ĐẠI HỌC
BÁCH KHOA

25 YEARS ANNIVERSARY
SOICT

HA NOI UNIVERSITY OF SCIENCE AND TECHNOLOGY
SCHOOL OF INFORMATION AND COMMUNICATION TECHNOLOGY

HA NOI UNIVERSITY OF SCIENCE AND TECHNOLOGY
SCHOOL OF INFORMATION AND COMMUNICATION TECHNOLOGY

Lesson 6
Storage and Index

Outline

- Overview of database storage structures
- Physical database files
- Database index

Objectives

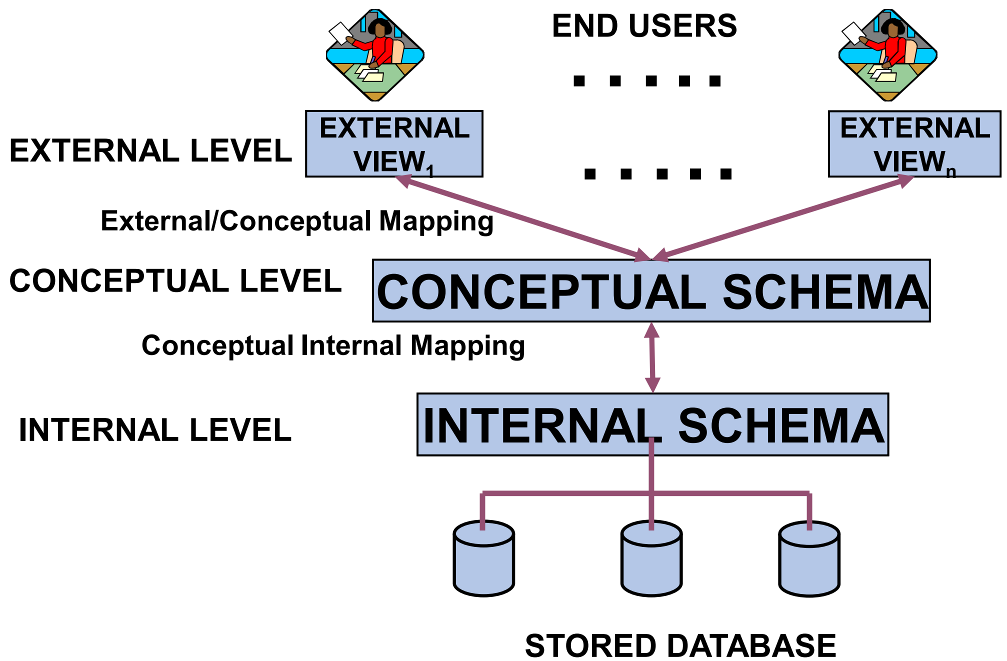
- Upon completion of this lesson, students will be able to:
  - Understand the physical database files
  - Understand the role of database indexes

Keywords

Heap file	Files of Unordered Records
Ordered file	Physically order the records of a file on disk based on the values of one of their fields (key field)
Index	A data structure that improves the speed of data retrieval operations
B-tree	A self-balancing tree data structure that keeps data sorted

1. Overview of database storage structures

3-tier Schema Model (ANSI-SPARC Architecture)



1. Overview of database storage structures

How does Mariadb store data

```
MariaDB [(none)]> SHOW VARIABLES LIKE 'datadir';
+---------------+-----------------+
| Variable_name | Value           |
+---------------+-----------------+
| datadir       | /var/lib/mysql/ |
+---------------+-----------------+
```

```
MariaDB [student_management]> show tables;
+-------------------------------+
| Tables_in_student_management |
+-------------------------------+
| class                         |
| enrolled                      |
| faculty                       |
| student                       |
+-------------------------------+
```

```
:/var/lib/mysql/student_management# ls -la

ql mysql   4096 Mar 12 02:05 .
ql mysql   4096 May  5 06:06 ..
ql mysql   1547 Mar 12 02:05 class.frm
ql mysql 114688 Mar 12 02:21 class.ibd
ql mysql     65 Mar 12 01:59 db.opt
ql mysql   1466 Mar 12 02:03 enrolled.frm
ql mysql 114688 Mar 12 02:18 enrolled.ibd
ql mysql   1005 Mar 12 02:04 faculty.frm
ql mysql  98304 Mar 12 02:16 faculty.ibd
ql mysql   1101 Mar 12 02:00 student.frm
ql mysql  98304 Mar 12 02:23 student.ibd
```

the .frm table file stores the table's format
the .ibd file stores the table's data

