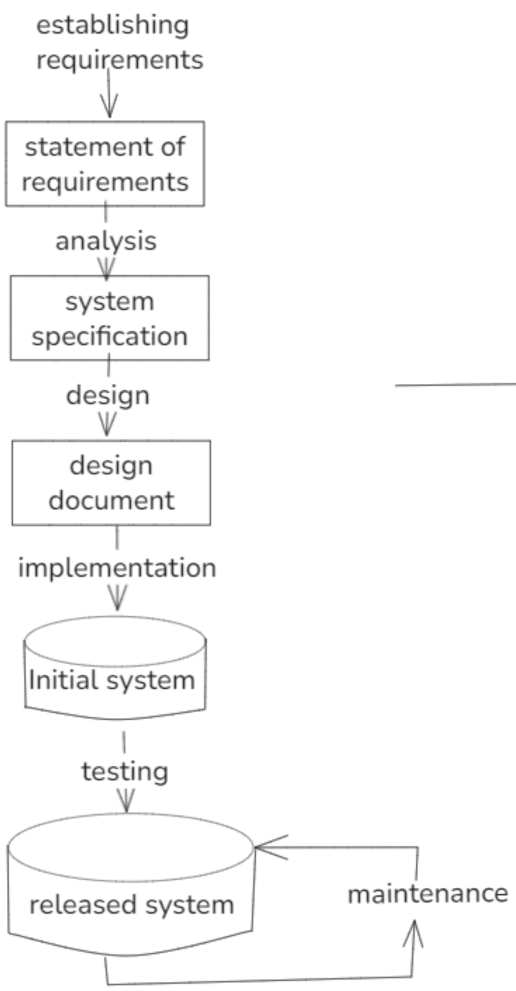
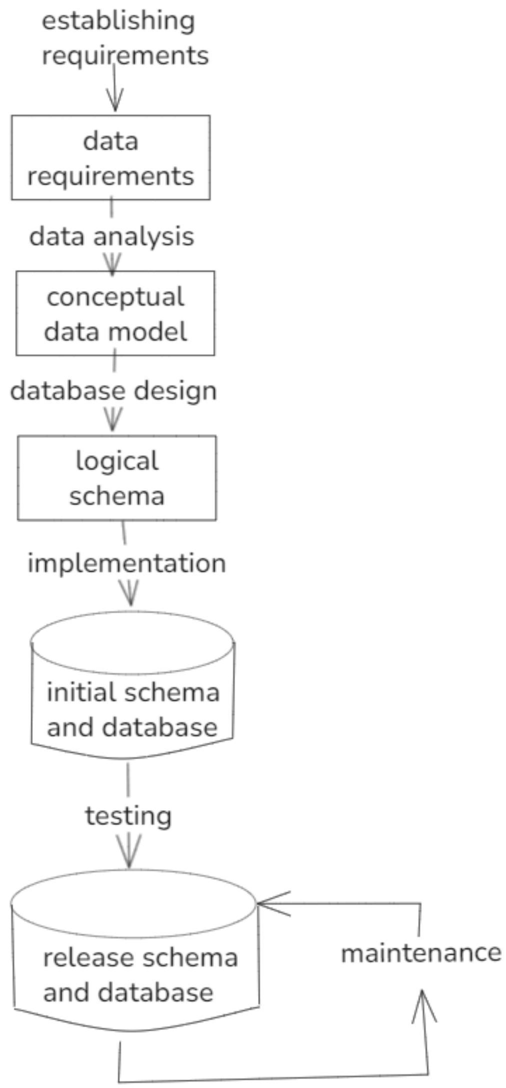
1 - Database Development Process

- Among information system development methods, we discussed the software development lifecycle.
- This methodology can be modeled using the Waterfall model.
- The waterfall model illustrates the process as a strict sequence of steps where the output of one step is the input of the next, and where an entire step must be completed before moving on to the next.
- This waterfall model can be adapted for database development:



waterfall model



A model waterfall of the activities and their outputs for database development.

- To use this model, we assume that:
  - The specification and creation of a schema to define the data in a database can be separated from the user processes that use the database.
  - The three-schema architecture can serve as a basis for distinguishing the activities associated with a schema (Conceptual, Logical, and Physical).

