Normalization Case Study

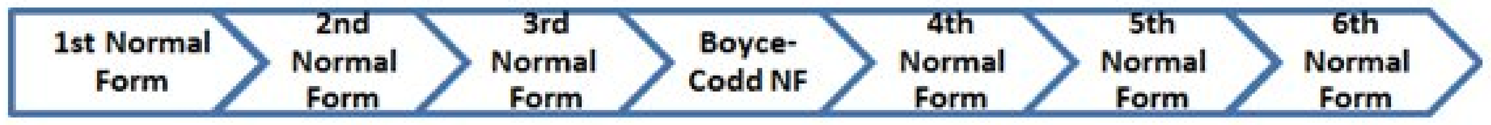
What is Normalization?

- **Normalization** is a database design technique that reduces data redundancy
- Normalization rules divides larger tables into smaller tables and links them using relationships
- Normalization in SQL is to eliminate redundant (repetitive) data and ensure data is stored logically
- The inventor of the relational model **Edgar Codd** proposed the theory of normalization of data with the introduction of the First Normal Form, and he continued to extend theory with Second and Third Normal Form. Later he joined **Raymond F. Boyce** to develop the theory of Boyce-Codd Normal Form.

Database Normal Forms

Here is a list of Normal Forms in SQL:

- 1NF (First Normal Form)
- 2NF (Second Normal Form)
- 3NF (Third Normal Form)
- BCNF (Boyce-Codd Normal Form)
- 4NF (Fourth Normal Form)
- 5NF (Fifth Normal Form)
- 6NF (Sixth Normal Form)

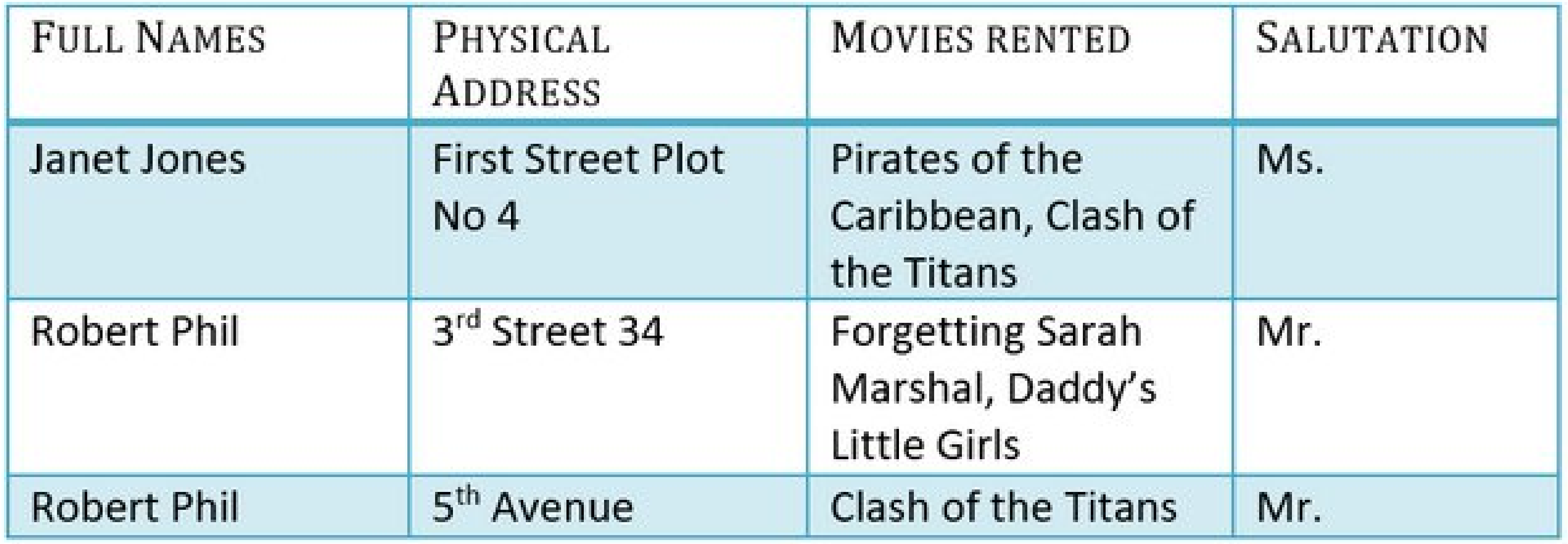


Database Normal Forms

The Theory of Data Normalization in MySQL server 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 process of Normalization in SQL theories is described below-

