https://doi.org/10.48009/2_iis_2008_305-312

IMPROVING END-USER DATABASE DEVELOPMENT QUALITY: A 5C DATA MODELING METHOD

Hsiang-Jui Kung, Georgia Southern University, hjkung@georgiasouthern.edu
Hui-Lien Tung, Paine College, tungh@mail.paine.edu
Adrian Gardiner, Georgia Southern University, agardine@georgiasouthern.edu

ABSTRACT

The growth of end-user computing has been driven by advances in both personal computer hardware and software technology and their ever-decreasing pricing. Most casual end-users shy away from database design due to its complexity. However, more sophisticated end-users try to take advantage of complex PC-based database management system products to develop their own database, but frequently experience difficulties in producing quality databases because they do not possess sound data modeling training/skills. To address this problem, the authors propose a simple and straightforward normalization algorithm, the 5C data modeling method. The method focuses on easy-to-follow steps and eliminates references to technical terminology that may confuse the user-developer. The experiment results show that users can apply these techniques to build better databases without understanding traditional textbook terms such as first, second and third normal forms, and referential integrity.

Keywords: Data Model, Entity Relationship Diagram, Normalization, Relational Model

INTRODUCTION

Since the 90's, the rapid growth of end-user development (EUD) has earned it a place, together with outsourcing, as the top two most popular methods of organizational use of information resources. The growth of end-user computing has been driven by increasingly inexpensive hardware, increasingly powerful and easy to use software, and user demand for control of information resources [18, 23]. The active participation of end users means the tasks that are traditionally performed by professional software developers are transferred to end users. These applications support a wide range of information provision and decision-making activities and contribute to business processing in a wide range of tasks [21]. Increasingly, the ability to develop small applications forms part of the job requirements for many positions [12]. Four major phases of the IS application life cycle (analyze, design, implement, and maintenance) are performed by end users. Organizations that rely heavily on applications developed by end users have to offer good support and training so the end users can perform these tasks.

The benefits and risks of end-user development have been cited in the literature. While end-user computing empowers users, saves money and offers new opportunities to improve effectiveness of organizations, some researchers still caution the industry that allowing end users to perform these tasks might not be ideal since the technical abilities of end-user developers may vary considerably. Some drawbacks can affect EUD and create risk for an organization's information resources. For example, if some situated practices of end users are inadequate, they will be replicated in the software environment obtained through EUD activities. Lieberman and his colleagues [17] state that there is magic and power in creating programs by direct manipulation activities, as opposed to writing code. To reduce risk, while at the same time enhancing development, end user development needs technologies that foster collaboration between communities of end user designers and between users and managers, while increasing motivation and reducing cognitive and organizational costs [8].

Although a wide range of tools are available for use by end-user developers, the most commonly used software tools have been spreadsheets. In a survey to determine the types of applications developed by end users, Rittenberg and his colleagues [21] identified over 130 different types of applications. Over half of these were accounting related, but marketing, operations and human resources applications were also heavily represented. Unlike these tools, database design has not gain popularity in the end user development arena mainly because it is a difficult task for novice database designers. The majority (88%) of the 34 organizations participating in Taylor, Moynihan, and Wood-Harper's [25] study used spreadsheets for end-user development, whereas only 35% used query languages, and 12% used databases [9, 19, 20].

To examine the cause of the slower rate of adoption for end user database design, we first look into the "how-to". The first step of database design includes the elicitation of information requirements and representation of these requirements within a conceptual model (data modeling). In the case of end-user developed applications, elicitation is minimal due to the elimination of user-analyst communication; thus, data modeling dominates the task of database design [3]. In the development of the database schema, users map the objects of the reality to a conceptual model representation, while a data model provides representation primitives to aid in the development process. The term "Data model" refers to the structure of how data is stored in the database and the operations available for the user to manipulate the data in the database. The Data model has to evolve because of the increasing complexity of data requirement. There are three traditional data models: the hierarchical model, the network model, and the relational data model [6]. Since the relational data model saves users from needing to know the tedious physical details of data, it has replaced other traditional models and become one of the most widely used data models in the commercial world. The relational data model (RDM) has also been criticized for its difficulty in capturing the semantics of the real world, leading to the development of many semantic and conceptual models. One of these is the entity relationship model [7]. To enhance and increase end user database development means making data model concepts and skills easy to learn and to master like other tools, i.e., increasing end-user performance in their interactions with database systems.

