UNIT 4:

Database Introduction:

What is Data?

Data is a collection of a distinct small unit of information. It can be used in a variety of forms like text, numbers, media, bytes, etc. it can be stored in pieces of paper or electronic memory, etc.

Word 'Data' is originated from the word 'datum' that means 'single piece of information.' It is plural of the word datum.

In computing, Data is information that can be translated into a form for efficient movement and processing. Data is interchangeable.

What is Database?

A database is an organized collection of data, so that it can be easily accessed and managed.

You can organize data into tables, rows, columns, and index it to make it easier to find relevant information.

Database handlers create a database in such a way that only one set of software program provides access of data to all the users.

The main purpose of the database is to operate a large amount of information by storing, retrieving, and managing data.

There are many dynamic websites on the World Wide Web nowadays which are handled through databases. For example, a model that checks the availability of rooms in a hotel. It is an example of a dynamic website that uses a database.

There are many databases available like MySQL, Sybase, Oracle, MongoDB, Informix, PostgreSQL, SQL Server, etc.

Modern databases are managed by the database management system (DBMS).

SQL or Structured Query Language is used to operate on the data stored in a database. SQL depends on relational algebra and tuple relational calculus.

A cylindrical structure is used to display the image of a database.

Types of databases

1. Relational databases

Relational databases have been around since the 1970s. The name comes from the way that data is stored in multiple, related tables. Within the tables, data is stored in rows and columns. The relational database management system (RDBMS) is the program that allows you to create, update, and administer a relational database. Structured Query Language (SQL) is the most common language for reading, creating, updating and deleting data. Relational databases are very reliable. They are compliant with ACID (Atomicity, Consistency, Isolation, Durability), which is a standard set of properties for reliable database transactions. Relational databases work well with structured data. Organizations that have a lot of unstructured or semi-structured data should not be considering a relational database.

Examples: Microsoft SQL Server, Oracle Database, MySQL, PostgreSQL and IBM Db2

2. NoSQL databases

NoSQL is a broad category that includes any database that doesn't use SQL as its primary data access language. These types of databases are also sometimes referred to as non-relational databases. Unlike in relational databases, data in a NoSQL database doesn't have to conform to a pre-defined schema, so these types of databases are great for organizations seeking to store unstructured or semi-structured data. One advantage of NoSQL databases is that developers can make changes to the database on the fly, without affecting applications that are using the database.

Examples: Apache Cassandra, MongoDB, CouchDB, and CouchBase

3. Cloud databases

A cloud database refers to any database that's designed to run in the cloud. Like other cloud-based applications, cloud databases offer flexibility and scalability, along with high availability. Cloud databases are also often low-maintenance, since many are offered via a SaaS model.

Examples: *Microsoft Azure SQL Database, Amazon Relational Database Service, Oracle Autonomous Database, Cloud Spanner, etc.*

4. Columnar databases

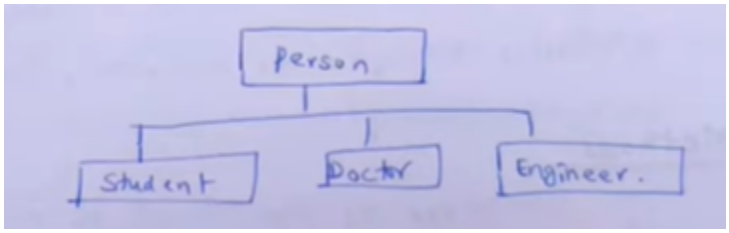
Also referred to as column data stores, columnar databases store data in columns rather than rows. These types of databases are often used in data warehouses because they're great at handling analytical queries. When you're querying a columnar database, it essentially ignores all of the data that doesn't apply to the query, because you can retrieve the information from only the columns you want.

Examples: Google BigQuery, Cassandra, HBase, MariaDB, Azure SQL Data Warehouse

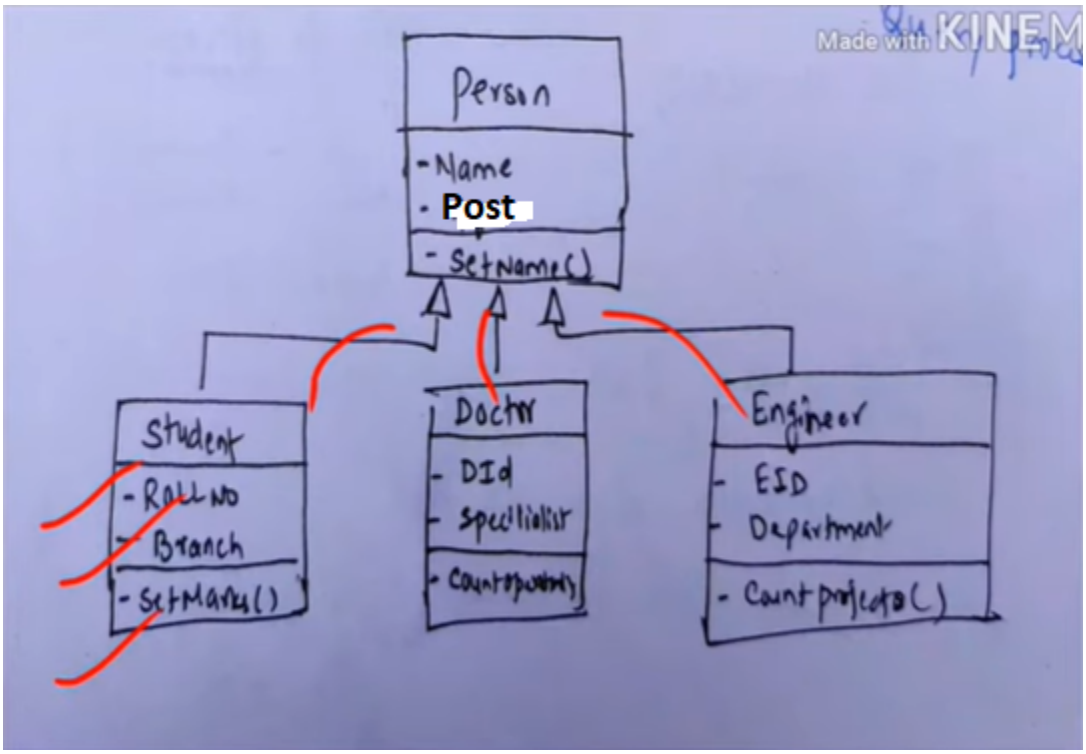
5. Object-oriented databases

An object-oriented database is based on object-oriented programming, so data and all of its attributes, are tied together as an object. Object-oriented databases are managed by object-oriented database management systems (OODBMS). These databases work well with object-oriented programming languages, such as C++ and Java. Like relational databases, object-oriented databases conform to ACID standards.

