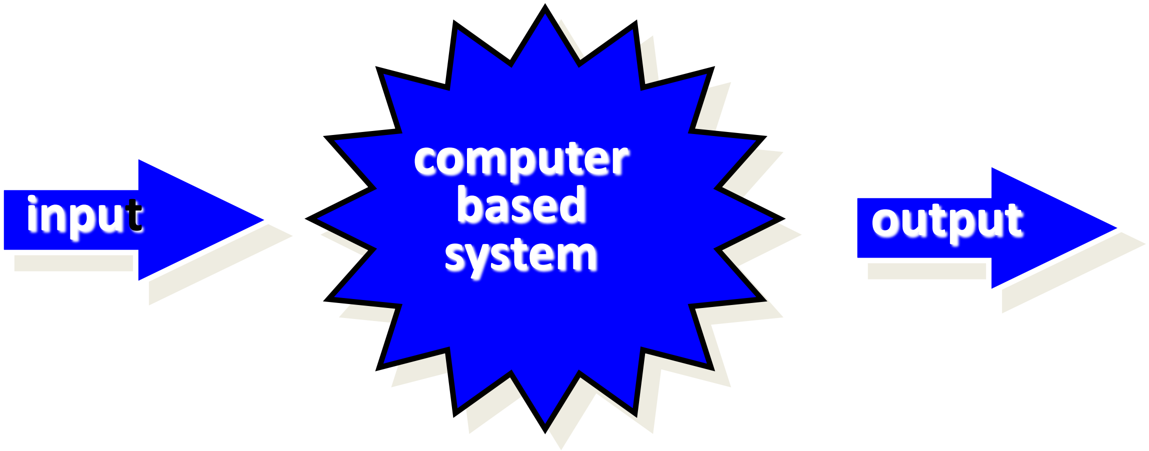
REQUIREMENTS MODELING

Flow-Oriented Modeling

- Data flow modeling is a core modeling activity in structured analysis.
- Flow oriented modeling represents **how data objects are transformed** when they move through the system.
- **data flow diagram (DFD)** is the diagrammatic form that is used to represent the **data flow.**
- The purpose of DFD is to provide a semantic bridge between user and systems developers.

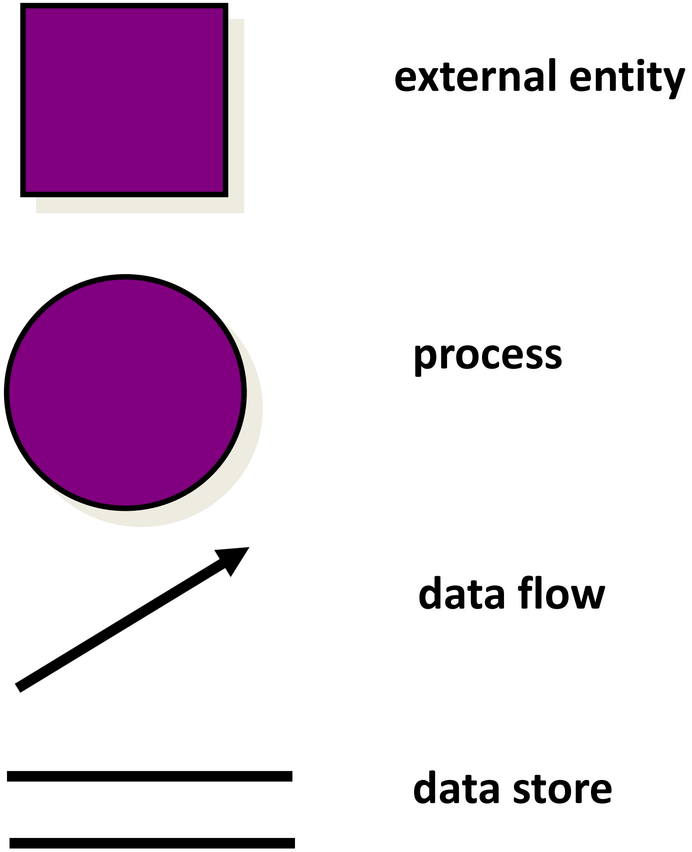
The Flow Model

Every computer-based system is an information transform ....



Creating a Data Flow Model Flow

Modeling Notations are:



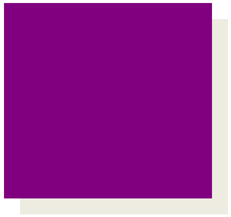
external entity

process

data flow

data store

External Entity



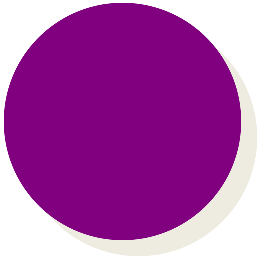
A producer or consumer of data

Examples: a person, a device, a sensor

Another example: computer-based system

Data must always originate somewhere and must always be sent to something

Process



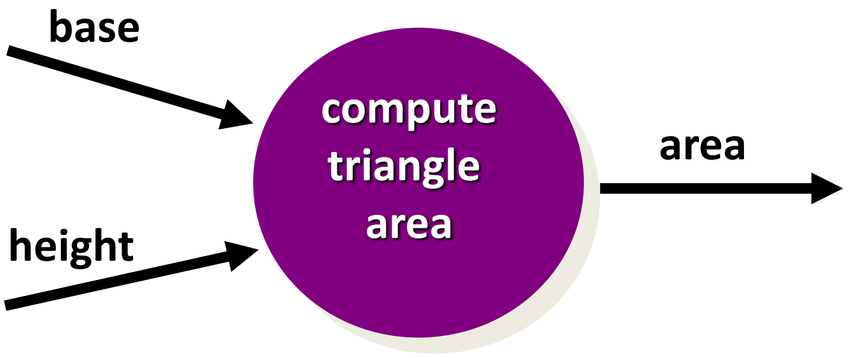
A data transformer
(changes input to output)

Examples: compute taxes, determine area, format report, display graph

Data must always be processed in some way to achieve system function

Data Flow

Data flows through a system, beginning as input and transformed into output.



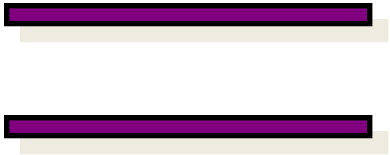
base

height

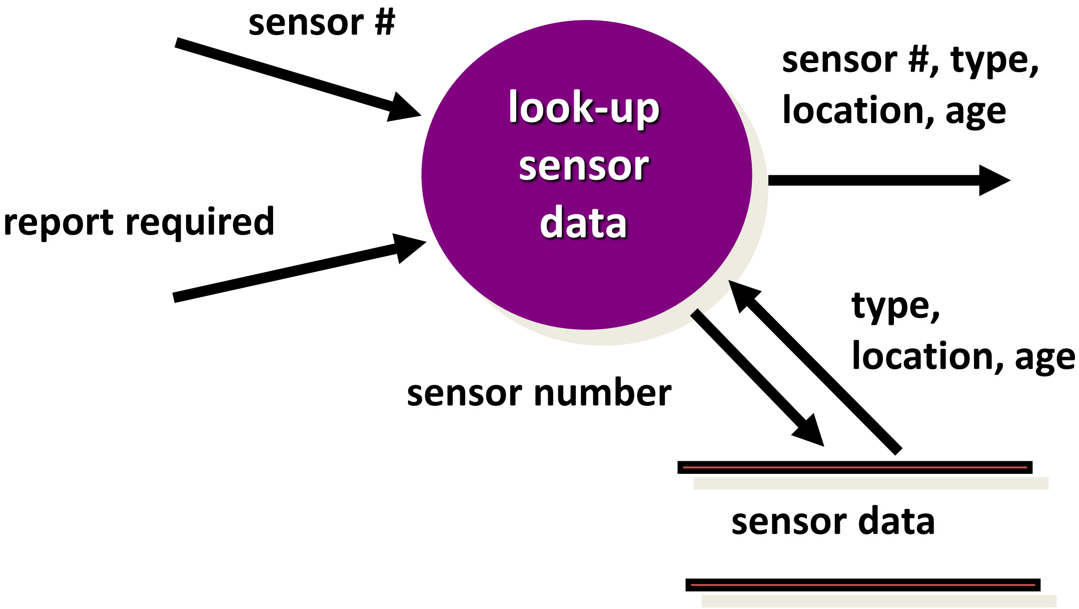
compute triangle area

area

Data Stores



Data is often stored for later use.



Creating a Data Flow Model

Guidelines for DFD::

1. The level 0 DFD should depict the software / system as a single bubble
2. Input and output should be clearly noted
3. Refinement should begin by isolating candidate processes, data objects, and data stores to be represented at the **next level**
4. All arrows and bubbles (processes) must be labeled with meaningful names
5. Information flow continuity must be maintained from level to level

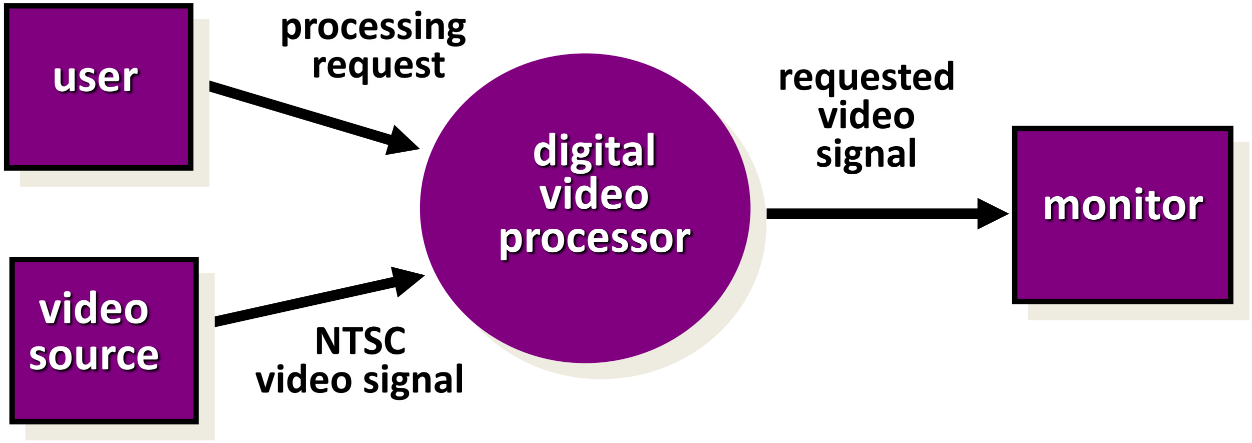
Constructing a DFD

- Review user scenarios and/or the data model to isolate data objects and use a grammatical parse to determine “operations”

- Determine external entities (producers and consumers of data)

