TÉCNICO LISBOA

Sistemas de Informação e Bases de Dados

Class 06: Entity-Association Model (cont.)

Prof. Paulo Carreira

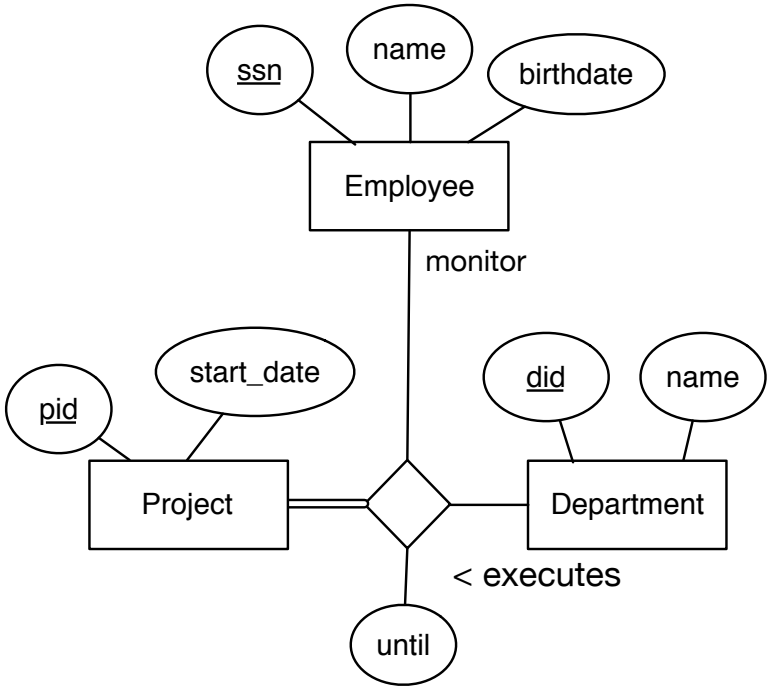
Class Outline

- Aggregation
- Model Refactoring
- Model Quality
- Solution for Lab 01
- Q&A

Aggregation

Motivation for Aggregation:

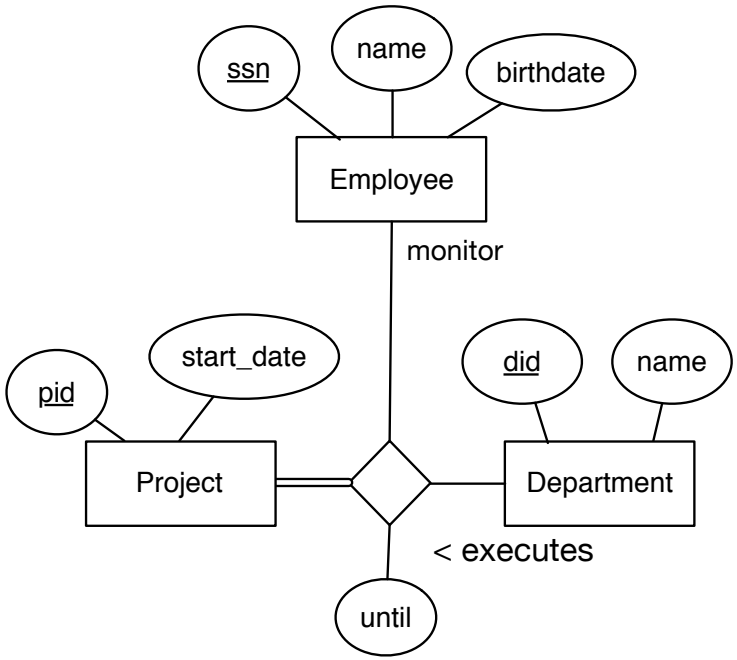
"The optional leg"



- All Project **executions** must be monitored by Employee
- How can we make monitoring optional?

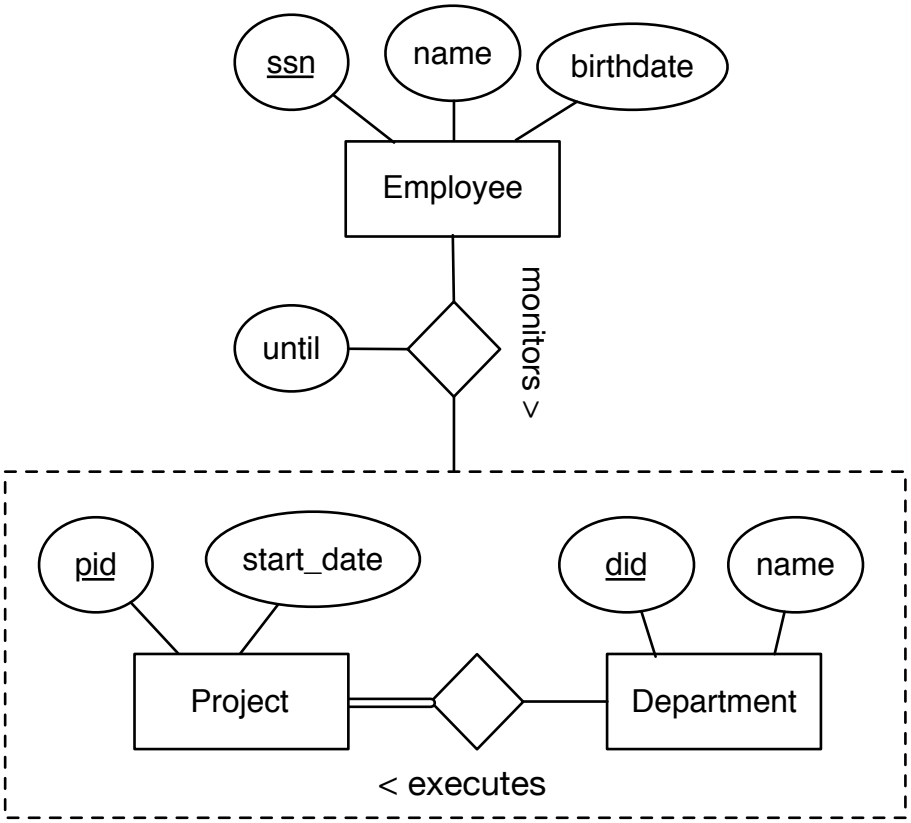
Motivation for Aggregation:

"*Associating multiple facts*"



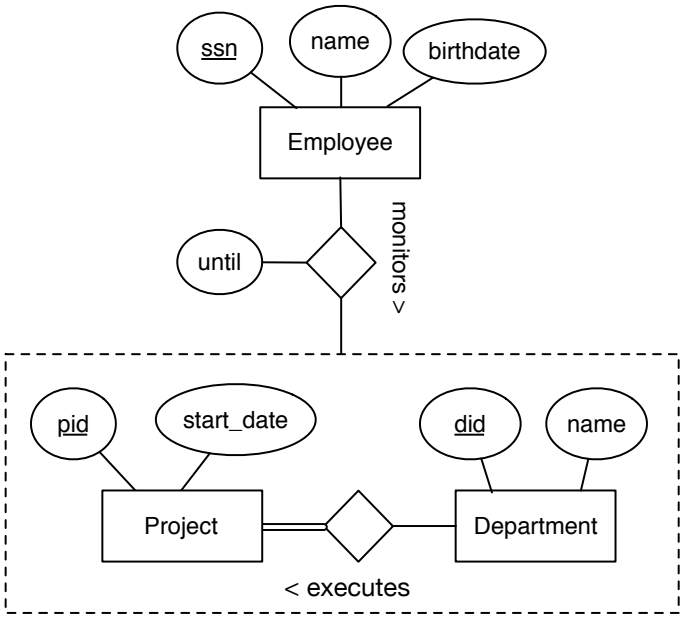
- Each **execution** can only be monitores by one employee (n.b.: Adding new Employee implies establishing a new **execution**)
- How can we associate one **execution** to multiple Employees?