    - Constraints can be represented once in the database rather than in each user process that uses the data.
- The steps can be summarized as follows:

1-1 Requirements Gathering

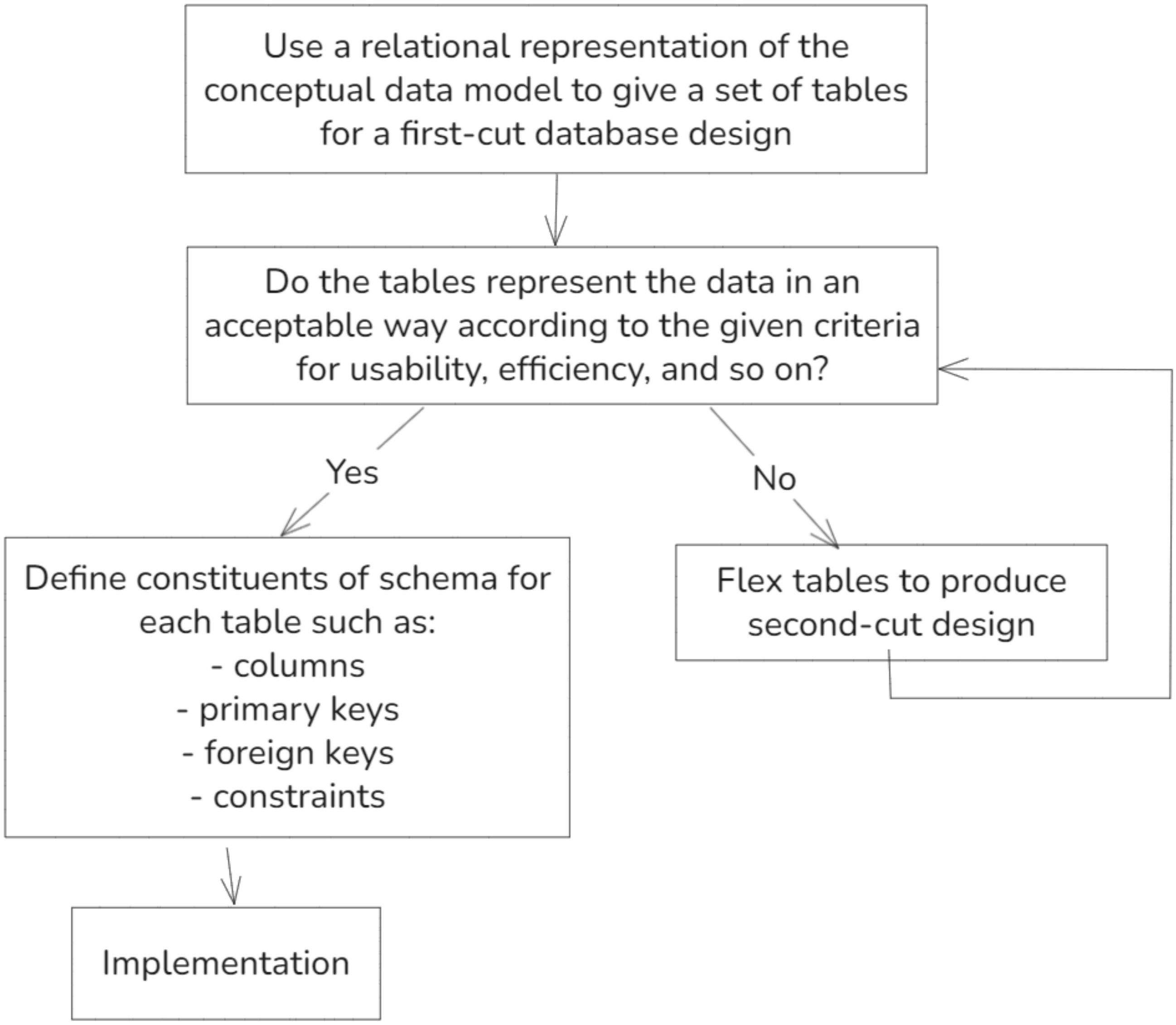
- Database designers must interview users to understand their needs.
- The result is a document that includes the detailed requirements provided by the users.

1-2 Data Analysis

- The goal is to obtain a detailed description of the data that will meet the users' requirements.
- The description must address both the properties of the data and its use.
- This step will produce the conceptual data model.
- The conceptual data model focuses on the meaning and structure of the data, but not on the details affecting how it is implemented.

1-3 Logical Design

- A relational representation of the conceptual data model can be used as input in the logical design process.
- The result of this step is a detailed relational specification, the logical schema, of all the tables and constraints necessary to satisfy the data description in the conceptual data model.



