Лекция 2

СОРТИРОВКА ДАННЫХ

Использование ORDER BY

```
ORDER BY { order_by_expression [ ASC | DESC ] } [ ,...n ]
```

- ORDER BY сортирует записи в результирующем наборе
  - Без ORDER BY порядок записей результирующей выборки не гарантируется
  - Сортирует все NULL значения вместе
- ORDER BY может ссылаться на:
  - Имя столбца, псевдоним или позицию столбца в результирующей выборке (не рекомендуется)
  - Результат выражения
  - Столбцы, не используемые в результирующей выборке
    - Если не используется DISTINCT
- ORDER BY не поддерживается в инструкциях SELECT/INTO

Пример использования ORDER BY

```
SELECT companyname, contactname
FROM "Sales"."Customers" c
ORDER BY country ASC,  city desc;
```

	companyname	contactname
1	Customer PSNMQ	Ray, Mike
2	Customer TDKEG	Tiano, Mike
3	Customer LWGMD	Gaffney, Lawrie
4	Customer LOLJO	Meston, Tosh
5	Customer THHDP	Kane, John

```
SELECT custid, orderdate
FROM "Sales"."Orders" o
ORDER BY DATE_PART('year',orderdate) DESC;
```

	custid	orderdate
1	55	2008-01-01 00:00:00.000
2	88	2008-01-01 00:00:00.000
3	42	2008-01-01 00:00:00.000
4	47	2008-01-02 00:00:00.000
5	66	2008-01-02 00:00:00.000

```
SELECT companyname, contactname
FROM "Sales"."Customers" c
ORDER BY 1;
```

	companyname	contactname
1	Customer AHPOP	Welcker, Brian
2	Customer AHXHT	Fakhouri, Fadi
3	Customer AZJED	Carlson, Jason
4	Customer BSVAR	Rizaldy, Arif
5	Customer CCFIZ	Petrov, Ivan

Фильтрация с помощью LIMIT *

```
SELECT
FROM
[ORDER BY …]
LIMIT {integer_expression | ALL }  [ OFFSET integer_expression ]
```

- Ограничивает число строк, возвращаемых в результирующем наборе
  - *integer_expression* - число или числовое выражение, определяющее количество возвращаемых строк
  - *ALL* - равносильно отсутствию указания LIMIT
  - *OFFSET* - указывает число строк, которые необходимо пропустить, прежде чем начать выдавать строки
  - Для получения предсказуемого и согласованного результата необходимо использовать фильтрацию отсортированного набора - **ORDER BY**

Фильтрация с помощью LIMIT *

```
SELECT productname, unitprice
FROM "Production"."Products"
WHERE  unitprice <= 40::money
ORDER BY unitprice DESC;
```

	productname	unitprice
1	Product WVJFP	40.0000
2	Product BLCAX	39.0000
3	Product OSFNS	38.0000
4	Product VKCMF	38.0000
5	Product COAXA	36.0000
6	Product GEEOO	34.8000
7	Product WHBYK	34.0000
8	Product HCQDE	33.2500

```
SELECT productname, unitprice
FROM "Production"."Products"
WHERE  unitprice <= 40::money
ORDER BY unitprice desc
limit 3;
```

	productname	unitprice
1	Product WVJFP	40.0000
2	Product BLCAX	39.0000
3	Product OSFNS	38.0000

```
SELECT productname, unitprice
FROM "Production"."Products"
WHERE  unitprice <= 40::money
ORDER BY unitprice desc
limit 2 offset 2;
```

	productname	unitprice
1	Product OSFNS	$38.00
2	Product VKCMF	$38.00

Фильтрация в ORDER BY с помощью OFFSET-FETCH

PostgreSQL

```
OFFSET { integer_expression } { ROW | ROWS }
[ FETCH { FIRST | NEXT } {integer_expression } { ROW | ROWS } ONLY ] }
```

- OFFSET-FETCH – это расширение ORDER BY:
  - Позволяет отфильтровать требуемый диапазон строк
  - Предоставляет механизм для разбиения результирующего набора на страницы
- Определяет количество строк, которые необходимо:
  - Пропустить - OFFSET (может быть ноль, если не нужно пропускать строки)
  - Вернуть - FETCH (должно быть больше или равно единице)
- Если FETCH опущено – возвращаются все записи до конца набора

