Database Systems 2

Lecture 2

Entity Relationship Modelling

Entities

An entity is any object or concept in the system that we want to model and store information about.

- Some books refer to entity types or entity sets (eg Lecturer)
- Individual objects are called entities (eg Kevin Wilson)
- The general term entity is often used for both where the meaning is obvious.

Chen's notation

Barker's notation

Weak and Strong Entities

Strong Entity

- One who's existence does not depend on another entity.

Weak Entity

- One who's existence does depend on the existence of another entity.
- Sometimes called child or dependent entities
- Shown with double lines

Is there always a relationship between them?

Attributes

A property of an entity or a relationship.
Each attribute can have any value from its domain.

Attributes can be:

- simple or composite
  - Salary is a single number
  - Address can be broken down into (Street, Area, City, Postcode)

- single-valued or multi-valued (for a single entity)
  - Branch would have a single Branch Number
  - Branch might have several Telephone Numbers

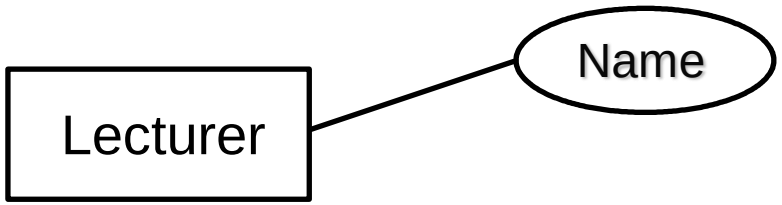
- Derived
  - The age attribute can be derived from the Date of Birth attribute
  - They are related
  - A derived attribute may involve attributes from several entities.

Attributes can be shown on ER models

- They appear inside ovals and are attached to their entity.

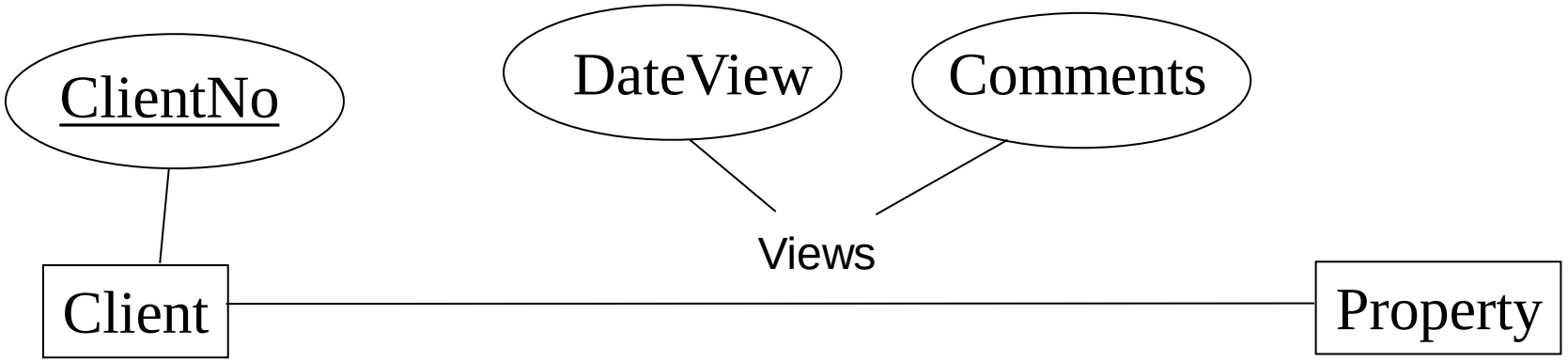
- Note that entity types can have a large number of attributes... If all are shown then the diagrams would be confusing.

- Only show an attribute if it adds information to the ER diagram, or clarifies a point.

