1. Лабораторная работа №3.

Использование агрегатных функций в среде PostgreSQL. Массивы.

Цель работы: Изучить базовые операции по работе с массивами. Изучить синтаксис команд. Приобрести навыки работы с агрегатными функциями в PostgreSQL.

Задание. Общая часть: Ознакомиться с теоретическими сведениями о создании массивов. Создать таблицу с полем-массивом, таблицу с полем, содержащим многомерный массив. Выполнить вставку значений в созданные таблицы (минимум по 6 записей в каждой). Выполнить выборку из созданных таблиц (в том числе продемонстрировать предотвращение выборки NULL в массивах). Осуществить выборку с использованием среза. Продемонстрировать работу функции array_dims(). Выполнить обновление данных в созданных таблицах. Осуществить модификацию среза массива, отдельного элемента массива. Проанализировать полученную в результате выполнения операций информацию.

Вариант 1: Заполнить базу данных игроков хоккейной команды (при необходимости). Найти максимальный рост у всех нападающих. Найти минимальный вес у защитников с правым хватом клюшки. Найти средний размер премии у хоккеистов из Москвы. Найти количество нападающих с левым хватом клюшки. Найти сумму окладов у полузащитников из Санкт-Петербурга. Продемонстрировать результаты работы.

Вариант 2: Заполнить базу данных для учета работы продуктового магазина (при необходимости). Найти товар с максимальной ценой среди бакалейных товаров. Найти товар с минимальной ценой у производителя «Сибирь-агро». Найти среднюю стоимость товаров из Новосибирска. Найти количество кондитерских изделий из Москвы. Найти общую стоимость всех товаров из Санкт-Петербурга. Продемонстрировать результаты работы.

Вариант 3: Заполнить базу данных для службы такси (при необходимости). Найти заказ с максимальным расстоянием проезда. Найти заказ для автомобиля марки Тойота с минимальным расстоянием проезда. Найти среднее расстояние проезда для водителей из Москвы. Найти количество водителей с водительским стажем больше 10 лет. Найти сумму всех заказов для водителей из Санкт-Петербурга. Продемонстрировать результаты работы.

Вариант 4: Заполнить базу данных для обработки данных по работе книжной лавки (при необходимости). Найти книгу с максимальным количеством страниц. Найти книгу жанра «комедия» с минимальным количеством страниц. Найти среднюю цену книг жанра «фантастика». Найти количество книг издательства «Сибирь». Найти общую стоимость книг для авторов из Москвы. Продемонстрировать результаты работы.

Вариант 5: Заполнить базу данных для осуществления учета работы автосалона (при необходимости). Найти среди автомобилей с правым рулем автомобиль с минимальной ценой. Найти среди автомобилей с автоматической коробкой передач автомобиль с максимальной ценой. Найти среднюю стоимость машин с полным приводом. Найти количество машин с типом кузова «седан». Найти общую стоимость машин автосалона для производителя «АВТОВАЗ». Продемонстрировать результаты работы.

Вариант 6: Заполнить базу данных учета поступления товара на овощную базу (при необходимости). Найти товар с максимальной ценой среди товаров высшего сорта. Найти товар с минимальной ценой среди товаров второго сорта. Найти среднюю стоимость товаров для поставщика «Агрофирма». Найти количество товаров первого сорта. Найти общую стоимость товаров из Москвы. Продемонстрировать результаты работы.

Вариант 7: Заполнить базу данных для учета работы типографии (при необходимости). Найти продукцию типа «Этикетка» с минимальной ценой. Найти продукцию типа «Блокнот» с максимальной ценой. Найти количество продукции, проданной за наличный расчет. Найти среднюю стоимость

продукции, проданной за безналичный расчет. Найти общую стоимость продукции для заказчиков из Москвы. Продемонстрировать результаты работы.

Вариант 8: Заполнить базу данных жилищной управляющей компании (при необходимости). Найти ремонтные работы с максимальной стоимостью. Найти ремонтные работы с минимальной стоимостью для мастеров из Москвы. Найти количество не оплаченных ремонтных работ. Найти среднюю стоимость ремонтных работ, оплаченных за безналичный расчет. Найти общую стоимость ремонтных работ для мастеров из Санкт-Петербурга. Продемонстрировать результаты работы.

