

ORACLE®

All material available for download at:

<http://asktom.oracle.com> -> files tab -> psoug.zip

Worst Practices For Developers and DBAs Alike

ORACLE®

Who am I

Ask Tom Home - Mozilla Firefox

File Edit View Go Bookmarks Tab Tools Help

http://asktom.oracle.com/pls/ask/f?p=4950:1

Ask Tom Home

Search

times are East coast times (GMT-04:00)
In the last 4 weeks, I've taken 354 new questions, read 1,530 followups, and responded to 1,139 of the followups

Hit	Subject	Last Updated	First Asked	Age
1	wrap	16 Jan 2005 1pm	02 Mar 2001 9pm	3.9 years old
2	Error Trapping - Reg	16 Jan 2005 12pm	29 Mar 2001 10pm	3.8 years old
3	lost update	16 Jan 2005 11am	21 Dec 2004 2pm	3 weeks old
4	OLTP Rollback segment planning	16 Jan 2005 11am	21 May 2000 7am	4.7 years old
5	What's your opinion about the DBA job in the future	16 Jan 2005 11am	29 Nov 2003 9pm	1.1 years old
6	any performance hit using sequence with nocache in OLTP	16 Jan 2005 11am	08 Nov 2002 1pm	2.2 years old
7	Full outer join in 8i	16 Jan 2005 11am	16 Jan 2003 10pm	2 years old
8	Delete Conflict Resolution	16 Jan 2005 11am	19 Oct 2000 3pm	4.2 years old
9	Loading Lobs Using SQL Loader	16 Jan 2005 11am	23 Aug 2000 4pm	4.4 years old
10	data files	16 Jan 2005 11am	03 Nov 2003 8am	1.2 years old

Submit a New Question



- Been with Oracle since 1993
- User of Oracle since 1987
- The “Tom” behind AskTom in Oracle Magazine
www.oracle.com/oramag
- Expert Oracle: Database Architecture
- Effective Oracle by Design
- Beginning Oracle
- Expert One on One Oracle

**You Should Probably
Never Question
Authority
Never
Not Ever**

(it bothers them when you do)

“Never Question Authority.”

- Experts are *always right*
- You know the information is accurate when the author clearly states:
 - It is my opinion...
 - I claim...
 - I think...
 - I feel...
 - I **KNOW...**
- Nothing need be backed up with evidence
- Things **never** change
- Counter Cases do not prove anything
- If it is written down, it must be true



You Probably Do Not Need to Use Bind Variables

It is so much easier to code without them!

```
query =  
'select *  
  from t  
 where x = ?  
    And y = ?'
```

```
Prepare it  
Bind x  
Bind y  
Execute it  
Close it
```

Too much code!

```
query =  
'select *  
  from t  
 where x = '||x||'  
    And y = '||y'
```

```
Execute it
```

Look at how efficient I am!

And very secure too!

Enter Username: tom' or 1=1 –

Enter Password: i_dont_know' or 1=1 –

Query =

```
"select count(*) " +  
"  from user_pw " +  
" where uname = \" + uname + \"'\" +  
"    and pword = \" + pword + \"'\"
```

```
Select count(*)  
  From user_pw  
 Where uname = 'tom' or 1=1 - \  
       And pword = 'i_dont_know' or 1=1 - \'
```


Performance isn't a concern

- It is not a problem that a large percent of my program runtime will be spent parsing. That is ok!

```
SQL> set timing on
SQL> begin
  2     for i in 1 .. 100000
  3     loop
  4         execute immediate
  5         'insert into t (x,y)
  6         values ( ' || i ||
  7         ', ''x'' )';
  8     end loop;
  9 end;
10 /
```

PL/SQL procedure successfully completed.

