# Entity-Relationship Modelling

5DV119 — Introduction to Database Management

Umeå University
Department of Computing Science

Stephen J. Hegner
hegner@cs.umu.se
http://www.cs.umu.se/~hegner

# Modelling Languages

- Techniques for normalizing a relational schema have been studied.
- However, creating an unnormalized schema and then normalizing is not the best design approach.
- There are *conceptual modelling languages* which are specifically designed for the purpose of describing the setting for a potential database schema.
- Three of the most common are:

  ER: *Entity-Relationship* modelling is the classical tool.

  EER: *Enhanced Entity-Relationship* modelling is an extension of ER which includes ideas related to types and type hierarchies.

  UML: *Universal Modelling Language* is a general modelling language with many uses within software engineering, including the representation of concepts relevant to database schemata.
- In these slides, a brief introduction to the classical ER model, as well as techniques for realizing normalized relational schemata from an ER specification, are presented.

# The ER Approach

- ER is a *conceptual modelling language*.
- It is not normally used to represent final database schemata themselves.
- Rather, it is a tool within the overall *design process* of database schemata.
- There are three fundamental building blocks in the ER model:

  Entity types: are "things" such as employees and projects.

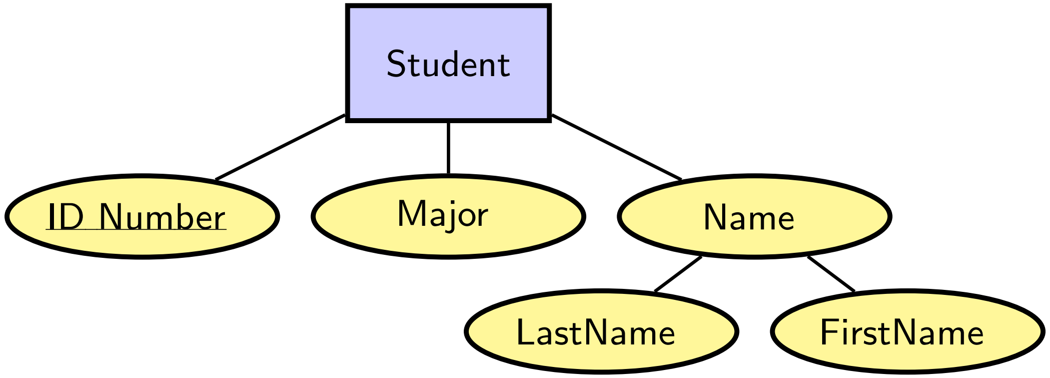
  Relationship types: connect things; *e.g.*, Works_On connects employees and projects.

  Attributes: are the components of entities; *e.g.*, SSN, LastName.

# Entity Types

- Begin with a record structure in a typical imperative language:
- The corresponding ER representation appears as shown to the right.

```
Type Student = Record
  ID_Number: ID_Type;
  Name:    Record
             LastName:   NameType;
             FirstName:  NameType;
           End;
  Major:   MajorType;
End;
```



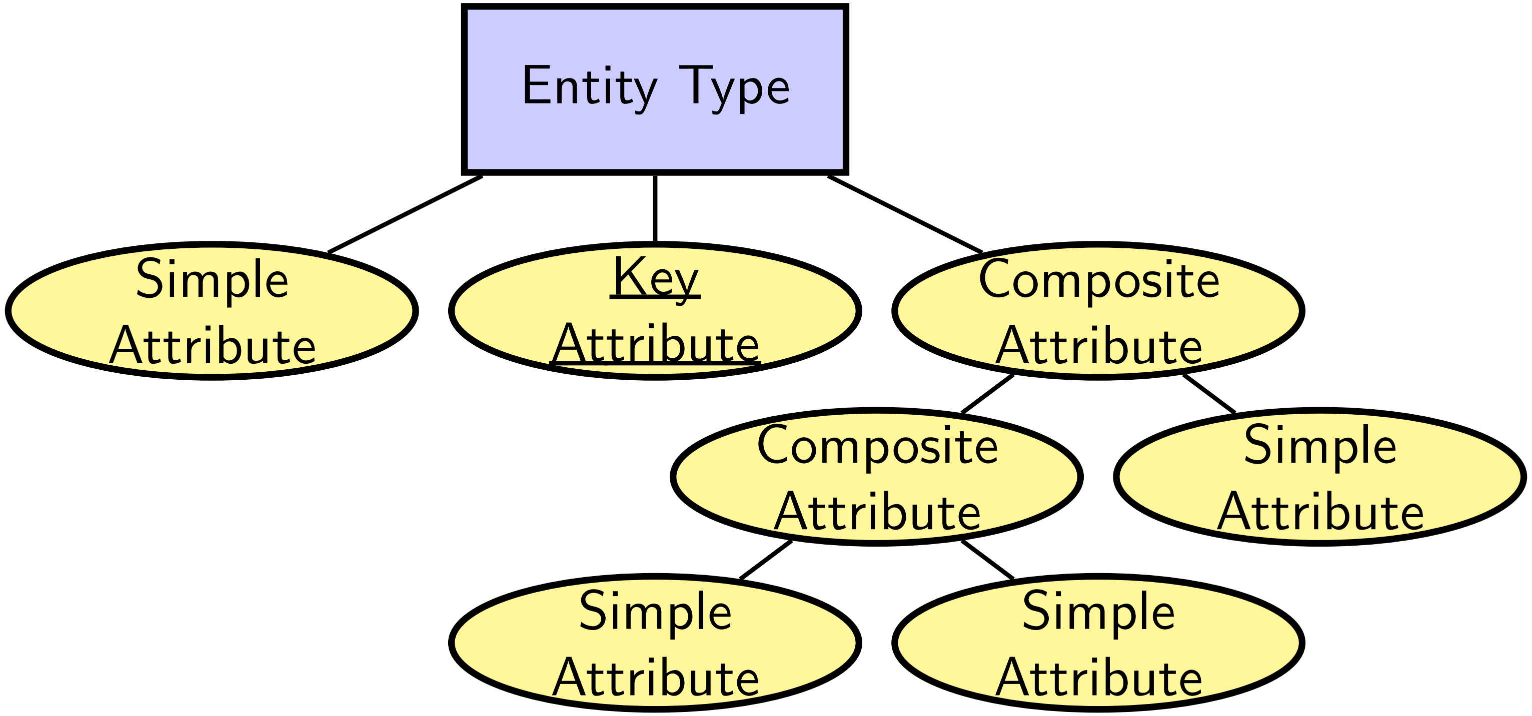

- Note that the types of the data items are not represented in the ER diagram.

Entity type: Each entity type is represented using a rectangle.

Attribute: Each attribute is represented using an ellipse.

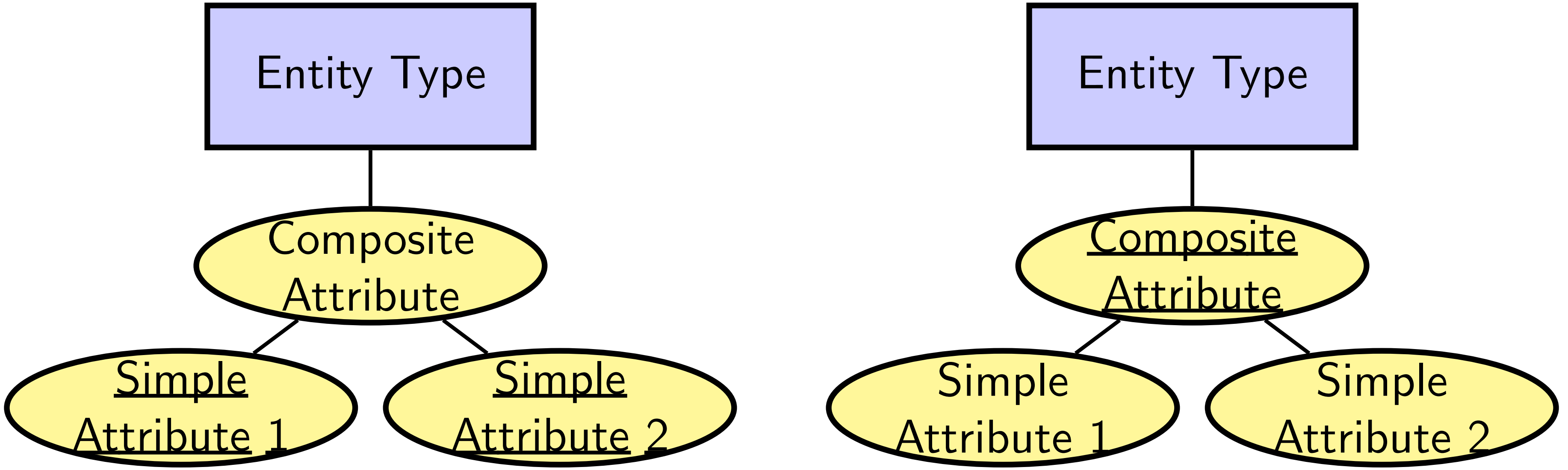
- Keys are underlined.

