Киселев Ярослав
КРОК

PGDAY'RUSSIA 17
КОНФЕРЕНЦИЯ ПО БАЗАМ ДАННЫХ

Мониторинг и диагностика производительности приложений с точки зрения Oracle DB

КРОК

Кто я?

Киселев Ярослав
performance engineer, КРОК
yaki@croc.ru

О чем доклад

- ~~Oracle DB — самая лучшая СУБД~~

О чем доклад

- Что делать если что-то тормозит?
- Проблемы сбора и анализа диагностической информации
- Обходим проблемы, изобретая велосипед

О каких приложениях пойдет речь

- Учетные системы: документооборот, ERP…
- Используем SQL полностью

Нефункциональные требования

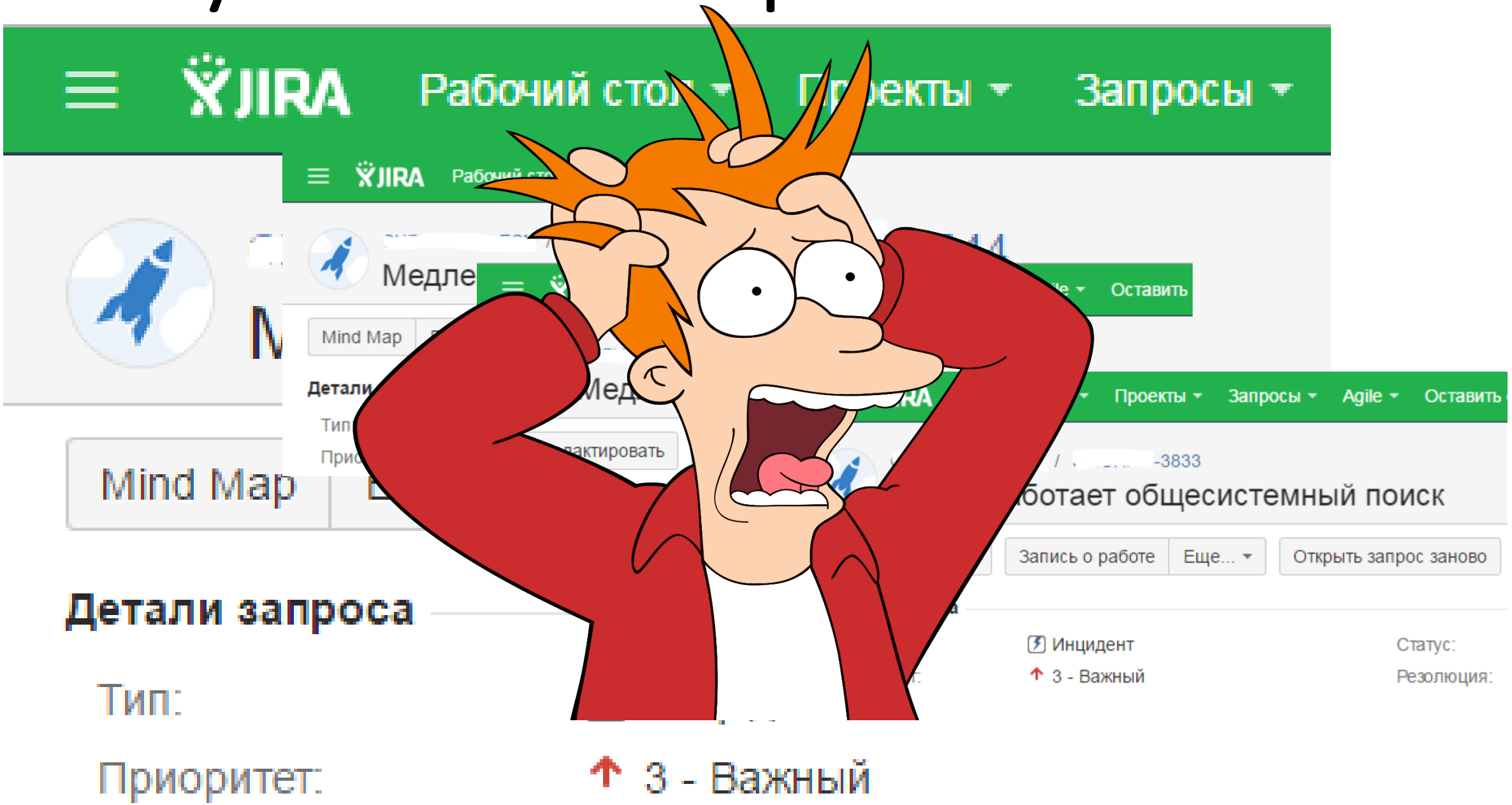
- Задачи пользователя < 5 сек
- Журналы < 7 сек
- Поиск < 15 сек

Конечно замеряем!

Действие	Среднее время, сек	90% Line, сек	Требуемое время, сек
Задачи	0,2	2,1	< 5
Журнал	1,2	4,0	< 7
Поиск	1,7	7,8	< 15

Вжух! И в продакшн!

Но спустя какое-то время...



