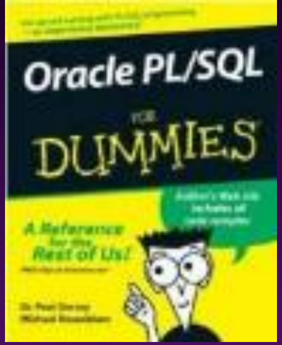
Server-Side Development for the Cloud

Michael Rosenblum

www.dulcian.com

Who Am I? – “Misha”

- Oracle ACE
- Co-author of 3 books
  - *PL/SQL for Dummies*
  - *Expert PL/SQL Practices*
  - *Oracle PL/SQL Performance Tuning Tips & Techniques*



- Known for:
  - SQL and PL/SQL tuning
  - Complex functionality
    - Code generators
    - Repository-based development

Yet another cloud presentation?!

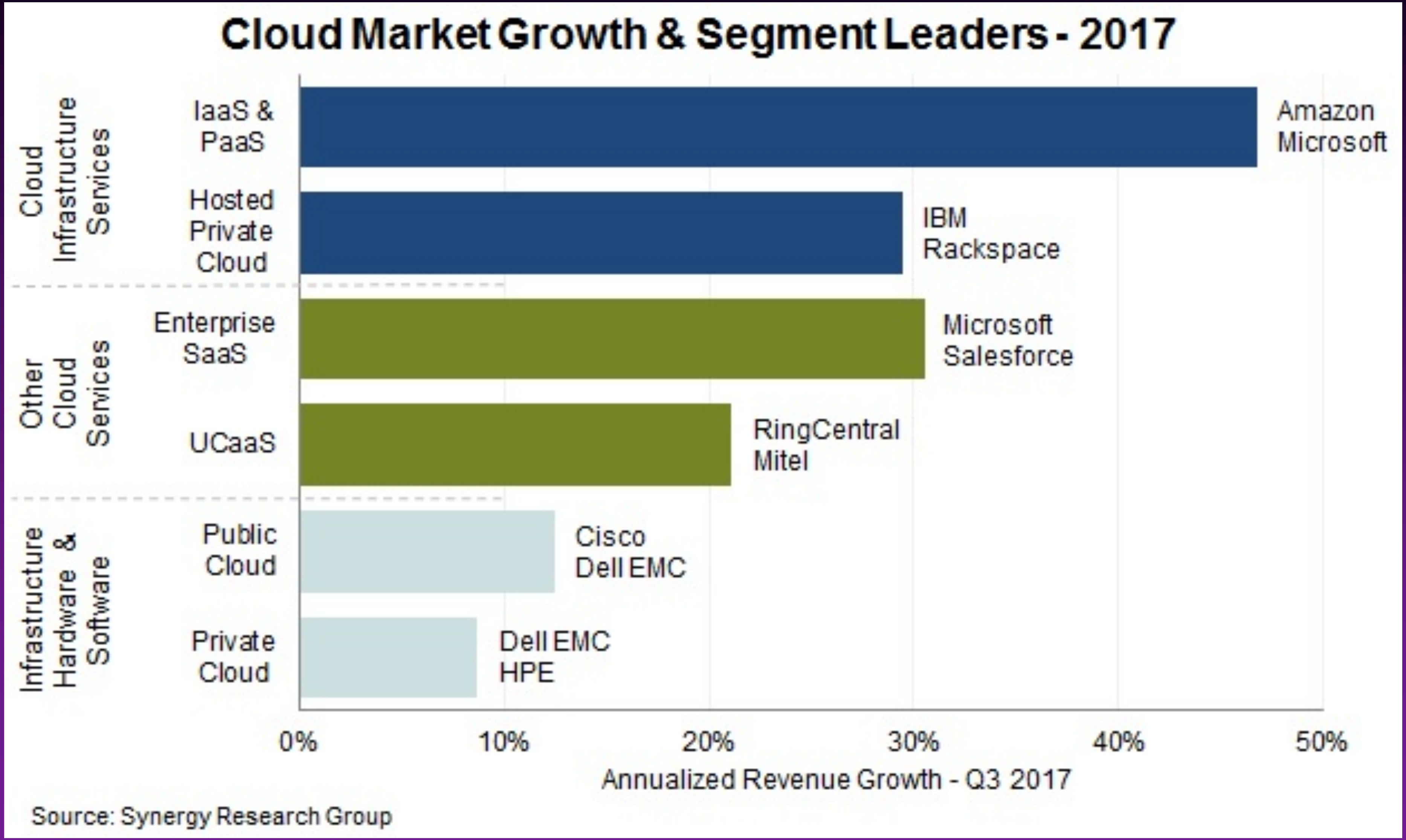
- NO, because:
  - I have been building actual systems for the last two decades.
  - I have hosted systems both in the cloud and on-premises.
- Also, beware:
  - I don’t work for Oracle/Amazon/IBM/etc.
    - …so, I WILL use the right of Free Speech, guaranteed by the FIRST amendment ☺

Parts of the Equation

- Cloud
  - … i.e. what is the environment?
- Server-side
  - … i.e. what is the architecture?
- Development
  - … i.e. what is the implementation?

