos observar nos resultados, a inserção dos arquivos OWL no banco de dados geraram um aumento de tamanho dos arquivos (de aproximadamente 13%), este aumento está diretamente relacionado as indexações realizadas automaticamente pelo CouchDB para o aumento de velocidade nos consultas realizadas. Além disso, podemos observar que o número de documentos no banco de dados se manteve alto, mostrando alta fragmentação, uma das características desejáveis para auxiliar a replicação automática de documentos realizada pelo CouchDB.

Caso uma entidade seja consultada via um dispositivo móvel, e consequentemente este dispositivo fique *offline*, este indivíduo poderá ser modificado e atualizado nos outros nós quando o dispositivo ficar *online* novamente pelo banco de dados CouchDB. A partir das definições e testes realizados até o momento, realizamos uma análise dos trabalhos relacionados e as características que a arquitetura proposta possui como contribuição.

6. Agenda de Pesquisa e Conclusões

A utilização de ontologias em sistemas computacionais depende diretamente das ferramentas de persistência, pois estas permitirão a utilização de uma parte da ontologia, sem a necessidade de armazenamento da ontologia em um único arquivo.

As ferramentas atuais não possuem estas características, assim, esse trabalho tem caráter inovador ao propor uma arquitetura, em prototipação, para atender a esses requisitos. Esta arquitetura permitirá a utilização de informações de ontologias de forma distribuída e simplificada. A próxima etapa do projeto será a implementação de um *reasoner SQWRL*, que permitirá consultas mais específicas utilizando os métodos de consulta já definidos na arquitetura

Referências

[1] Weiser, M. (1991) “The Computer of the 21st Century”, Scientific American, v. 265, n. 9, 1991.

[2] Kukhun, D.A.A., Sedes, F. (2007) "*Step Towards Pervasive Software: Does Software Engineering Need Reengineering?*" In the book: Complex Systems Concurrent Engineering, Ed. Springer. ISBN 978-1-84628-975-0, 2007, pp. 143-150.

[3] Swartout, W. and Tate, A. (1999). Ontologies. In IEEE Inteligent Systems and their applications, volume vl 14, n 1. IEEE.

[4] Machado, A., Vicentini, C., Librelotto, G., Augustin, I. (2010) "*Ciência do Contexto para tarefas clínicas em um sistema de saúde pervasivo*", In: XXXVI Conferência Latino-Americana de informática – CLEI, Assuncion, Paraguai. 2010.

[5] Batzios, A., Mitkas, P.A., (2009), "db4OWL: An Alternative Approach to Organizing and Storing Semantic Data," *IEEE Internet Computing*, vol. 13, no. 6, pp. 48-55, Nov./Dec. 2009.

[6] NoSQL. Website. Disponível em: nosql-database.org/. Acessado em fevereiro de 2012.

[7] "Resource Description Framework (RDF)". Website. Disponível em: http://www.w3.org/RDF/. Acessado em fevereiro de 2012.

[8] Dey, A.K. Abowd, G.D. (2000) "Towards a Better Understanding of Context and Context-Awareness". CHI 2000 Workshop on the What, Who, Where, When, and How of Context-Awareness

[9] Strang, T., Popien, C. (2005) "A Context modelling survey". In Proc. Of the Workshop on Advanced Context Modelling, Reasoning and Management as Part of UbiComp, pp.33-40.

[10] Anderson, J. C., Lehnardt, J., Slater, N. (2010) "CouchDB: The Definitive Guide". Livro. Ed. O'Reilly Media. ISBN: 978-0-596-15589-6. 272 pag. 2010.

[11] O'Connor, M.J.; Das, A.K. (2009) "*SQWRL: A Query Language for OWL*", In: Proc. of 6th OWL: Experiences and Directions Workshop (OWLED2009), 2009.

[12] "Introducing JSON". Website. Disponível em: http://www.json.org/. Acessado em março de 2012.

[13] "Web Ontology Language (OWL)". Website. Disponível em: http://www.w3.org/2004/OWL/. Acessado em fevereiro de 2012.

[14] "The OWL API". Website. Disponível em: http://owlapi.sourceforge.net/. Acessado em fevereiro de 2012.

[15] "The Apache CouchDB project". Website. Disponível em http://couchdb.apache.org. Acessado em janeiro de 2012.

[16] "Ektorp Website". Website. Disponível em: http://www.ektorp.org/. Acessado em fevereiro de 2012.