Найдем интересующий запрос

```
select sql_id, executions,
       elapsed_time, sql_text
from v$sql
where sql_text like '%documents%' // поиск
```

SQL_ID	EXECS	ELAPSED_TIME	SQL_TEXT
8uz9w8p163rxt	2	36383334	select * from...

```
    where author_id = :1
      and reg_date between :2 and :3
```

И посмотрим на план

```
select *
from table(dbms_xplan.display('8uz9w8p163rxt'));
```

```
-----------------------------------------------------------------------------
| Id  | Operation                    | Name      | Rows  | Time     | Cost |
-----------------------------------------------------------------------------
|   0 | SELECT STATEMENT             |           |  1232 |00:00:02|  234 |
|*  1 |  TABLE ACCESS BY INDEX ROWID| DOCUMENTS |  1232 |00:00:02|  234 |
|*  2 |   INDEX RANGE SCAN           | AUTHOR_IX | 36378 |00:00:01|    8 |
-----------------------------------------------------------------------------

Predicate Information :
-----------------------
   1 - filter(REG_DATE>=:2 AND REG_DATE<=:3)
   2 - access(AUTHOR=:1)
```

Ого, как много документов!

А можно ли верить числам из плана?

```
select *
from audit a,
     documents d
where a.event_message = 'DocumentRegistration'
  and a.card_id = d.r_object_id
  and d.author = 'ETL Administrator';
```

Вернется куча строк, инфа 100%

```
-----------------------------------------------------------------------
| Id  | Operation                    | Name             | Rows  | Cost |
-----------------------------------------------------------------------
|   0 | SELECT STATEMENT             |                  |  309K| 87823|
|*  1 |  HASH JOIN                   |                  |  309K| 87823|
|   2 |   TABLE ACCESS BY INDEX ROWID| AUDIT            |  309K| 60345|
|*  3 |    INDEX RANGE SCAN          | AUDIT_EVENT_IX   |  309K| 56802|
|   4 |   TABLE ACCESS BY INDEX ROWID| DOCUMENTS        |  564K|  3720|
|*  5 |    INDEX RANGE SCAN          | AUTHOR_IX        |  572K|   119|
-----------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   1 - access(A.CARD_ID=D.R_OBJECT_ID)
   3 - access(EVENT_MESSAGE='DocumentRegistration')
   5 - access(AUTHOR='ETL Administrator')
```

Ожидание vs. реальность

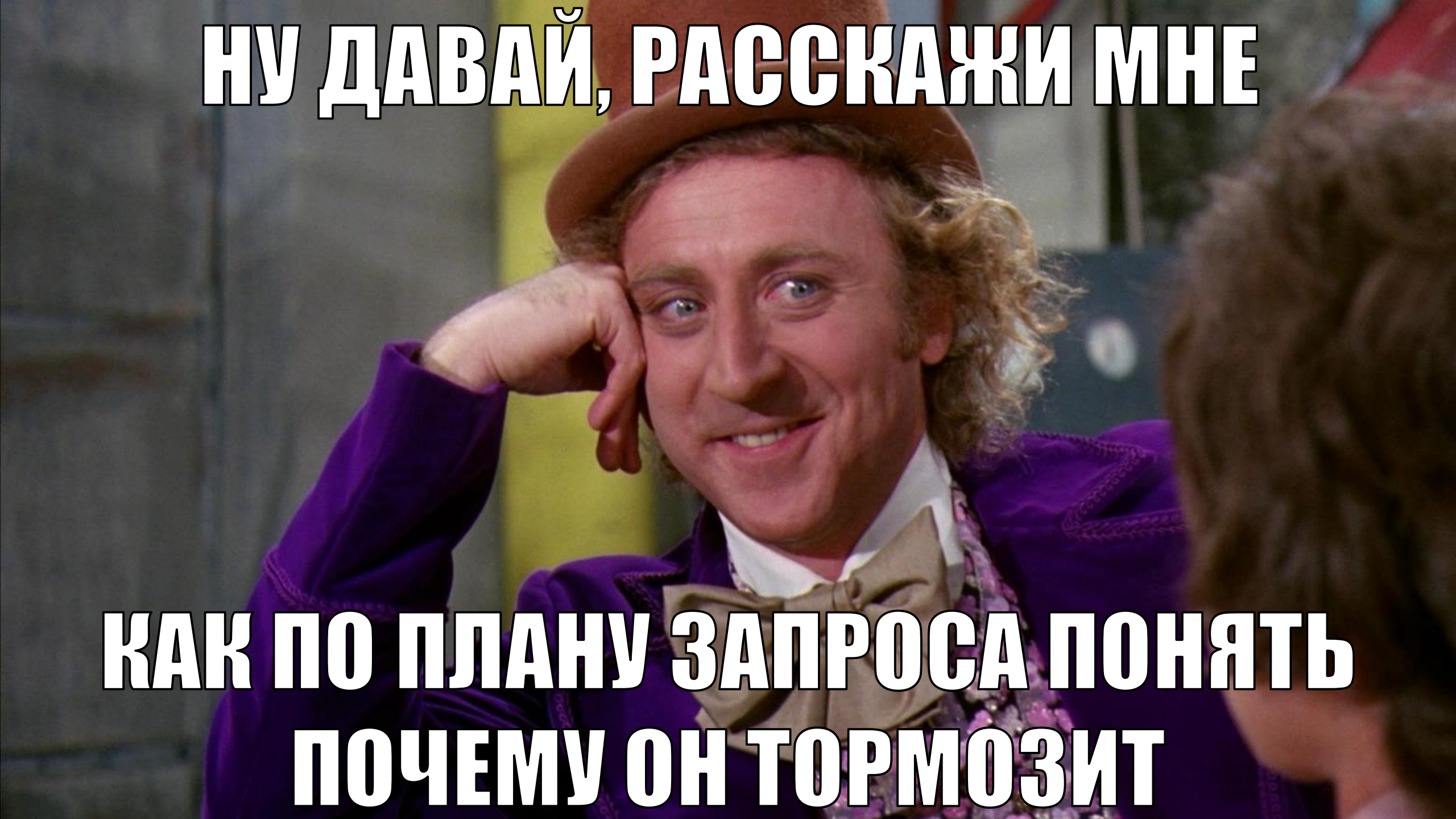
Id	Operation	Name	E-Rows	A-Rows	Buffers
0	SELECT STATEMENT				1785K
* 1	HASH JOIN		309K	?	1785K
2	TABLE ACCESS BY INDEX ROWID	AUDIT	309K	386K	1200K
* 3	INDEX RANGE SCAN	AUDIT_EVENT_IX	309K	386K	827K
4	TABLE ACCESS BY INDEX ROWID	DOCUMENTS	564K	533K	585K
* 5	INDEX RANGE SCAN	AUTHOR_IX	572K	533K	533K

