**Database and Data Modeling in Business Systems**

Key Concepts:

- During analysis, data models are drawn to organize data without focusing on storage or flow.

- Entity Relationship Diagram (ERD) helps validate these models.

Data Modeling Stages:

1. Conceptual: High-level, focuses on entities and relationships.

2. Logical: Defines data structure in detail (e.g., attributes, keys).

3. \*\*Physical\*\*: Implements the model with data types, indexes, etc.

**Conceptual Model Components**:

- Entities: Objects or persons in the system (e.g., CUSTOMER, ORDER).

- Attributes: Information about entities (e.g., Name, BirthDate).

- Relationships: Define how entities are linked.

**Attribute Types:**

- Simple Attribute: Single component (e.g., `SSN`, `Gender`).

- Composite Attribute: Composed of multiple components (e.g., `Address` = [Street, City]).

- Multi-Valued Attribute: Can have multiple values (e.g., multiple phone numbers).

**Logical Model Components**:

- Entity: Represented as a person, place, or thing about which data is collected.

- Attributes: Include unique identifiers (Primary Key), concatenated identifiers, and foreign keys.

**Data Dictionary (Metadata):**

- Describes the entities, attributes, and relationships in an ERD.

- Helps define the structure of the system.

**Creating an ERD**:

1. \*\*Identify Entities\*\*: Major categories of information.

2. \*\*Add Attributes\*\*: Assign necessary details to each entity.

3. \*\*Draw Relationships\*\*: Define how entities are connected (one-to-one, one-to-many, many-to-many).

**Physical Model Components**:

- \*\*Data Types\*\*: For attributes (e.g., `VARCHAR`, `INTEGER`).

- \*\*Primary & Foreign Keys\*\*: Establish unique IDs and relationships.

- \*\*Relationships\*\*: Include cardinality (e.g., 1-to-1, 1-to-Many).

### Special Entity Types:

1. \*\*Independent Entity\*\*: Exists without another entity.

2. \*\*Dependent Entity\*\*: Requires attributes from the parent entity for unique identification.

#**Summary**:

- \*\*Entity\*\*: People, places, or things.

- \*\*Attribute\*\*: Information about entities.

- \*\*Relationship\*\*: Associations between entities.

**Steps to Create an ERD:**

1. Identify entities.

2. Add attributes.

3. Draw relationships and assign cardinality.

### Designing User and System Interfaces

#### 1. \*\*Understanding User and System Interfaces\*\*

- \*\*User Interface (UI)\*\*: The interaction between a user (actor) and a system.

- \*\*System Interface\*\*: The part of a system that connects to other systems, often with minimal human intervention.

#### 2. \*\*Metaphors of Human-Computer Interaction\*\*

- \*\*Direct Manipulation\*\*: Users interact with graphic symbols or objects (e.g., icons).

- \*\*Desktop Metaphor\*\*: Visual display is like a workspace with tools/icons around the edges.

- \*\*Document Metaphor\*\*: Data is displayed as pages or forms.

- \*\*Dialog Metaphor\*\*: Interaction between user and system mimics a conversation via text, voice, or buttons.

#### 3. \*\*User Interface Design Concepts\*\*

- Focus on making user interaction efficient and effective by ensuring \*\*visibility\*\* and \*\*affordance\*\* (controls must be visible and intuitive).

#### 4. \*\*User Interface Design Guidelines\*\*

- \*\*Consistency\*\*: Maintain uniformity in design elements.

- \*\*Shortcuts\*\*: Provide users with efficient ways to perform actions.

- \*\*Feedback\*\*: Inform users of the system's state.

- \*\*Dialog Closure\*\*: Ensure tasks are clearly concluded.

- \*\*Error Handling\*\*: Offer guidance when errors occur.

- \*\*Reversal of Actions\*\*: Allow easy undoing of actions.

- \*\*Memory Load\*\*: Minimize the need for users to remember details between steps.

#### 5. \*\***Dialog Design**\*\*

- Create a \*\*natural flow\*\* between user and system.

- Use \*\*natural language\*\* in interactions.

- Develop a \*\*storyboard\*\* to map out the dialog sequence, and review it with users for validation.

#### 6. \*\*System Interfaces\*\*

- \*\*System Inputs/Outputs\*\*: Should operate with minimal human involvement.

- \*\*Automated Inputs\*\*: Use electronic data sources instead of manual entry whenever possible.

- \*\*External Databases\*\*: Systems may interface with external databases for input or output.

#### 7. \*\*Designing System Inputs\*\*

- \*\*Objective\*\*: Ensure error-free input by minimizing human involvement and leveraging electronic devices (e.g., barcode readers, RFID).

- \*\*Examples\*\*: Use devices like magnetic strip readers, touchscreens, and speech recognition systems to capture data.

