Database Tutorial 1: Entity Relationship Diagrams

with extra notes!

2015-01-27

Disclaimer: Usually several solutions are possible. The solution presented at the tutorial as well as the exam sample solution will be shown here, with a short explanation of the motivation for the presented solution. In doubt just state your motivation for decisions taken in the exam.

1. (11 points) Modeling and Design

flight code	airline	prime flight	operating airline	departure city	departure airport	destination city	destination airport	aircraft type	seats
SK111	SAS	SK111	SAS	Gothenburg	GOT	Frankfurt	FRA	B737	140
LH555	Lufthansa	SK111	SAS	Gothenburg	GOT	Frankfurt	FRA	B737	140
AF111	Air France	AF111	Air France	Gothenburg	GOT	Paris	CDG	A320	170
LH111	Lufthansa	LH111	Lufthansa	Frankfurt	FRA	Paris	CDG	A321	200
LH222	Lufthansa	LH222	Lufthansa	Frankfurt	FRA	Malta	MLA	A320	170
AF222	Air France	AF222	Air France	Paris	ORY	Malta	MLA	A320	170
AB222	Air Berlin	AB222	Air Berlin	Frankfurt	FRA	Munich	MUC	A320	170
KM111	Air Malta	KM111	Air Malta	Munich	MUC	Malta	MLA	A319	140
LH333	Lufthansa	KM111	Air Malta	Munich	MUC	Malta	MLA	A319	140
SK222	SAS	KM111	Air Malta	Munich	MUC	Malta	MLA	A319	140
AF333	Air France	AF333	Air France	Paris	CDG	Frankfurt	FRA	A320	170

We assume the following (slightly simplified) conventions for this domain:

- the "flight code" attribute determines all other attributes on a row
- the "prime flight" is the flight code used by the airline operating the flight; the "flight code" in the first column can thus belong to another airline that has a code sharing agreement with the operating airline
- the "prime flight" appears in the table as a "flight code" as well, having itself as prime flight
- each airport has a unique code
- every aircraft of the same type has the same number of seats

(It is a common practice that one and the same flight can be booked using different airlines. Each airline uses a different "flight code", but the passengers end up in the same plane. The code used by the actual operating airline is called the "prime flight" code. For example, whether you book flight LH333 with Lufthansa or flight SK222 with SAS, you end up in the plane of Air Malta flight KM111.)

(a) (2 points) Find at least four redundancies in Table 1.

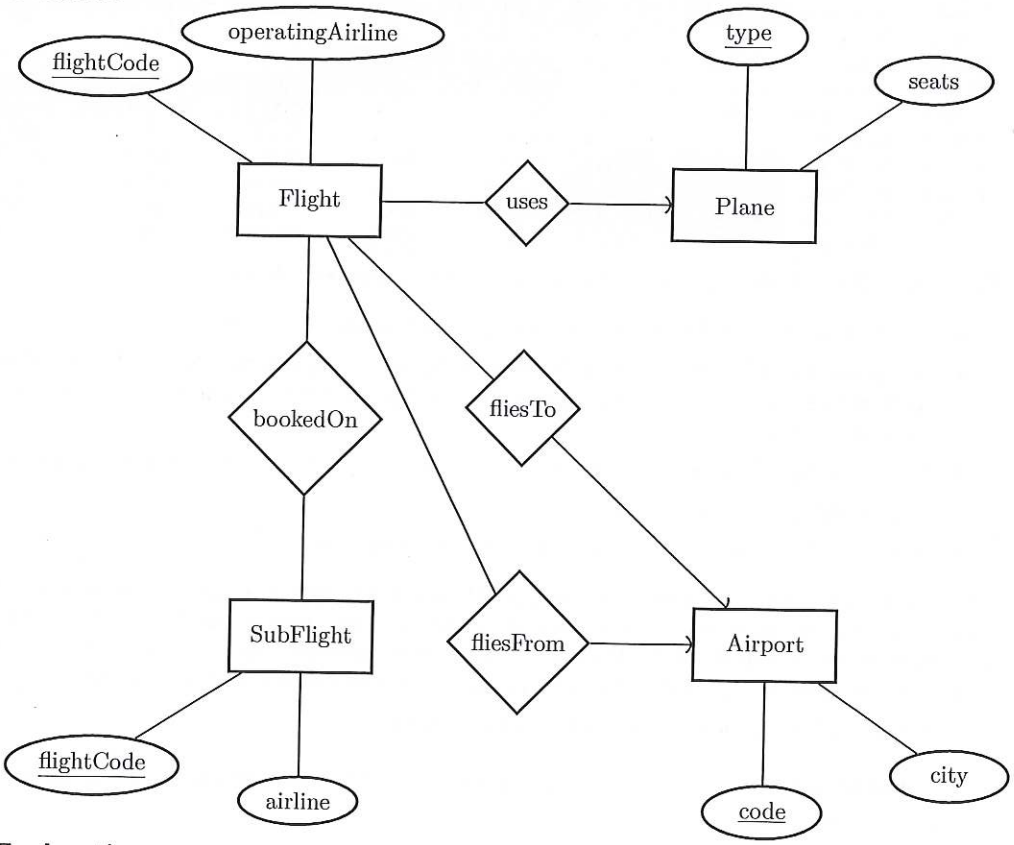
> **Definition** *Redundancy: repetition of the same information (in particular, of the same argument-value pair)*

