Unit – 4

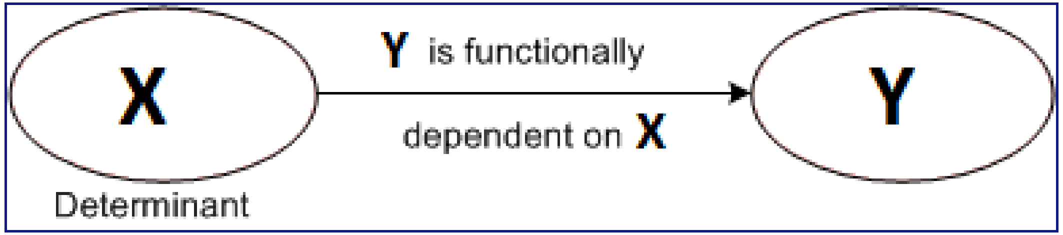
Database Normalization

Informal Design Guidelines for Relational Schemas

1. **Making sure that the Semantics of the Attributes is clear in the schema**
   - It ensure that each attribute's meaning is unambiguous.
   - To achieve this, we can use descriptive names, document definitions, maintain consistency.
2. **Reducing the Redundant Value in Tuples.**
   - It is crucial for optimizing database storage and improving efficiency.
   - By normalizing the database schema, eliminating repeating groups, and breaking down complex attributes, redundant values can be minimized.
3. **Reducing Null values in Tuples.**
   - It is important to improves data integrity, query efficiency and storage.
   - By using not null constraints and default value we can reduce it.
4. **Dissallowing the property of generating spurious (unwanted/incorrect) Tuples.**
   - It helps to ensure that query results are accurate and meaningful.
   - By enforcing proper join conditions, constraints, and data validation rules, the occurrence of spurious tuples can be minimized.

Functional Dependencies:

- Functional dependency in DBMS refers to a relationship that is present between attributes of any table that are dependent on each other.
- It plays a key role in differentiating good database design from bad database design.
- If R is a relation with attributes X and Y, a functional dependency between attributes is represented as X→ Y, which specifies Y is functionally dependent on X. Here X is a determinant set and Y is a dependent attribute.



Functional dependency between X and Y

Advantages of Functional Dependencies

- The benefits of functional dependency in a database management system can help businesses, organizations and companies prevent data redundancy.
- Functional dependency helps ensure the same data doesn't exist repetitively across a database or network of databases.
- Maintain the quality and integrity of data.
- It plays a key role in differentiating good database design from bad database design.

Types of Functional Dependency

There are many types of functional dependencies, depending on several criteria

- **Trivial functional dependency**
- **Non-trivial functional dependency**
- **Fully functional dependency**
- **Partial functional dependency**
- **Transitive dependency**
- **Multivalued dependency.**

Trivial functional dependency:

- The dependency of an attribute on a set of attribute is known as trivial functional dependency if the set of attributes includes that attribute.
- Symbolically: A→ B is trivial functional dependency if B is a subset of A.
- **Example 1:**

  Consider a table with two columns student_id and student_name.
  {student_id , student_name} → student_id is a trivial functional dependency as student_id is a subset of {student_id , student_name}
- **Example 2:**

  Consider relation R(A,B,C)
  The functional dependency
  (A,B) → A is trivial because A (the right-hand side) is a subset of (A,B) (the left-hand side).
  Similarly, A → A and B→ B is trivial because A is a subset of itself.

Non-trivial functional dependency

- If a functional dependency X→ Y holds true where Y is not a subset of X, then this dependency is called non-trivial functional dependency.

- Example:

Consider a table employee with three attributes: {emp_id, emp_name,emp_address}
The following functioan dependencies are non-trivial:

emp_id →emp_name (emp_name is not a subset of emp_id)
emp_id → emp_address (emp_address is not a subset of emp_id)

Fully functional dependency

- An attribute is fully functionally dependent on second attribute if and only if it is functionally dependent on the second attribute but not on any subset of the second attribute.
- For a relation R, in a functional dependency X→Y, Y is said to be fully functional dependent on X if Z→Y is false for all $Z \subset X$.

ProjectID	ProjectCost
001	1000
001	5000

EmpID	ProjectID	Days
E099	001	320
E056	002	190

{EmpID, ProjectID} → (Days)

Partial functional dependency

- An attribute is partial functionally dependent on second attribute if and only if it is functionally dependent on the second attribute and also dependency occur on any subset of the attributes of the composite determinant to identify its object
- For a relation R, in a functional dependency X→Y, Y is said to be partial functional dependent on X if by removal of some attributes from X and the dependency still holds.

name	**roll_no**	**course**
Ravi	2	DBMS
Tim	3	OS
John	5	Java

Here, the dependency,
{name,roll_no} →course is partial because **name→course** also holds.

