Big Data Modeling (UCAML403 (F))

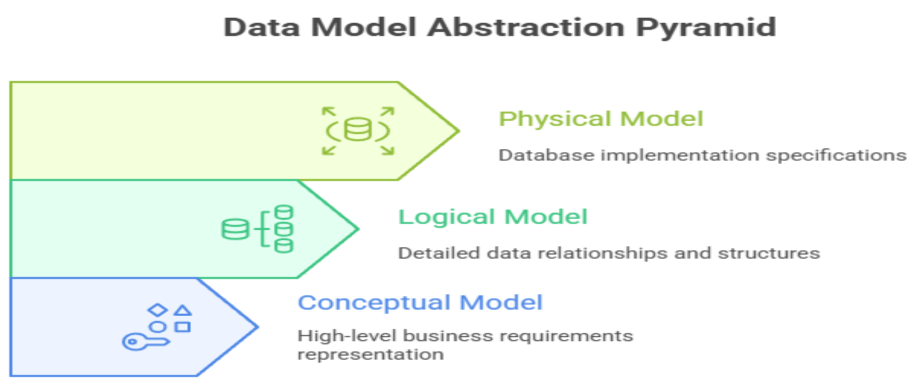
UNIT II: MODELING TYPES

UNIT II: Modeling Types
Data Modeling Introduction, Data Modeler duties, Classification of Data Modeling, Data Modeling tools, IDEF1X and IE methodology.

2.Data Modeling Introduction:

- Data modeling is the process of creating a visual representation of either a whole information system or parts of it to communicate connections between data points and structures.
- The goal of data modelling to illustrate the types of data used and stored within the system, the relationships among these data types, the ways the data can be grouped and organized and its formats and attributes.
- Data models are built around business needs. Rules and requirements are defined upfront through feedback from business stakeholders so they can be incorporated into the design of a new system or adapted in the iteration of an existing one.

- Data can be modeled at various levels of abstraction. The process begins by collecting information about business requirements from stakeholders and end users.


Data Model Abstraction Pyramid
Physical Model
Database implementation specifications
Logical Model
Detailed data relationships and structures
Conceptual Model
High-level business requirements representation

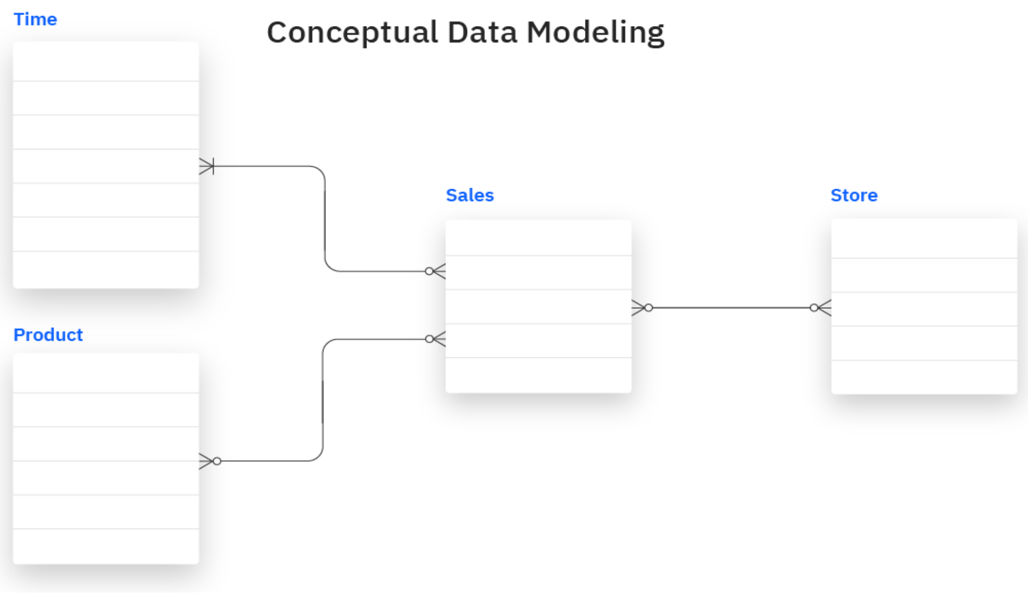
2.1 Classification of data models:

Data models can generally be divided into three categories, which vary according to their degree of abstraction. The process will start with a conceptual model, progress to a logical model and conclude with a model physical.