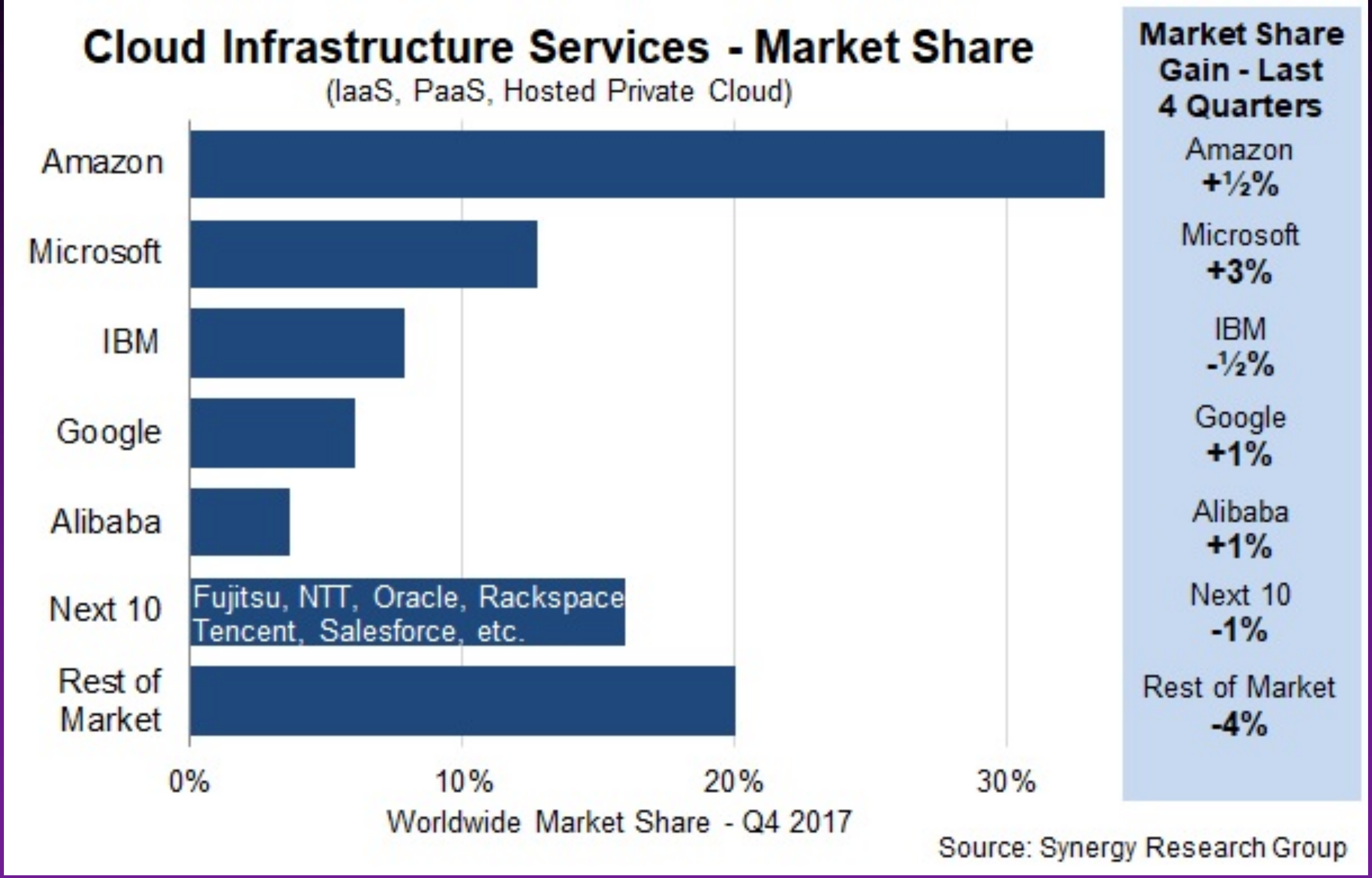
I. State of the Cloud

It's Growing!



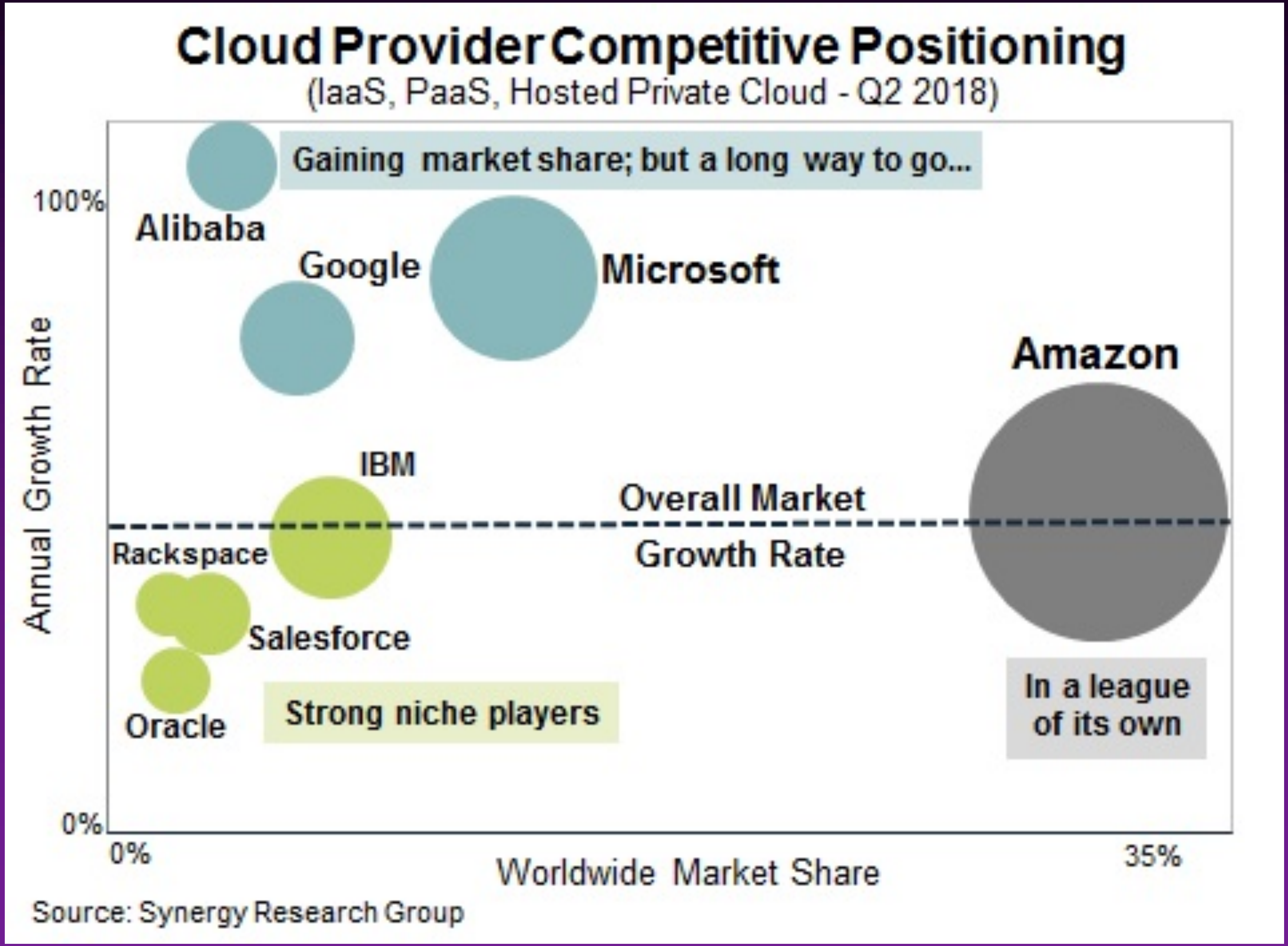
Source: Synergy Research Group

It’s about infrastructure



Source: Synergy Research Group

It’s … well… Amazon



Source: Synergy Research Group

Observations

- IaaS is the fastest growing segment because…
  - … companies don’t want to lose control over their environments
  - … it is the most flexible one

