# Chapter 12

## Entity-Relationship Modeling

# Chapter 12 - Objectives

- How to use Entity–Relationship (ER) modeling in database design.
- Basic concepts associated with ER model.
- Diagrammatic technique for displaying ER model using Unified Modeling Language (UML).
- How to identify and resolve problems with ER models called connection traps.
- How to build an ER model from a requirements specification.

# ER diagram of Branch user views of *DreamHome*

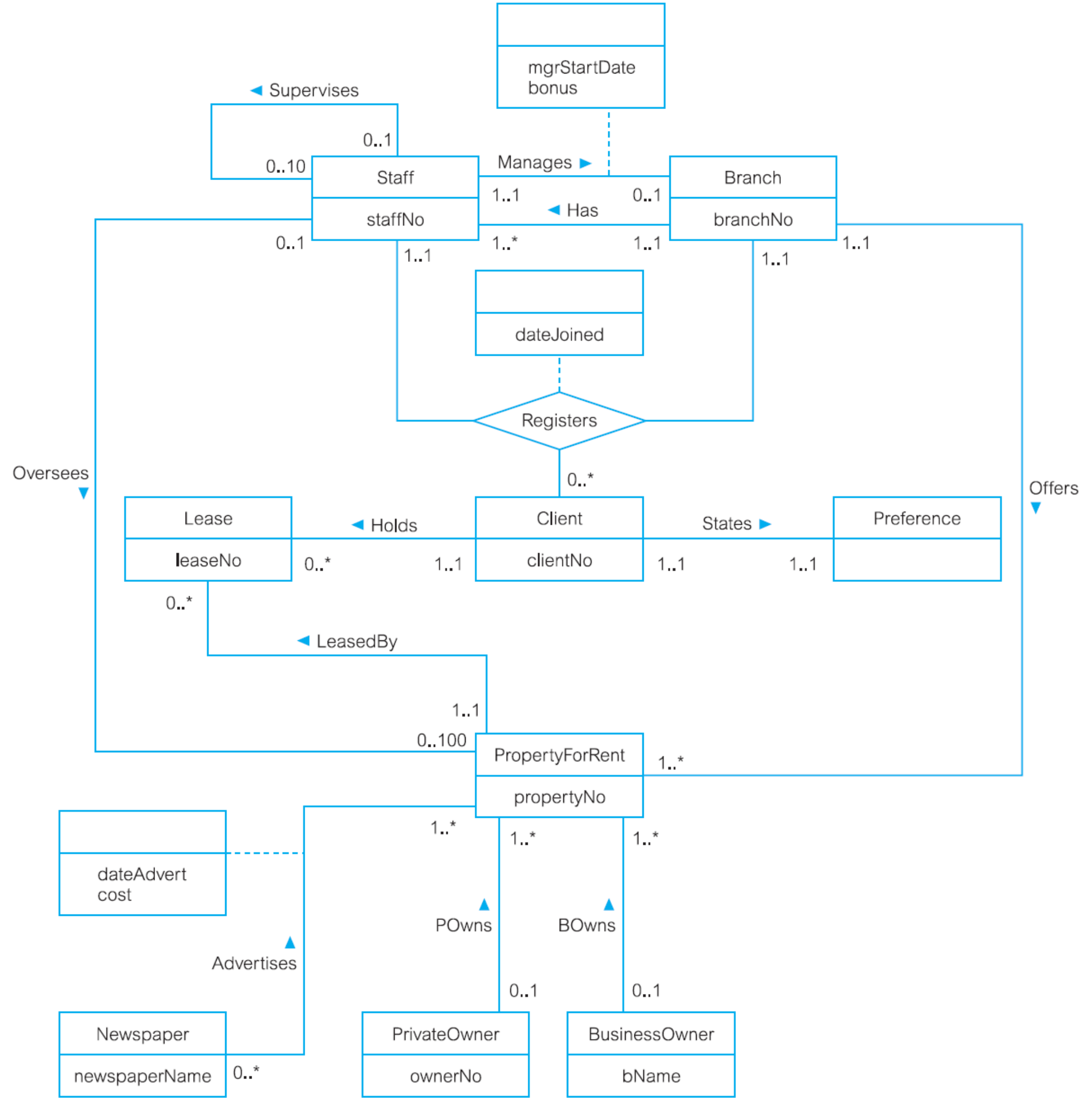


**Figure 12.1** An Entity–Relationship (ER) diagram of the Branch view of *DreamHome*.

# Concepts of the ER Model

- Entity types
- Relationship types
- Attributes

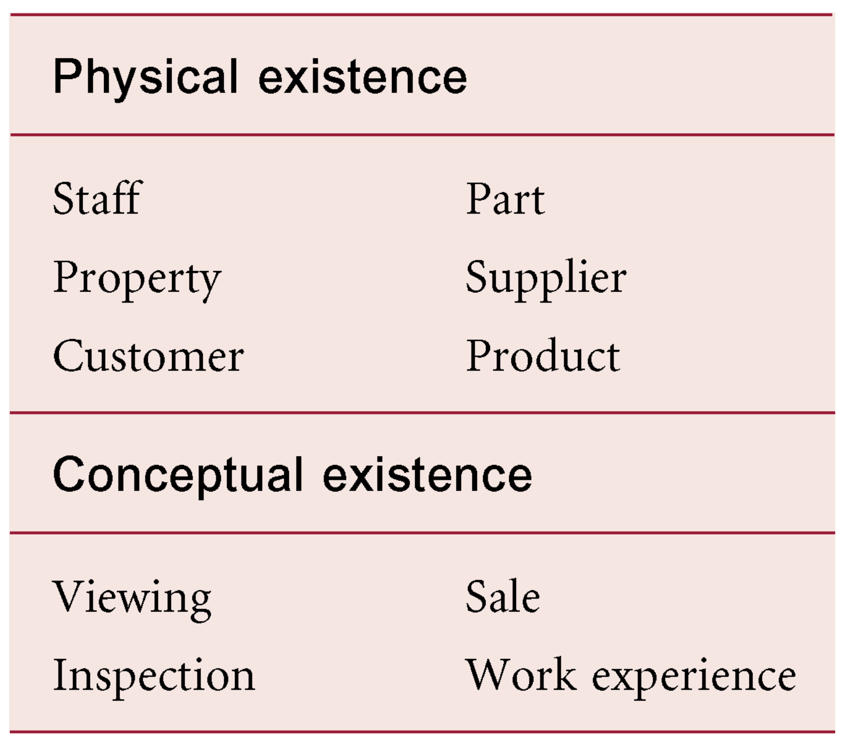
# Entity Type

- Entity type
  - Group of objects with same properties, identified by enterprise as having an independent existence.

- Entity occurrence
  - Uniquely identifiable object of an entity type.

# Examples of Entity Types

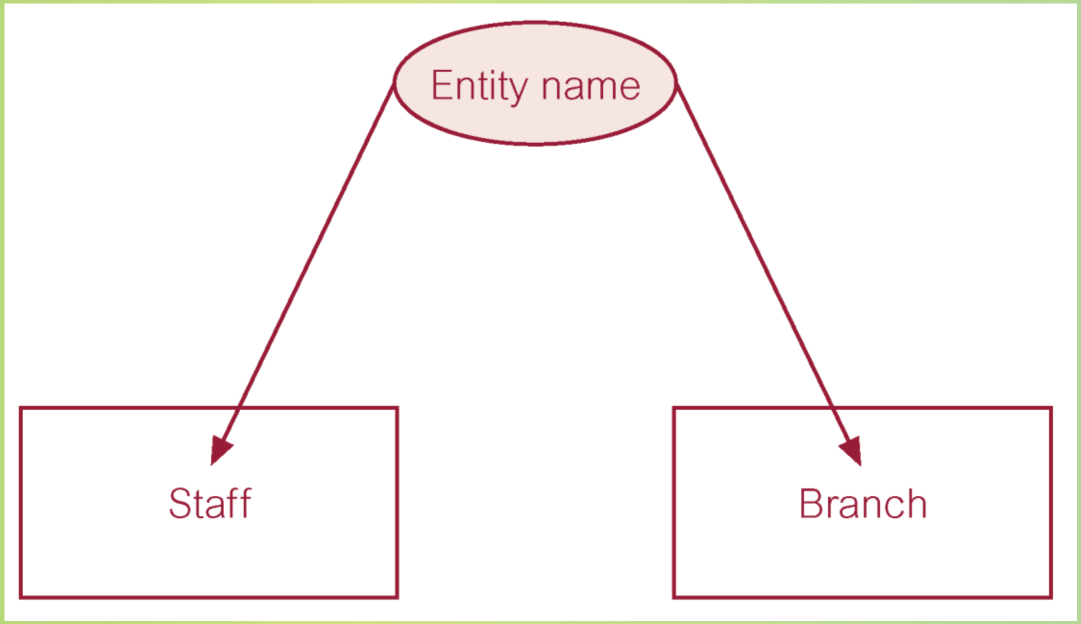
**An entity type has an independent existence and can be objects with a physical (or “real”) existence or objects with a conceptual (or “abstract”) existence.**



| | |
|---|---|
| **Physical existence** | |
| Staff | Part |
| Property | Supplier |
| Customer | Product |
| **Conceptual existence** | |
| Viewing | Sale |
| Inspection | Work experience |

# ER diagram of Staff and Branch entity types

**Each entity type is shown as a rectangle, labeled with the name of the entity, which is normally a singular noun.**



# Relationship Types

- Relationship type
  - Set of meaningful associations among entity types.
  - An example of a relationship type called Powns is shown in Figure 12.1, which associates the *PrivateOwner* and PropertyForRent entities.

- Relationship occurrence
  - Uniquely identifiable association, which includes one occurrence from each participating entity type.
  - An example of a relationship type called Has is shown in Figure 12.1, which represents an association between *Branch* and *Staff* entities, that is Branch Has Staff. Each occurrence of the Has relationship associates one Branch entity occurrence with one Staff entity occurrence.

# Semantic net of *Has* relationship type

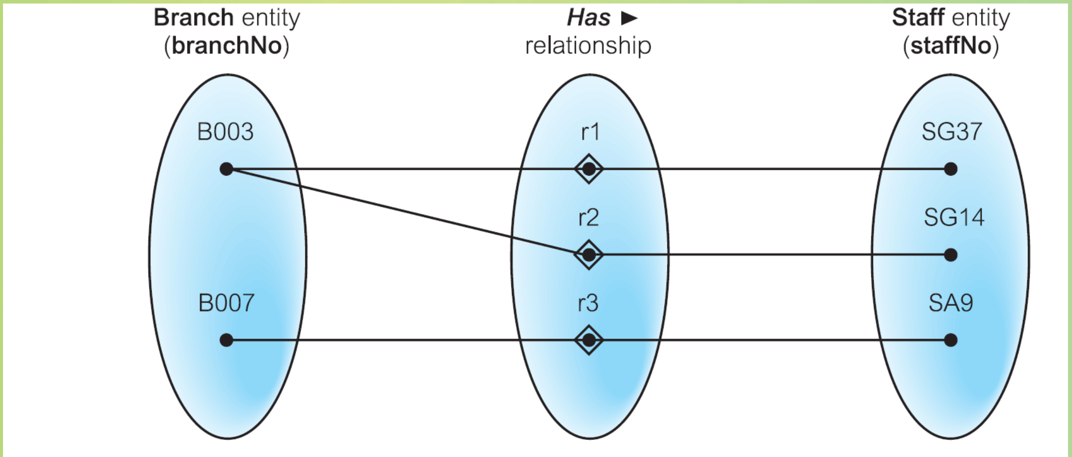
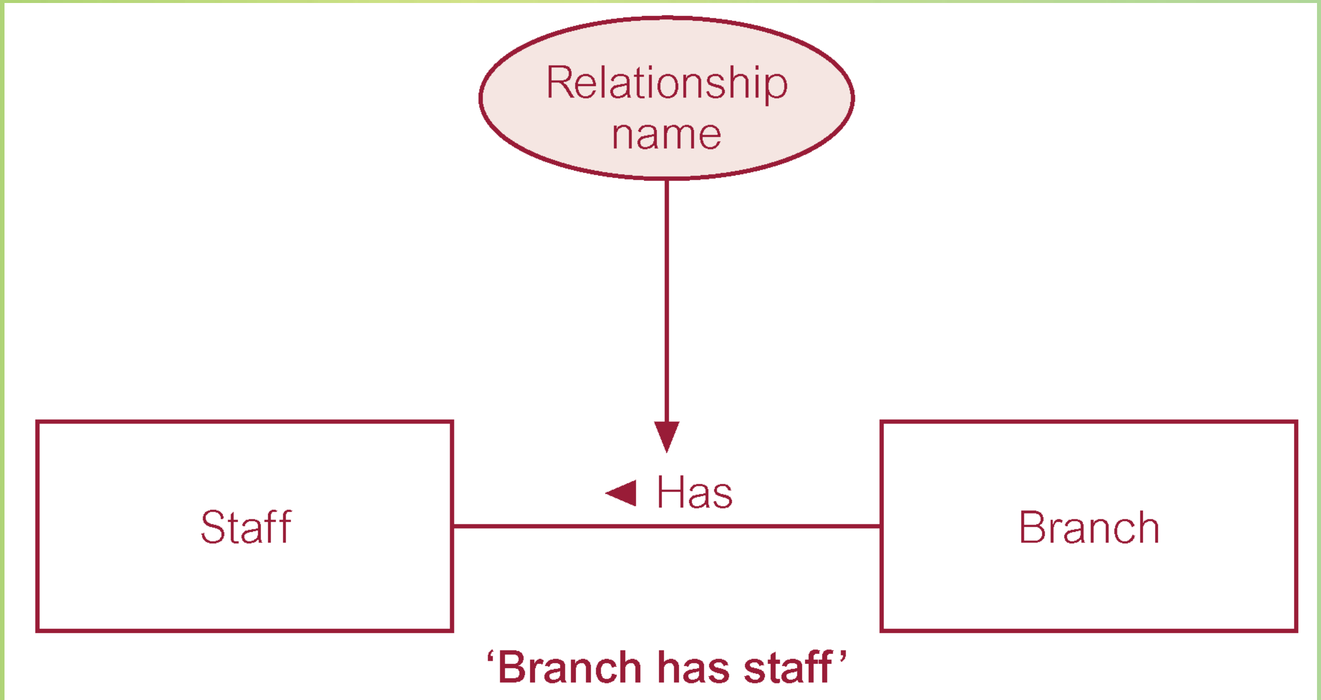


