

Oracle 12c for Developers



Alex Nuijten





nuijten.blogspot.com


Notes on Oracle
Oracle Things I Got to Remember Not to Forget


25 September 2013
Oracle In-Memory Database

One of the big announcements of Oracle Open World 2013 is the Oracle Database In-Memory option. By simply "flipping the switch" the application will run faster than ever before. From my understanding flipping the switch is making changes to some (not sure which) initialisation parameters.

Secret Sauce
What is the big deal with Oracle In-Memory Database? Well, it is fast - really, really fast. DRAM is faster than Flash, and Flash is faster than disks. Another big deal is the dual format in which the data is stored. The data is stored in a columnar and row fashion, both in memory. This means that analytical type queries can be answered by the columnar storage while other type of queries can be resolved using the row storage. And yes, the optimizer is aware of columnar storage as well. Oracle takes care of storing the data in both formats, this is done completely transparent. It is simultaneously active and transactionally consistent. The secret sauce is the dual format storage, this is what it really makes it fast. The columnar format of the data is memory only, the row format is stored in memory and persisted on disk, just as it is nowadays. In order to take full advantage of this you don't need to learn anything else - from a developer point of view. During DML the affected data is marked as stale (in memory) thereby limiting the overhead. You can pick and choose which tables, partitions, columns you want to have in columnar format.

How does it differ from TimesTen?
How does it differ from TimesTen? Good Question. TimesTen is also an in memory database which has been around for a number of years. TimesTen is a more specialized approach to

About Me
 **ALEX NUIJTEN**
Follow 167
Working as an Oracle Consultant for Ordina Oracle Solutions in The Netherlands. Presented at National and International Conferences. Oracle ACE Director for Database Development. Trainer for SQL and PL/SQL. Married, two children, likes to Barbecue.
[View my complete profile](#)

ORDINA
 **ORACLE**
ACE Director



Professional Expertise Distilled

Oracle APEX Best Practices

Accentuate Oracle APEX development with proven best practices

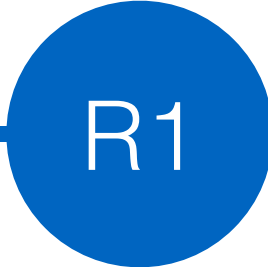
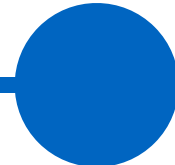
Learco Brizzi
Ilona Ellen-Wolff Alex Nuijten

PACKT enterprise
PUBLISHING

← 4 Years →

2009

2013

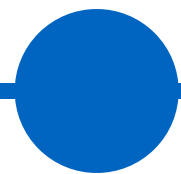


ORACLE
DATABASE **11^g**

ORACLE
DATABASE **12^c**

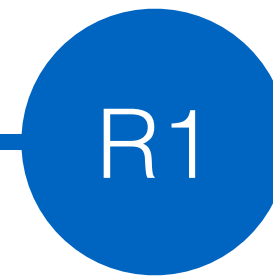
Multitenant Architecture

2009



ORACLE
DATABASE **11^g**

2013



ORACLE
DATABASE **12^c**

In Memory Option

2013



2014



ORACLE[®]
DATABASE **12^c**

... but what's in it for the Database Developer?



A vintage measuring tape is shown against a white background. The tape is made of cork and is marked with black numbers and vertical lines. The numbers visible include 2, 3, 4, 5, and 6. The word "TINK" is printed in a stylized font on the tape. The tape is attached to a dark brown wooden case with a metal clasp. The text "Increased Size Limit" is overlaid in white on the tape.

Increased Size Limit


```
SQL> create table t  
      2  (str varchar2(4000));
```

Table created.

```
SQL> create table t
  2  (str varchar2(4001));
(str varchar2(4001))
      *
```

ERROR at line 2:

ORA-00910: specified length too long for its datatype


```
SQL> create table t
  2  (str varchar2(32767))
  3  /
(str varchar2(32767))
      *
```

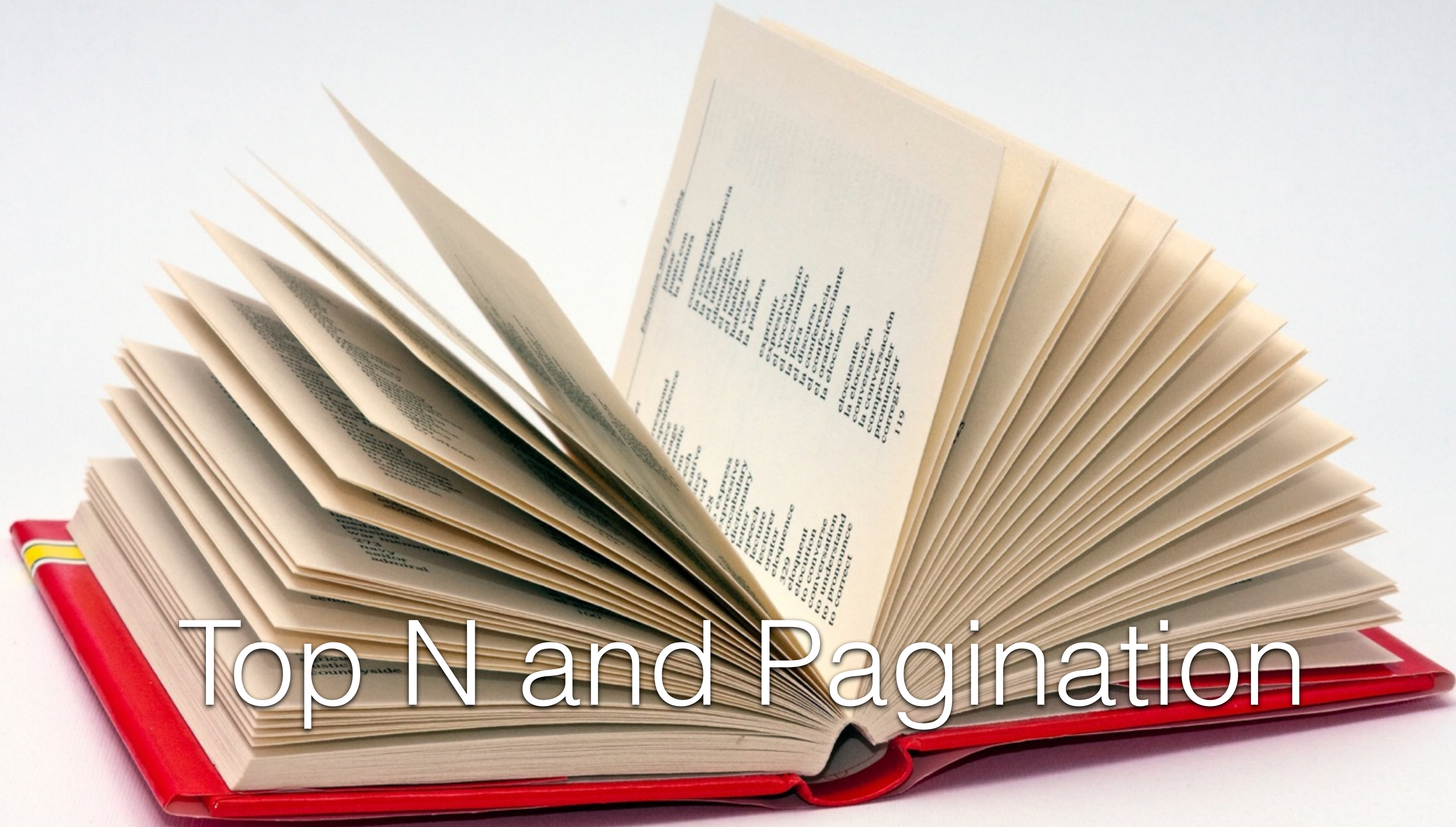
ERROR at line 2:

ORA-00910: specified length too long for its datatype

a little bit of DBA magic


```
SQL> create table t  
      2  (str varchar2(32767));
```

Table created.