Database Normalization With Examples

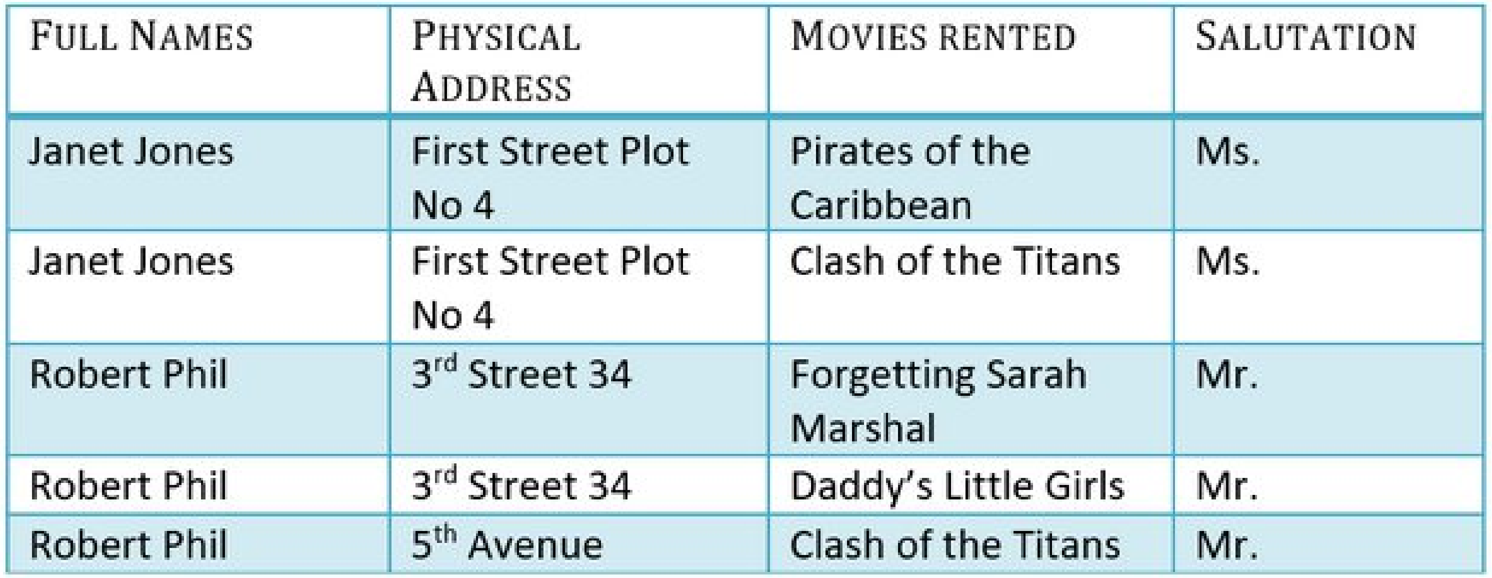
- Database **Normalization Example** can be easily understood with the help of a case study. Assume, a video library maintains a database of movies rented out. Without any normalization in database, all information is stored in one table as shown below. Let’s understand Normalization database with normalization example with solution:



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.

1NF (First Normal Form) Rules

- Each table cell should contain a single value.
- Each record needs to be unique.
- The above table in 1NF-

