Advanced ER Modelling

Topics List

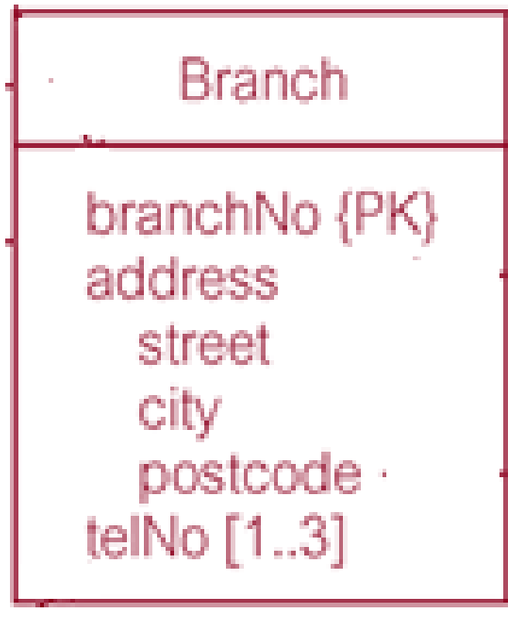
- Multivalued Attributes
- Recursive Relationships
- Weak Entity Types

Multivalued Attributes

Modelling

- Recall that a multivalued attribute holds multiple values for each occurrence of an entity type.

- To model a multivalued attribute you write the attribute followed by square brackets [ ] and inside the square brackets you write down the min and max values.



Multivalued Attributes
Modelling

- In general, composite and multi-valued attributes may be nested arbitrarily to any number of levels, although this is rare.
- For example, PreviousDegrees of a STUDENT is a composite multi-valued attribute denoted by [PreviousDegrees (College, Year, Degree, Field)].
- Multiple PreviousDegrees values can exist.
- Each has four subcomponent attributes: College, Year, Degree, Field.

