Top 5 Issues that Cannot be Resolved by DBAs (other than missed bind variables)

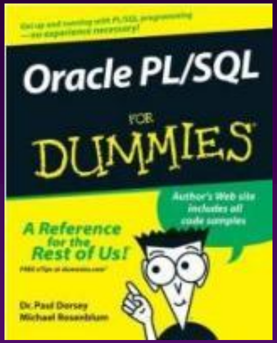
March 12, 2013

Michael Rosenblum
Dulcian, Inc.

www.dulcian.com

Who Am I? – “Misha”

- Oracle ACE
- Co-author of 2 books
  - *PL/SQL for Dummies*
  - *Expert PL/SQL Practices*


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

- Won ODTUG 2009 Speaker of the Year
- Known for:
  - SQL and PL/SQL tuning
  - Complex functionality
    - Code generators
    - Repository-based development

Houston, we have a problem!

- Common thought process:
  - Our IT system has an new issue… OMG!
  - Production code should not be touched (scary!)
  - DBAs should be able to “do something.”
- Reasoning:
  - Configuration of the database is NOT considered production code.
  - DBAs are usually on staff, while the majority of developers are contractors.
  - In the Oracle universe, DBAs are considered to be the most knowledgeable.

Black Hawk Down…

- Results:
  - Significant system architectural problems are covered up using tactical bug-fixes.
  - Systems become even less maintainable and more fragile (I’ve seen 11g systems with RBO still enabled!)
  - Architects and developers become lazy. They expect DBAs to adjust everything afterwards.
  - DBAs become frustrated and remove all privileges from developers.

So what?

- Yes, there are problems that DBAs cannot fix.
- No, I will NOT talk about bind variables ☺
- But I will discuss:
  - Problems usually passed to DBAs
  - Correct solutions of those problems
  - Potential workarounds in cases when a real fix is indeed impossible

Personal Top 5 Non-DBA issues

- Architect’s mistakes:
  - 1. “Smart” columns
  - 2. “STUFF” table
  - 3. “Insufficient” hierarchical structures
- Developer’s mistakes:
  - 4. Datatype misuse
  - 5. Misuse of user-defined functions

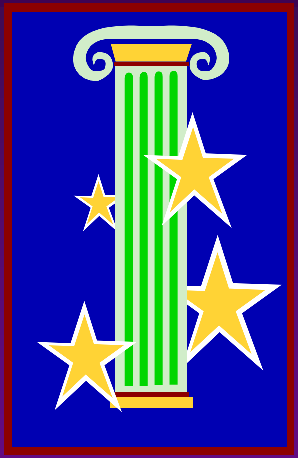
Issue 1: “Smart” Columns

Column vs. Attribute

- Column
  - Represents a single logical attribute
  - Does not make sense if split
- "Smart" column
  - Has internal structure
  - May even change meaning depending upon the data
- Reasons for use:
  - Save time when querying closely related data elements
  - Avoid changes to table structures

Example of “Smart” Columns (1)

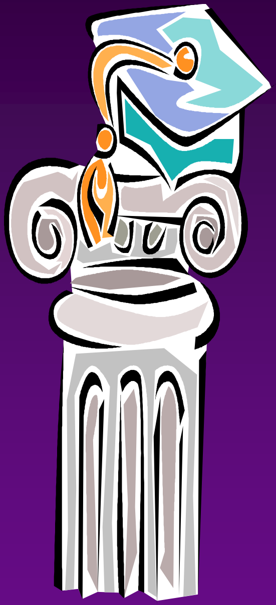
- Organization rollup
  - Pipe-delimited combination of Region/State/City/Zip
- Why is it a problem?
  - Adding extra level to rollup is an extremely challenging task.
  - Search is very expensive.