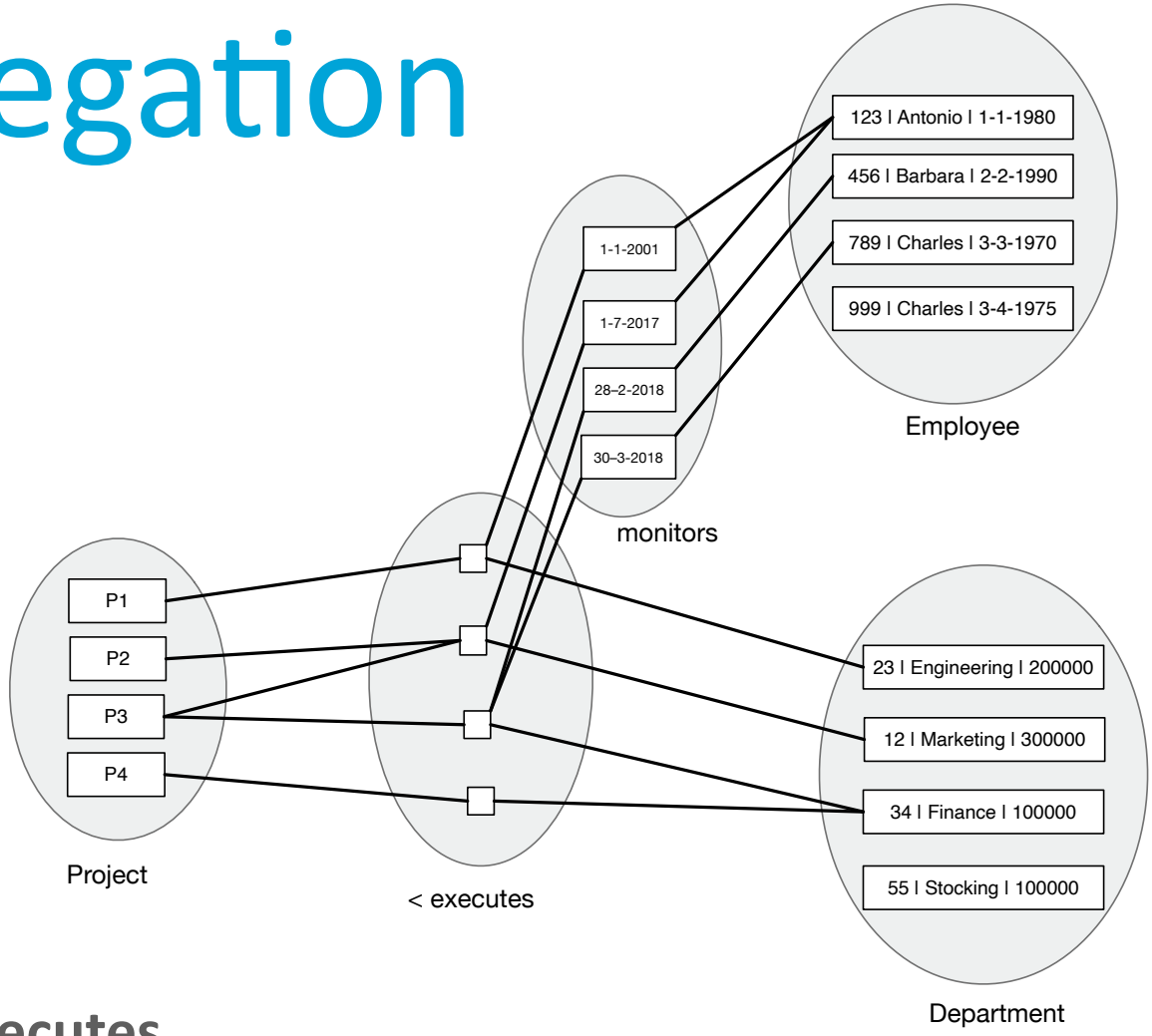
Aggregation



- Instances of **execution** can now be associated to Employees
- Not every **execution** needs to be monitored
- The same **execution** can be monitored by multiple Employees

Set interpretation for the Aggregation

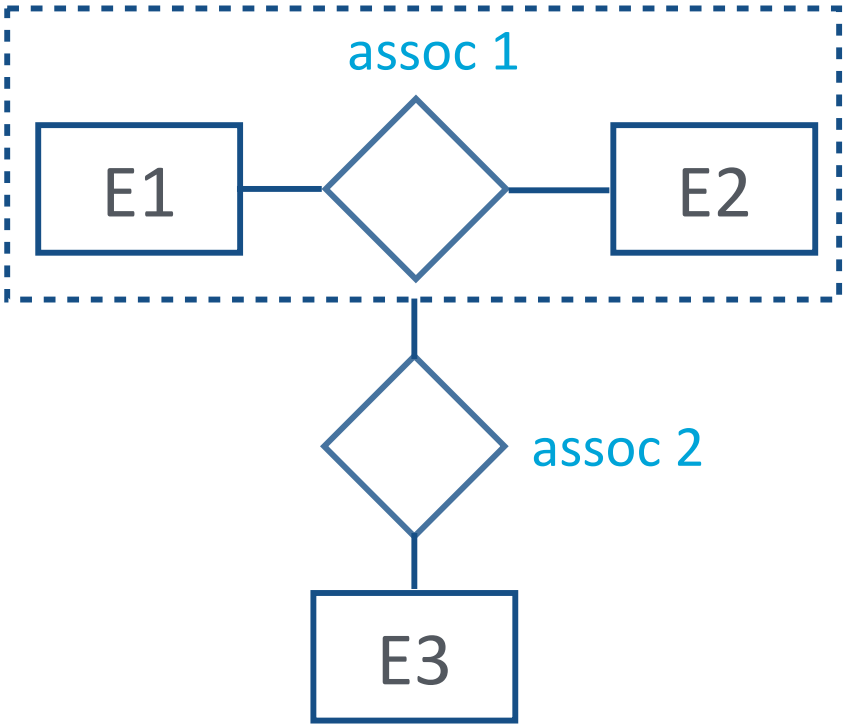




- All Projects are **executed**
- Not every Department **executes**
- Some **executions** are not **monitored**
- Some **executions** are **monitored** by more than one Employee
- Some Employees **monitor** more than one **execution**

E-A Graphic Language

Aggregation is used to model the association between an entity and another association



