

## Практика №6

1. В базе данных bookstore создайте схему bookstore. Настройте путь поиска к этой схеме на уровне подключения к БД.
2. В схеме bookstore создайте таблицы books, authors, authorship и operations.
3. Вставьте в таблицы данные о нескольких книгах. Проверьте себя с помощью запросов.
4. В схеме bookstore создайте представления authors\_v, catalog\_v и operations\_v так, чтобы они соответствовали показанным в демонстрации.

Проверьте, что приложение стало показывать данные на вкладках «Книги», «Авторы» и «Каталог».

## Решение

1. Вспомните материал темы «Организация данных. Логическая структура».
2. Ориентируйтесь на показанный в демонстрации вывод команд \d утилиты psql.
4. Попробуйте написать запросы к базовым таблицам. Затем оформите запросы в виде представлений.

После выполнения практики обязательно сверьте свои запросы с приведенным решением. При необходимости внесите коррективы.

### 1. Схема и путь поиска

```
student$ psql bookstore
```

```
=> CREATE SCHEMA bookstore;
```

```
CREATE SCHEMA
```

```
=> ALTER DATABASE bookstore SET search_path = bookstore, public;
```

```
ALTER DATABASE
```

```
=> \c bookstore
```

```
You are now connected to database "bookstore" as user "student".
```

```
=> SHOW search_path;
```

```
search_path
```

```
-----
```

```
bookstore, public
```

(1 row)

## 2. Таблицы

Авторы:

```
=> CREATE TABLE authors(  
  author_id integer PRIMARY KEY GENERATED ALWAYS AS IDENTITY,  
  last_name text NOT NULL,  
  first_name text NOT NULL,  
  middle_name text  
);
```

CREATE TABLE

Книги:

```
=> CREATE TABLE books(  
  book_id integer PRIMARY KEY GENERATED ALWAYS AS IDENTITY,  
  title text NOT NULL  
);
```

CREATE TABLE

Авторство:

```
=> CREATE TABLE authorship(  
  book_id integer REFERENCES books,  
  author_id integer REFERENCES authors,  
  seq_num integer NOT NULL,  
  PRIMARY KEY (book_id,author_id)  
);
```

CREATE TABLE

Операции:

```
=> CREATE TABLE operations(  
  operation_id integer PRIMARY KEY GENERATED ALWAYS AS IDENTITY,  
  book_id integer NOT NULL REFERENCES books,  
  qty_change integer NOT NULL,  
  date_created date NOT NULL DEFAULT current_date  
);
```

CREATE TABLE

## 3. Данные

Авторы:

```
=> INSERT INTO authors(last_name, first_name, middle_name) VALUES
```

```
('Пушкин', 'Александр', 'Сергеевич'), ('Тургенев', 'Иван', 'Сергеевич'),  
('Стругацкий', 'Борис', 'Натанович'), ('Стругацкий', 'Аркадий', 'Натанович'),  
('Толстой', 'Лев', 'Николаевич'), ('Свифт', 'Джонатан', NULL);
```

INSERT 0 6

Книги:

```
=> INSERT INTO books(title) VALUES
```

```
('Сказка о царе Салтане'),  
('Муму'),  
('Трудно быть богом'),  
('Война и мир'),  
('Путешествия в некоторые удаленные страны мира в четырех частях:  
сочинение Лемюэля Гулливера, сначала хирурга, а затем капитана  
нескольких кораблей'),  
('Хрестоматия');
```

INSERT 0 6

Авторство:

```
=> INSERT INTO authorship(book_id, author_id, seq_num) VALUES
```

```
(1, 1, 1), (2, 2, 1), (3, 3, 2), (3, 4, 1), (4, 5, 1), (5, 6, 1), (6, 1, 1), (6, 5, 2), (6, 2, 3);
```

INSERT 0 9

Операции.

Другой способ вставки данных в таблицу — команда COPY. Она обычно используется, если нужно загрузить большой объем информации. Но в этом случае надо не забыть «передвинуть» значение последовательности:

```
=> COPY operations (operation_id, book_id, qty_change) FROM stdin;
```

```
1 1 10  
2 1 10  
3 1 -1
```

\.

COPY 3

```
=> SELECT pg_catalog.setval('operations_operation_id_seq', 3, true);
```

setval

-----

3 (1 row)

#### 4. Представления

Представление для авторов:

```
=> CREATE VIEW authors_v AS
```

```
SELECT a.author_id,  
       a.last_name || ' ' || a.first_name || coalesce(' ' || nullif(a.middle_name, ''), '') AS  
display_name
```

```
FROM authors a;
```

CREATE VIEW

Представление для каталога:

```
=> CREATE VIEW catalog_v AS
```

```
SELECT b.book_id,  
       b.title AS display_name  
FROM books b;
```

CREATE VIEW

Представление для операций:

```
=> CREATE VIEW operations_v AS SELECT book_id,
```

```
CASE  
WHEN qty_change > 0 THEN 'Поступление' ELSE 'Покупка'
```

```
END op_type,  
abs(qty_change) qty_change,  
to_char(date_created, 'DD.MM.YYYY') date_created
```

```
FROM operations  
ORDER BY operation_id;
```

CREATE VIEW

## Практика №7

1. Какие дополнительные атрибуты могут появиться у выделенных сущностей при развитии приложения?

2. Допустим, требуется хранить информацию об издательстве. Дополните таблицы.

3. Некоторые книги могут входить в серии (например, «Библиотека приключений»). Как изменится схема данных?

4. Пусть наш магазин стал торговать компьютерными комплектующими (материнскими платами, процессорами, памятью, жесткими дисками, мониторами и т. п.).

Какие сущности и какие атрибуты вы бы выделили? Учтите, что на рынке постоянно появляются новые типы оборудования со своими характеристиками.

## Решение

3. Разные издательства вполне могут иметь серии, названные одинаково.

### 1. Дополнительные атрибуты

Несколько примеров:

- Авторы: роль (автор, редактор, переводчик и т. п.);
- Книги: аннотация;
- Операции: текущий статус (оплачено, передано в службу доставки и т. п.).

### 2. Издательства

Надо добавить сущность «Издательство» с атрибутом «Название» (как минимум).

Книги связаны с издательствами отношением «многие ко многим»: книга может публиковаться в разных издательствах. Поэтому на физическом уровне потребуется промежуточная таблица «Публикации» с атрибутом «Год издания».

(Разумеется, это упрощенная модель; при желании ее можно уточнять еще очень долго.)

### 3. Серии

Добавим сущность «Серия». К серии относится не сама книга, а ее конкретная публикация, так что имеет смысл вывести «Публикацию» на уровень ER-модели и связать ее с серий отношением «один ко многим» (каждая публикация принадлежит к одной серии, каждая серия может включать несколько публикаций).

Также серия связана отношением «один ко многим» с издательством (у издательства может быть несколько серий, а каждая серия принадлежит конкретному издательству).

Остается вопрос о внесерийных изданиях. Его можно решить либо введением фиктивной серии «Без серии», либо возможностью не указывать для публикации внешний ключ серии.

#### **4. Компьютерные комплектующие**

Рассматривая каждый конкретный тип комплектующих, можно без труда выделить необходимые атрибуты. Какие-то атрибуты будут общими (скажем, фирма-производитель и название модели), а какие-то будут иметь смысл только для данного конкретного типа. Например:

- Процессор: частота;
- Монитор: диагональ, разрешение;
- Жесткий диск: типоразмер, емкость.

Проблема в том, что рынок комплектующих очень динамичен. Некоторое время назад жесткие диски определялись частотой вращения и емкостью, а сейчас важен тип (твердотельный, вращающийся, гибридный). Для мониторов во времена ЭЛТ была важна частота обновления, а сейчас важен тип матрицы. Дисководы уже никому не нужны, зато появились флеш-накопители. И так далее.

