

Chapter 12

Entity-Relationship Modeling

1

COMP211

Objectives

- What is Entity-Relationship modelling?
- 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).

2

COMP211

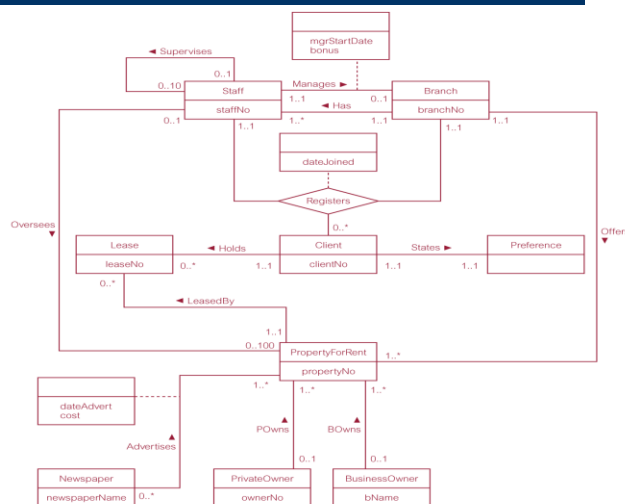
What is Entity-Relationship modelling?

- Entity–Relationship (ER) modeling is a top-down database design approach.
- Starts with identification of entities and relationships between the entities, which are of interest to the organization.
- And then applies successive top-down refinements to identify lower-level entities, relationships, and the associated attributes.

3

COMP211

ER diagram of Branch user views of *DreamHome* (Fig. 12.1)



4

COMP211

Concepts of the ER Model

- Entity types
- Relationship types
- Attributes

5

COMP211

Entity Type

- Entity types
- Relationship types
- Attributes

- Entity type
 - Group of objects with same properties, identified by enterprise as having an independent existence.
 - The basic concept of ER model.
 - Each uniquely identifiable object of an entity type is referred to as an **entity occurrence**.

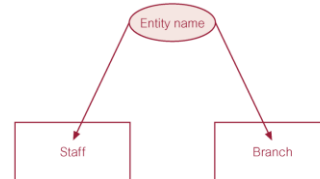
6

COMP211

Diagrammatic representation of Staff and Branch entity types

- Entity types
- Relationship types
- Attributes

- Each entity type is shown as a rectangle, labeled with the name of the entity, which is usually a singular noun.
- In UML, the first letter of each word in the entity name is uppercase.



7

COMP211

Strong and Weak Entity Types

- Entity types
- Relationship types
- Attributes

- Strong Entity Type
 - Entity type that is *not* existence-dependent on some other entity type.
 - A characteristic is that each entity occurrence is uniquely identifiable using the primary key attribute(s) of that entity type.
- Weak Entity Type
 - Entity type that is existence-dependent on some other entity type. It cannot exist without the entity with which it has a relationship.
 - Such weak entity has a primary key that is partially or totally derived from the owner entity in the relationship.
 - A characteristic is that each entity occurrence has a mandatory foreign key — **a foreign key attribute that cannot be null.**

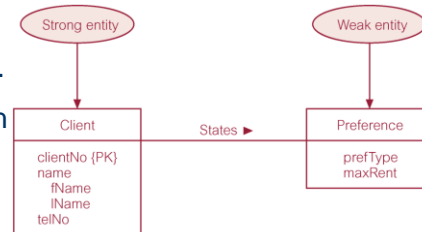
8

COMP211

Example on Strong and Weak entity type

- Entity types
- Relationship types
- Attributes

- Note that there is no primary key for the Preference entity.
- We can uniquely identify each preference only through the relationship that a preference has with a client who is uniquely identifiable using the primary key for the Client entity type, namely clientNo.
- In the example, the Preference entity is described as having existence dependency for the Client entity, which is referred to as being the owner entity.



9

COMP211

Other Examples of weak entity types

- Entity types
- Relationship types
- Attributes

- Can you think of any other examples of weak entities?
 - A company insurance policy insures an employee and any dependents, the DEPENDENT cannot exist without the EMPLOYEE; that is, a person cannot get insurance coverage as a dependent unless the person is a dependent of an employee. DEPENDENT is the weak entity in the relationship "EMPLOYEE has DEPENDENT".
 - An Order has OrderItems