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

- The key of an association is the the combination of the keys of E1 and E2

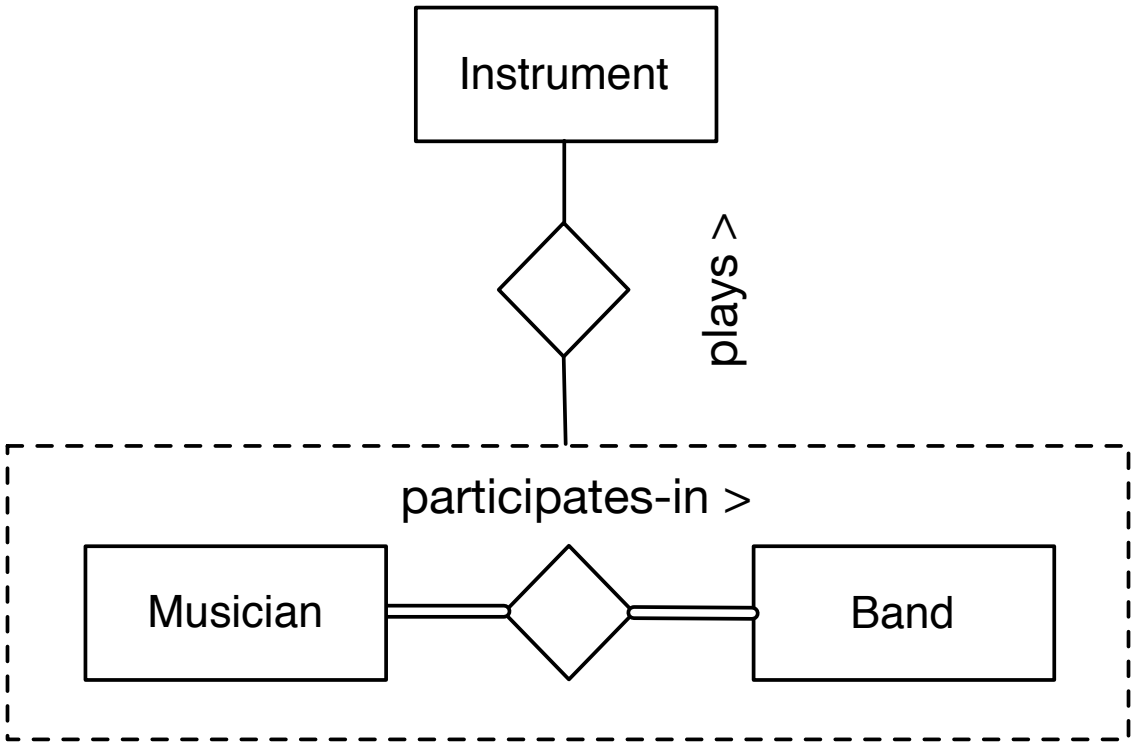
Exercise D.

Aggregation (Optional Third Entity)

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

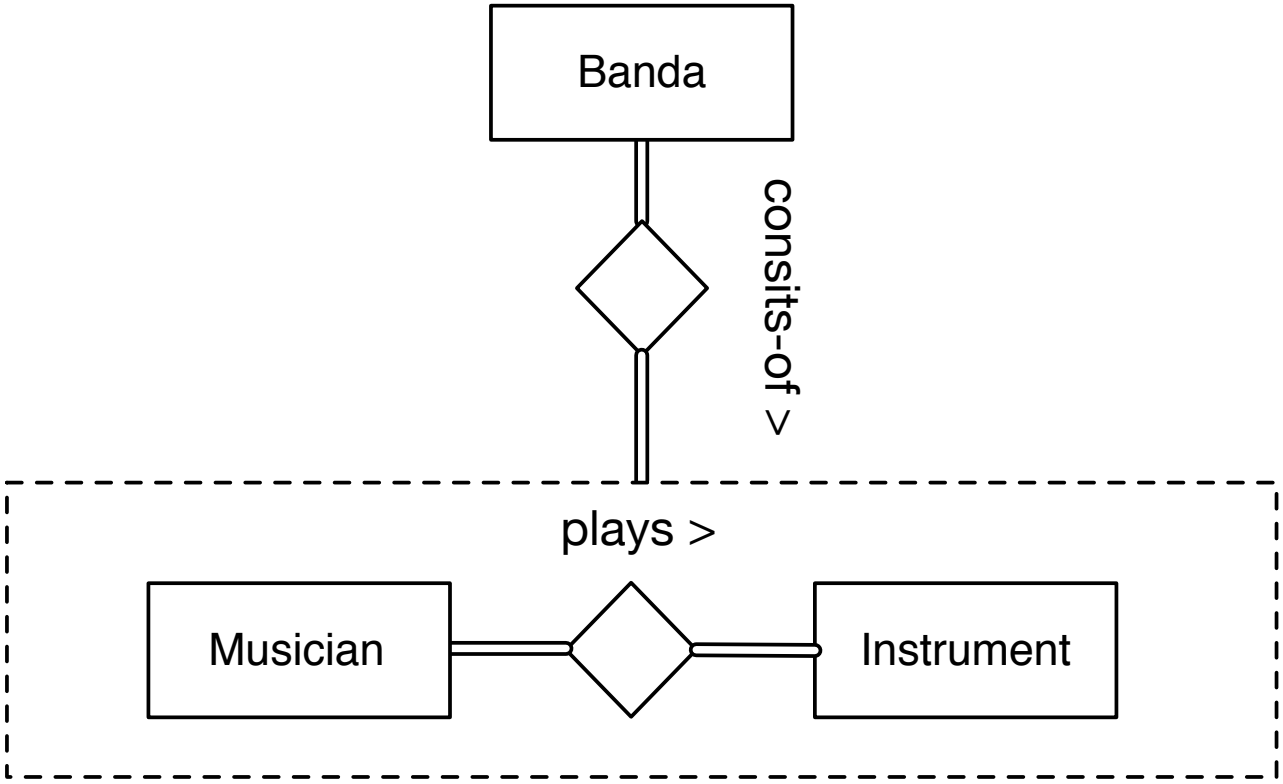
1. Musicians participate in Bands
2. Some Musicians can play Instruments in the bands they belong to
3. Musicians can play distinct instruments on each band

Solution 1



- Every Musician participates in a Band
- One each ‘participation’, musicians may play zero, one, or many Instruments

Solution 2



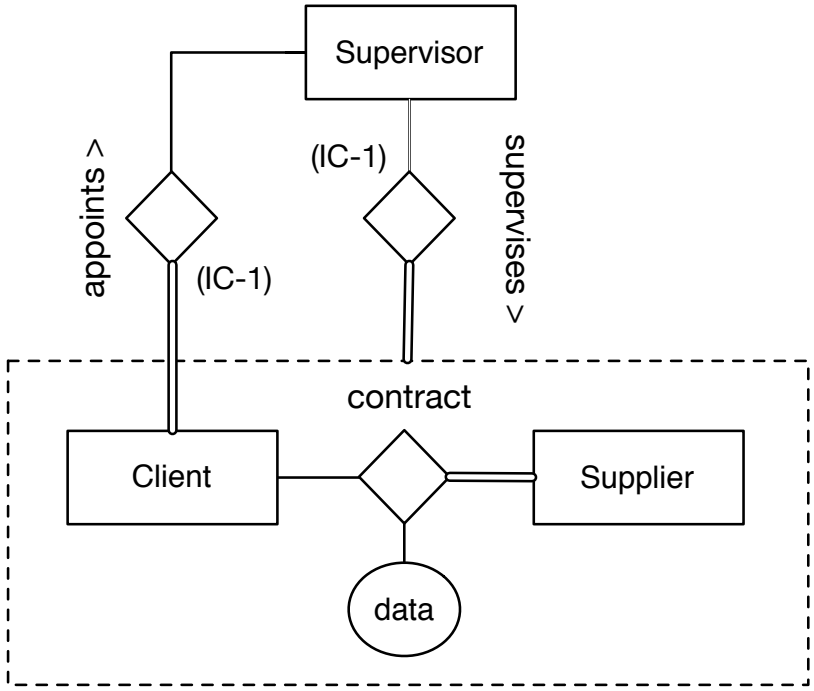
- Bands consists of Musicians playing instruments
- Every Musician must, necessarily play an Instrument

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 appoint 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

