COMP9120 Relational Database Systems

Tutorial Week 11 Solution: Storage and Indexing

Exercise 1. Calculating space and time

Suppose we have a table Rel1(A, B, C). Each A field occupies 4 bytes, each B field occupies 12 bytes, each C field occupies 8 bytes. Rel1 contains 100,000 records. There are 100 different values for A represented in the database, 1000 different values for B, and 50,000 different values for C. Rel1 is stored with a primary B+ tree index on the composite key consisting of the pair of attributes (A,B); assume this index has 2 levels, including the root page and excluding the leaf (record) pages. Assume that in this database, each page is 4K bytes, of which 250 bytes are taken for header information. Record locations within the index use a 4-byte *rowid*. Assume that reading a page into memory takes 150 msec, and that the time needed for any query can be approximated by the time spent doing disk I/O.

1a) Calculate the space needed for the records/data of Rel1.
Hint: How much space is used by a single record? How many records per page? How many pages for the table? Then convert this back to space. [Note that each record is not split across pages, and each page has a header.]

Each page has 4096 – 250 = 3846 bytes available to store records. As each record needs 4+12+8 = 24 bytes, we can fit 3846/24 = 160 records in a page (note that we round down to an integer, as one doesn't store just part of a record in a page). Since Rel1 has 100,000 records, we need 100,000/160=625 pages to store the table; measured in bytes it is approximately 2.4 MB. (Note that our simplifying assumption here is 100% occupancy and you should always state your occupancy assumption).

1b) Calculate the time taken to perform a table scan (i.e., linear scan) through the table Rel1.
Hint: How many pages are needed to make up the space occupied by the table? How many pages will be read from disk during a table scan?

To do a table scan we must fetch each of the 625 pages of the table; measured in seconds, this takes 625*150 = 93750 ms = 93.75 s or approximately 1.5 minutes.

1c) Calculate the time taken, using the primary index, to execute the following query. Assume the selectivity of range condition on B is 10%.

```
SELECT C
FROM Rel1
WHERE A = 'AQG' AND (B BETWEEN 'WPQ' AND 'XYZ');
```

We must descend the primary index, reading 1 page at each level (that is, we read 2 pages of index). Once we have a pointer to the first data record that satisfies the condition, we need to fetch all the data records that match (because the table is clustered on the composite search key (A,B), the matching records will all be together, one after another, in the data pages). The question doesn't give us enough information to know how many matching records there are, so we need to make some guesses. We know that there are 100 different A values, so we can guess that 100,000/100 = 1000 data records have the A value 'AQG'. As the selectivity of range condition on B (i.e., the fraction

of records satisfying the range condition on B) is 10%, we can guess that 1000*10%=100 records will satisfy the matching condition, and be fetched from the data pages. Since each page stores 160 records, we probably only need to fetch 1 or perhaps 2 pages of data to get all the matching records; let's guess that we only fetch 1 data page. Overall, we will need to fetch 2 index pages, and 1 data page; that is, we read 3 pages from disk, taking 3*150=450ms.

Exercise 2. Index creation with PostgreSQL

Find which indices already exist for the tables you defined in early tutorials and assignments.
Identify an extra index, which would be useful for one of the queries you wrote (supposing that the database became much bigger, with many more records in each table!). Create this index.

Note: To check which indexes are present in your schema (plus some details), you can use the following SQL commands in PostgreSQL:

```
SELECT * FROM PG_INDEXES;
```

There are many valid choices here! For a query like "give names of authors born after 1940" an index on birthyear would be useful.

```
CREATE INDEX author_birthyear_idx on Author(birthyear)
```