Normal Forms

from 9th ed Modern Database Management by Jeffrey Hoffer, Mary Prescott, Heikki Topi
Pearson Prentice Hall 2009

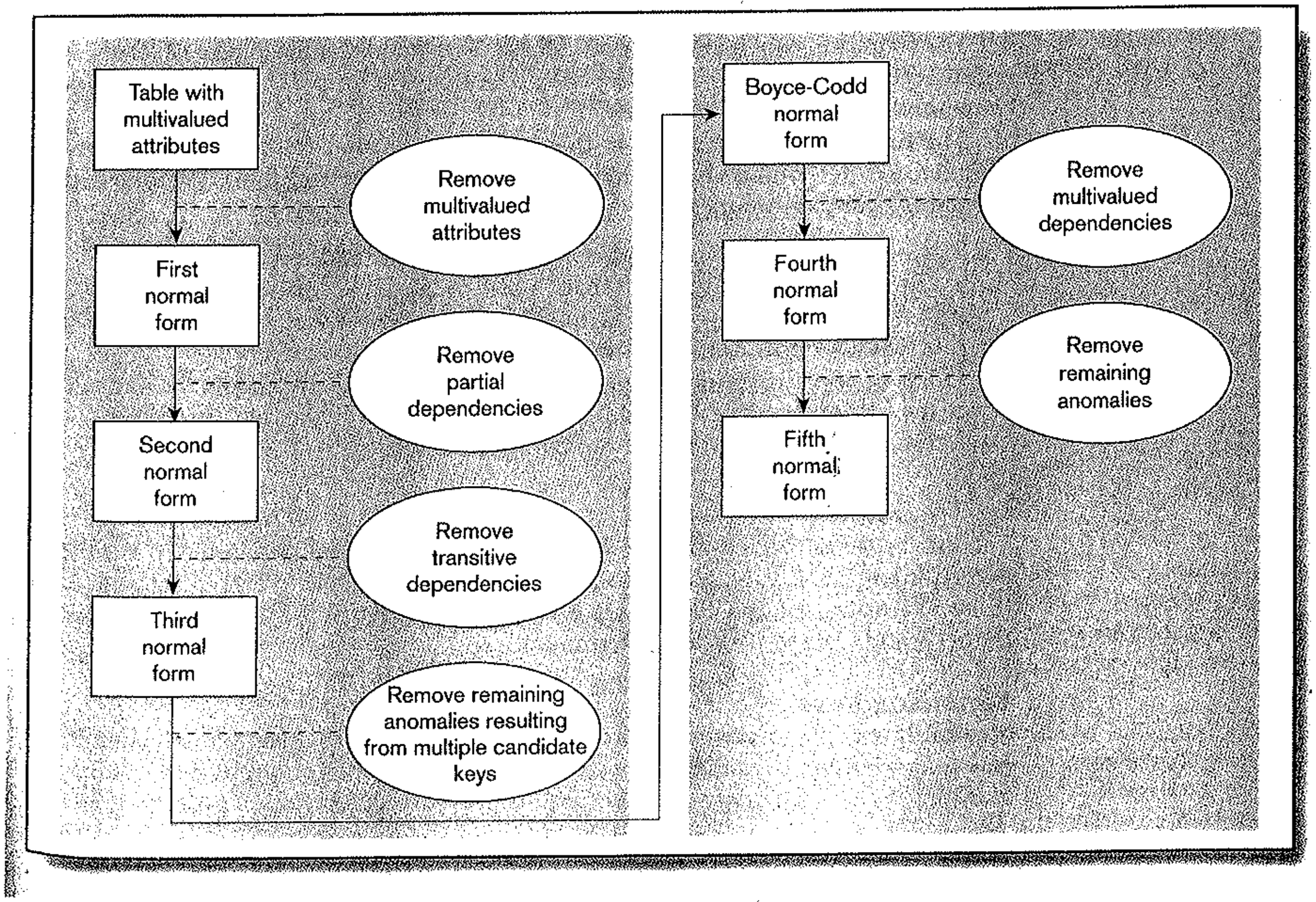
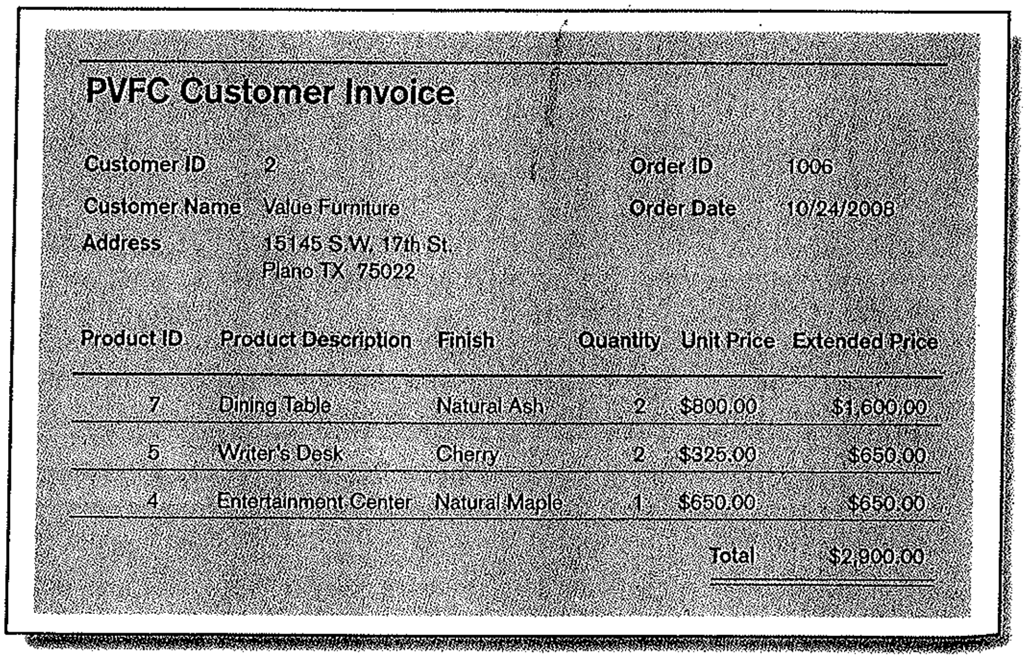


