Modeling Quality Requirements in Conceptual Database Design

Veda C. Storey
Georgia State University
College of Business Administration
Dept. of Computer Information Systems
P.O. Box 4015 Atlanta, Georgia 30302
Tel: (404) 651-3894
Fax: (404) 651-3842
E-mail: vstorey@gsu.edu

Richard Y. Wang
Massachusetts Institute of Technology
Sloan School of Management
E53-317, 50 Memorial Dr.
Cambridge, Massachusetts, 02139
Tel: (617) 253-0442
Fax: (617) 253-3321
E-mail: rwang@mit.edu

Key Words: Data Quality, Data Modeling, Application Requirements, Application Quality Requirements, Data Quality Requirements

ABSTRACT The quality aspects of data in a database are important from the user's perspective. However, they have largely been overlooked in the database design literature. This research presents an approach to incorporating quality requirements into the design process and representing them in a conceptual model. Distinctions are made among *application requirements, application quality requirements,* and *data quality requirements.* The implications of incorporating these requirements into database design are analyzed, and the research results applied to a detailed example.

1. INTRODUCTION

We currently have great capabilities for capturing and organizing large volumes of data, with virtually all commercial information being digitally encoded (Fayyad et al., 1996). Today's managers operate within a global environment and are increasingly called upon to make major decisions in shrinking time frames. They are also expected to process large amounts of both internal and external data. Data upon which a manager bases his or her decision that is inaccurate, out-of-date, or has other types of quality problems can lead to decisions that are suboptimal or even detrimental to the welfare of the organization.

The quality aspects of data in a database management system (DBMS) have been treated through various techniques such as concurrency, recovery, integrity, and security control [Date, 1990]. Although these techniques are necessary to ensure the

correct operation of a DBMS, they were not developed specifically to address issues related to the quality aspects that are important to the user.

The need to incorporate quality requirements into a database is well-recognized as is the impact of data quality problems [Ballou & Pazer, 1995; Laudon, 1986; Bruce, 1991; Wang, Kon & Madnick, 1993; Wang, Storey & Firth, 1995; Zahedi, 1995; Redman, 1992]. Unfortunately, quality requirements have been overlooked as an integral part of the database design process. The objective of this research, therefore, is to:

> *develop an approach to incorporating quality requirements into the database design process, focusing particularly on how they will be represented in a conceptual model.*

The procedure for doing so involves differentiating *application quality requirements* and *data quality requirements* from *application requirements,* and then developing the constructs and procedures necessary to represent the data quality requirements in an entity-relationship model. The translation of these requirements into a relational model is also examined.

1.1 Data Quality Dimensions

Data quality is widely recognized as having multiple dimensions [Ballou and Pazer, 1985; Ballou and Tayi, 1989; Wang and Strong, 1996; Wand and Wang, 1996]. Ballou and Pazer, for example, discuss the trade-offs involved in timeliness versus accuracy. Based upon an examination of possible failure, Wand and Wang [1996] identify four dimensions of data quality: 1) completeness, 2) lack of ambiguity, 3) meaningfulness, and 4) correctness. Wang and Strong propose a framework for data quality dimensions that is based on more than 150 data quality requirements. Thus, data quality means different things to different users. The challenge in this research, therefore, is to devleop a mechanism for all users to capture the data quality dimensions that are appropriate for their applications.

1.2 Motivating Example

Consider a training database, the purpose of which is to keep track of employee training for career development purposes. Data is needed on courses and the vendors who supply them, students who are enrolled and those who are wait-listed to take the classes, jobs skills, evaluations of courses, proficiency levels and other information.1 A course can be offered by different vendors as shown in the entity-relationship model [Chen, 1976] in Figure 1. The min/max cardinalities [Tsichritzis and Lochovsky, 1982] indicate that, for each *Course*, there can be zero to many vendors who offer the course and one to many courses offered by each vendor. This conceptual model represents the traditional approach to database design, but does not capture quality requirements such as the reputation of the vendor or the standards for the course.

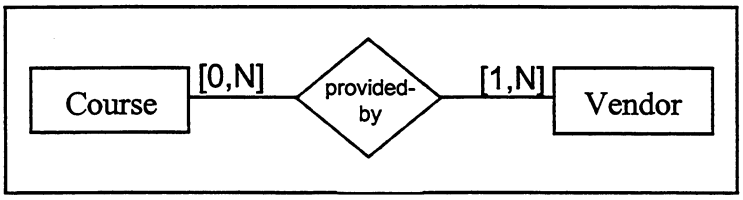

Figure 1: A training database example

Entities:
Course: [CourseId, base cost, recertification period, description]
Vendor: [VendorId, phone]
Relationship:
Course provided-by Vendor
[0,N] [1,N]

1.3 Overview of the Paper

This paper is divided into six sections. Section 2 presents a quality design example and illustrates its usefulness. The research foundations are presented in

1 The examples discussed in this paper are adapted from a human resources system, *Ingenium*, a product of Meliora Systems Inc.

Section 3. Section 4 defines the constructs needed to incorporate data quality requirements into a conceptual design and applies them to an application in Section 5. Section 6 summarizes and concludes the paper.

2. QUALITY DESIGN EXAMPLE

An example of a relational database that incorporates data quality is shown in Figure 2. This is followed by samples of populated tables in Figure 3, and representative queries to indicate the value of this design. The analysis that lead to the development of this model is given in the following sections.