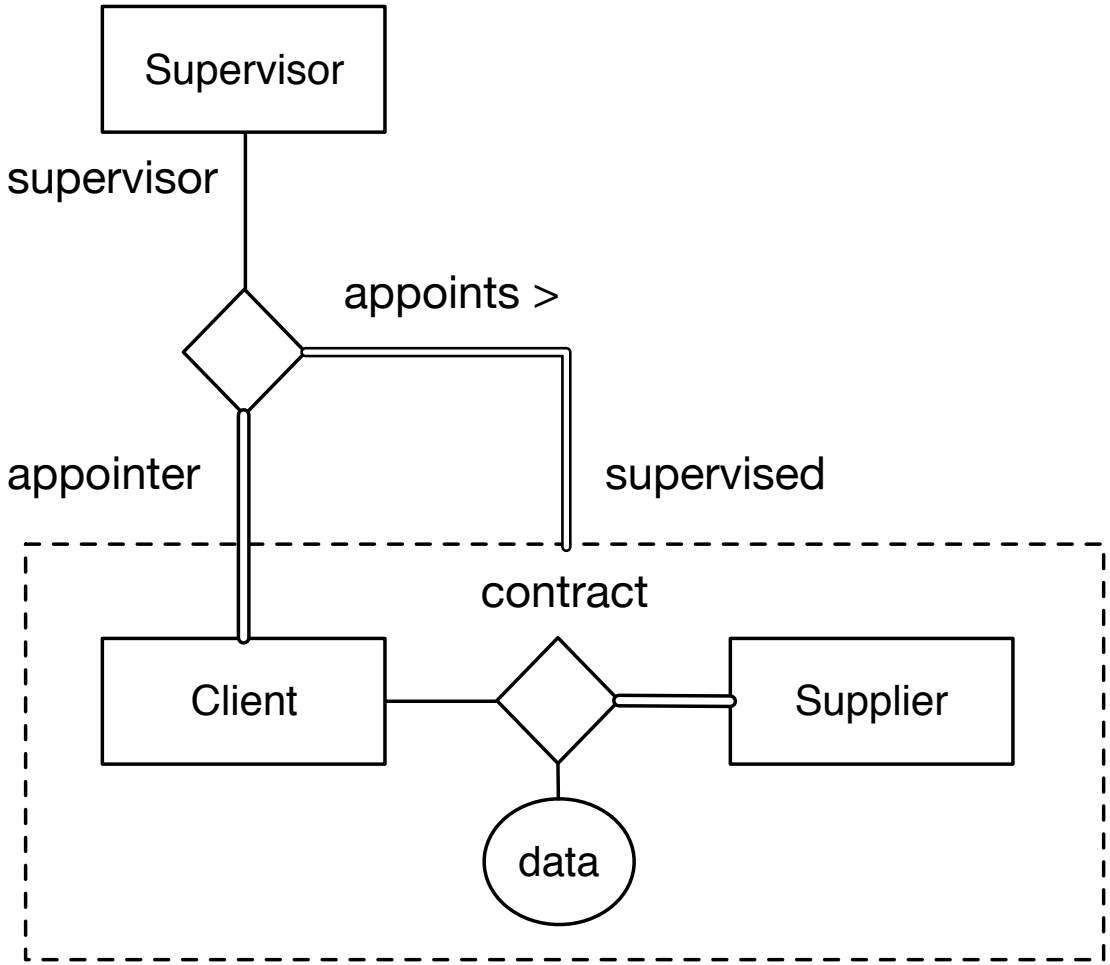
Solution



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

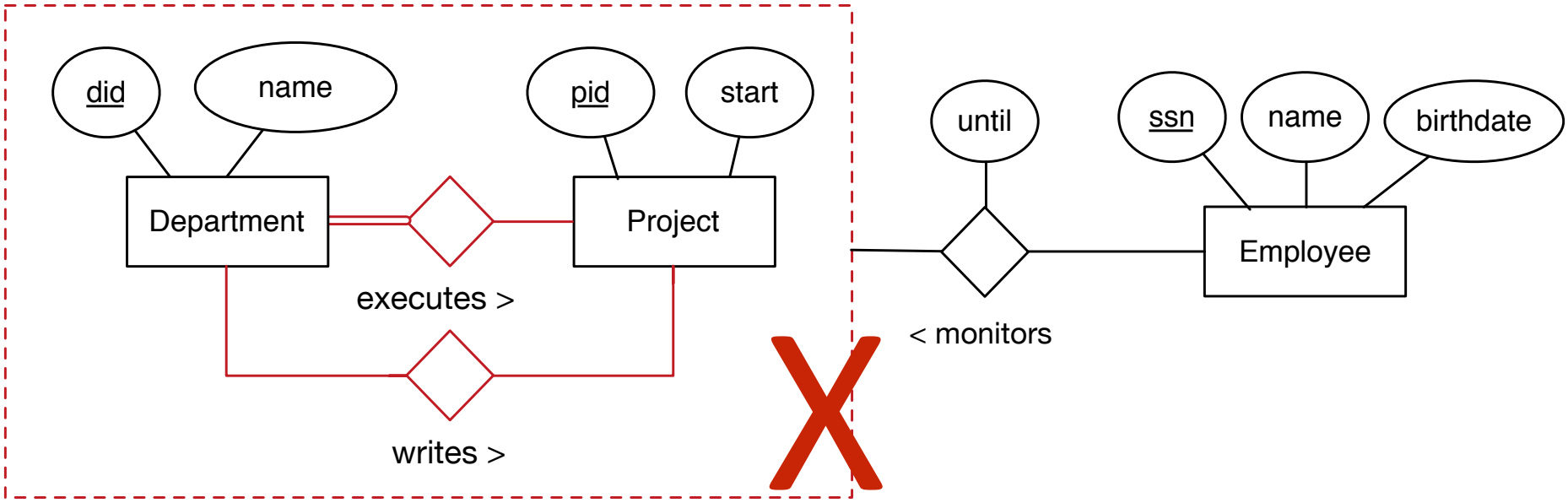

Supervisor
supervisor
appoints >
appointer
supervised
contract
Client
Supplier
data

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 associations



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

Iterative Development Methodology

Criteria for finishing an E-A model

A model is said to be finished when it is:

- **Complete:** When it captures all the information requirements for a given reality (domain)
- **Correct:** When it cannot represent invalid instances of the reality (domain) the it is modelling.
- **Sound:** When it cannot encode inconsistent (contradicting) information.
- **Non-redundant:** When given aspect of reality (domain) is not captured in the model in distinct ways.

