COMP7640 @ HKBU: Tutorial 7
Query Processing

Wei Wang
weiw AT cse.unsw.edu.au

School of Computer Science & Engineering
University of New South Wales

November 13, 2014
Indexes can be classified according to several dimensions:
- Clustered/Non-clustered
- Primary/Secondary
- Sparse/Dense

Others:
- indexed attribute(s) is/are a candidate key or not
- tree-based or hash-based

Not all the combinations are valid.
- E.g., *sparse index must be clustered, but not every clustered index is sparse.*

Also need to realize the impact of different kind of indexes when it comes to query processing. E.g., *usually a bad idea to use a non-clustered index to answer range queries with large selectivities.*

**Advanced**

What is the essence of an index?
Indexes can be classified according to several dimensions:
- Clustered/Non-clustered
- Primary/Secondary
- Sparse/Dense

Others:
- indexed attribute(s) is/are a candidate key or not
- tree-based or hash-based

Not all the combinations are valid.
- E.g., \textit{sparse index must be clustered, but not every clustered index is sparse.}

Also need to realize the impact of different kind of indexes when it comes to query processing. E.g., \textit{usually a bad idea to use a non-clustered index to answer range queries with large selectivities.}

Advanced

What is the essence of an index? \Rightarrow Some data structure that is
- able to provide a \textit{superset} of answers for a given type of query, and/or
- \textit{smaller} than the data
External Sorting

Must know

- Algorithm:
  1. (Phase I) Run-generation phase
  2. (Phase II) Merging Phase (possibly need multiple passes)

- Buffer usage
  1. (Phase I) 0 output buffer page
  2. (Phase II) 1 output buffer page

- Cost: \( M \cdot (2 \cdot (1 + k) - 1) \), where \( k = \left\lceil \log_{(B-1)} \left\lfloor \frac{M}{B} \right\rfloor \right\rceil \)
  - 1 because the cost of writing out the final result is uniformly ignored here.
  - 2 However, when computing the cost of naive sort-merge join (i.e, not the hybrid sort-merge join), the cost should include writing out the final result, i.e., \( M \cdot 2 \cdot (1 + k) \).
Consider the relations:

\[ \text{Student}(id, name, age, course) \]
\[ \text{Subject}(code, title, description) \]

which contain 50,000 and 5,000 records respectively and are stored in files sorted on \( id \) and \( code \) respectively. Imagine that we are using simple flat indexes (i.e., one-level index). Assume that record pointers are 4-bytes and that data pages and index pages are all 4096 bytes.

1. Although you may build a dense index on any field, you cannot do so for sparse indexes. On which field(s) could you build a sparse index?

2. If the student id is a 4-byte quantity, how large would a dense index on \( \text{Student}.id \) be?

3. If the subject code is an 8-byte quantity, how large would a dense index on \( \text{Subject}.code \) be?
If the *blocking factor* for Student is 100 and for Subject is 20, and other values are as above, how large would sparse indexes on `Student.id` and `Subject.code` be?

If you had just dense indexes on `Student.id` and `Subject.code`, briefly describe efficient processing strategies for the following queries and their costs.

1. SELECT name FROM Student WHERE id=2233445
2. SELECT title FROM Subject WHERE code >= 'COMP1000' AND code <= 'COMP2000'
3. SELECT id, name FROM Student WHERE age=18
4. SELECT code, title FROM Subject WHERE title LIKE 'Comput%'

What techniques could you use to improve the performance of each of the above queries? And how would it impact the other queries?
Computing the Cost of External Sorting

You have an unsorted heap file containing 4500 records and a select query is asked that requires the file to be sorted. The DBMS uses an external merge-sort that makes efficient use of the available buffer space. Assume that: records are 48-bytes long (including a 4-byte sort key); the page size is 512-bytes; each page has 12 bytes of control information in it; 4 buffer pages are available.

1. How many sorted subfiles will there be after the initial pass of the sort algorithm? How long will each subfile be?
2. How many passes (including the initial pass considered above) will be required to sort this file?
3. What will be the total I/O cost for sorting this file (including writing out the final result)?
4. What is the largest file, in terms of the number of records, that you can sort with just 4 buffer pages in 2 passes? How would your answer change if you had 257 buffer pages?