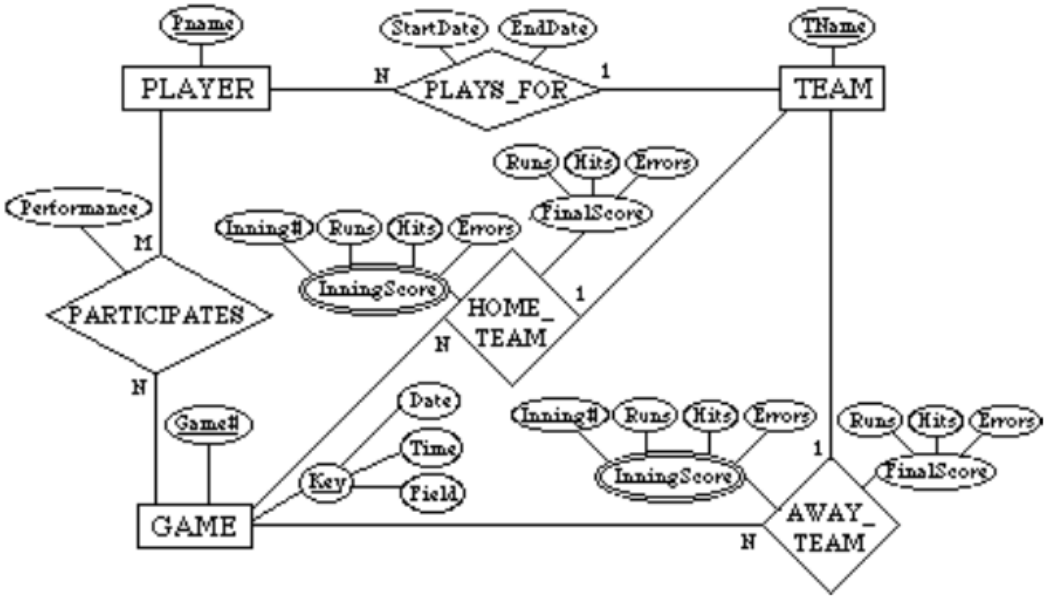
1. A database is being constructed to keep track of the teams and games of a sports league.

- Each team has a name and a number of players
- Each player has a name
- There is a home team and away team for each game
- Not all players participate in each game.
- For each game, we keep track of the date, time, the field and the final score.
- We also keep track of the players participating in each game for each team, the positions they played in that game, and their performance.

a) Design an ER schema diagram for this application, stating any assumptions you make.
b) Map the ER schema to a relational schema. Choose your favorite sport (soccer, football, baseball, etc).

Answer:

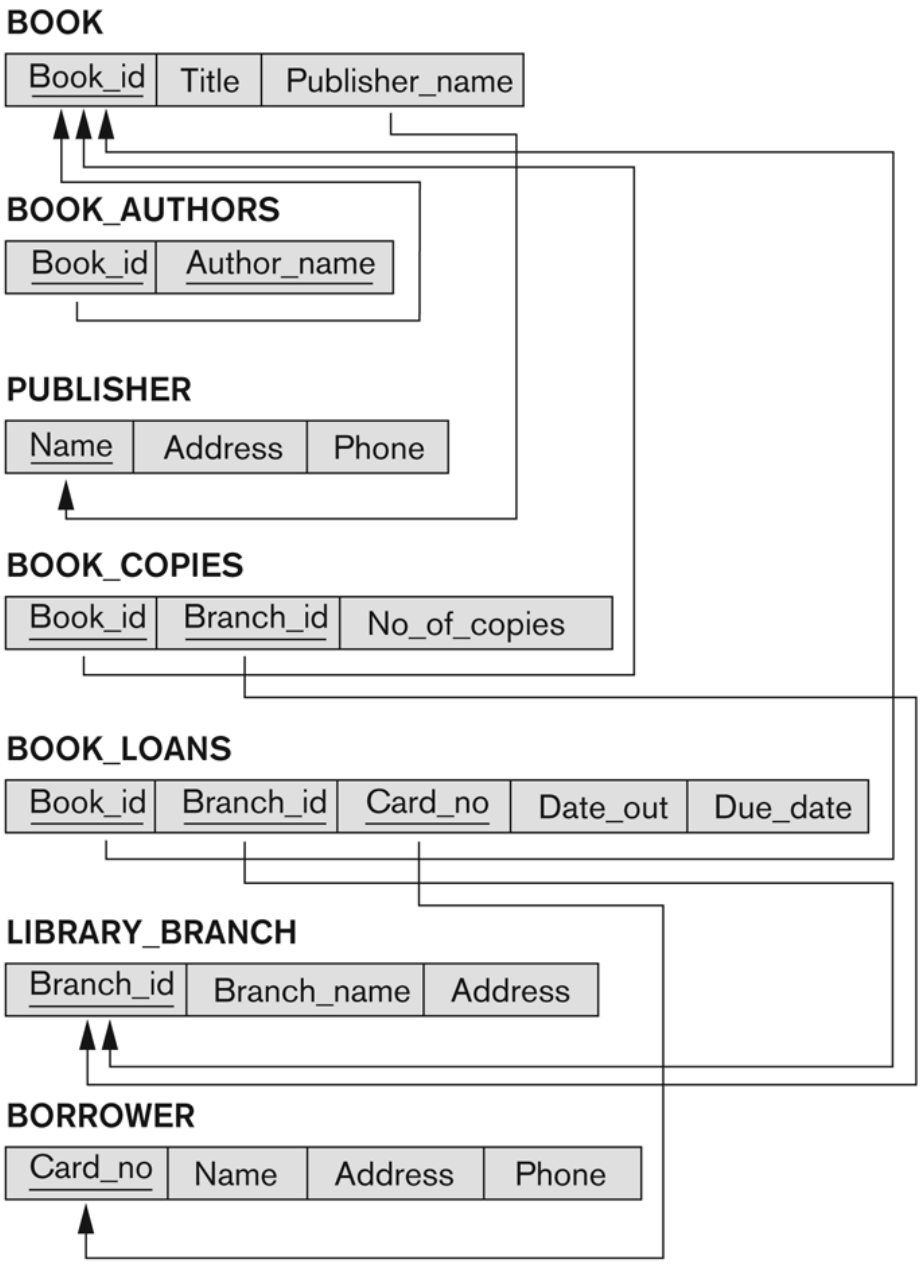
The following design may be used for a baseball league. Here, we assumed that each game in the schedule is identified by a unique Game#, and a game is also identified uniquely by the combination of Date, starting Time, and Field where it is played. The Performance attribute of PARTICIPATE is used to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



