Advanced Data Access Technologies, Inc.
www.adatinc.com (310)395-6889

Our SQL/XML Breakthrough Technology Solution

We are developing the only ANSI SQL processor that transparently and fully integrates native XML. This breakthrough disruptive EII (federated real-time) technology has already been developed, prototyped, and patented. Our SQL/XML integration technology solution has been published in academic journals and industry publications demonstrating its validity and interest.

Relational and XML data are ubiquitous and require integrated processing. SQL vendors and standards groups in this multibillion dollar market have failed to seamlessly perform this problematic integration, resorting to nonstandard and complex solutions that still do not integrate XML fully or seamlessly. Their solutions are also very inefficient, unstable, and require much training producing a huge opportunity for the company that can solve this elusive solution and bring it to market as a product.

Our founder and CTO, one of the original members of the ANSI SQLX Group investigating this integration problem, has solved the SQL/XML integration problem and developed the Holy Grail of SQL-based native XML integration solutions. Remarkably, it is ANSI standard, transparent, and efficient while fully integrating XML in SQL. It solves every outstanding problem while additionally supporting many advanced XML hierarchical capabilities thought impossible with SQL as shown below.

Comparison of SQL Native XML Integration Solutions

SQL Native XML Capabilities	*Our SQL Solution*	*XQuery In SQL*	*Proprietary SQL*	*SQL/XML Standard*
Relational Mathematically Sound				
ANSI SQL Hierarchically Correct				
Transparent Native XML Support				
Small Code Footprint				
Utilizes Semantics in Structure				
Dynamic, Ad Hoc Capability				
Hierarchical Data Modeling				
Full Multi-leg Data Filtering				
Variable Structures				
Structural Transformation				
Logical Network Structures				
Hierarchical Processing				
Hierarchical Optimization				
Hierarchical Join Control				
Fragment Processing				
Efficient Memory and CPU				
SQL Update Using XML Data				
Native XML Input				
Structured XML Output				

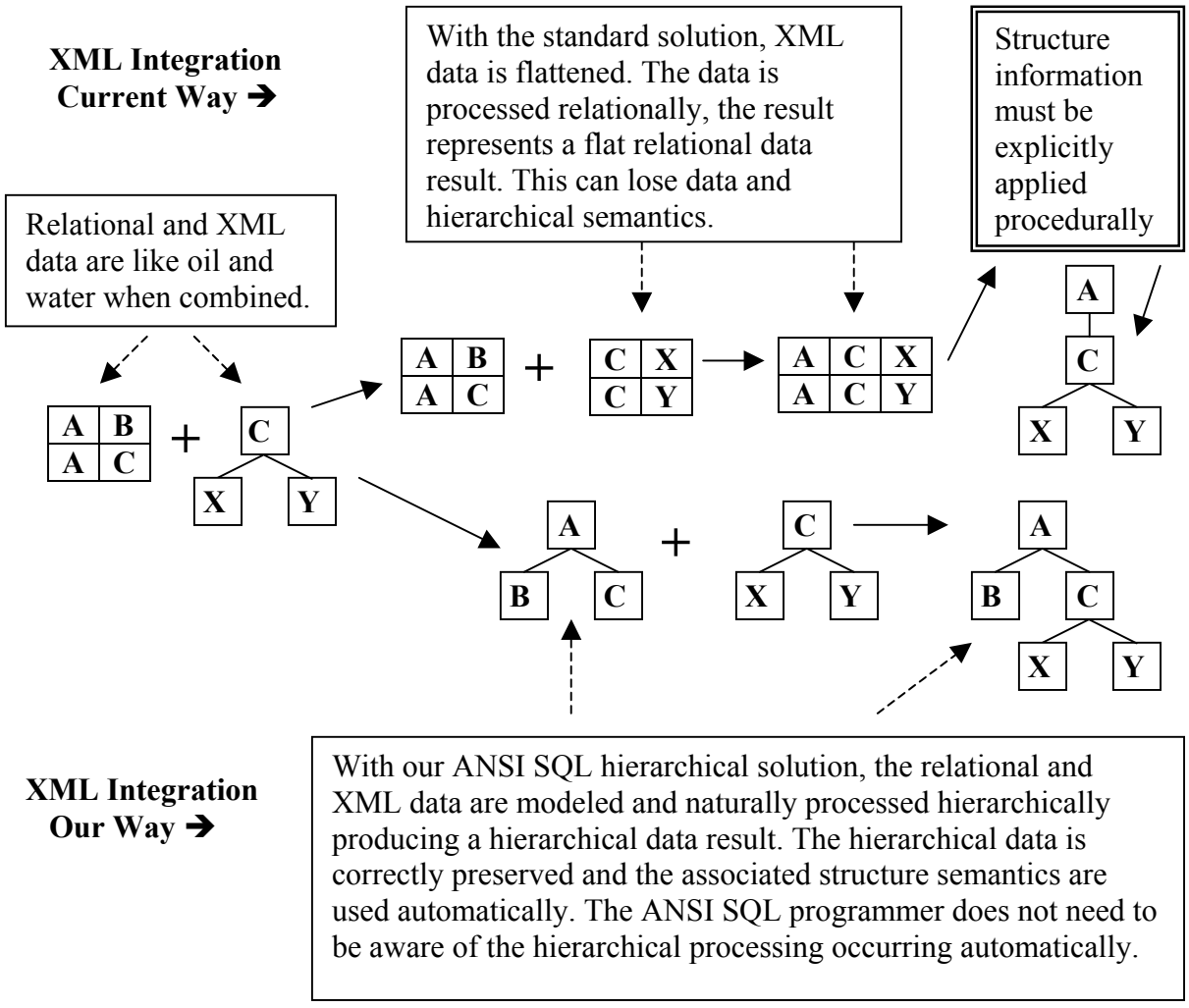
Legend:

ANSI SQL Solution: Nonprocedural and Seamless (no training or coding)	
XML Centric Standard: Procedural (requires training and coding)	
SQL Proprietary Solution: XML Centric (requires training, risky)	
No Support for this feature	

Our Unique SQL/XML Solution Overview

SQL, the older technology, uses flat two dimensional tables, while XML, the newer technology, uses hierarchical structures. There are two possibilities for integrating these two structure types in SQL. The most obvious and commonly used is to flatten the XML hierarchical data into a flat relational structure which removes any hierarchical semantic value in the data. The other possibility, not thought possible in SQL, is to model and operate on the relational structures hierarchically. This would allow the XML and relational data to be fully and transparently integrated if SQL could model and operate on data hierarchical. This has been accomplished and incorporated into our solution enabling our standard SQL to perform total seamless integration even when processing XML's advanced hierarchical operations.

What is truly unique about our hierarchical SQL solution for native XML integration is that no SQL or XML capabilities have been sacrificed to achieve integration. The capabilities of both these languages are naturally preserved and combined. What makes this even more impressive, is that this feat is performed within the current ANSI SQL standard operating syntax and semantics. This is possible because hierarchical processing is a subset of relational processing, and is backed by ANSI SQL's mathematically sound foundation. This gives our unique technology a significant advantage over all proprietary and competing SQL/XML integration standard efforts. This is truly the next generation SQL processor for the XML Internet generation.



Benefits of Transparent Native XML Support in SQL

Besides the advantages that no XML coding, debugging or training is necessary either for XML in general or for the XML feature interface, it also means that that all of SQL's capabilities automatically and naturally operate on XML. XML data does not have to be handled separately by special XML centric functions. SQL's full and complete data processing capabilities are available to the XML data. Additionally, with our solution, standard SQL is operating hierarchically so that XML's transparent integration remains at a hierarchical processing level across all of SQL's capabilities.

Unique Advanced Capabilities of Our Technology

- ANSI SQL standard, no XML centric syntax needed, no risk
- Seamless and transparent native XML access and integration, no training
- Automatically utilizes hierarchical semantics in native XML data, increasing the value of data
- Automatically produces hierarchical result as rowset or fully structured XML
- Supports full hierarchical and dynamic joining of hierarchical structures
- Supports advanced hierarchical processing capabilities introduced by XML
- Eliminates common SQL/XML integration CPU and memory bottlenecks
- All SQL features can operate on native XML dynamically and with no limitations imposed
- Result is both valid hierarchically and relationally insuring the most complete integration possible
- Since the hierarchical result is relationally accurate, it is also mathematically sound
- Hierarchical data modeling in SQL views, makes SQL easier and more accurate to use
- All hierarchical views naturally expand into a single unified and heterogeneous ANSI SQL view
- All heterogeneous data sources take part consistently in all hierarchical processing operations

Our Significant Research Findings

- Hierarchical processing is a valid subset of relational processing
- Complete and flexible hierarchical data modeling in SQL is possible
- Hierarchical processing optimizations also apply to relational processing
- Variable XML hierarchical structures and their processing in SQL are possible
- Hierarchical logical and physical data structure transformation is possible in SQL
- Linking below the root of the lower joined structure is possible and very useful
- The relational engine can be transparently replaced with an efficient hierarchical engine

