Accounting For The Benefits Of Database Normalization

Ting J. (T.J.) Wang, Governors State University, USA
Hui Du, University of Houston-Clear Lake, USA
Constance M. Lehmann, University of Houston-Clear Lake, USA

ABSTRACT

This paper proposes a teaching approach to reinforce accounting students' understanding of the concept of database normalization. Unlike a conceptual approach shown in most of the AIS textbooks, this approach involves with calculations and reconciliations with which accounting students are familiar because the methods are frequently used in accounting courses. The approach gives students the opportunity to justify the changes made to database structure during the database normalization process in terms of its efficiency and effectiveness. Further, the justification will help students realize and understand the purpose and benefits of database normalization. Instructors can easily adopt this approach to any of their own exercises used to cover database normalization concept.

Keywords: Relational Database, Database Normalization, Normal Forms

INTRODUCTION

Today's accountants are increasingly called upon to participate in decision-making about the design and development of their company's information systems (IS) due to their knowledge of the business processes and expertise in accounting. Consequently, it would be very beneficial for accountants to have understanding of the tasks and processes of the IS design and implementation, including the pros and cons of database normalization. Moreover, twenty-first century accountants, as end-users of a database system, often need to create, maintain, and query their database. With knowledge about database normalization, i.e., taking data apart as it is stored, students will be able to understand the tasks involved in the query of the database (i.e., reassembling data together). It is our responsibility as AIS educators to insure that our accounting graduates are equipped with adequate database knowledge and skills before they enter into the accounting profession.

EDUCATION OF DATABASE NORMALIZATION

It is often difficult to motivate students to learn database normalization because of the dry and theoretical way in which it is presented in textbooks and classes. Most textbooks in the field of Management Information Systems (MIS) rely on the traditional technique of defining the normal forms to cover and explain the process of database normalization. For example: a table is in its first normal form (1NF) if each attribute contains simple values; a table is in its second norm form (2NF) if it is in 1NF and non-key attributes depend on the whole key; and a table is in its third normal form (3NF) if it is in 2NF and non-key attributes do not depend on other non-key attributes. Similarly, most AIS textbooks use the same traditional approach with the following definition: to remove any repeating groups in 1NF, remove any partial dependencies in 2NF, and remove any transitive dependencies in 3NF.

In the MIS field, alternative techniques involving algorithms and procedures are available in textbooks and the literature (Kung and Tung, 2006; Silberschatz, Korth, and Sudarshan, 2002). These alternative techniques are algorithm-based because MIS students are familiar with them and more comfortable following procedures than working with conceptual approaches. However, for accounting students, these alternative techniques are not suitable because they are algorithm-based and accounting students will have difficulty with them. It is possible that because

other alternative approaches available to accounting students have been lacking, the coverage of database normalization in AIS textbooks has decreased in recent years. Some AIS textbooks even exclude the topic entirely (e.g., Romney and Steinbart, 2009; Vaassen, 2002; and Jones and Rama, 2006).

The objective of database normalization is to allow the storage of data without unnecessary redundancy and thereby eliminate data inconsistency so that users can maintain and retrieve data from a database without difficulty. A normalized database eliminates anomalies in updating, inserting, and deleting data, which improves the efficiency and effectiveness of the database. Databases not being properly normalized to third normal form (3NF) will have serious problems with insertion, deletion, and updating anomalies. As a result, database normalization is an important process in systems analysis and design.

Although it is sometimes possible that the tables generated from a semantic data model, e.g., Entity-Relationship (E-R) or REA (Resources-Events-Agents) diagrams, may not need further normalization when all necessary entities are identified correctly and precisely, there still can be functional dependence between attributes of an entity. Some of the semantic data modeling, especially the E-R data modeling, is part of a conceptual schema that belongs to the so called "top-down design". After the E-R data model is mapped into a set of tables using mapping procedure, normalization must be applied to ensure that each table represents a logically coherent grouping of attributes and possesses the measures of goodness associated with normalization (Elmasri and Navathe, 2006). Nevertheless, some AIS textbook authors (e.g. Romney and Steinbart, 2009) claim that semantic data modeling, specifically referring to REA data modeling, can achieve a set of relational tables that are in 3NF without following normalization process. In fact, REA type of semantic data modeling relies heavily on detailed business processes to identify the tables and their attributes, while the traditional database normalization approach appears to be more general and can be applied to any situation without depending on specific business processes. The importance of database normalization has been proven and advocated in the literature for decades (Codd, 1970). Problems that occur from failing to apply database normalization (e.g., modification anomalies) are well known and documented. In general, semantic data modeling cannot assure 3NF unless it is embedded with a normalization algorithm in the modeling prior to the database implementation.

