Lecture Notes
On
System Analysis and Design
Spiral Model

UNIT-II

Prepared by,
Alok Haldar
Assistant Professor,
Department of Computer Science and BCA,
Kharagpur College

UNIT-II : Data Dictionary, Pseudo code, The Process of System Design, Difference between Logical Design and Physical Design, Top-Down Design and Functional Decomposition, Forms-Driven Methodology.

Data Dictionary

A data dictionary is a centralized repository of metadata. Metadata is data about data. Some examples of what might be contained in an organization's data dictionary include:

•The names of fields contained in all of the organization's databases

•What table(s) each field exists in

•What database(s) each field exists in

•The data types, e.g., integer, real, character, and image of all fields in the organization's databases

•The sizes, e.g., LONG INT, DOUBLE, and CHAR(64), of all fields in the organization's databases

•An explanation of what each database field means

•The source of the data for each database field

•A list of applications that reference each database field

•The relationship between fields in all of the organization's databases

•Default values that exist for all fields in all of the organization's databases

•Who has access to each field

Logical data model :

A logical data model describes the data in as much detail as possible, without regard to how they will be physical implemented in the database. Features of a logical data model include:

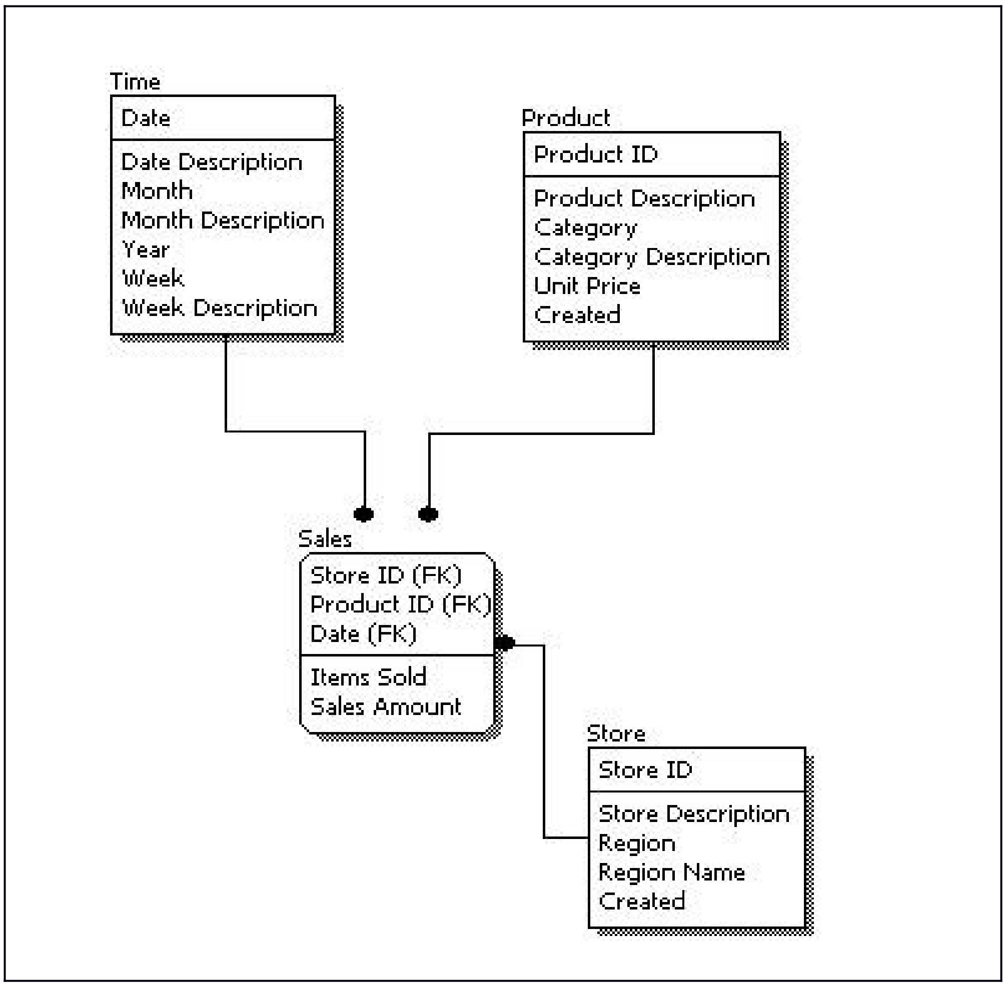
- Includes all entities and relationships among them.
- All attributes for each entity are specified.
- The primary key for each entity is specified.
- Foreign keys (keys identifying the relationship between different entities) are specified.
- Normalization occurs at this level.

