SYLLABUS: Introduction to Normalization Using Functional and Multivalued Dependencies: Informal Design Guidelines for Relation Schema, Functional Dependencies, Normal Forms Based on Primary Keys, General Definitions of Second and Third Normal Forms, Boyce-Codd Normal Form, Multivalued Dependency and Fourth Normal Form, Join Dependencies and Fifth Normal Form.

Introduction to Normalization:

Normalization is a database design technique divides larger tables into smaller tables and links them using relationships. The purpose of Normalization in SQL is to eliminate redundancy and provide security also improve data integrity.

Why do we need Normalization?

The main reason for normalizing the relations is removing these anomalies. Failure to eliminate anomalies leads to data redundancy and can cause data integrity and other problems as the database grows.

Data modification anomalies can be categorized into three types:

1. **Insertion anomaly**: It occurs when we cannot insert data to the table without the presence of another attribute
2. **Update anomaly**: It is a data inconsistency that results from data redundancy and a partial update of data.
3. **Deletion Anomaly**: It occurs when certain attributes are lost because of the deletion of other attributes.

****

Informal Design Guidelines for Relation Schema

Informal guidelines that may be used as measures to determine the quality of relation schema design. There are four methods of informal designs.

1. Making sure that the semantics of the attributes is clear in the schema
2. Reducing the redundant information in tuples
3. Reducing the NULL values in tuples
4. Disallowing the possibility of generating spurious tuples

❖ **Making sure that the semantics of the attributes is clear in the schema:**
Informally, each tuple in a relation should represent one entity or relationship instance. (Applies to individual relations and their attributes).

- Attributes of different entities (EMPLOYEEs, DEPARTMENTs, PROJECTs) should not be mixed in the same relation
- Only foreign keys should be used to refer to other entities
- Entity and relationship attributes should be kept apart as much as possible.

Figure 10.1 A simplified COMPANY relational database schema

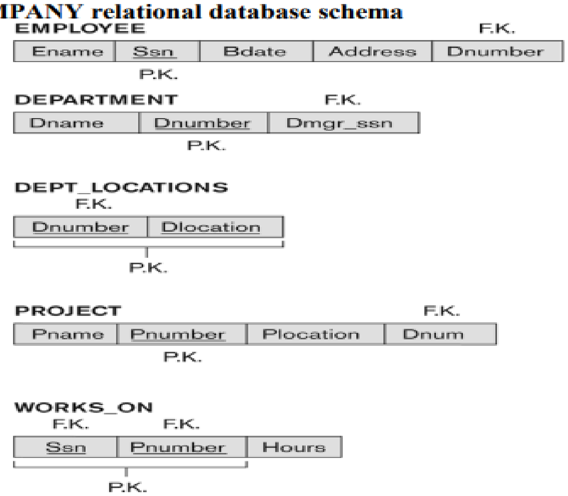


Figure 10.1
A simplified COMPANY relational database schema.

- **Reducing the redundant information in tuples:**
  Information is stored redundantly
  - Wastes storage
  - Causes problems with update anomalies
  - Insertion anomalies
  - Deletion anomalies
  - Modification anomalies

  Design a schema that does not suffer from the insertion, deletion and update anomalies.
- **Reducing the NULL values in tuples:**
  - Relations should be designed such that their tuples will have as few NULL values as possible
  - Attributes that are NULL frequently could be placed in separate relations (with the primary key)
- **Disallowing the possibility of generating fake tuples**
  - The relations should be designed to satisfy the lossless join condition.
  - No spurious tuples should be generated by doing a natural-join of any relations.

****

Functional dependency:

The functional dependency is a relationship that exists between two attributes. It typically exists between the primary key and non-key attribute within a table.

X → Y

The left side of FD is known as a determinant, the right side of the production is known as a dependent.

For example:

Assume we have an employee table with attributes: Emp_Id, Emp_Name, Emp_Address.

Here Emp_Id attribute can uniquely identify the Emp_Name attribute of employee table because if we know the Emp_Id, we can tell that employee name associated with it.

Functional dependency can be written as:

Emp_Id → Emp_Name

Rules of functional dependency:

William Armstrong in *1974* suggested a few rules related to functional dependency. They are called **RAT** rules.

- **Reflexivity**: If **A** is a set of attributes and **B** is a subset of **A**, then the functional dependency **A → B** holds true.
  Example:{ Employee_Id, Name } → Name is valid.
- **Augmentation**: If a functional dependency **A → B** holds true, then appending any number of the attribute to both sides of dependency doesn't affect the dependency. It remains true.
  Example: X → Y holds true then, **ZX → ZY** also holds true.
  Example, if **{ Employee_Id, Name } → { Name }** holds true then, **{ Employee_Id, Name, Age } → { Name, Age }**
- **Transitivity**: If two functional dependencies **X → Y** and **Y → Z** hold true, then **X → Z** also holds true by the rule of Transitivity.
  Example, if **{ Employee_Id } → { Name }** holds true and **{ Name } → { Department }** holds true, then **{ Employee_Id } → { Department }** also holds true.

