Topic 9 / Chapter 11

Database Management & CIS 1246

|  |  |
| --- | --- |
| Author: | Christopher Sigouin |
| Date: | April 24 |
| Date Due: | April 24 by 10pm |
| Assignment: | Topic 9 |
|  | Task: 1 |
|  |  |
|  |  |
| Mark: \_\_\_\_\_\_\_\_\_\_\_\_ | |

**Chapter 11 / Exercises 17 to 21 inclusive**

**Problems 17–21 are based on the following query:**

**SELECT V\_CODE, V\_NAME, V\_CONTACT, V\_STATE**

**FROM VENDOR**

**WHERE V\_STATE = 'TN' ORDER BY V\_NAME;**

**17. What indexes should you create and why? Write the SQL command to create the indexes.**

You should create an index on V\_STATE based on the following query. The index selectivity of V\_STATE would be highest compared to V\_NAME and V\_CONTACT. V\_CODE would likely be a PK so it would already be indexed.

SQL command to create the index:

CREATE INDEX V\_STATE\_NDX ON VENDOR (V\_STATE);

The keyword UNIQUE would be omitted in this case as there is the possibility of repeating values under the attribute of V\_CODE.

**18. Assume that 10,000 vendors are distributed as shown in Table P11.18. What percentage of rows will be returned by the query?**

The percentage of rows that will be returned by the query based on the distribution of vendors shown in Table 11.18 would be .01%.

113 Vendors / 10,000 Total Vendors = 0.0113%

**19. What type of I/O database operations would most likely be used to execute the query?**

Read operation to retrieve the specified attributes from the VENDOR table based on the WHERE condition.

Read operation to compare values for the ORDER BY clause.

Write operation to write the data retrieved into the data cache.

Write operation to write the processed version of the SQL statement in the SQL cache.

Write operation to output the results to the screen for the user.

**20. Using Table 11.4 as an example, create two alternative access plans.**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| PLAN | STEP | OPERATION | I/O OPERATION | I/O COST | RESULTING SET ROWS | TOTAL I/O COST |
| A | A1 | Select rows with V\_STATE = 'TN' | 10,000 | 10,000 | 113 | 10,000 |
|  | A2 | Order by V\_NAME | 113 | 113 | 113 | 10,113 |
| B | B1 | Order by V\_NAME | 10,000 | 10,000 | 10,000 | 10,000 |
|  | B2 | Select rows with V\_STATE = 'TN' | 10,000 | 10,000 | 113 | 20,000 |

**21. Assume that you have 10,000 different products stored in the PRODUCT table and that you are writing a Web-based interface to list all products with a quantity on hand (P\_QOH) that is less than or equal to the minimum quantity, P\_MIN. What optimizer hint would you use to ensure that your query returns the result set to the Web interface in the least time possible? Write the SQL code.**

In order to ensure my query returns the result set to the web interface in the least time possible, I would use INDEX (name) as a first choice assuming that P\_QOH has an index created for use. Otherwise my next choice would be ALL\_ROWS as it would be required to list “all” products so an overall optimization would be required to return the result set as soon as possible.

With MySQL optimization, I would utilize an index on P\_QOH as above if it exists. If not, I would use HIGH\_PRIORITY with the select to ensure that it gets executed as soon as possible.

(These SQL queries assumes P\_MIN is a derived attribute within the table )

SELECT /\*+ INDEX(P\_QOH\_NDX) \*/

FROM PRODUCT

WHERE P\_QOH <= P\_MIN;

OR

SELECT /\*+ ALL\_ROWS \*/\*

FROM PRODUCT

WHERE P\_QOH <= P\_MIN;

(MySQL version)

SELECT P\_QOH FROM PRODUCT USE INDEX( P\_QOH )

WHERE P\_QOH <= P\_MIN;

OR

SELECT HIGH\_PRIORITY P\_QOH FROM PRODUCT

WHERE P\_QOH <= P\_MIN;