Вариант 9: Заполнить базу данных ремонтной мастерской (при необходимости). Найти предметы с минимальной стоимостью ремонта. Найти предметы с максимальной стоимостью ремонта, выполненные мастерами из Москвы. Найти количество отремонтированных предметов, выполненных за безналичный расчет. Найти среднюю стоимость отремонтированных предметов, выполненных за наличный расчет. Найти общую стоимость отремонтированных предметов для мастеров со стажем работы более 5 лет. Продемонстрировать результаты работы.

Вариант 10: Заполнить базу данных туристического агентства (при необходимости). Найти авиатуры с минимальной стоимостью. Найти железнодорожные туры с максимальной стоимостью. Найти количество автобусных туров. Найти среднюю стоимость туров в город Москву. Найти общую стоимость туров, выполненных руководителями туров со стажем работы более 10 лет. Продемонстрировать результаты работы.

Содержание отчета

Отчет должен содержать титульный лист, цель работы, задание, коды команд на каждом этапе выполнения работы, результаты (скриншоты) выполнения команд (работа с массивами), результаты выполнения команд

при использовании агрегатных функций (скриншоты), выводы и анализ результатов работы.

Теоретические сведения. Массивы.

Поля данных PostgreSQL вместо отдельных величин могут содержать конструкции, называемые массивами.

Чтобы создать простейшее поле-массив, включите в команду CREATE TABLE или ALTER TABLE пару квадратных скобок после имени поля. Квадратные скобки показывают, что вместо одного значения в поле может храниться массив указанного типа.

Например, команда для создания поля single_array типа type выглядит так:

single_array type[] -- Одномерный массив

Дополнительные квадратные скобки определяют многомерные массивы, то есть «массивы массивов». Пример:

multi_array type[][] -- Многомерный массив

Теоретически в квадратных скобках можно указать целое число, чтобы созданный массив имел фиксированный размер (то есть всегда состоял из n элементов по указанному измерению и не более).

Пример 1. Создание таблицы с полем-массивом

```
CREATE TABLE book1 (employee_id integer, books text[])
```

Таблица позволяет хранить в одном поле неограниченное количество названий книг. Многомерные массивы создаются аналогичным образом, просто за первой парой квадратных скобок добавляются дополнительные пары.

Пример 2. Создание таблицы с полем, содержащим многомерный массив:
CREATE TABLE f_book2 (employee_id integer, authors_and_titles text[][]);

В примере создается таблица f_book2 с целочисленным полем employee_id и многомерным массивом author_and_titles. Фактически создается массив текстовых массивов.

Вставка значений в поля-массивы.

В PostgreSQL предусмотрен специальный синтаксис вставки нескольких значений в одно поле. Этот синтаксис основан на определении массивов-констант. Массив-константа состоит из фигурных скобок, апострофов и запятых, заключенных в апострофы. Кавычки нужны только при работе с массивами строк. Таким образом, обобщенная форма массива-константы выглядит так:
'{ "текст" [, ...] }' -- массив строк
'{ число [, ...]}' -- числовой массив

Поле может определяться также как массив произвольного типа (включая типы boolean, date и time). Как правило, если для описания величины в скалярном контексте должны использоваться апострофы, в контексте массива эта величина заключается в кавычки.

Пример 3. Вставка с использованием массивов-констант
INSERT INTO book1 VALUES (1, '{"The Student\'s Guide"}');
INSERT INTO book1 VALUES (2, '{"The Student1", "Student1, Student2"}');

В примере в таблицу book1 вставляются две записи. Первая команда создает массив с одним элементом для работника с кодом 1, а вторая запись

создает массив с двумя элементами для работника с кодом 2. В обеих командах INSERT используются массивы-константы.

Даже при вставке одного элемента массив заключается в фигурные скобки. Кроме того, апостроф в названии книги (первая команда INSERT) экранируется символом \, хотя и находится внутри кавычек. Это связано с тем, что массив-константа сначала обрабатывается как одна длинная строка, а затем интерпретируется как массив по контексту целевого поля.

