Database Normalization

Unnormalized Table

⇩

1 NF

Atomic Domains

⇩

2 NF

Non-Key attributes is fully dependent on primary key.

⇩

3 NF

Removes Transitive Functional Dependency

⇩

4 NF

Removes multi-valued dependency

⇩

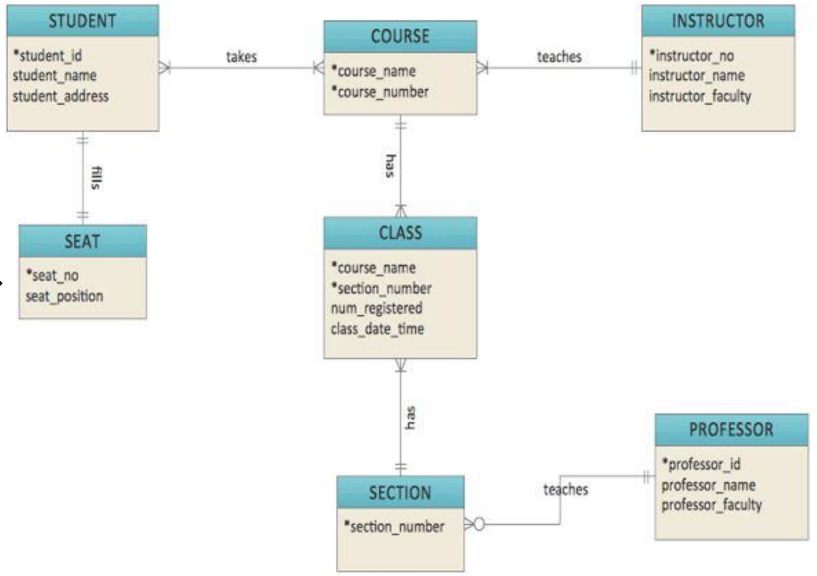
5 NF

Removes join dependency

⇩

Normalized Table

TO



ER DIAGRAM

Objectives

At the end of this module, you will be able to

- Data Normalization Fundamental
- Type of Data Normalization
- Example of Data Normalization
- Data Normalization Process
- De-Normalization Fundamental
- Difference between Normalization Vs De-Normalization
- What is ER Diagram ?
- What is Enity, Attributes, Joins ?
- Reverse & Forward Enginerring ?
- ER Model Management
- Relational Data Model concepts
- Operations in Relational Model

Data Normalization

Data Normalization

Normalization can be defined as :-

- Normalization is a database design technique of organizing the data in database to avoid data redundancy, insertion anomaly, update anomaly & deletion anomaly.
- A process of organizing data into tables in such a way that the results of using the database are always unambiguous and as intended. Such normalization is intrinsic to relational database theory.
- It may have the effect of duplicating data within the database and often results in the creation of additional tables.
- It divides larger tables to smaller tables and links them using relationships.
- It is a technique that helps in designing the schema of the database in an optimal manner so as to ensure the above points.

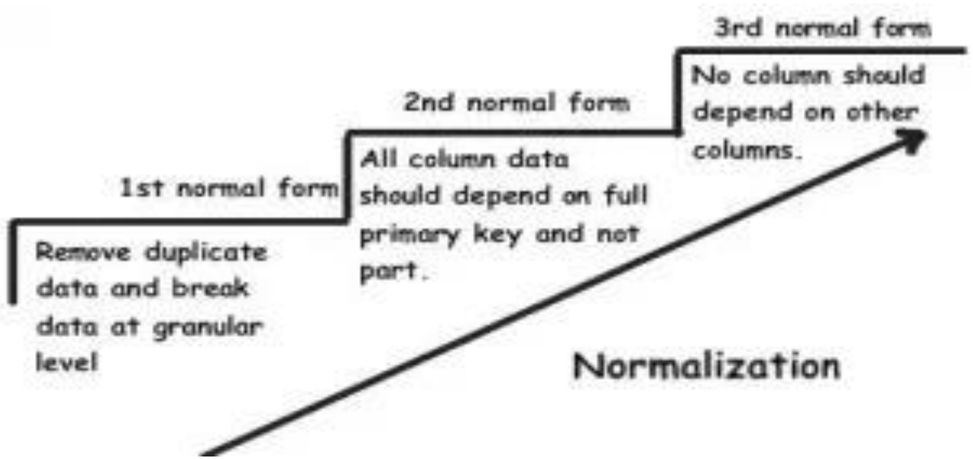
Data Normalization (Contd.)

- The core idea of database normalization is to divide the tables into smaller subtables and store pointers to data rather than replicating it.
- Database Normalization is used for following Purpose:
- To Eliminate the redundant or useless data
- To Reduce the complexity of the data
- To Ensure the relationship between tables as well as data in the tables
- To Ensure data dependencies and data is logically stored.

Data Normalization Types

Normalization Type can be defined as :-

- First normal form(1NF)
- Second normal form(2NF)
- Third normal form(3NF)

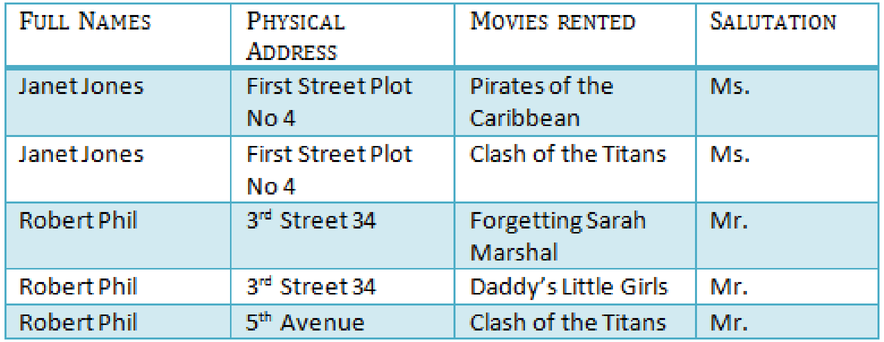

3rd normal form
No column should depend on other columns.
2nd normal form
All column data should depend on full primary key and not part.
1st normal form
Remove duplicate data and break data at granular level
Normalization

Data Normalization Types – First Normal Form(1NF)

1st NF Normalization can be defined as :-

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

1st NF Example



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.