Top N and Pagination


```
SQL> select ename  
2         ,sal  
3         from emp  
4         where rownum < 4  
5         order by sal desc  
6         /
```

ENAME	SAL
ALLEN	1200
WARD	1250
SMITH	800

```
SQL> select *
      2     from (select ename
      3                  ,sal
      4                  from emp
      5                  order by sal desc
      6                )
      7     where rownum < 4
      8 /
```

ENAME	SAL
-----	-----
KING	5000
SCOTT	3000
FORD	3000

```
SQL> select ename
2         ,sal
3     from emp
4     order by sal desc
5     fetch first 3 rows only
6 /
```

ENAME	SAL
KING	5000
SCOTT	3000
FORD	3000


```
SQL> select ename
2         ,sal
3     from emp
4     order by sal desc
5     offset 3 rows
6     fetch next 3 rows only
7 /
```

ENAME	SAL
JONES	2975
BLAKE	2850
CLARK	2450

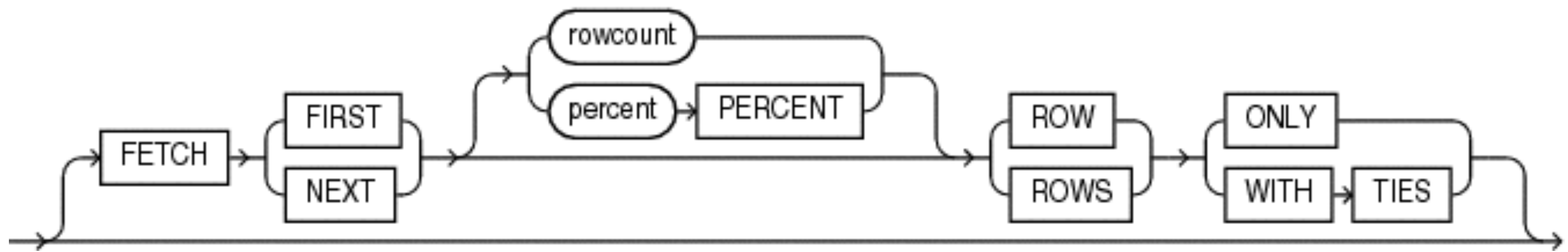
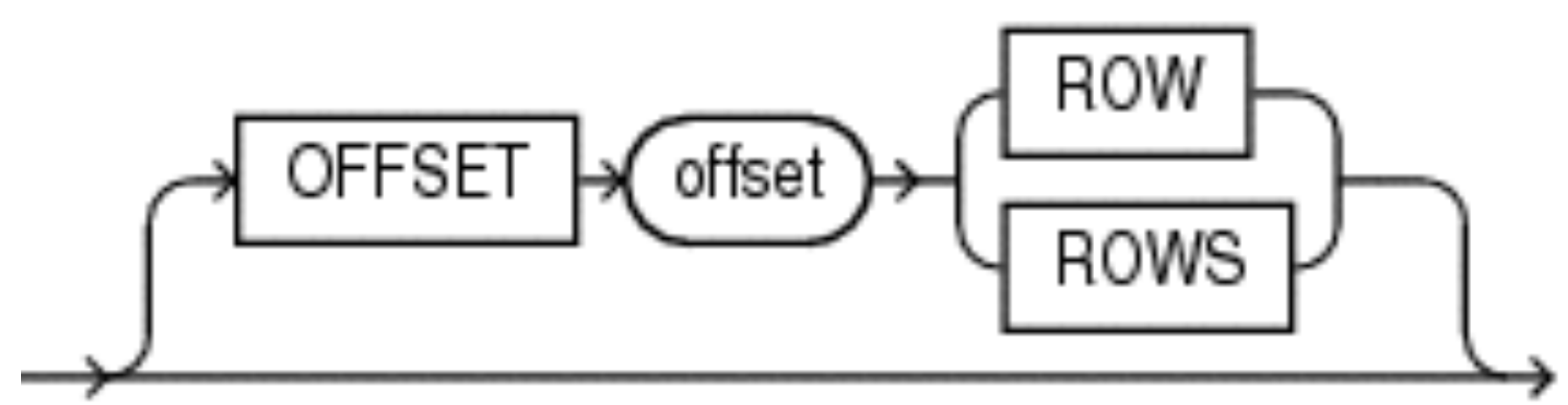
Execution Plan

Plan hash value: 3291446077

Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time
0	SELECT STATEMENT		14	644	4 (25)	00:00:01
* 1	VIEW		14	644	4 (25)	00:00:01
* 2	WINDOW SORT PUSHED RANK		14	140	4 (25)	00:00:01
3	TABLE ACCESS FULL	EMP	14	140	3 (0)	00:00:01

Predicate Information (identified by operation id):

-
- 1 - filter("from\$_subquery\$_002"."rowlimit_\$\$_rownumber"<=CASE WHEN (3>=0) THEN 3 ELSE 0 END +3 AND "from\$_subquery\$_002"."rowlimit_\$\$_rownumber">3)
 - 2 - filter(**ROW_NUMBER()** OVER (ORDER BY INTERNAL_FUNCTION("SAL") DESC)<=CASE WHEN (3>=0) THEN 3 ELSE 0 END +3)




```
SQL> select ename
2         ,sal
3     from emp
4     order by sal desc
5     fetch first 25 percent row only;
```

ENAME	SAL
KING	5000
FORD	3000
SCOTT	3000
JONES	2975

Execution Plan

Plan hash value: 4130734685

Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time
0	SELECT STATEMENT		14	826	4 (25)	00:00:01
* 1	VIEW		14	826	4 (25)	00:00:01
2	WINDOW SORT		14	140	4 (25)	00:00:01
3	TABLE ACCESS FULL	EMP	14	140	3 (0)	00:00:01

Predicate Information (identified by operation id):

1 - filter("from\$_subquery\$_002"."rowlimit_\$\$_rownumber"<=CEIL("from\$_subquery\$_002"."rowlimit_\$\$_total"*25/100))

The image is a collage of various plaid and tartan fabric patterns. The patterns are arranged in a grid-like fashion, with some overlapping. The colors used include shades of green, blue, red, orange, brown, and white. The patterns vary in their grid structures, with some having thin lines and others having thicker, more prominent lines. The overall effect is a rich and diverse display of traditional textile designs.

Pattern Matching


```
SQL> create table weather
      2  (dt    date    not null
      3  ,rain number not null
      4  );
```

Table created.

```
SQL> insert into weather
  2  with rsf(r)
  3  as
  4  (select 1
  5     from dual
  6     union all
  7     select r + 1
  8     from rsf
  9     where r < 31)
 10  select to_date (to_char (r) || '-08-2014', 'dd-mm-yyyy') dt
 11         ,round ((dbms_random.value * 20)) rain
 12  from rsf
 13  /
```

31 rows created.

```
SQL> select dt
2         ,rain
3         ,trend
4   from weather
5 match_recognize(
6   order by dt
7   measures classifier() as trend
8   all rows per match
9   pattern (Better* Worse* same*)
10  define Better as Better.rain < prev (rain)
11         ,Worse  as Worse.rain > prev (rain)
12         ,same   as same.rain = prev (rain)
13 );
```



DT	RAIN	TREND
01-AUG-14	14	
02-AUG-14	0	BETTER
03-AUG-14	19	WORSE
04-AUG-14	6	BETTER
05-AUG-14	20	WORSE
06-AUG-14	1	BETTER
07-AUG-14	17	WORSE
08-AUG-14	17	S
09-AUG-14	14	B
10-AUG-14	18	W
11-AUG-14	9	B
12-AUG-14	4	B
13-AUG-14	17	WORSE
14-AUG-14	16	BETTER
15-AUG-14	5	BETTER

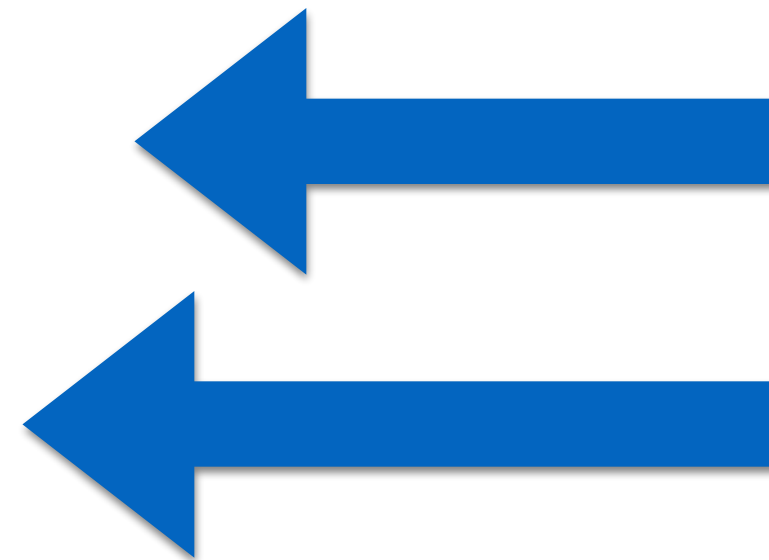
What about Analytic Functions?

```
SQL> select dt
2         ,rain
3         ,case
4         when rain < lag (rain) over (order by dt)
5         then 'Better'
6         when rain > lag (rain) over (order by dt)
7         then 'Worse'
8         when rain = lag (rain) over (order by dt)
9         then 'Same'
10        end trend
11 from weather;
```

