Week 5 – Lab #4 – Correcting Logical Models through Normalization.

Table of Contents

Logical Models and Normalization

Welcome to your fourth lab in IS380! During lab #3 we saw how we can transform a conceptual model into a logical model. This gave us a higher level of details and opens the opportunity to create our actual database and explore it using SQL.

Before we go into the actual implementation of a relational database, we must first evaluate the quality of our logical models through the technique of normalization. This process will enable us to detect flaws in our original model and our transformation, and, more importantly, it will minimize redundancy which in turn minimizes the risks against the integrity of our data.

We start this lab by conducting a review of what normalization is, and what the different normal forms (steps in the normalization process) are. We will then conduct 3 exercises to practice our skills at normalizing tables into well structured relations.

Lab Goals (continues form Lab#3)

- **SLO1**- To describe the differences between conceptual and logical models.
- **SLO2**- To transform conceptual models into logical models (i.e. transform EERDs into relations).
- **SLO3**- To create effective logical models with tables and relational integrity constraints.
- **SLO4**- To use the normalization process to correct poor logical models into well-structured relations.

What You Will Need to Begin

- Visit draw.io (https://www.draw.io) to gain access to a modeling tool either through its web version or by downloading a copy into your own computer.
- Please complete the modeling exercises in draw.io and copy your models (with your name visible) into this document. Screenshots are an acceptable choice for copying the models.
- Once completed, submit the exercise (this document) into a corresponding DropBox folder within BeachBoard.

Part 2 – Normalization and Logical Models.

What is normalization?

A technique used to divide tables into smaller tables that are connected through relationships. Specifically, it is:

"The goals of normalization are to:

- Be able to characterize the level of redundancy in a relational schema.
- Provide mechanisms for transforming schemas in order to remove redundancy."
  - Adrienne Watt "Database Design" 2nd Edition.

In short, the goal of normalization is to reduce redundancy in our databases ensuring the integrity of the data in the process. It does this by dividing tables with many attributes into smaller more focused tables reducing the need to repeat information throughout our database.

Normalization is meant to ensure that the database has no major flaws, but it is important to note that if we developed an appropriate conceptual model and followed the transformation steps from Lab #3, we WILL get a logical model that is normalize to some significant extent. Therefore, it is important to start our database design by:

1) examining the business rules,

2) creating an appropriate conceptual model based on them,

3) transforming the conceptual model into a logical model following the guidelines from lab#3

4) confirming that it is normalized at least to Normal form #3, or if it is not

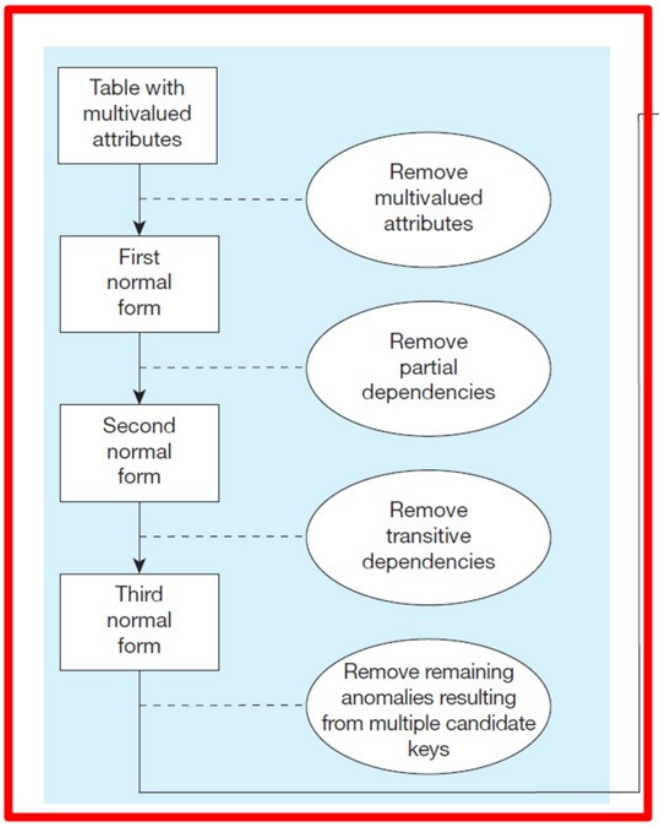
5) normalizing the database to normal form 3.

