

# Построение профиля сессии в СУБД Oracle на основе триггера on-logoff

Для СУБД Oracle 9i, 10g

(<http://dsvolk.msk.ru/oracle/tuning/>)

Дмитрий Волков, [dsvolk@jet.msk.su](mailto:dsvolk@jet.msk.su)

Игорь Старостин, [starostin@jet.msk.su](mailto:starostin@jet.msk.su)

Инфосистемы Джет, 2006 г.

<b>1. ВВЕДЕНИЕ .....</b>	<b>4</b>
<b>2. ВРЕМЯ ОТКЛИКА .....</b>	<b>4</b>
2.1    ВРЕМЯ ОТКЛИКА НА УРОВНЕ ЭКЗЕМПЛЯРА .....	5
2.2    ВРЕМЯ ОТКЛИКА НА УРОВНЕ СЕССИИ.....	6
<b>3. СБОР ДАННЫХ .....</b>	<b>6</b>
3.1    НА УРОВНЕ ЭКЗЕМЛЯРА .....	6
3.2    НА УРОВНЕ СЕССИИ .....	6
<b>4. АНАЛИЗ ДАННЫХ .....</b>	<b>7</b>
4.1    ОЦЕНКА ВОЗМОЖНОСТЕЙ ОПТИМИЗАЦИИ НА СИСТЕМНОМ УРОВНЕ .....	7
4.2    ОЦЕНКА ВОЗМОЖНОСТЕЙ ОПТИМИЗАЦИИ НА УРОВНЕ СЕССИЙ .....	7
4.2.1    Поиск “проблемных” сессий.....	8
4.2.2    Поиск наиболее тяжелых сессий .....	8
4.2.3    Определение времени бизнес транзакции .....	9
4.2.4    Учет передачи данных по сети.....	9
<b>5. ГРАНИЦЫ ПРИМЕНЕНИЯ .....</b>	<b>10</b>
<b>6. ВКЛЮЧЕНИЕ ТРАССИРОВКИ.....</b>	<b>11</b>
<b>7. ЗАКЛЮЧЕНИЕ .....</b>	<b>12</b>
<b>СПИСОК ЛИТЕРАТУРЫ.....</b>	<b>13</b>

## АННОТАЦИЯ

Оптимизация ИС путем оптимизации пользовательских модулей последовательно, модуль за модулем, наверно самый эффективный путь. Но такая оптимизация, скорее всего, потребует очень продолжительного времени, которого, как правило, у заказчика нет. Не получив быстрого результата, заказчик вообще можем принять решение о прекращении проекта.

Стоимость такой оптимизации также вряд ли устроит заказчика. Следовательно, необходимо сформировать критерии удовлетворительной работы пользователей, начав оптимизацию с наиболее критических модулей.

Определение списка модулей, в свою очередь, ведет к необходимости проводить опрос пользователей. Таким образом, необходимо затратить существенное время еще до принятия заказчиком решения о направлении движения и стоимости работ. Получается замкнутый круг – нельзя указать стоимость, направление и продолжительность работ, не начав достаточно затратное обследование.

Данная статья показывает, как используя формулу для расчета времени отклика, с минимальными временными затратами:

1. Оценить возможность оптимизации ИС на системном уровне;
2. Определить наиболее “проблемные” сессии;
3. Определить сессии, создающие наибольшую нагрузку на систему;
4. Определить время выполнения бизнес-транзакций.

Собранные данные являются основой для принятия решения о направлении оптимизации: оптимизации на системном уровне или прикладного ПО. В случае принятия о выполнении оптимизации прикладного ПО собранные данные уже содержат критерии для включения трассировки пользовательских сессий.

На основании собранных данных можно предоставить бизнесу план оптимизации системы с указанием:

1. Продолжительности и состава работ: списка модулей для оптимизации;
2. Достижимого результата: возможного ускорения;
3. Стоимости работ.