# A Closer Look



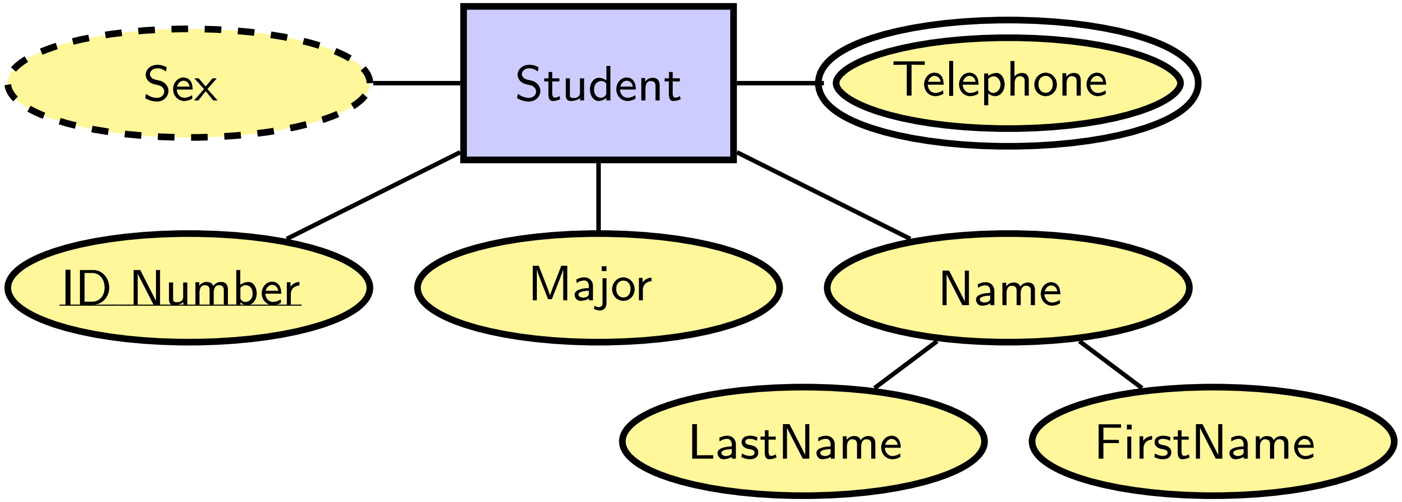

- The most basic options for attributes on an entity are summarized in the figure above.

# Notation for Keys




- In the figure on the left, the entity type has two distinct keys, each a simple attribute which is part of the composite attribute.
- In the figure on the right, there is a single key consisting of two simple attributes.

# Multivalued and Derived Attributes




Multivalued attribute: May take on multiple values for the same entity value.

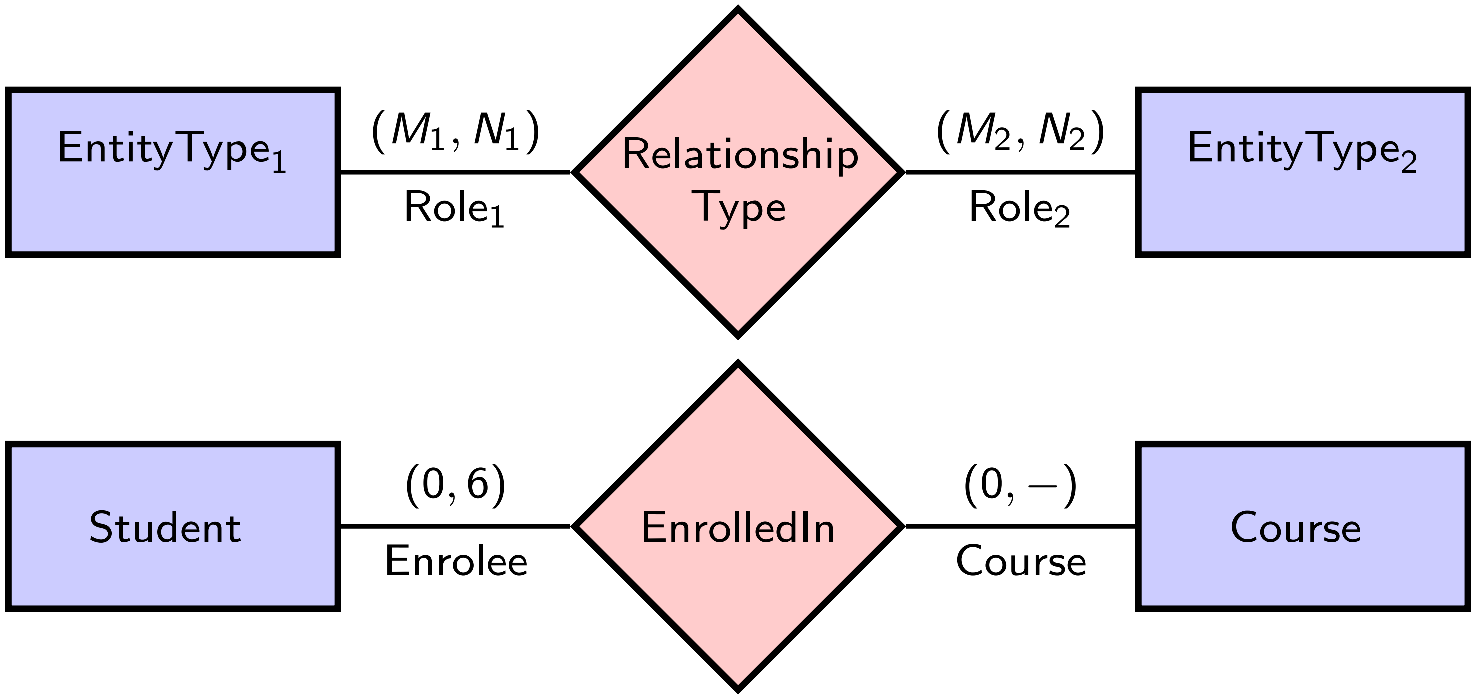
- Denoted by a double ellipse.

Derived attribute: The value is determined by the values of other attributes.

- Denoted by a dashed ellipse.

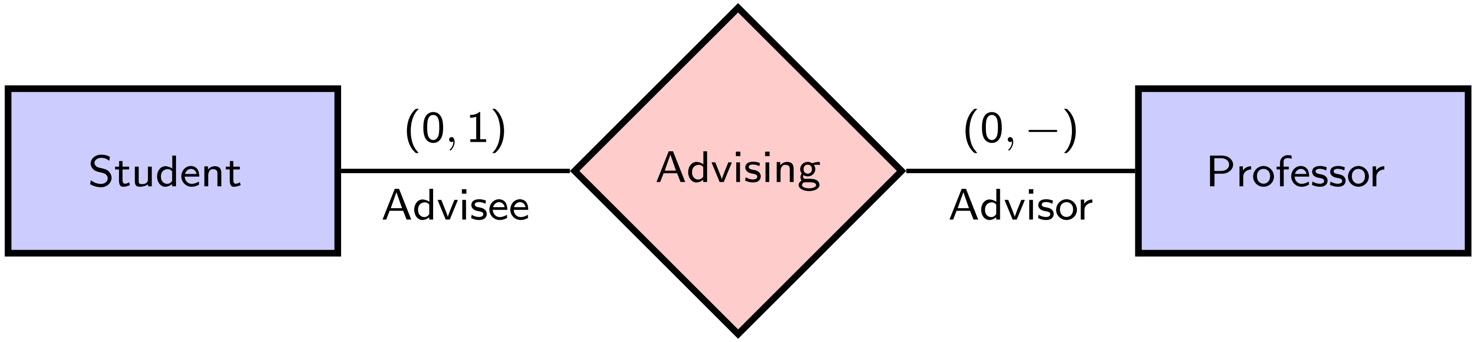
Example: The sex of a person may be determined from the Swedish identification number.

