**Практическое занятие № 6**

**Язык SQL. Оператор запроса SELECT**

**Цель занятия**. Научиться составлять оператор запроса к таблицам базы данных, использовать предикаты для отбора нужных данных вБДPostgreSQL.

Cтандартнымязыком доступа к базам данных является язык SQL(StructuredQueryLanguage). Все реляционные СУБД, используют тот или иной диалект SQL. Теоретической основой языка SQL являются реляционная алгебра и реляционное исчисление.

Базовые операторы SQL по функциональному признаку можно разделить на группы:

* Язык определения данных DDL(DataDefinitionLanguage); DDL включает команды, которые создают объекты данных (таблицы, индексы, представления, и т.д.). Совокупность всех объектов БД – каталог БД или ее структура;
* Язык манипулирования данными DML (DataManipulationLanguage); DML включает команды заполнения БД и формирования запросов к БД;
* Язык управления данными DCL (DataControlLanguage) – организация доступа к БД (средства определения прав пользователей на выполнение определенных действий над данными: операторы GRANT и REVOKE);
* Языкуправлениятранзакциями TCL (Transaction Control Language) COMMIT|ROLLBACK

Основными операторами языка SQL для манипулирования данными (DML) являются:

* оператор запроса (SELECT)
* операторы обновления (INSERT, UPDATE, DELETE)

**Оператор запроса SELECT**

Оператор SELECT является фактически самым важным для пользователя и самым сложным оператором SQL. Оператор SELECT*предназначен для выборки данных из таблиц*, т.е. он реализует одно их *основных назначений* базы данных - *предоставлять информацию* пользователю.

Оператор SELECT всегда выполняется над некоторыми таблицами, входящими в базу данных (постоянно хранимыми в БД, временными таблицами и представлениями). *Результатом выполнения* оператора SELECT всегда является *таблица*.

Запрос может включать подзапросы, т.е. допускается вложение операторов запроса.

Общий вид оператораSELECT (в квадратных скобках указаны необязательные элементы):

**SELECT [ALL|DISTINCT] <списоквыборки>**

**<табличное выражение>**

**[<ORDERBY раздел>]**

где

< **табличное выражение**>**::=<FROM раздел>**

**[<WHERE раздел>]**

**[<GROUP BY раздел>]**

**[<HAVING раздел>]**

В определении запроса задается:

1. **список выборки** (список арифметических выражений, включающих столбцы табличного выражения и константы). Элементы списка разделяются запятыми. В частном случае можно задать символ \* - в результат включаются все элементы табличного выражения.

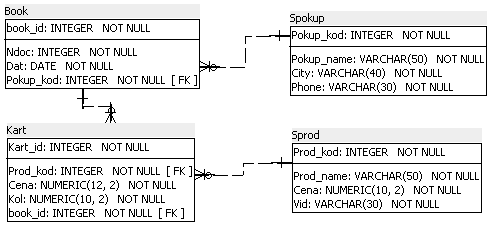
*Пример.* Выбрать все данные из справочника продукции

SELECT \* FROM Sprod

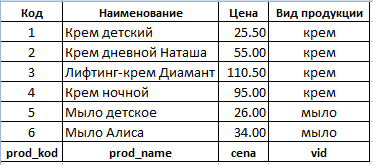
1. **Ключевые слова ALL или DISTINCT**. При наличии ключевого слова DISTINCT из таблицы-результата, удаляются строки-дубликаты; при указании ALL (или отсутствии слова) удаление строк-дубликатов не производится.
2. **Табличное выражение** формируется на основе последовательного применения разделов FROM, WHERE, GROUPBY и HAVING из таблиц, заданных в разделе FROM. FROM - единственный обязательный раздел. В результирующей таблице порядок следования строк не определен и среди строк могут находиться дубликаты.
3. **Раздел ORDERBY** используется для упорядочения строк результата.

Рассмотрим структуру и назначение разделов табличного выражения.

При рассмотрении оператора SQLв примерах будем использовать БД Продажи (рис.1), с заполненными таблицами (рис.2-5).



