From Entity-relationship Diagrams to Fourth Normal Form: A Pictorial Aid to Analysis

KATHRYN S. DAWSON AND LORRAINE M. PURGAILIS PARKER*
Department of Mathematical Sciences, Virginia Commonwealth University, Richmond, Virginia 23284–2014, U.S.A.

The analysis of a relational database to reduce its component schemes into fourth normal form is not trivial. Decomposition into Boyce–Codd normal form is not difficult, but the concepts and analysis needed to further decompose into fourth normal form are usually difficult to comprehend and use. A diagrammatic representation of dependencies is described which provides an informal aid to the construction of relational schemes in fourth normal form.

Received September 1986, revised February 1987

1. INTRODUCTION

Relational databases are becoming more and more widely used. Among the reasons for their popularity are the simple data structures and manipulation language for the user, and the foundation of mathematics upon which they are based. This mathematical foundation allows for the definition of the various types of data dependencies which can exist. It also gives us methods for removing the problems caused by these dependencies. Therefore, using the definitions and methodologies prescribed by normal form theory, it is possible to construct a database which will not cause unexpected problems when manipulated by a query language such as relational algebra.

In order to construct a database with this obviously desirable property, the database administrator must be able to detect all existing dependencies. Most often the functional dependencies and transitive dependencies which violate second, third and Boyce–Codd normal form are easily spotted. Rules for decomposition into Boyce–Codd normal are therefore the easiest to follow.

However, the multi-valued dependencies which violate fourth normal form are not so easy to detect. Therefore, many attempts to decompose into this form are not successful. However, once shown the result of decomposing a relational database into fourth normal form, it is easy to understand why the decomposition is useful. There is an intuitive understanding of the dependencies which is not easily reached via their mathematical definitions.

This paper intends to detail a method which aids in constructing relations and which makes use of the intuitive understanding of dependencies.

2. THE DIAGRAMMATIC APPROACH

2.1 The Entity-relationship Diagram

A good starting place in developing an understanding of the prospective contents of a database is the entity-relationship diagram.[1] This allows a pictorial view of the entities to be contained in the database, and the relationships between them.

For example, an entity-relationship diagram for a University department might look like Fig. 1. This example, which is used throughout the paper, was designed to aid in the process of scheduling faculty in the Mathematical Sciences Department at Virginia Commonwealth University.[2] The rectangular boxes contain entity set names. These entities are the real-world items which are to be represented in the database.

Each entity within a set has certain characteristics or attributes which uniquely describe it. Each attribute takes its value from a specified domain. The attribute domains are shown as bubbles joined to their entity sets. When constructing the entity-relationship diagram, care must be taken to associate all the attribute domains which are to be stored in the database with the correct entity sets. Thus the entity-relationship diagram shows all the data items which are to occur in the database. Every group of sets in the diagram must be considered to decide the relationships which exist between them. At this point the emphasis is on identifying the relationships, not on characterising their nature. Each relationship is denoted by a diamond and is connected to the entity sets it relates. The complete entity-relationship diagram provides a visual picture of what is to be stored in the database.

2.2 The Bubble Diagram

The bubble diagram is derived from the entity-relationship diagram, but contains more information. The major differences between the two diagrams are as follows.

Entity sets no longer exist explicitly in the bubble diagram. Just as entities are represented by their attributes within the database, so their attribute sets can represent them in the bubble diagram. The attribute sets from which values are chosen to uniquely identify an entity (i.e. the primary key of the entity set) are now used to show the relationships in which the entity set is involved.

The entity-relationship diagram only shows relationships between entity sets. This is expanded on in the bubble diagram to show relations and therefore dependencies between attribute sets. The bubble diagram corresponding to Fig. 1 is shown in Fig. 4.

2.3 Conversion of an Entity-relationship Diagram to a Bubble Diagram

2.3.1 Ensure relationships are binary

The rest of this paper assumes that a relationship is only defined between two entity sets. However, n-ary rela-

* To whom correspondence should be addressed.

