CHAPTER 1: DATABASES AND DATABASE USERS

Answers to Selected Exercises

1.8 - Identify some informal queries and update operations that you would expect to apply to the database shown in Figure 1.2.

Answer:
(a) (Query) List the names of all students majoring in Computer Science.

(b) (Query) What are the prerequisites of the Database course?

(c) (Query) Retrieve the transcript of Smith. This is a list of <CourseName, SectionIdentifier, Semester, Year, Grade> for each course section that Smith has completed.

(d) (Update) Insert a new student in the database whose Name=Jackson, StudentNumber=23, Class=1 (freshman), and Major=MATH.

(e) (Update) Change the grade that Smith received in Intro to Computer Science section 119 to B.

1.9 - What is the difference between controlled and uncontrolled redundancy?

Answer:
Redundancy is when the same fact is stored multiple times in several places in a database. For example, in Figure 1.5(a) the fact that the name of the student with StudentNumber=8 is Brown is stored multiple times. Redundancy is controlled when the DBMS ensures that multiple copies of the same data are consistent; for example, if a new record with StudentNumber=8 is stored in the database of Figure 1.5(a), the DBMS will ensure that StudentName=Smith in that record. If the DBMS has no control over this, we have uncontrolled redundancy.

1.10 - Specify all the relationships among the records of the database shown in Figure 1.2.

Answer:
(a) Each SECTION record is related to a COURSE record.

(b) Each GRADE_REPORT record is related to one STUDENT record and one SECTION record.

(c) Each PREREQUISITE record relates two COURSE records: one in the role of a course and the other in the role of a prerequisite to that course.

1.11 - Give some additional views that may be needed by other user groups for the database shown in Figure 1.2.

Answer:

CHAPTER 2: DATABASE SYSTEM CONCEPTS AND ARCHITECTURE

Answers to Selected Exercises

2.12 - Think of different users for the database of Figure 1.2. What type of applications would each user need? To which user category would each belong and what type of interface would they need?

Answer:
(a) Registration Office User: They can enter data that reflect the registration of students in sections of courses, and later enter the grades of the students. Applications can include:
- Register a student in a section of a course
- Check whether a student who is registered in a course has the appropriate prerequisite courses
- Drop a student from a section of a course
- Add a student to a section of a course
- Enter the student grades for a section

Application programmers can write a number of canned transactions for the registration office end-users, providing them with either forms and menus, or with a parametric interface.

(b) Admissions Office User: The main application is to enter newly accepted students into the database. Can use the same type of interfaces as (a).

(c) Transcripts Office User: The main application is to print student transcripts. Application programmers can write a canned transaction using a report generator utility to print the transcript of a student in a prescribed format. The particular student can be identified by name or social security number. Another application would be to generate grade slips at the end of each semester for all students who have completed courses during that semester. Again, this application could be programmed using a report generator utility.

2.13 - No solution provided.

2.14 - if you were designing a Web-based system to make airline reservations and to sell airline tickets, which DBMS Architecture would you choose from Section 2.5? Why? Why would the other architectures not be a good choice?

Answer:
2.5.4 Three-Tier Client/Server Architecture for Web Application is the best choice. The Client consists of Web User Interface. The Web Server contains the application logic which includes all the rules and regulations related to the reservation process and the issue of tickets; the Database Server contains the DBMS.

2.5.1 Centralized DBMS Architecture would not work since the user interface and database server are on different machines for a web-based system.

2.5.2 Basic Client/Server Architecture and 2.5.3 Two-Tier Client/Server Architecture would work if the Business Logic can reside on server other than the DBMS Server. In general, if the business logic was on the DBMS Server, it will put an excessive burden on the server. If the business logic were to reside on the web client, it will burden the communication network as well a possibly thin client.