Before we proceed let's understand a few concepts for better understanding :-

What is a **Primary Key?**
What is **Composite Key?**

Data Normalization Types – First Normal Form(1NF)

What is a Primary Key?

A Primary Key is a single column value used to identify a database record uniquely.

It has following attributes :-

- A primary key cannot be NULL
- A primary key value must be unique
- The primary key values should rarely be changed
- The primary key must be given a value when a new record is inserted

LearnOA To The Next Level...

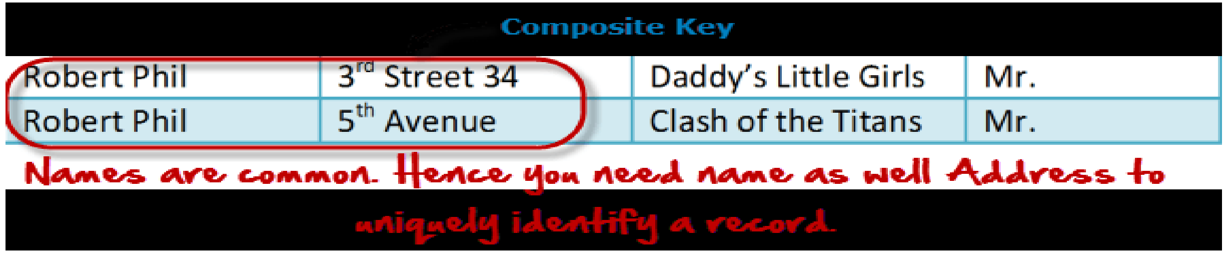
Data Normalization Types – First Normal Form(1NF)

What is a Composite Key?

A composite key is a primary key composed of multiple columns used to identify a record uniquely.

- 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.

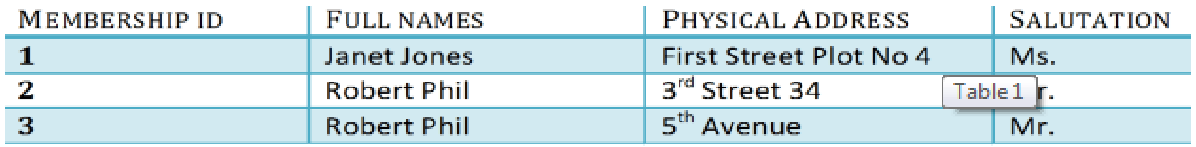
Names are common. Hence you need name as well Address to uniquely identify a record.

Data Normalization Types – Second Normal Form(2NF)

2nd NF Normalization can be defined as :-

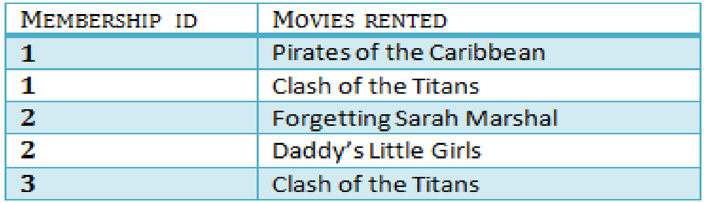
- Rule 1- Be in 1NF
- Rule 2- Single Column Primary Key
- It is clear that we can't move forward to make our simple database in 2nd Normalization form unless we partition the table above.

2nd NF Example



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

Table 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

Table 2

Data Normalization Types – Second Normal Form(2NF)

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

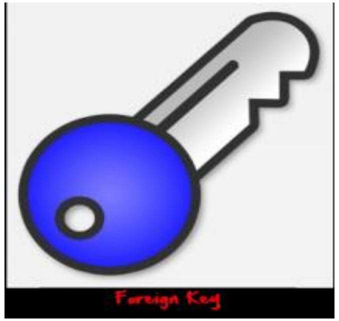
- We have introduced a new column called Membership_id which is the primary key for table 1. Records can be uniquely identified in Table 1 using membership id

Data Normalization Types – Second Normal Form(2NF)

What is a Foreign Key?

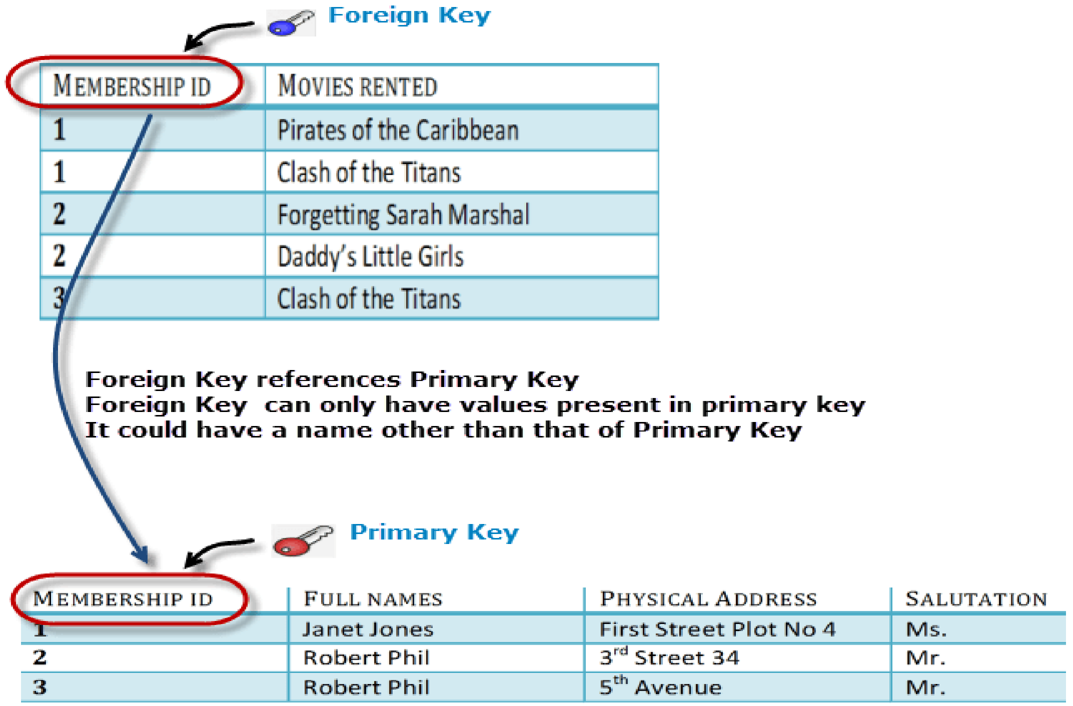
Foreign Key references the primary key of another Table! It helps connect your Tables

