

CPT201

overview

- **Assessment**
 - Three coursework + final exam
- **Coursework**
 - Homework 1 (15%): indexing techniques
 - Homework 2 (15%): in the form of labs exercises (e.g. JDBC, indexing, Web storage, NoSQL, etc), marking to be done in labs.
 - Coursework 3 (20%): in the form of in-class test, (part 1 and part 2, corresponding to Homework 1 and 2, respectively)
- **Final exam (50%)**
 - Part A: short answer questions
 - 20 questions, 2 marks each, in total 40 marks
 - Part B: Problem-Solving and Quantitative Questions
 - 2 'large' questions, 30 marks each, in total 60 marks

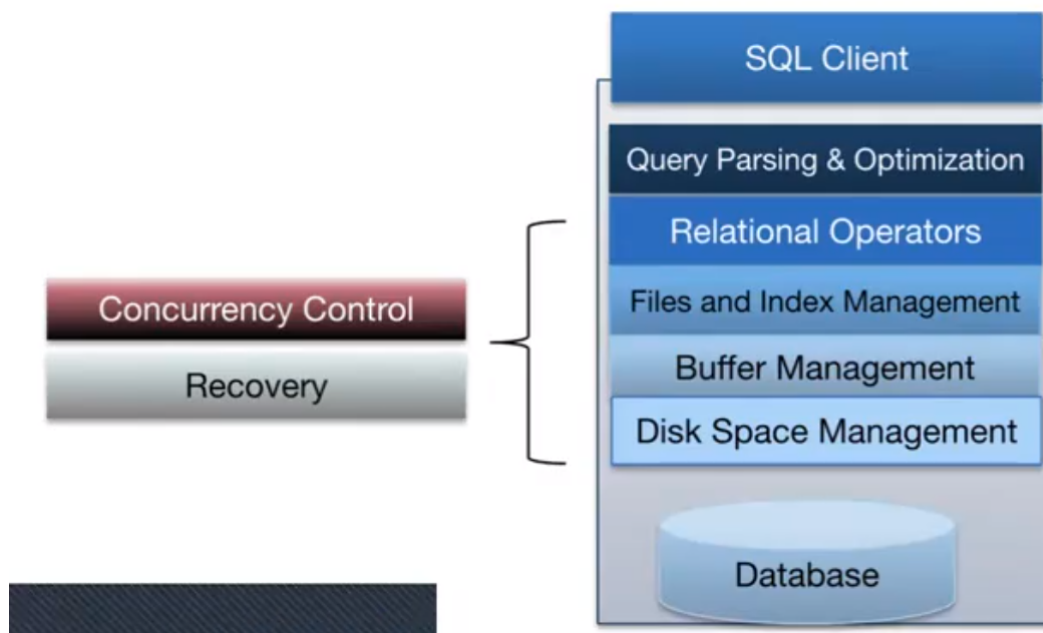
hw1: w3 (27 Sep)

hw2: w13 (12 Dec) in labs, source code will be given, 比hw1简单

CW3: ttl w4 & w13

大纲

- Lecture 1a - Introduction to the Module
- Lecture 1b - Data Storage Structures
- Lecture 2a - Indexing Techniques
- Lecture 2b - B+Tree Indexing
- Lecture 3a - Hash Indexing
- Lecture 3b - Advanced Indexing
- Lecture 4a - Introduction to Relational Model
- Lecture 4b - Query Evaluation Basics
- Lecture 5a - Query Evaluation: Selection
- Lecture 5b - Query Evaluation: Join
- Lecture 6a - Query Optimisation 1
- Lecture 6b - Query Optimisation 2
- Lecture 7a - Transaction Management 1
- Lecture 7b - Transaction Management 2
- Lecture 8a - Concurrency Control
- Lecture 8b - Failure Recovery
- Lecture 9a - Object-Oriented Database
- Lecture 9b - Distributed Database
- Lecture 10a - Web Technologies and Data Storage 1
- Lecture 10b - Web Technologies and Data Storage 2
- Lecture 11a - Big Data Storage
- Lecture 11b - Blockchain-based Storage (guest lecture by Prof. Xin Huang)
- Lecture 12 - Data Analytics
- Lecture 13 - Revision



不是应用而是原理 (low level), 一半内容涉及数据结构。

一些重点:

index (data structure)

查找优化 (算法)

