Serviços para Acesso a Bancos de Dados Relacionais como um Módulo de SOA

Diogo Didoné[1], Helena Graziottin Ribeiro[1]

[1] Departamento de Informática – CCET – Universidade de Caxias do Sul (UCS)
Caixa Postal 1352 – 95001-970 – Caxias do Sul – RS – Brasil
diogodidone@gmail.com, hgrib@ucs.br

Resumo. O acesso a diferentes bancos de dados através de interfaces padronizadas simplifica o desenvolvimento de aplicações que precisam utilizar dados integrados e facilita a interoperabilidade entre sistemas heterogêneos. Este artigo apresenta um conjunto de serviços padrão básicos para acesso a banco de dados relacionais, o qual é estruturado como um dos módulos de serviços de SOA. Os serviços foram implementados e disponibilizados como *web services*, que são tipos de serviços bastante conhecidos e utilizados atualmente no desenvolvimento de aplicações distribuídas.

Keywords: serviços de bancos de dados, acesso a bancos de dados relacionais, *web services*, SOA, aplicações distribuídas.

1 Introdução

Grande parte dos dados estruturados acessados via *web* são mantidos em bancos de dados relacionais, cujo padrão para acesso é a linguagem SQL (*Structured Query Language*), com particularidades oferecidas por cada SGBD (Sistema de Gerência de Banco de Dados) [3,8]. Para acessar esses bancos de dados pode-se utilizar algumas API's (*Application Program Interfaces*) padrão específicas, como JDBC (*Java Data Base Connectivity*) para aplicações desenvolvidas em Java e ODBC (*Open Data Base Connectivity*) para aplicações desenvolvidas em C/C++, mas é necessário o uso de *drivers* específicos para cada SGBD [2].

Para ganhar agilidade e eficiência no desenvolvimento de aplicações é possível utilizar plataformas como J2EE e .Net, que permitem estruturar as aplicações em camadas, em geral duas (cliente (apresentação)/servidor (lógica da aplicação+gerência de dados)) ou três (apresentação/lógica da aplicação/gerência de dados) e padronizam a comunicação interna à plataforma, incluindo o acesso a dados (JTS - *Java Transaction Service* - para J2EE, por exemplo). Mas diferentes empresas que possuem aplicações desenvolvidas em plataformas heterogêneas podem precisar trocar dados ou se comunicar, o que torna necessário a existência de recursos de interoperabilidade entre seus sistemas, que podem estar disponíveis ou não no momento da utilização.

Uma abordagem recente para prover interoperabilidade entre sistemas heterogêneos são os serviços. Serviços que utilizam os padrões de *web services* [1,9]

são os tipos de serviços mais populares atualmente, que permitem acessar um ou mais bancos de dados de forma remota e com interface padronizada de comunicação, não restrita a apenas alguns ambientes como JDBC e ODBC.

Porém, os *web services* oferecem apenas a infraestrutura para construir aplicações distribuídas. SOA (*service oriented architecture*) é uma tendência atualmente no que se refere à construção de aplicações distribuídas baseadas em serviços, oferecendo um conjunto de padrões de desenvolvimento que permite construir aplicações a partir da composição de serviços [2,6] acessados de forma assíncrona ou orientado a eventos. Em uma SOA, todos os recursos de software são definidos como serviços, o que inclui de funções de negócios a transações de negócios compostas de funções de baixo nível.

Este artigo apresenta um conjunto de serviços básicos para manipular dados em Sistemas de Gerenciamento de Banco de Dados (SGBD) relacionais. Eles foram implementados utilizando a tecnologia de *web services*, e disponibilizados como um módulo de SOA. Dessa forma, podem ser publicados em um repositório de registros e assim estar disponíveis para serem localizados e consumidos por aplicações distribuídas que utilizam bancos de dados relacionais.

O texto a seguir possui a seguinte estrutura: a seção 2 apresenta os principais conceitos e elementos de *web services* e de SOA que foram utilizados como base de concepção e implementação dos serviços de bancos de dados, e os principais aspectos relacionados ao acesso a bancos de dados relacionais através de SOA. A proposta dos serviços e a implementação realizada são descritas na seção 3, seguidas de um exemplo de utilização. Finalmente, a seção 4 apresenta alguns trabalhos relacionados e a seção 5 as conclusões do trabalho realizado, assim como sugestões de trabalhos futuros.