**Figure 12.4** A semantic net showing individual occurrences of the *Has* relationship type.

- We can examine examples of **individual occurrences of the Has relationship** using a ***semantic net***. A semantic net is an object-level model, which uses the symbol • to represent entities and the symbol ◈ to represent relationships.
- The semantic net in Figure 12.4 shows three examples of the Has relationships (denoted rl, r2, and r3). Each relationship describes an association of a single Branch entity occurrence with a single Staff entity occurrence.
- Relationships are represented by lines that join each participating Branch entity with the associated Staff entity. For example, relationship rl represents the association between Branch entity B003 and Staff entity SG37.

# ER diagram of Branch *Has* Staff relationship

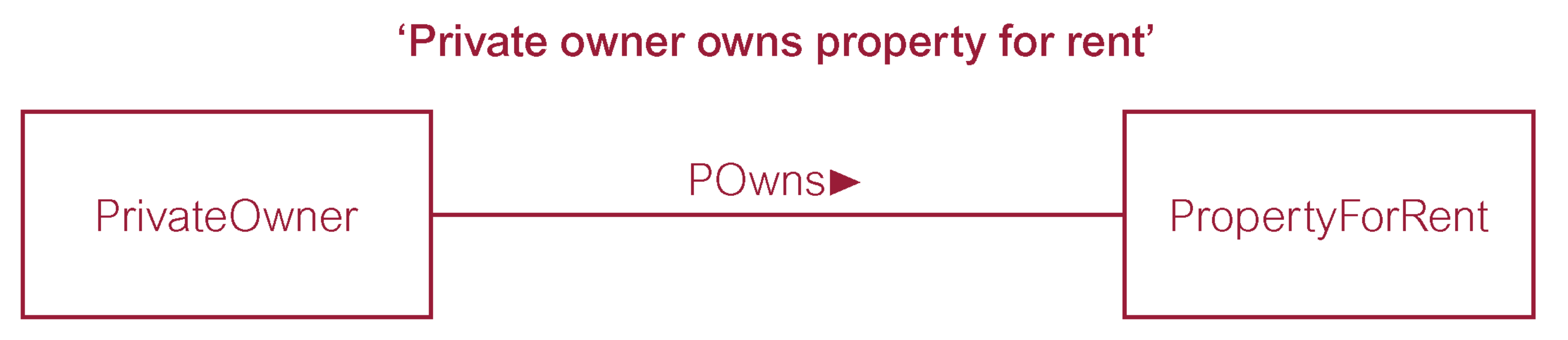
- Each relationship type is shown as a line connecting the associated entity types and labeled with the name of the relationship.
- Normally, a relationship is named using a verb (for example, Supervises or Manages) or a short phrase including a verb (for example, LeasedBy).



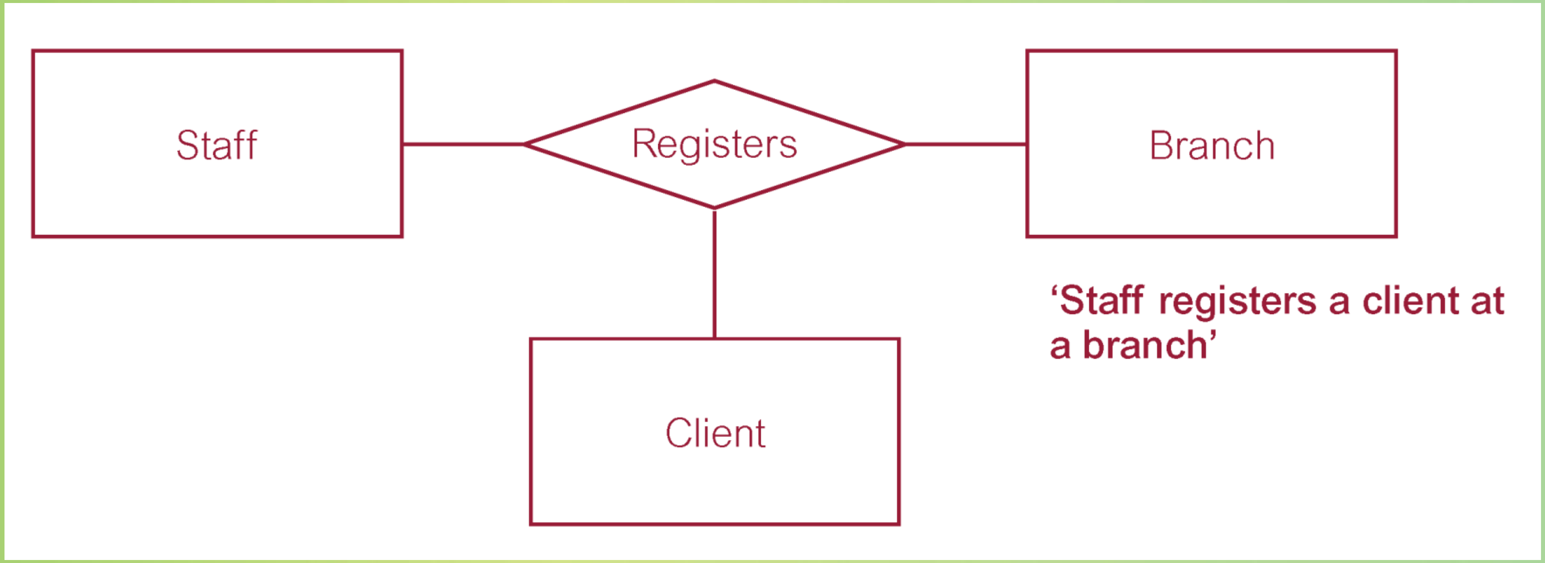
# Relationship Types

- Degree of a Relationship
  - Number of participating entities in relationship.
- Relationship of degree:
  - two is binary
  - three is ternary
  - four is quaternary.

# Binary relationship called *POwns*

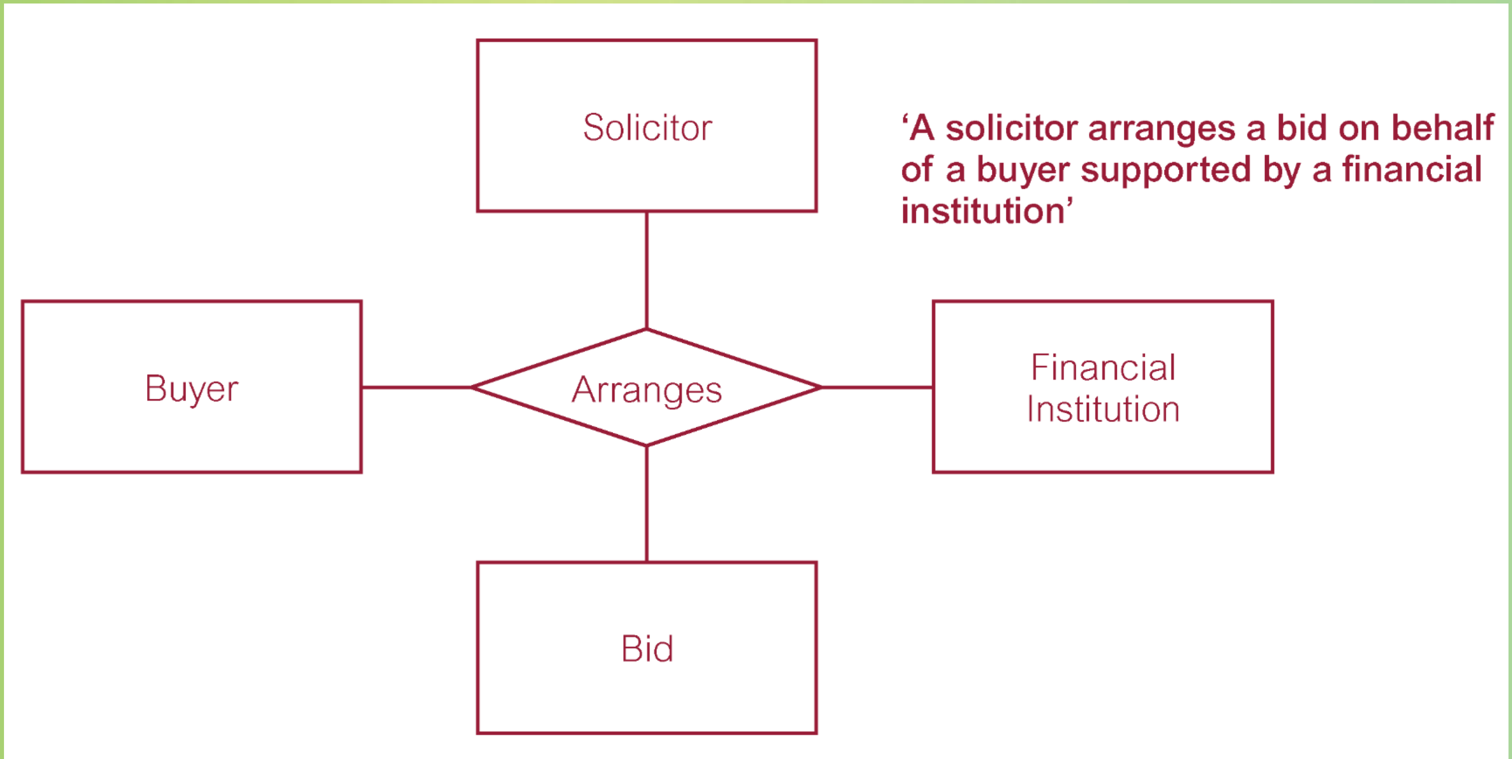


# Ternary relationship called *Registers*



'Staff registers a client at a branch'