****

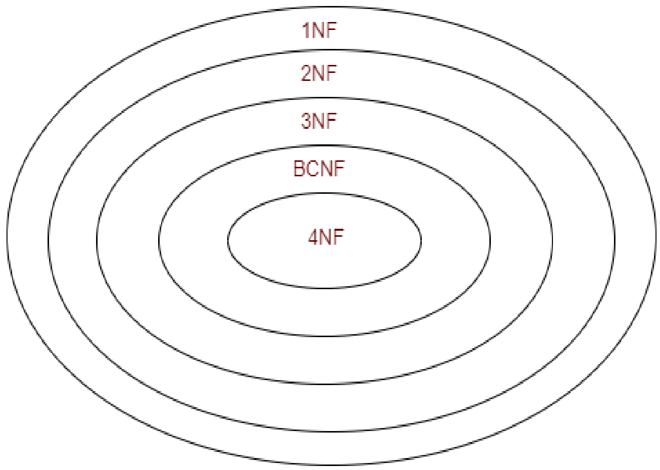
Normal Forms Based on Primary Keys:

Normal forms are used to eliminate or reduce redundancy in database tables. There are several types of keys used in normalization in database management systems (DBMS). These are explained as follows.

- **Super Key:** A super key is a set of one or more attributes that uniquely identifies each record in a table. A super key may contain more attributes than necessary to uniquely identify each record.

- **Candidate Key:** A candidate key is a minimal super key that can uniquely identify each record in a table. In other words, it is a super key that does not contain any unnecessary attributes.
- **Primary Key**: A primary key is a candidate key that has been selected to uniquely identify each record in a table. It is used to enforce entity integrity, and is typically denoted by an underline or a key symbol.
- **Alternate Key**: An alternate key is a candidate key that is not selected to be the primary key. It is used to enforce uniqueness, and may be used as a reference key in another table.
- **Foreign Key**: A foreign key is a key that is used to link two tables together. It is a column (or set of columns) in one table that refers to the primary key of another table.

Normalization is the process of organizing data in a database to minimize redundancy and dependency. In database design, there are different normal forms based on the primary keys of a table. These include



S.No	Normal form	Rule
1	**1NF**	• Contains only atomic values • There are no repeating groups • Every table have primary key.
2	**2NF**	• It is in first normal form • All non-key attributes are fully functional dependent on the primary key
3	**3NF**	• It is in second normal form • There is no transitive functional dependency
4	**4BCNF**	• It should already follow the properties of 3NF • For a functional dependency, A->B, A must be a super key or candidate key.
5	**4NF**	• A relation will be in 4NF if it is in Boyce Codd normal form and has no multi-valued dependency. • For a dependency A → B, if for a single value of A, multiple values of B exists, then the relation will be a multi-valued dependency.
6	**5NF**	• A relation is in 5NF if it is in 4NF and not contains any join dependency and joining

		should be lossless.
		• 5NF is satisfied when all the tables are broken into as many tables as possible in order to avoid redundancy. • 5NF is also known as Project-join normal form (PJ/NF)

****

First Normal Form:

A database is in first normal form if it satisfies the following conditions:

- Contains only atomic values
- There are no repeating groups
- Every table have primary key.

❖ An atomic value is a value that cannot be divided. For example, in the table shown below, the values in the [Color] column in the first row can be divided into "red" and "green", hence [TABLE_PRODUCT] is not in 1NF.

❖ A repeating group means that a table contains two or more columns that are closely related. For example, a table that records data on a book and its author(s) with the following columns: [Book ID], [Author 1], [Author 2], [Author 3] is not in 1NF because [Author 1], [Author 2], and [Author 3] are all repeating the same attribute.

❖ After splitting table every table must be a primary key.

Example

How do we bring an normalized table into first normal form? Consider the following example:

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

This table is not in first normal form because the [Color] column can contain multiple values. For example, the first row includes values "red" and "green."

To bring this table to first normal form, we split the table into two tables and now we have the resulting tables:

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

Now first normal form is satisfied, as the columns on each table all hold just one value.

****

Second Normal Form:

A database is in second normal form if it satisfies the following conditions:

- It is in first normal form
- All non-key attributes are fully functional dependent on the primary key
- ❖ In a table, if attribute B is functionally dependent on A, but is not functionally dependent on a proper subset of A, then B is considered fully functional dependent on A. Hence, in a 2NF table, all non-key attributes cannot be dependent on a subset of the primary key. All non-key attributes are always fully functional dependent on the primary key.
- ❖ A table that is in 1st normal form and contains only a single key as the primary key is automatically in 2nd normal form.

