Chapter 11/12 Problems

Problems 1 and 2 are based on the following query:

SELECT FROM WHERE ORDER BY

EMP\_LNAME, EMP\_FNAME, EMP\_AREACODE, EMP\_SEX EMPLOYEE  
EMP\_SEX = 'F' AND EMP\_AREACODE = '615' EMP\_LNAME, EMP\_FNAME;

1. What is the likely data sparsity of the EMP\_SEX column?

**The data sparsity is likely low, considering the database probably only accepts 1 of 2 values, M or F.**

1. What indexes should you create? Write the required SQL commands.

**You could create an index for the employee sex attribute, the employee area code attribute, and an index for the employee name attributes.**

**CREATE INDEX emp\_sex**

**ON EMPLOYEE(EMP\_SEX)**

**CREATE INDEX emp\_acode**

**ON EMPLOYEE(EMP\_AREACODE)**

**CREATE INDEX emp\_name**

**ON EMPLOYEE(EMP\_LNAME, EMP\_FNAME)**

**CONTINUED BELOW**

1. Using Table 11.4 as an example, create two alternative access plans. Use the following assumptions:
   1. There are 8,000 employees.
   2. There are 4,150 female employees.
   3. There are 370 employees in area code 615.
   4. There are 190 female employees in area code 615.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Plan** | **Step** | **Operation** | **I/O Ops** | **I/O Cost** | **Resulting Set Rows** | **Total Cost** |
| **A** | **A1** | **Search EMPLOYEE, SELECT “F” ROWS** | **8000 + 4150** | **12,150** | **4150** | **12,150** |
|  | **A2** | **SEARCH “F” ROWS FOR “615”, SELECT** | **4150+190** | **4340** | **190** | **12,150+4340**  **=16,490** |
| **B** | **B1** | **SEARCH EMPLOYEE FOR “615” AREA** | **8000+370** | **8370** | **370** | **8370** |
|  | **B2** | **SEARCH THROUGH “615” FOR “F”** | **370 + 190** | **560** | **190** | **8370 +190**  **=8560** |

Problems 4−6 are based on the following query:

SELECT FROM WHERE

EMP\_LNAME, EMP\_FNAME, EMP\_DOB, YEAR(EMP\_DOB) AS YEAR EMPLOYEE  
YEAR(EMP\_DOB) = 1976;

1. What is the likely data sparsity of the EMP\_DOB column?

**The data sparsity is likely very high. There is a very wide range of possible values for employee DOB.**

1. Should you create an index on EMP\_DOB? Why or why not?

**Due to the high degree of data sparsity of the DOB attribute it would be wise to create an index. The index would allow the DBMS to first pinpoint your desired DOB and then tell the system where to look for said data point. This saves the DBMS the trouble of having to search through the entire EMPLOYEE table for a single DOB value.**

1. What type of database I/O operations will likely be used by the query? (See Table 11.3.)

**Assuming an index has been made for the EMP\_DOB attribute, and index scan will be completed. The DBMS will scan the DOB index for 1976 row IDs, and the index will then point the DBMS to the location in the EMPLOYEE table.**

Problems 7−32 are based on the ER model shown in Figure P11.7. Problems 7-10 are based on the following query:

SELECT P\_CODE, P\_PRICE

FROM PRODUCT

WHERE P\_PRICE >= (SELECT AVG(P\_PRICE) FROM PRODUCT);

1. Assuming there are no table statistics, what type of optimization will the DBMS use?

**If there are no stats then the DBMS will most likely use a rule-based optimizer.**

1. What type of database I/O operations will likely be used by the query? (See Table 11.3.)

**The query could take one of two paths. It could first calculate the average price of the products and then complete a full table scan looking for items with a price greater than or equal to the average. Alternatively, the query could compute the average and then utilize a nested loop to compare the price of each product to the average.**

1. What is the likely data sparsity of the P\_PRICE column?

**This column would probably have a mild level of sparsity. A lot of products will probably be priced the same amount, but the company could also have a very wide range of products with a wide range of prices, so it depends on the company.**

1. Should you create an index? Why or why not?

**I would create an index. The P\_Price attribute appears in a WHERE statement, and the attribute has the potential to have high sparsity and high index selectivity.**

**CHAPTER 12 PROBLEMS**

1. Specify the minimum types of operations the database must support to perform the following operations. These operations include remote requests, remote transactions, distributed transactions, and distributed requests.

(Sorry for not having the problem here, the copy/paste messes up the formatting)

* 1. **This represents a remote request**
  2. **This represents a remote request**
  3. **This request must access two sites so therefore it is a distributed request**
  4. **Because the transaction must access more than one DP site, this will require a distributed request.**
  5. **This will be a distributed transaction as the request requires access to two sites**
  6. **This is a distributed request as it requires access to one local and one remote site**
  7. **This is a remote request as it requires access to one remote site**
  8. **This will be a distributed request as it requires accessing two sites becase the product table is fragmented**
  9. **This will be a remote request**
  10. **This will be a distributed request as it needs to access two different sites, one being remote**
  11. **This will be a distributed request as the product table is fragmented into two sites**

1. The following data structure and constraints exist for a magazine publishing company.
   1. The company publishes one regional magazine each in Florida (FL), South Carolina (SC), Georgia (GA), and Tennessee (TN).
   2. The company has 300,000 customers (subscribers) distributed throughout the four states listed in Part a.
   3. On the first of each month, an annual subscription INVOICE is printed and sent to each customer whose subscription is due for renewal. The INVOICE entity contains a REGION attribute to indicate the state (FL, SC, GA, TN) in which the customer resides:

CUSTOMER (CUS\_NUM, CUS\_NAME, CUS\_ADDRESS, CUS\_CITY, CUS\_STATE, CUS\_ZIP,

CUS\_SUBSDATE)

INVOICE (INV\_NUM, INV\_REGION, CUS\_NUM, INV\_DATE, INV\_TOTAL)

The company's management is aware of the problems associated with centralized management and has decided that it is time to decentralize the management of the subscriptions in its four regional subsidiaries. Each subscription site will handle its own customer and invoice data. The company's management, however, wants to have access to customer and invoice data to generate annual reports and to issue ad hoc queries, such as:

* List all current customers by region.
* List all new customers by region.
* Report all invoices by customer and by region.

Given these requirements, how must you partition the database?

**You should fragment the database horizontally, breaking it up into different fragments by state. Each state will have its own fragment for both in CUSTOMER and INVOICE table.**