**Objective:** Now that you have a conceptual model (ER diagram), map it into a robust logical relational schema using the algorithms described in the classroom. This schema will define the structure of your database tables, including attributes, data types, primary keys, and foreign keys, functional dependencies, ensuring data integrity and efficiency.

**Introduction [5 points].** ~~*Project Overview:* Briefly reiterate the purpose and main functionalities of your database, maintaining consistency with your previous project parts.~~ *Scope*~~: Re-emphasize the boundaries of your project, clarifying what data and functionalities are included and excluded.~~ *Glossary*: Update your glossary with any new terms or concepts relevant to relational database design. The introduction section is for continuity; the *italic* parts are subsections.

Project Overview: The purpose of this database is to store a collection of different books, magazines, movies able to be rented out by different users.

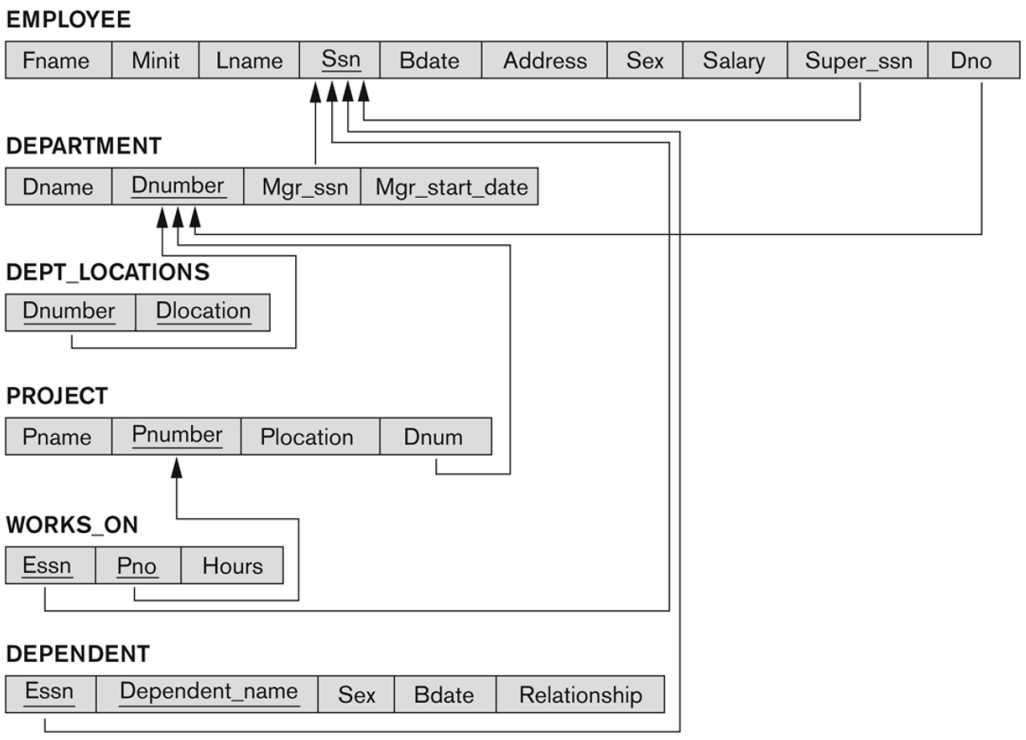
- **Scope:**
    - The purpose of a Library Management System is to make a useful, scalable, and educational tool that works like a real library. As part of the project, we will be able to work directly with all aspects of database development, from conceptual modeling to logical design to physical implementation. We will also add ways to make sure that borrowing rules are followed and useful reports will be made to help with operational decisions, collection growth, and member participation. At the end of the day, this project will be both a useful way to learn and show how well-designed database systems can help solve difficult information management issues.

