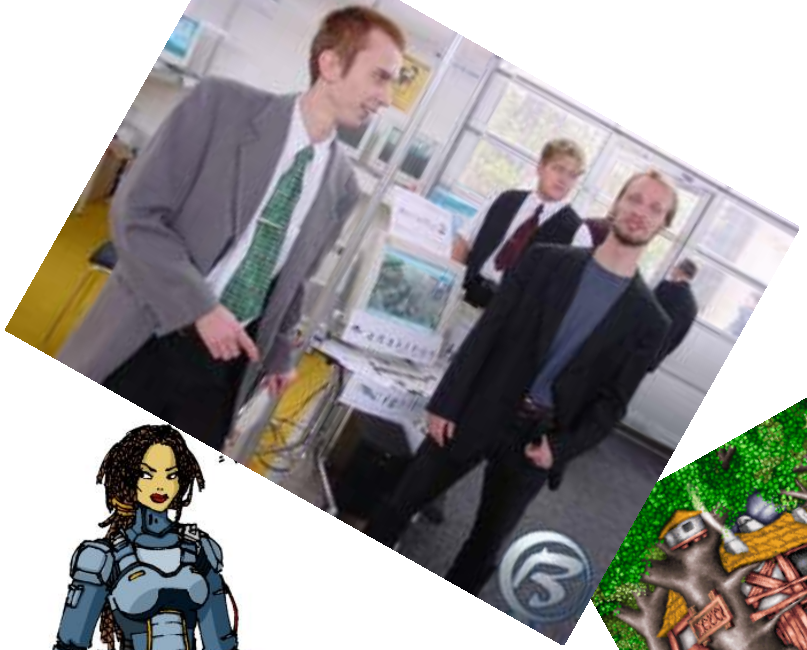
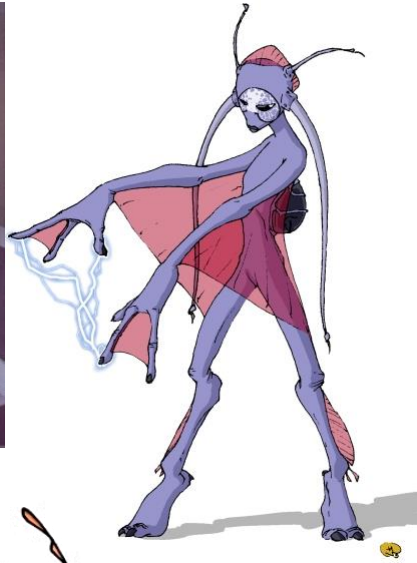
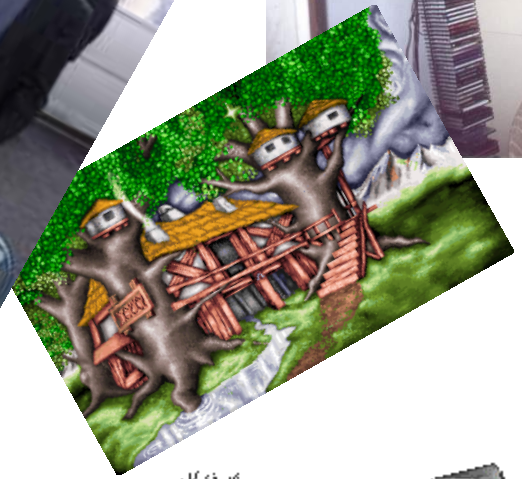
Think simple and spare yourself a facepalm

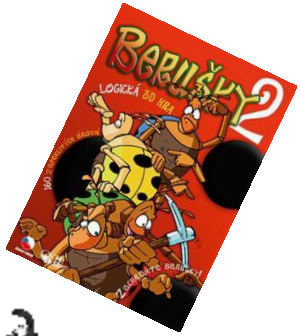
Michal Simonik

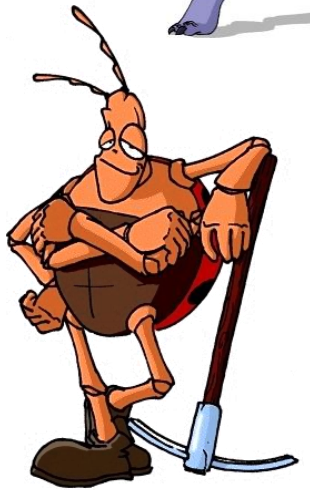
@michalsimonik
misimonik@gmail.com
http://www.michalsimonik.com











