Theory of Normal Forms
Examples

Database Anomalies

Consider the following relation Purchases:

Purchases(ReceiptNo,Ordinal, PDate, Customer, Item, Price)

Each tuple of the relation stores information about a single item purchased by a customer. The information stored is the receipt number, the position of the item on the receipt (item scanned first has Ordinal = 1, next item — Ordinal = 2, etc.), the date of the purchase, the name of the customer, the name of the purchased item and its price. Consider the following fragment of the Purchases table.

ReceiptNo	Ordinal	PDate	Customer	Item	Price
1234	1	01/21/2008	"Fausto Klosner"	"Chocolate Chip Cookie"	0.95
1234	2	01/21/2008	"Fausto Klosner"	"Rye Bread"	2.50
3042	1	01/23/2008	"Susie Gladney"	"Rye Bread"	2.50
3042	2	01/23/2008	"Susie Gladney"	"Opera Cake"	15.00
3403	1	01/26/2008	"Susie Gladney"	"Ganache Cookie"	1.30
5612	1	01/26/2008	"Dean Dews"	"Chocolate Chip Cookie"	0.95
1564	1	01/21/2008	"Fausto Klosner"	"Ganache Cookie"	1.30
1564	2	01/21/2008	"Fausto Klosner"	"Coffee"	1.50

Example 1: Redundancy. Some information stored in the table Purchases is redundant. For example, the receipt number (ReceiptNo) uniquely identifies the date of purchase. Yet, the date of purchase is repeated in multiple tuples for the same receipt number. Similarly, the price is uniquely determined by the item purchased. This information also appears in many tuples.

Example 2: Update Anomaly. A database application program can execute an update:

```
UPDATE Purchases SET Price = Price+ .2
WHERE ReceiptNo = 1234 AND Ordinal = 2;
```

This update will make the price of `"Rye Bread"` purchased by `"Fausto Klosner"` on 01/21/2008 to 2.70. However, the price of the same item purchased by `"Susie Gladney"` on 01/23/2008 will remain 2.50.

Example 2: Insertion Anomaly. If Purchases is the only relational table maintained by the BAKERY application, the following anomaly may also occur. Suppose the bakery introduced a new product on the menu: `"Bavarian Beer Bread"`[1] The information of this product *cannot be inserted* in the Purchases table, until some customer actually purchases a loaf. (The reason for this is our inability to insert records whithout primary key attributes. In the Purchases table, as we now know, the (primary) key is ReceiptNo, Ordinal.)

Example 3: Deletion Anomaly. Suppose a database application executes the following statement:

```
DELETE FROM Purchases WHERE ReceiptNo = 3042;
```

The side effect of this deletion is that we no longer have information about the price of `"Opera Cake"` in our database.

Normal Forms

First Normal Form

Example 4: Non-atomic values. Consider a relational table Customer(Name, City, State, Phone). Suppose, some customers may have more than one phone. This can be, generally speaking represented as follows:

Name	City	State	Phone
"Nichole Taves"	"Arroyo Grande"	"CA"	8052342342 8053343344
"Gino Demont"	"Templeton"	"CA"	8054323432
"Dorothy Lovitt"	"Los Osos"	"CA"	8056550055 8054400440 8057897897

This instance of the Customer table, however, is NOT in 1NF, because some of the cells in the Phone column contain multiple (non-atomic) values.

Example 5: Repeated attribute groups . Another approach to dealing with multiple phone numbers is to create as many PhoneN fields as necessary for the current isntance: Customer(Name, City, State, Phone1, Phone2, Phone3) will be the new schema for our instance:

Name	City	State	Phone1	Phone2	Phone3
"Nichole Taves"	"Arroyo Grande"	"CA"	8052342342	8053343344	NULL
"Gino Demont"	"Templeton"	"CA"	8054323432	NULL	NULL
"Dorothy Lovitt"	"Los Osos"	"CA"	8056550055	8054400440	8057897897

This instance of the table is NOT in 1NF either. 1NF does not allow for repeated attrbiutes. (Also, it violates the *"no NULL values"* condition).

Note, however, that if we were to determine that each customer may have up to three different types of phone numbers: office, home, and mobile, then, Customer(Name, City, State, HomePhone, OfficePhone, MobilePhone) does not have repeated attrbiute groups. It still is prone to NULL values, and it still may be better to decompose Customer to ensure 1NF:

[1] Yes, there is such a bread.