10

COMP211

Relationship Types

- Entity types
- Relationship types
- Attributes

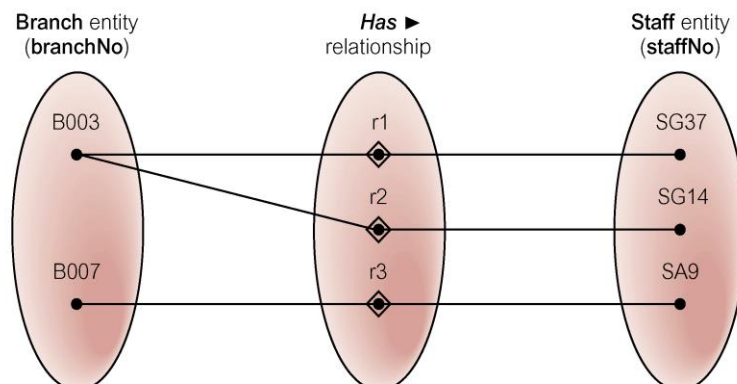
- Relationship type
 - Set of meaningful associations among entity types.
 - Each relationship type is given a name that describes its function.
 - A uniquely identifiable association which includes one occurrence from each participating entity type is called relationship occurrence.

11

COMP211

Semantic net of *Has* relationship type

- Entity types
- Relationship types
- Attributes



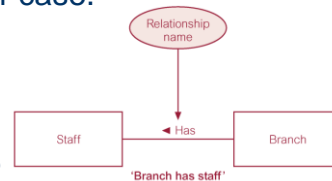
12

COMP211

ER diagram of Branch *Has* Staff relationship

- Entity types
- Relationship types
- Attributes

- The concepts of ER model can be used to represent the relationships between entities.
- Each relationship type is shown as a line connecting the associated entity types and labeled with the name of the relationship.
- Usually, a relationship name is a verb, with the first letter of each word shown in upper case.
- An arrow symbol is placed beside the relationship name indicating the correct direction.
- Whenever possible, a relationship name should be unique for a given ER model.



13

COMP211

Degree of Relationship Types

- Entity types
- Relationship types
- Attributes

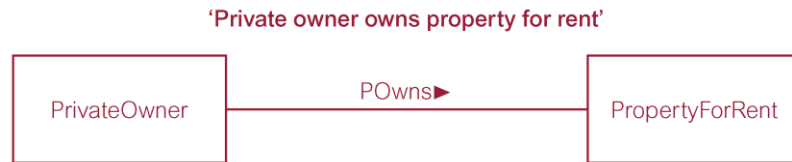
- Number of participating entity types (also referred to as **participants**) in a relationship is called the **degree of relationship type**.
- Relationship of degree
 - two is **binary**
 - three is **ternary**
 - four is **quaternary**.