# Quaternary relationship called *Arranges*



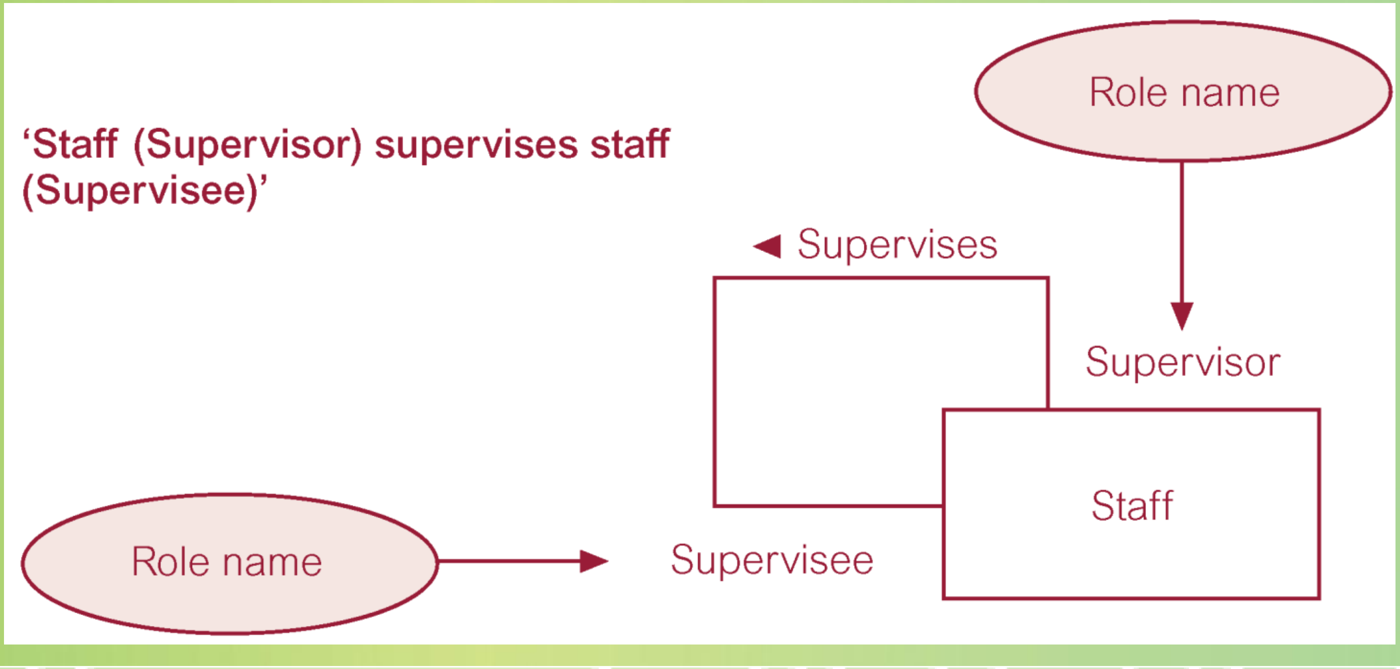
**‘A solicitor arranges a bid on behalf of a buyer supported by a financial institution’**

# Relationship Types

- **Recursive Relationship**
  - **Relationship type where *same* entity type participates more than once in *different roles*.**
- **Relationships may be given role names to indicate purpose that each participating entity type plays in a relationship.**
- **Recursive relationships are sometimes called unary relationships**

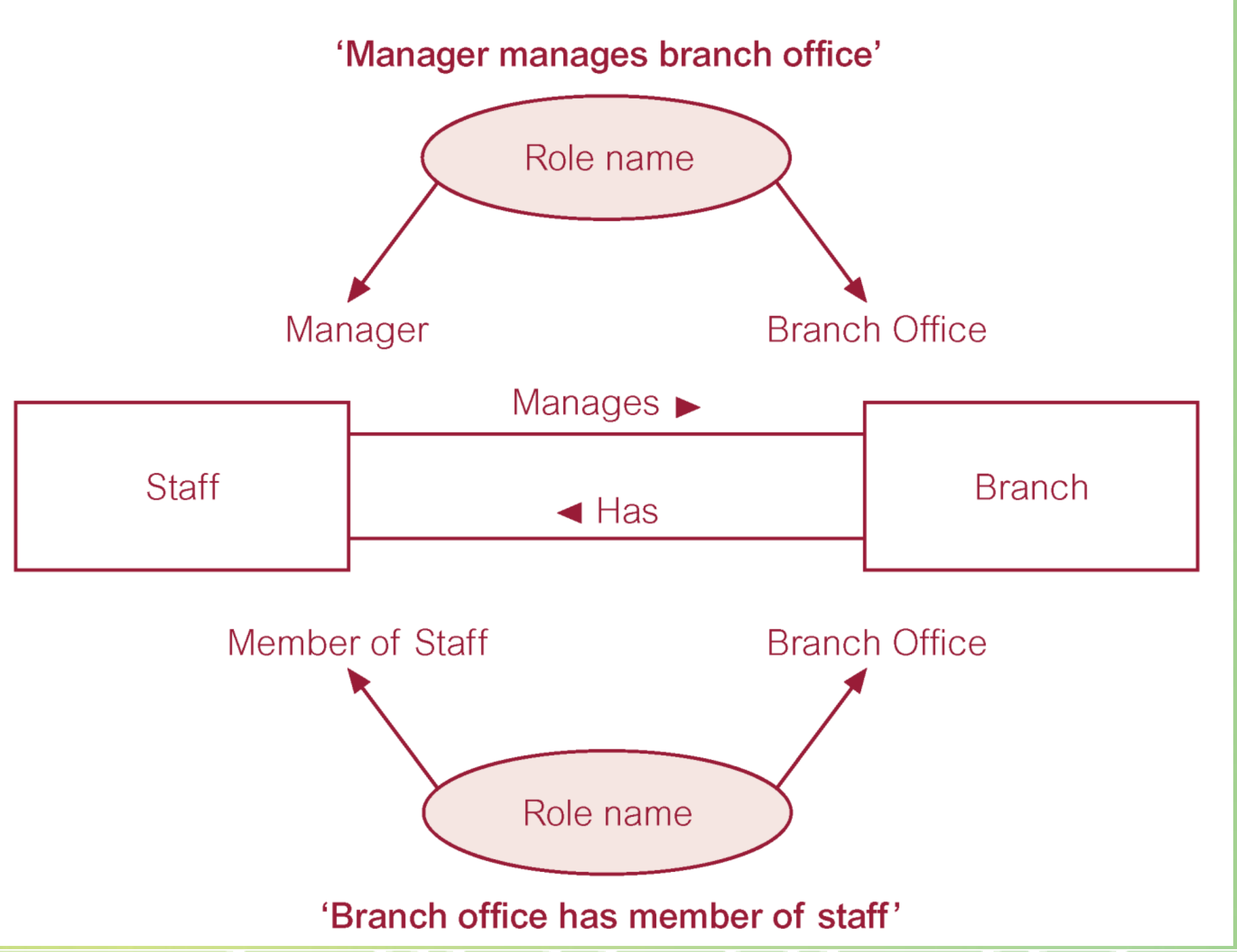
# Recursive relationship called *Supervises* with role names

- The use of role names to describe the *Supervises recursive relationship* is shown in the Figure.
- The first participation of the Staff entity type in the *Supervises relationship* is given the role name "**Supervisor**" and the second participation is given the role name "**Supervisee.**"



# Entities associated through two distinct relationships with role names

- Role names may also be used when two entities are associated through **more than one relationship**.
- For example, the *Staff* and *Branch* entity types are associated through two distinct relationships called **Manages** and **Has**.



# Attributes

- Attribute
    - Property of an entity or a relationship type.
- Attribute Domain
    - Set of allowable values for one or more attributes.
    - For example, the number of rooms associated with a property is between 1 and 15 for each entity occurrence. We therefore define the set of values for the number of rooms (rooms) attribute of the PropertyForRent entity type as the set of integers between 1 and 15.
- Attributes can be classified as being: simple or composite, single valued or multi-valued, or derived.

# Attributes

- Simple Attribute
  - An attribute composed of a single component with an independent existence.
  - Simple attributes cannot be further subdivided into smaller components.
  - Examples of simple attributes include position and salary of the Staff entity.
  - Simple attributes are sometimes called atomic attributes.
- Composite Attribute
  - An attribute composed of multiple components, each with an independent existence.
  - These attributes can be further divided to yield smaller components with an independent existence of their own.
  - For example, the address attribute of the Branch entity with the value (163 Main St, Glasgow, G11 9QX) can be subdivided into street (163 Main St), city (Glasgow), and postcode (G11 9QX) attributes.

# Attributes

- **Single-valued Attribute**
  - **Attribute that holds a single value for each occurrence of an entity type.**
  - **For example, each occurrence of the Branch entity type has a single value for the branch number (branchNo) attribute (for example, B003), and therefore the branchNo attribute is referred to as being single-valued.**
- **Multi-valued Attribute**
  - **Attribute that holds multiple values for each occurrence of an entity type.**
  - **For example, each occurrence of the Branch entity type can have multiple values for the telNo attribute (for example, branch number B003 has telephone numbers 0141-339-2178 and 0141-339-4439) and therefore the telNo attribute in this case is multi-valued.**

# Attributes

- **Derived Attribute**
  - **An attribute that represents a value that is derivable from value of a related attribute, or set of attributes, *not necessarily* in the same entity type.**
  - **For example, the value for the duration attribute of the Lease entity is calculated from the rentStart and rentFinish attributes (of the Lease entity type).**
  - ***Another example* is when an attribute called *deposit* of the *Lease* entity type. The value of the *deposit* attribute is calculated as twice the monthly rent for a property. Therefore, the value of the *deposit* attribute of the *Lease* entity type is derived from the *rent* attribute of the *PropertyForRent entity* type.**

# Keys

- **Candidate Key**
  - **Minimal set of attributes that uniquely identifies each occurrence of an entity type.**
  - **For example, the branch number (branchNo) attribute is the candidate key for the Branch entity type, and has a distinct value for each branch entity occurrence.**

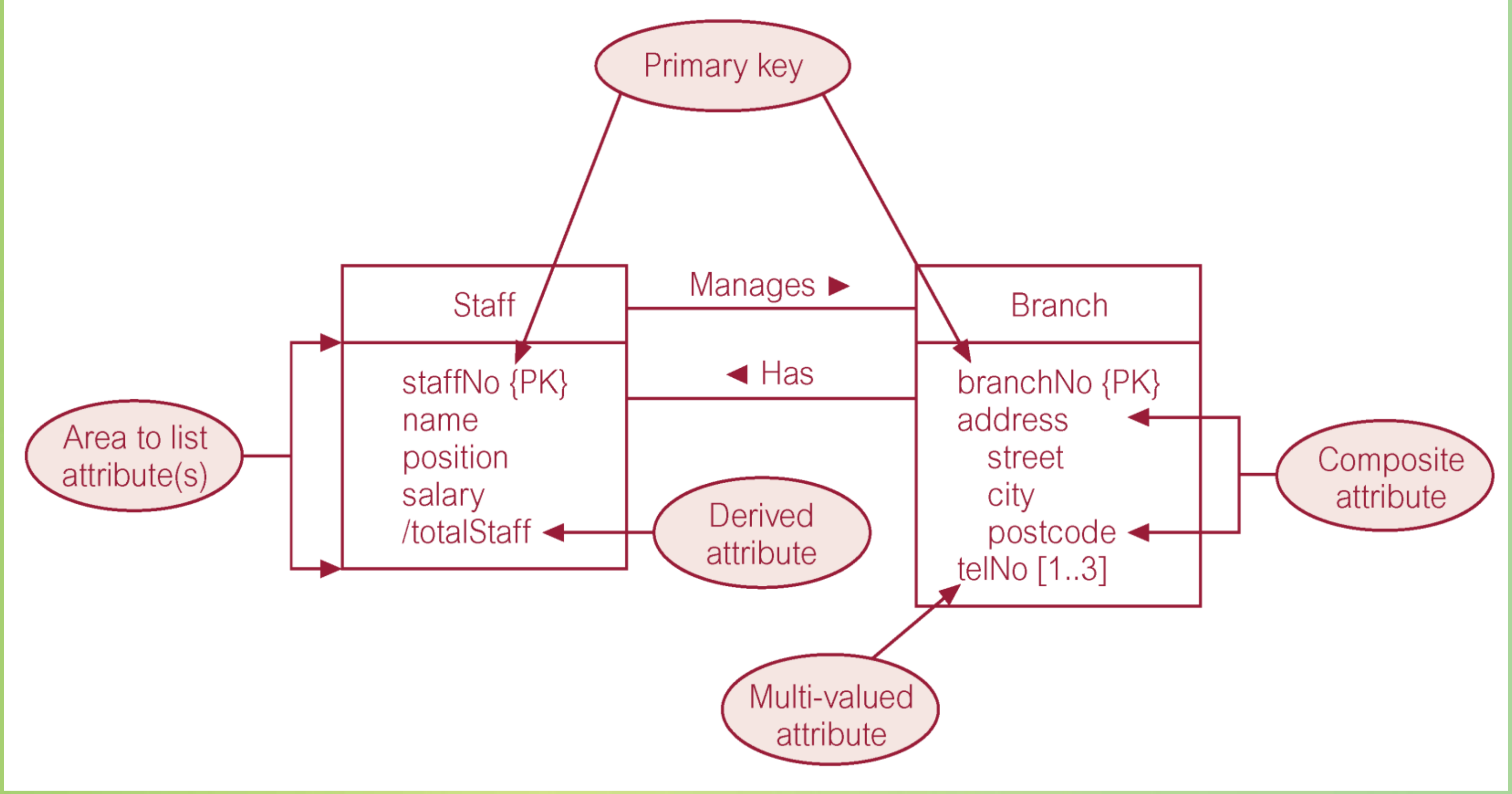
# Keys

- **Primary Key**
  - **The candidate key that is selected to uniquely identify each occurrence of an entity type.**
  - **An entity type may have more than one candidate key.**
  - **For example, a member of staff has a unique company-defined staff number (staffNo) and also a unique National Insurance Number (NIN). One of which must be selected as the primary key.**
  - **The choice of primary key for an entity is based on considerations of attribute length, the minimal number of attributes required, and the future certainty of uniqueness.**
  - **Suppose that the company has defined staff number to be a maximum of five characters (for example, SG14), and the NIN to be of a maximum of nine characters (for example, WL220658D).**
  - **In such case, we select staffNo as the primary key of the Staff entity type and NIN is then referred to as the alternate key.**

