

Базы данных

А5. Анализ данных (продолжение)



Московский государственный технический университет
имени Н.Э. Баумана

Факультет ИБМ

Мар 2025 года

Москва

Артемьев Валерий Иванович © 2025

Дополнительные вычисления при анализе данных

Объединения строк UNION

Итоги по столбцам

Доли (проценты) по строкам и по столбцам

Оконные функции OVER

Отличия по времени

Приведённые отличия по времени

Индексы по времени

Скользящая сумма

Скользящее среднее

Ранжирование

Статистические распределения

Объединение строк UNION



Инструкции UNION соединяют результаты двух и более запросов SELECT, обеспечивающих *промежуточные и общие итоги*, а также «склеивают» таблицы с одинаковым или почти одинаковым составом полей.

1-ая инструкция **SELECT**

UNION

2-ая инструкция **SELECT**

...

UNION

N-ая инструкция **SELECT**

Пример 1. Расчёт итогов численности по столбцам



```
SELECT Образование, -SUM(Пол="М") AS Мужчины,  
-SUM(Пол="Ж") AS Женщины, COUNT(Образование) AS Итого  
FROM СОТРУДНИКИ GROUP BY Образование  
UNION
```

```
SELECT " Всего", -SUM(Пол="М"), -SUM(Пол="Ж"), COUNT(*)  
FROM СОТРУДНИКИ ORDER BY 1 DESC;
```

35д Распределение образование-пол с итогами (UNION)				
Образовани	Мужчины	Женщины	Итого	
среднее специ	1	1	2	
среднее	4	1	5	
высшее	2	2	4	
Итого	7	4	11	

Пример 2. Расчёт долей (процентов) по столбцам



SELECT Образование, -**SUM**(Пол="М") **AS** Мужчины,

-**SUM**(Пол="Ж") **AS** Женщины

FROM СОТРУДНИКИ **GROUP BY** Образование

UNION

SELECT " Доля", -Round(**SUM**(Пол="М")/**COUNT**(*),3)*100&"%",

- Round(**SUM**(Пол="Ж")/**COUNT**(*),3)*100&"%

FROM СОТРУДНИКИ **ORDER BY** 1 **DESC**;

35д Итоговые доли в процентах					
Образование	Мужчины	Женщины			
Доля	63,6%	36,4%			
высшее	2	2			
среднее	4	1			
среднее специальное	1	1			

Оконные функции SQL



Выполняют вычисления по набору строк, связанных с текущей строкой для более детализированных и информативных аналитических отчётов и выполнения сложных вычислений.

В отличие от агрегирования, оконные функции не группируют записи в одну строку результата, а сохраняют исходные строки и добавляют к ним результаты вычислений.

Применение для анализа временных рядов и прогнозирования:

- вычисление нарастающих итогов (скользящих сумм)
- вычисление скользящих средних
- расчёт моды, медианы и процентилей
- ранжирование
- расчёт изменений во времени.

Синтаксис оконных функций



Синтаксис оконных функций включает использование *оконной аналитической функции* и ключевого слова **OVER**, за которым следует *определение окна*:

```
SELECT поле, аналитическая_функция(поле)  
      OVER (PARTITION BY поле_раздела  
            ORDER BY поле_порядка) AS алиас  
FROM таблица;
```

- **PARTITION BY** группирует строки окна. Вычисления выполняются отдельно для каждой группы строк.
- **ORDER BY** определяет порядок строк в каждой группе, что важно для функций, зависящих от порядка строк, таких как **LAG()** и **LEAD**.

Классификация оконных функций



Поддержка СУБД оконных функций



Оконные функции в той или иной степени поддерживаются во всех современных реляционных СУБД:

- MySQL 8.0.2+ (MariaDB 10.2+)
- PostgreSQL 11+
- SQLite 3.28+

Полнее всего окошки реализованы в PostgreSQL и SQLite.

MySQL поддерживает основные возможности, но лишен некоторых более продвинутых.

Oracle 11g+, MS SQL 2012+ и Google BigQuery поддерживают «окошки» примерно так же, как MySQL.