#### 8. \*\*Designing System Outputs\*\*

- \*\*Types of Reports\*\*:

- \*\*Detailed Reports\*\*: Contain specific business transaction data.

- \*\*Summary Reports\*\*: Provide summaries of detailed data.

- \*\*Exception Reports\*\*: Highlight errors or unusual transactions.

- \*\*Executive Reports\*\*: High-level summaries for managers.

- \*\*Outputs\*\*:

- \*\*Internal\*\*: Used within the organization.

- \*\*External\*\*: Provided to external stakeholders.

- \*\*Turnaround Documents\*\*: Sent out and returned with new data (e.g., utility bills).

- \*\*Graphical Outputs\*\*: Charts and graphs to visualize data.

### Conclusion:

- \*\*User Interface\*\* focuses on user-system interaction and efficiency.

- \*\*System Interfaces\*\* deal with automated interactions between systems, minimizing human input.

- The goal is to design both inputs and outputs to be error-free and efficient while providing clear, actionable information to users.

### 1. \*\*Managing the Programming Process\*\*

- \*\*Goal\*\*: Assign tasks, manage schedules, and handle delays.

- \*\*Common Issues\*\*: Scope creep (adding features), slippage (delays).

### 2. \*\*Testing\*\*

- \*\*Types\*\*:

- \*\*Unit\*\*: Tests individual modules.

- \*\*Integration\*\*: Tests module interaction.

- \*\*System\*\*: Ensures the system meets requirements.

- \*\*Acceptance\*\*: Client approves the system (Alpha, Beta testing).

### 3. \*\*Documentation\*\*

- \*\*System Documentation\*\*: For developers (design specs, code).

- \*\*User Documentation\*\*: For users (guides, tutorials).

### 4. \*\*Types of Documentation\*\*

- \*\*Online\*\*: Easier to update, searchable.

- \*\*Printed\*\*: Costly but sometimes required.

### 5. \*\*Installation Methods\*\*

- \*\*Direct\*\*: Immediate switch.

- \*\*Parallel\*\*: Both systems run together.

- \*\*Phased\*\*: Introduce in steps.

- \*\*Pilot\*\*: Test in one area first.

### 6. \*\*Training\*\*

- \*\*Types\*\*: Class training, one-on-one, online tutorials, help desk support.

- \*\*Focus\*\*: System use, troubleshooting, help resources.

### 7. \*\*Support & Maintenance\*\*

- \*\*Support\*\*: Help desk handles issues.

- \*\*Maintenance\*\*: Fixes, updates, performance improvements.

### 8. \*\*Change Management\*\*

- \*\*Goal\*\*: Smooth user adoption.

- \*\*Activities\*\*: User involvement, communication, training, support.

### 9. \*\*Conversion Strategies\*\*

- \*\*Direct Cutover\*\*: Immediate replacement.

- \*\*Parallel\*\*: Both systems run together.

- \*\*Phased\*\*: Gradual rollout.

- \*\*Pilot\*\*: Test in one location first.

### 10. \*\*Post-Implementation Review\*\*

- \*\*Goal\*\*: Evaluate success, user satisfaction, issues, improvements.

- \*\*System Design and Requirements:\*\* Involves deciding how the system will be built and documenting the requirements.

- \*\*Three Strategies for System Creation:\*\*

1. Custom development.

2. Buy packaged software.

3. Outsource to a service provider.

### 2. \*\*System Acquisition Strategies:\*\*

- \*\*Custom Development:\*\*

- \*\*Pros:\*\*

- Flexible and creative design.

- Leverages current technology.

- Enhances in-house skills.

- \*\*Cons:\*\*

- Requires significant effort and skilled professionals.

- High risks associated with development from scratch.

- \*\*Packaged Software:\*\*

- \*\*Pros:\*\*

- Efficient, pre-built, and tested.

- Quick implementation.

- \*\*Cons:\*\*

- Limited by the provided functionality.

- May require customization or workarounds.

- \*\*Outsourcing:\*\*

- \*\*Pros:\*\*

- Lower initial costs and faster setup.

- \*\*Cons:\*\*

- Risks include data security, loss of control, and losing internal expertise.

### 3. \*\*Outsourcing Contracts:\*\*

- \*\*Time and Materials:\*\* Pay for the resources used.

- \*\*Fixed-Price:\*\* Predetermined cost for a specific scope.

- \*\*Value-Added:\*\* Based on performance or success metrics.

### 4. \*\*Influences on Acquisition Strategy:\*\*

- \*\*Business Need:\*\*

- Common needs favor packaged solutions.

- Unique needs warrant custom development.

- \*\*In-House Experience:\*\*

- If expertise exists, custom development is viable.

