December 2001

A Knowledge Management Scheme for Meta-Data: An Information Structure Graph

Choon Lee
Kookmin University

A Knowledge Management Scheme for Meta-Data: An Information Structure Graph

Choon Yeul Lee
Kookmin University

Abstract

For an effective management of data, we need various kinds of meta-data knowledge such as what they mean; how they are created; how they are organized, etc. Some of them are managed as a database schema; others are not. This article proposes a scheme - an Information Structure Graph (ISG) - to represent these kinds of meta-data that are not managed as a database schema. An Information Structure Graph describes structures that data are created or derived from other data. It is a directed graph, where nodes represent data objects and arcs connect nodes. That is, it is built on a database schema and extends it to include data creation structures. An ISG can be used to answer following questions about data; which data are comparable to each other? If some data includes wrong values, what impacts does it have on other data items? How long does it take to update a data item after something has happened in a real world?

Keywords: data management, meta-data, derivation rule, data models

1. Introduction

To provide relevant information about data is an indispensable task in data management. It helps users understand data correctly. Especially, with regard to data usage, the primary concern has been misuse of data, which results from users ignorance of data semantics [Brackett 1996] [Redman 1995][Liepens and Uppuluri 1990]. This kind of information is stored as meta-data in data management.

To manage meta-data effectively, we need a representation scheme just like we use a data model to manage data itself. In the relational data model, meta-data are primarily about data items and tables. The relationships between tables are limited to referential integrity rules.

Semantic data models have been proposed to extend the limitations of the relational model. They include some features like IS-A relationship, many-to-many relationship, etc. However, these features are mostly for a database schema and describe how data are to be organized in a database [Teory et. al. 1986], [Hammer and McLeod 1977], [Shipman 1981], [Abiteboul and Hull 1987].

Another kind of meta-data is about the processes that data are created. For example, if a quantity on hand is measured by an observation, it represents the quantity that is stored in a warehouse. However, if it is calculated from the previous month's quantity by deducting an outbound quantity and adding an inbound quantity, it represents the number that appears on an inventory book. If we can conceptualize this knowledge, we might utilize it to improve data quality of a quantity on hand; to detect that these two kinds of quantities on hands are not same data items and thus they are to be managed distinctly, etc.

Based on this idea, a scheme is proposed to materialize data creation structures. A formalization of data creation structures includes several tasks. The first is to define scopes of data creation structures. Data creation structures are not a well-defined terminology.

They might describe who creates data, when they are created, how they are created, to name a few. The second is to develop a formal scheme. Some models have been proposed for process modeling. SADT, DFD are typical ones [Gane and Sarson 1979][Yourdon 1990]. However, they are defined separately from a database schema and thus do not provide a comprehensive view to data management and data creation structures. The third is to propose a query system. A query system helps us to analyze data creation structures and to understand their implications.

In the next section, scopes of data creation structures are described. In section 3, a scheme is proposed to formalize them. It is formalized as a directed graph, which shall be called an Information Structure Graph (ISG). In section 4, ideas are proposed to apply an ISG to data management. At last, in section 5, implications of the research are summarized.

2. A Description of Data Creation structures

2.1 A Classification of meta-data

To understand data, we need various kinds of information about data, which shall be called meta-data. The followings are typical examples of meta-data:

- How data are structured?
- Which values can be assigned to data
- What do data mean?
- What kinds of activities and/or processes are performed to create data?
- How data are calculated, derived and/or observed from other data items or a real world?

The above questions can be classified into two categories. The first three are about a database schema, which is used to organize and store data. The fourth and the fifth are about processes, which are performed by a person and/or a system to create data. Based on this categorization, meta-data are often classified into the following two categories:

- Meta-data about a database schema
- Meta-data about processes

2.1.1 Meta-data about a database schema

A database schema describes schemes that data are organized in a database system. When we say meta-data, we usually mean this one. Meta-data about a database schema are stored in a data dictionary in conventional database systems - a relational database system, for example. However, to describe this kind of meta-data more effectively than we do using a data dictionary, we need a data model that describes semantics among data objects. For this purpose, semantic models have been proposed.

Semantic data models expand limitations of the relational data model. Typical ones are the Entity-Relationship model [Chen 1976][Teory et. Al. 1986], SDM [Hammer and McLeod 1977], the functional data model [Shipman 1981] and IFO [Abiteboul and Hull 1987]. In comparison to the relational data model that supports two kinds of data objects – relations and attributes, these semantic data models support various types of data objects. And, they also support various kinds of relationships to better represent the semantics of real-world relationships among data.

2.1.2 Meta-data about processes

Processes describe activities that are applied to data. Thus, process meta-data have been described as a part of information system specifications. For example, SADT (structured analysis and design technique) and DFD (data flow diagram) include them as a part of process specifications [Gane and Sarson 1979] [Warnier 1981] [Yourdon 1990].

Nowadays, there have been attempts to model processes meta-data for the sake of data management. One is an information manufacturing system [Wang 1995, 1998]. It decomposes information production into data units, vendors, data quality blocks, processing blocks and consumers. A data unit supplied by a vendor passes through data quality blocks and processing blocks. Then, it is delivered to consumers.

Another one is FIP model [Redman 1996]. It models data processing as applying FIP (functions of information processing) to produce an OIP (output information product) from IIP (input information product). For example, the following models processes of creating data_set_C from data_set_A and data_set_B:

Data_set_A FIP Data_set_B = Data_set_C

Here, data_set_A and data_set_B are IIPs, and data_set_C is an OIP. In FIP model, information processes are modeled by six primitive functions - associate, filter, prompt, queue, regulate and transmit.

