DATABASE NORMALIZATION

Partially Edited and Presented by

Dr. Md. Abir Hossain

What is Normalization ?

- **NORMALIZATION** is a database design technique that organizes tables in a manner that reduces redundancy and dependency of data.
- Normalization divides larger tables into smaller tables and links them using relationships.
- The purpose of Normalization is to eliminate redundant (useless) data and ensure data is stored logically.
- The inventor of the relational model E.F.Codd proposed the theory of normalization.

What is an Anomaly?

- Problems that can occur in poorly planned, unnormalized databases where all the data is stored in one table (a flat-file database).
- Types of Anomalies:
  - Insert
  - Delete
  - Update

Anomalies in DBMS

- **Insert Anomaly :** An Insert Anomaly occurs when certain attributes cannot be inserted into the database without the presence of other attributes.
- **Delete Anomaly:** A Delete Anomaly exists when certain attributes are lost because of the deletion of other attributes.
- **Update Anomaly:** An Update Anomaly exists when one or more instances of duplicated data is updated, but not all.

Anomaly Example

- Below table University consists of seven attributes: **Sid, Sname, Cid, Cname, Fid, Fname,** and **Salary.** And the Sid acts as a key attribute or a primary key in the relation.

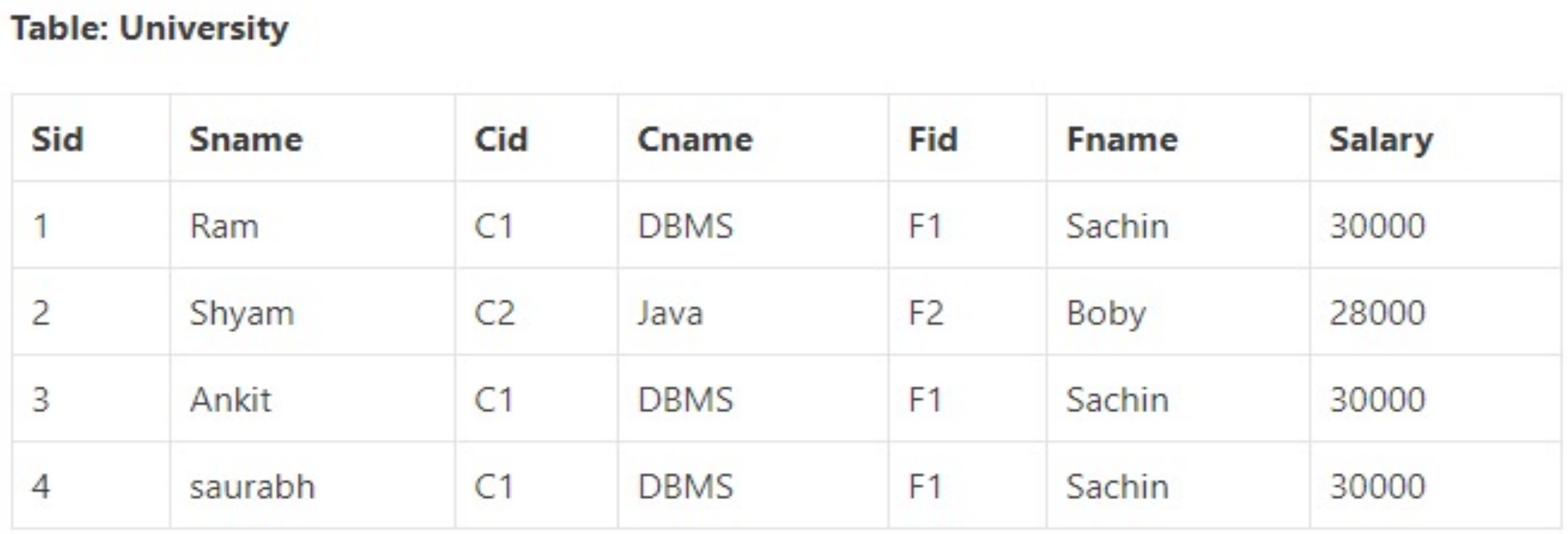


Table: University

Sid	Sname	Cid	Cname	Fid	Fname	Salary
1	Ram	C1	DBMS	F1	Sachin	30000
2	Shyam	C2	Java	F2	Boby	28000
3	Ankit	C1	DBMS	F1	Sachin	30000
4	saurabh	C1	DBMS	F1	Sachin	30000

Insertion Anomaly

- Suppose a new faculty joins the University, and the Database Administrator inserts the faculty data into the above table. But he is not able to insert because Sid is a primary key, and can’t be NULL. So this type of anomaly is known as an insertion anomaly.

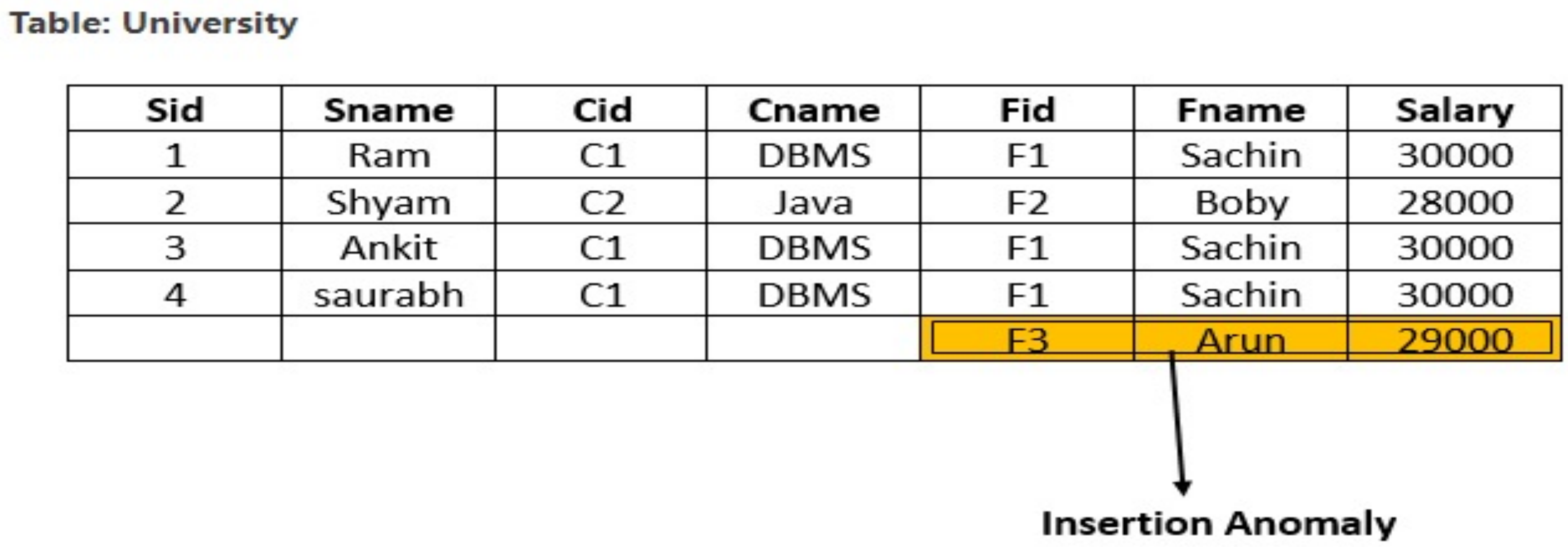


Table: University

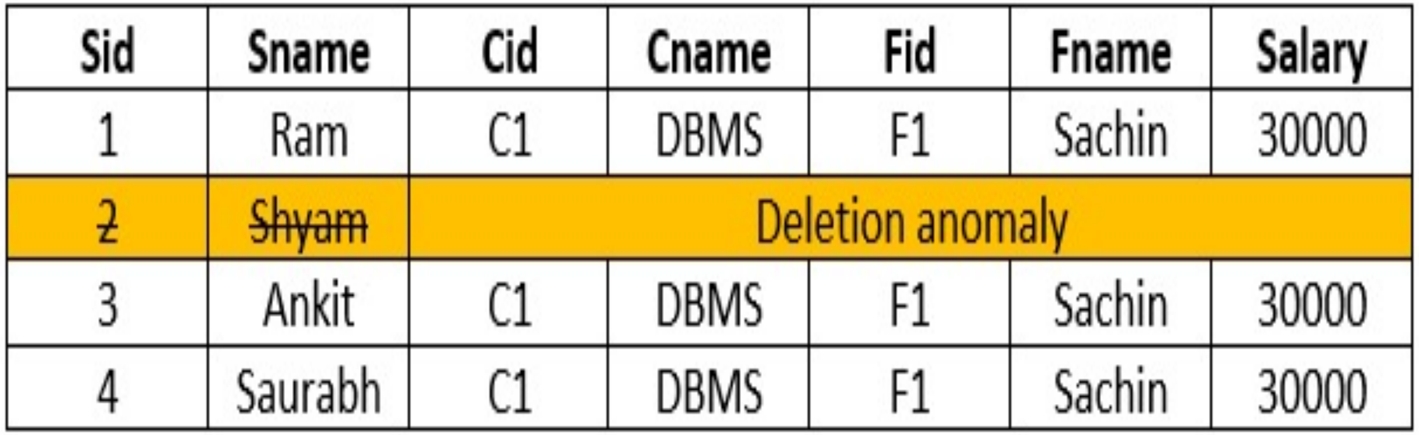
Sid	Sname	Cid	Cname	Fid	Fname	Salary
1	Ram	C1	DBMS	F1	Sachin	30000
2	Shyam	C2	Java	F2	Boby	28000
3	Ankit	C1	DBMS	F1	Sachin	30000
4	saurabh	C1	DBMS	F1	Sachin	30000
				F3	Arun	29000

Insertion Anomaly

Delete Anomaly

- When the Database Administrator wants to delete the student details of Sid=2 from the above table, then it will delete the faculty and course information too which cannot be recovered further.

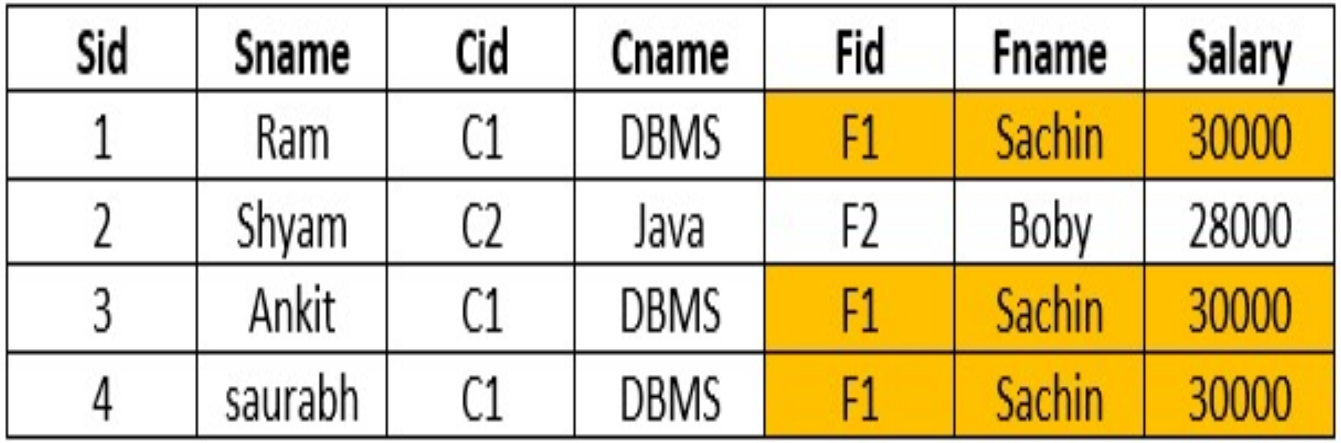
SQL:
DELETE FROM *University* WHERE *Sid=2*;



Sid	Sname	Cid	Cname	Fid	Fname	Salary
1	Ram	C1	DBMS	F1	Sachin	30000
~~2~~	~~Shyam~~	Deletion anomaly				
3	Ankit	C1	DBMS	F1	Sachin	30000
4	Saurabh	C1	DBMS	F1	Sachin	30000

Update Anomaly

- When the Database Administrator wants to change the salary of faculty F1 from 30000 to 40000 in above table University, then the database will update salary in more than one row due to data redundancy. So, this is an update anomaly in a table.