Агрегатные оконные функции

Агрегатные функции:

- суммирование **SUM()**,
- среднее значение **AVG()**,
- подсчёт количества **COUNT()**,
- минимальное значение **MIN()**,
- максимальное значение **MAX()**

Позволяют выполнять *агрегирование данных по окну строк*, вычислять *скользящие агрегаты*.

Эти функции полезны для *анализа временных рядов*, в частности, *скользящих сумм (нарастающих итогов)* и *скользящих средних*.

Пример 3. Анализ успеваемости учеников (1)



Рассчитать все агрегатные оконные функции для учеников.
Исходная таблица оценок учеников по предметам **Student_grades**:

abc name ↑↓	abc subject ↑↓	123 grade ↑↓
Маша	история	3
Маша	математика	4
Маша	русский	3
Маша	физика	5
Петя	математика	3
Петя	русский	4
Петя	физика	5
Петя	история	4

Пример 3. Анализ успеваемости учеников (2)



```
SELECT name, subject, grade,  
       SUM(grade)      OVER (PARTITION BY name) AS sum_grade,  
       AVG(grade)      OVER (PARTITION BY name) AS avg_grade,  
       COUNT(grade)    OVER (PARTITION BY name) AS count_grade,  
       MIN(grade)      OVER (PARTITION BY name) AS min_grade,  
       MAX(grade)      OVER (PARTITION BY name) AS max_grade  
FROM Student_grades;
```

name	subject	grade	sum_grade	avg_grade	count_grade	min_grade	max_grade
Маша	история	3	15	3,75	4	3	5
Маша	математика	4	15	3,75	4	3	5
Маша	русский	3	15	3,75	4	3	5
Маша	физика	5	15	3,75	4	3	5
Петя	математика	3	16	4	4	3	5
Петя	русский	4	16	4	4	3	5
Петя	физика	5	16	4	4	3	5
Петя	история	4	16	4	4	3	5

Пример 4. Анализ средней зарплаты сотрудников по отделам



Для каждого сотрудника вычисляется средняя зарплата в его отделе, упорядоченная по дате найма.

```
SELECT employee_id AS табель_номер,  
        salary AS зарплата,  
        AVG(salary) OVER (PARTITION BY department_id  
        ORDER BY hire_date) AS средняя_зарплата,  
FROM Employees;
```

табель_номер	зарплата	средняя_зарплата

Ранжирующие функции



В ранжирующей функции под ключевым словом **OVER** обязательным идет указание условия **ORDER BY**, по которому будет происходить сортировка ранжирования:

- **ROW_NUMBER()** - вычисляет *порядковый номер строк внутри окна*, независимо от того, есть ли в строках повторяющиеся значения или нет.
- **RANK()** – определяет *ранг каждой строки внутри окна*. Если есть повторяющиеся значения, функция возвращает одинаковый ранг для таких строчек, пропуская при этом следующий числовой ранг.
- **DENSE_RANK()** – рассчитывает *плотный ранг по окну*, но в случае одинаковых значений не пропускает следующий числовой ранг, а идет последовательно.

NULL значения будут определяться одинаковым рангом.

Пример 5. Ранжирование сотрудников на основе зарплаты



Можно использовать **RANK()** для присвоения рангов сотрудникам на основе их зарплаты, что поможет *выявить высокооплачиваемых сотрудников* в компании.

Этот запрос присваивает ранг каждому сотруднику на основе его зарплаты:

```
SELECT employee_id AS табель_номер,  
        salary AS зарплата,  
        RANK() OVER (ORDER BY salary DESC) AS ранг  
FROM Employees;
```

табель_номер	зарплата	ранг

Пример 5. Ранжирование учеников на основе оценок



```
SELECT name, subject, grade,  
       ROW_NUMBER(grade) OVER (PARTITION BY name) AS row_number,  
       RANK(grade) OVER (PARTITION BY name) AS rank,  
       DENSE_RANK(grade) OVER (PARTITION BY name) AS dense_rank,  
FROM Student_grades;
```

name	subject	grade	row_number	rank	dense_rank
Маша	физика	5	1	1	1
Маша	математика	4	2	2	2
Маша	история	3	3	3	3
Маша	русский	3	4	3	3
Петя	физика	5	1	1	1
Петя	русский	4	2	2	2
Петя	история	4	3	2	2
Петя	математика	3	4	4	3

Пропуск значения «3»

Без пропуска

Пример 6. Ранжирование сотрудников на основе зарплаты



Можно использовать **RANK()** для присвоения рангов сотрудникам на основе их зарплаты, что поможет *выявить высокооплачиваемых сотрудников* в компании.

Этот запрос присваивает ранг каждому сотруднику на основе его зарплаты:

```
SELECT employee_id AS табель_номер,  
       salary AS зарплата,  
       RANK() OVER (ORDER BY salary DESC) AS ранг  
FROM Employees;
```

табель_номер	зарплата	ранг

Функции смещения



Функции смещения:

- получение значения из предыдущей строки в окне **LAG()**
- получение значения из следующей строки **LEAD()**
- получение значения первой строки **FIRST_VALUE()**
- получение значения последней строки **LAST_VALUE()**
- получение значения N-ой строки **NTH_VALUE()**

Требуется *обязательное указание ORDER BY* условия для окна.

Это функции полезны для определения :

- *изменений (отличий)* между текущей и предыдущей строкой (временем),
- *первых и последних в упорядоченном диапазоне*, например, для определения остатков на начало и конец периода.

Пример 7. Анализ изменения успеваемости по четвертям (1)



Нужно с помощью функций смещения для каждой четверти указать предыдущую и последующую четвертные оценки. Таблица Grades_quartal четвертной успеваемости ученика по предметам:

ABC name ↑↓	ABC quartal ↑↓	ABC subject ↑↓	123 grade ↑↓
Петя	1 четверть	физика	4
Петя	2 четверть	физика	3
Петя	3 четверть	физика	4
Петя	4 четверть	физика	5

Пример 7. Анализ изменения успеваемости по четвертям



```
SELECT name, quartal, subject, grade,  
       LAG(grade) OVER (ORDER BY quartal)  
       AS previous_grade,  
       LEAD(grade) OVER (ORDER BY quartal)  
       AS next_grade  
FROM Grades_quartal;
```

name	quartal	subject	grade	previous_grade	next_grade
Петя	1 четверть	физика	4	[NULL]	3
Петя	2 четверть	физика	3	4	4
Петя	3 четверть	физика	4	3	5
Петя	4 четверть	физика	5	4	[NULL]

Пример 8. Анализ изменений продаж



Этот запрос вычисляет разницу в продажах между текущей и предыдущей строкой, упорядоченных по дате продажи:

```
SELECT sale_date                AS дата_продажи,  
       sales_amount             AS объём_продаж,  
       sales_amount – LAG(sales_amount)  
       OVER (ORDER BY sale_date) AS изменение  
FROM Sales;
```

дата_продажи	объём_продаж	изменение

Расширенный пример с несколькими оконными функциями



Получение более детализированной информации о зарплатах сотрудников и их изменениях.

Запрос вычисляет *среднюю зарплату, её ранг и предыдущую зарплату* для каждого *сотрудника отдела*.

```
SELECT employee_id, salary,  
        AVG(salary) OVER (PARTITION BY department_id  
        ORDER BY hire_date) AS avg_salary,  
        RANK() OVER (PARTITION BY department_id  
        ORDER BY salary DESC) AS salary_rank,  
        LAG(salary) OVER (PARTITION BY department_id  
        ORDER BY hire_date) AS prev_salary  
FROM employees;
```

**Терпения и удачи всем, кто
связан с базами данных**

Спасибо за внимание!

Валерий Иванович Артемьев

МГТУ имени Н.Э. Баумана, кафедра ИУ-5

Банк России

Департамент данных, проектов и процессов

Тел.: +7(495) 753-96-25

e-mail: viart@bmstu.ru