# Keys

- **Composite Key**
  - **A candidate key that consists of two or more attributes.**
  - **In some cases, the key of an entity type is composed of several attributes whose values <u>together are unique</u> for each entity occurrence <u>but not separately</u>.**
  - **For example, consider an entity called *Advert* with propertyNo (property number), newspaperName, dateAdvert, and cost attributes.**
  - **Many properties are advertised in many newspapers on a given date.**
  - **To uniquely identify each occurrence of the Advert, entity type requires values for the propertyNo, newspaperName, and dateAdvert attributes.**
  - **Thus, the Advert entity type has a composite primary key made up of the propertyNo, newspaperName, and dateAdvert attributes.**

# ER diagram of Staff and Branch entities and their attributes

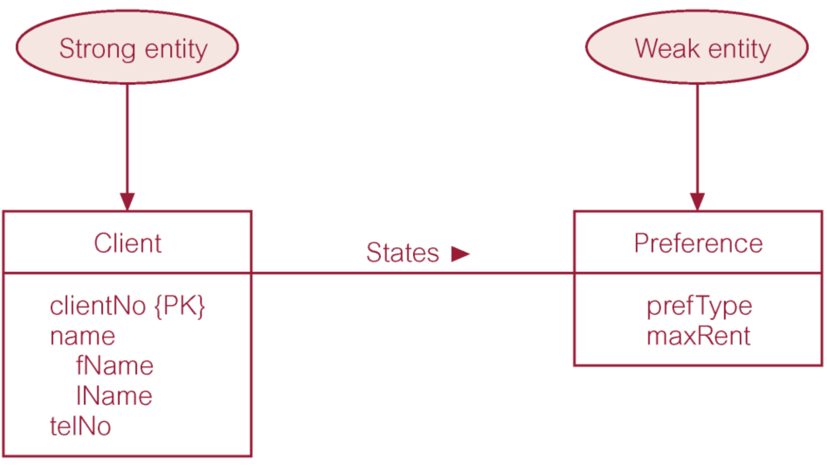


# Entity Type

- Strong Entity Type
  - An entity type that is *not* existence-dependent on some other entity type.
  - Examples of strong entities are shown in Figure 12.1 and include the Staff, Branch, PropertyForRent, and Client entities.
  - A characteristic of a strong entity type is that each entity occurrence is uniquely identifiable using the primary key attribute(s) of that entity type.
  - For example, we can uniquely identify each member of staff using the staffNo attribute, which is the primary key for the Staff entity type.

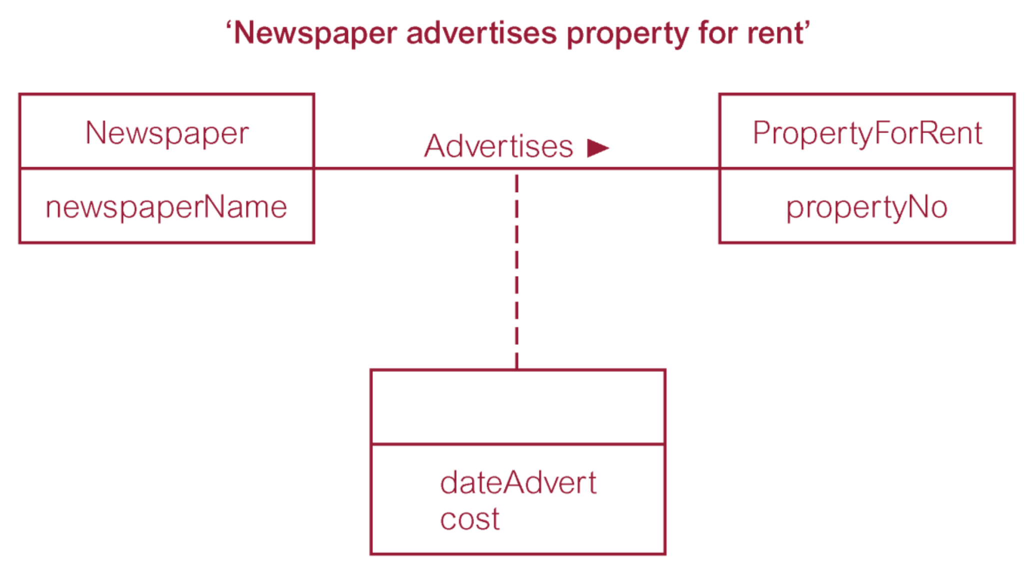
# Entity Type

- Weak Entity Type
  - An entity type that is existence-dependent on some other entity type.
  - An example of it is an entity called *Preference* shown in the Figure.
  - Since there is no primary key for the *Preference* entity, we cannot identify each occurrence of it using only its attributes.
  - We can uniquely identify each *preference* only through the relationship that a preference has with a *client* who is uniquely identifiable using the its primary key for namely clientNo.
  - Weak entity types are sometimes referred to as child, dependent, or subordinate entities and strong entity types as parent, owner, or dominant entities.



# Relationship called *Advertises* with attributes

- **Attributes can also be assigned to relationships.**
- **For example, consider the relationship *Advertises*, which associates the *Newspaper* and *PropertyForRent* entity types as shown in the Figure.**
- **To record the date on which the property was advertised and the cost, we associate this information with the *Advertises* relationship as attributes called *dateAdvert* and cost, rather than with the Newspaper or the PropertyForRent entities.**
- **To distinguish between a relationship with an attribute and an entity, the rectangle representing the attribute(s) is associated with the relationship using a dashed line.**



# Structural Constraints

- **Main type of constraint on relationships is called** *multiplicity***.**

- **Multiplicity: number (or range) of possible occurrences of** ***an entity type*** **that may relate to a single occurrence of** ***an associated entity*** **type** ***through a particular relationship*****.**

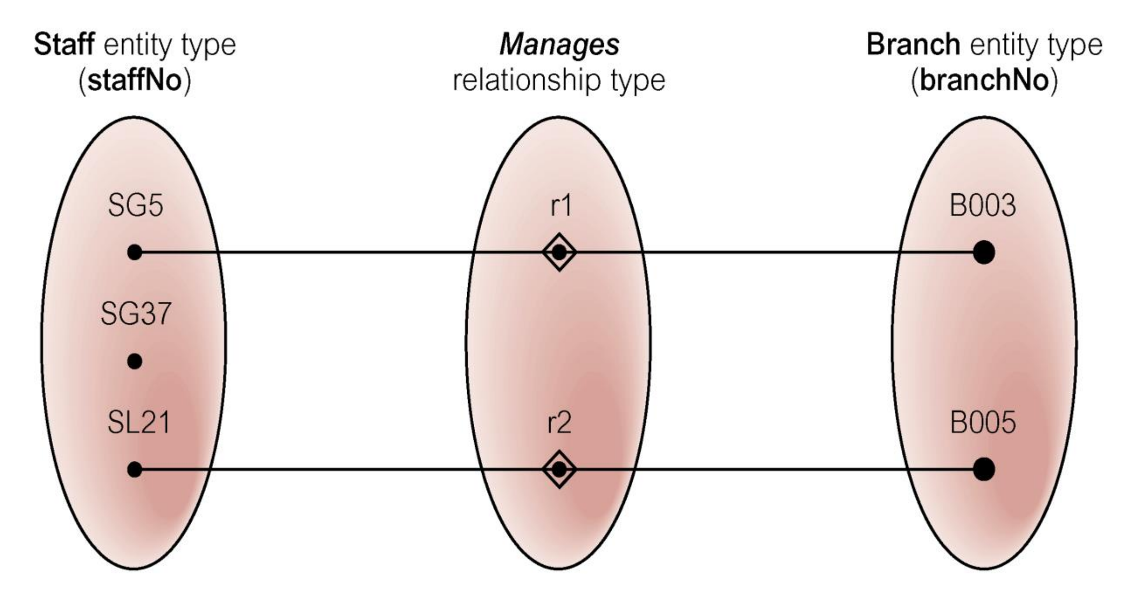
- **Represents policies (called** ***business rules*****) established by user or company.**

# Structural Constraints

- The most common degree for relationships is binary.
- Binary relationships are generally referred to as being:
  - one-to-one (1:1)
  - one-to-many (1:*)
  - many-to-many (*:*)

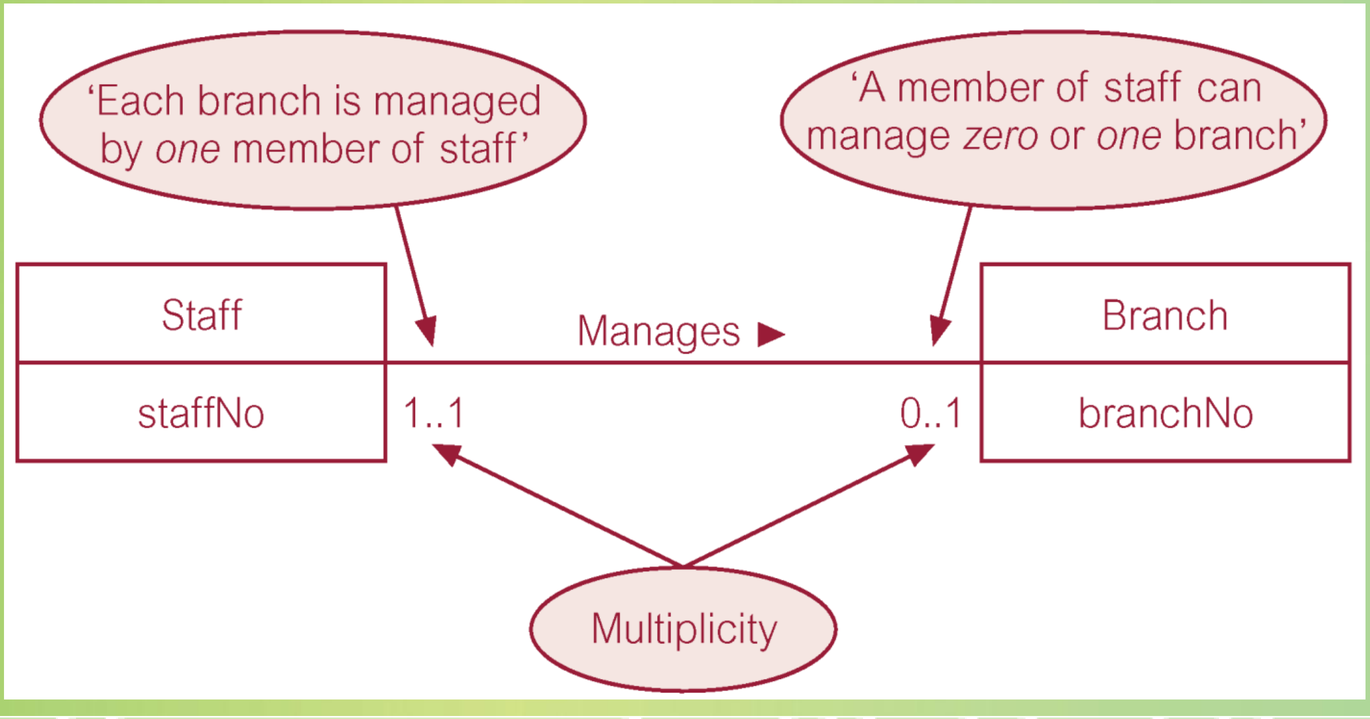
# Semantic net of Staff *Manages* Branch relationship type

- **Each relationship (r*n*) represents the association between a single Staff entity occurrence and a single Branch entity occurrence.**
- **We represent each entity occurrence using the values for the primary key attributes of the Staff and Branch entities, namely staffNo and branchNo.**
- **We see that staffNo SG5 manages branchNo B003 and staffNo SL21 manages branchNo BOO5, but staffNo SG37 does not manage any branch.**



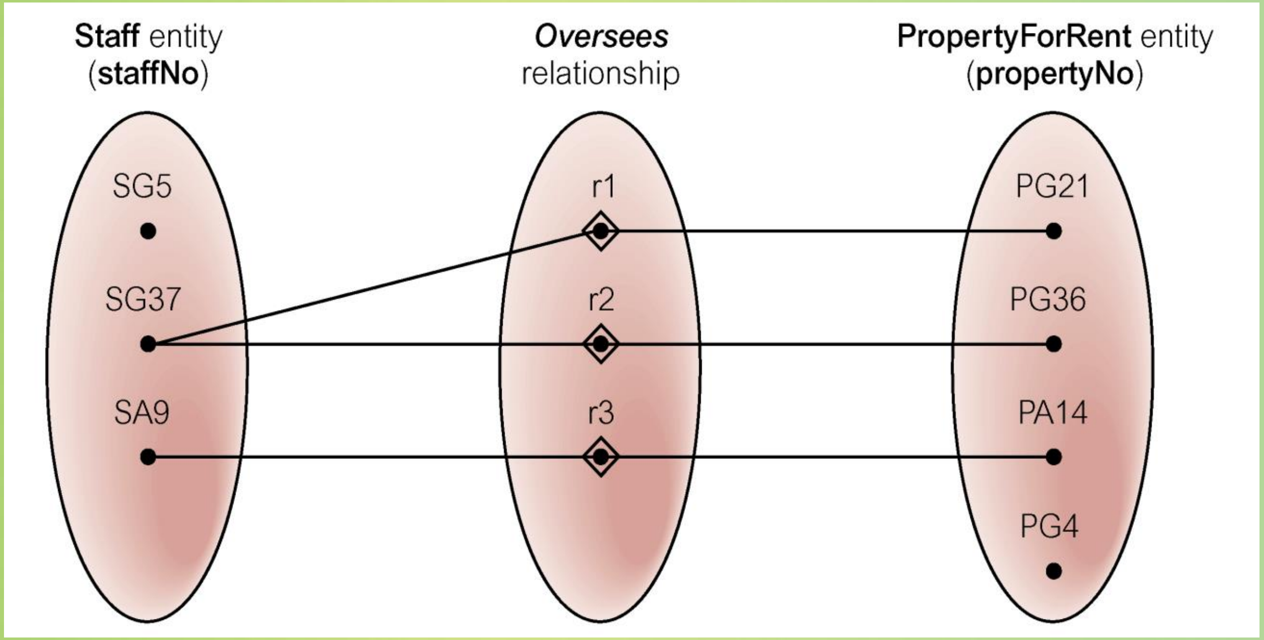
# Multiplicity of Staff *Manages* Branch (1:1) relationship

- As there is a maximum of one branch for each member of staff involved in this relationship and a maximum of one member of staff for each branch, we refer to this type of relationship as *one-to-one*, which we usually abbreviate as (1:1).


'Each branch is managed by *one* member of staff'
'A member of staff can manage *zero* or *one* branch'
Staff
staffNo
Manages ►
1..1
0..1
Branch
branchNo
Multiplicity

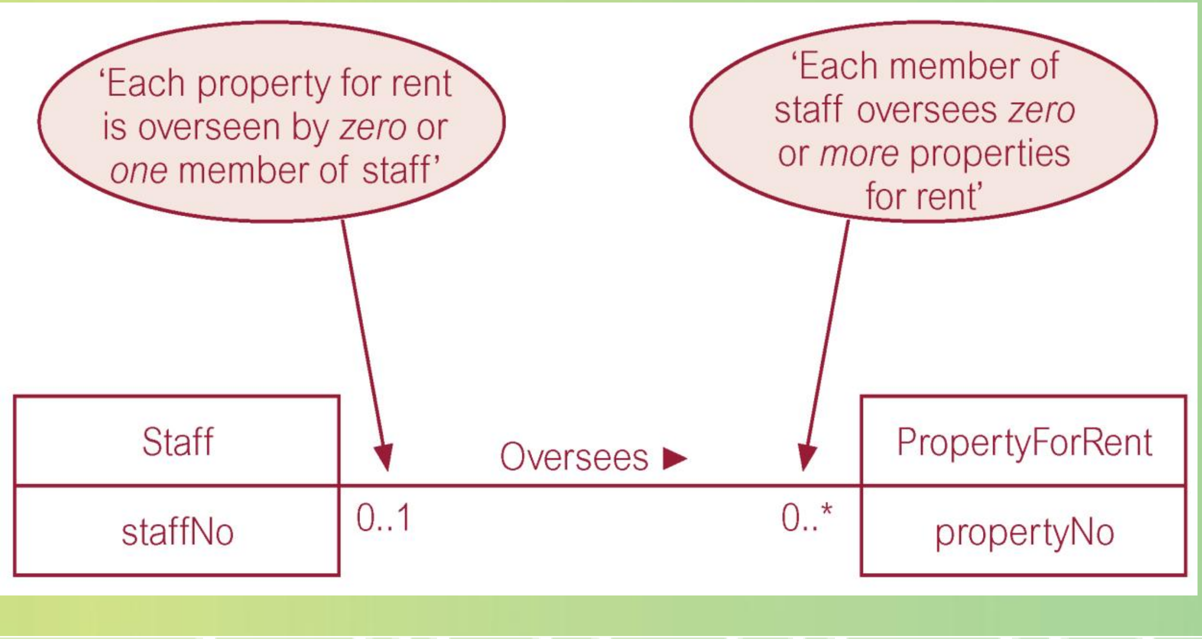
# Semantic net of Staff *Oversees* PropertyForRent relationship type

- **The relationship *Oversees* relates the Staff and PropertyForRent entity types.**
- **Each relationship (r*n*) represents the association between a single Staff entity occurrence and a single PropertyForRent entity occurrence.**



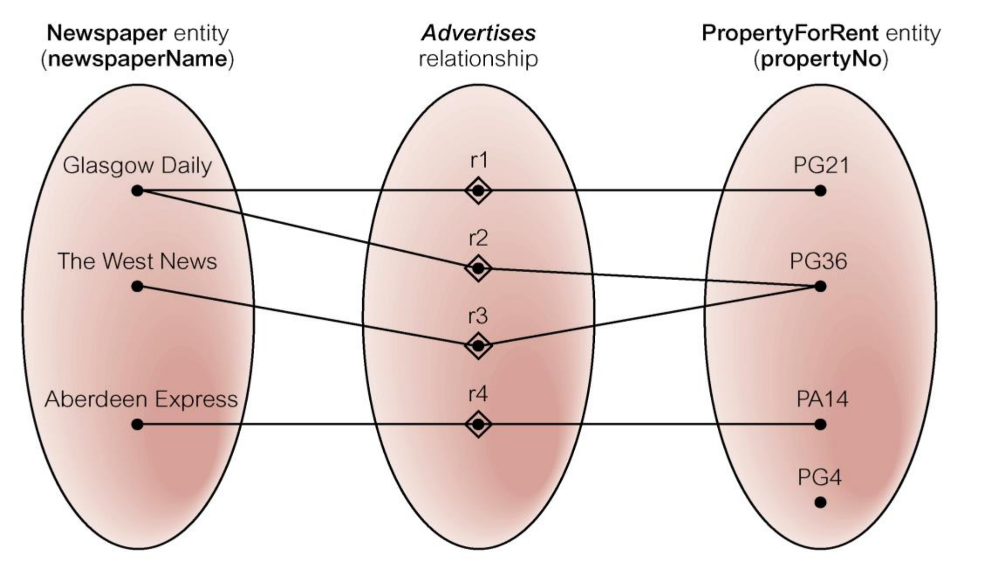
# Multiplicity of Staff *Oversees* PropertyForRent (1:*) relationship type

- We can see that a member of staff can oversee *zero or more* properties for rent and a propertyforrent is overseen by *zero or one* member of staff. Therefore, for members of staff participating in this relationship there are *many* properties for rent, and for properties participating in this relationship there is a maximum of *one* member of staff.
- We refer to this type of relationship as *one-to-many*, which we usually abbreviate as (1:*).


'Each property for rent is overseen by *zero* or *one* member of staff'
'Each member of staff oversees *zero* or *more* properties for rent'
Staff
staffNo
Oversees ►
0..1
0..*
PropertyForRent
propertyNo

# Semantic net of Newspaper *Advertises* PropertyForRent relationship type

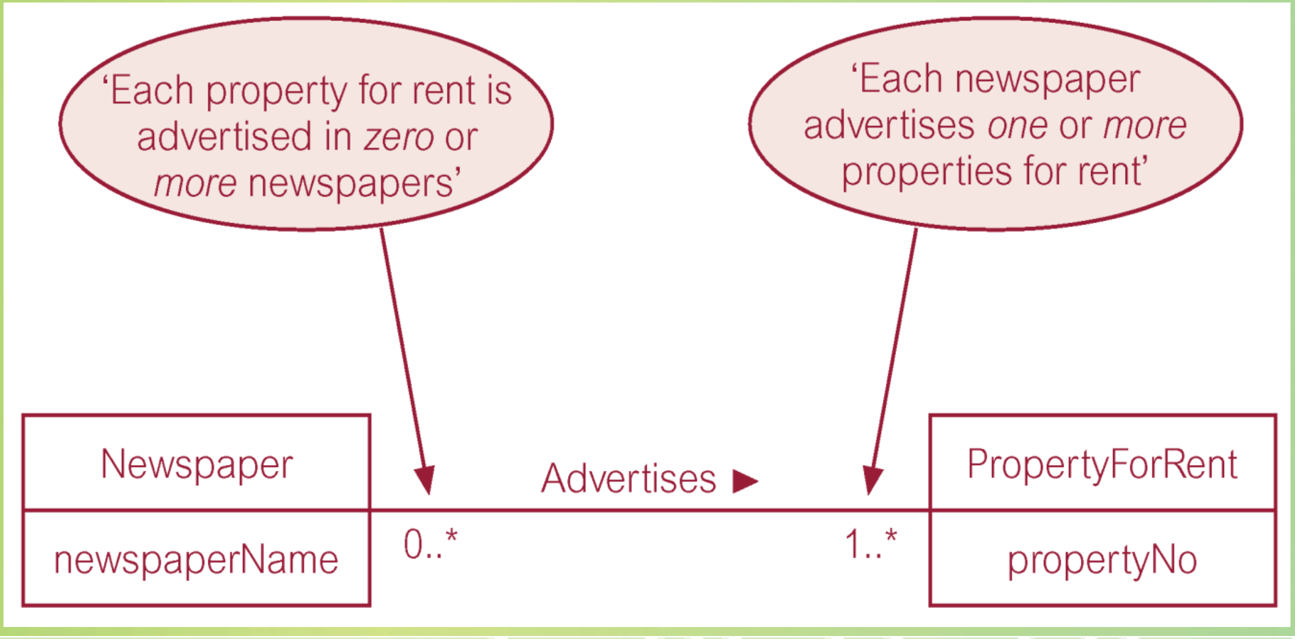
- Consider the relationship Advertises, which relates the Newspaper and PropertyForRent entity types.
- The figure displays four occurrences of the Advertises relationship (denoted rl, r2, r3, and r4) using a semantic net.
- Each relationship (r*n*) represents the association between a single Newspaper entity occurrence and a single PropertyForRent entity occurrence.



**Newspaper** entity (**newspaperName**)
***Advertises*** relationship
**PropertyForRent** entity (**propertyNo**)
Glasgow Daily
The West News
Aberdeen Express
r1
r2
r3
r4
PG21
PG36
PA14
PG4

# Multiplicity of Newspaper *Advertises* PropertyForRent (*:*) relationship

- We can see that one newspaper advertises *one or more* properties for rent and *one* property for rent is advertised in *zero or more* newspapers.
- Therefore, for newspapers there are *many* properties for rent, and for each property for rent participating in this relationship there are *many* newspapers. We refer to this type of relationship as many-to-many, which we usually abbreviate as (*:*).

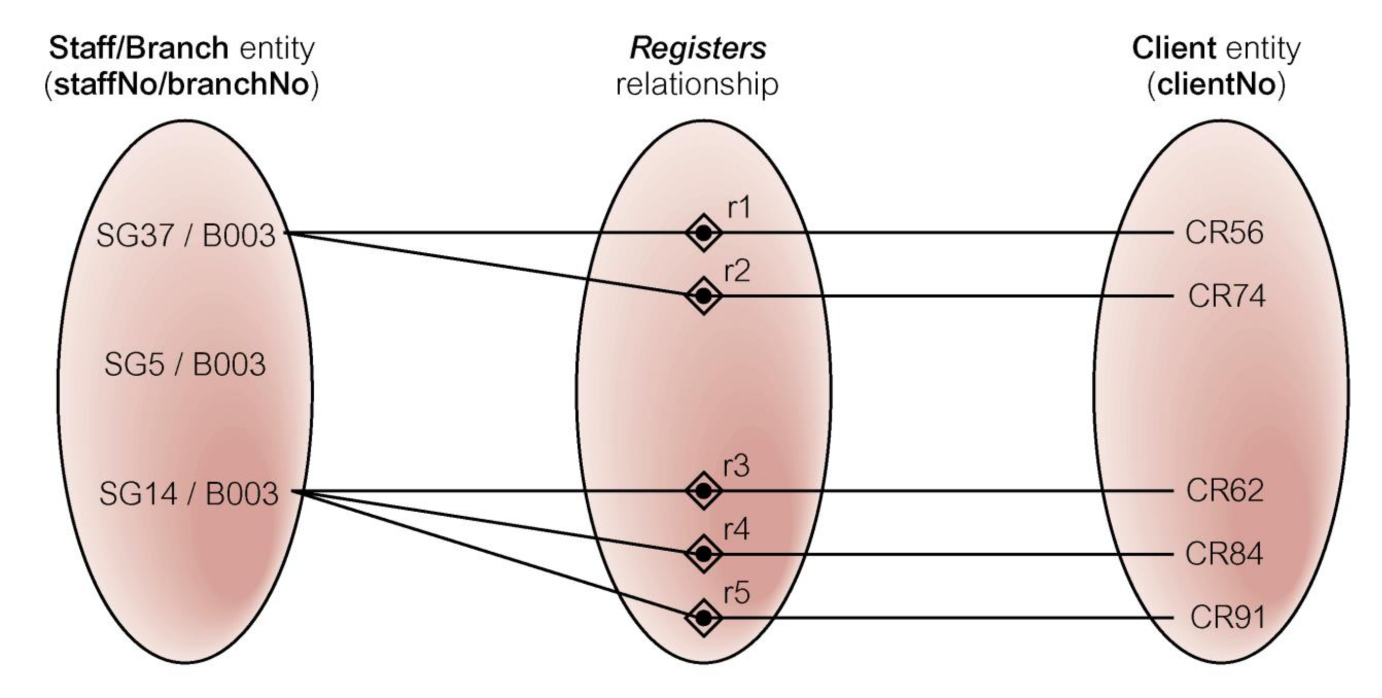

'Each property for rent is advertised in *zero* or *more* newspapers'
'Each newspaper advertises *one* or *more* properties for rent'
Newspaper
newspaperName
Advertises ▶
0..*
1..*
PropertyForRent
propertyNo

# Structural Constraints

- **Multiplicity for complex relationships (higher than binary) is slightly more complex.**
- **Multiplicity for Complex Relationships**
  - **The number (or range) of possible occurrences of an entity type in an *n*-ary relationship when other (*n*-1) values are fixed.**

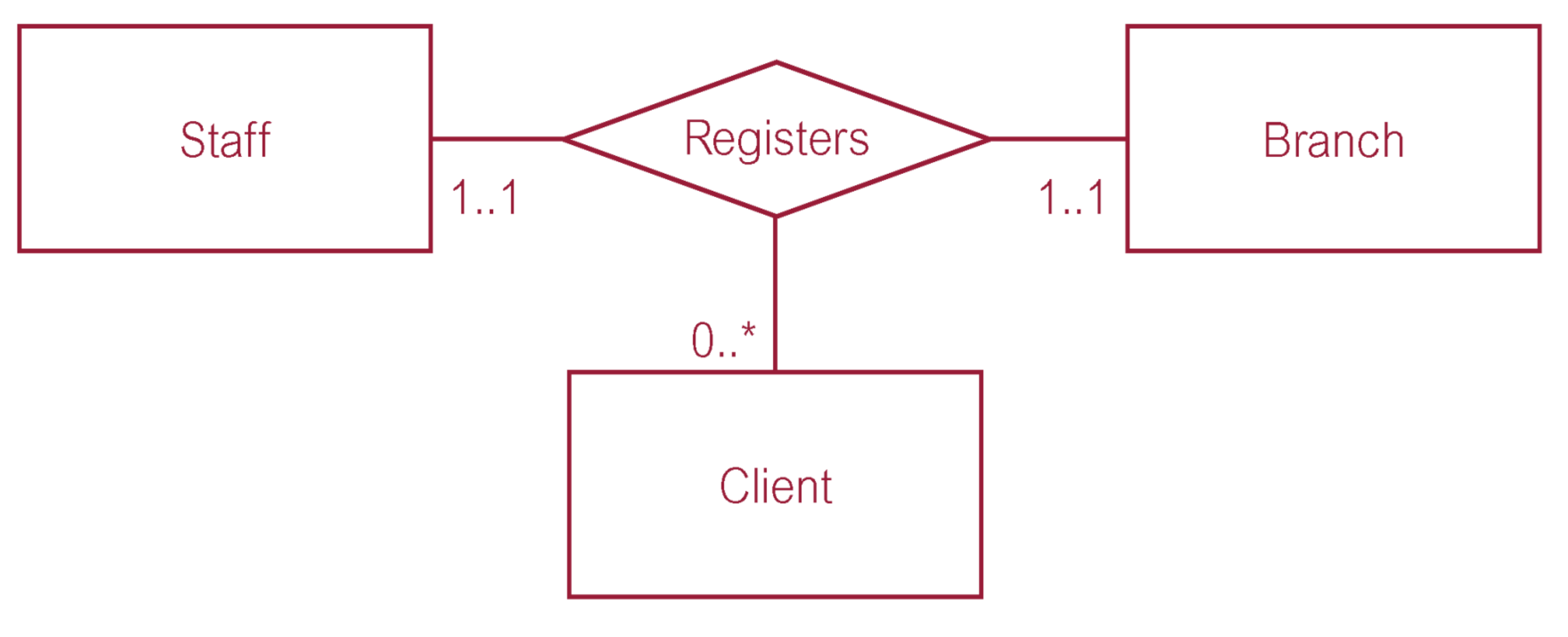
# Semantic net of ternary *Registers* relationship with values for Staff and Branch entities fixed

- **In the figure, we can notice that with the staffNo/branchNo values fixed there are *zero or more* clientNo values.**
- **For example, staffNo SG37 at branchNo B003 registers clientNo CR56 and CR74, and staffNo SG14 at branchNo B003 registers clientNo CR62, CR84, and CR91.**
- **However, SG5 at branchNo B003 registers no clients. In other words, when the staffNo and branchNo values are fixed the corresponding clientNo values are *zero or more.***
- **Therefore, the multiplicity of the Registers relationship from the perspective of the Staff and Branch entities is 0..*, which is represented in the ER diagram by placing the 0..* beside the Client entity.**

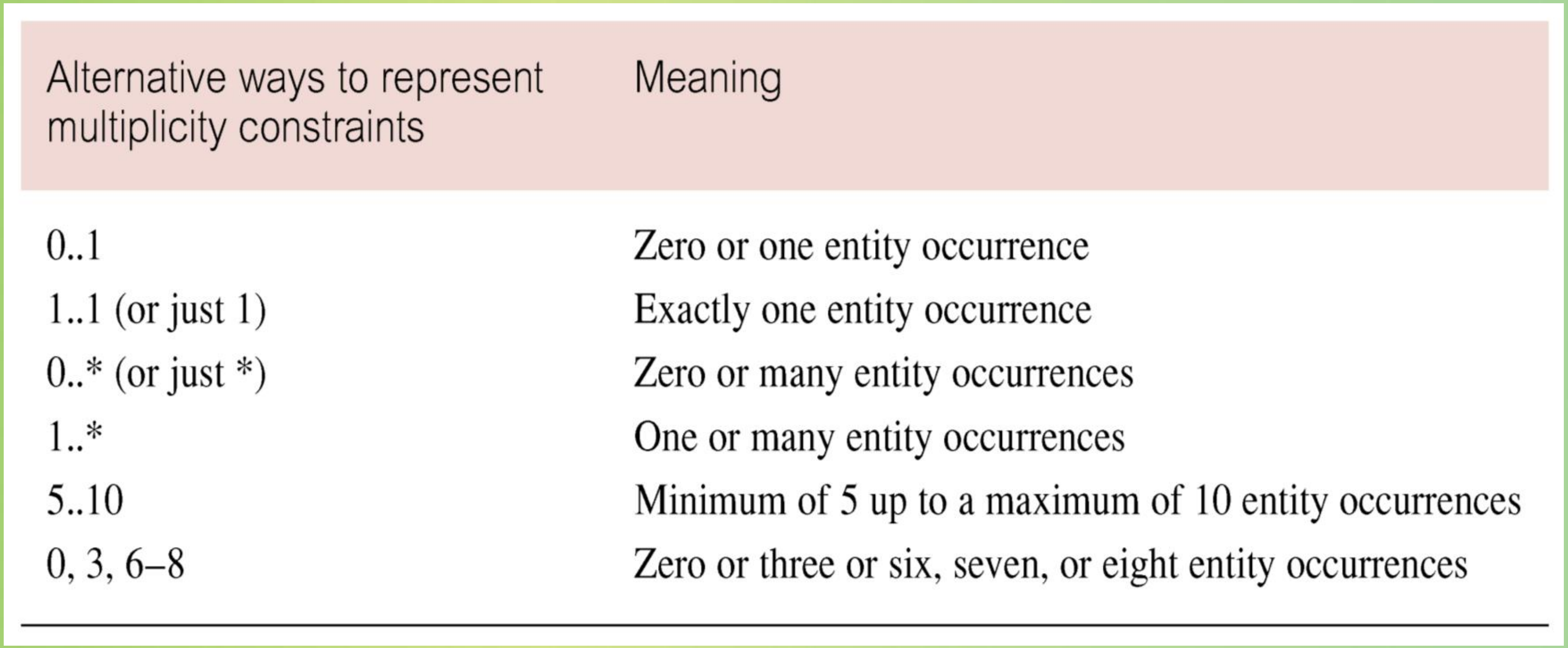


# Multiplicity of ternary *Registers* relationship

- **If we repeat this test we find that the multiplicity when Staff/Client values are fixed is 1..1, which is placed beside the Branch entity, and we can see also that the Client/Branch values are fixed is 1..1, which is placed beside the Staff entity.**



# Summary of multiplicity constraints



| Alternative ways to represent multiplicity constraints | Meaning |
|---|---|
| 0..1 | Zero or one entity occurrence |
| 1..1 (or just 1) | Exactly one entity occurrence |
| 0..* (or just *) | Zero or many entity occurrences |
| 1..* | One or many entity occurrences |
| 5..10 | Minimum of 5 up to a maximum of 10 entity occurrences |
| 0, 3, 6–8 | Zero or three or six, seven, or eight entity occurrences |

# Structural Constraints

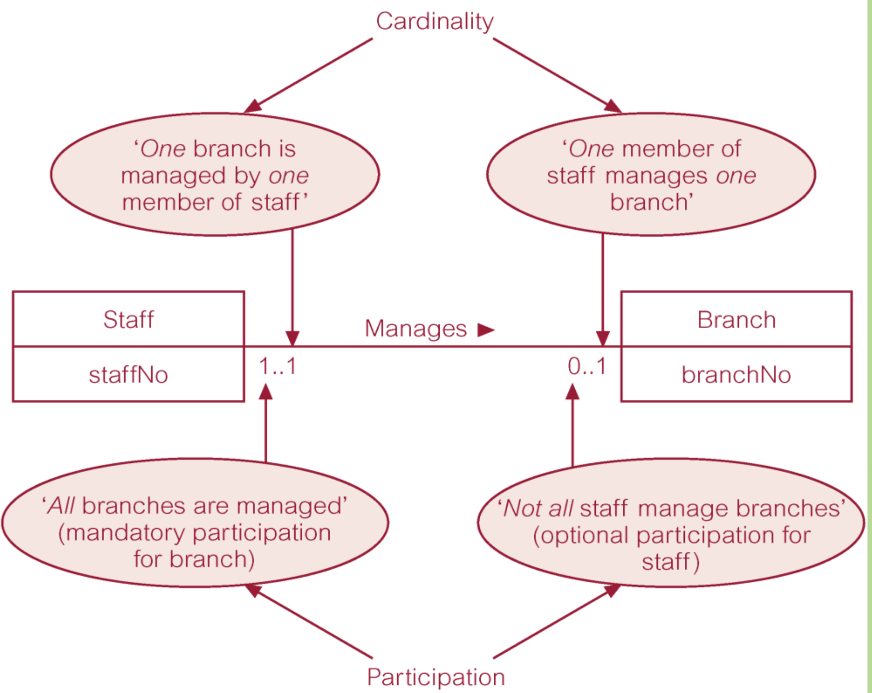
- Multiplicity is made up of two types of restrictions on relationships: *cardinality* and *participation*.

# Structural Constraints

- Cardinality
  - Describes maximum number of possible relationship occurrences for an entity participating in a given relationship type.
- Participation
  - Determines whether all or only some entity occurrences participate in a relationship.

