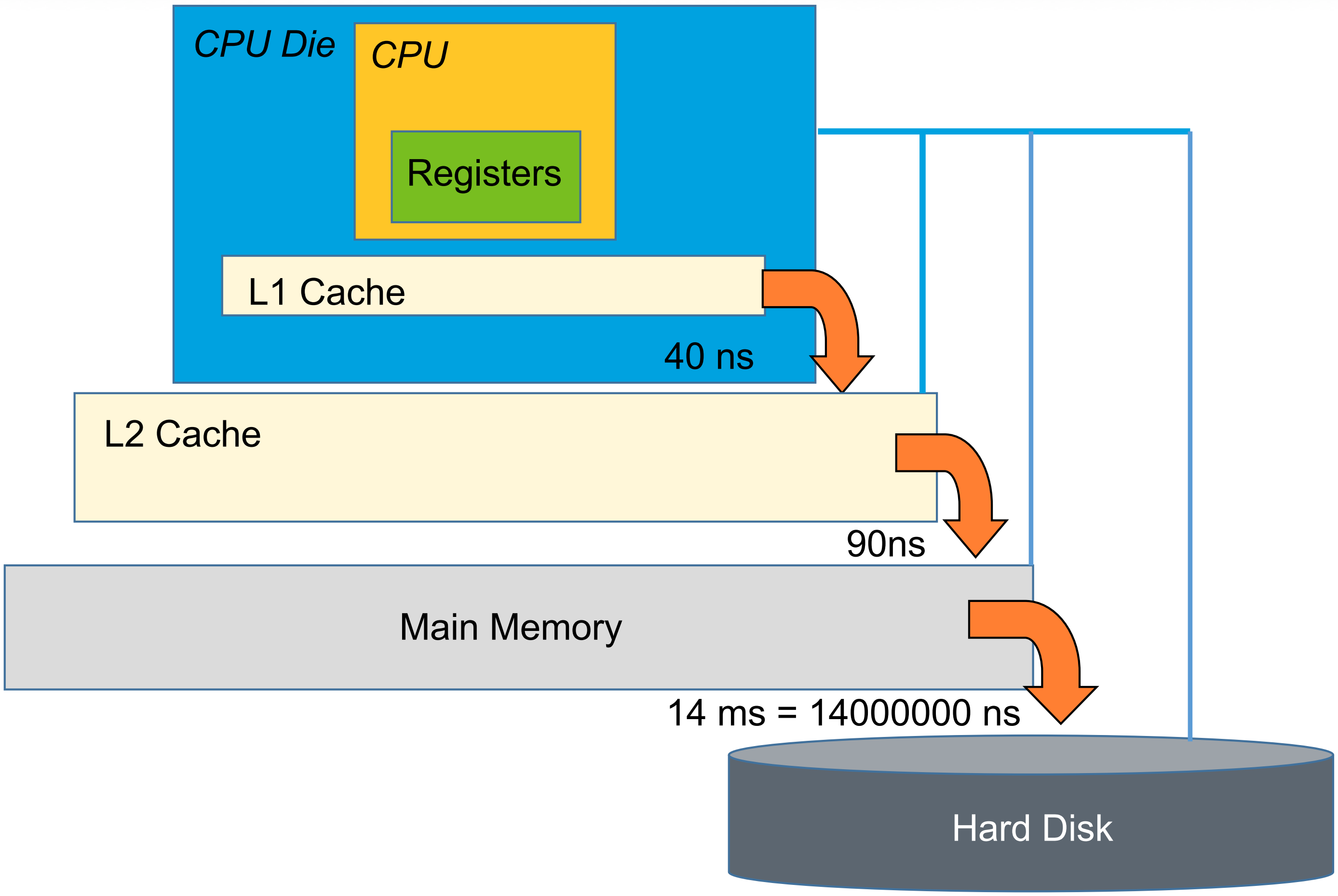
Fundamental Operators

Let **r** and **s** be relations with **schemas R and S**

union	$r \cup s = \{ t \mid t \in r \vee t \in s \}$
difference	$r - s = \{ t \mid t \in r \wedge t \notin s \}$
cartesian_product	$r \times s = \{ t \mid t = t_r\, t_s \text{ where } t_r \in r \wedge t_s \in s \}$
selection	$\sigma_P(r)$
projection	$\pi_A(r)$