Таким образом, либо придется постоянно изменять схему данных (а, значит, и постоянно изменять приложение, которое работает с этими данными!), либо искать более универсальную модель за счет отказа от жесткой структуры и контроля согласованности.

## Практика №8

1. Создайте функцию `author_name` для формирования имени автора. Функция принимает три параметра (фамилия, имя, отчество) и возвращает строку с фамилией и инициалами.

Используйте эту функцию в представлении `authors_v`.

2. Создайте функцию `book_name` для формирования названия книги.

Функция принимает два параметра (идентификатор книги и заголовок) и возвращает строку, составленную из заголовка и списка авторов в порядке `seq_num`. Имя каждого автора формируется функцией `author_name`.

Используйте эту функцию в представлении `catalog_v`. Проверьте изменения в приложении.

## Решение

Напомним, что необходимые функции можно посмотреть в раздаточном материале «Основные типы данных и функции».

```
1. FUNCTION author_name(  
    last_name text, first_name text, surname text  
)
```

RETURNS text

Например: `author_name('Толстой','Лев','Николаевич')` → → 'Толстой Л. Н.'

```
2. FUNCTION book_name(book_id integer, title text) RETURNS text
```

Например: `book_name(3,'Трудно быть богом')` →  
→ 'Трудно быть богом. Стругацкий А. Н., Стругацкий Б. Н.'

Все инструменты позволяют «непосредственно» редактировать хранимые функции. Например, в `psql` есть команда `\ef`, открывающая текст функции в редакторе и сохраняющая изменения в базу.

Такой возможностью лучше не пользоваться (или как минимум не злоупотреблять). Нормально построенный процесс разработки предполагает хранение всего кода в файлах под версионным контролем. При необходимости изменить функцию файл редактируется и выполняется (с помощью `psql` или средствами IDE). Если же менять определение функций сразу в БД, изменения легко потерять.

## 1. Функция author\_name

```
=> CREATE OR REPLACE FUNCTION author_name(  
    last_name text,  
    first_name text,  
    middle_name text  
) RETURNS text  
AS $$  
SELECT last_name || ' ' ||  
    left(first_name, 1) || ' ' ||  
    CASE WHEN middle_name != '' -- подразумевает NOT NULL  
        THEN ' ' || left(middle_name, 1) || ' '  
        ELSE ''  
    END  
$$ IMMUTABLE LANGUAGE sql;  
CREATE FUNCTION
```

Категория изменчивости — immutable. Функция всегда возвращает одинаковое значение при одних и тех же входных параметрах.

```
=> CREATE OR REPLACE VIEW authors_v AS  
SELECT a.author_id,  
    author_name(a.last_name, a.first_name, a.middle_name) AS display_name  
FROM authors a  
ORDER BY display_name;
```

CREATE VIEW

## 2. Функция book\_name

```
=> CREATE OR REPLACE FUNCTION book_name(book_id integer, title text)  
RETURNS text  
AS $$  
SELECT title || ' ' ||  
    string_agg(  
        author_name(a.last_name, a.first_name, a.middle_name), ', '  
        ORDER BY ash.seq_num  
    )  
FROM authors a  
    JOIN authorship ash ON a.author_id = ash.author_id  
WHERE ash.book_id = book_name.book_id;  
$$ STABLE LANGUAGE sql;
```

CREATE FUNCTION



Категория изменчивости — stable. Функция возвращает одинаковое значение при одних и тех же входных параметрах, но только в рамках одного SQL-запроса.

```
=> DROP VIEW IF EXISTS catalog_v;  
DROP VIEW
```

```
=> CREATE VIEW catalog_v AS  
SELECT b.book_id,  
       book_name(b.book_id, b.title) AS display_name  
FROM   books b  
ORDER BY display_name;  
CREATE VIEW
```

## Практика №9

1. Напишите функцию, выдающую случайное время, равномерно распределенное в указанном отрезке.

Начало отрезка задается временной отметкой (timestampz), конец — либо временной отметкой, либо интервалом (interval).

2. В таблице хранятся номера автомобилей, введенные кое-как: встречаются как латинские, так и русские буквы в любом регистре; между буквами и цифрами могут быть пробелы. Считая, что формат номера «буква три-цифры две-буквы», напишите функцию, выдающую число уникальных номеров. Например, «K 123 ХМ» и «k123хм» считаются равными.

3. Напишите функцию, находящую корни квадратного уравнения.

## Решение

Во всех заданиях обратите особое внимание на категорию изменчивости функций.

2. Сначала напишите функцию «нормализации» номера, то есть приводящую номер к какому-нибудь стандартному виду. Например, без пробелов и только заглавными латинскими буквами.

В номерах используются только 12 русских букв, имеющих латинские аналоги схожего начертания, а именно: АВЕКМНОРСТУХ.

3. Для уравнения  $y = ax^2 + bx + c$ :  
дискриминант  $D = b^2 - 4ac$ .

- при  $D > 0$  два корня  $x_{1,2} = (-b \pm \sqrt{D}) / 2a$
- при  $D = 0$  один корень  $x = -b / 2a$  (в качестве  $x^2$  можно вернуть null)
- при  $D < 0$  корней нет (оба корня null).

### 1. Случайная временная отметка

Функция с двумя временными отметками:

```
=> CREATE FUNCTION rnd_timestamp(t_start timestampz, t_end timestampz)
RETURNS timestampz
AS $$
    SELECT t_start + (t_end - t_start) * random();
$$ VOLATILE LANGUAGE sql;
```

CREATE FUNCTION

Категория изменчивости — volatile. Используется функция random, поэтому функция будет возвращать разные значения при одних и тех же входных параметрах.

```
=> SELECT current_timestamp,  
       rnd_timestamp(  
         current_timestamp,  
         current_timestamp + interval '1 hour'  
       )  
FROM generate_series(1,10);
```

current_timestamp		rnd_timestamp
-----+-----		
2021-10-19 23:27:34.592997+03		2021-10-19 23:39:34.277715+03
2021-10-19 23:27:34.592997+03		2021-10-20 00:08:01.566791+03
2021-10-19 23:27:34.592997+03		2021-10-20 00:16:25.454624+03
2021-10-19 23:27:34.592997+03		2021-10-20 00:23:54.028175+03
2021-10-19 23:27:34.592997+03		2021-10-19 23:39:26.902572+03
2021-10-19 23:27:34.592997+03		2021-10-20 00:18:45.160602+03
2021-10-19 23:27:34.592997+03		2021-10-20 00:00:20.545024+03
2021-10-19 23:27:34.592997+03		2021-10-19 23:36:57.254788+03
2021-10-19 23:27:34.592997+03		2021-10-20 00:05:48.622317+03
2021-10-19 23:27:34.592997+03		2021-10-19 23:56:35.507319+03

(10 rows)

Вторую функцию (с параметром-интервалом) можно определить через первую:

```
=> CREATE FUNCTION rnd_timestamp(t_start timestamptz, t_delta interval)  
RETURNS timestamptz  
AS $$  
  SELECT rnd_timestamp(t_start, t_start + t_delta);  
$$ VOLATILE LANGUAGE sql;
```

CREATE FUNCTION

```
=> SELECT rnd_timestamp(current_timestamp, interval '1 hour');
```

rnd_timestamp
-----
2021-10-19 23:55:00.112021+03

(1 row)

## 2. Автомобильные номера

Создадим таблицу с номерами.

```
=> CREATE TABLE cars(  
    id integer PRIMARY KEY GENERATED ALWAYS AS IDENTITY,  
    regnum text  
);
```

CREATE TABLE

```
=> INSERT INTO cars(regnum) VALUES  
    ('K 123 XM'), ('k123xm'), ('A 098BC');
```