www.anakreongames.com

About me

Independent consultant

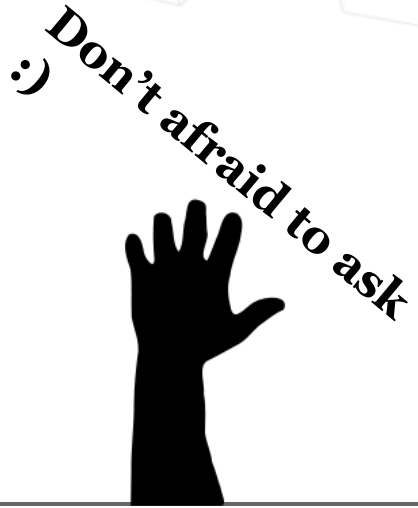
+11 years in IT
+10 years with Oracle

Database Architect
Data Modeling and Tuning
SQL Tuning
Database Troubleshooting
Consulting

ORACLE
Certified Professional
Oracle Database 11*g*
Administrator

ORACLE
Certified Expert
Oracle Database SQL

ORACLE
Certified Specialist



Few things before we start …

- Know your enemy
  - Format the code fist
  - It might be good to paint a picture
  - What are the table sizes?
  - What are their relations
  - What about selectivity?
  - What indexes do I have at my disposal?
  - Check database statistics!
    - 50% of time you're done after that

Few things before we start …

- Golden rules
  - Eliminate as soon as possible as much as possible
  - Most important questions are
    - Where do I start?
    - Which table I take next
  - Be precise
    - You don’t want to read database blocks just to throw them away
      - If it’s possible - sometimes brute force is necessary

Few things before we start …

- What do we tune?
  - Resources
    - CPU
    - I/O
    - Memory (can manifest as I/O)
    - Network
  - Locks
  - Restrictions based on application design

How to measure

- How to measure
  - Compare what is comparable - **time is just a good hint!**
    - Logical reads
    - Memory
  - Compare under same conditions
    - Data
    - Buffer cache
    - Shared pool
    - Beware of different environments
      - Test vs Production

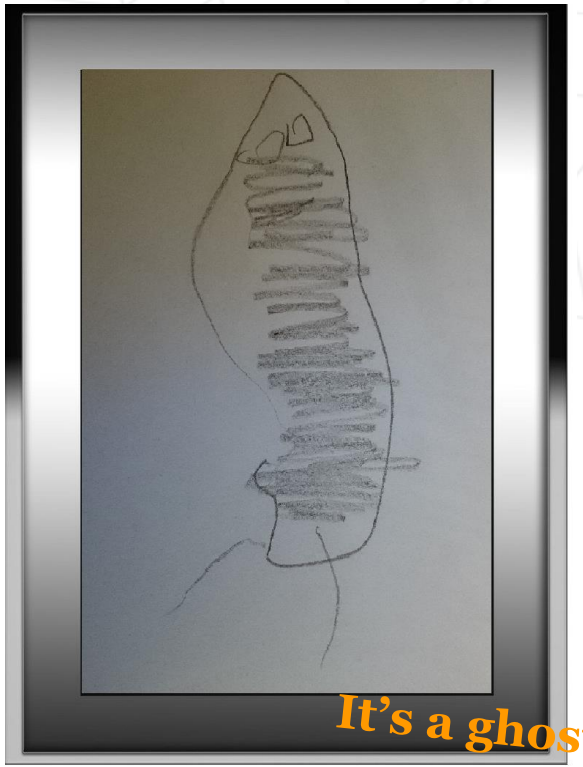
How to test

1. ALTER SYSTEM FLUSH SHARED_POOL;
   a. *EXEC DBMS_SHARED_POOL.PURGE('adress, hash','C');*
2. ALTER SYSTEM FLUSH BUFFER_CACHE;
   a. *DBMS_RESULT_CACHE.FLUSH;*
3. Run old SQL with GATHER_PLAN_STATISTICS hint - *Fetch whole query!*
4. ALTER SYSTEM FLUSH BUFFER_CACHE;
   a. *DBMS_RESULT_CACHE.FLUSH;*
5. Run new SQL with GATHER_PLAN_STATISTICS hint - *Fetch whole query!*
6. Get your SQL IDs
7. SELECT * FROM TABLE(DBMS_XPLAN.DISPLAY_CURSOR(*sql_id*, *child#*, FORMAT => 'ALLSTATS'));
8. **Compare**

