BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE
PILANI, RAJASTHAN, 333031

Advanced Database Systems
[CS G516]

WORKING WITH DATES

Prerequisites

1. Basic understanding of relational database concepts (relations, tuples, attributes)
2. Familiarity with SQL `SELECT`, `INSERT`, and `WHERE` clauses
3. Access to MySQL Server 8.0 or higher
4. MySQL client (MySQL Shell / MySQL Workbench)

Introduction

Temporal data plays a critical role in database systems for logging, auditing, scheduling, and transactional consistency. Earlier database systems often stored date and time together, leading to precision issues, index misuse, and incorrect comparisons. Modern DBMSs, including MySQL, provide specialized temporal data types to overcome these limitations.

MySQL supports the following core temporal data types:

- `DATE` – Stores only calendar date
- `TIME` – Stores only time of day
- `DATETIME` – Stores date and time without timezone
- `TIMESTAMP` – Stores date and time with UTC conversion

Unlike SQL Server, MySQL does not have a native `DATETIMEOFFSET` type. Time zone handling is achieved using `TIMESTAMP` and conversion functions such as `CONVERT_TZ()`.

MySQL Syntax and Implementation

This section demonstrates how MySQL implements temporal data types and functions. Each code block is accompanied by conceptual explanations explaining *why* the syntax is used and *how* MySQL processes temporal values internally.

1. DATE Data Type

The `DATE` data type stores only the calendar date in `YYYY-MM-DD` format. No time component is stored internally, which makes it ideal for date-based comparisons and indexing.

```
-- Assign the current system date to a session variable
-- CURRENT_DATE() is evaluated at query execution time
SET @dt = CURRENT_DATE();

-- Display the stored value
SELECT @dt AS only_date;
```

Explanation:

- MySQL stores `DATE` values as compact integers internally, making comparisons efficient.
- Since there is no hidden time component, equality comparisons (=) work reliably.
- Indexes on DATE columns are fully usable without function-based filtering.

2. TIME Data Type

The `TIME` data type represents time of day or elapsed time without any associated date.

```
-- CURRENT_TIME() returns the current session time
SET @tm = CURRENT_TIME();

SELECT @tm AS only_time;
```

Explanation:

- `TIME` can represent both clock time and durations.
- MySQL allows negative and large values to support interval calculations.
- Suitable for login times, shift durations, or process runtimes.

3. DATETIME Data Type

`DATETIME` stores both date and time together, but without time zone awareness.

```
-- NOW() returns the current date and time
SET @dtm = NOW();

SELECT @dtm AS datetime_value;
```

Explanation:

- The value is stored exactly as inserted, independent of server or session time zone.
- Recommended for applications where local time consistency is required.
- Comparisons are straightforward, but global time synchronization is not handled automatically.

4. TIMESTAMP Data Type (Time Zone Aware)

Unlike `DATETIME`, `TIMESTAMP` values are stored internally in UTC.

```
SET @ts = CURRENT_TIMESTAMP();

SELECT @ts AS timestamp_value;
```

Explanation:

- On insertion, MySQL converts the value from session time zone to UTC.
- On retrieval, it converts UTC back to the session time zone.
- This behavior makes `TIMESTAMP` ideal for distributed and cloud-based systems.

5. Fractional Seconds Precision

MySQL allows precision up to microseconds (6 digits).

```
CREATE TABLE event_log (
        event_time DATETIME(6)
);

INSERT INTO event_log VALUES (NOW(6));

SELECT * FROM event_log;
```

Explanation:

- Fractional precision is essential for logging, auditing, and high-frequency events.
- Precision is defined at schema level and enforced consistently.
- Higher precision increases storage size slightly but improves temporal accuracy.