При вставке значений в многомерный массив все подмассивы заключаются в отдельные фигурные скобки и разделяются запятыми.

Пример 4. Вставка данных в многомерный массив

```
INSERT INTO f_book2 VALUES (1, '{{"Avtor1", "Kniga1"}, {"Avtor2", "Kniga2"}, {"Avtor3", "Kniga3"}}');
```

Многомерный массив в примере содержит три текстовых массива, каждый из которых состоит из двух элементов. Первые элементы каждого массива обозначают авторов, а вторые элементы обозначают названия книг, написанных этими авторами.

Выборка из полей-массивов

При выборке из поля-массива весь массив возвращается в формате константы. На практике чаще требуется получить некоторое подмножество элементов. Для решения этой задачи необходимо познакомиться с такими понятиями, как индексы элементов и срезы массивов.

Индексы элементов

К отдельным элементам можно обращаться при помощи индексов — целых чисел, заключенных в скобки и описывающих позицию искомого элемента в массиве. В PostgreSQL индексация в массивах начинается с 1, а не с 0.

Пример 5. Выборка отдельного элемента массива

```
SELECT books[1] FROM book1;
```

При указании индекса несуществующего элемента массива выборка возвращает NULL. Обычно для обработки таких ситуаций используется конструкция IS NOT NULL.

Пример 6. Предотвращение выборки NULL в массивах

```
SELECT books[2] FROM book1 WHERE books[2] IS NOT NULL;
```

Запрос возвращает только название, а запись с NULL исключается из выборки в результате использования секции WHERE с проверкой условия NOT NULL.

При выборке из многомерного массива за исходным индексом перечисляются дополнительные индексы.

Пример 7. Выборка из многомерного массива

```
SELECT authors_and_titles[1][1] AS author, authors_and_titles[1][2] AS title FROM f_book2;
```

В примере 7 из таблицы f_book2, созданной в примере 4, выбираются два элемента — имя автора и название книги.

Срезы

В PostgreSQL также поддерживается возможность создания срезов при выборке из массива. Срез аналогичен обычному обращению к элементам по индексу, но он описывает интервал значений. Срез задается парой целочисленных индексов, разделенных двоеточием и заключенных в квадратные скобки. Например, конструкция [2:5] описывает второй, третий, четвертый и пятый элемент заданного массива. Результат среза возвращается в виде константы-массива, которая фактически описывает подмножество элементов исходного массива.

Пример 8. Выборка с использованием среза

```
SELECT books[l:2] FROM book1;
```

В примере 8 выбираются первые два элемента массива books в записях таблицы book1.

Определение количества элементов

Чтобы узнать количество значений, хранящихся в массиве, следует воспользоваться функцией array_dims(). В качестве параметра функции передается идентификатор — имя поля-массива, для которого вызывается функция. Результат возвращается в виде строки, содержащей описание массива в синтаксисе среза.

Пример 9. Функция array_dims()

```
 SELECT array_dims(books) FROM book1;
```

В примере 9 приведен вызов функции array_dims() для поля books таблицы book1.

Обновление данных в полях-массивах

Существует три варианта модификации данных в полях-массивах.

- Полная модификация. Все содержимое массива заменяется новыми данными, заданными в виде массива-константы.
- Модификация среза. Срез (то есть интервальное подмножество элементов) заменяется новыми данными, заданными в виде массива-константы. Количество элементов в константе должно соответствовать количеству элементов в обновляемом срезе.
- Модификация элемента. Отдельный элемент массива заменяется новой константой, относящейся к базовому типу данных массива. Элемент задается индексом.

Пример 10. Полная модификация массива

```
UPDATE book1 SET books='{"The Student\'s Guide", "The Student4"}'  WHERE employee_id=1;
```

Команда UPDATE, приведенная в примере 10, заменяет все текущее содержимое массива. Этот способ подходит и для модификации среза массива. Для этого в конец идентификатора поля добавляется определение среза, например, books[l:3] означает первый, второй и третий элементы массива. На практике чаще возникает задача замены не всего массива и не среза, а отдельных элементов. При обновлении отдельного элемента к идентификатору поля присоединяется индекс, определяющий конкретный обновляемый элемент.