INSERT 0 3

Функция нормализации:

```
=> CREATE FUNCTION normalize(regnum text) RETURNS text  
AS $$  
    SELECT upper(translate(regnum, 'ABEKMHOПCTYX ', 'ABEKMHOПCTYX'));  
$$ IMMUTABLE LANGUAGE sql;
```

CREATE FUNCTION

Категория изменчивости — immutable. Функция всегда возвращает одинаковое значение при одних и тех же входных параметрах.

```
=> SELECT normalize(regnum) FROM cars;
```

normalize

-----

K123XM

K123XM

A098BC

(3 rows)

Теперь легко найти дубликаты:

```
=> CREATE FUNCTION num_unique() RETURNS bigint  
AS $$  
    SELECT count(DISTINCT normalize(regnum))  
    FROM cars;  
$$ STABLE LANGUAGE sql;
```

CREATE FUNCTION

```
=> SELECT num_unique();
```

num\_unique

-----

2 (1 row)

### 3. Корни квадратного уравнения

```
=> CREATE FUNCTION square_roots(  
  a float,  
  b float,  
  c float,  
  x1 OUT float,  
  x2 OUT float  
)  
AS $$  
WITH discriminant(d) AS (  
  SELECT b*b - 4*a*c  
)  
SELECT CASE WHEN d >= 0.0 THEN (-b + sqrt(d))/2/a END,  
       CASE WHEN d > 0.0 THEN (-b - sqrt(d))/2/a END  
FROM discriminant;  
$$ IMMUTABLE LANGUAGE sql;
```

CREATE FUNCTION

Категория изменчивости — immutable. Функция всегда возвращает одинаковое значение при одних и тех же входных параметрах.

```
=> SELECT square_roots(1, 0, -4);
```

square\_roots

-----

(2,-2)

(1 row)

```
=> SELECT square_roots(1, -4, 4);
```

square\_roots

-----

(2,) (1 row)

```
=> SELECT square_roots(1, 1, 1);
```

square\_roots

-----

(,)

(1 row)

## Практика №10

1. В таблице authors имена, фамилии и отчества авторов

по смыслу должны быть уникальны, но это условие никак не проверяется. Напишите процедуру, удаляющую возможные дубликаты авторов.

2. Чтобы необходимость в подобной процедуре не возникала, создайте ограничение целостности, которое не позволит появляться дубликатам в будущем.

### Решение

#### 1. Устранение дубликатов

В целях проверки добавим второго Пушкина:

```
=> INSERT INTO authors(last_name, first_name, middle_name)
VALUES ('Пушкин', 'Александр', 'Сергеевич');
```

INSERT 0 1

```
=> SELECT last_name, first_name, middle_name, count(*)
FROM authors
GROUP BY last_name, first_name, middle_name;
```

```
last_name | first_name | middle_name | count
-----+-----+-----+-----
```

```
Свифт    | Джонатан  |              |
Стругацкий | Борис    | Натанович   |
Пушкин   | Александр | Сергеевич   |
Стругацкий | Аркадий   |              |
Толстой  | Лев       |              |
Тургенев | Иван      |              |
```

(6 rows)

Задачу устранения дубликатов можно решить разными способами. Например, так:

```
=> CREATE PROCEDURE authors_dedup()
AS $$
DELETE FROM authors
WHERE author_id IN (
```

```

SELECT author_id
FROM (
    SELECT author_id,
           row_number() OVER (
               PARTITION BY first_name, last_name, middle_name
               ORDER BY author_id
           ) AS rn
FROM authors )t
WHERE t.rn > 1
);
$$ LANGUAGE sql;

```

CREATE PROCEDURE

=> **CALL** authors\_dedup();

CALL

```

=> SELECT last_name, first_name, middle_name, count(*)
FROM authors
GROUP BY last_name, first_name, middle_name;

```

```

last_name | first_name | middle_name | count
-----+-----+-----+-----

```

```

| Натанович
| Николаевич | 1
| Сергеевич  | 1

```

```

| 1 | 1 | 2 | 1

```

```

Свифт      | Джонатан |
Стругацкий | Борис    | Натанович
Пушкин     | Александр | Сергеевич
Стругацкий | Аркадий
Толстой    | Лев
Тургенев   | Иван

```

(6 rows)

## 2. Ограничение целостности

Создать подходящее ограничение целостности мешает тот факт, что отчество может быть неопределенным (NULL). Неопределенные значения считаются различными, поэтому ограничение



**UNIQUE**(first\_name, last\_name, middle\_name)

не помешает добавить второго Джонатана Свифта без отчества. Задачу можно решить, создав уникальный индекс:

```
=> CREATE UNIQUE INDEX authors_full_name_idx ON authors(  
    last_name, first_name, coalesce(middle_name, "  
);
```

CREATE INDEX

```
| Натанович  
| Николаевич | 1  
| Сергеевич  | 1
```

```
| 1 | 1 | 1 | 1
```

Проверим:

```
=> INSERT INTO authors(last_name, first_name)  
VALUES ('Свифт', 'Джонатан');
```

ERROR: duplicate key value violates unique constraint "authors\_full\_name\_idx"  
DETAIL: Key (last\_name, first\_name, COALESCE(middle\_name, "::text"))=(Свифт, Джонатан, ) already exists.

```
=> INSERT INTO authors(last_name, first_name, middle_name)  
VALUES ('Пушкин', 'Александр', 'Сергеевич');
```

ERROR: duplicate key value violates unique constraint "authors\_full\_name\_idx"  
DETAIL: Key (last\_name, first\_name, COALESCE(middle\_name, "::text"))=(Пушкин, Александр, Сергеевич) already exists.

## Практика №11

1. Получится ли создать в одной и той же схеме и имеющие одно и то же имя: 1) процедуру с одним входным параметром, 2) функцию с одним входным параметром того же типа, возвращающую некоторое значение? Проверьте.
2. В таблице хранятся вещественные числа (например, результаты каких-либо измерений). Напишите процедуру нормализации данных, которая умножает все числа на определенный коэффициент так, чтобы все значения попали в интервал от -1 до 1.

Процедура должна возвращать выбранный коэффициент.

### Решение

2. В качестве коэффициента возьмите максимальное абсолютное значение из таблицы.

### 1. Перегрузка процедур и функций

Не получится, так как в сигнатуру подпрограммы входит только имя и тип входных параметров (возвращаемое значение игнорируется), и при этом процедуры и функции имеют общее пространство имен.

```
=> CREATE PROCEDURE test(IN x integer)
AS $$
    SELECT 1;
$$ LANGUAGE sql;
```

CREATE PROCEDURE

```
=> CREATE FUNCTION test(IN x integer) RETURNS integer
AS $$
    SELECT 1;
$$ LANGUAGE sql;
```

ERROR: function "test" already exists with same argument types

В некоторых сообщениях, как и в этом, вместо слова «процедура» используется «функция», поскольку во многом они устроены одинаково.

### 2. Нормализация данных

Таблица с тестовыми данными:

```
=> CREATE TABLE samples(a float);
```

CREATE TABLE

```
=> INSERT INTO samples(a)
    SELECT (0.5 - random())*100 FROM generate_series(1,10);
```

INSERT 0 10

Процедуру можно написать, используя один SQL-оператор:

```
=> CREATE PROCEDURE normalize_samples(INOUT coeff float)
AS $$
    WITH c(coeff) AS (
        SELECT 1/max(abs(a))
        FROM samples
    ),
    upd AS (
        UPDATE samples
        SET a = a * c.coeff
        FROM c
    )
    SELECT coeff FROM c;
$$ LANGUAGE sql;
```

CREATE PROCEDURE

```
=> CALL normalize_samples(NULL);
```

```
      coeff
-----
0.020675409803181007
(1 row)
```

```
=> SELECT * FROM samples;
```

```
      a
-----
-0.43176889021778525
-0.14976273061774292
0.18216316126192833
-0.6185568897163543
1
0.28503155546210784
-0.8263285266664495
```

-0.8434720239121521  
0.3123270208872319  
-0.5060415892614566

(10 rows)

## Практика №12

1. Создайте функцию `onhand_qty` для подсчета имеющихся в наличии книг. Функция принимает параметр составного типа `books` и возвращает целое число.