Examples: Wakanda, ObjectStore



(inheritance)



(count operation)

Database Design:

Database Design is a collection of processes that facilitate the ***designing, development, implementation and maintenance of enterprise data management systems***. Properly designed database are easy to maintain, improves data consistency and are cost effective in terms of disk storage space. The database designer decides how the data elements correlate and what data must be stored.

The main objectives of database design in DBMS are to produce logical and physical designs models of the proposed database system.

The logical model concentrates on the data requirements and the data to be stored independent of physical considerations. It does not concern itself with how the data will be stored or where it will be stored physically.

The physical data design model involves translating the logical DB design of the database onto physical media using hardware resources and software systems such as database management systems (DBMS).

Why Database Design is Important?

- It helps produce database systems
- That meet the requirements of the users
- Have high performance.

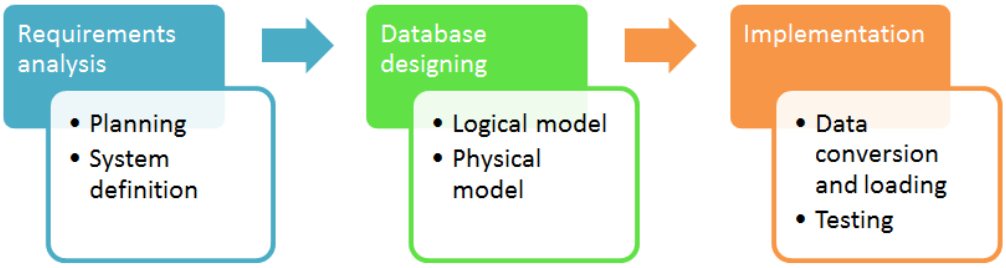
Database design process in DBMS is crucial for high performance database system.

The database development life cycle has a number of stages that are followed when developing database systems.

The steps in the development life cycle do not necessarily have to be followed religiously in a sequential manner.

On small database systems, the process of database design is usually very simple and does not involve a lot of steps.

In order to fully appreciate the above diagram, let's look at the individual components listed in each step for overview of design process in DBMS.



Requirement's analysis

Planning – These stages of database design concepts are concerned with planning of entire Database Development Life Cycle. It takes into consideration the Information Systems strategy of the organization.

System definition – This stage defines the scope and boundaries of the proposed database system.

Database designing

Logical model – This stage is concerned with developing a database model based on requirements. The entire design is on paper without any physical implementations or specific DBMS considerations.

Physical model – This stage implements the logical model of the database taking into account the DBMS and physical implementation factors.

Implementation

Data conversion and loading – this stage of relational databases design is concerned with importing and converting data from the old system into the new database.

Testing – this stage is concerned with the identification of errors in the newly implemented system. It checks the database against requirement specifications.

Two Types of Database Techniques

- Normalization
- ER Modeling

Relational Database Model:

Relational Model (RM) 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:

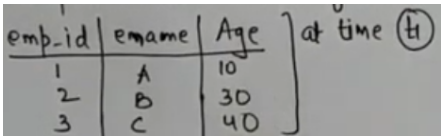
DB2 and Informix Dynamic Server – IBM

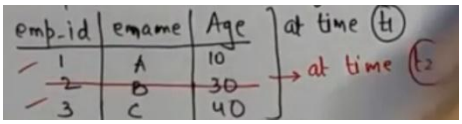
Oracle and RDB – Oracle

SQL Server and Access – Microsoft

Relational Model Concepts

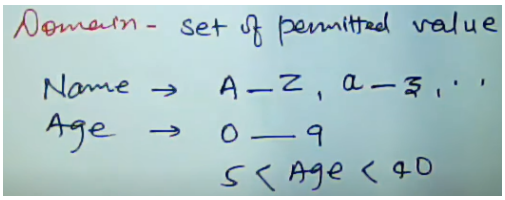
- **Attribute**- Each column in a Table. Attributes are the properties which define a relation. e.g., Student_Rollno, NAME, etc.
- **Tables** – In the Relational model the, relations are saved in the table format. It is stored along with its entities. A table has two properties rows and columns. Rows 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 (Emp) with its attributes (emp_id, emp_name, Age). *Eg: Emp (emp_id, emp_name, Age)*
- **Relation Database Schema:** collection of one or more relation schema.
  Eg: *Emp (emp_id, emp_name, Age)*
  Dep (dep_id, dep_name)
- **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.

(i) 

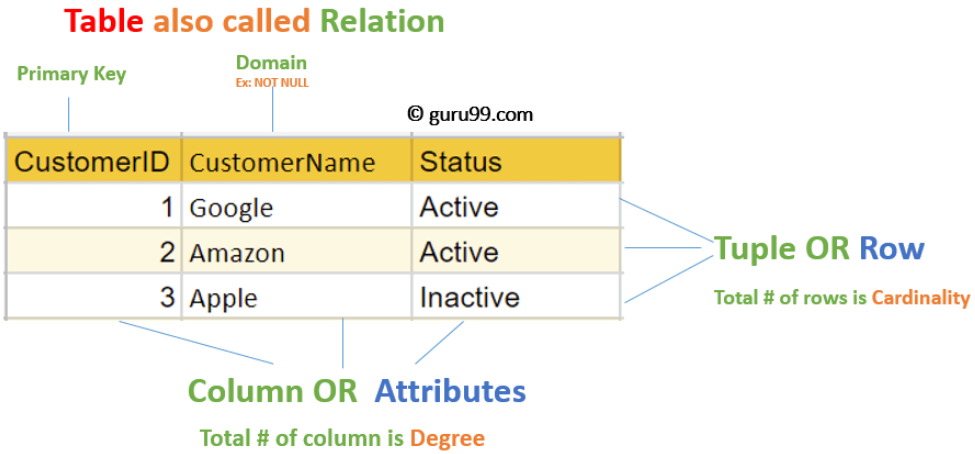
(ii) 

the no. 2 entry has been deleted and updated database.

- **Relation key** – Each row has one or more attributes, known as relation key, which can identify the row in the relation (table) uniquely.
- **Attribute domain** – Every attribute has some pre-defined value and scope which is known as attribute domain.



Definition of RDBMS through Table:



Constraints = (Conditions)

Every relation has some conditions that must hold for it to be a valid relation. Eg: if you are creating Mail Id, if its available, then it will show you the available gmail.

