ЛАБОРАТОРНАЯ РАБОТА №14

Тема: Аналитические функции в SQL запросах.

Цель: Пробрести навыки составления SQL запросов использующих аналитические функции.

В данной лабораторной работе используются база данных HR, созданная в рамках первой лабораторной работы. Также используется таблица TB_ELEKTROSTAL_2018, которая создана и наполнена данными в процессе выполнения лаб. работы №.1.

Дополнительно используется удаленная база данных проекта stackoverflow.com. Структура этой базы данных описана в методических указаниях к выполнению лабораторной работы № 3. Напомним, адрес ресурса http://data.stackexchange.com/stackoverflow/query/new.

Для решения большинства задач используется стандартный SQL. Однако, можно выделить ряд типов запросов, которые довольно трудно решить в рамках стандартного SQL.

Сфера применения аналитических функций.

Ряд запросов, которые сложно сформулировать на обычном языке SQL, весьма типичны. С помощью аналитических функций подобные операции не только проще записываются, но и быстрее выполняются по сравнению с использованием чистого языка SQL.

- Подсчет промежуточной суммы. Показать суммарную зарплату сотрудников отдела построчно, чтобы в каждой строке выдавалась сумма зарплат всех сотрудников вплоть до указанного.
- Подсчет процентов в группе. Показать, какой процент от общей зарплаты по отделу составляет зарплата каждого сотрудника. Берем его зарплату и делим на сумму зарплат по отделу.
- Запросы первых N. Найти N сотрудников с наибольшими зарплатами или N наиболее продаваемых товаров по регионам.

- Подсчет скользящего среднего. Получить среднее значение по текущей и предыдущим N строкам.
- Выполнение ранжирующих запросов. Показать относительный ранг зарплаты сотрудника среди других сотрудников того же отдела.

Классификация видов аналитических функций.

Аналитические функции могут быть следующих видов:

(a) функции ранжирования

(b) оконные функции (функции для плавающего интервала)

(c) функции для создания отчетов (в частности, функции подсчета долей)

(d) статистические функции **LAG/LEAD** с запаздывающим/опережающим аргументом

(e) статистические функции (линейная регрессия и т. д.)

Место аналитических функций в SQL-предложении

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.

Сравнение с обычными агрегатными функциями

Многие аналитические функции действуют подобно обычным агрегатным функциям SUM, MAX и прочим, примененным к группам строк, сформированным с помощью GROUP BY. Однако обычные агрегатные функции уменьшают степень детализации, а аналитические функции нет. В качестве поясняющего примера сравним результаты выполнения двух следующих запросов к таблице employees, содержащейся в схеме HR:

```
SELECT department_id
     , job_id
     , SUM(salary) sum_sal
     FROM employees
     GROUP BY department_id
            , job_id;
```

```
SELECT first_name||' '||last_name
     , department_id
     , job_id
     , SUM(salary) OVER (PARTITION BY department_id,
job_id) sum_sal
       FROM employees;
```

Ниже в примере продемонстрирован запрос к БД HR с использованием аналитической функции SUM. Результатом запроса является список всех сотрудников, в котором отображена сумма заработной платы каждого сотрудника и нарастающий итог по зарплате для всех сотрудников.

```
select last_name
     , first_name, salary
     , SUM (salary)
      OVER (ORDER BY last_name
                   , first_name) running_total
      from employees
      order by last_name
             , first_name;
```

Результат этого запроса получен с помощью выражения:

```
SUM (salary)
  OVER (ORDER BY last_name, first_name) running_total
```

Структура аналитических функций.

Синтаксис аналитического предложения можно представить в общем виде:

```
FUNCTION_NAME( column | expression,column |
expression,... )
OVER
( Order-by-Clause )
```

В рассмотренном выше примере имя функции **SUM**. Аргументом функции **SUM** является столбец **salary** (хотя также может быть и выражение). Предложение **OVER** говорит то, что это аналитическая функция (без него это была бы обычная агрегатная функция). Предложение **ORDER BY** обозначает часть данных «над» которыми будет выполняться аналитическая функция.

