REA Accounting Database Evolution[14]

Jia-Lin Chen, Dennis McLeod, & Daniel O'Leary

Synopsis

There are a number of evolutionary forces in firms that lead to the need to evolve accounting database systems. For example, systems are being reengineered resulting in substantial reductions in the number of agents and redefinitions of events. As firms downsize changing product lines, resources change, requiring further evolution of accounting databases. As a result, it is necessary to evolve accounting databases that are structured based on resources, events and agents (REA), as is the case with all such enterprise models. This paper discusses these motivations for evolving accounting database systems.

This paper presents one approach to aid in the evolution of accounting databases. A number of "evolution" events are elicited as part of that evolution process. The actual tasks necessary to evolve those databases in through those events are summarized in SEAtool, a prototype system designed to assist in the evolution of REA accounting database systems.

Keywords:REA Accounting Databases; Entity-Relationship Databases; Evolution of Accounting Databases

Introduction

The change in firms and the way those firms operate impacts their accounting databases. However, firms cannot develop an entirely new database with every change in the firm. As a result, databases must evolve to meet new needs. The need for database evolution is particularly critical with enterprise databases, like REA (Resources, Events and Agents -- e.g., McCarthy 1979b, 1982), where the database is designed to support an entire range of activities in the enterprise through the use of different views. The importance of this need for evolution is further exemplified in a recent analysis of research opportunities in database systems, where Silberschatz et al. (1991) identified evolving databases as an issue for future research.

This paper discusses SEAtool (Schema Evolution and Administration tool), a research prototype database tool. SEAtool is used to assist in the evolution of evolution of REA accounting databases. In particular, the evolution operators were developed in order to meet the needs of evolving an REA database. The paper illustrates how SEAtool can help REA databases respond to necessary evolutionary changes.

Objectives this Paper

The objectives of this paper are three-fold. The first objective is to establish the motivations for the need of REA database systems to evolve. Those motivations derive from a number of sources including mergers and acquisitions, changes in technology, changes in information and decision making requirements, reengineering, changes in product lines, and others. The second objective is to provide evolution "events" or "operations" for REA accounting database systems. A set of evolutionary operations is elicited to meet the needs of those evolutionary forces for change. The third objective is to discuss the architecture of a prototype system that provides one approach to the evolution of REA databases. SEAtool has three modules that are used to structure the evolutionary process, interfacing with an object-oriented database management system.

Outline of this Paper

[14] An earlier version of this paper was presented at the American Accounting Association National Meeting, New York, 1994. The author would like to acknowledge the comments of the participants.

This paper proceeds as follows. Section 2 provides background on REA accounting systems and evolutionary databases. Section 3 discusses some of the motivations for the need for accounting systems to evolve over time. Section 4 investigates some of the operations necessary for evolution of an REA database. Section 5 summarizes the architecture of SEAtool designed to accomplish evolution of REA systems. Finally, section 6 provides a brief summary, discusses some contributions and reviews some extensions of the paper.

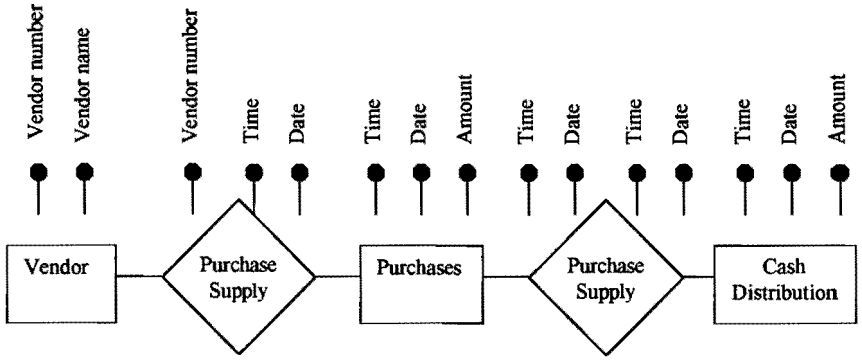
Background: REA Models and Object-Oriented Database Models

Perhaps one of the most extensive continuing lines of research in accounting information systems has been that of REA database model (McCarthy 1982). The REA model is based on accounting theory, including Sorter's (1969) work on events accounting, and on database theory, including Chen's (1976) work on entity-relationship databases. That accounting theory suggests that all of the relevant data about events be captured in a database and left to the user to classify and sort as they saw fit. A critical part of Chen's entity-relationship model is its emphasis on the semantic expressiveness. Previous database models were limited in their ability to specify complex databases (Hammer and McLeod, 1981), such as events databases. As a result, employing an entity-relationship approach, REA can be used to capture and store complex events in a robust model.

REA Models

The REA framework is based on the development of a system to capture the interaction of economic resources, economic events and economic agents (McCarthy, 1982). McCarthy used a range of accounting theory literature to generate the REA framework. The initial and subsequent research on the development and application of the REA model is provided in a sequence of papers (McCarthy, 1979a, 1979b, 1981, 1982) and other research, including Gal and McCarthy (1986) and Denna and McCarthy (1987).

Resources, events and agents are captured using Chen's notion of an entity. The connections between those resources, events and agents are then captured using Chen's notion of a relationship. An example REA schema is summarized in figure 1.



(Source: Gal and McCarthy, 1986)

Figure 1

Prototype Implementations of REA Databases

There have been some prototype models built to study REA databases. McCarthy and Gal (1981) developed an events accounting implementation using the CODASYL approach to database management. Gal and McCarthy (1986) developed an implementation of a relational accounting system using an REA approach. Denna and McCarthy (1987) developed an implementation of an events accounting system specifically designed for a decision support environment. However, that research has not focused on prototypes for the evolution of REA systems. Most recently, McCarthy and Rockwell (1989) and Rockwell (1993) developed a prototype system to assist in the design of accounting information systems.

Evolution of Database Models

Evolution of databases has been recognized as an important issue for a long time. For example, Everest (1974) listed four major objectives to database management. One of those four objectives was the ability of the database management system to change and grow to meet the needs of its users.

There are at least three components associated with the evolution of accounting database systems. First, there is the motivation and theory: "why do we expect accounting database systems to evolve?" This question is the subject of section 3 of the paper. Second, there is the process: "How can we evolve

accounting database systems?" This question is the focus of section 4 of the paper, where a number of different evolution tasks are presented for REA databases. The third question is "Can we build a system to evolve accounting database systems?" To answer this question SEAtool was built. The architecture of SEAtool is discussed in section 5.

