Data Normalization

Functional Dependency

Functional dependency (FD) is set of constraints between two attributes in a relation. Functional dependency says that if two tuples have same values for attributes A1, A2,..., An then those two tuples must have to have same values for attributes B1, B2, ..., Bn. Functional dependency is represented by arrow sign (→), that is X→Y, where X functionally determines Y. The left hand side attributes determines the values of attributes at right hand side.

Transitivity rule: Same as transitive rule in algebra, if a → b holds and b → c holds then a → c also hold. a → b is called as a functionally determines b.

We might say that Salesperson Number defines Salesperson Name. If I give you a Salesperson Number, you can give me back the one and only name that goes with it. These defining associations are commonly written with a right-pointing arrow like this:
Salesperson Number →Salesperson Name
In the more formal terms of functional dependencies, the attribute on the left side is referred to as the **determinant attribute.** This is because its value determines the value of the attribute on the right side. Conversely, we also say that the attribute on the right is functionally dependent on the attribute on the left.
Figure 4-23 Salesperson entity attributes.

Salesperson Number
Salesperson Name
Commission
Percentage
Year of Hire
Department
Number
Manager Name
Product Number
Product Name
Unit Price
Quantity

Figure 4-24

Salesperson Number → Salesperson Name
Salesperson Number → Commission Percentage
Salesperson Number → Year of Hire
Salesperson Number → Department Number
Salesperson Number → Manager Name
Product Number → Product Name
Product Number → Unit Price
Department Number → Manager Name
Salesperson Number, Product Number → Quantity

Salesperson entity defining associations (functional dependencies).

Normalization

If a database design is not perfect it may contain anomalies, which are like a bad dream for database itself. Managing a database with anomalies is next to impossible. **Data normalization** is a methodology for organizing attributes into tables so that redundancy among the nonkey attributes is eliminated. Each of the resultant tables deals with a single data focus, which is just another way of saying that each resultant table will describe a single entity type or a single many-to-many relationship. Furthermore, foreign keys will appear exactly where they are needed. In other words, the output of the data normalization process is a properly structured relational database

- **Update anomalies:** if data items are scattered and are not linked to each other properly, then there may be instances when we try to update one data item that has copies of it scattered at several places, few instances of it get updated properly while few are left with there old values. This leaves database in an inconsistent state.
- **Deletion anomalies:** we tried to delete a record, but parts of it left undeleted because of unawareness, the data is also saved somewhere else.
- **Insert anomalies:** we tried to insert data in a record that does not exist at all.

Normalization is a method to remove all these anomalies and bring database to consistent state and free from any kinds of anomalies.

* A database anomaly is a fault in a database

Here are three additional points to remember:

1. Once the attributes are arranged in third normal form (and if none of the exception conditions is present), the group of tables that they comprise is, in fact, a well-structured relational database with no data redundancy.

2. A group of tables is said to be in a particular normal form if every table in the group is in that normal form.

3. The data normalization process is progressive. If a group of tables is in second normal form, it is also in first normal form. If the tables are in third normal form, they are also in second normal form.

Understanding Unnormalized Data or zero normal form

The table in Figure 4-25 is unnormalized. The table has four records, one for each salesperson. But since each salesperson has sold several products and there is only one record for each salesperson, several attributes of each record must have multiple values. For example, the record for salesperson 137 has three product numbers, 19440, 24013, and 26722, in its Product Number attribute because salesperson 137 has sold all three of those products. Having such multivalued attributes is not permitted and so this table is unnormalized.

Normalizing to First Normal Form (1NF)

In the **first normal form,** each attribute value is atomic, that is, no attribute is multivalued. The table in Figure 4-26 is the first normal form representation of the data. The attributes under consideration have been listed in one table, and a primary key has been established. In this

"Atomic" just means anything which is not a relation definition of normal forms, the requirement for a primary key is not listed as part of any normal form, but is considered an assumed requirement of the initial E-R diagramming process.

As the sample data in Figure 4-27 shows, the number of records has increased compared to the unnormalized representation. Every attribute of every record has just one value. The multivalued attributes from Figure 4-25 are eliminated.

