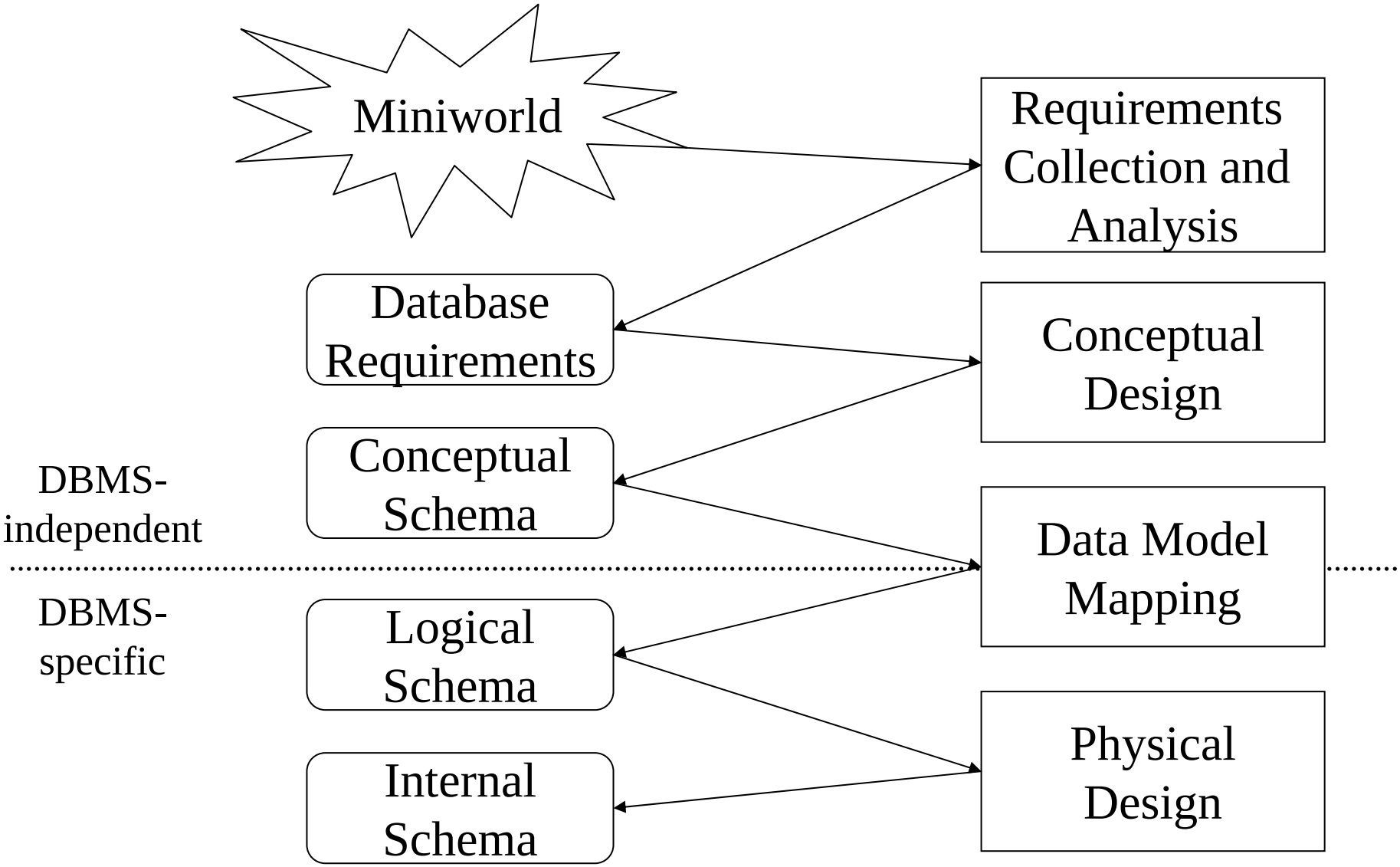
Conceptual Design: Outline

- The Data Modelling Process
- The Entity-Relationship Model
- Mapping Entity-Relationship Model to Tables

Data Modelling Process



Conceptual Design: Outline

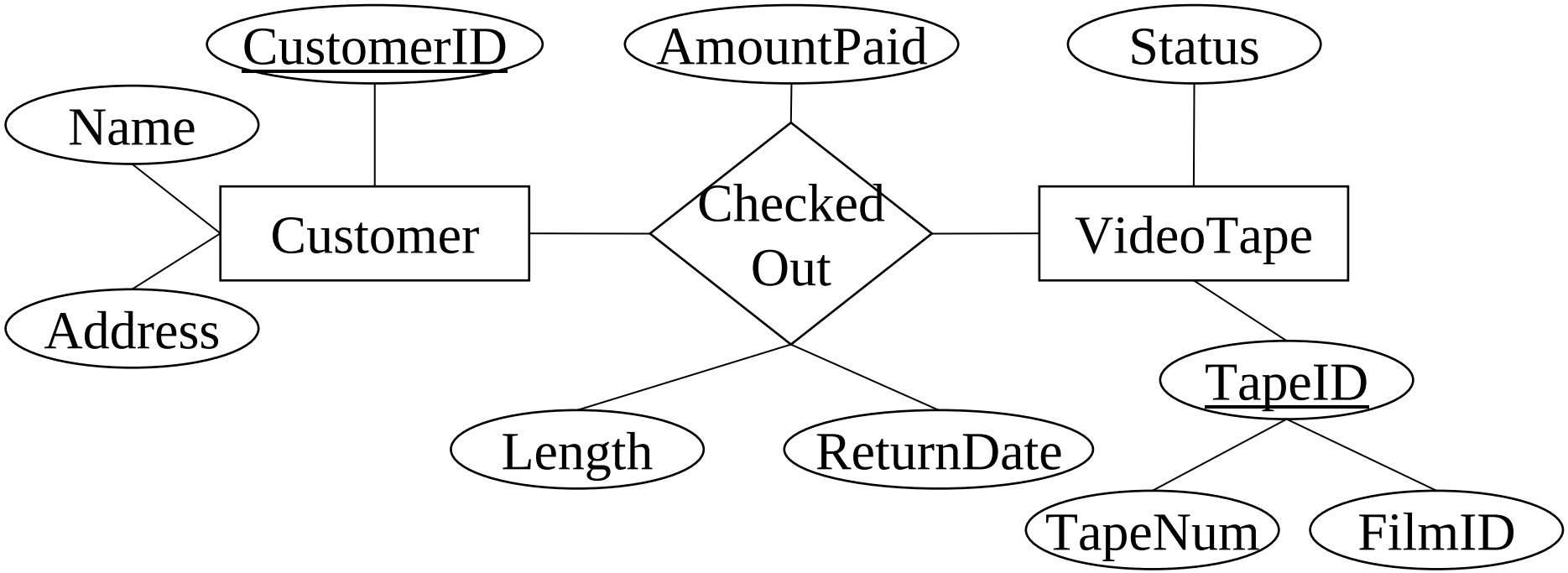
- The Data Modelling Process
- The Entity-Relationship Model
  - Entities
  - Attributes
  - Relationships
    - Binary/tertiary/n-ary
    - Roles
    - Participation
  - Keys
  - Weak Entity Types
- Mapping Entity-Relationship Model to Tables

Entity-Relationship Model

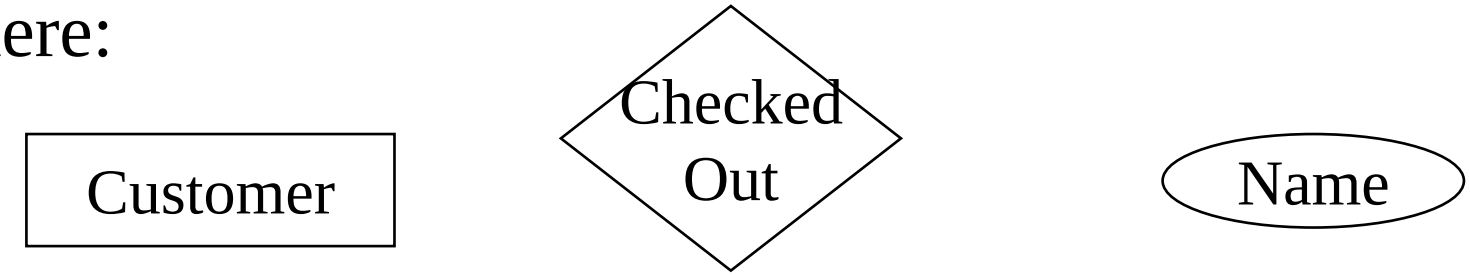
- The entity-relationship (ER) model is frequently used as a design model, such as in CASE tools, e.g., Dia and Visio.
- A *database* can be modelled as:
  - a collection of entities, and
  - relationships among entities.
- The ER model is used in conceptual design.

- Result of ER modelling is an *ER Schema* or an *ER Diagram*.

Core in Conceptual Design: ER-Diagram



- Where:



Entity Type Relationship Type Attribute

Entities

- An *entity* is an object that exists and is distinguishable from other objects.
  - Example: specific person, department, event, plant
- An entity is represented by a set of *attributes*.
  - Example: person-name, address, phone#, cpr#
- An *entity set* is a set of entities of the same type.
  - Example: set of all persons, departments, holidays, trees.
- An *entity type* describes the entity set.
  - person is an entity type with attributes:
    - name, ss#, phone#, address
  - department is an entity type with attributes:
    - dept-name, building

Entities Types in the Video Store Database

- Customer

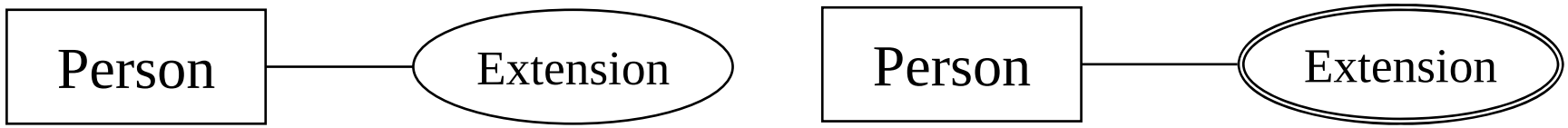
- Film

- Video tape
  - For each film,
    the store maintains a number of tapes

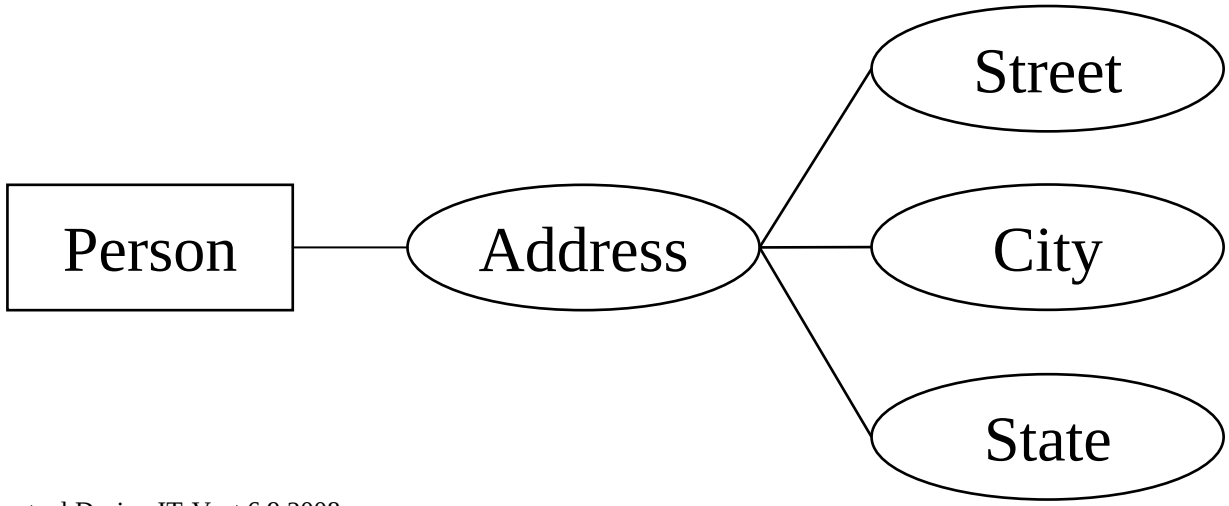
- Entities are represented in rectangles
- How to find entities when modelling?
  - Look for *nouns* in the database requirements.

Attributes

- Attributes are represented in ellipses
- Single valued versus multi-valued
  - E.g., single versus multiple telephone extensions

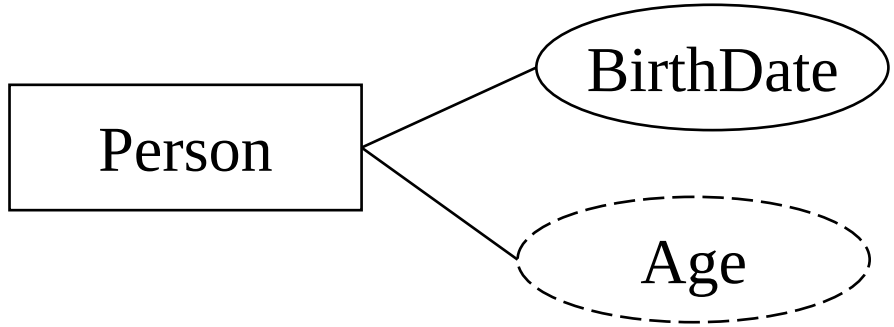


- Simple attributes versus structured attributes
  - E.g., an address is composed of a street, a city and a state

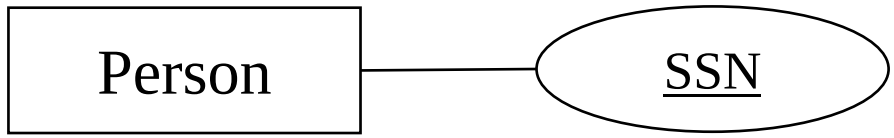


Attributes, cont.

- Stored versus derived
  - E.g., an age can be derived from a stored birth-date



- Null versus non-null
- Keys
  - Uniquely identifies an entity within an entity set



- Domain constrained
  - E.g., a social security number must be exactly 9 decimal digits

Attributes in the Video Store Database

- Customers can have names, addresses, customer IDs (domain constrained to six digits).
- Films can have titles, directors (multi-valued), distributors, and kinds (e.g., foreign film, music video).
- Video tapes can be numbered, and can have a status to indicate whether it is checked out or not.

Relationships

- A *relationship* is an object that associates several entities.
  - Example:

Customer Entity	CheckedOut Relationship	VideoTape *Entity*
000001	CH01	3

- A *relationship set* is a set of objects that each relates $n \geq 2$ entities, each taken from an entity set.

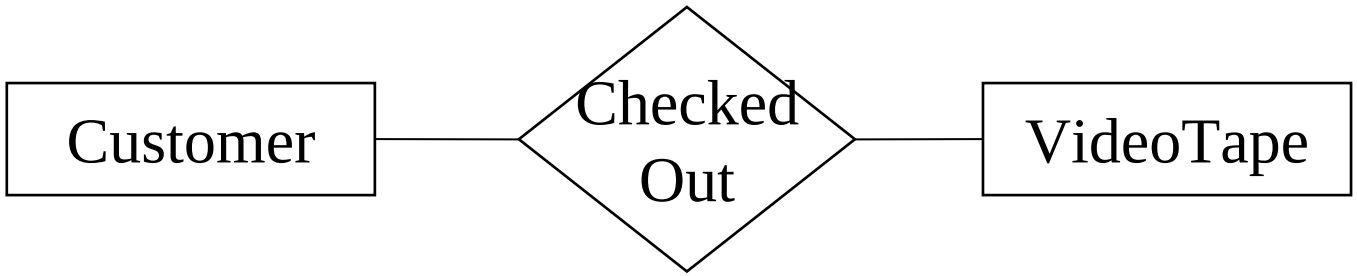
$$\{(e_1, e_2, ..., e_n) \mid e_1 \in E_1 \wedge e_2 \in E_2 \wedge .... \wedge e_n \in E_n\}$$

where the relationship is $(e_1, e_2, ..., e_n)$.

  - Example: CH01 = (000001, 3) $\in$ CheckedOut
- A *relationship type* describes a relationship set.

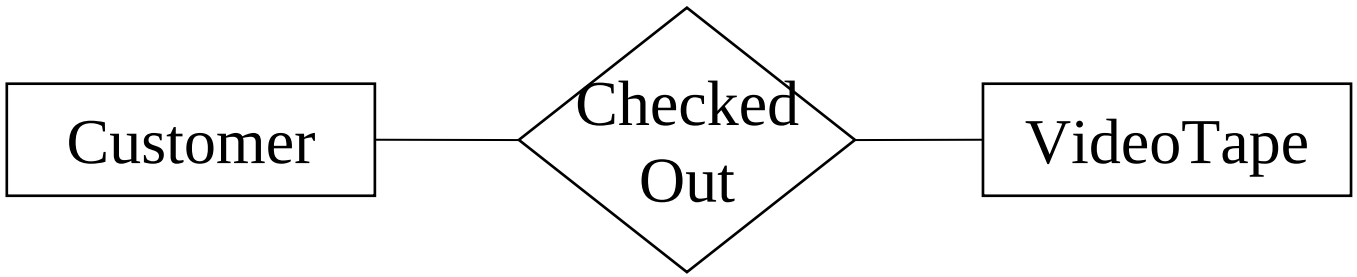
Relationships (cont.)