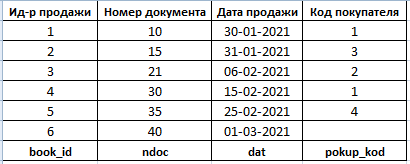
**Рис. 1 База данных Продажи**



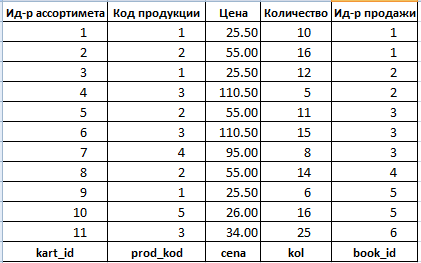
**Рис. 2 Данные для таблицы sprod (Продукция)**



**Рис. 3 Данные для таблицы spokup (Покупатели)**

****

**Рис. 4 Данные для таблицы book (Книга продаж)**



**Рис. 5 Данные для таблицы kart (Ассортимент продаж)**

**Раздел FROM**

Раздел FROM задает таблицы-источники и имеет следующий синтаксис:

**FROM <список таблиц>**

где

**<таблица>::= <имя таблицы> [<алиас>]**

Рядом с именем таблицы можно указывать еще одно имя – **алиас(псевдоним)** таблицы. Алиас (псевдоним) вводится для использования в других разделах оператора SELECT в качестве префикса перед именем столбца и отделяются от имени столбца точкой. Алиасы используются:

а) Обязательно, если в разделе FROM используются копии одной и той же таблицы,

б) Необязательно - для замены длинных имен таблиц.

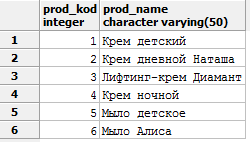
*Результатом выполнения раздела FROM является РАСШИРЕННОЕ ДЕКАРТОВО ПРОИЗВЕДЕНИЕ таблиц-источников.*

Если табличное выражение содержит только раздел FROM, то результат - это вся таблица или декартово произведение таблиц, т.е каждая строка одной таблицы сцепляется (приписывается справа) с каждой строкой второй таблицы.

*Пример1*. SELECT prod\_kod,prod\_name FROM sprod

*Из всех записей таблицы SPROD выбираются код продукции и наименование продукции.*

Результат запроса*:*



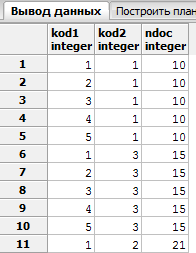
*Пример 2*.SELECTs.pokup\_kodkod1,d.pokup\_kodkod2, ndoc

FROMspokups, bookb

*Вычисляется декартово произведение (каждая строка таблицы SPROD сцепляется с каждой строкой таблицы BOOK)*

*Выбираются: код покупателя (kod1) из таблицы Покупатель (spokup), код покупателя (kod2) и номер документа из таблицы Книга продаж из всех полученных строк прямого произведения таблиц.*

Первые 11 строк результата:

**

***Упражнение 1****.* Вывести наименование и адрес всех покупателей.

**Раздел WHERE**

Из результатов запроса последнего примеравидно, что выбрать данные из нескольких связанных таблиц, используя только раздел FROM невозможно, т.к. получается много бессмысленных соединенных строк. Здесь на помощь приходит раздел WHERE.

Раздел WHERE задает условия отбора данных. Синтаксис раздела:

**WHERE <условие отбора>**

В качестве условия в разделе WHERE можно использовать сложные логические выражения, включающие:

* поля таблиц, константы, выражения;
* операторы условия выборки (предикаты);
* скобки;
* логические операторы AND (и), OR (или), NOT (не);
* подзапросы (select …).

Условие отбора применяется ко всем строкам таблицы-результата, полученным после применения раздела FROM. В конечный результат оператора запроса попадают те строки, для которых условие дает «истину» (true).

Поскольку SQL допускает наличие в базе данных неопределенных значений, то вычисление условия производится не в булевой (двухзначной), а в трехзначной логике со значениями True, False и Unknown (null - неизвестно). Булевские операции AND, OR и NOT работают в трехзначной логике по следующим правилам (true - обозначим 1, false - 0, null - un) (таблица 1).

Таблица 1 - Таблицы истинности в трехзначной логике

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **A** | **B** | **А and B** | **A or B** | **not A** |
| 0 | 0 | 0 | 0 | 1 |
| 0 | 1 | 0 | 1 |  |
| 0 | unknown | 0 | unknown |  |
| 1 | 0 | 0 | 1 | 0 |
| 1 | 1 | 1 | 1 |  |
| 1 | unknown | unknown | 1 |  |
| unknown | 0 | 0 | unknown | unknown |
| unknown | 1 | unknown | 1 |  |
| unknown | unknown | unknown | unknown |  |

В разделе WHERE используются следующие ***операторы условия отбора (предикаты)***:

* простые сравнения (>, <, = и т.д.),
* between,
* in,
* like,
* isnull,
* одноместные операторы exists, all, any

*Рассмотрим основные операторы условий отбора (предикаты)*

**1) Предикат сравнения** реализует простое сравнение

**<выражение1>{= | < | > | <= | >= | <>}**

**{<выражение2> | <подзапрос>}**

*Выражение1* и *выражение2* могут включать в общем случае имена столбцов таблиц из раздела FROM и константы.

