CAB
Relational Data Model

Prepared by:
Kamaldeep Kaur Sarna
Assistant Professor
Shri Ram College of Commerce

Learning Objectives

- The 12 Rules of E.F. Codd for an RDBMS
- Concept of Keys
  - Super Key
  - Candidate Key
  - Primary Key
  - Secondary Key
  - Composite Key
  - Foreign Key

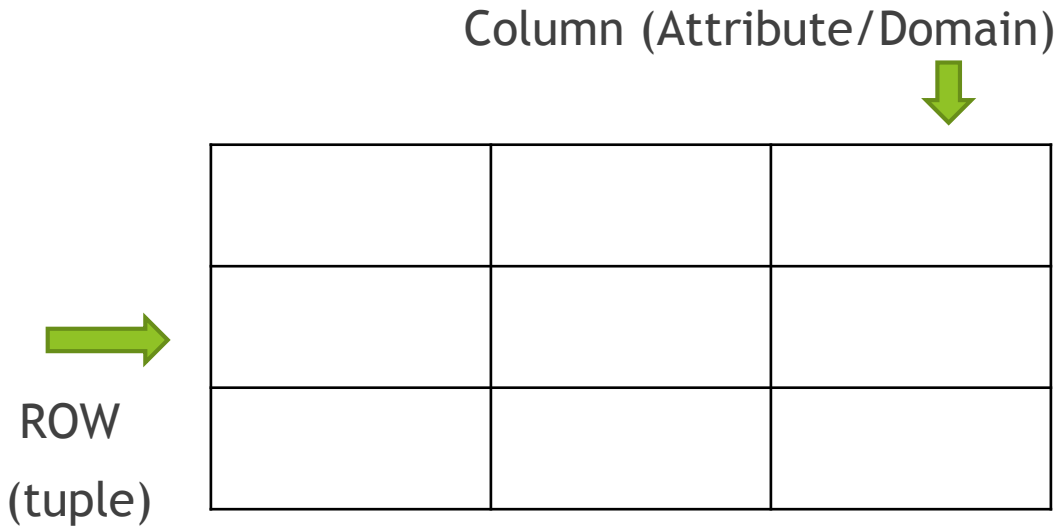
Learning Objectives (Contd.)

- ER diagram to Relational Mapping
  - Entity Conversion
  - Conversion of Relationships
  - Conversion of Multivalued Attributes
- Normalization of Relational Database
  - First Normal Form (1NF)
  - Second Normal Form (2NF)
  - Third Normal Form (3NF)

The 12 Rules of E.F. Codd for an RDBMS

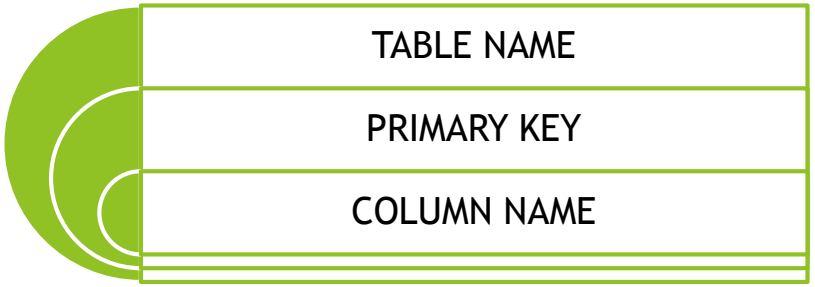
RULE 1: Information Representation Rule

- As per this rule all data and metadata (data about data) stored in the relational database are represented by only one way i.e. values in table (combination of rows and columns) as shown below.



RULE 2: Guaranteed Access Rule

This rule requires that every value of data must be logically addressable by using a combination of Table name, primary key, column name.



RULE 3: Systematic treatment of null values

- A null values means nothing.
- RDBMS must have the capability and consistent method of dealing with null values , independent of data type.
- It must be capable of distinguishing a null value from 0 (for numeric data) and spaces.

Rule 4 : Database Description Rule:

- The description of a database is stored and maintained in the form of tables.
- The structure description of the entire database must be stored in an online catalog, known as **data dictionary**.
- It can be accessed by authorized users.
- Users can use the same query language to access the catalog which they use to access the database itself.

RULE 5: Comprehensive data sub-language rule

- An RDBMS must support a clearly defined language that should support data definitions, data manipulation, views, data integrity constraints and database transaction control.

RULE 6: View Updating

- Views means the logical combinations in which data can be presented to the users.
- Each view should support same range of data manipulation by RDBMS which is available to users accessing direct table.

