Databases

The Entity Relationship (ER) Model

Dr Konrad Dabrowski
konrad.dabrowski@durham.ac.uk

Online Office Hour:
Mondays 13:30–14:30
See Duo for the Zoom link

Why a Data Model?

- A model:
  - an abstract representation of something that exists in the real world
- Models help to make complex things understandable
  - e.g. architects use 3D-models to give an impression of the building
- In Databases:
  - Data Definition Language (DDL) is too low-level
  - not easily understandable by most of users
- We have seen:
  - Relational Data Model (logical design of the DB)
  - relations between data ⟶ stored in tables
  - based on the concept of mathematical relations

Entity-Relationship (ER) model

- Relational Data Model:
  - sometimes still too low level for big companies:
  - designers / programmers / users understand data and its use in different ways!
  - we need a model of communication that is non-technical
    and free of ambiguities

⇒ Entity-Relationship (ER) model
  - graphical description of the DB (conceptual design of the DB)

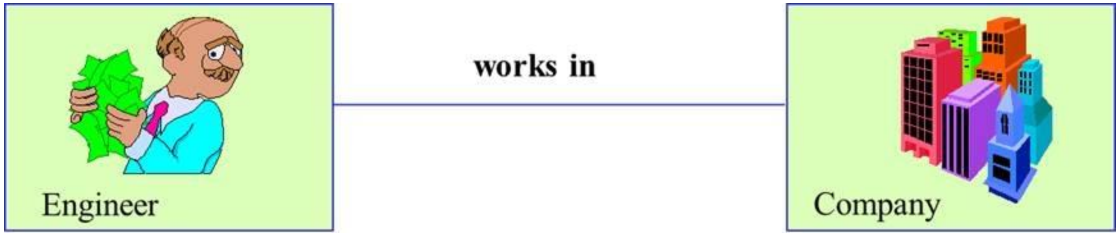
- A top-down approach to database design:
  - start with a set of requirements (informal system description)
  - identify the types of "things" that you need to represent data about
  - identify the attributes of those "things"
    (which will later become attributes in the tables)

Entity-Relationship (ER) model

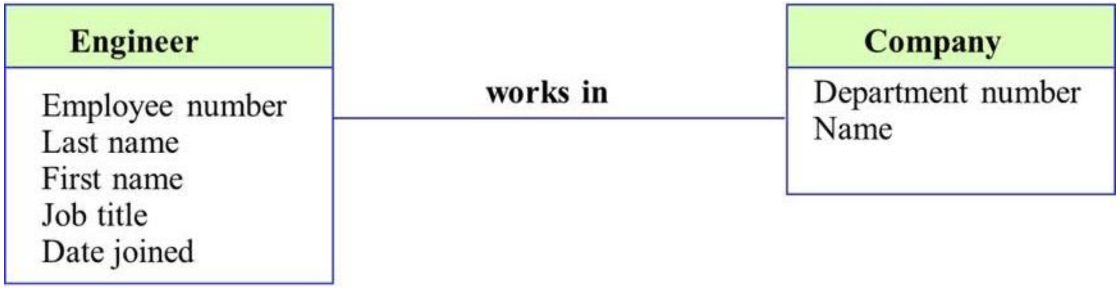
- Objective of the ER model:
  - to help understand the nature & relationships among the data
  - to help derive the tables in the Relational Data Model
    (i.e. the logical design of the DB)

- Most important before building the ER model:
  - proper understanding of the problem domain (the scenario)
  - lack of understanding → wrong ER model → wrong DB design!

scenario:



ER model:



Engineer

Employee number
Last name
First name
Job title
Date joined

works in

Company

Department number
Name

Entity-Relationship (ER) model

- Basic concepts:
  - the important data objects (entities)
  - the important properties of the entities (attributes)
  - the associations between the entities (relationships)

- Furthermore:
  - constraints on the entities, relationships and attributes

- Several notations for representing the ER model:
  - Crow's foot notation
  - UML notation (Unified Modeling Language)

