Database Management Systems

Subject Teacher: Zartasha Baloch

Normalization

Lecture # 9 & 10

Disclaimer: The material used in this presentation to deliver the lecture i.e., definitions/text and pictures/graphs etc. does not solely belong to the author/presenter. The presenter has gathered this lecture material from various sources on web/textbooks. Following sources are especially acknowledged:

1. Connolly, Thomas M., and Carolyn E. Begg. *Database systems: a practical approach to design, implementation, and management*. Pearson Education, 2005.
2. 2. Hoffer, Jeffrey A., Venkataraman Ramesh, and Heikki Topi. *Modern database management*. Upper Saddle River, NJ: Prentice Hall,, 2011.

In this Lecture you will Learn about:

- What is Normalization
- Why to Normalize Table?
- Forms of Normalization

Data Normalization

- A tool to validate and improve a logical design so that it satisfies certain constraints that ***avoid unnecessary duplication of data***
- The process of decomposing relations with anomalies to produce smaller, ***well-structured*** relations

Normalization

- Normalization is the process of organizing data into a set of related tables; it also minimizes redundancy and increases the integrity which improves performance of the query.
- Data normalization is a process in which data attributes within a data model are organized to increase the organization of entity types.
- To normalize a database, we divide the database into tables and establish relationships between the tables.
- The purpose of normalization is to identify a suitable set of relations that support the data requirements of an enterprise.

Normalization (cont'd)

- The characteristics of a suitable set of relations include the following:
  - The minimal number of attributes necessary to support the data requirements of the enterprise;
  - Attributes with a close logical relationship (described as functional dependency) are found in the same relation;
  - Minimal redundancy with each attribute represented only once with the important exception of attributes that form all or part of foreign keys, which are essential for the joining of related relations.
- The benefits of using a database that has a suitable set of relations is that the database will be easier for the user to access and maintain the data, and take up minimal storage space on the storage device.
- When the database is not normalized there are three types of anomalies that occur in a Database.

Well-Structured Relations

- A relation that contains minimal data redundancy and allows users to insert, delete, and update rows without causing data inconsistencies
- Goal is to avoid anomalies
  - **Insertion Anomaly**–adding new rows forces user to create duplicate data
  - **Deletion Anomaly**–deleting rows may cause a loss of data that would be needed for other future rows
  - **Modification Anomaly**–changing data in a row forces changes to other rows because of duplication

General rule of thumb: A table should not pertain to more than one entity type

Why Normalizing a Table?

- These are; ***Insertion***, ***update*** and ***deletion*** anomaly.

1) **Insertion Anomaly,** it occurs when certain attributes cannot be inserted into the database without the presence of other attributes.

   E.g: Let us assume that a new department has been started by the organization but initially there is no employee appointed for that department, then the record for this department cannot be inserted into this table as the ***Employee number*** will have **NULL,** which is not allowed as ***Employee number*** is primary key.

2) **Deletion Anomaly,** it exists when certain attributes are lost because of the deletion of other attributes.

   E.g: Consider there is only one employee in some department and that employee leaves the organization, then the record of that employee has to be deleted from the table, but in addition to that the information about the department also will get deleted.

Why to Normalize a Table?

3) **Update Anomaly,** it exists when one or more instances of duplicated data is updated, but not all.

   E.g: Suppose the manager of a department has changed, this requires that the ***Dept: Manager Number*** in all the records corresponding to that department must be changed to reflect the new status.

   If we fail to update all the records of the given department, then two different records of employee working in the same department might show different ***Dept: Manager Number*** leading to inconsistency in the database.

Forms of Normalization

- There are 05 forms of Normalization, but a table can be normalized up to 3rd form.

1st Normal Form, An entity is in the first normal form if it contains no repeating groups.

In relational terms, a table is in the first normal form if it contains no repeating columns. Repeating columns make your data less flexible, waste disk space, and make it more difficult to search for data.

For 1NF

- The table cells must be of single value.
- Eliminate repeating groups in individual tables.
- Create a separate table for each set of related data.
- Identify each set of related data with a primary key.

Forms of Normalization

2nd Normal Form, A relation is in 2NF if it is in 1NF and every non-key attribute is fully dependent on each candidate key of the relation.