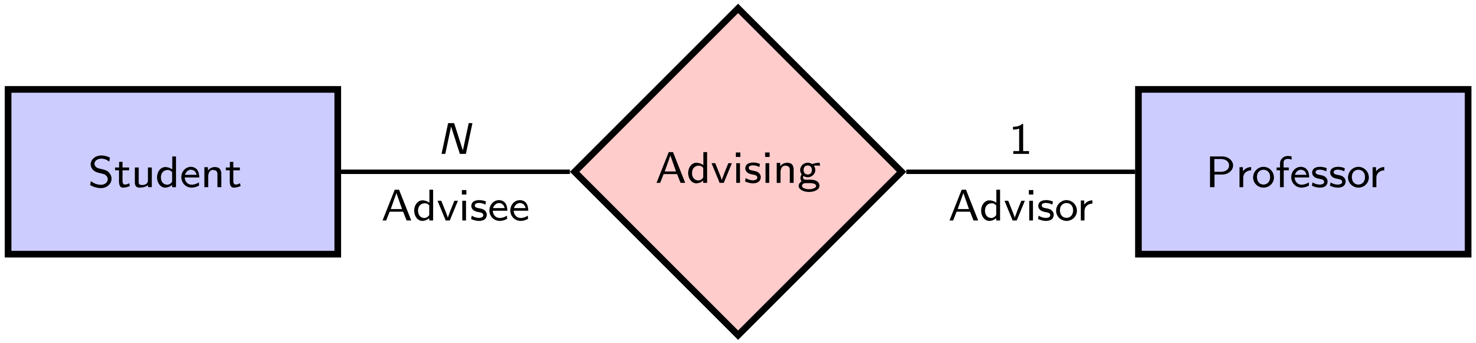
# Relationship Types




- Relationship types are represented using diamonds.
- The minimum $M$ and maximum $N$ number of participants of an entity for each instance is denoted $(M, N)$.
  - A dash for $N$ indicates that there is no upper bound.
- The *rôle* of each entity type in the relationship type may also be assigned a name.
  - However, it is not necessary to indicate a name for each rôle.

# Relationships – Alternate Notation



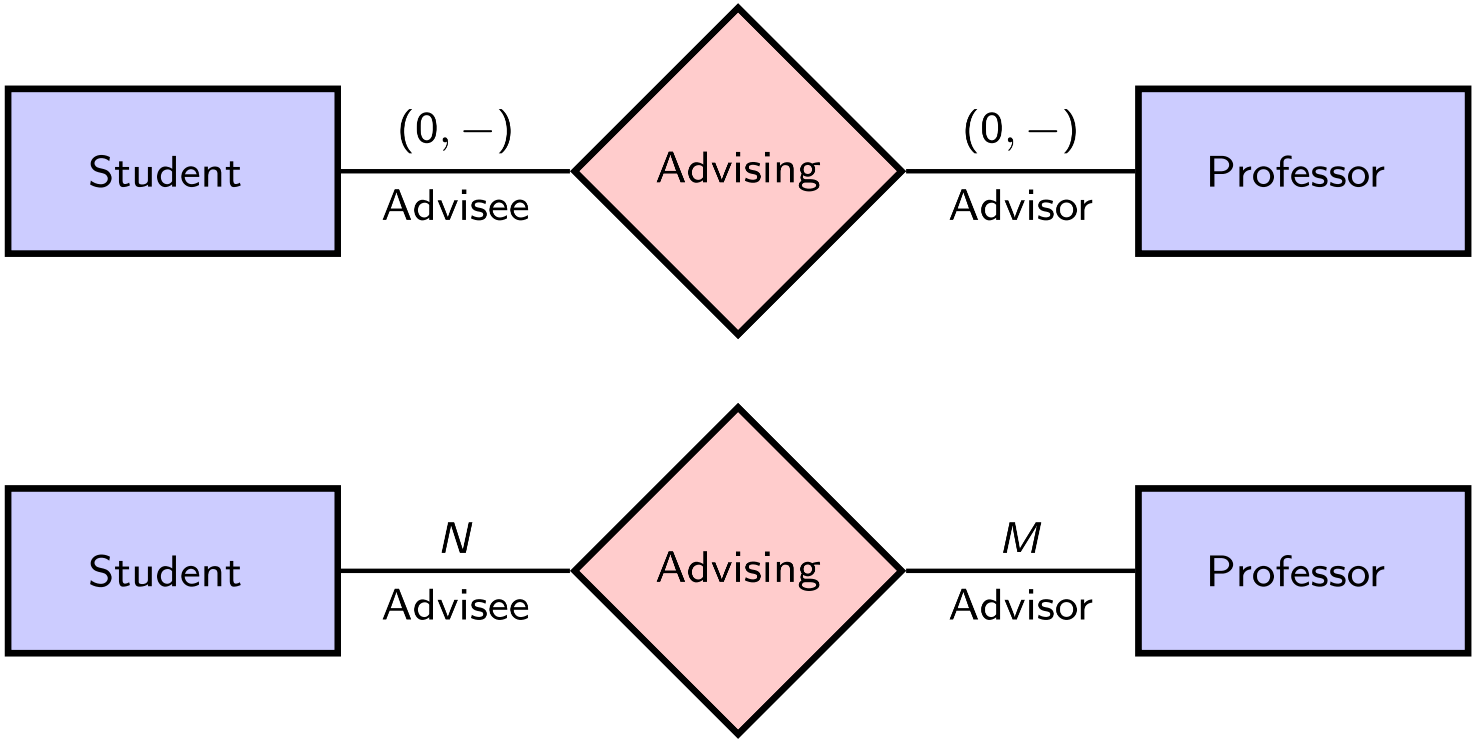

- The lower figure shows the alternate notation.

Note the reversal of sense from the notation in the upper figure.

  - The "$N$" part has only one participation per instance.
  - The "1" part has multiple participations per instance.

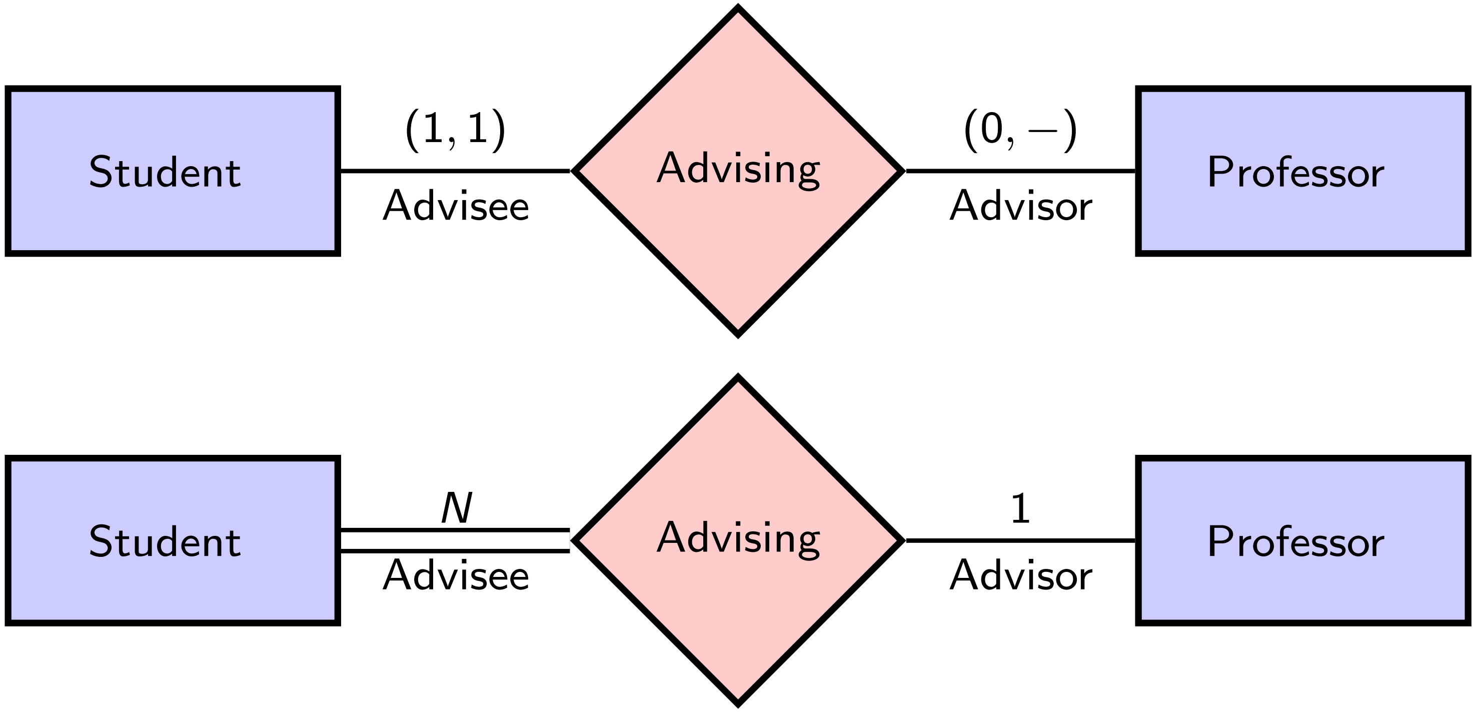
- It means that $N$ students may have one advisor.

# Relationships – Alternate Notation —2




- Now suppose that a student may have several advisors.
- The lower figure shows the alternate notation.
- Different letters $M$ and $N$ may be used to indicate that the number of participations need not be the same.
    - But it is also allowed to use the same letter for each.

