ЛАБОРАТОРНАЯ РАБОТА №19

Тема: Агрегирование строк запроса

Цель: Пробрести навыки создания и использования запросов на языке SQL с использованием средств агрегирования строк (инструментов CUBE и ROLLUP).

Теоретические сведения

В лабораторной работе рассматриваются вопросы агрегирования данных с использованием агрегатных функций и предложения GROUP BY. Для выполнения лабораторной работы необходимо выполнить скрипт, помещенный в приложении к лабораторной работе, в облачной системе APEX.ORACLE.COM.Для проверки корректности подготовки данных к лабораторной работе необходимо выполнить следующий запрос:

```
select * from bricks;
```

Агрегатные функции комбинируют (соединяют) множество строк в одну. Запрос возвращает одну строку для каждой группы. В случае, если в запросе не используется конструкция group by, получим одну группу. И запрос вернет ровно одну строку.

Например, count() вернет количество строк в источнике данных, обрабатываемых запросом. Таким образом, запрос вернет одну строку – количество строк в таблице bricks:

```
select count(*) from bricks;
```

В данном примере функция Count использует параметр *. Это обозначает подсчет полного количества строк в источнике данных. В функцию count() можно передать выражение (колонку) в качестве параметра. В этом случае будет возвращено количество не пустых (not null) строк для заданного параметра функции. Например, следующий запрос вернет количество строк, в которых поле **colour** не пустое (not null):

```
select count(colour) from bricks;
```

Все прочие (не count) агрегатные функции в качестве параметра используют выражение. СУБД Oracle включает много статистических агрегатных функций. К наиболее часто используемым функциям используют:

- Sum: суммирует все значения в группе для заданного выражения
- Min: возвращает наименьшее значение указанного выражения в группе
- Max: возвращает наибольшее значение указанного выражения в группе
- Avg: среднее арифметическое значение указанного выражения в группе
- Stddev: стандартное отклонение (статистическая функция)
- Median: среднее значение в наборе данных
- Variance: вариация значений в группе (статистическая функция)
- Stats_mode: наиболее часто встречающееся значение в группе

Следующий пример демонстрирует использование этих функций:

```
select sum ( weight )
     , min ( weight )
     , max ( weight )
     , avg ( weight )
     , stddev ( weight )
     , median ( weight )
     , variance ( weight )
     , stats_mode ( weight )
        from bricks;
```

Существует много прочих специализированных агрегатных функций.

Ключевые слова Distinct и All

По умолчанию агрегатные функции используют каждое входное значение. Многие функции позволяют обрабатывать только уникальные значения. Для этой цели используют ключевое слово **distinct**.

Например, требуется подсчитать количество различных значений в поле colour column. Для этого используем значение "distinct colour" в качестве параметра функции count(). Таких цветов три: red, green, blue. Таким образом, будет возвращено значение 3.

```
select count ( distinct colour )
number_of_different_colours from bricks;
```

Для явного указания обрабатывать все значения следует использовать ключевое слово **all**. Также можно использовать ключевое слово **unique** в качестве синонима ключевого слова **distinct**:

```
select count ( all colour ) total_number_of_rows
     , count ( distinct colour )
number_of_different_colours
     , count ( unique colour ) number_of_unique_colours
from bricks;
```

Ключевое слово distinct можно использовать в большинстве статистических функций, таких как sum и avg. Таблица brick содержит значений весов (weights) 1, 1, 1, 2, 2, и 3. Видно, что различные значения 1, 2 и 3. Таким образом, сумма весов будет 10, а среднее значение веса - 1.66. Но, с использованием ключевого слова distinct сумма весов будет 6 а среднее значение - 2:

```
select sum ( weight ) total_weight
     , sum ( distinct weight ) sum_of_unique_weights
     , avg ( weight ) overall_mean
     , avg ( distinct weight ) mean_of_unique_weights
       from bricks;
```

Не все агрегатные функции поддерживают использование ключевого слова distinct.

Использование группировки

Для получения итоговых данных по группам можно разделить результирующее множество строк на такие группы, и для каждой группы подсчитать агрегатную функцию отдельно. Для разбиения множества строк результата на группы используется ключевое слово **group by**. В случае использования этого ключевого слова возвращается одна строка для каждой группы. При разбиении множества строк на группы используется значение (значения) в указанной колонке (колонках).