Criteria of good E-A models

Models can be compared according to:

- **Conciseness**: A model is said to be concise if the concepts are modelled explicitly using the full expressive power of the language
- **Incrementality**: A model is said to be incremental if new requirements result in additions to the model (instead of changes to the model)

Iterative design Methodology

1. Identify Entities and Associations between them
2. Develop partial sub-models if necessary and integrate them
3. Decorate the models with Multiplicity and Participation constraints
4. List and organize the Integrity Constraints that cannot be expressed graphically
5. Iterate and refine Schemes
   - Find / remove entities, associations, attributes
   - Identify additional constraints (question circuits)
   - Resolve conflicts
6. Refactor parts of the model

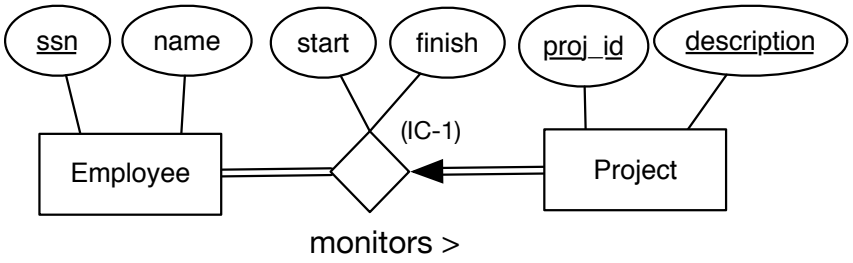
Model Refactoring Heuristics

Heuristics for Concise and Incremental models

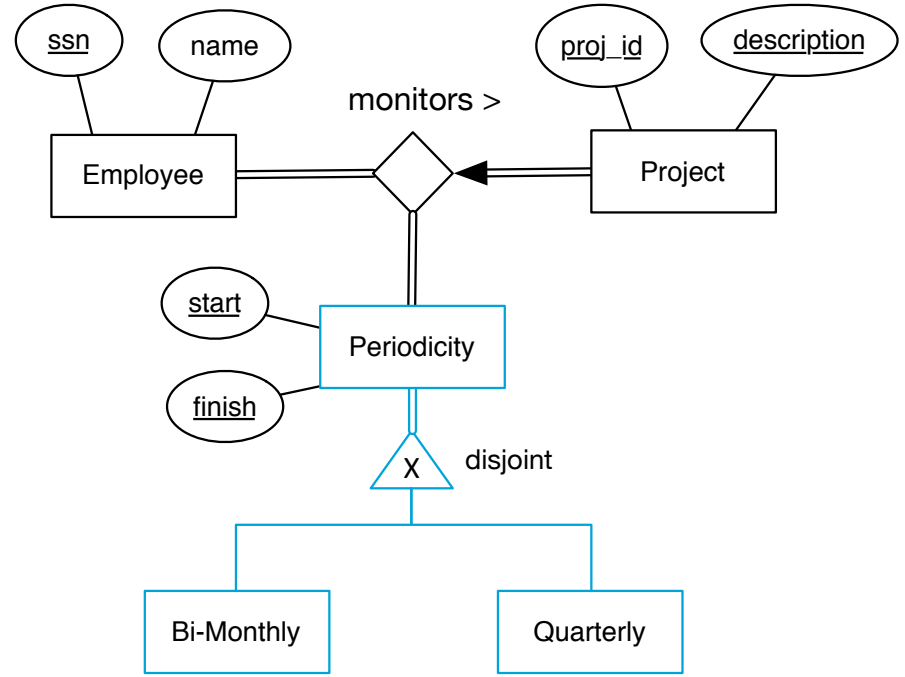
- Models should strive to use the maximum expressiveness of the language, therefore making as many decisions graphically visible as possible
- Models should minimize textual specified constraints as much as possible — because they have higher cognitive overload by comparison with graphically represented ones

Remove Textual Integrity Constraints

- Uses a textual Ic

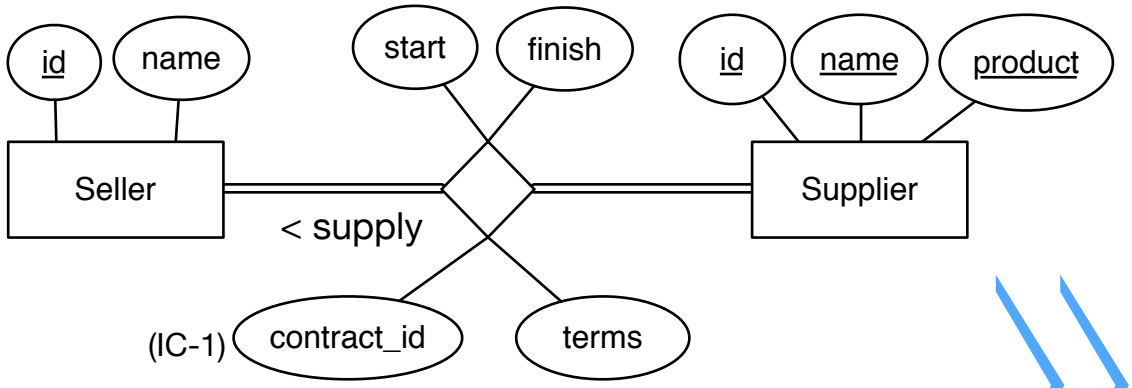


Integrity Constraints:
(IC-1) Employees may only **monitor** Projects bi-monthly or in quarters