1-4 Implementation and Schema Realization:

- Build a database according to the specification of a logical schema.
- Implementation is influenced by the choice of DBMS, database tools, and operating environment.
- Create the database according to the logical schema. This can be done:
  - Either using SQL:
    - Using SQL to create tables and constraints
    - Saving SQL statements to a text file
- Or using a database tool such as SQL Server Management Studio or Microsoft Access

1-5 Populating the Database

- Two approaches to populating database tables:
  - Using existing data
  - Using user applications developed for the database
- Existing data sources: => use the import and export tools already used in DBMSs
  - Other databases

- Data files
- Converting paper documents into computer files

2- Functional Dependency

2-1 Definition

- A functional dependency (FD) is a relationship between two attributes, usually between the primary key and other non-key attributes within a table.
- For any relation R, the attribute Y is functionally dependent on the attribute X (usually the primary key, PK), if for each valid instance of X, this value of X uniquely determines the value of Y.
- The notation for a functional dependency is X → Y, where X is the determinant and Y is the dependent.
- *Example:*
  - SSN ——-> Name, Address, Date of Birth
  - ISBN ———> Title

2-2 Inference Rules and Armstrong's Functional Dependency Axioms

- Armstrong's axioms are a set of rules that can be used to infer all functional dependencies (FDs) in a table.
- Let R(U) be a relation defined on the set of attributes U. The letters X, Y, and Z represent any subset and the union of any two sets of attributes.
- There are 3 axioms:

2-2-1 Axiom of Reflexivity:

- If Y is a subset of X, then X → Y.

$$\text{If } Y \subseteq X, \text{ then } X->Y$$

2-2-2 Augmentation Axiom:

If X → Y, then XZ → YZ for all Z.

- In other words:
  - Every non-key attribute must be entirely dependent on the primary key (PK).
- *Example:*
  - In the example below, StudentName, Address, City, Province, and PostalCode depend only on StudentNo. and not on the course.

StudentCourse(<u>StudentNo, Course</u>, StudentName, Address, City, Province, PostalCode, Grade, CompletionDate)
StudentNo, Course ---> StudentName, Address, City, Province, PostalCode, Grade, CompletionDate

  - This situation is undesirable because every non-key attribute must be entirely dependent on the PK. In this case, the student information is only partially dependent on the primary key (StudentID).

- To resolve this issue, we need to split the original table into two as follows:
  - **Table 1:** <u>StudentID, Course</u>, Grade, CompletionDate
  - **Table 2:** <u>StudentID</u>, StudentName, Address, City, Province, PostalCode

2-2-3 Transitivity Axiom:

If X → Y and Y → Z, then X → Z.

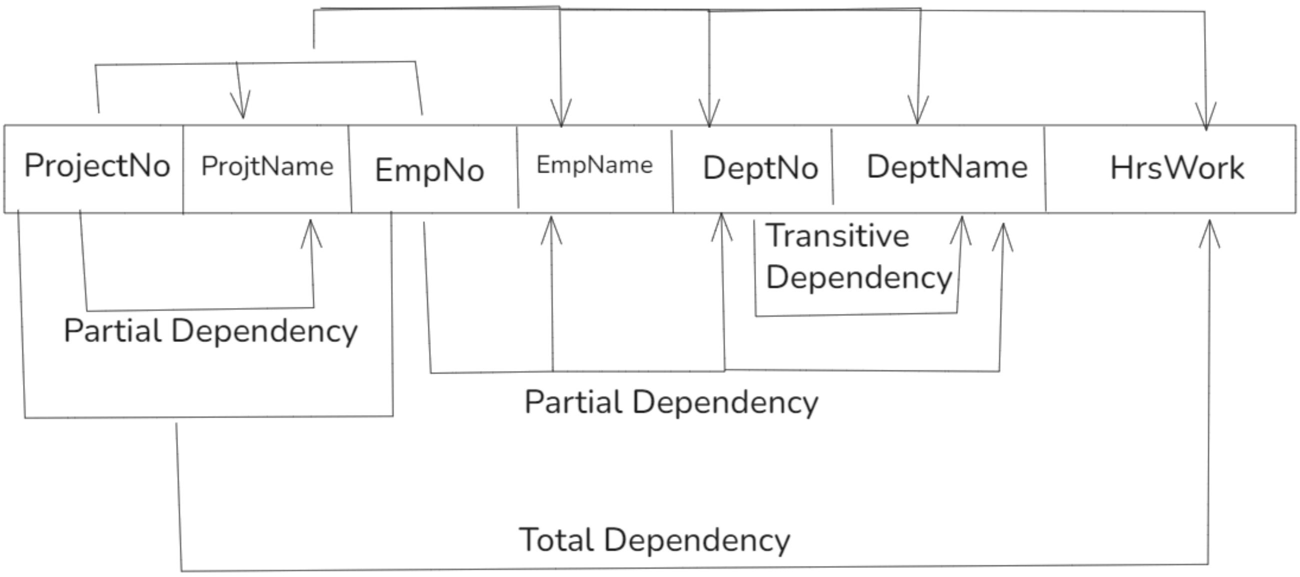
- *Example:*
  - The table below contains information that is not directly related to the student: StudentID —> StudentName, Address, City, Province, PostalCode, ProgramID, ProgramName
- For example, ProgramID and ProgramName should have their own table. ProgramName does not depend on StudentID; It depends on ProgramID.
- This situation is undesirable because a non-key attribute (ProgramName) depends on another non-key attribute (ProgramID).
- To resolve this issue, we need to split this table into two: one to contain student information and the other to contain program information.
- Table 1: StudentID → StudentName, Address, City, Province, PostalCode, ProgramID
- Table 2: ProgramID → ProgramName
- However, we still need to leave a foreign key in the Student table to identify the program in which the student is enrolled.

