УДК 681.3

**В.А.Резниченко**
**Институт программных систем НАН Украины**

# Работа с окнами в SQL

*Рассматриваются вопросы использования в языке SQL механизма окон и специальных оконных функций. Эти возможности языка раскрываются на многочисленных примерах запросов. Для демонстрации запросов используется СУБД Oracle и его инструмент SQL*Plus.*

*Розглядаються питання використання в мові SQL механізму вікон і спеціальних віконних функцій. Ці можливості мови розкриваються на численних прикладах запитів. Для демонстрації запитів використовується СУБД Oracle і його інструмент SQL * Plus.*

*The problem of using in the SQL language the windows mechanism and the special window functions is considered. These language features are represented in numerous examples of queries. To demonstrate the queries the DBMS Oracle and its tool SQL * Plus is used.*

**Ключевые слова: б**аза данных, язык запросов, работа с окнами в SQL, оконные функции

## 1 Таблицы с окнами

В предложении `SELECT` указанное в нем табличное выражение может ассоциироваться с так называемыми окнами — изменяющимися структурами данных. С одним табличным выражением может ассоциироваться одно или несколько окон. Окно определяется либо непосредственно (явно) с помощью фразы `WINDOW` в предложении `SELECT`, либо неявно при указании функции, которая предполагает использование окна.

Окно предполагает задание способа разбиения строк таблицы на разделы (части), указание упорядочения строк внутри разделов, а также способа выделения динамического подмножества строк внутри любого из разделов (фрейм).

### 1.1 Составляющие окна

Итак, *окно таблицы* определяется через такие понятия, как разделы, упорядоченность строк внутри разделов и фреймы. Рассмотрим их.

*Разделы окна* определяются точно так же, как и группы, то есть указанием одного или нескольких столбцов, равенство значений которых определяет строки, входящие в одну группу. Однако в отличие от сгруппированной таблицы, в таблице с окном каждая строка раздела сохраняется в результирующей таблице запроса. Формирование разделов окна является факультативным, и если соответствующая фраза построения разделов отсутствует, то предполагается, что имеется один раздел, совпадающий со всей таблицей.

Упорядочение строк внутри разделов определяется точно так же, как и обычное упорядочение строк таблицы с помощью фразы `ORDER BY`.

Факультативно окно может определять фрейм для КАЖДОЙ СТРОКИ таблицы. *Фрейм* — это изменяющаяся совокупность строк, ассоциируемая с каждой строкой, таблицы в пределах раздела.

Окна используются в так называемых *оконных функциях*, входящих в состав функций аналитической обработки данных. Оконные функции могут использоваться только во фразах `SELECT` и `ORDER BY`. Это объясняется тем, что построение окон производится после выполнения фраз `FROM`, `WHERE`, `GROUP BY`, `HAVING`. Если с запросом связано несколько оконных функций, то каждая из них вычисляется независимо, каждая в своем окне или в одном и том же окне. Так как упорядочение строк в разделах в общем случае определяет неполную сортировку, то процедура вычисления оконных функций является недетерминированной.

### 1.2 Определение окна

Явное определение окон производится с помощью фразы `WINDOW`, которая располагается вслед за фразой `HAVING`, и имеет следующий вид:

```
WINDOW список_определений_окон
```

где определение окна в этом списке производится согласно следующему синтаксису:

```
имя_нового_окна AS спецификация_окна
```

причем спецификация окна состоит из следующих четырех факультативных конструкций, заключенных в круглые скобки:

```
([имя_существующего_окна]
 [определени**е**_разделов]
 [определение_упорядочения]
 [определение_фрейма])
```

Итак, с помощью одной фразы WINDOW можно определить множество окон. Определение окна предполагает задание его имени и ряда параметров. Первым из них является факультативный параметр имени существующего окна. В этом случае предполагается, что определяемое окно наследует свойства указанного ранее во фразе WINDOW окна за исключением тех свойств, которые задаются явно при его определении.

### *1.2.1 Определение раздела и сортировка строк*

Определение разделов окна имеет следующий синтаксис:

```
PARTITION BY список_имен_столбцов
```

Как мы уже сказали, разделы формируются согласно равенству значений всех столбцов приводимого списка. Определение упорядоченности имеет следующий формат:

```
ORDER BY спецификация_сортировки_1
 [,спецификация_сортировки_2]…
```

Эта фраза является идентичной синтаксически одноименной фразе предложения SELECT. Смысловое различие заключается в том, сортировка производится в пределах разделов, а не по всей таблице. Строки, которые имеют одинаковое значение столбцов, по которым производится сортировка, называются равноправными (одноранговыми).

