TÉCNICO LISBOA

Sistemas de Informação e Bases de Dados

Aula 03: Modelo Entidade-Associação

Prof. Paulo Carreira

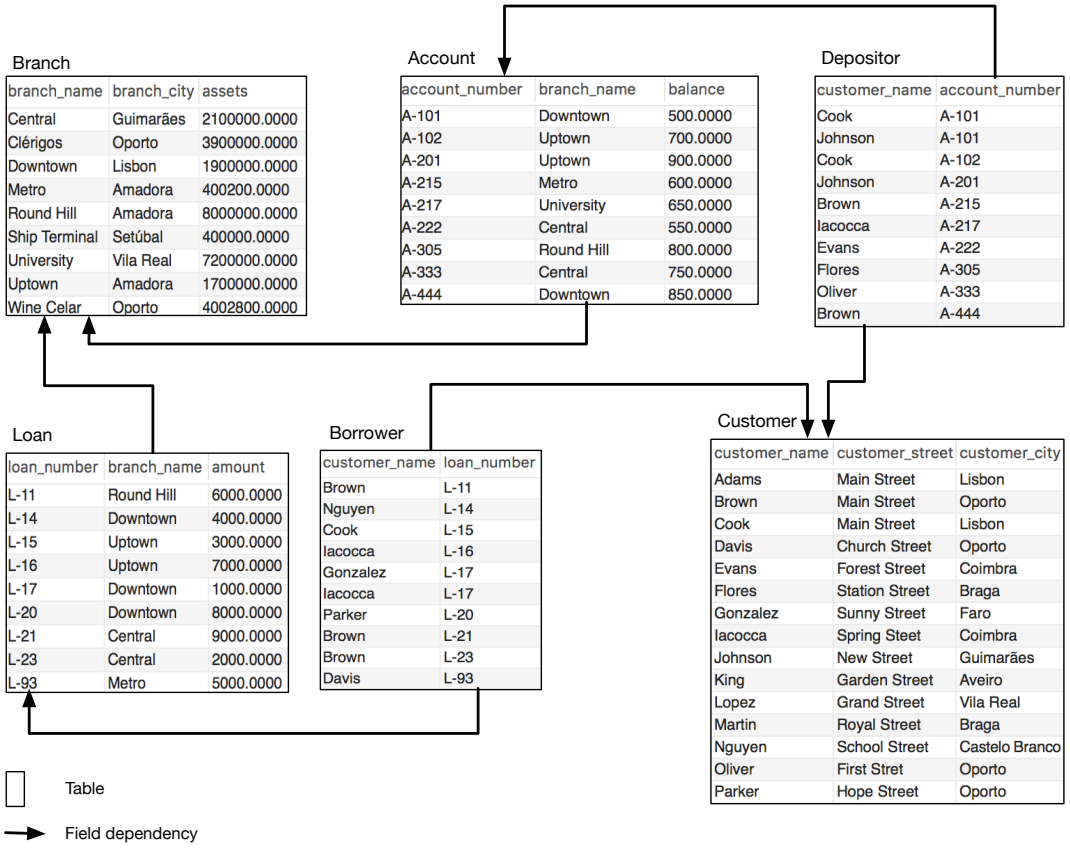
Introduction to Database Modeling

Database Design

Creating Databases and Applications

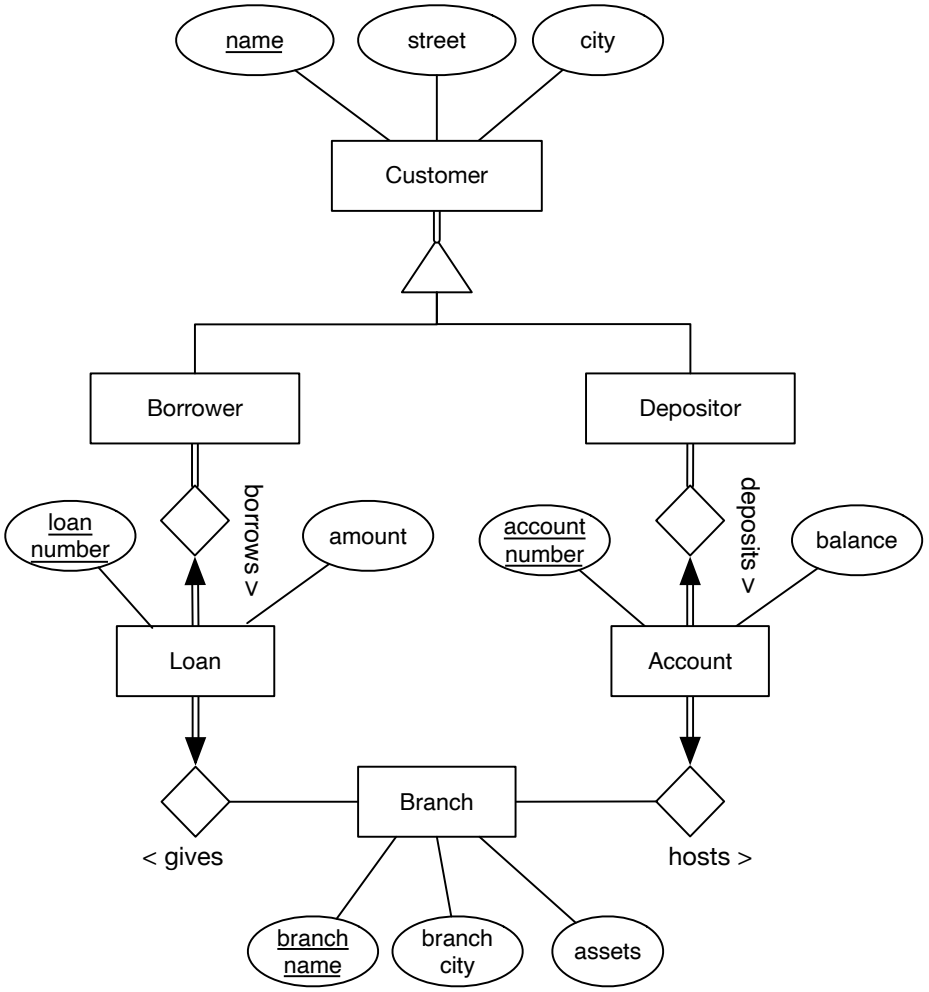
- How can we describe rigorously a very large database in terms of the information to be saved?
- What tables should be created and what are the relationships between them?
- What factors must be taken into account when deciding how data should be organised?

Tables of the Banking System Database

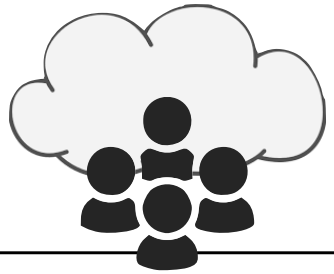


How do we figure out that these are the relevant tables (and not others)?

Banking System Database Model

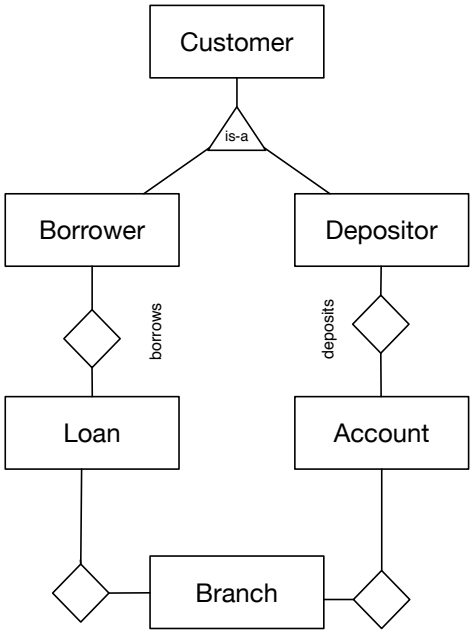


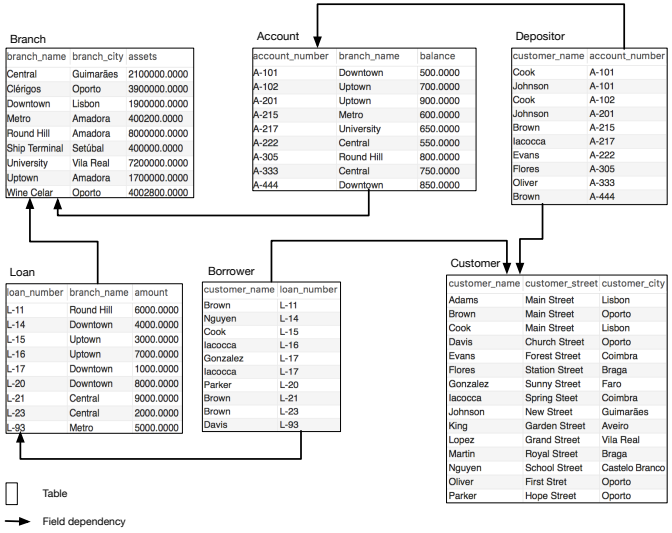
Determining the Tables for a Banking System



Requirements

- Functional Requirement 1
- Functional Requirement 2
- …
- **Integrity Constraint 1**
- **Integrity Constraint 2:**





Achieving a good design

What is a good database design?