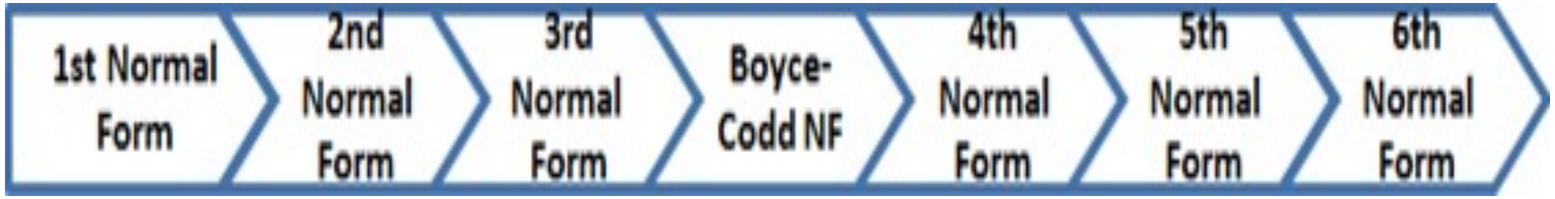
Sid	Sname	Cid	Cname	Fid	Fname	Salary
1	Ram	C1	DBMS	F1	Sachin	30000
2	Shyam	C2	Java	F2	Boby	28000
3	Ankit	C1	DBMS	F1	Sachin	30000
4	saurabh	C1	DBMS	F1	Sachin	30000

SQL:
UPDATE *University*
SET *Salary= 40000*
WHERE *Fid="F1";*

To remove all these anomalies, we need to normalize the data in the database.

Normal forms

- The Theory of Data Normalization in SQL is still being developed further. For example, there are discussions even on 6th Normal Form. **However, in most practical applications, normalization achieves its best in 3rd Normal Form**. The evolution of Normalization theories is illustrated below-



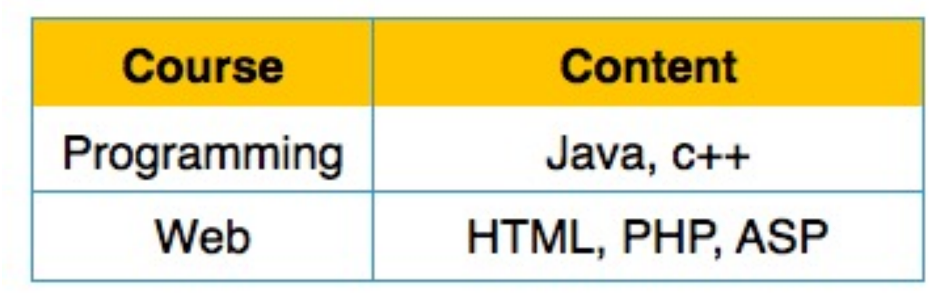
First Normal Form (1NF)

- According to the E.F. Codd, a relation will be in 1NF, if each cell of a relation contains only an atomic value.

1NF Example

- Example:

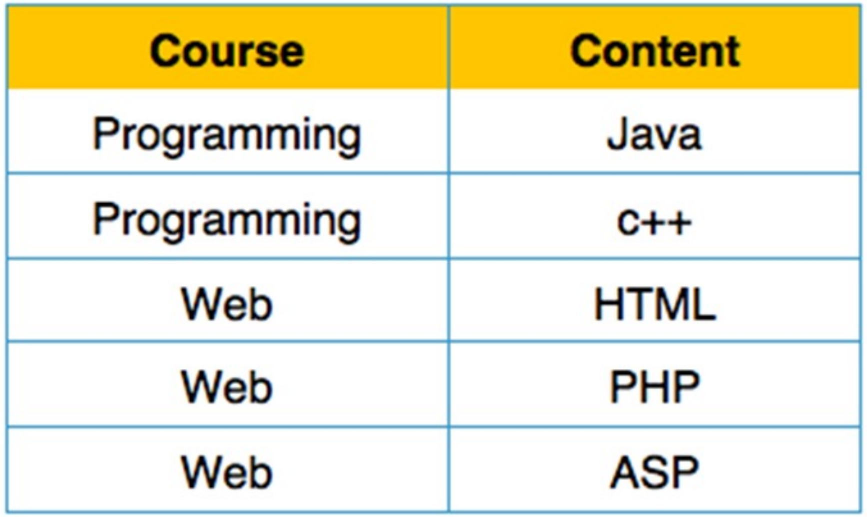
The following Course_Content relation is not in 1NF because the Content attribute contains multiple values.



Course	Content
Programming	Java, c++
Web	HTML, PHP, ASP

1NF Example (Cont..)

- The below relation student is in 1NF:



Course	Content
Programming	Java
Programming	c++
Web	HTML
Web	PHP
Web	ASP

Rules of 1NF

The official qualifications for 1NF are:

1. Each **attribute name** must be unique.
2. Each **attribute value** must be single.
3. Each **row** must be unique.

- Additional:
  - Choose a primary key.
- Reminder:

A primary key is ***unique***, ***not null***, ***unchanged***. A primary key can be either an attribute or combined attributes.

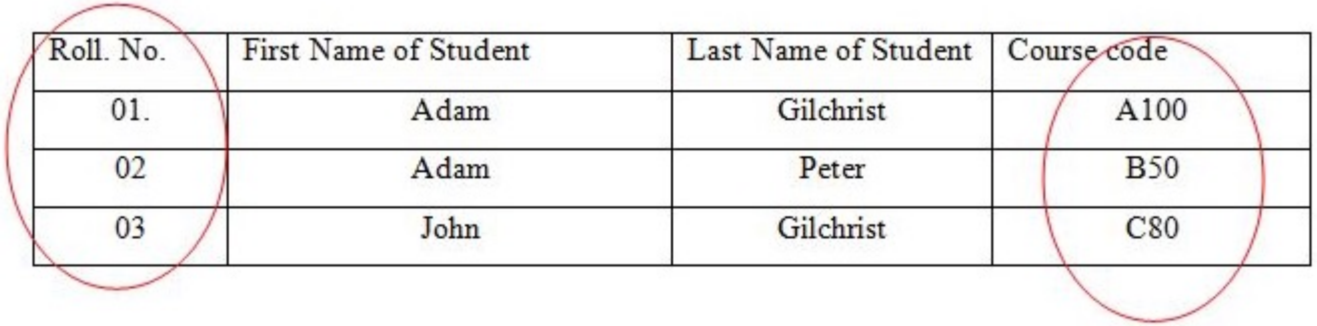
Second Normal Form (2NF)

- According to the E.F. Codd, a relation is in 2NF, if it satisfies the following conditions:
    - The table should be in the First Normal Form.
    - There should be no Partial Dependency.

Prime and Non Prime Attributes

Prime attributes: The attributes which are used to form a candidate key are called prime attributes.

Non-Prime attributes: The attributes which do not form a candidate key are called non-prime attributes.



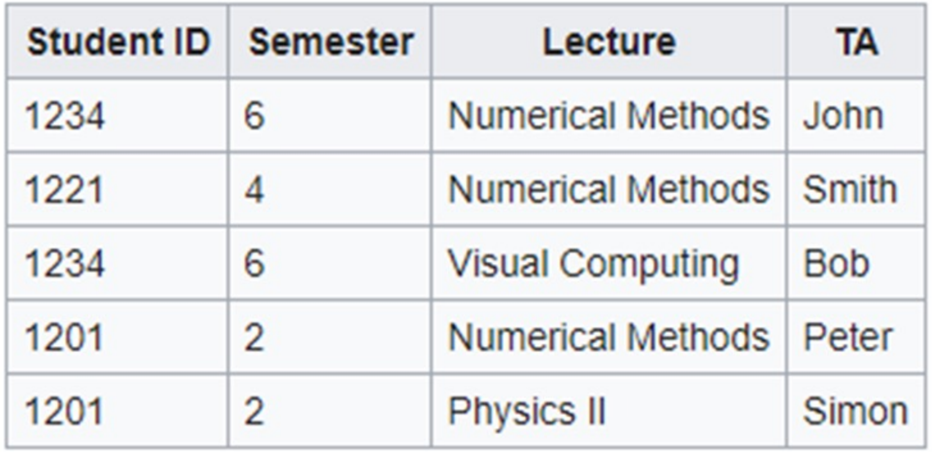
Roll. No.	First Name of Student	Last Name of Student	Course code
01.	Adam	Gilchrist	A100
02	Adam	Peter	B50
03	John	Gilchrist	C80

- Prime Attribute: Roll No., Course Code
- Non-Prime Attribute: First Name of Student, Last Name of Student

Functional Dependency

- A dependency FD: X → Y means that the values of Y are determined by the values of X. Two tuples sharing the same values of X will necessarily have the same values of Y.
- We illustrate this as:
  - X→ Y *(read as: X determines Y or Y depends on X)*

Functional Dependency



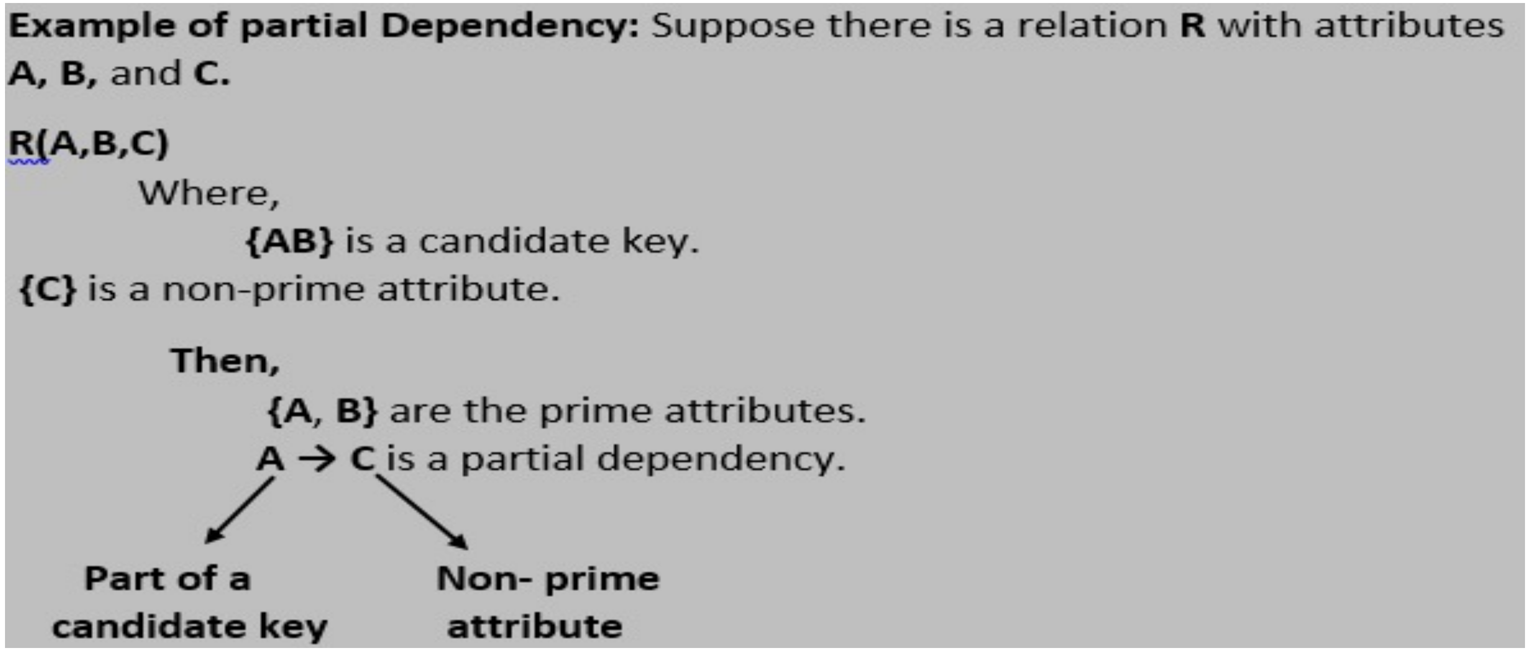
Student ID	Semester	Lecture	TA
1234	6	Numerical Methods	John
1221	4	Numerical Methods	Smith
1234	6	Visual Computing	Bob
1201	2	Numerical Methods	Peter
1201	2	Physics II	Simon

- Whenever two rows in this table feature the same StudentID, they also necessarily have the same Semester values. This basic fact can be expressed by a functional dependency:

$$\text{StudentID} \rightarrow \text{Semester.}$$

Partial Dependency

- If a non-prime attribute can be determined by the part of the candidate key in a relation, it is known as a partial dependency.