2.15 - Consider Figure 2.1. In addition to constraints relating the values of columns in one table to columns in another table, there are also constraints that impose restrictions on

values in a column or a combination of columns within a table. One such constraint forces that a column or a group of columns must be unique across all rows in the table. For example, in the STUDENT table, the StudentNumber column must be unique (to prevent two different students from having the same StudentNumber). Identify the column or the group of columns in the other tables that must be unique across all rows in the table?

Answer:

Table	Column(s)
COURSE	CourseNumber Since this contains the combination of the department and the number that must be unique within the department. Note we will overlook the fact this does not accommodate a department from offering several "Special Topics" course with the same CourseNumber but different titles. We could make this a combination of CourseNumber and CourseName, but this is more succeptible to someone mistyping while entering data.
PREREQUISITE	The combination of CourseNumber and PrerequisiteNumber
SECTION	SectionIdentifier We assume that no two sections can have the same SectionIdentifier. If we were to consider that SectionIdentifier is unique only within a given course offered in a given term (such as section 2 of CS101) then the answer changes to the combination of SectionIdentifier, CourseNumber, Semester, and Year.
GRADE_REPORT	StudentNumber and SectionIdentifier As per assumption stated in SECTION, the SectionIdentifier will be different if a student takes the same course or a different course in another term.

CHAPTER 3: DATA MODELING USING THE ENTITY-RELATIONSHIP (ER) MODEL

Answers to Selected Exercises

3.16 - Consider the following set of requirements for a UNIVERSITY database that is used to keep track of students' transcripts. This is similar but not identical to the database shown in Figure 1.2:

(a) The university keeps track of each student's name, student number, social security number, current address and phone, permanent address and phone, birthdate, sex, class (freshman, sophomore, ..., graduate), major department, minor department (if any), and degree program (B.A., B.S., ..., Ph.D.). Some user applications need to refer to the city, state, and zip of the student's permanent address, and to the student's last name. Both social security number and student number have unique values for each student.

(b) Each department is described by a name, department code, office number, office phone, and college. Both name and code have unique values for each department.

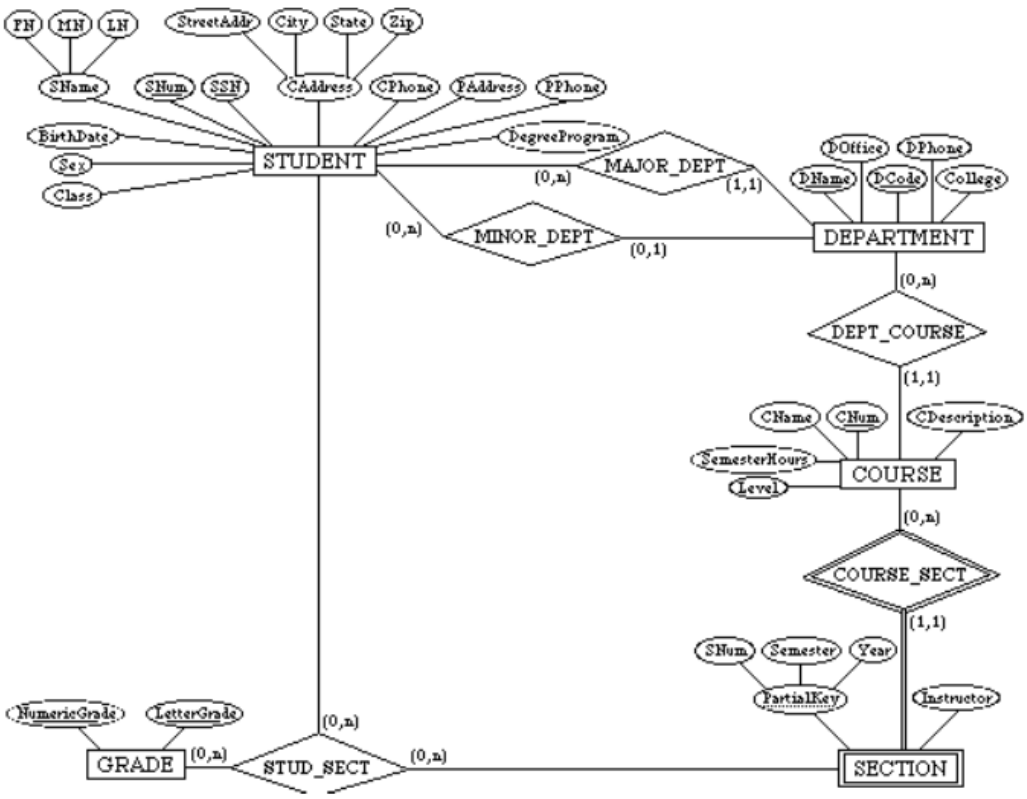
(c) Each course has a course name, description, course number, number of semester hours, level, and offering department. The value of course number is unique for each course.