- A foreign key can have a different name from its primary key
- It ensures rows in one table have corresponding rows in another
- 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



Data Normalization Types – Second Normal Form(2NF)

What is a Foreign Key?



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.

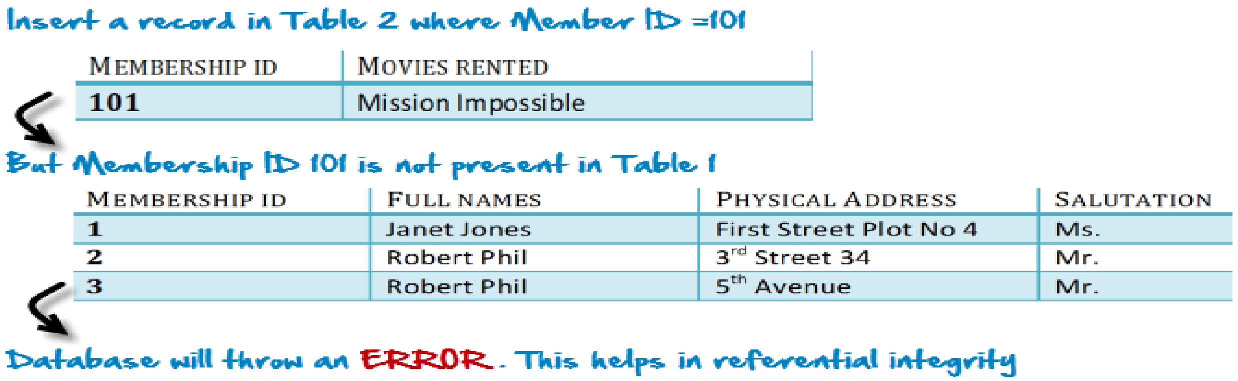
Data Normalization Types – Second Normal Form(2NF)

What is a Foreign Key?

Why do you need a foreign key?

- Suppose, a novice inserts a record in Table B such as
- You will only be able to insert values into your foreign key that exist in the unique key in the parent table.
- This helps in referential integrity.

Data Normalization Types – Second Normal Form(2NF)



Insert a record in Table 2 where Member ID =101

MEMBERSHIP ID	MOVIES RENTED
101	Mission Impossible

But Membership ID 101 is not present in Table 1

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.

Database will throw an ERROR. This helps in referential integrity

- The above problem can be overcome by declaring membership id from Table2 as foreign key of membership id from Table1
- Now, if somebody tries to insert a value in the membership id field that does not exist in the parent table, an error will be shown!

Data Normalization Types – Third Normal Form(3NF)

3rd NF Normalization can be defined as :-

- 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.

Data Normalization Types – Third Normal Form(3NF)

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

TABLE 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

Table 2

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

Table 3

- 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

De-Normalization Fundamental

What Is Denormalization?

- Denormalization is a database optimization technique in which we add redundant data to one or more tables.
- This can help us avoid costly joins in a relational database.
- Note that denormalization does not mean not doing normalization.
- It is an optimization technique that is applied after doing normalization.
- In a traditional normalized database, we store data in separate logical tables and attempt to minimize redundant data.
- We may strive to have only one copy of each piece of data in database.
- For example, in a normalized database, we might have a Courses table and a Teachers table

LearnOA To The Next Level...

De-Normalization Fundamental

What Is Denormalization?

Pros of Denormalization:-

- Retrieving data is faster since we do fewer joins Queries to retrieve can be simpler(and therefore less likely to have bugs)
- since we need to look at fewer tables.

Cons of Denormalization:-

- Updates and inserts are more expensive.
- Denormalization can make update and insert code harder to write.
- Data may be inconsistent . Which is the “correct” value for a piece of data?
- Data redundancy necessitates more storage.

When To Denormalize A Database

- To enhance query performance
- To make a database more convenient to manage
- To facilitate and accelerate reporting

Normalization Vs De-Normalization

Normalization	De-Normalization
• Normalization is the process of dividing the data into multiple tables, so that data redundancy and data integrities are achieved.	• De-Normalization is the opposite process of normalization where the data from multiple tables are combined into one table, so that data retrieval will be faster.
• It removes data redundancy i.e.; it eliminates any duplicate data from the same table and puts into a separate new table.	• It creates data redundancy i.e.; duplicate data may be found in the same table.
• It maintains data integrity i.e.; any addition or deletion of data from the table will not create any mismatch in the relationship of the tables.	• It may not retain the data integrity.
• It increases the number of tables in the database and hence the joins to get the result.	• It reduces the number of tables and hence reduces the number of joins. Hence the performance of the query is faster here compared to normalized tables.
• Even though it creates multiple tables, inserts, updates and deletes are more efficient in this case. If we have to insert/update/delete any data, we have to perform the transaction in that particular table. Hence there is no fear of data loss or data integrity.	• In this case all the duplicate data are at single table and care should be taken to insert/delete/update all the related data in that table. Failing to do so will create data integrity issues.
• Use normalized tables where more number of insert/update/delete operations are performed and joins of those tables are not expensive.	• Use de-normalization where joins are expensive and frequent query is executed on the tables.

