**DATABASE DESIGN**

Database design can be defined as a collection of steps geared towards creation, execution and maintenance of a business database management system.

A good design is very paramount. There are certain rules that must be followed when designing a database system for an organization.

How can we ensure we have a good database design? A well-designed database is one that:

* Data must be distributed into different tables on a specific subject area to decrease data redundancy.
* Deliver the database the information needed to link the data to the tables.
* Ensure uniqueness in every table. The uniqueness of records present in any relation is a very crucial part of database design that helps us avoid redundancy. Identify the key attributes to uniquely identify every row from columns. You can use various key constraints to ensure the uniqueness of your table, also keep in mind the uniquely identifying records must consume as little space as possible & shall not contain any NULL values.
* Caters to your information processing and reporting requirements.
* Functions interactively with the database operators.

**Importance of Database Design**

Database design helps to define the database structure used for planning, storing, and managing information. To ensure data accuracy, you must design a database that only stores relevant and valuable information.

A well-structured database is quite essential to guarantee information consistency, reduce data redundancy, efficient execution of queries and improves database’s performance.

Table structure determines the reliability of data, creating a primary and unique keys ensures uniformity in stored information. Data replication can be avoided by forming tables of probable values and using a key to denote those values.

A good performance and efficiency depends on a good design. A good database design uses simple queries and a much faster implementation.

**Basic Terminologies used in Database Design**

There are basic terminologies to familiarize with before designing a database:

**Redundancy:** Redundancy refers to the duplicity of the data. There can be specific use cases when we need or don’t need redundancy in our Database. For example: If we have a banking system application then we may need to strictly prevent redundancy in our Database.

**Schemas:** schemas refers to a logical container that defines the structure & manages the organization of the data stored in it. It consists of rows and columns having data types for each column.

**Record/Tuples:** A Record or tuples is basically where data is stored inside a table.

**Indexing:** Indexing is a data structure technique that promotes efficient retrieval of data stored in the database.

**Data Integrity and Consistency:** Data integrity refers to the quality of data stored in our database. Consistency is the correctness of data stored.

**Data Models:** Data models provide us with visual modeling techniques to visualize the data & the relationship that exists among those data. Ex: model, Network Model, Object Oriented Model, Hierarchical model, etc.

**Normalization:** The process of organizing data to reduce redundancy and dependency by dividing larger tables into smaller ones and defining relationships. It ensures data storage and consistency.

**Functional dependency:** This can be defined as a relationship between two attributes of the table that represents that the value of one attribute can be determined by another. Ex: {A -> B}, A & B are two attributes and attribute A can uniquely determine the value of B.

**Transaction:** Transaction can be defined as a single logical unit of work. It signifies some changes that are made in the database. A transaction must satisfy the ACID or BASE properties depending on the type of database.

**Schedule:** Schedule refers to the sequence of transactions which are executed by one or more users.

**Concurrency:** This means allowing multiple transactions to occur simultaneously without any form of interference.

**Constraints:** Constraints are the rules applied to fields in a table to enforce data integrity. e.g., NOT NULL, UNIQUE, CHECK, etc. It ensures data quality and accuracy.

**DATABASE DEVELOPMENT LIFE CYCLE**

There various stages involved in database development. Here we’ll be discussing on the three phases of data development: requirement analysis, database designing and implementation.

**1.Requirement analysis:**

Requirement analysis requires two steps:

**Planning:** In this stage of database development, the entire Database Development Life Cycle plan is decided. It also requires an analysis of the organization’s information systems strategy.

**Defining the system:** This stage explains the proposed database system’s scope.

**2.Database Designing:**

The actual database design takes in account two fundamental data model:

**Logical Data Model design:** Logical Data Model Design: This phase consists of coming up with a high-level design of our database based on initially gathered requirements to structure & organize our data accordingly. A high-level overview on paper is made of the database without considering the physical level design, this phase proceeds by identifying the kind of data to be stored and what relationship will exist among those data.  
Entity, Key attributes identification & what constraints are to be implemented is the core functionality of this phase.

**Physical model:** This stage comes after the logical model and therefore involves physically implementing the logical model. It takes the DBMS and other physical implementation factors into consideration.

**3. Implementation:**

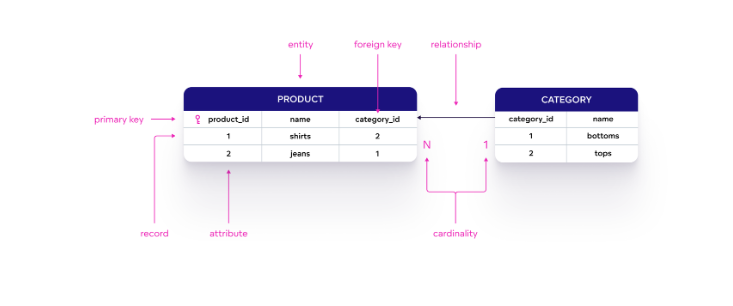
The implementation phase is concerned with:

* **Data conversion and loading** involve importing and converting data from the old system into the new database.
* **Testing:** Finally, this stage identifies errors in the new system and meets all the database requirement specifications.