(Foreign key*)
Player: Pname, TName*, StartDate, EndDate
Team: TName
Game: Game#, Date, Time, Field, Home_Team*, Home_Runs, Home_Hits, Home_errors, Away_Team*, Away_Runs, Away_Hits, Away_Errors
Participate: Pname*, Game#*, Performance
InningScore: Game#*, TName*, Inning#, Runs, Hits, Errors

2. Consider the LIBRARY relational database schema shown below that is used to keep track of books, borrowers, and book loans. Write relational algebra expressions and SQL queries for each of the following queries.

a) Retrieve the names of the branches and the number of copies of the book titled The Lost Tribe owned by each branch.
b) Retrieve the names of all borrowers who do not have any books checked out.
c) Retrieve the names, addresses, and numbers of books checked out for all borrowers who have more than five books checked out.



Relational algebra expressions (Note: We use **S** for SELECT, **P** for PROJECT, * for NATURAL JOIN, **-** for SET DIFFERENCE, **F** for AGGREGATE FUNCTION)

(a) Retrieve the branch name and the number of copies of the book titled 'Steve Jobs' owned by each library branch.

P Branch_Name, No_Of_Copies ((**S** Title='Steve Jobs' (BOOK)) * BOOKCOPIES * LIBRARY_BRANCH)

```
SELECT Branch_Name, No_Of_Copies
FROM BOOK b, BOOK_COPIES c, LIBRARY_BRANCH l
WHERE b.book_id = c.book_id
AND c.branch_id = l.branch_id
AND Title='Steve Jobs'
```

(b) Retrieve the names of all borrowers who do not have any books checked out.

NO_CHECKOUT_B <-- **P** CardNo (BORROWER) **-** **P** CardNo (BOOK_LOANS)
RESULT <-- **P** Name (BORROWER * NO_CHECKOUT_B)

```
SELECT Name
FROM BORROWER B
WHERE NOT EXISTS
( SELECT *
FROM BOOK_LOANS L
WHERE B.Card_No = L.Card_No )
```

```
select name
from borrower b
where card_no not in
(select card_no
 from book_loans)
```

(c) Retrieve the names, addresses, and numbers of books checked out for all borrowers who have more than five books checked out.

B(CardNo,TotalCheckout) <-- CardNo **F** COUNT(BookId) (BOOK_LOANS)
B5 <-- **S** TotalCheckout > 5 (B)
RESULT <-- **P** Name,Address,TotalCheckout (B5 * BORROWER)

```
SELECT B.Name, B.Address, COUNT(*)
FROM BORROWER B, BOOK_LOANS L
WHERE B.Card_No = L.Card_No
GROUP BY B.CardNo, B.name, B.address
HAVING COUNT(*) > 5
```