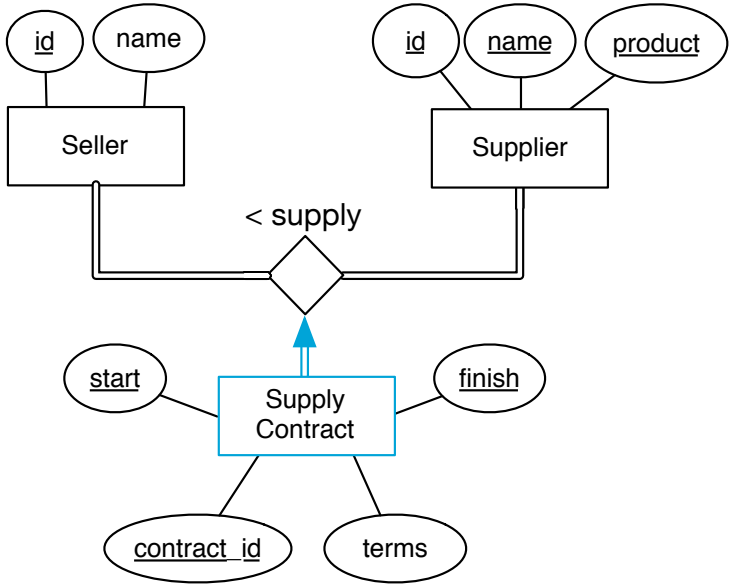


- Better: The model replaces the text IC

Replace Association with many attributes by an Entity

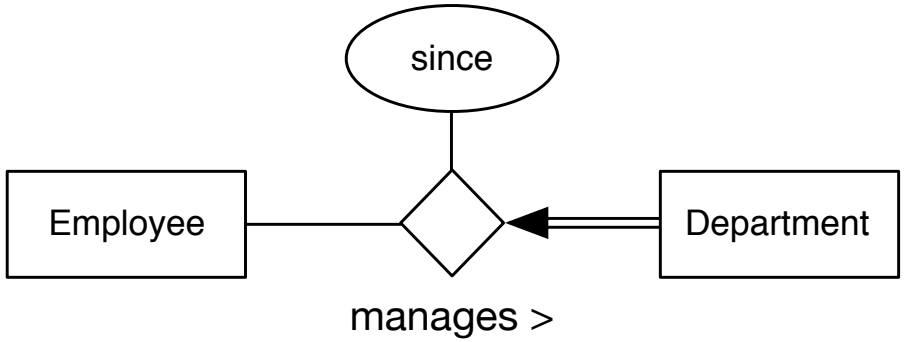


Integrity Constraints:
(IC-1) The contract_id of a **supply** is unique

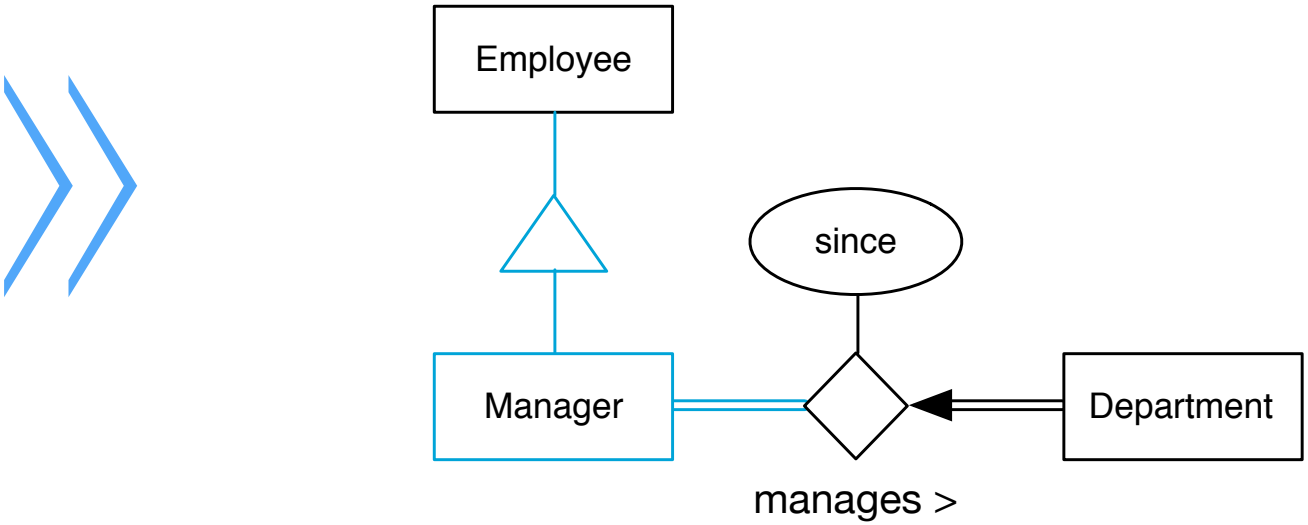


- One of the attributes of the association is a key and is captured as an IC
- The key is now captured in the model

Introduce Generalisation

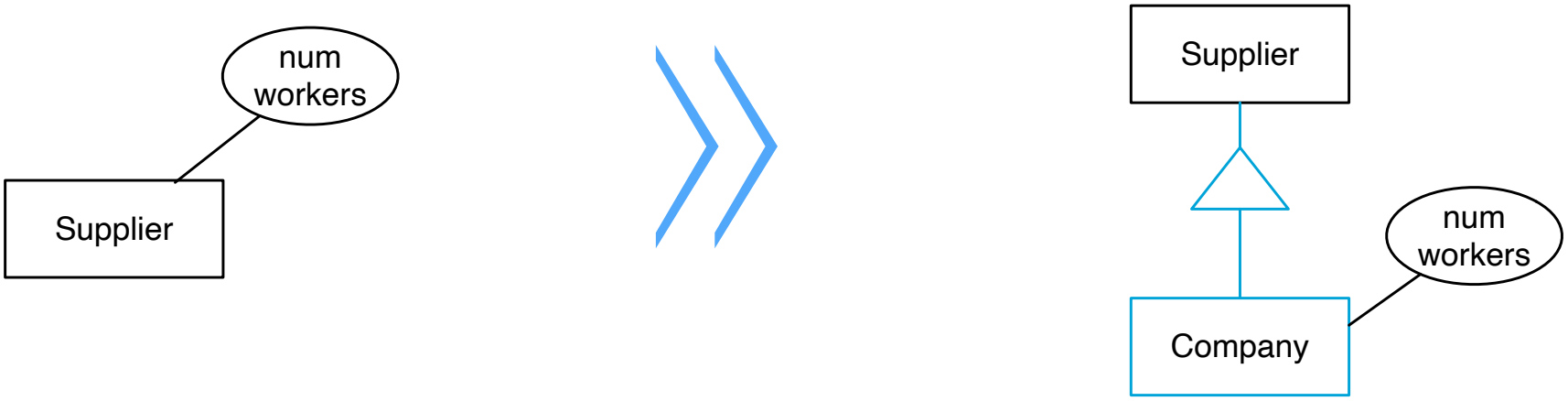


- Not every Employee manages (some are Managers, some are not, but that information is never captured)



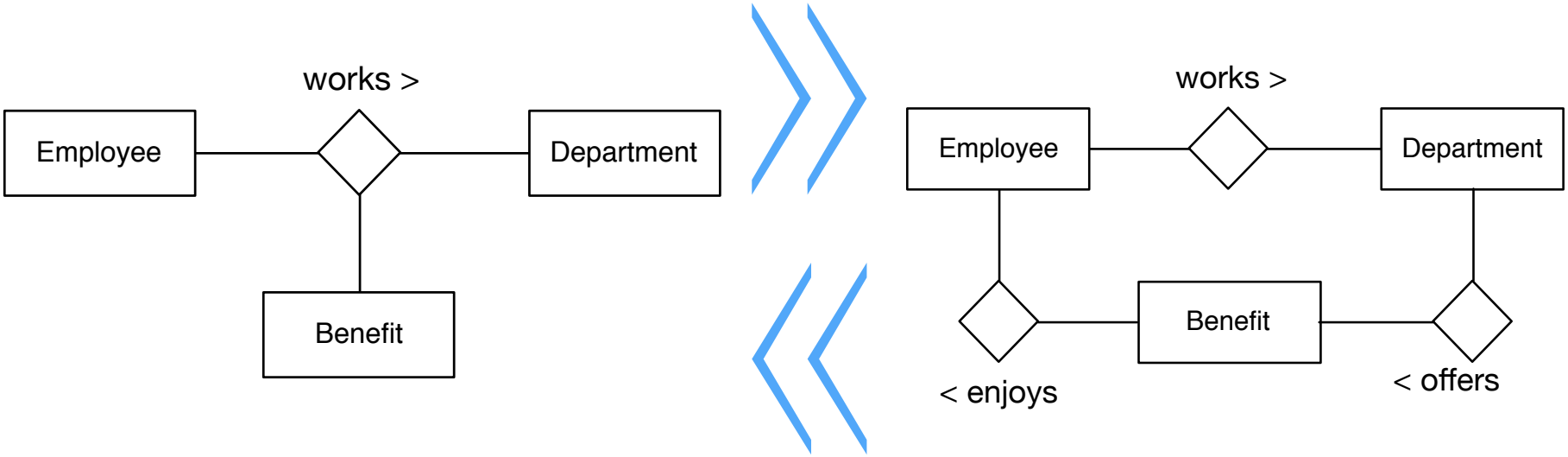
- Better: Every Employee in the role of a Manager manages a Department