SALESPERSON/PRODUCT table

Salesperson Number	Product Number	Salesperson Name	Commission Percentage	Year of Hire	Department Number	Manager Name	Product Name	Unit Price	Quantity
137	19440	Baker	10	1995	73	Scott	Hammer	17.50	473
	24013						Saw	26.25	170
	26722						Pliers	11.50	688
186	16386	Adams	15	2001	59	Lopez	Wrench	12.95	1745
	19440						Hammer	17.50	2529
	21765						Drill	32.99	1962
	24013						Saw	26.25	3071
204	21765	Dickens	10	1998	73	Scott	Drill	32.99	809
	26722						Pliers	11.50	734
361	16386	Carlyle	20	2001	73	Scott	Wrench	12.95	3729
	21765						Drill	32.99	3110
	26722						Pliers	11.50	2738

SALESPERSON/PRODUCT table

Salesperson Number	Product Number	Salesperson Name	Commission Percentage	Year of Hire	Department Number	Manager Name	Product Name	Unit Price	Quantity
137	19440	Baker	10	1995	73	Scott	Hammer	17.50	473
137	24013	Baker	10	1995	73	Scott	Saw	26.25	170
137	26722	Baker	10	1995	73	Scott	Pliers	11.50	688
186	16386	Adams	15	2001	59	Lopez	Wrench	12.95	1475
186	19440	Adams	15	2001	59	Lopez	Hammer	17.50	2529
186	21765	Adams	15	2001	59	Lopez	Drill	32.99	1962
186	24013	Adams	15	2001	59	Lopez	Saw	26.25	3071
204	21765	Dickens	10	1998	73	Scott	Drill	32.99	809
204	26722	Dickens	10	1998	73	Scott	Pliers	11.50	734
361	16386	Carlyle	20	2001	73	Scott	Wrench	12.95	3729
361	21765	Carlyle	20	2001	73	Scott	Drill	32.99	3110
361	26722	Carlyle	20	2001	73	Scott	Pliers	11.50	2738

1NF

Data normalized to the first normal form.

Normalizing to Second Normal Form (2NF)

More formally, second normal form does not allow **partial functional dependencies** where data is dependent on part of the primary key. That is, in a table in **second normal form,** every nonkey attribute must be fully functionally dependent on the entire key of that table. In plain language, a nonkey attribute cannot depend on only part of the key, the way that Salesperson Name, Product Name, and most of the other nonkey attributes of Figure 4-26 violate this restriction.

SALESPERSON table

Salesperson Number	Salesperson Name	Commission Percentage	Year of Hire	Department Number	Manager Name

PRODUCT table

Product Number	Product Name	Unit Price

QUANTITY table

Salesperson Number	Product Number	Quantity

Relational tables in the second normal form.

Normalizing to Third Normal Form (3NF)

In **third normal form,** nonkey attributes are not allowed to define other nonkey attributes. Stated more formally, third normal form does not allow **transitive dependencies** in which one nonkey attribute is functionally dependent on another.

SALESPERSON table

Salesperson Number	Salesperson Name	Commission Percentage	Year of Hire	Department Number

DEPARTMENT table

Department Number	Manager Name

PRODUCT table

Product Number	Product Name	Unit Price

QUANTITY table

Salesperson Number	Product Number	Quantity

Relational tables in the third normal form.

Example 1 Student Database
0NF: Un normalized data with multivalued attributes.

1NF:
Remove **multivalued attributes**
Student database (Student_ID, Student_Name, Batch, Advisor, Department_Name, Department_Head, Course_No, Course_Title)

2NF:
Remove **partial functional dependencies.** data is dependent on part of the primary key.
Student (Student _ID, Student_Name, Batch, Advisor, Department_Name, Department_Head)
Student_Course (Student_ID,Course_No, Course_Title)

3NF:
Remove **transitive dependencies**
Student (Student _ID, Student_Name, Batch, Department_Name)
Advisor (Batch, Advisor)
Department (Department_Name, Department_Head)
Student_Course (Student_ID, Course_ID)
Course (Course_ID, Course_Title)

Example 2 Employee Database Primary Key
1NF
Employee Database (Empoyee_ID, Employee_Name, Mobile, Department_Name, Department_Location, Project_ID, Project_Name)

2NF
Employee (Empoyee_ID, Employee_Name, Mobile, Department_Name, Department_Location)
Project (Project_ID, Project_Name, Employee_ID)

3NF
Employee (Empoyee_ID, Employee_Name, Mobile,Department_ID)
Department (Department_ID, Department_Name, Department_Location)
Project (Project_ID, Project_Name, Employee_ID)