Например, следующий запрос вернет количество строк для каждого цвета (**colour**):

```
select colour
```

```
, count (*)
  from bricks
  group by colour;
```

Разработчик имеет возможность не указывать колонки, по которым выполняется группировка в списке полей оператора **select**. Указанный ниже запрос также как и предыдущий разбивает множество возвращаемых строк на подмножество цветов. Обратите внимание, что здесь не используется поле группировки **colour** в списке оператора **select**:

```
select count (*) from bricks
     group by colour;
```

Такой запрос может выглядеть не совсем понятным, поэтому рекомендуется включать поля группировки в список полей оператора **select.**

Обратное утверждение будет неверным. Все единичные поля (не агрегатные функции) в списке полей оператора **select** должны быть использованы в операторе группирования **group by**.

В представленном ниже примере запроса **select** содержится ошибка. Причиной ошибки является использование единичного поля **shape** в списке полей. Однако, это поле отсутствует в операторе группирования **group by**, что и приводит к ошибке :

```
select colour
     , shape
     , count (*)
       from bricks
       group by colour;
```

Имеется возможность группировать по нескольким колонкам. И для исправления ошибки необходимо группировать по нескольким колонками: как по колонке **shape**, так и по колонке **weight**:

```
select shape
     , weight
     , count (*)
       from bricks
       group by shape, weight;
```

Фильтрация значений агрегатных функций

Алгоритм выполнения запрос предусматривает процесс группировки после окончания процесса фильтрации, условие которой задается ключевым словом **where**. Например, следующий запрос исключит из результирующего набора строки, в которых значение в поле **weight <= 1**. Эти строки выпадут из процесса подсчета количества:

```
select colour
     , count (*)
       from bricks
       where weight > 1
       group by colour;
```

В разделе **where** можно фильтровать только единичные поля (не агрегатные функции). При попытке в этом месте использовать агрегатные функции получим ошибку.

Попытка сформулировать запрос на вывод групп, в которых количество записей с заданным цветов более одной, в следующей форме выдаст ошибку:

```
select colour
     , count (*)
       from bricks
       where count (*) > 1
       group by colour;
```

Для задания условий фильтрации на агрегатные функции нужно использовать оператор **having**. Он может располагаться как до оператора **group by** так и после него. Следующий пример решает поставленную выше задачу (приводится два варианта запроса):

```
select colour
     , count (*)
       from bricks
       group by colour
       having count (*) > 1;

select colour
     , count (*)
       from bricks
       having count (*) > 1
       group by colour;
```

Разработчик имеет возможность использовать различные функции в списке полей оператора **select** и фильтре **having**. Например, следующий запрос выводит список цветов для кирпичей, у которых вес в группе больше одной единице:

```
select colour
     , count (*)
       from bricks
       having sum ( weight ) > 1
       group by colour;
```

Получение промежуточных итогов (Subtotals)

Поддерживает возможность получать промежуточные итоги с использованием группировки. Для решения этой задачи предназначены функции **rollup** и **cube.**

Rollup

Функция **Rollup** генерирует промежуточные итоги по колонкам. При этом предусматривается порядок справа – налево. Рассмотрим пример записи **rollup**:

```
rollup ( colour, shape )
```

В данном пример **rollup** вычислит:

- Итоги для каждой пары (**colour**, **shape**)
- Итоги для каждой колонки **colour**
- Полные итоги (grand total)

Группы , имеющие значения в каждой колонке являются регулярными (обычными) строками. Промежуточные итоги представляют суперагрегаты (supperaggregate rows). Для таких суперагрегатов СУБД возвращает **null** для колонок группирования. Так колонка промежуточных итогов полю **colour** содержит значение **null** для поля **shape**. Заключительная итоговая строка будет содержат **null** для обоих полей **colour** и **shape**:

```
select colour
     , shape
     , count (*)
       from bricks
       group by rollup ( colour, shape );
```

Разработчик имеет возможность использовать функцию **rollup** с колонкой не входящей в оператор **group by**. В этом случае "grand total" вычисляется для колонки не входящей в **rollup**. В примере вычисляется итоги для каждой пары полей (**colour**, **shape**) и количество строк для каждого цвета:

```
select colour
     , shape
     , count (*)
       from bricks
       group by colour
              , rollup ( shape );
```

Функция вычисляет N+1 группировку. Здесь N количество выражений в **rollup**. Таким образом, **rollup** с тремя колонками вернет 3+1 = 4 группы.

