Database Systems
Lecture08 – Advanced SQL

SUNG KYUN KWAN UNIVERSITY

Beomseok Nam (남범석)
bnam@skku.edu

Procedural Extensions and Stored Procedures

- SQL provides a **module** language
  - Permits definition of procedures in SQL, with if-then-else statements, for and while loops, etc.
- Stored Procedures
  - Can store procedures in the database
  - then execute them using the **call** statement
  - permit external applications to operate on the database without knowing about internal details

SQL Functions

- Define a function that, given the name of a department, returns the count of the number of instructors in that department.

```
DELIMITER //
create function dept_count (d_name varchar(20))
    returns integer
    begin
        declare d_count integer;
        select count (* ) into d_count
        from instructor
        where instructor.dept_name = d_name;
        return d_count;
     end //
DELIMITER ;
```

- Find the department name and budget of all departments with more than 12 instructors.

```
select dept_name, budget
from department
where dept_count (dept_name ) > 12
```

MySQL: DELIMITER

- The DELIMITER // statement sets a session variable so that the // becomes the statement terminator.
- For the purposes of that session, the “;” within the stored procedure are just like any other character.
- When the stored procedure is run, however, the “;” function the way that they normally do in MySQL.
- You always want to make the delimiter a “;” again when you change it.

Table Functions

- SQL:2003 added functions that return a relation as a result
- Example: Return all accounts owned by a given customer

```
create function instructors_of (dept_name char(20))
        returns table (     ID varchar(5),
                            name varchar(20),
                     dept_name varchar(20),
                            salary numeric(8,2))
        return table
        (select ID, name, dept_name, salary
         from instructor
         where instructor.dept_name = instructors_of.dept_name)
```

- Usage

```
    select *
    from table (instructors_of ('Music'))
```

MySQL: Functions

- Functions are declared using the following syntax:

```
function <function-name> (param_spec1, …, param_speck)
        returns <return_type>
        [not] deterministic             allow optimization if same output
                                         for the same input
begin
   -- execution code
end;

where param_spec is:
         [in | out | in out] <param_name> <param_type>
```

MySQL: Procedure

- In MySQL, procedures, not functions, can return a table
- Example: Return all accounts owned by a given customer

```
DELIMITER //
create procedure instructors_of (dept_name char(20))
begin
        select ID, name, dept_name, salary
        from instructor
        where instructor.dept_name = instructors_of.dept_name);
end //
DELIMITER ;
```

- Usage

    call *instructors_of* (‘Music’)

* Procedures cannot be called inside select statement

MySQL: Procedure

- A stored procedure contains a sequence of SQL commands stored in the database catalog so that it can be invoked later by a program

- Stored procedures are declared using the following syntax:

```
Create Procedure <proc-name>
    (param_spec1, param_spec2, …, param_specn )
begin
  -- execution code
end;
```

where each param_spec is of the form:

  [in | out | inout] <param_name> <param_type>

- in mode: allows you to pass values into the procedure,
- out mode: allows you to pass value back from procedure to the calling program

MySQL: IF

- Note that <condition> is a generic Boolean expression, not a condition in the MySQL sense of the word.

```
IF <condition> then
      <statements>
ELSEIF <condition> then
      <statements>
ELSE
      <statements>
END IF
```

- Note: END IF has an embedded blank, ELSEIF does not.

MySQL: Case Statement

- Case syntax:

```
CASE <expression>
      WHEN <value> then
         <statements>
      WHEN <value> then
         <statements>
      …
      ELSE
         <statements>
END CASE;
```

```
CASE
      WHEN <condition> then
            <statements>
      WHEN <condition> then
            <statements>
      …
      ELSE
            <statements>
END CASE;
```

MySQL: Repeat Until

- Syntax:

```
DELIMITER //
CREATE FUNCTION CalcIncome ( starting_value INT )
RETURNS INT
BEGIN
    DECLARE income INT;
    SET income = 0;
    label1: REPEAT
            SET income = income + starting_value;
            UNTIL income >= 4000
    END REPEAT label1;
    RETURN income;
END; //
DELIMITER ;
```

MySQL: While

- Syntax:

```
[begin_label:] WHILE <condition> DO
        <statements>
END WHILE [end_label]
```

Triggers

- A **trigger** is a statement that is executed automatically by the system as a side effect of a modification to the database.
  - Examples:
    - Charge $10 overdraft fee if the balance of an account after a withdrawal transaction is less than $500
    - Limit the salary increase of an employee to no more than 5% raise

```
CREATE TRIGGER trigger-name
trigger-time trigger-event
ON table-name
FOR EACH ROW
       trigger-action;
```

trigger-time $\in$ {BEFORE, AFTER}

trigger-event $\in$ {INSERT,DELETE,UPDATE}

Trigger Example

- Create a trigger to update the budget of a department when a new instructor is hired:

```
delimiter //
create trigger update_budget after insert on instructor
for each row
begin
   if new.dept_name is not null then
          update department
          set department.budget = department.budget + new.salary
          where department.dept_name = new.dept_name;
   end if;
end //
delimiter ;
```

- “new” refers to the new row inserted

Trigger Example

```
MariaDB> select * from department where dept_name = 'Comp. Sci.';
+------------+----------+-----------+
| dept_name  | building | budget    |
+------------+----------+-----------+
| Comp. Sci. | Taylor   | 100000.00 |
+------------+----------+-----------+
1 row in set (0.00 sec)


MariaDB> insert into instructor values (88888, 'Nam', 'Comp. Sci.',
10000.00);
Query OK, 1 row affected (0.02 sec)


MariaDB> select * from department where dept_name = 'Comp. Sci.';
+------------+----------+-----------+
| dept_name  | building | budget    |
+------------+----------+-----------+
| Comp. Sci. | Taylor   | 110000.00 |
+------------+----------+-----------+
1 row in set (0.00 sec)
```