При сравнении с подзапросом мощность результата подзапроса (количество возвращаемых строк) должна быть не более 1.

Значение оператора сравнения равно ***unknown*** если:

* выражение1и/или выражение2 имеют неопределенное значение;
* подзапрос возвращает пустой результат.

Значение арифметического выражения не определено ***(unknown)***, если хотя бы один операнд имеет неопределенное значение.

Рассмотрим примеры. В предыдущих примерах вначале задавался оператор запроса, а затем давались словесные формулировки. Теперь сделаем наоборот: вначале дадим текстовую формулировку, а затем оператор запроса.

*Пример 3*. Выбрать наименования покупателей, получивших продукцию в феврале 2021 года.

Пояснение: Для выполнения запроса нам потребуются таблицы:

1)Покупатель (spokup), т.к. нам нужно вывести наименование покупателей;

2)Книга продаж (book), т.к. в результат должны попасть покупатели, которые получили продукцию в заданном периоде времени (феврале 2021 г.).

Операторзапроса:

SELECT DISTINCTsp.pokup\_name

FROM book b, spokupsp

WHERE b.pokup\_kod=sp.pokup\_kod AND

b.dat>='01-02-2021'AND b.dat<'01-03-2021'

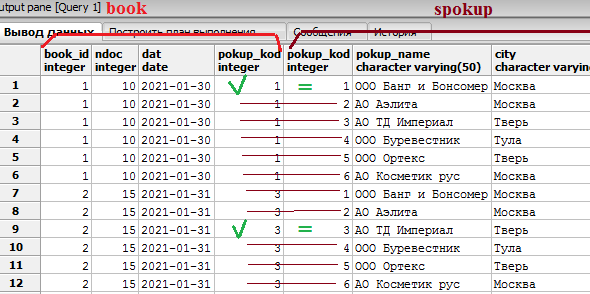
Алгоритм выполнения.

1) Вычисляется декартово произведение таблиц: каждая строка таблицы*Spokup* сцепляется с каждой строкой таблицы *Book*. Ниже показаны первые 12 строк.

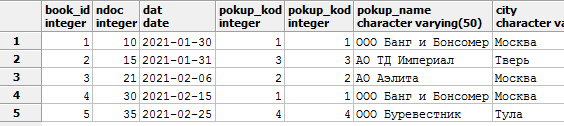


2) К каждой строке произведения таблиц применяется условие отбора раздела WHERE и в результат запроса отбираются строки, для которых условие дает «истину».

Первый предикатсравнения (sp.pokup\_kod=b.pokup\_kod)не задан конкретным условием самого запроса, а выполняет ***естественное соединение*** строкпо колонке первичного ключа *Код покупателя* в таблице Покупатель (spokup) и внешнего ключа *Код покупателя* в таблице Книга продаж (book). Строки с несовпадающими кодами будут отброшены.

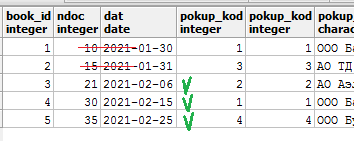


Результат выполнения первого предиката сравнения

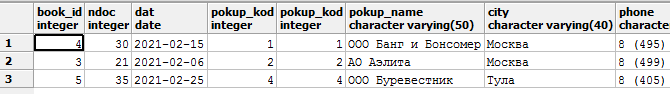


На следующих занятиях мы рассмотрим подробнее операцию соединения.

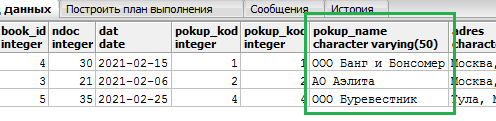
Два следующих предиката сравнения(b.dat>='01-02-2021' и b.dat<'01-03-2021') определяются условиями запроса и проверяют попадание продажи в заданный период времени.