RULE 7: High level of update, insert and delete

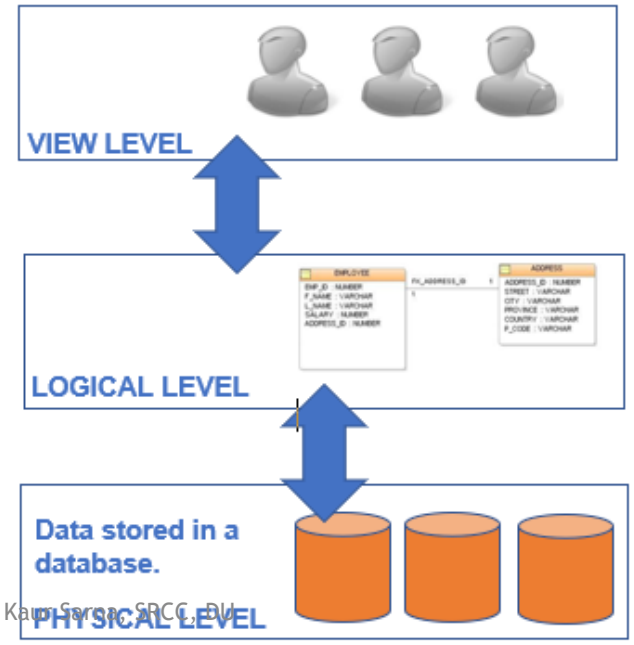
- An RDBMS must support the users to insert, update and delete a data in the database
- These options must be available to all retrievals.

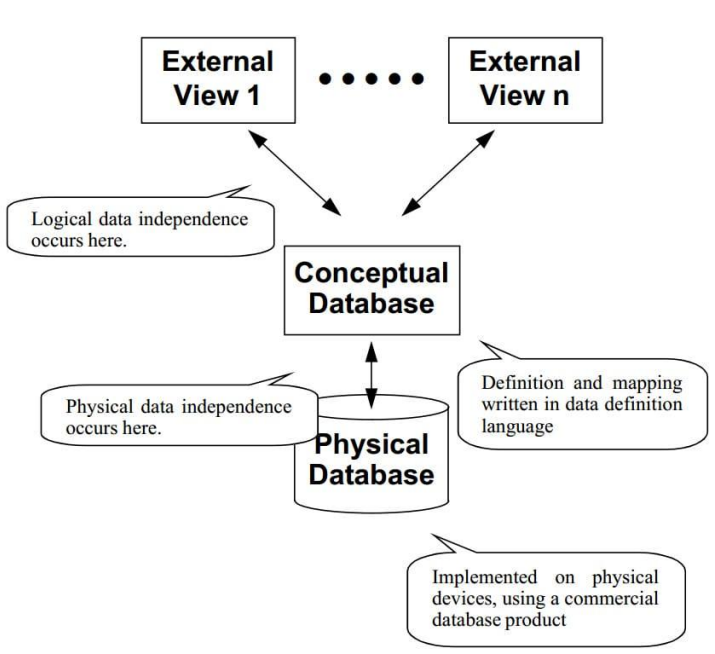
RULE 8: Physical data independence

- This rule require that the application program should be independent of physical changes made to database in the form of storage arrangements.

Rule 9: Logical Data Independence

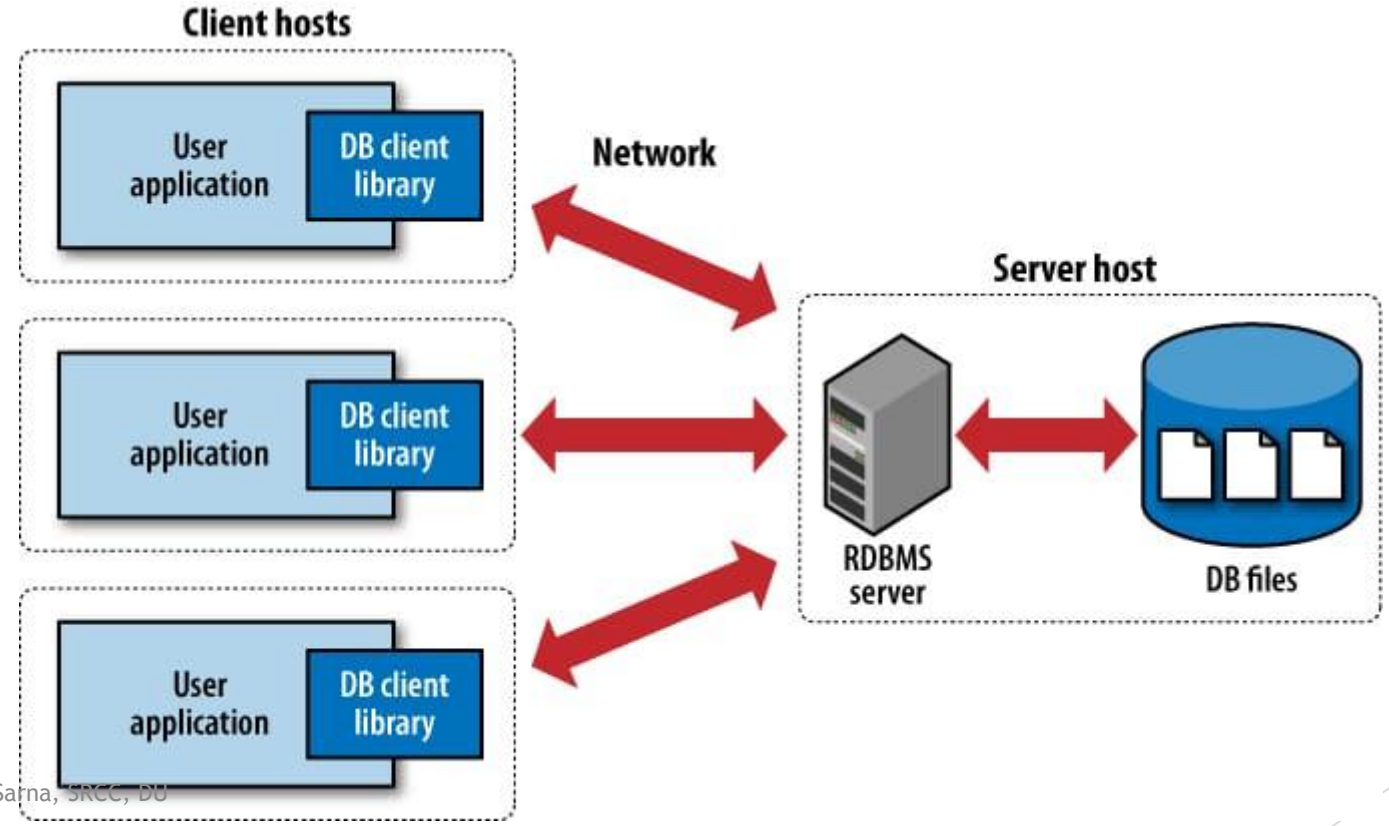
- It requires that any changes made to the table shouldn't require any changes made to the application program. Example; if a new column is inserted it should not affect the use of columns which were already present.