14

COMP211

Binary relationship called *POwns*

- Entity types
- Relationship types
- Attributes

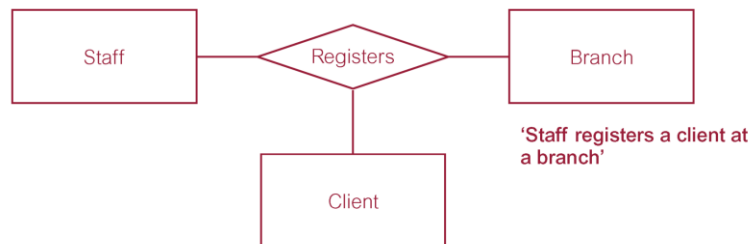


15

COMP211

Ternary relationship called *Registers*

- Entity types
- Relationship types
- Attributes

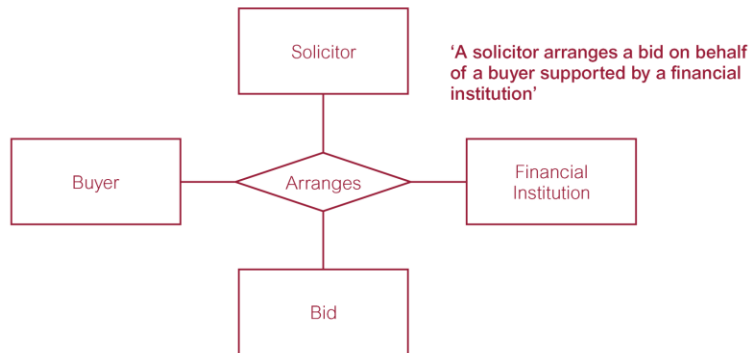


16

COMP211

Quaternary relationship called Arranges

- Entity types
- Relationship types
- Attributes



17

COMP211

Recursive Relationship

- Entity types
- Relationship types
- Attributes

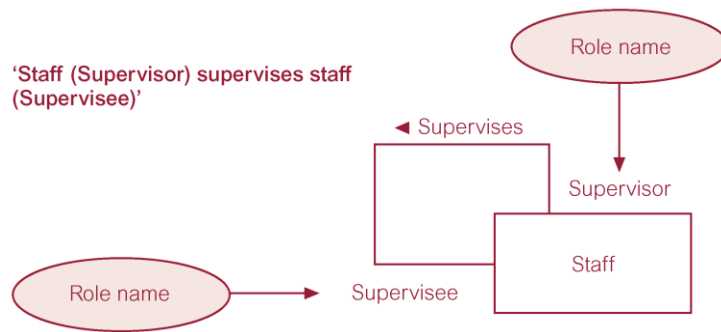
- Recursive Relationship
 - Relationship type where the *same* entity type participates more than once in *different roles*.
 - Sometimes called ***unary*** relationships.
- Relationships may be given **role names** to indicate the purpose that each participating entity type plays in a relationship.

18

COMP211

Recursive relationship called *Supervises* with role names

- Entity types
- Relationship types
- Attributes



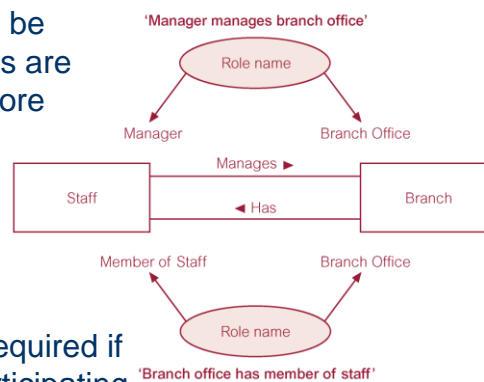
19

COMP211

Entities with Role Names

- Entity types
- Relationship types
- Attributes

- Role names may also be used when two entities are associated through more than one relationship.
- The use of role names clarifies the purpose of each relationship.
- Role names are not required if the function of the participating entities in a relationship is unambiguous.



20

COMP211

Attributes

- Entity types
- Relationship types
- Attributes

- Attribute
 - Property of an entity or a relationship type.
- Attribute Domain
 - Data type and set of allowable values for one or more attributes.
- Simple Attribute
 - Attribute composed of a single component with an independent existence. E.g. salary
- Composite Attribute
 - Attribute composed of multiple components, each with an independent existence.
 - For example, address attribute can be subdivided into street, city.

21

COMP211

Multi-valued Attributes

- Entity types
- Relationship types
- Attributes

- Single-valued Attribute
 - Attribute that holds a single value for each occurrence of an entity type.
 - For example, each occurrence of Branch entity type has a single value for the branch number.
- 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;
 - a person may have several college degrees
 - A multi-valued attribute may have a set of numbers with upper and lower limits.

22

COMP211

- Entity types
- Relationship types
- Attributes

Derived Attributes

- **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.
 - For example,
 - the duration attribute of the *Lease* entity is calculated from the *rentStart* and *rentFinish* attributes
 - an employee's age may be found by computing the integer value of the difference between the current date and employee's date of birth

23

COMP211

- Entity types
- Relationship types
- Attributes

Storing Derived Attributes?

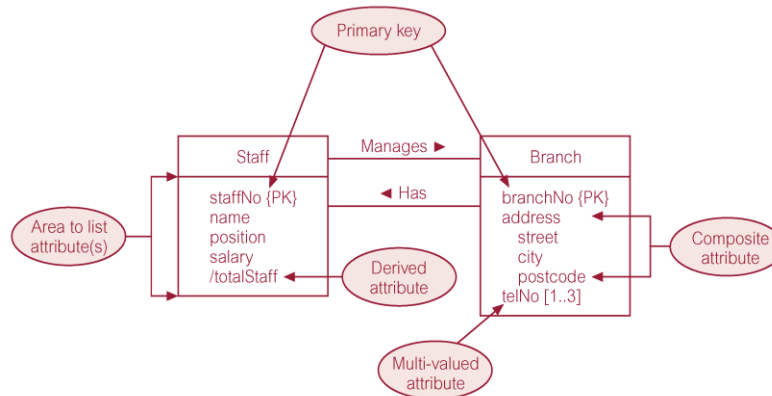
- **Should Derived Attribute be stored?**
 - The decision to store derived attributes in database tables depends on the processing requirements and the constraints placed on a particular application. The designer should be able to balance the design in accordance with such constraints.

