

# 12 Things Developers Will Love About Oracle Database 12c Release 2

ORACLE  
OPEN  
WORLD

October 1–5, 2017  
SAN FRANCISCO, CA

Chris Saxon  
Oracle Developer Advocate for SQL  
[blogs.oracle.com/sql](http://blogs.oracle.com/sql)  
[@chrisrsaxon](https://twitter.com/chrisrsaxon) & [@SQLDaily](https://twitter.com/SQLDaily)  
[youtube.com/c/TheMagicOfSQL](https://www.youtube.com/c/TheMagicOfSQL)

ORACLE

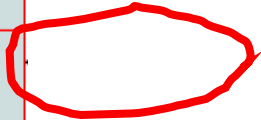
## Safe Harbor Statement

The following is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, and timing of any features or functionality described for Oracle's products remains at the sole discretion of Oracle.

*"There are only two hard things in  
Computer Science: ~~cache invalidation~~  
and naming things."*

*– Phil Karlton*

CUSTOMERS	
P *	CUSTOMER_ID



ADDRESSES	
P *	ADDRESS_ID

`<table_name>_<column_name>_fk`

```
alter table customer_addresses add constraint
customer_addresses_customer_id_fk
foreign key ( customer_id )
references customers ( customer_id );
```

**SQL Error: ORA-00972: identifier is too long**



## Increase maximum identifier length from 30 characters to 60 or more

Created on 22-Mar-2015 15:35 by MortenBraten - Last Modified: 02-Jun-2016 06:10

320

You have not voted

COMING SOON

Currently the maximum identifier length of tables, columns, packages, functions, etc. is limited to 30 characters.

This should be increased to 60, 90 or more characters, allowing for more flexibility and readability in object naming.

Realize this has the potential to break a lot of old applications, so it should probably be explicitly enabled by a configuration parameter, just like the "32K varchars in tables" option that was introduced in 12c.

Tags:



128 *bytes*



```
1 create table
2   with_a_really_really_really_really_really_long_name (
3     and_lots_and_lots_and_lots_and_lots          int,
4     of_really_really_really_really_really_long_cols int
5 );
6
```



```
create or replace procedure process_tab
is
    tab varchar2(30);
begin

    select table_name
    into   tab
    from   user_tables
    where  ...
```

```
exec process_tab;
```

```
ORA-06502: PL/SQL: numeric or value  
error: character string buffer too  
small
```



<expressions> in  
place of 'literals'

declare

```
v1 varchar2 ( 30 );
```

```
var_length constant pls_integer := 30;
```

```
v2 varchar2( var_length );
```

begin

Must know at  
*compile* time



✘ SQL

✘ User functions

✔ Constants

create or replace procedure process\_tab is

```
$if DBMS_DB_VERSION.VER_LE_12_1 $then
  tab varchar2 ( 30 ),
$else
  tab varchar2 ( ora_max_name_len );
$end

begin
```

12.1

12.2+

Only one is  
compiled

`user_tables.table_name%type`

```
select table_name,  
       listagg(index_name, ',') within group  
         (order by index_name) inds  
from   user_indexes  
group  by table_name;
```

4,000\* byte limit  
~~130~~ indexes/tab  
31



# "Self documenting" indexes...

```
create index
```

```
reducing_the_monthly_invoice_run_
```

```
from_four_hours_to_three_minutes_
```

```
PROJ12345_make_everything_faster_
```

```
csaxon_is_the_sql_tuning_master on
```

```
...
```

```
select table_name,  
       listagg(index_name, ',') within group  
         (order by index_name) inds  
from   user_indexes  
group by table_name;
```

**ORA-01489: result of string concatenation is  
too long**



```
select table_name,  
       listagg(index_name, ', '  
               on overflow truncate  
               ) within group  
         (order by index_name) inds  
from   user_indexes  
group  by table_name;  
  
...lots_and_lots_of_indexes, ... (42)
```

```
listagg (  
  things, ', '  
  [ on overflow (truncate|error) ]  
  [ text ] [ (with|without) count ]  
) within group (order by cols)
```