Multivalued Attributes

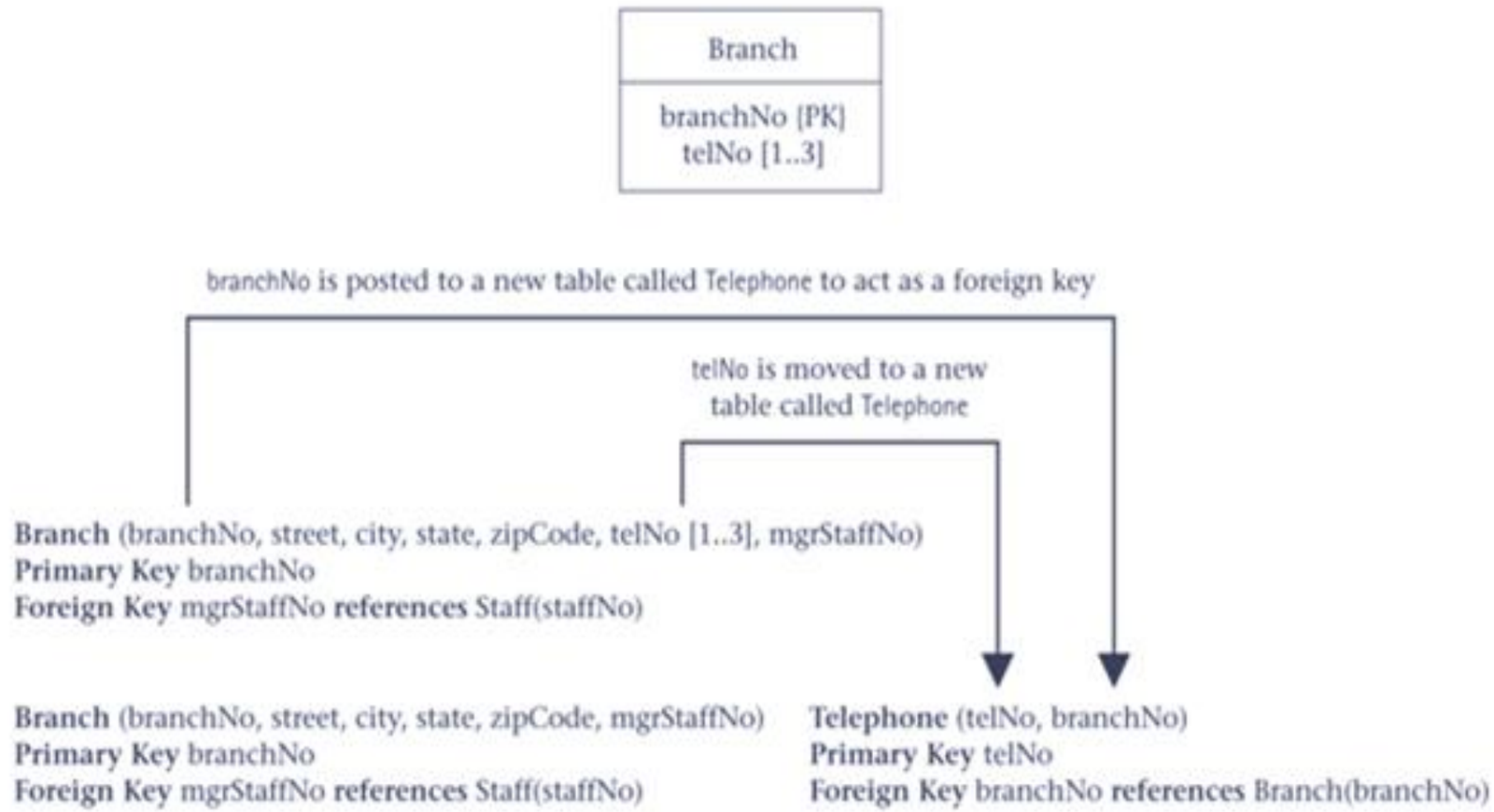
Mapping

- ***Create a new relation to represent the multi-valued attribute(s) and include the primary key of original entity in the new relation, to act as a foreign key.***

- Unless (one of) the multi-valued attribute(s) are an alternate key of the entity, the primary key of the new relation is the combination of (one of) the multi-valued attribute(s) and the primary key of the entity. Remove the multivalued attribute(s) from the original relation.

Multivalued Attributes

Mapping

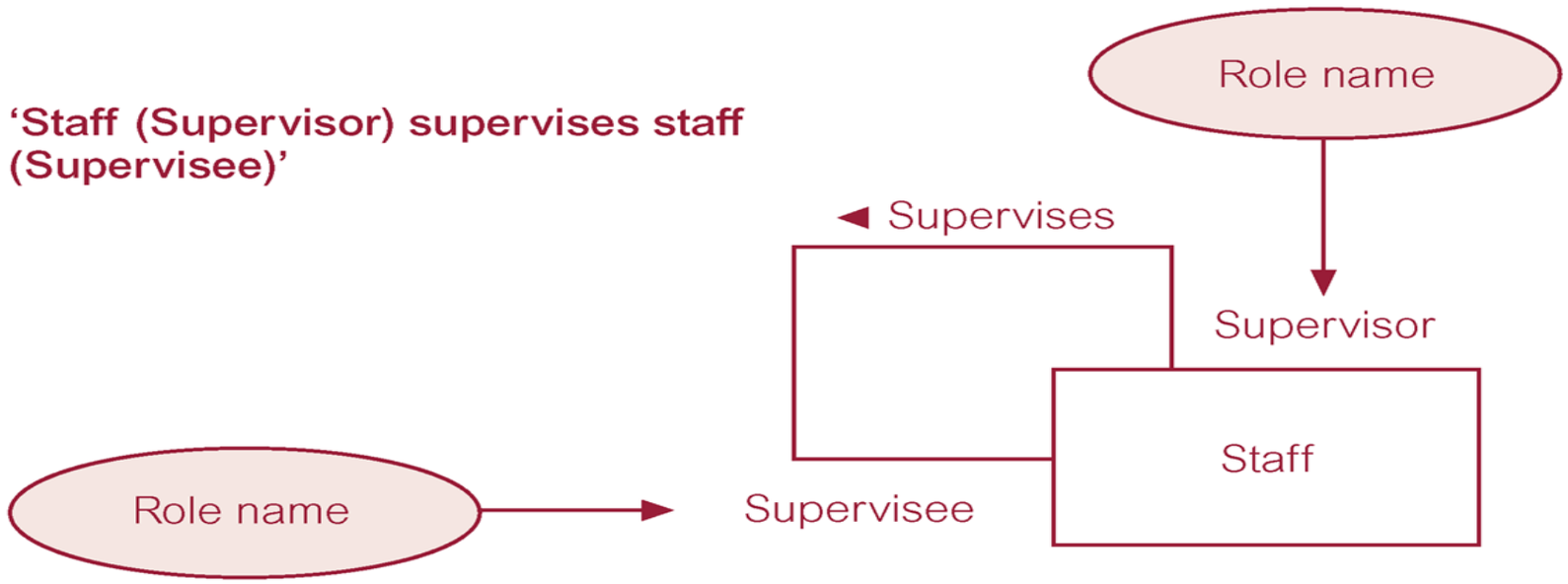


Topics List

- Multivalued Attributes
- Recursive Relationships
- Weak Entity Types

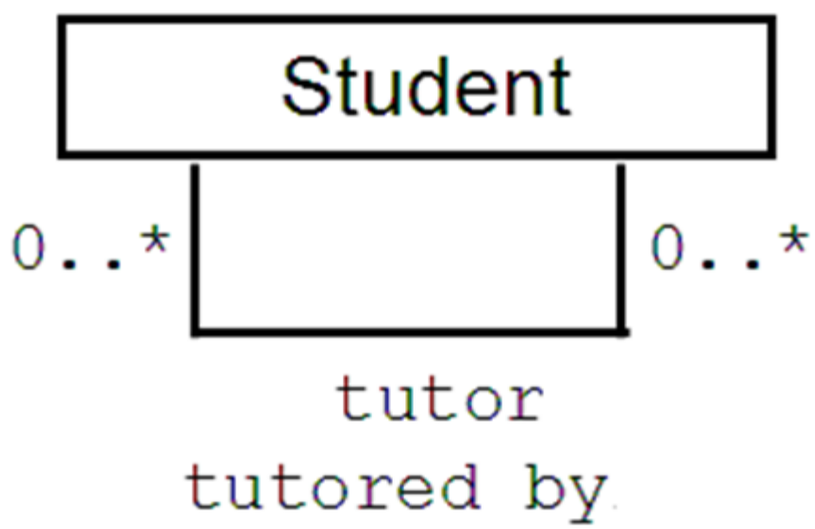
Recursive Relationships

- Recall that a Recursive Relationship type is where the *same* entity type participates more than once in *different* roles. Sometimes called *unary* relationships.



Recursive Relationships
Modelling

- As you can see from the previous Figure, to model a recursive relationship you draw a line to represent the relationship from one edge of the entity type to another edge of the same entity type.



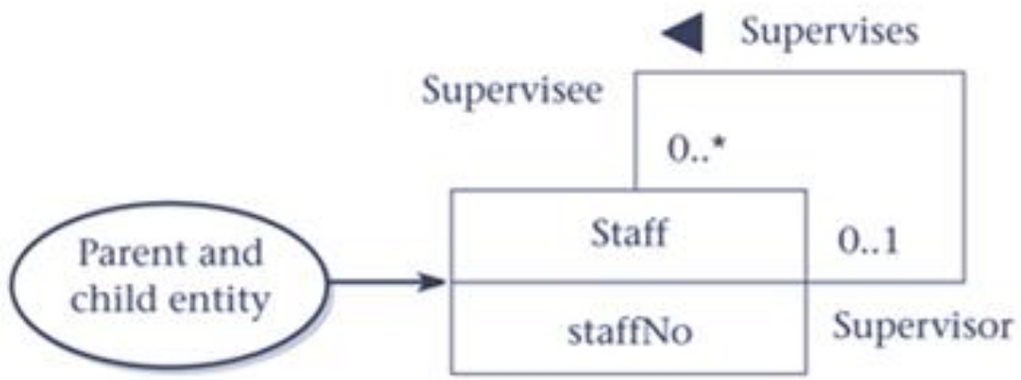
Recursive Relationships

Mapping

- ***1:* recursive relationships***
  - The representation of a 1:* recursive relationship is similar to 1:* binary relationship. However, in this case, both the parent and child entity is the *same* entity.
  - ***For a 1:* recursive relationship, post a copy of the primary key into the same entity (itself) to act as a foreign key. This new attribute is renamed to represent the relationship.***

Recursive Relationships
Mapping

- ***1:* recursive relationships***



staffNo is posted to represent the *Supervises* relationship and renamed supervisorStaffNo

Staff (staffNo, name, position, salary, branchNo, supervisorStaffNo)
Primary Key staffNo
Foreign Key branchNo **references** Branch(branchNo)
Foreign Key supervisorStaffNo **references** Staff(staffNo)

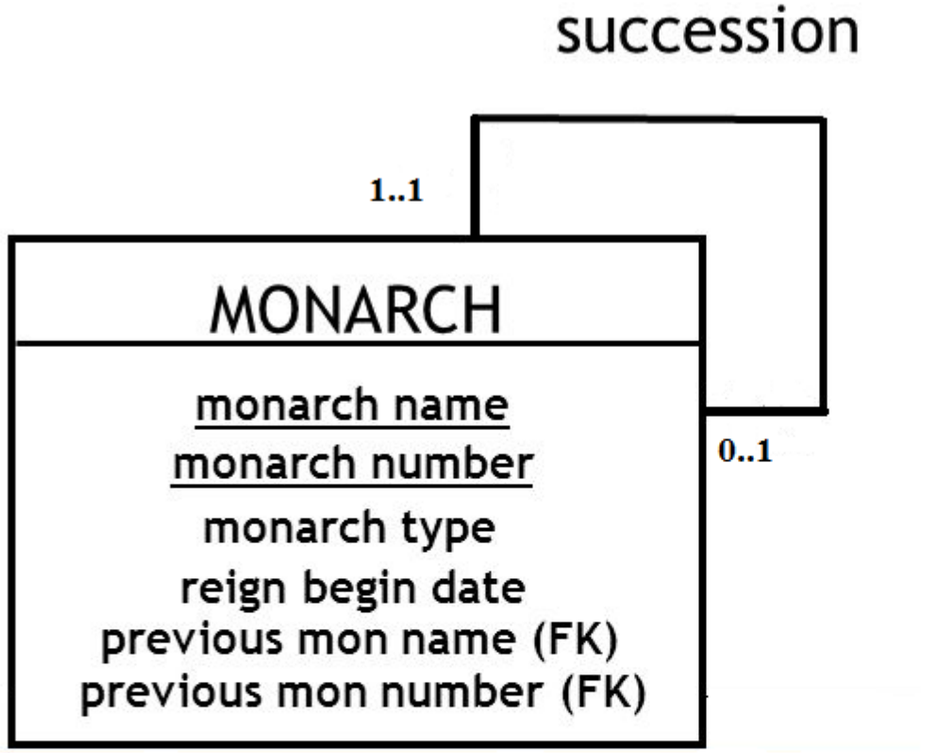
Recursive Relationships

Mapping

- ***1:1 recursive relationships***
  - ***For a 1:1 recursive relationship, post a copy of the primary key into the same entity (itself) to act as a foreign key. This new attribute is renamed to represent the relationship.***

Recursive Relationships
Mapping

- ***1:1 recursive relationships***



Recursive Relationships

Mapping

- ***1:1 recursive relationships***

monarch (monname, monnum, montype, rgnbeg,
premonname, premonnum)
Primary Key monname, monnum,
Foreign Key premonname, premonnum references monarch(monname, monnum)

Recursive Relationships

Mapping

- ***1:1 recursive relationships***

  In the example below, we can see who was each monarch's predecessor. For example, George VI is the predecessor to Elizabeth II. For each monarch we record their name and their number.

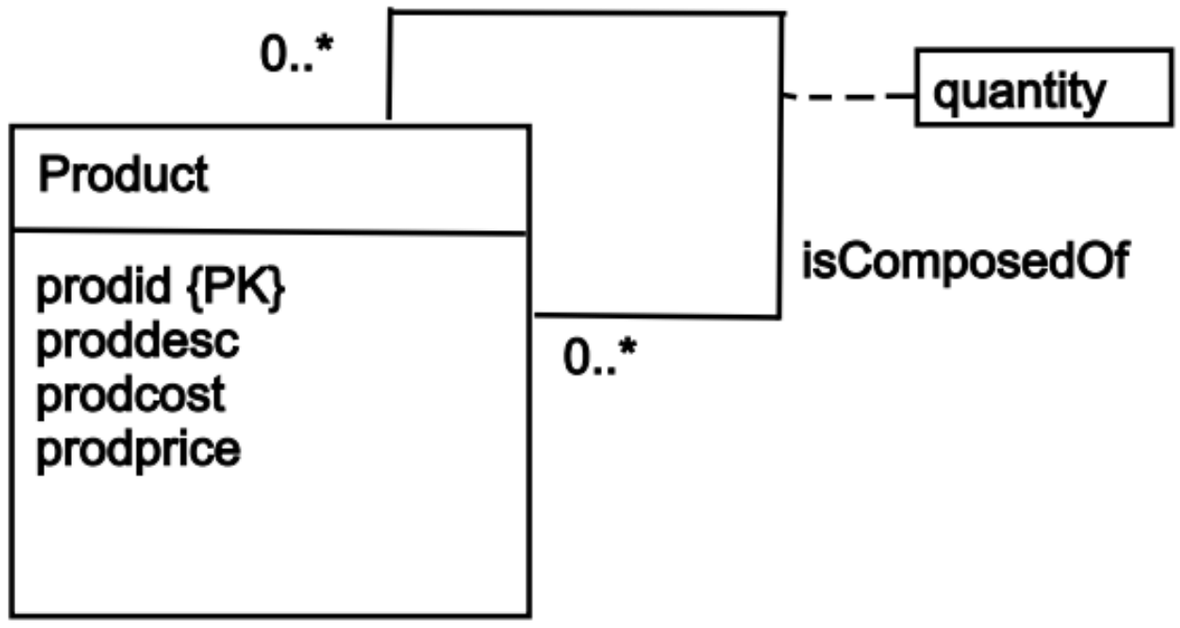
montype	monname	monnum	rgnbeg	*premonname*	*premonnum*
Queen	Victoria	I	1837/6/20	William	IV
King	Edward	VII	1901/1/22	Victoria	I
King	George	V	1910/5/6	Edward	VII
King	Edward	VIII	1936/1/20	George	V
King	George	VI	1936/12/11	Edward	VIII
Queen	Elizabeth	II	1952/2/6	George	VI

Recursive Relationships
Mapping

- ***:* recursive relationships***
  - The representation of a *:* recursive relationship is similar to *:* binary relationship.
  - ***For a *:* recursive relationship, we will create a new relation which will hold two copies of the original primary key. Again one of the primary key copies will be renamed to represent the relationship.***

Recursive Relationships
Mapping

- ***:* recursive relationships***



Recursive Relationships
Mapping

- ***:* recursive relationships***

  product(prodid, proddesc, prodcost, prodprice)
  Primary key prodid

  assembly (prodid , subprodid, quantity)
  Primary key prodid, subprodid,
  Foreign key prodid references product(prodid),
  Foreign key subprodid references product(prodid)

Recursive Relationships
Mapping

- ***:* recursive relationships***

prodid	proddesc	prodcost	prodprice
1000	Animal photography kit		725
101	Camera	150	300
102	Camera case	10	15
103	70-210 zoom lens	125	200
104	28-85 zoom lens	115	185
105	Photographer's vest	25	40
106	Lens cleaning cloth	1	1.25
107	Tripod	35	45
108	16 GB SDHC memory card	30	30

quantity	*prodid*	*subprodid*
1	1000	101
1	1000	102
1	1000	103
1	1000	104
1	1000	105
2	1000	106
1	1000	107
4	1000	108

Topics List

- Multivalued Attributes
- Recursive Relationships
- Weak Entity Types

Weak Entity Types

- All of the entity types that we have met so far are *Strong Entity Types*, meaning they do not depend on another entity type for its existence.

