COMPSCI 105: Lecture #20 Indexing

REVIEW OF SORTING

- Physically rearranges table to be in the desired order.
- As we've seen, sorting takes a long time.
- For N records:
  - $O(N^2)$ for bad sorting techniques,
  - $O(N \times \log_2(N))$ for good sorting techniques.
- Slow, but is advantageous for searching (binary search requires list be sorted).

What if we need to see Table in several different orders?

- Option #1: Re-sort table each time a new view is needed.
  - Only one table, but…
  - …takes lots of unnecessary time.
- Option #2: Make several copies of table, each sorted on a different field.
  - Many copies means lots of disk space used, and…
  - Adding/Deleting/Changing record in one means same change must be made to all (data consistency).
- Option #3: Use Indexes.

INDEXING

- Indexes are additional hidden data structures associated with individual fields in a table.
- Indexes are maintained automatically by the database package.
- An index makes a table "look sorted" on the indexed field.
- Every field may have its own index.
- Indexes speed up both ***searches*** and ***joins***.

A Table without Indexing

PEOPLE

	ID	NAME
1	12345	Fred
2	72401	Joe
3	22222	Mary
4	54321	Sam
5	20202	Martha
6	11111	Bob
7	47904	Tom

Tables without Indexes

- In a table without any indexes, data records may appear in any random order.
- A table might have its records in a sorted order without an index.
- Searching an unindexed table for matching records requires examining every record in the table. This is a linear search, which runs in O(N) time for N records.

Same Table Indexed on ID Field

PEOPLE

	ID	NAME
1	11111	Bob
2	12345	Fred
3	20202	Martha
4	22222	Mary
5	47904	Tom
6	54321	Sam
7	72401	Joe

Table Indexed on ID Field

- The People table *apparently* is indexed on the ID field.
- People *might be* unindexed, but if it is indexed it would have to be on the ID field.
- Internally, People is still unsorted, but because there is an index on ID the database can show the table as if it was sorted by ID.

Searching Indexed Tables

- Typically, indexed fields contain unique information (no duplicates).
- Such fields can be searched with binary search in $O(\log_2(N))$ time:
  - Check list item pointed to by middle ***of index***,
  - Discard half ***of index*** not containing search term,
  - Repeat until item found or ***index*** becomes empty.

Index on ID (not seen by user)

PEOPLE

	INDEX
1	6
2	1
3	5
4	3
5	7
6	4
7	2

	ID	NAME
1	12345	Fred
2	72401	Joe
3	22222	Mary
4	54321	Sam
5	20202	Martha
6	11111	Bob
7	47904	Tom

Linear Search, Binary Search, Search with Indexing (Video).