Data Storage and Querying in Various Storage Devices

Unit IV

By

Muheet Ahmed Butt

Classification of Physical Storage Media

- Speed with which data can be accessed
- Cost per unit of data
- Reliability
  - data loss on power failure or system crash
  - physical failure of the storage device
- Can differentiate storage into:
  - **volatile storage**: loses contents when power is switched off
  - **non-volatile storage**: contents persist even when power is switched off. Includes secondary and tertiary storage, as well as battery-backed up main-memory.

Physical Storage Media

- Cache-fastest and most costly form of storage; volatile; managed by the hardware/operating system.
- Main memory:
  - general-purpose machine instructions operate on data resident in main memory
  - fast access, but generally too small to store the entire database
  - sometimes referred to as **core** memory
  - volatile — contents of main memory are usually lost if a power failure or system crash occurs

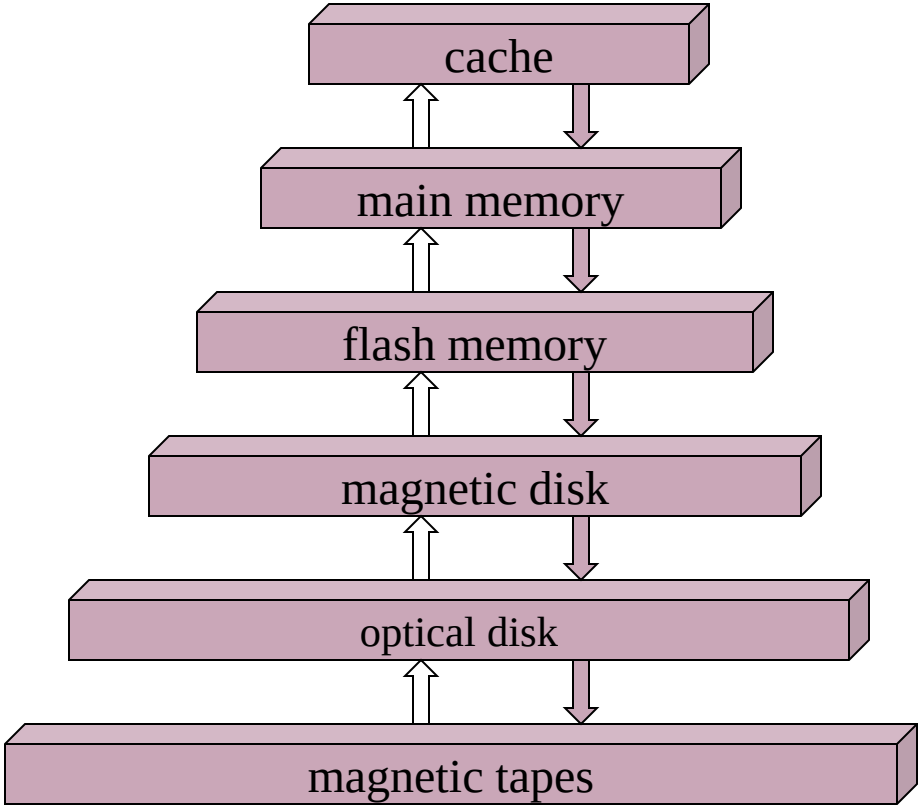
Physical Storage Media (Cont.)

- Flash memory – reads are roughly as fast as main memory; data survives power failure; but can support a only limited number of write/erase cycles
- Magnetic-disk storage – primary medium for the long-term storage of data; typically stores entire database.
  - data must be moved form disk to main memory for access, and written back for storage
  - **direct-access** – possible to read data on disk in any order
  - usually survives power failures and system crashes; disk failure can destroy data, but is much less frequent than system crashes

Physical Storage Media (Cont.)

- Optical storage – non-volatile. CD-ROM most popular form.
- Write-once, read-many (WORM) optical disks used for archival storage.
- Tape storage – non-volatile, used primarily for backup (to recover from disk failure), and for archival data
  - **sequential-access** – much slower than disk
  - very high capacity (5 GB tapes are common)
  - tape can be removed from drive $\Rightarrow$ storage costs much cheaper than disk

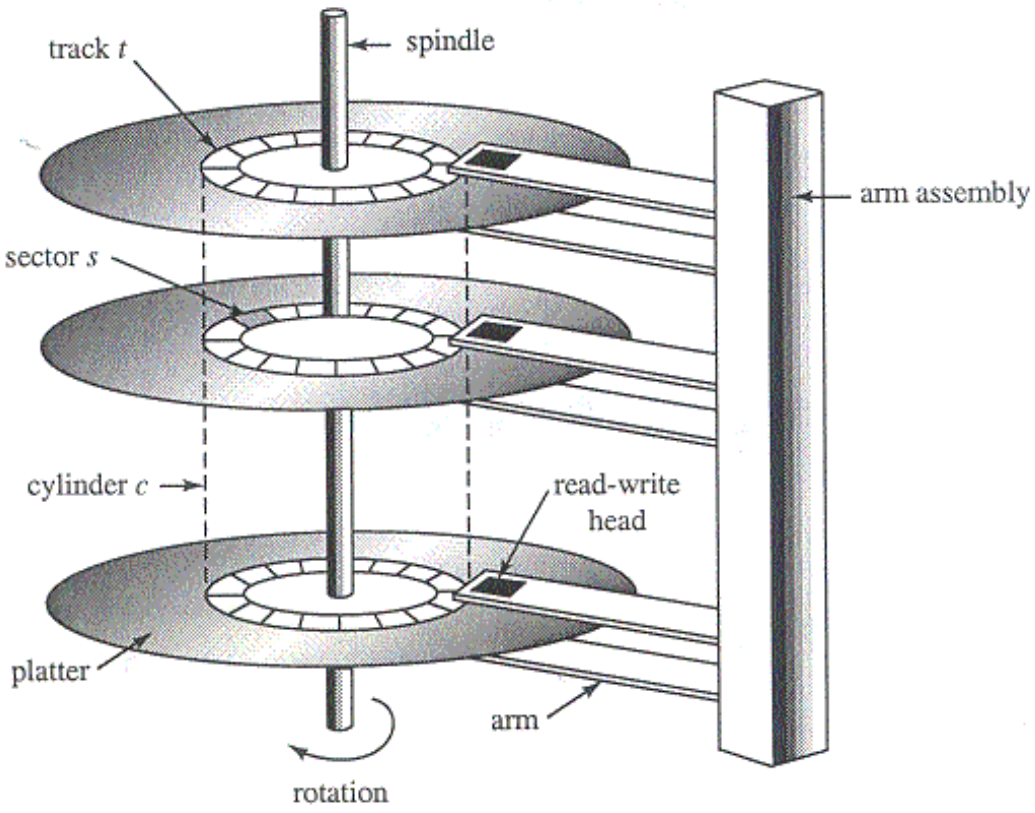
Storage Hierarchy



Storage Hierarchy (Cont.)

