Contents

TERMS

Data – facts or figures stored in databases

Database – systematic collection of data, organized and structured in a meaningful way.

Relational database - data is stored in one or more tables

DBMS - software systems used to store, retrieve, and run queries on data.

RDBMS - program used to create, update, and manage relational databases.

Null – no value

Anomalies – errors within data integrity. E.g wrong update

Entity – anything (object type, person, concept, organization) about which information is stored.

Attributes – columns/ things that store data about an entity.

Relation – the connection between 2 sets of data

Tuple – all attributes about an entity.

Table – a collection of related data held in a **table** format within a **database** and consisting of columns and rows

Rows/record – stores entities in tables

Columns/field- stores attributes in tables

Key - an attribute/a set of attributes that help us identify a row (or tuple) uniquely in a table.

atomic value - an instance of one of the built-in atomic data types

DATA INTEGRITY

Overall accuracy, consistency, and completeness of data and compliance to regulatory data safety.

Types of data integrity

- Physical integrity: protection of the wholeness and accuracy of that data as it's stored and retrieved. When natural disasters strike, power goes out, or hackers disrupt database functions, physical integrity is compromised.
- Logical Integrity – It includes:
    - Entity integrity – Ensures data is not listed more than once and that no field is null. It relies on primary keys.
    - Referential integrity - processes that ensure data is used and stored uniformly. It eliminates the entry of inaccurate or duplicate data.
    - User-defined integrity – rules and constraints created by the user to fit particular needs
    - Doman integrity – ensures the accuracy of data in the domain. A domain is a set of values that a column is allowed to contain. It ensures correct data types, formats, and amounts are entered.

DATABASE RELATIONSHIPS

These are associations between tables. They include:

- One-to-one relationship – there's only one record on each side of the relationship. E.g. husband and wife relationship, user and their username,
- One-to-many relationship – the primary key table has one record that relates to many other records in related tables. E.g. parent and children (one parent has several children but one child can only have one parent)
- Many to many - each record in both tables can relate to any number of records in another table. This requires an association third table.

KEYS

These are attribute/s that help in identifying a row in a table. They enable the identification of relationships between two tables. They also help in finding unique records from a table.

Types of Keys in DBMS

- **Super Key –** A super key is a group of single or multiple keys which identifies rows in a table.

- **Primary Key –** a column or group of columns in a table that uniquely identifies every row in that table.
    - The Primary Key can't be a duplicate meaning the same value can't appear more than once in the table.
    - A table cannot also have more than one primary key.
    - Two rows can't have the same primary key value
    - It must be for every row to have a primary key value.
    - The primary key field cannot be null.
    - The value in a primary key column can never be modified or updated if any foreign key refers to that primary key.
- **Candidate Key –** a set of attributes that uniquely identify tuples in a table. Candidate Key is a super key with no repeated attributes.
    - The Primary key should be selected from the candidate keys. Every table must have at least a single candidate key. A table can have multiple candidate keys but only a single primary key.
    - It must contain unique values
    - Candidate key in SQL may have multiple attributes
    - Must not contain null values
    - It should contain minimum fields to ensure uniqueness
    - Uniquely identify each record in a table
- **Alternate Key –** a column or group of columns in a table that uniquely identifies every row in that table.
    - A table can have multiple choices for a primary key but only one can be set as the primary key. All the keys which are not primary keys are called Alternate keys.
- **Foreign Key –** a column that creates a relationship between two tables. The purpose of Foreign keys is to maintain data integrity and allow navigation between two different instances of an entity. It acts as a cross-reference between two tables as it references the primary key of another table.
- **Compound Key –** has two or more attributes that allow you to uniquely recognize a specific record. It is possible that each column may not be unique by itself within the database.. However, when combined with the other column or columns the combination of composite keys become unique. The purpose of the compound key in a database is to uniquely identify each record in the table.
- **Composite Key –** a combination of two or more columns that uniquely identify rows in a table. The combination of columns guarantees uniqueness, though individual uniqueness is not guaranteed. Hence, they are combined to uniquely identify records in a table. The

difference between the compound and the composite key is that any part of the compound key can be a foreign key, but the composite key may or maybe not be a part of the foreign key.