LearnOA®
To The Next Level...

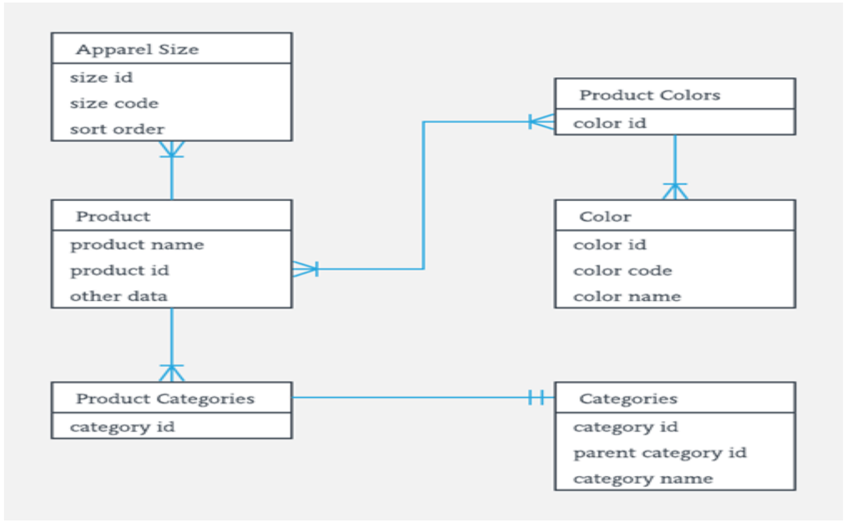
ER - Entity Reltionship Modelling

History of ER Models

- ER diagrams are a visual tool which is helpful to represent the ER model.
- It was proposed by Peter Chen in 1971 to create a uniform convention which can be used for relational database and network.
- He aimed to use an ER model as a conceptual modeling approach.

What is ER Diagrams?

- Entity relationship diagram displays the relationships of entity set stored in a database.
- In other words, we can say that ER diagrams help you to explain the logical structure of databases.
- At first look, an ER diagram looks very similar to the flowchart.
- However, ER Diagram includes many specialized symbols, and its meanings make this model unique.



What is ER Diagrams?

Facts about ER Diagram Model :-

- ER model allows you to draw Database Design
- It is an easy to use graphical tool for modeling data
- Widely used in Database Design
- It is a GUI representation of the logical structure of a Database
- It helps you to identifies the entities which exist in a system and the relationships between those entities

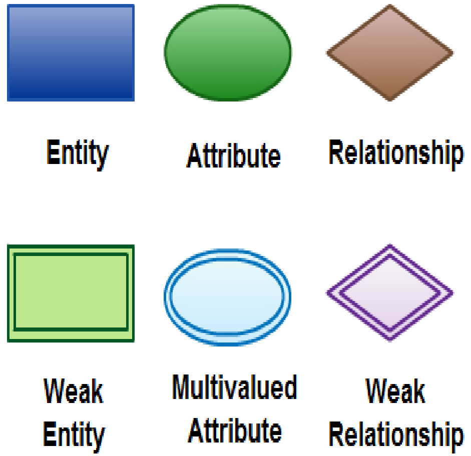
What is ER Diagrams?

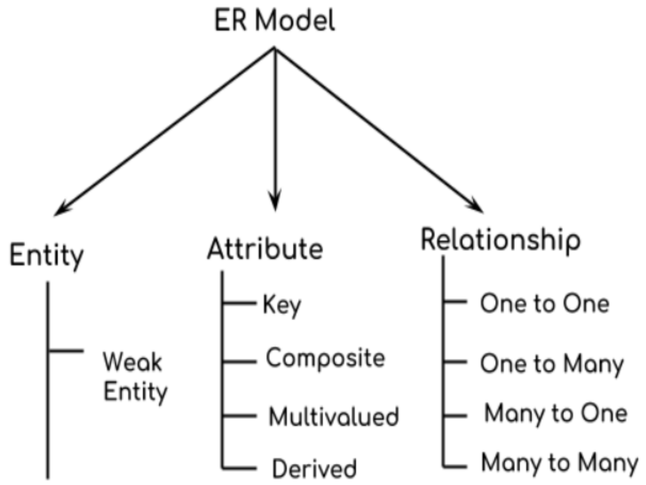
Why use ER Diagrams?

Here, are prime reasons for using the ER Diagram

- Helps you to define terms related to entity relationship modeling
- Provide a preview of how all your tables should connect, what fields are going to be on each table
- Helps to describe entities, attributes, relationships
- ER diagrams are translatable into relational tables which allows you to build databases quickly
- ER diagrams can be used by database designers as a blueprint for implementing data in specific software applications
- The database designer gains a better understanding of the information to be contained in the database with the help of ERP diagram
- ERD is allowed you to communicate with the logical structure of the database to users

Components of the ER Diagram - ER Diagram Symbols and Notations





Components of the ER Diagram - ER Diagram Symbols and Notations

Entity

- An entity can be a person, place, event, or object that is relevant to a given system.
- For example, a school system may include students, teachers, major courses, subjects, fees, and other items. Entities are represented in ER diagrams by a rectangle and named using singular nouns.

Weak Entity

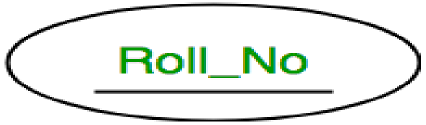
- A weak entity is an entity that depends on the existence of another entity.
- In more technical terms it can be defined as an entity that cannot be identified by its own attributes.
- It uses a foreign key combined with its attributed to form the primary key.
- An entity like order item is a good example for this.
- The order item will be meaningless without an order so it depends on the existence of the order.