Entity Relations:
- Course:[CourseId,base cost, recertification period, description]
- Standard: [Std-Name,description]
- Standard Rating: [Std-Name, Value, interpretation]
- DQ Dimension: [D-Name, Rating]
- DQ Measure: [Rating, description]
- Instructor: [InstructorId, phone]

Extended Entity Relations:
- Course-Standard: [CourseID, Std-Name, value]
- Class-Attendance-DQ-Dimension: [ClassId, D-Name, Rating]
- Class: [ClassId, attendance, actual cost, courseId]

Relationship Relation:
- Instructor-teaches-Class: [InstructorId, ClassId]

Figure 2: Relational model for course database that incorporates data quality

Course Relation

CourseId	base cost	recertification period	description
MIS 401	300	1 year	Introduction to Access
MIS 412	250	1.5 years	Advanced Excel
MIS 500	350	1 year	Novell Networks

Standard Relation

Std-Name	description
MS-Cert	Microsoft certified course
Novell-Cert	Novell certified course

Standard Rating Relation

Std-Name	Value	interpretation
MS-Cert	yes	Microsoft certification courses passed by instructor
MS-Cert	no	Microsoft certification courses not passed by instructor
Novell-Cert	yes	Novell certification courses passed by instructor
Novell-Cert	no	Novell certification courses not passed by instructor

DQ Dimension Relation

D-Name	Rating
Accuracy	1
Accuracy	2
Timeliness	no

DQ Measure Relation

Rating	description
1	excellent
2	acceptable
3	poor
yes	up-to-date
no	not up-to-date

Instructor Relation

InstructorId	phone
Ramsey	716-277-5448
MacKenzie	716-266-9944

Course-Standard Relation

CourseId	Std-Name	value
MIS 401	MS-Cert	yes
MIS 412	MS-Cert	no
MIS 500	Novell-Cert	yes

Class-Attendance-DQ-Dimension Relation

ClassId	D-Name	Rating
21	Accuracy	2
35	Timeliness	no

Class Relation

ClassId	attendance	actual cost	courseId
21	40	$300	MIS 401
35	35	$290	MIS 500
55	27	$198	MIS 412

Instructor-teaches-Class Relation

InstructorId	ClassId
Ramsey	21
MacKenzie	35

Figure 3: Sample data for quality relational model

The following queries illustrate the types of information that can be obtained from the quality design.

- Is MIS 401 (Introduction to Access) a "MS-Cert" course? This can be obtained from the relation *Course-Standard* where CourseID = 'MIS 401', Std-Name = 'MS-cert', and "value" = 'yes' .

- What does a value of `yes' for `MS-cert' mean? This is obtained from the relation Standard Rating where "Std-Name" = `MS-Cert' and "Value"=`yes'. Then, the interpretation = `Microsoft certification courses passed by instructor' can be retrieved.

- What is the accuracy of "attendance" for Class 21 and what does its value mean? The relation *Class-Attendance-DQ-Dimension* is selected for ClassId = `21' and D-Name=`accuracy'; a rating of `2' is obtained. This relation is joined with the *DQ-Measure* relation on the attribute "rating" . For a value of `2', the description `acceptable' is retrieved.

3. FOUNDATIONS FOR QUALITY REQUIREMENTS ANALYSIS

Traditional user requirements as well as user quality requirements must be represented in the database design. To do so, user requirements are divided into three categories: 1) application requirements, 2) application quality requirements, and 3) data quality requirements.

- **Application Requirements:** Widely accepted and practiced, traditional application requirements analysis identifies data items such as employees and their ranks and salaries, or departments and their names and locations, that are fundamental to a

database application [Teorey, 1990]. These are the requirements that are usually discussed and represented when designing a database.

- **Application Quality Requirements:** Application quality requirements may arise at the input, process, and output stages associated with the production of a product, [Hauser & Clausing, 1988; ISO, 1992; Juran, 1992]. Such requirements include measurements against standards and comparisons of final products to initial design specifications. Traditionally, these have not been recognized or captured during the design process.

- **Data Quality Requirements:** Data quality requirements correspond to the quality of the actual data in the database. They can be associated with either application requirements or application quality requirements. As previously mentioned, these requirements are multi-dimentsional in nature.

In the training example, *Course* and *Instructor* and their relationships are application requirements. One may want to know what standards a course has. This is an application quality requirement. An example of a data quality requirement is the accuracy of the class attendance data.

There are various reasons why the three types of requirements are separated. First, quality requirements (both application quality and data quality requirements) differ from application requirements. Since they are often overlooked, this forces the user to identify them specifically during requirements analysis. Second, most existing systems were not built with quality requirements in mind. Therefore, the requirements can be elicited separately and added without requiring a completely new design. This is important for legacy systems. Finally, there is subjectivity in deciding what quality items to include and what measurement scales to use.

3.1 Quality Requirements Identification

The identification of quality requirements must begin at the requirements analysis phase because quality requirements are both application and user-dependent. During the identification of application requirements, the designer collects requirements and expresses them in English sentences [Chen, 1993]. Application quality requirements and data quality requirements can be identified in a similar manner; that is, by the designer working with the user to elicit the requirements and possibly suggesting some quality requirements based upon the designer's experience with similar or related application domains.

Wang and Strong [1996] provide a framework that categorizes data quality into four main categories: 1) intrinsic data quality (believability, accuracy, objectivity, reputation), 2) contextual data quality (value-added, relevancy, timeliness, completeness, appropriate amount of data), 3), representation data quality (interpretability, ease of understanding, representational consistency, concise representation), and 4) accessibility data quality (accessibility, access security). A designer, possibly with the assistance of an automated design tool, could use this framework during requirements elicitation as a check-list of data quality items, drilling down as appropriate to the underlying elements. Alternatively, the system could use them to try to verify that the user had chosen appropriate data quality measures.

