TÉCNICO LISBOA

Sistemas de Informação e Bases de Dados

Class 05: Entity-Association Model (cont.)

Prof. Paulo Carreira

Class Outline

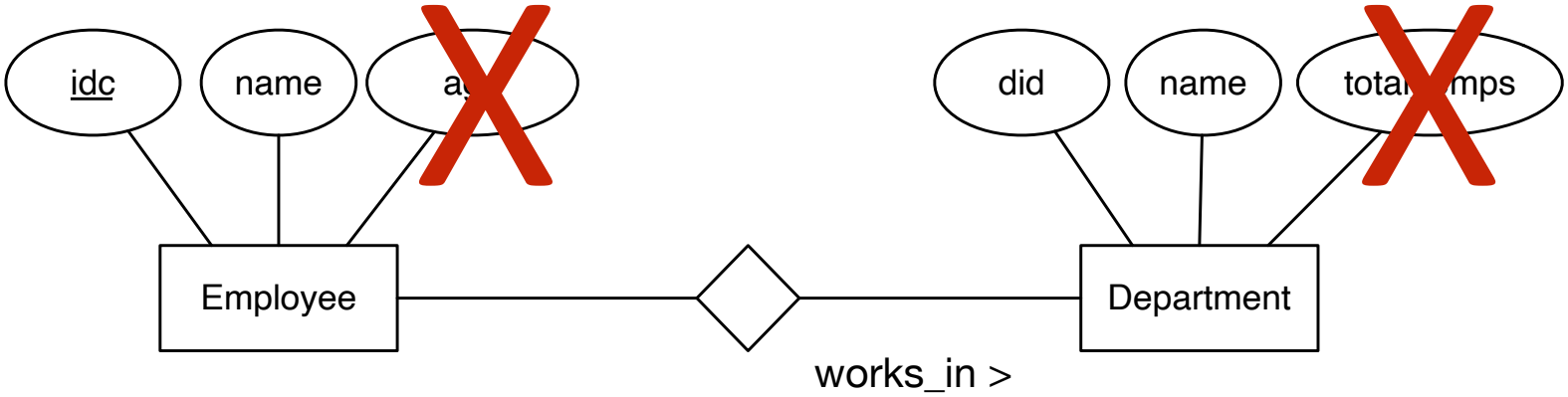
- Avoiding Modelling Mistakes
- Recursive Associations
- Integrity Constraints
- Weak Entities
- Applications of Weak Entities
- Aggregation
- Applications of Aggregation

Avoiding typical Modelling Mistakes

Typical Modelling Mistakes

- Derivable (computable) attributes: E.g. Age vs Birthdate
- Entities without Attributes or Primary keys
- Entities modelled as attributes: To encode implicit associations
- Using custom Integrity Constraints: That could be modelled using the E-A graphic language
- Not considering association circuits and the related associations

Derivable Attributes

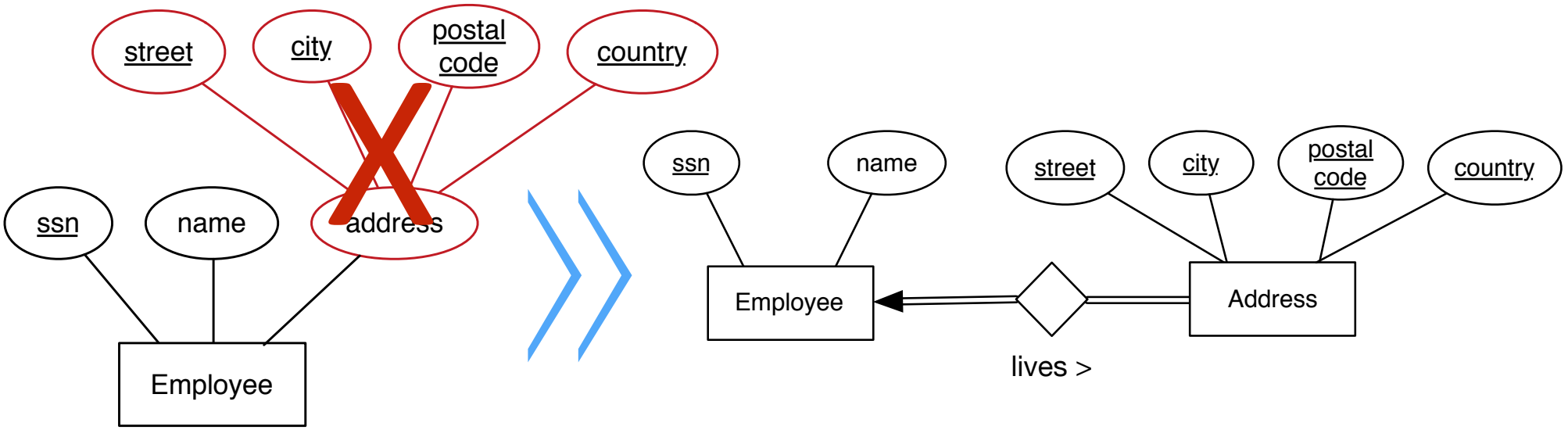


- Employee has the ‘age’ attribute that is derived from the birthdate (as time progresses this attribute would have to be updated)
- Department has the ‘total_emps’ attribute that can be computed (each time an employee is added to or removed from a department this attribute would have to be updated)

Referencing Attribute

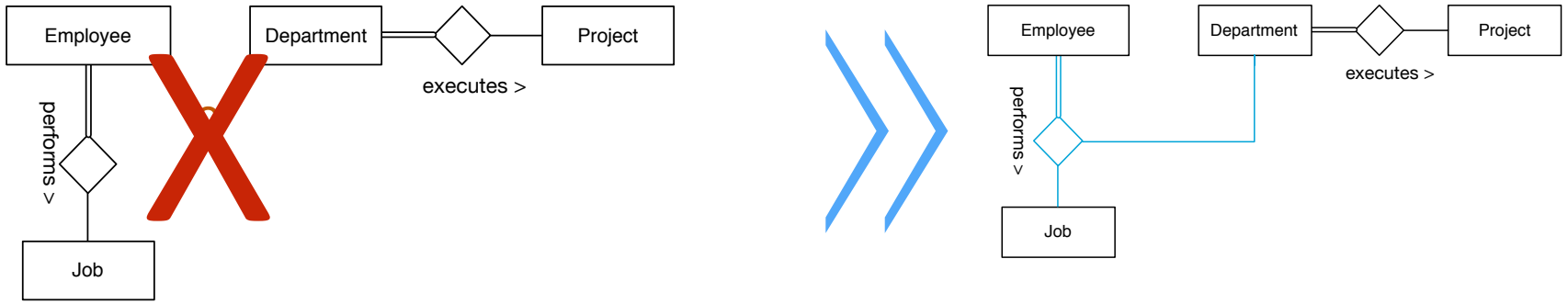
- The Manager entity has a ‘department’ attribute that references the Department where the manager works.
- Modelling a reference using an attribute is masking a an existing association (that will not be captured)

Atribute vs. Entity



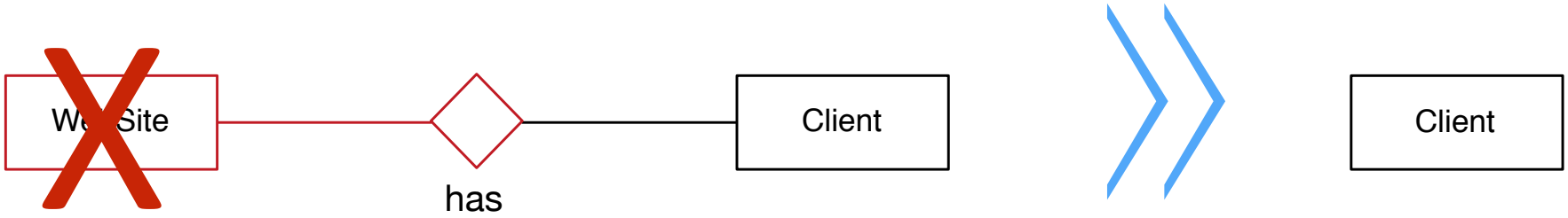
- An attribute can never consist of other attributes

Concept Islands



- A correct Entity-Relationship diagram is fully connected
- There can be no islands of concepts

