2DV513
Database Theory
Assignment 1

Jonas Sjöberg
860224-xxxx
Linnaeus University
js224eh@student.lnu.se
https://github.com/jonasjberg
http://www.jonasjberg.com

Written during: December 2, 2017 – December 3, 2017
Teacher responsible: Maria Ulan
Lars Karlsson

Abstract

The first assignment in the course *2DV513 – Database Theory* at Linnaeus University during the autumn term of 2017.

Contents

1 Task 1 — MoviesDB

Exercise from Fundamentals of Database Systems[1].

I have used the 7th edition which differs slightly from the 5th edition. The relevant diagram is included in Figure 1 for reference.

1.1 Problem Description

The problem description [2] is stated as follows:

> Assume that MoviesDB is a populated database.
>
> Given the constraints shown in the E/R diagram, respond to the following statements with True, False, or Maybe.
>
> Assign a response of Maybe to statements that, although not explicitly shown to be True, cannot be proven False based on the schema as shown.

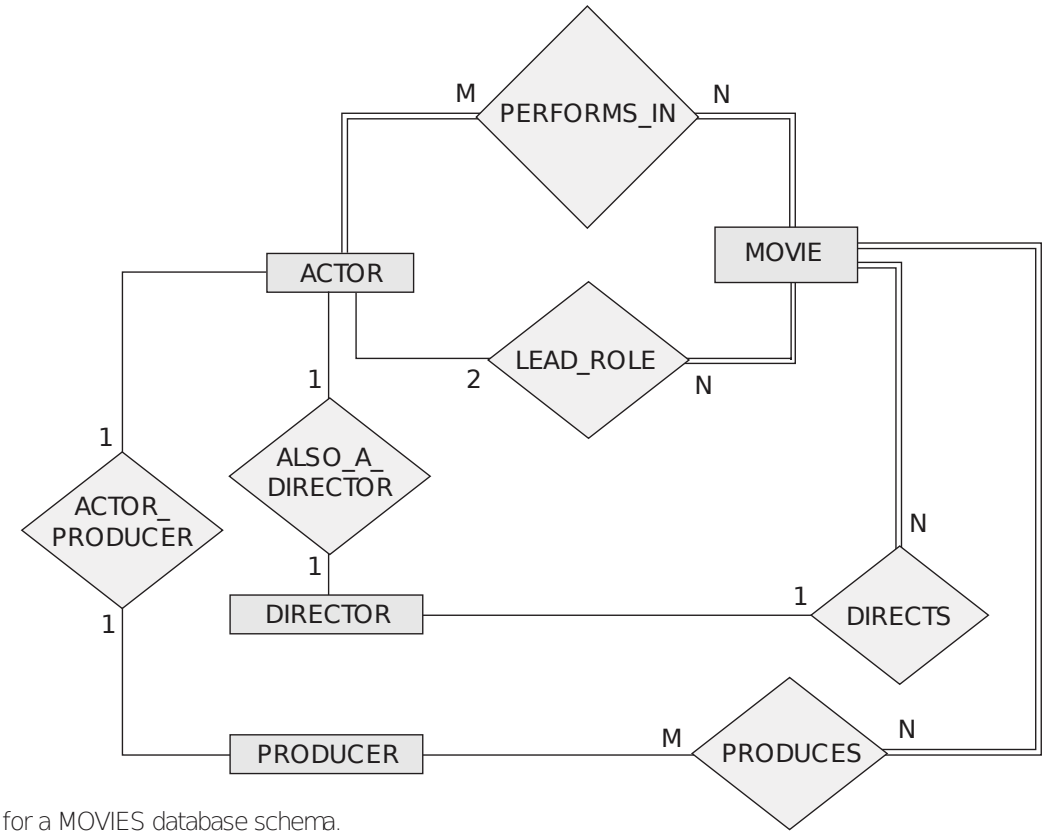

Figure 3.25
An ER diagram for a MOVIES database schema.

Figure 1: Diagram from the 7th edition of Fundamentals of Database Systems[1]

Justify each answer.

