UNIT 2 DATABASE IMPLEMENTATION AND TOOLS

Structure **Page Nos.**

2.0 INTRODUCTION

This unit provides a bridge between the database design principles and their implementation in an organisation. This unit first briefly discusses the role of database and information systems in an organisation. During this discussion we will also try to recollect what we have learnt about the development of information base. Thereafter, the steps of database design and the implementation of this design will be discussed. UML has now become one of the important tools in software system modeling and implementation. Databases are generally, an interim part of most systems. Thus, this unit also discusses UML based tools and their relationship to database design and implementation. Finally we will be introducing some of the automated database design tools that automatically implement the databases from the design.

2.1 OBJECTIVES

After going through this unit, you should be able to:

- define the role of database in an organisation's information system;
- relate database design process in an organisation to its implementation;
- relate UML diagram to database design and implementation, and
- identify some automatic design and implementation tools.

2.2 INFORMATION SYSTEM AND ORGANISATION

For any organisation, information provides the key to success. In the present day world, information technology and management of information resources is very important for the success of an organisation. It is said that an organisation today cannot last for more than 48-72 hours if its information resources have failed. The basis of information in any organisation is the data. The information and the data have the following properties:

- Information is one of the most important corporate resources. Data should be properly controlled and managed for the effective working of the organisation.

- The need of up-to-date data is ever increasing as more and more functions of organisations are being made electronic.
- The complexity of applications is increasing, as complex relationships among the data are needed to be modeled and maintained.
- The long lasting data may be cleaned, consolidated and archived for future decision making.
- Information is very useful for future planning. It provides the major input for supply chain management, enterprise resource planning, financial control and many other managerial functions of an organisation.
- For the day-to-day financial, retail good transactions, a publicly accessible and updatable operational database must be designed and made available for the transactions.
- Electronic selling and purchasing is cutting the cost of business transactions.

Information, which is generated from data, is a very important aspect for survival of an organisation. But, why is the Database Management System important for generating information for an organisation? The following are the major characteristics of database management that make it very valuable for an organisation.

- **Data Independence:** DBMS protects data base application programs from the changes in the logical database design and the physical database structure including access paths and storage structures.
- **Controlled Data Sharing:** DBMS allows the same data to be used for multiple applications with each application having its own **view** of the data. This data of views can also be suitably protected.
- **Simple Operation and Query Mechanism:** SQL provides a very simple style query language across the DBMS. Easy data handling, support for transactions, and security are the other major features of a DBMS. DBMS also has the extended feature of knowledge generation. Thus, DBMS are major tools for an organisation.

Thus, the database is typically a part of the information system. But, what are the activities required in an information system?

(a) Collection of data.
(b) Management of data using DBMS software.
(c) Use of the hardware and storage Media.
(d) Dissemination of the information Resources to the persons who use and manage the data. (Also known as DBA).
(e) The Application software that accesses and updates the data, and the application programmers.

We can easily conclude that the Database system is part of a much larger organisational information system.

Information System Development

The information system development follows the complete Software Development Life Cycle (SDLC). The information System Life Cycle is quite fuzzy for information systems where databases are a major integral component. What does it typically include? Well typically it includes all the basic phases involved in the waterfall model of SDLC.

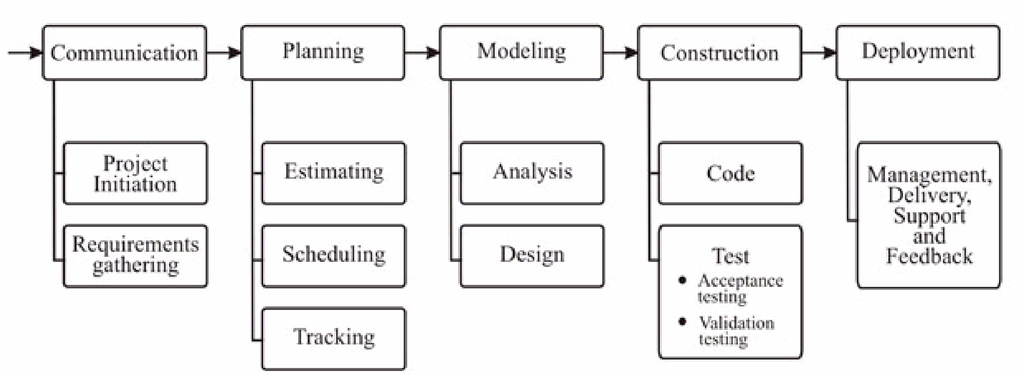


Figure 1: The information system life cycle

Let us define these phases briefly.

(1) Communication: This basically includes:

- Analysing potential application areas.
- Identifying the economics of information.
- Performing preliminary cost-benefit studies.
- Determining complexity of data and processes.
- Setting up priorities among applications.
- Gathering detailed requirements by interacting with users and user groups.-
- Identification of inter-application dependencies, communication and reporting procedures.