Singleton Entity

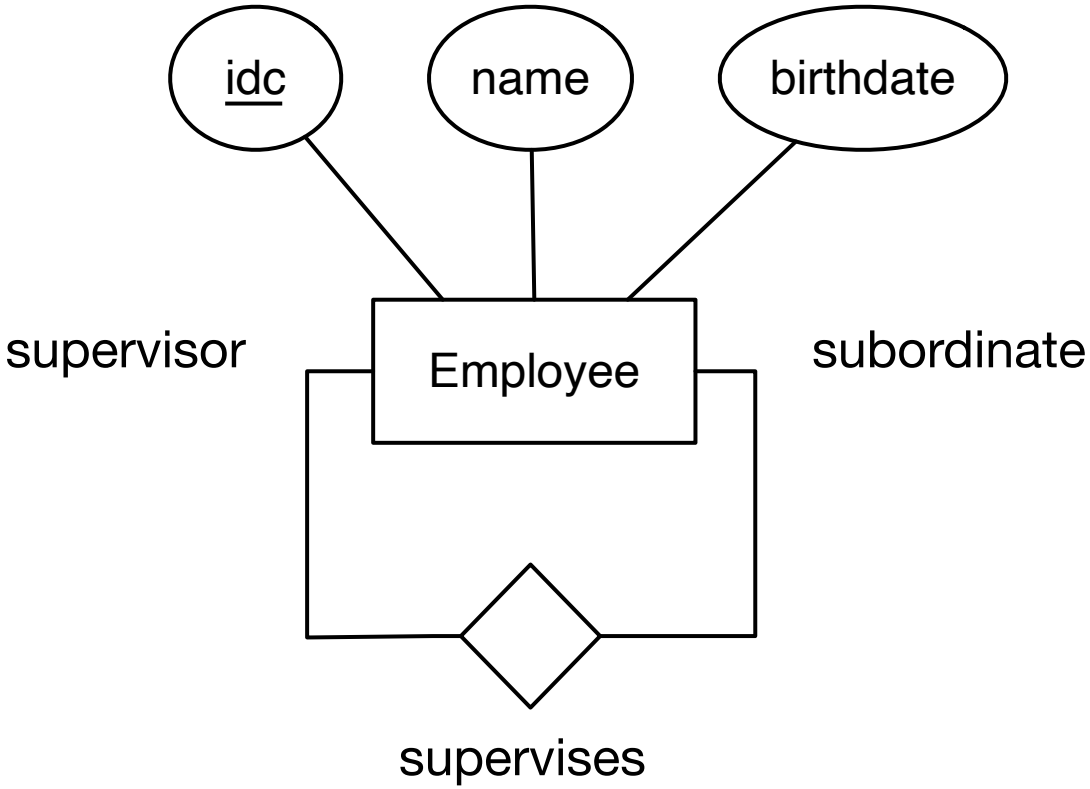


- A superfluous "System" singleton entity is included in the diagram.
- We are not recording information about Web Sites. Therefore, this entity is not should not be in the domain.
  - The entity 'Web Site' is superfluous because the the goal of the system is not to manage web sites. The 'Web Site' entity type will have only one instance.
  - Ask what are the attributes of WebSite and how many instance will there be. What is the key?

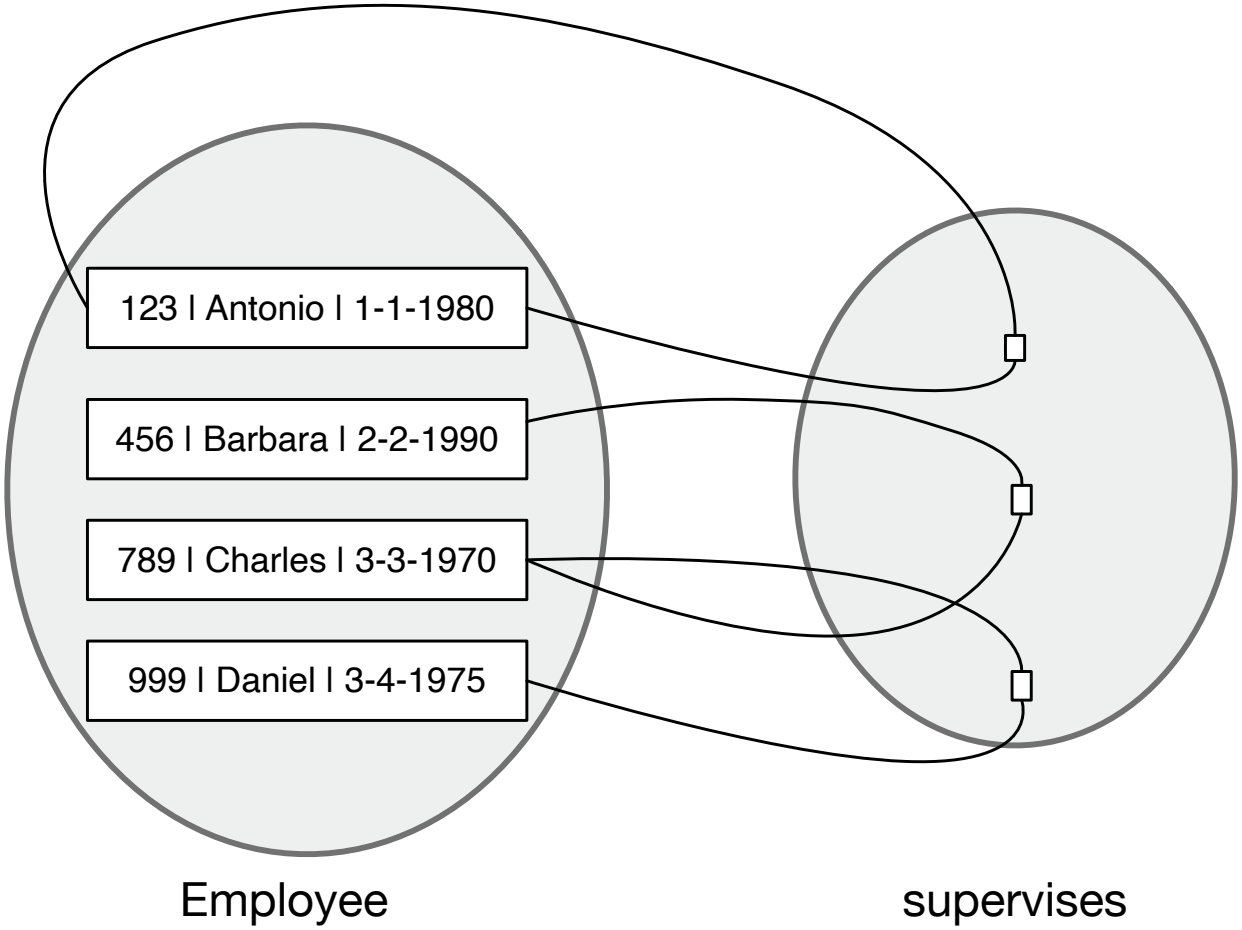
Recursive Associations (or auto-associations)

Roles

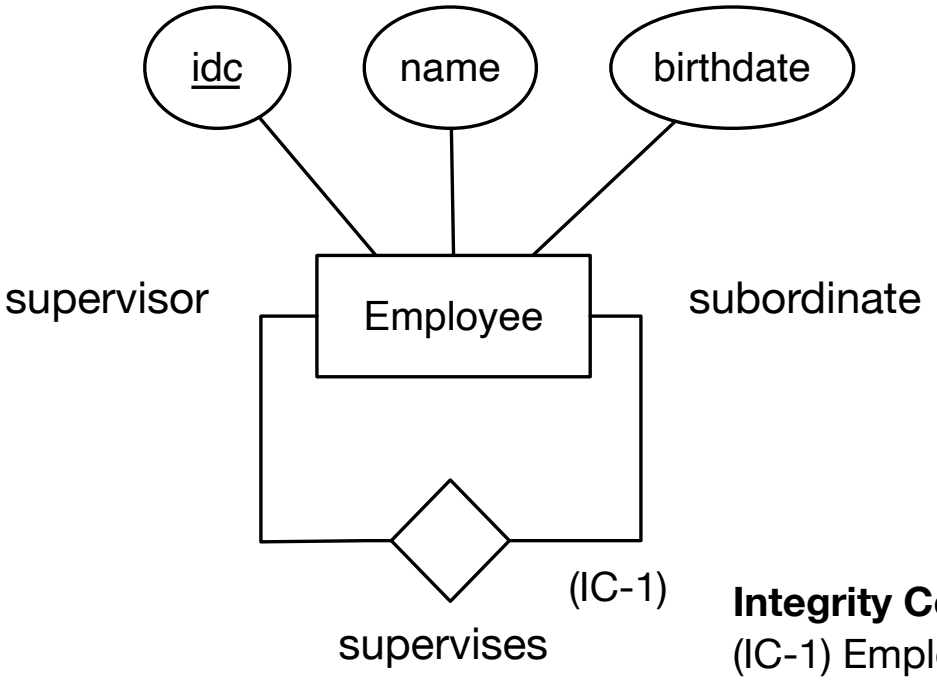
The sets of entities involved in an association are not necessarily distinct.



Auto-association Example



Auto-association with IC



Integrity Constraints:
(IC-1) Employees are not allowed to **supervise** themselves

Typical constraints for *self-associations*

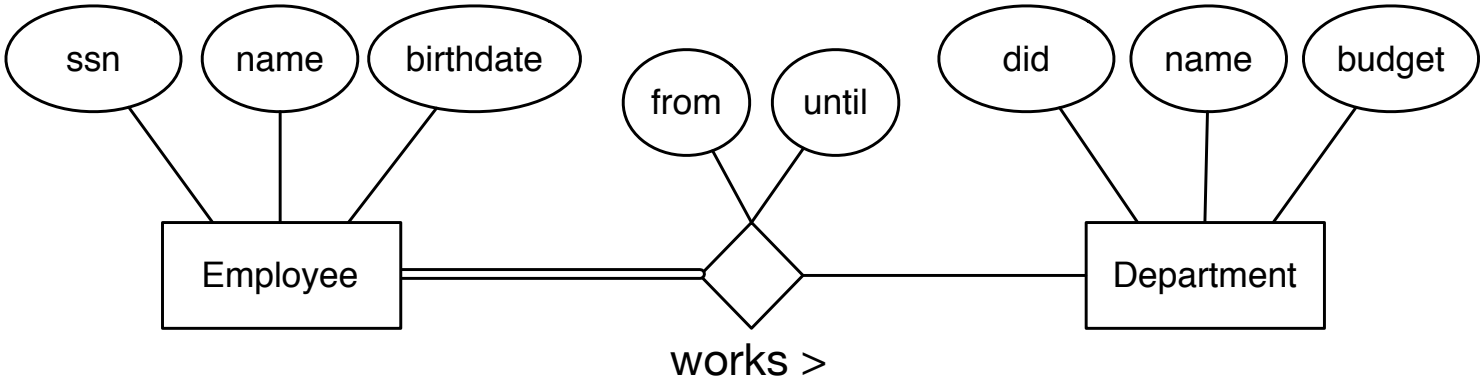
A number of typical restrictions apply to self associations

- Non-reflexivity: No employee can supervise himself.
- Non-circularity: Two employees can not supervise one another (or more generally: Circular supervision relationships can not exist).
- Depth limit: The chain of supervision can only be up to level *k* (where *k* is a concrete value dependent on the problem domain).