Main components of an ER model

- An entity:
  - any real thing (or abstract notion) that we recognize as a separate concern within the database
  - important enough to be represented separately

- Examples:
  - in a university DB: a student, a college, a course, a lecturer etc.
  - a bank stores data about you ⇒ you are an entity
  - a business stores an invoice ⇒ the invoice is an entity
  - a library stores data about a particular book ⇒ the book is an entity

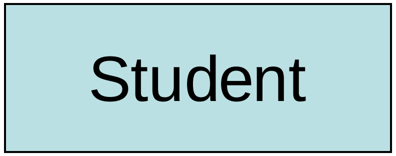
- Diagrammatic representation of entities:
  - a rectangle labelled with the name of the entity

Student

College

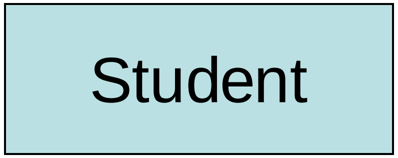
Main components of an ER model

- What does “entity” mean?
  - – are all students the same entity called “Student”?
  - – or is every student a different entity?


Student

- Entity type:
  - – the group of all objects with the same properties, identified as having independent existence, e.g. “Student”
- Entity occurrence:
  - – a uniquely identifiable instance of an entity type,
    e.g. a specific student called “James Smith”
- Example:
  - – a bank stores data about all the people who do business with them (entity occurrences)
  - – it refers to them generically as “customer” (entity type)

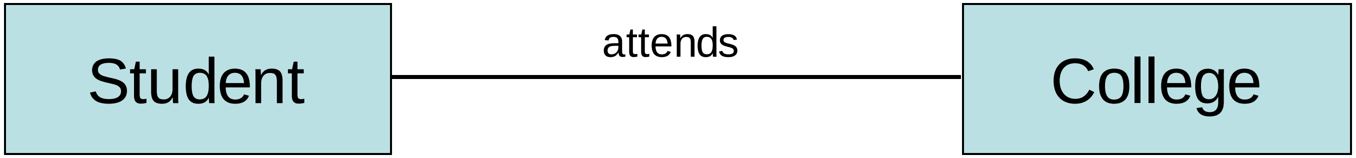
Main components of an ER model



- In an ER model:
  - a rectangle represents an entity type (i.e. a set of entity occurrences)
  - we use “entity” for both notions, whenever it is clear from the context
  - entities are associated with each other via relationships

- Relationship:
  - a named association between two entity types, which has some meaning in the context of the database, e.g.
  - “attends” is a relationship between “student” and “college”
  - “is citizen of” is a relationship between “person” and “country”

- Diagrammatic representation of a relationship:
  - a labelled line connecting the two entities



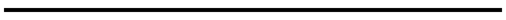
Cardinality

- Cardinality of a relationship:
  - the number of entity occurrences that are related to a single occurrence of an associated entity type through this relationship
- The cardinality of a relationship can be:
  - one-to-one (1:1),
    e.g. “director manages school”
  - one-to-many (1:*),
    e.g. “school employs teacher”
  - many-to-many (*:*),
    e.g. “teacher teaches child”
- Do we need many-to-one (*:1) relationships?
  - we can always rewrite it into a “one-to-many” (1:*) relationship:

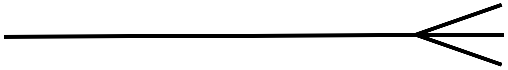
“teacher works in school” (*:1)
is the same as: “school employs teacher” (1:*)

