Lab Assignment 2

Exercise 3.33: Consider a MOVIE database in which data is recorded about the movie industry. The data requirements are summarized as follows:

- Each movie is identified by title and year of release. Each movie has a length in minutes. Each has a production company, and each is classified under one or more genres (such as horror, action, drama, and so forth). Each movie has one or more directors, and one or more actors appear in it. Each movie also has a plot outline. Finally, each movie has zero or more quotable quotes, each of which is spoken by a particular actor appearing in the movie.
- Actors are identified by name and date of birth and appear in one or more movies. Each actor has a role in the movie.
- Directors are also identified by name and date of birth and direct one or more movies. It is possible for a director to act in a movie (including one that he or she may also direct).
- Production companies are identified by name and each has an address. A production company produces one or more movies.

Design a CROWS FOOT ERD for the movie database and enter the design using a data modeling tool such as Erwin or Rational Rose. Here's how you can represent the entities and their relationships in the movie database:

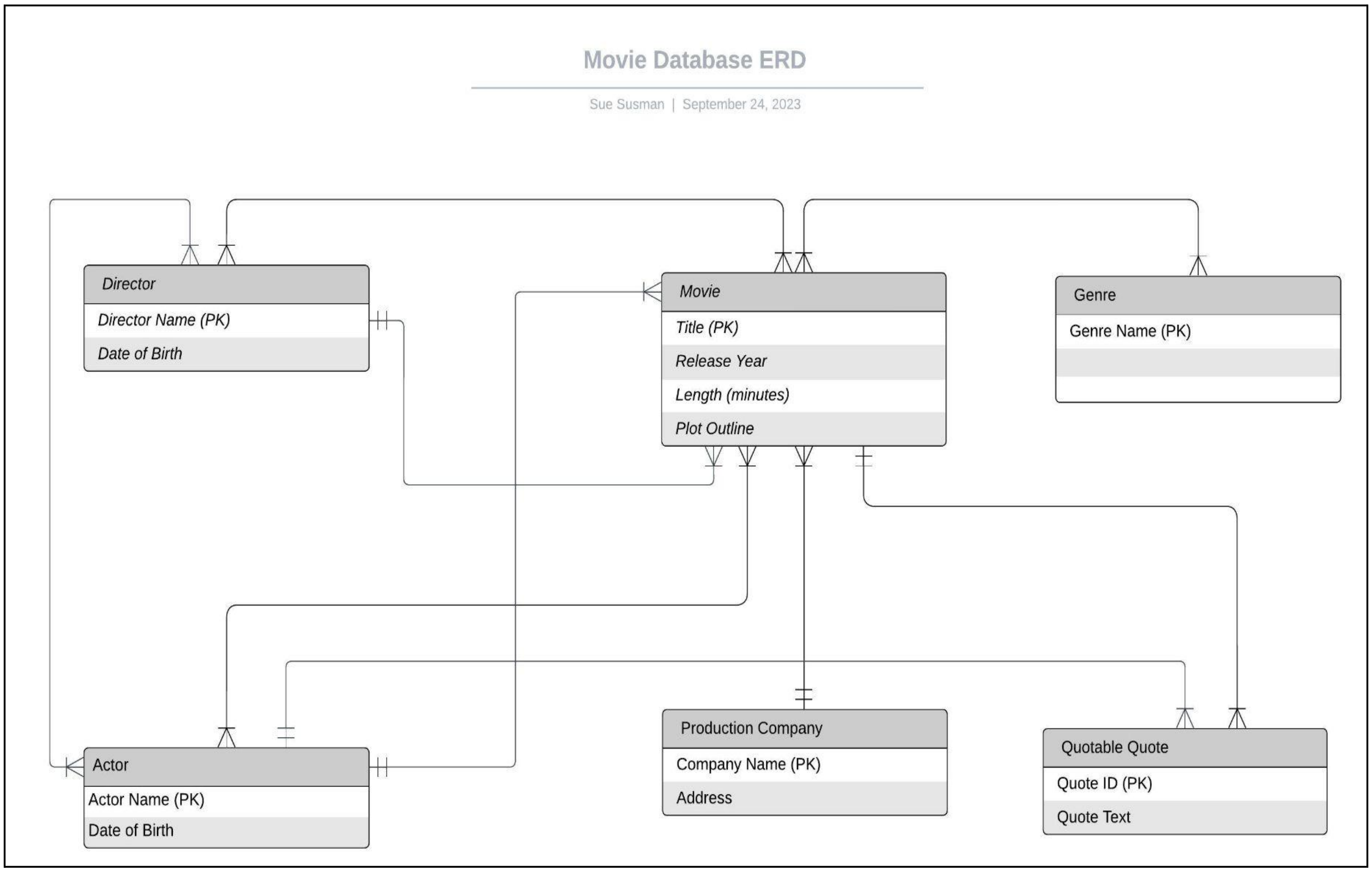
Entities:

1. Movie
    a. Attributes:
        i. Title (Primary Key)
        ii. Year of Release, Length (in minutes)
        iii. Plot Outline
    b. Relationships:
        i. Many-to-Many with Genre (A movie can belong to multiple genres)
        ii. Many-to-One with Production Company (A movie is produced by one production company)
        iii. Many-to-Many with Director (A movie can have multiple directors)
        iv. Many-to-Many with Actor (Multiple actors can appear in a movie)
        v. One-to-Many with Quotable Quotes (A movie can have multiple quotable quotes)
2. Genre
    a. Attributes:
        i. Genre Name (Primary Key)
3. Director
    a. Attributes:
        i. Name (Primary Key)
        ii. Date of Birth
    b. Relationships:
        i. One-to-Many with Movie (A director can direct multiple movies)
        ii. Many-to-Many with Movie (A director can act in multiple movies)
4. Actor
    a. Attributes:
        i. Name (Primary Key)
        ii. Date of Birth
    b. Relationships:
        i. One-to-Many with Movie (An actor can appear in multiple movies)
5. Production Company
    a. Attributes:
        i. Name (Primary Key)

            ii. Address
    b. Relationships:
    c. One-to-Many with Movie (A production company can produce multiple movies)
6. Quotable Quote
    a. Attributes:
        i. Quote ID (Primary Key)
        ii. Quote Text
    b. Relationships:
        i. Many-to-One with Movie (A quote is spoken in a particular movie)
        ii. Many-to-One with Actor (A quote is spoken by a particular actor)

Relationships (with cardinality):

1. Movie-Genre: Many-to-Many
2. Movie-Production Company: Many-to-One
3. Movie-Director: Many-to-Many
4. Movie-Actor: Many-to-Many
5. Movie-Quotable Quote: One-to-Many
6. Director-Movie: One-to-Many
7. Director-Actor: Many-to-Many
8. Actor-Movie: One-to-Many
9. Production Company-Movie: One-to-Many
10. Quotable Quote-Movie: Many-to-One
11. Quotable Quote-Actor: Many-to-One



Exercise 3.34: Consider a CONFERENCE_REVIEW database in which researchers submit their research papers for consideration. Reviews by reviewers are recorded for use in the paper selection process. The database system caters primarily to reviewers who record answers to evaluation questions for each paper they review and make recommendations regarding whether to accept or reject the paper. The data requirements are summarized as follows:

- Authors of papers are uniquely identified by e-mail id. First and last names are also recorded.
- Each paper is assigned a unique identifier by the system and is described by a title, abstract, and the name of the electronic file containing the paper.
- A paper may have multiple authors, but one of the authors is designated as the contact author.
- Reviewers of papers are uniquely identified by e-mail address. Each reviewer's first name, last name, phone number, affiliation, and topics of interest are also recorded.
- Each paper is assigned between two and four reviewers. A reviewer rates each paper assigned to him or her on a scale of 1 to 10 in four categories: technical merit, readability, originality, and relevance to the conference. Finally, each reviewer provides an overall recommendation regarding each paper.
- Each review contains two types of written comments: one to be seen by the review committee only and the other as feedback to the author(s).