1. Conceptual data model
2. Logical data models
3. Physical data models

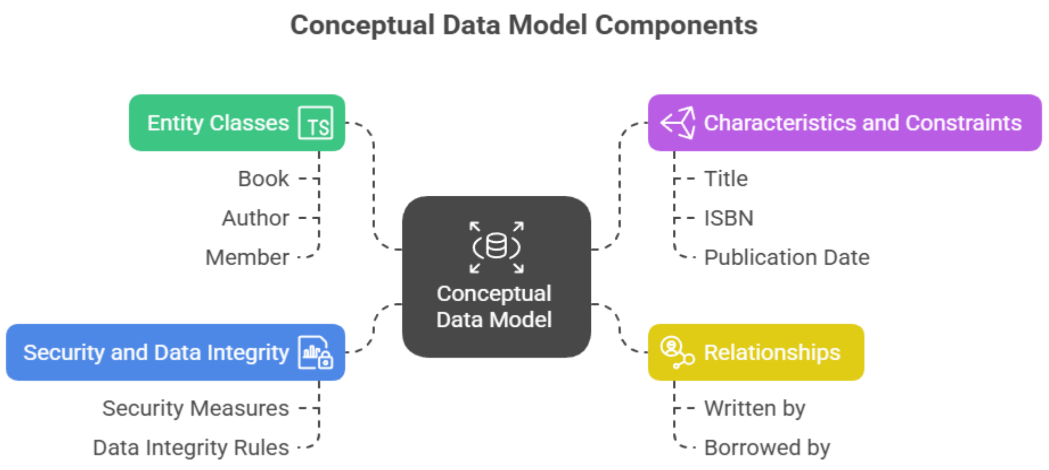
1.Conceptual data model:

Conceptual models are usually created as part of the process of gathering initial project requirements. Typically, they include entity classes (defining the types of things that are important for the business to represent in the data model), their characteristics and constraints, the relationships between them and relevant security and data integrity requirements.



Key characteristics of conceptual data models:

- Entity Classes: They identify and define the types of things (entities) that are important for the business to represent in the data model. For example, in a library system, entity classes might include "Book," "Author," and "Member."
- Characteristics and Constraints: They describe the attributes or properties of each entity class and any constraints that apply to them. For instance, a "Book" entity might have attributes like "Title," "ISBN," and "Publication Date," with a constraint that the ISBN must be unique.
- Relationships: They define the relationships between entity classes. For example, a "Book" is "written by" an "Author," and a "Member" can "borrow" a "Book."
- Security and Data Integrity Requirements: They outline the necessary security measures and data integrity rules to ensure data accuracy and protection.



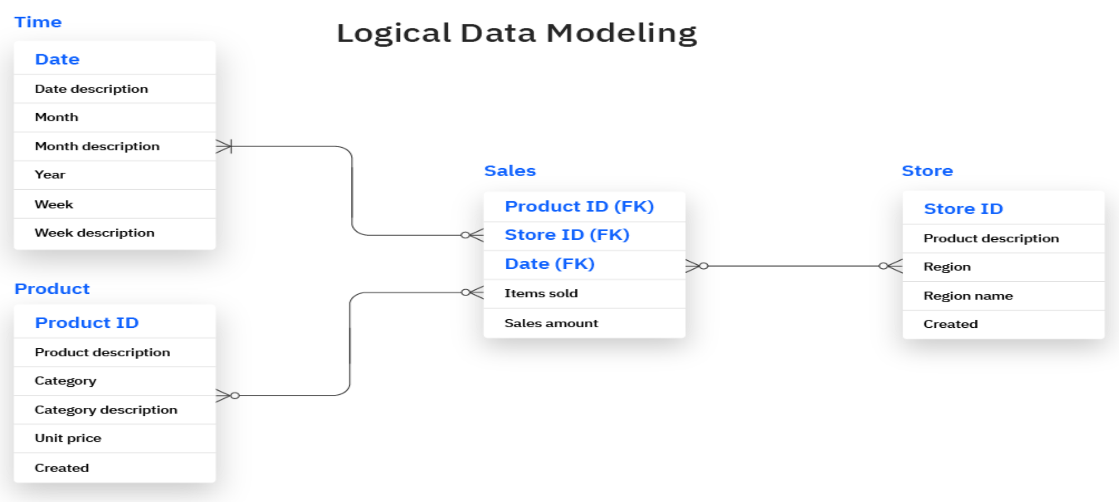
2.Logical data models:

They are less abstract and provide greater detail about the concepts and relationships in the domain under consideration.

