Database Operations

Tushar B. Kute,
http://tusharkute.com

Constructs in SQL

- Queries : Retrieves data against some criteria.
- Statements : Controls transactions, program flow, connections, sessions, or diagnostics.
- Clauses : Components of Queries and Statements.
- Expressions : Combination of symbols and operators and a key part of the SQL statements.
- Predicates : Specifies conditions.

Data Model

- Data modeling (data modelling) is the process of creating a data model for the data to be stored in a database.
- This data model is a conceptual representation of Data objects, the associations between different data objects, and the rules.
- Data modeling helps in the visual representation of data and enforces business rules, regulatory compliances, and government policies on the data.
- Data Models ensure consistency in naming conventions, default values, semantics, security while ensuring quality of the data.

Data Model

- The Data Model is defined as an abstract model that organizes data description, data semantics, and consistency constraints of data.
- The data model emphasizes on what data is needed and how it should be organized instead of what operations will be performed on data.
- Data Model is like an architect's building plan, which helps to build conceptual models and set a relationship between data items.
- The two types of Data Modeling Techniques are
  - Entity Relationship (E-R) Model
  - UML (Unified Modelling Language)

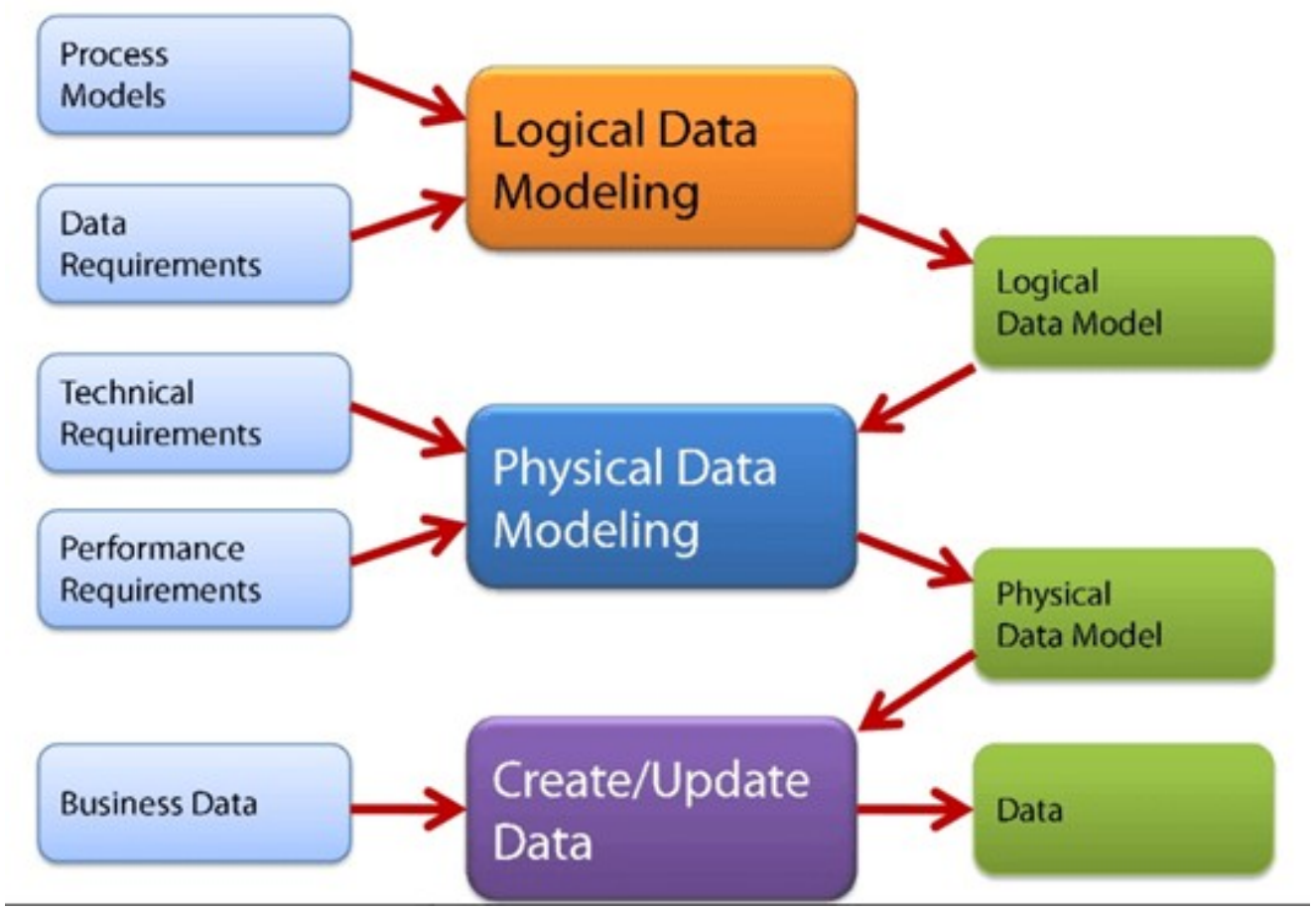
Why Data Model?

- Ensures that all data objects required by the database are accurately represented. Omission of data will lead to creation of faulty reports and produce incorrect results.
- A data model helps design the database at the conceptual, physical and logical levels.
- Data Model structure helps to define the relational tables, primary and foreign keys and stored procedures.
- It provides a clear picture of the base data and can be used by database developers to create a physical database.
- It is also helpful to identify missing and redundant data.
- Though the initial creation of data model is labor and time consuming, in the long run, it makes your IT infrastructure upgrade and maintenance cheaper and faster.

Types of Models

- Conceptual Data Model: This Data Model defines WHAT the system contains. This model is typically created by Business stakeholders and Data Architects. The purpose is to organize, scope and define business concepts and rules.
- Logical Data Model: Defines HOW the system should be implemented regardless of the DBMS. This model is typically created by Data Architects and Business Analysts. The purpose is to developed technical map of rules and data structures.
- Physical Data Model: This Data Model describes HOW the system will be implemented using a specific DBMS system. This model is typically created by DBA and developers. The purpose is actual implementation of the database.

Types of Models



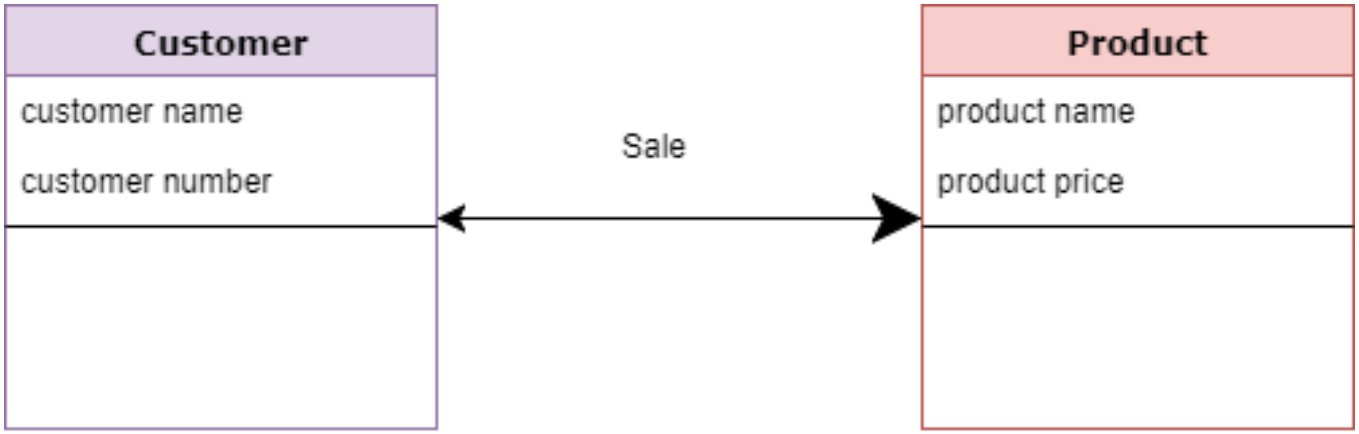
Conceptual Data Model

- A Conceptual Data Model is an organized view of database concepts and their relationships. The purpose of creating a conceptual data model is to establish entities, their attributes, and relationships.
- In this data modeling level, there is hardly any detail available on the actual database structure. Business stakeholders and data architects typically create a conceptual data model.
- The 3 basic tenants of Conceptual Data Model are
  - Entity: A real-world thing
  - Attribute: Characteristics or properties of an entity
  - Relationship: Dependency or association between two entities