3.2 Requirements Modeling

The quality design process is summarized in Figure 4. Since the identification of traditional application requirements is well understood, it is the logical foundation upon which to analyze application quality requirements. The acquisition of quality requirements is shown as Steps 1 and 2. Since both application requirements and application quality requirements will ultimately be captured and stored as data in a database management system, a mechanism must be available to measure the quality of

the data that are stored to represent these two types of requirements. This is the task of Step 3: identifying data quality requirements.

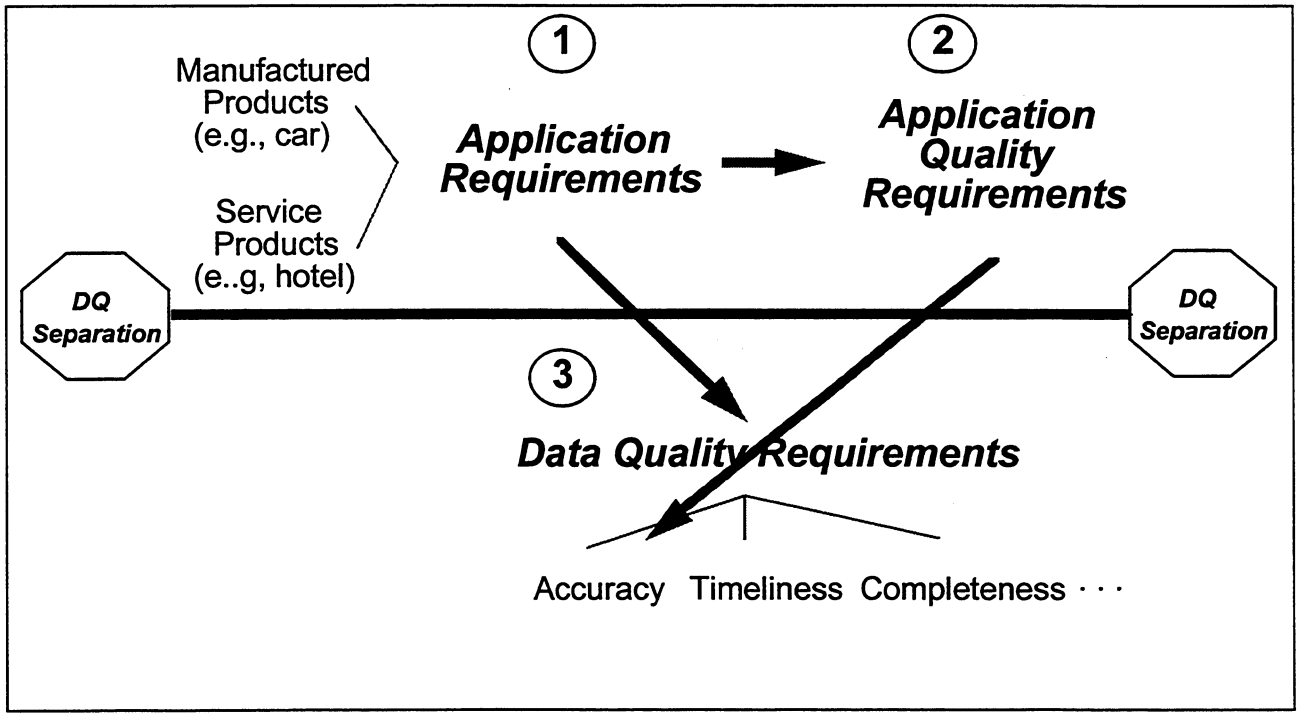

Figure 4: Quality Requirements Modeling Process

The arrow from Step 1 to Step 2 signifies that application requirements identification precedes application quality requirements identification. The arrows from Steps 1-2 to Step 3 indicate that application and application quality requirements are modeled before data quality requirements. As in conventional database design and software engineering practices, these steps may be iterated.

The distinctions among application requirements, application quality requirements, and data quality requirements are important at both the requirements analysis and conceptual design stages. Both application requirements and application quality requirements are identified to represent a real-world system [Kent, 1978; Wand & Weber, 1993]. They are modeled as entities, relationships, and attributes. Data quality requirements are identified to measure the quality of the ***attribute values*** (data)

that are stored in a database that represents a real-world system. Therefore, new constructs are needed to model data quality requirements in a conceptual design.

It might be argued that the distinctions made among application requirements, application quality requirements, and data quality requirements are not necessary because a view mechanism can be applied to provide separate views to the user for application-oriented data, application quality-oriented data, and data quality-oriented data. We reject this argument outright. In order to make use of the view mechanism to provide separate views, an underlying schema containing all the data items that represent application requirements, application quality requirements, and data quality requirements is needed. However, developing such a schema is exactly why the distinctions are made in the first place. On the other hand, once a conceptual schema that incorporates application, application quality, and data quality requirements is obtained and the corresponding database system developed, then view mechanisms can be applied to restrict views that correspond to each of these types of requirements. This leads to the following principle:

> **•The Data Quality (DQ) Separation Principle: Data quality requirements are modeled separately from application requirements and application quality requirements.**

4. MODELING DATA QUALITY REQUIREMENTS IN CONCEPTUAL DESIGN

Underlying the *Data Quality Separation Principle* is the need to measure the quality of attribute values. One might try to model the quality of an attribute value as another attribute for the same entity. This would have undesirable results such as violation of normalization principles in the relational model (e.g., "address quality" is dependent upon "address" which is dependent upon the "person-id"). Another approach is to model the quality of an attribute as a meta-attribute [Wang, Kon & Madnick, 1993; Wang, Reddy & Kon, 1995]. However, no constructs in the conceptual

design level are explicitly defined for capturing meta-attributes. We, therefore, introduce and define two types of data quality entities: 1) a *Data Quality Dimension* entity and 2) a *Data Quality Measure* entity. To model the quality of attribute values, an *attribute gerund representation* is needed.

