Process and Data Modeling

Lecturer: Adel Vahdati

Data Flow Diagram (DFD)

What is Data Flow Diagram (DFD)?

- Graphical representation of data flow within information systems
- Easily understood by both technical and non-technical users
- Used for system analysis and requirements specification
- Does not include control flow, loops, or decision rules
- Represents inputs, processes, outputs, and data storage

Characteristics of DFD

- Uses symbols for data, processes, and flow
- Simplifies complex systems through abstraction
- Provides system hierarchy using levels (0-level, 1-level, etc.)
- Easy to understand and useful for communication
- Supports modularity for easier analysis and design

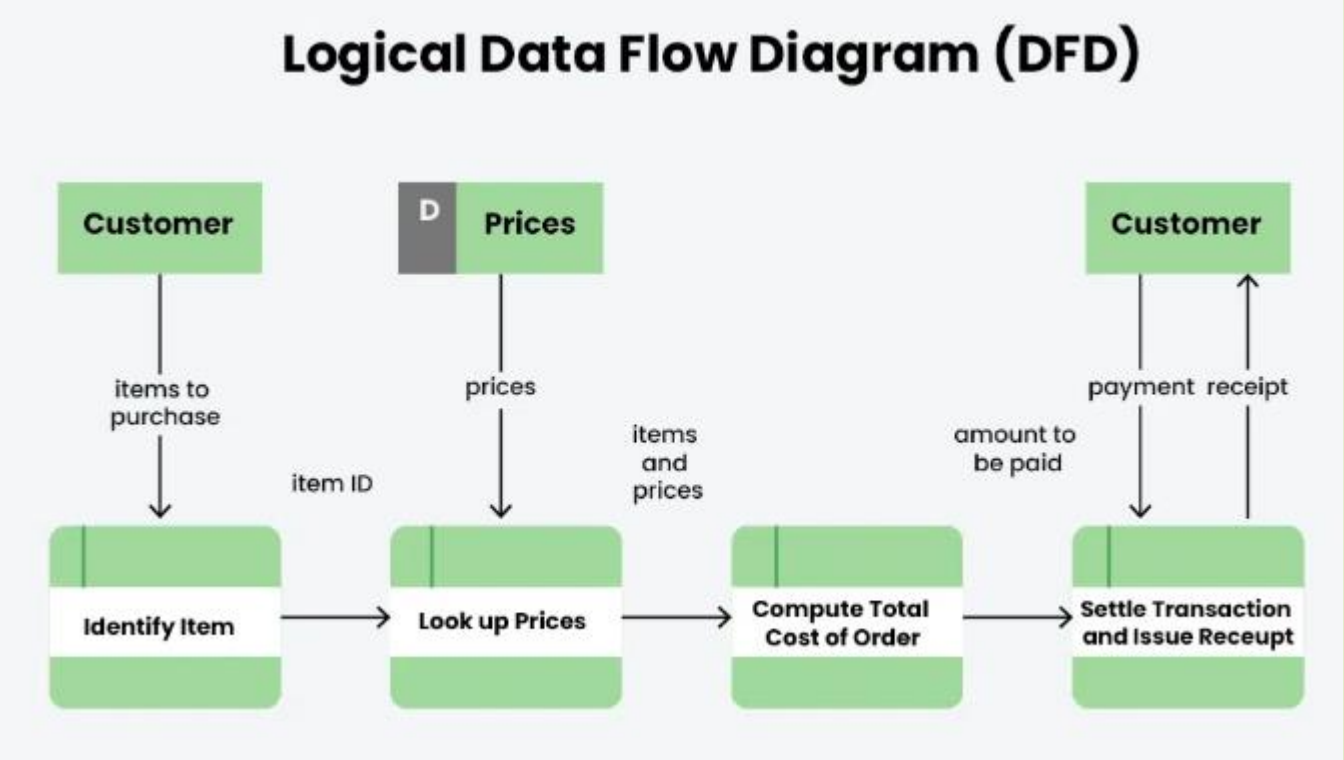
Types of DFD

- **Logical DFD:** Focuses on system processes and data flow
- **Physical DFD:** Shows implementation details, data transmission and storage

Logical Data Flow Diagram

- Logical data flow diagram mainly focuses on the system process.
- It illustrates how data flows in the system.
- Mainly focuses on high level processes and data flow without diving deep into technical implementation details.