- **Glossary:**
    - ISBN: International Standard Book Number; is a unique, 13-digit numerical identifier for a specific book edition and format, used by libraries, bookstores, and online retailers to catalog, track, and sell books globally.
    - SQL: Structured Query Language
    - Functional Requirement: specify what a system must do, detailing the specific actions, functions, and features it needs to perform to meet user and business needs
    - Non-Functional Requirement: specify what a system must do, detailing the specific actions, functions, and features it needs to perform to meet user and business needs

**Relational Schema Mapping [30]**

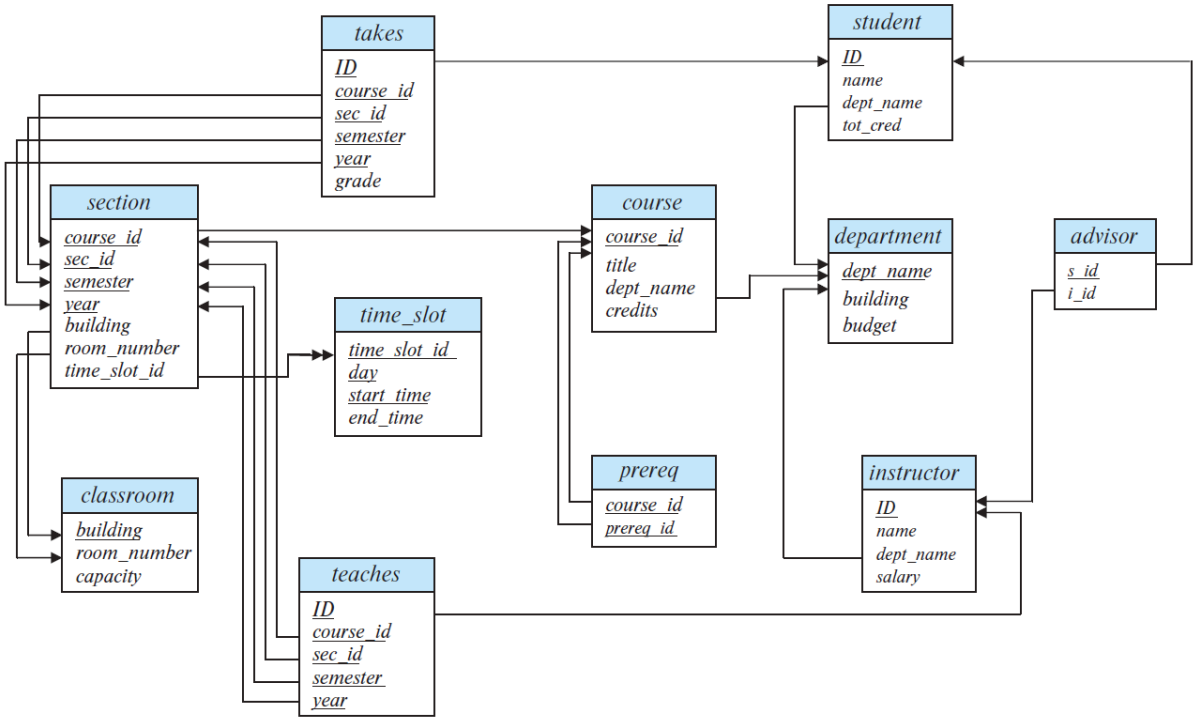
- **Identify relations:** For each entity and relationship sets in your ER diagram, create a corresponding relation (table).
- **Define attributes and domains:** For each attribute in your entity and relationship sets, define the corresponding attribute in the appropriate relation. Clearly document the domain of each attribute (i.e., the valid range of values). This can be done as a separate  within a data dictionary (see below).
- **Determine primary keys:** Identify the primary key for each relation. This key will uniquely identify each tuple (row) in the relation.
- **Establish foreign keys:** For each relationship in your ER diagram, implement foreign key constraints in the corresponding relations to maintain referential integrity. Ensure that foreign keys reference the primary key of the related relation.
- **Establish the functional dependencies (FDs):** The FDs represent the constraints from the real world and are useful for maintaining the integrity of the database.

[Schema Mapping Canva]



**Create a relational schema diagram [10 points]**. This diagram should clearly show all relations, attributes, primary keys, and foreign keys. Draw a line from a foreign key to its corresponding primary key. See the Company and University sample diagrams below (one from each of the recommended textbooks).