2.1.3 Meta-data about Data Creation Structures

As shown in the above, database meta-data and process meta-data do not share a common presentation scheme. Thus, it is not easy to relate meta-data about processes with data objects in a database schema. In other words, to find processes applied to a data object, it is necessary to go through all details of process specifications.

To understand meta-data and manage them effectively as a whole, we need the third artifact that connects these two categories of meta-data. We shall call it data creation meta-data. Data creation meta-data answer the following questions within a framework of a database schema:

- What kinds of data items do we need to create a certain data item?
- How data are calculated, derived and/or observed from other data items or real world?

2.2 A Classification of Data Creation Structures

In general, data creation means two different things. One is to create new data objects; the other is to create values of existent data objects. For example, we can create new customers by inserting customer records into a database. Or, we can create new addresses of current customers by updating their addresses. The former means that we create new data objects and assign new values to them. And, the latter means that we assign new values to the existent data objects.

In addition to these two cases, we may create reports or documents. These are assembled from existent data objects in a database. We treat this kind of data creation as creating a new data object without creating new values[1].

By combining these two perspectives of data objects and values, we categorize data creation into four classes. That is, depending upon whether data objects and/or values are created, data creation is classified into four categories; however, if neither is created, it means that nothing has been created. In this sense, the fourth one is meaningless and is not

[1] This one corresponds to creating views (virtual tables) in a relational database.

included in the categories of data creation.

In sum, we categorize data creation into three distinct categories of object creation, value update and object assembly, as shown in Table 1. Object creation and value update are the first and the second cases of data creation, which are discussed above. And, object assembly is the third case.

[Table1] Classification of Data Creation

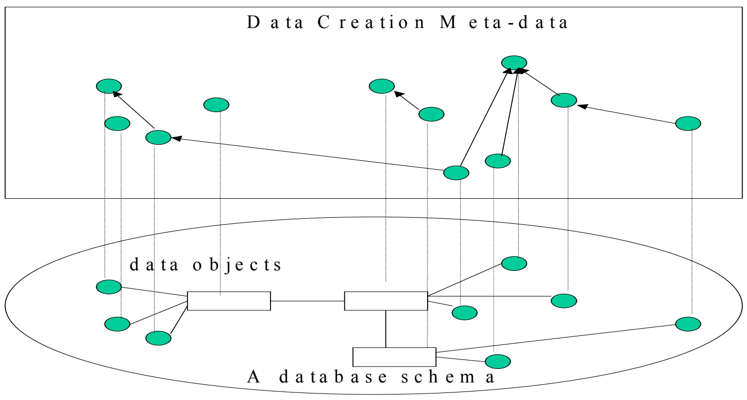
		Values	
		created	not created
Data Objects	created	Object creation	Object assembly
	not created	Value update	-

(1) An object creation is to produce a new data object. A typical example might be to calculate a purchase order quantity based on a quantity on hand and a safety stock. However, some data objects such as customer order are captured without any reference to other data objects. They are called primitives or primitive data objects.
(2) A value update is to update a current value of a data object. A typical example is an update of a quantity on hand when goods are deposited or shipped.
(3) An object assembly is to produce a new data object without creating new data values. Typical examples are composing an inventory report from inventory items or generating a sales report from customer orders.

2.3 A conceptualization of data creation meta-data

To conceptualize data creation meta-data, this research proposes to add data creation structures onto a database schema. We assume that data objects are pre-defined in a database schema. For each object in the schema, data creation meta-data is defined as shown in Figure 1.

[Figure 1] A database Schema and Data Creation Meta-data



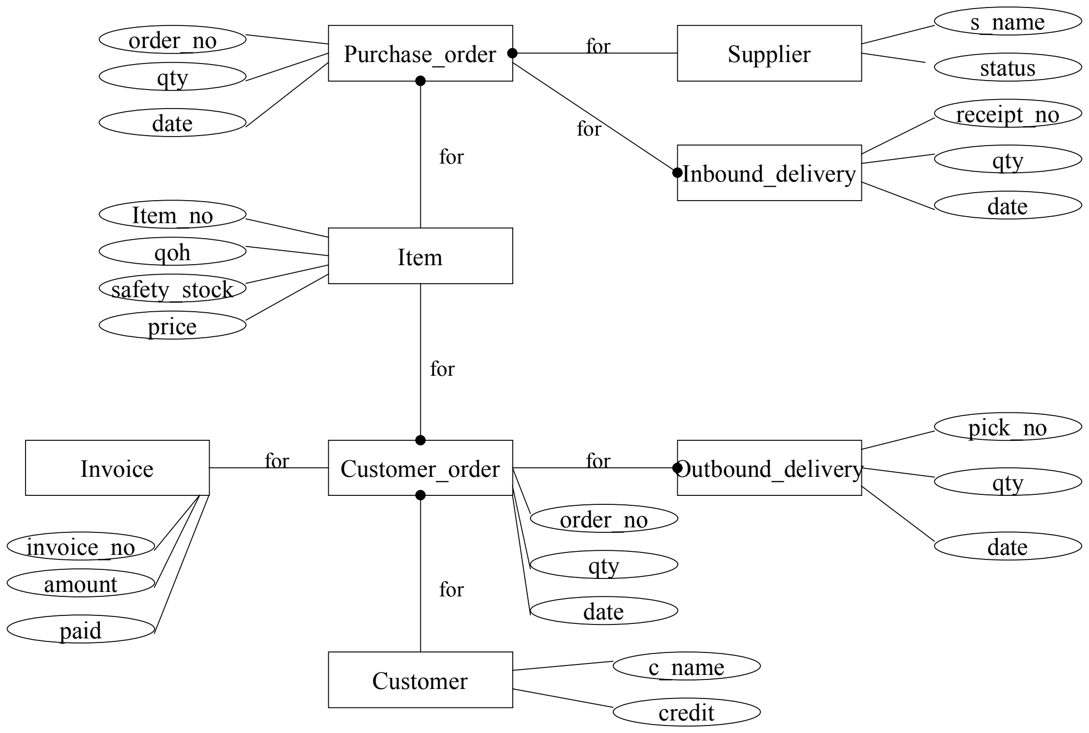
We assume that a database schema is modeled based on the Entity-Relationship (ER) model [Chen 1986]. The ER model is selected because it is the most popular one for conceptual database design. We assume that ER diagrams are designed as follows:

- A database schema is composed of two kinds of data objects - entities and attributes. Entities are denoted by rectangles and attributes are by ovals.