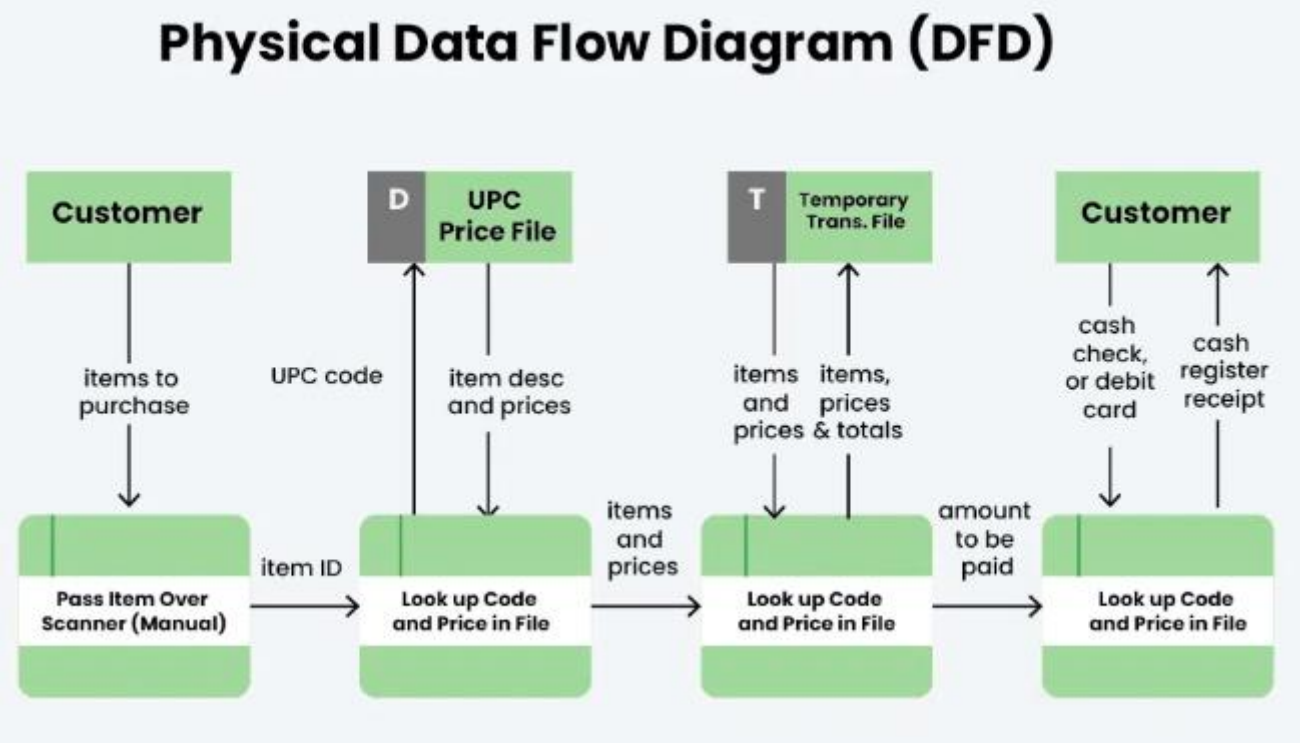
Logical Data Flow Diagram



Physical Data Flow Diagram

- Physical data flow diagram shows how the data flow is actually implemented in the system.
- It includes additional details such as data storage, data transmission, and specific technology or system components.
- It is more specific and close to implementation.

