Extended Set Theory – A Summary

Tim Ellis (tim.ellis@sri.com)

Extended Set Theory (XST) was originally developed by D. L. Childs under an ARPA contract to address the limitations of Classical Set Theory (CST). At the root of the problem is the set membership definition of CST, which defines sets and the results of set operations based on a single membership condition: content.

CST: $x \in A$

Given set $A = \{x, y, z\}$, x is a member of set A iff x is contained in set A

CST does not capture the structure of data (e.g. order or hierarchy), only its contents. Since computer based data systems must manage and adhere to a defined physical structure of data in order to store, access, and manipulate the data within memory and persistent storage, CST is an insufficient basis for implementation of data management systems. For example, CST typically defines the concept of an ordered set, or n-tuple, in terms of the Kuratowski definition for ordered sets:

$<a, b> = \{\{a\}, \{a, b\}\}$

This can give unexpected results for set operations such as intersections. For example, using the above definition for an ordered pair in classical set theory:

$<a, b> \cap <a, c>$ yields $\{\{a\}\} = \{\{a\}, \{a\}\} = \{\{a\},\{a,a\}\} = <a, a>$, and

$<a, b> \cap <x, b>$ yields $\varnothing$ (null set)

Neither of which are particularly useful in developing information processing systems where records of elements must be handled consistently and in precise order. It gets worse for larger n-tuples as this CST formulation is recursively nested resulting in very large and complex sets to represent the n-tuple elements.

The Relational Data Model (RDM), which underlies virtually all relational database systems in use today, and which is based on classical set theory, limits the mathematical foundations for relational database systems. To compensate for this weakness, relational database systems handle the structure of the data programmatically and in proprietary ways, which is then abstracted from the application level via the SQL interface. This approach, however, implicitly requires significant overhead in terms of indexes and secondary bookkeeping structures, which consume storage and processing resources.

XST resolves the problem in CST by defining set membership in terms of two independent conditions: content & scope, and thereby forms the basis for a formal capture of both data content and structure.

XST: $x \in_a A$

Given set $A=\{x^a, y^b, z^c\}$, x is a member in scope 'a' of set A iff it is contained in scope 'a' of set A.

(Note: the use of superscripts for element scopes is merely a notational convenience typically used in extended set notation and does not imply exponents. Scopes may also be sets as well, supporting very complex structures.)

Scope is used to represent both order and hierarchy, or structure, of the content. XST extends the set operations to define both membership conditions of the result set and leads naturally to an extended set of axioms for set operations (Blass, 2011). CST set operations are a special case of XST operations when null scopes are used. So for the intersection example posed earlier:

$$\langle a, b\rangle = \{a^1, b^2\}, \langle a, c\rangle = \{a^1, c^2\}, \text{ and } \langle a, b\rangle \cap \langle a, c\rangle \text{ yields } \{a^1\}$$

This is not only more intuitive and better suited to data processing systems, it also scales nicely for large n-tuples as well.

This simple extension to CST provides a solid mathematical basis, leading to several benefits for processing and managing large data sets. Set elements in XST can themselves be sets, as in CST. In addition, scopes can also be sets, supporting unlimited complexity in capturing the order and hierarchy of any data model. In current practical implementations, scopes are typically integers to capture order and atomic names to capture attribute scopes such as the columns of a table or XML tags.

In data management systems a key to high performance is minimizing the amount of data flowing across the slowest boundary in the system, the storage I/O interface. By modeling the structure and content of data independently and at the same level of abstraction, extended set processing can be independently performed on either content or structure, or both at the same time. During the processing of a set operation (e.g. executing a query), new intermediate and resultant sets are created with structure and content specifically tailored to the requested operation, and with a mathematical identity that can be tracked and used in subsequent set operations to improve performance. These new sets contain a high density of data relevant to the recent set operations and significantly improve performance by excluding irrelevant data from subsequent set operations. Here, relevant is defined as the minimum data needed to respond accurately to a specific query operation.

When executing a requested set operation, the XSP engine first conducts algebraic search (i.e., it uses equations) and substitutions to create the lowest cost equivalent set operation based on existing sets and subsets. This is a fast process and requires very little I/O. As a result, the set operations with minimum I/O are selected and executed resulting in overall performance improvement and continual optimization.

In contrast, relational database systems often need to re-transport unused or irrelevant portions of RDM table records continually across the slow I/O boundary. This is a significant performance barrier in RDM based systems. Memory caching schemes are a partial solution, but the cached data may still contain significant amounts of irrelevant data, it must be carefully managed and prioritized, and it cannot be saved in memory indefinitely.

This reformulation process in XSP is referred to as Automatic Data Restructuring (ADR). The physical data model is built by the system during operation and is continually optimized as a byproduct of all extended set operations, resulting in minimizing the movement and processing of irrelevant data. For large databases typical of IC data centers, the ratio of relevant data to irrelevant data is typically very, very small for any specific query (i.e. most of the data in the database is not required to produce the requested result data set). Elimination of irrelevant data from any data movement and processing as early as possible in the set operation results in significant speed up of processing performance and is at the heart of XSP’s high performance.

This concept of the physical data structure being defined and redefined by the set operations is called an “operation centric” data model. In practice, only the desired logical data model needs to be defined prior to deploying a XSP based system in order to describe how the external applications will view and access the data. It can be modified at any time, even during operation, by redefining scopes and their relationships. A natural consequence is the capability to have multiple data models and multiple views of the data. The physical data model is dynamically built and optimized by the system.

In contrast, RDM based systems can be described as “structure centric” data models in that their structure at both the logical and physical levels must be defined in advance and strictly adhered to throughout the life of the data system. This requires extensive analysis and data model design effort in an attempt to predict exactly how the system will be used and how the data will be ingested, stored, and accessed. Changes to the data model after the relational database system has been built and deployed are very expensive, if not impossible in some circumstances.

References

Blass, A. C., Childs D L, (2011). *Axioms and Models for an Extended Set Theory.* Retrieved from University of Michigan, Mathematics Dept: http://www.math.lsa.umich.edu/%7Eablass/XST_Axioms.pdf

[First drafted November 2006, revised in collaboration with DL Childs July 2015]