Chapter 2: Requirement Analysis and Specification

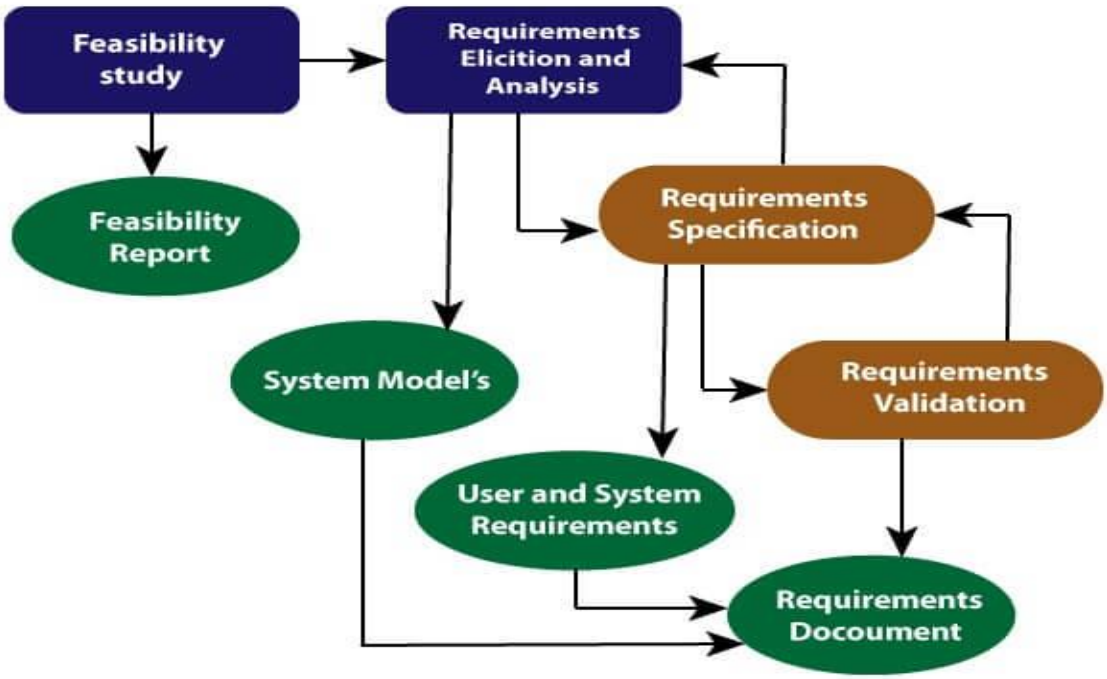
Requirement Engineering

Requirements engineering (RE) refers to the process of defining, documenting, and maintaining requirements in the engineering design process. Requirement engineering provides the appropriate mechanism to understand what the customer desires, analyzing the need, and assessing feasibility, negotiating a reasonable solution, specifying the solution clearly, validating the specifications and managing the requirements as they are transformed into a working system. Thus, requirement engineering is the disciplined application of proven principles, methods, tools, and notation to describe a proposed system's intended behaviour and its associated constraints.

Requirement Engineering Process

It is a four-step process, which includes -

1. Feasibility Study
2. Requirement Elicitation and Analysis
3. Software Requirement Specification
4. Software Requirement Validation
5. Software Requirement Management



Requirement Engineering Process

1. Feasibility Study:

The objective behind the feasibility study is to create the reasons for developing the software that is acceptable to users, flexible to change and conformable to established standards.

Types of Feasibility:

1. **Technical Feasibility** –

   Technical feasibility evaluates the current technologies, which are needed to accomplish customer requirements within the time and budget.

2. **Operational Feasibility** –

   Operational feasibility assesses the range in which the required software performs a series of levels to solve business problems and customer requirements.

3. **Economic Feasibility** –

   Economic feasibility decides whether the necessary software can generate financial profits for an organization.

2. Requirement Elicitation and Analysis:

This is also known as the **gathering of requirements**. Here, requirements are identified with the help of customers and existing systems processes, if available. Analysis of requirements starts with requirement elicitation.

The requirements are analyzed to identify inconsistencies, defects, omission, etc. We describe requirements in terms of relationships and also resolve conflicts if any.

Requirements elicitation is perhaps the most difficult, most error-prone and most communication intensive software development. It can be successful only through an effective customer-developer partnership. It is needed to know what the users really need.