The Memory Hierarchy

Example:
Intel PIII
CPU: 450MHz
Memory: 512MB



Internal Data Storage

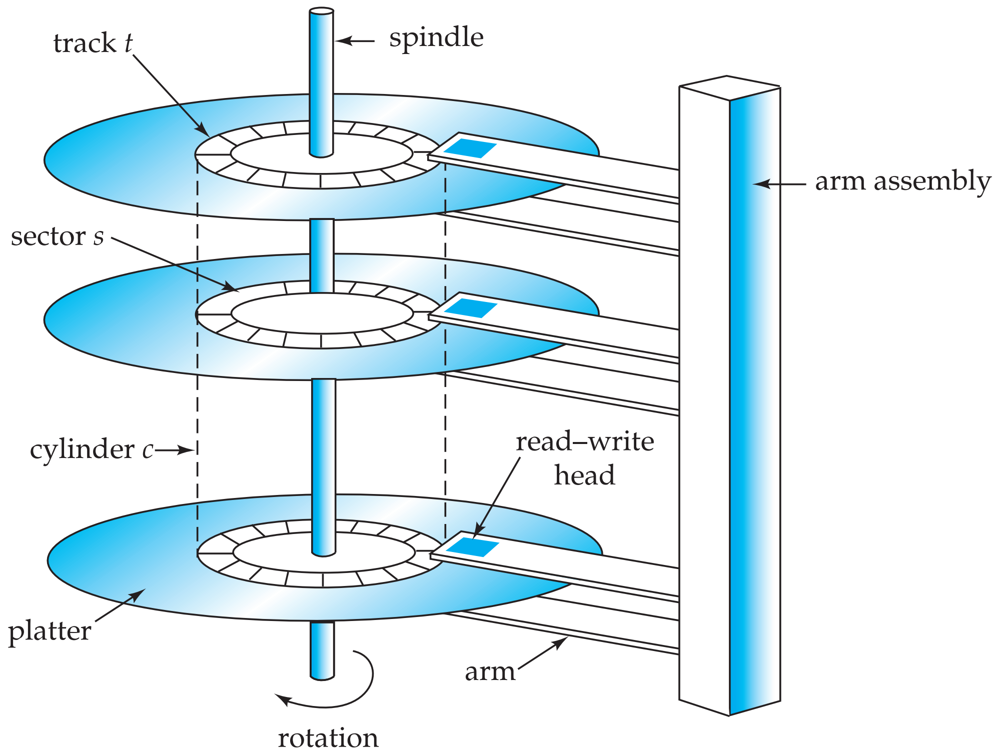
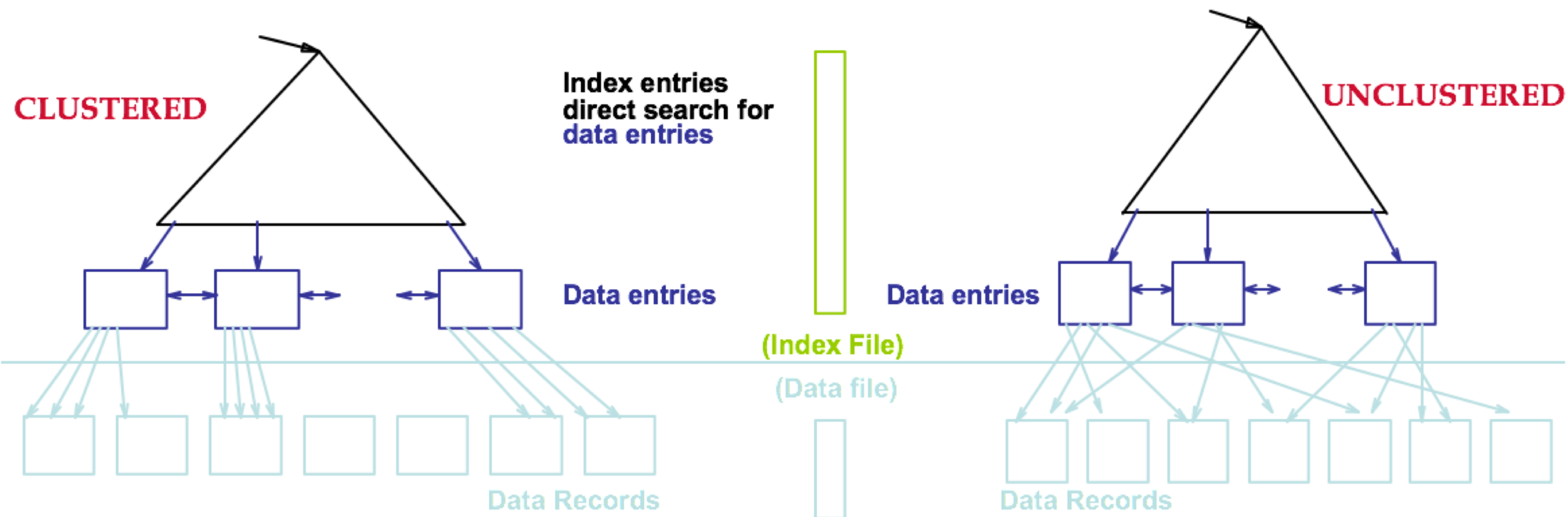
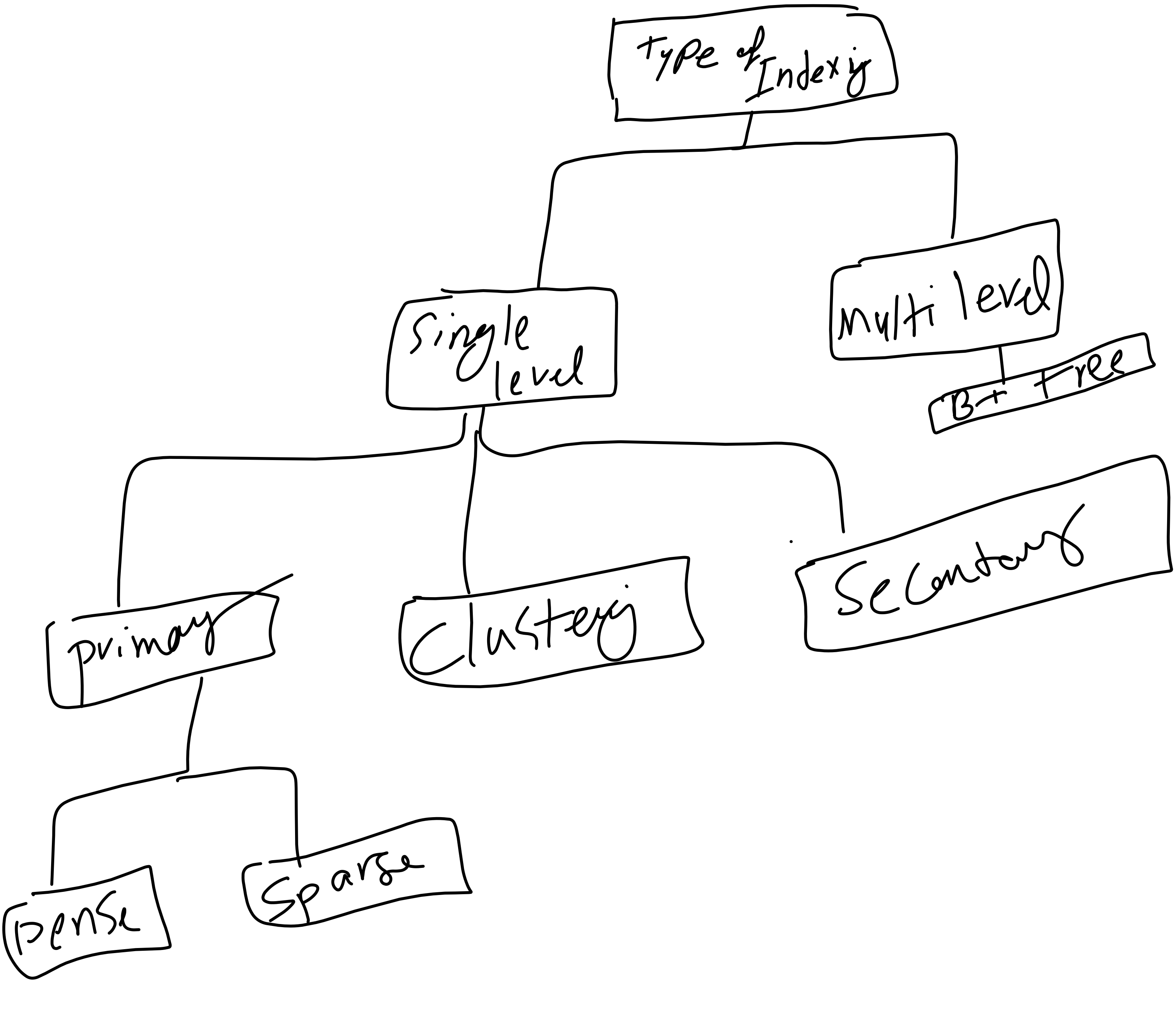


