# Database Systems

## Digital Assignment 1

Slot: A2 + TA2 Course code: BCSE302L

1) Apply the following types of Join to the given relations using relational algebra and write its equivalent SQL Command.

→ Inner Joins:
- Theta join
- EQUI join
- Natural join

→ Outer Joins:
- Left Outer join
- Right Outer join
- Full Outer join

Customer

| Cid | Cname | Age |
|---|---|---|
| 101 | Ajay | 20 |
| 102 | Vijay | 19 |
| 103 | Sita | 21 |

Order

| Oid | Oname |
|---|---|
| 101 | Pizza |
| 101 | Noodles |
| 103 | Burger |

Inner join is used to return rows from both tables which satisfy the given condition.

- Theta join allows you to merge 2 tables based on the condition represented by theta.

Let's consider the condition "cid = Oid".

Relational Algebra: $R = Customer \bowtie_{(cid=Oid)} Order$

SQL Equivalent:
```
SELECT * FROM Customer
INNER JOIN
Order ON Customer.Cid = Order.Oid;
```

- EQUI join is a type of theta join where the join condition is based on equality (equi-predicate).

Here, we will use the condition "Cid = Oid" as well.

Relational Algebra: R = Customer ⋈(Cid=Oid) Order

SQL Equivalent: SELECT * FROM Customer
INNER JOIN
Order ON Customer.Cid = Order.Oid;

- Natural Join combines rows from two relations based on common attributes.

In this case, common attribute is "Cid".

Relational Algebra: R = Customer ⋈ Order

SQL Equivalent: SELECT * FROM Customer
NATURAL JOIN Order;

Outer join doesn't require each record in the two join tables to have a matching record. In this type of join, the table retains each record even if no other matching record exists.

Left Outer join returns all rows from the left relation and the matched rows from the right relation. If there is no match, NULL values are filled for the right relation's attributes.

Relational Algebra : $R = \text{Customer} \bowtie_{(Cid = Order)} \text{Order}$

SQL Equivalent : SELECT * FROM Customer

LEFT OUTER JOIN

Order ON Customer.Cid = Order.Oid;

Right Outer join returns all rows from right relation and the matched rows from the left relation. If there is no match, NULL values are filled for the left relation's attributes.

Relational Algebra : $R = \text{Customer} \bowtie_{(cid = Oid)} \text{Order}$

SQL Equivalent : SELECT * FROM Customer

RIGHT OUTER JOIN

Order ON Customer.Cid = Order.Oid;

Full Outer join returns all rows from both relations and fills NULL values where there are no matches.

Relational Algebra : $R = \text{Customer} \bowtie_{(cid = Oid)} \text{Order}$

SQL Equivalent : SELECT * FROM Customer

FULL OUTER JOIN

Order ON Customer.Cid = Order.Oid;

Tables resulting from each type of join using the given relations.

⇒ Theta join

| Cid | Cname | Age | Oid | Oname |
|---|---|---|---|---|
| 101 | Ajay | 20 | 101 | Pizza |
| 101 | Ajay | 20 | 101 | Noodles |
| 103 | sita | 21 | 103 | Burger |

⇒ Equi Join

| Cid | Cname | Age | Oid | Oname |
|---|---|---|---|---|
| 101 | Ajay | 20 | 101 | Pizza |
| 101 | Ajay | 20 | 101 | Noodles |
| 103 | sita | 21 | 103 | Burger |

⇒ Natural Join

| Cid | Cname | Age | Oid | Oname |
|---|---|---|---|---|
| 101 | Ajay | 20 | 101 | Pizza |
| 101 | Ajay | 20 | 101 | Noodles |
| 103 | sita | 21 | 103 | Burger |

⇒ Left Outer join

| Cid | Cname | Age | Oid | Oname |
|---|---|---|---|---|
| 101 | Ajay | 20 | 101 | Pizza |
| 101 | Ajay | 20 | 101 | Noodles |
| 102 | Vijay | 19 | NULL | NULL |
| 103 | sita | 21 | 103 | Burger |

