# RDBMS - SQL

## Agenda

- Transaction
- Row locking
- DUAL table
- SQL Functions

## Transaction

- TCL - Transaction Control Language

```
SELECT USER(), DATABASE();

CREATE TABLE accounts(id INT, type CHAR(10), balance DECIMAL(10,2));

INSERT INTO accounts VALUES
(1, 'Saving', 20000),
(2, 'Saving', 500),
(3, 'Current', 200000),
(4, 'Saving', 40000);

SELECT * FROM accounts;

START TRANSACTION;

UPDATE accounts SET balance = balance - 1000 WHERE id = 1; -- dml1
UPDATE accounts SET balance = balance + 1000 WHERE id = 2; -- dml2

SELECT * FROM accounts;

COMMIT; -- finalize changes in rdbms (end of transaction)

SELECT * FROM accounts;

START TRANSACTION;

UPDATE accounts SET balance = balance - 1000 WHERE id = 1; -- dml1
UPDATE accounts SET balance = balance + 1000 WHERE id = 3; -- dml2

SELECT * FROM accounts;

ROLLBACK; -- discard changes in transaction  (end of transaction)

SELECT * FROM accounts;
```

```
START TRANSACTION;
INSERT INTO accounts VALUES(10, 'Saving', 10000);
INSERT INTO accounts VALUES(20, 'Saving', 10000);
SELECT * FROM accounts;
SAVEPOINT sa1;
INSERT INTO accounts VALUES(30, 'Saving', 10000);
INSERT INTO accounts VALUES(40, 'Saving', 10000);
SELECT * FROM accounts;
SAVEPOINT sa2;
INSERT INTO accounts VALUES(50, 'Saving', 10000);
INSERT INTO accounts VALUES(60, 'Saving', 10000);
SELECT * FROM accounts;
ROLLBACK TO sa2; -- discard 50 & 60 rows
SELECT * FROM accounts;
INSERT INTO accounts VALUES(70, 'Saving', 10000);
INSERT INTO accounts VALUES(80, 'Saving', 10000);
SELECT * FROM accounts;
COMMIT;
```

```
START TRANSACTION;

INSERT INTO accounts VALUES(90, 'Saving', 10000);
INSERT INTO accounts VALUES(100, 'Saving', 10000);

SELECT * FROM accounts;

DROP TABLE newpeople;
-- DDL query auto-commit current transaction.
SHOW TABLES;

ROLLBACK;
-- have no effect

SELECT * FROM accounts;
```

```
SELECT @@autocommit;

SET autocommit = 0;

SELECT @@autocommit;

INSERT INTO accounts VALUES (110, 'Saving', 10000);
INSERT INTO accounts VALUES (120, 'Saving', 10000);

ROLLBACK; -- or COMMIT

SELECT * FROM accounts;
```

```
DELETE FROM accounts;

SELECT * FROM accounts;

ROLLBACK; -- or COMMIT

SELECT * FROM accounts;

EXIT;
```

- Ensure than autocommit is 1.
- Open two different command prompts and login with two different users.

```
-- user1
SELECT * FROM accounts;

-- user2
SELECT * FROM accounts;

-- user1
START TRANSACTION;

-- user1
UPDATE accounts SET balance = balance - 1000 WHERE id = 1; -- dml1
UPDATE accounts SET balance = balance + 1000 WHERE id = 3; -- dml2
SELECT * FROM accounts;

-- user2
SELECT * FROM accounts;
-- changes from user1 are not visible

-- user1
COMMIT;

-- user2
SELECT * FROM accounts;
-- changes from user1 are visible
```

```
-- user1
SELECT * FROM accounts;

-- user2
SELECT * FROM accounts;

-- user1
START TRANSACTION;

-- user1
UPDATE accounts SET balance = balance - 1000 WHERE id = 1; -- dml1
```

```
UPDATE accounts SET balance = balance + 1000 WHERE id = 2; -- dml2
SELECT * FROM accounts;

-- user2
UPDATE accounts SET balance = balance - 1000 WHERE id = 3; -- dml1
-- query execution is blocked until timeout or user1 transaction is completed
(commit or rollback).
-- whole table was locked due to DML operation in the table (by user1)
UPDATE accounts SET balance = balance + 1000 WHERE id = 4; -- dml2
SELECT * FROM accounts;
```