(2) Planning: This basically includes:

- Estimating the cost and time.
- Estimating the resource requirements.
- Identification of step-by-step (micro) procedure for each phase of SDLC.
- Scheduling of resources as per the needs.
- Tracking the resources as the project progresses.

(3) Modeling: Now let us see what we do in modeling:

- We create and refine analysis and design models.
- The information system is designed, it includes:
  - Design of the database system and the design of Application systems.
  - Design of information needs at different levels, system architecture design.
  - Forms and report design.
  - Design of Database Transactions and processes in detail.

(4) Construction: Construction mainly includes:

- Coding and implementing the system as per the detailed design.
- Creation of database and loading some sample data.
- Implementation of database transactions.
- Testing of all the above.
- Checking acceptability of the system in meeting user's requirements and performance criteria.
- The system is tested against performance criteria and behaviour specifications.

(5) Deployment: This is the last stage but necessarily an important one:

- Operational phase checks whether all system functions are operational and have been validated.
- New requirements and applications are validated and tested.
- Maintenance and monitoring of system performance is an important activity done here.

Thus, information system development is a major system development life cycle. However, in this unit we will focus our attention mainly on database application system development.

☞ Check Your Progress 1

1) Why is the DBMS most suitable for Information System implementation?
……………………………………………………………………………………
……………………………………………………………………………………
……………………………………………………………………………………

2) Which of the information System Life Cycle stages are important for database design?
……………………………………………………………………………………
……………………………………………………………………………………
……………………………………………………………………………………

2.3 DATABASE DESIGN AND IMPLEMENTATION

The database application design activity in an organisation is very important. This activity determines the overall quality of the information system.

The database application design involves database design and application system design. The *Figure 2* shows the steps of these processes and the linkages for database application design and implementation.

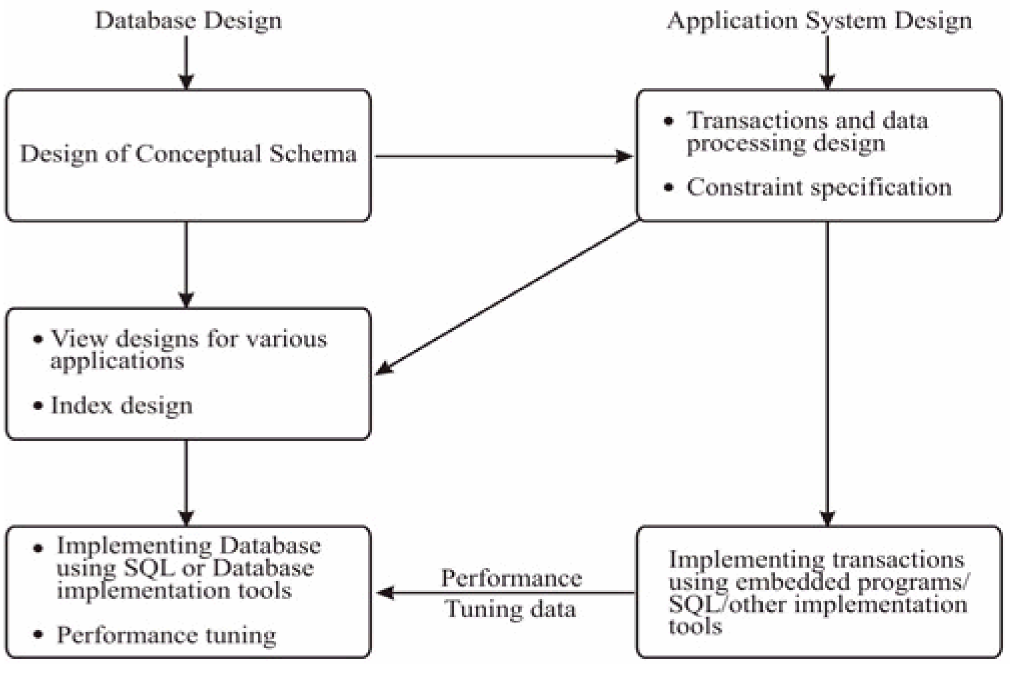

Database Design
Application System Design
Design of Conceptual Schema
• Transactions and data processing design
• Constraint specification
• View designs for various applications
• Index design
• Implementing Database using SQL or Database implementation tools
• Performance tuning
Performance Tuning data
Implementing transactions using embedded programs/ SQL/other implementation tools

Figure 2: Database application system design and implementation

2.3.1 The Database Application Life Cycle

The database application life cycle defines the phases of development of a database application. Let us list the activities that help us to create a database:

(1) **System definition:** In this activity, the scope of the database system, its users and its applications are defined with the help of users information. Other activities of this phase are:

- to find the interfaces for various categories of users.
- to find the response time constraints.
- to identify the storage and processing needs.