事务管理

并发、恢复

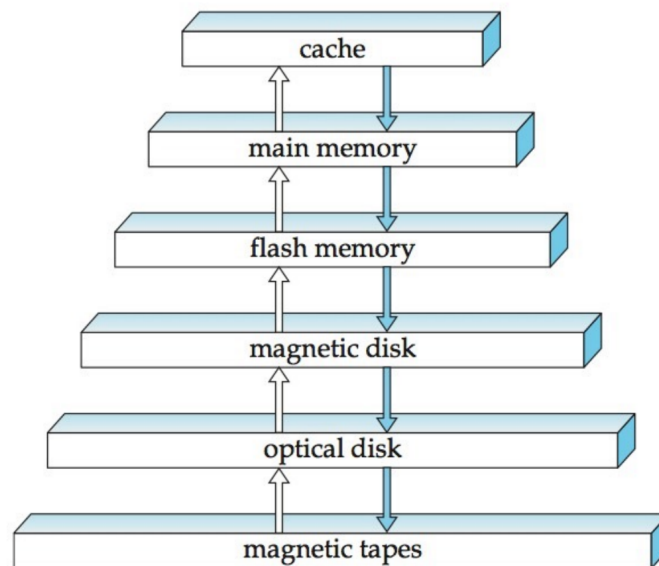
面向对象db、分布式db

L1

物理存储

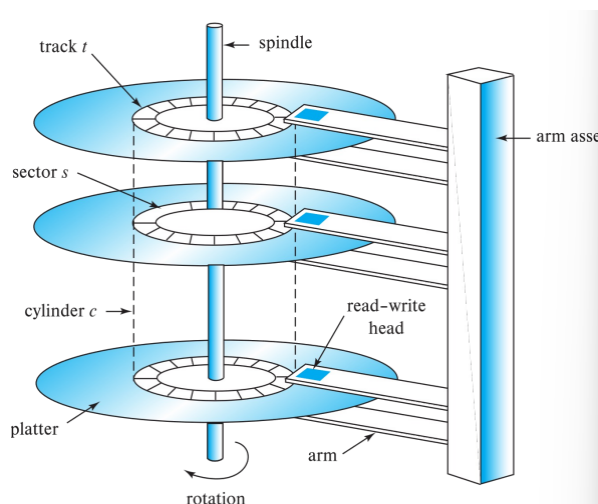
操作系统知识回顾

DBMS主要存储在磁盘/SSD上，因此I/O很重要。



物理存储可以两两一组分成三级存储。第一级的缓存和主存具有易失性（volatile）

磁盘



磁盘以tracks和其中的sectors分区，轨道转动，柱头摆动读取。速度很慢

读取时间=寻道 (track) 时间+旋转 (sector) 延迟+传输时间（相对很短）

磁盘和内存之间的数据传输的单位为block，最小的有512bytes，现在的典型大小是4到16kb。

page可以认为是同义词，作为内存映射的逻辑单位。

优化的方法：磁盘臂调度算法、电梯调度

闪存和SSD

有两种闪存，NOR和NAND，SSD用NAND，速度更快，但需要先擦除页面再写入。

RAID

通过冗余来提高可靠性，发生损坏时可以恢复丢失信息

通过并行来提高性能，提高数据在多个磁盘上的传输效率

Record

A **Database file (DB FILE)** is a collection of **pages**, which each contain a collection of **records**.

records是数据的基本存储单位，指table中一条完整的数据

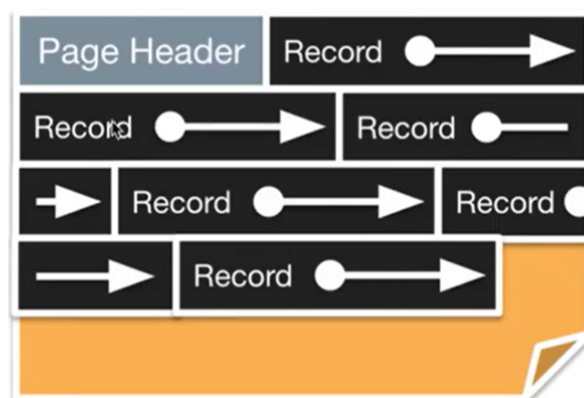
假设：每个record都包含在一个block内

- 每个页面上的**记录**如何组织，
- 每条**记录**的**格式**。

两种存储records方式：固定长度和可变长度

固定长度records

原始实现在删除时需要重新排列填补空位



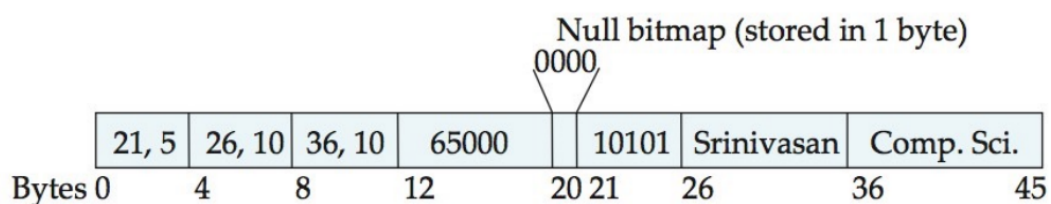
solution：用file header维护一个空余空间的链表（free list），插入记录时有空位则在变动指针，没有就添加在尾部

可变长度records

包括两部分：记录固定长度的信息的**初始部分**和属性长度可变的**数据区**。

在record的起始部分用固定长度中记录属性的信息(offset, length)