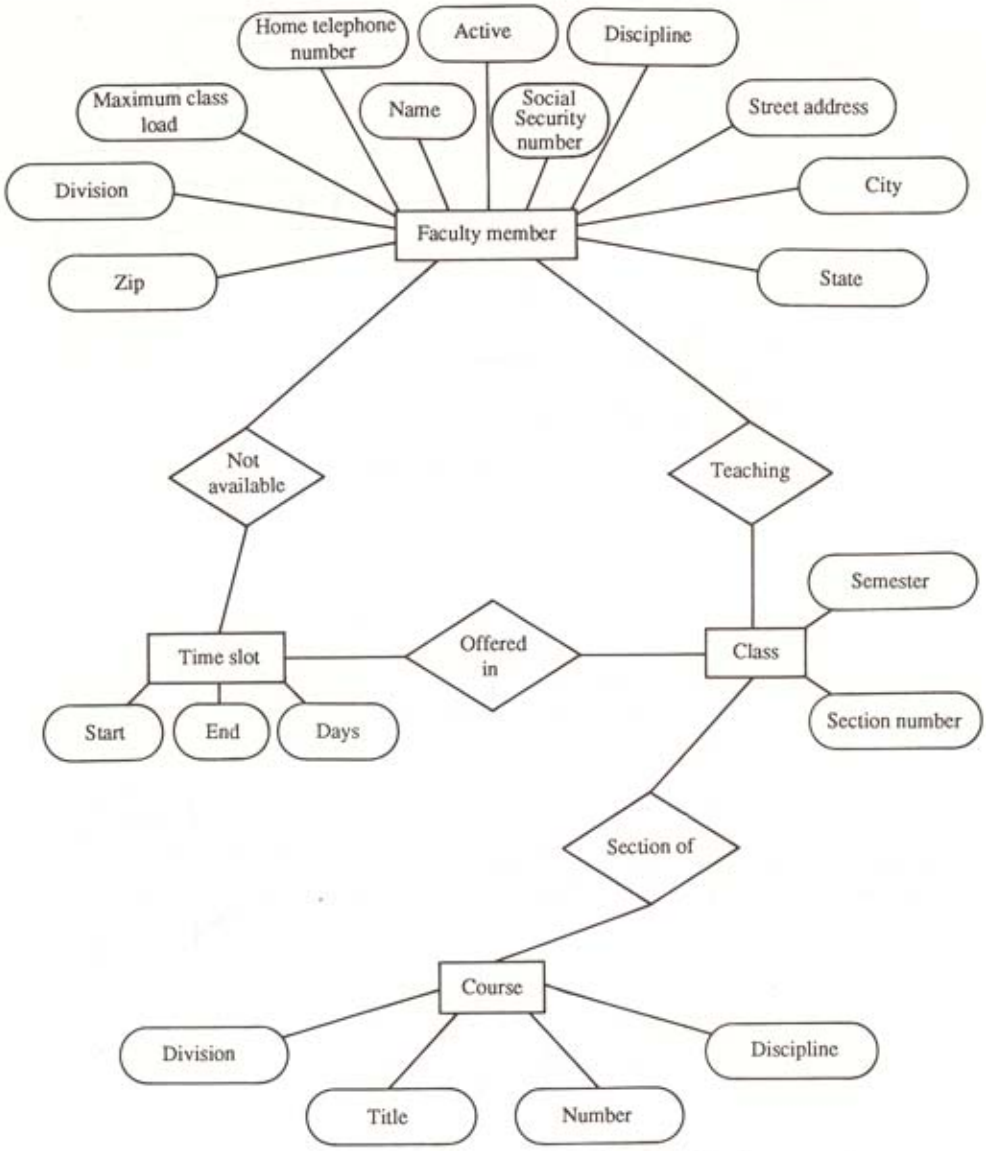


Figure 1. An entity-relationship diagram of a university department.

tionships may exist in the original entity-relationships diagram. These may be converted to binary relationships by replacing relationships with entity sets.[3] The relationship is replaced by an entity set consisting of identification numbers. This is then used to give binary relationships between the new entity set and each of the original entity sets.

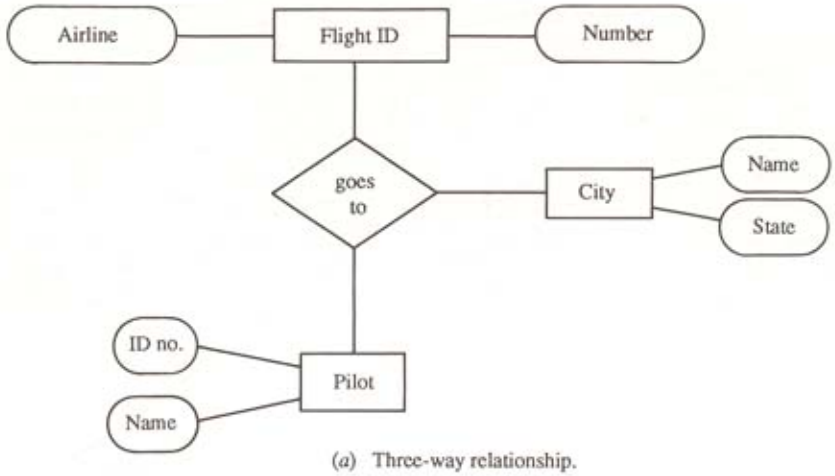
For example, in Fig. 2(*a*) the relationship 'goes to' connects a pilot to a specific leg of a flight, the leg going to the city in the relationship. There is a three-way relationship between a flight, a pilot and a city. To convert it to binary relationships create a new entity 'Flying' (see Fig. 2(*b*)) having one attribute which will serve as the key. This new entity set has a binary relationship with each of the other entity sets, and the information contained in these new relationships makes the 3-way relationship redundant.

2.3.2 Identify primary keys

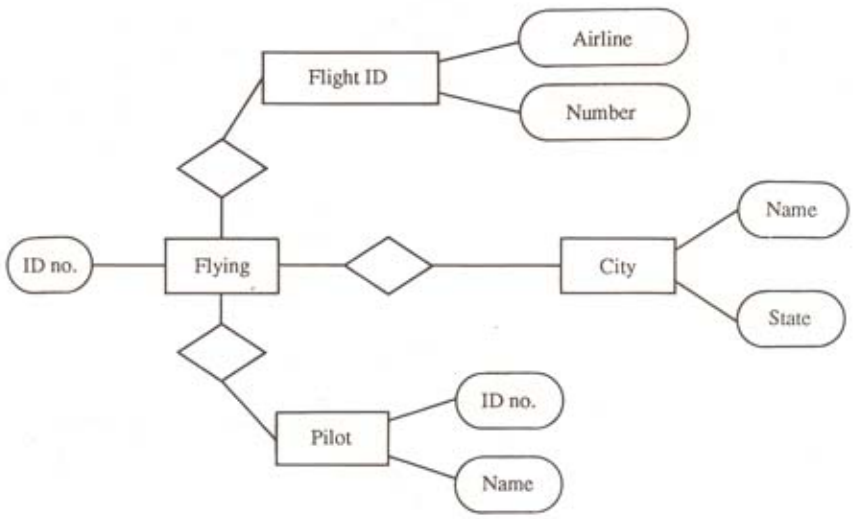
The first step in converting to a bubble diagram is to remove the entity sets. This is done by selecting a primary key. Of course, there may be more than one available choice, but one can be selected. As always, it is desirable to have a key which contains no redundant attribute sets. This primary key (which may consist of several attribute sets) then replaces the entity rectangle as a bubble containing the key. The attribute sets are now connected to the key bubble, and relationships are shown between key bubbles (see Fig. 3). If the key is a single attribute (as social security number is in Fig. 3), the corresponding bubble can be removed from the set surrounding the key bubble. Otherwise, the original bubbles must be maintained to preserve the existence of the individual attributes.