- Lack of technical skills makes packaged software or outsourcing better options.

- \*\*Project Skills:\*\* Either technical (e.g., SQL) or functional (e.g., e-commerce) expertise is necessary.

- \*\*Project Management:\*\* Custom solutions need robust management; packaged solutions require vendor coordination; outsourcing needs strong project agreements.

- \*\*Time Frame:\*\* Packaged software is faster; custom development can be time-boxed; outsourcing can be slow.

### 5. \*\*Selecting an Acquisition Strategy:\*\*

- Consider available tools, vendors, and service providers.

- Use a \*\*Request for Proposal (RFP)\*\* to get formal bids from vendors or developers.

### 6. \*\*Summary:\*\*

- Transition from requirements to design.

- Overview of system acquisition strategies.

- Key influences on choosing the right strategy.

### 1. \*\*Elements of an Architecture Design:\*\*

- \*\*Software and Hardware Assignment:\*\* Determines how software components of the system are distributed across hardware devices.

- \*\*Software Components:\*\*

- \*\*Data Storage:\*\* Holds data.

- \*\*Data Access Logic:\*\* Manages data retrieval and storage operations.

- \*\*Application Logic:\*\* Connects user interfaces and databases (CRUD operations).

- \*\*Presentation Logic:\*\* Manages the display and user interactions.

- \*\*Hardware Components:\*\*

- \*\*Client Computers:\*\* Devices used by users (e.g., laptops, smartphones).

- \*\*Servers:\*\* Multi-user systems for data storage and processing.

- \*\*Network:\*\* Connects the hardware components.

### 2. \*\*Client-Server Architectures:\*\*

- \*\*Client-Server Connection:\*\* Establishes communication between client devices and servers.

- \*\*Thick Client:\*\* Devices have their own storage and computing power.

- \*\*Pros:\*\* More processing power.

- \*\*Cons:\*\* Less secure, higher environmental impact.

- \*\*Thin Client:\*\* Relies on server resources.

- \*\*Pros:\*\* Lower environmental impact, more secure.

- \*\*Server-Based Architecture:\*\* Servers handle the majority of processing.

- \*\*Client-Based Architecture:\*\* Clients perform more tasks independently of the server.

### 3. \*\*Client-Server Tiers:\*\*

- \*\*Two-Tier Architecture:\*\* Clients communicate directly with servers.

- \*\*Three-Tier Architecture:\*\* Adds an intermediate layer between clients and servers.

- \*\*N-Tier Architecture:\*\* Multiple layers enhance flexibility and scalability.

### 4. \*\*Advances in Architecture Configurations:\*\*

- \*\*Virtualization:\*\* Creation of virtual resources like servers.

- \*\*Server Virtualization:\*\* Partitions a physical server into multiple virtual servers.

- \*\*Storage Virtualization:\*\* Combines multiple storage devices into a single unit.

- \*\*Benefits:\*\* Higher server efficiency, simplified management, lower costs, disaster recovery.

- \*\*Cloud Computing:\*\* Utilizes network or internet-based resources for storage, processing, and data transfer.

- \*\*Advantages:\*\* Reduces infrastructure and maintenance costs, improves collaboration, scalability, security, and performance.

### 5. \*\*Creating an Architecture Design:\*\*

- \*\*Operational Requirements:\*\* Ensures the system operates effectively under the environment it will be deployed in.

- \*\*Performance Requirements:\*\* Ensures the system meets performance benchmarks like speed and capacity.

- \*\*Security Requirements:\*\* Establishes controls to protect data and systems.

- \*\*Cultural and Political Requirements:\*\* Considers organizational and geographical constraints.

### 6. \*\*Hardware and Software Specification:\*\*

- \*\*Hardware Selection:\*\* Choose database servers, peripheral devices, clients, etc.

- \*\*Software Selection:\*\* Includes the operating system, special-purpose software, and licensing agreements.

- \*\*Document Specification:\*\* A formal document that lists the hardware and software requirements for the system.

### 7. \*\*Summary:\*\*

- \*\*Transition from Requirements to Design:\*\* Moves from gathering business needs to technical design.

- \*\*System Acquisition Strategies:\*\*

- \*\*Custom Development:\*\* Tailored to the business but costly.

- \*\*Packaged Software:\*\* Pre-built and efficient but limited customization.

- \*\*Outsourcing:\*\* Reduces internal development costs but comes with risks.

- \*\*Influences on Acquisition Strategy:\*\*

- \*\*Business Uniqueness:\*\* Custom development for unique needs, packaged software for common needs.

- \*\*In-House Skills:\*\* Determines whether to build in-house or outsource.

- \*\*Selecting an Acquisition Strategy:\*\* Uses tools like alternative matrices and requests for proposals (RFPs) to make decisions.

This breakdown provides a structured overview of the chapter's key points, explaining each topic in detail.  
  
"Transition to New System”

### 1. \*\*Introduction\*\*

- Transitioning to a new system is challenging due to business, technical, and people-related issues.

- Follow-up activities are essential after system installation.

### 2. \*\*Migration Plan\*\*

- Outlines activities, timing, and responsibilities for the transition.

- Includes business contingency plans.

### 3. \*\*Conversion Strategy\*\*

- \*\*Conversion Styles\*\*:

- \*\*Direct\*\*: Immediate replacement (risky but fast).

- \*\*Parallel\*\*: Old and new systems run together (less risk, higher cost).

- \*\*Conversion Locations\*\*:

- \*\*Pilot\*\*: Test in one location first.

- \*\*Phased\*\*: Rollout in stages.

- \*\*Simultaneous\*\*: All locations at once.

- \*\*Conversion Modules\*\*:

- \*\*Modular\*\*: Transition one part at a time.

- \*\*Whole-System\*\*: Entire system converted at once.

### 4. \*\*Business Contingency Plan\*\*

- Prevent small system malfunctions from becoming major problems.

- Parallel conversion helps with contingency planning.

### 5. \*\*Preparing the Technology\*\*

- Steps: Install hardware, software, and convert data.

- Data conversion is the most complex.

### 6. \*\*Change Management\*\*

- Helping people adapt with minimal stress.

- Key roles: sponsor, change agent, adopters.

- Change management plan includes revising policies, assessing costs/benefits, motivating adoption, and providing training.

### 7. \*\*Training\*\*

- Types: Classroom, one-on-one, and computer-based training.

- New system requires new skills, which may involve hiring or outsourcing.

### 8. \*\*Post-Implementation Activities\*\*

- \*\*Support\*\*: Help desk and on-demand training.

- \*\*Maintenance\*\*: Adjust the system based on problem reports, enhancements, and changes in software/hardware.

### 1. \*\*System Maintenance Concepts:\*\*

- \*\*Purpose:\*\* Ensures the system remains operational and continues to meet business requirements.

- \*\*Economic Life:\*\* Maintenance expenses impact the longevity of a system. Operational costs and maintenance expenses must be considered.

### 2. \*\*User Support Activities:\*\*

- \*\*User Training:\*\* New employees must be trained to use the company’s systems effectively.

- \*\*Help Desk Support:\*\*

- Assists with issues like data queries, network access, and password problems.

- Provides guidance on advanced features and software recovery.

- May include online chat support for real-time assistance.

### 3. \*\*Maintenance Activities:\*\*

- \*\*Corrective Maintenance (Bug Fixing):\*\*

- Addresses errors in the system that affect users.

- Users report issues, which are diagnosed and corrected by the maintenance team.

- \*\*Perfective Maintenance:\*\*

- Enhances system performance, efficiency, and functionality.

- Often results in reliability improvements, though it is not always necessary.

- \*\*Adaptive Maintenance:\*\*

- Modifies the system to remain usable in a changing environment.

- Example: Updating API integration when external platforms (e.g., social media) change their protocols.

- \*\*Preventive Maintenance:\*\*

- Proactively modifies software to detect and correct issues before they occur.

- Helps prevent future system errors.

### 4. \*\*Managing System Support:\*\*

- \*\*Maintenance Team:\*\*

- \*\*System Administrator:\*\* Responsible for installing, configuring, monitoring, troubleshooting, and ensuring security.

- \*\*Systems Analysts:\*\* Involved in planning, analysis, design, and implementation phases of support.

- \*\*Programmers:\*\*

- \*\*Applications Programmer:\*\* Works on software-specific development and maintenance.

- \*\*Systems Programmer:\*\* Focuses on system-level programming.

- \*\*Database Programmer:\*\* Manages the database structure and logic.

- \*\*Programmer Analyst:\*\* Combines both system and application-level programming skills.

### 5. \*\*System Obsolescence:\*\*

- \*\*End of Life:\*\*

- A system's operational life continues until maintenance costs increase significantly, or users demand new features.

- As this happens, new system requests are submitted, restarting the System Development Life Cycle (SDLC).

**Formula**

Weighted Scoring Method

WS =∑(Criteria Weight×Project Score)  
  
Payback period

Normal payback period =>   
(initial investment / annual cash flow)  
  
Unequal payback period   
Cumulative cash flow => (last year of cumulative cash flow + annual cash flow)

Payback period =>

last year of negative cash flow + (last year of cumulative cash flow / first year of positive cashflow)  
  
Net Present Value  
PV => cash flow / (1 + $rate of $return)^year  
NPV => SUM(all present value) – initial investment