One of several formal data modeling notation systems is followed. These indicate data attributes, such as data types and their corresponding lengths, and show the relationships among entities.

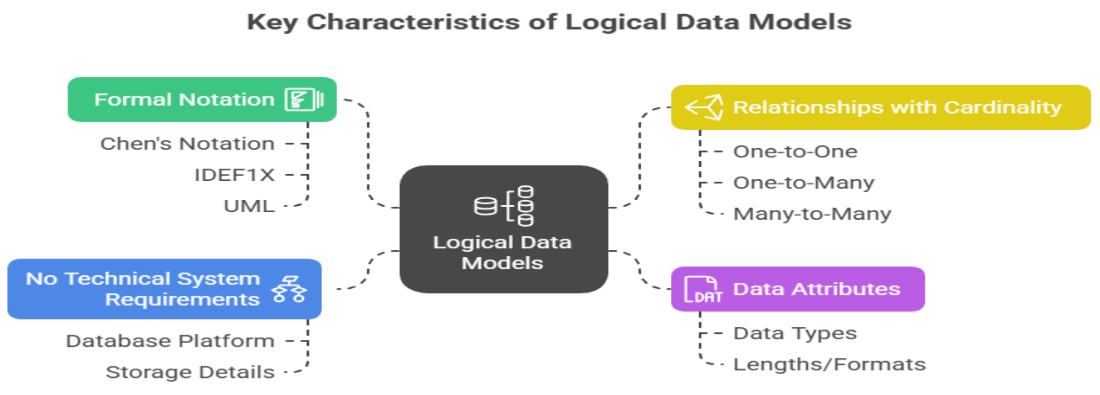
Logical data models don’t specify any technical system requirements.

This stage is frequently omitted in agile or DevOps practices. Logical data models can be useful in highly procedural implementation environments, or for projects that are data-oriented by nature, such as data warehouse design or reporting system development.



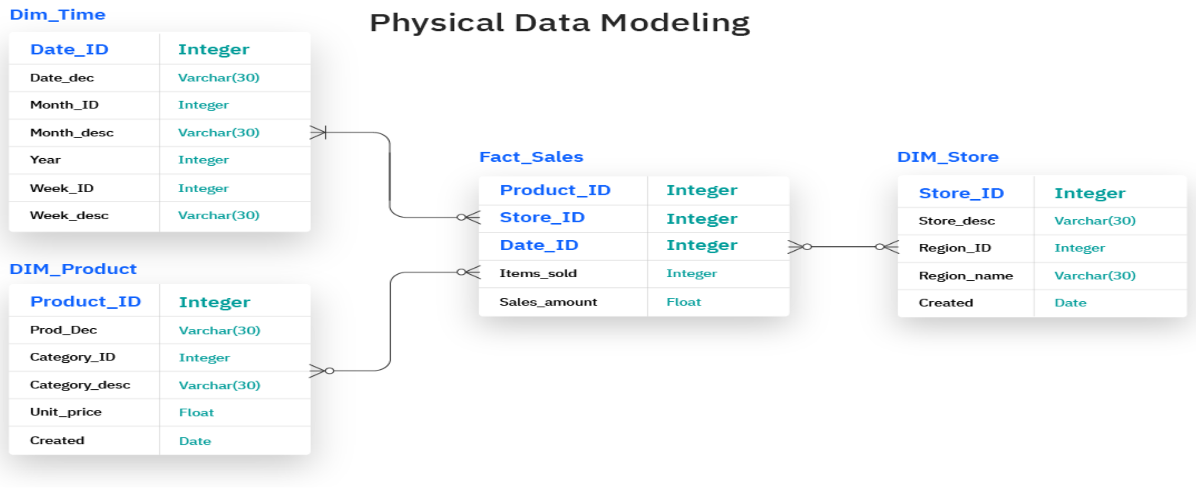
Key characteristics of logical data models:

- **Formal Notation:** They follow a formal data modeling notation system (e.g., Chen's notation, IDEF1X, or UML) to represent entities, attributes, and relationships.
- **Data Attributes:** They specify data attributes for each entity, including data types (e.g., integer, string, date) and their corresponding lengths or formats. For example, the "Publication Date" attribute of the "Book" entity might be defined as a date data type.
- **Relationships with Cardinality:** They define the relationships between entities with greater precision, including cardinality constraints (e.g., one-to-one, one-to-many, many-to-many). For example, one "Author" can write many "Books," but each "Book" is written by only one "Author" (one-to-many relationship).
- **No Technical System Requirements:** Logical data models focus on the logical structure of the data and do not specify any technical system requirements, such as database platform or storage details.