How to arrive at the minimal set of tables that encode data without redundancy and can navigated to and aggregated to derive information?

Database Design

Good** database **design** **eliminates thousands of lines of code

https://rodgersnotes.wordpress.com/2010/09/14/database-design-mistakes-to-avoid/

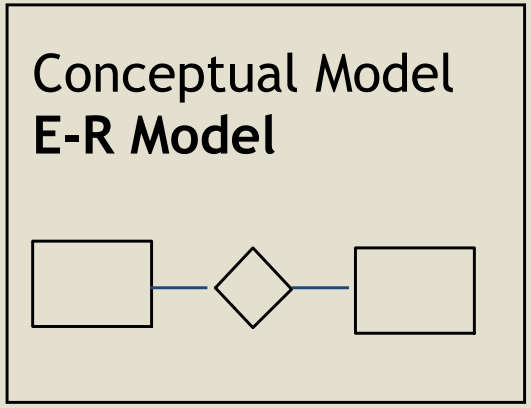
Three related ideas to achieve good (database) design

Idea 1: **Derive** the solution **automatically from the problem description** (Model-Driven Engineering)

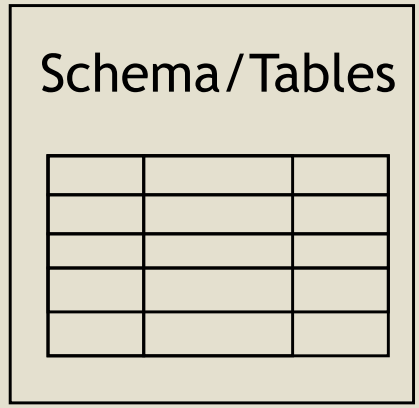
How to arrive at a Physical (tables) Model?

Conceptualisation

Requirements



Implementation



Problem Space

Solution Space

Idea 2: Model the problem and **defer** the solution

It pays to clearly define what the problem is!

Fifty-five minutes *defining the problem and only* ***five*** *minutes finding the solution.*

A. Einstein

The Problem is:

"To Know What the Problem Is"

> ***The definition of the problem will be the focal point of all your problem-solving efforts****. As such, it makes sense to devote as much attention and dedication to problem definition as possible. What usually happens is that as soon as we have a problem to work on* ***we're so eager to get to solutions that we neglect spending any time refining it****.* — A. Einstein

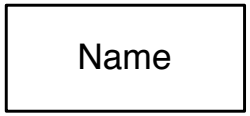
What most of us don't realize — and what supposedly Einstein might have been alluding to — **is that the quality of the solutions we come up with will be in direct proportion to the quality of the description of the problem we're trying to solve.** Not only will your solutions be more abundant and of higher quality, but they'll be achieved much, much more easily. Most importantly, you'll have the confidence to be tackling a worthwhile problem.

https://litemind.com/problem-definition/

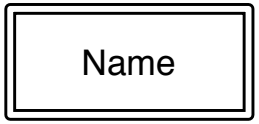
Idea 3: Use a **minimalistic** formal **language** to rigorously model the problem

E-A Modeling Notation

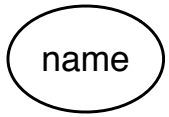
Basic Elements



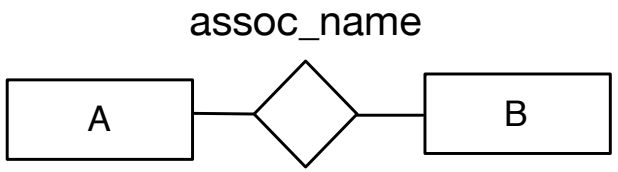
Entity



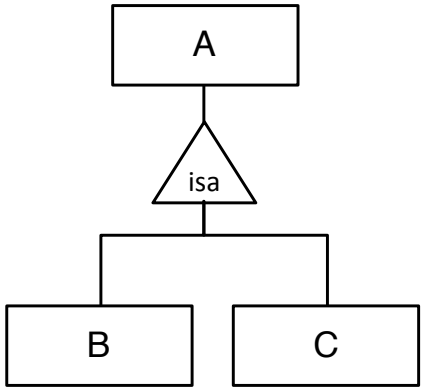
Weak Entity



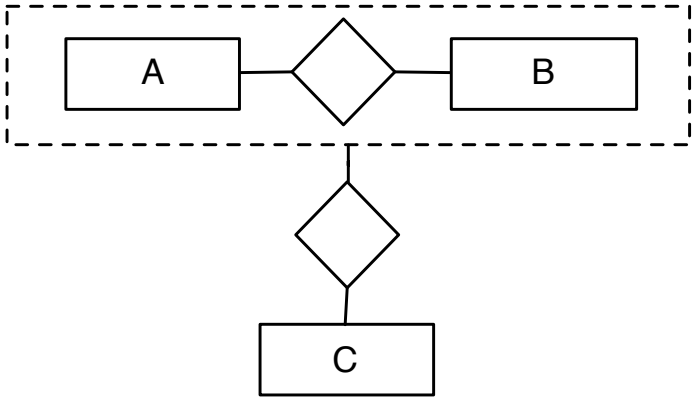
Attribute



Association



Generalization/
Specialization



Aggregation

Entity-Association Modeling

E-A modelling aims at **formalising** the information structure (only) of the problem domain focusing on **identifying** following three fundamental abstractions:

1. **Entities** of the domain
2. **Associations** between entities
3. **Constraints** that must be ensured by any correct implementation

Characteristics of E-A modelling language

1. High level language that enables the communication among the project stakeholders (very much like the blueprint of a house enables the communication between the owner, the architect and the builder)
2. A graphical language with few symbols and rigorous semantics that can be easily mastered (complex constrains still must be written textually)
3. Can be seen as a network of inter-related concepts that increases the understanding of the problem domain

Fundamental characteristics of E-A modelling language

- **Rigorous:** A diagram is formal description of a given reality (domain) using a language with a well defined semantics
- **Cognitively effective:** Creates descriptions that are simple to understand and manipulate. The abstract representations of reality are expressed in a minimalistic language (with a reduced number of concepts: Entities, Associations, Constraints) and purged of superfluous detail regarding the problem to be solved.