- **Surrogate Key –** An artificial key that aims to uniquely identify each record is called a surrogate key.
    - These kinds of keys are unique because they are created when you don't have any natural primary key.
    - They do not lend any meaning to the data in the table.
    - The surrogate key in DBMS is usually an integer.
    - A surrogate key is a value generated right before the record is inserted into
    - Surrogate keys in SQL are allowed when
        - No property has the parameter of the primary key.
        - In the table when the primary key is too big or complicated.

LOOKUP TABLE

It maps keys to values.

Its benefits include:

- can replace complex runtime calculations with simple lookup operations.
- maintain data integrity in our application.

ENTITY RELATIONSHIP DIAGRAM

An Entity Relationship (ER) Diagram is a type of flowchart that illustrates how "entities" such as people, objects, or concepts relate to each other within a system.

They are used for:

- Database design
- Database troubleshooting
- Business information systems
- Business process re-engineering (BPR)
- Education
- Research

Its features and components include:

Entity

Relationship

Attribute

CARDINALITY AND MODALITY

Modality

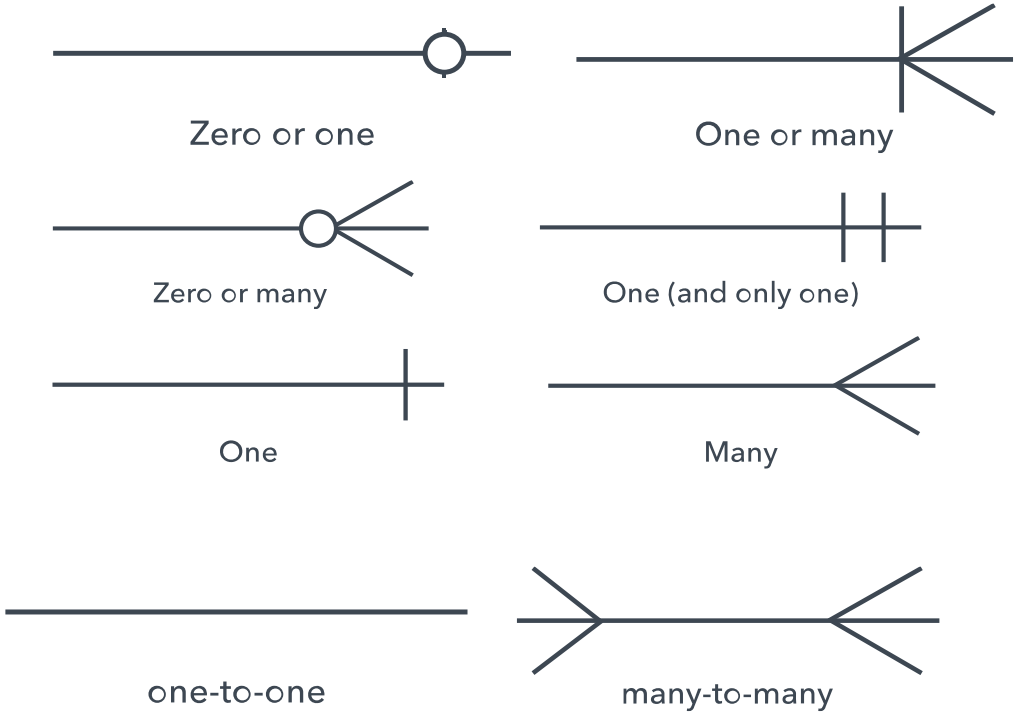
It describes whether a relationship between two or more entities is needed or not. Thus, in the case of modality, the modality can be classified into two types namely nullable modality and non−nullable modality. The value is computed as "**0**", if the relationship is optional or if there is no need for the relationship to occur. If there is a necessity for the occurrence of a relationship, then the value of the modality is computed as "**1**".