⇒ Right Outer join

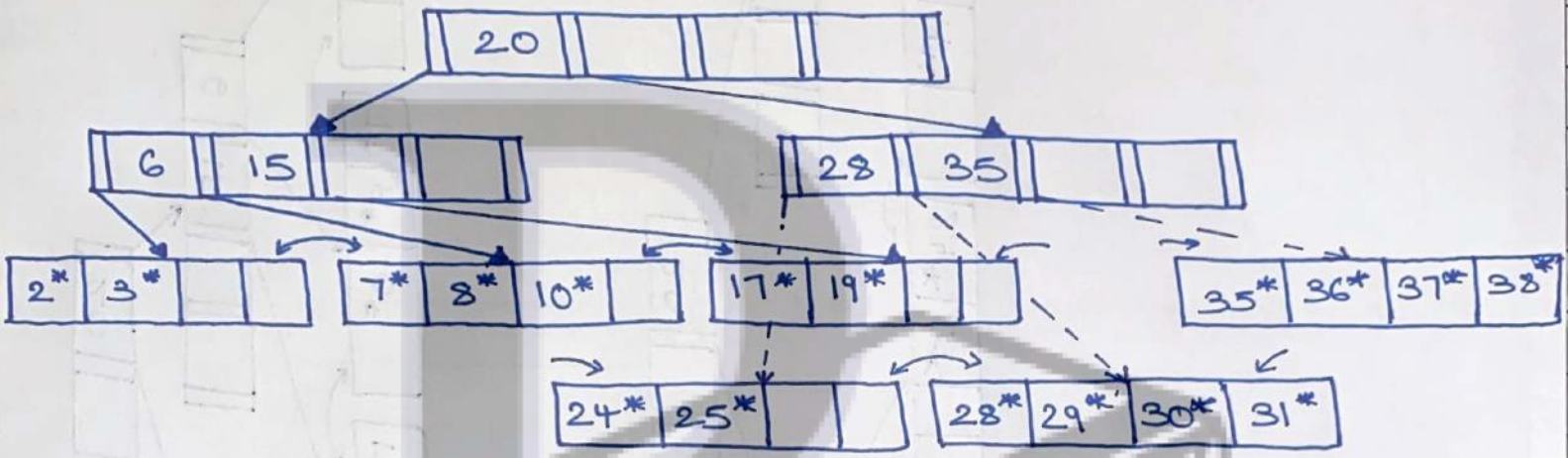
| Cid | Cname | Age | Oid | Oname |
|---|---|---|---|---|
| 101 | Ajay | 20 | 101 | Pizza |
| 101 | Ajay | 20 | 101 | Noodles |
| 103 | Sita | 21 | 103 | Burger |
| NULL | NULL | NULL | 101 | Pizza |
| NULL | NULL | NULL | 101 | Noodles |
| NULL | NULL | NULL | 103 | Burger |

⇒ Full Outer join

| cid | cname | Age | Oid | Oname |
|---|---|---|---|---|
| 101 | Ajay | 20 | 101 | Pizza |
| 101 | Ajay | 20 | 101 | Noodles |
| 102 | Vijay | 19 | NULL | NULL |
| 103 | Sita | 21 | 103 | Burger |
| NULL | NULL | NULL | 101 | Pizza |
| NULL | NULL | NULL | 101 | Noodles |
| NULL | NULL | NULL | 103 | Burger |

② Apply the B+ tree indexing to the following.