Design a CROWS FOOT ERD for the CONFERENCE_REVIEW database and build the design using a data modeling tool such as Erwin or Rational Rose. Here's how you can represent the entities and their relationships:

Entities:

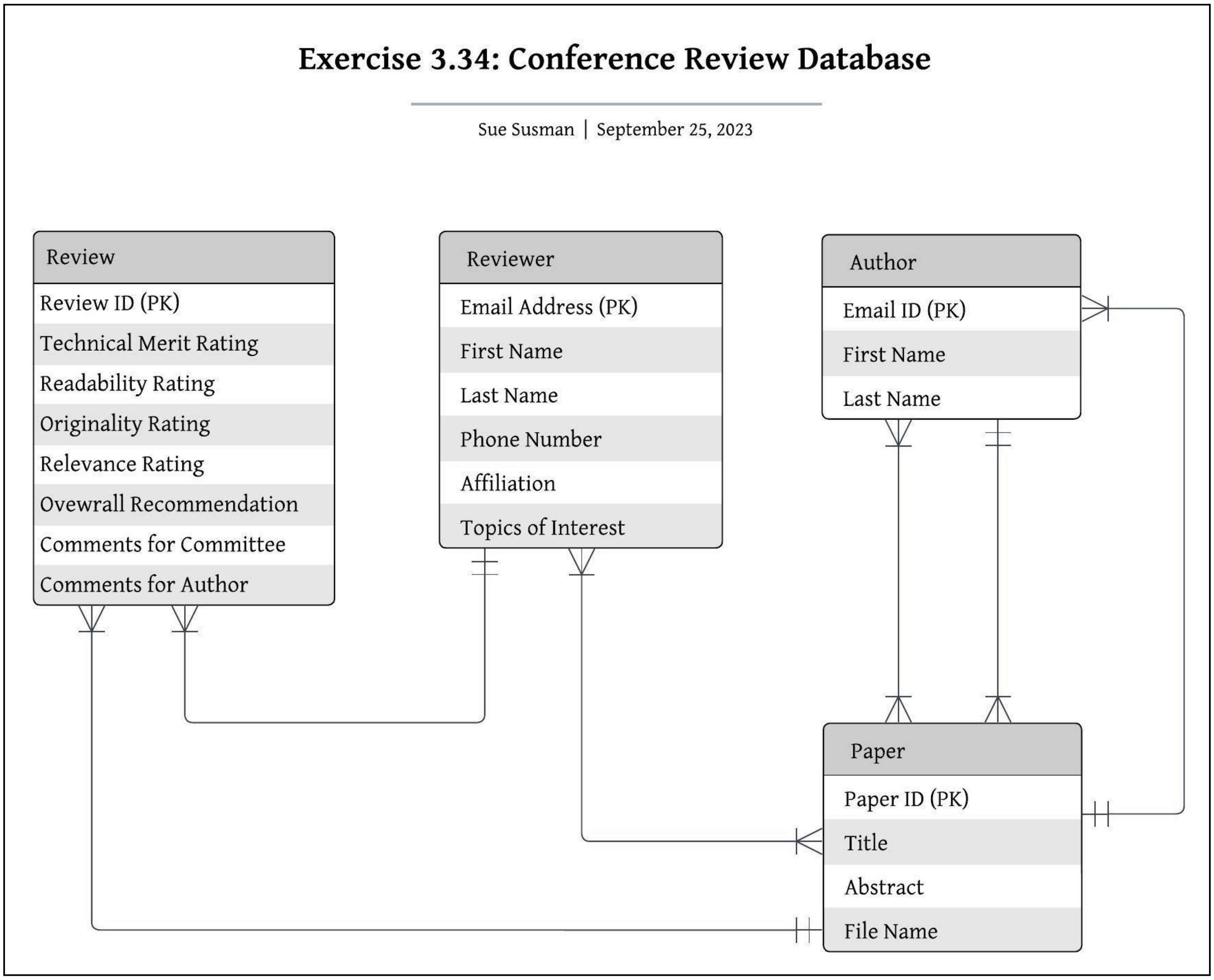
1. Author
    a. Attributes:
        i. Email ID (Primary Key)
        ii. First Name
        iii. Last Name
    b. Relationships:
        i. One-to-Many with Paper (An author can write multiple papers)
        ii. Many-to-One with Paper (One of the authors is the contact author of a paper)
2. Paper
    a. Attributes:
        i. Paper ID (Primary Key)
        ii. Title
        iii. Abstract
        iv. File Name
    b. Relationships:
        i. Many-to-Many with Author (Multiple authors can write a paper)
        ii. Many-to-Many with Reviewer (A paper is reviewed by multiple reviewers)
3. Reviewer
    a. Attributes:
        i. Email Address (Primary Key)
        ii. First Name
        iii. Last Name
        iv. Phone Number
        v. Affiliation
        vi. Topics of Interest
    b. Relationships:
        i. One-to-Many with Review (A reviewer can write multiple reviews)
        ii. Many-to-Many with Paper (A reviewer can review multiple papers)
4. Review