# Relationships – Alternate Notation —3

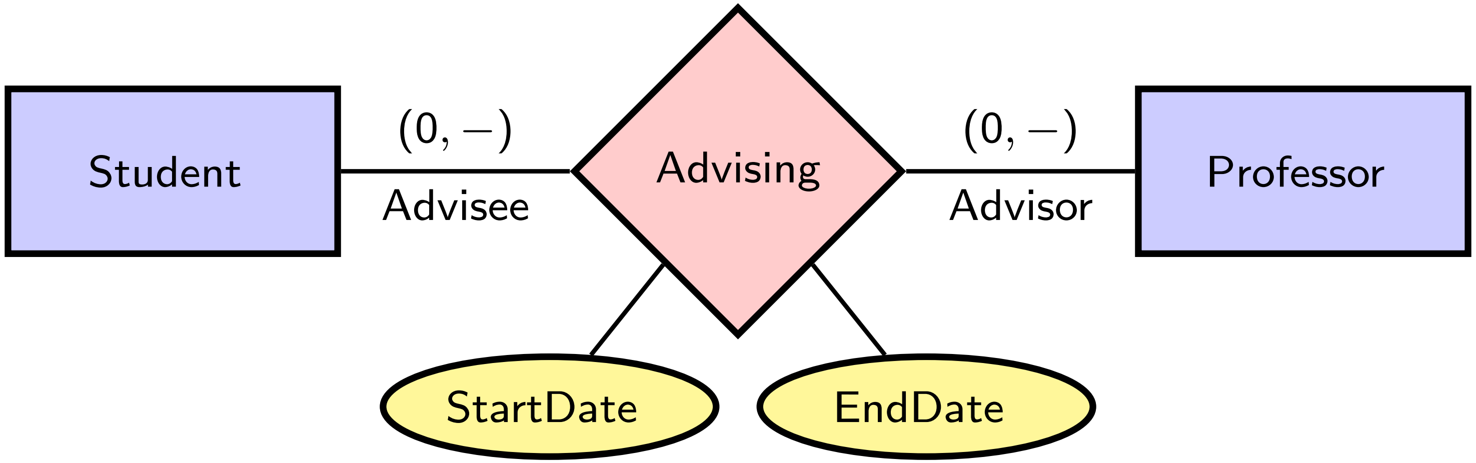



- To distinguish $(0, -)$ from $(1, -)$ and $(0, 1)$ from $(1, 1)$, a double bar is used for 1 in $(1, x)$.
- This is called *total participation*.
- But note that this sense is not swapped along with the 1 and $-$.

# Use of the Alternate Notation

- The $(M, N)$ notation will be used in these slides.
- It is more precise and extends much better to the case of relationships with more than two entities.
- The alternate notation is presented because it is used for most of the presentation in the textbook
- However, the textbook does explain the notation used in these slides as well.
- The only difference is that the textbook uses $(x, N)$ or $(x, M)$ instead of $(x, -)$.
- $(x, -)$ is a more generic representation, because $N$ could also represent a specific number.

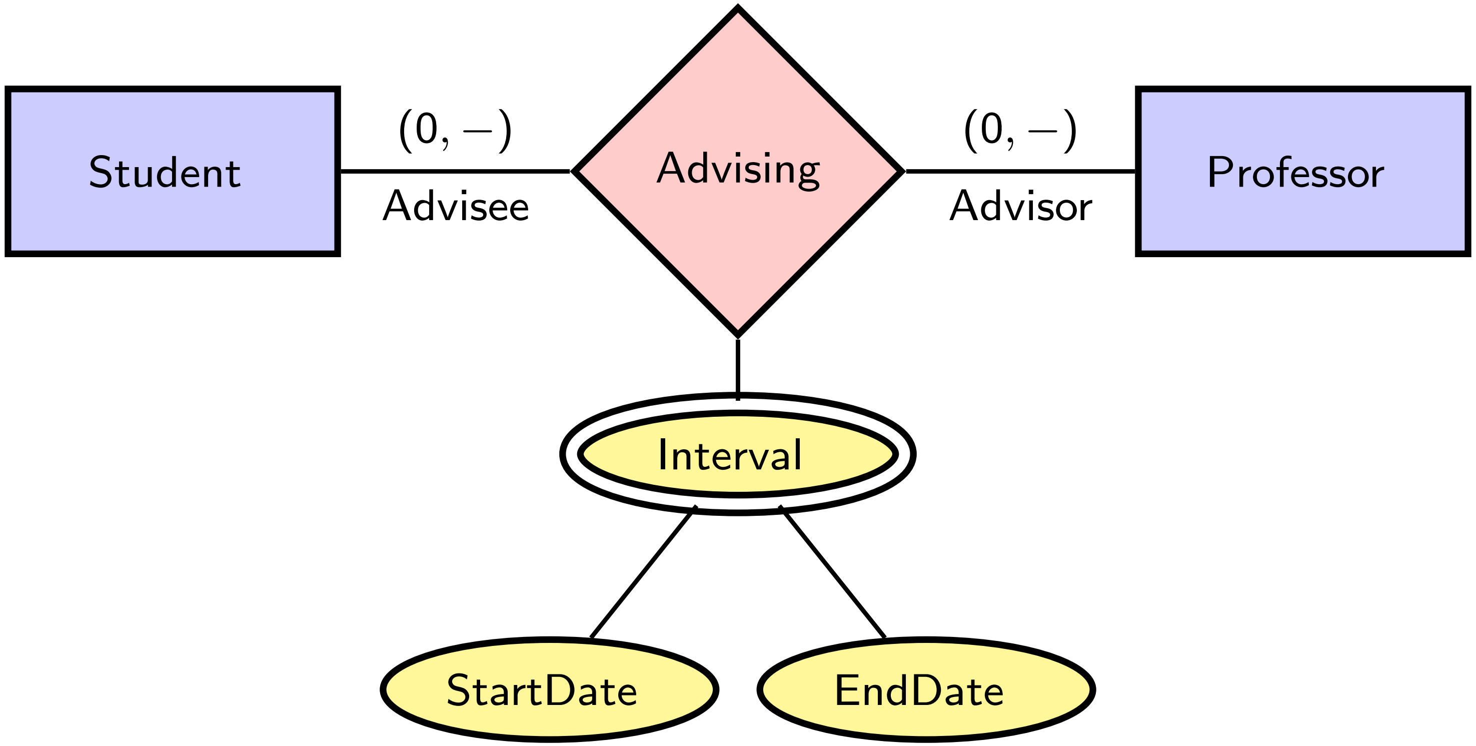
# Relationships May Have Attributes




- Relationships may also have attributes.
- Note that the two attributes do not make sense with either entity individually.
- They are attributes of the association between a student and an advisor.

Modelling problem: Suppose that a student may have the same advisor over distinct time intervals.

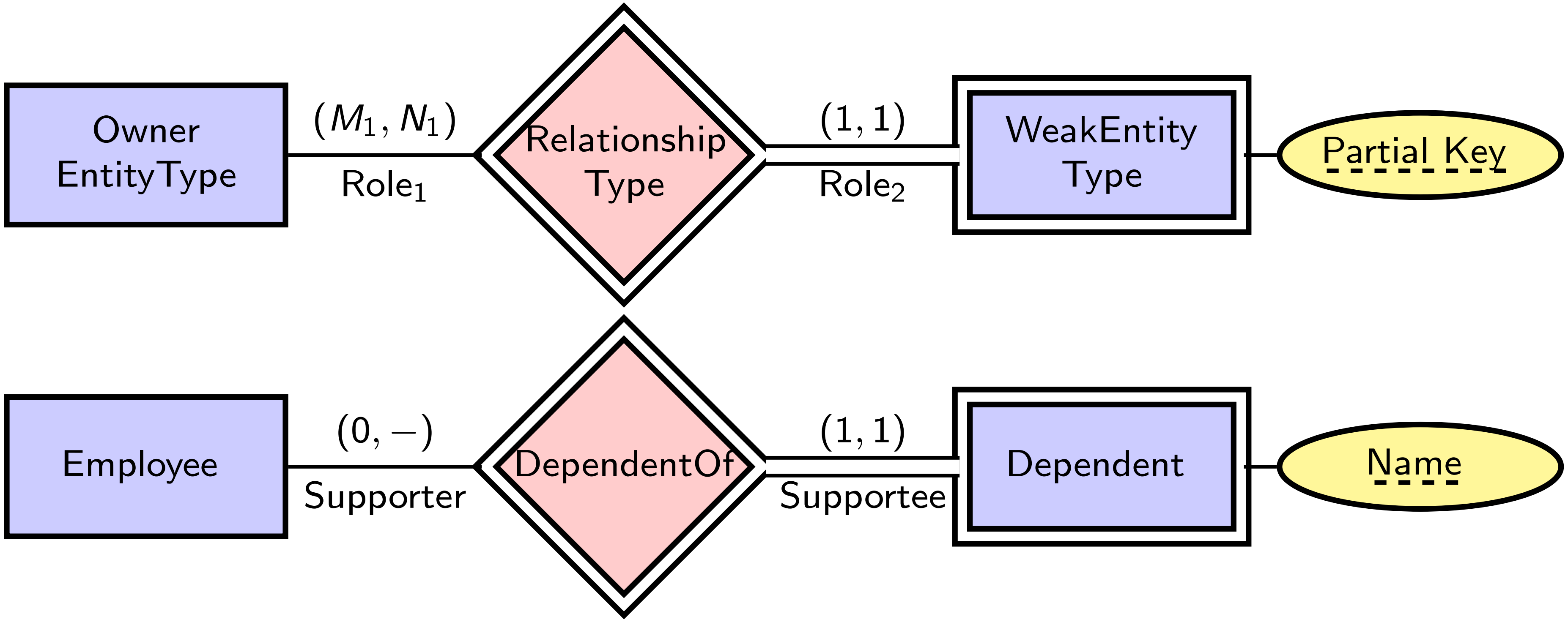
# Relationships May Have Attributes — 2




Modelling problem: Suppose that a student may have the same advisor over distinct time intervals.