- What should be done:
  - Split smart columns
  - Aggregate them back using either virtual columns or views

Example of "Smart" Columns (2)

- Answers on questionnaires:
  - Single text line where number of characters = number of questions: "YYYNNNNYYNY"
- Why is it a problem?
  - Versioning of question sets could cause data corruption.
- What should be done:
  - High-quality version control
  - Function-based indexes for the most frequently referenced questions

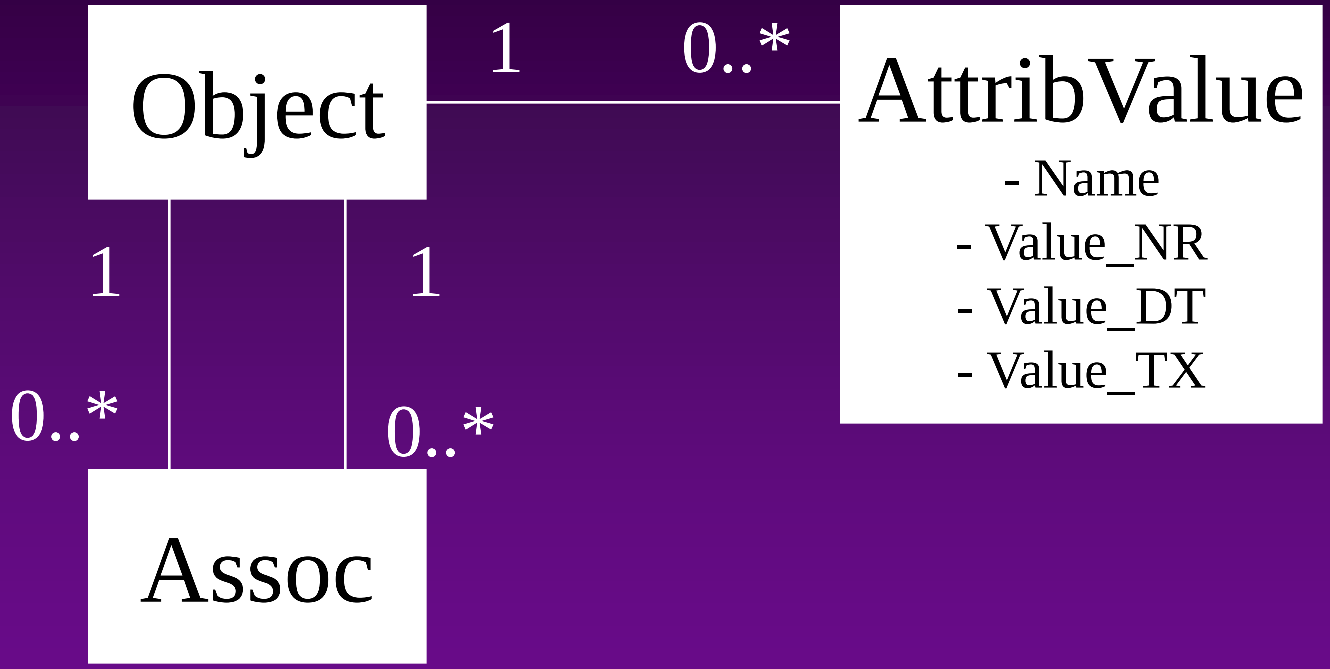


Issue 2: “STUFF” Table

Over-Generalization Trap

- Reasons for generic solutions:
  - Changes are costly.
  - We feel “protected” against the future.
  - Generic models are “cool” (especially now with the Big Data movement)
- BUT
  - Generic solutions often mask incomplete understanding of subject area.
  - Generic solutions in one area could cause major issues in others.

