A New Approach to Enterprise Data Organization – A Cuboid

Enterprises are generally overwhelmed with data, making the ability to store, process, analyze, interpret, consume, and act upon that data a primary concern. For large-scale, multi-national organizations and those in heavily regulated industries— such as finance, supply chain, or those covering multiple industry verticals — the situation becomes even more complex and challenging. The question then becomes, how do enterprise best work with all this evolving data and transform them into actionable information which can be used as part of an enterprise information system?

Business Information Lifecycle Flow

Since most enterprises are carefully measured on the evolution of information, specifically how does a given set of information change over time and how easy it for all process participants to provide their input on this change, you tend to see the following lifecycle:

- Newly-Created, Work-in-Progress (WIP), Approval Cycle, Approved
  or
- Copied-from-Previously-Approved, Work-in-Progress (WIP), Approval Cycle, Approved

For most businesses, data organization tends to be very rigid when it reaches the approval stage and is handled well by common transactional systems. The problem area is business information in the WIP stage. It's during this stage where businesses have struggled finding a suitable data organization approach that works effectively and yet this is where the highest business value is derived. This is where business users need to access the right data in a meaningful format, apply relationships between the data either for business logic or presentation, understand how it's changed, make updates, and then share updates with others as part of a decision-making process.

Current Methods for Organizing Enterprise Data

The current methods for organizing and managing enterprise data can generally be classified into the following types of information systems:

1. Custom data structures with custom access methods available independently (pre-database)
   - The lack of standardization caused this to be replaced with newer technologies
2. Custom data structures with access methods encapsulated in the form of classes (object oriented databases)
   - While this offers the benefit of class management in a centralized manner, the behavior and APIs are not standardized and are known only to the class developers.
3. Documents with document editors (Microsoft Word)
   - This offers a standard format and therefore a set of standard access methods so that a large community of developers and users can participate in the process of development and maintenance. However, the document is typically serialized in terms of updates because of the inability to branch and merge.
4. Relational data base with a standardized access method in the form of SQL language
   - This offers a standard way to access data and therefore a large community of developers and users can participate in the process of development. However, the inability to branch and merge data into the relational model poses a challenge. Typically, only one value is held in a single data field.
5. Keyword-Value pairs or Hash tables with a standardized access method (No SQL)

- Same advantages as relational, however, the inability to branch and merge data into the No SQL model poses a challenge. This data organization method also cannot express complex models.

If we focus on information in the WIP stage, there are various problems which most enterprises struggle to address with the conventional organization approaches outlined above. These problems include a lack of usability when updating data, difficulty in consolidation, and the non-availability of historical data. Traditional systems work at the record-object or line level making it difficult to make a set of changes across multiple related items (all of my forecast items rather than 1 rows at a time) and commit them as a single related update. To reduce the time in consolidation, the data must be automatically branched and merged as required for simultaneous and parallel information operations, otherwise serial operations tend to extend business cycles. Security must also be ensured while managing the information. For example, the users of an enterprise should have access to information regarding all the departments within the enterprise, but an external vendor should not have access to this information. Further, in order to use the information for planning, historical information must also be available. Hence, the information must be versioned or chronicled so that data of a particular time or state can be accessed later.

A New Approach to Enterprise Data Organization

There exists a need for a new method and system for managing information within an enterprise that allows viewing, comparing, updating, and reverting to historical data. The system should allow information to be viewed in a manner which is useful to the business users of an enterprise. It should also allow the users to compare current information with previous information and analyze trends. Further, the system should also allow reversion/auditability to a previous state. Finally, it should also address the problems arising due to lack of data security and difficulty in consolidation.

Boardwalktech has been awarded a patent for a “method and system for versioned sharing, consolidating and reporting information” (United States Patent 7383272) which is a new digital ledger approach to enterprise data organization. It’s called a **Cuboid** and has the following characteristics:

A Cuboid is a time dimension rectangle where the data elements are immutable addresses with the ability to position them in an X (row), Y (column), and T (time) coordinate system

- It supports standard access methods
- It supports the ability to branch and merge values
- It’s able to express complex models

Distributed Architecture

- A single immutable address is the atomic element of a Cuboid
- This is distributed to “N” nodes where a program or user can link new values to this address at each of these nodes
- At any given time one or more of the nodes can reconcile the values to single value
- This allows distributed nodes to participate collaboratively in the evolution of data

Access Control

- The data is modelled as a cuboid with (X,Y,T) coordinates and access can be controlled on which immutable addresses of X,Y,Y are available to any node.