- We say each entity set E_i *participates* in this relationship set
  $\{(e_1, e_2, ..., e_n) \mid e_1 \in E_1 \wedge e_2 \in E_2 \wedge .... \wedge e_n \in E_n\}$
  - *n*-ary relationship set in general
  - Binary relationship set if n=2
- A relationship can also have *attribute values*.
- A relationship set is represented by a diamond



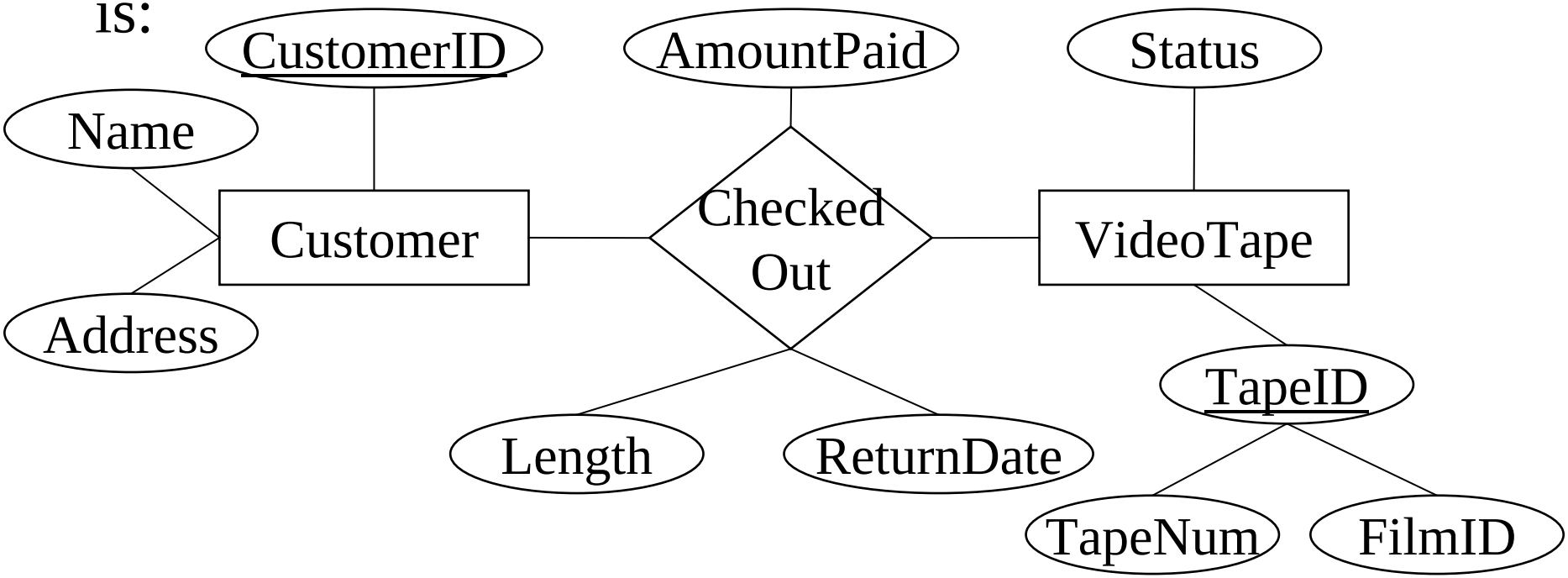
Video Store Database Example

- A VideoTape is an entity type with attributes
  - TapeNum, FilmID (the key attributes), and Status
- A Customer is an entity type with attributes
  - CustomerID (the key), Name, Address
- CheckedOut is a relationship type with attributes
  - ReturnDate, Length, AmountPaid

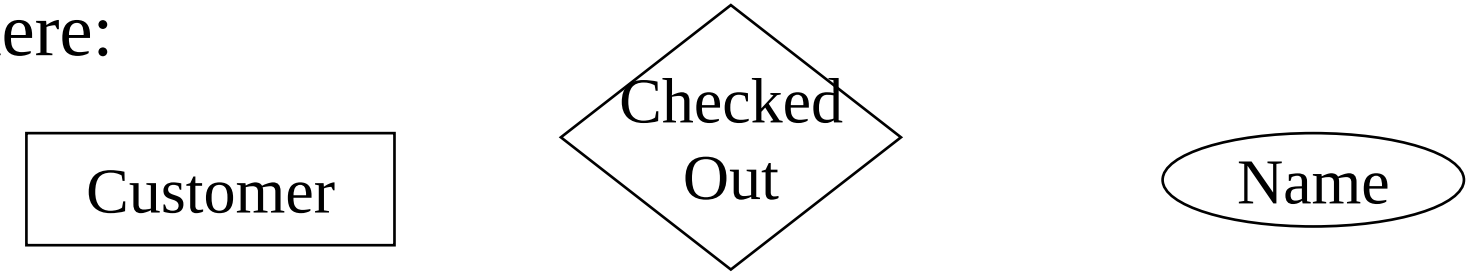


Example, cont.

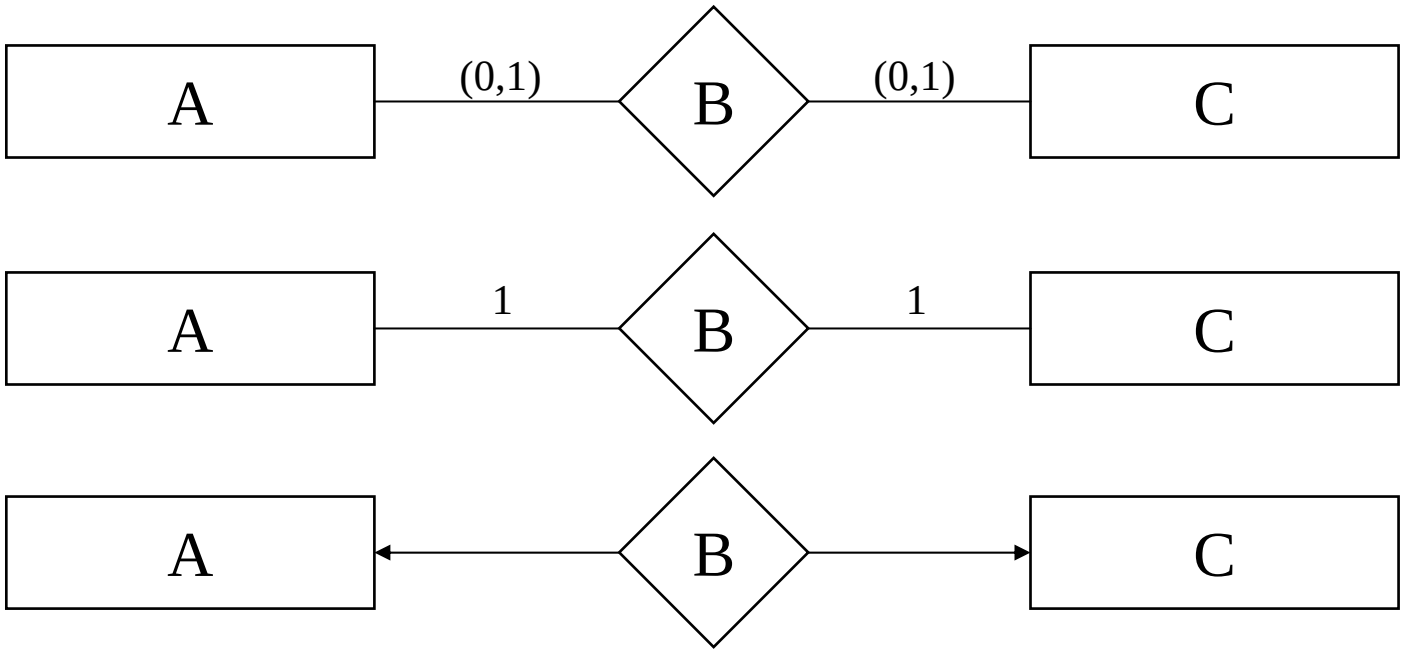
- The entity-relationship diagram for the above example is:



- Where:



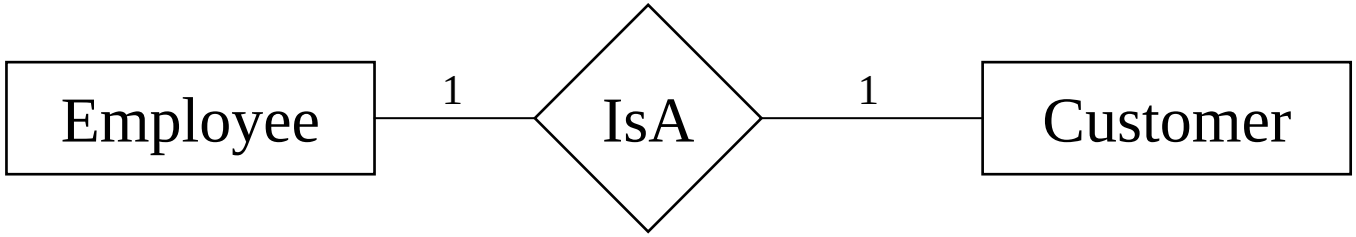
One-To-One Relationship



- An element in A is associated with at most one element in C via the relationship B.
- An element in C is associated with at most one element in A via B.

