An Introduction To Enterprise-Participant Data Model

Kevin Houzhi Xu
Department of Computer Sciences
Purdue University
West Lafayette, IN 47907

Bharat Bhargava
Department of Computer Sciences
Purdue University
West Lafayette, IN 47907

Abstract

A data model, called enterprise-participant (EP) model, is proposed. Unlike any other traditional data models including object oriented models, the enterprise-participant model takes a different angle to look at the real world. The concepts participant, enterprise hierarchy and biography hierarchy are introduced to construct databases representing the universe of discourse. Participants, as basic units, provide finer granularity than entities in traditional data models. Enterprise hierarchies provide recursive abstractions of enterprises from entire database applications to individual printable attributes. And biography hierarchies provide integrated view of all facts about entities. Because of its capacity to simplify the complexities of applications and of its natural hierarchical characteristics, the EP model promises: 1) more "natural ways" to represent data for database applications with increasing complexities, 2) a uniform, advanced manipulating language, and 3) world-wide unique participant names to immediately support data integrations and distributions in worldwide, distributed, and heterogeneous computing environments.

1. Introduction

While our life has been more and more relying on database systems, we haven't had a commonly acceptable data model to satisfy the requirements of increasing database applications. Since the middle of 1970's when the relational data model began to dominate database research areas, the relational data model has been criticized for its too simple mechanism for representing complex objects and supporting useful semantic concepts [5], [7], [13], [20], [21], [25]. Although relational database management systems are dominating commercial database marketing places, applications in CAD, CAM, CASE, and other knowledge-based applications suggest that the relational data model is not the future of database management systems [25], [15], [1]. With the dream of modeling the real world "in ways that correlate more directly to how data arise in the world" [12] around the beginning of the 1980's, researchers introduced semantic data models with typically richer abstractions such as generalization and aggregation, and derived attributes [10], [5], [20], [19]. However these are too complicated to be completely implemented [10]. After the middle of the 1980's, object-oriented data models, the marriage of the object-oriented programming technologies with the previous data models including the network models, the relational model and some modeling concepts of the semantic models had seemed to be the future of the database management systems [15], [24], [3]. However, we believe that object-oriented data models inherited many of flaws while they absorbed many of good features from the previous models [32].

Data distributions and integrations used to be issues of DBMSs beyond data models. In other words, traditional data models concentrated on data representations and manipulating languages with the assumption of centralized data. While the transparency (data independence) of data distributions and integrations was the central focus of the researches in distributed database management systems, researches in data models ignored implications of data models themselves to the implementation of data distributions and integrations. Consequently, Additional databases "dictionaries" and associated software "clients" must be provided to support data communications in distributed environments. Even worse, there was no theoretical solutions for distributed database designs and query optimizations in the distributed DBMSs supported by traditional data models [18]. To be fitted into distributed and heterogeneous environments, two hierarchical-based object-oriented models, which sacrificed some modeling powers of network-based object oriented models, have been established by ISO/ITU [28], [29], [30], [23], [11], and [31], [22], as the standard data communication protocols at the application layer. The first standard protocol, X.500, provided "directory" and its corresponding "abstract services" as the common data model and the common manipulating languages respectively for general data manipulations across

interconnected (distributed and heterogeneous) computing environments. The second standard protocol, X.700, derived from X.500, provided "Naming Hierarchy" and CMIP (Common Management Information Protocol) for the specific environments of telecommunication network management systems. The most impressive features of X.500 (we will use X.500 as the typical protocol to represent both X.500 and X.700 in the rest of the paper) are the concept of world-wide unique object identities (Distinguished Names) and the concept of the advanced manipulating language - the "Directory Abstract Services". Since X.500, as the common data model, is different from others supported by current available DBMSs, interpreters (called "data gateways") sitting above database management systems, must be built to talk to each others by speaking the "international languages" - "Directory Abstract Service". The overheads both of the extra amount of the development work and of the system complexity are significant due to the additional interpretations. Another problem with the ISO/ITU standards is that they have too limited power to model future applications with increasing complexities, while advanced networking technologies in the future will allow complex data transactions to be executed across broadband network systems or wireless network systems. For example, we may be expecting that a mobile user can, through a mobile computer, interact with a complex database system in his/her own business unit, and also can interact with various public information servers such as weather report in Chicago, purchasing order services of a supermarket in Los Angles, and the stock information in New York. A standard, powerful data model and data manipulating language will be essential to support this scenario.

