

# Stranger SQL

Philipp Salvisberg  
20<sup>th</sup> September 2024





Wife

Darling, can you please go to the shop buy one bottle of milk and if they have eggs, get a dozen!

Gladly, I'll be right back.



Husband,  
Working in IT

30 minutes later.

Why on earth did you get 12 bottles of milk?

Well... they had eggs.

# A Piece Now – Or More At the End



# Welcome



## Philipp Salvisberg

Founder, Owner and CEO of Grisselbav GmbH

- Database-centric Development
- Model-driven Software Development
- Open-Source Development

[philipp.salvisberg@grisselbav.com](mailto:philipp.salvisberg@grisselbav.com)

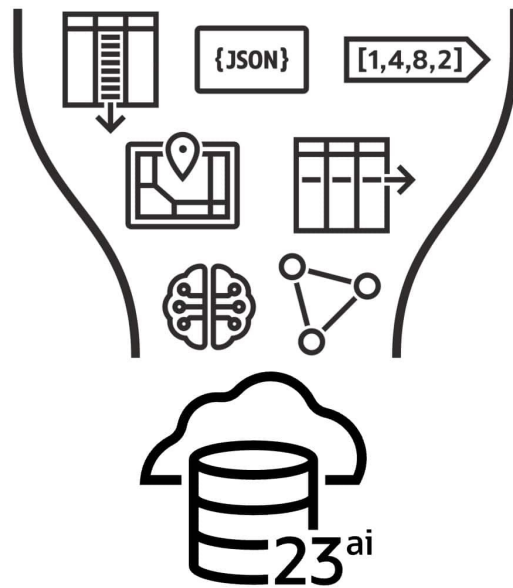
<https://www.salvis.com/blog/>

# Why This Talk?

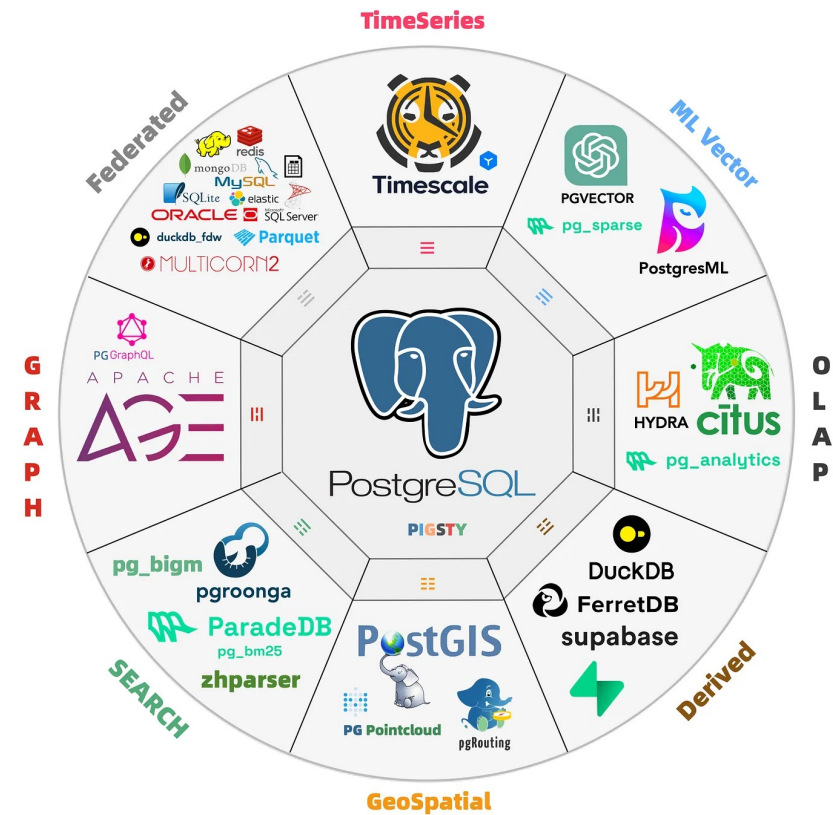
# IslandSQL – Parser for SQL Scripts



## Oracle Database 23ai



## PostgreSQL 16/17



<https://blogs.oracle.com/database/post/oracle-database-23ai-now-available-in-the-cloud>