Example:

Let's assume, a school can store the data of teachers and the subjects they teach. In a school, a teacher can teach more than one subject.

TEACHER table

TEACHER_ID	SUBJECT	TEACHER_AGE
25	Chemistry	30
25	Biology	30
47	English	35
83	Math	38
83	Computer	38

In the given table, non-prime attribute TEACHER_AGE is dependent on TEACHER_ID. TEACHER_AGE not dependent to SUBJECT That's why it violates the rule for 2NF. To convert the given table into 2NF, we decompose it into two tables:

TEACHER_DETAIL table:

TEACHER_ID	TEACHER_AGE
25	30
47	35
83	38

TEACHER_SUBJECT table:

TEACHER_ID	SUBJECT
25	Chemistry
25	Biology
47	English
83	Math
83	Computer

Now second normal form is satisfied,

****

Third Normal Form:

A database is in third normal form if it satisfies the following conditions:

- It is in second normal form
- There is no transitive functional dependency

By transitive functional dependency, we mean we have the following relationships in the table: A is functionally dependent on B, and B is functionally dependent on C. In this case, C is transitively dependent on A via B.

Example:

EMPLOYEE_DETAIL table:

EMP_ID	EMP_NAME	EMP_ZIP	EMP_STATE	EMP_CITY
222	Harry	201010	UP	Noida
333	Stephan	02228	US	Boston
444	Lan	60007	US	Chicago
555	Katharine	06389	UK	Norwich
666	John	462007	MP	Bhopal

Here, EMP_STATE & EMP_CITY dependent on EMP_ZIP and EMP_ZIP dependent on EMP_ID. The non-prime attributes (EMP_STATE, EMP_CITY) transitively dependent on super key(EMP_ID). It violates the rule of third normal form.That's why we need to move the EMP_CITY and EMP_STATE to the new <EMPLOYEE_ZIP> table, with EMP_ZIP as a Primary key

EMPLOYEE table:

EMP_ID	EMP_NAME	EMP_ZIP
222	Harry	201010
333	Stephan	02228
444	Lan	60007
555	Katharine	06389
666	John	462007

EMPLOYEE_ZIP table:

EMP_ZIP	EMP_STATE	EMP_CITY
201010	UP	Noida
02228	US	Boston
60007	US	Chicago
06389	UK	Norwich
462007	MP	Bhopal

Now third normal form is satisfied.

****

BCNF Normal Form:

The properties followed by BCNF in DBMS are as-

- It should already follow the properties of 3NF
- For a functional dependency, A->B, A must be a super key or candidate key.

Example of BCNF

Assume there is a hospital where an employee works in more than one department**.**

Employee table

Emp_ID	Nationality	Emp_Dept	Dept_Type	Dept_No
#088	Pakistan	Surgery	X12	301
#088	Pakistan	Dental	X12	482
#112	Canada	General Medicine	X97	212
#112	Canada	Radiology	X97	356

Functional dependencies

- Emp_ID → Nationality
- Emp_Dept → {Dept_Type, Dept_No}

Candidate key

- {Emp_ID, Emp_Dept}

In this example, the table is not in BCNF form as both the Emp_ID and Emp_Dept alone are not keys. To convert the table into BCNF form, decompose the table into three tables based on the functional dependency.

Nationality table

Emp_ID	Nationality
#088	Pakistan
#112	Canada

Dept table

Emp_Dept	Dept_Type	Dept_No
Surgery	X12	301
Dental	X12	482
General Medicine	X97	212
Radiology	X97	356

Dept Mapping table

Emp_ID	Emp_Dept
#088	Surgery
#088	Dental
#112	General Medicine
#112	Radiology

Functional dependencies

- Emp_ID → Nationality
- Emp_Dept → {Dept_Type, Dept_No}

Candidate key

- Nationality Table: Emp_ID
- Dept Table: Emp_Dept
- Dept Mapping Table: {Emp_ID, Emp_Dept}

The relation is now in BCNF form because it satisfies both conditions which are that the table is already in 3NF form and on the LHS of the functional dependency there is a candidate key.

****

<u>Multi valued dependency:</u>

- ❖ Multivalued dependency occurs when two attributes in a table are independent of each other but, both depend on a third attribute.
- ❖ A multivalued dependency consists of at least two attributes that are dependent on a third attribute that's why it always requires at least three attributes.

Example: Suppose there is a bike manufacturer company which produces two colors(white and black) of each model every year.