– Relationships are defined between data objects. We assume that all relationships are binary ones and the relationships have cardinalities of one-to-one or one-to-many.

Figure 2 depicts a typical ER diagram. In the diagram, small circles at the end of the lines denote many sides of the relationships.

[Figure 2] Inventory database schema



Data creation meta-data are represented for each data object in an ER diagram. For example, let us assume that an inventory database is designed as shown in Figure 2. Then, data creation meta-data are defined for each attribute like order_no, qty, date and each entity like purchase_order.

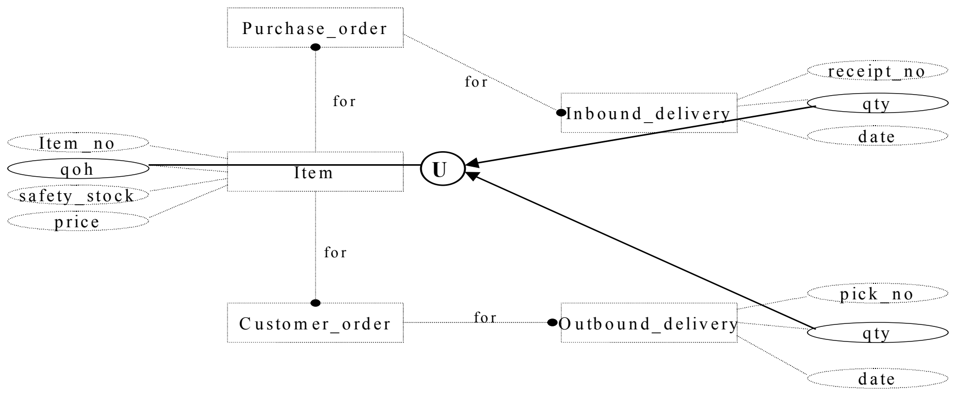
To define data creation meta-data, we propose an information structure graph (ISG), which is based on a conceptual graph. A conceptual graph has been applied to systems requirements formalization, especially for hardware systems [Cyre 1997] and semantics of natural language [Sowa 1984], etc.

An information structure graph is a directed graph that connects data objects. In an ISG, a parent node is an output data object and children nodes are input data objects.

For example, let us assume that we update inventories by either an inbound delivery or an outbound delivery. Then it is conceptualized as shown in Figure 3. In the figure, inbound_delivery.quantity[2] and outbound_delivery.quantity are input data objects. And, item.qoh is an output data object. A symbol assigned to item.qoh, ⓤ, represents that it is a value update.

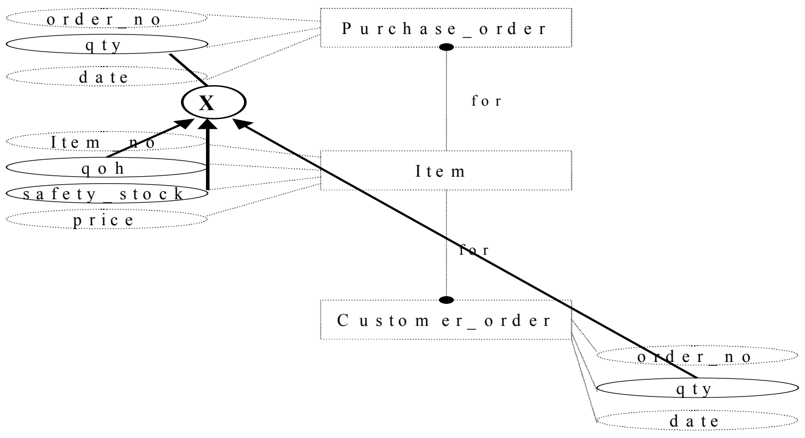
[Figure 3] An Information Structure Graph of a Quantity On Hand

[2] In this paper, an attribute name follows an entity name with a period as a delimiter. Thus, inbound_delivery.quantity denotes quantity attribute of inbound_delivery entity.



Further, let us assume that a customer order quantity is compared against a quantity on hand. If the calculated one is below a safety stock, a purchase order is issued. Then the process is conceptualized as shown in Figure 4. It depicts that purchase_order.qty is produced from item.qoh, customer_order.qty and item.safety_stock. A symbol assigned to order quantity, ⓧ, represents that it is an object creation.