Cardinality

It describes the maximum number of data objects that can participate in a relationship. In databases, cardinality is defined as the uniqueness of data values that are contained in a column. High cardinality means the column contains a large percentage of totally unique values. On the other hand, low cardinality means the column has a lot of "**repeats**" in its data range.

It also defines the numerical attributes of the relationship between two entities or entity sets. The three main cardinal relationships are:

- A one-to-one. (a user with their email address)
- A one-to-many example (one student with many courses)
- Many-to-many example: (faculty members and multiple students)



one-to-many

The most basic difference between cardinality and modality is that cardinality is a metric that specifies the number of occurrences of a data object with respect to the number of occurrences of another data object, whereas, modality specifies whether a data object must participate in a relationship or not.

.

NORMALIZATION

- It is a database design technique that reduces data redundancy and eliminates undesirable characteristics like Insertion, Update, and Deletion Anomalies.
- Normalization rules divide larger tables into smaller tables and link them using relationships.
- The purpose of Normalisation in SQL is to eliminate redundant (repetitive) data and ensure data is stored logically.

The normal forms include:

- 1NF (First Normal Form)
    - Each table cell should contain a single value.
    - Each record needs to be unique.
- 2NF (Second Normal Form)
    - Rule 1- Be in 1NF
    - Rule 2- all non-key attributes are fully functionally dependent on the primary key.
- 3NF (Third Normal Form)
    - Rule 1- Be in 2NF
    - Rule 2- Has no transitive functional dependencies (changing a non-key column, might cause any of the other non-key columns to change)
- BCNF (Boyce-Codd Normal Form)
- 4NF (Fourth Normal Form)
    - If no database table instance contains two or more, independent and multivalued data describing the relevant entity, then it is in 4^{th} Normal Form.
- 5NF (Fifth Normal Form)
    - A table is in 5^{th} Normal Form only if it is in 4NF and it cannot be decomposed into any number of smaller tables without loss of data.

- 6NF (Sixth Normal Form)

Difference between 1NF and 2NF

	1NF	2NF
1.	In order to be in 1NF any relation must be atomic and should not contain any composite or multi-valued attributes.	In order to be in 2NF any relation must be in 1NF and should not contain any partial dependency.
2.	The identification of functional dependency is not necessary for the first normal form.	The identification of functional dependency is necessary for the second normal form.
3.	The first Normal form only deals with the schema of the table and it does not handle the update anomalies.	The second normal form handles the update anomalies.
4.	A relation in 1NF may or may not be in 2NF.	A relation in 2NF is always in 1NF.
5.	The primary key in the case of the first normal form can be a composite key.	The primary key in the case of the second normal form cannot be a composite key in case it arises any partial dependency.
6.	The main goal of the first normal form is to eliminate the redundant data within the table.	The main goal of the second normal form is to actually ensure the data dependencies.