Conceptual Data Model

- Data model example:
  - Customer and Product are two entities. Customer number and name are attributes of the Customer entity
  - Product name and price are attributes of product entity
  - Sale is the relationship between the customer and product

Conceptual Data Model



Conceptual Data Model: Features

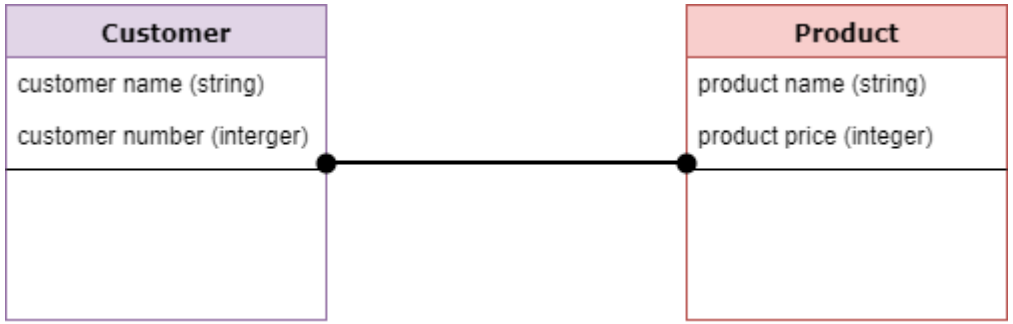
- Offers Organisation-wide coverage of the business concepts.
- This type of Data Models are designed and developed for a business audience.
- The conceptual model is developed independently of hardware specifications like data storage capacity, location or software specifications like DBMS vendor and technology. The focus is to represent data as a user will see it in the “real world.”

Logical Data Model

- The Logical Data Model is used to define the structure of data elements and to set relationships between them.
- The logical data model adds further information to the conceptual data model elements.
- The advantage of using a Logical data model is to provide a foundation to form the base for the Physical model. However, the modeling structure remains generic.

Logical Data Model

- At this Data Modeling level, no primary or secondary key is defined.
- At this Data modeling level, you need to verify and adjust the connector details that were set earlier for relationships.



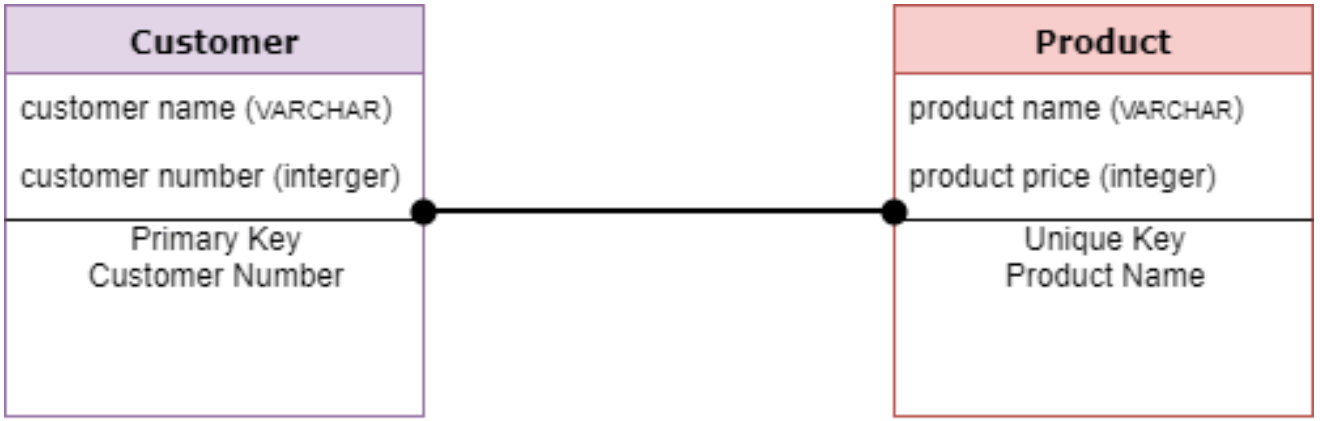
Logical Data Model: Features

- Describes data needs for a single project but could integrate with other logical data models based on the scope of the project.
- Designed and developed independently from the DBMS.
- Data attributes will have datatypes with exact precisions and length.
- Normalization processes to the model is applied typically till 3NF.

Physical Data Model

- A Physical Data Model describes a database-specific implementation of the data model.
- It offers database abstraction and helps generate the schema. This is because of the richness of meta-data offered by a Physical Data Model.
- The physical data model also helps in visualizing database structure by replicating database column keys, constraints, indexes, triggers, and other RDBMS features.

Physical Data Model



Physical Data Model: Features

- The physical data model describes data need for a single project or application though it maybe integrated with other physical data models based on project scope.
- Data Model contains relationships between tables that which addresses cardinality and nullability of the relationships.
- Developed for a specific version of a DBMS, location, data storage or technology to be used in the project.
- Columns should have exact datatypes, lengths assigned and default values.
- Primary and Foreign keys, views, indexes, access profiles, and authorizations, etc. are defined.

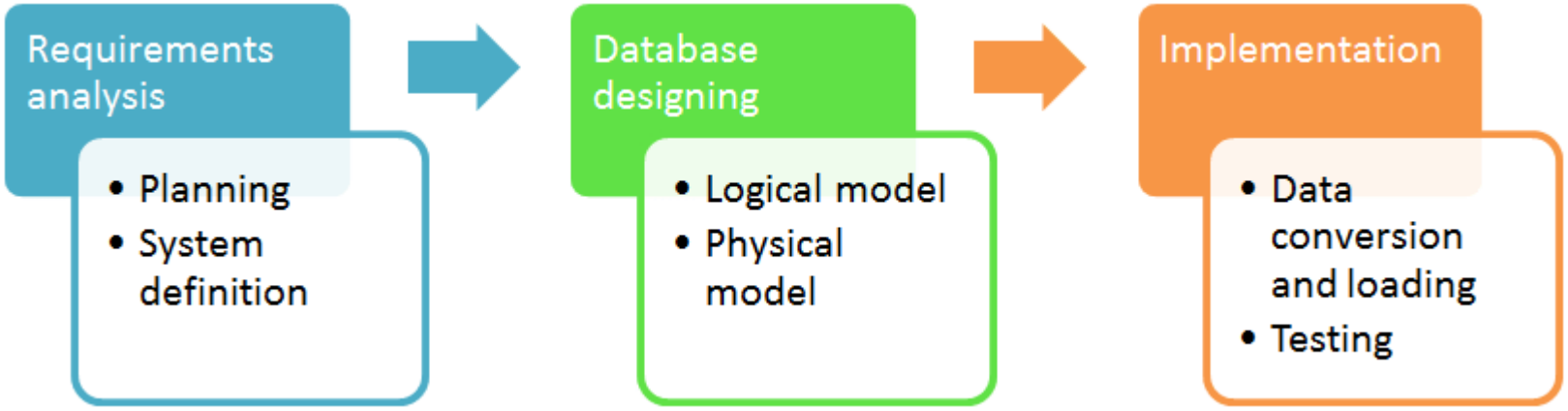
DBMS Design

- 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.

