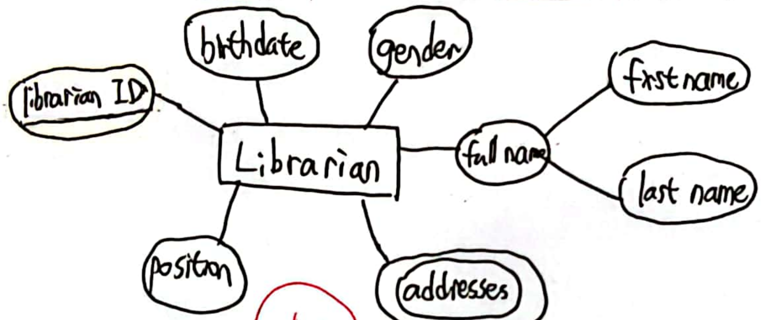
Problem ONE: ER Diagram [20 points]

Consider a library management system database that consists of the following entities: (a) **Librarian**, which has a unique librarian ID and other attributes such as full name (composed of a first name and a last name), birthdate, gender, position, and multiple addresses. (b) **Reader**, which has a unique reader ID, a unique username, and other attributes like gender, membership type, membership duration, and the number of books currently borrowed in total. (c) **Book**, which has a unique book ID and other attributes like book title, authors, publisher, genre, and the number of total copies. (d) **Book Copy**, which has attributes including the ID of the copy, the current status (available/borrowed), and the arrival date of the copy in the library.

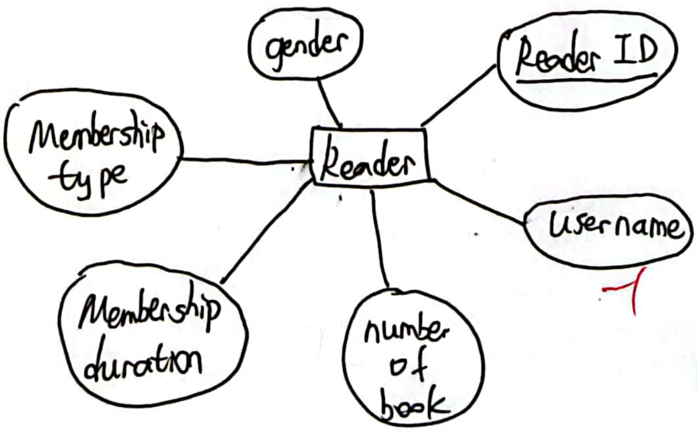
The database also keeps track of three relationships: (a) **Has**, which describes which book has which book copies. (b) **Manage**, which describes which librarian manages which book during a specific period. (c) **Borrow**, which describes which book copy is currently borrowed by which reader, as well as the borrowing date and returning date of the book copy for the reader. Based on the above description, please answer the following questions about the ER diagram of this database:

1. Please draw the ER diagram for the entity type **Librarian**. [4 points]



4

2. Please draw the ER diagram for the entity type **Reader**. [4 points]

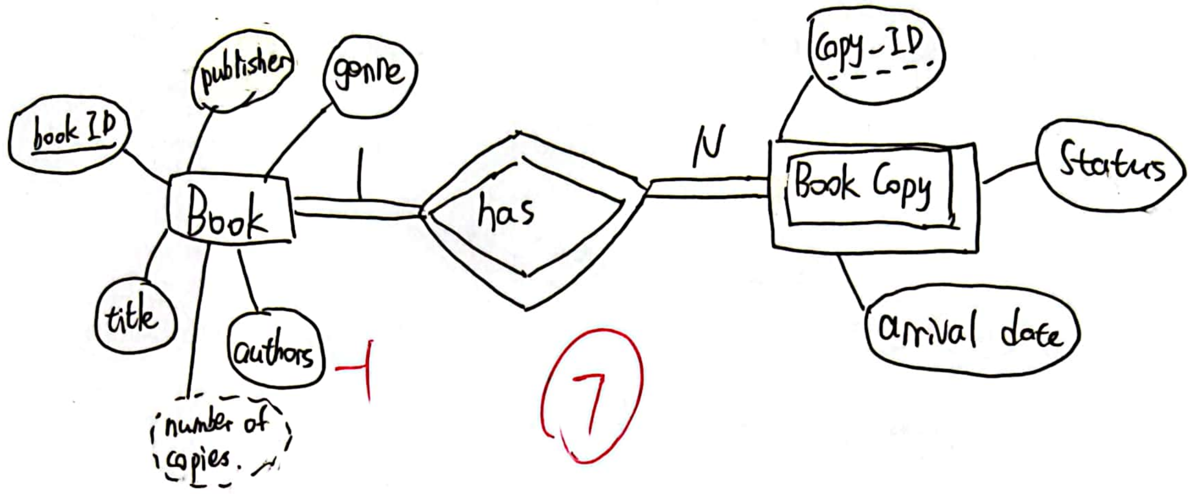


-1

2

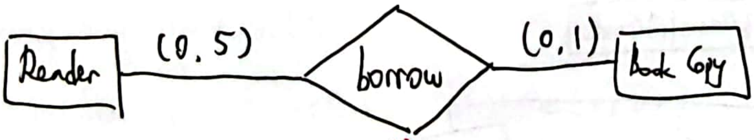
-1

3. Suppose each book has n (n>=1) copies, with the copy ID ranging from 1 to n. Please draw the entity type **Book Copy, Book** and the relationship **Has** between them. [8 points]



Assumption: Suppose <authors> is an atomic attribute of Book.

4. Suppose a reader Department can borrow a maximum of 5 book copies at the same time, and each book Employee copy can be at most borrowed by only one reader at a time. Please draw the ER diagram for the relationship **Borrow** between **Reader** and **Book Copy** by using the **min-max notation**. (The attributes of both entities can be ignored.) [4 points]



-1

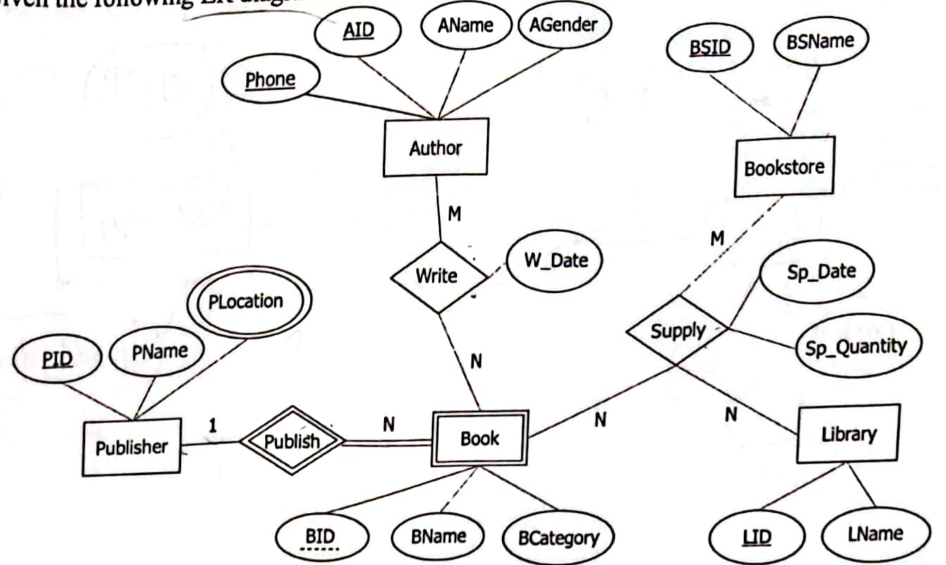
3

扫描全能王 创建

17

Problem TWO: Relational Model [20 points]

Given the following ER diagram for a database of bookstore:



1 Please convert the ER diagram into Relational Schema. Note: you can define a relation in the sample format below. [8 points]

8

EMPLOYEE

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary

DEPENDENT

Essn	Dependent_name	Sex	Bdate	Relationship

Author

Phone	AID	AName	AGender

Bookstore

BSID	BSName

Write

Phone	AID	BID	PID	W-date

Library

LID	LName

supply

BSID	LID	PID	BID	Sp_Date	Sp_Quantity

publisher

PID	PName

Book

PID	BID	BName	Bcategory

Plocation

PID	location

2. How many primary keys, candidate keys, and superkeys are there for the relation 'Author'? [4 points]

1

1 Primary key : (Phone, AID)

1 Candidate key : (Phone, AID)

4 Super keys : $2^2 = 4$

8

3. Assuming that the tables for the entities 'Publisher', 'Author', 'Bookstore', and 'Library' already exist, please create tables for the entity ① Book and the relationship ② Supply, while defining the primary keys and foreign keys using SQL statements. (Hint: you can define the datatype of attributes by yourself). [8 points]

```
create table Book
 ( PID   int   not null,
   BID   int   not null,
   Bname   varchar(20),
   Bcategory   varchar(20),
   primary key (PID, BID),
   foreign key (PID) references Publisher(PID));
```

```
create table Supply
 ( BSID   int   not null,
   LID   int   not null,
   SP_Date   Date,
   SP_Quantity   int,
   primary key (BSID, LID, PID, BID),
   foreign key (BSID) references Bookstore (BSID),
   foreign key (LID) references Library (LID),
   foreign key (PID) references Publisher (PID),
   [illegible] Book ([illegible])
```

PID int not null
BID int not null

Assumption:

① The referenced key in the referenced table has the same name as the foreign key.