a. Attributes:
   i. Review ID (Primary Key)
   ii. Technical Merit Rating
   iii. Readability Rating
   iv. Originality Rating
   v. Relevance Rating
   vi. Overall Recommendation
   vii. Comments for Committee
   viii. Comments for Author(s)

b. Relationships:
   i. Many-to-One with Paper (A review is for a specific paper)
   ii. Many-to-One with Reviewer (A review is written by a specific reviewer)

Relationships (with cardinality):

1. Author-Paper: Many-to-Many
2. Paper-Reviewer: Many-to-Many
3. Review-Reviewer: Many-to-One
4. Review-Paper: Many-to-One



Exercise 5.17: Consider the following relations for a database that keeps track of booking of apartments by a constructor. (OPTION refers to some specific optional requirements/designs stated by the client to be implemented in the flat):

- APARTMENT (Apartment#, Model, Address, Price_perSquareFt)
- OPTION(Apartment#, Option_name, Extra_price)
- BOOKING(Agent_id, Apartment#, Date, Booking_price)
- AGENT(Agent_id, Name, Phone)

I. First, specify the foreign keys for this schema, stating any assumptions you make.
II. Next, populate the relations with a few sample tuples
III. Then give an example of an insertion in the BOOKING and AGENT relations that violates the referential integrity constraints and of another insertion that does not.

In the given database schema, we have four relations:

1. APARTMENT
2. OPTION
3. BOOKING
4. AGENT

I. First, specify the foreign keys for this schema, stating any assumptions you make.
   A. Foreign Keys:
      i. In the APARTMENT relation, the foreign key is Apartment# in the OPTION relation.
      ii. In the OPTION relation, the foreign key is Apartment# in the APARTMENT relation.
      iii. In the BOOKING relation, the foreign keys are Agent_id and Apartment# referencing AGENT and APARTMENT, respectively.

II. Now, let's populate the relations with sample tuples:

APARTMENT:

Apartment	Model	Address	Price/SqFt
A101	Model X	123 Main St.	150.00
A102	Model Y	456 Elm St.	200.00
A103	Model Z	789 Oak St.	180.00

OPTION:

Apartment	Option Name	Extra Price
A101	Balcony	50.00
A102	Parking	30.00
A103	Pool	70.00

AGENT:

Agent ID	Name	Phone
101	Agent A	555-123-4567
102	Agent B	555-987-6543
103	Agent C	555-456-7890

III. Now, let's provide an example of an insertion that violates the referential integrity constraint and another insertion that does not.

A. Violation of Referential Integrity (Invalid Insertion): Suppose we try to insert a booking in the BOOKING relation for an apartment that does not exist in the APARTMENT relation.