Normal forms refer to the specific steps that we follow to minimize redundancy within the normalization technique. Each step consists of reviewing the database for a particular type of error/problem and correcting it in an appropriate form. Practitioners typically aim to normalize their databases to third normal as it provides significant benefits with minimal impact to performance (the more tables and relationships we must describe a particular scenario, the slower performance will typically be).

Understanding Normal Forms

Now that we mentioned the concept of normal forms, it is important that we review what they are:

NOTE: Keep in mind that each normal form builds on top of the previous one.



First Normal Form (also refer to as 1st normal form or 1NF)

For a table to be in first normal form, it must meet 2 requirements:

- It must have no multivalued attributes.
- Every attribute value (think of a cell) must be atomic (i.e. it cannot be divided further).

Tables are not typically considered relations unless they are in first normal form.

SOLUTION: We need to eliminate composite and multivalued attributes by breaking them into different rows. For example:

Examples gathered from “studytonight”. Please visit them form more details at:

- https://www.studytonight.com/dbms/second-normal-form.php

roll_no	name	subject
101	Akon	OS, CN
103	Ckon	Java
102	Bkon	C, C++

In the picture above, the subject is a multivalued attribute (Ckon knows only 1 subject which is Java while Akon and Bkon know 2 and all of them know different subjects). We solve this by separating each value within the subject into a different row so that we end up with atomic values (1 value that cannot be divided further within each cell).

roll_no	name	subject
101	Akon	OS
101	Akon	CN
103	Ckon	Java
102	Bkon	C
102	Bkon	C++

While we had to repeat values in the other cells (including the roll_no which is arguably the identifier), we now have atomic values in ALL cells. We will solve the repetition automatically as we progress through the next 2 normal forms.

Second Normal Form (also refer to as 2nd normal form or 2NF)

For a table to be in 2nd normal form, it must first be in 1st normal form and then it must satisfy the following requirements:

- Every attribute that is not a key (primary key or foreign key) must be functionally dependent on the entire primary key (i.e. it must be fully functionally dependent on the primary key).
- Moreover, the value of an attribute cannot be dependent on only part of the primary key (that is called a partial functional dependency).

A **functional dependency** is simply means that the value for a give attribute in 1 row depends on some other attribute (typically the primary key).

A **full functional dependency** means that the value for an attribute depends on the entire primary key (if it is a composite it MUST depend on all components of the primary key and not just 1 or some of them).

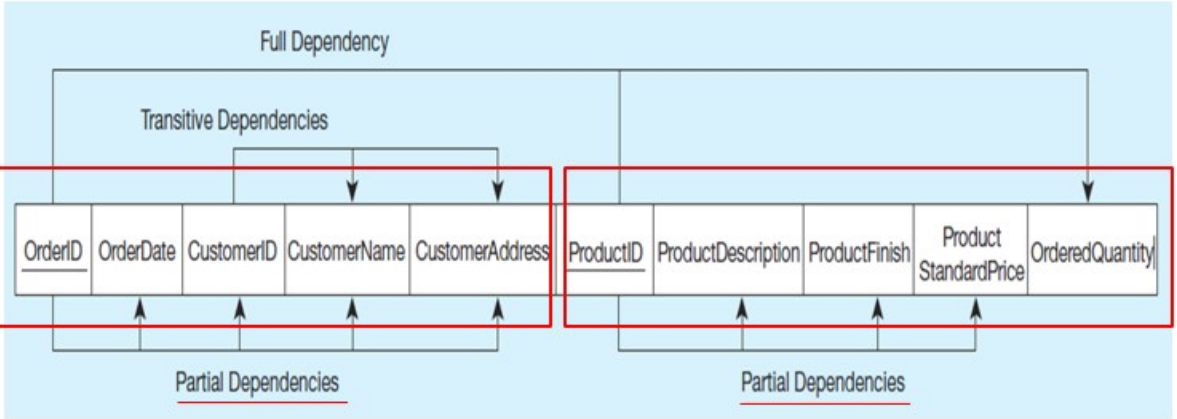
A **partial functional dependency** refers to an attribute's value being dependent on only part of the primary key rather than on its entirety (i.e. if the composite primary key of an associative table is made of 2 FKs, any value for an attribute in that table must be dependent on both components of the primary key rather than just on 1).