Fundamental of E-A Modelling: Entities and Associations

Entities

Entity

Definition

An **Entity** or (**Entity Type**) represents a concept for which distinct set of *objects* (*instances*, *exemplars* or '*individuals*') exist in the real world

E-A Graphic Language

Entity

An **Entity (Type)** of the domain representing a concept for which multiple distinct objects may exist (and corresponding data must be recorded)

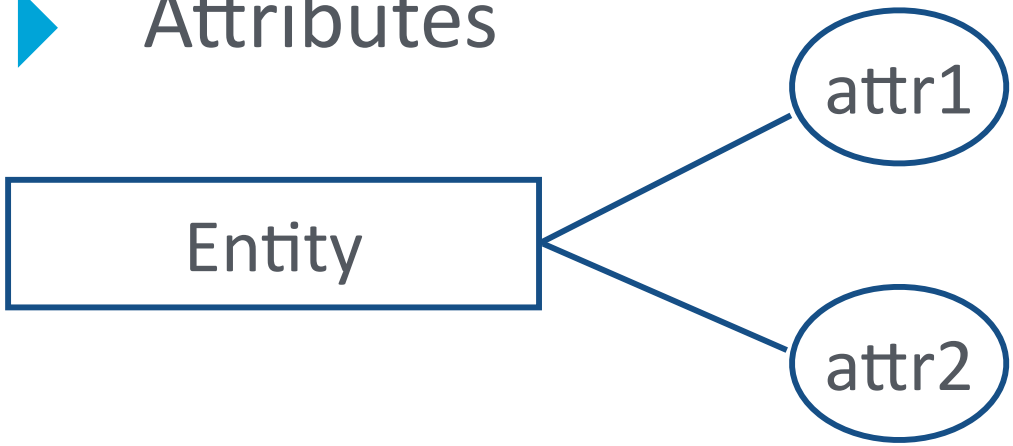
For example, 'Employee' or 'Invoice' are entities in the sense that the system may keep information regarding, respectively, multiple 'employees', and multiple 'invoices'.

NOTE: Entities whose that are not to be recorded should not be modeled.

Example Entities

E-A Graphic Language

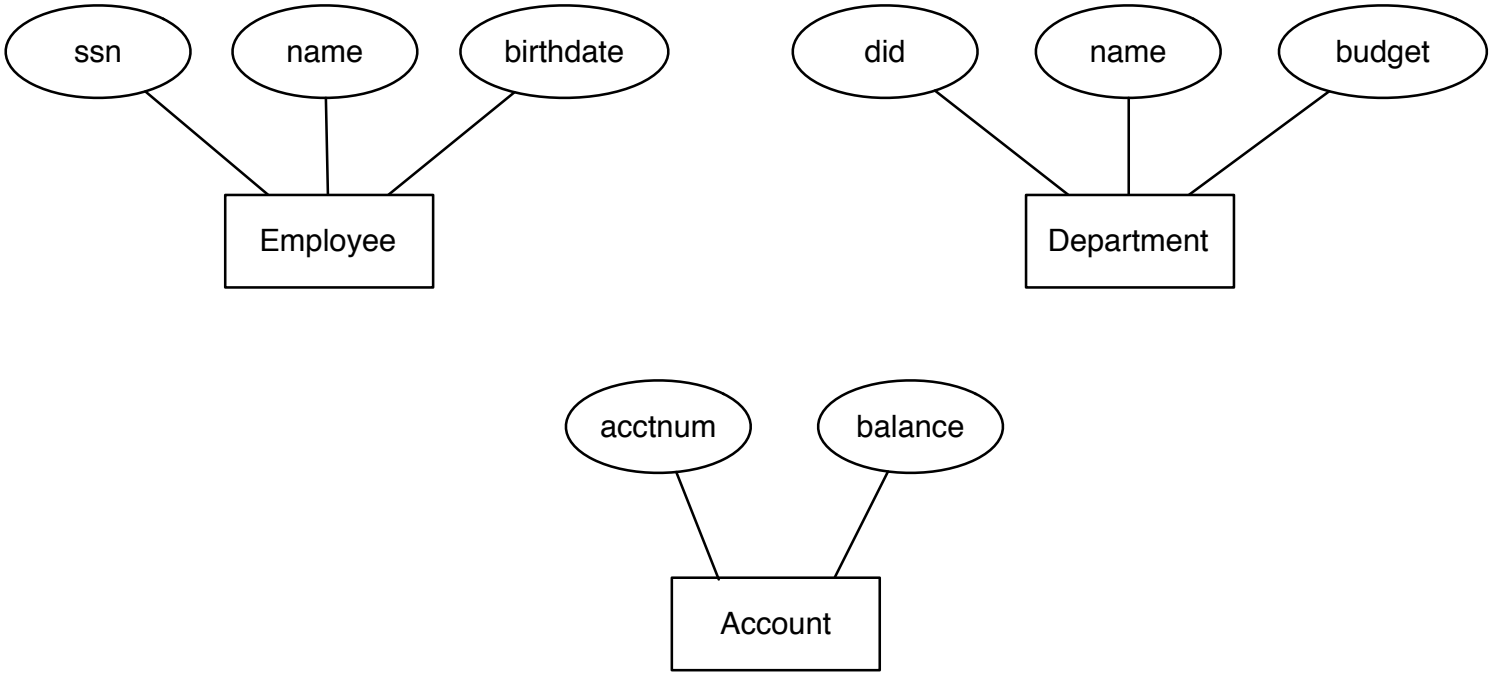
- Attributes



Attributes are characteristics of entities that represent the information (data values) to be captured for each instance of an entity set

Each instance has one value for each attribute.

Entities with Attributes (Examples)



The choice of attributes reflects the level of detail with which we want to represent information about entities

Entities

- Given a set of **atributes** $A_1 \ldots A_n$ where
  - Each atribute A_i is associated a **domain** of possible values $D(A_i)$

- **An Entity E** is: a set of object with a similar information structure:
  - $\mathbf{E} = \{(v_1, \ldots, v_n) \mid v_1 \in D(A_1), \ldots, v_n \in D(A_n)\}$
  - Where each element $e \in \mathbf{E}$ is an **instance** of **E**

Associations

Association

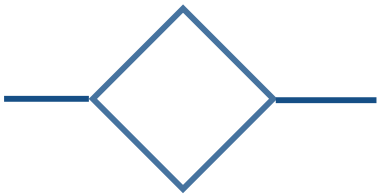
Definition

An **Association** between two or more entities represents all possible relationships between instances of the entities involved in the association

- Binary Association: two entities…
- Ternary Association: three entidades …
- Participating entities may be distinct or not (auto-association)
- Can have descriptive **attributes**

E-A Graphic Language

- Binary Association



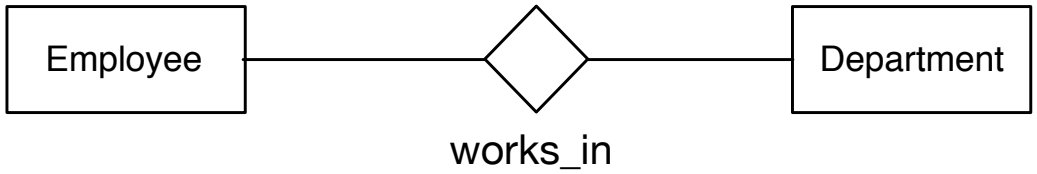
association name

A **Binary Association** between two entities capturing the relationship between them in the sense that instance of the distinct entities may can be related to one another through the association.

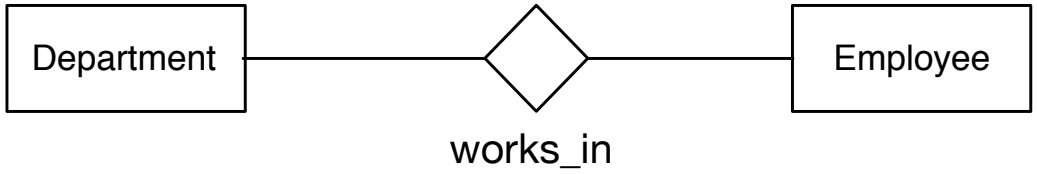
Note: Association = Relationship

Commutativity of Associations

The information represented by:

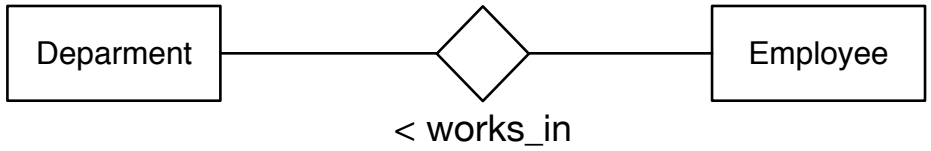


Is the same as:



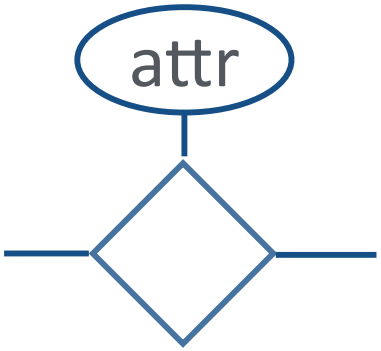
Naming Associations

- Typically a verb
- Should be unique in the diagram because:
  - The model is a communication tool and therefore, there should be no ambiguity when referring to an association
  - If names were not unique, in the limit all associations could have the name, rendering the model useless
- Whenever the direction of reading is not obvious, it can be made clear by using ’>’ or ‘<'
- Lowercase (convention)



E-A Graphic Language

- Association with attribute



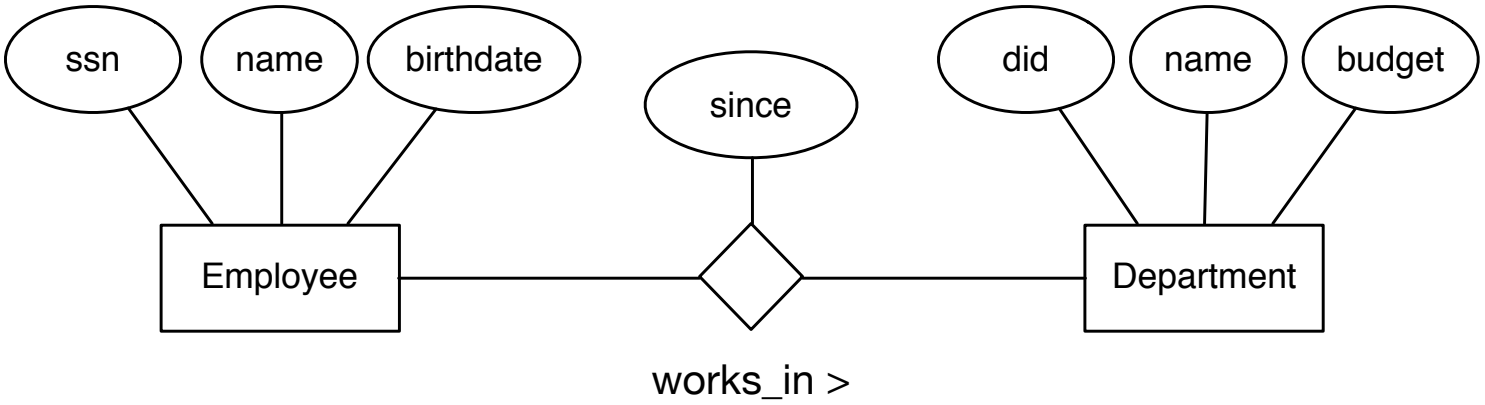
Association Name

An **association** with an attribute captures additional information that characterises the relationship being established.

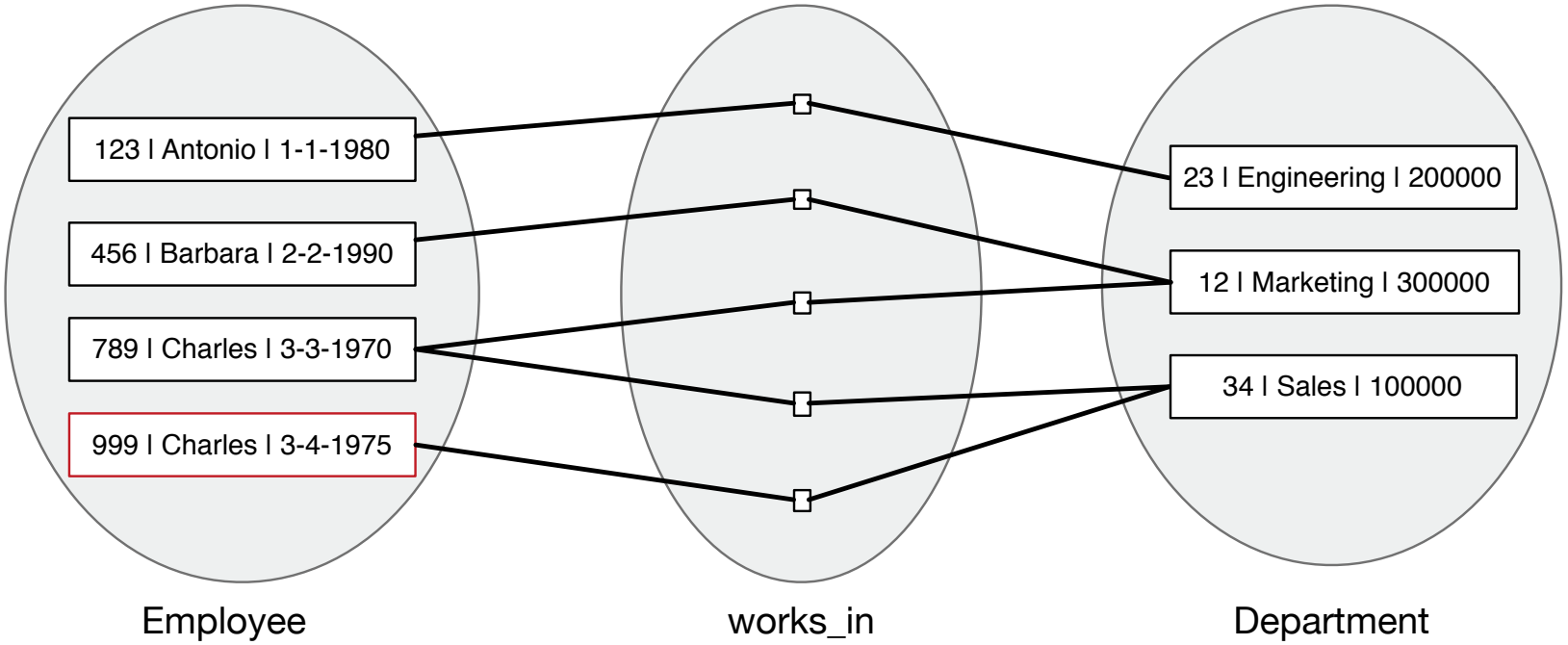
Association

- Given the entities E_1… E_n and the attributes A_1…A_m
- An **association A** is a set of elements:
  - $A = \{(e_1, ..., e_n, v_1, \ldots, v_m) \mid e_1 \in E_1, ..., e_n \in E_n \wedge v_1 \in D(A_1), ..., v_m \in D(A_m)\}$
- Where each element $a \in$ **A** is an **instance of the association A.**

Associations with attributes (Example)

