Ministério da Agricultura, Pecuária e Abastecimento

Comunicado Técnico 16

Dezembro, 2001
Campinas, SP

ISSN 1677-8464

Traduzindo XML para o Modelo Relacional: Experiência no SIGI

Marcos Cezar Visoli[1]
Moacir Pedroso Júnior[2]
João Francisco Gonçalves Antunes[3]
Wagner Correa Ramos[4]

1. Introdução

O uso do XML (Extensible Markup Language) (Harold, 2001) em aplicações Web está em expansão devido a alguns fatores como: facilidade de troca de dados entre plataformas heterogêneas, melhor estruturação de informações, independência de fornecedor por ser uma especificação pública, entre outros. Devido a este crescimento tornou-se freqüente a necessidade do acesso ou da entrada de dados em XML em bases de dados já construídas e em funcionamento nas empresas, na maioria dos casos, há vários anos, onde as base de dados são mantidas na forma relacional apresentada pelos grandes vendedores de Sistemas Gerenciadores de Banco de Dados (SGBDs) do mercado.

O que a maioria dos projetos atuais tem feito é construir programas que importem e/ou exportem os dados de uma base de dados relacional para XML. A maioria dos SGBDs relacionais do mercado oferecem kits para facilitar esta tarefa, e o problema, como de toda tecnologia nova, é a falta de documentação adequada, e também a pouca quantidade de pessoas de uma comunidade que utiliza tal tecnologia.

2. Conversão de Dados de Banco de Dados Relacional para/de o Formato XML

2.1. Arquitetura do SIGI

A arquitetura do Sistema de Informação Gerencial do Instituto Nacional de Investigaciones Agrícolas de Venezuela – SIGI (Pedroso, 2001), pode ser chamada de cliente/servidor off-line, isto é, o cliente não opera em comunicação direta com o servidor.

A parte cliente trata dos processos de planejamento, acompanhamento, reprogramação e avaliação final de projetos e subprojetos de Pesquisa e Desenvolvimento, de Fortalecimento Institucional e Negociação Tecnológica, com uma base de dados local obtida do servidor.

A criação de projetos e subprojetos, dentre outras funcionalidades, é feita diretamente no servidor através de uma aplicação Web. A partir da conexão das máquinas clientes com Windows, os projetos cria-

[1] *Bsc. em Ciência da Computação, Pesquisador da Embrapa Informática Agropecuária, Caixa Postal 6041, Barão Geraldo – 13083-970 - Campinas, SP. (E-mail: visoli@cnptia.embrapa.br)*

[2] *Ph.D. em Pesquisa Operacional, Pesquisador da Embrapa Informática Agropecuária. (E-mail: pedroso@cnptia.embrapa.br)*

[3] *Bsc. em Matemática Aplicada e Estatística, Pesquisador da Embrapa Informática Agropecuária. (E-mail: joaof@cnptia.embrapa.br)*

[4] *Bsc. em Ciência da Computação, Consultor da Embrapa Informática Agropecuária na área de banco de dados e desenvolvimento para WEB do SIGI. (E-mail: wagnerc@cnptia.embrapa.br)*

Embrapa

dos são trazidos da base central e são reenviados ao servidor quando o usuário o quer atualizá-lo ou enviá-lo para um outro usuário ou para um grupo de usuários (Pedroso, 2001). Os dados de projetos sempre estarão sincronizados com a base local nos clientes e a base central no servidor, para que estejam disponíveis para outros interessados com a respectiva autorização de acesso.

2.2. Banco de dados relacional e XML

Temos hoje um mercado crescente na área de bancos de dados, muito disputado pelas empresas de software, que é o do gerenciador de banco de dados em XML nativo, isto é, um gerenciador de banco de dados capaz de armazenar e extrair dados no formato XML sem a necessidade de nenhuma programação.

Existem diversas vantagens e desvantagens na utilização destes produtos, o que ocasiona uma discussão verdadeiramente infindável e, como sempre, rodeada de interesses comerciais atrás de cada opinião.

