Chapter-1

Project Team Specialization

- systems analyst
- Business Analyst
- Infrastructure Analyst
- Change Management Analyst
- Project Manager

The Systems Development Life Cycle (SDLC)

- Planning
  - **project initiation**
  - **project management**
- Analysis
  - **Analysis strategy**
  - **Requirements gathering**
  - **System proposal**
- Design
  - **Design Strategy**
  - **Architecture Design**
  - **Database and File Specifications**
  - **Program Design**
- Implementation
  - **System Construction**
  - **Installation**
  - **Support Plan**

Feasibility Analysis

- **Technical feasibility** (Can we build it?)
- **Economic feasibility** (Will it provide business value?)
- **Organizational feasibility** (If we build it, will it be used?)
- **Steps for Cost-benefit Analysis:**
- Identify Costs and Benefits
  - **Development costs (Salaries, Hardware and software expenses)**
  - **Operational costs (Salaries for operation staff, Software licensing fees)**
  - **Tangible benefits**
  - **Intangibles benefits**
- Assign Values to Costs and Benefits
- Determine Cash Flow
- Assess Project's Economic Value
  - **Determine Return on Investment**
  - **Determine Break-Even Point**
  - **Determine Net Present Value**

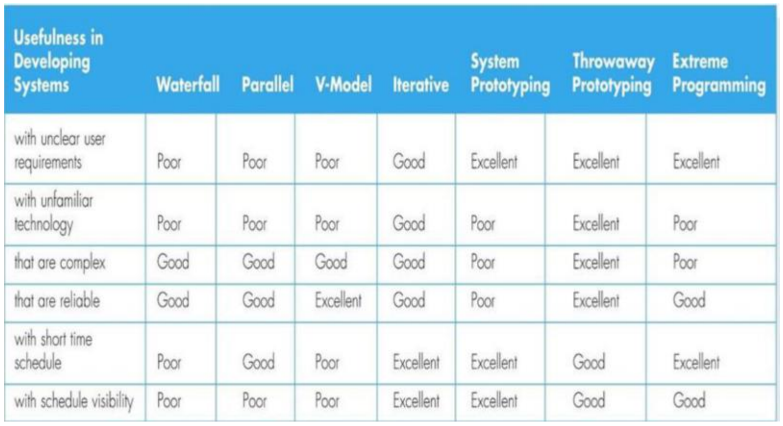
Chapter-2

How Do Projects Begin?

- ❖ Business needs should drive projects
- ❖ Project sponsor recognizes business need for new system and desires to see it implemented

Project Methodology Options

- Waterfall Development
- Parallel Development
- V-model (variation of the Waterfall Development
- Rapid Application Development (RAD)
- Iterative Development
- Agile Development



Usefulness in Developing Systems	Waterfall	Parallel	V-Model	Iterative	System Prototyping	Throwaway Prototyping	Extreme Programming
with unclear user requirements	Poor	Poor	Poor	Good	Excellent	Excellent	Excellent
with unfamiliar technology	Poor	Poor	Poor	Good	Poor	Excellent	Poor
that are complex	Good	Good	Good	Good	Poor	Excellent	Poor
that are reliable	Good	Good	Excellent	Good	Poor	Excellent	Good
with short time schedule	Poor	Good	Poor	Excellent	Excellent	Good	Excellent
with schedule visibility	Poor	Poor	Poor	Excellent	Excellent	Good	Good

Advantages for waterfall model:

- ❖ Simple and easy to understand and use.
- ❖ Easy to manage due to the rigidity of the model.
- ❖ Phases are processed and completed one at time.

Disadvantages for waterfall model:

- ❖ You cannot go back a step.
- ❖ High amounts of risk and uncertainty.
- ❖ Not a good model for complex and large project.
- ❖ Bad model for long and on-going projects

Advantages for prototype model:

- ❖ Reduced time and costs.
- ❖ Excellent for complex project.
- ❖ Improved and increased our involvement.
- ❖ Missing functionality can be identified easily

Disadvantages for prototype model:

- ❖ User confusion of prototype and finished system
- ❖ Developer misunderstanding of user objectives.
- ❖ Incomplete or inadequate problem analysis.