4.1 Data Quality Dimension Entity

A user needs to be able to capture different data quality dimensions for some (or all) attributes of an entity; for example, the accuracy and timeliness of the attribute ``attendance" or the timeliness of the "recertification period" (is it up-to-date?). One representation is desired, but it needs to be flexible enough to represent different data quality dimensions for different attribute values. A *Data Quality Dimension* entity is introduced for this purpose. Its primary key is the combination of Dimension Name (D-Name) which can take on values such as 'timeliness' and `completeness' and Rating which stores the corresponding value that is obtained on that dimension.

E.g.:

Data Quality Dimension: [Dimension-Name, Rating]

[Accuracy, 1]

[Accuracy,2]

[Timeliness, yes]

The values that appear in the populated entity are the set of values that exist in the database and will usually be a subset of all possible combinations. They represent the actual dimensions upon which an attribute's data quality is assessed and the values obtained on those dimensions. Both attributes are part of the key because it allows us to capture (using only one entity) for example, that the attribute "actual cost" has an accuracy of `1' whereas the accuracy of "attendance" is `2'.

This representation assumes that the accuracy for all attributes is rated on the same scale. Of course, if, for example, the accuracy scale can differ depending upon the attribute, then this can be accommodated by adding the attribute name to the key.

E.g.:

Data Quality Dimension: [Dimension-Name, Attribute, Rating]

[accuracy, class attendance, 1]

[accuracy, actual cost, good]

4.2 Data Quality Measure Entity

To completely capture the quality aspects, it is necessary to store information on the values that are assigned to a data quality dimension, e.g., the interpretation of "rating". A *Data Quality Measure* entity is introduced to capture the interpretation of the quality attribute values. It consists, for example, of a primary key, "Rating", that could have values '1', '2', and '3', and a textual non-key attribute, "description" which stores a description of what the rating values mean (e.g., 1 is excellent, 2 is acceptable, 3 is poor). The *Data Quality Measure* entity enables different data quality dimensions with heterogeneous measurement scales to be modeled (e.g., *accuracy* is measured between `1' and `3', and *timeliness* measured as 'yes,' or 'no').

E.g.:

Data Quality Measure: [Rating, description]

[1, excellent] (for accuracy)

[yes, up-to-date] (for timeliness)

There is another reason for introducing this separate Data Quality Measure entity. We want to be able to capture *all* of the descriptions of "rating" whether or not they currently appear in the database. If "description" were a non-key attribute of the Data

Quality Dimension entity, then only those that are instantiated would appear. The purpose of the Data Quality Measure entity is, in essence, for its corresponding relation to serve as a complete "look-up" table. Furthermore, including "description" in the the Data Quality Dimension entity would lead to a second normal form violation in the relational model because "description" would depend on only part of the key ("rating").

Analogous to the above, the Data Quality Measure entity could be expanded to model different scales for different attributes for the same data quality measure. For example, suppose the accuracy rating for "attendance" has a different interpretation than the accuracy for "base cost", then the key would be expanded to include both the dimension and attribute names:

Data Quality Measure: [Dimension-Name, Attribute, Rating, description]

[accuracy, attendance, 1, excellent]

[accuracy, base cost, 90%, 90% correct]

[timeliness, attendance, yes, up-to-date]

4.3 Attribute Gerund Representation

As shown in Figure 5, the application entity *Class* has "attendance" as an attribute for which the user might be interested in its data quality ratings. In the entity-relationship model there is no direct mechanism to associate an attribute ("attendance") of one entity (*Class)* with another entity (e.g., a *Data Quality Dimension* entity or even a *Data Quality Measure* entity). Furthermore, "attendance" is an application attribute and by the *DQ Separation Principle*, an application requirement must remain part of the application specification, and not become an entity or attribute below it. This problem is resolved using the *Attribute Gerund Representation.* First the relationship, *Class attendance-data-has Data Quality Dimension*, is created that includes "attendance" as part

of its name. This data quality relationship can be represented as a gerund entity [Chen, 1993]. Then, a relationship between the gerund and *Data Quality Measure* entity provides a mechanism for retrieving the descriptions of the data quality dimension values.

An alternative representation would be to make "rating" an attribute of the relationship *Class attendance-data-has Data Quality Dimensions*. This representation, however, does not capture additional information on rating such as descriptive information on its value. It would only be appropriate if the interpretation of rating is obvious, but this is not usually the case. Further analysis of this and alternative representations is provided in the appendix. In general, the Attribute Gerund Representation can be applied when an application entity attribute needs to be associated directly with a data quality entity that has various quality categories, each of which has a corresponding Rating and "description" (or other information).

