On the Relevance of Classification Theory to Database Design

Jeffrey Parsons
Faculty of Business Administration
Memorial University of Newfoundland
St. John's, Newfoundland, Canada A1B 3X5
jeffrey@morgan.ucs.mun.ca

A major activity in designing databases is identifying the classes of things about which data will be kept. Current approaches to database design involve constructing an integrated schema, or global set of interrelated classes, which may be used by many people with different views as to which classes are needed and how they should be defined. These approaches are inconsistent with research on classification indicating that there is generally no globally correct way of classifying entities given diverse views.

This paper discusses a stream of research which applies some classification principles to database design. We present a data model explicitly based on classification research. Subsequently, we examine implications of the model for flexibly handling diverse views of the class requirements for a domain, for developing methods and techniques to design databases, and for implementing data structures which support dynamic classification. The paper concludes by proposing that applying classification research to database research can lead to advances in the former, as well as in the latter.

1. DATABASES AND CLASSIFICATION

Modern information technologies allow organizations to collect large volumes of data about many aspects of their operations. For instance, department store credit cards have led to the collection of a wealth of information about customer purchasing behavior, including the dates, times, and locations of purchases. However, the mere collection of such data does not guarantee that it can be effectively used; it must also be organized in a manner that unanticipated requests for information can be satisfied. ***Database technology*** aims to organize data to allow its timely retrieval for various legitimate users and uses.

Data modeling and ***database design*** have emerged as a collection of techniques for determining, verifying, and codifying data requirements for various information systems applications or uses. One key element of data modeling is identifying the classes or categories of things about which data are to be kept, which attributes of those things must be kept track of, and what relationships exist among instances of different classes. For example, an application to maintain course registration records at a university might identify relevant classes of things such as students, courses, instructors, sections, and rooms. Relevant attributes of students might include name, address, date of birth, year, and major. Important relationships among classes might include: "students enroll in courses", "instructors teach courses", and "courses have multiple sections".

A widely used technique for data modeling, the Entity-Relationship (ER) model, uses diagrams to express data requirements in terms of classes (termed *entity types* in ER jargon), relationships among classes, and attributes, which jointly are referred to as a ***conceptual schema.*** Conceptual schema diagrams provide the basis for designing the files that will store the data. The primary implementation technologies for databases today are ***relational*** and ***object-oriented*** database systems, both of which are also based on organizing data according to a fixed schema of classes. In a relational database system, each file (called a *relation*) is defined in terms of a number of fields

ISSN: 2324-9773

and corresponds loosely to a class (e.g., a student relation, a course relation) with corresponding attributes (Teorey, 1994). Similarly, in an object-oriented database system, class diagrams provide the basis for defining a schema of object classes which implement the database (Cattell, 1994).

The ER model works reasonably well as a design tool, and relational or object database systems as implementation tools, in applications for which different users agree on what the relevant classes are, and on the properties which define membership in these classes. However, problems arise when a database is being designed for many users or groups, each having a different perspective on which classes are relevant, and how they should be defined to include or exclude members. Research on database design in this context focuses on combining or "integrating" the views of different users into a single global class schema (Batini et al., 1986; Navathe et al., 1986). Discrepancies in the identification and definition of classes by different users are seen as issues to be resolved in the ***view integration*** process. For example, in a university, the class "student" may be defined differently by the registrar and scholarship offices. This may not mean simply that the two offices are interested in different attributes of the same population; instead, the scholarship office may consider as students only those scholarship applicants who qualify for scholarships, whereas the registrar's office may be interested in all people who are currently registered in courses at the institution. The former group need not be a subset of the latter, since some scholarship applicants may not currently be enrolled at the university.

The issues involved in reconciling user views are complex. Integration requires, as a prerequisite, identifying correspondences between entity types, attributes, and relationships, as well as resolving synonyms, homonyms, and conflicts. Hence, for complex databases, designing a global schema can be difficult and time-consuming, even if some activities can be automated (Navathe et al., 1986). Moreover, if changing data requirements require the evolution or redefinition of classes over time, updating a global schema is difficult (Tresch and Scholl, 1993; Lerner and Habermann, 1990).