5

Problem Three: Integrity Constraints [20 points]

1 Suppose we have a relational database of University system which contains three tables Professor(Prof_id, Name, Department, Gender, Birth_data, Email), Course(Course_id, Title, Department, Prof_id) and Enrollment(Student_id, Course_id, Grade). The current state of the database is shown in the following tables. [15 points]

Professor

Prof_id	Name	Department	Gender	Birth_date	Email
P001	John Doe	Computer Science	Male	May. 5, 1990	johndoe@university.edu
P002	Jane Smith	Mathematics	Female	Jul. 27, 2006	janesmith@university.edu
P003	Richard Roe	Physics	Male	Aug. 13, 1990	richardroe@university.edu
P004	David Johnson	Biology	Male	Dec. 31, 1998	sacraver@hotmail.com
P005	Emily Johnson	Chemistry	Female	Nov. 8, 1996	emilyjohnson@university.edu
P006	Michael Anderson	English	Male	Feb. 17, 1993	michaelanderson@university.edu
P007	Linda White	History	Female	Aug. 13, 1995	lindawhite@university.edu
P008	David Johnson	Computer Science	Male	Apr. 30, 1990	dav.johnson@example.com
P009	Linda White	Physics	Female	Apr. 14, 1993	linw@verizon.net

Course

Course_id	Title	Department	Prof_id
C001	Introduction to Python	Computer Science	P001
C002	Advanced Mathematics	Mathematics	P002
C003	Theoretical Physics	Physics	P003
C004	General Biology	Biology	P004
C005	Organic Chemistry	Chemistry	P005
C006	Shakespearean Literature	English	P006
C007	World History	History	P007
C008	Data Structures	Computer Science	P001
C009	Calculus II	Mathematics	P002

Enrollment

Student_id	Course_id	Grade
S001	C001	A
S002	C001	B
S003	C002	A
S004	C003	C
S005	C003	B
S001	C004	A
S002	C005	B
S006	C006	A
S007	C007	A

(1) Supposing all tables are created, please use the command "Alter Table" to define all primary keys and foreign keys of all tables (Write corresponding SQL statements). [5 points]

Professor: ① alter table Professor add constraint PK_Professor primary key(Prof_id);

Course: ① alter table Course add constraint PK_Course primary key (Course_id);

② alter table Course add constraint FK_Department foreign key (Department) references (Professor (Department));

③ alter table Course add constraint FK_Profid foreign key (Prof_id) references (Professor (Prof_id)

Enrollment ① alter table Enrollment add constraint PK_Enrollment primary key (Student_id, Course_id);

② alter table Enrollment add constraint FK_Course_id foreign key (Course_id) references (Course (Course_id

(2) For a) and b) below, suppose each of the following operations is applied directly to the database. Discuss all integrity constraints violated by each operation if any, and the different actions of enforcing these constraints.

a) Insert < 'S002', 'C001', 73 > into Enrollment. [5 points]

① violates both key constraint and domain constraint. Violates key constraint because there already exists an Enrollment tuple with sid 'S002' and cid 'C001'. Violates domain constraint because grade should not be an integer.

② we may 1>. reject the insertion

2>. changing the value of sid & cid to a non-existing one.

Meanwhile we need to change the Grade to a char(1).

b) Insert <NULL, 'Computational Imaging', 'Computer Science', 'P012' > into Course. [5 points]

① violates both entity integrity constraint and referential integrity constraint. violates entity integrity because Course-ID is the key attribute that can't be null. violates referential constraint because a tuple with P-id as 'P012' does not exist.

② we may 1>. reject the insertion

2>. add a tuple with Prof-id 'P012' into the Professor table.

Meanwhile we need to find a unique Course-id for the tuple instead of 'null' for insertion.

2. Given a relation schema R (A,B,C,D,E) with the function dependency set F={ AB→CD, C→B, D→E, E→A}, please determine whether each of the following functional dependency is in F+. (Hint: no need to show the proof.) [5 points]

1) AC→E True

2) AE→B False

3) BE→D True

4) BC→A False

5) CD→AB True

5

Problem Four: Normalization [20 points]

1. Suppose we have a relation R with attributes A, B, C, D, E, F, G, H and the functional dependencies are: AC→B, BD→E, CE→FG, A→H. Please prove that FD: ACD→ FG holds by using inference rules. [5 points]

1. AC → B (given)
2. ACD → BD (augmentation)
3. BD → E (given)
4. ACD → E (transitive, 2& 3)
5. CE → FG (given)
6. ACD → CE (augmentation)
7. ACD → FG (transitive 5&6)

4

2. Let's consider the following relation R storing the information about album retailers. R(StoreID, StoreAddress, AlbumID, ReleaseYear, Artist, BirthPlace, BirthYear, Inventory, Price).