Используйте эту функцию в представлении `catalog_v` в качестве «вычисляемого поля».

Проверьте, что приложение отображает количество книг.

2. Создайте табличную функцию `get_catalog` для поиска книг. Функция принимает значения полей формы поиска

(«имя автора», «название книги», «есть на складе»)

и возвращает подходящие книги в формате `catalog_v`.

Проверьте, что в «Магазине» начал работать поиск и просмотр.

## Решение

1.

```
FUNCTION onhand_qty(book books) RETURNS integer
```

2.

```
FUNCTION get_catalog(
```

```
    author_name text, book_title text, in_stock boolean
```

```
)
```

```
RETURNS TABLE(
```

```
    book_id integer, display_name text, onhand_qty integer
```

```
)
```

При решении хотелось бы воспользоваться уже готовым представлением `catalog_v`, просто наложив ограничения на строки. Но в этом представлении и название книги, и авторы находятся в одном поле, к тому же в сокращенном виде. Очевидно, что поиск автора «Лев» по полю «Л .Н. Толстой» не даст результата.

Можно было бы повторить в функции `get_catalog` запрос из `catalog_v`, но это дублирование кода, что плохо. Поэтому расширьте представление `catalog_v`, добавив в него дополнительные поля: заголовок книги и полный список авторов.

Проверьте, что корректно обрабатываются пустые поля на форме. Когда клиент вызывает функцию `get_catalog`, передает ли он в этом случае пустые строки или неопределенные значения?

## 1. Функция `onhand_qty`

```
=> CREATE OR REPLACE FUNCTION onhand_qty(book books) RETURNS integer
AS $$
    SELECT coalesce(sum(o.qty_change),0)::integer
    FROM operations o
    WHERE o.book_id = book.book_id;
$$ STABLE LANGUAGE sql;
```

CREATE FUNCTION

```
=> DROP VIEW IF EXISTS catalog_v;
```

DROP VIEW

```
=> CREATE VIEW catalog_v AS
SELECT b.book_id,
       book_name(b.book_id, b.title) AS display_name,
       b.onhand_qty
FROM   books b
ORDER BY display_name;
```

CREATE VIEW

## 2. Функция `get_catalog`

Расширяем `catalog_v` заголовком книги и полным списком авторов (приложение игнорирует неизвестные ему поля). Функция, возвращающая полный список авторов:

```
=> CREATE OR REPLACE FUNCTION authors(book books) RETURNS text
AS $$
    SELECT string_agg(
        a.last_name ||
        ' ' ||
        a.first_name ||
        coalesce(' ' || nullif(a.middle_name, ''), ''),
        ','
    ORDER BY ash.seq_num
    )
```

```

FROM authors a
  JOIN authorship ash ON a.author_id = ash.author_id
WHERE ash.book_id = book.book_id;
$$ STABLE LANGUAGE sql;

```

CREATE FUNCTION

Используем эту функцию в представлении catalog\_v:

=> DROP VIEW catalog\_v;

DROP VIEW

```

=> CREATE VIEW catalog_v AS
SELECT b.book_id,
       b.title,
       b.onhand_qty,
       book_name(b.book_id, b.title) AS display_name,
       b.authors
FROM books b
ORDER BY display_name;

```

CREATE VIEW

Функция get\_catalog теперь использует расширенное представление:

```

=> CREATE OR REPLACE FUNCTION get_catalog(
  author_name text,
  book_title text,
  in_stock boolean
)
RETURNS TABLE(book_id integer, display_name text, onhand_qty integer)
AS $$
  SELECT cv.book_id,
         cv.display_name,
         cv.onhand_qty
  FROM catalog_v cv
  WHERE cv.title ILIKE '%'||coalesce(book_title,'')||'%'
  AND cv.authors ILIKE '%'||coalesce(author_name,'')||'%'
  AND (in_stock AND cv.onhand_qty > 0 OR in_stock IS NOT TRUE)
  ORDER BY display_name;
$$ STABLE LANGUAGE sql;

```

CREATE FUNCTION

## Практика №13

1. Напишите функцию, переводящую строку, содержащую число в шестнадцатеричной системе, в обычное целое число.
2. Добавьте в функцию второй необязательный параметр — основание системы счисления (по умолчанию — 16).
3. Табличная функция `generate_series` не работает со строковыми типами. Предложите свою функцию для генерации последовательностей строк из заглавных английских букв.

## Решение

1. Например:

`convert('FF') → 255`

Для решения пригодятся: табличная функция `regexp_split_to_table`, функции `upper` и `reverse`, конструкция `WITH ORDINALITY`.

Другое решение возможно с помощью рекурсивного запроса. Проверить реализацию можно, используя шестнадцатеричные

константы: `SELECT X'FF'::integer`; 2. Например:

`convert('0110',2) → 6`

3. Считайте, что на вход подаются строки равной длины. Например:

`generate_series('AA','ZZ') →`

`→ 'AA'`

`'AB'`

`'AC'`

`...`

`'ZY'`

`'ZZ'`

## 1. Функция для шестнадцатеричной системы



Сначала для удобства определим функцию для одной цифры.

```
=> CREATE FUNCTION digit(d text) RETURNS integer
AS $$
SELECT ascii(d) - CASE
    WHEN d BETWEEN '0' AND '9' THEN ascii('0')
    ELSE ascii('A') - 10
END;
$$ IMMUTABLE LANGUAGE sql;
```

CREATE FUNCTION

Теперь основная функция:

```
=> CREATE FUNCTION convert(hex text) RETURNS integer
AS $$
WITH s(d,ord) AS (
SELECT *
FROM regexp_split_to_table(reverse(upper(hex)), '') WITH ORDINALITY
)
SELECT sum(digit(d) * 16^(ord-1))::integer
FROM s;
$$ IMMUTABLE LANGUAGE sql;
```

CREATE FUNCTION

```
=> SELECT convert('0FE'), convert('0FF'), convert('100');
```

```
convert | convert | convert
-----+-----+-----
    254 |    255 |    256
(1 row)
```

## 2. Функция для любой системы счисления

Предполагаем, что основание системы счисления от 2 до 36, то есть число записывается цифрами от 0 до 9, либо буквами от A до Z. В этом случае изменения минимальные.

```
=> DROP FUNCTION convert(text);
```

DROP FUNCTION

```
=> CREATE FUNCTION convert(num text, radix integer DEFAULT 16)
RETURNS integer
AS $$
```

```

WITH s(d,ord) AS (
SELECT *
  FROM regexp_split_to_table(reverse(upper(num)),') WITH ORDINALITY
)
SELECT sum(digit(d) * radix^(ord-1))::integer
FROM s;
$$ IMMUTABLE LANGUAGE sql;

```

CREATE FUNCTION

```
=> SELECT convert('0110',2), convert('0FF', convert('Z',36);
```

```

convert | convert | convert
-----+-----+-----
      6 |    255 |    35
(1 row)

```

### 3. Функция generate\_series для строк

Сначала напомним вспомогательные функции, переводящие строку в числовое представление и обратно. Первая очень похожа на функцию из предыдущего задания:

```

=> CREATE FUNCTION text2num(s text) RETURNS integer
AS $$
WITH s(d,ord) AS (
  SELECT *
  FROM regexp_split_to_table(reverse(s),') WITH ORDINALITY
)
SELECT sum( (ascii(d)-ascii('A')) * 26^(ord-1))::integer
FROM s;
$$ IMMUTABLE LANGUAGE sql;

