

Lecture 11: Course overview

Rasmus Pagh



The lectures at a glance

- SR: Tree Indexes.
- RP: Hash Indexes, Index Tuning
- SR: Data storage, external sorting, lower bound.
- SR: Implementation of relational operations
- RP: Query Optimization, Query tuning
- RP: Concurrency control
- SR: Spatial databases
- SR: Temporal databases
- SR: Text indexing
- RP: Decision support, OLAP
- RP: ITU research in databases
- Invited lecture



B-trees

- Indexing is a key database technology.
- Conventional indexes (e.g., ISAM) work well when there are few updates.
- B-trees (and variants) are more flexible
 - The choice of most DBMSs
 - Range queries.
 - Deterministic/reliable.
 - Theoretically “optimal”: $O(\log_B N)$ I/Os per operation.
 - Buffering can be used to achieve fast updates, at the cost of increasing the height of the tree.



Hash indexes

- External memory hash tables generalize hash tables as you know them. We looked at different schemes:
 - Static hashing
 - Extendible hashing
 - Linear hashing
 - Newer techniques:
Buffering, two-choice hashing
- Faster than B-trees in some situations.
- Need to understand to choose!

Rule of thumb 1:

Index the most selective attribute

- Argument: Using an index on a selective attribute will help reducing the amount of data to consider.
- Example:

```
SELECT count(*) FROM R  
WHERE a>'UXS' AND b BETWEEN 100 AND 200
```
- Counterexamples:
 - Full table scan may be faster than an index
 - It may not be possible/best to apply an index.



Rule of thumb 2: Cluster the most important index of a relation

- Argument:
 - Range and multipoint queries are faster.
 - Usually sparse, uses less space.
- Counterexamples:
 - May be slower on queries “covered” by a dense index.
 - If there are many updates, the cost of maintaining the clustering may be high.
 - Clustering does not help for point queries.
 - Can cluster according to several attributes by duplicating the relation!



Rule of thumb 3:

Prefer a hash index over a B-tree if point queries are more important than range queries

- Argument:
 - Hash index uses fewer I/Os per operation than a B-tree.
 - Joins, especially, can create many point queries.
- Counterexamples:
 - If a real-time guarantee is needed, hashing can be a bad choice.
 - Might be best to have **both** a B-tree and a hash index.



Rule of thumb 4:

Balance the increased cost of updating with the decreased cost of searching

- Argument: The savings provided by an index should be bigger than the cost.
- Counterexample:
 - If updates come when the system has excess capacity, we might be willing to work harder to have indexes at the peaks.
- If buffered B-trees are used, the cost per update of maintaining an index may be rather low. Especially if binary (!) trees are used.

External sorting

- External sorting is important; DBMS may dedicate part of buffer pool for sorting!
- External merge sort minimizes disk I/O cost:
 - Pass 0: Produces sorted **runs** of size **B** (# buffer pages). Later passes: **merge** runs.
 - # of runs merged at a time depends on **B** and **block size**.
 - In practice, # of runs rarely more than 2 or 3.

Buffer management

- Disks provide cheap, non-volatile storage.
 - Random access, but cost depends on location of page on disk; important to arrange data sequentially to minimize seek and rotation delays.
- Buffer manager brings pages into RAM.
 - Page stays in RAM until released by requestor.
 - Written to disk when frame chosen for replacement (which is sometime after requestor releases the page).
 - Choice of frame to replace based on replacement policy.
 - Tries to pre-fetch several pages at a time.

Relational algebra operations

- The building blocks in DBMS query evaluation are algorithms that implement relational algebra operations.
- May be based on:
 - sorting,
 - hashing, or
 - using existing indexes
- The DBMS knows the characteristics of each approach, and attempts to use the best one in a given setting.



Query optimizer

- A virtue of relational DBMSs: queries are composed of a few basic operators; the implementation of these operators can be carefully tuned (and it is important to do this!).
- Many alternative implementation techniques for each operator; no universally superior technique for most operators.
- Must consider available alternatives for each operation in a query and choose best one based on system statistics, etc.