Transitive dependency

- A functional dependency is said to be transitive if it is indirectly formed by two functional dependencies.
- For example, $X \rightarrow Z$ is a transitive dependency if the following three functional dependencies hold true:
    - $X \rightarrow Y$
    - $Y \rightarrow Z$
    - $X \rightarrow Z$
- Let's take an example of a table with following attributes

Movie_id	**Listing_id**	**Listing_type**	**DVD-Price**
M08	**L09**	**Crime**	**180**
M03	**L05**	**Drama**	**250**
M05	**L09**	**Crime**	**180**

Movie_id → Listing_id
Listing_id → Listing_type
Also, Movie_id→ Listing_type

Multivalued dependency

If there are more than one independent multi-valued attributes in a relation, that is called multivalued dependency.

Bike_model	Manuf_year	Color
M1001	2007	Black
M1001	2007	Red
M2012	2008	Black
M2012	2008	Red
M2222	2009	Black
M2222	2009	Red

Here columns **manuf_year** and **color** are **independent of each other** and **dependent on bike_model**. In this case these two columns are said to be **multivalued dependent on bike_model**. These dependencies can be represented as:

Bike_model →manuf_year

Bike_model → color

Functional Dependency Inference Rules (Armstrong's Axioms)

1. Reflexive Rule (IR1)

- In the reflexive rule, if Y is a subset of X, then X determines Y.
- If $X \supseteq Y$ then $X \rightarrow Y$
- Example:

  X = {a, b, c, d, e}
  Y = {a, b, c}

2. Augmentation Rule (IR2)

- The augmentation is also called as a partial dependency. In augmentation, if X determines Y, then XZ determines YZ for any Z.
- If $X \rightarrow Y$ then $XZ \rightarrow YZ$
- Example:

  For R(ABCD), if $A \rightarrow B$ then $AC \rightarrow BC$

3. Transitive Rule (IR3)

- In the transitive rule, if X determines Y and Y determine Z, then X must also determine Z.
- If X $\rightarrow$ Y and Y $\rightarrow$ Z then X $\rightarrow$ Z

-

4. Union Rule (IR4)

- Union rule says, if X determines Y and X determines Z, then X must also determine Y and Z.
- If X $\rightarrow$ Y and X $\rightarrow$ Z then X $\rightarrow$ YZ

5. Decomposition Rule (IR5)

- Decomposition rule is also known as project rule. It is the reverse of union rule.
- This Rule says, if X determines Y and Z, then X determines Y and X determines Z separately.
- If $X \rightarrow YZ$ then $X \rightarrow Y$ and $X \rightarrow Z$

6. Pseudo transitive Rule (IR6)

- In Pseudo transitive Rule, if X determines Y and YZ determines W, then XZ determines W.
- If $X \rightarrow Y$ and $YZ \rightarrow W$ then $XZ \rightarrow W$

7. Composition Rule (IR7)

- If $A \rightarrow B$ and $X \rightarrow Y$ hold, then $AX \rightarrow BY$ holds.

Problems without Normalization (Anomalies)

- A database anomaly is an inconsistency(fault) in the data resulting from an operation like an update, insertion, or deletion.
- There can be inconsistencies when a record is held in multiple places and not all of the copies are updated.
- The anomalies mostly occurs due to the absence of normalization.
- There are three types of anomalies:
    - **Insertion Anomaly:** An insertion anomaly is the inability to add data to the database due to the absence of other data.
    - **Update Anomaly:** An update anomaly is a data inconsistency that results from data redundancy and a partial update.
    - **Deletion Anomaly:** A deletion anomaly occurs when you delete a record that may contain attributes that shouldn't be deleted.

Normalization

- Normalization is a database design technique which organizes tables in a manner that reduces redundancy and dependency of data.
- It divides larger tables to smaller tables and links them using relationship.

Benefits of Normalization

- Reduce the amount of storage needed to store the data.
- Avoids unnecessary data conflicts that may creep in because of multiple copies of the same data getting stored.