Image source: https://www.snia.org

Clustered vs. Unclustered Index



Types of Indexing



1-single level

a- Primary Indexing

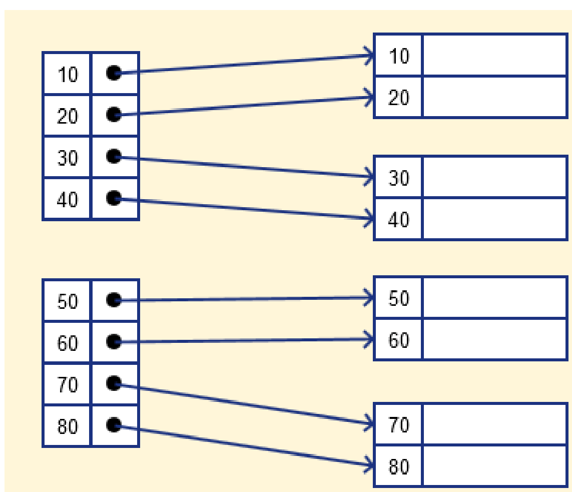
Primary Index is an ordered file which is fixed length size with two fields. The first field is the same a primary key and second, filed is pointed to that specific data block. In the primary Index, there is always one to one relationship between the entries in the index table.

The primary Indexing is also further divided into two types.

- Dense Index
- Sparse Index

a-1- Dense Index

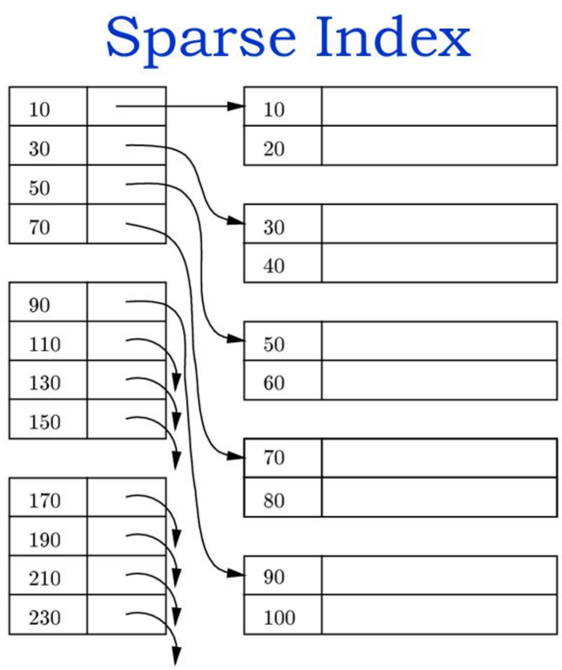
- The dense index contains an index record for every search key value in the data file. It makes searching faster.
- In this, the number of records in the index table is same as the number of records in the disk.
- It needs more space to store index record itself. The index records have the search key and a pointer to the actual record on the disk.