1. Overview of database storage structures

How does Mariadb store data

- the .frm file stores the table's format

```
MariaDB [student_management]> describe student;
+-------+-------------+------+-----+---------+-------+
| Field | Type        | Null | Key | Default | Extra |
+-------+-------------+------+-----+---------+-------+
| snum  | int(11)     | NO   | PRI | NULL    |       |
| sname | varchar(40) | YES  |     | NULL    |       |
| major | varchar(30) | YES  |     | NULL    |       |
| level | varchar(10) | YES  |     | NULL    |       |
| age   | int(11)     | YES  |     | NULL    |       |
+-------+-------------+------+-----+---------+-------+
```

```
root@285e07e9458f:/var/lib/mysql/student_management# cat student.frm
?

VM?\!   ?s?$??%?艂 B??
??PRIMARY??InnoDB??f\P
                     (/?
N?
?snum?sname?major?level?age?root@285e07e9458f:/var/lib/mysql/student
```

1. Overview of database storage structures

How does Mariadb store data

- the .ibd file stores the table's data

```
MariaDB [student_management]> select * from student;
+------+-------------------+---------+-------+------+
| snum | sname             | major   | level | age  |
+------+-------------------+---------+-------+------+
|    1 | Nguyen Van A      | CS      | JR    |   18 |
|    2 | Nguyen Viet Cuong | History | JR    |   19 |
|    3 | Nguyen Hong Ngoc  | CS      | JR    |   19 |
|    4 | Mark Juke         | History | JR    |   20 |
|    5 | Elon Mulk         | CS      | JR    |   20 |
|    6 | Donal Trump       | CS      | JR    |   20 |
|    7 | Obama             | CS      | JR    |   20 |
|    8 | Tan Dung          | History | SR    |   30 |
+------+-------------------+---------+-------+------+
```

```
root@285e07e9458f:/var/lib/mysql/student_management# cat student.ibd
???]&!??????????????????????????&&??????????????????????????]&??Y?&??Y?&???j?&???
?????????????????????????????i??????????????????????????????????????????????????????
????????????????????????????????????????????????????????????????????????????????????
????????????i???????????????????????????????????????????????????????????????????????
????????????????????????????????????????????????????????????j?&??Q?????????'??E?
9infimum
        supremum
                .?8?WNguyen Van ACSJR?8?:?cNguyen Viet CuongHistoryJR? 2?@??Nguyen H
ong NgocCSJR?   (0?1?iMark JukeHistoryJR?        0+?O??Elon MulkCSJR?
                                                                    8-?Q?kDonal Trum
pCSJR?@'?W??ObamaCSJRH?????Tan DungHistorySR?pc??Q?'??root@285e07e9458f:/var/lib/mys
```

2. Physical database files

Motivation

Magnetic disks as data storage

Primary file organizations

2.1. Motivation

- Databases typically store large amounts of data persistently on disks:
  - Databases are too large to fit entirely in main memory.
  - Disk - nonvolatile storage vs. Main memory - volatile storage
  - The cost of storage per unit is much cheaper

2.2. Magnetic disks as data storage

- A disk is a random access addressable device.
- Transfer of data between main memory and disk takes place in units of disk blocks.
- Typical disk block sizes: 4KB – 8KB.
- Disk I/O (read/write from disk to main memory) overhead is the key factor of database performance optimization.

2.2.1. Physical database design

- The process of physical database design involves choosing the particular data organization techniques that best suit the given application requirements (on SELECT, INSERT, UPDATE, DELETE).
- The data stored on disk is organized as files of records:
  - Primary file organizations: determine how the file records are physically placed on the disk, and hence how the records can be accessed.
  - Secondary organization or auxiliary access structure allows efficient access to file records based on alternate fields.

