CS179G : Project in C.S (Databases)

Department of Computer Science & Enginnering
University of California - Riverside
Spring 2003
TA: Demetris Zeinalipour csyiazti@cs.ucr.edu

Introduction to the Architecture of a DBMS

Structure of a DBMS

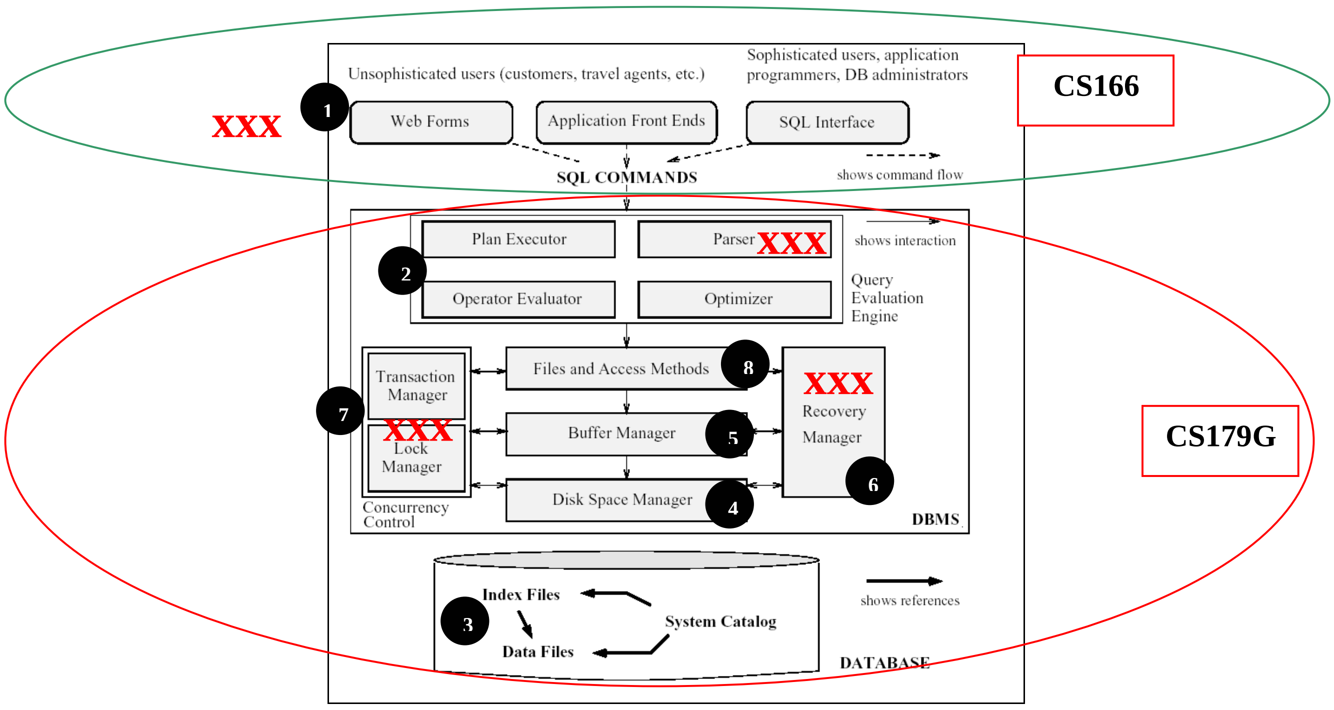


Fig1: The Achitecture of a DBMS

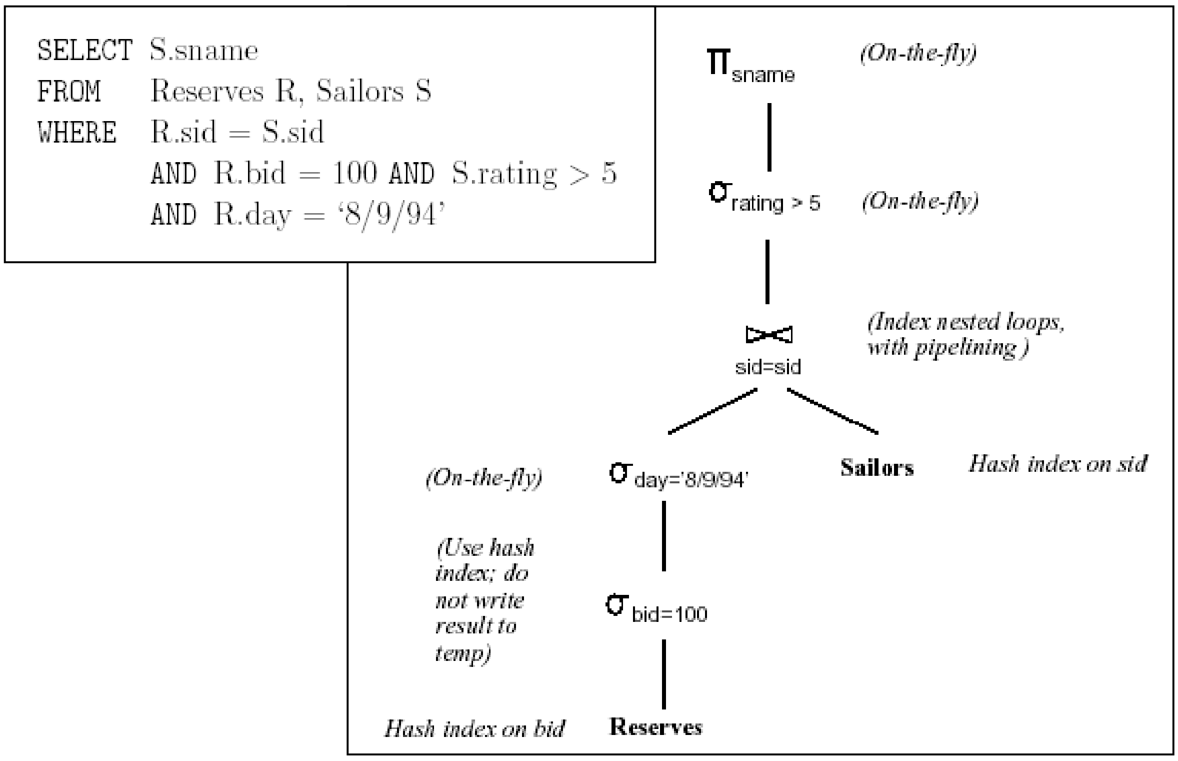
● **USER INTERFACE (the user)**
- o The DBMS accepts **SQL commands** generated from variety of User Interfaces.

- **DATABASE MANAGEMENT SYSTEM (the management)**

  ● **Query Evaluation Engine**
  - ▪ **Parser:** parses the users SQL and pass it to **Optimizer.**
  - ▪ **Optimizer:** The optimizer uses the lower levels and knows how the data is stored what type of *Indexes* are available

