آزمایشگاه پایگاه داده

فرآیندها و توابع ذخیره شده

نیوشا عطار – نیمسال دوم 1400-1401

مرور جلسه گذشته

```
CREATE TABLE person (
    …
    FOREIN 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 می‌توان آن را ذخیره کرده و مجددا استفاده نمود.
- می‌توان کوئری‌های پرتکرار و کاربردی را به صورت فرایند ذخیره کرد و به دفعات آن را فراخوانی نمود.
- می‌توان برای هر فرایند تعدادی پارامتر ورودی تعریف کرد.

فرآیندهای ذخیره شده

تعریف یک فرایند ذخیره شده	```delimiter //
	CREATE PROCEDURE procedure_name()
	BEGIN
	sql_statement
	END//
	delimiter ;```

فراخوانی و اجرای یک فرایند ذخیره شده	```CALL procedure_name;```

- عبارت delimiter تعیین‌کننده جداساز دستورات است.
- delimiter به طور پیشفرض ; است.
- قبل از شروع تعریف یک فرایند، delimiter را تغییر می‌دهیم تا بتوانیم درون تعریف از ; به عنوان مشخص کننده‌ی انتهای دستورات استفاده کنیم.
- پس از پایان تعریف، delimiter را به ; برمی‌گردانیم.

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

فراخوانی فرآیند	`CALL stored_procedure_name;`
حذف فرآیند	`DROP PROCEDURE [IF EXISTS] stored_procedure_name;`

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

تعریف یک تابع ذخیره شده	```delimiter //
	CREATE FUNCTION func_name(param1,param2,…)
	RETURNS datatype
	[NOT] DETERMINISTIC
	BEGIN
	    code of statements to be executed
	END //
	delimiter ;```

- توابع می‌توانند قطعی (DETERMINSTIC) یا غیرقطعی (NOT DETERMINISATIC) 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 می‌گیرند.
- تنها توسط کاربری که آن را ساخته قابل دسترسی هستند و بعد از پایان سشن حذف می‌شوند.
- نام آنها `case-sensitive` نیست.
- تنها از انواع داده‌ی خاصی پشتیبانی می‌کند:
  - `Integer`
  - `Float`
  - `Decimal`
  - `String`
  - `NULL`
- دیگر انواع داده در صورت امکان به `string` تبدیل می‌شوند.

USER-DEFINED VARIABLE

تعریف یک متغیر کاربر	`SET @var_name = value;`
مثال	`SET @avg_age = (SELECT AVG(get_age(birth)) FROM person);` `SELECT * FROM person WHERE get_age(birth) > @avg_age;`

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

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