Our SQL/XML Solution's Capabilities and Operation Overview

Advanced Data Access Technologies, Inc
www.adatinc.com

XML is becoming ubiquitous for storage and universal transport. SQL needs to support XML satisfactorily or some other XML query product will take its place. Unfortunately, all attempts at integrating XML in SQL have resulted in XML centric, inefficient, complex, and incomplete solutions. The primary integration problem has been that SQL processes flat relational data while XML is hierarchical, producing a significant impedance mismatch. The popular approach has been to flatten XML data in order to integrate it with relational data. This introduces significant memory and processing inefficiencies and loses the useful semantics in the XML data, producing an incomplete and complex solution that requires the help of XML centric SQL syntax. The breakthrough solution presented here is transparent, complete, and efficient while remaining ANSI standard. This means there is no training, no risk, just standard SQL. This solution naturally raises ANSI SQL processing to a hierarchical level allowing complete, accurate, and transparent XML integration efficiently within standard SQL. Because this integration is occurring transparently, a description of the XML integration processing occurring automatically in the relational engine is also presented for a full understanding of the complete XML integration process that is occurring naturally.

The ANSI SQL inherent hierarchical processing described here automatically occurs when the data relationships involved are joined hierarchically. The most complex multi-leg hierarchical structures can be modeled precisely by using the standard Left Outer Join syntax. When specified properly, it enables the relational engine to fully operate hierarchically because the associated Left Outer Join's semantics model the exact operation and principles of hierarchical structures. This natural process enables hierarchical data preservation (parents can exist without children, children can not exist without a parent) which is necessary for correct hierarchical results. Hierarchical data preservation produces legs of dynamically varying length. This is handled in relational storage (rowsets) automatically by null values inserted in place of the missing data naturally performed by the outer join operation. Most impressive is that full hierarchical processing across the entire multi-leg structure is performed accurately by the standard operation of the hierarchically restricted Cartesian product. It can support such advanced hierarchical logic as selecting data from one leg of the structure based data from another leg of the structure.

The above description of ANSI SQL's defined operation and semantics show that SQL's natural hierarchical processing is a valid subset of relational processing. This means that the hierarchically valid results are also ANSI standard in their results and semantics. This adds SQL's significant body of relational research and mathematical soundness to this hierarchical processing XML integration solution. Since this hierarchical processing is inherent in ANSI SQL, it operates across and within all SQL features and processing including SQL views. Hierarchical views allow hierarchical structures to be separately modeled and freely combined by dynamically embedding or joining to build larger hierarchical structures.

Even though standard SQL is performing hierarchically, the relational engine is not aware of this. To naturally extend SQL hierarchical processing to other hierarchical data sources and advanced hierarchical operations, the modeled data structure must be dynamically determined at run time. Advanced Data Access Technologies, Inc. has developed and patented a process for this. The hierarchical capabilities described from this point on can transparently utilize this process.

SQL vendors with proprietary and XML centric syntax support may argue that the outer join operation is inefficient. This is true because relational engines are not utilizing SQL's inherent hierarchical structure processing capabilities. The hierarchical data preservation that is naturally occurring enables the access of hierarchical structures to be significantly optimized, allowing unselected and unreferenced hierarchical legs in views to be dynamically removed from access consideration. While the tables in a standard inner join view always require all tables to be accessed, this is not the case with outer join views that are modeling hierarchical structures. Every leg not accessed not only cuts down on the physical access, it also reduces the storage required even though these unselected legs are not stored. This is because the access of these unselected legs causes replications of the stored data for each matched occurrence found. This hierarchical optimization can improve processing efficiency beyond current levels while producing a more accurate result without unneeded replications. It also eliminates the need for multiple tailored optimized views of the same structure.

Native XML and legacy data can be integrated fully and transparently in SQL by modeling physical structures in views in the same fashion as described previously. Modeled in a view, XML becomes integrated at the data item level allowing transparent access. The semantics of hierarchical structures are the same whether they are logical structures like relational data or physical structures like XML. Different hierarchical types of physical structures like XML, structured VSAM, and IMS which have different hierarchical structure implementations still have the same hierarchical structure semantics. This allows all flavors of hierarchical data structures to be combined. This is automatically performed because SQL's structured views used in the query automatically expand into a single unified data structure which can be composed of disparate hierarchical structures. Hierarchical processing logic and optimization apply across the entire unified view enabling the SQL engine to completely process the hierarchical data uniformly and efficiently.