First name

Last name



1 `select * from people;`

2

3 `FIRST_NAME LAST_NAME`

4 `Chris Saxon`

5 `CHRIS SAXON`

6 `chrIS SAxon`

7

8

9

10

11

12

13

14

15

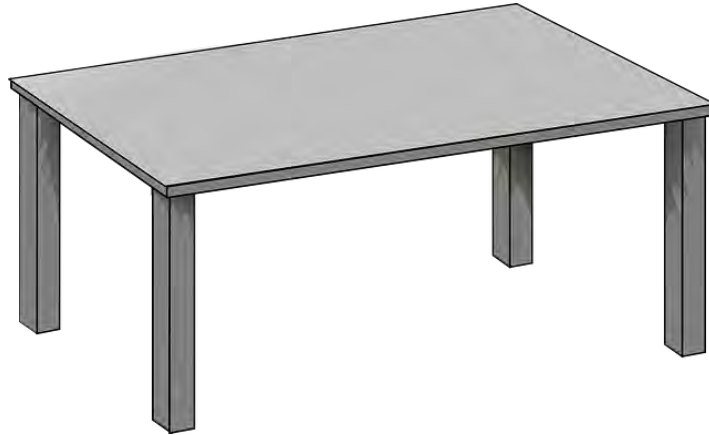
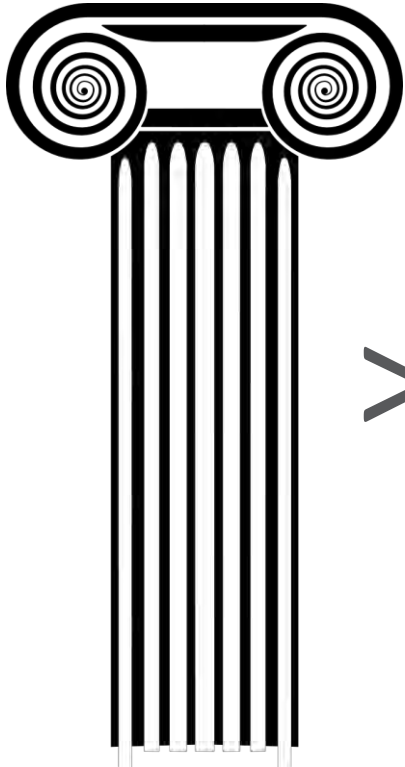
16





```
create table t (  
  case_and_accent_sensitive varchar2(10),  
  case_insensitive_accent_sensitive  
    varchar2(10) collate binary_ci,  
  case_and_accent_insensitive  
    varchar2(10) collate binary_ai  
);
```

```
alter table people modify (  
    first_name collate binary_ai,  
    last_name collate binary_ai  
);
```



# Existing columns unaffected!

```
alter table people default collation binary_ai;
```

```
ORA-43929: Collation cannot be specified if  
parameter MAX_STRING_SIZE=STANDARD
```

Must be **EXTENDED!**

create or replace package customer\_pkg as

```
/* get_customers_upper is deprecated. Do not use or  
I'll come for you with a rusty spoon! */
```

```
cursor get_customers_upper (p_name varchar2) is  
select * from customers  
where upper(customer_name) = upper(p_name);
```

```
cursor get_customers (p_name varchar2) is  
select * from customers  
where customer_name = p_name;
```



```
create or replace package customer_pkg as
/* get_customers_upper is deprecated. Do not use or
   I'll come for you with a rusty spoon */
cursor get_customers_upper (p_name varchar2) is
select * from customers
where upper(customer_name) = upper(p_name);
pragma deprecate(get_customers_upper,
  'Use get_customers instead');
cursor get_customers (p_name varchar2) is
select * from customers
where customer_name = p_name;
```

```
alter system set plsql_warnings =  
  'enable: (6019,6020,6021,6022) ' ;
```