Predicate Information (identified by operation id):

```
   1 - access(A.CARD_ID=D.R_OBJECT_ID)
   3 - access(EVENT_MESSAGE='DocumentRegistration')
   5 - access(AUTHOR='ETL Administrator')
```

Почему плана не достаточно

- Это оценка, основанная на статистике и предположениях оптимизатора
- ~~Бездушный~~ оптимизатор ничего не знает про бизнес
- Доверяй (плану), но проверяй (снимая статистику выполнения)


НУ ДАВАЙ, РАССКАЖИ МНЕ
КАК ПО ПЛАНУ ЗАПРОСА ПОНЯТЬ
ПОЧЕМУ ОН ТОРМОЗИТ

Без статистики никуда:

- Хинт `/*+ gather_plan_statistics */`
  - для отдельных запросов
- `alter session/system set statistics_level=all`
  – на уровне сессии/экземпляра
- Получаем через `dbms_xplan.display_cursor`

Найдем параметры выполнения

```
select name, last_captured, value_string
from v$sql_bind_capture
where sql_id = '8uz9w8p163rxt'
```

NAME	LAST_CAPTURED	VALUE_STRING
:1	15.06.17 19:53:59	IvanovII
:2	15.06.17 19:53:59	28.04.2016
:3	15.06.17 19:53:59	29.04.2016

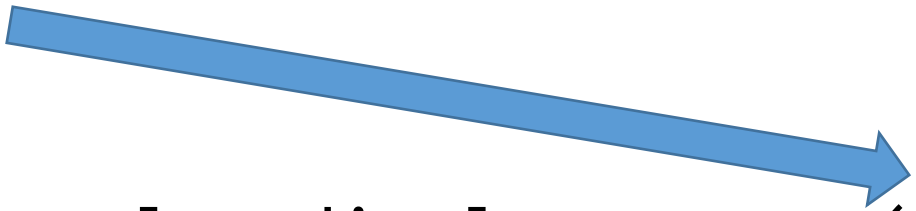
И выполним запрос с хинтом

```
select /*+ gather_plan_statistics hi*/ *
from documents
where author = 'IvanovII'
  and reg_date between '28.04.2016'
                   and '29.04.2016'
```

Найдем sql_id

```
select sql_id
from v$sql
where sql_text like '%statistics hi%'

SQL_ID
-------------
0cv7v8gkz8605
```



```
select *
from table(dbms_xplan.display_cursor('0cv7v8gkz8605',
                                     null, 'ALLSTATS LAST'));
```

Посмотрим на статистику

```
-------------------------------------------------------------------------------
| Id  | Operation                    | Name      | E-Rows| A-Rows| Buffers |
-------------------------------------------------------------------------------
|   0 | SELECT STATEMENT             |           |       |   738 |    9250 |
|*  1 |   TABLE ACCESS BY INDEX ROWID| DOCUMENTS |    86 |   738 |    9250 |
|*  2 |    INDEX RANGE SCAN          | R_DATE_IX |  3774 |  4601 |     354 |
-------------------------------------------------------------------------------
```

А план-то не настоящий!

```
Predicate Information :
----------------------------------------
   1 - filter(AUTHOR=:1)
   2 - access(REG_DATE>=:2 AND REG_DATE<=:3)
```

Вот настоящий план

```
select *
from table(dbms_xplan.display('8uz9w8p163rxt'));
```

```
-----------------------------------------------------------------------------
| Id  | Operation                    | Name      | Rows  | Bytes | Cost |
-----------------------------------------------------------------------------
|   0 | SELECT STATEMENT             |           |  1232 |  914K |  234 |
|*  1 |   TABLE ACCESS BY INDEX ROWID| DOCUMENTS |  1232 |  914K |  234 |
|*  2 |    INDEX RANGE SCAN          | AUTHOR_IX | 36378 |       |    8 |
-----------------------------------------------------------------------------

Predicate Information :
---------------------------------------------
   1 - filter(REG_DATE>=:2 AND REG_DATE<=:3)
   2 - access(AUTHOR=:1)
```

Почему другой?

```
select *
from table(dbms_xplan.display('8uz9w8p163rxt',
                              format => '+peeked binds'));
```

```
-----------------------------------------------------------------------------
| Id  | Operation                           | Name      | Rows  | Bytes | Cost |
-----------------------------------------------------------------------------
|   0 | SELECT STATEMENT                    |           |  1232 |  914K |  234 |
|*  1 |   TABLE ACCESS BY INDEX ROWID       | DOCUMENTS |  1232 |  914K |  234 |
|*  2 |    INDEX RANGE SCAN                 | AUTHOR_IX | 36378 |       |    8 |
-----------------------------------------------------------------------------

Peeked Binds (identified by position):
--------------------------------------

   1 - :1 (VARCHAR2(30)): 'IvanovII'
   2 - :2 (VARCHAR2(30)): '28.02.2016'
   3 - :3 (VARCHAR2(30)): '29.04.2016'
```

```
select /*+ gather_plan_statistics hi*/ *
from documents
where author = 'IvanovII'
  and reg_date between '28.04.2016'
                   and '29.04.2016'
```