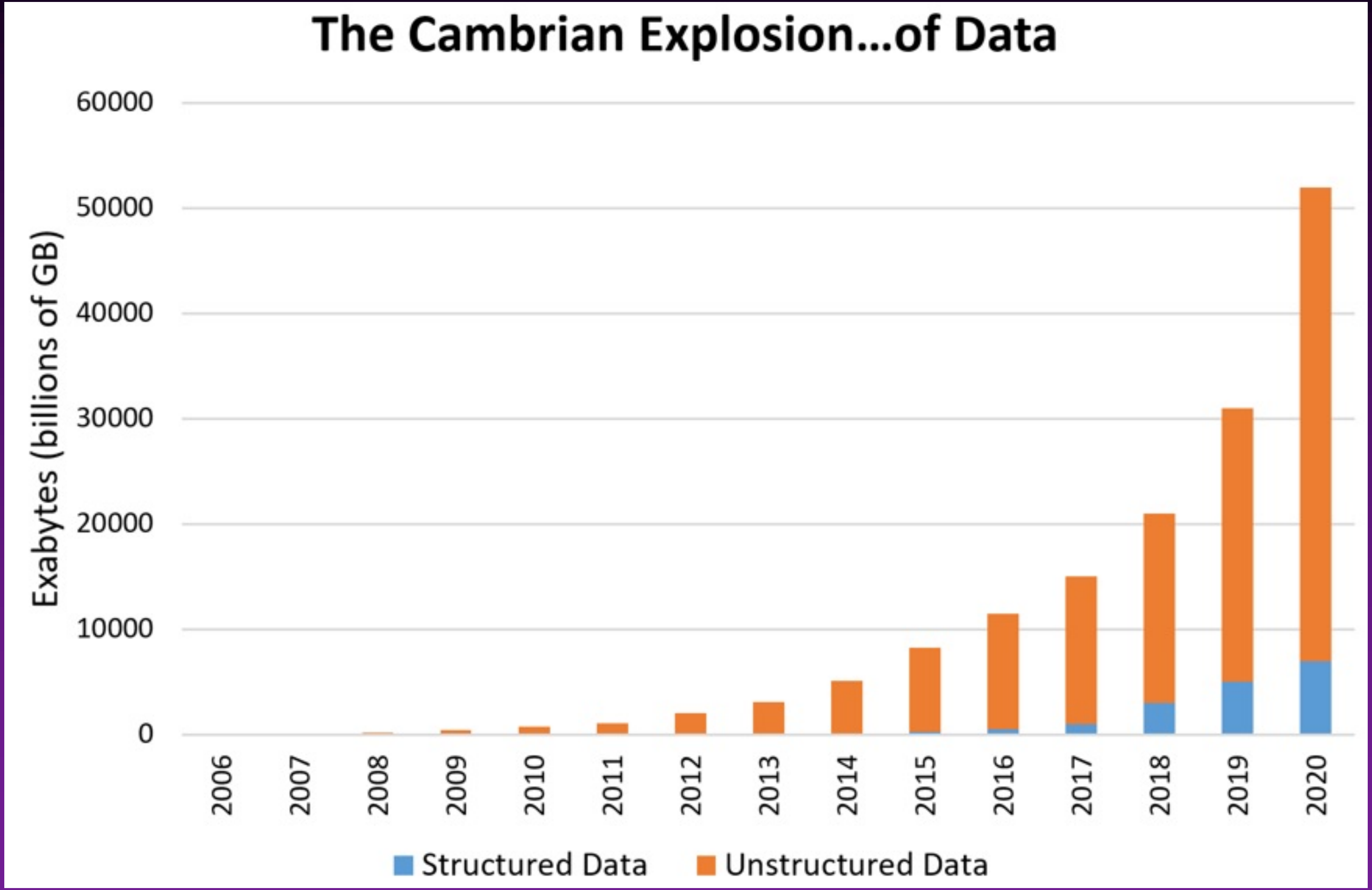
- PaaS (including DBaaS) is growing, but not as fast as promised
  - … because it is really hard to do everything properly
  - ... so, providers have to add restrictions.