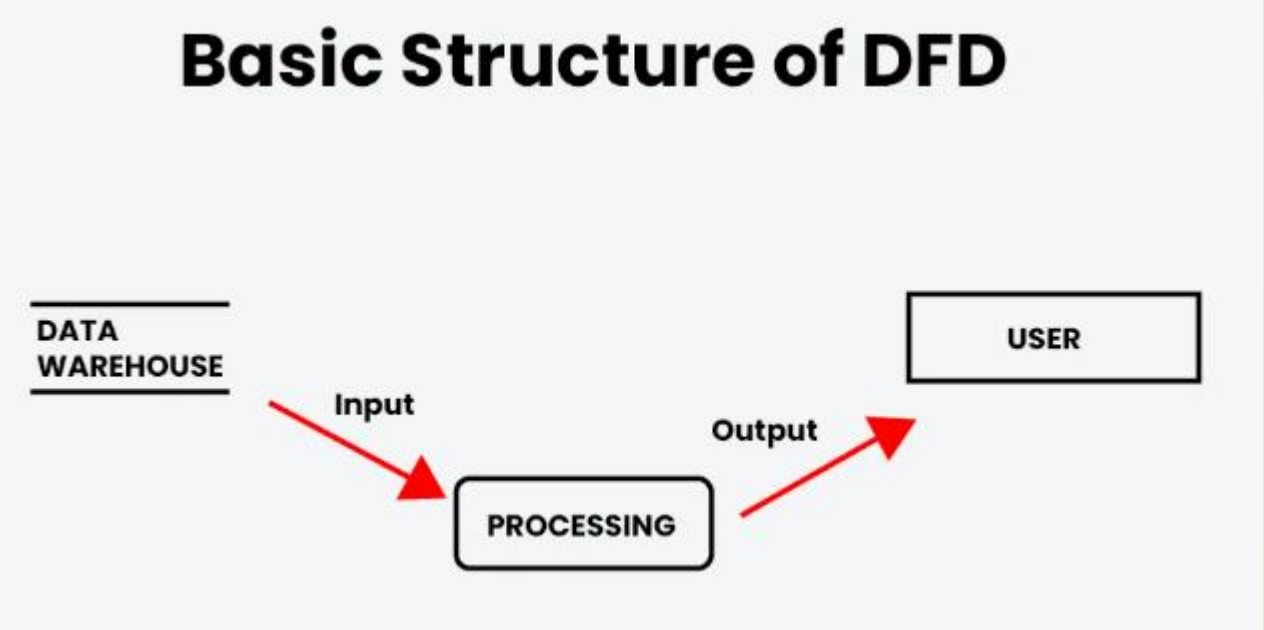
Physical Data Flow Diagram



Components of DFD

- **Process:** Transforms input to output; represented with rounded shapes
- **Data Flow:** Arrows showing data movement between parts of the system
- **Data Store:** Represented by two horizontal lines, stores data
- **External Entity (Terminator):** Outside systems or users interacting with the system