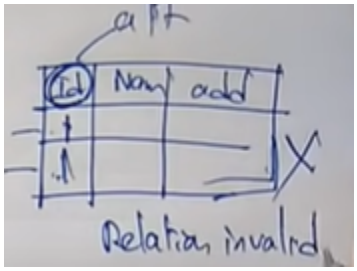
- Should be Unique. Eg, abc123@gmail.com cannot be dedicated to multiple users. Or while creating the password, you need to input the password according to the condition, ie, capital+small alphabets+symbol = strong password.
- The field shouldn't be empty. Not allowed of Null Value.

These conditions are called Relational Integrity Constraints. There are three main integrity constraints −

- Key constraints or Entity Constraints
- Domain constraints (Domain means the value inside the field)
- Referential integrity constraints

Key Constraints

There must be at least one minimal subset of attributes in the relation, which can identify a tuple uniquely. This minimal subset of attributes is called key for that relation. If there are more than one such minimal subsets, these are called candidate keys. Eg: we are selecting one attribute as a key like Emp_id.



(if we enter value '1' in two fields, then it will be invalid. We need to enter 2 on next field. No two tuples should have the same value.)

Key constraints force that −

- in a relation with a key attribute, no two tuples can have identical values for key attributes.
- a key attribute cannot have NULL values.
- Key constraints are also referred to as Entity Constraints.

Domain Constraints

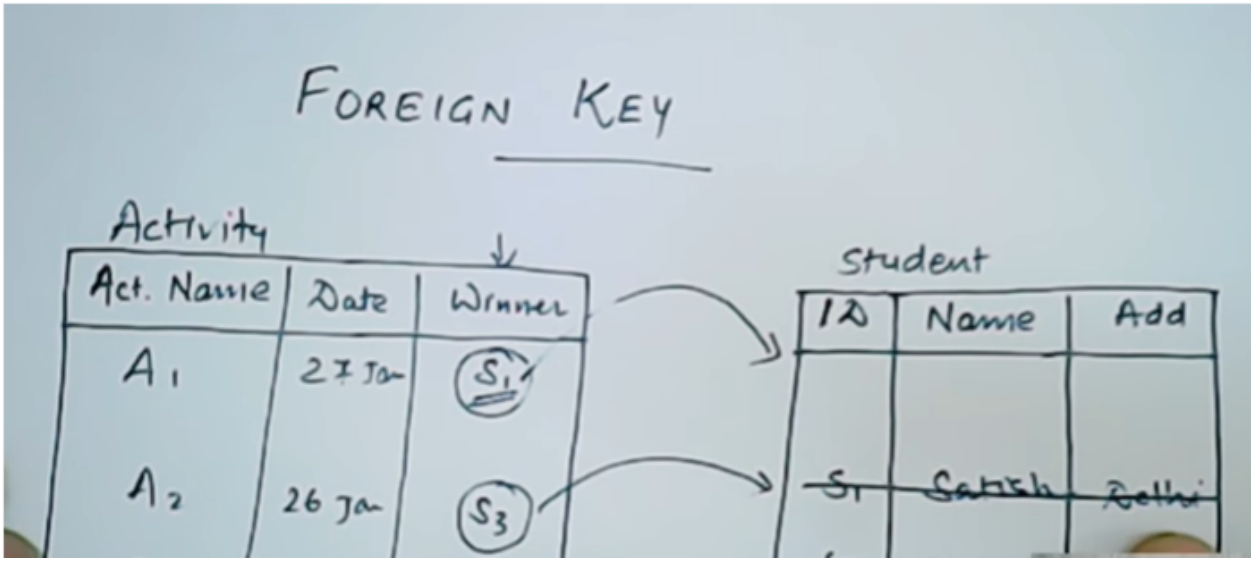
Attributes have specific values in real-world scenario. For example, age can only be a positive integer. The same constraints have been tried to employ on the attributes of a relation. Every attribute is bound to have a specific range of values. *For example, age cannot be less than zero and telephone numbers cannot contain a digit outside 0-9. We cannot enter person's name in their id and so on. ID should always be an integer, name should always be Character.*

Referential integrity Constraints

Referential integrity constraints work on the concept of Foreign Keys. A foreign key is a key attribute of a relation that can be referred in other relation.

Referential integrity constraint states that if a relation refers to a key attribute of a different or same relation, then that key element must exist.

The set of attributes that form foreign key of relation R1 should have the same domain as the primary key of the referenced relation R2.



(So while deleting the data in student, we should delete the main source value in activity.)

Normalization:

It is the process of organizing the data in a database. It helps in removing the duplicate values in the database. Normalization divides the large table into smaller tables and links them using relationships.

The normal form is used to reduce redundancy from the database table. Normalization is the name given to the process of simplifying the relationship among data elements in a record.

In simple words we can say,

Normalization is the process of organizing data to minimize.

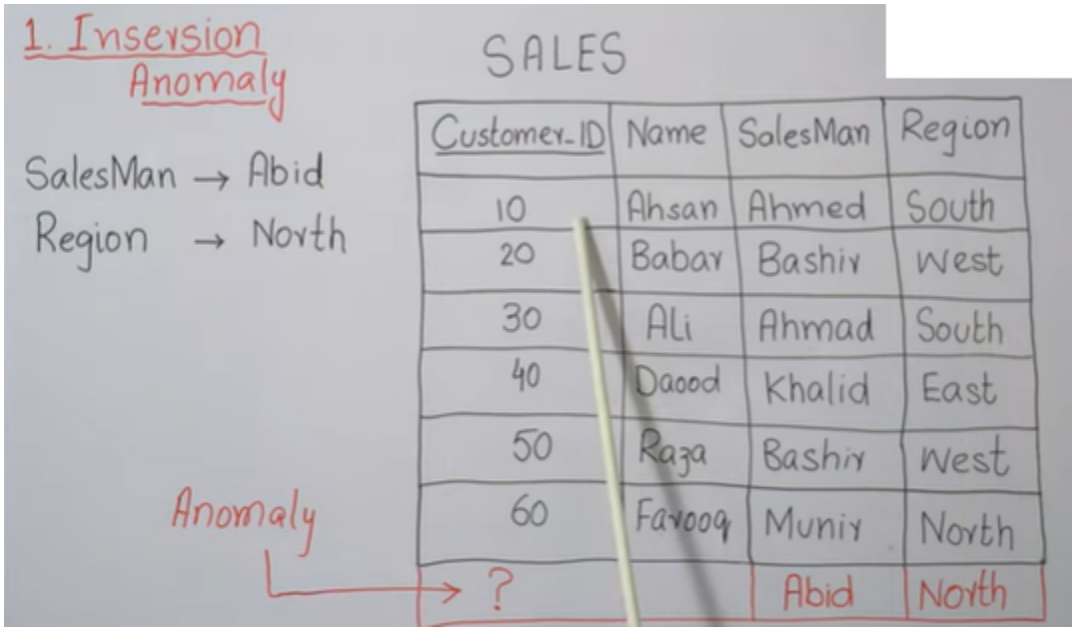
- Redundancy/duplication/repetition.
- Insertion, deletion, updating anomalies.