II. State of Server-Side Development

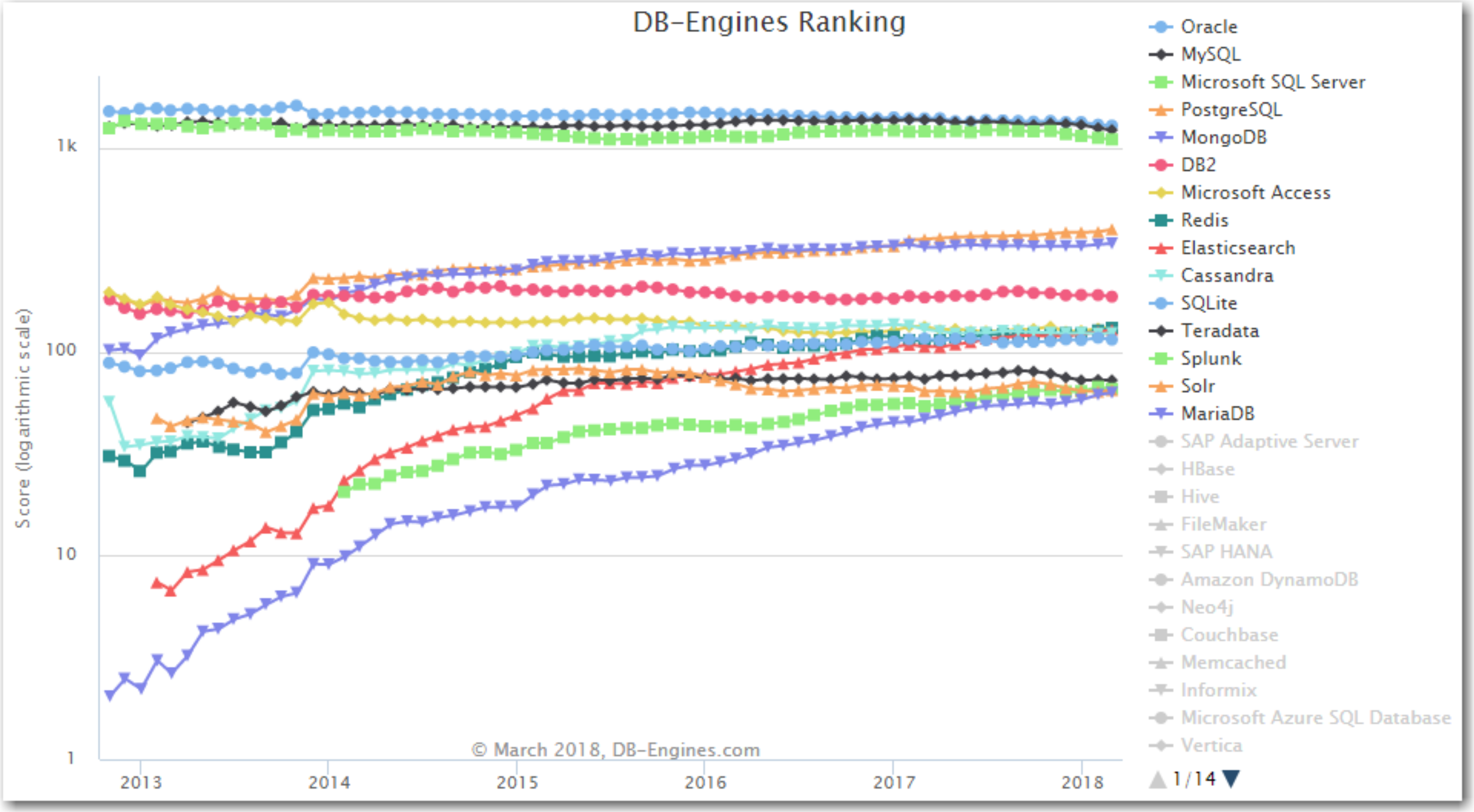
Why Server-Side?

- SQL is the most efficient method of data manipulation.
- PL/SQL is the most efficient way of encapsulating SQL into procedural logic.
- Roundtrips between database and middle-tier are still the most wide-spread performance killers (after missing bind variables ☺)
- “Thick Database” (aka “smart database”) just WORKS!

Data is growing!



Databases are the same



Problem detected!

- Growth of data + old technologies ➔
  - means more pressure on the same solution patterns
  - … which means critical resources become limited faster
  - … which means design mistakes become obvious
  - … which means looking for ~~scapegoat~~s quick-fixes ☹

III. Problem vs. Opportunity

Cloud?!

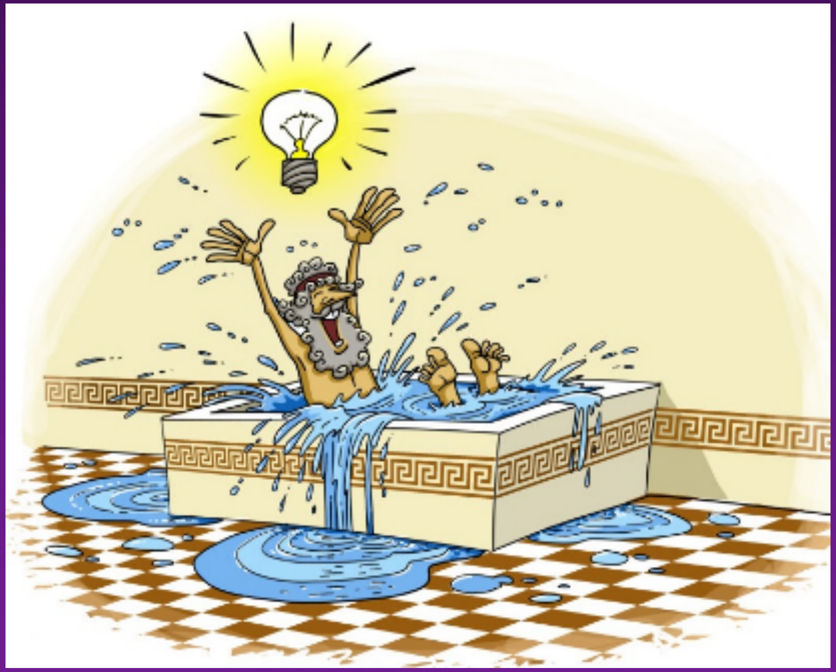
- Resource utilization is easily monitored by providers.
- Hardware resources are no longer static.
- Expense model is “pay-per-use.”

Cloud!!!

- Resource elasticity works both ways!
  - Solving problems by adding resources → Spending money
  - Solving problems by optimizing systems → Saving money

Eureka!

- Total quality of the code base (including tuning efforts) has a DIRECT cost impact!



Impact

- Political:
  - “Good vs bad” can be easily quantified → at least some objectivity in decision-making
  - Good architecture pays off → good architects are being nurtured and supported by the management
  - Performance tuning is back in style → quality control
  - Solutions usually cross boundaries → DBAs and developers are forced to work together
- Technical:
  - Developers are constantly reminded about resource utilization → less sloppy code

IV. Top-Down View

Core Elements

- 1. Detect problem areas → Code instrumentation
- 2. Pinpoint exact location → Profiling

IV.1 - Logging

System Logging

- Levels of information:
  - Core info
    - Process
    - Session
  - Granular info
    - Client
    - Module
    - Action
- Why bother?
  - StateLESS implementation spawns logical session between multiple physical sessions.

Setting Granular Info (1)

```
-- Client Stuff
Begin
  -- set it to anything you want to describe the session.
  -- Otherwise useless
  DBMS_APPLICATION_INFO.SET_CLIENT_INFO
    ('This is my test-run');

  -- Key setting for debugging!
  -- This ID is traceable.
  DBMS_SESSION.SET_IDENTIFIER ('misha01');
end;
/

-- Visibility:
select sid, client_info, client_identifier
from v$session
```

Setting Granular Info (2)

```
-- Client Stuff
Begin
  -- Additional info: module and action
  DBMS_APPLICATION_INFO.SET_MODULE
           (module_name=>'HR',
            action_name=>'SALARY_MAINT');
end;
/

-- Visibility:
select sid, module, action
from v$session
```

Application Logging

- Advantages:
  - Customized information when needed
- Disadvantages:
  - Requires discipline of the whole development group
- Key technologies
  - Autonomous transactions
  - Conditional compilation

Indestructible Log

```
create or replace package log_pkg
is
    procedure p_log (i_tx varchar2);
    procedure p_log (i_cl CLOB);
end;
/
create or replace package body log_pkg is
    procedure p_log (i_tx varchar2) is
        pragma autonomous_transaction;
    begin
      insert into t_log (id_nr, timestamp_dt, log_tx, log_cl)
      values (log_seq.nextval, systimestamp,
          case when length(i_tx)<=4000 then i_tx else null end,
          case when length(i_tx)>4000 then i_tx else null end);
      commit;
    end;

    procedure p_log (i_cl CLOB) is
        pragma autonomous_transaction;
    begin
        insert into t_log (id_nr, timestamp_dt,log_cl)
        values (log_seq.nextval, systimestamp,i_cl);
        commit;
    end;
end;
/
```

Conditional Compilation

```
create or replace procedure p_conditional
is
    v_tx varchar2(256);
begin
  $if $$DebugTF $then
    log_pkg.p_log
       ('Before query:'||dbms_utility.format_call_stack);
  $end

  select ename
  into v_tx
  from scott.emp;

  $if $$DebugTF $then
        log_pkg.p_log ('After query');
  $end
exception
  when others then
     log_pkg.p_log(dbms_utility.format_error_stack);
     log_pkg.p_log
                  (dbms_utility.format_error_backtrace);
     raise;
end;
```

IV.2 - Profiling

PL/SQL Hierarchical Profiler

- Gathers hierarchical statistics of all calls (both SQL and PL/SQL) for the duration of the monitoring
  - … into a portable trace file
- Has powerful aggregation utilities
  - … both within the database and using a command-line interface
- Available since Oracle 11.1 [replaced PL/SQL Profiler]
  - … and constantly improved even in 18c

Intro (1)

```
SQL> CREATE DIRECTORY IO AS 'C:\IO';
SQL> exec dbms_hprof.start_profiling
                      (location=>'IO',filename=>'HProf.txt');

SQL> DECLARE
2       PROCEDURE p_doSomething (pi_empno NUMBER) IS
3       BEGIN
4           dbms_lock.sleep(0.1);
5       END;
6       PROCEDURE p_main IS
7       BEGIN
8           dbms_lock.sleep(0.5);
9           FOR c IN (SELECT * FROM emp) LOOP
10              p_doSomething(c.empno);
11          END LOOP;
12      END;
13  BEGIN
14      p_main();
15  END;
16  /
SQL> exec dbms_hprof.stop_profiling;
```

Destination folder:
WRITE is enough

Spend time

Intro (2)

- Raw file (C:\IO\HProf.txt) is not very readable…

```
P#V PLSHPROF Internal Version 1.0
P#! PL/SQL Timer Started
P#C PLSQL."".""."__plsql_vm"
P#X 8
P#C PLSQL."".""."__anonymous_block"
P#X 6
P#C PLSQL."".""."__anonymous_block.P_MAIN"#980980e97e42f8ec #6
P#X 63
P#C PLSQL."SYS"."DBMS_LOCK"::9."__pkg_init"
P#X 7
P#R
P#X 119
P#C PLSQL."SYS"."DBMS_LOCK"::11."SLEEP"#e17d780a3c3eae3d #197
P#X 500373
P#R
P#X 586
P#C SQL."".""."__sql_fetch_line9" #9."4ay6mhcbhvbf2"
P#! SELECT * FROM SCOTT.EMP
P#X 3791
P#R
P#X 17
<<… and so on …>>
```

Call

Elapsed time between events

Return from sub-program

Intro (3)

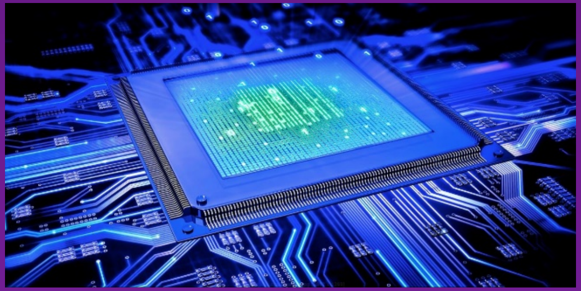
- … but you can and make it readable via the command-line utility:

```
C:\Utl_File\IO>plshprof -output hprof_intro HProf.txt
PLSHPROF: Oracle Database 12c Enterprise Edition Release 12.2.0.1.0
  - 64bit Production
[8 symbols processed]
[Report written to 'hprof_intro.html']
```

V. Down to Earth

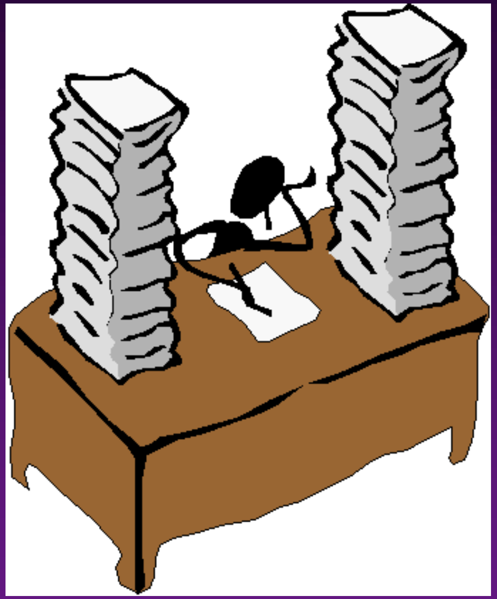
It's all about CPU now!