- offset表示了数据区的起始
- length表示这个属性的长度

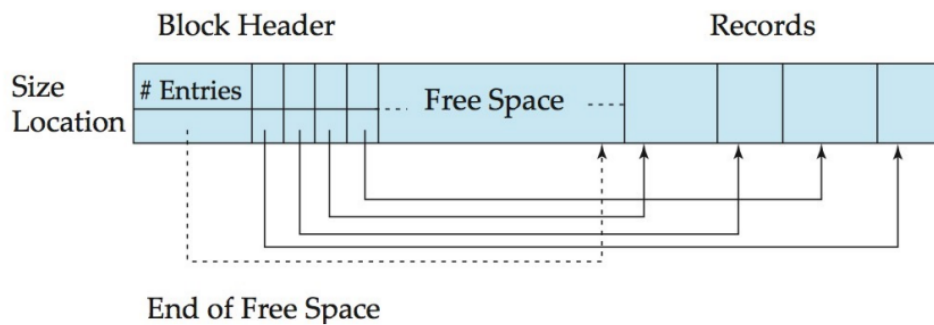


bitmap集中记录了哪些字段空闲

slotted-page structure

slotted-page structure的record中包含

1. record的数量
2. 空闲空间的尾部
3. 一个记录所有record地址和长度的数组



空闲空间连续分布，

在空闲空间的尾部插入，删除时移动删除项前的record，使得空闲空间连续

文件

records在文件中的组织形式

- heap, 无序
- sequential, 根据键的数值顺序排列记录
- Multitable clustering,
- B+树, 更高效的访问记录
- hashing, hash函数计算键

heap

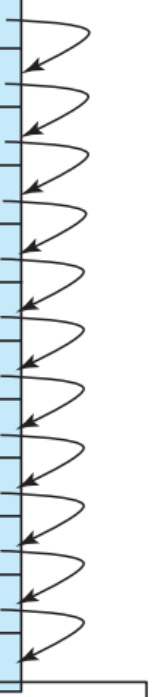
无序。与数据结构的堆不同

找到空闲空间就分配，无序

Sequential

顺序排列search key

10101	Srinivasan	Comp. Sci.	65000	
12121	Wu	Finance	90000	
15151	Mozart	Music	40000	
22222	Einstein	Physics	95000	
32343	El Said	History	60000	
33456	Gold	Physics	87000	
45565	Katz	Comp. Sci.	75000	
58583	Califieri	History	62000	
76543	Singh	Finance	80000	
76766	Crick	Biology	72000	
83821	Brandt	Comp. Sci.	92000	
98345	Kim	Elec. Eng.	80000	



多层聚类

分层次地将相关记录聚集在一起，在一个文件中存储多种关系

■ department

<i>dept_name</i>	<i>building</i>	<i>budget</i>
Comp. Sci.	Taylor	100000
Physics	Watson	70000

■ instructor

<i>ID</i>	<i>name</i>	<i>dept_name</i>	<i>salary</i>
10101	Srinivasan	Comp. Sci.	65000
33456	Gold	Physics	87000
45565	Katz	Comp. Sci.	75000
83821	Brandt	Comp. Sci.	92000

■ multitable clustering of *department* and *instructor*

Comp. Sci.	Taylor	100000	
10101	Srinivasan	Comp. Sci.	65000
45565	Katz	Comp. Sci.	75000
83821	Brandt	Comp. Sci.	92000
Physics	Watson	70000	
33456	Gold	Physics	87000

Index和b+树

引入

分析堆和有序文件的cost，前者需要full scan，后者二分查找 \log_2 仍然不够快（五分、十分查找？）

为什么需要索引？类比成文件系统中的GPS，可以从指定区域快速检索。

index是一种通过search key快速查找、修改数据项的数据结构。

idea of b+ 树

multiple level index 多级索引，索引之索引，分级可以减少读取的块数量，减少io开销。

对应到一种静态数据结构就是**M-way Search Tree**，m指每个节点的子节点数量和分支数量，而节点内的key数量则是m-1

问题在于，M-way ST没有任何规则保持良好的树性能，导致退化成线性。

B树 是一种-mway查找树 with rules，详细规则如下

1. 至少有 $m/2$ 的子节点才能创建新节点（控制高度）
2. root至少有两个子节点
3. 所有叶节点在同一层
4. 自下向上生成

插入的规则为节点过载+节点分裂+向上移动

B+树是一种典型的多级索引结构。只有叶节点有指向record的指针，并且叶节点之间是链表。不把key向上移动，而是向上**复制**，在叶节点保留了所有的key值。

- 只有叶节点有records指针：减少其余节点的空间占用（只需要索引的指针，无需数据指针），容纳更多索引，减少树高度
- 叶节点之间的链表结构：顺序扫描叶结点比中序遍历整棵树要快，范围查询更非常高效