No projeto SIGI não houve a necessidade de uma maior exploração de um banco de dados XML, uma vez que o cliente tem definido como seu SGBD o Oracle (Oracle, 2002). Neste sentido é possível comentar apenas sobre os prós e contras do uso do banco de dados relacional e sua importação/exportação em XML:

- os dados, sendo mantidos em um banco de dados relacional facilitam a integração entre sistemas;
- ao utilizar-se um banco de dados relacional automaticamente é necessário realizar um trabalho de modelagem de dados na qual a normalização, questões de desempenho e de integridade dos dados são levadas em consideração;
- existem várias ferramentas, bibliotecas para a geração e visualização dos dados a partir de um banco de dados relacional, facilitando assim o uso da informação para outros fins além de aplicações Web;
- os SGBDs relacionais do mercado são produtos maduros, estáveis, com muitos anos de pesquisa e correções de erros, oferecendo assim a melhor opção para a garantia da integridade dos dados. Há uma grande quantidade de informações e experiências capazes de garantir como um banco de dados relacional se comporta quanto existem tabelas com uma grande quantidade de informações (números em terabytes), o que proporciona um certo nível de segurança em uma tomada de decisão;
- o esforço necessário para a exportação de dados de um banco relacional para XML é muito pequeno, geralmente demandando apenas uma cláusula ou comando a mais em SQL. O que às vezes é necessário é a definição de um arquivo XSL (Extensible Stylesheet Language) (Muench, 2000) para que o XML gerado esteja em um determinado formato, diferente do padrão gerado pelo SGBD;
- a importação de dados de um arquivo XML para o relacional requer um pouco mais de esforço, pois existe a necessidade de conhecimento da aplicação para o tratamento adequado de exceções, realizar a correta validação dos dados, retornar mensagens de erros elucidativas, etc. O ideal é a construção de um programa servidor que trabalhe com transações previamente conhecidas e que cada uma destas transações aceite somente um determinado arquivo XML, num formato previamente conhecido pelo programa que processa a transação. Isso agiliza a realização das validações, tratamento de exceções e retorno de mensagens de erro. Os pacotes oferecidos pelos SGBDs relacionais oferecem mecanismos de importação direta dos dados em XML para relacional, mas isso causa uma perda de controle imensa da aplicação em relação a validação, tratamento de exceções e de erros.

3. A Experiência no SIGI: XML e Oracle 8i

A arquitetura do SIGI foi construída baseada na impossibilidade de manter uma conexão constante do cliente com a aplicação servidora ou o banco de dados. Isto determinou a necessidade de definir um formato para os dados que seriam enviados e trazidos para o banco de dados. A escolha do XML como formato foi uma decisão natural devido a grande disponibilidade de material bibliográfico, de ferramentas para tratamento e manipulação e de ser um padrão cada vez mais utilizado no mercado.

Assim, os dados precisam ser constantemente exportados e importados, onde a aplicação cliente sempre trabalha com um subconjunto dos dados, desconectada da base central, e após possíveis alterações, que podem ser feitas num intervalo grande de tempo, esse subconjunto é submetido à base central para a sincronização dos dados.

Inicialmente imaginamos que poderíamos construir um servidor genérico, escrito na linguagem Java, com a tecnologia Servlets (Hunter, 2001), que pudesse importar dados de um arquivo XML para a base de dados e também exportar dados da base para XML.

Houve a disponibilidade do XDK (Muench, 2000), que é o XML Development Kit da Oracle, pacote que permite importar e exportar dados XML. O XDK reduziu a tarefa de exportação de dados para o formato XML, pois através do uso de uma classe (OracleXMLQuery) foi possível, com poucos comandos, gerar arquivos no formato desejado para todos os dados necessários pela aplicação cliente.

Já para a operação inversa, ou seja, importar os dados para a base de dados em Oracle, não foi possível usar as ferramentas disponíveis no XDK, pois é necessário um tratamento um pouco mais flexível do que o oferecido.

A solução adotada foi a construção de um servidor em Java, que recebe as requisições dos clientes através de um formulário HTML com um arquivo anexado compactado contendo todos arquivos XMLs necessários para a transação requisitada.