Агрегирование данных

В SQL существует ряд специальных стандартных функций (SQL-функций). Кроме специального случая *COUNT(*)* каждая из этих функций

оперирует совокупностью значений столбца некоторой таблицы и создает единственное значение, определяемое так:

COUNT
- число значений в столбце,
SUM
- сумма значений в столбце,
AVG
- среднее значение в столбце,
MAX
- самое большое значение в столбце,
MIN
- самое малое значение в столбце.

Для функций *SUM* и *AVG* рассматриваемый столбец должен содержать числовые значения.

Следует отметить, что здесь столбец - это столбец виртуальной таблицы, в которой могут содержаться данные не только из столбца базовой таблицы, но и данные, полученные путем функционального преобразования и (или) связывания символами арифметических операций значений из одного или нескольких столбцов. При этом выражение, определяющее столбец такой таблицы, может быть сколь угодно сложным, но не должно содержать SQL-функций (вложенность SQL-функций не допускается). Однако из SQL-функций можно составлять любые выражения.

Аргументу всех функций, кроме *COUNT(*),* может предшествовать ключевое слово *DISTINCT* (различный), указывающее, что избыточные дублирующие значения должны быть исключены перед тем, как будет применяться функция. Специальная же функция *COUNT(*)* служит для подсчета всех без исключения строк в таблице (включая дубликаты).

Пример 11. Вывести количество студентов, общую сумму стипендий, среднюю стипендию.

```
SELECT count(*), sum(student.stipendia), avg(student.stipendia)
   FROM student;
```

Контрольные вопросы

1. Каким образом можно создать одномерный массив?
2. Каким образом можно осуществить вставку значений в поля-массивы?
3. Как выполнить выборку отдельного элемента массива?
4. Какая команда дает возможность выполнить срез при выборке из массива?
5. Каким образом можно узнать количество значений, хранящихся в массиве?
6. Каким образом можно выполнить полную модификацию массива?
7. В каких ситуациях используется конструкция IS NOT NULL?
8. Какая команда позволяет вычислять максимальное значение?
9. Какая команда позволяет вычислять среднее значение?
10. Какая команда позволяет вычислять минимальное значение?
11. Для чего используется функция *COUNT(*)*?
12. В каком случае используется функция *SUM* ?

Список источников

1. Малыхина М.П. Базы данных: основы, проектирование, использование: учебное пособие для вузов по направлению подготовки "Информатика и вычислительная техника" / М. П. Малыхина. - СПб., 2006. - 517 с. : ил.
2. Хомоненко А.Д. Базы данных: учебник для вузов по техническим и экономическим специальностям / [Хомоненко А. Д., Цыганков В. М., Мальцев М. Г.] ; под ред. А. Д. Хомоненко. - М., 2006. - 736 с. : ил., табл.
3. Разработка приложений на C# с использованием СУБД PostgreSQL : учебное пособие / [И. А. Васюткина и др.] ; Новосиб. гос. техн. ун-т. - Новосибирск, 2015. - 141, [1] с. : ил., табл.. - Режим доступа: http://elibrary.nstu.ru/source?bib_id=vtls000220068
4. Дж Уорсли Дж., Дрейк Дж. PostgreSQL. Для профессионалов - СПб.: Питер, 2003. -496 с.
5. PostgreSQL (русский) [Электронный ресурс]: Документация по PostgreSQL. – Режим доступа: http://postgresql.ru.net . - загл. с экрана.
6. Википедия свободная энциклопедия (русский) [Электронный ресурс]: PostgreSQL. – Режим доступа: http://ru.wikipedia.org. - загл. с экрана.
7. Компьютерная документация (русский) [Электронный ресурс]: Функции для работы с датой и временем. – Режим доступа: http://www.hardline.ru/ . - загл. с экрана.
8. Linux и Windows: помощь админам и пользователям (русский) [Электронный ресурс]: 15 команд для управления PostgreSQL. – Режим доступа: http://www.guruadmin.ru . - загл. с экрана.