Solution:
See colored cells in table.

Sample Solution (Exam VT2015):
number of seats repeated for each occurrence of aircraft
departure city repeated for each departure airport code
destination city repeated for each destination airport code
airline, cities, and aircraft of prime flight repeated for each flight code
airline is redundant if the flight code is given, because it can be computed from the two-letter prefix of the code
(The last one is a bit arguable, but was mentioned in the Sample Solution for the exam. You can argue if it is really a redundancy according to the definition)

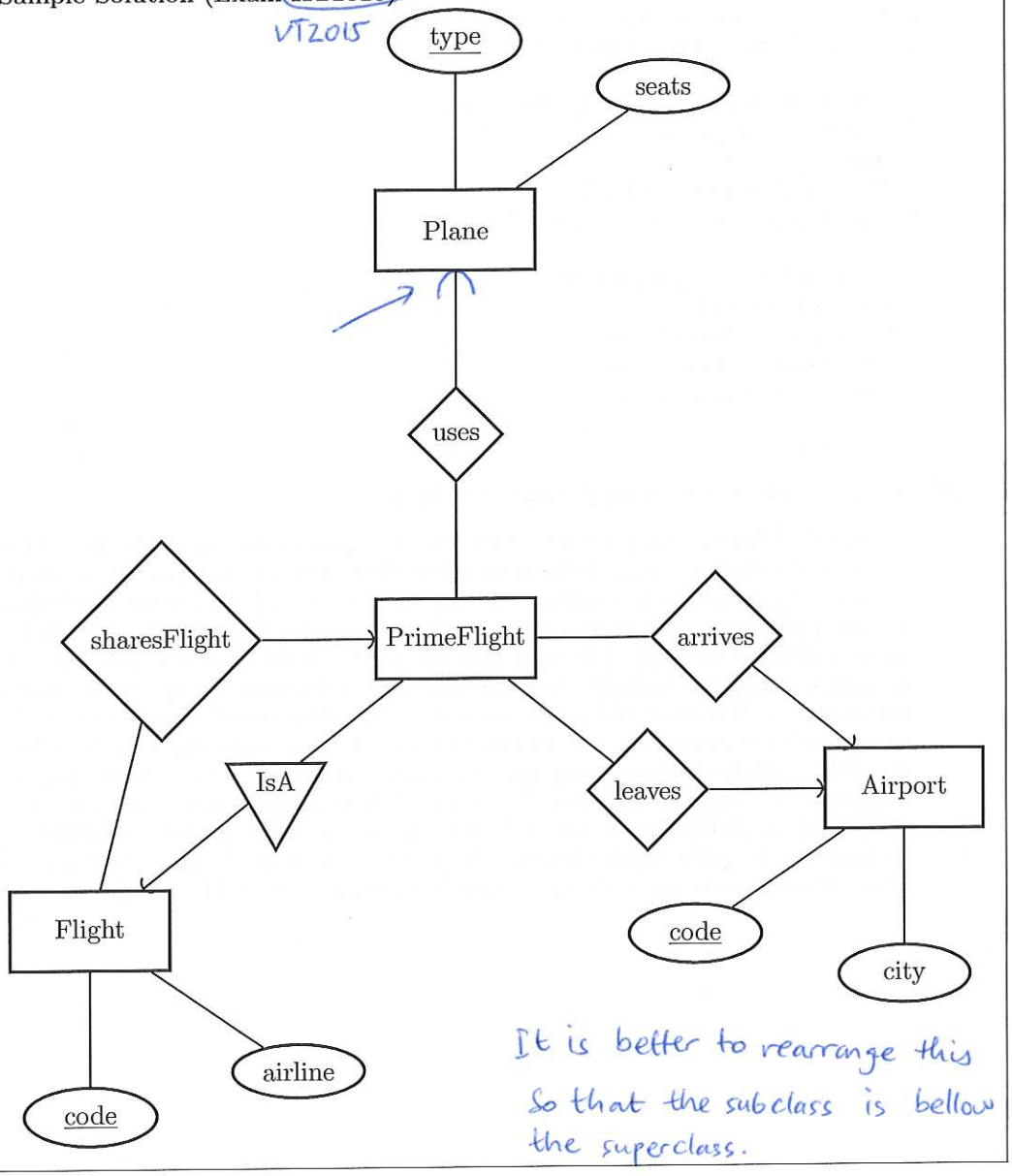
(b) (5 points) Draw an Entity-Relationship diagram that models the data in Table 1 in a meaningful way. The diagram must have some separate entities and Relationships. Mark the keys by underlining them.