### *1.2.2 Определение фрейма*

Наиболее сложной является фраза, определяющая фрейм, соответствующий любой строке таблицы. Обратим ваше внимание на то, что диапазон строк, составляющих фрейм, определяется по отношению к текущей строке. А так как при вычислении оконных функций предполагается последовательный перебор всех строк таблицы, то все они последовательно становятся текущими и, таким образом, обладающими собственным фреймом. Эта фраза выглядит следующим образом:

```
{ROWS|RANGE} диапазон_фрейма
 [исключение_строк]
```

Ключевое слово ROWS|RANGE указывает, в каких единицах будет указываться диапазон фрейма.

- ROWS свидетельствует, что диапазон указывается в физических единицах (строках);
- RANGE свидетельствует, что диапазон указывается в единицах логического смещения.

Объяснение см. далее.

**Примечание:** для указания диапазона фрейма обязательно должна присутствовать фраза ORDER BY, так как диапазон определяется согласно выбранному упорядочению.

Диапазон фрейма определяется его двумя крайними строками таблицы. Допускается явное задание одной или двух строк диапазона. В первом случае предполагается, что задаваемая строка указывает на начало диапазона, а его конец совпадает с текущей строкой. В этом случае диапазон указывается следующим образом:

```
UNBOUNDED PRECEDING |
 CURRENT ROW |
 значение PRECEDING
```

Приведенные фразы имеют следующий смысл:

- UNBOUNDED PRECEDING — начало диапазона совпадает с первой строкой текущего раздела.
- CURRENT ROW — начало диапазона совпадает с текущей строкой. Следовательно диапазон вырождается в текущую строку.
- значение PRECEDING — если единица измерения диапазона ROWS, то указывается, какая по счету предшествующая строка по отношению к текущей является начальной строкой диапазона. Если единица измерения диапазона RANGE, то указывается, насколько меньшее значение по отношению к текущей строке содержит начальная строка диапазона (имеется в виду значение столбца, по которому произведено упорядочение, при этом следует помнить, что при использовании ключевого слова RANGE допускается указывать сортировку только по одному столбцу (выражению))

Приведем примеры, предполагая, что упорядочение произведено по столбцу Salary.

ROWS 10 PRECEDING – диапазон фрейма начинается на 10 строк выше текущей строки.

RANGE 250 PRECEDING – диапазон начинается со строки, у которой значение столбца Salary на 250 меньше, чем значение Salary текущей строки.

Для указания двух строк диапазона фрейма используется следующий синтаксис:

```
   BETWEEN
{UNBOUNDED PRECEDING | CURRENT ROW | значение {PRECEDING | FOLLOWING }
AND
{UNBOUNDED FOLLOWING | CURRENT ROW | значение {PRECEDING | FOLLOWING }
```

Здесь имеются две дополнительных фразы по сравнению с предыдущим случаем, а именно:

- `UNBOUNDED FOLLOWING` —конец диапазона совпадает с последней строкой текущего раздела;
- `значение FOLLOWING` — начальная или конечная строка диапазона располагается за текущей строкой на указанном расстоянии точно так же, как и во фразе `значение PRECEDING`

Обратите внимание, этот синтаксис позволяет определять диапазон таким образом, что он может располагаться выше текущей строки, включать текущую строку или ниже текущей строки.

Приведем примеры, предполагая, как и выше, что упорядочение произведено по столбцу `Salary`.

| | |
|---|---|
| `RANGE BETWEEN UNBOUNDED PRECEDING AND 250 PRECEDING` | Диапазон от начала раздела до строки, имеющей Salary на 250 меньше, чем у текущей строки. |
| `RANGE BETWEEN 250 PRECEDING AND 150 PRECEDING` | Диапазон, начальная и конечная строки которого имеют Salary на 250 и 150 соответственно меньше, чем у текущей строки. |
| `RANGE BETWEEN 350 PRECEDING AND CURRENT ROW` | Диапазон от строки, имеющей Salary на 250 меньше, чем у текущей строки, и до текущей строки. |
| `RANGE BETWEEN 350 PRECEDING AND UNBOUNDED FOLLOWING` | Диапазон от строки, имеющей Salary на 350 меньше, чем у текущей строки, и до конца раздела |
| `RANGE BETWEEN CURRENT ROW AND 250 FOLLOWING` | Диапазон от текущей строки и до имеющей Salary на 250 больше, чем у текущей строки |
| `RANGE BETWEEN 100 FOLLOWING AND 250 FOLLOWING` | Диапазон, начальная и конечная строки которого имеют Salary на 100 и 250 соответственно больше, чем у текущей строки. |