Elapsed: 00:01:33.85

Performance isn't a concern

- It is not a problem that a large percent of my program runtime will be spent parsing. That is ok!

```
SQL> set timing on
SQL> begin
  2     for i in 1 .. 100000
  3     loop
  4         execute immediate
  5         'insert into t (x,y)
  6         values ( :i, ''x'' )'
  7         using i;
  8     end loop;
  9 end;
10 /
```

PL/SQL procedure successfully completed.

Elapsed: 00:00:04.69

Performance isn't a concern

- It is not a problem that a large percent of my program runtime will be spent parsing. That is ok!
- That 95% of my runtime was spent parsing SQL in a single user test is perfectly OK!

And I'm sure memory utilization is OK

```
SQL> select case when instr( sql_text, ':' ) > 0
2         then 'bound'
3         else 'not bound'
4         end what, count(*), sum(sharable_mem) mem
5   from v$sql
6  where sql_text like 'insert into t (x,y)          values (%)'
7  group by case when instr( sql_text, ':' ) > 0
8         then 'bound'
9         else 'not bound'
10         end;
```

WHAT	COUNT(*)	MEM
not bound	6640	56,778,665
bound	1	8,548

```
SQL> show parameter shared_pool_size
```

NAME	TYPE	VALUE
shared_pool_size	big integer	152M

And it'll absolutely scale up!

- Oracle is the most scalable database in the world, it'll take care of it.

Run1 latches total versus runs...

Run1	Run2	Diff	Pct
13,349,321	548,684	-12,800,637	2,432.97%

**Probably
You don't want to
expose end users to
errors**

When others then null;

- End users would never want to know there was a problem
- Even if the “end user” is really another module calling you
- Just log it – don’t raise it

```
Begin
  ...
Exception
When others Then
    log_error( ... );
End;
```

**Probably
The More Generic
You Can Make
Something, The
Better It Is.**

Or...

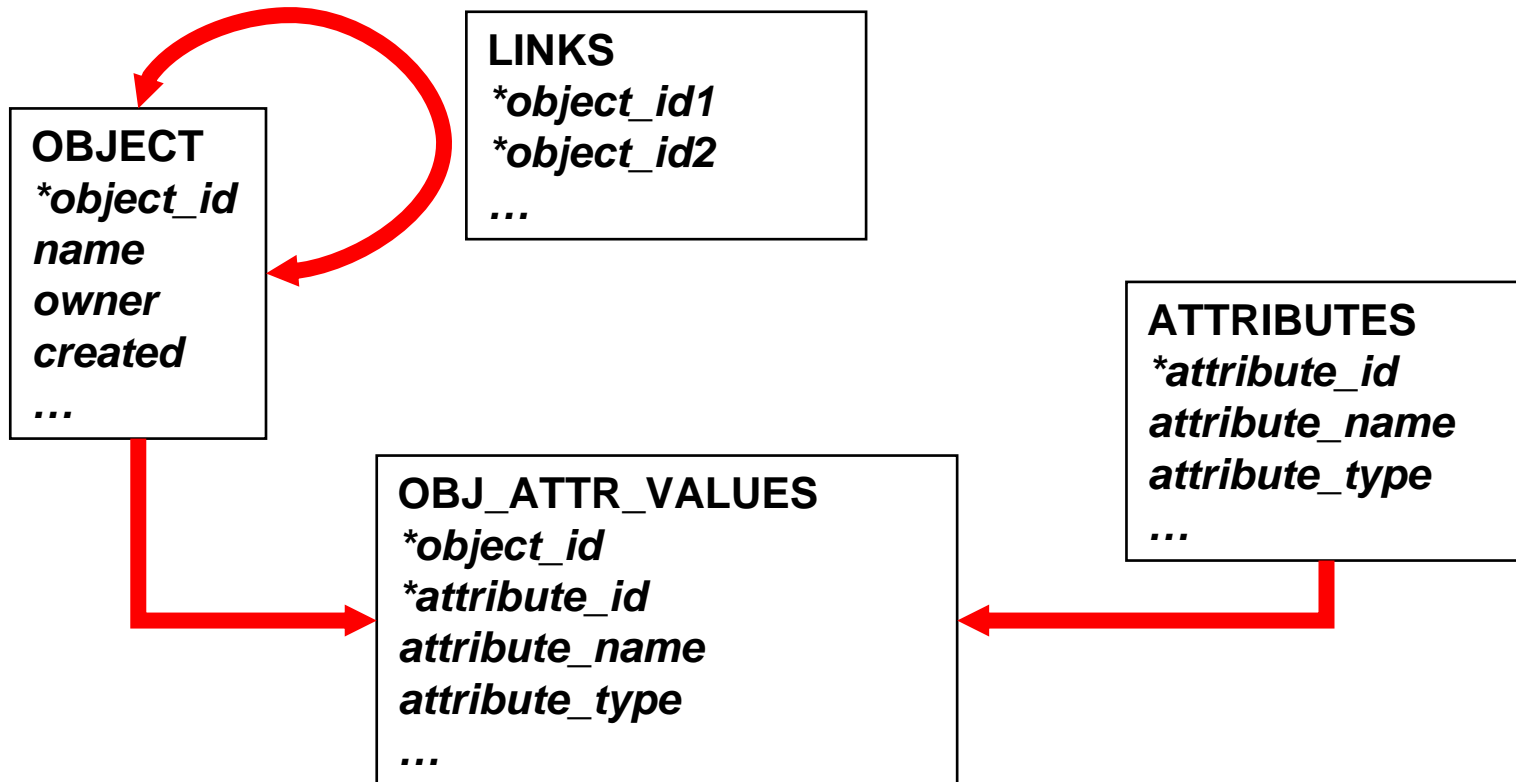
**Probably
You Do Not Need to
Actually Design
Anything**

