How to overcome the relational – XML Impedance Mismatch: XML processing within the JDBC Sender Adapter

Applies to:

SAP NetWeaver, SAP Exchange Infrastructure, SAP Process Integration

Summary

Constructing messages from multiple interdependent database tables via the JDBC Sender Adapter can be quite tricky. This document shows a very elegant way to solve the multi table problem by using the XML features of modern SQL databases

Author: Manfred Lerner

Company: SAP SI AG

Created on: 13.12.2007

Author Bio

Manfred Lerner works for SAP Consulting since fall 2004.

With more than 15 years experience in the Enterprise Application Integration area, Manfred Lerner has a deep knowledge and understanding of the requirements of the whole set of integration scenarios.

Formerly responsible for the product management of an integration platform, he joined SAP in 2004. Since then, he has been active in several projects where his skill as a NetWeaver Solution Architect helped leveraging the benefits customers could get from the SAP NetWeaver platform. Thereby he helped setting up Technical Program Management for customers. As architectures and products get more and more versatile and comprehensive, the management of technology by the means of architecture management is one of the decisive key success factors..

Table of Contents

Summary 1
Author Bio 1
Table of Contents 2
Introduction 3
The Impedance Mismatch between SQL and XML 3
The relevance of this Impedance Mismatch for SAP PI 3
How to overcome the Impedance Mismatch between RDBMS and XML 4
Copyright 6

Introduction

The concept of message processing within the SAP Process Integration – further called SAP PI – is based on the paradigm of XML based message handling. When connecting to platforms who speak XML like marketplaces etc. messages can be processed straight forward.

A XML message is sent to SAP PI via the SOAP Adapter, transformed within SAP PI and then forwarded to any business system. The XML message can be arbitrary complex, with nested hierarchies on n levels.

But when connecting to the world of relational databases with SQL as the query language, things can get sometimes hard.

The Impedance Mismatch between SQL and XML

The relational model represents it data structures in terms of tables and columns. In the past this normalized table model was very successful for storing a large amount of data. This led us to applications which could be designed according to the ACID principle:

- A – atomicity
- C – consistency
- I – isolation
- D – durability

The table – column oriented storage model of relational database management systems has some limitations when querying the data. It is impossible to retrieve data sets which are correctly modeled in accordance with the theory of normalization and thereby distributed among several database tables within one SQL statement. You have to do some coding which gets the master table and then is iterating over the corresponding detail tables.

XML has a hierarchical data model. Data can be arranged in any kind of hierarchy. So it is very easy to put tree structures into XML documents.

The relevance of this Impedance Mismatch for SAP PI

Within the JDBC Sender Adapter of SAP PI, data can be extracted from any JDBC compliant database by issuing a SQL statement against the underlying database system.

When data from one database table should be sent to SAP PI, the corresponding SELECT statement is simply entered in the JDBC Sender Channel. What is SAP PI doing with this statement?

We have the following process flow:

- send the SQL query to the RDBMS
- The RDBMS executes the query
- The result set is sent back to SAP PI
- The JDBC Sender Adapter is processing the result set via the JDBC API using the standard processing around the javax.sql.ResultSet class
- The result set is wrapped into a XML structure. There each column of the result set is converted to a XML tag with the corresponding name.
- These columns are put into a XML element with the tag name <row>
- The adapter loops over all rows until the end of the result set is reached
- The optional UPDATE statement is executed
- The XML message is persisted within SAP PI

- A commit is sent to the external database

The JDBC Sender Adapter can only process the results of such a SQL query. And here we have the structural issue between the RDBMS world and the XML world.

The JDBC Sender Adapter has no way to collect data which is distributed over a set of n tables. It can simply process SQL Statements including stored procedures/functions which return result sets. So when you want to retrieve data out of a SQL database which is correctly distributed over 5 tables, you will have to write a SELECT statement which joins the 5 tables. You will get the result as one big Cartesian product which will be very difficult to handle (of course there are further disadvantages like the increase of data due to the Cartesian product).

