Лабораторная работа №12
Хранимые процедуры и функции

Особенность выполнения запросов на языке SQL в современных информационных системах заключается в том, что каждый оператор языка выполняется индивидуально на сервере БД. При этом клиент формирует выражение запроса и отправляет его на сервер для выполнения, а сервер после выполнения формирует набор данных и отправляет его клиенту. Как правило, клиент и сервер расположены на разных компьютерах, и описанный способ выполнения запросов довольно сильно нагружает сеть.

В целях снижения накладных расходов на реализацию «клиент-серверного» взаимодействия были разработаны процедурные языки, обеспечивающие группировку последовательности запросов и их выполнение на сервере. Одним из таких языков является язык PL/pgSQL, который для управления порядком выполнения запросов на языке SQL содержит в своем арсенале управляющие структуры, аналогичные используемым в языках высокого уровня. Основным назначением языка PL/pgSQL – создание функций хранимых процедур и триггеров.

1. Общие понятия хранимых процедур

Хранимые процедуры – это набор SQL-команд, который компилируется и хранится на сервере как исполняемый программный модуль и именованный объект БД. Хранимые процедуры похожи на обыкновенные процедуры, написанные на языках высокого уровня, и для них также могут быть определены входные и выходные аргументы, локальные переменные. В хранимых процедурах могут производиться числовые вычисления и операции над символами, а результаты выполнения операций могут присваиваться аргументам и переменным.

Хранимая процедура принимает входные параметры, выполняет инструкции языка описания данных (DDL) и языка обработки данных (DML), возвращает клиенту табличные или скалярные результаты, а также выходные параметры.

Организация взаимодействия между клиентом и сервером с помощью хранимых процедур предлагает следующее: клиент осуществляет вызов блока команд, хранящегося на сервере базы данных, по его имени; сервер выполняет этот блок команд и возвращает клиенту результат. Таким образом, использование хранимых процедур снижает сетевой трафик и сокращает число запросов клиентов, т.к. вместо пересылки по сети нескольких операторов передается лишь вызов необходимой процедуры.

Группировка операторов SQL в хранимой процедуре повышает гибкость работы программиста, так как вызов одной хранимой процедуры осуществить гораздо проще, чем организовать последовательное выполнение этих операторов из программы клиента. Выполнение в БД хранимых процедур вместо отдельных операторов имеет следующие преимущества:

- команды SQL находятся в БД, и их не требуется пересылать по каналу связи при вызове из прикладных программ;
- хранимые процедуры имеют модульную структуру и позволяют разбивать большие задачи на более мелкие подзадачи, что повышает гибкость управления отдельными фрагментами большой процедуры;
- хранимые процедуры после компиляции хранятся в исполняемом формате и не требуют реализации процедур лексического, синтаксического и семантического анализа каждый раз перед их выполнением, т. к. эти процедуры выполняются однократно при компиляции. Это существенно повышает скорость выполнения команд SQL, относительно выполняемых в режиме интерпретации при вызове из прикладных программ;
- хранимые процедуры могут быть вызваны из клиентских программ, других хранимых процедур или триггеров;
- сервер БД при компиляции определяет оптимизированный план выполнения хранимых процедур. Это также приводит к сокращению времени выполнения хранимых процедур.

Функции – это подпрограммы, которые имеют параметры и возвращаемые значения; они выполняют внутри себя некоторую логику. Они не могут управлять транзакциями.

Хранимые процедуры – это подпрограммы, которые имеют параметры, могут вернуть через них результат, и, в отличие от функций, могут управлять транзакциями.

2. Создание функции

```
CREATE FUNCTION имя_функции (параметры)
RETURNS тип_возвращаемого_значения
AS
$$ тело функции $$
LANGUAGE язык;
```

«$$» – это знак, который заменяет кавычки, то есть тело функции сохраняется как строка и проверить, работает функция или нет, можно только запустив ее.

Параметры являются необязательными, но могут быть входные (IN), выходные (OUT), входные и выходные (INOUT), то есть через параметр данные будут переданы внутрь функции, а результат можно вернуть через нее же.