DT	RAIN	TREND
-----	-----	-----
01-AUG-14	14	
02-AUG-14	0	Better
03-AUG-14	19	Worse
04-AUG-14	6	Better
05-AUG-14	20	Worse
06-AUG-14	1	Better
07-AUG-14	17	Worse
08-AUG-14	17	Same
09-AUG-14	14	Better
10-AUG-14	18	Worse
11-AUG-14	9	Better
12-AUG-14	4	Better
13-AUG-14	17	Worse
14-AUG-14	16	Better
15-AUG-14	5	Better


```
SQL> select *
  2   from weather
  3 match_recognize
  4 (
  5   order by dt
  6   measures first (wet.dt) as first_wetday
  7             ,dry.dt as dryday
  8             ,last (wet.dt) as last_wetday
  9   one row per match
 10   pattern (wet dry{2,} wet*)
 11   define
 12     wet as wet.rain > 10,
 13     dry as dry.rain <= 10
 14 );
```

Looking for Dry Spells



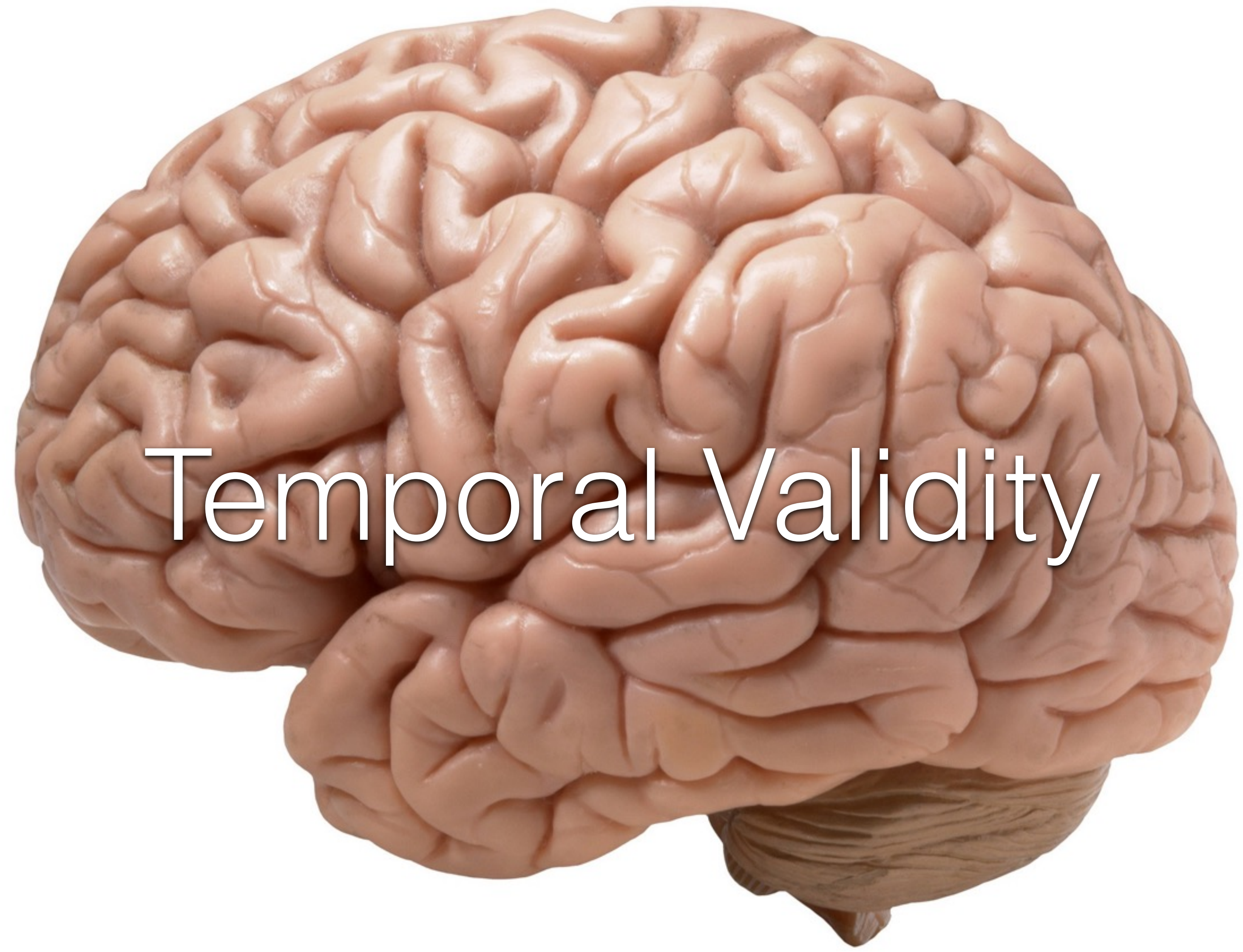
pattern (wet dry{2,} wet*)

```
FIRST_WET DRYDAY LAST_WETD
-----
10-AUG-14 12-AUG-14 14-AUG-14
23-AUG-14 26-AUG-14 27-AUG-14

2 rows selected.
```

09-AUG-14	14
10-AUG-14	18
11-AUG-14	9
12-AUG-14	4
13-AUG-14	17
14-AUG-14	16
15-AUG-14	5

21-AUG-14	15
22-AUG-14	12
23-AUG-14	18
24-AUG-14	2
25-AUG-14	5
26-AUG-14	4
27-AUG-14	15
28-AUG-14	7



Temporal Validity


```
SQL> create table addresses
  2  (empno      number
  3  ,street    varchar2(50)
  4  ,houeno    number
  5  ,start_date date
  6  ,end_date  date
  7  ,period for address_valid (start_date, end_date)
  8  );
```

Table created.

```
SQL> select street
2      , houseno
3      , start_date
4      , end_date
5      from addresses;
```

STREET	HOUSENO	START_DAT	END_DATE
-----	-----	-----	-----
Brandywine Rd	345	05-AUG-95	15-FEB-96
Spencer Run	12	16-FEB-96	30-APR-00
Via Palm Lake	741	01-MAY-00	31-JAN-13
76th Way	7616	01-FEB-13	


```
SQL> select street
2         ,houeno
3         ,start_date
4         ,end_date
5   from addresses as of period
6   for address_valid sysdate;
```

STREET	HOUSENO	START_DAT	END_DATE
-----	-----	-----	-----
76th Way	7616	01-FEB-13	

```
SQL> select street
2         ,houeno
3         ,start_date
4         ,end_date
5     from addresses
6     versions period for address_valid
7     between date '2010-06-01'
8         and sysdate
```

STREET	HOUSENO	START_DAT	END_DATE
-----	-----	-----	-----
Via Palm Lake	741	01-MAY-00	31-JAN-13
76th Way	7616	01-FEB-13	


```
SQL> begin
  2     dbms_flashback_archive.enable_at_valid_time ('CURRENT');
  3 end;
  4 /
```

PL/SQL procedure successfully completed.

```
SQL> select street
      2         ,houeno
      3         ,start_date
      4         ,end_date
      5         from addresses;
```

STREET	HOUSENO	START_DAT	END_DATE
76th Way	7616	01-FEB-13	

```
SQL> begin
  2      dbms_flashback_archive.enable_at_valid_time ('ALL');
  3  end;
  4  /
```

PL/SQL procedure successfully completed.


```
SQL> select street
2         , houseno
3         , start_date
4         , end_date
5         from addresses;
```

STREET	HOUSENO	START_DAT	END_DATE
-----	-----	-----	-----
Brandywine Rd	345	05-AUG-95	15-FEB-96
Spencer Run	12	16-FEB-96	30-APR-00
Via Palm Lake	741	01-MAY-00	31-JAN-13
76th Way	7616	01-FEB-13	

A close-up photograph of a human fingertip, showing the intricate ridges and valleys of the skin. The fingertip is positioned horizontally across the frame. The background is a soft, out-of-focus gradient of light green and yellow. Overlaid on the center of the fingertip is the text "Identity Columns" in a clean, white, sans-serif font.