For 2NF

- Remove Partial Dependencies.
- *Functional Dependency*: The value of one attribute in a table is determined entirely by the value of another.
- *Partial Dependency*: A type of functional dependency where an attribute is functionally dependent on only part of the primary key (primary key must be a composite key).
- Create separate table with the functionally dependent data and the part of the key on which it depends. Tables created at this step will usually contain descriptions of resources.

Forms of Normalization

3rd Normal Form, A relation is in third normal form, if it is in 2NF and every non-key attribute of the relation is non-transitively dependent on each candidate key of the relation.

For 3NF

- Remove transitive dependencies.
- Transitive Dependency is a type of functional dependency where an attribute is functionally dependent on an attribute other than the primary key. Thus its value is only indirectly determined by the primary key.
- Create a separate table containing the attribute and the fields that are functionally dependent on it. Tables created at this step will usually contain descriptions of either resources or agents. Keep a copy of the key attribute in the original file.

Example

EMPLOYEE2

Emp_ID	Name	Dept_Name	Salary	Course_Title	Date_Completed
100	Margaret Simpson	Marketing	48,000	SPSS	6/19/200X
100	Margaret Simpson	Marketing	48,000	Surveys	10/7/200X
140	Alan Beeton	Accounting	52,000	Tax Acc	12/8/200X
110	Chris Lucero	Info Systems	43,000	Visual Basic	1/12/200X
110	Chris Lucero	Info Systems	43,000	C++	4/22/200X
190	Lorenzo Davis	Finance	55,000		
150	Susan Martin	Marketing	42,000	SPSS	6/19/200X
150	Susan Martin	Marketing	42,000	Java	8/12/200X

Question–Is this a relation?

Answer–Yes: Unique rows and no multivalued attributes

Question–What's the primary key?

Answer–Composite: Emp_ID, Course_Title

Anomalies in this Table

- **Insertion**–can't enter a new employee without having the employee take a class
- **Deletion**–if we remove employee 140, we lose information about the existence of a Tax Acc class
- **Modification**–giving a salary increase to employee 100 forces us to update multiple records

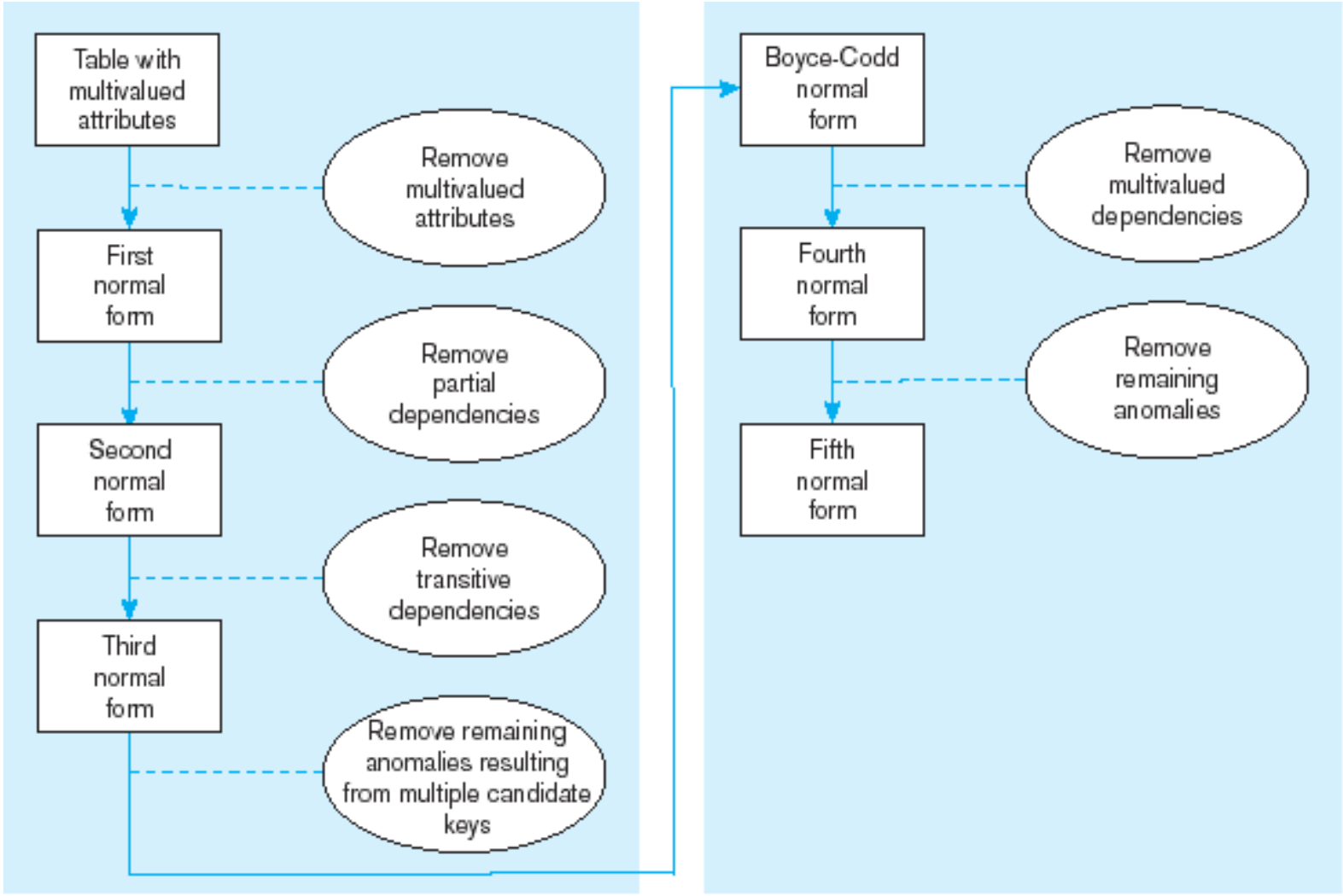
Why do these anomalies exist?