Отметим, что для достижения результата, полученного в примере можно использовать скалярные подзапросы. Однако такой запрос будет выполняться значительно медленнее и будет менее читабельным.

Во втором примере рассмотри запрос на получение заработной платы с нарастающим итогом, построчно, для каждого отдела.

```
select last_name
     , first_name
     , department_id
     , salary
     , SUM (salary) OVER (PARTITION BY department_id
ORDER BY last_name, first_name) department_total
      from employees
    order by department_id
        , last_name
        , first_name;
```

Представленный запрос суммирует значение заработной платы каждого сотрудника построчно в рамках департамента. Предложение **PARTITION** говорит о том, что аналитическая функция применяется к каждой группе отделов (или разделов — **partition**) независимо. Если посмотреть на результат запроса, то можно заметить, что нарастающий итог сбрасывается после изменения департамента с 10 до 20, и снова с 20 до 30, и с 30 до записи сотрудника, который не прикреплен ни к одному департаменту и своего рода является отдельной группой.

Теперь синтаксис аналитической функции можно представить следующем общем виде:

```
FUNCTION_NAME( argument,argument,… )
OVER
( Partition-Clause Order-by-Clause )
```

Разбиение данных на группы для вычислений

Аналитические функции агрегируют данные порциями (partitions; группами), количество и размер которых можно регулировать специальной синтаксической конструкцией. Ниже она указана на примере агрегатной 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 first_name||' `||last_name
     , department_id
     , job_id
     , SUM(salary) OVER () sum_sal
   FROM employees;
```

Сортировка в аналитических функциях

Рассмотренные ранее запросы сортируют возвращаемые строки по фамилии и имени сотрудника. Далее представлен запрос, который использует несколько иной критерий сортировки для вычисления аналитической функции.

Вычисление каждой строки на основе значения заработной платы.

```
 select last_name
      , first_name
      , department_id
      , salary
      , SUM (salary) (PARTITION BY department_id
                      ORDER BY salary) department_total
      from employees
      order by department_id
             , salary
             , last_name
             , first_name;
```

Аналитическая функция в этом случае вычисляет суммарные значения департамента на основе заработной платы, в порядке возрастания для каждого раздела, включая значение заработной платы равное NULL, которое при сортировке окажется последним. Таким образом, запись, где **last_name = Dovichi Lori** — единственная запись с зарплатой **NULL**, которая имеет значение **DEPARTMENT_TOTAL** такому же значению как и у сотрудника, который имеет самую высокую зарплату в департаменте.

Предложение **ORDER BY** в аналитической функции работает независимо от предложения **ORDER BY** общего запроса, который содержит аналитическую функцию. Кроме этого, между ними существует взаимосвязь, если они используют одни и те же столбцы или выражения в том же порядке.

Упорядочение в границах отдельной группы

С помощью синтаксической конструкции **ORDER BY** строки в группах вычислений можно упорядочивать. Синтаксис иллюстрируется на примере агрегатной 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 first_name||' `||last_name
     , department_id
     , job_id
     , SUM(salary) OVER (PARTITION BY deptno, job ORDER
BY hire_date) sum_sal
FROM employees;
```

Виды аналитических функций

В качестве базовой в аналитической функции могут быть указаны традиционные агрегатные функции COUNT, MIN, MAX, SUM, AVG и другие ("стандартные агрегатные функции" по документации). Следует отметить, что аналитические функции корректно обрабатывают значения NULL:

```
SELECT first_name||' `||last_name
     , hire_date
     , salary
     , AVG(salary)OVER (ORDER BY hire_date
 RANGE BETWEEN UNBOUNDED PRECEDING AND INTERVAL '1'
SECOND PRECEDING) avg_sal
     FROM employees;
```

Ниже приводится перечень аналитических функций.

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.

Некоторые из этих функций рассматриваются ниже.