Table with multivalued attributes
Remove multivalued attributes
First normal form
Remove partial dependencies
Second normal form
Remove transitive dependencies
Third normal form
Remove remaining anomalies resulting from multiple candidate keys
Boyce-Codd normal form
Remove multivalued dependencies
Fourth normal form
Remove remaining anomalies
Fifth normal form



PVFC Customer Invoice

Customer ID	2	Order ID	1006
Customer Name	Value Furniture	Order Date	10/24/2008
Address	15145 S.W. 17th St. Plano TX 75022		

Product ID	Product Description	Finish	Quantity	Unit Price	Extended Price
7	Dining Table	Natural Ash	2	$800.00	$1,600.00
5	Writer's Desk	Cherry	2	$325.00	$650.00
4	Entertainment Center	Natural Maple	1	$650.00	$650.00
				Total	$2,900.00

Figure 5-25 INVOICE data (Pine Valley Furniture Company)

Order_ID	Order_ Date	Customer_ ID	Customer_ Name	Customer_ Address	Product_ID	Product_ Description	Product_ Finish	Unit_ Price	Ordered_ Quantity
1006	10/24/2008	2	Value Furniture	Plano, TX	7	Dining Table	Natural Ash	800.00	2
					5	Writer's Desk	Cherry	325.00	2
					4	Entertainment Center	Natural Maple	650.00	1
1007	10/25/2008	6	Furniture Gallery	Boulder, CO	11	4–Dr Dresser	Oak	500.00	4
					4	Entertainment Center	Natural Maple	650.00	3

Figure 5-25. Notice that data for a second order (Order_ID 1007) are included in Figure 5-25 to clarify further the structure of this data.

Step 1: Convert to First Normal Form

First normal form (1NF) A relation that has a primary key and in which there are no repeating groups.

A relation is in **first normal form (1NF)** if the following two constraints both apply:

1. There are no repeating groups in the relation (thus, there is a single fact at the intersection of each row and column of the table).
2. A primary key has been defined, which uniquely identifies each row in the relation.

Remove Repeating Groups

As you can see, the invoice data in Figure 5-25 contains a repeating group for each product that appears on a particular order. Thus, Order_ID 1006 contains three repeating groups, corresponding to the three products on that order.

In a previous section, we showed how to remove repeating groups from a table by filling relevant data values into previously vacant cells of the table (see Figures 5-2a and 5-2b). Applying this procedure to the invoice table yields the new table (named INVOICE) shown in Figure 5-26.

Figure 5-26 INVOICE relation (1NF) (Pine Valley Furniture Company)

Order_ID	Order_ Date	Customer_ ID	Customer_ Name	Customer_ Address	Product_ID	Product_ Description	Product_ Finish	Unit_ Price	Ordered_ Quantity
1006	10/24/2008	2	Value Furniture	Plano, TX	7	Dining Table	Natural Ash	800.00	2
1006	10/24/2008	2	Value Furniture	Plano, TX	5	Writer's Desk	Cherry	325.00	2
1006	10/24/2008	2	Value Furniture	Plano, TX	4	Entertainment Center	Natural Maple	650.00	1
1007	10/25/2008	6	Furniture Gallery	Boulder, CO	11	4–Dr Dresser	Oak	500.00	4
1007	10/25/2008	6	Furniture Gallery	Boulder, CO	4	Entertainment Center	Natural Maple	650.00	3

Select the Primary Key

There are four determinants in INVOICE, and their functional dependencies are the following:

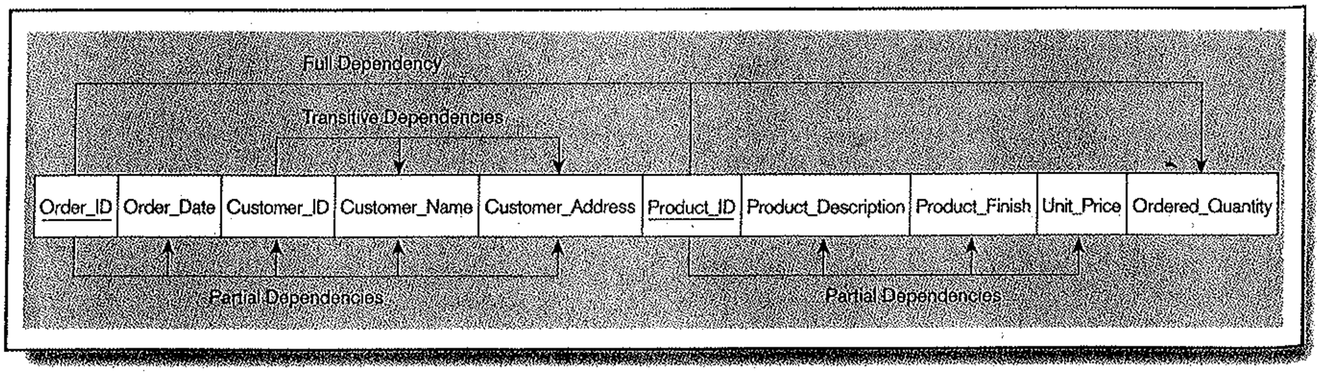
Order_ID → Order_Date, Customer_ID, Customer_Name, Customer_Address
Customer_ID → Customer_Name, Customer_Address
Product_ID → Product_Description, Product_Finish, Unit_Price
Order_ID, Product_ID → Ordered_Quantity

Anomalies in 1NF

Although repeating groups have been removed, the data in Figure 5-26 still contains considerable redundancy. For example, the Customer_ID, Customer_Name, and Customer_Address for Value Furniture are recorded in three rows (at least) in the table. As a result of these redundancies, manipulating the data in the table can lead to anomalies such as the following:

1. *Insertion anomaly* If the customer calls and requests another product be added to his Order_ID 1007, a new row must be inserted in which the order date and all

Figure 5-27 Functional dependency diagram for INVOICE



of the customer information must be repeated. This may lead to data entry errors (e.g., the customer name may be entered as "Valley Furniture").

2. *Deletion anomaly* If the customer calls and requests that the Dining Table be deleted from her Order_ID 1006, this row must be deleted from the relation and we lose the information concerning this item's finish (Natural Ash) and price ($800.00).
3. *Update anomaly* If Pine Valley Furniture (as part of a price adjustment) increases the price of the Entertainment Center (Product_ID 4) to $750.00, this change must be recorded in all rows containing that item. (There are two such rows in Figure 5-26.)

Step 2: Convert to Second Normal Form

Second normal form (2NF) A relation in first normal form in which every nonkey attribute is fully functionally dependent on the primary key.

Partial functional dependency A functional dependency in which one or more nonkey attributes are functionally dependent on part (but no all) of the primary key.

We can remove many of the redundancies (and resulting anomalies) in the INVOICE relation by converting it to second normal form. A relation is in **second normal form (2NF)** if it is in first normal form and contains no partial dependencies. A **partial functional dependency** exists when a nonkey attribute is functionally dependent on part (but not all) of the primary key. As you can see, the following partial dependencies exist in Figure 5-27:

```
Order_ID → Order_Date, Customer_ID, Customer_Name,
           Customer_Address
Product_ID → Product_Description, Product_Finish, Unit_Price
```

The first of these partial dependencies (for example) states that the date on an order is uniquely determined by the order number and has nothing to do with the Product_ID.

To convert a relation with partial dependencies to second normal form, the following steps are required:

1. Create a new relation for each primary key attribute (or combination of attributes) that is a determinant in a partial dependency. That attribute is the primary key in the new relation.
2. Move the nonkey attributes that are dependent on this primary key attribute (or attributes) from the old relation to the new relation.

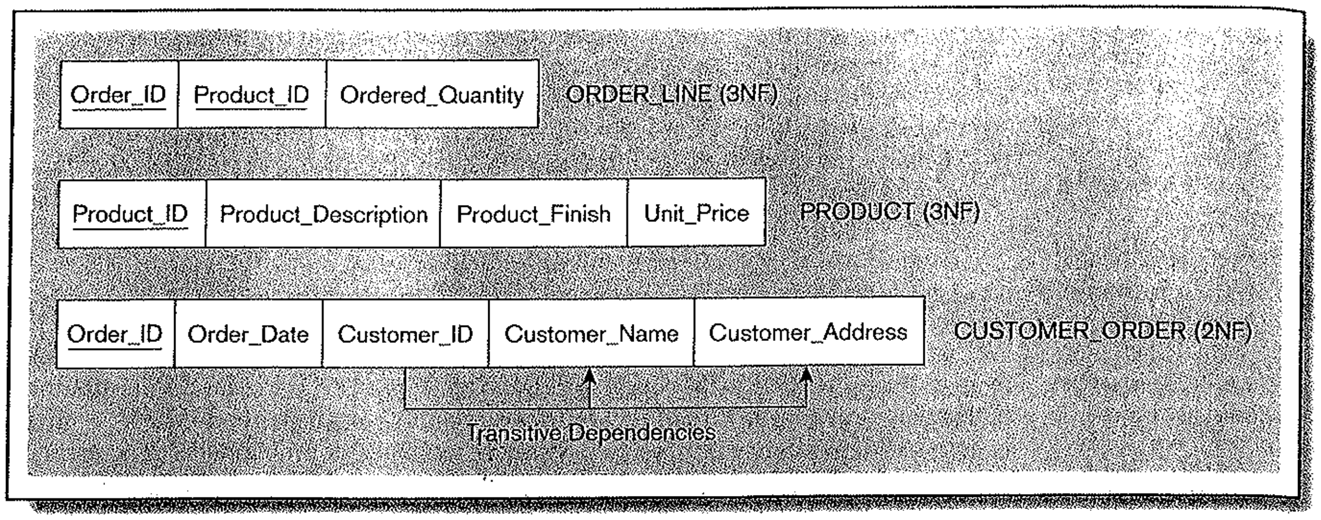
The results of performing these steps for the INVOICE relation are shown in Figure 5-28. Removal of the partial dependencies results in the formation of two new relations: PRODUCT and CUSTOMER_ORDER. The INVOICE relation is now left with just the primary key attributes (Order_ID and Product_ID) and Ordered_Quantity, which is functionally dependent on the whole key. We rename this relation ORDER_LINE, because each row in this table represents one line item on an order.

As indicated in Figure 5-28, the relations ORDER_LINE and PRODUCT are in third normal form. However, CUSTOMER_ORDER contains transitive dependencies and therefore (although in second normal form) is not yet in third normal form.

A relation that is in first normal form will be in second normal form if any one of the following conditions applies:

1. The primary key consists of only one attribute (such as the attribute Product_ID in the PRODUCT relation in Figure 5-28). By definition, there cannot be a partial dependency in such a relation.
2. No nonkey attributes exist in the relation (thus all of the attributes in the relation are components of the primary key). There are no functional dependencies in such a relation.
3. Every nonkey attribute is functionally dependent on the full set of primary key attributes (such as the attribute Ordered_Quantity in the ORDER_LINE relation in Figure 5-28).

Figure 5-28 Removing partial dependencies



Step 3: Convert to Third Normal Form

Third normal form (3NF) A relation that is in second normal form and has no transitive dependencies.

Transitive dependency A functional dependency between two (or more) nonkey attributes.

A relation is in **third normal form** (3NF) if it is in second normal form and no transitive dependencies exist. A **transitive dependency** in a relation is a functional dependency between two (or more) nonkey attributes. For example, there are two transitive dependencies in the CUSTOMER_ORDER relation shown in Figure 5-28:

Customer_ID → Customer_Name, and Customer_ID → Customer_Address

In other words, both customer name and address are uniquely identified by the Customer_ID, but Customer_ID is not part of the primary key (as we noted earlier).

Transitive dependencies create unnecessary redundancy that may lead to the type of anomalies discussed earlier. For example, the transitive dependency in CUSTOMER_ORDER (Figure 5-28) requires that a customer's name and address be reentered every time a customer submits a new order, regardless of how many times they have been entered previously. You have no doubt experienced this type of annoying requirement when ordering merchandise online, visiting a doctor's office, or any number of similar activities.

Removing Transitive Dependencies

You can easily remove transitive dependencies from a relation by means of a three-step procedure:

1. For each nonkey attribute (or set of attributes) that is a determinant in a relation, create a new relation. That attribute (or set of attributes) becomes the primary key of the new relation.
2. Move all of the attributes that are functionally dependent on the attribute from the old to the new relation.
3. Leave the attribute (which serves as a primary key in the new relation) in the old relation to serve as a foreign key that allows you to associate the two relations.

The results of applying these steps to the relation CUSTOMER_ORDER are shown in Figure 5-29. A new relation named CUSTOMER has been created to receive the components of the transitive dependency. The determinant Customer_ID becomes the primary key of this relation, and the attributes Customer_Name and Customer_Address are moved to the relation. CUSTOMER_ORDER is renamed ORDER, and the attribute Customer_ID remains as a foreign key in that relation. This

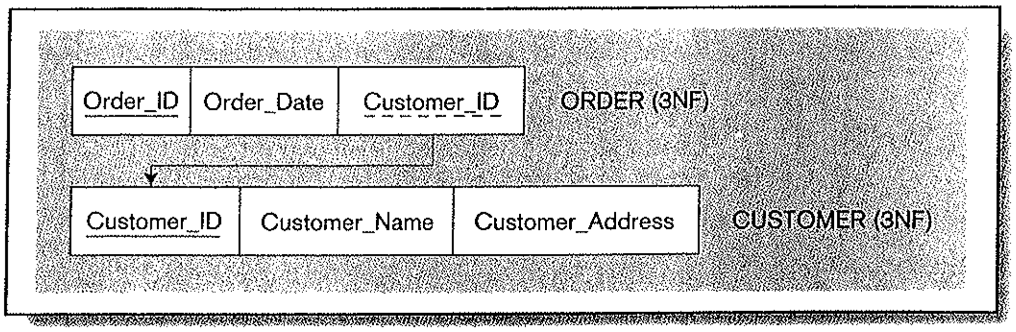


Figure 5-29
Removing transitive dependencies

allows us to associate an order with the customer who submitted the order. As indicated in Figure 5-29, these relations are now in third normal form.

Normalizing the data in the INVOICE view has resulted in the creation of four relations in third normal form: CUSTOMER, PRODUCT, ORDER, and ORDER_LINE. A relational schema showing these four relations and their associations (developed using Microsoft Visio) is shown in Figure 5-30. Note that Customer_ID is a foreign key in ORDER and Order_ID and Product_ID are foreign keys in ORDER_LINE. (Foreign keys are shown in Visio for logical, but not conceptual, data models.) Also note that minimum cardinalities are shown on the relationships even though the normalized relations provide no evidence of what the minimum cardinalities should be. Sample data for the relations might include, for example, a customer with no orders, thus providing evidence of the optional cardinality for the relationship Places. However, even if there were an order for every customer in a sample data set, this would not prove mandatory cardinality. Minimum cardinalities must be determined from business rules not illustrations of reports, screens, and transactions. The same statement is true for specific maximum cardinalities (for example, a business rule that no order may contain more than 10 line items).