O servidor, chamado de SIGISync, processa as seguintes transações:

- LISTPRY – produz uma lista com os títulos dos projetos que podem ser acessados pelo usuário. A aplicação cliente usa essa lista para apresentar um menu de opções ao usuário, para que este então escolha o projeto sob o qual deseja interagir.
- GETPRY – exporta todos os dados relativos a um determinado projeto.
- SYNCSUP – atualiza as tabelas suplementares (se necessário) com as tabelas do banco de dados central. Se no servidor central houver uma tabela mais atualizada do que na aplicação cliente, esta é enviada para a aplicação cliente.
- SYNCPRY – sincroniza os dados relativos a um determinado projeto. Inicialmente verifica se a versão do projeto na base central é a mesma que a versão existente no cliente, e, caso não seja, a versão existente no cliente é sobreposta com a versão da base central. No caso das versões serem as mesmas, as alterações realizadas pelo usuário são efetivadas na base central e a versão do projeto é incrementada. Essa tarefa é realizada tanto para projetos como para subprojetos.
- SUBMITPRY – submete um projeto para ser analisado pela instância superior. Após um projeto ser submetido o usuário perde o direito de alteração no mesmo.
- SUBMITSPY – submete um subprojeto para ser analisado pela instância superior. Após esta transação o usuário perde o direito de alteração no mesmo.
- CHECKLOGIN – verifica a validade do usuário e senha.

Para todas as transações a aplicação cliente envia um formulário do tipo *post* (HTML) contendo o código da transação e um arquivo compactado anexado.

O servidor processa a transação e retorna um arquivo compactado contendo os dados solicitados em arquivos XML, que são lidos pela aplicação cliente.

Na próxima seção são apresentados alguns exemplos relacionados a exportação e importação de arquivos XML para/de bases no servidor Oracle 8i.

4. Exemplos

O método exportRecurrenciaProyecto mostrado a seguir é um exemplo do uso do XDK para a geração de saída em XML a partir do servidor Oracle. Usa-se um comando SQL SELECT normal para realizar a consulta na base de dados. A diferença está no uso da cláusula CURSOR usada para produzir uma saída hierárquica dos dados – um recurso valioso que o Oracle oferece para facillitar a geração de dados em XML.

```
/**
Exporta todas as recurrencias de un projeto e suas relacoes para XML,
gravando o arquivo XML no diretorio downloadDir, e usando o arquivo
recurrencia_proyecto.xsl (stylesheet) no  diretorio xslDir.
@param idProyecto ID do projeto.
@param xslDir Nome do diretorio de arquivos XSL.
@param downloadDir Nome do diretorio de download da sessão.
@param conn Conexão com a base de dados.
@return 0 se bem sucedido ou o código de erro.
*/
  public synchronized int exportRecurrenciaProyecto(String idProyecto, String xslDir, String
downloadDir, Connection conn)
  {
```

```
    try {

    // init the OracleXMLQuery
    String query = "select  R.ID_RECURRENCIA, " +
                   "        R.ID_PROYECTO, " +
                   "        R.ID_SUBPROYECTO, " +
                   "        R.DT_RECURRENCIA, " +
                   "        R.ID_ESTADO, " +
                   "        E1.DS_ESTADO, " +
                   "        R.VL_PRESUPUESTO, " +
                   "        R.ID_EVALUACION_CT, " +
                   "        R.DS_EVALUACION_CT, " +
                   "        R.ID_EVALUACION_CG, " +
                   "        R.DS_EVALUACION_CG, " +
                   "  cursor(  " +
                   "     select  " +
                   "        M.ID_META_PORCENTAJE, " +
                   "        M.ID_META,  " +
                   "        M.ID_RECURRENCIA,  " +
                   "        M.VL_PERCENTUAL,  " +
                   "        N.DS_RESUMIDA  " +
                   "     from META_PORCENTAJE M, META N " +
                   "     where M.ID_RECURRENCIA = R.ID_RECURRENCIA and " +
                   "           M.ID_META = N.ID_META) as DSF_MTA_POR " +
                   "  from   RECURRENCIA R, ESTADO E1  " +
                   "  where  R.ID_ESTADO = E1.ID_ESTADO and " +
                   "         R.ID_SUBPROYECTO IS NULL and  " +
                   "         R.ID_PROYECTO = " + idProyecto;

    OracleXMLQuery qry =  new OracleXMLQuery(conn,query);

    qry.setRaiseException(true);

    //------
    // shape the XML doc generated

    qry.useUpperCaseTagNames();                    // use upper case tag names
    qry.setEncoding("ISO-8859-1");
    qry.setDateFormat("yyyyMMdd");
    qry.setXSLT(xslDir + File.separatorChar + "recurrencia_proyecto.xsl", null);

    // get the XML document in the string format
    String xmlString1 = qry.getXMLString();
    String xmlString = xmlString1.substring(0, xmlString1.indexOf("UTF-8")) + "ISO-8859-1" +
                       xmlString1.substring(xmlString1.indexOf("UTF-8")+5);

    qry.close();
    //
    File outputFile = new File(downloadDir + File.separatorChar + "recpry.xml");
    FileWriter out = new FileWriter(outputFile);
    out.write(xmlString);
    out.close();
    return 0;
    }
    catch (Exception e) {
      cat.error("Excecao em exportRecurrenciaProyecto.", e);
      return 1;
    }
  }
```

