Disks, Files, and Indexes

Yanlei Diao

Slides Courtesy of R. Ramakrishnan and J. Gehrke
Data Storage: Fun Questions

- How can we store our data for 1 hundred years?
- How can we store our data 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
UNIVAC introduces the "UNISERVO" tape drive for the UNIVAC I computer. It was the first tape storage device for a commercial computer, and the relative low cost, portability and unlimited offline capacity of magnetic tape made it very popular. UNIVAC tapes were ½" wide, 0.0015" thick, up to 1,500' long, and made of phosphor-bronze with a metallic coating. Weighing about three pounds, each reel could hold 1,440,000 decimal digits and could be read at 100 inches/sec.
The era of magnetic disk storage dawns with IBM’s shipment of a RAMAC 305 computer system to Zellerbach Paper in San Francisco. The computer was based on the new technology of the hard disk drive — the world’s first. The
Intel 1103 memory chip

DEC VAX memory board with Intel 1103 memory chips

The introduction of the 1 KB Intel 1103 memory chip marks the beginning of the end for magnetic core memory and ushers in the era of dynamic random-access memory (DRAM) integrated circuits for main memory in computers. The 1103 sold slowly at first, but this likely helped the development team at Intel, which was still ironing out details about the chip's specifications after its initial release. However, at a price of 1¢ per bit and with a speed compatible with existing logic circuits, sales skyrocketed after several design revisions.
Computer Architecture 101

Diagram showing the hierarchy of computer memory:
- **Registers**
- **Caches**
- **Main Memory**
- **Disk Storage**
- **Tape**
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.
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, or 5ms on average
  - 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 ms, or 5ms on average
  - *rotational delay*: 0 to 10 ms, or 5ms on average
  - *transfer rate*: < 1ms/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
- Lock Manager
- Access Methods
- Buffer Manager
- Log 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?

```
F1   F2   F3   F4
L1   L2   L3   L4
```

Base address (B)  Address = B+L1+L2
Record Format: **Variable Length**

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

Fields Delimited by Special Symbols

Array of Field Offsets

2\textsuperscript{nd} 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 \( rid = <\text{page id}, \text{slot #}> \)
  - Record ids (rids) are used in indexes. More on this later…
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!

Search?
Deletion?
Insertion?

Pointer to start of free space

- **SLOT DIRECTORY**
  - `N`... `2`... `1`... #slots
  - `N`... `100`... `100,16`... `120,44`... `164`... `N`
  - `Rid = (10,N)`
  - `Rid = (10,2)`
  - `Rid = (10,1)`

Page 10
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!