Previous database design studies indicate that end-user performance is tightly linked to the characteristics of the user-database interface, in particular to the data model and the query language, which are vital tools for end-users to design and access the database [4, 5, 16, 22]. Human factor studies on database design and data manipulation show that user performance may be affected by the following variables: data models/query languages, tasks, human characteristics, and system characteristics [3]. The common factor is the data model. From our teaching experience, we also believe that data modeling and normalization concepts are difficult concepts for IS/IT students to master, many of whom, in turn, become end users. Such impediments may affect users' performance, thereby presenting a significant obstacle for end-user database development.

The purpose of a data model is to model data or retrieve data correctly. The key factor to measure user performance is modeling accuracy. Modeling accuracy has also been the most important indicator affecting user performance in human factor studies [3]. Some studies have included other performance measures, such as time taken and user confidence [16], but most analyses have generally focused on accuracy. Modeling accuracy refers to the correctness of the data model developed by subjects. It indicates the user's ability to represent information with a data model based on a natural language description of the application. Data models may affect user performance during database design.

Strategies for reducing risks associated with end-user development have been presented in the literature, and there is some evidence to suggest that employing them is effective. For example, Alavi, Nelson, and Weiss [1] presented a comprehensive framework of controls for addressing risks at different stages of the application life cycle, and several studies have demonstrated the value of introducing controls during the design and development of spreadsheets [2, 11]. End-user training has also been shown to positively influence attitudes toward technology [24] and to improve the quality of end-user developed applications [13, 14]. The acknowledgment of the importance of training is quite interesting; despite having received little training themselves, the respondents considered training to be the most important approach to reducing the risks of end-user Web development. Nelson and Todd [19] suggested that training is perhaps the most effective tool for minimizing risks associated with end-user development. To incorporate these findings, we will propose an easy method that can be used to train end-users to apply in data modeling/normalization techniques to the design of their databases to ensure they will be free of anomalies. It is our belief (supported in the literature) that the easier the learning/training, the better the performance; and the better the performance (accuracy rate), the higher the willingness will be to use the tool.

Conceptual Data Modeling

Traditionally, database textbooks [10] have tackled conceptual data model problems using the top-down approach (e.g., Entity-Relationship (ER) modeling [7]), or bottom-up approach (e.g., normalization), or both. The textbook method contains two parts: normalization and data model visualization. The normalization steps will decompose relations to 3NF. The data model visualization converts the normalized relations/tables to the ER model. This paper presents

an easier-to-follow method which consists of two parts: normalization and entity relationship diagram (ERD). By following this method, end-users are not required to memorize the definitions of 1NF, 2NF or 3NF. The second part of the 5C method is an easy-to-follow algorithm to draw the ERD. This method will assist the end-user in developing better database design without requiring them to understand complicated database theory. This method has been proven to be sound and complete (see Appendix).

THE 5C DATA MODELING METHOD– COUNT, COMPLY, COMPARE, CONSOLIDATE, AND CONVERT

The 5C data modeling method is the extension of the alternative normalization technique [15]. We use the example below to illustrate the 5C method. The example is based on these assumptions: (1) the universal relation covers a single business process domain; (2) the universal relation is in the first normal form (1NF); and (3) the set of functional dependencies are given and are in closure. We assume that we are assigned to design a database for a publishing company. A universal relation *Book* and the set of functional dependencies (FDs) are given as follows. *ISBN* and *AuthorID* form the composite primary key of the *Book* table.

Book (PublisherID, PublisherName, Address, **ISBN**, BookTitle, Category, Loyalty, **AuthorID**, AuthorName, AuthorPhone)

FDs:

PublisherID → PublisherName, Address (FD1)

ISBN → PublisherID, PublisherName, Address, BookTitle, Category (FD2)

AuthorID → AuthorName, AuthorPhone (FD3)

AuthorID, ISBN → PublisherID, ublisherName, Address, BookTitle, Category, AuthorName, AuthorPhone, Loyalty (FD4)

Normalize the universal relation/table

1C. **Count** and write down the number of attributes on left-hand-side (LHS) and right-hand-side (RHS) of every functional dependency (FD). Don't change the numbers throughout the method.

(1) PublisherID → PublisherName, Address (2) (FD1)

(1) ISBN → PublisherID, PublisherName, Address, BookTitle, Category (5) (FD2)

(1) AuthorID → AuthorName, AuthorPhone (2) (FD3)

(2) AuthorID, ISBN → PublisherID, PublisherName, Address, BookTitle, Category, AuthorName, AuthorPhone, Loyalty (8) (FD4)

2C. **Comply**: Keep LHS attributes intact. All LHS attributes should be kept on the left-hand side, as is, even when the same LHS attributes appear on the LHS in another functional dependency. For this example, even though *AuthorID* appears two times on the left hand side (FD3, FD4), no change is to be made.