[Figure 4] An Information Structure Graph for an Order Quantity

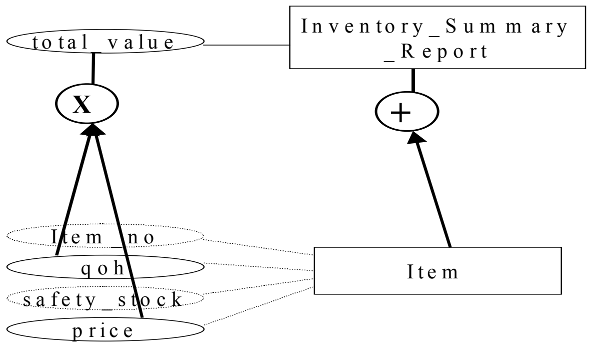


In an information structure graph, an object assembly depicts a composition of an output data object. Let us assume that an inventory summary report is generated from items by calculating a total value of quantities on hand as shown in Figure 5. In the figure, an object creation symbol (ⓧ) denotes that a total value is calculated from a quantity on hand and a price. The summary report is a combination of details of inventory items with the total value. The symbol assigned to inventory_summary_report, ⊕, denotes that it is an assembly of items. In an ER diagram, an entity is an assembly of attributes. Thus, total_value is automatically included in inventory_summary_report.

In most cases of an object assembly, generated data objects do not exist in a database schema. Thus, as shown in Figure 5, they are included in the information structure graph in addition to database schema objects.

To summarize discussions, an information structure graph depicts input data objects, output data objects and categories of data creation. Processes or functions are not described in detail.

[Figure 5] An Information Structure Graph for an Inventory Summary Report



3. A Formal Description of Information Structure Graph

An information structure graph is a directed graph composed of nodes and edges.

Definition 1*: An information structure graph is a graph G = <V, E, μ> such that,*

(1) V are nodes that represent data objects, and E are edges that connect nodes. I.e., <V, E> is a directed graph.

(2) μ is a function from V to the set of data creation types {$\odot$, $\otimes$, $\oplus$, ⓤ} that satisfy the following conditions:

(a) $\mu(\nu) = \odot$, iff ν is a leaf.

(b) If $\mu(\nu) = \otimes, \oplus,$ ⓤ, then there exist children and the children of ν are distinct nodes.

Where, $\odot$, $\otimes$, ⓤ, $\oplus$ denotes primitives, an object creation, a value update and an object assembly, respectively.

$\mu(\nu)$ is called a data creation type of a data object ν. For the sake of simplicity, we include input data objects into data creation types. Then, as shown in the followings, $\mu(\nu)$ denotes input data objects as well as a data creation type of ν.

(1) $\mu(\nu) = \odot$ denotes that “ν is a primitive data objects and there exists no input data object. I.e., there exists no edge with head ν.”

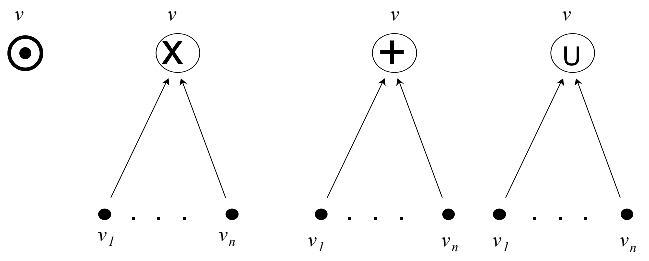
(2) $\mu(\nu) = (\otimes, \nu_1, \ldots, \nu_n)$ denotes that “$\mu(\nu) = \otimes$ and ν has n children $\nu_1, \ldots, \nu_n$, which are input data objects. And, there exist exactly n edges with head ν and their tails are $\nu_1, \ldots, \nu_n$.”

(3) $\mu(\nu) =$ (ⓤ, $\nu_1, \ldots, \nu_n$) denotes that “$\mu(\nu) =$ ⓤ and ν has n children $\nu_1, \ldots, \nu_n$, which are input data objects. And, there exist exactly n edges with head ν and their tails are $\nu_1, \ldots, \nu_n$.”

(4) $\mu(\nu) = (\oplus, \nu_1, \ldots, \nu_n)$ denotes that “$\mu(\nu) = \oplus$ and ν has n children $\nu_1, \ldots, \nu_n$, which are input data objects. And, there exist exactly n edges with head ν and their tails are $\nu_1, \ldots, \nu_n$.”

In short, $\mu(\nu)$ denotes a single level ISG for a data object ν (see Figure 6).

[Figure 6] Nodes in an Information Structure Graph



Root(G) is the root of an information structure graph G; leaf(G) is a set of leaves of the graph G. Thus, the root of an information structure graph G(ν) is the data object ν. I.e., ν = root(G(ν)).

In the definition of an ISG, we assumed that there exists only one data creation type for a data object. However, there may exist more than one data creation types for a data object.

For example, a quantity on hand can be measured by counting number of items in a warehouse. Or, it can be calculated by adding (subtracting) an inbound (outbound) delivery to the previous quantity on hand. In this case, either of the followings can be true:

μ(item.qoh) = ⊙ , or

μ(item.qoh) = (ⓤ, inbound_delivery.quantity, outbound_delivery.quantity)

This means that the data creation type is not a simple one but a composite one. Thus, we need to combine these two data creation types to formulate a composite data creation type of a quantity on hand. For this, let us introduce artificial data objects of qoh_measured and qoh_calculated. Then, as shown in the followings, we can decompose item.qoh into two distinct data objects based on data creation types:

μ(item.qoh_measured) = ⊙,

μ(item.qoh_calculated) = (ⓤ, inbound_delivery.quantity, outbound_delivery.quantity)

Then, μ(item.qoh) is defined as follow, which denotes that it can be either of qoh_measured or qoh_calculated depending on a situation:

μ(item.qoh) = (⊕, item.qoh_measured, item.qoh_calculated)

Or, μ(item.qoh) can be defined based on qoh_measured and inbound_delivery and outbound_delivery.

μ(item.qoh)=(ⓤ, item.qoh_measured, inbound_delivery.quantity, outbound_delivery.quantity)

The example shows that data objects are identified based on their data creation structures. And, there exist more than one way to represent the same data creation structures. Here, we introduce rules to test an equivalence of information structure graphs.

