Лабораторная работа №6-7

Функции в PostgreSQL. Проектирование структуры БД.

Цель работы: Изучить правила создания функций. Приобрести практические навыки создания функций в среде PostgreSQL. Научиться проектировать БД в целом на основе поставленного задания с использованием всех полученных ранее базовых знаний и навыков.

Задание. Общая часть: Ознакомиться с теоретическими сведениями о возможностях создания пользовательских функций в PostgreSQL. Разработать БД в соответствии с индивидуальным заданием. Создать функции, реализующие интерфейс для работы с базой данных. Проверить работоспособность функций путем выполнения этих функций с параметрами, обеспечивающими как успешное выполнение функции, так и невыполнение функции.

Обязательные требования к БД:

1) Наличие таблиц-справочников и таблиц, использующих справочники. Предусмотреть сохранение ссылочной целостности для таблиц, использующих таблицы-справочники.
2) Предусмотреть следующие роли:
    а) оператор БД (пополнение справочников)
    б) пользователь БД (основная работа с БД, с ограничениями для некоторого вида операций)
    в) аналитик (разрешено выполнение запросов и функций, не изменяющих данные в БД)
    г) администратор БД (просмотр протокола операций, любые изменения БД)
3) Действия, изменяющие БД пользователем с любой ролью протоколируются в таблице-журнале операций.

4) Для всех запросов необходимо создать индексы (для гарантированного использования индексов можно использовать отключение параметра enable_seqscan в текущей сессии)

Создать функции, реализующие интерфейс для работы с базой данных. Проверить работоспособность функций путем выполнения этих функций с параметрами, обеспечивающими как успешное выполнение функции, так и невыполнение функции.

Вариант 1: База данных хоккейной лиги. Должна содержать следующие данные: составы команд и информацию о каждом игроке, проведенные игры с информацией о проданных билетах и затраченных средствах. Предусмотреть анализ следующих показателей: популярность команд, рейтинг и эффективность игроков за указанный период, заполняемость стадионов (в среднем, максимальная, минимальная), среднее время на площадке за игру (для каждого игрока).

Вариант 2: База данных сети магазинов продуктов питания. Должна содержать следующие данные: информация о магазинах и имеющихся запасах продуктов, данные о продажах. Предусмотреть анализ следующих показателей: рейтинг популярности товаров, сумма среднего чека по магазину в разное время суток, анализ объемов продаж товаров (товар производится за пределами региона или в пределах города, района).

Вариант 3: База данных транспортной компании. Должна содержать следующие данные: информацию о водителях, поставщиках товаров и потребителях, завершенные и незавершенные доставки. Предусмотреть анализ следующих показателей: среднее время простоя водителя, соотношение доходности междугородних и местных доставок, анализ объемов перевозок (в ночное время или в светлое время суток), анализ объемов незавершенных доставок (в зависимости от сезона, например, лето, осень, зима, весна).

Вариант 4: База данных сети книжных магазинов. Должна содержать следующие данные: текущие складские запасы печатной продукции, информацию о заказах и продажах. Предусмотреть анализ следующих показателей: наиболее часто заказываемые книги, средний чек по разным группам товаров, средний чек в зависимости от сезона (например, лето, осень, зима, весна), рейтинг популярности книг для разных возрастных групп (например, дети, подростковый возраст, студенты, пенсионеры).

Вариант 5: База данных сети автосалонов. Должна содержать следующие данные: информацию об автосалонах, продавцах-консультантах, имеющихся в наличии и проданных автомобилях. Предусмотреть анализ следующих показателей: рейтинг продаж для продавцов-консультантов по различным моделям, рекомендации к заказу моделей на основании имеющегося запаса и популярности модели, анализ объемов продаж автомобилей (в зависимости от страны производителя), анализ спроса автомобилей среди покупателей местных и иногородних, стоимость автомобиля (средняя, максимальная, минимальная) в зависимости от страны производителя.

Вариант 6: База данных складского комплекса. Должна содержать следующие данные: данные о товарах, данные о складах, имеющихся товарах и выполненных и невыполненных заявках на них. Предусмотреть анализ следующих показателей: рейтинг востребованности товаров, рейтинг дефицитности товаров, среднее время наличия товара на складе, среднее время выполнения заявок на товар, анализ объемов продаж товаров в зависимости от города производителя.

Вариант 7: База данных сети типографий. Должна содержать следующие данные: информацию об имеющихся изданиях, информацию о читателях, формуляры для каждого издания. Предусмотреть анализ следующих показателей: выдать рекомендации для читателя с учетом его пола, возраста и прочитанных книг на основании общей статистики, популярность книг (в зависимости от жанра), среднее время наличия книги у

читателя, среднее время ожидания книги для чтения (в зависимости от возраста).

Вариант 8: База данных жилищной управляющей компании. Должна содержать следующие данные: информацию об исполнителях работ и выполненных работах, жильцах, выставленных им счетах и выполненных ими платежах. Предусмотреть анализ следующих показателей: составить рейтинг злостных неплательщиков, рейтинг исполнителей работ с указанием их доли в статье расходов, среднее время выполнения ремонта, стоимость ремонта (средняя, максимальная, минимальная), соотношение сумм, полученных за ремонт наличным и безналичным расчетом.