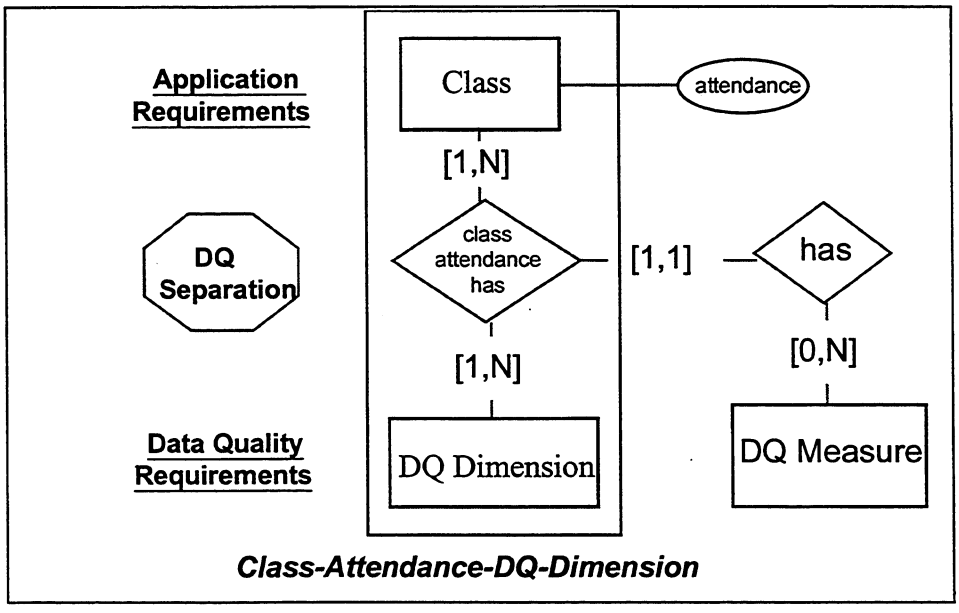

Figure 5: Attribute Gerund Representation

5. APPLICATION EXAMPLE

This section extends the training database example to incorporate information on course standards along with other quality requirements. The impact on both the conceptual and logical designs is discussed.

5.1 Conceptual Design

A user might want to know what standards (an application quality entity) a course (an application entity) might have. This can be modeled by the relationship, *Course has Standard,* as shown in Figure 6. The application quality entity *Standard Rating* is created with primary key [Std-Name, Value] and a textual non-key attribute "interpretation."

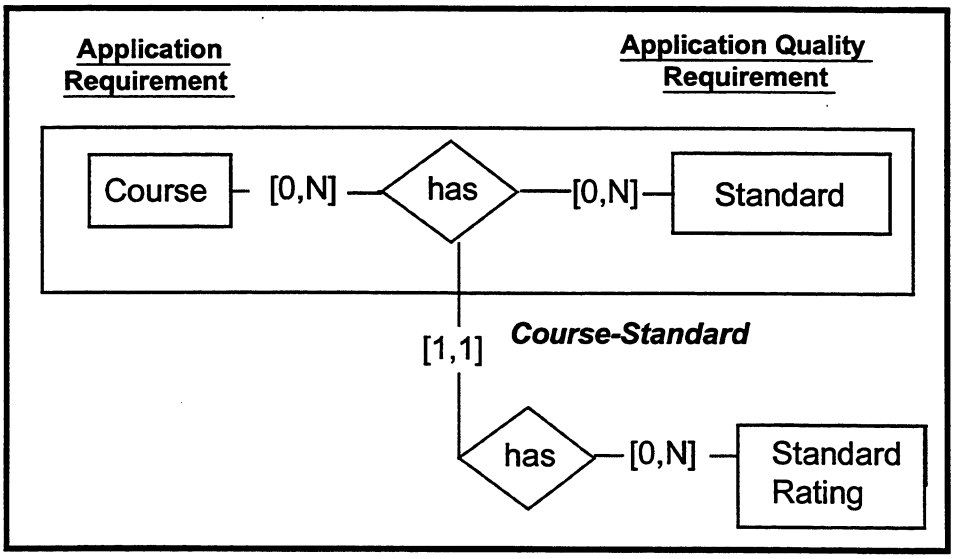

Figure 6: Course Standards

Each occurrence of the relationship *Course has Standard* has a standard rating value. However, in the entity-relationship model, an entity can only be associated with another entity. Therefore, this relationship must first be converted to an entity gerund [Chen, 1993], called *Course-Standard.* Since *Course has Standard* is a many-to-many relationship, the primary key of the gerund is the concatenation of the keys of *Course* and *Standard.* Then, a new relationship is created between this gerund and *Standard-*

Rating; that is, *Course-Standard has Standard-Rating*. The final entity-relationship model is shown in Figure 7 with the steps involved in its development outlined in Figure 8. In the first step, identify user requirements, the user, with assistance from the designer, would be asked to identify not only the entities, attributes and relationships of interest, but also the data quality aspects that need to be captured. Note, for example, that, analogous to the "attendance" attribute of Class, the "value" attribute of *Standard Rating* could have a data quality dimension. The attribute gerund representation would again be used. First the relationship, *Standard Rating value-data-has DQ Dimension* would be created. This would be represented by the entity gerund *Standard-Rating-Value-DQ-Dimension* which, in turn, would be associated with the *DQ Measure* quality entity.