ISG Reduction Rules: *An information structure graph G = <V, E, μ> is reducible with an order of ⓤ, ⊗ > ⊕. That is, the following rules are satisfied;*

(1) If $\mu(v) = (\oplus, v_1, v_2)$ and $\mu(v_1) = (\oplus, v_3, v_4)$, then $\mu(v) = (\oplus, v_3, v_4, v_2)$.

(2) If $\mu(v) = (\otimes, v_1, v_2)$ and $\mu(v_1) = (\oplus, v_3, v_4)$, then $\mu(v) = (\otimes, v_3, v_4, v_2)$.

(3) If $\mu(v) = (\oplus, v_1, v_2)$ and $\mu(v_1) = (\otimes, v_3, v_4)$, then $\mu(v) = (\otimes, v_3, v_4, v_2)$.

(4) If $\mu(v) = (ⓤ, v_1, v_2)$ and $\mu(v_1) = (\oplus, v_3, v_4)$, then $\mu(v) = (ⓤ, v_3, v_4, v_2)$.

(5) If $\mu(v) = (\oplus, v_1, v_2)$ and $\mu(v_1) = (ⓤ, v_3, v_4)$, then $\mu(v) = (ⓤ, v_3, v_4, v_2)$.

(6) If $\mu(v) = (\otimes, v_1, v_2)$ and $\mu(v_1) = (\otimes, v_3, v_4)$, then $\mu(v) = (\otimes, v_3, v_4, v_2)$.

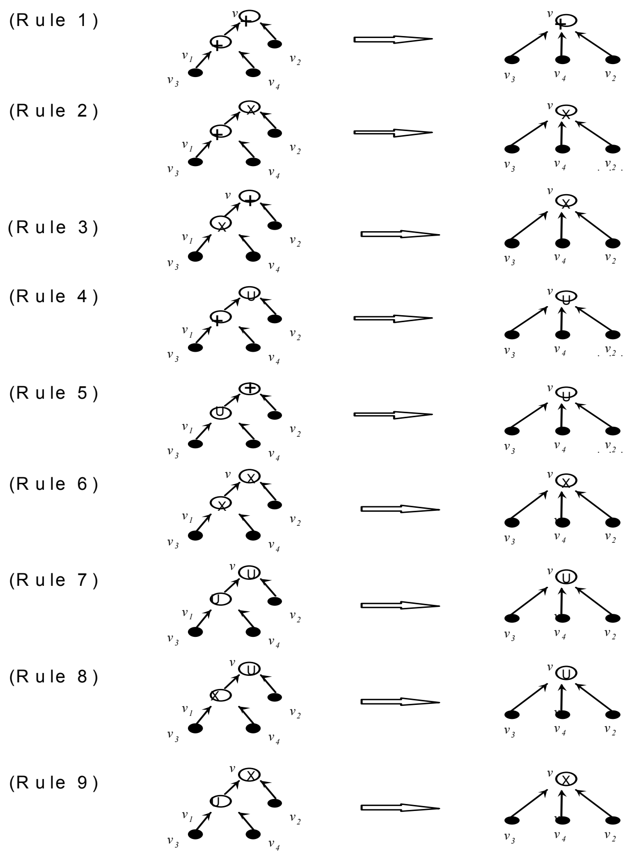
(7) If $\mu(v) = (ⓤ, v_1, v_2)$ and $\mu(v_1) = (ⓤ, v_3, v_4)$, then $\mu(v) = (ⓤ, v_3, v_4, v_2)$.

(8) If $\mu(v) = (ⓤ, v_1, v_2)$ and $\mu(v_1) = (\otimes, v_3, v_4)$, then $\mu(v) = (ⓤ, v_3, v_4, v_2)$.

(9) If $\mu(v) = (\otimes, v_1, v_2)$ and $\mu(v_1) = (ⓤ, v_3, v_4)$, then $\mu(v) = (\otimes, v_3, v_4, v_2)$.

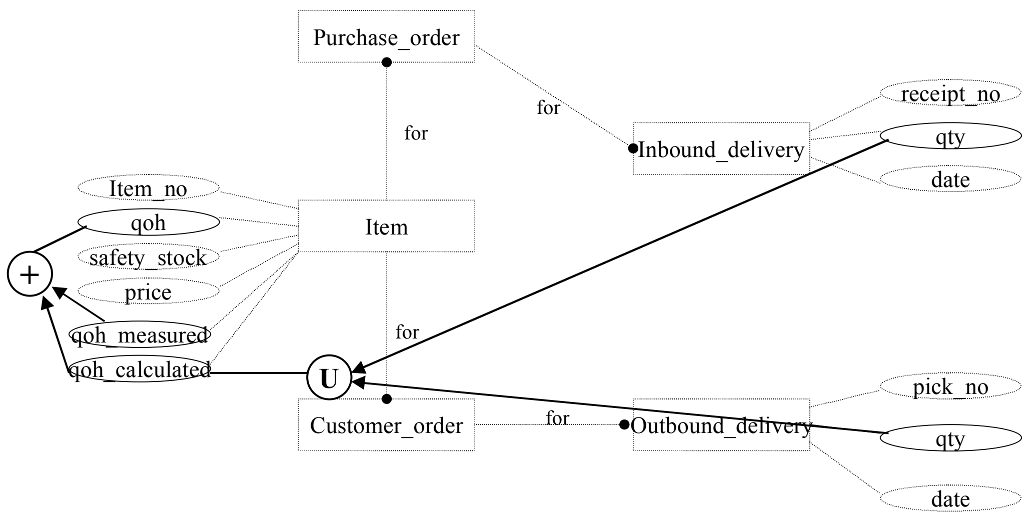
These rules are shown in Figure 7. In the figure, information structure graphs in the left hand side are said to be reducible to the ones in the right hand side.