With advanced graphical displaying technologies, SQL or SQL-based languages are no longer sufficient to support a good graphical user interface. In recent commercial database applications, vendors like to use Motif-based or CAD-based tools to display semantically related data (or symbols representing data) on screens. The extra interpretation from SQL has to be done to feed the required data by the GUI. Actually, SQL-based languages were rarely used by end users; instead they are only used by system administrators or as embedded languages in programming environments. Although the recursive query languages are supported by object oriented data models for applications, e.g., CAD, with only "part-of" relationships [14], [15], [3], other general-purpose applications do not have the corresponding advanced query languages supported by object oriented data models. Providing uniform, advanced manipulating languages in database management systems can better fit user interface requirements.

In this paper, we attempt to develop a more powerful data model, called the enterprise-participant (EP) model, with a uniform, advanced manipulating language and the potential to immediately support data integrations and distributions in distributed and heterogeneous computing environments. The EP model is more powerful in the sense that database designers can apply the EP data model to applications with increasing complexities in intuitive fashion. The EP-supported manipulating language is uniform and advanced in the sense that a language better than SQL-based, similar to the recursive query languages for composite objects in object oriented data models and the directory abstract services in X.500, is applied to any kind of applications. The EP model supports data distributions in the sense that no application-dependent dictionaries and clients are needed in homogeneous distributed environments so that the simple combinations of local schemata are the "global schema". The analogy of homogenous distributed database management systems in the EP model will be network file management systems with the file mounting functionality in Unix systems. The EP model support data integrations in the sense that the EP model can be an efficient common data model in the data communications of worldwide heterogeneous database systems.

In the rest of the paper, we first give the motivation to choose hierarchies as the fundamental structures in the EP model. Section 3 introduces the core concepts of the EP model. In Section 4 summarize the features of the EP model.

2. Hierarchies

Humans have long used hierarchical organization of information to help them better understand the world [8], [20]. The physical objects in the world are hierarchically structured. Logical or abstractive applications can be organized into hierarchies; for example, the contents in a book with nested chapters and sections. New terminologies can be used to abstract relationships between entities; for example, a "reservation" abstracts the relationship among a person, a date, and a hotel, which was called aggregation in database researches. The traditional hierarchical model would have been the most popular data model if any relationships between entities in the world were simply one-to-many or one-to-one relationships [26], [4], [6], [16], [9]. The most impressive features of a tree-like hierarchical model is the simplicity of the tree structure representing the relationships between entities, which benefits both the user interfaces and the system implementations. Although network-based models, including object-oriented models are powerful for capturing any kind of relationships between entities,

network-oriented relationships formed by object references between entities complicated user interfaces and hence did not support a general-purpose advanced manipulating language. One of the evidences of this point is the experiences of the Hypermedia/Hypertext researches and commercial productions, where the researchers or designers intended to provide the end users with the hierarchical views for underlying databases with network relationships [2], [27].

"Many-to-many relationships" from the viewpoint of the relational data model, and the "cyclic relationships" from the viewpoint of the network-based object-oriented data models, which exist in the real world, have been blocking researchers from further developing a hierarchical-based model for database applications after recognizing the limitations of the hierarchical models in 1970's. Can we eliminate, by taking a different approach, the dilemma caused by the "many-to-many relationships" and the "cyclic relationships" in a hierarchical-based data model? "Yes" is the answer provided in this paper, which becomes the foundation of the research. The idea is to decompose entities in the world into multiple, semantically independent components so that each component has only a single relationship with the rest of the world. In the next, we provides an example of "many-to-many relationships" to illustrate the general idea. Section 3 will formally define the EP model.