- Shift to cloud → going from I/O-bound to CPU-bound:
  - On-premises servers usually had CPUs over-allocated:
    - Storage is upgradable and scalable / CPU is not
    - Servers have to support the highest workload (Black Friday!)
  - Cloud storage usually means SSD
    - low latency → much faster I/O → no longer a bottleneck

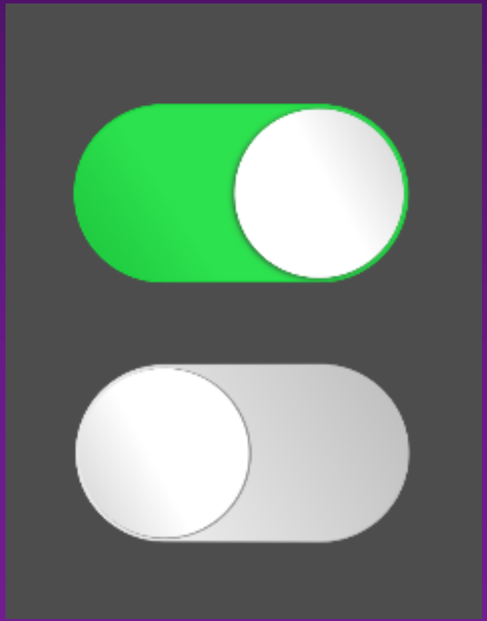


Ways to Lower CPU Workload

- 1. Avoid context switches
- 2. Don’t reinvent the wheel
- 3. Don‘t do things multiple times
- 4. Work in SETs



V.1 - Avoid Context Switches



Ways to Solve the Problem

- Decrease frequency:
  - Help CBO to fire PL/SQL functions in SQL less often
- Decrease the cost:
  - PRAGMA UDF
  - Functions in WITH clause

Counting Function Calls

- **Output:**

```
SQL> SELECT empno, ename, f_change_nr(empno) change_nr
  2  FROM emp
  3  WHERE f_change_nr(empno) IS NOT NULL
  4  AND deptno = 20;
...
5 rows selected.
SQL> exec counter_pkg.p_check;
Fired:10
```

Twice the number of returned rows

- **Explanation:**
  - CBO orders predicates to decrease the total cost
    - DEPNO=20 is applied first to get 5 rows back
    - CBO needs correct info (statistics, indexes, constraints etc.) to make that decision
  - The same functions in SELECT and WHERE clauses are being fired independently.

PRAGMA UDF (1)

- Meaning:
  - PL/SQL function is compiled in the way that is optimized for SQL usage (different memory management).
- Example:

```
CREATE FUNCTION f_change_udf_nr (i_nr NUMBER)
RETURN NUMBER IS
    PRAGMA UDF;
BEGIN
    counter_pkg.v_nr:=counter_pkg.v_nr+1;
    RETURN i_nr+1;
END;
```

PRAGMA UDF (2)

- Much faster in SQL:

```
SQL> SELECT MAX(f_change_nr(object_id)) FROM TEST_TAB;
MAX(F_CHANGE_NR(OBJECT_ID))
---------------------------
                      51485
Elapsed: 00:00:00.48

SQL> SELECT MAX(f_change_udf_nr(object_id)) FROM TEST_TAB;
MAX(F_CHANGE_UDF_NR(OBJECT_ID))
-------------------------------
                          51485
Elapsed: 00:00:00.06
```

Functions in WITH Clause

- Meaning:
  - Functions with the visibility scope of just one SQL query
  - Compilation is tightly linked with SQL

```
SQL> WITH FUNCTION f_changeWith_nr (i_nr number)
  2    RETURN NUMBER IS
  3        BEGIN
  4           RETURN i_nr+1;
  5        END;
  6   SELECT max(f_changeWith_nr(object_id))
  7   FROM test_tab
  8   /
MAX(F_CHANGEWITH_NR(OBJECT_ID))
-------------------------------
                          51485
Elapsed: 00:00:00.07
```

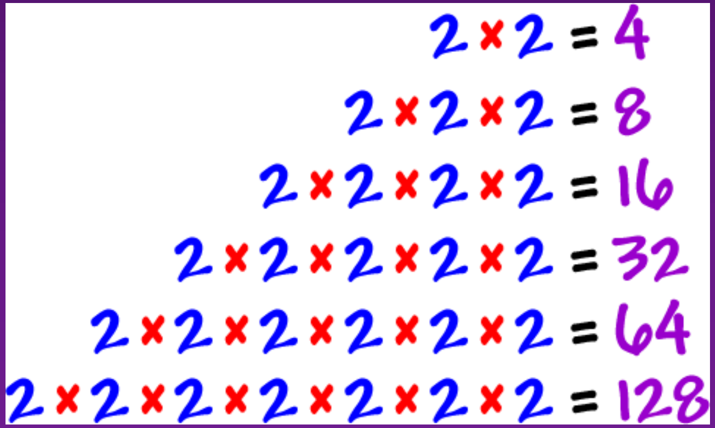
Comparable to
PRAGMA UDF timing

V.2 - Don’t reinvent the wheel

These features are FREE!!!