3. Physical data models

- They provide a schema for how the data will be physically stored within a database. As such, they're the least abstract of all.
- They offer a finalized design that can be implemented as a relational database, including associative tables that illustrate the relationships among entities as well as the primary keys and foreign keys that will be used to maintain those relationships.
- Physical data models can include database management system (DBMS)-specific properties, including performance tuning.



Key characteristics of physical data models:

- Database Schema: They provide a schema for the database, including table structures, column definitions, data types, and constraints.
- Primary and Foreign Keys: They define primary keys for each table to uniquely identify records and foreign keys to establish relationships between tables.
- Associative Tables: They include associative tables (also known as junction tables) to represent many-to-many relationships between entities.
- DBMS-Specific Properties: They can incorporate DBMS-specific properties, such as data types, indexing strategies, and storage parameters, to optimize performance.
- Performance Tuning: They consider performance tuning aspects, such as indexing, partitioning, and caching, to ensure efficient data access and retrieval

2.3 Data modeling process:

Data modeling techniques have different conventions that dictate which symbols are used to represent the data, how models are laid out, and how business requirements are conveyed.

1. Identify the entities. The process of data modeling begins with the identification of the things, events or concepts that are represented in the data set that is to be modeled. Each entity should be cohesive and logically discrete from all others.
2. Identify key properties of each entity. Each entity type can be differentiated from all others because it has one or more unique properties, called attributes. For instance, an entity called "customer" might possess such attributes as a first name, last name, telephone number and salutation, while an entity called "address" might include a street name and number, a city, state, country and zip code.
3. Identify relationships among entities. The earliest draft of a data model will specify the nature of the relationships each entity has with the othersIf that model were expanded to include an entity called "orders," each order would be shipped to and billed to an address as well. These relationships are usually documented via unified modeling language (UML).

4. Map attributes to entities completely. This will ensure the model reflects how the business will use the data. Several formal data modeling patterns are in widespread use. Object-oriented developers often apply analysis patterns or design patterns, while stakeholders from other business domains may turn to other patterns.

5. Assign keys as needed, and decide on a degree of normalization that balances the need to reduce redundancy with performance requirements.

   Normalization is a technique for organizing data models (and the databases they represent) in which numerical identifiers, called keys, are assigned to groups of data to represent relationships between them without repeating the data.

6. Finalize and validate the data model. Data modeling is an iterative process that should be repeated and refined as business needs change.

Types of data modeling:

Data modeling has evolved alongside database management systems, with model types increasing in complexity as businesses' data storage needs have grown**.**

1. **Hierarchical data models**
2. **Relational data models**

1. Hierarchical data models represent one-to-many relationships in a treelike format. In this type of model, each record has a single root or parent which maps to one or more child tables. This model was implemented in the IBM Information Management System (IMS), which was introduced in 1966 and rapidly found widespread use, especially in banking. Though this approach is less efficient than more recently developed database models, it's still used in Extensible Markup Language (XML) systems and geographic information systems (GISs).

2. Relational data models were initially proposed by IBM researcher E.F. Codd in 1970. They are still implemented today in the many different relational databases commonly used in enterprise computing.

-Relational data modeling doesn't require a detailed understanding of the physical properties of the data storage being used. In it, data segments are explicitly joined through the use of tables, reducing database complexity.

- **Entity-relationship (ER) data models** use formal diagrams to represent the relationships between entities in a database.

  Several ER modeling tools are used by data architects to create visual maps that convey database design objectives.

- **Object-oriented data models** gained traction as object-oriented programming and it became popular in the mid-1990s. The "objects" involved are abstractions of real-world entities.

  Objects are grouped in class hierarchies, and have associated features. Object-oriented databases can incorporate tables, but can also support more complex data relationships. This approach is employed in multimedia and hypertext databases as well as other use cases.

- **Dimensional data models** were developed by Ralph Kimball, and they were designed to optimize data retrieval speeds for analytic purposes in a data warehouse. While relational and ER models emphasize efficient storage, dimensional models increase redundancy in order to make it easier to locate information for reporting and retrieval. This modeling is typically used across OLAP systems.