This paper proposes a teaching/reinforcement approach to enhance accounting students' understanding of the concept of database normalization. This teaching approach involves calculations and reconciliations consistent with techniques and methods used in accounting courses and with which students are familiar and comfortable. Using the calculation and reconciliation approach, students are given the opportunity to visually account for the changes made to the database structure in the database normalization process. When students relate the changes to the benefits of each of the database normal forms progressed, students are able to effectively grasp the purpose of the database normalization.

We use a sales database, a commonly seen exercise in most of Accounting Information Systems (AIS) textbooks, in the paper to demonstrate this approach to teaching database normalization. Instructors can easily adopt this teaching approach to any of their own exercises used in covering concept of database normalization. The paper can be used as a stand-alone or an add-on exercise with any textbook materials that teach the concept of database normalization.

The paper is organized as follows: Section I describes the requirements used in the sales database exercise; Section II provides suggested solutions to the requirements, which demonstrates the reinforcement approach involving calculations and reconciliations; Section III presents teaching notes to instructors who are interested in using the reinforcement approach to cover the concept of database normalization.

I. SALES DATABASE EXERCISE

A sales file consists of data such as invoices issued, products sold, and customers involved as shown as a flat data file format in Figure 1. The sales file displays field names, field sizes, and a portion of sample data. The size of a particular record/row, R, in the sales file can be determined by the sum of data fields required to store invoice-related information, product-related information, and customer-related information per record/row in the file. The total file size is thus N * R (let N be the number of records/rows in the file). Be aware that the number of

records may not be equal to the number of *invoices* stored in the sales file since an invoice may involve multiple records due to multiple products sold.

There are attributes such as: Date (8 bytes) and Invoice Number (4 bytes) for the sales-related information; Product Number (4 bytes), Description (30 bytes), Unit Price (8 bytes) and Quantity Sold (2 bytes) for the product-related information; and Customer Name (30 bytes) and Customer Address (62 bytes) for the customer-related information as shown in the un-normalized sales database in Figure 1. As a result, R = 12 + 44 + 92 = 148 bytes; thus, the total size of the database is 148N bytes.

Exercise Requirements:

Step 1

a) Calculate the total size of the data fields for a 1NF sales database.

Display the database in its first normal form by going through the first normal form process. Assume that the field size in Customer Address attribute involves 30 bytes of Street Address, 20 bytes of City, 2 bytes of State, and 10 bytes of Zip Code. Compare 1NF to an un-normalized sales database in terms of their database file size.

b) Reconcile the difference in total database file size between an un-normalized and a 1NF database.

Are there any savings in storage space in 1NF? If so, where did the savings come from?

c) Discuss about the advantages of 1NF as compared to an un-normalized database.

Are there any advantages/benefits gained in a 1NF database?

Step 2

a) Calculate the total size of the data fields for a 2NF sales database.

Describe the normalization process used to develop the database from 1NF to 2NF. Create new attributes with reasonable field size if necessary, and identify the key attribute in each table. Go through the relationships among tables and establish foreign keys if necessary. Make arbitrary numbers of records for each of any newly created tables: for example, there are 40 records in the Customer table. Compare 2NF to 1NF database with respect to their total database file sizes.

b) Reconcile the difference in total database file size between a 1NF and a 2NF database.

Are there any savings in storage space? If so, where did the savings come from?

c) Discuss about the advantages of a 2NF as compared to a 1NF database.

Are there any advantages/benefits gained in a 2NF database?

Step 3

a) Calculate the total size of the data fields for a 3NF sales database.

Describe the normalization process taken to develop the database from 2NF to 3NF. Create new attributes with reasonable field sizes if necessary, and identify the key attribute(s) in each new table. Go through the relationships among tables and establish foreign keys if necessary. Make arbitrary numbers of records for each of any newly created tables.

b) Reconcile the difference in total file size between a 2NF and a 3NF database.

Compare the total file size between 1NF/2NF and 3NF database. Are there any savings in storage space? If so, where did the savings come from?

c) Discuss about the advantages gain in a 3NF as compared to a 2NF database.

Are there any advantages/benefits gained in a 3NF database?