Example of partial Dependency: Suppose there is a relation **R** with attributes **A, B,** and **C.**

R(A,B,C)

Where,

{AB} is a candidate key.

{C} is a non-prime attribute.

Then,

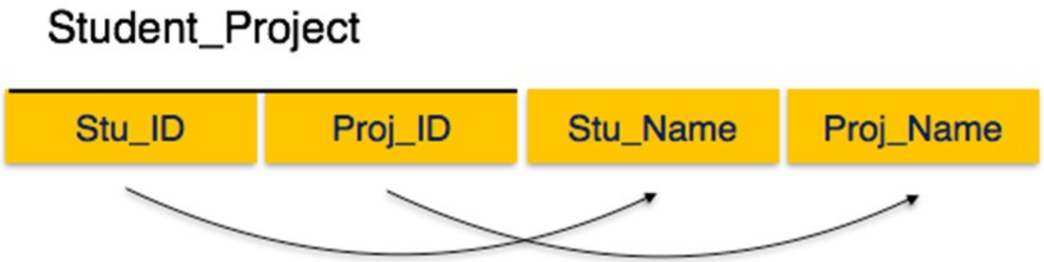
{A, B} are the prime attributes.

A → C is a partial dependency.

Part of a candidate key
Non- prime attribute

2NF Example

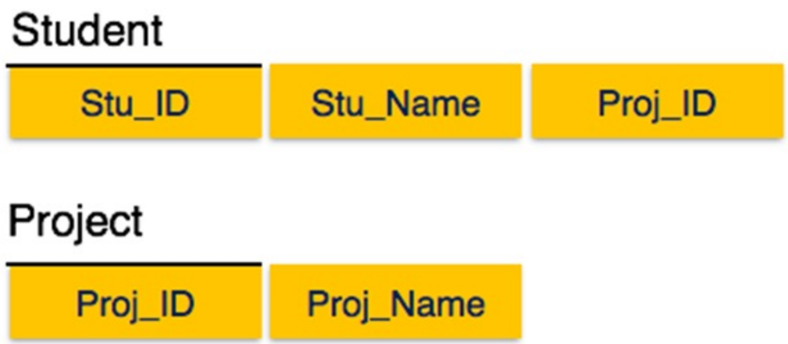
- In Student_Project relation that the prime key attributes are Stu_ID and Proj_ID.
- According to the rule, non-key attributes, i.e. Stu_Name and Proj_Name must be dependent upon both and not on any of the prime key attribute individually.
- But we find that Stu_Name can be identified by Stu_ID and Proj_Name can be identified by Proj_ID independently. This is called partial dependency, which is not allowed in Second Normal Form.



- **Candidate Keys:** {Stu_ID, Proj_ID}
- **Non-prime attribute:** Stu_Name, Proj_Name

2NF Example (Cont..)

- We broke the relation in two as depicted in the above picture. So there exists no partial dependency.



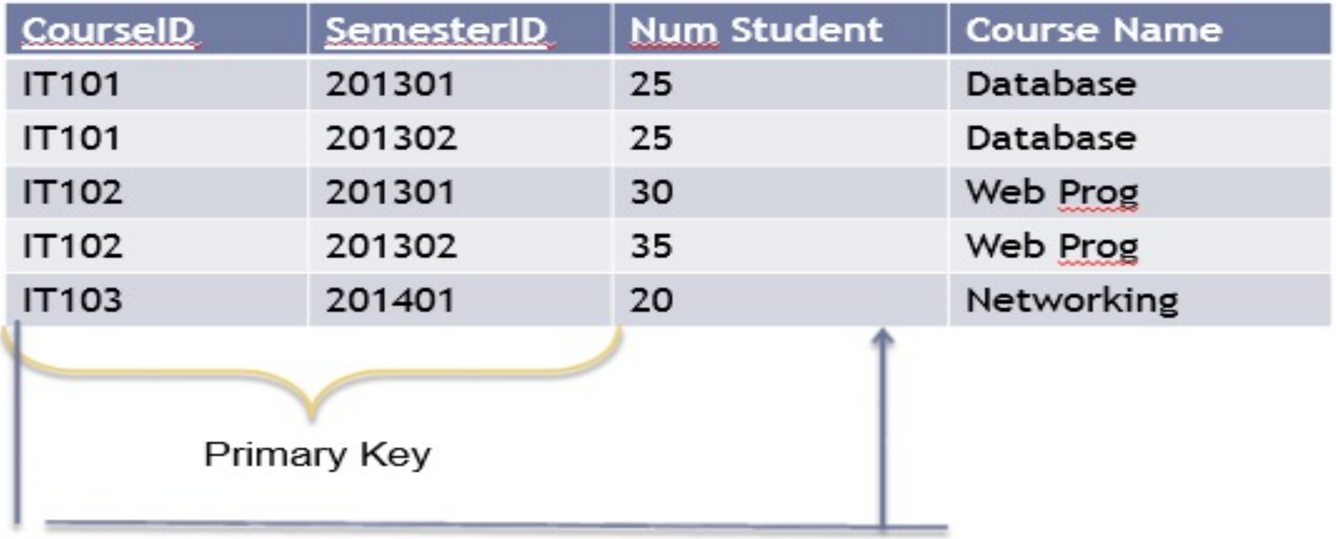
Student

Stu_ID	Stu_Name	Proj_ID

Project

Proj_ID	Proj_Name

Example 2NF



CourseID	SemesterID	Num Student	Course Name
IT101	201301	25	Database
IT101	201302	25	Database
IT102	201301	30	Web Prog
IT102	201302	35	Web Prog
IT103	201401	20	Networking

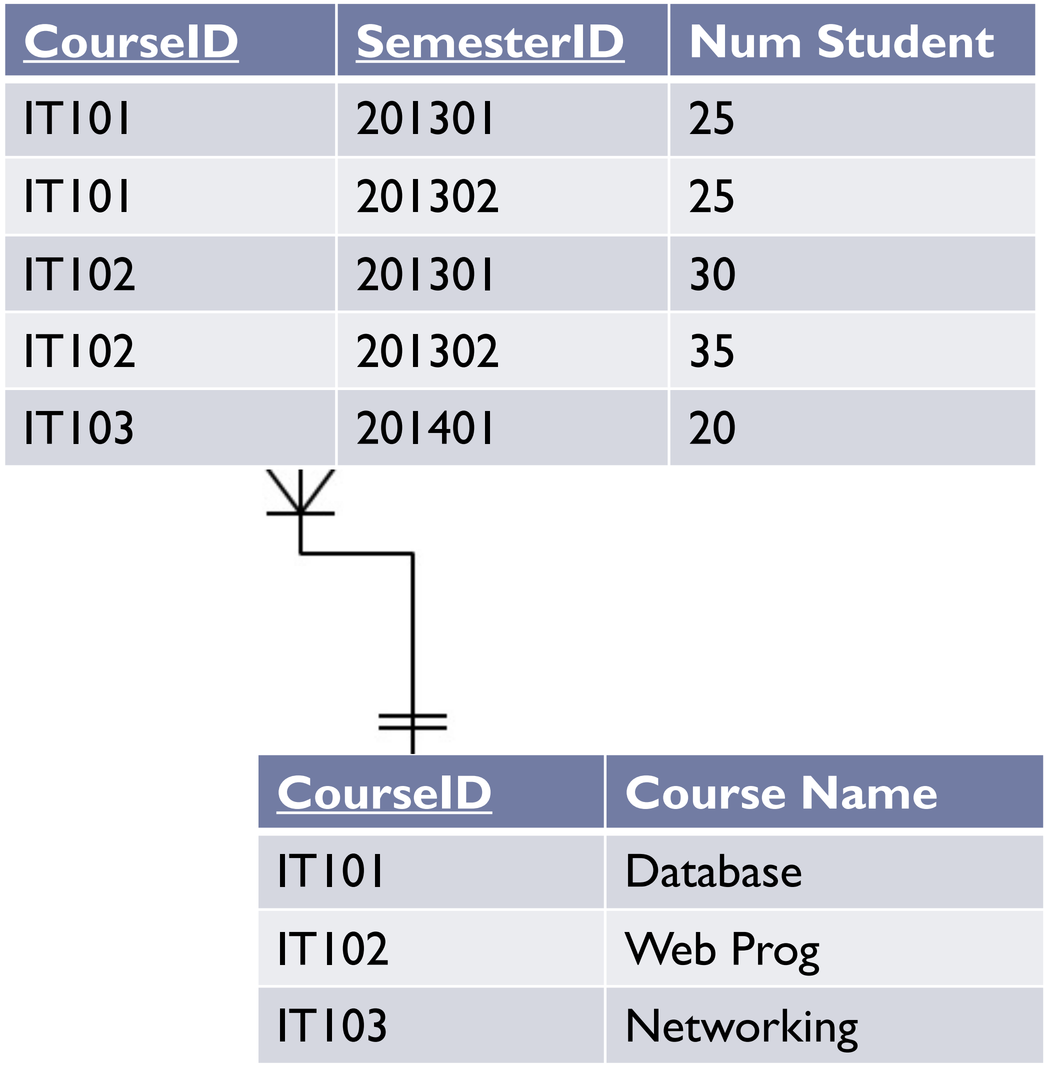
Primary Key

- The Course Name depends on only CourseID, a part of the primary key not the whole primary {CourseID, SemesterID}.It’s called partial dependency.
- **Solution:**
- *Remove **CourseID** and **Course Name** together to create a new table.*

Example 2NF (Cont..)

CourseID	Course Name
IT101	Database
IT101	Database
IT102	Web Prog
IT102	Web Prog
IT103	Networking

Done? Oh no, it is still not in 1NF yet.
Remove the repeating groups too.
Finally, connect the relationship.



CourseID	SemesterID	Num Student
IT101	201301	25
IT101	201302	25
IT102	201301	30
IT102	201302	35
IT103	201401	20

CourseID	Course Name
IT101	Database
IT102	Web Prog
IT103	Networking

Third Normal Form (3NF)

- According to the E.F. Codd, a relation is in third normal form (3NF) if it satisfies the following conditions:
  - ✓ It should be in the Second Normal form.
  - ✓ It should not have Transitive Dependency.
  - ✓ All transitive dependencies are removed to place in another table.

Transitive Dependency

- A functional dependency is said to be transitive if it is indirectly formed by two functional dependencies. For e.g.
- X -> Z is a transitive dependency if the following three functional dependencies hold true:

  X->Y

  Y does not ->X

  Y->Z

Transitive Dependency(Cont..)

