Курс «Базы данных»

Тема. Программирование на языке PL/SQL. Часть 4

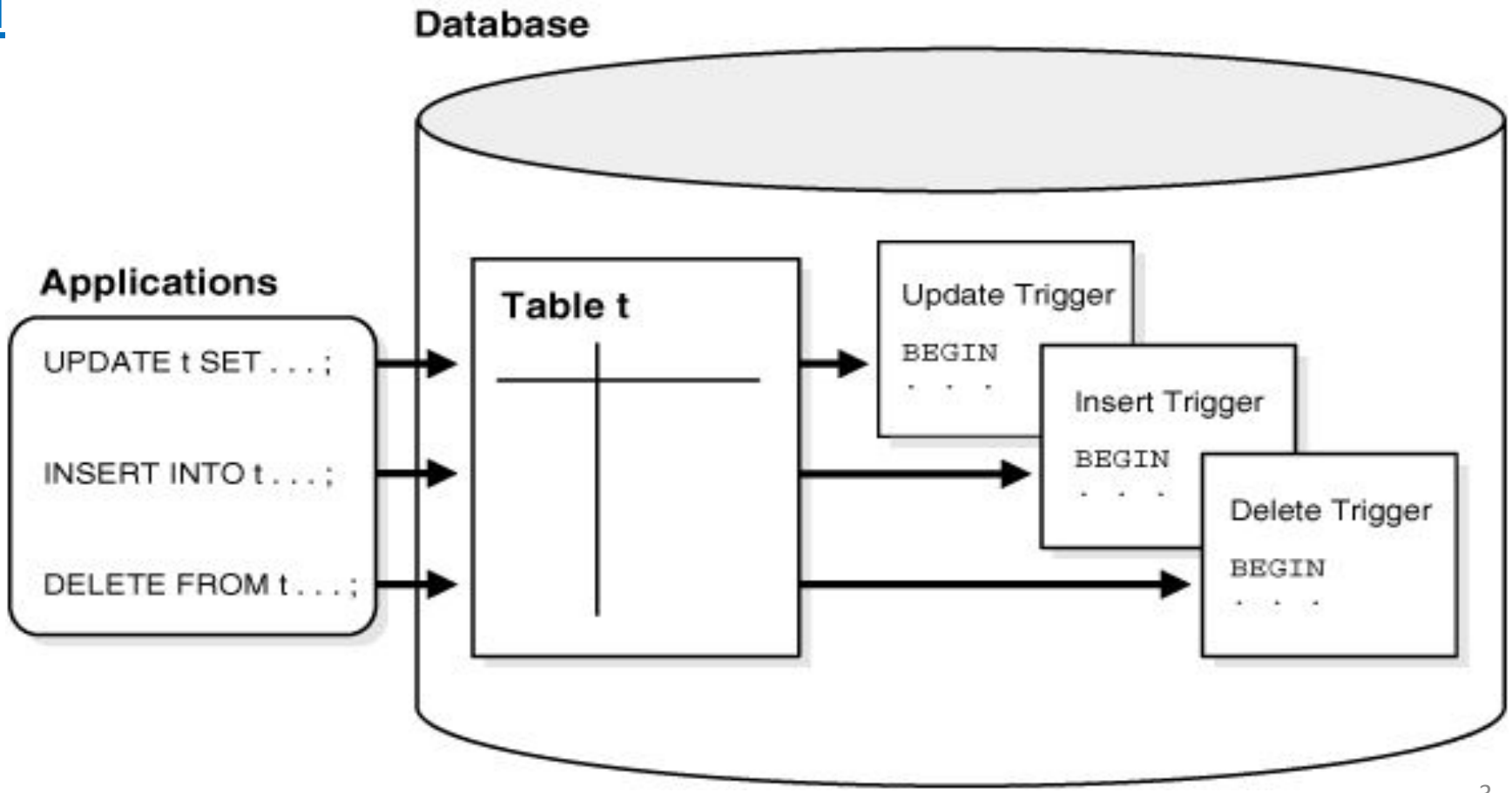
Барабанщиков
Игорь Витальевич

План лекции

1. Назначение триггеров
2. Создание DML-триггеров
3. Выполнение триггеров
4. Мутирующие таблицы
5. Сопровождение триггеров

Триггеры

Триггер – это блок PL/SQL, который **автоматически** запускается при возникновении определенных событий



Триггеры и хранимые процедуры

Процедура или функция:

- можно явно вызывать (по имени процедуры)
- им могут быть переданы параметры.

Триггер:

- ***не может быть вызван явно*** из другой процедуры БД
- ***не принимает никаких параметров*** при вызове.
- это блок **PL/SQL**, который ***автоматически срабатывает*** при определенном событии.

Назначение триггеров

- **Проверка сложных ограничений** целостности.
- **Автоматизация обработки** данных.
- **Аудит действий** пользователей.
- **Установка начальных значений** при добавлении данных в таблицы.
- **Проверка дифференцированных прав** доступа.

DML - триггеры

В команде создания триггера определяется:

- **Время срабатывания** (ДО или ПОСЛЕ)
- **Событие**, вызывающее срабатывание
- **Имя таблицы**
- **Тип триггера** (строчный или операторный)
- **Предложение WHEN** (ограничительное условие срабатывания триггера)
- **Тело триггера** (блок PL/SQL)

Синтаксис создания триггера

```
CREATE   [OR REPLACE]  TRIGGER  <имя триггера>
  { BEFORE | AFTER }
 { INSERT | DELETE | UPDATE  [ OF column_commalist ] }
  ON  <имя таблицы>
  [ REFERENCING old_or_new_values_alias_list ]
  [ FOR EACH  { ROW | STATEMENT } ]
  [ WHEN <условие>]
[ DECLARE
  -- описание переменных, констант и
  -- других элементов программы
]
BEGIN
   -- программа на процедурном языке (PL/SQL)
END;
```

Параметры триггеры DML

- **INSERT | DELETE | UPDATE** [of column] – **событие триггера.**

  Событием триггера может быть одна команда или любая комбинация указанных команд.
- **BEFORE | AFTER** – **время срабатывания триггера**: перед выполнением события триггера или после него.

  Ограничения целостности проверяются во время выполнения события триггера.
- **ON** <имя таблицы> – **таблица, к которой привязан триггер.**
- **FOR EACH { ROW | STATEMENT }** – **область действия триггера** (для каждой строки или для команды).
- **WHEN** <условие> – **условие срабатывания** триггера.

  Если оно не выполняется, триггер не будет запущен.

Запуск и выполнение триггеров

- Триггер запускается **автоматически** при наступлении события триггера.
- Триггер выполняется в рамках той транзакции, к которой относится событие триггера.
- Процедура триггера **не может содержать команды управления транзакциями и команды DDL**.
- Процедура триггера выполняется в режиме интерпретации.

Последовательность срабатывания

Последовательность срабатывания

Когда обрабатывается много строк, триггеры таблицы срабатывают в следующей последовательности:

```
UPDATE employees
    SET salary = salary * 1.1
    WHERE department_id = 30;
```

6 rows updated.

EMPLOYEE_ID	LAST_NAME	DEPARTMENT_ID
114	Raphaely	30
115	Khoo	30
116	Baida	30
117	Tobias	30
118	Himuro	30
119	Colmenares	30

→ Операторный триггер BEFORE

→ Строчный триггер BEFORE
→ Строчный триггер AFTER
...
→ Строчный триггер BEFORE
→ Строчный триггер AFTER
...
→ Строчный триггер BEFORE
→ Строчный триггер AFTER

→ Операторный триггер AFTER

ORACLE

Пример 1. Ограничение целостности

```
-- Зарплата сотрудника должна попадать в один из
-- интервалов  значений зарплаты, которые
-- установлены для разных категорий сотрудников
-- в таблице SQL_GRADE.
create or replace trigger check_salary
   before INSERT or UPDATE of salary ON emp
   for each row
   when :new.salary >= 4500
declare cnt number(3);
begin
   select count(*) into cnt from sal_grade
      where :new.salary  between  low_val  and  high_val
   if cnt=0 then raise_application_error(-20050, ’
    Зарплата не попадает ни в один из допустимых
    интервалов');
   end if;
end;
```

Псевдозаписи

- Позволяют обратиться к полям изменяемой записи и получить значения полей *до и после изменения*.
- Это записи **old** и **new**.
- С помощью конструкции **Referencing** можно изменить их имена.
- Эти записи есть только у **триггеров уровня строки** (row level).

DML → / Pseudo Records ↓	Insert	Update	Delete
New	✓	✓	✗
Old	✗	✓	✓

Пример 2. Аудит действий

-- Отслеживаем действия пользователей над таблицей TAB
-- и фиксируем данные о них в специальной таблице TAB_LOG

```
create or replace trigger audit_tab
  after INSERT or UPDATE or DELETE ON tab
declare ch char := 'U';
begin
  if INSERTING   then ch := 'I';
  elsif DELETING  then ch := 'D';
  elsif UPDATING then ch:= ‘U';
  end if;
  insert into tab_log values (substr (user, 1, 30), ch,
   sysdate);
end;
```

*-- **INSERTING, DELETING и UPDATING** – условные предикаты,*
-- позволяющие определить, какая операция явилась
-- событием триггера

Пример 3. Значения по умолчанию

```
create or replace trigger set_emp
   before INSERT or UPDATE ON emp
   for each row
begin
   if :new.date_get  IS NULL  then
      :new.date_get := trunc(sysdate);
   else
      :new.date_get := trunc(:new.date_get);
   end if;
   :new.name := upper(:new.name);
end;
```

Мутирующие таблицы

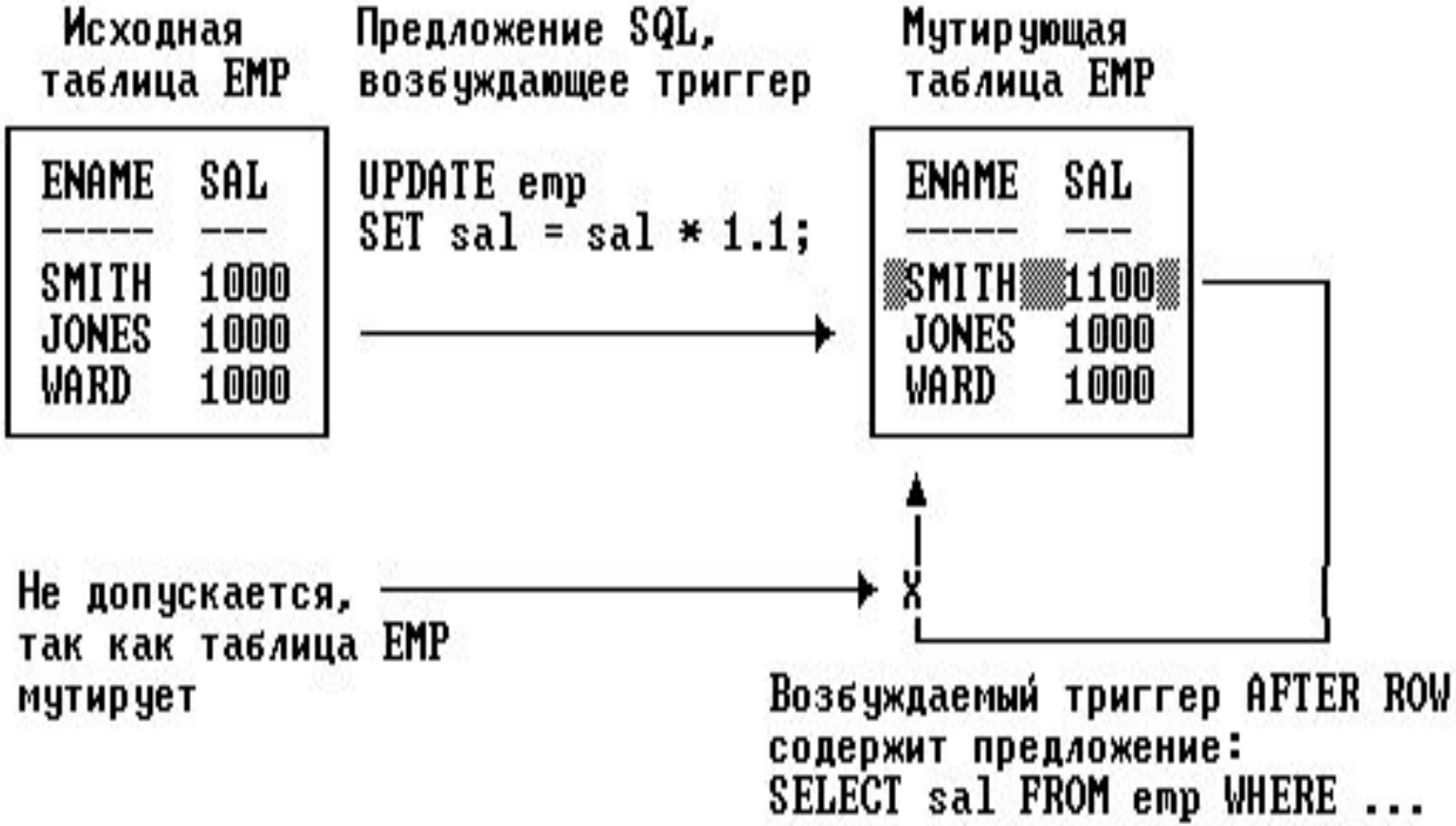
- **Мутирующей** считается таблица, **изменяемая командой INSERT, UPDATE, DELETE, которая запустила этот триггер**, а также таблицы, которые связаны с ней ссылочной целостностью DELETE CASCADE.
- **Таблица не считается мутирующей для триггера уровня предложения**, за исключением случая, когда триггер запускается как результат DELETE CASCADE.
- **Триггер не может обращаться к мутирующей таблице**: читать или

Пример мутирующей таблицы

```
CREATE TABLE emp(ename char(20), sal number(5));

CREATE TRIGGER emp_upd
   AFTER UPDATE ON emp FOR EACH ROW
DECLARE
   v_sum number(7);
BEGIN
  SELECT sal INTO v_sum FROM emp WHERE emp_id=10 ;
END;
/
UPDATE emp SET sal=sal*1.1 WHERE depno=50;
```

Пример мутирующей таблицы



Управление триггерами

- **Отключение/включение триггера**

  ALTER TRIGGER имя_триггера DISABLE | ENABLE;

- **Отключение/включение всех триггеров**

  ALTER TABLE имя_таблицы DISABLE | ENABLE ALL TRIGGERS;

- **Перекомпиляция триггера**

  ALTER TRIGGER имя_триггера COMPILE;

- **Удаление триггера**

  DROP TRIGGER имя_триггера;

- **Получение информации о триггерах**

  SELECT * FROM user_triggers;

Итоги

- Триггер – это блок PL/SQL, который автоматически выполняется при наступлении определенного события в БД.
- Триггеры используют для реализации сложных правил бизнес-логики, которые нельзя реализовать с помощью декларативных ограничений целостности.