Customer(Name, City, State)
CustomerPhone(Name,Type,Phone)

Second Normal Form

Example 6. Dependencies on incomplete keys Let us modify the Customer table: Customer(Name, Zipcode, City, State).

Assume for the moment that in this table, the key is Name, Zipcode. For this table, the following FD is asserted:

Zipcode$\rightarrow$ City,State

The left side of this FD is not a superkey. (in fact, it is not a key at all). Therefore, Customer is NOT in 2NF.

To convert Customer to 2NF, we decompose it into two tables:

Customer(Name, Zipcode)
Locations(Zipcode, City, State)

Example 7. Consider the Purchases(ReceiptNo,Ordinal, PDate, Customer, Item, Price) table from Example 1. Recall that the functional dependencies asserted over this table are:

1. ReceiptNo $\rightarrow$ Customer
2. ReceiptNo $\rightarrow$ PDate
3. Item $\rightarrow$ Price
4. ReceiptNo, Ordinal $\rightarrow$ Item

Is Purchases in **2NF**? If not, how can we modify the database, to restore Second Normal Form?

Third Normal Form

Example 8. Transitive functional dependencies. Consider the following table Books(ISBN, Title, Author, Year, Publisher, PubCity). A table may look as follows:

ISBN	Title	Author	Year	Publisher	PubCity
0-914894-36-6	Database Systems	Ullman	1982	Pittman Publishing	London
0-201-74128-8	Data Mining	Rogier	2003	Addison-Wesley	Boston
0-321-33025-0	Programming Languages	Sebesta	2005	Addison-Wesley	Boston
0-13–17480-7	Spatial Databases	Shekhar	2003	Prentice Hall	Upper Saddle River, NJ
0-13-678012-1	Programming Languages	Pratt	1996	Prentice Hall	Upper Saddle River, NJ

We assert the following FDs on this table:

1. ISBN $\rightarrow$ Title, Author, Year
2. ISBN $\rightarrow$ Publisher
3. Title, Author, Year $\rightarrow$ ISBN

4. Title, Author, Year → Publisher

5. Publisher → PubCity

We can derive from this set of FDs that Books has two keys: ISBN and Title,Author,Year.

However, one attribute PubCity does not directly depend on any of the keys. Instead, it is functionally determined by Publisher, and in order to obtain FDs:

ISBN→ PubCity
Title, Author, Year → PubCity

we must use the transitivity rule. (more importantly, Publisher→ PubCity is an FD which does not contain a key on the left side).

We conclude that Books is NOT in **3NF**. To restore Third Normal Form, we decompose Books into two tables:

- Books(ISBN, Title, Author, Year, Publisher
- Publishers(Publisher, PubCity)

Example 9. Consider the Purchases(ReceiptNo,Ordinal, PDate, Customer, Item, Price) table from Example 1. Recall that the functional dependencies asserted over this table are:

1. ReceiptNo → Customer
2. ReceiptNo → PDate
3. Item → Price
4. ReceiptNo, Ordinal → Item

Is Purchases in **3NF**? If not, how can we modify the database, to restore Third Normal Form?

Boyce-Codd Normal Form

Example 10. 3NF but not BCNF. Consider the table Schedule(Classroom, Day, Course, Time, Instructor) which stores information about the classess offered by some college. A sample table may be:

Classroom	Day	Course	Time	Instructor
14-253	T	CSC 366	12:10	Dekhtyar
14-253	R	CSC 366	12:10	Dekhtyar
14-253	M	CSC 357	1:10	Staley
14-251	M	CSC 309	1:10	Fisher

Suppose the following rules about the course schedule are observed at the college:

- No two classes may meet at the same time and at the same location on the same day of week.
- No professor may teach two different classes on the same day and time.

- No class is taught more than once by the same professor on any given day. (that is, Dekhtyar may teach two sections of CSC 366, but they must meet on different days).

Using this information, we assert the following functional dependencies:

1. Classroom, Day, Time $\rightarrow$ Instructor, Course

2. Instructor, Course $\rightarrow$ Classroom, Time

From this set of FDs, we observe that Schedule has (at least) two keys: Classroom, Day, Time and Instructor, Course, Day.

All attributes of this table are prime, therefore it is in **2NF** and, it is also in **3NF**, as the second FD has prime attributes on the right side.

It is, however, not in **BCNF**. **BCNF** requires **every** rule to have a superkey on the left side, and the second rule does not satisfy this condition.