2 Web Services e SOA

Os *web services* são a evolução natural das plataformas de *middleware* convencional, representadas pelo RPC, CORBA, TP-Monitors, entre outras [1]. Eles são a base comum de comunicação para construir aplicações distribuídas em um contexto independente da plataforma de implementação. SOA surge como um padrão emergente para o desenvolvimento de aplicações distribuídas a partir da integração de serviços, tendo como base os princípios da computação distribuída de fraco acoplamento, independência de protocolos e uso de padrões [2, 6]. Nesse contexto, serviços são módulos bem definidos, autônomos e autocontidos que oferecem funcionalidades-padrão de negócios, e que são independentes do estado ou contexto de outros serviços.

A implementação de SOA não está vinculada a uma tecnologia específica, podendo ser realizada através de um amplo conjunto delas, como RPC, REST, DCOM, CORBA, *Web Services* ou WCF (*Windows Communication Foundation*). *Web services* é então uma das abordagens possíveis para implementar uma arquitetura orientada a serviços.

2.1 Os Web Services

Segundo a W3C [9], *web services* são módulos de software autocontidos que possuem interface para formatar dados e mensagens baseada em XML (*Extensible Markup Language*), linguagem amplamente utilizada como padrão no intercâmbio de dados entre sistemas. A arquitetura dos *web services* [1] é baseada na interação entre provedor (sistema que oferece os serviços), consumidor (sistema que precisa do serviço, cliente) e registro de serviços (sistema que publica os serviços, *service broker*), através de um canal de comunicação comum.

Para a definição de um *web service* utiliza-se um conjunto de padrões (incluindo o XML) para promover a comunicação entre aplicações e facilitar o processamento distribuído: WSDL, UDDI e SOAP. WSDL – (*Web Services Description Language*): é uma especificação que permite descreve um *web service*, mostrando como ele deve ser utilizado, seus tipos de dados e seus serviços. UDDI – (*Universal Description, Discovery and Integration registry*): é um diretório publico onde são publicados e a partir de onde podem ser localizados os serviços. SOAP - (*Simple Object Access Protocol*): é uma gramática padronizada que permite definir os protocolos para enviar mensagens e fazer a comunicação das chamadas remotas ("ligação") entre o provedor e o consumidor.

A padronização dos *web services* é de responsabilidade principalmente dos consórcios W3C *(World Wide Web Consortium)* [9] e OASIS *(Organization for the Advancement of Structured Information Standards)*. Algumas especificações tem sido desenvolvidas para estender os recursos dos *web services*: *WS-security* para assegurar trocas de mensagens seguras via SOAP, *WS-reliability* para que as mensagens entre dois *web services* sejam confiáveis, e *WS-transaction* [10] para manipular transações entre *web services*, entre outras.

2.2 SOA

Em SOA, os recursos (ou módulos) de software são representados por serviços organizados em repositórios e descritos em uma linguagem de definição padrão ("contrato"). Os serviços possuem uma interface publicada, e se comunicam através da requisição da execução de suas operações para suportar uma tarefa ou processo de forma coletiva [4,6] através de um barramento. Em uma analogia à arquitetura dos *web services*, SOA também considera as estruturas de *provedor*, *consumidor* e *registro* (*service broker)* para a comunicação dos módulos de software. Cada módulo pode ter um ou mais papéis entre *provedor* e *consumidor*, e o *service broker*, além de servir como catálogo de serviços, incorpora padrões de segurança a privacidade.

Uma aplicação distribuída construída utilizando SOA é um sistema multicamadas. Além das camadas de apresentação, negócio e persistência, que constituem um sistema tradicional em três camadas, são adicionadas mais duas: a camada de serviços e a camada de processos de negócio. Na figura 1 tem-se um exemplo dessa estrutura de camadas utilizando como base de implementação J2EE (EJB), conforme apresentado em [5].