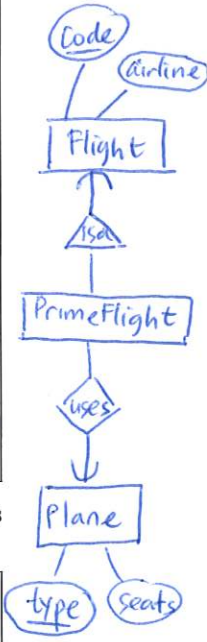
Solution:



Explanation:
The IsA relation only works if you model it in the right direction so that flightCode is the common key for Flight and PrimeFlight. In the presented model PrimeFlight is called Flight and Flight is called SubFlight. So the relation between SubFlight and Flight could be replaced by an isA relation in the direction of SubFlight (Flight is a SubFlight), so that the naming

seems rather unfortunate. Otherwise the solution is pretty close to the Sample Solution below.
Sample Solution (Exam HT2013):





(c) (4 points) Convert your Entity-Relationship (E-R) diagram to a database schema. Mark all keys and references.

Solution:
Plane(type,seats)
Airport(code,city)
Flight(flightCode,operatingAirline,fromAirport,toAirport,plane) fromAirport → Airport.code
toAirport → Airport.code

plane → Plane.type

SubFlight(flightCode,airline,primaryFlight)
primaryFlight → Flight.flightCode

Sample Solution (Exam VT2015):
Airport(code,city)
Aircraft(type,seats)
Flight(code,airline,primeFlight)
primeFlight → PrimeFlight.code

