Disks, Files, and Indexes

Yanlei Diao

Slides Courtesy of R. Ramakrishnan and J. Gehrke
Outline

- Data Storage: Disks, Disk Space Manager
- Disk-Resident Data Structures (Access Methods)
  - Files of records
  - Indexes
    - Tree index: B+ tree (for an ordered domain)
    - Tree index: R-tree (for an unordered domain)
    - Hash indexes
Fun Questions:

- How can we store our data for 1 hundred years?
- How can we store our data (papers, pictures, videos) for 1 million years?

DNA as a storage medium: “fantastically dense, stable, energy efficient, and proven to work over 3.5 billion years.”

- George Church. “Writing the Book in DNA”. Harvard Medical School Genetics.
(Rather Brief)

History of Data Storage

- Mechanical Punch Card: '90
- 8" Floppy: '81
- Hard Disk Drive (HDD): '56
- DRAM: '66
- Solid State Drive (SSD): '76
- DVD: '95
- Software Defined Storage (SDS): '00
- USB Flash Drive (Microdrive): '99
- Magnetic Tape: '28
- Virtualization: '68
- CD-Rom: '84
- Digital Linear Tape (DLT): '93
- Blu-Ray: '03
- Cloud: '99

Timeline:
- 1890's
- 1920's
- 1950's
- 1960's
- 1970's
- 1980's
- 1990's
- 2000's
- Tiered Storage: '74
Computer Architecture 101

- Registers
- Caches
- Main Memory
- Disk Storage

Tape

5 ns
10 ns
100 ns
5 ms

Speed
Memory Hierarchy

- **Main Memory** (RAM)
  - Random access, fast, usually volatile
  - Main memory for currently used data

- **Magnetic Disk**
  - Random access, relatively slow, nonvolatile
  - Persistent storage for all data in the database.

- **Tape**
  - Sequential scan (read the entire tape to access the last byte), nonvolatile
  - For archiving older versions of the data.
Disks and DBMS Design

- A database is stored on disks. This has major implications on DBMS design!
  - **READ:** transfer data from disk to RAM for data processing.
  - **WRITE:** transfer data (new/modified) from RAM to disk for persistent storage.
  - Both are high-cost operations relative to in-memory operations, so must be planned carefully!
Basics of Disks

- Unit of storage and retrieval: *disk block* or *page*.
  - A contiguous sequence of bytes.
  - Size is a DBMS parameter, 4KB or 8KB.

- Unlike RAM, *time to retrieve a page* varies!
  - It depends upon the location on disk.
  - Relative placement of pages on disk has major impact on DBMS performance!
Components of a Disk

- **Spindle, Platters**
  E.g. spin at 7200 or 15,000 rpm (revolutions per minute)

- **Disk heads, Arm assembly**
  - Arm assembly moves in or out, e.g., 1-10ms
  - **Only** one head reads/writes at any one time.
Data on Disk

- A platter consists of *tracks*.
  - single-sided platters
  - double-sided platters

- Tracks under heads make a *cylinder* (imaginary!)

- Each track is divided into *sectors* (whose size is fixed).

- *Block (page) size* is a multiple of *sector size* (DBMS parameter).
Accessing a Disk Page

- Time to access (read/write) a disk block:
  1. *seek time* (moving arms to position a disk head on a track)
  2. *rotational delay* (waiting for a block to rotate under the head)
  3. *transfer time* (actually moving data to/from disk surface)

- Seek time and rotational delay dominate.
  - *seek time*: 1 to 10 msec
  - *rotational delay*: 0 to 10 msec
  - *transfer rate*: < 1msec/page, or 10’ s-100’ s megabytes/sec
    (sequential IO speed)

- Key to lower I/O cost: reduce seek/rotation delays!
  Hardware vs. software solutions?
Arranging Pages on Disk

- Software solution uses the ‘next’ block concept:
  - blocks on the same track, followed by
  - blocks on the same cylinder, followed by
  - blocks on an adjacent cylinder

- Pages in a *file* should be arranged sequentially on disk (by ‘next’), to minimize seek and rotational delay.
  - Scan of the file is a *sequential scan*. 
Disk Space Manager

- Lowest layer of DBMS managing space on disk. Higher levels call it to:
  - allocate/de-allocate a page
  - allocate/de-allocate a sequence of pages
  - read/write a page

- Requests for a sequence of pages are satisfied by *allocating the pages sequentially* on disk!
  - Higher levels don’t need to know any details.
**DBMS Architecture**

- Query Parser
- Query Rewriter
- Query Optimizer
- Query Executor

- Access Methods
- Buffer Manager
- Log Manager

- Lock Manager

- Disk Space Manager

- DB
Outline

- Data Storage: Disks, Disk Space Manager
- Disk-Resident Data Structures (Access Methods)
  - Files of records
  - Indexes
    - Tree index: B+ tree (for an ordered domain)
    - Tree index: R-tree (for an unordered domain)
    - Hash indexes
File of Records

- Abstraction of disk-resident data for query processing: a file of records residing on multiple pages
  - A number of fields are organized in a record
  - A collection of records are organized in a page
  - A collection of pages are organized in a file
Record Format: *Fixed Length*

- **Record type:** the *number of fields and type of each field* (defined in the schema), stored in *system catalog*.

- **Fixed length record:** (1) the number of fields is fixed, (2) each field has a fixed length.

- **Store fields consecutively in a record.** How do we find *i*'th field of the record?

```plaintext
Record Format: Fixed Length

F1  F2  F3  F4

L1  L2  L3  L4

Base address (B)  Address = B + L1 + L2

- Record type: the *number of fields and type of each field* (defined in the schema), stored in *system catalog*.
- Fixed length record: (1) the number of fields is fixed, (2) each field has a fixed length.
- Store fields consecutively in a record. How do we find *i*'th field of the record?
```
Record Format: **Variable Length**

- **Variable length record**: (1) number of fields is fixed, (2) some fields are of variable length

2nd choice offers direct access to $i^{th}$ field; but small directory overhead.
Page Format

- How to store a collection of records on a page?

- View a page as a collection of slots, one for each record.

- A record is identified by $rid = <\text{page id, slot #}>$
  - Record ids (rids) are used in indexes. More on this later…
Page Format: Fixed Length Records

If we move records for free space management, we may change rids! Unacceptable for performance.
Page Format: **Variable Length Records**

- **Compaction Alg:** get all slots whose offset is not -1, sort by start address, move their records up in sorted order. No change of rids!
Files of Records

- **File**: a collection of pages, each containing a collection of records. Typically, one file for each relation.
  - **Updates**: insert/delete/modify records
  - **Index scan**: read a record given a record id – more later
  - **Sequential scan**: scan all records (possibly with some conditions on the records to be retrieved)

- Files in DBMS versus Files in OS?
Heap (Unordered) Files

- **Heap file**: contains records in no particular order.
- As a file grows and shrinks, disk pages are allocated and de-allocated.
- To support record-level operations, we must:
  - keep track of the *pages* in a file
  - keep track of *free space* on pages
  - keep track of the *records* on a page
A directory entry per page, containing a pointer to the page, # free bytes on the page.

The directory is a collection of pages; a linked list is one implementation.
  - Much smaller than the linked list of all data pages.

Search for space for insertion: fewer I/Os.