- **primary storage**: Fastest media but volatile (cache, main memory)
- **secondary storage**: next level in hierarchy, non-volatile, moderately fast access time; also called **on-line storage** (flash memory, magnetic disks)
- **tertiary storage**: lowest level in hierarchy, non-volatile, slow access time; also called **off-line storage** (magnetic tape, optical storage)

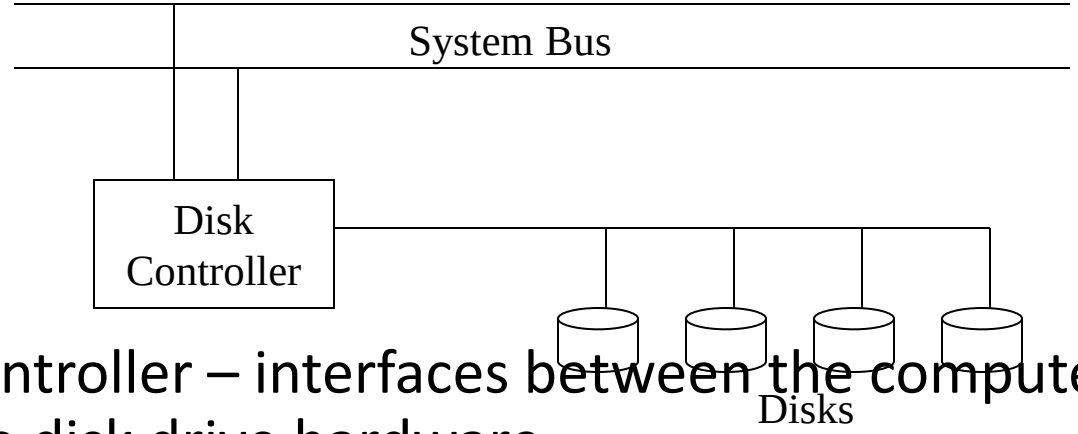
Magnetic Disks Mechanism



Magnetic Disks

- Read–write head – device positioned close to the platter surface; reads or writes magnetically encoded information.
- Surface of platter divided into circular **tracks**, and each track is divided into sectors. A sector is the smallest unit of data that can be read or written.
- **Cylinder** *i* consists of *ith* track of all the platters
- To read/write a **sector**
  - disk arm swings to position head on right track
  - platter spins continually, data is read/written when sector comes under head
- Disk assemblies – multiple disk platters on a single spindle, with multiple heads (one per platter) mounted on a common arm.

Disk Subsystem



- Disk controller – interfaces between the computer system and the disk drive hardware.
- Accepts high-level commands to read or write a sector
- initiates actions such as moving the disk arm to the right track and actually reading or writing the data

Performance Measures of Disks

- **Access time** – the time it takes from when a read or write request is issued to when data transfer begins. Consists of:
  - **Seek time** – time it takes to reposition the arm over that correct track. Average seek time is 1/3rd the worst case seek time.
  - **Rotational latency** – time it takes for the sector to be accessed to appear under the head. Average latency is 1/2 of the worst case latency.
- **Data-transfer rate** – the rate at which data can be retrieved from or stored to the disk.
- **Mean time to failure (MTTF)** – the average time the disk is expected to run continuously without any failure.

Optimization of Disk-Block Access

- **Block** – a contiguous sequence of sectors form a single track
  - data is transferred between disk and main memory in blocks
  - sizes range from 512 bytes to several kilobytes
- Disk-arm–scheduling algorithms order accesses to tracks so that disk arm movement is minimized (**elevator algorithm** is often used)
- File organization – optimize block access time by organizing the blocks to correspond to how data will be accessed. Store related information on the same or nearby cylinders.
- **Nonvolatile write buffers** speed up disk writes by writing blocks to a non-volatile RAM buffer immediately; controller then writes to disk whenever the disk has no other requests.
- **Log disk** – a disk devoted to writing a sequential log of block updates; this eliminates seek time. Used like nonvolatile RAM.

Indexing and Hashing

Muheet Ahmed Butt

Insertion in Index

- Dense Index
  - If entry is present in index

(Value coming more times)

If I store pointer to all records with same values

update the index with this pointer to this record

Else

do nothing

Else

Include a new entry and pointer to this new record

End if

Insertion in Index

- Spare Index (Assume index stores one value in the block and that values is the least search key valued record

  - If Adding a in a New Block Then
    - Add a New Index Record
    - Insert This entry in the new Block
  - Else

    We are adding in some old block

    If (this record search value is < index records search value) Then

        Update the index record pointer to the new record

        Insert the record in the appropiate block

        insert the new record

    Else

        Do Nothing

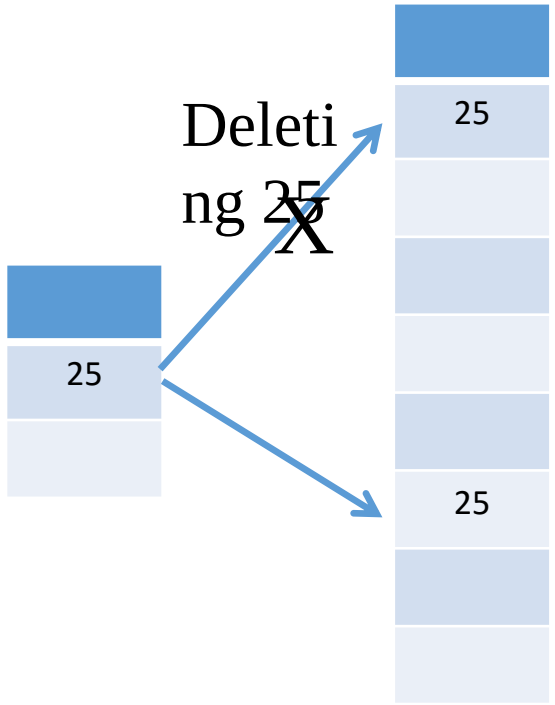
        Just insert the new record.