Database tuning

- The database tuner should
 - Be aware of the range of possibilities the DBMS has in evaluating a query.
 - Consider the possibilities for providing more efficient access paths to be chosen by the optimizer.
 - Know ways of circumventing shortcomings of query optimizers.



Query tuning

Key techniques:

- Denormalization
- Vertical/horizontal partitioning
- Aggregate maintenance
- Query rewriting (examples from SB p. 143-158, 195)
- Sometimes: Optimizer hints

Concurrency control

- DBMSs are distinguished by their design of concurrency control system
 - Pessimistic (locking based) vs optimistic
 - Granularity
- To handle concurrency control problems, an understanding of the system in use is often required.
- SQL offers several *isolation levels*, which can be explained in terms of locking implementations.

Lock tuning

- SB offers this advice on locking:
 - Use special facilities for long reads.
 - Eliminate locking when unnecessary.
 - Use the weakest isolation guarantee the application allows.
 - Change the database schema only during “quiet periods” (catalog bottleneck).
 - Think about partitioning.
 - Select the appropriate granularity of locking.
 - If possible, chop transactions into smaller pieces.



Spatial data management

- Spatial data management has many applications, including GIS, CAD/CAM, multimedia indexing.
 - Point and region data
 - Overlap/containment and nearest-neighbor queries
- Many approaches to indexing spatial data
 - R-tree approach is widely used in GIS systems
 - Other approaches include Grid Files, Quad trees, and techniques based on “space-filling” curves.
 - For high-dimensional datasets, unless data has good “contrast”, nearest-neighbor may not be well-separated

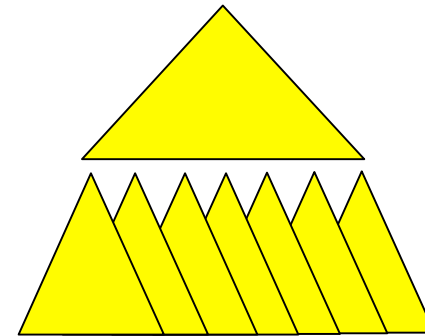


Temporal databases

- It is increasingly feasible to never delete data (i.e., keep old versions)
- \Rightarrow Demand for capability to query old data.
- Need indexing capability also for old data!
- There exists a surprisingly efficient way of doing this.

Persistent B-tree

- Persistent B-tree
 - Update current version
 - Query all versions



- Efficient implementation obtained using existence intervals

- Standard technique



- During N operations
 - $O(N/B)$ space
 - $O(\log_B N)$ update
 - $O(\log_B N + T/B)$ query



Text indexing

- Many database applications contain lots of text
- ... but the relational model is not well suited to represent the structure of text.
- Result: Text datatype that may contain long strings that have to be handled in queries.
- We look at two topics:
 - B-trees optimized for strings
 - Full-text indexing



String B-tree

String B-tree performance: [Ferragina-Gross '95]

- **Search(P)** in $O(p/B + \log_B N + \text{occ}/B)$
I/Os
- **Update(S)** takes $O(s \log_B N)$ I/Os
- **Space** is $O(N/B)$ disk pages

“Search time” is independent of the length of the strings.

Text indexes

- Indexed string matching problem
 - Word-based
 - Full-text
- Internal memory data structures
 - Suffix array
 - Suffix tree
- External memory data structures
 - Patricia trie and Pat tree
 - Short Pat array
 - String B-tree

Decision support (OLAP)

- OLAP systems are specialized databases for decision support applications.
- Idea: Read-only (or write-rarely), optimized for fast answers to queries.
- Special indexing techniques for read-only data are used (bitmap indexing).
- Precomputation of aggregates important for performance.
- Sampling-based techniques can give early join results and "on-line aggregation".

Thesis/project topics

- Five DB related suggestions on the web
(www.itu.dk/cla/tiki-index.php?page=ProjectProposals):
 - Computing joins in databases
 - Buffered indexes
 - Filtering - with applications in databases (distr.)
 - Adaptive sorting & adaptive join processing
 - Using multiple disks efficiently
- However: No supervision available at ITU in fall (parental leave).
- I will be happy to help students find an external supervisor.