Occasionally, there exists an entity set for which no combination of attributes form a key. For example, the class entity set in Fig. 1. This can be resolved in two ways. The simplest is to add an attribute which is an identification number. This can be used to uniquely identify a particular class and forms the primary key.

The second option is to realise that the entity is in fact uniquely determined by one or more of its relationships. A class is defined in part by its relationship with a course. The particular course a class is related to and the attributes of the class together uniquely identify that class.



(*a*) Three-way relationship.



(*b*) Relationship replaced by entity set.

Figure 2. Conversion to binary relationship.

Thus, to form a primary key for an entity set identified in part by its relationships, the attribute sets of the entity sets it is related to are examined. A combination of attributes is then selected to form the key. In particular, the key for the entity set class is the combination of attribute sets: division, number, section number and semester.

Identifying a primary key for each entity set, and replacing the entity set with a key bubble in this conversion process, is a reflection of the fact that entities are not stored in the database, but are in fact represented by their attributes. Thus this is a practical first step towards a database scheme.

2.3.3 Examine each relationship

The relationships shown in the entity-relationship diagram can now be removed, and replaced by simple connectors between the primary keys. In doing so, however, the type of each relationship should be noted. To do this the attribute sets being connected should be considered to determine the degree of the relation. If the relation is 1–1 the connector should be labelled with a single arrow coming from each attribute set. If the relationship is 1–many, the connector should be labelled with a double arrow pointing to the attribute set, which can have many values associated with each value of the other set. The arrow in the opposite direction is single. For many–many relationships, both arrows are double.

For example, in Fig. 3 the relationship between social security number and class is 1–many, since one teacher teaches many classes, but each class has only one teacher.

2.3.4 Examine connections between attributes

There are three steps to this process. First, the existing connections between attribute sets (which are between non-key attribute sets and keys) are examined and labelled as to their type, i.e. 1–1, 1–many or many–many. Many–many relations between attributes are not of interest and are therefore not marked. This is done in the same manner as described above.

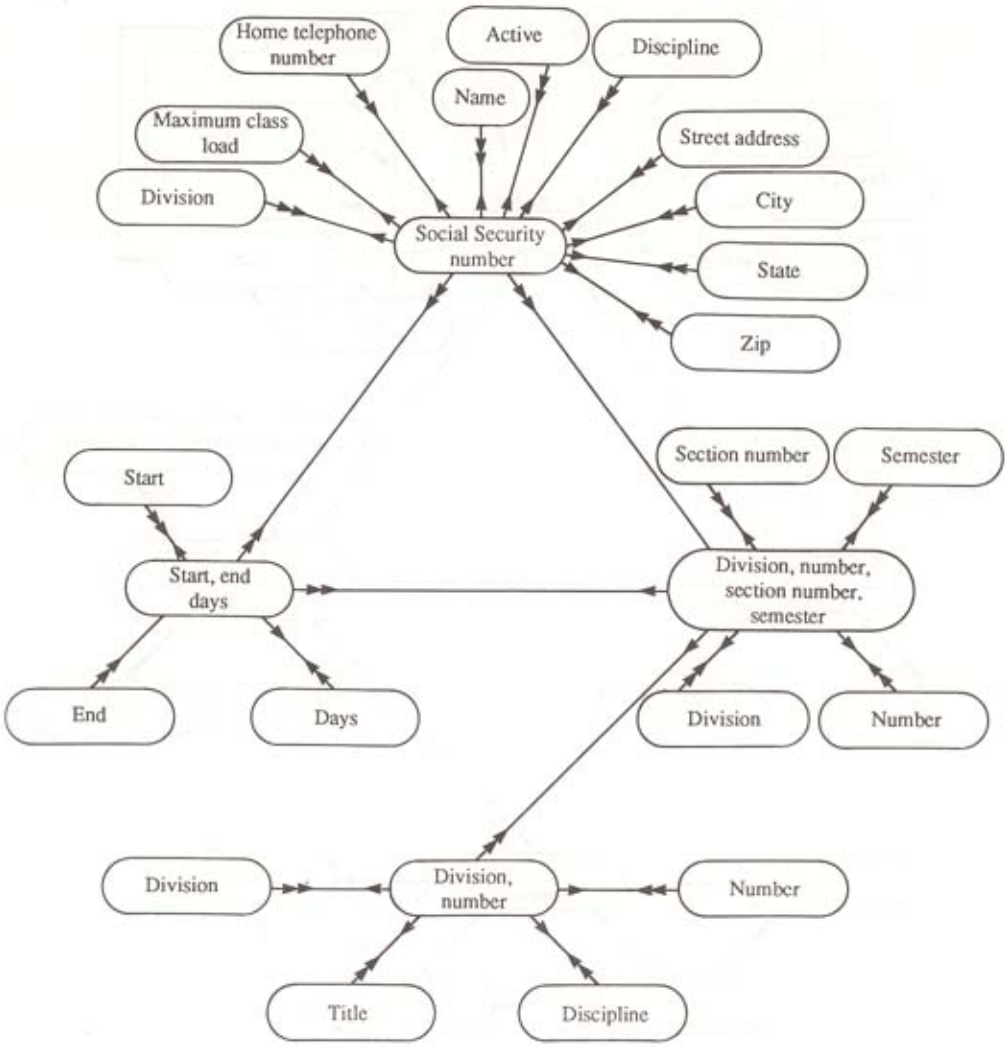


Figure 3. Replacement of entities by key bubbles.