Benefits of data modeling:

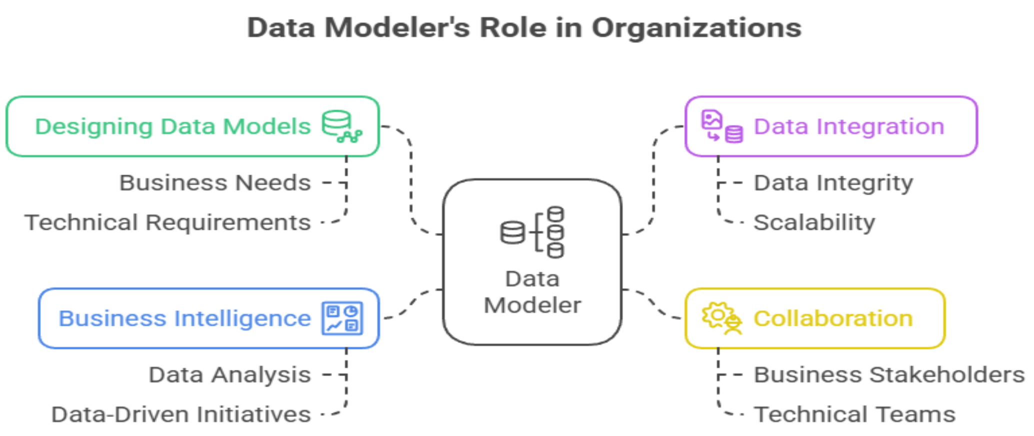
Data modeling makes it easier for developers, data architects, business analysts, and other stakeholders to view and understand relationships among the data in a database or data warehouse. In addition, it can:

- Reduce errors in software and database development.
- Increase consistency in documentation and system design across the enterprise.
- Improve application and database performance.
- Ease data mapping throughout the organization.
- Ease and speed the process of database design at the conceptual, logical and physical levels.

Data Modeler duties:

- A data modeler's primary duty is to create, manage, and optimize data models that align with an organization's business needs and technical requirements.
- They translate business requirements into structured data models, ensuring data integrity, scalability, and efficient data integration.
- Data modelers collaborate with various stakeholders, including business analysts and data engineers, to design and implement data models that support business intelligence, data analysis, and other data-driven initiatives.

1. Designing Data Models:
   - Creating conceptual, logical, and physical data models.
   - Ensuring data models are scalable, flexible, and optimized for performance.
2. Data Integration and Transformation:
   - Integrating data from various sources (databases, cloud platforms, etc.).
   - Transforming data into formats suitable for analysis and reporting.
3. Collaboration and Communication:
   - Working with stakeholders (business analysts, data engineers, database administrators).
   - Understanding business requirements and translating them into data models.
4. Performance Optimization and Maintenance:
   - Optimizing data models for performance and efficiency.
   - Monitoring data models for performance issues and errors.
5. Data Governance and Security:
   - Ensuring data security and compliance with relevant regulations.
   - Implementing data governance policies and procedures.
   - Managing and protecting sensitive data.



Why Use Data Modeling Tools?

Data modeling tools are software applications that help users create visual representations of data
Data modeling tools offer several benefits:

- **Improved Communication:** Data models provide a common language for stakeholders, including business users, developers, and database administrators, to understand and discuss data requirements.
- **Reduced Development Time:** By visualizing data structures and relationships, data modeling tools help identify potential issues early in the development process, reducing rework and saving time.
- **Enhanced Data Quality:** Data models enforce data integrity rules, ensuring data accuracy and consistency.
- **Better Database Performance:** Well-designed data models optimize database performance by minimizing data redundancy and improving query efficiency.
- **Simplified Maintenance:** Data models provide a clear understanding of the database structure, making it easier to maintain and modify the database over time.
- **Support for Data Governance:** Data modeling tools can help organizations implement data governance policies by defining data standards and ensuring data quality.