Cardinality

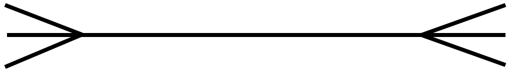
- Diagrammatic representation of cardinalities:
  - crow’s foot notation *(Chen, 1976)*
  - the crow’s foot is used to indicate the ‘many’ end of the relationship



one-to-one (1 : 1) relationship



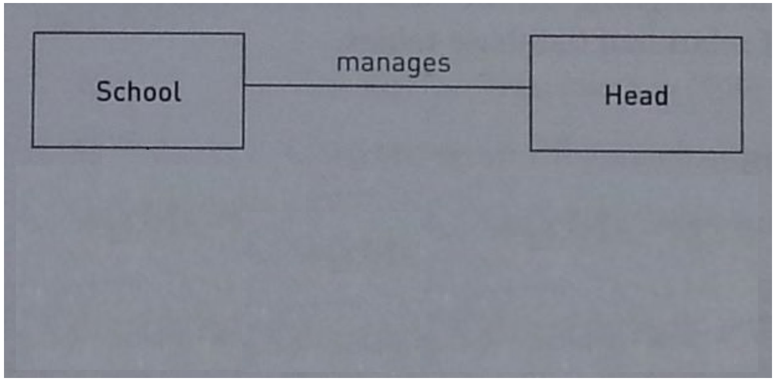
one-to-many (1 : *) relationship



many-to-many (* : *) relationship

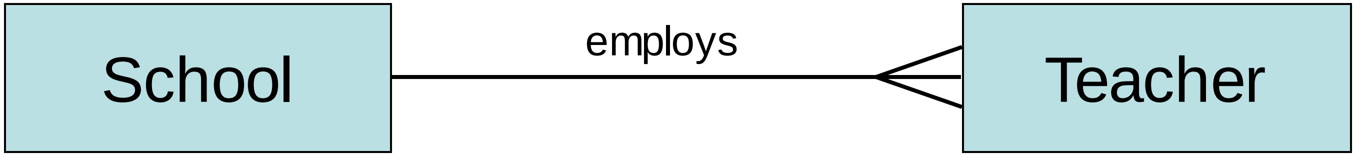
Cardinality

- one-to-one (1:1) relationship:
  - The relationship “manages” between “Head” and “School” means:
    - “one Head manages one School” and “one School is managed by one Head”
- one-to-many (1:*) relationship:
  - The relationship “employs” between “School” and “Teacher” means:
    - “one School employs many Teachers” but “one Teacher is employed by one School”
- many-to-many (*:*) relationship:
  - The relationship “teaches” between “Teacher” and “Child” means:
    - “one Teacher teaches many Children” and “one Child is taught by many Teachers”



Optionality and participation

- Consider this one-to-many (1:*) relationship:



"one School employs many Teachers"

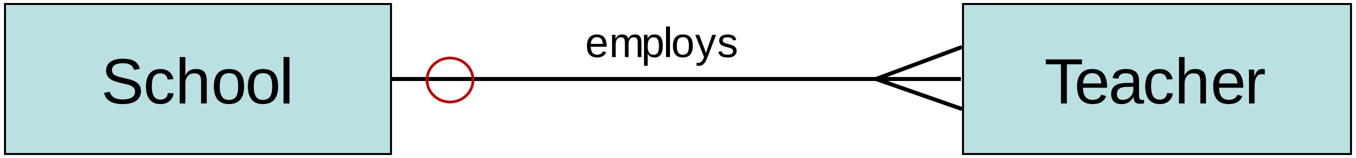
and "one Teacher is employed by one School"

- What if some teacher is not employed by any school?

"one Teacher is employed by one School or by zero Schools"

  – from the point of view of the Teacher, this relationship is optional

  – in the ER diagram: optionality is denoted by a circle

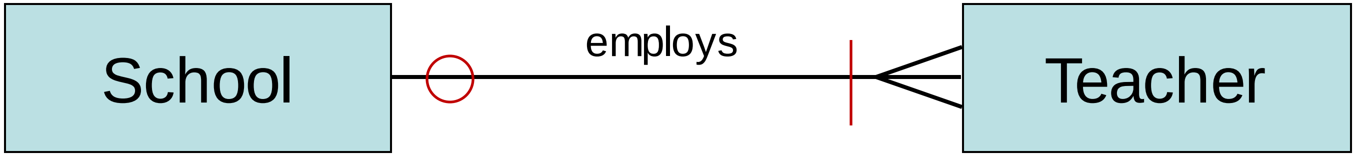


Note: on the relationship line, this circle is at the opposite end from the entity ("Teacher") concerned in the optionality