RULE 10: The Distribution Independence Rule

- The RDBMS package should be able to spread on more than one computer system and across several networks even though they have different types of hardware and OS.



RULE 11: Non-Subversion Rule

- If RDBMS has a low level language, that low level language cannot be used to subvert or bypass the integrity rules written in higher level relational language.

Rule 12: Integrity Independence Rule

- Integrity constraints which are specific to a particular RDBMS must be able to be defined in the relational data sub-language and should be storable in the catalogue and not in application program.

Concept of Keys

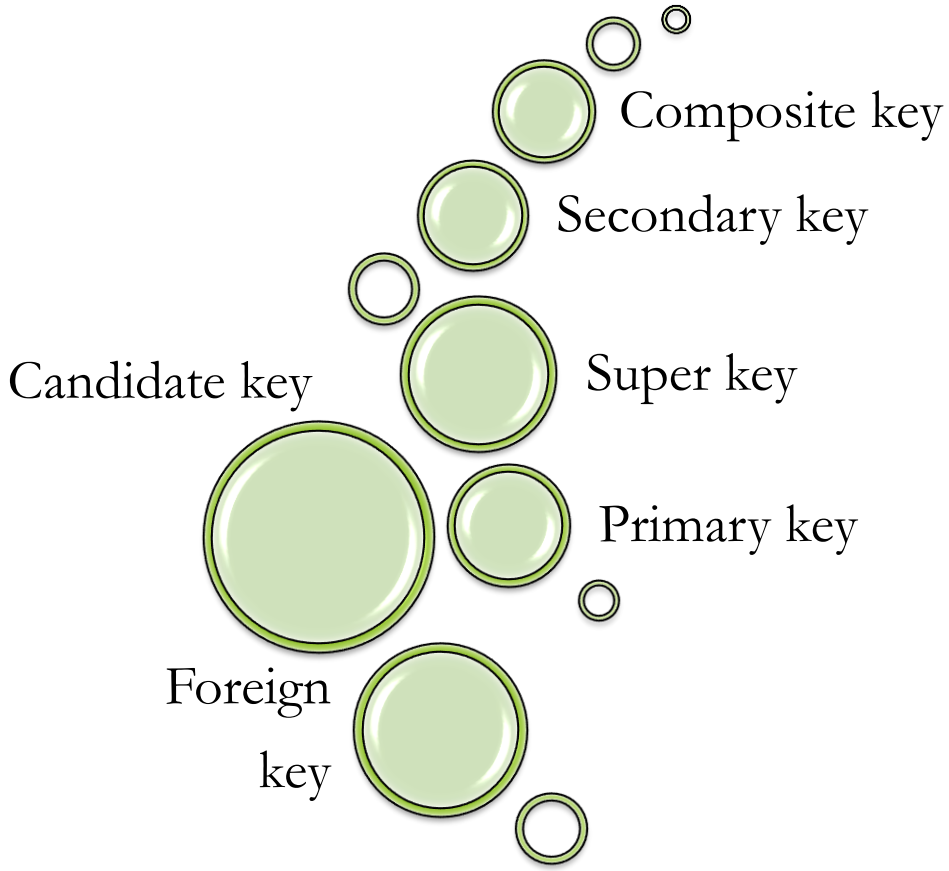
Database Keys

- Keys are very important part of Relational database model.
- They are used to establish and identify relationships between tables and also to uniquely identify any record or row of data inside a table.
- A Key can be a single attribute or a group of attributes, where the combination may act as a key.