(d) Each section has an instructor, semester, year, course, and section number. The section number distinguishes different sections of the same course that are taught during the same semester/year; its values are 1, 2, 3, ..., up to the number of sections taught during each semester.

(e) A grade report has a student, section, letter grade, and numeric grade (0, 1, 2, 3, 4 for F, D, C, B, A, respectively).

Design an ER schema for this application, and draw an ER diagram for that schema. Specify key attributes of each entity type and structural constraints on each relationship type. Note any unspecified requirements, and make appropriate assumptions to make the specification complete.

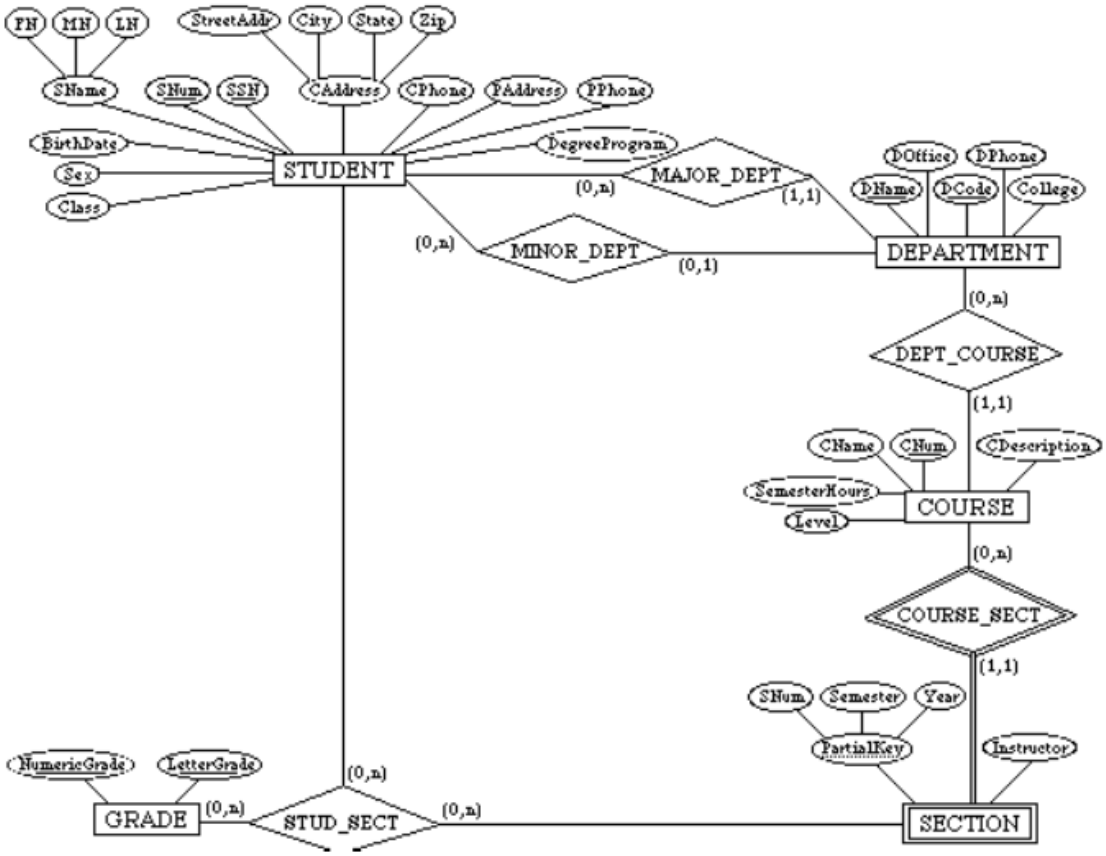
Answer:



ER Schema diagram for exercise 3.16:

3.17 - Composite and multi-valued attributes can be nested to any number of levels. Suppose we want to design an attribute for a STUDENT entity type to keep track of previous college education. Such an attribute will have one entry for each college previously attended, and this entry is composed of: college name, start and end dates, degree entries (degrees awarded at that college, if any), and transcript entries (courses completed at that college, if any). Each degree entry is formed of degree name and the month and year it was awarded, and each transcript entry is formed of a course name, semester, year, and grade. Design an attribute to hold this information. Use the conventions of Figure 7.5.

Answer:
{ PreviousEducation (CollegeName, StartDate, EndDate,
{ Degree (DegreeName, Month, Year) },
{ Transcript (CourseName, Semester, Year, Grade) }) }

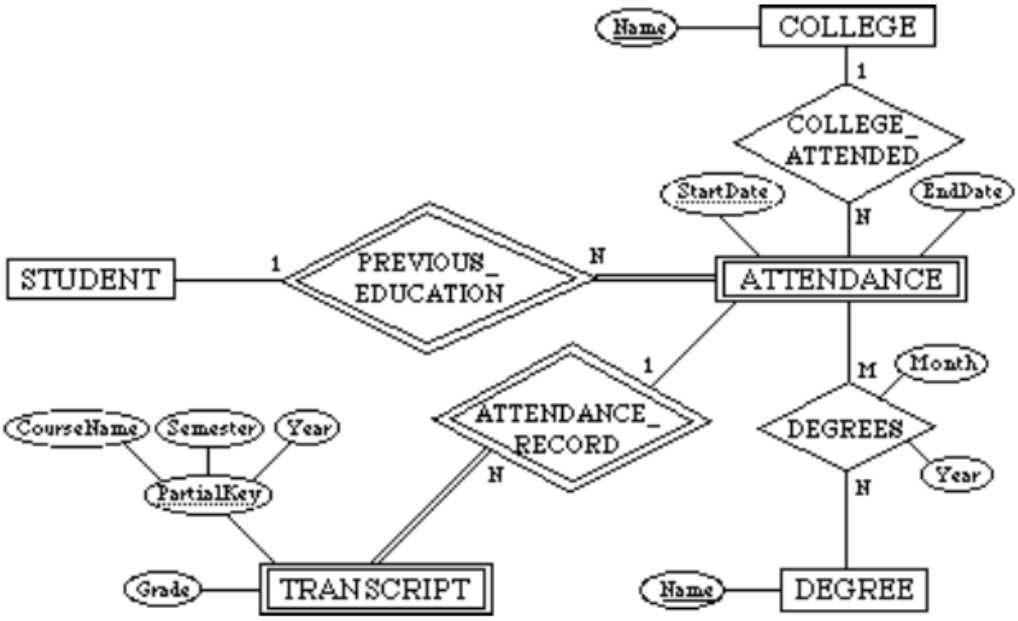


ER Schema Diagram for Exercise 3.16

3.18 - Show an alternative design for the attribute described in Exercise 7.17 that uses only entity types (including weak entity types if needed) and relationship types.

