Joseph Annand

DSE 6210 Assignment 2

Textbook Exercises 3.33, 3.34, 5.17, 5.19

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 entity-relationship diagram for the movie database and enter the design using a data modeling tool such as Erwin or Rational Rose.

A diagram of a movie

Description automatically generated

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 entity-relationship diagram for the CONFERENCE\_REVIEW database and build the design using a data modeling tool such as Erwin or Rational Rose.

A diagram of a company

Description automatically generated with medium confidence

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)

First, specify the foreign keys for this schema, stating any assumptions you make. Next, populate the relations with a few sample tuples, and 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.

Assuming that each agent has a unique ID and each apartment has a unique number, Agent\_id and Apartment are primary keys in the AGENT and APARTMENT relations, respectively. Agent\_id is a foreign key in the BOOKING relation, and Apartment# is a foreign key in OPTION and BOOKING.

Figure 1 below shows a schema of the four relations with three example tuples in each one.

A group of white squares with numbers and letters

Description automatically generated

Figure : Example Schema

Figure 2 shows an example of violating the referential integrity constraint upon an insertion into the BOOKING relation. A group of squares with numbers and text

Description automatically generated

Figure : Violation of Referential Integrity Constraint

The inserted tuple, highlighted in red, violates the referential integrity constraint because the Agent\_id value “004” does not exist in the AGENT relation. As mentioned previously, Agent\_id in BOOKING is a foreign key with Agent\_id in AGENT being the primary key. To comply with the constraint, the value of the foreign key attribute for all tuples in BOOKING must correspond to an existing primary key attribute value in AGENT.

Figure 3 shows an example of an insertion that satisfies the referential integrity constraint.A group of squares with numbers and text

Description automatically generated

Figure : Insertion that Complies with Referential Integrity Constraint

Now, the inserted tuple has an Agent\_id value that corresponds to an existing one in the AGENT relation.

5.19

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 |

a. 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?)

**The LocalPhone and CellPhone attributes are missing the area codes or first three digits of a US phone number.**

b. Would you store this additional information in the LocalPhone and CellPhone attributes or add new attributes to the schema for STUDENT?

**I would store the area codes in the existing attributes rather than adding new attributes because the data in the existing attributes is not very useful on its own, thus it would be better for users who are looking for the student’s phone numbers to find all the necessary information in one attribute.**

c. 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)?

**The advantage of splitting the Name attribute into three attributes is that it allows the user more control in searching for data. A user may be interested in only the last name of the student, and by splitting the Name attribute, they will receive one name in their query rather than three names that they must manually parse to get the information relevant to them. The disadvantages are that it could lead to inaccurate data entries, and it adds infrastructure to the database. This question asks to consider splitting the attribute into three attributes for first, middle, and last name, yet the example tuple shown above has four names. Looking at the name alone does not tell us whether “William” is a second middle name or the first half of a last name. This confusion could assign names to the wrong attribute, thus taking a user more time to find the data for which they are requesting. Also, additional attributes make the relation larger and would require storing more metadata to describe the added attributes.**

d. What general guideline would you recommend for deciding when to store information in a single attribute and when to split the information.

**If splitting the information better fits the users’ needs, then split the information. If not, don’t.**

e. Suppose the student can have between 0 and 5 phones. Suggest two different designs that allow this type of information.

**Each tuple in student could have 5 attributes for the possible phone numbers (i.e. Phoen1, Phone2, etc.) or a new relation called PHONE could be created in which each tuple is a phone number that uses SSN of the student who owns the phone number as a foreign key.**