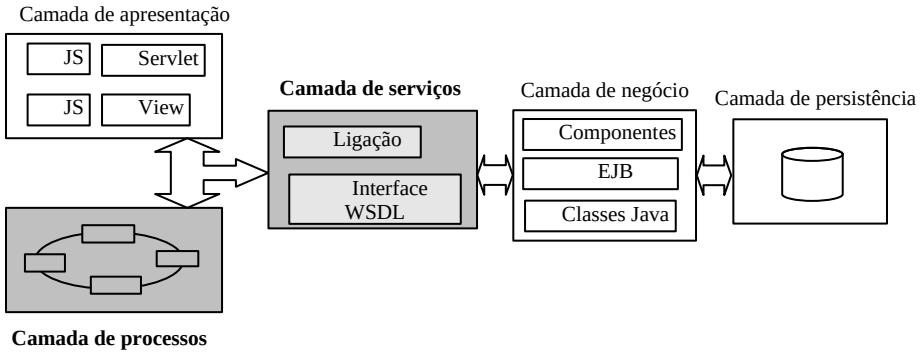

Fig. 1. As camadas de abstração de SOA (apresentação, serviços, processos, negócio e persistência) em uma implementação utilizando J2EE.

SOA é uma especificação divulgada por um grupo informal de líderes da indústria de software, o *Open SOA Collaboration* (http://www.osoa.org), e grandes empresas como IBM, BEA, Oracle , IONA e Microsoft entre outras, tem investido na sua implementação. Os principais projetos desse grupo atualmente são o SCA (*Service Componente Architecture*) e o SDO (*Service Data Objects*) [7]. É através do SDO então que é definido o acesso aos bancos de dados em SOA. O *Apache Tuscany Project* (http://incubator.apache.org/tuscany) é uma implementação de SOA baseada nas especificações do Open CSA (*OASIS Open Composite Services Architecture* (http://www.oasis-opencsa.org/). Em uma implementação paralela, o projeto SODA (*Service-Oriented Database Architecture*) Sql Server [10], apresenta uma arquitetura na qual o acesso a dados é suportado por um provedor de serviços de banco de dados.

2.3 Acesso a dados usando SOA

O problema do acesso a bancos de dados através de API's como JDBC é que ele fica dependente dos *drivers* disponibilizados por cada distribuidor do SGBD utilizado, os quais estão sujeitos a alterações e devem ser atualizados a cada vez que isso ocorre. Assim, uma forma de generalizar esse acesso é utilizar serviços de bancos de dados: as interfaces permanecem inalteradas e as mensagens são trocadas através de um barramento comum. SDO define uma interface uniforme para manipular diferentes formatos de dados, considerando os vários formatos de armazenamento de dados possíveis (XML, relacional, sistema de arquivos, etc). DAS (*Data Access Service*) é uma interface SDO simples que inclui primitivas específicas para acesso a bancos de dados relacionais. Já *SODA* propõe um *middleware que* suporta serviços, inclusive o de acesso a bancos de dados relacionais.

3 Serviços de Bancos de Dados Relacionais

Os serviços propostos nesse trabalho têm como base de definição os comandos da DDL e DML da linguagem SQL. Da DDL considerou-se os comandos básicos de criação e eliminação de usuários, esquemas, tabelas e colunas, além da alteração de colunas. Da DML considerou-se as operações de inserção, eliminação e alteração de linhas, além de consulta a linhas, tabelas e colunas. Além disso, a execução de outros comandos da SQL podem ser solicitados via serviços chamados *script*.

A abordagem proposta é uma versão mais leve em termos de estruturas de implementação para utilizar serviços de acesso a banco de dados como um módulo de SOA. Considera-se apenas o uso de *web services* como recurso de interoperabilidade, enquanto que no SDO (DAS), que se comunica através do ESB (*Enterprise Service Bus*), a interoperabilidade não se restringe a *web services*, pois muitos outros recursos de implementação podem ser utilizados (inclusive *web services*).

3.1 Os Serviços de BDR como módulo de SOA

Os serviços de bancos de dados relacionais (BDR) integram o módulo Serviços de BDR e são inseridos na Camada de Serviços de SOA, ou seja, no mesmo nível de outros serviços com funcionalidades diversas. Como serviços de SOA, ele podem prover independência dos clientes, modularidade e autonomia, entre outras características. A estrutura dos serviços possui duas camadas, uma de definição, que inclui as interfaces dos serviços disponíveis, e outra de implementação (figura 2).

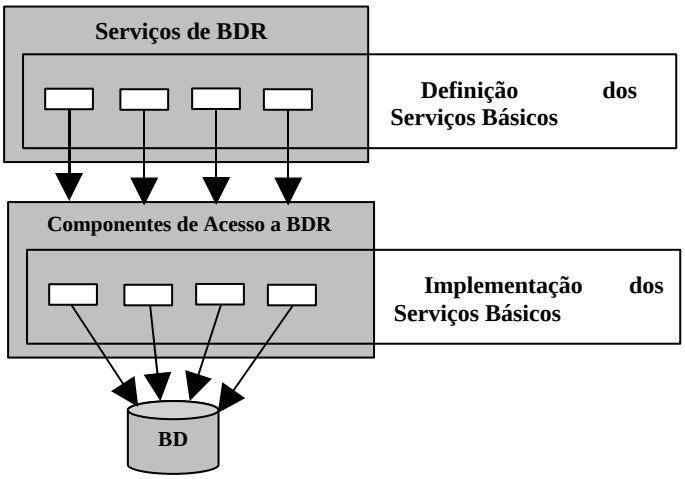

Fig. 2. As camadas de definição e de implementação dos serviços de acesso a BDR.

Desenvolvedores de aplicações que utilizam chamadas a bancos de dados relacionais podem utilizar diretamente as interfaces disponibilizadas pela camada de definição. Para a implementação, pode-se ter diferentes alternativas, sendo uma delas apresentada na seção 3.3.

3.2 Definição dos Serviços de BDR

Os serviços contidos no módulo Serviços de BDR são divididos em três categorias: *Serviços DDL (Data Definition Language), Serviços DML (Data Manipulation Language)* e *Serviços Script*. Os dois primeiros grupos foram definidos conforme os padrões da linguagem SQL, uma vez que um SGBD relacional deve oferecer interfaces apropriadas para cada categoria de serviço. O terceiro tipo de serviço permite tratar os acessos não padronizados definidos através de *strings* de comandos SQL, e irá atender as necessidades de usuários avançados como DBAs (*Database Administrators*) e desenvolvedores, oferecendo a flexibilidade de montar em um formato de texto contínuo seu próprio comando SQL. Os serviços básicos propostos neste trabalho são apresentados na tabela 1. Como exemplos de descrição dos serviços, tem-se na tabela 2 os serviços **Inserir linhas** e **Consultar esquemas**.

Tabela 1. Conjunto de serviços básicos pertencentes ao módulo Serviços de BD

Serviços DDL	Serviços DML	Serviços Script
Criar usuário	Inserir linha	SQL Script geral
Eliminar usuário	Eliminar linha(s)	SQL Script geral consultar
Criar esquema	Alterar linha(s)	
Eliminar esquema	Consultar linha(s)	
Criar tabela	Consultar tabelas	
Eliminar tabela	Consultar colunas	
Criar coluna	Consultar índices	
Eliminar coluna		
Alterar coluna		

Tabela 2. Descrição e interface dos serviços **Inserir linhas** e **Consultar esquemas**

Inserir linhas	
Descrição	Inserir uma linha em uma tabela
Entrada	SGBD, instância, esquema, usuário, senha, tabela, campos e valores
Saída	Resposta da solicitação

Consultar esquemas	
Descrição	Consultar os esquemas de uma instância do BD
Entrada	SGBD, instância, usuário, senha
Saída	Lista dos esquemas

Os serviços *Script* são utilizados para solicitações não padronizadas, visando atender as solicitações baseadas em formatos de chamadas do tipo *APIs*, como a JDBC. Neste caso o usuário monta a sua *string* SQL, submete-a à API e aguarda o retorno, que vai depender do tipo de operação SQL realizada. Serão tratadas nessa categoria as respostas simples (*string*) e as respostas que trazem tabelas. A tabela 3 apresenta os serviços *Script*.