- Just a reminder about:
  - Analytic functions (RANK, LEAD, LAG…)
  - Pivot/Unpivot
  - MODEL
  - JSON and XML support
  - Etc.…

V.3 - Don’t do things multiple times



$2 \times 2 = 4$

$2 \times 2 \times 2 = 8$

$2 \times 2 \times 2 \times 2 = 16$

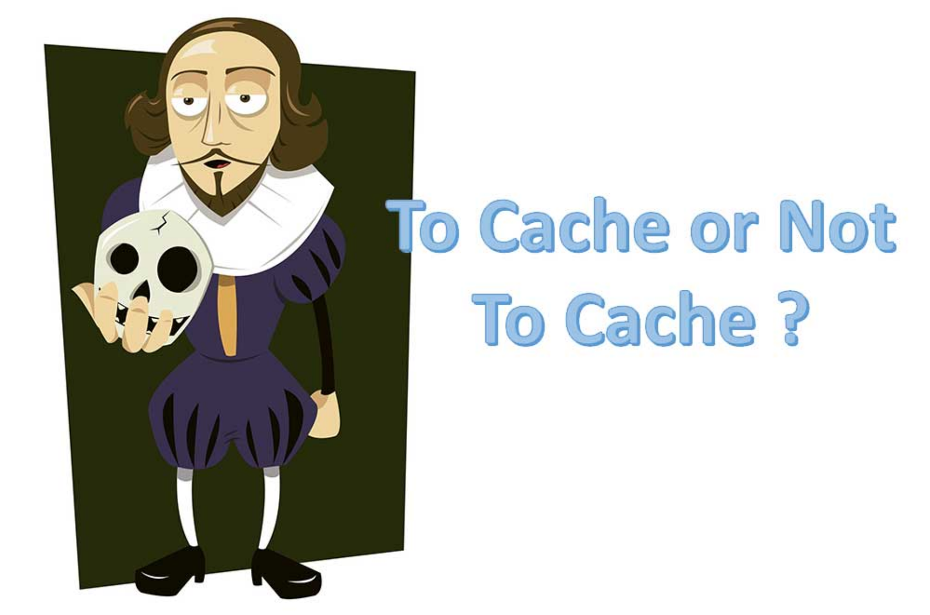
$2 \times 2 \times 2 \times 2 \times 2 = 32$

$2 \times 2 \times 2 \times 2 \times 2 \times 2 = 64$

$2 \times 2 \times 2 \times 2 \times 2 \times 2 \times 2 = 128$

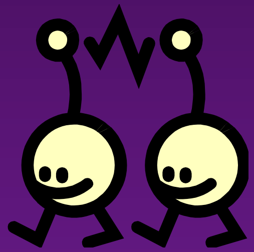
Caching Techniques

- Query-level:
  - Scalar sub-query caching
  - DETERMINISTIC clause
- Database-level
  - PL/SQL function Result Cache



Side Effect of SELECT from DUAL

- Definitions:
  - Scalar sub-query returns a single column of a single row (or from the empty rowset)
  - Scalar sub-query caching is an Oracle internal mechanism to preserve results of such queries while processing more complex ones.
    - Implemented as in-memory hash table
    - Cache size is defined by "*_query_execution_cache_max_size*" [65536 bytes by default]
    - Cache is preserved for the duration of the query.
    - Last value is preserved even if hash table is full.
- Example:



```
SQL> SELECT empno, f_change_tx(job) FROM emp;
SQL> exec counter_pkg.p_check;
Fired:14
SQL> SELECT empno, (SELECT f_change_tx(job) FROM dual)
  2    FROM emp;
SQL> exec counter_pkg.p_check;
Fired:5
```

Only 5 distinct jobs

Same OUT for the same IN

- DETERMINISTIC clause:
  - If function does the same thing for exactly the same IN, it can be defined as DETERMINISTIC.
  - Oracle may reuse already calculated values via in-memory hash tables [same as for scalar sub-query and using the same parameter/limit]
  - Oracle does not check to see whether the function is deterministic or not!
- Example:

```
CREATE FUNCTION f_change_det_tx (i_tx VARCHAR2) RETURN VARCHAR2
DETERMINISTIC IS
BEGIN
    counter_pkg.v_nr:=counter_pkg.v_nr+1;
    RETURN lower(i_tx);
END;

SQL> select empno, f_change_tx(job) from emp;
SQL> exec counter_pkg.p_check;
Fired:14
SQL> select empno, f_change_det_tx(job) from emp;
SQL> exec counter_pkg.p_check;
Fired:5
```

Only 5 distinct jobs

PL/SQL Result Cache

- PL/SQL Function Result Cache
  - Database-level cache (cross-session)
  - Stored in SGA
  - Automatic cache monitoring and invalidation (Oracle 11g R2+)
- Example:

```
create function f_getdept_dsp (i_deptno number)
return varchar2 result_cache is
    v_out_tx varchar2(256);
begin
    if i_deptno is null then return null; end if;
    select initcap(dname) into v_out_tx
    from dept where deptno=i_deptno;
    counter_pkg.v_nr:=counter_pkg.v_nr+1;
    return v_out_tx;
end;
```

Result Cache Basics

```
SQL> SELECT empno, f_getDept_dsp(deptno) dept_dsp
  2  FROM emp;
     EMPNO  DEPT_DSP
----------- -----------------------
      7369  Research
       ...
14 rows selected.
SQL> exec counter_pkg.p_check;
Fired:3

SQL> SELECT empno, f_getDept_dsp(deptno) dept_dsp
   2 FROM emp;
     EMPNO DEPT_DSP
---------- ----------
      7369 Research
       ...
14 rows selected.
SQL> exec counter_pkg.p_check;
Fired:0
```

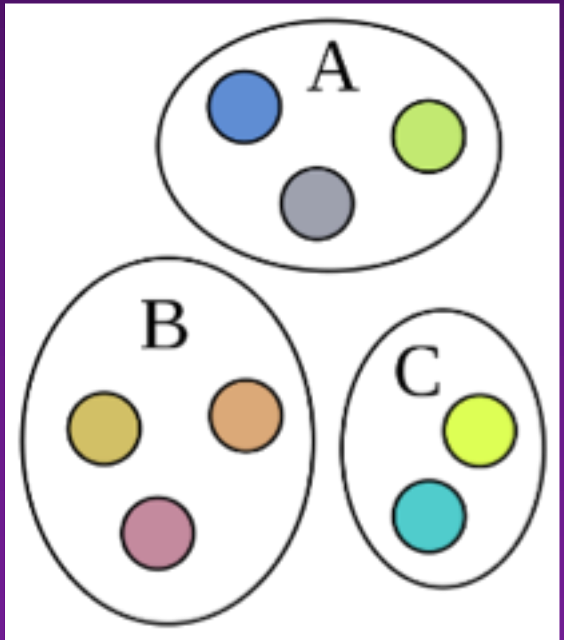
No calls at all!