Identity Columns


```
SQL> create table t
  2  (id number generated as identity
  3  ,name varchar2(35)
  4  );
```

Table created.


```
SQL> insert into t (name) values ('Alex');
```

```
1 row created.
```

```
SQL> insert into t (id, name) values (42, 'TEST');  
insert into t (id, name) values (42, 'TEST')  
*
```

ERROR at line 1:

ORA-32795: cannot insert into a **generated always** identity column

```
SQL> insert into t (id, name) values (null, 'TEST');
insert into t (id, name) values (null, 'TEST')
      *
```

ERROR at line 1:

ORA-32795: cannot insert into a generated always identity column


```
SQL> insert into t (id, name) values (default, 'TEST');
```

```
1 row created.
```

```
GENERATED  
[ ALWAYS | BY DEFAULT [ ON NULL ] ]  
AS IDENTITY [ ( identity_options ) ]
```



Similar to Sequence Options



```
SQL> create table t
  2  (id number generated as identity (start with 42 increment by 2)
  3  ,name varchar2(35)
  4  );
```

Table created.


```
SQL> insert into t
  2      (name)
  3  values
  4      ('Alex');
```

1 row created.

```
SQL> select *
  2      from t
  3  /
```

ID	NAME
42	Alex

```
SQL> insert into t
  2      (name)
  3  values
  4      ('Tim');
```

1 row created.

```
SQL> select *
  2      from t
  3  /
```

ID	NAME
42	Alex
44	Tim

Default Values


```
SQL> create table t
      2  (str varchar2(10) default 'hello'
      3  );
```

Table created.

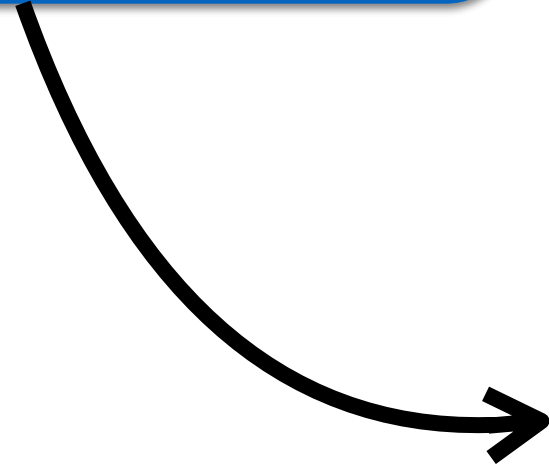
```
SQL> insert into t values (null)
      2  /
```

1 row created.

Yes, there is a row there

```
SQL> select *  
      2   from t  
      3   /
```

STR



```
SQL> insert into t values (default)  
2 /
```

```
1 row created.
```

```
SQL> select *  
2 from t  
3 /
```

```
STR
```

```
-----
```

```
hello
```



```
SQL> create table t
  2  (str varchar2(10) default on null 'Hello'
  3  );
```

Table created.

```
SQL> create sequence seq  
2 /
```

Sequence created.

```
SQL> create table t  
2 (id number default seq.nextval  
3 ,name varchar2(35)  
4 );
```

Table created.

```
SQL> insert into t (name)
      2 values ('Testing');
```

```
1 row created.
```

```
SQL> select *  
      2     from t  
      3 /
```

ID	NAME
1	Testing
42	Testing
	Testing


```
SQL> create sequence seq  
2 /
```

Sequence created.

```
SQL> create table t  
2 (id number default on null seq.nextval  
3 ,name varchar2(35)  
4 );
```

Table created.

Only when NULL



```
SQL> insert into t values (null, 'test')
  2 /
```

1 row created.

```
SQL> select * from t
  2 /
```

ID	NAME
1	test

```
SQL> create sequence master_seq  
2 /
```

Sequence created.

```
SQL> create sequence detail_seq  
2 /
```

Sequence created.

```
SQL> create table masters
  2  (id      number default master_seq.nextval
  3  ,name   varchar2(35)
  4  );
```

Table created.

```
SQL> create table details
  2  (id      number default detail_seq.nextval
  3  ,master_id number default master_seq.currval
  4  ,name   varchar2(35)
  5  );
```

Table created.


```
SQL> insert into masters (name)
  2  values ('First Master')
  3  /
```

1 row created.

```
SQL> insert into details (name)
  2  select 'Detail ' || to_char (rownum)
  3  from dual
  4  connect by level <= 5
  5  /
```

5 rows created.

```
SQL> insert into masters (name)
  2  values ('Second Master')
  3  /
```

1 row created.

```
SQL> insert into details (name)
  2  select 'Detail ' || to_char (rownum)
  3  from dual
  4  connect by level <= 5
  5  /
```

5 rows created.

```
SQL> select *  
      2     from details  
      3     /
```

ID	MASTER_ID	NAME
1	1	Detail 1
2	1	Detail 2
3	1	Detail 3
4	1	Detail 4
5	1	Detail 5
6	2	Detail 1
7	2	Detail 2
8	2	Detail 3
9	2	Detail 4
10	2	Detail 5

10 rows selected.

Subquery Factoring

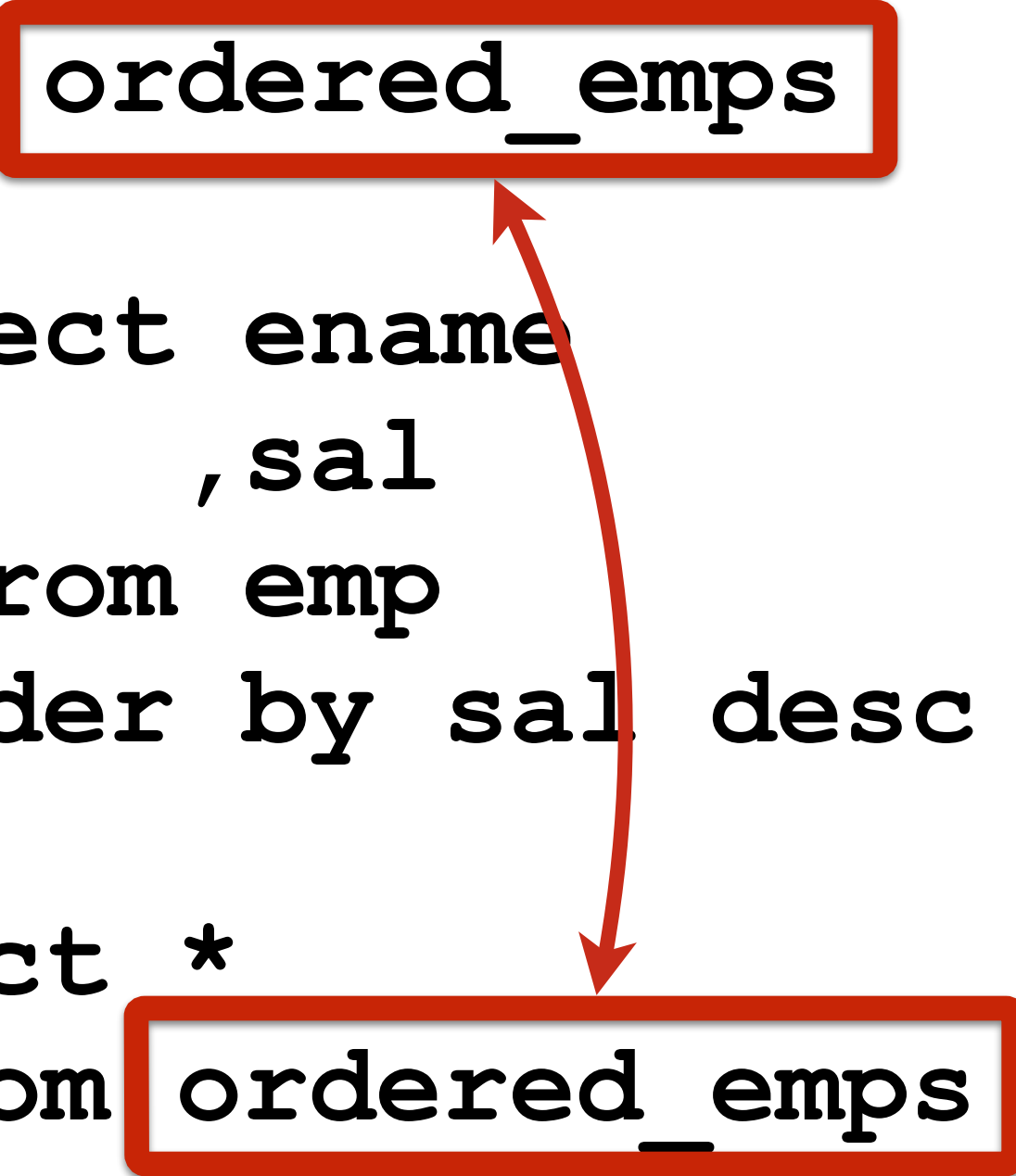


"The WITH Clause"


```
SQL> select *
  2     from (select ename
  3             ,sal
  4             from emp
  5             order by sal desc
  6           )
  7     where rownum < 4
  8     /
```

ENAME	SAL
KING	5000
SCOTT	3000
FORD	3000

```
SQL> with ordered_emps
  2     as
  3     (select ename
  4           ,sal
  5           from emp
  6           order by sal desc
  7         )
  9     select *
 10     from ordered_emps
 11     where rownum < 4
 12     /
```



ENAME	SAL
KING	5000
SCOTT	3000
FORD	3000

```
SQL> with lots(r)
2   as
3   (select 1 r from dual
4     union all
5     select r+1 from lots
6     where r < 5
7   )
8   select *
9     from lots
10  /
```

R

1
2
3
4
5

5 rows selected.

```
SQL> with
 2      function formatname (p_name in varchar2)
 3          return varchar2
 4      is
 5      begin
 6          return initcap (p_name);
 7      end formatname;
 8  select ename
 9          ,formatname(ename)   formatted
10  from emp;
```

ENAME	FORMATTED
SMITH	Smith
ALLEN	Allen
WARD	Ward
JONES	Jones
MARTIN	Martin

SQL> with

```
2 procedure show (p_what in varchar2)
3 is
4 begin
5     dbms_output.put_line (p_what);
6 end show;
```

```
7 function formatname (p_name in varchar2)
8     return varchar2
9 is
10 begin
11     show ('The input was: ' || p_name);
12     return initcap (p_name);
13 end formatname;
```

```
14 select ename
15         ,formatname (ename) formatted
16 from emp;
```


ENAME	FORMATTED
SMITH	Smith
ALLEN	Allen
WARD	Ward
JONES	Jones
MARTIN	Martin
BLAKE	Blake
CLARK	Clark
SCOTT	Scott
KING	King
TURNER	Turner
ADAMS	Adams
JAMES	James
FORD	Ford
MILLER	Miller

14 rows selected.

The input was: SMITH
The input was: ALLEN
The input was: WARD
The input was: JONES
The input was: MARTIN
The input was: BLAKE
The input was: CLARK
The input was: SCOTT
The input was: KING
The input was: TURNER
The input was: ADAMS
The input was: JAMES
The input was: FORD
The input was: MILLER

```
SQL> with
 2  procedure show (p_what in varchar2)
 3  is
 4  begin
 5      dbms_output.put_line ('input is: ' || p_what);
 6  end show;
 7  function formatname (p_name in varchar2)
 8      return varchar2
 9  is
10  begin
11      show (p_name);
12      return initcap (p_name);
13  end formatname;
14  ordered_emps as
15  (select ename from emp order by ename asc)
16  select ename
17      ,formatname(ename) formatted
18  from ordered_emps
19  /
```


ENAME	FORMATTED
ADAMS	Adams
ALLEN	Allen
BLAKE	Blake
CLARK	Clark
FORD	Ford
JAMES	James
JONES	Jones
KING	King
MARTIN	Martin
MILLER	Miller
SCOTT	Scott
SMITH	Smith
TURNER	Turner
WARD	Ward

14 rows selected.

input is: SMITH
input is: ALLEN
input is: WARD
input is: JONES
input is: MARTIN
input is: BLAKE
input is: CLARK
input is: SCOTT
input is: KING
input is: TURNER
input is: ADAMS
input is: JAMES
input is: FORD
input is: MILLER

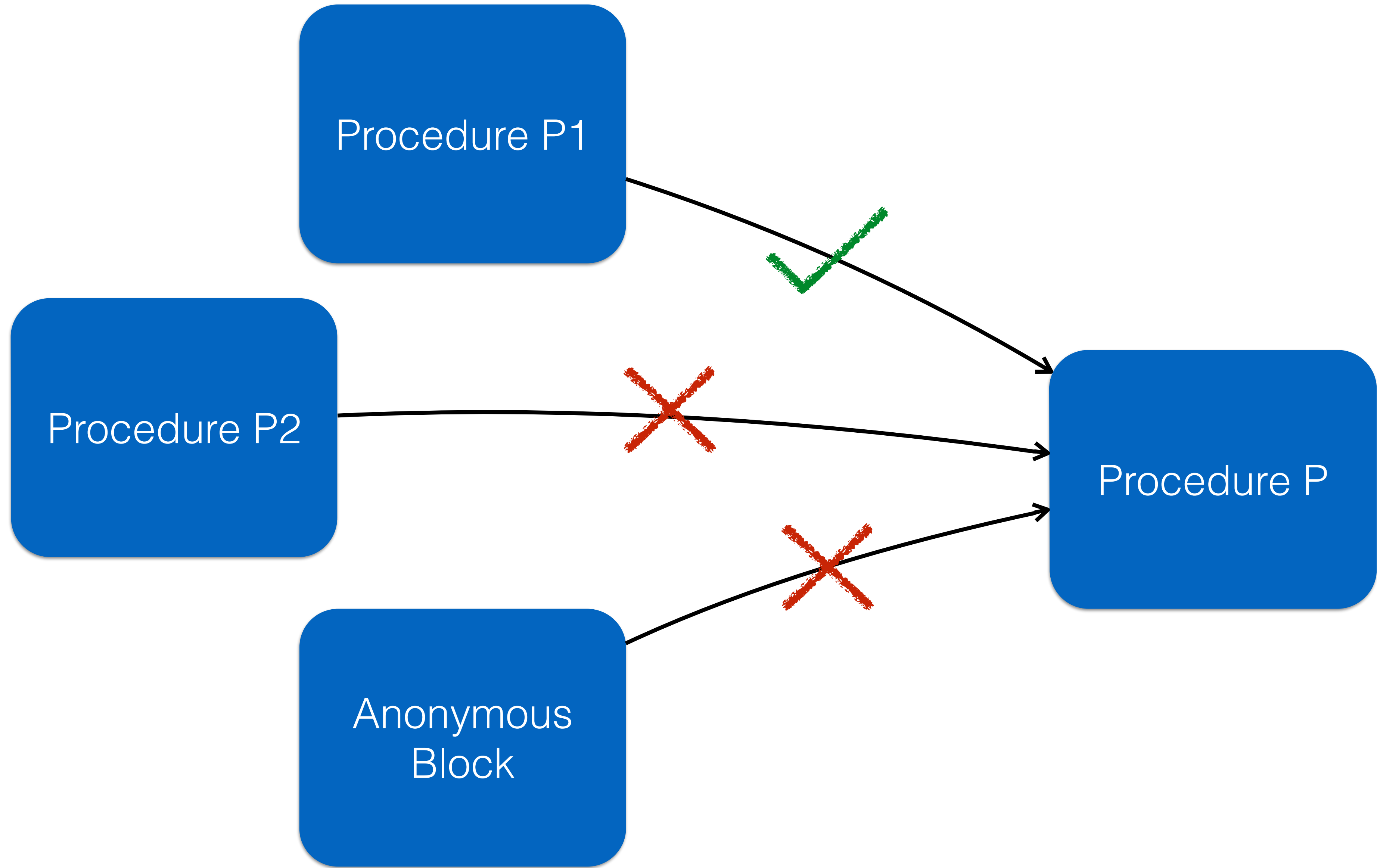
Pragma UDF

```
SQL> create or replace
  2 function formatname (p_name in varchar2)
  3     return varchar2
  4 is
  5     pragma udf;
  6 begin
  7     return initcap (p_name);
  8 end;
  9 /
```