RECAP: Therefore, for a table to be in 2nd normal form, it must be in 1st normal form and all attributes must be fully functional dependent on the primary key. That is not a problem for simple primary keys (i.e. with 1 column), but if we have a composite primary key we must ensure that the values for any attribute recorded in that table are dependent on all components of the primary key rather than just 1. If that is not the case, we have a partial functional dependency.

To solve a partial functional dependency, we simply separate the attributes that are only partially dependent on the primary key and place them on their own table which will continue to be referenced through a foreign key in the original table.

EXAMPLE:

Figure 4-27 from the book:

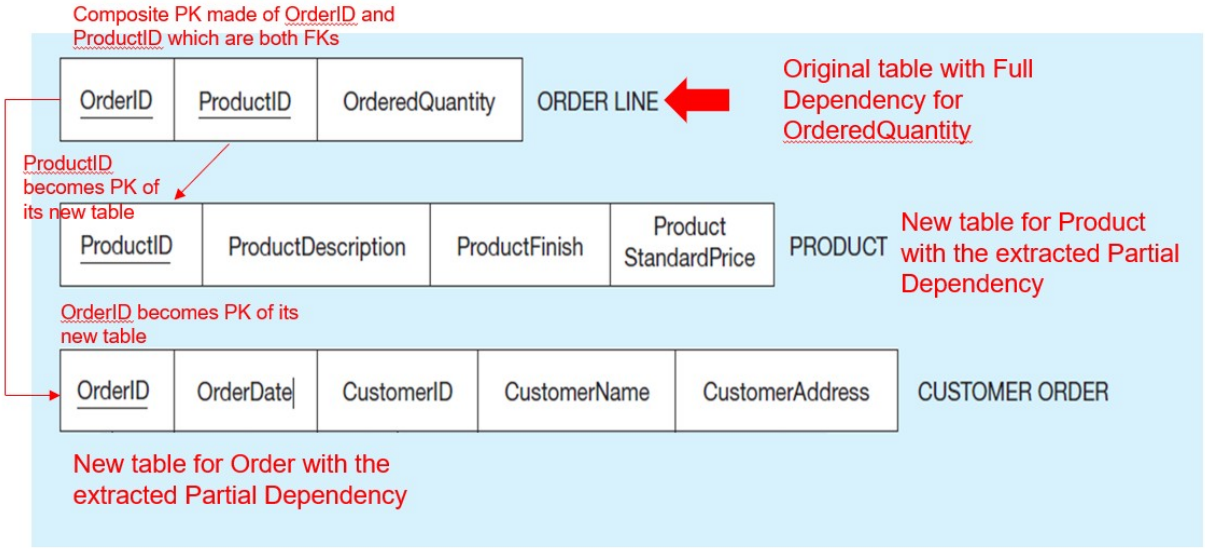


In this example from the book, we have a table with 10 attributes including 2 that combined form the primary key for the table (OrderID + ProductID). The "Ordered Quantity" is dependent on both components of the primary key (i.e. a full dependency) since how many items were ordered depends entirely on both which item and which order we are referring to.

Nevertheless, the values for all other attributes are dependent on either the OrderID or the ProductID rather than the combination of both making them a partial dependency. That is, OrderDate, CustomerID, CustomerName, & CustomerAddress depend on the OrderID and are entirely independent from the Product that was ordered (the ProductID). At the same time, ProductDescription, ProductFinish, and Product Standard Price are all dependent on the Product that we are referring to (identified by the productid) and have nothing to do with the OrderID.