Answer:

This example illustrates a perceived weakness of the ER model, which is: how does the database designer decide what to model as an entity type and what to model as a relationship type. In our solution, we created a weak entity type ATTENDANCE; each (weak) entity in ATTENDANCE represents a period in which a STUDENT attended a particular COLLEGE, and is identified by the STUDENT and the StartDate of the period. Hence, the StartDate attribute is the partial key of ATTENDANCE. Each ATTENDANCE entity is related to one COLLEGE and zero or more DEGREEs (the degrees awarded during that attendance period). The TRANSCRIPT of the STUDENT during each attendance period is modeled as a weak entity type, which gives the records of the student during the attendance period. Each (weak) entity in TRANSCRIPT gives the record of the sudent in one course during the attendance period, as shown in the ER diagram below. Other ER schema designs are also possible for this problem.



3.19 - Consider the ER diagram of Figure 7.20, which shows a simplified schema for an airline reservations system. Extract from the ER diagram the requirements and constraints that resulted in this schema. Try to be as precise as possible in your requirements and constraints specification.

Answer:
(1) The database represents each AIRPORT, keeping its unique AirportCode, the AIRPORT Name, and the City and State in which the AIRPORT is located.

(2) Each airline FLIGHT has a unique number, the Airline for the FLIGHT, and the Weekdays on which the FLIGHT is scheduled (for example, every day of the week except Sunday can be coded as X7).

(3) A FLIGHT is composed of one or more FLIGHT LEGs (for example, flight number CO1223 from New York to Los Angeles may have two FLIGHT LEGs: leg 1 from New York to Houston and leg 2 from Houston to Los Angeles). Each FLIGHT LEG has a DEPARTURE AIRPORT and Scheduled Departure Time, and an ARRIVAL AIRPORT and Scheduled Arrival Time.

(4) A LEG INSTANCE is an instance of a FLIGHT LEG on a specific Date (for example, CO1223 leg 1 on July 30, 1989). The actual Departure and Arrival AIRPORTs and Times are recorded for each flight leg after the flight leg has been concluded. The Number of available seats and the AIRPLANE used in the LEG INSTANCE are also kept.

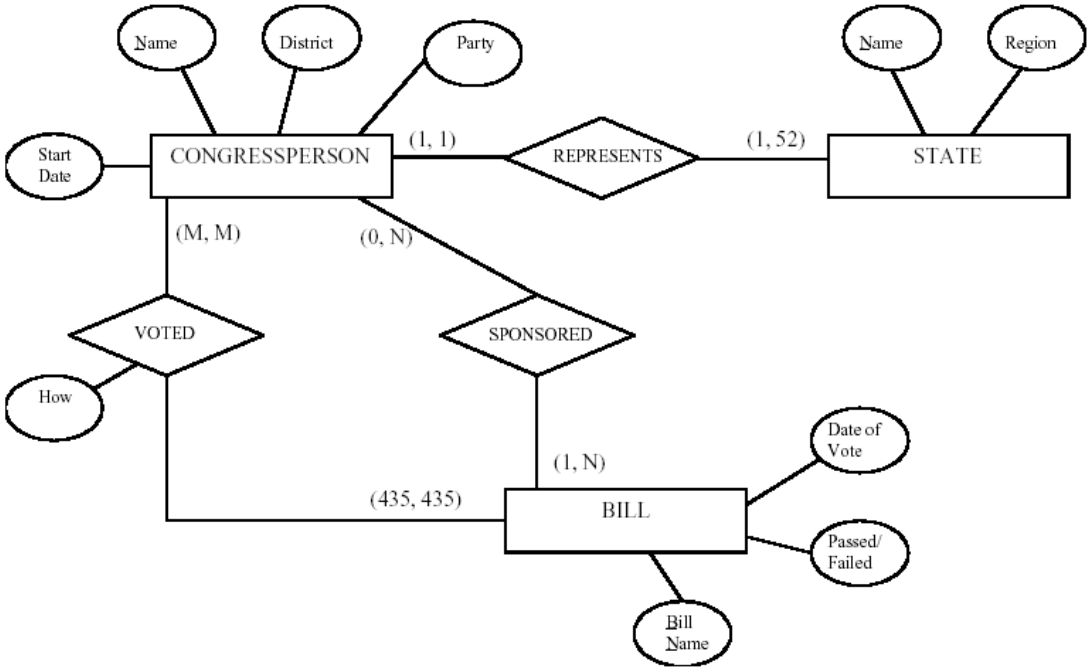
(5) The customer RESERVATIONs on each LEG INSTANCE include the Customer Name, Phone, and Seat Number(s) for each reservation.