2.1 Identifying Stakeholders:

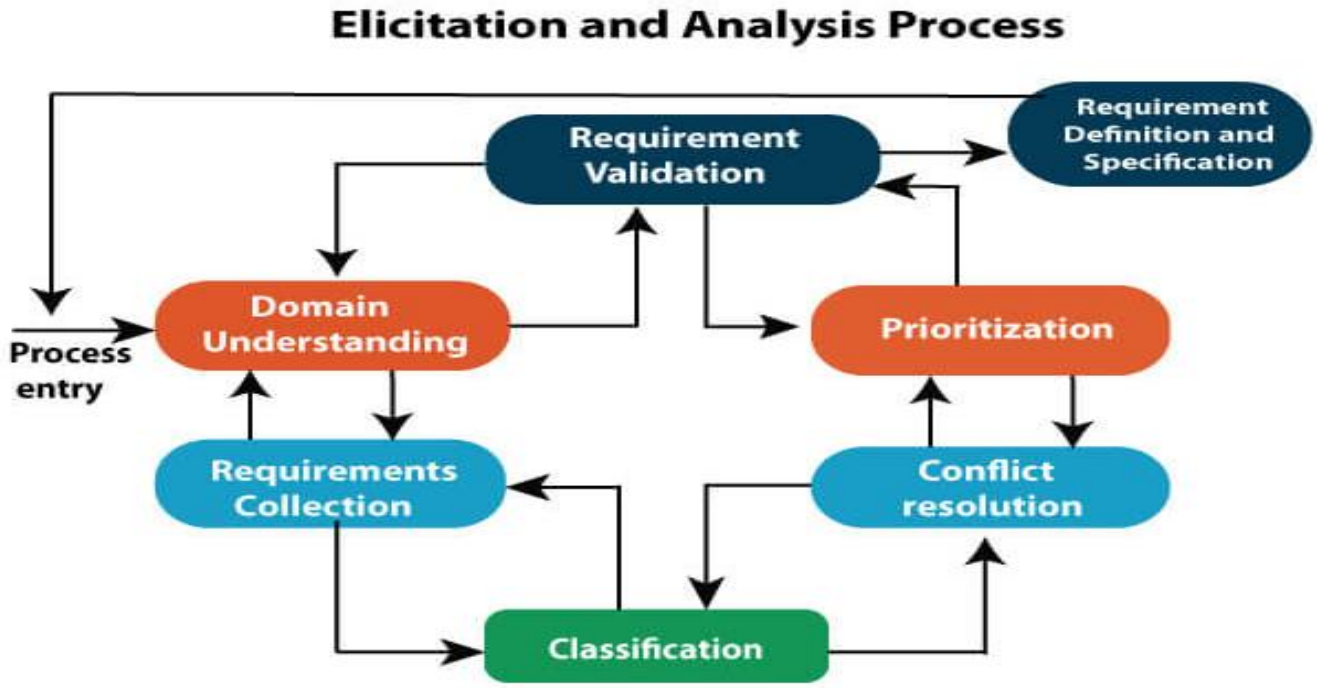
Before the requirements gathering, it is important to know the people who contribute and help gather the requirements.

2.2 Characteristic of Stakeholders

Stakeholder interest may be positively or negatively impacted by the performance of the project. They make confliction for changing requirements and objectives which are not be easy to handle. Involvement of stakeholder in the project phase improves the probability of success of the project.

2.3 Problems of Elicitation and Analysis

- User not sure
- Communication gap
- Conflicting requirements
- Volatiles requirements



2.4 Elicitation techniques:

Requirements elicitation Activities:
Requirements elicitation includes the subsequent activities. Few of them are listed below –

- Knowledge of the overall area where the systems is applied.
- The details of the precise customer problem where the system are going to be applied must be understood.
- Interaction of system with external requirements.
- Detailed investigation of user needs.
- Define the constraints for system development.

Requirements elicitation Methods: There are a number of requirements elicitation methods. Few of them are listed below –

1. Interviews
2. Focus group
3. Brainstorming Sessions
4. Facilitated Workshop
5. Prototyping
6. Direct observation

7. Questionnaires

His success of an elicitation technique used depends on the maturity of the analyst, developers, users, and the customer involved.

