**Assignment Five**

**Problem One**

Consider the following two relations for Millenium College:

|  |
| --- |
| STUDENT(StudentID, StudentName, CampusAddress, GPA)  REGISTRATION(StudentID, CourseID, Grade) |

Following is a typical query against these relations

|  |
| --- |
| SELECT Student\_T.StudentID, StudentName, CourseID, Grade  FROM Student\_T, Registration\_T  WHERE Student\_T.StudentID =  Registration\_T.StudentID  AND GPA > 3.0  ORDER BY StudentName; |

1. On what attributes should indexes be defined to speed up the query? Give the reasons for each attribute selected.

Indexes can be defined on StudentID and GPA, each sequentially. StudentID is likely commonly selected and can be assumed to be an integer allowing for fast access. Similarly, storing GPA sequentially would mean that upon scanning the database, upon reaching 3.0 GPA, all students following could be selected.

1. Write SQL commands to create indexes for each attribute you defined in part a.

As primary keys are automatically indexed, I need only write one for GPA.

|  |
| --- |
| CREATE INDEX GPA\_IDX ON STUDENT(GPA) |

**Problem Seven**

When a student has not chosen a major at a University, the university often enters a value of “Undecided” for the major field. Is “Undecided” a way to represent the null value? Should it be used as a default value? Justify your answer carefully.

Having worked alongside several committees in the University, I would argue that this is not a good way to represent the null value. For sake of data, it’s important to know specifically how may students are truly undecided. During my application to Austin Peay, I wasn’t even offered a choice of major until I signed up for my orientation session. I can see the reasoning for it being the default value, but for sake of data analysis, this is not a good choice of default value.

**Problem Nine**

Consider the following normalized relations for a sports league:

|  |
| --- |
| TEAM(TeamID, TeamName, TeamLocation)  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) |

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

As is addressed in problem fourteen, one would want to know what kind of database operations are commonly performed. For example, consider a talent scout that is interested in recruiting certain players with certain specialties. They are going to want to know if a player is available, their age, and their specialties. So, combinations of the contract column and player column could happen quite often and so it might be valuable to denormalize the two relations into one. This logic could be further applied to players and their specialties value outside of just the ID.

**Problem Fourteen**

Consider the relations specified in Exercise 9. Assume that the database has been implemented without denormalization. Further assume that the database is global in scope and covers thousands of leagues, tens of thousands of teams, and hundreds of thousands of players. In order to accommodate this, a new relation has been added:

|  |
| --- |
| LEAGUE(LeagueID, LeagueName, LeagueLocation) |

In addition, TEAM has an additional attribute TeamLeague. The following database operations are typical:

* Adding new players
* Adding new player contracts
* Updating player specialty codes
* Updating city populations
* Reporting players by team
* Reporting players by team and specialty
* Reporting players ordered by salary
* Reporting teams and their players by city

1. Report the foreign keys.

TeamLeague is a foreign key in Team. TeamLocation is a foreign key in Team. PlayerSpecialtyCode is a foreign key in Player. ManagerTeam is a foreign key in Manager.

1. Specify the types of indexes you would recommend for this situation. Explain how you used the list of operations described above to arrive at your recommendation.

It is immediately very clear that players need their own index. Almost all of the typical database operations involve operations on players. PlayerID is automatically indexed, and so PlayerLastName would serve well as a secondary index. Similarly, teams could use an index because several of these database operations are functioning on teams. Although there are many leagues though, and likely many managers, they do not show often in terms of the typical database operations and so adding these indexes causes uneccessary complexity.

**Problem Fifteen**

Consider Figure 5-7b. Assuming that the empty rows in the leaves of this index show space where new records can be stored, explain where the record for Sooners would be stored. Where would the record for Flashes be stored? What might happen when one of the leaves is full and a new record needs to be added to that list?

The record for sooners would be stored in the furthest leaf to the right and directly under Seminoles. The Flashes would be stored in the furthest leaf to the left and under the flyers. In the case where a leaf is full, another subindex could be formed further distinguishing letters and thus creating more leaves.