II. SUGGESTED SOLUTION TO THE EXERCISE REQUIREMENTS

Step 1**:** A table is in its first normal form (1NF) if each attribute contains simple values (i.e., atomic attribute) and there are no repeating attributes.

1NF focuses on the atomic of the attributes in the tables. Based on the current structure of the sales file/table, it is not in first normal form (1NF) since the attribute, Customer Address, is not atomic (i.e., containing only a tiny/simple value). In other words, the Customer Address is a multi-valued attribute consisting of multiple data values such as Street Address (30 bytes), City (20 bytes), State (2 bytes), and Zip Code (10 bytes). As a result, we must split multiple data values into each individual attribute as shown in the 1NF sales database in Figure 1.

a, b, and c

The number of records as well as size of a record in the sales table remains unchanged (i.e., R = 148 bytes) in 1NF sales database. However, the atomic attribute makes the maintenance and retrieval of the data from the Customer Address attribute much easier. For example, when we have a multi-valued attribute, i.e., Customer Address, in the sales file, the users must use certain separators (e.g., comma and/or space) for the multiple data values in the data entry. When entering data, one user might use a comma followed by a space, and another user might use a space alone as the separator between multiple data values. Due to the use of inconsistent separators or the lack of a standardized format for the multi-valued attribute (i.e., lack of control over the input format), it is difficult to maintain and/or retrieve individual data value (e.g., City information) from the database. However, 1NF resolves these problems by separating multi-valued attributes into single-valued attributes. As a result, we have greater control over the input format of the data in each of the attributes, making maintenance and retrieval of data from the database more efficient and effective.

An Exception to Multi-Valued Attributes with a Standardized Input Format and Functions

Although 1NF focuses on the atomic of the attributes, there is an exception. For example, Date is also a multi-valued attribute since it consists of Day, Month, and Year data values. However, we do not need to separate them in the storage ***as long as*** we have no frequent uses for these data values individually ***and*** we store them in a standardized "date" format so that we can identify them easily whenever we need to access them individually with the use of the Date-related functions in the database management system. We will split Date into Day, Month, and Year attributes ***only when*** we need to retrieve these individual data values more frequently than their combined/merged data value. This way, we do not need to separate them every time when we access them individually, and accessing the data values in the attributes is more efficient. In sum, we do not need to split the multi-valued attributes ***as long as*** there are standardized input formats to capture them and functions to extract them.

A Difference in the Definition of 1NF

Most AIS textbooks use a different definition of the first normal form as compared to those in the Management Information System/Computer Information System (MIS/CIS) textbooks. The definition of 1NF provided is to eliminate repeating groups of data in the rows, instead of in the columns (i.e., exactly one simple value for each attribute) as it was originally referred (Kent, 1983). Actually, the problem of repeating groups referred to in the rows will be resolved in the second normal form. However, by following such a definition from the AIS textbooks, we may not achieve the purpose of database normalization unless, to begin with, atomic attributes

are ensured in each table in the database. For example, data values of Street Address, City, State and Zip Code are combined and stored in one attribute called Customer Address with commas and/or spaces in between depending on the person who handles the data entry, as in our sales exercise in Figure 1. In this situation, we store the same City, State and Zip Code many times and may use different separators based on the preference of the person who performs the data entry. As a result, there are data redundancy and inconsistency in City, State and Zip Code data. In addition, users may not be able to find certain records using a sub-value (e.g., search by city) contained in the attribute in a query. This makes data maintenance and retrieval extremely difficult if not impossible.

Step 2**:** A table is in its 2NF if it is in 1NF and non-key attributes depend on the whole key (i.e., functional dependency).

The sales file becomes a 1NF database after splitting the Customer Address attribute. However, it is not in 2NF since it does not satisfy the fully functional dependency requirement. We need to identify a key for the table and apply 2NF criteria (i.e., full functional dependency between non-key and key attributes). The key for the 1NF sales table is the Invoice Number and Product Number attributes together, a composite key. Since there exists partial dependencies (e.g., Date is a dependent of the Invoice Number, not a dependent of the Product Number) in the table, they violate the requirement of 2NF, i.e., fully functional dependency. To resolve the partial dependencies, we split the table into multiple tables according to the partial dependencies in the table. For example, we can identify three sets of partial dependencies on Invoice Number, Product Number, and Customer Name. Therefore, we create three tables: Invoice, Product, and Customer tables, and include their related data in the tables (see Figure 2). The Invoice-Product table is created because of a many-to-many relationship between Invoice and Product tables. Figure 4 shows the relationships between the tables. The table, Zip Code, is with the Customer table in the 2NF process and is split from the Customer table in the 3NF process.