O trecho do método updateProyecto, mostrado a seguir, realiza a atualização dos dados de um projeto na base de dados. A origem dos dados do projeto está em arquivos XML (Pry.xml, RecPry.xml e SegPry.xml).

O método monta uma árvore DOM - Document Object Model (Muench, 2000) a partir dos arquivos XML recebidos da aplicação cliente, variável doc do tipo XMLDocument e a partir desta árvore DOM realiza as atualizações na base de dados na ordem correta, respeitando as integridades referenciais do modelo de dados.

```
/**
Atualizar um projeto a partir dos arquivos Pry.xml (projeto),
Recpry.xml (recurrencias de projeto) y segpry.xml (Seguimientos de projeto).
@param idProyecto ID do projeto.
@param uploadDir Nome do diretorio de upload da sessão.
@param loginName Login do usuario.
@param conn Conexão com a base de dados.
@return 0 se bem sucedido ou o código de erro.
*/

public int updateProyecto(String idProyecto, String uploadDir, String loginName, Connection conn)
throws SQLException
{
```

```
FileReader fileinput = null;
int status;
int st;
Element e;
Element pe;

NodeList n1;
NodeList n2;
NodeList n3;
NodeList n4;
Node x;
int s1, s2;
File F;
DOMParser parser;
XMLDocument doc;
String idRecurrencia = null;
String idSeguimiento = null;
String idProductoSeguimiento = null;

XMLNode r;
int result;

try {

   if (hashURIPry == null) {
      buildAccessTables(conn, loginName, idProyecto);
   }
   cat.debug("Entrou em updateProyecto.");

   F = new File(uploadDir+File.separatorChar+"Pry.xml");
   if (F.exists()) {
      F = null;
      cat.debug("Existe atualizacao a fazer em Proyecto.");
      parser = new DOMParser();
      fileinput = new FileReader(uploadDir+File.separatorChar+"Pry.xml");
      if (fileinput != null) {
         parser.setErrorStream(System.err);
         parser.setValidationMode(XMLParser.NONVALIDATING);
         parser.showWarnings(false);
         parser.parse(fileinput);
         doc = parser.getDocument();

         n1 = doc.selectNodes("XMLSIGI/cdsProyecto/registro");
         if (n1 != null) {
            int len = n1.getLength();
            for (int j=0; j < len; j++)
            {
               pe = (Element)n1.item(j);
               status = (new Integer(pe.getAttribute("status"))).intValue();
               if (status == 1) {   // analizar nested..
                  cat.debug("Proyecto registro status = 0");
                  // Procura por actualizaciones en nesteds...
                  n2 = pe.getChildNodes();
                  if (n2 != null) {
                     s1 = n2.getLength();
                     for (int i=0; i < s1; i++) {
                        e = (Element)n2.item(i);
                        // Processa actualizaciones en palabra_clave.
                        if (e.getTagName().equals("DSF_PALCVEPRY")) {
                           n3 = e.getChildNodes();
                           if (n3 != null) {
                              s2 = n3.getLength();
                              for (int u=0; u < s2; u++) {
                                 e = (Element)n3.item(u);
                                 st = (new Integer(e.getAttribute("status"))).intValue();
                                 result = updatePalabraClave(conn, st, e, idProyecto, "");
                                 if (result != 0) {
                                    if (result == 999) {  // Autorizacion negada.
                                       return 999;
                                    }
                                    else {
                                       return 101;
                                    }
                                 }
                              }
                           }
                        }
                        // Processa actualizaciones en naturaleza_proyecto.
                        if (e.getTagName().equals("DSF_NATURALEZA")) {
                           n3 = e.getChildNodes();
                           if (n3 != null) {
                              s2 = n3.getLength();
```

```
                                for (int u=0; u < s2; u++) {
                                   e = (Element)n3.item(u);
                                   st = (new Integer(e.getAttribute("status"))).intValue();
                                   result = updateNaturalezaProyecto(conn, st, e, idProyecto, "");
                                   if (result != 0) {
                                      if (result == 999) {  // Autorizacion negada.
                                         return 999;
                                      }
                                      else {
                                         return 102;
                                      }
                                   }
                                }
                             }
                          }  // DSF_NATURALEZA

...

                      }
                      cat.debug("Proyecto registro status = 1 (update)");
                      result = updatePry(conn, status, pe, idProyecto, "");
                      if (result != 0) {
                         if (result == 999) {  // Autorizacion negada.
                            return 999;
                         }
                         else {
                            return 100;
                         }
                      }
                   }
                   else {
                      cat.error("Proyecto registro status = " + status);