Basic Structure of DFD

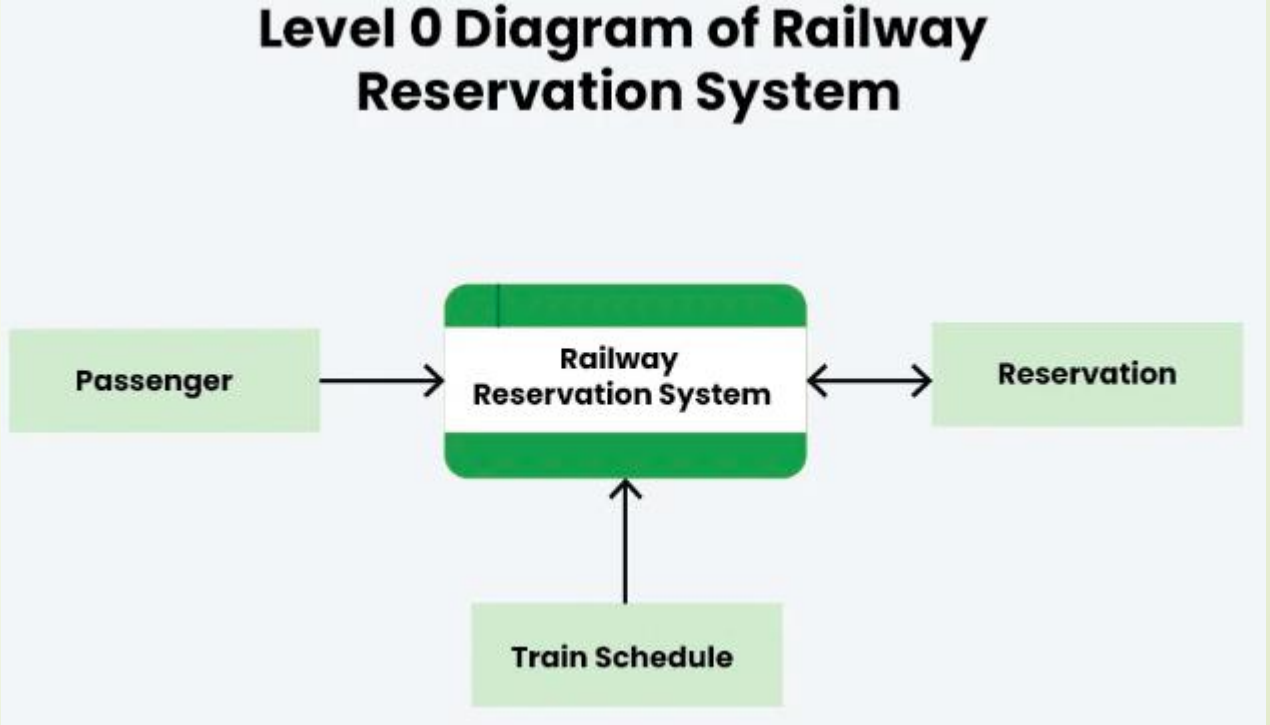


Levels of DFD

- **Level-0 DFD (Context Diagram):** High-level overview of system and its environment
- **Level-1 DFD:** Breaks down the system into subprocesses
- **Level-2 DFD:** Further decomposes 1-level processes for detailed analysis

