MySQL - RDBMS

Agenda

- Cursors
- Functions
- Triggers
- Mongo

Cursor

- T1 (C1) --> 1, 2, 3, 4
- T2 (C2) --> 10, 22, 35, 46

```
DECLARE v_cur1 CURSOR FOR SELECT c1 FROM t1;
DECLARE v_cur2 CURSOR FOR SELECT c2 FROM t2;

OPEN v_cur1;
OPEN v_cur2;

SET v_i = 1;

again: LOOP
    FETCH v_cur1 INTO v1;
    IF v_flag = 1 THEN
        LEAVE again;
    END IF;

    FETCH v_cur2 INTO v2;
    IF v_flag = 1 THEN
        LEAVE again;
    END IF;

    INSERT INTO result VALUES (v_i, CONCAT(v1, ' - ', v2));
    SET v_i = v_i + 1;
END LOOP;

CLOSE v_cur1;
CLOSE v_cur2;
```


Characteristics of MySQL Cursors

- Readonly
  - We can use cursor only for reading from the table.
  - Cannot update or delete from the cursor.
    - SET v_cur1 = (1, 'NewName'); -- not allowed
  - To update or delete programmer can use UPDATE/DELETE queries.

- Non-scrollable
  - Cursor is forward only.
  - Reverse traversal or random access of rows is not supported.
  - When FETCH is done, current row is accessed and cursor automatically go to next row.
  - We can close cursor and reopen it. Now it again start iterating from the start.
- Asensitive
  - When cursor is opened, the addresses of all rows (as per SELECT query) are recorded into the cursor (internally). These rows are accessed one by one (using FETCH).
  - While cursor is in use, if other client modify any of the rows, then cursor get modified values. Because cursor is only having address of rows.
  - Cursor is not creating copy of the rows. Hence MySQL cursors are faster.

Functions

- MySQL Function Types
  - DETERMINISTIC
    - If input is same, output will remain same ALWAYS.
    - Internally MySQL cache input values and corresponding output.
    - If same input is given again, directly output may return to speedup execution.
  - NOT DETERMINISTIC
    - Even if input is same, output may differ.
    - Output also depend on current date-time or state of table or database settings.
    - These functions cannot be speedup.

Triggers

```
DROP TABLE IF EXISTS accounts;

CREATE TABLE accounts(id INT, type CHAR(20), balance DOUBLE);
INSERT INTO accounts VALUES (1, 'Saving', 10000);
INSERT INTO accounts VALUES (2, 'Saving', 2000);
INSERT INTO accounts VALUES (3, 'Current', 25000);
INSERT INTO accounts VALUES (4, 'Saving', 7000);

CREATE TABLE transactions(accid INT, type CHAR(20), tim DATETIME, amount DOUBLE);
```

- Implement trigger -- psm19.sql

```
SELECT * FROM accounts;

INSERT INTO transactions VALUES (1, 'WITHDRAW', NOW(), 2000);

SELECT * FROM accounts;

INSERT INTO transactions VALUES (2, 'DEPOSIT', NOW(), 500);

SELECT * FROM accounts;
```

```
SELECT * FROM transactions;
```

```
SHOW TRIGGERS FROM classwork;
```

- Assign: When sal of emp is modified, make entry into result table of old and new sal.
    - Trigger --> BEFORE UPDATE ON emp

```
CREATE TRIGGER trig_salchange
AFTER UPDATE ON emp
FOR EACH ROW
BEGIN
    DECLARE v_empno DOUBLE DEFAULT OLD.empno;
    DECLARE v_oldsal DOUBLE DEFAULT OLD.sal;
    DECLARE v_newsal DOUBLE DEFAULT NEW.sal;
    IF OLD.sal != NEW.sal THEN
        INSERT INTO result VALUES (v_empno, CONCAT(v_oldsal, ' --> ', v_newsal));
    END IF;
END;
$$
```

```
START TRANSACTION;

UPDATE emp SET sal = 1200 WHERE empno = 7900;
-- OLD.sal = 950.0 --> NEW.sal = 1200.0
-- OLD.empno = NEW.empno -- both are same (because we are not modifying empno.)

COMMIT; -- both changes will be permanent
-- or
ROLLBACK; -- both changes will be discarded
```

NoSQL - MongoDb

- Mongo Server is running in backgroud (mongod).
    - Version: 3.6+
- Open command prompt -- Start Mongo Shell.
- terminal> mongo
    - By default security is disabled in Mongo.
- Try next queries in mongo shell ">".
- JS is case sensitive. Most of Mongo queries are case sensitive.

```

```

```
show databases;

use classwork;
// classwork db will be created when first record is added in it.

db;
// db is keyword -- represent current database.

show collections;

db.people.insert({
    name: "Nilesh",
    age: 37,
    addr: {
        city: "Pune",
        pin: 411037
    },
    email: "nilesh@sunbeaminfo.com"
});
// insert a document {...} json into 'people' collection in current database (db).

show collections;

show databases;

db.people.insert({
    name: "Nitin",
    mobile: "9881208115",
    email: "nitin@sunbeaminfo.com"
});

db.people.insertMany([
{
    name: "Prashant",
    mobile: "9881208114",
    email: "prashant@sunbeaminfo.com"
},
{
    name: "Sunbeam Infotech",
    phone: "020-24260308",
    website: "www.sunbeaminfo.com"
}
]);
```

```
db.people.find();

db.people.insert({
    _id: 1,
    name: "Sarang",
    addr: {
        area: "Karad",
```

```
        city: "Satara"
    }
});

db.people.find();

db.people.insert({
    _id: 1,
    name: "Rachana",
    addr: {
        area: "Karad",
        city: "Satara"
    }
});
// error: _id cannot be duplicated

db.people.insert({
    _id: 2,
    name: "Rachana",
    addr: {
        area: "Karad",
        city: "Satara"
    }
});

db.people.find();
```

- find() operation returns mongo cursor.
- Cursor is used to access elements one by one.
- Cursor functions:
    - pretty() -- format the output
    - limit(n) -- LIMIT n
    - skip(m) -- skip m records
    - sort() -- ORDER BY

```
db.people.find().pretty();

// LIMIT 2;
db.people.find().limit(2);

db.people.find().skip(2);

// LIMIT 2, 3;
db.people.find().skip(2).limit(3);

db.people.find().skip(2).limit(3).pretty();
```

```
db.dept.remove({});
db.emp.remove({});

db.dept.insert({_id:10,dname:"ACCOUNTING",loc:"NEW YORK"});
db.dept.insert({_id:20,dname:"RESEARCH",loc:"DALLAS"});
db.dept.insert({_id:30,dname:"SALES",loc:"CHICAGO"});
db.dept.insert({_id:40,dname:"OPERATIONS",loc:"BOSTON"});

db.emp.insert({_id:7369,ename:"SMITH",job:"CLERK",mgr:7902,sal:800.00,deptno:20});
db.emp.insert({_id:7499,ename:"ALLEN",job:"SALESMAN",mgr:7698,sal:1600.00,comm:300
.00,deptno:30});
db.emp.insert({_id:7521,ename:"WARD",job:"SALESMAN",mgr:7698,sal:1250.00,comm:500.
00,deptno:30});
db.emp.insert({_id:7566,ename:"JONES",job:"MANAGER",mgr:7839,sal:2975.00,deptno:20
});
db.emp.insert({_id:7654,ename:"MARTIN",job:"SALESMAN",mgr:7698,sal:1250.00,comm:14
00.00,deptno:30});
db.emp.insert({_id:7698,ename:"BLAKE",job:"MANAGER",mgr:7839,sal:2850.00,deptno:30
});
db.emp.insert({_id:7782,ename:"CLARK",job:"MANAGER",mgr:7839,sal:2450.00,deptno:10
});
db.emp.insert({_id:7788,ename:"SCOTT",job:"ANALYST",mgr:7566,sal:3000.00,deptno:20
});
db.emp.insert({_id:7839,ename:"KING",job:"PRESIDENT",sal:5000.00,deptno:10});
db.emp.insert({_id:7844,ename:"TURNER",job:"SALESMAN",mgr:7698,sal:1500.00,comm:0.
00,deptno:30});
db.emp.insert({_id:7876,ename:"ADAMS",job:"CLERK",mgr:7788,sal:1100.00,deptno:20})
;
db.emp.insert({_id:7900,ename:"JAMES",job:"CLERK",mgr:7698,sal:950.00,deptno:30});
db.emp.insert({_id:7902,ename:"FORD",job:"ANALYST",mgr:7566,sal:3000.00,deptno:20}
);
db.emp.insert({_id:7934,ename:"MILLER",job:"CLERK",mgr:7782,sal:1300.00,deptno:10}
);

db.emp.find();

// ORDER BY sal ASC
db.emp.find().sort({sal: 1});

// ORDER BY sal DESC
db.emp.find().sort({sal: -1});

// ORDER BY deptno ASC, sal DESC
db.emp.find().sort({deptno: 1, sal: -1});

// ORDER BY sal DESC LIMIT 1;
db.emp.find().sort({sal: -1}).limit(1).pretty();
```