Если определение фрейма отсутствует, то по умолчанию предполагается диапазон:

```
BETWEEN UNBOUNDED PRECEDING AND
 CURRENT ROW
```

Конструкция `исключение_строк` позволяет выбросить из диапазона некоторые строки. Для указания исключаемых строк можно использовать одну из следующих фраз:

```
EXCLUDE CURRENT ROW |
 EXCLUDE GROUP | EXCLUDE TIES |
 EXCLUDE NO OTHERS
```

Они означают следующее:

`EXCLUDE CURRENT ROW` — если в диапазон входит текущая строка, то она из него исключается;

`EXCLUDE GROUP` — если в диапазон входит текущая строка, то она исключается вместе с равноправными ей строками;

`EXCLUDE TIES` — из диапазона исключаются все строки, являющиеся равноправными текущей строке; сама текущая строка остается;

`EXCLUDE NO OTHERS` — никакие строки не исключаются. Является значением по умолчанию в случае отсутствия исключения фрейма.

### 1.3 Примеры определения окон

Приведем примеры фраз `WINDOW`.

```
 WINDOW Tch_dep_post AS
(PARTITION BY DepFK,Post ORDER BY Salary UNBOUNDED PRECEDING)
```

Здесь определяется окно с именем `Tch_dep_post`, имеющее разделы, построенные согласно уникальности значений столбцов DepFK, Post, упорядочение внутри разделов определяется на основе значений столбца Salary и, наконец, для каждой строки его фрейм

состоит из строк от начала раздела до самой строки, причем никакого исключения строк из диапазона нет.

```
 WINDOW Tch_post AS
(PARTITION BY Post, ORDER BY Salary+Rise),
     Tch_post_2 AS
(Tch_post ROWS BETWEEN UNBOUNDED PRECEDING AND 10 FOLLOWING
     EXCLUDE TIES),
     Tch_post_3 AS
(Tch_post ROWS BETWEEN 1 PRECEDING AND 1 FOLLOWING
     EXCLUDE CURRENT ROW)
```

Здесь во фразе WINDOW определены три окна. Первое окно имеет имя Tch_post, его разделы формируются по уникальности значений столбца Post, упорядоченность определяется по выражению Salary+Rise, фрейм задается по умолчанию, то есть от первой строки раздела и до текущей, исключение строк диапазона отсутствует. Второе окно имеет имя Tch_post_2, оно определяется на основе предыдущего окна Tch_post и, следовательно наследует его свойства за исключением тех, которые в Tch_post_2 определены явно, а именно, фрейм определяется от первой строки раздела и до 10-й по порядку строки после текущей, причем из этого диапазона выбрасывается все строки, равноправные текущей. Наконец, третье окно под именем Tch_post_3 также определяется на основе окна Tch_post с указанием собственного фрейма, имеющего диапазон по одной строке выше и ниже текущей, причем сама текущая строка в диапазон не включается.

## 2 Оконные функции

*Оконная функция* — это такая функция, результат вычисления которой для заданной строки определяется на основе фрейма окна этой строки. Оконная функция может использоваться только в списке фразы SELECT или во фразе ORDER BY. Оконные функции имеют следующий синтаксис:

```
имя_функции_с_параметрами OVER
{имя_окна | спецификация_окна}
```

Сначала указывается имя функции с возможными ее параметрами, а затем после ключевого слова OVER приводится окно, по отношению к которому указанная функция вычисляется. Окно можно указать двумя способами. Либо задается имя окна, определенное во фразе WINDOW, либо приводится спецификация окна с указанием всех необходимых его свойств.

Стандарт SQL предлагает следующие оконные функции:

- функции ранга;
- функции распределения;
- функция нумерации строк;
- агрегатные оконные функции.

Рассмотрим эти функции. Отметим, что среди перечисленных выше оконных функций только агрегатные оконные функции используют фреймы.

**Примечание.** Обратите внимание, ВСЕ перечисленные выше типы оконных функций, кроме агрегатных, не предполагают использование фреймов, поэтому их указание при спецификации окна рассматривается как ошибка.

### 2.1 Функции ранга

Функции ранга вычисляют порядковый ранг строк в пределах разделов окон с учетом указанного упорядочения строк в разделах. Равноправные строки имеют одинаковый ранг. Имеются две функции ранга, RANK и DENSE_RANK.

Функция RANK вычисляет ранг следующим образом: ранг строки равен 1 плюс количество строк раздела, предшествующих заданной строке за исключением строк, равноправных ей. Это, в частности, предполагает, что при существовании равноправных строк номера рангов не будут непрерывными. Приведем пример.

**Примечание**. Все последующие примеры демонстрируются с использование Oracle и его инструмента SQL*Plus. Окна будут специфицироваться в самих функциях.