a) Calculation of Data Storage Space

Assume there are 400 records (i.e., N=400) in the sales table consisting of 200 invoices, 30 products, 40 customers in the sales file. As a result, the size of the sales file is equal to 59,200 [i.e., (12 + 44 + 92) * 400] bytes in both the un-normalized database and the 1NF form database. However, the total file size in the 2NF database is only 12,300 (i.e., 12 * 200 + 42 * 30 + 96 * 40 + [4 * 200] + [(4 + 4 + 2) * 400]) bytes.

Un-Normalized Database:
NR = [(8 + 4) + (4 + 30 + 8 + 2) + (30 + 62)] * 400
= (12 + 44 + 92) * 400
= 59,200 bytes

1NF:
NR = [(8 + 4) + (4 + 30 + 8 + 2) + (30 + 30 +20 + 2 +10)] * 400
= (12 + 44 + 92) * 400
= 59,200 bytes

2NF:
NR = Invoice table + Product table + Customer table + Invoice-Product table
= {(12 * 200) + [4 * 200]} + (42 * 30) + {(92 * 40) + [4 * 40]} + [(4 + 4 + 2) * 400]
= 12,300 bytes

b) Reconciliation

The savings in the file size, 46,900 (59,200-12,300), in 2NF database are the result of eliminating redundant data storage from 2,400 (12 * 200) bytes of Invoice-related data (i.e., Date and Invoice Number), 15,540 (42 * 370) bytes of Product-related data (i.e., Product Number, Description, and Unit Price), and 33,120 (92 * 360) bytes of Customer-related data (i.e., Customer Number, Customer Name, Street Address, City, State, and Zip Code), even though it takes 160 (4 * 40) more bytes to create the key in the Customer table (i.e., Customer Number), 800 (4 * 200] more bytes for the foreign key in the Invoice table to link with the Customer tables, and 3,200 [(4 + 4) * 400]

more bytes for the composite key in the relationship table to link the Quantity Sold to the Invoice Number and Product Number in the Invoice and Product tables, respectively.

In the un-normalized database, there are non-key attributes that are NOT dependents of the key attribute, which cause data redundancy in the table (e.g., products- and customer-related information are stored repeatedly for each invoice and different invoices). By allocating the attributes in different tables so that non-key attributes are dependents of the key attribute, that kind of data redundancy is eliminated. As a result, Invoice- (or Product- and Customer-) related information is stored only once in each of the tables.

c) Benefits

In addition to greater efficiency in storage space through eliminating data redundancy, the 2NF process allows us to update, insert, and delete data that are related to the key of a table without any difficulty. For example, we can update address information for a customer in the Customer table without any difficulty since there is only one record that stores address information for each customer as compared to many records in the 1NF database.

The 2NF also enhances data consistency in the database. It requires that data in the related informational attributes (e.g., product description, customer street address) be entered and stored to the same place, which eliminates the possibility of having inconsistent data values stored everywhere as is evident in the un-normalized data file and 1NF database. Maintenance (e.g., updating, inserting, and deleting) and retrieval of data in the database is thus made more effective.

Step 3**:** A table is in its 3NF if it is in 2NF and non-key attributes do not depend on other non-key attributes (i.e., there is no transitive dependency).

The 3NF focuses on the concept of transitive dependency. It makes sure that ***no*** non-key attribute is transitively dependent on any non-key attribute. The 2NF sales database does not satisfy such a requirement because one of the attributes, State (and City when we store a 9-digit zip code), is a dependent of another attribute, Zip Code, in the Customer table. Consequently, we must split them into two tables, Customer and Zip Code tables, as shown in Figure 3. In the 2NF, i.e., functional dependency, we eliminate any data redundancy related to the key of each table. However, it is still possible that data redundancy might exist among the non-key attributes in a table. As a result, the 3NF addresses that possibility and ensures the elimination of that kind of data redundancy.

