Data Collection and Modeling

Course 8 - Data Modeling

Definition

“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 is 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.” (IBM)

Data Models

Is a specification of data structures and rules together with a visual representation of data and relations between elements. It answers to:

- Who
- What
- Where
- Why
- When

Data model should evolve with changes and should be shared

Data Models Types

Can be defined at different levels of abstraction and start at higher level with an overview, becoming more specific, in a top-down approach; usually the following data models are used during the 3 phases of data modeling:

- Conceptual data model
- Logical data model
- Physical data model

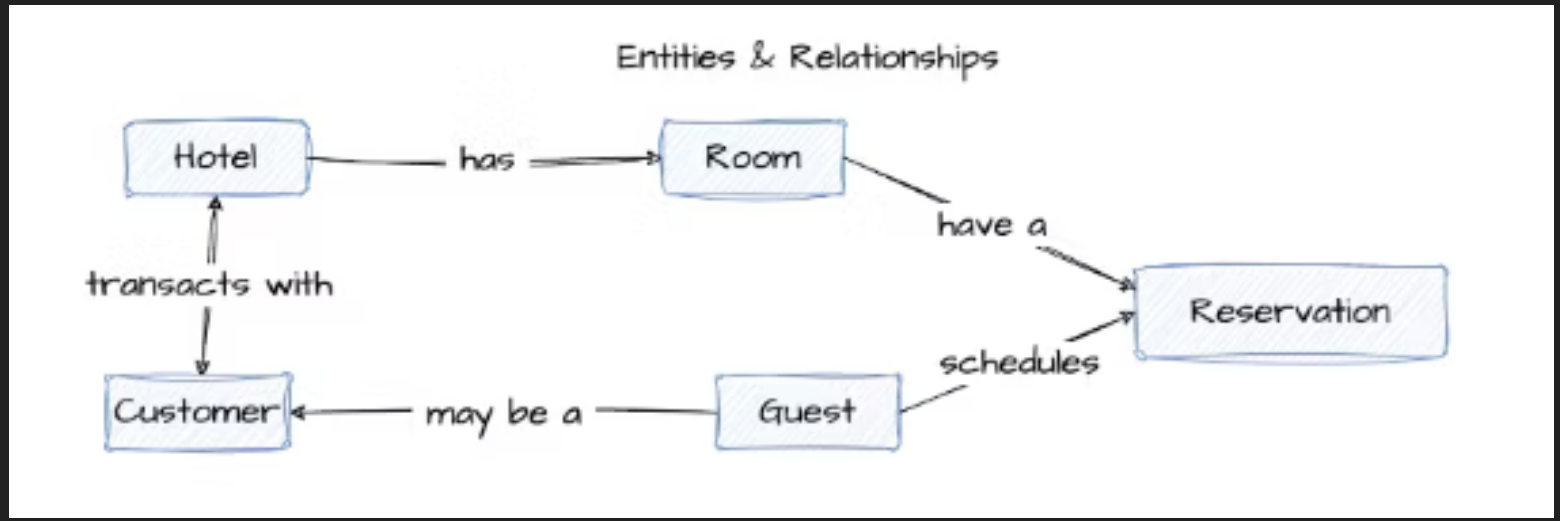
Conceptual Model

identifies data needed in processes and analytics/reporting applications, with business rules and concepts. It doesn't define the data processing flow or physical characteristics.

it is also referred to as domain model and provides an overview of what the system contains, how it is organized, and which business rules are used.

is created during gathering initial project requirements. It includes entity classes, characteristics and constraints, relationships between entities and security and data integrity requirements.

Conceptual Model



Source: www.thoughtspot.com/data-trends/data-modeling

Logical Model

is less abstract; provides more details about data structures (tables), attributes (columns), their data types and lengths, and show the relationships among entities (foreign keys).

the model is independent of database technology or file structure.

can be implemented as relational, multidimensional or NoSQL system; logical data models don’t specify any technical system requirements

Physical Model

provides a schema for the physical storage within a database.

offers a design that can be directly implemented as a relational database, with tables that implement the relationships between entities by means of primary keys and foreign keys.

physical data models can include database management system specific properties, including performance tuning (constraints, indexes, triggers, tablespaces and partitions).

Modeling Process

1. Identify entities: each should be cohesive and discrete from others
2. Identify properties for each entity: unique properties (attributes)
3. Identify relationships between entities: nature of relationship with others; may be specified using UML
4. Map attributes to entities: using some design patterns (OOP design patterns, business patterns)
5. Decide degree of normalization (and assign keys): obtain a good balance between redundancy and performance
6. Validate model: in an iterative process with the business

Types of Modeling

- Hierarchical: tree structure with parent-child relationship, with links and types
- Network: extends previous model to allow more parents for a child record
- Relational: data stored in tables with columns, having specified relations between entities
- ER: formal diagrams to represent entities and their relationships
- OO: derive from OOP, abstractions of domain entities (data and behavior); uses inheritance
- Dimensional: facts to represent measurements and dimensions to represent the context
- Graph: uses nodes to represent each entity instance and edges to depict relationships (can have direction and label); can be property or semantic

Database Design

- Requirements analysis: identifies needed data ,supported operations, applications that will access data
- Conceptual design: high level description of data
- Logical design: translation of conceptual design to DB schema (and model)
- Schema refinement: normalization, redundancy elimination
- Physical design: performance considerations (indexes, partitioning, MV) and possibly some schema redesign

Database Conceptual Modeling

Is a structured business view of data; focused on identifying data used in business processes, not the processing flow

Represents the structure of data independent of software or storage

Defines:

- Non technical data concepts/entities
- High level relationships between entities, without their cardinalities

Consider a database to be a collection of objects/entities

Database Logical Modeling

Contains more details than the conceptual model, with specifications of entity attributes and relationships between these entities.

It is still independent from the DBMS

Defines:

- Entities (tables)
- Attributes (columns)
- Relationships (keys)

Uses business names for entities and attributes

Logical Model

There are 2 popular high-level design models:

- Entity-Relationship (ER)
- Unified Modeling Language (UML)

Both are visual (graphical) models.

UML became more popular with the adoption of OOP

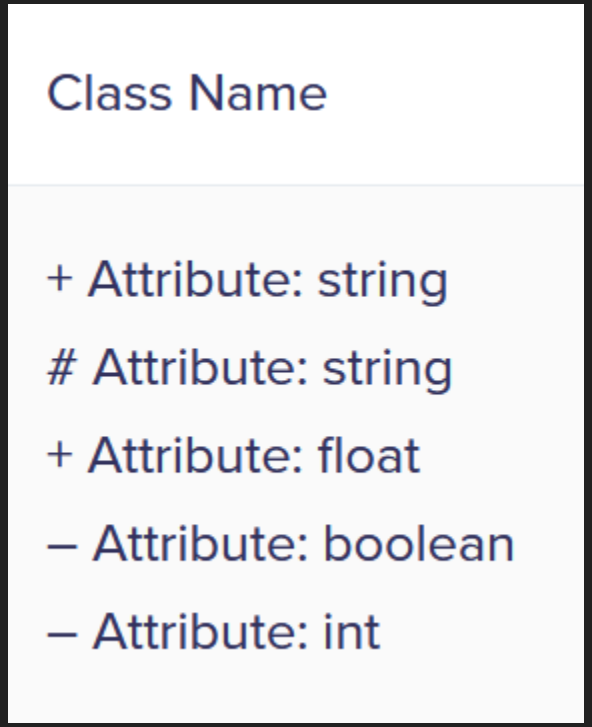
UML Database Modeling

UML class diagrams describe the type of objects and the relationships between them

It contains:

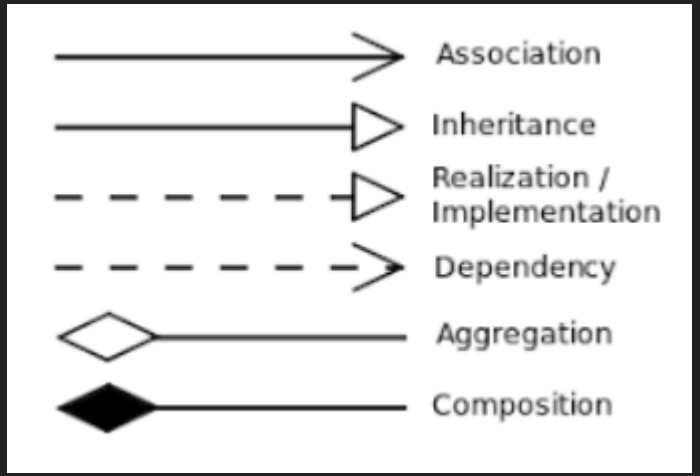
- Classes: have a name, attributes and behaviors/methods
- Relationships: different types represented by different styles of arrows

