National University of Computer and Emerging Sciences, Lahore Campus

Course:	Data Warehousing & Data Mining	Course Code:	CS409
Program:	BS(Computer Science)	Semester:	Fall 2016
Duration:	60 min	Total Marks:	30
Paper Date:	14-Nov-2016	Weight	12.5%
Section:	All	Page(s):	4
Exam:	Mid II	Reg. No. (Section)	---------- ( )

Instruction/Notes: **All the questions are to be solved on question paper, write your Roll no on every sheet**

Question1: (5 Points)

What are the two general categories of data stored in source operational systems? Give an example for each.

Ans:

These source systems generally store data in two ways. The type of data extraction technique you have to use depends on the nature of each of these two categories.

Current Value Most of the attributes in the source systems fall into this category. Here the stored value of an attribute represents the value of the attribute at this moment of time. The values are transient or transitory. As business transactions happen, the values change. There is no way to predict how long the present value will stay or when it will get changed next. Customer name and address, bank account balances, and outstanding amounts on individual orders are some examples of this category.

Periodic Status This category is not as common as the previous category. In this category, the value of the attribute is preserved as the status every time a change occurs. At each of these points in time, the status value is stored with reference to the time when the new value became effective. This category also includes events stored with reference to the time when each event occurred. Look at the way data about an insurance policy is usually recorded in the operational systems of an insurance company. The operational databases store the status data of the policy at each point of time when something in the policy changes. Similarly, for an insurance claim, each event, such as claim initiation, verification, appraisal, and settlement, is recorded with reference to the points in time.

Question 2: (5 Points)

Assume that 50,000 rows out of 10 million rows in Account dimension table changes on each data refresh. Which loading strategy should you follow? Explain the reasons for your selection. Also suggest some practical steps that expedite the data loading process.

Ans:

As the data changes on each refresh is low (i.e. 0.5%) which is less than 10%, so Incremental data refresh loading strategy is more efficient.

Question 3 (10 Points)

Consider the following tables and statistics which are part of a student system:

Student (RollNo, Name, gpa, DeptID, BatchID, DegreeID, …..); Attendance (RollNo, CourseCode, Semester, AttFlag, …..);

Assume student and attendance tables containing 128,000 and 2,560,000 rows respectively (*Student:Attendance* ratio is *1:20*). Each row and each index entry takes 256 bytes and 16 bytes space respectively. Data block size is 16KB and available memory size is 100 blocks. Suppose degree= 'MS' has a selectivity of 3%, batch= ('2015' or '2005') has a selectivity of (4% + 2%), and dept= ('CS or 'EE') has a selectivity of (10% + 5%).

Query:

SELECT AVG(gpa) FROM student JOIN attendance ON student.rollno=attendance.rollno
WHERE DegreeID='MS' AND (BatchID='2015' OR BatchID='2005') AND (DeptID='CS' OR DeptID='EE');

Calculate the total I/O cost (including the I/O cost to filter the condition on student table) for the above Query using hash join and block nested loop join techniques. You are supposed to filter the condition first and then join. Show all steps clearly.

Ans:

As the combine selectivity of student is 3% of (6% of (15% of (128000))) = 35 rows, so

- **Hash Join** because hash table may fit in memory which requires only one block.

HJ cost = student's filter cost + hashing cost = 2000 + (**1**+ 40,000) = **42,001**

- Block NLJ: cost = student's filter & read cost + qualifying blocks * attendance blocks = 2000 + 1 + (1 * 40,000) = **42,001 blocks**

Question 4 (3+3+4= 10 Points)

Consider the following tables and statistics which are part of a bank system:

ACCOUNT (accId, title, accType, rating, openingDate, …);

Block Size= 4 KB; Available Memory= 100 Blocks; Rows= 250,000; Row Width= 500 bytes; Index entry size (i.e. RID Width)= 8 bytes. Assume accounts with 'SAVING' accType are 4%, accounts with 'CHECKING' accType are 10%, and accounts with '1' rating are 6%.

Query: SELECT COUNT(*) FROM account WHERE (accType= 'SAVING' OR accType= 'CHECKING') AND Rating= 1

Calculate the I/O cost for the above query using

a) Composite index access (Assume a composite index exist on accType and rating columns)
b) Dynamic Bitmap index access (Assume indexes exist on accType and rating columns separately)
c) Clustered index access (Assume only clustered index exist on accType column)

Ans:
B=4K, bfr= 8, bfr_i= 512, b= 31,250 (or 30,518).

a) Composite Index:
4% of (6% of 250000)= 600
10% of (6% of 250000)= 1500
Total I/Os (Index access cost only) = 600/512 + 1500/512 = 2 + 3= **5 blocks**

b) Dynamic Bitmap Indexes:
Combined selectivity for combination ('SAVING' or 'CHECKING' and '1') = (4+10)% of (6% of 250000) = **2100 rows**
I/Os to access index for accType ('SAVING' or 'CHECKING') i.e. (4%+10%) = (10,000/512) + (25000/512) = 20+49= **69 blocks**
I/Os to access index for rating ('1') i.e. 6% = 15000/512 = **30 blocks**
Total I/Os (Index access + Base table access) = (30 + 69) + 2100 = **2199 blocks**

c) Clustered Index Access
I/Os to access base table = (10,000/8) + (25,000/8)= 1250 + 3125= **4375 blocks**
I/Os to access index for accType ('SAVING' or 'CHECKING') = (10,000/512) + (25000/512) = 20+49= **69 blocks (for dense index)**
Total I/Os (Index access + Base table access) = 69 + 4375 = **4444 blocks**
OR
I/Os to access index for accType ('SAVING' or 'CHECKING') = 1+1= **2 blocks (for sparse index)**
Total I/Os (Index access + Base table access) = 2 + 4375 = **4377 blocks**