```

CREATE FUNCTION

Обратную функцию напомним с помощью рекурсивного запроса:

```

=> CREATE FUNCTION num2text(n integer, digits integer) RETURNS text
AS $$
WITH RECURSIVE r(num,txt, level) AS (
  SELECT n/26, chr( n%26 + ascii('A') )::text, 1
  UNION ALL
  SELECT r.num/26, chr( r.num%26 + ascii('A') ) || r.txt, r.level+1
  FROM r
  WHERE r.level < digits

```

```
)
SELECT r.txt FROM r WHERE r.level = digits;
$$ IMMUTABLE LANGUAGE sql;
```

CREATE FUNCTION

```
=> SELECT num2text( text2num('ABC'), length('ABC') );
```

num2text

-----

ABC

(1 row)

Теперь функцию generate\_series для строк можно переписать, используя generate\_series для целых чисел.

```
=> CREATE FUNCTION generate_series(start text, stop text)
RETURNS SETOF text
AS $$
    SELECT num2text( g.n, length(start) )
    FROM generate_series(text2num(start), text2num(stop)) g(n);
$$ IMMUTABLE LANGUAGE sql;
```

CREATE FUNCTION

```
=> SELECT generate_series('AZ','BC');
```

generate\_series

-----

AZ

BA

BB

BC

(4 rows)

## Практика №14

1. Измените функцию `book_name` так, чтобы длина возвращаемого значения не превышала 45 символов. Если название книги при этом обрезается, оно должно завершаться на троеточие.

Проверьте реализацию в SQL;

при необходимости добавьте книги с длинными названиями.

2. Снова измените функцию `book_name` так, чтобы избыточно длинное название уменьшалось на целое слово.

Проверьте реализацию.

## Решение

1. Например:

Путешествия в некоторые удалённые страны мира в четырёх частях:

сочинение Лемюэля Гулливера, сначала хирурга, а затем капитана

нескольких кораблей →

→ Путешествия в некоторые удалённые страны м...

Вот некоторые случаи, которые имеет смысл проверить:

- длина названия меньше 45 символов (не должно измениться);
- длина названия ровно 45 символов (не должно измениться);
- длина названия 46 символов (от названия должны быть отрезаны 4 символа, т. к. добавятся еще три точки).

Лучше всего написать и отладить отдельную функцию укорачивания, которую затем использовать в `book_name`. Это полезно и по другим соображениям:

- такая функция может пригодиться где-то еще;
- каждая функция будет выполнять ровно одну задачу.

2. Например:

Путешествия в некоторые удалённые страны мира в четырёх частях:

сочинение Лемюэля Гулливера, сначала хирурга, а затем капитана

нескольких кораблей →

→ Путешествия в некоторые удалённые страны...

Как поведет себя ваша реализация, если название состоит из одного длинного слова без пробелов?

## 1. Укорачивание названия книги

Напишем более универсальную функцию, принимающую строку, максимальную длину и суффикс, добавляемый при укорачивании. Это не потребует усложнения кода, и позволит обойтись без «магических констант».

```
=> CREATE OR REPLACE FUNCTION shorten( s text,
```

```
max_len integer DEFAULT 45,
```

```
suffix text DEFAULT '...')
```

```
RETURNS text AS $$DECLARE
```

```
suffix_len integer := length(suffix);BEGIN
```

```
RETURN CASE WHEN length(s) > max_len
```

```
THEN left(s, max_len - suffix_len) || suffixELSE s
```

```
END;END;
```

```
$$ IMMUTABLE LANGUAGE plpgsql;
```

```
CREATE FUNCTION
```

Проверим:

```
=> SELECT shorten(
```

```
'Путешествия в некоторые удаленные страны мира в четырех частях:  
сочинение Лемюэля Гулливера, сначала хирурга, а затем капитана  
нескольких кораблей'
```

```
);
```

```
shorten
```

```
-----
```

```
Путешествия в некоторые удаленные страны м...
```

```
(1 row)
```

```
=> SELECT shorten(
```

```
'Путешествия в некоторые удаленные страны мира в четырех частях:
```

сочинение Лемюзля Гулливера, сначала хирурга, а затем капитана  
нескольких кораблей',30

);

shorten

-----

Путешествия в некоторые уда...

(1 row)

Используем написанную функцию:

```
=> CREATE OR REPLACE FUNCTION book_name(book_id integer, title  
text) RETURNS text
```

```
AS $$
```

```
SELECT
```

```
FROM WHERE
```

```
author_name(a.last_name, a.first_name, a.middle_name), ', '
```

```
ORDER BY ash.seq_num )
```

```
authors a
```

```
JOIN authorship ash ON a.author_id = ash.author_id ash.book_id =
```

```
book_name.book_id;
```

```
shorten(book_name.title) || '. ' ||
```

```
string_agg(
```

```
$$ STABLE LANGUAGE sql;
```

CREATE FUNCTION

## 2. Укорачивание названия книги по словам

```
=> CREATE OR REPLACE FUNCTION shorten( s text,
```

```
max_len integer DEFAULT 45,
```

```
suffix text DEFAULT '...')
```

```
RETURNS text AS $$
```

```
DECLARE
```

```
suffix_len integer := length(suffix); short text := suffix;
```

```
pos integer;
```

```
BEGIN
```

```
IF length(s) < max_len THEN
```

```
RETURN s;END IF;
```

```
FOR pos in 1 .. least(max_len-suffix_len+1, length(s))
```

```
LOOP
```

```
IF substr(s,pos-1,1) != ' ' AND substr(s,pos,1) = ' ' THEN short := left(s, pos-1) ||  
suffix;
```

```
END IF;  
END LOOP;
```

```
RETURN short;  
END;
```

```
$$ IMMUTABLE LANGUAGE plpgsql;
```

```
CREATE FUNCTION
```

Проверим:

```
=> SELECT shorten(  
    'Путешествия в некоторые удаленные страны мира в четырех частях:  
    сочинение Лемюэля Гулливера, сначала хирурга, а затем капитана  
    нескольких кораблей'  
);
```

shorten

-----  
Путешествия в некоторые удаленные страны...

(1 row)

```
=> SELECT shorten(  
    'Путешествия в некоторые удаленные страны мира в четырех частях:  
    сочинение Лемюэля Гулливера, сначала хирурга, а затем капитана  
    нескольких кораблей',30  
);
```

shorten

-----  
Путешествия в некоторые...

(1 row)

## Практика №15

1. Напишите PL/pgSQL-функцию, которая возвращает строку заданной длины из случайных символов.

2. Задача про игру в «наперстки».

В одном из трех наперстков спрятан выигрыш.

Игрок выбирает один из этих трех. Ведущий убирает один из двух оставшихся наперстков (обязательно пустой) и дает игроку возможность поменять решение, то есть выбрать второй из двух оставшихся.

Есть ли смысл игроку менять выбор или нет смысла менять первоначальный вариант?

Задание: используя PL/pgSQL, посчитайте вероятность выигрыша и для начального выбора, и для измененного.

## Решение

Предварительно можно создать функцию `rnd_integer`, которая возвращает случайное целое число в заданном диапазоне. Функция будет полезна для решения обоих заданий.

