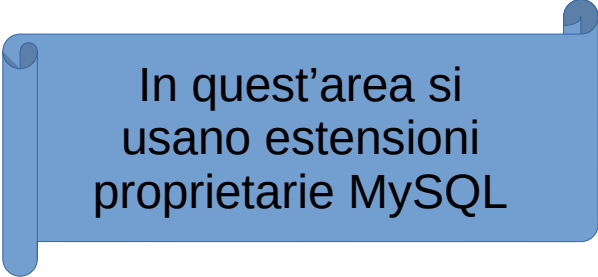


# Stored procedure


Supporto procedurale a SQL fornito da alcuni DBMS

- Non fa parte dello standard SQL
- Presente in MySQL dalla versione 5
- **procedura**: accetta parametri (in/out)
- **funzione**: procedura che ritorna un valore
- **trigger**: procedura eseguita automaticamente
  - in seguito ad una operazione DML su una tabella
- Progetto di riferimento
  - <https://github.com/egalli64/jd> – folder mySql (*modulo 5*)

# La vita di una stored procedure



In quest'area si  
usano estensioni  
proprietarie MySQL



```
drop procedure if exists hello;  
  
delimiter //  
create procedure hello()  
begin  
    select "Hello!" as greetings;  
end;  
// delimiter ;  
  
call hello();
```

# Variabili

```
declare v_a varchar(20);  
declare v_b int default 42;  
declare v_c decimal(8, 2);  
  
set v_a = "hello";  
  
select salary into v_c  
from coders  
where coder_id = 103;  
  
select concat(v_a, ", ", v_b, ", ", v_c) as result;
```

# Condizioni

```
if v_a > 0 then
    set v_b = 'v_a is positive';
elseif v_a = 0 then
    set v_b = 'v_a is zero';
else
    set v_b = 'v_a is negative';
end if;
```

```
case v_a
    when -1 then
        set v_c = 'v_a is minus one';
    when 0 then
        set v_c = 'v_a is zero';
    when 1 then
        set v_c = 'v_a is plus one';
    else
        set v_c = 'v_a is unknown';
end case;
```

# Loop

```
my_loop : loop
    set loop_message = concat(loop_message, ' ', v_i);
    set v_i = v_i + 1;
    if v_i > 6 then
        leave my_loop;
    end if;
end loop my_loop;
```

```
while v_i < 7 do
    set while_message = concat(while_message, ' ', v_i);
    set v_i = v_i + 1;
end while;
```

```
repeat
    set repeat_message = concat(repeat_message, ' ', v_i);
    set v_i = v_i + 1;
until v_i > 6 end repeat;
```

# Esempio di procedura

```
delimiter //  
create procedure total_salaries_coders()  
begin  
    declare v_total decimal(8, 2);  
  
    select sum(salary) into v_total from coders;  
  
    if v_total > 0 then  
        select v_total as "total salary for coders";  
    else  
        select "no salary information available for coders!" as warning;  
    end if;  
end;  
// delimiter ;
```

# CURSOR

```
declare cur_coders cursor for
    select first_name, last_name from coders;
declare continue handler for not found
    set v_done = true;
```

definizione di  
cursore e terminatore

uso del  
cursore

```
open cur_coders;
while not v_done do
    fetch cur_coders into v_first_name, v_last_name;
    /* ... */
end while;

/* ... */

close cur_coders;
```

# Procedure con parametri

IN (default)

OUT

INOUT

```
create procedure get_coder_salary(  
    in p_coder_id integer,  
    out p_salary decimal(8, 2)  
) begin  
    select salary  
    into p_salary  
    from coders  
    where coder_id = p_coder_id;  
end;
```

```
call get_coder_salary(9104, @result);  
select @result;
```

user-defined variable  
estensione MySQL  
session scoped



# FUNCTION

```
SET GLOBAL log_bin_trust_function_creators = 1;
```



deterministic



```
create function get_salary(  
    p_coder_id integer  
) returns decimal(8, 2)  
begin  
    declare v_result decimal(8, 2);  
  
    /* ... */  
  
    return v_result;  
end;
```

solo parametri 'in'

return type

```
select get_salary(104) as salary;
```

# TRIGGER

- Procedura eseguita automaticamente
  - prima o dopo un comando DML
- Row-level
  - Eseguito per ogni riga coinvolta
  - Accesso alla riga nello stato precedente e successivo
    - via OLD e NEW

# Un esempio di trigger

```
create trigger before_update_salary
before update on coders
for each row
begin
    set new.salary = round(new.salary, -1);
end;
```

Generazione di eventi che scatenano il trigger



```
update coders
set salary = salary + 3;
```

# SQL EXCEPTION

- Uso di un **exit handler** per **sqlexception**
  - In un blocco si definisce l'handler
  - Dopo il blocco si verifica il valore della variabile associata
- Alternativa: continue handler, il controllo è a seguire nello stesso blocco

```
declare v_error boolean default false;  
  
begin  
    declare exit handler for sqlexception  
        set v_error = true;  
  
    /* ... */  
end;
```

```
if v_error then  
    /* ... */  
else  
    /* ... */  
end if;
```

# Esercizi

- Scrivere la procedura `day_after()` che ha parametri
  - Input: una data
  - Output: la data di domani
- Riscrivere la `day_after()` come funzione
- Scrivere la procedura `get_coder()` con parametri
  - Input: id di un coder
  - Output: nome e cognome associato
- Riscrivere la `get_coder()` come funzione