    End If

    End IF

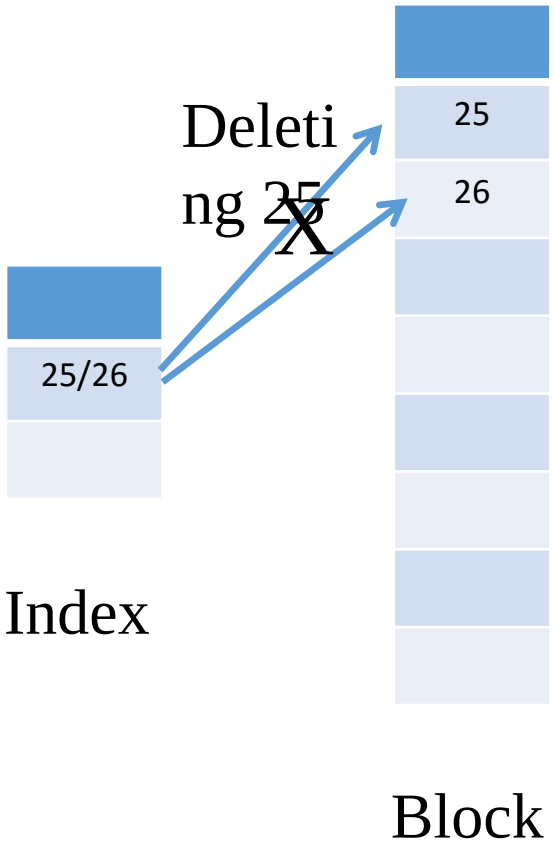
Deletion

- Dense index
  - If (the record deleted was the only record with that value) Then
    - Delete the Index Entry
    - Delete the Record
  - Else
    - We update the Pointer of the index entry to the next record with same search values



Deletion

- Sparse index (We are storing least search value per Block)
  - If (record doesn't contain an index entry)
    - Just Delete the record from the file
  - Else
    - Replace the pointer so that it points to next higher search value in the block
  - endif



- Hash Table is a data structure by which we can search elements very fast.
- Complexity O(1) Times
- Basic Concept is a Hash function Hash(x)

Disadvantages in Indexing

- Disadvantage in sequential file organization is we must access an index structure to locate data.
- More I/O Operations required
- Must use binary search for faster access

Hashing Definition

- Hashing allows us to avoid accessing the index strucutre.
- It also provides a way of constructing indexes

- **Bucket : Unit of storage that can store on or more records.**
  - **Actually a Disk block that could be chosen smaller or larger than the disk block.**

Hash Function

- **Let K denote the set of all search-key values, let B denote the set of bucket address**
- **A hash function h is a function from K to B.**
- **To insert a record with search key Ki we compute h(Ki) which gives us the address of the bucket for that record.**
- **If there is a space in the bucket to store the record. Then the record is stored in the bucket.**

Hashing Basics

- **To perform a lookup on a search key value Ki compute h(Ki) then search the bucket with that address.**
- Suppose two search keys K5 and K7 have the same hash value that is

  h(K5) = h(K7)

  If we perform lookup on K5 and K7

  The buckets with the h(K5) contains records with search –key values K5

  The buckets with the h(K7) contains records with search –key values K7

Linear Probing

- Make table with indices (0 to mod-1)
- Mode number and insert the moded index
- If moded index is full
- Do Linear Probing
- If you reach end go back to index 0

Example 1

- Draw a result of hashing 19, 50, 89, 40 into a table
- Here h(x) = x mod 5 using linear probing

0	50
1	89
2	40
3	
4	19

19 mod 5
= 4
50 mod 5
= 0
89 mod 5
= 4
40 mod 5
= 0

Examples To Do

- Draw a result of hashing 4, 14, 19, 9004 into a table if the h(x) = x mod 5 using linear probing

- Draw a result of hashing 3, 58, 8, 68 into a table if the h(x) = x mod 5 using linear probing

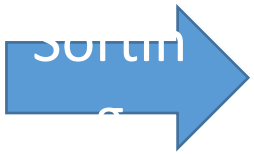
Open Chaining

- Make table with indices (0 to mod-1)
- Mode number and insert the moded index
- If moded index is full and collision
- Do Open Chaining
- After adding all values sort all chains from lowest to highest

Example 1

- Implemented as linked list
- Draw the result of hashing 11, 33, 36, 16, 55, 69, 49, 82, 66, 5 into a table if the h(x) = x mod 7 using Open Chaining

0	49		
1	36		
2	16		
3			
4	11		
5	33	82	5
6	55	69	



0	49		
1	36		
2	16		
3			
4	11		
5	5	33	82
6	55	69	

Things to do

- Draw the result of hashing 4, 12, 8, 3, 10 into a table if the h(x) = x mod 4 using open Chaining

- Draw the result of hashing 3, 58, 8, 68 into a table if the h(x) = x mod 5 using open Chaining

Double Hashing

- Make table with indices (0 to mod-1)
- Mode number and insert the moded index
- If moded index is full and collision
- Do Quadratic Probing
  - Insert the original number into a second equation
  - Second equation does not give u not the place to insert the number but the number of buckets jump by

Example 1

- Draw the result of hashing 19, 15, 89, 39 into a table if the h1(x) = x mod 5 using Double Hashing Equation h2(x) = 3 – (x mod 3)