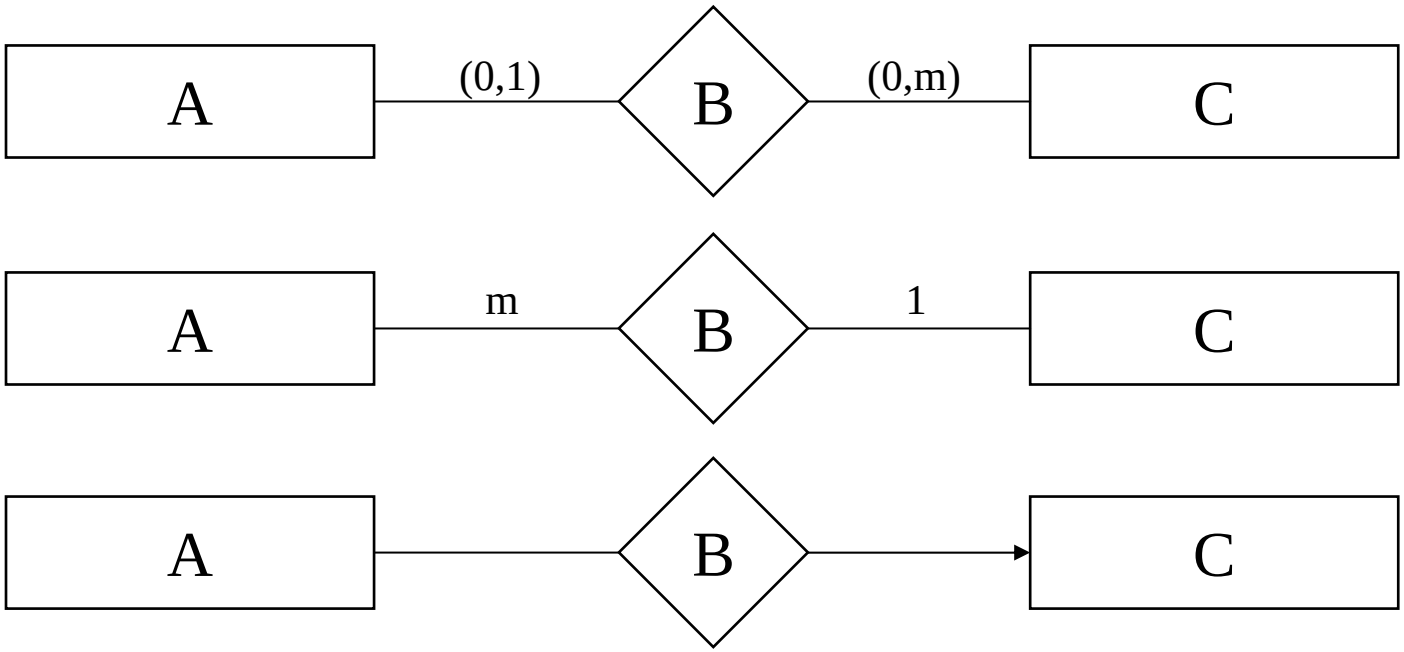
Video Store Database Example

- Say that there was an Employee entity type, and employees could also check out video tapes.
- This could be effected with a 1-to-1 relationship type.



- Each Customer entity can be matched with at most one Employee entity, and each Employee entity can be matched with at most one Customer entity (both entities represent the same person).
- Often this can be better represented using additional attributes, or by using subclasses.

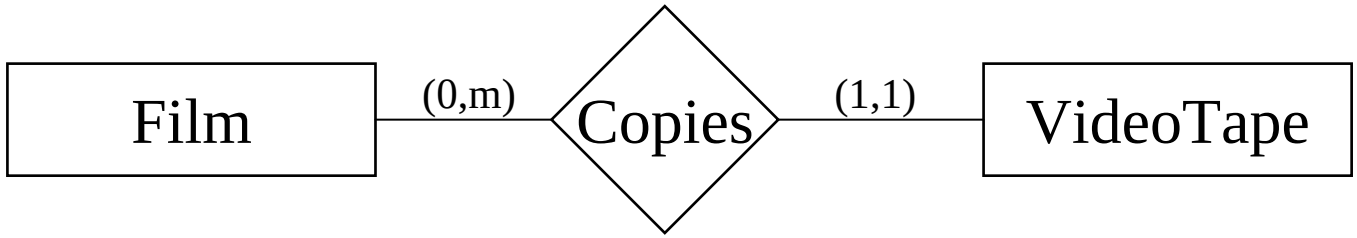
Many-To-One Relationship



- An element in C is associated with several (including 0) elements in A via B.
- An element in A is associated with at most one element in C via B.

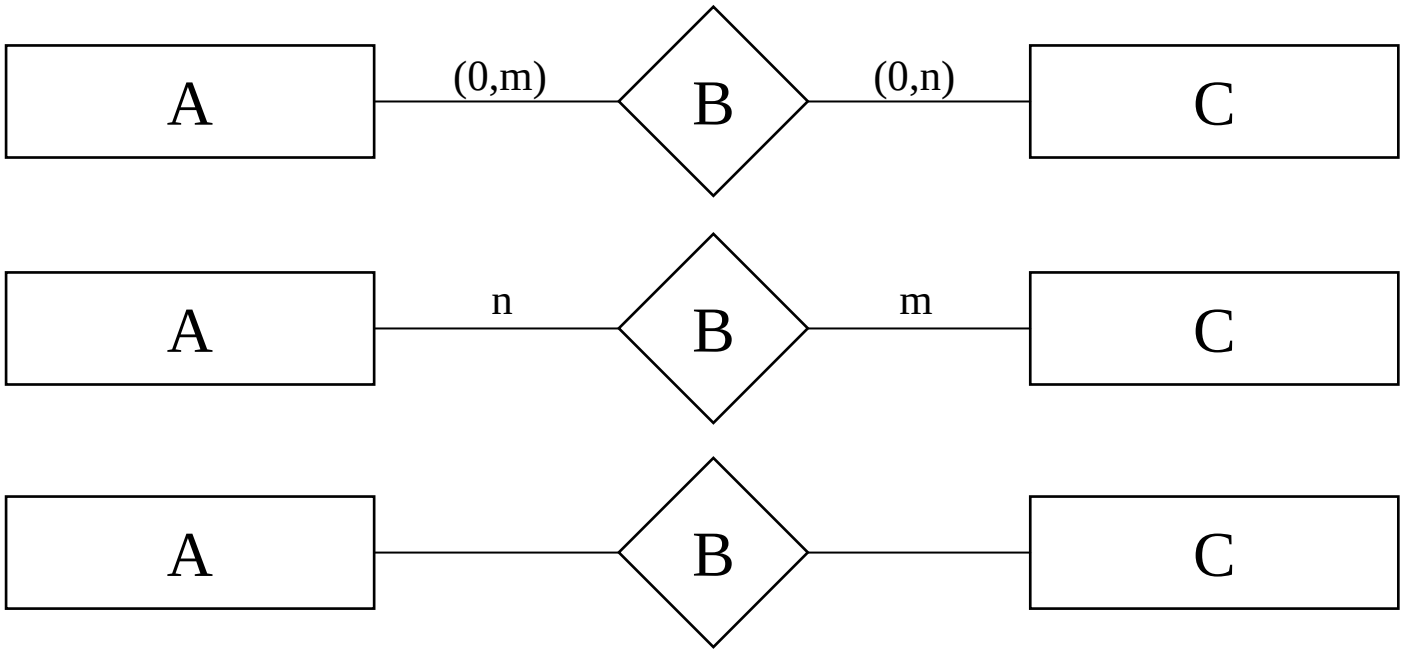
Video Store Database Example

- Each Film entity can be associated with possibly many VideoTape entities.
- This can be captured by a many-to-one relationship type.



- Some Film entities may be associated with no VideoTape entities.
- Each VideoTape entity is associated with exactly one Film entity.

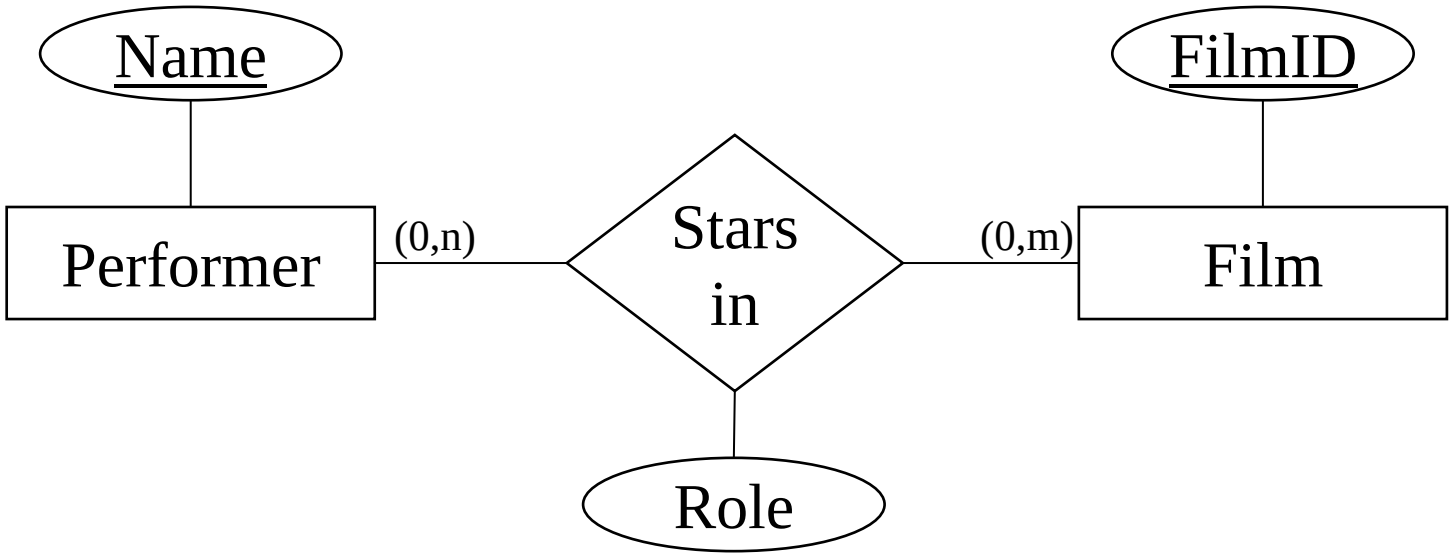
Many-To-Many Relationship



- An element in A is associated with several elements in C via B.
- An element in C is associated with several elements in A via B.

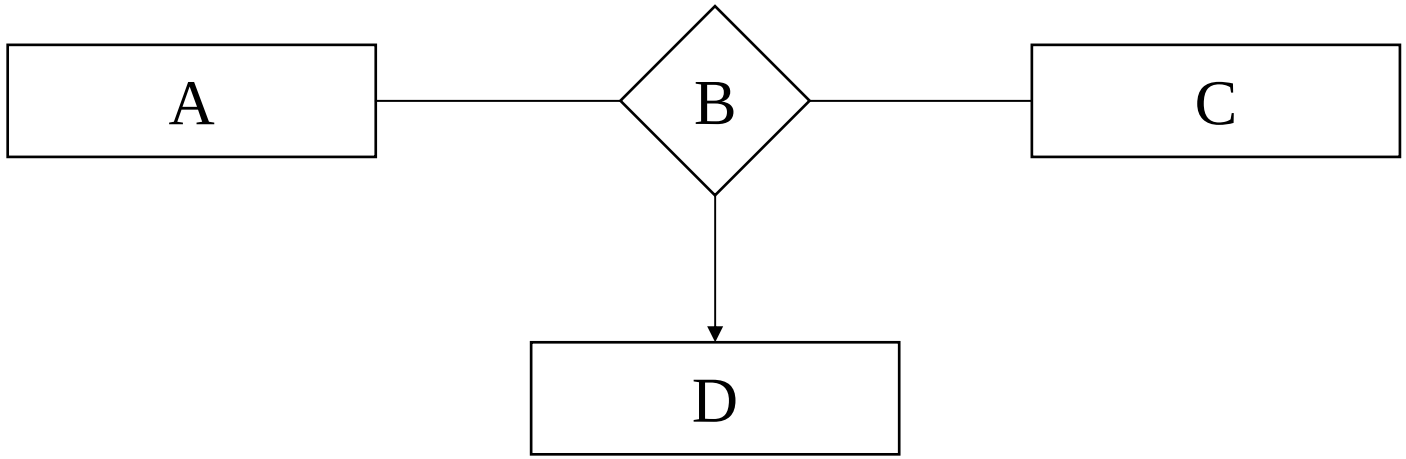
Video Store Database Example

- Each Film entity can be associated with possibly many Performer entities.
- Each Performer entity can be assoicated with possibly many Film entities.



Ternary Relationship

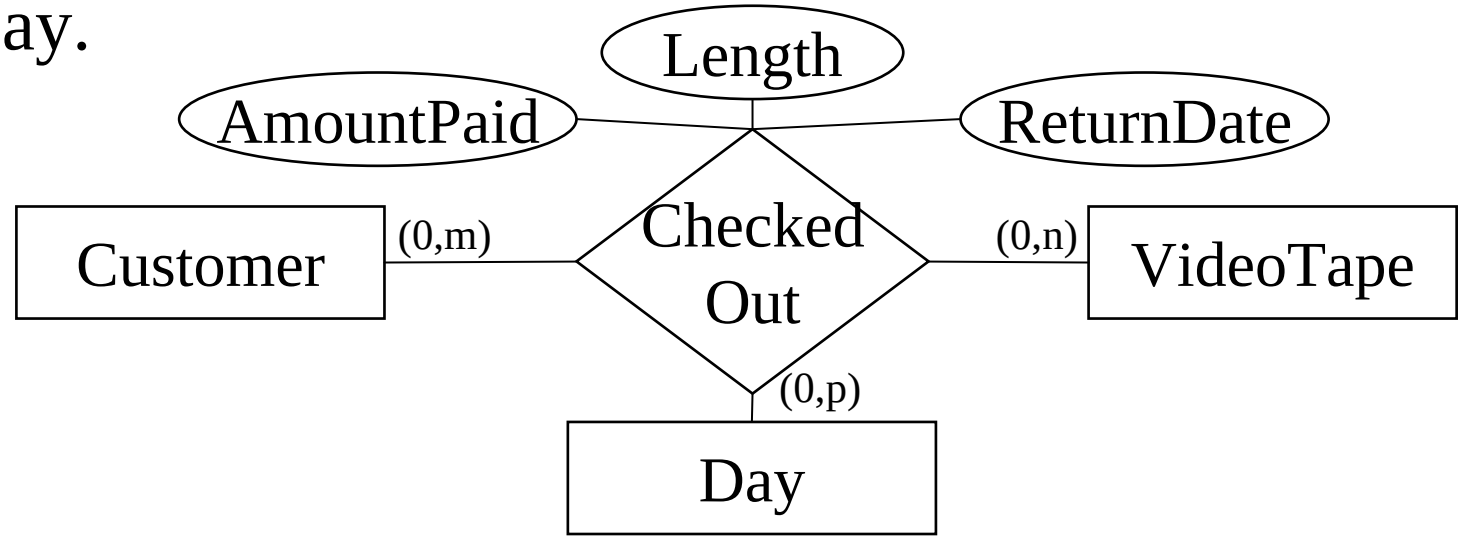
- A *ternary relationship* relates three entity types through an associative entity type that usually describes some sort of action.



- In this example, a pair of elements in A,C is associated with at most one element in D via B.
- The two pairs (a,c), (c,d) => (a,d), a in A, c in C, and d in D.

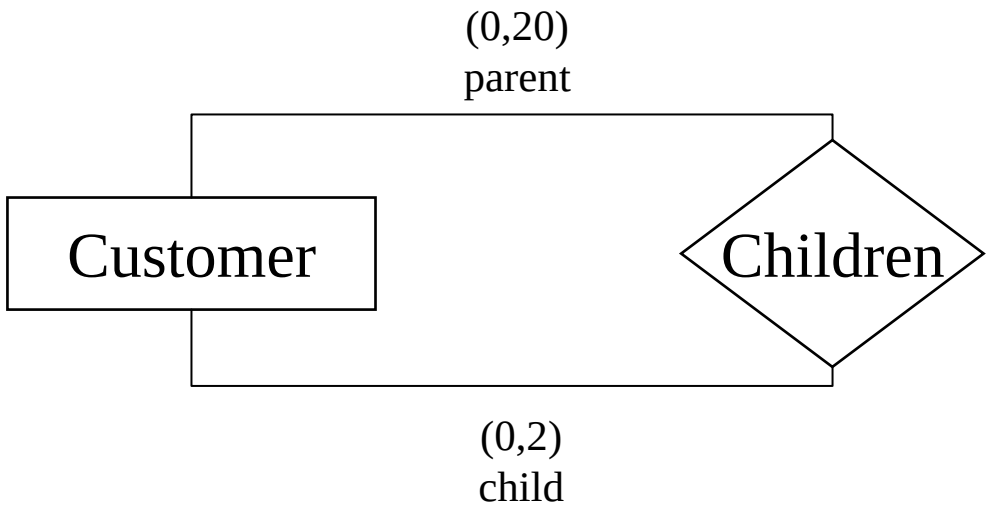
Video Store Database Example

- A Customer can *check out* a VideoTape on a particular Day.



- Each Customer entity can have an arbitrary number of VideoTapes checked out on a particular day.
- Customers can check out VideoTapes on an arbitrary number of days.
- A VideoTape may be checked out by an arbitrary number of Customers.

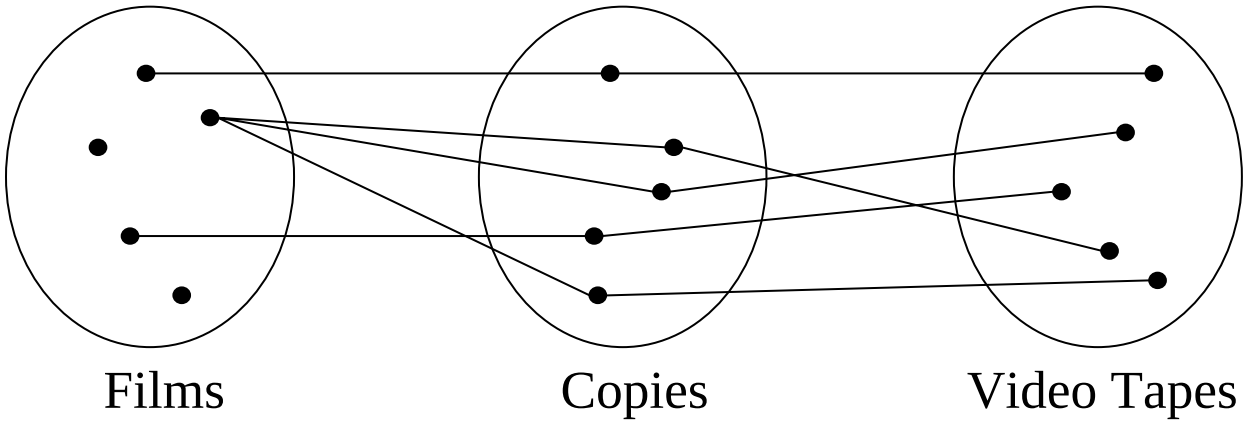
Roles



- Entity sets of a relationship need not be distinct.
  - Relationship on the same entity set.
- Note: labels "parent" and "child" are called *roles*.
- Role labels are optional, and are used to clarify the semantics of the relationship.

Participation

- *Total participation* (indicated by double line): every entity in the entity set participates in at least one relationship in the relationship set
  - E.g., participation of VideoTape in Copies is total
- *Partial participation*: some entities may not participate in any relationship in the relationship set
  - E.g., participation of Film in Copies is partial



Keys

- A *super key* of an entity type is a set of one or more attributes whose values uniquely determine one entity.
- A *candidate key* of an entity type is a minimal super key.
  - CustomerID is a candidate key of Customer.
- Generally, there are may be several candidate keys. One of the keys is selected to be the *primary key*.
  - CustomerID is also the primary key of Customer.
  - (FilmID, TapeNum) is the primary key of VideoTape.
- A *key* of a relationship type is the combination of primary keys of the entity types that are part of the relationship.
  - (FilmID, TapeNum, CustomerID, CheckDate) is the primary key of CheckedOut.

Primary Keys

- Primary keys allow entity sets and relationship sets to be expressed uniformly as tables which represent the contents of the database.

Customer Table:

CustomerID	CustomerName	Address: Street	Address: City	Address: State
000001	John	Elm	Tucson	AZ
000004	Mary	Stone	Tucson	AZ

VideoTape Table:

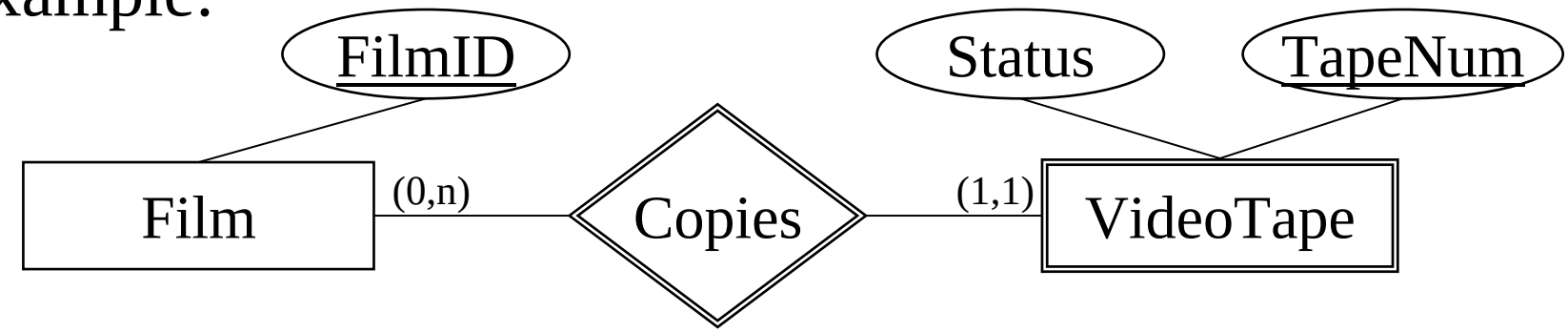
TapeNum	FilmID	Status
1	000001	Incirculation
2	000001	Damaged
1	000002	Incirculation

CheckedOut Table:

CustomerID	FilmID	TapeNum	Date	ReturnDate	AmountPaid	Length
000001	000001	3	2005-08-28	2005-08-29	3.25	2 days
000004	000002	1	2005-08-28	2005-08-30	3.25	3 days

Weak Entity Types

- Not every entity type has a primary key. Such an entity type is referred to as a *weak* entity type.
- An entity type with attributes <TapeNum, Status> does not have a key, since, e.g., <1, incirculation> does not uniquely identify a particular video tape.
- Example:

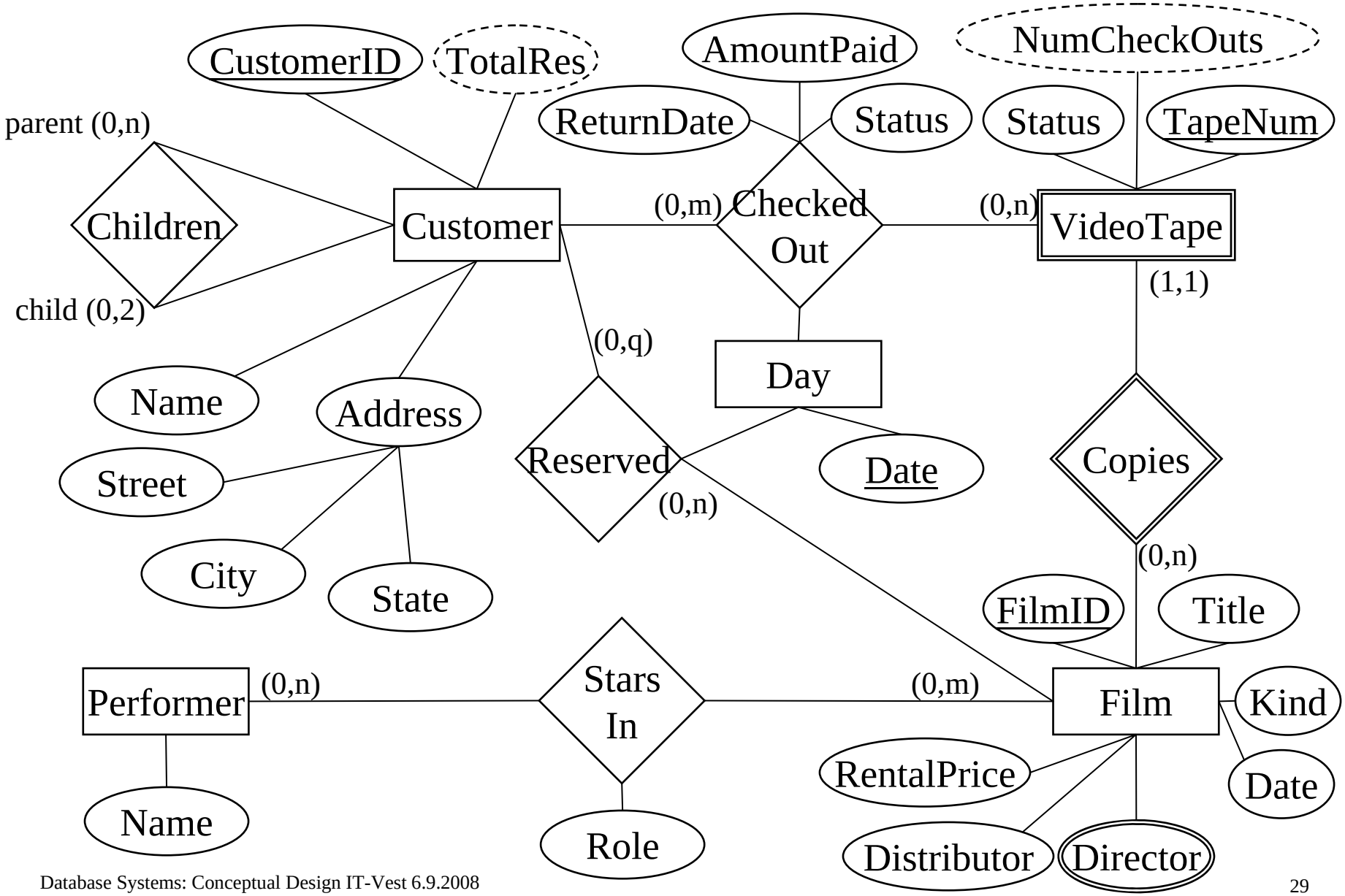


  - Double box represents weak entity type.
  - The existence of a VideoTape entity depends on the existence of a Film entity.

Weak Entity Types, cont.

- Semantics:
  - Deletion of a Film entity requires deletion of that film's video tape entities.
- A weak entity can be related to *precisely one* strong entity in an entity type, via a 1-1 or 1-n relationship.
- It is of course possible to introduce more attributes to the video tape entity type, so that a primary key will exist, but they may not be needed for database processing.

A Video Store ER Schema



Design Notes

- Entities are nouns, relationships are verbs.
- Each statement in the requirement specification should be located somewhere in the ER schema.
- Each ER schema construct should be located somewhere in the requirement specification.
- Conceptual design often reveals inconsistencies and ambiguities in the requirement specification, which must be first resolved.

Review of ER-Model

- Basic ER model has entities, relationships, and attributes.
- Extended ER model adds subclasses and superclasses. Also a concept called categories, not dicussed here .
- The ER is used extensively.
- The notation used various ER modeling tools differs.

Conceptual Design: Outline

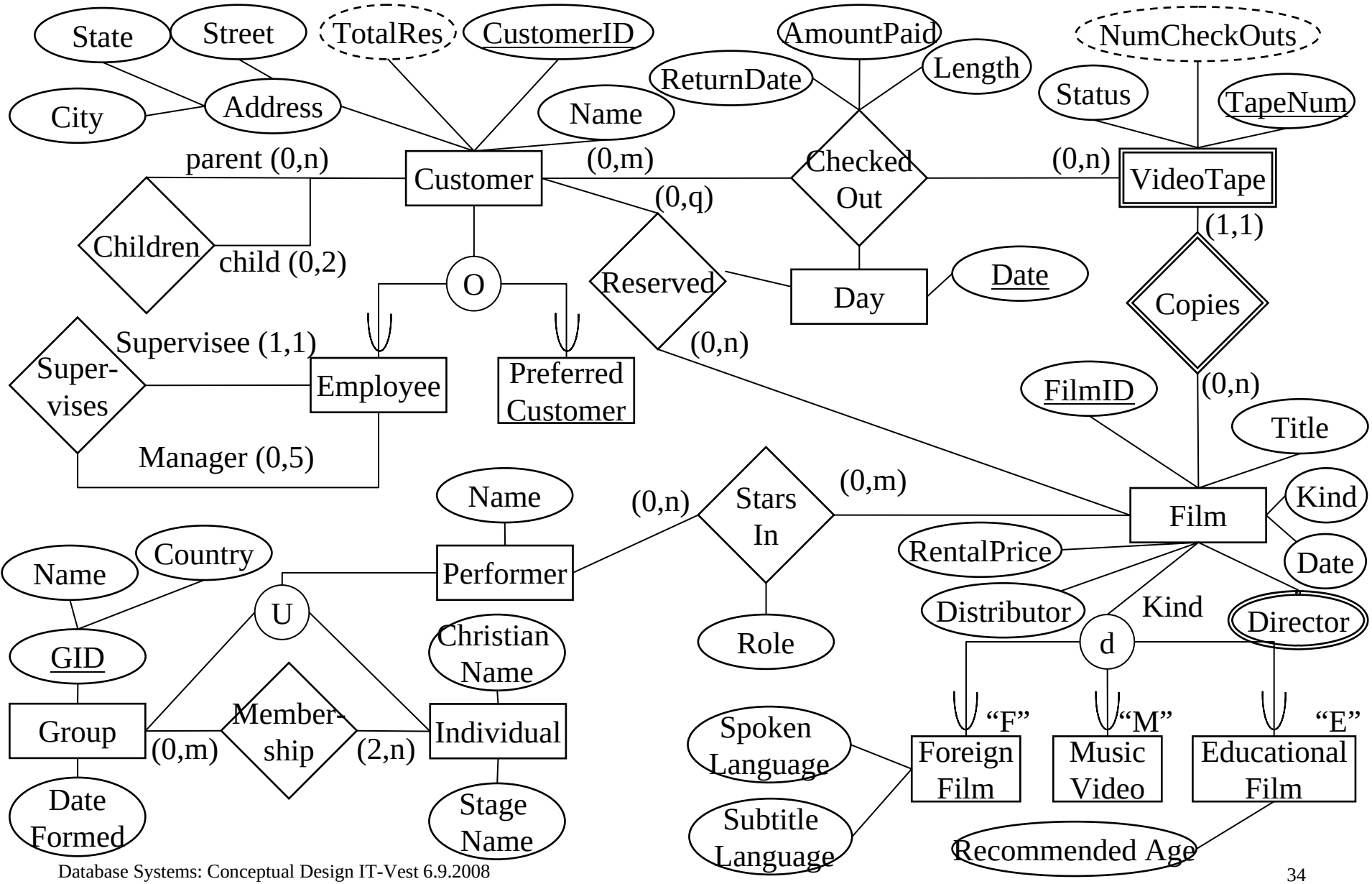
- The Data Modelling Process
- The Entity-Relationship Model
- Mapping Entity-Relationship Model to Tables
  - 7 out of a total 9 steps

Mapping an ER Schema to Tables

- In a sequence of steps, a set of tables is created.
- Sometimes automated in CASE tools

1. Regular entity types
2. Weak entity types
3. Binary 1:1 relationship types
4. Binary 1:N relationship types
5. Binary M:N relationship types
6. *n*-ary relationship types
7. Multi-valued attributes
8. Superclass/subclass relationship types (not considered)
9. Categories (not considered)

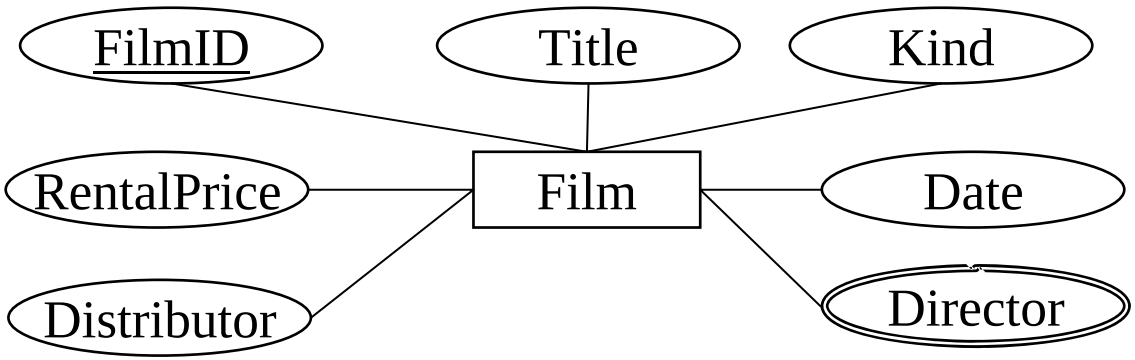
The Video Store ER Schema



ER To Tables: Step 1

- Create a table for each regular entity type.
- The table has one column for each simple attribute of its corresponding entity type.
- The primary key for the table is the primary key of the entity type.
- If there are no attributes other than the primary key, and if the entity participates totally in a relationship, then the table can be eliminated.

ER To Tables: Step 1, Example



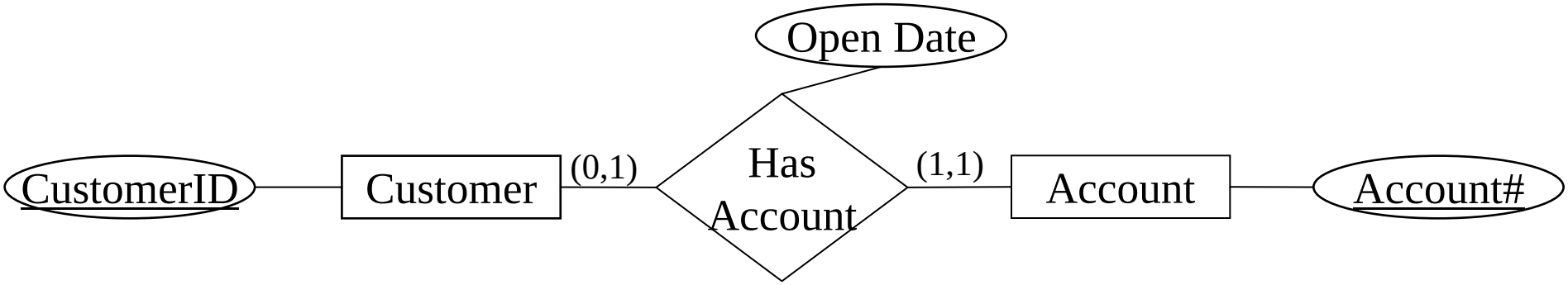
- Film (FilmID, Title, PubDate, RentalPrice, Distributor, Kind)
- Customer (CustomerID, Name, Street, City, State)
- Day (Date) (in this case, this table can be eliminated, even though it doesn't participate in a total relationship, because we don't care about dates not participating in a relationship)

ER To Tables: Step 3

- For each 1:1 binary relationship type, extend a table.
- Extend the table that corresponds to one of the participating entity types with the primary key of the other participating entity type. This is the foreign key.
- It is best to extend a table of an entity type with total participation.
- Add also columns for each of the simple attributes of the relationship type.

ER To Tables: Step 3, Example

- No 1:1 relationships are present in the example, so let's assume that there is also an Account entity type, and a HasAccount relationship type.



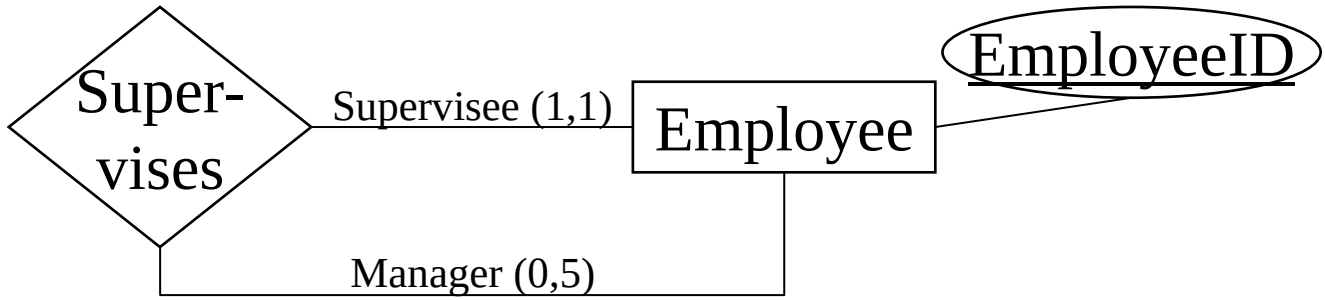
- For this ER schema, there would already by a Customer table and an Account table from step 1.
- Extend the Account table to include the key of Customer, which is CustomerID.
- Account(Account#, CustomerID, OpenDate)

ER To Tables: Step 4

- For each regular 1:N binary relationship type, there are several approaches.
  - Option 1: If the relationship is total, then extend a table.
  - Option 2: If the relationship is not total, extend the table with nullable attributes (sometimes not allowed for foreign keys).
  - Option 3: Create a separate table for the relationship.