**Logical Data Model Design**

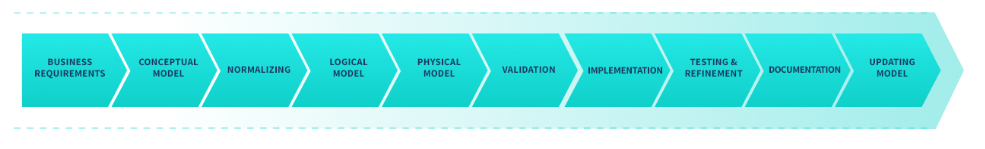
The logical data model design defines the structure of data and what relationship exists among those data. The following are the major components of the logical design:

**Data Models:** Data modeling is a visual modeling technique used to get a high-level overview of our database. Data models help us understand the needs and requirements of our database by defining the design of our database through diagrammatic representation. Ex: model, Network model, Relational Model, object-oriented data model.



**Data modeling process:**

Here are the 10 steps a data modeler or data architect typically takes during the modeling process:



1. Identify the Business Requirements: Make sure you understand the business requirements and the purpose of your data model. Identify the entities, attributes, and relationships that need to be represented in your model.
2. Create a Conceptual Model: As described above, here you create a high-level model that represents the entities, attributes, and relationships you just identified. Use standard notation such as entity-relationship (ER) diagrams to create the model. Data modeling  
   tools can help with creating your data model diagram.
3. Normalize Your Model: Normalize your model to eliminate data redundancy and improve data integrity. Use the normalization rules such as first normal form, second normal form, and third normal form to achieve this.
4. Create a Logical Model: Create a detailed data model that represents the entities, attributes, and relationships identified in step 1. As with the conceptual model, use standard notation such as ER diagrams to show how entities are connected.
5. Create a Physical Model: Create a data model that represents the physical storage of your data. This includes the database schema and the data types used to store your data.
6. Validate Your Model: Make sure your model meets the needs of the business and that it’s accurate and complete.
7. Implement Your Model: Implement your model in a database management system. Create the database schema and load the data into the database.
8. Test and Refine Your Model: Run queries and reports against your data and refine the model as needed to improve performance or fix errors.
9. Document Your Model: Document your model in a way that is easily understood by others. Include detailed descriptions of the entities, attributes, and relationships in the model.
10. Update Your Data Model: You’ll likely need to revise your model as your organization's business needs and data assets evolve.

## **Benefits:**

Data modeling allows stakeholders from across your company, such as business analysts, developers, and data architects, to understand the organization and relationships in your data. Other key benefits are that it will help you meet your business requirements, help you develop and implement a data architecture, and help ensure that your databases and apps are using the correct data. Here are additional benefits of data models:

* **Improved data organization:** They help to organize and structure data in a way that is logical and easy to understand, making it easier to access and use.
* Better data integrity. They help to ensure that data is accurate, consistent, and complete, reducing the risk of errors and inconsistencies.
* **Increased scalability:** They allow for the efficient management and scalability of large and complex data sets, making it easy to add or remove data as needed.
* Enhanced data security: They can help to identify and mitigate security risks, making it easier to protect sensitive data from unauthorized access.
* Better performance: They help to optimize data storage and retrieval, resulting in faster query performance and improved overall system performance.
* **Improved collaboration:** They can facilitate collaboration between different departments and teams, making it easier to share and access data across the organization and improve data literacy.
* **Better decision-making:** They can help break down data silos and ensure accurate data. Analysis of this data can provide valuable insights and information that can be used to inform strategic decision-making and business planning.

**Entities:** Entities are objects in the real world, which can have certain properties & these properties are referred to as attributes of that particular entity. There are 2 types of entities: Strong and weak entities, weak entities do not have a key attribute to identify them, their existence solely depends on one 1-specific strong entity & also have full participation in a relationship whereas strong entity does have a key attribute to uniquely identify them.

**Relationships:** How data is logically related to each other defines the relationship of that data with other entities. In simple words, the association of one entity with another is defined here.

A relationship can be further categorized into – unary, binary, and ternary relationships.