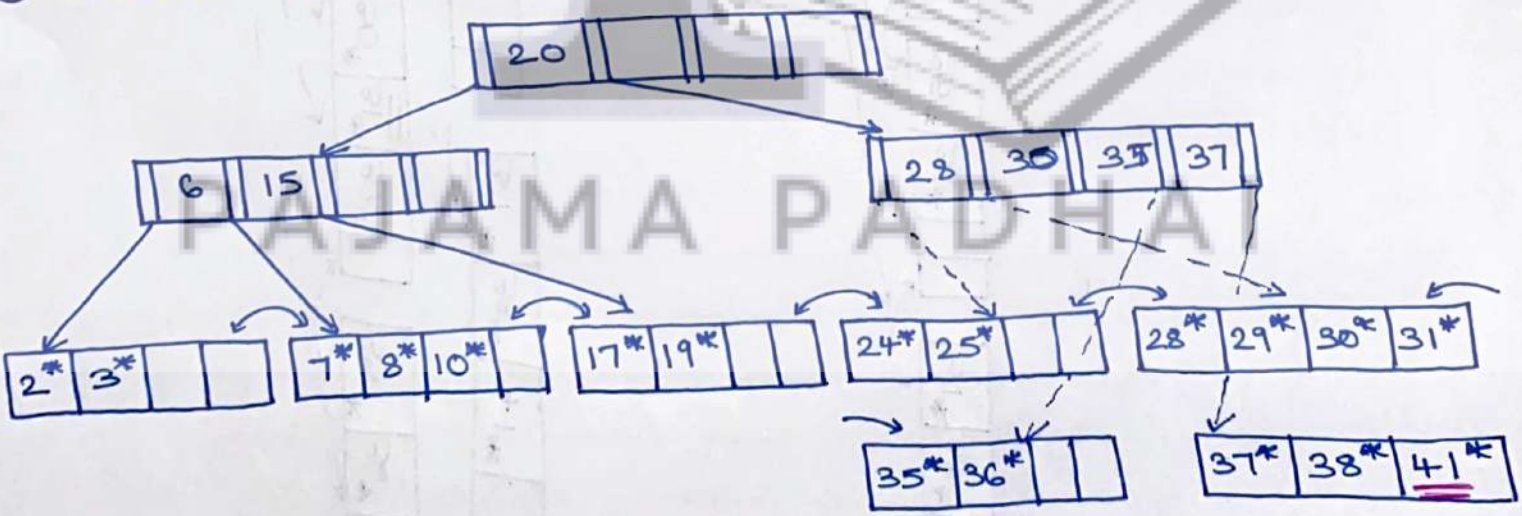
Using the following B+ tree index, answer the questions that follow.



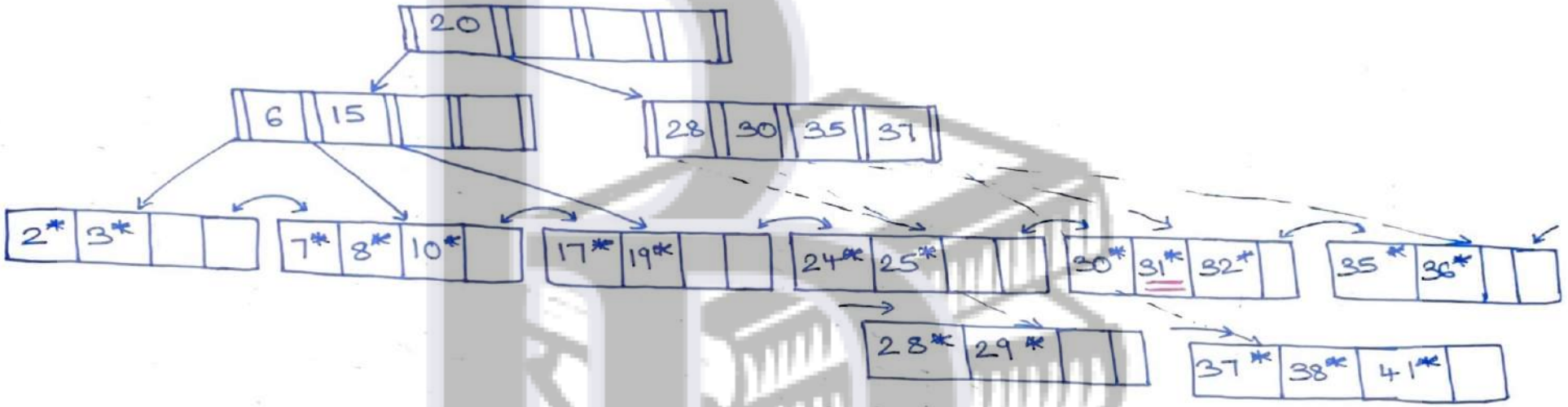

Show the resulting B+ Tree after each of these actions :-

1. Inserting data entry 41*
2. Inserting data entry 32*
3. Delete data entry 3*
4. Deleting data entry 2*
5. Inserting data entry 33*