При попытке воспроизведения:

- Нужно следить за новым планом
- `v$sql_bind_capture` хранит не все параметры (сэмплирует каждые N секунд)

Статистика выполнения

+ Показывает реальные значения:
  - Rows
  - Buffer Gets, Physical Reads
  - Memory Used
  - Time Elapsed

- Нужно воспроизводить запрос, постфактум не получить

На помощь приходит SQL Monitor

- Появился в 11g (на 10g страдаем)
- Статистика выполнения с плюшками:
  - Мониторятся все запросы дольше 5 сек – возможен анализ постфактум
  - Параметры запроса
  - Buffer gets, IO time, CPU time

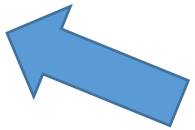
Чтобы получить отчет

```
select dbms_sqltune.report_sql_monitor(
            sql_id => :1,
            sql_exec_id => :2)
from dual;
```

Найдем наш запрос

```
select sql_id, sql_exec_id
from v$sql_monitor
where sql_text like '%audit%documents%';
```

```
0 Rows selected
```



Можно опоздать

Найдем наш запрос и получим отчет

```
select sql_id, sql_exec_id
from v$sql_monitor
where sql_text like '%audit%documents%';
```

```
SQL_ID        SQL_EXEC_ID
------------- -----------
bt34bh6ygbx7w    16777216
```

```
select dbms_sqltune.report_sql_monitor(
            sql_id =>'bt34bh6ygbx7w',
            sql_exec_id =>'16777216')
from dual;
```

Общая информация и параметры

```
Global Information
------------------------------
Status                  :  DONE (ALL ROWS)
Instance ID             :  1
Session                 :  ECM (435:9871)
SQL ID                  :  bt34bh6ygbx7w
SQL Execution ID        :  16777216
Execution Started       :  06/20/2017 15:23:39
First Refresh Time      :  06/20/2017 15:39:45
Last Refresh Time       :  06/20/2017 15:39:45
Duration                :  966s
```

```
Binds
========================================================================
|    Name    | Position |      Type      |              Value           |
========================================================================
| :SYS_B_0 |        1 | VARCHAR2(32) | DocumentRegistration     |
| :SYS_B_1 |        2 | VARCHAR2(32) | ETL Administrator        |
========================================================================
```

Общая статистика и выполнения

Global Stats

Elapsed Time(s)	Cpu Time(s)	IO Waits(s)	Fetch Calls	Buffer Gets	Read Reqs	Read Bytes
981	38	943	1	2M	1M	9GB

SQL Plan Monitoring Details (Plan Hash Value=2944863125)

Id	Operation	Name	Rows (Estim)	Time	Rows (Actual)
0	SELECT STATEMENT				
1	HASH JOIN		310K	66	0
2	TABLE ACCESS BY INDEX ROWID	AUDIT	310K	965	387K
3	INDEX RANGE SCAN	AUDIT_EVENT_IX	310K	956	387K
4	TABLE ACCESS BY INDEX ROWID	DOCUMENTS	564K	2	534K
5	INDEX RANGE SCAN	AUTHOR_IX	573K	1	534K

Общая статистика и выполнения

```
Global Stats
=============================================================
| Elapsed |   Cpu   |    IO    | Fetch | Buffer | Read | Read  |
| Time(s) | Time(s) | Waits(s) | Calls |  Gets  | Reqs | Bytes |
=============================================================
|     981 |      38 |      943 |     1 |     2M |   1M |   9GB |
=============================================================


SQL Plan Monitoring Details (Plan Hash Value=2944863125)
=============================================================================
| Id |           Operation            |      Name      | Rows    | Time |  Rows   |
|    |                                |                |(Estim)  |      |(Actual) |
=============================================================================
|  0 | SELECT STATEMENT               |                |         |      |         |
|  1 |   HASH JOIN                    |                |    310K |   66 |       0 |
|  2 |    TABLE ACCESS FULL           | AUDIT          |    310K |  965 |    387K |
|  4 |    TABLE ACCESS BY INDEX ROWID | DOCUMENTS      |    564K |    2 |    534K |
|  5 |     INDEX RANGE SCAN           | AUTHOR_IX      |    573K |    1 |    534K |
=============================================================================
```

А какой индекс создать?


КОГДА В SQL MONITOR
НЕ НАШЕЛ ПРЕДИКАТЫ
TURBOPORTAL

SQL Monitor – панацея?

- Нет предикатов
- Устаревает

Для анализа проблемы нужны:

- План запроса
- Статистика (+ bind параметры)
- Доступ к продакшну:
  - Удаленный доступ
  - Адекватные админы
  - Выезд к заказчику

Что мы хотим видеть?

- Сводка по работе запросов
- Планы запросов
- Статистика выполнения запросов
- Удобство использования

Как ограничить множество запросов?