0	15
1	89
2	
3	39
4	19

19 Mod 5 = 4
15 Mod 5 = 0
89 Mod 5 = 4
39 Mod 5 = 4

89 Mod 3 = 2

3 – 2 = 1

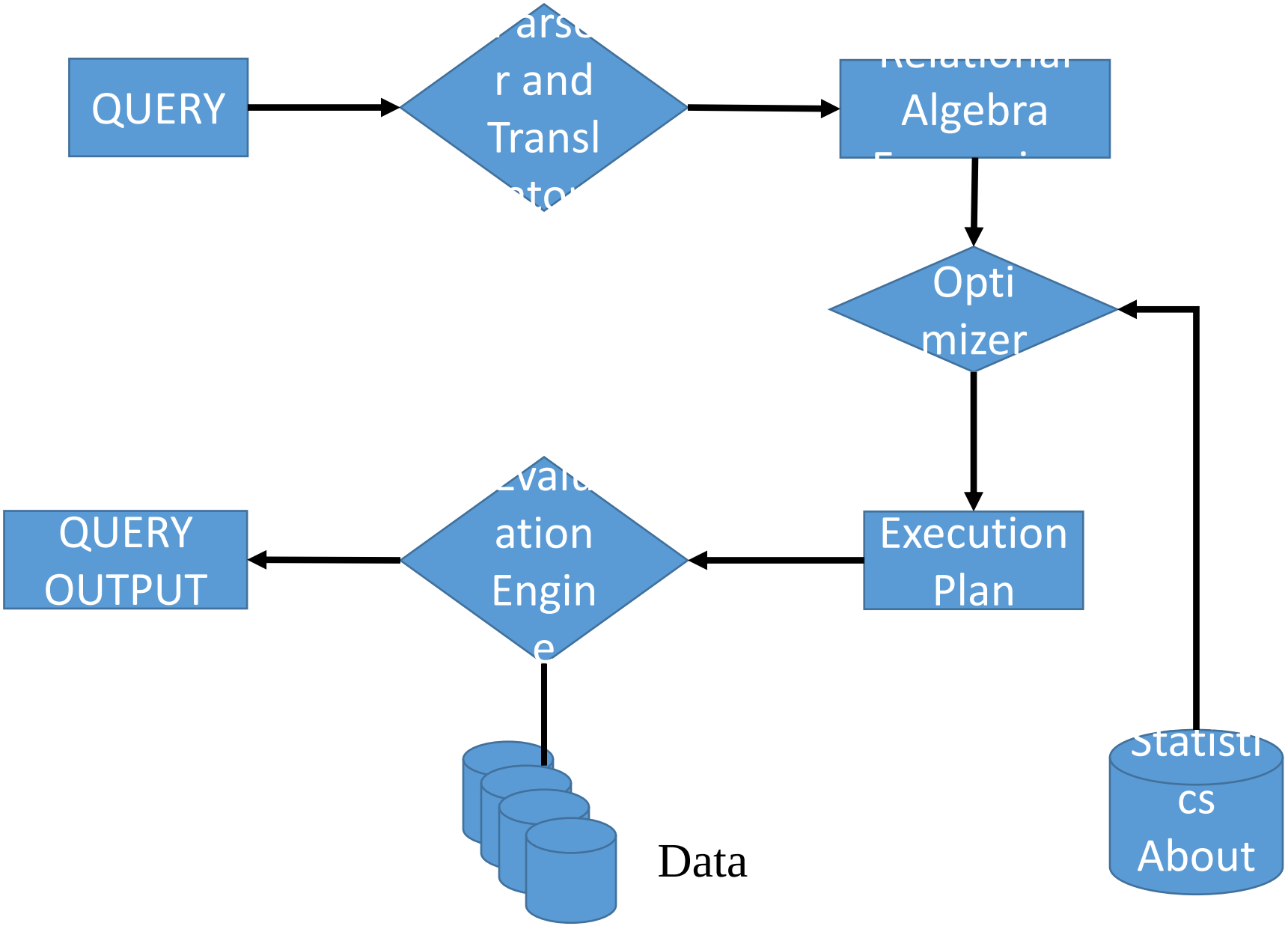
Query Processing and Optimization

Muheet Ahmed Butt

Query Processing

- Query processing is a list of activities required to obtain the required tuples that satisfy a given query

QUERY PROCE


QUERY
r and
Transl
Algebra
Opti
mizer
Execution
Plan
ation
Engin
e
QUERY
OUTPUT
Statisti
cs
About
Data

Queries

- Select * from EMP
- Insert into EMP Values (1001, ‘Majid Zaman’,’Nigeen Srinagar’)
- Update EMP set empno = 1001 Where Empno = 345
- Grand Update, Insert on EMP to user1, user2

Parser and Translator

- Checks for Syntax of the Query
  - [Select EMP Name from EMP Having EMPNAME = ‘ABC’]
- Checks for Schema Elements
  - [Select EMP.Ename, EMP.USR, EMP.EMP123 from EMP]
- Converts the query into relational algebra Expression

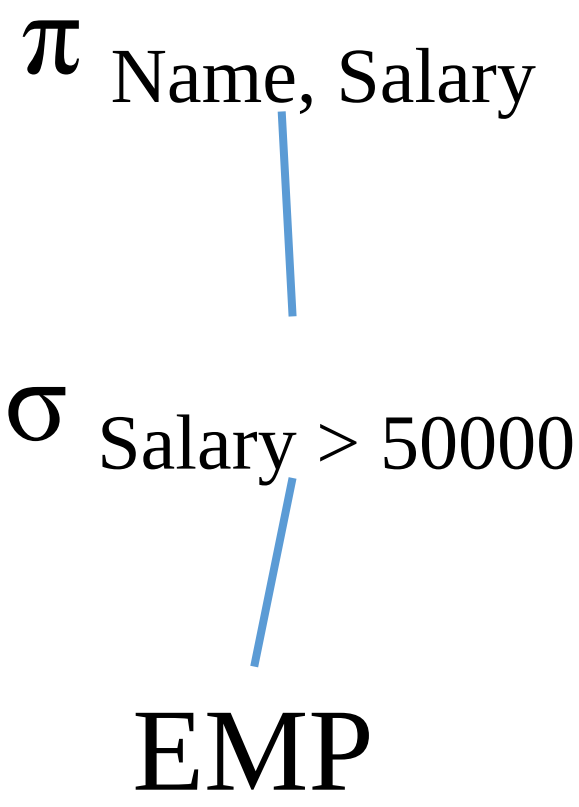
Optimizer

- Find the best plan to evaluate a relational algebra
  - Find the names of employees whoes salaries is greater than 5000
    - Select EMP.Name, EMP.Salary from EMP where EMP.salary > 50000
  - Convert into Relational Algebra Expression
    - [Method 1] [Project] $\pi_{\text{EMP.Name, EMP.Salary}}$[Restrict] $\sigma_{\text{(EMP.Salary > 50000)}}$ EMP
    - [Alternate] $\sigma_{\text{(EMP.Salary > 50000)}}$ $\pi_{\text{EMP.Name, EMP.Salary}}$ EMP