This paper contends that the assumptions about classification on which current approaches to database design rest fail to account for what has been learned about human classification in recent years, and that this failure contributes to the problems of modeling data requirements for multiple users. If one accepts that the use of classification to organize data should be consistent with the theory and research on human classification, new models and techniques for database design are needed. The remainder of this paper discusses a line of research aimed at applying classification theory to several problems in database design. Section 2 presents a data model based on classification principles. Section 3 discusses implications of this model for supporting multiple users. Section 4 examines the role of the model in developing a new methodology for data modeling. Section 5 discusses requirements for providing implementation support for the model. Section 6 summarizes what has been learned so far and outlines further research plans in this area.

2. A DATA MODEL BASED ON CLASSIFICATION RESEARCH

Classification research recognizes that classifying perceptions is vital to human survival and adaptation, and attempts to explain the nature of concepts (categories/classes) and why humans classify things (Lakoff, 1987; Smith and Medin, 1981). According to Lakoff (1987, p. 1):

ISSN: 2324-9773

Parsons, J. (1994). On the Relevance of Classification Theory to Database Design. 5th ASIS SIG/CR Classification Research Workshop, 129-138. doi:10.7152/acro.v5i1.13782

Without the ability to categorize, we could not function at all, either in the physical world or in our social and intellectual lives. An understanding of how we categorize is central to any understanding of how we think and how we function ....

Concepts provide two primary functions that enable us to cope physically and socially: ***cognitive economy*** and ***inference*** (Rosch,1978; Smith, 1988). *Cognitive economy* means that concepts reduce the cognitive burden associated with storing and organizing knowledge. A concept abstracts important similarity among its members. Knowing that a thing belongs to some concept is sufficient helps establish its similarity to other instances of the concept. Likewise, new concepts are distinguished from existing ones if there are "meaningful differences" among their respective instances (Rosch, 1978).

Inference means that conclusions may be drawn about unobserved properties by classifying a thing as a member of a concept based on observed properties (Smith, 1988). According to Rosch (1978, 0.29), "the material objects of the world are perceived to possess high correlational structure." If a thing is classified by observing only some of its properties, it may possess additional properties of interest by virtue of being an instance of the specified concept.

Early studies in classification assumed that concepts have well-defined boundaries and exist independent of human perception. This so-called "classical" view has been shown to be inadequate to account for many concepts that have vague or indeterminate boundaries (e.g., Rosch, 1978; Smith and Medin, 1981). In addition, Lakoff (1987) argues that empirical research from a variety of fields suggests that classes are artifacts of human perception, constructed as useful abstractions for specific purposes. Hence, different people (or the same person at different times) may organize knowledge about things according to a different set of classes or categories.

In database design, entities important to an organization (such as products, equipment, contracts, customers, and suppliers) can frequently be placed in classes where membership is determined by the properties that define the entity types. This suggests that a "classical" view of concepts is an appropriate starting point for data modeling (Parsons, 1994a). However, data modeling research on schema integration and evolution clearly indicates that different users can disagree both on what are the important classes, and on what attributes define classes that are jointly recognized as important. According to Stamper (1987, p. 49),"despite all the subjectivist language of 'concepts', [the database community] adopts a naive assumption of a single valid view of the world."

In short, a data model based on classification theory must account for the subjectivity of classes and offer mechanisms to support multiple classification schemas over a domain of entities or things, and at the same time allow for well-defined classes with crisp boundaries. Consequently, our research takes a hybrid approach. The view that classes are defined by properties possessed by all instances serves as the basis for formalizing the model.[1] [2] However, such classes are not taken

1. This may be an accident of the fact that, historically, information systems have primarily represented information about things that can be placed in relatively well-defined categories. The assumption of well-defined categories can be relaxed in the proposed model by introducing elements of "fuzzy" membership.
2. Since prototype or exemplar models of concepts (see e.g., Smith and Medin, 1981) would likely suggest different representation constructs, comparative modeling work based on these theories is recommended as future research.