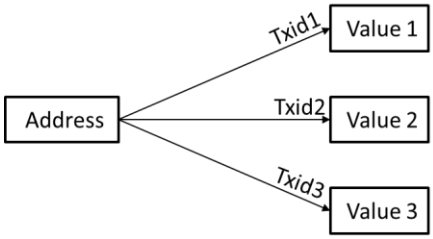
Time Dimension

- The Cuboid time dimension allows multiple values to be linked to a single immutable address differentiated by time.

- This allows branches to be established, managed and merged. Changes made at a single time can be grouped together thus allowing dependencies to be tracked.

Immutable Address – The Building Blocks for a Cuboid

For a Cuboid, the world of information is made up of immutable addresses. An address is a global coordinate in the information space. The addresses are the atomic elements of this data organization. The addresses are linked to values with immutable transaction identifiers. The transaction identifiers contain the time stamp and the user who created the link between the value and the address.



Operations for a Cuboid

The CRUD operations supported are:

- **Link**: Links a value to an address with a transaction ID
- **Mark as Inactive**: That marks a link as "Inactive"

These two operations support the required CRUD operations as follows:

- **Create**: An address is created when a value is linked to it the first time.
- **Update**: An address is updated when another value is linked to it. The first value is not overwritten but a new address-value link is created and assigned a transaction identifier.
- **Delete**: No values are deleted but address-value links are marked as "Inactive." This allows deleted values to be made active if the reason for the deletion is overridden by a new reason that requires the value to be made active again. This allows the support of "Undo."
- **Read**: The entire set or subset of address-value links associated with an address can be read based on specification.

Semantics

The semantics of the addresses are derived in several ways.

- By lining up the addresses as rows, columns, or rectangles of rows and columns. Inside a row, the columns can be assigned a sequence and inside a column, rows can be assigned a sequence, or inside a rectangle the rows and columns can be assigned a sequence. The relative position of the address provides semantics in the information space.
- By embedding the address class in objects.
- By embedding the address class in documents.
- By tagging the address with information that is present in the document or can be manually entered

This provides a powerful capability for associating Cuboid cells via business logic or position across multiple enterprise information systems without having to code the relationship.

Semantic views

These immutable, addressable cells are available within the context of multiple semantic views which can simultaneously be shared and individually updated while still reconciled (merged) by virtue of the evolving time dimension of the business.

- A single address can be available for CRUD operations in several views concurrently. Each of the views can serve its own purpose. For example, relevant cells in an Excel file can be assigned an immutable address. Once this cell is managed by Boardwalk, then it can manifest itself concurrently in a database object, a Word document or a different Excel document. The "cost" a specific SKU item can be used to calculate the margin on forecasted revenue, the current value of inventories, and the projected cost of the replenishment – all from the same Cuboid cell without painful replication.
- Semantic Tags are attached to the immutable Boardwalk address and not to the business presentation layer address. This allows for reliable link between the Tag and the presentation layer value.
- Multiple tags can be attached to the Boardwalk address. These tags can be static or dynamic in nature. This allows for simplified Tag representation and management.
- These tags can be attached to a model file Boardwalk address by a subject matter expert. When the user makes a copy of the model, an indirect linkage is created between the Tag(s) → Model address → Instance address.
- The advantage is that one can add a new Tag to the Model and one can extract new Tag data from existing instance files without opening them or writing a line of code.

Transaction Identifiers

A new value enters into the system by creating a new link between the value and a new address or an existing address. The link is marked by a transaction identifier. The transaction identifier contains the time, the user ID, and the reason for the link. The transaction identifier provides a way to The semantics of the addresses are derived in several ways that were "deemed" to be created at the same time.

The transaction identifiers are lined up in an ascending sequence which allows an easy way to compare the transactions and consequently the links that were associated with those transactions. This allows the ability to look at the information as:

- Links created as of a specified transaction
- The links that were created between two specified transactions
- The links that were created before a specified transaction
- The links that were created after a specified transaction

This provides a powerful way to branch the information into multiple semantic views and reconcile them to a single value as necessary. This also allows for synchronization between isolated offline views. This improves process efficiency and removes workflow serialization.

Transaction Chains

The transaction identifiers are chained together to create a chain of transactions that can provide traceability of all changes. This forms the basis for a digital ledger which is an essential part of any time-based business activity and also for emerging technology areas like Blockchain applications.