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

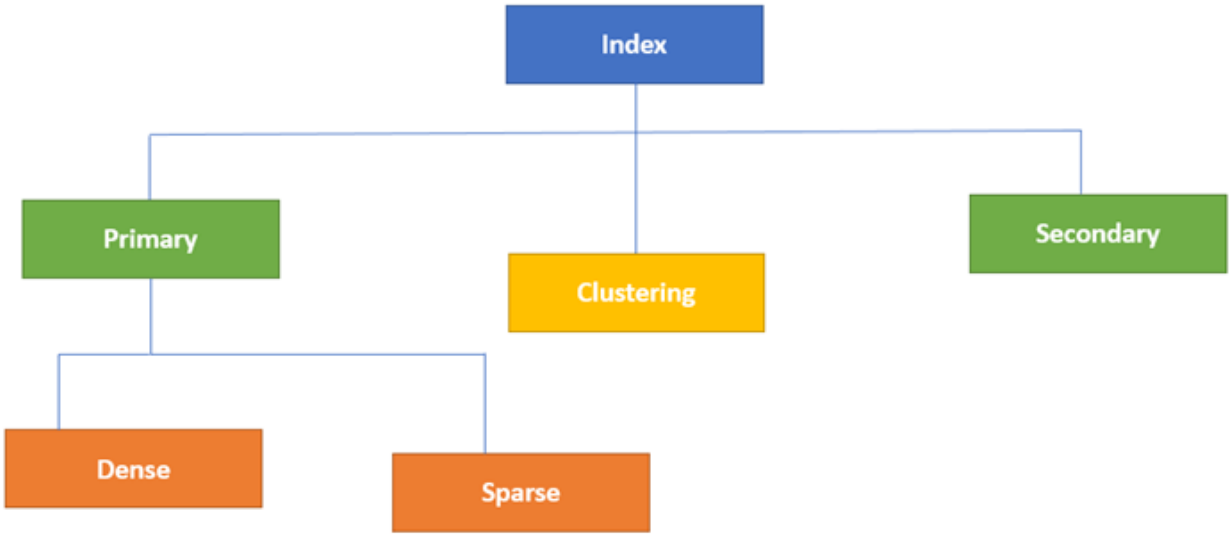
- You cannot start building the database before knowing what the user needs.
- The performance degrades when normalizing the relations to higher normal forms, i.e., 4NF, 5NF.
- It is very time-consuming and difficult to normalize relations of a higher degree.
- Careless decomposition may lead to a bad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

INDEXING

Indexing is a data structure technique that allows you to quickly retrieve records from a database file. An Index is a small table having only two columns. The first column comprises a copy of the primary or candidate key of a table. Its second column contains a set of pointers for holding the address of the disk block where that specific key value stored.

It:

- Takes a search key as input
- Efficiently returns a collection of matching records.



DATA TYPES

SQL data types can be broadly divided into the following categories.

1. Numeric data types such as:

- INT	- BIGINT	- REAL
- TINYINT	- FLOAT	- Decimal

- Bit
- Small int

2. Date and Time data types such as:

   - DATE - YYYY-MM-DD
   - TIME - HH:MI:SS
   - DATETIME - YYYY-MM-DD HH:MI:SS
   - Timestamp
   - Year

3. Character and String data types such as:

   - CHAR
   - VARCHAR
   - TEXT

4. Unicode character string data types such as:

   - NCHAR
   - NVARCHAR
   - NTEXT

5. Binary data types such as:

   - BINARY
   - VARBINARY
   - Image

6. Miscellaneous data types :

   - CLOB
   - LOB
   - XML
   - CURSOR
   - TABLE

JOINS

JOIN is an SQL clause used to query and access data from multiple tables, based on logical relationships between those tables.

In other words, JOINS indicate how SQL Server should use data from one table to select the rows from another table.

Types of joins:

- Inner join. creates a result table by combining rows that have matching values in two or more tables. (Retrieves data that appears in both tables).
- Outer join. It will retrieve not only the matching rows but also the unmatched rows as well.
- Full outer join. Returns a result that includes rows from both left and right tables.

- Left outer join. Includes in a result table unmatched rows from the table that is specified before the LEFT OUTER JOIN clause. (joins two or more tables and returns all rows from the left table and matched records from the right table or returns null if it does not find any matching record.)
- Right outer join. creates a result table and includes into it all the records from the right table and only matching rows from the left table. (joins two or more tables and returns all rows from the right-hand table, and only those results from the other table that fulfilled the specified join condition. If it finds unmatched records from the left side table, it returns Null value)
- Self join. joins the table to itself and allows comparing rows within the same table.
- Cross join. creates a result table containing paired combination of each row of the first table with each row of the second table.

ALIASES

- SQL Aliases are used to give a table, or a column in a table, a temporary name.
- Aliases are often used to make column names more readable.
- An alias only exists for the duration of that query.
- An alias is created with the AS keyword.