Popular Data Modeling Tools

Here are some popular data modeling tools available in the market:

- **erwin Data Modeler:** A comprehensive data modeling tool that supports a wide range of data modeling techniques and DBMS. It offers advanced features for data governance, metadata management, and collaboration.
- **Toad Data Modeler:** A user-friendly data modeling tool that supports multiple platforms and databases. It offers features for reverse engineering, forward engineering, and code generation.
- **dbForge Studio for MySQL:** A powerful IDE for MySQL database development, management, and administration. It includes a visual database designer for creating and editing data models.
- **SQL Developer Data Modeler:** A free data modeling tool from Oracle that supports Oracle databases and other DBMS. It offers features for creating conceptual, logical, and physical data models.
- **Lucidchart:** A web-based diagramming tool that can be used for data modeling. It offers a simple and intuitive interface for creating ERDs and other types of diagrams.
- **draw.io:** A free, open-source diagramming tool that can be used for data modeling. It offers a wide range of shapes and connectors for creating ERDs and other types of diagrams.
- **Enterprise Architect:** A comprehensive UML modeling tool that can also be used for data modeling. It supports a wide range of modeling languages and techniques.
- **Visual Paradigm:** A modeling platform that supports UML, BPMN, and other modeling languages. It also offers features for data modeling, including ERD diagrams and database design.
- **ER/Studio:** A data modeling tool that supports multiple databases and platforms. It offers features for reverse engineering, forward engineering, and data dictionary management.

Choosing the Right Data Modeling Tool

When choosing a data modeling tool, consider the following factors:

- **Data Modeling Needs:** Determine the specific data modeling requirements of your organization, such as the types of data models you need to create, the DBMS you need to support, and the level of collaboration required.
- **Ease of Use:** Choose a tool that is easy to learn and use, with a user-friendly interface and clear documentation.
- **Features:** Evaluate the features offered by different tools and choose one that meets your specific needs.

- **Cost:** Consider the cost of the tool, including licensing fees, maintenance fees, and training costs.
- **Integration:** Ensure that the tool integrates with your existing development tools and infrastructure.
- **Scalability:** Choose a tool that can scale to meet the growing data modeling needs of your organization.
- **Support:** Look for a tool that offers good customer support and a strong user community.

IDEF1X and IE Methodology

IDEF1X

IDEF1X is a data modeling methodology used to create semantic data models. It is particularly useful for designing relational databases. It's a public domain methodology, derived from the earlier Information Modeling Technique (IMT) developed by Robert G. Brown of D. Appleton Company (DACOM). The U.S. Air Force standardized IDEF1X in 1993.

Core Principles

- **Semantic Clarity:** IDEF1X emphasizes creating models that accurately reflect the meaning and relationships within the data.
- **Entity-Relationship Focus:** The methodology centers around identifying entities (real-world objects or concepts) and the relationships between them.
- **Rigorous Notation:** IDEF1X employs a precise graphical notation to represent entities, attributes, and relationships, ensuring consistency and unambiguous communication.
- **Top-Down Approach:** IDEF1X typically follows a top-down approach, starting with a high-level overview and progressively refining the model to greater detail.

Modeling Techniques

IDEF1X uses a specific set of graphical symbols to represent different components of a data model:

- **Entities:** Represented by rectangles, entities are the fundamental building blocks of the model. They represent real-world objects, concepts, or events about which information is stored.
- **Attributes:** Listed within the entity rectangle, attributes describe the characteristics or properties of the entity.
- **Relationships:** Represented by lines connecting entities, relationships define how entities are associated with each other. IDEF1X distinguishes between different types of relationships

Information Engineering (IE) Methodology

Information Engineering (IE) is a comprehensive methodology for planning, analyzing, designing, and implementing information systems across an entire enterprise. It takes a broader perspective than IDEF1X, encompassing both data and process modeling.