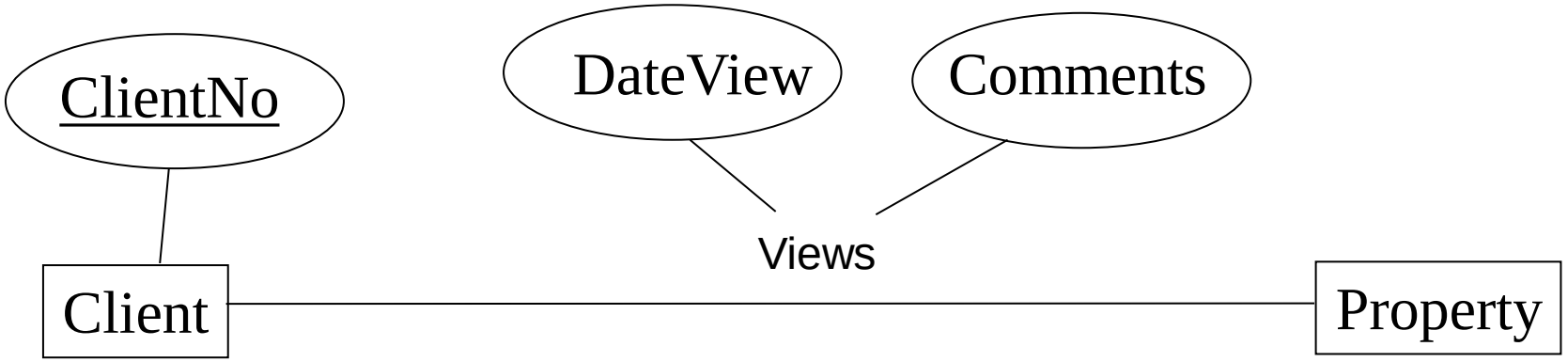


Attributes on Relationships

Attributes can be assigned to relationships.



How would you implement this relationship?



ViewNo	DateView	Comments	FKClientNo*	FKPropNo*

ClientNo	Name

PropNo	Address

Cardinality

Most relationships are not one-to-one

- For example, a manager usually manages more than one employee

This is described by the *cardinality* of the relationship, for which there are four possible categories.

- One to one (1:1) relationship
- One to many (1:M) relationship
- Many to one (M:1) relationship
- Many to many (M:N) relationship

How is the cardinality of a relationship shown on an ER diagram?

Participation Constraints

The participation of an entity in a relationship can be optional or mandatory.

If the participation is mandatory:

- Every instance of an entity at one end of the relationship must be related to an instance of the entity at the other end.

If the participation is optional:

- An instance of an entity at one end of the relationship might not be related to an instance of the entity at the other end.

The optionality can be different at each end of the relationship

- For example, a student must be on a course. This is mandatory.
- So the relationship 'student studies course' is mandatory.

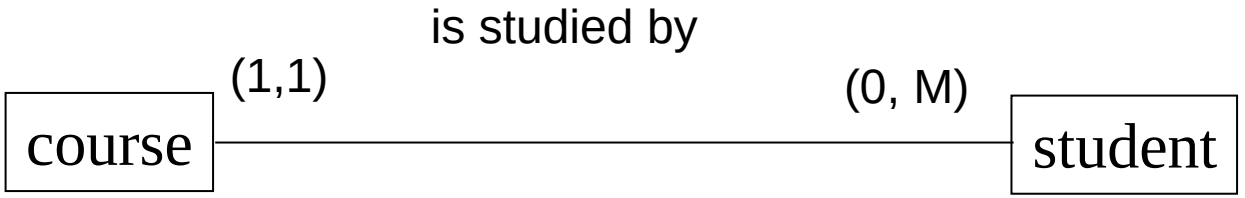
- But a course can exist before any students have enrolled.
- Thus the relationship 'course is_studied_by student' is optional.

How is optionality shown on an ER diagram?

Optionality cont...