Before we discuss the further activities, it is necessary to know that this part of the database application design and implementation is part of the information system life cycle.

(2) **Design of Conceptual Database Application:** It involves design of the complete conceptual schema of the database and the transaction and Data processing. We will discuss it in more detail in the next sub-section.

(3) **Logical and Physical Database Design:** This involves creation of views, creation of storage structure, indexes etc. This is also discussed in more detail later.

(4) **Database Implementation:** This is the process by which we implement the following:

- Conceptual Database
- External Database
- Internal Database
- The mappings among these designs
- Creating empty database files
- Implementing the software applications.

(5) **Application Conversion:** Any software application from a previous system is converted to the new system. The data is populated either by loading the data directly or by converting existing files into the database system format.

(6) **Testing and Validations:** The new system, which is made needs to be tested and validated for good results. Testing and validation is done for this particular purpose.

(7) **Operations:** During this phase, the old as well as the new systems are operated in parallel.

(8) **Monitoring and Maintenance:** Change is an unchanging phenomenon. Similarly, modifications and reorganisation are unavoidable and may be needed from time to time. During this phase, the system is constantly monitored and maintained.

Let us now discuss the two major stages as above – viz., conceptual Database Application Design and Logical and Physical Design in more detail. Performance tuning is a specialised area and DBMS specific so this topic has not been discussed in this unit.

2.3.2 Conceptual Database Application Design

The process of conceptual database design is given in *Figure 3*.

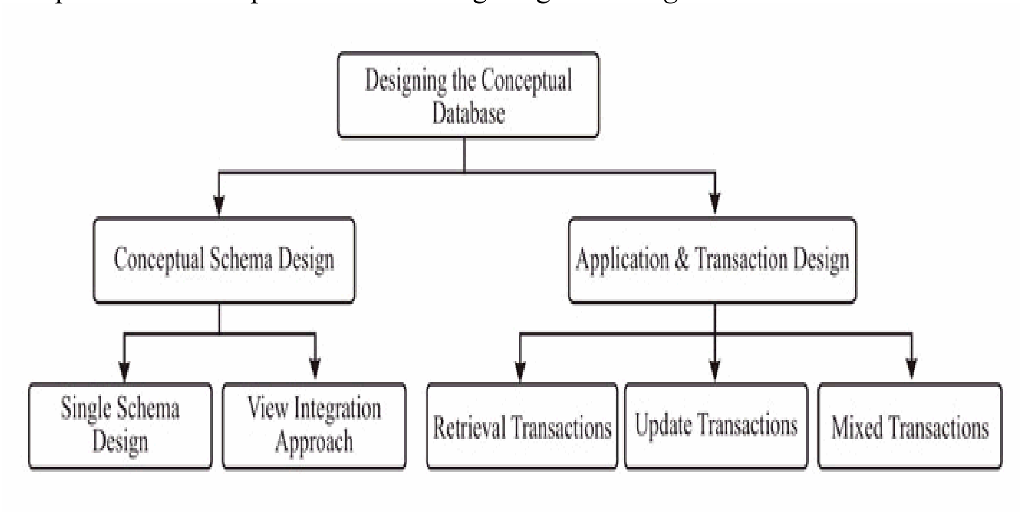


Figure 3: The conceptual database design

Conceptual Schema Design

The following details are produced during the conceptual database design:

- Database structures and data types associated with each field.
- Keys and constraints.
- Data interrelationships such as referential constraints
- Data dictionary.

But how do we do the conceptual schema design?

The process of conceptual schema design requires primarily modification and conversion of E-R diagram into tables keeping the following points in consideration:

- Proper identification of entities, relationship, and attributes
- Identification of key attributes
- Identification of cardinality of relationship
- Identifying weak entities, specialisation/generalisation, etc.

We have already discussed how an ER diagram is converted into tables in MCS-023 Block-1, Unit-2. Let us discuss the two common approaches for conceptual schema design:

(a) Single Schema Design

(b) View Integration

(i) Single Schema Design: This process results in the development of single schema. This process is explained with the help of an example:

Consider the following EER diagram of a University:

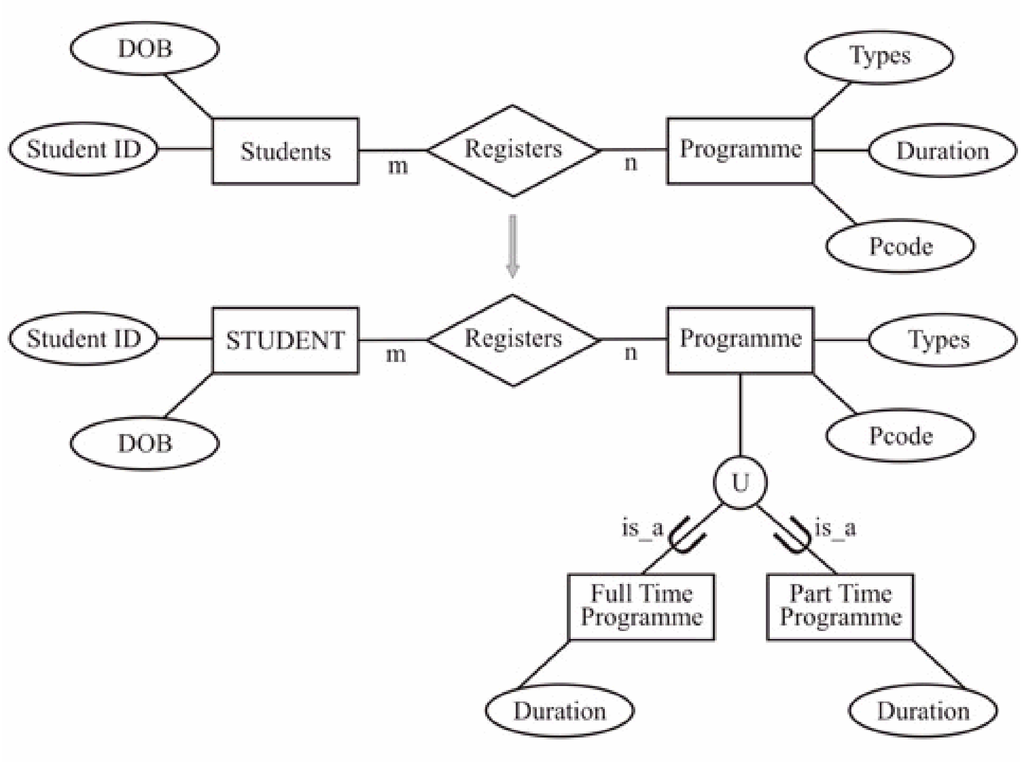


Figure 4: Design of database

Programme Types are: Full time or part time.

Please note the ER diagram is further refined to have an 'is-a' relationship. The simple conceptual tables can now be designed in such case as:

STUDENT (st-id, dob)
FULL-TIMEPROG (p-code, duration)
PART-TIMEPROG (p-code, duration)
REGISTERS (p-code, st-id)

Please note that the design above is not complete; a complete design would require all types defined and constraints identified. All such information is made available by the detailed analysis document.

(ii) **View Integration:** A commercial database may be very complex. It may be a good idea to model this database as per the viewpoints of various stakeholders, thus creating many views. A conceptual schema may be an integration of all such views. Such integration may have the following steps:

- Find out the common entities and relationships of interest.
- Find out if any conflicting requirements exist, such as name, type, constraints etc.
- Modify and manage the views.
- Refine the final model, if necessary.

We do not attempt a detailed discussion of this topic here, since it is beyond the scope of this unit.

Transaction Design

Transaction design specifies the functional requirements of a database system. The three types of transactions that may need to be designed are:

(i) **Retrieval Transaction:** These transactions will not update data in a database but help to access some information, for example, finding the status of a booking in a Railway reservation database. Such transactions can be executed concurrently without any problems.

(ii) **Updating Transactions:** Such transaction modifies some data value and requires careful concurrency control. For example, booking a ticket in a railway reservation system would update a lot of data items.

(iii) **Mixed Transactions:** A third type of transaction may have both these characteristics, that is, it reads some data and also updates some data. These are called **mixed transactions.**

The dataflow models created during the modeling phase help in identifying the transactions / procedure etc. on a database.

Selecting the desirable DBMS

Once we are through with conceptual design, we need to make a choice of the DBMS. Some of the factors on which the selection depends are:

(1) **Technical:** The technical issues we need to consider:

- Type of DBMS (Relational, object-relational etc.)
- Storage Structures and Access paths supported by DBMS.
- The simplicity of user/ programmer interface.
- Support for SQL.
- Availability of development Tools.
- Ability to interface with other DBMS like conformance to ODBC.
- Support for client-server architecture.

(2) **Non-Technical:** These include factors like cost and status of the organisation, which is supplying the product. The following cost factors may be considered while choosing DBMS:

- **Software Acquisition Cost:** It not only includes the cost of software, but also any utility cost, development tool cost or any other cost that the selling organisation may charge.
- **Additional Hardware Acquisition Cost:** If the DBMS requires additional memory, disk controllers, etc. then it must be included in the overall cost calculation.
- **Maintenance Cost:** The cost for maintaining and keeping the database perfect.
- **Operating Cost:** The cost related to continuous operation of the DBMS. It includes hiring cost of personnel employed only for DBMS such as DBA. It may also include the training cost.

2.3.3 Logical and Physical Database Design

The next phase aims to create an external schema and physical schema, including the mapping. Some of the important considerations at this stage of database design are:

- The database views should be identified.
- The security considerations and constraints on the view should be identified and duly specified.
- The performance criteria for views may be identified.
- The physical database design which includes specific storage structure (e.g., clustered file organisation) and access paths (indexes etc.) may be identified.

But why is physical database design required?

A high-performance *transactions processing systems* requires continuous operation of the system. Transactions performance, in terms of the average number of transactions per minute and maximum transaction response time, is critical. Hence, a careful physical database design that meets the organisation's transaction processing needs is a must.

Why do we choose a specific physical structure?

We do it so that the database applications can achieve good performance. The physical database design process is restricted to choosing the most appropriate structure for the database files from among the options offered by that DBMS. Let us discuss the criteria on which the choice of physical database design lie:

(1) **Response Time:** The elapsed time between the start of a database transaction to receive a response on it.

Please note that response time is also a factor of non-DBMS controlled issues like:

- Access Time of the database
- Present user Load
- Scheduling operation of operating system
- Communication Delays on a client-server system etc.

(2) **Space Utilisation:** This is the amount of storage space used by the database files including index. It also includes the storage of the physical database on the disk.

(3) **Transaction Throughput:** The average number of transactions that are processed per minute. It is measured under peak conditions on the system. This is very important in applications like Railway reservations or banking.

The outcome of the physical database design phase is an *initial* determination of storage structures and access paths for the database files. These structures do change in the lifetime of a database.

☞ Check Your Progress 2

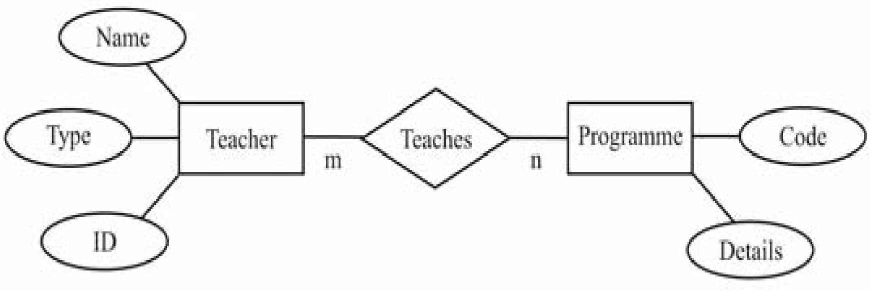
1) What are the steps for design of database system?

……………………………………………………………………………………

……………………………………………………………………………………

……………………………………………………………………………………

2) Consider the following EER diagram.



'Type' can be regular or visiting faculty. A visiting faculty members can teach only one programme. Make a suitable database application design for this.
………………………………………………………………………………

………………………………………………………………………………

………………………………………………………………………………

3) What are the criteria for physical design?
………………………………………………………………………………

………………………………………………………………………………

………………………………………………………………………………

2.4 UML DIAGRAMS: AN AID TO DATABASE DESIGN SPECIFICATIONS

First of all, before defining UML, we should first ask the question why UML? When there are so many database implementation tools/ languages available why has UML become so popular?

Let us consider this issue in detail.

Databases are an integral part of any information system and there exists many database implementation tools. Thus, the industry is in need of a standard approach that spawns all the phases of SDLC viz., requirement analysis, modeling, design, implementation and deployment. Also, the industry is looking for techniques to automate the production of software having an improved quality and reduced cost and time.

Since UML is accepted by the Object Management Group (OMG) as the standard for modeling object-oriented programs it becomes one of the automatic choices. Now the next question is why only UML? UML has quickly become one of the popularly used tools for modeling business and software application needs. The UML became popular due to the following reasons:

1) It is very flexible. It allows for many different types of modeling. Some of them include:

   - Business Process modeling event workflow,
   - Sequencing of events,
   - Defining database architecture etc.

2) It is language and platform - independent. It allows software architects to model any application, on any operating system in any programme language or network.

3) UML supports object-oriented methodologies. Please note however, that even if the concepts are based on object-oriented methods, the resulting structure and behaviour of UML can be used to design both relational and object-oriented models.

4) The integration of various SDLC stages through UML tools has brought the analysts, modeler, designers, and the software application developers closer to each other.

Types of UML Diagrams

You have studied UML diagrams in course MCS-032. Let us briefly recapitulate those concepts again. *Figure 5* defines the various types of UML diagrams.

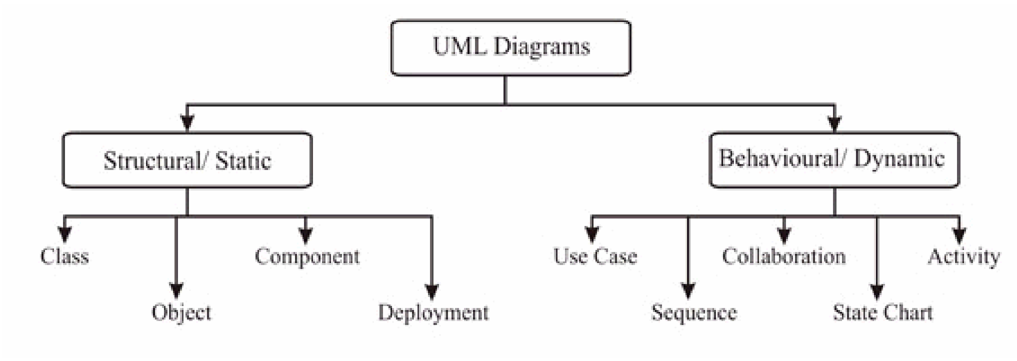


Figure 5: The UML diagrams

Structural/Static Diagrams: These diagrams are used to model the structural or static relationship among various objects or their components.

Class	Object	Component	Deployment
* backbone of almost every object-oriented method, including UML. * describes the static class structure of a system viz., classes, interfaces, dependencies etc.	* subset of class diagrams, sometimes treated as separate techniques. * organises elements of a system into related groups to minimise dependencies.	* describes the organisation of physical software components including source code, run time code and executables.	* depicts the physical resources in a system * includes nodes, components and connection.

Figure 6: The structural/static diagrams

Behavioral/dynamic Diagrams: These diagrams are used to model the behavioural or dynamic relationship among various objects or their components.

Use Case	Sequence	Collaboration	State Chart	Activity
* models the functionality of system using Actors and use cases.	* describes interaction between classes in terms of an exchange of messages over time.	* represents interactions between objects as a sequence of messages. * describes both static structure and dynamic behaviour	* describes the dynamic behaviour of a system in response to external stimuli. * useful in modeling objects where states are triggered by specific events.	* defines an activity representation operation on some class in the system that results in a change in the state of the system.

Figure 7: The behavioural/dynamic diagrams

Use of UML Diagrams: During the construction of an application system design we use all the major UML diagrams. But how are these diagrams useful for the database design.

Let us first discuss the behavioural/dynamic diagrams of UML. The use Case Diagrams are used to gather information during the requirement analysis phase. Likewise the sequence diagram is used to visualise the execution of the use cases, while the collaboration diagram defines the sequence of messages. The state Chart Diagram and activity diagram are used to graphically represent some of the states and activities of a system.

Thus, these diagrams are very useful for providing the behavioural or functional information about databases. Thus, they may be useful for transaction and processing design. For example, a use case diagram of a student of an on-line University may be:

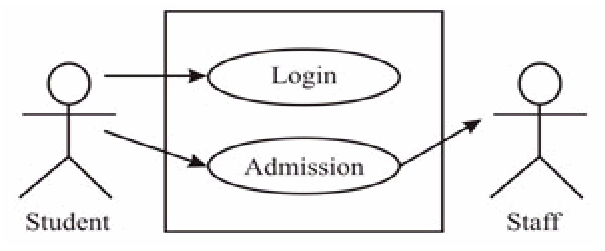


Figure 8: A simple use case diagram for a university

This use case diagram is indicating that the student and staff may have to login first, thus, the implementation of such a system would require a table for user-name and password and for taking admission a student may need to have a valid user name and password. It also indicates that the process of admission will be certified by a staff member, who will allow the student to take admission after due checking. Thus, UML behavioral diagrams may be useful to define processing/transactions. They also provide some information about entities and attributes. However, the UML diagram is closely associated with database structure design in the class diagram. Let us discuss these diagrams in more detail in respect of database design and implementation.

2.4.1 UML Class Diagrams and Database Design

To begin with, let us first recollect what we have learnt about class diagrams. The class diagrams sometimes may be considered as an alternative notation to E-R diagram. In UML class diagram contains:

- 'Class' – is displayed as a box. It has three sections:

  "Top Section" – Class Name
  "Middle Section" – Attributes
  "Last Section" – Operations
- Associations: Relationship Types are called Associations.
- Links: Relationship instances are called Links.
- Binary Association: Represented as a line connecting the participating classes.
- Linked Attribute: Connected to the Association's line by dashed line.
- Multiplicities: The (Minimum and Maximum) constraints.
- Reflexive Association: A Recursive Relationship.
- Aggregation: To represent a relationship between a whole object and its component parts.
- Different unidirectional and bi-directional Associations.

The specialisation/generalisation can be represented in UML class diagrams as:

- Basic notation is to connect the subclass by vertical lines to horizontal lines.
- A blank triangle indicates a specialisation /generalisation with the **disjoint constraint** and a filled triangle indicates an overlapping constraint.
- The root super class is known as the base class and the leaf nodes are called **terminal nodes**.

Let us explain the role of class diagrams with generalisation/specialisation hierarchy in a database structure design with the help of an example. Consider the following UML class diagram for a University.