Almost Totally Useless Generic Model



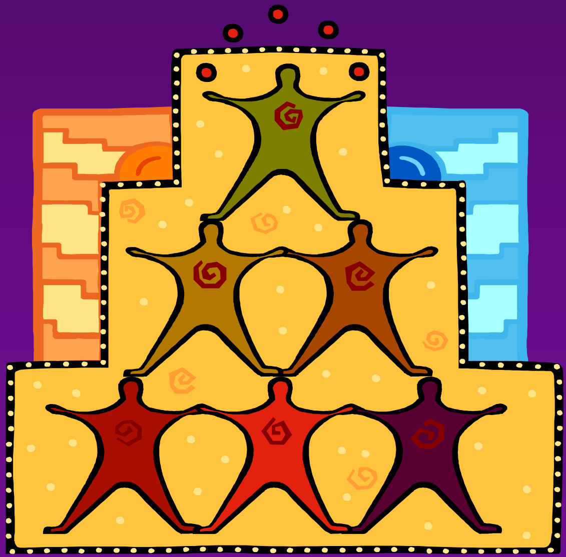
Why is it a bad idea?

- Data entry:
  - Uses a lot of operations to retrieve a single object
  - Data quality is hard to enforce.
- Data retrieval
  - Indexes are useless.
  - CBO goes crazy.
  - Performance is sporadic and does not follow any meaningful logic.
- Functional complex reporting is impossible.

Although…

- There are cases when key-value stores are perfect (NoSQL environments)
- BUT they should not be mixed with:
  - OLTP solutions when high data quality is required
  - Heavy reporting workload
- What could be done:
  - Storage is cheap. Create duplicate structures that would look like real tables

Issue 3: Insufficient Hierarchical Structures



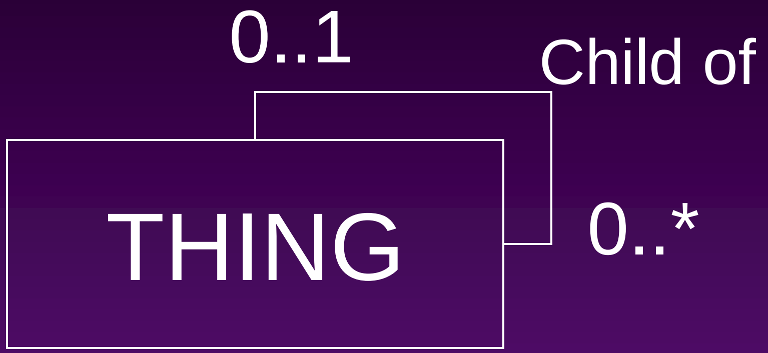
Good Idea/Bad Implementation

- Recursion
  - Powerful modeling technique
  - Can be used for a number of reasons
    - Linked lists (e.g. contract versions)
    - Storage of tree structures (e.g. organizational hierarchy)

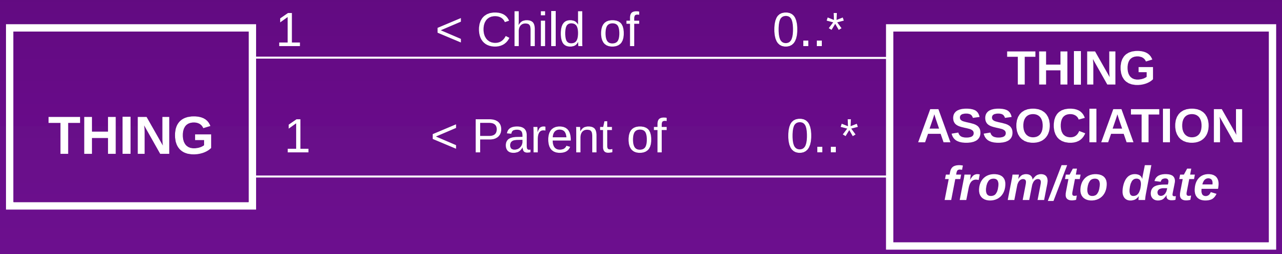
- BUT
  - Storage mechanisms are wrong, which causes a lot of issues

Pseudo-Recursion Trap

- Real recursion



- “Kind of”-recursion



Why is it a trap?

- Reasons why people do it:
  - Versioning
  - Historical data
  - Reporting purposes
- Why it is challenging:
  - Hierarchical data consistency is not enforced.
  - Timing can be very easily be off.
- What should be done:
  - Very strict data quality checking!
  - Denormalized data sources for querying

Issue 4: Datatype Misuse

Datatypes as Constraints (1)

- Datatypes ARE parts of metadata
  - Oracle uses them to make a lot of decisions about execution plans.
  - Wrong datatypes often mean wrong Explain Plans.
  - Wrong datatypes open possibilities for corrupted data.
- What should be done:
  - Fix datatypes as much as possible.
  - Use views/virtual columns to separate storage and representation.
  - Worst case – Add check constraints to at least enforce data quality.

Date vs. Varchar2 (1)

- Problem:
  - storing DATE as VARCHAR2 (~ YYYYMMDD)
- Reasons of issues
  - Date range {December 31, 2012 to January 1, 2013} consists of only two distinct date values
  - The textual range {‘20121231’,’20130101’} is huge. Since it is text, starting with the 4th character there could be any valid character in the current charset.
- Result:
  - Column-level statistics are not utilized and indexes are often ignored.
- What could be done:
  - Build virtual column (TO_DATE) and let developers use it.

Date vs Varchar2(2)

```
create table misha_date01
as
select owner, object_name,
       to_char(created,'YYYYMMDD') created_tx,
       created created_dt
from dba_objects

create index misha_date_tx_idx on
                  misha_date01(created_tx);
create index misha_date_dt_idx on
                  misha_date01(created_dt);

begin
   dbms_stats.gather_table_stats(user,'MISHA_DATE01');
end;
```

Date vs Varchar2 (3)

```
SQL> explain plan for
  2  select *
  3  from misha_date01
  4  where created_tx between '20121231' and '20130101';
Explained.
SQL> select * from table(dbms_xplan.display());
PLAN_TABLE_OUTPUT
------------------------------------------------------------------------
|   0 | SELECT STATEMENT  |                | 48100 |  2113K|   299   (1)|
|*  1 |  TABLE ACCESS FULL| MISHA_DATE01 | 48100 |  2113K|   299   (1)|
------------------------------------------------------------------------
SQL> explain plan for
  2  select *
  3  from misha_date01
  4  where created_dt between to_date('20121231','YYYYMMDD')
  5           and to_date('20130101','YYYYMMDD');
Explained.
SQL> select * from table(dbms_xplan.display());
-------------------------------------------------------------------------
|   0 | SELECT STATEMENT            |                   |   212 |  9540
|   1 |  TABLE ACCESS BY INDEX ROWID| MISHA_DATE01      |   212 |  9540
|*  2 |   INDEX RANGE SCAN          | MISHA_DATE_DT_IDX |   212 |
-------------------------------------------------------------------------
```

Full table scan

Index is used!

Implicit datatype conversion

- Implicit datatype conversion is EVIL!
  - Security nightmare
  - A lot of confusion everywhere:
    - Statistics
    - Execution Plans
    - Overload calls

Number vs Varchar2

```
SQL> explain plan for select * from misha_date01
  2  where created_tx = 20121231;
SQL> select * from table(dbms_xplan.display());
-------------------------------------------------------------------------
| Id  | Operation         | Name           | Rows  | Bytes | Cost (%CPU)|
-------------------------------------------------------------------------
|   0 | SELECT STATEMENT  |                |   573 | 25785 |   300   (1)|
|*  1 |  TABLE ACCESS FULL| MISHA_DATE01 |   573 | 25785 |   300   (1)|
-------------------------------------------------------------------------
```

Full table scan

```
SQL> explain plan for select * from misha_date01
  2  where created_tx = '20121231';
SQL> select * from table(dbms_xplan.display());
-------------------------------------------------------------------------
| Id  | Operation                    | Name                | Rows  | Bytes
-------------------------------------------------------------------------
|   0 | SELECT STATEMENT             |                     |   573 | 25785
|   1 |  TABLE ACCESS BY INDEX ROWID| MISHA_DATE01        |   573 | 25785
|*  2 |   INDEX RANGE SCAN           | MISHA_DATE_TX_IDX |   573 |
-------------------------------------------------------------------------
```

Index is used!