Optionality and participation

- If an entity participates optionally in a relationship:
  - it has a partial participation
  - otherwise it has total participation
    (i.e. it is associated to at least one occurrence of the other entity type)

- In the ER diagram:
  - total participation is denoted by a vertical bar
  - again, on the opposite side of the relationship line

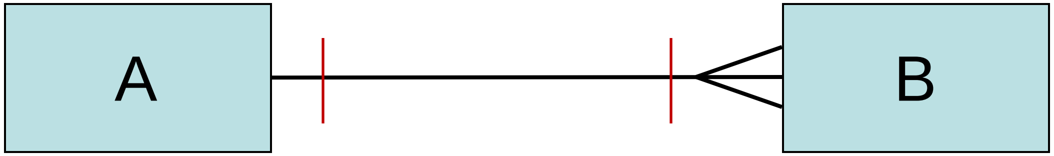


“one School employs at least one Teacher”

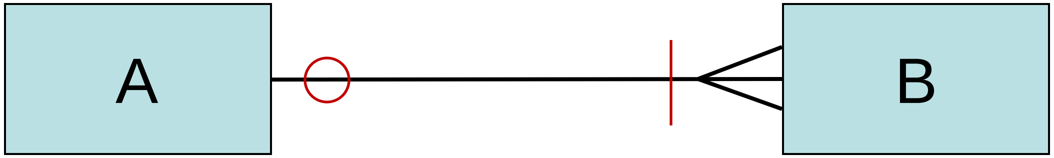
and “one Teacher is employed by one or zero Schools”

Optionality and participation

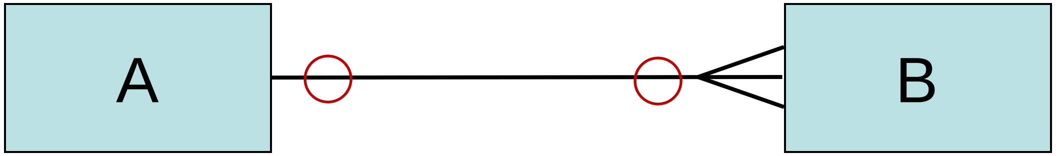
- In general:



(1: *) relationship with
total participation
for both entities A and B



(1: *) relationship with
optional participation for B
and total participation for A



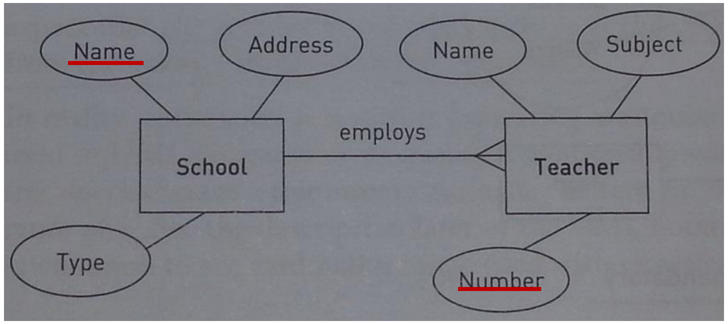
(1: *) relationship with
optional participation
for both A and B

Attributes

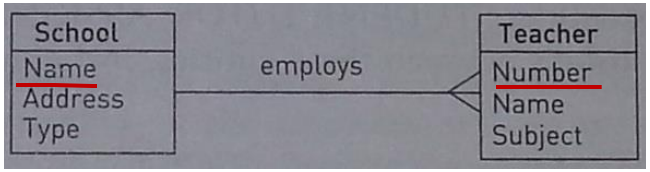
- The attributes of an entity:
  - the set of all common characteristics that are shared by all entity occurrences of an entity type
  - e.g. an entity representing a person may have attributes such as Name, Age, Sex etc.

- Diagrammatic representation of attributes:
  - Primary keys are always underlined

1. for every attribute one labelled ellipse attached to the entity rectangle



2. all attributes in the lower part of the entity rectangle (more neat alternative)



Determining the attributes

Which are the appropriate attributes?