Our SQL Native XML Integration Solution is ANSI Standard

Our solution is not another proprietary solution where we decided how it will operate and what new XML centric syntax will be added. Nor is it another standardization attempt where external and post processing operations with their procedural XML centric syntax are classified as new SQL standards. These methods do not work well because they can not seamlessly integrate relational and XML data. Our solution does not require adding SQL syntax or defining how to perform the integration— our solution utilizes SQL's inherent hierarchical processing to seamlessly and completely integrate relational and XML data naturally. This is performed inherently using ANSI SQL's inherent hierarchical operation. Our seamless and complete solution remains ANSI SQL standard while achieving transparent XML integration.

Our ANSI SQL Native XML Integration Example

Using stored structured view:

```
SELECT CustName, CustID, ItemName, ItemPrice
FROM CustomerView
WHERE ItemType=”Clothing”
FOR XML Element, ROOT=”Container”
```

XML processing and output is driven transparently by the predefined SQL hierarchical view which defines Customer over Invoice over Item.

With inline data modeling:

```
SELECT CustName, CustID, ItemName, ItemPrice
FROM Customer
   LEFT JOIN Invoice ON Customer.CustID=Invoice.CustID
   LEFT JOIN Item ON Invoice.InvNo=Item.InvNo
WHERE ItemType=”Clothing”
FOR XML Element, ROOT=”Container”
```

This query example uses dynamic, ad hoc inline data modeling to define Customer over Invoice over Item. Views can be used with inline data modeling.

The above two SQL queries automatically produces the following XML:

```
<Container>
    <Customer>
        <CustName> Mike </CustName>
        <CustID> 1056 </CustID>
        <Item>
            <ItemName> Pants </ItemName>
            <ItemPrice> $4.98 </ItemPrice>
        </Item>
        <Item>
            <ItemName> Pants </ItemName>
            <ItemPrice> $5.98 </ItemPrice>
        </Item>
    </Customer>
    <Customer>
        <CustName> Mary </CustName>
        <CustID> 4059 </CustID>
        <Item>
            <ItemName> Shirt </ItemName>
            <ItemPrice> $9.98 </ItemPrice>
        </Item>
    </Customer>
</Container>
```

Simply adding or removing a data item from the above SQL Select list will automatically tailor this XML output.

XQuery and SQL/XML Standard Queries Produce the Same XML But With a Lot More Work

XQuery Example:

```
<Container>
{
 FOR $inv IN document("invoice.xml")//invoice
 WHERE $inv//item/itemtype="Clothing"
 RETURN
  <Customer>
    <CustName>($inv/customer/CustName)</CustName>
    <CustID>($inv/customer/@CustID)</CustID)
        {
        FOR $cust IN $inv/customer/name
        RETURN
        <Item>
        <ItemName>($inv/item/@name)</ItemName>
        <ItemPrice> ($inv/item/@price)</ItemPrice>
        </Item>
        }
  </Customer>
 }
</Container>
```

Even though the XQuery W3C design group says that XQuery is nonprocedural and declarative, it is fairly obvious with it's embedded FOR loops that it is very procedural requiring programming to perform hierarchically.

They also say it is SQL-like and the Select list is simply part of the XQuery FLOWR (For, Let, Order, Where, Return) statement shown. Select list items are spread out all over the query.

For these two reasons, ad hoc processing is not possible, simply adding another Select list item requires programming.

SQL/XML Standard Example:

```
SELECT
  XMLELEMENT (NAME "Container",
    XMLELEMENT NAME "Customer",
      XMLELEMENT(NAME "CustName", CustName,),
      XMLELEMENT(NAME "CustID", CustID),
      XMLELEMENT(NAME, "Item,
        XMLELEMENT(NAME "ItemName", ItemName),
        XMLELEMENT(NAME "ItemPrice", ItemPrice),
   ))) AS "result"
FROM Customer
  LEFT JOIN Invoice ON Customer.CustID=Invoice.CustID
  LEFT JOIN Item ON Invoice.InvNo=Item.InvNo
WHERE ItemType="Clothing"
```

The SQL/XML Standard does not offer hierarchical processing, just XML centric output of XML producing less accuracy.

This XML output is done by using XML centric functions in the SQL SELECT list.

To create the hierarchical structure, these functions require nesting, a form of procedural programming.

This SQL/XML method, like XQuery, also does not support ad hoc processing or the simple addition of a SELECT list item.

Our ANSI SQL Native XML Integration Annotated Example