                      return 901;
                   }
                }
             }
             else {
                cat.debug("Pry.xml vazio");
             }
             fileinput.close();
          }
       }  // if F.exists()


       // Actualizacion de las recurrencias de proyecto.

       F = new File(uploadDir+File.separatorChar+"RecPry.xml");
       if (F.exists()) {
          F = null;
          cat.debug("Existe atualizacao a fazer em recurrencia de Proyecto.");

          parser = new DOMParser();
          fileinput = new FileReader(uploadDir+File.separatorChar+"RecPry.xml");
          if (fileinput != null) {
          parser.setErrorStream(System.err);
          parser.setValidationMode(XMLParser.NONVALIDATING);
          parser.showWarnings(false);
          parser.parse(fileinput);
          doc = parser.getDocument();
          n1 = doc.selectNodes("XMLSIGI/cdsRecPry/registro");
          if (n1 != null) {
             int len = n1.getLength();
             for (int j=0; j < len; j++) {
                pe = (Element)n1.item(j);
                status = (new Integer(pe.getAttribute("status"))).intValue();
                if (status == 1) {
                   // Procura por actualizaciones en nesteds...
                   n2 = pe.getChildNodes();
                   if (n2 != null) {
                      s1 = n2.getLength();
                      for (int i=0; i < s1; i++) {
                         e = (Element)n2.item(i);
                         if (e.getTagName().equals("ID_RECURRENCIA")) {
                            r = (XMLNode) e;
                            idRecurrencia = r.getText().trim();
                            cat.debug("Processando Recurrencia Pry (nested) ID=" + idRecurrencia);
                         }
                         // Processa actualizaciones en meta-percentual
                         if (e.getTagName().equals("DSF_MTA_POR")) {
                            n3 = e.getChildNodes();
                            if (n3 != null) {
```

```
                        s2 = n3.getLength();
                        for (int u=0; u < s2; u++) {
                           e = (Element)n3.item(u);
                           st = (new Integer(e.getAttribute("status"))).intValue();
                           result = updateMetaPercentual(conn, st, e, idRecurrencia, "");
                           if (result != 0) {
                              if (result == 999) {  // Autorizacion negada.
                                 return 999;
                              }
                              else {
                                 return 110;
                              }
                           }
                        }
                     }
                  } // DSF_MTA_POR
               }
            }
            result = updateRecurrencia(conn, status, pe, idProyecto, "");
            if (result != 0) {
               if (result == 999) {  // Autorizacion negada.
                  return 999;
               }
               else {
                  return 111;
               }
            }
         }
      }
   }
   else {
      cat.debug("RecPry.xml vazio");
   }
   fileinput.close();
}  // if F.exists()

// Actualizacion de los seguimientos de proyecto.

F = new File(uploadDir+File.separatorChar+"SegPry.xml");
if (F.exists()) {
   F = null;
   cat.debug("Existe atualizacao a fazer em seguimientos de Proyecto.");

   parser = new DOMParser();
   fileinput = new FileReader(uploadDir+File.separatorChar+"SegPry.xml");
   if (fileinput != null) {
      parser.setErrorStream(System.err);
      parser.setValidationMode(XMLParser.NONVALIDATING);
      parser.showWarnings(false);
      parser.parse(fileinput);
      doc = parser.getDocument();
      n1 = doc.selectNodes("XMLSIGI/cdsSegPry/registro");
      if (n1 != null) {
         int len = n1.getLength();
         for (int j=0; j < len; j++)
         {
            pe = (Element)n1.item(j);
            status = (new Integer(pe.getAttribute("status"))).intValue();
            if (status > 0) {
               result = updateSeguimiento(conn, status, pe, idProyecto, "");
               if (result != 0) {
                  if (result == 999) {  // Autorizacion negada.
                     return 999;
                  }
                  else {
                     return 112;
                  }
               }
               result = updateEvaluacionProyecto(conn, status, pe, idProyecto);
               if (result != 0) {
                  if (result == 999) {  // Autorizacion negada.
                     return 999;
                  }
                  else {
                     return 113;
                  }
               }
            }
         }
      }
      else {
         cat.debug("SegPry.xml vazio");
```

```
            }
            fileinput.close();
         }
      }  // if F.exists()

      if (pryUpdated) {
         // Incrementa version del proyecto.
         cat.debug("Incrementando version del proyecto");
         if (updateDB(conn, "update PROYECTO set cd_version = cd_version + 1 where id_proyecto = " +
idProyecto) != 0) {
           return 903;
         }
      }

      return 0;
   }
   catch (Exception ex1) {
     cat.error("Excecao em updateProyecto.", ex1);
     return 908;
   }
   finally {
      try {
         if (fileinput != null) {
             fileinput.close();
         }
      } catch (Exception ex2) {};
   }
}
```