Функции ранжирования

Функции ранжирования позволяют "раздать" строкам "места" в зависимости от имеющихся в них значений. Например:

```
SELECT first_name||' `||last_name
     , salary
     , ROW_NUMBER () OVER (ORDER BY salary DESC) AS
salbacknumber
     , ROW_NUMBER () OVER (ORDER BY salary) AS
salnumber
     , RANK() OVER (ORDER BY salary) AS salrank
     , DENSE_RANK() OVER (ORDER BY salary) AS
saldenserank
    FROM employees;
```

(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 first_name||' '||last_name
     , salary
     , RATIO_TO_REPORT(salary) OVER () AS salshare
    FROM employees;
```

(доли сотрудников в общей сумме зарплат)

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

```
SELECT job_id
       , first_name||' '||last_name
   , salary
   ,CUME_DIST() OVER (PARTITION BY job_id ORDER BY
salary) AS cume_dist FROM employees;
```

(видно, что три четверти клерков имеют зарплату, меньше чем ADAMS).

Проранжировать эту выдачу по доле сотрудников в группе можно функцией PERCENT_RANK:

```
SELECT job_id
     , first_name||' '||last_name
     , salary
     , CUME_DIST() OVER (PARTITION BY job_id ORDER BY
salary) AS cume_dist
    , PERCENT_RANK() OVER (PARTITION BY job_id ORDER BY
salary) AS pct_rank
    FROM employees;
```

Процентный ранг отсчитывается от 0 и изменяется до 1.

Аналитическая функция	Назначение
AVG([DISTINCT | ALL] выражение)	Используется для вычисления среднего значения выражения в пределах группы и окна. Для поиска среднего после удаления дублирующихся значений можно указывать ключевое слово DISTINCT

CORR (выражение, выражение)	Выдает коэффициент корреляции для пары выражений, возвращающих числовые значения. Это сокращение для выражения: COVAR_POP<выражение1, выражение2) / STDDEV_POP(выражение!.) * STDDEV_POP(выражение2)). В статистическом смысле, корреляция — это степень связи между переменными. Связь между переменными означает, что значение одной переменной можно в определенной степени предсказать по значению другой. Коэффициент корреляции представляет степень корреляции в виде числа в диапазоне от -1 (высокая обратная корреляция) до 1 (высокая корреляция). Значение 0 соответствует отсутствию корреляции
COUNT([DISTINCT][*] [выражение])	Эта функция считает строки в группах. Если указать * или любую константу, кроме NULL, функция count будет считать все строки. Если указать выражение, функция count будет считать строки, для которых выражение имеет значение не NULL. Можно задавать модификатор DISTINCT, чтобы считать строки в группах после удаления дублирующихся строк
COVAR_POP(выражение,выражение)	Возвращает ковариацию генеральной совокупности (population covariance) пары выражений с числовыми значениями.
COVAR_SAMP (выражение, выражение)	Возвращает выборочную ковариацию (sample covariance) пары выражений с числовыми значениями.
CUME_DIST	Вычисляет относительную позицию строки в группе. Функция CUME_DIST всегда возвращает число большее 0 и меньше или равное 1. Это число представляет "позицию" строки в группе из N строк. В группе из трех строк, например, возвращаются следующие значения кумулятивного распределения: 1/3, 2/3 и 3/3
DENSE_RANK	Эта функция вычисляет относительный ранг каждой возвращаемой запросом строки по отношению к другим строкам, основываясь на значениях выражений в конструкции ORDER BY. Данные в группе сортируются в

	соответствии с конструкцией ORDER BY, а затем каждой строке поочередно присваивается числовой ранг, начиная с 1. Ранг увеличивается при каждом изменении значений выражений, входящих в конструкцию ORDER BY. Строки с одинаковыми значениями получают один и тот же ранг (при этом сравнении значения NULL считаются одинаковыми). Возвращаемый этой функцией "плотный" ранг дает ранговые значения без промежутков.
FIRST_VALUE	Возвращает первое значение в группе
LAG(выражение, <смещение>,<стандартное значение>)	Функция LAG дает доступ к другим строкам результирующего множества, избавляя от необходимости выполнять самосоединения. Она позволяет работать с курсором как с массивом. Можно ссылаться на строки, предшествующие текущей строке в группе. О том, как обращаться к следующим строкам в группе, см. в описании функции LEAD. Смещение — это положительное целое число со стандартным значением 1 (предыдущая строка). Стандартное значение возвращается, если индекс выходит за пределы окна (для первой строки группы будет возвращено стандартное значение).
LAST_VALUE	Возвращает последнее значение в группе.
LEAD(выражение, <смещение>, <стандартное значение>)	Функция LEAD противоположна функции LAG. Если функция LAG дает доступ к предшествующим строкам группы, то функция LEAD позволяет обращаться к строкам, следующим за текущей. Смещение — это положительное целое число со стандартным значением 1 (следующая строка). Стандартное значение возвращается, если индекс выходит за пределы окна (для последней строки группы будет возвращено стандартное значение)
MAX(выражение)	Находит максимальное значение выражения в пределах окна в группе.
MIN(выражение)	Находит минимальное значение выражения в пределах окна в группе.
RANK()	Нумерует строки, которые имеют одинаковые значения в столбцах, по которым выполняется упорядочивание. Такие строки получают

	одинаковые номера (ранги).
ROW_NUMBER()	нумерует строки, возвращаемые запросом.