Например: `rnd_integer(30, 1000) → 616`

1. Помимо длины строки на вход функции можно подавать список допустимых символов. По умолчанию, это могут быть все символы алфавита, числа и некоторые знаки. Для определения случайных символов из списка можно использовать функцию `rnd_integer`. Объявление функции может быть таким:

```
CREATE FUNCTION rnd_text(  
    len int,  
    list_of_chars text DEFAULT  
'АБВГДЕЁЖЗИЙКЛМНОПРСТУФХЦЧШЩЪЫЬЭЮЯабвгдеёжзийклмнопрстуфхцчш  
щъыьэю  
яABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz_0123456789'  
) RETURNS text AS ...
```

Пример вызова: `rnd_text(10) → 'ЛждфбЁ_ООJ'`

2. Для решения можно использовать анонимный блок.



Сначала нужно реализовать одну игру и посмотреть, какой вариант выиграл: начальный или измененный. Для загадывания и угадывания одного из трех наперстков можно использовать `rnd_integer(1,3)`.

Затем игру поместить в цикл и «сыграть», например, 1000 раз, подсчитывая, какой вариант сколько раз победил. В конце через RAISE NOTICE вывести значения счетчиков и выявить победивший вариант (или отсутствие такового).

## 1. Случайная строка заданного размера

Вначале определим вспомогательную функцию для получения случайного целого числа в заданном диапазоне. Такую функцию легко написать на чистом SQL, но здесь представлен вариант на PL/pgSQL:

```
=> CREATE FUNCTION rnd_integer(min_value integer, max_value integer)
RETURNS integer
AS $$
DECLARE
```

```
retval integer; BEGIN
```

```
IF max_value <= min_value THEN RETURN NULL;
```

```
END IF;
```

```
retval := floor(
(max_value+1 - min_value)*random()
```

```
::integer RETURN retval;
```

```
END;
```

```
$$ STRICT LANGUAGE
```

```
CREATE FUNCTION
```

Проверяем работу:

```
+ min_value; plpgsql;
```

```
=> SELECT rnd_integer(0,1) as "0 - 1",rnd_integer(1,365) as "1 - 365",
```

```
rnd_integer(-30,30) as "-30 - +30"FROM generate_series(1,10);
```

```
0 - 1 | 1 - 365 | -30 - +30 -----+-----+-----
```

```
1| 1| 0| 1| 1| 0| 0| 1| 0| 1|
```

(10 rows)

44| 15 254 | -2 90| 16 348 | -3 94| -17 132 | 3 214 | -27 152 | 25 318 | -18 19| 17

Функция гарантирует равномерное распределение случайных значений по всему диапазону, включая граничные значения:

```
=> SELECT rnd_value, count(*) FROM (  
SELECT rnd_integer(1,5) AS rnd_value  
FROM generate_series(1,100000) ) AS t  
GROUP BY rnd_value ORDER BY rnd_value;
```

rnd\_value | count -----+-----

1 | 20030 2 | 19859 3 | 20071 4 | 20116 5 | 19924

(5 rows)

Теперь можно приступить к функции для получения случайной строки заданного размера. Будем использовать функцию rnd\_integer для получения случайного символа из списка.

```
=> CREATE FUNCTION rnd_text( len int,  
list_of_chars text DEFAULT  
'АБВГДЕЁЖЗИЙКЛМНОПРСТУФХЦЧШЩЪЫЬЭЮЯабвгдеёжзийклмнопрстуфх  
цшщъыьэюяABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz  
_0123456789') RETURNS text  
AS $$  
DECLARE  
len_of_list CONSTANT integer := length(list_of_chars); i integer;  
retval text := "";  
BEGIN  
FOR i IN 1 .. len LOOP  
-- добавляем к строке случайный символ  
retval := retval ||  
substr(list_of_chars, rnd_integer(1,len_of_list),1);  
END LOOP;  
RETURN retval;  
END;
```

```
$$ STRICT LANGUAGE plpgsql;
```

```
CREATE FUNCTION
```

```
Проверяем:
```

```
=> SELECT rnd_text(rnd_integer(1,30)) FROM generate_series(1,10);
```

```
rnd_text ----- юDNfU8ЧnzseypJ0урищёфХДКИВWTс
```

```
Бf1EOЁвОqiAooЩЁТшEPFЩх
```

```
3
```

```
1YЙЭsД3чИpNK
```

```
гтСЭwxKPЩе уытBakOWOЛfOW3rkeR46tQп4GL ЖИaи4fNSЦeРюзшЛ2bR
```

```
вЛОЗьBoУыIvxbQyЗьзиёБj9pdёMP hДCdCR7eОбу22лПоWr
```

```
Р
```

```
(10 rows)
```

## 2. Игра в наперстки

Для загадывания и угадывания наперстка используем rnd\_integer(1,3).

```
=> DO $$DECLARE
```

```
x integer;
```

```
choice integer;
```

```
new_choice integer;
```

```
remove integer;
```

```
total_games integer := 1000; old_choice_win_counter integer := 0;
```

```
new_choice_win_counter integer := 0;
```

```
BEGIN
```

```
FOR i IN 1 .. total_games LOOP
```

```
-- Загадываем выигрышный наперсток
```

```
x := rnd_integer(1,3);-- Игрок делает выбор
```

```
choice := rnd_integer(1,3);
```

```
-- Убираем один неверный ответ, кроме выбора игрока
```

```
FOR i IN 1 .. 3 LOOP
```

```
IF i NOT IN (x, choice) THEN
```

```
remove := i;
```

```
EXIT;
```

```
END IF;
```

```

END LOOP;

-- Нужно ли игроку менять свой выбор?

-- Измененный выбор

FOR i IN 1 .. 3 LOOP

IF i NOT IN (remove, choice) THEN
new_choice := i;
EXIT;

END IF;
END LOOP;

-- Или начальный, или новый выбор обязательно выиграют

IF choice = x THEN
old_choice_win_counter := old_choice_win_counter + 1;

ELSIF new_choice = x THEN
new_choice_win_counter := new_choice_win_counter + 1;

END IF;
END LOOP;

RAISE NOTICE 'Выиграл начальный выбор: % из %', old_choice_win_counter,
total_games;

RAISE NOTICE 'Выиграл измененный выбор: % из %',
new_choice_win_counter, total_games;

END; $$;

```

NOTICE: Выиграл начальный выбор: 351 из 1000 NOTICE: Выиграл измененный выбор: 649 из 1000 DO

Вначале мы выбираем 1 из 3, поэтому вероятность начального выбора 1/3. Если же выбор изменить, то изменится и вероятность на противоположные 2/3. Таким образом, вероятность выиграть при смене выбора выше. Поэтому есть смысл выбор поменять.

## Практика №16

1. Напишите функцию `add_author` для добавления новых авторов. Функция должна принимать три параметра (фамилия, имя, отчество) и возвращать идентификатор нового автора.

Проверьте, что приложение позволяет добавлять авторов.

2. Напишите функцию `buy_book` для покупки книги. Функция принимает идентификатор книги и уменьшает количество таких книг на складе на единицу. Возвращаемое значение отсутствует.

## Решение

1.

```
FUNCTION add_author(last_name text, first_name text, surname text)
RETURNS integer
```

3.

```
FUNCTION buy_book(book_id integer)
RETURNS void
```

Вы можете обратить внимание, что при покупке книг приложение позволяет «уйти в минус». Если бы количество книг хранилось в столбце, простым и хорошим решением было бы сделать ограничение CHECK. Но в нашем случае количество рассчитывается, и мы отложим написание проверки до темы «Триггеры».

### 1. Функция `add_author`

```
=> CREATE OR REPLACE FUNCTION add_author(
    last_name text,
    first_name text,
    middle_name text
) RETURNS integer
AS $$
DECLARE
    author_id integer;
BEGIN
    INSERT INTO authors(last_name, first_name, middle_name)
```

```
VALUES (last_name, first_name, middle_name)
RETURNING authors.author_id INTO author_id;
RETURN author_id;
END;
```

```
$$ VOLATILE LANGUAGE plpgsql;
```

```
CREATE FUNCTION
```

## 2. Функция buy\_book

```
=> CREATE OR REPLACE FUNCTION buy_book(book_id integer)
RETURNS void
AS $$
BEGIN
    INSERT INTO operations(book_id, qty_change)
    VALUES (book_id, -1);
END;
$$ VOLATILE LANGUAGE plpgsql;
CREATE FUNCTION
```

## Практика №17

Напишите игру, в которой сервер пытается угадать загаданное пользователем животное, задавая последовательные уточняющие вопросы, на которые можно отвечать «да» или «нет».

Если сервер предложил неправильный вариант, он запрашивает у пользователя имя животного и отличающий вопрос. Эта новая информация запоминается и используется в следующих играх.

1. Создайте таблицу для представления информации.
2. Придумайте интерфейс и реализуйте необходимые функции.
3. Проверьте реализацию.

## Решение

Пример диалога (между людьми):

— Это млекопитающее?	— Да.
— Это слон?	— Нет.
— Сдаюсь. Кто это?	— Кит.
— Как отличить кита от слона?	— Он живет в воде.

1. Информацию удобно представить в виде двоичного дерева. Внутренние узлы хранят вопросы, листовые узлы — названия животных. Один из дочерних узлов соответствует ответу «да», другой — ответу «нет».

2. Между вызовами функций надо передавать информацию о том, на каком узле дерева мы остановились («контекст» диалога). Функции могут быть, например, такими:

- начать игру (нет входного контекста)

```
FUNCTION start_game(OUT context integer, OUT question text)
```

- продолжение игры (получаем ответ, выдаем следующий вопрос)

```
FUNCTION continue_game(  
    INOUT context integer, IN answer boolean,  
    OUT you_win boolean, OUT question text)
```

- завершение игры (внесение информации о новом животном)

```
FUNCTION end_game(  
    IN context integer, IN name text, IN question text)
```

RETURNS void

## 1. Таблица

```
=> CREATE TABLE animals(  
    id integer PRIMARY KEY GENERATED ALWAYS AS IDENTITY,  
    yes_id integer REFERENCES animals(id),  
    no_id integer REFERENCES animals(id),  
    name text  
);
```

CREATE TABLE

```
=> INSERT INTO animals(name) VALUES  
    ('млекопитающее'), ('слон'), ('черепаха');
```

INSERT 0 3

```
=> UPDATE animals SET yes_id = 2, no_id = 3 WHERE id = 1;
```

UPDATE 1

```
=> SELECT * FROM animals ORDER BY id;
```

```
id | yes_id | no_id | name  
----+-----+-----+-----  
1 | 2 | 3 |  
(3 rows)
```

Первая строка считается корнем дерева.

## 2. Функции

```
=> CREATE FUNCTION start_game(  
    OUT context integer,  
    OUT question text  
)  
AS $$  
DECLARE
```



```
    root_id CONSTANT integer := 1;
BEGIN
    SELECT id, name||'?'
    INTO context, question
    FROM animals
    WHERE id = root_id;
END;
$$ LANGUAGE plpgsql;
```

CREATE FUNCTION

```
=> CREATE FUNCTION continue_game(
    INOUT context integer,
    IN answer boolean,
    OUT you_win boolean,
    OUT question text
)
```

```
AS $$
```

```
DECLARE
```

```
    new_context integer;
```

```
BEGIN
```

```
    SELECT CASE WHEN answer THEN yes_id ELSE no_id END
```

```
    INTO new_context
```

```
    FROM animals
```

```
    WHERE id = context;
```

```
    IF new_context IS NULL THEN
```

```
        you_win := NOT answer;
```

```
        question := CASE
```

```
            WHEN you_win THEN 'Сдаюсь'
```

```
            ELSE 'Вы проиграли'
```

```
        END;
```

```
    ELSE
```

```
        SELECT id, null, name||'?'
```

```
        INTO context, you_win, question
```

```
        FROM animals
```

```
        WHERE id = new_context;
```

```
END IF;
```

```
END;
```

```
$$ LANGUAGE plpgsql;
```

CREATE FUNCTION

```

=> CREATE FUNCTION end_game(
    IN context integer,
    IN name text,
    IN question text
) RETURNS void
AS $$
DECLARE
    new_animal_id integer;
    new_question_id integer;
BEGIN
    INSERT INTO animals(name) VALUES (name)
        RETURNING id INTO new_animal_id;
    INSERT INTO animals(name) VALUES (question)
        RETURNING id INTO new_question_id;
    UPDATE animals SET yes_id = new_question_id
    WHERE yes_id = context;
    UPDATE animals SET no_id = new_question_id
    WHERE no_id = context;
    UPDATE animals SET yes_id = new_animal_id, no_id = context
    WHERE id = new_question_id;
END;
$$ LANGUAGE plpgsql;

```

CREATE FUNCTION

### 3. Пример сеанса игры

Загадываем слово «КИТ».

```

=> SELECT * FROM start_game();

```

context	question
1	млекопитающее?

(1 row)

```

=> SELECT * FROM continue_game(1,true);

```

context	you_win	question
2		слон?

(1 row)

```

=> SELECT * FROM continue_game(2,false);

```

context	you_win	question
2	t	Сдаюсь

(1 row)

```
=> SELECT * FROM end_game(2,'кит','живет в воде');
```

end\_game

-----

(1 row)

Теперь в таблице:

```
=> SELECT * FROM animals ORDER BY id;
```

id | yes\_id | no\_id | name

-----+-----+-----

1. 1 | 5 |

1. 2 | |

1. 3 | |

1. 4 | |

1. 5 | 4 |

(5 rows)

3 | млекопитающее

| слон

| черепаха

| кит

2 | живет в воде

Снова загадали «кит».

```
=> SELECT * FROM start_game();
```

context | question

-----+-----

1 | млекопитающее?

(1 row)

```
=> SELECT * FROM continue_game(1,true);
```

context | you\_win | question

-----+-----+-----

5 | | живет в воде?

(1 row)

```
=> SELECT * FROM continue_game(5,true);
```

```

context | you_win | question
-----+-----+-----
      4 |      | кит?
(1 row)

```

=> **SELECT \* FROM continue\_game(4,true);**

```

context | you_win | question
-----+-----+-----
      4 | f      | Вы проиграли
(1 row)

```

#### Практика №18

1. Измените функцию book\_name: если у книги больше двух авторов, то в названии указываются только первые два и в конце добавляется «и др.».

Проверьте работу функции в SQL и в приложении.

2. Попробуйте написать функцию book\_name на SQL. Какой вариант нравится больше — PL/pgSQL или SQL?

#### Решение

1. Например:

Хрестоматия. Пушкин А. С., Толстой Л. Н., Тургенев И. С. →

→ Хрестоматия. Пушкин А. С., Толстой Л. Н. и др.

#### 1. Функция book\_name (сокращение авторов)

Напишем более универсальную функцию с дополнительным параметром — максимальное число авторов в названии.

Поскольку функция меняет сигнатуру (число и/или типы входных параметров), ее необходимо сначала удалить, а потом создать заново. В данном случае у функции есть зависимый объект — представление catalog\_v, в котором она используется. Представление тоже придется пересоздать (в реальной работе все эти действия надо выполнять в одной транзакции, чтобы изменения вступили в силу атомарно).