Integrity Constraints

Integrity Constraints

Used when a constraint cannot be modelled in E-A

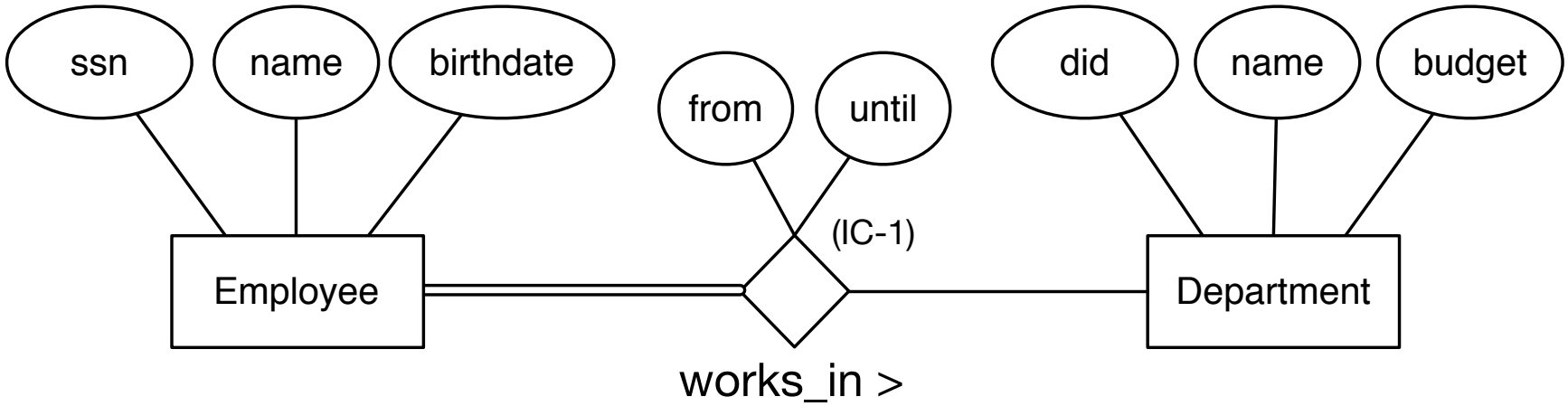


- Example:

  RI - 1: An Employee cannot **work** on a Department for a period of less that 3 months

Integrity Constraints

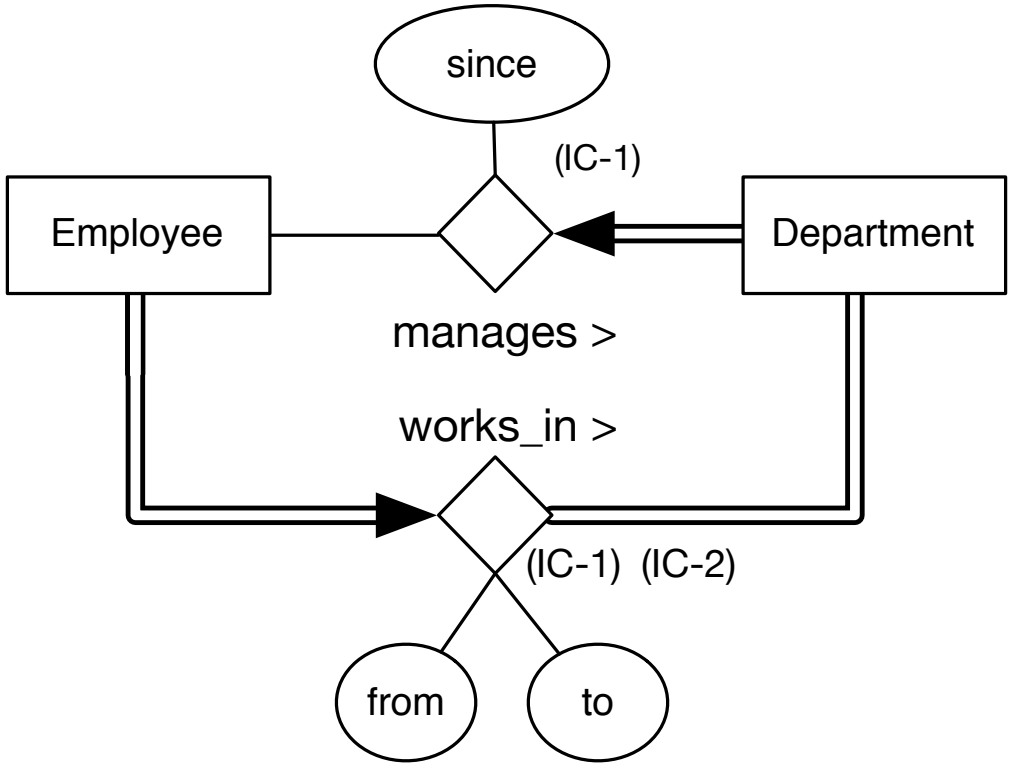
Example with one constraint:



Integrity Constraints:
(IC-1) Employees must **work** at least for three months in a Department

Integrity Constraints

Example with multiple constraints



Integrity Constraints:
(IC-1) Employees must **work** in the Departments they **manage**
(IC-2) Employees must **work** at least for three months in a Department

Integrity Constraint

Definition

An **Integrity Constraint** is a statement (an assertion) expressed on entities, associations, and corresponding attributes, that is **falsifiable** at one (only one) point in time

- An integrity constraint restricts the combinations of (data) values that can be assumed by instances of entities and associations.

Integrity Constraints

1. Written in a simple and objective fashion;
2. Expressed as conditions about entities, attributes and associations;
3. Expressed in terms of obligations (using the words must, has to, ...), or impossibilities / prohibitions (can never, ...)
4. Must be added to the model with numbering
5. Should not have any words that imply analysing more than one snapshot of data (words such as ‘before', ‘after’, etc ...)
6. Prefer multiple simple constraints that can be verified separately to a single very complex one.

Good examples of ICs

- IC-1: An Employee can only **manage** a Department where he / she **works**
- IC-2: A Patient can only **receive** Receipts from a Physician who **follows** it
- IC-3: A Supervisor may only be **designated** by a Customer for Contracts **contracted** into with the Supplier

Bad Examples of ICs

- RI-1: When an Employee is removed from the database, their Dependents must also be
  - It is not a constraint: it refers to the dynamics of the system - it can be a consequence of a restriction (moreover, there is no “removes” association)
- RI-2: An Employee can only become a **manager** of Departments where he has previously **worked**.
  - We are not recording the history of places where an Employee as worked. It is not a constraint because it requires two snapshots of past data to falsify the word ‘*previously*'

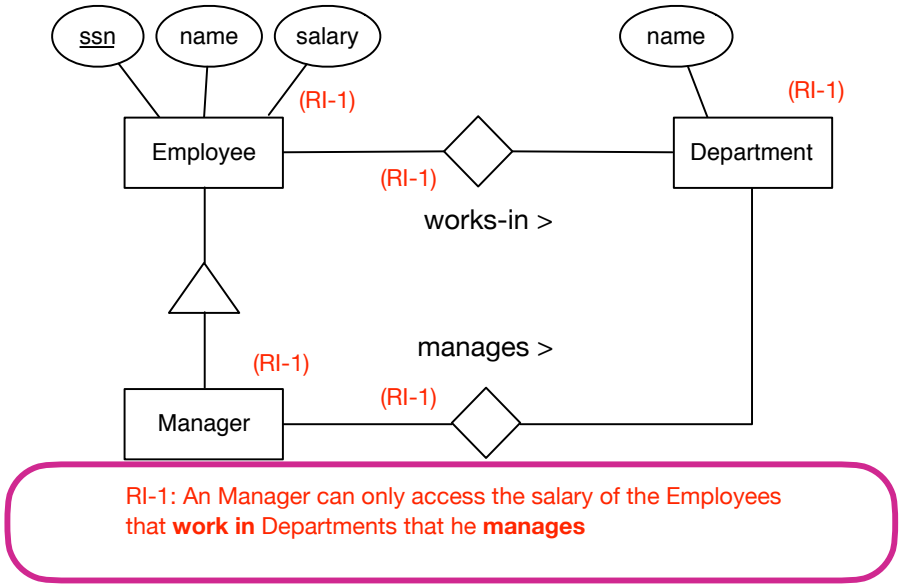
Bad examples of ICs

- RI-3: An Employee remains associated with a Department while enjoying working there
  - This is not a constraint: (i) Remains associated with the need to analyse different versions of the database and (ii) '*like to work*' is not an association, nor an entity, nor an attribute — in other words, the system is not saving any information about where employees likes to work in a department or not
  - A constraint can not be expressed on data/ information that is not captured/derivable from in/ from the model

Bad examples of ICs

- RI-4: There must be a sequential counter for the numbers of the operations
  - It is not a constraint: because it refers to functionality aspects and is not a condition on the data
- RI-4: The numbering of operations should be sequential
  - It is a valid constraint

Bad examples



- "***Can only access***" sounds like a constraint, but it is not!
  - No information is captured about 'accesses' and we cannot express constraints about data that we are not capturing
  - "Can only access" describes a functionality of the authorisation system of the application. The authorisation system needs to query information from the model to decide about authorising or not.