Handling Conflict IN in a project

A conflict is a situation where different stakeholders or team members have different interpretations of project requirements and want to adopt different methods to achieve them. The conflicts can be resolved by:

1. Clearly define plans for the project
2. make sure the team understands how the project is important to the organization;
3. develop detailed operating procedures and communicate these to the team members;
4. develop a project charter
5. develop schedule commitments ahead of time;
6. forecast other priorities and their possible impact on the projects

Timeboxing:

- Fixed deadline
- Reduced functionality, if necessary
- Fewer "finishing touches"

Timeboxing Steps

1. Set delivery date
    a. **Deadline should not be impossible**
    b. **Should be set by development group**
2. Prioritize features by importance
3. Build the system core
4. Postpone unfinished functionality
5. Deliver the system with core functionality
6. Repeat steps 3-5 to add refinements and enhancements

Managing Risk

- Risk assessment
- Actions to reduce risk
- Revised assessment

Classic Mistakes

- Overly optimistic schedule
- Failing to monitor schedule
- Failing to update schedule
- Adding people to a late project

Documentation

- Project binder
- Table of contents
- Continual updating

Standards

- Formal rules for naming files
- Forms indicating goals reached
- Programming guidelines

Motivation

- Use monetary rewards cautiously
- Use intrinsic rewards
    - **Recognition**
    - **Achievement**
    - **The work itself**
    - **Responsibility**
    - **Advancement**
    - **Chance to learn new skills**

Chapter-3

- The As-Is system is the current system and may or may not be computerized
- The To-Be system is the new system that is based on updated requirements
- The System Proposal is the key deliverable from the Analysis Phase
- Requirements determination is the single most critical step of the entire SDLC.

Requirement Types:

- Functional Requirements
  - A process the system has to perform
  - Information the system must contain
- Nonfunctional Requirements
  - Behavioral properties the system must have
  - Operational
  - Performance
  - Security
  - Cultural and political

models of identifying requirement

- Business Process Automation (BPA) **(small change)**
  - Goal: Efficiency for users
- Business Process Improvement (BPI) **(middle change)**
  - Goal: Efficiency and effectiveness for users
- Business Process Reengineering (BPR) **(big change)**
  - Goal: Radical redesign of business processes

Basic Process of Analysis (Determining Requirements)

- Understand the "As-Is" system
- Identify improvement opportunities
- Develop the "To-Be" system concept

REQUIREMENTS-GATHERING TECHNIQUES Interviews

- Interviews
- Joint Application Development (JAD)
- Questionnaires
- Document Analysis
- Observation

Planning .

- Work Plan
- Staffs who will work on the project
- Techniques to control and direct the project

2.Analysis

- System Proposal

Design

- Design Strategy
- Architecture Design
- Database and files Specification
- Program Design

Implementation

- System Construction
- Installation
- Support Plan

- **Logical process models** describe processes without suggesting how they are conducted
- **Physical process models** provide information that is needed to build the system

Timeboxing: is a technique that is used to organize a project when time is a critical issue. With timeboxing, a fixed deadline is established, and the project team prioritizes the functionality of the system so that the essential features are delivered within the set deadline. If some features must be omitted given that time frame, they are postponed to a later version of the system. With this technique, the users are assured of getting a system with essential functionality by the project deadline, and other, less essential features and refinements are added in later system versions.

Chapter-4

Use Case:

- Use cases are a text-based method of describing and documenting complex processes
- Use cases add detail to the requirements outlined in the requirement definition. Systems analysts work with users to develop use cases.

primary actor: is usually the trigger of the use case - the person or thing that starts the execution of the use case

Event-driven modeling: everything in the system is a response to some triggering event

Elements of a Use Case:

- **Name** (verb-noun phrase)
- **number** and brief **description**
- **Trigger**: event that causes the use case to begin
  - **External trigger**: some from outside the system
  - **Temporal triggers:** time-based occurrences
- Major **inputs** and **outputs**: Sources and destinations, Goal is to be all inclusive
- **Details**: Steps performed and the data inputs and outputs

Process of Developing Use Cases:

1. Identify the major use cases
2. Identify the major steps within each use case
3. Identify elements within steps
4. Confirm the use case
5. Cycle through the above steps iteratively

Use Case Diagram: summarizes all of the use cases for the part of the system being modeled together in one picture.