Core Principles

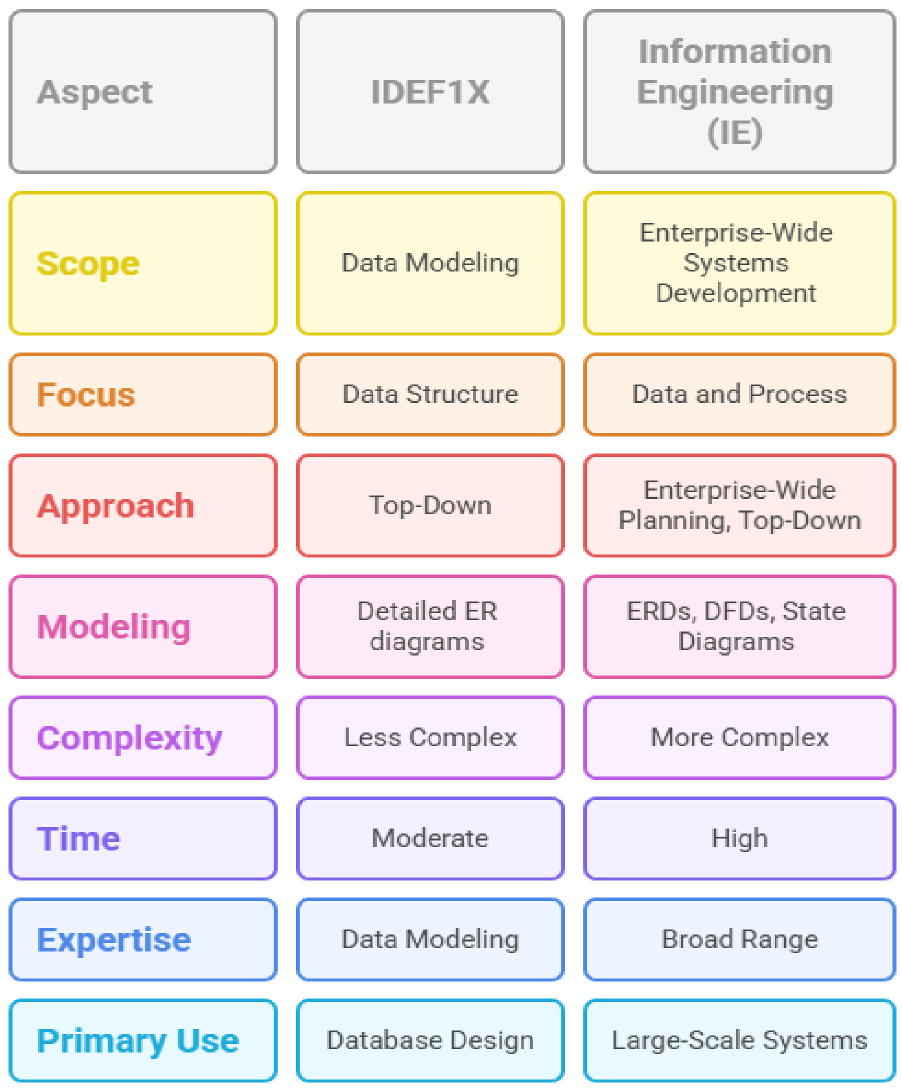
- **Enterprise-Wide Perspective:** IE aims to align information systems with the overall business strategy of the organization.
- **Data-Driven Approach:** IE emphasizes the importance of data as a shared resource and focuses on creating a stable and consistent data architecture.
- **Process-Oriented:** IE also considers the processes that use and manipulate the data, ensuring that the system supports business operations effectively.
- **Iterative Development:** IE typically involves an iterative development approach, with prototypes and feedback loops to refine the system.

Modeling Techniques

IE uses a variety of modeling techniques to represent different aspects of the system:

- **Entity-Relationship Diagrams (ERDs):** Similar to IDEF1X, ERDs are used to model the data structure, including entities, attributes, and relationships. However, IE ERDs may use a slightly different notation.
- **Data Flow Diagrams (DFDs):** DFDs are used to model the flow of data through the system, showing processes, data stores, and external entities.
- **State Transition Diagrams:** State transition diagrams are used to model the behavior of entities over time, showing the different states an entity can be in and the transitions between those states.
- **Process Decomposition Diagrams:** These diagrams break down complex processes into smaller, more manageable sub-processes.
- **Action Diagrams:** Action diagrams are used to represent the logic of individual processes.

Comparison of IDEF1X and Information Engineering



Aspect	IDEF1X	Information Engineering (IE)
Scope	Data Modeling	Enterprise-Wide Systems Development
Focus	Data Structure	Data and Process
Approach	Top-Down	Enterprise-Wide Planning, Top-Down
Modeling	Detailed ER diagrams	ERDs, DFDs, State Diagrams
Complexity	Less Complex	More Complex
Time	Moderate	High
Expertise	Data Modeling	Broad Range
Primary Use	Database Design	Large-Scale Systems

IDEF1X and IE are both valuable methodologies for information systems development, but they differ in scope and focus. IDEF1X is a powerful tool for data modeling and relational database design, while IE provides a comprehensive framework for developing enterprise-wide information systems. The choice between the two depends on the specific project requirements and the organization's goals