Лабораторная работа №8

Процедуры

В лабораторной работе рассматриваются возможности процедур, не возвращающих результат. Эти процедуры вносят изменения в хранящуюся в базе данных информацию. Предполагается одновременное изучение возможностей операций INSERT, UPDATE и DELETE.

Все задачи выполняются в созданной студентами личной копии базы данных «Учет на складе».

ПРИМЕР 1.

Написать процедуру для добавления нового товара в таблицу Goods. Параметры процедуры – код товара (id), номенклатура и единица измерения.

Текст процедуры вносим в окне SQL редактора

```
CREATE PROCEDURE add_goods(
     p_id character,
     p_name character,
     p_measure character)
LANGUAGE plpgsql
AS $$
BEGIN
INSERT INTO goods (id_goods, nomenclature, measure)
    VALUES (p_id, p_name, p_measure);
--RAISE NOTICE 'Товар % добавлен', p_name;
END;
$$;
```

Для компиляции используем команду «Выполнить SQL скрипт».

Если компиляция прошла без ошибок, можно выполнить процедуру SQL командой

```
CALL ADD_GOODS('T10', 'Ручка', 'шт.');
```

Чтобы увидеть созданную процедуру в схеме базы данных, необходимо обновить соединение (F5). Теперь процедуру можно просмотреть, вызвать или изменить через контекстное меню. Обратите

внимание, что текст теперь генерируется самим сервером и имеет вид, отличающийся от введённого

```
CREATE OR REPLACE PROCEDURE public.add_goods(
     IN p_id character,
     IN p_name character,
     IN p_measure character)
LANGUAGE plpgsql
AS $procedure$
BEGIN
    INSERT INTO goods (id_goods, nomenclature, measure)
    VALUES (p_id, p_name, p_measure);
--RAISE NOTICE 'Товар % добавлен', p_name;

END;
$procedure$
;
```

Поле `id_goods` в таблице `GOODS` является первичным ключом, поэтому при выполнении процедуры необходимо задавать для параметра `p_id` различные значения (для поля `nomenclature` тоже рекомендуется использовать разные значения параметра).

Закомментированная в конце процедуры команда `RAISE NOTICE` служит для отладки, позволяя в клиенте отслеживать этапы выполнения процедуры. Чтобы убрать комментарий, откройте процедуру через контекстное меню и выберите команду «Открыть в SQL консоли». Теперь текст доступен для исправления и после компиляции будет сохранен уже в исправленном виде. Повторите запуск процедуры с другими параметрами и проследите за сообщениями в окне вывода.

Поскольку данная процедура содержит только операторы SQL, ее можно было реализовать не на языке `plpgsql`, а на языке `sql`.

Задание 1. Создайте процедуры для добавления новых данных в таблицу `AGENT`.

ПРИМЕР 2.

Написать процедуру для добавления остатка товара на склад. Параметры процедуры – название товара, название склада, количество. В значениях параметров могут быть ошибки. В этом случае процедура должна выбросить исключение, и записать информацию о возникновении ошибки в таблицу LOG_FILE.

```
CREATE PROCEDURE REST_INSERT (
     GOODS character, WH character,
     QT numeric(15,2))
LANGUAGE plpgsql AS $$
DECLARE
   idg GOODS.ID_GOODS%type;
   idw warehouse.id_wh%type;
BEGIN
  idg = (select A.id_goods
          from goods A where A.nomenclature = goods);
  idw = (select W.id_wh from warehouse W where W.name=wh);
  IF (idg is null or idw is null or qt<=0)
  THEN
    insert into log_file (inform)
     values('Ошибка в параметрах процедуры REST_INSERT ');
    commit;
    RAISE EXCEPTION 'Ошибка в параметрах процедуры';
  END IF;
  insert into goods_wh (id_wh,id_goods,quantity)
      values (idw, idg, qt);
  RAISE NOTICE 'Остаток % добавлен на склад %', GOODS, WH;
END;
$$;
```

В случае некорректных значений параметров процедура завершится выбрасыванием исключения и автоматическим выполнением команды ROLLBACK для всех произведённых в ней изменений в базе данных.

Поскольку нам необходимо сохранить информацию о неправильном вызове в таблице LOG_FILE необходимо явно выполнить команду COMMIT.

Если бы не было необходимости в проверке входных параметров (контроль выполнялся на стороне клиента), можно было бы упростить процедуру, используя подзапросы в команде INSERT:

```
insert into goods_wh (id_wh,id_goods,quantity)
    values (
     (select id_wh from warehouse W where W.name=wh),
     (select A.id_goods from goods A
          where A.nomenclature = goods), qt);
```

Для локальных переменных `idg` и `idw` использован механизм связывания типа с типом поля в базе данных. При создании локальных переменных будет использована информация из системных каталогов, если в таблицах потребуется изменить типы полей, это не скажется на синтаксической корректности процедуры и не потребуется ее изменение и перекомпиляция.

Задание 2. Создайте процедуру изменения остатка на складе. Параметры – название товара, название склада, новое количество. Предположить, что параметры заданы правильно.

Задание 3. Создайте процедуру удаления остатка со склада (должна быть удалена запись из таблицы `GOODS_WH`). Параметры – название товара, название склада. Предположить, что параметры могут быть заданы неправильно.

Задание 4. Изменить процедуру `REST_INSERT` так, чтобы дополнительно контролировалось условие – комбинация `id_goods`, `id_wh` в таблице `GOODS_WH` должна быть уникальной.

ПРИМЕР 3.

Написать процедуру, которая для заданных идентификаторов склада (`id_wh`) и товара (`id_goods`) определяет по операциям какой остаток этого товара на этом складе должен быть. При вычислении остатка предполагается,

что сначала остаток был 0, данные по операциям R добавляются к остатку, данные по операциям A вычитаются. Найденное значение остатка сравнивается с данными из таблицы GOODS_WH. Если данные совпали, в таблицу LOG_FILE заносится информация о правильных результатах инвентаризации. Если данные не совпадают, заносится информация о некорректности данных по результатам инвентаризации. Данные инвентаризации должны содержать текстовую информацию, дату и пользователя, проводившего инвентаризацию.

```
CREATE PROCEDURE INVENTORY (ID_GOODS GOODS.id_goods%Type,
     ID_WH WAREHOUSE.id_wh%Type,
     user_name character)
LANGUAGE plpgsql
AS $$
DECLARE
  QT_OPR GOODS_WH.quantity%Type;
  QT_OPA GOODS_WH.quantity%Type;
  QT GOODS_WH.quantity%Type;
BEGIN
  QT = coalesce((select a.quantity
                 from goods_wh a
                 where a.id_goods = INVENTORY.id_goods
                 and a.id_wh = INVENTORY.id_wh), 0) ;
  QT_OPA = coalesce((select sum (op.quantity)
                 from operation op
                 where op.id_goods = INVENTORY.id_goods
                 and op.id_wh = INVENTORY.id_wh
                 and op.typeop ='A'),0);
  QT_OPR = coalesce((select sum (op.quantity)
                 from operation op
                 where op.id_goods = INVENTORY.id_goods
                 and op.id_wh = INVENTORY.id_wh
                 and op.typeop ='R'),0);
  if QT = (QT_OPA-QT_OPR) then
    insert into LOG_FILE (inform,user_name,ddata)
```

```
    values('Остаток соответствует операциям '||qt,
            INVENTORY.user_name,current_date);
    RAISE notice 'Остаток соответствует операциям%',QT;
  else
    insert into LOG_FILE (inform,user_name,ddata)
    values('Остаток ' ||qt
        ||' не соответствует операциям'||(QT_OPA-QT_OPR),
        INVENTORY.user_name,current_date);
    RAISE notice 'Остаток % не соответствует операциям %',QT,
    QT_OPA-QT_OPR;
  end if;
  RAISE NOTICE 'Инвентаризация проведена ';
END;
$$;
```

Обратите внимание, что попытка подставить определение типа в параметры процедуры, будет проигнорирована на этапе компиляции и для этих параметров будет проведена замена на стандартный тип SQL.

```
type reference goods.id_goods%TYPE converted to character
type reference warehouse.id_wh%TYPE converted to character
```

Ещё одна особенность, продемонстрированная в этом примере – имена локальных переменных (в данном случае – параметров) совпадают с именами полей. Для разрешения коллизии имён пришлось в запросах уточнять имена этих переменных именем процедуры. В процедурных языках других СУБД используется, например, нотация с двоеточием:

`id_goods` – имя поля

`:id_goods`– имя локальной переменной

Задание 5. Данные в таблицах могут быть такими, что процедура всегда будет выдавать ошибку. Для полной отладки внесите изменения в таблицы так, чтобы проверить все случаи.

Задание 6. Измените (или создайте по аналогии новую) процедуру `INVENTORY`, которая будет уточнять результаты инвентаризации. Уточнение

должно для случая ошибки разделять ситуации «недостача на складе» и «неучтённый остаток на складе).

Задание 7. Измените (или создайте по аналогии новую) процедуру INVENTORY, которая будет дополнительно в начале работы записывать в LOG_FILE информацию о начале инвентаризации с указанием названия склада и названия товара, для которых проводится инвентаризации.

ПРИМЕР 4. Дополнить процедуру из примера 3 так, чтобы в случае, когда остаток на складе не совпадает с данными по операциям, дополнительно выводились данные по операциям – номер операции, дата, тип операции и количество по операции.

Воспользуемся курсорным циклом для получения данных по операциям. Изменения будут внесены только в ветку «else»

```
else
    insert into LOG_FILE (inform,user_name,ddata)
      values('Остаток ' ||qt
      ||' не соответствует операциям'||(QT_OPA-QT_OPR),
       INVENTORY.user_name,current_date);
    RAISE notice 'Остаток % не соответствует операциям %',QT,
                QT_OPA-QT_OPR;
      insert into LOG_FILE (inform)
        values('Расшифровка по операциям');
    for cur_op in
      select * from operation op
        where op.id_goods = INVENTORY.id_goods
            and op.id_wh = INVENTORY.id_wh order by op.op_date
      loop
         insert into LOG_FILE (inform,ddata)
          values('Операция номер ' ||cur_op.id ||
        ' тип - '||cur_op.typeop||
        ' количество - '||cur_op.quantity, cur_op.op_date);
       end loop;
end if;
```

Используемая в заголовке цикла переменная `cur_op` является локальной переменной-кортежем, тип такой переменной `record`. Она должна быть объявлена в разделе переменных.

```
declare
   QT_OPR GOODS_WH.quantity%Type;
   QT_OPA GOODS_WH.quantity%Type;
   QT GOODS_WH.quantity%Type;
   cur_op RECORD;
```

Задание 8. Дополните процедуру из примера 4 так, чтобы она перед выдачей информации в случае ошибки, проверяла, есть ли вообще операции для заданных товара и склада.