Index Tuning Exercises

Database Management and Performance Tuning

The exercises use the following tables:

- `Employee(ssnum, name, dept, manager, salary)`
- `Student(ssnum, name, course, grade, stipend, evaluation)`

1. When the Student relation was created, a nonclustering index was created on `name`. However, the following query does not use that index:

   ```
   SELECT *
   FROM Student
   WHERE name = 'Bayer'
   ```

   Please explain any possible reason and provide a tuning solution.

2. You discover that the following important query is too slow. It is a simple query, and there is a nonclustering index on salary. You have tried to update the catalog statistics, but that did not help.

   ```
   SELECT *
   FROM Employee
   WHERE salary/12 = 4000
   ```

   Please explain any possible reason and provide a tuning solution.

3. The system uses the index on salary without improving performance of the following query:

```
SELECT *
FROM Employee
WHERE salary = 48000
```

Please explain any possible reason and provide a tuning solution.

4. Your system has pages with size of 2 KB. The Student records are very long (about 1 KB) because of the length of the evaluation attribute. There is a clustering index on ssnum, but the table suffers overflow chaining when new evaluation data is added. Please explain any possible reason and provide a tuning solution.

5. Suppose there are 30 Employee records per page. Each employee belongs to one of 50 departments. Should you put a nonclustering index on dept to speed up the following query?

```
SELECT ssnum
FROM Employee
WHERE dept = 'IT'
```

Please explain the reason.

6. Suppose there are 30 Employee records per pages. However, in this case, there are 5000 departments. Should you put a nonclustering index on dept to support the same query as in *Exercise 5*? Please explain the reason.

7. Auditors take a copy of the Employee file to which they wish to apply a statistic analysis. They allow no updates but want to support the following accesses:

   a) Count all the employees that have a certain salary (frequent).

   b) Find the employees that have the maximum (or minimum) salary within a particular department (frequent).

   c) Find the employee with a certain ssnum (rare).

   Initially there is no index. Please design indexes to support the above queries.

8. Suppose that the student stipend correspond to monthly salaries, whereas the employee salaries are yearly. To find out which employees are paid as much as which students, we have two choices.

```
SELECT *
FROM Employee, Student
WHERE salary = 12*stipend
```

   or

```
SELECT *
FROM Employee, Student
WHERE salary/12 = stipend
```

   Which is better? Please explain the reason.

9. A purchasing department maintains the relation
   `Onorder(supplier, part, quantity, price)`
   The department makes the following queries to Onorder:
   a) Add a record, specifying all attributes (very frequent).
   b) Delete a record, specifying supplier and part (very frequent).
   c) Find the total quantity of a given part on order (frequent).
   d) Find the total value of the orders to a given supplier (rare).

   Please design indexes to support the above queries.

10. A table has a clustering B^+-tree index on ssnum and performs simple retrievals and updates of records based on ssnum. The performance is still not good. Please explain any possible reason and provide a tuning solution.

11. Ellis Island is a small island south of Manhattan through which flowed some 17 million immigrants to the United States between the late 1800s and the mid-1900s. Immigration workers filled in some 200 fields on each immigrants, containing information such as late name, first name, city of origin, ship taken, nationality, religion, arrival date, and so on. You are to design a database management system to allow the approximately 100 million descendants of these 17 million to retrieve the record of their ancestors.

    To identify an immigrant, the querier must know the last name of the immigrant as well as some other information. Most queriers will know the last name and either the first name or the year of arrival. Please design indexes to support such queries.

12. An airline manages 1000 flights a day. In their database, there is a table for each flight (called Flight) containing a flight identifier, a seat identifier, and a passenger name. There is also a master table (called Totals) that has the total number of passengers in each flight. Each reservation transaction updates a particular Flight table and the Totals table. They have a performance bottleneck whose symptom is high lock contention. They try to resolve this by breaking each reservation transaction into two: one that updates the appropriate Flight table and the other that updates the Totals table. That helps but not enough. Please explain any possible reasons and provide a tuning solution.