Components of the ER Diagram - ER Diagram Symbols and Notations

Attributes

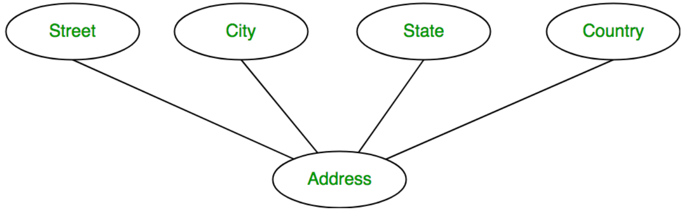
- Attributes are the properties which define the entity type. For example, Roll No, Name, DOB, Age, Address, Mobile No are the attributes which defines entity type Student.
- In ER diagram, attribute is represented by an oval.

Key Attribute : The attribute which **uniquely identifies each entity** in the entity set is called key attribute. For example, Roll No will be unique for each student. In ER diagram, key attribute is represented by an oval with underlying lines.



Components of the ER Diagram - ER Diagram Symbols and Notations

Composite Attribute : An attribute composed of many other attribute is called as composite attribute. For example, Address attribute of student Entity type consists of Street, City, State, and Country. In ER diagram, composite attribute is represented by an oval comprising of ovals.

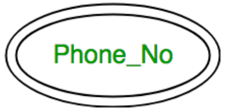


Components of the ER Diagram - ER Diagram Symbols and Notations

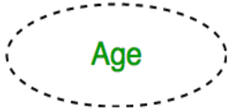
Attributes

- Attributes are the properties which define the entity type. For example, Roll No, Name, DOB, Age, Address, Mobile No are the attributes which defines entity type Student.
- In ER diagram, attribute is represented by an oval.

Multivalued Attribute : An attribute consisting more than one value for a given entity. For example, Phone No (can be more than one for a given student). In ER diagram, multivalued attribute is represented by double oval.



Derived Attribute : An attribute which can be derived from other attributes of the entity type is known as derived attribute. e.g.; Age (can be derived from DOB). In ER diagram, derived attribute is represented by dashed oval.



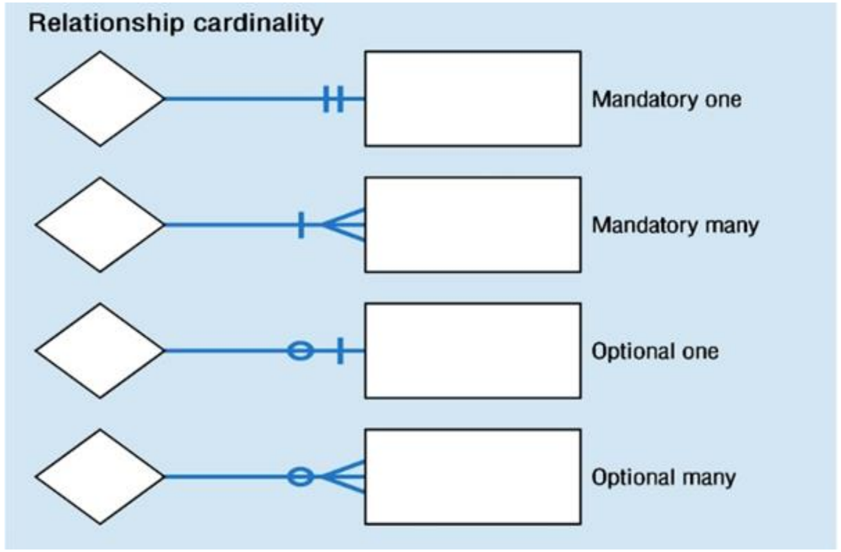
LearnOA To The Next Level...

ER Relationship

- A relationship is represented by diamond shape in ER diagram.
- it shows the relationship among entities.

There are four types of relationships :-

- One-to-One
- One-to-Many
- May to One
- Many-to-Many

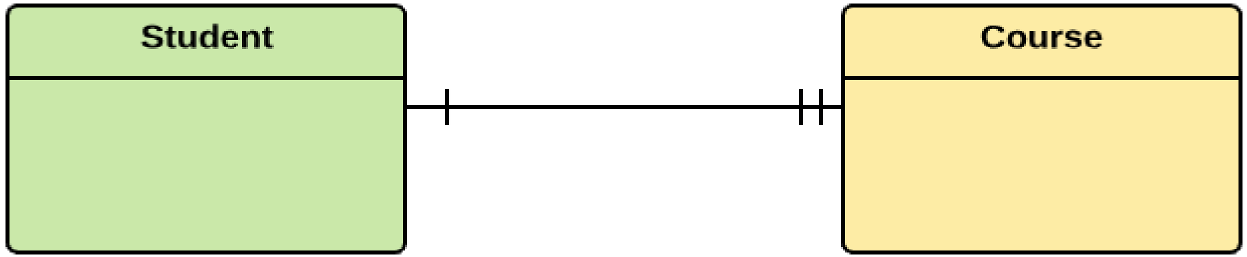


One-to-One

- One entity from entity set X can be associated with at most one entity of entity set Y and vice versa.

Example

- One student can register for numerous courses. However, all those courses have a single line back to that one student.

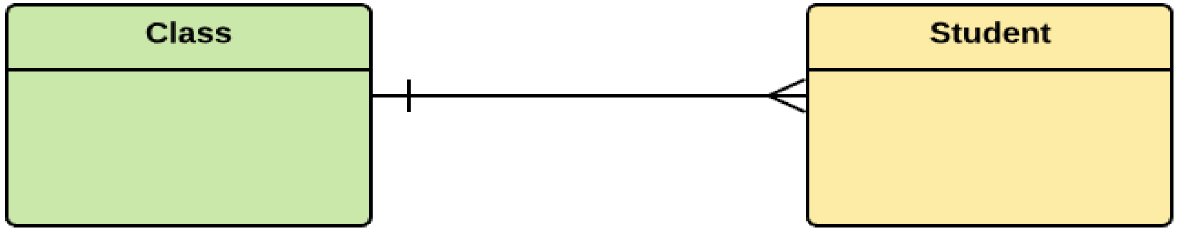


One-to-many

- One entity from entity set X can be associated with multiple entities of entity set Y, but an entity from entity set Y can be associated with at least one entity.

