DATABASE DESIGN PHASES

Database design is generally divided into three main phases:
1) CONCEPTUAL DATABASE DESIGN
Conceptual database design is the initial phase of database development, in which the logical structure of the data is defined without considering specific technical details of the database management system (DBMS). The goal is to create a clear and understandable model of the data and their relationships. Here are the main phases of conceptual design:
• Requirements analysis
In this phase, the conceptual model of the database is defined. User needs and business processes are identified.
• Identification of entities
Entities represent objects or concepts of interest (e.g. Customer, Product, Order).
• Definition of relationships
Determine the relationships between entities (e.g. a Customer places one or more Orders). Specify the cardinality of the relationships (1:1, 1:N, N:M).
• Creation of the E-R (Entity-Relationship) Model
The E-R diagram is used to graphically represent entities, relationships and attributes. The primary keys and foreign keys are defined.
The hierarchies are identified (specialization/generalization).
Main Characteristics of the E-R Model
1. Entity
• An entity represents a concrete or abstract object of reality that must be stored in the database.
• It can be a physical object (e.g. Student, Book, Car) or a concept (e.g. Course, Order).
Types of entities
• Strong entity → It has an autonomous existence and a primary key that uniquely identifies it (e.g. Student with ID).
• Weak entity → It depends on another entity and does not have its own key (e.g. Invoice that depends on a Customer).
2. Attributes
• Attributes are the characteristics or properties of an entity.
• Each entity has one or more attributes.
Attribute Types
• Simple → Cannot be split (e.g. Name, Surname).
• Compound → Can be split into sub-attributes (e.g. Address → Street, House Number, City).
• Multivalued → Can have multiple values for a single object (e.g. a Teacher can have multiple Phone Numbers).
• Derived → Can be calculated from other attributes (e.g. age can be derived from birth date).
• Primary Key → Attribute that uniquely identifies an instance of the entity (e.g. Student ID).
3. Relationships

• A relationship connects two or more entities.
• Defines how objects are connected to each other in the database.
Relationship Types:
• One-to-One (1:1) → An entity is associated with only one instance of the other entity (e.g. Person - Fiscal Code).
• One-to-Many (1:N) → An entity is linked to multiple instances of another entity (e.g. Professor - Students).
• Many-to-Many (N:N) → An entity can be associated with multiple instances of the other and vice versa (e.g. Students - Courses).
4. Cardinality
• Cardinality specifies how many instances of an entity can be linked to an instance of another entity.
Types of cardinality:
• (0,1) → The instance can exist without relationships or have only one link (e.g. a Person can have a Passport, but it is not mandatory).
• (1,1) → Each entity must have a fixed relationship with another entity (e.g. each Person has only one Fiscal Code).
• (0,N) → The entity can be linked to multiple instances, but it is not mandatory (e.g. a Customer can place multiple Orders, but not necessarily).
• (1,N) → An entity must be linked to at least one other (e.g. each Student must be enrolled in at least one Course).
5. Associations with Attributes
• Sometimes a relation has its own properties, which do not belong to either of the two entities.
• In these cases, attributes are added to the relation itself.
6. External Identifiers
• If an entity cannot be identified without referring to another entity, an external identifier is used.
7. Generalization and Specialization
• Generalization represents a "parent-child" relationship between entities.
• A more general entity (Superclass) can be divided into more specific entities (Subclasses).
Advantages of the ER Model

It facilitates the design of databases in a clear and structured way.
Independent of the implementation technology.
Understandable even for non-technical users.
It facilitates the optimization of the database before implementation.
Disadvantages of the ER Model

It can become complex in very large systems.
It does not directly represent business processes, but only data.
Example: Suppose we design a database for an online store. The main entities could be:
• Customer (with attributes such as name, email, address)
• Product (with attributes such as name, price, category)
• Order (with attributes such as date, status)
The relationships between entities could be:
• A customer can make many orders (Customer-Order relationship).
• An order can contain multiple products (Order-Product relationship).
2) LOGICAL DESIGN OF A DATABASE
The logical design of a database is the phase in which the conceptual model is transformed into a more detailed representation, suitable for subsequent implementation in a database management system (DBMS). In this phase, the tables, keys and integrity constraints are defined, maintaining an abstract level that is independent of the specific DBMS.
Phases of logical design