ISSN: 2324-9773

to be an objective characterization of reality, but a construction which identifies similarities among instances that are important to a group of users. The remainder of this section elaborates on aspects of this hybrid view by informally discussing aspects of the MIMIC (Morphological Information Model of Instances and Classes) data model (the model is formally defined in Parsons, 1994a).

To accommodate the position that classes abstract perceived similarities among instances, MIMIC proposes two primitive modeling constructs — ***object*** and ***property*** — which recognize that things exist independent of any classification, and that subsequent classification should be based on the properties that these instances possess. In the model, an ***object*** is a surrogate which designates the existence of some thing of interest. For example, the string "Jeff" may be taken to refer to some, as yet unclassified, real world entity. A ***property*** is a function which describes an object along some dimension. The model distinguishes three kinds of properties. ***Structural*** properties describe objects independent of other objects. For example, birthdate may be identified in an application as a structural property. In that case, birthdate (Jeff)=63/05/01 means that the property or function has the value "63/05/01" for the object "Jeff". ***Relational*** properties associate objects with other objects. For example, teaches (Jeff)=b6300 might indicate a relationship between the yet unclassified objects "Jeff" and "b6300". ***Behavioral*** properties constrain the allowable changes to values of structural and relational properties. For example, a university may constrain that an instructor may not be permitted to teach a certain advanced course unless s/he has previously taught the prerequisite to that course.

The role of classification in organizing knowledge to support cognitive economy and inference indicates there is more to defining classes than simply grouping properties arbitrarily. Accordingly, MIMIC develops four *principles of classification*, based on cognitive economy and inference, that jointly limit the kinds of collections of properties which may be considered classes. The first two principles restrict which sets of properties can potentially form classes. The remaining principles restrict which classes can coexist. Briefly,[1] the principles are:

I. ***Abstraction from Instances:*** The properties defining a class should be shared by a non-empty set of objects.
II. ***Maximal Abstraction:*** Every property shared by all objects of some class should be part of the definition of that class.
III. ***Non-redundancy:*** A class should not contain exactly the union of the properties of any other classes.
IV. ***Completeness:*** Given a domain of properties and potential classes, every property should appear in the definition of at least one class in the set.

In MIMIC, a ***potential class*** is defined as a set of properties which satisfies principles I and II. A collection of properties which does not satisfy both these conditions is a poor abstraction of similarity in a domain. A ***class structure*** is a set of potential classes which satisfies principles III and IV, and may be interpreted as a view of the domain. A set of potential classes which does not satisfy these conditions does not provide good coverage of a domain.

1. Space precludes explaining the rationale for these principles here, other than to state that they support the notions of cognitive economy and inference. A detailed rationale for these principles is given in (Parsons, 1994a).

ISSN: 2324-9773

Parsons, J. (1994). On the Relevance of Classification Theory to Database Design. 5th ASIS SIG/CR Classification Research Workshop, 129-138. doi:10.7152/acro.v5i1.13782

In conjunction with the formal specification of the model, several theorems can be proved which show that MIMIC is a practical data model for multi-view domains (see Parsons, 1994a for details and proofs).

Theorem: For any domain, a class structure can be constructed.

The notion of class structure would be of little use unless a set of classes can always be formed which satisfies the four principles of classification outlined above.

Theorem: For any relevant universe containing more than one property, there exists more than one class structure.

This means there is no single "correct" set of classes for modeling an application. Therefore, the effect of this theorem is to challenge data modeling approaches which advocate identifying a fixed set of classes (see Section 3 for discussion).

Theorem: Every potential class belongs to some class structure.

This illustrates a complementarity between potential class and class structure. A potential class supports some aspects of cognitive economy and inference. The theorem shows that every potential class can be combined with other potential classes to support further aspects of economy and inference.