There has been limited research in the area of evolving database systems until recently. ORION (Banerjee et al., 1987), ENCORE (Skarra and Zdonik, 1986) and GemStone (Penny and Stein, 1987) are systems that employ an object-oriented model for which research database evolution mechanisms have been explored. ORION and ENCORE employ a screening approach and Gemstone uses a conversion approach to ensure that the database extension is consistent with the evolved schema. A summary of top-down and bottom-up incremental design of entity-relationship systems has also been addressed. Batini et al. (1992) summarizes research from Batini and Santucci (1980) and Ceri et al. (1981). However, there has been virtually no analysis of the evolution of accounting databases.

Task or Request Level of Evolution

Given that evolution is important and possible there is another question: "At what level should the user be able to evolve the accounting database?" Previous database evolution researchers have forced the user to make each step of the evolution process. The system discussed in this paper facilitates the evolution process by aggregating a number of minor evolutionary steps into a "task." The user needs only choose the appropriate task and the system performs all the steps for the user. Section 4 defines the set of tasks used by the system for REA database evolution.

Issues in Evolving Accounting Models: Motivations for Accounting Database Evolution

REA models need to evolve to meet changes required of the accounting database. We assume that different firms have different databases and that there is no universal implementation of the same database. The purpose of this section is to begin to establish a theory of the evolution of accounting databases.

There are a number of reasons for the search for a theory or theories of evolution of accounting databases. First, if there is a theory then that theory might prove useful in forecasting the amount and type of evolution that would be required of accounting databases. Second, given an estimate of the amount and the type of evolution would then prove useful in generating tools to assist in the evolution process.

The existence of a theory of evolution could also exploit domain specificity (e.g., Chen et al., 1995). In particular, changes that affect an accounting database might have no effect other domains.

Theories of Evolution of Accounting Databases

There is no established theory of the evolution of accounting systems or databases. Generally systems researchers recognize the need for evolution but do not specify its sources or approaches to evolution (e.g., Everest, 1974). There has been work in the economic evolution of industries (e.g., Porter, 1980), historic evolution of technology (Basalla, 1990), and the evolution of computational artifacts (e.g., Simon, 1985).

Porter (1980) provided "a framework for forecasting evolution." Porter identified a number of evolutionary processes including changes in reduction of uncertainty, expansion (or contraction) in scale, changes in costs, product innovation, marketing innovation, government policy changes, and other changes. However, many of the changes addressed by Porter would not necessarily result in database evolution (e.g., changes in specific costs). Thus, the theory probably is not directly applicable to accounting databases. However, Porter's theory does suggest that changes in the firm and its environment are necessary concerns for evolution.

Basella (1990) provided a history of the evolution of technology, with a number of detailed case studies. Basella (1990, pp. 6 and 7) noted both "necessity is the mother of invention" and "invention gives birth to necessity." Thus, need and availability lead to evolution of technology. Basella (1990) argued that technology evolves gradually, with some periods of more rapid evolution. In any case, the arguments of Basella would be consistent with the evolution of a database over time, rather than just constant replacement of the accounting system and corresponding database. In addition, this research suggests the importance of evolution of systems as a variable influencing accounting databases.

Simon (1985) had a number of observations about evolution of artifacts and complex systems, including the following. First, he argued for the development of modular systems. The existence of modularity facilitates evolution. REA is modular, thus facilitating evolution. Second, systems are man-made artifacts and thus, those artifacts can be changed to meet different needs. This suggests the development of a system to facilitate the direct evolution of the database.

Towards a Theory of Evolution of Accounting Databases

Although these theories provide some insights there is no one theory that seems directly applicable to the evolution of accounting database systems. However, these theories suggest three basic sources of direct evolutionary change in accounting database systems, including changes in firm (changes in the firm through mergers and acquisitions; changes in the firm's location on the life cycle; changes in accounting controls as systems are reengineered and changes in the firm product line); environment (changes in technology; and changes in client requests for information,); and decision making (changes in decision making needs and allocation of decision making activities).

Firm Changes: Mergers and Acquisitions

Perhaps the most compelling arguments for the need for evolution in accounting database systems are the cases of mergers and acquisitions. Once a firm has been acquired there are at least four approaches used to integrate the accounting information of the two firms.

First, the acquiring firm can issue requirements of specific accounting report information or desire information about additional accounting events. In some cases that reporting may require new information be captured by various entities or relationships within the accounting database system. For example, more detailed information regarding resources, events, or agents might be required. Alternatively, if the firm's database already prepares that information, then no changes would be required.

Second, the acquired company may be required to use specific documents or processes in their accounting systems with additional information beyond that of previous requirements and documents. In that situation, the information captured about resources, events or agents would need to be extended or changed to capture that additional information. This could result in changes to the entities or relationships within the accounting database.

Third, the acquiring company may require that a specific accounting information system would be used. This likely would require a completely new database system for the acquired company.

Finally, the nature of the acquired firm may change with acquisition. For example, rather than dealing with a broad range of firms, the acquired firm may be treated as an intermediary between vendors and the acquiring company or internal consultant, thus changing the relationships that must be accounted for in the database. This would change the relationships between different resources, events and agents, and could redefine the event set.

Firm Change: Growth or Location of Firm in Life Cycle

The growth of a firm can ultimately result in changes in accounting databases. Changes in location in the life cycle or growth of the firm can result in changes to resources, events or agents. For example, very

small firms may require cash with each sale. Thus, cash receipts mirror sales and the events are redundant, cash is sales and sales are cash. As firms increase in size, they are likely to allow the event of cash receipt and sale to drift apart.

Firm Change: Diversification

Similarly, as firms increase in size, they are likely to diversify both geographically and in product line. Both can force the need to account for different locations and products. For example, if a new product line is brought out with different characteristics than the old product line then those characteristics often must be a part of the database. In this situation, there would be new resources, possibly new agents monitoring those resources and potentially a different set of events.

Environmental Change: Technology

An important source of changes in information captured for accounting databases occurs with changes in technology. For example, as multimedia storage becomes possible, new media will likely become a part of the firm's accounting database system. This would allow capture of specific types of information that could be used for various purposes (e.g., imaging). In addition, changes in technology facilitate changes in accounting system processes and events. The recent interest in "reengineering" (e.g., Hammer, 1991) has illustrated the major changes that occur in accounting processes with changes in technology.