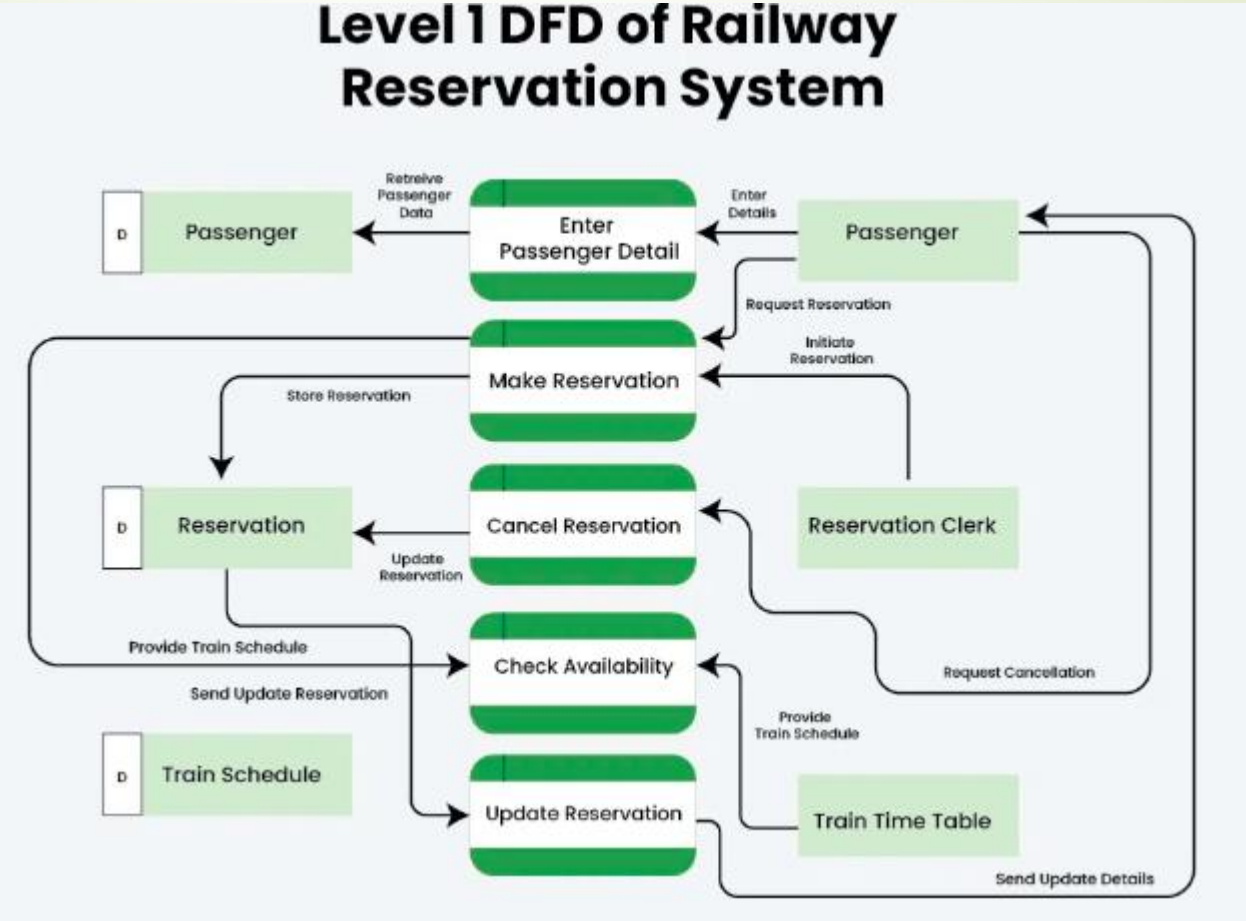
Level-0 DFD

- It is also known as a context diagram.
- It's designed to be an abstraction view, showing the system as a single process with its relationship to external entities.
- It represents the entire system as a single bubble with input and output data indicated by incoming/outgoing arrows.


Level 0 Diagram of Railway Reservation System
Passenger
Railway Reservation System
Reservation
Train Schedule

Level-1 DFD

- This level provides a more detailed view of the system by breaking down the major processes identified in the level 0 DFD into sub-processes.
- Each sub-process is depicted as a separate process on the level 1 DFD. The data flows and data stores associated with each sub-process are also shown.
- In 1-level DFD, the context diagram is decomposed into multiple bubbles/processes.
- In this level, we highlight the main functions of the system and breakdown the high-level process of 0-level DFD into subprocesses.


Level 1 DFD of Railway Reservation System
D
Passenger
Retreive Passenger Data
Enter Passenger Detail
Enter Details
Passenger
Request Reservation
Initiate Reservation
Make Reservation
Store Reservation
D
Reservation
Cancel Reservation
Update Reservation
Reservation Clerk
Provide Train Schedule
Send Update Reservation
Check Availability
Request Cancellation
Provide Train Schedule
D
Train Schedule
Update Reservation
Train Time Table
Send Update Details

DFD Rules - Data Can Flow From

- External Entity to Process
- Process to External Entity
- Process to Data Store
- Data Store to Process
- Process to Process

DFD Rules - Data Cannot Flow From

- External Entity to External Entity
- External Entity to Data Store
- Data Store to External Entity
- Data Store to Data Store

Advantages of DFD

- Simplifies understanding of complex systems
- Provides visual representation for better clarity
- Useful in both system design and documentation
- Easily understood by non-technical users

Disadvantages of DFD

- Can be confusing for programmers at times
- Time-consuming to create detailed DFDs
- May require training to interpret complex diagrams

How to Draw a DFD

- Understand the system functionality
- Identify external entities, processes, and data stores
- Use standard symbols for drawing
- Create Level 0 diagram first
- Develop further levels as needed
- Review and validate with stakeholders

Conclusion

- DFD is a visual tool to map information flow in systems
- Includes processes, data stores, external entities, and data flow
- Helps users and developers collaborate in system analysis and design

Entity Relationship Diagram (ERD)

What is an ER Diagram?

- Visual representation of entities and their relationships within a system
- Illustrates how data is interconnected
- Essential for designing and modeling relational databases
- An ER Diagram is a blueprint for your database. It shows entities (like 'Customer' or 'Order') and how they relate, ensuring a clear understanding before actual implementation.

Importance of ER Diagrams

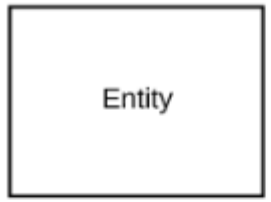
- Clarifies database structure before implementation
- Facilitates communication among stakeholders
- Identifies redundancies and inconsistencies early
- Serves as documentation for future reference
- ER Diagrams are not just technical tools; they bridge the gap between developers, analysts, and business stakeholders, ensuring everyone is aligned on the data model.

Components of ER Diagrams

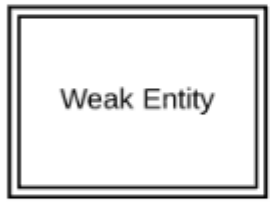
- **Entities:** Objects or concepts (e.g., Student, Course)
- **Attributes:** Details about entities (e.g., Student Name, Course ID)
- **Relationships:** Associations between entities (e.g., Enrolled In)
- Understanding these components is crucial. Entities are the nouns, attributes are the properties, and relationships are the verbs connecting entities.

