

Chapter 04

Logical Database Design



5th Edition

Elmasri / Navathe

Database Normalization

- **Database normalization** is the process of removing redundant data from your tables to **improve storage efficiency, data integrity, and scalability**.
- In the relational model, methods exist for quantifying how efficient a database is. These classifications are called **normal forms** (or **NF**), and there are algorithms for converting a given database between them.
- Normalization generally involves splitting existing tables into multiple ones, which must be re-joined or linked each time a query is issued.

Data redundancy

- Data redundancy in database means that some data fields are repeated in the database.
- This data repetition may occur either if a field is repeated in two or more tables or if the field is repeated within the table.
- Data can appear multiple times in a database for a variety of reasons. For example, a shop may have the same customer's name appearing several times if that customer has bought several different products at different dates.

Disadvantages of data redundancy

- 1.Increases the size of the database unnecessarily.
- 2.Causes data inconsistency.
- 3.Decreases efficiency of database.
- 4.May cause data corruption.

What is an Anomaly?

- Problems that can occur in poorly planned, un-normalized databases where all the data is stored in one table (a flat-file database).
- Without Normalization, it becomes difficult to handle and update the database, without facing data loss. Insertion, Updating and Deletion Anomalies are very frequent if Database is not Normalized.
- There are 3 types of anomalies in database because of redundancy.
 1. Insertion
 2. Deletion
 3. modification/updating

Insert Anomaly

- An **Insert Anomaly** occurs when certain attributes cannot be inserted into the database without the presence of other attributes. For example this is the converse of delete anomaly - we can't add a new course unless we have at least one student enrolled on the course.

StudentNum	CourseNum	Student Name	Address	Course
S21	9201	Jones	Edinburgh	Accounts
S21	9267	Jones	Edinburgh	Accounts
S24	9267	Smith	Glasgow	physics
S30	9201	Richards	Manchester	Computin g
S30	9322	Richards	Manchester	Maths

Delete Anomaly

- A **Delete Anomaly** exists when certain attributes are lost because of the deletion of other attributes. For example, consider what happens if Student S30 is the last student to leave the course - All information about the course is lost.

StudentNum	CourseNum	Student Name	Address	Course
S21	9201	Jones	Edinburgh	Accounts
S21	9267	Jones	Edinburgh	Accounts
S24	9267	Smith	Glasgow	physics
S30	9201	Richards	Manchester	Computin g
S30	9322	Richards	Manchester	Maths

Update Anomaly

- An **Update Anomaly** exists when one or more instances of duplicated data is updated, but not all. For example, consider Jones moving address - you need to update all instances of Jones's address.

StudentNum	CourseNum	Student Name	Address	Course
S21	9201	Jones	Edinburgh	Accounts
S21	9267	Jones	Edinburgh	Accounts
S24	9267	Smith	Glasgow	physics
S30	9201	Richards	Manchester	Computing
S30	9322	Richards	Manchester	Maths

2. Functional Dependencies

- Functional dependencies (FDs)
 - A functional dependency in a database enforces a set of constraints between attributes. **This occurs when one attribute in a relation uniquely determines another attribute.** This can be written $A \rightarrow B$ which means "B is functionally dependent upon A." This is also called a database dependency.

In this relationship, A determines the value of B, while B depends on A.

- A set of attributes X *functionally determines* a set of attributes Y if the value of X determines a unique value for Y

Why Functional Dependency is Important in Database Design

- **Functional dependency helps ensure the validity of data.** Consider a table Employees that lists characteristics including Social Security Number (SSN), name, date of birth, address and so on. The attribute SSN will determine the value of name, date of birth, address and perhaps other values, because a social security number is unique, while a name, date of birth or address may not be.
 - We can write it like this: $SSN \rightarrow \text{name, date of birth, address}$ Therefore, name, date of birth and address are functionally dependent on SSN.
 - However, the reverse statement ($\text{name} \rightarrow SSN$) is not true because more than one employee can have the same name but will never have the same SSN.

Functional Dependencies

- Consider a relation with attributes A and B, where attribute B is functionally depends on attribute A. Let say an A is a PK of R.

$R(A,B)$
 $A \rightarrow B$



- To describe the relationship between attributes A and B is to say that “**A functionally determines B**”.

Functional Dependencies

- When a functional dependency exist, the attribute or group of attributes on the left-handed side of the arrow is called **determinant**.

Determinant:

Refers to the attributes, or a group of attributes, on the left handed side of the arrow of a functional dependency.



Functional Dependencies

staff

staffNO	sName	position	salary	branchNo
S21	Johan	Manager	3000	B005
S37	Ana	Assistant	1200	B003
S14	Daud	Supervisor	1800	B003
S9	Mary	Assistant	900	B007
S5	Siti	Manager	2400	B003
S41	Jani	Assistant	900	B005

branch

branchNO	bAddress
B005	123, Kepong
B007	456, Nilai
B003	789, PTP

Determinant



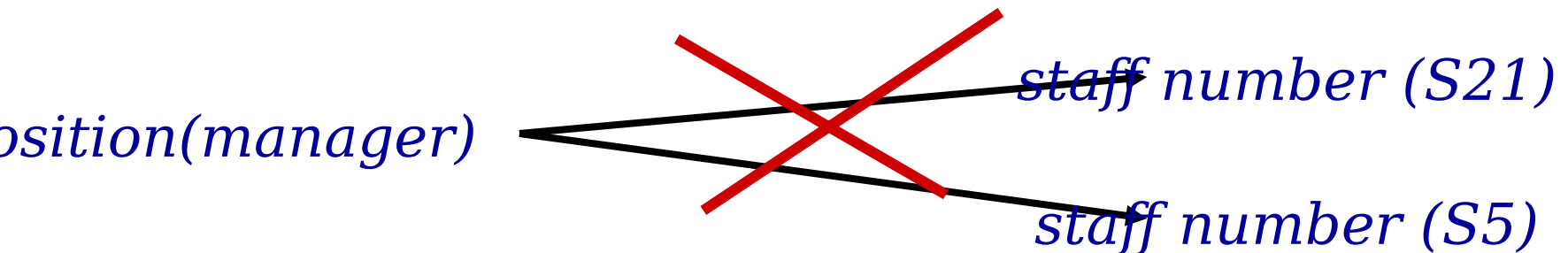
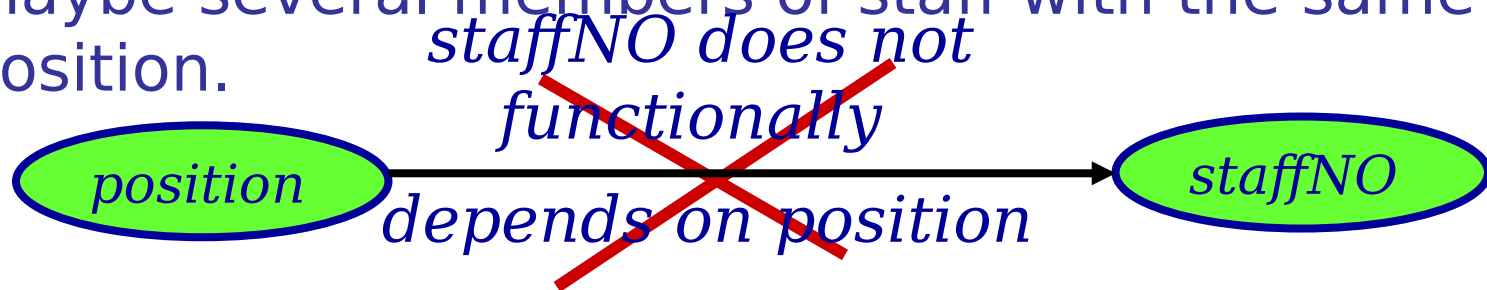
Functional Dependencies

- Consider the attributes **staffNO** and **position** of the staff relation.
- For a specific **staffNO** (S21), we can determine the **position** of that member of staff as Manager.
- **staffNO functionally determines position.**



Functional Dependencies

- However the next figure illustrate that the opposite is not true, as **position does not functionally determines staffNO.**
- A member of staff holds one position; however, they maybe several members of staff with the same position.



Examples of FD constraints (1)

- Social security number determines employee name
 - $SSN \rightarrow ENAME$
- Project number determines project name and location
 - $PNUMBER \rightarrow \{PNAME, PLOCATION\}$
- Employee ssn and project number determines the hours per week that the employee works on the project
 - $\{SSN, PNUMBER\} \rightarrow HOURS$

Fully Functional Dependency

- A functional dependency $P \rightarrow Q$ is **full functional dependency** if removal of any attribute A from P means that the dependency does not hold any more. or
- In a relation R , an attribute Q is said to be fully functional dependent on attribute P , if it is functionally dependent on P and not functionally dependent on any proper subset of P . The dependency $P \rightarrow Q$ is left reduced, there being no extraneous attributes in the left hand side of the dependency.

If $AD \rightarrow C$, is fully functional dependency, then we cannot remove A or D . i.e. C is fully functional dependent on AD . If we are able to remove A or D , then it is not full functional dependency.

$\{SSN, PNUMBER\} \rightarrow HOURS$ is a full FD
since neither $SSN \rightarrow HOURS$ nor $PNUMBER \rightarrow HOURS$ hold

Partial Functional Dependency

- A Functional Dependency in which one or more non key attributes are functionally depending on a part of the primary key is called partial functional dependency.

Or

- where the determinant consists of key attributes, but not the entire primary key, and the determined consist of non-key attributes.

For Example, Consider a Relation $R(A,B,C,D,E)$ having

$FD : AB \rightarrow CDE$ where PK is AB.

Then, $\{ A \rightarrow C; A \rightarrow D; A \rightarrow E; B \rightarrow C; B \rightarrow D; B \rightarrow E \}$ all are Partial Dependencies.

$\{SSN, PNUMBER\} \rightarrow ENAME$ is not a full FD (it is called a partial dependency) since $SSN \rightarrow ENAME$ also holds

3 Normal Forms Based on Primary Keys

- 3.1 Normalization of Relations
- 3.2 Practical Use of Normal Forms
- 3.3 Definitions of Keys and Attributes Participating in Keys
- 3.4 First Normal Form
- 3.5 Second Normal Form
- 3.6 Third Normal Form

3.1 Normalization of Relations (1)

- **Normalization:**

- The process of decomposing unsatisfactory "bad" relations by breaking up their attributes into smaller relations

- **Normal form:**

- Condition using keys and FDs of a relation to certify whether a relation schema is in a particular normal form

The Process of Normalization

- Formal technique for analysing a relation based on its primary key and functional dependencies between its attributes.
- Often executed as a series of steps. Each step corresponds to a specific normal form, which has known properties.
- As normalization proceeds, relations become progressively more restricted (stronger) in format and also less vulnerable to update anomalies.

3.2 Practical Use of Normal Forms

- **Normalization** is carried out in practice so that the resulting designs are of high quality and meet the desirable properties
- The practical utility of these normal forms becomes questionable when the constraints on which they are based are *hard to understand* or to *detect*
- The database designers *need not* normalize to the highest possible normal form
- (usually up to 1st NF, 2nd NF, 3rd NF)

I. First Normal Form (1NF)

■ A relation in which intersection of each row and column contains one and only one value. The official qualifications for 1NF are:

1. Each attribute must have a unique name.
2. Each attribute must have a single value.
3. Row cannot be duplicated.
4. There is no repeating groups.

Doesn't allow

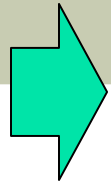
- composite attributes
- multivalued attributes

Additional:

1. Choose a primary key. The primary key can be an attribute or combined attributes.

Name	DOB	Course		Payment
Sok	11/5/1990	IT		450 Dollars
Sao	4/4/1989	Mgt		400 Dollars
Chan	7/7/1991	IT	Mgt	IT: 450 Dollars Mgt: 400 Dollars
Sok	11/5/1990	Mgt		400 Dollars
Sao	4/4/1989	Tour		1) 200 Dollars 2) 200 Dollars

1. Each attribute has unique name -> Good
2. All rows are not duplicated -> Good
3. *The Course and Payment have repeating groups -> Bad*



Name	DOB	Course	Payment (\$)
Sok	11/5/1990	IT	450
Sao	4/4/1989	Mgt	400
Chan	7/7/1991	IT	450
Chan	7/7/1991	Mgt	400
Sok	11/5/1990	Mgt	400
Sao	4/4/1989	Tour	200
Sao	4/4/1989	Tour	200

All correct? Not yet. Choose a

Name? No. Name has duplicated values.

Or DOB, or Course or Payment? No. Each one has duplicated values.

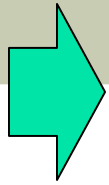
Name and DOB? No. They still have duplicated values.

Name and DOB and Course? No. Still duplicated.

Combine all attribute? Still no. The last two rows are duplicated.

So what else we can do?

Of course, there is a way. Add a new attribute to be a primary key. So let's call it **ID**.



<u>ID</u>	Name	DOB	Course	Payment
1	Sok	11/5/1990	IT	450
2	Sao	4/4/1989	Mgt	400
3	Chan	7/7/1991	IT	450
4	Chan	7/7/1991	Mgt	400
5	Sok	11/5/1990	Mgt	300
6	Sao	4/4/1989	Tour	200

Now it is completely in 1NF.

Next, check it if it is not in 2NF.

II Second Normal Form (1)

- Uses the concepts of **FDs, primary key**
- Definitions
 - **Prime attribute:** An attribute that is member of the primary key K
 - **Full functional dependency:** a FD $Y \rightarrow Z$ where removal of any attribute from Y means the FD does not hold any more

The official qualifications for 2NF are:

1. A table is already in 1NF.
2. All non key attributes are fully dependent on the primary key.

All partial dependencies are removed and placed in another table.

- Examples:
 - $\{SSN, PNUMBER\} \rightarrow HOURS$ is a full FD since neither $SSN \rightarrow HOURS$ nor $PNUMBER \rightarrow HOURS$ hold
 - $\{SSN, PNUMBER\} \rightarrow ENAME$ is not a full FD (it is called a partial dependency) since $SSN \rightarrow ENAME$ also holds

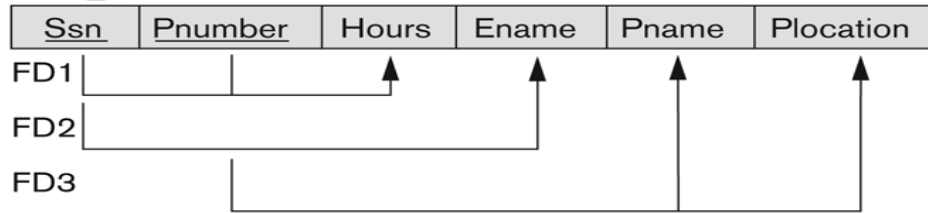
Normalizing into 2NF and 3NF

Figure 10.10

Normalizing into 2NF and 3NF.
(a) Normalizing EMP_PROJ into 2NF relations.
(b) Normalizing EMP_DEPT into 3NF relations.

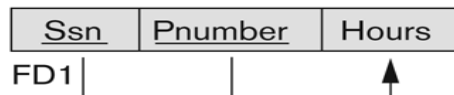
(a)

EMP_PROJ

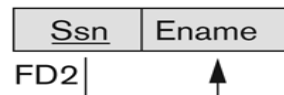


2NF Normalization

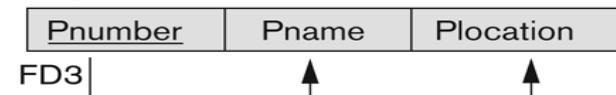
EP1



EP2

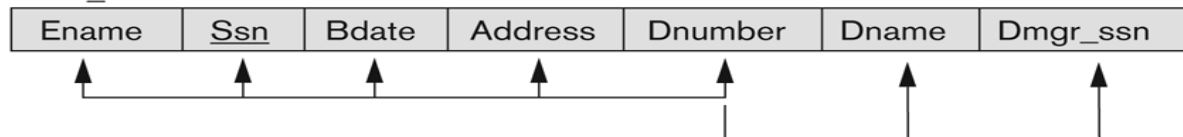


EP3



(b)

EMP_DEPT

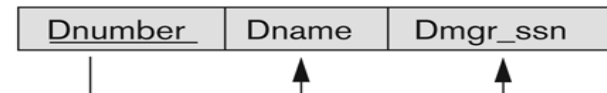


3NF Normalization

ED1



ED2



III. Third Normal Form (3NF)

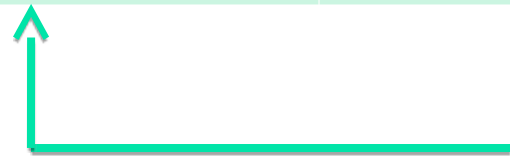
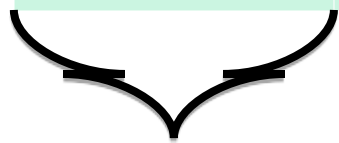
The official qualifications for 3NF are:

1. A table is already in 2NF.
2. Non primary key attributes **do not depend** on other non primary key attributes (i.e. no transitive dependencies)

All transitive dependencies are removed and placed in another table.

Assume you have a table below contain a primary (StudyID):

<u>StudyID</u>	Course Name	Teacher Name	Teacher Tel
1	Database	Sok Piseth	012 123 456
2	Database	Sao Kanha	0977 322 111
3	Web Prog	Chan Veasna	012 412 333
4	Web Prog	Chan Veasna	012 412 333
5	Networking	Pou Sambath	077 545 221



Primary Key

*The Teacher Tel is a non key attribute, and the Teacher Name is also a non key attribute. But Teacher Tel depends on Teacher Name. It is called **transitive dependency**.*

Conclusion:

*Remove **Teacher Name** and **Teacher Tel** together and create a new table.*

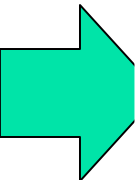
Teacher Name	Teacher Tel
Sok Piseth	012 123 456
Sao Kanha	0977 322 111
Chan Veasna	012 412 333
Chan Veasna	012 412 333
Pou Sambath	077 545 221

Done?

Oh no, it is still not in 1NF yet.

So you have to remove the repeating groups, and ...

<u>Stud yID</u>	Course Name	T.I D
1	Database	T1
2	Database	T2
3	Web Prog	T3
4	Web Prog	T3
5	Networking	T4



<u>Teacher Name</u>	Teacher Tel
Sok Piseth	012 123 456
Sao Kanha	0977 322 111
Chan Veasna	012 412 333
Pou Sambath	077 545 221

Note about primary key:

- In theory, you can choose Teacher Name to be a primary key
- But in practice, you should add Teacher ID as the primary key.



<u>T.I D</u>	Teacher Name	Teacher Tel
T1	Sok Piseth	012 123 456
T2	Sao Kanha	0977 322 111
T3	Chan Veasna	012 412 333

Boyce Codd normal form (BCNF)

- BCNF is the advance version of 3NF. It is stricter than 3NF.
- A table is in BCNF if every functional dependency $X \rightarrow Y$, X is the super key of the table.
- For BCNF, the table should be in 3NF, and for every FD, LHS is super key.
- Functional dependencies are

$EMP_ID \rightarrow EMP_COUNTRY$, $EMP_DEPT \rightarrow \{DEPT_TYPE, EMP_DEPT_NO\}$

EMP_ID	EMP_COUNTRY	EMP_DEPT	DEPT_TYPE	EMP_DEPT_NO
264	India	Designing	D394	283
264	India	Testing	D394	300
364	UK	Stores	D283	232
364	UK	Developing	D283	549

CONT...

EMP_DEPT table:

EMP_DEPT	DEPT_TYPE	EMP_DEPT_NO
Designing	D394	283
Testing	D394	300
Stores	D283	232
Developing	D283	549

EMP_DEPT_MAPPING table:

EMP_ID	EMP_DEPT
D394	283
D394	300
D283	232
D283	549

EMP_COUNTRY table:

EMP_ID	EMP_COUNTRY
264	India
264	India

Candidate keys:

For the first table: EMP_ID

For the second table: EMP_DEPT

For the third table: {EMP_ID,
EMP_DEPT}

Now, this is in BCNF because left side part of both the functional dependencies is a key.

Fourth normal form (4NF)

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

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

CONT...

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

Fifth normal form (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).

SUBJECT	LECTURER	SEMESTER
Computer	Anshika	Semester 1
Computer	John	Semester 1
Math	John	Semester 1
Math	Akash	Semester 2
Chemistry	Praveen	Semester 1

CONT...

P1

SEMESTER	SUBJECT
Semester 1	Computer
Semester 1	Math
Semester 1	Chemistry
Semester 2	Math

P2

SUBJECT	LECTURER
Computer	Anshika
Computer	John
Math	John
Math	Akash

P3

SEMSTER	LECTURER
Semester 1	Anshika
Semester 1	John
Semester 1	John
Semester 2	Akash
Semester 1	Praveen

Advantage of Normalization

- The database does not have redundant data, it is smaller in size so less money needs to be spent on storage
- Because there is less data to search through, it is much faster to run a query on the data
- Because there is no data duplication there is better data integrity and less risk of mistakes.
- Because there is no data duplication there is less chance of storing two or more different copies of the data
- Once change can be made which can instantly be cascaded across any related records.
- Data integrity (because there is no redundant, neglected data).
- Optimized queries (because normalized tables

unNormalized

- 1) Repeat Group
- 2) PK is not defined

FirstNormalForm

repeating group
- PK defined

partial dependency

SecondNormalForm

ThirdNormalForm

No repeating group
-PK defined
No partial dependency
-No transitive dependency

- No

-Test
-No repeating group
-PK defined
*-No partial
dependency*
*-Test transitive
dependency*



amentals of
base
ems

dition

Navathe