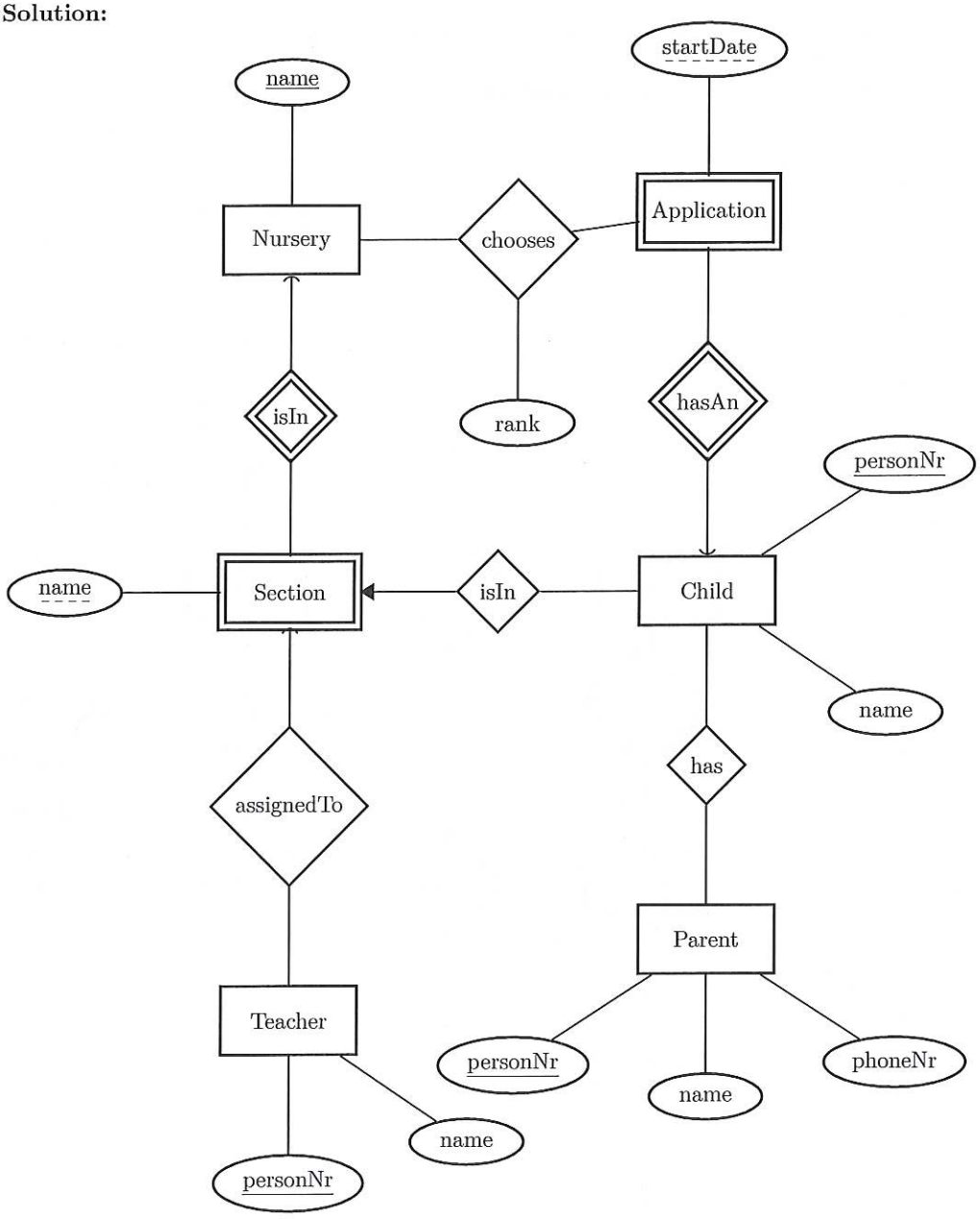
PrimeFlight(code,depAirport,destAirport,aircraft)
code → Flight.code
depAirport → Airport.code
destAirport → Airport.code
aircraf → Aicraft.type

2. (12 points) Consider the following domain description.

A local authority manages several nurseries which provide daycare for children. They want to use the database to record information about their nurseries. Each nursery is identified by it s name. Each nursery is organized into several sections, each with about 15 children. The sections within each nursery have unique names, but there might be sections with the same name in different nurseries. The local authority employs several teachers and each teacher is assigned to one of the sections. Each teachers name and person number should be stored in the database. The name and person number of each child should also be stored. Initially, an application is made for a nursery place for a child. The application contains information about the child, the childs starting date at nursery, and a list of nursery choices (e.g. choice 1 is nursery "A", choice 2 is nursery "B", etc.). After an application is processed, the child is allocated an available place in one of the sections of one of the nurseries. Information about the application and the childs placement should be stored in the database. For each child, the person number, name and telephone number of each parent should be stored in the database.