Журналы < 7 сек

```
create or replace view croc_perf_info_query
as select
        'Журнал' category,
        'sql_text like ''%_journal%''' search_clause,
        7 limit
union all
select 'Журнал.Входящие',
        'sql_text like ''%incoming_journal%''',
        7
```

Сводка по работе запросов

- Тип операции
- Кол-во выполнений
- Время выполнения
- Текущее предельное значение
- *А есть ли у нас проблемы?*

Сэмплируем v$sql каждые 10 минут

```
insert into v_sql
select current_timestamp,
  case when sql_text like '%_journal%' then
    case when sql_text like '%incoming_journal%'
         then 2 -- Журнал.Входящие
         else 1 -- Журнал
    end category,
    t.executions, t.elapsed_time
from v$sql t
where sql_text like '%_journal%';
```

Сводка по работе запросов

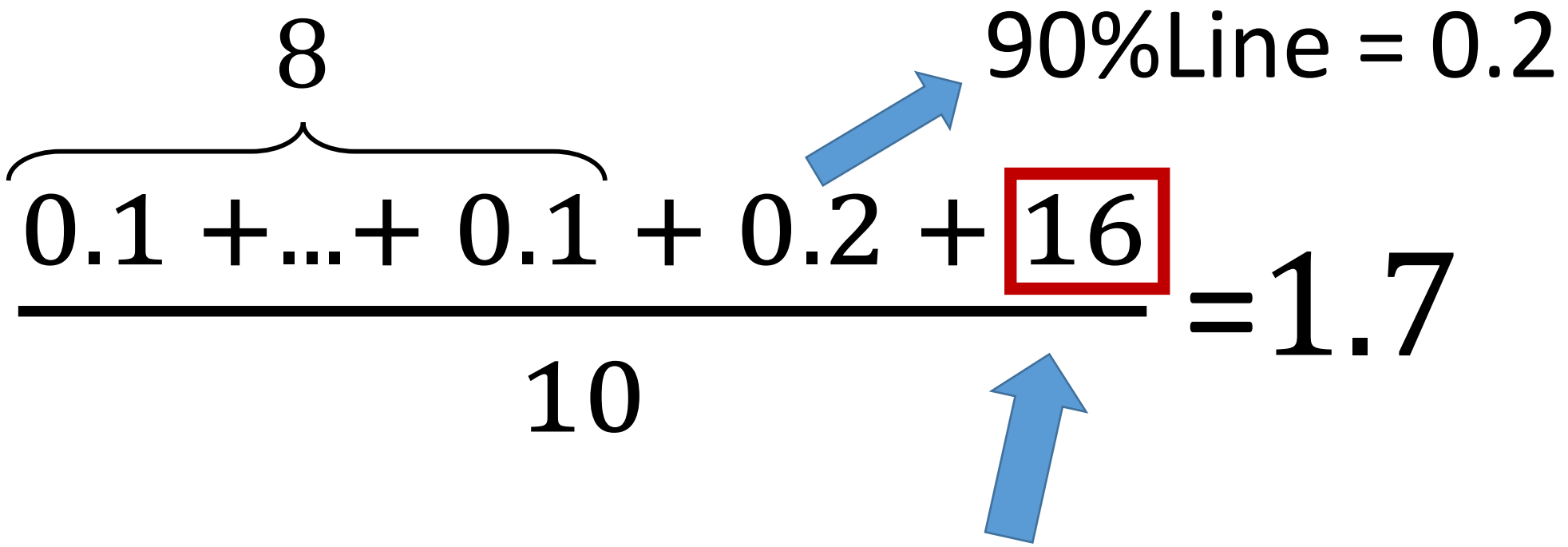
Operation name	Executions	Avg. Time	Limit
Журнал.Внутренние	67	1.6 s	7 s
Журнал.Входящие	72	246 ms	7 s
Журнал.ДД	302	1.7 s	7 s
Журнал.Доверенности	34	990 ms	7 s
Журнал.Заявки	54	2.6 s	7 s
Журнал.Исходящие	113	914 ms	7 s
Журнал.ЛНА	9	**14.6 s**	7 s

Среднее – в чем подвох?

$$\frac{1.7 + 1.7 + 1.7 + 1.7 + 1.7 + 1.7 + 1.7 + 1.7 + 1.7 + 1.7}{10} = 1.7$$

$$\frac{0.1 + 0.1 + 0.1 + 0.1 + 0.1 + 0.1 + 0.1 + 0.1 + 0.2 + 16}{10} = 1.7$$

90% Line – спасает?



- Коллеги, наблюдаются зависания журнала!

Сводка по работе запросов

Operation name	> Limit	> Limit +5s	> Limit +10s	> Limit +20s	Executions	Duration	Limit
Журнал	-	-	-	-	203	193 ms	7 s
Журнал.Внутренние	3% (2)	-	-	1% (1)	67	1.6 s	7 s
Журнал.Входящие	-	-	-	-	72	246 ms	7 s
Журнал.ДД	2% (5)	0% (1)	1% (3)	1% (3)	302	1.7 s	7 s
Журнал.Доверенности	-	-	-	-	34	990 ms	7 s
Журнал.Заявки	4% (2)	2% (1)	-	2% (1)	54	2.6 s	7 s
Журнал.Исходящие	-	-	-	-	113	914 ms	7 s
Журнал.ЛНА	11% (1)	-	33% (3)	11% (1)	9	14.6 s	7 s
Журнал.ОРД	3% (3)	2% (2)	1% (1)	1% (1)	105	2.2 s	7 s
Журнал.Тематики	-	-	-	-	26	70 ms	7 s
Журнал.Удаленные	-	-	-	-	14	172 ms	7 s
Журнал.ФПД	-	0% (1)	2% (4)	1% (3)	231	2.2 s	7 s

Планы выполнения

- Планы всех запросов по операции
- Детальная информация по каждому запросу:
  - Кол-во выполнений
  - Время выполнения
  - Buffer Gets + IO Time + CPU Time