2.2.2. Placing File Records on Disk

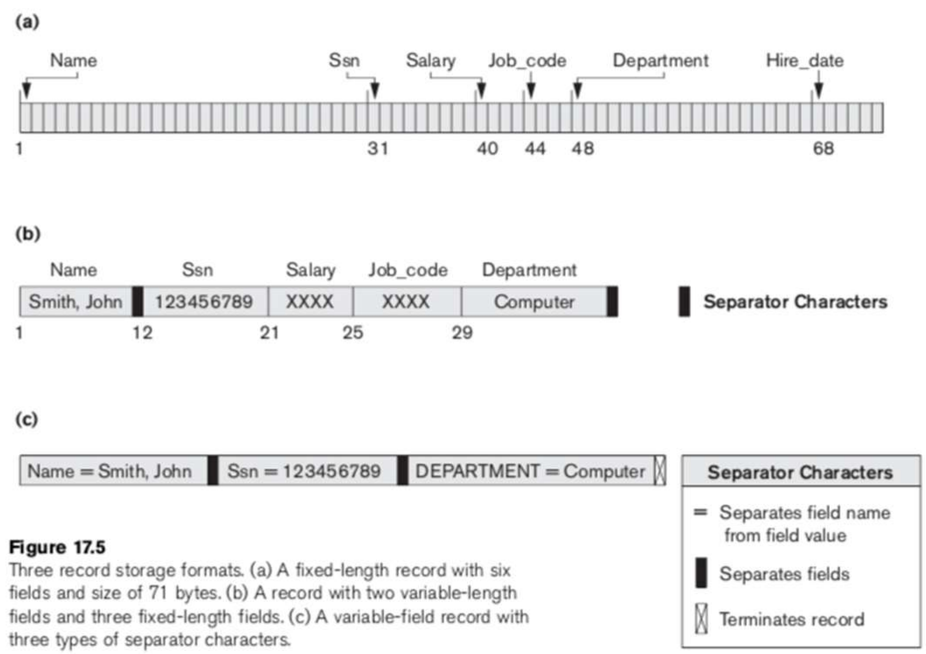


Figure 17.5
Three record storage formats. (a) A fixed-length record with six fields and size of 71 bytes. (b) A record with two variable-length fields and three fixed-length fields. (c) A variable-field record with three types of separator characters.

© Elmasri, Ramez. *Fundamentals of database systems*. Pearson Education India, 2008

2.3. Primary file organizations

- Files of Unordered Records (Heap Files)
- Files of Ordered Records (Sorted Files)
- Hashing Techniques

2.3. Primary file organizations

- Files of Unordered Records (Heap Files)
  - Records are placed in the file in the order in which they are inserted
  - INSERT: Inserting a new record is very efficient
    - New records are inserted at the end of the file
  - UPDATE/SELECT: Searching for a record on any search condition is not efficient – linear search
  - DELETE: leaves unused space in the disk block
    - require periodic reorganization

2.3. Primary file organizations

- Files of Ordered Records (Sorted Files)
  - Physically order the records of a file on disk based on the values of one of their fields (key field)
  - SELECT: binary search (very fast)
  - INSERT/DELETE/UPDATE: more expensive

2.3. Primary file organizations

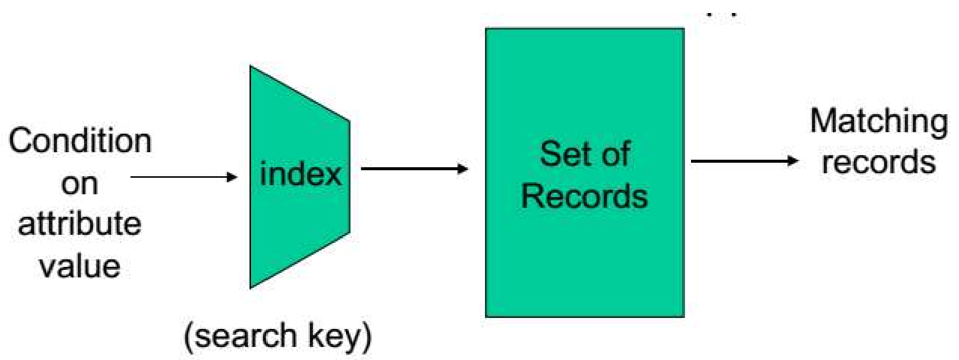
- Hash files
  - The address of the disk block in which the record is stored is the result of applying a hash function to the value of a particular field (hash field) of the record.
  - Very fast access to records for search on equality condition on the hash field.