The steps for designing the logical data model are as follows:

1. Specify primary keys for all entities.
2. Find the relationships between different entities.
3. Find all attributes for each entity.
4. Resolve many-to-many relationships.
5. Normalization.

The figure below is an example of a logical data model.

Logical Data Model



Comparing the logical data model shown above with the conceptual data model diagram, we see the main differences between the two:

- In a logical data model, primary keys are present, whereas in a conceptual data model, no primary key is present.
- In a logical data model, all attributes are specified within an entity. No attributes are specified in a conceptual data model.
- Relationships between entities are specified using primary keys and foreign keys in a logical data model. In a conceptual data model, the relationships are simply stated, not specified, so we simply know that two entities are related, but we do not specify what attributes are used for this relationship.

Physical data model :

Physical data model represents how the model will be built in the database. A physical database model shows all table structures, including column name, column data type, column constraints, primary key, foreign key, and relationships between tables. Features of a physical data model include:

- Specification all tables and columns.

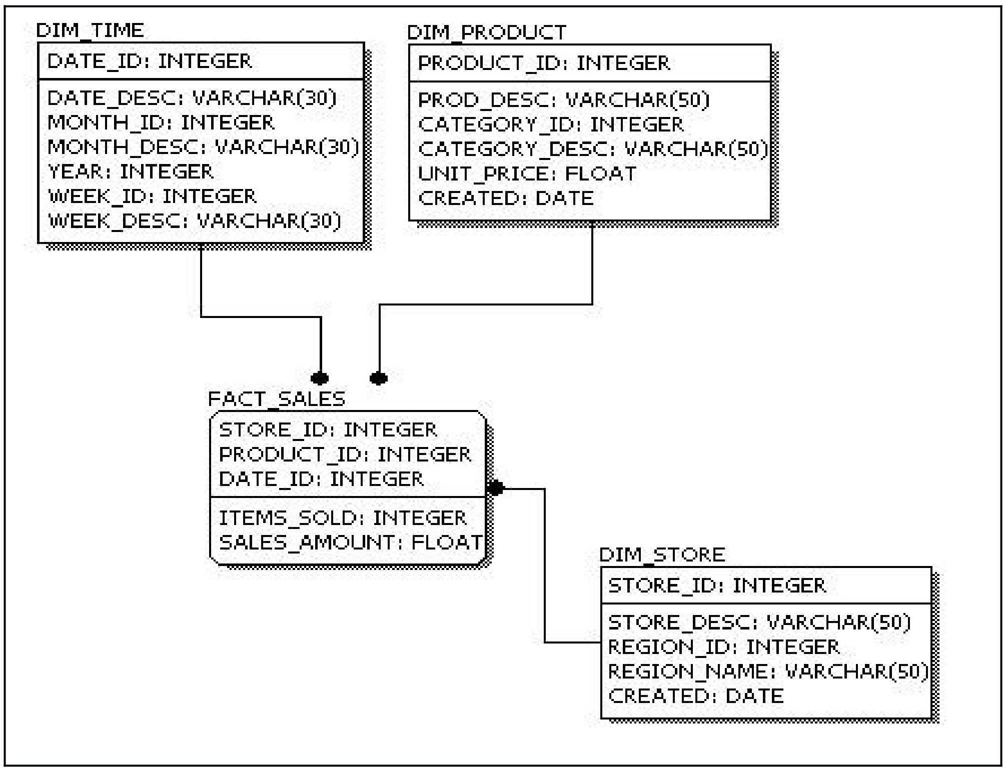
- Foreign keys are used to identify relationships between tables.
- Denormalization may occur based on user requirements.
- Physical considerations may cause the physical data model to be quite different from the logical data model.
- Physical data model will be different for different RDBMS. For example, data type for a column may be different between MySQL and SQL Server.