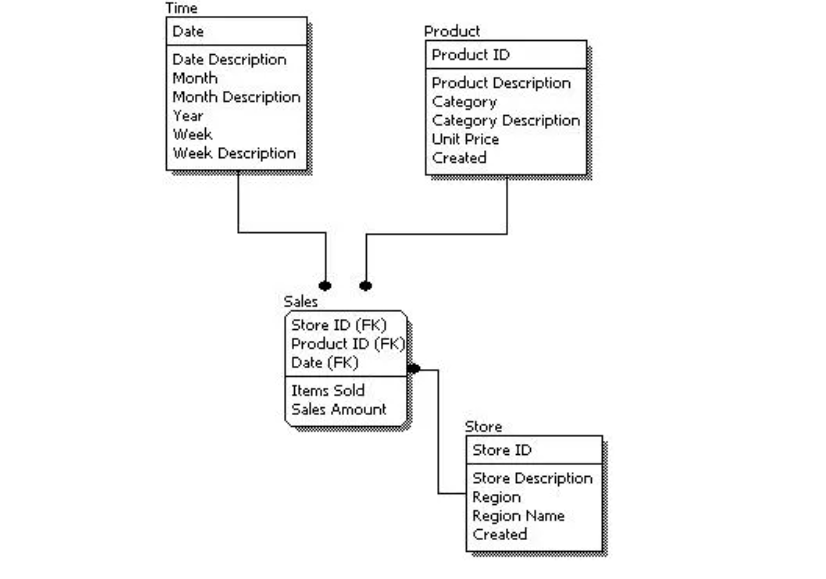
* Unary: In this, the associating entity & the associated entity both are the same. Ex: Employee Manages themselves, and students are also given the post of monitor hence here the student themselves is a monitor.
* Binary: This is a very common relationship that you will come across while designing a database.  
  Example: Students are enrolled in courses, Employees are managed by different managers, One student can be taught by many professors.
* Ternary: In this, we have 3 entities involved in a single relationship. Ex: an employee works on a project for a client. Note that, here we have 3 entities: Employee, Project & Client.

**Attributes:** Attributes are nothing but properties of a specific entity that define its behavior. For example, an employee can have unique\_id, name, age, date of birth (DOB), salary, department, Manager, project id, etc.

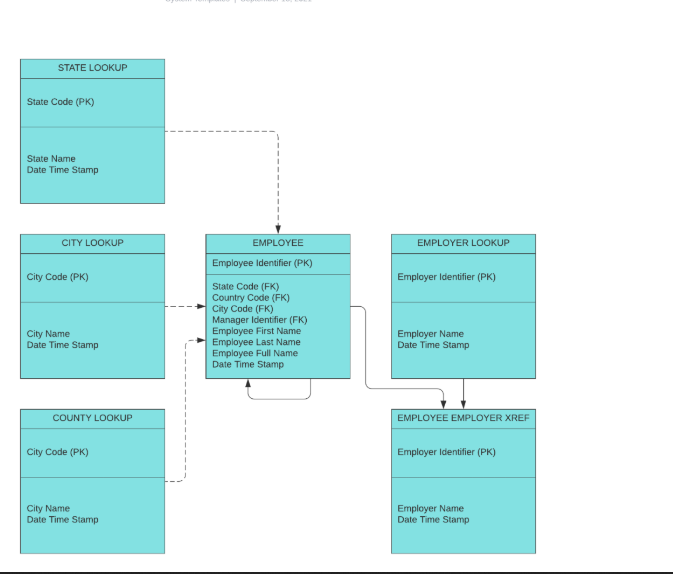
**Normalization:** Normalization: After all the entities are put in place and the relationship among data is defined, we need to look for loopholes or possible ambiguities that may arise as a result of CRUD operations. To prevent various Anomalies such as INSERTION, UPDATION, and DELETION Anomalies.

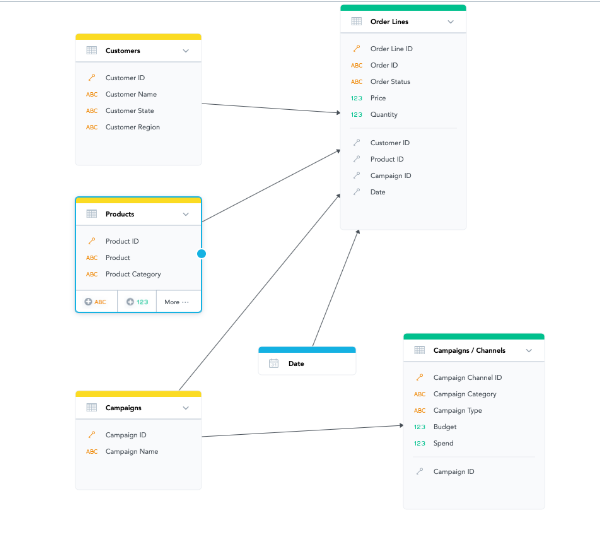
Data Normalization is a basic procedure defined for databases to eliminate such anomalies & prevent redundancy.

