



CSE 311L(Database Management System)

LAB-Week 08 (Part A)

Instructor: Nazmul Alam Diptu

MySQL Transaction

- A transaction is a logical unit of work that contains one or more SQL statements.
- Transactions are atomic units of work that can be committed or rolled back.
- When a transaction makes multiple changes to the database, either all the changes succeed when the transaction is committed, or all the changes are undone when the transaction is rolled back.
- In MySQL, some statements cannot be rolled back. CREATE, ALTER or DROP tables or stored routines.

These statements provide control over use of transactions :

- The START TRANSACTION or BEGIN statement begins a new transaction.
- COMMIT commits the current transaction, making its changes permanent.
- ROLLBACK rolls back the current transaction, canceling its changes.
- The SET autocommit statement disables or enables the default autocommit mode for the current session.
- We can enable or disable the autocommit mode explicitly by setting

SET autocommit = 0;

SET autocommit = 1;

Transaction :

```
select * from job_history;
```

```
insert into job_history(id, start_date,end_date,job_id,department_id)
```

```
values(1, '2019-01-01','2021-07-01','IT_PROG',60),
```

```
(2, '2021-01-01','2021-03-01','IT_PROG',60);
```

```
UPDATE job_history
```

```
SET start_date='2020-01-01' WHERE id=1;
```

```
ROLLBACK;
```

```
select * from job_history;
```

There is no roll back as MySQL runs with autocommit mode enabled. To disable autocommit mode, use the START TRANSACTION statement. See the following example :

```
START TRANSACTION;
```

```
UPDATE job_history
```

```
SET start_date='2018-06-01' WHERE id=1;
```

```
select * from job_history;
```

```
COMMIT;
```

```
ROLLBACK;
```

```
select * from job_history;
```

```
START TRANSACTION;
```

```
UPDATE job_history
```

```
SET start_date='2019-06-01' WHERE id=1;
```

```
select * from job_history;
```

```
ROLLBACK;
```

```
select * from job_history;
```

```
START TRANSACTION;
```

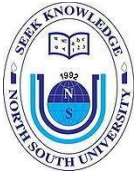
```
-- truncate job_history;
```

```
delete from job_history where id = 5;
```

```
select * from job_history;
```

```
ROLLBACK;
```

```
select * from job_history;
```



CSE 311L(Database Management System)

LAB-Week 08 (Part B)

Instructor: Nazmul Alam Dipu

Stored Procedure/Functions Examples & Demo

PROCEDURE Example

DELIMITER \$\$

CREATE PROCEDURE payroll_info()

BEGIN

SELECT

d.name AS "DEPARTMENT NAME",

e.job_id AS "JOB",

SUM(salary) AS BUDGET

FROM employees e

JOIN departments d

ON e.dept_id = d.id

GROUP BY d.name, e.job_id;

END\$\$

DELIMITER ;

CALL Payroll_info();

DROP PROCEDURE IF EXISTS payroll_info;

```
DELIMITER $$

CREATE PROCEDURE emp_info(

IN sal decimal,

IN job VARCHAR(255)

)

BEGIN

    SELECT

        d.name AS "DEPARTMENT NAME",

        e.job_id AS "JOB",

        e.salary

    FROM employees e

        JOIN departments d

            ON e.dept_id = d.id

    where e.salary >= sal

    and e.job_id = job;

END$$

DELIMITER ;


CALL emp_info(6000,'IT_PROG');


DROP PROCEDURE IF EXISTS emp_info;
```

Function Example

DELIMITER \$\$

CREATE FUNCTION SalaryStatus(

 salary DECIMAL(10,2)

)

RETURNS VARCHAR(20)

DETERMINISTIC

BEGIN

 DECLARE IncomeStatus VARCHAR(20);

 IF salary > 20000 THEN

 SET IncomeStatus = 'HIGH';

 ELSEIF (salary >= 10000 AND

 salary <= 20000) THEN

 SET IncomeStatus = 'MEDIUM';

 ELSEIF salary < 10000 THEN

 SET IncomeStatus = 'LOW';

 END IF;

 -- return the income level

 RETURN (IncomeStatus);

END\$\$

DELIMITER ;

SELECT Iname, SalaryStatus(salary)

FROM employees ORDER BY Iname;

Calling a stored function in a stored procedure

DELIMITER \$\$

```
CREATE PROCEDURE GetIncomeLevel(  
    IN id INT,  
    OUT IncomeLevel VARCHAR(20)  
)  
BEGIN  
    DECLARE income DEC(10,2) DEFAULT 0;  
    -- get credit limit of a customer  
    SELECT salary  
        INTO income  
    FROM employees e  
    WHERE e.id = id;  
    -- call the function  
    SET IncomeLevel = SalaryStatus(income);  
END$$  
DELIMITER ;
```

```
CALL GetIncomeLevel(100,@SalaryStatus);  
SELECT @SalaryStatus;  
drop procedure GetIncomeLevel;
```