Steps in Creating the Use Case Diagram:

1. Identify Use Cases
2. Draw the system boundary
3. Place Use Cases on the diagram
4. Identify the actors
5. Add associations

Chapter-5

Process model: A formal way of representing how a business system operates

Data flow diagramming: A common technique for creating process models

Logical process models: describe processes without suggesting how they are conducted

Physical process models: provide information that is needed to build the system

Elements of a DFD:

- **Process:** An activity or function performed for a specific business reason
- **Data flow:** A single piece of data or a logical collection of data
- **Data Store:** A collection of data that is stored in some way, Data flowing out is retrieved and updates from and in the data store

- **External entity** A person, organization, or system that is external to the system but interacts with it.

Using a DFD to Define Business Processes:

- **Decomposition:** is the process of representing the system in a hierarchy of DFD diagrams
- **Balancing:** involves insuring that information presented at one level of a DFD is accurately represented in the next level DFD

levels of DFD:

Context Diagram

- First DFD in every business process
- Shows the context into which the business process fits
- Shows the overall business process as just one process (process 0)
- Shows all the external entities that receive information from or contribute information to the system

Level 0 Diagram

- Shows all the major processes that comprise the overall system – the internal components of process 0
- Shows how the major processes are interrelated by data flows
- Shows external entities and the major processes with which they interact
- Adds data stores

Level 1 Diagrams

- Generally, one level 1 diagram is created for every major process on the level 0 diagram
- Shows all the internal processes that comprise a single process on the level 0 diagram
- Shows how information moves from and to each of these processes
- If a parent process is decomposed into, for example, three child processes, these three child processes wholly and completely make up the parent process

Level 2 Diagrams

- Shows all processes that comprise a single process on the level 1 diagram
- Shows how information moves from and to each of these processes
- Level 2 diagrams may not be needed for all level 1 processes
- Correctly numbering each process helps the user understand where the process fits into the overall system

Logical process models describe processes without suggesting how they are conducted
Physical process models provide information that is needed to build the system

Steps in Building DFDs:

- Build the context diagram
- Create DFD fragments for each use case
- Organize DFD fragments into level 0 diagram 0
- Decompose level 0 processes into level 1 diagrams as needed, decompose level 1 processes into level 2 diagrams as needed
- Validate DFDs with user to ensure completeness and correctness

Chapter-6

Data model:

- ❖ A formal way of representing the data that are used and created by a **business system**
- ❖ Data models should balance with **process models**

Logical data model: shows the organization of data without indicating how it is **stored**, **created**, or **manipulated**.

Physical data model: shows how the data will actually be stored in **databases** or **files**

What Is an Entity Relationship diagram (ERD)?

An Entity Relationship Diagram (ERD) is a **visual representation** of different data using conventions that describe how these data are **related to each other**.

What Is the purpose and working of ERD?

- ERD symbols can show when one instance of an entity **must exist** for an instance of another or **related** to only one or many instances of another entity

ERD Elements:

- ❖ **Entity :**A person, place, event, or thing about which data is collected
- ❖ **Attributes:** Information captured about an entity and Attribute names are **nouns**
- ❖ **Identifiers:** One or more attributes can serve as the entity identifier, uniquely identifying each instance
- ❖ **Relationships**
  - o Associations between entities
  - o The first entity in the relationship is the **parent** entity; the second entity in the relationship is the **child** entity
  - o Relationships should have active **verb names**
  - o Relationships go in **both** directions
- ❖ **Cardinality:**
  - o refers to the number of times instances in one entity can be related to instances in another entity
    - ▪ One instance in an entity refers to only one instance in the related entity **(1:1)**
    - ▪ One instance in an entity refers to one or more instances in the related entity **(1: N)**
    - ▪ One or more instances in an entity refer to one or more instances in the related entity **(M: N)**
- ❖ **Modality:**
  - o Refers to whether or not an instance of a child entity can exist **without** a related instance in the parent entity
    - ▪ **Not Null** means that an instance in the related entity **must** exist for an instance in another entity to be valid
    - ▪ **Null** means that no instance in the related entity is **necessary** for an instance in another entity to be valid
- ❖ **Data Dictionary:** Metadata is information stored about components of the data model. Metadata is stored in the data dictionary so it can be shared by developers and users throughout the SDLC

Steps in Building ERDs

- Identify the **entities**
- Add **attributes** and assign identifiers
- Identify **relationships**

ERD Building Tips:

- **Data stores** of the **DFD** should correspond to **entities**
- Only include entities with **more than one instance** of information
- Don't include entities associated with implementation of the system

Entity Type:

- **Independent Entity:** Can exist without the help of another entity
- **Dependent Entity:** Relationships when a child entity does require attributes from the parent entity to uniquely identify an instance
- **Intersection Entity:** Exists in order to capture some information about the relationship that exists between two other entities

Normalization: is the process analysts use to **validate** data models.

- Technique used to validate data models
- Series of rules applied to logical data model to improve its organization
- Three normalization rules are common

Normalization levels:

- **First Normal Form (1NF)**
  - Look for **repeating** groups of attributes and remove them into separate entities.
- **Second Normal Form (2NF)**
  - If an entity has a concatenated identifier, look for attributes that depend only on **part** of the identifier. If found, remove to new entity.
- **Third Normal Form (3NF)**
  - Look for attributes that depend only on **another** non-identifying attribute.
  - If found, remove to new entity. Also remove any **calc`ulated** attributes

Balancing ERDs with DFDs:

- All analysis activities are **interrelated**
- Process models contain two data components
  - **Data flows** and **data stores**
- The DFD data components need to **balance** the ERD's data stores (**entities**) and data elements (**attributes**)

Chapter 7

Design phase Decide *how* to build the system and Create *system requirements* that describe technical details for building the system.

Architecture Design Document: Final deliverable from design phase and Conveys exactly what system the design team will implement during the implementation phase.

Classical Design Mistakes: Reducing design time - Feature creep - Silver bullet syndrome - Switching tools in mid-project.

Design Strategies: (مُهم جداً)

- Custom development (build from scratch) in-house
- Purchase software package (and customize it)
- Outsource development to third party

1- Custom Development

PROS

- Allows flexibility and creativity
- Consistent with existing technology and standards
- Builds technical skills and functional knowledge in-house

CONS

- Requires significant time and effort
- May exacerbate existing backlogs
- May require missing skills
- Often costs more
- Often takes more calendar time
- Risk of project failure

2- Packaged Software

Available for many common business needs Tested, proven; cost and time savings. And rarely a perfect fit with business needs.

May allow for customization:

- Manipulation of system parameters
- Changing way features work
- Synchronizing with other application interfaces
- May require workarounds

3- Outsourcing

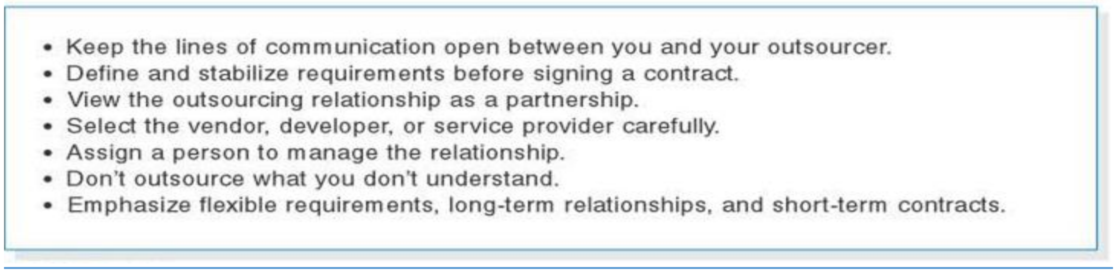
Hiring an external vendor, developer, or service provider and may reduce costs or add value.

Risks include possibly

- Losing confidential information
- Losing control over future development
- Losing learning opportunities

Outsourcing Contracts Time and arrangements , Fixed-price , Value- added

Outsourcing Guidelines (مهم جداً)



- Keep the lines of communication open between you and your outsourcer.
- Define and stabilize requirements before signing a contract.
- View the outsourcing relationship as a partnership.
- Select the vendor, developer, or service provider carefully.
- Assign a person to manage the relationship.
- Don't outsource what you don't understand.
- Emphasize flexible requirements, long-term relationships, and short-term contracts.

Selecting a Design Strategy Consider each of the following when deciding what strategy to use:

- Business need
- In-house experience
- Project skills
- Project management
- Time frame

وضعت الاستراتيجيات فقط الأفضل العودة الى الشابتر لقراءة المعلومات التى تتعلق بكل استراتيجية

Request for proposal (RFP) a document that solicits a formal proposal from a potential vendor, developer, or service provider.
Request for quote (RGQ) may be used when a list of equipment is so complete that the vendor only need provide a price.

Developing an Alternative Matrix

- What tools and technologies are needed for a custom development project?
- What vendors make products that address the project needs?
- What service providers would be able to build this application if outsourced?
- Combine several feasibility analyses into one matrix
- Include technical, budget, and organizational feasibilities
- Assign weights to indicate the relative importance of the criteria
- Assign scores to indicate how well the alternative meets the criteria

Chapter 8

Architecture design Plans for how the system will be distributed across computers and what the hardware and software will be used for each computer.

Hardware and software specification Describes the hardware/software components in detail to aid those responsible for purchasing those products.

Architectural Components (Functions) of Software (مُهمة جداً)

- Data storage
- Data access logic - Processing required to access stored data
- Application logic - Processing logic of the application
- Presentation logic - Information display and user command processing

Architectural Design Purpose: Determine what parts of the application software will be assigned to what hardware.

Hardware options:

- ✓ **Clients:** Input/output devices employed by users. PCs, laptops, handheld devices, cell phones
- ✓ **Servers:** Larger computers storing software accessible by many users

The three fundamental computing architectures

- server-based
- client-based
- client-server based.

Server-Based Architectures

- Mainframe
- Minicomputer
- Microcomputer (personal computer

Client-Based Architectures

- Terminals
- Microcomputer (personal computer)
- Special purpose terminals (ATMs, kiosks, Palm Pilots, and many others)

Client-Server Architectures

- Server-based Architecture
- Client-based Architecture
- Client-server based Architecture

Client-Server Attributes

- ✓ **Benefits**
- Scalable
- Works with multiple vendors/products through middleware
- Improved modularity of web-based systems
- No central point of failure
- ✓ **Limitations**
- Complexity
- New programming languages and techniques (adds stress for personnel)
- More complex to update

N-Tiered versus 2-Tiered Client-Server Architectures

- ✓ **Benefits**
  - Separates processing to better balance load on different servers
  - More scalable
- ✓ **Limitations**
  - Greater load on the network
  - More difficult to program and test

Selecting an Architecture Design

- ✓ **Recommended selection process:**
  - Expand nonfunctional requirement details
  - Base architecture selection on the detailed nonfunctional requirements

Chapter 9

The user interface defines how the system will interact with external entities
The system interfaces define how systems exchange information with other systems
The navigation mechanism provides the way for users to tell the system what to do
The input mechanism defines the way the system captures information
The output mechanism defines the way the system provides information to users or other systems
Graphical user interface (GUI) is the most common type of interfaces most students are likely to use personally and for developing systems.

Principles for User Interface Design (مهم جداً)

- Layout
- Content awareness
- Aesthetics
- User experience
- Consistency
- Minimize user effort

Layout Concepts:

The screen is often divided into three boxes

- Navigation area (top)
- Status area (bottom)
- Work area (middle)

Information can be presented in multiple areas and the Like areas should be grouped together.
Areas and information should minimize user movement from one to another Ideally, areas will remain consistent in: Size – Shape - Placement for entering data - Reports presenting retrieved data.

Content Awareness All interfaces should have titles
Menus should show: **where you are, where you came from to get there**
It should be clear what information is within each area.
Fields and field labels should be selected carefully.
Use dates and version numbers to aid system.

Aesthetics

- Interfaces need to be functional and inviting to use
- Avoid squeezing in too much, particularly for novice users
- Design text carefully
- Be aware of font and size
- Avoid using all capital letters
- Colors and patterns should be used carefully.

Consistency

- Enables users to predict what will happen
- Reduces learning curve
- Considers items within an application and across applications

Pertains to many different levels

- Navigation controls
- Terminology
- Report and form design

Three clicks rule Users should be able to go from the start or main menu of a system to the information or action they want in no more than three mouse clicks or three keystrokes.

User Interface Design Process.

- Interface evaluation
- Interface design prototyping
- Interface structure design
- Interface standards design
- Use scenario development

وضعت العمليات فقط الأفضل العودة الى الشابتر لقراءة بعض المعلومات المتعلقة بكل علمية.

HTML Prototype Built with the use of Web pages created in HTML
The user uses HTML to create a series of Web pages that show the fundamental parts of the system. And the users have the ability to interact with the pages by clicking on buttons and entering pretend data.

Basic Principles of Navigation Design

- ✓ **Assume users**
- Have not read the manual
- Have not attended training
- Do not have external help readily at hand
- ✓ **Prevent mistakes**
- Limit choices
- Never display commands that can’t be used (or “gray them out”)
- Confirm actions that are difficult or impossible to undo

Types of Navigation Control

- Languages: Command language , Natural language
- Menus: Generally aim at broad shallow menu , Consider using "hot keys"
- Direct Manipulation Used with icons to start programs and Used to shape and size objects and May not be intuitive for all commands

Message Tips Should be clear, concise, and complete, Should be grammatically correct and free of jargon and abbreviations (unless they are the users) , Avoid negatives and humor .

Basic Principles of Input Design

- The goal is to simply and easily capture accurate information for the system
- Reflect the nature of the inputs
- Find ways to simplify their collection

Online processing immediately records the transaction in the appropriate database
Batch processing collects inputs over time and enters them into the system at one time in a batch.
Batch processing simplifies data communications and other processes, master files are not updated real time

Capture Data at the Source

- Reduces duplicate work
- Reduces processing time
- Decreases cost
- Decreases probability of error

Source Data Automation

Can be obtained by using the following technologies:

- ✓ bar code readers
- ✓ optical character recognition
- ✓ magnetic stripe readers
- ✓ smart cards
- ✓ RFID (radio frequency identification tags)

Minimize Keystrokes Never ask for information that can be obtained other ways

- ✓ Lookups
- ✓ Dropdown lists
- ✓ Default values

Types of Inputs

Data items linked to fields

- ✓ Text
- ✓ Numbers
- ✓ Selection boxes :

- Check boxes
- Radio buttons
- On-screen list boxes
- Drop-down list boxes
- Combo boxes
- Sliders

Chapter 10

Program design - creating instructions for the programmers.
***The top-down, modular approach - begin with the "big picture" and gradually add detail**
Program design document – all structure charts and specifications needed by programmers to implement the system.
Analysis phase – focus on logical processes and data flows
Design phase – create physical process models showing "how" the final system will work
Physical process models convey the "system view" of the new system
***The physical DFD contains the same components as the logical DFD, and the same rules apply**

There are five steps to perform to make the transition to the physical DFD

- Add implantation references
- Draw a human machine boundary
- Add system related data store , data flow , process
- Update the data element in data flow
- Update the metadata in the CASE repository

Designing Programs

- Resist temptation to write code to quickly
- System quality is enhanced with top-down, modular design
- Program design document is the final deliverable for this task

The Structure Chart Important program design technique Shows all components of code in a hierarchical format.

- ✓ Sequence
- ✓ Selection
- ✓ Iteration

Building the Structure Chart
Processes in the DFD tend to represent one module on the structure chart

- Afferent processes – provide inputs to system
- Central processes – perform critical system operations
- Efferent processes – handle system outputs

Types of Structure Charts –
Transaction structure: control module calls subordinate modules, each of which handles a particular transaction

- ✓ Many afferent processes
- ✓ Few efferent processes
- ✓ Higher up levels of structure chart
- ✓ Using inputs to create a new output

Steps in Building the Structure Chart:

- Identify top level modules and decompose them into lower levels
- Add control connections
- Add couples
- Review and revise again and again until complete

Design Guidelines

- High quality structure charts result in programs that are modular, reusable and easy to implement.
- Measures include:
- ✓ Cohesion
- ✓ Coupling
- ✓ Appropriate levels of fan-in and fan-out

Factoring

- Process of dealing with "low" cohesion
- Separates tasks into different modules
- Reduces use of control flags

Quality Checklist

- Library modules have been created where ever possible
- The diagram has a high fan-in structure
- Control modules have no more than 7 subordinates
- Each module performs only one function (high cohesion)
- Modules sparingly share information (loose coupling)
- Data couples that are passed are actually used by the accepting module
- Control couples are passed from "low to high"
- Each module has a reasonable amount of code associated with it

Program Specifications Content

- No standard approach
- Include program information
- Note events that trigger actions
- List inputs and outputs
- Include pseudocode
- Present additional notes and comments

Chapter 11

The data storage function manages how data is stored and handled by programs that run the system.

Goals of data storage design

- ✓ Efficient data retrieval (good response time)
- ✓ Access to the information users need

Types of Data Storage Formats:

- **Files:** electronic lists of data optimized to perform a particular transaction
- **Database:** a collection of groupings of information the relate to each other in some way.
- **A Database Management System (DBMS)** is software that creates and manipulates databases

File Attributes

- Files contain information formatted for a particular transaction
- Typically organized sequentially
- Pointers used to associate records with other records
- Linked Lists are files with records linked together using pointers

File Types

- **Master files** – store core, important information
- **Look-up files** – store static values
- **Transaction files** – store information that updates a master file
- **Audit files** – record before and after versions of data
- **History (archive) files** – store past information

Database Types

- Legacy database
- ✓ Hierarchical (depict parent-child relationships using inverted trees)
- ✓ Network (depict nonhierarchical associations using pointers)
- Relational database
- Object database
- Multidimensional database

Relational Database Concepts

- Popular; easy for developers to use **Primary and foreign keys** used to identify and link tables
- Referential integrity ensures correct and valid table synchronization
- **Structured Query Language (SQL)- standard language for accessing data**

Object Database Concepts Built around objects consisting of both data and processes and objects are encapsulated (self-contained)

Object classes – major object categories

OODBMS – used primarily for applications with multimedia or complex data

Hybrid OODBMS – both object and relational features

Multidimensional Database Concepts

- Stores data for easy aggregation and manipulation across many dimensions
- Used for data warehouses and data marts
- Summary data is pre-calculated and stored for fast access

Selecting a Storage Format

- ✓ **Type of Application System:**

the best choices for these systems usually are relational databases and multidimensional databases as the formats can be configured.

- ✓ **Existing Storage Formats:**

Data storage format should be selected primarily on the basis of the kind of data and application system being developed.

The Physical Entity Relationship Diagram: Five Steps

- Change Entities to Tables or Files
- Change Attributes to Fields
- Add Primary Keys
- Add Foreign Keys
- Add System-Related Component

There are several techniques that the project team can use to try to speed up access to data:

- Denormalization
- Clustering
- Indexing
- Estimating the size of data for hardware planning purposes

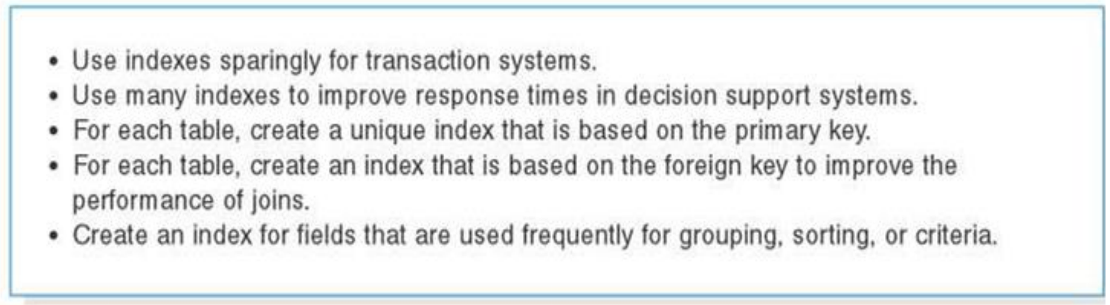
Clustering:

reduce the number of times storage must be accessed by physically placing like records close together.

- ✓ **Intrafile clustering** – similar records in a table are stored together
- ✓ **Interfile clustering** – combine records from more that one table that are typically retrieved together

Indexing: A minitable that contains values from one or more fields in a table and the location of the values within the table Similar to the index of a book.

Guidelines for Creating Indexes(مهم جداً)



- Use indexes sparingly for transaction systems.
- Use many indexes to improve response times in decision support systems.
- For each table, create a unique index that is based on the primary key.
- For each table, create an index that is based on the foreign key to improve the performance of joins.
- Create an index for fields that are used frequently for grouping, sorting, or criteria.

Volumetrics – Estimating Data Storage Size

- ✓ Raw data – sum of the average widths of all fields in a table.
- ✓ Calculate overhead requirements based on DBMS vendor recommendations
- ✓ Estimate initial number of records
- ✓ Estimate growth rate of records

تم بحمد الله..........دعواتكم