SP2-0804: Procedure created **with compilation warnings**

LINE/COL ERROR

```
-----  
1/1      PLW-05018: unit PROCESS_CUSTOMERS omitted optional AUTHID  
         clause; default value DEFINER used  
4/29     PLW-06020: reference to a deprecated entity:  
         GET_CUSTOMERS_UPPER declared in unit CUSTOMER_PKG[2,10].  
         Use get_customers instead  
7/9     PLW-06020: reference to a deprecated entity:  
         GET_CUSTOMERS_UPPER declared in unit CUSTOMER_PKG[2,10].  
         Use get_customers instead  
8/9     PLW-06020: reference to a deprecated entity:
```

# Whatever

```
alter system set plsql_warnings =  
'error:6020';
```

Warning: Procedure altered with **compilation errors**.

LINE/COL ERROR

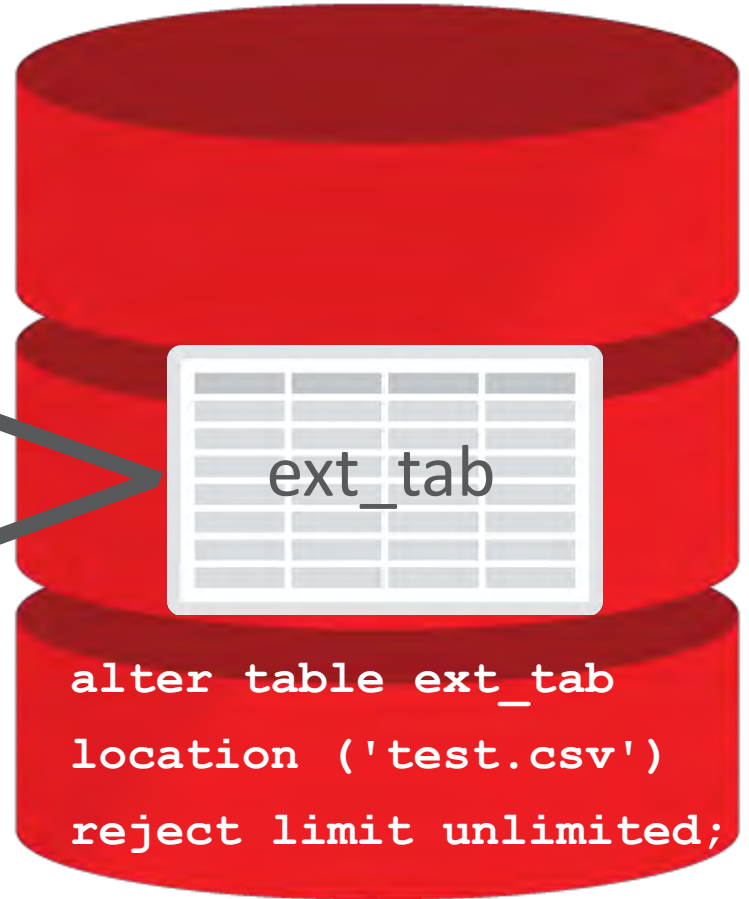
```
-----  
4/29      PLS-06020: reference to a deprecated entity:  
          GET_CUSTOMERS_UPPER declared in unit CUSTOMER_PKG[2,10].  
          Use get_customers instead  
7/9      PLS-06020: reference to a deprecated entity:  
          GET_CUSTOMERS_UPPER declared in unit CUSTOMER_PKG[2,10].  
          Use get_customers instead  
8/9      PLS-06020: reference to a deprecated entity:  
          GET_CUSTOMERS_UPPER declared in unit CUSTOMER_PKG[2,10].  
          Use get_customers instead
```



```
select * from ext_tab;
```