The relationship between entities class and student is a many-to-many relationship from the viewpoint of the relational data model. By closely looking at the facts of a student involved in two classes, we notice that the involved facts can be decomposed into two disjointed facts, and each one is solely determined by its corresponding class. For example, a student takes CS500 and CS541. The student takes two classes at different times, learns the different contents, and get two grades for the two classes. Other than depending on this student's intelligence and study attitudes, the fact {the student took CS500 at 2:00 pm, graded A} and the fact {the student took CS541 at 10:00 am, graded C} are only dependent on class CS500 and class CS541 respectively. Although the student's dropping one course may improve the student's performance in the other class in the real life, we can say approximately that to create, delete, or update the information of {the student took CS500 at 2:00 pm, graded A} neither impacts the fact {the student took CS541 at 10:00 am, graded C}, nor impacts the other properties or behaviors such as the student's name and the fact that he played basketball at night. By decomposing the facts related to the entity student, the relationship between class and student can be simply viewed as superior and subordinate relationship, and the "many-to-many relationship" has been eliminated.

3. Modeling concepts

This section formally introduces the core concepts of this data model: entity, biography, enterprise (or participation), participant, biography hierarchy, enterprise hierarchy, and participant identities. Before moving forward, we review some common important concepts.

The world is too big and too complex to be captured and recorded completely. We establish a database only for a special aspect of the world called the universe of discourse (or the application) for a special purpose. The definition of database here is taken from [8]: A **database,** as the objective of an application, is a collection of related data which represents a special aspect of the real world. The application is actually an enterprise as we can see after its formal definition is given below. An enterprise can be an organization, an activity, a plan, or an abstract concept. The purpose of designing a database is to collect and organize useful data and to filter out irrelevant data so that the enterprise can be effectively and efficiently represented. For example, the organization of a university, a Christmas party, and the content of mathematics are all the enterprises. An **entity** is something that exists as a discrete unit in the real world; for example, a student in the university, a rectangle in mathematics, and a scientific theory.

3.1. Biography

The term "biography" used to be an account of a person's life which collects all information about the characteristics, activities, and achievements of this person. We borrow this terminology to define that a **biography** is a collection of the properties of an entity including the characteristics, the behaviors, and the performances which are associated with the life of the entity. For example, the biography of John Smith in a database may looks like: {typed a person; called "John Smith"; born on 02/23/1960 in USA; majored in CS in Purdue University in 1980; graded A in course CS500; married in 1990; working as project manager in ABC company; playing baseball as pitcher in DEF baseball club}. This object not only contains the permanent properties of John Smith such as name and birth, but also contains the roles and the performances in the school, the company, the family, and the baseball club. Another example of the entity biographies in a database might look like: {typed a class; called CS500; offered since 1980; opened with 40 students and with Tom as the instructor in the spring of 1985; opened with 45 students and with Peter as the instructor in 1990}. This biography collects the facts of the entity CS500 in some school.

To our knowledge, we couldn't find a better

terminology than "biography" to represent the concept defined above. There are the following two reasons for us to make this choice. One is that a biography must be a data collection in an existing database about an entity's history, but not the history itself. While the history of an entity is a set of the solid facts that occurred in the real world about the entity, a biography is a subset of the history retrievable from the database. The second reason is that a biography is about an entity, which is active, not passive. Like the concept object in the object oriented concept, we treat an entity as being active. We didn't choose "entity" in the position of "biography" because "entity" is an ambiguous constructor used by traditional models in the sense that it was not clear how many facts of a real entity should actually be wrapped in "entity".

3.2. Participation and participant

Before we define concept participant, we introduce terminology participation. A **participation** is a subset of data from a biography, which corresponds to the properties (facts) the entity possesses when it participates in a particular environment such as an activity or an organization. A participation is syntactically disjointed with the rest of participations in the biography, There is one and only one **primary participation** in the biography which consists of the minimum properties to form the entity in the world. For example, the biography "John Smith" in the database given in Section 3.1 can be decomposed into the following participations: {typed a person; called "John Smith"; born on 02/23/1960 in USA} which is the primary participation; {majored in CS in Purdue University since 1980}; {graded A in course CS500}; {married in 1990}; {working as product manager in ABC company}; and {playing baseball as a pitcher in DEF baseball club}.