DBMS Design

- 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).

Database Development Life Cycle



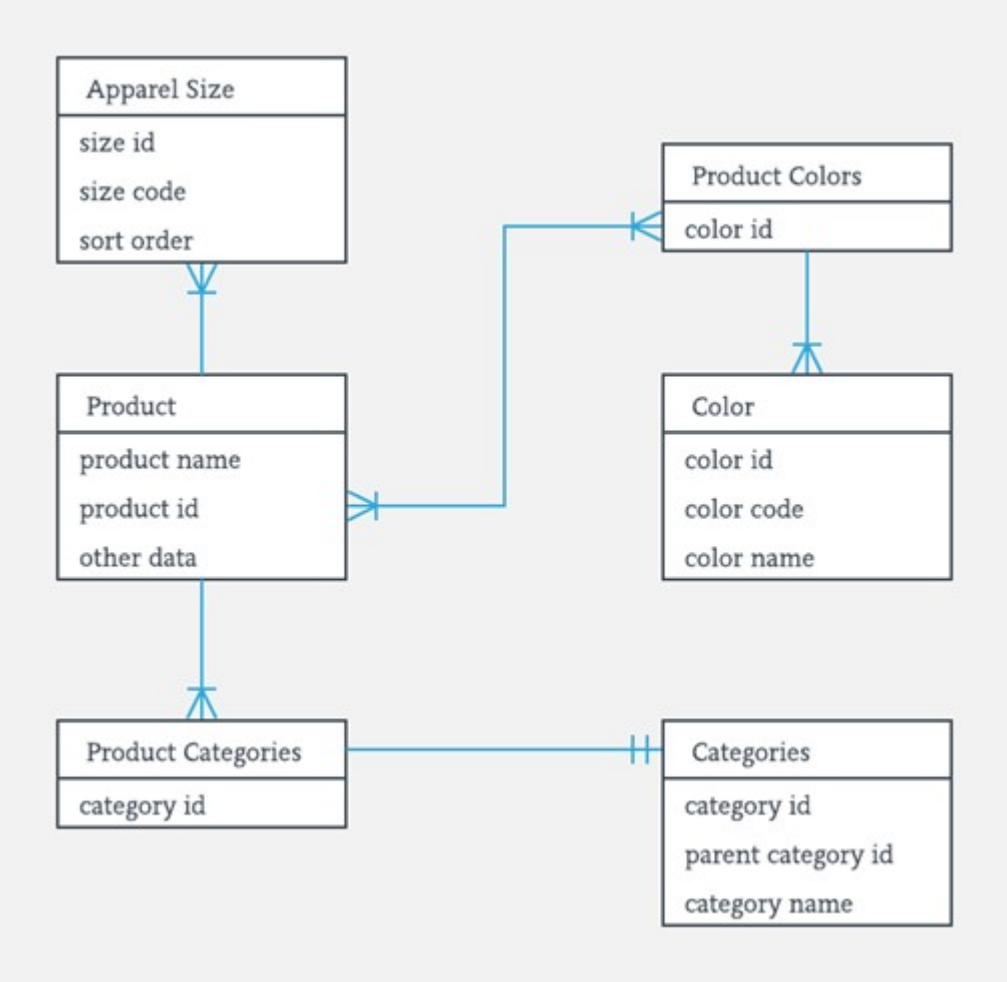
Database Design Techniques

- Two Types of Database Techniques
  - Normalization
  - ER Modeling

ER Diagram

- ER Diagram stands for Entity Relationship Diagram, also known as ERD is a diagram that displays the relationship of entity sets stored in a database.
- In other words, ER diagrams help to explain the logical structure of databases. ER diagrams are created based on three basic concepts: entities, attributes and relationships.
- ER Diagrams contain different symbols that use rectangles to represent entities, ovals to define attributes and diamond shapes to represent relationships.

ER Diagram


Apparel Size
size id
size code
sort order
Product Colors
color id
Product
product name
product id
other data
Color
color id
color code
color name
Product Categories
category id
Categories
category id
parent category id
category name

ER Diagram: Why?

- 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 Diagram allows you to communicate with the logical structure of the database to users

ER Diagram: Symbols and Notations

- Entity Relationship Diagram Symbols & Notations mainly contains three basic symbols which are rectangle, oval and diamond to represent relationships between elements, entities and attributes.
- There are some sub-elements which are based on main elements in ERD Diagram. ER Diagram is a visual representation of data that describes how data is related to each other using different ERD Symbols and Notations.

ER Diagram: Symbols and Notations

- Following are the main components and its symbols in ER Diagrams:
  - Rectangles: This Entity Relationship Diagram symbol represents entity types
  - Ellipses : Symbol represent attributes
  - Diamonds: This symbol represents relationship types
  - Lines: It links attributes to entity types and entity types with other relationship types
  - Primary key: attributes are underlined
  - Double Ellipses: Represent multi-valued attributes

ER Diagram: Symbols and Notations

ER Diagram: Symbols and Notations

Entity Name

Entity

Person,place,object,event or concept about which data is to be maintained

Example: Car, Student

Attribute

Property or characteristic of an entity

Example: Color of car Entity Name of Student Entity

Relation

Association between the instances of one or more entity types

Example: Blue Car Belongs to Student Jack

Entity

- A real-world thing either living or non-living that is easily recognizable and nonrecognizable. It is anything in the enterprise that is to be represented in our database.
- It may be a physical thing or simply a fact about the enterprise or an event that happens in the real world.
- An entity can be place, person, object, event or a concept, which stores data in the database.
- The characteristics of entities are must have an attribute, and a unique key. Every entity is made up of some ‘attributes’ which represent that entity.

Entity

- Examples of entities:
  - Person: Employee, Student, Patient
  - Place: Store, Building
  - Object: Machine, product, and Car
  - Event: Sale, Registration, Renewal
  - Concept: Account, Course

Entity Set

- Student
  - An entity set is a group of similar kind of entities. It may contain entities with attribute sharing similar values.
  - Entities are represented by their properties, which also called attributes. All attributes have their separate values.
  - For example, a student entity may have a name, age, class, as attributes.

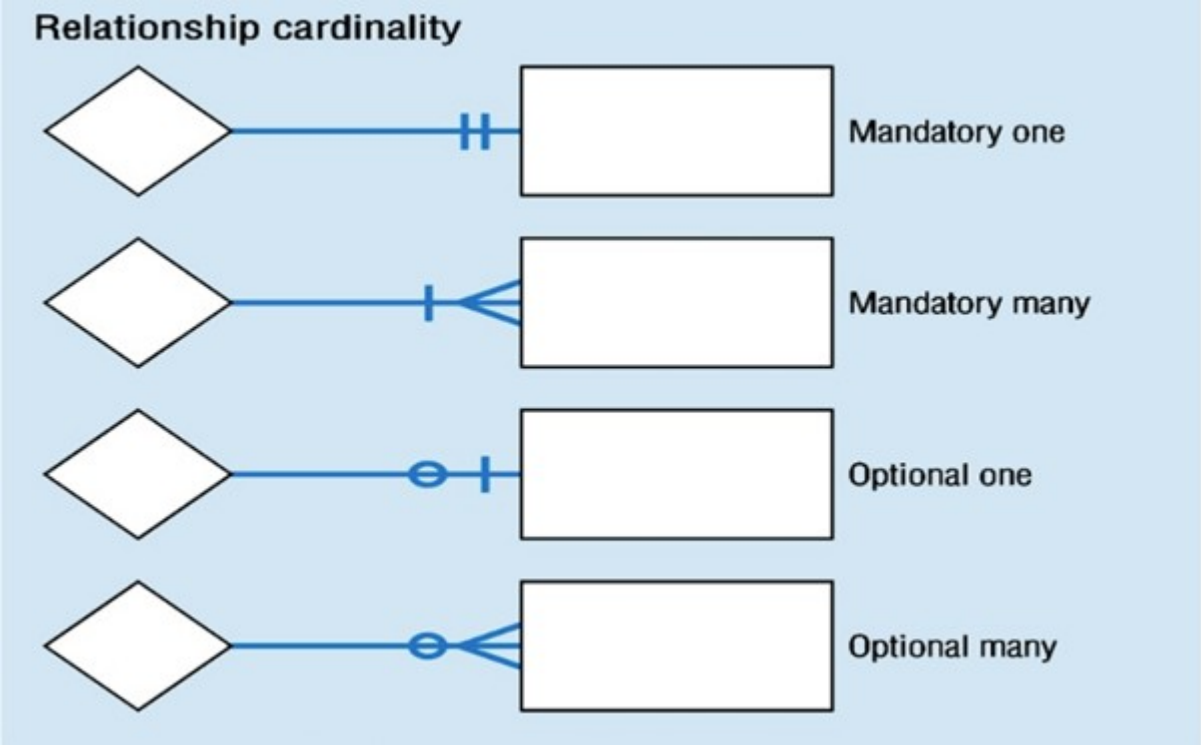
Relationships

- Relationship is nothing but an association among two or more entities. E.g., Tom works in the Chemistry department.
- Entities take part in relationships. We can often identify relationships with verbs or verb phrases.

Cardinality

- Defines the numerical attributes of the relationship between two entities or entity sets.
- Different types of cardinal relationships are:
  - One-to-One Relationships
  - One-to-Many Relationships
  - Many to One Relationships
  - Many-to-Many Relationships

Cardinality



Data redundancy

- In DBMS, when the same data is stored in different tables, it causes data redundancy.
- Sometimes, it is done on purpose for recovery or backup of data, faster access of data, or updating data easily.
- Redundant data costs extra money, demands higher storage capacity, and requires extra effort to keep all the files up to date.

Data redundancy

- Sometimes, unintentional duplicity of data causes a problem for the database to work properly, or it may become harder for the end user to access data.
- Redundant data unnecessarily occupy space in the database to save identical copies, which leads to space constraints, which is one of the major problems.

Data redundancy

Student_id	Name	Course	Session	Fee	Department
101	Devi	B. Tech	2022	90,000	CS
102	Sona	B. Tech	2022	90,000	CS
103	Varun	B. Tech	2022	90,000	CS
104	Satish	B. Tech	2022	90,000	CS
105	Amisha	B. Tech	2022	90,000	CS

Problems caused by redundancy

- Redundancy in DBMS gives rise to anomalies.
- In a database management system, the problems that occur while working on data include inserting, deleting, and updating data in the database.

student_id	student_name	student_age	dept_id	dept_name	dept_head
1	Shiva	19	104	Information Technology	Jaspreet Kaur
2	Khushi	18	102	Electronics	Avni Singh
3	Harsh	19	104	Information Technology	Jaspreet Kaur

Insertion Anomaly

- Insertion anomaly arises when you are trying to insert some data into the database, but you are not able to insert it.
- Example:
  - If you want to add the details of the student in the above table, then you must know the details of the department; otherwise, you will not be able to add the details because student details are dependent on department details.

Deletion Anomaly

- Deletion anomaly arises when you delete some data from the database, but some unrelated data is also deleted; that is, there will be a loss of data due to deletion anomaly.
- Example:
  - If we want to delete the student detail, which has student_id 2, we will also lose the unrelated data, i.e., department_id 102, from the above table.

Updation Anomaly

- An update anomaly arises when you update some data in the database, but the data is partially updated, which causes data inconsistency.
- Example:
  - If we want to update the details of dept_head from Jaspreet Kaur to Ankit Goyal for Dept_id 104, then we have to update it everywhere else; otherwise, the data will get partially updated, which causes data inconsistency.

Redundancy: Advantages

- Provides Data Security:
  - Data redundancy can enhance data security as it is difficult for cyber attackers to attack data that are in different locations.
- Provides Data Reliability:
  - Reliable data improves accuracy because organizations can check and confirm whether data is correct.
- Create Data Backup:
  - Data redundancy helps in backing up the data.

Redundancy: Disadvantages

- Data corruption:
  - Redundant data leads to high chances of data corruption.
- Wastage of storage:
  - Redundant data requires more space, leading to a need for more storage space.
- High cost:
  - Large storage is required to store and maintain redundant data, which is costly.

How to reduce data redundancy?

- Database Normalization: We can normalize the data using the normalization method. In this method, the data is broken down into pieces, which means a large table is divided into two or more small tables to remove redundancy. Normalization removes insert anomaly, update anomaly, and delete anomaly.
- Deleting Unused Data: It is important to remove redundant data from the database as it generates data redundancy in the DBMS. It is a good practice to remove unwanted data to reduce redundancy.

How to reduce data redundancy?

- Master Data:
  - The data administrator shares master data across multiple systems. Although it does not remove data redundancy, but it updates the redundant data whenever the data is changed.

Data Anomaly

- Anomaly means inconsistency in the pattern from the normal form. In Database Management System (DBMS), anomaly means the inconsistency occurred in the relational table during the operations performed on the relational table.
- There can be various reasons for anomalies to occur in the database.
  - For example, if there is a lot of redundant data present in our database then DBMS anomalies can occur. If a table is constructed in a very poor manner then there is a chance of database anomaly. Due to database anomalies, the integrity of the database suffers.

Data Anomaly

Worker_id	Worker_name	Worker_dept	Worker_address
65	Ramesh	ECT001	Jaipur
65	Ramesh	ECT002	Jaipur
73	Amit	ECT002	Delhi
76	Vikas	ECT501	Pune
76	Vikas	ECT502	Pune
79	Rajesh	ECT669	Mumbai

Updation / Update Anomaly

- When we update some rows in the table, and if it leads to the inconsistency of the table then this anomaly occurs.
- This type of anomaly is known as an updation anomaly. In the above table, if we want to update the address of Ramesh then we will have to update all the rows where Ramesh is present.
- If during the update we miss any single row, then there will be two addresses of Ramesh, which will lead to inconsistent and wrong databases.

Insertion Anomaly

- If there is a new row inserted in the table and it creates the inconsistency in the table then it is called the insertion anomaly.
- For example, if in the above table, we create a new row of a worker, and if it is not allocated to any department then we cannot insert it in the table so, it will create an insertion anomaly.

Deletion Anomaly

- If we delete some rows from the table and if any other information or data which is required is also deleted from the database, this is called the deletion anomaly in the database.
- For example, in the above table, if we want to delete the department number ECT669 then the details of Rajesh will also be deleted since Rajesh's details are dependent on the row of ECT669. So, there will be deletion anomalies in the table.

Example:

Stu_id	Stu_name	Stu_branch	Stu_club
2018nk01	Shivani	Computer science	literature
2018nk01	Shivani	Computer science	dancing
2018nk02	Ayush	Electronics	Videography
2018nk03	Mansi	Electrical	dancing
2018nk03	Mansi	Electrical	singing
2018nk04	Gopal	Mechanical	Photography

Updation Anomaly

- In the above table, if Shivani changes her branch from Computer Science to Electronics, then we will have to update all the rows.
- If we miss any row, then Shivani will have more than one branch, which will create the update anomaly in the table.

Insertion Anomaly

- If we add a new row for student Ankit who is not a part of any club, we cannot insert the row into the table as we cannot insert null in the column of stu_club. This is called insertion anomaly.

Deletion Anomaly

- If we remove the photography club from the college, then we will have to delete its row from the table.
- But it will also delete the table of Gopal and his details. So, this is called deletion anomaly and it will make the database inconsistent.

Functional Dependency

- The functional dependency is a relationship that exists between two attributes.
- It typically exists between the primary key and non-key attribute within a table.

  $X \rightarrow Y$

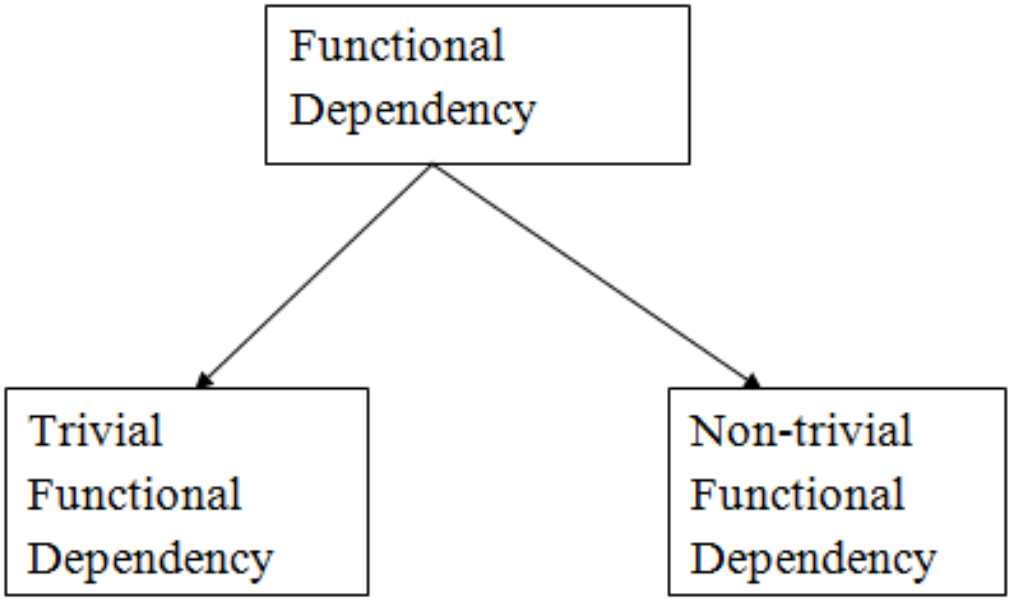
- The left side of FD is known as a determinant, the right side of the production is known as a dependent.

Functional Dependency

- *For example:*
- Assume we have an employee table with attributes: Emp_Id, Emp_Name, Emp_Address.
- Here Emp_Id attribute can uniquely identify the Emp_Name attribute of employee table because if we know the Emp_Id, we can tell that employee name associated with it.
- Functional dependency can be written as:

  Emp_Id → Emp_Name
- We can say that Emp_Name is functionally dependent on Emp_Id.

Functional Dependency



Trivial Functional Dependency

- A → B has trivial functional dependency if B is a subset of A.
- The following dependencies are also trivial like: A → A, B → B
- Example:
  - Consider a table with two columns Employee_Id and Employee_Name.
  - {Employee_id, Employee_Name} → Employee_Id is a trivial functional dependency as
  - Employee_Id is a subset of {Employee_Id, Employee_Name}.
  - Also, Employee_Id → Employee_Id and Employee_Name → Employee_Name are trivial dependencies too.

Trivial Functional Dependency

- A → B has a non-trivial functional dependency if B is not a subset of A.
- When A intersection B is NULL, then A → B is called as complete non-trivial.
- Example:
  - ID → Name,
  - Name → DOB

Normalization

- A large database defined as a single relation may result in data duplication. This repetition of data may result in:
  - Making relations very large.
  - It isn't easy to maintain and update data as it would involve searching many records in relation.
- Wastage and poor utilization of disk space and resources.
- The likelihood of errors and inconsistencies increases.

Normalization

- Normalization is the process of organizing the data in the database.
- Normalization is used to minimize the redundancy from a relation or set of relations.
- It is also used to eliminate undesirable characteristics like Insertion, Update, and Deletion Anomalies.
- Normalization divides the larger table into smaller and links them using relationships.
- The normal form is used to reduce redundancy from the database table.

Normalization: Why?

- The main reason for normalizing the relations is removing these anomalies.
- Failure to eliminate anomalies leads to data redundancy and can cause data integrity and other problems as the database grows.
- Normalization consists of a series of guidelines that helps to guide you in creating a good database structure.