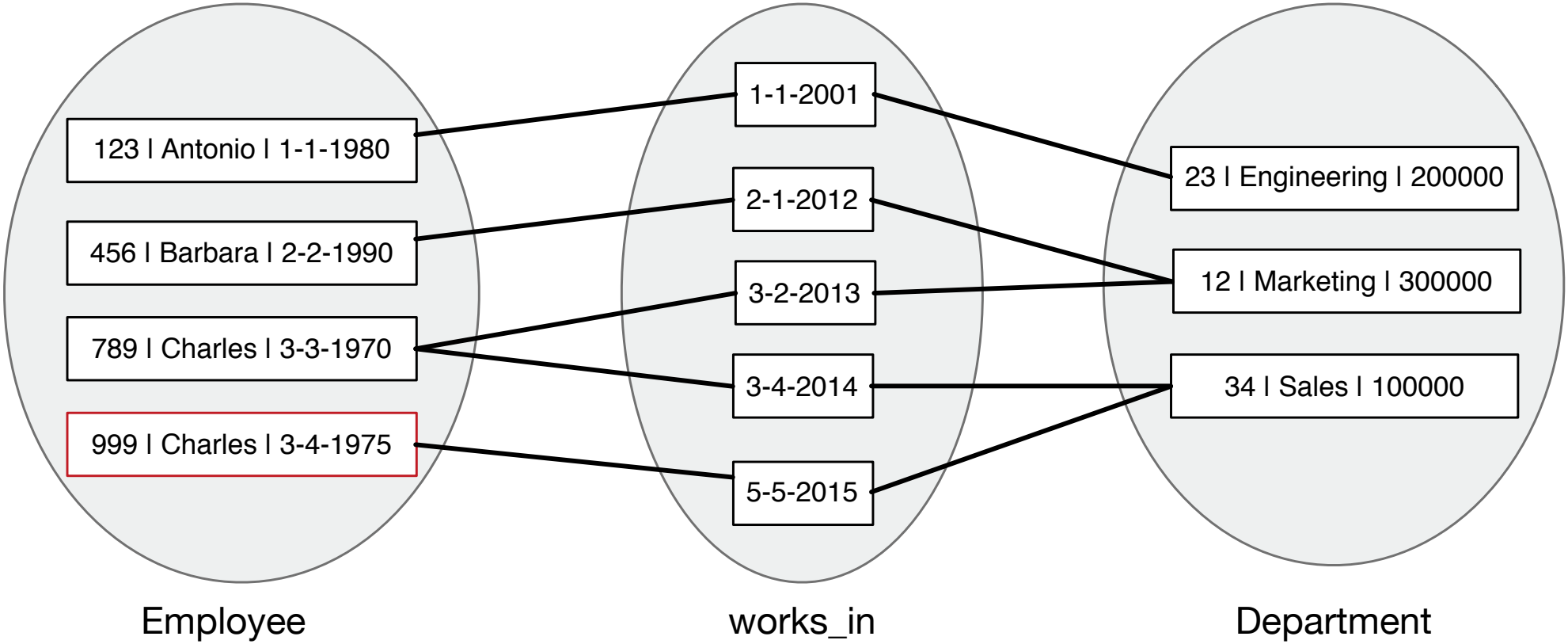


Instances of an Association



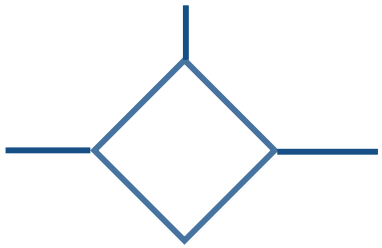
Associating an Employee to **multiple** departments is the same creating multiple associations of the 'works_in' association.