# Multiplicity as cardinality and participation constraints

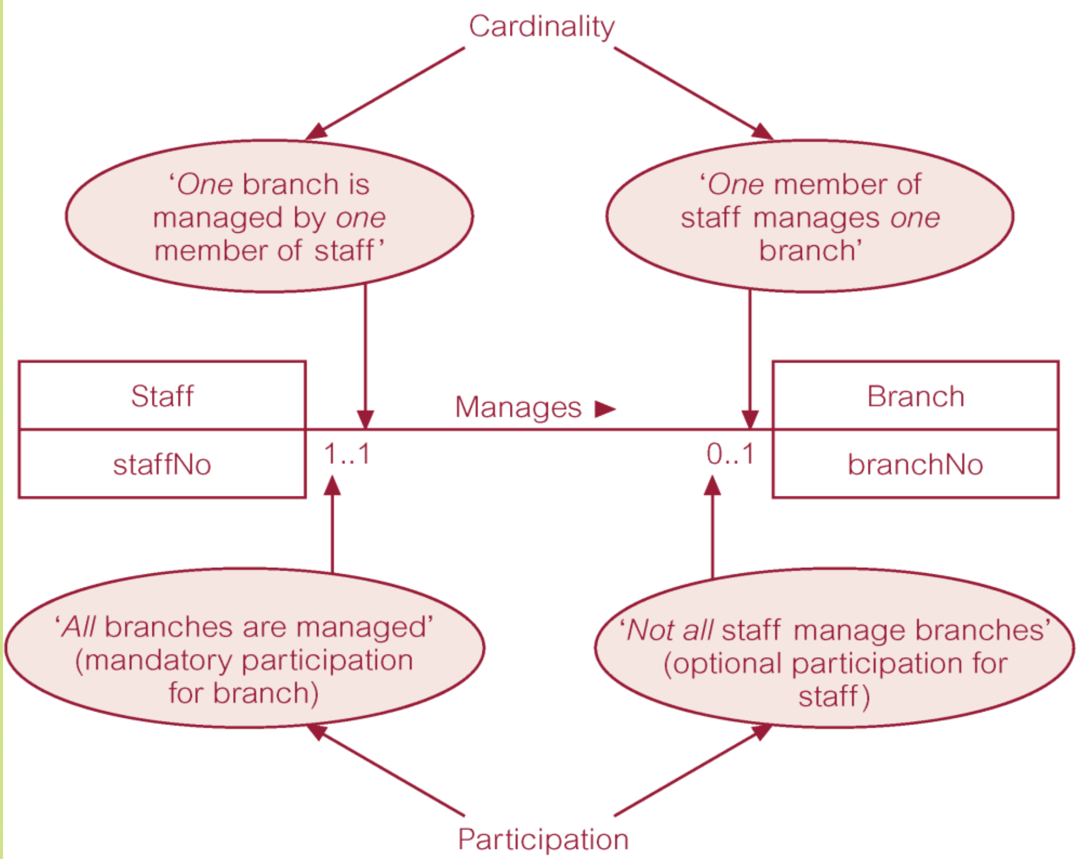
- **The cardinality of a binary relationship is what we previously referred to as a one-to-one (1:1), one-to-many (1:*), and many-to-many (*:*).**
- **The cardinality of a relationship appears as the *maximum* values for the multiplicity ranges *on either side of the relationship*.**
- **For example, the Manages relationship shown in the figure has a one-to-one (1:1) cardinality, and this is represented by multiplicity ranges with a maximum value of 1 on both sides of the relationship.**



Cardinality
'*One* branch is managed by *one* member of staff'
'*One* member of staff manages *one* branch'
Staff
staffNo
Manages ►
1..1
0..1
Branch
branchNo
'*All* branches are managed' (mandatory participation for branch)
'*Not all* staff manage branches' (optional participation for staff)
Participation

# Multiplicity as cardinality and participation constraints

- The participation of entities in a relationship appears as the *minimum* values for the multiplicity ranges *on either side* of the relationship.
- Optional participation is represented as a minimum value of 0, and mandatory participation is shown as a minimum value of 1.
- The participation for a given entity in a relationship is represented by the minimum value on *the opposite side* of the relationship.
- For example, the optional participation for the Staff entity in the Manages relationship is shown as a minimum value of 0 for the multiplicity beside the Branch entity and the mandatory participation for the Branch entity in the Manages relationship is shown as a minimum value of 1 for the multiplicity beside the Staff entity


Cardinality
'*One* branch is managed by *one* member of staff'
'*One* member of staff manages *one* branch'
Staff
staffNo
Manages ►
1..1
0..1
Branch
branchNo
'*All* branches are managed' (mandatory participation for branch)
'*Not all* staff manage branches' (optional participation for staff)
Participation

# Problems with ER Models

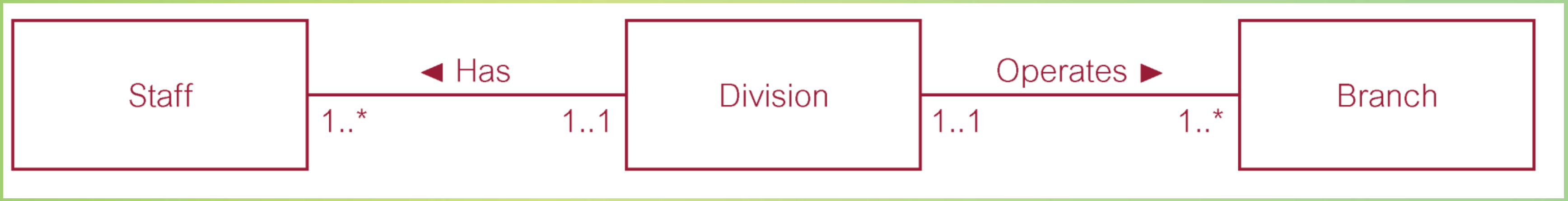
- Problems may arise when designing a conceptual data model called *connection traps*.
- Often due to a misinterpretation of the meaning of certain relationships.
- Two main types of connection traps are called *fan traps* and *chasm traps*.

# Problems with ER Models

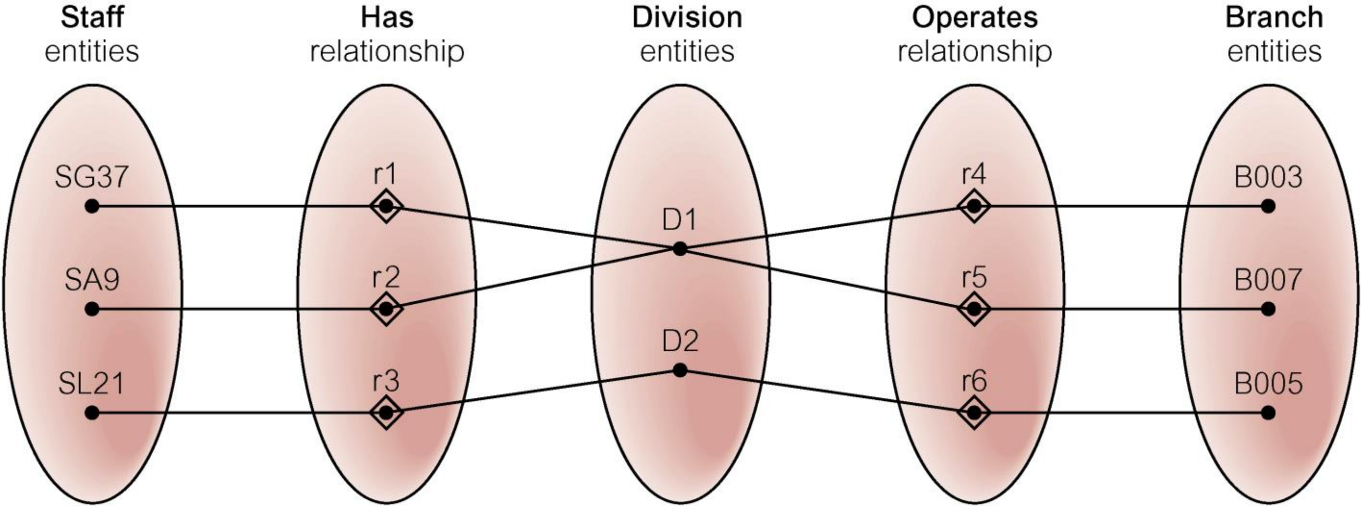
- **Fan Trap**
  - **Where a model represents a relationship between entity types, but pathway between certain entity occurrences is ambiguous.**
- **Chasm Trap**
  - **Where a model suggests the existence of a relationship between entity types, but pathway does not exist between certain entity occurrences.**

# An Example of a Fan Trap

- **A fan trap may exist where two or more 1:* relationships fan out from the same entity.**
- **The model in the figure represents the facts that a single division operates *one or more* branches and has *one or more* staff. However, a problem arises when we want to know which members of staff work at a particular branch.**

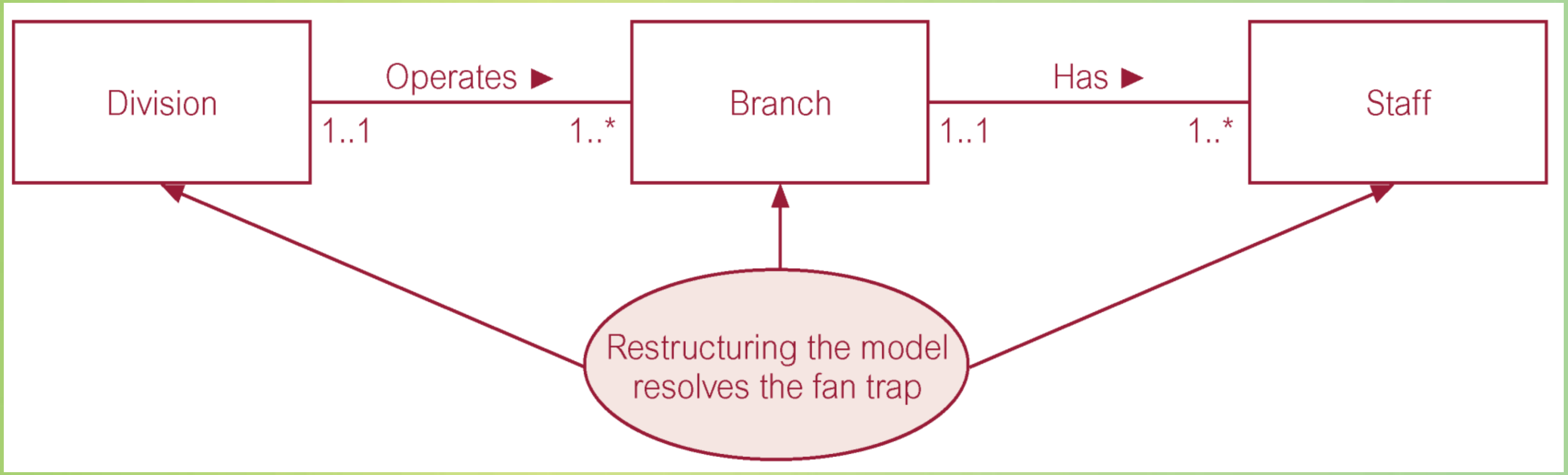


# Semantic Net of ER Model with Fan Trap

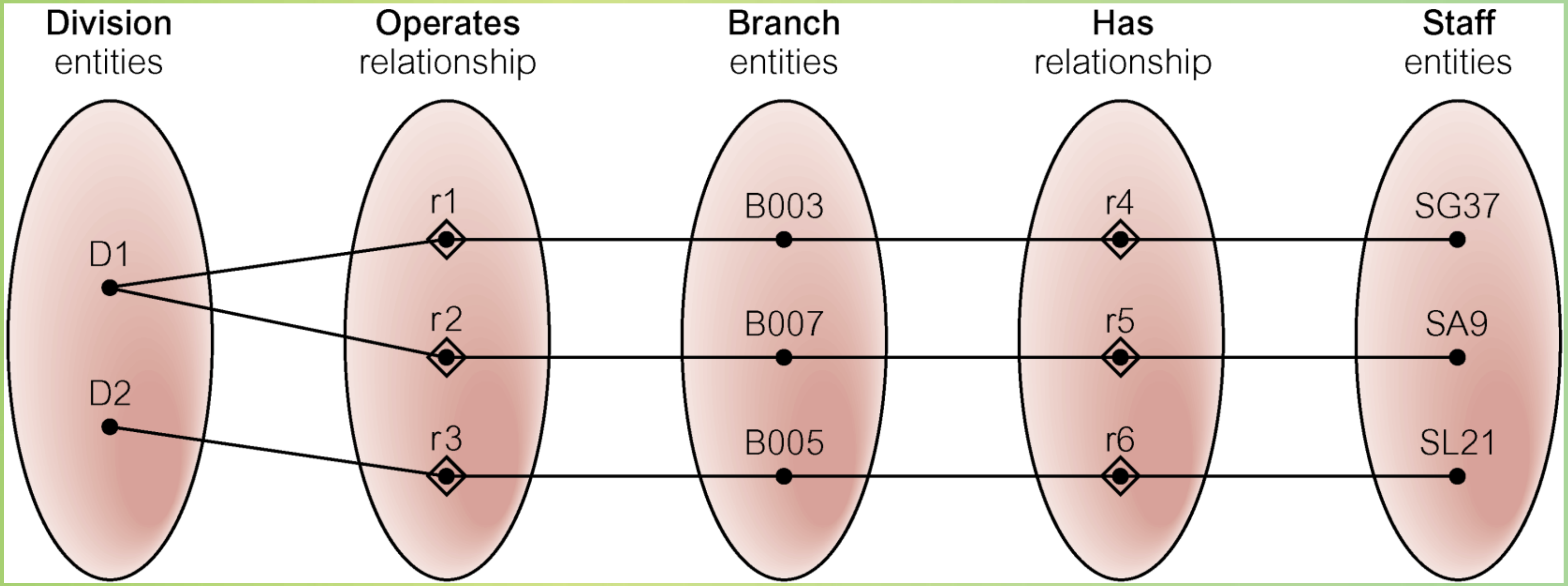


- At which branch office does staff number SG37 work?
- We are unable to give a specific answer based on the current structure.
- We can determine only that staff number SG37 works at Branch B003 or B007.
- The inability to answer this question specifically is the result of a fan trap associated with the misrepresentation of the correct relationships between the Staff, Division, and Branch entities.