Достаем планы:

```
for i in (select case when sql_text like '%_journal_view%' then
                 case when sql_text like '%incoming_journal_view%'
                   then 2 -- Журнал.Входящие
                   else 1 -- Журнал
                 end category, current_timestamp, t.*
          from v$sql t
          where sql_text like '%_journal_view%') loop
  insert into v_display_cursor
  select i.sql_id, i.child_number, i.plan_hash_value, rownum, plan_table_output
  from table(dbms_xplan.display_cursor(i.sql_id,
                                       i.child_number
                                       +peeked_binds'));
end loop;
```

Статистика работы операций

Operation name	> Limit	> Limit +5s	> Limit +10s	> Limit +20s	Executions	Duration	Limit
Журнал	-	-	-	-	203	193 ms	7 s
Журнал.Внутренние	3% (2)	-	-	1% (1)	67	1.6 s	7 s
Журнал.Входящие	-	-	-	-	72	246 ms	7 s
Журнал.ДД	2% (5)	0% (1)	1% (3)	1% (3)	302	1.7 s	7 s
Журнал.Доверенности	-	-	-	-	34	990 ms	7 s
Журнал.Заявки	4% (2)	2% (1)	-	2% (1)	54	2.6 s	7 s
Журнал.Исходящие	-	-	-	-	113	914 ms	7 s
Журнал.ЛНА	11% (1)	-	33% (3)	11% (1)	9	14.6 s	7 s
Журнал.ОРД	3% (3)	2% (2)	1% (1)	1% (1)	105	2.2 s	7 s
Журнал.Тематики	-	-	-	-	26	70 ms	7 s
Журнал.Удаленные	-	-	-	-	14	172 ms	7 s
Журнал.ФПД	-	0% (1)	2% (4)	1% (3)	231	2.2 s	7 s

Планы выполнения

Журнал.ДД

SQL ID	Snap time	Duration (ms)	Exec	SQL Text	Buffer gets	IO Time (ms)	CPU Time (ms)	Info
2ab7hm2c0s20p		15,973	7	select distinct r_object_id ...	158479	13,789	2,205	plan
fnwawm7f20cb6		3,774	25	select distinct r_object_id ...	45515	2,885	886	plan
41s7kv32xt5pm		2,139	17	select distinct r_object_id ...	3729	1,923	161	plan

```
select *
from v_display_cursor
where category = 5
  and sql_id = '2ab7hm2c0s20p'
  and snap_time between '06.06.2017' and '07.06.2017';
```

План открывается в новой вкладке

about:blank | C:\Users\yakiselev\AppDat... | BrowserMetrix

```
SQL_ID  dnt2fa937j3ac, child number 0
-------------------------------------
select all r_object_id, is_paper_doc, is_urgent, attachment,
attachments_count, doc_kind, reg_number, reg_date, create_date,
content, item_state, signer, performer, receiver, lock_owner from
ecm.internal_journal_view_all doc where ( ((is_mistaken=:"SYS_B_0" OR
is_mistaken IS NULL))  and r_object_id in (select all r_object_id from
ecm.internal_journal_pret_view where (employee_position_id in
(:"SYS_B_1", :"SYS_B_2")) ) and  exists (select *  from
pretreatment_process_users_sp  pret where ((BITAND(pret.user_policy,
doc.weight)=doc.weight) and (pret.document_id=doc.r_object_id) and
pret.employee_position_id in (:"SYS_B_3", :"SYS_B_4")))) order by
reg_date desc

Plan hash value: 2920991145

-----------------------------------------------------------------------------------------------------
| Id  | Operation                                | Name                                | E-Rows |  OMem |  1Mem | Used-Mem |
-----------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT                         |                                     |        |       |       |          |
|   1 |  SORT ORDER BY                           |                                     |      5 |  1024 |  1024 |          |
|   2 |   NESTED LOOPS OUTER                     |                                     |      5 |       |       |          |
|   3 |    NESTED LOOPS OUTER                    |                                     |      3 |       |       |          |
|   4 |     NESTED LOOPS OUTER                   |                                     |      3 |       |       |          |
|   5 |      NESTED LOOPS                        |                                     |      3 |       |       |          |
|   6 |       NESTED LOOPS OUTER                 |                                     |      3 |       |       |          |
|   7 |        NESTED LOOPS SEMI                 |                                     |      3 |       |       |          |
|*  8 |         HASH JOIN                        |                                     |     15 |  715K |  715K | 214K (0) |
```

Peeked binds и предикаты на месте

about:blank C:\Users\yakiselev\AppDat... BrowserMetrix

```
|  30 |     NESTED LOOPS                |                          |      1 |        |        |            |
|  31 |      NESTED LOOPS               |                          |      1 |        |        |            |
|* 32 |       INDEX RANGE SCAN          | IDX_DM_SYSOBJECT_S       |      1 |        |        |            |
|* 33 |       INDEX RANGE SCAN          | IDX_OBJ_DESC_CRC_DICT_CAT|      1 |        |        |            |
|* 34 |      INDEX UNIQUE SCAN          | D_1F0027D980000D1B       |      1 |        |        |            |
-------------------------------------------------------------------------------------------------------------

Peeked Binds (identified by position):
--------------------------------------

   1 - :SYS_B_0 (NUMBER): 0
   2 - :SYS_B_1 (VARCHAR2(30), CSID=873): '080027d981e39ce7'
   3 - :SYS_B_2 (VARCHAR2(30), CSID=873): '080027d98295ad3c'
   4 - (VARCHAR2(30), CSID=873): '080027d981e39ce7'
   5 - (VARCHAR2(30), CSID=873): '080027d98295ad3c'

Predicate Information (identified by operation id):
---------------------------------------------------

   8 - access("DOC"."CROC_DOCUMENT_TYPE_ID"="PERM"."DOC_TYPE_ID")
  14 - access("INTERNAL"."R_OBJECT_ID"="PRET"."DOCUMENT_ID")
  16 - access(("PRET"."EMPLOYEE_POSITION_ID"=:SYS_B_1 OR "PRET"."EMPLOYEE_POSITION_ID"=:SYS_B_2))
  18 - filter(("DOC"."IS_MISTAKEN"=:SYS_B_0 OR "DOC"."IS_MISTAKEN" IS NULL))
  19 - access("DOC"."R_OBJECT_ID"="R_OBJECT_ID")
  20 - access("INTERNAL"."R_OBJECT_ID"="DOC"."R_OBJECT_ID")
  22 - filter((INTERNAL_FUNCTION("UXPB_"."EMPLOYEE_POSITION_ID") AND
              "PERM"."POLICIES TYPE WEIGHT"=BITAND("UXPB_"."USER POLICY","PERM"."POLICIES TYPE WEIGHT")))
```

