# Практическая работа № 5 MySQL. Агрегатные и аналитические функции

Объём учебного времени – 2ч

Методические рекомендации

1. Цель работы: научиться работать с оконными функциями
2. Основные теоретические положения:

**Агрегатные функции** используются в предложении SELECT запроса, в качестве аргументов используются имена полей. В запросах используются следующие агрегатные функции:

* + - **COUNT** подсчёт количества строк, которые выбрал запрос
    - **SUM** арифметическая сумма выбранных значений данного поля
    - **AVG** среднее значение выбранных значений
    - **MAX** наибольшее значение поля
    - **MIN** наименьшее значение поля

**Группировка данных**

**Группировка данных** — это объединение записей в соответствии со значениями некоторого поля.

Для группировки результатов выборки совместно с оператором SELECT используется предложение GROUP BY. Данное предложение должно следовать после предложения WHERE, но перед предложением ORDER BY. После ключевых слов GROUP указывается список полей, включённых в выборку с помощью оператора SELECT, причем нужно обязательно указывать все отбираемые поля (за исключением полей, относящихся к агрегирующим функциям), хотя порядок их перечисления после предложения GROUP BY может не соответствовать порядку списка после слова SELECT.

Синтаксис оператора SELECT с предложением GROUP BY следующий:

**SELЕСТ поле1, поле2.....nameN**

**FROM Таблица**

**WHERЕ условие**

**GROUP BY namel, поле2,.....nameN**

**ORDER BY поле1 {ASC | DESC};**

Предложение HAVING определяет критерии, используемые для удаления определённых групп из вывода.

Получение итоговых данных с помощью оператора ROLLUP

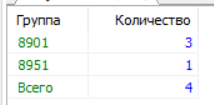
**select** **case**

**when** gr **is** **null** **then** 'Всего'

**else** **cast**(gr **as** **char**(4)) **end** 'Группа', **count**(st) **as** 'Количество'

**from** stud

**group** **by** gr **with** **rollup**;



**Произведение значений столбца**

В отличие от обычных скалярных функций аналитические функции берут аргументом SQL-таблицу, представляющую логический промежуточный результат обработки SQL-оператора, где использовано обращение к такой функции, и возвращают в качестве своего результата обычно тоже SQL-таблицу.

**Аналитические функции**

**Аналитические функции** принимают в качестве аргумента столбец промежуточного результата вычисления SQL-предложения и возвращают тоже столбец. Поэтому местом их использования в SQL-предложении могут быть только фразы ORDER BY и SELECT, выполняющие завершающую обработку логического промежуточного результата.

Предложение PARTITION BY определяет «окно», т.е. набор строк, характеризуемых равенством значений списка выражений, указанного в этом предложении. Если предложение PARTITION BY отсутствует, то агрегатные функции применяются ко всему результирующему набору строк запроса. В отличие от классической группировки, где мы получаем на каждую группу одну строку, которая может содержать агрегатные значения, подсчитанные для каждой такой группы, здесь мы можем добавить агрегат к детализированным (несгруппированным) строкам.

Многие аналитические функции действуют подобно обычным скалярным функциям агрегирования SUM, MAX и прочим, примененным к группам строк, сформированным с помощью GROUP BY. Однако обычные функции агрегирования уменьшают степень детализации, а аналитические функции нет. Поясняющий сравнительный пример:

**SELECT deptno, job, SUM(sal) sum\_sal  
FROM emp  
GROUP BY deptno, job;**

**SELECT ename, deptno, job,  
            SUM(sal) OVER (PARTITION BY deptno, job) sum\_sal      
FROM emp;**

Аналитические функции агрегируют данные порциями (partitions; группами), количество и размер которых можно регулировать специальной синтаксической конструкцией. Ниже она указана на примере агрегирующей функции SUM:

**SUM(выражение 1) OVER([PARTITION BY выражение 2 [, выражение 3 [, …]]])**

Пример использования такой конструкции см. выше.

Если PARTITION BY не указано, то в качестве единственной группы для вычислений будет взят полный набор строк:

**SELECT ename, deptno, job,  
SUM(sal) OVER () sum\_sal  
FROM emp;**

**Упорядочение в границах отдельной группы**

С помощью синтаксической конструкции ORDER BY строки в группах вычислений можно упорядочивать. Синтаксис иллюстрируется на примере агрегирующей функции SUM:

**SUM(выражение 1) OVER([PARTITION …]**

**ORDER BY выражение 2 [,…] [{ASC|DESC}] [{NULLS FIRST|NULLS LAST}])**

**Правила работы ORDER BY - как в обычных SQL-операторах. Пример:**

**SELECT ename, deptno, job,**

**SUM(sal) OVER (PARTITION BY deptno, job ORDER BY hiredate) sum\_sal**

**FROM emp;**

Разницу между ROWS и RANGE (определяющими, как говорится в документации, "физические" и "логические" интервалы-окна) удобно продемонстрировать следующим примером:

**SELECT ename, hiredate, sal,  
SUM(sal)  
OVER (ORDER BY hiredate  
ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW) rows\_sal,  
SUM(sal)  
OVER (ORDER BY hiredate  
RANGE BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW) range\_sal  
FROM emp;**

### Функции FIRST\_VALUE и LAST\_VALUE для интервалов агрегирования

Эти функции позволяют для каждой строки выдать первое значение ее окна и последнее. Пример:

**SELECT ename, hiredate, sal,  
FIRST\_VALUE(sal)  
OVER (ORDER BY hiredate  
ROWS BETWEEN 2 PRECEDING AND CURRENT ROW) first\_rows,  
LAST\_VALUE(sal)  
OVER (ORDER BY hiredate  
ROWS BETWEEN 2 PRECEDING AND CURRENT ROW) last\_rows,  
FIRST\_VALUE(sal)  
OVER (ORDER BY hiredate  
RANGE BETWEEN 2 PRECEDING AND CURRENT ROW) first\_range,  
LAST\_VALUE(sal)  
OVER (ORDER BY hiredate  
RANGE BETWEEN 2 PRECEDING AND CURRENT ROW) last\_range  
FROM emp;**

### Интервалы времени

Для интервалов (окон), упорядоченных внутри по значению ("логическом", RANGE) в случае, если это значение имеет тип "дата", границы интервала можно указывать выражением над датой, а не конкретными значениями из строк. Примеры таких выражений:

**INTERVAL число {YEAR | MONTH | DAY | HOUR | MINUTE | SECOND}**

**NUMTODSINTERVAL(число, '{DAY | HOUR | MINUTE | SECOND}')**

**NUMTOYMINTERVAL(число, '{YEAR | MONTH}')**

Пример выдачи зарплат сотрудников и средних зарплат за последние полгода на момент приема нового сотрудника:

**SELECT ename, hiredate, sal,  
AVG(sal)  
OVER (ORDER BY hiredate  
RANGE BETWEEN INTERVAL '6' MONTH PRECEDING AND CURRENT ROW) avg\_sal  
FROM emp;**

## Виды аналических функций

В качестве базовой в аналитической функции могут быть указаны традиционные для Oracle статистические (агрегатные, то есть обобщающие) функции COUNT, MIN, MAX, SUM, AVG и другие ("стандартные агрегатные функции" по документации). Можно обратить внимание на то, что аналитические функции со статистическими агрегатами разумно обрабатывают NULL:

**SELECT ename, hiredate, sal,  
AVG(sal)  
OVER (ORDER BY hiredate  
RANGE BETWEEN UNBOUNDED PRECEDING AND INTERVAL '1' SECOND PRECEDING) avg\_sal  
FROM emp;**

Ниже приводится полный перечень аналитических функций в версии СУБД 9.2:

|  |  |  |
| --- | --- | --- |
| AVG \* CORR \* COVAR\_POP \* COVAR\_SAMP \* COUNT \* CUME\_DIST DENSE\_RANK FIRST FIRST\_VALUE \* LAG LAST | LAST\_VALUE \* LEAD MAX \* MIN \* NTILE PERCENT\_RANK PERCENTILE\_CONT PERCENTILE\_DISC RANK RATIO\_TO\_REPORT | REGR\_ (вид\_функции\_линейной\_регрессии) \* ROW\_NUMBER STDDEV \* STDDEV\_POP \* STDDEV\_SAMP \* SUM \* VAR\_POP \* VAR\_SAMP \* VARIANCE |

Звездочкой помечены функции, допускающие использование плавающего интервала расчета.

## Функции ранжирования

Функции ранжирования позволяют "раздать" строкам "места" в зависимости от имеющихся в них значениях. Некоторые примеры:

SELECT ename, sal,  
**ROW\_NUMBER () OVER (ORDER BY sal DESC)** AS salbacknumber,  
**ROW\_NUMBER () OVER (ORDER BY sal)** AS salnumber,  
**RANK() OVER (ORDER BY sal)**AS salrank,  
**DENSE\_RANK() OVER (ORDER BY sal)**AS saldenserank  
FROM emp;

(раздать сотрудникам места в порядке убывания/возрастания зарплат)

## 

## Функции подсчета долей

Функции подсчета долей позволяют одной SQL-операцией получить для каждой строки ее "вес" в таблице в соответствии с ее значениями. Некоторые примеры:

**SELECT ename, sal, RATIO\_TO\_REPORT(sal) OVER () AS salshare FROM emp;**

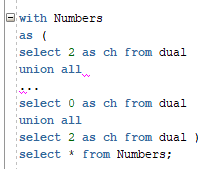
(доли сотрудников в общей сумме зарплат)

Пример выдачи доли сотрудников с меньшей или равной зарплатой, чем у "текущего":

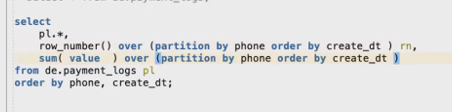
**SELECT job, ename, sal,  
CUME\_DIST() OVER (PARTITION BY job ORDER BY sal) AS cume\_dist  
FROM emp;**