Schema Mapping Canva




**Schema Documentation with a Data Dictionary [10 points]:** Develop a data dictionary that documents each relation, its attributes, data types, domains, and any constraints. This dictionary serves as a central repository of information about your database schema. While data dictionaries can be very complex, a simple table created using Excel (or Word or your favorite editing tool), like the following contrived sample, will suffice.

Our Data Dictionary

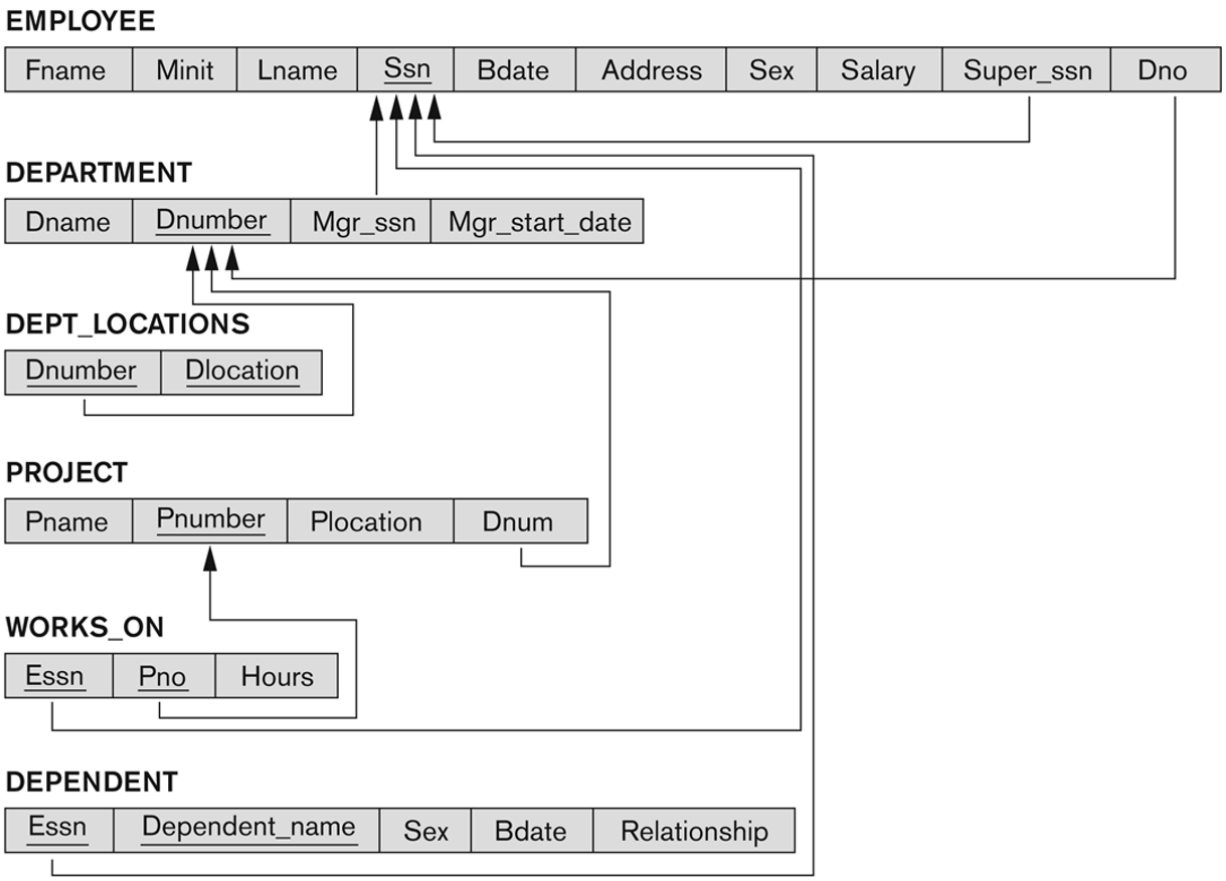
| Attribute Name | Data Type | Description |
|---|---|---|
| `Title` | `VARCHAR` | The title of the book |
| `Author` | `VARCHAR` | The author(s) of the book |
| `Publisher` | `VARCHAR` | The publisher of the book |
| `Genre` | `VARCHAR` | The genre of the book |
| `ISBN` | `CHAR(13)` | International Standard Book Number, unique to each edition and variation of a book |
| `No_Pages` | `INT` | Number of pages in the book |
| `Pub_Year` | `YEAR` | The year the book was published |
| `Language` | `VARCHAR` | The language in which the book is written |
| `Format` | `VARCHAR` | The format of the book (e.g., Hardcover, Paperback, eBook) |
| `Price` | `DECIMAL(5,2)` | The price of the book |