1. There are no actors in this database that have been in no movies.
2. There are some actors who have acted in more than ten movies.
3. Some actors have done a lead role in multiple movies.
4. A movie can have only a maximum of two lead actors.
5. Every director has been an actor in some movie.
6. No producer has ever been an actor.
7. A producer cannot be an actor in some other movie.
8. There are movies with more than a dozen actors.
9. Some producers have been a director as well.
10. Most movies have one director and one producer.
11. Some movies have one director but several producers.
12. There are some actors who have done a lead role, directed a movie, and produced a movie.
13. No movie has a director who also acted in that movie.

1.2 Solutions

1.2.1 1 — There are no actors in this database that have been in no movies

False. Double lines between `ACTOR` and `PERFORMS_IN` describe a total participation of `ACTOR` in the `PERFORMS_IN` relationship to the `MOVIE` entity. This essentially means that individuals that have been an actor is considered an actor.

1.2.2 2 — There are some actors who have acted in more than ten movies

Maybe. The relationship between `ACTOR` and `MOVIE` is many-to-many. The diagram does not include specific counts.

1.2.3 3 — Some actors have done a lead role in multiple movies

Maybe. The `LEAD_ROLE` relationship only specifies a maximum of `ACTOR` entities per `MOVIE`. There is no minimum requirement; a movie can have either none, one or two actors in lead roles. However, all movies must have at least one `LEAD_ROLE` relationship, because of the double lines — total participation.

1.2.4 4 — A movie can have only a maximum of two lead actors

True. See above and Figure 2 from [3].

1.2.5 5 — Every director has been an actor in some movie

False. The relationship `ALSO_A_DIRECTOR` is one-to-one, meaning that an `ACTOR` entity that is also a director must be related to one DIRECTOR entity. An `ACTOR` must per definition have a `ALSO_A_DIRECTOR` relationship to `DIRECTOR` in order to actually be a director, as far as this system is concerned.

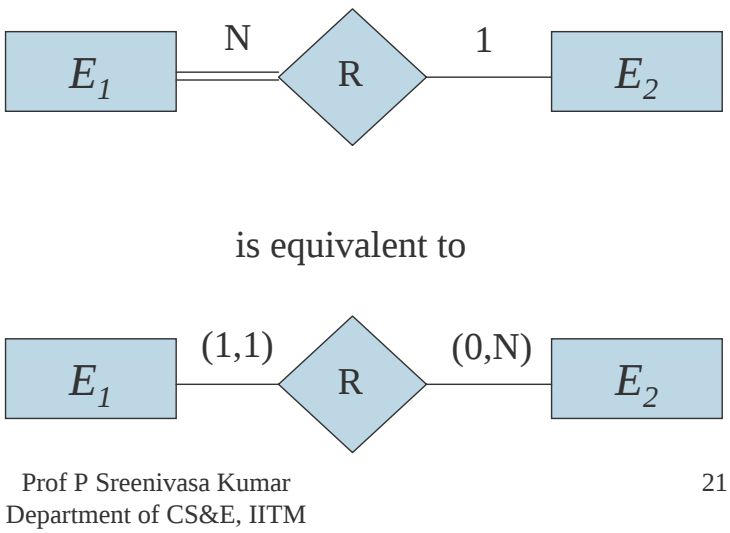

Figure 2: Diagram comparing the different notations [3]

1.2.6 6 — No producer has ever been an actor

Maybe. This might be true, but the diagram only specifies that *if* an `ACTOR` was a `PRODUCER`, it would be a one-to-one relationship through `ACTOR_PRODUCER`.

See the previous motivation.

1.2.7 7 — A producer cannot be an actor in some other movie

False. There are no such restrictions.

1.2.8 8 — There are movies with more than a dozen actors

Maybe. This might be true, but the diagram does not include any such restriction or requirement.

1.2.9 9 — Some producers have been a director as well

Maybe. This might be true. It is *allowed*, but we would have to query the actual database in order to get this information.

1.2.10 10 — Most movies have one director and one producer

Maybe. This might be true. It is *allowed*, but we would have to query the actual database in order to get this information.

1.2.11 11 — Some movies have one director but several producers

Maybe. This might be true. It is *allowed*, but we would have to query the actual database in order to get this information.

