Database Systems Mid 1

- ✅ Intro Slide -- Done
- ✅ Relational Algebra
- ✅ Chapter 1 -- Done
- ✅ Chapter 2 -- Done
- ✅ Chapter 3 - Done
- ✅ Chapter 4 -- Done
- ✅ Book Chapter -- Done

Intro Slide

- Data, Information, Database, DBMS, Database System, Metadata
- Computer System Components
- Database Functionality [4]
- Three Schema Architecture
  - Two reasons for proposal [Prog Data Independence, Support for multiple views]
  - Three Schemas and their definitions
  - Diagram
  - Mappings
- Miniworld
- Data Independence
  - Logical Data Independence
  - Physical Data Independence
- DDL
  - SDL
  - VDL
- DML
  - Host Language
  - Query Language
- High Level Language - Non Procedural
- Low Level Language - Procedural
- ER Models
- Hierarchical Model
  - Tree Structure, Links
  - Difference with Network

    - DAG Structure, boxes and lines.
    - Diagram rules.
    - Single vs Many to many
- Network Model
    - Restricted form of ER
    - Diagram rules
    - RLink and dummy records
- Relational Model
    - Column, Row, Table
    - Domain
    - Attribute
    - Relational Schema
- Integrity Constraints
    - Domain, Key, Entity, Referential Integrity
- SQL Summary
- Functional Dependencies
- Normalization
    - 1NF
    - 2NF
    - 3NF
    - BCNF
    - Algo for preserving NF in Join

Chapter 1

- Capabilities of DBMS
    - Persistent Storage
    - Programming Interface
    - Transaction Management
- Problems with Megaton
    - Tuple Layout on Disk [Change, Deletion, ASCII Storage] all expensive.
    - Search expensive as no indexing, read entire file
    - Brute force querying
    - No buffer, cache
    - No concurrency control
    - No reliability
    - No Security
- Main memory Buffer Management
- Transaction Processing and Control
- ACID
- Query Processing

    - Parsing
    - Pre-processing
    - Optimiser

Chapter 2

- Cache
    - Onboard, L2
    - Single/Multiprocessor system updation
- Virtual Memory
- Moore's Law
    - Applicable on
    - Not applicable on
- Secondary Storage
    - Reading Writing
    - Block Allocation
- Tertiary Storage
- Volatility
- Disk
    - Platter, Head, Actuator, Cylinder
    - Track
    - Sectors, Gaps
    - Measures
- Time = Seek Time + Rotational Delay + Transfer Time + Other Delays
- Average Random Seek Time [Formula]
- What does other time consist of
- Why sequential access is better than random access
- Cost of reading vs writing [*With/without verification*]
- Typical values of all quantities
- Block Modification
- Block Address [4 constituents]

- **Secondary Storage Optimised Usage**

- Elevator Algorithm
- Disk Failures:
    - Intermitten - checksum parity
    - Media Decay
    - Drive Crash
    - Write failure

- Stable Storage
- Disk Mirroring
- Redundant Arrays of Independent Disks [RAID]
  - Solves 2 problems, transfer rate and reliability
- RAID Advantages:
  - Read speed due to multiple disks ***2x***
  - Read speed due to **Bit level Data Stripping *8x***
  - Redundancy via stripping and parity
- RAID Levels:
  - **LEVEL 0**
    - Block stripping, no redundancy, no parity
  - **LEVEL 1**
    - Disk Mirroring
  - **LEVEL 2**
    - Bit level stripping
  - **LEVEL 3**
    - Bit interleaved parity
  - **LEVEL 4**
    - Block interleaved parity
  - **LEVEL 5**
    - Block interleaved Distributed Parity
  - **LEVEL 6**
    - Additional information
  - **0-1 vs 1-0**

Chapter 3

- Fields -> Records -> Blocks -> Files -> Memory
- Representation of boolean, numbers, characters, words [using count and null]
- Record header data:
  - pointer to schema
  - length
  - timestamp
  - the record
- Schema data:
  - attributes of relation
  - data types of attributes
  - order
  - constraints
- Block Header data:
  - Link to other blocks