A **participant** is the abstraction of a participation of an entity, which surrogates the entity in the participation. From the definitions in Section 3.3, we will see that participation is the equivalent concept of enterprise, and a participant is a physical node in an enterprise hierarchy, and physically belongs to its participation.

3.3. Enterprise and its hierarchy

At the beginning of this section, we mentioned that an organization, an activity, a plan or the content of a book can be an enterprise. After the concept participation and participant were given, we formally define the concept enterprise. Our definition is a generalization of its conventional definition.

An enterprise is the state of a thing in the real world at a given time. A participation in the EP model represents "a thing". Therefore, a participation is an enterprise. On the other hand, we view "a thing" in the world as a participation of an entity with respect to a particular environment. In the EP model paradigm, therefore, we redefine that an **enterprise** is a participation. We call a thing as an enterprise with respect to the thing itself, and call it as a participation with respect to its environment. For example, John Smith's {graded A in course CS500} alone is an enterprise, and it is a participation in the course CS500.

An enterprise consists of a participant and zero or multiple sub-enterprises. The participant in the enterprise is the abstraction of the enterprise itself, and the sub-enterprise(s) is (are) the components of the enterprise, or the participations of other entities under the given enterprise. For example, a party is an enterprise, which consists of the participant "The party", and a number of components: the people participating in the party, the room holding the party, and its organizer.

From the definition of the enterprise, we noticed that it can be recursively decomposed into a hierarchy consisting of participants only. The **enterprise hierarchy** of an enterprise is the hierarchical structure representing the enterprise itself, which is formed by semantically and recursively decomposing the enterprise into sub-enterprises until all participants at leaves don't have to have further sub-enterprise(s). Figure 1 gives a complete enterprise hierarchy for a sample university database in the EP model, where each node is a participant. The contents of participants are discussed in Section 3.5 and 3.6.

We call the participant abstracting an enterprise as the **superior participant** of the enterprise, and the participants below the enterprise as the **subordinate participants**. In an enterprise hierarchy, a participant acts as a superior with respect to the enterprise that the participant abstracts, and acts as a subordinate with respect to the higher level enterprise where the participation is a component of the enterprise.

We summarize the features and the constrains of the EP model below.

1. As an enterprise, a given application is represented by a single enterprise hierarchy.

2. An application represented by the EP model is extensible in two dimensions. One is the further specification to a participant at a leaf of the enterprise hierarchy. For example, a department head can be further specified by adding subordinates: performance, agenda, and secretaries. Another dimension to extend applications is to integrate the enterprise hierarchies representing individual applications to form a higher level abstraction. For example, we may integrate all databases of the universities in the United State to form an integrated

database called "U.S. Universities". The extensibility of enterprise hierarchies determines the possibility of the database distributions and integrations.

3. An enterprise hierarchy is a pure tree structure. This is the major goal of this research. This feature contributes to the potential of a good user interface (the manipulating language) and better system organizations and performances associated with the EP data model (see [32] for more information). The tree structure implies that a participant only relies on its enterprise and an enterprise dominates its participants.

4. A participant can't be created unless its superior has been created. And the deletion of a participant implies that its child participants are deleted automatically.