1.2.12 12 — There are some actors who have done a lead role, directed a movie, and produced a movie

Maybe. This might be true. It is *allowed*, but we would have to query the actual database in order to get this information.

1.2.13 13 — No movie has a director who also acted in that movie

Maybe. This might be true. It is *allowed*, but we would have to query the actual database in order to get this information.

2 Task 2 — Births

Exercise from *Database Systems – The Complete Book* [4], sections 2.2.6 and 2.2.7.

2.1 Problem Description

The problem description [2] is stated as follows:

> Consider a model where an entity set Births is related to Babies, Mothers, Doctors, and Nurses by four binary relationships.
>
> How can you use multiplicity to represent the following conditions?
>
> 1. Every baby is the result of a unique birth, and every birth is of a unique baby.
> 2. In addition to (1), every baby has a unique mother.
> 3. In addition to (1) and (2), for every birth there is a unique doctor.
>
> In each case, what design flaws do you see?
>
> Suppose we change our viewpoint to allow a birth to involve more than one baby born to one mother. How would you represent the fact that every baby still has a unique mother?

2.2 Solution

See Figure 3.

All possible flaws are related to the restrictions in multiplicities between babies and births, as well as between babies and mothers.

There are many possible edge-cases to consider, a mother might give birth to many babies during one birth; twins, etc. A mother might also not give birth to a living child. But the birth would nevertheless have employed a doctor, nurses, etc. as to require storage in a database for keeping track of hours worked, etc. However, if the primary concern of the system is to keep track of babies, this case might be considered "illegal". But otherwise, the case of a *NULL* baby is not handled by this design, the birth explicitly requires one baby but it should probable allow for any number, or at least one or more.

3 Task 3 — The Registrars Office

Exercise from *Database System Concepts, 5th Edition* [5], section 6.2.