Changes in technology, etc. can result in changes in accounting controls. In some cases that control information can be captured as part of accounting databases. For example, as the use of imaging increases, data such as "signatures" can be captured by databases and used for control purposes. Images of signatures can be compared for similarity. In those cases where signatures are not appropriately similar that can indicate a potential control problem. In order to generate such controls, the new information ("signature") must be captured and integrated into, e.g., event or agent databases. Such control information might be treated as changes to entities, depending on the particular situation.

Environmental Change: Client Requests for Information

Another source of change in the accounting database occurs when clients or suppliers have changes in their information needs. Those changes in needs could arise from any of the other sources for change discussed in this section. Such changes are likely to result as changes in entities (e.g., purchase orders) or the generation of new entities. Changes in information could be desired about resources, events or agents.

Environmental Change: Information Requirements

If there are changes in the requirements for decision making information or requirements in the legal requirements for accounting disclosure then that also can require changes to the accounting database. For example, if the Securities and Exchange Commission (SEC) or the Financial Accounting Standards Board (FASB) requires changes in accounting disclosures then that indicates that new information must be a part of the accounting database. Those changes may reflect additional information on resources, events or agents. In addition, event sets for tax and financial accounting are not the same and both change over time. There may also be changes in the event set.

Decision Making

The initiation of a decision process is often structured with different information then if the process were understood. In some cases information is ignored that is needed, while in other situations, information is included that is not necessary. Once a decision process is better understood, useful information can be better identified. In such situations, that would suggest that databases inevitably include some of the "wrong" information and some of the "right" information. As the decision becomes better understood, the database could be evolved to meet the information needs of the users.

Allocation of Decision Making Activities

As organizations evolve, they also evolve the allocation of decision making activities. For example, some reengineering activity aims at pushing down the decision making to the lowest possible level, that is changing the allocation of decision making activities. This indicates that the agents involved in the process change (e.g., movement of capture of receiving information to the receiving dock). It also can indicate a change in the information that is to be collected (e.g., scanning can lead to different information being collected).

Other Issues

There are likely to be other issues that cause the accounting database system to evolve. In particular, any activity of the firm that would lead to changes in the portfolio of accounting activities could provide a need for the system to evolve. Thus, it is not the need to evolve particular accounting functions, although that could require evolution, but instead the portfolio of accounting functions that leads to the evolution of an accounting system.

In addition, there could be implementation compromises made in the development of accounting systems. Such compromises would typically be made for cost-benefit reasons. Accordingly, evolution would be desirable if that cost-benefit relationship changed over time.

The important issue is that the systems will need to evolve to meet a variety of requirements placed on the system. In addition, those changes can influence any of resources, events or agents.

Alternatives

There are effectively three alternatives to the changes discussed in this section. First, a completely new data model might be developed with each such change. In this case, most large firms would be constantly generating new data models. Existing database systems would become obsolete before completion. Second, no changes might be made. Instead, patchwork systems of paper and computer media would be the rule of the day. Decision making needs would go unanswered. Third, the database could evolve. That is the approach proposed in this paper. In addition, this paper provides a discussion of tool to facilitate the evolution process.

Operations on REA Databases

The above discussion indicates that the evolution of databases must accommodate a variety of changes in the firm that impact the accounting database. Throughout, these changes are designed to reflect evolutionary changes in the life of the firm.

Changing the database can occur at the a number of levels, including the schema. At that level there are least two approaches that can be used. The evolutionary changes could be at the individual activity level or at the task level, where a number of individual activities are aggregated to form one basic evolutionary change in the systems. For example, one of the operations discussed below is that task of replacing an entity. That task consists of a number of activities, such as renaming the entity, renaming the attributes, etc.

Providing a system that operates at the task level makes the process of evolution easier on the user, more timely and can improve the quality of the evolution. If the user is responsible for individual activities, instead of tasks, then it is always possible that activities will be ignored or done out of order, influencing the quality of the evolution. By embedding evolution into the task sets, these problems are circumvented.

The prototype system, SEAtool, includes a variety of evolutionary tasks. Those tasks include: adding/changing/ deleting attributes; replacing entities and relationships; adding new entities and relationships; deleting existing entities and relationships; merging entities and relationships; tearing (or expanding) entities and relationships; generalization; uncorrelated entities; and generating parallel relationships. The operations derive from a search of the evolutionary database literature (e.g., Batini et al., 1992) and the evolution changes in firms described above. This set of operations is discussed and illustrated using a variety of examples.

Adding, Deleting and Replacing Entities and Relationship Attributes

Changing entity and relationship attributes refers to the addition, deletion, or replacement of database attributes in the respective resources, events or agent databases, associated with each entity. Similarly, relationships may also need those same operations. For example, additional attributes might be required for an entity, in order to capture the entire range of new product information. These changes could appear as in figure 2. Throughout the remainder of this section the figures are divided into two sections (a) and (b). The (a) figure indicates the "before" and the (b) figure indicates the "evolved" schema.