To solve this, we keep the attributes that form a full dependency in the table, but extract the partial dependencies into new tables. For example, OrderDate, CustomerID, CustomerName, and CustomerAddress are extracted together with the key they depend on (the OrderID) into a new table. The same happens for the ProductID partial dependency.

Finally, we keep a primary key/foreign key relationships between the newly formed tables and the original one. We can clearly see this below where Order has its own table with PK OrderID, Product has its own table with PK ProductID, and the original table with the composite key becomes an associative entity between them with a PK formed by combining the foreign keys OrderID and ProductID.



Third Normal Form

For a table to be in 3rd normal form, it must be in 2nd normal form and:

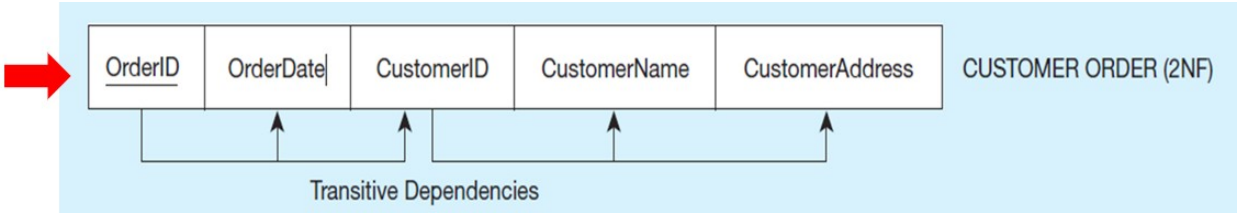
- Have **no** transitive dependencies.

Transitive dependencies refer to attributes whose value is directly dependent on a different attribute and indirectly on the primary key rather than being determined by the primary key itself. Solving transitive dependencies is no different from how we solved partial dependencies. That is, we take the columns/attributes that are transitive (that depend on a column other than the primary key) and extract them into their own table.

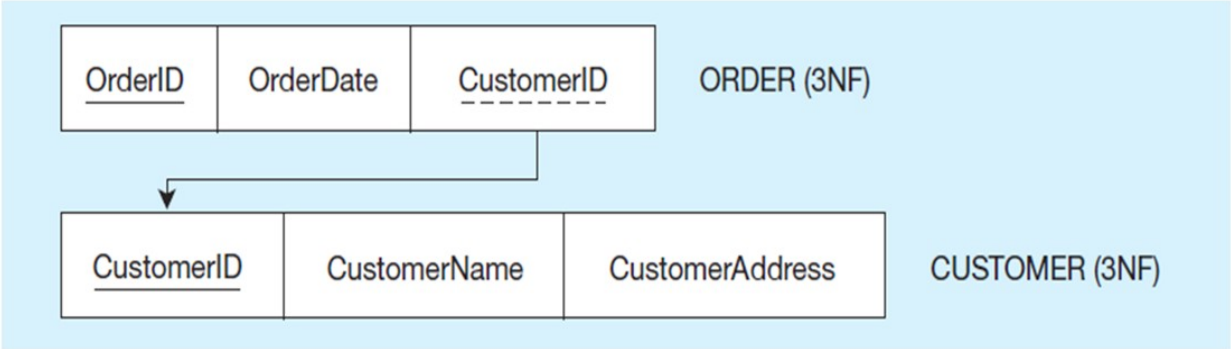
For example: In the example for 2NF we create a table for Customers' orders which included: Order ID, Order Date, Customer ID, Customer Name, and Customer Address. While we determine that these attributes depended on the OrderID alone rather than on both the OrderID and the ProductID forming a partial dependency on OrderID, the reality is a bit more complex than that.

Specifically, while all attributes can be correctly identified by the Order ID, the attributes of CustomerName, and CustomerAddress are actually dependent on who the customer is (meaning, they

are dependent on the CustomerID) and are only indirectly related to the OrderID (since the CustomerID is the one related to the OrderID. That makes it a transitive dependency.