Secondly, all possible pairs of attributes should be examined and the possibility of a 1–1 or 1–many relationship between them considered. If such a relationship exists, the attributes are connected and labelled as previously described.

Thirdly, each attribute bubble should be evaluated as to its relationship to combinations of other attribute bubbles. If its value can be deduced from the values of some combination of attributes (excluding itself and the key) this needs to be noted. A new bubble is introduced containing the combination of attribute values. This bubble is connected to the key bubble, the bubbles containing its component attributes, and the bubble containing the attribute whose value can now be derived. All of these connections are labelled as previously described.

For example, in Fig. 4 a bubble is created containing street address, city and state. This is necessary, because this combination of information uniquely determines the zip code. The U.S. Post Office has divided the country into areas, each uniquely described by a 7-digit zip code. Therefore a zip code uniquely determines a city or a state, but not a street address. However, one city may have several zip codes, so the reverse is not true. Also street address and city is not enough, as 100 Main Street, Richmond could be in Virginia or in California.

This examination of combinations of attribute sets is the first step in identifying various dependencies which must he removed from the database. For example, see the dependencies marked between zip, city and state in Fig. 4.

This conversion process does not eliminate the need for ascertaining the dependencies between attribute sets, which must be done any time a relational database is created. However, the visual description helps ensure a complete analysis.

3. USING THE BUBBLE DIAGRAMS TO REACH FOURTH NORMAL FORM

3.1 Construction of Relations

At this point the bubble diagram is broken up into smaller units, each of which represents one relation in the resulting database scheme. Note that this is not the final version of the scheme, as dependencies may still have to be removed.

Bubbles that have connections with a single arrow leaving them uniquely identify the attributes to which they point. Therefore these bubbles are a good choice for a primary key of a relation. Note that the conversion process from entity-relationship diagram to bubble

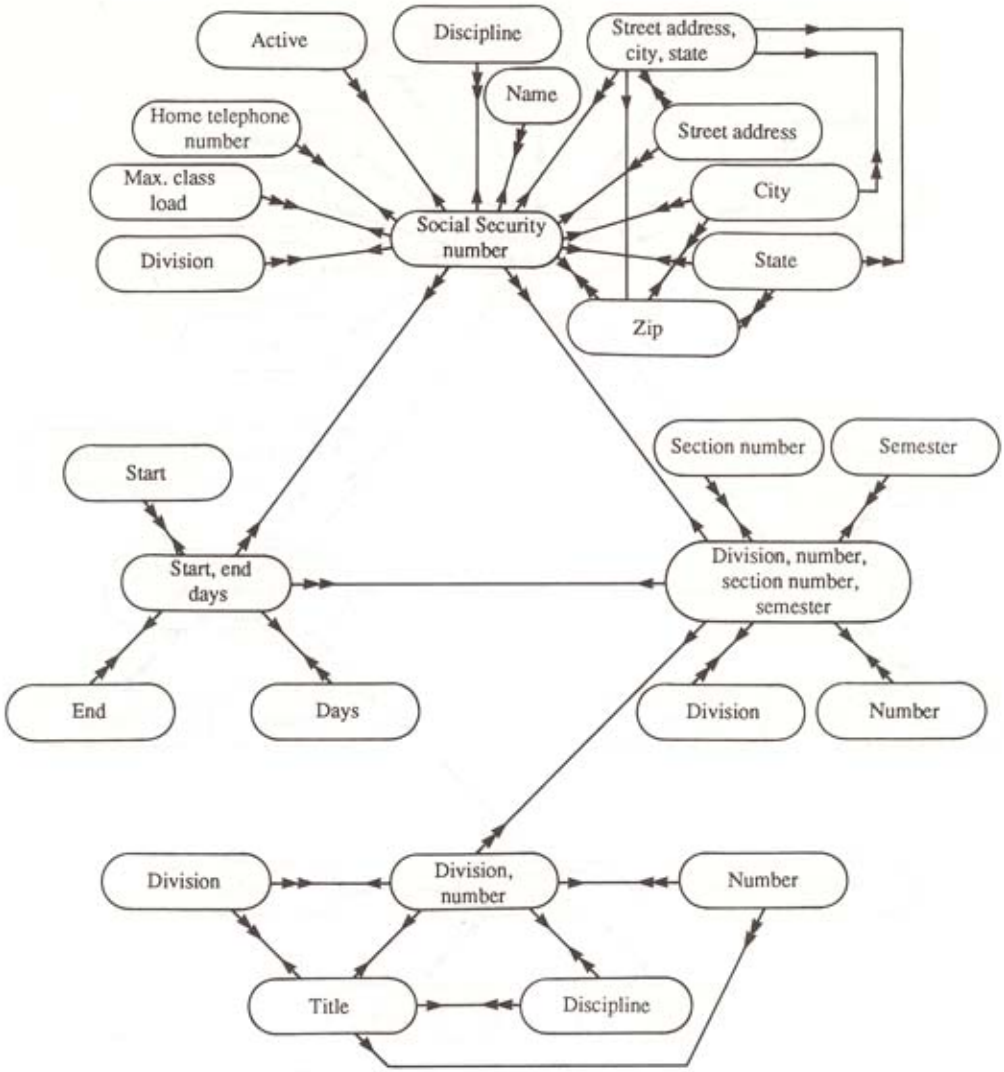


Figure 4. Bubble diagram corresponding to Fig. 1.