SQL Art

- What is difficult for you is difficult for Oracle
  - 6 table join is 720 ways how to join them (not including indexes)
  - Overcomplicated SQL
  - Views

- You know your data best
  - You should also know what Oracle can do with them



Something to warm you up ...

```
SELECT *
FROM cards c,
  order_items i
WHERE c.id       = i.card_id
AND i.ITEM_PRICE > 1.20 *
  (SELECT AVG(a.item_price) FROM order_items a WHERE a.card_id = c.id);
```

Something to warm you up …

```
SELECT *
FROM cards c,
  order_items i
WHERE c.id        = i.card_id
AND i.ITEM_PRICE > 1.20 *
  (SELECT AVG(a.item_price) FROM order_items a WHERE a.card_id = c.id);

SELECT *
FROM cards c,
  order_items i,
  (SELECT card_id,
    AVG(item_price) avg_item_price
  FROM order_items
  GROUP BY card_id
  ) a
WHERE c.id        = i.card_id
AND i.card_id     = a.card_id
AND i.ITEM_PRICE > 1.20 * a.avg_item_price;
```

That's a good MERGE, right?

```
MERGE INTO metadata_tab met USING
( SELECT DISTINCT zp,
  obd_from_rb,
  obd_to_rb,
  obd_from_rz,
  obd_to_rz
FROM fakta_tab
WHERE module_code = 'P304'
  AND period      = 201209
) fak ON (    NVL(fak.zp, 0)          = NVL(met.zp, 0)
          AND NVL(fak.obd_from_rb, 0) = NVL(met.obd_from_rb, 0)
          AND NVL(fak.obd_to_rb, 0)   = NVL(met.obd_to_rb, 0)
          AND NVL(fak.obd_from_rz, 0) = NVL(met.obd_from_rz, 0)
          AND NVL(fak.obd_to_rz, 0)   = NVL(met.obd_to_rz, 0))
WHEN NOT MATCHED THEN
  INSERT
    (
      id_meta_ident,
      zp,
      obd_from_rb,
      obd_to_rb,
      obd_from_rz,
      obd_to_rz
    )
    VALUES
    (
      metadata_seq.NEXTVAL+100,
      fak.zp,
      fak.obd_od_rb,
      fak.obd_do_rb,
      fak.obd_od_rz,
      fak.obd_do_rz
    );
```

That’s a good MERGE, right?

```
-------------------------------------------------------------------------------------------------------------
| Id  | Operation                      | Name             | Starts | E-Rows | A-Rows | Buffers | Reads  |
-------------------------------------------------------------------------------------------------------------
|   0 | MERGE STATEMENT                |                  |      1 |        |      0 |     252 |      4 |
|   1 |  MERGE                         | METADATA_TAB     |      1 |        |      0 |     252 |      4 |
|   2 |   VIEW                         |                  |      1 |        |     15 |     252 |      4 |
|   3 |    SEQUENCE                    | METADATA_SEQ     |      1 |        |     15 |     252 |      4 |
|*  4 |     HASH JOIN OUTER BUFFERED   |                  |      4 |      4 |     15 |       0 |      0 |
|   5 |      VIEW                      |                  |      4 |      4 |     15 |       0 |      0 |
|   6 |       HASH UNIQUE              |                  |      4 |      4 |     15 |       0 |      0 |
|   7 |        HASH UNIQUE             |                  |      4 |      4 |     59 |   15912 |  10332 |
|*  8 |         TABLE ACCESS FULL      | FAKTA_TAB        |     62 |   2480K|   2544K|   15912 |  10332 |
|*  9 |       TABLE ACCESS FULL        | METADATA_TAB     |      5 |    166 |    180 |      10 |      5 |
-------------------------------------------------------------------------------------------------------------

* Optimal memory for HASH - 774k
```

