SQL: Аналитические функции

Источник: [docs .google](http://www.interface.ru/iservices/goto.asp?Url=enc:rn%7Ej%40%255%5EuYy%24meu%5Dv%5F4Yug5%3EuYEcn7n%5En%2E%3C%2ECre%2Em%5Dx%2Etji%29)

Аналитические функции на примере [**Oracle**](http://www.itshop.ru/Oracle/l2t1f2c0sc0), функции LAG.  
Прочитав этот материал вы поймете, как работают аналитические функции в [**Oracle**](http://www.itshop.ru/Oracle/l2t1f2c0sc0). Я рассмотрю только одну функцию, но принцип действия у них один.

**Постановка задачи**

Будем рассматривать на простом примере:  
Есть таблица APP\_STAT, которая содержит данные о совершаемой операции. Каждая строка содержит время, описание операции и имя пользователя, который эту операцию выполнил. Своеобразный log.  
Нам нужно узнать, сколько пользователь "думает" между кликами и вычислить среднее время, которое **каждый** пользователь тратит на "обдумывание".  
  
Создадим таблицу:

**create table** *"APP\_STAT"*  
(  
*"T"* date not null,  
*"PAGE"* varchar2(50 char) not null,  
*"USER\_NAME"* varchar2(50 char) not null  
);

Соответственно, T - время, в которое выполнялась операция; PAGE - название страницы/операции, которую совершил пользователь; USER\_NAME - имя пользователя.

Заполним ее данными:

**alter session** set **NLS\_DATE\_FORMAT**='*DD-MM-YY HH24:MI:SS*';  
insert into **APP\_STAT** (T, PAGE, USER\_NAME)  
select *'22-06-08 10:30:00','login','bart'* from dual union  
select *'22-06-08 10:31:00','search 1','bart'* from dual union  
select *'22-06-08 10:31:20','search 1','bart'* from dual union  
select *'22-06-08 10:31:20','login','homer'* from dual union  
select *'22-06-08 10:31:30','search 1','bart'* from dual union  
select *'22-06-08 10:32:00','new doc','bart'* from dual union  
select *'22-06-08 10:32:10','list doc','homer'* from dual union  
select *'22-06-08 10:32:20','view doc','homer'* from dual union  
select *'22-06-08 10:33:40','list doc','homer'* from dual union  
select *'22-06-08 10:34:00','view doc','homer'* from dual union  
select *'22-06-08 10:36:20','save doc','bart'* from dual union  
select *'22-06-08 10:36:30','delete doc','homer'* from dual union  
select *'22-06-08 10:36:30','list doc','bart*' from dual union  
select *'22-06-08 10:37:00','logout','homer'* from dual union  
select *'22-06-08 10:37:00','view doc','bart'* from dual union  
select *'22-06-08 10:37:10','edit doc','bart'* from dual union  
select *'22-06-08 10:38:30','save doc','bart'* from dual union  
select *'22-06-08 10:38:45','logout','bart'* from dual;

Посмотрим, что мы сделали:

select \* from **APP\_STAT**  
order by T asc

|  |  |  |
| --- | --- | --- |
| **T** | **PAGE** | **USER\_NAME** |
| 22-06-08 10:30:00 | login | bart |
| 22-06-08 10:31:00 | search 1 | bart |
| 22-06-08 10:31:20 | login | homer |
| 22-06-08 10:31:20 | search 1 | bart |
| 22-06-08 10:31:30 | search 1 | bart |
| 22-06-08 10:32:00 | new doc | bart |
| 22-06-08 10:32:10 | list doc | homer |
| 22-06-08 10:32:20 | view doc | homer |
| 22-06-08 10:33:40 | list doc | homer |
| 22-06-08 10:34:00 | view doc | homer |
| 22-06-08 10:36:20 | save doc | bart |
| 22-06-08 10:36:30 | delete doc | homer |
| 22-06-08 10:36:30 | list doc | bart |
| 22-06-08 10:37:00 | logout | homer |
| 22-06-08 10:37:00 | view doc | bart |
| 22-06-08 10:37:10 | edit doc | bart |
| 22-06-08 10:38:30 | save doc | bart |
| 22-06-08 10:38:45 | logout | bart |