Need for Keys

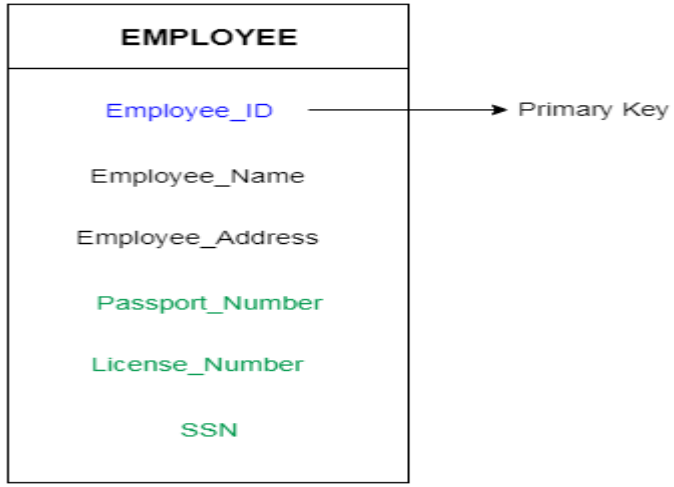
- In real world applications, number of tables required for storing the data is huge, and the different tables are related to each other as well.
- Also, tables store a lot of data in them. Tables generally extend to thousands of records stored in them, generally, unsorted and unorganised.

Types of Keys



Primary Key

- A primary key is an attribute or a group of attributes which uniquely identifies a tuple.
- A primary key can never be same for two or more records.
- For eg; In the EMPLOYEE table, ID is primary key since it is unique for each employee as shown below.
- In the EMPLOYEE table, we can also select License_Number and Passport_Number as primary key since they are also unique.



Candidate Key

- A candidate Key is an attribute or set of attributes which can uniquely identify a tuple.
- It is a minimal super key for that relation.
- Every table will have at least one candidate key.
- But, there can be many candidate keys in one table.
- All of these have potential to be selected as primary key.

Super Key

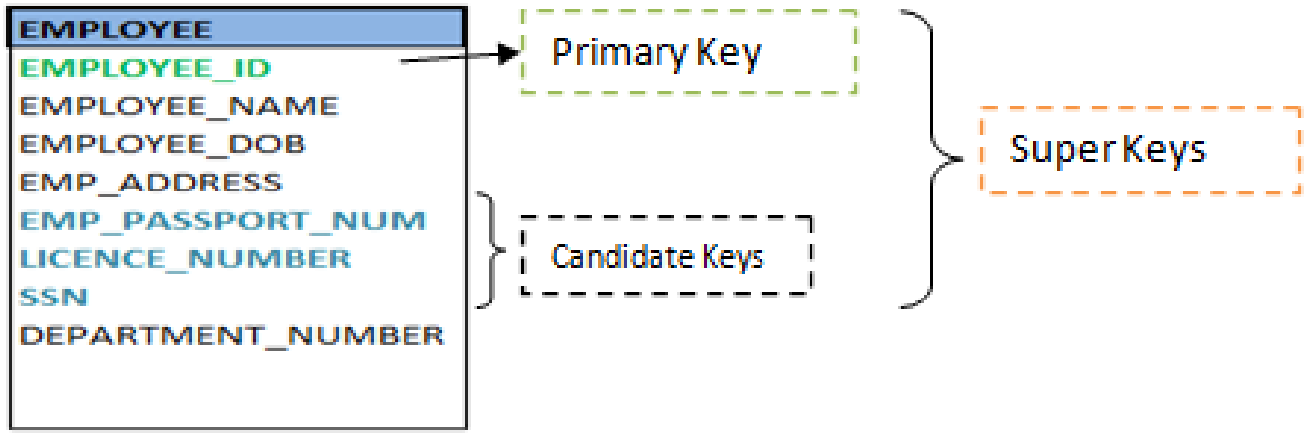
- Super key is a set of one or more attributes of a relation (table) whose combined value uniquely identifies a database record.
- Assuming there is a table named Employee having attributes such as Emp_Name, Emp_ID, Emp_Address, Emp_Phone.
- To identify a record there are multiple alternatives such as Emp_ID; (Emp_Name, Emp_Address), (Emp_Name, Emp_Address, Emp_Phone), etc.
- All the alternatives are super keys.

Foreign Key

- A foreign key is that attribute of a table (relation) whose values correspond to the values of a primary key of another table.
- A foreign key should use the same properties as that of the associated primary key.

