CSE-A1150 Databases

Example 15.4.2019

Example database

- We consider a database of the webstore. The database consists of the following relations.

  `Customers(`<u>`custNo`</u>`, name, born, bonus, address, email)`
  `Products(`<u>`number`</u>`, prodName, description, price, manufID)`
  `Manufacturers(`<u>`ID`</u>`, manufName, phone)`
  `Orders(`<u>`orderNo`</u>`, deliver, status, custNo)`
  `BelongsTo(`<u>`orderNo`</u>`, `<u>`productNo`</u>`, count)`

- The following example resembles the example in Section 8.4.3 in the course text book. However, the relations and the number of the disk pages differ from the text book example.

Example how to choose indexes

- We consider the relation

  `BelongsTo(orderNo, productNo, count)`

  and calculate an estimation which indexes we should create, if
  - Two kinds of queries are performed on this relation. In Q_1 we look for product numbers which belong to an order which has a given order number. We assume that the fraction of these queries of all database operations on this relation is p_1.
    In Q_2 we look for orders where the given product number belongs to. We assume that the fraction of these queries of all database operations on this relation is p_2.
  - In addition, new tuples are inserted to the relation. The fraction of insertions of all database operations on this relation is $1 - p_1 - p_2$.
  - The relation `BelongsTo` occupies 100 disk pages, so if we need to examine the entire relation, the cost is 100 disk accesses.
  - Each order has 3 products on the average, and each product belongs to 30 orders on the average.
  - The tuples are not clustered on any attribute.

Example continues

- Because the number of rows in the results of the queries is usually much smaller than the number of the disk pages, we can assume that each result row is located in the separate disk page. (We do not calculate exact probabilities, because a rough estimate is enough for our purposes.)
- If there is an index on order number, anwsering Q_1 demands 3 disk accesses (on the average) to find the actual tuples and 1 disk access to use the index.
- If there is an index on product number, anwsering Q_2 demands 30 disk accesses (on the average) to find the actual tuples and 1 disk access to use the index.
- In the case of insertion, each index must be updated. Two disk accesses are needed for each index (one to read the disk page into main memory and another to write the updated page back to the disk) and two disk accesses are needed for the page where the tuple is inserted.

Example continues

- We calculate the average cost of one action using the various combinations of indexes:
  1. No indexes.
  2. There is an index on order number.
  3. There is an index on product number.
  4. There are both indexes.

Example continues

Action	No Index	orderNo Index	productNo Index	Both Indexes
Q_1	100	4	100	4
Q_2	100	100	31	31
I	2	4	4	6
Average	$2 + 98p_1 + 98p_2$	$4 + 96p_2$	$4 + 96p_1 + 27p_2$	$6 - 2p_1 + 25p_2$

- Denotation:
  1. Q_1 query for tuples with a given order number.
  2. Q_2 query for tuples with a given product number.
  3. I insertion to relation `BelongsTo`.
- If there is no index, the whole relation (100 disk pages) must be scanned to answer to either of the queries
- The optimal combination of indexes depends on the values of p_1 and p_2.

Explanations etc.

- The last row of the table in the previous slide has been received by multiplying the fraction of each operation by the number of disk accesses it demands, and by summing these products.
- For example, case *No index*:
  $100p_1 + 100p_2 + 2(1 - p_1 - p_2) = 2 + 98p_1 + 98p_2$
- If the query searches for tuples having a given key value and such tuple is found, it is not necessary to read the rest of the relation even if there is no index. Thus, the average number of disk accesses needed for successful key value searches without an index is only half of the total number of the disk pages of the relation.
  - In this example, the queries did not use key values. Thus the whole relation had to be scanned when there were no index.