Q: Can Operating System direct operate on disk?
Ans: No.

1. Disks are much slower but CPU tasks are very fast and it can not keep waiting. Content from disk is brought into the memory and then application works on that data.
2. CPU makes the io call to the disk for bringing the data to memory. While the data is being copied, CPU does something else.

By default, tables are sorted by primary key.

STUDENTS

ID	NAME	BATCH_ID	PSP
1	A	3	20
2	B	3	30
3	C	2	40
4	D	4	50
5	E	1	70
6	F	2	80
.	.	.	.
.	.	.	.
.	.	.	.

```
SELECT * FROM STUDENTS WHERE BATCH_ID = 2;
```

Result = 20 rows

If our table has million rows, then it will take billion disk operations. This is where indexing comes into the picture. It reduces the number of disk operations in our system.

```
SELECT * FROM STUDENTS WHERE ID = 15;
```

1. Since primary key is sorted. Once we get the required data, we do not need to scan other rows.
2. But in worst case it can be 1 million disk fetches.

ID	REFERENCE
1	#1
2	#2
3	#3
4	#4
5	#5
6	#6
.	.
.	.

HashMap

After storing the row in database, we can store its reference in mapping table.

16 bytes x 1 million = 16mb

CASE1: No Sorting | No Mapping Table = 1 million

CASE2: ID Sorting | No Mapping Table = 15

CASE3: ID Sorting | Mapping Table = 1

STUDENTS

ID	NAME	BATCH_ID	PSP	PHONE
1	A	3	20	A1
2	B	3	30	B1
3	C	2	40	C1
4	D	4	50	D1
5	E	1	70	E1
6	F	2	80	F1
.	.	.	.	
.	.	.	.	
.	.	.	.	

SELECT * FROM STUDENTS WHERE PHONE = K;

ID	REFERENCE
A1	#1
B1	#2
C1	#3
D1	#4
E1	#5
F1	#6
.	.
.	.

SELECT *FROM STUDENTS WHERE PSP BETWEEN 10 AND 40;

PSP	PSP Range	Reference
0	[0, 10)	1 6 7 11
10	[10, 20)	10 9 8
20	[20, 30)	2 4 5
30	[30, 40)	31 63 32 45
40	[40, 50)	50 60 61 68
50	[50, 60)	.
60	[60, 70)	.
70	[70, 80)	.
80	[80, 90)	.
90	[90, 100)	.
100	100	.

Indexes

1. These mapping tables are called as indexes.
2. Indexes helps us to prevent unnecessary disk fetches.
3. Helps to load the query faster.

Should we create the indexes always?
No, it depends.

1. CUD causes indexes to update. Although indexes make the read faster but indexes make the write slower.
2. Indexes tables are stored in disk leads to increase in memory usage.

When should we use indexes?

1. Do not create indexes in the beginning.
2. Do performance testing.
3. Create indexes for queries that are used a lot and are slow.
4. Create indexes on patterns not on predictions.

```
USE SCALER;

CREATE TABLE IF NOT EXISTS EMPLOYEES(
      ID INT PRIMARY KEY,
    FIRSTNAME VARCHAR(255),
    LASTNAME VARCHAR(255),
    DEPARTMENT VARCHAR(255)
);

INSERT INTO EMPLOYEES VALUES(1, 'JOHN', 'DOE', 'HR');
INSERT INTO EMPLOYEES VALUES(2, 'ALICE', 'SMITH', 'IT');
INSERT INTO EMPLOYEES VALUES(3, 'BOB', 'JOHNSON', 'FINANCE');
INSERT INTO EMPLOYEES VALUES(4, 'EVA', 'WILLIAMS', 'TECH');

EXPLAIN SELECT * FROM EMPLOYEES WHERE DEPARTMENT = 'HR';
```

ID	SELECT_TYPE	TABLE	PARTITIONS	TYPE	POSSIBLE_KEYS	ROWS
1	SIMPLE	EMPLOYEES	NULL	ALL	NULL	4

```
CREATE INDEX IDX_EMPLOYEE_DEPARTMENT ON EMPLOYEES(DEPARTMENT);

EXPLAIN SELECT * FROM EMPLOYEES WHERE DEPARTMENT = 'HR';
```

ID	SELECT_TYPE	TABLE	PARTITIONS	TYPE	POSSIBLE_KEYS	ROWS
1	SIMPLE	EMPLOYEES	NULL	ALL	NULL	1