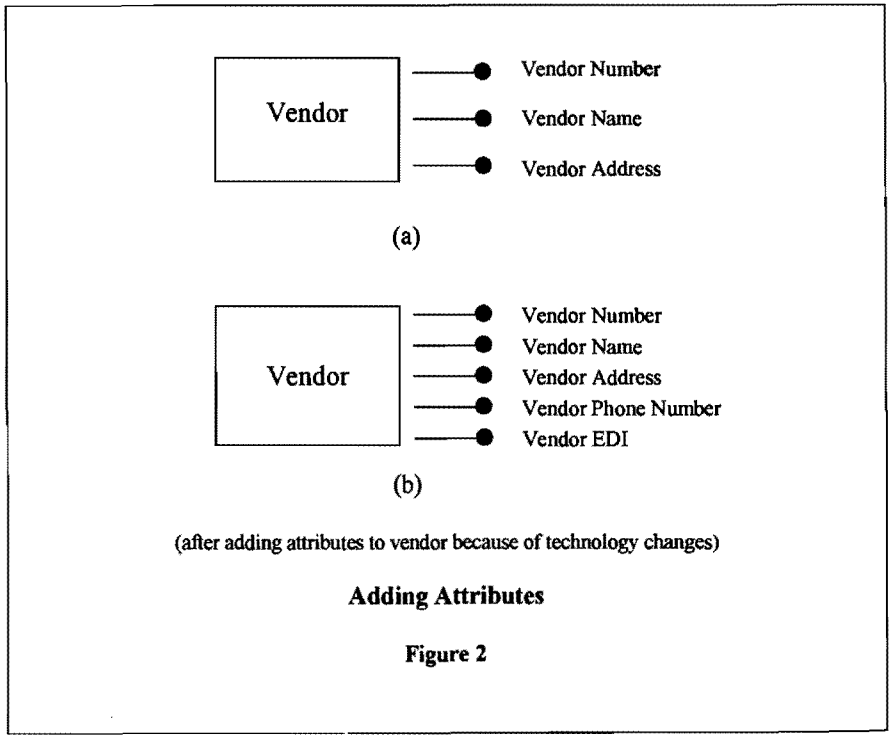


Figure 2

Replacing Entities and Relationships

Changing entities and relationships refers to the replacement of an entity or relationship with another entity or relationship. That replacement process might involve renaming a given entity or relationship or replacing an entity (relationship) with an entity-relationship-entity group (relationship-entity-relationship) group. Thus, a resource, event or agent is replaced, or a relationship is changed. For example, an acquired firm that used to deal with external vendors exclusively may be cast in the role of dealing only with the acquiring company. For example, the acquired firm may go from performing consulting for a variety of clients to the point of only performing internal consulting. These changes would appear as in figure 3.

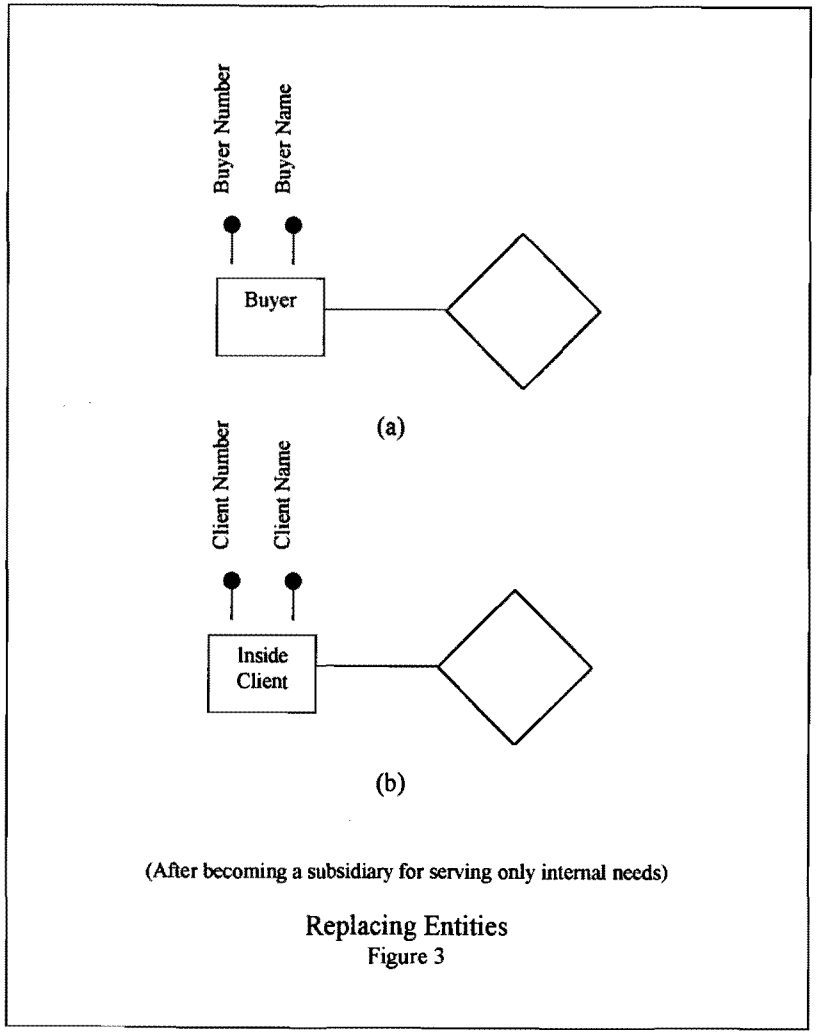
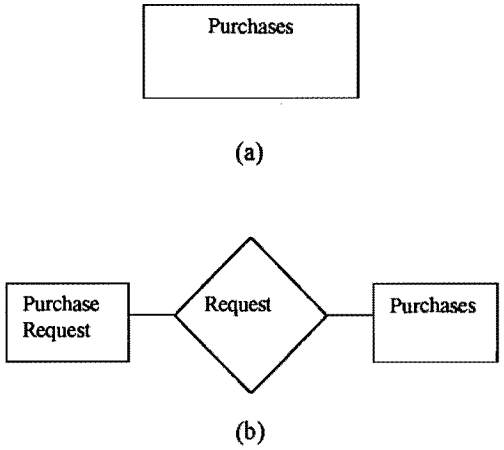


Figure 3

Adding Entities and Relationships

Adding entities and relationships can be accomplished generally by adding a new entity and relationship pair. This approach would recognize an entirely new set of information that needs to be accounted for. Typically this would indicate that the firm is recognizing a more detailed base of resources, events or agents, and the corresponding relationships. For example, in the situation where a request for purchase is added to a system with purchase orders as the only formal part of the purchasing system, we would see the change in structure exhibited in figure 4. Additional data would be required to meet the needs of a request for a purchase. Note this is independent of the specific artifacts but depends on the resources, events and agents. For example, the request for a purchase forces accounting for a broader base of agents and places another event in the database.

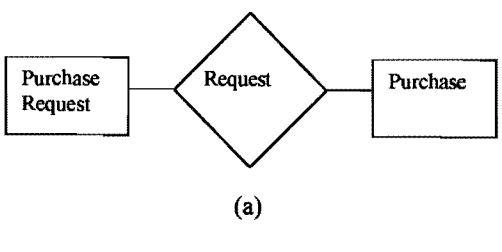


(Adding purchase requests to system to improve accounting controls)

Adding Entities and Relationships
Figure 4

Deleting Entities and Relationships

Similarly, pairs of entities and relationships might be deleted as part of a reengineering effort. This would suggest that resource, events or agents are eliminated, a major issue in the downsizing of firms. Deleting indicates, e.g., that an event is not going to be accounted for in the database. For example (Hammer, 1991), in some firms, individuals do not make purchase order requests. Instead, purchase orders are generated directly. The event of requesting a purchase order is eliminated. If such a reengineering effort is made then that can result in the deletion of a pair of entities and relationships. Figure 5 contains an example.