- In MySQL by default whole table is locked when DML operations are performed in a transaction.
- To achieve row locking MySQL table must be indexed (or have primary key).

```
ALTER TABLE accounts ADD PRIMARY KEY(id);

-- user1
SELECT * FROM accounts;

-- user2
SELECT * FROM accounts;

-- user1
START TRANSACTION;

-- user1
UPDATE accounts SET balance = balance - 1000 WHERE id = 1; -- dml1
UPDATE accounts SET balance = balance + 1000 WHERE id = 2; -- dml2
SELECT * FROM accounts;

-- user2
UPDATE accounts SET balance = balance - 1000 WHERE id = 3; -- dml1
UPDATE accounts SET balance = balance + 1000 WHERE id = 4; -- dml2
SELECT * FROM accounts;

-- user2
UPDATE accounts SET balance = balance - 1000 WHERE id = 1; -- dml1
-- query execution is blocked until timeout or user1 transaction is completed
(commit or rollback).
-- only rows modified by user1 are locked (by user1)
```

```
-- user1
SELECT * FROM accounts;

-- user1
START TRANSACTION;

-- user1
SELECT * FROM accounts WHERE id = 1 FOR UPDATE;
```

```
-- lock account 1 for update/delete.

-- user2
SELECT * FROM accounts;

UPDATE accounts SET balance = balance - 1000 WHERE id = 1; -- dml1
-- will be blocked upto time or tx completion from user1

SELECT * FROM accounts WHERE id = 1 FOR UPDATE;
-- will be blocked upto time or tx completion from user1

-- user1
UPDATE accounts SET balance = balance - 1000 WHERE id = 1; -- dml1

COMMIT; -- or rollback
```

## DUAL Table

- SELECT columns FROM tablename;
- DUAL table is a dummy/pseduo/virtual table for one row and one column.
- Used for executing arbitrary expression or few functions.
- You cannot perform DML & DDL operations on DUAL table.

```
SELECT USER();

SELECT USER() FROM DUAL;

SELECT 2 + 3 * 4;

SELECT 2 + 3 * 4 FROM DUAL;

SELECT NOW();

SELECT NOW() FROM DUAL;

DESCRIBE DUAL; -- error

SELECT * FROM DUAL; -- error
```

## SQL functions

```
-- single row function
SELECT ename, LOWER(ename) FROM emp;

-- multi row function
SELECT SUM(sal) FROM emp;
```