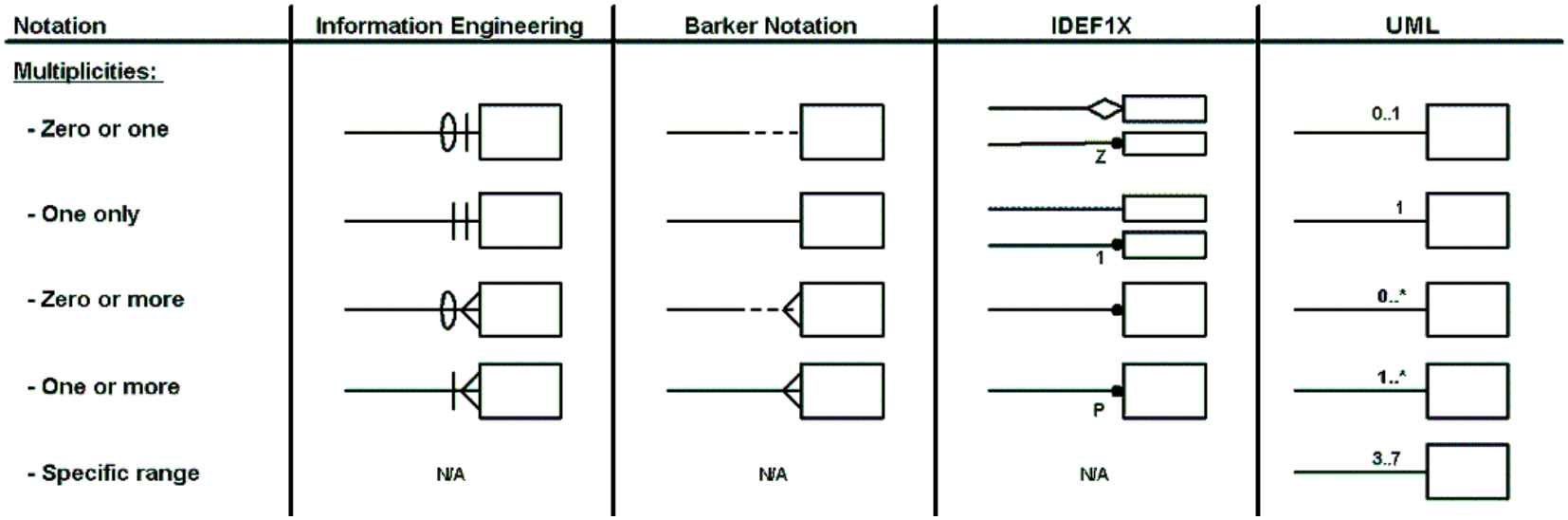
One course is studied by zero or more students
One student studies one and only one course

Alternative notation:



What would it mean if the student was changed to (1, M)?

Alternative Notations



http://www.agiledata.org/essays/dataModeling101.html

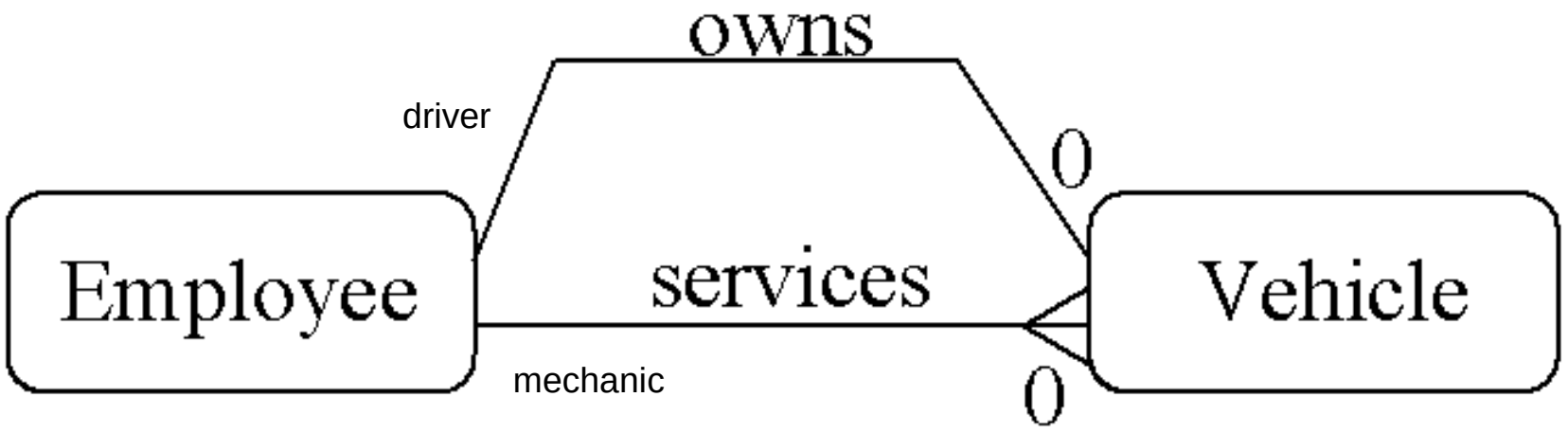
Just remember that all ER diagrams do the same job. They display:

Entities, Relationships, Cardinality, Optionality

Historical evolution of ER notation: (also on the VLE)
http://www.essentialstrategies.com/documents/comparison.pdf

Parallel relationships

Parallel relationships occur when there are two or more relationships between two entity types (e.g. employees own and service cars).



Note the use of role names to clarify the model.