Cube

Функция **Cube** вычисляет промежуточный итоги для каждой комбинации колонок, указанной в параметрах функции. Например: для выражения

```
cube ( colour, shape )
```

получим группировку для:

- Каждой пары значений (**colour**, **shape**)
- Каждого значения поля **colour**
- Каждого значения поля **shape**
- Всех строк в таблице

```
select colour
     , shape
     , count (*)
       from bricks
       group by cube ( colour, shape );
```

Функция **Cube** вычисляет 2^N группировки. Здесь N количество выражений в параметрах функции. Например, cube с тремя колонками сгенерирует 2^3 = 8 группировок.

Порядок выполнения работы

1. Составьте запрос, который возвращает:

- Количество различных форм
- Стандартное отклонение (**stddev**) уникальных весов

```
select /* TODO */ number_of_shapes
     , /* TODO */ distinct_weight_stddev
       from bricks;
```

Результат выполнения запроса должен быть следующим:

```
NUMBER_OF_SHAPES   DISTINCT_WEIGHT_STDDEV
               3                        1
```

2. Составьте запрос, который возвращает суммарный вес (weight) каждой формы (**shape**) в таблице **bricks**:

```
select shape
     , /* TODO */ shape_weight
       from bricks
       /* TODO */;
```

Результат выполнения запроса должен быть следующим:

```
SHAPE    SHAPE_WEIGHT
cube                5
cuboid              1
pyramid             4
```

3. Составьте запрос для вывода форм (**shapes**), для которых суммарный вес меньше 4.

```
select shape
     , sum ( weight )
       from bricks
       group by /* TODO */;
```

Результат выполнения запроса должен быть следующим:

```
SHAPE    SUM(WEIGHT)
cuboid
```

4. Составьте запрос для группирования записей таблицы **TB_ELEKTROSTAL_2018** по типам и названиям улиц. Для каждой улицы вывести количество абонентов. Вывести только те улицы, на которых живет не менее 1000 абонентов. Использовать функции **rollup** и **cube** в этом запросе.

5. Составьте запрос для группирования домов из таблицы **TB_ELEKTROSTAL_2018** по типам и названиям улиц. Для каждой улицы вывести количество домов. Учитывать только дома, в которые состоят не менее чем из 40 квартир. Использовать функции **rollup** и **cube** в этом запросе.
6. Составьте запрос для группирования работников из таблицы **employees** по странам, должностям и отделам. Для каждой записи вывести количество работников, сумму заработной платы с учетом премии. Составить вариант запроса с использованием функции **rollup**. Составить вариант запроса с использованием функции **cube.**

Содержание отчета.

1. Название работы
2. Цель работы
3. Листинги запросов
4. Скриншоты результатов выполнения запросов.
5. Выводы

Контрольные вопросы

1. Объяснить назначение операции группировки.
2. Объяснить особенности сочетания единичных полей и агрегатных функций в запрос на группировку.
3. Назначение функции **rollup**. Привести примеры.
4. Назначение функции **cube**. Привести примеры.
5. Проанализировать последний вопрос в задании к лабораторной работе и выяснить, от каких параметров больше всего зависит размер заработной платы работников.

Приложение

```
create table bricks ( colour varchar2(10)
         , shape varchar2(10)
         , weight integer );
insert into bricks values ( 'red', 'cube', 1 );
insert into bricks values ( 'red', 'pyramid', 2 );
insert into bricks values ( 'red', 'cuboid', 1 );
insert into bricks values ( 'blue', 'cube', 1 );
insert into bricks values ( 'blue', 'pyramid', 2 );
insert into bricks values ( 'green', 'cube', 3 );
```