Последующие примеры используют таблицу преподавателей Teacher со следующими столбцами:

- DepFK – номер кафедры, на которой работает преподаватель;
- TchPK – номер преподавателя, является первичным ключом;
- Name – фамилия преподавателя;
- Post – должность преподавателя;

- Salary – ставка преподавателя.

**Запрос**. Из таблицы преподавателей вывести номера кафедр, фамилии преподавателей, их ставку и ранг ставки в пределах одной кафедры.

```
SELECT DepFK  AS "Кафедра", Name AS "Преподаватель",
       Salary AS "Ставка",
       RANK() OVER (PARTITION BY DepFK ORDER BY Salary)
              AS "Ранг по ставке"
FROM TEACHER;
```

| Кафедра | Преподаватель | Ставка | Ранг по ставке |
|---|---|---|---|
| 2 | Мартынов | 530 | 1 |
| 2 | Козлутин | 530 | 1 |
| 2 | Недеведеев | 570 | 3 |
| 2 | Вибровский | 570 | 3 |
| 2 | Воропаев | 570 | 3 |
| 2 | Кузинцев | 630 | 6 |
| 2 | Хоренко | 670 | 7 |
| 2 | Завратинский | 770 | 8 |
| 2 | Рамишевский | 830 | 9 |
| 2 | Лекарь | 890 | 10 |
| 2 | Сидоров | 1070 | 11 |
| 2 | Резван | 1100 | 12 |
| | Дараганов | 570 | 1 |
| | Наумов | 770 | 2 |
| | Ахромеев | 1050 | 3 |

15 строк выбрано.

Как, видим, построены два раздела, для кафедры 2 и кафедры без номера. Нумерация рангов в каждом разделе начинается сначала. Нумерация рангов в пределах раздела не является непрерывной (в ней имеются разрывы), например, среди номеров отсутствуют 2, 4, 5.

Функция DENSE_RANK вычисляет ранг строки таким образом, что он равен количеству строк, предшествующих заданной, включая и ее, которые отличаются относительно условия упорядочения в разделах окна. Это, в частности, подразумевает, что в пределах раздела номера рангов являются последовательными, то есть без разрывов. Приведем пример функции DENSE_RANK

```
SELECT DepFK  AS "Кафедра", Name AS "Преподаватель",
    Salary AS "Ставка",
    DENSE_RANK() OVER (PARTITION BY DepFK ORDER BY Salary)
           AS "Ранг по ставке"
FROM TEACHER;
```

| Кафедра | Преподаватель | Ставка | Ранг по ставке |
|---|---|---|---|
| 2 | Мартынов | 530 | 1 |
| 2 | Козлутин | 530 | 1 |
| 2 | Недеведеев | 570 | 2 |
| 2 | Вибровский | 570 | 2 |
| 2 | Воропаев | 570 | 2 |
| 2 | Кузинцев | 630 | 3 |
| 2 | Хоренко | 670 | 4 |
| 2 | Завратинский | 770 | 5 |
| 2 | Рамишевский | 830 | 6 |
| 2 | Лекарь | 890 | 7 |
| 2 | Сидоров | 1070 | 8 |

```
         2 Резван              1100              9
           Дараганов            570              1
           Наумов               770              2
           Ахромеев            1050              3
15 строк выбрано.
```

Как видите, в отличие от RANK, в данном случае в пределах раздела нумерация производится без пропусков номеров.

**2.2 Функции распределения**

Функции распределения вычисляют относительный ранг строки в пределах раздела окна, выраженный в виде приближенного числового коэффициента распределения между 0,0 и 1,0. Имеются две функции распределения с именами PERCENT_RANK и CUME_DIST.

В функции PERCENT_RANK относительный ранг строки определяется по формуле (RK–1)/(NR–1), где RK является рангом (RANK) строки, а NR — количество строк в разделе окна. Приведем пример.

```
SELECT DepFK, Name, Salary,
    RANK() OVER (PARTITION BY DepFK ORDER BY Salary) AS Rank,
    PERCENT_RANK() OVER (PARTITION BY DepFK ORDER BY Salary)
                    AS Percent_rank
FROM  TEACHER
WHERE LOWER(Post) != 'доцент'
ORDER BY DepFK, Salary;
```

```
    DEPFK NAME           SALARY    RANK PERCENT_RANK
------ ------------ -------- ------ ------------
     2 Козлутин          530      1            0
     2 Мартынов          530      1            0
     2 Вибровский        570      3          ,25
     2 Воропаев          570      3          ,25
     2 Недеведеев        570      3          ,25
     2 Кузинцев          630      6         ,625
     2 Хоренко           670      7          ,75
     2 Сидоров          1070      8         ,875
     2 Резван           1100      9            1
       Дараганов         570      1            0
       Ахромеев         1050      2            1
11 строк выбрано.
```