(a) (6 points) Draw an E-R diagram that correctly models this domain.

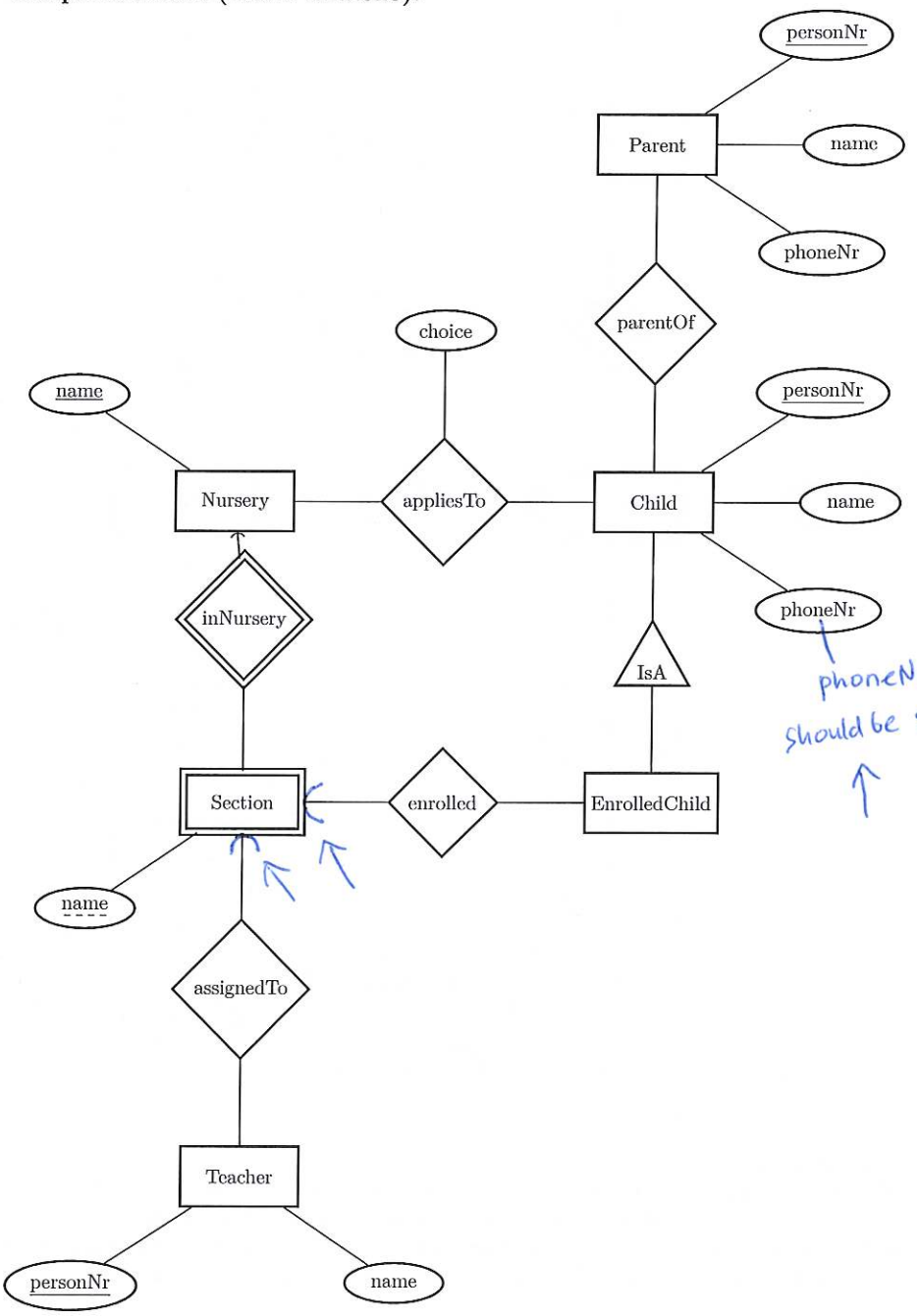
Solution:



Explanation:
The left side with Nursery, Section and Teacher as well as Parent and Child on the right are quite the same. The differences are in the treatment of the application, once as a weak entity and once as a relationship. The interpretation as a entity is rather not obvious interpretation, but it still works. Especially since according to the specification the start date should be part of the application.
On the other hand, having an at-most-one relation between Child and Section seems more

natural than having an extra entity EnrolledChild that has a referential integrity constraint to Section.
Another possible way would have been to introduce a entity Person and model Parent, Teacher and Child as subclasses.

Sample Solution (Exam HT2013):



phoneNr for "Child" is not important. We can derive this from the "Parent". startDate is important!.

(b) (6 points) Translate this E-R diagram into a set of relations, clearly marking all references and keys. If any attributes can contain null values, state which ones.

Solution: Nursery(<u>name</u>)
Section(<u>name</u>,<u>nursery</u>)
nursery → Nursery.name

Teacher(<u>personNr</u>, name, section, nursery)
(section,nursery) → Section.(name,nursery)

Application(<u>child</u>,<u>startingDate</u>)
child → Child.personNr

Chooses(<u>child</u>,<u>startDate</u>,<u>nursery</u>,rank)
child → Child.personNr
nursery → Nursery.name

Two options for Child:
ER approach:
Child(<u>personNr</u>,name)
ChildIsIn(<u>child</u>,section,nursery)
child → Child.personNr
(section,nursery) → Section.(name,nursery)

Nullable approach:
Child(<u>personNr</u>,name,section or NULL, nursery or NULL)
(section,nursery) → Section.(name,nursery)

Sample Solution (Exam HT2013):
Nursery(<u>name</u>)
Section(<u>name</u>,<u>nursery</u>) nursery → Nursery.name

Missing Child! →

Teacher(<u>personNr</u>,name,section,nursery)
(section,nursery) → Section.(name,nursery)
Child(<u>personNr</u>, name, startDate)
Parent(<u>personNr</u>,name,phoneNr)
EnrolledChild(<u>personNr</u>,nursery,section)
personNr → Child.personNr
(section,nursery) → Section(name,nursery)

AppliesTo(<u>child</u>,<u>nursery</u>,choice)
child → Child.personNr
nursery → Nursery.name

ParentOf(<u>parent</u>,<u>child</u>)
parent → Parent.personNr
child → Child.personNr

Note:
"Naming" in solution of exam HT 2013 is using plural for relation.