Determinants and Normalization

We demonstrated normalization through 3NF in steps. There is an easy shortcut, however. If you look back at the original set of four determinants and the associated

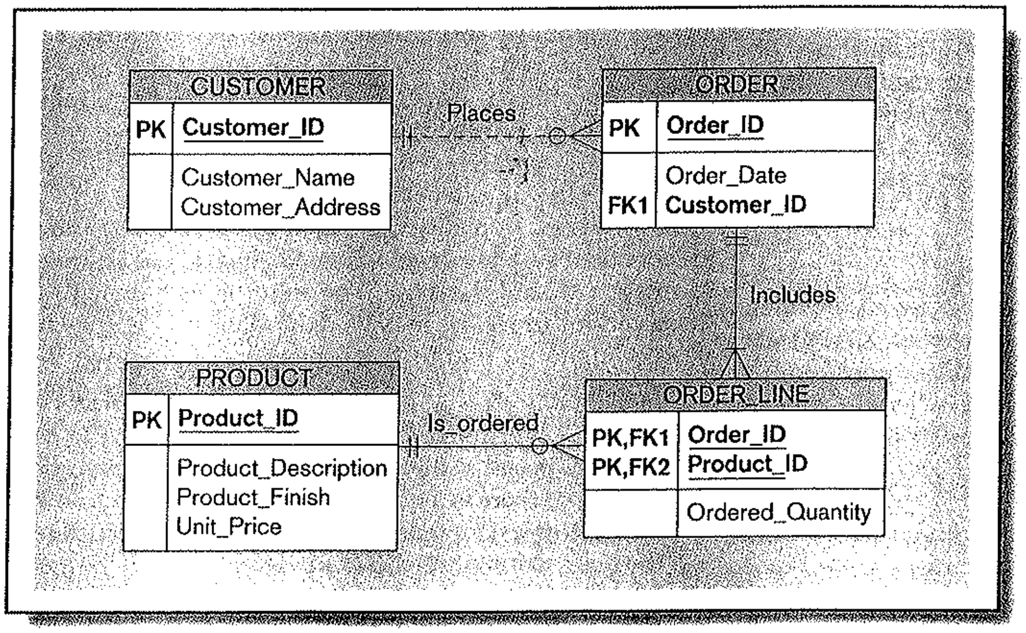


Figure 5-30
Relational schema for INVOICE date (Microsoft Visio notation)

Appendix B

Advanced Normal Forms

In Chapter 5, we introduced the topic of normalization and described first through third normal forms in detail. Relations in third normal form (3NF) are sufficient for most practical database applications. However, 3NF does not guarantee that all anomalies have been removed. As indicated in Chapter 5, several additional normal forms are designed to remove these anomalies: Boyce-Codd normal form, fourth normal form, and fifth normal form (see Figure 5-22). We describe Boyce-Codd normal form and fourth normal form in this appendix.

Boyce-Codd Normal Form

When a relation has more than one candidate key, anomalies may result even though that relation is in 3NF. For example, consider the STUDENT_ADVISOR relation shown in Figure B-1. This relation has the following attributes: SID (student ID), Major, Advisor, and Maj_GPA. Sample data for this relation are shown in Figure B-1a, and the functional dependencies are shown in Figure B-1b.

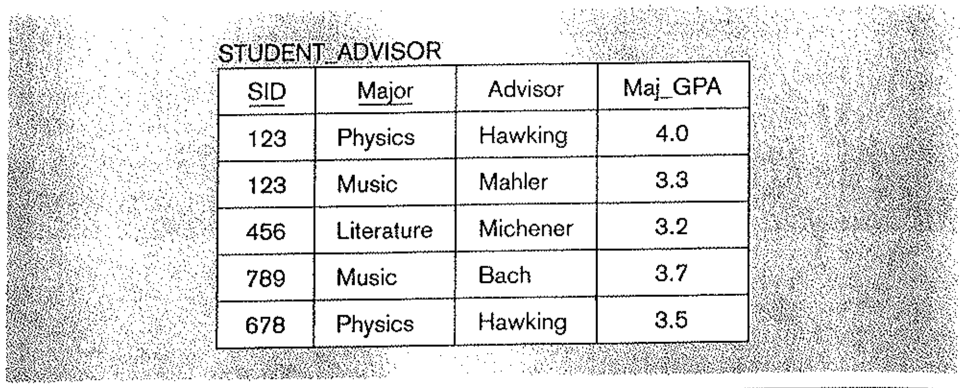
As shown in Figure B-1b, the primary key for this relation is the composite key consisting of SID and Major. Thus, the two attributes Advisor and Maj_GPA are functionally dependent on this key. This reflects the constraint that although a given student may have more than one major, for each major a student has exactly one advisor and one GPA.

There is a second functional dependency in this relation: Major is functionally dependent on Advisor. That is, each advisor advises in exactly one major. Notice that this is not a transitive dependency. In Chapter 5, we defined a transitive dependency as a functional dependency between two nonkey attributes. In contrast, in this example a key attribute (Major) is functionally dependent on a nonkey attribute (Advisor).