The steps for physical data model design are as follows:

1. Convert entities into tables.
2. Convert relationships into foreign keys.
3. Convert attributes into columns.
4. Modify the physical data model based on physical constraints / requirements.

The figure below is an example of a physical data model.

Physical Data Model



Comparing the physical data model shown above with the logical data model diagram, we see the main differences between the two:

- Entity names are now table names.
- Attributes are now column names.
- Data type for each column is specified. Data types can be different depending on the actual database being used.

Top-down and **bottom-up Design :**

Top-down and **bottom-up** are both strategies of information processing and knowledge ordering, used in a variety of fields including software, humanistic and scientific theories and management and organization.

A **top-down** approach is essentially the breaking down of a system to gain insight into its compositional sub-systems in a reverse engineering fashion. In a top-down approach an overview of the system is formulated, specifying, but not detailing, any first-level subsystems.

Each subsystem is then refined in yet greater detail, sometimes in many additional subsystem levels, until the entire specification is reduced to base elements.

A top-down model is specified with the assistance of "black boxes", which makes it easier to manipulate. However, black boxes may fail to clarify elementary mechanisms or be detailed enough to realistically validate the model. Top down approach starts with the big picture. It breaks down from there into smaller segments.

Top-down approaches emphasize planning and a complete understanding of the system. It is inherent that no coding can begin until a sufficient level of detail has been reached in the design of at least some part of the system.

Top-down approaches are implemented by attaching the stubs in place of the module. This, however, delays testing of the ultimate functional units of a system until significant design is complete.

Top-down is a programming style of traditional procedural languages, in which design begins by specifying complex pieces and then dividing them into successively smaller pieces. The technique for writing a program using top–down methods is to write a main procedure that names all the major functions it will need. Later, the programming team looks at the requirements of each of those functions and the process is repeated.

A **bottom-up** approach is the piecing together of systems to give rise to more complex systems, thus making the original systems sub-systems of the emergent system. Bottom-up processing is a type of information processing based on incoming data from the environment to form a perception.

In a **bottom-up approach** the individual base elements of the system are first specified in detail. These elements are then linked together to form larger subsystems, which then in turn are linked, sometimes in many levels, until a complete top-level system is formed. This strategy often resembles a "seed" model, by which the beginnings are small, but eventually grow in complexity and completeness. Object-oriented programming (OOP) is a paradigm that uses "objects" to design applications and computer programs.

Bottom-up emphasizes coding and early testing, which can begin as soon as the first module has been specified. This approach, however, runs the risk that modules may be coded without having a clear idea of how they link to other parts of the system, and that such linking may not be as easy as first thought. Re-usability of code is one of the main benefits of the bottom-up approach.

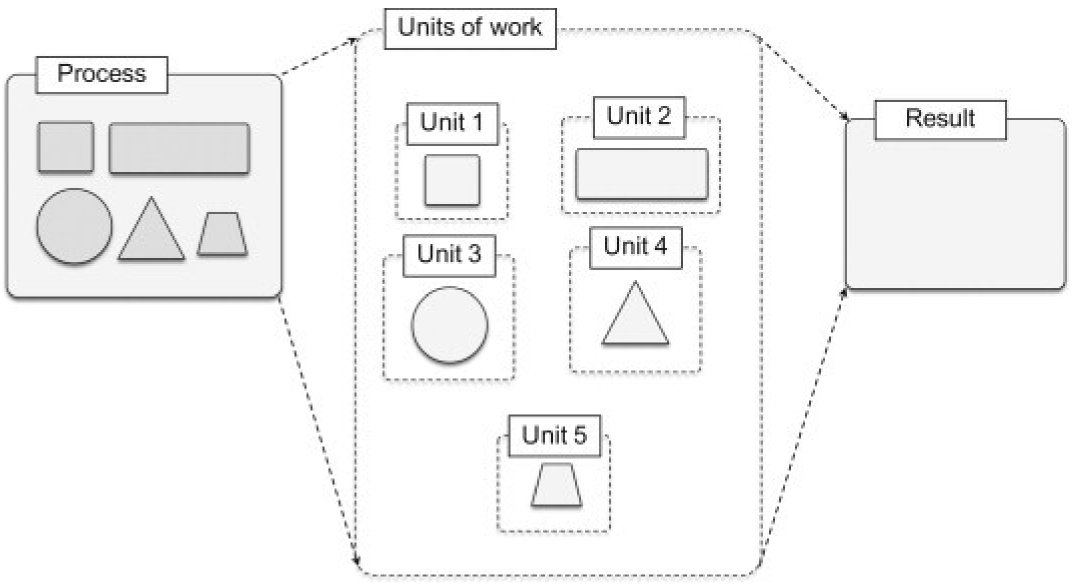
Functional decomposition :