1. Interviews: Objective of conducting an interview is to understand the customer's expectations from the software. It is impossible to interview every stakeholder hence representatives from groups are selected based on their expertise and credibility. Interviews maybe are open-ended or structured.

1. In open-ended interviews there is no pre-set agenda. Context free questions may be asked to understand the problem.
2. In structured interview, agenda of fairly open questions is prepared. Sometimes a proper questionnaire is designed for the interview.

2. Brainstorming Sessions:

- It is a group technique
- It is intended to generate lots of new ideas hence providing a platform to share views
- A highly trained facilitator is required to handle group bias and group conflicts.
- Every idea is documented so that everyone can see it.
- Finally, a document is prepared which consists of the list of requirements and their priority if possible.

3. Facilitated Application Specification Technique: Its objective is to bridge the expectation gap – difference between what the developers think they are supposed to build and what customers think they are going to get.
A team oriented approach is developed for requirements gathering. Each attendee is asked to make a list of objects that are-

1. Part of the environment that surrounds the system
2. Produced by the system
3. Used by the system

Each participant prepares his/her list, different lists are then combined, redundant entries are eliminated, team is divided into smaller sub-teams to develop mini-specifications and finally a draft of specifications is written down using all the inputs from the meeting.

3. Software Requirement Specification:

Software requirement specification is a kind of document which is created by a software analyst after the requirements collected from the various sources - the requirement received by the customer written in ordinary language. It is the job of the analyst to write the requirement in technical language so that they can be understood and beneficial by the development team.

The models used at this stage include ER diagrams, data flow diagrams (DFDs), function decomposition diagrams (FDDs), data dictionaries, etc.

- **Data Flow Diagrams:** Data Flow Diagrams (DFDs) are used widely for modelling the requirements. DFD shows the flow of data through a system. The system may be a company, an organization, a set of procedures, a computer hardware system, a software system, or any combination of the preceding. The DFD is also known as a data flow graph or bubble chart.
- **Data Dictionaries:** Data Dictionaries are simply repositories to store information about all data items defined in DFDs. At the requirements stage, the data dictionary should at least define customer data items, to ensure that the customer and developers use the same definition and terminologies.
- **Entity-Relationship Diagrams:** Another tool for requirement specification is the entity-relationship diagram, often called an "***E-R diagram***." It is a detailed logical representation of the data for the organization and uses three main constructs i.e. data entities, relationships, and their associated attributes.

1. **Specify Requirements in the SRS Documents –**
    - Contractual agreements between the customer and suppliers.
2. **IEEE Standard and SRS Template**

1.	**Introduction**
	1.1 Purpose
	1.2 Scope
	1.3 Overview

3. **Checklist for writing a good SRS Documents**
    - Correctness
    - Completeness
    - Consistency
    - Verifiability
    - Clarity
    - Priority
    - Modifiability

4. Software Requirement Validation:

After requirement specifications developed, the requirements discussed in this document are validated. The user might demand illegal, impossible solution or experts may misinterpret the needs. Requirements can be the check against the following conditions -

- If they can practically implement
- If they are correct and as per the functionality and specially of software
- If there are any ambiguities
- If they are full
- If they can describe

Requirements Validation Techniques

- **Requirements reviews/inspections:** systematic manual analysis of the requirements.
- **Prototyping:** Using an executable model of the system to check requirements.
- **Test-case generation:** Developing tests for requirements to check testability.
- **Automated consistency analysis:** checking for the consistency of structured requirements descriptions.

5. Software Requirement Management:

Requirement management is the process of managing changing requirements during the requirements engineering process and system development. New requirements emerge during the process as business needs a change, and a better understanding of the system is developed. The priority of requirements from different viewpoints changes during development process. The business and technical environment of the system changes during the development.

Prerequisite of Software requirements

Collection of software requirements is the basis of the entire software development project. Hence they should be clear, correct, and well-defined. A complete Software Requirement Specifications should be:

- Clear
- Correct
- Consistent
- Coherent
- Comprehensible
- Modifiable
- Verifiable
- Prioritized
- Unambiguous
- Traceable
- Credible source

Software Requirements: Largely software requirements must be categorized into two categories:

1. **Functional Requirements:** Functional requirements define a function that a system or system element must be qualified to perform and must be documented in different forms. The functional requirements are describing the behaviour of the system as it correlates to the system's functionality.
2. **Non-functional Requirements:** This can be the necessities that specify the criteria that can be used to decide the operation instead of specific behaviours of the system. Non-functional requirements are divided into two main categories:

- **Execution qualities** like security and usability, which are observable at run time.
- **Evolution qualities** like testability, maintainability, extensibility, and scalability that embodied in the static structure of the software system.

