Exercises on SQL query evaluation

Riccardo Rosati

Data management for Data Science
Sapienza Università di Roma
2016/2017

Exercise

Consider a database containing relation Movie, whose attributes are title, year, directorID, genre, duration, country, and relation Director, whose attributes are directorID, name, country, birthDate. Relation Movie contains 40.000 records, and the size of every such record is N/50, where N is the size of a memory page. Relation Director contains 5000 records, and the size of every such record is N/25. Moreover, the attribute directorID is a key for relation Director. Now consider the following query Q:

```
SELECT F.title, R.name
FROM Movie F, Director R
WHERE F.directorID = R.directorID
```

and assume that the buffer has 32 available slots for the execution of the query. Moreover, assume that the average time for accessing a page in mass memory is 5 milliseconds.

Exercise

1. Assuming that joins are executed through the Block Nested Loop algorithm, choose the physical organization of the relations Movie and Director such that the number of accesses to mass memory pages during the execution of the above query is minimum;

2. assuming that joins are executed through the Block Nested Loop algorithm, and ignoring the time needed for processing data in central memory, compute the time (in milliseconds) needed to execute the above query according to the physical organization chosen at the above point 1;

Exercise

3. assuming that joins are executed through the Index Nested Loop algorithm, choose the physical organization of the relations Movie and Director such that the number of accesses to mass memory pages during the execution of the above query is minimum;

4. assuming that joins are executed through the Index Nested Loop algorithm, and ignoring the time needed for processing data in central memory, compute the time (in milliseconds) needed to execute the above query according to the physical organization chosen at the above point 3.

Exercise

5. assuming that joins are executed through the Sort Merge Join algorithm, choose the physical organization of the relations Movie and Director such that the number of accesses to mass memory pages during the execution of the above query is minimum;

6. assuming that joins are executed through the Sort Merge Join algorithm, and ignoring the time needed for processing data in central memory, compute the time (in milliseconds) needed to execute the above query according to the physical organization chosen at the above point 3.

Solution – Point 1

The Block Nested Loop algorithm accesses the data files for the relations sequentially, therefore no particular file organization is needed. Consequently, we choose the simplest one, i.e., the heap file.

Solution – Point 2

We have the following values:

Relation Movie:

- number of records: 40.000
- size of every record = N/50
- number of pages in the data file = 40000/50 = 800

Relation Director:

- number of records = 5000
- size of every record = N/25
- number of pages in the data file = 5000/25 = 200

Consequently, we choose Director as the outer relation, and Movie as the inner relation.

Solution – Point 2

The number of pages read by the Block Nested Loop algorithm is: M + M*N/(G-2). In our case:

- M = 200
- N = 800
- G = 32

Applying the above values, we obtain 200 + 800 * 200/30 = 5533

Therefore, the time needed by the algorithm is:

5533 * 5ms = 27665ms = 27,665 seconds.

Solution – Point 3

The Index Nested Loop algorithm requires an index on the inner relation (with search key equal to the join attributes). Moreover, the execution time of this algorithm is minimized by choosing as outer relation the one with minimum number of records.

Consequently:

- we choose Director as the outer relation, and Movie as the inner relation
- we choose a heap file organization for the relation Director
- we choose a hashed file organization for the relation Movie

Solution – Point 4

The number of pages read by the Index Nested Loop algorithm (using a hashed file for the inner relation) is:

M + 1.2 * k

where k is the number of records of the outer relation (we assume that every search by equality over the hased file requires an average number of page accesses equal to 1.2, to take overflow pages into account).

In our case, M = 200 and k = 5000, thus we obtain

200 + 1.2 * 5000 = 6200

Therefore, the time needed by the algorithm is:

6200 * 5ms = 31000ms = 31 seconds.

Solution – Point 5

The Sort Merge Join algorithm is faster if the the data files for the relations are already ordered (otherwise it has to first sort the relations and then execute the join operation). Consequently, we choose the sorted file organization for both relations.

Solution – Point 6

Assuming that the join operation is sufficiently selective (i.e., the tuples of records of the inner relation that join with a given tuple of the outer relation is about 1 on average), the number of pages read by the Sort Merge Join algorithm using the sorted file organization for both relations is:

M + N

We choose Movie as the outer relation: however, notice that the choice of the outer and the inner relation does not affect the execution cost of the Sort Merge Join algorithm.

In our case, M = 200 and N = 800, thus we obtain

200 + 800 = 1000

Therefore, the execution time of the algorithm is:

1000 * 5ms = 5000ms = 5 seconds.