Фильтрация с помощью OFFSET-FETCH

Извлекает первые 50 строк

```
SELECT orderid, custid, orderdate
FROM "Sales"."Orders" o
ORDER BY orderdate DESC
OFFSET 0 ROWS FETCH FIRST 50 ROWS ONLY;
```

	orderid	custid	orderdate
1	11 077	65	2008-05-06
2	11 075	68	2008-05-06
3	11 076	9	2008-05-06
4	11 074	76	2008-05-06
5	11 073	58	2008-05-05
6	11 070	44	2008-05-05
7	11 072	20	2008-05-05

Извлекает строки 51-100

```
SELECT orderid, custid, orderdate
FROM "Sales"."Orders" o
ORDER BY orderdate DESC
OFFSET 50 ROWS FETCH FIRST 50 ROWS ONLY;
```

	orderid	custid	orderdate
1	11 027	10	2008-04-16
2	11 025	87	2008-04-15
3	11 024	19	2008-04-15
4	11 026	27	2008-04-15
5	11 020	56	2008-04-14
6	11 023	11	2008-04-14
7	11 021	63	2008-04-14

Фильтрация дубликатов

```
SELECT DISTINCT column1, column2, ...
FROM table_name;
```

- **SELECT DISTINCT** используется для возврата только уникальных записей
  - Удаляет дубликаты, базируясь на результирующем списке столбцов (не на основе таблицы-источника)
  - Работает с записями уже обработанными в выражениях **WHERE**, **HAVING** и **GROUP BY**
  - **NULL** значения - уникальны

```
SELECT DISTINCT country, city
FROM "Sales"."Customers" c;
```

	country	city
1	Argentina	Buenos Aires
2	Austria	Graz
3	Austria	Salzburg
4	Belgium	Bruxelles
5	Belgium	Charleroi
6	Brazil	Campinas
7	Brazil	Resende
8	Brazil	Rio de Janeiro
9	Brazil	Sao Paulo

```
SELECT DISTINCT region
FROM "Sales"."Customers" c ;
```

	region
1	Lara
2	DF
3	ID
4	[NULL]
5	Táchira
6	SP
7	Nueva Esparta
8	CA

Фильтрация записей

Фильтрация исходных записей - WHERE

```
SELECT ...
...
WHERE <условие фильтрации строк>
```

- Содержит логическое условие
  - Записи, для которых условие возвращает **TRUE** - попадают в результирующую выборку
  - Записи, для которых условие возвращает **FALSE** или **UNKNOWN** - отфильтровываются
- В предложении WHERE не доступны псевдонимы столбцов
- Данные фильтруются на стороне сервера
  - Оптимизация за счет использования индексов
  - Снижение нагрузки на сеть и использование памяти на клиенте

Операторы сравнения

Оператор	Описание	Пример
< > = <= >= <> !< !> !=	Операторы сравнения	Name **!=** ‘Vasia’ или Name **<>** ‘Vasia’
~ и !~	Проверка соответствия строки регулярному выражению POSIX с учетом регистра	'Thomas' **~** '.*thomas.*' → False
~* и !~*	Проверка соответствия строки регулярному выражению POSIX без учёта регистра	'Thomas' **~*** '.*thomas' → True

https://postgrespro.ru/docs/postgresql/14/functions-matching#FUNCTIONS-POSIX-REGEXP

Предикативные операторы

Оператор	Описание	Пример
BETWEEN / NOT BETWEEN	Проверка по диапазону	orderdate **BETWEEN** '2006-07-05' **AND** '2006-07-25'
IN / NOT IN	Проверка на основе списка	Price **IN** (50, 125, 253, 264)
LIKE / NOT LIKE	Сравнение строк по маске с учетом регистра	City **LIKE** 'London' → True City **LIKE** 'london' → False
ILIKE / NOT ILIKE	Сравнение строк по маске без учета регистра	City **iLIKE** 'London' → True City **iLIKE** 'london' → True
SIMILAR TO/ NOT SIMILAR TO	Сравнение строк по шаблону на основе регулярных выражений в стандарте SQL	'abc' **SIMILAR TO** '%(b|d)%' → True
IS NULL / IS NOT NULL	Проверка на наличие /отсутствие NULL значений	region **IS NOT NULL**
IS DISTINCT FROM / IS NOT DISTINCT FROM	Проверка на неравенство/ равенство заданному значению. При этом NULL воспринимается как обычное значение	region **IS DISTINCT FROM** 'WA' (все регионы, в том числе NULL, кроме WA)