a) Calculation of Data Storage Space

Following the previous example, assume there are 20 records in the Zip Code table (i.e., 20 different zip codes from 40 customers). We know that the size of the sales database and the 1NF database is 59,200 [i.e., (12 + 44 + 92) * 400] bytes and the 2NF database is 12,300 (i.e., 16 * 200 + 42 * 30 + 96 * 40 + 10 * 400) bytes. The size of the 3NF database is only 12,060 (i.e., 16 * 200 + 42 * 30 + 74 * 40 + 10 * 400 + 32 * 20) bytes. The calculations are as follows:

2NF:
NR = Invoice table + Product table + Customer table + Sales-Product table
= {(12 * 200) + [4 * 200]} + (42 * 30) + {(92 * 40) + [4 * 40]} + [(4 + 4 + 2) * 400]
= (16 * 200) + (42 * 30) + (96 * 40) + (10 * 400)
= 12,300 bytes

3NF:
NR = Invoice table + Product table + Customer table + Sales-Product table + Zip Code table
= (16 * 200) + (42 * 30) + (64 * 40) + [10 * 40] + (10 * 400) + (32 * 20)
= 12,060 bytes

b) Reconciliation

The savings in the file size, 240 (12,300 - 12,060) bytes, comes from the elimination of redundant data storage of 880 [(96 – 74) * 40] bytes of City, State, and Zip Code data in the Customer table with an additional 640 (32 * 20) bytes used for the storage of data in the Zip Code table. Or, the savings consist of 20 (40 – 20) records of redundant information of State, City, and Zip Code in the Customer table (i.e., 640 bytes) with an additional storage for the foreign key in the Customer table (i.e., 400 bytes).

c) Benefits

The 3NF process further enhances data consistency in the database. Even though the 2NF is a necessary condition for ensuring data consistency in the tables and in the database (that it eliminates repeating groups of data in the rows, as is defined in most AIS textbooks), it is not sufficient due to the possibility of data redundancy (thus data inconsistency) among non-key attributes. The 3NF guarantees data consistency in the database: it requires that ***all*** non-key attributes are dependents of the key and ***no*** non-key attribute is a dependent of any other non-key attribute. This process totally eliminates data redundancy in the database before establishing the relationships and creating foreign keys (the only allowed redundancy in the database).

III. TEACHING NOTES

We provided copies of Figures 1-4 and went through every single step of the requirements with students in class. We also asked students to apply the calculation and reconciliation approach to a database normalization exercise from an AIS textbook. We conducted a survey to find out about students' satisfaction about this reinforcement approach as compared to the traditional method. Students found this calculation and reconciliation approach very easy to follow (mean=5.89 on a Likert-type scale, 1 to 7 with 7 being "Strongly Agree"), and it helped them to understand the database normalization concept (mean=6.11).

Instructors can incorporate the calculation and reconciliation approach to their own materials as related to database normalization in order to enhance students' understanding of the benefits of the normalization concept. In that case, apply simple numbers to the size of the tables, both the field size and record size (e.g., 10, 20, 30, etc.). Remember that the record sizes of the tables grow over time, and that some tables grow fast than others.

Instructors can also use this paper as stand-alone material to teach the concept of database normalization after students have acquired an understanding about creating entities and relationships. The suggested solutions in Section II cover the definition of normal forms and provide a description of the application of the calculation and reconciliation approach, which is sufficient to be used as stand-alone material.

After students have gone through the development of each norm form, remind them to include appropriate foreign keys in the tables before they start to perform the calculation and reconciliation, as required in Steps 2b and 3b; to do this correctly, students must first figure out the required relationships among all of the tables.

When reviewing Steps 2b and 3b with students, instructors should encourage them to focus on the *causes* for the savings in storage space: on functional and transitive dependencies. In other words, these savings are the result of the elimination of data redundancies using the concept of functional and transitive "dependencies", i.e., the purpose of 2NF and 3NF. Data related to any unique thing (e.g., a particular Resource/Object, Event, Agent/Person, Location/Place, or Concept) is only stored once at a particular space in a table in the database. That is how the relational database eliminates data inconsistency.

IV. CONCLUSION

This paper presents a reinforcement approach, an alternative teaching method to the traditional conceptual approach, to enhance accounting students' understanding of the concept of database normalization. The reinforcement approach involves calculations and reconciliations which accounting students are familiar and comfortable with. Using the calculation and reconciliation approach, students are given the opportunity to visually

account for the changes made to the database structure in the database normalization process. When students relate the changes to the benefits of each of the database normal forms progressed, students are able to effectively grasp the purpose of the database normalization.