Another problem with all previous SQL-based native XML integration solutions is producing an accurate XML document from a flat relational data result that does not reflect the hierarchically preserved result or structure. With, the hierarchical processing solution, the data is preserved hierarchically. Producing the structured result usually requires procedural processing to specify the structure. But with the hierarchical processing solution, the output structure can be automatically determined from the expanded unified view described earlier. This allows the structured XML to be generated automatically and seamlessly.

XML has also introduced new advanced capabilities such as: node promotion, fragment processing, structure transformation and variable structures. These capabilities are hierarchical operations and can be performed or processed naturally when SQL is performing hierarchically. Most of these operations can be controlled automatically simply by what data fields are selected for output because SQL is a true nonprocedural language. Since these advanced XML features are hierarchical, they operate fully across the unified global hierarchical runtime structure regardless of the form of the underlying logical or physical structures composing the global expanded view.

With XML becoming common, SQL should also be able to update its relational database from XML documents. This can be naturally supported by the hierarchical processing solution. With SQL's XML support operating transparently, XML can simply be modeled in SQL and naturally input into the SQL update operations. Since this hierarchical input is a multilevel structure, data from any node occurrence in the structure can be isolated using the WHERE clause.

To correctly enable ANSI SQL to operate hierarchically, an Outer Join data modeling methodology is followed to model the hierarchical structures to be processed. Hierarchical structures are very powerful and useful because of their structure semantics. Since hierarchical structures have only a single path to each data item, the hierarchical structures are unambiguous so that queries specified on them produce unambiguous results. This allows complex hierarchical logic to be easily performed nonprocedurally. This is a very powerful feature, enhanced further by the fact that these unambiguous structures can be hierarchically joined dynamically using the standard Outer join. This joining operation easily produces larger more complex unambiguous structures that will automatically be utilized in solving more difficult problems. These capabilities of the outer join data modeling methodology are comparable to the advantages gained when the structured programming methodology was introduced. It offers faster more accurate query development.

This data modeling methodology can be used in utility programs to have the hierarchical outer join data modeling views built automatically from XML DTDs and Schemas or from data definition sources for legacy data such as COBOL File Definitions. This greatly reduces the effort, training, and accuracy in constructing the hierarchical views.

SQL hierarchical processing offers a complete SQL/XML integration solution. SQL/XML integration comes in two approaches. The SQL-based real-time native XML support already described. The other approach is a batch ETL utility operation for the bulk pre-loading of XML into relational databases. This approach shreds the XML into the columns of the database tables. There has been no correlated integration between these two approaches themselves. The hierarchical processing solution proposed here for the SQL-based native XML integration can also be applied to the ETL batch solution to enhance its operation saving the data structure in a hierarchical structured view. This structured view can be utilized by the SQL hierarchical processor when the shredded data is accessed. This will enable the shredded XML data in the database to be processed hierarchically, utilizing its structure semantics. The two SQL/XML integration approaches can be seamlessly integrated, sharing technology and producing the same advanced hierarchical results.

There are two SQL/XML working standards that are being developed. These are the W3C XQuery which is being used in SQL, and the ANSI SQL/XML Standard. XQuery was designed for heavy duty XML processing in the XML environment and is very procedure-like with looping and programming constructs. It is a complex new language that was not meant to be used in SQL. In SQL it is incorporated as an interoperation with a complex new procedural language and is not conducive for ad hoc use.

The ANSI SQL/XML Standard has specified very useful SQL/XML mappings, but its XML integration solution is incomplete, XML centric and procedural. XML output is not seamless. XML centric functions are used in the Select list to format XML output produced from the flat relational result. To specify the hierarchical structure, the XML centric output functions are procedurally embedded. This means changes to the SELECT list will involve programming instead of a simple addition to the Select list. This also means that ad hoc SQL processing is not possible.

In conclusion, our hierarchical processing solution is ideal for SQL XML native XML support for the Internet generation. It is transparent, complete, hierarchical, efficient, and ANSI standard by today's standards— keeping SQL pure in syntax and operation. Other benefits are significantly reduced: training, programming, and risk. The XML support code is also reduced since standard SQL is automatically performing most of the integration, and doing it at a more accurate and hierarchically powerful level that automatically utilizes the semantics saving on computer cycles.