Weak Entities

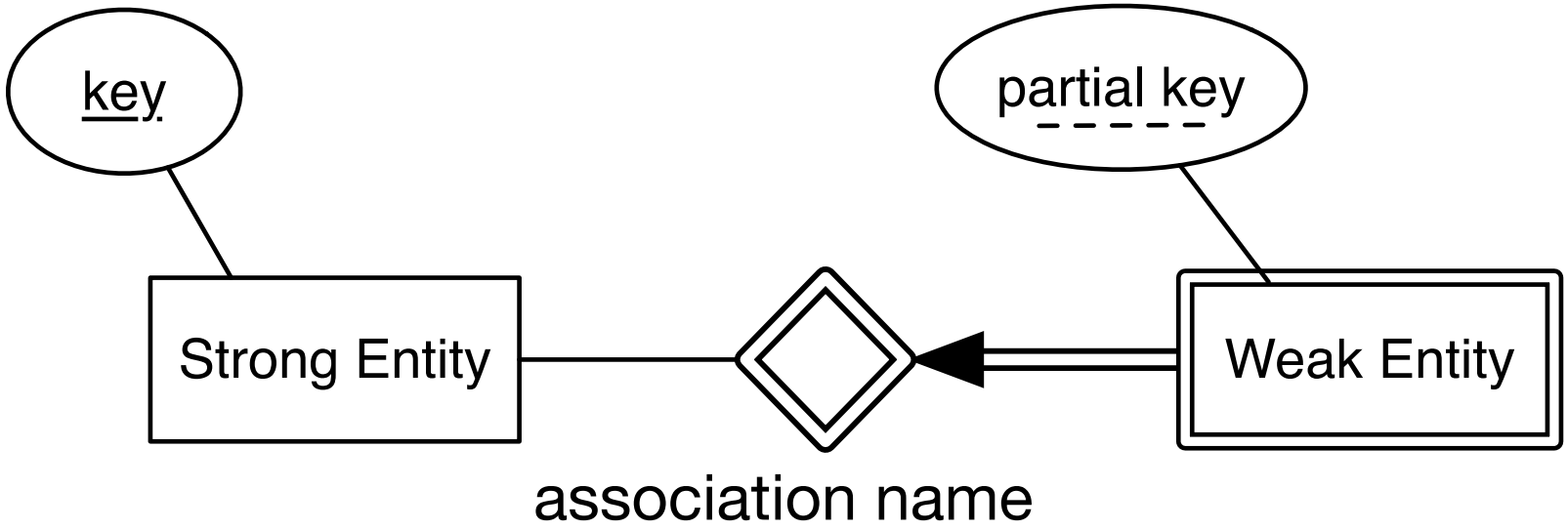
Weak Entities

Definition

A **Weak Entity** is one in which its key is not sufficient to uniquely discriminate each of the instances

E-A Graphic Language

Weak Entity



Each instance on the Weak entity must always be associated to an instance of the Strong entity type (through a binary association)

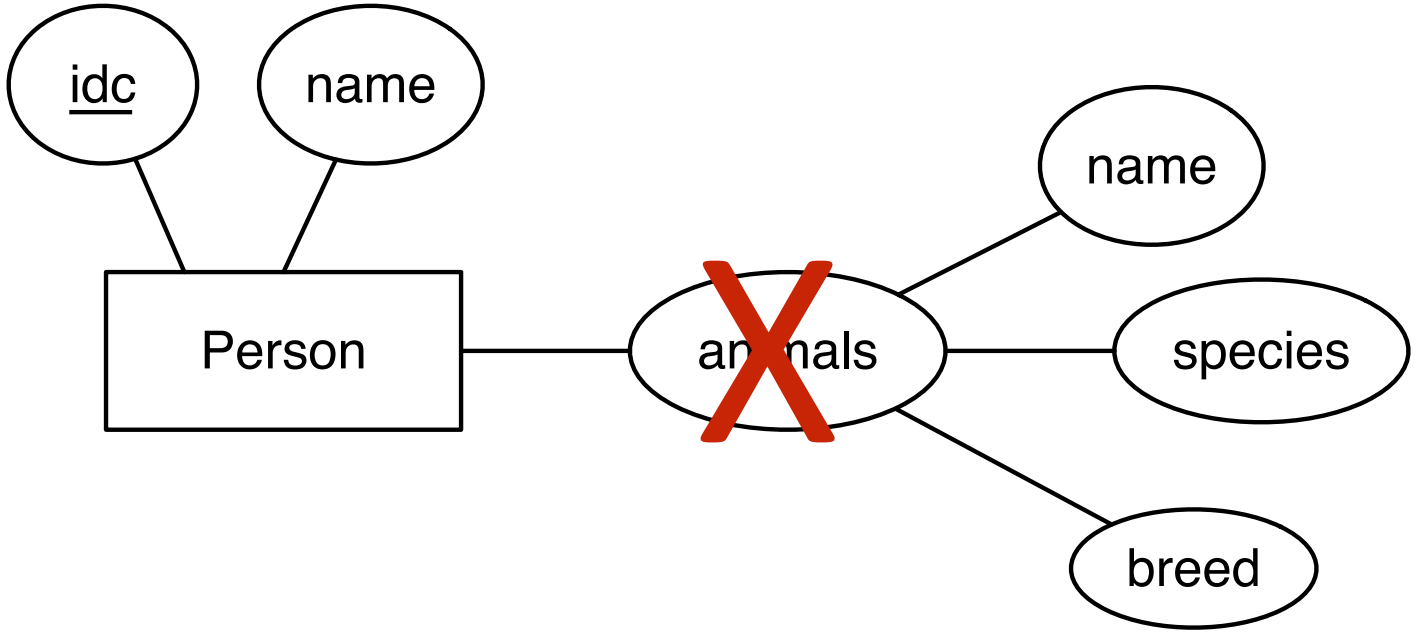
Heuristics for Weak Entities

Heuristics for Weak Entities

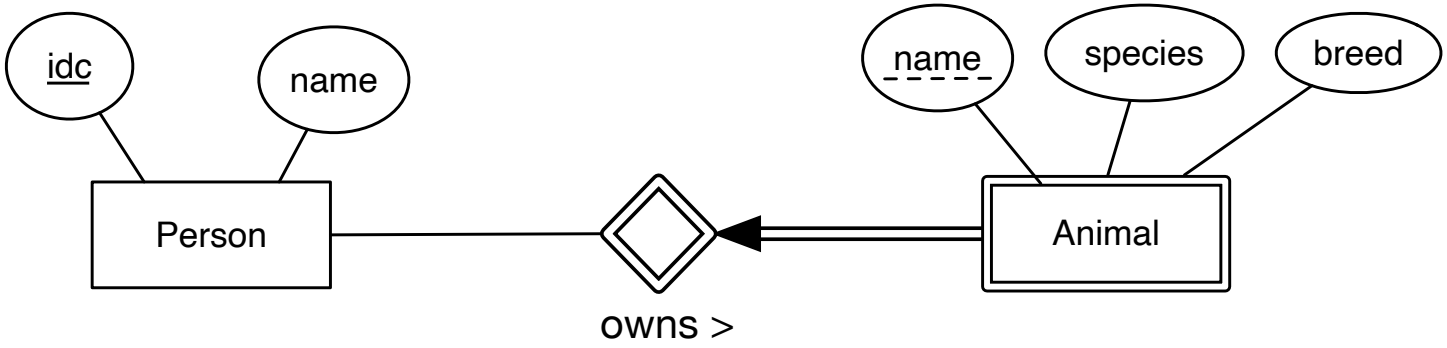
- **Multi-valued attributes**: Situations where an "attribute" must support multiple values for the same instance (e.g. a customer's favorite colors)
- **Structured attributes**: In situations where an entity has an "attribute" that is structured as other attributes (eg, an attribute that is structured in **another attribute)**
- ***"Part-of*" relationships**: Situations (e.g., a system consisting of several parts)

Multi-valued and structured attributes

Suppose that a Person can have multiple animals and each animal can have its own atributes (characteristics)



Solution using a Weak Entity

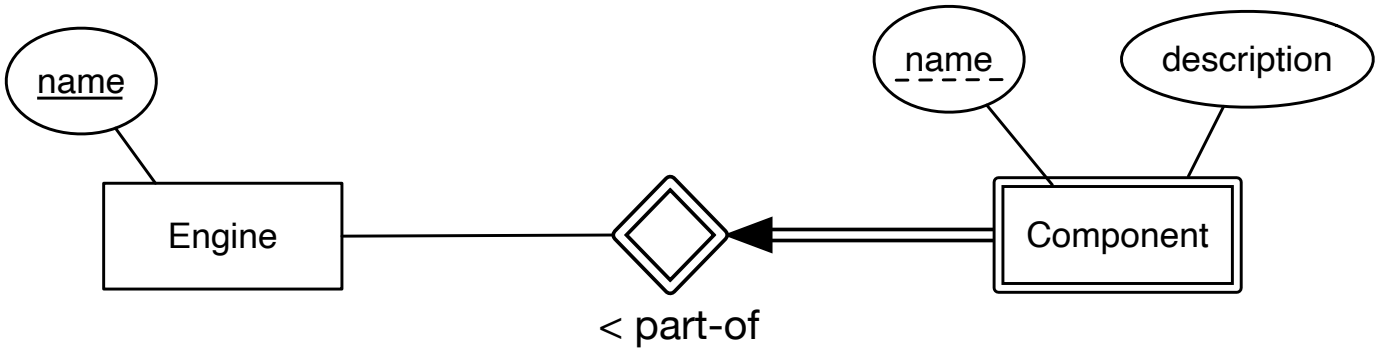


- Represented by double (or bold) lines
- Diamond is double lined
- The **partial key** is underlined

Notes about Weak Entities

- The instances of a **Weak Entity** are identified by:
  1. Its own Partial Key
  2. The Primary Key inherited from another entity (called the Strong Entity)
- The following constraints apply:
  1. A one-to-many association (known as identifying association) that links the strong entity and weak entities
  2. All weak entities must have total participation on the strong entity

“Part-of” relationship



- An Engine consists of multiple Components
- Each Component is identified by its name and can only be part of one Engine