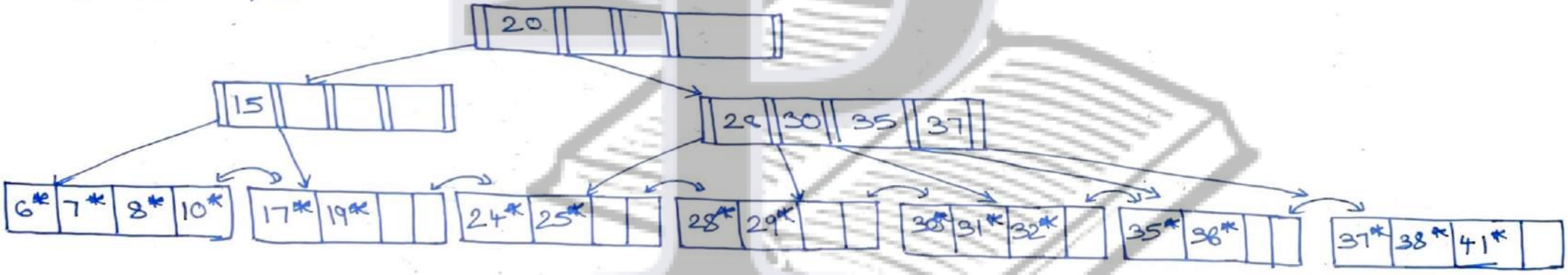
① Insert 41*



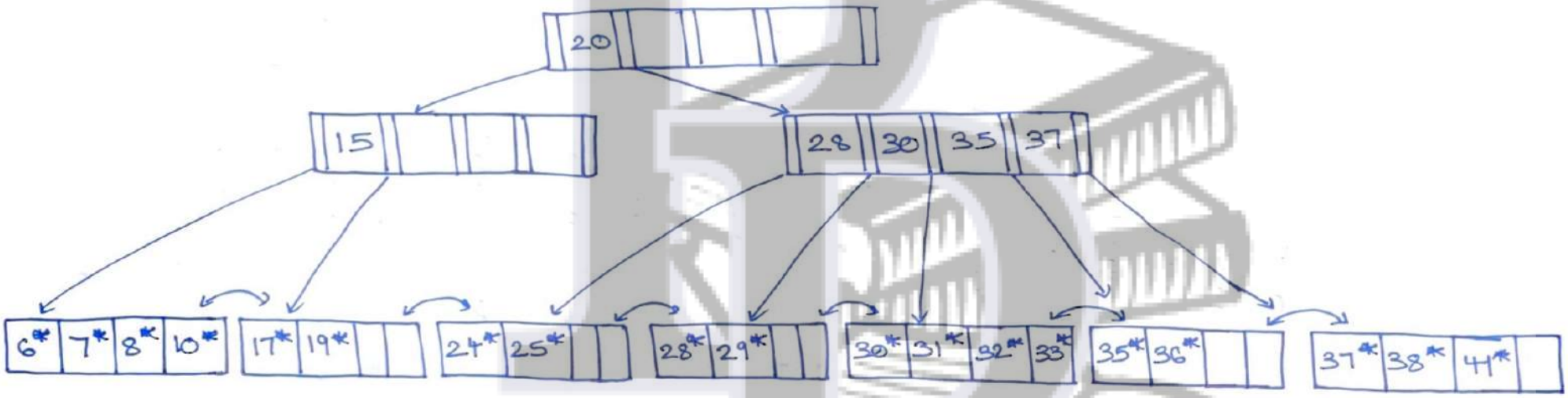
② Insert 32*




③,④ Delete 3*, 2*



⑤ Insert 33*




which is required B+ Tree after performing all operations.