- Create a level 0 DFD

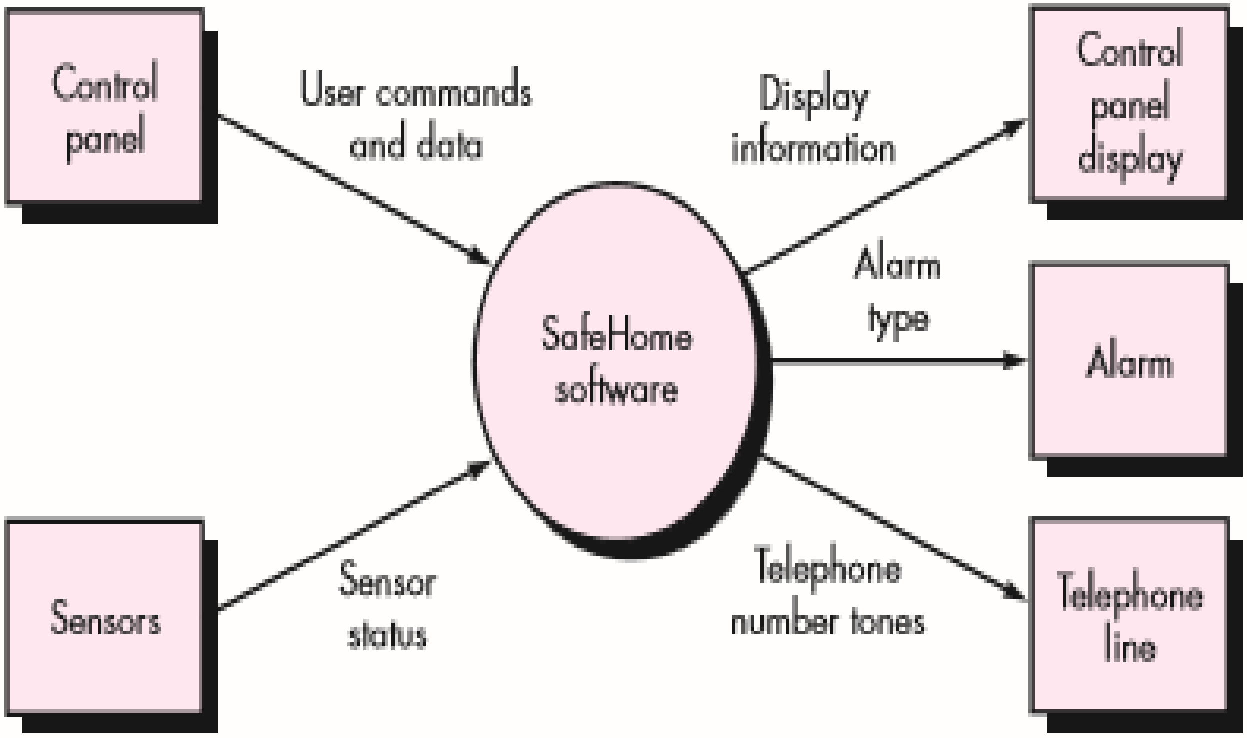
Level-0 DFD --example



Another Example for level-0 DFD

FIGURE 7.1

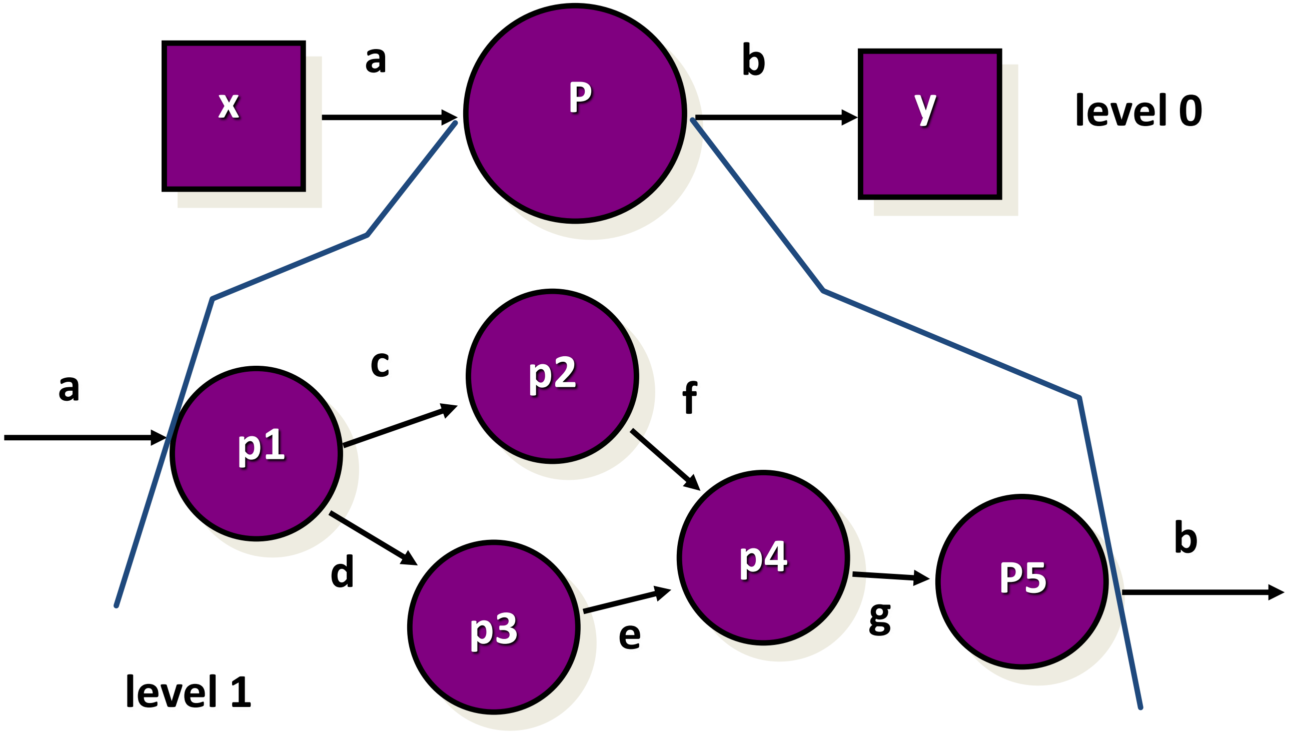
Context-level DFD for the *SafeHome* security function



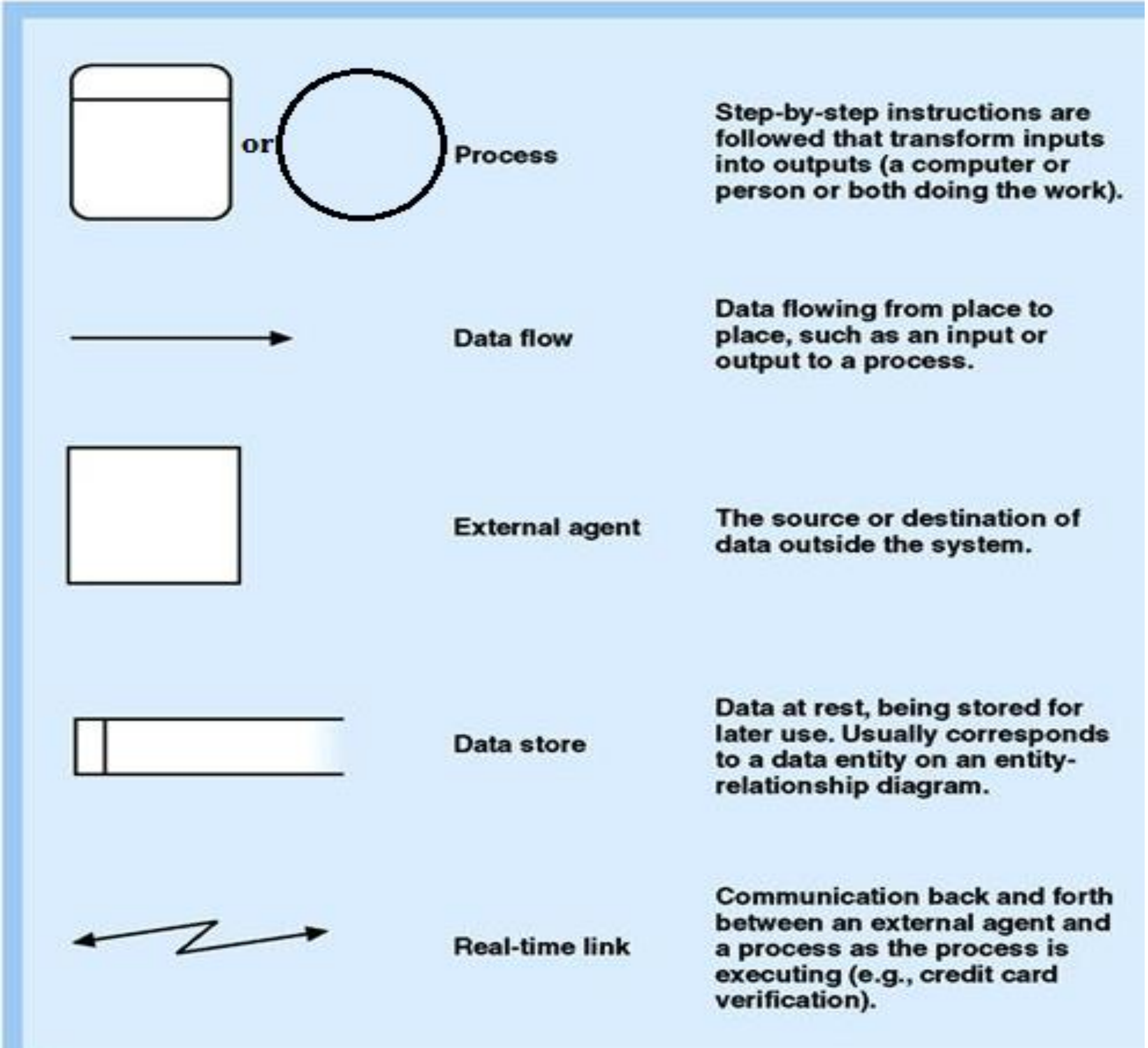
Constructing a Level -2 DFD

- Write a narrative describing the transform
- Parse to determine next level transforms
- “Balance” the flow to maintain data flow continuity
- Develop a level -1 DFD
- Use a 1:5 (approx.) expansion ratio