## 1. Введение

При принятии решения об ускорении какой-либо информационной системы (ИС) возникает вопрос – что, собственно, необходимо ускорять?

Варианта может быть 2: не устраивает работа ИС в целом, или есть конкретные жалобы на тот или иной модуль.

Как оптимизировать конкретный модуль достаточно ясно, благодаря работам Cary Millsap: следует включать трассировку и анализировать ее вывод. Но mr. Millsap полагает, что, указать модули для оптимизации должен заказчик. Но такой подход обладает и рядом недостатков. Во-первых, реакция на жалобы пользователей - реактивный подход, а хотелось бы иметь возможность предупреждения подобных ситуаций (проактивный подход). Во-вторых, список модулей может оказаться достаточно велик, а общая причина крыться в недостаточно производительном дисковом массиве. В этом случае, оптимизация модуль за модулем может оказаться слишком долгой, дорогой, и поэтому невостребованной.

Если не устраивает работа ИС в целом, выделение конкретных модулей может стать весьма сложной задачей. В компаниях не всегда существуют согласованные бизнес-требования ко времени выполнения. А сбор таких требований, их согласование – предмет отдельного обследования. Если же требования существуют, не обязательно существует система регистрации недовольства пользователей (help desc). Возможна и ситуация, когда все смирились с существующим положением вещей и уже не считают нужным обращаться в службу help desc.

Таким образом, в идеале, следует самостоятельно оценить текущую загрузку ИС, возможности оптимизации ИС на системном уровне, собрать список модулей, с указанием их влияния на ИС. А затем уже сравнить полученные данные с требованиями заказчика.

Произвести необходимые оценки удобнее всего, построив профиль для каждой сессии пользователя. Профиль сессии позволит оценить время отклика пользователя, и таким образом оценить время выполнения бизнес-операций.

Построенный профиль сессии показывает: общую продолжительность работы сессии, время реальной работы, число транзакций, распределение времени обслуживания и времени ожидания, а также статистику сессии.

Статистика сессии: физический ввод-вывод, логический ввод-вывод, сетевой ввод-вывод, потребление ресурсов

```

Session info :
-----
Os user / Oracle user / Machine :DSBOOK\dsvolk / DSVOLK / GPVK\DSBOOK
Module / Program                :SQL*Plus / sqlplusw.EXE

Responce time:
-----
Total (s) / Work time (s) / Transactions : 34.00 / 23.40 / 1
Service (s) / Wait (s) / Unaccounted time (s) : 1.25 / 21.40 / .75
Service (%) / Wait (%) / Unaccounted time (%) : 5.34% / 91.45% / 3.21%

Session stats:
-----
IO (Mb) / Cache Memory (Mb) / Net IO (Mb) : 10.20 / 6.34 / 1.92

Total work :
-----
PGA Memory usage (Mb) / Total changes (Mb) : 4.99 / .03

```

## 2. Время отклика

Для понимания данной статьи необходимо быть знакомым с формулой времени отклика:

Время отклика = Время обслуживания + Время ожидания  
*(Response Time = Service Time + Wait Time)*

Данные формула была впервые опубликована в работе “The COE perfomance method”

Время обслуживания может быть получено из динамических представлений V\$SYSSTAT или V\$SESSTAT как компонента “CPU used by this session”

```
select a.value "Total CPU time"
from v$sysstat a
where a.name = 'CPU used by this session';
```

Время ожидания может быть получено из динамических представлений V\$SYSTEM\_EVENT и V\$SESSION\_EVENT, суммируя все времена ожидания, за исключением некоторых из них:

```
select sum(time_waited) "Total Wait Time"
from v$system_event
where event not in (<Some Idle events>);
```

Мы можем несколько улучшить нашу формулу, понимая, насколько аккуратно Oracle накапливает времена. Так в Oracle Database Reference 10g Release 2 (10.2) для статистики CPU used by this session сказано: если пользовательский вызов завершился быстрее чем 10 мс, то к статистике будет добавлено 0 мс. Также известно, что подобная “неаккуратность” может происходить в сильно перегруженных серверных комплексах.