Результат выполнения предикатов сравнения даты с заданным периодом

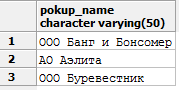


3) Из полученных строк отбирается колонка*Наименование покупателя*.



4) Дублированные строки отбрасываются. В данном примере повторений нет.

Результат запроса:

**

*Пример 4*(сравнение срезультатом подзапроса). Выбрать наименование продукции, которая отпускалась покупателю «ООО Банг иБонсомер».

Пояснение: Для выполнения запроса нам потребуются таблицы:

1) Продукция (sprod), т.к. нам нужно вывести наименование продукции;

2) Ассортимент продаж (kart), т.к. в результат должна попасть продукция, которая продавалась;

3) Книга продаж (book), т.к. необходимо иметь информацию о покупателе;

4) Покупатель (spokup), т.к. нас интересуют только продажи заданному покупателю«ООО Банг иБонсомер»;

Первые 3 таблицы будем соединять в запросе, а четвертую будем включать в подзапрос.

Подзапрос можно использовать, если он возвращает не более 1 значения. Это гарантируется тем, что в таблице Spokup имеется ограничение уникальности по колонке *Наименование покупателя* (pokup\_name).

Операторзапроса:

SELECT DISTINCTprod\_name

FROM sprod s, kart k, book as b

WHEREs.prod\_kod =k.prod\_kod-- *СоединениетаблицSprodи Kart*

AND k.book\_id=b.book\_id-- *Соединениетаблиц Kart и Book*

AND pokup\_kod = (SELECT pokup\_kod FROM spokup

WHERE pokup\_name='ОООБангиБонсомер')

Алгоритм выполнения.

1) Вычисляется декартово произведение таблиц: каждая строка таблицы *Sprod* сцепляется с каждой строкой таблицы *Kart*, а затем с каждой строкой таблицы *Book*.

2) Выполняется подзапрос: определяется и запоминается код покупателя 'ООО Банг иБонсомер'

3) К каждой строке произведения таблиц применяется условие отбора раздела WHERE и в результат запроса отбираются строки, для которых условие дает «истину».

Первый предикат сравнения (s.prod\_kod =k.prod\_kod) выполняет ***естественное соединение*** строк таблиц *Sprod* и *Kart*по общей колонке *Код продукции*.

Второй предикат сравнения (k.book\_id=b.book\_id) выполняет ***естественное соединение*** строк *Kart* и *Book* по общей колонке *Идентификатор продажи*.

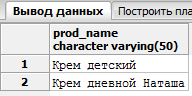
Строки с несовпадающими кодами будут отброшены.

Последний предикат выполняет сравнение кода покупателя продажи с полученным ранее кодомпокупателя 'ООО Банг иБонсомер', полученного подзапросом.

4) Из полученных строк отбирается колонка *Наименование продукции*.

5) Дублированные строки отбрасываются.

Результат запроса:

****

***Упражнение 2****.* Выбрать всю продукцию из накладной с заданным номером и датой. Вывести код и наименование продукции, цену, количество и стоимость каждой позиции накладной.

Пояснение: 1) *Стоимость* определяется выражением, равным произведению колонок *Цена* и *Количество*.

2) Для округления значения до заданного количества разрядов используют функцию CAST («выражение» as «тип»), где типом можно выбрать NUMERIC (10,2)

**2) Предикат between** проверяет принадлежность колонки (выражения) заданному диапазону.

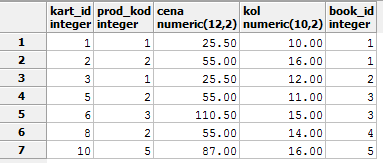
**<колонка> [NOT] BETWEEN <значение1> AND <значение2>**

Предикат возвращает истину, если значения «колонки» находятся в диапазоне от «значения1» до «значения2». Колонка и значения должны быть одного типа

***Пример 5***. Вывести продажи, объем которых (колонка *Количество*)был в диапазоне от 10 до 20.

SELECT \* FROM kart WHERE kol BETWEEN 10 AND 20

Результат запроса:



***Упражнение 3***. Выбрать продукцию, представленную в накладных за январь 2021 г. Вывести код и наименование продукции, цену, количество и стоимость каждой позиции накладной.

**3) Предикат in**проверяет принадлежность списку.

**<выражение> [NOT] IN {<подзапрос> | (<список значений>)}**

Сравниваемое выражение должно быть равно одному из элементов списка. Если используется **NOT IN**, то проверяется не принадлежность диапазону.