(from the system catalog), what is the distribution of data (with histograms) and many others. Therefore the optimizer selects from a list of *execution plans* the most appropriate one (the one that has the least estimated cost)
The Optimizer has 2 subcomponents which are the **Plan Generator** and the **Plan cost Estimator.**

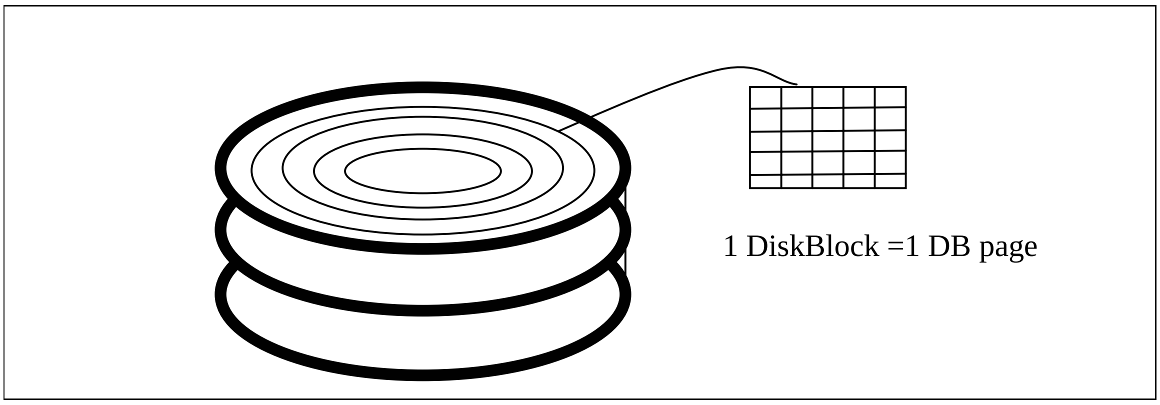
- **Plan Executor:** Generates a set of *Execution Plans* which are basically trees of relational operators with additional information about which access methods to use.



```
SELECT  S.sname
FROM    Reserves R, Sailors S
WHERE   R.sid = S.sid
        AND R.bid = 100 AND S.rating > 5
        AND R.day = '8/9/94'
```

3 **DATABASE (the core)**
Files: A collection of *pages*. Picture that Pages are chunks of bits stored on a Hard Disk. The size of a DBMS PAGE is a parameter to the DBMS usually 4Kb to 8Kb).