Для событий ожидания действует такая же логика: если ожидания было менее 1 мкс для oracle 9i, 10g и менее 1 мс для версии 8i то в результате будет записан 0.

Таким образом, следует записать:

**Время отклика = Время обслуживания + Время ожидания + Неучтенное время**  
*(Response Time = Service Time + Wait Time + Unaccounted Time)*

Unaccounted Time представляет собой ошибку измерений. Если в результате анализа, окажется что Unaccounted Time составляет значительный процент от времени ожидания, следует обнаружить и исключить причину такого поведения.

### Время отклика на уровне экземпляра

Время отклика на уровне экземпляра удобнее всего получить из отчета statspack с помощью следующего запроса:

```
select event, time, pctwtt from
(
    select 'Responce time' event
        , (:tcpu*10000 + :twtt)/1000000 time
        , to_number('100') pctwtt
    from dual
    Union
    select 'Service time' event
        , (:tcpu*10000)/1000000 time
        , decode(:twtt + :tcpu*10000, 0, 0,
            100
            * :tcpu*10000
            / (:twtt + :tcpu*10000)) pctwtt
    from dual
    union
    select 'Wait time' event
        , (:twtt)/1000000 time
        , decode(:twtt + :tcpu*10000, 0, 0,
            100
            * :twtt
            / (:twtt + :tcpu*10000)) pctwtt
    from dual
)
order by pctwtt desc;
```

К сожалению, результат представляет собой “среднюю температура по больнице” – т.е. показывает общую ситуацию, но для отдельных сессий может сильно отличаться. Поэтому, построение распределение времени отклика для отдельных сессий очень полезно.

## 2.1 Время отклика на уровне сессии

Можно построить формулу распределения времени отклика для сессии, воспользовавшись динамическими представлениями `v$session_event` и `v$mystat`

На уровне сессии будет интересно получить не только распределение времени отклика, но и определить источник наибольших ожиданий. Однако большое кол-во событий ожидания (более 800 в версии 10g) и статистик (более 300 в 10g) сильно затрудняет дальнейший анализ.

Поэтому для упрощения дальнейшего анализа можно перейти к классам событий ожидания и статистики. Функция

```
apt_stat_class_t (p_statname varchar2 ) return varchar2
```

возвращает класс статистики, а функция

```
apt_event_class_t (p_event varchar2 ) return varchar2
```

Возвращает класс ожидания. Надо отметить, что классы событий ожидания не совпадают с классами, введенными в версии 10g, и это сделано намеренно.

Таким образом, время ожидания получается из `v$session_event`

```
select apt_event_class_t(event) event_class, sum(time_waited_micro) time_waited_micro
from v$session_event se, (select /*+ no_merge */ sid from v$mystat where rownum = 1) ms
where
total_waits != 0 and
se.sid = ms.sid
group by apt_event_class_t(event)
```

А Время обслуживания из `v$mystat`:

```
select
apt_stat_class_t (sn.name) stat_class,
sum(ms.value) value
from
v$mystat ms,
v$statname sn
where
ms.value != 0 and
sn.statistic# = ms.statistic#
group by apt_stat_class_t (sn.name)
```

Как заметит внимательный читатель, на самом деле из статистики извлекается не только информацию о потреблении CPU. Подробности использования прочей информации будут приведены в следующих главах.

## 3. Сбор данных

### 3.1 На уровне экземпляра

Как я уже упоминал, на уровне экземпляра вполне достаточно воспользоваться несколько модифицированным отчетом `statspack`. Установка, сбор данных `statspack` – это хорошо документированная процедура, не оказывающая большого влияния на работу экземпляра, особенно при условии сбора уровня статистики не более 5.