В функции CUME_DIST относительный ранг строки определяется по формуле RK/NR, где RK является рангом (RANK) строки, а NR — количество строк в разделе. Приведем пример.

```
SELECT DepFK, Name, Salary,
    RANK() OVER (PARTITION BY DepFK ORDER BY Salary) AS Rank,
    CUME_DIST() OVER (PARTITION BY DepFK ORDER BY Salary)
                    AS Cume_dist
FROM  TEACHER
WHERE LOWER(Post) NOT IN ('доцент', 'ассистент')
ORDER BY DepFK, Salary;
```

```
    DEPFK NAME           SALARY    RANK  CUME_DIST
------ ------------ -------- ------ ----------
     2 Недеведеев        570      1         ,2
     2 Кузинцев          630      2         ,4
     2 Хоренко           670      3         ,6
     2 Сидоров          1070      4         ,8
     2 Резван           1100      5          1
```

```
        Дараганов          570       1          ,5
        Ахромеев          1050       2           1
   7 строк выбрано.
```

**2.3 Функция нумерации строк и анализ Top N**

Функция нумерации строк ROW_NUMBER вычисляет последовательный номер строки в пределах раздела окна с учетом заданного упорядочения. Пример:

```
   SELECT DepFK, Name, TchPK, ROW_NUMBER()
       OVER (PARTITION BY DepFK ORDER BY TchPK) AS "Пор.номер"
FROM   TEACHER
WHERE  LOWER(Post) NOT IN ('доцент', 'ассистент');
    DEPFK NAME           TCHPK  Пор.номер
------ ------------ ------- ----------
     2 Хоренко            2          1
     2 Кузинцев           5          2
     2 Резван            10          3
     2 Недеведеев        11          4
     2 Сидоров           16          5
       Ахромеев          13          1
       Дараганов         15          2
   7 строк выбрано.
```

Так как функция ROW_NUMBER позволяет нумеровать строки, то ее можно использоваться для решения задач класса Top N. Приведем пример, как это можно сделать.

**Запрос**. По каждой из должностей вывести по два преподавателя, имеющих набольшую ставку.

```
   BREAK ON Post skip 1
   SELECT *
FROM (SELECT Post, Name, Salary, ROW_NUMBER()
             OVER (PARTITION BY Post ORDER BY Salary DESC) Top2
      FROM TEACHER )
WHERE Top2 <= 2;
   POST           NAME          SALARY    TOP2
-------------- -----------  ------- -------
ассистент      Вибровский       570       1
               Воропаев         570       2

доцент         Лекарь           890       1
               Рамишевский      830       2

   Преподаватель  Хоренко          670       1
               Кузинцев         630       2

   профессор      Резван          1100       1
               Сидоров         1070       2
   8 строк выбрано.
```

Данный способ позволяет находить TOP N строки в пределах формируемых разделов (которых может быть один, совпадающий со всей таблицей).

**2.4 Оконные агрегатные функции**

Все агрегатные функции имеют разновидность, которая позволяет им работать с окнами. Смысл их остается тем же, только вместо области действия в виде множества строк всей таблицы или строк, принадлежащих группам, определенным фразой GROUP BY запроса, они

имеют область, определенную фреймом текущей строки раздела окна. Это, в частности, означает, что групповые функции вычисляются по отношению ко всем строкам таблицы.

Для агрегатных функций отсутствие фрейма окна означает, что оно совпадает с разделом окна для всех строк раздела.

В последующих примерах мы будем использовать таблицу (ACCOUNT), в которой по каждому из ее кафедр (DepNO) ведется учет финансовой деятельности (Income) в определенные моменты времени (AccDate). В столбце Income указываются доходы, полученные кафедрой с момента предыдущей отчетной даты. Таблица имеет следующую структуру:

```
CREATE TABLE ACCOUNT
(DepNo  NUMERIC(2),
 AccDate  Date,
 Income NUMERIC(5));
```

Ее содержимое следующее:

```
SELECT * FROM ACCOUNT;
 DEPNO ACCDATE  INCOME
------ -------- ------
    10 01.07.06    100
    10 07.07.06    200
    10 13.07.06    -50
    10 23.07.06    350
    10 28.07.06    150
    10 10.08.06    100
    20 03.07.06     50
    20 10.07.06   -100
    20 15.07.06    150
    20 25.07.06    150
    20 28.07.06    200
    20 10.08.06   -100
12 строк выбрано.
```

### *2.4.1 Кумулятивное агрегирование*

Под кумулятивными в данном случае мы подразумеваем такие данные, которые являются агрегированными по отношению к текущей строке таблицы, начиная с некоторой фиксированной предыдущей строки. Приведем несколько примеров.

