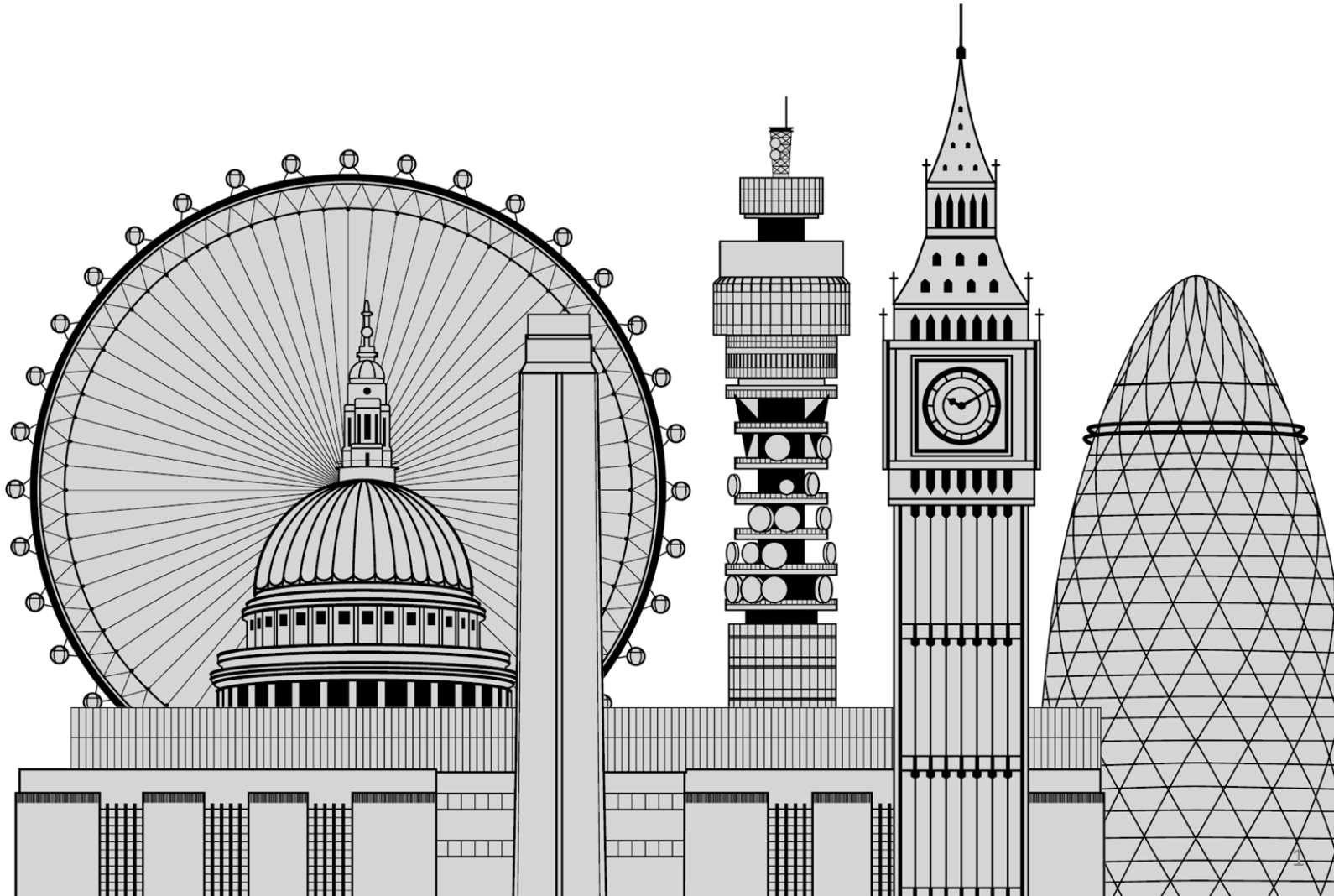
# 5COSC020W DATABASE SYSTEMS – LECTURE 01

## Database Approach & Conceptual Database Design – Entity-Relationship Modelling

UNIVERSITY OF WESTMINSTER

Dr Francois ROUBERT
F.Roubert@westminster.ac.uk



# Lecture 01 – Outline

- **Database Systems**
  - Databases, Database Management Systems, Database Applications.

- **Data Models.**
  - Relational Model, XML Model, NoSQL Model, Object Model.

- **Database Design**
  - Conceptual Design, Logical Design and Physical Design.

- **Conceptual Design for Relational Databases**
  - Conceptual Entity-Relationship Modelling and ERDs.
  - Entities, relationships, multiplicity of a relationship: participation & cardinality.
  - Attributes, single & composite attributes, single-valued multi-valued, attributes.
  - Candidate keys, primary keys, alternate keys, compound keys, composite keys.

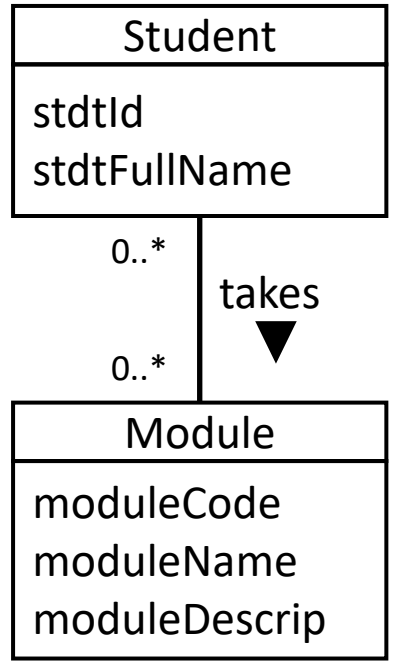
# Database Systems

- **Database System = DB + DBMS + Database Applications**

- **DB: Database**
  - Collection of logically related data for business purpose
  - e.g. University Database