3C. **Compare**: Compare the number of LHS and RHS attributes.

First, find out whether there are repeated attributes on the right side in the set of FDs, and then compare the number of LHS and RHS attributes of those FDs.

We found: *PublisherID*, *PublisherName*, *Address*, *BookTitle*, and *Category* are in both FD2 and FD4. FD2 has one LHS attribute and five RHS attributes. FD4 has two LHS attributes and eight RHS attributes.

4C. **Consolidate**

Consolidate the RHS attributes.

a. Keep the attributes in the FD that have the smaller LHS number on the right side, and delete the additional copies in the other FD. (This step will eliminate partial dependency.)

Since FD2 has one LHS attribute while FD4 has two, we are going to keep these five attributes on the RHS of FD2 and delete them from FD4.

(1) ISBN → PublisherID, PublisherName, Address, BookTitle, Category (5) (FD2)

(2) AuthorID, ISBN → ~~PublisherID, PublisherName, Address, BookTitle, Category~~, AuthorName, AuthorPhone, Loyalty (8) (FD4)

AuthorName and *AuthorPhone* are in FD3 and FD4. Keep these in FD3.

(1) AuthorID → AuthorName, AuthorPhone (2) (FD3)
(2) AuthorID, ISBN → ~~PublisherID, PublisherName, Address, BookTitle, Category, AuthorName, AuthorPhone~~, Loyalty (8) (FD4)

b. When two FDs have the same LHS number, keep the copy that has the smaller RHS number and delete the additional copies. (This step will eliminate transitive dependency).

PublisherName and *Address* are in FD1 and FD2. Keep these two attributes in FD1

(1) PublisherID → PublisherName, Address (2) (FD1)
(1) ISBN → PublisherID, ~~PublisherName, Address~~, BookTitle, Category (5) (FD2)

5C. **Convert** the FDs to relations. Make the LHS attribute(s) the primary key(s) of that relation.
Now we have four consolidated FDs (no repeating RHS attributes):
PublisherID → PublisherName, Address (FD1)
ISBN → PublisherID, BookTitle, Category (FD2)
AuthorID → AuthorName, AuthorPhone (FD3)
AuthorID, ISBN → Loyalty (FD4)

We convert the four consolidated FDs to four relations/tables as follows:

T_1 (**PublisherID**, PublisherName, Address)
T_2 (**ISBN**, PublisherID, BookTitle, Category)
T_3 (**AuthorID**, AuthorName, AuthorPhone)
T_4 (**AuthorID, ISBN** → Loyalty)

Draw ERD

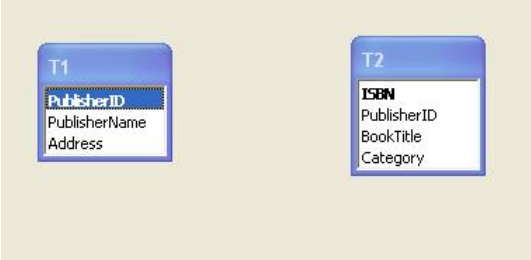
After completing the 5C steps, we can draw the ERD accordingly. The ER modeling approach is a widely accepted approach to identifying data requirements and incrementally developing a sound data structure.

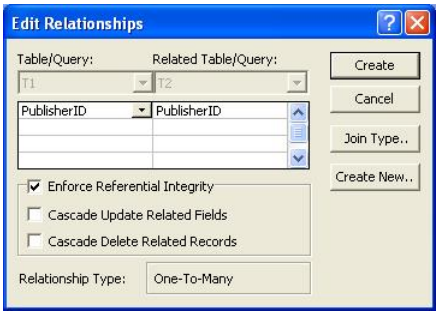
1. Draw an entity for every 3NF relation and add attributes to the entity. We will use Microsoft Access to create tables based on the normalized relations in the previous step. First, add the tables to the work space of the relationship diagram.

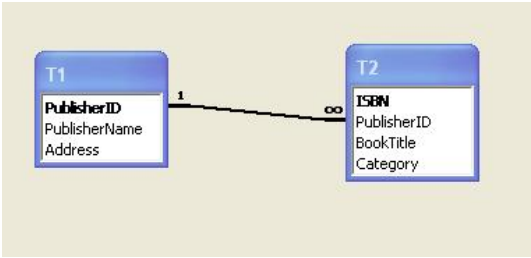
2. Next, look for common attributes (same attribute showing up in two tables) between tables.

