Задание 1

Выполним запрос, используя функцию to_tsvector.

```
SELECT to_tsvector('Съешь ещё этих мягких французских булок, да выпей чаю');
```

Результатом выполнения запроса будет являться следующая строка лексем

	to_tsvector tsvector
1	'булок':6 'вып':8 'ещё':2 'мягк':4 'съеш':1 'французск':5 'ча':9 'эт':3

Изучив документацию, выясняется, что «да» не отображается в векторе, так как является «стоп-словом» - встречается слишком, чтобы поиск по ним был осмысленным.

Задание 2

Выполним седующие запросы.

```
SELECT to_tsvector('The quick brown fox jumped over the lazy dog')
    @@ to_tsquery('fox');
```

Результат:

	?column? boolean
1	true

Следующим запросом является:

```
SELECT to_tsvector('The quick brown fox jumped over the lazy dog')
    @@ to_tsquery('foxes');
```

Результат:

	?column? boolean
1	true

Последний запрос:

```
SELECT to_tsvector('The quick brown fox jumped over the lazy dog')
    @@ to_tsquery('foxhound');
```

Результат:

	?column? boolean
1	false

Отвечая на вопросы:

1) Символ @@ является оператором соответствия для полнотекстового поиска.
2) Во втором запросе, в отличие от третьего используется множественное число слова fox, поэтому, при выполнении функции to_tsvector, первый словарь, который *распознал* фрагмент, выдаёт одну или несколько представляющих его *лексем*. Поэтому foxes превращается в fox, так как один из словарей понимает, что слово foxes — это слово fox во множественном числе. Слово foxhound - не является множественным числом, либо однокоренным со словом fox. Поэтому результатом третьего запроса было false.
3) Выполним запрос

```
SELECT to_tsvector('Russian', 'Съешь ещё этих мягких французских булок, да выпей чаю.')
    @@ to_tsquery('Russian','булка');
```

Результатом выполнения является следующее:

	?column? boolean
1	false

Для ответа на вопрос выполним следующий запрос:

```
SELECT ts_lexize('russian_stem', 'булок');
```

Результатом которого будет:

	ts_lexize text[]
1	{булок}

Функция ts_lexize возвращает массив лексем, если входной *фрагмент* известен словарю, либо пустой массив, если этот фрагмент считается в словаре стоп-словом.

Так же выполним следующий запрос:

```
SELECT ts_lexize('russian_stem', 'булка');
```

.

Результат:

	ts_lexize text[]
1	{булк}

Соответственно, при выполнении исходного запроса to_tsvector распознает слово булок как лексему ‘булок’, а слово булка имеет лексему ‘булк’, именно из-за разницы лексем слово булка и не было найдено в запросе.

4) Выполним следующие запросы:

```
SELECT to_tsvector('Russian', 'Съешь ещё этих мягких французских пирожков, да выпей чаю.')
    @@ to_tsquery('Russian','пирожки');
```

Результатом является следующее:

	?column? boolean
1	true

Выполним следующий запрос:

```
SELECT to_tsvector('Russian', 'Съешь ещё этих мягких французских пирожков, да выпей чаю.')
    @@ to_tsquery('Russian','пирожок');
```

Результатом будет следующее:

	?column? boolean
1	false

При выполнении аналогичного запроса представленного в пункте 3, выясняется, что лексемой слова пирожки является “пирожк”, лексемой пирожок является “пирожок”, лексемой же слова пирожков является так же “пирожк”. Именно поэтому результатом первого запроса является true, а второго false.