This slide is intentionally left blank

White Listing



```
SQL> create or replace
2 procedure p
3     accessible by (p1)
4 is
5 begin
6     dbms_output.put_line
7     ('This can only be called by the p1 program');
8 end p;
9 /
```

Procedure created.

```
SQL> begin  
  2      p;  
  3  end;  
  4  /
```

```
SQL> create or replace
  2 procedure p1
  3 is
  4 begin
  5     dbms output.put_line ('This is the p1 program');
  6     p;
  7 end p1;
  8 /
```

Procedure created.

```
SQL> begin  
  2      p1;  
  3  end;  
  4  /
```

This is the p1 program

This can only be called by the p1 program

PL/SQL procedure successfully completed.


```
SQL> create or replace
  2  procedure p2
  3  is
  4  begin
  5      dbms output.put_line ('This is the p2 program');
  6      p;
  7  end p1;
  8  /
```



```
select ...  
from ...
```

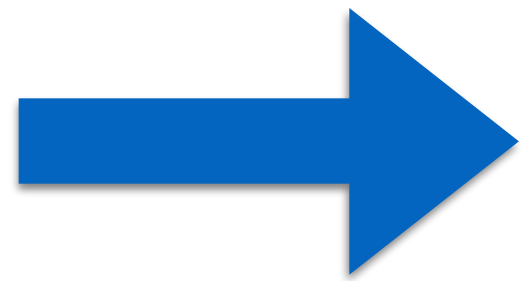
Redacted
Data



```
SQL> select ename
      2         ,hiredate
      3         ,credit_card
      4   from emp
      5 /
```

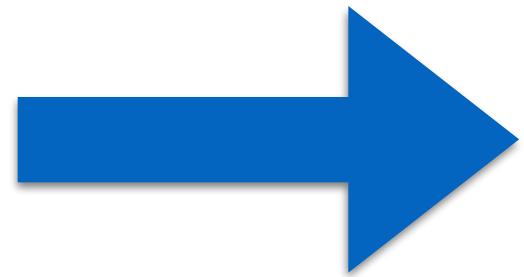
ENAME	HIREDATE	CREDIT_CARD
SMITH	17-DEC-80	4916328952854172
ALLEN	20-FEB-81	4539313944751949
WARD	22-FEB-81	4716066949870198
JONES	02-APR-81	5535742924238587
MARTIN	28-SEP-81	5570058606015920
BLAKE	01-MAY-81	5103520492409737
CLARK	09-JUN-81	377819476595275
SCOTT	19-APR-87	349683522367948

```
begin
  dbms_redact.add_policy
    (object_schema => 'ALEX'
    ,object_name   => 'EMP'
    ,policy_name   => 'Hide Creditcard'
    ,expression    => '1=1'
    ,column_name   => 'CREDIT_CARD'
    ,function_type => dbms_redact.regexp
    ,regexp_pattern => dbms_redact.re_pattern_any_digit
    ,regexp_replace_string => 'X'
    );
end;
/
```



ENAME	HIREDATE	CREDIT_CARD
SMITH	17-DEC-80	XXXXXXXXXXXXXXXXXXXX
ALLEN	20-FEB-81	XXXXXXXXXXXXXXXXXXXX
WARD	22-FEB-81	XXXXXXXXXXXXXXXXXXXX
JONES	02-APR-81	XXXXXXXXXXXXXXXXXXXX
MARTIN	28-SEP-81	XXXXXXXXXXXXXXXXXXXX
BLAKE	01-MAY-81	XXXXXXXXXXXXXXXXXXXX
CLARK	09-JUN-81	XXXXXXXXXXXXXXXXXXXX
SCOTT	19-APR-87	XXXXXXXXXXXXXXXXXXXX

```
begin
  dbms_redact.add_policy
    (object_schema => 'ALEX'
    ,object_name    => 'EMP'
    ,policy_name   => 'Hide Creditcard'
    ,expression    => '1=1'
    ,column_name   => 'CREDIT_CARD'
    ,function_type => dbms_redact.partial
    ,function_parameters => dbms_redact.redact_ccn16_f12
    );
end;
/
```

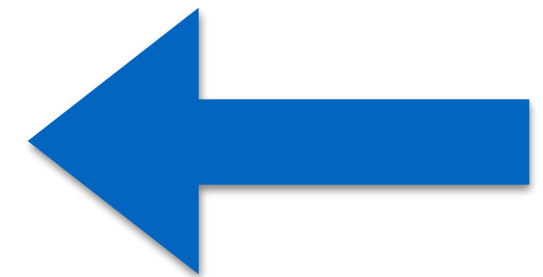


ENAME	HIREDATE	CREDIT_CARD
SMITH	17-DEC-80	****-****-****-2
ALLEN	20-FEB-81	****-****-****-9
WARD	22-FEB-81	****-****-****-8
JONES	02-APR-81	****-****-****-7
MARTIN	28-SEP-81	****-****-****-0
BLAKE	01-MAY-81	****-****-****-7
CLARK	09-JUN-81	****-****-****-
SCOTT	19-APR-87	****-****-****-

```
begin
  dbms_redact.add_policy(
    object_schema      => 'ALEX'
    ,object_name       => 'EMP'
    ,column_name       => 'CREDIT_CARD'
    ,policy_name       => 'Hide Credit Card'
    ,function_type     => dbms_redact.regexp
    ,function_parameters => null
    ,expression        => '1=1'
    ,regexp_pattern    => dbms_redact.re_pattern_cc_16_t4
    ,regexp_replace_string => dbms_redact.re_redact_cc_middle_digits
    ,regexp_position   => dbms_redact.re_beginning
    ,regexp_occurrence => dbms_redact.re_first
  );
end;
/
```

ENAME	HIREDATE	CREDIT_CARD
SMITH	17-DEC-80	491632XXXXXX4172
ALLEN	20-FEB-81	453931XXXXXX1949
WARD	22-FEB-81	471606XXXXXX0198
JONES	02-APR-81	553574XXXXXX8587
MARTIN	28-SEP-81	557005XXXXXX5920
BLAKE	01-MAY-81	510352XXXXXX9737
CLARK	09-JUN-81	377819XXXXXX5275
SCOTT	19-APR-87	349683XXXXXX7948

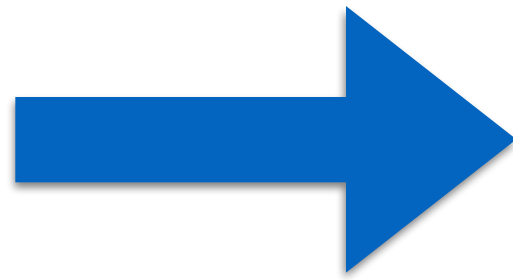

```
begin
  dbms_redact.add_policy(
    object_schema      => 'ALEX'
    ,object_name       => 'EMP'
    ,column_name       => 'CREDIT_CARD'
    ,policy_name       => 'gibberish'
    ,function_type     => DBMS_REDACT.RANDOM
    ,expression        => '1=1'
  );
END;
/
```



ENAME	HIREDATE	CREDIT_CARD
-----	-----	-----
SMITH	17-DEC-80	_z^\1LYqgbfd9Tay
ALLEN	20-FEB-81	dN9Vpp_.1VA^xxg6
WARD	22-FEB-81	?8!%` :BQG@) -hBJY
JONES	02-APR-81	\~+bn/Mad' 3jv7Ui
MARTIN	28-SEP-81	hg}BQ8etpo&JY@m
BLAKE	01-MAY-81	*PJ" ,qL2XR*(4yT
CLARK	09-JUN-81	!e5`E)g))m=hM1o
SCOTT	19-APR-87	* V]>n%y2%^eFv-

ENAME	HIREDATE	CREDIT_CARD
-----	-----	-----
SMITH	17-DEC-80	_ ;T99 ' L@gC\AA/TH
ALLEN	20-FEB-81	J/Zs) aE@R7b{1iMH
WARD	22-FEB-81	{?-Z#/b0\$G5b+7j8
JONES	02-APR-81	IKn4cRczQSv<kZk#
MARTIN	28-SEP-81	qX=>90 ;?y `EFA8CG
BLAKE	01-MAY-81	oTSNzmVhw\[V#u^p
CLARK	09-JUN-81	<\$/9d5T'D,7A1=\
SCOTT	19-APR-87	au\$0]"P i},We*X

```
begin
  dbms_redact.add_policy
    (object_schema => 'ALEX'
    ,object_name   => 'EMP'
    ,policy_name   => 'Hide Hire Date'
    ,expression    => '1=1'
    ,column_name   => 'HIREDATE'
    ,function_type => dbms_redact.partial
    ,function_parameters => dbms_redact.redact_date_millennium
    );
end;
/
```