For example

- one class is consisting of multiple students.

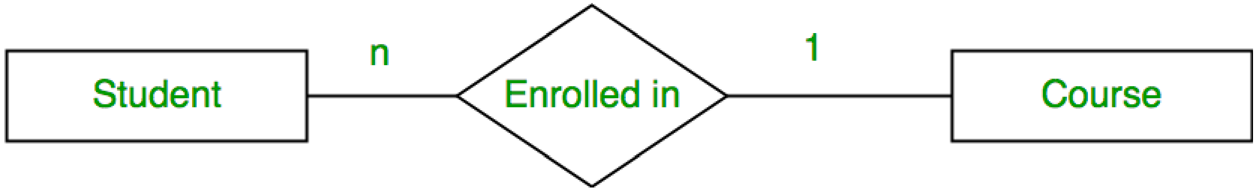


Many-to-one

- More than one entity from entity set X can be associated with at most one entity of entity set Y.
- However, an entity from entity set Y may or may not be associated with more than one entity from entity set X.

For example

- Many students belong to the same class.

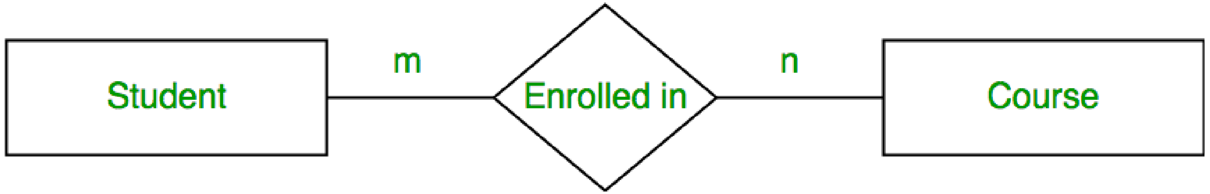


Many-to-many

- One entity from X can be associated with more than one entity from Y and vice versa.

For example

- Students as a group are associated with multiple courses, and courses can be associated with multiple students.



LearnOA To The Next Level...

Relational Data Model concepts

What is Relational Model ?

- The relational model represents the database as a collection of relations.
- A relation is nothing but a table of values.
- Every row in the table represents a collection of related data values.
- These rows in the table denote a real-world entity or relationship.
- The table name and column names are helpful to interpret the meaning of values in each row.
- The data are represented as a set of relations.
- In the relational model, data are stored as tables.
- However, the physical storage of the data is independent of the way the data are logically organized.
- Some popular Relational Database management systems are:

  Oracle

  - SQL Server and Access – Microsoft
  - DB2 and Informix Dynamic Server - IBM

Relational Model Concepts

- Attribute: Each column in a Table. Attributes are the properties which define a relation. e.g., Student Roll no, NAME etc.

- Tables – In the Relational model the, relations are saved in the table format. It is stored along with its entities. A table has two properties rows and columns. Rows represent records and columns represent attributes.

- Tuple – It is nothing but a single row of a table, which contains a single record.

- Relation Schema: A relation schema represents the name of the relation with its attributes.

Relational Model Concepts (Contd.)

- Degree: The total number of attributes which in the relation is called the degree of the relation.
- Cardinality: Total number of rows present in the Table.
- Column: The column represents the set of values for a specific attribute.
- Relation instance – Relation instance is a finite set of tuples in the RDBMS system. Relation instances never have duplicate tuples.
- Relation key - Every row has one, two or multiple attributes, which is called relation key.
- Attribute domain – Every attribute has some pre-defined value and scope which is known as attribute domain

Relational Model Concepts

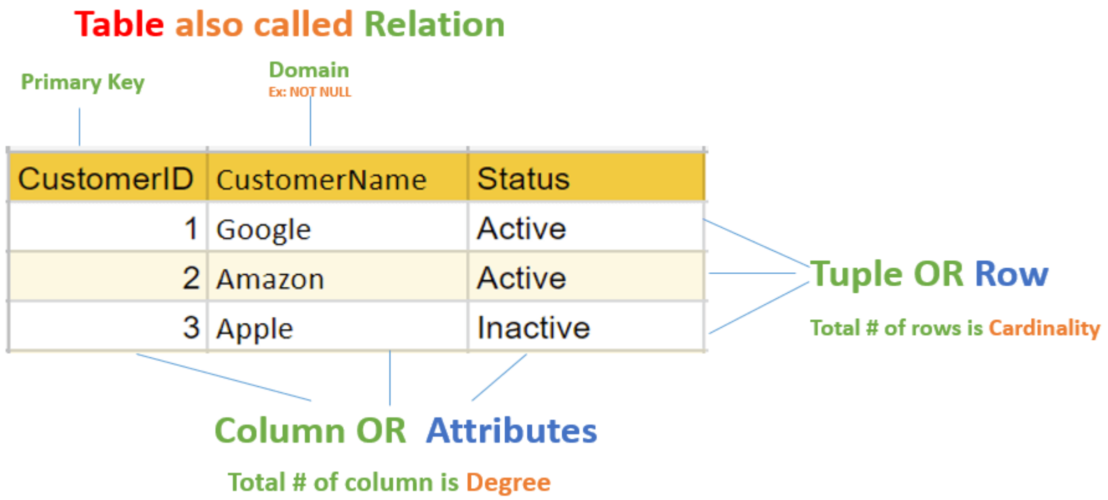


Table also called Relation
Primary Key
Domain
Ex: NOT NULL
Tuple OR Row
Total # of rows is Cardinality
Column OR Attributes
Total # of column is Degree

CustomerID	CustomerName	Status
1	Google	Active
2	Amazon	Active
3	Apple	Inactive

Relational Integrity Constraints

- Relational Integrity constraints is referred to conditions which must be present for a valid relation.
- These integrity constraints are derived from the rules in the mini-world that the database represents.
- There are many types of integrity constraints.
- Constraints on the Relational database management system is mostly divided into three main categories are:
  - Domain constraints
  - Key constraints
  - Referential integrity constraints

Domain / Null Constraints

- Domain constraints can be violated if an attribute value is not appearing in the corresponding domain or it is not of the appropriate data type.
- Domain constraints specify that within each tuple, and the value of each attribute must be unique.
- This is specified as data types which include standard data types integers, real numbers, characters, Booleans, variable length strings, etc.