Classes


Class Name
+ Attribute: string
# Attribute: string
+ Attribute: float
– Attribute: boolean
– Attribute: int

Relationships

- Association: relationship between 2 different classes; can also have cardinality
- Inheritance: relationship between parent and children/descendants
- Realization: relationship between interface and objects that implement it
- Dependency: an object of a class uses an object of another class in its methods and is not stored in any field
- Aggregation: a class is part of another
- Composition: same like aggregate but it gets destroyed when the aggregate is deleted



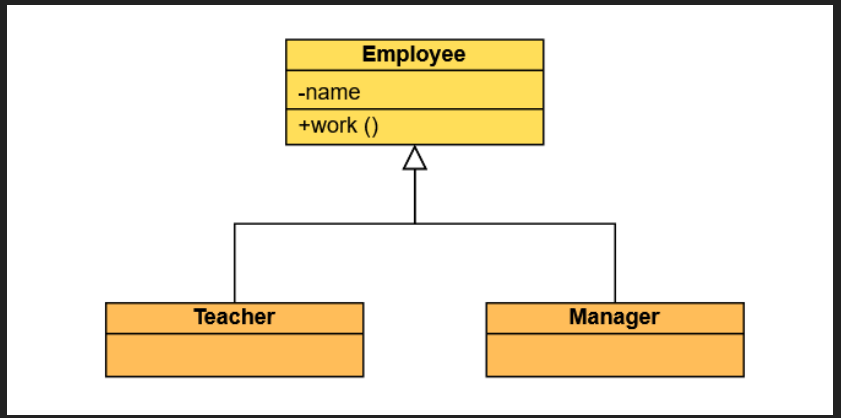
source: visual-paradigm

Relationships

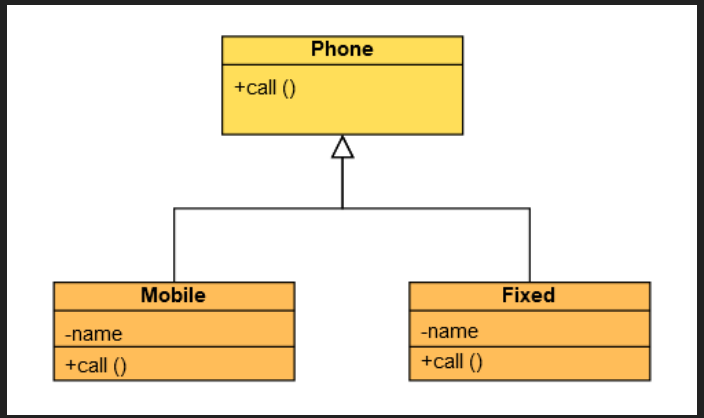
For association relationship there are several possible cardinalities:

Expression	Description
1	Exactly 1
0..1	0 or 1
*	0 or more
0..*	0 or more
1..*	1 or more
5..10	Value in the range specified
1,3,5,7	Exactly one of the enumerated values