Eliminate optional Attribute



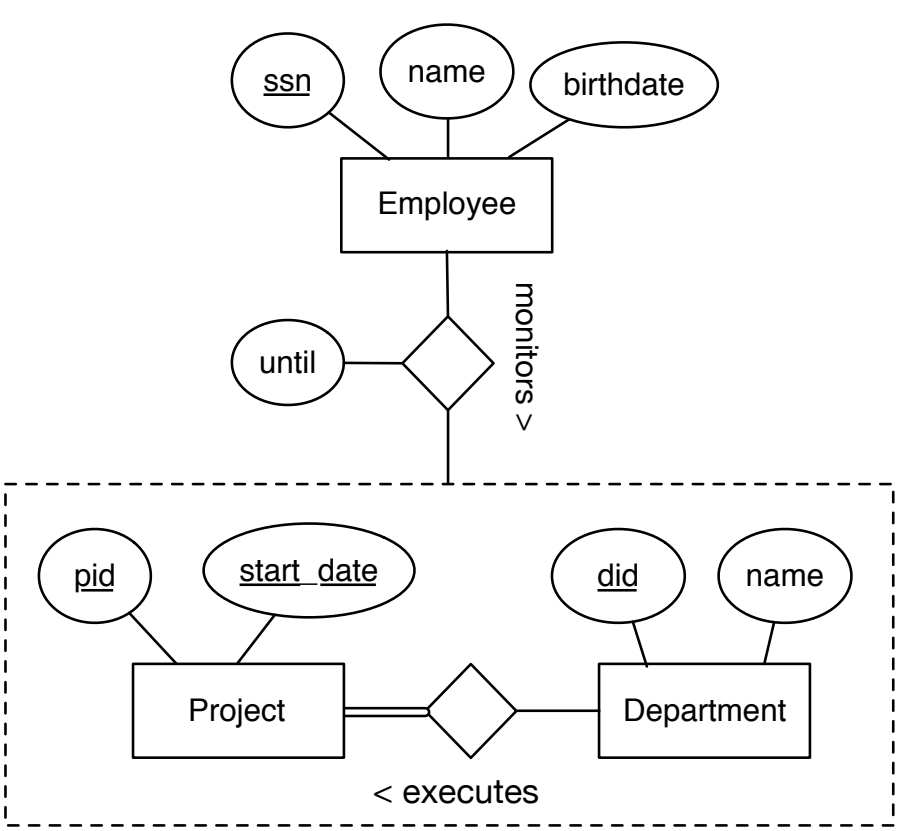
- An entity is modelled with an optional attribute. If an attribute is optional for an entity, this often means that a specialisation for that entity exists.
- In this example, the Company Name only applies when the Client is a Company. Therefore the the Client should be specialised optionally into a Company.

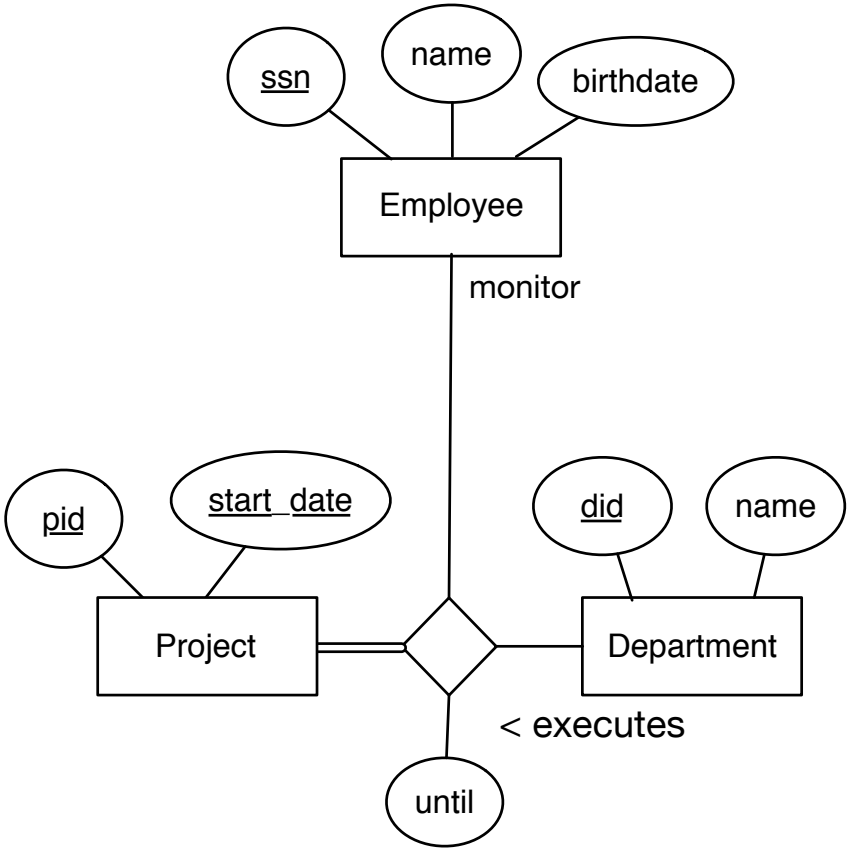
Ternary vs. many Binaries



- Every Employee must *(i)* have at least one Benefit to work in a Department, or *(ii)* a Department to have a benefit
- In this solution: Working in a department is independent from having a benefit

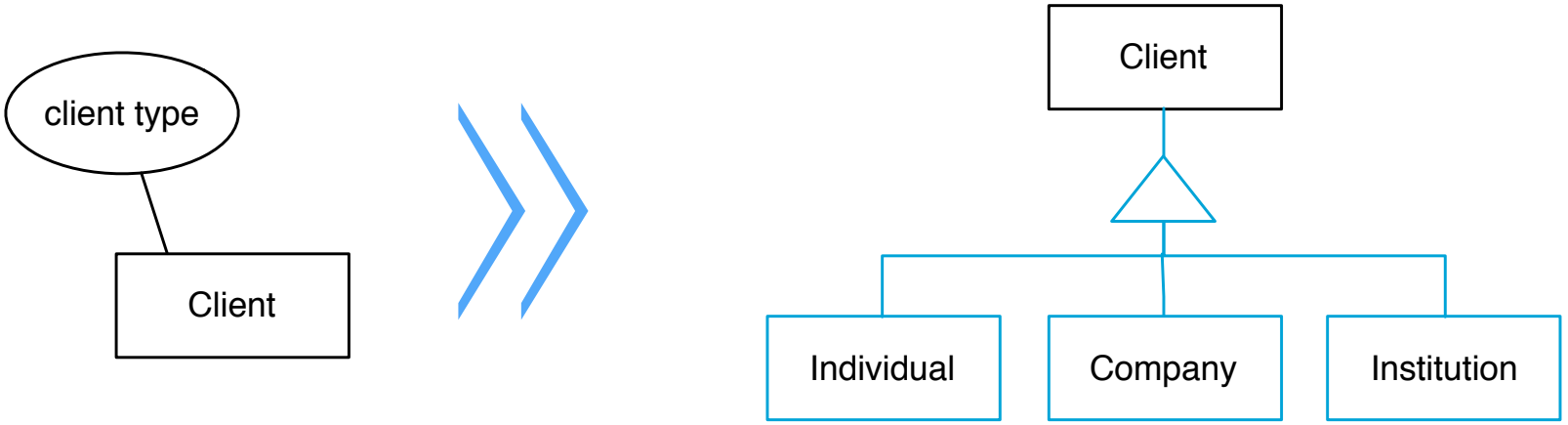
Ternary vs. Aggregation





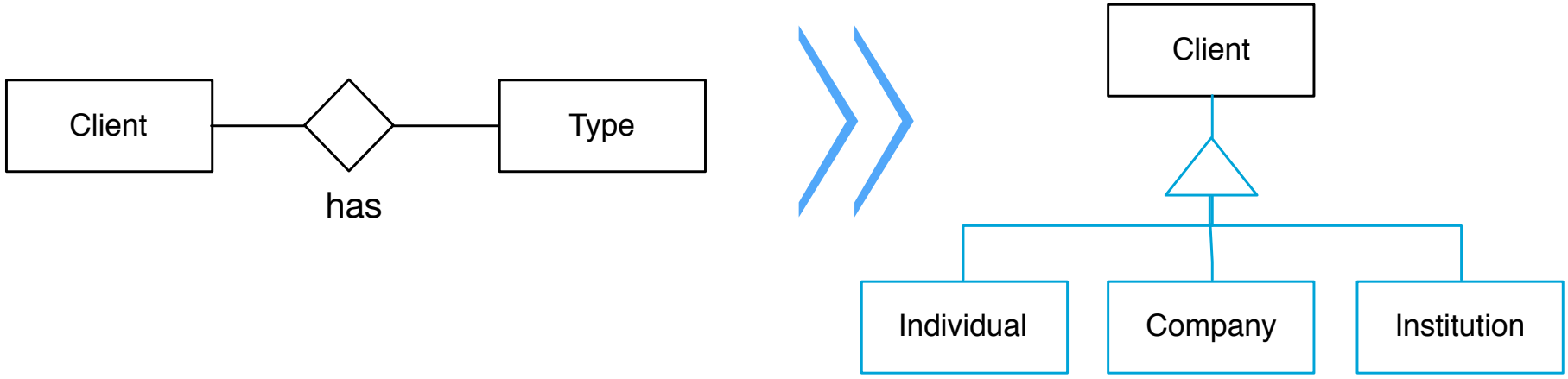
- Not all project executions are monitored
- All project executions are monitored