The next section considers the implications of MIMIC on the state of the art for handling multiple views in data modeling.

3. IMPLICATIONS FOR MULTIPLE VIEWS

The need for multiple views has been recognized in data modeling research on view or ***schema integration,*** which involves combining user views of the important entity types and reconciling differences to form an integrated conceptual schema (Batini et al.,1986; Navathe et al., 1986). Related research on: (1) ***schema evolution,*** which involves modifying a class schema as data requirements change (Banerjee et al., 1987; Gibbs et al., 1990; Tresch and Scholl, 1993); and (2) ***interoperability***, which involves the exchange of data between independent databases (Kent, 1991;Sheth and Larson, 1990), also points to the importance of diverse users views and the difficulty of supporting them in database design. In particular, the difficulties arise from the need to define a database in terms of an integrated class schema.

MIMIC offers a radically different and simpler treatment of multiple views through class structures. In pursuing the implications of Lakoff's claim that the world does not contain categories or classes, MIMIC separates the representation of data about objects by their identity and properties from the organization of data in various possible ways through classification. At the representation level, there are no classes. At the organization level, there may be as many class structures as are needed to meet the requirements of different user groups. Since the representation model is not based on classes, there is no need to define an integrated or global schema of classes.

ISSN: 2324-9773

Moreover, class evolution can be accommodated by restructuring/redefining classes at the organization level. The underlying data or representation is not affected by such reorganization.

Implementation approaches for relational or object database systems are based on classes (e.g., an object class schema or a set of relations). Although views may be supported by mechanisms which construct virtual classes from the stored classes, the implementation fixes a global schema. This approach to implementation is implicitly based on a model of categories in which requirements can be represented by a single correct schema. When that assumption is exposed, the question arises whether the need for schema integration and evolution techniques is an artifact of the categorization model supported by existing relational and object database implementations. While it may be argued that a global view is needed construct a shared database using existing relational or object-oriented database systems, this constitutes only a limitation of the implementation model rather than a requirement of data modeling.

To illustrate, consider the following example. In a typical business, the class CUSTOMER may be defined from at least two points of view. From an accounting perspective, CUSTOMER might be defined to include: Name, Billing Address, Credit Rating, and Purchases. From a marketing point of view, attributes of interest might include: Name, Contact Address, Occupation, Income, and Sales Potential. From these definition, it is clear the individuals considered to be customers by accounting may not be identical with those considered to be customers by marketing (e.g.,marketing may include "potential" customers).

In the traditional approach to data modeling, the definitions of CUSTOMER proposed by the accounting and marketing departments have to be reconciled to define a global entity type or object class CUSTOMER. If both departments are interested in the same entities, but each in only a subset of their attributes, the resolution is simply to define a global entity/object type CUSTOMER and define views for each department which present only a subset of the attributes of the globally defined class. If, however, the departments are interested only in sets of entities having different attributes, two other solutions are possible. First, a generic class CUSTOMER may be defined, containing the attributes shared by both departments. Subclasses of CUSTOMER (A_CUSTOMER and M_CUSTOMER) may be defined to contain the additional attributes of interest to users in each department. Alternately, a single class CUSTOMER could be defined to contain all attributes of interest to both departments, allowing null (or not applicable) values of attributes for entities. Views are created by extracting the attributes of interest to each department. In either case, the implementation is based on the global conceptual schema.

Both these solutions are inflexible in a changing environment. For example, if one department wishes to change its definition of customer, this may necessitate changes to the implementation schema. In the case of a relational database, this may involve redefining the customer relation, and modifying all the tuples to include new attributes or interest (or possibly remove attributes no longer needed). Similarly, if another user group (e.g., the shipping department) identifies a CUSTOMER class, and produces a third definition, the implementation schema may have to again be modified to accommodate additional attributes.