Anomalies in STUDENT_ADVISOR

The STUDENT_ADVISOR relation is clearly in 3NF, because there are no partial functional dependencies and no transitive dependencies. Nevertheless, because of the

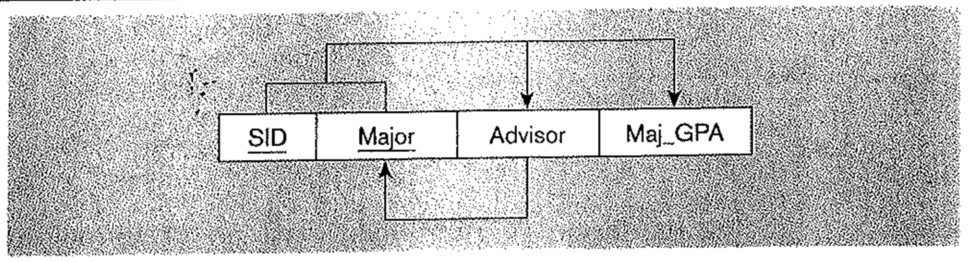
Figure B-1
Relation in 3NF, but not in BCNF
(a) Relation with sample data



STUDENT_ADVISOR

SID	Major	Advisor	Maj_GPA
123	Physics	Hawking	4.0
123	Music	Mahler	3.3
456	Literature	Michener	3.2
789	Music	Bach	3.7
678	Physics	Hawking	3.5

(b) Functional dependencies in STUDENT_ADVISOR



functional dependency between Major and Advisor, there are anomalies in this relation. Consider the following examples:

1. Suppose that in Physics the advisor Hawking is replaced by Einstein. This change must be made in two (or more) rows in the table (update anomaly).
2. Suppose we want to insert a row with the information that Babbage advises in Computer Science. This, of course, cannot be done until at least one student majoring in Computer Science is assigned Babbage as an advisor (insertion anomaly).
3. Finally, if student number 789 withdraws from school, we lose the information that Bach advises in Music (deletion anomaly).

Definition of Boyce-Codd Normal Form (BCNF)

Boyce-Codd normal form (BCNF)
A relation in which every determinant is a candidate key.

The anomalies in STUDENT_ADVISOR result from the fact that there is a determinant (Advisor) that is not a candidate key in the relation. R. F. Boyce and E. F. Codd identified this deficiency and proposed a stronger definition of 3NF that remedies the problem. We say a relation is in **Boyce-Codd normal form (BCNF)** if and only if every determinant in the relation is a candidate key. STUDENT_ADVISOR is not in BCNF because although the attribute Advisor is a determinant, it is not a candidate key (only Major is functionally dependent on Advisor).

Converting a Relation to BCNF

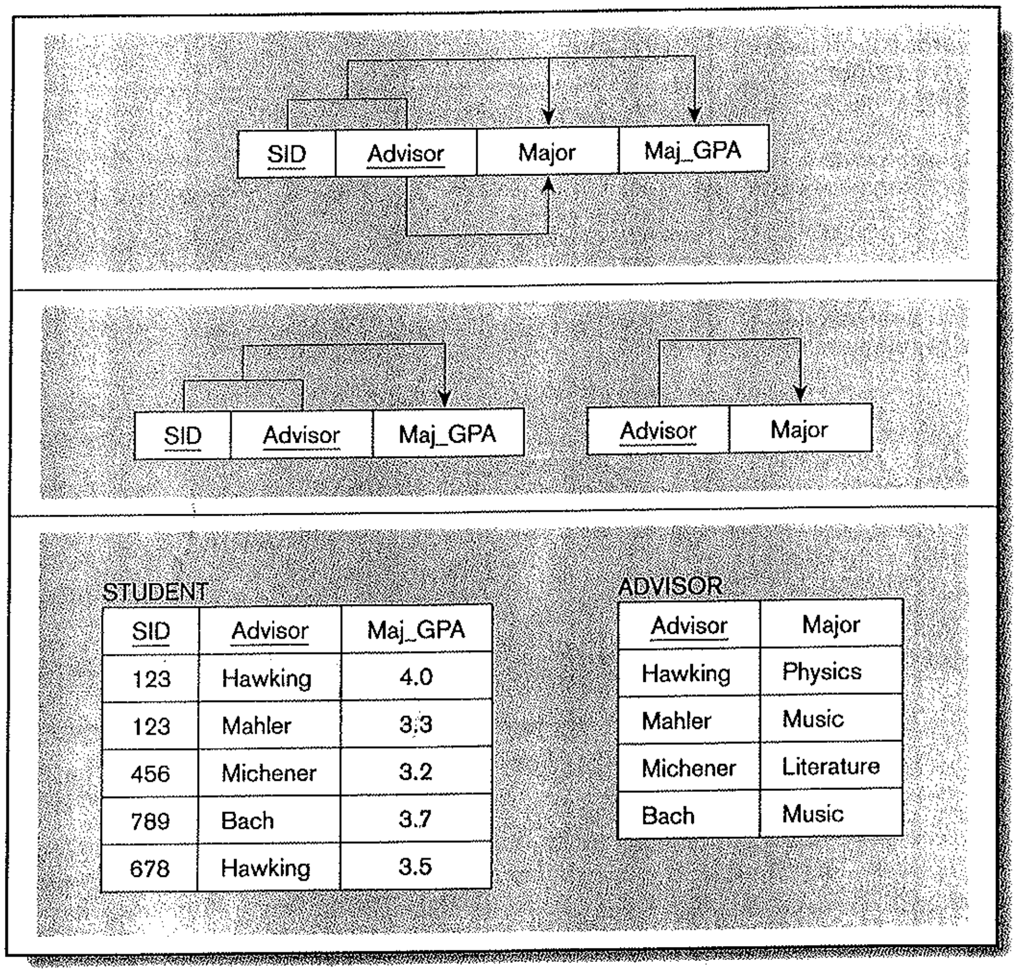
A relation that is in 3NF (but not BCNF) can be converted to relations in BCNF using a simple two-step process. This process is shown in Figure B-2.