Requirement Modelling

Requirements for a computer-based system can be seen in many different ways. Some software people argue that it's worth using a number of different modes of representation while others believe that it's best to select one mode of representation. The specific elements of the requirements model are dedicated to the analysis modelling method that is to be used.

Structured Analysis	
Types	**Corresponding Diagram**
Data Modeling	ER Diagram
Functional Modeling	Data Flow Diagram
	Control Flow Diagram
Behavioural Modeling	State Transition Diagram

- **Data Modelling:** Using a scenario-based approach, system is described from user's point of view. For example, basic use cases and their corresponding use-case diagrams evolve into more elaborate template-based use cases. Figure 1(a) depicts a UML activity diagram for eliciting requirements and representing them using use cases. There are three levels of elaboration.

  Data modeling (data modelling) is the process of creating a data model for the data to be stored in a database. This data model is a conceptual representation of Data objects, the associations between different data objects, and the rules.

  Data modeling helps in the visual representation of data and enforces business rules, regulatory compliances, and government policies on the data. Data Models

ensure consistency in naming conventions, default values, semantics, security while ensuring quality of the data.

Data Model

The **Data Model** is defined as an abstract model that organizes data description, data semantics, and consistency constraints of data. The data model emphasizes on what data is needed and how it should be organized instead of what operations will be performed on data. Data Model is like an architect's building plan, which helps to build conceptual models and set a relationship between data items.
The two types of Data Modeling Techniques are

1. Entity Relationship (E-R) Model
2. UML (Unified Modelling Language)

Why use Data Model?

The primary goal of using data model is:

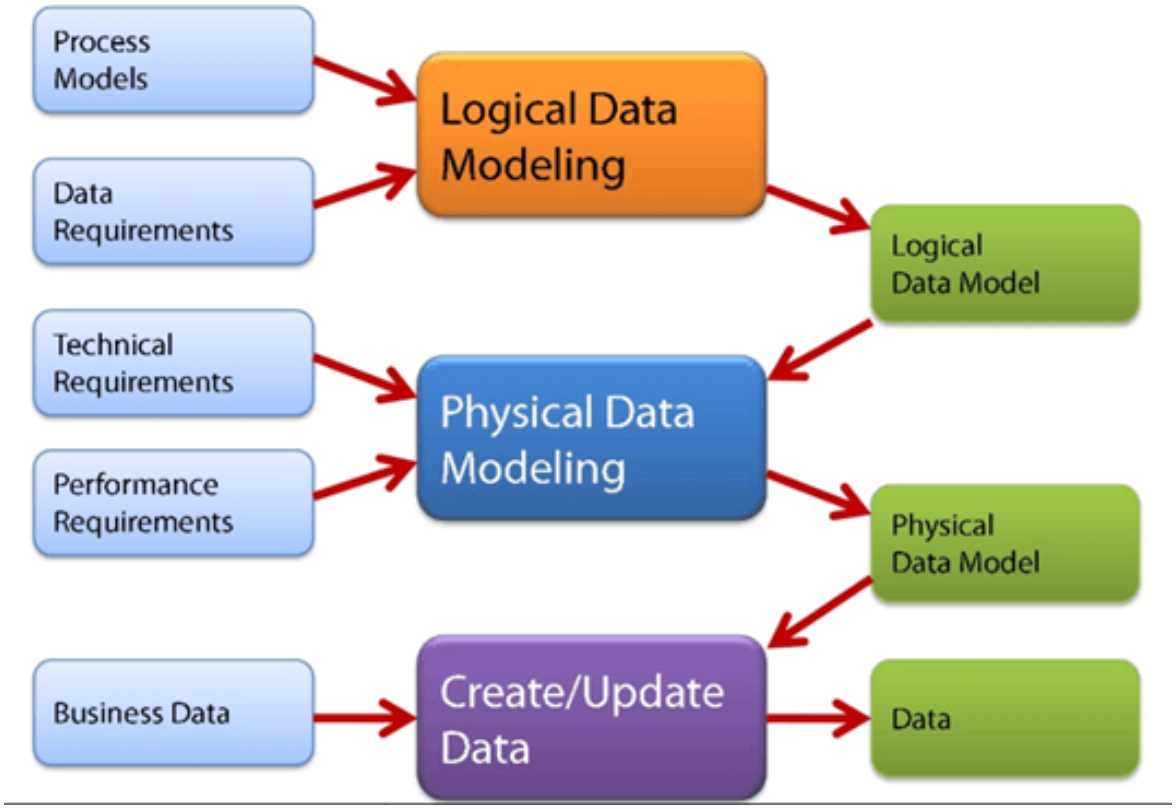
- Ensures that all data objects required by the database are accurately represented. Omission of data will lead to creation of faulty reports and produce incorrect results.
- A data model helps design the database at the conceptual, physical and logical levels.
- Data Model structure helps to define the relational tables, primary and foreign keys and stored procedures.
- It provides a clear picture of the base data and can be used by database developers to create a physical database.
- It is also helpful to identify missing and redundant data.
- Though the initial creation of data model is labor and time consuming, in the long run, it makes your IT infrastructure upgrade and maintenance cheaper and faster.

