**Data** Some values referring to real world facts, may be in various formats; **Database** A large collection of inter-related data; **Database management system** databases + a set of programs that store and access the data; Drawbacks of storing databases in file systems: Difficulty/Inefficiency in accessing data, Data redundancy and inconsistency, Data isolation (files may be in different formats), Atomicity problems, Concurrent access anomalies, Integrity problem;

**Data abstraction** physical level (how the data are actually stored, complex low-level data structured in detail), logical level (what data are stored, relations among data, physical data independence), view level (only part of the database, simplify user interaction)

**DBMS functional components** Storage manager, interface between the low-level data and the application programs/queries, consists of: File manager (manages the allocation of space on disk storage & the data structures used to represent information on disk), Transaction manager (ensures that the database remains in a consistent state), Authorization & integrity manager (tests for the satisfaction of integrity constraints and checks the authority of users to access data), Buffer manager (Responsible for fetching data from disk storage into main memory, decide what data to cache in main memory); Disk storage, consists of: Data files, Data dictionary (database schema), Indices; Query processor, converts high-level user requests to efficient low-level commands, consists of: DDL interpreter (Interprets DDL statements and records the definition in the data dictionary), DML compiler (Translates DML statements in a query language into an evaluation plan, query optimization), Query evaluation engine (Executes low-level instructions generated by the DML compiler)

**E-R Modeling** Entity: An object that exists & is distinguishable from other objects. Entities have Attributes. Entity set: a set of entities of the same type that share the same attributes. Relationship: an association among entities; Relationship set: a set of relationships of the same type. Super key: a set of one or more

attributes whose values uniquely determine each entity. Candidate key: a minimal super key. Weak entity set: An entity set without a primary key.

**Relational Algebra** Select (σ), Project (π), Union (∪), Difference (-), Cartesian product (×), Rename (ρ), Intersection (∩), Natural Join, Left join, Right join, Division (÷), Assignment (←), Aggregation (g)

**SQL** Create: CREATE TABLE table\_name (variable\_name VARIABLE\_TYPE); Edit table: ALTER TABLE table\_name OPERATION; Primary Key: PRIMARY KEY (variable\_name); Foreign Key: FOREIGN KEY(foreign\_key) REFERENCES table\_name(foreign\_key); Insert: INSERT INTO table\_name (column\_name) VALUES (new\_value); Delete: DELETE FROM table\_name WHERE condition; Update: UPDATE table\_name SET variable\_name = new\_value WHERE condition; Case: UPDATE table\_name SET variable\_name = CASE WHEN condition THEN new\_value ELSE new\_value END;

Select: SELECT column\_name FROM table\_name WHERE condition ORDER BY order; (order = ASC or DESC) Like: % matches any substring, \_ matches any character, patterns are case insensitive; In: specifies discrete values in the WHERE search criteria: WHERE variable\_name IN (selection); Aggregation: Aggregation Functions: AVG, MIN, MAX, SUM, COUNT; Group by: GROUP BY variable\_name; Conditional for groups: HAVING condition; Join: SELECT column\_name FROM table\_name JOIN table\_name ON condition; (JOIN = OUTER JOIN or LEFT JOIN or RIGHT JOIN)

**File & Storage** Media: CPU Cache: extremely fast, volatile, managed by the computer system hardware (not DBMS); Main Memory: volatile, fast (in nanoseconds), volatile, a few GBs, too small for the entire database; Magnetic Disk: Primary medium for long‐term storage, non-volatile, much slower than main memory (data are buffered into memory before accessed by DBMS). Magnetic Disk Components: Read‐Write Heads: positioned closely

to the platter surface; Platters: each has two surfaces covered with magnetic materials for recording information, each is divided into circular tracks, approx. 50k-100k tracks per platter., each track is further divided into sectors (the smallest unit of data that can be read/written, typically 512B), typically 500-10k sectors per inner track & 1k-2k sectors per outer track. I/O steps: seek (locate track), rotation (locate sector), transfer data. Values: Access Time: the time between the request and the start of data transfer, consists of seek time (typically 2-30ms) & rotational latency (typically 4-11ms per rotation); Data Transfer Rate (typically 25-300MB/s). Data Block: The data transfer unit between disk and memory, typically 4-16KB, when a single item is needed its whole block is transferred, reading/writing of a disk block is one I/O operation; time for one I/O operation = seek time + rotational delay + transfer time (usually dominates the cost of processing a query). Magnetic Tape: mostly for offline backup, non-volatile, very low cost, only sequential access & slow access speed. Optical Disk: non-volatile, slow read/write speed compared to magnetic disks, usually write-once, read many disks for archive storage. Flash Memory: non-volatile, read/write in page units, erase (a page can only be written to after being erased, high latency), each cell has limited lifetime.

**Reliability & Efficiency** MTTF: mean time to failure, average time the disk is expected to run continuously without failure; Mean time to data loss: depends on MTTF & disk organization. Mirroring: Storing a redundant copy of data in another disks, increases mean time to data loss; read request handling rate is doubled, read speed is the same as in a single-disk system. Data Striping: partition data to several disks, faster reading with parallel reads (does not improve reliability). RAID: 0: striping, no mirroring; 10: striping + mirroring; 5: parity disk + striping (all disks share read request workload).

**File Organization** does not rely directly on the underlying operating system for file management, instead allocate one large file for DBMS. Each file is logically partitioned into fixed-length storage units called blocks. No record is larger than a block. Free list: for fixed-length records, store the address of the first deleted record in the file header, use the first record to store the address of the second deleted record, etc. Slotted page structure: for variable-length records. **Organization** Heap File: no ordering, can place anywhere, simple, new blocks are allocated/destroyed dynamically causing blocks to be scattered. Sequential File: Store records in sequential order based on the value of the search key, has to maintain the order during record insertion. Hashing: A hash function specifies where the record should be placed. Multitable Clustering: put more than one related relation in the same file to achieve faster joins.

**Inverted indexing** given a keyword, provides access to the list of documents that contain the keyword; mapper: input <docID, docContent>, output <keyword, docID>; reducer: input <keyword, list<docID>>, output <keyword, sorted list<docID>>. **Term frequency - inverse document frequency** a numerical statistic that is intended to reflect how important a term t (keyword) is to a document d in a collection. Preprocessing: Tokenization (Separate text into tokens), Stemming (Merge different forms of the same word/closely related words into a single stem), Stop word removal (Remove uninformative words from documents) **Suffix Tree** is prefix tree apparently, edge store characters, vertices store prefix occurrences.