242466

DBMS

[EXPERIMENT - 7]

DATE:

AIM TO STUDY STRING AND DATE FUNCTIONS IN "MYSQL"

[STRING FUNCTIONS]

- CONCAT => CONCAT() FUNCTIONS ADDS TWO OR MORE STRINGS TOGETHER
  SYN => CONCAT (s1, s2 .... sN)
  Eg => SELECT CONCAT ("SQL", "DATA")

- SUBSTRING => REMOVES A PART OF A STRING
  SYN => SUBSTRING (s, start, length)
  Eg => SELECT SUBSTRING (emp_name, 2, 8)

- LENGTH => RETURNS LENGTH OF STRING
  SYN => LENGTH (string)
  Eg => SELECT LENGTH ("SQL DATA")

- UPPER => CONVERTS STRING TO UPPERCASE
  SYN => UPPER (string)
  Eg => SELECT UPPER (emp_name)

- LOWER => CONVERTS STRING TO LOWERCASE
  SYN => LOWER (string)
  Eg => SELECT LOWER (emp_name)

- REPLACE => REPLACES ALL OCCURENCES OF A SUBSTRING WITHIN A SUBSTRING, IT IS CASE-SENSITIVE
  SYN => REPLACE (string, old, new)
  Eg => SELECT REPLACE ("SQL DATA", "SQL", "LOL")

DATE:

DATE FUNCTIONS

- CURDATE => RETURNS CURRENT DATE
  SYN => CURDATE()
  Eg => SELECT CURDATE()

- NOW => RETURNS CURRENT DATE & TIME, IS IN "YYYY-MM-DD HH-MM-SS"
  SYN => NOW()
  Eg => SELECT NOW()

- DATE_ADD => ADDS A TIME/DATE INTERVAL TO A DATE AND RETURNS DATE
  SYN => DATE_ADD (int, num, date)
  Eg => SELECT DATE_ADD (year, 1, '2017/19/20')

- DATE_DIFF => RETURNS DIFFERENCE BETWEEN TWO DATES AS AN INTEGER
  SYN => DATE_DIFF(int, d1, d2)
  Eg => DATE_DIFF (year, "2017/02/20", "2017/04/05")

- MONTH => RETURNS MONTH FROM A DATE
  SYN => MONTH (date)
  Eg => SELECT MONTH ("2017/06/18")

- YEAR => RETURNS YEAR FROM A DATE
  SYN => YEAR (date)
  Eg => SELECT YEAR ("2017/08/20")

EXPERIMENT - 8

DATE:

TO STUDY TRIGGERS IN "MYSQL"

WHAT IS A TRIGGER

- A DATABASE OBJECT THAT IS AUTOMATICALLY CALLED OR EXECUTED OR FIRED WHEN CERTAIN EVENTS OCCUR
- THESE CAN BE USED TO AUTOMATICALLY UPDATE TIMESTAMPS ON ROWS WHEN THEY GET MODIFIED.
- ENFORCING DATA INTEGRITY BY VALIDATING BEFORE INSERTING OR UPDATING
- TYPES:
  - BEFORE => EXECUTED BEFORE TRIGGER EVENT
  - AFTER => EXECUTED AFTER TRIGGER EVENT

SYNTAX

```
DELIMITER //

CREATE TRIGGER trigger_name
{BEFORE | AFTER}
{INSERT | UPDATE | DELETE} ON table_name
FOR EACH ROW
BEGIN
        --// TRIGGER LOGIC HERE
END //

DELIMITER ;
```

DATE:

```
CREATE TABLE emp-log (

    emp_no  INT,
    emp_name  VARCHAR(50),
    dept_no  INT,
    job  VARCHAR (50),
    hire_date  DATE,
    salary  INT,
    mgr  INT,
    comm  INT,
    log_date  TIMESTAMP DEFAULT current_timestam

)
```

```
CREATE OR REPLACE TRIGGER try
AFTER INSERT ON emp IS
FOR EACH ROW
BEGIN
INSERT INTO emp_log (

    emp_no, emp_name, dept_no,
    job, hire_date, salary, mgr, comm

) VALUES (

    :NEW.emp_no, :NEW.emp_name, :NEW.dept_no,
    :NEW.job, :NEW.salary, :NEW.salary,
    :NEW.mgr, :NEW. comm

);
END;
```

DATE:

```
INSERT INTO emp VALUES (
        7786,
        "adam",
        20,
        "Analyst",
        "19-Apr-87",
        100,
        7560,
        null
);
```

OUTPUT

```
SELECT * FROM emp_log;
```

no	emp_name	dept_no	job	hire_date	sal	mgr	Comm
86	adam	20	Analyst	19-Apr-87	7560		

NXT Pg →

DATE:

```
CREATE TRIGGER emp_sun
BEFORE INSERT OR UPDATE OR DELETE
ON EMPLOYEE
FOR EACH ROW
BEGIN
IF RTRIM(UPPER(TO_CHAR(SYSDATE,"DAY")))
        = "SUNDAY"
THEN RAISE_APPLICATION_ERROR(-20002,
        "NO OPERATION DONE ON SUNDAY");
END IF;
END;

INSERT INTO emp VALUES(
4657,
"XYZ",
"CLERK",
7562,
"23-SEP-2024",
1000,
"    ",
20
);
```