Quickly Answer:

- How many tables do you *really* need?

Quickly Answer:

- How many tables do you *really* need?
- **FOUR** at *most!*



Quickly Answer:

- How many tables do you *really* need?
- But of course **ONE** is *best!*
- And you are industry standard as well!

```
Create table Object  
( object_id number primary key,  
  data          xmltype );
```

In case you think I make this stuff up...

From - Wed Nov 08 07:39:19 2006

X-Mozilla-Status: 0001

X-Mozilla-Status2: 00000000

Return-Path: <xxxxx@xxxxx.com>

Received: from rgmum105.us.oracle.com by rcsmt251.oracle.com

with ESMTP id 2180055871162956506; Tue, 07 Nov 2006 20:28:26 -0700

...

id C7431B2F2B; Tue, 7 Nov 2006 20:28:09 -0700 (MST)

Mime-Version: 1.0 (Apple Message framework v752.2)

Content-Type: multipart/alternative; boundary=Apple-Mail-107--34306936

Message-Id: <EDEA1DBE-CF47-4D52-9A91-24CC4A208836@mac.com>

From: Dan XXXXX <xxxxxxxx@xxx.com>

Subject: Worst Practices

Date: Tue, 7 Nov 2006 19:28:06 -0800

To: Thomas Kyte <thomas.kyte@oracle.com>

X-Mailer: Apple Mail (2.752.2)

X-Virus-Scanned: by Barracuda Spam Firewall at theedge.ca

X-Brightmail-Tracker: AAAAAQAAAAI=

X-Whitelist: TRUE

In case you think I make this stuff up...

Sorry about the unrequested email, but I couldn't resist...

I read your Worst Practices presentation the other day - Very nice, hit a bit close to home for comfort in many cases!

Then today I got an email from one of the contract "developers" our organization deals with, it describes a rewrite of a system that was rolled out a few years back. It was a bit experimental and was always problematic - architectural mess - stuff flying around in files between ftp sites and windows shares and in and out of databases. I (and my cohort DBA) kept asking "Why doesn't this just stay in a database and you query it from wherever".

In case you think I make this stuff up...