PublisherID in Tables T_1 and T_2
AuthorID in Tables T_3 and T_4
ISBN in Tables T_2 and T_4

3. Connect entities using common attributes. Draw a relationship between any two entities where they have in-common attribute(s).
   (1) Assign cardinalities to every relationship based on the common attribute(s). When the common attribute is a single primary key, assign a cardinality of 1 on that side of the relationship. When the common attribute of an entity is a non-key attribute, assign a cardinality of many toward that entity in the relationship.
   - Add a relationship between two tables with the common attribute(s). *PublisherID* is the single primary key of Table T_1, so click on Table T_1's *PublisherID* first and then drag the attribute towards the *PublisherID* on Table T_2. When the relationship dialogue window pops up, choose "Enforce Referential Integrity". Access draws the relationship line and assigns cardinality between these two tables.

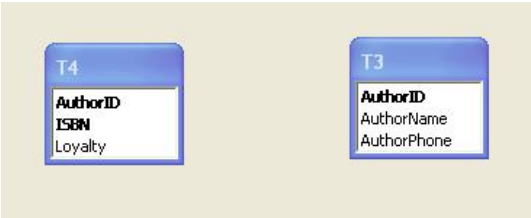


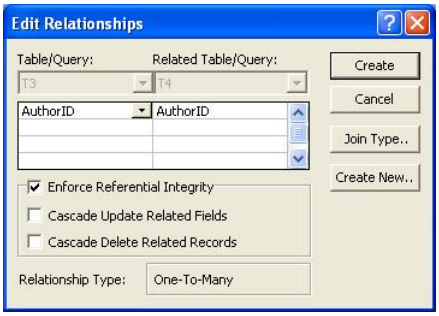


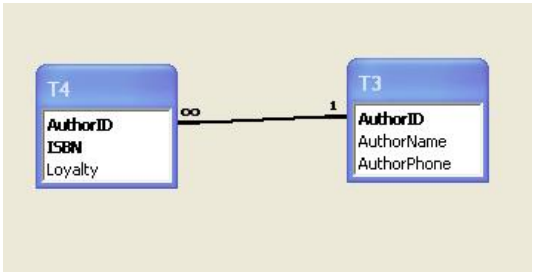


(2) When the common attribute(s) of an entity is/are part of the composite primary key, assign a cardinality of many toward that entity in the relationship.

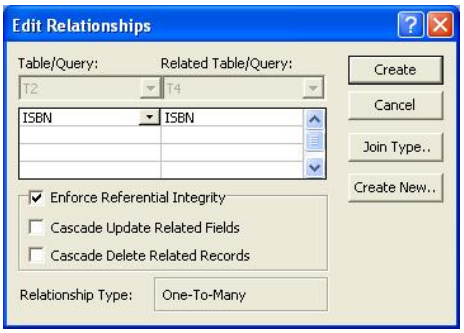
- *AuthorID* on Tables T_3 and T_4: is part of the composite primary key of Table T_4. Still start from Table T_3 because *AuthorID* is the single primary key of Table T_3.

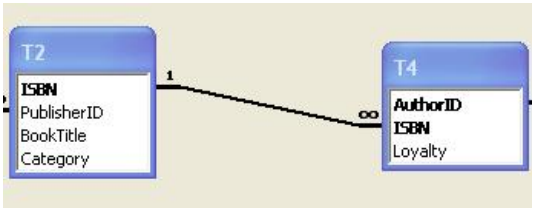




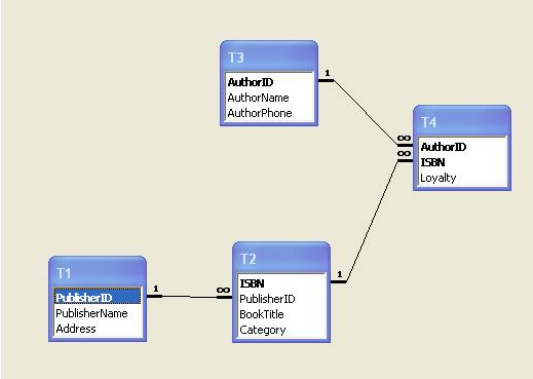


- *ISBN* appears on Tables T_2 and T_4. Start from Table T_2 because it is the single primary key of Table T_2.