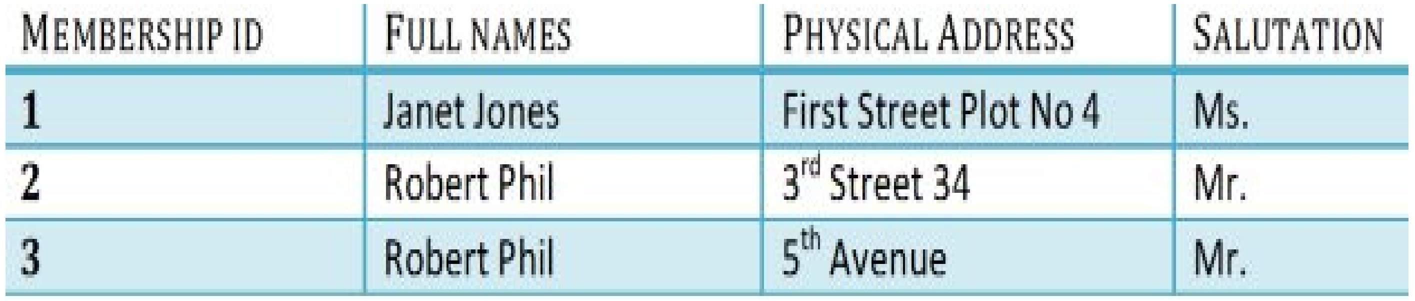


Full Names	Physical Address	Movies rented	Salutation
Janet Jones	First Street Plot No 4	Pirates of the Caribbean	Ms.
Janet Jones	First Street Plot No 4	Clash of the Titans	Ms.
Robert Phil	3rd Street 34	Forgetting Sarah Marshal	Mr.
Robert Phil	3rd Street 34	Daddy’s Little Girls	Mr.
Robert Phil	5th Avenue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

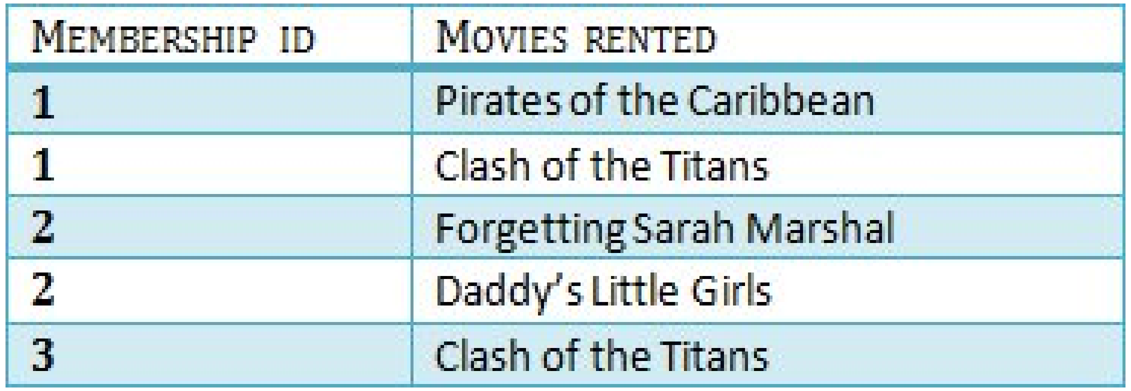
Example of 1NF in DBMS

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
- It is clear that we can’t move forward to make our simple database in 2nd Normalization form unless we partition the table above.



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

2NF (Second Normal Form) Rules

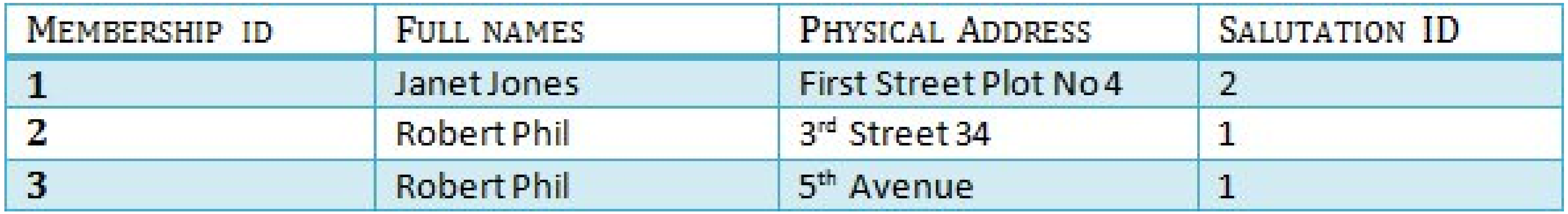
- We have divided our 1NF table into two tables into Table 1 and Table2. Table 1 contains member information. Table 2 contains information on movies rented.

- We have introduced a new column called Membership_id which is the primary key for table 1. Records can be uniquely identified in Table 1 using membership id.