3 Задание

1) Выполним следующие запросы:

```
SELECT to_tsvector('The quick brown fox jumped over the lazy dog')
    @@ to_tsquery('fox & dog');
```

Результат:

	?column? boolean
1	true

Следующий запрос:

```
SELECT to_tsvector('The quick brown fox jumped over the lazy dog')
    @@ to_tsquery('fox | rat');
```

Результат:

	?column? boolean
1	true

Следующий запрос:

```
SELECT to_tsvector('The quick brown fox jumped over the lazy dog')
    @@ to_tsquery('!clown');
```

Результат:

	?column? boolean
1	true

Последний запрос:

```
SELECT to_tsvector('The quick brown fox jumped over the lazy dog')
    @@ to_tsquery('fox & (dog | rat) & !mice');
```

Результат:

	?column? boolean
1	true

Выполним аналогичные запросы для предложения на русском языке

```
SELECT to_tsvector('Russian', 'В центре большого города стоит деревянный дом')
    @@ to_tsquery('Russian','город & дом');
```

Результат:

	?column? boolean
1	true

Следующий запрос:

```
SELECT to_tsvector('Russian', 'В центре большого города стоит деревянный дом')
    @@ to_tsquery('Russian','дом | деревня');
```

Результат:

	?column? boolean
1	true

Следующий результат:

```
SELECT to_tsvector('Russian', 'В центре большого города стоит деревянный дом')
    @@ to_tsquery('Russian','!квартира');
```

Результат:

	?column? boolean
1	true

Последний запрос:

```
SELECT to_tsvector('Russian', 'В центре большого города стоит деревянный дом')
    @@ to_tsquery('Russian','город & (дом | квартира) & !деревня');
```

Результат:

	?column? boolean
1	true

2) Английский язык является языком по умолчанию, поэтому его не нужно указывать в первом аргументе. Если не указывать анализатор, то возьмётся язык по умолчанию.

4 Задание

Выполним следующий запрос:

```
SELECT to_tsvector('Russian', 'Съешь ещё этих мягких французских булок, да выпей чаю.')
    @@ to_tsquery('Russian','мягких<2>булок');
```

Результатом выполнения запросов является:

	?column? boolean
1	true

1) Оператор <-> является оператором поиска фраз <-> (ПРЕДШЕСТВУЕТ). В нашем случае представлена вариация оператора ПРЕДШЕСТВУЕТ вида <*N*>, где *N* — целочисленная константа, задающая расстояние между двумя искомыми лексемами. Соответственно <2> обозначает расстояние между лексемами ‘мягких’ и ‘булок’.
2) Выполним запрос для поиска фразы:

```
SELECT to_tsvector('Russian', 'Съешь ещё этих мягких французских булок, да выпей чаю.')
    @@ to_tsquery('Russian','Съешь<->ещё');
```

Результат:

	?column? boolean
1	true

3) Функция phraseto_tsquery преобразует неформатированный текст запроса в значение tsquery. Она вставляет между оставшимися словами оператор <->. Стоп-слова не отбрасываются, а подсчитываются, и вместо операторов <-> используются операторы <*N*> с подсчитанным числом.

Задание 5

1) Функция ts_debug - выводит информацию обо всех фрагментах данного документа, которые были выданы анализатором и обработаны настроенными словарями.

```
1  SELECT ts_debug('english', 'Big fat cat eat fat rats');
```

Результат | План выполнения | Сообщения | Notifications

	ts_debug record
1	(asciiword,"Word, all ASCII",Big,{english_stem},english_stem,{big})
2	(blank,"Space symbols"," ",{},,)
3	(asciiword,"Word, all ASCII",fat,{english_stem},english_stem,{fat})
4	(blank,"Space symbols"," ",{},,)
5	(asciiword,"Word, all ASCII",cat,{english_stem},english_stem,{cat})
6	(blank,"Space symbols"," ",{},,)
7	(asciiword,"Word, all ASCII",eat,{english_stem},english_stem,{eat})
8	(blank,"Space symbols"," ",{},,)
9	(asciiword,"Word, all ASCII",fat,{english_stem},english_stem,{fat})
10	(blank,"Space symbols"," ",{},,)
11	(asciiword,"Word, all ASCII",rats,{english_stem},english_stem,{rat})

2) Функция ts_headline принимает документ вместе с запросом и возвращает выдержку из документа, в которой выделяются слова из запроса.

```
1  SELECT ts_headline('Russian','Большие коты едят больших крыс.', to_tsquery('больших & крыс'));
```

Результат | План выполнения | Сообщения | Notifications

	ts_headline text
1	Большие коты едят больших крыс.