Типы левого операнда и значений из списка правого операнда должны быть сравнимыми. Подзапрос должен возвращать только один столбец.

Значение оператора равно:

* true в том случае, когда значение левого операнда совпадает хотя бы с одним значением списка правого операнда;
* false **-** если список правого операнда пуст (так может быть, если правый операнд задается подзапросом), или значение левого операнда не совпадает ни с одним из элементов списка правого операнда;
* unknown - в других случаях.

*Пример 6.* Отобрать все продажис кодом продукции '1' или '2'

Вариант запроса*с предикатами сравнения*:

SELECT prod\_kod, cena, kol

FROM kart

WHERE prod\_kod =1 OR prod\_kod =2

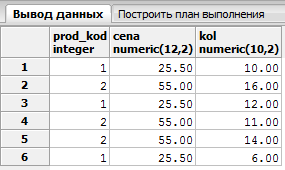
Вариант запроса *с предикатом in* будет более кратким:

SELECT prod\_kod, cena, kol

FROM kart

WHERE prod\_kod in (1,2)

Результат:



*Пример 7.*Вывести номера документов, в которых не было продукции с кодом '1'

SELECT ndoc

FROM Book

WHERE book\_id NOT IN **(**SELECT DISTINCT book\_id

FROM kart WHEREprod\_kod = 1**)**

В данном запросе список для сравнения получается подзапросом.

Результат:



***Упражнение 4.***Использовать предикат in с фиксированным списком и списком, полученным запросом.

1. Вывести наименования покупателей, которые получали продукцию с кодом 3 или 4.

2. Вывести наименования продукции, которая продавалась в январе 2021 г.

**4) Предикат like**используется для **поиска подстрок в столбцах символьного типа**. Он проверяет, совпадает ли часть текста с заданной строкой-шаблоном

**<колонка> [NOT] LIKE<шаблон>**

В шаблоне можно использовать 2 символа, обозначающие «любой текст»:

* символ подчеркивания ("\_") обозначает любой одиночный символ;
* символ процента ("%") обозначает последовательность (в том числе пустую) произвольного количества любых.

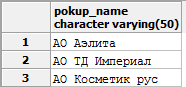
*Пример 8.*Вывести покупателей, наименования которых начинаются на «АО»

SELECT pokup\_name FROM Spokup

WHERE Pokup\_nameLIKE 'AО%'

При этом шаблоны 'AO' или 'AO\_' вернут неверный результат.

Результат запроса:



***Упражнение 5****.* Получить список продукции, в названии которых есть слово «крем».

*Пояснение:* Поиск является регистрозависимым, т.е. «Крем» и «крем» - это разные строки. Чтобы сделать поиск независимым от регистра, надо строки перевести в верхний регистр функцией UPPER(строка), а константу записать ПРОПИСНЫМИ символами.

**5) Предикат ISNULL**позволяет проверить незаполненность колонки

Null-значения нельзя сравнивать с другими значениями, даже если они Null. Поэтому, для того, чтобы отобрать строки, в которых некоторый столбец не определен, используется специальный оператор сравнения с Null:

**<колонка>IS [NOT] NULL**

Этот оператор всегда принимает значения true или false. При этом значение "x IS NULL" равно true тогда и только тогда, когда значение x не определено.

В данной ситуации нельзя использовать простое сравнение:

x=NULL илиx<>NULL

***Упражнение 6.***Вывести наименование покупателей, у которых не заполнен телефон.

**6) Операторы подзапросов Exists,All, Any**относятся кспециальным одноместные операторы, аргументом которых является подзапрос.

***Оператор exists***используется для определения наличия данных в результате подзапроса.

**EXISTS<подзапрос>**

Значение этого предиката равноtrue , когда результат вычисления подзапроса не пуст, иначе false (результата ***unknown*** – не может быть).

Пример 9. Получить перечень покупателей, которые купили продукцию с кодом 1

SELECT pokup\_name FROM spokupsp

WHERE EXISTS(SELECT 1 FROM book b, kart k

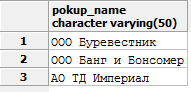
WHERE b.pokup\_kod=sp.pokup\_kod

andb.book\_id=k.book\_id

andk.prod\_kod=1)

В операторе подзапроса на месте списки выборки можно поставить любое выражение: \*, любую колонку или константу. Проверяется только наличие строк результата и не важно, что возвращает подзапрос.