**Решение**

Итак, необходимо узнать, как долго в среднем пользователь задерживается на страницах. Не хотелось бы грузить клиентскую часть, тем более, цель этой статьи как раз таки научить избегать этого. Будем использовать аналитическую функцию LAG, для лучшего понимания рассмотрим все мысли по шагам.

|  |  |
| --- | --- |
| **Шаг 0** |  |
| Выберем лог посещений для одного пользователя *'bart'*  sql **[1]**  select  to\_char(T, 'HH24:MI:SS') T, PAGE, USER\_NAME from **APP\_STAT** where **USER\_NAME** = *'bart'* order by T  Как видим, для того, чтобы узнать количество времени, которое пользователь проводит, например, на странице 'login', нужно из времени, которое указано во второй строке вычесть время, которое указано в первой: 22/06/2008 10:31:00 [минус] 22/06/2008 10:30:00 [равно] 60 секунд  Маленькое замечание для тех, кто не помнит: в [**Oracle**](http://www.itshop.ru/Oracle/l2t1f2c0sc0) результат (date - date) - это количество дней между датами. Число это может быть дробным, так что легко узнать количество минут или секунд, умножив на соответственное число.  Первой моей мыслью обычно было загрузить эти данные на клиента и обработать результат на Java (C#, C++, [подставьте сюда свой любимый язык программирования]), но теперь я знаю, что существует такая аналитическая функция, как **LAG** | |  |  |  | | --- | --- | --- | | **T** | **PAGE** | **USER\_NAME** | | 10:30:00 | login | bart | | 10:31:00 | search 1 | bart | | 10:31:20 | search 1 | bart | | 10:31:30 | search 1 | bart | | 10:32:00 | new doc | bart | | 10:36:20 | save doc | bart | | 10:36:30 | list doc | bart | | 10:37:00 | view doc | bart | | 10:37:10 | edit doc | bart | | 10:38:30 | save doc | bart | | 10:38:45 | logout | bart | |
| **Шаг 1** |  |
| Выберем данные для пользователя *'bart'* так, чтобы увидеть разницу во времени между текущей операцией в строке лога и предыдущей  sql **[2]**  select  to\_char(T, 'HH24:MI:SS') T, (T - **lag**(T, 1, null) over (**order by T**)) \* (24 \* 60 \* 60) DIFF, PAGE,  USER\_NAME from **APP\_STAT** where **USER\_NAME** = 'bart' order by T  Как видите, в колонке **DIFF** выводится разница между временем в колонке **T** для текущей строки и предыдущей. Не вдаваясь в подробности, пойдем дальше, объяснения будут позже. | |  |  |  |  | | --- | --- | --- | --- | | **T** | **DIFF** | **PAGE** | **USER\_NAME** | | 10:30:00 |  | login | bart | | 10:31:00 | 60 | search 1 | bart | | 10:31:20 | 20 | search 1 | bart | | 10:31:30 | 10 | search 1 | bart | | 10:32:00 | 30 | new doc | bart | | 10:36:20 | 260 | save doc | bart | | 10:36:30 | 10 | list doc | bart | | 10:37:00 | 30 | view doc | bart | | 10:37:10 | 10 | edit doc | bart | | 10:38:30 | 80 | save doc | bart | | 10:38:45 | 15 | logout | bart | |
| **Шаг 2** |  |
| Мы выбрали с помощью [sql [2]](http://www.interface.ru/home.asp?artId=22447#sql%5B2%5D) список операций для пользователя и время, которое прошло между двумя подряд идущими операциями. Но нас интересуют все пользователи (мы хотим вычислить, сколько времени пользователь тратит между кликами). Во втором шаге мы выберем все данные, не фильтруя записи по пользователям. Тут есть определенная проблема, так как у нас много пользователей и когда *'bart'* думает, '*homer'* уже успел что-то сделать. Т.е. последовательность действий и, соответственно, вычисление разности, нужно вести в контексте одного пользователя. Если мы используем [sql [2]](http://www.interface.ru/home.asp?artId=22447#sql%5B2%5D) без фильтра, фактически мы получим время, в течении которого приходили клики, не важно от кого. Нас это не устраивает, потому пишем так:  sql **[3]**  select  to\_char(T, 'HH24:MI:SS') T, (T - **lag**(T, 1, null) over (**partition by USER\_NAME** **order by T**)) \* (24 \* 60 \* 60) DIFF, PAGE, USER\_NAME from **APP\_STAT** order by T  Теперь у нас есть результирующий набор с временем в секундах, которое пользователь думает между кликами. Вычислить среднее время - очень просто | |  |  |  |  | | --- | --- | --- | --- | | **T** | **DIFF** | **PAGE** | **USER\_NAME** | | 10:30:00 |  | login | bart | | 10:31:00 | 60 | search 1 | bart | | 10:31:20 |  | login | homer | | 10:31:20 | 20 | search 1 | bart | | 10:31:30 | 10 | search 1 | bart | | 10:32:00 | 30 | new doc | bart | | 10:32:10 | 50 | list doc | homer | | 10:32:20 | 10 | view doc | homer | | 10:33:40 | 80 | list doc | homer | | 10:34:00 | 20 | view doc | homer | | 10:36:20 | 260 | save doc | bart | | 10:36:30 | 10 | list doc | bart | | 10:36:30 | 150 | delete doc | homer | | 10:37:00 | 30 | view doc | bart | | 10:37:00 | 30 | logout | homer | | 10:37:10 | 10 | edit doc | bart | | 10:38:30 | 80 | save doc | bart | | 10:38:45 | 15 | logout | bart | |
| **Шаг 3** |  |
| Просто вычисляем среднее значение:  sql **[4]**  select **avg**(DIFF) from ( select (T - **lag**(T, 1, null) over (**partition by USER\_NAME** **order by T**)) \* (24 \* 60 \* 60) DIFF from **APP\_STAT** )  Обратите внимание: аналитические функции нельзя использовать в агрегациях, так что мы использовали подзапрос. | Получили: 54.0625 |

