Data Models

Data models define how data is structured, stored, and accessed in a database system. They provide a blueprint for creating databases that align with business requirements.

Components of a Data Model

Database design consists of both structural components and rules that govern the organization and behavior of data.

- **Entities**: Represent real-world objects or concepts that have a distinct existence. Each entity is stored as a table in a relational database.
    - Example: In a university database, entities could include `Students`, `Courses`, and `Professors`.
- **Attributes**: Characteristics or properties of entities. Attributes are the columns in a table.
    - Example: A `Student` entity might have attributes like `StudentID`, `Name`, and `EnrollmentDate`.
- **Relations**: Define how entities are connected to one another. These are represented using foreign keys in tables.
    - Example: A relationship between `Students` and `Courses` can be defined through a `Registrations` table, where each record links a student to a course.
- **Constraints**: Rules that restrict the values that can be stored in the database to maintain data integrity.
    - Example: A constraint ensuring that the `Email` attribute in a `Students` table is unique for each student.

Data Modeling and Business Rules

Data modeling involves creating a visual representation of data entities, their attributes, and relationships, typically using Entity-Relationship Diagrams (ERDs). Business rules define specific policies or conditions that the database must enforce (See *Business Rules* in the *Database Design* handout).

Stages of Data Modeling

- **Conceptual Database Design**: Focuses on what data needs to be stored and how it relates to other data.
    - Uses Entity-Relationship (ER) diagrams to represent entities, attributes, and relationships.

- **Logical Database Design**: Translates the conceptual design into a schema suitable for a specific DBMS.
    - Specifies tables, columns, data types, and relationships.
    - Ensures normalization to eliminate redundancy and maintain data integrity.
- **Physical Database Design**: Focuses on how data is stored and accessed to optimize performance.
    - Includes considerations for indexing, partitioning, and storage formats.
    - Example: Using a clustered index on the `OrderDate` column to optimize range queries.