Example:

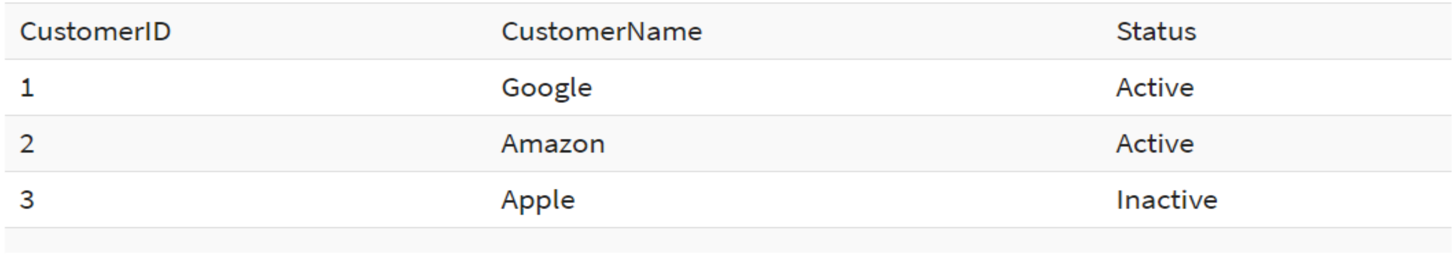
- Create DOMAIN Customer Name CHECK (value not NULL)
- The example shown demonstrates creating a domain constraint such that Customer Name is not NULL

Key / Primary Key Constraints

- An attribute that can uniquely identify a tuple in a relation is called the key of the table.
- The value of the attribute for different tuples in the relation has to be unique.

Example:

- In the given table, **CustomerID** is a key attribute / Primary Key of Customer Table.
- It is most likely to have a single key for one customer, CustomerID =1 is only for the CustomerName =" Google".



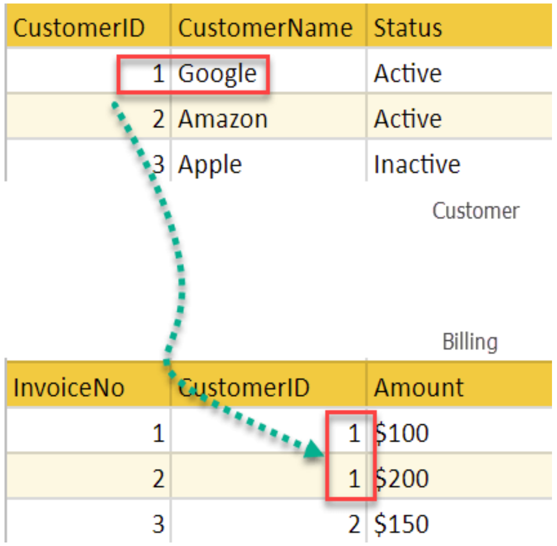
CustomerID	CustomerName	Status
1	Google	Active
2	Amazon	Active
3	Apple	Inactive

Referential Integrity / Foreign Key Constraints

- Referential integrity constraints is base on the concept of Foreign Keys.
- A foreign key is an important attribute of a relation which should be referred to in other relationships.
- Referential integrity constraint state happens where relation refers to a key attribute of a different or same relation.
- However, that key element must exist in the table.

Example

- In the above example, we have 2 relations, Customer and Billing.
- Tuple for CustomerID =1 is referenced twice in the relation Billing. So we know CustomerName=Google has billing amount $300.



CustomerID	CustomerName	Status
1	Google	Active
2	Amazon	Active
3	Apple	Inactive

Customer

Billing

InvoiceNo	CustomerID	Amount
1	1	$100
2	1	$200
3	2	$150

Operations in Relational Model

- Four basic update operations performed on relational database model are

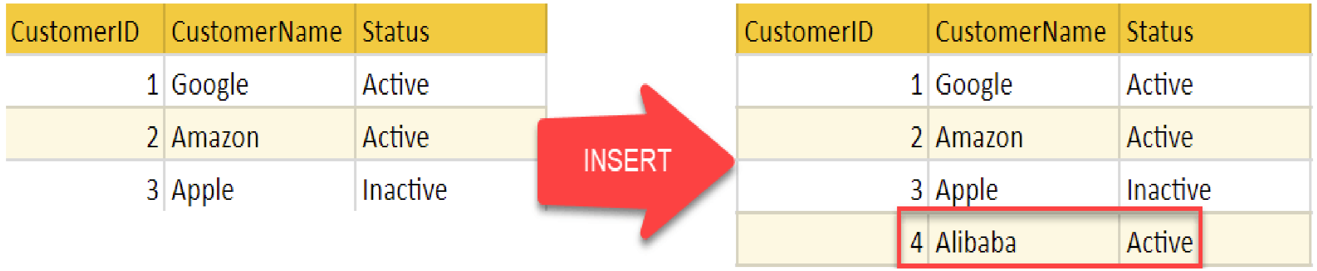
Insert, update, delete and select.

- Insert is used to insert data into the relation
- Delete is used to delete tuples from the table.
- Modify allows you to change the values of some attributes in existing tuples.
- Select allows you to choose a specific range of data.

Whenever one of these operations are applied, integrity constraints specified on the relational database schema must never be violated.

Insert Operation

- The insert operation gives values of the attribute for a new tuple which should be inserted into a relation.