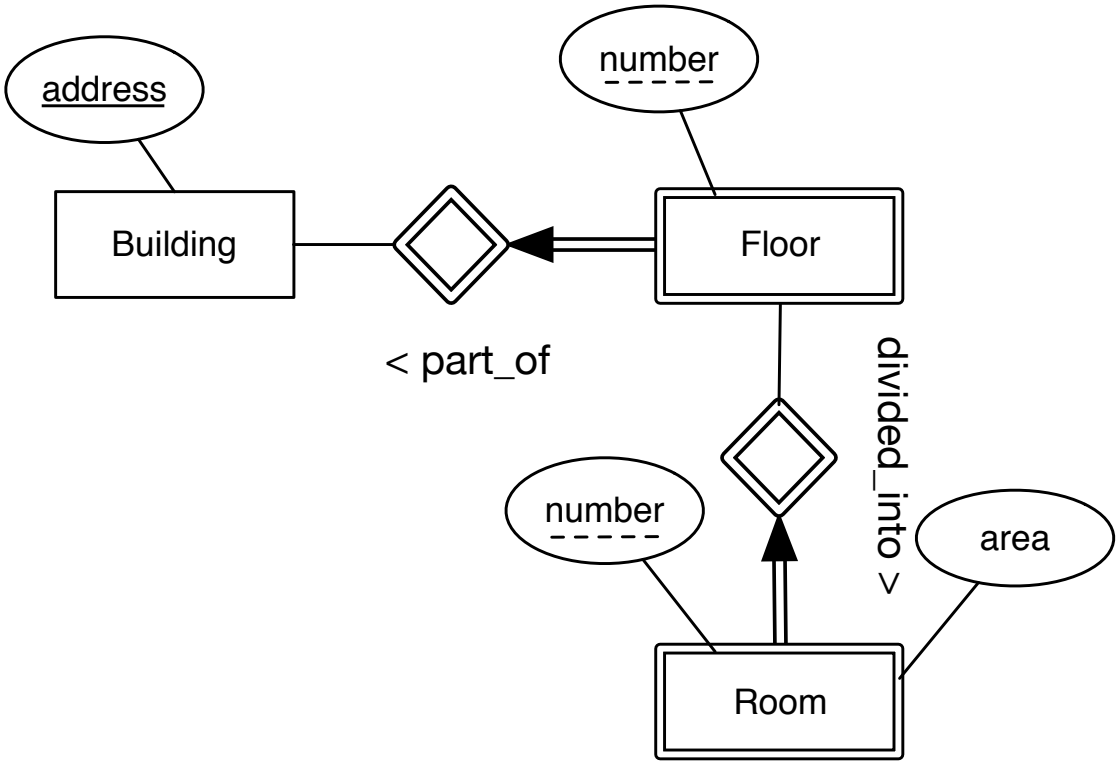
Exercise C.

Modelling of Buildings, Floors, and Rooms

- A building is identified by its address
- Each building is divided into floors, each floor has a floor number
- All buildings have at least one floor
- Floors can be divided into rooms that are identified by their number and have an area
- Each building has an entrance floor (not necessarily the ground zero)

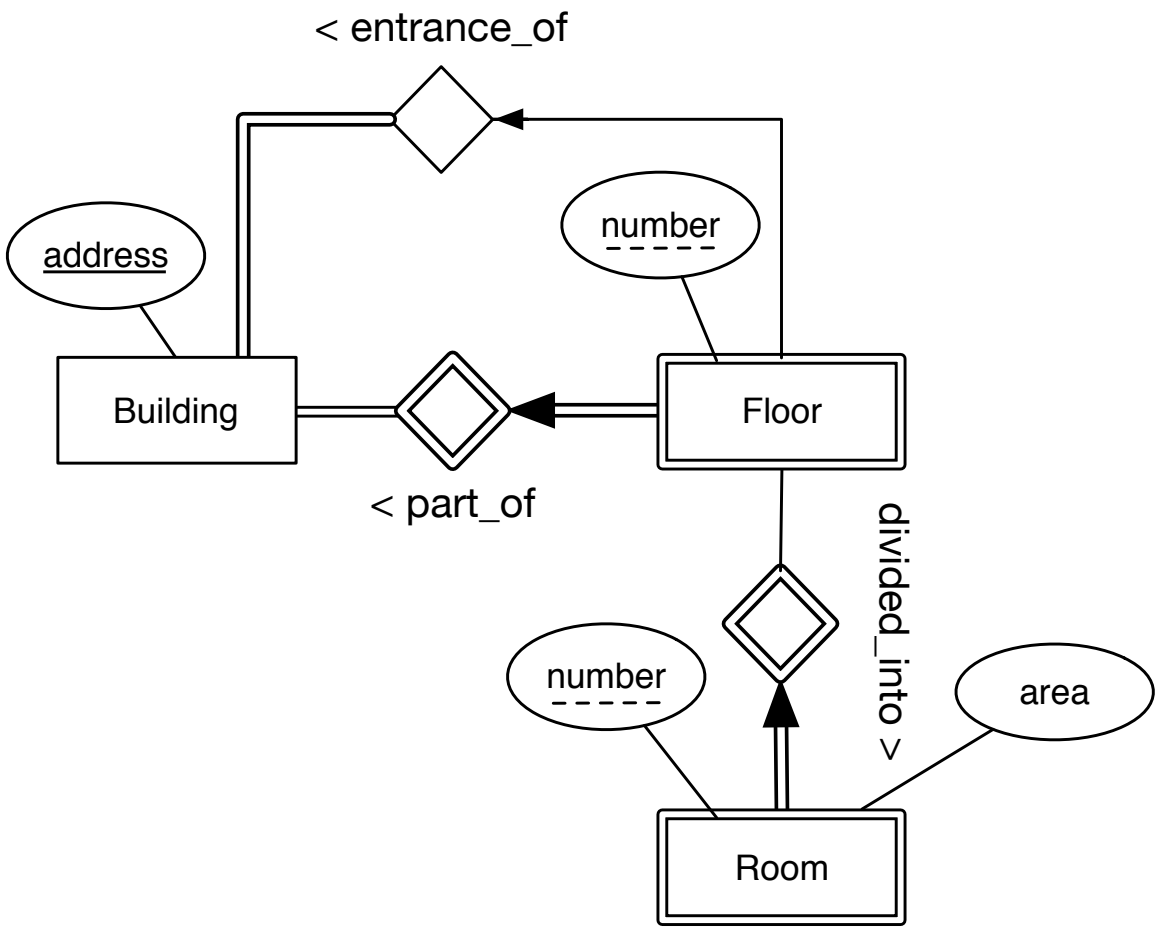
Solution - Part I

Nested weak entity



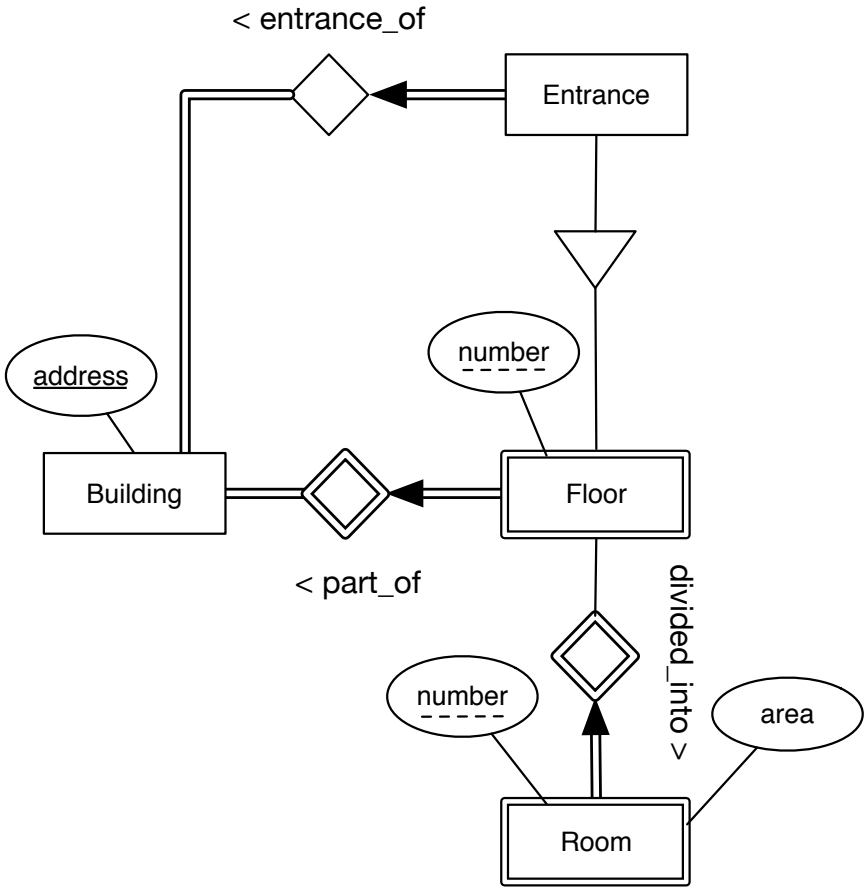
What's missing?