Implementing Parallel Relationships

Each relationship is mapped according to the rules, and we end up with two foreign keys added to the Vehicle table.

In order to distinguish between the two roles we can give the foreign keys different names.

So we a field called *owner_no* in order to represent the 'owns' relationship.

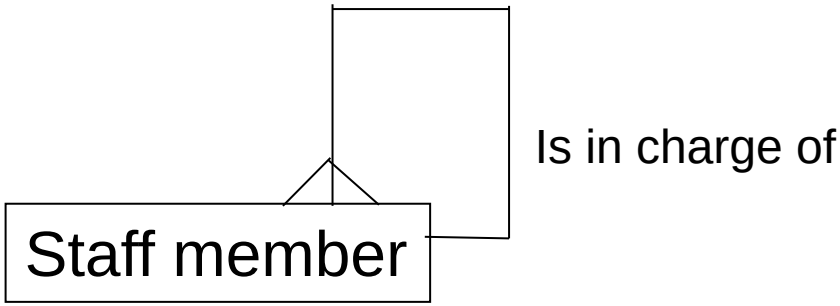
We then add a field called *serviced_by* attribute in order to represent the 'services' relationship.

Employee_no	Emp_Name	Age
E05	Jones	22
E22	Galbraith	39
E10	Wilson	47

Reg_no	Owner_no*	Serviced_by*	Make	Model
AB51 DVL	E05	E10	Ford	Granada
AA22 TER	E10	E10	Ford	Ka
BG33 TTT	E22	E05	Opel	Kadett

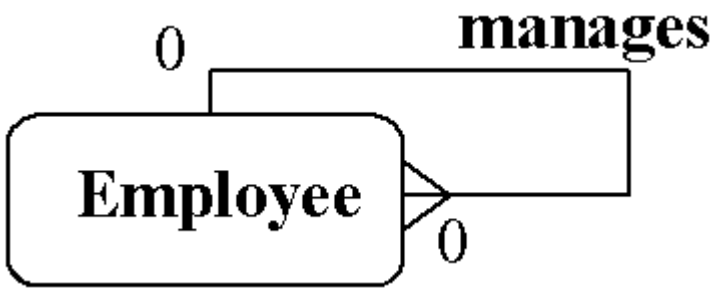
Recursive Relationships

It is possible to have a unary (degree one) relationship.



How would you implement this?

Implementing Unary relationships



Some employees can manage other employees

Each employee has an employee_no as its primary key

We represent the manages relationship by adding a *manager_no** as a foreign key. – or by adding intermediate entity.

This is in fact the employee_no of the manager.

It is given a different name to clearly convey what it represents, and to ensure that all the entity type's attributes have unique names, as to do otherwise would be invalid.

Implementing Unary Relationships

After mapping

Employee(employee_no,*manager_no, name,...)**

So in general, for unary 1:n relationships, the foreign key is the primary key of the same table, but is given a different name.

Note that the relationship is optional in both directions because not all staff can be managers, and the top manager is not managed by anybody else.

Employee_no	Manager_no*	Emp_Name	Age
E05	E21	Jones	22
E22	E05	Galbraith	39
E21	NULL	Tiddlington-Smythe	61
E93	E22	Brown	39
E10	E05	Wilson	47

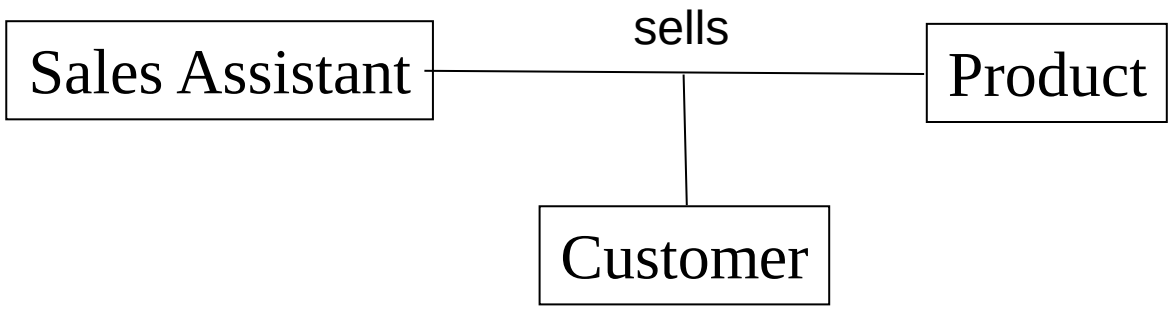
What is the management hierarchy?