- **Data Files:** These basically contain the DATA in terms of pages or sometimes records.
- **Index Files:** These files again contain any indexes stored on disk.
- **System Catalog = Data Files + Index Files**



More about Pages

- Reading or Writing pages requires the disk arm and heads to move and transfer a **Disk-Block (also called DB-page**) between *Main Memory* and *Disk*. **This is called an I/O operation.**
- In other words suppose you want to read 1 record (which is not in main memory – referred later on as the buffer manager). You need to transfer 1 Block which may contain e.g. 10 records/block. Therefore you are bringing in 10 records
- The time to read or write a block varies depending on the location of the data:
    - AccessTime= SeekTime + RotationalDelay + TransferTime

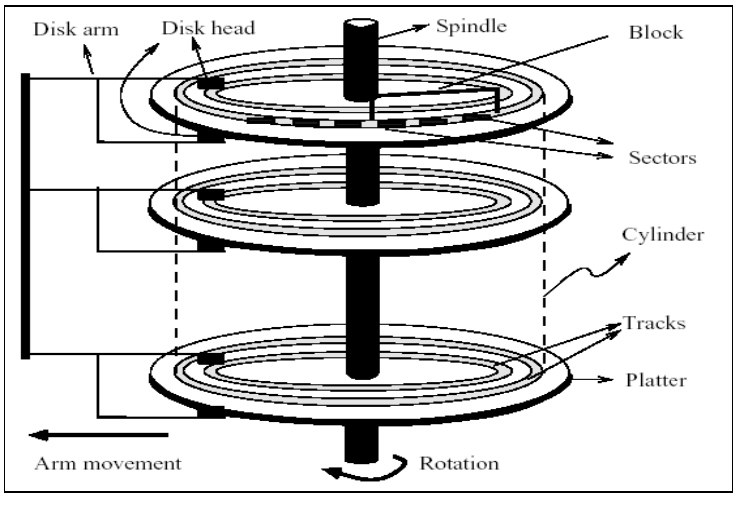


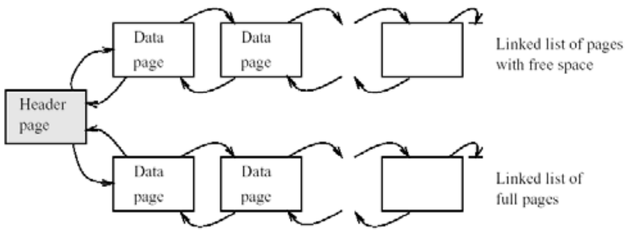
Fig1: Structure of a Disk The values of the parameters shown in the disk picture are **disk-specific**. This implies that they vary from disk to disk. The Operating System is in charge of dealing with those parameters. Reading or Writing to disk is done in **disk blocks** which are contiguous sequences of bytes. The DBMS uses its own parameters (i.e. pages) in order to provide an abstraction layer between the hardware specific details of a disk and the database implementation.

Issues Omitted

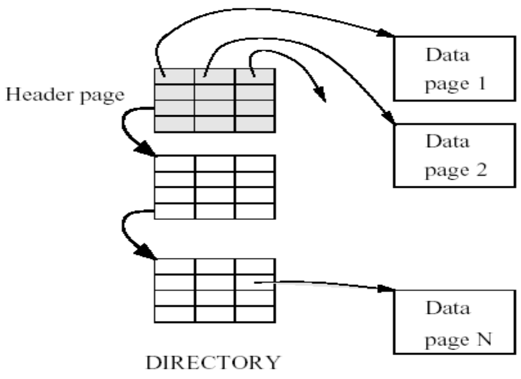
1) **How are pages organized on disk**

In other words we know that pages are stored either contiguously or are dispersed on some area of the disk but how do the higher levels know where is each page located

a. We will see Heap **Files as Linked List of pages** (unordered pages) => adding easy, finding without indexes is difficult


Header page
Data page
Data page
Linked list of pages with free space
Data page
Data page
Linked list of full pages

b. OR **Heap Files as Directory of Pages** (unordered pages but searching is done through the Directory which is much faster) => adding easy