2-2-4 Union and Decomposition Rules

- The union rule states that if X → Y and X → Z, then X → YZ.
- The decomposition rule states that if X → Y X , then X → Y and X → Z separately.
- When two tables are linked by a key attribute, they can be merged into a single table.
- Some database administrators (DBAs) prefer to keep tables separate for the following reasons:
  - Each table describes a different entity, and entities should be kept separate.
  - If the value of the attribute in question is often zero, it is not necessary to include it in the same table as the associated attribute.

2-3 Dependency Diagram

- A dependency diagram is a graphical representation of the dependencies (FDs) in a table.



- ProjectNo and EmpNo, combined, form the primary key (PK).
- Partial dependencies:
  - ProjectNo → ProjectName
  - EmpNo → EmpName, DeptNo
- Transitive dependency:
  - DeptNo → DeptNo
- Total dependency:
  - ProjectNo, EmpNo → HoursWork

3- Normalization

- Normalization should be part of the database design process.
- Normalization reduces data redundancy, which can improve database performance, integrity, and maintainability.
- The result of the normalization process is the transformation of relationships into normal forms (NF).
- There are six normal forms (NF), each with specific requirements for functional dependencies.
- The most commonly used normal forms are the first normal form (1NF), the second normal form (2NF), and the third normal form (3NF).

3-1 First Normal Form (1NF)

- A relation is in 1NF if it does not contain repeating groups.
- A repeating group is a set of attributes that can take multiple values for the same value of another attribute.

- To normalize a relation to 1NF, repeating groups must be removed and new relations created.
- Example:
  - Student_Grade_Report (<u>StudentNo</u>, StudentName, Specialization, CourseNo, CourseName, InstructorNo, InstructorName, InstructorLocation, Grade)

    => Student (<u>StudentNo</u>, StudentName, Specialization)
    StudentCourse (<u>StudentNo, CourseNo</u>, CourseName, InstructorNo, InstructorName, InstructorLocation, Grade)

3-2 Second Normal Form (2NF)

- A relation is in 2NF if it is in 1NF and all non-key attributes are fully dependent on the primary key.
- A partial dependency is a dependency in which a non-key attribute depends on a subset of the primary key.
- To normalize a relation to 2NF, partial dependencies must be eliminated.
- Example:
  - Student (<u>StudentNo</u>, StudentName, Major)
  - CourseGrade (<u>StudentNo, CourseNo</u>, Grade)
  - CourseInstructor (<u>CourseNo</u>, CourseName, InstructorNo, InstructorName, InstructorLocation)

3-3 Third Normal Form (3NF)

- A relation is in 3NF if it is in 2NF and all transitive dependencies are eliminated.
- A transitive dependency is one in which a non-key attribute depends on another non-key attribute.
- To normalize a relationship to 3NF, transitive dependencies must be eliminated.
- Example:
  - Student (<u>StudentNo</u>, StudentName, Major)
  - CourseGrade (<u>StudentNo, CourseNo</u>, Grade)
  - Course (<u>CourseNo</u>, CourseName, InstructorNo)
  - Instructor (<u>InstructorNo</u>, InstructorName, InstructorLocation)

4- Developing Entity-Relationship Diagrams

- Record all entities discovered during the information gathering phase.
- Document all attributes belonging to each entity. Select candidate and primary keys. Ensure that all non-key attributes of each entity are fully dependent on the primary key.
- Develop an initial ER diagram and review it with the appropriate personnel.
- Create new entities for multivalued attributes and repeated groups.
- Verify the ER model by normalizing the tables.

References

- Watt, Adrienne, and Nelson Eng. *Database design*. 2nd Edition. BCcampus, 2014