That's a good MERGE, right?

```
MERGE INTO metadata_tab met USING
( SELECT DISTINCT zp,
  obd_from_rb,
  obd_to_rb,
  obd_from_rz,
  obd_to_rz
FROM fakta_tab
WHERE module_code = 'P304'
  AND period      = 201209
) fak ON (    COALESCE(fak.zp, 0)           = COALESCE(met.zp, 0)
          AND COALESCE(fak.obd_from_rb, 0) = COALESCE(met.obd_from_rb, 0)
          AND COALESCE(fak.obd_to_rb, 0)   = COALESCE(met.obd_to_rb, 0)
          AND COALESCE(fak.obd_from_rz, 0) = COALESCE(met.obd_from_rz, 0)
          AND COALESCE(fak.obd_to_rz, 0)   = COALESCE(met.obd_to_rz, 0))
WHEN NOT MATCHED THEN
  INSERT
    (
      id_meta_ident,
      zp,
      obd_from_rb,
      obd_to_rb,
      obd_from_rz,
      obd_to_rz
    )
    VALUES
    (
      metadata_seq.NEXTVAL+100,
      fak.zp,
      fak.obd_od_rb,
      fak.obd_do_rb,
      fak.obd_od_rz,
      fak.obd_do_rz
    );
```

That's a good MERGE, right?

```
DECLARE
 v_null     VARCHAR2(32) := NULL;
 v_not_null VARCHAR2(32) := 'X';
 v_dummy    VARCHAR2(32);
BEGIN
 FOR l_idx IN 1 .. 1000000
 LOOP
  v_dummy := NVL(v_null, RAWTOHEX(SYS_GUID()));
 END LOOP;
 --
 FOR l_idx IN 1 .. 1000000
 LOOP
  v_dummy := NVL(v_not_null, RAWTOHEX(SYS_GUID()));
 END LOOP;
END;
/
```

- NVL

```
Time NULL:          71.66
Time NOT NULL:      72.39
```

That's a good MERGE, right?

```
DECLARE
 v_null     VARCHAR2(32) := NULL;
 v_not_null VARCHAR2(32) := 'X';
 v_dummy    VARCHAR2(32);
BEGIN
 FOR l_idx IN 1 .. 1000000
 LOOP
  v_dummy := NVL(v_null, RAWTOHEX(SYS_GUID()));
 END LOOP;
 --
 FOR l_idx IN 1 .. 1000000
 LOOP
  v_dummy := NVL(v_not_null, RAWTOHEX(SYS_GUID()));
 END LOOP;
END;
/
```

- NVL

```
Time NULL:          71.66
Time NOT NULL:      72.39
```

- COALESCE

```
Time NULL:          74.54
Time NOT NULL:        .08
```

That's a good MERGE, right?

```
INSERT INTO metadata_tab
SELECT metadata_seq.NEXTVAL,
       sel.*
FROM
  (SELECT DISTINCT zp_zisku,
    obd_od_rb,
    obd_do_rb,
    obd_od_rz,
    obd_do_rz
  FROM fakta_tab
  WHERE kod_uloha = 'P304'
    AND obdobi    = 201209
  MINUS
  SELECT zp_zisku,
    obd_od_rb,
    obd_do_rb,
    obd_od_rz,
    obd_do_rz
  FROM metadata_tab
  ) sel;
```

That’s a good MERGE, right?

```
-------------------------------------------------------------------------------------------------------------
| Id  | Operation                    | Name             | Starts | E-Rows | A-Rows | Buffers | Reads  |
-------------------------------------------------------------------------------------------------------------
|   0 | INSERT STATEMENT             |                  |      1 |        |      0 |     186 |      1 |
|   1 |  LOAD TABLE CONVENTIONAL     |                  |      1 |        |      0 |     186 |      1 |
|   2 |   SEQUENCE                   | METADATA_SEQ     |      1 |        |      0 |     186 |      1 |
|   3 |    VIEW                      |                  |      4 |      4 |      0 |       0 |      0 |
|   4 |     MINUS                    |                  |      4 |        |      0 |       0 |      0 |
|   5 |      SORT UNIQUE             |                  |      4 |      4 |     15 |       0 |      0 |
|*  6 |       TABLE ACCESS FULL      | FAKTA_TAB        |     62 |   2480K|   2544K|   15912 |  10332 |
|   7 |      SORT UNIQUE             |                  |      4 |    166 |    180 |       0 |      0 |
|*  8 |       TABLE ACCESS FULL      | METADATA_SEQ     |      5 |    166 |    180 |      10 |      5 |
-------------------------------------------------------------------------------------------------------------

* Optimal memory for SORT- 2k
```

That's a good MERGE, right?