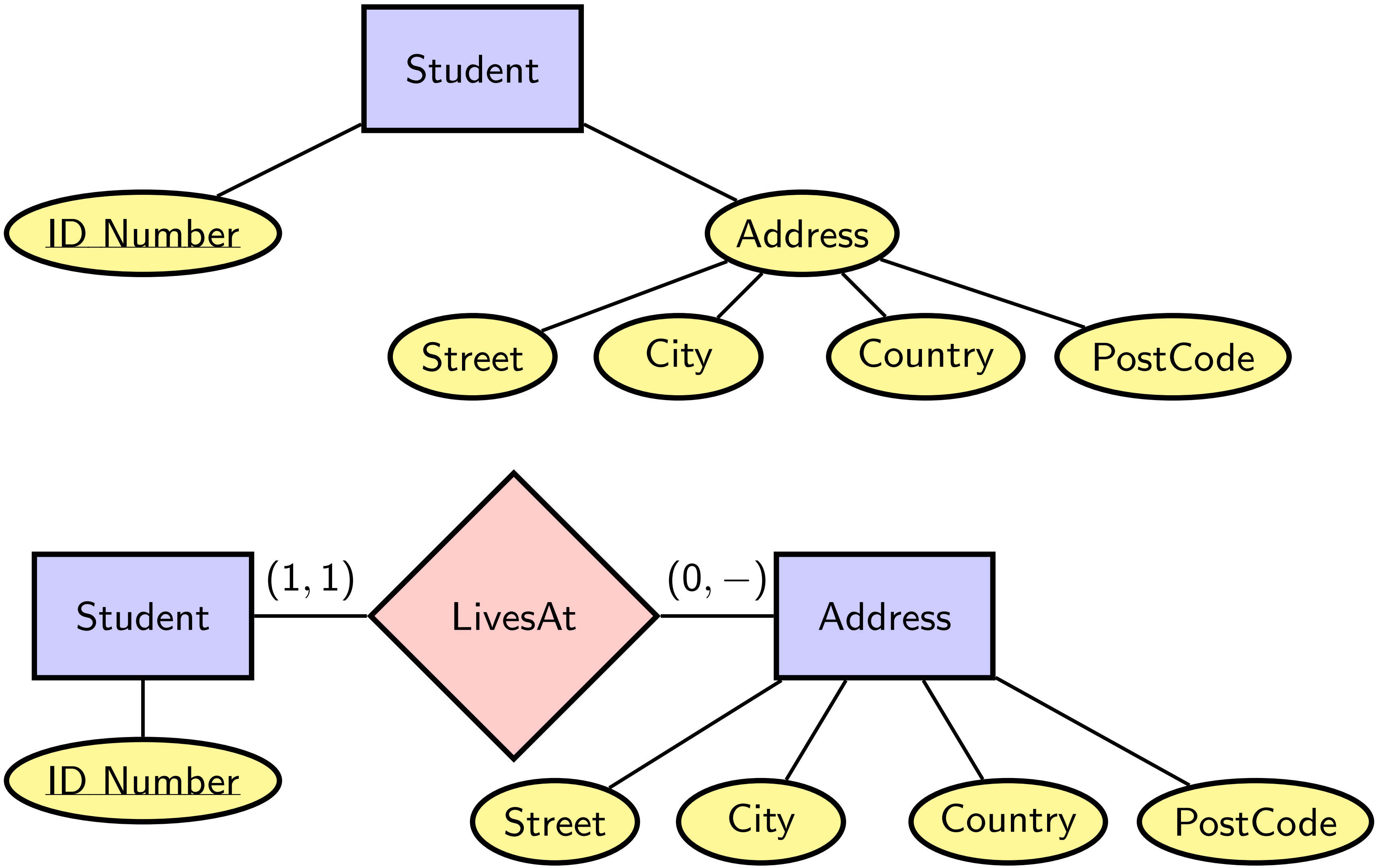
- No problem; just use a multivalued compound attribute.

# Weak Entities and Identifying Relationships



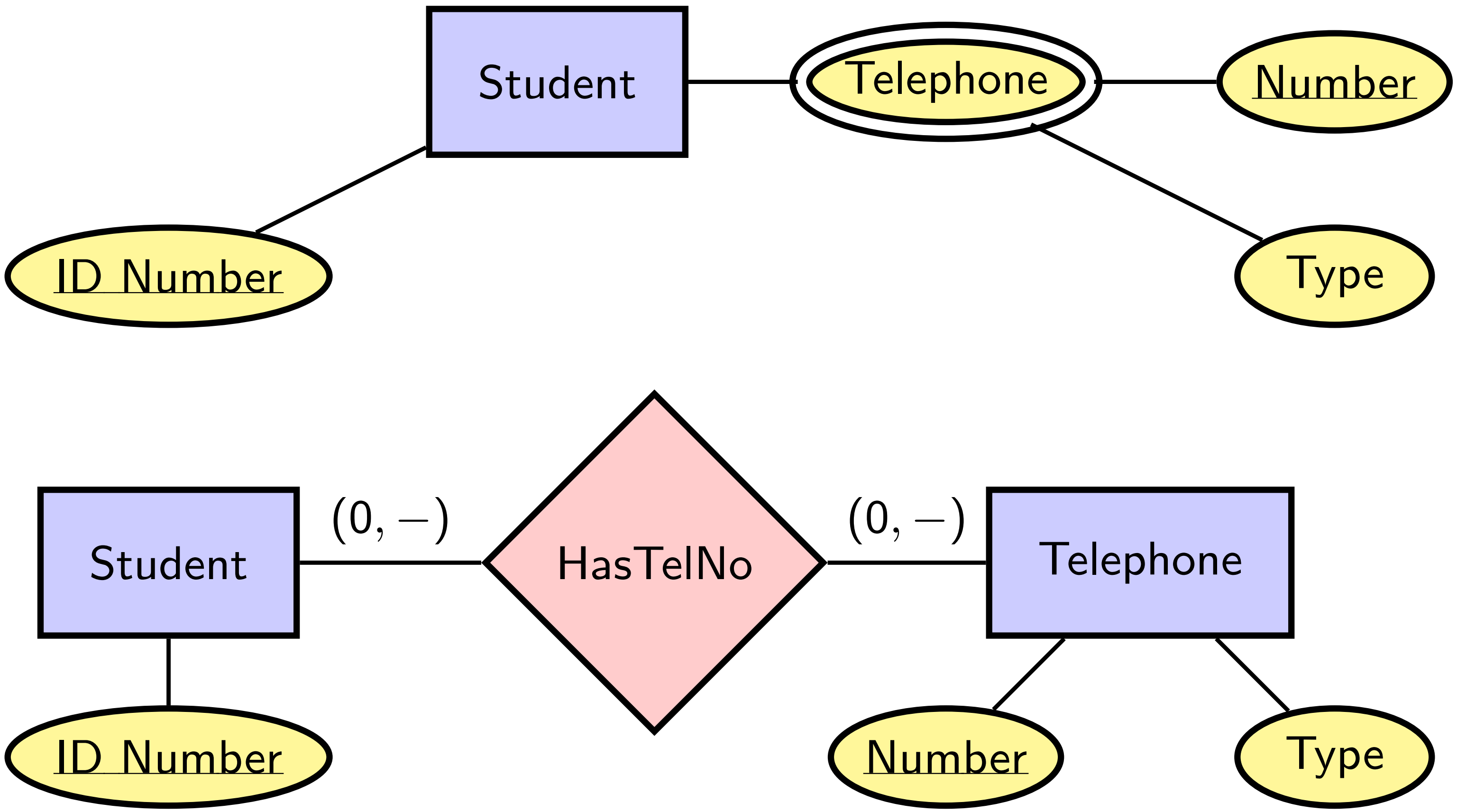

- A *weak entity type* is one which must get part of its key from another entity type, called the *owner entity type* or *identifying entity type*.
- The *partial key* is indicated by a dashed underline.
- The association to the relationship with the entity which completes the key is shown with double lines on both the entity and the relationship.
- Note that a weak entity only makes sense with total participation, so this notation is consistent with the alternate one discussed earlier.