1. Практические указания:

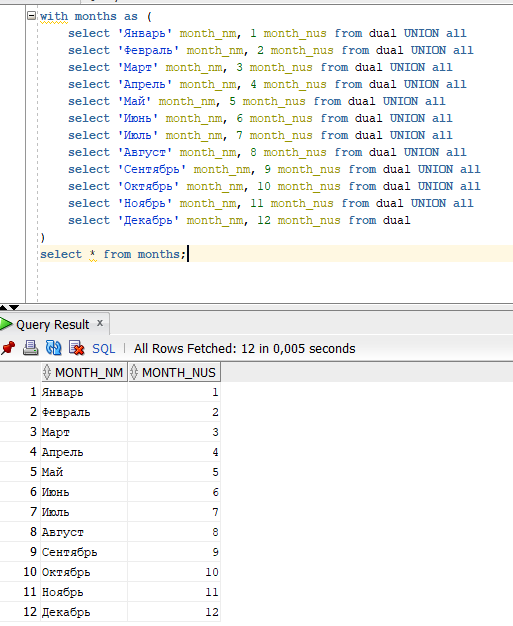
* 1. **Открытие базы данных**
     1. Откройте программу **sqldeveloper**. Выберите ваше подключение.
     2. Перейдите в **SqlWorksheet**
  2. **Использование агрегатных функций**
     1. Найти средний год рождения всех пользователей библиотеки;
     2. Подсчитайте общее количество строк в таблице **Avtor**.
     3. Найдите минимальную цену книги для каждого раздела.
     4. Напишите запрос для нахождения максимальной цены по каждому разделу и максимальной итоговой цены.
     5. Напишите запрос, который по каждому автору находит количество книг, если данное количество будет от 10 до 15;
     6. Найдите среднюю цену тех книг, год выпуска у которых больше 2000.
     7. Найдите по каждому пользователю количество взятых книг в период от начала года до текущей даты.
     8. Найдём количество положительных и количество отрицательных чисел, и количество числе равных 0 (возьмем 10 чисел)
        + Построим набор данных по примеру



* + - * Добавим агрегатные функции подсчета сумм (положительных, отрицательных, равных 0), при этом воспользуйтесь функцией case, в которой сравнивайте число с 0. В случае, если условие выполняется возвращайте 1, иначе 0.
  1. **Работа с оконными функциями**
     1. Расставьте порядковый номер выдачи книг для каждого дня
        + Воспользуйтесь функцией **Row\_number()** с окном по году издания с сортировкой по номеру выдачи
        + При выполнении запроса проанализируйте пример ниже



* + 1. Для каждого месяца 2020 года выведите количество выданных книг
       - Получим список месяцев



* + - * Добавим в sql-запрос выборку полей month\_nm и количества выдач книг, соединим данные из построенной таблицы months с данными таблицы bilet по номеру месяца (из таблицы месяцев это поле month\_nus), а для получения значения номера месяца из таблицы bilet – воспользуйтесь функцией extract
      * Добавьте условие выборки – год выдачи книг 2020 (тоже функция extract для получения года из даты)
      * Добавьте группировку по полю месяц и номер месяца
      * Добавьте сортировку по полю номер месяца
    1. Напишите аналогичный запрос с применением оконной функции
       - Укажите поле для выборки в select как количество строк id\_bilet
       - Укажите окно over как формирование окон для каждого номера месяца с сортировкой по дате выдачи
       - Уберите группировку и сортировку прошлого запроса
       - Сравните полученные результаты
  1. **Выполнение самостоятельной работы**
     1. Найдем в каждом месяце первый день выдачи книг автора Толстой
     2. Найдем разницу между ценой книги и средней ценой книг автора
     3. Выведем даты выдачи книг вместе с предыдущей и следующей строками

**SELECT code,**

**LAG(code) OVER(ORDER BY code) prev\_code,**

**LEAD(code) OVER(ORDER BY code) next\_code**

**FROM printer;**

1. Содержание отчёта по практической работе
   1. Цель работы.
   2. Результаты выполнения работы;
   3. Выполненные работы в электронном виде представляются на сетевом диске H (X)
2. Контрольные вопросы
   1. Агрегатные функции. Использование.
   2. Запросы с использованием оператора group by.
   3. Оконные функции
3. Список рекомендуемой литературы:
   1. Агальцов В.П. Базы данных. Распределённые и удалённые базы данных. Учебник М.: ИД «Форум»: ИНФРА-М, 2011. - 272с.
   2. Дунаев В.В. Базы данных. Язык SQL. СПб.: БХВ-Петербург-2006, 288с.
   3. Дюбуа П. MySQL.: учеб.пособие.- М.-:Вильямс,2007.- 1168с.
   4. Кузнецов М.В., MySQL 5, СПб.: БХВ-Петербург, 2010, 1024с.
   5. Петров В.Н. Избачков Ю.С., Информационные системы. СПб.: Питер, 2006, 656с.
   6. Шелдон Р. MySQL.: учеб.пособие.- М.-:Вильямс,2007.- 880с.