Relationship Degree

The number of participating entities in a relationship is known as the degree of the relationship.

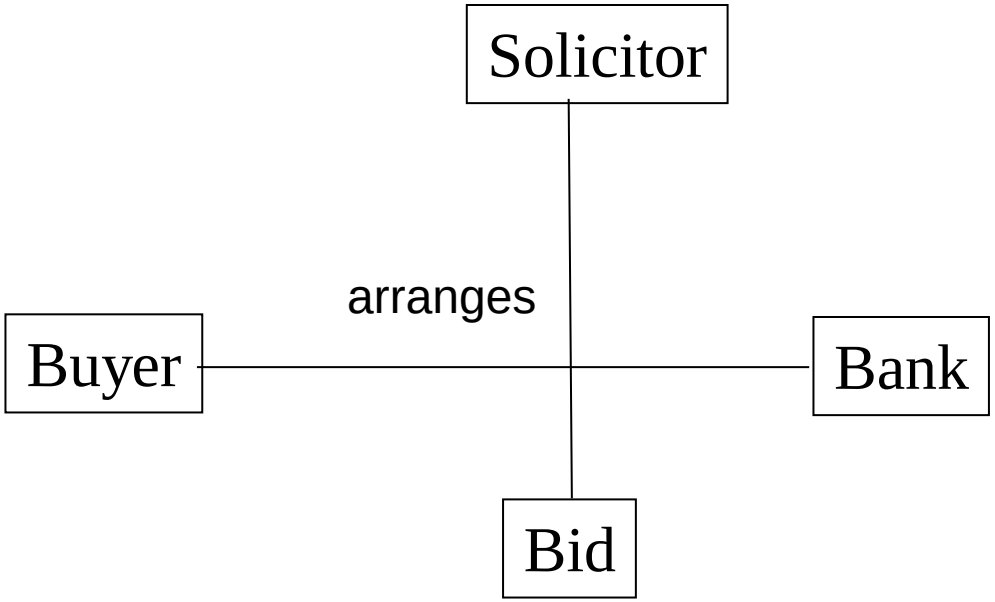
If there are two entity types involved it is a *binary* relationship type

If there are three entity types involved it is a *ternary* relationship type



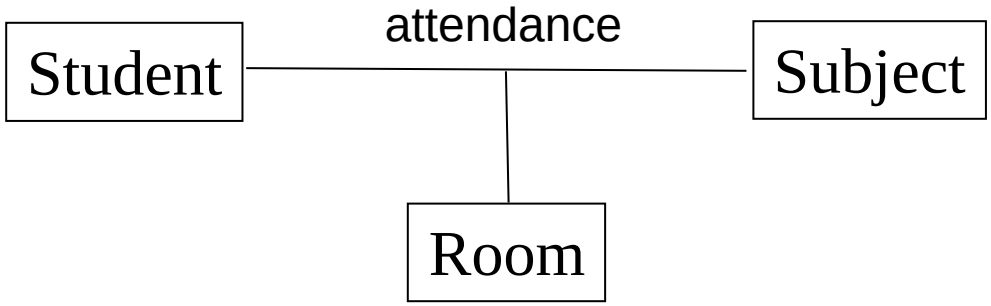
Higher order Relationships

Can you think of an example of a quaternary (degree four) relationship?