In the first step, the relation is modified so that the determinant in the relation that is not a candidate key becomes a component of the primary key of the revised relation. The attribute that is functionally dependent on that determinant becomes a nonkey attribute. This is a legitimate restructuring of the original relation because of the functional dependency.

Figure B-2
Converting a relation to BCNF relations
(a) Revised STUDENT_ADVISOR relation (2NF)

(b) Two relations in BCNF

(c) Relations with sample data



STUDENT

SID	Advisor	Maj_GPA
123	Hawking	4.0
123	Mahler	3.3
456	Michener	3.2
789	Bach	3.7
678	Hawking	3.5

ADVISOR

Advisor	Major
Hawking	Physics
Mahler	Music
Michener	Literature
Bach	Music

The result of applying this rule to STUDENT_ADVISOR is shown in Figure B-2a. The determinant Advisor becomes part of the composite primary key. The attribute Major, which is functionally dependent on Advisor, becomes a nonkey attribute.

If you examine Figure B-2a, you will discover that the new relation has a partial functional dependency (Major is functionally dependent on Advisor, which is just one component of the primary key). Thus the new relation is in first (but not second) normal form.

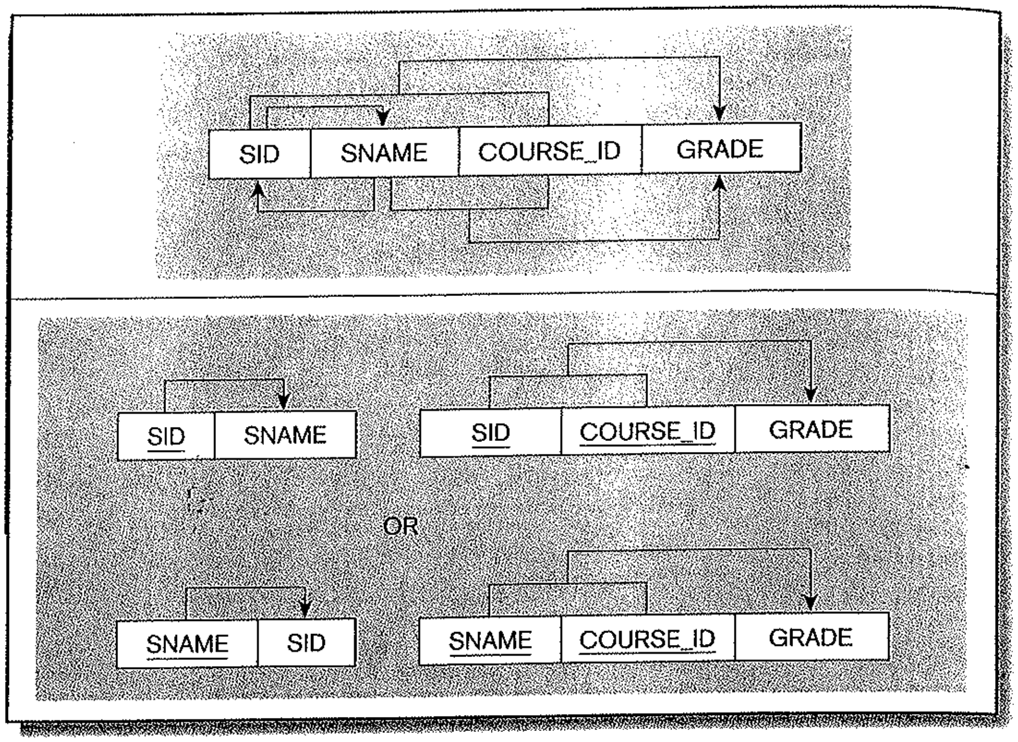
The second step in the conversion process is to decompose the relation to eliminate the partial functional dependency, as we learned in Chapter 5. This results in two relations, as shown in Figure B-2b. These relations are in 3NF. In fact, the relations are also in BCNF, because there is only one candidate key (the primary key) in each relation. Thus, we see that if a relation has only one candidate key (which therefore becomes the primary key), then 3NF and BCNF are equivalent.

The two relations (now named STUDENT and ADVISOR) with sample data are shown in Figure B-2c. You should verify that these relations are free of the anomalies that were described for STUDENT_ADVISOR. You should also verify that you can recreate the STUDENT_ADVISOR relation by joining the two relations STUDENT and ADVISOR.

Another common situation in which BCNF is violated is when there are two (or more) overlapping candidate keys of the relation. Consider the relation in Figure B-3a. In this example, there are two candidate keys, (SID,COURSE_ID) and (SNAME,COURSE_ID), in which COURSE_ID appears in both candidate keys. The problem with this relationship is that we cannot record student data (SID and SNAME) unless the student has taken a course. Figure B-3b shows two possible solutions, each of which creates two relations that are in BCNF.