ENAME	HIREDATE	CREDIT_CARD
SMITH	01-JAN-00	4916328952854172
ALLEN	01-JAN-00	4539313944751949
WARD	01-JAN-00	4716066949870198
JONES	01-JAN-00	5535742924238587
MARTIN	01-JAN-00	5570058606015920
BLAKE	01-JAN-00	5103520492409737
CLARK	01-JAN-00	377819476595275
SCOTT	01-JAN-00	349683522367948

Same Policy Name

```
begin
  dbms_redact.alter_policy(
    object_schema => 'ALEX'
    ,object_name  => 'EMP'
    ,policy_name  => 'Obscure some data'
    ,action       => dbms_redact.ADD_COLUMN
    ,column_name  => 'HIREDATE'
    ,function_type => dbms_redact.partial
    ,function_parameters => dbms_redact.redact_date_epoch
  );
end;
/
```



Choose Policy Name
Wisely!

ENAME

HIREDATE

CREDIT_CARD

SMITH

01-JAN-70

491632XXXXXX4172

ALLEN

01-JAN-70

453931XXXXXX1949

WARD

01-JAN-70

471606XXXXXX0198

JONES

01-JAN-70

553574XXXXXX8587

MARTIN

01-JAN-70

557005XXXXXX5920

BLAKE

01-JAN-70

510352XXXXXX9737

CLARK

01-JAN-70

377819XXXXXX5275

SCOTT

01-JAN-70

349683XXXXXX7948



a word of warning

```
select ...  
from ...
```

Redacted
Data



```
SQL> select ename
2         ,hiredate
3         ,credit_card
4     from emp
5     where substr (credit_card, 1, 1) = 4
6     /
```


ENAME	HIREDATE	CREDIT_CARD
-----	-----	-----
SMITH	17-DEC-80	XXXXXXXXXXXXXXXXXXXX
ALLEN	20-FEB-81	XXXXXXXXXXXXXXXXXXXX
WARD	22-FEB-81	XXXXXXXXXXXXXXXXXXXX

```

1 WITH
2 function show_cc (p_ename in varchar2)
3   return varchar2
4 is
5   digit varchar2(25);
6   buff varchar2(25);
7 begin
8   for cc in 1..17
9   loop
10  for i in 0..9
11  loop
12  for r_emp in (select credit_card
13                from alex.emp
14                where ename = upper (p_ename)
15                  and substr (credit_card, cc, 1) = i
16                )
17  loop
18    buff := buff ||to_char (i);
19  end loop;
20 end loop;
21 end loop;
22 return 'Credit Card for '||upper (p_ename)||' is '||buff;
23 end show_cc;
24 select ename,CREDIT_CARD, show_cc (p_ename => ename)
25 from alex.emp

```

Query Result x

SQL | All Rows Fetched: 14 in 0.363 seconds

	ENAME	CREDIT_CARD	SHOW_CC(P_ENAME=>ENAME)
1	SMITH	XXXXXXXXXXXXXXXXXX	Credit Card for SMITH is 4916328952854172
2	ALLEN	XXXXXXXXXXXXXXXXXX	Credit Card for ALLEN is 4539313944751949
3	WARD	XXXXXXXXXXXXXXXXXX	Credit Card for WARD is 4716066949870198

A full-page image of Jason Voorhees from the Friday the 13th movie series. He is standing in a dark, rainy forest at night, wearing his signature yellow hockey mask with red markings and a dark, tattered jacket. He holds a large machete in his right hand. The word "JASON" is written across his chest in a large, orange, distressed font with blood splatters. The background shows rain falling and trees in the shadows.

JASON



ORACLE
DATABASE

12^c

12.1.0.2

Store
Query
Index

{Name : Value}

$\left\{ \begin{array}{l} \text{Name : Value} \\ \text{Name : Value,} \\ \text{Name : Value,} \end{array} \right\}$

[{ Name : Value
Name : Value,
Name : Value, } , { Name : Value
Name : Value,
Name : Value, }]

```
{ "ACCOUNTING" : {  
  "EMPLOYEES" : [  
    { "ENAME" : "KING",  
      "JOB" : "PRESIDENT",  
      "SAL" : 5000  
    },  
    { "ENAME" : "MILLER",  
      "JOB" : "CLERK",  
      "SAL" : 1300  
    }  
  ]  
}  
}
```

```
SQL> create table t
  2  (json_data varchar2(4000)
  3  );
```

Table created.

```
SQL> insert into t  
      2 values ('{"valid":"json"}');
```

```
1 row created.
```



```
SQL> insert into t  
      2 values ('Just a string');
```

Json Path Expression

```
SQL> select json_value (json_data  
2      , '$.valid'  
3      )  
4      from t  
5      /
```

```
JSON_VALUE (JSON_DATA, '$.VALID')
```

```
json
```

```
insert into t
values ('{"ACCOUNTING" : {
    "EMPLOYEES" : [
        {"ENAME" : "KING",
         "JOB" : "PRESIDENT",
         "SAL" : 5000
        },
        {"ENAME" : "MILLER",
         "JOB" : "CLERK",
         "SAL" : 1300
        }
    ]
}
}') ;
```

Json Path Expression

```
SQL> select json_value (json data  
2      , '$.ACCOUNTING.EMPLOYEES[0].ENAME' )  
2      from t;
```

```
JSON_VALUE (JSON_DATA, '$.ACCOUNTING.EMPLOYEES[0].ENAME' )  
-----
```

KING

2 rows selected.

```
SQL> select ename
2         , job
3         , sal
4     from t
5         , json_table (json_data, '$.ACCOUNTING.EMPLOYEES[*]'
6             columns (ename varchar2(20) path '$.ENAME'
7                 , job   varchar2(20) path '$.JOB'
8                 , sal   number       path '$.SAL'
9             )) ;
```

ENAME	JOB	SAL
KING	PRESIDENT	5000
MILLER	CLERK	1300

2 rows selected.


```
SQL> select *
      2     from t
      3     where json_exists (json_data
      4                        , '$.*.EMPLOYEES' ) ;
```

JSON_DATA

```
{ "ACCOUNTING" : {
  "EMPLOYEES" : [
    { "ENAME" : "KING",
      "JOB" : "PRESIDENT",
      "SAL" : 5000
    },
    { "ENAME" : "MILLER",
      "JOB" : "CLERK",
      "SAL" : 1300
    }
  ]
}
```

JSON Generation

post 12.1.0.2

JSON_OBJECT
JSON_ARRAY



Callstack


```
SQL> create or replace
  2  procedure show_callstack
  3  is
  4  begin
  5      dbms_output.put_line (dbms_utility.format_call_stack());
  6  end show_callstack;
  7  /
```

Procedure created.

```
1  create or replace
2  package body pkg is
3      procedure p
4      is
5          procedure q
6          is
7              procedure r
8              is
9                  procedure p is
10                 begin
11                     show_callstack();
12                     raise program_error;
13                 end p;
14                 begin
15                     p();
16                 end r;
17                 begin
18                     r();
19                 end q;
20             begin
21                 q();
22             end p;
23         end pkg;
```



```
1 create or replace
2 package body pkg is
3     procedure p
4     is
5         procedure q
6         is
7             procedure r
8             is
9                 procedure p is
10                begin
11                    show_callstack();
12                    raise program_error;
13                end p;
14            begin
15                p();
16            end r;
17        begin
18            r();
19        end q;
20    begin
21        q();
22    end p;
23 end pkg;
```

```
1 create or replace
2 package body pkg is
3     procedure p
4     is
5         procedure q
6         is
7             procedure r
8             is
9                 procedure p is
10                begin
11                    show_callstack();
12                    raise program_error;
13                end p;
14            begin
15                p();
16            end r;
17        begin
18            r();
19        end q;
20    begin
21        q();
22    end p;
23 end pkg;
```

```
1 create or replace
2 package body pkg is
3     procedure p
4     is
5         procedure q
6         is
7             procedure r
8             is
9                 procedure p is
10                begin
11                    show_callstack();
12                    raise program_error;
13                end p;
14                begin
15                    p();
16                end r;
17            begin
18                r();
19            end q;
20        begin
21            q();
22        end p;
23    end pkg;
```

```
1 create or replace
2 package body pkg is
3     procedure p
4     is
5         procedure q
6         is
7             procedure r
8             is
9                 procedure p is
10                begin
11                    show_callstack();
12                    raise program_error;
13                end p;
14            begin
15                p();
16            end r;
17        begin
18            r();
19        end q;
20    begin
21        q();
22    end p;
23 end pkg;
```


