Тема: Процедуры и триггеры

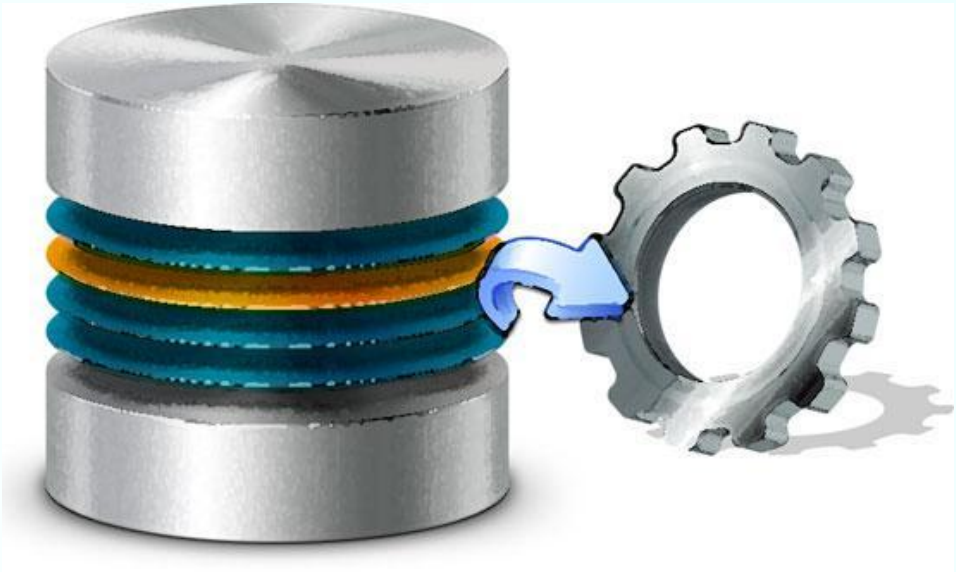
Вопросы лекции:

1. Хранимые процедуры
2. Триггеры

1. Хранимые процедуры

Хранимые процедуры

Хранимые процедуры – это объекты базы данных, которые представляют собой программы, манипулирующие данными и выполняемые на сервере. Эти программы, кроме команд языка SQL, могут использовать немногочисленные управляющие команды.



Хранимые процедуры

Структура хранимой процедуры следующая:

```
DELIMITER //
CREATE PROCEDURE имя_процедуры [(параметры)]
#Код процедуры
//
```

Объявление переменных имеет вид:

```
DECLARE имя_переменной тип_переменной [(длина)];
```

Блок операторов заключается в операторные скобки:

BEGIN … END

Оператор присвоения выглядит так:

```
SET переменная=значение;
```

Хранимые процедуры

Если нужно присвоить переменной результат команды SELECT, то используется следующий формат (многоточие означает стандартное продолжение команды):

SELECT **имя_столбца INTO переменная** FROM ...;

Условный оператор имеет вид:

IF условие THEN
    Оператор1 или Группа операторов1
[ELSE
    Оператор2 или Группа операторов2]
END IF;

Есть несколько операторов цикла, самый распространенный из них:

WHILE условие DO
    Оператор или Группа операторов
END WHILE;

Хранимые процедуры

Выражение CASE применяется для выбора на основании нескольких опций:

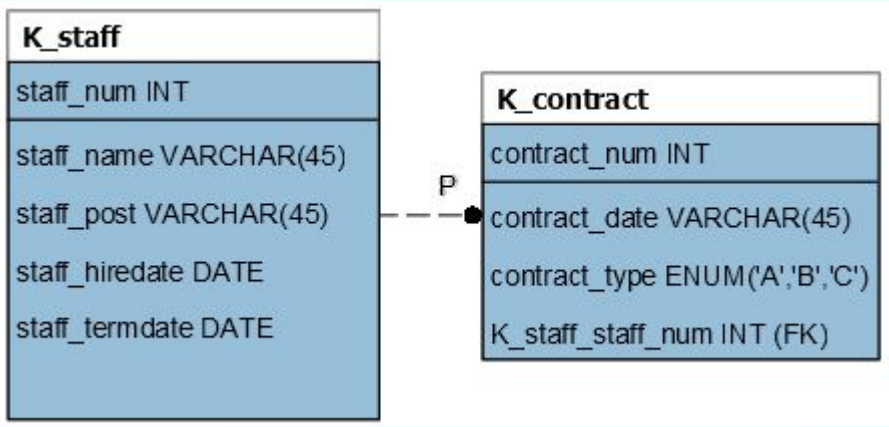
```
CASE выражение
    WHEN вариант1 THEN выражение1
    WHEN вариант2 THEN выражение2
    …
    ELSE выражениеN
END CASE;
```

Для удаления процедур используется команда:

```
DROP PROCEDURE IF EXISTS Имя_процедуры;
```