    - meta information about block
    - meta information about tuples
    - dictionary of block offset
    - bock ID
    - timestamp
- Record identified by block+offset in secordary storage
- Logical and physical address
- Advantages of using a logical and a physical address:
    - easy modification
    - easy deletion
- Databse Address
- Memory Address
- Following Database Address is comparatively expensive.
- Translation Table
- Pointer Swizzling
    - Automatic
    - On Demand
    - No
- Unswizzling
- Pinned block
- Variable length field:
    - pointer contains length of record, offset field.
- Repeating fields:
    - group occurences, use pointer
- Variable format records:
    - Sequence of:
        - #Fields | code identifying field | Field type | Length of field if applicable
- Spanned Records
    - bit to tell if block fragmented or not
    - bit to tell if first record partial or last
    - pointer to next fragment
- Unspanned Records
- BLOBS stored in a sequence of blocks.
- **INSERTION:**
    - Add to neighbour block/ create overflow block with an overflow pointer
- **DELETION:**
    - Maintain header, take care of pointers.
- **UPDATE:**
    - No change on storage system.

- Column Store Advantange:
  - Compact Storage
  - Easiser Read for data mining
- Row Store Advantage:
  - More efficient write
  - efficient read for record access

Chapter 4

- Index file is a sorted file
- Search Key
- Dense/Sparse Index
- Efficient, permanently in main memory.
- Multilevel index perks
- Handling Duplicate Keys
- Index modification on data manipulation:
  - To Manipulate Data:
    - Create overflow block
    - Insert new blocks in sequential order
    - slide tuples to adjacent block if no space.
  - **Index Modification Table**
- **Secondary Index**
  - What it does
  - Why it's dense
  - Multi-level secondary index (can be sparse)
  - **Indirection** - When
- Clustered File Structure
  - When
- Inverted Indexes
  - When
  - Posting
- Vector Space Model
- Weighted Vector Space Queries
- Benefits of simple index:
  - Good for scans
  - simple
- Demerits:
  - insert expensive
  - Loss of sequentiality
- **B-Tree**
  - **LOGIC**

    - block usage [2]
    - indexing
    - **Rules**
        - At least n+1/2 pointers used
        - Notation for leaf and non-leaf.
        - leaf has a node pointing to the next leaf.
        - balanced
    - Minimum table
    - B-Tree lookup
    - B-Tree insertions
        - *Simple, Logical, just focus*
    - B-Tree Deletion
        - *Minimize the tree as much as possible*
    - **Efficient**
        - Few I/O
        - Rare splitting, merging
        - root in main memory
- **Hash Tables**
    - Basic logic
    - Hash Function
    - Secondary Storage
    - Insertion
    - Deletional - Consolidation
    - Good Hash function criterion [***equal distribution***]
    - keys sorted in a bucket if CPU is time critical, and insertion deletion not frequent
    - **Utilization**
        - 50-80 Rule
    - **Very Efficient:**
        - 1 Disk I/O for lookup
        - 2 for insetion deletion
    - Blocks per bucket should be kept low
    - **Dynamic Hash Tables**
        - **Extendable Hash Table**
            - Indirection introduced
            - Blocks power of 2
            - Insertion
            - Deletion
            - Overflow for duplicate
            - Less wasted space, no full reorganisation
            - Directory doubles in size, indirection

- **Linear Hashing**
  - No of buckets remains less
  - i, r, n important, r/n determinal factor for creating new bucket
  - # used slots / # of slots = U is important, we expand file
  - Less wasted space, no full reorganisation
  - No indirection, can still have overflow chains

File System Implementation

- inode
- allocation structure
- free list
- bitmap [inode + data]
- Super Block[1] + inode[5] + data[56] + inode bitmap[1] + data bitmap[1]
- block = inumber * size of inode / block size
- sector = (block * block size) + inode start address / sector size
- Directory Organisation
- Free Allocation Policy
  - Pre-Allocation of Contiguous Blocks
- Reading from a file
- Writing to a file [5 I/O]
- Creating a file [6 I/O min]
- Static Partitioning
- Dynamic Partitioning
- Unified Pages Cache
- Write Buffering and advantages [3]