### 3.2 На уровне сессии

На уровне сессии задача сбора данных несколько сложнее. В книге Oracle Wait Interface: A Practical Guide to Performance Diagnostics & Tuning приводится пример триггера `before logoff on database`. Действительно, как показывает практика, влияние такого триггера на работающую производственную систему минимально, а собираемая статистика наиболее полна.

Однако сохранение статистики в БД может негативно сказаться на производительности. Поэтому статистика будет сохраняться во внешнем текстовом файле в формате csv.

```

create or replace trigger jjlogoff_stat_trigger
before logoff on database
declare
type my_lookups_t is table of varchar2(64) index by binary_integer;
.....

v_logf      utl_file.file_type;
v_out       varchar2(32767);

begin

v_logf := utl_file.fopen('LOGOFF_STAT_DIR','jjlogoff_stat.log','a');
v_out := '';

--
-- Session identification
--
select
  se.sid, serial#, username, osuser, machine, server, module, program,
  to_char(logon_time,'dd/mm/rr hh24:mi:ss'), to_char(sysdate,'dd/mm/rr hh24:mi:ss')
into
  v_infos(1), v_infos(2), v_infos(3), v_infos(4), v_infos(5), v_infos(6), v_infos(7),
  v_infos(8), v_infos(9), v_infos(10)
from
  (select /*+ no_merge */ sid from v$mystat where rownum = 1) ms,
  v$session se
where se.sid = ms.sid;

-- Fill result string
for i in 1..v_infos.count loop
  v_out := v_out || v_infos(i) || c_sep;
end loop;
.....

utl_file.put_line(v_logf,v_out);
utl_file.fclose(v_logf);

```

## 4. Анализ данных

### 4.1 Оценка возможностей оптимизации на системном уровне

Обработывая отчеты statspack можно заметить, что время ожидания на уровне экземпляра редко превышает 40-50%. А это значит, что если мы с помощью настроек СУБД или аппаратуры уменьшим время ожидания в 2 раза, наши пользователи, в среднем, получат не более 20% -25%. Возможно отсюда и следует экспериментально известный факт, что с помощью настроек ОС и экземпляра получить более 20% выигрыша в производительности очень сложно.

Хочется отметить, что сложно – не значит невозможно. Хотя и редко, но все еще встречаются случаи, когда неудачное расположение журналов протоколов транзакций (redo logs) тормозит всю систему. Можно привести еще несколько подобных “стандартных” ошибок. В этом случае мы из отчета statspack видим, что время ожидания составляет значительную часть общего времени ответа. Таким образом – перед началом проекта по оптимизации сначала лучше убедиться, что на системном уровне все в порядке. Если необходимо, предложить пути системной оптимизации, но указать и ожидаемый эффект.

### 4.2 Оценка возможностей оптимизации на уровне сессий

Для анализа данных проще всего загрузить собранные с помощью триггера `on-logoff` данные в СУБД. Начиная с версии 9i можно воспользоваться внешними таблицами (organization external) для подобной операции:

```

CREATE TABLE jj_trigger_stat
(
  sid number,
  serial number,
  username varchar2(30), <прочие поля>
)
Organization external
(type oracle loader
DEFAULT DIRECTORY LOGOFF_STAT_DIR
ACCESS PARAMETERS
( RECORDS DELIMITED BY NEWLINE
  FIELDS TERMINATED BY ';'
  MISSING FIELD VALUES ARE NULL
(
  sid, serial, username, <прочие поля>
)
)
)
LOCATION ('jjlogoff_stat.log')
)
PARALLEL 1
REJECT LIMIT UNLIMITED;

```

#### 4.2.1 Поиск “проблемных” сессий

Получив из statspack среднюю оценку времени ожидания по экземпляру, не стоит отчаиваться. Наша задача как раз и состоит в том, чтобы найти те сессии, у которых времена ожидания составляют значительный процент времени отклика.