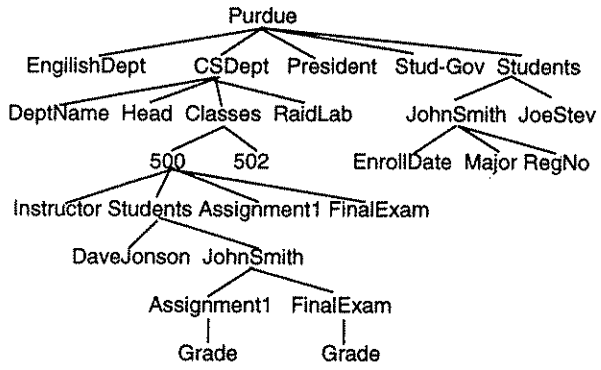

Figure 1. A sample Database with EP Model

3.4. Biography hierarchy

After defining the concepts biography, participant, enterprise (or participation) and its hierarchy, the concept biography hierarchy is introduced in this section. The **biography hierarchy** of an entity is a hierarchical structure representing the biography of the entity, in which nodes are participations from the biography, and links represent dependent relationships between participations. The biography hierarchy is rooted at its primary participation, and is formed by linking all of its participations occurred in an entire enterprise hierarchy. The levels of the participations in the biography hierarchy are determined by the dependencies existed between the participations. In a biography (or a sub biography) hierarchy, the root of the hierarchy is called the parent participation of its successive participations in the hierarchy, and the successive participations are called the child participations. The participants corresponding to the parent participations and child participations are called **parent participants** and **child participants** respectively. For example, John Smith's {majored in CS at Purdue University in 1980} is the parent participation of his {graded A in course CS500}, and in turn the latter is the child participation of the former. Another example is that a person as a participation in the society is the parent participation of him/herself as a scientist in a research lab. Two participations can be either a pair of the siblings under a common parent participation or can have parent/child relationship. For example, John Smith's participation {majored in CS at Purdue University in 1980} and participation {working as product manager in ABC company} are siblings, and {majored in CS at Purdue University in 1980} and {graded A in Class CS500} have parent/child participation relationship. Figure 2 shows the biography hierarchy for John Smith's biography in the database given in section 3.1. Note that each node represents a sub-enterprise in its corresponding enterprise hierarchy.

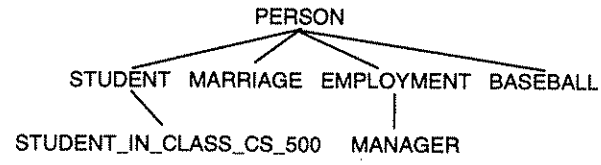

Figure 2. A Sample Biography Hierarchy

Biography hierarchies have the following features and constrains.

1. Biography hierarchies are represented implicitly in enterprise hierarchies. Participations (or syntactically the participants of the participations) are the intersections of the two types of the hierarchies.

2. Two siblings in the biography hierarchies can be created or deleted in any orders.

3. A child participant can't be created unless its parent participant exist.

4. The deletion of a participant implies that the child participations of the participant are deleted automatically.

The biography hierarchy provided a way to retrieve any information in the biography of an entity by giving participant.

3.5. Participant identity

As one of the core concepts of the EP model, participant identity (Distinguished Names and Relative Distinguished Names) are briefly introduced. The contribution of the participant identity to data distributions and integrations was analyzed in [32].

Like the naming scheme in X.500, there are two names associated with a participant, **Relative Distinguished Name (RDN)** and **Distinguished Name (DN),** in the EP model. the RDN of a participant is the name of the

participant unique under its enterprise, and it is an element as a part of the participant. The DN of the participant, by given an enterprise hierarchy, is the ordered list of the RDNs of the participants along the path from the root of the database to the participant itself. For example, in Figure 3, the RDN of participant G is G, and its DN is /A/B/E/G. In the University database example, John Smith has the two RDNs called "JohnSmith" under /Purdue/Students and under /Purdue/CSDept/Classes/500/Students respectively, and the two corresponding DNs respectively called "/Purdue/Students/JohnSmith" which uniquely identifies the fact {majored in CS since 1980} and "/Purdue/CSDept/Classes/500/Students/JohnSmith" which uniquely identifies the fact {John Smith was graded A in class CS500}.

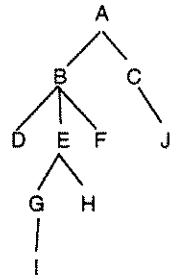

Figure 3. A Generic Enterprise Hierarchy

Like the names of simple variables in programming languages, RDNs of the EP model are the names mapping to the physical addresses where the participants are located and further mapping to the values of the participants (see Section 3.6 for the definition of participant values). In addition, RDNs are the strings representing the participations of the entities or are the abstractions of the corresponding enterprises. Thus like a structured variables' names in programming languages, the RDNs of the EP model are the names from which the physical addresses, where its immediate subordinates, children, superior and parent are located, can be inferred. Because the RDN of a participant is required to be unique among the immediate subordinates of its superior, its superior can be locally responsible for the naming authority of the participant.