Types of Data Models

Types of Data Models: There are mainly three different types of data models: conceptual data models, logical data models, and physical data models, and each one has a specific purpose. The data models are used to represent the data and how it is stored in the database and to set the relationship between data items.

1. **Conceptual Data Model:** This Data Model defines **WHAT** the system contains. This model is typically created by Business stakeholders and Data Architects. The purpose is to organize, scope and define business concepts and rules.

2. **Logical Data Model:** Defines **HOW** the system should be implemented regardless of the DBMS. This model is typically created by Data Architects and Business Analysts. The purpose is to developed technical map of rules and data structures.
3. **Physical Data Model**: This Data Model describes **HOW** the system will be implemented using a specific DBMS system. This model is typically created by DBA and developers. The purpose is actual implementation of the database.



Conceptual Data Model

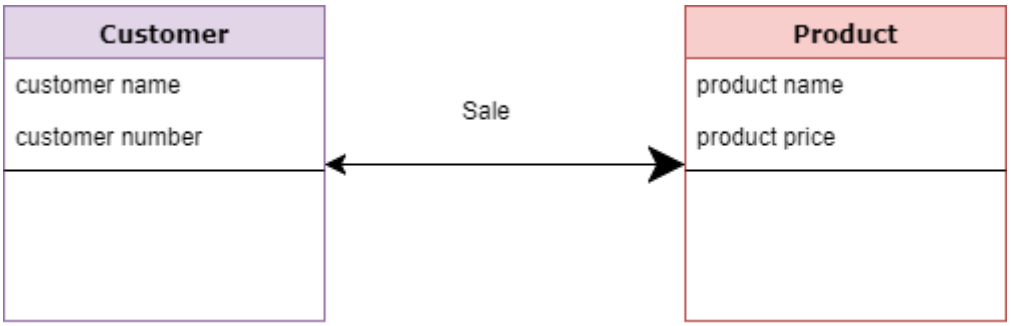
A **Conceptual Data Model** is an organized view of database concepts and their relationships. The purpose of creating a conceptual data model is to establish entities, their attributes, and relationships. In this data modeling level, there is hardly any detail available on the actual database structure. Business stakeholders and data architects typically create a conceptual data model.

The 3 basic tenants of Conceptual Data Model are

- **Entity**: A real-world thing
- **Attribute**: Characteristics or properties of an entity
- **Relationship**: Dependency or association between two entities

Data model example:

- Customer and Product are two entities. Customer number and name are attributes of the Customer entity
- Product name and price are attributes of product entity
- Sale is the relationship between the customer and product