https://postgrespro.ru/docs/postgresql/14/functions-comparison

Использование операторов сравнения

- Простая фильтрация

Операторы сравнения	=, >, <, >=, <=, <>, !=, !>, !<

```
SELECT productname, unitprice
FROM "Production"."Products" p
WHERE  discontinued !=1::bit;
```

	productname	unitprice
1	Product HHYDP	18.0000
2	Product RECZE	19.0000
3	Product IMEHJ	10.0000
4	Product KSBRM	22.0000
5	Product VAIIV	25.0000
6	Product HMLNI	30.0000
7	Product WVJFP	40.0000

```
SELECT productname, unitprice
FROM "Production"."Products" p
WHERE  unitprice >= 50::money;
```

	productname	unitprice
1	Product AOZBW	97.0000
2	Product CKEDC	62.5000
3	Product QHFFP	81.0000
4	Product VJXYN	123.7900
5	Product QDOMO	263.5000
6	Product APITJ	53.0000
7	Product UKXRI	55.0000

Фильтрация с использованием логических операторов

- Приоритет логических операторов – **NOT, AND, OR**
- Если два оператора в выражении имеют один и тот же уровень приоритета, они вычисляются в порядке слева направо по мере их появления в выражении
- Чтобы изменить приоритет операторов в выражении, следует использовать скобки

```
SELECT  *
FROM "Production"."Products" p
WHERE  categoryid = 1 OR categoryid = 2 AND unitprice >= 40::money;
```

	productid	productname	supplierid	categoryid	unitprice	discontinued
1	1	Product HHYDP	1	1	18.0000	0
2	2	Product RECZE	1	1	19.0000	0
3	8	Product WVJFP	3	2	40.0000	0
4	24	Product QOGNU	10	1	4.5000	1

Фильтрация NULL значений

- NULL значения используются для маркировки отсутствующих значений (missing values)
- Для корректной обработки необходимо использовать предикаты:
  - **IS NULL (ISNULL)** или **IS NOT NULL (NOTNULL)**
  - **IS DISTINCT FROM (!=)** или **IS NOT DISTINCT FROM (=)**

```
SELECT  companyname, region
FROM "Sales"."Customers" c
WHERE region IS NULL;
```

	companyname	region
1	Customer NRZBB	[NULL]
2	Customer MLTDN	[NULL]
3	Customer KBUDE	[NULL]
4	Customer HFBZG	[NULL]
5	Customer HGVLZ	[NULL]
6	Customer XHXJV	[NULL]

```
SELECT  companyname, region
FROM "Sales"."Customers" c
WHERE region IS DISTINCT FROM 'WA';
```

	companyname	region
25	Customer LCYBZ	OR
26	Customer NLTYP	MT
27	Customer YQQWW	SP
28	Customer SRQVM	SP
29	Customer NRZBB	[NULL]
30	Customer MLTDN	[NULL]
31	Customer KBUDE	[NULL]

Проверка на принадлежность диапазону

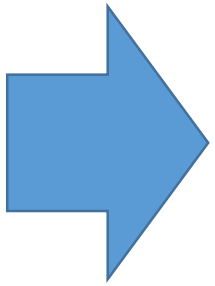
PostgreSQL

```
test_expression [ NOT ] BETWEEN begin_expression AND end_expression
```

- Границы диапазона включены
- Для задания исключающего диапазона используйте операторы "больше" (>) и "меньше" (<)
- Если любой параметр предиката **BETWEEN** или **NOT BETWEEN** имеет значение NULL, результат не определен (UNKNOWN)

```
SELECT productname, unitprice
FROM "Production"."Products" p
WHERE  unitprice BETWEEN 30::money AND 38::money;
```

```
SELECT productname, unitprice
FROM "Production"."Products" p
WHERE  unitprice >= 30::money
   AND unitprice <= 38::money;
```



	productname	unitprice
1	Product HMLNI	$30.00
2	Product YHXGE	$31.00
3	Product OSFNS	$38.00
4	Product HLGZA	$31.23
5	Product NUNAW	$32.00
6	Product BKGEA	$32.80
7	Product VKCMF	$38.00
8	Product WHBYK	$34.00