2) **Page Formats**

Each Page has N slots which can accommodate 1 or more records

a. Using Fixed Length Records

i. Each Page contains same number of records
ii. Fetching/Inserting a record is easy => simply by offset manipulation (if occupied slots are always stored in the first K positions) if Not stored in first K slots occupy 1 slot as a bitmap index which will indicate which is empty
iii. Drawback: We may end up wasting too much sequential space

b. Using Variable Length Records

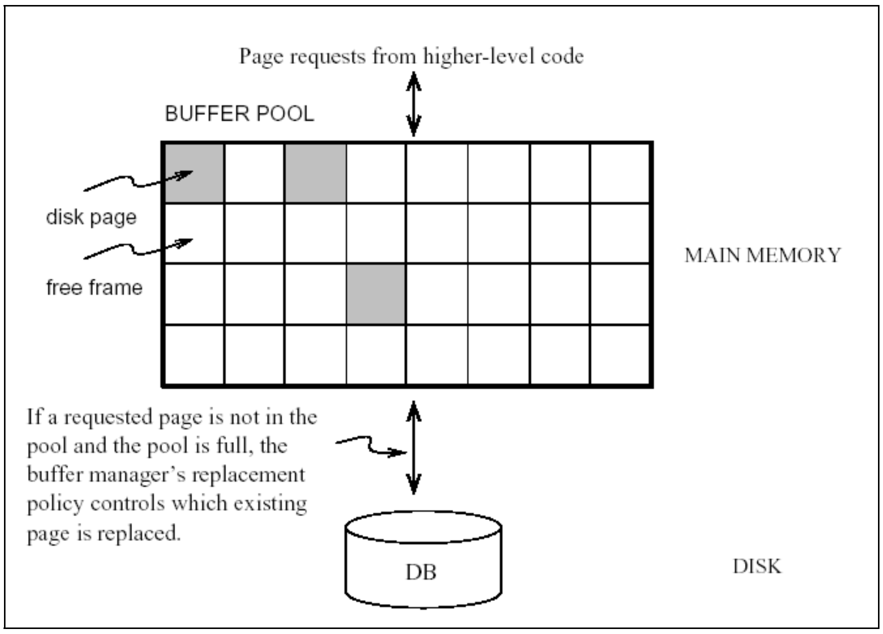
i. Good space utilization
ii. Inserting a new record might be difficult because we have to find just the right space which might not be available.
iii. Directory of slots <record offset (start), record length (end)> solution

3) **Record Formats**
   a. RecordID= < PageID, SlotID>
   b. Information Common to all records of a table is stored in **System Catalog** rather than in the record

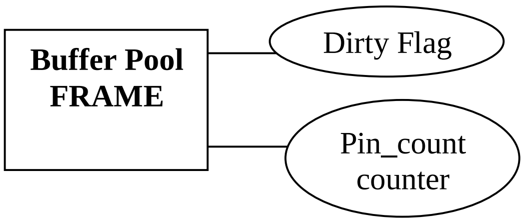
DATABASE MANAGEMENT SYSTEM (continued)

4 **Disk Space Manager (DSM)**
- It hides the details of the underlying hardware (and OS) and allows higher levels of the software to think of the data as a collection of pages
- It provides commands to read / write / allocate / deallocate units of data (i.e. a PAGE)
- Initially the DSM allocates a big (sequential) portion of the disk
- The DSM keeps track of DiskBlocks in use. Therefore addition and deletion may create areas with “holes”
- The DSM has its own Disk Management policy and does not rely on OS files for many reasons
  - A DBMS cannot assume any specific features of the OS filesystem
  - A 32-bit OS uses files of size 2^32=4GB. A DBMS may need to store and address larger files

5 **Buffer Manager (7.4)**
- The role of the Buffer Manager is to keep track of the pages that are used most (so that access to those pages becomes cheap)


Page requests from higher-level code
BUFFER POOL
disk page
free frame
MAIN MEMORY
If a requested page is not in the pool and the pool is full, the buffer manager’s replacement policy controls which existing page is replaced.
DB
DISK

- Every Page(DiskBlock) required from the DBMS must be brought into the buffer pool which resides in **main memory (**if it is not already there)
- If a page X is updated by the higher level (application - e.g. updating the employees salary) then the Buffer Pool must be aware of that so that it can WRITE the page X once it decides to remove X from the pool
- The Buffer Pool has a pre-specified space. Therefore some **Replacement Policy** must be in place which will save to secondary storage the e.g. Least Used Pages. We will discuss more policies LRU, MRU, Random, FIFO
- Each Frame (page) in the Buffer Pool has 2 flags associated with it
    - **Dirty Flag** – Indicates that the page was modified
    - **Pin_count** counter – The number of times a page was requested but not released.
    - **Pinning** means incrementing counter
    - **Unpinning** means decrementing counter



- It is much more complex in reality. For instance what happens in case of system of a crash, concurrent access to the same records by different users or processes?

6 **Recovery Manager**
It maintains a log and restores the system to a consistent state after a crash

7 **Transaction Manager & Lock Manager**
They ensure that Transactions request & release locks according to a suitable locking protocol and they schedule transactions.
The Lock Manager keeps track for locks and grants locks on database objects When they become available

8 **Files & Access Methods**
Maintains the System Catalog and provides method to access the underlying data and indexes. It organizes information within a page, records + other functionality.