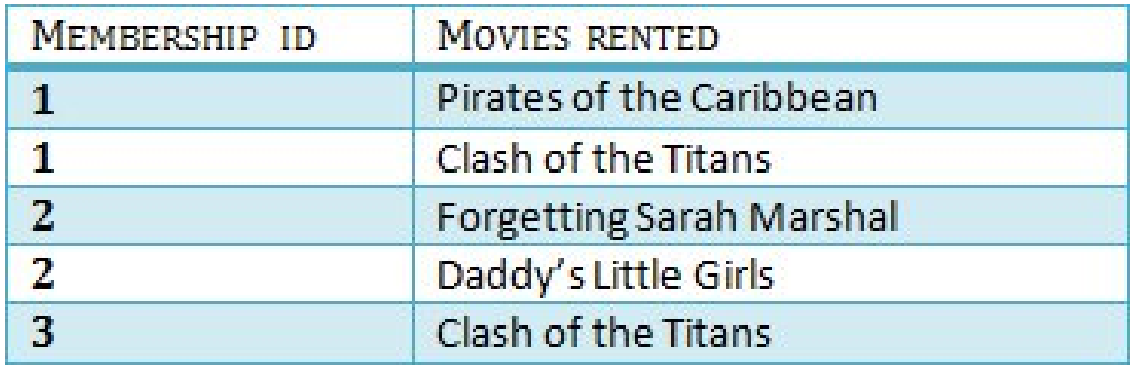
3NF (Third Normal Form) Rules

- Rule 1- Be in 2NF
- Rule 2- Has no transitive functional dependencies

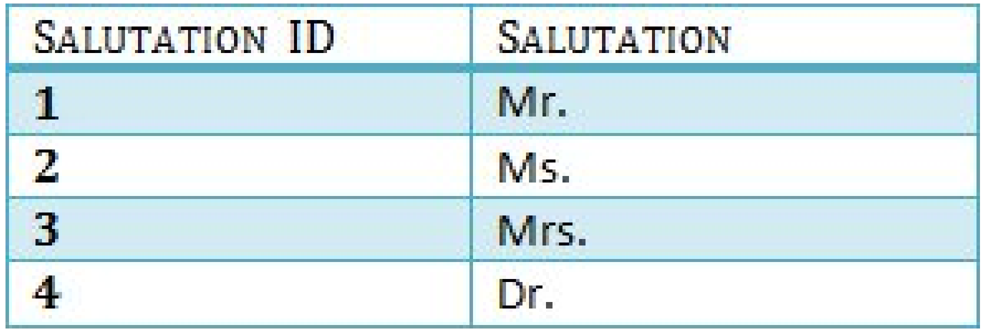
To move our 2NF table into 3NF, we again need to again divide our table.



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.

3NF (Third Normal Form) Rules

- We have again divided our tables and created a new table which stores Salutations.
- There are no transitive functional dependencies, and hence our table is in 3NF
- In Table 3 Salutation ID is primary key, and in Table 1 Salutation ID is foreign to primary key in Table 3

Now our little example is at a level that cannot further be decomposed to attain higher normal form types of normalization in DBMS. In fact, it is already in higher normalization forms. Separate efforts for moving into next levels of normalizing data are normally needed in complex databases. However, we will be discussing next levels of Normalization in DBMS in brief in the following.

BCNF (Boyce-Codd Normal Form)

- Even when a database is in 3rd Normal Form, still there would be anomalies resulted if it has more than one **Candidate** Key.
- Sometimes is BCNF is also referred as **3.5 Normal Form.**

4NF (Fourth Normal Form) Rules

- If no database table instance contains two or more, independent and multivalued data describing the relevant entity, then it is in 4th Normal Form.

5NF (Fifth Normal Form) Rules

- A table is in 5th Normal Form only if it is in 4NF and it cannot be decomposed into any number of smaller tables without loss of data.

6NF (Sixth Normal Form) Proposed

- 6th Normal Form is not standardized, yet however, it is being discussed by database experts for some time. Hopefully, we would have a clear & standardized definition for 6th Normal Form in the near future…

That's all to SQL Normalization!

Terms

- A **KEY in SQL** is a value used to identify records in a table uniquely. An SQL KEY is a single column or combination of multiple columns used to uniquely identify rows or tuples in the table.

- **A primary key** is a single column value used to identify a database record uniquely.
- A primary key cannot be NULL, must be unique, rarely be changed, be changed
- **A composite key** is a primary key composed of multiple columns used to identify a record uniquely.
- **A Foreign Key** references the primary key of another Table.
- A foreign key can have a different name from its primary key
- Unlike the Primary key, they do not have to be unique. Most often they aren't
- Foreign keys can be null even though primary keys can not

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