AUTHOR INFORMATION

Dr. TJ Wang is an associate professor at Governors State University. He is the corresponding author of this paper. He has written a few papers related to relational database and won the Best Paper awards in the AIS Educator Annual Conference in 2002, 2004 and 2005. He has published some of the papers in the *Review of Business Information Systems* and the *Journal of College Teaching and Learning*. He has taught at Robert Morris University, University of Wisconsin-Milwaukee, Bellevue Community College, Rutgers University and New Jersey Institute of Technology.

Hui Du, MBA, PhD, is an assistant professor at the University of Houston – Clear Lake . Her research interests include the impact of new technologies on accounting and accounting professionals, accounting information systems, systems control and auditing, and corporate governance. She teaches Intermediate Accounting I and Accounting Information Systems. She has published in *Journal of Accounting and Public Policy*, *International Journal of Auditing*, *Journal of Accountancy*, and *The CPA Journal*. She can be reached by email: duhui1@uhcl.edu.

Dr. Lehmann earned a PhD in 2001, and her CISA in 2008. She has published in journals such as *Behavioral Research in Accounting, Journal of Information Systems, Internal Auditor, Journal of Theoretical Accounting Research, Journal of Education for Business,* and *Advances in Accounting Education.* She is also a co-author for the textbook *Accounting Information Systems: A Practitioner Approach* (Atomic Dog Publishing), currently being revised for its 6th edition. She teaches Accounting Information Systems and IT Audit. Dr. Lehmann is an active member of the Information Systems Audit and Control Association (ISACA), as well as the Institute of Internal Auditors (IIA).

REFERENCES

1. Bagranoff, Nancy A., Simkin, Mark G., and Carolyn Strand Norman. 2008. *Core Concepts of Accounting Information Systems*. 10th ed. NY: John Wiley and Sons.
2. Codd, Edgar F. 1970. "A Relational Model of Data for Large Shared Data Banks." *Communications of the ACM*, Vol. 13, No. 6, pp. 377-387.
3. Dimitrov, Dimiter M., and Phillip D. Rumrill, Jr. 2003. "Pretest-posttest designs and measurement of change." *Work*, Vol. 20, No. 2, pp. 159-165.
4. Elmasri, Ramez and Shamkant B. Navathe. 2006. *Fundamentals of Database Systems*. 5th ed. Reading, MA: Addison-Wesley.
5. Jones, and Rama. 2006. *Accounting Information Systems: A Business Process Approach*. 2nd ed. Mason, OH: South-Western College Pubs.
6. Kent, William. 1983. "A Simple Guide to Five Normal Forms in Relational Database Theory." *Communications of the ACM*, Vol. 26, No. 2, pp. 120-125.
7. Kung, Hsiang-Jui, and Hui-Lien Tung. 2006. "An Alternative Approach to Teaching Database Normalization: A Simple Algorithm and An Interactive e-Learning Tool." *Journal of Information Systems Education*, Vol. 17 (Fall), Issue 3, pp. 315-325.
8. Romney, Marshall B., and Paul John Steinbart. 2009. *Accounting Information Systems*. 11th ed. Upper Saddle River, NJ: Pearson/Prentice Hall.
9. Silberschatz, A., Korth, H. F., and Sudarshan, S. 2002. *Database System Concepts*. 4th ed. Boston, MA: McGraw-Hill.
10. Vaassen, E.H.J. 2002. *Accounting Information Systems: A Managerial Approach*. NY: John Wiley and Sons.

FIGURE 1. Un-Normalized vs. 1NF Sales Database

STEP 1a: File Size

Un-Normalized Sales Database

	- 8 - Date	- 4 - Invoice Number	- 4 - Product Number	- 30 - Description	- 8 - Unit Price	- 2 - Qty Sold	- 30 - Customer Name	- 62 - Customer Address
N Records	10/25/06	1001	A320	MP3	10.00	4	ABC	1 Major Road, New City, NJ 07154-0345
	10/25/06	1001	B101	iPod	100.00	2	ABC	1 Major Road, New City, NJ 07154-0345
	10/25/06	1002	C220	MP4	50.00	3	PBS	1546 State Street, Chicago, IL 63458-2165
	……..	……..	……..	……..	……..	…	……..	……..