Вариант 9: База данных гарантийного ремонта. Должна содержать следующие данные: информацию о выпускаемых производителем предметов, гарантийных мастерских в разных городах и предметов, ремонт которых они могут производить, данные о выполненных ремонтах. Предусмотреть анализ следующих показателей: рейтинг убыточности предметов для ремонта, обеспеченность каждого города мастерскими по разным группам предметов (для ремонта) и всему ассортименту в целом, анализ объемов работ (по гарантийным мастерским), среднее время ожидания окончания ремонта предмета, стоимость ремонта (средняя, максимальная, минимальная) в зависимости от города.

Вариант 10: База данных туристического агентства. Должна содержать следующие данные: информацию об имеющихся турах, информацию о руководителях тура, данные о выполненных турах. Предусмотреть анализ следующих показателей: выдать рекомендации для клиента с учетом его пола, возраста и выполненных туров на основании общей статистики, популярность туров в зависимости от вида тура (автобусный, железнодорожный, авиа), среднее время длительности тура в зависимости от сезона (например, лето, осень, зима, весна), стоимость тура (средняя, максимальная, минимальная) в зависимости от города, анализ спроса туров среди путешественников местных и иногородних.

Содержание отчета

Отчет должен содержать титульный лист, цель работы, задание, коды команд на каждом этапе выполнения работы, результаты выполнения команд (скриншоты), выводы и анализ результатов работы.

Контрольные вопросы

1. Что такое агрегатные функции?
2. Как выполнить пользовательскую функцию?
3. Синтаксис команды создания функции.
4. Как в теле функции связать переменную со значением входного параметра?
5. Синтаксис языка PL/pgSQL.
6. Возможно ли использование в теле функции запросов?
7. Как организовать условие в теле функции?
8. Как создать цикл в теле функции?
9. Используются ли ограничения, наложенные на таблицу, при добавлении в нее записей через созданную функцию?
10. Можно ли использовать созданные функции за пределами базы данных, в которой они созданы? Почему?

3.1. Создание функций в СУБД PostgreSQL

PostgreSQL не ограничивает пользователя встроенными функциями и операторами, позволяя ему создавать собственные расширения. Если вам приходится часто выполнять некоторую стандартную последовательность команд SQL или программных операций, пользовательские функции помогут

решить эту задачу более надежно и эффективно. Также в PostgreSQL предусмотрена возможность определения операторов для вызова пользовательских (или встроенных) функции, что делает команды SQL понятнее и эффективнее.

Функции и операторы тоже существуют как объекты базы данных и поэтому связываются с конкретной базой.

Разновидность команды SQL99 *CREATE FUNCTION*, поддерживаемая в PostgreSQL, не обладает прямой совместимостью со стандартом, но зато обеспечивает широкие возможности для расширения PostgreSQL за счет создания пользовательских функции.

Синтаксис команды *CREATE FUNCTION*:

CREATE FUNCTION имя

([тип_аргумента [...]])

RETURNS тип_возвращаемого_значеиия

AS 'определение' LANGUAGE 'язык'

[WITH (атрибут [...])]

CREATE FUNCTION имя ([тип_аргумента [...]]). После ключевых слов *CREATE FUNCTION* указывается имя создаваемой функции, после чего в круглых скобках перечисляются типы аргументов, разделенные запятыми. Если список в круглых скобках пуст, функция вызывается без аргументов (хотя сами круглые скобки обязательно должны присутствовать как в определении функции, так и при ее использовании).

RETURNS тип_возвращаемого_значения. Тип данных, возвращаемый функцией.

AS 'определение'. Программное определение функции. В процедурных языках (таких, как PL/pgSQL) оно состоит из кода функции. Для откомпилированных функций С указывается абсолютный системный путь к файлу, содержащему объектный код.

LANGUAGE 'язык'. Название языка, на котором написана функция. В аргументе может передаваться имя любого процедурного языка (такого, как

plpgsql или plperl, если соответствующая поддержка была установлена при компиляции), C или SQL.

[WITH (атрибут [...])]. Аргумент *атрибут* может принимать два значения: iscachablen и isstrict. Оптимизатор может использовать предыдущие вызовы функций для ускоренной обработки будущих вызовов с тем же набором аргументов. Кэширование обычно применяется при работе с функциями, сопряженными с большими затратами ресурсов, но возвращающими один и тот же результат при одинаковых значениях аргументов. isstrict: Функция всегда возвращает NULL в случае, если хотя бы один из ее аргументов равен NULL. При передаче атрибута isstrict результат возвращается сразу, без фактического выполнения функции.

Из всех разновидностей функций в PostgreSQL проще всего создаются «чистые» функции SQL, поскольку их создание не требует ни знания других языков, ни серьезного опыта программирования. Функция SQL определяется как обычная команда с позиционными параметрами.

