3ISY402 – DATABASE SYSTEMS

Lecture 1 – Introduction

Material from essential text:

T CONNOLLY & C BEGG. Database Systems – A Practical Approach to Design, Implementation and Management, 4th Edition. Addison-Wesley, 2005.

Lecture - Objectives

- Introduction to the module.
- Assessments breakdown.
- Main phases in Database Design.
- Basic concepts associated with an E–R model.
- How to use E-R modelling to build a conceptual data model based on the data requirements of an organisation.
- Diagrammatic technique for displaying E-R model using Unified Modelling Language (UML) notation.
- E-R modelling for a given Case Study using MS-Visio diagramming tool.

Module - Topics

- Database Architecture.
- Relational Database Management Systems.
- Database design and implementation.
  - **Conceptual Design** (E-R Diagramming):
    - UML diagramming notation,
    - MS-Visio diagramming tool.
  - **Logical Design** (Conceptual E-R to Relational Mapping).
  - **Physical Design and Implementation**:
    - Oracle relational database,
    - SQL Developer for Data Definition (SQL DDL).
- Manipulation of the database.
    - SQL Developer for Data Manipulation (SQL DML).
- DBA, Security and Concurrency.

Module - Introduction

- Lectures – 2-hour slots per week.
- Tutorials – 2-hour slots per week.
- 2-3 hours background reading per week.
- Module information is on Blackboard.
- You may be requested to attend VIVAs.
- Assessment summary:

ASSESSMENT	WEIGHT	DETAILS
Lecture Test	(20%)	•Multiple Choice Questions.
Coursework	(30%)	•Group Coursework Report.
Examination	(50%)	•Written Paper.

Module - Introduction

- Advice regarding lectures/tutorials:
  - Catch up on any missed lectures/tutorials.
  - Print & organise lecture/tutorial notes in a folder.
  - Read lecture notes before attending the tutorials.
  - Attend all ***tutorials*** and do the work set ***individually.***
  - Complete unfinished tutorials in your own time.
  - Time-management is essential – use a diary.
  - ***Coursework*** – make sure you all ***participate equally in all parts of the work.***
  - The practical knowledge is essential to passing exam.
  - This module teaches foundation for other modules.

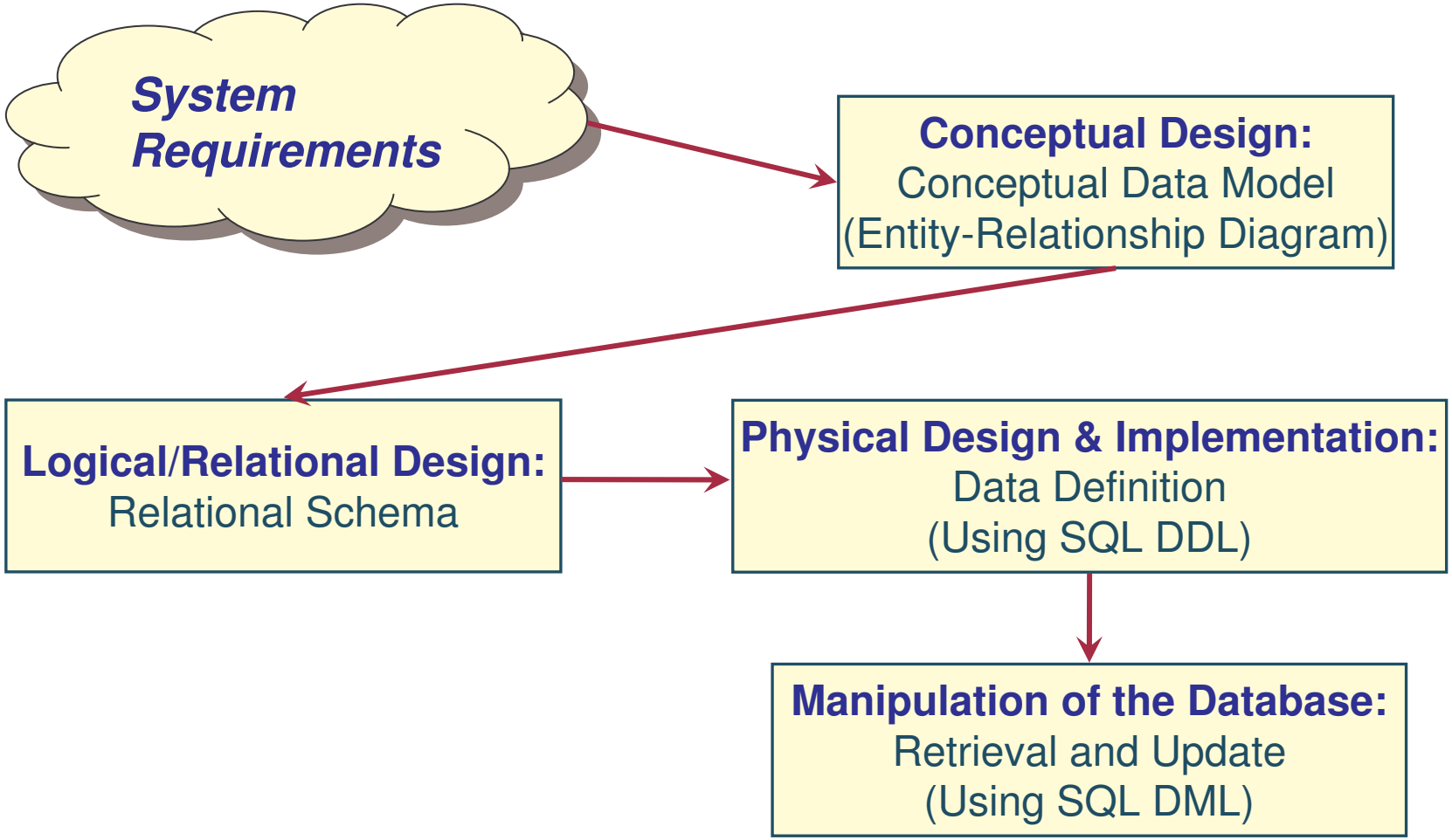
3ISY402 – DATABASE SYSTEMS

Lecture 1 – Entity-Relationship Modelling

Material from essential text:

T CONNOLLY & C BEGG. Database Systems – A Practical Approach to Design, Implementation and Management, 4th Edition. Addison-Wesley, 2005.

Relational Database design and implementation



Three main phases of Database Design

- Conceptual Database Design:
  - The process of constructing a model of the data used in an enterprise, independent of *all* physical considerations.
- Logical Database Design:
  - The process of constructing a model of the data used in an enterprise based on a specific data model (e.g. relational), but independent of a particular DBMS and other physical considerations.
- Physical Database Design:
  - The process of producing a description of the implementation of the database given a particular DBMS and other physical considerations.

Concepts of the ER Model

- Top-down approach to database design to reduce data duplication.
- The basic concepts of an ER model are
  - *entity types*,
  - *relationship types*, and
  - *attributes*.
- Start by identifying the important data (called entities) and relationships between the data.
- Then add more details such as the information we want to hold about the entities and relationships (called attributes) and any constraints on the entities, relationships, and attributes.

Conceptual Database Design

- Basic steps in building conceptual data model:
  1. Identify entity types.
  2. Identify relationship types.
  3. Identify attributes and associate them with entity or relationship types.
  4. Determine attribute domains.
  5. Determine candidate and primary key attributes.
  6. Determine degree of relationship.
  7. Determine multiplicity constraints on relationships (cardinality and participation).

Entity Type

- Entity type:
  - *Group of objects with same properties, identified by enterprise as having an independent existence.*
  - These objects may be tangible (concrete/physical) or intangible (conceptual/abstract).
  - Entity is named using a noun or nounal clause.
  - Entity name should be in the singular form.
  - Often referred to simply as *entity.*
- Entity occurrence:
  - Uniquely identifiable object or instance of an entity.

Examples of Entity Types

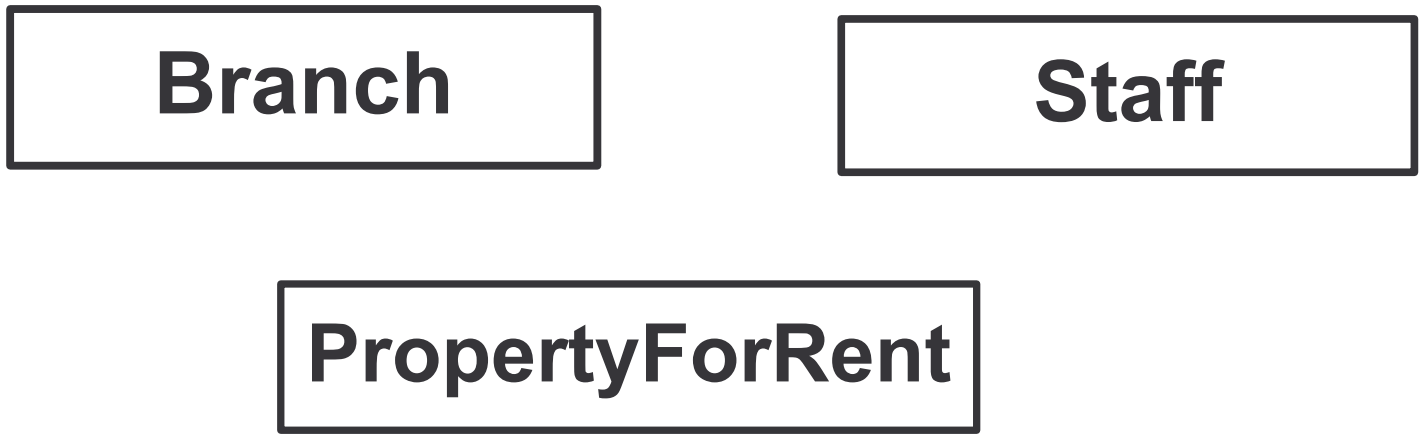
Physical existence	
Staff	Part
Property	Supplier
Customer	Product
Conceptual existence	
Viewing	Sale
Inspection	Work experience

Entity Types – Criteria

- A valid entity must have the following properties:
  1. Each occurrence should be uniquely identifiable.
  2. There must be more than one occurrence of the entity.
  3. The entity must have data (more than one attribute) that is required to be stored.
  4. It must be of direct interest to the system under consideration.