## String Functions

```
HELP Functions;

HELP String Functions;

HELP LOWER;

SELECT LOWER('Sunbeam Infotech');

SELECT UPPER('Sunbeam Infotech');

SELECT name, LOWER(name), UPPER(name) FROM books;

UPDATE books SET name = UPPER(name);

SELECT id, name FROM books;

SELECT CONCAT('Nilesh', ' ', 'Ghule');

SELECT ename, job, sal FROM emp;

SELECT CONCAT(ename, job), sal FROM emp;

SELECT CONCAT(ename, ' ', job), sal FROM emp;

SELECT CONCAT(ename, ' ', job) AS name_job, sal FROM emp;

SELECT CONCAT(empno, '-', ename), sal FROM emp;

SELECT ename, sal, comm, CONCAT(sal, '-', comm) FROM emp;

SELECT ASCII('A'), ASCII('a');
SELECT ASCII('ABCD');

SELECT CHAR(65);
SELECT CHAR(65 USING ASCII); -- A

SELECT UPPER(NULL), LOWER(NULL), ASCII(NULL);

SELECT TRIM('    ABCD     ');
SELECT LTRIM('    ABCD     ');
SELECT RTRIM('    ABCD     ');

SELECT LPAD('Sunbeam', 10, '*');
SELECT RPAD('Sunbeam', 10, '*');
SELECT RPAD( LPAD('Sunbeam', 17, '*'), 27, '*');

-- syntax: SUBSTRING(expression, start_pos, length)

-- substring start position +ve means from left
SELECT SUBSTRING('SUNBEAM', 4, 2);
```

```
SELECT SUBSTRING('SUNBEAM', 4, 4);
SELECT SUBSTRING('InfoTech', 3, 3);
SELECT SUBSTRING('SUNBEAM', 10, 3);

-- substring start position -ve means from right
SELECT SUBSTRING('SUNBEAM INFOTECH', -4, 4);
SELECT SUBSTRING('SUNBEAM INFOTECH', -8, 4);

-- in mysql, length cannot be -ve. Prints empty string.
SELECT SUBSTRING('SUNBEAM INFOTECH', 4, -2);

-- display first letter of all emp names.
SELECT SUBSTRING(ename, 1, 1) FROM emp;

-- find all emps starting with 'M'.
SELECT ename FROM emp WHERE SUBSTRING(ename, 1, 1) = 'M';

-- find all emps whose name start from 'C' to 'M'.
SELECT ename FROM emp WHERE SUBSTRING(ename, 1, 1) BETWEEN 'C' AND 'M';

SELECT LEFT('Sunbeam Infotech', 4);
SELECT RIGHT('Sunbeam Infotech', 4);
```

## Numeric Functions

```
HELP Numeric Functions;

SELECT POWER(2, 5);
SELECT POWER(2, 0.5);
SELECT SQRT(2);

SELECT ABS(-3), ABS(6);

SELECT ROUND(1234.5678, 2), ROUND(1234.4321, 2);
SELECT ROUND(1234.5678, -2), ROUND(3456.1234, -2);
SELECT ROUND(1234.5678, 0), ROUND(3456.1234, 0);

SELECT ROUND(5678.12, -4);

-- CEIL() --> Round-Up to next "Int".
SELECT CEIL(3.4), CEIL(-3.4);

-- FLOOR() --> Round-Down to prev "Int".
SELECT FLOOR(3.4), FLOOR(-3.4);
```

## Date and Time Functions

- Data Types
  - DATE --> calendar DATE --> yyyy-mm-dd
    - '1000-01-01' to '9999-12-31'

- TIME --> time duration --> hh:mm:ss
  - -838:59:59 to 838:59:59
- DATETIME --> calendar date and wall time --> yyyy-mm-dd hh:mm:ss
  - '1000-01-01 00:00:00' to '9999-12-31 23:59:59'
- TIMESTAMP --> number of seconds from epoch time '01-01-1970 00:00:00' --> stored as int
  - '1970-01-01 00:00:00' to '2038-01-19 03:14:07'
- YEAR --> stored as int (1)
  - 1901 to 2155

```
SELECT NOW(), SYSDATE();

SELECT NOW(), SLEEP(5), SYSDATE();

SELECT DATE(NOW()), TIME(NOW());

SELECT DAYOFMONTH(NOW());
SELECT MONTH(NOW());
SELECT MONTHNAME(NOW());
SELECT YEAR(NOW());
SELECT WEEKDAY(NOW());

-- find all emps hired on Tuesday
SELECT * FROM emp WHERE WEEKDAY(hire) = 1;

-- arg1 > arg2 --> +ve value (number of days)
SELECT DATEDIFF(NOW(), '1983-09-28');

SELECT TIMESTAMPDIFF(YEAR, '1983-09-28', NOW());
SELECT TIMESTAMPDIFF(MONTH, '1983-09-28', NOW());
SELECT TIMESTAMPDIFF(DAY, '1983-09-28', NOW());
SELECT TIMESTAMPDIFF(SECOND, '1983-09-28', NOW());

SELECT DATE_ADD(NOW(), INTERVAL 28 DAY);

-- print experience of all emps in months.
SELECT ename, hire, TIMESTAMPDIFF(MONTH, hire, NOW()) exp FROM emp;
```