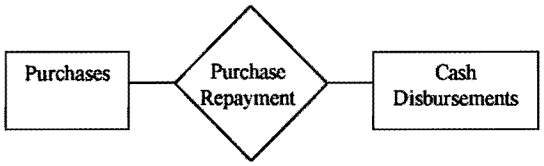
(Reengineering the purchase function)

Deleting Entities and Relationships
Figure 5

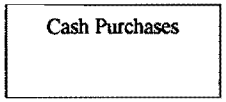
Merging Entities and Relationships

The merger of entities and relationships generally results in the collapse of the schema from two entities and one relationship into one entity, or two relationships and one entity into one relationship. This approach would be an alternative explanation to the deletion of entities and relationships for reengineering efforts. This would result in the collapsing of resources, events or agents and the corresponding relationships.

A merger is different than a deleting operation. A merger does not result in the loss of data associated with, e.g., either entity, since the information would redundant. An example would be to adapt an REA system to a small firm that only takes cash for sales. In such small firms the sale and the cash receipt are redundant and simultaneous. The redundancy between the information in the sale and cash receipt would lead to requiring only one of the entities. Figure 6 contains an example.



(a)



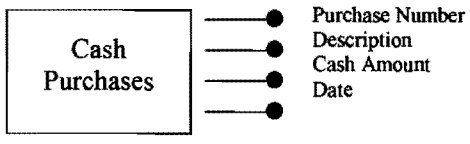
(b)

(In small firms where only cash purchases are made, only one entity is required)

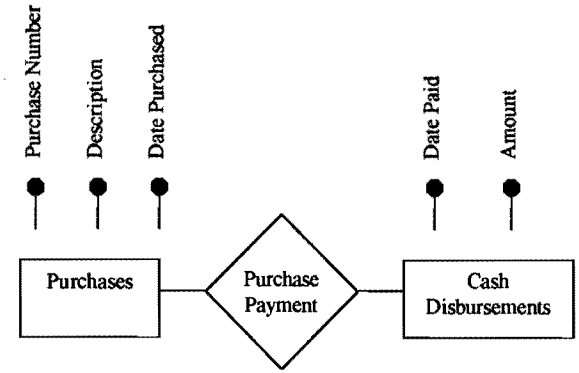
Merging Entities and Relationships
Figure 6

Tearing Entities and Relationships

The opposite to merging entities and relationships is to tear existing entities and relationships to generate additional entities and relationships. Tearing separates the specific entity or relationship into two entities or relationships. Thus, resources, events or agents would be split into two resources, events or agents. Generally, if either an entity or relationship is torn, then an additional relationship or entity, respectively can also be added. Such an approach could be appropriate where accounting activity is divided between accounting and some other functions. Tearing is different than adding in that previously redundant information is not redundant anymore and must be accounted for separately. This approach is illustrated in figure 7 with the same example as in figure 6, only reversed. A small firm that only makes cash sales evolves to a firm that must account for cash receipts and sales separately.



(a)

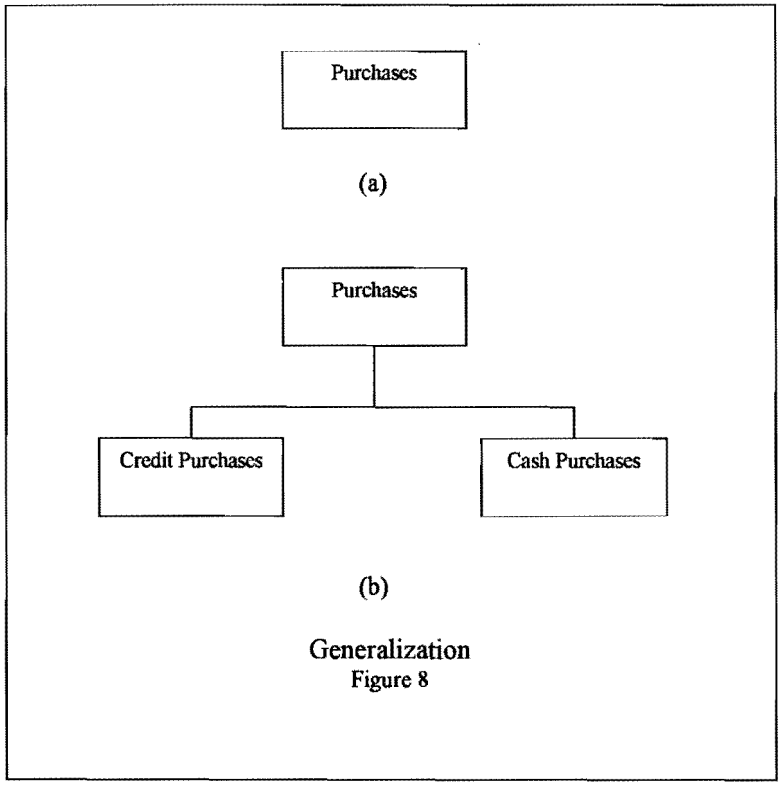


(As the firm expands, they make purchases on credit)