Instances of an Association



E-A Graphic Language

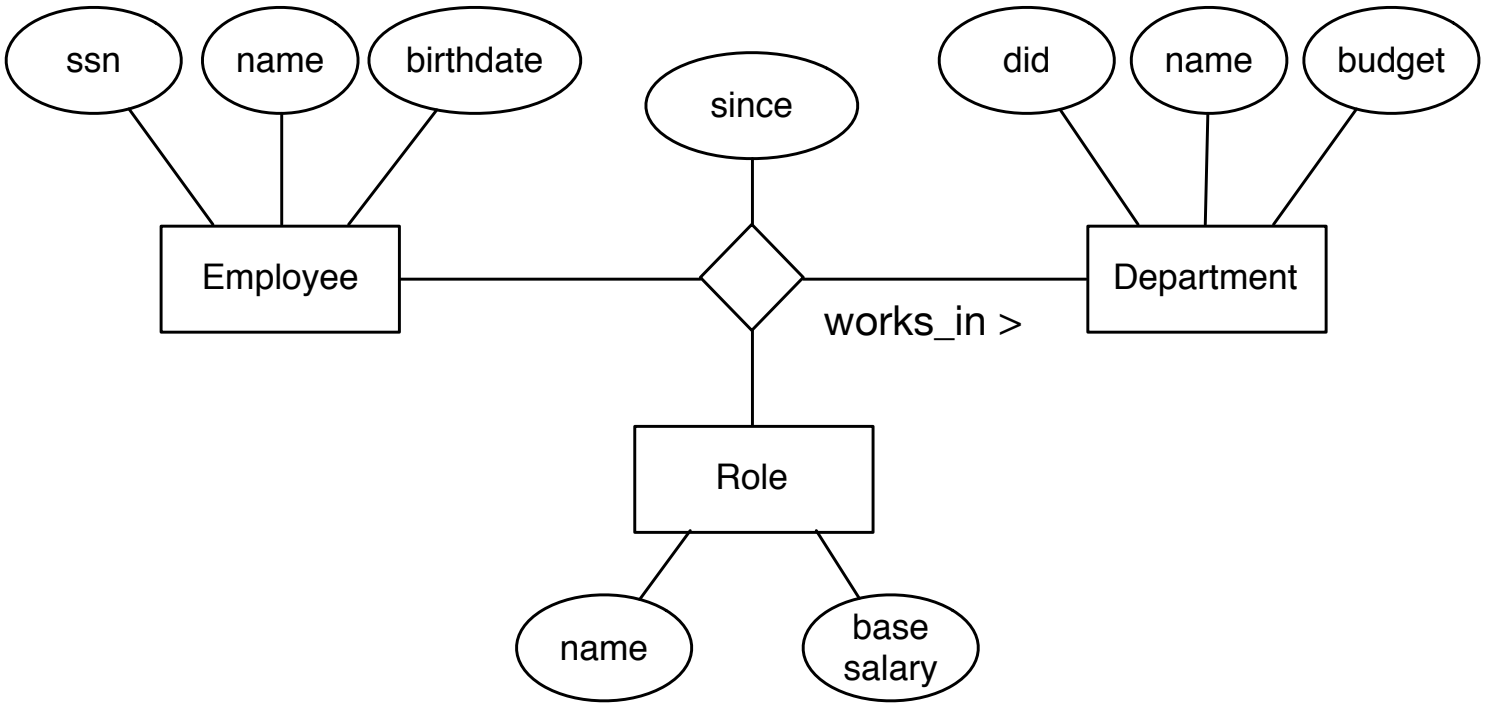
- Ternary Association



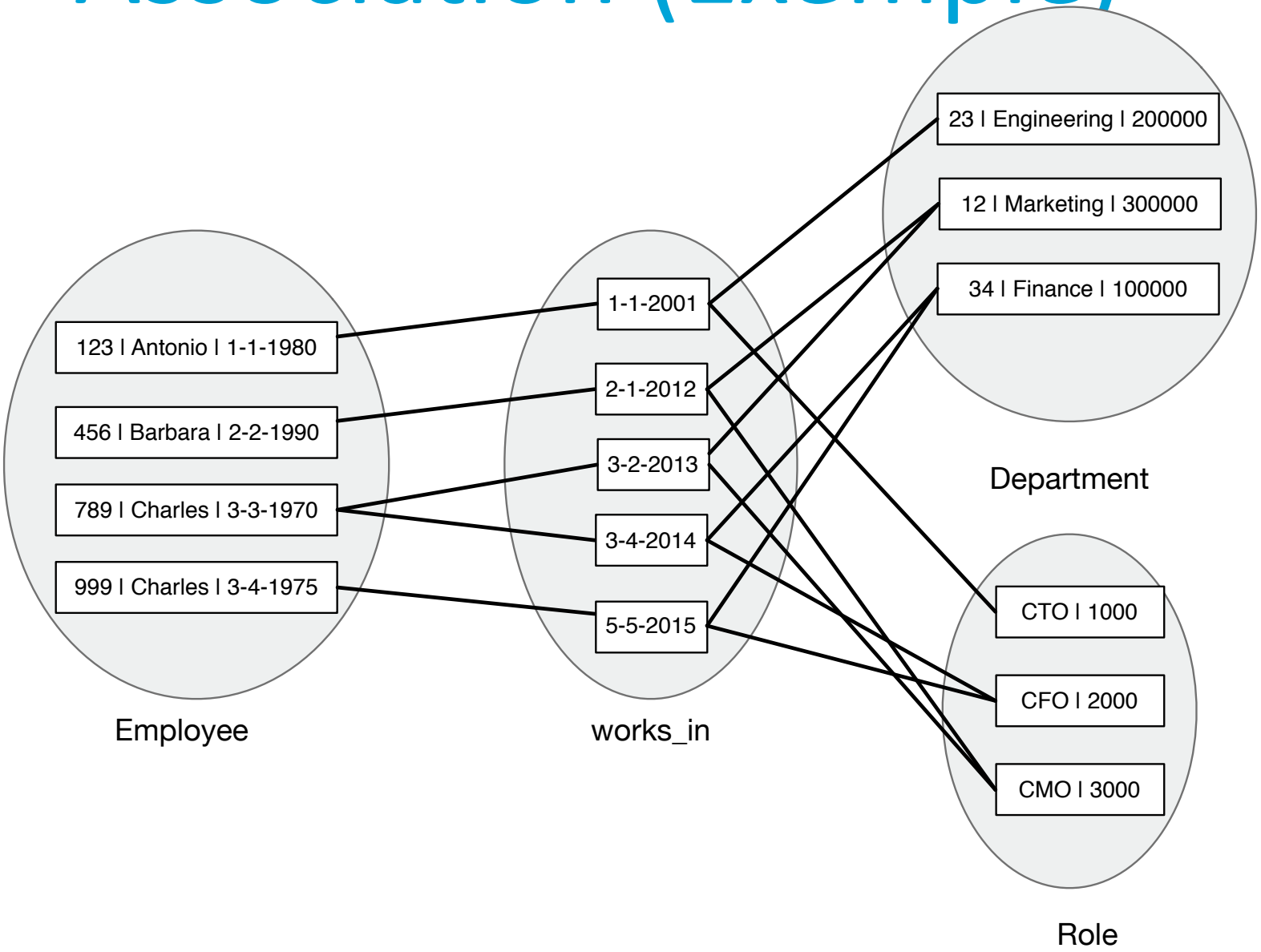
Association Name

A **Ternary Association** captures a relationship between three entities meaning that individuals of each of the three entities may be related through the specified association named 'assoc name'.

Ternary Association (Example)



Instance of a Ternary Association (Exemple)

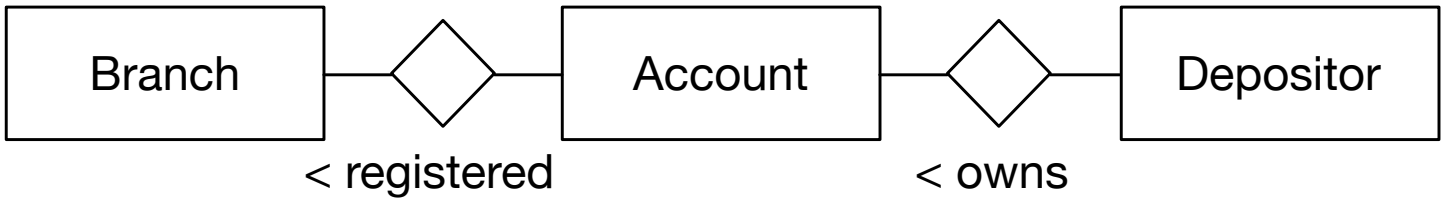


Exercise: Modelling the Bank Domain

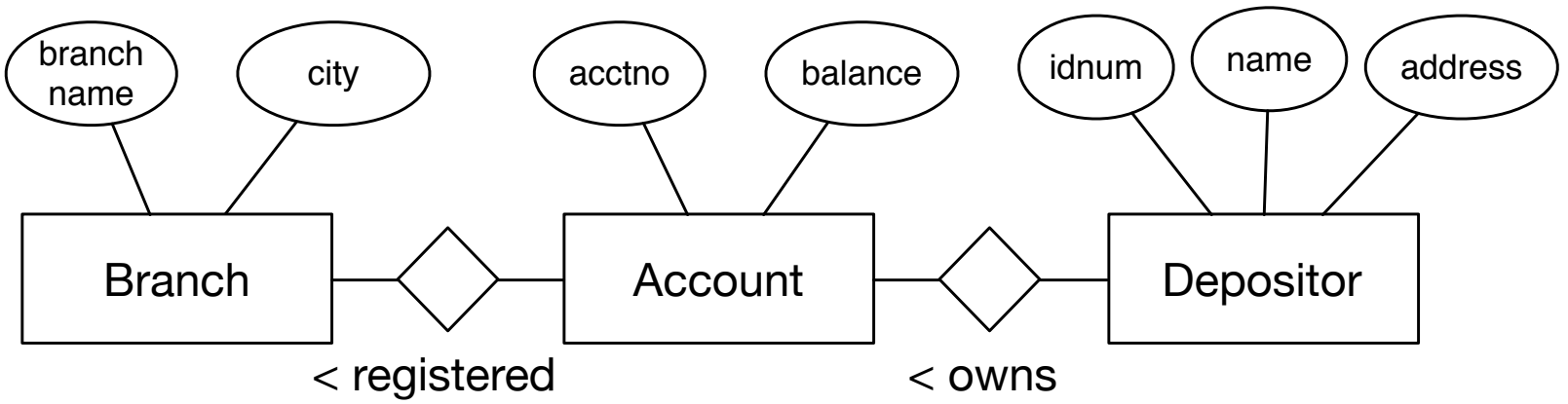
Domain Requirements

- Store the information regarding *accounts* with attributes *acctnum* and *balance;*
- Every *account* is owned by a *depositor* with attributes *id card number*, *name* and *address;*
- Accounts are registered in *branches*. A branch is characterised by a *branch name* and a *city;*
- It is necessary to know which *depositors* have an *account* in the *city* where they live.

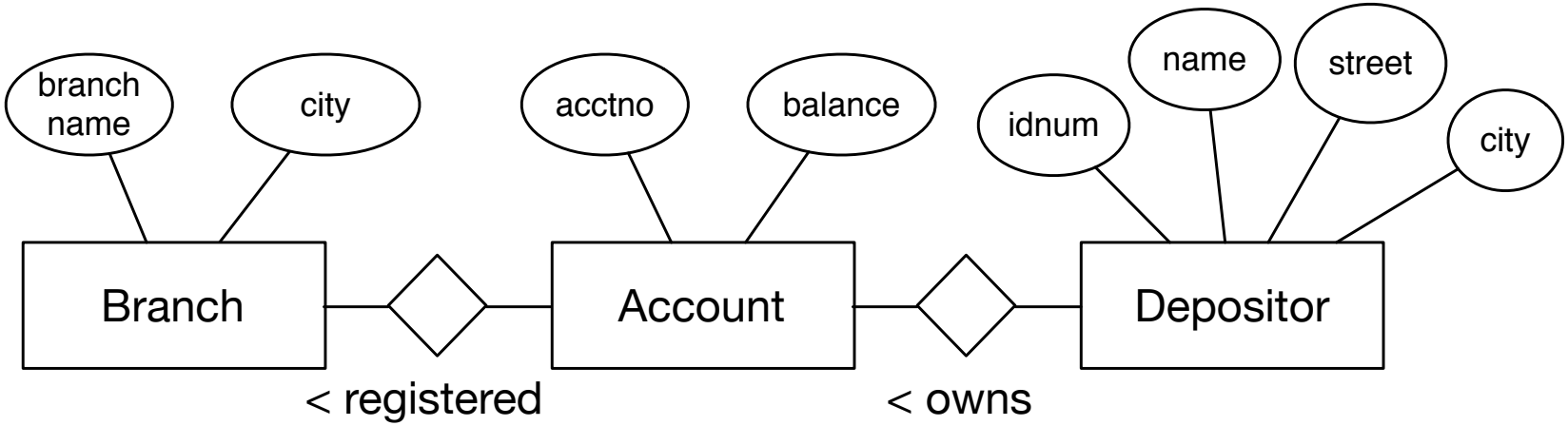
Bank Partial Solution - 1



Bank Partial Solution - 2



Bank Partial Solution - 3



Requirements that cannot be formalised on E-A

1. Requirements that are not information requirements (e.g., user wishes, considerations on the graphical aspect of the system)
2. Requirements for which it is not possible to determine the attributes
3. Functional requirements that reflect the behaviour/functionality of the system (e.g., "*the system must do X*")