**Normalization considerations**. Keep in mind that you may need to further refine your relational schema by normalizing your relations to Boyce-Codd Normal Form (BCNF) or Third Normal Form (3NF) in the future. This process helps to minimize data redundancy and improve data integrity. If you've developed a well-thought-out ER diagram, the resulting relations may already conform to BCNF or 3NF.

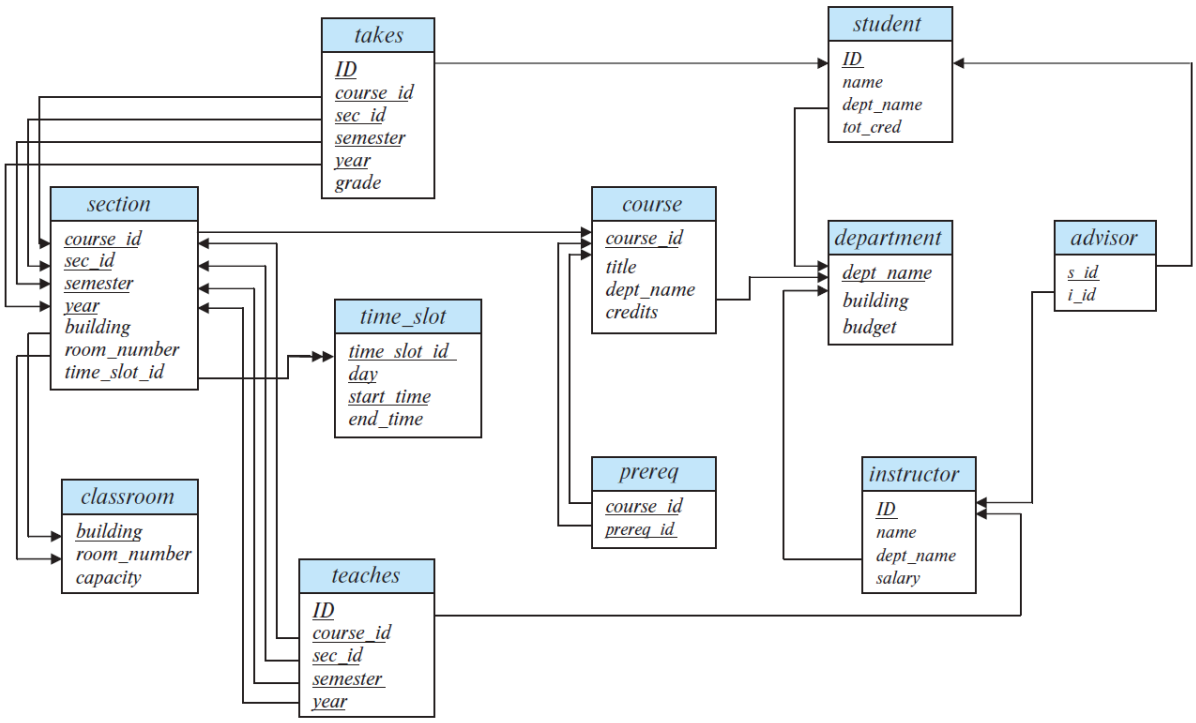
---

### Sample relation schema diagram

The Company relational database from Elmasri and Navathe:




The University relational database schema from Silberschatz, Korth, and Sudarsha:




**Appendices**. You are welcome to use appendices to provide additional information, e.g., your design choices, explain why you chose certain entities, how you determined the relationships, and any assumptions you made during the modeling process.

**GitHub Repository Management**. Continue to maintain all project artifacts in your GitHub repository. The team leader should then submit the repository URL for this project part on Canvas.

**https://github.com/JaidenTGreen/EECS-447-Project/tree/main**