Composite Key

- A composite key is a combination of attributes which uniquely identifies a record in a table.
- We use a composite key in case when one attribute solely cannot uniquely identify a record.



Secondary Key

- A Secondary key is an attribute or a combination of attributes which classifies the entity set on a particular characteristics.
- For example: In the Students table the students can be classified on the basis of the department in which they study.
- Secondary key may not bring a uniqueness in the record and hence it is not a candidate key.

Example:

STUDENT ID	NAME	PHONE	Course	E-mail id
1	Jai	9876723452	B.Com	jai@gmail.com
2	Alan	9991165674	B.Com	alan3@gmail.com
3	Sam	7898756543	B.A.(Hons.)Economics	sam@gmail.com
4	Anil	8987867898	B.Com	ani@gmail.com
5	Bob	9990080080	B.A.(Hons.)Economics	bob@gmail.com

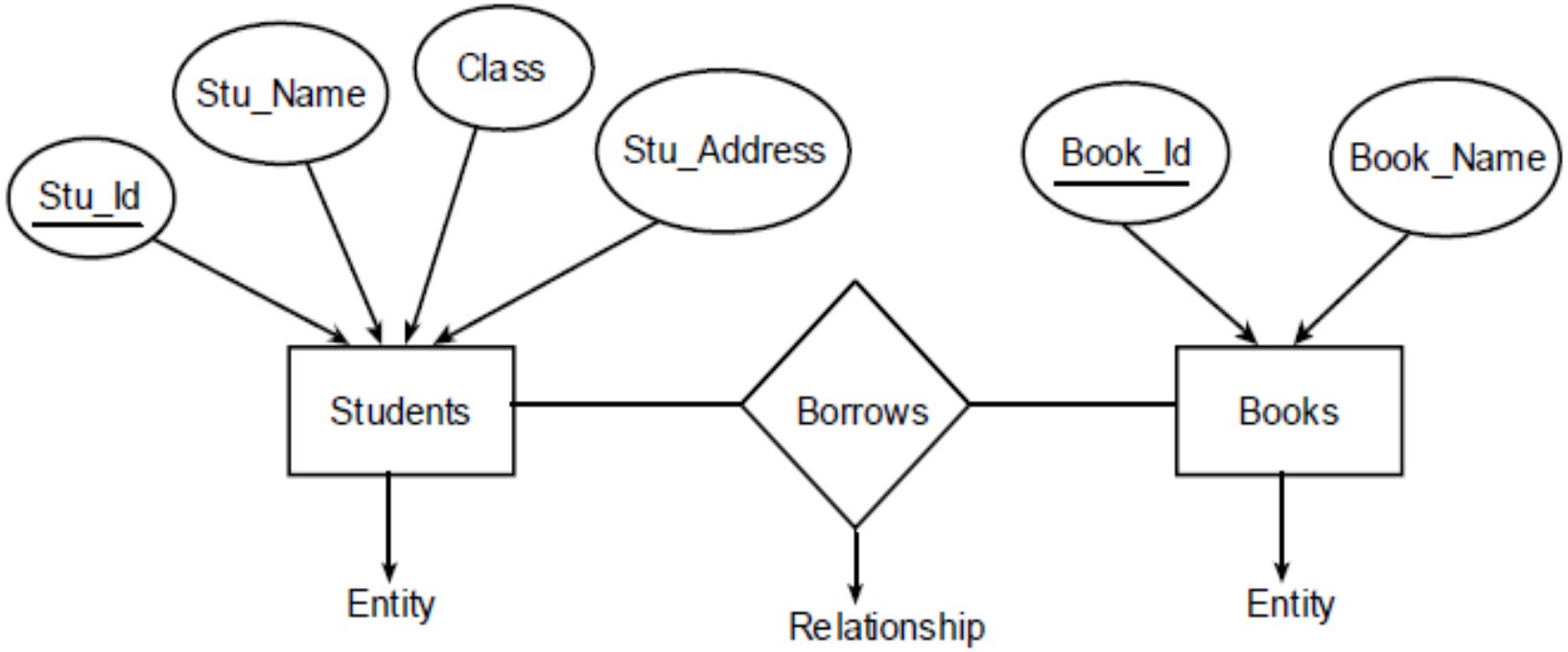
ER-Diagram to Relational Mapping

ER-Diagram to Relational Mapping

- An ER-diagram is prepared by the designer to capture a real world situation and to give a pictorial representations of the interaction between the entities.
- Later the ER-diagram need to be mapped to the relational model.
- To explain the process, the example taken is that a library issues books to the students.
- For simplicity, two entities student and books are taken.