	DERIVED ATTRIBUTE	
	STORED	NOT STORED
Advantage	Saves CPU processing cycles Saves data access time Data value is readily available Can be used to keep track of historical data	Saves storage space Computation always yields current value
Disadvantage	Requires constant maintenance to ensure derived value is current, especially if any values used in the calculation change	Uses CPU processing cycles Increases data access time Adds coding complexity to queries

24

COMP211

ER diagram of Staff and Branch entities and their attributes



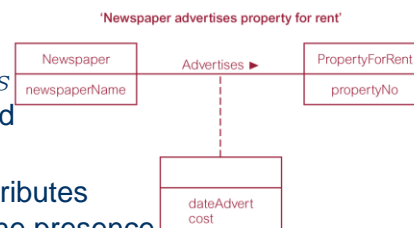
25

COMP211

Attributes on Relationships

- Entity types
- Relationship types
- Attributes

- The presence of one or more attributes assigned to a relationship may indicate that the relationship conceals an unidentified entity type.
- The diagram shows a relationship called *Advertises* with attributes dateAdvert and cost.
- The presence of these two attributes on the relationship indicates the presence of an entity called *Advert*.



26

COMP211

Examples of Attributes on Relationships

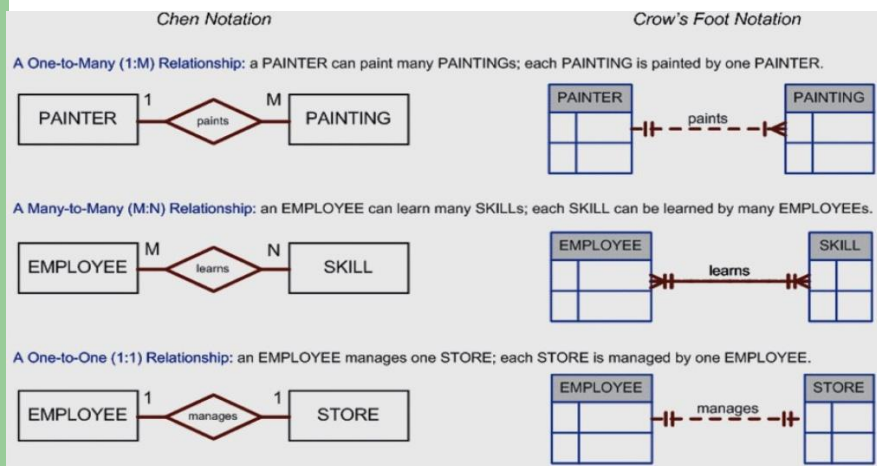
- Entity types
- Relationship types
- Attributes

- Can you think of any other examples of attributes on relationships?
 - A student enrolls in courses
 - A customer books flight tickets

27

COMP211

Different ER notation



28

COMP211

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 enterprise.

29

COMP211

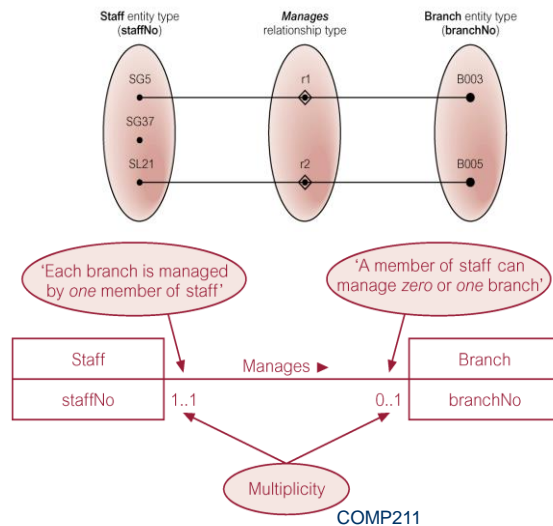
Structural Constraints (cont'd)

- 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 (*:*)
- Examples:
 - A member of staff manages a branch (1:1)
 - A member of staff oversees properties for rent (1:*)
 - Newspapers advertise properties for rent (*:*)

30

COMP211

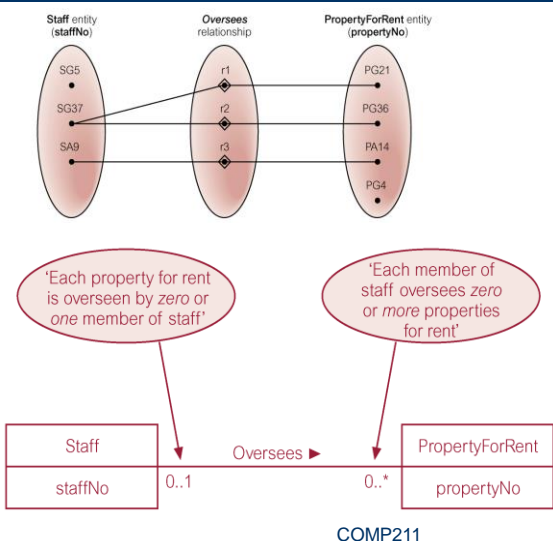
One-to-One (1:1) Relationships



ER diagram showing the multiplicity of the Staff *Manages* Branch one-to-one (1:1) relationship.