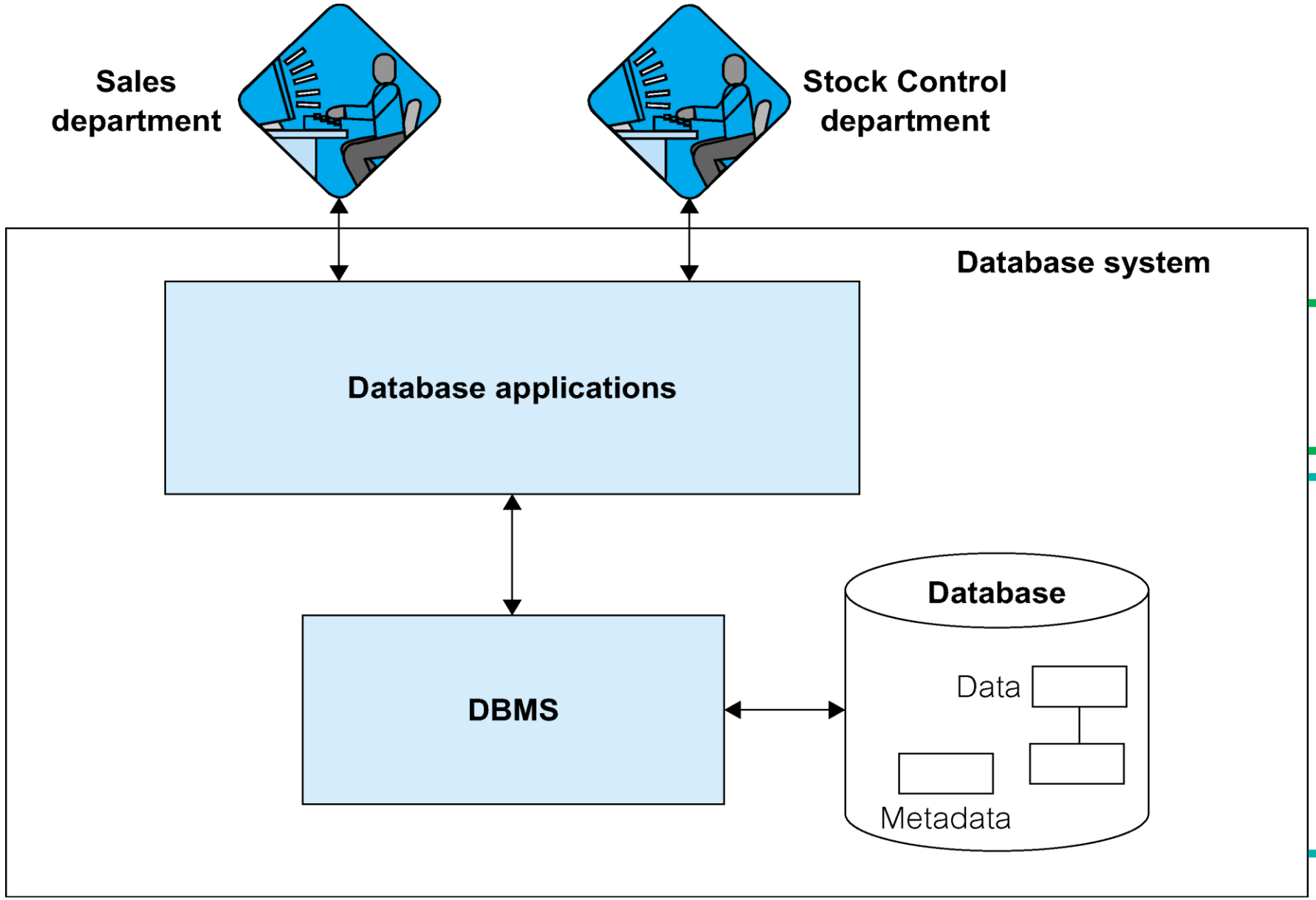
- **DBMS: Database Management System**
  - IT system to create & manage DB and control access to DB
  - e.g. Oracle, MySQL, MS SQL Server, MongoDB

- **Database Application**
  - Software that interacts with DBMS by sending requests to DBMS
  - e.g. student record system, VLE, library system, online timetabling, online retailer, medical appointment system, ticketing system




| stdtId | stdFN |
|---|---|
| w12345 | Jim Bim |
| w12346 | Tim Lim |
| w12347 | Kim Lin |

# Database System




5COSC024W
Server-side
Web Development

**5COSC020W**
**DATABASE**
**SYSTEMS**

# Database (DB)

- Logically coherent collection of related data with inherent meaning.
- Designed, built & populated to meet information needs of an organisation and to allow users to operate business transactions.
- Abstraction of the real world i.e. representation of some aspect of the real world that is able to rapidly reflect changes.
- Self-describing collection of integrated records: it stores data and metadata i.e. a description of this data

# Database Management System (DBMS)

Software that allows users to:

- **Define a database** *Data Definition Language (DDL)*
  - Specify data types, structures & constraints of the data
  - Specify the meta-data: description of the data
- **Construct a database** *Data Manipulation Language (DML)*
  - Store data in storage medium controlled by DBMS
- **Manipulate a database** *Data Manipulation Language (DML)*
  - Querying, updating and reporting on the data
- **Control access to a database** *Data Control Language (DCL)*
  - Ensuring security, integrity, concurrency & recovery

# Main characteristics of the database approach

- Self-Describing Nature of a Database System
  - Relational databases: separation of data and meta-data stored in data dictionary.
  - NoSQL databases: self-describing data with data items & values in 1 structure.

- Program Data Independence & Program Operation Independence
  - Relational databases: separation of structure of data and application program.
  - O.O. and object-relational databases: separation of implementation of an operation (method) from interface (includes operation names & parameters).

- Data Model enables Data Abstraction
  - Conceptual representation of data to describe structure of the database.
  - It hides details of data storage and of implementation of operations from users.
  - Collection of concepts such as entities (or objects), their properties (attributes) and relationships to capture the data architecture.

# Categories of Data Models

- **Conceptual Data Models**
  - Concepts close to how users perceive data, independent from IT considerations.
  - e.g. entities, attributes and relationships.

- **Representational Data Models**
  - Concepts understood by users but not too far from how data organised in computer.
  - e.g. tables, columns, primary and foreign keys.

- **Physical Data Models** (not covered in this module)
  - Concepts that describe how data is organised in computer storage.
  - e.g. record formats, record ordering and access paths.

- **Self-describing Data Models**
  - Combine description of data and data values in one structure.
  - e.g. XML, NoSQL databases.

# Classifying DBMSs based on Data Models

- **Relational Model or SQL Model** (covered in lectures 01-07)
  - Structured Data as collection of tables with fields, records, PKs and FKs.
  - Uses high level Structured Query Language (SQL) for CRUD operations.

- **XML Model** (covered in lectures 08-10)
  - Semi-structured data as elements that can be nested to create tree structures.
  - Standard for structuring and exchanging data over the Web.

- **NoSQL Model** (covered in lectures 11)
  - Key-value data model: each data value associated to unique key for fast retrieval.
  - Graph data model: data represented as graphs i.e. labelled nodes and edges.
  - Document data model: data as collection of documents in JSON or XML.

- **Object Model** (not covered in this module)
  - Data as collection of objects (in classes) with properties & operations.

