**NAME:**

**CS 623**

**HW 7**

|  |  |
| --- | --- |
| **Exercise** | **Points** |
| Exercise 1 | 20 |
| Exercise 2 | 25 |
| Exercise 3 | 25 |
| Exercise 4 | 30 |
| Total | 100 |

**Exercise 1**

Using a counter example show that



**Exercise 2**

Suppose a database has the following schema:

*Trip(fromAddrId: INTEGER, toAddrId: INTEGER, date: DATE)*

*Address(id: INTEGER, street: STRING, townState: STRING)*

(a) Write an SQL query that returns the street of all addresses in ’New York’ that are destinations of a trip on ’5/14/25’.

(b) Translate the SQL query in (a) into the corresponding “naive” relational algebra expression.

Draw the corresponding query plan.

(c) Translate the relational algebra expression in (b) into an equivalent expression using pushing of selections and projections.

Draw the corresponding query plan.

(d) Translate the relational algebra expression in (c) into a most directly corresponding SQL query.

**Exercise 3**

A company sells merchandise through agents. Each sales agent is assigned one or more cities to cover. Consider the following schema, where the keys are underlined:

Agent(Name,City)

Location(City, ZIP)

Transaction(Date, ZIP, Item)

(*d*, *z*, *i*) ∈ Transaction means that item *i* was sold on date *d* in the area with zip code *z*.

Assume 50 buffer pages and the following statistics and indices:

* Agent: 50,000 tuples, 10 tuples/page.

Index: Unclustered hash on Name.

* Location: 1,000 tuples, 5 tuples/page; 100 cities.

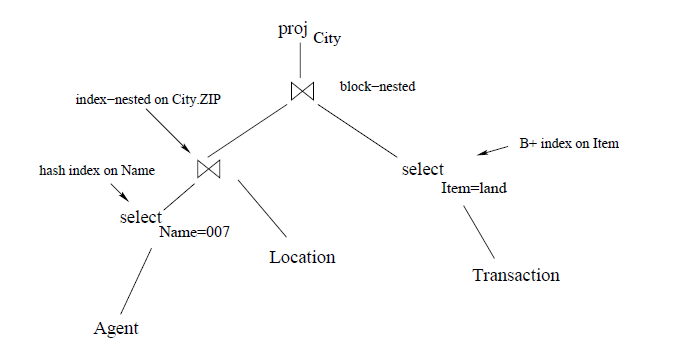
Index1: Unclustered hash index on Name.

Index2: Clustered 2-level B+ tree on City.

* Transaction: 500,000 tuples, 25 tuples/page; 10 items bought per store per day. The relation stores transactions committed over a 50 day period.

Index: 2-level clustered B+ tree on the pair of attributes StoreName, Date.

Explain why the most promising query plan is the following.



Estimate the cost of this plan.

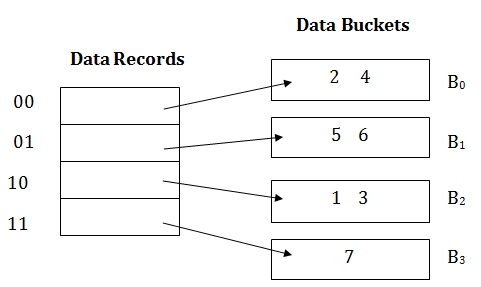
**Exercise 4**

1. Below is an example of Extendable hashing (dynamic hashing)

Key Table

|  |  |
| --- | --- |
| **Key** | **Address** |
| **1** | **11010** |
| **2** | **00000** |
| **3** | **11110** |
| **4** | **00000** |
| **5** | **01001** |
| **6** | **10101** |
| **7** | **10111** |

2 and 4's final two bits are 00. As a result, bucket B0 will receive it. Since the last two bits of both 5 and 6 are 01, place it in bucket B1. Since the final two parts of 1 and 3 add up to 10, it will be placed in bucket B2. 7 will fall into B3 since its final two bits are 11.



Insert Key 10 with hash address 10110 into the above data structure. Draw out the data buckets after hashing.

1. You have a slot directory-based page format for variable-length data. It is utilizing a maximum size (i.e., a maximum number of slots) and allocating the directory array when the page forms is one method of managing the slot directory. What are the advantages of this layout over the traditional heap file in DBMS?
2. Propose a change to the layout of this page that would enable us to sort the records (following the value of a field) without moving the records or modifying the record ids.
3. Is the following tree a clustered or unclustered tree index? Give your reasons why.

Diagram

Description automatically generated