```
select first_name
     , salary
     , row_number() over (order by salary desc)
            as row_number_desc
     , row_number() over (order by salary)
            as row_number_asc
     , rank() over (order by salary) as rank
     , dense_rank() over (order by salary)
            as dense_rank
   from employees;
```

Порядок выполнения работы

Составить следующие запросы

- Составить запрос к БД HR с использованием необходимой аналитической функции, который выводит список работников ранжированных по размеру оклада (salary) в рамках каждого департамента.
- Составить запрос к БД HR с использованием аналитической функции FIRST_VALUE. Запрос выводит 5 колонок. Первая колонка имя работника (поля first_name и last_name, разделенные пробелом). Вторая колонка – department_id. Третья колонка – hire_date. Четвертая колонка – оклад работника (поле salary). Пятая колонка – оклад сотрудника первым принятого в департамент (сортировка по полю hire_date).
- Составить запрос к БД HR с использованием аналитической функции row_number(). Запрос выводит 3 колонки. Первая колонка – departament_id. Вторая - имя работника (поля first_name и last_name, разделенные пробелом). Третья колонка – порядковый номер сотрудника в порядке увеличения оклада в пределах департамента. Для каждого департамента вывести пять сотрудников – первых в порядке сортировки во возрастанию оклада.

- Составить запрос к TB_ELEKTROSTAL_2018 с использованием аналитической функции row_number(). Запрос выводит 3 колонки. Первая колонка – название улицы. Вторая - ФИО абонента. Третья колонка – порядковый номер абонента в алфавитном порядке в пределах улицы. Для каждой улицы вывести семь абонентов – первых по алфавиту. Примечание: учитывать только улицы, проспекты и переулки.

Содержание отчета:

1. Тема, цель лабораторной работы.
2. Примеры выполнения запросов к базе данных.
3. Составленные согласно заданию запросы и скриншоты полученных результатов.
5. Выводы.

Контрольные вопросы:

1. Перечислите основные типы аналитических функций.
2. Перечислить причины (цели), объясняющие необходимость наличия аналитических функций.
3. Какие виды аналитических функций существуют?
4. Где записываются аналитические функции в SQL-операторах?
5. Какие конкретные возможности предоставляют аналитические функции по сравнению с обычными агрегатными функциями?

Файл:	Лабораторная работа 14
Каталог:	C:\Users\sss\Documents
Шаблон:	C:\Users\sss\AppData\Roaming\Microsoft\Шаблоны\Normal.dotm
Заголовок:	
Содержание:	
Автор:	sss
Ключевые слова:	
Заметки:	
Дата создания:	19.11.2018 11:25:00
Число сохранений:	80
Дата сохранения:	19.11.2018 21:18:00
Сохранил:	sss
Полное время правки:	274 мин.
Дата печати:	19.11.2018 21:19:00
При последней печати	
страниц:	12
слов:	2 523 (прибл.)
знаков:	14 383 (прибл.)