Для поиска проблемных сессий, воспользуемся следующим критерием: время ожидания составляет более 60% от общего времени ответа.

```

select username, logon_time, (logoff_time - logon_time)*3600*24 T, to_char(R, '99.99') || ' (100%)' R,
to_char(S, '99.99') S, to_char((S/R)*100, '99.99') S_PCT,
to_char(W, '99.99') W, (W/R)*100 W_PCT,
to_char((R - S - W), '99.99') || ' (' || to_char(((R - S - W)/R)*100, '99.99') || '%)' U, s_netrt
from (
select sid, serial, username, logoff_time, logon_time, w_idle, ((logoff_time - logon_time)*3600*24 -
w_idle/1000000) R,
s_cpu/100 S, s_netrt,
(w_administrative + w_application + w_cluster + w_commit + w_concurrency + w_configuration +
w_network + w_systemio + w_userio + w_directio+ w_user_full_scan_io + w_px_activity +
w_enqueue_activity +w_latch_activity + w_other )/1000000 W
from jj_trigger_stat where username <> 'SYS'
)
where W/R*100 > 60
order by logon_time

```

Из листинга ниже, мы видим, что есть сессия, у которой % ожидания составляет 66.62% от времени ответа.

USERNAME	DSVOLK
LOGON_TIME	02.09.2006 10:22:52
T	92
R	3.19 (100%)
S	1.21
S_PCT	37.97
W	2.12
<b>W_PCT</b>	<b>66,62</b>
U	-.15
U_PCT	-4,59478034910652

Для выбранных сессий следует произвести более детальный анализ причин ожидания.

#### 4.2.2 Поиск наиболее тяжелых сессий

Для поиска наиболее тяжелых сессий сначала необходимо определить, какие компоненты (процессор, ввод-вывод, память, сеть) вашей ИС являются узким местом. На листинге ниже приводится пример поиска сессий, выполнивших наибольшее количество чтений.

```
select sid, serial, username, logon_time , (s_user_io + s_directio)/1024/1024 TotalIO
from jj_trigger_stat
order by (s_user_io + s_directio) desc
```

На листинге ниже мы видим, что сессия пользователя DWH (warehouse) выполнил наибольшее кол-во физических чтений.

```
SID      161
SERIAL   72
USERNAME DWH
LOGON TIME 02.09.2006 10:22:52
TOTALIO 2952
```

#### 4.2.3 Определение времени бизнес транзакции

Рассмотрим упрощенный случай, когда в OLTP системе работают операторы по вводу заявок. Ввод каждой заявки завершается операцией фиксации или отказа (commit или rollback). Тогда, по завершении сессии можно рассчитать время, потраченное на каждую транзакцию по следующей формуле:

Время транзакции = (Время обслуживания + Время ожидания) / кол-во транзакций.

```
select
to_char(T, '99999.99') T,
to_char(R, '99999.99') R,
to_char(R/s_transactions, '99999.99') R_P_M,
from (
select logoff_time, username, (logoff_time - logon_time)*3600*24 T,
(logoff_time - logon_time)*3600*24 - w_idle/1000000 - ((w_net_from_c/1000000) - (s_netrt*5/1000)) R
from jj_trigger_stat where username <> 'SYS'
)
```

#### 4.2.4 Учет передачи данных по сети

Во время передачи данных клиенту формируется пара сообщений, которые составляют, так называемый, SQL\*Net roundtrip:

```
WAIT #1: nam='SQL*Net message from client' ela= 5103 p1=1413697536 p2=1 p3=0
WAIT #1: nam='SQL*Net message to client' ela= 2 p1=1413697536 p2=1 p3=0
```

Обратите внимание, что сообщение from client длительностью примерно 5 мс, а сообщение to client 2 мкс, что составляет незначительный процент от общего времени на 1 roundtrip. Однако проблема состоит в том, что если сессия простаивает, например, ожидает ввода данных на клиенте, ожидание SQL\*Net message from client также накапливается. Таким образом, сложно отличить ожидание передачи данных от простоя. К счастью, существует статистика SQL\*Net roundtrips to/from client. Если была передача данных, эта статистика увеличивается.