Проверка на принадлежность множеству

```
test_expression [ NOT ] IN ( subquery | expression [ ,...n ] )
```

- Определяет, совпадает ли указанное значение с одним из значений, содержащихся во вложенном запросе или списке
- Использование значений NULL с предикатами **IN** или **NOT IN** может привести к непредвиденным результатам

```
SELECT productname, unitprice
FROM "Production"."Products" p
WHERE  unitprice IN ( 22::money, 30::money, 32::money,
                      38::money);
```

```
SELECT productname, unitprice
FROM "Production"."Products" p
WHERE unitprice = 22::money OR unitprice = 30::money
   OR unitprice = 32::money OR unitprice = 38::money;
```

	productname	unitprice
1	Product KSBRM	$22.00
2	Product HMLNI	$30.00
3	Product OSFNS	$38.00
4	Product NUNAW	$32.00
5	Product VKCMF	$38.00

Проверка на соответствие шаблону

match_expression [NOT] LIKE pattern [ESCAPE escape_character] – регистро-чувствителен
match_expression [NOT] ILIKE pattern [ESCAPE escape_character] – регистро-нечувствителен

- Символы шаблона

_	любой один символ	`WHERE LName LIKE '_етров'`
%	любая строка, содержащая ноль или более символов	`WHERE BookName LIKE '%компьютер%'` `WHERE BookName ILIKE '%Компьютер%'`
ESCAPE 'символ'	символ, помещаемый перед символом-шаблоном, чтобы символ-шаблон (%, _, [,]) рассматривался как обычный символ	`WHERE c1 LIKE '%10-15!% off%' ESCAPE '!'`

https://postgrespro.ru/docs/postgresql/14/functions-matching#FUNCTIONS-LIKE

Фильтрация по шаблону

```
SELECT   custid
       , contactname
       , contacttitle
FROM "Sales"."Customers" c
WHERE contacttitle LIKE '%Manager%';
```

	custid	contactname	contacttitle
1	10	Bassols, Pilar Colome	Accounting Manager
2	32	Krishnan, Venky	Marketing Manager
3	34	Cohen, Shy	Accounting Manager
4	38	Lee, Frank	Marketing Manager
5	43	Deshpande, Anu	Marketing Manager
6	46	Dressler, Marlies	Accounting Manager
7	48	Szymczak, Radosław	Sales Manager
8	61	Florczyk, Krzysztof	Accounting Manager

```
SELECT   custid
       , contactname
       , contacttitle
FROM "Sales"."Customers" c
WHERE  contactname LIKE 'C%'
    OR contactname LIKE 'L%';
```

	custid	contactname	contacttitle
1	31	Cheng, Yao-Qiang	Sales Associate
2	34	Cohen, Shy	Accounting Manager
3	35	Langohr, Kris	Sales Representative
4	37	Crăciun, Ovidiu V.	Sales Associate
5	38	Lee, Frank	Marketing Manager
6	47	Lupu, Cornel	Owner
7	88	Li, Yan	Sales Manager
8	25	Carlson, Jason	Marketing Manager

Сопоставление с началом строки

Использование функции **starts_with**

```
SELECT   custid
       , contactname
       , contacttitle
FROM "Sales"."Customers" c
WHERE  starts_with(contactname,'L');
```

	custid	contactname	contacttitle
1	35	Langohr, Kris	Sales Representative
2	38	Lee, Frank	Marketing Manager
3	47	Lupu, Cornel	Owner
4	88	Li, Yan	Sales Manager
5	41	Litton, Tim	Sales Manager
6	44	Louverdis, George	Sales Representative
7	87	Ludwig, Michael	Accounting Manager