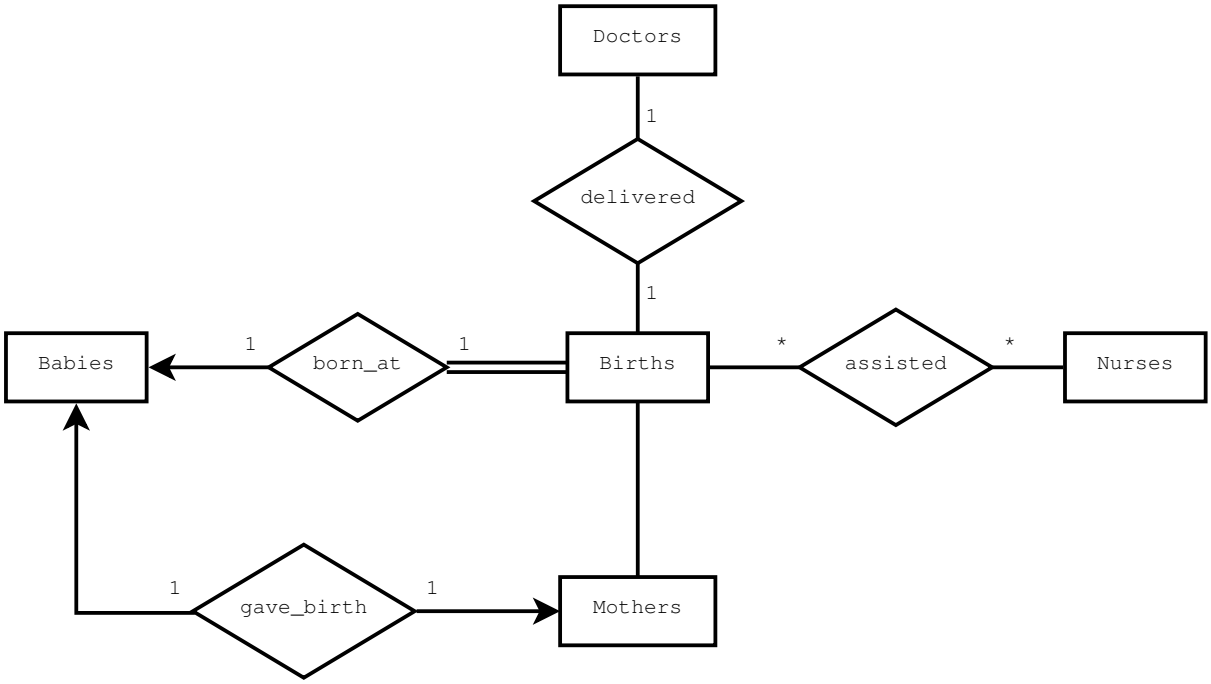

Figure 3: Entity-relationship Diagram for Task 2

3.1 Problem Description

The problem description [2] is stated as follows:

> A university registrar's office maintains data about the following entities:
>
> 1. (a) courses, include number, title, credits, syllabus, and prerequisites;
> 2. (b) course offerings, include course number, year, semester, section number, instructor(s), timings, and classroom;
> 3. (c) students, including student-id, name, and program; and
> 4. (d) instructors, including identification number, name, department, and title.
>
> Further, the enrollment of students in courses and grades awarded to students in each course they are enrolled for must be appropriately modeled.
>
> Construct an E/R diagram for the registrar's office.

3.2 Solution

The proposed solution is shown in Figure 5.

Some of the attributes have been bundled into new entities.

4 Task 4 — Classroom Scheduling

Exercise from *Database System Concepts, 5th Edition* [5], section 6.6.