Because the DN of a participant are the ordered list of RDNs of the participants along the path from the root to the participant itself within a given enterprise hierarchy, the DN is unique in the hierarchy and is a means to identify the participant any where in the enterprise hierarchy. Because of the recursive structure of enterprise hierarchies, a participant recursively has multiple relative DNs and each DN is associated with a superior of the participant in the higher level enterprise hierarchy. Because of the extensibility of enterprise hierarchies in the EP model (see Section 3.3), the uniqueness of DNs in the worldwide scope is guaranteed.

3.6. Values of participants

In the previous sections, we described that a participant are a physical node in an enterprise hierarchy, and a RDN is one of elements of a participant. In this section, we will define the values of a participant which are the second elements of participants.

Although a participant has up-stream and down-stream connections with other participants through either superior/subordinate relationship or parent/child relationships, the connections are maintained by systems. In other words, what users can manipulate are RDNs and values of participants only. Hence, the only elements of a participant are its RDN and its value. While RDNs are represented by strings uniformly across all participants, the values of participants differentiate the various types of participants which are the issues to be discussed in this section. There are four types of participants: primary participant, extended participant, atomic participant, and user-defined participant.

1. Primary participant. A primary participant is the participant representing the primary participation of an entity. Like object identifiers in object oriented data models, values associated with primary participants are system-generated. While DNs uniquely identify their participants in a given time, values of primary participants uniquely identify their entities along the entire life of a databases. See [32] for more discussion about the necessity and the scope of the uniqueness of both DNs and values of primary participants.

2. Extended participant. An extended participant is a child participant. We call it extended participant in the sense that it is an extension of its primary (recursively through parent participants), and depends on its parent participants. Values of extended participants are pointers to its parent participants (either physical addresses or the DNs of its parent participant depending on different implementations). For example, "JohnSmith" under /Purdue/CSDept/Classes/500/Students has a value pointing to /Purdue/Students/JohnSmith. Intersections between enterprise hierarchies and biography hierarchies are located exactly at values of extended participants.

3. Atomic participant. An atomic is a participant which has a system-supported (build-in) value. For example, the value of the participant "Grade" under /Purdue/CSDept/Classes/500/Students/JohnSmith is 'A', a character. We call this type of participants as atomic participants in the sense that they are non-decomposable participants located at leaves of enterprise hierarchies. In other words, atomic participants, supported by computer systems, are the

foundations of enterprise hierarchies and biography hierarchies in the EP model.

4. User-defined participant. A user-defined participant is the one representing an enterprise structured by a prototype other than the EP model. The examples of the user-defined participants are unstructured complex objects such as video, voice, and image; derived attributes such as minimum, maximum, average, and sum; and other non-EP applications. Values of user-defined participants are managed by non-EP functions. User-defined participants provide a way to provide functions beyond the capacities of the EP model, and provide a way for systems in the EP model to communicate with non-EP systems.

Before we finish the definition of the EP model, we give Figure 4 for a sample data schema in EP model, from which readers might have further understanding to the EP model. In the figure, "pri" stands for primary, "parti" stands for participant, regular lines stand for one-to-one fixed relationships, boldface lines stand for one-to-many dynamical relationships. Note that data schema in the EP model is not covered in this paper, and the symbols in the figure are used for the discussion only. See [32] for more information.