----- PL/SQL Call Stack -----

object handle	line number	object name
0x9faa8f18	4	procedure ALEX.SHOW_CALLSTACK
0x9dbd2c00	10	
		package body ALEX.PKG
0x9dbd2c00	14	package body ALEX.PKG
0x9dbd2c00		
17		package body ALEX.PKG
0x9dbd2c00	20	package body ALEX.PKG
0x9fc73e18		
2		anonymous block

begin

*

ERROR at line 1:

ORA-06501: PL/SQL: program error

ORA-06512: at "ALEX.PKG", line 11

ORA-06512: at "ALEX.PKG", line 14

ORA-06512: at "ALEX.PKG", line 17

ORA-06512: at "ALEX.PKG", line 20

ORA-06512: at line 2

UTL_CALL_STACK

```
SQL> create or replace
  2  procedure show_callstack
  3  as
  4      depth pls_integer := utl_call_stack.dynamic_depth();
  5      procedure headers
  ..
 11      end headers;
 12  begin
 13      headers;
 14      for j in reverse 1..depth loop
 15          dbms_output.put_line(
 16              rpad( utl_call_stack.lexical_depth(j), 10 ) ||
 17                  rpad( j, 7) ||
 18                  rpad( to_char(utl_call_stack.unit_line(j), '99'), 9 ) ||
 19                  utl_call_stack.concatenate_subprogram
 20                      (utl_call_stack.subprogram(j)) );
 21      end loop;
 22  end show_callstack;
```

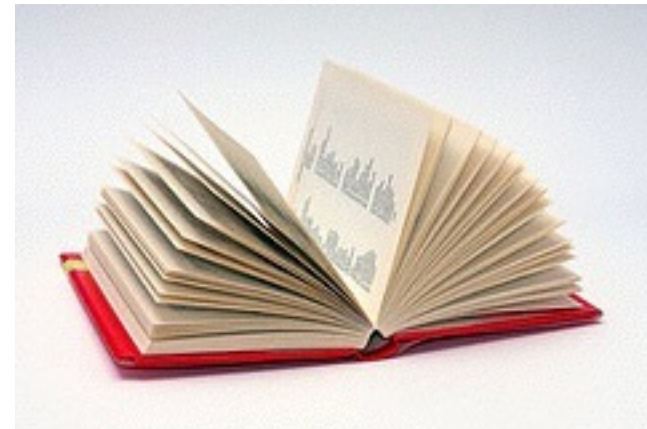
```
SQL> create or replace
  2  procedure show_callstack
  3  as
  4  depth pls_integer := utl_call_stack.dynamic_depth();
  5  procedure headers
  ..
 11  end headers;
 12  begin
 13  headers;
 14  for j in reverse 1..depth loop
 15  dbms_output.put_line(
 16  rpad( utl_call_stack.lexical_depth(j), 10 ) ||
 17  rpad( j, 7) ||
 18  rpad( to_char(utl_call_stack.unit_line(j), '99'), 9 ) ||
 19  utl_call_stack.concatenate_subprogram
 20  (utl_call_stack.subprogram(j)));
 21  end loop;
 22  end show_callstack;
```


Lexical Depth	Depth Number	Line	Name
-----	-----	-----	-----
0	6	2	__anonymous_block
1	5	20	PKG.P
2	4	17	PKG.P.Q
3	3	14	PKG.P.Q.R
4	2	10	PKG.P.Q.R.P
0	1	14	SHOW_CALLSTACK

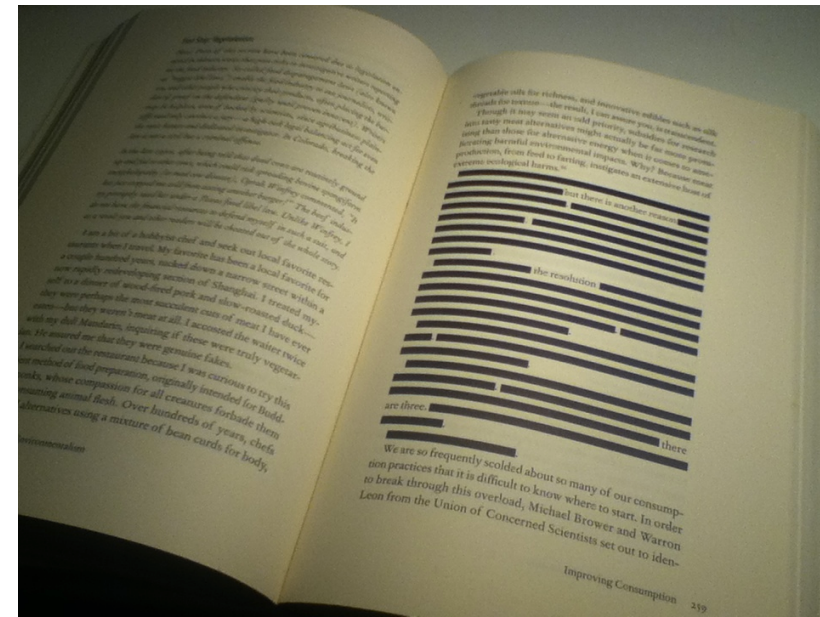
```
begin
*
```

ERROR at line 1:

```
ORA-06501: PL/SQL: program error
ORA-06512: at "ALEX.PKG", line 11
ORA-06512: at "ALEX.PKG", line 14
ORA-06512: at "ALEX.PKG", line 17
ORA-06512: at "ALEX.PKG", line 20
ORA-06512: at line 2
```

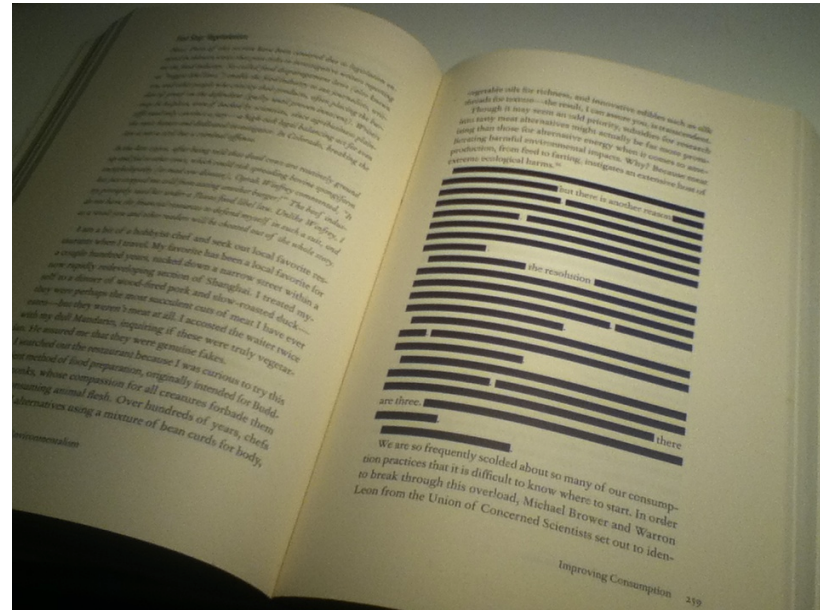
Subquery Factoring





Geoffrey Fairchild

<https://flic.kr/p/6tKgHz>



Mutant669

<http://bit.ly/1tqcAJu>



Monica Arellano-Ongpin

<https://flic.kr/p/cZ6YT9>



James MacDonald

<https://flic.kr/p/9gX5fM>



DJ

<https://flic.kr/p/dLbzPm>



aussiegall

<https://flic.kr/p/rksBi>



<http://savethestack.blogspot.com>



Horia Varlan

<https://flic.kr/p/7vedzj>



New Line Cinema

Alex Nuijten
Alex.Nuijten@Ordina.nl

nuijten.blogspot.com
@alexnuijten