Tabela 3. Descrição e interface dos serviços Script

SQL Script Geral	
Descrição	Inserir comando SQL sem retorno de tipos
Entrada	SGBD, instância, esquema, usuário, senha, comando SQL
Saída	Resposta da solicitação

SQL Script Geral Selecionar	
Descrição	Inserir comando SQL com retorno de uma tabela
Entrada	SGBD, instância, esquema, usuário, senha, comando SQL
Saída	Tabela

Alguns serviços definidos não contemplam completamente os comandos propostos pela SQL. É o caso do serviço "consultar Linhas", que possui apenas uma entrada para definir a tabela da pesquisa, sendo assim não suporta junções de tabelas. Nesse caso, podem ser utilizados os serviços *Script*, que suportam a livre definição de um comando SQL através de um *string* definido pelo cliente.

3.3 Implementação dos Serviços de BDR

Os serviços foram implementados através de um conjunto de tecnologias que permitem dar suporte aos *web services:* TOMCAT 5.5, servidor WEB gratuito oferecido pelo projeto Jakarta para manter os *web services*; JDBC, para o envio de instruções SQL que chegam no container TOMCAT aos Bancos de Dados Relacionais (PostgreSQL e Oracle); JAX-WS(2), biblioteca para a plataforma Java que simplifica a invocação de *web services* via TOMCAT, fazendo o mapeamento de classes Java para mensagens SOAP. As solicitações de serviços e envio de respostas é feita através do TOMCAT.

A implementação dos serviços foi desenvolvida na ferramenta Netbeans. Entre outras características, ela dispõe de uma ferramenta para gerar a publicação do projeto (deploy). Esse processo consiste em gerar um arquivo empacotado com os arquivos binários e demais arquivos de configuração que serão necessários para trabalhar no servidor WEB. Foram implemantadas três classes em Java para organizar os três grupos de serviços (DML, DDL, Script). Elas farão o papel de apresentar a interface externa para o cliente. Cada uma delas contém os serviços propostos em forma de métodos. Além dessas, foram implementadas outras classes para o controle do negócio, entre elas o acesso ao banco de dados via JDBC.

Tanto as classes como os métodos desta interface podem ser documentados através de cláusulas previstas pela biblioteca JAX-WS(2), a qual é desenvolvida para a plataforma Java para simplificar a invocação de *web services*, fazendo o mapeamento de classes Java para mensagens SOAP. Essa documentação fornece auxílio ao cliente pois irá fazer parte da documentação do WSDL.

Utilizando as facilidades das novas formas de declaração para um *web service* da biblioteca JAX-WS2, é possível utilizar anotações que são suportadas a partir da versão 6 do J2EE. Essas anotações são uma espécie de diretivas que implementam uma API para criação do ambiente de *web services*. Por exemplo, a anotação

"@WebService" é colocada no início do programa (antes da declaração da classe), definindo que esta classe é um web service. Desta forma, os desenvolvedores podem se concentrar na lógica da aplicação de seus *web services* sem preocupar-se com as codificações complicadas das APIs como protocolos e descritores da distribuição.

3.4 Exemplo de uso dos serviços

A figura 3 apresenta uma arquitetura básica SOA na qual estariam inseridos os serviços de banco de dados, e mostra a interação dos clientes com os SGBDs através dos serviços. A estrutura proposta neste trabalho aparece através de Serviços BDR (definição) e Acesso BDR (implementação) .