- From Oracle 11g Release 2
- Ignores ORA-00001
- Requires unique index
  - ORA-38912 if you pass wrong arguments

```
INSERT /*+ ignore_row_on_dupkey_index(metadata_tab, metadata_tab_pk) */ INTO metadata_tab
SELECT metadata_seq.NEXTVAL,
       sel.*
FROM
  (SELECT DISTINCT zp_zisku,
    obd_od_rb,
    obd_do_rb,
    obd_od_rz,
    obd_do_rz
  FROM fakta_tab
  WHERE kod_uloha = 'P304'
    AND obdobi    = 201209
  ) sel;
```

SELECT from SELECT from SELECT ... and from SELECT

- Goal is to identify multiplicities
- Present them in form of table

ID	Patient ID	Data	ID of duplicate
1	FWER241	1.1.2015	5
1	FWER241	1.1.2015	11
1	FWER241	1.1.2015	23
2	33GSW52	4.5.2015	632
2	33GSW52	4.5.2015	745

SELECT from SELECT from SELECT ... and from SELECT

```
SELECT id_uni, patient_id_uni, procedure_date_uni, id_dpl
FROM
  (SELECT uni.procedure_date procedure_date_uni,
          uni.patient_id patient_id_uni,
          uni.id id_uni,
          dpl.procedure_date procedure_date_dpl,
          dpl.patient_id patient_id_dpl,
          dpl.id id_dpl
   FROM
    (SELECT master.*
     FROM patient_data master
     WHERE EXISTS
      (SELECT 1
       FROM
       (SELECT * FROM patient_data) source
        WHERE master.procedure_date = source.procedure_date
          AND master.patient_id     = source.patient_id
          AND master.ROWID          > source.ROWID
        )
      ) dpl,
      (SELECT *
       FROM patient_data
       WHERE ROWID IN (SELECT ID_ROW
                       FROM
                       (SELECT MIN(ROWID) AS ID_ROW, procedure_date,patient_id
                        FROM patient_data
                        GROUP BY procedure_date,
                                 patient_id
                       )
                      )
      ) uni
   WHERE dpl.procedure_date  = uni.procedure_date
     AND dpl.patient_id      = uni.patient_id
  ) unirowid,
  patient_data master_data
WHERE unirowid.patient_id_uni     = master_data.patient_id
  AND unirowid.procedure_date_uni = master_data.procedure_date
  AND unirowid.id_uni             = master_data.id
ORDER BY unirowid.patient_id_uni,
         unirowid.procedure_date_uni,
         unirowid.id_dpl;
```

Michal Simonik

Think simple and spare yourself a facepalm

SELECT from SELECT from SELECT ... and from SELECT

```
-------------------------------------------------------------------------------------------------
| Id  | Operation                      | Name         | E-Rows |  OMem |  1Mem |  O/1/M    |
-------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT               |              |        |       |       |           |
|   1 |  SORT ORDER BY                 |              |      1 |  2048 |  2048 |     1/0/0|
|   2 |   NESTED LOOPS SEMI            |              |      1 |       |       |           |
|*  3 |    HASH JOIN                   |              |      1 |   992K|   992K|     1/0/0|
|*  4 |     HASH JOIN                  |              |      1 |  1368K|  1368K|     1/0/0|
|   5 |      NESTED LOOPS              |              |      1 |       |       |           |
|   6 |       VIEW                     | VW_NSO_1     |     10 |       |       |           |
|   7 |        HASH GROUP BY           |              |     10 |  1186K|  1186K|     1/0/0|
|   8 |         TABLE ACCESS FULL      | PATIENT_DATA |     10 |       |       |           |
|   9 |       TABLE ACCESS BY USER ROWID| PATIENT_DATA |      1 |       |       |           |
|  10 |      TABLE ACCESS FULL         | PATIENT_DATA |     10 |       |       |           |
|  11 |     TABLE ACCESS FULL          | PATIENT_DATA |     10 |       |       |           |
|* 12 |    TABLE ACCESS BY ROWID RANGE | PATIENT_DATA |      1 |       |       |           |
-------------------------------------------------------------------------------------------------
```