1. Transformation of the E-R Model into a Relational Model
• Entities → Tables: each entity becomes a table with its attributes.
• Attributes → Columns: the entity attributes become columns of the table.
• Relations → Foreign keys: the relations between entities are represented with foreign keys.
2. Key Definition
• Primary Key (PK): unique identifier for each record.
• Foreign Key (FK): references to other tables to maintain relationships.
• Unique Keys (Unique Key): constraints that prevent duplication of certain values
3. Normalization
Normalization is a process that serves to reduce data redundancy and avoid inconsistency problems. The information is divided into multiple tables, with the aim of maintaining only the necessary data in each table. Normal forms are applied to eliminate redundancies and anomalies.
1NF (First Normal Form): elimination of repeated groups and creation of tables with atomic attributes.
• All attributes must contain atomic values (not divisible).
• There must be no repeated groups or multiple values in a column.
• Each row must have a unique identifier (primary key).
2NF (Second Normal Form): elimination of partial dependencies.
• Must be in 1NF.
• Non-key attributes must depend completely on the primary key.
3NF (Third Normal Form): Elimination of transitive dependencies.
• Must be in 2NF.
• Non-key attributes must depend only on the primary key and not on other non-key attributes.
Example: Instead of directly storing the customer address data in each order (causing redundancy), you can create a separate table for addresses, associated with the customer.
Normalization is essential to ensure efficiency, consistency and scalability in a relational database. However, in some cases it can increase the complexity of queries due to the increased number of tables and relationships between them.
4. Defining Integrity Constraints
• Referential integrity: ensures consistency between primary keys and foreign keys.

• Data integrity: enforces rules on acceptable values in fields (e.g. NOT NULL, CHECK).
5. Optimizing the logical model
• Performance analysis and complexity reduction.
• Creating indexes to speed up searches.
• Using views to improve data management.

3) PHYSICAL DESIGN OF A DATABASE
The physical design of a database is the final phase of database development, in which the logical model is implemented in a specific database management system (DBMS). In this phase, performance is optimized and physical aspects such as space allocation, indexes to speed up searches, and security are managed.
Physical design phases
1. Choice of DBMS and storage structures

• Selection of the database management system (e.g. MySQL, PostgreSQL, SQL Server, Oracle).
• Configuration of database files and storage strategies.
• Definition of optimal data types for each field.
2. Definition of table structure
• Optimization of data types (e.g. INT, VARCHAR, TEXT, BLOB).
• Definition of constraints such as NOT NULL, DEFAULT, CHECK.
3. Creating indexes
• Primary Index: based on the primary key.
• Secondary indexes: to speed up searches on other fields used in filters (INDEX in SQL).
• Full-text indexes: to optimize text searches.
4. Key and constraint management
• Implementation of primary keys and foreign keys with FOREIGN KEY.
• Definition of ON DELETE CASCADE, ON UPDATE SET NULL policies, etc.
• Management of uniqueness constraints (UNIQUE).
5. Partitioning and storage optimization
• Partitioning: splitting very large tables into multiple parts to improve performance.
• Denormalization (when necessary): creation of controlled redundancies to speed up queries.
6. Concurrency management and transactions
• Implementation of locks (e.g. READ COMMITTED, SERIALIZABLE).
• Configuration of transaction isolation strategies.
7. Security and access control
• Creating users and roles with specific permissions (GRANT, REVOKE in SQL).
• Implementing encryption for sensitive data.
8. Backup and recovery
• Defining backup strategies (full, incremental, differential).
• Configuring replicas for high availability.

Conclusions

Designing a database is a fundamental step in creating applications that manage information. A good design ensures that data is easily accessible, efficient and secure. At each stage of the design, it is important to consider the specific needs of the application and choose the most suitable structures for the context in which it will operate.