- ***Weak Entity Type***: An entity type that will depend on another entity type for its existence.

- Each entity occurrence cannot be uniquely identified using only the attributes associated with that entity type. A weak entity type does not exist on its own but must participate in a relationship with another (strong) entity type.

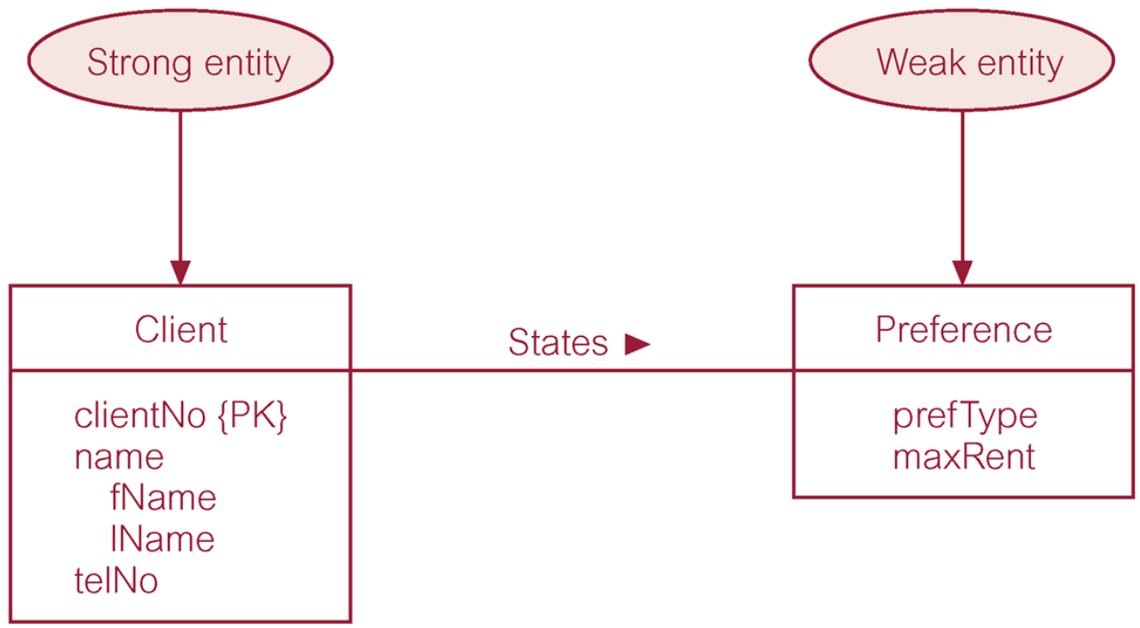
Weak Entity Types

- For instance, in a library system, a weak entity type would be *edition* as related to a *book* entity type; each *book* has several editions, and certainly we cannot speak about an edition if this does not happen in the context of a specific book. The *book* entity type is called the **owner** entity type or **identifying** entity type for the weak entity type *edition*.

- Another example would be *dependent* as related to an *Employee* entity type. An instance of a dependent would depend entirely on some instance of an employee or else the dependent would not be kept in the database. . The *employee* entity type is called the **owner/identifying** entity type for the weak entity type *dependent*.

Weak Entity Types
Modelling

- In this example, we have a *Client* entity type and a *Preference* entity type. A *Preference* entity cannot exist in its own right but must be related back to the owner/identifying entity type *(Client*).



- You model a weak entity type like a strong entity type except it has no Primary Key field.

Weak Entity Types

Mapping

- For each weak entity in the data model:
  - Create a relation that includes all the simple attributes of that entity.
  - The primary key of a weak entity is partially or fully derived from each owner entity and so the identification of the primary key of a weak entity cannot be made until after all the relationships with the owner entities have been mapped.