[Figure 7] Information Structure Graph Reduction Rules

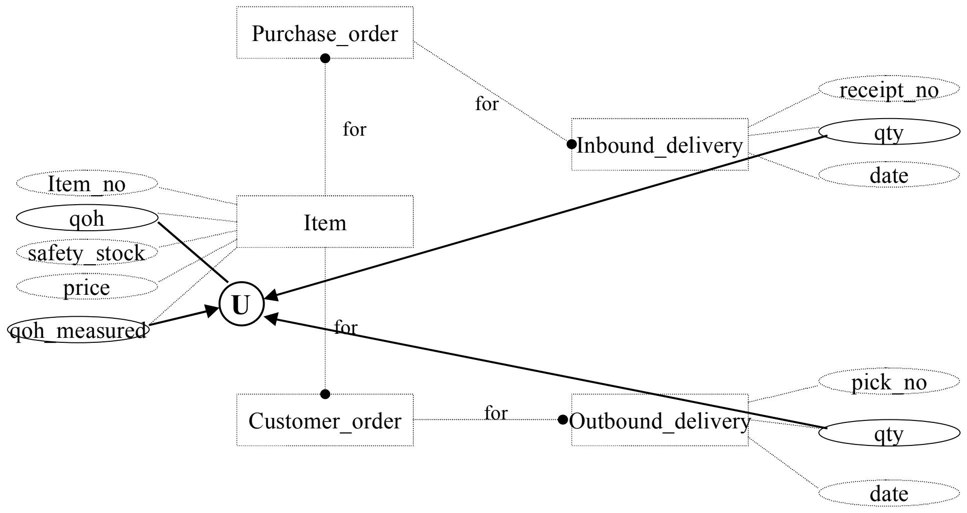


For example, Figure 8 is an ISG of item.qoh based on item.qoh_measured and item.qoh_calculated. Then, by applying the Rule 5, it can be reduced to the Figure 9. It equals to μ(item.qoh) that we defined in the above based on item.qoh_measured and inbound_delivery and outbound_delivery.

[Figure 8] An Information Structure Graph for a Quantity On Hand with measurement



[Figure 9] An Information Structure Graph for a Quantity On Hand with measurement - reduced



Information structure graphs grow as new nodes are inserted as input and/or output data objects. An information structure graph is said augmented if nodes and edges are added without altering its current structure.

Definition 2. *An information structure graph $G' = <V', E', \mu>$ is an augmentation of a graph $G = <V, E, \mu>$ if the following conditions are satisfied;*

(1) $V \subseteq V'$

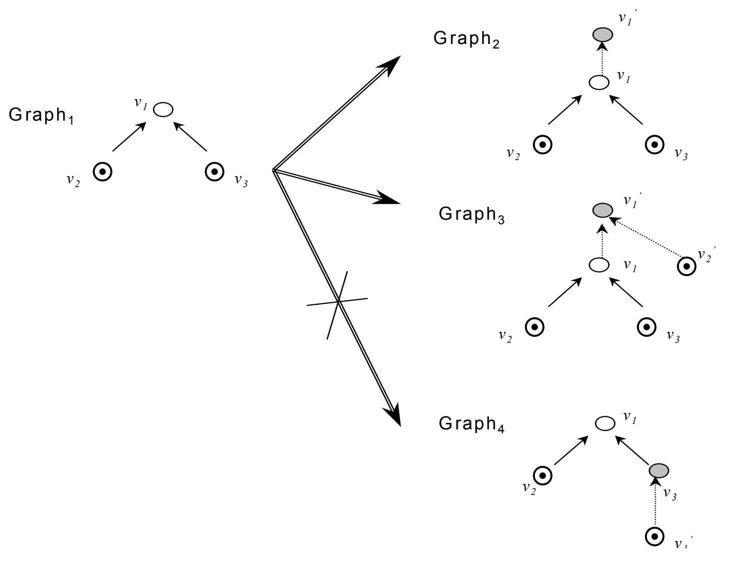
(2) $E \subseteq E'$

(3) If $(v_1, v_2) \in E' - E$, then $v_2 \in V' - V$.

I.e., all new edges are between new nodes, or from nodes in V to a new node.

An augmentation adds new nodes to a graph without altering its structure. For example, in Figure 10, graph$_2$ and graph$_3$ are augmentations of graph$_1$. Graph$_2$ adds a root to graph$_1$. Graph$_3$ adds a leaf as well as a root to graph$_1$. However, graph$_4$ is not an augmentation of graph$_1$ because the structure of graph$_1$ has been changed in graph$_4$.

[Figure 10] Augmentation of an Information Structure Graph



Augmentations are transitive. I.e., if G_2 is an augmentation of G_1 and G_3 is an augmentation of G_2, then G_3 is an augmentation of G_1. Among augmented graphs, some have identical leaves. They are called homogeneously augmented graphs.

Definition 3. *Information structure graphs* G_1 *is called a homogeneous augmentation of* G_2 *if* G_1 *is an augmentation of* G_2 *(or vice versa), and leaf(*G_1*) = leaf(*G_2*). A homogeneous augmentation set is a set of information structure graphs that are homogeneous to each other.*

Homogeneously augmented graphs share the same leaves. This means that data objects are produced from the same primitive data objects.

4. Applications of an ISG

An information structure graph abstracts data creation structures succinctly. Further, it is defined on a database schema. Thus, for each data object in a database schema, we can define its creation structures. These data creation structures are applied to plan and operate data management activities. In the followings, we describe typical applications of an ISG.

