Tutorial: Entity-Relationship Model

Students at the National University of Ngendipura (NUN) buy books for their studies. They also lend and borrow books to and from other students. Your company, **Apasaja Private Limited**, is commissioned by NUN Students Association (NUNStA) to implement an online book exchange system that records information about students, books that they own and books that they lend and borrow.

The database records the name, faculty, and department of each student. Each student is identified in the system by his/her email. The database also records the date at which the student joined the university. If a student has graduated, the database record the date of graduation. A department in NUN must belong to exactly one faculty.

The database records the title, authors, publisher, language, year as well as the ISBN-10 and ISBN-13 for each book. A book can have several authors but it must have at least one author. The database also records author that currently has no book. It should also record the format of the book (i.e., if the book is hardcover or softcover). The International Standard Book Number, ISBN-10 or -13, is an industry standard for the unique identification of books. It is possible that the database records books that are not owned by any students (e.g., because the owners of a copy graduated or because the book was advised by a lecturer for a course but not yet purchased by any student).

A student may own multiple copy of the same book. We differentiate the copy by its copy number. For instance, John may own two copies of the book Database Systems with ISBN-13 number of 9780131873254. The first copy has a copy number of 1 while the second copy has a copy number of 2. The copy number should be a consecutive number starting from 1.

The database also records the date at which a book copy is borrowed and the date at which it is returned. We refer to this information as a loan record. Obviously, a student can only borrow or lend book after he/she is enrolled.

For auditing purposes the database records information about the books, the copies and the owners of the copies as long as the owners are students or as there are loan records concerning the copies. For auditing purposes the database records information about graduated students as long as there are loan records concerning books that they owned.

Questions

Not all questions will be discussed during tutorial. You are expected to attempt them before coming to the tutorial. You are encouraged to discuss them on Canvas Discussion.

1. **Entity-Relationship Design**.

   (a) Identify the entity sets.

   > **Comments:**
   > We can draw the ERD in multiple attempts. After each attempt, we can refine our diagram. In a real-world, we typically stop at a reasonable design which may have `NULL` values.
   > As a general rule, the **nouns** generally becomes an entity set. So, on our first attempt, we can see that `students`, `books`, and `authors` are entity sets. However, we should also note that `departments` need to be an entity set because "A department in NUN must belong to exactly one faculty." This implies that department name can uniquely identify the name of the faculty.

   (b) Identify the relationship sets.

   > **Comments:**
   > In general, the **verbs** are relationship set. So, `own` is a relationship between `students` and `books`. `enrolled` is a relationship between `students` and `departments`. `write` is a relationship between `authors` and `books`.
   >
   > `loan` should also be a relationship. However, it is not between `students` and `books` as students are not borrowing a book but borrowing a *copy of a book* owned by another student. So, it is an association between `own` and `students`. The `students` in this case is the borrower as the owner is already recorded in `own`.
   > But we cannot form an association between relationship sets. As such, one of them has to be promoted to an aggregate. `own` is an aggregate.

   (c) For each entity set and relationship set, identify its attributes.

   > **Comments:**
   > The answer should be obvious for most of them. However, let us discuss a few important ones.
   > Firstly, note that graduation date can be `NULL`. So, we can promote this to an entity set. Thus, we have a new entity set (`graduation dates`) and relationship set (`for` between `students` and `graduation dates`).
   >
   > Secondly, return date can also be `NULL`. So, we do the same promotion into entity set `return date` and a relationship set `return` between `loan` (now an aggregate too) and `return date`.

   (d) For each entity set, identify its identifying attributes.

   > **Comments:**
   > The answer should be obvious for most of them. However, if we also look at the possible "keys" for relationship sets, then we can discover a few interesting constructs.
   > Firstly, note that `own` should also be identified by a copy number ("A student may own multiple copy of the same book."). But since `own` currently only associates

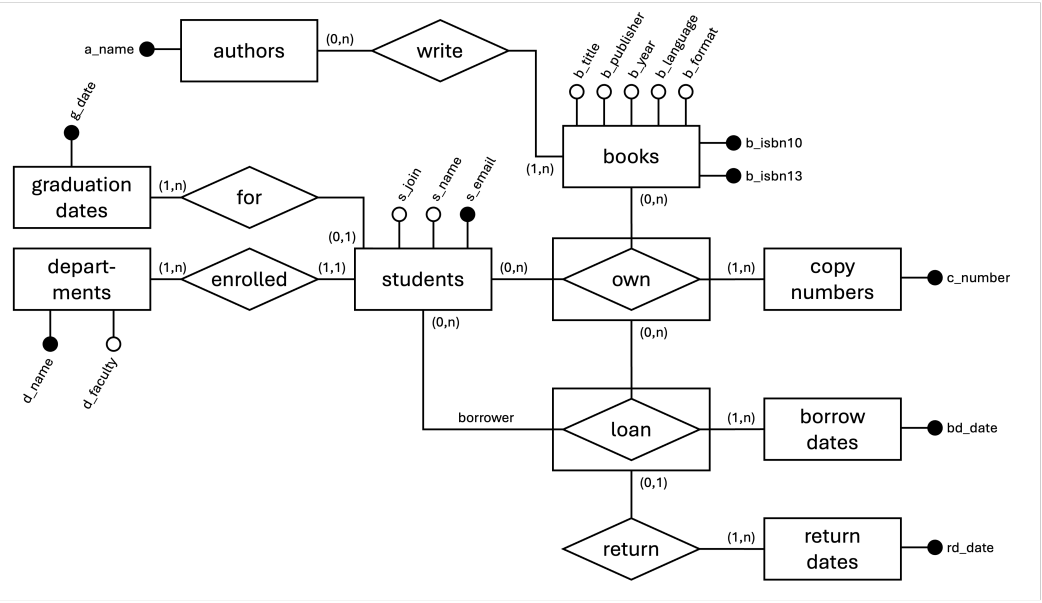
between `students` and `books`, it disallows a student from owning multiple books. Hence, we need another entity set called `copy numbers` to allow this.

Secondly, we have the same issue for `loan`. Currently it associates between `copy` and `students` (as borrower). But the same copy can be borrowed by the same student on different dates. So, we need another entity set called `borrow date`.

(e) Draw the corresponding entity-relationship diagram with the key and participation constraints. Indicate in English the constraints that cannot be captured, if any.

Comments:

One possible entity-relationship diagram is as follows.



There are some constraints not enforced. For instance, we cannot enforce the following. They can be enforced with triggers.

- The copy number should be a consecutive number starting from 1.
- Obviously, a student can only borrow or lend book after he/she is enrolled.

2. **Logical Design**.

(a) Translate your entity-relationship diagram into a relational schema. Give the SQL DDL statements to create the schema. Declare the necessary integrity constraints. Indicate in English the constraints that cannot be captured, if any.

Comments:

The translation mainly follows the 3 rules + 3 exceptions. However, there are some deviations as some constructs in the entity-relationship diagram cannot be easily translated.

- For the "graduation dates", we choose to merge the entity set with the `students` (which automatically merge this with the relationship set). Unfortunately, this means `g_date` can be `NULL`.

The alternative is to separate the entity sets. However, with this, the lower-bound 1 is not enforced. Additionally, we cannot easily check that `g_date` is greater than or equal to `s_join`.

There is a similar issue with "return dates".

- For "copy numbers", we also merge the entity set to the relationship set. This is the same issue of (1,n) participation discussed in lecture. Luckily, "copy numbers" has no other attributes. So merging it allows for all constraints to be enforced.

  There is a similar issue with "borrow dates" and with the same solution.

See the attached SQL code for the code.

References

[1] S. Bressan and B. Catania. *Introduction to Database Systems*. McGraw-Hill Education, 2006. ISBN: 9780071246507.

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

[3] Raghu Ramakrishnan and Johannes Gehrke. *Database Management Systems*. 2nd. USA: McGraw-Hill, Inc., 2000. ISBN: 0072440422.

[4] *W3schools Online Web Tutorials*. `https://www.w3schools.com/`. [Online; last accessed 2025].