SELECT from SELECT from SELECT ... and from SELECT

```
WITH tmp AS
  (SELECT
    /*+ materialize or result_cache */
    *
  FROM
    (SELECT m.patient_id,
      m.procedure_date,
      m.id,
      ROW_NUMBER() OVER (PARTITION BY m.patient_id, m.procedure_date ORDER BY m.patient_id, m.procedure_date)
c,
      COUNT(*) OVER (PARTITION BY m.patient_id, m.procedure_date) ct
    FROM patient_data m
    )
  WHERE ct > 1
  )
SELECT uni.id,
  uni.patient_id,
  uni.procedure_date,
  dpl.id
FROM tmp dpl,
     tmp uni
WHERE dpl.patient_id   = uni.patient_id
AND dpl.procedure_date = uni.procedure_date
AND uni.c              = 1
AND dpl.c              > 1;
```

SELECT from SELECT from SELECT … and from SELECT

```
-------------------------------------------------------------------------------------------------------
| Id  | Operation                  | Name                          | E-Rows |  OMem |  1Mem |  O/1/M   |
-------------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT           |                               |        |       |       |          |
|   1 |  TEMP TABLE TRANSFORMATION |                               |        |       |       |          |
|   2 |   LOAD AS SELECT           |                               |        |  1024 |  1024 |     1/0/0|
|*  3 |    VIEW                    |                               |      1 |       |       |          |
|   4 |     WINDOW SORT            |                               |      1 |  1024 |  1024 |     1/0/0|
|   5 |      TABLE ACCESS FULL     | PATIENT_DATA                  |      1 |       |       |          |
|*  6 |   HASH JOIN                |                               |      1 |   814K|   814K|     1/0/0|
|*  7 |    VIEW                    |                               |      1 |       |       |          |
|   8 |     TABLE ACCESS FULL      | SYS_TEMP_0FD9FC85C_9DE45319   |      1 |       |       |          |
|*  9 |    VIEW                    |                               |      1 |       |       |          |
|  10 |     TABLE ACCESS FULL      | SYS_TEMP_0FD9FC85C_9DE45319   |      1 |       |       |          |
-------------------------------------------------------------------------------------------------------
```

Copy and paste INSERT

```
PROCEDURE ...
  ...
  seq NUMBER;
BEGIN
  ...
  INSERT INTO dm_owner.ms_cs2vd501011_tmp
      (SELECT * FROM csu2011.cs2vd501011@vm15dbv_ora112i WHERE stat_date = p_stat_date_in);
  ...
  FOR sel IN (SELECT * FROM dm_owner.cs2vd501011_tmp)
  LOOP
    SELECT dmv_s_statobj.NEXTVAL INTO seq FROM dual;

    INSERT INTO dmv_t_statobj (id_statobj, … , module)
      VALUES (seq, … , p_module_in);
    INSERT INTO dmv_t_msr (kode_p, … , id_fak_meta_neident)
      VALUES ('ISEKTOR', … ,seq , 1);
    INSERT INTO dmv_t_msr (kode_p, … , id_fak_meta_neident)
      VALUES ('FORMA', … ,seq , 1);
    INSERT INTO dmv_t_msr (kode_p, … , id_fak_meta_neident)
      VALUES ('KATP', … ,seq , 1);
    INSERT INTO dmv_t_msr (kode_p, … , id_fak_meta_neident)
      VALUES ('MNACE', … ,seq , 1);
    INSERT INTO dmv_t_msr (kode_p, … , id_fak_meta_neident)
      VALUES ('POCP', … ,seq , 1);
  END LOOP;
  ...
END;
```