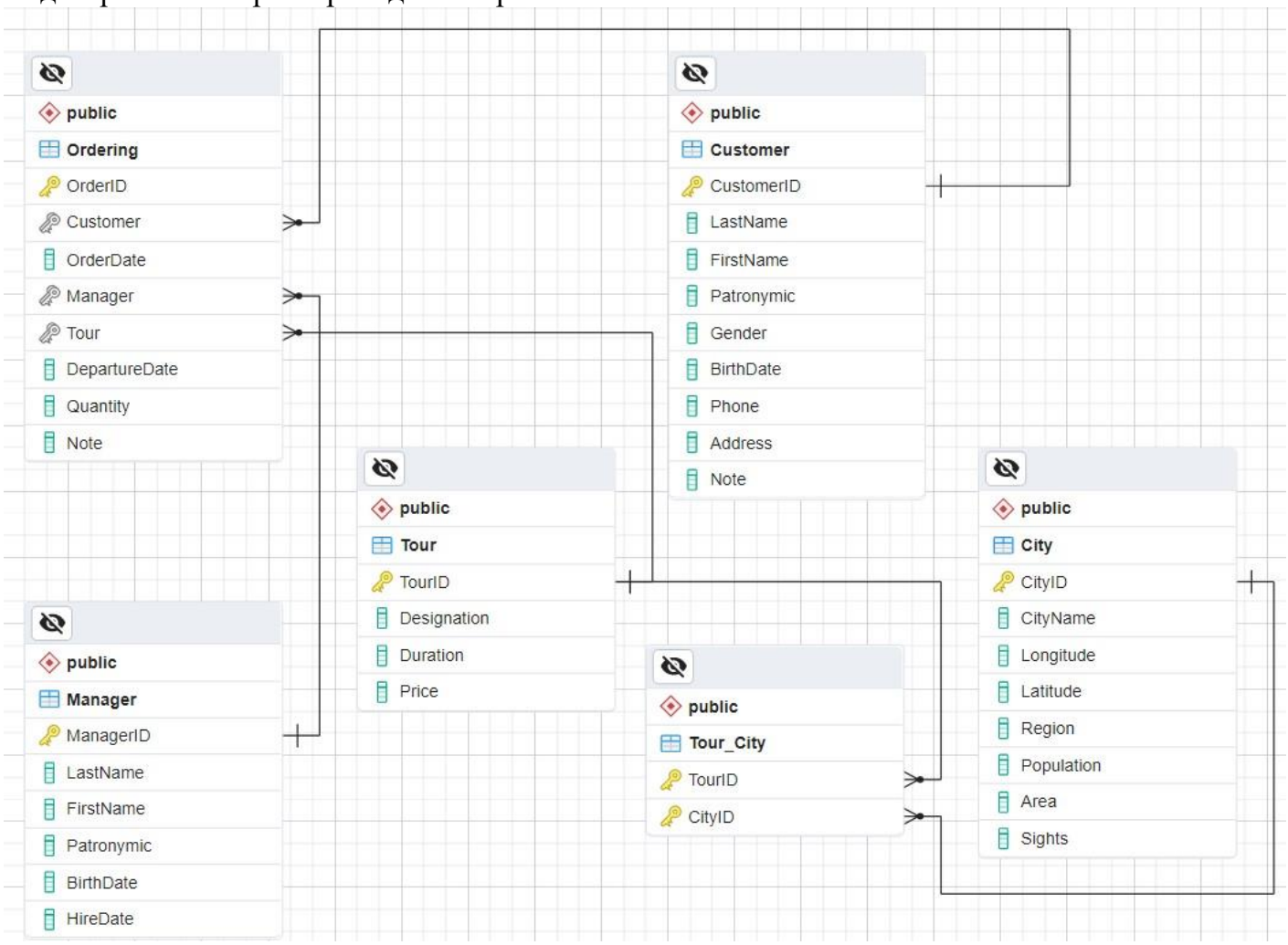
Возвращаемым значением может быть любой тип данных (даже таблица).

3. Вызов функции

Чтобы вызвать функцию, используется запрос одного из следующих видов.

```
SELECT * FROM имя_функции (параметры);
SELECT имя_функции (параметры);
```

Далее рассмотрим создание функций и хранимых процедур на примерах БД «Tour_Agency», ER-диаграмма которой приведена на рис.



Пример 1. Функция, возвращающая среднюю цену туров.

```
-- создаем функцию с именем AVG_Tour
CREATE OR REPLACE FUNCTION AVG_Tour()
-- вернется из функции значение вещественного типа
RETURNS numeric
-- начало функции
AS
-- тело функции в виде строки
$$
-- поиск средней цены тура
SELECT AVG(Price) FROM Agency.Tour;
-- конец строки с телом функции
$$
-- функция написана на языке SQL
LANGUAGE SQL;
```

Можно использовать (вызвать) эту функцию одним из следующих образов.

```
SELECT * FROM AVG_Tour();
SELECT AVG_Tour();
```

Результат выполнения запроса:

	avg_tour real
1	18973.334

Или использовать функцию в более сложном запросе, который тоже оформлен функцией.

Пример 2. Функция, возвращающая таблицу туров цена, которых выше средней цены. Для каждого такого тура дополнительно вывести среднюю цену и разницу со средней ценой.

```
CREATE OR REPLACE FUNCTION List_Tours()
RETURNS table
-- возвращаем таблицу
(TourID int, Designation varchar(50), Duration int,
-- порядок полей соответствует порядку в SELECT тела функции
Price numeric(8, 2), AVG_Price numeric(8, 2), Dif_Price numeric(8, 2))
AS
$$ SELECT
        TourID AS "Код тура",
        Designation AS "Название",
        Duration AS "Длительность",
        Price AS "Цена", round(AVG_Tour(), 2) AS "Средняя цена",
        round(Price - AVG_Tour(), 2) AS "Разница в цене"
    FROM Agency.Tour
    WHERE Price > AVG_Tour();$$
LANGUAGE SQL;
```

