Readings:
chapter 5: "Physical Database Design and Performance" (Hoffer, Ramesh, & Topi)
Assignment:
Chapter 5 (Hoffer, Ramesh, & Topi)
Problems and Exercises 1,5,8,9,13,16,17

1. Consider the following two relations for Millennium 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;

a. On what attributes should indexes be defined to speed up this query? Give the reasons for each attribute selected.
   In STUDENT, add StudentID(PK) as index, because it's unique and can be found easily.
   In REGISTRATION, add StudentID(FK) as index, because it's unique and it's the primary key of STUDENT table.

b. Write SQL commands to create indexes for each attribute you identified in part a.
   CREATE UNIQUE INDEX StuIndex_PK ON Student_T(StudentID);
   CREATE UNIQUE INDEX RegIndex_FK ON Registration_T(StudentID);

5. Say that you are interested in storing the numeric value 3,456,349.2334. What will be stored, with each of the following Oracle data types:
a. NUMBER(11)
3,456,349

b.NUMBER(11,1)
3,456,349.2

c. NUMBER(11,-2)
3,456,300

d. NUMBER(6)

(not accepted, exceeds precision)

e. NUMBER
3,456,349.2334

8.Consider the following normalized relations from a database in a large retail chain:
STORE (StoreID, Region, ManagerID, SquareFeet) EMPLOYEE (EmployeeID, WhereWork, EmployeeName,
EmployeeAddress)
DEPARTMENT (DepartmentID, ManagerID, SalesGoal) SCHEDULE (DepartmentID, EmployeeID, Date)
What opportunities might exist for denormalizing these relations when defining the physical records for this database? Under what circumstances would you consider creating such denormalized records?
STORE (StoreID, Region, ManagerID, SquareFeet)
EMPLOYEE (EmployeeID, WhereWork, EmployeeName, EmployeeAddress, DepartmentID, ManagerID, SalesGoal, Date)

One manager is responsible for one store.

9. 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?
TEAM(TeamID, TeamName, TeamLocation)
PLAYER(PlayerID, PlayerFirstName, PlayerLastName, PlayerDateOfBirth, PlayerSpecialtyCode, SpecialtyDescription, ContractStartTime, ContractEndTime, Salary)
LOCATION(LocationID, CityName, CityState, CityCountry, CityPopulation)
MANAGER(ManagerID, ManagerName, ManagerTeam)

Specialty is for each player in the team or for team. Location is important or not.

13.Assume that a student table in a university database had an index on StudentID (the primary key) and indexes on Major, Age, MaritalStatus, and HomeZipCode (all secondary keys). Further, assume that the university wanted a list of students majoring in MIS or computer science, over age 25, and married OR students majoring in computer engineering, single, and from the 45462 zip code. How could indexes be used so that only records that satisfy this qualification are accessed?
Use StudentID as index, and use Major, age, MaritalStatus, HomeZipCode as second key.

16. Can clustering of files occur after the files are populated with records? Why or why not?
Yes, although cluster are assigned to the table when creating the table, if the files are populated with physical records, clustering of files can occur.

17. Parallel query processing, as described in this chapter, means that the same query is run on multiple processors and that each processor accesses in parallel a different subset of the database. Another form of parallel query pro- cessing, not discussed in this chapter, would partition the query so that each part of the query runs on a different processor, but that part accesses whatever part of the data- base it needs. Most queries involve a qualification clause that selects the records of interest in the query. In general, this qualification clause is of the following form: (condition OR condition OR . . .) AND (condition OR condition OR. . .)AND. . . Given this general form, how might a query be broken apart so that each parallel processor handles a subset of the query and then combines the subsets together after each part is processed?
Those OR conditions can be a query, so just use AND to combine those subsets.