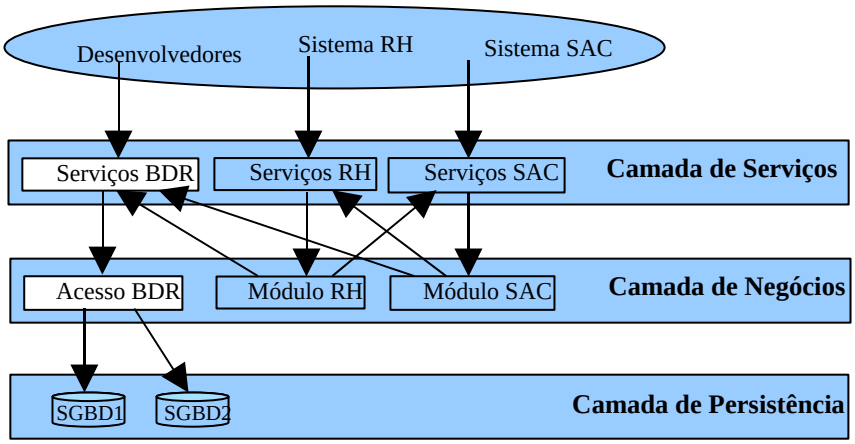

Fig.3.Uso dos serviços de BDR por aplicações e desenvolvedores de aplicações

O exemplo considera o uso de aplicações RH e SAC, que utilizam serviços específicos consultar dados de funcionários (RH) e para retornar informações de interesse do consumidor (SAC – Serviço de Atendimento a Consumidores). Os serviços específicos, como "Serviços RH" e "Serviços SAC", fazem parte de uma especialização dos processos de funções específicas do negócio. Da mesma forma que, a partir da camada de serviços é possível retornar a categoria profissional de um funcionário em um sistema de RH (Recursos Humanos), ele pode utilizar-se dos serviços propostos neste artigo para fazer a consulta e acessar o repositório de dados.

As aplicações de uma empresa (clientes) RH e SAC invocam serviços destinados a atender suas solicitações: 1) Os clientes fazem a chamada para atender sua necessidade ou diretamente a Serviços BDR (geralmente desenvolvedores), ou aos demais serviços específicos do negócio (como Serviços RH e Serviços SAC), na camada de serviços; 2) No caso de utilizarem serviços específicos do negócio, cada módulo específico correspondente na camada de negócio (Módulo RH e Módulo SAC), ao necessitar persistir as informações, invocará o ServiçoBDR na camada de serviços; 3) Serviços BDR acessa a camada correspondente Acesso BDR na camada de negócios; 4) Na camada de negócios, Acesso BDR realiza o acesso ao(s) SGBD(s) específico(s).

O acesso à camada de persistência é realizado apenas através do módulo Serviços BDR na camada de serviços. Nesse contexto, podemos ter um serviço consumindo outros serviços e a camada de negócios, através de Módulo SAC e Módulo RH no exemplo, irá acessar novamente a camada de serviços via ServiçosBD para fazer a ligação até a sua solicitação.

4 Trabalhos relacionados

O projeto *Apache Tuscany* (http://incubator.apache.org/tuscany), que tem por base as especificações definidas pelo Open SCA, considera, além do SCA e do SDO, o DAS, o qual oferece uma interface SDO simples para bancos de dados relacionais. O uso de SDO e de DAS pressupõe porém, o uso do ambiente SCA. SDO é muito abrangente para aplicações que acessam dados armazenados exclusivamente em bancos de dados relacionais, o que compreende a situação de uma grande parte das aplicações. A proposta de DAS reforça essa constatação. Como parte do SCA, o DAS, ao ser instanciado, contém três elementos: *Connection, Config e Commands*. *Connection* é utilizado por todas as operações de banco de dados realizadas no DAS e também para controle de transações, *Config* mantém a referência com a configuração daquela instância do DAS, e *Commands* mantêm todos os comandos de acesso a banco de dados do arquivo de configuração mais aqueles criados durante a execução (*Read*, *Insert*, *Update*, *etc*). A interação do usuário com o DAS é feita através de APIs disponíveis através do *Commands*, como por exemplo *getCommand(name)*, *applyChanges(DataObject)*, *addColumn(...)*, *addPrimaryKey(...)*, entre outros. Os serviços propostos nesse artigo são semelhantes em termos de interface.

Cada instância do DAS é sempre associada com uma conexão de banco de dados. São suportadas transações locais, gerenciadas internamente por cada DAS, e também a participação de um DAS em uma transação global (externa). A versão atual dos Serviços BDR não contempla transações.