(6) Information on AIRPLANEs and AIRPLANE TYPEs are also kept. For each AIRPLANE TYPE (for example, DC-10), the TypeName, manufacturing Company, and Maximum Number of Seats are kept. The AIRPORTs in which planes of this type CAN LAND are kept in the database. For each AIRPLANE, the AirplaneId, Total number of seats, and TYPE are kept.

3.20 - No solution provided.

3.21 -
Additional information:
- There are 435 congresspersons in the U.S. House of Representatives.
- States have between one (AK, DE, MT, ND, SD, VT, and WY) and 52 (CA) representatives.
- M represents number of bills during the 2-year session.

The resulting ER Diagram is shown in Figure A.

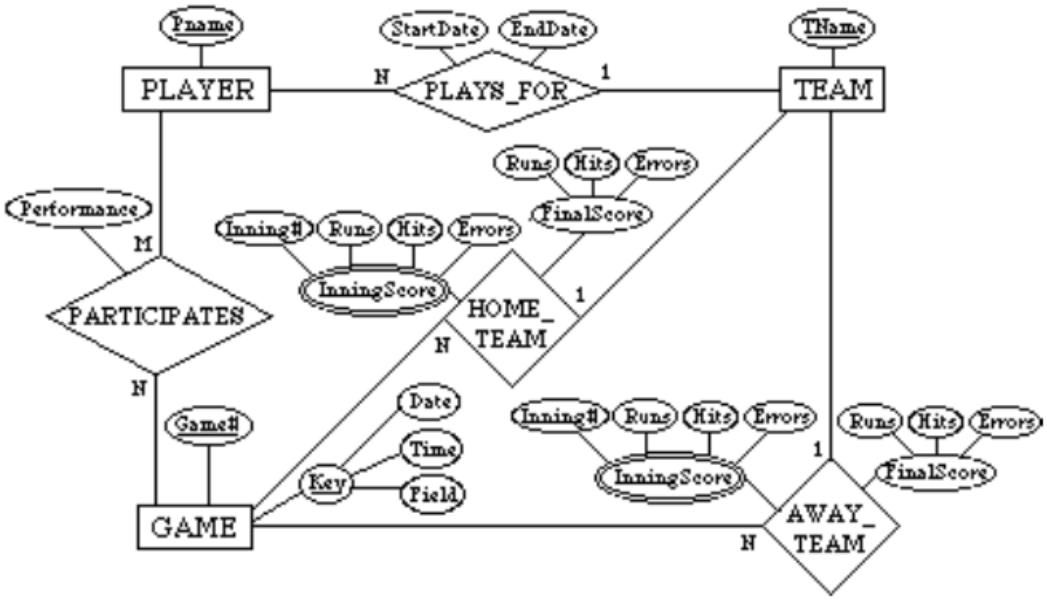


3.22 - A database is being constructed to keep track of the teams and games of a sports league. A team has a number of players, not all of whom participate in each game.
It is desired to keep track of the players participating in each game for each team, the positions they played in that game, and the result of the game. Try to design an ER schema diagram for this application, stating any assumptions you make. Choose your favorite sport (soccer, football, baseball ...).