31

One-to-Many (1:*) Relationships



ER diagram showing the multiplicity of the Staff *Oversees* PropertyForRent one-to-many (1:*) relationship.

32

One-to-Many (1:*) Relationships

- Staff oversees PropertyForRent

Staff								
PK Relation	<u>staffNo</u>	fName	lName	position	sex	DOB	salary	branchNo
	SL21	John	White	Manager	M	1-Oct-45	30000	B005
	SG37	Ann	Beech	Assistant	F	10-Nov-60	12000	B003
	SG14	David	Ford	Supervisor	M	24-Mar-58	18000	B003
	SA9	Mary	Howe	Assistant	F	19-Feb-70	9000	B007
	SG5	Susan	Brand	Manager	F	3-Jun-40	24000	B003
	SL41	Julie	Lee	Assistant	F	13-Jun-65	9000	B005

Foreign key implementation of "each property is overseen by 0..1 staff", thus allowing nulls to exist.

PropertyForRent									
propertyNo	street	city	postcode	type	rooms	rent	ownerNo	<u>staffNo</u>	branchNo
PA14	16 Holhead	Aberdeen	AB7 5SU	House	6	650	CO46	SA9	B007
PL94	6 Argyll St	London	NW2	Flat	4	400	CO87	SL41	B005
PG4	6 Lawrence St	Glasgow	G11 9QX	Flat	3	350	CO40		B003
PG36	2 Manor Rd	Glasgow	G32 4QX	Flat	3	375	CO93	SG37	B003
PG21	18 Dale Rd	Glasgow	G12	House	5	600	CO87	SG37	B003
PG16	5 Novar Dr	Glasgow	G12 9AX	Flat	4	450	CO93	SG14	B003

33

COMP211

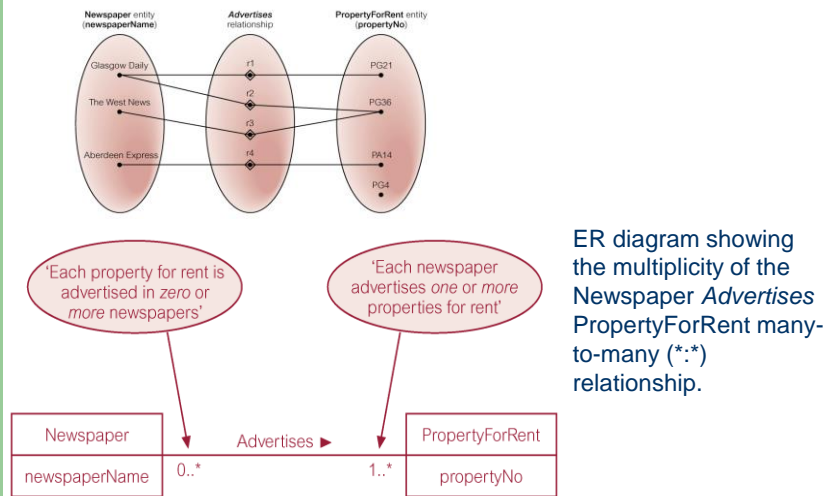
One-to-Many (1:*) Relationships (cont'd)

- If we know the actual minimum and maximum values for the multiplicity, we can display these instead.
- For example, if a member of staff oversees a minimum of zero and a maximum of 100 properties for rent, we can write "0..*" with "0..100"

34

COMP211

Many-to-Many (*:*) Relationships



ER diagram showing the multiplicity of the Newspaper *Advertises* PropertyForRent many-to-many (*:*) relationship.

35

COMP211

Question to Reflect on

- Compare the difference between the two models we have discussed:
 - Relational model
 - Entity-relationship model

36

COMP211

Summary

We have covered the following:

- Binary relationships
 - One-to-one (1:1)
 - One-to-many(1:*)
 - Many-to-many (*:*)
- Terms:
 - Composite attribute, Multi-valued attribute, Derived attribute