Позиционный параметр представляет собой ссылку на один из аргументов, переданных при вызове функции SQL. Он называется позиционным, поскольку в ссылке указывается его позиция в списке переданных аргументов. Позиционный параметр состоит из знака $, за которым следует номер (нумерация начинается с 1). Например, $1 означает первый аргумент в переданном списке.

Позиционный параметр не заключается в отдельные апострофы, поскольку апострофы являются частью переданного аргумента. Остальные составляющие определения функции являются либо идентификаторами, либо стандартными ключевыми словами SQL.

Сообщение CREATE означает, что создание функции прошло успешно.

Созданная функция доступна для всех пользователей, обладающих соответствующими правами.

СУБД PostgreSQL, написанная на языке C, может динамически подгружать откомпилированный код C без перекомпиляции пакета.

Использование команды *CREATE FUNCTION* для компоновки с функциями C разрешено только суперпользователям, поскольку эти функции могут содержать системные вызовы, представляющие потенциальную угрозу для безопасности системы.

Функции уничтожаются владельцем или суперпользователем при помощи команды SQL *DROP FUNCTION*.

Синтаксис команды *DROP FUNCTION*:

DROP FUNCTION имя ([тип_аргумента [...]])

Сообщение сервера DROP означает, что функция была успешно удалена. Команда *DROP FUNCTION*, как и большинство команд *DROP*, необратима, поэтому перед ее выполнением убедитесь в том, что функцию действительно требуется удалить.

3.1. Язык PL/pgSQL в СУБД PostgreSQL

Синтаксис функции на языке PL/pgSQL следующий:

create [or replace] function <имя функции>(<аргументы>)

returns <тип возврата> as '<тело функции>'

language 'plpgsql';

Необязательная фраза «*or replace*» позволяет перезапись функции при попытке создать другую с таким же именем (например, записать исправленный вариант). Без этого ключевого слова будет сгенерирована ошибка и потребуется сначала удалить существующую функцию и лишь затем записать на ее место новую.

Скобки после имени функции обязательны, даже если функция не имеет аргументов. Слово «*returns*» задает тип возвращаемых функцией данных. Указание языка также обязательно.

Тело функции имеет следующую структуру:

DECLARE

необязательный раздел определений

BEGIN

Операторы функции

END;

Последним оператором функции должен быть оператор «*return*», возвращающий данные указанного выше типа, даже если выход из функции осуществляется раньше (например, в блоке проверки условия) и эта команда никогда не получит управление.

Как и любой нормальный язык программирования, PL/pgSQL позволяет оперировать переменными. Все переменные (за одним исключением, о котором будет упомянуто ниже) должны быть описаны в разделе *DECLARE*, т.е. их необходимо перечислить с указанием типа данных. В этом же разделе допускается и инициализация переменной начальным значением с помощью ключевого слова *DEFAULT*. Общий синтаксис описания следующий:

<переменная> <тип> [DEFAULT <значение>];

Вместо слова «*DEFAULT*» допускается использование оператора присваивания «:=». Точка с запятой в конце каждого описания обязательна. Переменная может быть любого типа, который поддерживается в PostgreSQL. Кроме того, существуют три специальных типа данных: *RECORD, table%RECTYPE* и *table.field%TYPE*. Первый описывает запись любой таблицы, второй - запись указанной таблицы *table*, третий создает переменную такого же типа, как и тип указанного поля *field* таблицы *table*.

В основной секции, заключенной в операторные скобки *BEGIN-END*, могут использоваться операторы присваивания, математические операторы, ветвления, циклы, вызовы других функций. Кроме того, тело функции может содержать вложенные блоки, имеющие ту же структуру (т.е. DECLARE-BEGIN-END). Видимость переменных распространяется на блок, в котором она описана, и на все вложенные блоки. Ниже конспективно перечислены основные операторы языка PL/pgSQL, которые понадобятся нам в дальнейшем:

Оператор присваивания :=. Такой же, как в языке Pascal. Сопоставляет переменную с некоторым значением или результатом выражения.

Оператор ветвления *IF-ELSE-END IF*. Знакомая всем с детства конструкция, позволяющая выполнить проверку некоторого условия и в зависимости от результата проверки перейти на выполнение того или иного блока команд. Синтаксис:

If <условие> then

<операторы;…>

[else

<операторы;…>]

end if;

Оператор цикла *FOR*. Наиболее распространенный цикл. Переменная цикла *<var>* - то самое исключение, когда переменная может не описываться в секции *DECLARE*. В этом случае зона ее видимости ограничивается циклом. Синтаксис:

For <var> in <start>..<stop> loop

<тело цикла>

End loop;

Существует вариант этого цикла и для «прохода» по результату выборки:

For <row-var> in <select> loop

<тело цикла>

End loop;

В этом случае в переменную *<row-var>* последовательно подставляются строки из выборки, и тип этой переменной должен быть либо *<table>%rowtype*, отражающий запись конкретной таблицы *<table>*, либо *RECORD*, описывающий обобщенную запись таблицы.