Because there are two themes (entity types) in this one relation. This results in data duplication and an unnecessary dependency between the entities

Functional Dependencies and Keys

- Functional Dependency: The value of one attribute (the ***determinant***) determines the value of another attribute
- Candidate Key:
  - A unique identifier. One of the candidate keys will become the primary key
    - E.g. perhaps there is both credit card number and SS# in a table…in this case both are candidate keys
  - Each non-key field is functionally dependent on every candidate key

Steps in normalization



First Normal Form

- No multivalued attributes
- Every attribute value is atomic
- Fig. *is not* in 1st Normal Form (multivalued attributes) ➔ it is not a relation
- ***All relations* are in 1st Normal Form**

Table with multivalued attributes, not in 1st normal form

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	10/24/2006	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	10/25/2006	6	Furniture Gallery	Boulder, CO	11	4-Dr Dresser	Oak	500.00	4
					4	Entertainment Center	Natural Maple	650.00	3

Note: this is NOT a relation

Table with no multivalued attributes and unique rows, in 1st normal form

Order_ID	Order_ Date	Customer_ ID	Customer_ Name	Customer_ Address	Product_ID	Product_ Description	Product_ Finish	Unit_ Price	Ordered_ Quantity
1006	10/24/2006	2	Value Furniture	Plano, TX	7	Dining Table	Natural Ash	800.00	2
1006	10/24/2006	2	Value Furniture	Plano, TX	5	Writer's Desk	Cherry	325.00	2
1006	10/24/2006	2	Value Furniture	Plano, TX	4	Entertainment Center	Natural Maple	650.00	1
1007	10/25/2006	6	Furniture Gallery	Boulder, CO	11	4-Dr Dresser	Oak	500.00	4
1007	10/25/2006	6	Furniture Gallery	Boulder, CO	4	Entertainment Center	Natural Maple	650.00	3

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

Product_ID → Product_Description, Product_Finish, Unit_Price
Order_ID, Product_ID → Ordered_Quantity

Note: this is relation, but not a well-structured one

Anomalies in this Table

- **Insertion**–if new product is ordered for order 1007 of existing customer, customer data must be re-entered, causing duplication
- **Deletion**–if we delete the Dining Table from Order 1006, we lose information concerning this item's finish and price
- **Update**–changing the price of product ID 4 requires update in several records