Solution - Part 2



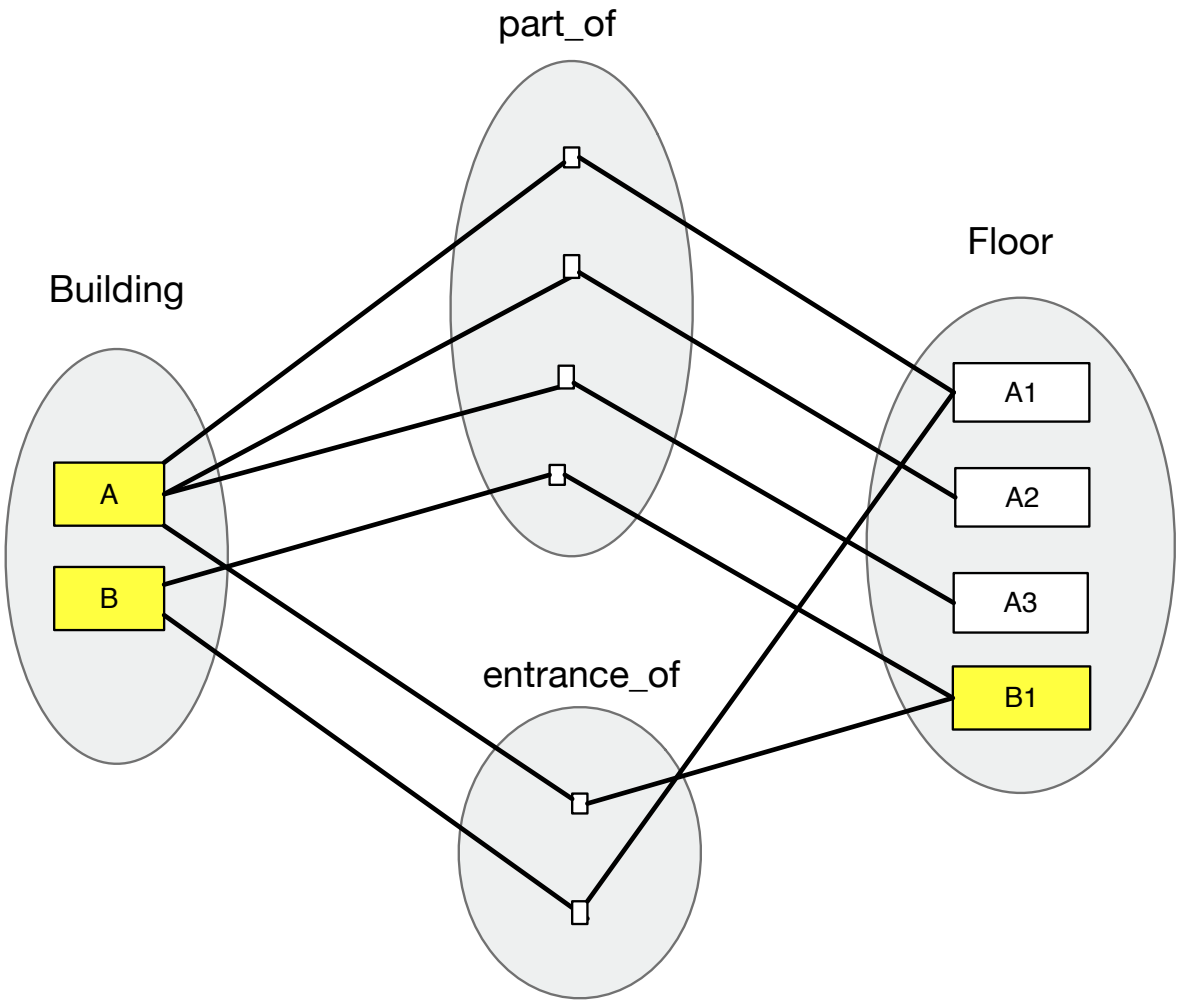
What's missing?

Solution - Part 3



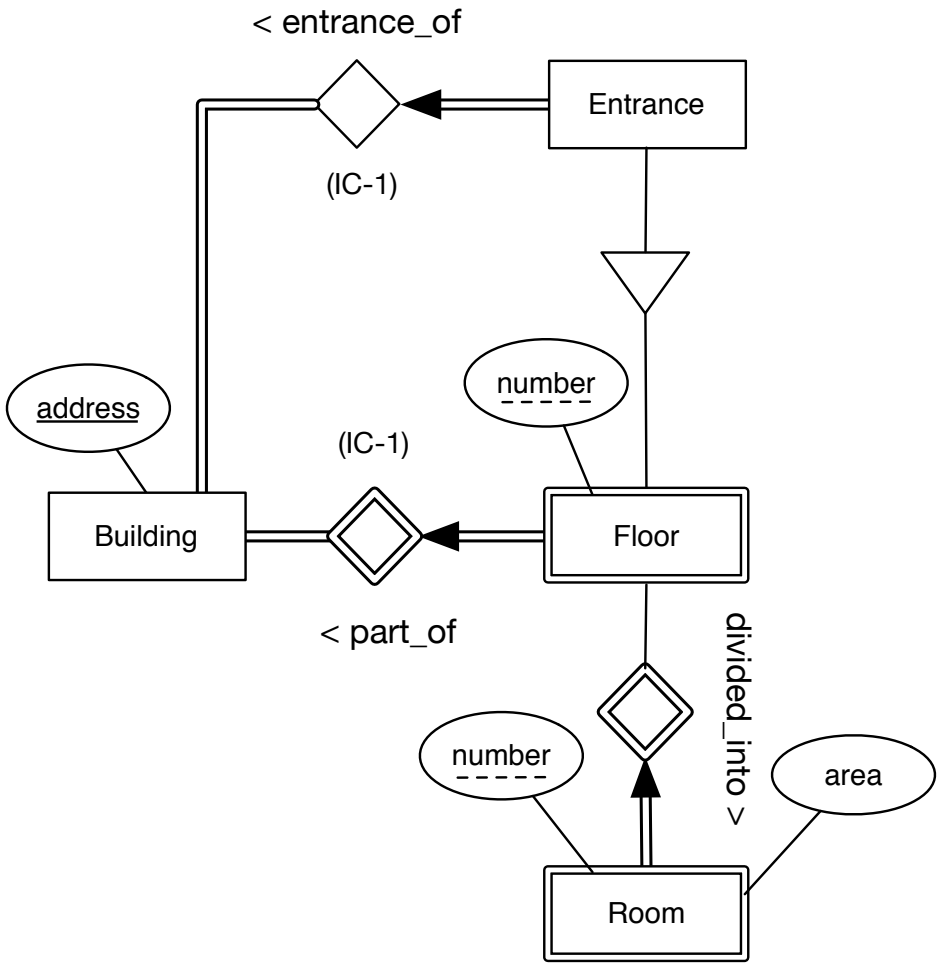
Can a building be associated to the entrance of another building?

Solution - Part 4



Floor B1 is associated as **part_of** building **B** but also associated as **entrance_of** building **A**

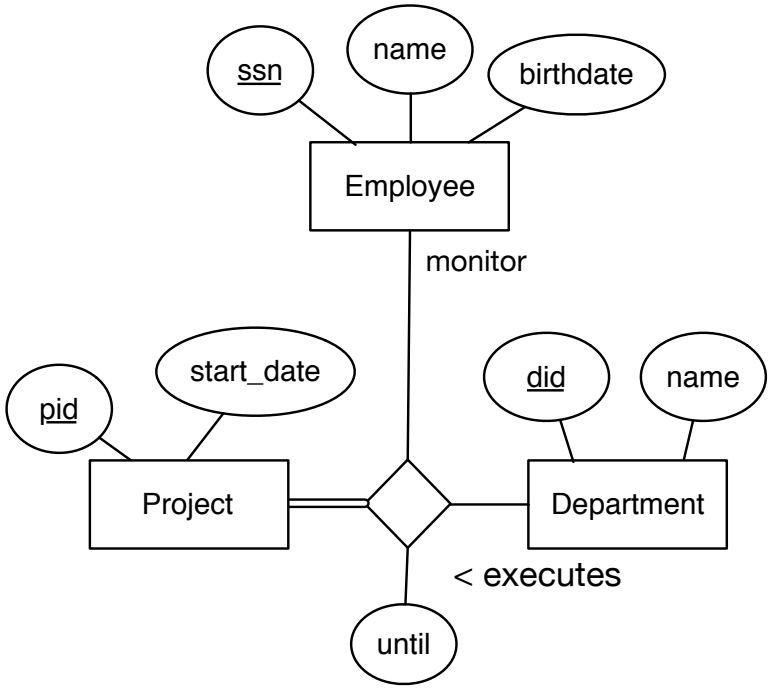
Solution - Part 4



Integrity Constraints:
(IC-1) The **entrance of** a Building must be a Floor that is **part of** that Building