# Database Design: producing a Data Model

Client — IDEA

Archi-tect — DATA MODEL

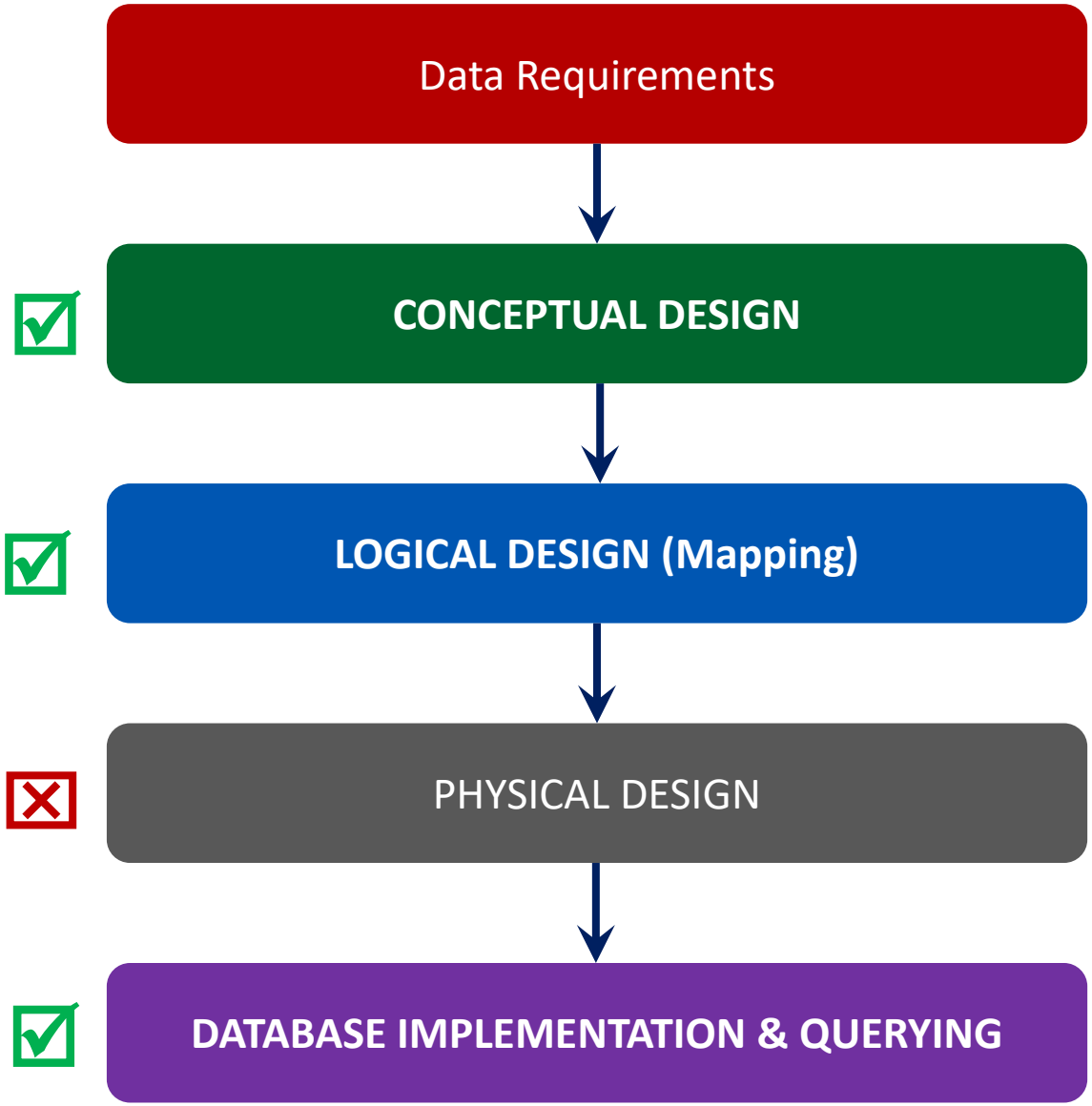
Deve-loper — SYSTEM

Database Analysis

Database Design

Database Implementation

# Phases and outputs of Database Design




**BUSINESS USER'S DATA NEEDS**
to support business processes

**CONCEPTUAL DATA MODEL OR CONCEPTUAL SCHEMA**
of the data required by business users

**LOGICAL DATA MODEL OR LOGICAL SCHEMA**
based on specific file data organisation (e.g. relational model)

**PHYSICAL OR INTERNAL SCHEMA**
storage structures, file organizations, indexes, access paths

**DATABASE IMPLEMENTED & QUERIED IN DBMS**
data structures, constraints, data values, data description

# Design for Relational Databases: E.R. Modelling

**Entity-Relationship Modelling:** top-down design approach to DB design

- Representation of the data architecture to understand:
  - The nature of the data.
  - How data is stored and used by business.

- Both for conceptual and logical design
  - Conceptual: entities, attributes, relationships, PKs.
  - Logical: relations (or tables), attributes (or columns), relationships, PKs and FKs.

- Outcome: Entities-Relationship Diagrams (ERDs) or Schemas
  - Non technical and unambiguous.
  - Flexible enough to be understandable by both users and IT technologists.
  - Not tied to any technology or business methodology.

# Step-by-step approach to Conceptual Design

## I. Identity ENTITIES

- Objects or things with independent existence on which data needs to be stored.

## II. Identify RELATIONSHIPS

- Meaningful associations between occurrences of entities.

## III. Identify ATTRIBUTES

- Properties of Entities (and sometimes of Relationships) that capture data values.

## IV. Identify PRIMARY KEYS (PKS)

- Attribute that is a unique identifier, irreducible and selected.

## V. Consider enhanced modelling concepts e.g. specialisations/generalisations

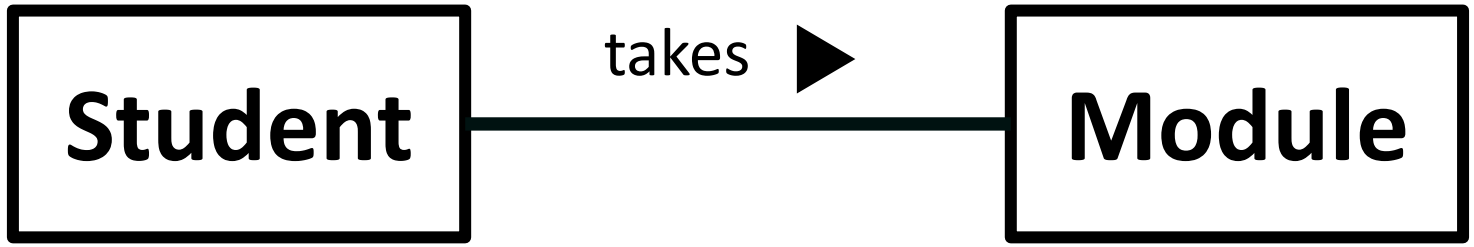
## VI. Check and remove redundancies

# Entities


- **Entity**
  - Building block of conceptual ERD.
  - Essential “business thing” of key importance for the firm to store data about.
  - Group of items with same properties that exists independently from each other.

- **Entity Occurrence**
  - Uniquely identifiable instance of an entity
  - Example: w12345 Jim Bim, w12346 Tim Lim, w12347 Kim Lin
  - Example: 5COSC020W Database Systems, 5COSC024W Server-Side Web Dev

# Relationships



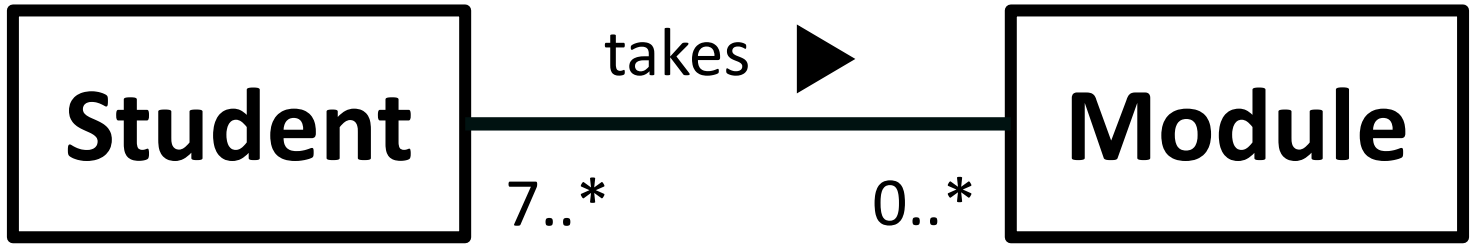

- **Relationship**
  - Meaningful association between occurrences of entities.
  - It is conceptually essential for the business but it does need to have a physical existence.

- **Notations**
  - Verb to indicate meaningful name.
  - Reading direction ▸ just to help understanding the meaning, not an arrow!
  - Can be inverted with different verb, if meaning makes more sense.

# Multiplicities


- **Multiplicity**
  - Number of possible occurrences of an entity related to a single occurrence of the other entity.
  - Represents the business rules or policies of the business.
- **Multiplicity = Participation + Cardinality**
  - **Participation**: for one occurrence of an entity, minimum number of occurrences of other entities it is related to.
  - **Cardinality**: for one occurrence of an entity, maximum number of occurrences of other entities it is related to.

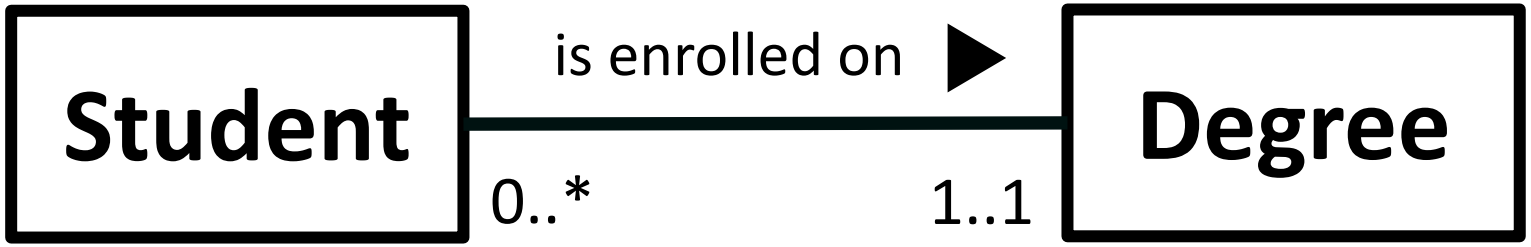
# Example of multiplicities (1)




- **ONE Student may not be taking ANY Modules (Participation is 0)**
  - It is possible for a student not to have selected modules yet or to be suspended.
- **ONE Student may be taking MANY Modules (Cardinality is *)**
  - A student may take many modules e.g. 6 modules a year for 3 years.

- **ONE Module must have AT LEAST 7 Students (Participation is 7)**
  - A module only exists if at least 7 students are registered on it: University rule.
- **ONE Module can have MANY Students on it (Cardinality is *)**
  - It makes sense for a module to have multiple students taking it.

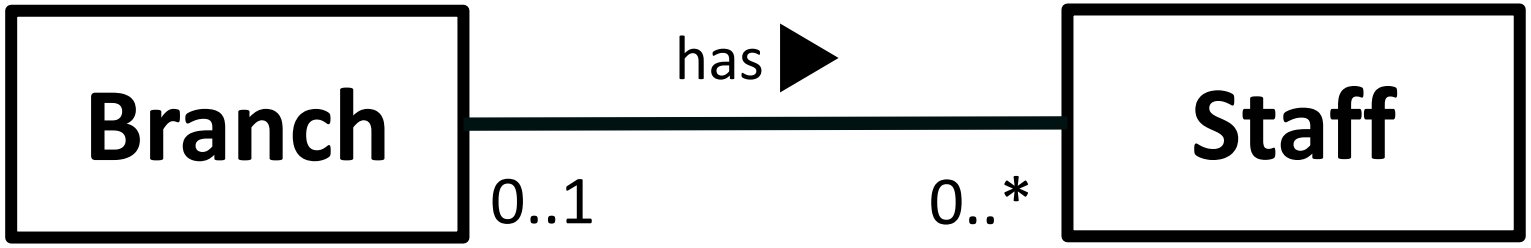
# Example of multiplicities (2)




- **ONE Student must be enrolled on AT LEAST ONE Degree (Participation is 1)**
  - A student must be signed on at least a programme to be considered a student.
- **ONE Student can be enrolled on AT MOST ONE Degree (Cardinality is 1)**
  - It is not possible for a student to be enrolled on more than one Degree.

- **ONE Degree may have NO students enrolled on it (Participation is 0)**
  - A degree may be brand new and has not recruited yet or may be currently frozen.
- **ONE Degree can have MANY students enrolled on it (Cardinality is *)**
  - It makes sense for a degree to have multiple students enrolled on it.

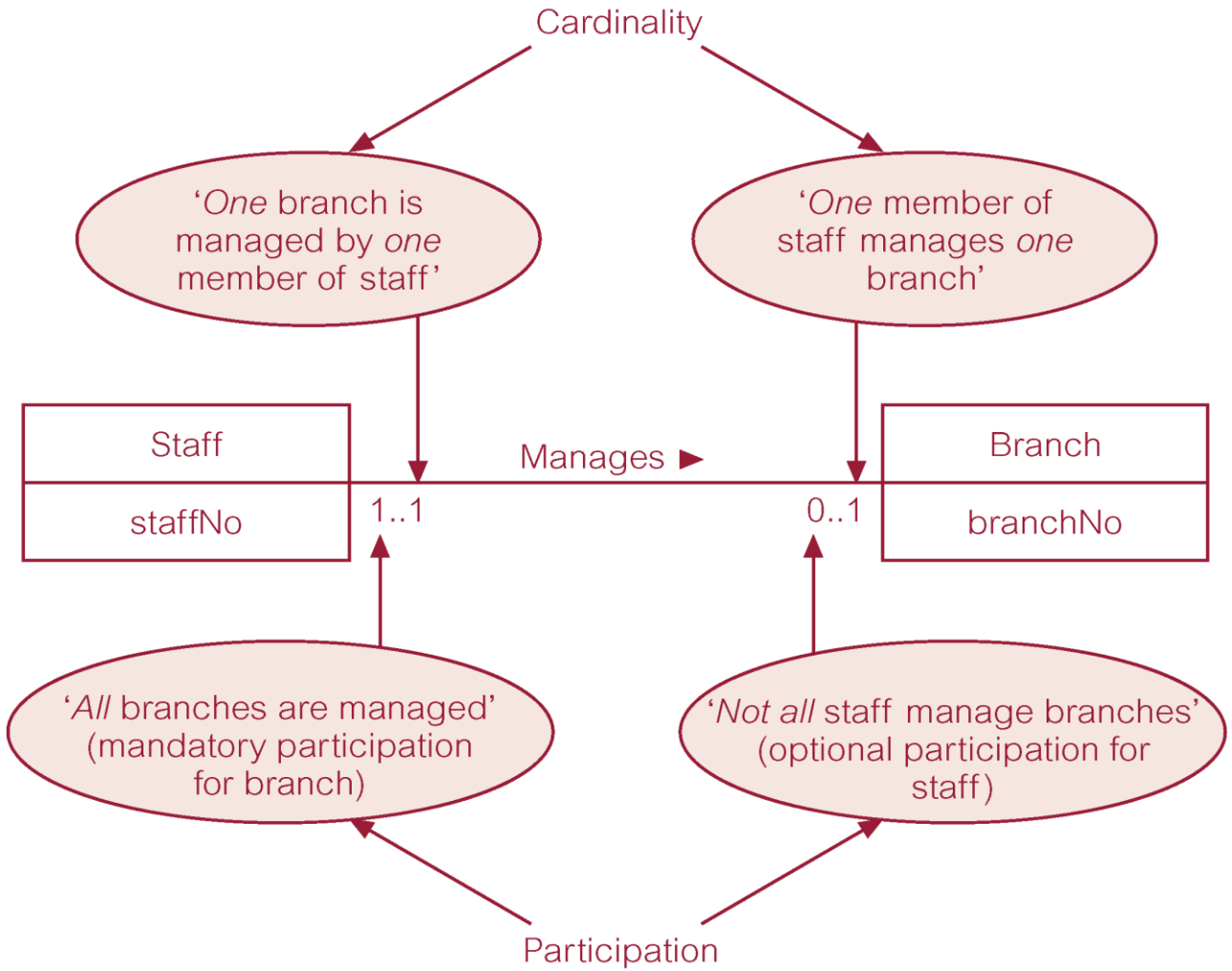
# Example of multiplicities (3)



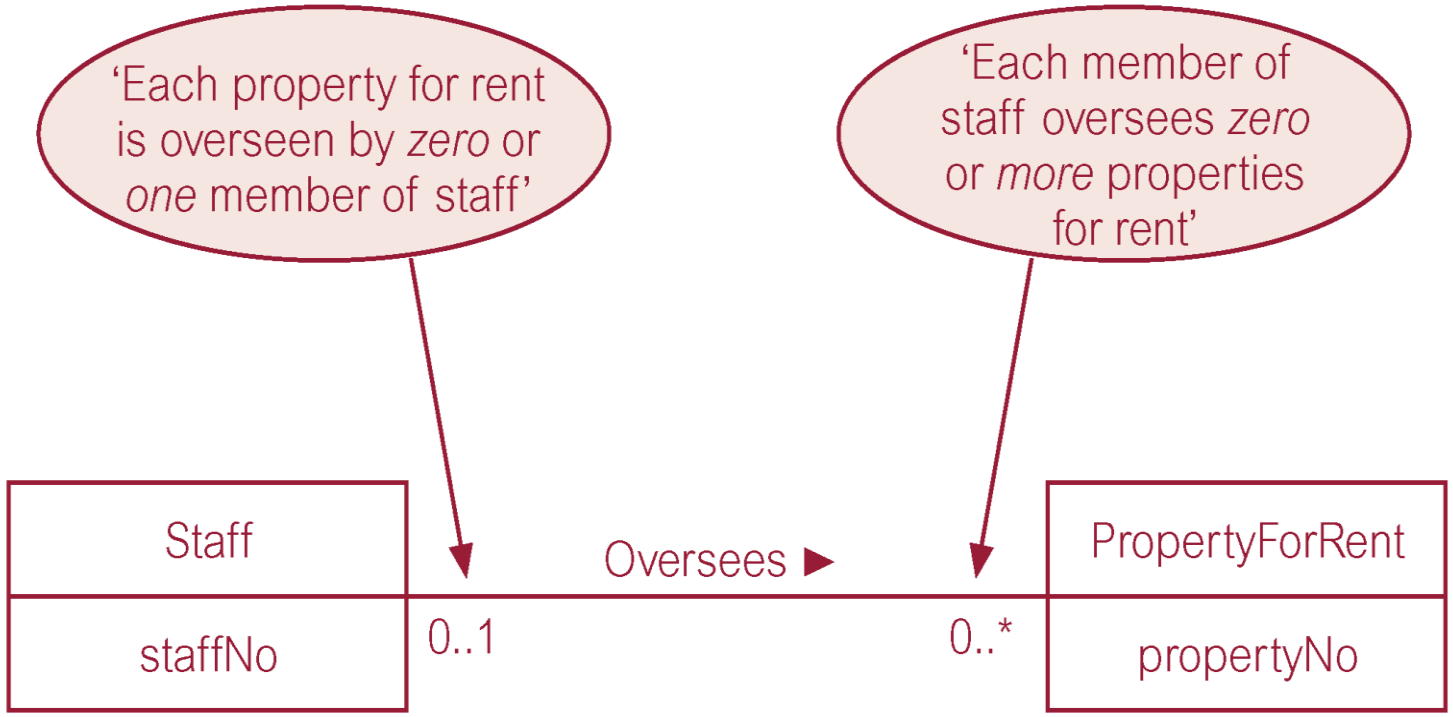

- **ONE Branch may have NO Staff assigned to it** **(Participation is 0)**
  - A branch may be brand new (not staffed yet) or underperforming (cleared out).
- **ONE Branch may have MANY Staff assigned to it** **(Cardinality is *)**
  - A branch will typically have many staff allocated to work at that specific branch.

- **ONE Staff may NOT BE ASSIGNED TO ANY Branch** **(Participation is 0)**
  - A staff may be working outside the branch system e.g. Director, Driver, IT support.
- **ONE Staff must be assigned to UP TO ONE Branch** **(Cardinality is 1)**
  - The structure of the firm says that staff can only be given one branch, no more.

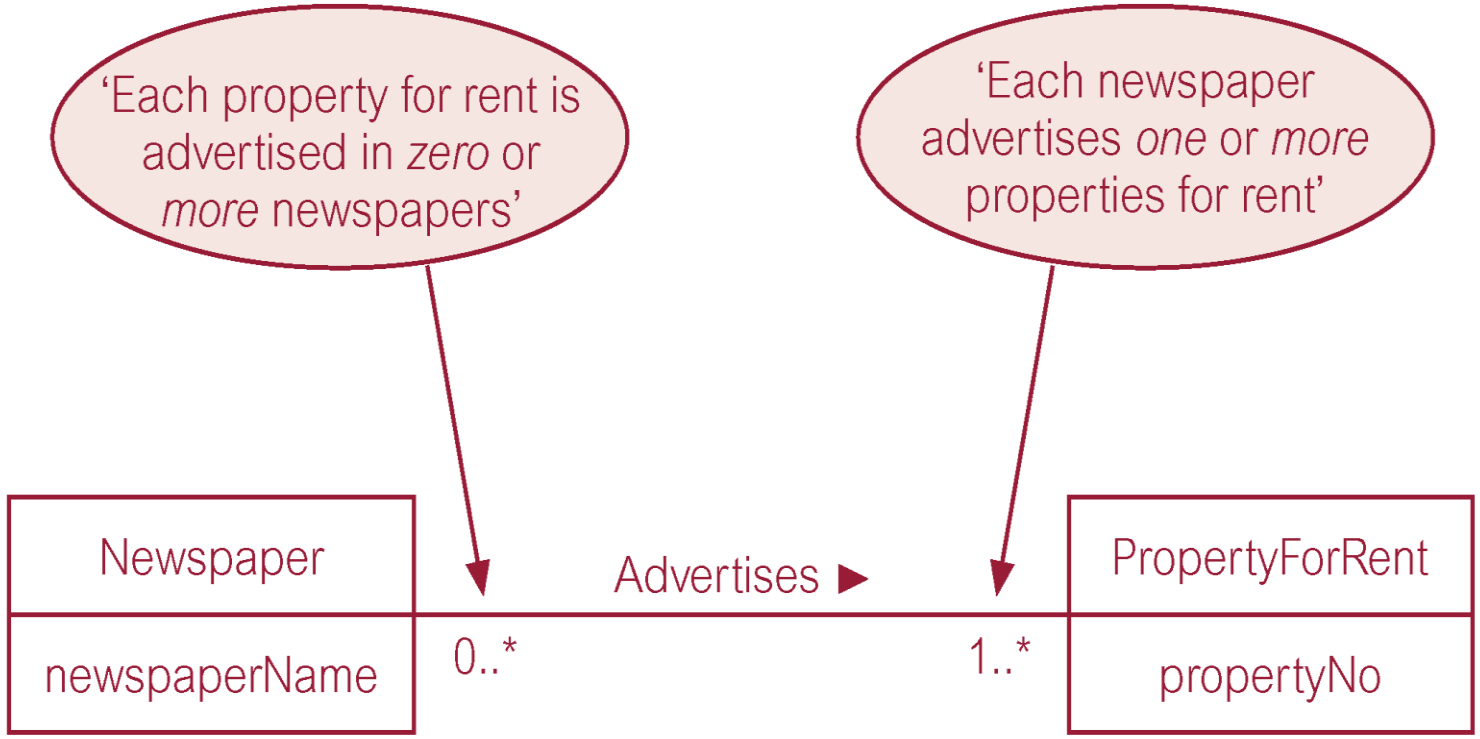
# Example of multiplicities (4)



# Example of multiplicities (5)



# Example of multiplicities (6)



# Attributes

- **Attribute**
  - Property of an entity (or a relationship) that captures data value e.g. studentId, dob
- **Attribute Domain**
  - Set of allowable values for one or more attributes
  - e.g. dob: date in range [01/01/1900 – 01/10/2021]
- **Simple Attribute**
  - Attribute composed of a single component with an independent existence.
  - e.g. studentId: w1234567
- **Composite Attribute**
  - Attribute composed of multiple components, each with independent existence.
  - e.g. studentAddress : (2 Lovely Road, London, NW2 6TD)

# More types of attributes

- **Single-valued Attribute**
  - Attribute that holds a single value for each occurrence of an entity.
  - e.g. telNo: 02079115000

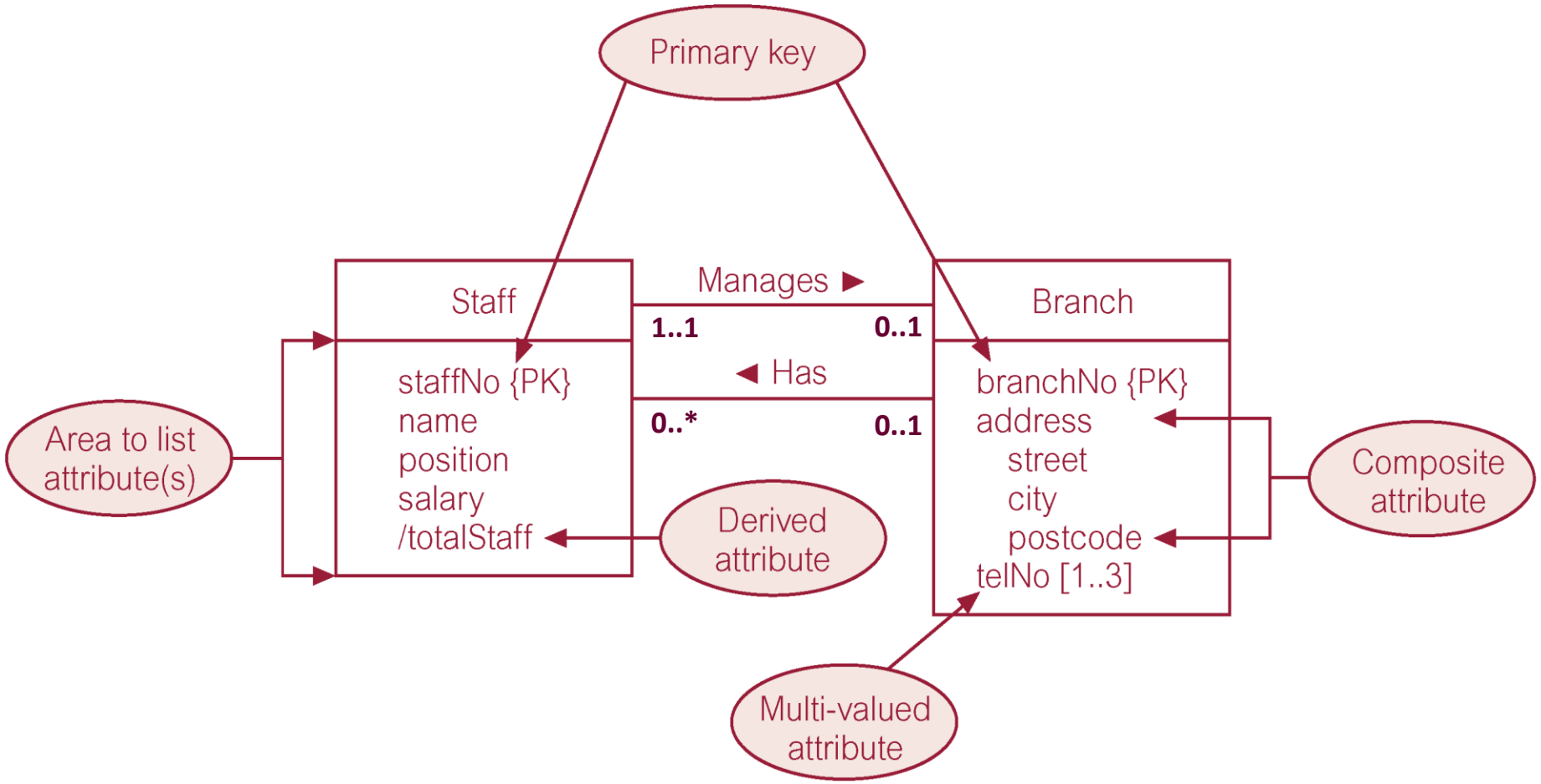
- **Multi-valued Attribute**
  - Attribute that holds multiple values for each occurrence of an entity.
  - e.g. telNo: (02079115000, 07912345678)

- **Derived Attribute**
  - Attribute that represents a value that is derivable from value of a related attribute, or set of attributes, not necessarily in the same entity.
  - Denoted with a forward slash
  - e.g. monthlySal: 4000, yearlyBonus: 2000 and /fullYearlySal: 50000

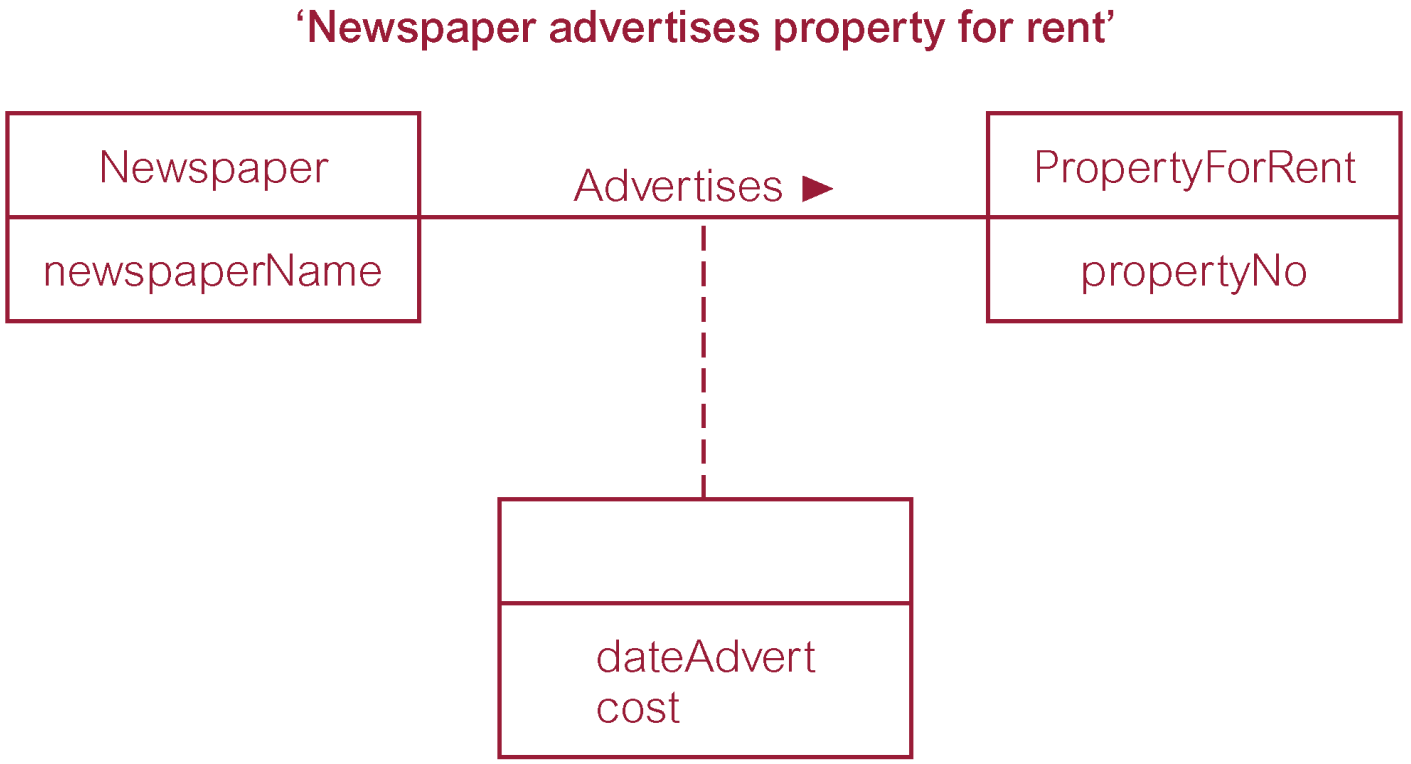
# Conceptual Keys

- **Superkey**
  - Set of attributes that uniquely identifies each occurrence of an entity.
- **Candidate Key**
  - Minimal set of attributes that uniquely identifies each occurrence of an entity.
- **PRIMARY KEY**
  - Selected minimal set of attributes that uniquely identifies each occurrence of an entity.
- **Alternate Key**
  - Not selected minimal set of attributes that uniquely identifies each occurrence of an entity.
- **Compound Key**
  - Candidate key made of at least 2 attributes with each being a simple key in its own right.
- **Composite Key**
  - Candidate key made of at least 2 attributes with at least 1 attribute not being a simple key.

# Visual Recap

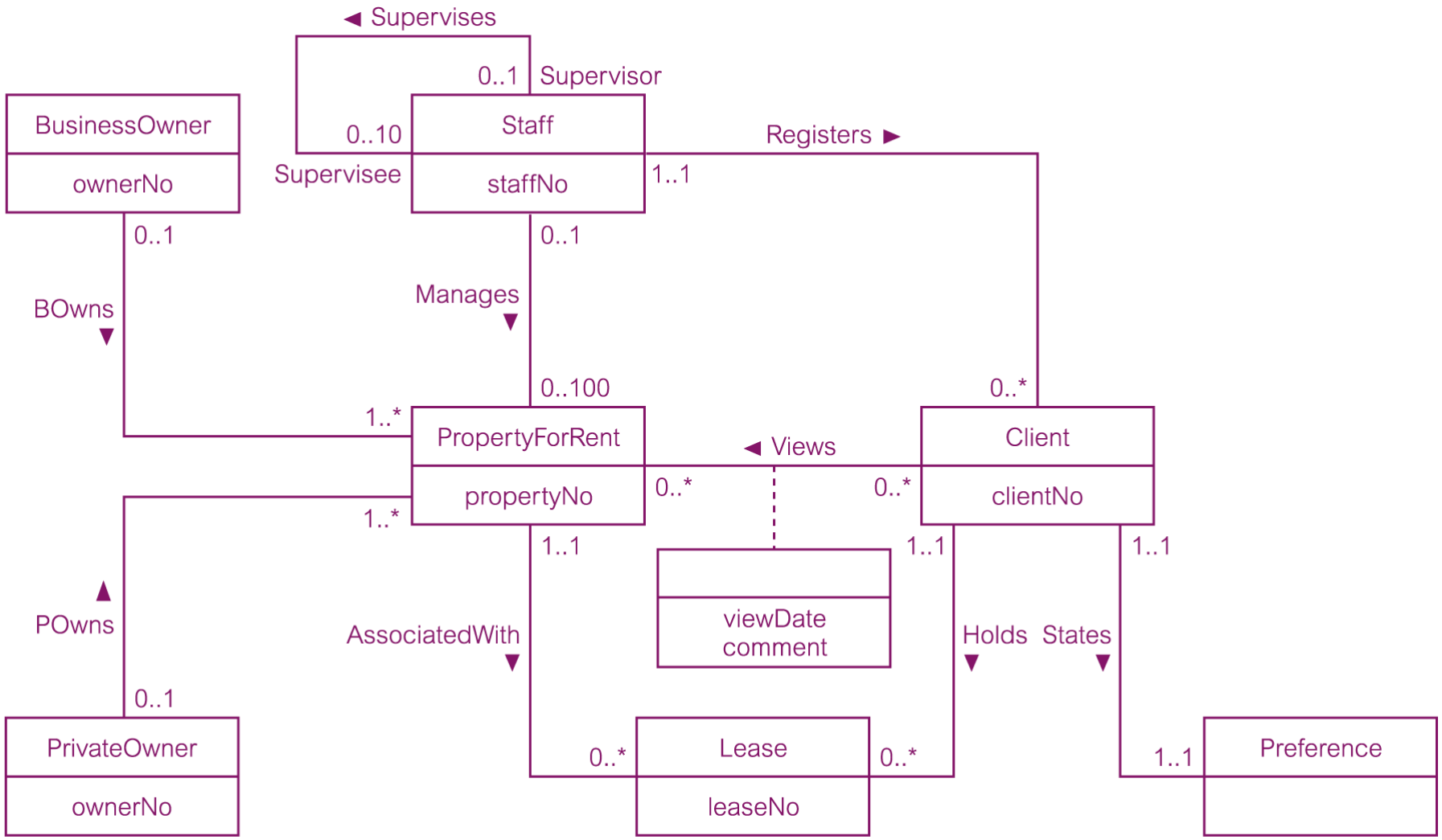


# Attributes on relationships

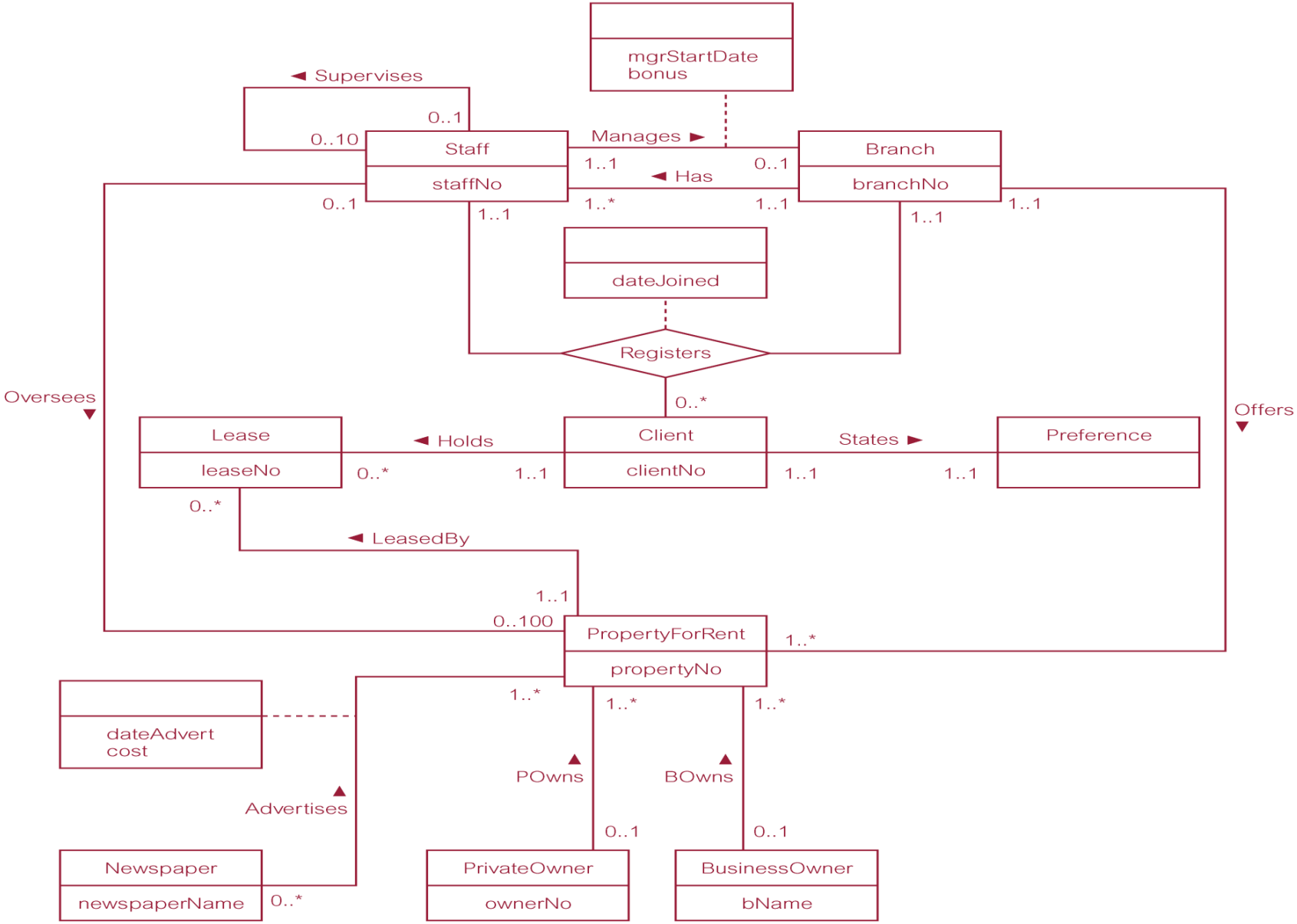



- In rare situations where location of attributes not immediately obvious.
- It may indicate that relationship conceals unidentified entity.

# *DREAMHOME* Conceptual ERD (Staff View with PKs)



# *DreamHome* Conceptual ERD (Branch View with PKs)



# References and Essential Readings


- Module Reading List: https://rl.talis.com/3/westminster/lists/2CAA7D6B-DCAD-AB71-C97B-7FEFCB499C28.html

- Connolly, T. & Begg, C. E. (2015). Database systems: a practical approach to design, implementation and management. 6th Edition (Global Edition). Pearson Education. Ch. 1, 12, 13, 16.

- Elmasri, R. & Navathe, S. (2017). Fundamentals of Database Systems. 7th Edition (Global Edition). Pearson Education. Ch. 1, 2, 3.