**ORA-29913: error in executing  
ODCIEXTTABLEFETCH callout**

**ORA-30653: reject limit reached**



```
alter table ext_tab  
location ('test.csv')  
reject limit unlimited;
```





```
select * from ext_tab external modify (  
    [ default directory ]  
    [ location ]  
    [ access parameters ]  
    [ reject limit ]  
);
```

```
select * from ext_tab external modify (  
    location ('test.csv')  
    reject limit unlimited  
);
```

no rows selected

2016-09-01			
2016-09-02			
2016-09-03			
2016-09-04			
2016-09-05			
2016-09-06			
2016-09-07			
2016-09-08			
2016-09-09			
2016-09-10			
2016-09-11			

```
create table ext_tab (  
  ...  
  created_date date )  
  ...  
) organization external (  
  ...  
  created_date char(20)  
  date_format date mask 'dd-MON-yyyy'  
  ...
```


```
select to_date(created_date, 'dd-mon-yyyy')  
from ext_tab;
```

**ORA-01843: not a valid month**



# Defaults to NLS settings!

```
validate_conversion (  
  <expression> as <data type>,  
  [ <format mask> ],  
  [ <nls_parameters> ]  
)
```



1 = Success!

0 = Failure 😞

```
select to_date(created_date, 'dd-mon-yyyy')  
from ext_tab  
where validate_conversion (  
    created_date as date, 'dd-MON-yyyy'  
) = 1;
```



```
insert into error_table
  select created_date
  from   ext_tab
  where  validate_conversion (
    created_date as date, 'dd-MON-yyyy'
  ) = 0;
```



this...

...must match this

```
cast (  
  <expression> as <data type>  
  [ default <value> on conversion error ],  
  [ <format mask> ],  
  [ <nls_parameters> ]  
)
```

Also...

```
to_date()  
to_number()  
to_timestamp()  
etc.
```

```
select to_date(  
    created_date  
    default '01-JAN-0001'  
    on conversion error,  
    'dd-mon-yyyy'  
)  
from ext_tab;
```

01-JAN-0001

```
create table file_imports (  
  file_type varchar2(30),  
  file_data clob )  
partition by list (file_type) (  
  partition p_fixed values ('FIXED'),  
  partition p_csv values ('CSV')  
);
```

```
insert into file_imports values ('XML',  
'<?xml version="1.0" encoding="UTF-8"?> ...');
```

**SQL Error: ORA-14400: inserted partition key does not map to any partition**

```
alter table file_imports add partition  
p_xml values ('XML');
```

```
alter table file_imports add partition  
p_default values (default);
```

p\_fixed



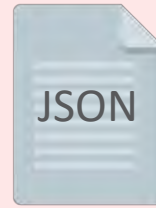
p\_csv



p\_XML



p\_default







```
create table file_imports (  
  file_type varchar2(30),  
  file_data clob )  
partition by list (file_type) automatic (  
partition p_fixed values ('FIXED'),  
partition p_csv values ('CSV'),  
partition p_XML values ('XML'),  
partition p_JSON values ('JSON'),  
partition p_default values (default)  
);
```

Mutually  
exclusive!



```
alter table file_imports split partition
p_default into (
    partition p_json values ('JSON'),
    partition p_yaml values ('YAML'),
    partition p_default
);
```

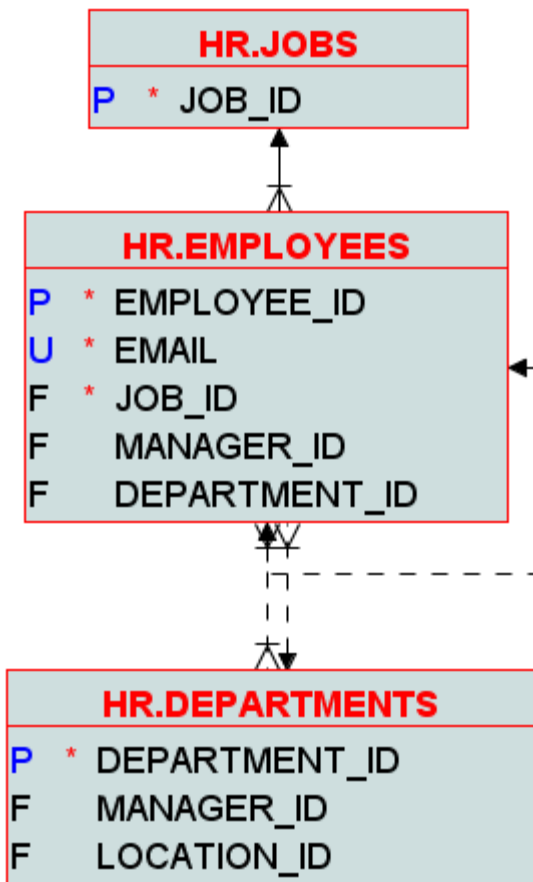
```
alter table file_imports drop partition
p_default;
```

```
alter table file_imports
set partitioning automatic;
```