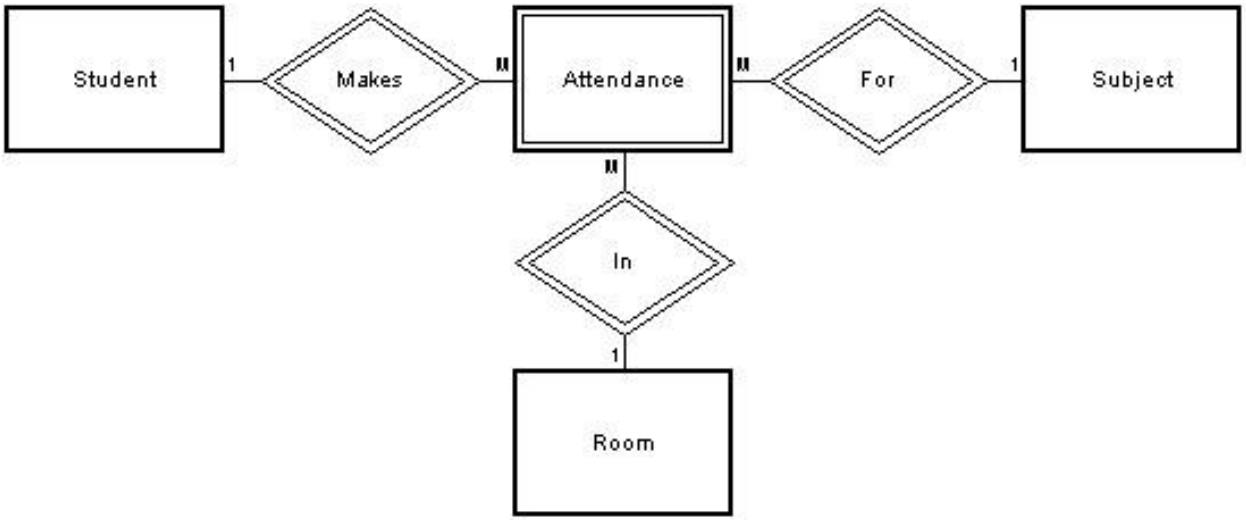
Higher order relationships are also possible.

Implementing n-ary Relationships



Complex relationships such as this are generally implemented as tables

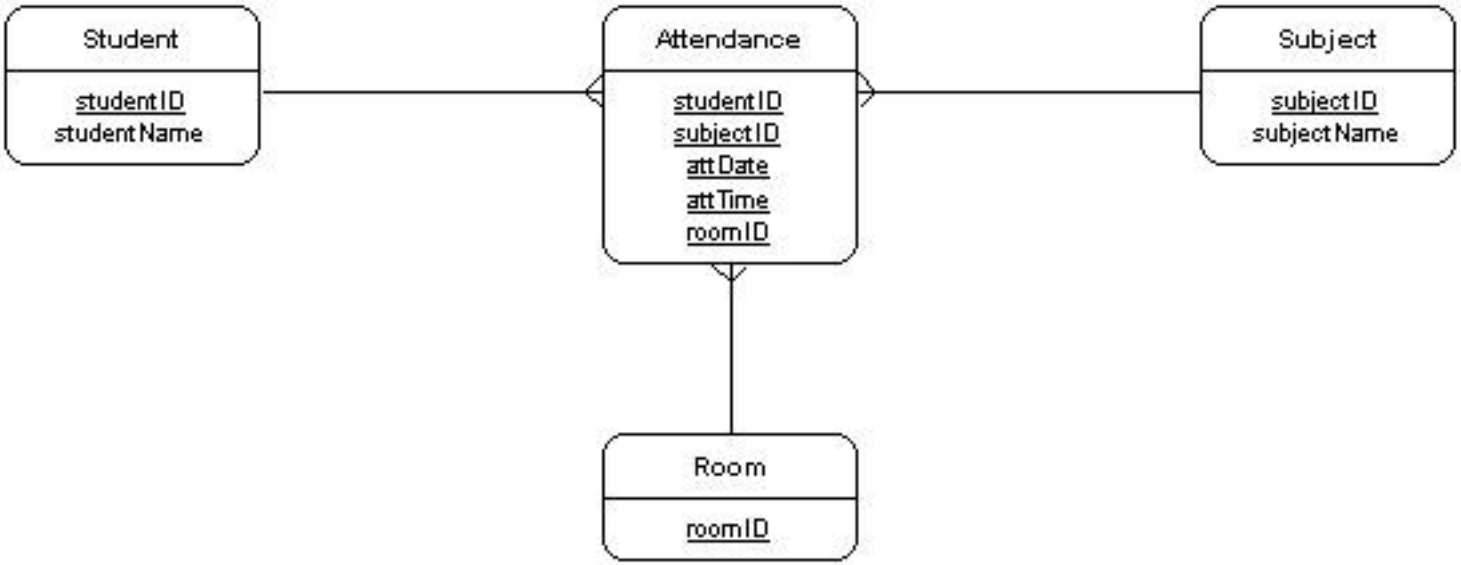
Implementing n-ary Relationships



Notice that Attendance is a weak entity type, and the three relationships are also weak.

Together with the appropriate cardinality of 1:M, this ensures that the primary keys of the dominant entities become foreign primary keys in the weak entity.

Implementing n-ary Relationships



Do all of the fields in the 'Attendance' table have to be primary keys for that table, in this particular case?

Does this apply to a normal link table?