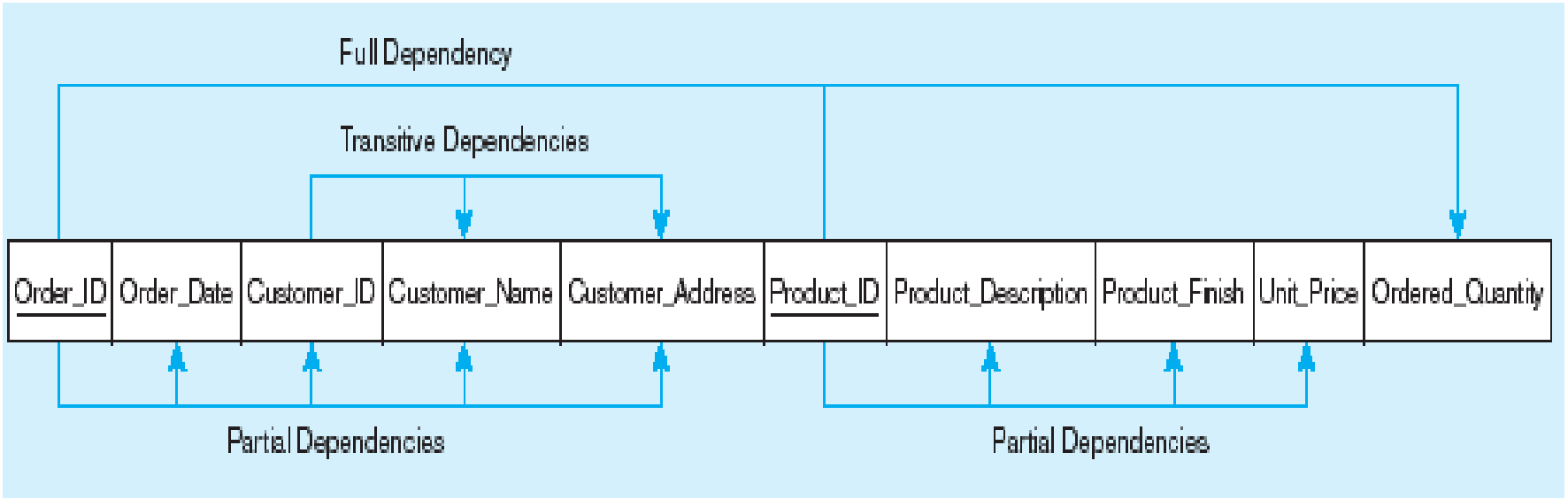
Why do these anomalies exist?

Because there are multiple themes (entity types) in one relation. This results in duplication and an unnecessary dependency between the entities

Second Normal Form

- 1NF PLUS ***every non-key attribute is fully functionally dependent on the ENTIRE primary key***
  - Every non-key attribute must be defined by the entire key, not by only part of the key
  - No partial functional dependencies

Functional dependency diagram for INVOICE



Order_ID ➔ Order_Date, Customer_ID, Customer_Name, Customer_Address

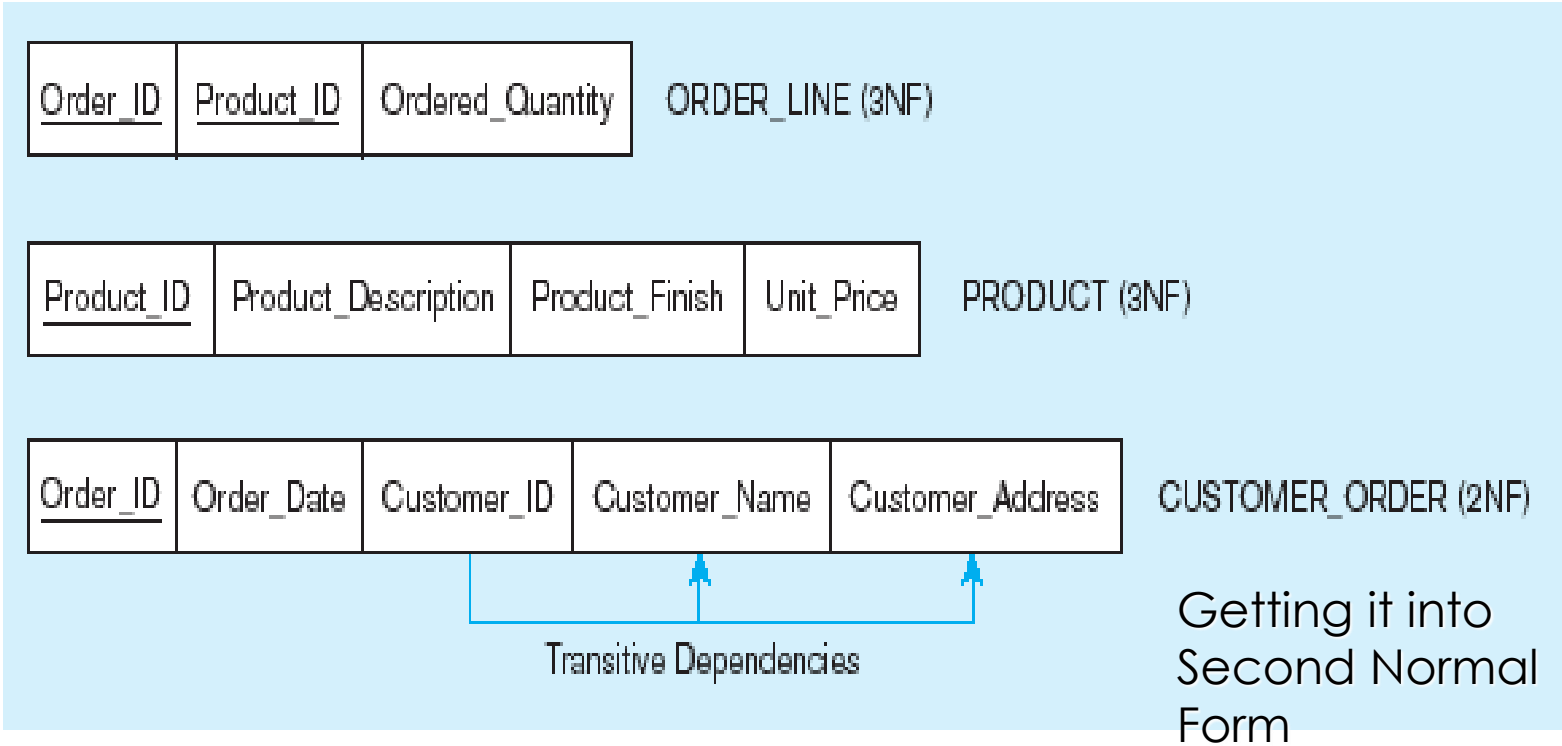
Customer_ID ➔ Customer_Name, Customer_Address

