CS222/CS122C: Principles of Data Management

UCI, Fall 2019
Notes #04

Schema versioning and File organizations

Instructor: Chen Li

Schema Versioning

- How to handle existing records when the schema is changed?
- Scheme versioning technique adds the version of the schema to each record
  - All versions of the schema are kept in the catalog
  - When the schema changes, create a new schema version
  - Records are interpreted based on it's schema version and current schema during a query

Table

Record	Version
a1, b1, c1	1
a2, b2, c2	1
a3, b3, c3, d3	2
a4, b4, d4	3

Version	Schema
1	A, B, C
2	A, B, C, D
3	A, B, D

Catalog

Example

- Create table R(A, B, C)
- Insert 1 million records
  - All records are with schema version 1
- Add attribute D
  - Create new schema version 2
- Insert 10 records
  - 10 records are with schema version 2
- Select * from R
  - records with version 1 are padded with null for field D
- Drop attribute D
  - Create new schema version 3
- Select * from R
  - field D are truncated from records with version 2

Next topic: *File Organizations*

Many alternatives exist. *Each one is ideal for some situations, but not so good in others:*

- Heap (random ordered) files: Suitable when typical access is a file scan retrieving all record or access comes through a variety of secondary indexes.
- Sorted Files: Best if records must be retrieved in some order, or only a `range' of records is needed.
- Indexes: Data structures to organize records via trees or hashing.
  - Like sorted files, they speed up searches for a subset of records, based on values in certain ("search key") fields.
  - Updates are much faster than in sorted files.

Cost Model

We will ignore CPU costs, for simplicity, so:

- **B:** The number of data pages
- **R:** Number of records per page
- **D:** (Average) time to read or write disk page
- Counting the number of page I/Os ignores gains of prefetching a sequence of pages; thus, even the real I/O cost is only roughly approximated for now.
- Average-case analysis; based on several simplistic assumptions.

** Good enough to convey the overall trends!*

Comparison of File Organizations

- Heap files (random order; insert at eof)
- Sorted files, sorted on <*age, sal*>

Operations to Compare

- Scan: Fetch all records from disk
- Equality search
- Range selection
- Insert a record
- Delete a record

Assumptions for Our Analysis

- Heap Files:
    - Equality selection on key; exactly one match.
- Sorted Files:
    - File compacted after a deletion (*vs.* a deleted bit).

Cost of Operations

	(a) Scan	(b) Equality	(c) Range	(d) Insert	(e) Delete
(1) Heap					
(2) Sorted					

** Several assumptions underlie these (rough) estimates!*

Scan: scan all the records in the file
Equality: like "id = 123", suppose only one record satisfy the equality
Range: like "5 < id < 100"
Insert:
Delete:

Assume # of pages in the file: B

Cost of Operations (disk IOs)

assume no cost to locate page with enough free space

id=5

5<id<100

	(a) Scan	(b) Equality /Search	(c) Range	(d) Insert	(e) Delete
(1) Heap	BD	0.5BD record distributed uniformly	BD	2D read page and flush page back	Search +D
(2) Sorted T(id, name, sal,...) Sorted only based on one attribute	BD	$D\log_2 B$	$D(\log_2 B +$ # pgs with match recs)	Search + BD B/2*d + B/2*D	Search +BD

locate id=5

Get benefit only for the sorted attribute

scan from id=5 to id=100

shift all the following page

** Several assumptions underlie these (rough) estimates!*

B: total num of records
D: cost of operation for one disk IO

Delete of heap file:
1) given a RID -> 2D(read and write)
2) given a value -> search + D(write back)

Average shift B/2 pages
-> read to memory then write back to disk