Types of Normalization

- ✓ The degree of normalization is defined by normal forms. Each normal form is a set of conditions on a schema that guarantees certain properties relating to redundancy and update anomalies.
- ✓ In general, 3NF is considered good enough.
- ✓ The normal forms in an increasing level of normalization are :
  - **First Normal Form (1NF) :** remove repeating groups
  - **Second Normal Form (2NF) :** remove partial dependencies
  - **Third Normal Form (3NF) :** remove transitive dependencies
  - **Boyce-Codd Normal Form (BCNF) :** remove remaining functional dependency anomalies
  - **Fourth Normal Form (4NF) :** remove multi-valued dependency
  - **Fifth Normal Form (5NF) :** remove remaining anomalies

1NF (First Normal Form)

If a relation contain composite or multi-valued attribute, it violates first normal form, or a relation is in first normal form if it does not contain any composite or multi-valued attribute.

Simply, A relation R is in 1NF if it holds the following two properties:

- If R has no multi-valued attributes
- If R has no composite attributes i.e.; it must have simple attributes

First normal form (1NF)

As per the rule of first normal form,

- An attribute (column) of a table cannot hold multiple values.
- It should hold only **atomic** (forming a single unit) values.
- Each record needs to be unique.

Atomic Value

An atomic value is a value that cannot be divided.

Employee Table			
Emp_Id	**Name**	**MobileNo**	**Dept**
E1001	John	3343434	Sales
E1002	Smith	5454546, 5454547	Purchase
E1003	Kim	6565656, 6565657	IT
E1004	Arnold	90876545	Sales
E1005	Moba	8787656	Accounts

First Normal Form(1NF)

Not a Atomic value

Employee Table			
Emp_Id	**Name**	**MobileNo**	**Dept**
E1001	John	3343434	Sales
E1002	Smith	5454546, 5454547	Purchase
E1003	Kim	6565656, 6565656	IT
E1004	Arnold	90876545	Sales
E1005	Moba	8787656	Accounts

After 1st Normalisation

Employee Table			
Emp_Id	**Name**	**MobileNo**	**Dept**
E1001	John	3343434	Sales
E1002	Smith	5454546	Purchase
E1002	Smith	5454547	Purchase
E1003	Kim	6565656	IT
E1003	Kim	6565657	IT
E1005	Moba	8787656	Accounts

2NF (Second Normal Form)

A relation R(table) is said to be in 2NF if both the following conditions hold:

- Table is in 1NF
- No non-prime attributes of R fully depends upon the proper subset of any candidate key of table

An attribute that is not part of any candidate key is known as non-prime attribute. Simply a relation is in 2NF if there are **no partial dependencies** between the primary key and non-primary keys of given relation R.

2nd Normal Form Definition

- It should be in the First Normal form.
- All non-key attributes are fully functional dependent on the primary key. In simple words it should not have Partial Dependency.

Emp_id	Qualification	Age
111	MA	38
111	BTech	38
222	MCA	38
333	MS	40
333	MBA	40

Composite Key: Emp_ID+Qualiication

Non-Key Attribute

- Composite primary key : **Emp_ID+Qualiication**
- Non-key attribute: **Age**

How to find age of employee

Example:

2nd Normal Form Definition

➢ To make the table complies with 2NF we can break it in two tables like this:

emp_id	Age
111	38
222	38
333	40

Emp_id: Primary Key

Relationship
1:Many

emp_id	Qualification
111	MA
111	BTech
222	MCA
333	MS
333	MBA

Emp_id:Foreign Key

1st NF

- It should hold only **atomic (**forming a single unit**)** values.
- Each record needs to be unique.

2nd NF

- Non-key attribute should depend on single key

3NF (Third Normal Form)

A table design is said to be in 3NF if both the following conditions hold:

- Table must be in 2NF
- No transitive functional dependency between the attributes of given table.

In other words, 3NF can be explained like this: A table is in 3NF if it is in 2NF and for each functional dependency X→ Y at least one of the following conditions hold:

- X is a super key of table
- Y is a prime attribute of table

3rd Normal Form Definition

- It is in second normal form
- There is no transitive functional dependency.
- **A transitive functional dependency is when changing a non-key column, might cause any of the other non-key columns to change.**

Emp_ID	**Name**	**ZIP**	**City**
E100	Harry	20101	Noida
E101	Stephan	02228	Boston
E102	Lan	60007	Chicago
E103	Katharine	06389	Norwich
E104	John	02228	Boston

Emp_id: Primary Key

Change in zip requires change in City

Emp_id -> Zip -> City	**City dependent on Zip dependent on Emp_id**
Emp_id -> City	**City dependent on Emp_id** City Indirectly dependent on Emp_Id is called **Transitive Functional Dependency** SOLUTION: Create New Table for Zip and City

3rd Normal Form Solution

Emp Table

Emp_ID	Name	ZIP
E100	Harry	20101
E101	Stephan	02228
E102	Lan	60007
E103	Katharine	06389
E104	John	02228

Emp_id: Primary Key

Zip: Foreign Key

Zip Table

ZIP	City
20101	Noida
02228	Boston
60007	Chicago
06389	Norwich

Zip: Primary Key

BCNF (Boyce Codd Normal Form)

- It is an advance version of 3NF that’s why it is also referred as 3.5NF.
- BCNF is stricter than 3NF. A table complies with BCNF if it is in 3NF and for every functional dependency X→ Y, X should be the super key of the table.