- Let’s take an example to understand it better:

Book	Author	Author_age
Windhaven	George R. R. Martin	66
Harry Potter	J. K. Rowling	49
Dying of the Light	George R. R. Martin	66

{Book} ->{Author} (if we know the book, we knows the author name)

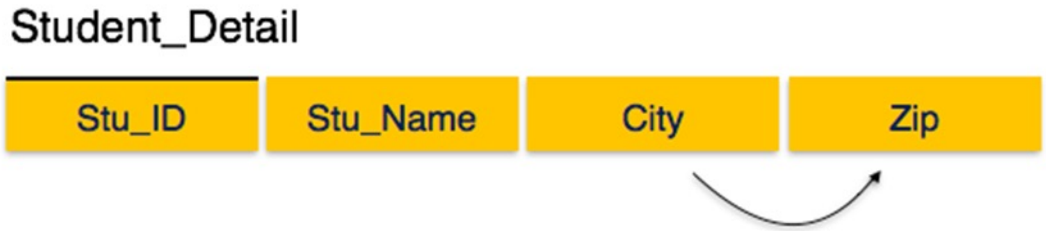
{Author} does not ->{Book}

{Author} -> {Author_age}

Therefore as per the rule of **transitive dependency**: {Book} -> {Author_age} should hold, that makes sense because if we know the book name we can know the author’s age.

3NF Example

- We find that in the above Student_detail relation, Stu_ID is the key and only prime key attribute.
- We find that City can be identified by Stu_ID as well as Zip itself.
- Neither Zip is a superkey nor is City a prime attribute. Additionally, Stu_ID → Zip → City, so there exists transitive dependency.



Student_Detail

Stu_ID	Stu_Name	City	Zip

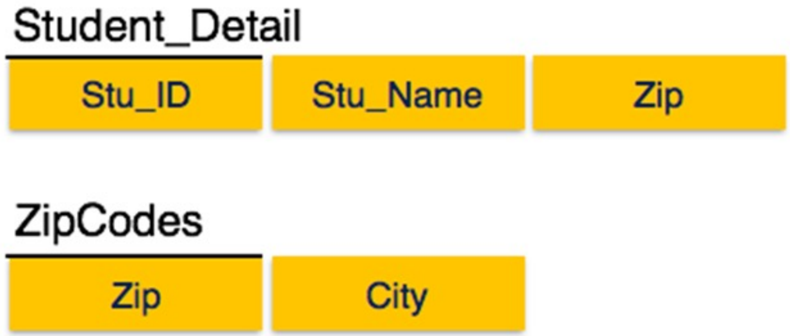
Candidate Key: {Stu_ID}

Prime attribute: Stu_ID

Non-prime attribute: {Stu_Name, City, Zip}

3NF Example (Cont..)

- To bring this relation into third normal form, we break the relation into two relations as follows −



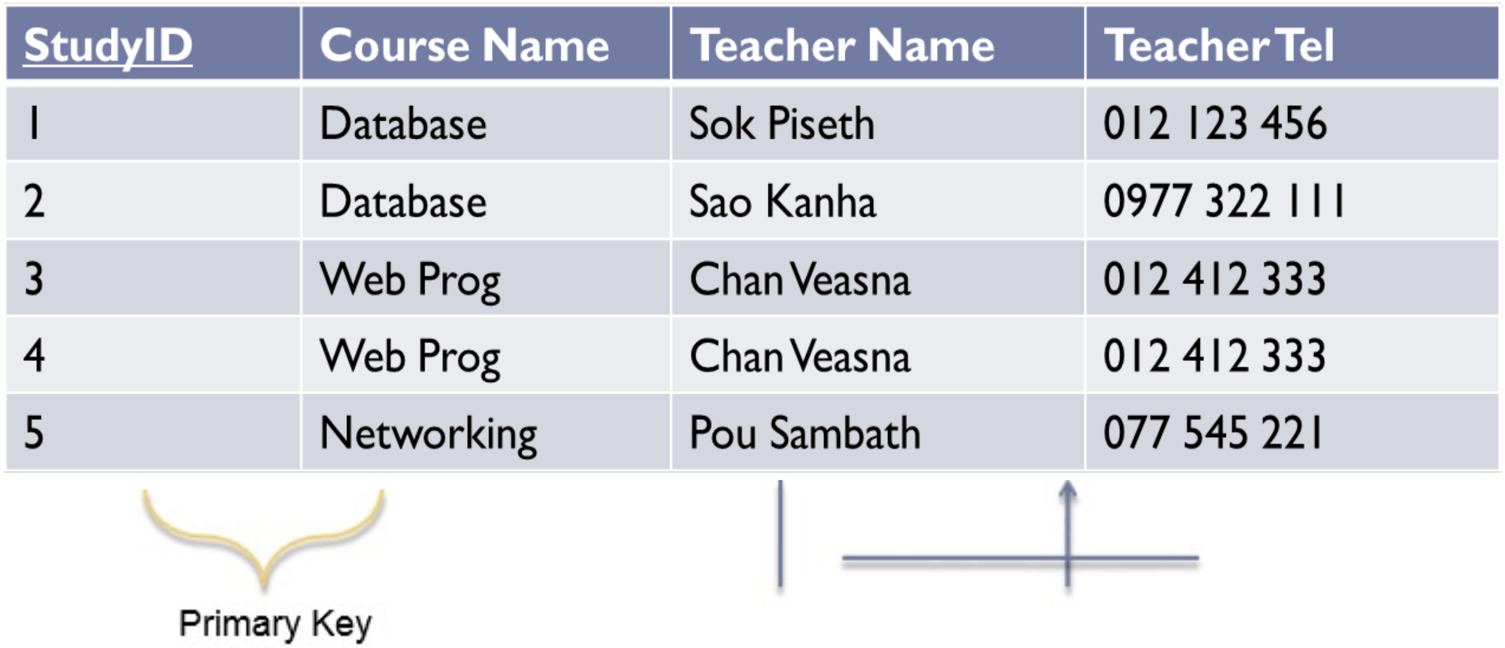
Student_Detail

Stu_ID	Stu_Name	Zip

ZipCodes

Zip	City

Example 3NF



<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

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

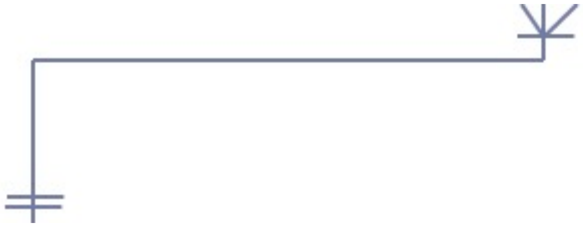
The Teacher Tel is a nonkey attribute, and the Teacher Name is also a nonkey atttribute. But Teacher Tel depends on Teacher Name. It is called **transitive dependency**.

Example 3NF

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.
Remove Repeating row.

StudyID	Course Name	T.ID
1	Database	T1
2	Database	T2
3	Web Prog	T3
4	Web Prog	T3
5	Networking	T4



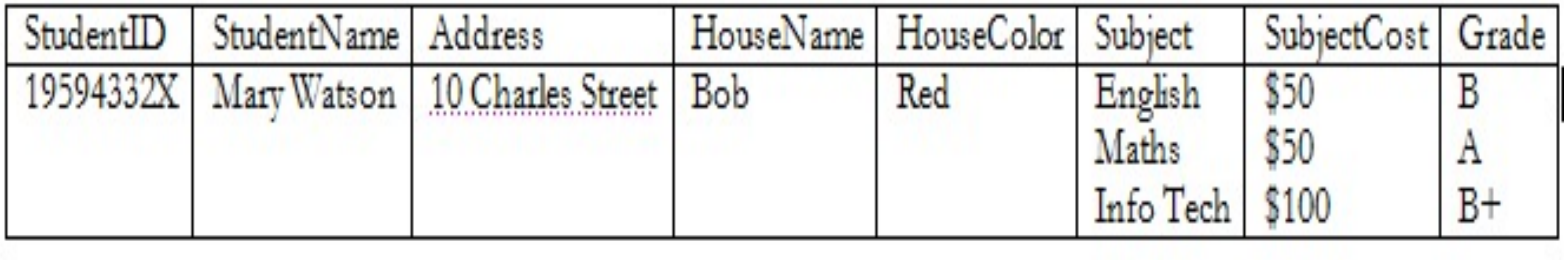
ID	Teacher Name	Teacher Tel
T1	Sok Piseth	012 123 456
T2	Sao Kanha	0977 322 111
T3	Chan Veasna	012 412 333
T4	Pou Sambath	077 545 221

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.

Example Table

- StudentID is the primary key.

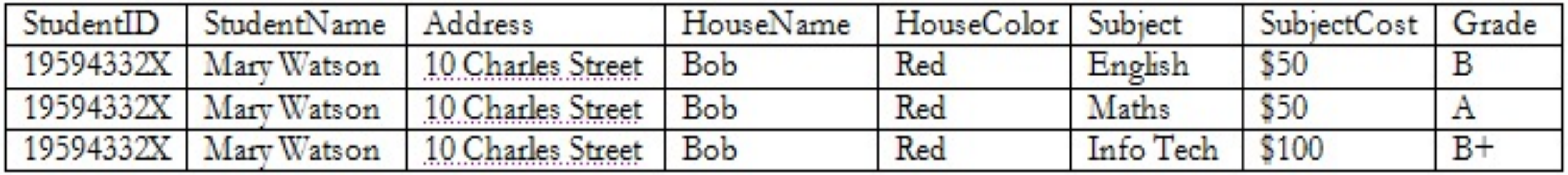


StudentID	StudentName	Address	HouseName	HouseColor	Subject	SubjectCost	Grade
19594332X	Mary Watson	10 Charles Street	Bob	Red	English Maths Info Tech	\$50 \$50 \$100	B A B+

How can you make it 1NF?

Example 1 (Cont..)

- Create new rows so each cell contains only one value



StudentID	StudentName	Address	HouseName	HouseColor	Subject	SubjectCost	Grade
19594332X	Mary Watson	10 Charles Street	Bob	Red	English	$50	B
19594332X	Mary Watson	10 Charles Street	Bob	Red	Maths	$50	A
19594332X	Mary Watson	10 Charles Street	Bob	Red	Info Tech	$100	B+

- But now the studentID no longer uniquely identifies each row. You now need to declare *studentID* **and** *subject* **together** to uniquely identify each row. So the new **key** is StudentID *and* Subject.

Is it 2NF?

Example 1 (Cont..)

- **Studentname** and **address** are dependent on studentID (which is part of the key) This is good. But they are **not** dependent on *Subject* (the *other* part of the key)
- **And 2NF requires…**

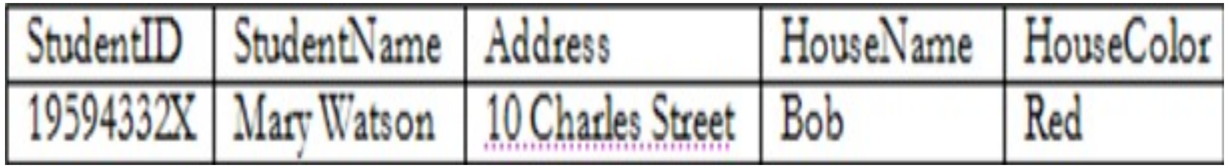
  All non-key fields are dependent on the ENTIRE key (studentID + subject)

Example 1 (Cont..)

- Make new tables
- Make a new table for each primary key field
- Give each new table its own primary key
- Move columns from the original table to the new table that matches their primary key…

Example (Cont..)

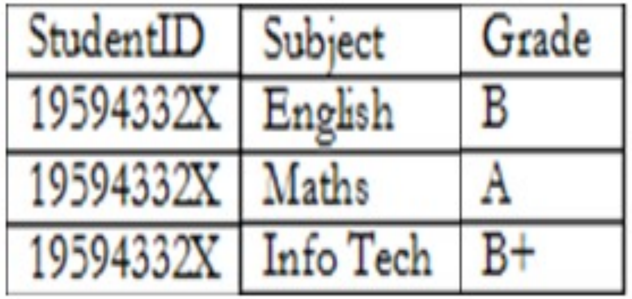
- STUDENT TABLE (key = StudentID)



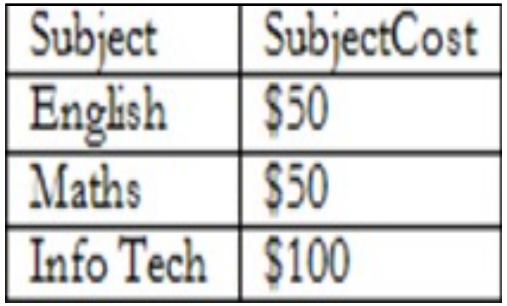
StudentID	StudentName	Address	HouseName	HouseColor
19594332X	Mary Watson	10 Charles Street	Bob	Red

- RESULTS TABLE (key = StudentID+Subject)

SUBJECTS TABLE (key = Subject)



StudentID	Subject	Grade
19594332X	English	B
19594332X	Maths	A
19594332X	Info Tech	B+



Subject	SubjectCost
English	$50
Maths	$50
Info Tech	$100

But is it 3NF?