4. After you are done, every table should be connected to at least one other table.



CONCLUSION

The 5C method offers simple and straightforward steps for teaching bottom-up database design. We believe that the 5C method is a valuable addition and supplement to database education. One would not use the 5C method as the only coverage of database design. In our view, students also need to understand the rationale behind the concepts of normalization and ERD and to develop the skills needed to determine the functional dependencies in the first place. To prove the effectiveness of the 5C data modeling method, we will conduct a comparison study of the traditional and 5C methods in the future.

REFERENCES

1. Alavi, M., Nelson, R. R., & Weiss, I. R. (1987) Strategies for end-user computing: An integrative framework, *Journal of Management Information Systems, 4*(3), 28-49.
2. Alavi, M., Phillips, J. S., & Freedman, S. M. (1990). An empirical investigation of two alternative approaches to control of end-user application development process, *Database, 20*(4), 11-19.
3. Batra, D., Hoffer, J. A., and Bostrom, R. P. (1990). Comparing representations with relational and EER models, *Communications of the ACM, 33*(2), 126-139.
4. Batra, D. & Antony, S. R. (1994). Effects of data model and task characteristics on designer performance: A laboratory study, *International Journal of Human-Computer Studies, 41*(4), 481-508.
5. Chan, H., Siau, K. & Wei, K-K. (1998). The effect of data model, system and task characteristics on user query performance: An empirical study, *Database, 29*(1), 31-49.
6. Chan, H. C., Teo, H. H., & Zeng, X. H. (2005). An evaluation of novice end-user computing performance: Data modeling, query writing, and comprehension, *Journal of the American Society for Information Science and Technology, 56*(8), 843-853.
7. Chen, P. P. (1976) The entity-relationship model—Toward a unified view of data. *ACM Transactions on Database Systems, 1*(1), 9-36.
8. Fischer, G., Giaccardi, E., Ye, Y., Sutcliffe, A. G., *and* Mehandjiev, N. (2004). Meta-Design: A Manifesto for End-User Development. *Communications of the ACM, 47*(9), 33-37.
9. Govindarajulu, C. (2003). End users: Who are they, *Communications of the ACM, 46*(9), 152-159.
10. Hoffer, J. A., Prescott, M. B., and McFadden, F. R. (2007). *Modern database management*, 8th Ed., Prentice-Hall, Upper Saddle River, NJ.
11. Janvrin, D. & Morrison, J. (2000). Using a structured design approach to reduce risks in end user spreadsheet development, *Information and Management, 37*(1), 1-12.
12. Jawahar, I. M. & Elango, B. (2001). The effect of attitudes, goal setting and self-efficacy on end user performance, *Journal of End User Computing, 13*(2), 40-45.
13. Kreie, J., Cronan, T. P., Pendley, J., & Renwick, J. S. (2000). Applications development by end-users: Can quality be improved, *Decision Support Systems, 29*(2), 143-152.
14. Kruck, S. E., Maher, J. J., Barkhi, R. (2003). Framework for cognitive skill acquisition and spreadsheet training in end-users, *Journal of End User Computing, 15*(1), 20-37.
15. Kung, H. & Reichgelt, H. (2004). An alternative technique to improve IS/IT students' learning of database normalization, *Journal of Informatics Education Research*, *6*(3), 45-55.
16. Liao, C. & Palvia, P. C. (2000). The impact of data models and task complexity on end-user performance: An experimental investigation, *International Journal of Human-Computer Studies, 52*(5), 831-845.
17. Lieberman, H., Paterno, F., Klann, M., & Wulf, V. (2006). *End-User Development: an Emerging Paradigm*, Netherlands: Springer.
18. McLean, E. R., Kappelman, L. A., & Thompson, J. P. (1993). Converging end-user and corporate computing. *Communications of the ACM*, *36*(12), 79-92.
19. Nelson, R. R., & Todd, P. (1999). Strategies for managing EUC on the Web. *Journal of End User Computing. 11*(1), 24-31.

20. Ouellette, T. (1999). Giving users the keys to their Web content. *Computerworld* (July 26), 66-67.
21. Rittenberg, L. E., Senn, A., & Bariff, M. (1990). *Audit and Control of End-User Computing*, Altamonte Springs, FL: The Institute of Internal Auditors Research Foundation.
22. Santhanam, R. & Batra, D. (1998). The human-computer interface in information systems design: Introduction to the special issue, *Database for Advances in Information Systems, 29*(1), 25-30.
23. Shayo, C., Guthrie, R., & Igbaria, M. (1999). Exploring the measurement of end user computing success. *Journal of End User Computing, 11*(1), 5-14.
24. Simmers, C. A. & Anandarajan, M. (2001). User satisfaction in the Internet-anchored workplace: Anexploratory study, *The Journal of Information Technology Theory and Application, 3*(5), 39-61.
25. Taylor, M. J., Moynihan, E. P., & Wood-Harper, A. T. (1998). End-user computing and information systems methodologies. *Information Systems Journal, 8*(1), 85-96.