Issue 5: Misuse of User-Defined Functions

Why bother?

- PL/SQL functions as a part of SQL can cause a lot of side effects.
  - Cost of SQL to PL/SQL context switch is very high.
  - Depending upon the execution plan, the same function could be called different numbers of times for the same SQL statement.
- What could be done:
  - Make sure that the CBO takes into account the impact of PL/SQL functions on the overall cost.
  - Manage the total number of calls.

Problem Areas/Solutions

- OO-like get/set APIs
- PL/SQL functions in SELECT and WHERE clauses
  - Managing execution order
    - Short-circuit evaluation
    - Statistics-based cost
  - Decreasing total number of function calls
    - Scalar sub-query caching
    - RESULT_CACHE

- In-line views based on PL/SQL functions returning nested tables

OO-like thinking

- People are accustomed to GET/SET APIs for every attribute
  - Real story of 1 insert into table with 100 attributes
    - 1 insert with only PK column
    - 99 updates using PK
  - System collapsed under its own weight because of thousands of roundtrips
- What could be done:
  - train your developers to NOT use JAVA-style coding in PL/SQL development

PL/SQL functions inside of SQL

- The CBO is not psychic and cannot figure out what is going on inside of your PL/SQL function.
- UNLESS you tell it using associated statistics, because Oracle defaults are not perfect:
  - Selectivity – 1% (0.01)
  - CPU cost – 3000
  - I/O cost – 0
  - Network cost - 0
- There are two ways of doing it:
  - Simple way

```
Associate statistics with
functions <function name>
Default selectivity <value>
Default cost (<CPU>,<IO>,<NETWORK>)
```

  - Complex way [outside of the scope for today]

```
Associate statistics with
functions <function name>
using <special object type>
```

Why does it matter?

- Because you may have multiple functions in the same SQL statement!
- Example
  - Two functions: One is light and one is heavy

```
associate statistics with functions f_misha_light_tx
default selectivity 0.1
default cost (0,0,0);

associate statistics with functions f_misha_heavy_tx
default selectivity 0.1
default cost (99999,99999,99999);
```

  - Both of them are used in the query

```
select /*+ gather_plan_statistics */*
from emp
where f_misha_heavy_nr(empno) = 1
and f_misha_light_nr (empno) = 0
```

Explain Plan Impact

```
SQL_ID  a5u0gvdt0ju36, child number 0
-------------------------------------
select /*+ gather_plan_statistics */* from emp where
f_misha_heavy_tx(empno) = 1 and f_misha_light_tx (empno) = 0

Plan hash value: 3956160932

-----------------------------------------------------------------------------
| Id  | Operation         | Name | E-Rows | A-Rows |   A-Time   | Buffers |
-----------------------------------------------------------------------------
|   0 | SELECT STATEMENT  |      |        |     14 |00:00:00.01 |      33 |
|*  1 |  TABLE ACCESS FULL| EMP  |      1 |     14 |00:00:00.01 |      33 |
-----------------------------------------------------------------------------

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

   1 - filter(("F_MISHA_LIGHT_TX"("EMPNO")=0 AND
              "F_MISHA_HEAVY_TX"("EMPNO")=1))
```

Order of functions has been changed!

Function calls (1)

- Setup:

```
create package misha_pkg is
    v_nr number:=0;
end;

create or replace function f_change_tx (i_tx varchar2)
return varchar2 is
begin
    misha_pkg.v_nr:=misha_pkg.v_nr+1;
    return lower(i_tx);
end;

Create or replace procedure p_check is
begin
     dbms_output.put_line('Fired:'||misha_pkg.v_nr);
     misha_pkg.v_nr:=0;
end;
```

Function calls (2)

```
SQL> select empno, ename, f_change_tx(job) job_change_tx
  2  from emp;
    ...
14 rows selected.

SQL> exec p_check
Fired:14
PL/SQL procedure successfully completed.

SQL> select empno, ename, (select f_change_tx(job) from dual)
  2  from emp;
   ...
14 rows selected.

SQL> exec p_check
Fired:5
PL/SQL procedure successfully completed.
SQL>
```

Scalar sub-query

Only 5 executions!

Function calls (3)

```
create or replace function f_change_tx (i_tx varchar2)
return varchar2 result_cache is
begin
    misha_pkg.v_nr:=misha_pkg.v_nr+1;
    return lower(i_tx);
end;
```

Enable function result cache

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

Only distinct values

```
SQL> select empno, ename, f_change_tx(job) from emp;
...
14 rows selected.
SQL> exec p_check
Fired:0
```

No calls – cache only!

Collection IN-lists (1)

- It is very convenient to build an IN-list as a collection and pass it to a WHERE clause
  - But Oracle may or may not correctly interpret incoming data!

- Example (setting)

```
create table misha_demo_inlist as
select object_id, created
from dba_objects
where owner = 'MISHA'
and object_id is not null;

alter table misha_demo_inlist add constraint
misha_demo_inlist_pk primary key (object_id) using index;

begin
dbms_stats.gather_table_stats(user,'MISHA_DEMO_INLIST');
end;
```

Collection IN-lists (2)

```
create type id_tt is table of number;

select /*+ gather_plan_statistics*/
         max(created)
from misha_demo_inlist
where object_id in (
        select t.column_value
        from table(id_tt(227011,227415)) t
        )
```

Collection IN-lists (3)

```
SQL_ID  6509b6f6d1mgy, child number 0
-------------------------------------
select /*+ gather_plan_statistics */ max(created) from
misha_demo_inlist where object_id in (
select t.column_value
from table(id_tt(227011,227415)) t)

Plan hash value: 22551403

---------------------------------------------------------------------------------------
| Id  | Operation                                | Name              | E-Rows | A-Rows |
---------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT                         |                   |        |      1 |
|   1 |  SORT AGGREGATE                          |                   |      1 |      1 |
|*  2 |   HASH JOIN                              |                   |   8168 |      2 |
|   3 |    COLLECTION ITERATOR CONSTRUCTOR FETCH|                   |   8168 |      2 |
|   4 |    TABLE ACCESS FULL                     | MISHA_DEMO_INLIST |  29885 |  29885 |
---------------------------------------------------------------------------------------


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

   2 - access("OBJECT_ID"=VALUE(KOKBF$))
```

Wrong cardinality

Collection IN-lists (4)

- Oracle does not correctly recognize how many objects are in the collection.

- Alternatives:
  - Explicit cardinality hint

```
select /*+ gather_plan_statistics */ max(created)
from misha_demo_inlist
where object_id in (
    select /*+ cardinality (t 2) */t.column_value
    from table(id_tt(227011,227415)) t
    )
```

  - Dynamic sampling

```
select /*+ gather_plan_statistics */ max(created)
from misha_demo_inlist
where object_id in (
    select /*+ dynamic_sampling (t 4) */t.column_value
    from table(id_tt(227011,227415)) t
    )
```

Collection IN-lists (5)

- Result for both options is the same – and uses the index!

```
-------------------------------------------------------------------------------------
| Id  | Operation                                | Name                |E-Rows |A-Rows
-------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT                         |                     |       |     1
|   1 |  SORT AGGREGATE                          |                     |     1 |     1
|   2 |   NESTED LOOPS                           |                     |       |     2
|   3 |    NESTED LOOPS                          |                     |     2 |     2
|   4 |     COLLECTION ITERATOR CONSTRUCTOR FETCH|                     |     2 |     2
|*  5 |     INDEX UNIQUE SCAN                    | MISHA_DEMO_INLIST_PK |     1 |     2
|   6 |    TABLE ACCESS BY INDEX ROWID           | MISHA_DEMO_INLIST   |     1 |     2
-------------------------------------------------------------------------------------

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

   5 - access("OBJECT_ID"=VALUE(KOKBF$))
```

Correct cardinality!