Example 1 (Cont..)

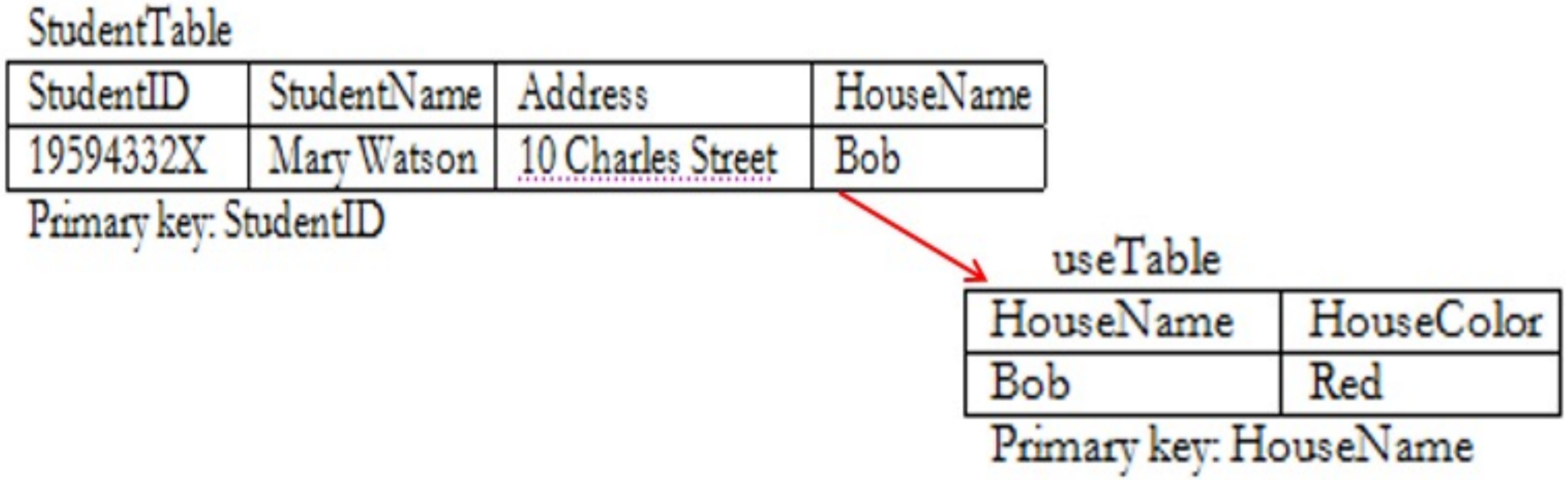
- **HouseName** is dependent on both **StudentID + HouseColour**

Or

- **HouseColour** is dependent on both **StudentID + HouseName**

- But either way, non-key fields are dependent on MORE THAN THE PRIMARY KEY (studentID). And 3NF says that non-key fields must depend on **nothing but the key**

Example 1 (Cont..)



StudentTable

StudentID	StudentName	Address	HouseName
19594332X	Mary Watson	10 Charles Street	Bob

Primary key: StudentID

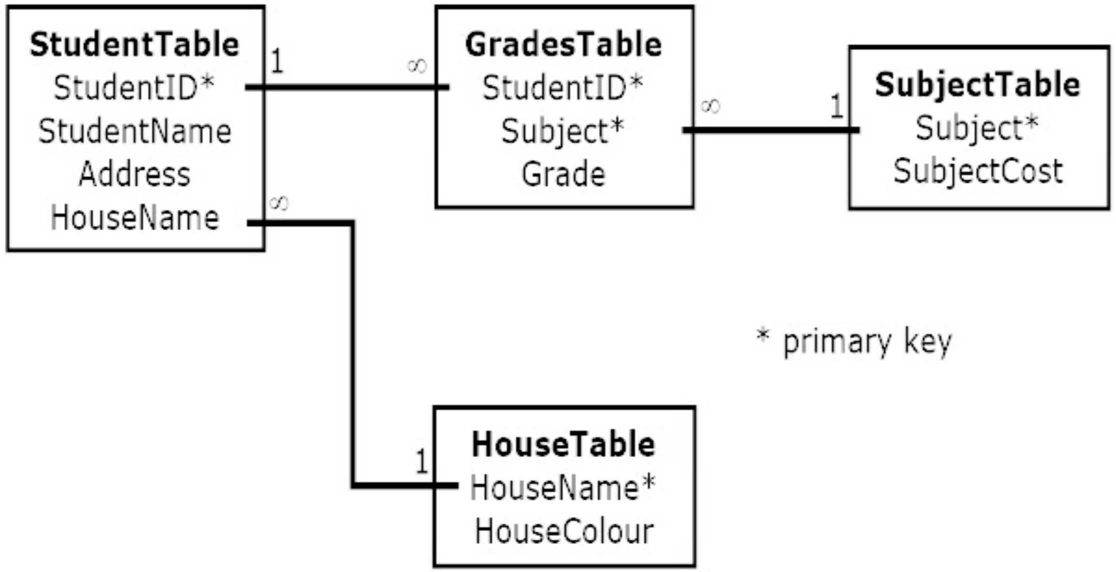
useTable

HouseName	HouseColor
Bob	Red

Primary key: HouseName

Example 1 (Cont..)

- The Final Scheme



StudentTable
StudentID*
StudentName
Address
HouseName

GradesTable
StudentID*
Subject*
Grade

SubjectTable
Subject*
SubjectCost

HouseTable
HouseName*
HouseColour

* primary key

Example 2 (Cont..)

Consider the employee table that is not in 1NF

EMPLOYEE_ID	NAME	JOB_CODE	JOB	STATE_CODE	HOME_STATE
E001	Alice	J01, J02	Chef, Waiter	26	Michigan
E002	Bob	J02, J03	Waiter, Bartender	56	Wyoming
E003	Reka	J01	Chef	56	Wyoming

Example 2 (Cont..)

All the entries are atomic and there is a composite primary key (employee_id, job_code) so the table is in the **first normal form (1NF)**.

EMPLOYEE_ID	NAME	JOB_CODE	JOB	STATE_CODE	HOME_STATE
E001	Alice	J01	Chef	26	Michigan
E001	Alice	J02	Waiter	26	Michigan
E002	Bob	J02	Waiter	56	Wyoming
E002	Bob	J03	Bartender	56	Wyoming
E003	Reka	J01	Chef	56	Wyoming

Example 2 (Cont..)

if you only know someone's employee_id, then you can determine their name, home_state, and state_code (because they should be the same person). This means name, home_state, and state_code are dependent on employee_id (a part of primary composite key). So, the table is not in **2NF**. We should separate them to a different table to make it 2NF.

Employee_role

EMPLOYEE_ID	JOB_CODE
E001	J01
E001	J02
E002	J02
E002	J03
E003	J01

Employee

EMPLOYEE_ID	NAME	STATE_CODE	HOME_STATE
E001	Alice	26	Michigan
E002	Bob	56	Wyoming
E003	Reka	56	Wyoming

jobs

JOB_CODE	JOB
J01	Chef
J02	Waiter
J03	Bartender

Example 2 (Cont..)

home_state is now dependent on state_code. So, if you know the state_code, then you can find the home_state value. Employee_id → home_state → state_code is in transitive relationship. To take this a step further, we should separate them again to a different table to make it 3NF.

Employee_role

EMPLOYEE_ID	JOB_CODE
E001	J01
E001	J02
E002	J02
E002	J03
E003	J01

Employee

EMPLOYEE_ID	NAME	STATE_CODE
E001	Alice	26
E002	Bob	56
E003	Reka	56

jobs

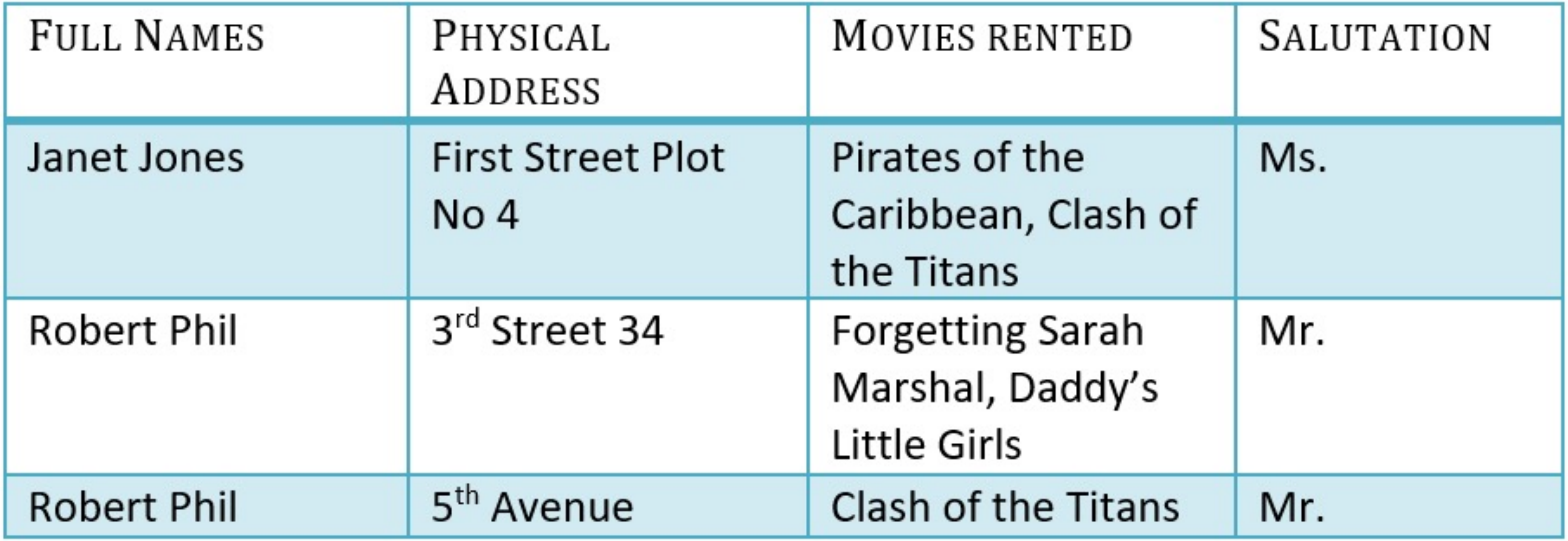
JOB_CODE	JOB
J01	Chef
J02	Waiter
J03	Bartender