3. Database indexes

1. What is database index?
2. Index data structures
3. B+tree
4. Spare vs. Dense index
5. Clustered vs. Non-clustered index
6. Index creation in SQL

3.1. What is database index?

- Auxiliary access structure (commonly index) allows efficient access to file records based on alternate fields

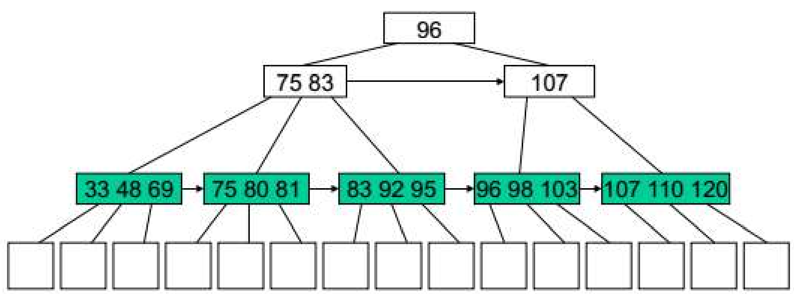


3.2. Index data structures

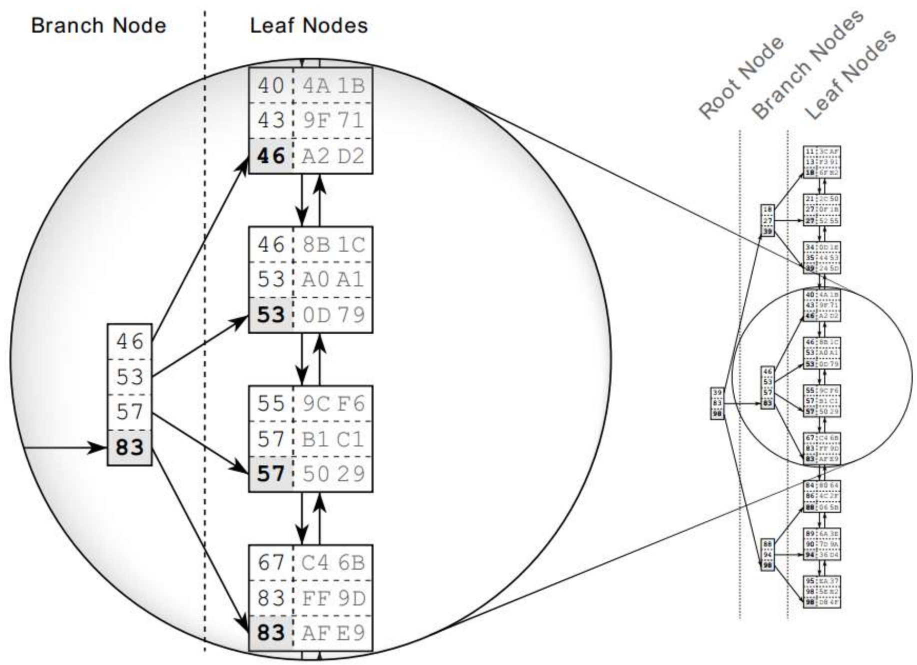
- Indexes can be implemented with different data structures.
  - B+-tree index
  - hash index
  - bitmap index (briefly)
  - dynamic hash indexes: number of buckets modified dynamically
  - R-tree: index for special data (points, lines, shapes)
  - quadtree: recursively partition a 2D plane into four quadrants
  - octree: quadtree version for three dimensional data
  - main memory indexes: T-tree, binary search tree

3.3. B+Tree

- Balanced tree of key-pointer pairs
- Keys are sorted by value
- Nodes are at least half full
- Access records for key: traverse tree from root to leaf



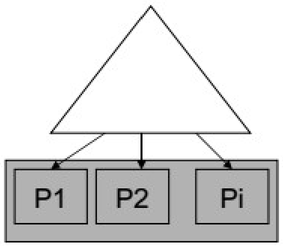
3.3.1. Example: B+ tree

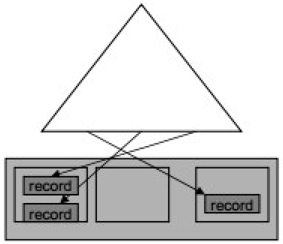


© Gulutzan, Peter, and Trudy Pelzer. *SQL Performance Tuning*. Addison-Wesley Professional, 2003.