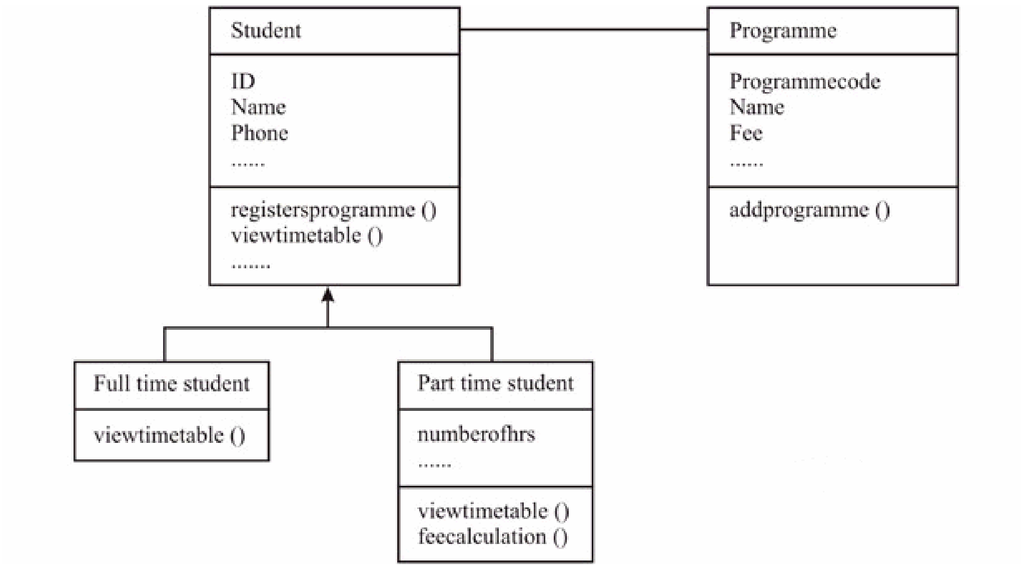


Figure 9: A UML class diagram

Please note that in the diagram above, you can make clear-cut table design. One such possible design may be:

STUDENT (ID, Name, Phone, type (PT, FT))
PT (ID, numberofhrs)
PROGRAMME (Programmecode, Name, Fee)
Stuprog (ID, Programmecode)

You can also identify several functions/data processing functions/ triggers relating to functions of classes. For example, feecalculation( ) may need to be implemented as a stored procedure while calculating the fee of part-time students, whereas addprogramme( ) is a simple data entry function. Thus, UML Class diagram with generalisation/specialisation is a very useful database design tool.

2.4.2 UML Based Design Tools

There are many tools used in the industry to develop information system. These tools provide the initial specification in UML that eventually leads to the database development. These tools provide support for conceptual, logical and physical database modeling design. Some of the features of these tools are:

- Support for Reverse Engineering: These tools allow the user to create a data model based on the already existing database structure.

- Forward Engineering and creation of database definitions: The tools can read the schema of the data models available in UML diagram – class diagram and create the database and the data storage model and generate appropriate DDL code for the same.

- Conceptual Design in UML notation: It allows modeling of databases using UML notations.

You must search for different tools and the facilities provided by them on the World Wide Web.

2.5 AUTOMATED DATABASE DESIGN AND IMPLEMENTATION TOOLS

Database design was initially handled manually. But as the complexity and volume of data increased, automation of the same became essential. So, why was Database design automated? The reasons are:

- As the complexity of a system increases, the number of options or in other words different design models that can represent the information model also increase rapidly. Hence it becomes difficult to deal with the complexity and the corresponding design alternatives manually.

- The size and volume of some databases may contain hundreds of entity types and relationship types. Manually managing such a design becomes extremely difficult.

Thus, because of the above two reasons, the Database Design Process had to be automated.

However, database design is not a process in isolation and as discussed earlier, database application design is part of SDLC. Thus, database design tools are part of tools or commercial database management system development environment.

Some of the essential facilities that need to be provided by the database design tools are:

(1) Facility for drawing a conceptual schema Design: It basically allows the database designers to represent the database diagrammatically. Thus, a tool must support the representation of the following:

- Entity Types
- Relationship Types
- Cardinality Constraints
- Attributes, Keys, foreign keys etc.
- Showing inheritance hierarchies or generalisation/specialisation hierarchy.

The diagrams should be stored and should allow modifications whenever necessary.

(2) Allows Model Mapping: Mapping Algorithms for external schemas, security etc. should be creatable and implementable. Such mapping is mostly system-specific.

(3) Support of Normalisation: Normalisation process uses a set of functional dependencies. Such FDs should be represented with the help of this tool at the time of logical design.

What should be the characteristics of good design tools?

A good design tool must have the following features:

(1) An easy Interface: Graphical user interfaces are commonly used by such tools so that the designer can concentrate more on complex high level abstraction and concepts rather than lower level implementations. Different interfaces may be tailored to suit beginners and expert designers.

