Foundations

Lecture 2

Searching

- Common operation performed by database system
- Ex: SELECT
- Linear search: baseline for efficiency
    - Start sequentially
- Record: collection of values for attributes of a single entity instance
    - A row of a table
- Collection: set of records of the same entity type
    - A table
- Search key: a value for an attribute from the entity type
    - >= 1
- Contiguous allocated list: all n*x bytes allocated as a single chunk of memory
    - In the form of an **array**
    - Continuously in order
    - Faster for random access
    - Slow for random insertions
        - Slow for inserting anywhere but the end
- Linked list: each record needs x bytes, additional space for 1 or 2 memory addresses
    - Individual records are linked together in a chain using memory addresses
    - May not be in order
    - Faster for random insertions

Binary search

- Input: array of values in sorted order; target value
- Output: index of target value

Lecture 4 1/15/25

- CPU
    - Root processor
- Registar
    - Memory that they use is expensive
- L1 cache
- L2 cache
    - Further away from processor
- RAM
- SDD/HDD
    - Lots of storage
    - Persistent

Hard drive

- Reading a “block” of byte
- 64 bit integer is 8 bytes; if i wanted to reach it, it can be read by the 2048 byte block size

Other abstracted layers

- Data base management systems

AVL tree
K: V (64 bit int),
4x8 bytes -> 32 bytes

- To index on disk storage vs in ram storage
- Possible that every node is on a separate block on a hard drive
    - 7 x 2048 byte block size
    - But not optimizing the structure – make better use of the 2048 bytes of memory
    - In order to increase performance → decrease height of AVL tree
- 7 x32 bytes of memory to store 7 nodes
-

Maximize value of each disk block

B+ tree

- Nodes will have up to 128, 256 values that i am indexing (so just the keys)
- Indexing the data structure to minimize height of the tree
- Cant keep them unsorted array because hard to know where the value would fall into between the values of the node
- K1 through K5 are on one block, and each of the children are on a diff block

Lecture 5 1/16/2025

- Set up group project
    - Github repo
    - Git classroom?

Lecture 1/27/25

- Relational Models
    - benefits
    - Standardized data model and query language
    - ACID compliance
        - Atomicity, consistency, isolation, durability
    - Works well with highly structured data
    - Can handle large amounts of data
    - Well understood
    - Lots of tooling
    - Lots of experience
- Increase efficiency
    - Indexing
    - Direct controlling of storage
        - Column oriented vs row oriented
    - Query optimization
    - caching/prefetching
    - Materialized views
    - Precompiled stored procedures
    - Data replication and partitioning
- Transaction Processing
    - Transaction: a sequence of 1+ CFUD operations performed as a single logical unit of work
    - COMMIT → entire sequence succeeds
    - ROLLBACK or ABORT → entire seq fails
    - Ensures data integrity, error recovery/simple error handling, concurrency control, reliable data handling
- Atomicity
    - Transaction is an atomic unit of work
        - All or nothing
- Consistency
    - Transaction move from one consistent state to another consistent state
    - Consistent dsta: data models are enforced
- Isolation
    - Two transactions being executed at the same time but dont affect eh other
    - Or else, leads to ‘dirty read’, ‘non-repetable read’ phantom reads

Lecture 2/5

Redis data types

- Keys: strings or any binary sequence
- Values: strings, list, sets, hashes
- Supports concurrency
    - Occurring in parallel
    - Two processing two datasets and indexing into redis

Introspection: collects all the metadata, for code completion