4.1 Planning and Control of Database Operations

An ISG shows data creation structures. Further, it is based on a database schema. Thus, it can be stored just like a database schema is stored in a data dictionary.

For example, in a relational database, ISGs can be described by the following tables, which denotes nodes and edges of ISG, respectively.

ISG_node (data_object, data_object_type[3], data_creation_type, …)

ISG_edge (output_data_object, input_data_object, time_lag, application_name, …)

Then, using ISG tables and SQL, we can query relevant information about data creation structures. For example, given a data item, we can retrieve all input data items that are used to create a quantity on hand:

```
SELECT          output_data_object, input_data_object
FROM            ISG_edge
CONNECT BY      PRIOR input_data_object = output_data_object
START WITH      input_data_object = 'item.qoh';
```

The query retrieves all input data objects for item.qoh. If an ISG for a quantity on hand is defined as shown in Figure 8, then the result is shown in Table 2.

<Table 2> A Result of an ISG query

Output Data Object	Input Data Object	Input of Input Data Object
item.qoh	item.qoh_measured	
item.qoh	item.qoh_calculated	item.inbound_delivery.qty
item.qoh	item.qoh_calculated	item.outbound_delivery.qty

Using the query, we can identify that primitive data objects of item.qoh are item.qoh_measured, inbound_delivery.qty and outbound_delivery.qty. And, thus, we know that they are to be managed correct to keep quantities on hand correct.

Further, we can identify data items that are derived from some data objects. Let's

[3] Data_object_type denotes whether a data object is an attribute or an entity.

assume that manual counting had been incorrect. Then, by querying output data objects that includes qoh_measured as an input data objects, we can filter out data items that may have been infected by the poor manual counting.

```
SELECT              input_data_object, output_data_object
FROM                ISG_node
CONNECT BY          PRIOR output_data_object = input_data_object
START WITH          output_data_object = 'item.qoh_measured';
```

In most relational database systems, a database schema is stored in data dictionary tables as shown in the following:

```
TBL (table_name, creator, table_type, …)
COL (table_name, col_no, col_name, col_type, width, nulls, …)
```

Then, we can combine data creation meta-data stored in ISG tables with these data dictionary tables.

```
SELECT      ISG_node.output_data_object, ISG_node.data_creation_type,
            COL.table_name, COL.col_name,
FROM        ISG_node, COL
WHERE       ISG_node.data_object = COL.col_name
    AND     ISG_node.data_object_type = 'attribute';
```

As shown in the above, queries on data creation structures may include data dictionary tables as well as ISG tables. Thus, for each attribute or table in a data dictionary, we can easily look up its creation structures.

An ISG is also a directed graph. Thus, using data management statistics such as error rates, we can convert an ISG to a distance network in which arcs are assigned with error rates. Then, we can search a critical path that shows highest error rates.

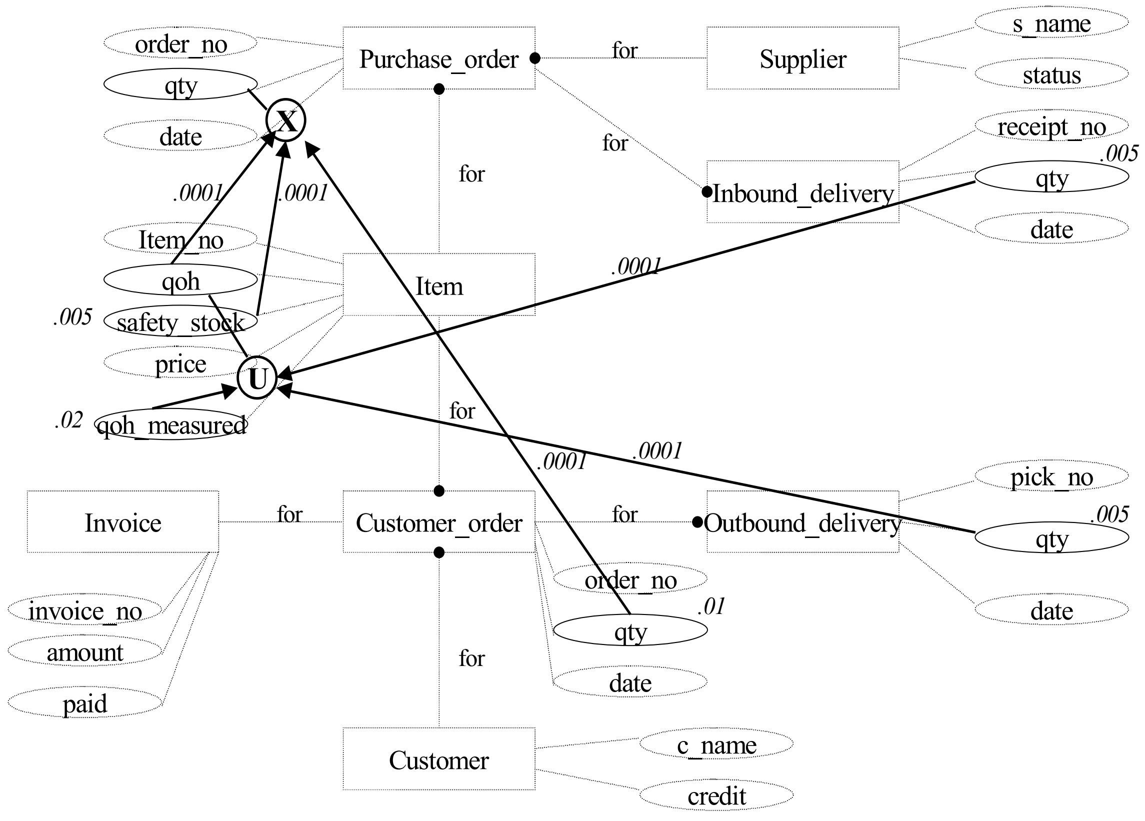
For example, a company has noticed that it has issued many unnecessary purchase orders. To correct this problem, we may go through the ISG of a purchase order quantity, which is shown in Figure 4. Let us assume that error rates are collected for each primitive data items and arcs as shown in Figure 11. Then, we can infer that incorrect manual counting has been the major source of unnecessary orders.