OUTPUT

```
ORA-20002: NO OPERATION DONE ON SUNDAY
```

DBMS

EXPERIMENT - 9

DATE:

TO STUDY CURSOR IN "MYSQL"

WHAT IS A CURSOR

- A CURSOR IS A DATABASE OBJECT THAT ALLOWS YOU TO RETRIEVE, MANIPULATE THROUGH A SET OF ROWS ONE AT A TIME RETURNED BY A QUERY
- CURSORS ARE PARTICULARLY USEFUL WHEN YOU NEED TO PROCESS MULTIPLE ROWS ONE AT A TIME AND PERFORM OPERATIONS ON EACH ROW ONE AT A TIME

TYPES OF CURSOR

1. IMPLICIT:
   - AUTOMATICALLY CREATED WHEN A QUERY IS EXECUTED. YOU DO NOT NEED TO DECLARE OR MANAGE THEM

2. EXPLICIT:
   - DEFINED BY USER FOR BETTER CONTROL OVER ROW PROCESSING. YOU NEED TO DECLARE, OPEN, FETCH AND CLOSE THEM
   - DECLARE CURSOR
   - OPEN CURSOR
   - FETCH ROWS
   - CLOSE THE CURSOR

SYNTAX

```
DECLARE c_name CURSOR
FOR select_statement;

DECLARE CONTINUE HANDLER FOR NOT
FOUND SET done = TRUE;

OPEN c_name;

FETCH c_name INTO var_list;

CLOSE c_name;
```

- DEFINE CURSOR WITH A SQL SELECT STATEMENT
- OPEN INITIALIZERS THE CURSOR AND PREPARE IT FOR ROW FETCH
- FETCH RETRIEVE ONE ROW AT A TIME
- FINALLY, RELEASE THE CURSOR

NXT Pg →

DATE:

```
DECLARE CURSOR c_emp IS
    SELECT emp_no, salary FROM emp
    WHERE dept_no = 30;
BEGIN
    FOR rec IN c_emp LOOP
        UPDATE emp
        SET salary = (rec.salary * 0.5) + rec.salary;
        WHERE emp_no = rec.emp_no;
    END LOOP
COMMIT;
END;

SELECT * FROM emp;
```

emp_no	e_name	job	mgr	H-D	sal	comm	dept_no
7441	ALLEN	S	731	10/2	16K	300	20
7524	MILE	M	641	8/9	20K	500	30
7653	JOHN	S	731	3/12	15K	1000	10
7561	NICK	A	642	4/6	18K	800	20

OLD SALARY ORDER => 11500K
16000K
7500K
9000K

DATE:

[OLD TABLE]

Name	Gender	rollno
Nick	M	47
Aaisha	F	85
Sana	F	61

```
DECLARE CURSOR c_stud IS
    SELECT * FROM student
    WHERE name = "Nick"
BEGIN
    FOR rec in c_stud LOOP
    UPDATE student SET
    rec.name = "Joh"
    WHERE name = rec.name;
    END LOOP;
COMMIT;
END;
```

[NEW TABLE]

Name	Gender	roll_no
John	M	47
Aaisha	F	85
Sana	F	61

DATE:

```
DECLARE
    c_id    customer.id %type;
    c_name  customer. name %type;
    CURSOR  c_customer  IS
    SELECT  id, name,  FROM  customer;
BEGIN
    OPEN  c_customer;
    LOOP
    FETCH  c_customer  INTO  c_id, c_name;
    EXIT  WHEN  c_customer %NOTFOUND
    dbms_output.put_line (
        c_id || ' ' || c_name
    );
    END  LOOP;
    CLOSE  c_customer;
END;
```

DBMS

[EXPERIMENT - 10]

DATE:

TO STUDY [JDBC] IN "MYSQL"

[STEPS]

1) CREATE A NEW FOLDER AND OPEN IT IN "VS CODE"

2) IN "VS CODE", CREATE A NEW JAVA PROJECT VIA THE COMMAND PALLETE.

3) CHOOSE "JAVA PROJECT"

4) SELECT "NO BUILD TOOLS"

5) NAME THE PROJECT AND CLICK CREATE

6) A NEW WINDOW WILL OPEN CONTAINING "lib" AND "src" FOLDER.

7) OPEN "App.java" IN "src" FOLDER

8) DOWNLOAD "my-sql-connector.jar" FILE

9) INCLUDE THE ".jar" FILE IN THE "lib" FOLDER.

10) MODIFY THE CLASSPATH OF THE CURRENT PROJECT VIA THE COMMAND PALETTE.

11) TYPE "Java: CONFIGURE CLASSPATH"

DATE:

SELECT CURRENT PROJECT AND ADD IT TO LIBRARY

SELECT LOCATION OF ".jar" TO INCLUDE

IF ".jar" IS PROPERLY ADDED IT WILL BE ADDED TO THE CLASSPATH.

TYPE YOUR "JDBC" CODE IN "App.java" AND CONNECT IT TO "MySQL".

[JDBC]

CONNECTION:

⟹ Connection conn = DriverManager.getConnection(
String url,
String user,
String pass
);

url ⟶ DATABASE LINK
user ⟶ DATABASE USER NAME
pass ⟶ DATABASE USER PASSWORD

(2) CREATING SQL STATEMENTS:

⟹ Statement s = conn.createStatement()

(3) EXECUTING SQL QUERIES:

⟹ ResultSet rs = s.executeQuery(string query);