The Data Flow Hierarchy

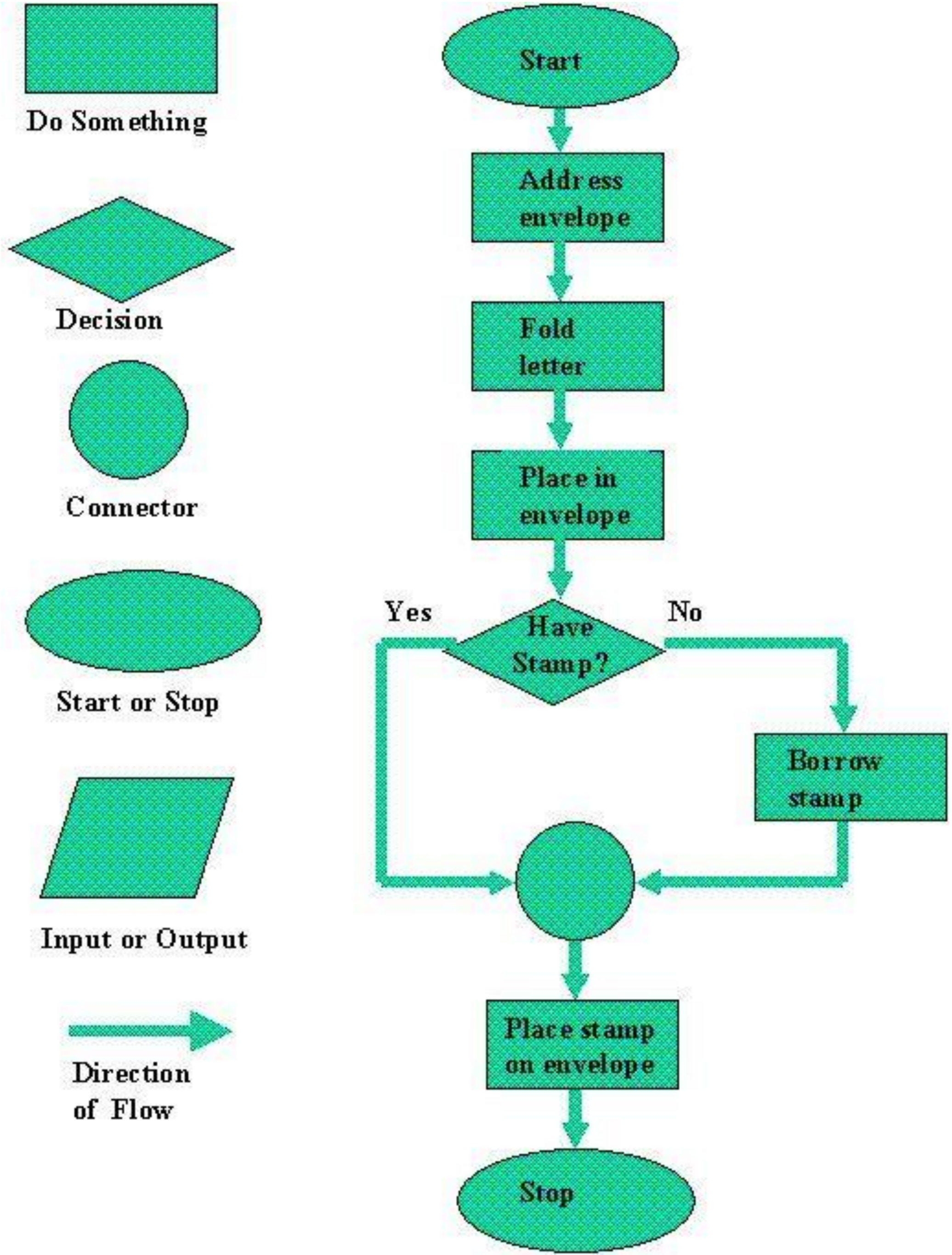


Summary: DFD symbols



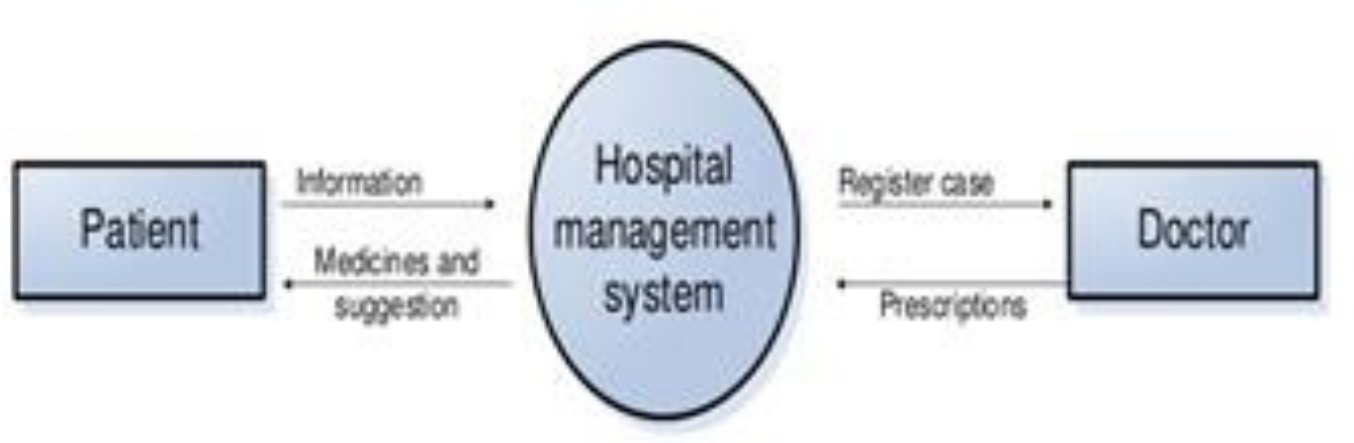
or	Process	Step-by-step instructions are followed that transform inputs into outputs (a computer or person or both doing the work).
	Data flow	Data flowing from place to place, such as an input or output to a process.
	External agent	The source or destination of data outside the system.
	Data store	Data at rest, being stored for later use. Usually corresponds to a data entity on an entity-relationship diagram.
	Real-time link	Communication back and forth between an external agent and a process as the process is executing (e.g., credit card verification).

Flow chart notations→

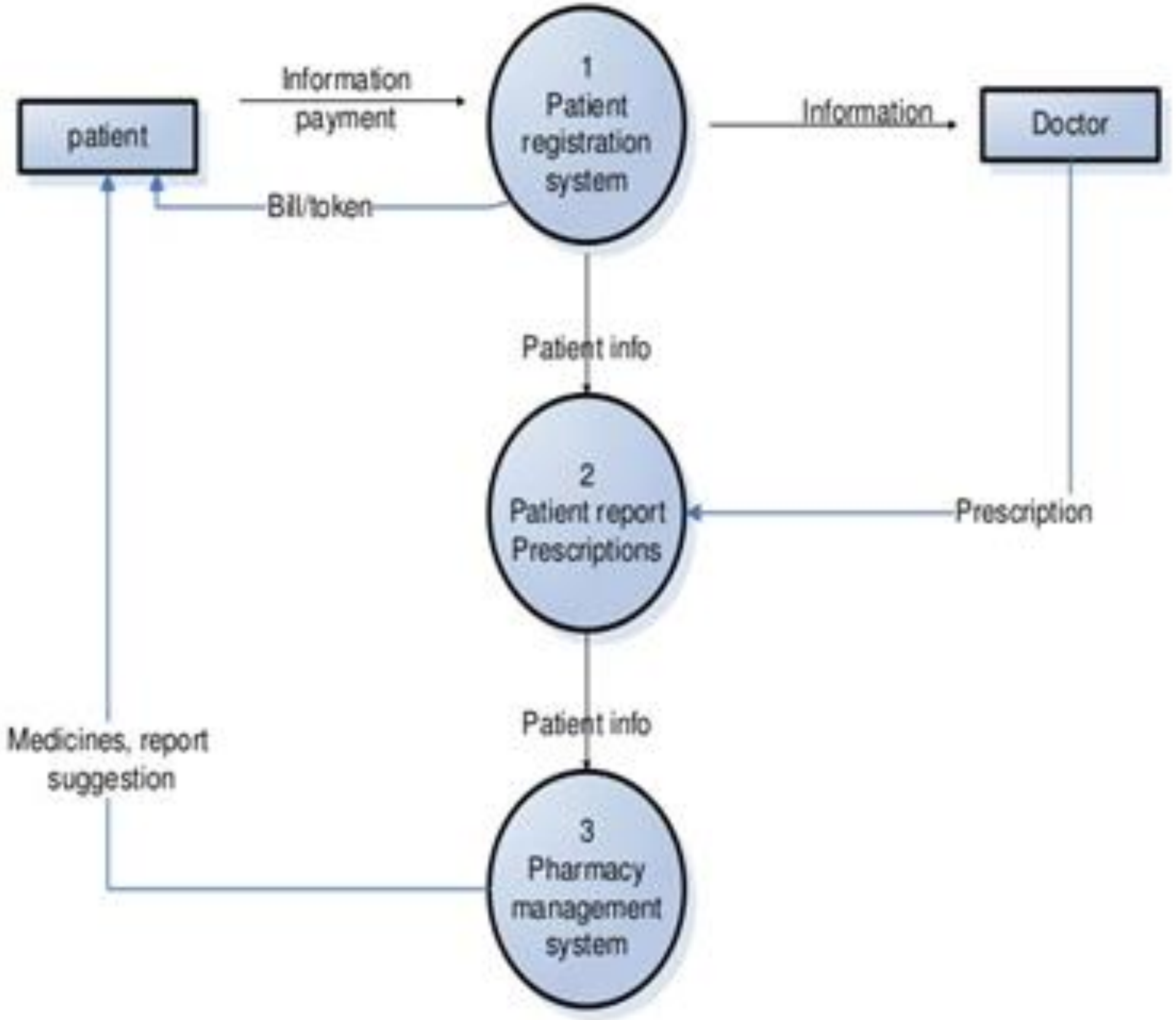


DFD: Hospital Management System

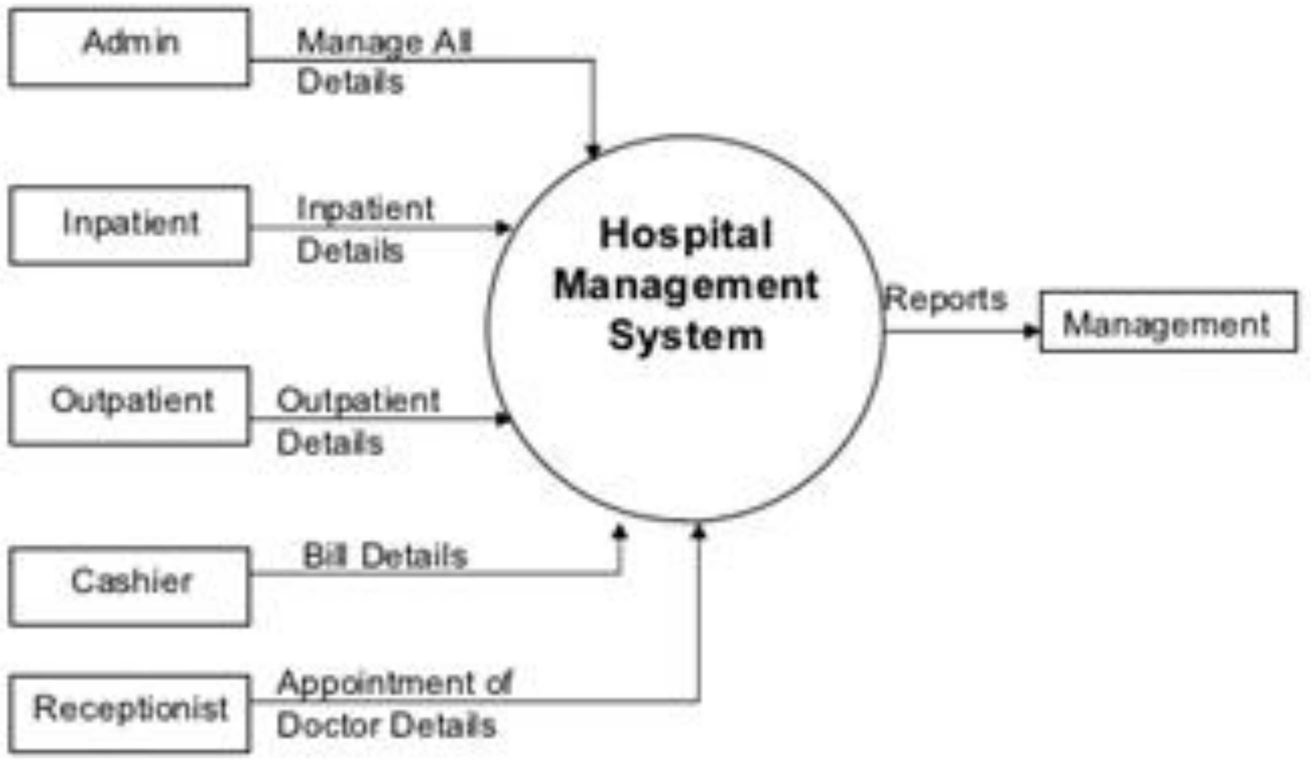
Level-0 DFD



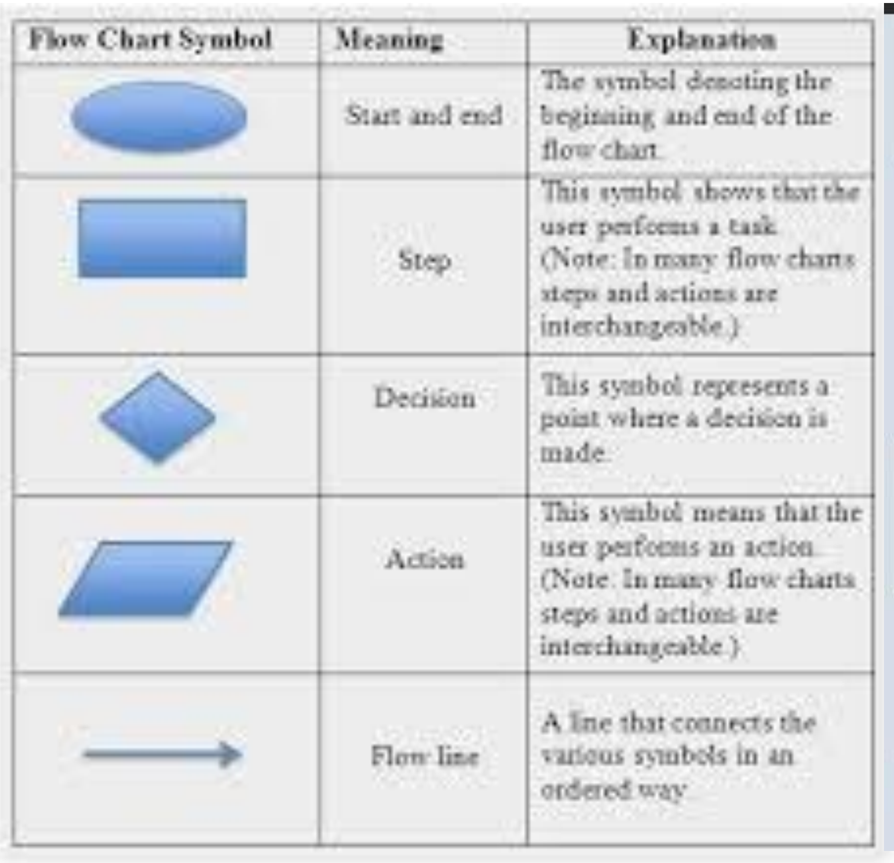
DFD: Hospital management system



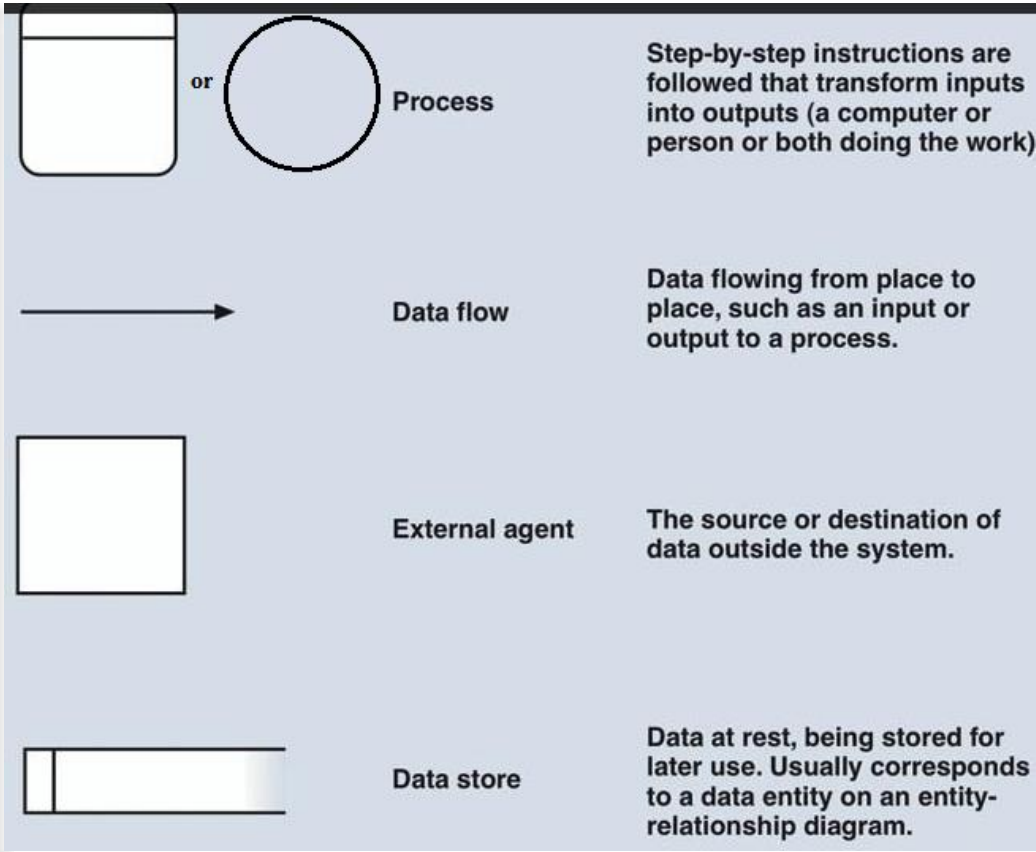
Level 0 DFD – Hospital Management System



Flow chart Vs DFD



Flow Chart Symbol	Meaning	Explanation
⬭	Start and end	The symbol denoting the beginning and end of the flow chart.
▭	Step	This symbol shows that the user performs a task. (Note: In many flow charts steps and actions are interchangeable.)
◇	Decision	This symbol represents a point where a decision is made.
▱	Action	This symbol means that the user performs an action. (Note: In many flow charts steps and actions are interchangeable.)
→	Flow line	A line that connects the various symbols in an ordered way.



▭ or ○	Process	Step-by-step instructions are followed that transform inputs into outputs (a computer or person or both doing the work)
→	Data flow	Data flowing from place to place, such as an input or output to a process.
□	External agent	The source or destination of data outside the system.
▭	Data store	Data at rest, being stored for later use. Usually corresponds to a data entity on an entity-relationship diagram.

Use Case Diagram

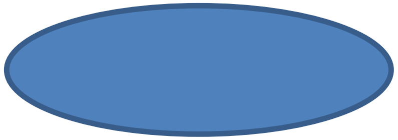
- A system involves a **set of use cases** and **a set of actors.**

- Set of use cases:- shows the complete functionality of the system at some level of detail.

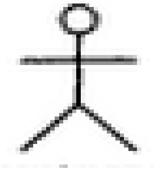
- Set of actors :- represents the complete set of objects that the system can serve.

Use Case diagram elements

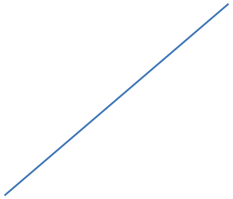
- Use case:



- Actor:



- Association:



Guidelines for use case models

1. **First determine the system boundary:-**It is impossible to identify use cases or actors if the system boundary is un clear.

2. **Ensure that actors are focused:-**Each actor should have a single, coherent purpose.

3. **Each use case must provide value to users:-**use case should represent complete transaction that provide value to user and should not define too narrowly.

Cont…

4. **Relate use case and actors:-**every use case should have at least one actor and every actor should participate in atleast one use case.

5. **Remember that use cases are informal:-**do not obsess formalism in specifying use cases.

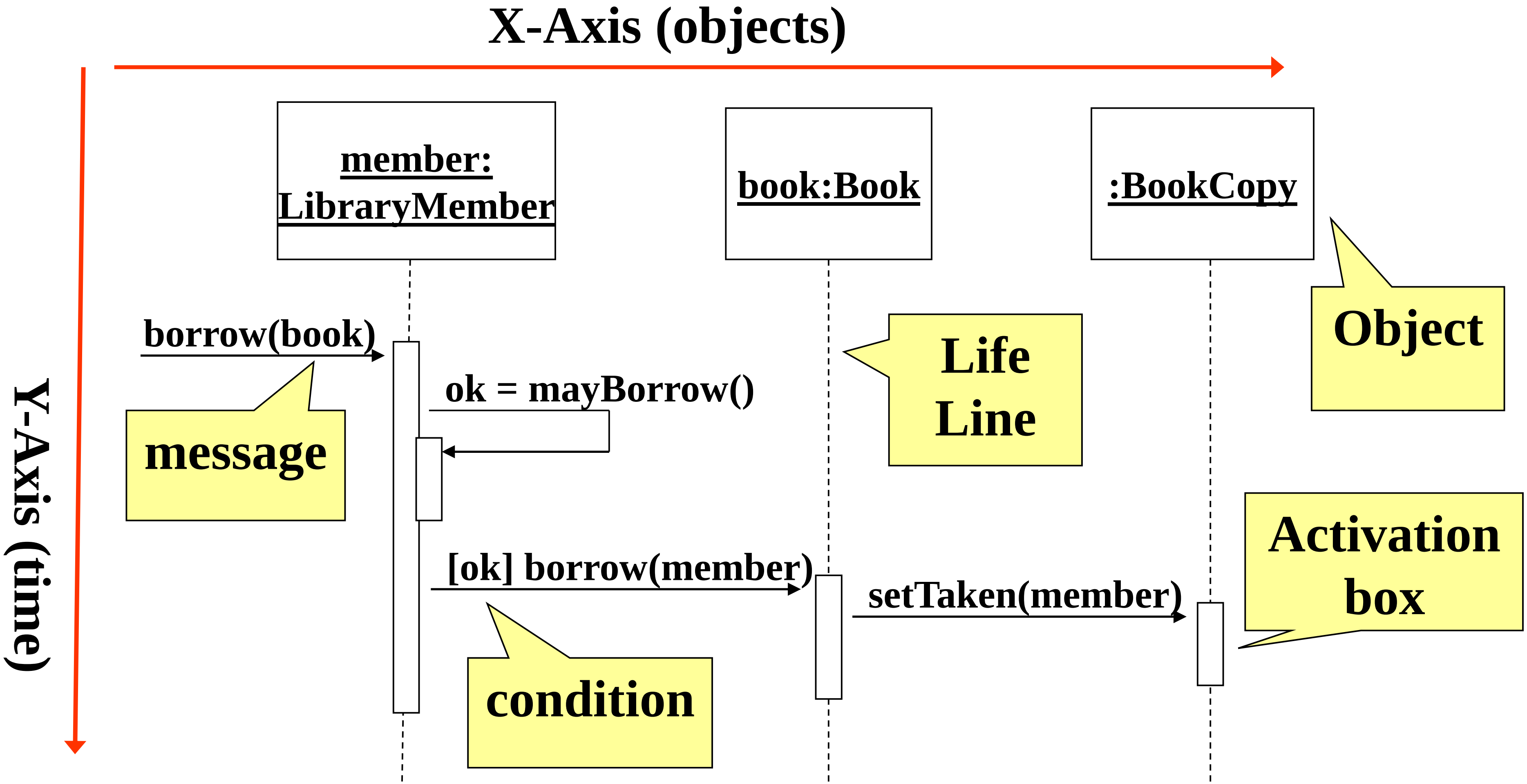
6. **Use cases can be structured:-**For large systems, use cases can be built out of smaller fragments using relationship.

Scenario based model

- At early stages of development, Scenarios are expressed at a high level, later stages we can show exact messages.

- First step of writing a scenario is to identify the objects exchanging messages.
- Determine the sender and receiver of each message and sequence of messages.
- Add activities for internal computations.

A Sequence Diagram example



Guidelines for Sequence model

1. Prepare at least one scenario per use case:- the steps in the scenario should be logical commands, not individual button clicks.
2. Abstract the scenario into sequence diagrams:- Sequence diagrams clearly shows the contribution of each actor.
3. Divide complex interactions:-Break large interaction into their constituent tasks and prepare a sequence diagram for each of them.
4. Prepare a sequence diagram for error condition:- shows the system response to error condition.

Tools to draw diagram

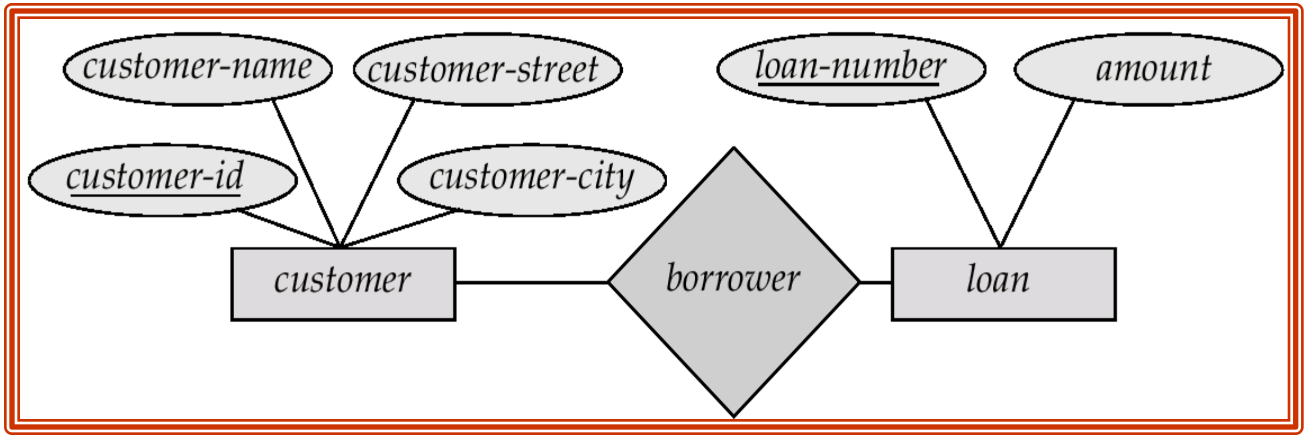
- StarUML → for usecase diagram, Sequence diagram
- Creately → online tool to draw Data Flow Diagram (DFD)

ER DIAGRAM

Basic Concepts

- **Entity set** – an abstraction of similar things, e.g. cars, students
    - An entity set contains many entities
- **Attributes**: common properties of the entities in a entity sets
- **Relationship** – specify the relations among entities from two or more entity sets

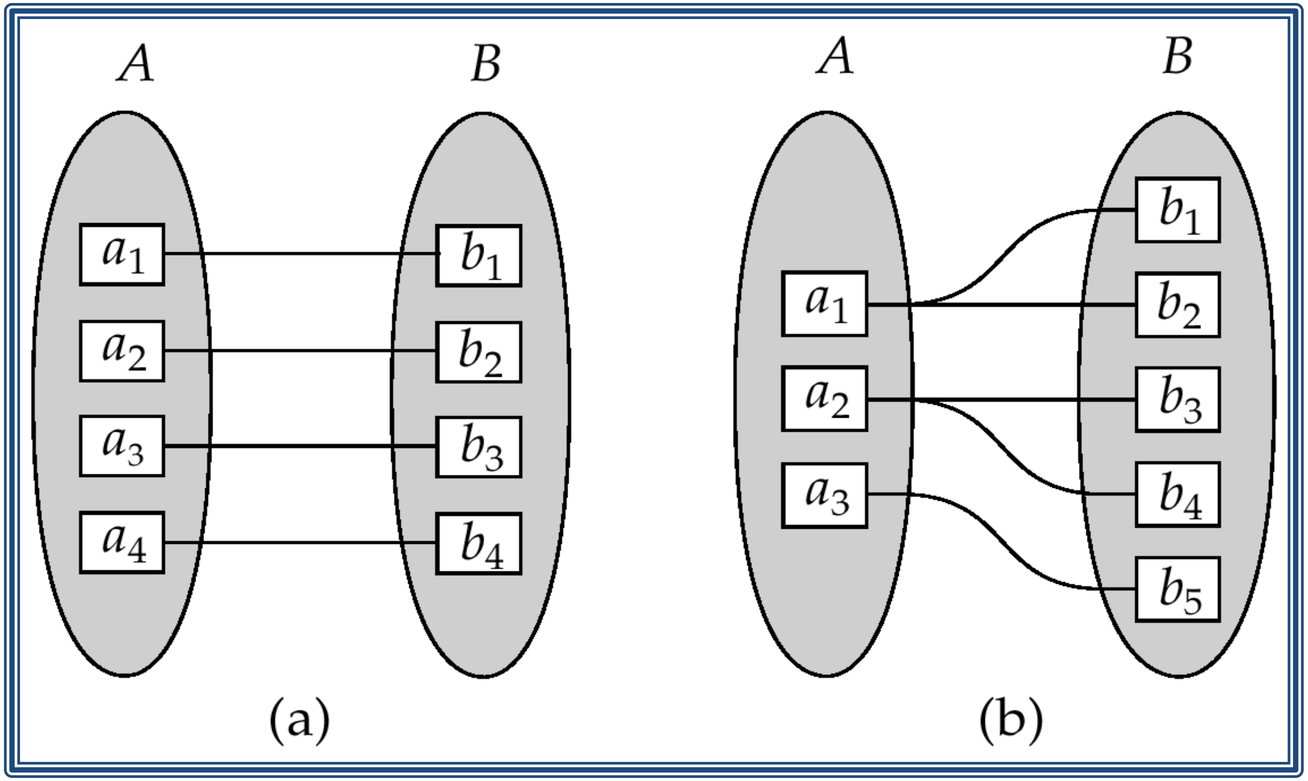
An Example


customer-name
customer-street
loan-number
amount
customer-id
customer-city
customer
borrower
loan

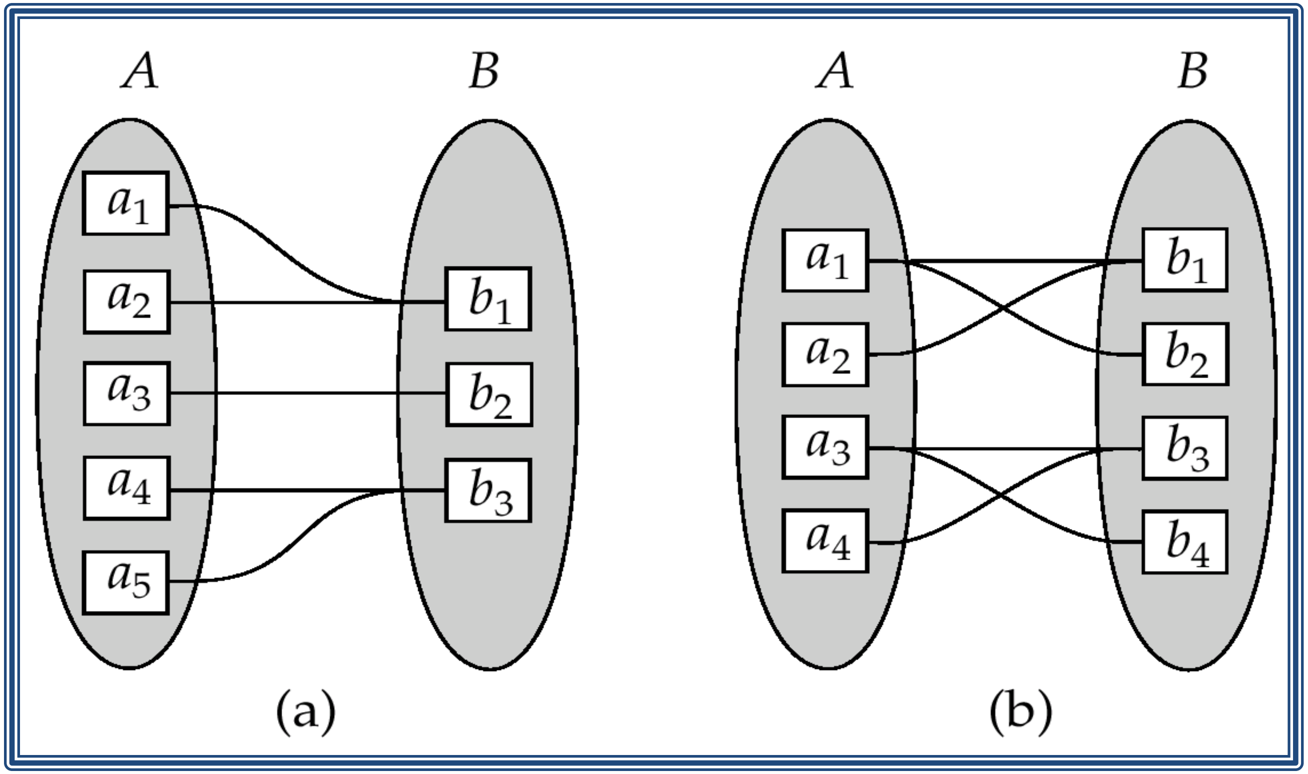
Relationship

- The degree of a relationship = the number of entity sets that participate in the relationship
  - Mostly binary relationships
  - Sometimes more
- Mapping cardinality of a relationship
  - 1 –1
  - 1 – many
  - many – 1
  - Many-many