**Запрос**. В целом по всему факультету по каждой из дат, когда фиксировалось финансовое состояние любой из кафедр, вывести суммарное значение доходов к этому моменту времени. Результаты упорядочить по дате.

```
SELECT DepNO AS "Кафеда", AccDate AS "Дата", Income AS "Доход",
    SUM(Income) OVER (ORDER BY AccDate
                      ROWS UNBOUNDED PRECEDING)
                      AS "Сум.доход"
FROM   ACCOUNT
ORDER BY AccDate;
```

```
   Кафедра  Дата        Доход  Сум.доход
---------- -------- ---------- ----------
        10 01.07.06        100        100
        20 03.07.06         50        150
        10 07.07.06        200        350
        20 10.07.06       -100        250
        10 13.07.06        -50        200
        20 15.07.06        150        350
        10 23.07.06        350        700
        20 25.07.06        150        850
        20 28.07.06        200       1050
        10 28.07.06        150       1200
        20 10.08.06       -100       1100
        10 10.08.06        100       1200
      12 строк выбрано.
```

При спецификации окна мы не указали способ построения разделов, поэтом вся таблица рассматривается как один раздел, который упорядочивается по столбцу AccDate. Фраза ROWS UNBOUNDED PRECEDING указывает, что для каждой строки строится фрейм от начала раздела (начала таблицы) и до этой строки. Агрегатная функция SUM вычисляется по столбцу Income в пределах фрейма текущей строки и становится значением столбца «Сум. Доход» для этой текущей строки.

**Запрос**. Получить ту же информацию, что и в предыдущем запросе, но в пределах каждой из кафедр. Результат упорядочить по кафедрам и датам.

```
BREAK ON Кафедра skip 1
SELECT DepNO AS "Кафедра", AccDate AS "Дата", Income AS "Доход",
    SUM(Income) OVER (PARTITION BY DepNO ORDER BY AccDate
                      ROWS UNBOUNDED PRECEDING)
                      AS "Сум. доход"
FROM   ACCOUNT
ORDER BY DepNO, AccDate;
   Кафедра Дата        Доход Сум. Доход
---------- -------- ------- ----------
        10 01.07.06     100        100
           07.07.06     200        300
           13.07.06     -50        250
           23.07.06     350        600
           28.07.06     150        750
           10.08.06     100        850

           20 03.07.06      50         50
           10.07.06    -100        -50
           15.07.06     150        100
           25.07.06     150        250
           28.07.06     200        450
           10.08.06    -100        350
   12 строк выбрано.
```

**Примечание** Расположенное в начале запроса выражение BREAK ON Кафедра skip 1 является командой SQL*Plus, а не предложением SQL. Она удаляет дубликаты в столбце Кафедра и перед строкой с изменяющимся значением этого столбца вставляет пустую строку.

Отличие текста этого запроса от предыдущего заключается в том, что при спецификации окна мы добавили определение разделов по номеру кафедры (PARTITION BY DepNO) и указали другой способ упорядочения полученных результатов (по DepNO, AccDate).

Кумулятивное агрегирование может производиться также вычислением нарастающего

среднего значения, как это показано в следующем примере.

**Запрос**. В таблице ACCOUNT, упорядоченной по столбцу AccDate, для каждой строки кафедры 10 вывести среднее значение столбца Income, начиная с первой строки и по текущую.

```
   SELECT DepNO AS "Кафедра", AccDate AS "Дата", Income AS "Доход",
       AVG(Income) OVER (ORDER BY AccDate
                         ROWS UNBOUNDED PRECEDING)
                         AS "Сред.доход"
FROM   ACCOUNT
WHERE  DepNO = 10
ORDER BY AccDate;
   Кафедра Дата       Доход Сред.доход
------- -------- ------- ----------
     10 01.07.06     100        100
        07.07.06     200        150
        13.07.06     -50 83,3333333
        23.07.06     350        150
        28.07.06     150        150
        10.08.06     100 141,666667
   6 строк выбрано.
```

***2.4.2 Интервальное агрегирование***

С помощью фрейма можно для каждой строки определить свой собственный интервал строк, в пределах которых будет вычисляться агрегированное значение для текущей строки. Такой интервал перемещается вместе с перемещением текущей строки. Приведем примеры.