BIKE_MODEL	MANUF_YEAR	COLOR
M2011	2008	White
M2001	2008	Black
M3001	2013	White
M3001	2013	Black
M4006	2017	White
M4006	2017	Black

Here columns COLOR and MANUF_YEAR are dependent on BIKE_MODEL and independent of each other.

In this case, these two columns can be called as multivalued dependent on BIKE_MODEL. The representation of these dependencies is shown below:

BIKE_MODEL → → MANUF_YEAR
BIKE_MODEL → → COLOR

This can be read as "BIKE_MODEL multidetermined MANUF_YEAR" and "BIKE_MODEL multidetermined COLOR".

****

Fourth Normal Form:

- A relation will be in 4NF if it is in Boyce Codd normal form and has no multi-valued dependency.
- For a dependency A → B, if for a single value of A, multiple values of B exists, then the relation will be a multi-valued dependency.

Example

STUDENT

STU_ID	**COURSE**	**HOBBY**
21	Computer	Dancing
21	Math	Singing
34	Chemistry	Dancing
74	Biology	Cricket
59	Physics	Hockey

The given STUDENT table is in 3NF, but the COURSE and HOBBY are two independent entity. Hence, there is no relationship between COURSE and HOBBY.

In the STUDENT relation, a student with STU_ID, **21** contains two courses, **Computer** and **Math** and two hobbies, **Dancing** and **Singing**. So there is a Multi-valued dependency on STU_ID, which leads to unnecessary repetition of data. So to make the above table into 4NF, we can decompose it into two tables:

STUDENT_COURSE

STU_ID	**COURSE**
21	Computer
21	Math
34	Chemistry
74	Biology
59	Physics

STUDENT_HOBBY

STU_ID	**HOBBY**
21	Dancing
21	Singing
34	Dancing
74	Cricket
59	Hockey

****

Join Dependencies:

A relation is said to have join dependency if it can be recreated by joining multiple sub relations and each of these sub relations has a subset of the attributes of the original relation.

A relation R satisfies join dependency if R is equal to the join of R1,R2,.....Rn where Ri is a subset of the set of attributes of R.

Relation R

Dept	Subject	Name
CSE	C	Ammu
CSE	C	Amar
CSE	Java	Amar
IT	C	bhanu

Here,

dept ->-> subject

dept->-> name

The above relation is in 4NF. Anomalies can occur in relation in 4NF if the primary key has three or more fields. The primary key is (dept, subject, name). Sometimes decomposition of a relation into two smaller relations does not remove redundancy.The above relation says that dept offers many elective subjects which are taken by a variety of students. Students have the opinion to choose subjects. Therefore all three fields are needed to represent the information.

The above relation does not show non-trivial MVDs since the attributes subject and name are dependent; they are related to each other (A FD subject->name exists). The relation cannot be decomposed in two relations (dept, subject) and (dept,sname).

Therefore the relation can be decomposed into following three relations −

R1(dept, subject)

R2(dept, name) and

R3(subject, name) and it can be shown that decomposition is lossless.

R1

Dept	Subject
CSE	C
CSE	Java
IT	C

R2

Dept	Name
CSE	Ammu
CSE	Amar
IT	bhanu

R3

Subject	Name
C	Ammu
C	Amar
Java	Amar
C	bhanu

****

Fifth Normal Form:

- A relation is in 5NF if it is in 4NF and not contains any join dependency and joining should be lossless.
- 5NF is satisfied when all the tables are broken into as many tables as possible in order to avoid redundancy.
- 5NF is also known as Project-join normal form (PJ/NF)

Relation R

Dept	Subject	Name
CSE	C	Ammu
CSE	C	Amar
CSE	Java	Amar
IT	C	bhanu

Here,

dept ->-> subject

dept->-> name

The above relation is in 4NF. Anomalies can occur in relation in 4NF if the primary key has three or more fields. The primary key is (dept, subject, name). Sometimes decomposition of a relation into two smaller relations does not remove redundancy.The above relation says that dept offers many elective subjects which are taken by a variety of students. Students have the opinion to choose subjects. Therefore all three fields are needed to represent the information.

The above relation does not show non-trivial MVDs since the attributes subject and name are dependent; they are related to each other (A FD subject->name exists). The relation cannot be decomposed in two relations (dept, subject) and (dept,sname).

Therefore the relation can be decomposed into following three relations −

R1(dept, subject)

R2(dept, name) and R3(subject, name) and it can be shown that decomposition is lossless.

R1

Dept	Subject
CSE	C
CSE	Java
IT	C

R2

Dept	Name
CSE	Ammu
CSE	Amar
IT	bhanu

R3

Subject	Name
C	Ammu
C	Amar
Java	Amar
C	bhanu

****