Product_ID ➔ Product_Description, Product_Finish, Unit_Price

Order_ID, Product_ID ➔ Order_Quantity

Therefore, NOT in 2nd Normal Form

Removing partial dependencies

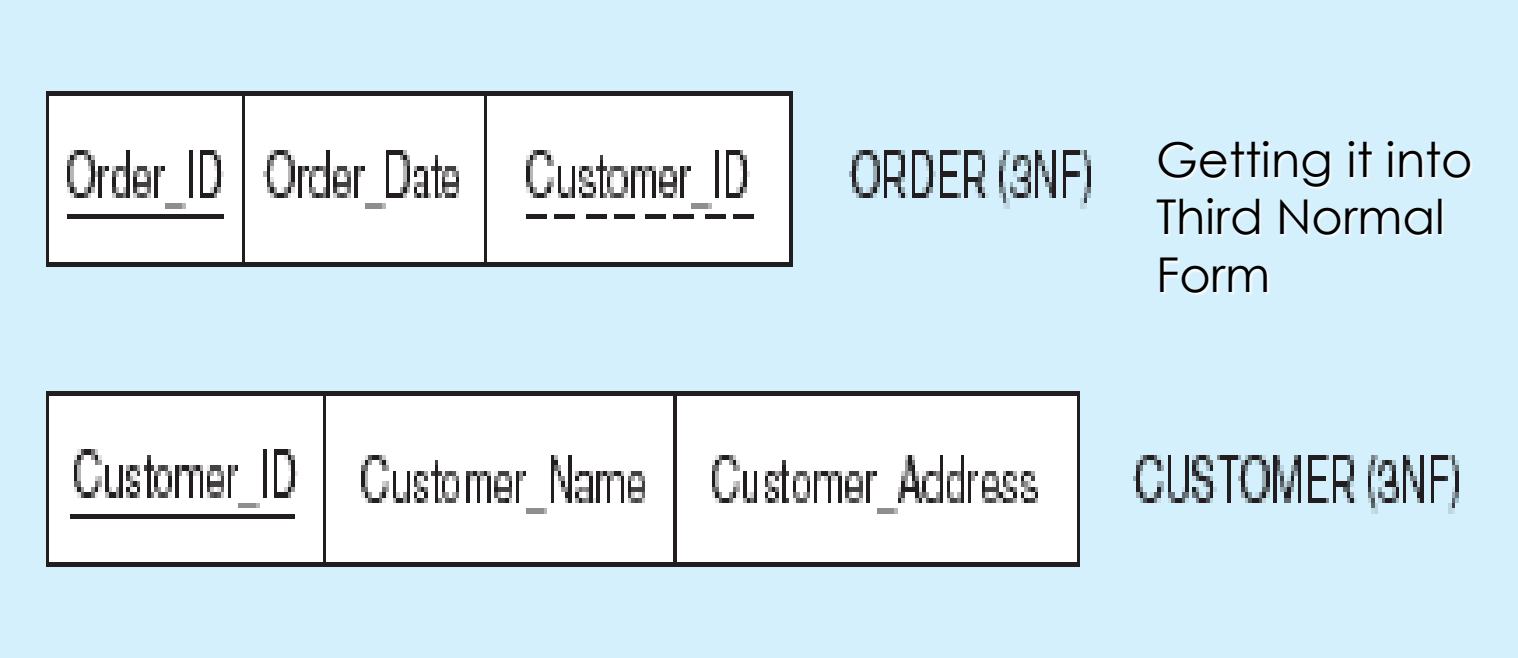


Partial dependencies are removed, but there are still transitive dependencies

Third Normal Form

- 2NF PLUS **no transitive dependencies** (functional dependencies on non-primary-key attributes)
- Note: This is called transitive, because the primary key is a determinant for another attribute, which in turn is a determinant for a third
- Solution: Non-key determinant with transitive dependencies go into a new table; non-key determinant becomes primary key in the new table and stays as foreign key in the old table

Removing partial dependencies



Transitive dependencies are removed

Example of Normalization

TABLE_PRODUCT

Product ID	Color	Price
1	red, green	15.99
2	yellow	23.99
3	green	17.50
4	yellow, blue	9.99
5	red	29.99

Un-Normalized Table

TABLE_PRODUCT_PRICE

Product ID	Price
1	15.99
2	23.99
3	17.50
4	9.99
5	29.99

TABLE_PRODUCT_COLOR

Product ID	Color
1	red
1	green
2	yellow
3	green
4	yellow
4	blue
5	red

Table in 1st Normal Form

2nd Form of Normalization

- This table has a composite primary key [Customer ID, Store ID].
- The non-key attribute is [Purchase Location].
- In this case, [Purchase Location] only depends on [Store ID], which is only part of the primary key. Therefore, this table does not satisfy second normal form.
- To bring this table to second normal form, we break the table into two tables.

TABLE_PURCHASE_DETAIL