Boyce-Codd Normal Form (BCNF)

- **Boyce-Codd Normal** Form or BCNF is an extension to the third normal form, and is also known as 3.5 Normal Form.
- In BCNF if every functional dependency **A → B**, then **A** has to be the **Super Key(** is a key that uniquely identifies rows in a table) of that particular table.

student_id	subject	professor
101	Java	P.Java
101	C++	P.Cpp
102	Java	P.Java2
103	C#	P.Chash
104	Java	P.Java

- One student can enroll for multiple subjects.
- For each subject, a professor is assigned to the student.
- And, there can be multiple professors teaching one subject.

student_id + subject(primary key , subject as prime attribute)
professor -> subject

subject is a prime attribute, professor is a non-prime attribute, which is not allowed by BCNF.

Boyce-Codd Normal Form (BCNF)

- . Student able

student_id	professor_id
101	1
101	2
102	3
103	4
104	1

Professors table

professor_id	professor	subject
1	P.Java	Java
2	P.Cpp	C++
3	P.Java2	Java
4	P.Chash	C#

Difference/Comparison of BCNF and 3NF

3NF	BCNF
A table or a relation is considered to be in 3NF only if the table is already inn 2NF and there is no non-prime attribute transitively dependent on the candidate key of a relation	BCNF is considered to be the stronger than 3NF. The relation R to be in BCNF must be in 3NF. And wherever a non-trivial functional dependency $A \rightarrow B$ holds in relation R, then A must be a super key of relation R.
No non-prime attribute must be transitively dependent on the candidate key.	For any trivial dependency in a relationship R say $X \rightarrow Y$, X should be a super key of relation R.
3NF can be obtained without sacrificing all dependencies.	Dependencies may not be preserved in BCNF.
Lossless decomposition can be achieved in 3NF.	Lossless decomposition is hard to achieve in BCNF.

4NF (Fourth Normal Form)

A table design is said to be in 4NF if the following conditions hold:

- Table must be in BCNF or in 3NF
- If it contains no multi-valued attributes

4th Normal Form

- A relation will be in 4NF if it is in Boyce Codd normal form
- It 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.

What is multi-valued dependency?

Student_ID	Course	Hobby
21	Computer	Dancing
21	Math	Singing
34	Chemistry	Dancing
74	Biology	Cricket
59	Physics	Hockey

Student_id -> Course
Student_id -> Hobby

➢ COURSE and HOBBY are two independent entity. Hence, there is no relationship between COURSE and HOBBY.

➢ To remove **multi-valued dependency** decompose it into two tables:

4th Normal Form

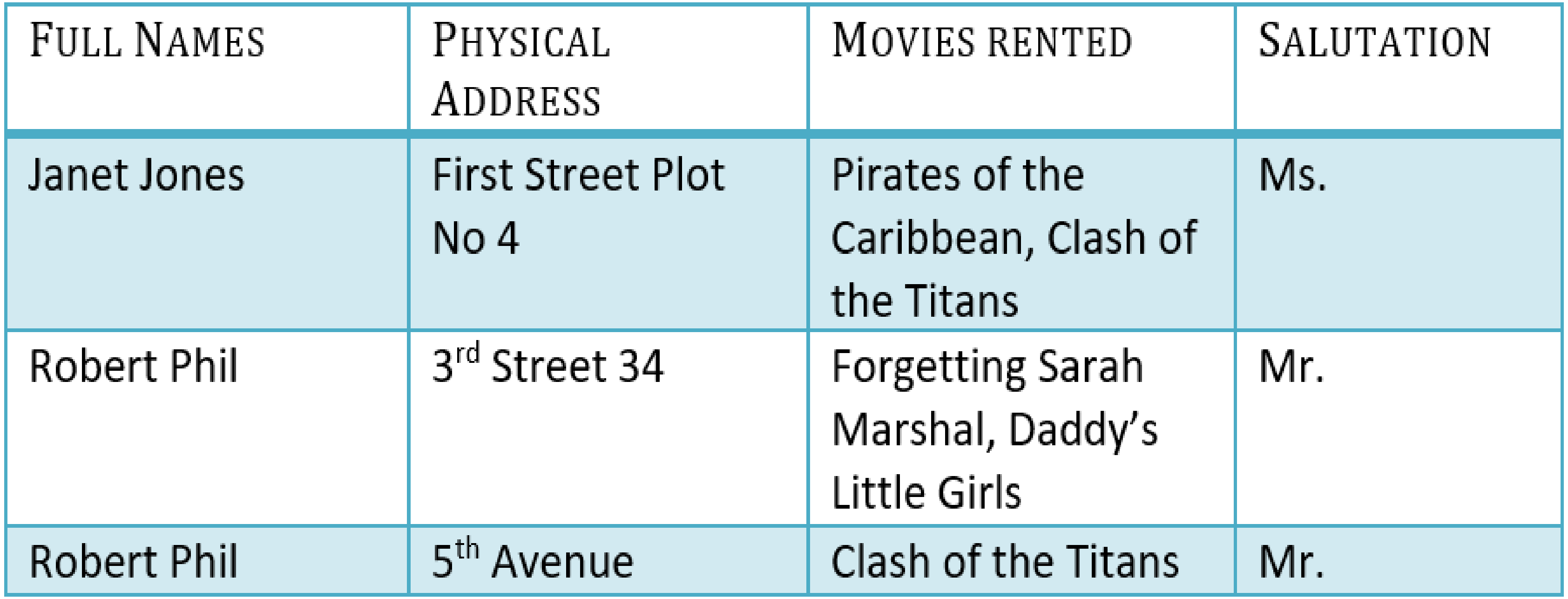
Student_Course Table

Student_ID	Course
21	Computer
21	Math
34	Chemistry
74	Biology
59	Physics

Student_Hobby Table

Student_ID	Hobby
21	Dancing
21	Singing
34	Dancing
74	Cricket
59	Hockey

Example



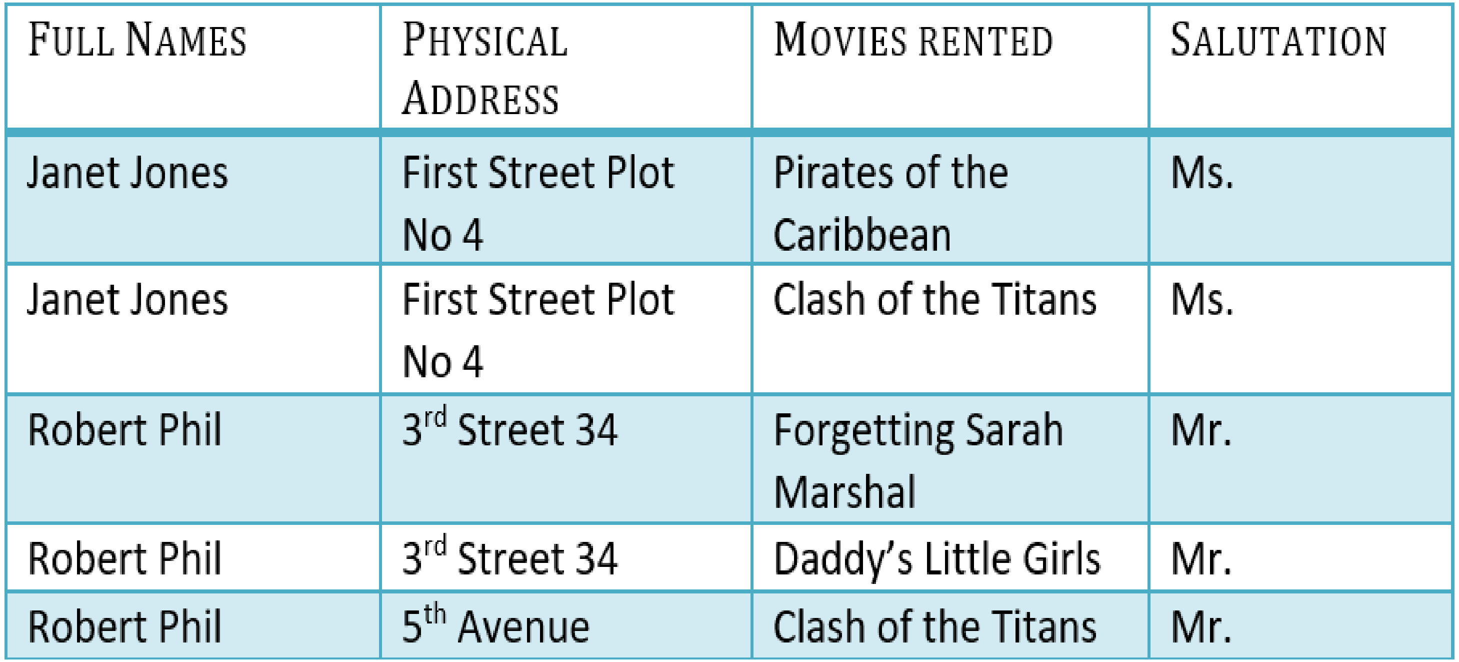
Full Names	Physical Address	Movies rented	Salutation
Janet Jones	First Street Plot No 4	Pirates of the Caribbean, Clash of the Titans	Ms.
Robert Phil	3rd Street 34	Forgetting Sarah Marshal, Daddy’s Little Girls	Mr.
Robert Phil	5th Avenue	Clash of the Titans	Mr.