Использование оператор проверки префикса **^@**

```
SELECT   custid
       , contactname
       , contacttitle
FROM "Sales"."Customers" c
WHERE  contactname ^@ 'C'
    OR contactname ^@ 'L';
```

	custid	contactname	contacttitle
1	31	Cheng, Yao-Qiang	Sales Associate
2	34	Cohen, Shy	Accounting Manager
3	35	Langohr, Kris	Sales Representative
4	37	Crăciun, Ovidiu V.	Sales Associate
5	38	Lee, Frank	Marketing Manager
6	47	Lupu, Cornel	Owner
7	88	Li, Yan	Sales Manager
8	25	Carlson, Jason	Marketing Manager

Проверка на соответствие шаблону POSIX*

```
строка SIMILAR TO шаблон [ESCAPE спецсимвол]
строка NOT SIMILAR TO шаблон [ESCAPE спецсимвол]
```

- возвращает **true** или **false** в зависимости от того, соответствует ли данная строка шаблону или нет
  - шаблоны соответствуют определению регулярных выражений в стандарте SQL
  - регулярные выражения SQL представляют собой гибрид синтаксиса LIKE с синтаксисом обычных регулярных выражений POSIX
- Условие SIMILAR TO истинно, только если шаблон соответствует всей строке
  - это отличается от условий с регулярными выражениями, в которых шаблон может соответствовать любой части строки

Символы шаблона POSIX

.	любой один символ
[...]	любой одиночный символ в диапазоне или наборе
[^...]	любой символ, кроме указанных в диапазоне или наборе
*****	повторение предыдущего элемента 0 и более раз
+	повторение предыдущего элемента 1 и более раз
?	вхождение предыдущего элемента 0 или 1 раз
{m}	повторение предыдущего элемента ровно m раз
{m,}	повторение предыдущего элемента m или более раз
{m,n}	повторение предыдущего элемента не менее чем m и не более чем n раз
()	объединение нескольких элементов в одну логическую группу
|	выбор (одного из двух вариантов)
^	привязывает шаблон к началу строки
$	привязывает шаблон к концу строки

https://postgrespro.ru/docs/postgresql/12/functions-matching#FUNCTIONS-POSIX-REGEXP

Фильтрация с помощью SIMILAR TO

```
SELECT   custid
       , contactname
       , contacttitle
FROM "Sales"."Customers" c
WHERE  contactname SIMILAR to 'S(i|m)%';
```

	custid	contactname	contacttitle
1	33	Sigurdarson, Hallur	Owner
2	36	Smith, Denise	Sales Representative
3	89	Smith Jr., Ronaldo	Owner

```
SELECT   custid
       , contactname
       , contacttitle
FROM "Sales"."Customers" c
WHERE  contactname SIMILAR to '[^B-R]%';
```

	custid	contactname	contacttitle
1	33	Sigurdarson, Hallur	Owner
2	36	Smith, Denise	Sales Representative
3	42	Steiner, Dominik	Marketing Assistant
4	45	Sunkammurali, Krishna	Owner
5	48	Szymczak, Radosław	Sales Manager
6	51	Taylor, Maurice	Marketing Assistant
7	75	Wojciechowska, Agnieszka	Sales Manager
8	78	Young, Robin	Marketing Assistant
9	82	Veninga, Tjeerd	Sales Associate

Операторы регулярных выражений POSIX

Оператор	Описание	Пример
~	Проверка соответствия строки регулярному выражению с учётом регистра	'thomas' ~ 't.*ma' → true
~*	Проверка соответствия строки регулярному выражению без учёта регистра	'thomas' ~* 'T.*ma' → true
!~	Проверка несоответствия строки регулярному выражению с учётом регистра	'thomas' !~ 't.*max' → true
!~*	Проверка несоответствия строки регулярному выражению без учёта регистра	'thomas' !~* 'T.*ma' → false

Фильтрация с использованием операторов регулярных выражений

```
SELECT region
FROM "Sales"."Customers" c
where region ~ '.*ra$';
```

	region
1	Lara
2	Táchira

```
SELECT   companyname, contactname, city
FROM "Sales"."Customers" c
where  contactname ~ '^(B|K|S).*(e|k)$';
```

	companyname	contactname	city
1	Customer EEALV	Bassols, Pilar Colome	Tsawassen
2	Customer LVJSO	Smith, Denise	Elgin
3	Customer IAIJK	Steiner, Dominik	Vancouver
4	Customer DTDMN	Bueno, Janaina Burdan, Neville	Madrid