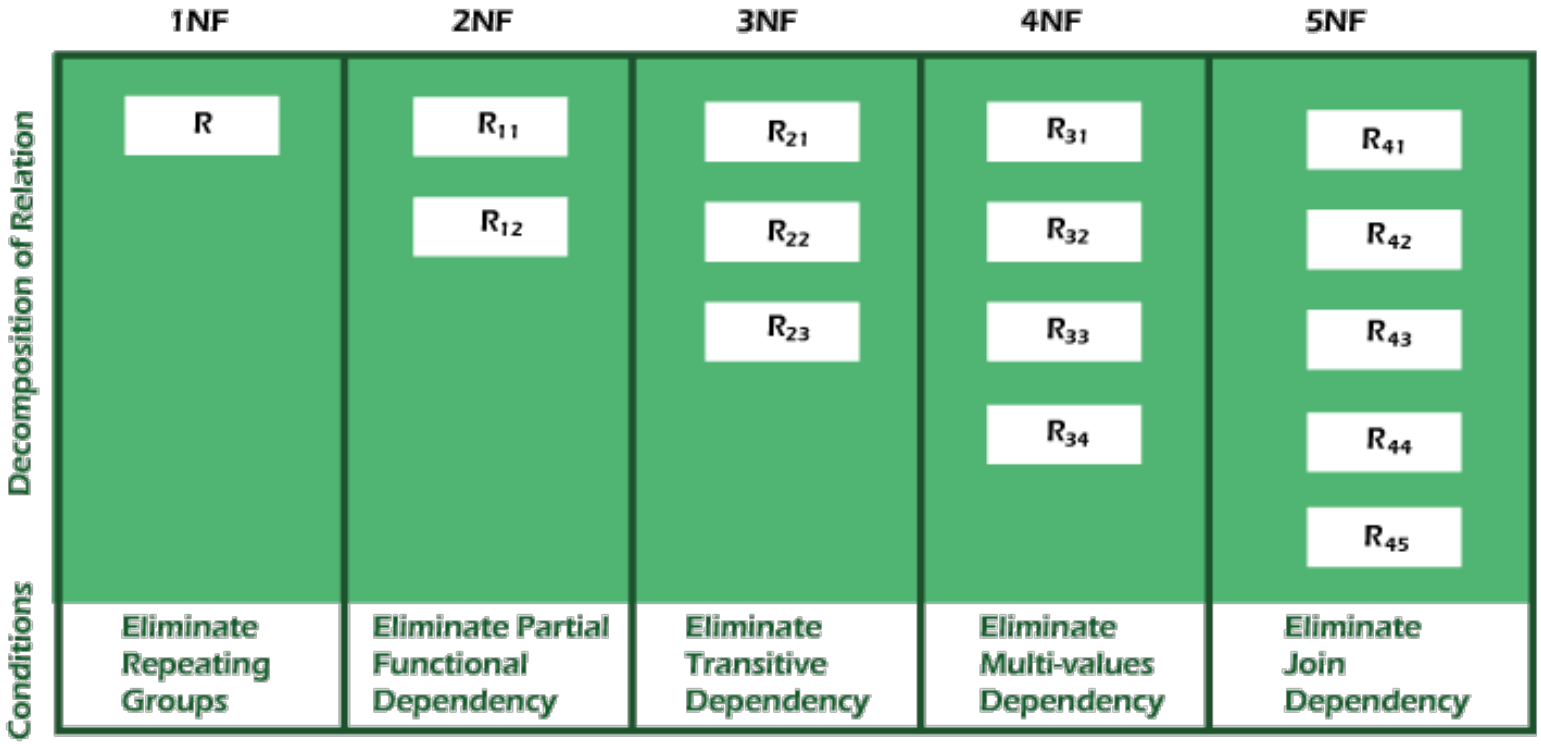
Normalization

- Data modification anomalies can be categorized into three types:
  - Insertion Anomaly: Insertion Anomaly refers to when one cannot insert a new tuple into a relationship due to lack of data.
  - Deletion Anomaly: The delete anomaly refers to the situation where the deletion of data results in the unintended loss of some other important data.
  - Updation Anomaly: The update anomaly is when an update of a single data value requires multiple rows of data to be updated.

Normal Forms

- Normalization works through a series of stages called Normal forms.
- The normal forms apply to individual relations. The relation is said to be in particular normal form if it satisfies constraints.

Normal Forms



Normal Forms

- 1NF
  - A relation is in 1NF if it contains an atomic value.
- 2NF
  - A relation will be in 2NF if it is in 1NF and all non-key attributes are fully functional dependent on the primary key.
- 3NF
  - A relation will be in 3NF if it is in 2NF and no transition dependency exists.
- BCNF
  - A stronger definition of 3NF is known as Boyce Codd's normal form.

Normal Forms

- 4NF
  - A relation will be in 4NF if it is in Boyce Codd's normal form and has no multi-valued dependency.
- 5NF
  - A relation is in 5NF. If it is in 4NF and does not contain any join dependency, joining should be lossless.

Advantages of Normalization

- Normalization helps to minimize data redundancy.
- Greater overall database organization.
- Data consistency within the database.
- Much more flexible database design.
- Enforces the concept of relational integrity.

Disadvantages of Normalization

- You cannot start building the database before knowing what the user needs.
- The performance degrades when normalizing the relations to higher normal forms, i.e., 4NF, 5NF.
- It is very time-consuming and difficult to normalize relations of a higher degree.
- Careless decomposition may lead to a bad database design, leading to serious problems.

1NF

- A relation will be 1NF if it contains an atomic value.
- It states that an attribute of a table cannot hold multiple values. It must hold only single-valued attribute.
- First normal form disallows the multi-valued attribute, composite attribute, and their combinations.

1NF

- Example: Relation EMPLOYEE is not in 1NF because of multi-valued attribute EMP_PHONE.

EMP_ID	EMP_NAME	EMP_PHONE	EMP_STATE
14	John	7272826385, 9064738238	UP
20	Harry	8574783832	Bihar
12	Sam	7390372389, 8589830302	Punjab

1NF

- The decomposition of the EMPLOYEE table into 1NF has been shown below:

EMP_ID	EMP_NAME	EMP_PHONE	EMP_STATE
14	John	7272826385	UP
14	John	9064738238	UP
20	Harry	8574783832	Bihar
12	Sam	7390372389	Punjab
12	Sam	8589830302	Punjab

2NF

- In the 2NF, relational must be in 1NF.
- In the second normal form, all non-key attributes are fully functional dependent on the primary key

2NF

- 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_ID	SUBJECT	TEACHER_AGE
25	Chemistry	30
25	Biology	30
47	English	35
83	Math	38
83	Computer	38

2NF

- 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_ID	TEACHER_AGE
25	30
47	35
83	38

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 at least one of the following conditions for every non-trivial function dependency X → Y.
  - X is a super key.
  - Y is a prime attribute, i.e., each element of Y is part of some candidate key.

3NF

- Super key in the table above:
- {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.

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

3NF

- 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.

3NF

EMP_ID	EMP_NAME	EMP_ZIP
222	Harry	201010
333	Stephan	02228
444	Lan	60007
555	Katharine	06389
666	John	462007

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

BCNF

- BCNF is the advance version of 3NF. It is stricter than 3NF.
- A table is in BCNF if every functional dependency X → Y, X is the super key of the table.
- For BCNF, the table should be in 3NF, and for every FD, LHS is super key.

BCNF

- Example: Let's assume there is a company where employees work in more than one department.

EMP_ID	EMP_COUNTRY	EMP_DEPT	DEPT_TYPE	EMP_DEPT_NO
264	India	Designing	D394	283
264	India	Testing	D394	300
364	UK	Stores	D283	232
364	UK	Developing	D283	549

- In the above table Functional dependencies are as follows:

  EMP_ID → EMP_COUNTRY

  EMP_DEPT → {DEPT_TYPE, EMP_DEPT_NO}

- Candidate key: {EMP-ID, EMP-DEPT}

BCNF

- The table is not in BCNF because neither EMP_DEPT nor EMP_ID alone are keys.
- To convert the given table into BCNF, we decompose it into three tables:
- EMP_COUNTRY table:

EMP_ID	EMP_COUNTRY
264	India
264	India

BCNF

EMP_DEPT	DEPT_TYPE	EMP_DEPT_NO
Designing	D394	283
Testing	D394	300
Stores	D283	232
Developing	D283	549

EMP_ID	EMP_DEPT
D394	283
D394	300
D283	232
D283	549

BCNF

- Functional dependencies:

  EMP_ID → EMP_COUNTRY

  EMP_DEPT → {DEPT_TYPE, EMP_DEPT_NO}
- Candidate keys:
  - For the first table: EMP_ID
  - For the second table: EMP_DEPT
  - For the third table: {EMP_ID, EMP_DEPT}

4NF

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

4NF

STU_ID	COURSE	HOBBY
21	Computer	Dancing
21	Math	Singing
34	Chemistry	Dancing
74	Biology	Cricket
59	Physics	Hockey

4NF

- The given STUDENT table is in 3NF, but the COURSE and HOBBY are two independent entity. Hence, there is no relationship between COURSE and HOBBY.
- In the STUDENT relation, a student with STU_ID, 21 contains two courses, Computer and Math and two hobbies, Dancing and Singing.
- So there is a Multi-valued dependency on STU_ID, which leads to unnecessary repetition of data.
- So to make the above table into 4NF, we can decompose it into two tables:

4NF

STU_ID	COURSE
21	Computer
21	Math
34	Chemistry
74	Biology
59	Physics

STU_ID	HOBBY
21	Dancing
21	Singing
34	Dancing
74	Cricket
59	Hockey

5NF

- A relation is in 5NF if it is in 4NF and not contains any join dependency and joining should be lossless.
- 5NF is satisfied when all the tables are broken into as many tables as possible in order to avoid redundancy.
- 5NF is also known as Project-join normal form (PJ/NF).