Теперь пришла пора раскрыть магию. Как же все это работает:  
Синтаксис функции LAG:

**LAG** (выражение, [смещение,] [значение-по-умолчанию]) **over** ([**partition by раздел**] **выражение-сортировки**)

Что же она возвращает? Все очень просто: значение [выражения] для строки, которая находится выше текущей на [смещение] строчек, как если бы все строки сортировались по [**выражение-сортировки**]. При этом весь набор строк разбивается на независимые наборы так, что значения в столбцах [**раздел**] для одного набора - одинаковые. Каждый набор обрабатывается отдельно.

sql **[2]**

select  
  
to\_char(T, 'HH24:MI:SS') T,  
(T - **lag**(T, 1, null) over (**order by T**)) \* (24 \* 60 \* 60) DIFF,  
PAGE,   
USER\_NAME  
from **APP\_STAT**  
where **USER\_NAME** = 'bart'  
order by T

В sql **[2]** мы использовали простой вариант без использования раздела (**partition**). На самом деле раздел есть, но он является полным результирующим набором (т.е. все строки из таблицы **APP\_STAT**, где пользователь == *'bart'*).

Кроме того, мы использовали **сортировку**. Для чего она? Как Вы знаете, база данных вольна вернуть строки в любом порядке, а нам необходимо вычислять разницу между текущей операцией в логе и предыдущей. Этот порядок можно получить, отсортировав результат по времени (поле **T**). Для этого и нужна сортировка, чтобы база данных поняла, что такое "предыдущая строка".

Итак, в sql **[2]** функция **LAG** возвращает:

Значение поля **T** из предыдущей строки (смещение 1) или null (значение-по-умолчанию), если предыдущей строки не существует. При этом **сортировка** выполняется по полю **T**.

sql **[3]**

select  
  
to\_char(T, 'HH24:MI:SS') T,  
(T - **lag**(T, 1, null) over (**partition by USER\_NAME** **order by T**)) \* (24 \* 60 \* 60) DIFF,  
PAGE,  
USER\_NAME  
from **APP\_STAT**  
order by T

В sql **[3]** мы использовали дополнительное выражение раздела (**partition**). Мотивация разделения на раздел с точки зрения логики тут такова: мы хотим понять, сколько думал отдельный пользователь, а не как часто на сервер приходил запрос. Для этого разности во времени нужно считать в контексте каждого отдельного пользователя. Как получить это? Ответ напрашивается сам с собой: нужно рассматривать выборку строк по каждому отдельному пользователю. Для этого и был придуман параметр [**partition by раздел**]. Мы знаем, что нам нужно разделить полученный набор так, чтобы каждая группа строк содержала упоминание только одного пользователя, затем мы хотим применить ту же операцию, что и в [sql [2]](http://www.interface.ru/home.asp?artId=22447#sql%5B2%5D), то есть подсчитать разность во времени. Визуально это выглядит так:

|  |  |
| --- | --- |
| **Шаг 0** |  |
| Это все данные из таблицы лога  select  to\_char(T, 'HH24:MI:SS') T,  PAGE,  USER\_NAME USR from **APP\_STAT** order by T asc | |  | | --- | |  | | **T** | **PAGE** | **USER** | | 10:30:00 | login | bart | | 10:31:00 | search 1 | bart | | 10:31:20 | login | homer | | 10:31:20 | search 1 | bart | | 10:31:30 | search 1 | bart | | 10:32:00 | new doc | bart | | 10:32:10 | list doc | homer | | 10:32:20 | view doc | homer | | 10:33:40 | list doc | homer | | 10:34:00 | view doc | homer | | 10:36:20 | save doc | bart | | 10:36:30 | delete doc | homer | | 10:36:30 | list doc | bart | | 10:37:00 | logout | homer | | 10:37:00 | view doc | bart | | 10:37:10 | edit doc | bart | | 10:38:30 | save doc | bart | | 10:38:45 | logout | bart | |
| **Шаг 1** |  |
| Разделим данные для каждого пользователя  select   to\_char(T, 'HH24:MI:SS') T, PAGE, USER\_NAME USR from **APP\_STAT** where USER\_NAME = *'bart'* order by T asc  select   to\_char(T, 'HH24:MI:SS') T,  PAGE,  USER\_NAME USR from **APP\_STAT** where USER\_NAME = *'homer'* order by T asc | bart:   |  | | --- | |  | | **T** | **PAGE** | **USR** | | 10:30:00 | login | bart | | 10:31:00 | search 1 | bart | | 10:31:20 | search 1 | bart | | 10:31:30 | search 1 | bart | | 10:32:00 | new doc | bart | | 10:36:20 | save doc | bart | | 10:36:30 | list doc | bart | | 10:37:00 | view doc | bart | | 10:37:10 | edit doc | bart | | 10:38:30 | save doc | bart | | 10:38:45 | logout | bart |   homer:   |  | | --- | |  | | **T** | **PAGE** | **USR** | | 10:31:20 | login | homer | | 10:32:10 | list doc | homer | | 10:32:20 | view doc | homer | | 10:33:40 | list doc | homer | | 10:34:00 | view doc | homer | | 10:36:30 | delete doc | homer | | 10:37:00 | logout | homer | |
| **Шаг 2** |  |
| Подсчитаем время, которое пользователь тратил на обдумывание "между кликами"  Вы уже знаете, что это можно сделать с помощью функции **LAG**:  select   to\_char(T, 'HH24:MI:SS') T, (T - lag(T, 1, null) over (**order by T**))  \* (24 \* 60 \* 60) DIFF, PAGE, USER\_NAME  from **APP\_STAT** where USER\_NAME = *'bart'* order by T  select   to\_char(T, 'HH24:MI:SS') T, (T - lag(T, 1, null) over (**order by T**)) \* (24 \* 60 \* 60) DIFF, PAGE, USER\_NAME  from **APP\_STAT** where USER\_NAME = *'homer'* order by T | bart:   |  | | --- | |  | | **T** | **DIFF** | **PAGE** | **USER\_NAME** | | 10:30:00 |  | login | bart | | 10:31:00 | 60 | search 1 | bart | | 10:31:20 | 20 | search 1 | bart | | 10:31:30 | 10 | search 1 | bart | | 10:32:00 | 30 | new doc | bart | | 10:36:20 | 260 | save doc | bart | | 10:36:30 | 10 | list doc | bart | | 10:37:00 | 30 | view doc | bart | | 10:37:10 | 10 | edit doc | bart | | 10:38:30 | 80 | save doc | bart | | 10:38:45 | 15 | logout | bart |   homer:   |  | | --- | |  | | **T** | **DIFF** | **PAGE** | **USER\_NAME** | | 10:31:20 |  | login | homer | | 10:32:10 | 50 | list doc | homer | | 10:32:20 | 10 | view doc | homer | | 10:33:40 | 80 | list doc | homer | | 10:34:00 | 20 | view doc | homer | | 10:36:30 | 150 | delete doc | homer | | 10:37:00 | 30 | logout | homer | |
| **Шаг 3** |  |
| Ну, а теперь сольем результаты в одну таблицу  Как это сделать с помощью **LAG** в одном запросе - ниже. | |  | | --- | |  | | **T** | **DIFF** | **PAGE** | **USER\_NAME** | | 10:30:00 |  | login | bart | | 10:31:00 | 60 | search 1 | bart | | 10:31:20 |  | login | homer | | 10:31:20 | 20 | search 1 | bart | | 10:31:30 | 10 | search 1 | bart | | 10:32:00 | 30 | new doc | bart | | 10:32:10 | 50 | list doc | homer | | 10:32:20 | 10 | view doc | homer | | 10:33:40 | 80 | list doc | homer | | 10:34:00 | 20 | view doc | homer | | 10:36:20 | 260 | save doc | bart | | 10:36:30 | 10 | list doc | bart | | 10:36:30 | 150 | delete doc | homer | | 10:37:00 | 30 | view doc | bart | | 10:37:00 | 30 | logout | homer | | 10:37:10 | 10 | edit doc | bart | | 10:38:30 | 80 | save doc | bart | | 10:38:45 | 15 | logout | bart | |