Tearing Entities
Figure 7

Generalization

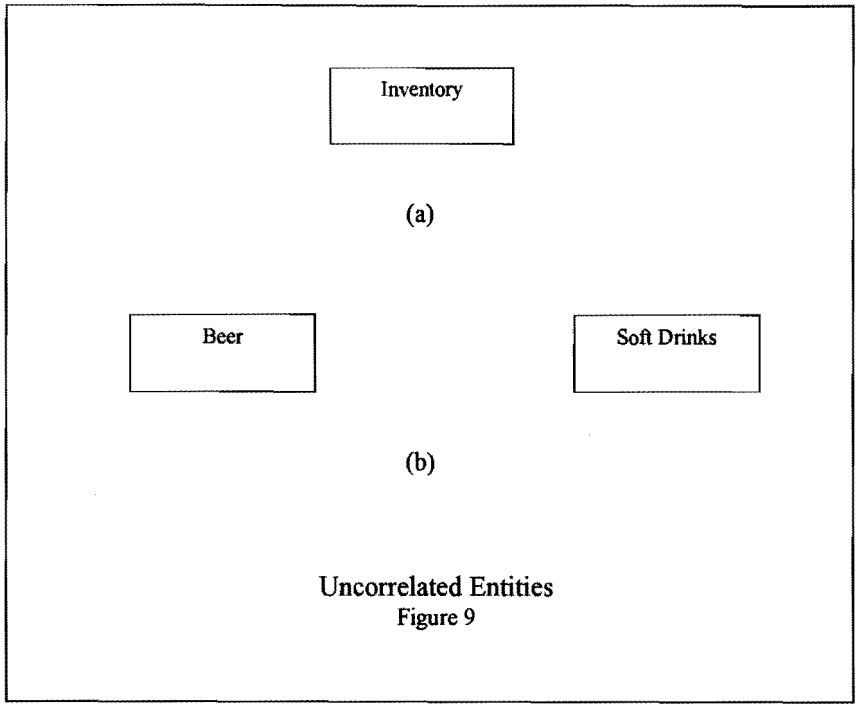
Entities, such as resources, events and agents can be generalized. For example, the specific types of inventory can be generalized into the notion of inventory. This is summarized in figure 8.



Generalization
Figure 8

Uncorrelated Entities

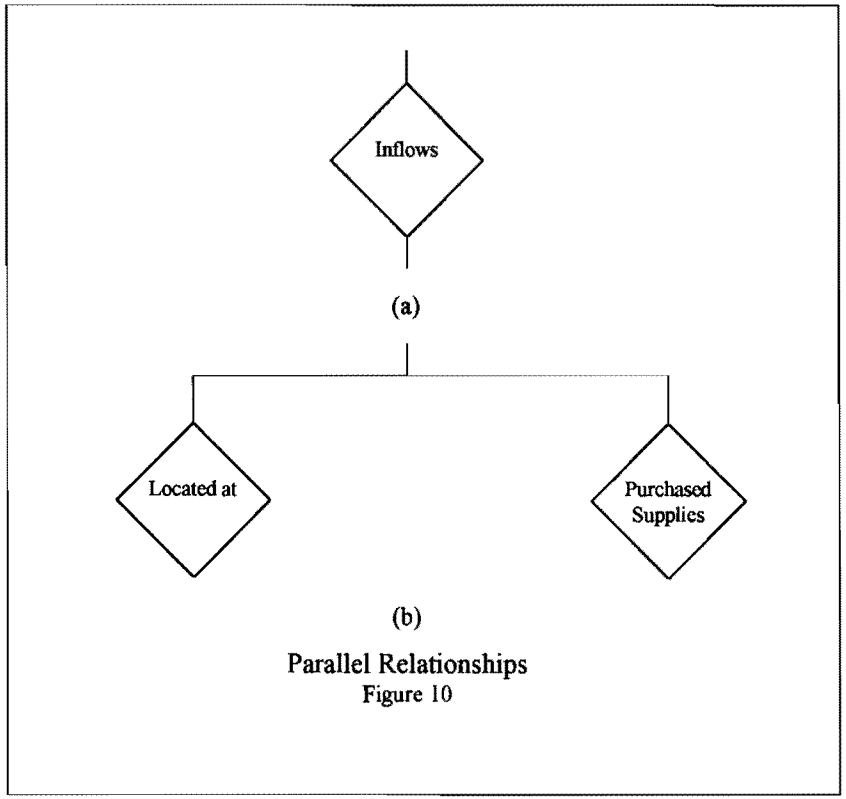
Uncorrelated entities also can be captured. For example, resources such as different lines of inventory can be represented as uncorrelated entities. Figure 9 gives an example. Other examples could focus on agents or events.



Uncorrelated Entities
Figure 9

Parallel Relationships

Parallel relationships occur with the addition of new relationships between entities. For example, new systems such as "just-in-time" inventory systems have changed the concept of inventory. In such systems, inventory may be purchased, but stored at the supplier's location until needed. This is illustrated in figure 10.



Parallel Relationships
Figure 10

Relationship between Evolutionary Forces and Operations

The evolutionary forces discussed in the previous section are captured in the tasks of this section. A number of examples have been provided that illustrate that the issues evoked in the previous section can be evolved using the tasks generated in this section.

SEAtool- A Tool for Evolution

This section discusses the architecture and design of SEAtool in the REA environment. The prototype implements a proposed schema evolution methodology and assists a designer/administrator in the management and evolution of the schema. SEAtool allows the user the ability to choose evolutionary "tasks." Those tasks are then translated into a portfolio of requests and ultimately procedures that are used by the system to evolve the database. The aggregation at the task level makes the evolution of REA databases easier since users only need to know the evolution task that is required, not the specific requests or procedures necessary to complete that task. The three layer framework used in SEAtool is illustrated in Figure 11.