Normal forms

- First Normal Form (1NF)
- Second Normal Form (2NF)
- Third Normal Form (3NF)
- Boyce-Codd Normal Form (BCNF)

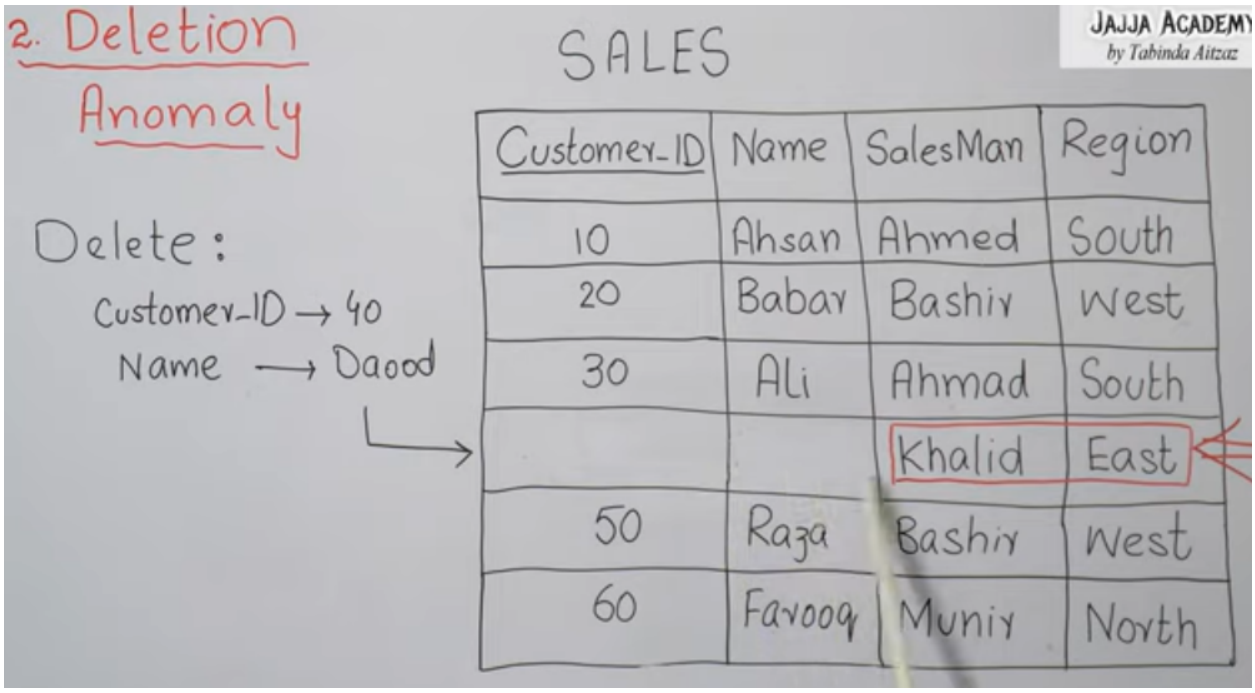
If a database design is not perfect, it may contain anomalies, which are like a bad dream for any database administrator. Managing a database with anomalies is next to impossible.

- **Insert anomalies** – We tried to insert data in a record that does not exist at all.



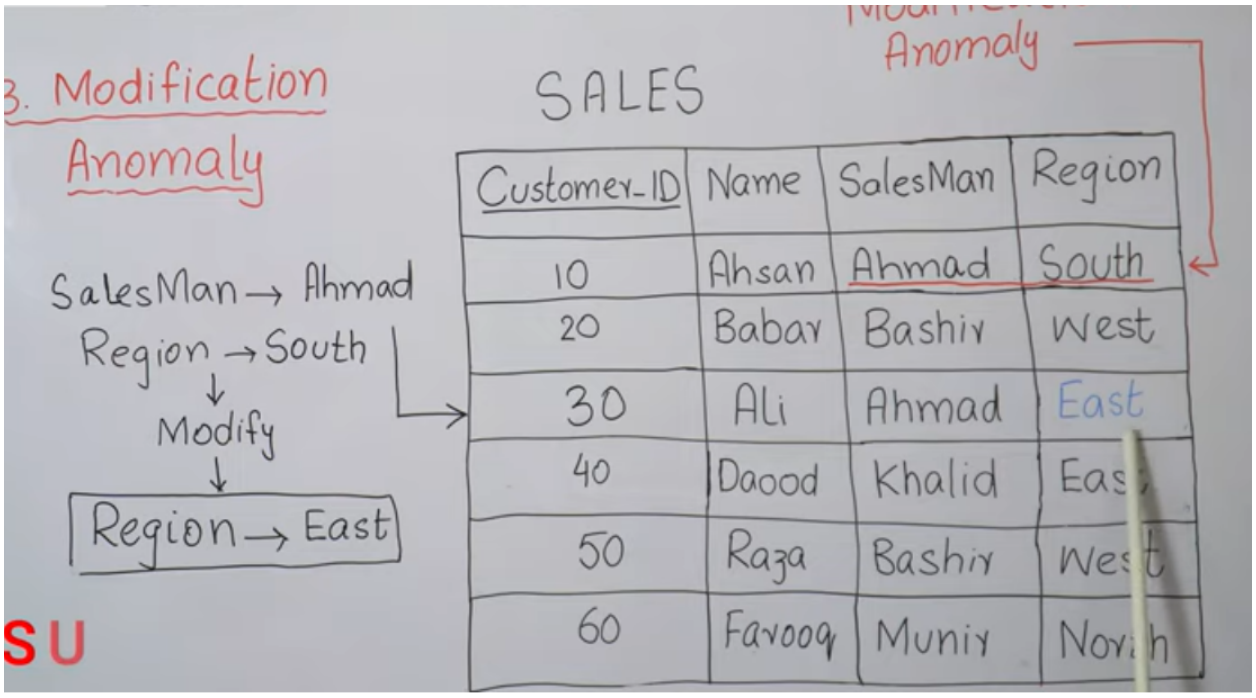
(Here, we tried to enter the new value Abid as a salesman and North as a Region. But without Primary Key (Customer_ID) its not possible to do so. Or if we put any random customer id (like #,$) then it might create problem in future also we can't leave it NULL.)