Эти все операции производятся с помощью одного SQL:

select   
T,  
(T - lag(T, 1, null)  
over (**partition by USER\_NAME** **order by T**))   
\* (24 \* 60 \* 60) DIFF,  
PAGE,   
USER\_NAME   
from APP\_STAT  
order by T

Выражение '**partition by USER\_NAME**' как раз и выполнило разделение таблицы из Шага 0 так, что мы получили две таблицы из Шага 1. Далее данные обрабатывались отдельно как в Шаге 2, строки склеились и результат вернулся отсортированным по полю T так, как записано в основном запросе. В результате вы получим такую-же таблицу, как и в результате Шага 3.  
  
Обратите внимание, что финальная сортировка не влияет на вычисление функции **LAG**, так как в качестве параметра ей также передаются правила, по которому нужно отсортировывать каждый набор. Другими словами, финальная сортировка выполняется после вычисления в аналитических функциях. Например, если мы хотим посмотреть, как вел себя каждый пользователь, мы можем поменять сортировку в основном запросе:

select   
T,  
(T - lag(T, 1, null)  
over (**partition by USER\_NAME** **order by T**))   
\* (24 \* 60 \* 60) DIFF,  
PAGE,   
USER\_NAME   
from APP\_STAT  
order by USER\_NAME

Получим правильный результат:

|  |  |  |  |
| --- | --- | --- | --- |
| **T** | **DIFF** | **PAGE** | **USER\_NAME** |
| 10:30:00 |  | login | bart |
| 10:31:00 | 60 | search 1 | bart |
| 10:31:20 | 20 | search 1 | bart |
| 10:31:30 | 10 | search 1 | bart |
| 10:32:00 | 30 | new doc | bart |
| 10:36:20 | 260 | save doc | bart |
| 10:36:30 | 10 | list doc | bart |
| 10:37:00 | 30 | view doc | bart |
| 10:37:10 | 10 | edit doc | bart |
| 10:38:30 | 80 | save doc | bart |
| 10:38:45 | 15 | logout | bart |
| 10:31:20 |  | login | homer |
| 10:32:10 | 50 | list doc | homer |
| 10:32:20 | 10 | view doc | homer |
| 10:33:40 | 80 | list doc | homer |
| 10:34:00 | 20 | view doc | homer |
| 10:36:30 | 150 | delete doc | homer |
| 10:37:00 | 30 | logout | homer |

**Заключение**

В заключении отмечу, что аналитических функций - немало и они предоставляют различные возможности. В этой статье не описывались так называемые "окна" (window), которые наряду с "разделом" используются для определения группы строк, над которым нужно производить вычисления.

**Ссылки по теме**

* [Обратиться в "Интерфейс" за дополнительной информацией/по вопросу приобретения продуктов](http://www.interface.ru/iservices/mailfrompage.asp?To=mail@interface.ru&final=close)
* [Подробнее о продуктах Oracle](http://www.interface.ru/home.asp?artId=64&vId=17)
* [Приобрести продукты Oracle в ITShop.ru](http://www.itshop.ru/Oracle/l2t1f2c0sc0)
* [Курсы по продуктам Oracle](http://www.itshop.ru/Oracle/l2t3f2c0sc0)