Copy and paste INSERT

```
PROCEDURE ...
  ...
  seq NUMBER;
BEGIN
  ...
  INSERT INTO dm_owner.ms_cs2vd501011_tmp
    (SELECT z, statjed, isektor, forma, katp, mnace, pocp, NULL seq FROM
csu2011.cs2vd501011@vm15dbv_ora112i WHERE stat_date = p_stat_date_in);
  ...
  INSERT ALL
    INTO dmv_t_statobj (id_statobj, … , module)
      VALUES (seq, … , p_module_in);
    INTO dmv_t_msr (kode_p, … , id_fak_meta_neident)
      VALUES ('ISEKTOR', … ,seq , 1);
    INTO dmv_t_msr (kode_p, … , id_fak_meta_neident)
      VALUES ('FORMA', … ,seq , 1);
    INTO dmv_t_msr (kode_p, … , id_fak_meta_neident)
      VALUES ('KATP', … ,seq , 1);
    INTO dmv_t_msr (kode_p, … , id_fak_meta_neident)
      VALUES ('MNACE', … ,seq , 1);
    INTO dmv_t_msr (kode_p, … , id_fak_meta_neident)
      VALUES ('POCP', … ,seq , 1)
  SELECT * FROM dm_owner.cs2vd501011_tmp;
  ...
END;
```

DML trigger for sequence generation

I have to SELECT to see!

```
PROCEDURE delete_order (p_id orders.id%TYPE) IS
BEGIN
  dbms_output.put_line('Deleting Order ID: '||p_id);

  FOR l_idx IN (SELECT * FROM mtg.order_items WHERE order_id = p_id)
  LOOP
    dbms_output.put_line('Deleting Item ID: '||l_idx.id);
  END LOOP;

  DELETE mtg.order_items WHERE order_id = p_id;
END;
```

I have to SELECT to see!

```
PROCEDURE delete_order (p_id orders.id%TYPE) IS
  TYPE t_id IS TABLE OF mtg.order_items.id%TYPE INDEX BY PLS_INTEGER;
  l_id t_id;
BEGIN
  dbms_output.put_line('Deleting Order ID: '||p_id);

  DELETE mtg.order_items WHERE order_id = p_id RETURNING id BULK COLLECT INTO l_id;

  FOR l_idx IN l_id.FIRST .. l_id.LAST
  LOOP
    dbms_output.put_line('Deleting Item ID: '||l_id(l_idx));
  END LOOP;
END;
```

I have found #@$!

SQL ordered by Executions

- %CPU - CPU Time as a percentage of Elapsed Time
- %IO - User I/O Time as a percentage of Elapsed Time
- Total Executions: 2,071,438,395
- Captured SQL account for 63.2% of Total

Executions	Rows Processed	Rows per Exec	Elapsed Time (s)	%CPU	%IO	SQL Id
195,864,188	3,530,984	0.02	5,529.75	99	0	80arg30ynd8qd
153,244,574	153,235,946	1.00	51,497.41	19,6	81	18xfs8w6hjsxp
134,343,651	134,338,794	1.00	25,021.35	60,4	39,4	0b6gk19a9a8pf
106,596,676	106,593,769	1.00	1,028,317.81	2,2	98,4	ghumaf11674ft
101,310,855	73,099,529	0.72	2,900.45	99	0	1fjj2h8xjf4n4
101,015,990	249,329,542	2.47	623,710.41	21,3	79,1	aa5qc5f96m2x5
95,757,817	94,166,235	0.98	2,729.54	99	,7	gszxc6dar1ft2
83,304,554	53,339,413	0.64	190,648.23	2,7	98	16t0406brthwk
46,936,143	43,595,198	0.93	1,322.04	98,4	0	dd6h51y9q7r80
39,589,652	39,585,208	1.00	22,199.04	7,4	93,3	23uvfwvkt6738
37,261,236	51,086,993	1.37	112,368.15	5,2	95,3	cua1tbj0y0h50
36,117,130	208,259,098	5.77	50,346.40	96,2	,9	gd3ncrwjhrq3y

I have found #@$!