**Запрос**. В таблице ACCOUNT для каждой строки финансового отчета кафедры 10 подсчитать среднее значение прибыли в диапазоне одной строки выше и одной строки ниже. Подсчет произвести при условии упорядочения строк по дате.

```
   SELECT DepNO AS "Кафедра", AccDate AS "Дата", Income AS "Доход",
       AVG(Income) OVER (ORDER BY AccDate
       ROWS BETWEEN 1 PRECEDING AND 1 FOLLOWING) AS "Сред.доход"
FROM   ACCOUNT
WHERE  DepNO = 10
ORDER BY AccDate;
   Кафедра Дата       Доход Сред.доход
------- -------- ------- ----------
     10 01.07.06     100        150
        07.07.06     200 83,3333333
        13.07.06     -50 166,666667
        23.07.06     350        150
        28.07.06     150        200
        10.08.06     100        125
   6 строк выбрано.
```

Здесь мы использовали интервалы, которые фиксируются количеством строк до и после текущей, однако имеется возможность специфицировать диапазон фрейма логически, указанием фразы RANGE в спецификации фрейма.

**Запрос**. В таблице ACCOUNT для каждой строки финансового отчета кафедры 20 подсчитать суммарное значение прибыли в диапазоне 7 дней до и 7 дней после даты финансового отчета текущей строки. Подсчет произвести при условии упорядочения строк по дате.

```
    SELECT DepNO AS "Кафедра", AccDate AS "Дата", Income AS "Доход",
        SUM(Income) OVER (ORDER BY AccDate
        RANGE BETWEEN INTERVAL '7' DAY PRECEDING
              AND INTERVAL '7' DAY FOLLOWING) AS "Сум.доход"
FROM   ACCOUNT
WHERE  DepNO = 20
ORDER BY AccDate;
   Кафедра Дата       Доход Сум.доход
------- -------- ------- ---------
     20 03.07.06      50       -50
        10.07.06    -100       100
        15.07.06     150        50
        25.07.06     150       350
        28.07.06     200       350
        10.08.06    -100      -100
   6 строк выбрано.
```

Здесь для указания диапазона в 7 дней до и после текущей строки мы воспользовались интервальной константой (`INTERVAL '7' DAY`). Разность/сумма даты и интервала дает дату меньше/больше указанной даты на величину интервала, в нашем случае минус/плюс 7 дней.

Синтаксис фразы спецификации фрейма позволяет определить произвольные диапазоны строк относительно текущей строки.

### 2.5 Использование нескольких оконных функций

В одном предложении `SELECT` можно использовать множество оконных функций, причем каждая может иметь свое собственное окно. Приведем пример.

```
    BREAK ON Кафедра SKIP 1
    SELECT DepNO AS "Кафедра", AccDate AS "Дата", Income AS "Доход",
        SUM(Income) OVER (ORDER BY AccDate
                          ROWS UNBOUNDED PRECEDING)
                          AS "Сум.доход",
        AVG(Income) OVER (PARTITION BY DepNO
                          ORDER BY AccDate
                          ROWS BETWEEN 1 PRECEDING AND
                          1 FOLLOWING) AS "Сред.доход",
        RANK() OVER (PARTITION BY DepNO ORDER BY AccDate)
                          AS "Ранг"
FROM   ACCOUNT
ORDER BY DepNO, AccDate;
   Кафедра Дата      Доход  Сум.доход Сред.доход  Ранг
------- -------- ------ ---------- ---------- -----
     10 01.07.06    100        100        150     1
        07.07.06    200        350 83,3333333     2
        13.07.06    -50        200 166,666667     3
        23.07.06    350        700        150     4
        28.07.06    150       1200        200     5
        10.08.06    100       1200        125     6

        20 03.07.06     50        150        -25     1
        10.07.06   -100        250 33,3333333     2
        15.07.06    150        350 66,6666667     3
        25.07.06    150        850 166,666667     4
        28.07.06    200       1050 83,3333333     5
        10.08.06   -100       1100         50     6
   12 строк выбрано.
```

Здесь функция SUM представляет собой кумулятивное суммирование по всей таблице, функция AVG применяется к диапазонам ± 1 строка относительно текущей, причем для каждой кафедры отдельно, а RANK применяется по кафедрам с учетом упорядочения по дате финансового отчета. В этом случае для каждой функции строится свое собственное окно независимо друг от друга и они вычисляются каждая в своем окне.

При использовании многих окон следует помнить, что они НЕЗАВИСИМЫ. Это, в частности, означает, если вы определили два идентичных окна, то из этого не обязательно следует, что они будут одними и теми же. Это может быть связано, например, с тем, что задаваемая в окне сортировка является не полностью определенной, о чем мы говорили в начале статьи. Если вы хотите, чтобы несколько функций использовали действительно одно окно, то его следует определить самостоятельно с помощью фразы WINDOWS, а в оконных функциях использовать его имя.