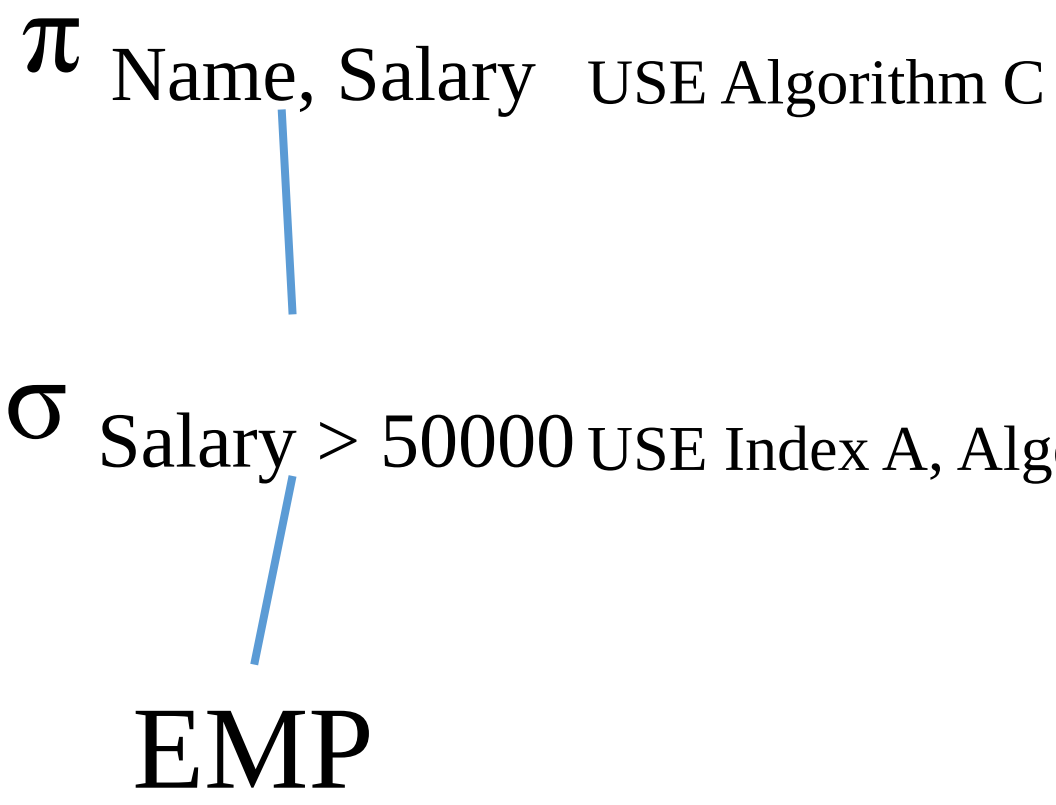
Optimizer

- Let this is our Query given by Parset
- [Project] $\pi_{\text{EMP.Name, EMP.Salary}}$[Restrict/Select] $\sigma^{\text{EMP}}_{\text{(EMP.Salary > 50000)}}$
- Optimizer Prepares a Evaluation Plan
    - All equivalent Relational Algebra Expressions
    - Best Relational Algebra Expression which it would find the minimum cost/Time.
        - [Find Expression with Least Cost]
    - For Selection and Projection can be performed using various algorithms

Query Tree/ Operator Query Tree



Query Evaluation Plan



Query Evaluation Plan is an Output c

Use of Statistics

- Contain Information regarding Query Execution Plan
  - Size of Relation [See 10000 Records]
  - No of Blocks being used to store a Relation
  - Does it have a index or not
  - Number of Records in a Relation

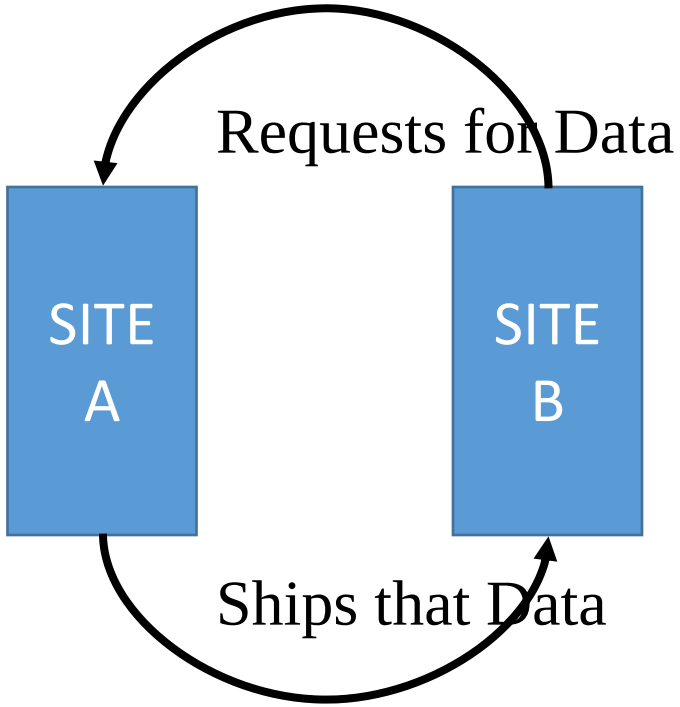
  - Majorly used to Arrive at a Particular Plan

Query Evaluation Engine

- Evaluate the Above Plan and Get the Results
  - Execute those Algorithms
  - Access the Data Using those Algorithms for Accessing Data.

Measures of Query Cost

- What do I take into account when I calculate the cost of the query
- Basic Measures
  - Disk Access
    - Primary Concern is Disk Access Time
  - CPU Cycles have consumed
    - Difficult to Calculate
    - CPU Speed is much faster than disk speed
    - CPU Cost is relatively lower than Disk Cost
  - Transit Time in Network
    - Concerned with Distributed or Parallel Systems



Disk Access Cost

- No of Seeks we have to make to a particular block
  - Cost = N * AvgSeekTime
- No of blocks i am going to read
  - Cost = N * AvgBlockReadCost
- No of Blocks I am going to write
  - Cost = N * AvgWriteCost

- Seek cost be will be different for Random and Sequential IO
  - Random [Less] Sequential [More]
- Cost of Writing is more than cost of reading

For Real Time Systems

- Response Time has to be minimized
- Buffer Size should be High

Algorithm for Selection Operators

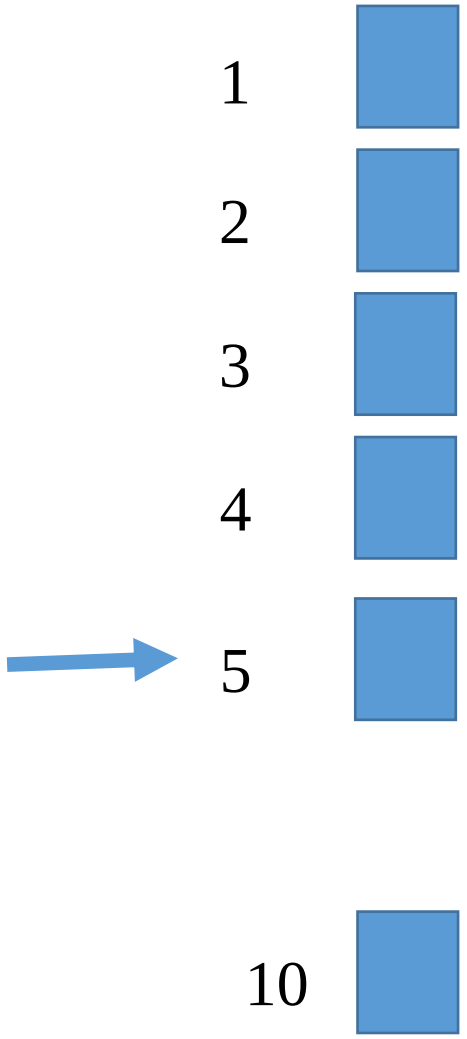
- File Scan
  - Sequencial I/O
- Index Scan
  - Random I/O