Статистика выполнения

- Информации из v$sql_monitor достаточно:
  - Время начала запроса
  - Длительность
  - Параметры выполнения
  - Использование Buffer Gets + CPU + IO

Сохраняем SQL Monitor

```
for i in (select case when sql_text like '%_journal_view%' then
                 case when sql_text like '%incoming_journal_view%'
                   then 2 -- Журнал.Входящие
                   else 1 -- Журнал
                 end category, current_timestamp, t.*
          from v$sql t
          where sql_text like '%_journal_view%') loop
  insert into v_sql_monitor
  select v.*, dbms_sqltune.report_sql_monitor(sql_id,sql_exec_id, ...)
  from v$sql_monitor v
  where v.sql_id = i.sql_id
    and v.status like 'DONE%'
    and not exists (select 1 from v_sql_monitor t
                    where t.sql_id = v.sql_id
                      and t.sql_exec_start = v.sql_exec_start
                      and t.sql_exec_id = v.sql_exec_id));
end loop;
```

SQL Monitor

SQL ID	Snap time	Duration (ms)	Exec	SQL Text	Buffer gets	IO Time (ms)	CPU Time (ms)	Info
2ab7hm2c0s20p		15,973	7	select distinct r_object_id ...	158479	13,789	2,205	plan
	12:49:17	68,373	1	...	184218	65,881	2,804	monitor
	08:11:50	24,934	1	...	491381	22,501	2,488	monitor
	12:47:55	6,729	1	...	21748	6,312	398	monitor
fnwawm7f20cb6		3,774	25	select distinct r_object_id ...	45515	2,885	886	plan
	09:42:08	36,344	1	...	159855	33,785	2,695	monitor
	15:05:14	18,120	1	...	160074	16,464	1,846	monitor
	10:41:33	15,782	1	...	159897	14,089	1,570	monitor
	13:59:48	4,312	1	...	45657	3,728	587	monitor
	15:57:53	4,074	1	...	160078	2,579	1,281	monitor
41s7kv32xt5pm		2,139	17	select distinct r_object_id ...	3729	1,923	161	plan
	09:20:57	32,741	1	...	56504	29,462	2,421	monitor

SQL Monitor в новой вкладке


```
Global Information
------------------------------
 Status                :  DONE (FIRST N ROWS)
 Instance ID           :  1
 Session               :  ECM (815:16807)
 SQL ID                :  dnt2fa937j3ac
 SQL Execution ID      :  16777241
 Execution Started     :  06/10/2016 08:32:03
 First Refresh Time    :  06/10/2016 08:32:07
 Last Refresh Time     :  06/10/2016 08:32:37
 Duration              :  34s
 Module/Action         :  documentum@s001as-ecmcs02
 Service               :  ecmdb
 Program               :  documentum@s001as-ecmcs02
 Fetch Calls           :  2

Binds
========================================================================================
|    Name    | Position |      Type      |                                  Value
========================================================================================
| :SYS_B_0   |        1 | NUMBER         | 0
| :SYS_B_1   |        2 | VARCHAR2(32)   | 080027d9810a9653
| :SYS_B_2   |        3 | VARCHAR2(32)   | 080027d984211a4f
```

Статистика выполнения запроса

about:blank | CROC Monitoring Tool | C:\Users\yakiselev\Downlo... | BrowserMetrix