Types of Entities

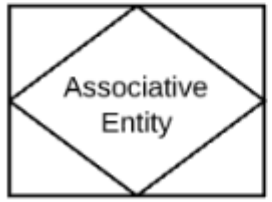
- **Strong Entities:** Exist independently
- **Weak Entities:** Depend on other entities
- **Associative Entities:** Link entities in many-to-many relationships (e.g., Enrollment)
- Recognizing entity types helps understanding how data interrelates, especially in complex databases.



Strong entity



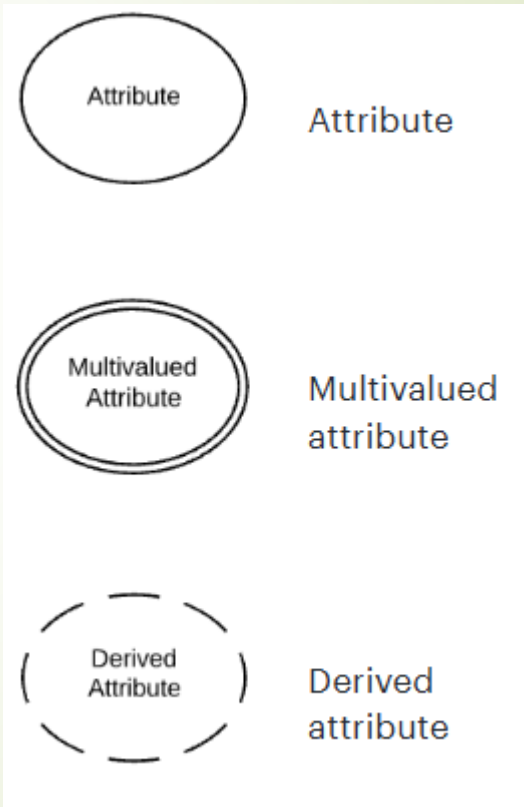
Weak entity



Associative entity

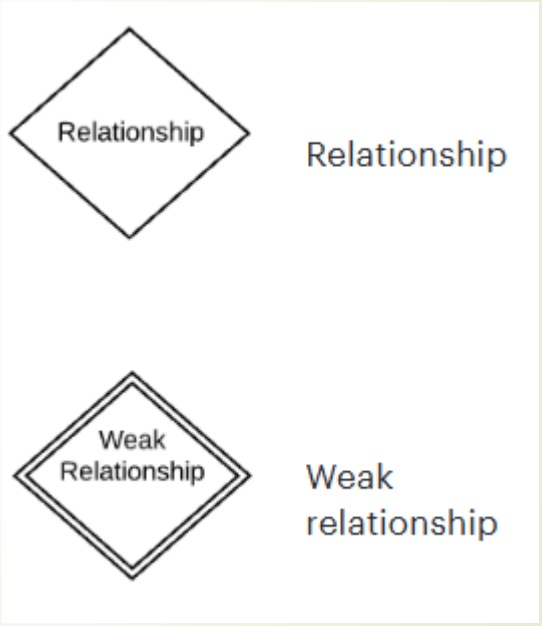
Attributes

- **Simple Attributes:** Indivisible (e.g., Age)
- **Composite Attributes:** Can be divided (e.g., Full Name)
- **Derived Attributes:** Calculated from other attributes (e.g., Age from DOB)
- **Multivalued Attributes:** Can have multiple values (e.g., Phone Numbers)



Understanding Relationships

- **One-to-One (1:1):** Each entity instance relates to one instance of another entity
- **One-to-Many (1:N):** One entity instance relates to multiple instances of another
- **Many-to-Many (M:N):** Multiple instances of entities relate to each other



Cardinality and Participation

- **Cardinality:** Specifies the number of instances in a relationship
- **Participation (Modality):** Indicates whether all instances of an entity are involved in a relationship
  - Relationships have a modality of either **"required"** or **"optional,"** which refers to whether an instance of an entity can exist without a related instance in the related entity.
  - The modality of a relationship indicates whether one entity instance is required to participate in the relationship.