Further, if we can assign time lags to arcs in an ISG, we can analyze accumulated time lags between real world events and database values. This kind of analysis on an ISG may be applied to schedule database back-ups and recoveries.

Data creation types in an ISG correspond to triggering types in database operations. For example, item.qoh_calculated is a value update. Thus, by querying all value updates we can identify potential update triggers that might be useful for data management. We can also check inconsistencies in nested triggers by checking loops in an ISG. This kind of example shows that we can apply various network analysis and management tools to data management.

At last, but not the least, a pictorial presentation of data creation structures helps data administrators build a holistic view on data objects.

[Figure 11] An Information Structure Graph with Error Rates



4.2 Testing Semantic Homogeneity of Data Objects

Information structure graphs conceptualize data creation structures. By utilizing this feature of information structure graphs, we can test such semantic relationships among data objects as synonyms and homonyms. For example, a quantity on hand of Figure 8 is not identical to that of Figure 3. Thus, they are synonyms and have to be treated as distinct data items.

In addition to detecting synonyms and homonyms, information structure graphs can also be used to test miscellaneous semantic relationships among data objects. If an information structure graph is an augmentation of another, the one is semantically built upon the other. If an augmentation is homogeneous (i.e., an augmentation does not includes additional primitives), one is derived from the other. If not, one utilizes additional input data and thus cannot be derived from the other.

These kinds of semantic relationships are very helpful and may ease processes of database aggregations. With the advent of data warehouses and business intelligence databases, data extractions and transformations have become major issues in data management. Without ISGs, we may have to go through all details of database schemas and application specifications; however, with ISGs, we can easily identify inter-related data objects.

For example, we can group data items that share common primitive data objects. We can also partition data items into clusters by augmenting primitive data objects. As shown in Definition 2 and Definition 3, augmentations are transitive. Thus, we can build clusters of data objects using these definitions. These clusters help to work on relatively homogeneous data objects and ease data transformation tasks

5. Conclusions

This study has proposed to conceptualize data creation structures using information structure graphs. An information structure graph is a directed graph and is defined on a database schema.

Data creation structures can be utilized for management of data. Typical examples might be retrieving all input data objects to keep data values correct; retrieving all output data objects derived from a certain data object; calculating time-lags between real world events and database values; identifying a critical path to lower accumulated error rates.

An ISG can also be used to define and manipulate information semantics. By comparing information structure graphs, we can identify such semantic relationships as synonyms and homonyms.

Utilizing these characteristics of ISGs, we can identify data objects that are inter-related to each other in terms of data creation. These kinds of relationships may ease the processes of database aggregations and data transformation.

This paper is an initial attempt to abstract structures on data creation. However, its applications and potential usefulness might be extensive, especially in areas of data warehousing and database aggregation.

References

Abiteboul, S. and Hull, R. "IFO: A Formal Semantic Database Model," *ACM Transactions on Database Systems* (12:4), 1987, pp 525-565

Brackett, M.H., *The Data Warehouse Challenge: Taming Data Chaos*, John Wiley & Sons, Inc., New York, NY, 1996.

Chen, P. "The Entity Relationship Model – Toward a Unified View of Data," *ACM Transactions on Database Systems* (1:1), 1976, pp 9-36

Cyre, W.R., "Capture, Integration, and Analysis of Digital System Requirements with Conceptual Graphs," *IEEE Transactions on Knowledge and Data Engineering* (9:1), 1997, pp 8-23.

Gane, C., and Sarson, T., Structured Systems Analysis: Tools and Techniques, Prentice-Hall, Englewood Cliffs, NJ, 1979.

Hammer, M. and McLeod, D. "Database Description with SDM: A Semantic Database Model", *ACM Transactions on Database Systems* (6:3), 1981, pp 351-386

Liepens, G.E. and Uppuluri, V.R.R., *Data Quality Control: Theory and Pragmatics*, Marcel Dekker, 1990.

Redman, T.C., "Improve Data Quality for Competitive Advantage", *Sloan Management Review*, 1995 winter, pp 99-107.

Redman, T.C., *Data Quality for the Information Age*, Artech House, Norwood, MA, 1996.
Shipman D.W., "The Functional Data Model and the Data Language DAPLEX", *ACM Transactions on Database Systems* (6:1), 1981, 141-173

Sowa, J.F., *Conceptual Structures: Information Processing in Mind and Machine* , Addison-Wesley Publishing Company, Reading, MA, 1984.

Teory, T.J., Wang, D., & Fry, J.P. "A Logical Design Methodology for Relational Databases Using the Augmented Entity-Relationship Model." *Computing Survey* (18:2), 1986, pp 197-221.

Wang, R.Y., Storey, V.C. and Firth, C.P., "A Framework for Analysis of Data Quality Research," *IEEE Transactions on Knowledge and Data Engineering* (7:4), 1995, pp 623-639.
Wang, R.Y., "A Product Perspective on Total Data Quality Management," *Communications of ACM* (41:2), 1998, pp 58-65.

Warnier, J.D. *Logical Construction of Systems*, Van Nostrand Reinhold, 1981.
Yourdon, E.N. *Modern Structured Analysis*, Prentice-Hall, Englewood Cliffs, NJ, 1990.