5NF

SUBJECT	LECTURER	SEMESTER
Computer	Anshika	Semester 1
Computer	John	Semester 1
Math	John	Semester 1
Math	Akash	Semester 2
Chemistry	Praveen	Semester 1

5NF

- In the above table, John takes both Computer and Math class for Semester 1 but he doesn't take Math class for Semester 2. In this case, combination of all these fields required to identify a valid data.
- Suppose we add a new Semester as Semester 3 but do not know about the subject and who will be taking that subject so we leave Lecturer and Subject as NULL. But all three columns together acts as a primary key, so we can't leave other two columns blank.
- So to make the above table into 5NF, we can decompose it into three relations P1, P2 & P3:

5NF

SEMESTER	SUBJECT
Semester 1	Computer
Semester 1	Math
Semester 1	Chemistry
Semester 2	Math

SUBJECT	LECTURER
Computer	Anshika
Computer	John
Math	John
Math	Akash
Chemistry	Praveen

SEMSTER	LECTURER
Semester 1	Anshika
Semester 1	John
Semester 1	John
Semester 2	Akash
Semester 1	Praveen

MySQL Programs

- mysql
  - The command-line tool for interactively entering SQL statements or executing them from a file in batch mode.
- mysqladmin
  - A client that performs administrative operations, such as creating or dropping databases, reloading the grant tables, flushing tables to disk, and reopening log files. mysqladmin can also be used to retrieve version, process, and status information from the server.
- mysqlcheck
  - A table-maintenance client that checks, repairs, analyzes, and optimizes tables.

MySQL Programs

- mysqldump
  - A client that dumps a MySQL database into a file as SQL, text, or XML.
- mysqlimport
  - A client that imports text files into their respective tables using LOAD DATA.
- mysqlpump
  - A client that dumps a MySQL database into a file as SQL.

MySQL Programs

- mysqlsh
  - MySQL Shell is an advanced client and code editor for MySQL Server. See MySQL Shell 8.0. In addition to the provided SQL functionality, similar to mysql, MySQL Shell provides scripting capabilities for JavaScript and Python and includes APIs for working with MySQL. X DevAPI enables you to work with both relational and document data
- mysqlshow
  - A client that displays information about databases, tables, columns, and indexes.
- mysqlslap
  - A client that is designed to emulate client load for a MySQL server and report the timing of each stage. It works as if multiple clients are accessing the server.

Running a MySQL script

- mysql -u root -p college < test.sql

Stored Procedure

- A procedure (often called a stored procedure) is a collection of pre-compiled SQL statements stored inside the database.
- It is a subroutine or a subprogram in the regular computing language. A procedure always contains a name, parameter lists, and SQL statements.
- We can invoke the procedures by using triggers, other procedures and applications such as Java, Python, PHP, etc.
- It was first introduced in MySQL version 5. Presently, it can be supported by almost all relational database systems.

Stored Procedure

- If we consider the enterprise application, we always need to perform specific tasks such as database cleanup, processing payroll, and many more on the database regularly. Such tasks involve multiple SQL statements for executing each task.
- This process might easy if we group these tasks into a single task. We can fulfill this requirement in MySQL by creating a stored procedure in our database.
- A procedure is called a recursive stored procedure when it calls itself. Most database systems support recursive stored procedures.

Stored Procedure: Features

- Stored Procedure increases the performance of the applications. Once stored procedures are created, they are compiled and stored in the database.
- Stored procedure reduces the traffic between application and database server. Because the application has to send only the stored procedure's name and parameters instead of sending multiple SQL statements.

Stored Procedure: Features

- Stored procedures are reusable and transparent to any applications.
- A procedure is always secure. The database administrator can grant permissions to applications that access stored procedures in the database without giving any permissions on the database tables.

Syntax:

- The following syntax is used for creating a stored procedure in MySQL.
- It can return one or more value through parameters or sometimes may not return at all.
- By default, a procedure is associated with our current database. But we can also create it into another database from the current database by specifying the name as database_name.procedure_name.

Syntax:

```
DELIMITER &&
CREATE PROCEDURE get_data()
BEGIN
   SELECT * FROM student WHERE marks > 70;
   SELECT COUNT(id) AS Total_Student FROM student;
END &&
DELIMITER ;
```

How to call procedure?

- We can use the CALL statement to call a stored procedure.
- This statement returns the values to its caller through its parameters (IN, OUT, or INOUT).
- The following syntax is used to call the stored procedure in MySQL:

```
CALL procedure_name ( parameter(s))
```

IN Parameter

- A stored procedures and functions may have input, output, and input/output parameters.
- Input parameter is a parameter whose value is passed into a stored procedure/function module.
- The value of an IN parameter is a constant; it can't be changed or reassigned within the module.

IN Parameter

```
DELIMITER &&
CREATE PROCEDURE get_data(IN var1 INT)
BEGIN
   SELECT * FROM student WHERE marks > 70 limit var1;
   SELECT COUNT(id) AS Total_Student FROM student;
END &&
DELIMITER ;
```

OUT Parameter

- Output parameter is a parameter whose value is passed out of the stored procedure/function module, back to the calling PL/SQL block.
- An OUT parameter must be a variable, not a constant. It can be found only on the left-hand side of an assignment in the module.
- You cannot assign a default value to an OUT parameter outside of the module's body.
- In other words, an OUT parameter behaves like an uninitialized variable.

OUT Parameter

```
DELIMITER &&
CREATE PROCEDURE get_data(OUT top INT)
BEGIN
   SELECT MAX(marks) INTO top FROM student;
END &&
DELIMITER ;
```

OUT Parameter

- `mysql> call get_data(@top);`

- `mysql> select @top;`

INOUT Parameter

- An input/output parameter is a parameter that functions as an IN or an OUT parameter or both.
- The value of the IN/OUT parameter is passed into the stored procedure/function and a new value can be assigned to the parameter and passed out of the module.
- An IN/OUT parameter must be a variable, not a constant. However, it can be found on both sides of an assignment.
- In other words, an IN/OUT parameter behaves like an initialized variable.

INOUT Parameter

```
DELIMITER &&
CREATE PROCEDURE get_data(INOUT var1 INT)
BEGIN
   SELECT marks INTO var1 FROM student where id = var1;
END &&
DELIMITER ;
```

INOUT Parameter

- `mysql> set @a = 3;`
- `mysql> call get_data(@a);`
- `mysql> select @a;`

Display the procedures

- When we have several procedures in the MySQL server, it is very important to list all procedures. It is because sometimes the procedure names are the same in many databases. We can list all procedure stored on the current MySQL server as follows:

```
SHOW PROCEDURE STATUS [LIKE 'pattern' |
        WHERE search_condition]
```

- This statement displays all stored procedure names, including their characteristics. If we want to display procedures in a particular database, we need to use the WHERE clause.