Customer ID	Store ID	Purchase Location
1	1	Los Angeles
1	3	San Francisco
2	1	Los Angeles
3	2	New York
4	3	San Francisco

2nd Form of Normalization

TABLE_PURCHASE

Customer ID	Store ID
1	1
1	3
2	1
3	2
4	3

TABLE_STORE

Store ID	Purchase Location
1	Los Angeles
2	New York
3	San Francisco

What we have done is to remove the partial functional dependency that we initially had. Now, in the table [TABLE_STORE], the column [Purchase Location] is fully dependent on the primary key of that table, which is [Store ID].

3rd Form of Normalization

TABLE_BOOK_DETAIL

Book ID	Genre ID	Genre Type	Price
1	1	Gardening	25.99
2	2	Sports	14.99
3	1	Gardening	10.00
4	3	Travel	12.99
5	2	Sports	17.99

In the table [Book ID] determines [Genre ID], and [Genre ID] determines [Genre Type]. Therefore, [Book ID] determines [Genre Type] via [Genre ID] and we have transitive functional dependency, and this structure does not satisfy third normal form.

To bring this table to third normal form, we split the table into two tables.

3rd Form of Normalization

TABLE_BOOK

Book ID	Genre ID	Price
1	1	25.99
2	2	14.99
3	1	10.00
4	3	12.99
5	2	17.99

TABLE_GENRE

Genre ID	Genre Type
1	Gardening
2	Sports
3	Travel

Now all non-key attributes are fully functional dependent only on the primary key. In [TABLE_BOOK], both [Genre ID] and [Price] are only dependent on [Book ID]. In [TABLE_GENRE], [Genre Type] is only dependent on [Genre ID].

Rollno	St_Name	St_Address	Course_Id	Course_Desc	Marks
19BSCS01	ABC	HYD	DBS	dvgd	80
19BSCS01	ABC	HYD	DLD	xzvx	85

RollNo→ St_Name, St_Address
Course_ID → Course_Desc
RollNo, Course_Id → Marks