

# آزمایشگاه پایگاه داده

فرآیندها و توابع ذخیره شده



نیوشا عطار - نیمسال دوم 1401-1402

## مرور جلسه گذشته

```
CREATE TABLE person (  
    ...  
    FOREIGN KEY (vaccineID) REFERENCES vaccine(vaccineID),  
    PRIMARY KEY personID  
);
```

```
SELECT *  
FROM person LEFT JOIN vaccine ON  
person.vaccineID=vaccine.vaccineID;
```

```
SELECT vaccine.name, COUNT(person.personID)  
FROM person RIGHT JOIN vaccine ON  
person.vaccineID=vaccine.vaccineID  
GROUP BY vaccine.name;
```

کلید خارجی ■

پیوند ■

داخلی ■

راست ■

چپ ■

گروه بندی سطرها ■

## فرآیندهای ذخیره شده

- یک فرایند ذخیره شده (Stored Procedure) یک قطعه کد SQL است که می توان آن را ذخیره کرده و مجددا استفاده نمود.
- می توان کوئری های پرتکرار و کاربردی را به صورت فرایند ذخیره کرد و به دفعات آن را فراخوانی نمود.
- می توان برای هر فرایند تعدادی پارامتر ورودی تعریف کرد.

## فرآیندهای ذخیره شده

<pre>delimiter // CREATE PROCEDURE <i>procedure_name</i>() BEGIN     <i>sql_statement</i> END// delimiter ;</pre>	تعریف یک فرایند ذخیره شده
<pre>CALL <i>procedure_name</i>;</pre>	فراخوانی و اجرای یک فرایند ذخیره شده

- عبارت delimiter تعیین کننده جداساز دستورات است.
- delimiter به طور پیش فرض ; است.
- قبل از شروع تعریف یک فرایند، delimiter را تغییر می دهیم تا بتوانیم درون تعریف از ; به عنوان مشخص کننده ی انتهای دستورات استفاده کنیم.
- پس از پایان تعریف، delimiter را به ; برمی گردانیم.

## فرآیندهای ذخیره شده - مثال 1

تعریف یک فرایند برای انتخاب تمام کسانی که زنده هستند.

```
delimiter //  
CREATE PROCEDURE select_alive ()  
BEGIN  
    SELECT *  
    FROM person  
    WHERE death IS NULL;  
END//  
delimiter ;
```

## فرآیندهای ذخیره شده

<code>CALL stored_procedure_name;</code>	فراخوانی فرآیند
<code>DROP PROCEDURE [IF EXISTS] stored_procedure_name;</code>	حذف فرآیند

## فرآیندهای ذخیره شده - مثال 2

تعریف یک فرایند برای انتخاب تمام کسانی که از سنی خاص بزرگتر هستند.

```
delimiter //
CREATE PROCEDURE select_person_by_min_age(IN min_age INT)
BEGIN
    SELECT *
    FROM person
    WHERE TIMESTAMPDIFF(YEAR,birth,CURDATE()) >= min_age;
END//
delimiter ;
```

## توابع ذخیره شده

- مانند هر زبان برنامه‌نویسی دیگری، در SQL هم می‌توان تابع تعریف کرد.
- برخلاف فرایندهای ذخیره شده که کوئری اجرا می‌کردند، توابع ذخیره شده اعمال محاسباتی انجام می‌دهند.
- اگر با محاسبه‌ی زمانی، عددی یا منطقی پرتکراری مواجه هستید آنها را به صورت تابع پیاده‌سازی کنید.



## توابع ذخیره شده

```
delimiter //  
CREATE FUNCTION func_name(param1,param2,...)  
RETURNS datatype  
[NOT] DETERMINISTIC  
BEGIN  
    code of statements to be executed  
END //  
delimiter ;
```

تعریف یک تابع ذخیره شده

- توابع می‌توانند قطعی (DETERMINISTIC) یا غیرقطعی (NOT DETERMINISTIC) تعریف شوند.
- تعریف یک تابع به عنوان قطعی به MySQL اعلام می‌کند که خروجی تابع به ازای ورودی یکسان همیشه یکسان است. MySQL از این ویژگی برای بهبود پرفرمنس و کش کردن اجراهای تابع استفاده می‌کند.
- تعریف یک تابع به عنوان غیرقطعی به MySQL اعلام می‌کند که خروجی تابع به ازای ورودی یکسان ممکن است متفاوت باشد (مثلا توابعی که عملکرد رندم دارند).
- اگر از تابعی غیرقطعی بعد از گزاره WHERE استفاده شود، MySQL تابع را به ازای هر سطر پردازش می‌کند.

## توابع ذخیره شده - مثال 1

تعریف یک تابع که تاریخ تولد را می‌گیرد و خروجی‌اش تعیین می‌کند که آیا فرد می‌تواند رای دهد یا نه!

```
delimiter //  
CREATE FUNCTION can_vote(birth DATE)  
RETURNS BOOL  
DETERMINISTIC  
BEGIN  
    IF TIMESTAMPDIFF(YEAR,birth,CURDATE()) > 18 THEN  
        RETURN 1;  
    ELSE  
        RETURN 0;  
    END IF;  
END//  
delimiter ;
```

```
SELECT * FROM person WHERE can_vote(birth);
```

## توابع ذخیره شده - مثال 2

تعریف یک تابع که تاریخ تولد را می‌گیرد و سن را خروجی می‌دهد:

```
delimiter //  
CREATE FUNCTION get_age(birth DATE)  
RETURNS INT  
DETERMINISTIC  
BEGIN  
    RETURN TIMESTAMPDIFF(YEAR, birth, CURDATE());  
END//  
delimiter ;
```

```
SELECT get_age(birth) FROM person;
```

# متغیرها در MySQL

■ MySQL به سه روش می‌تواند از متغیرها استفاده کند:

User-Defined Variable ■

Local Variable ■

System Variable ■

# USER-DEFINED VARIABLE

- این متغیرها به صورت `@var_name` تعریف و مورد استفاده قرار می گیرند.
- تنها توسط کاربری که آن را ساخته قابل دسترسی هستند و بعد از پایان سشن حذف می شوند.
- نام آنها case-sensitive نیست.
- تنها از انواع داده ی خاصی پشتیبانی می کند:
  - Integer ■
  - Float ■
  - Decimal ■
  - String ■
  - NULL ■
- دیگر انواع داده در صورت امکان به string تبدیل می شوند.

# USER-DEFINED VARIABLE

<code>SET @var_name = value;</code>	تعریف یک متغیر کاربر
<code>SET @avg_age = (SELECT AVG(get_age(birth)) FROM person); SELECT * FROM person WHERE get_age(birth) &gt; @avg_age;</code>	مثال

# LOCAL VARIABLE

Strongly-typed هستند. ■

از کلیدواژه **DECLARE** برای تعریف متغیرهای محلی استفاده می‌شود. ■

از کلیدواژه **DEFAULT** برای تعیین مقدار اولیه متغیرهای محلی استفاده می‌شود (اختیاری). ■

درون فرآیندها و توابع ذخیره شده استفاده می‌شود ■

# LOCAL VARIABLE

<code>DECLARE var_name datatype DEFAULT default_val;</code>	تعریف یک متغیر محلی
<code>DECLARE min_age INT DEFAULT 0; SET min_age = (SELECT MIN(get_age(birth)) FROM person);</code>	مثال