File Scan Algorithm

- A1 or Linear Search
  - Look at Blocks One by One Sequentially to find that particular record
  - If File is Sorted
    - Cost = No of Blocks in Relation [BR]/ 2
    - Worst Case Cost = BR [Record is Last]
- Features
  - Any Selection Condition
  - Any Ordering
  - Availability of Indexes

Binary Search

- If the Salary 50000 is in 5th Block
- A2 or Binary Search
  - Cost = Log_2 BR + No of Blocks Containing Records with same value



Transaction Processing Concepts

Muheet Ahmed Butt

What is Transaction

- A transaction is an atomic unit of access with is either completely executed or not executed at all
- “All or nothing” Principle [All instructions should be executed in full or none of them is executed]
- Example: Transfer of Funds
- Transaction Modifies the state of the database
- A Program that modifies the state of database

Transactions

- Many Enterprises use databases to store information about their state
  - Eg : Balances of all depositors
- The Occurrence of Real word event that changes the enterprise state requires the execution of a program that changes the database state in corresponding way
  - Eg; Balance must be updated when you deposit
- A Transaction is a program that accesses the database in response to real world events

Examples of Transactions

- Debit Transaction
  - Debit (Account No, Debit Amount)
  - Begin Transaction
  - Read (Account No, Balance) ID1
  - Balance = Balance – Withdraw-Amount ID2
  - Write (Account Number, Balance) ID3
  - End Transaction

Events

Three Instructions
Jointly constitute as

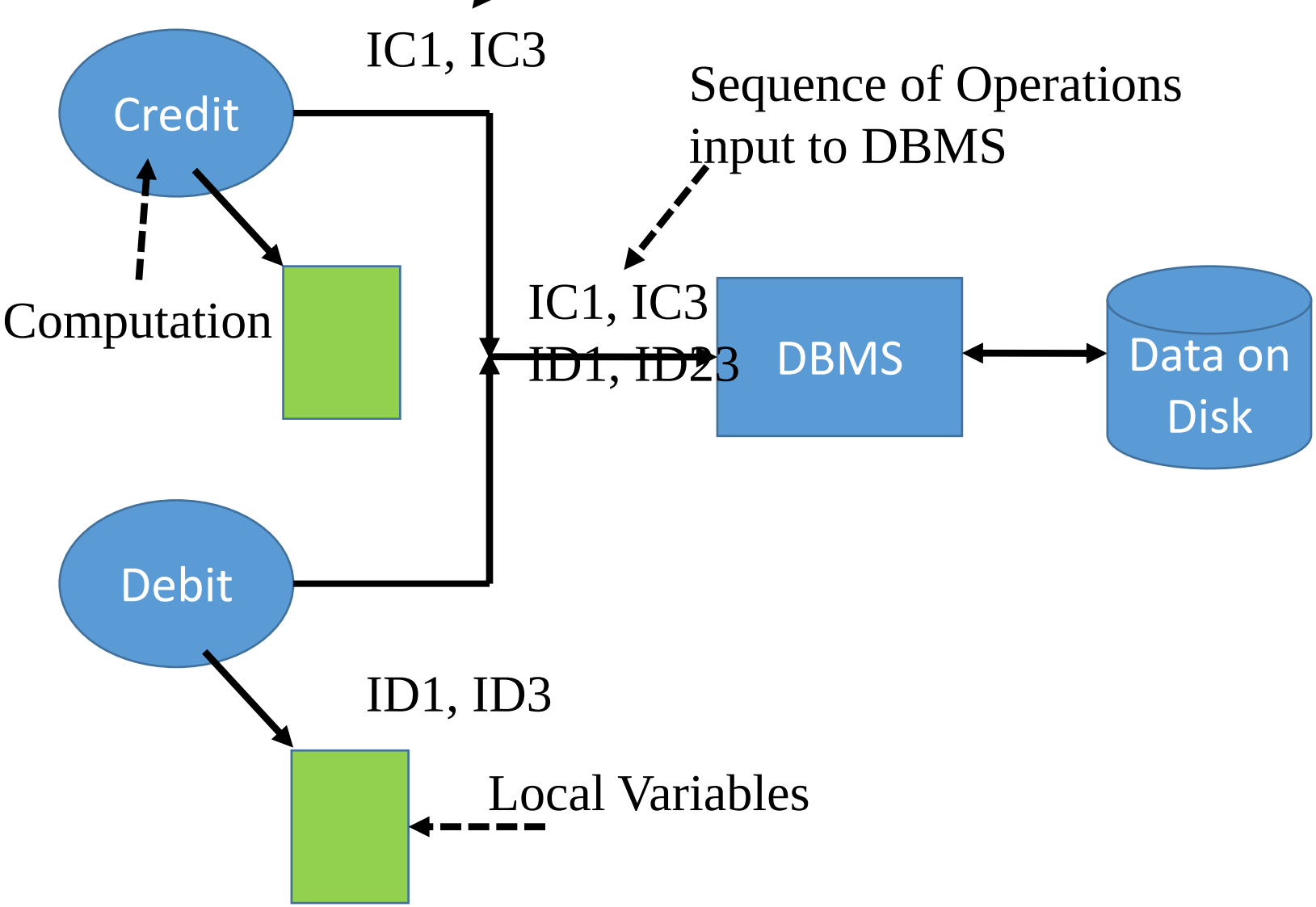
Examples of Transactions

- Credit Transaction
  - Debit (Account No, Debit Amount)
  - Begin Transaction
  - Read (Account No, Balance) IC1
  - Balance = Balance + Credit Amount IC2
  - Write (Account Number, Balance) IC3
  - End Transaction

Events

Three Instructions Jointly constitute as Transaction

CREDIT & DEBIT Card Transactions



Read Operations – Read (X)

- X Can be any Data like Account No, Balance
- Read Data from Database
  - Find the Address of the Disk Block that contains Item X
  - Copy Disk block into a buffer in Main memory
  - Copy Item X from the Buffer to Program Variable named X

Write Operation – Write (X)

- Find the Address of disk block that contains X
- Copy that Disk Block into a buffer in the Main Memory
- Copy Item X from the program variable named X into the correct location in the buffer
- Store the updated block from the buffer back to the disk.