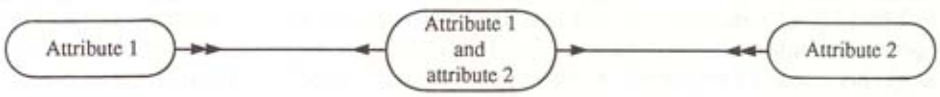
diagram has caused the formation of at least one such bubble for every original entity.

Relations, then, are formed by choosing such a bubble and all the bubbles it points to, using a single arrow. Care must be taken that every connection is used. The result is a collection of smaller bubble diagrams as shown in Fig. 5.

Note that this can lead to two identical relations. For example, choosing division and number as a primary key gives relation 'Course' of Fig. 5. However, 'title' can also be chosen as a primary key, which leads to the same relation. Obviously only one copy is kept.

Also, the possibility exists of creating two relations where one is contained within the other. For example see Fig. 6. Relations 'Address' and 'Zip code' are obviously contained in relation 'Faculty'. Therefore, 'Address' and 'Zip code' are not kept as they give no new information.

A problem arises with many–many connections. Since there is no single arrow present (for example between time slot and social security number in Fig. 4) the connection cannot be represented in the new set of bubble diagrams. This can be solved quite easily by borrowing a technique used in network databases.[4] The many–many relationship is broken into two 1–many relationships, as shown below.



Here a dummy bubble is created which interfaces between the two attributes. This bubble simply contains the information taken from both attributes and therefore the relationships are guaranteed to be 1–many.

Using the technique described above, we can now form a relation which will contain only the two attributes; its key will be both attributes combined (see Fig. 5).

3.2 Recognition of dependencies

3.2.1 Functional dependency

Given a relation R, a set of attributes Y of R is functionally dependent on a set of attributes X of R if and only if, whenever two tuples of R agree on their X values, they also agree on their Y values.[5]