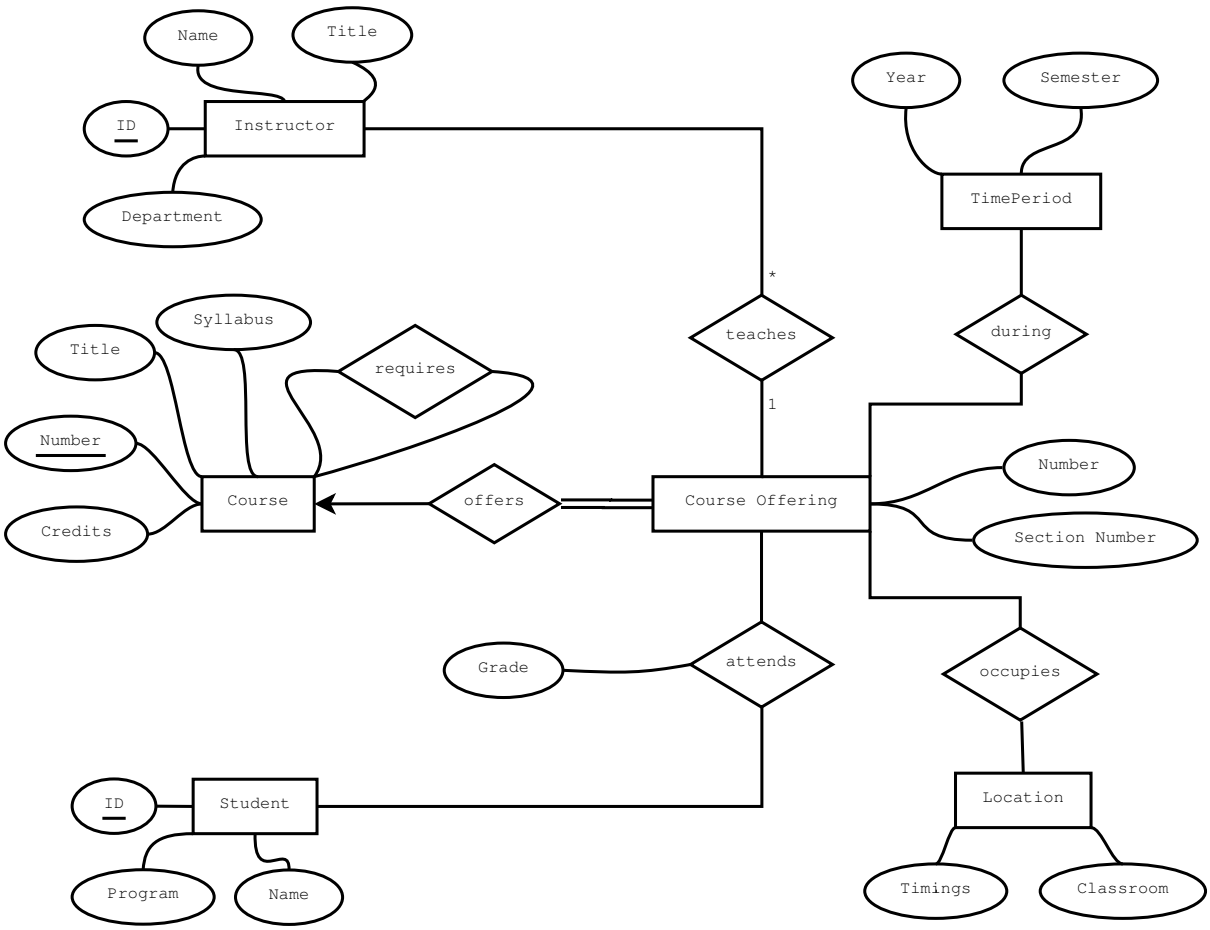

Figure 4: Entity-relationship Diagram for Task 3

4.1 Problem Description

The problem description [2] is stated as follows:

> Consider a university database for the scheduling of classrooms for the final exams.
>
> This database could be modeled as a single entity set *exam* with attributes *course_name*, *section_number*, *room_number*, and *time*.
>
> Alternatively, one or more additional entity sets could be defined, along with relationship sets to replace some of the attributes of the *exam* entity set, as:
>
> 1. *course* with attributes *name*, *department*, and *c_number*.
> 2. *section* with attributes *s_number* and *enrollment*, and dependent as a weak entity set on *course*.
> 3. *room* with attributes r_number, capacity, and building.
>
> Show an E/R diagram illustrating the use of all three additional entity sets listed.
>
> Explain what application characteristics would influence a decision to include or not include each of the additional entity sets.
>
> *Note:* A section is a part of course. How sections are used varies from university to university, but they could for example be used to separate multiple versions of the course (imaging that a course has so many students that there has to be parallel lectures) or if a course is given multiple times per year.

4.2 Solution

The proposed solution is shown in Figure **??**.

Some of the original attributes have been bundled into new entities. The original attributes that contained a underline and a shared prefix was extracted into a new entity. The common prefix became the entity name. This change seemed very similar to refactoring techniques I frequently use when writing software.

The new entities were then linked using the most basic relations, with names borrowed from the previous task.

Broadly, applications where maintainability and adjusting the change are likely; I.E. the schema will have to be reworked or the software application somehow needs to query the data in some new, initially unforeseen way.

This seems analogous to structuring software; smaller, modules with well defined responsibilities are easier to maintain and reason about than a bigger module that handles many different tasks.

It is often desirable to modularize systems and integrate these modules using well defined boundaries, which makes replacing modules much easier. In this case, the scheduling database really should not be all too tangled up with course-specific data — a course might use a different method for examining the students. Some courses might not have any exams at all, while some might use a online system to do the exams.

The entities are related through the unique identifier attributes but they each handle a distinctively different kind of data, that is open for modifications without disrupting the other tables.