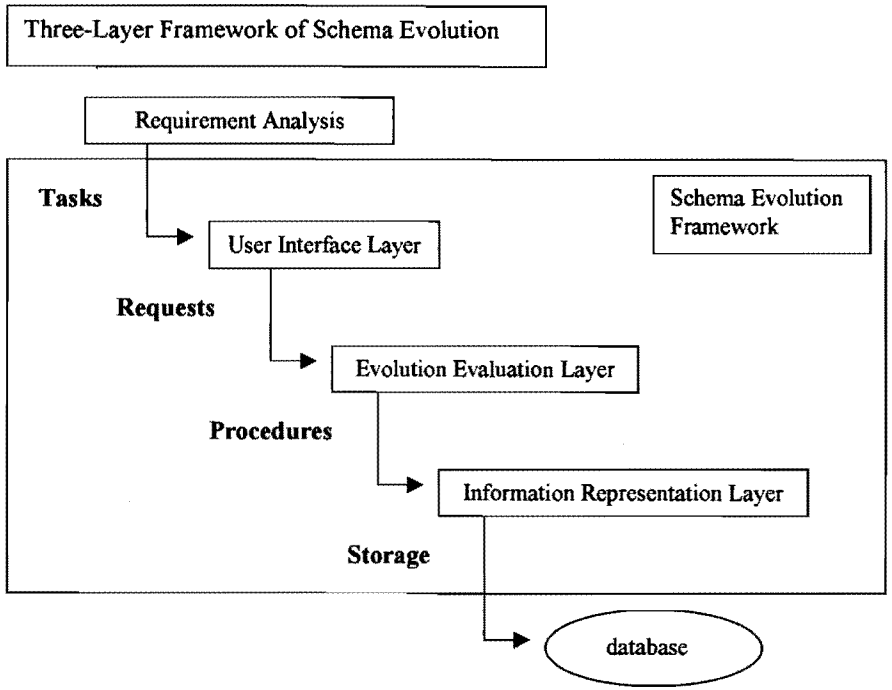


Figure 11

SEAtool's Architecture

The three layer framework results in the architecture of three modules in SEAtool: SEAshell, SEAengine and SEAbase. In addition, SEAtool interfaces with the commercial package, Versant[15] Object Oriented Database Management System (OODBMS). As a prototype, SEAtool builds its own storage operations on Versant OODBMS. SEAtool's structural architecture is illustrated in figure 12.

[15] Versant is the trademark of Versant Object Technology Co.

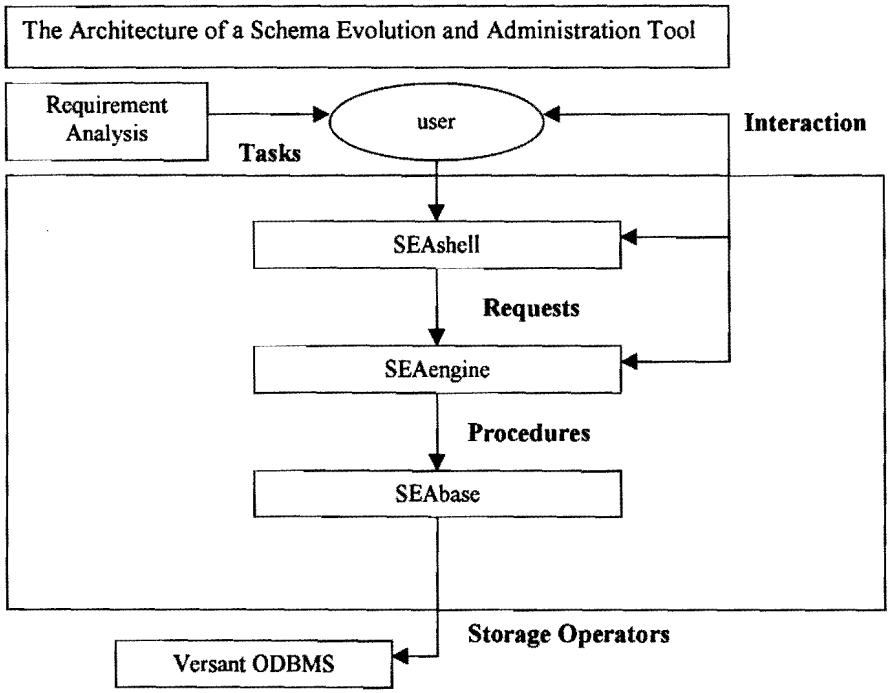


Figure 12

SEAshell performs the user-interface function. A user of SEAtool recognizes the evolution task and issues the schema evolution task in the context of SEAshell. SEAshell accepts the task, interacts with the user to complete the job required by the task. In particular, SEAshell:

- .Provides the user interface to the user in order to determine the users evolution tasks.
- .Interacts with SEAengine through evolution requests to provide the tasks that the user has specified.
- SEAengine analyzes the requests issued by SEAshell and executes the schema evolution procedures to complete the requests. In particular, SEAengine
  - .Checks the status of the database and the context of the schema.
  - .Interacts with SEAShell to determine what requests are necessary to complete the user specified tasks.
  - .Handles the propagation effects of the evolution requests.
  - .Interacts with SEAbase by issuing the appropriate procedures to update the data and meta data in SEAbase.

The SEAbase module accepts the procedural call issued by SEAengine and completes the procedures using Versant functions. SEAbase is built to support an object-based model. SEAbase defines two system classes, called SEAobject and SEAclass. Those two classes are initialized, while a Versant database is created. All of the SEAbase instances are stored as instances of a system class, SEAobject. All of the SEAbase classes are stored as instances of SEAclass.

Construction of SEAtool

SEAtool, was built in a NeXTstep[16] environment to experiment with the evolution of REA databases. SEAshell and SEAengine are written in Objective C. SEAbase is defined in C++. The interface of SEAbase matches the procedure level of the schema evolution methodology. System classes such as SEAobject and SEAclass are defined by Versant C functions and encapsulated in SEAbase classes.

Evolving the Schema: Task Panels

SEAtool guides the user through the schema evolution process using "Task Guidance Panels" (TGP). For each generic task there is a TGP used to evolve the schema. A Task Guidance Panel (TGP) is a windows-based system with two components: Task Display (TD) and Task Working Space (TWS). The TD is an area used to show the schema's entities and relationships under consideration. The TWS is used to collect a user's specifications for the current task and to guide the user to complete the task. The TWS consists of an Evolution Task Catalog and a stack of Subtask Working Sheets for each task. The Evolution Task Catalog is a list of schema evolution tasks for given initial status and target pair.

The user brings the portion of the schema to be evolved into the TD within a TGP. SEAshell guides the user step-by-step through a stack of Subtask Working Sheets to complete the particular evolution task. Each Subtask Working Sheet has two parts: a Message Area and an Operation Area. The Message Area shows the system's suggestions and the user operates the evolution subtasks in the Operation Area, supplying information, as required.

Browser Interface

SEAtool also has a windows-based browser-oriented interface. There are three main browsers: the class hierarchy browser; the class network browser; and the class inspection browser.

The class hierarchy browser allows the user to browse the class hierarchy of a forest of classes. Each hierarchal tree has its own root class. A user can focus on one branch of a subclass each time. The class network browser allows the user the ability to browse the relationship of a class and traverse to others through the whole network of the classes of a database. A user can focus on one relationship of a class each time. The class inspection browser lets the user browse the attributes defined in a class and the inherited attributes from its superclass. A user can trace the inherited attributes up to its root class. For each attribute, a user can inspect the domain class and its inverse attribute.