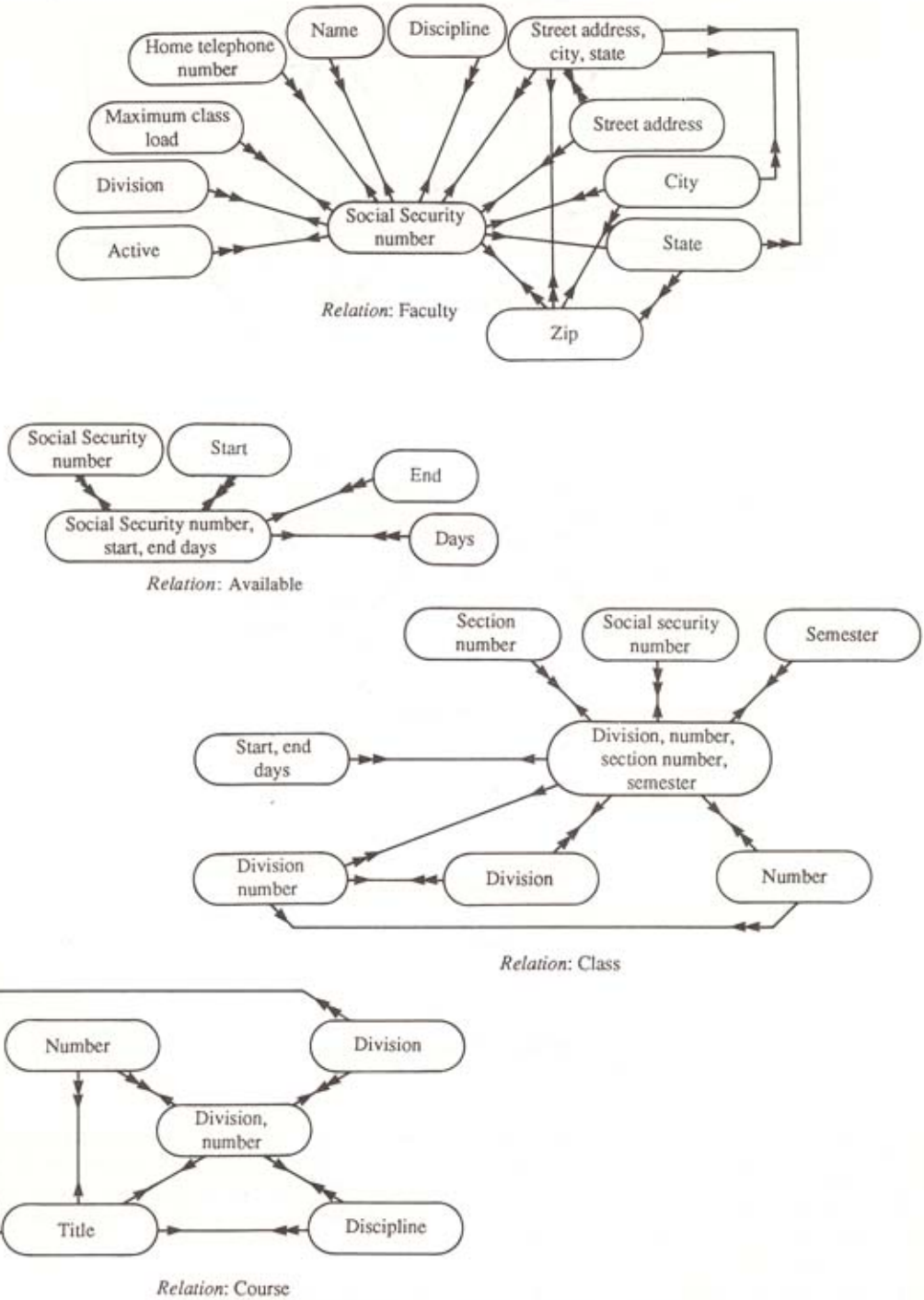


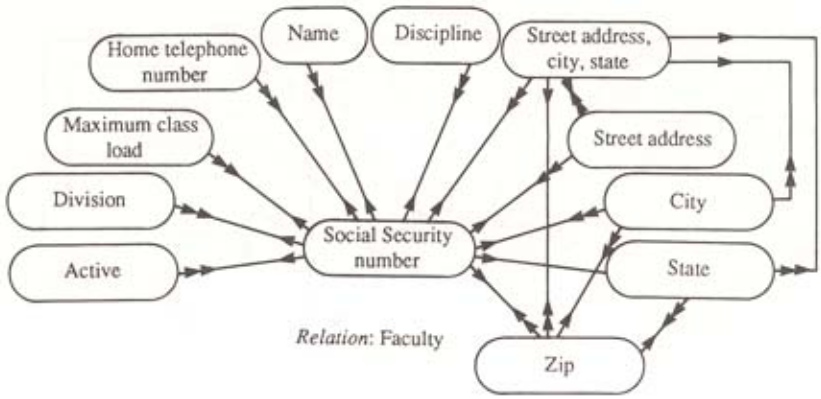
Figure 5. Bubble diagram showing separate relations.

Therefore, every connection on the bubble, diagram which is labelled with at least one single arrow represents a functional dependency. The attribute(s) the arrow points to are functionally dependent on the attribute(s) in the bubble the arrow is pointing from. Since all many–many connections have been removed, all connections on the bubble diagram represent at least one functional dependency.

3.2.2 Transitive dependency

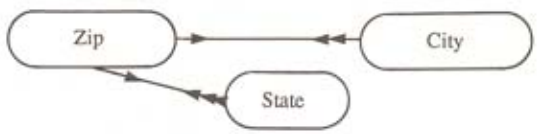
Given a relation R, a set of attributes X of R, attribute A of R (A not in X) is transitively dependent upon X in R if there is a set of attributes Y of R (A not in Y) such that Y is functionally dependent on X, X is not functionally dependent on Y and A is functionally dependent on Y.[6]

There are nine possible configurations between bubbles representing X, Y and A. These are shown in Fig. 7. In (a), (b) and (c) Y is functionally dependent on X, but X is functionally dependent on Y as well. Therefore these configurations do not show a transitive dependency. In the next three configurations ((d), (e) and (f)) Y is not functionally dependent on X, therefore there is no transitive dependency. In the final three configurations ((g), (h) and (i)), Y is functionally dependent on X, but the reverse is not true. However, A is functionally



Relation: Faculty

Relation: Address



Relation: Zipcode

Figure 6. Redundant relations.

dependent on Y only in (g) and (h). These two configurations therefore show a possible transitive dependency while (i) does not.

Thus, to find transitive dependencies, only two patterns need be remembered and looked for on the bubble diagram. These patterns are those shown in (g) and (h) of Fig. 7. Note that A may still be a subset of X or Y within these patterns, since some bubbles contain several attributes. Obviously these 'apparent' transitive dependencies would be disregarded.

3.2.3 Multivalued dependency

Given a relation R, let X and Y be disjoint subsets of R and let $Z = R-(X \bigcup Y)$. Y is multivalued dependent on X if for any two tuples t_1, t_2 in R with identical values for each of the attributes in X, there exists a tuple t_3 in R with values for the X and Y attributes matching those of t_1 and values for the Z attributes which match those of t_2.[7]

This implies that there is a 1–many relationship from X to Y which is independent of Z. At the same time, by symmetry there is a 1–many relationship from X to Z independent of Y. This would be recognised in a bubble diagram by the pattern shown in Fig. 8. Note that this is not a multivalued dependency if any other connectors appear besides those shown in Fig. 8.

3.3 Normalisation

3.3.1 Second normal form

It is assumed that the relational tables constructed from the bubble diagrams will be in first normal form. This is a reasonable assumption as most tables are intuitively constructed in this manner.

To have a relation in second normal form, all non-prime attributes must be fully functionally dependent on every key of the relation.[8] A non-prime attribute is one which is not part of a key.

3.3.2 Third normal forms

For a relation to be in third normal form no non-prime attribute should be transitively dependent on any key of the relation.[9]