APPENDIX: PROOF OF THE 5C DATA MODELING METHOD

Notation: In what follows, capital letters represent non-empty sets of attributes.

Definition 1: A functional dependency $A \rightarrow U$ is non-trivial if $A \cap U = \varnothing$. In other words, the left-hand side and the right-hand side of a non-trivial functional dependency have no attributes in common.

Definition 2: A functional dependency $A \rightarrow U$ is closed under a set of functional dependencies FD if U is the set of all attributes that are functionally dependent on A given FD.

Theorem: If each functional dependency in a set of functional dependencies (FDs) is non-trivial and closed under FD, then the set of tables generated using the 5C data modeling method fully normalized up to 3NF and the corresponding ERD is correct.

Proof: The 5C method reduces FD to a set FD', from which the tables are generated. Assume that the set of tables generated from FD' through the algorithm is not in 3NF. Then, at least one of the following must hold (definition of 3NF):

(i) There is a table T that violates 2NF (some non-key attributes depend on partial key(s));
(ii) There is a table T that violates 3NF (some non-key attributes depend on other non-key attributes).
(iii) There is a One-to-One relationship between 2 tables/entities.
(iv) There is a many-to-many relationship between 2 tables/entities.

Case (i)
The proof is by refutation. Suppose a table T violates 2NF and that the primary key of T is a set of attributes A. FD' must contain functional dependencies

FD_i: $A \rightarrow U$ $(1 \leq i \leq n)$ and
FD_j: $D \rightarrow V$ $(1 \leq j \leq n)$

where $D \subset A$ and $V \subseteq U$ (definition of 2NF, and the fact that all functional dependencies are closed). Thus, D is a subset of A, and U and V have attribute(s) in common.

By step 4C(a), the common attribute V would have been eliminated from FD_i. Hence, we derive a contradiction and conclude that all tables are in 2NF.

Case (ii)
Suppose there is a functional dependency between non-key attributes in a table T. FD' must contain functional dependencies

FD_k: $B \rightarrow W$ and
FD_l: $C \rightarrow S$

where $C \subset W$ and $S \subseteq W$ (definition of 3NF). Thus, C and S are subsets of W.

Since FD_l is non-trivial, $C \cap S = \varnothing$. Since $C \subset W$ and there are attributes in C and hence in W that are not also in S, $S \subset W$. Thus, S is a complete subset of W, and S is the set of common attribute(s) in the right-hand side of FD_k and FD_l. Moreover, since S contains fewer attributes than W, by step 4C(b), we would have eliminated S from FD_k. Hence, we derive a contradiction and conclude that no functional dependencies can be found between non-key attributes. Since we derive a contradiction for both cases, we conclude that the tables generated through the 5C method are fully normalized up to 3NF.

Case (iii)
Suppose there is a One-to-One relationship between two tables/entities. The common attribute(s) between the two tables must be the single primary key of both tables. FD' must contain functional dependencies

FD_p: A $\rightarrow$ X ($1 \leq p \leq$ n) and
FD_q: A $\rightarrow$ Y ($1 \leq q \leq$ n)
where the two FDs have identical LHS attribute.

Since each FD is in closure, no two FDs have identical determinant(s). Thus, X = Y and $FD_p = FD_q$. We conclude that there is no one-to-one relationship in a normalized ERD.

Case (iv)
Suppose there is a Many-to-Many relationship between two tables/entities. The common attribute(s) between the two tables must be in part of the RHS attributes. FD' must contain functional dependencies

FD_m: E $\rightarrow$ R ($1 \leq m \leq$ n) and
FD_t: F $\rightarrow$ Z ($1 \leq t \leq$ n)
where G $\subseteq$ R, G $\subseteq$ Z and G $\neq \varnothing$. Thus, there are more than one copies of some attributes (G) on the RHS of FD'.

Since Cases (i) and (ii) demonstrate that all the repeating attributes on the RHS of FD' have been deleted, there will be only one copy of each attribute on the RHS. We concluded that there is no many-to-many relationship in a normalized ERD.