- Dynamic sampling will also have a special note about its level (it can be lower than requested)

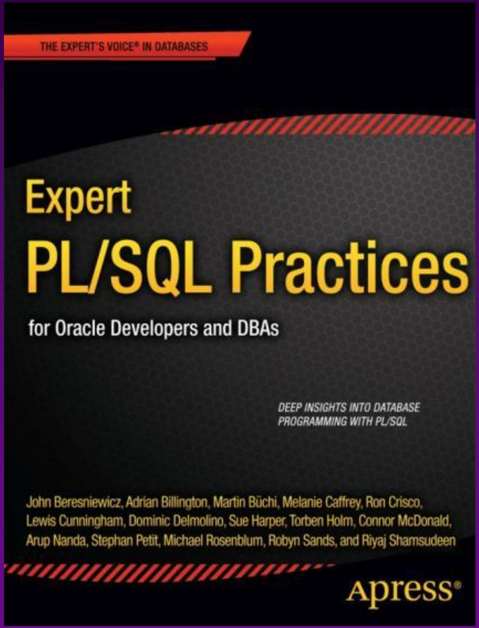
```
Note
-----
   - dynamic sampling used for this statement (level=2)
```

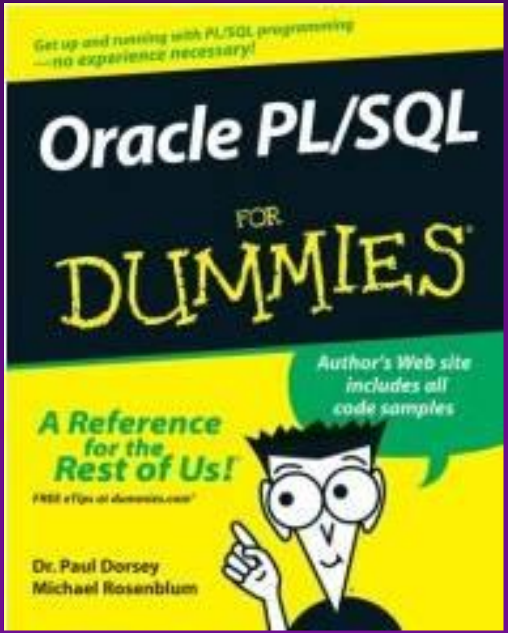
Summary

- Not all errors can be fixed by DBAs.
- Strategic problems should not be covered by tactical solutions.
- Enterprise-level thinking is required from the very beginning.
- … and let's not forget about bind variables ☺

Contact Information

- Michael Rosenblum – mrosenblum@dulcian.com
- Blog – wonderingmisha.blogspot.com
- Website – www.dulcian.com


THE EXPERT'S VOICE® IN DATABASES
Expert
PL/SQL Practices
for Oracle Developers and DBAs
DEEP INSIGHTS INTO DATABASE PROGRAMMING WITH PL/SQL
John Beresniewicz, Adrian Billington, Martin Büchi, Melanie Caffrey, Ron Crisco, Lewis Cunningham, Dominic Delmolino, Sue Harper, Torben Holm, Connor McDonald, Arup Nanda, Stephan Petit, Michael Rosenblum, Robyn Sands, and Riyaj Shamsudeen
Apress®


Get up and running with PL/SQL programming —no experience necessary!
Oracle PL/SQL
FOR
DUMMIES®
Author's Web site includes all code samples
A Reference for the Rest of Us!®
FREE eTips at dummies.com®
Dr. Paul Dorsey
Michael Rosenblum

Available now:

Expert PL/SQL Practices