Such problems do not occur in MIMIC, either in principle or in an implementation based on the model (see Section 5). Objects and properties are schema-independent. A class is simply a set of

Parsons, J. (1994). On the Relevance of Classification Theory to Database Design. 5th ASIS SIG/CR Classification Research Workshop, 129-138. doi:10.7152/acro.v5i1.13782

properties, and can be redefined by adding properties to, or removing properties from, the set. The implications of this are: (1) multiple class structures (views) can be constructed over a shared base of objects and properties, and (2) classes can be redefined without changing the way the underlying data is structured. MIMIC does not use an integrated conceptual schema as the basis for modeling. As a result, data cannot be lost by deleting classes that are no longer needed. Moreover, by storing all data only by unique properties, no redundancy will result when classes are added since no property is stored more than once. The model is flexible since different class structures can be constructed over time without altering the basic choice of objects and properties. This is a surprising contribution since the model is based on classification theory but, in the end, relegates classes to a "derived" position in relation to objects and properties.

4. IMPLICATIONS FOR DATA MODELING METHODOLOGY

The dominant data modeling methods and techniques are schema-based, in the sense that they are used to identify and depict classes of entities (Teorey, 1994). While this may be appropriate when a single view of the application is modeled, we have seen that these techniques present challenges when there is a need to integrate multiple views to form a global schema or set of classes.In contrast, by separating the organization of classes from the representation of objects and properties, MIMIC suggests that, in eliciting data requirements from users, classes should be used in two ways. First, classes suggested by users can serve to identify important objects and properties in the domain and, thereby, in constructing the underlying representation. This does not mean that the modeler will seek to identify properties independent of classes (in real life, it is practically impossible to refer to things except by name or as members of a class). Instead, users will likely state data requirements in terms of entity types, such as "students" and "courses". However, these initially suggested classes should not be immediately accepted and included in an ER or similar schema. Instead, they should provide the starting point for identifying attributes. Second, given a set of objects and properties, one or more class structures can be built from them. The designer must keep in mind that the classes identified by one user or group need not be reconciled with those of another. Instead, multiple class structures may need to be identified. Under this approach, view integration becomes largely a non-issue. The only remaining view integration issues involve such activities as reconciling use of the same class name for different classes (homonyms), and different names for the same class (synonyms). These problems are more easily solved in a database consisting only of objects and properties (with agreed upon property names).

The idea of modeling from instances to classes has been proposed in both the object-oriented and artificial intelligence literature (e.g., Lieberherr et al., 1991; Stepp and Michalski, 1984). In particular, the Demeter system (Lieberherr et al., 1991)is particularly relevant to MIMIC. Like MIMIC, Demeter also deals with the abstraction of classes from instances; however, Demeter focuses on developing efficient algorithms for designing an "optimal" class schema, defined as one which minimizes the number of ***a-kind-of*** or ***a-part-of*** edges on a class graph. The rationale for this criterion is based on software engineering issues (Lieberherr et al., 1991, p. 223), and assumes a system will be implemented with a single class schema. By contrast, MIMIC suggests that, in selecting classes to model categories of things identified by human users, there is no "optimal" schema. Different users may legitimately define different class schemas over the same objects.

ISSN: 2324-9773

5. IMPLEMENTATION SUPPORT FOR THE MODEL

The approach to classification in MIMIC raises questions about implementing a database in the absence of an integrated conceptual schema. Clearly, a relational schema or object class hierarchy is suitable when there is a global schema. For instance, there are algorithms to map an integrated Entity-Relationship conceptual schema into a relational database schema. One approach to capturing multiple views at the implementation level is to use the view mechanism in relational databases. However, the designer must then choose which classes (from which views) to include in base relations. This may be difficult in the absence of theory to prescribe which views are "better" or should take precedence over others. Moreover, it does not solve the problems of schema evolution or sharing among multiple databases.

These problems can be avoided by using an implementation technology that separates the classification and view levels from the object and property levels (Parsons, 1994b; Parsons and Leung, 1994). The essence of this approach is to store all data about objects according to *functions* or tables, each representing a property. The domain of each function is a set of objects. The codomain is a set of values (where values are other objects in the case of relational properties).