ER-Diagram to Relational Mapping

- Diagrammatically it can be presented as :



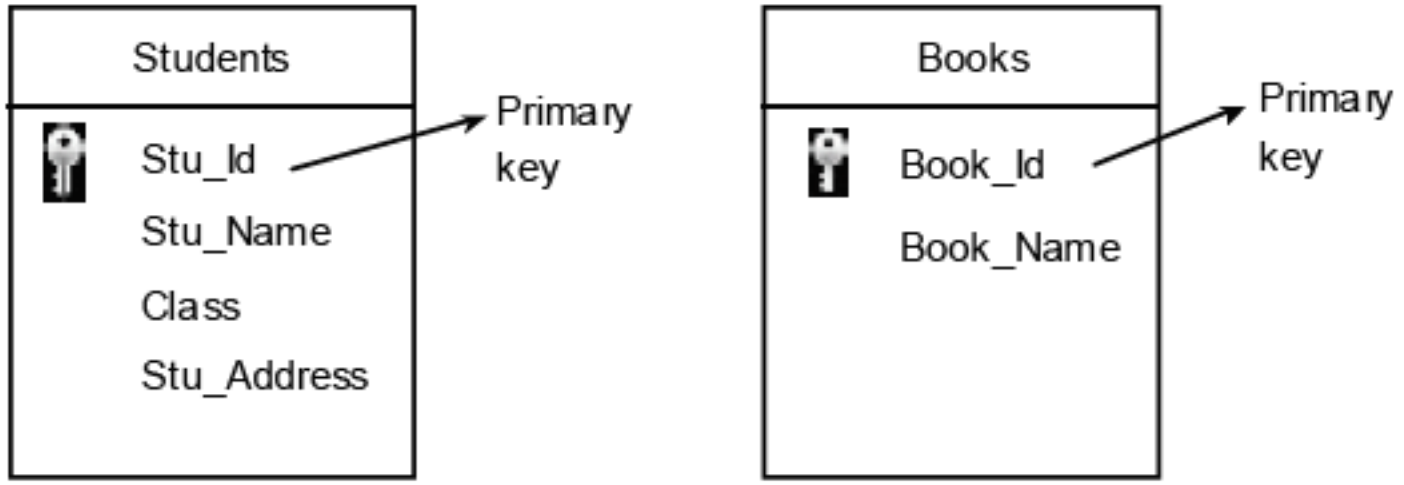
ER-Diagram to Relational Mapping

- The designer proceeds with the mapping, preserving as many integrity constraints as possible. There are three steps to be taken in the process :

1. Conversion of entities,
2. Conversion of relationships, and
3. Conversion of multivalued attributes.

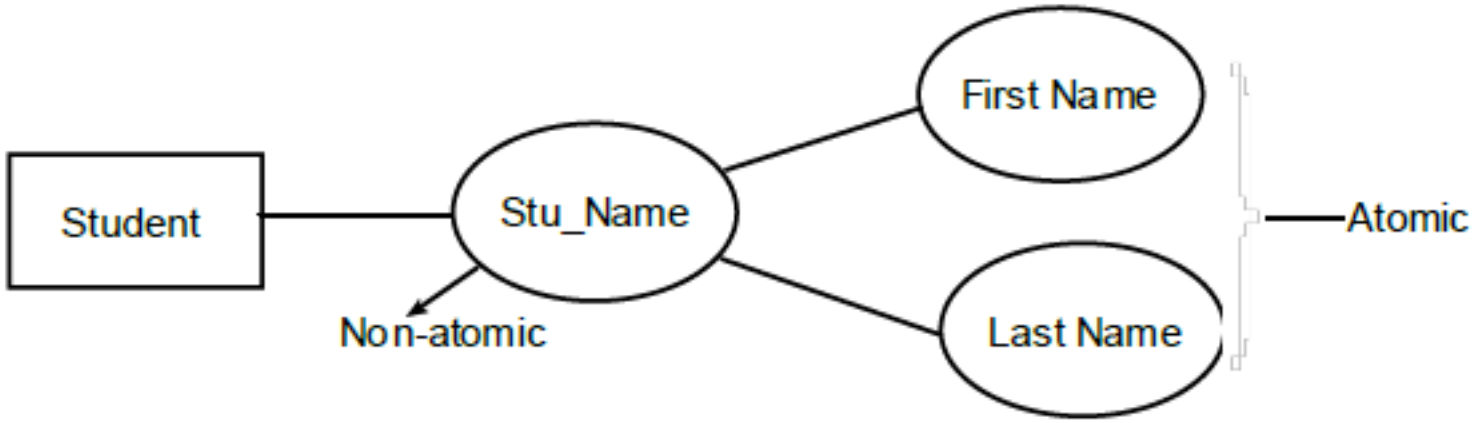
Entity Conversion

- The entity sets are translated into relations (tables).
- The attributes of entities become the attributes of the relations. Any key attribute is assigned a primary key.
- In the previous example, there will be two tables (Relations).



Entity Conversion

- During this conversion all non-atomic attribute are converted into atomic attributes.
- For example, if Student Name contains first name and last name then Stu_Name should be replaced with atomic attributes.