Non-editable Information

The information that is entered into the system cannot be modified. Further, no information is lost because new information simply forms new links without updating the past.

Access Control
Access can be controlled on any of the CRUD operations by explicitly declaring permissions on the addresses. Further, the semantic views can provide access control by restricting any of the CRUD operations in the manner of a SQL query constraints.

Integration
Cuboid information needs to be connected to the extended enterprise information network. Cuboid addresses can be mapped or integrated with enterprise normalized and denormalized data sets. The Cuboid address can be updated using APIs. Any integration update is captured as a new value attached to the address with a Transaction identifier and is reliably delivered to all semantic views the next time the view refreshes its data.

Programmatic Support
The Cuboid supports the four popular ways to perform CRUD operations popular within the programming community today:

- SQL Language
- Java and other Object-oriented languages
- Browser, Java Script, HTML, JSON etc.
- Document Editors like Excel, Word and PowerPoint

How Does This New Data Organization Break Down the Walls Around Information Islands?
An information island can be called a domain. The world of information can be broken down into two areas.

1. Intra-Domain: The data organization within the domain is created to optimize the usage inside the domain.
2. Multi-Domain: This provides the ability for the domain to collaborate with multiple parties and arrive at a consensus.

The Cuboid data organization provides a neutral and efficient way to organize information across both of these areas. For example: An address can participate in an Intra-Domain purpose as well as Multi-Domain purpose. A simple example would be the negotiated value of a multi-party purchase which then participates in the Intra-Domain purposes of finance, operations etc. through other views.

WIP Efficiency
During Work-In-Progress, the information is collaborated upon and evolves through multiple branches of evolution. In existing methodologies, branching is given limited support. This invariably results in serialized work that then results in inefficiency. The ability of the immutable Cuboid address to participate in multiple views and therefore in multiple branches cuts through this inefficiency and allows the information to be developed in parallel in multiple branches. Since all values are linked to a single address with the transaction identifiers, values from all branches are preserved. At the end of the WIP, when the values are reconciled to provide a final value, the links that hold the selected values can be promoted to the next stage of the life cycle. This provides a way of merging the branches into a single main branch.

Change Management
Existing data organization methodologies also give limited support to change management. By definition, change management requires an understanding of the before and after states in a given temporal data information flow. The only way you can capture this is through a versioning mechanism which captures the before and after states of a given atomic cell. A Cuboid captures every value change

as an insert into a data organization schema which captures the old and new values or a given atomic cell. This provides a natural method for supporting change management.

Understanding changes:
As each set of changes are tagged with a Transaction identifier the user can benefit from the following

- Access history at an address, row, range level at any point in time without opening multiple files
- Compare historical changes as of and between tow time stamps to understand changes
- Reduce review time to understand changes

Revision Management and Information lifecycle
When users or process milestones need to apply a workflow step on business information, their signatures can be attached to a specific baseline or transaction identifier. This allows the information to evolve further without losing the time stamp and information as of that time when the signatures were requested. Information change management allows executives to confidently understand the changes between iterations without spending time to go through every detail as to what changed and by whom.

Distributed Architecture

- The Immutable Addresses can be distributed at multiple nodes to be available for CRUD operations by any of the popular data editors such as java programs, browser forms, document editors, mobile applications.
- The distribution can be Peer-to-Peer (decentralized) or central server with slave nodes (centralized).

Technology
The Cuboid data organization can be implemented on any of the technologies such as SQL databases, Distributed File Systems or NO SQL.

Current State of Development
Cuboid Data Organization has been developed and successfully deployed for the following scenarios:

- Server Technology: SQL Databases
- Information Editors: Excel, Browser and Mobile Phone Apps.
- Centralized server architecture.

Current Implementations

The deployments have covered a wide range of areas from finance to manufacturing. Examples include taxation by PwC and multi-party collaboration and reconciliation for companies such as Apple, Coca-Cola, Levi Strauss & Co and Rockwell.

The Cuboid Expressed in Business Terms: A Digital Ledger
Enterprise information is widely interacted upon by business users in the form of a ledger which is a set of information associated via a set of meta data describing the data itself (columns) and then a set of rows representing instances of the data which get updated over time to capture the changes in process data. This definition – a digital ledger - aligns perfectly with the design of the Cuboid information model and how it works with enterprise information.