Dense Index - index record for every search key value in the database
Sparse Index - not every search key has a record in the index, rather every block has a record

Primary Key - unique key, the data file is generally ordered on this key
Secondary Key - the file is not ordered on this key but there is an index record for the key
Super Key - key capable of uniquely identifying a row in the database
Composite Key - a combination of two or more columns in a table that can be used to uniquely identify each row in the table when the columns are combined uniqueness is guaranteed, but when it taken individually it does not guarantee uniqueness
Surrogate Key - is a unique identifier for either an entity in the modeled world or an object in the database
Foreign Key - identifier that links one table to another, usually the primary key of another table
Candidate Key - like a composite key, usually the minimal number of attributes
Alternate Key - a **key** associated with one or more columns whose values uniquely identify every row in the table, but which is not the primary **key**

Database - collection of data
DBMS - software system that allows for a set of services on a database
HDFS - Hadoop Distributed File System

Strong transactions are ACID
ACID (Atomicity, Consistency, Isolation, Durability)
Atomicity - transactions are a single unit, either completely succeed or fail
Consistency - all transactions are valid
Isolation - Concurrent execute and sequential execution end at the same result
Durability - once a transaction succeeds, it remains there/can't be lost

Weak Transaction - quorum consistency, distributed machine, updates are propagated over time, can read old values

Access Path - method to retrieve something from disk - more specifically for an index, the sequence of blocks needed to traverse through the index to the record

Round Robin Partitioning - basically doing modulus, should be even partitions and random if data is ~random, like dealing a deck of cards to players
Hash Partitioning - just hash - should be random and roughly the same size
Range Partitioning - partition based on ranges, $a < t_i < b$, for each partition

B(R) - #Blocks in relation R
T(R) - #Tuples/Records in relation R
V(R, attr) - #values of attribute attr in relation R

Types of Joins:

- Inner Join - only when entire condition is matched
- Left Outer Join - all from left table, nulls for right table columns if no match
- Right Outer Join - all from right table, nulls for left table columns if no match
- Full Outer Join - all from both tables, nulls for columns if no match

- Left Semi Join - only left table columns if match
- Left Anti Semi Join - only left table columns if no match