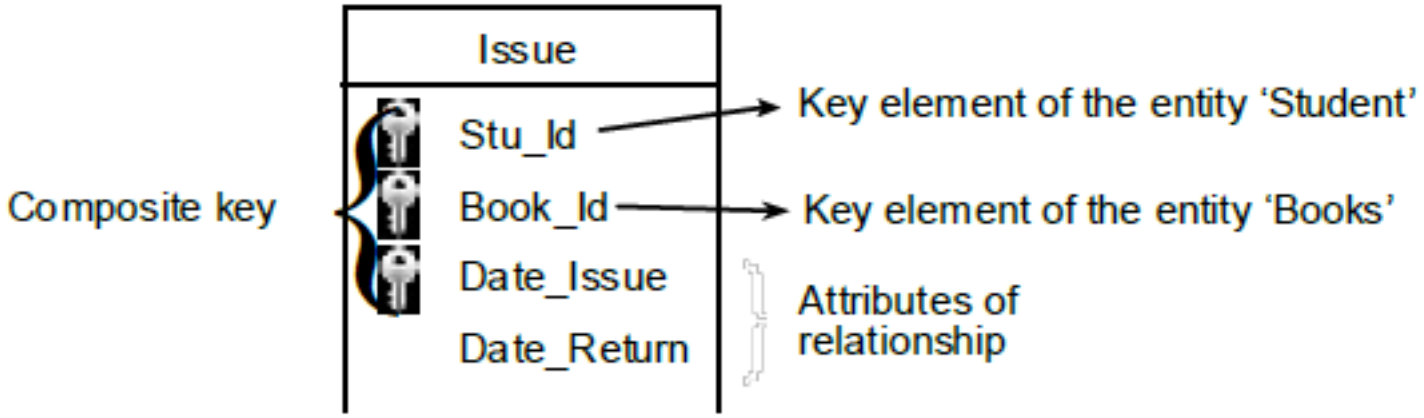


Conversion of Relationships

- The relationship sets between the entities are translated into tables (Relations).
- This table assumes the attributes of the relationship set and the key attributes of participating entities.
- To continue with the example of the library, the Relationship between the entities is borrowing of books and the key attributes of participating entities are book_id, and stu_id.

Conversion of Relationships

- An ‘Issue’ table will be created as given below:



Conversion of Multivalued Attributes

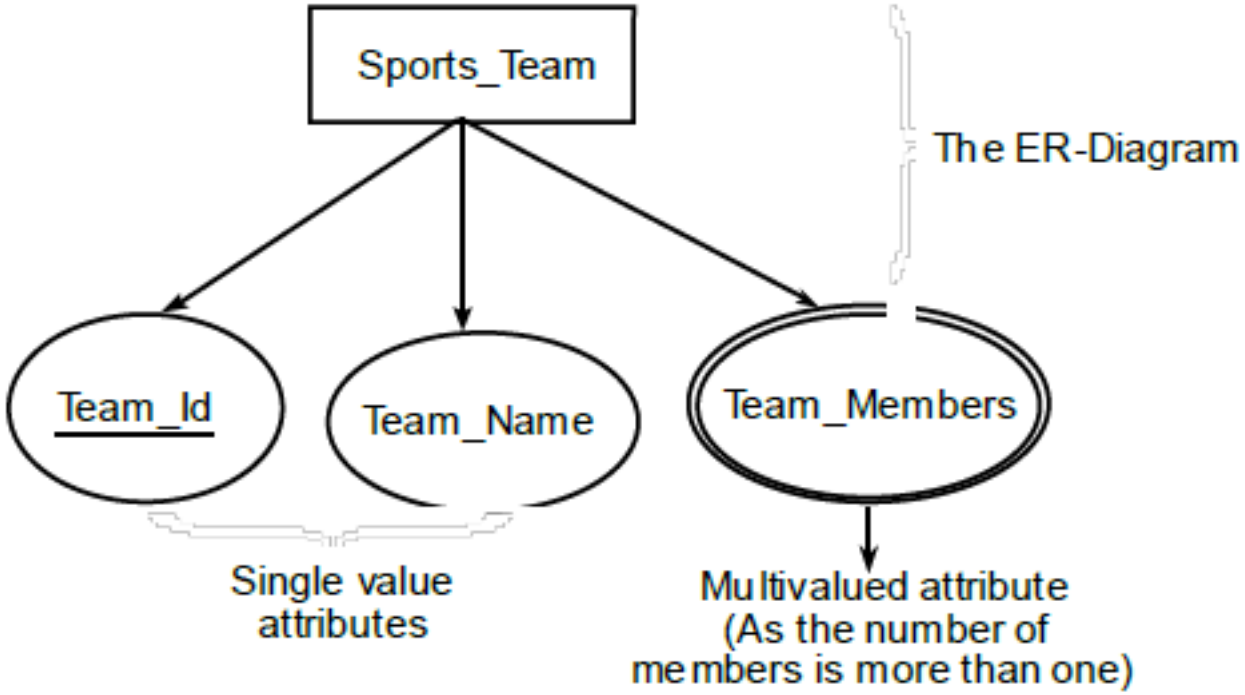
- In conventional database design, each row and column must store only one value.
- For example, in a relation if you add a column (attribute) qualification of employees, this attribute may assume more than one value like matriculation, graduation, masters, etc.
- Since only one row belongs to one employee, all these values are added in one cell only that is known as multivalued attributes.
- But in conventional database a multivalued attribute need to be converted into a single value attribute.