Результат выполнения запроса SELECT * FROM List_Tours();

tourid integer	designation character varying	duration integer	price numeric	avg_price numeric	dif_price numeric
2	Таинственный Алтай	5	27500.00	18973.33	8526.67
3	Сказочный Урал	5	23500.00	18973.33	4526.67
5	Величественный Байкал	8	53000.00	18973.33	34026.67
6	Загадочный Кавказ	9	48000.00	18973.33	29026.67
10	Дагестан	7	42000.00	18973.33	23026.67

Добавим наименование столбцов.

```
SELECT
        TourID AS "Код тура",
        Designation AS "Название",
        Duration AS "Длительность",
        Price AS "Цена", AVG_Price AS "Средняя цена",
        Dif_Price AS "Разница в цене"
FROM List_Tours();
```

	Код тура integer	Название character varying	Длительность integer	Цена numeric	Средняя цена numeric	Разница в цене numeric
1	2	Таинственный Алтай	5	27500.00	18973.33	8526.67
2	3	Сказочный Урал	5	23500.00	18973.33	4526.67
3	5	Величественный Байкал	8	53000.00	18973.33	34026.67
4	6	Загадочный Кавказ	9	48000.00	18973.33	29026.67
5	10	Дагестан	7	42000.00	18973.33	23026.67

Пример 3. Функция с параметром, возвращающая таблицу клиентов, посетивших определенный город, например, Йошкар-Олу. Город – параметр функции

```
CREATE OR REPLACE FUNCTION List_Customer(IN cityName_ varchar(30))
RETURNS table                          -- возвращаем таблицу
 (customerid integer, lastname varchar(20), firstname varchar(20), patronymic varchar(20),
gender char(1), birthdate date, phone char(12), address varchar(100), note text)
-- порядок полей соответствует порядку в SELECT тела функции
AS $$
SELECT cus.*
FROM Agency.Customer cus                    -- выведем только поля таблицы клиентов
INNER JOIN Agency.Ordering o ON cus.CustomerID = o.Customer
-- присоединим клиентов к их заказам
INNER JOIN Agency.Tour t ON o.Tour = t.TourID
-- по номеру тура в заказе найдем туры
INNER JOIN Agency.TourCity tc ON tc.TourID = t.TourID
-- по номеру тура найдем все строки из соединительной таблицы
INNER JOIN Agency.City c ON tc.CityID = c.CityID
-- найдем соответствующие города
WHERE c.Cityname = cityName_;
-- где выполняется условие равенства города значению параметра
$$
LANGUAGE SQL;
```

Вызов функции может быть осуществлен следующим образом.

```
SELECT * from List_Customer('Йошкар-Ола');
```

Пример 4. Создадим хранимую процедуру с параметром на основе запроса на удаление заданного города из таблицы городов.

```
CREATE PROCEDURE delete_data(IN cityName_ varchar(30))
LANGUAGE SQL
AS $$
DELETE FROM Agency.City          -- удалить строки из таблицы City схемы Agency
WHERE CityName = cityName_;              -- где город равен значению параметра
$$;
```

```
CALL delete_data('Москва');                -- вызов хранимой процедуры
```

Иногда требуется выполнить несколько операторов неразрывно: в одной транзакции, – для это как раз подойдут хранимые процедуры. Если никаких ошибок или нарушений ограничений целостности не произошло, то транзакция завершится сохранением изменений (commit), иначе БД вернется в состояние, которое было до начала транзакции (rollback).

Пример 5. Процедура, создающая новый тур, проходящий через несколько городов.

```
CREATE PROCEDURE insert_data()
LANGUAGE SQL
AS $$
-- начинаем транзакцию, все что будет выполнено внутри нее,
-- будет выполнено полностью, либо не выполнено совсем
BEGIN;
INSERT INTO Agency.Tour          -- добавим строку в таблицу Тур
      ( tourid, designation, duration, price)
      VALUES (100, 'Вятка', 1, 5000);
INSERT INTO Agency.Tourcity      -- добавим строку в таблицу связи городов и туров
      ( cityid, tourid)
      VALUES (100, 16);
END;                             -- завершаем транзакцию
$$;

CALL insert_data()               -- вызов хранимой процедуры
```

В предыдущих лабораторных работах были рассмотрены различные запросы, превратим некоторые их них, и не только, в функции или процедуры.

Оформите решение следующих задач в виде функций и хранимых процедур в БД «**Faculty**».

Задание 1. Напишите функцию, которая будет возвращать список студентов указанной группы. Код группы – входной параметр.

Задание 2. Напишите функцию, которая будет возвращать список компьютерных кабинетов, расположенных в указанном корпусе. Номер корпуса – входной параметр.

Задание 3. Напишите функцию, которая будет возвращать средний балл студента по фамилии, имени и отчеству. Ф, И, О – входные параметры.

Задание 4. Создайте хранимую процедуру с параметром на основе запроса на удаление заданного кабинета из таблицы аудиторий.

Задание 5. Напишите процедуру, добавляющую нового преподавателя, работающего на двух кафедрах.