... BUT It was developed shortly after one of our architect types had heard of XML, so XML had to be used, it wasn't really important what it was to be used for - it was just to be used... and so it was decreed, and it was made so, and it was good... well until the Xindice "database" thing started crapping out every few days... but then some sys admin wrote a script to check and restart Xindice every few minutes, and it was good again.... fast forward a few years.... decision is taken to rewrite and since our Oracle databases don't seem to crash every seventh minute, move the backend from Xindice to Oracle....

In case you think I make this stuff up...

Here is the "punch line" from the email describing the database aspects of the proposed system (slightly edited to remove reference to specific client):

"My current design for the Oracle-ized (Oracle 10g) version requires only a single Oracle table, which will have two columns: a pseudo key (simple varchar2) which will likely actually contain the path to a corresponding document in the WebDAV environment, and a document column of XMLType which will contain the xml for an individual "notice" within the `[[snip]]`, plus an index on the pseudo key column."

In case you think I make this stuff up...

Excellent - one table, with a key and XMLType column - the perfect system... Is this a a cut and paste off slide 21 of your Worst Practices ppt or what???

If I could make this stuff up I could quit my job and work in stand-up.

sigh.

Quickly Answer:

- How many tables do you *really* need?
- Either ONE or FOUR, not any more...
- You'll never have to put up with asking the DBA for anything again!
- End users will never want to actually use this data except from your application!
- Performance – it should be OK, if not the DBA will tune the database
- Or we'll just get a new database if the one we are using is not fast.

**Probably
You want as many
instances per server
as possible**

Many Instances

- It'll be easier to tune of course – each database can be it's own unique “thing”
 - Multiple dbwr's would never contend with each other
 - Of course there is some magic global view that will point out areas of contention for us
- Everyone will have their “own” memory
 - There won't be *any duplication* or *increased memory* usage due to this
- A runaway process on one instance won't be *my* problem

**Probably
You should reinvent
as many database
features as possible**

Reinvent the Wheel

- Writing Code is fun
 - Using built in functionality will not demonstrate your technical capabilities to your manager!
- The builtin stuff only solves 90+% of your extremely unique, sophisticated, 22nd century needs after all
 - It is not good enough
- Besides, you would not want to become dependent on the vendor
 - Much better to be dependent on *you* after all!
- It must cost less, doesn't it?

**Probably
You Do Not Need To
Test**

Testing would be such a waste of time

- It might not break
- So why spend the time trying to make it break
- It *probably* won't have any scalability issues
- If you test at all, a single user test on your PC does as well as a fully loaded test on a server
- If you test at all part 2; testing on an empty database is just as good as testing on a full one.
- Just do the upgrade, it'll probably work
- Besides, if I test – they'll expect it works and if it doesn't then I'll be in trouble

**Probably
You Should Only Use
The Varchar
Datatype**

Varchar2

- It is so much easier after all
- It would never confuse the optimizer

```
ops$tkyte%ORA10GR2> create table t ( str_date, date_date )
2  as
3  select to_char( dt+rownum,'yyyymmdd' ),
4         dt+rownum
5  from (select to_date('01-jan-1995','dd-mon-yyyy') dt
6         from all_objects)
7  /
```

```
ops$tkyte%ORA10GR2> create index t_str_date_idx on t(str_date);
ops$tkyte%ORA10GR2> create index t_date_date_idx on t(date_date);
ops$tkyte%ORA10GR2> begin
2  dbms_stats.gather_table_stats
3  ( user, 'T',
4    method_opt=> 'for all indexed columns',
5    cascade=> true );
6  end;
7  /
```

Varchar2

- It is so much easier after all
- It would never confuse the optimizer

```
SQL> select * from t where str_date between '20001231' and '20010101';
```

Operation	Name	Rows
SELECT STATEMENT		300
TABLE ACCESS BY INDEX ROWID	T	300
INDEX RANGE SCAN	T_STR_DATE_IDX	300

```
SQL> select * from t where date_date between to_date('20001231','yyyymmdd')
2 and to_date('20010101','yyyymmdd');
```