Entity Types – UML Conventions

- The first letter of each word in the entity name is uppercase and the rest of the letters are lowercase, and all words are joined together.
- E.g. - Branch, Staff, PropertyForRent, Student, Course, VideoCopy, Reservation.



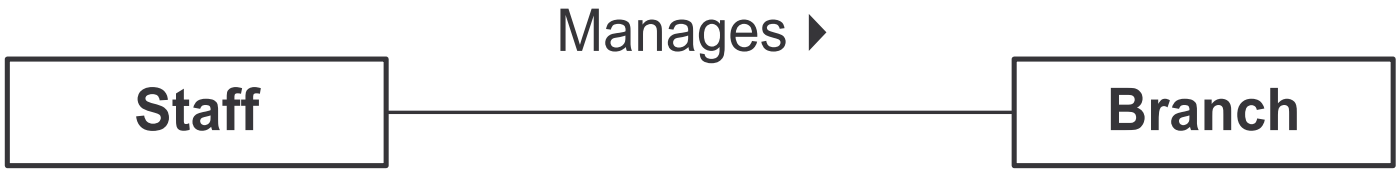
Relationship Types

- Relationship type:
  - *A set of meaningful associations among entities.*
  - The business rules that link together business information needs.
  - Named using a verb or a short verbal phrase.
  - Label the relationship name.
  - A relationship name should be unique for a given ER model.
- Relationship occurrence:
  - Each uniquely identifiable instance or association within a set.

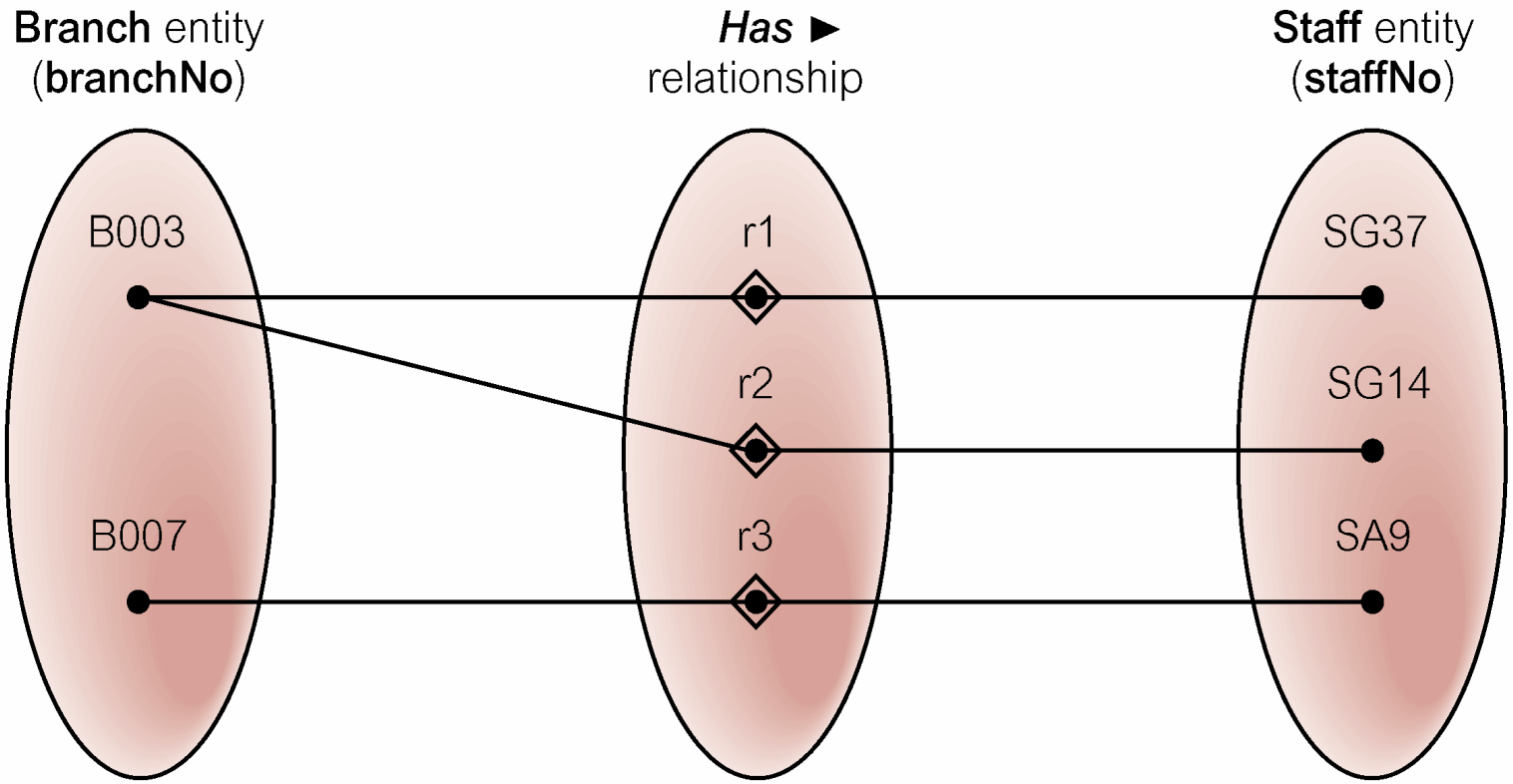
Relationship Types – UML Conventions

- Shown as a line connecting the associated entities.
- Only label the relationship in one direction (and indicate the direction using the arrow symbol).

- The first letter of each word in the relationship name is uppercase and the rest of the letters are lower-case, and all words are joined together.
- E.g. – Assigned, Has, LeasedBy, Manages.



Relationship Occurrences - Semantic net of Branch *Has* Staff relationship type



2 occurrences of entity Branch

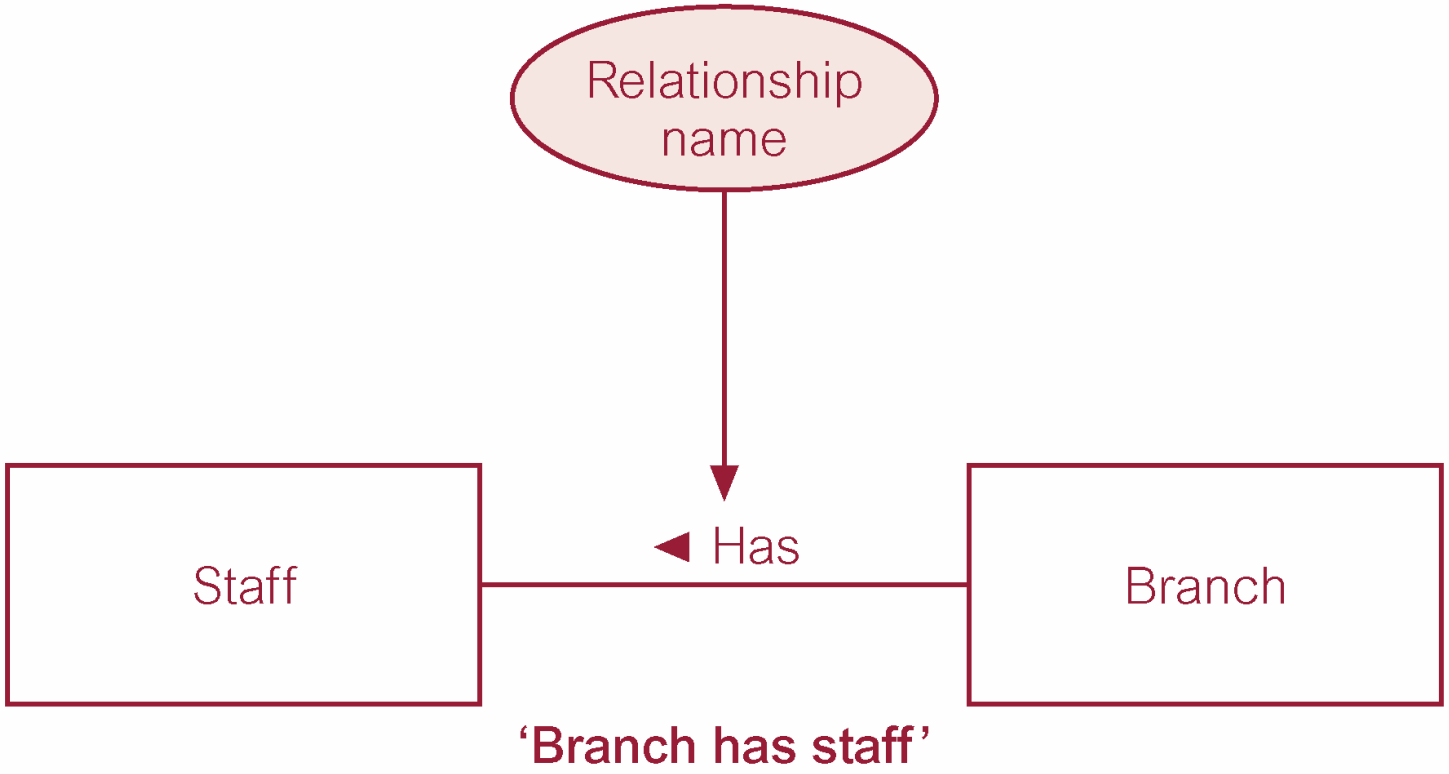
3 occurrences of relationship Has

3 occurrences of entity Staff

Relationship Types - Degree

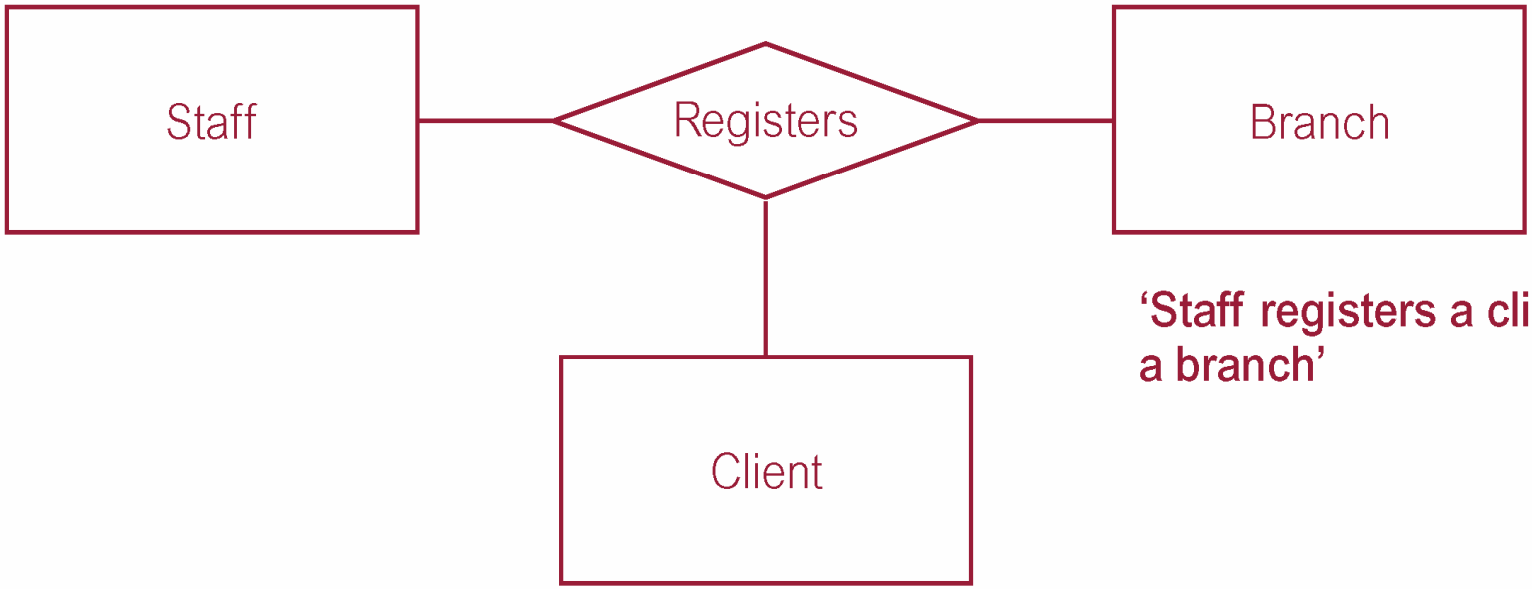
- Degree of a Relationship:
  - Number of participating entities in relationship.
- Relationship of degree:
  - One is called *unary* (or recursive relationship)
  - Two is *binary*
  - Three is *ternary*
  - Four is *quaternary*.
- The most common degree for relationships is binary.
- Complex relationships – have three or more entities.

Binary relationship - Branch *Has* Staff



'Branch has staff'

Ternary relationship called *Registers*



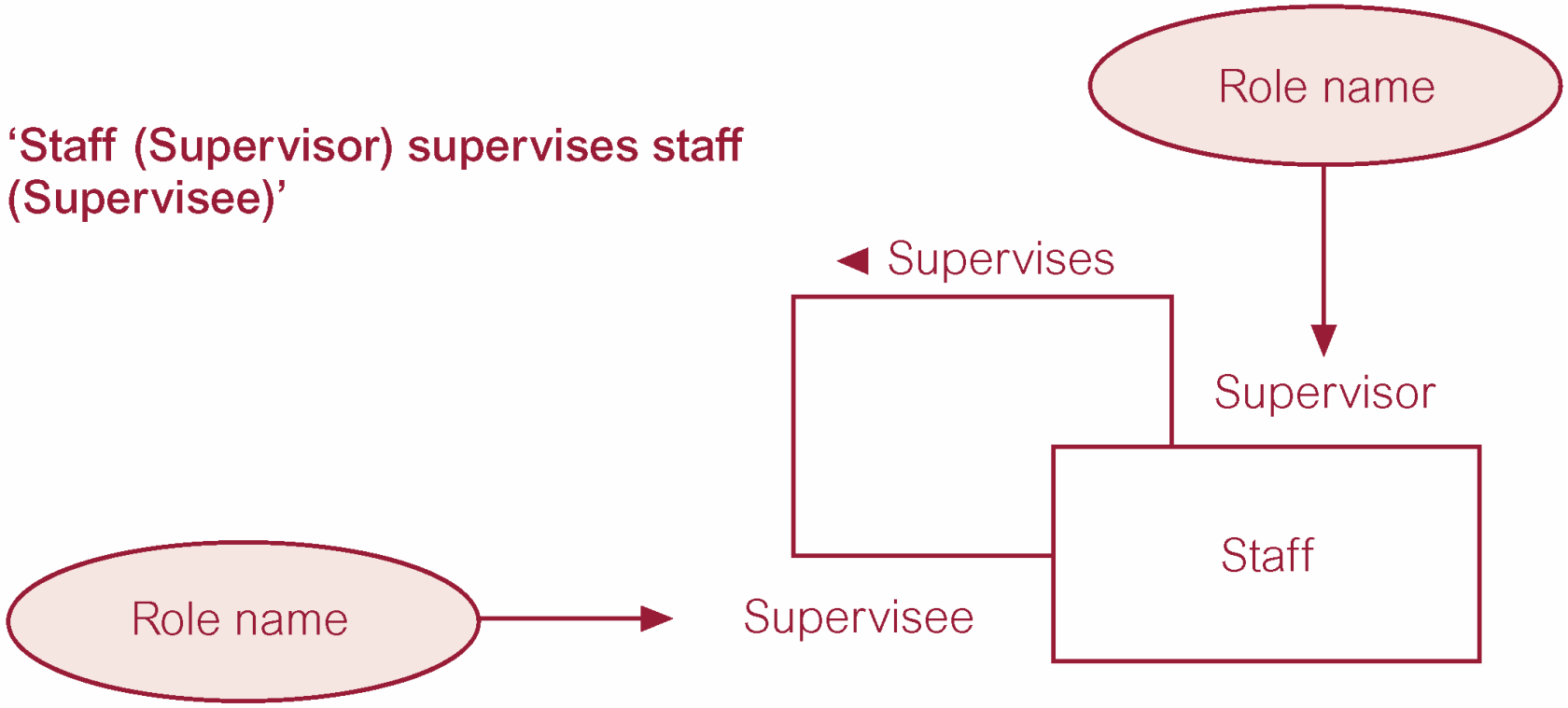
'Staff registers a client at a branch'