Id	Operation	Name	Rows (Estim)	Cost	Time Active(s)	Start Active	Execs	Rows (Actual)	Read Reqs	Read Bytes	Mem (Max
0	SELECT STATEMENT				31	+4	1	240			
1	SORT ORDER BY		4	22	31	+4	1	240			886
2	NESTED LOOPS OUTER		4	21	31	+4	1	1748			
3	NESTED LOOPS OUTER		2	15	31	+4	1	1748			
4	NESTED LOOPS OUTER		2	14	31	+4	1	1748			
5	NESTED LOOPS		2	13	31	+4	1	1748			
6	NESTED LOOPS OUTER		2	12	31	+4	1	1749			
7	NESTED LOOPS SEMI		2	11	31	+4	1	1749			
8	NESTED LOOPS		15	10	31	+4	1	1749			
9	NESTED LOOPS		30	8	31	+4	1	1749			
10	NESTED LOOPS		30	7	31	+4	1	1749			
11	VIEW	VW_NSO_1	30	6	31	+4	1	1749			
12	HASH UNIQUE		30	6	31	+4	1	1749			1
13	VIEW	INTERNAL_JOURNAL_PRET_VIEW	30	5	1	+4	1	1783			
14	HASH JOIN		6164	5	1	+4	1	1783			2
15	INLIST ITERATOR				1	+4	1	6980			
16	INDEX RANGE SCAN	IDX_PRET_PROCESS_USERS_E_D	6164	1	1	+4	2	6980			
17	INDEX FULL SCAN	D_1F0027D980000990	10841	3	1	+4	1	23413			
18	TABLE ACCESS BY INDEX ROWID	CROC_DOCUMENT_COMMON_S	1	1	34	+1	1749	1749	713	6MB	
19	INDEX UNIQUE SCAN	D_1F0027D980000982	1	1	31	+4	1749	1749	21	168KB	
20	INDEX UNIQUE SCAN	D_1F0027D980000990	1	1	31	+4	1749	1749			
21	TABLE ACCESS BY INDEX ROWID	CROC_PERMISSION_POLICIES_S	1	1	31	+4	1749	1749			
22	INDEX UNIQUE SCAN	IDX_PERMISSION_POLICIES_DT	1	1	31	+4	1749	1749			
23	TABLE ACCESS BY INDEX ROWID	PRETREATMENT_PROCESS_USERS_S	1027	1	31	+4	1749	1749	4378	34MB	
24	INDEX RANGE SCAN	IDX_2_PRETR_PROCESS_USERS_S	26	1	35	+0	1749	10202	1900	15MB	
25	TABLE ACCESS BY INDEX ROWID	PRETREATMENT_INBOX_S	1	1	31	+4	1749	1749	988	8MB	
26	INDEX RANGE SCAN	IDX_GA_PRETR_INBOX_SP_DOCID	1	1	31	+4	1749	1749	243	2MB	
27	INDEX RANGE SCAN	IDX_CROC_LIFECYCLE_POLICY_RFST	1	1	32	+3	1749	1748	827	6MB	
28	INDEX RANGE SCAN	IDX_CRC_DICT_VALUE_S	1	1	31	+4	1748	1748			
29	INDEX RANGE SCAN	IDX_CRC_DICT_VALUE_S	1	1	31	+4	1748	1746			
30	VIEW PUSHED PREDICATE	CRC_DICT_DOCKIND_NAME_VIEW	1	3	31	+4	1748	1748			
31	NESTED LOOPS		1	3	31	+4	1748	1748			

Удобство использования

- Можно снять отчет за любой период

Система тормозила с 11:00 до 12:00

```
spool perf_report.html
select *
from table(croc_perf_info.get_report_html(
                    '06.06.2017 11:00:00',
                    '06.06.2017 12:00:00'));
```

perf_report.html

Удобство использования

- Можно запросить отчет за период
- Отчет каждое утро с отчетом за день

Утро начинается не с кофе

```
select to_char(sysdate, 'YYYYMMDD_HH24MI') cur_date
from dual;

spool performance_info_&cur_date..html
select *
from table(croc_perf_info.get_report_html(sysdate - 1,
                                          sysdate));
```

cron -> zip -> email

Текущие проблемы

- Сборщик данных

Как оно крутится:

- Сборщик каждые 10 минут находит:
  - До 1300 строк в v$sql (120К строк в день)
  - 12 SQL Monitor
  - Длится от 3 сек ночью до 20 сек днем

Текущие проблемы

- Сборщик данных
  - Используем небольшие shared pool (< 10GB)
  - Получение SQL Monitor тормозит

Сохраняем SQL Monitor

```
select v.*,
   dbms_sqltune.report_sql_monitor(
      sql_id, sql_exec_start, sql_exec_id)
from v$sql_monitor v
where v.sql_id = i.sql_id
   and v.status like 'DONE%'
   and not exists (select 1 from v_sql_monitor t
                   where t.sql_id = v.sql_id
                   and t.key = v.key));
```

Похожие исключаем

```
SELECT v.*,
coalesce((select sql_id||'_'||sql_exec_start||'_'||sql_exec_id
        from v_sql_monitor i
        where i.sql_id = v.sql_id
          and i.last_refresh_time > v.last_refresh_time - 1
          and i.elapsed_time
                between v.elapsed_time - v.elapsed_time*0.1
                    and v.elapsed_time + v.elapsed_time*0.1
          and rownum = 1),
        (select dbms_sqltune.report_sql_monitor(
                    v.sql_id, v.sql_exec_start, v.sql_exec_id)))
FROM v$sql_monitor v
...
```

Текущие проблемы

- Сборщик данных
  - Используем небольшие shared pool (< 10GB)
  - Получение SQL Monitor тормозит
- Скорость генерации отчета
  - Без секционирования - грустно
  - Приходится удалять старые данные

Старые данные подчищаем

```
dbms_scheduler.create_job (
  job_name        =>  CROC_PERF_INFO_CLEANUP,
  job_action      => 'begin
       delete from v_sql where snap_id <
              (select snap_id
               from (select snap_id, snap_time
                     from croc_perf_snapshot
                     where snap_time < sysdate - 7
                     order by snap_id desc)
                where rownum = 1);
            commit;
          end;',
  repeat_interval => FREQ=DAILY;BYHOUR=23);
```

Удаляет 120К строк за 2 минуты

Итоги

- Используется у заказчиков
- Отчетами пользуются менеджеры
- Категории регулярно пополняются
- Экономит время

Выводы

- Oracle DB – *хорошо диагностируемая* СУБД
- Доверяй, но проверяй
- Велосипеды – не всегда плохо

Q&A

Киселев Ярослав
yaki@croc.ru