# JSON in 12.1.0.2

```
{  
  "department": "Accounting",  
  "employees": [  
    {  
      "name": "Shelley,Higgins",  
      "job": "Accounting Manager"  
    },  
    {  
      "name": "William,Gietz",  
      "job": "Public Accountant"  
    }  
  ]  
}  
check ( employees_json is json )
```





```
{
  "department": "Accounting",
  "employees": [
    {
      "name": "Shelley,Higgins",
      "job": "Accounting Manager"
    },
    {
      "name": "William,Gietz",
      "job": "Public Accountant"
    }
  ]
}
```

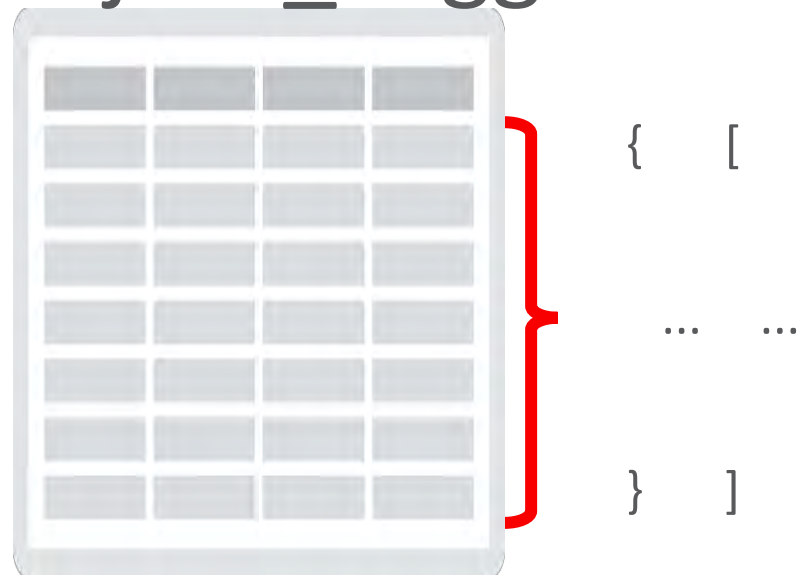
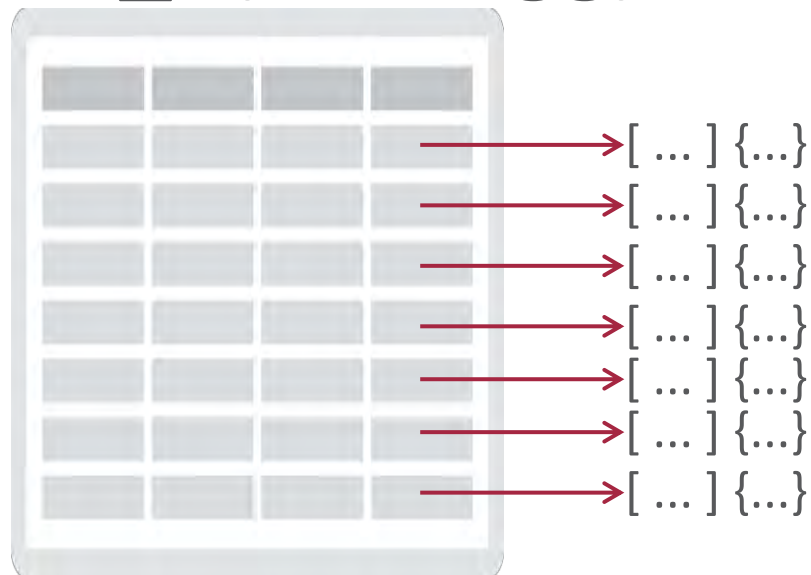