Here, **Movies Rented column has multiple values.**

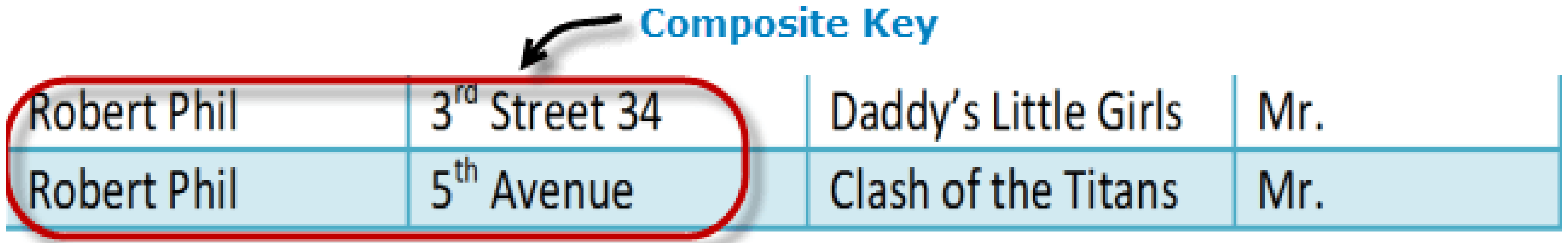
1NF (First Normal Form) Rules

- Each table cell should contain a single value.
- Each record needs to be unique.

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.

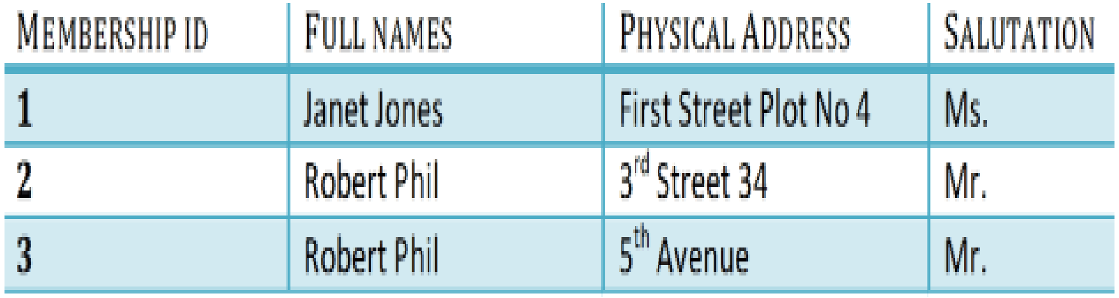

Composite Key

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

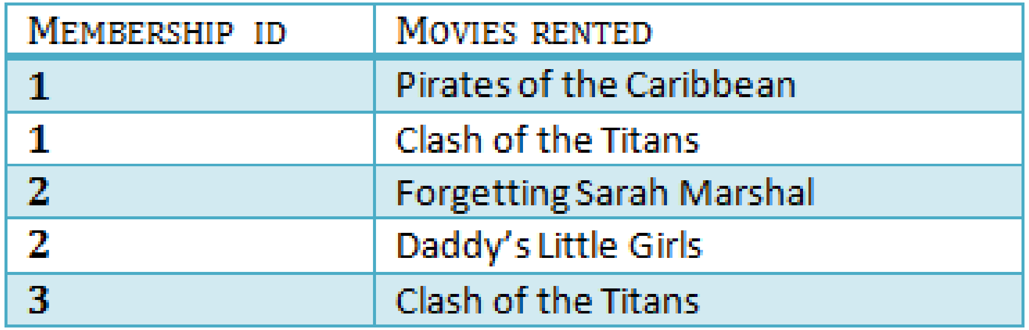
Names are common. Hence you need name as well Address to uniquely identify a record.