SQL ordered by Executions

- %CPU - CPU Time as a percentage of Elapsed Time
- %IO - User I/O Time as a percentage of Elapsed Time
- Total Executions: 2,071,438,395
- Captured SQL account for 63.2% of Total

Executions	Rows Processed	Rows per Exec	Elapsed Time (s)	%CPU	%IO	SQL Id
195,864,188	3,530,984	0.02	5,529.75	99	0	80arg30ynd8qd
153,244,574	153,235,946	1.00	51,497.41	19,6	81	18xfs8w6hjsxp
134,343,651	134,338,794	1.00	25,021.35	60,4	39,4	0b6gk19a9a8pf
106,596,676	106,593,769	1.00	1,028,317.81	2,2	98,4	ghumaf11674ft
101,310,855	73,099,529	0.72	2,900.45	99	0	1fjj2h8xjf4n4
101,015,990	249,329,542	2.47	623,710.41	21,3	79,1	aa5qc5f96m2x5
95,757,817	94,166,235	0.98	2,729.54	99	,7	gszxc6dar1ft2
83,304,554	53,339,413	0.64	190,648.23	2,7	98	16t0406brthwk
46,936,143	43,595,198	0.93	1,322.04	98,4	0	dd6h51y9q7r80
39,589,652	39,585,208	1.00	22,199.04	7,4	93,3	23uvfwvkt6738
37,261,236	51,086,993	1.37	112,368.15	5,2	95,3	cua1tbj0y0h50
36,117,130	208,259,098	5.77	50,346.40	96,2	,9	gd3ncrwjhrq3y

Ultimate facepalm

```
SELECT company,
       COUNT(*)
FROM invoices
WHERE can_access( company ) = 1
GROUP BY company;
```

```
-------------------------------------------------------------------------------------------------
| Id  | Operation               | Name              | Starts | A-Rows | Buffers |  OMem |  1Mem |
-------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT        |                   |      2 |   2587 |    2402 |       |       |
|   1 |  VIEW                   |                   |      2 |   2587 |    2402 |       |       |
|   2 |   HASH GROUP BY         |                   |      2 |   2587 |    2402 |    17M|  3187K|
|   3 |    PARTITION RANGE ALL  |                   |      2 |    586K|    2402 |       |       |
|*  4 |     INDEX FAST FULL SCAN| INVOICES          |     82 |    586K|    2402 |       |       |
-------------------------------------------------------------------------------------------------
```

Ultimate facepalm

```
SELECT * FROM
            (SELECT /*+ no_merge */
                        company,
                        COUNT(*)
            FROM invoices
            GROUP BY company)
WHERE (SELECT can_access( company ) FROM DUAL) = 1;
```

```
-------------------------------------------------------------------------------------------------------
| Id  | Operation                | Name              | Starts | A-Rows | Buffers |  OMem |  1Mem |
-------------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT         |                   |      1 |   2537 |    1201 |       |       |
|*  1 |  FILTER                  |                   |      1 |   2537 |    1201 |       |       |
|   2 |   VIEW                   |                   |      1 |   2537 |    1201 |       |       |
|   3 |    HASH GROUP BY         |                   |      1 |   2537 |    1201 |   17M|  3187K|
|   4 |     PARTITION RANGE ALL  |                   |      1 |    293K|    1201 |       |       |
|   5 |      INDEX FAST FULL SCAN| INVOICES          |     41 |    293K|    1201 |       |       |
|   6 |   FAST DUAL              |                   |   2537 |   2537 |       0 |       |       |
-------------------------------------------------------------------------------------------------------
```

Ultimate facepalm

```
SELECT company,
       COUNT(*)
FROM invoices
GROUP BY company
HAVING can_access(company) = 1;
```

```
-------------------------------------------------------------------------------------------------
| Id  | Operation               | Name              | Starts | A-Rows | Buffers |  OMem |  1Mem |
-------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT        |                   |      1 |   2537 |    1201 |       |       |
|*  1 |  FILTER                 |                   |      1 |   2537 |    1201 |       |       |
|   2 |   HASH GROUP BY         |                   |      1 |   2537 |    1201 |    17M|  3187K|
|   3 |    PARTITION RANGE ALL  |                   |      1 |    293K|    1201 |       |       |
|   4 |     INDEX FAST FULL SCAN| INVOICES          |     41 |    293K|    1201 |       |       |
-------------------------------------------------------------------------------------------------
```

Thank you for your attention

Q&A

@michalsimonik
misimonik@gmail.com
http://www.michalsimonik.com