Пример хранимой процедуры

Создадим процедуру, которая в качестве параметра получает фамилию сотрудника и печатает список всех договоров, которые он курирует.




```
DELIMITER //

CREATE PROCEDURE show_contracts
    (v_staff_name CHAR(50))
BEGIN

SELECT contract_num, contract_date, contract_type
  FROM k_contract c JOIN k_staff s ON
       c.k_staff_staff_num=s.staff_num
  WHERE s.staff_name=v_staff_name;

END//
```

Хранимые процедуры

Для запуска этой процедуры нужно выполнить, например, команду

```
CALL show_contracts('Иванов');
```

Overview | Output | History | Result (1)

Export

contract_num	contract_date	contract_type
1	2011-11-02	A
3	2011-09-01	C
6	2011-07-15	C
7	2011-11-12	A

или

```
CALL show_contracts('Петров');
```

Overview | Output | History | Result (1)

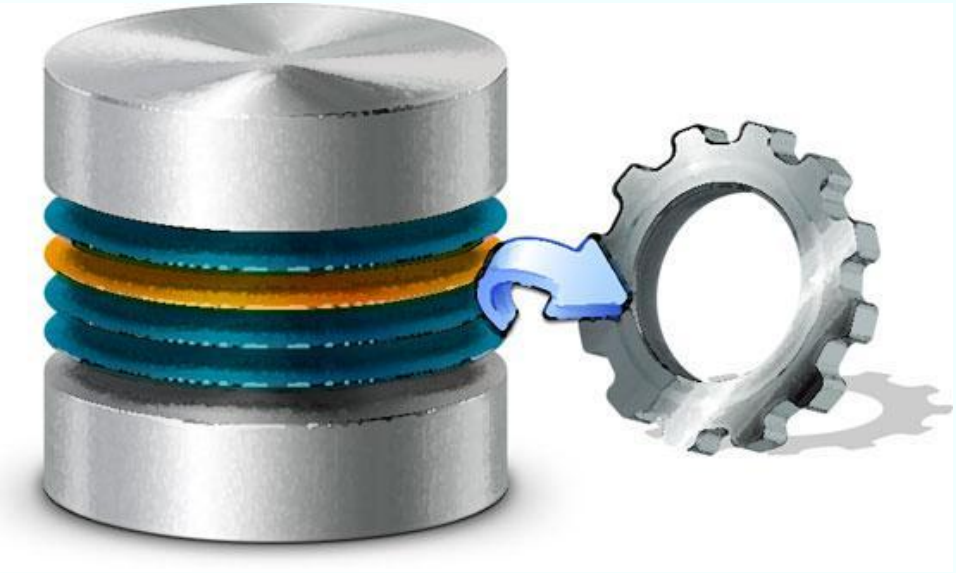
Export

contract_num	contract_date	contract_type
2	2011-10-01	B
4	2011-11-15	A
5	2011-08-01	B

2. Триггеры

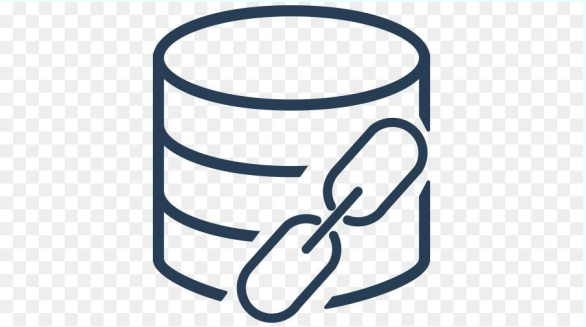
Триггеры

Триггеры – это хранимые процедуры специального вида, которые автоматически выполняются при изменении данных с помощью операторов INSERT, UPDATE и DELETE. Триггер создается для определенной таблицы, но может использовать данные других таблиц и объекты других баз данных.



Триггеры

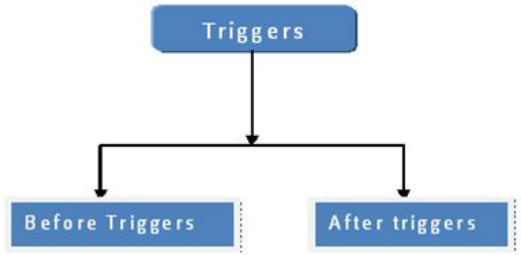
Триггеры применяются для обеспечения целостности данных и реализации сложной бизнес-логики.



Триггер запускается сервером автоматически при попытке изменения данных в таблице, с которой он связан. Все производимые им модификации данных рассматриваются как выполняемые в транзакции, в которой выполнено действие, вызвавшее срабатывание триггера.

Триггеры

- Момент запуска триггера определяется с помощью ключевых слов **BEFORE** (триггер запускается до выполнения связанного с ним события; например, до добавления записи) или **AFTER** (после события).
- В случае, если триггер вызывается до события, он может внести изменения в модифицируемую событием запись.
- Некоторые СУБД накладывают ограничения на операторы, которые могут быть использованы в триггере (например, может быть запрещено вносить изменения в таблицу, на которой «висит» триггер, и т. п.).