Recursive Relationship Types

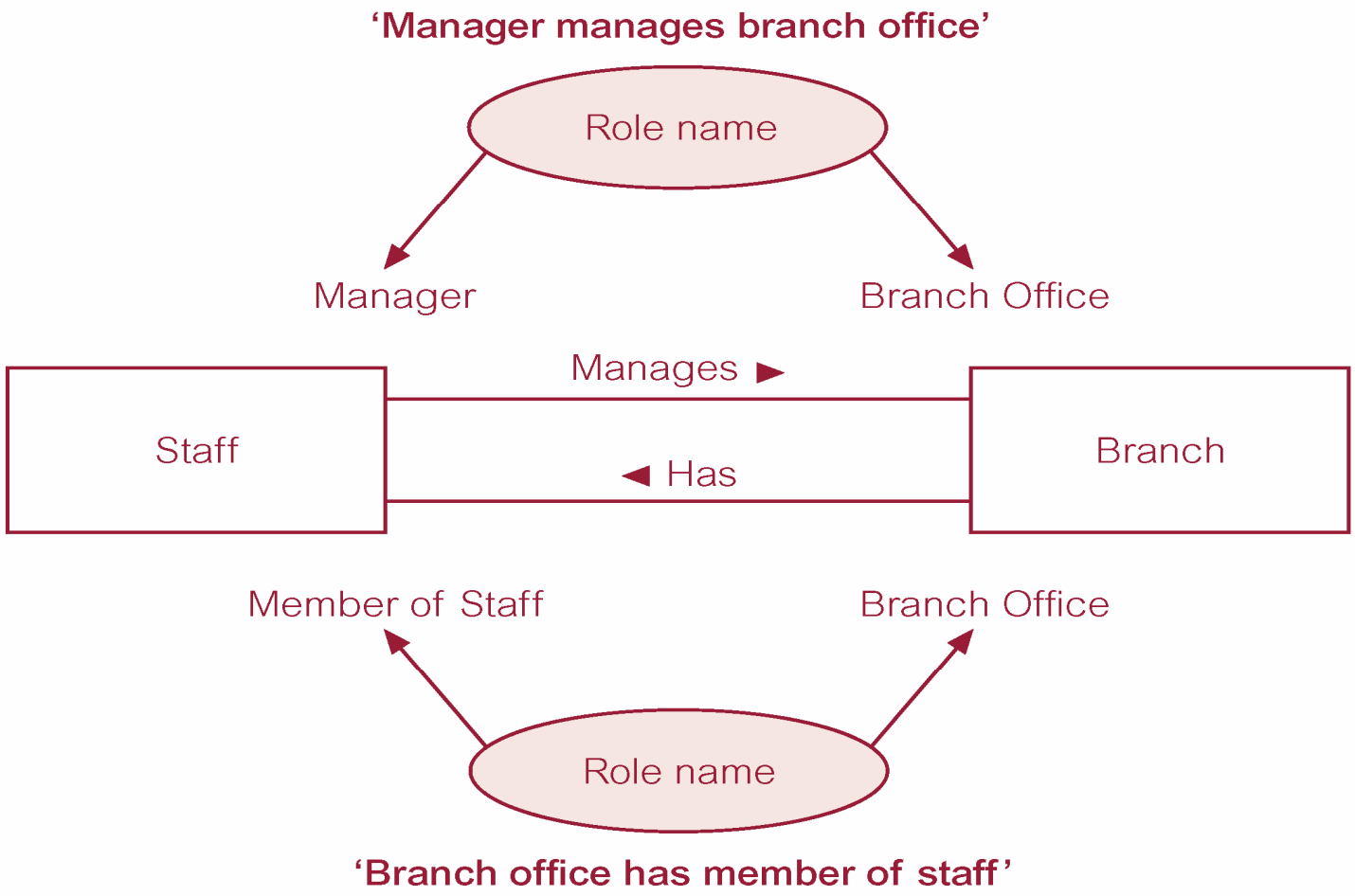
- Recursive Relationship:
  - Relationship type where the *same* entity type participates more than once in *different roles*.
- Relationships may be given role names to indicate purpose that each participating entity type plays in a relationship.

Recursive relationship called *Supervises* with role names

'Staff (Supervisor) supervises staff (Supervisee)'



Entities associated through two distinct relationships with role names

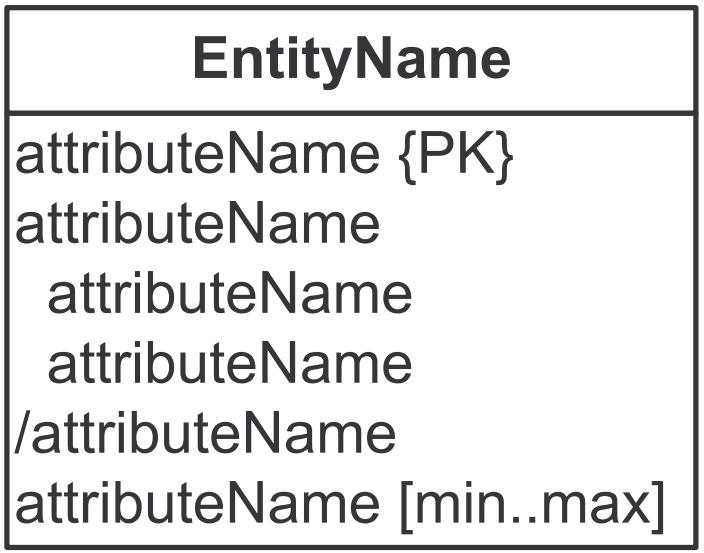


Attributes

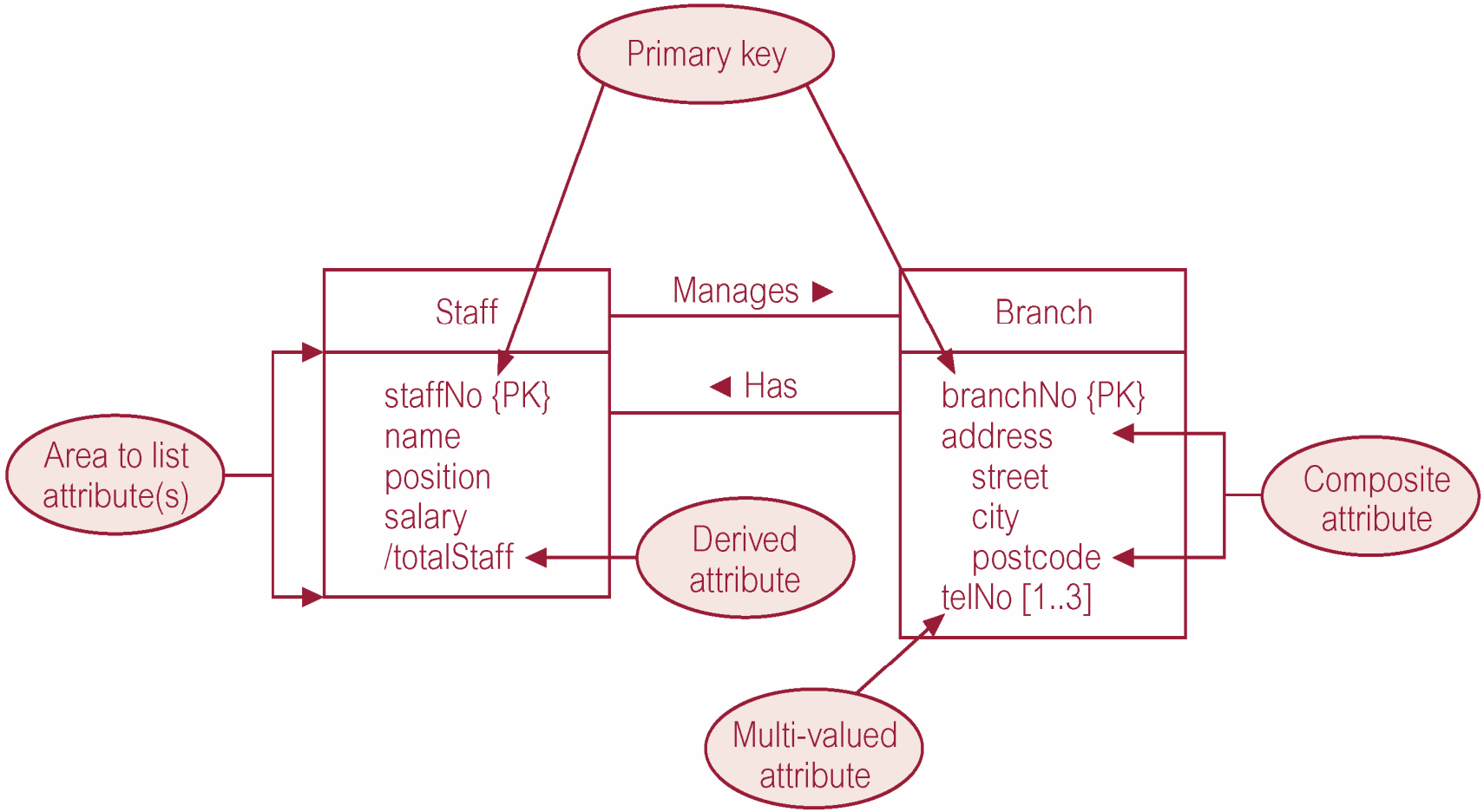
- Attribute:
  - *Property of an entity or a relationship type.*
  - E.g. - EMPLOYEE entity may have attributes name, address, date of birth, salary, gender.
- Attribute Domain:
  - Set of allowable values for one or more attributes.
- Attribute can be classified as being:
  - simple or composite,
  - single-valued or multi-valued,
  - or derived.

Attributes – UML Conventions

- If attributes for entity type are to be shown, divide the rectangle representing the entity into two parts.
- First list the primary key attribute, labeled with {PK}.
- The first letter in the attribute name is lower case and the first letter of each subsequent word is uppercase.
- E.g.
  - address, branchNo,
  - studentId, courseCode,
  - telNo, dateOfBirth.



Entity Attributes – Example ER diagram of Staff and Branch



Attributes

- Simple Attribute:
  - Attribute composed of a single component with an independent existence.
  - E.g. – name, position, salary.
- Composite Attribute:
  - Attribute composed of multiple components (simple attributes), each with an independent existence.
  - E.g. address (street, city, postcode)

Attributes

- Single-valued Attribute:
  - Attribute that holds a single value for an entity occurrence.
- Multi-valued Attribute:
  - Attribute that holds multiple values for an entity occurrence.
  - E.g. telNo [1..3] denotes up to 3 telephone numbers.
- Derived Attribute:
  - Attribute that represents a value that is derivable from the value of a related attribute, or set of attributes, not necessarily in the same entity type.
  - E.g. /totalStaff
  - Derived attributes are preceeded with a / (forward-slash).

Keys

- Candidate Key:
  - Minimal set of attributes that uniquely identifies each occurrence of an entity type.
  - Can have more than one candidate key for an entity.
- Primary Key:
  - Candidate key selected to uniquely identify each occurrence of an entity type.
  - Primary key is denoted by suffixing attribute with {PK}.
  - E.g. staffNo {PK}, branchNo {PK}.