Summary, Contributions and Extensions

This section briefly summarizes the paper, discusses the contributions and investigates some extensions.

Summary

This paper discussed the forces behind the evolution of accounting database systems. In addition, this paper analyzed the operations necessary to change REA accounting databases to meet the requirements of evolution. Finally, this paper briefly discussed the architecture of a prototype system designed to evolve REA systems, SEAtool.

Contributions

This paper has three basic contributions. First, there has been little, if any, analysis of the forces behind the need for accounting databases to evolve. This paper provides a discussion of some of those issues and a preliminary theoretical analysis.

[16] NeXTstep is a registered trademark of NeXT Computer Inc.

Second, although there have been some efforts to summarize general evolutionary operations for entity - relationship databases (e.g., Batini et al. 1992), there have been no such efforts to evolve accounting databases and tie that evolution to evolutionary forces of the firm. In addition, this paper used those forces to elicit additional operations of change and evolution.

Third, previous research on REA systems has generated at least three prototype systems. However, this paper provides the first prototype system designed to evolve REA databases.

Extensions

Although this paper investigated some of the forces behind the need for accounting databases to evolve, there is as yet no general theory. Future research could investigate a unifying theory or elicit alternative forces for the need to evolve accounting databases.

Much of the discussion here was couched in terms of relational database structures. Alternatively, an object-based approach could be used (e.g., Chen et al. 1994). This paper investigated a number of different operations that are part of the evolution process. However, there may be other approaches that can be used to evolve accounting databases. The system discussed in this paper is a prototype system designed to solve problems of evolution as a proof of concept. However, such systems typically can be improved from an efficiency and effectiveness perspective, e.g., to run faster. Such concerns are beyond the scope of this paper.

References

Basalla, G., The Evolution of Technology, Cambridge History of Science Series, Cambridge University Press, Cambridge, 1990.

Batini, C. and Santucci, G., "Top-Down Design in the Entity-Relationship Model," in P. Chen, editor, Entity-Relationship Approach to systems Analysis and Design, North-Holland, 1980.

Batini, C., Ceri, S., and Navathe, S., Conceptual Database Design: An Entity-Relationship Approach, Benjamin - Cummings, 1992.

Baneree, J., Kim, W, Kim, H., Korth, H., "Semantics and Implementation of Schema Evolution in Object-Oriented Databases," ACM SIGMOD Annual Conference on the Management of Data, San Francisco, CA, May 1987.

Ceri, S., Pelagatti, G., and Bracchi, G., "Structured Methodology for Designing Static and Dynamic Aspects of Database Applications," Information Systems, Volume 6, 1981, pp. 31-45.

Chen, P., "The Entity-Relationship Model -- Toward a Unified View of Data," ACM Transactions on Database Systems, March 1976, pp. 9-36.

Chen, N., McLeod, and O'Leary, D., "Schema Evolution for Object-based Accounting Systems," in Object-Oriented Methodologies and Systems, Springer-Verlag, 1994, pp. 40-52.

Chen, N., McLeod, and O'Leary, D., "Domain Guided Schema Evolution for Accounting Database Systems," Expert Systems with Applications, Volume 9, Number 4, pp. 491-501, 1995.

Denna, E. and McCarthy, W., "An Events Accounting Foundation for DSS Implementation," in Decision Support Systems: Theory and Application, Springer-Verlagg, Berlin, 1987.

Everest, G., "The Objectives of Database Management," in J. Tou, Ed., Implementation Systems: Coins IV, Plenum, 1974, pp. 1-35.

Gal, G. and McCarthy, W., "Operation of a Relational Accounting System," Advances in Accounting, Volume 3, pp. 83-112, 1986.

Hammer, M., "Reengineer Work: Don't Automate, Obliterate," Harvard Business Review, 1991, pp. 104-112.

Hammer, M. and McLeod, D., "Database Description with SDM: A Semantic Database Model," ACM Transactions on Database Systems, Volume 6, Number 3, September 1981, pp. 351-387.

King, R. and McLeod, D., "A Database Design Methodology and Tool for Information Systems," ACM Transactions on Office Information Systems, Volume 3, Number 1, 1985, pp. 2-21.

McCarthy, W., "Construction and Use of Integrated Accounting Models with Entity-Relationship Modeling," Working Paper 79-1, Michigan State University, 1979a.

McCarthy, W., "An Entity Relationship View of Accounting Models," Accounting Review, October 1979b, pp. 667-686.

McCarthy, W., "Multidimensional and Disaggregate Accounting Systems: A Review of the Events Accounting Literature," MAS Communications, July 1981, Volume 3, Nos. 1-2, pp. 7-13.

McCarthy, W., "The REA Accounting Model: A Generalized Framework for Accounting Systems in a Shared Environment," Accounting Review, July 1982, pp. 554-578.

McCarthy, W. and Gal, G., "Declarative and Procedural Features of a CODASYL Accounting System," in Entity-Relationship Approach to Information Modeling and Analysis, edited by P. Chen, E-R Institute, 1981.

McCarthy, W. and Rockwell, S., "The Integrated Use of First Order Theories, Reconstructive Expertise, and Implementation Heuristics in an Accounting Information System Design tool," Proceedings of the Ninth International Workshop on Expert Systems and their Applications, pp. 537-548, Avignon, France, 1989.

Penny, D. and Stein, J., "Class Modification in the GemStone Object Oriented DBMS," Proceedings of the Conference on Object-Oriented Programming Systems, Languages and Applications, pp. 111-117, 1987.

Porter, M., Competitive Strategy, The Free Press, New York, 1980.

Rockwell, S., Unpublished Ph. D. Dissertation, 1989.

Skarra, A. and Zdonik, S., "The Management of Changing Types in an Object-Oriented Database," Proceedings of the ACM Conference on Object-Oriented Programming Systems, Languages, and Applications, Portland, Oregon, September 1986.

Silberschatz, A., Stonebraker, M., and Ullman, J., "Database Systems: Achievements and Opportunities," Communications of the ACM, October 1991, Vol. 34, No. 10, pp. 110-120.

Simon, H., The Sciences of the Artificial, MIT Press, Cambridge, Massachusetts, 1985.

Sorter, G., "An Events Approach to Basic Accounting Theory," Accounting Review, January 1969, pp. 12-19.