???? SQL (????????, ?????????, ???????, ? ????? ???????, ?????, ???????? ?????????, ????????? ???????).

???????

хранимая процедура, которая автоматически выполняется в ответ на определённые события в базе данных

События:

- INSERT
- UPDATE
- DELETE

Время срабатывания:

- BEFORE
- AFTER

Пример триггера:

```
CREATE TRIGGER <name_trigger>
BEFORE INSERT ON <table_name>
FOR EACH ROW
BEGIN
    SET NEW.created_at = NOW();
END;
```

???????? ?????????

набор SQL-операторов, сохранённых в БД и выполняемых по явному вызову

Особенности:

- могут принимать входные и выходные параметры (необязательно);
- возвращают данные через OUT-параметры;
- выполняются на стороне базы данных;
- могут содержать управляющие конструкции (циклы, условия, курсоры);
- могут изменять данные.

Пример:

```
CREATE PROCEDURE add_user(IN p_name VARCHAR(50))
BEGIN
    INSERT INTO users(name) VALUES (p_name);
END;

-- Вызов процедуры
CALL add_user('Ivan');
```

???????

подпрограмма, которая обязательно возвращает значение

Особенности:

- может использоваться в SQL-запросах и выражениях;
- как правило, не изменяет данные;
- всегда возвращает одно значение.

Пример:

```
CREATE FUNCTION get_user_count()
RETURNS INT
BEGIN
    RETURN (SELECT COUNT(*) FROM users);
END;

SELECT get_user_count();
```

??????

механизм для построчной обработки результата запроса, когда невозможно обработать данные одним SQL-запросом

Используется:

- чаще всего в хранимых процедурах;
- при необходимости последовательной обработки строк.

Основные шаги работы с курсором:

1. объявление курсора
2. открытие
3. чтение строк (FETCH)
4. закрытие
5. освобождение ресурсов

Пример:

```
-- Объявление
DECLARE cur CURSOR FOR
SELECT id FROM users;

-- Открытие
OPEN cur;

-- Чтение строк
FETCH NEXT FROM cur;

-- Закрытие
CLOSE cur;

-- Освобождение ресурсов
DEALLOCATE cur;
```

?????

В SQL (в хранимых программах) используются следующие циклы:

- WHILE (проверка условия происходит перед выполнением тела цикла)

```
DECLARE counter INT DEFAULT 1;
```

```
WHILE counter <= 5 DO
    INSERT INTO numbers(value) VALUES (counter);
    SET counter = counter + 1;
END WHILE;
```

- REPEAT ... UNTIL (проверка условия происходит после выполнения тела цикла)

```
DECLARE counter INT DEFAULT 1;

REPEAT
    INSERT INTO numbers(value) VALUES (counter);
    SET counter = counter + 1;
UNTIL counter > 5
END REPEAT;
```

- LOOP (базовый бесконечный цикл, выход осуществляется вручную через LEAVE)

```
DECLARE counter INT DEFAULT 1;

my_loop: LOOP
    IF counter > 5 THEN
        LEAVE my_loop;
    END IF;
    INSERT INTO numbers(value) VALUES (counter);
    SET counter = counter + 1;
END LOOP my_loop;
```

□

???????? ?????????

Для ветвления логики применяются:

- IF ... ELSE

```
DECLARE user_age INT DEFAULT 20;

IF user_age >= 18 THEN
    SET @status = 'Adult';
ELSE
    SET @status = 'Minor';
END IF;
```

- CASE

```
SELECT
    name,
    CASE
        WHEN salary < 1000 THEN 'Low'
        WHEN salary BETWEEN 1000 AND 3000 THEN 'Medium'
        ELSE 'High'
    END AS salary_level
FROM employees;
```

????????? ???????

таблица для временного хранения данных

```
-- Создание временной таблицы
CREATE TEMPORARY TABLE temp_users (
    id INT,
    name VARCHAR(50)
);

-- Вставка данных
INSERT INTO temp_users VALUES (1, 'Ivan'), (2, 'Anna');

-- Использование временной таблицы
SELECT * FROM temp_users;

-- После окончания сессии таблица автоматически удаляется
```

Особенности:

- существует только в рамках текущего соединения;
- автоматически удаляется после завершения сессии;
- используется для хранения промежуточных результатов;
- недоступна другим пользователям.

Revision #1
Created 2026-01-14 16:56:00 UTC by Fedmog1lnkv
Updated 2026-01-14 19:05:26 UTC by Fedmog1lnkv