Результат запроса:



***Упражнение 7*.** Вывести продукцию, которая не продавалась в январе 2021 г.

***Операторы с квантором****существования или всеобщности* имеет следующий синтаксис:

**<выражение>{= | < | > | <= | >= | <>}<квантор><подзапрос>**

**где <квантор> ::= ALL | ANY (синоним SOME)**

Пример KOL > ALL(select KOL FROM …)

Обозначим через

X - результат вычисления арифметического выражения левой части предиката

S - результат вычисления подзапроса.

Предикат всеобщности **"*X<оператор> ALL S*"** имеет значение:

* true, если S пусто или значение предиката "x <оператор> s" равно true для каждой строки, входящей в S.
* false, если значение предиката "x <оператор> s" равно false хотя бы для одной строки, входящей в S.
* unknown - в остальных случаях.

Предикат существования **"*X < оператор >ANY S*"** имеет значение

* false, если S пусто или значение предиката "x < оператор > s" равно false для каждой строки, входящей в S.
* true, если значение предиката "x < оператор > s" равно true хотя бы для одной строки, `входящей в S.
* unknown - в остальных случаях.

*Пример 10.* Вывести покупателей, которыекупили продукцию, относящуюся к виду «Пена»

SELECT DISTINCTpokup\_name FROM spokup s, book b, kart k

WHERE b.pokup\_kod=s.pokup\_kod

andb.book\_id=k.book\_id

andprod\_kod =ANY(SELECT prod\_kod FROM sprod

WHEREvid='Пена')

Результат запроса:



*Пример 11.* Вывести покупателя, для которого объем покупки продукции с кодом 1 превосходит любую из поставок этой продукции других поставщиков.

SELECT DISTINCTpokup\_name FROM spokup s, book b, kart k

WHERE b.pokup\_kod=s.pokup\_kod

andb.book\_id=k.book\_id

andprod\_kod =1

and kol> ALL(SELECT kol FROM book b1, kart k1

WHERE b1.book\_id=k1.book\_id

and k1.prod\_kod =k.prod\_kod

AND b1.pokup\_kod<>b.pokup\_kod)

Результат запроса:



***Упражнение 8***

1) Вывести покупателей, которые купили продукцию в феврале 2021 г.

2) Вывести продукцию, объем продажи которой был наименьшим среди продаж любой другой продукции

**Раздел ORDER BY**

Раздел ORDER BY задает желаемый порядок просмотра результата запроса.

Правила записи

**ORDER BY <список колонок|**

**список порядковых номеров колонок>**

**[ASC | DESC]**

Ключевые слова ASC (обеспечивает упорядочение по возрастанию), DESC (по убыванию) добавляются к каждому элементу списка. Если ключевое слово опущено, то применяется вариант ASC.

Колонки можно задавать:

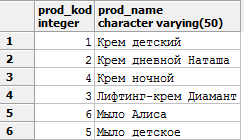
* их именами
* порядковым номером колонки

*Пример 12*. Вывести продукцию, упорядочив по наименованию.

SELECTprod\_kod,prod\_name FROM sprod

ORDER BY prod\_name

Результат запроса:



***Упражнение 9*.** Вывести продажи в обратном хронологическом порядке.

**Дополнительные полезные запросы на PostgreSQL**

1) Генерация фиксированной последовательности

Вызов функции

generate\_series(нач\_значение,конечное\_значение [,шаг])

*Пример.* Сформировать нечетные числа от 1 до 10

SELECT \* FROM generate\_series(1, 10,2)

2) Генерация случайной последовательности

SELECT random() \* макс\_значение

FROM generate\_series(нач, конец)

*Пример.* Сформировать 1000 случайных чисел в диапазоне от 0 до 100000

SELECT random() \* 100000 FROM generate\_series(1, 1000)

3) Выборка части строк

SELECT список\_выборки

FROM табличное\_выражение

[ ORDER BY ... ]

[ LIMITчисло ] [OFFSET число ]

где

LIMITчисло - задает количество возвращаемых строк

OFFSET число - задает количество пропускаемых от начала выборки строк

*Пример.* Сформировать нечетные числа от 1 до 100. Вывести 10 чисел, пропустив от начал 5 чисел

SELECT \* FROM generate\_series(1, 100,2)

LIMIT 10 OFFSET 5

**Задание**

1. Выполнить упражнения
2. Для БД индивидуального задания составить запросы, использующие различные предикаты