Types of Failures

- Computer Failure
  - Hardware or software error in computer during transaction execution
  - External Failure
- Transaction Error
  - Failure Caused by an operation in the transaction
    - Divide by Zero
  - Internal Failure
- Data Access Inconsistency

Failure Types Contd.

- Local Errors
  - Conditions that cause the cancellation of Transaction
    - Eg: Data need for transaction not found.
  - Concurrency Control Enforcement
    - Transaction may be aborted by the concurrency control method

Failure Types

- Disk Failure
  - Loss of Data in disk blocks during a transaction due to disk read write head crash
- Physical Problems in Catastrophes
  - Problems like power failure fire etc.

Intuitive Understanding

- Scenario One
  - What happens if the credit card transaction fails after executing IC1 and before Executing IC2.

  - ATOMICITY

Scenario Two

- What happens if Credit Transaction and Debit Transaction Execute Simultaneously [If both Operate on Same/ Different account No]
  - IC1, ID1, IC2, ID3, IC3, ID3

  - IC1 – 500
  - ID1 – 500
  - If you are Depositing 200 and Withdrawing 100 Rs.
  - IC2 = 500 + 200 = 700
  - IC3 = 700 Back to Database
  - ID2 = 500-400 = 100
  - ID3 = 400
  - Credit Amount Completely Lost

  - CONSISTENCY PROPERTY

Scenario 3

- What happens if the result of the credit transaction are visible to debit transaction before it is actually written onto the databases
  - IC2 : Write Balance: Read by ID1
  - Lead to CASCADING ABORTS
  - ISOLATION PROPERTY

Scenario 4

- What happens if the Database Server Crashes before the changed data is written to the stable storage

- DURABILITY PROPERTY

Transaction Properties

- Atomicity : Either All the instructions are executed in full or none
- Consistency: When Multiple transaction are accessing data simultaneously, the access is protected through concurrency control mechanisms.
- Isolation: Results of one transaction will not be visible to the other transactions till the transaction commits. [Cascading Abort]
- Durability: The Effects of the committed Transactions are not lost after commitment.

- Whole Called as **ACID**

Concurrency and Recovery

Muheet Ahmed Butt

Concurrency and Recovery Problem

- Database Recovery means the process of restoring the database to a correct state in the event of a failure.
- Concurrency Control is the ability to manage simultaneous processes involving the databases without having them interfere with one another.
- These Mechanisms are database Specific

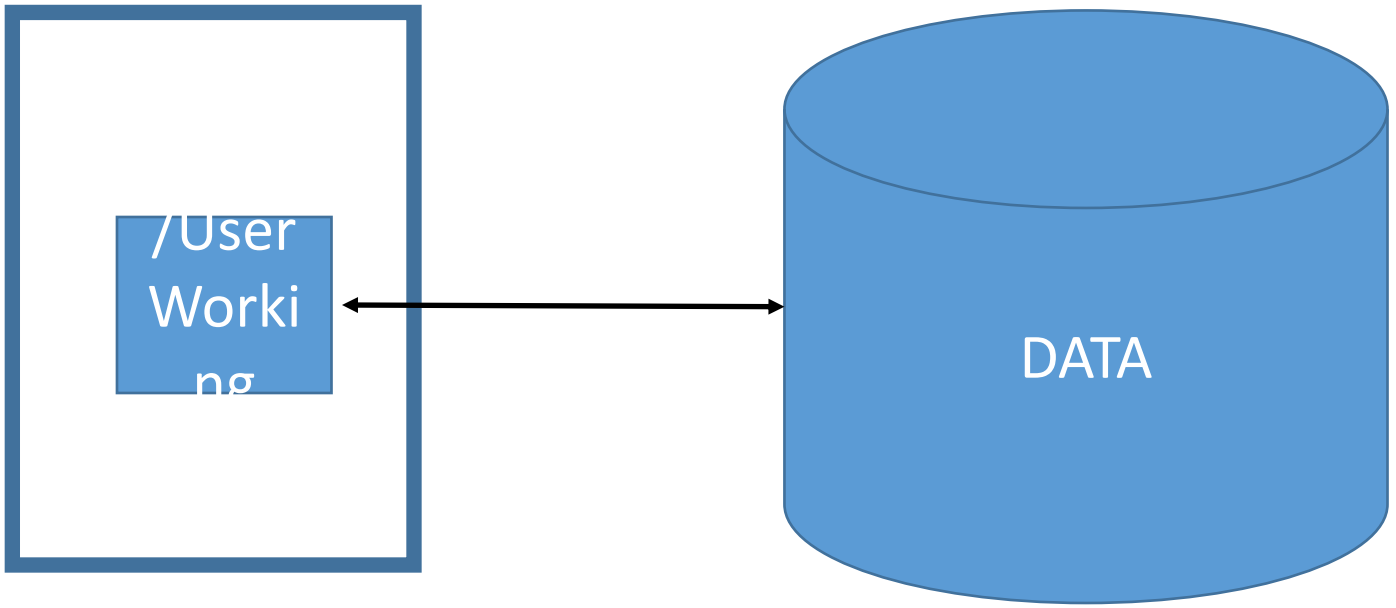
Concept of Transaction

- Transaction can be thought of as a logical unit of work on the database.

- It may be an entire program, a portion of a program or single command
- Should be Atomic Access to the Database

Example

- Consider Relation Schema Having 4 Relations
  - STUDENT (SID, NAME, MAJOR, CREDITS)
  - FACULTY (FID, FNAME, DEPT, RANK)
  - CLASS (C#, FID, SCHED, ROOM)
  - ENROL (C#, SID, GRADE
- Transactions
  - Update the number of CREDITS in a STUDENT Record
    - Read No of Credits
    - Update No of Credits
    - Write No of Credits
  - Change the SID assigned to a particular student.
    - We need to Change in Student and ENROL
    - Enroll can have many SID tuples.