=> **DROP FUNCTION book\_name(integer,text) CASCADE;**

NOTICE: drop cascades to view catalog\_v

DROP FUNCTION

```
=> CREATE OR REPLACE FUNCTION book_name(  
    book_id integer,  
    title text,  
    maxauthors integer DEFAULT 2  
)  
RETURNS text  
AS $$  
DECLARE  
  
    r record;  
  
    res text;  
BEGIN  
    res := shorten(title) || ' ';  
    FOR r IN (  
        SELECT a.last_name, a.first_name, a.middle_name, ash.seq_num  
        FROM authors a  
            JOIN authorship ash ON a.author_id = ash.author_id  
        WHERE ash.book_id = book_name.book_id  
        ORDER BY ash.seq_num  
    )  
  
        LOOP  
            EXIT WHEN r.seq_num > maxauthors;  
            res := res || author_name(r.last_name, r.first_name, r.middle_name) || ' ';  
        END LOOP;  
        res := rtrim(res, ' ');  
        IF r.seq_num > maxauthors THEN  
            res := res || ' и др.';  
        END IF;  
        RETURN res;  
    END;  
$$ STABLE LANGUAGE plpgsql;
```

CREATE FUNCTION

```
=> CREATE OR REPLACE VIEW catalog_v AS  
SELECT b.book_id,  
    b.title,  
    b.onhand_qty,  
    book_name(b.book_id, b.title) AS display_name,  
    b.authors  
FROM books b
```

**ORDER BY** display\_name;

CREATE VIEW

=> **SELECT** book\_id, display\_name **FROM** catalog\_v;

book_id	display_name
4	Война и мир. Толстой Л. Н.
2	Муму. Тургенев И. С.
5	Путешествия в некоторые удаленные страны.... Свифт Д.
1	Сказка о царе Салтане. Пушкин А. С.
3	Трудно быть богом. Стругацкий А. Н., Стругацкий Б. Н.
6	Хрестоматия. Пушкин А. С., Толстой Л. Н. и др.

(6 rows)

## 2. Вариант на чистом SQL

```
=> CREATE OR REPLACE FUNCTION book_name(  
    book_id integer,  
    title text,  
    maxauthors integer DEFAULT 2  
)  
RETURNS text  
AS $$  
SELECT shorten(book_name.title) ||  
    ' ' ||  
    string_agg(  
        author_name(a.last_name, a.first_name, a.middle_name), ', '  
        ORDER BY ash.seq_num  
    ) FILTER (WHERE ash.seq_num <= maxauthors) ||  
    CASE  
        WHEN max(ash.seq_num) > maxauthors THEN ' и др.'  
        ELSE ''  
    END  
END  
  
FROM authors a  
    JOIN authorship ash ON a.author_id = ash.author_id  
WHERE ash.book_id = book_name.book_id;  
$$ STABLE LANGUAGE sql;
```

CREATE FUNCTION

=> **SELECT** book\_id, display\_name **FROM** catalog\_v;

book_id	display_name
-----+-----	
4	Война и мир. Толстой Л. Н.
2	Муму. Тургенев И. С.
5	Путешествия в некоторые удаленные страны.... Свифт Д.
1	Сказка о царе Салтане. Пушкин А. С.
3	Трудно быть богом. Стругацкий А. Н., Стругацкий Б. Н.
6	Хрестоматия. Пушкин А. С., Толстой Л. Н. и др.

(6 rows)

## Практика №19

### 1. Требуется распределить расходы на электроэнергию

по отделам компании пропорционально количеству сотрудников (перечень отделов находится в таблице). Напишите функцию, которая примет общую сумму расходов и запишет распределенные расходы в строки таблицы. Числа округляются до копеек; сумма расходов всех отделов должна в точности совпадать с общей суммой.

2. Напишите табличную функцию, имитирующую сортировку слиянием. Функция принимает две курсорные переменные; оба курсора уже открыты и возвращают упорядоченные по неубыванию целые числа. Требуется выдать общую упорядоченную последовательность чисел из обоих источников.

## Решение

1. В качестве таблицы можно взять:

```
CREATE TABLE depts(  
    id integer PRIMARY KEY GENERATED ALWAYS AS IDENTITY,  
    employees integer,  
    expenses numeric(10,2)  
);  
  
INSERT INTO depts(employees) VALUES (10),(10),(10);
```

Функция:

```
FUNCTION distribute_expenses(amount numeric) RETURNS void;
```

Ожидаемый результат после вызова функции с параметром 100.00:

expenses

-----

33.33

33.34

33.33



2. Функция:

FUNCTION merge(c1 refcursor, c2 refcursor) RETURNS SETOF integer;

Например, если первый курсор возвращает последовательность 1, 3, 5, а второй — 2, 3, 4, то ожидается результат:

merge

-----

1

2

3

3

4

5

## 1. Распределение расходов

=> **CREATE DATABASE** plpgsql\_cursors;

CREATE DATABASE

=> **\c** plpgsql\_cursors

You are now connected to database "plpgsql\_cursors" as user "student". Таблица:

=> **CREATE TABLE** depts(  
    id **integer** **PRIMARY KEY GENERATED ALWAYS AS IDENTITY**,  
    employees **integer**,  
    expenses **numeric**(10,2)  
);

CREATE TABLE

=> **INSERT INTO** depts(employees) **VALUES** (10),(10),(10);

INSERT 0 3

Функция:

=> **CREATE FUNCTION** distribute\_expenses(amount **numeric**) **RETURNS void**

```

AS $$
DECLARE
    depts_cur CURSOR FOR
        SELECT employees FROM depts FOR UPDATE;
    total_employees numeric;
    expense numeric;
    rounding_err numeric := 0.0;
    cent numeric;
BEGIN
    SELECT sum(employees) FROM depts INTO total_employees;
    FOR dept IN depts_cur LOOP
        expense := amount * (dept.employees / total_employees);
        rounding_err := rounding_err + (expense - round(expense,2));
        cent := round(rounding_err,2);
        expense := expense + cent;
        rounding_err := rounding_err - cent;
        UPDATE depts SET expenses = round(expense,2)
        WHERE CURRENT OF depts_cur;
    END LOOP;
END;
$$ VOLATILE LANGUAGE plpgsql;

```

CREATE FUNCTION

Проверка:

=> **SELECT** distribute\_expenses(100.00);

distribute\_expenses

-----

(1 row)

=> **SELECT** \* **FROM** depts;

id | employees | expenses

----+-----+-----

1 | 2 | 3 |

(3 rows)

10 | 10 | 10 |

33.33

33.34

33.33

Разумеется, возможны и другие алгоритмы, например, перенос всех ошибок округления на одну строку и т. п. В курсе DEV2 рассматривается другое решение этой задачи с помощью пользовательских агрегатных функций.

## 2. Слияние отсортированных наборов

Эта реализация предполагает, что числа не могут иметь неопределенные значения NULL.

```
=> CREATE FUNCTION merge(c1 refcursor, c2 refcursor)
RETURNS SETOF integer
AS $$
DECLARE
```

```
a integer;

    b integer;
BEGIN
    FETCH c1 INTO a;
    FETCH c2 INTO b;
    LOOP
        EXIT WHEN a IS NULL AND b IS NULL;
        IF a < b OR b IS NULL THEN
            RETURN NEXT a;
            FETCH c1 INTO a;
        ELSE
            RETURN NEXT b;
            FETCH c2 INTO b;
        END IF;
    END LOOP;
END;
```

```
$$ VOLATILE LANGUAGE plpgsql;
```

```
CREATE FUNCTION
```

Проверяем.

```
=> BEGIN;
```

```
BEGIN
```

```
=> DECLARE c1 CURSOR FOR
SELECT * FROM (VALUES (1),(3),(5)) t;
```

DECLARE CURSOR

=> **DECLARE** c2 **CURSOR FOR**  
**SELECT \* FROM (VALUES (2),(3),(4)) t;**

DECLARE CURSOR

=> **SELECT \* FROM** merge('c1','c2');

merge

-----

1 2 3 3 4 5

(6 rows)

=> **COMMIT;**

COMMIT