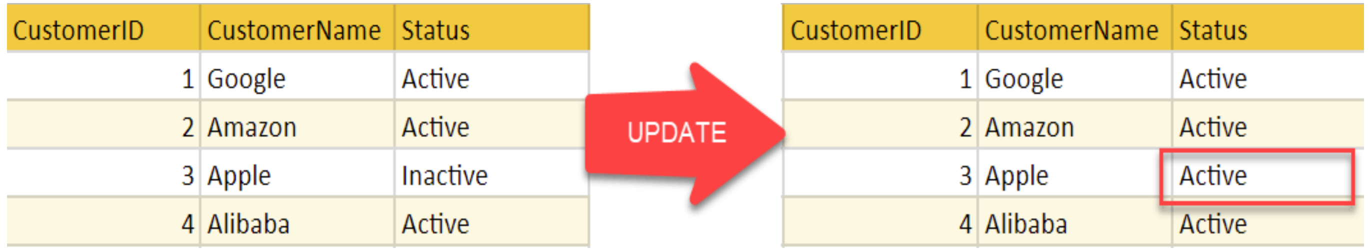


CustomerID	CustomerName	Status
1	Google	Active
2	Amazon	Active
3	Apple	Inactive

CustomerID	CustomerName	Status
1	Google	Active
2	Amazon	Active
3	Apple	Inactive
4	Alibaba	Active

Update Operation

- You can see that in the below-given relation table CustomerName= 'Apple' is updated from Inactive to Active.

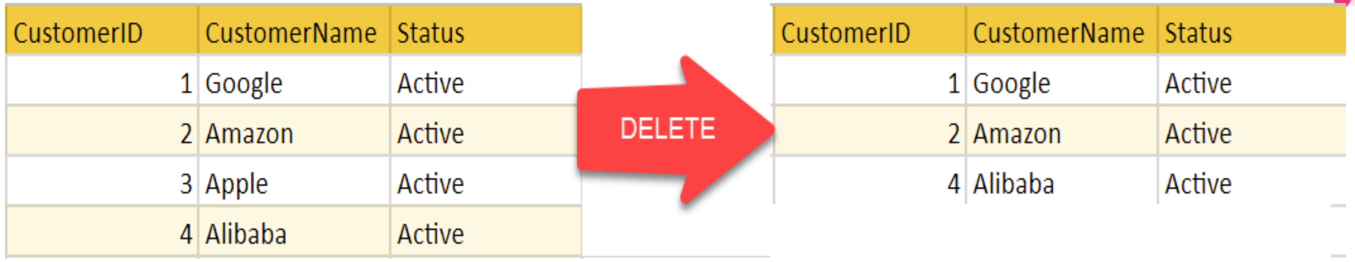


CustomerID	CustomerName	Status
1	Google	Active
2	Amazon	Active
3	Apple	Inactive
4	Alibaba	Active

CustomerID	CustomerName	Status
1	Google	Active
2	Amazon	Active
3	Apple	Active
4	Alibaba	Active

Delete Operation

- To specify deletion, a condition on the attributes of the relation selects the tuple to be deleted.



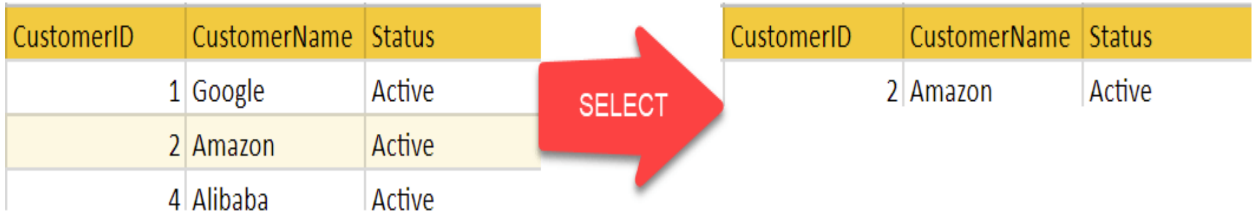
CustomerID	CustomerName	Status
1	Google	Active
2	Amazon	Active
3	Apple	Active
4	Alibaba	Active

CustomerID	CustomerName	Status
1	Google	Active
2	Amazon	Active
4	Alibaba	Active

- In the above-given example, CustomerName= "Apple" is deleted from the table.
- The Delete operation could violate referential integrity if the tuple which is deleted is referenced by foreign keys from other tuples in the same database.

Select Operation

- In the above-given example, CustomerName="Amazon" is selected



CustomerID	CustomerName	Status
1	Google	Active
2	Amazon	Active
4	Alibaba	Active

CustomerID	CustomerName	Status
2	Amazon	Active

Best Practices for creating a Relational Model

- Data need to be represented as a collection of relations
- Each relation should be depicted clearly in the table
- Rows should contain data about instances of an entity
- Columns must contain data about attributes of the entity
- Cells of the table should hold a single value
- Each column should be given a unique name
- No two rows can be identical
- The values of an attribute should be from the same domain

Advantages of using Relational Model

- **Simplicity:** A relational data model is simpler than the hierarchical and network model.
- **Structural Independence:** The relational database is only concerned with data and not with a structure. This can improve the performance of the model.
- **Easy to use:** The relational model is easy as tables consisting of rows and columns is quite natural and simple to understand
- **Query capability:** It makes possible for a high-level query language like SQL to avoid complex database navigation.
- **Data independence:** The structure of a database can be changed without having to change any application.
- **Scalable:** Regarding a number of records, or rows, and the number of fields, a database should be enlarged to enhance its usability.

Disadvantages of using Relational Model

- Few relational databases have limits on field lengths which can't be exceeded.

- Relational databases can sometimes become complex as the amount of data grows, and the relations between pieces of data become more complicated.

- Complex relational database systems may lead to isolated databases where the information cannot be shared from one system to another

Questions

Survey

- Your feedback is important to us, be it a compliment, a suggestion or a complaint. It helps us to make the course better!

  Please spare few minutes to take the survey after the webinar

Thank You!