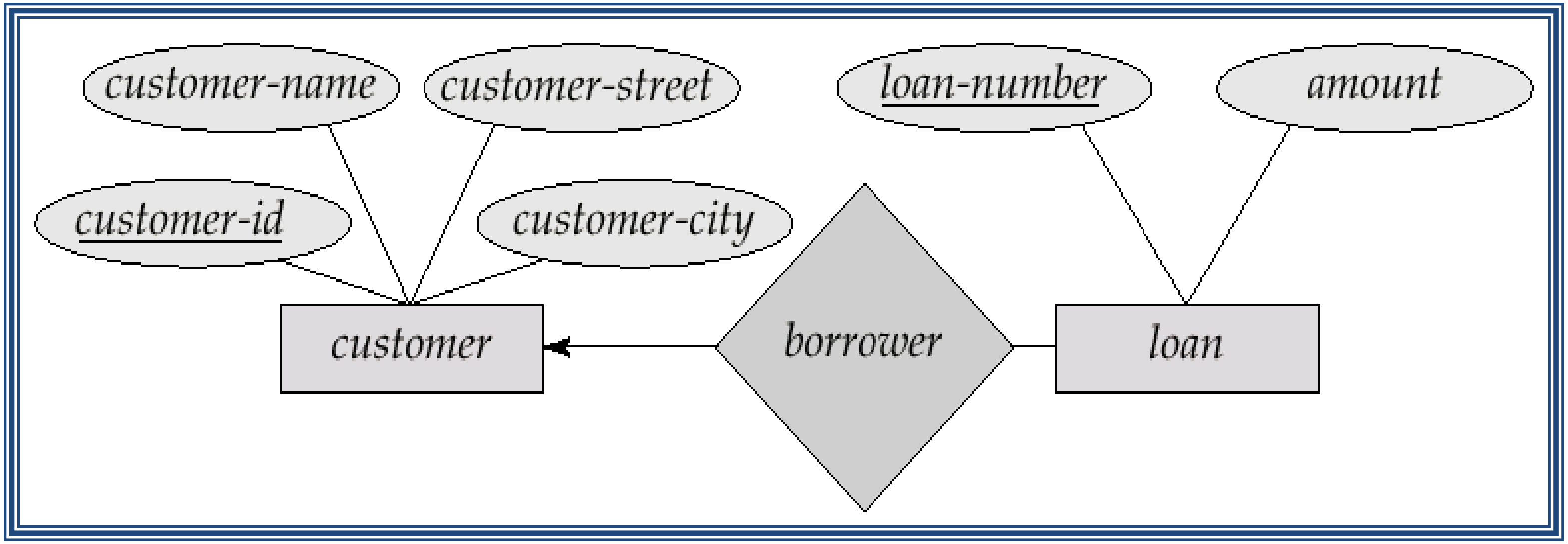
One-One and One-Many



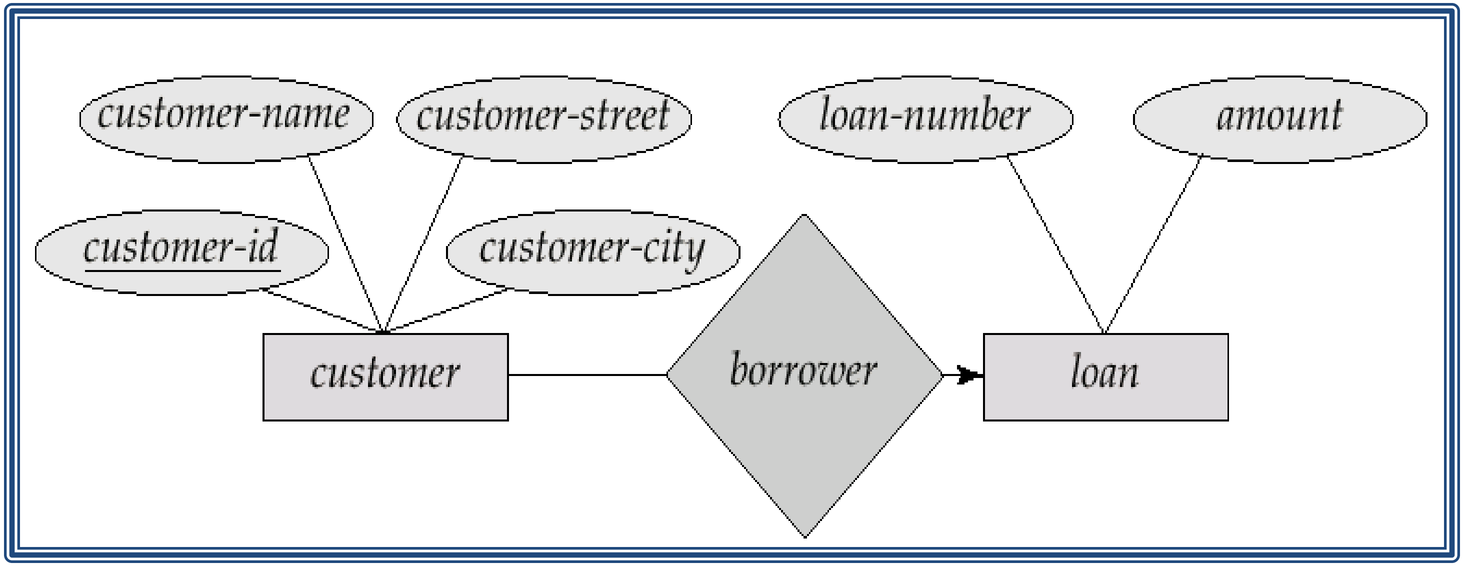
Many-one and many-many



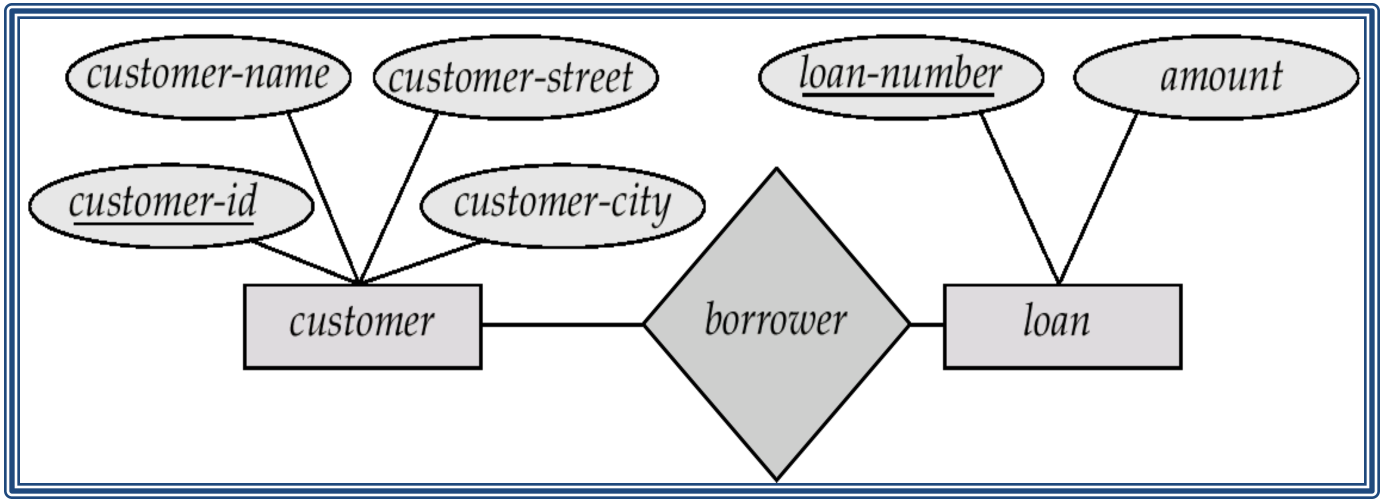
1- many


customer-name
customer-street
loan-number
amount
customer-id
customer-city
customer
borrower
loan

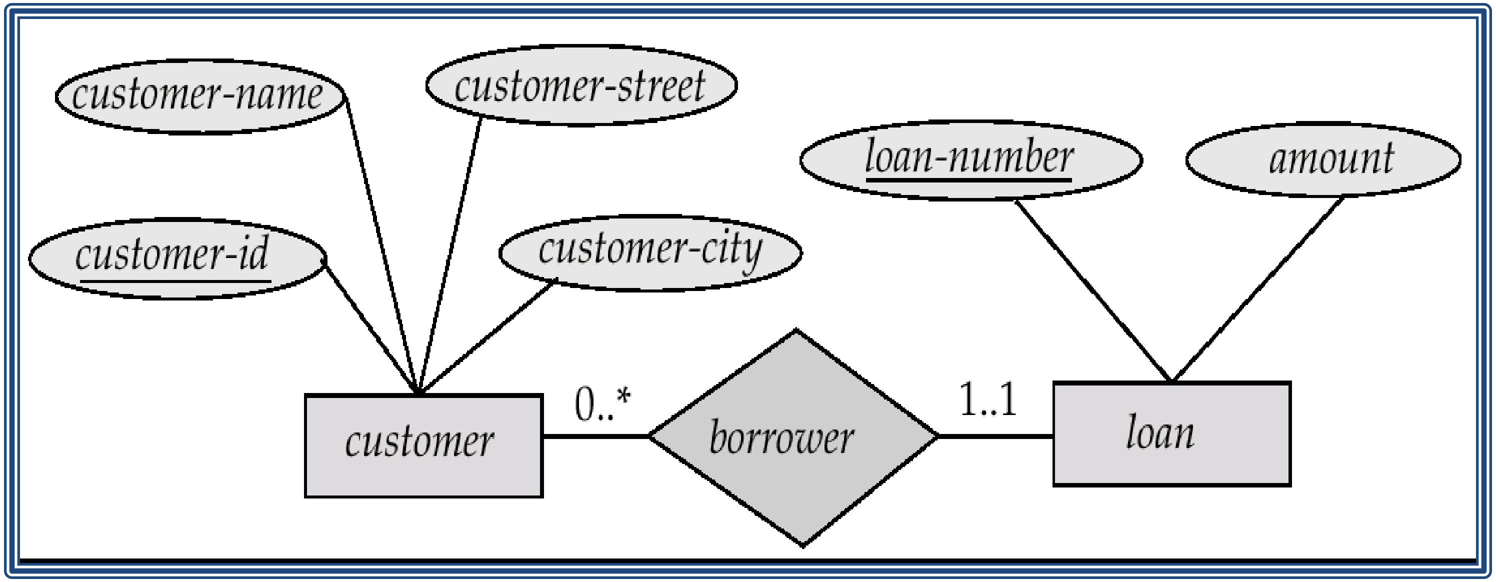
Many - 1


customer-name
customer-street
loan-number
amount
customer-id
customer-city
customer
borrower
loan

Many - many


customer-name
customer-street
customer-id
customer-city
customer
borrower
loan-number
amount
loan

Alternative Cardinality Specification



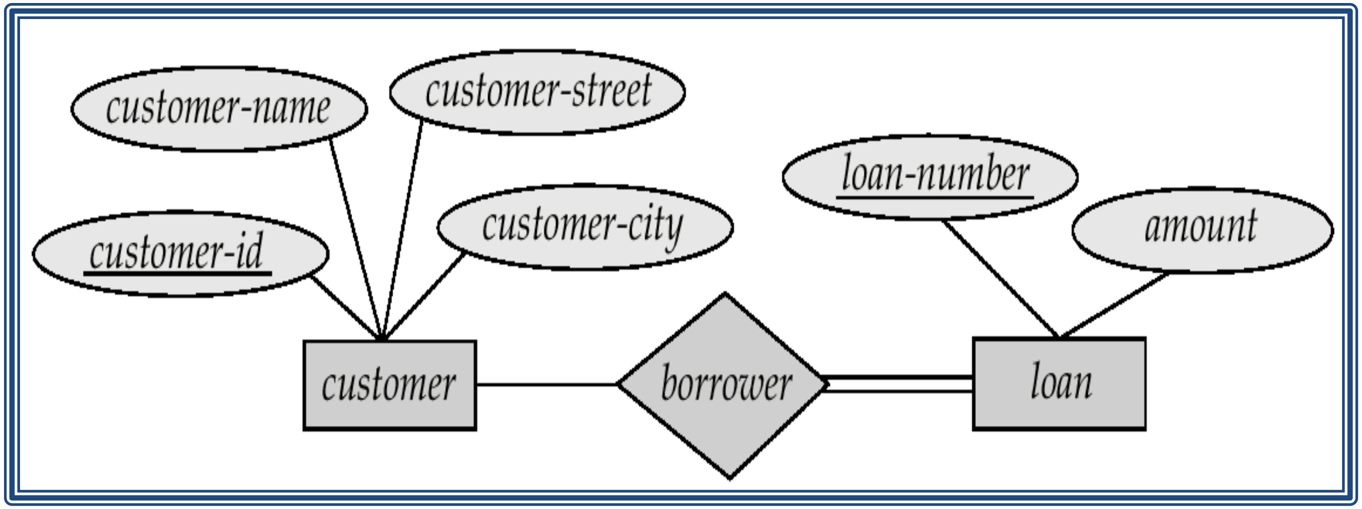
Note on Mapping Cardinality

- Both many and 1 include 0
  - Meaning some entity may not participate in the relationship

Total Participation

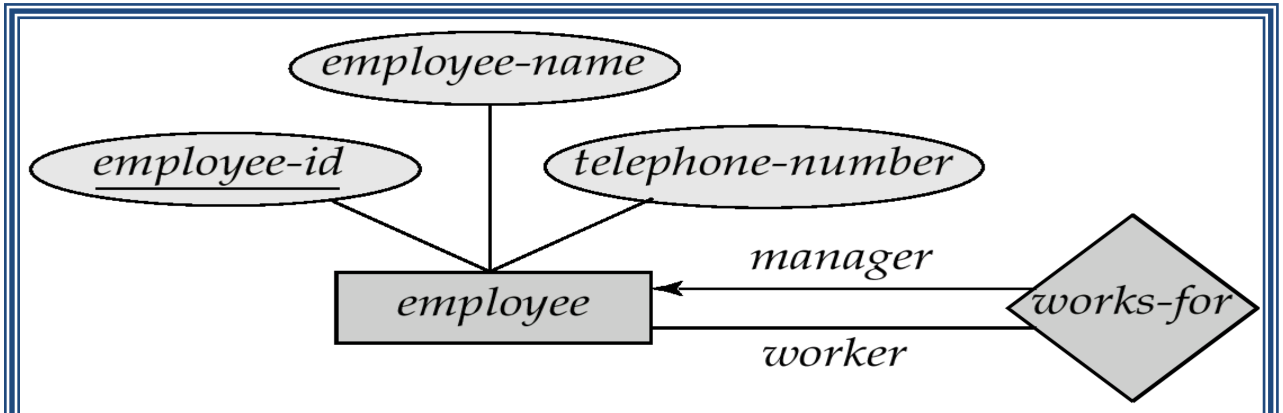
- When we require all entities to participate in the relationship (total participation), we use double lines to specify

Every loan has to have at least one customer



Self Relationship

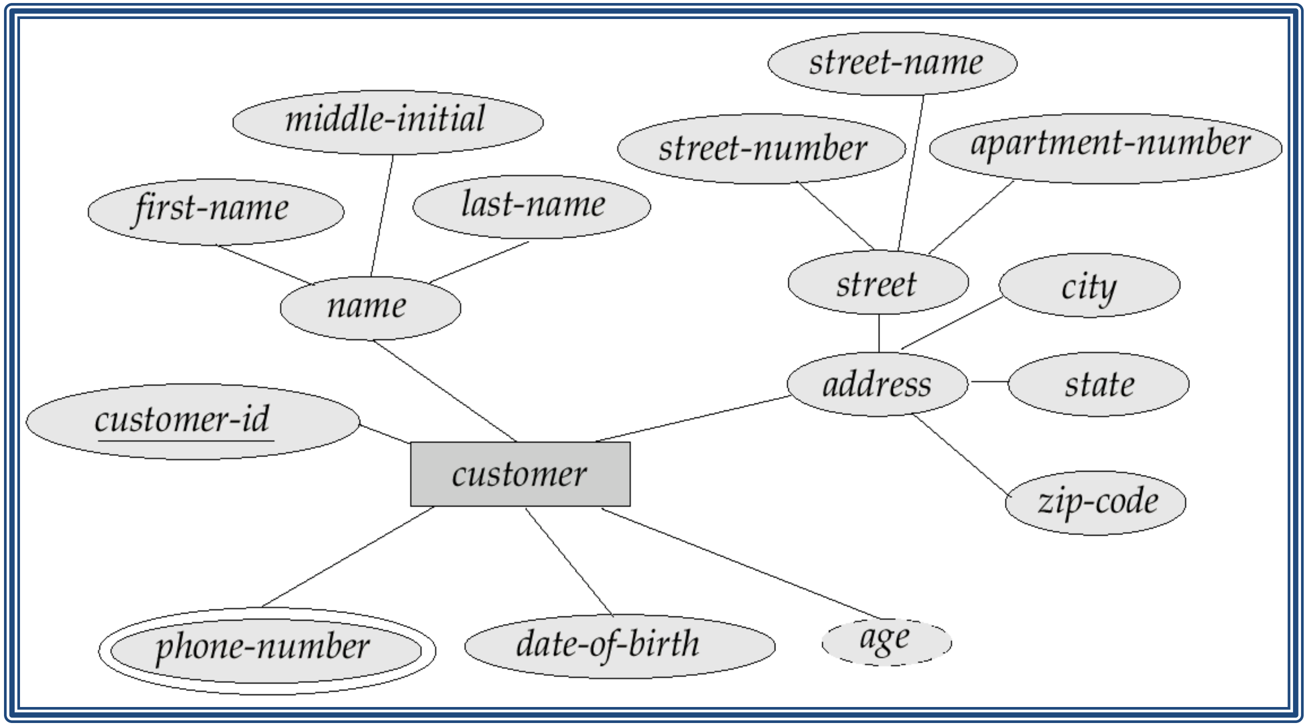
- Sometimes entities in a entity set may relate to other entities in the same set. Thus self relationship
- Here employees mange some other employees
- The labels “manger” and “worker” are called *roles* the self relationship



Attributes

- Both entity sets and relationships can have attributes
- Attributes may be
  - Composite
  - Multi-valued (double ellipse)
  - Derive (dashed ellipse)

Another Example



Keys

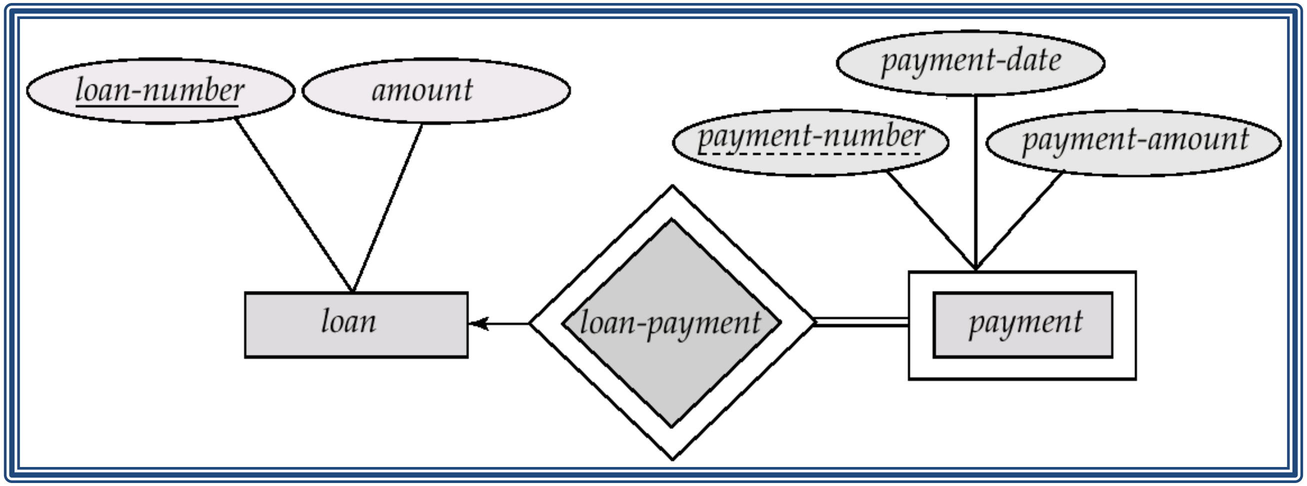
- A *super key* of an entity set is a set of one or more attributes whose values uniquely determine each entity.
- A *candidate key* of an entity set is a minimal super key
- Although several candidate keys may exist, one of the candidate keys is selected to be the *primary key*.

Weak Entity Set

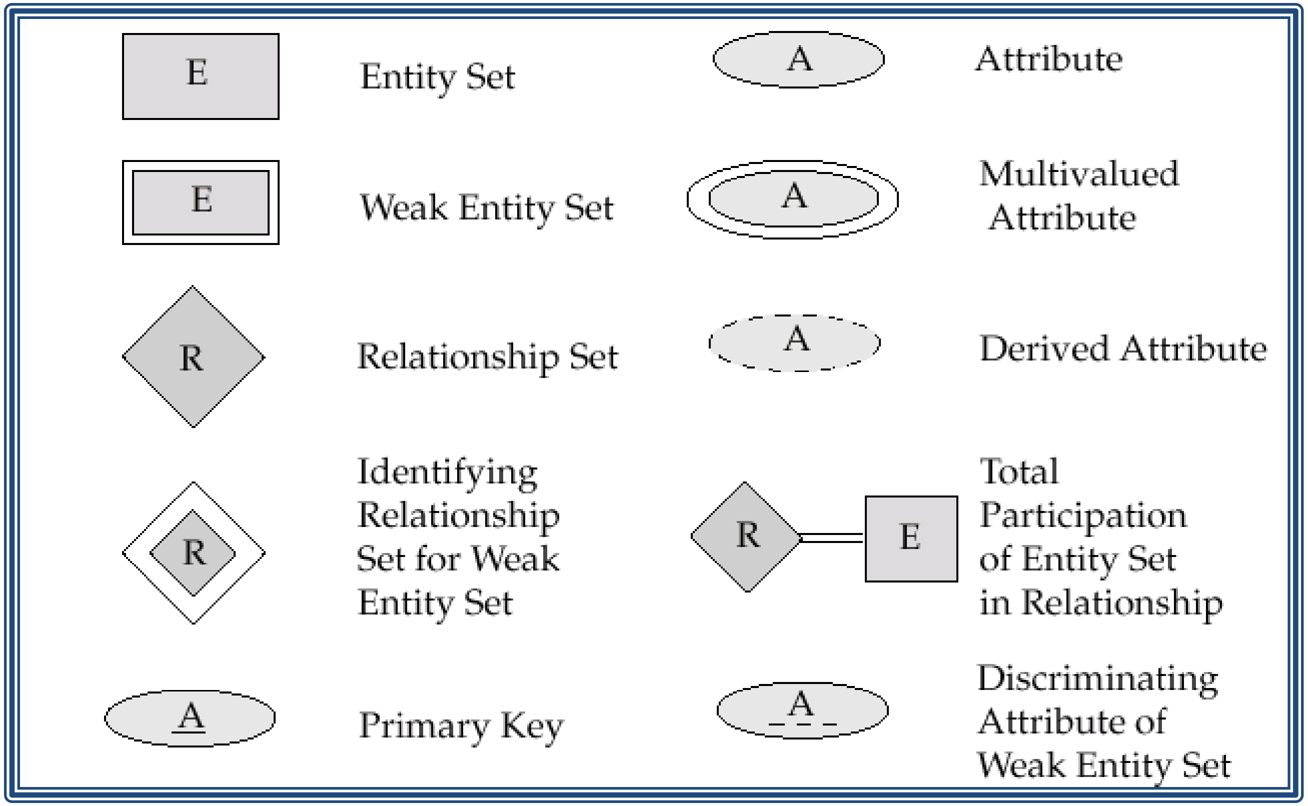
- Some entity sets in real world naturally depend on some other entity set
  - They can be uniquely identified only if combined with another entity set