<https://medium.com/@fengruohang/postgres-is-eating-the-database-world-157c204dcfc4>

# Comments

--

/\* \*/

## Setup

```
create table if not exists dept as select * from (values
  (10, 'ACCOUNTING', 'NEW YORK'),
  (20, 'RESEARCH', 'DALLAS'),
  (30, 'SALES', 'CHICAGO'),
  (40, 'OPERATIONS', 'BOSTON')
) s (deptno, dname, loc);
```

**SOL:2023**

```
create table if not exists emp as select * from (values
  (7839, 'KING', 'PRESIDENT', null, date '1981-11-17', 5000, null, 10),
  (7566, 'JONES', 'MANAGER', 7839, date '1981-04-02', 2975, null, 20),
  (7698, 'BLAKE', 'MANAGER', 7839, date '1981-05-01', 2850, null, 30),
  (7782, 'CLARK', 'MANAGER', 7839, date '1981-06-09', 2450, null, 10),
  (7788, 'SCOTT', 'ANALYST', 7566, date '1987-04-19', 3000, null, 20),
  (7902, 'FORD', 'ANALYST', 7566, date '1981-12-03', 3000, null, 20),
  (7499, 'ALLEN', 'SALESMAN', 7698, date '1981-02-20', 1600, 300, 30),
  (7521, 'WARD', 'SALESMAN', 7698, date '1981-02-22', 1250, 500, 30),
  (7654, 'MARTIN', 'SALESMAN', 7698, date '1981-09-28', 1250, 1400, 30),
  (7844, 'TURNER', 'SALESMAN', 7698, date '1981-09-08', 1500, 0, 30),
  (7900, 'JAMES', 'CLERK', 7698, date '1981-12-03', 950, null, 30),
  (7934, 'MILLER', 'CLERK', 7782, date '1982-01-23', 1300, null, 10),
  (7369, 'SMITH', 'CLERK', 7902, date '1980-12-17', 800, null, 20),
  (7876, 'ADAMS', 'CLERK', 7788, date '1987-05-23', 1100, null, 20)
) s (empno, ename, job, mgr, hiredate, sal, comm, deptno);
```



# Nested Multiline Comment

OracleDB

```
/*
  A nested multi-line comment:

  /* the next statement should
     not be executed, right? */

  drop table emp;
*/
select count(*) from emp;
```

Table EMP dropped.

Error starting at line : 8 in command -

```
*/
```

Error report -

Unknown Command

PostgreSQL

```
/*
  A nested multi-line comment:

  /* the next statement should
     not be executed, right? */

  drop table emp;
*/
select count(*) from emp;
```

```
count
-----
      14
(1 row)
```

**SQL:2023**

# Unterminated, Nested Comment

OracleDB

```
/*  
  A nested multi-line comment:  
  
  /* the next statement should  
     not be executed, right?  
  
drop table emp;  
*/  
select count(*) from emp;
```

COUNT(\*)

-----  
14

PostgreSQL

```
/*  
  A nested multi-line comment:  
  
  /* the next statement should  
     not be executed, right?  
  
drop table emp;  
*/  
select count(*) from emp;
```

(no result, waiting for the comment  
to be terminated)

# Quoted Identifiers

# Unquoted vs. Quoted Identifiers

```

-- setup
drop table if exists t;
create table t as
select * from (values
  ('lower', 'UPPER')
) as s("a", "A");

-- demo
select a      as c1,
       A      as c2,
       "a"    as c3,
       "A"    as c4
from t;
```

```

select * from t;
```

a	A
-----	-----
lower	UPPER

What's the  
result of c1, c2?