It has following functional dependencies: [15 points]

StoreID → StoreAddress
AlbumID → {ReleaseYear, Artist}
Artist → {BirthPlace, BirthYear}
{StoreID, AlbumID} → Inventory
Inventory → Price

(1) Identify all the candidate keys in this table. [2 Points]

{StoreID, AlbumID} ✓

2

(2) Is the relation R in 2NF and why? If not, decompose it into Three tables which satisfy 2NF but not 3NF. [5 Points]

No. Because StoreAddress has partial dependency on StoreID and {ReleaseYear, Artist} has partial depency on AlbumID.

R1(StoreID, StoreAddress) R2(AlbumID, ReleaseYear, Artist, Birthplace, BirthYear)

R3(StoreID, AlbumID, Inventory, Price)

5

(3) Does your decomposition in (2) satisfy 3NF and why? If not, normalize it into 3NF. [5 Points]

No. Because transitive function dependecies exist in R2 & R3.
AlbumID → Artist → Birthplace. {StoreID, AlbumID} → Inventory → Price
R1(StoreID, Store Address)
R2A(AlbumID, ReleaseYear, Artist) R2B(Artist, Birthplace, BirthYear)

R3A(StoreID, AlbumID, Inventory) R3B(Inventory, Price)

5

(4) Does your decomposition in (3) satisfy BCNF and why? If not, normalize it into BCNF. [3 Points]

Yes. It's already in BCNF.

Because in each table, all the left-hand sides of the function dependencies are super keys.

3

Problem FIVE: SQL [20 points]

Given the following four relations about the information of course offerings in a university. [20 points]

- Student (**StudentID**: integer, **Name**: string, **Age**: integer, **Department**: string, **GPA**:float)
  // describe the student's information including ID, name, age, GPA, and the major department the student belongs to.

- Teacher (**TeacherID**: integer, **Name**: string, **Department**: string)
  // describe the teacher's information including ID, name, and the department the teacher belongs to.

- Course (**CourseID**: integer, **Name**: string, **Department**: string, **TeacherID**: integer)
  // describe the course's information including ID, name, and the department which offers the course.

- Grade (**StudentID**: integer, **CourseID**: integer, **Score**: integer)
  // describe which student takes which course and get how many scores in that course

Suppose now we have a valid database state. Answer the following questions by completing missing parts of given SQL statement.

①

5 (1) List the StudentID of students who are majoring in **'Physics'** and have enrolled in more than two courses that are offered by the 'CS' department and are taught by teachers from the **'Math'** Department. Hint: Assume that a student will enroll in the same course at most once.. [5 points]

SELECT s.StudentID

FROM Student s, Teacher t, Course c, Grade g

WHERE s.StudentID = g.StudentID and c.TeachID = t.TeacherID and

GROUP BY s.StudentID

HAVING count(*) > 2;

g.CourseID = c.CourseID

and s.Department = 'Physics' and c.Department = 'CS'

and t.Department = 'Math'

5 (2) List the StudentID of students who have not taken any courses outside their major department. [5 points]

SELECT DISTINCT s.StudentID

FROM Student AS s

WHERE not exists (

SELECT *

FROM Course c, Grade g.

WHERE s.StudentID = g.StudentID, and c.CourseID = g.CourseID

);

and c.Department != s.Department

5 (3) The 'Algorithms' course offered by the 'CS' department has students from different departments. List the average scores of students of different departments and arrange the list in descending order of the scores. Only those departments with more than 5 students enrolled are included. [5 points]

SELECT s.Department, avg(g.Score)

FROM Student AS s, Grade AS g, Course AS c

WHERE s.StudentID = g.StudentID and g.CourseID = c.CourseID and c.Department = 'CS' and C.Name = 'Algorithms'

GROUP BY s.Department

HAVING count(*) > 5

ORDER BY avg(g.Score) desc;

5 (4) The school wants to identify exceptional students to mentor incoming freshmen. List the StudentID and Department of students who have a GPA greater than 3.6 or have scored above 90 in at least one course **within their major department.** [5 points]

(SELECT s.StudentID, s.Department

FROM Student s

WHERE s.GPA > 3.6)

union

(SELECT g.StudentID, s.Department

FROM Grade g, Student s, Course c

WHERE g.StudentID = s.StudentID and c.CourseID = g.CourseID and c.Department = s.Department and g.Score > 90);