Answer:
The following design may be used for a baseball league. Here, we assumed that each game in the schedule is identified by a unique Game#, and a game is also identified uniquely by the combination of Date, starting Time, and Field where it is played. The Performance attribute of PARTICIPATE is used to store information on the individual performance of each

player in a game. This attribute can be designed to keep the information needed for statistics, and may be quite complex. One possible design for the Performance attribute may be the following (using the notation of Figure 7.8):

Performance({Hitting(AtBat#, Inning#, HitType, Runs, RunsBattedIn, StolenBases)},
{Pitching(Inning#, Hits, Runs, EarnedRuns, StrikeOuts, Walks, Outs,
Balks, WildPitches)},
{Defense(Inning#, {FieldingRecord(Position, PutOuts, Assists, Errors)})})
Here, performance is a composite attribute made up of three multivalued components: Hitting, Pitching, and Defense. Hitting has a value for each AtBat of a player, and records the HitType (suitable coded; for example, 1 for single, 2 for double, 3 for triple, 4 for home run, 0 for walk, -1 for strikeout, -2 for fly out, ...) and other information concerning the AtBat. Pitching has a value for each inning during which the player pitched. Defense has a value for each inning a player played a fielding position. We can have a less detailed or a more detailed design for the performance of a player in each game, depending on how much information we need to keep in the database. Suitable variations of the ER diagram shown below can be used for other sports.



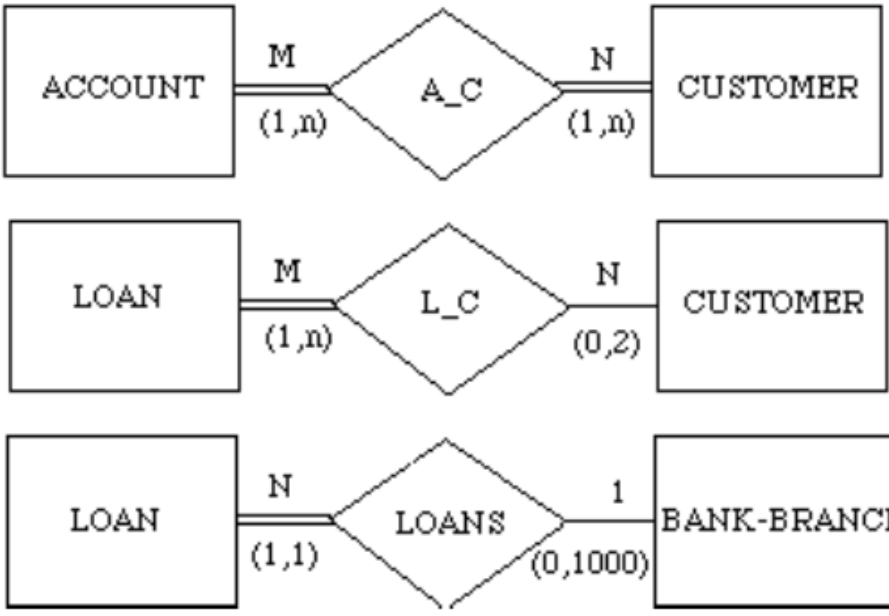
3.23 - Consider the ER diagram shown in Figure 7.21 for part of a BANK database. Each bank can have multiple branches, and each branch can have multiple accounts and loans.

(a) List the strong (nonweak) entity types in the ER diagram.

(b) Is there a weak entity type? If so, give its name, its partial key, and its identifying relationship.

(c) What constraints do the partial key and the identifying relationship of the weak entity type specify in this diagram?

(d) List the names of all relationship types, and specify the (min,max) constraint on each participation of an entity type in a relationship type. Justify your choices.



(e) List concisely the user requirements that led to this ER schema design.

(f) Suppose that every customer must have at least one account but is restricted to at most two loans at a time, and that a bank branch cannot have more than 1000 loans. How does this show up on the (min,max) constraints?

Answer:
(a) Entity types: BANK, ACCOUNT, CUSTOMER, LOAN

(b) Weak entity type: BANK-BRANCH. Partial key: BranchNo.
Identifying relationship: BRANCHES.

(c) The partial key BranchNo in BANK-BRANCH specifies that the same BranchNo value ay occur under different BANKs. The identifying relationship BRANCHES specifies that