Total file size = 148 N bytes

1NF Sales Database

	- 8 - Date	- 4 - Invoice Number	- 4 - Product Number	- 30 - Description	- 8 - Unit Price	- 2 - Qty Sold	- 30 - Customer Name	- 30 - Street Address	- 20 - City	- 2 - State	- 10 - Zip Code
N Records	10/25/06	1001	A320	MP3	10.00	4	ABC	1 Major Road	New City	NJ	07154-0345
	10/25/06	1001	B101	iPod	100.00	2	ABC	1 Major Road	New City	NJ	07154-0345
	10/25/06	1002	C220	MP4	50.00	3	PBS	1546 State …	Chicago	IL	63458-2165
	……..	……..	……..	……..	……..	…	……..	……..	……..	….	……..

Total file size = 148 N bytes

STEP 1b: Reconciliation

- No difference in file sizes (i.e., 148 N bytes)

STEP 1c: Benefits

- Same storage efficiency (i.e., 148 N bytes)
- More effective in maintaining and retrieving customer address information

FIGURE 2. 1NF vs. 2NF Sales Database

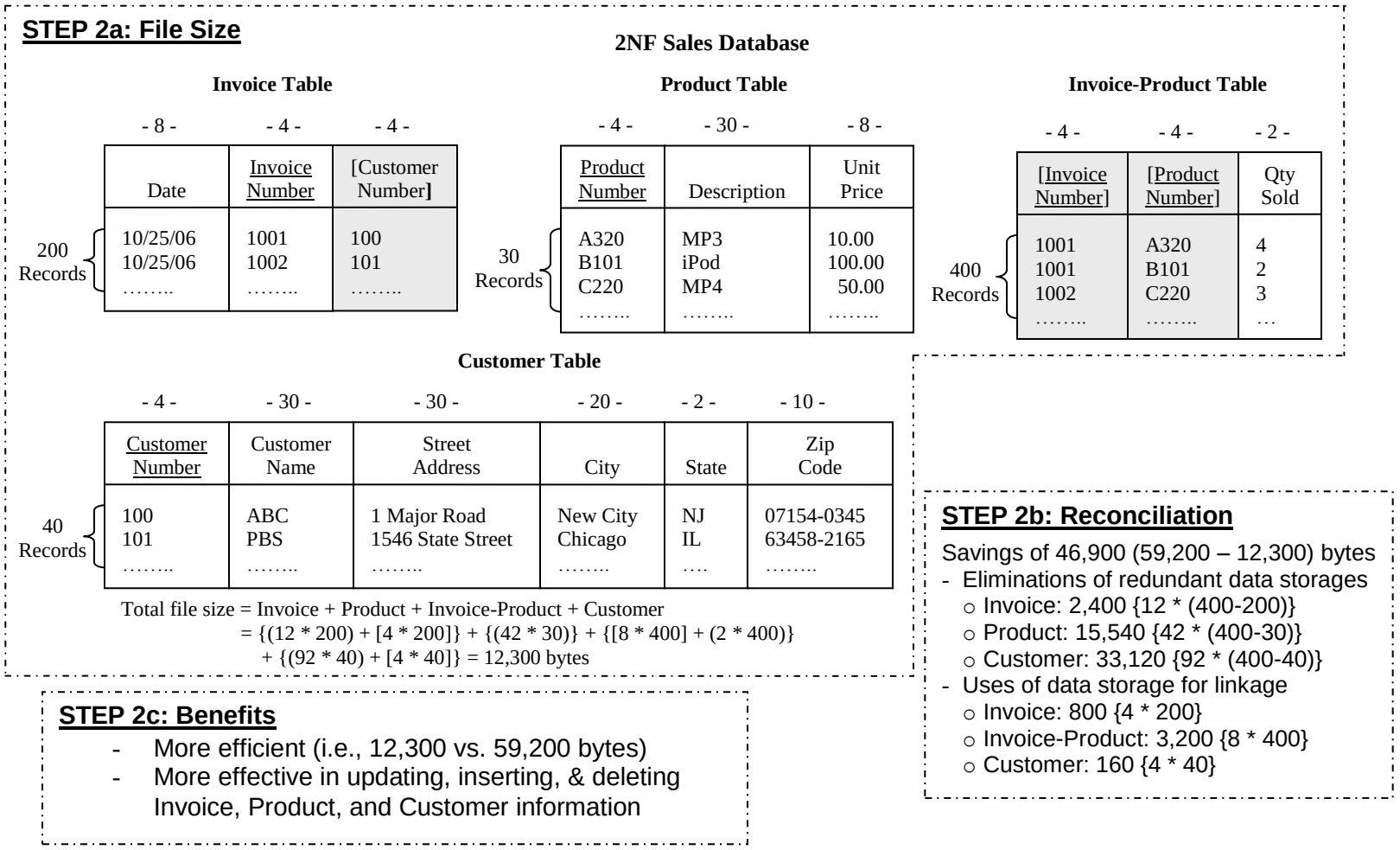

FIGURE 3. 2NF vs. 3NF Sales Database

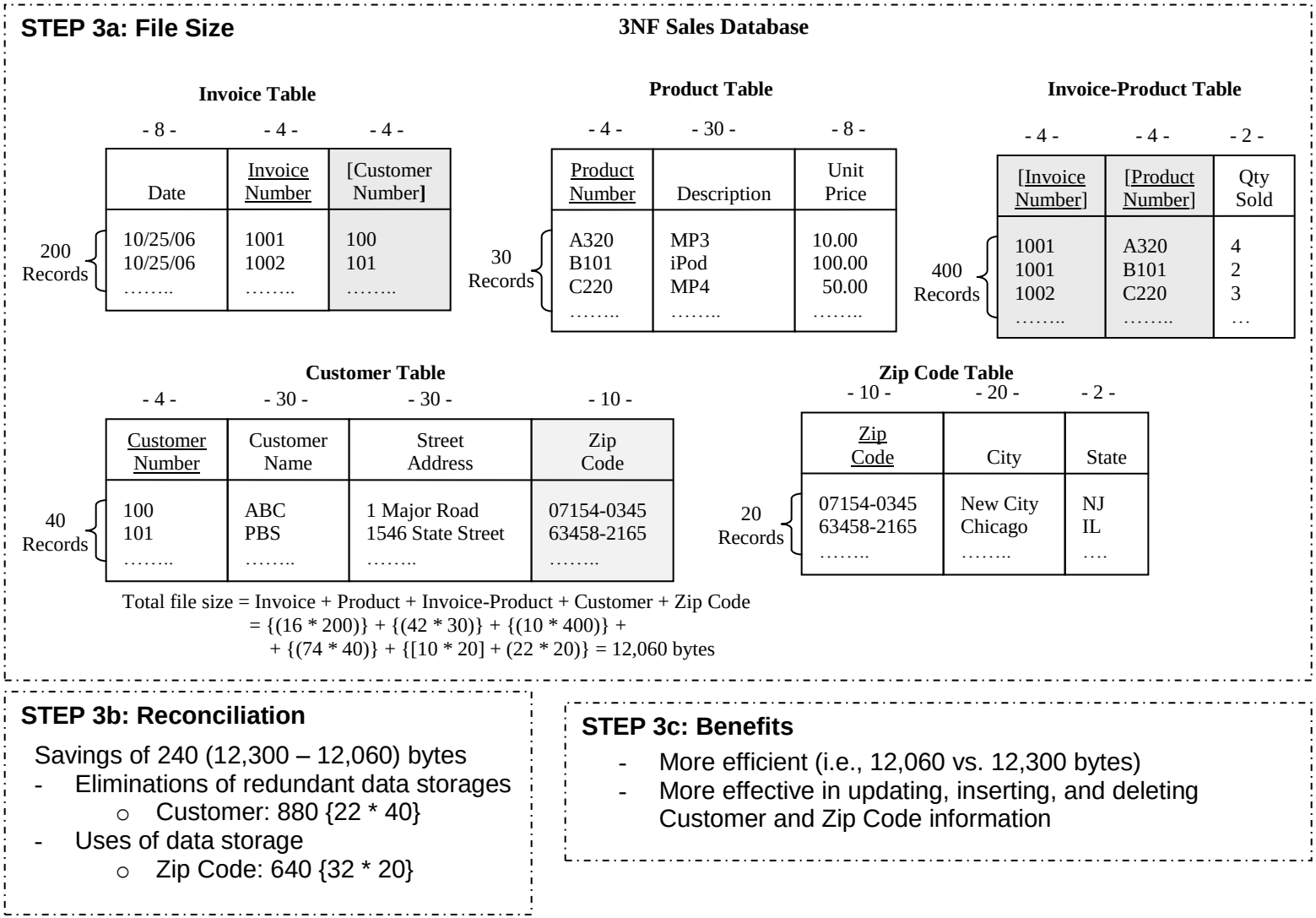

FIGURE 4. Simplified Data Model for the 3NF Sales Database