The only way to handle this in the conventional way within SAP PI is to

a) setting up a separate JDBC Sender Channel for each set of relation in the database and build the hierarchical structure within a BPE process
b) select only the master table in the JDBC channel and use the lookup API within the mapping to get the data out of the other 4 detail tables

Scenario a) has the disadvantage of setting up a BPE process which can be CPU intensive.

Scenario b) has the disadvantage that the external database lookups have to be manually coded in Java.

How to overcome the Impedance Mismatch between RDBMS and XML

Within the last few years, vendors of relational database systems like IBM, Microsoft and Oracle have spent a lot of effort in enabling their RDBMS for the world of XML. Their intention was to close the gap between the static relational model and the flexible XML data model.

Result of these efforts was the definition of a standardized function set with the RDBMS for XML processing: the SQL/XML specification.

With the implementation of the SQL/XML specification into the function set of a RDBMS, it is possible to build XML documents by using the corresponding SQL/XML function set like

- XMLDOCUMENT
- XMLELEMENT
- XMLATTRIBUTE
- XMLFORREST
- XMLAGG
- etc.

With the help of SQL/XML a RDBMS can generate valid XML documents out of any set of database tables.

How does the SAP PI JDBC Sender Adapter benefit from SQL/XML?

With the use of SQL/XML the process of gathering data from a RDBMS which is distributed to several database tables is radically simplified. But let us describe this kind of access architecture step by step.

Step 1: Collect the relational data and put it into an XML document

It is a good idea to encapsulate all the SQL/XML and database access stuff into a stored procedure or function which can return result sets which can be processed by the javax.sql.ResultSet class.

This approach has the advantage that all database access and XML setup logic is hidden from SAP PI. By using this Façade Pattern the logical layers are strictly separated: SAP PI only sees the definition of the service offered by the stored procedure/function. Guess what? Hey, here we able to encapsulate the whole database access logic into a set of access function. Greetings from the eSOA approach: build services with operations and data.

Important note: a good SQL/XML implementation guarantees that the XML documents it produces are well formed. That means for example, that not allowed content like the ‘<’ sign is escaped into ‘&lt;’.

Step 2: Setup the JDBC Sender Adapter

In Step 1 we have created a service layer which enables us to get XML documents out of the database. Now we must take care that the result set of the database function being called is properly converted into a XML message for SAP PI.

Therefore the following steps must be followed:

- De-escape the XML string which is escaped by the JDBC adapter.
- Put this XML data properly into the <row>…</row> structure of the JDBC Sender adapter

These steps can be done at two places:

a) within the mapping of the message as a java coding
b) as a SAP PI module

Scenario b) has the advantage that all the processing is done transparently for the succeeding SAP PI mappings.

Conclusion

This was a very short description of how to process XML messages generated by a RDBMS with SQL/XML support. What seems to be very simple has a lot of implications between the interactions of RDBMS with SAP PI.

The main implications can be found in the area of

- Separation of concerns by using the Façade Pattern
  the logic providing the XML documents is separated from the logic of the message mappings. Thereby the dependencies between the database access and the mapping within SAP PI are avoided. A group of database experts for example can provide the stored procedures/functions, a group of SAP PI consultants can do the mapping work
- Performance aspects
  the CPU time for building the XML document is spent with the sending RDBMS. This is optimized for collecting huge bulk of data. This has an impact for example on the sizing of a SAP PI system
- Maintainability
  with this above described architecture model it is possible to improve the maintainability of an integration solution. The responsibilities are clearly defined: the RDBMS gathers the data and build valid XML document, SAP PI is doing the mapping and routing work

With this approach it is possible to setup a service oriented layer on top of a RDBMS. When exchanging data with external systems, it is no longer necessary to think in terms of tables and fields. All content can be exchanged via XML document. This leverages the RDBMS into a whole new level of service.

Copyright