Figure B-3
Converting a relation with overlapping candidate keys to BCNF
(a) Relation with overlapping candidate keys

(b) Two alternative pairs of relations in BCNF



FOURTH NORMAL FORM

When a relation is in BCNF, there are no longer any anomalies that result from functional dependencies. However, there may still be anomalies that result from multivalued dependencies (defined in the next section). For example, consider the user view shown in Figure B-4a. This user view shows for each course the instructors who teach that course and the textbooks that are used. (These appear as repeating groups in the view.) In this table view, the following assumptions hold:

1. Each course has a well-defined set of instructors (e.g., Management has three instructors).
2. Each course has a well-defined set of textbooks that are used (e.g., Finance has two textbooks).
3. The textbooks that are used for a given course are independent of the instructor for that course (e.g., the same two textbooks are used for Management regardless of which of the three instructors is teaching Management).

In Figure B-4b, this table view has been converted to a relation by filling in all of the empty cells. This relation (named OFFERING) is in 1NF. Thus, for each course, all possible combinations of instructor and text appear in OFFERING. Notice that the primary key of this relation consists of all three attributes (Course, Instructor, and Textbook). Because there are no determinants other than the primary key, the relation is actually in BCNF. Yet it does contain much redundant data that can easily lead to update anomalies. For example, suppose that we want to add a third textbook (author: Middleton) to the Management course. This change would require the addition of three new rows to the relation in Figure B-4b, one for each Instructor (otherwise that text would apply to only certain instructors).

Figure B-4 Data with multivalued dependencies

(a) View of courses, instructors, and textbooks

COURSE STAFF AND BOOK ASSIGNMENTS

Course	Instructor	Textbook
Management	White Green Black	Drucker Peters
Finance	Gray	Jones Chang

(b) Relation in BCNF

OFFERING

Course	Instructor	Textbook
Management	White	Drucker
Management	White	Peters
Management	Green	Drucker
Management	Green	Peters
Management	Black	Drucker
Management	Black	Peters
Finance	Gray	Jones
Finance	Gray	Chang

Multivalued Dependencies

Multivalued dependency The type of dependency that exists when there are at least three attributes (e.g., A, B, and C) in a relation, with a well-defined set of B and C values for each A value, but those B and C values are independent of each other.

The type of dependency shown in this example is called a **multivalued dependency,** which exists when there are at least three attributes (e.g., A, B, and C) in a relation, and for each value of A there is a well-defined set of values of B and a well-defined set of values of C. However, the set of values of B is independent of set C, and vice versa.

To remove the multivalued dependency from a relation, we divide the relation into two new relations. Each of these tables contains two attributes that have a multivalued relationship in the original relation. Figure B-5 shows the result of this decomposition for the OFFERING relation of Figure B-4b. Notice that the relation called TEACHER contains the Course and Instructor attributes, because for each course there is a well-defined set of instructors. Also, for the same reason, TEXT contains the attributes Course and Textbook. However, there is no relation containing the attributes Instructor and Course because these attributes are independent.

Figure B-5
Relations in 4NF

TEACHER

Course	Instructor
Management	White
Management	Green
Management	Black
Finance	Gray

TEXT

Course	Textbook
Management	Drucker
Management	Peters
Finance	Jones
Finance	Chang

Fourth normal form (4NF) A relation in BCNF that contains no multivalued dependencies.

A relation is in **fourth normal form (4NF)** if it is in BCNF and contains no multivalued dependencies. You can easily verify that the two relations in Figure B-5 are in 4NF and are free of the anomalies described earlier. Also, you can verify that you can reconstruct the original relation (OFFERING) by joining these two relations. In addition, notice that there are fewer data in Figure B-5 than in Figure B-4b. For simplicity, assume that Course, Instructor, and Textbook are all of equal length. Because there are 24 cells of data in Figure B-4b and 16 cells of data in Figure B-5, there is a space savings of 33 percent for the 4NF tables.

Higher Normal Forms

At least two higher-level normal forms have been defined: fifth normal form (5NF) and domain-key normal form (DKNF). Fifth normal form deals with a property called "lossless joins." According to Elmasri and Navathe (2000), 5NF is not of practical significance because lossless joins occur very rarely and are difficult to detect. For this reason (and also because 5NF has a complex definition), we do not describe 5NF in this text.

Domain-key normal form is an attempt to define an "ultimate normal form" that takes into account all possible types of dependencies and constraints (Elmasri and Navathe, 2000). Although the definition of DKNF is quite simple, its practical value is minimal. For this reason, we do not describe DKNF in this text.

For more information concerning these two higher normal forms see Elmasri and Navathe (2000) and Dutka and Hanson (1989).

Appendix Review

Key Terms

- Boyce-Codd normal form (BCNF)
- Fourth normal form (4NF)
- Multivalued dependency

References

Dutka, A., and H. Hanson. 1989. *Fundamentals of Data Normalization*. Reading, MA: Addison-Wesley.

Elmasri, R., and S. Navathe. 2000. *Fundamentals of Database Systems*, 3rd ed. Reading, MA: Addison-Wesley.

Web Resources

www.bkent.net/Doc/simple5.htm A simple but understandable guide to first through fifth normal forms.