In this approach, the intension of a class is represented by a pointer to a set of properties. The extension is the set of objects that jointly share the properties defining the class. Furthermore, a view is a pointer to a set of classes which constitutes one perspective on the important objects and properties, as well as of the important similarities abstracted through the classes in the view. In short, we attempt to make the implementation model a more direct model of "reality" by developing data structures that, in contrast with relational and object implementations, do not use any class notion. On top of this basic architecture, a view management component is added to define and manage different views of the important class abstractions.

Support for schema changes is inherent in a property-based implementation. In class-based systems, reorganizing the schema as domain changes occur (e.g., adding new properties to classes, or removing some classes as they become obsolete) is difficult and costly (Banerjee et al., 1987; Lerner and Habermann, 1990). In a property-based implementation, schema reorganization is independent of the implementation model. That is, if classes are pointers to sets of properties, it is simpler to modify the set of properties a class points to (or to add pointers and remove pointers) than to make such changes when the storage of data is linked to the classes that are defined (as in current relational and object systems). Moreover, detaching data from classes means that classes can be added, deleted, or redefined without any impact on the data. This important feature is analogous to the data independence argument in favor of database environments over file-oriented environments. Data independence separates the interface to data from its implementation, thereby allowing the implementation to change without requiring changes to programs using the data. ***Class independence*** separates the interface to data through classes from its storage. An advantage of this approach is that it avoids the need to reorganize a database at the object and property levels as the understanding of classes in the domain changes over time.

6. CONCLUSIONS AND FUTURE WORK

The problem of designing database classes for applications having multiple user views appears to be an artifact of the primitive model of classification implicit in existing data modeling methods

ISSN: 2324-9773

and design techniques. Specifically, those techniques assume that a domain should be modeled globally by a single classification schema. This view is inconsistent with conclusions from classification research that classes do not exist independent of human perception, but instead capture useful abstractions of the similarity of things. These conclusions point to the need for a data model which provides support for diverse user views without relying on a fixed class schema.

Our research has focused first on defining a formal data model, MIMIC, which is novel in separating the representation of data about objects from the classification of those objects. Unlike existing data models, MIMIC provides inherent support for diverse user views and allows classes to be modified without affecting the underlying data.

In addition, we are developing a bottom-up data modeling technique based on the identification of objects and properties, as opposed to the traditional approach of defining the (integrated)class schema first. We believe such an approach corresponds better with the function of classification as a way to organize information about things. The data modeling technique will be evaluated with respect to a popular technique, such as the entity-relationship model. The evaluation will be performed through a series of experiments in which the modeling technique serves as the independent variable, and criteria such as ease of use and comprehensibility as dependent variables.

MIMIC is also being used as a basis for specifying implementation primitives for instance-based modeling. Currently, a prototype implementation has been developed in which the notions of object, property, class, and class structure are supported directly (Parsons and Leung, 1994). The intent of this prototype is to serve as a vehicle for empirically testing the predicted gains in flexibility of MIMIC over schema-based implementationenvironments, such as relational databases.

Finally, we believe that rather than simply borrowing from advances in classification research, the application to database design can advance classification research. In particular, we believe MIMIC provides a hybrid view of classification which can accommodate a world where classes are constructed, rather than existing independent of human perception. At the same time, the scope of typical database applications necessitates a model in which class boundaries and membership conditions can be precisely specified. MIMIC allows such specifications to exist within the context of a class structure or view, while allowing different people to define classes differently or view the same world of instances and properties in terms of different classes. We believe this model can account for many of the experimental results that caused the classical view of concepts to fall out of favor, and offer the model as an alternative to prototype and exemplar views that dominate research today (Smith, 1989). This sets the stage for experimental research to validate or refute the model as a mechanism to account for various important aspects of human classification.