Operation	Name	Rows	Bytes
SELECT STATEMENT		1	17
TABLE ACCESS BY INDEX ROWID	T	1	17
INDEX RANGE SCAN	T_DATE_DATE_IDX	1	

Varchar2

- Datatypes are overrated.
 - They are just fancy integrity constraints after all
 - They won't affect client memory usage at all
 - We'll only put numbers in that string, it'll be just OK

**Probably
You Should Commit
Frequently**

Commit Frequently

- Auto Commit is best
 - If I didn't mean for something to be permanent I wouldn't have done it after all!
- Definitely commit frequently to save resources and go faster
 - It won't generate *more redo* would it?
 - It won't generate *more total undo* would it?
 - Log_file_sync (the wait event observed during commit) is something the DBA will tune away for us won't they?

Commit Frequently

- My code won't fail:

```
For x in (select * from t1)
Loop
    insert into t2 values ..;
    cnt := cnt + 1;
    if (mod(cnt,100)=0)
    then
        commit;
    end if;
End loop;
```

- So we don't need to make it restartable or anything

**Probably
You Should Be
Database
Independent**

The Promise

- Write Once
- Deploy Everywhere on anything
- Less Work overall

The Reality

- Write Once
 - For each database
 - They are different
- Deploy Everywhere on anything

- Less Work overall

The Reality

- Write Once
 - For each database
 - They are different
- Deploy Everywhere on anything
 - Deploy on specific dot releases
 - Of specific databases
 - On certain platforms
 - (it is a support issue)
- Less Work overall

The Reality

- Write Once
 - For each database
 - They are different
- Deploy Everywhere on anything
 - Deploy on specific dot releases
 - Of specific databases
 - On certain platforms
 - (it is a support issue)
- Less Work overall
 - More work overall

**Probably
You Do Not Need
Configuration
Management Of Any
Sort**

We probably do not need CM

- Database code isn't really code after all
 - It is a bunch of scripts
 - Scripts are not *code* really, they are something less than code
 - No need to keep track of the
 - Grants, Creates, Alters and so on...
 - Besides, we can probably just get it from the data dictionary
 - Because the scratch test database we develop on is maintained just like a production instance is!

We probably do not need CM

- “Diffing” databases to see what’s different schema wise to do application updates
 - Is completely acceptable
 - Very professional
 - Makes it easier to document
 - Leads to much better designs
 - You don’t really need to know what is changing between version 1 and 2