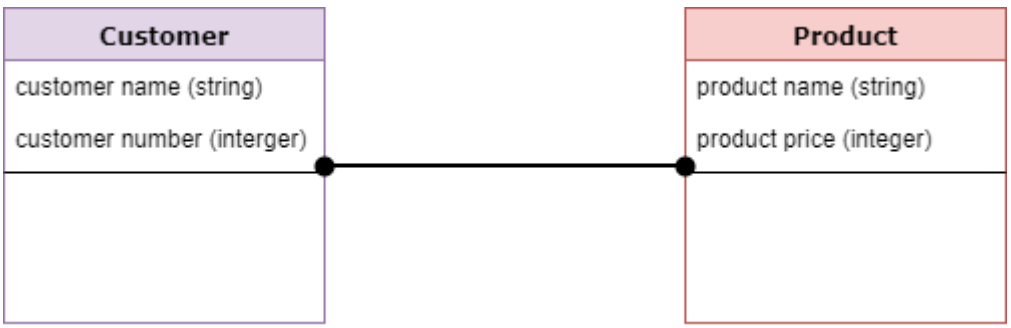
Characteristics of a conceptual data model

- Offers Organisation-wide coverage of the business concepts.
- This type of Data Models are designed and developed for a business audience.
- The conceptual model is developed independently of hardware specifications like data storage capacity, location or software specifications like DBMS vendor and technology. The focus is to represent data as a user will see it in the "real world."

Conceptual data models known as Domain models create a common vocabulary for all stakeholders by establishing basic concepts and scope.

Logical Data Model

The **Logical Data Model** is used to define the structure of data elements and to set relationships between them. The logical data model adds further information to the conceptual data model elements. The advantage of using a Logical data model is to provide a foundation to form the base for the Physical model. However, the modeling structure remains generic.



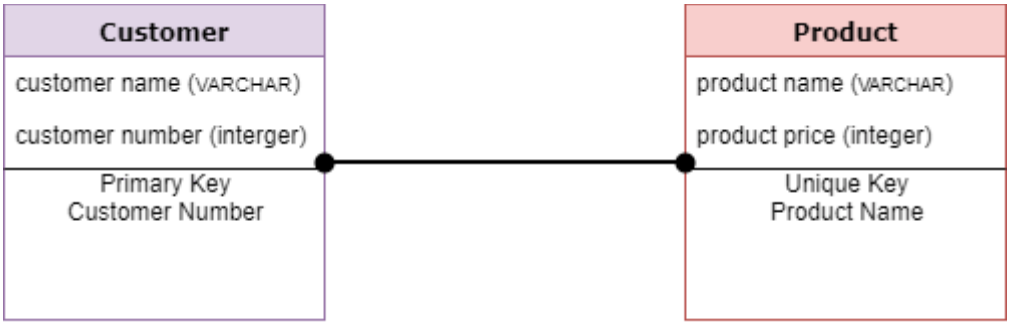
At this Data Modeling level, no primary or secondary key is defined. At this Data modeling level, you need to verify and adjust the connector details that were set earlier for relationships.

Characteristics of a Logical data model

- Describes data needs for a single project but could integrate with other logical data models based on the scope of the project.
- Designed and developed independently from the DBMS.
- Data attributes will have data types with exact precisions and length.
- Normalization processes to the model is applied typically till 3NF.

Physical Data Model

A **Physical Data Model** describes a database-specific implementation of the data model. It offers database abstraction and helps generate the schema. This is because of the richness of meta-data offered by a Physical Data Model. The physical data model also helps in visualizing database structure by replicating database column keys, constraints, indexes, triggers, and other RDBMS features.



Characteristics of a physical data model:

The physical data model describes data need for a single project or application though it may be integrated with other physical data models based on project scope. Data Model contains relationships between tables that which addresses cardinality and null ability of the relationships. Developed for a specific version of a DBMS, location, data storage or technology to be used in the project. Columns should have exact data types, lengths assigned and default values. Primary and Foreign keys, views, indexes, access profiles, and authorizations, etc. are defined.

Advantages and Disadvantages of Data Model:

Advantages of Data model:

- The main goal of a designing data model is to make certain that data objects offered by the functional team are represented accurately.
- The data model should be detailed enough to be used for building the physical database.
- The information in the data model can be used for defining the relationship between tables, primary and foreign keys, and stored procedures.

- Data Model helps business to communicate the within and across organizations.
- Data model helps to documents data mappings in ETL process
- Help to recognize correct sources of data to populate the model

Disadvantages of Data model:

- To develop Data model one should know physical data stored characteristics.
- This is a navigational system produces complex application development, management. Thus, it requires knowledge of the biographical truth.
- Even smaller change made in structure requires modification in the entire application.
- There is no set data manipulation language in DBMS.