Триггеры

- Триггеры могут быть привязаны не к таблице, а к представлению (VIEW). В этом случае с их помощью реализуется механизм «обновляемого представления». В этом случае ключевые слова BEFORE и AFTER влияют лишь на последовательность вызова триггеров, так как собственно событие (удаление, вставка или обновление) не происходит.

- В некоторых серверах триггеры могут вызываться не для каждой модифицируемой записи, а один раз на изменение таблицы. Такие триггеры называются табличными.

Триггеры

Оператор CREATE TRIGGER позволяет создать новый триггер и имеет следующий синтаксис:

```
CREATE TRIGGER имя_триггера
время_триггера событие_триггера
  ON имя_таблицы FOR EACH ROW
тело_триггера
```

Триггеры

Конструкция **время_триггера** указывает момент выполнения триггера и может принимать два значения:

BEFORE - действия триггера производятся до выполнения операции изменения таблицы;

AFTER - действия триггера производятся после выполнения операции изменения таблицы.

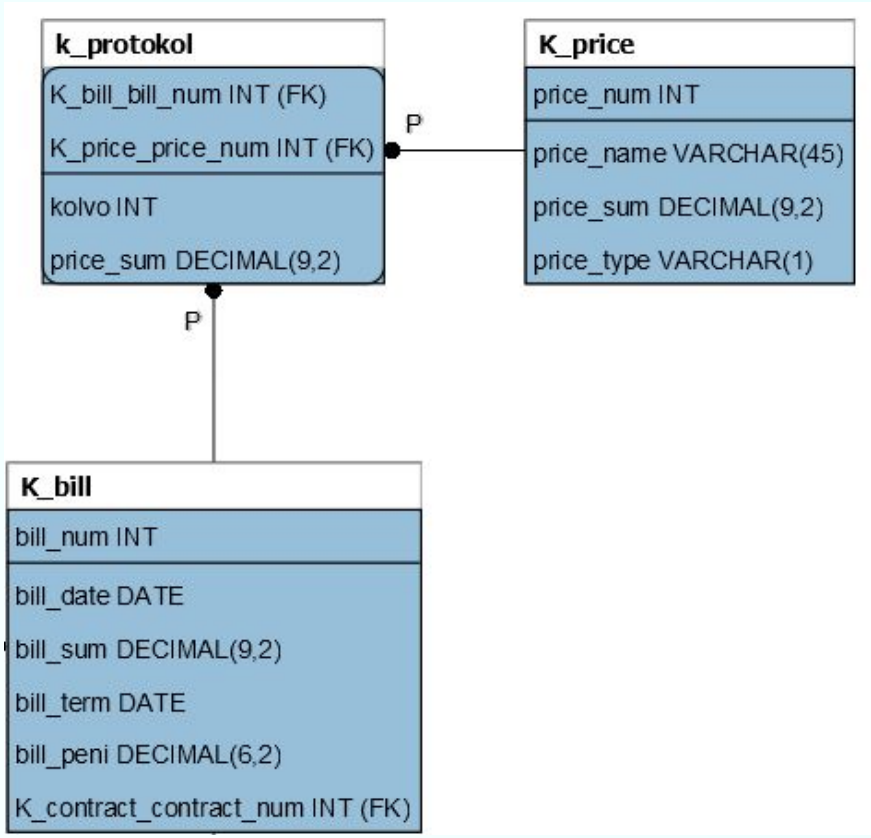
Конструкция **событие_триггера** может принимать значения

INSERT, UPDATE и DELETE.

Идентификаторы **OLD** и **NEW** означают старое и новое значение изменяемых данных.

Триггеры

Рассмотрим пример триггера вставки, который вызывается при выполнении команды INSERT в таблице **протоколов счетов (k_protocol)**. При добавлении новой позиции в счете нам нужно заново пересчитать его общую сумму (bill_sum).



Пример триггера

```
DELIMITER //
-- триггер запускается перед добавлением строки в протокол
--счетов
CREATE TRIGGER ins_prot BEFORE INSERT ON k_protokol
FOR EACH ROW
BEGIN
    DECLARE v_kolvo NUMERIC(6);       --количество
    DECLARE v_bill_num NUMERIC(6);    --номер счета
    DECLARE v_price_num NUMERIC(6);   --номер товара
    DECLARE v_price_sum NUMERIC(9,2); --цена товара
    SET v_kolvo=New.kolvo;
    SET v_bill_num=New.k_bill_bill_num;
    SET v_price_num=New.k_price_price_num;

    IF v_kolvo>0 THEN  -- только если количество >0
--из прайс-листа получаем цену товара
       SELECT p.price_sum INTO v_price_sum FROM k_price p
            WHERE p.price_num=v_price_num;
      -- обновляем общую сумму счета
       UPDATE k_bill
            SET bill_sum=bill_sum+v_kolvo*v_price_sum
                WHERE k_bill.bill_num=v_bill_num;
       -- цену товара продублируем в протоколе счета
       SET New.price_sum=v_price_sum;
    END IF;
END//
```

Спасибо за внимание!