Relationships

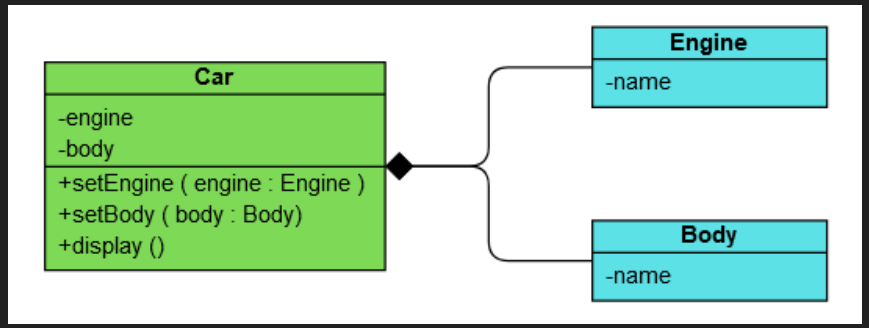


Inheritance

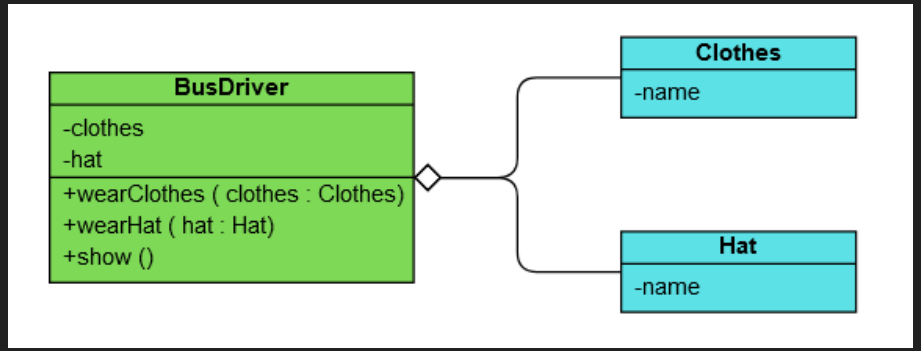


Realization

Relationships



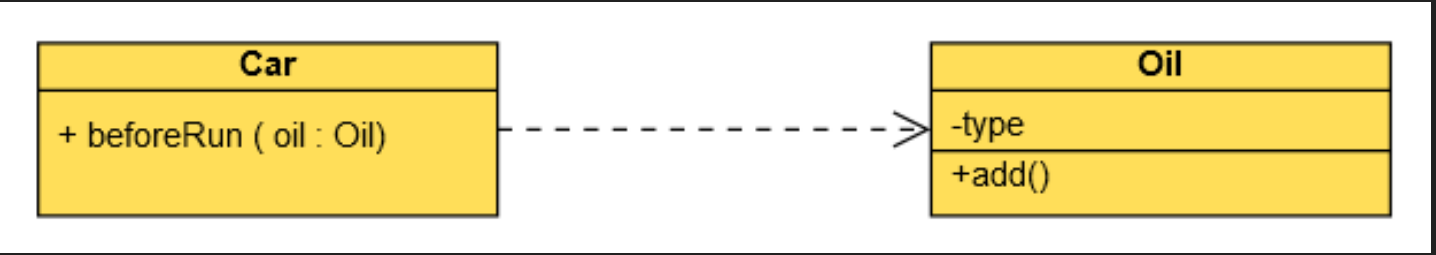
Composition



Aggregation

Relationships

Association



Dependency

Model to Database

Classes:

- become tables
- by convention tables have plural names (as opposed to classes)
- simple attributes are mapped into fields
- composite attributes are mapped into new tables
- derived attributes are not mapped

Model to Database

Associations:

- 1 to 0..1 : 1 table for each class (using FK)
- 1 to many: 1 table for each class (using FK)
- 1 to 1 : 1 table with reunion of the attributes
- many to many : 1 table for each class and a mapping table between them

Inheritance options:

1. A view is created for each subclass
2. A table for each subclass and superclass attributes are found in each one
3. A single table with reunion of superclass and subclass fields and an attribute to indicate the subclass

Principles of Well Design Data Architecture

1. Share data across enterprise: avoid creating silos and supply a transparent view of correlated information from all departments/business functions
2. Provide access to data: build proper interfaces that allow different levels of access
3. Analytics should be close to data: it is more efficient to have analytics apps close to data source than to ship large amounts of data to tools
4. Assure data compliance and governance: comply with all legal requirements but also take into consideration to follow best practices and comply with policies and procedures defined
5. Integrate multiple platforms in data architecture: different services could run on different platforms and be exposed in various ways

Advantages

- Reduce errors in database development (and software)
- Assist the design process at conceptual, logical and physical levels
- Better consistency in documentation and design
- Better communication between teams
- Improved data mapping in the enterprise
- Better app and database performance

Data Modeling Tools

- ErWin
- Enterprise Architect
- ER/Studio
- Lucidchart
- Draw.io
- Visual Paradigm