7. ACKNOWLEDGEMENT

This research was supported in part by a grant from the Social Sciences and Humanities Research Council of Canada.

Parsons, J. (1994). On the Relevance of Classification Theory to Database Design. 5th ASIS SIG/CR Classification Research Workshop, 129-138. doi:10.7152/acro.v5i1.13782

8. REFERENCES

Batini, C., M. Lenzerini and S. Navathe, "A Comparative Analysis of Methodologies for Database Schema Integration", *ACM Computing Surveys*, 18(4), December 1986, 323-364.

Banerjee, J., H.-T. Chou, J. Garza, W. Kim, D. Woelk, N. Ballou, and H.-J. Kim, "Data Model Issues for Object-oriented Applications", *ACM Transactions on Office Information Systems*, 5(1), January 1987, 3-26.

Cattell, R., *Object Data Management*, Reading, MA: Addision-Wesley, 1994.

Gibbs, S., D. Tsichritzis, E. Casais, O. Nierstrasz and X. Pintado, "Class Management for Software Communities", *Communications of the ACM*, 33(9), September 1990, 90-103.

Kent, W., "The Breakdown of the Information Model in Multi-Database Systems", *SIGMOD Record*, 20(4), December 1991, 10-15.

Lakoff, G., *Women, Fire and Dangerous Things: What Categories Reveal About the Mind*, Chicago: University of Chicago Press, 1987.

Lerner, B. and N. Habermann, "Beyond Schema Evolution to Database Reorganization", in E*COOP/OOPSLA'90 Conference Proceedings, SIGPLAN Notices,* 25(10), October 1990, 67-76.

Lieberherr, K. and C. Xiao, "Formal Foundations for Object-Oriented Data Modeling", *IEEE Transactions on Knowledge and Data Engineering*, 5(3), June 1993, 462-478.

Navathe, S., R. Elmasri and J. Larson, "Integrating User Views in Database Design", *IEEE Computer,* June 1986, 50-62.

Parsons, J., "An Information Systems Model Based on Classification Theory", Working Paper, Faculty of Business Administration, Memorial University of Newfoundland, 1994a.

Parsons, J., "On an Appropriate Role for Classification in Data Modeling", Working Paper, Faculty of Business Administration, Memorial University of Newfoundland, 1994b.

Parsons, J. and S. Leung, "Implementation Primitives for Instance-based Data Modeling", Working Paper, Faculty of Business Administration, Memorial University of Newfoundland, 1994.

Rosch, E., "Principles of Categorization", in E. Rosch and B. Lloyd (eds.), *Cognition and Categorization,* Hillsdale, NJ: Erlbaum, 1978, 27-48.

Sheth, A. and J. Larson, "Federated Database Systems for Managing Distributed, Heterogeneous, and Autonomous Databases", *ACM Computing Surveys,* 22(3), September 1990, 184-236.

Smith, E., "Concepts and Thoughts", in R. Sternberg and E. Smith (eds.), *The Psychology of Human Thought*, Cambridge, MA: Cambridge University Press, 1988.

Smith, E., "Concepts and Induction", in M. Posner (ed.), *Foundations of Cognitive Science,* Cambridge, MA: MIT Press, 1989, 501-526.

Smith, E. and D. Medin, *Categories and Concepts*, Cambridge, MA: Harvard University Press, 1981.

Stamper, R., "Semantics", Chapter 2 in R. Boland and R. Hirschheim (eds.), *Critical Issues in Information Systems Research*, John Wiley and Sons, 1987, 43-78.

Stepp, R. and R. Michalski, "Conceptual Clustering: Inventing Goal-Oriented Classification of Structured Objects", in R. Michalski (ed.), *Machine Learning: An Artificial Intelligence Approach,* Vol. 2, San Franciso: Morgan-Kaufmann, 1986, 471-498.

Teorey, T., *Database Modeling and Design* (2nd edition), San Francisco: Morgan-Kaufmann, 1994.

ISSN: 2324-9773