**Probably
You Do Not Need To
Design To Be
Scalable**

Scalability

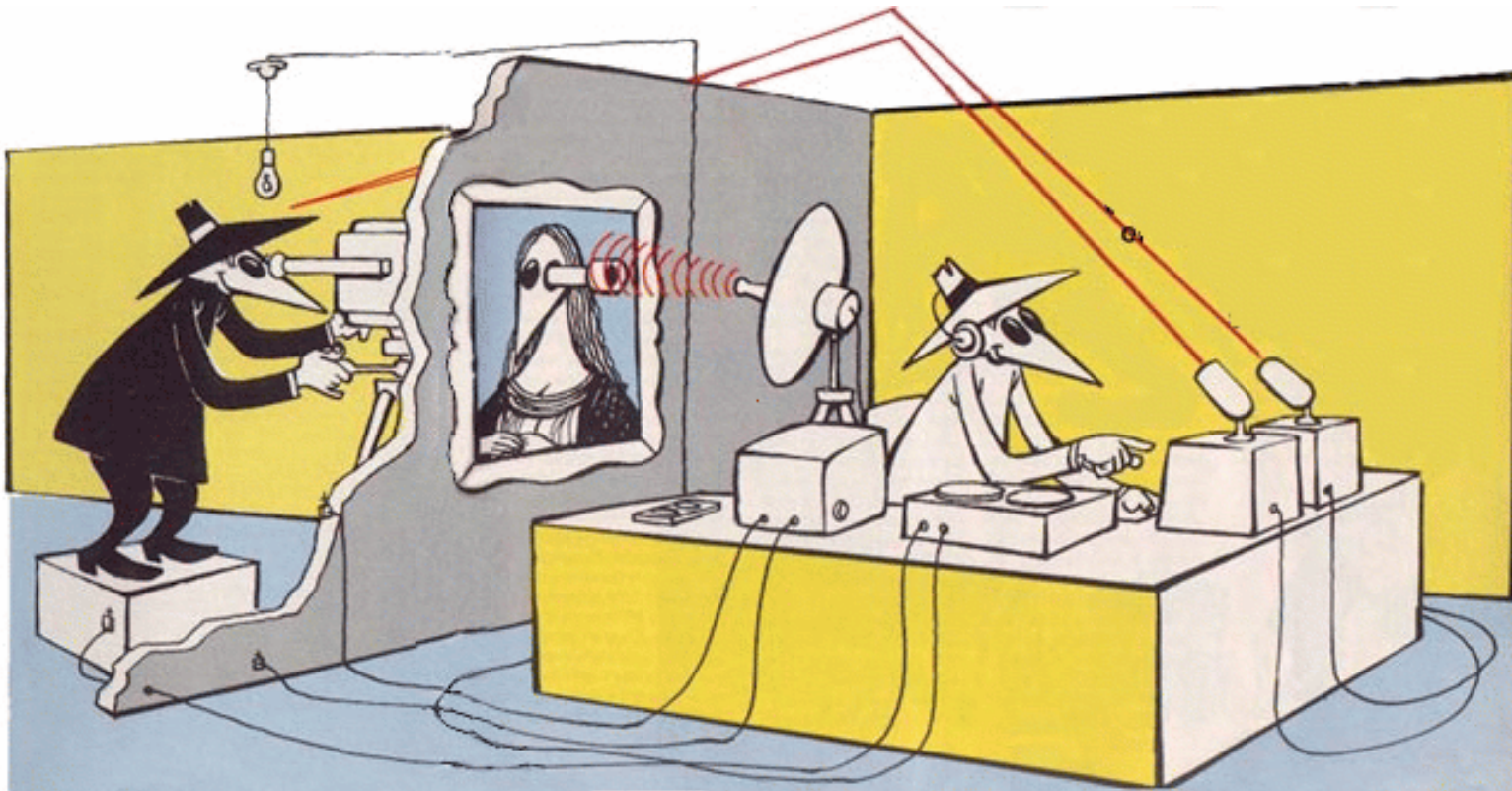
- Scalability just happens
- Oracle is very scalable
 - Therefore, so shall ye be scalable
- It is a *shared pool* – we all just share it together
 - Contention free
- This is really why you probably do not need to test
- Besides, you can just add more
 - CPU
 - Memory
 - Disk

Probably
You do not need to
design to be secure



**DBAs And
Developers Are Just
Different, So Get
Over It**

DBA vs Developer vs DBA



The Job of the DBA is...

- Priority #1 is to protect the database from the developers
- Outlaw features, they might be mis-used
 - Views, had a bad experience with a view once...
 - Stored procedures, they just use CPU
 - Any feature added after version 6
 - No feature can be used until it is at least 5 versions old
 - software is just like fine wine

The Job of the DBA is...

- It is not your job to educate
- Just say *no*. You need not explain why, you are the DBA after all.
- These are perfectly valid reasons to avoid using a database feature:
 - “I heard it was slow”
 - “I’ve heard it is buggy”

Developers

- It is true, the DBA is not there to work with you
- Try to find ways to avoid having to work with them, such as..
 - Don't ask any questions
 - Do as much as you can outside of the database
- Do not join, you can write code to do that
- Do not use database features, you can write code to do that
- Do not use integrity constraints in the database, you can write code to do that
- Try to be as generic and general purpose as possible
- And remember – the DBA is responsible for performance, scalability, and security. You are not.

ORACLE®