## OracleDB

```
-- setup
drop table if exists t;
create table t as
select * from (values
  ('lower', 'UPPER')
) as s("a", "A");

-- demo
select a      as c1,
       A      as c2,
       "a"    as c3,
       "A"    as c4
from t;
```

C1	C2	C3	C4
UPPER	UPPER	lower	UPPER

**SQL:2023**

## PostgreSQL

```
-- setup
drop table if exists t;
create table t as
select * from (values
  ('lower', 'UPPER')
) as s("a", "A");

-- demo
select a      as c1,
       A      as c2,
       "a"    as c3,
       "A"    as c4
from t;
```

c1	c2	c3	c4
lower	lower	lower	UPPER

(1 row)

# Identifiers & Operators

# What's the Result of C2?

```
OracleDB
-- setup
drop table if exists t;
create table t as
select * from (values
  (1, 2, 41, 42)
) as s(a, b, "a#b", "A#B");

-- demo
select a-b as c1,
       a#b as c2
from t;
```

C1	C2
-1	42

```
PostgreSQL
-- setup
drop table if exists t;
create table t as
select * from (values
  (1, 2, 41, 42)
) as s(a, b, "a#b", "A#B");

-- demo
select a-b as c1,
       a#b as c2
from t;
```

c1	c2
-1	3

(1 row)

# Characters in Unquoted Identifiers

Character Group	First Character			Subsequent Characters		
	OracleDB	PostgreSQL	SQL:2023	OracleDB	PostgreSQL	SQL:2023
Latin Letter (A-Z, a-z)	✓	✓	✓	✓	✓	✓
Digit (0-9)	✗	✗	✗	✓	✓	✓
Hash Symbol (#)	✗	✗	✗	✓	✗	✗
Underscore (_)	✗	✓	✗	✓	✓	✓
Dollar Sign (\$)	✗	✗	✗	✓	✓	✗
Other Letter (äàáψ, ...)	✓	✓	✗	✓	✓	✗
Emoji (😄😎🤓, ...)	✗	✓	✗	✗	✓	✗



# Keywords & Identifiers

# What's "wrong"?

```
OracleDB
select count(*) as wrong_count
  from (select * from emp)
  wrong join dept using (deptno);
```

```
WRONG_COUNT
-----
              14
```

```
PostgreSQL
select count(*) as wrong_count
  from (select * from emp)
  wrong join dept using (deptno);
```

```
wrong_count
-----
              14
(1 row)
```

# What's "right"?

```
OracleDB  
select count(*) as right_count  
  from (select * from emp)  
 right join dept using (deptno);
```

```
RIGHT_COUNT  
-----  
              15
```

```
PostgreSQL  
select count(*) as right_count  
  from (select * from emp)  
 right join dept using (deptno);
```

```
right_count  
-----  
              15  
(1 row)
```

# What's "right" Now?

```
OracleDB  
with right as (select * from emp)  
select count(*) as right_count  
  from right right  
right join dept using (deptno);
```

```
RIGHT_COUNT  
-----  
15
```

```
PostgreSQL  
with right as (select * from emp)  
select count(*) as right_count  
  from right right  
right join dept using (deptno);
```

```
ERROR:  syntax error at or near "right"  
LINE 1: with right as (select * from emp)
```

*SQL:2023*

# String Literals

## OracleDB

```
select 'Hello World!';
```

```
select 'Hello '  
       'World!';
```

```
select 'Hello ' || 'World!';
```

```
'HELLOWORLD!  
-----  
Hello World!
```

```
Error starting at line : 3 in command -
```

```
select 'Hello '  
       'World!'
```

```
Error at Command Line : 4 Column : 8
```

```
Error report -
```

```
SQL Error: ORA-00923: FROM keyword not found  
where expected
```

```
'HELLO' || 'WO  
-----  
Hello World!
```

## PostgreSQL

```
select 'Hello World!';
```

```
select 'Hello '  
       'World!';
```

```
select 'Hello ' || 'World!';
```

```
?column?  
-----  
Hello World!  
(1 row)
```

```
?column?  
-----  
Hello World!  
(1 row)
```

```
?column?  
-----  
Hello World!  
(1 row)
```

Is this  
SQL:2023  
compliant?

# Apostrophe in String

```
select '1: ' ' 2: '' ' 3: '''' ' as str;
```

STR
1: ' 2: '' 3: ''''

SQL:2023

Is there a better way?