# Choices in Design




- It is possible to use a relationship instead of a compound attribute.
- But this has implications in the translation to a relational schema.

# Choices in Design — 2




- There is often a choice between using a multivalued attribute and using a relationship.

# Conversion of an ER Specification to a Relational Schema

- The conversion process has a procedure for each type of construction.

  Conversion of regular entities:

  Conversion of weak entities:

  Conversion of one-to-one binary relationships:

  Conversion of many-to-one binary relationships:

  Conversion of many-to-many binary relationships:

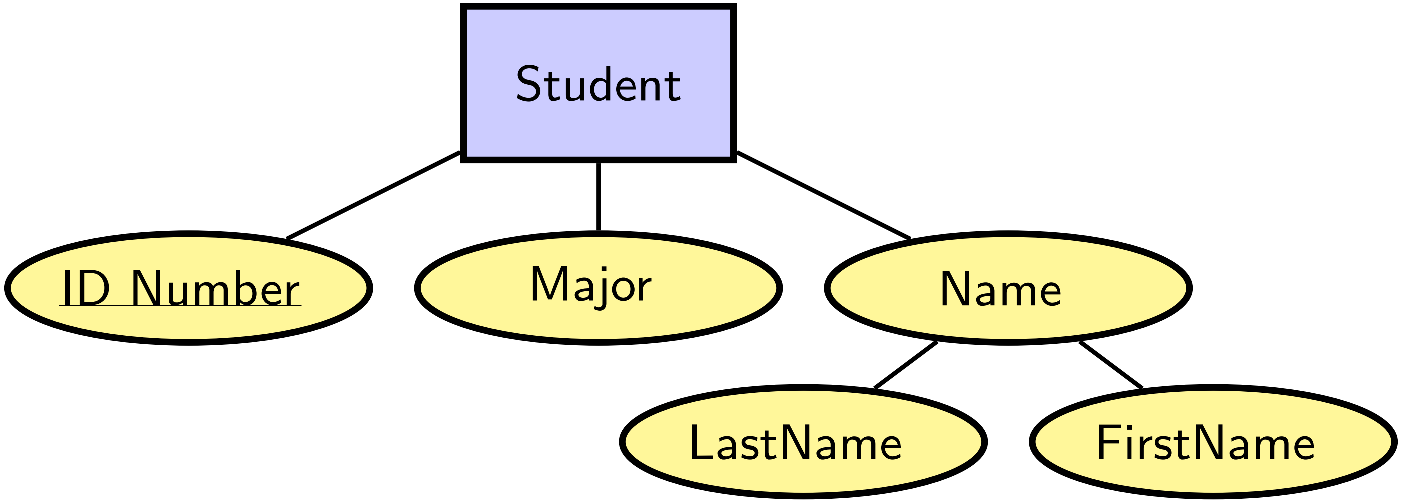
  Conversion of ternary and higher relationships:

  Conversion of multivalued attributes:

- The rules are applied in “bottom-up” fashion.

- Each will be illustrated via a simple example.

# Regular Entity Conversion



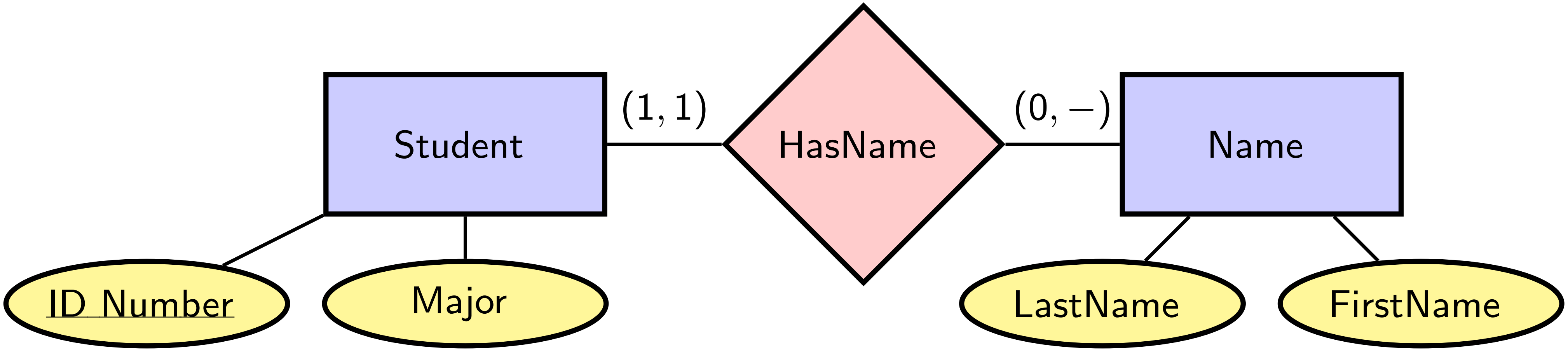

- Create a relation with one attribute for each *simple* attribute of the entity.

Student

| <u>ID_Number</u> | Major | LastName | FirstName |
|---|---|---|---|

- Compound attributes are lost.
- With a relational model which supports subtuples (non-1NF), this construction may be extended in the obvious fashion.

# Regular Entity Conversion — 2




- If the name was modelled as a separate entity, then this construction must be applied separately to each entity.

Student

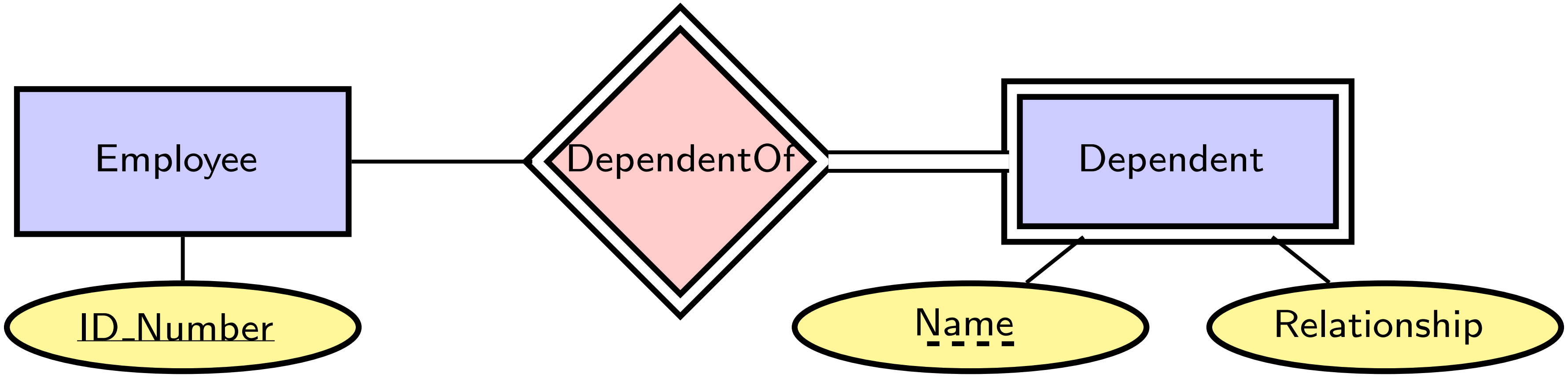
| <u>ID_Number</u> | Major |
|---|---|

Name

| LastName | FirstName |
|---|---|

- The combination of these two relations requires the step for relationships, to be described.

# Conversion Involving Weak Entity Types



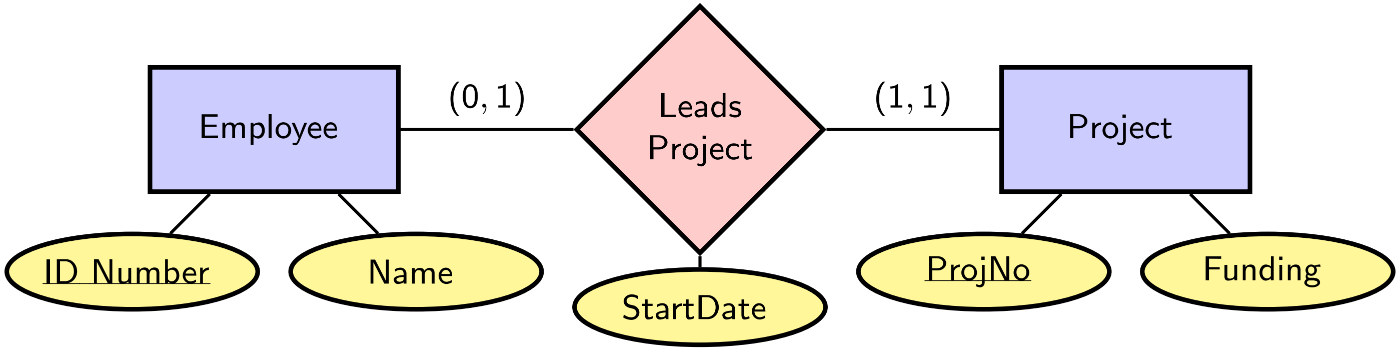

- In the case of a weak entity type, the key of the owner entity type must be added.