2NF (Second Normal Form) Rules

- Rule 1- Be in 1NF
- Rule 2- Single Column Primary Key that does not functionally dependent on any subset of candidate key relation.



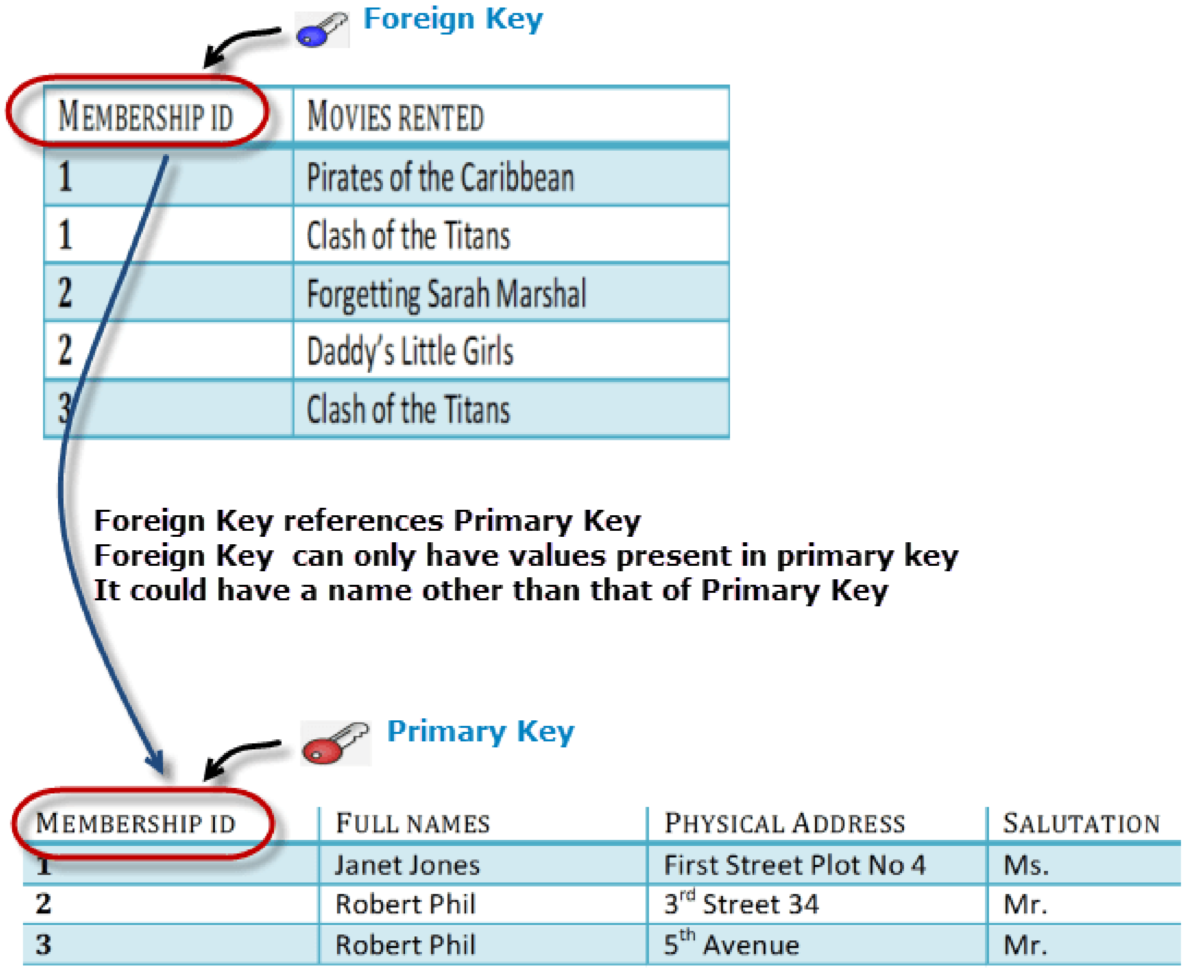
MEMBERSHIP ID	FULL NAMES	PHYSICAL ADDRESS	SALUTATION
1	Janet Jones	First Street Plot No 4	Ms.
2	Robert Phil	3rd Street 34	Mr.
3	Robert Phil	5th Avenue	Mr.