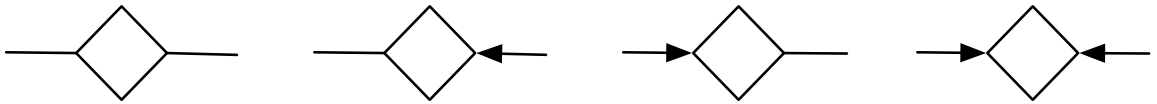
Modeling Constraints

E-A Modeling Notation

Key Constraints

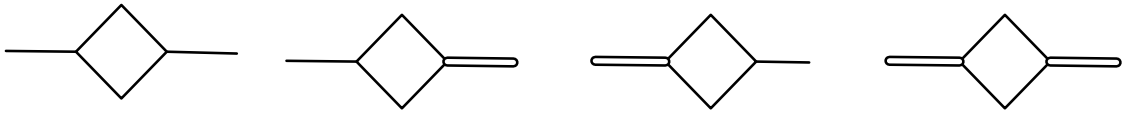
Key Partial Key

Cardinality Constraints



Many-to-many One-to-many Many-to-one One-to-one

Participation Constraints



Optional Mandatory (R) Mandatory (L) Mutual

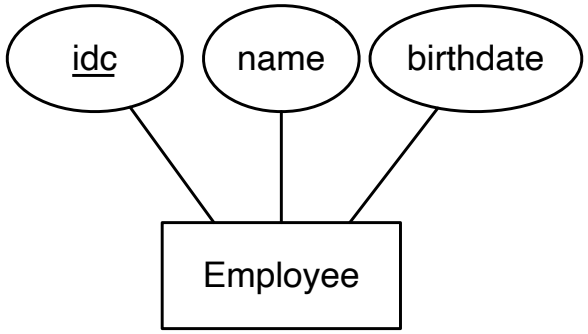
Key Constraints

Keys of Entities

Definition

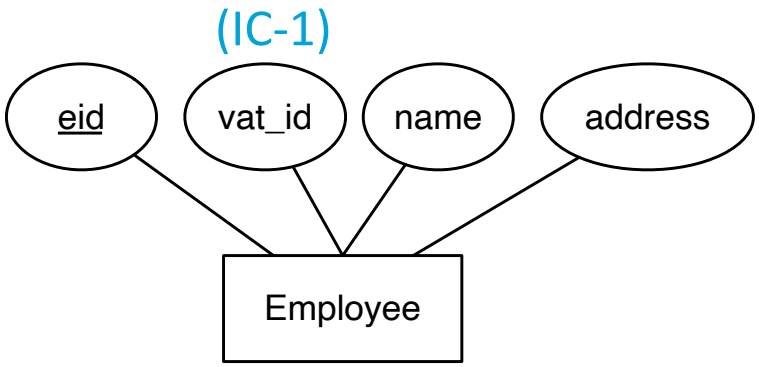
A **key** is the minimal set of attributes whose values uniquely identify each instance of the Entity

- The key is underlined
- The values of key attributes are Unique, i.e., their values cannot repeat



Candidate Keys

- There may be several attributes (or sub-sets of attributes) that define the keys, that is:
  - There may be multiple **candidate keys**
  - Of the Candidate Keys, one chosen to be **primary key**
- If there are multiple candidate keys, only the primary key is represented graphically in the model, the others are described as a textual RI.



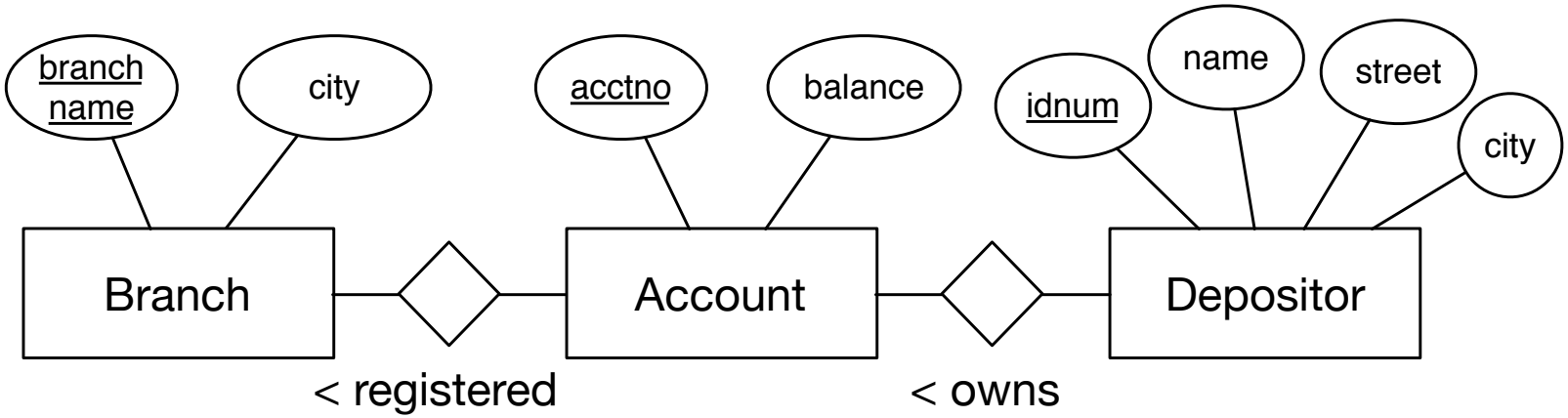
(IC-1) Vat ids are unique

How to select the key

1. The **shortest key** (but not necessarily)
2. The **most recognised** key for domain users
3. The key is the atribute (or set of atributes) that **determine the values of the remaining** atributes (i.e., does not depend on any other attribute.

For example: 'phone' depends on 'address' but 'address' does not depend on ‘phone’. Therefore ‘address’ is likely to be the key.

Bank Partial Solution - 4

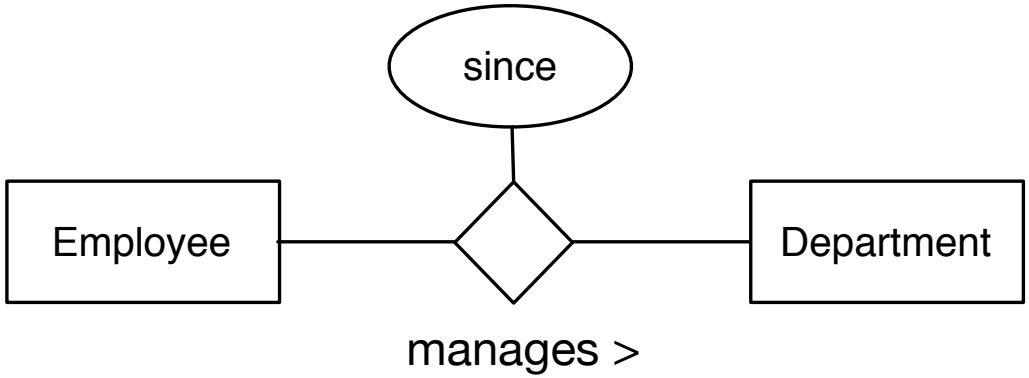


Anti-Heuristics for Entities

1. **Entity with only one attribute:** May not be an entity (possibly the attribute belongs to another entity)
2. **Keyless Entity**: It may not be an entity but an association

Multiplicity Constraints

No constraints



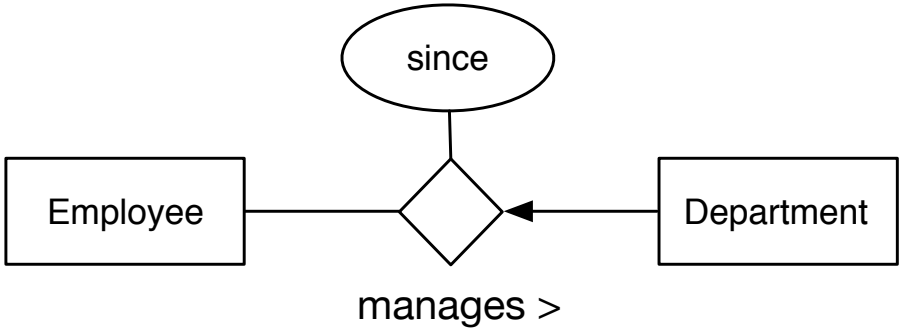
- A department can be managed by how many employees?

A: None, one, or many

- An employee can manage how many departments?