Replace ‘type’ Attribute with Specialisation



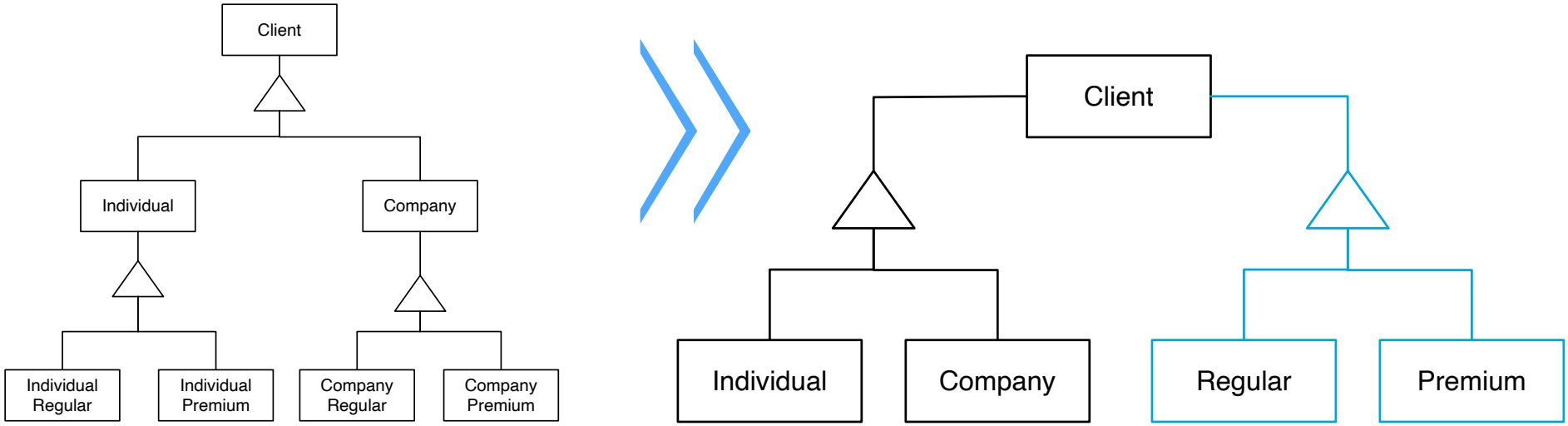
- A type attribute is used to distinguish between Client types. Instead, distinct client types should be modelled using a specialisation.
- Using 'type' attribute is only acceptable if there are way too many values for the attribute ‘type’, or the values are indeed unknown at modelling time.

Replace ‘type' Entity with Specialisation



- The type of a client is distinguished through an association with a Type entity. Distinct client types should be modelled using a specialisation. A 'type' entity is only acceptable if the type has attributes of its own.

Introduce Multi-way Specialisation



- When multiple specialisation relationships are used to model the combination of distinct sub-types of an entity type.
- Cluttering a domain diagram in this way can be avoided by using multiple specialisation.

Model Quality Checklists

Entities Checklist

ITEM	DESCRIPTION
The entity names are unique	There should not be any ambiguity when referring to an entity type in the system. Therefore, names must be unique.
The entity names are capitalised	The names os entities should have the fist letter capitalised (e.g. 'Client', 'Store', 'Company')
Attribute names are coherent	Attribute names are short and coherent across entities. The same attribute is never written differently (e.g. 'Desc' vs 'Description').
Each attribute is a qualifier or properties of the relation or entity	Attributes define placeholders of information for each individual of an entity type. Attributes that represent derived data (data can be derived from other attributes, such as a count of individuals or the age of an individual) should never be represented in the model. Also, attributes that point to information in other entities are not allowed.
Names of specialised entities can be read independently	The names of specialised entities should be readable independently. Suppose that 'Inner' and 'Outer' are specialisations of the entity 'Area'. Then, the names of these entities should be named 'Inner Area' and 'Outer Area'.

Association Checklist

ITEM	DESCRIPTION
The association name is a verb	An association captures a relationship. Relationships typically have names that are verbs. Very seldom an association should be named after a subject.
The association name is lowercase	The association name is all lowercase (e.g., **'participates'**, **'sells-to'**, **'manages'**,**' works-at'**)
The association name is meaningful	Generic names such as **'has'**, **'belongs_to'** should be avoided since they convey no useful information. Indeed, every association could be named 'has' or 'is_associated_with', but this would result in a useless diagram.
The association name has '<' prefix or '>' suffix	To make the direction of reading unambiguous.
The association name is unique	There should not be any ambiguity when referring to associations. Therefore, names must be unique.

Diagram Aspect Checklist

ITEM	DESCRIPTION
Graphic symbols are uniform in shape and size	Distinct sizes and shapes tendo to confuse the reader. Size, for example, can be wrongly interpreted as relative importance.
Entities are layed out in a way that facilitates diagram comprehension	In order to simplify the diagram reading entities should be spread ou logically enabling the diagram to be read left-to-right, top-to-bottom, or from the center outwards. Diagram layout should be uncluttered and convey clarity.
Lines are mostly orthogonal	The readability of a diagram with sloppy lines is low.
Lines cross-overs is avoided	Less line cross-overs clutter the diagram. Less cross-overs mean a more readable diagram.

Constraints Checklist

ITEM	DESCRIPTION
Cardinality, participation and specialisation constraints are explicitly included in the model diagram	This means that no constraints are missing. This is required to implement correct User Interface and database.
Alternative association circuits between entities have been double-checked for constraint inconsistencies	When two or more entities are can be connected by distinct association paths, these path must be coherent with one another in terms of constraints.
Entities and associations have been double-checked for domain constraint inconsistencies	Sometimes entities and associations can be created in ways that become inconsistent. These inconsistencies have to be verified, either by the database system or by the impelmentations of the functional requirements.
All domain constraints are placed with a numeric key '(n)' near the entities and associations they pertain to	This simplifies the process of maintaing the constraints, since it is much more difficult to maintain the constraints inside the diagram

Summary of Learning Objectives

You should be able to

- Identify Entities and the corresponding Attributes
- Identify Associations, associations with attributes, associations with more than two participants and recurrent associations
- Distinguish valid constraints, from constraints that cannot be modelled
- Identify Generalisations and their constraints
- Model Multiple- and Nested- generalisations
- Model Weak Entities/relationships and nested Weak Entities
- Identify and model Aggregation scenarios

You should be able to

- Identify missing attributes in entities, missing entities, missing associations, missing generalisations
- Question participation and cardinality constraints or the lack thereof
- Re-write models to make them more expressive
- Re-write models to make remove/minimize the number of written constraints