6. Time Zone Conversion

MySQL replaces SQL Server's `DATETIMEOFFSET` using explicit conversion functions.

```
-- Convert UTC time to Indian Standard Time (IST)
SELECT CONVERT_TZ(UTC_TIMESTAMP(), '+00:00', '+05:30') AS ist_time;
```

Explanation:

- `UTC_TIMESTAMP()` ensures a globally consistent reference time.
- `CONVERT_TZ()` adjusts the value without altering the original instant.
- Named zones (e.g., 'Asia/Kolkata') require time zone tables to be loaded.

7. System Date and Time Functions

These functions return values based on current session context.

```
SELECT
NOW() AS current_datetime,
CURRENT_DATE() AS current_date,
CURRENT_TIME() AS current_time,
UTC_TIMESTAMP() AS utc_datetime;
```

Explanation:

- `NOW()` is evaluated once per statement.
- Session time zone directly affects output of non-UTC functions.
- Useful for transactional timestamps and audit columns.

8. Extracting Date and Time Components

Extraction functions separate logical components from composite values.

```
SELECT
DATE(NOW()) AS extracted_date,
TIME(NOW()) AS extracted_time;
```

Explanation:

- Avoid using functions on indexed columns inside `WHERE` clauses.
- Prefer range queries when filtering large relations.

9. Combining DATE and TIME Values

MySQL does not allow arithmetic addition of `DATE` and `TIME`.

```
SET @d = CURRENT_DATE();

SET @t = CURRENT_TIME();

SELECT TIMESTAMP(@d, @t) AS combined_datetime;
```

Explanation:

- `TIMESTAMP(date, time)` constructs a valid temporal value explicitly.
- Prevents type ambiguity and preserves internal consistency.
- Recommended over implicit casts or arithmetic hacks.

Exercise

Create the following tables

1. **SAILORS(sid:**numeric**,s_name:**varchar2**,rating:**numeric**,age:**numeric**)**
2. **BOATS(bid:**numeric**,boat_name:**varchar2**, colour:**varchar2**)**
3. **RESERVES(sid:**numeric**,bid:**numeric**,day:**date)

Enter records from Lab2.1.sql in these tables and maintain all the foreign key constraints. Solve the following problems.

1. Find the names of the sailors who have duty on '23-JAN-2018'.
2. Find the name of the boat on which the sailor with SID = 121 was reserved for duty on 25-NOV-2017.
3. Find the names of the sailors and boats that have been reserved for duty on the last day of the current month.
4. Find the number of months between today and the next scheduled reservation of the sailor with SID = 911.
5. Find the names of sailors who have been reserved for duty on a red-coloured boat on the first day of the next month.
6. Find the total number of reservations made for each boat and display only those boats that have been reserved more than 5 times.
7. Find the sailors who have never been reserved for any duty.
8. Find the sailor(s) with the maximum number of duty reservations.
9. Find the boats that were reserved at least once in the previous month but not reserved in the current month.
10. Find the average age of sailors assigned to each boat and display only those boats where the average age is greater than 30 years.
11. Find the details of sailors whose next duty date is scheduled within the next 7 days.

12. Find the sailors who have been assigned to more than one distinct boat on different duty dates.
13. Find the month-wise count of reservations for the current year, sorted in descending order of workload.
14. Find the sailors who have consecutive duty assignments (two duties on consecutive calendar days).
15. Find the boats that were reserved on the same day by more than one sailor, indicating potential operational overlap.

References

1. **Date and Time Functions (MySQL – Official Reference)**
   https://dev.mysql.com/doc/refman/8.4/en/date-and-time-functions.html
2. **DATE, DATETIME, TIMESTAMP Data Types (MySQL)**
   https://dev.mysql.com/doc/refman/8.4/en/datetime.html
3. **MySQL Documentation Search – Date Related Topics**
   https://dev.mysql.com/doc/search/?d=12&p=1&q=date
4. **SQL Date Functions (General SQL – Examples & Explanation)**
   https://www.geeksforgeeks.org/sql/sql-date-functions/
5. **MySQL Date Functions (Beginner-Friendly Tutorial)**
   https://www.w3schools.com/mysql/mysql_dates.asp