States

STATE_CODE	HOME_STATE
26	Michigan
56	Wyoming

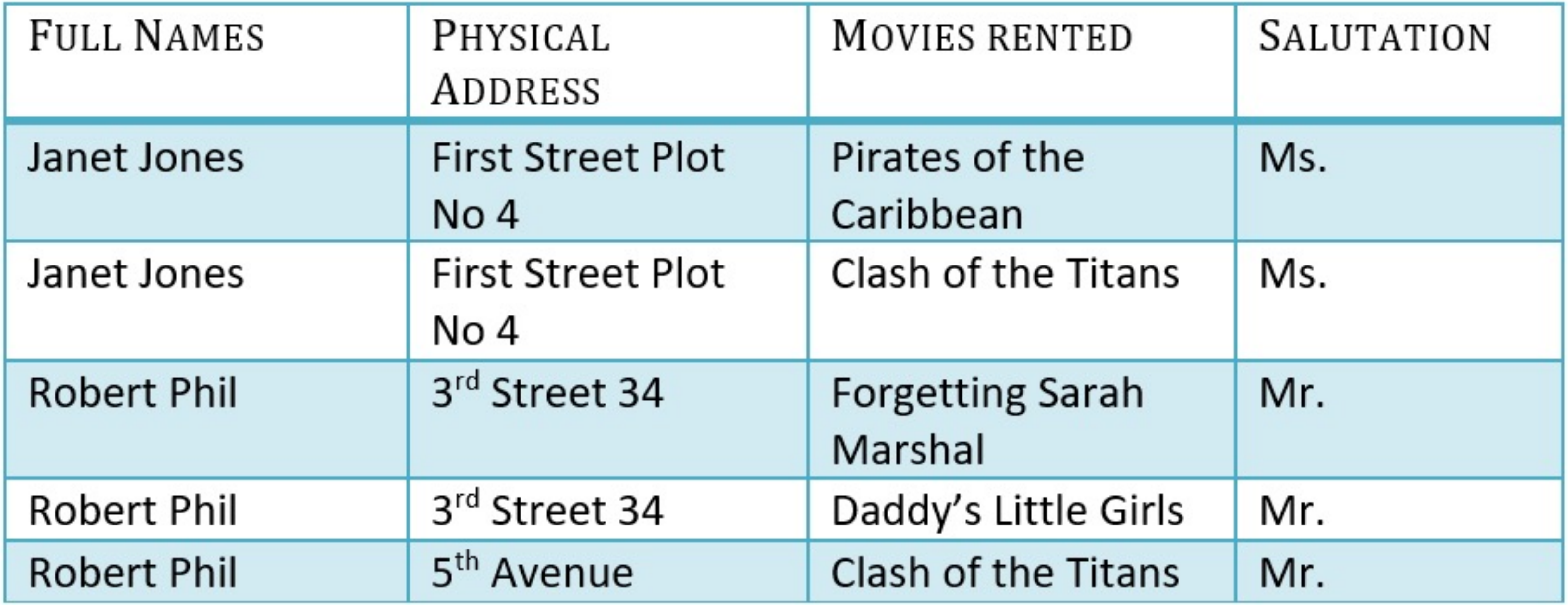
Example 3

Assume, a video library maintains a database of movies rented out. Without any normalization in database, all information is stored in one table as shown below



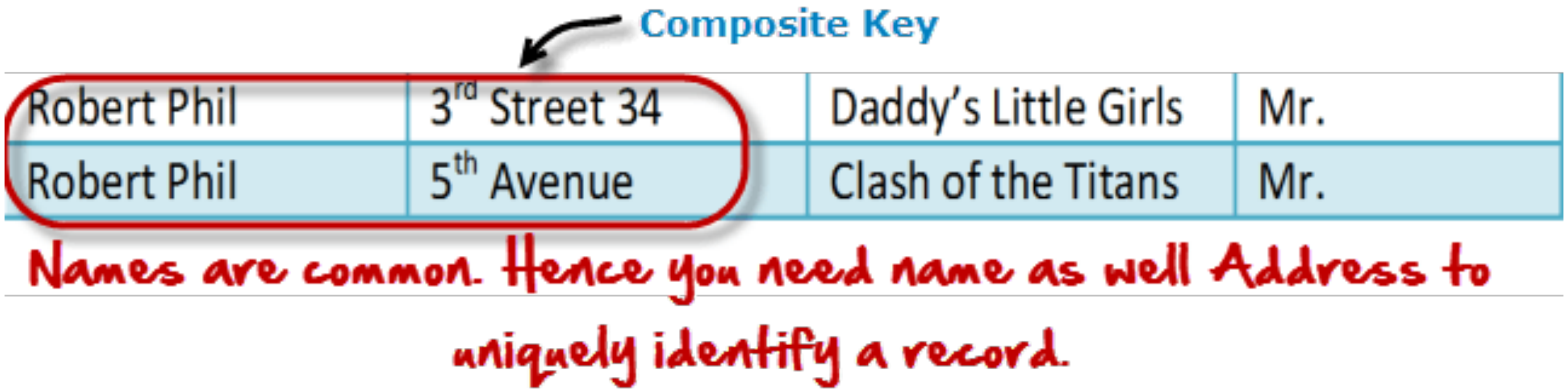
FULL NAMES	PHYSICAL ADDRESS	MOVIES RENTED	SALUTATION
Janet Jones	First Street Plot No 4	Pirates of the Caribbean, Clash of the Titans	Ms.
Robert Phil	3rd Street 34	Forgetting Sarah Marshal, Daddy's Little Girls	Mr.
Robert Phil	5th Avenue	Clash of the Titans	Mr.

Each table cell should contain a single value and record needs to be unique. The below table is in 1Normal Form (1NF)



Full Names	Physical Address	Movies rented	Salutation
Janet Jones	First Street Plot No 4	Pirates of the Caribbean	Ms.
Janet Jones	First Street Plot No 4	Clash of the Titans	Ms.
Robert Phil	3rd Street 34	Forgetting Sarah Marshal	Mr.
Robert Phil	3rd Street 34	Daddy’s Little Girls	Mr.
Robert Phil	5th Avenue	Clash of the Titans	Mr.

In our database, we have two people with the same name Robert Phil, but they live in different places. Hence, we require both Full Name and Address to identify a record uniquely. That is a composite key.



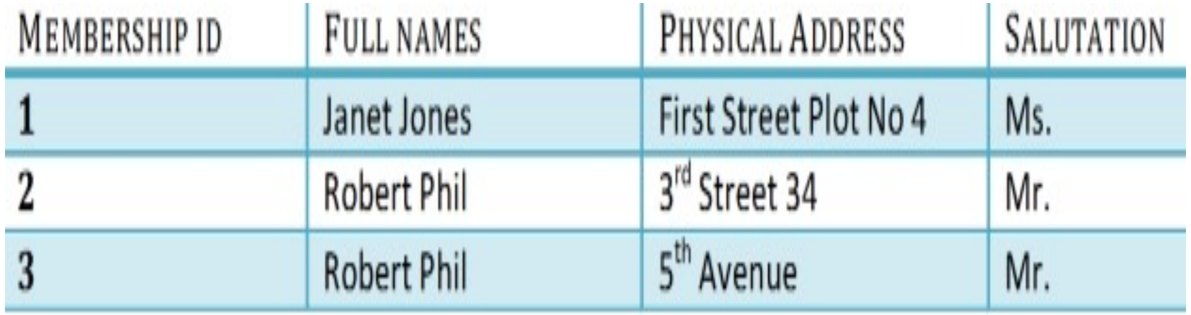
Composite Key

Robert Phil	3rd Street 34	Daddy's Little Girls	Mr.
Robert Phil	5th Avenue	Clash of the Titans	Mr.

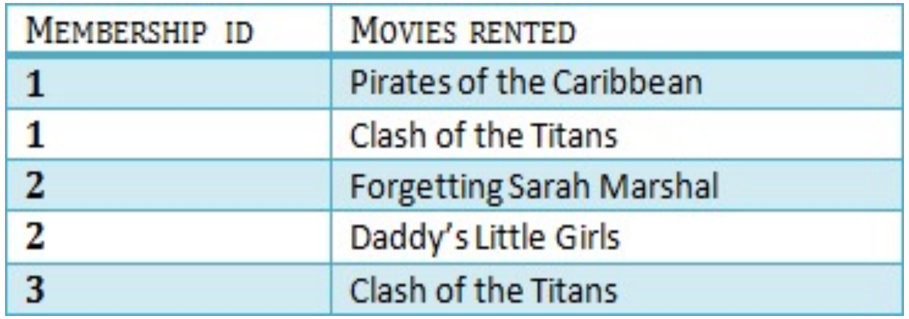
Names are common. Hence you need name as well Address to uniquely identify a record.

In 2NF

The table must be in 1NF and Single Column Primary Key that does not functionally dependent on any subset of candidate key relation

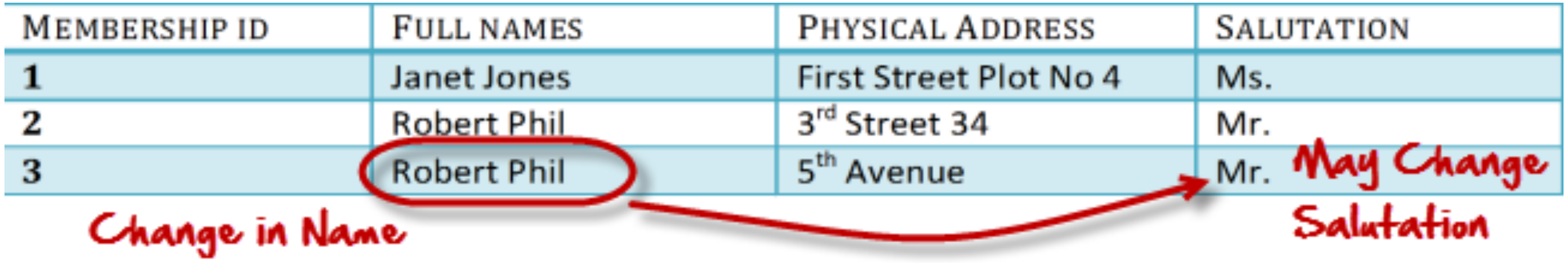


MEMBERSHIP ID	FULL NAMES	PHYSICAL ADDRESS	SALUTATION
1	Janet Jones	First Street Plot No 4	Ms.
2	Robert Phil	3rd Street 34	Mr.
3	Robert Phil	5th Avenue	Mr.



MEMBERSHIP ID	MOVIES RENTED
1	Pirates of the Caribbean
1	Clash of the Titans
2	Forgetting Sarah Marshal
2	Daddy's Little Girls
3	Clash of the Titans

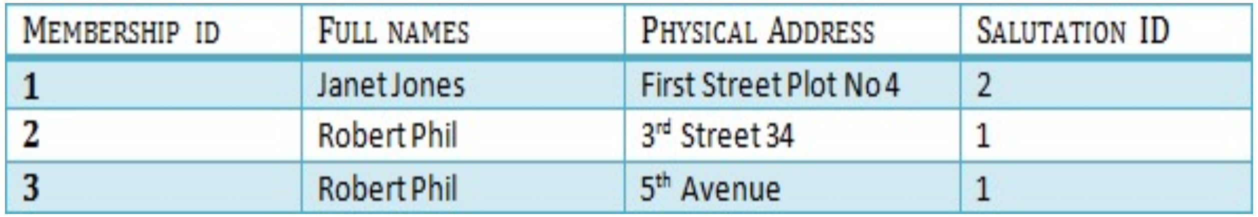
A transitive functional dependency is when changing a non-key column, might cause any of the other non-key columns to change.



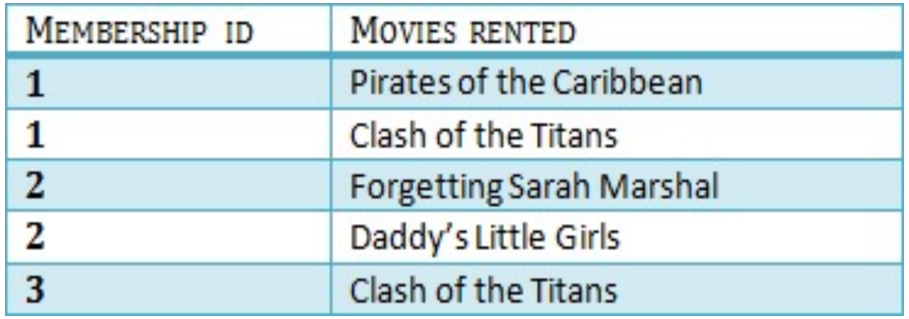
MEMBERSHIP ID	FULL NAMES	PHYSICAL ADDRESS	SALUTATION
1	Janet Jones	First Street Plot No 4	Ms.
2	Robert Phil	3rd Street 34	Mr.
3	Robert Phil	5th Avenue	Mr.

Change in Name
May Change
Salutation

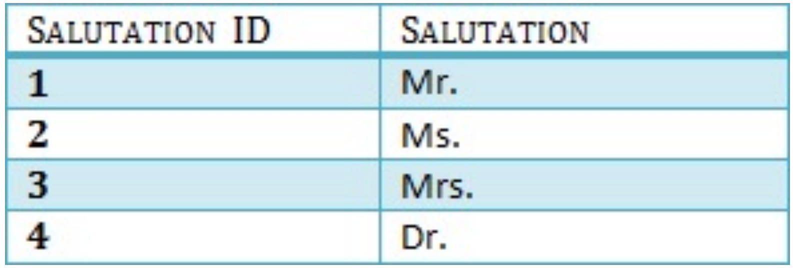
In 3NF
The table must be in 2NF and has no transitive functional dependencies



MEMBERSHIP ID	FULL NAMES	PHYSICAL ADDRESS	SALUTATION ID
1	Janet Jones	First Street Plot No 4	2
2	Robert Phil	3rd Street 34	1
3	Robert Phil	5th Avenue	1



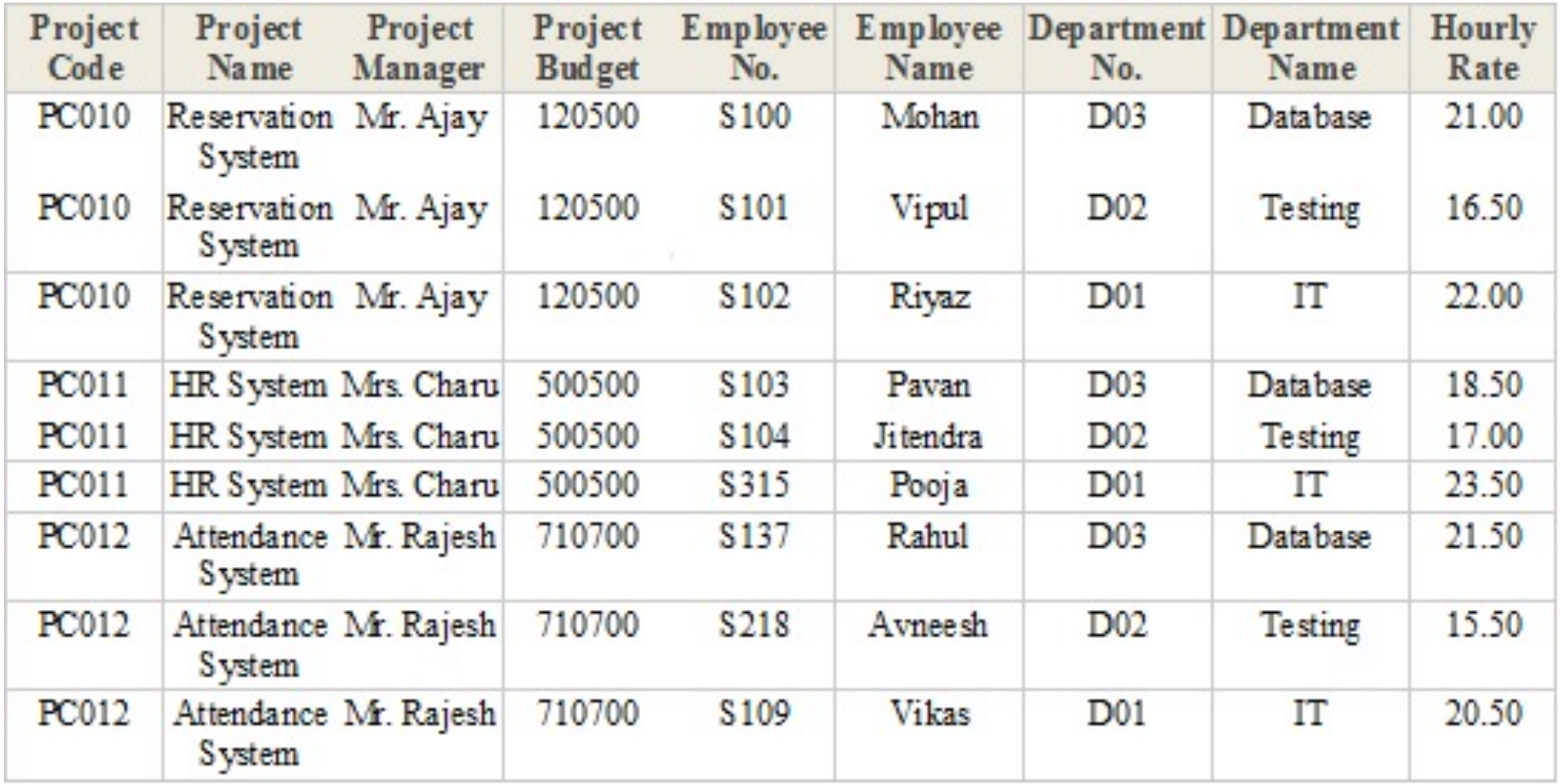
MEMBERSHIP ID	MOVIES RENTED
1	Pirates of the Caribbean
1	Clash of the Titans
2	Forgetting Sarah Marshal
2	Daddy's Little Girls
3	Clash of the Titans



SALUTATION ID	SALUTATION
1	Mr.
2	Ms.
3	Mrs.
4	Dr.

Example 4

Consider the table is in unnormalized form.

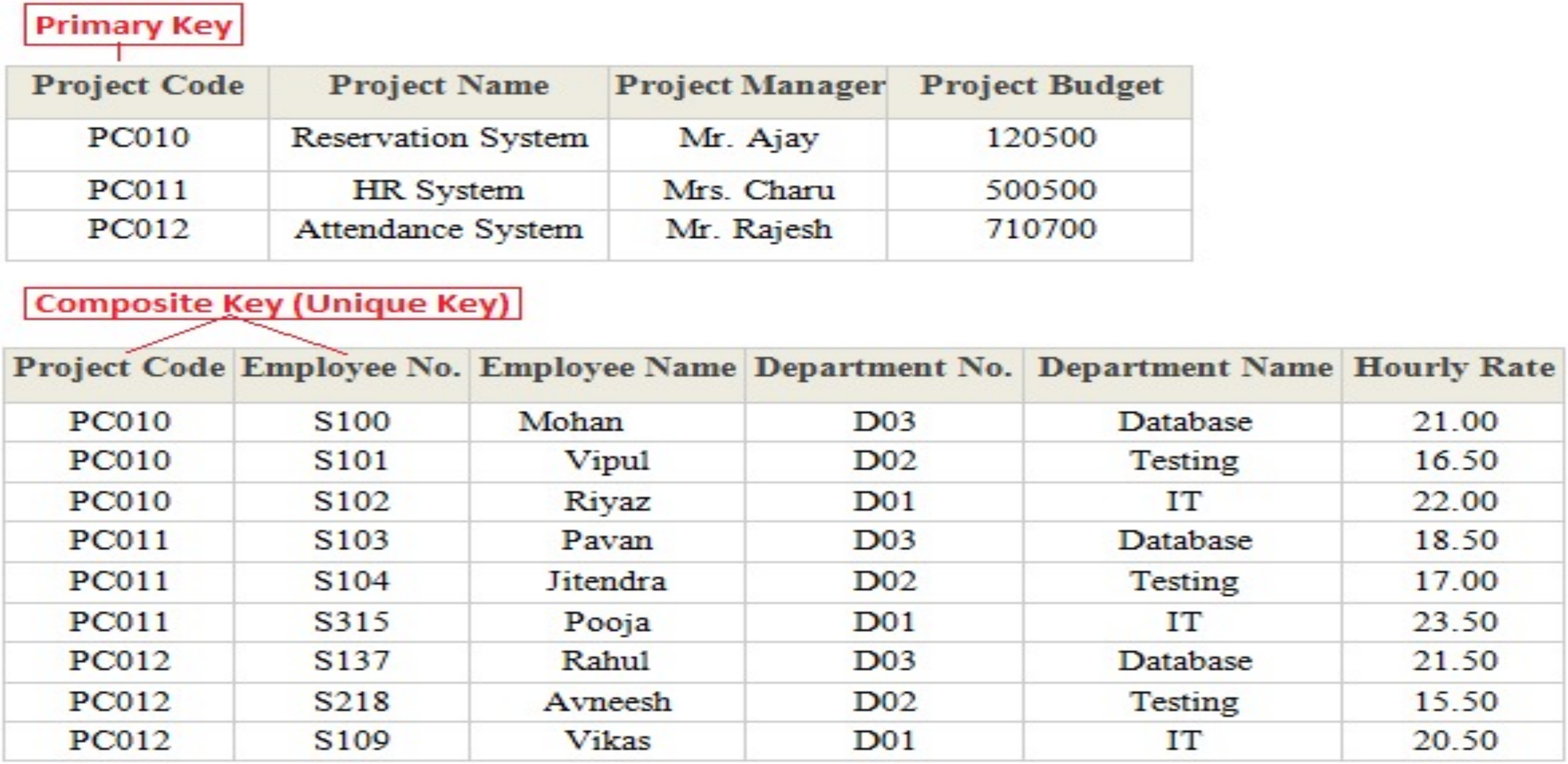


Project Code	Project Name	Project Manager	Project Budget	Employee No.	Employee Name	Department No.	Department Name	Hourly Rate
PC010	Reservation System	Mr. Ajay	120500	S100	Mohan	D03	Database	21.00
PC010	Reservation System	Mr. Ajay	120500	S101	Vipul	D02	Testing	16.50
PC010	Reservation System	Mr. Ajay	120500	S102	Riyaz	D01	IT	22.00
PC011	HR System	Mrs. Charu	500500	S103	Pavan	D03	Database	18.50
PC011	HR System	Mrs. Charu	500500	S104	Jitendra	D02	Testing	17.00
PC011	HR System	Mrs. Charu	500500	S315	Pooja	D01	IT	23.50
PC012	Attendance System	Mr. Rajesh	710700	S137	Rahul	D03	Database	21.50
PC012	Attendance System	Mr. Rajesh	710700	S218	Avneesh	D02	Testing	15.50
PC012	Attendance System	Mr. Rajesh	710700	S109	Vikas	D01	IT	20.50

UNF

A database table is said to be in 1NF if it contains no repeating fields/columns. The process of converting the UNF table into 1NF is as follows:

- Separate the repeating fields into new database tables along with the key from the unnormalized database table.
- The primary key of new database tables may be the composite key.