ER To Tables: Step 4, cont.

- Option 3: Create a separate table for the relationship.
- Primary key is simply the primary key of the "many" side.
- Add columns for each of the simple attributes of the relationship type.

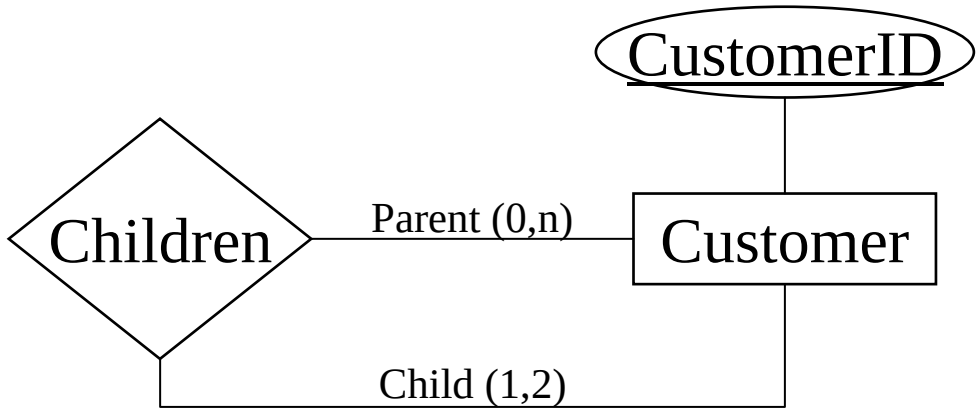


- Supervises(Manager, Supervisee)

ER To Tables: Step 5

- Create a table for each binary M:N relationship type.
- A table has as columns the primary keys of the participating entity types, as its primary key.
- These are also foreign keys.
- Include also columns for each of the simple attributes of the relationship type.
- The primary key is the union of the primary keys of the participating entity types.

ER To Tables: Step 5, Example

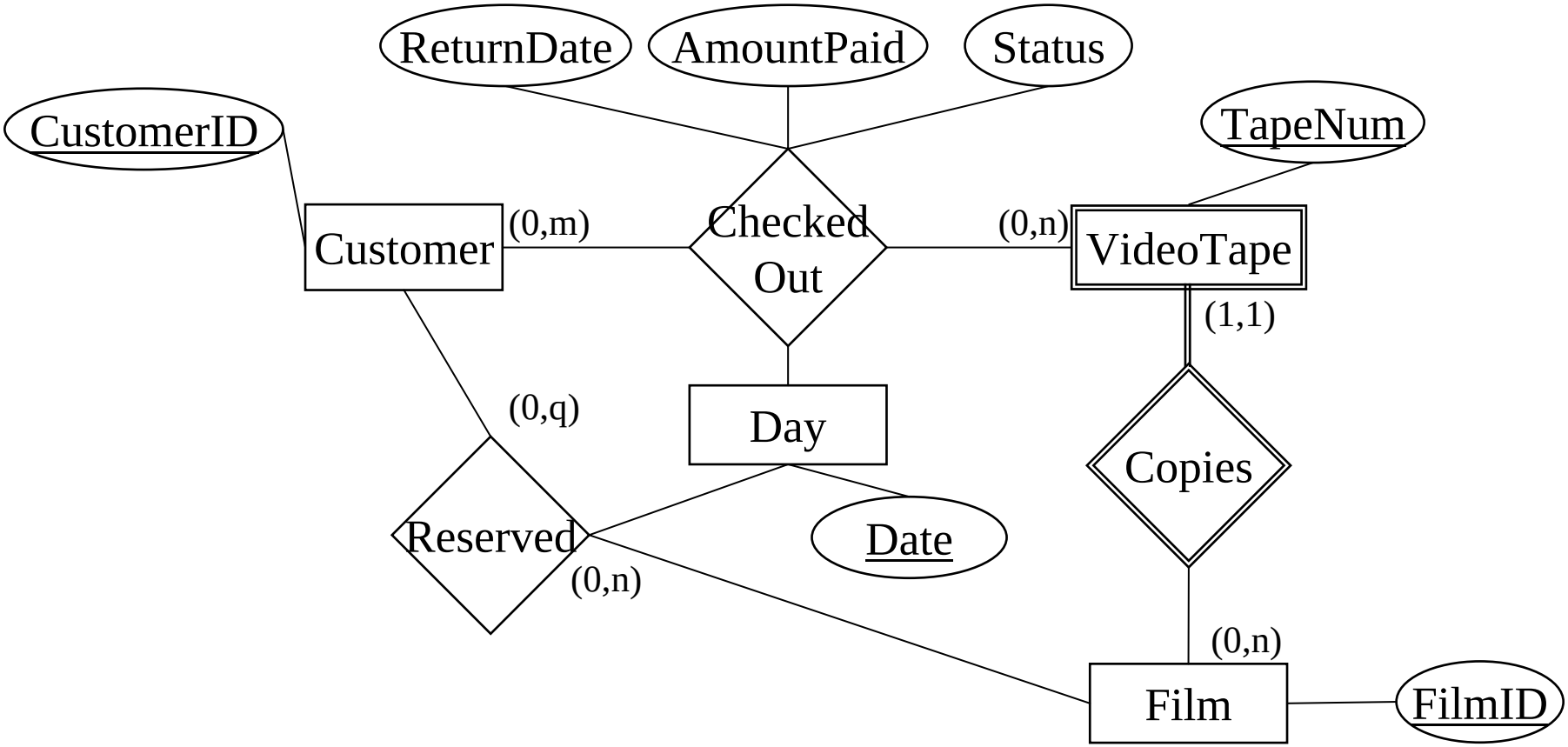


- Children (Parent, Child)
- Membership (GroupName, Country, StageName)
- StarsIn (PerformerID, FilmID, Role)

ER To Tables: Step 6

- Create a table for each *n*-ary ($n > 2$) relationship type.
- The table has as columns the primary keys of the participating entity types.
- These are also the (*n*) foreign keys.
- Include also columns for each of the simple attributes of the relationship type.
- The primary key is the union of the primary keys of the participating entity types.

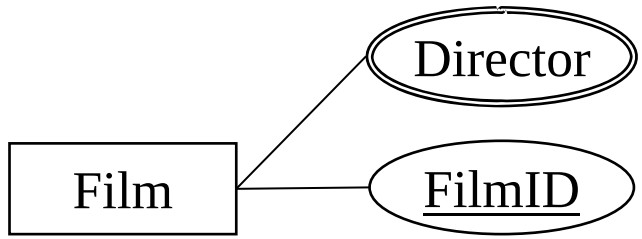
ER To Tables: Step 6, Example



- Reserved (CustomerID, FilmID, ResDate)
- CheckedOut (CustomerID, FilmID, TapeNum, CheckDate, ReturnDate, AmountPaid, Status)

ER To Tables: Step 7

- Create a table for each multivalued attribute.
- The table has a column for each simple attribute of the multivalued attribute.
- Include also a column for the primary key of the entity or relationship type that the attribute belongs to. This is the foreign key.
- The primary key is the combination of all the attributes.
- Example:



  - Director (<u>FilmID</u>, <u>Name</u>)

General Discussion

- Note that derived attributes and composite attributes are not explicitly included.
- Different options may be used for a multilevel specialization hierarchy.
- Superclasses that share a subclass have the same key, and all options are applicable.
- The resulting tables do not differentiate entity and relationship types.

Resulting Video Store Relational Schema

- Entities
  - Customer (CustomerID, Name, Street, City, State)
  - Film (FilmID, Title, PubDate, RentalPrice, Distributor, Kind, RecommendedAge, SpokenLanguage, SubtitleLanguage)
- Relationships
  - VideoTape (FilmID, TapeNum, Status)
  - Children (Parent, Child)
  - Membership (GroupName, Country, StageName)
  - StarsIn (PerformerID, FilmID, Role)
  - Reserved (CustomerID, FilmID, ResDate)
  - CheckedOut (CustomerID, FilmID, TapeNum, CheckDate, ReturnDate, AmountPaid, Length)

Video Store Relational Schema, cont.

- Multi-valued Attributes
  - Director (FilmID, Name)
- Subclasses
  - Employee (EmployeeID, CustomerID, Name, Street, City, State, Manager)
  - PreferrredCustomer (CustomerID, Name, Street, City, State, DiscountLevel)
- Categories
  - Group (Name, Country, DateFormed, PerformerID)
  - Individual (Name, ChristianName, PerformerID)

Review of Steps

- Step 1: Regular entity type
  - Create a table.
- Step 2: Weak entity type
  - Create a table.
- Step 3: 1:1 binary relationship type
  - Extend a table with foreign key.
- Step 4: Regular 1:N binary relationship type
  - Extend a table with foreign key.
- Step 5: Binary M:N relationship type
  - Create a table.

Review of Steps, cont.

- Step 6: N-ary relationship type
  - Create a table.
- Step 7: Multi-valued attribute
  - Create a table.
- Step 8: Superclass/subclass relationship
  - Four options; some extend a table, other create new tables.
- Step 9: Category/subcategory
  - Create tables.