3.4. Spare vs. Dense index

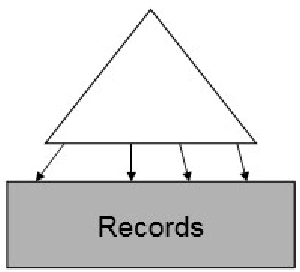
- Sparse index
  - pointers to disk pages
  - at most one pointer per disk page
  - usually much less pointers than records
- Dense index
  - pointers to individual records
  - one key per record
  - usually more keys than sparse index optimization: store repeating keys only once, followed by pointers

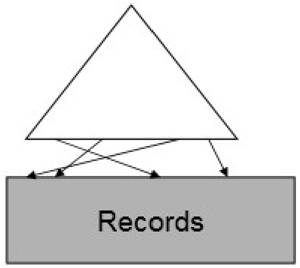




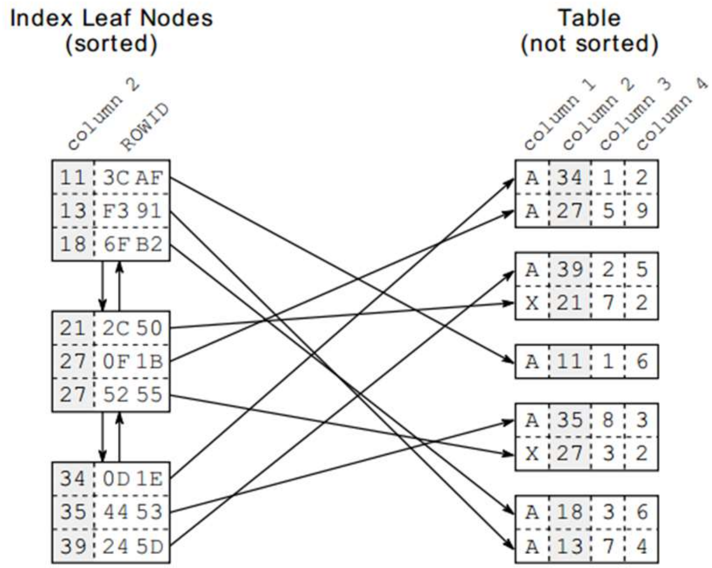
3.5. Clustered vs. Non-Clustered

- Clustered index on attribute X
  - This index controls the placement of records on disk
  - only one clustering index per table
  - dense or sparse
- Non-clustered index on attribute X
  - no constraint on table organization
  - Can have more than one index per table
  - always dense





3.5.1. Example: Non-clustered index



© Gulutzan, Peter, and Trudy Pelzer. *SQL Performance Tuning*. Addison-Wesley Professional, 2003.

3.6. Creating Index

- CREATE [UNIQUE|FULLTEXT|SPATIAL] INDEX index_name [index_type] ON tbl_name (index_col_name,...) [index_option] [algorithm_option | lock_option] ...
- index_type: USING {BTREE | HASH}

Remark

- Databases typically store data persistently on disks
  - Files of unordered records (Heap files)
  - Files of ordered records (Sorted files)
  - Hash files
- Index allows efficient access to file records based on “indexed” fields

Quiz 1.

<table>
<tr><td rowspan="2">Quiz Number</td><td rowspan="2">1</td><td rowspan="2">Quiz Type</td><td>OX</td><td>Example Select</td></tr>
<tr><td></td><td></td></tr>
<tr><td>Question</td><td colspan="4">Does heap files support INSERT query efficiently?</td></tr>
<tr><td>Example</td><td colspan="4">A. Yes
B. No</td></tr>
<tr><td>Answer</td><td colspan="4">A</td></tr>
<tr><td>Feedback</td><td colspan="4">New records are appended to the end of the head file</td></tr>
</table>

Quiz 2.

Quiz Number	2	Quiz Type	OX	Example Select
Question	Are ordered files better for heavy Insert operation?			
Example	A. Yes B. No			
Answer	B			
Feedback	Insertion to ordered files requires reorganizing w.r.t. new records			

Summary

- Overview of database storage structures
  - 3-tier Schema Model (ANSI-SPARC Architecture)
  - How Mariadb stores data
- Physical database file structures
  - Motivation
  - Magnetic disks as data storage
  - Primary file organizations
- Database index
  - What is database indexes?
  - Index data structures
  - B+tree
  - Spare vs. Dense index
  - Clustered vs. Non-clustered index
  - Index creation in SQL

Next lesson: Query processing

- Hector Garcia-Molina, Jeffrey D. Ullman, Jennifer Widom. Database Systems: The Complete Book. Pearson Prentice Hall. the 2nd edition. 2008: Chapter 7
- Nguyen Kim Anh, Nguyên lý các hệ cơ sở dữ liệu, NXB Giáo dục. 2004: Chương 7