Agent ID	Apartment #	Date	Booking Price
101	A104	09/15/2023	2000.00

1. In this case, Apartment# "A104" does not exist in the APARTMENT relation, violating the referential integrity constraint.

B. No Violation of Referential Integrity (Valid Insertion): Now, let's insert a booking in the BOOKING relation for an existing apartment and an agent that exist in their respective relations:

Agent ID	Apartment #	Date	Booking Price
101	A101	09/15/2023	2500.00

1. In this case, Apartment# "A101" exists in the APARTMENT relation, and Agent_id "101" exists in the AGENT relation, so there is no violation of the referential integrity constraint.

Exercise 5.17: Consider a STUDENT relation in a UNIVERSITY database with the following attributes (Name, SSN, Local_phone, Address, Cell_phone, Age, Gpa). Note that the cell phone may be from a different city and state (or province) from the local phone. A possible tuple of the relation is shown below:

Name	SSN	LocalPhone	Address	CellPhone	Age	GPA
George Shaw William Edwards	123-45-6789	555-1234	123 Main St., Anytown, CA 94539	555-4321	19	3.75

1. **Identify the critical missing information from the LocalPhone and CellPhone attributes as shown in the example above. (Hint: How to call someone who lives in a different state or province?)**
    a. In the example, the critical missing information from the LocalPhone and CellPhone attributes is the area code or country code. When calling someone from a different state, province, or country, you need to specify the area code or country code to ensure the call is directed to the correct location. Without this information, it would be challenging to make long-distance or international calls.

2. **Would you store this additional information in the LocalPhone and CellPhone attributes or add new attributes to the schema for STUDENT?**
    a. It would be more appropriate to add new attributes to the schema for STUDENT to store the missing area code or country code information for both LocalPhone and CellPhone. This additional information can help ensure that phone numbers are complete and accurate.

3. **Consider the Name attribute. What are the advantages and disadvantages of splitting this field from one attribute into three attributes (first name, middle name, and last name)?**
    a. Advantages:
        i. Improved Data Quality: Splitting the name into first name, middle name, and last name allows for better data quality and consistency. It can help avoid issues with variations in how names are entered.
        ii. Enhanced Search and Sorting: Separating names into distinct fields makes it easier to search, sort, and filter data based on individual name components.
        iii. Flexibility: It allows for flexibility in accommodating individuals with different name structures (e.g., some may have middle names, while others may not).
    b. Disadvantages:
        i. Increased Complexity: Splitting the name into multiple attributes increases the complexity of the schema and may require additional validation to ensure data integrity.
        ii. Space Overhead: Adding extra attributes for first name, middle name, and last name can consume more storage space.

4. What general guideline would you recommend for deciding when to store information in a single attribute and when to split the information.
    a. The decision to store information in a single attribute or split it into multiple attributes depends on the specific requirements and goals of the database. Here are some general guidelines:
        i. Split information when it enhances data accuracy, searchability, and consistency.
        ii. Consider splitting data when different parts of the information have distinct meanings or need to be queried separately.
        iii. Keep data together in a single attribute when it's inherently linked and doesn't require separation for the database's intended use.
        iv. Balance data normalization with practicality; avoid excessive splitting for simplicity when it doesn't significantly benefit data management.

5. Suppose the student can have between 0 and 5 phones. Suggest two different designs that allow this type of information.
    a. Design 1: Create Separate Phone Number Attributes
        i. LocalPhone1, LocalPhone2, LocalPhone3, LocalPhone4, LocalPhone5
        ii. CellPhone1, CellPhone2, CellPhone3, CellPhone4, CellPhone5

This design allows for up to five local and five cell phone numbers per student, but it can result in a large number of attributes.

   b. Design 2: Create a Separate Phone Number Table for phone numbers linked to students:
        i. StudentPhone
        ii. StudentID (Foreign Key)
        iii. PhoneNumber
        iv. PhoneType (Local or Cell)

This design allows for a variable number of phone numbers per student without the need for numerous attributes.