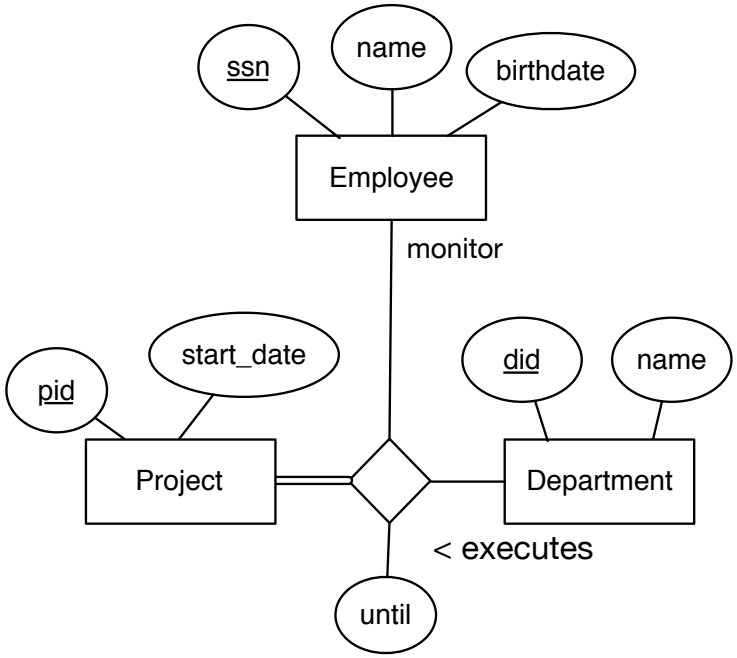
Aggregation

Motivation for Aggregation: The optional leg



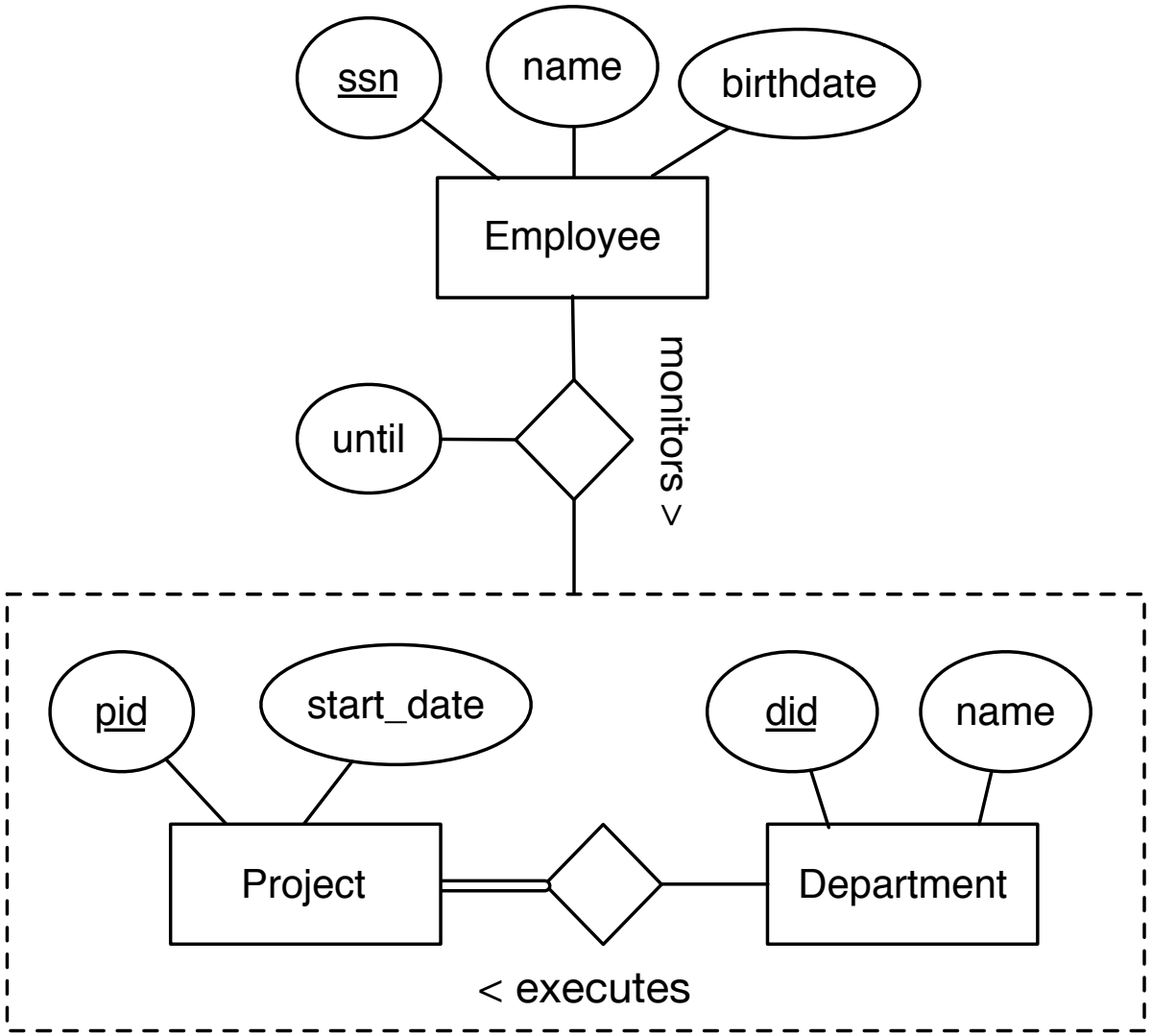
- This diagram implies that all Project **executions** must be monitored by Employee
- How can we make monitoring optional?

Motivation for Aggregation: Associating



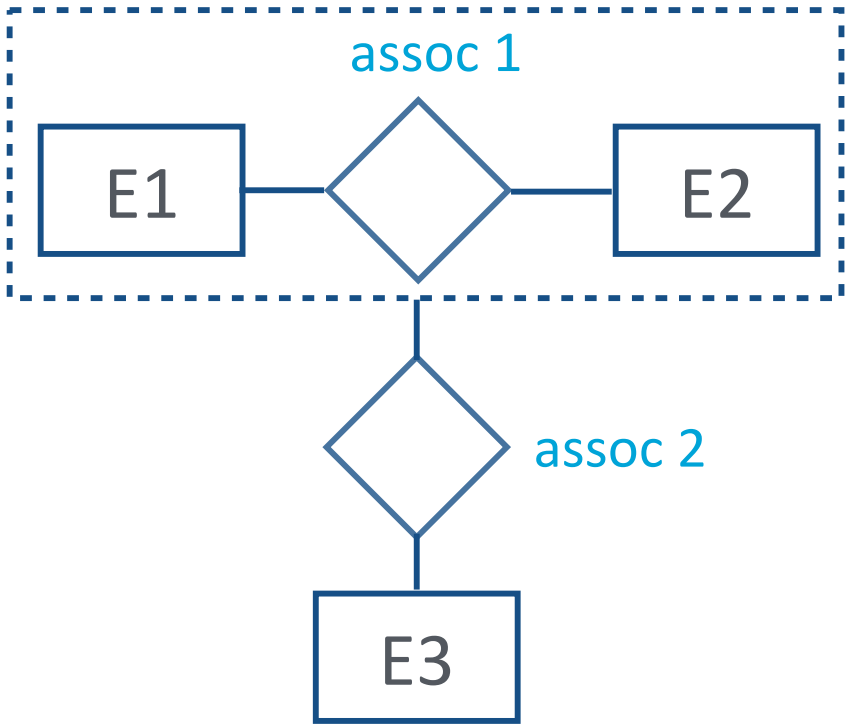
- Adding a new Employee implies establishing a new Execution
- How can we associate an Employee to multiple Executions?

Aggregation



E-A Graphic Language

- Aggregation



Instances of E3 can be associated with associations (combinations) of E1 with E2.

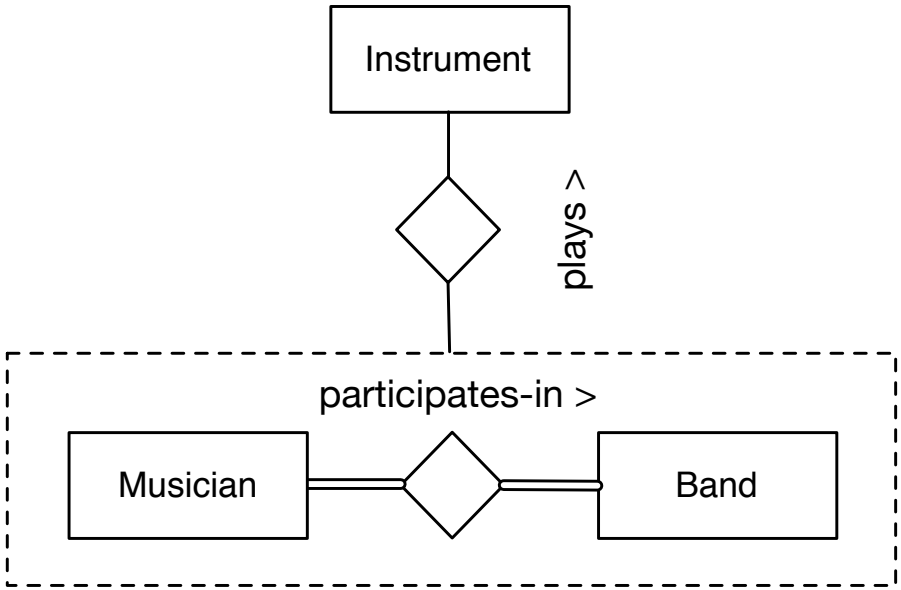
Exercise D.

Aggregation (Optional Third Entity)

- Present an E-A model that is consistent with the following requirements

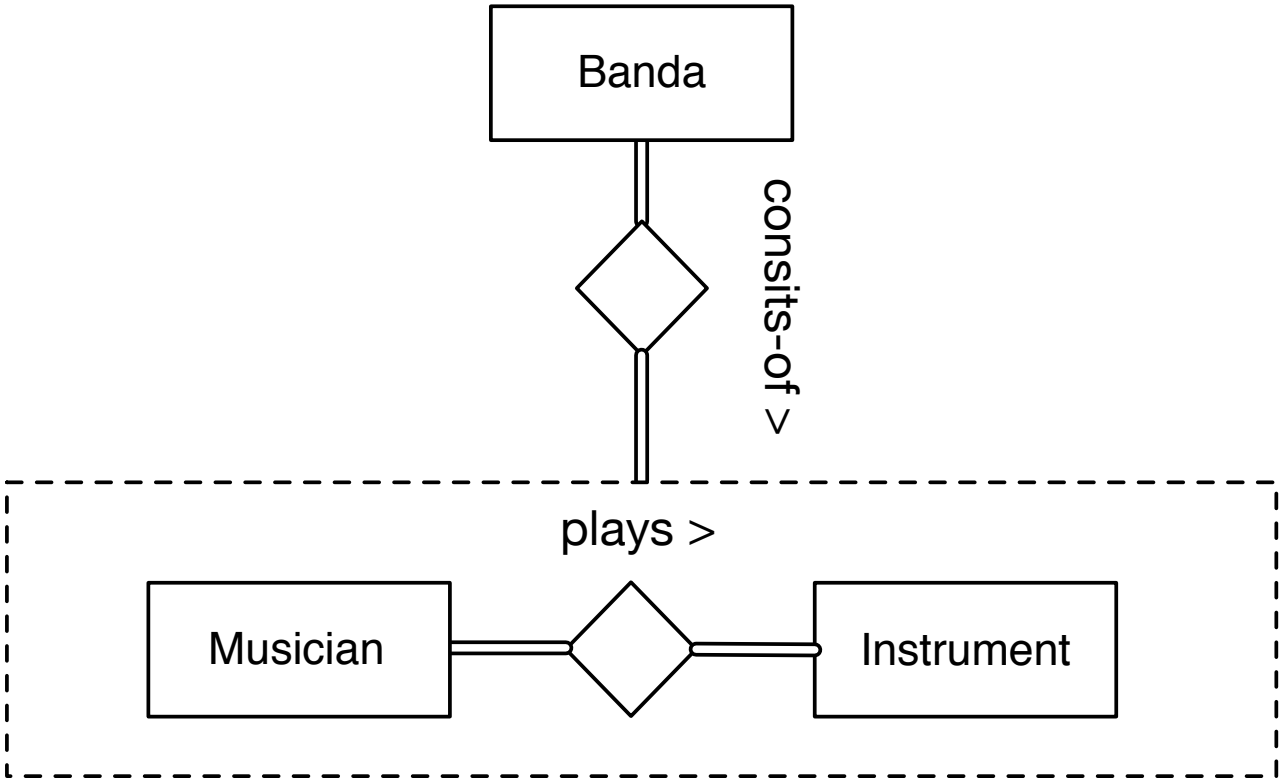
1. Musicians belong to Bands
2. Some Musicians can play Instruments in the bands they belong to
3. The instruments that the musicians play may be different in each band