Weak Entity Types

Mapping

- Here we create a relation for FilmCopy and include the simple attributes: copyNo, location, and status. Since the relationship between Film and FilmCopy is one-to-many, we post a copy of the primary key from the Film entity type (FilmId) as a foreign key. This now becomes part of the primary key of the FilmCopy relation (along with one or more other attributes as there may be many film copies).

Weak Entity Types

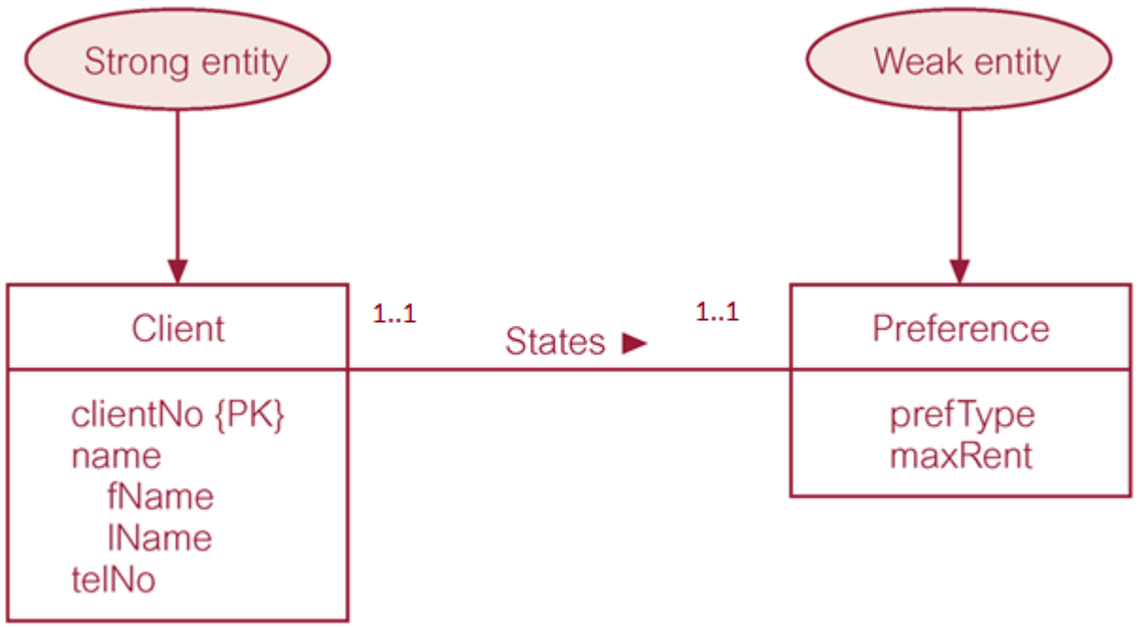
Mapping

Film(filmId, title, producer, director, minutes)
Primary key filmId

FilmCopy(filmId, copyNo, location, status)
Primary key filmId, copyNo
Foreign key filmId references Film(filmId)

Weak Entity Types

Mapping



Client(clientNo, fName, lName, telNo)
Primary key clientNo

Preference(clientNo, prefType, maxRent)
Primary key clientNo
Foreign key clientNo references Client(clientNo)

Weak Entity Types
Mapping

OR

- Since the relationship is one:one (fully mandatory) we could merge the above 2 relations into one as follows:

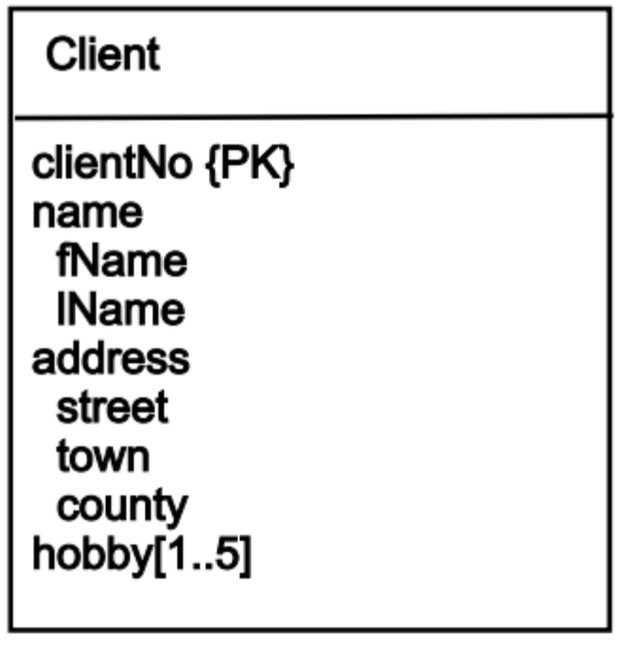
Client(clientNo, fName, lName, telNo, prefType, maxRent)
Primary key clientNo

Exercise

- Using the figures specified below, create a logical data model for the respective relations that represent:
  - Multi-valued attributes
  - A Recursive relationship type
  - A Weak Entity Type

Exercise

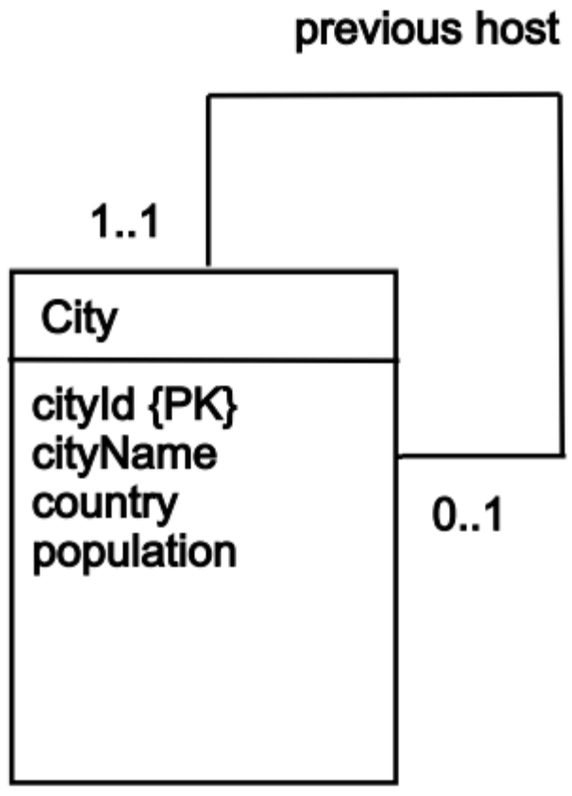
- Multivalued attribute



Exercise

Exercise

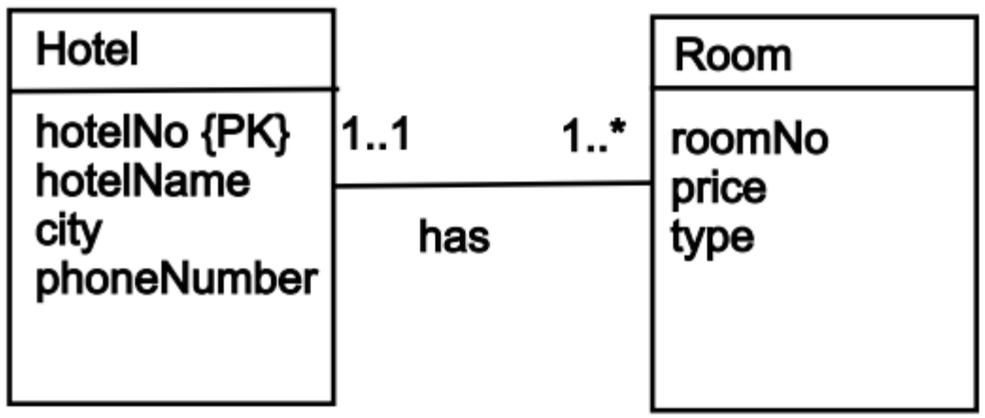
- Recursive relationship



Exercise

Exercise

- Weak Entity Type



Exercise