(2) Analytical Capabilities: Such tools need to provide analytical capabilities for tasks that are difficult to perform manually. One such capability may be evaluating design alternatives. Such capabilities may also detect conflicting constraints among views. The designer should also be able to compare multiple alternatives. This helps in the process of physical design in a given system.

(3) Display of Design: Some of the features expected here are:

- diagrammatic display results such as schemas representation.
- well laid out diagrams (easily creatable automatically).
- easy to read design.
- the design should be automatically implemented as tables, lists, or reports.

(4) Support for Design Verification: The tool should enable verification of the resulting design. It should allow checking whether the design satisfies the initial requirements. We have avoided detailing any proprietary tool here, but you can refer to websites for many such tools.

The CASE tools address both schema design and application design concurrently.

☞ Check Your Progress 3

1) Which of the UML diagrams help in database schema design the most and why?

……………………………………………………………………………

……………………………………………………………………………

……………………………………………………………………………

2) What are the various UML based design tools?

……………………………………………………………………………

……………………………………………………………………………

……………………………………………………………………………

3) What are the features of automated database design and implementation tools?

……………………………………………………………………………

……………………………………………………………………………

……………………………………………………………………………

2.6 SUMMARY

In this unit, we have discussed the database implementation and tools, Information system and organisation, role of UML Diagrams in database design and Automated Database design tools.

The database management systems are used to implement the information system in an organisation. The Information systems are developed by following the complete SDLC phases. The database application design and implementation involves the conceptual database design, physical and logical design, procedural design and implementation. We have briefly reviewed the use of UML diagrams in different categories such as structural/static and behavioural/dynamic. We have also explained how the class diagrams with specialisation/generalisation hierarchies can be used to design database structure. We also gave an idea to the learner with different types of database design and implementation tools.

2.7 SOLUTIONS/ANSWERS

Check Your Progress 1

1) The following characteristics make DBMS a very valuable tool for information systems:

- Data independence
- Data sharing under secure environment
- Simple query mechanism
- Support for transaction and recovery
- Availability of tools for knowledge extraction.

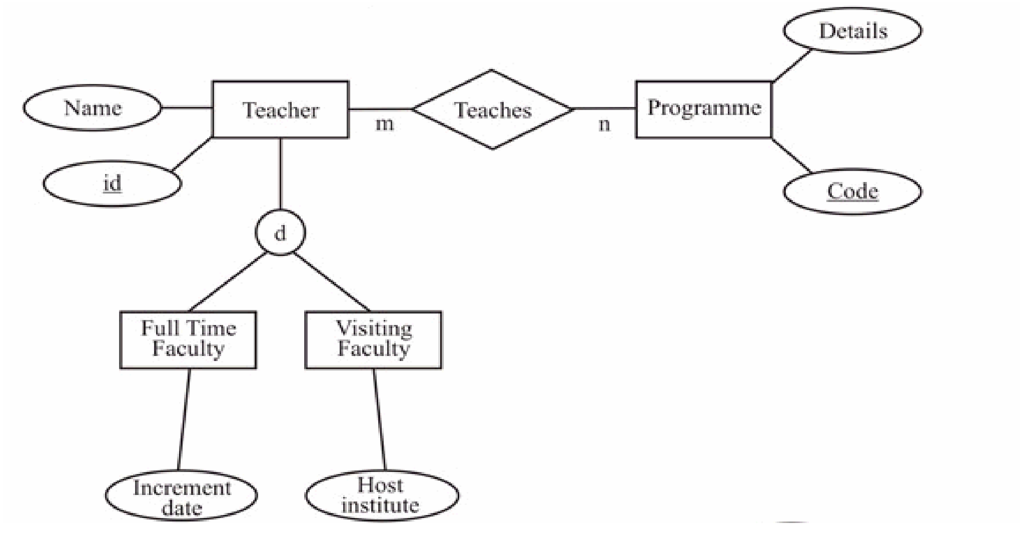
2) The stages where a lot of useful input is generated for database design are communication and modeling.

Check Your Progress 2

1) Design steps:

- Conceptual Schema Design
- Logical and physical schema design
- Data processing and transaction design.

2) The modified EER diagram is:



TEACHER (<u>id</u>, Name, Increment-date)
VISITING (<u>id</u>, Name, Host-institute)
TEACHES (<u>id, code</u>)
PROGRAMME (<u>code</u>, details)

Transaction/constraints: The **id** in TEACHES, if exists in id in visiting then COUNT on 'id' in teaches should not exceed 1.

3) The criteria are:

- Response time
- Space utilisation
- Transaction throughput.

Check Your Progress 3

1) The class diagram – it resembles ER diagram and also shows the possible transactions. Thus, it is useful for schema design as well as application design.

2) There are many open sources and proprietary UML based design tools available. You need to find them from a website.

3) They should have facilities for:

- Drawing conceptual schema
- Mapping implementation
- Normalisation
- Simple User interaction and displays
- Task analysis and
- Design verification.