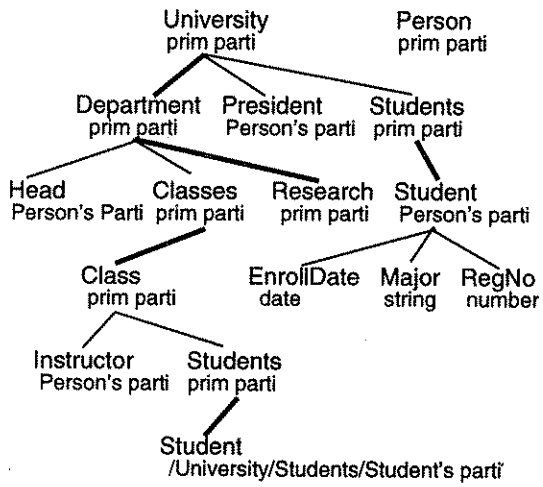

Figure 4. A Data Schema in EP Model

4. Summary

This paper presented the EP model, which is a different approach from the traditional data models including the object oriented data models. An application, as the initial and the highest-level enterprise, is semantically and syntactically refined into its enterprise hierarchy, where the only elements are the participants, and the relationships between the participants are represented by the simple superior and subordinate relationships. More than aggregations (composite objects) in semantic data models, enterprise hierarchies provide uniform, complete views of entire database applications, which are recursive abstractions of enterprises from database applications themselves to individual printable attributes (atomic participants). The biography hierarchies of entities, more than entities in semantic data models, integrate the participations of the same entities together and provide a complete view of entity related data. A participant syntactically is an intersection of the enterprise hierarchy and its biography hierarchy, and semantically is the abstraction of the facts associated with the entity when it participates in a specific environment such as an activity, an organization, a plan, or a logical concept.

In addition to the data representation of the EP model, supporting an uniform, advanced query language, and data distribution and integration are two other major objectives of the EP model. Because of the simplicity and the uniformity of enterprise hierarchies and biography hierarchies, an advanced manipulating language similar to recursive query languages for composite objects in object oriented data models [14], or the directory abstract services in X.500, is uniformly applied to any applications. While "the CPU as an island, contained and valuable in itself, is dying in the nineties" [17], the EP model takes the responsibility of data distribution and integration. While the distribution transparency is preserved from both end users and the design of centralized conceptual schemata, participants Distinguished Names in the EP model reduce the complexity of data address resolutions from O(N) in the traditional DBMSs to O(Lg N), hence the EP model practically support very large volume of data in the worldwide scope. Further, with system-managed superior/subordinate and parent/child relationships, DNs of participants eliminate the necessity of application-dependent "dictionaries" - secondary databases, and their associated "client" software. Due to the limited space available, readers who are interested in these topics can refer to [32] for more information.

We did not address the issues of data security, transaction, versioning, replication, and system implementation in this research, but we believe that the EP model promotes the optimized solutions for the above issues.

Acknowledgment: Thanks to Richard Jesmajian who provided the related information about X.500.

References:

[1] Ilsoo Ahn. Database Issues in Telecommunications Network Management. SIGMOD 94 - 5/94 Minneapolis, Minnesota, UAS.

[2] Rodrigo A. Botafogo, Ehud Rivlin, and Ben Shneiderman. "Structural Analysis of Hypertexts: Identifying Hierarchies and Useful Metrics". ACM Transactions on Information Systems, Vol. 10, No. 2, April 1992, Pages 142 -180.

[3] Michael J. Cary, David J. DeWitt, Scott L. Vandenberg. A Data Model and Query Language for EXODUS. Proc. ACM SIGMOD Intl. Conf. on Management of Data, Chicago, Ill., June 1988, pp. 413-423.

[4] Donald. D. Chamberlin. Relational Data-Base Management Systems. Computing Surveys, Vol. 8, No. 1, March 1976.

[5] Peter Pin-Shan Chen. The Entity-Relationship Model - Toward a Unified View of Data. ACM Transactions on Database Systems, Vol. 1, No. 1, March 1976, Pages 9-36.

[6] E.F. Codd. A Relational Model of Data for Large Shared Data Banks. Comm. ACM 13, 6, June 1970, Pages 377 - 387.

[7] E.F. Codd. Extending the Database Relational Model to Capture More Meaning. ACM Transactions on Database Systems, Vol. 4, No. 4, December 1979, Pages 397 - 434.