a-2-Sparse Index

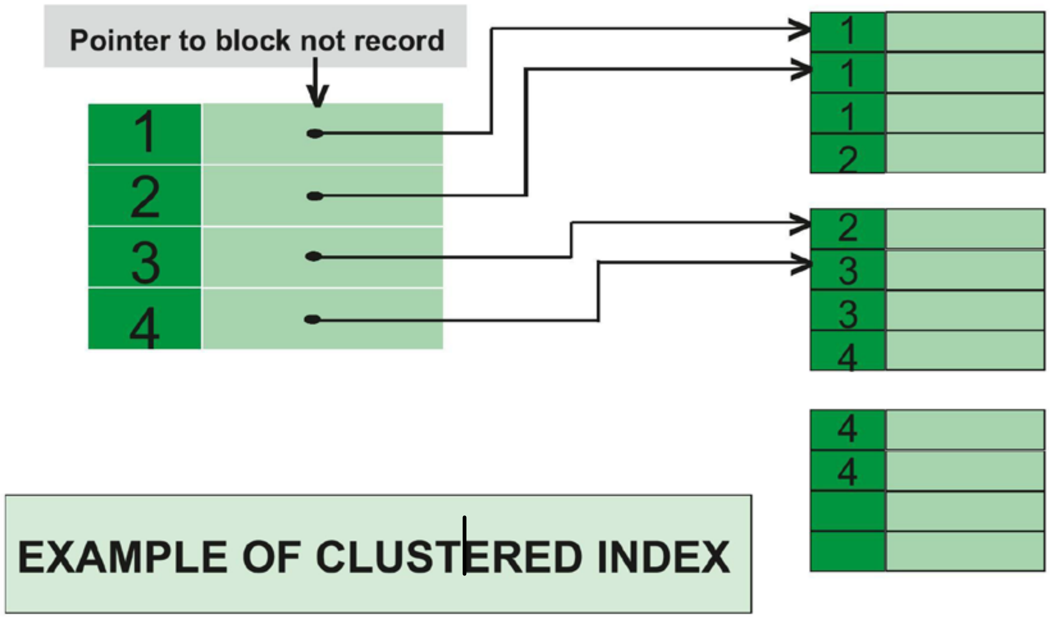
- In the data file, index record appears only for a few items. Each item points to a block.
- In this, instead of pointing to each record in the main table, the index points to the records in the main table in a gap.

Example of Sparse Index



b- Clustering Index

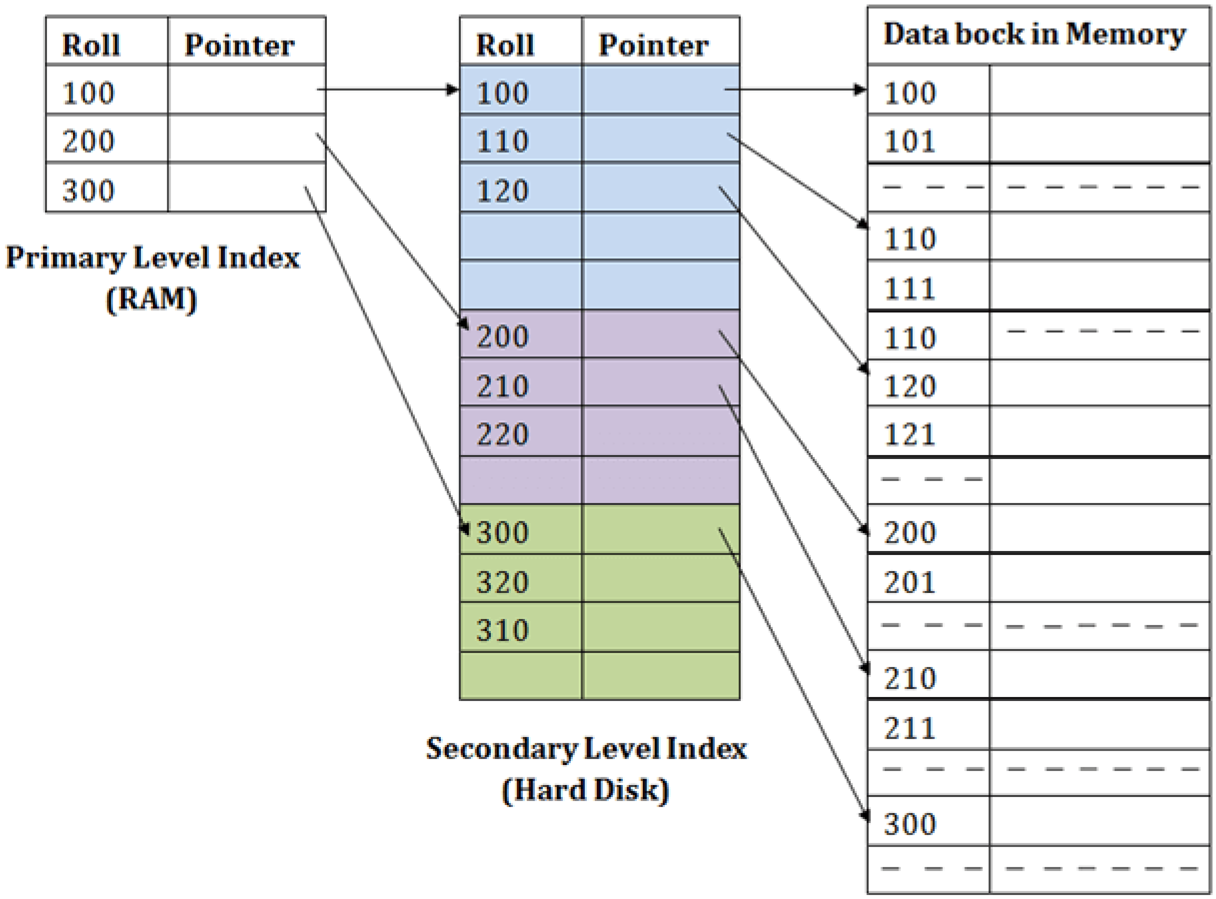
- A clustered index can be defined as an ordered data file. Sometimes the index is created on non-primary key columns which may not be unique for each record.