# Restructuring ER model to remove Fan Trap



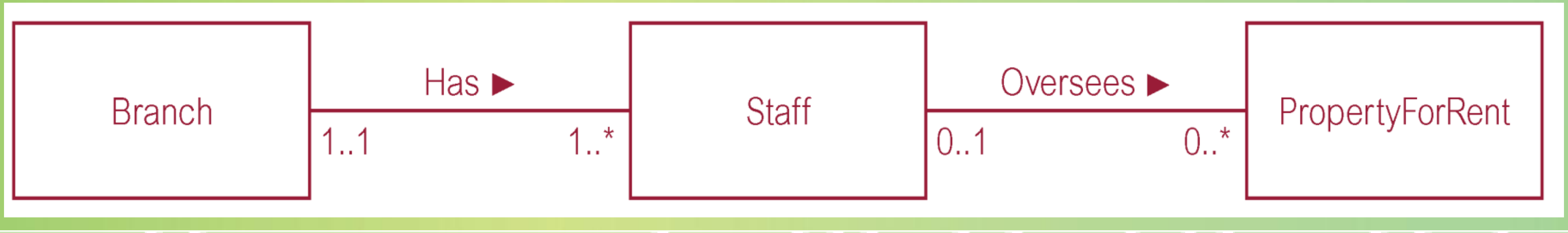
# Semantic Net of Restructured ER Model with Fan Trap Removed



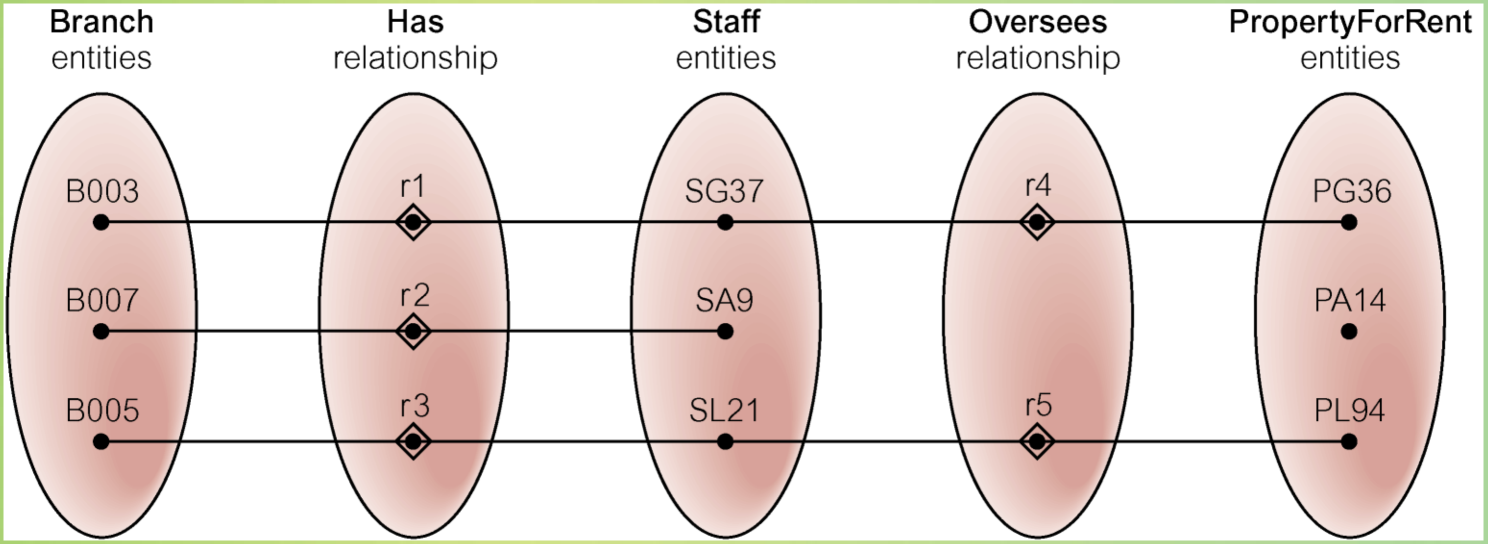
- **SG37 works at branch B003.**

# An Example of a Chasm Trap

- A chasm trap may occur where there are one or more relationships with a minimum multiplicity of zero (that is, optional participation) forming part of the pathway between related entities.
- The model in the figure shows relationships between the Branch, Staff, and PropertyForRent entities.
- This model represents the facts that a single branch has one or more staff who oversee zero or more properties for rent.
- We also note that not all staff oversee property, and not all properties are overseen by a member of staff.
- A problem arises when we want to know which properties are available at each branch.



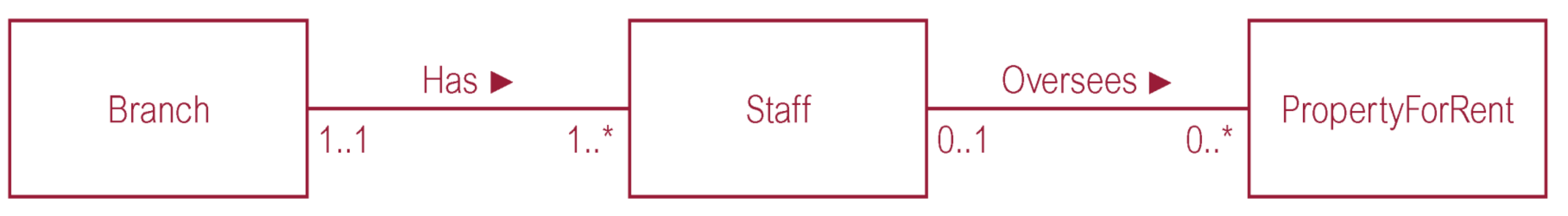
# Semantic Net of ER Model with Chasm Trap



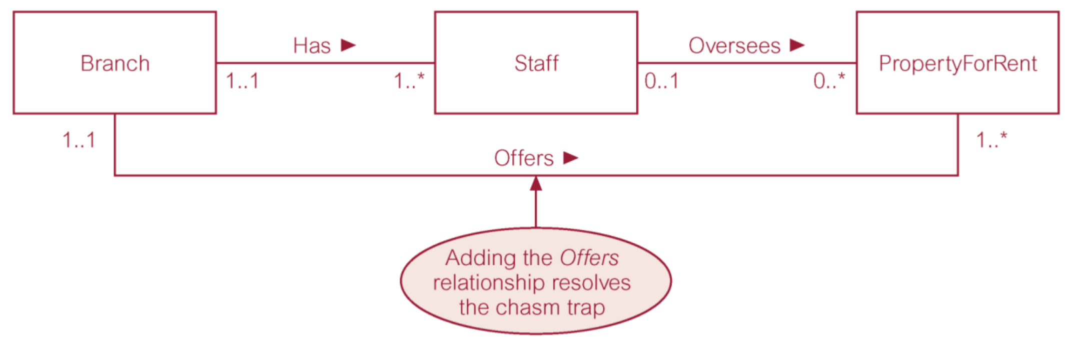
- **At which branch office is property PA14 available?**
- **We are unable to answer this question, as this property is not yet allocated to a member of staff working at a branch.**
- **The inability to answer this question is considered to be a loss of information (as we know a property must be available at a branch), and is the result of a chasm trap.**

# ER Model restructured to remove Chasm Trap

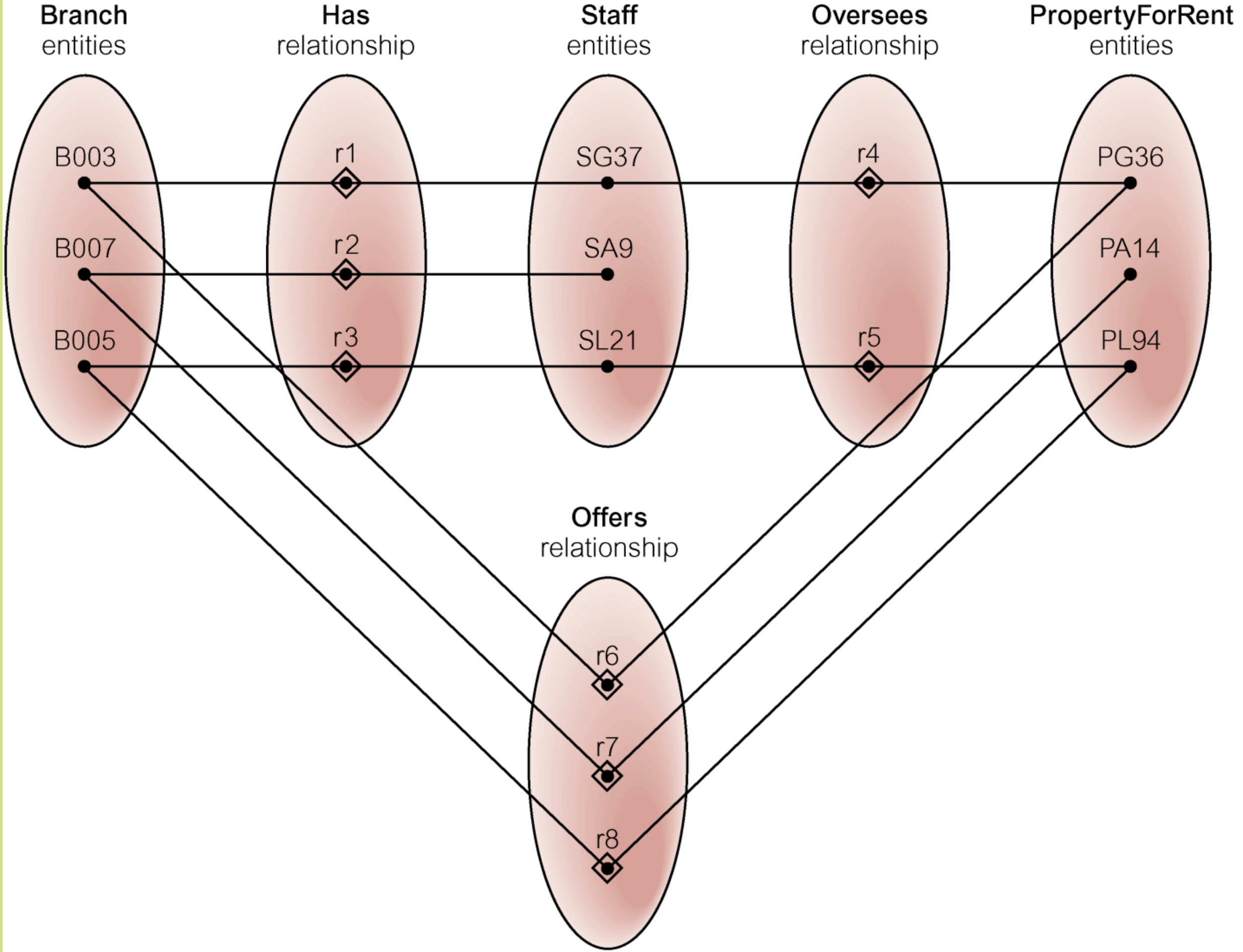
- The multiplicity of both the Staff and PropertyForRent entities in the Oversees relationship has a minimum value of zero, which means that some properties cannot be associated with a branch through a member of staff.



- Therefore, to solve this problem, we need to identify the missing relationship, which in this case is the Offers relationship between the Branch and PropertyForRent entities.



# Semantic Net of Restructured ER Model with Chasm Trap Removed


Branch
entities
Has
relationship
Staff
entities
Oversees
relationship
PropertyForRent
entities
B003
B007
B005
r1
r2
r3
SG37
SA9
SL21
r4
r5
PG36
PA14
PL94
Offers
relationship
r6
r7
r8