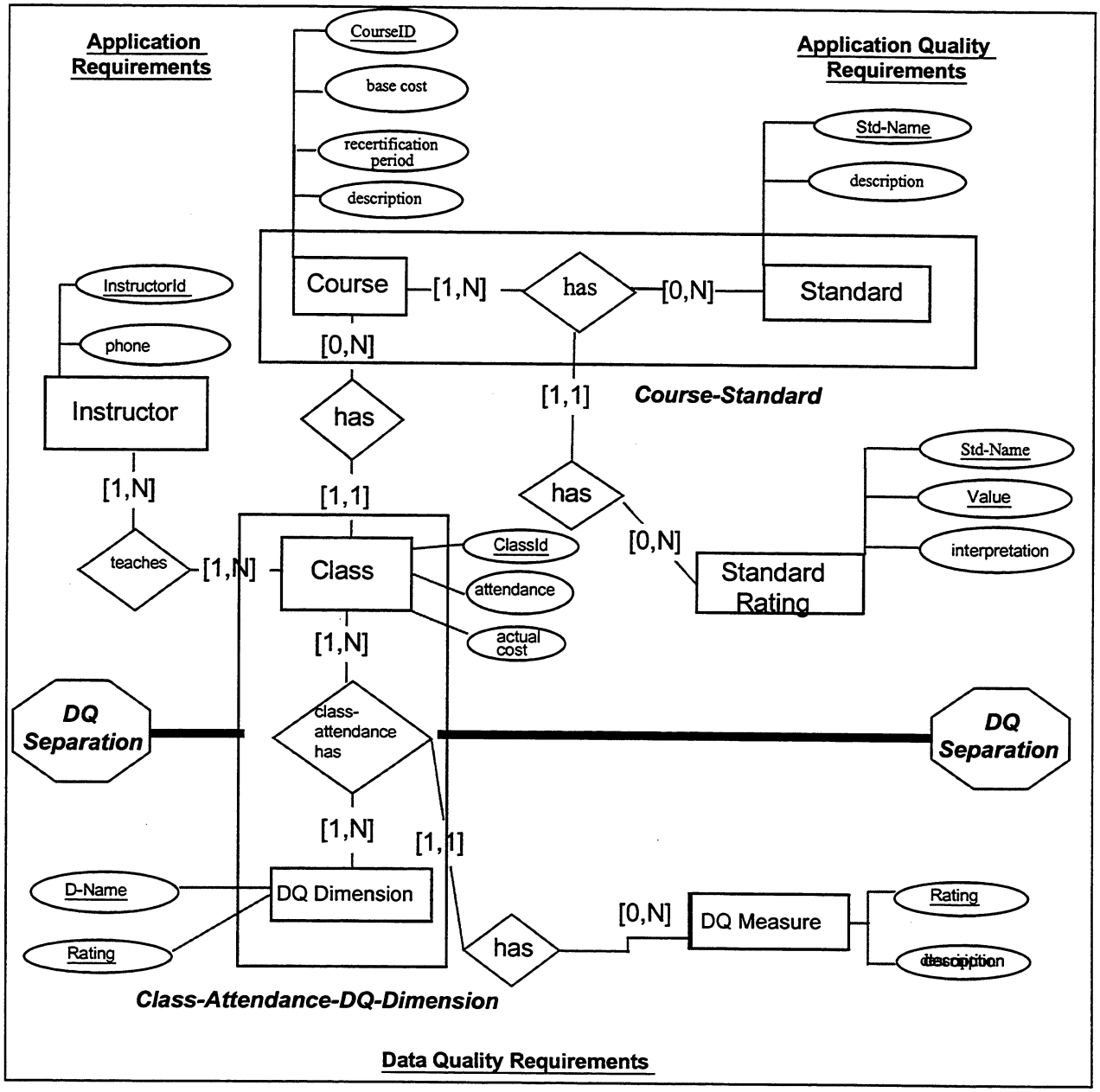

Figure 7: An extended conceptual design of the course database

Step 1: Identify user requirements

- What courses are available, their certification information and cost.
- What classes are offered for a course.
- Accuracy and timeliness measures of class attendance, along with a description of what the accuracy and timeliness values mean. (Data quality requirements)
- The instructor of a class.
- What standards are required for a course and a course's rating on them. (Application quality requirements)

Step 2: Identify application entities:

Course: [CourseId, base cost, recertification period, description]
Class: [ClassId, attendance, actual cost]
Instructor: [InstructorId, phone]

Step 3: Identify corresponding application quality entities:

Standard: [Std-Name, description]
Standard Rating: [Std-Name, Value, interpretation]

Step 4: Identify data quality entities:

DQ Dimension: [D-Name, Rating]
DQ Measure: [Rating, description]

Step 5: Associate application entities with application quality entities:

Course-Standard: [CourseID, Std-Name]

Step 6: Associate application attributes with data quality entities:

Class-Attendance-DQ-Dimension: [ClassID, D-Name]

Step 7: List the application relationships:

Course has Classes
[0,N] [1,1]
Instructor teaches Class
[1,N] [1,N]

Step 8: List the application quality relationships:

Course has Standards
[1,N] [0,N]
Course-Standard has Standard Rating
[1,1] [0,N]

Step 9: List the data quality relationships:

Class attendance-data-has DQ Dimension
[1,N] [1,N]
Class-Attendance-DQ-Dimension has DQ Measure
[1,1] [0,N]

Figure 8: Conceptual design for the course database

5.2 Logical Design

The conceptual design can now be translated into a relational model. Every entity becomes a separate *entity relation*. A relationship is represented either by a *foreign*

key (for a one-to-many relationship) or a separate *relationship relation* (for a many-to-many relationship) [Teorey, Yang, and Fry, 1986].

The relationship from which a gerund entity is created is inherently represented. This is because the relationship is first converted to an entity. In the case of a many-to-many relationship, the primary key of the gerund entity is the concatenation of the keys of the involved entities, which is exactly what the primary key of the corresponding relationship relation would be. The entity is then represented by an entity relation in the relational model. Both the entity relation and relationship relation would be exactly the same: they would include the concatenation of the two primary keys and there would be no non-key attributes. Of course, a foreign key may be added to this relation to represent another relationship as is the case when representing the relationship *Course-Standard has Standard Rating.* The resulting relational model was shown in Figure 2.

5.3 Performance Considerations

There are several performance factors to be considered:

- There is additional work on the part of both the user and the designer to elicit the quality aspects of an application.
- Additional relations are needed to represent the application quality requirements. There is extra overhead associated with setting up and maintaining these relations.
- There is currently no way to match the name of a relationship to a list of data quality attributes.
- Procedures need to be developed for updating quality attributes, just like any others.
- DBMS overhead is needed for installing entity integrity constraints and referential integrity constraints (foreign key references to DQ-dimension relations as well as to the base entity relations) for each attribute that has quality requirements.

Obviously, there are tradeoffs that need to be assessed. For many of these, the improved quality of the data should more than offset the extra processing complexity. Ultimately, the increase in the confidence that the decision-maker has in the data should offset these concerns.

