



Week 9

Database Physical Design/Heaps

Ts. Chng Chern Wei



Database Programming

- Week 9
- Database Physical Design/Heaps

Introduction

- Need for Physical Design
- Use of Access Methods
- A Cost Model for Physical Access
- The Heap Access Method

Database Storage

- Database organised into 1+ files
 - each file consists of 1+ records
 - each record consists of 1+ fields.

Typically 1 record = 1 tuple; 1 field = 1 attribute.

When user requests a tuple (row), the DBMS maps **logical** record into a **physical** record and retrieves the physical record into the buffers in primary storage (main memory).

A **block** or **page**, which contains 1+ Physical records, is the unit of transfer between disk and primary storage and back.

Database Design

- After producing logical design with elegant maintainable structures:
- Need to do physical design to **make it run fast.**
- **Performance** is often more important in database applications than in more general information system design:
 - Emphasis on number of transactions per second (**throughput**)

Aims of Physical Design

- Fast **retrieval** – usually taken as ≤ 5 disk accesses
 - Since disk access is very long compared to other access times, number of disk accesses is often used as a key indicator of performance
- Fast **placement** – within 5 disk accesses
 - Insertion of data, may be in middle of file not at end
 - Deleting data, actual removal or tombstone
 - Updating data, including primary key and other data

Retrieval/Placement

- Distinguish between actions involving primary and secondary keys
- Primary key
 - May be single or multiple attributes
 - Only one per table
- Secondary keys
 - Again may be single or multiple attributes
 - Many per table
 - Include attributes other than the primary key

Access Method



- An access method is the software responsible for storage and retrieval of data on disk
- Handles page I/O between disk and main memory
 - Pages may be blocked/grouped so that many are retrieved in as less number of disk access as possible.
- Many different access methods exist
 - Each has a particular technique for relating a primary key value to a page number

Processing of Data

- All processing by software is done in **main memory**
- Blocks of pages are moved
 - from disk to main memory for retrieval
 - from main memory to disk for storage
- Access methods drive the retrieval/storage process
- The order in which records are stored and accessed in the file is dependent on the file organisation, *i.e. the physical arrangement of data in a file in secondary storage.*

A Cost Model

Identify cost of each retrieval/storage operation

D = Access time for disk to read/write page

= seek time (time to move head to required cylinder)

+ rotational delay (time to rotate disk once the head is over the required track)

+ transfer time (time to transfer data from disk to main memory)

Typical value for $D = 15$ milliseconds (msecs) or
 15×10^{-3} secs

Other Times

C = average time to process a record in main memory = 100 nanoseconds (nsecs) = 10^{-7} (i.e. $100 * 10^{-9}$) secs.

R = number of records/page

B = number of pages in file

Note that $D > C$ by roughly 10^5 times

Access Method 1: the Heap

Heap files are the simplest type of file organisation. Records (tuples) are held on file:

- in the same order as they are inserted, i.e., in no particular order
- with no indexing
- that is in a 'heap' – Unix default file type

Access Method 1: the Heap

Heap strategy:

- Insertions usually at end, therefore very efficient
- Searching is exhaustive from start to finish until required record found (i.e., a linear search)
- Deletions are marked by tombstones, i.e., the record is marked as deleted and the space of deleted records is not reused.

Performance gradually worsens as more deletions occur. Heap files should therefore be periodically reorganised by the DBA to reclaim unused space of deleted records.

The Heap – Cost Model (1)

- Cost of complete scan: $B(D+RC)$
 - For each page, 1 disk access D and process of R records taking C each.
 - If $R=1000$, $B=1000$ (file contains 1,000,000 records)
 - Then cost = $1000(0.015+(1000*10^{-7})) = 1000(0.0150+0.0001)$
 $= 1000(0.0151) = 15.1$ secs

B = number of pages in file

C = average time to process a record in main memory = 100 nanoseconds
(nsecs) (10^{-7} secs)

D = Access time to disk to read/write page (0.0150 secs/ $15 * 10^{-3}$)

R = number of records/page

The Heap – Cost Model (1)

- Cost for finding particular record: $B(D+RC)/2$ (scan half file on average) = 7.55 secs
- Cost for finding a range of records, e.g., student names beginning with 'Sm': $B(D+RC)$ (must search whole file) = 15.1 secs

B = number of pages in file

C = average time to process a record in main memory = 100 nanoseconds (nsecs) (10^{-7} secs)

D = Access time to disk to read/write page ($0.0150 \text{ secs}/15 * 10^{-3}$)

R = number of records/page

The Heap – Cost Model (2)

- Insertion: $2D + C$
 - Fetch last page (D), process record (C), write last page back again (D).
 - Assumes:
 - all insertions at end
 - system can fetch last page in one disk access

$$\text{Cost} = (2 \times 0.015) + 10^{-7} = 0.0300001 \approx 0.030 \text{ secs}$$

B = number of pages in file

C = average time to process a record in main memory = 100 nanoseconds (nsecs)
(10^{-7} secs)

D = Access time to disk to read/write page ($0.0150 \text{ secs} / 15 \times 10^{-3}$)

R = number of records/page

The Heap – Cost Model (3)

- Deletions: $B(D+RC)/2 + C + D$
 - Find particular record (scan half file - $B(D+RC)/2$), process record on page (C), write page back (D).
 - Record will be flagged as deleted (tombstone)
 - Record will still occupy space
 - If reclaim space need potentially to read/write many more pages

$$\text{Cost} = 7.55 + 10^{-7} + 0.015 = 7.5650001 \approx 7.565 \text{ secs}$$



B = number of pages in file

C = average time to process a record in main memory = 100 nanoseconds (nsecs) (10^{-7} secs)

D = Access time to disk to read/write page ($0.0150 \text{ secs}/15 * 10^{-3}$)

R = number of records/page

Advantages of Heaps

Advantages:



- Cost effective where many records processed in a single scan (can process 1,000,000 records in 15.1 secs).
- Simple access method to write and maintain

Heap files are good storage structures for:



- Bulk-loading of data. Records inserted at the end of the heap, so no overhead of calculating what page the record should go on.
- When relation is only a few pages long. As time to locate any record is short, even if entire relation has to be searched serially.
- When every tuple in the relation has to be retrieved (in any order) every time the relation is accessed.

Disadvantages of Heaps

- Heaps not suitable for use when only selected tuples from a relation are to be accessed. 
- Very expensive for searching for single records in large files (1 record could take 15.1 secs to find) 
- Expensive for operations such as sorting due to no inherent order

Note: If heap is chosen as the initial file organisation for bulk loading etc, it may be more efficient to restructure the file after the insertions have been completed.

Disadvantages of Heaps

- Heaps not suitable for use when only selected tuples from a relation are to be accessed.
- Very expensive for searching for single records in large files (1 record could take 15.1 secs to find)
- Expensive for operations such as sorting due to no inherent order

Note: If heap is chosen as the initial file organisation for bulk loading etc, it may be more efficient to restructure the file after the insertions have been completed.

Q&A?

The End