Example of Logical Design:



Example 2:





**How To Design A Database: Steps of Designing Database**

The first question you need to ask when designing a database is, how will you specify the structure of the database?

Database designing generally starts with identifying the purpose of your database. The relevant data is then collected and organized into tables. Next, you specify the primary keys and analyze relationships between different tables for an efficient data design. After refining the tables, the last step is to apply normalization rules for table standardization.

* Define the objective of your database

The first step is to determine the purpose of your database. For example, if you are running a small home-based business, you can design a customer database that maintains a list of consumer info to generate emails and reports. Hence, understanding the importance of a database is vital.

At the end of this step, you will have a strong mission statement that you can refer to throughout the database design process. It will help you concentrate on your objectives when making important decisions.

* Locate and consolidate the necessary data

The next step is to collect all kinds of information you might want to store in the database. Begin with the existing data and mull over the questions you want your database to answer. It will help you decide which data needs to be recorded.

* Distribute the data into tables

Once you have amassed all the necessary data items, the next step is to divide them into main entities or subject areas. For example, if you are a retailer, some of your main entities could be products, customers, suppliers, and orders. Each entity will then become a separate table.

* Change data items into columns

Data is segregated into tables, such that every data item becomes a field and is shown as a column. For instance, a customer table might include fields like name, address, email address, and city.

After determining the initial set of columns for every table, you can refine them. For instance, you can record customer names in two separate columns: first and last names. Likewise, you can store the address in five distinct columns based on address, town, state, zip code, and region. It will make it more convenient for you to filter information.

* Identify primary keys

The next step to improve your database design is to select a primary key for every table. This primary key is a column or a set of columns that pinpoint each row distinctively. For instance, in your customer table, the primary key could be customer ID. It will allow you to identify unique rows based on the customer ID.

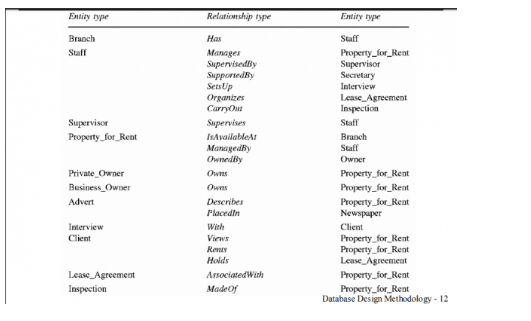
More than one primary key can also exist, called a composite key, including multiple columns. For example, the primary keys in your Order Details table could be order ID and product ID. The composite key can be made using fields with similar or varying data types.

Similarly, if you wish to get an idea of your product sales, you can identify the product ID from the Products table and the order number or ID from the Orders table.

* Determine how tables are related

After dividing data into tables, information needs to be brought together in a meaningful manner. So, you can explore each table and Identify the connection between the tables. If required, you can add fields or form new tables to simplify the relationship based on the types of information.

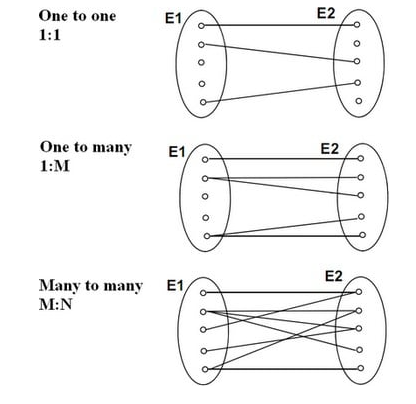
Example of different entity type and relationship types:



This step will create one-to-one, one-to-many, and many-to-many relationships between different table entries.

When a single item from a table is associated with an item from another table, it’s called a one-to-one (1:1) relationship. In a one-to-many (1:M) relationship, an item in one table is related to many items in the other table, such as one customer placing several orders.

A many-to-many (M:N) relationship occurs if more than one table item is related to many items in the other table.



* Enhance your database design

Now that you have all the required tables, fields, and relationships, the next step is to refine your database design by creating and populating your tables with mockup information. Experiment with the sample data by running queries or adding new items. It will help you analyze your design for faults, and you will be able to highlight possible errors. If needed, adjust your design to mitigate those problems.

* Implement the normalization rules

The last step is to implement the normalization rules for your database design. A systematic approach removes redundancy and unwanted characteristics, such as Insertion, Update, and Deletion irregularities.

The multi-step process stores data in a tabular form, which helps eliminate redundant data from the relation tables.