- We first look for the natural characteristics of an entity
  - e.g. a Staff entity could have the attributes:
    “staff number”, “first name”, “surname”, “address”, “salary”, ...
- An attribute can be associated with a value domain
  - e.g. staff number is made up of 10 digits – 0000123479
- Simple attribute:
  - it has only one single component – (cannot be further divided)
  - e.g. age, sex, marital status
- Composite attribute:
  - it has multiple components
  - e.g. address can be subdivided into street, city, postcode

Determining the attributes

Which are the appropriate attributes?

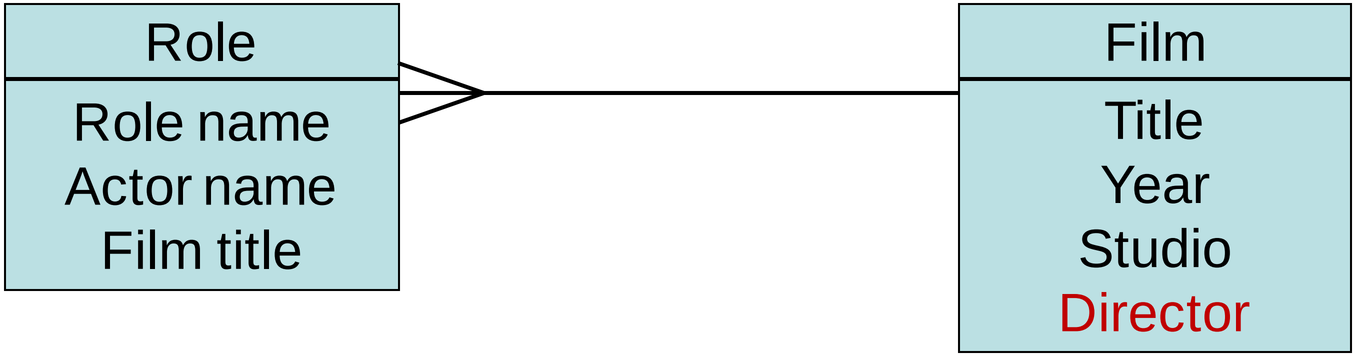
- Single Value attribute:
  - it holds only one value
  - e.g. the serial number of a manufactured part

- Multivalued attribute:
  - it can hold many values
  - e.g. several university degrees for a person;
    many telephone numbers in an office

- Derived attribute:
  - derivable from value of a related attribute
  - e.g. age can be derived from date-of-birth
    (which is stored somewhere else)

Determining the attributes

- Sometimes confusion between entities and attributes:
  - the name of the film director can be an attribute of “Film”



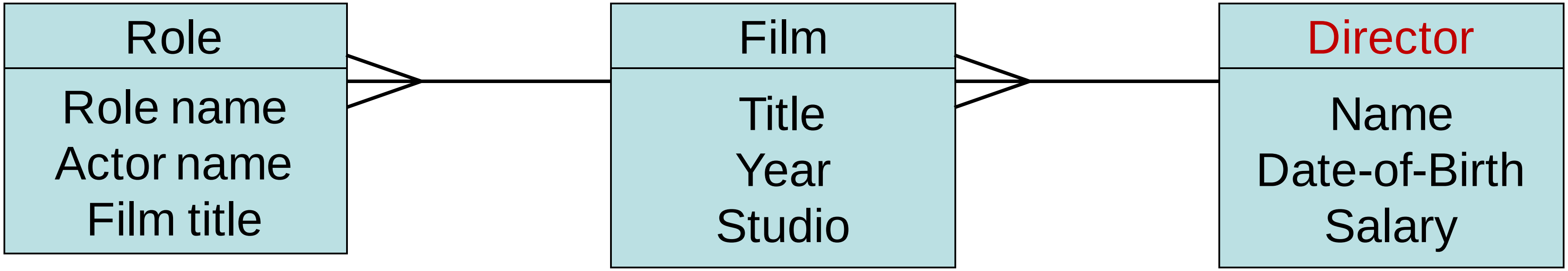
Role

Role name
Actor name
Film title

Film

Title
Year
Studio
Director

  - or it can be a separate entity,
    e.g. if we want to store more information about the Director
    (age, salary etc.)