Functional decomposition is the process of identifying *functionally distinct but independent computations*.

It is the breakdown of activity requirements in terms of a hierarchical ordering. A *function* is defined as a continuously occurring activity that exists to meet the needs of the corporation. Within each function are many *processes*. These processes have a start activity, a process activity, and a termination activity, which completes the process. Each process may or may not be broken down into subprocesses. Each subprocess, like its parent, also has an initiation, an activity state, and a termination, and it differs from the process in that it represents activity at the lowest level. That is, it is the activity or event that takes place at the entity level.

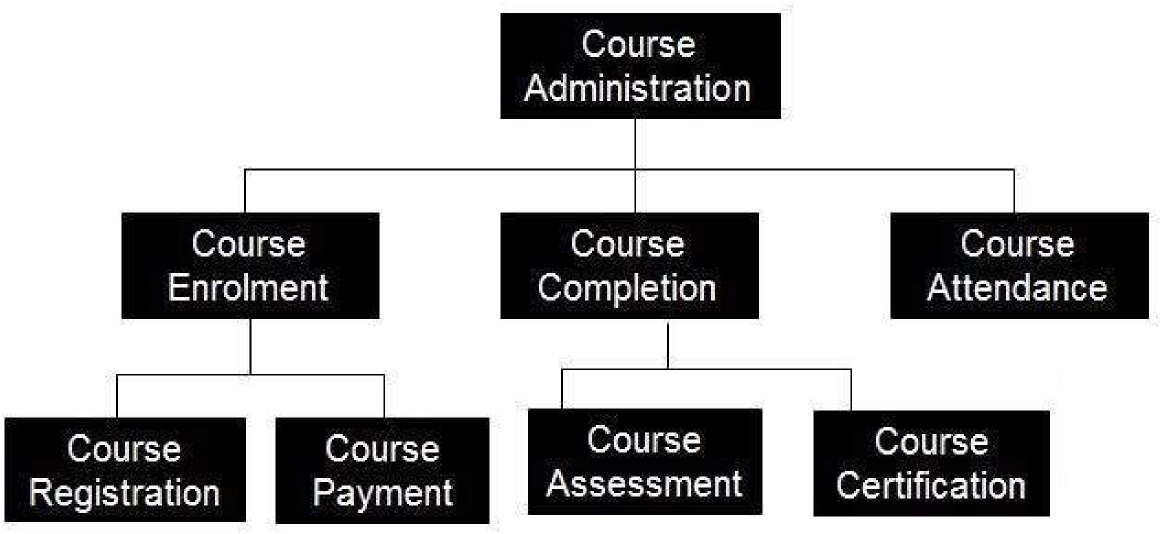
When and How?

- Functional decomposition is mostly used during the project analysis phase in order to produce functional decomposition diagrams as part of the functional requirements document.
- Functional Decomposition is done after meeting with business analysts and subject matter expertise.
- Decompose the first level components with their functions and continue to decompose to lower levels until sufficient level of detail is achieved
- Perform an end-to-end walk-through of the business operation and check each function to confirm that it is correct.



Example:

Dscribes the Functional Decomposition :



The Role of Design Methodology

Software design methodology provides a logical and systematic means of proceeding with the design process as well as a set of guidelines for decision-making. The design methodology provides a sequence of activities, and often uses a set of notations or diagrams. The design methodology is especially important for large complex projects that involve programming-in-the-large (where many designers are involved); the use of a methodology establishes a set of common communication channels for translating design to codes and a set of common objectives.
In addition, there must be an objective match between the overall character of the problem and the features of the solution approach, in order for a methodology to be efficient.