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

✓ Table 1 contains member information.

✓ Table 2 contains information on movies rented.



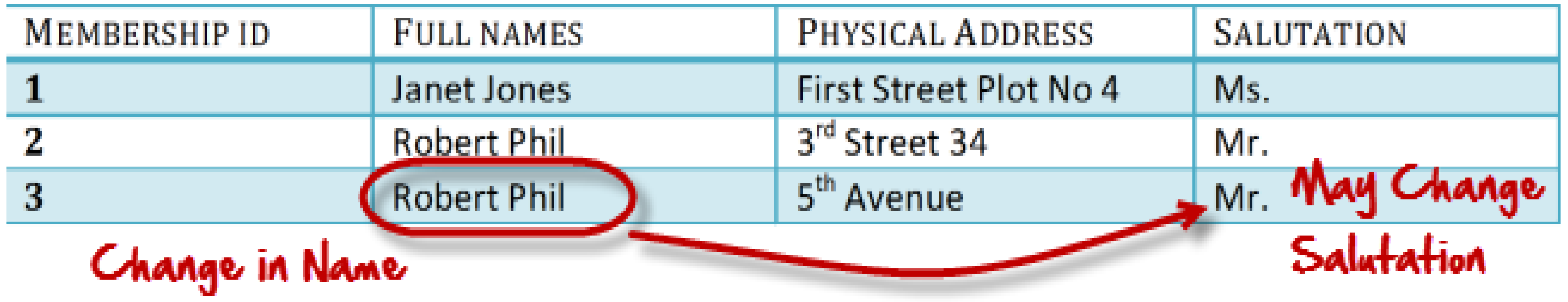
Membership ID	Movies rented
1	Pirates of the Caribbean
1	Clash of the Titans
2	Forgetting Sarah Marshal
2	Daddy's Little Girls
3	Clash of the Titans

Foreign Key references Primary Key
Foreign Key can only have values present in primary key
It could have a name other than that of Primary Key

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.

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

Consider the table 1. Changing the non-key column Full Name may change Salutation.



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

3NF (Third Normal Form) Rules

- Rule 1- Be in 2NF
- Rule 2- 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.

Decomposition

- Decomposition is the process of breaking down in parts or elements.
- It replaces a relation with a collection of smaller relations.
- It breaks the table into multiple tables in a database.
- It should always be lossless, because it confirms that the information in the original relation can be accurately reconstructed based on the decomposed relations.
- It can be of two types:
  - **Loss less decomposition**
  - **Lossy decomposition :** When a relation gets decomposed into multiple relational schemas, in such a way that retrieving the original relation leads to a loss of information it is called lossy decomposition

Properties of Decomposition:

Following are the properties of decomposition:

- **Lossless Decomposition :** The information should not get lost from the relation that is decomposed. It gives guarantee that the join will result in the same relation as it was decomposed.

- **Dependency preservation** : Functional dependencies should be maintained on the tables which are being decomposed.

- **Lack of data redundancy :** According to this property, decomposition must not suffer from data redundancy.

Example: Lossless Decomposition

EMPLOYEE_DEPARTMENT table

EMP_ID	EMP_NAME	EMP_AGE	EMP_CITY	DEPT_ID	DEPT_NAME
22	Denim	28	Mumbai	827	Sales
33	Alina	25	Delhi	438	Marketing
46	Stephan	30	Bangalore	869	Finance
52	Katherine	36	Mumbai	575	Production
60	Jack	40	Noida	678	Testing

The above relation is decomposed into two relations EMPLOYEE and DEPARTMENT

EMPLOYEE table

EMP_ID	EMP_NAME	EMP_AGE	EMP_CITY
22	Denim	28	Mumbai
33	Alina	25	Delhi
46	Stephan	30	Bangalore
52	Katherine	36	Mumbai
60	Jack	40	Noida

DEPARTMENT table

DEPT_ID	EMP_ID	DEPT_NAME
827	22	Sales
438	33	Marketing
869	46	Finance
575	52	Production
678	60	Testing

Now, when these two relations are joined on the common column "EMP_ID", then the resultant relation will look like: Employee ⋈ Department with no information loss. So, it is lossless decomposition.

EMP_ID	EMP_NAME	EMP_AGE	EMP_CITY	DEPT_ID	DEPT_NAME
22	Denim	28	Mumbai	827	Sales
33	Alina	25	Delhi	438	Marketing
46	Stephan	30	Bangalore	869	Finance
52	Katherine	36	Mumbai	575	Production
60	Jack	40	Noida	678	Testing

End!!