Boyce Codd Normal Form (BCNF)
When a relation has more than one candidate key, anomalies may result even though the relation is in 3NF. 3NF does not deal satisfactorily with the case of a relation with overlapping candidate keys. –i.e. composite candidate keys with at least one attribute in common.
•BCNF is based on the concept of a determinant.
–A determinant is any attribute (simple or composite) on which some other attribute is fully functionally dependent.
•A relation is in BCNF is, and only if, every determinant is a candidate key.
The theory
•Consider the following relation and determinants.
R(a,b,c,d)

a,c -> b,d
a,d -> b
•To be in BCNF, all valid determinants must be a candidate key. In the relation R, a,c->b,d is the determinate used, so the first determinate is fine.
•a,d->b suggests that a,d can be the primary key, which would determine b. However this would not determine c. This is not a candidate key, and thus R is not in BCNF.

Example 1

Appointment Table

Patient No	Patient Name	Appointment Id	Time	Doctor
1	Jhon	0	09:00	Zorro
2	Kerr	0	9:00	Killer
3	Adam	1	10:00	Zorro
4	Robert	0	13:00	Killer
5	Zane	1	14:00	Zorro

Two possible keys
•DB(Patno, PatName, appNo, time, doctor)
•Determinants:
–Patno-> PatName
–Patno, appNo-> Time, doctor
–Time -> appNo

•Two options for 1NF primary key selection:
–DB(Patno, PatName, appNo, time, doctor) (example 1a)
–DB(Patno, PatName, appNo, time, doctor) (example 1b)

Example 1a

•DB(Patno, PatName, appNo, time, doctor)
•No repeating groups, so in 1NF
•2NF –eliminate partial key dependencies:
–DB(Patno, appNo, time, doctor)
–R1(Patno, PatName)
•3NF –no transient dependences so in 3NF
•Now try BCNF.

BCNF Every determinant is a candidate key
DB(Patno, appNo, time, doctor)
R1(Patno, PatName)
•Is determinant a candidate key?
–Patno-> PatName
Patno is present in DB, but not PatName, so irrelevant.
-Patno, appNo-> Time, doctor
All LHS and RHS present so relevant. Is this a candidate key? Patno,appNoIS the key, so this is a candidate key.
–Time -> appNo

Time is present, and so is appNo, so relevant. Is this a
candidate key? If it was then we could rewrite DB as:
DB(Patno, appNo, time, doctor)
This will not work, so not BCNF.

Rewrite to BCNF
•DB(Patno, appNo, time, doctor)
R1(Patno, PatName)
•BCNF: rewrite to
DB(Patno, time, doctor)
R1(Patno, PatName)
R2(time, appNo)
•time is enough to work out the appointment number of a patient. Now BCNF is satisfied, and the final relations shown are in BCNF

Example 1b
•DB(Patno, PatName, appNo, time, doctor)
•No repeating groups, so in 1NF
•2NF –eliminate partial key dependencies:
–DB(Patno, time, doctor)
–R1(Patno, PatName)
–R2(time, appNo)
•3NF –no transient dependences so in 3NF
•Now try BCNF.

BCNF Every determinant is a candidate key
DB(Patno, time, doctor)
R1(Patno, PatName)
R2(time, appNo)
•Is determinant a candidate key?
–Patno-> PatName
Patnois present in DB, but not PatName, irrelevant.
–Patno, appNo-> Time, doctor
Not all LHS present so not relevant
–Time -> appNo
Time is present, but not appNo, so not relevant.
–Relations are in BCNF.

Summary -Example 1
This example has demonstrated three things:
•BCNF is stronger than 3NF, relations that are in 3NF are not necessarily inBCNF
•BCNF is needed in certain situations to obtain full understanding of the data model
•there are several routes to take to arrive at the same set of relations in BCNF.
–Unfortunately there are no rules as to which route will be the easiest one to take.

Summary:

1NF: no attribute is multivalued
2NF: no partial functional dependencies
3NF: no transitive dependencies
4NF *(Boyce CoddNormal Form (BCNF)):* no multi-valued dependencies.

What are the Benefits of Database Normalization?

Improved data integrity!
No INSERT or UPDATE anomalies.

Decreased storage requirements!
No redundant data stored.

Faster search performance!
Smaller file for table scans.
More directed searching.