## OracleDB

```
select q'[1: ' 2: '' 3: '''] as str;
select q'(1: ' 2: '' 3: ''') as str;
select q'{1: ' 2: '' 3: ''}' as str;
select q'<1: ' 2: '' 3: ''>' as str;
select q'~1: ' 2: '' 3: ''~' as str;
select q'#1: ' 2: '' 3: ''#' as str;
select q'@1: ' 2: '' 3: ''@' as str;
select q'$1: ' 2: '' 3: ''$' as str;
select q'£1: ' 2: '' 3: ''£' as str;
select q'"1: ' 2: '' 3: ''"' as str;
select q'|1: ' 2: '' 3: ''|' as str;
select q'!1: ' 2: '' 3: ''!' as str;
select q'+1: ' 2: '' 3: ''+' as str;
select q'/1: ' 2: '' 3: ''/' as str;
select q'$1: ' 2: '' 3: ''$' as str;
select q'β1: ' 2: '' 3: ''β' as str;
select q'Й1: ' 2: '' 3: ''Й' as str;
select q'Ω1: ' 2: '' 3: ''Ω' as str;
select q'あ1: ' 2: '' 3: ''あ' as str;
```

STR

-----  
1: ' 2: '' 3: '''

## PostgreSQL

```
select $$1: ' 2: '' 3: ''$$ as str;
select $id$1: ' 2: '' 3: ''$id$ as str;
select $name$1: ' 2: '' 3: ''$name$ as str;
select $😊$1: ' 2: '' 3: ''$😊$ as str;
```

str

-----  
1: ' 2: '' 3: '''  
(1 row)



# Generating SQL

```
OracleDB – SQLcl
set heading off
spool x.sql
select q'[select q'(That's cool!);]';
spool off
set heading on
@x.sql

SQL> select q'[select q'(That's cool!);]';

select q'(That's cool!);

SQL> spool off
SQL> set heading on
SQL> @x.sql

Q'(THAT'SCOO
-----
That's cool!
```

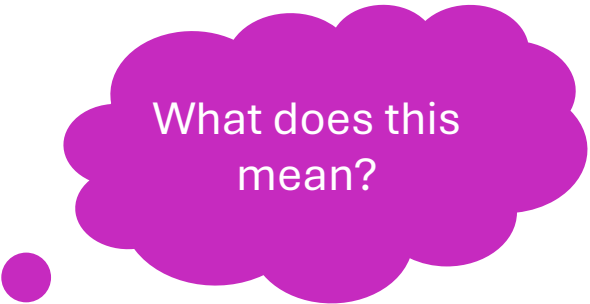
```
PostgreSQL – psql
\set ECHO queries
select $$select $i$That's cool!$i$;$$\gexec

select $$select $i$That's cool!$i$;$$
select $i$That's cool!$i$;
      ?column?
-----
      That's cool!
(1 row)
```

# Data Types

# Common OracleDB Data Types

OracleDB	PostgreSQL	Notes
blob	bytea	
<b>char</b>	<b>char</b>	Same semantics
clob	text	
<b>date</b>	<b>date</b>	Date without time
	timestamp(0)	Date with time, no sub seconds
<b>json</b>	<b>json</b>	non-binary JSON
	jsonb	binary JSON
number	decimal	
sdo_geometry	geometry	
<b>timestamp</b>	<b>timestamp</b>	Same semantics
<b>varchar</b>	<b>varchar</b>	" vs. NULL
varchar2	varchar	
xmltype	xml	



What does this mean?

## OracleDB

```
with
  t (name, c1) as (values
    ('null', null),
    ('empty', '' )
  )
select name,
       c1,
       c1 = '' as is_empty,
       case
         when c1 is null then
           'is null'
         else
           'is not null'
       end as result
from t;
```

NAME	C1	IS_EMPTY	RESULT
null			is null
empty			is null

## PostgreSQL

```
with
  t (name, c1) as (values
    ('null', null),
    ('empty', '' )
  )
select name,
       c1,
       c1 = '' as is_empty,
       case
         when c1 is null then
           'is null'
         else
           'is not null'
       end as result
from t;
```

name	c1	is_empty	result
null			is null
empty		t	is not null

(2 rows)

**SOL:2023**

# Functions

# Substr

```
OracleDB
with
  t(c1) as (values
    ('Hello World!')
  )
select substr(c1, 1, 5) one,
       substr(c1, 0, 5) zero,
       substr(c1, -6, 5) negative
from t;
```

ONE	ZERO	NEGAT
-----	-----	-----
Hello	Hello	World

```
PostgreSQL
with
  t(c1) as (values
    ('Hello World!')
  )
select substr(c1, 1, 5) one,
       substr(c1, 0, 5) zero,
       substr(c1, -6, 5) negative
from t;
```

one	zero	negative
-----	-----	-----
Hello	Hell	

(1 row)