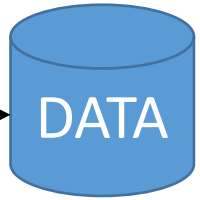
Main Memory

Secondary Storage Device/Disk

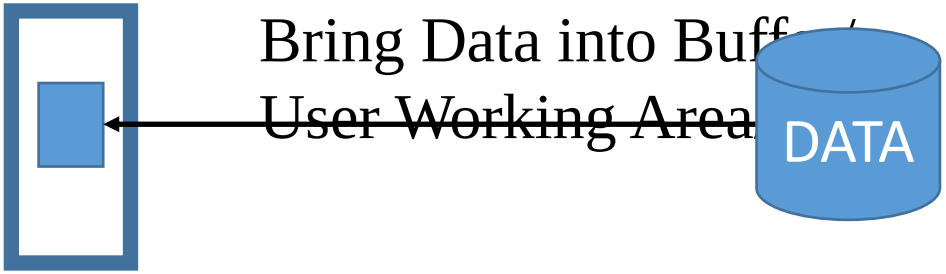
Buffer Corresponds to Each User working on Database

For Every Update Operations

STEP 1 Locate the Record

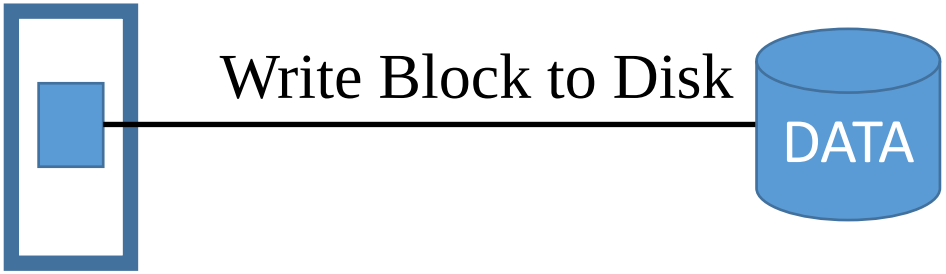


STEP 2 Bring Data into Buff[illegible]
User Working Area



STEP 3 Update in User Worki[illegible]a

STEP 4 Write Block to Disk



What can Go Wrong

- Need for Recovery
  - Physical Disasters
  - Malicious Sabotage [Intentional]
  - Carelessness
  - Disk Malfunction
  - System Crashes
  - Systems/ Softwares / Applications errors

  - Something has gone wrong from STEP 1 to STEP 5

Two Ways Transaction Can Terminate

- Successful Termination
- Abort
  - Something has gone Wrong
  - Cannot be Executed Successfully

- In case it is aborted it has to be brought back/ undone / Rolled back
  - Brought Back to a Consistent State

Rollback

- Treat Transaction a Atomic Unit
- Maintain a Log File
  - $<T_i, d_k, \_\_\_>$
  - $<T_{i+1}, d_{K+1}, \_\_\_>$
  - $<T_{i+2}, d_{k+2}, \_\_\_>$
  - $<T_{i+3}, d_{k+3}, \_\_\_>$
  - $<T_{i+4}, d_{k+4}, \_\_\_>$
  - …..
  - $<T_{i+5}, d_{k+1}, \_\_\_>$

  - Where T is Transaction
  - d is data item
  - ___ Old Value
  - All are kept in sequence Fashion
  - Timestamp can also be used
- If need for rollback arises we follow this log.
- Multiple Copies of Database are also kept for Recovery.

Concurrency Control

- Lost Update Problem
  - Suppose a Couple Mr. and Mrs. Andrabi have a joint account in a bank.
  - Both of them work in different offices.
  - On a certain data, after getting off from office at 5:00 PM.
  - They Decide to withdraw Rs. 2000/- each from ATM nearest to the respective office.
  - If they are lucky then only one withdraw should be shown in Database Transaction.

Transactions

- Read Balance
- Balance = Balance – 2000
- Write Balance

Withdrawal Sequence

	t0	t1	t2	t3	t4	t5	t6
Balance in the Database	10000	10000	10000	10000	10000	8000	8000
Bal in Mr. Andrabi Workspace	-	10000	10000	8000	8000	-	-
Bal in Mrs. Andrabi Workspace	-	-	10000	10000	8000	8000	-

Concurrency Control Problems

- Operating Systems
  - The Shared Resources is the Hardware [Memory, Printer, CPU etc.]
  - The Shared Resources is not Changeable Resource
  - Lost Update Problem not applicable to OS.

- Databases
  - The Shared Resource is Data
  - Data is a Changeable Resource

Traditional Solution for Currency is Locking

- A **lock** is an indication that a particular transaction is now using the data/resource and any other transaction which wishes to use the same data/resource will not be allowed unless the transaction which is using it has unlocked it.
- Any transaction which wishes to manipulate the resource has to lock it first.
- If no other transaction is using the resource as is free the lock is granted to the transaction
- When the transaction is over lock is unlocked.

Locking

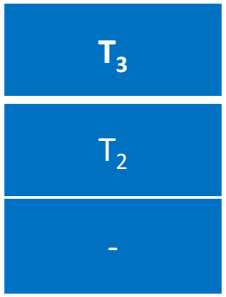
- T_k : Lock A_i [T=Transaction A = Resource]
- --
- Other Transactions for the Resource A_i enter in wait state
- --
- T_k : UnLock A_i [T=Transaction A = Resource]

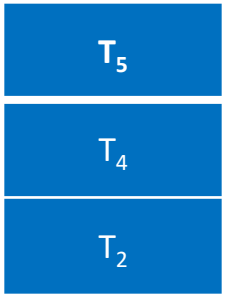
Problems with Locks

- These Problems occurred with Operating Systems in 1970
- Live Locks [Starvation in OS]
- Dead Locks

Live Locks [Not a Great Problem]

- Transaction T_1 : Lock A
- Transaction T_2 : Lock A [Wait State]
- Transaction T_3 : Lock A [Wait State]
- Transaction T_1 : Unlock A
  - **T_3 gets the Lock**
- Transaction T_4 : Lock A [Wait State]
- Transaction T_5 : Lock A [Wait State]
- Transaction T_3 : Unlock A
  - **T_5 gets the Lock**
- **Solution is USE QUEUE**





Dead Lock

- T1: Lock A Lock B Unlock A Unlock B
- T2: Lock B Lock A Unlock B Unlock A

- to: T1 Calls for Lock on resource A
- t1: T2 Calls for Lock on resource B
- t2: T1 Calls for Lock on resource B
- t3: T2 Calls for Lock on resource A
- t4: T1 …..
- t5: T2…..
- Nor T1 can Proceed or T2 can Proceed

Thank You Very Much & All the Very Best for Your Exams