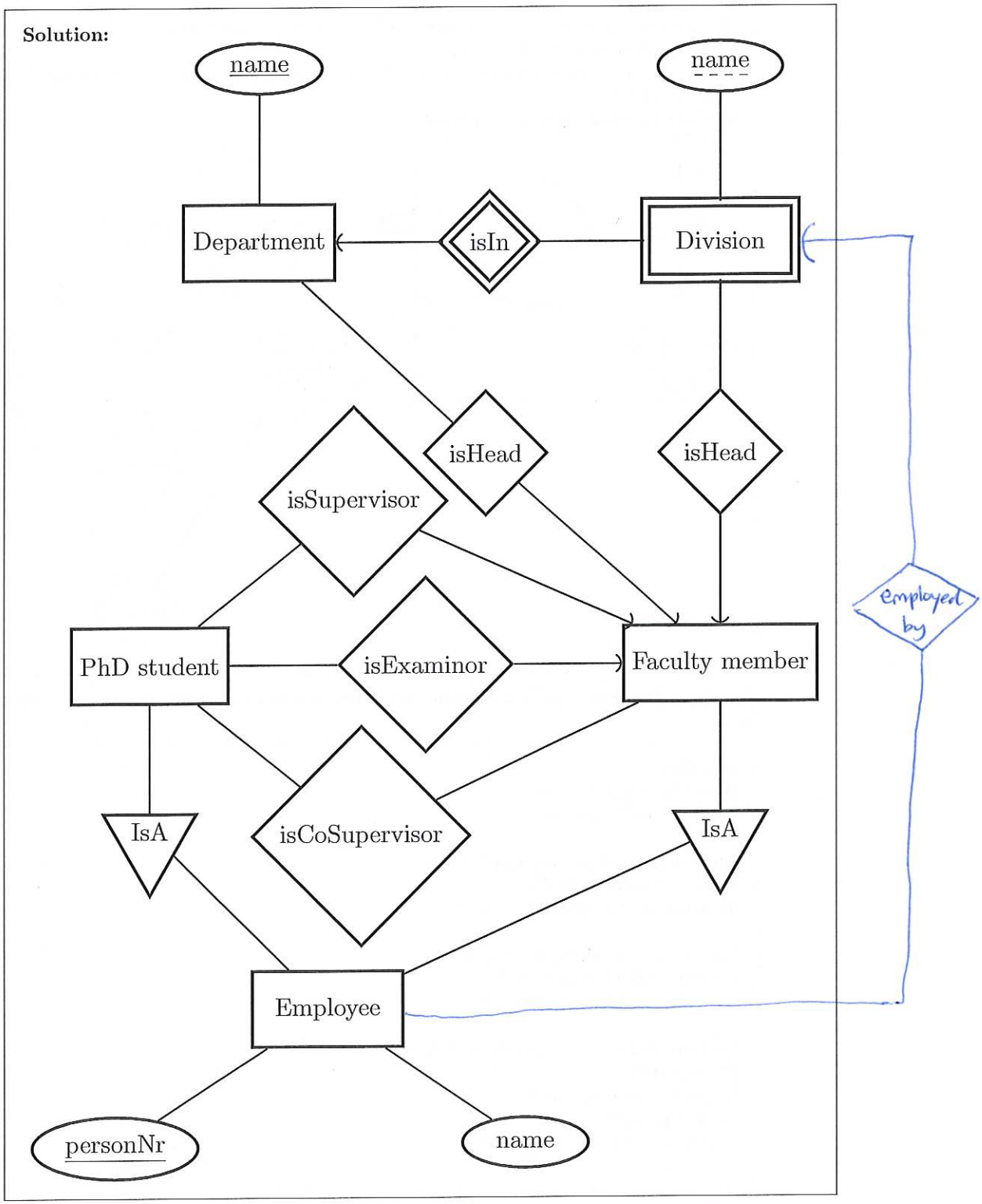
The solution shown here is not exactly as written in the exam solution.

3. (12 points) Consider the following domain description.

A university wants to use a database to store information about its departments, divisions and employees. Each department at the university has a unique name. Each department contains several divisions. Divisions in different departments can have the same name, but the division names within each department are unique. There can be many employees in each division, but each employee is employed at only one division. For each employee, their name and their unique personNumber should be stored. There are two kinds of employee at the university: faculty members and PhD students. For each PhD student, one faculty member is appointed to be their examiner. Each PhD student also has one main supervisor, but they can have zero or more co-supervisors. One faculty member at each department is appointed to be the head of that department. Similarly, one faculty member at each division is appointed to be the head of that division.

(a) (6 points) Draw an E-R diagram that correctly models this domain.