6. CONCLUDING REMARKS

This paper has developed an approach to representing data quality requirements at the logical database design stage. Data quality requirements are divided into *application quality requirements* and *data quality requirements*. A procedure was devloped for incorporating them into an E-R model, which maintains a separation with application quality requirements. A detailed example was presented which illustrated how quality requirements can be incorporated into a conceptual design and translated into a relational schema. Several new research areas could be further explored: (1) a query processing mechanism needs to be developed that will accommodate the new constructs developed in this research; (2) an analysis of the corresponding processes is necessary to ensure that the Data Quality Separation Principle is followed and data quality assured, given a conceptual design that incorporates quality requirements; (3) field studies of the approach in industrial applications should be undertaken to further extend and validate the research results; and (4) an object-oriented approach to modeling quality requirements could be developed.

7. REFERENCES

[1] Ballou, D.P. & Pazer, H.L. (1985). Modeling Data and Process Quality in Multi-input, Multi-output Information Systems. *Management Science*, 31(2), pp.150-162.

[2] Ballou, D.P. & Pazer , H.L. (1995). Designing Information Systems to Optimize the Accuracy Timeliness Tradeoff. *Information Systems Research*, 6(1), March 1995, pp.51-72.

[3] Ballou, D.P., and Tayi, K.G. (1989) Methodology for Allocating Resources for Data Quality Enhancement, *Communications of the ACM*, 32(3), 1989, pp.320-329.

[4] Bruce, T.A. (1991). *Designing Quality Databases with IDEF1X Information Models.* New York: Dorset House Publishing.

[5] Chen, P. (1976). The Entity-Relationship Model - Toward a Unified View of Data. *ACM Transactions on Database Systems*, 1, pp.166-193.

[6] Chen, P. (1993). The Entity-Relationship Approach. In *Information Technology in Action: Trends and Perspectives.* (pp. 13-36). Englewood Cliffs: Prentice Hall.

[7] Date, C.J. (1990), *An Introduction to Database Systems*, Reading: Addison-Wesley.

[8] Fayyad, U., Piaetsky-Shapiro, and Smyth, P., "The KDD Process for Extracting Useful Knowledge from Volumes of Data", *Communications of the ACM*, Vol, 39, No. 11, Novemeber, 1996, pp. 27-34.

[9] Hauser, J.D. & Clausing, D. (1988). The House of Quality. *Harvard Business Review*, 66(3), pp.63-73.

[10] ISO (1992). *ISO9000 International Standards for Quality Management.* (No. International Standard Organization, Geneva.

[11] Juran, J.M. (1992). *Juran on Quality by Design: The New Steps for Planning Quality into Goods and Services.* New York: Free Press.

[12] Kent, W. (1978). *Data and Reality.* New York: North Holland.

[13] Laudon, K.C. (1986). Data Quality and Due Process in Large Interorganizational Record Systems. *Communications of the ACM*, 29(1), pp.4-11.

[14] Redman, T. (1992). Data Quality: Management and Technology. New York: Bantam Books.

[15] Teorey, T.K., Yang, D., & Fry, J.P. (1986). A Logical Design Methodology for Relational Database Using the Extended Entity-Relationship Model. *Computing Surveys*, 18(2), pp.197-222.

[16] Teorey, T.K. (1990). Database Modeling and Design: The Entity-Relationship Approach. San Mateo, CA: Morgan Kaufman Publisher.

[17] Tsichritzis, D. & Lochovsky, F. (1982). *Data Models.* Englewood Cliffs, N.J.: Prentice Hall.

[18] Wand, Y. & Wang, R. (1996). Anchoring Data Quality Dimensions in Ontological Foundations,. *Communications of ACM*, 39(11), pp.86-94.

[19] Wand, Y. & Weber, R. (1993). On the Ontological Expressiveness of Information Systems Analysis and Design Grammars. *Journal of Information Systems*, 3(3), pp.217-237.

[20] Wang, R, Kon, H., & Madnick, S. (1993). Data Quality Requirements Analysis and Modeling. In *the 9th International Conference on Data Engineering*, (pp. 670-677) Vienna: IEEE Computer Society Press.

[21] Wang, R., Reddy, M.P. & Kon, H. (1995). Toward quality data: An attribute-based approach. *Decision Support Systems (DSS)*, 13, pp.349-372.

[22] Wang, R.., Storey V.C., and Firth, C. (1995). A Framework for Analysis of Data Quality Research. *IEEE Transactions on Knowledge and Data Engineering*, 7(4), August, pp.623-640.

[23] Wang, R. and Strong, D. (1996), Beyond Accuracy: What Data Quality Means to Data Consumers, *Journal of Management Information Systems*, 4, pp.5-34.

[24] Zahedi, F. (1995). *Quality Information Systems.* Danvers, Mass: Boyd & Fraser Publishing Company.

8. APPENDIX: ATTRIBUTE GERUND & ALTERNATIVE REPRESENTATIONS

Consider the application entity *Class* for which we want to associate "attendance" with its data quality entity *DQ Dimension* and obtain a description of its rating. *Class attendance-data-has DQ Dimension* is a many-to-many relationship.

8.1 Case 1: Attribute Gerund Representation

To associate the "attendance" attribute of *Class* to its *DQ Dimension,* apply the attribute gerund representation to produce the gerund *Class-Attendance-DQ-Dimension* and then associate the gerund with the data quality entity, *DQ Measure.* This is shown in Figure A1.