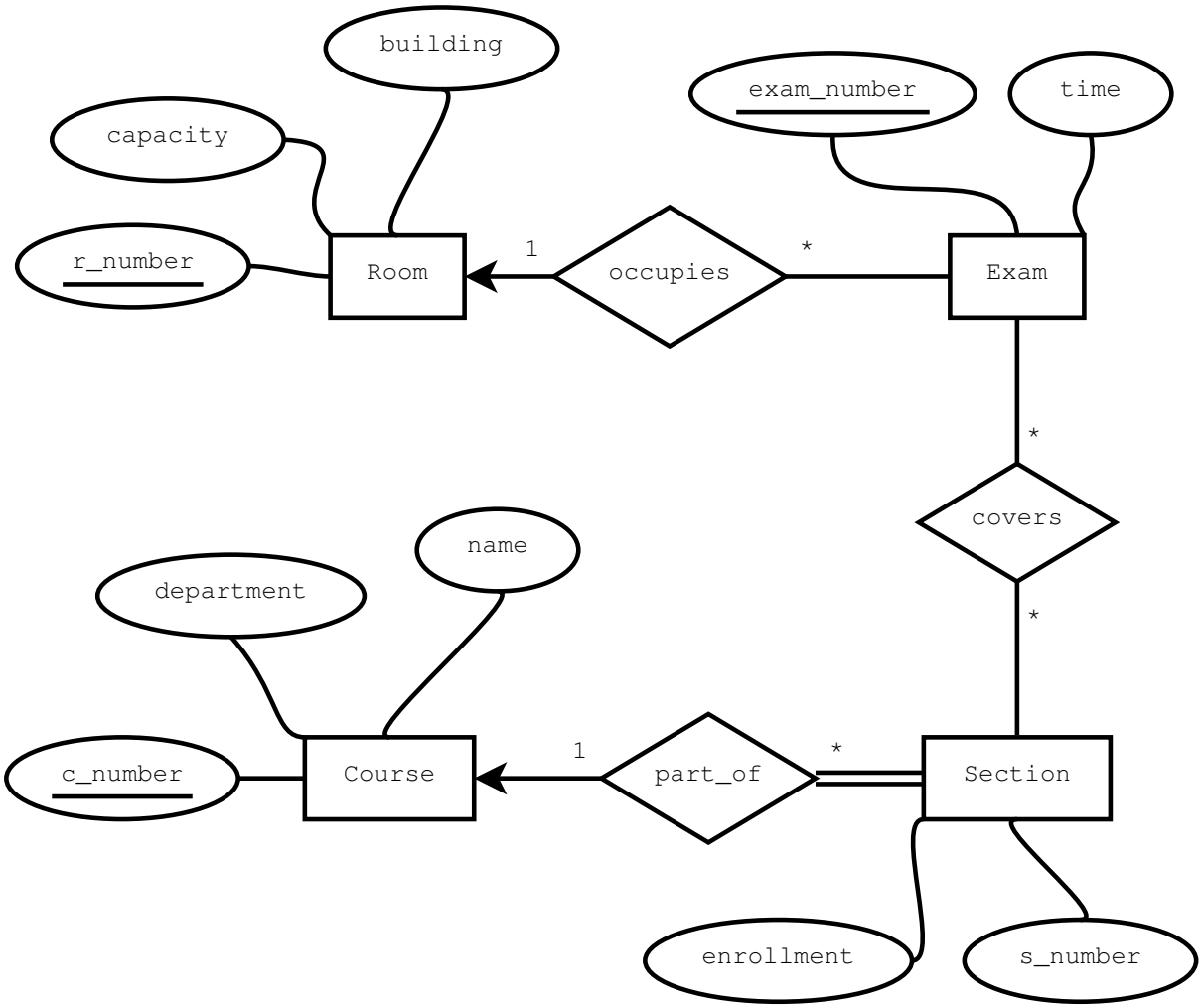

Figure 5: Entity-relationship Diagram for Task 4

References

[1] R. Elmasri and S. B. Navathe, *Fundamentals of Database Systems)*, 7th ed. Pearson, 2016, ISBN: 978-0-13-397077-7.

[2] *Assignment 1*, [Online; accessed 2-Dec-2017], 2017. [Online]. Available: `https://mymoodle.lnu.se/pluginfile.php/2760523/mod_assign/introattachment/0/Assignment1.pdf?forcedownload=1`.

[3] D. P. S. Kumar, *Entity-relationship model*, Lecture slides, [Online; accessed 2-Dec-2017], 2015. [Online]. Available: `http://nptel.ac.in/courses/106106095/pdf/2_Entity_Relationship_Model.pdf`.

[4] H. Garcia-Molina, J. D. Ullman, and J. Widom, *Database Systems: The Complete Book*, 2nd ed. Prentice Hall Press, 2008, ISBN: 9780131873254.

[5] A. Silberschatz, H. Korth, and S. Sudarshan, *Database Systems Concepts*, 5th ed. New York, NY, USA: McGraw-Hill, Inc., 2006, ISBN: 0072958863, 9780072958867.