### 2.6 Совместная работа с группами и разделами

Согласно определению фразы WINDOW, она отрабатывается после фраз FROM, WHERE, GROUP BY, HAVING. Однако, исходя из описания стандарта SQL, абсолютно не понятно, что это значит. Например, не понятно, как следует интерпретировать ситуацию, если предложение SELECT содержит фразу GROUP BY, а окна имеют фразу PARTITION BY. На этот вопрос нет вразумительного ответа. В связи с этим приведем один тип запроса, который предполагает совместное использование этих двух возможностей, и покажем, как в этом случае следует правильно писать запросы на SQL.

Пусть нам необходимо сначала получить следующую таблицу:

```
   SELECT DepFK, Post, SUM(Salary), AVG(Rise)
FROM   TEACHER
GROUP BY DepFK, Post
ORDER BY DepFK;
    DEPFK POST          SUM(SALARY)  AVG(RISE)
------ ------------ ----------- ----------
     2 ассистент            2200        190
     2 доцент               2490     383,66
     2 преподаватель        1870     243,33
     2 профессор            2170        470
       доцент                770        350
       преподаватель         570        250
       профессор            1050        500
   7 строк выбрано.
```

Теперь, если мы хотим к этой таблице добавить столбец, который бы содержал для каждой кафедры кумулятивную сумму ставки текущей строки, начиная от первой строки раздела, то попытка написать запрос следующим образом привела бы к неудаче.

```
   SELECT DepFK, Post, SUM(Salary), AVG(Rise),
       SUM(Salary) OVER (PARTITION BY DepFK ORDER BY Salary
                         ROWS UNBOUNDED PRECEDING) AS XXX
FROM TEACHER
GROUP BY DepFK, Post
ORDER BY DepFK;
   SUM(Salary) OVER (PARTITION BY DepFK ORDER BY Salary
    *
ошибка в строке 2:
ORA-00979: выражение не является выражением GROUP BY
```

В случаях, когда оконные функции следует применять к результатам выполнения всего сгруппированного запроса, как в нашем случае, его следует разместить во фразе FROM, а во внешнем запросе следует использовать только

оконные функции. Так, например, наш запрос должен выглядеть следующим образом.

```
   SELECT DepFK, Post, Sal_Sum, Rise_Avg,
       SUM(Sal_Sum) OVER (PARTITION BY DepFK ORDER BY Sal_Sum
                          ROWS UNBOUNDED PRECEDING) AS Sal_Cum
FROM (SELECT DepFK, Post, SUM(Salary) AS Sal_Sum,
             AVG(Rise) AS Rise_Avg
      FROM TEACHER
      GROUP BY DepFK, Post);
    DEPFK POST             SAL_SUM   RISE_AVG    SAL_CUM
------ ------------ ---------- ---------- ----------
     2 преподаватель        1870     243,33       1870
     2 профессор            2170        470       4040
     2 ассистент            2200        190       6240
     2 доцент               2490     383,66       8730
       преподаватель         570        250        570
       доцент                770        350       1340
       профессор            1050        500       2390
   7 строк выбрано.
```

Приведем еще один пример, когда оконные агрегатные функции с использованием разделов задаются в подзапросе фразы FROM, а группирование производится во внешнем запросе. Пусть необходимо получить по три преподавателя с самыми высокими ставками на каждой кафедре в виде отдельных столбцов.

```
   SELECT DepFK,
       MAX(DECODE(seq,1,Name,null)) first,
       MAX(DECODE(seq,2,Name,null)) second,
       MAX(DECODE(seq,3,Name,null)) third
FROM   (SELECT DepFK, Name,
               ROW_NUMBER()
                 OVER (PARTITION BY DepFK
                       ORDER BY Salary DESC NULLS LAST) seq
        FROM TEACHER)
   WHERE  seq <= 3
GROUP BY DepFK;
     DEPFK FIRST        SECOND       THIRD
------ ------------ ------------ ---------
     2 Резван       Сидоров      Лекарь
       Ахромеев     Наумов       Дараганов
```

**Вывод**

Итак, использование в SQL механизма окон и специальных функций предоставляет возможность проводить всестороннюю статистическую и аналитическую обработку данных, которая используется в научных исследованиях и экономических расчетах.

**Сведения об авторе:**

**Резниченко Валерий Анатольевич** - к.ф-.м.н., с.н.с., ведущий научный сотрудник Института программных систем НАН Украины, область научных интересов - информационные системы, базы данных и знаний, электронные библиотеки
E-mail: vreznichenko_47@mail.ru

Статья поступила в редакцию 15.08.2011 г.

 