- **Deletion anomalies** – We tried to delete a record, but parts of it was left undeleted because of unawareness, the data is also saved somewhere else.



While deleting the customer id (Primary Key) the entry is not consistent and cannot be feasible to delete that single entry. Also, when we need to delete the customer id and name, we removed all the entry of that row from the table.
DELETE FROM SALES,
WHERE Customer_ID=40; //it will remove the whole row including the salesman information too, which I cannot recover it later.

- **Update anomalies** − If data items are scattered and are not linked to each other properly, then it could lead to strange situations. For example, when we try to update one data item having its copies scattered over several places, a few instances get updated properly while a few others are left with old values. Such instances leave the database in an inconsistent state.



Normalization is a method to remove all these anomalies and bring the database to a consistent state. If we need to split the table, we can. And when we need to update, insert and update the table, then there won't be any problem to do so.

First Normal Form (1NF):

First Normal Form is defined in the definition of relations (tables) itself. This rule defines that all the attributes in a relation must have atomic domains. The values in an atomic domain are indivisible units.

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

We re-arrange the relation (table) as below, to convert it to First Normal Form.

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

Each attribute must contain only a single value from its pre-defined domain.

2NF:

To be in second normal form, a relation must be in first normal form and relation must not contain any partial dependency. A relation is in 2NF if it has No Partial Dependency, i.e., no non-prime attribute (attributes which are not part of any candidate key) is dependent on any proper subset of any candidate key of the table.

Partial Dependency – If the proper subset of candidate key determines non-prime attribute, it is called partial dependency.

Prime vs Non-Prime attributes: **Prime Attribute** means which is part of Candidate Key. **Candidate Key** can be a person's unique ID which can be made primary key like: Citizenship no., license no., voter id no., phone no, reg no, roll no, etc. **Non-Prime Attribute** an attribute that is not part of any candidate key.

Partial vs Transitive Dependencies:

As stated, the non-prime attributes i.e. StudentName and ProjectName should be functionally dependent on part of a candidate key, to be Partial Dependent. The StudentName can be determined by StudentID, which makes the relation Partial Dependent.

A transitive dependency occurs when one non-prime attribute is dependent on another non-prime attribute.

Example: Let's assume, a school can store the data of teachers and the subjects they teach. In a school, a teacher can teach more than one subject.

TEACHER table

TEACHER_ID	SUBJECT	TEACHER_AGE
25	Chemistry	30

25	Biology	30
47	English	35
83	Math	38
83	Computer	38

In the given table, non-prime attribute TEACHER_AGE is dependent on TEACHER_ID which is a proper subset of a candidate key. That's why it violates the rule for 2NF. To convert the given table into 2NF, we decompose it into two tables:

TEACHER_DETAIL table:

TEACHER_ID	TEACHER_AGE
25	30
47	35
83	38

TEACHER_SUBJECT table:

TEACHER_ID	SUBJECT
25	Chemistry
25	Biology
47	English
83	Math
83	Computer

3NF:

- A relation will be in 3NF if it is in 2NF and not contain any transitive partial dependency.
- 3NF is used to reduce the data duplication. It is also used to achieve the data integrity.
- If there is no transitive dependency for non-prime attributes, then the relation must be in third normal form.

A relation is in third normal form if it holds atleast one of the following conditions for every non-trivial function dependency X → Y.

1. X is a super key.
2. Y is a prime attribute, i.e., each element of Y is part of some candidate key.

Example:

EMPLOYEE_DETAIL table:

EMP_ID	EMP_NAME	EMP_ZIP	EMP_STATE	EMP_CITY
222	Harry	201010	UP	Noida
333	Stephan	02228	US	Boston
444	Lan	60007	US	Chicago
555	Katharine	06389	UK	Norwich
666	John	462007	MP	Bhopal

Super key in the table above:

The set of attributes that can uniquely identify a tuple is known as Super Key.
i.e:{EMP_ID}, {EMP_ID, EMP_NAME}, {EMP_ID, EMP_NAME, EMP_ZIP}..so on
Candidate key: {EMP_ID}

Non-prime attributes: In the given table, all attributes except EMP_ID are non-prime.

Here, EMP_STATE & EMP_CITY dependent on EMP_ZIP and EMP_ZIP dependent on EMP_ID. The non-prime attributes (EMP_STATE, EMP_CITY) transitively dependent on super key(EMP_ID). It violates the rule of third normal form.

That's why we need to move the EMP_CITY and EMP_STATE to the new <EMPLOYEE_ZIP> table, with EMP_ZIP as a Primary key.

EMPLOYEE table:

EMP_ID	EMP_NAME	EMP_ZIP
222	Harry	201010

333	Stephan	02228
444	Lan	60007
555	Katharine	06389
666	John	462007

EMPLOYEE_ZIP table:

EMP_ZIP	EMP_STATE	EMP_CITY
201010	UP	Noida
02228	US	Boston
60007	US	Chicago
06389	UK	Norwich
462007	MP	Bhopal

BCNF:

- (**Left hand side of each Functional Dependency should be Candidate Key**)
- (Functional Dependency (FD) is a constraint that determines the relation of one attribute to another attribute)