Weak Entity Set Notations

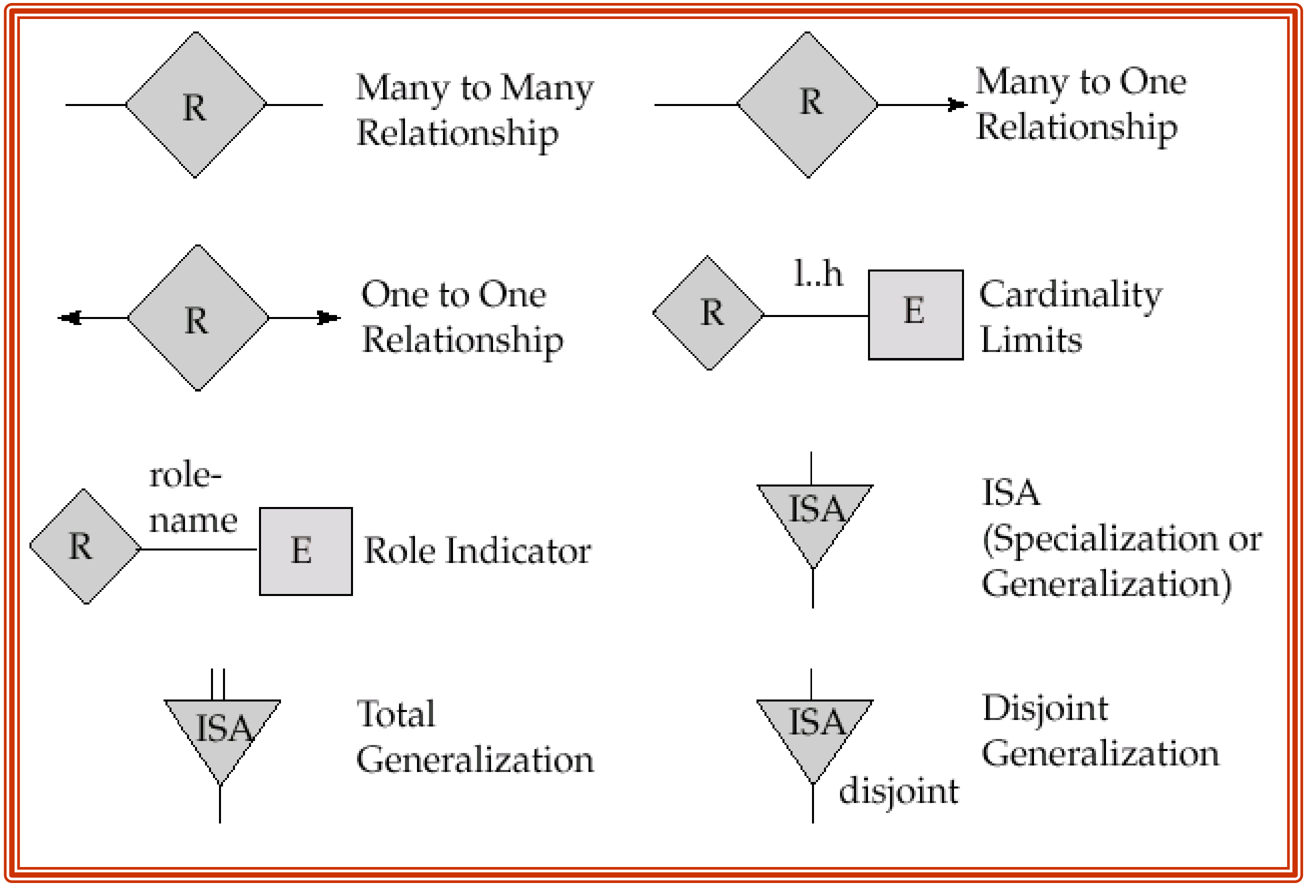
Double rectangles for weak entity set
Double diamond for weak entity relationship



Notations


E
Entity Set
A
Attribute
E
Weak Entity Set
A
Multivalued Attribute
R
Relationship Set
A
Derived Attribute
R
Identifying Relationship Set for Weak Entity Set
R
E
Total Participation of Entity Set in Relationship
A
Primary Key
A
Discriminating Attribute of Weak Entity Set

Notations


R
Many to Many Relationship
R
Many to One Relationship
R
One to One Relationship
R
l..h
E
Cardinality Limits
role-name
R
E
Role Indicator
ISA
ISA (Specialization or Generalization)
ISA
Total Generalization
ISA
disjoint
Disjoint Generalization

How to Draw ER Diagrams

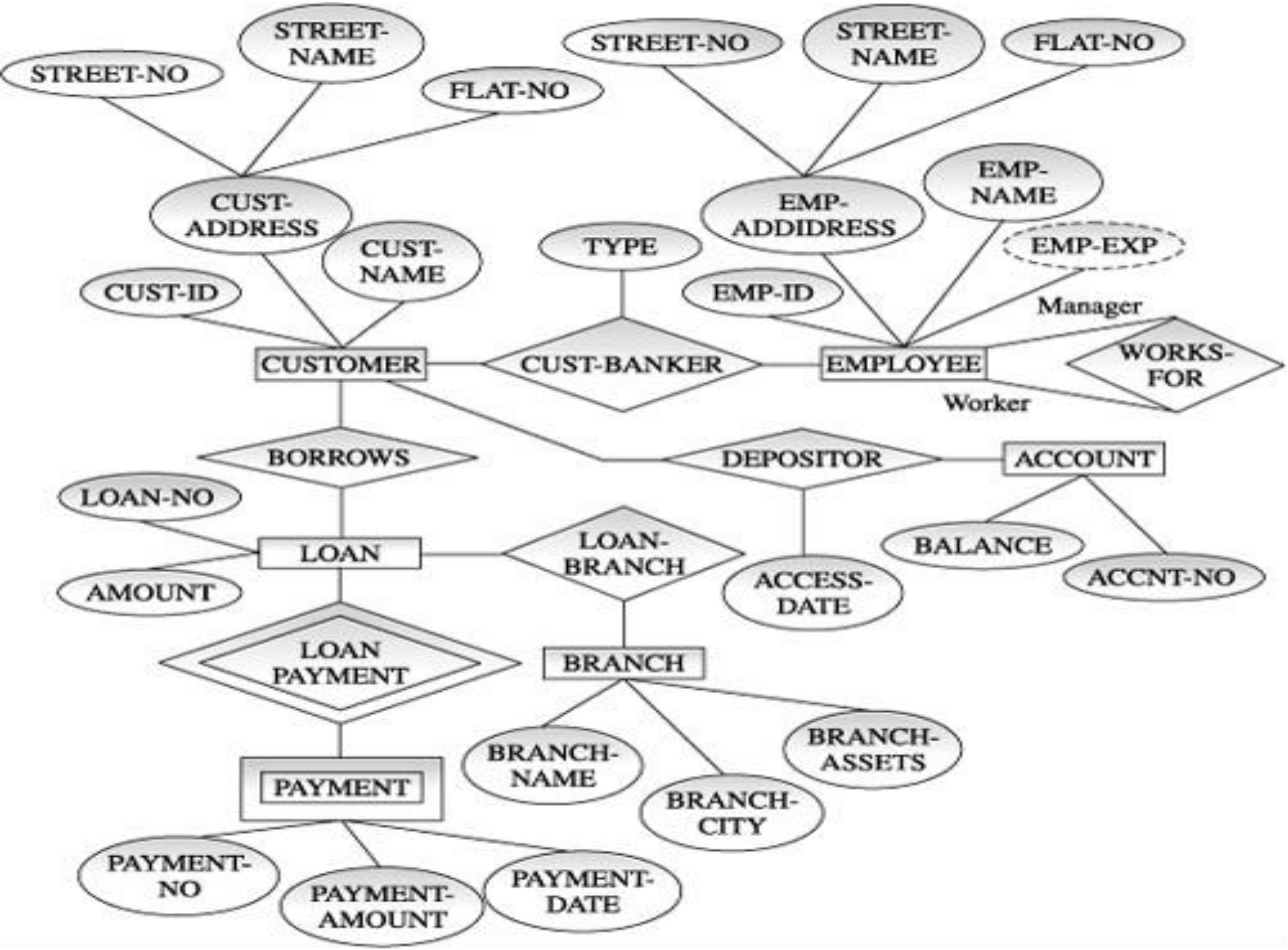
1. Identify all the entities in the system. An entity should appear only once in a particular diagram. Create rectangles for all entities and name them properly.
2. Identify relationships between entities. Connect them using a line and add a diamond in the middle describing the relationship.
3. Add attributes for entities. Give meaningful attribute names so they can be understood easily.

ER Diagram Best Practices

1. Provide a precise and appropriate name for each entity, attribute, and relationship in the diagram. Terms that are simple and familiar always beats vague, technical-sounding words. In naming entities, remember to use singular nouns. However, adjectives may be used to distinguish entities belonging to the same class (part-time employee and full-time employee, for example). Meanwhile attribute names must be meaningful, unique, system-independent, and easily understandable.
2. Remove vague, redundant or unnecessary relationships between entities.
3. Never connect a relationship to another relationship.
4. Make effective use of colors. You can use colors to classify similar entities or to highlight key areas in your diagrams.

ER Diagram

- Example--- Bank database


STREET-NAME
STREET-NO
FLAT-NO
CUST-ADDRESS
CUST-NAME
CUST-ID
STREET-NO
STREET-NAME
FLAT-NO
EMP-ADDIDRESS
EMP-NAME
EMP-EXP
EMP-ID
TYPE
Manager
CUSTOMER
CUST-BANKER
EMPLOYEE
WORKS-FOR
Worker
BORROWS
DEPOSITOR
ACCOUNT
LOAN-NO
LOAN
LOAN-BRANCH
BALANCE
ACCNT-NO
AMOUNT
ACCESS-DATE
LOAN PAYMENT
BRANCH
BRANCH-NAME
BRANCH-CITY
BRANCH-ASSETS
PAYMENT
PAYMENT-NO
PAYMENT-AMOUNT
PAYMENT-DATE

Tools- used to draw ER Diagrams

1. ERDPLUS
2. LUCID Chart
3. SmartDraw