Следовательно, для аккуратного подсчета времени простоя:

Время простоя сессии = Общее время ожидания SQL\*Net message from client – Кол-во roundtrip\* среднее время 1 roundtrip.

Среднее время 1 roundtrip для ИС предлагается определять на основе анализа трассировочных файлов. Для моей локальной системы оно составляет 5 мс.

Таким образом, формула для определения времени ответа сессии преобразуется:

**Время отклика - Время простоя = Время обслуживания + Время ожидания + Неучтенное время**

На листинге ниже (r2.sql) приводится запрос для определения значений:

```

select
to_char(T, '99999.99') T,
to_char(R, '99999.99') R,
'(100%)' R_PCT,
to_char(S, '9999.99') S, to_char((S/R)*100, '99.99') S_PCT,
to_char(W, '99.99') W, to_char((W/R)*100, '99.99') W_PCT,
to_char((R - S - W) , '9999.99') U, to_char(((R - S - W)/R)*100, '99.99') U_PCT
from (
select logoff_time, username, (logoff_time - logon_time)*3600*24 T,
(logoff_time - logon_time)*3600*24 - w_idle/1000000 - ((w_net_from_c/1000000) - (s_netrt*5/1000)) R,
s_cpu/100 S,
(w_administrative + w_application + w_cluster + w_commit + w_concurrency + w_configuration +
w_network + w_systemio + w_userio + w_directio+ w_user_full_scan_io + w_px_activity +
w_enqueue_activity +w_latch_activity + w_other )/1000000 + s_netrt*5/1000 W
from jj_trigger_stat where username <> 'SYS'
)

```

Для моей сессии sqlplus, в которой я выполнил большой full scan запрос, а затем просматривал полученные данные, профиль сессии выглядит так:

```

T          118.00
R           18.43
R_PCT      (100%)
S           1.22
S_PCT      (6.62%)
W           17.59
W_PCT      95.45%
U           -.38
U_PCT      (-2.07%)

```

Откуда видно: Общее время сессии 118 сек, из них только чуть более 18 сек – время работы, из которых более 95% - время ожидания (передача по сети).

Очевидно, что точность метода зависит от точности определения среднего времени на 1 roundtrip, но другого способа я не знаю.

## 5. Границы применения

Приводимый в данной статье метод обладает рядом ограничений, которые следует учитывать при применении.

Так как технически, сбор данных выполняется в триггере on-logoff, то информация накапливается только тогда, когда сессия выполняет нормальный выход из БД. Если сессия была прервана с помощью команды kill, информация об этой сессии не будет собрана.

Следует также учитывать, что во время включения триггера часть пользовательских сессий может уже работать, а также, что могут существовать сессии, которые не закончатся к моменту выключения триггера. Для более точного учета собираемых данных следует использовать процедуру, собирающую накопленную статистику сессий пользователей до включения/выключения триггера. Так, например, если сессия уже работала на момент включения триггера, то по окончании ее работы из ее статистики необходимо вычесть значения, накопленные до начала работы триггера.

Если сессия выполняет параллельную операцию, то к сожалению, данные об ожиданиях сессии остаются в подчиненных процессах которые выполняли реальное чтение данных. Наша сессия испытывает ожидания вида:

```

PX Deq: Execute Reply
PX Deq: Join ACK
PX Deq: Parse Reply
PX Deq: Signal ACK
PX Deq: Table Q Normal
PX qref latch

```

По завершении сессии, статистика о кол-ве прочитанных данных и потребленном процессорном времени увеличивается у головного процесса. Сами же подчиненные процессы не попадают в триггер on-logoff, что хорошо – не происходит удвоения данных. Перечисленные выше особенности следует учитывать при анализе причин ожидания.