Solution 1



- Every Musician participates in a Band an may play zero, one, or many Instruments

Solution 2



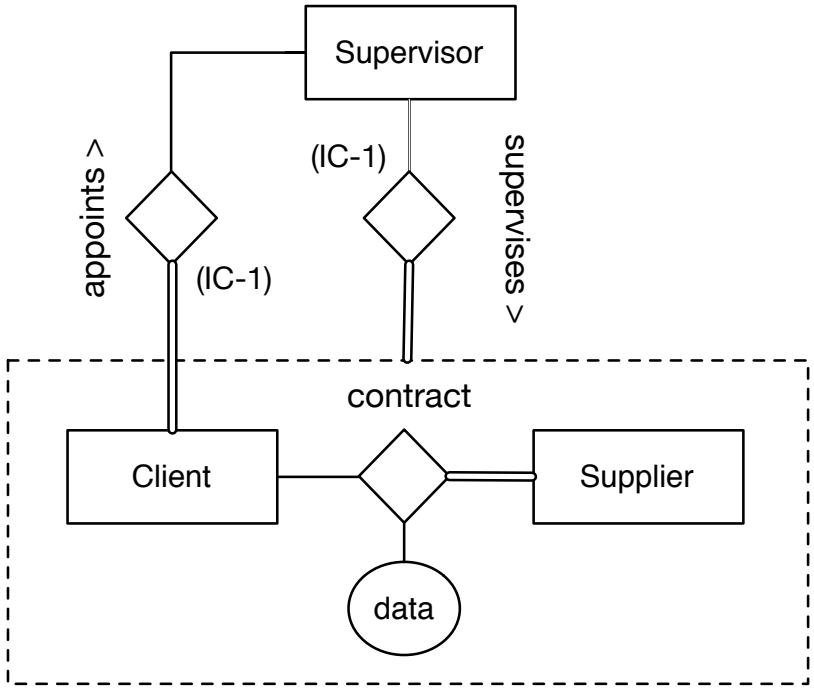
- In this solution, the Musician must, necessarily play an Instrument, to belong to a Band

Exercise E.

Aggregation (History of facts)

- Present an E-A model that implements the following requirements:
  - A Customer starts a **contract** with a Supplier on a date
  - Customers assign Supervisors to the contract
  - Every Contract has a Supervisor
  - Customers may change the Supervisor during the Contract
  - It is necessary to keep the history of all Supervisors associated with a contract

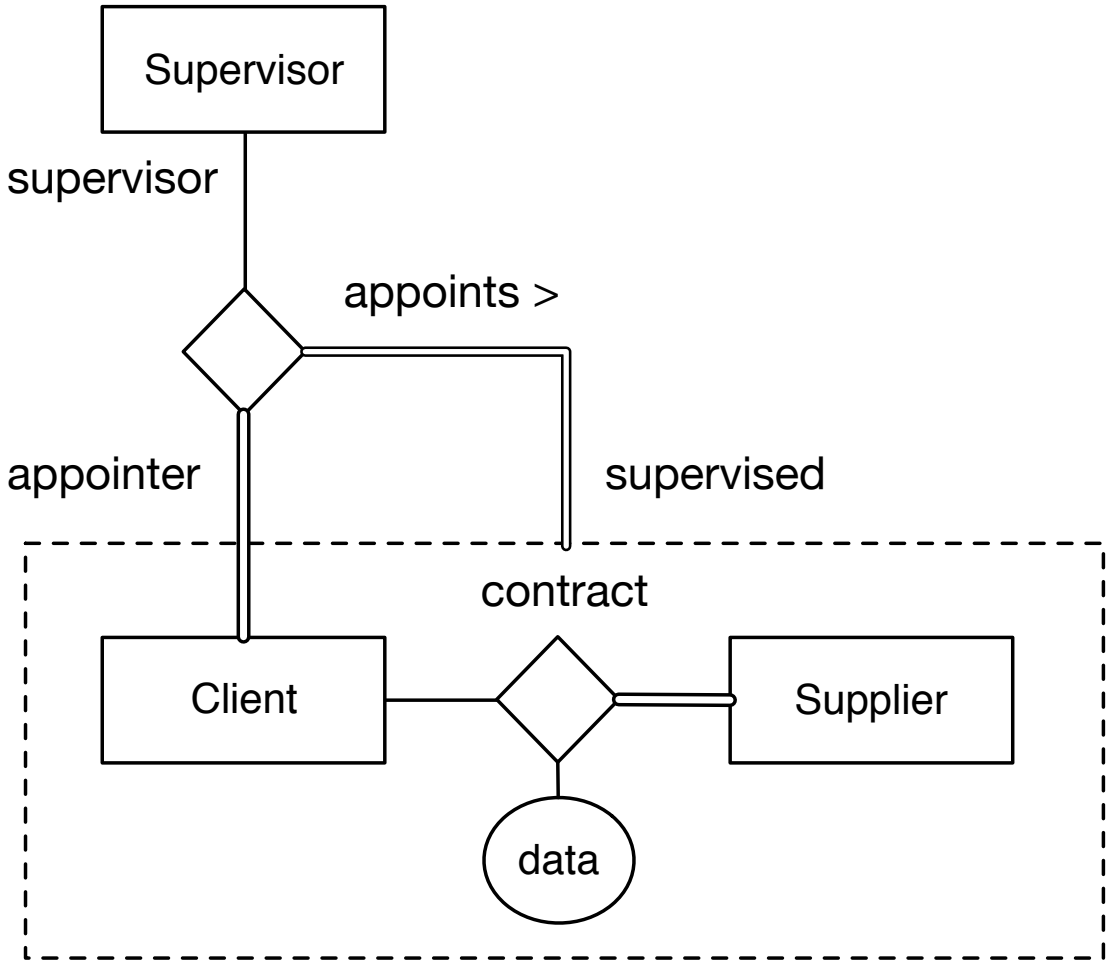
Solution



Integrity Constraints:
(IC-1) Supervisors can only **supervise contracts** for which they were **appointed**

- In this diagram, when a Supervisor supervises multiple contracts you can not tell who assigned you to contract

Solution

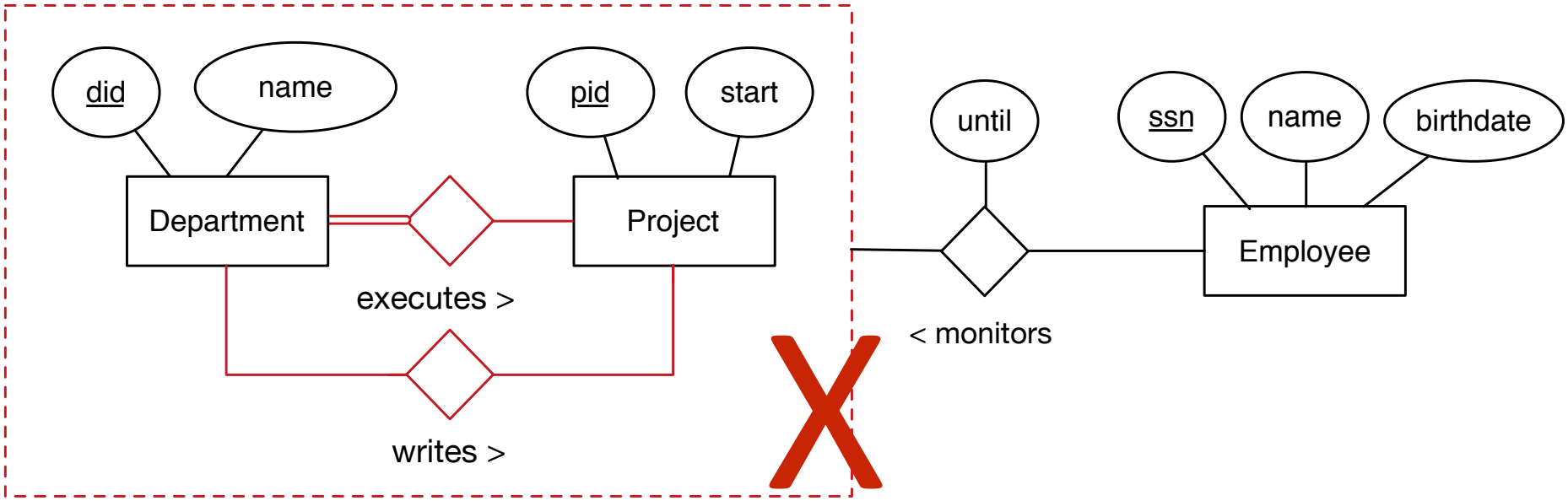

Supervisor
supervisor
appoints >
appointer
supervised
contract
Client
Supplier
data

Heuristics for Aggregation

- **Heuristic 1 (optional leg)**: Same as in a ternary relationship but in which we want to make a leg optional

- **Heuristic 2 (multiple facts)**: when we need to keep multiple facts (or an history) about an association

Typical Mistake: Aggregation of multiple



- An aggregation can only refer to (aggregate) one association
- In this example it is ambiguous what is the association being monitored