Dependent

| ID_Number | Name | Relationship |
|---|---|---|

- The partial key of the weak entity type together with a key of the owner entity type constitute the key of the relation.

# One-to-One Binary Relationship Conversion




- First the conversion of each entity type must be completed:

Employee

| ID_Number | Name |
|---|---|

Project

| ProjNo | Funding |
|---|---|

- The key of one relation is added as a foreign key to the other relation and the attributes of the relationship type are added:

Employee

| ID_Number | Name | LeadsProj | StartDate |
|---|---|---|---|

Project

| ProjNo | Funding |
|---|---|

- To allow the recapture the "not null" requirement of $(1,1)$, as well as to minimize the need for nulls, it is preferable to add the key from a $(0,1)$ relation to that of a $(1,1)$ relation whenever possible:

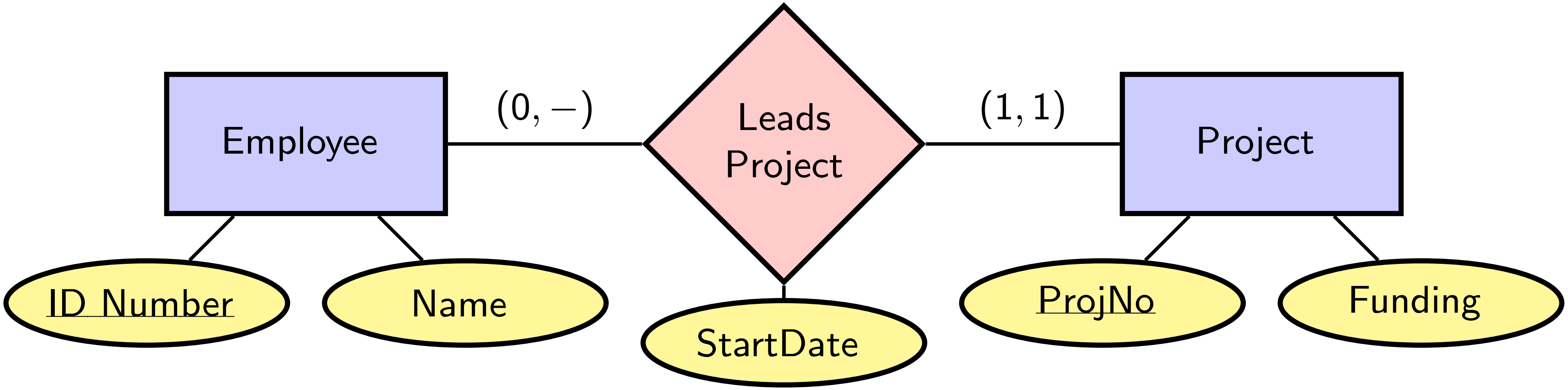
Employee

| ID_Number | Name |
|---|---|

Project

| ProjNo | Funding | LeaderID | StartDate |
|---|---|---|---|

# One-to-Many Binary Relationship Conversion




- Now suppose that an employee may lead several projects.
- First the conversion of each entity type must be completed:

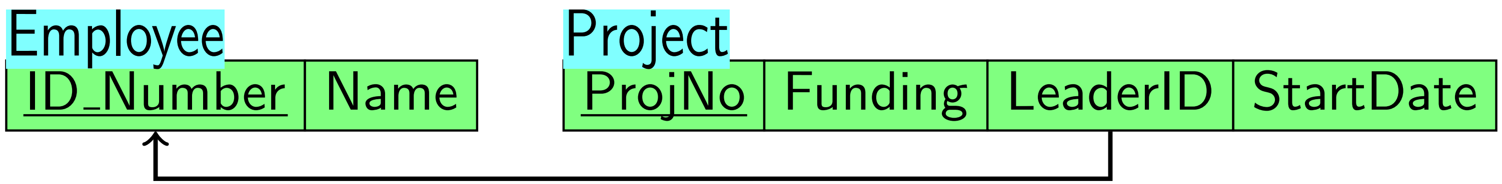
Employee

| <u>ID_Number</u> | Name |
|---|---|

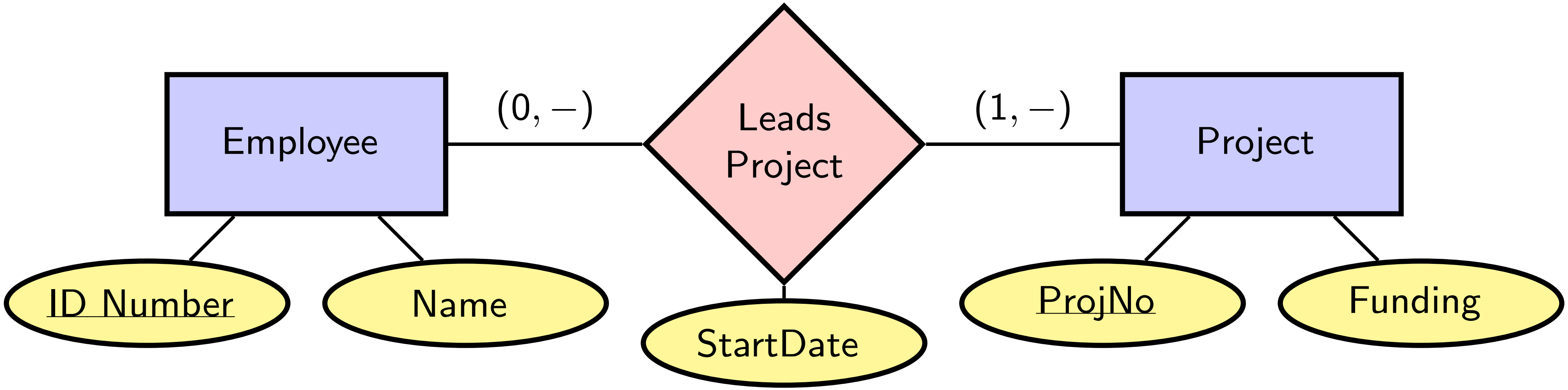
Project

| <u>ProjNo</u> | Funding |
|---|---|

- In this case, the only option is to add the key from the $(0, x)$ side $(x > 1)$ to that of the $(0/1, 1)$ side (plus the attributes of the relationship type):



# Many-to-Many Binary Relationship Conversion




- Now suppose that a project may have several leaders as well.
- First the conversion of each entity type must be completed:

Employee

| ID_Number | Name |
|---|---|

Project

| ProjNo | Funding |
|---|---|

- In this case, the solution is to create a new relation with keys from both entity types, as well as the attributes of the relationship type:

Employee

| ID_Number | Name |
|---|---|

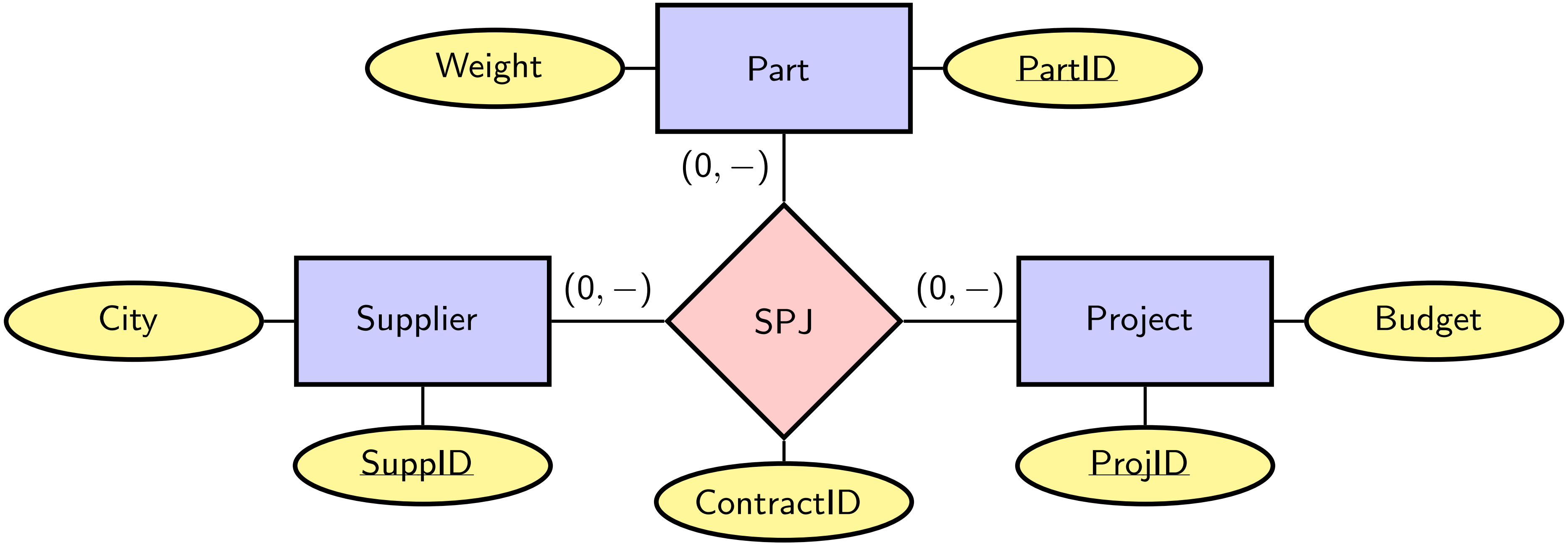
Project

| ProjNo | Funding |
|---|---|

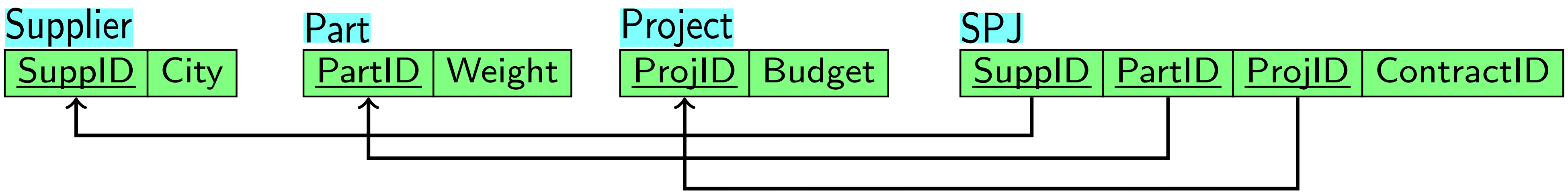
LeadsProject

| ID_Number | ProjNo | StartDate |
|---|---|---|

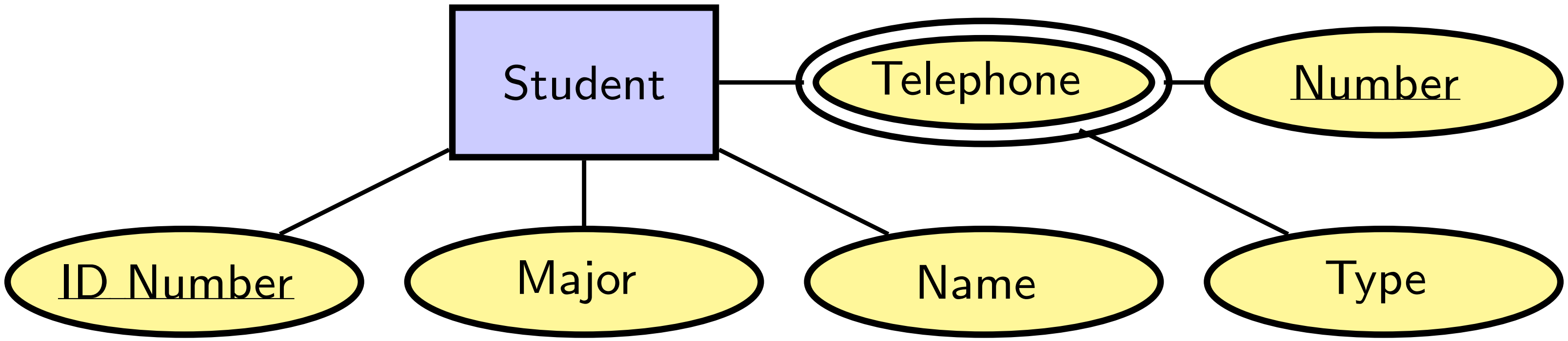
# Many-to-Many Ternary Relationship Conversion



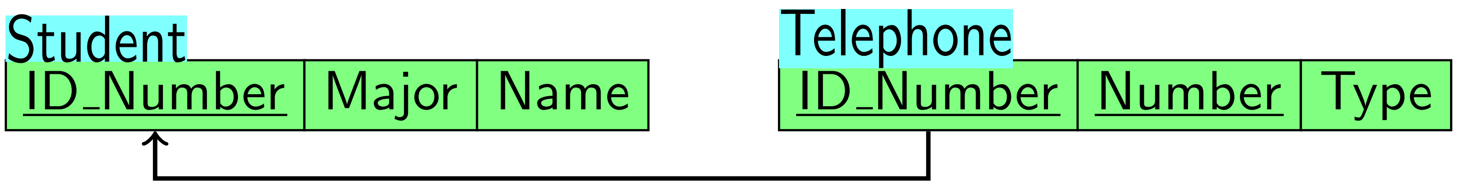

- In this case, in addition to a relation for each entity type, there is a relation containing the keys from each entity type.



# Conversion for Multivalued Attributes



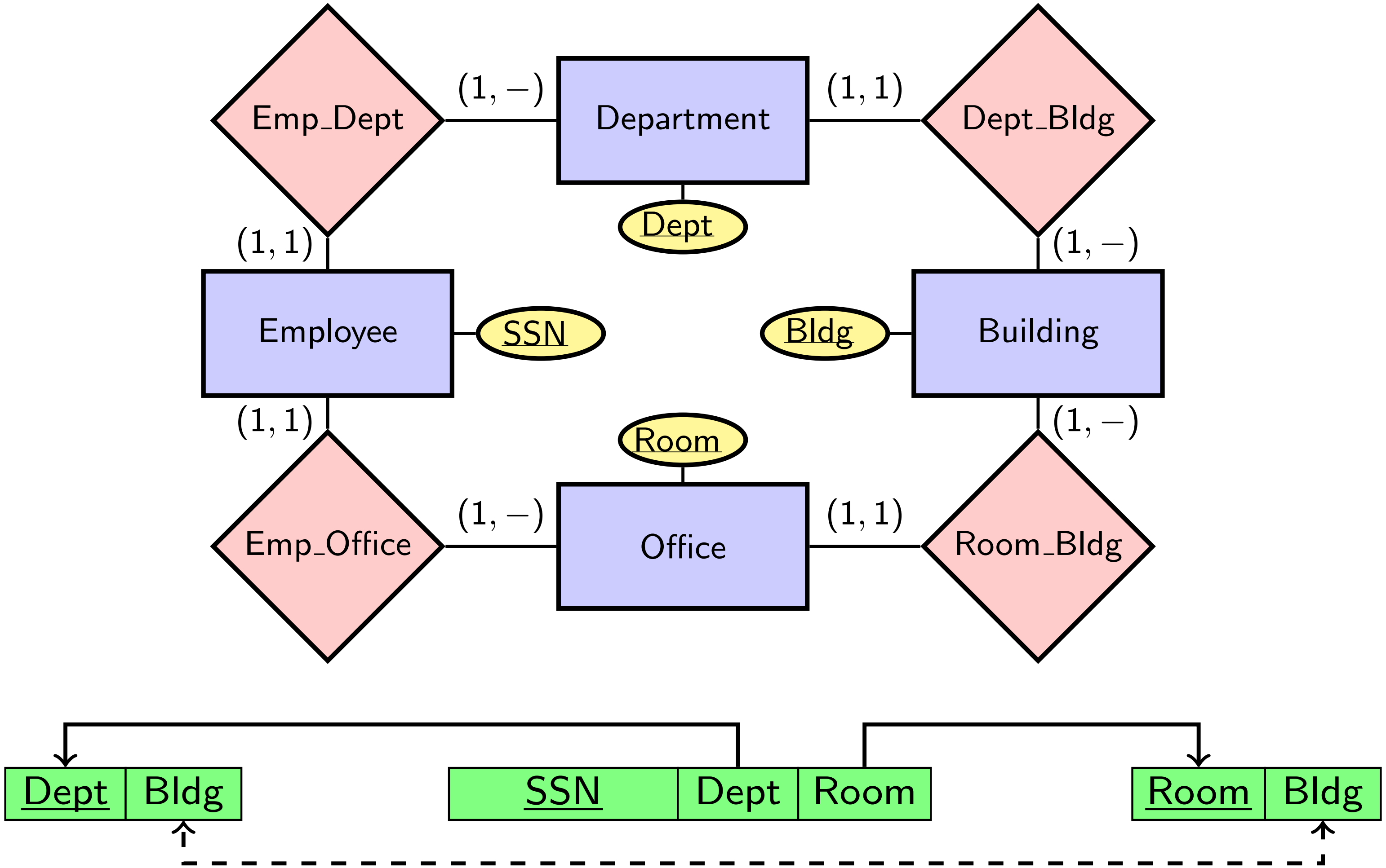

- A multivalued attribute requires a separate relation containing the key of the entity type together with the multivalued attribute and all of its subattributes.




- The key is the combination of the key of the entity type and the key of the multivalued attribute.

# ER and Non-Independent Designs

- If the ER design itself is cyclic, this will be reflected in the relational realization as well.




- In this case, it is perhaps the design itself which should be reconsidered and modified.