Role

Role name
Actor name
Film title

Film

Title
Year
Studio

Director

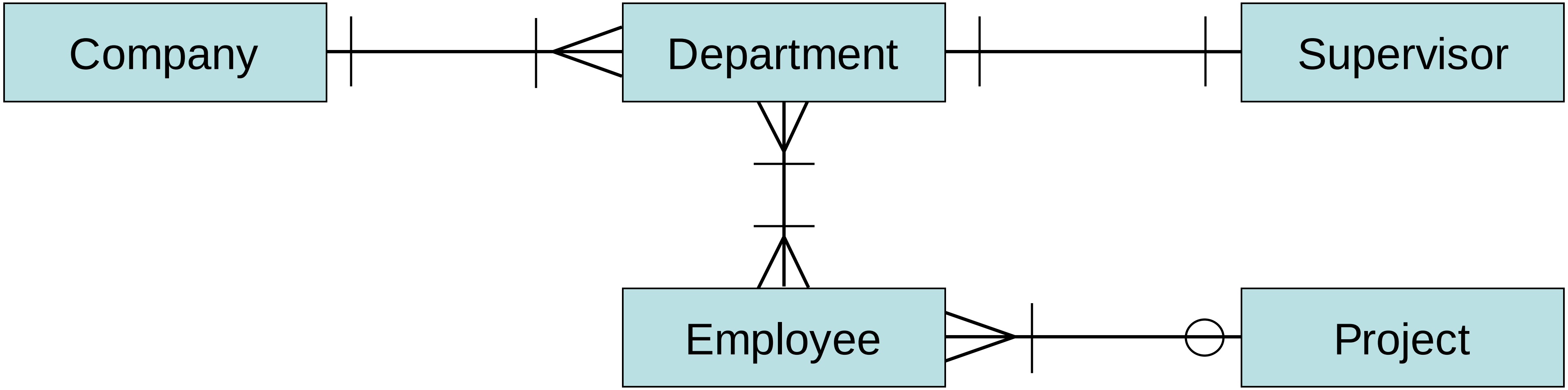
Name
Date-of-Birth
Salary

- Better a separate entity than a composite attribute!

An example

A company has several departments. Each department has one supervisor and at least one employee. Employees must be assigned to at least one, but possibly more departments. At least one employee is assigned to a project, but an employee may be on vacation and not assigned to any projects.

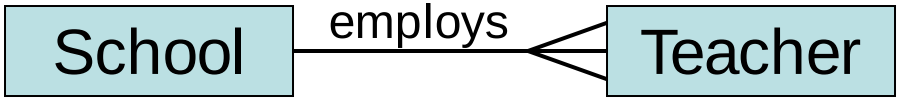
- Which are the likely entities in the above scenario?



- Or to include “Company” and “Supervisor” as attributes of “Department” ?

Many-to-Many Relationships

- Remember our final target:
  - to create a Relational Model from the ER model, where entities ⟷ tables
- If we have an (1:*) relationship:

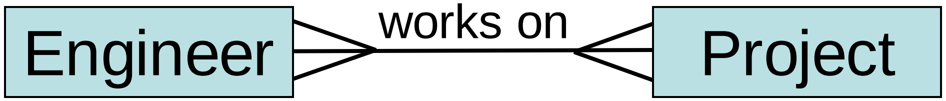


School

employs

Teacher

  - we map the foreign key of Teacher to the primary key of School
- If we have a (*:*) relationship:

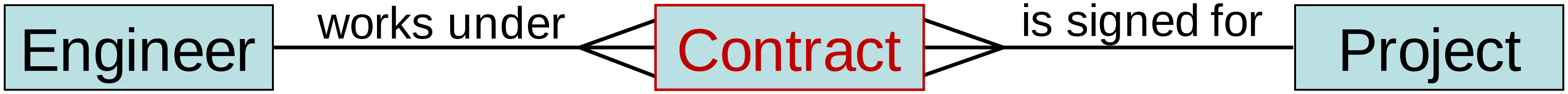


Engineer

works on

Project

  - we cannot map foreign keys to primary keys!
- The only solution is to introduce a new entity:



Engineer

works under

Contract

is signed for

Project

- This process is called: “resolve many-to-many relationships”

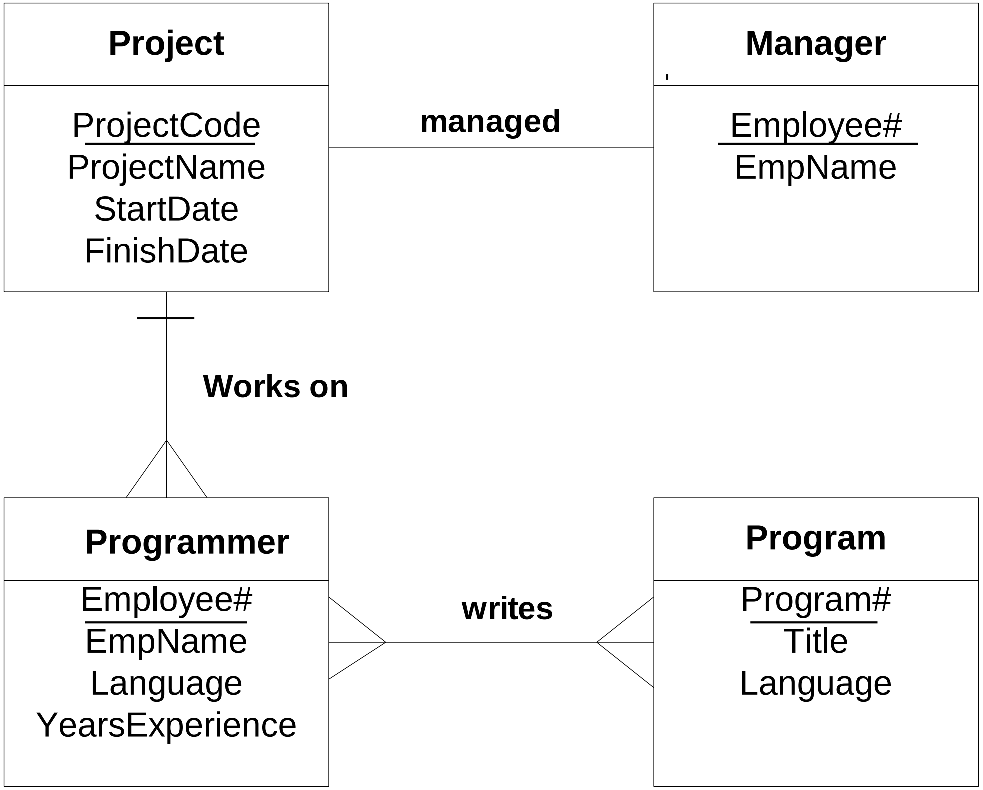
Step-by-step construction of an ER Model

1. Identify entities from the scenario/real world
2. Find relationships
3. Draw rough ER diagram (i.e. an ER model)
4. Fill in the relationships
5. Define primary keys and resolve many-to-many relationships
6. Identify attributes and map them to the entities
7. Draw full ER diagram showing keys, attributes, relationships
8. Check it! Does it reflect the real-world system?

- Moral to the story …
  - do not expect to get it right first time
    an ER diagram usually needs improvement
  - it gets easier with practice!

Converting ER model into a Relational model

A software company employs a number of programmers who are assigned to work on specific projects. Each project is controlled by one manager. Each programmer works on only one project, but contributes to the writing of several programs. Each program many be written by one or more programmers.



Construct a Relational model from an ER model:

- Resolve the many-to-many (*:*) relationships
- Build relational tables from the entities in the ER diagram
- Map foreign keys with primary keys