json\_array\* => [ ]

json\_object\* => { }

json\_\*(non-agg) =>

json\_\*agg =>



```
select json_object (  
    'department' value d.department_name,  
    'employees' value json_arrayagg (  
        json_object (  
            'name' value first_name || ',' || last_name,  
            'job' value job_title  
        ))  
    ))  
from hr.departments d, hr.employees e, hr.jobs j  
where d.department_id = e.department_id  
and e.job_id = j.job_id  
and d.department_id = 110  
group by d.department_name
```

json\_exists

JSON Partitioning

Search index

Materialized Views

... and more **JSON** goodies

GeoJSON

In-Memory support

JSON Data Guide



Also in 12.1.0.2...

```
select count(distinct ...)  
from   my_super_massive_table;
```



Give me  
your best  
estimate!



```
select approx_count_distinct (...)  
from my_super_massive_table;
```

*faster* &  
close enough

# 1,723



```
select count(*)  
from user_source  
where upper(text)  
like '%COUNT%DISTINCT%';
```





```
alter session set approx_for_count_distinct = true;
```

```
count ( distinct ... )  approx_count_distinct ( ... )
```

```
approx_for_aggregation Huh?  
approx_for_percentile
```

Also **approx\_median**

```
approx_percentile (  
  <expression> [ deterministic ],  
  [ ('ERROR_RATE' | 'CONFIDENCE') ]  
) within group ( order by <expr> )
```

Consistent;  
slower;  
numbers only

How certain are we?

It's time to  
start **testing!**

A female scientist with dark hair tied back, wearing a white lab coat and a brown skirt, is leaning over a laboratory bench. She is looking through the eyepiece of a white and black microscope. The lab bench is cluttered with various pieces of laboratory equipment, including a rack of test tubes with blue caps, a glass beaker with green liquid, and several bottles. In the background, there are large windows letting in bright light, and a computer monitor is visible on the right side of the frame. The overall scene is a professional laboratory environment.

...but have we  
tested **everything?**





```
begin
  if this or that then
    do_this ();
  else
    do_that ();
  end if;
end;
```

```
begin
```

```
    dbms_plsql_code_coverage.create_coverage_tables;
```

```
end;
```

```
declare
    run_id pls_integer;
begin
    run_id :=
        dbms_plsql_code_coverage.start_coverage ('TEST1');

    your_test_procedures;

    dbms_plsql_code_coverage.stop_coverage;
end;
/
```

```
select u.owner, u.name, u.type,  
       round( (sum(b.covered)*100) /count(*) )  
       as pct_covered  
from   dbmspcc_runs r  
join   dbmspcc_units u  
on     r.run_id = u.run_id  
join   dbmspcc_blocks b  
on     b.object_id = u.object_id  
and    b.run_id = u.run_id  
where  r.run_comment = 'TEST1'  
group by u.owner, u.name, u.type
```

```
1 create or replace procedure my_perfect_code is
2 begin
3
4     the_most_awesome_code ();
5
6     you_ever_saw ();
7
8     no_testing_needed ();
9
10 end my_perfect_code;
11 /
```



```
select u.owner, u.name, u.type,  
       round((sum(b.covered)*100)/count(*))  
         as pct_covered  
from   dbmspcc_runs r  
join   dbmspcc_units u  
on     r.run_id = u.run_id  
join   dbmspcc_blocks b  
on     b.object_id = u.object_id  
and    b.run_id = u.run_id  
where  r.run_comment = 'TEST1' and not_feasible = 0  
group by u.owner, u.name, u.type
```

[devgym.oracle.com/devgym/  
database-for-developers.html](https://devgym.oracle.com/devgym/database-for-developers.html)



#Make**Data**GreatAgain



## Safe Harbor Statement

The preceding is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, and timing of any features or functionality described for Oracle's products remains at the sole discretion of Oracle.

# Integrated Cloud

## Applications & Platform Services