c- Secondary Index(non-clustering index)

In the sparse indexing, as the size of the table grows, the size of mapping also grows. These mappings are usually kept in the primary memory so that address fetch should be faster. Then the secondary memory searches the actual data based on the address got from mapping. If the mapping size grows then fetching the address itself becomes slower. In this case, the sparse index will not be efficient. To overcome this problem, secondary indexing is introduced.

In secondary indexing, to reduce the size of mapping, another level of indexing is introduced. In this method, the huge range for the columns is selected initially so that the mapping size of the first level becomes small. Then each range is further divided into smaller ranges. The mapping of the first level is stored in the primary memory, so that address fetch is faster. The mapping of the second level and actual data are stored in the secondary memory (hard disk).



2- Multi-level:

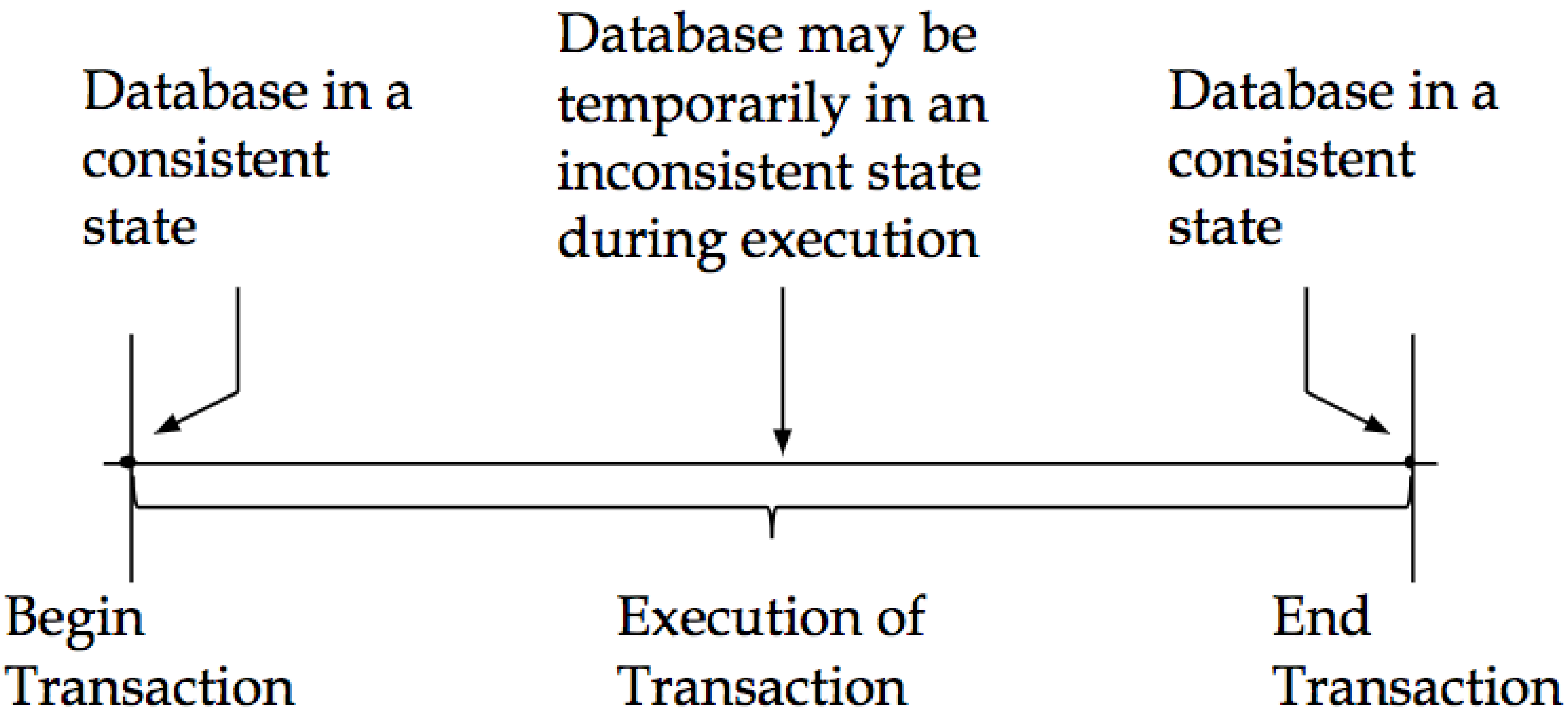
B+-tree

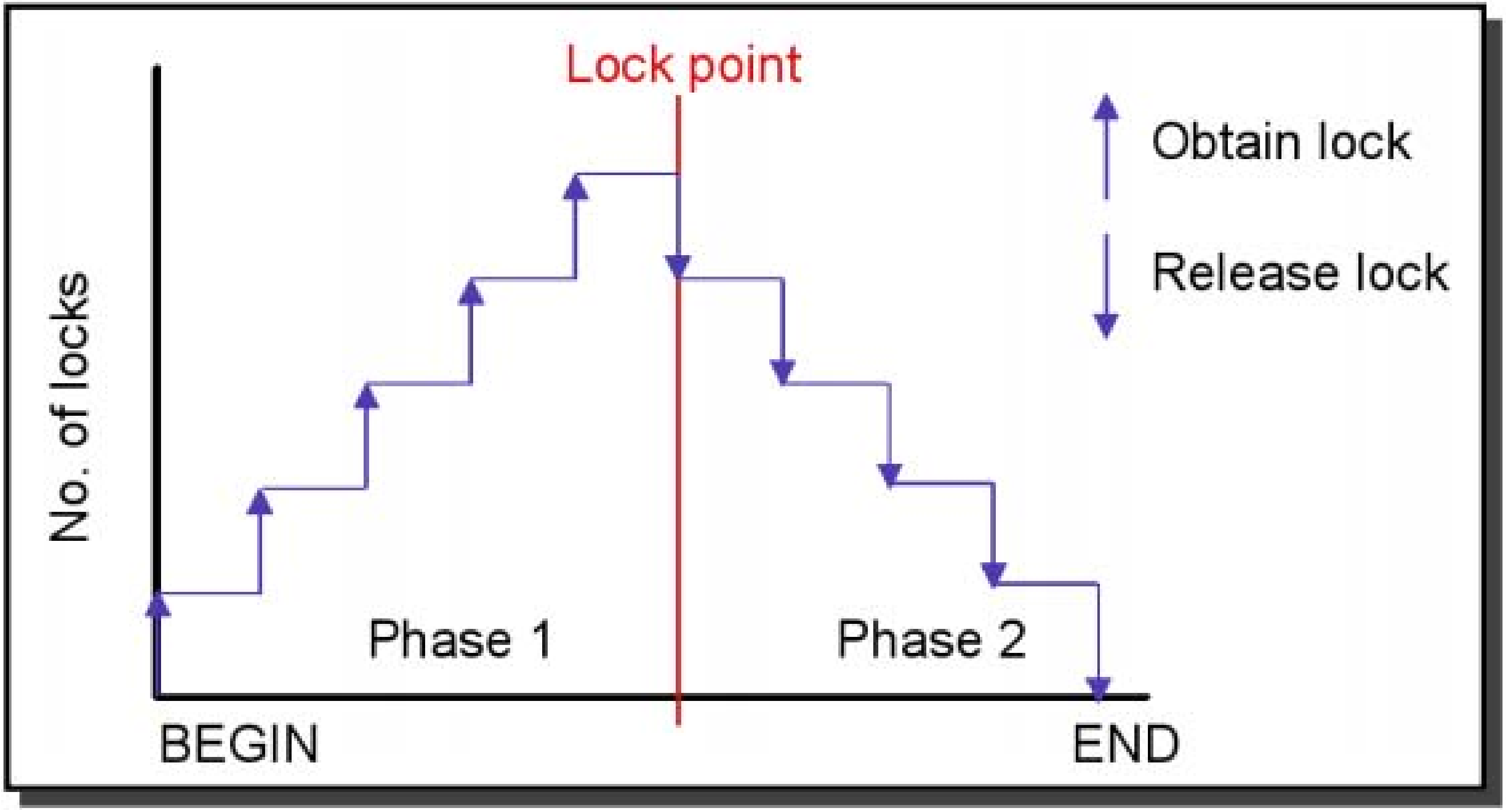
Transaction ACID Properties

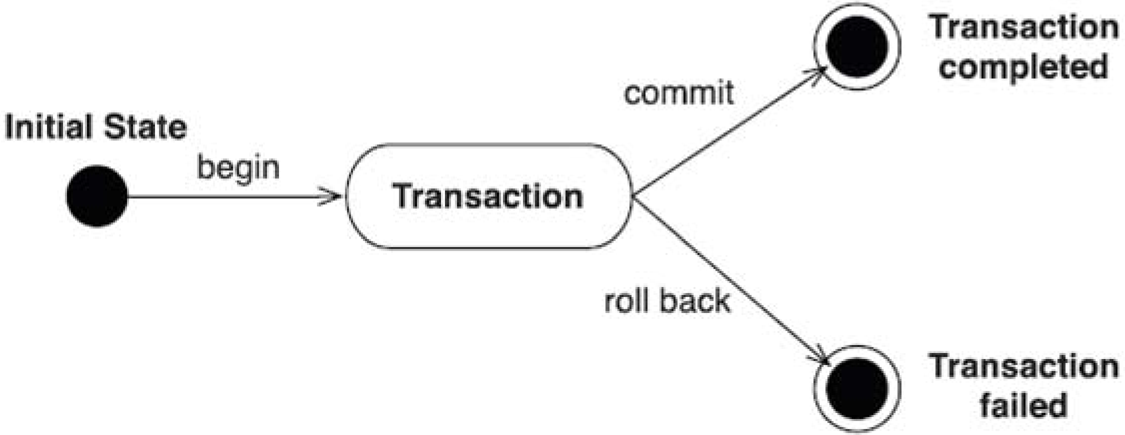
ACID

Atomic	**"ALL OR NOTHING"** Transaction cannot be subdivided
Consistent	Transaction ⟶ transform database from one consistent state to another consistent state
Isolated	Transactions execute independently of one another Database changes not revealed to users until after transaction has completed
Durable	Database changes are perminent The permanence of the database's consistent state

Transactions




Lock point
Obtain lock
Release lock
No. of locks
Phase 1
Phase 2
BEGIN
END


Initial State
begin
Transaction
commit
roll back
Transaction completed
Transaction failed

- **Wait-for graph** is a **graph** where:

    - **Node** represents a **transaction**

    - **Edge** $i \Rightarrow j$ represents the **fact** that:

        - The **transaction** i is **waiting** for a **lock** held by the **transaction** j

Example: