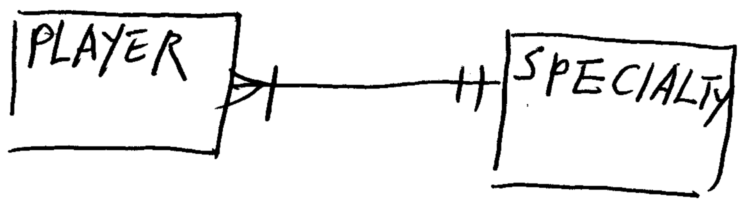
EE4791 Database Systems -Tutorial 9 and Sample Answer

1. Consider the following normalized relations for a sports league:

   TEAM(<u>TeamID</u>, TeamName, TeamLocation)
   PLAYER(<u>PlayerID</u>, PlayerFirstName, PlayerLastName, PlayerDateOfBirth, PlayerSpecialtyCode)
   SPECIALTY(<u>SpecialtyCode</u>, SpecialtyDescription)
   CONTRACT(<u>TeamID, PlayerID, StartTime, EndTime</u>, Salary)
   LOCATION(<u>LocationID</u>, CityName, CityState, CityCountry, CityPopulation)
   MANAGER(<u>ManagerID</u>, ManagerName, ManagerTeam)

   What recommendations would you make regarding opportunities for denormalization? What additional information would you need to make fully informed denormalization decisions?



Are they related in this way ?

Answer

One possibility for denormalization would be the inclusion of SpecialtyDescription attribute in the PLAYER relation. It appears that the PlayerSpecialtyCode refers to the SpecialtyCode in the SPECIALTY relation, using SPECIALTY as reference information for Players. To make the right decision, you would need to know more about Player specialties, in terms of occurrence (can they have more than one, in the future?) and the expected size of the SpecialtyDescription attribute. In addition to the reintroducing anomalies disadvantage to maintenance, denormalizing the relations by adding the SpecialtyDescription data to the PLAYER relation might be adding a lot of redundant information that could take up a great deal of storage.

Two other opportunities for further denormalization and questions to be asked before decision are as follows:

- Combing attributes from LOCATION that are frequently accessed together with TEAM into TEAM: What is the meaning and the contents of the TeamLocation attribute in the TEAM relation? Is this a code used to reference the LocationID in the LOCATION relation? Or is this a character field storing different information about the Team's location? Can a TEAM instance have more than one location?
- Combining attributes from TEAM that are frequently accessed together with MANAGER into MANAGER: What is the meaning and the contents of the ManagerTeam attribute in the MANAGER relation? Does this attribute refer to the TeamID attribute in the TEAM relation?

2. Consider the 3NF relational schema shown in Figure 1. Assume that the most important reports that the organization needs are as follows:

   a) A list of project assignments of a given developer
   b) A list of the total costs for all projects
   c) For each team, a list of its membership history
   d) For each country, a list of all projects, with projected end dates, in which the country’s developers are involved
   e) For each year separately, a list of all the assignments that were for the projects completed during that year.

   Based on the limited information, make a recommendation regarding the indexes that you would create for this database. Choose two relations and provide the SQL command that you would use to create secondary indexes for the relations.

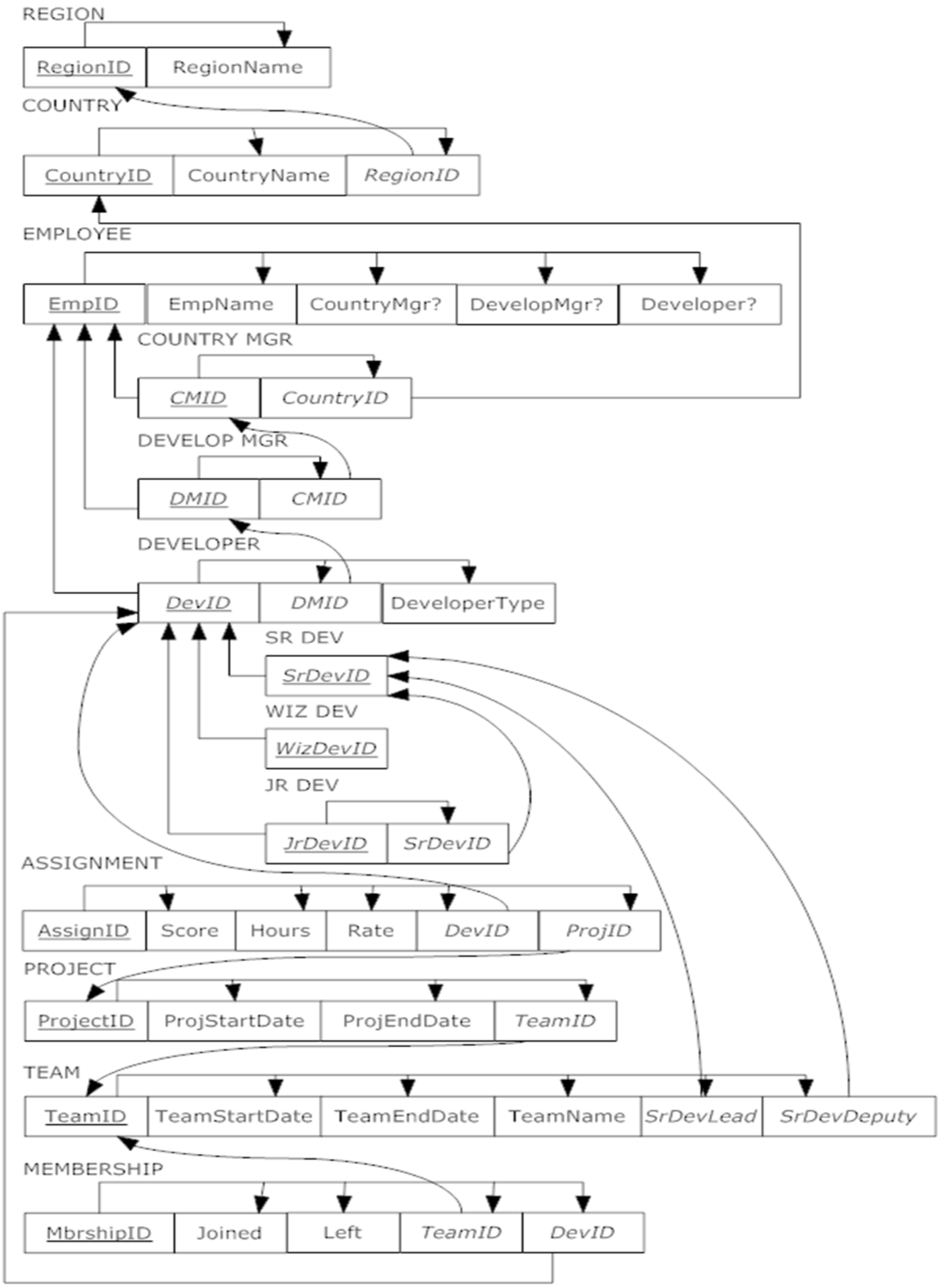


Figure 1. A 3NF Relational Schema

Answer

All Primary Keys in each relation should receive an index. In addition, based on the reporting needs, the following indices could be useful for improving the performance for generating each of the given reports:

a) DevID in Assignment relation: to speed up the direct accesses of ASSIGNMENT by DevID
b) ProjID in Assignment relation: to speed up the join of PROJECT with ASSIGNMENT by ProjID
c) TeamID in Membership relation: to speed up the join of TEAM with MEMBERSHIP by TeamID
d) CountryID in Country Mgr relation, CMID in Develop Mgr relation, DMID in Developer relation, and DevID in Assignment relation: to speed up the join of COUNTRY with COUNTRY MGR by CountryId, with DEVELOP MGR by CMID, with DEVELOPER by DMID, with ASSIGNMENT by DevID
e) ProjID in Assignment relation, and ProjEndDate in Project relation: the former is to speed up the join of PROJECT with ASSIGNMENT by ProjID, and the latter is to speed up the ordering of the result by year

```
CREATE INDEX ASSIGNDEVINDX ON ASSIGNMENT(DevID);
CREATE INDEX ASSIGNPROJINDX ON ASSIGNMENT(ProjID);
CREATE INDEX PROJCOMPLETE ON PROJECT(ProjEndDate);
```

3. Create a join index on the CourseCode fields of the Course_T and Registration_T tables in Figure 2.

Course_T

RowID	CourseCode	CourseTitle
2001	EE4791	Database Systems
2002	EE4717	Web Application Design
2003	EE4190	Introduction to Modern Radar

Registration_T

RowID	StudentID	CourseCode	Grade
1001	A001	EE4791	A
1002	A001	EE4717	A
1003	B001	EE4791	A+
1004	C001	EE4791	B

Figure 2. An instance of Course_T and Registration_T

Answer

Join Index on CourseCode filds of Course_T and Registration_T tables

CourseRowID	RegsitrationRowID	CourseCode
2001	1001	EE4791
2001	1003	EE4791
2001	1004	EE4791
2002	1002	EE4717

4. Consider the following normalized relations for sports leagues in global scope:

   LEAGUE(<u>LeagueID</u>, LeagueName, LeagueLocation)
   TEAM(<u>TeamID</u>, TeamName, TeamLocation, TeamLeague)
   PLAYER(<u>PlayerID</u>, PlayerFirstName, PlayerLastName, PlayerDateOfBirth, PlayerSpecialtyCode)
   SPECIALTY(<u>SpecialtyCode</u>, SpecialtyDescription)
   CONTRACT(<u>TeamID, PlayerID, StartTime, EndTime</u>, Salary)
   LOCATION(<u>LocationID</u>, CityName, CityState, CityCountry, CityPopulation)
   MANAGER(<u>ManagerID</u>, ManagerName, ManagerTeam)

   The following database operations are frequently used:

   a) Maintenance (insert, update and delete) of these relations
   b) Reporting players by team
   c) Reporting players by team and specialty
   d) Reporting players ordered by salary (the highest)
   e) Reporting teams and players by city

   Based on the above-mentioned information:

   a) Identify the foreign keys
   b) Make a recommendation regarding the indexes that you would create for this database.

   Explain how you used the list of operations to arrive at your recommendation.

Answer

a) The following foreign keys (shown in *italic*) are identified for enforcing the accuracy and semantics in the maintenance of the given relations:

TEAM(TeamID, TeamName, *TeamLocation*, *TeamLeague*)
PLAYER(PlayerID, PlayerFirstName, PlayerLastName, PlayerDateOfBirth, *PlayerSpecialtyCode*)
SPECIALTY(SpecialtyCode, SpecialtyDescription)
CONTRACT(TeamID, *PlayerID*, StartTime, EndTime, Salary)
LOCATION(LocationID, CityName, CityState, CityCountry, CityPopulation)
MANAGER(ManagerID, ManagerName, *ManagerTeam*)
LEAGUE(LeagueID, LeagueName, *LeagueLocation*)

b) Recommended indices and explanation:

For the performance of all the insertion, updating and deletion in these relations, all Primary Keys, TeamID, PlayerID, SpecialtyCode, LocationID, ManagerID, LeagueID, should be defined as indexes. We would also create a surrogate key of ContractID for the CONTRACT relation, and also creating an index on that primary key. These indexes will help in achieving better performance for the maintenance of the relations.

Additionally, the following indices could be useful for improving the performance for generating each of the given reports:

1. TeamID and PlayerID on CONTRACT: to speed up the ordering of the result of joining CONTRACT with PLAYER by PlayerID, by TeamID and PlayerID
2. TeamID on CONTRACT, and PlayerSpecialityCode and PlayerID on PLAYER: The former is to speed up the ordering of the result of joining by CONTRACT with PLAYER by PlayerID, by TeamID. The latter is to speed up the ordering of the result further by PlayerSpecialityCode and PlayerID.
3. PlayerID on CONTRACT: to speed up the ordering of the result of joining CONTRACT with PLAYER by PlayerID, by PlayerID. Note

that we do not recommend the index, Salary and PlayerID on CONTRACT as a player may play for different teams during the same period. In such a case, we will have to sum up all his/her salaries.

4. TeamLocation on TEAM, and TeamID and PlayerID on CONTRACT: The former is to speed up the ordering of the result of joining CONTRACT with TEAM by TeamID, by TeamLocation that represents city. The latter is to speed up the ordering of the result further by TeamID and PlayerID.

5. The following table shows some simple student data as of the date 06/20/2010:

Key	**Name**	**Major**
001	Amy	Music
002	Tom	Business
003	Sue	Art
004	Joe	Math
005	Ann	Engineering

The following transactions occur on 06/21/2010:

- Student 004 changes major from Math to Business.
- Student 005 is deleted from the file.
- New student 006 is added to the file: Name is Jim, Major is Phys Ed.

The following transactions occur on 06/22/2010:

- Student 003 changes major from Art to History.
- Student 006 changes major from Phys Ed to Basket Weaving.

Do the following:

a. Construct tables to represent transient data for 06/21/2010 and 06/22/2010, reflecting these transactions.
b. Construct tables represent periodic data for 06/21/2010 and 06/22/2010, reflecting these transactions.

Answer

a.

Transient (06/21)

Key	Name	Major
001	Amy	Music
002	Tom	Business
003	Sue	Art
004	Joe	Business
006	Jim	Phys Ed

Transient (06/22)

Key	Name	Major
001	Amy	Music
002	Tom	Business
003	Sue	History
004	Joe	Business
006	Jim	Bskt Weav

b. It should be noted that the actual PK of the rows of this table is a combination of the original Key and the Date fields.

Periodic (06/21)

Key	**Date**	**Name**	**Major**	**Action**
001	06/20	Amy	Music	C
002	06/20	Tom	Business	C
003	06/20	Sue	Art	C
004	06/20	Joe	Math	C
004	06/21	Joe	Business	U
005	06/20	Ann	Engineering	C
005	06/21	Ann	Engineering	D
006	06/21	Jim	Phys Ed	C

Periodic (06/22)

Key	**Date**	**Name**	**Major**	**Action**
001	06/20	Amy	Music	C
002	06/20	Tom	Business	C
003	06/20	Sue	Art	C
003	06/22	Sue	History	U
004	06/20	Joe	Math	C
004	06/21	Joe	Business	U
005	06/20	Ann	Engineering	C
005	06/21	Ann	Engineering	D
006	06/21	Jim	Phys Ed	C
006	06/22	Jim	Bskt Weav	U

6. Compare and contrast star schema with normalized relational data model.

Answer

The star schema is a denormalized implementation of the relational data model. The fact table plays the role of a normalized n-ary associative entity that links together the instances of the various dimensions. Usually, the dimension tables are in second normal form or possibly (but rarely) in third normal form. The dimension tables are denormalized and because they are not updated nor joined with one another, provide an optimized user view for specific information needs but could not be used for operational purposes.