Um *web service* genérico de banco de dados é proposto em [2]. No modelo apresentado, e na implementação em Java que o acompanha, as aplicações web não precisam trabalhar com *drivers* de bancos de dados, uma vez que essa comunicação é feita via web services. Porém, trata-se de uma proposta que considera *web services* para acesso a bancos de dados, sem inserção em uma arquitetura que considera o desenvolvimento de aplicações como um todo, como é o caso de SOA.

5 Conclusões

Considerando as limitações de acesso à rede, como *firewalls* e *proxys* que bloqueiam portas e níveis de transporte não comuns, o usuário pode-se utilizar dos Serviços BDR padronizados para acessar um banco de dados disponível através de serviços, tendo como base a SQL. Os serviços propostos são disponibilizados através de SOA, uma estrutura que constitui-se em um paradigma recente para construção de aplicações distribuídas. Além disso, utilizam como base de comunicação *web services*, os tipos de serviços utilizados em larga escala atualmente, e oferecem uma interação padrão

(com base em SQL) para acessar os dados de bancos de dados relacionais. A proposta apresentada é mais enxuta que DAS e tem como objetivo ter uma implementação mais leve, destinando-se a aplicações menores que procuram uma solução interoperável e se comunicam através da internet .

O conjunto de serviços proposto é básico no sentido de que permite utilizar apenas os comandos mais simples para criar e acessar os dados de um banco de dados relacional. Na proposta de implementação são utilizadas apenas tecnologias livres no contexto de desenvolvimento. Como a estrutura definida constitui um módulo independente a idéia é que seja de de fácil integração em aplicações, pois basta adicionar o componente implementado em um contexto de aplicativo web para que de imediato os serviços já estejam disponiveis para o cliente.

Como trabalhos a realizar propõe-se estender o conjunto de serviços para suportar junções de forma transparente, pois na estrutura atual, através dos serviços *script*, cada junção tem que ser escrita com a sintaxe do SGBD utilizado, o que a torna específica para cada SGBD. Além disso, outros trabalhos futuros incluem considerar explicitamente transações (locais ou globais tal qual a proposta de DAS), gerenciar aspectos de segurança em função do uso dos *web services*, possibilitar não apenas o acesso a bancos de dados individuais mas também prover recursos para permitir sua integração via web services, sempre tendo como base a arquitetura SOA.

6 Bibliografia

1. Alonso, G; Casati, F.; Kuno, h.; Machiraju, V. **Web Services: Concepts, Architectures and Applications**. Editora Springer-Verlag NY, 374p. (2004)
2. Dogdu, E.;Wang, Y.; Desetty, S. **A Generic Database Web Service.** Em: Proceedings of the International Conference on Semantic Web & Web Services (SWWS 2006), Las Vegas, USA, p.117-121. (2006)
3. Elmasri, Ramez; Navathe, Shamkant B.**Fundamentals of Database Systems**. 4ª edição São Paulo: Pearson Education, 724 p. (2004)
4. Krafzig, D., Banke, K., and Slama, D. **Enterprise SOA: Service-Oriented Architecture Best Practices** . Editora Prentice Hall, 382p. (2005)
5. Panda, D. **Turn EJB components into Web services.** [S.l.]. http://www.javaworld.com/javaworld/jw-08-2004/jw-0802-ejbws.html (2004)
6. Papazoglou, M.P.; Heuvel, W.J. **Service Oriented Architectures: approaches, technologies and research issues**. Em: The VLDB Journal, Springer-Verlag, Vol.16, N.3, p.389-415. (2007)
7. OSOA (Org.)**SDO V2.1 White Paper**. http://www.osoa.org (2007)
8. Silberschatz, A.; Korth, H.F.; Sudarshan, S. **Sistema de Banco de Dados**. Tradução da 5° edição, Editora Campus, 781 p. (2006)
9. W3C (Org.).**Web Services Architecture**. http://www.w3.org (2008)
10. BEA (Org.) **Web Services Transaction.** http://dev2dev.bea.com/pub/a/2004/01/ws-transaction.html
11. Kiely,Don. How SQL Server 2005 Enables Service-Oriented Database Architectures. http://www.microsoft.com/technet/prodtechnol/sql/2005/sqlsoda.mspx