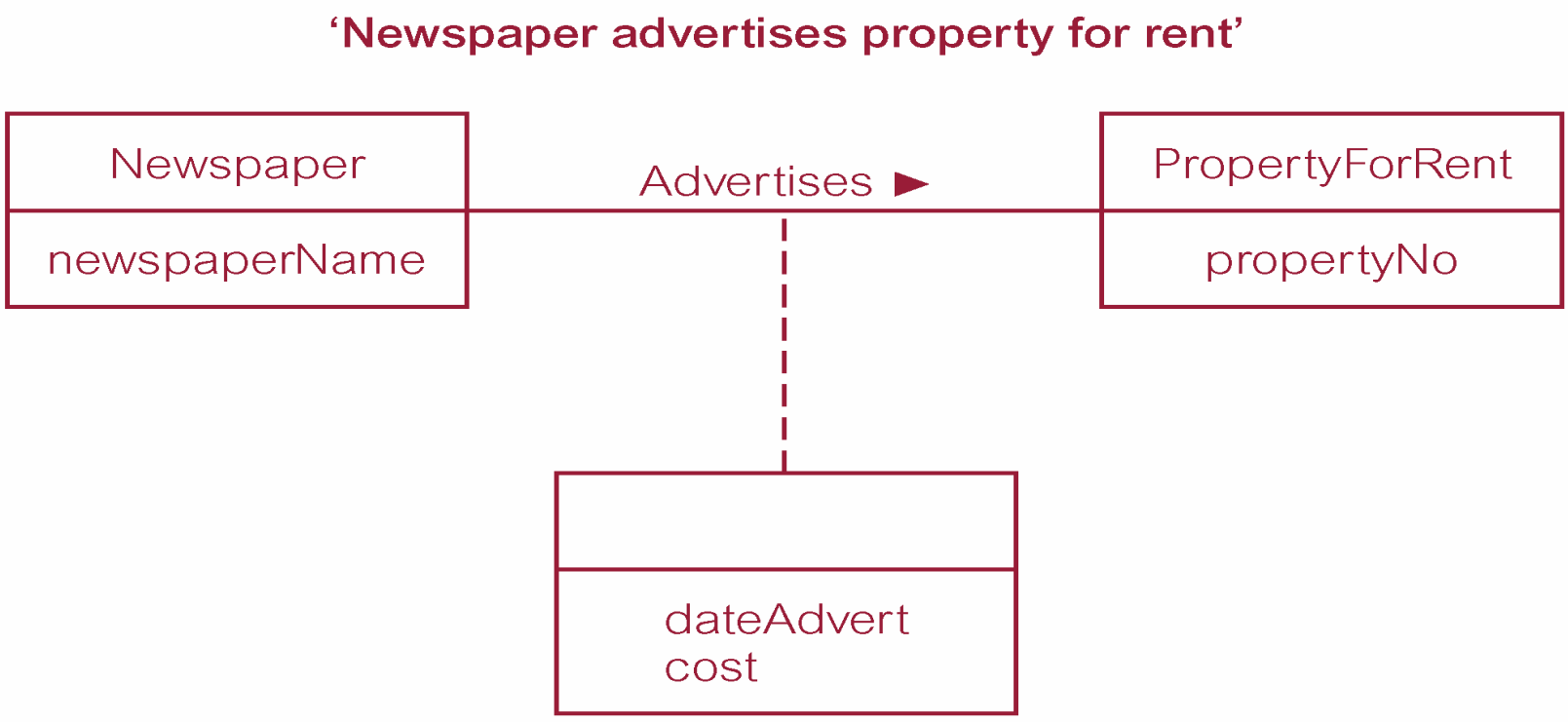
Keys

- Composite Key:
  - A candidate key that consists of two or more attributes.
  - Sometimes referred to as compound key.
- Surrogate Key:
  - A primary key which may be a generated attribute, planted into the entity, with contrived (made-up) values that bear no relationship to the real world.
  - E.g. staffNo, branchNo, studentReg, bankAccountNo.

Relationship Attributes

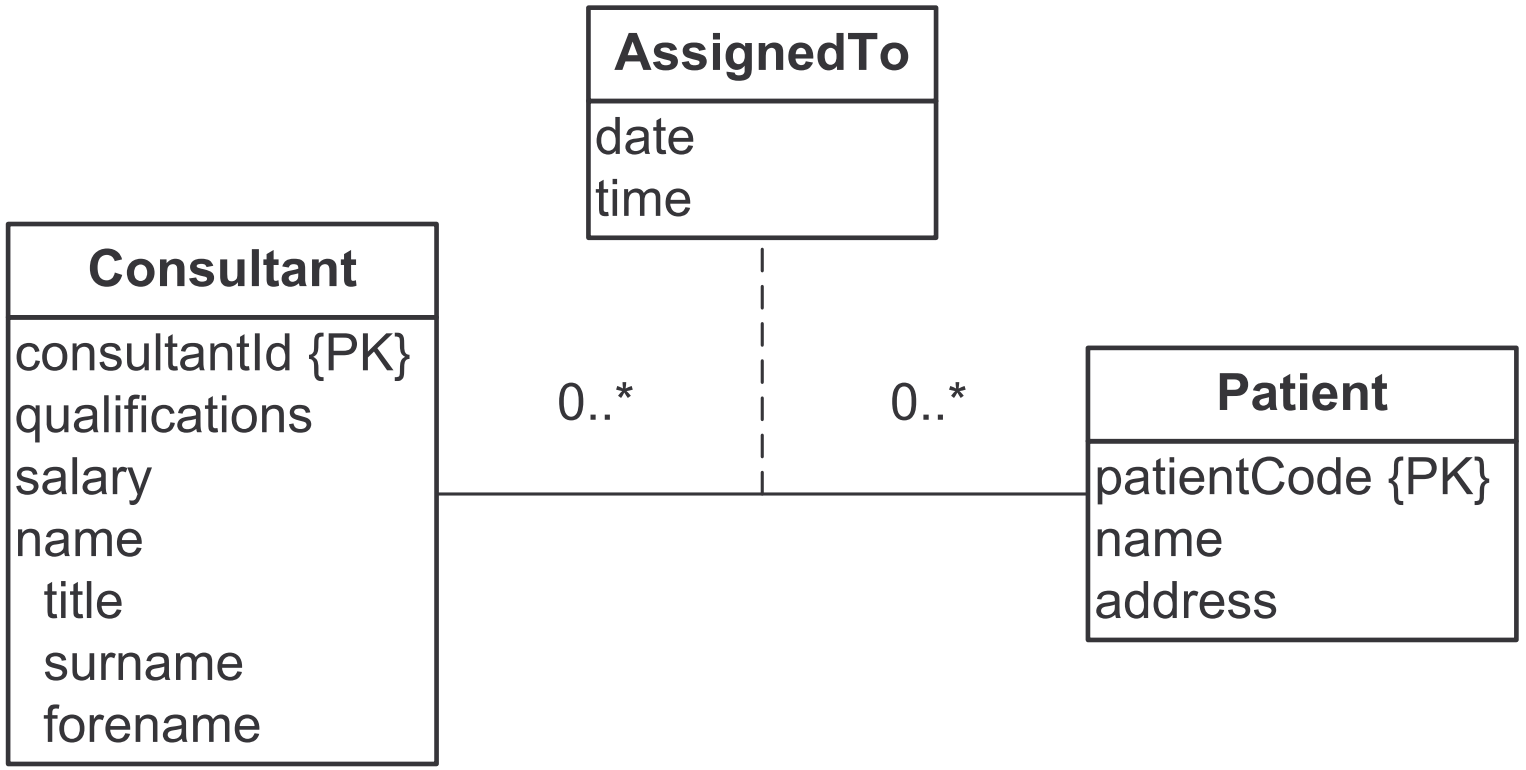
- Relationship attributes are *attributes that exist only when there is an association between entities*.
- ** In UML – relationship attributes are denoted using a dashed line.
- Example 1 – *Only when* a newspaper advertises a property for rent, will the attributes *dateAdvert* and *cost* be recorded.
- Example 2 – *Only when* a Consultant is assigned to a Patient would the attributes *date* and *time* (for the appointment) come into affect.

Relationship Attributes - Example 1



Relationship Attributes - Example 2

- ** Using Visio – relationship name is placed in the box instead of on the relationship itself.

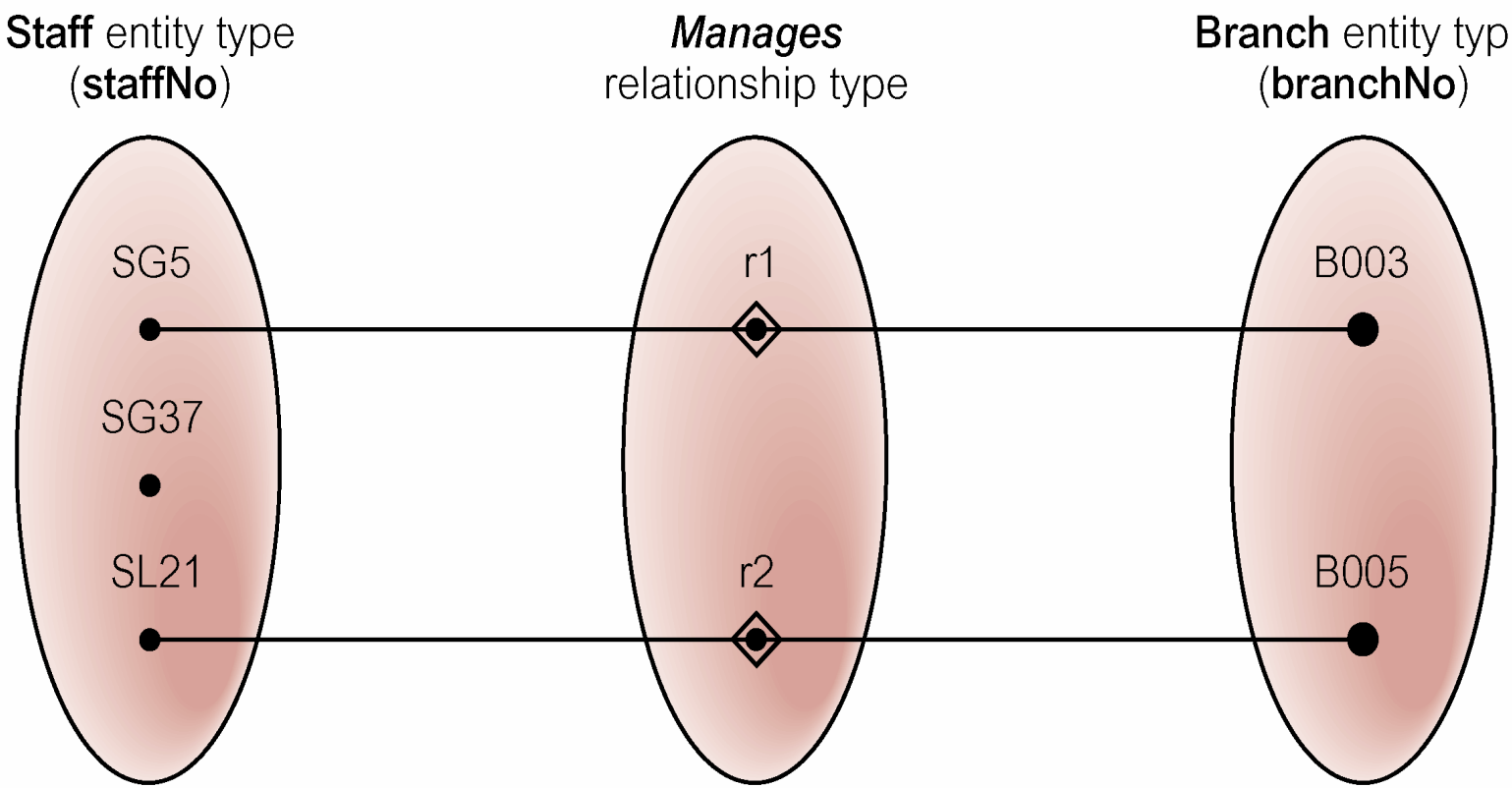


Structural Constraints

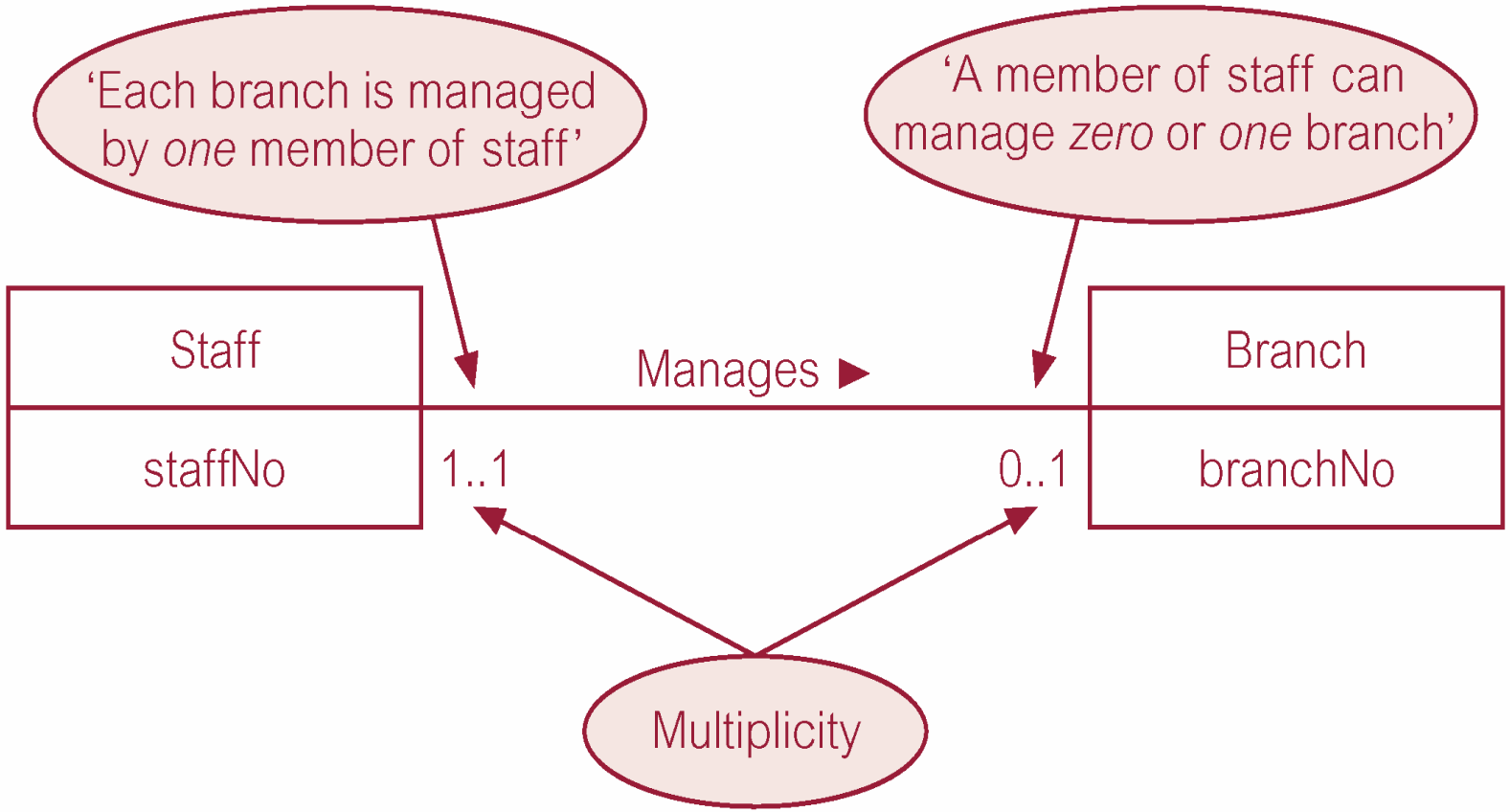
- Main type of constraint on relationships is called *multiplicity*.
- Represents company policies (called *business rules*).
- Multiplicity represents the range of possible occurrences of an entity that may relate to a single occurrence of an associated entity for a given relationship.

- Binary relationships are generally referred to as being:
  - one-to-one (1:1)
  - one-to-many (1:*)
  - many-to-many (*:*)

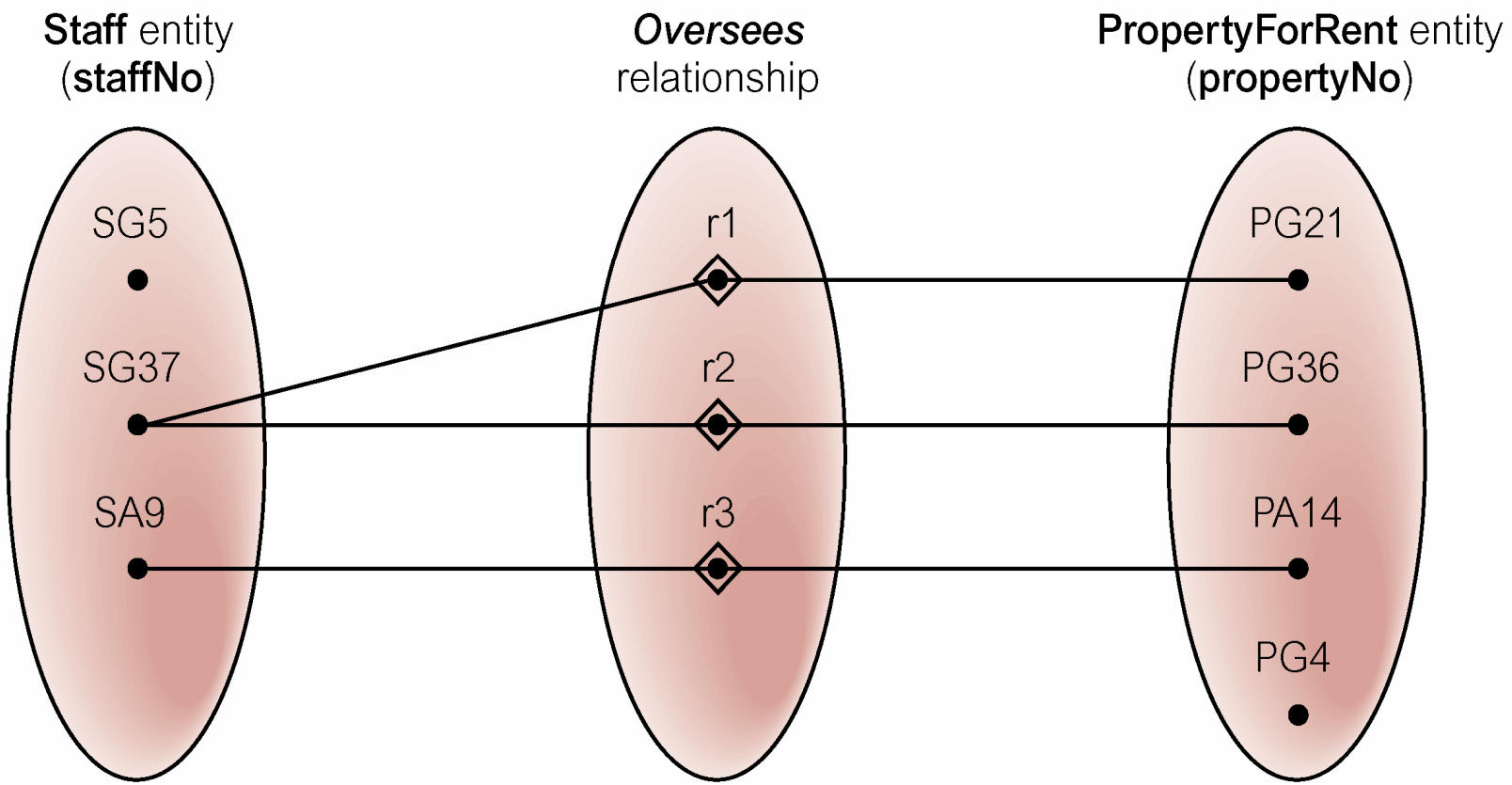
(1:1) relationship type example – Staff *Manages* Branch



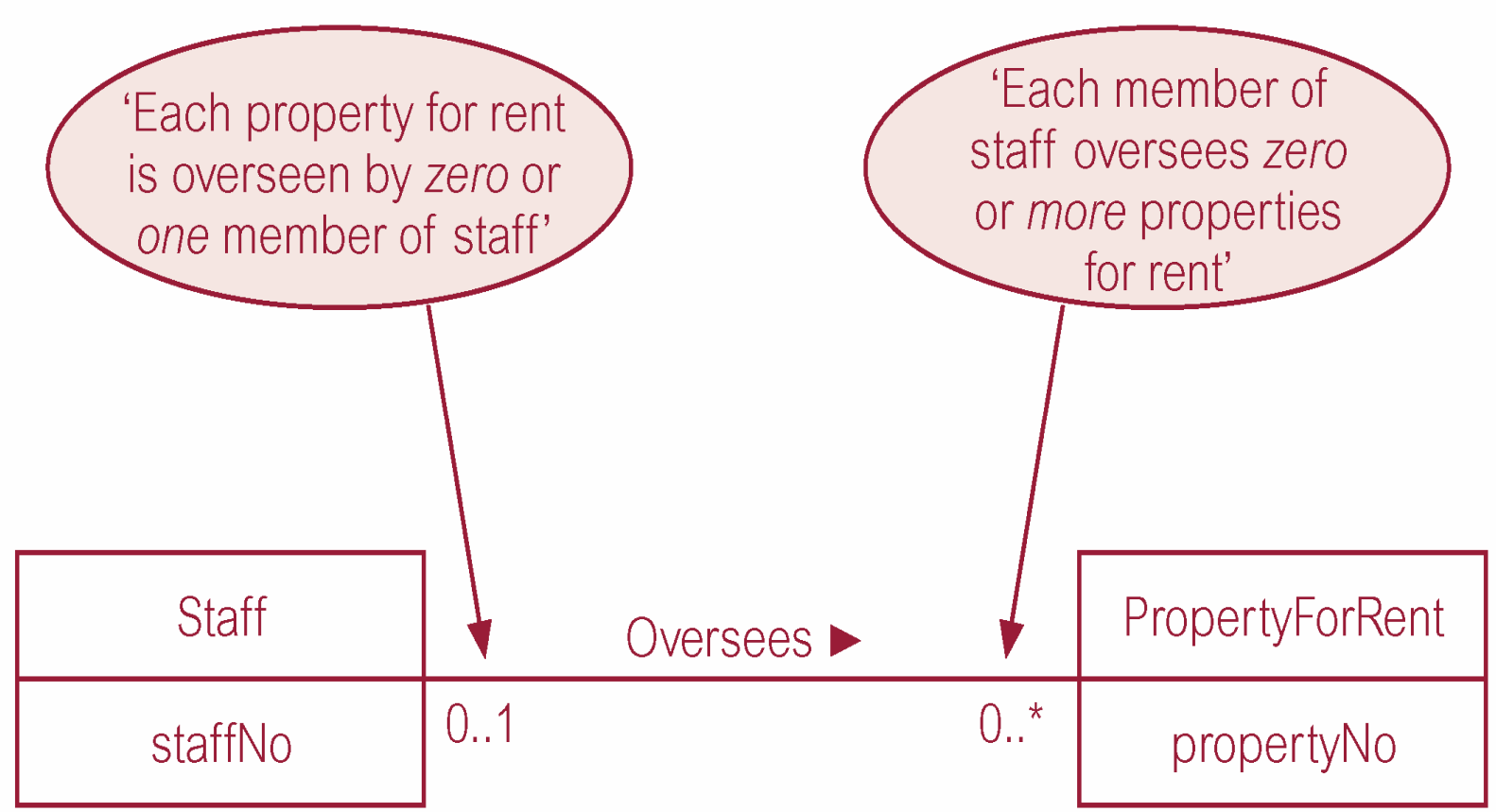
(1:1) relationship type – Multiplicity of Staff *Manages* Branch



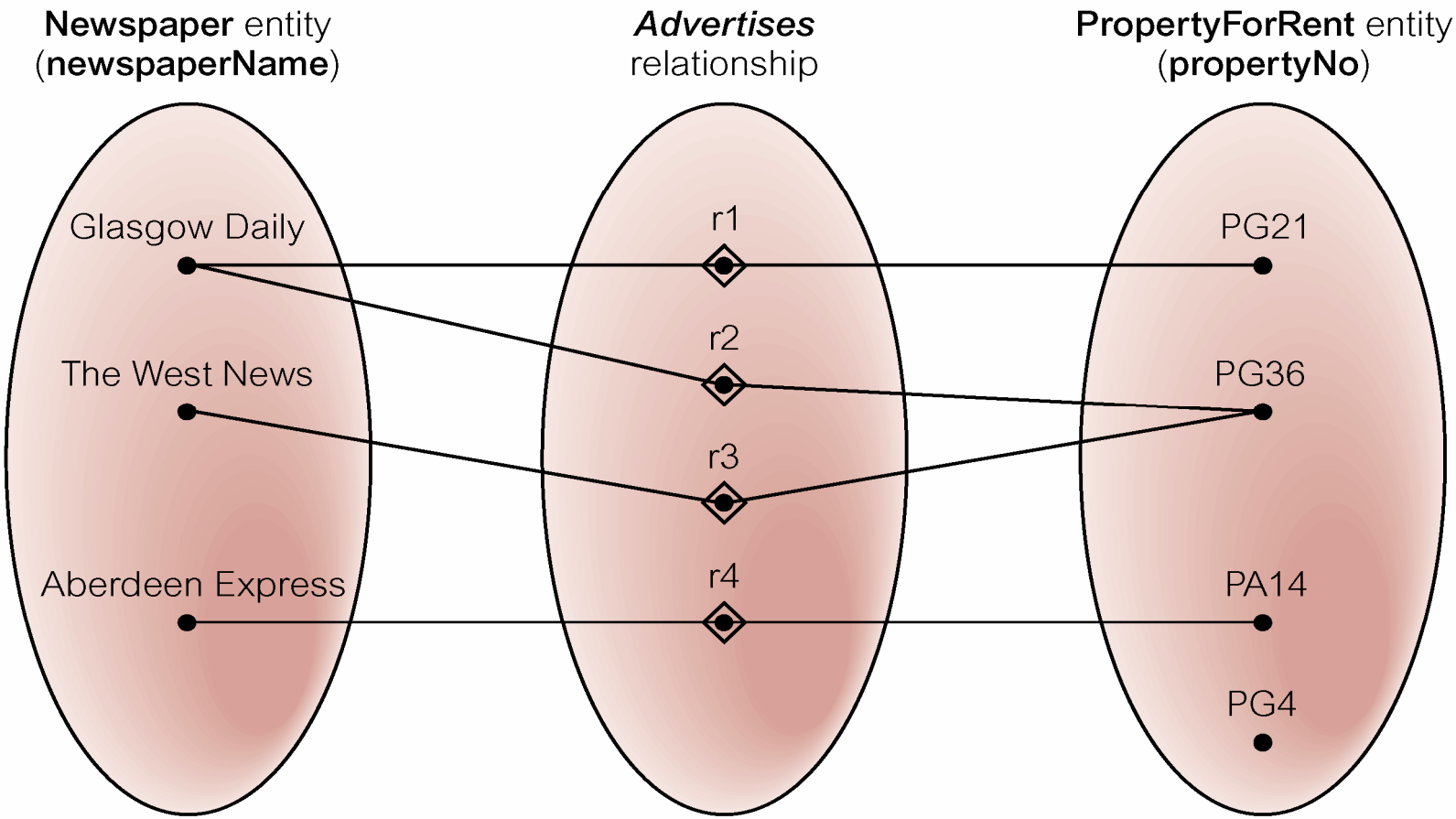
(1:*) relationship type example – Staff *Oversees* PropertyForRent



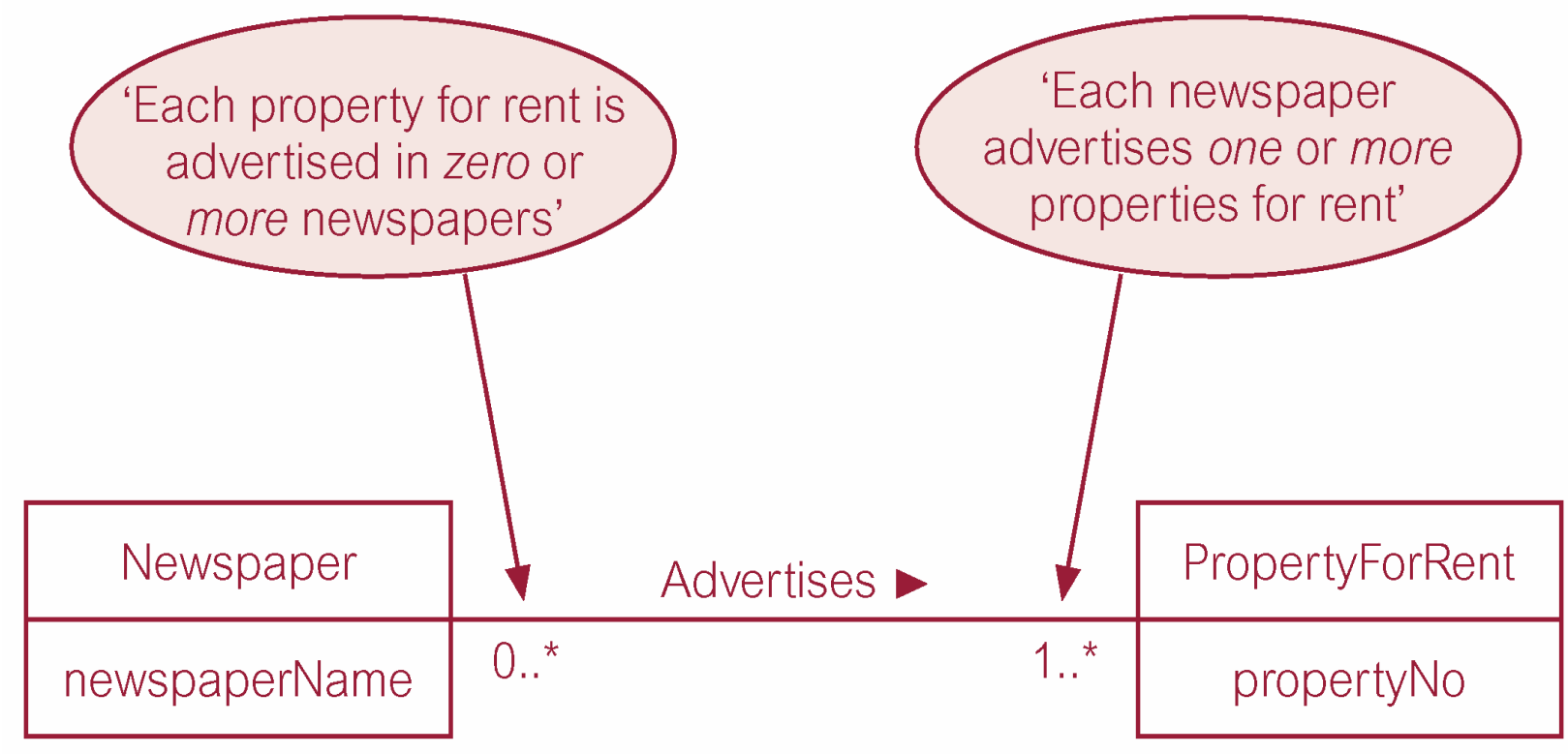
(1:*) relationship type – Multiplicity of Staff *Oversees* PropertyForRent



(*:*) relationship type example – Newspaper *Advertises* PropertyForRent



(*:*) relationship type – Multiplicity of Newspaper *Advertises* PropertyForRent



Structural Constraints - Cardinality

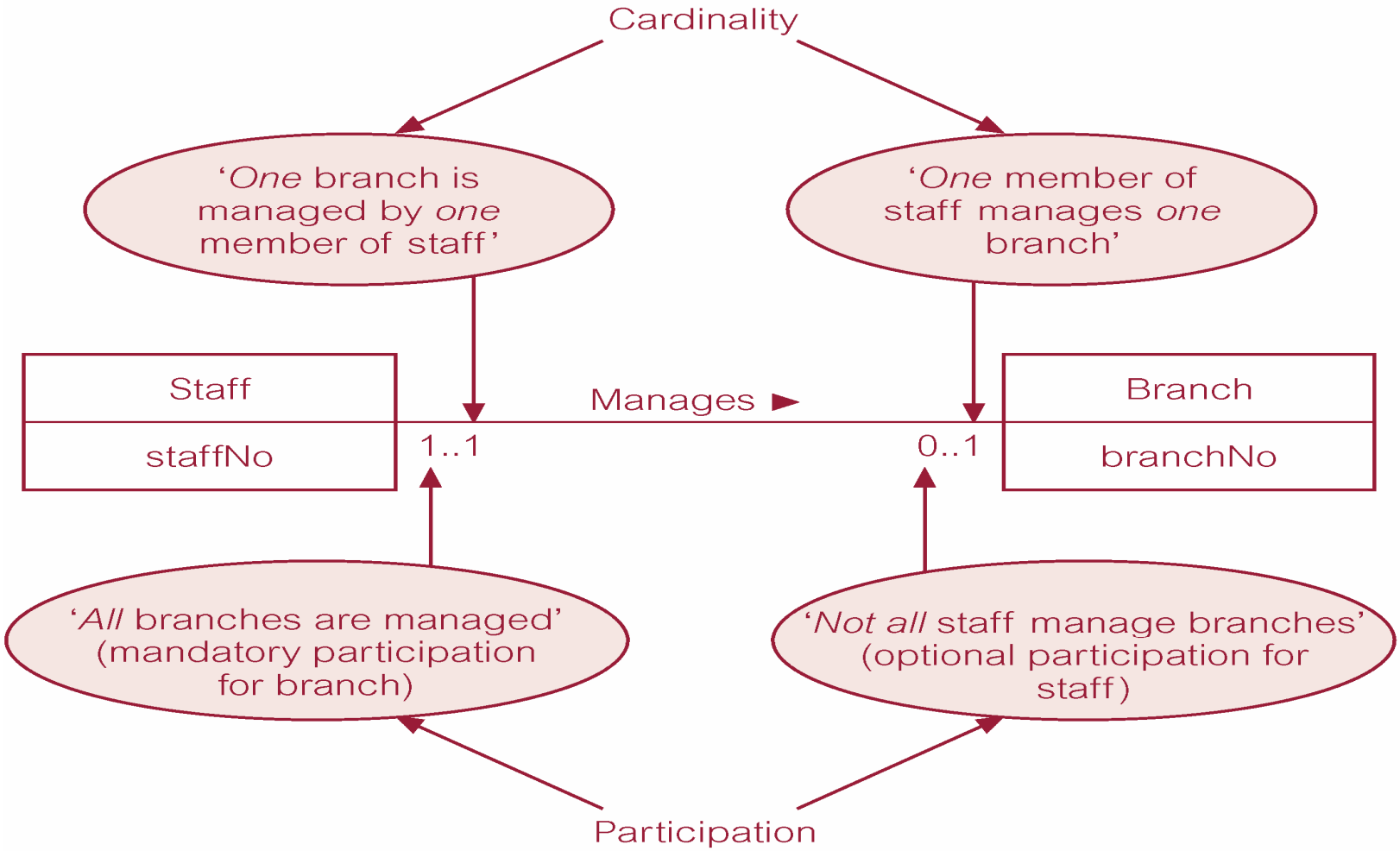
- Multiplicity is made up of two types of restrictions on relationships: *cardinality* and *participation*.
- Cardinality constraints:
  - Describes ***maximum*** number of possible relationship occurrences for an entity participating in a given relationship type.
- Participation constraints:
  - Describes ***minimum*** number of possible relationship occurrences for an entity participating in a given relationship type.

Structural Constraints - Participation

- Participation/Optionality constraints:
  - Determines whether all or only some entity occurrences participate in a relationship.
- *Optional* relationship shows that there *may* be an occurrence for its related entity.
- *Mandatory* relationship shows that for an occurrence of an entity, there *must* be an occurrence.

Participation constraint	Notation	Relationship sentence
Optional or Partial	0..?	*may*
Mandatory or Total	1..?	*must*

Multiplicity as cardinality and participation constraints



Describing Relationships - Example

- Relationships are described *bi-directionally*.
- Example of describing *Staff Manages Branch* relationship:
  - Each member of Staff *may* manage one and only one Branch.
  - Each Branch *must* be managed by one and only one member of Staff.
- OR
  - A given member of Staff *may* manage one and only one Branch.
  - A given Branch *must* be managed by one and only one member of Staff.

Summary of multiplicity constraints in ER Modelling using UML

- UML notation uses a single relationship line, and *participation..cardinality* are labelled at both ends.
- When describing relationships, the **participation..cardinality** are read at the *object* end.

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.

Terminology used in E-R Modelling and OO Class Diagrams using UML Notations

- Use MS Visio for ER Diagrams.
- Use Rational Rose case tool for OO Class Diagrams.

ER Diagram	OO Class Diagram
Entity	Class
Attribute	Attribute
Relationship	Association
Occurrence	Instance
Relationship Attributes	Class Association

Lecture Case Study 1 - ERD

- Lecture Case Study 1 - *Gudelf and Wellbean Health Authority Database System.*
- Tasks:
  - Identify the entities.
  - Identify relationships.
  - Identify groups of attributes for entities and relationships.
  - Identify primary keys for each entity.
  - You may NOT add additional attributes.
  - Describe each relationship.

References and Further Reading:

- T CONNOLLY & C BEGG. *Database Systems – A Practical Approach to Design, Implementation and Management,* 4th Edition. Addison-Wesley, 2005.
  - Chapter 11 – Entity-Relationship Modelling.
- T CONNOLLY & C BEGG. *Database Solutions – A step-by-step guide to building databases,* 2nd Edition. Addison-Wesley, 2004.
  - Chapter on Entity-Relationship Modelling.