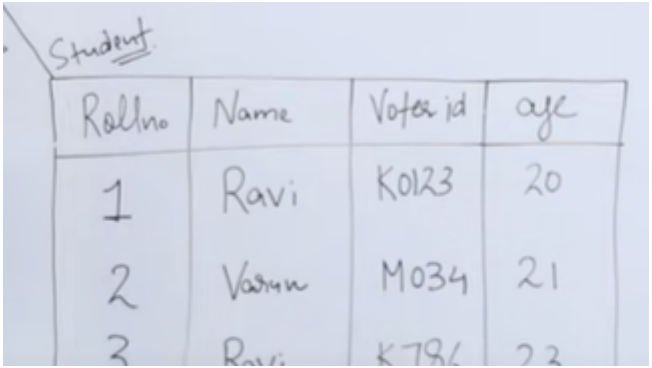
CK = Roll no, Voter Id

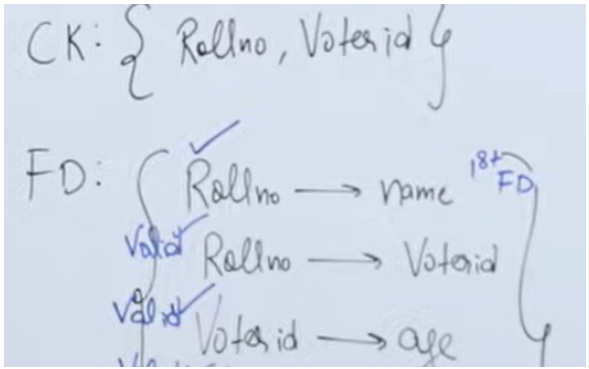
FD = Roll No --→ Name

Roll No --→ Voter ID

Voter ID -→ Age

Voter ID -→ Roll No

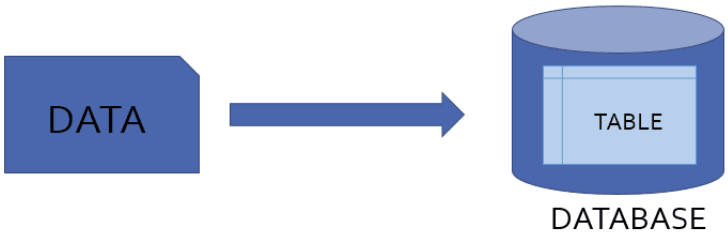




First step of any relational database design is to make ER Diagram for it and then convert it into relational Model.

What is relational model:

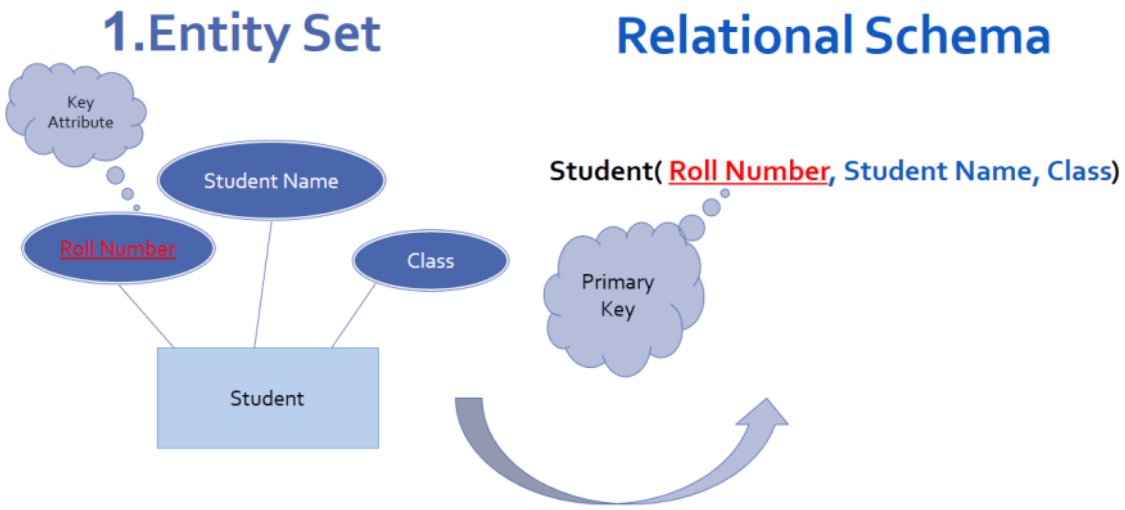
Relational Model represents how data is stored in database in the form of table.



1. Entity Set:

Consider we have entity STUDENT in ER diagram with attributes Roll Number, Student Name and Class.
To convert this entity set into relational schema

1. Entity is mapped as relation in Relational schema
2. Attributes of Entity set are mapped as attributes for that Relation.
3. Key attribute of Entity becomes Primary key for that Relation.



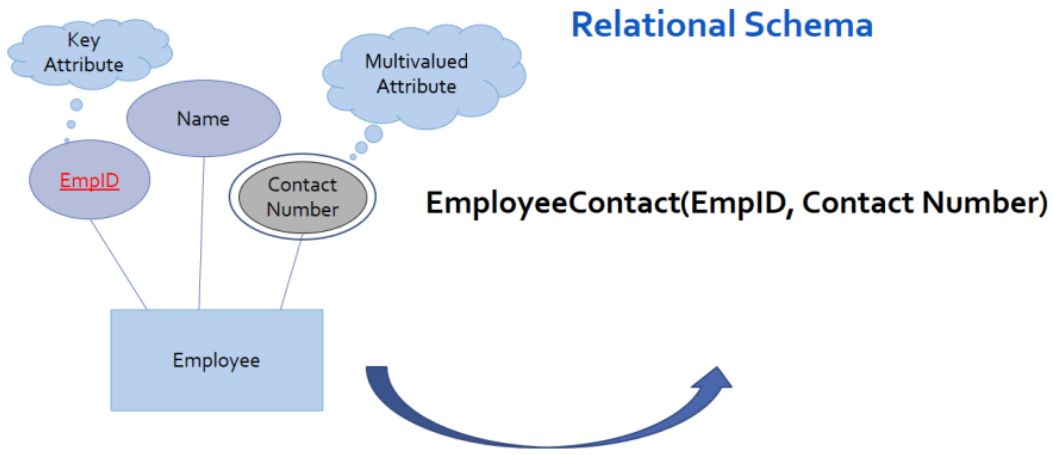
2.Entity set with multi valued attribute:

Consider we have entity set Employee with attributes Employee ID, Name and Contact number. Here contact number is multivalued attribute as it has multiple values. as an employee can have more than one contact number for that we have to repeat all attributes for every new contact number. This will lead to data redundancy in table.

Hence to convert entity with multivalued attribute into relational schema, separate relation is created for multivalued attribute in which:

1. Key attribute and multivalued attribute of entity set becomes primary key of relation.
2. Separate relation employee is created with remaining attributes.

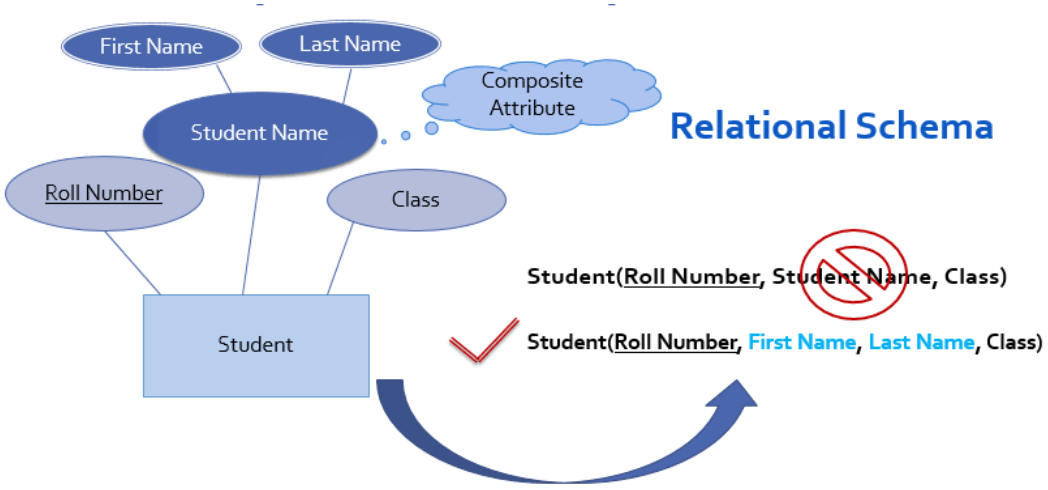
Due to this instead of repeating all attributes of entity now only one attribute is need to repeat.



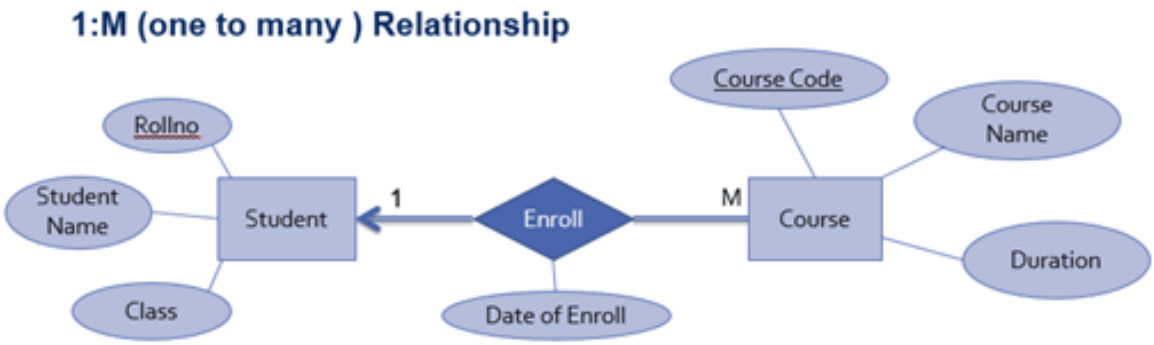
3. Entity set with Composite attribute:

Consider entity set student with attributes Roll Number, Student Name and Class. Here student name is composite attribute as it has further divided into First name, last name.

In this case to convert entity into relational schema, composite attribute student name should not be included in relation but all parts of composite attribute are mapped as simple attributes for relation



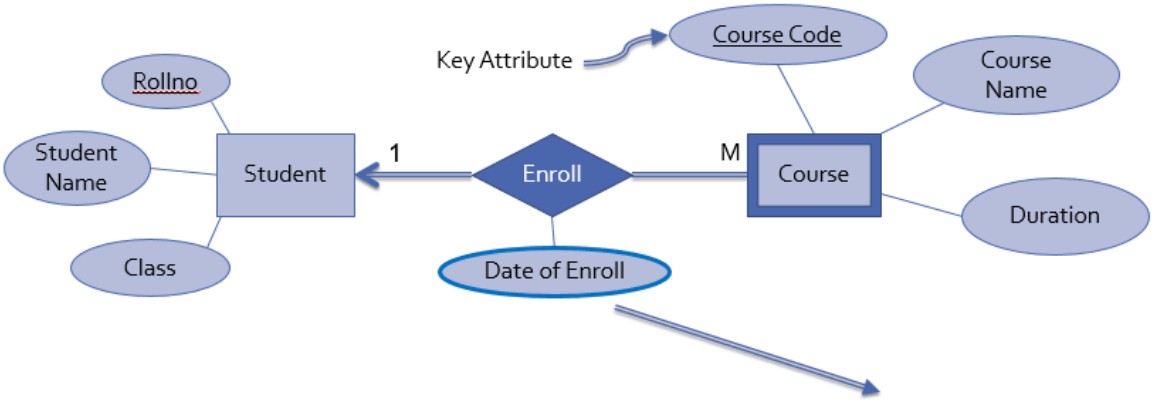
4. 1:M (one to many) Relationship: Consider 1:M relationship set enrolled exist between entity sets student and course as follow,



Attributes of entity set student are Roll no which is primary key, student name and class
Attributes of entity set course are Course code which is primary key, Course name and duration
And date of enroll is attribute of relationship set enroll.

Here Enroll is 1:M relationship exist between entity set student and course which means that *one student can enroll in multiple courses.* In this case to convert this relationship into relational schema,

1.Separate relation is created for all participating entity sets (student and course)
2. Key attribute of Many's side entity set (course) is mapped as foreign key in one's side relation (Student)
3.All attributes of relationship set are mapped as attributes for relation of one's side entity set (student)



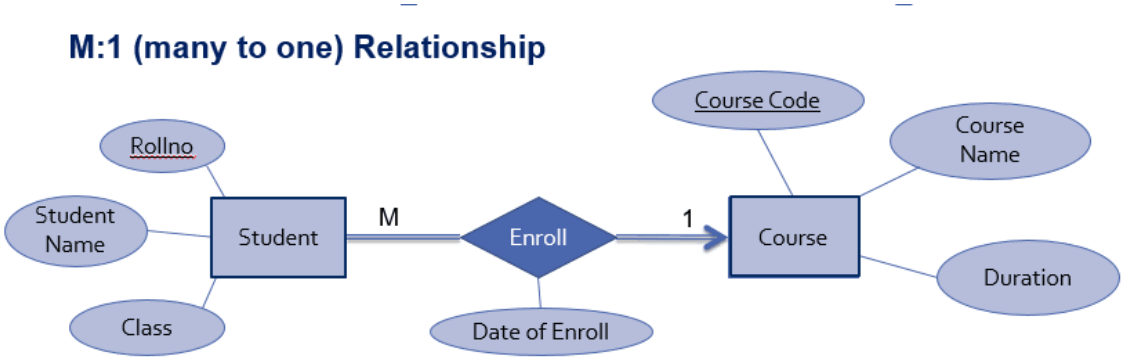
Student(Rollno, Student Name, Class , Course Code, Date of Enroll)

Foreign Key

Course(Course Code, Course Name, Duration)

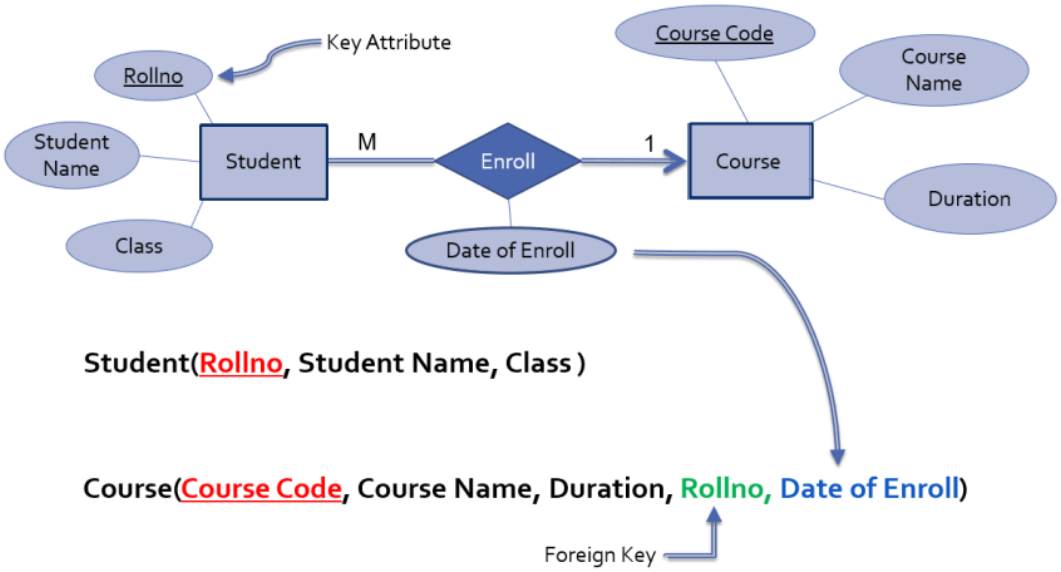
5. M:1 (many to one) Relationship:

Consider same relationship set enroll exist between entity sets student and course but here student is many side entity set while course is one side entity set. Which means many student can enroll in one course.



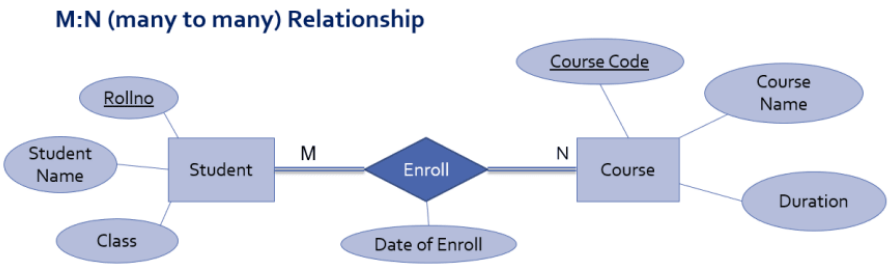
To convert this relationship set into relational schema:

1. Separate relation is created for all participating entity sets.
2. Key attribute of Many's side entity set student is mapped as foreign key in one's side relation
3. All attributes of relationship set are mapped as attributes for one's side relation course.



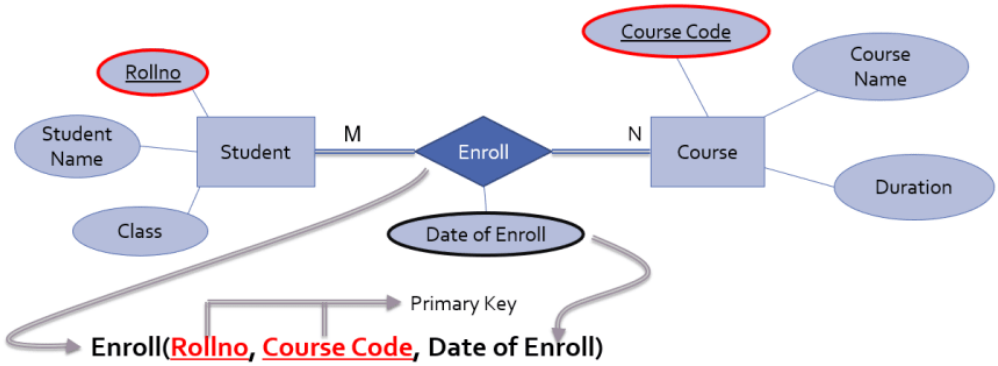
6. M:N (many to many) Relationship:

Consider same relationship set enrolled exist between entity sets student and course, which means multiple students can enroll in multiple courses.



To convert this Relationship set into relational schema,

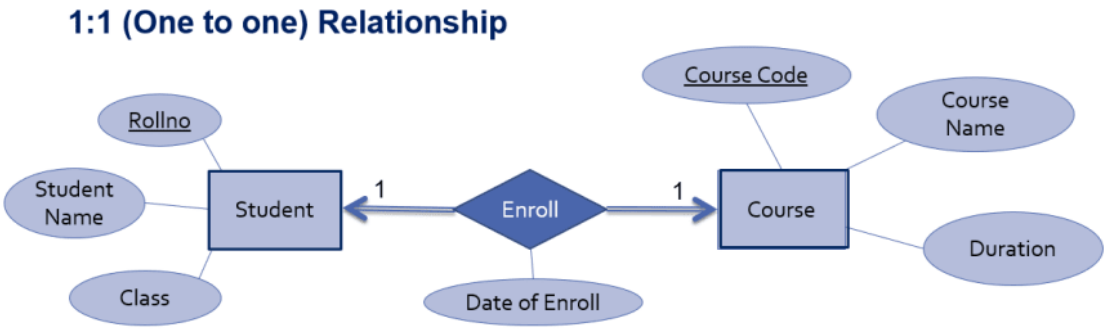
1. Relationship set is mapped as separate relation
2. Key attributes of participating entity sets are mapped as primary key for that relation
3. Attribute of relationship set becomes simple attributes for that relation
4. And separate relation is created for other participating entities



5.

7. 1:1 (one to one) Relationship:

Consider same relationship set enroll exist between entity sets student and course, which means one student can enroll in only one courses



To convert this Relationship set into relational schema,

1. Separate relation is created for all participating entity sets.
2. Primary Key of Relation Student can be act as foreign key for relation Course OR Primary Key of Relation Course act as foreign key for relation Student.