Result Cache Stats

```
SQL> SELECT * FROM v$result_cache_statistics;
   ID NAME                           VALUE
----- ------------------------------ ----------
    1 Block Size (Bytes)             1024
    2 Block Count Maximum            15360
    3 Block Count Current            32
    4 Result Size Maximum (Blocks)   768
    5 Create Count Success           3
    6 Create Count Failure           0
    7 Find Count                     25
    8 Invalidation Count             0
    9 Delete Count Invalid           0
   10 Delete Count Valid             0
   11 Hash Chain Length              1
   12 Find Copy Count                25
```

3 distinct INs

V.4 - Work in SETs



Object Types!

- Must use and understand object types
  - … and be aware of memory impact
- Read in SETs/write in SETs
  - BULK COLLECT
  - FORALL

Bulk Operations Use-Case

- Task:
  - Data needs to be retrieved from a remote location via DBLink.
  - Each row must be processed locally.
  - Source table contains 50,000 rows.
- Problem:
  - Analyze different ways of achieving the goal.
  - Create best practices.

RowByRow

```
SQL> connect scott/TIGER@localDB;
sql> declare
  2       v_nr number;
  3   begin
  4       for c in  (select * from test_tab@remotedb) loop
  5           v_nr :=c.object_id;
  6       end loop;
  7   end;
  8   /
Elapsed: 00:00:00.77

SQL> select name, value from stats where name in
  2      ('STAT...session pga memory max',
  3       'STAT...SQL*Net roundtrips to/from dblink');
NAME                                        VALUE
----------------------------------------- -------
STAT...session pga memory max             2609504
STAT...SQL*Net roundtrips to/from dblink      510
```

BULK LIMIT

Limit can variable

```
sql> declare
  2      type collection_tt is table of
  3               test_tab@remotedb%rowtype;
  4      v_tt collection_tt;
  5      v_nr number;
  6      v_cur sys_refcursor;
  7      v_limit_nr binary_integer:=5000;
  8  begin
  9      open v_cur for select * from test_tab@remotedb;
 10      loop
 11          fetch v_cur bulk collect into v_tt
 12              limit v_limit_nr;
 13          exit when v_tt.count()=0;
 14          for i in v_tt.first..v_tt.last loop
 15              v_nr:=v_tt(i).object_id;
 16          end loop;
 17          exit when v_tt.count<v_limit_nr;
 18      end loop;
 19      close v_cur;
 20  end;
 21  /
```

Analysis

- Results:

Limit size	Time	Max PGA	Roundtrips
100	0.78	2’543’968	510
250	0.58	2’675’040	210
500	0.49	2’806’112	110
1000	0.44	3’133’792	60
5000	0.40	4’247’904	20
10000	0.41	7’590’240	15
20000	0.43	14’340’448	12

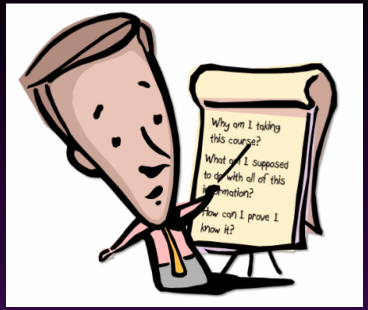
- Summary:
  - With the increase of bulk limit processing, time stops dropping because memory management becomes costly!
  - This point is different for different hardware/software
- Conclusion:
  - Run your own tests and find the most efficient bulk limit

FORALL

- FORALL command
  - The idea:
    - Apply the same action for all elements in the collection.
    - Have only one context switch between SQL and PL/SQL
  - Risks:
    - Special care is required if only some actions from the set succeeded

```
declare
    type number_nt is table of NUMBER;
    v_deptNo_nt number_nt:=number_nt(10,20);
begin
    forall i in v_deptNo_nt.first()..v_deptNo_nt.last()
      update emp
        set sal=sal+10
      where deptNo=v_deptNo_nt(i);
end;
```

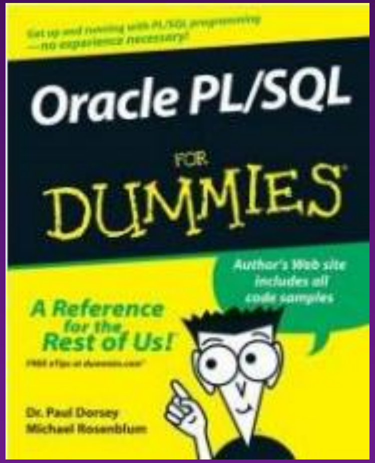
Summary

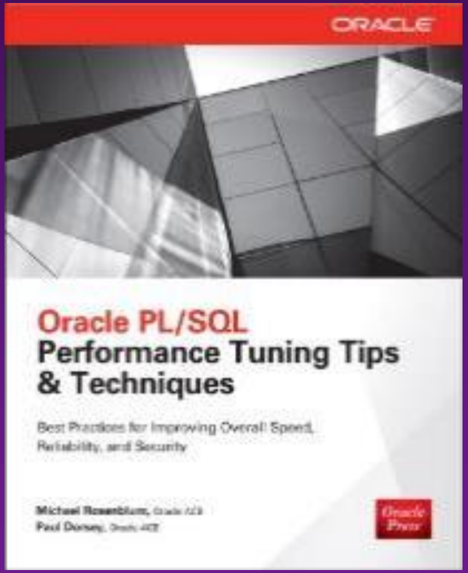

Why am I taking this course?
What am I supposed to do with all of this information?
How can I prove I know it?

- Cloud is here to stay
  - … so, you have to build your systems the right way
- Well-written code in a cloud-based system SAVES LOTS OF MONEY
  - .. so, developers are now visible to CFO
- Oracle provides enough tools to create well-written code ☺
  - … so, you have to learn new tricks - sorry ☺

Contact Information

- Michael Rosenblum – mrosenblum@dulcian.com
- Dulcian, Inc. website - www.dulcian.com
- Blog: wonderingmisha.blogspot.com


Oracle PL/SQL
FOR
DUMMIES
A Reference for the Rest of Us!
Author's Web site includes all code samples
Dr. Paul Dorsey
Michael Rosenblum


ORACLE
Oracle PL/SQL
Performance Tuning Tips & Techniques
Best Practices for Improving Overall Speed, Reliability, and Security
Michael Rosenblum
Paul Dorsey
Oracle Press