SQL:2023  
Same semantic  
as substring

# Greatest, Least

```
OracleDB
select greatest(null, 1, 2) as highest,
       least(null, 1, 2) as lowest;
```

```
HIGHEST LOWEST
-----
```

*SQL:2023*

```
PostgreSQL
select greatest(null, 1, 2) as highest,
       least(null, 1, 2) as lowest;
```

```
highest | lowest
-----+-----
          2 |          1
(1 row)
```

# User-Defined Functions (UDFs)



# Function Definitions

```
● ● ● OracleDB
create or replace function f (
  p1 in integer
) return varchar2 is
begin
  return to_char(p1);
end;
/

create or replace function f (
  p1 in date
) return varchar2 is
begin
  return to_char(p1, 'DD.MM.YYYY');
end;
/
```

Function F compiled

Function F compiled

```
● ● ● PostgreSQL
create or replace function f (
  p1 in integer
) returns varchar language sql
return cast(p1 as varchar);

create or replace function f (
  p1 in date
) returns varchar language sql
return to_char(p1, 'DD.MM.YYYY');
```

CREATE FUNCTION  
CREATE FUNCTION

# Calling Previously Created UDFs

OracleDB

```
select f(date '2024-09-20');  
select f(42);
```

```
F(DATE'2024-09-20')
```

```
-----  
20.09.2024
```

```
Error starting at line : 14 in command -  
select f(42)  
Error at Command Line : 14 Column : 8  
Error report -  
SQL Error: ORA-06553: PLS-306: wrong number  
or types of arguments in call to 'F'
```

PostgreSQL

```
select f(date '2024-09-20');  
select f(42);
```

```
f
```

```
-----  
20.09.2024  
(1 row)
```

```
f
```

```
----  
42  
(1 row)
```

# Function Identity

## OracleDB

```
select object_name,  
       object_type  
  from user_objects  
 where object_name = 'F';  
  
drop function f;
```

OBJECT_NAME	OBJECT_TYPE
F	FUNCTION

Function F dropped.

Overloading for  
standalone  
functions, procedures  
using default values

## PostgreSQL

```
select proname,  
       pg_get_function_identity_arguments(oid)  
  from pg_proc  
 where proname = 'f';  
  
drop function f(date);  
drop function f(integer);
```

proname	pg_get_function_identity_arguments
f	p1 integer
f	p1 date

(2 rows)

DROP FUNCTION  
DROP FUNCTION

**SQL:2023**

# Select Statement

# Without Select-List

```
PostgreSQL
select from dept;
select;

--
(4 rows)

--
(1 row)
```

# Exists Subquery without Select-List

```
PostgreSQL
select *
  from emp a
 where exists ( -- bosses only
               select
                 from emp b
                where b.mgr = a.empno
               );
```

empno	ename	job	mgr	hiredate	sal	comm	deptno
7839	KING	PRESIDENT		1981-11-17	5000.00		10
7698	BLAKE	MANAGER	7839	1981-05-01	2850.00		30
7782	CLARK	MANAGER	7839	1981-06-09	2450.00		10
7566	JONES	MANAGER	7839	1981-04-02	2975.00		20
7902	FORD	ANALYST	7566	1981-12-03	3000.00		20
7788	SCOTT	ANALYST	7566	1987-04-19	3000.00		20

(6 rows)

# Without Select Keyword

```
PostgreSQL
table dept;
```

deptno	dname	loc
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON

(4 rows)

It's a SELECT statement!

works also:  
select \* from (table dept);

# Returning Clause



## PostgreSQL

```
begin;
  update emp
    set sal = sal * 1.2
    where sal + coalesce(comm, 0) < 2000
returning empno,
          ename,
          cast(sal / 1.2 as int) as old_sal,
          sal as new_sal;
rollback;
```

```
BEGIN
 empno | ename | old_sal | new_sal
-----+-----+-----+-----
 7499 | ALLEN |    1600 | 1920.00
 7900 | JAMES |     950 | 1140.00
 7844 | TURNER |    1500 | 1800.00
 7521 | WARD  |    1250 | 1500.00
 7934 | MILLER |    1300 | 1560.00
 7369 | SMITH |     800 |  960.00
 7876 | ADAMS |    1100 | 1320.00
```