A: None, one, or many

One multiplicity constraint



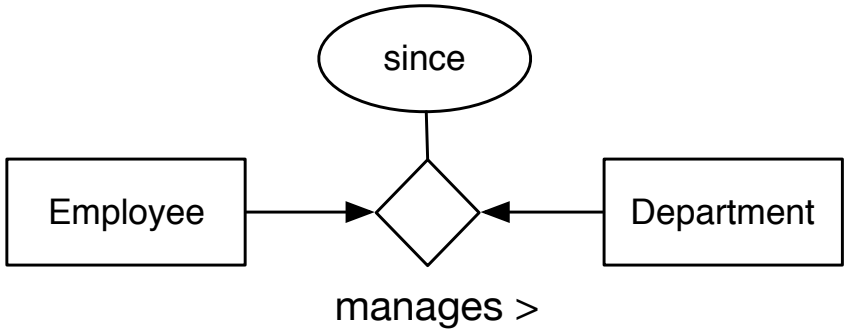
- A department can be managed by how many employees?

A: Atmost one

- An employee can manage how many departments?

A: None, one, or many

Two multiplicity constraints



- A department can be managed by how many employees?

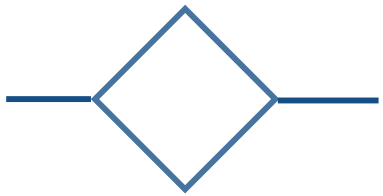
A: Atmost one

- An employee can manage how many departments?

A: Atmost one

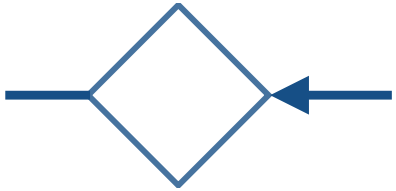
E-A Graphic Language

- Many-to-many (no constraints)



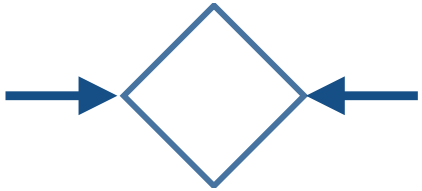
Each instance of either entity may be related many (as well as to none, one, or all) instances of the other entity.

- One-to-many



Each instance of the left entity is associated to potentiality many instances of the entity on the right. Each instance on the right can be associated with at most one instance from the left entity.

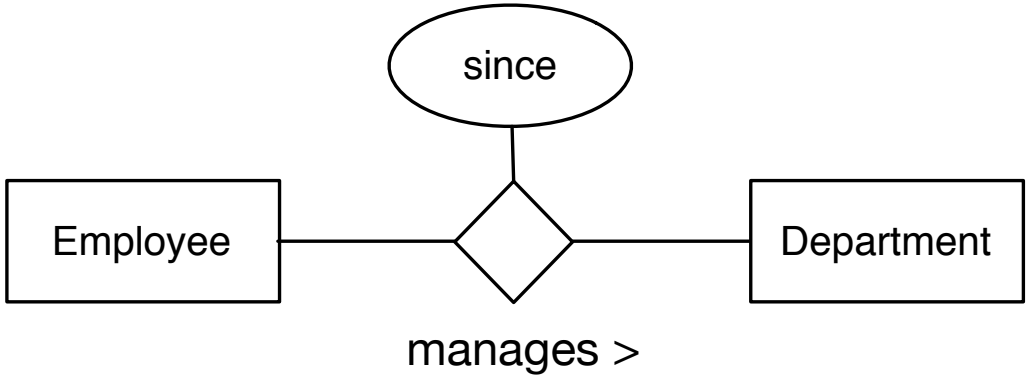
- One-to-one



Each instance of the left entity must be associated with at most one instance of right entity. Moreover, each instance of the right entity can be related with at most one instance of the left entity.

Participation Constraints

No constraints



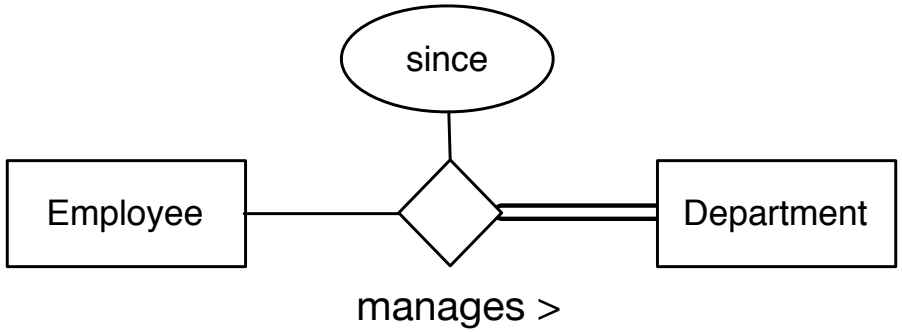
- Can a Department exist without a manager?

A: Yes

- Can an Employee exist that is not a manager?

A: Yes

Mandatory on one side



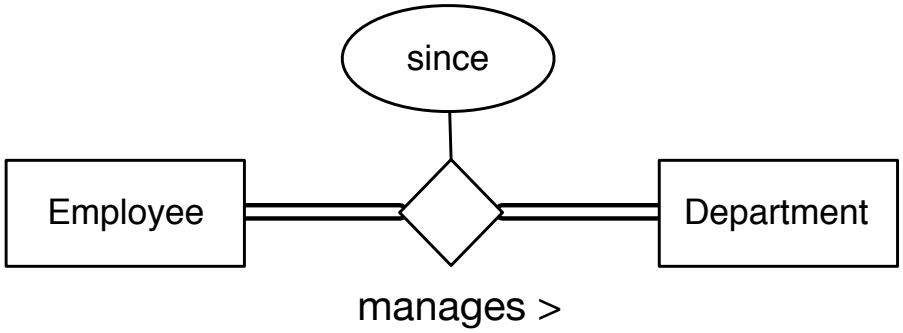
- Can a Department exist without a manager?

A: No

- Can an Employee exist that is not a manager?

A: Yes

Mandatory on both sides



- Can a Department exist without a manager?

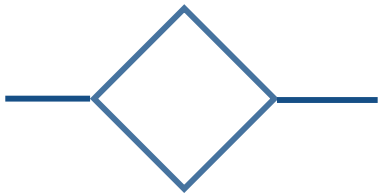
R: No

- Can an Employee exist that is not a manager?

R: No (all employees must necessarily be managers)

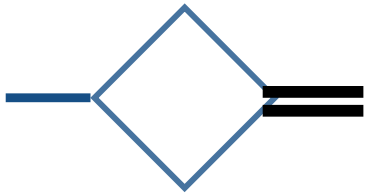
E-A Graphic Language

- Optional (or partial) participation (no constraint)



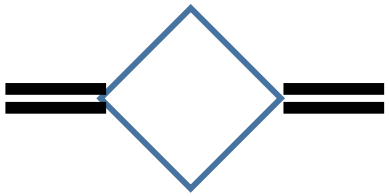
Instances of either entity type can exist in the system without necessarily participating in the association.

- Mandatory (or total) participation



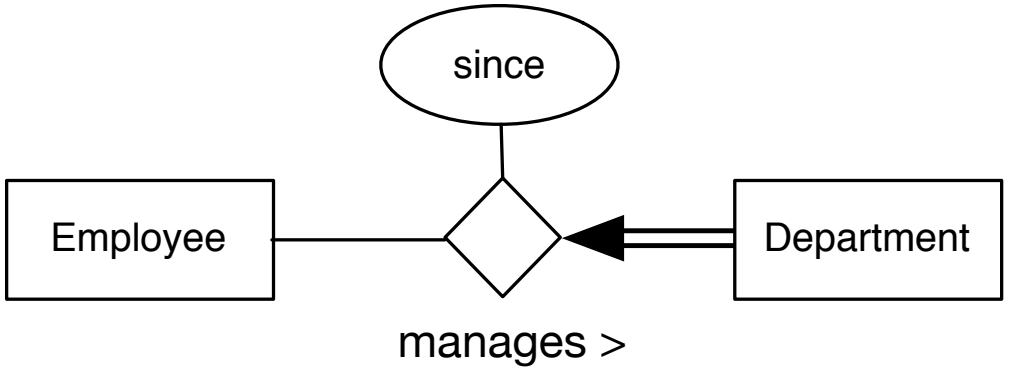
Each instance on the right-hand entity type must participate in the association in order to exist in the system.

- Mutual (total on both sides) participation



Instances of both entities must participate in the association.

Combining the constraints



- Can a Departament exist without a manager?

R: No

- How many managers can a Department have?

R: Atmost one