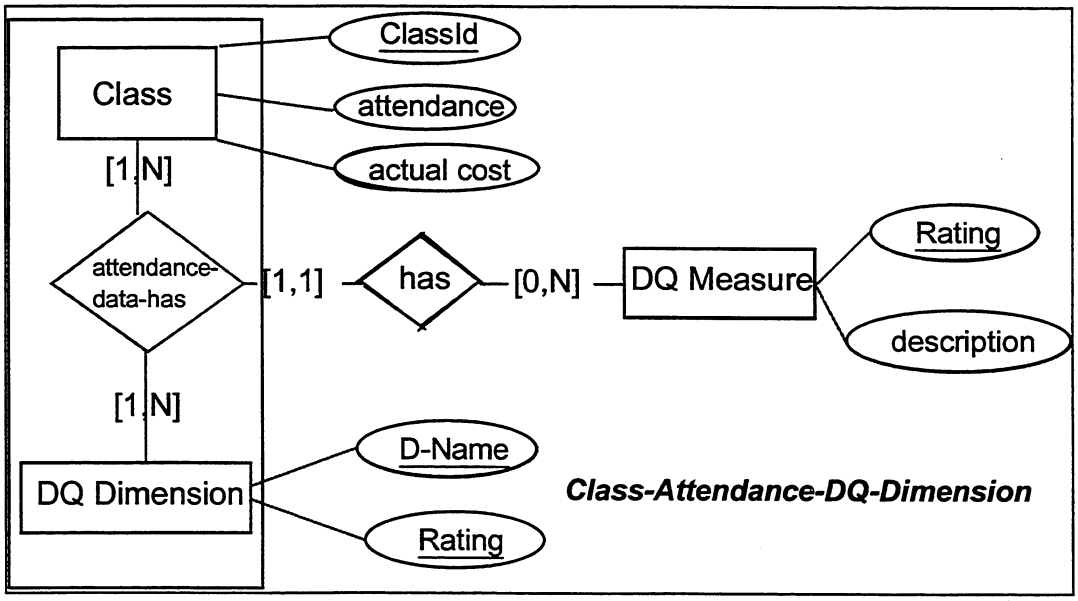

Figure A1: Attribute Gerund Representation

Class:	[ClassId, attendance, actual cost]
DQ Dimension:	[D-Name, Rating]
DQ Measure:	[Rating, description]
Class-Attendance-DQ-Dimension:	[ClassId, D-Name]

Class attendance-data-has DQ Dimension

[1,N] [1,N]

Class-Attendance-DQ-Dimension has DQ Measure

[1,1] [0,N]

8.2 Case 2: Ternary Relationship Representation

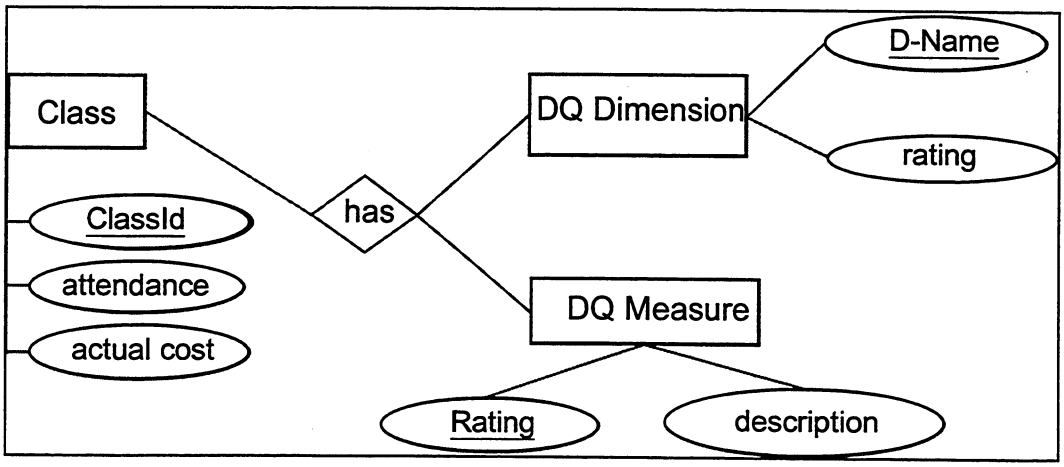

Figure A2: Ternary Relationship Representation

The association among *Class, DQ Dimension,* and *DQ Measure* could be represented as a ternary relationship as shown in Figure A2. The key of the corresponding relation would be the concatenation of the keys of *Class, DQ Dimension,* and *DQ Measure*. This is a poor design because it allows one to model such things as "class attendance has fair accuracy", "class attendance has poor accuracy", etc., simultaneously.

8.3 Case 3: Relationship Attribute Representation

Many-to-many relationships may have relationship attributes [Teorey et al., 1986] so "rating" might become an attribute of *Class attendance-data-has DQ Dimension* and the key of *DQ Dimension* reduced to "D-Name". The relation that represents the relationship *Class attandance-data-has DQ Dimension* would have "rating" as a nonkey

attribute. This would then need to be treated like a foreign key, when it is not, so that it could be joined to the relation that represents the data quality measure entity. Alternatively, it could not be treated as a foreign key, but then the representation would not capture the descriptive information that is needed for "rating". This would be practical only if the numerical values for "rating" are self-explanatory.

8.4 Case 4: Solo Entity Representation

DQ Measure could be treated as a separate entity with descriptive information as an attribute. It would be a solo entity (does not participate in any relationship) and appear in a relational model as a look-up table. This alternative does not make explicit the association between a given data quality dimension of a given entity and its *DQ Measure*.

8.5 Case 5: Non-key Attribute Representation

In the following representation each of the data quality dimension ratings is treated as a non-key attribute of *Class*.

Class: [ClassId, attendance, attendance-accuracy-rating, attendance-timeliness-rating, actual cost, actual cost-accuracy-rating]

This does not model descriptive information on the data quality dimension ratings.