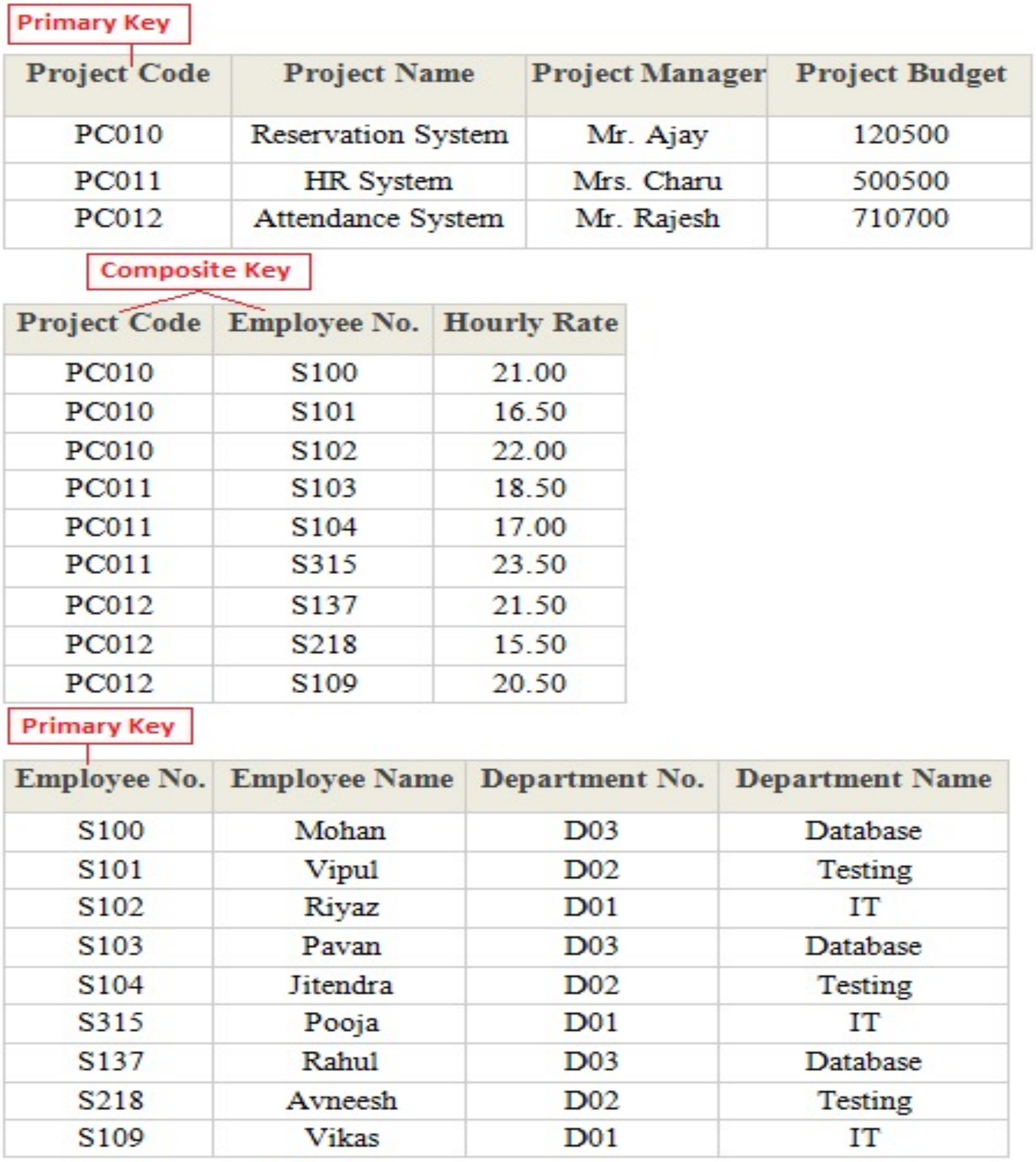
Primary Key

Project Code	Project Name	Project Manager	Project Budget
PC010	Reservation System	Mr. Ajay	120500
PC011	HR System	Mrs. Charu	500500
PC012	Attendance System	Mr. Rajesh	710700

Composite Key (Unique Key)

Project Code	Employee No.	Employee Name	Department No.	Department Name	Hourly Rate
PC010	S100	Mohan	D03	Database	21.00
PC010	S101	Vipul	D02	Testing	16.50
PC010	S102	Riyaz	D01	IT	22.00
PC011	S103	Pavan	D03	Database	18.50
PC011	S104	Jitendra	D02	Testing	17.00
PC011	S315	Pooja	D01	IT	23.50
PC012	S137	Rahul	D03	Database	21.50
PC012	S218	Avneesh	D02	Testing	15.50
PC012	S109	Vikas	D01	IT	20.50

1NF



Primary Key

Project Code	Project Name	Project Manager	Project Budget
PC010	Reservation System	Mr. Ajay	120500
PC011	HR System	Mrs. Charu	500500
PC012	Attendance System	Mr. Rajesh	710700

Composite Key

Project Code	Employee No.	Hourly Rate
PC010	S100	21.00
PC010	S101	16.50
PC010	S102	22.00
PC011	S103	18.50
PC011	S104	17.00
PC011	S315	23.50
PC012	S137	21.50
PC012	S218	15.50
PC012	S109	20.50

Primary Key

Employee No.	Employee Name	Department No.	Department Name
S100	Mohan	D03	Database
S101	Vipul	D02	Testing
S102	Riyaz	D01	IT
S103	Pavan	D03	Database
S104	Jitendra	D02	Testing
S315	Pooja	D01	IT
S137	Rahul	D03	Database
S218	Avneesh	D02	Testing
S109	Vikas	D01	IT

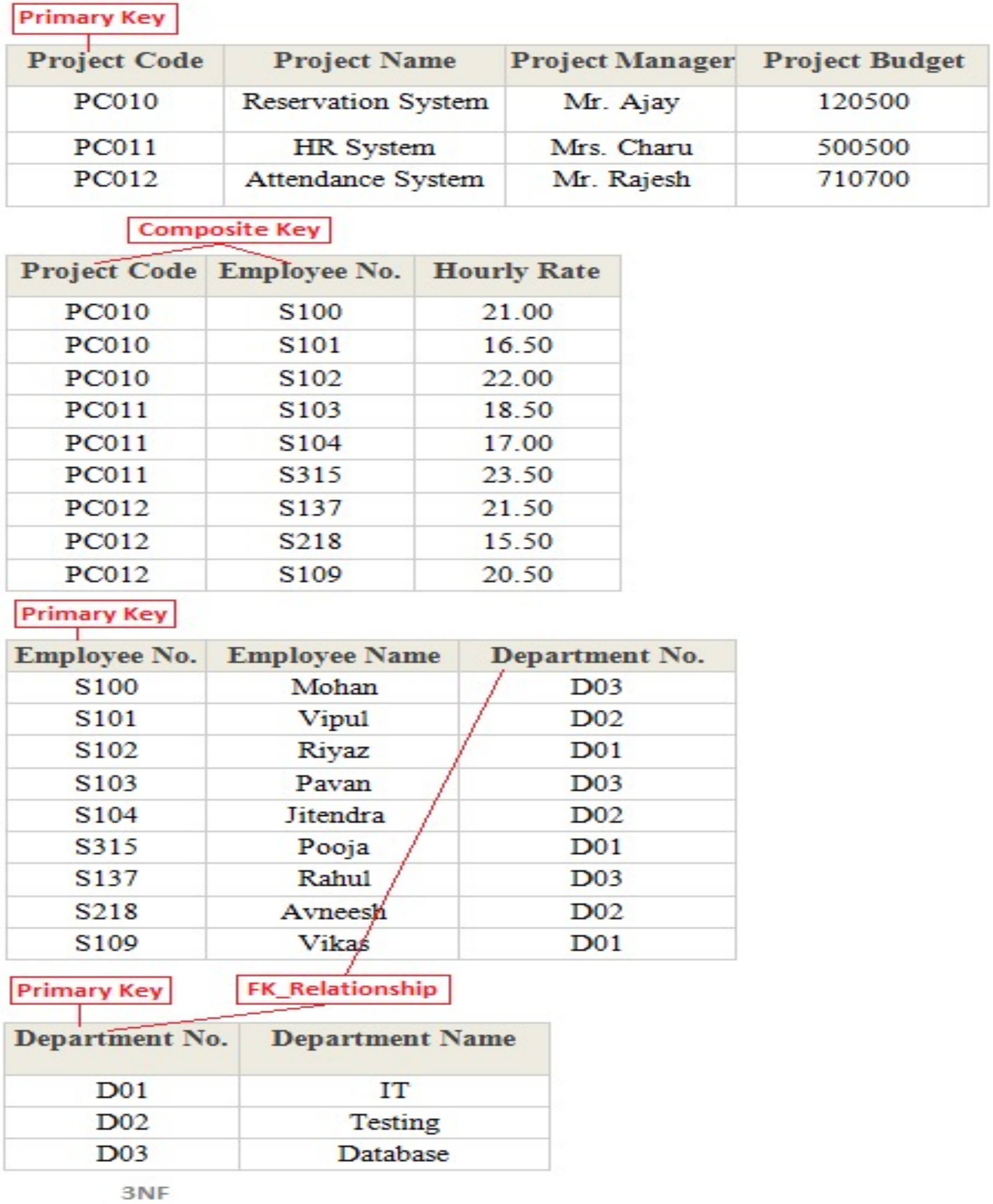
2NF

2 NF

- Remove partial dependencies field from table 2
- The composite key (project_code, employee No.)
- The employee name, department no, and department name is functially dependent on employee no.
- Hourly rate is not dependent because it is changeable to any moment.
- That`s why remove employee name, department no, and department from table 2 and make a new table.

3 NF

- Remove the transitive dependencies field from table 3
- The employee no → Department no → department name
- Make a separate table for transitive-dependent Fields.


Primary Key

Project Code	Project Name	Project Manager	Project Budget
PC010	Reservation System	Mr. Ajay	120500
PC011	HR System	Mrs. Charu	500500
PC012	Attendance System	Mr. Rajesh	710700

Composite Key

Project Code	Employee No.	Hourly Rate
PC010	S100	21.00
PC010	S101	16.50
PC010	S102	22.00
PC011	S103	18.50
PC011	S104	17.00
PC011	S315	23.50
PC012	S137	21.50
PC012	S218	15.50
PC012	S109	20.50

Primary Key

Employee No.	Employee Name	Department No.
S100	Mohan	D03
S101	Vipul	D02
S102	Riyaz	D01
S103	Pavan	D03
S104	Jitendra	D02
S315	Pooja	D01
S137	Rahul	D03
S218	Avneesh	D02
S109	Vikas	D01

Department No.	Department Name
D01	IT
D02	Testing
D03	Database

3NF

Example 5

- We will use the Student_Grade_Report table below, from a School database, as our example to explain the process for 1NF.

Student_Grade_Report (StudentNo, StudentName, Major, CourseNo, CourseName, InstructorNo, InstructorName, InstructorLocation, Grade)

Process for 1NF

- In the Student Grade Report table, the repeating group is the course information. A student can take many courses.
- Remove the repeating group. In this case, it's the course information for each student.
- Identify the PK for your new table.
- The PK must uniquely identify the attribute value (StudentNo and CourseNo).
- After removing all the attributes related to the course and student, you are left with the student course table (StudentCourse).
- The Student table (Student) is now in first normal form with the repeating group removed.
- The two new tables are shown below:

Student (StudentNo, StudentName, Major)

StudentCourse (StudentNo, CourseNo, CourseName, InstructorNo, InstructorName, InstructorLocation, Grade)

Example 2 (Cont..)

Student (StudentNo, StudentName, Major)

StudentCourse (StudentNo, CourseNo, CourseName, InstructorNo, InstructorName, InstructorLocation, Grade)

- To move to 2NF, a table must first be in 1NF.
- The Student table is already in 1NF because it has a single-column PK.
- When examining the Student Course table, we see that not all the attributes are fully dependent on the PK; specifically, all course information. The only attribute that is fully dependent is grade.
- Identify the new table that contains the course information.
- Identify the PK for the new table.
- The three new tables are shown below.

Example 2 (Cont..)

Student (StudentNo, StudentName, Major)

CourseGrade (StudentNo, CourseNo, Grade)

CourseInstructor (CourseNo, CourseName, InstructorNo, InstructorName, InstructorLocation)

Process for 3NF

- Eliminate all dependent attributes in transitive relationship(s) from each of the tables that have a transitive relationship.
- Create new table(s) with removed dependency.
- Check new table(s) as well as table(s) modified to make sure that each table has a determinant and that no table contains inappropriate dependencies.
- See the four new tables below.

Process for 3NF

Student (StudentNo, StudentName, Major)

CourseGrade (StudentNo, CourseNo, Grade)

Course (CourseNo, CourseName, InstructorNo)

Instructor (InstructorNo, InstructorName, InstructorLocation)

Process for 3NF

- At this stage, there should be no anomalies in third normal form.

Student (StudentNo, StudentName, Major)

CourseGrade (StudentNo, CourseNo, Grade)

Course (CourseNo, CourseName, InstructorNo)

Instructor (InstructorNo, InstructorName, InstructorLocation)

Thank you