To solve it, we simply extract the transitive dependency from the Customer Order table, and connect both through a primary key/foreign key relationship. Therefore, we take the CustomerName, CustomerAddress, and CustomerID and create a new table with the for the Customer. In the original Customer Order table, we keep the attributes/columns dependent on the OrderID including a Foreign Key (FK) connecting it to the newly created Customer table.



Part 2: Practicing Normalization

Exercise 1: Transforming Attributes

The manager of a company dinner club would like to have an information system that assists him to plan the meals and to keep track of who attends the dinners, and so on.

Because the manager is not an IS expert, the following table is used to store the information. As a member can attend many dinners and a member will not attend more than 1 dinner on the same date, **the primary key of the following table is Member ID and Dinner ID.** Dinners can have many courses, from one-course dinner to as many courses as the chef desired.

Member = customer

Dinner = meal

Venue = location

Food = specific food item

PK = MEMBERID + DINNERID

TEST: Does each attribute depend on both components of the Primary Key, or just a portition?

The manager of a company dinner club would like to have an information system that assists him to plan the meals and to keep track of who attends the dinners, and so on.

Because the manager is not an IS expert, the following table is used to store the information. As a member can attend many dinners and a member will not attend more than 1 dinner on the same date, the primary key of the following table is Member Num + Dinner Num. Dinners can have many courses, from one-course dinner to as many courses as the chef desired.

MEMBER
NUM
MEMBER

NAME
MEMBER
ADDRESS
DINNER
NUM
DINNER
DATE
VENUE
CODE
VENUE
DESCRIPTION
FOOD
CODE
FOOD
DESCRIPTION
214 Peter Wong
325
Meadow

Park
D0001 15-Mar-10 B01
Grand Ball
Room
EN3 Stu ed crab
DE8
Chocolate
mousse
235 Mary Lee
123 Rose
Court
D0002 15-Mar-10 B02 Pe)t Ball
Room EN5 Marinated steak
DE8
Chocolate
mousse

250 Peter Wong 9 Nine Ave D0003
20-Mar-10 C01 Café SO1 Pumpkin
soup
EN5 Marinated steak
DE2 Apple pie
235 Mary Lee
123 Rose
Court
D0003 20-Mar-10 C01 Café SO1
Pumpkin soup
EN5 Marinated steak
DE2 Apple pie
300 Paul Lee
123 Rose
Court
D0004 20-Mar-10 E10 Pe)t Ball
Room SA2 Apple pie
* This table has only 5 records.

a. Use proper notation to write down the above table. Use "Member Dinner" as the table name.

b. Convert the above Member Dinner table into 1st Normal Form table.

c. Assuming you can identify the functional dependencies from the table; draw a functional dependencies diagram for the 1st NF table.

d. Develop a set of 3NF tables. Show every step of normalization along the way.

MemberID	MemberName	Member Address	**DinnerID**	DinnerDate	VenueCode	VenueDescription	Food Code	FoodDescription
214	Peter Wong	325 Meadow Park	D0001	02/02/2020	L01	Grand_Ball_Room	EN3	Stu ed crab
							DEB	Beef
235	Mary Lee	123 Bellfower	D0002	02/02/2020	L02	Café	EN5	Chocolate Mousse
							DEB	Beef
250	John Doe	9923 Orange	D0003	03/03/20	L01	Grand_Ball_Room	SO1	Marinated Steak
							EN5	Chocolate Mousse
							DE2	Apple Pie
235	Mary Lee	123 Bellfower	D0004	03/03/2020	L02	Café	S01	Pumpkin Soup
							SA2	Marinated Steak
							DE2	Apple Pie
300	Paul Lee	123 Bellfower	D0004	03/03/2020	L03	Petit_Ball_Room	SA2	Marinated Steak

MemberID	MemberName	Member Address	DinnerID	DinnerDate	VenueCode	VenueDescription	Food Code	FoodDescription
214	Peter Wong	325 Meadow Park	D0001	02/02/2020	L01	Grand_Ball_Room	EN3	Stu ed crab
214	Peter Wong	325 Meadow Park	D0001	02/02/2020	L01	Grand_Ball_Room	DEB	Beef
235	Mary Lee	123 Bellfower	D0002	02/02/2020	L02	Café	EN5	Chocolate Mousse
235	Mary Lee	123 Bellfower	D0002	02/02/2020	L02	Cafe	DEB	Beef
250	John Doe	9923 Orange	D0003	03/03/20	L01	Grand_Ball_Room	SO1	Marinated Steak
250	John Doe	9923	D000	03/03/2	L01	Grand_Ball	EN5	Chocolate

		Orange	3	0		_Room		Mousse
250	John Doe	9923 Orange	D0003	03/03/20	L01	Grand_Ball_Room	DE2	Apple Pie
235	Mary Lee	123 Bellfower	D0004	03/03/2020	L02	Café	S01	Pumpkin Soup
235	Mary Lee	123 Bellfower	D0004	03/03/2020	L02	Café	SA2	Marinated Steak
235	Mary Lee	123 Bellfower	D0004	03/03/2020	L02	Café	DE2	Apple Pie
300	Paul Lee	123 Bellfower	D0004	03/03/2020	L03	Petit_Ball_Room	SA2	Marinated Steak

1. Is the above table considered a relation? Why or why not? Is it in any normal form? If so, which one?

2. Transform the table above into first normal form 1NF. (To do this, check if there are multivalued attributes and transform the table to get rid of them)

    a. Identify the dependencies and which type they are (full dependencies, partial dependencies, transitive dependencies.

3. Transform the table above into second normal form 2NF. (To do these separate partial dependencies into separate tables).
4. Transform the table above into third normal form 3NF. (To do this, remove the transitive dependencies by creating separate tables and relate them with the common attribute)

Exercise 2: Normalizing Grade Reports

Figure 4-4 shows a relation called GRADE REPORT for a university. Your assignment is as follows:

Table 4-4 Grade Report Relation

Grade Report								
StudentID	StudentName	CampusAddress	Major	CourseID	CourseTitle	Instructor Name	Instructor Location	Grade
168300458	Williams	208 Brooks	IS	IS 350	Database Mgt	Codd	B 104	A
168300458	Williams	208 Brooks	IS	IS 465	Systems Analysis	Parsons	B 317	B
543291073	Baker	104 Phillips	Acctg	IS 350	Database Mgt	Codd	B 104	C
543291073	Baker	104 Phillips	Acctg	Acct 201	Fund Acctg	Miller	H 310	B
543291073	Baker	104 Phillips	Acctg	Mkgt 300	Intro Mktg	Bennett	B 212	A

a. Draw a logical model for the table above, and graph and explain (with arrows) the functional dependencies in the relation.

b. In what normal form is this relation?

c. Transform the GRADE REPORT table into relations in 3NF.

d. Draw a model for your 3NF relations and show the primary key/foreign key relationships.

Exercise 3: Normalizing a Shipping Manifest

Figure 4-5 shows a relation for a shipping manifest.

Shipment ID:	00-0001	Shipment Date:	01/10/2018
Origin:	Boston	Expected Arrival:	01/14/2018
Destination:	Brazil		
Ship Number:	39	Captain:	002-15
			Henry Moore

Item Number	Type	Description	Weight	Quantity	TOTALWEIGHT
3223	BM	Concrete	500	100	50,000
		Form			
3297	BM	Steel	87	2,000	174,000
		Beam			
				Shipment Total:	224,000

NOTE: This is all a single table.

a. Draw a relational schema and diagram the functional dependencies in the relation.

b. In what normal form is this relation?

c. Decompose MANIFEST into a set of 3NF relations.

d. Draw a relational schema for your 3NF relations and show the referential integrity constraints.

e. Draw your answer to part d using Microsoft Visio (or any other tool specified by your instructor).