(7 rows)

```
UPDATE 7
ROLLBACK
```

## OracleDB

```
set serveroutput on size unlimited
declare
  cursor c1 is
    select empno,
           ename,
           sal as old_sal,
           sal as new_sal
    from emp;
  type t_row_type is table of c1%rowtype;
  t_row t_row_type;
begin
  update emp
    set sal = sal * 1.2
    where sal + coalesce(comm, 0) < 2000
  return empno, ename, old sal, new sal
  bulk collect into t_row;
  dbms_output.put_line(sql%rowcount || ' rows updated. ');
  dbms_output.put_line(null);
  dbms_output.put_line('EMPNO ENAME OLD_SAL NEW_SAL');
  dbms_output.put_line('-----');
  for i in t_row.first..t_row.last
  loop
    dbms_output.put_line(rpad(t_row(i).empno, 6)
      || rpad(t_row(i).ename, 7)
      || lpad(t_row(i).old_sal, 7)
      || lpad(t_row(i).new_sal, 7));
  end loop;
  rollback;
end;
/
```

# Archiving (Without Partitioning)

## PostgreSQL

```
begin;
drop table if exists emp_archive;
create table emp_archive as
select * from emp where false;

with
  del as (
    delete from emp a
      where not exists (select from emp b
                        where b.mgr = a.empno)
    returning *
  )
insert into emp_archive select * from del;

rollback;
```

```
BEGIN
NOTICE:  table "emp_archive" does not exist, skipping
DROP TABLE
SELECT 0
INSERT 0 8
ROLLBACK
```

## OracleDB

```
drop table if exists emp_archive;
create table emp_archive as
select * from emp where false;

lock table emp, emp_archive
      in share row exclusive mode;
insert into emp_archive s
select * from emp a
  where not exists (select 42 from emp b
                    where b.mgr = a.empno);
delete from emp a
  where not exists (select 42 from emp b
                    where b.mgr = a.empno);

rollback;
```

```
Table EMP_ARCHIVE dropped.
Table EMP_ARCHIVE created.
Lock succeeded.
8 rows inserted.
8 rows deleted.
Rollback complete.
```

# Transactions

# Setup

```
OracleDB
drop table if exists t1;

create table t1 (
    id integer not null primary key
);

create or replace
procedure p1(in_id in integer) as
begin
    insert into t1 values (in_id);
end;
/
```

```
PostgreSQL
drop table if exists t1;

create table t1 (
    id integer not null primary key
);

create or replace
procedure p1(in_id in integer) as '
begin
    insert into t1 values (in_id);
end
' language plpgsql;
```

# Rollback in UDF

```
OracleDB  
set transaction read write; -- read committed  
call p1(1);  
call p1(1);  
commit;  
select * from t1;
```

Transaction READ succeeded.

Call completed.

ORA-00001: unique constraint ... violated...

Commit complete.

ID
1

```
PostgreSQL  
start transaction read write; -- read committed  
call p1(1);  
call p1(1);  
commit;  
select * from t1;
```

START TRANSACTION

CALL

ERROR: duplicate key value...

ROLLBACK

id
(0 rows)

**SOL:2023**

# Statement Level Rollback – Manually

```
PostgreSQL
start transaction read write; -- read committed
call p1(1);
savepoint before_insert;
call p1(1);
rollback to before_insert; -- cannot be done in exception block of p1
select * from t1;
```

```
START TRANSACTION
CALL
SAVEPOINT
ERROR: duplicate key value violates unique constraint "t1_pkey"
DETAIL: Key (id)=(1) already exists.
CONTEXT: SQL statement "insert into t1 values (in_id)"
PL/pgSQL function p1(integer) line 3 at SQL statement
ROLLBACK
 id
----
  1
(1 row)
```

# Core Messages

# SQL Is Standardized, but ...

- Same
  - syntax
  - data type name
  - isolation level
  - etc.

does not mean same behaviour across DBMSs.

- No compile error does not mean expected results.
- Does not spare you from **tests, tests, tests**.





# Thank you!