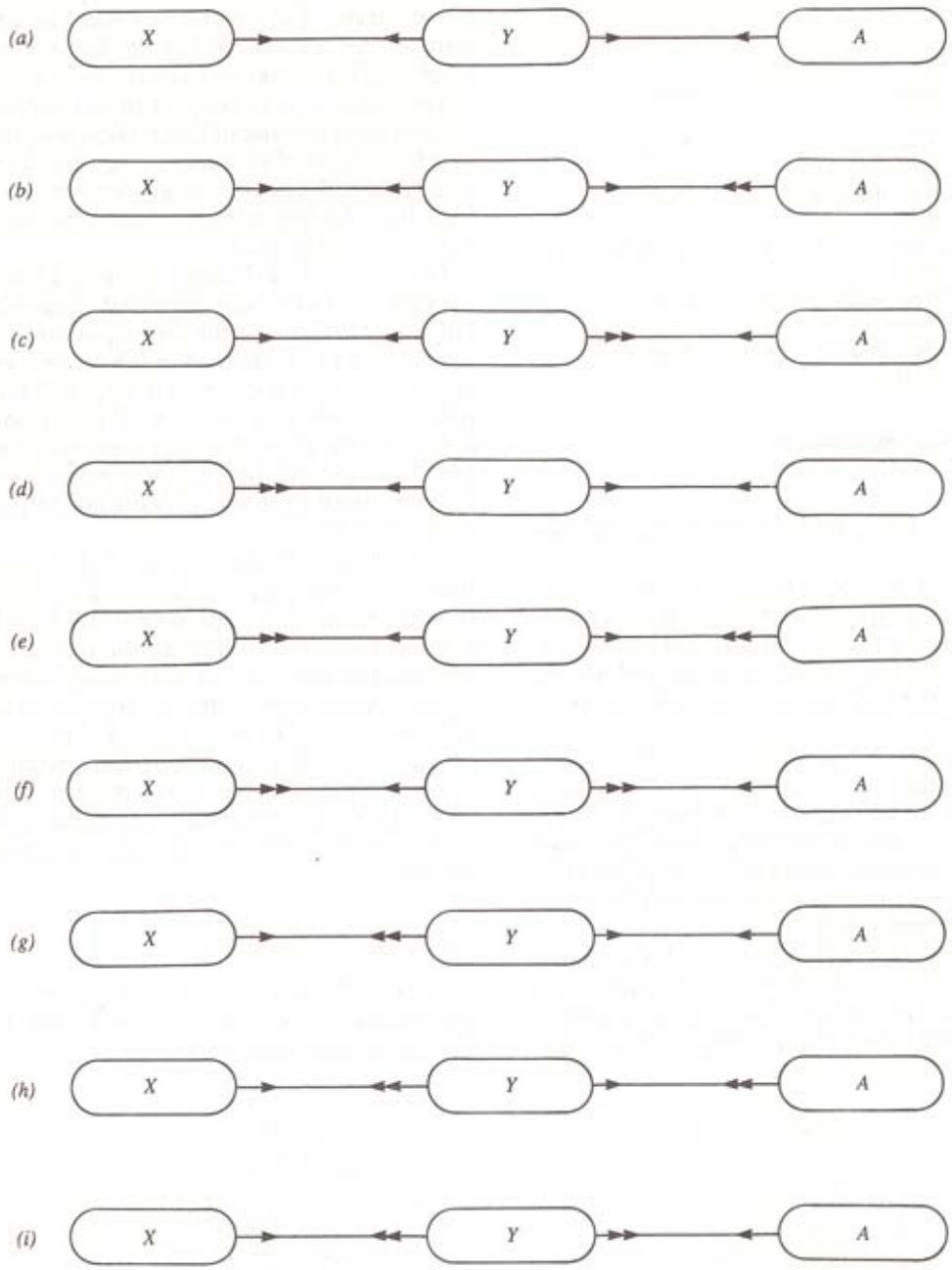


Figure 7. All possible connections between three bubbles.

The more rigorous version of third normal form, Boyce–Codd normal form, states that no attribute should be transitively dependent on any key of relation.[10]

In Section 3.1 the means for detecting secondary keys was discussed. In Section 3.2.2 the patterns which illustrate transitive dependencies were described.

All that is necessary to check that a relation is in Boyce–Codd normal form is to start from each key and check that neither pattern (*g*) nor (*h*) of Fig. 6 exists where X is the key. For example, pattern (*g*) does not exist in Fig. 4, but pattern (*h*) does. In relation Faculty, letting X = Social Security number, Y = Zip and A = State gives a transitive dependency. The complete list of transitive dependencies for relation Faculty is given in Fig. 9. No other relation has transitive dependencies.

The table in Fig. 9 was constructed by listing all the bubbles which satisfied pattern (*h*) of Fig. 6. In the last three cases it is obvious that A is a subset of Y. Thus these are not true transitive dependencies and can be ignored.

To remove dependencies, the standard decomposition rule is followed.[11] One new relation is formed by removing A from the original relation. A second relation is formed using Y and A, where Y is the key of this new relation. Applying this using the transitive dependencies of Fig. 9

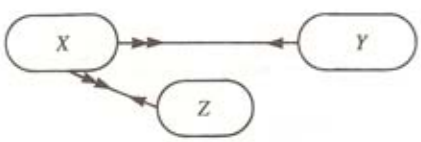


Figure 8. Multivalued dependency pattern.

X	Y	A
Social security number	Zip	City
Social security number	Zip	State
Social security number	Street address, city, state	Zip
Social security number	Street address, city, state	Street address
Social security number	Street address, city, state	City
Social security number	Street address, city, state	State

Figure 9. Apparent transitive dependencies.

gives the relations in Fig. 10 which replace the relation Faculty of Fig. 5.

It is obvious from the construction of the bubble diagram that every non-prime attribute is dependent on the primary key used to form the key bubble. If the primary key was properly chosen, with no redundant attribute set, then the non-prime attributes are fully dependent on the key.

The main concern is the possible existence of other keys. If there is another key it will have a 1–1 relation with the key bubble. In fact, keys were identified during the construction of the bubble diagram, although it was not thought of this way. For example, it was noted that attribute title could also be a key for relation Course in Fig. 5 when it was constructed.

If we can find a bubble which is not fully functionally dependent on some key the diagram must be split. The two new relations are formed as follows. The first relation is created by removing the troublesome key from the diagram. The second relation consists of the key bubble and the bubble for the key removed from the original. This breakdown ensures no loss of information.

Note that it is unlikely to find a violation of second normal form because of the method used to construct the relations. Since they were formed by choosing bubbles with outward-pointing single arrows, the chances are high that the breakdown would have been done automatically at this stage.