Не каждая пользовательская сессия обязательно выполняет операции фиксации или отказа (commit или rollback). Вполне могут существовать пользователи, осуществляющие только регулярное чтение данных. В этом случае не получится рассчитать время их бизнес-транзакций, опираясь на кол-во фиксаций – их просто нет. Но, в принципе можно перейти к анализу среднего времени выполнения одного пользовательского вызова (user calls). Конечно, данная ситуация требует более глубокого понимания приложения.

## 6. Включение трассировки

Итак, после определения “проблемных” сессий необходимо выполнить их трассировку. Удобнее всего выполнить такую трассировку в триггере **on-logon**. Во время сбора информации накоплено достаточно информации о времени выполнения сессии, имени пользователя, наименовании программы, чтобы построить условие, которые позволит точно отобрать необходимую сессию.

```
-- This trigger runs only for ordinary users, not for dba
--
-- todo syscontext('userenv', 'ip_adress')

create or replace trigger jj_logon_stat_trigger
after logon on database
declare
  v_sid      v$session.sid%type;
  v_serial   v$session.serial#%type;
  v_username v$session.username%type;
  v_osuser   v$session.osuser%type;
  v_machine  v$session.machine%type;
  v_server   v$session.server%type;
  v_module   v$session.module%type;
  v_program  v$session.program%type;
  v_need_trace varchar2(1) := 'N';
  v_stmt     varchar2(1024);
  v_logf     utl_file.file_type;
  c_sep      varchar2(1) := '|';
begin
-- Userenv dosn't have program and so ...

  select
    se.sid, se.serial#, se.username, se.osuser, se.machine, se.server, se.module, se.program
  into
    v_sid, v_serial, v_username, v_osuser, v_machine, v_server, v_module, v_program
  from
    (select /*+ no_merge */ sid from v$mystat where rownum = 1) ms,
    v$session se
  where se.sid = ms.sid;

-- Custom Logic is here

  if ( v_program = 'sqlplus@sf2 (TNS V1-V3)' ) then
    v_need_trace := 'Y';
    v_logf := utl_file.fopen('LOGOFF_STAT_DIR','jjlogon_stat.log','a');
    utl_file.put_line(v_logf,to_char(sysdate,'dd/mm/rr hh24:mi:ss')||c_sep||
      v_sid||c_sep||v_serial);
    utl_file.fclose(v_logf);

  end if;

-- Enable Trace

  if ( v_need_trace = 'Y' ) then

    v_stmt := 'alter session set timed_statistics = true';
    EXECUTE IMMEDIATE v_stmt;

    v_stmt := 'alter session set max_dump_file_size = 2147483647';
    EXECUTE IMMEDIATE v_stmt;

    v_stmt := 'alter session set tracefile_identifier=jj';
    EXECUTE IMMEDIATE v_stmt;

  end if;

end;
/
```

## 7. Заключение

Триггер [on-logout](#) предоставляет нам широкие возможности для выбора “проблемных” сессий по нескольким критериям. При этом сбор информации не оказывает влияния на работу конечных пользователей. Благодаря возможности одновременного анализа с статистики и событий ожидания можно вести поиск по нескольким направлениям. В тоже время объем собираемой информации существенно меньше, чем объем соответствующих файлов трассировки.

Выводы, полученные с помощью собранной информации, могут позволить Вам избежать значительных временных потерь на опрос пользователей, а сразу предложить к оптимизации действительно “проблемные” модули.

Собранная информация должна быть проанализированы и должны быть сформулированы критерии для включения трассировки “проблемных” модулей.

## **Список литературы**

1. Optimizing Oracle Performance, Cary Millsap with Jeff Hol, ISBN 0-596-00527-x
2. THE COE PERFORMANCE METHOD, Roger Snowden, Center of Expertise, Oracle Corporation
3. Oracle Wait Interface: A Practical Guide to Performance Diagnostics & Tuning by Richmond Shee, Kirtikumar Deshpande and K Gopalakrishnan ISBN:007222729
4. How to Activate Extended SQL Trace Cary Millsap Hotsos Enterprises, Ltd.