5. Conclusão

O processo importar/exportar dados de uma base de dados relacional para XML é muito semelhante a importar/exportar dados para um arquivo de texto seqüencial comum, sendo que a diferença é que no caso do XML o esforço é um pouco menor devido à existência de ferramentas disponibilizadas pelos fornecedores de SGBDs para exportar e importar dados em XML.

Do ponto de vista da aplicação cliente na arquitetura do projeto SIGI, também o uso do formato XML diminuiu o custo do tratamento da informação devido ao grande número de ferramentas e pacotes disponíveis para importar, manipular e exportar dados.

6. Referências Bibliográficas

HAROLD, E. R.; MEANS, W. S. *XML in a Nutshell*: a desktop quick reference. Beijing: O´Reilly, 2001. 480p.

HUNTER, J., CRAWFORD, W. *Java Servlet Programming*. 2.ed. Beijing: O´Reilly, 2001. 753p. (The Java series)

MUENCH, S. *Building Oracle XML applications*. Beijing: O´Reilly, 2000. 792p. Inckui CD-ROM.

ORACLE CORPORATION. URL: http:/www.oracle.com. Consultado em 15 jan. 2002.

PEDROSO, M. JR. *SIGI*: Sistema de Informação Gerencial de Projetos de Pesquisa Agropecuária para o Instituto Nacional de Investigaciones Agrícolas da Venezuela. Campinas: Embrapa Informática Agropecuária, 2001. (Sistema Embrapa de Planejamento, Projeto 12.2001.950). Projeto em andamento.

Comunicado Técnico, 16

MINISTÉRIO DA AGRICULTURA, PECUÁRIA E ABASTECIMENTO

GOVERNO FEDERAL
Trabalhando em todo o Brasil

Embrapa Informática Agropecuária
Área de Comunicação e Negócios
Av. Dr. André Tosello s/nº
Cidade Universitária - "Zeferino Vaz"
Barão Geraldo - Caixa Postal 6041
13083-970 - Campinas, SP
Telefone/Fax: (19) 3789-5743
E-mail: sac@cnptia.embrapa.br

1ª edição