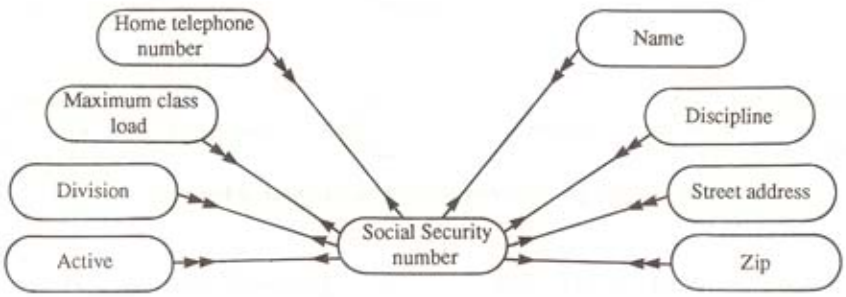
Notice that the bubble 'street address, city, state' appears to have been removed from the Faculty 2 relation, even though the decomposition rule states that the Y value should remain in the original relation. What has in fact happened is that city and state were each removed from the relation. This removal must be complete and thus they were removed from the 'street address, city, state' bubble, reducing it to 'street address'. Thus the third transitive dependency of Fig. 9 was dealt with indirectly.

This example raises the point that removing a bubble from a relation implies that its attributes be removed from other bubbles containing them. Only in this way is removal complete. Otherwise the transitive dependency continues to exist, but in a disguised manner.

Note that decomposition is not always desirable. Fig. 10 shows two relations, City and State, in Boyce–Codd normal form. However, both these relations are static and so there is no maintenance overhead associated with them. Also, it is rare to want one without the other. This gives a good argument for not decomposing relation Faculty.

3.3.3 Fourth normal form

A relation R is in fourth normal form if, whenever there is a multivalued dependency $X \twoheadrightarrow Y$, $Y \not\subset X$, $Y \neq R$, X is or contains a key of R.[12]



Relation: Faculty 2

Relation: City

Relation: State

Figure 10. Relation Faculty decomposed to Boyce–Codd normal form

Relation: Faculty 2

Social Security number	Active	Division	Maximum load	Telephone number	Name	Discipline	Address	Zip

Relation: City

Zip	City

Relation: State

Zip	State

Relation: Available

Social Security number	Start	End	Days

Relation: Course

Division	Number	Title	Discipline

Relation: Class

Division	Number	Section number	Semester	Social Security number	Start	End	Days

Figure 11. Final set of relations in table form.

Fig. 8 shows the pattern which depicts a multi-valued dependency. Using the process described in Section 3.1 to construct relations will prevent the existence of the multi-valued dependency pattern. The construction rules will lead to relations YX and ZX. Therefore any database created by these rules will be in fourth normal form.

3.3.4 Table construction

Each relation is now used to create a table. Bubbles containing one attribute give rise to one column of the table. Bubbles containing several attributes generate one column per attribute to ensure first normal form. Duplicate columns (e.g. generated by relation Available in Fig. 3) are removed. The relations are now presented in the standard relational database format (see Fig. 11).

4. SUMMARY

It should be clear from the presentation of the bubble diagram method that standard relational theory and techniques have been used throughout. This method simply provides a pictorial representation of the dependencies which exist within a collection data.

None of the steps usually used in creating a database in fourth normal form is bypassed by using this technique. However, the order of the steps and the visual mechanism help to ensure that appropriate effort is put into determining the dependencies that exist. It is felt that all

too often the relationships between attributes are not fully considered until late in the process of database construction. Early consideration of dependencies can lead to better understanding of the data and better design, irrespective of the methods used in the design process.

Obviously, all the problems that exist with traditional design methods still exist. Failure to identify relationships between data items correctly will still lead to un-normalised databases. Problems associated with decomposition into third and fourth normal forms, such as unenforceable dependencies and creation of redundant relations, are still likely to occur and must be watched for. However, use of the bubble diagram method does not introduce new problems.

An advantage of the bubble diagram is that once all relationships between data items have been noted, the normalisation-by-decomposition process is aided by a visual representation of the dependencies. In particular, multi-valued dependencies are easier to detect, making fourth normal form more reachable. It is thought that this method can be explained without needing to discuss the intricacies of the mathematics behind the normal forms. This is the major advantage of the bubble diagram method and the reason it is felt to be important.

REFERENCES

1. J. D. Ullmann, [*Principles of*] *Database Systems*. [Computer Science Press, Rockville, MD,] 17–18 (1982).
2. K. S. Dawson, Designing a relational database for a problem solving environment. *M.S. Thesis*, Department of Mathematical Sciences, Virginia Commonwealth University, (1985).
3. L. I. Brady, A universal relation assumption based on entities and relationships. *Proceedings of the 4th International Conference on Entity-Relationship Approach, Chicago*, (1985).
4. J. D. Ullmann, *Database Systems*, 102–106 (1982).
5. C. J. Date, *An Introduction to Database Systems*. vol. 1. Addison-Wesley, New York, 364–366 (1986).
6. D. Maier, [*The Theory of*] *Relational Databases*. [Computer Science Press, Rockville, MD,] p. 100 (1983).
7. D. Maier, *Relational Databases*, 124 (1983).
8. D. Maier, *Relational Databases*, 99 (1983).
9. D. Maier, *Relational Databases*, 100 (1983).
10. D. Maier, *Relational Databases*, 118 (1983).
11. J. D. Ullmann, *Database Systems*, 237–241 (1982).
12. D. Maier, *Relational Databases*, 135–136 (1983).