Conversion of Multivalued Attributes

- To convert multivalued attributes into single value attributes, a new relation (table) is created which contains the attribute and the Primary Key of the relation.
- For example, the sports team of a college has more than one players (data value).

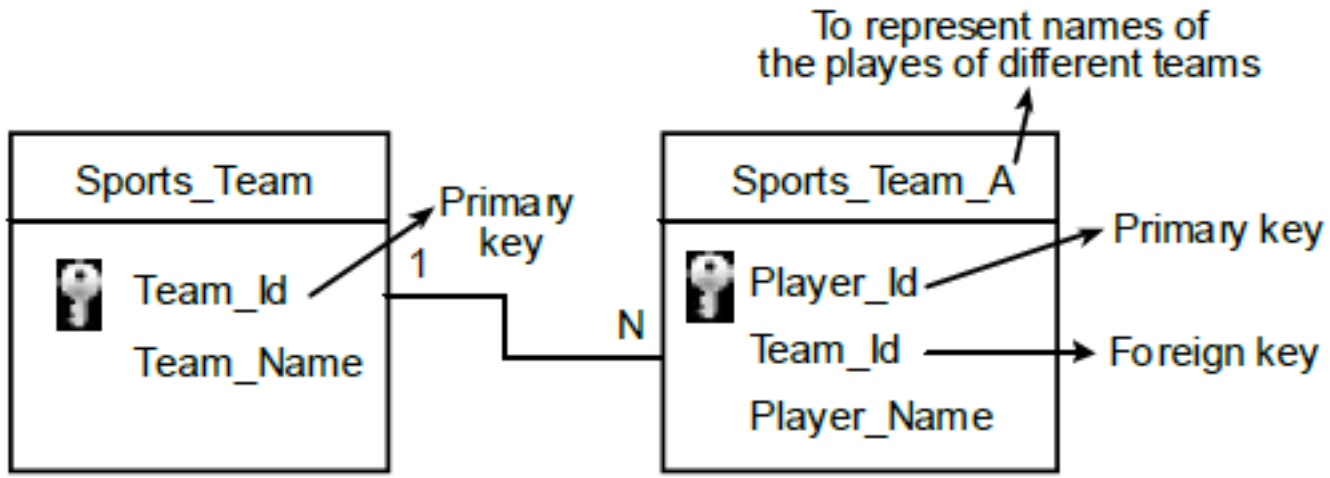
Conversion of Multivalued Attributes

- In this case, the table will have the following attributes.



Conversion of Multivalued Attributes

- To map this in a relational database, two relations (tables) will be created in the following manner:



Normalization of relational database

- Normalisation is the process of efficiently organising data.
- It is the process adopted to structure the data in a manner which removes or controls data redundancy.
- It contains a set of rules which are concerned with:
  - Identifying relationships among the attributes;
  - Converting those relationships in the form of relations(tables), and
  - Combining these relations to form a database.

Normalization of relational database

There are two objectives of data normalization:

1. Controlling redundant data, and

2. Making sure that the data dependencies make sense, i.e., only related data are stored into tables.

Normalization of relational database

- Normalization generally involves splitting of existing tables into multiple ones that are re-joined each time a query is made.

- It was proposed by **Edgar F. Codd** in his paper "**A Realtional Model of Data for Large Shared Data Banks**."

Advantages of Normalization:

1. Maintains data integrity,
2. Optimizes query response and
3. Improves process of updating data.

First Normal Form (1NF)

- The first normal form sets the very basic rule for organizing database.
- It is the first step of data normalization and is a prerequisite for the second and third normal form.
- A table is said to be in first normal form only if the following are satisfied:
  - Key attributes are defined
  - Attribute values are atomic
  - The table does not contain any repeating group.

Second Normal Form (2NF)

- A relation (table) is said to be in 2nd normal form
  - If it is in first normal form (1NF) and
  - If it contains a multi-attribute key (composite key) then all the non key attributes must be dependent on the complete set of the key attribute (composite key) and not on any part of it.

Third Normal Form (3NF)

- A table is said to be in Third Normal Form (3NF) if
  - It is in first and second normal form, and
  - Every non-key attribute is directly dependent on the key attribute.
- In a table which is in third normal form, all the non-key attributes are independent of each other and directly depend upon the key attribute.