Display the procedures

- `SHOW PROCEDURE STATUS WHERE db = 'college';`

Drop procedure

- MySQL also allows a command to drop the procedure.
- When the procedure is dropped, it is removed from the database server also.
- The following statement is used to drop a stored procedure in MySQL:

```
DROP PROCEDURE [IF EXISTS ]
procedure_name;
```

Benefits of procedures

- Reduce the Network Traffic:
  - Multiple SQL Statements are encapsulated in a stored procedure. When you execute it, instead of sending multiple queries, we are sending only the name and the parameters of the stored procedure
- Easy to maintain:
  - The stored procedure are reusable. We can implement the business logic within an SP, and it can be used by applications multiple times, or different modules of an application can use the same procedure.
  - This way, a stored procedure makes the database more consistent. If any change is required, you need to make a change in the stored procedure only.

Benefits of procedures

- Secure:
  - The stored procedures are more secure than the AdHoc queries.
  - The permission can be granted to the user to execute the stored procedure without giving permission to the tables used in the stored procedure.
  - The stored procedure helps to prevent the database from SQL Injection

Drawbacks of procedures

- If we use stored procedures, the memory uses of every connection that uses those stored procedures will increase substantially.
  - Also, if we overuse many logical applications inside stored procedures, the CPU usage will increase. It is because the database server is not well designed for logical operations.
- Stored procedure's constructs are not designed to develop complex and flexible business logic.

Drawbacks of procedures

- It is difficult to debug stored procedures.
  - Only a few database management systems allow us to debug stored procedures. Unfortunately, MySQL does not provide facilities for debugging stored procedures.
- It is not easy to develop and maintain stored procedures.
  - Developing and maintaining stored procedures are often required a specialized skill set that not all application developers possess.
  - It may lead to problems in both application development and maintenance phases.

Gathering Data in Systematic fashion

- Data collection is a systematic process of gathering observations or measurements.
- Whether you are performing research for business, governmental or academic purposes, data collection allows you to gain first-hand knowledge and original insights into your research problem.

Gathering Data in Systematic fashion

- While methods and aims may differ between fields, the overall process of data collection remains largely the same.
- Before you begin collecting data, you need to consider:
  - The aim of the research
  - The type of data that you will collect
  - The methods and procedures you will use to collect, store, and process the data

Gathering Data in Systematic fashion

- Step 1: Define the aim of your research
  - Before you start the process of data collection, you need to identify exactly what you want to achieve. You can start by writing a problem statement: what is the practical or scientific issue that you want to address and why does it matter?
  - Next, formulate one or more research questions that precisely define what you want to find out. Depending on your research questions, you might need to collect quantitative or qualitative data:

Gathering Data in Systematic fashion

- Quantitative data is expressed in numbers and graphs and is analyzed through statistical methods.
- Qualitative data is expressed in words and analyzed through interpretations and categorizations.

Gathering Data in Systematic fashion

- Examples of quantitative and qualitative research aims
- You are researching employee perceptions of their direct managers in a large organization.
  - Your first aim is to assess whether there are significant differences in perceptions of managers across different departments and office locations.
  - Your second aim is to gather meaningful feedback from employees to explore new ideas for how managers can improve.

Gathering Data in Systematic fashion

- Step 2: Choose your data collection method
  - Based on the data you want to collect, decide which method is best suited for your research.
  - Experimental research is primarily a quantitative method.
  - Interviews, focus groups, and ethnographies are qualitative methods.
  - Surveys, observations, archival research and secondary data collection can be quantitative or qualitative methods.

Gathering Data in Systematic fashion

Method	When to use	How to collect data
Experiment	To test a causal relationship.	Manipulate variables and measure their effects on others.
Survey	To understand the general characteristics or opinions of a group of people.	Distribute a list of questions to a sample online, in person or over-the-phone.
Interview/focus group	To gain an in-depth understanding of perceptions or opinions on a topic.	Verbally ask participants open-ended questions in individual interviews or focus group discussions.
Observation	To understand something in its natural setting.	Measure or survey a sample without trying to affect them.
Ethnography	To study the culture of a community or organization	Join and participate in a community and record your observations and reflections.

Gathering Data in Systematic fashion

Archival research	To understand current or historical events, conditions or practices.	Access manuscripts, documents or records from libraries, depositories or the internet.
Secondary data collection	To analyze data from populations that you can't access first-hand.	Find existing datasets that have already been collected, from sources such as government agencies or research organizations.

Gathering Data in Systematic fashion

- Step 3: Plan your data collection procedures
  - When you know which method(s) you are using, you need to plan exactly how you will implement them. What procedures will you follow to make accurate observations or measurements of the variables you are interested in?
  - For instance, if you're conducting surveys or interviews, decide what form the questions will take; if you're conducting an experiment, make decisions about your experimental design (e.g., determine inclusion and exclusion criteria).

Operationalization

- Sometimes your variables can be measured directly: for example, you can collect data on the average age of employees simply by asking for dates of birth.
- However, often you'll be interested in collecting data on more abstract concepts or variables that can't be directly observed.
- Operationalization means turning abstract conceptual ideas into measurable observations.
- When planning how you will collect data, you need to translate the conceptual definition of what you want to study into the operational definition of what you will actually measure.

Operationalization

- Example of operationalization
- You have decided to use surveys to collect quantitative data. The concept you want to measure is the leadership of managers. You operationalize this concept in two ways:
  - You ask managers to rate their own leadership skills on 5-point scales assessing the ability to delegate, decisiveness and dependability.
  - You ask their direct employees to provide anonymous feedback on the managers regarding the same topics.

Sampling

- You may need to develop a sampling plan to obtain data systematically.
- This involves defining a population, the group you want to draw conclusions about, and a sample, the group you will actually collect data from.
- Your sampling method will determine how you recruit participants or obtain measurements for your study.
- To decide on a sampling method you will need to consider factors like the required sample size, accessibility of the sample, and timeframe of the data collection.

Creating a data management plan

- Before beginning data collection, you should also decide how you will organize and store your data.
  - If you are collecting data from people, you will likely need to anonymize and safeguard the data to prevent leaks of sensitive information (e.g. names or identity numbers).
  - If you are collecting data via interviews or pencil-and-paper formats, you will need to perform transcriptions or data entry in systematic ways to minimize distortion.
  - You can prevent loss of data by having an organization system that is routinely backed up.

Data Collection

- Step 4: Collect the data
  - Finally, you can implement your chosen methods to measure or observe the variables you are interested in.
  - Examples of collecting qualitative and quantitative data
  - To collect data about perceptions of managers, you administer a survey with closed- and open-ended questions to a sample of 300 company employees across different departments and locations.
  - The closed-ended questions ask participants to rate their manager's leadership skills on scales from 1–5. The data produced is numerical and can be statistically analyzed for averages and patterns.

Data Collection

- To ensure that high quality data is recorded in a systematic way, here are some best practices:
  - Record all relevant information as and when you obtain data. For example, note down whether or how lab equipment is recalibrated during an experimental study.
  - Double-check manual data entry for errors.
  - If you collect quantitative data, you can assess the reliability and validity to get an indication of your data quality.

Thank you

@mitu_skillologies

@mITuSkillologies

@mitu_group

@mitu-skillologies

@MITUSkillologies

kaggle

@mituskillologies

Web Resources
https://mitu.co.in
http://tusharkute.com

@mituskillologies

contact@mitu.co.in

tushar@tusharkute.com