[8] Elmasri/Navathe. Fundamentals of Database Systems, Second Edition. Benjamin/Cummings Publishing, 1994.

[9] James P. Fry and Edgar H. Sibley. Evolution of Data-base Management Systems. Computing Surveys, Vol. 8, No. 1, March 1976.

[10] Michael Hammer and Dennis McLeod. Database Description with SDM: A Semantic Database Model. ACM Transactions on Database Systems, Vol. 6, No. 3, September 1981, Pages 351 - 386.

[11] Baha Hebrawi. OSI Upper Layer Standards and Practices. McGraw-Hill, 1992.

[12] Richard Hull, and Roger King. Semantic Database Modeling: Survey, Applications, and Research Issues. ACM Computing Surveys, Vol. 19, No. 3, September 1987.

[13] William Kent. Limitations of Record-based Information Models. ACM Transactions on Database Systems, Vol. 4, No. 1, March 1979, Pages 107 - 131.

[14] Won Kim, Hong-Tai Chou, and Jay Banerjee. Operations and Implementation of Complex Objects. IEEE Transactions on Software Engineering. Vol. 14, No. 7, July 1988.

[15] Won Kim. Introduction to Object-Oriented Databases. The MIT Press, 1990.

[16] Ann S. Michaels, Benjamin Mittman, and C. Robert Carlson. A Comparison of the Relational and CODASYL Approaches to Data-Base Management. Computing Surveys, Vol. 8, No. 1, March 1976.

[17] Richard Mark Soley, Christopher Stone. Object Management Architecture Guide, Third Edition. Jon Wiley & Sons, Inc., 1995.

[18] M. Tamer Ozsu, Patrick Valduriez. Principles of Distributed Database Systems. Prentice Hall, 1991.

[19] David W. Shipman. The Functional Data Model and the Data Language DAPLEX. ACM Transactions on Database Systems, Vol. 6, No. 1, March 1981, Pages 140-173.

[20] John Miles Smith and Diane C.P. Smith. Database Abstractions: Aggregation and Generalization. ACM Transactions on Database Systems, Vol. 2, June 1977, Pages 105 - 133.

[21] John Miles Smith and Diane C. P. Smith. Database Abstractions: Aggregation. Communications of the ACM, June 1977, Vol.20, Number 6.

[22] William Stallings. SNMP, SNMPv2, and CMIP - The Practical Guide to Network Management Standards. Addison Wesley, 1993.

[23] Douglas Steedman. X.500 - The Directory Standard and its Application. Technology Appraisals. [94]

[24] Michael R. Stonebraker and Lawrence A. Rowe. The Design of POSTGRES. Proc. ACM SIGMOD Intl. Conf. On Management of Data, Washington, D.C., May 1986.

[25] Michael R. Stonebraker. Future Trends in Database Systems. IEEE Trans. Knowledge and Data Eng., Vol. 1. No. 1. Mar. 1989, Pages 33 - 44.

[26] D. C. Tsichritzis and F. H. Lochovsky. Hierarchical Data-Base Management: A Survey. Computing Surveys, Vol. 8, No. 1, March 1976.

[27] Kenneth Utting and Nicole Yankelovich. Context and Orientation in Hypermedia Networks. ACM Transactions on Information Systems, Vol. 7, No. 1, January 1989, Pages 58 - 84.

[28] ITU-T Recommendation X.500 (1993). Information Technology - Open Systems Interconnection - The Directory: Overviews of Concepts, Models, and Service.

[29] ITU-T Recommendation X.501 (1993). Information Technology - Open Systems Interconnection - The Directory: Models

[30] ITU-T Recommendation X.511 (1993). Information Technology - Open Systems Interconnection - The Directory: Abstract Service Definition.

[31] CCITT Recommendation X.700 (09/92). Management Framework for Open Systems Interconnection (OSI) For CCITT Applications.

[32] Kevin Houzhi Xu. An Enterprise-Participant Data Model for Increasing Complexities of Database Applications, Ph.D. Thesis Proposal. Department of Computer Sciences, Purdue University, West Lafayette, IN 47907, 1996.