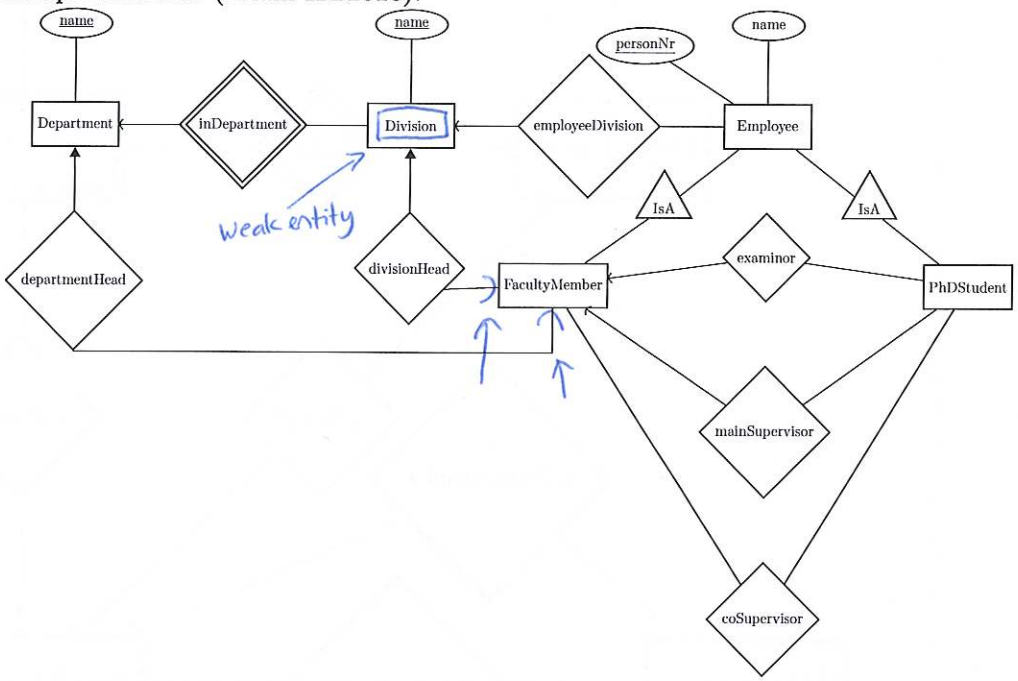
Solution:



This ER doesn't cover "each employee is employed at only one division".

Explanation:
The presented solution is almost identical to the Sample Solution below. The only difference is that the solution below allows only that a faculty member can be head of at most one department/division. This restriction was omitted in the presented solution even though it might be useful to add it.

Sample Solution (Exam HT2010):



(b) (6 points) Translate this E-R diagram into a set of relations, clearly marking all references and keys.

Solution:
Department(<u>name</u>,head)
head → Faculty.personNr

Division(<u>name</u>,<u>department</u>,head)
head → Faculty.personNr
department → Department.name

Cosupervisor(<u>student</u>,<u>supervisor</u>)
student → Student.personNr
supervisor → Faculty.personNr

Different options to translate the IsA:
ER approach:
Employee(<u>personNr</u>,name)
Faculty(<u>personNr</u>)
personNr → Employee.personNr

Phd(personNr,supervisor,examinor)
personNr → Employee.personNr
supervisor → Faculty.personNr
examinor → Faculty.personNr

OO approach:
Employee(personNr,name)
Faculty(personNr,name)
Phd(personNr,name,supervisor,examinor)
supervisor → Faculty.personNr
examinor → Faculty.personNr
Gives you redundancy in the duplicated attributes.
Nullable:
Employee(personNr,name,supervisor or NULL, examinor or NULL)
supervisor → Employee.personNr
examinor → Employee.personNr
In this case it would mean to remove both Faculty and PhD and instead have all the relations between employees. But that would allow us to select a PhD student as a supervisor or head of department. And that we want to avoid.

Sample Solution (Exam HT2010):
Departments(name, head)
head → FacultyMembers.personNumber
Divisions(dept, name, head)
dept → Departments.name
head → FacultyMembers.personNumber
Employees(personNumber, name, dept, division)
(dept, division) → Divisions.(dept, name)
FacultyMembers(personNumber)
personNumber → Employees.personNumber
PhDStudents(personNumber, examiner, mainSupervisor)
personNumber → Employees.personNumber
examiner → FacultyM embers.personNumber
mainSupervisor → FacultyMembers.personNumber
CoSupervisors(student, supervisor)
student → PhDStudents.personNumber
supervisor → FacultyMembers.personNumber