UPDATE :

Syntax

```
UPDATE table_name
SET column1 = value1, column2 = value2, ...
WHERE condition;
```

Important Points About the `UPDATE` Statement

1. **Mandatory `SET` Clause**:
   - Use the `SET` clause to specify the columns you want to update and their new values.
2. **Conditional Update with `WHERE`**:
   - Use the `WHERE` clause to limit the rows that will be updated.
   - If you omit `WHERE`, **all rows in the table will be updated**.
3. **Updating Multiple Columns**:
   - You can update multiple columns in a single `UPDATE` statement by separating them with commas.
4. **Expressions in Updates**:
   - You can use expressions, calculations, or subqueries to set the values of the columns.
5. **Combining with Transactions**:
   - Use `BEGIN TRANSACTION` and `COMMIT` / `ROLLBACK` to ensure the update is reversible if something goes wrong.

Examples of `UPDATE` Statement

1. Updating a Single Column

Update the hire date for a specific employee:

```
UPDATE employees
SET hire_date = '2023-01-01'
WHERE emp_no = 1001;
```

Explanation:

- Updates the `hire_date` for the employee with `emp_no = 1001`.

---

2. Updating Multiple Columns

Update both the `first_name` and `last_name` of an employee:

```
UPDATE employees
SET first_name = 'Alice', last_name = 'Johnson'
WHERE emp_no = 1002;
```

Explanation:

- Updates `first_name` to "Alice" and `last_name` to "Johnson" for the employee with `emp_no = 1002`.

---

3. Updating Multiple Rows

Increase the salary of all employees hired before 2020:

```
UPDATE employees
SET salary = salary + 5000
```

```
WHERE hire_date < '2020-01-01';
```

Explanation:

- Increases the `salary` of all employees hired before January 1, 2020, by 5000.

---

4. Updating Without `WHERE` Clause

Update the gender of all employees to 'M':

```
UPDATE employees
SET gender = 'M';
```

Warning:

- **Without the `WHERE` clause, all rows will be updated.**
- Use with caution unless the intention is to update every row.

Rolling Back an Update

If you make a mistake during an update, you can revert it using a transaction:

```
BEGIN TRANSACTION;

UPDATE employees
SET salary = salary - 1000
WHERE emp_no = 1001;

-- If satisfied, commit the change
COMMIT;

-- If an error occurred, rollback the change
```

```
ROLLBACK;
```

Best Practices

1. Always **back up the table** or use transactions for critical updates.
2. Use the `WHERE` clause carefully to avoid updating unintended rows.
3. Test the query using a `SELECT` statement before running the actual `UPDATE`.
4. When updating large datasets, test the impact in a development or staging environment first.
5. Use subqueries for complex updates, but optimize for performance.