ORACLE

Supercharge data processing with PL/SQL

Better SQL with PL/SQL

Chris Saxon, Oracle Developer Advocate
@ChrisRSaxon & @SQLDaily
https://blogs.oracle.com/sql
https://www.youtube.com/c/TheMagicofSQL

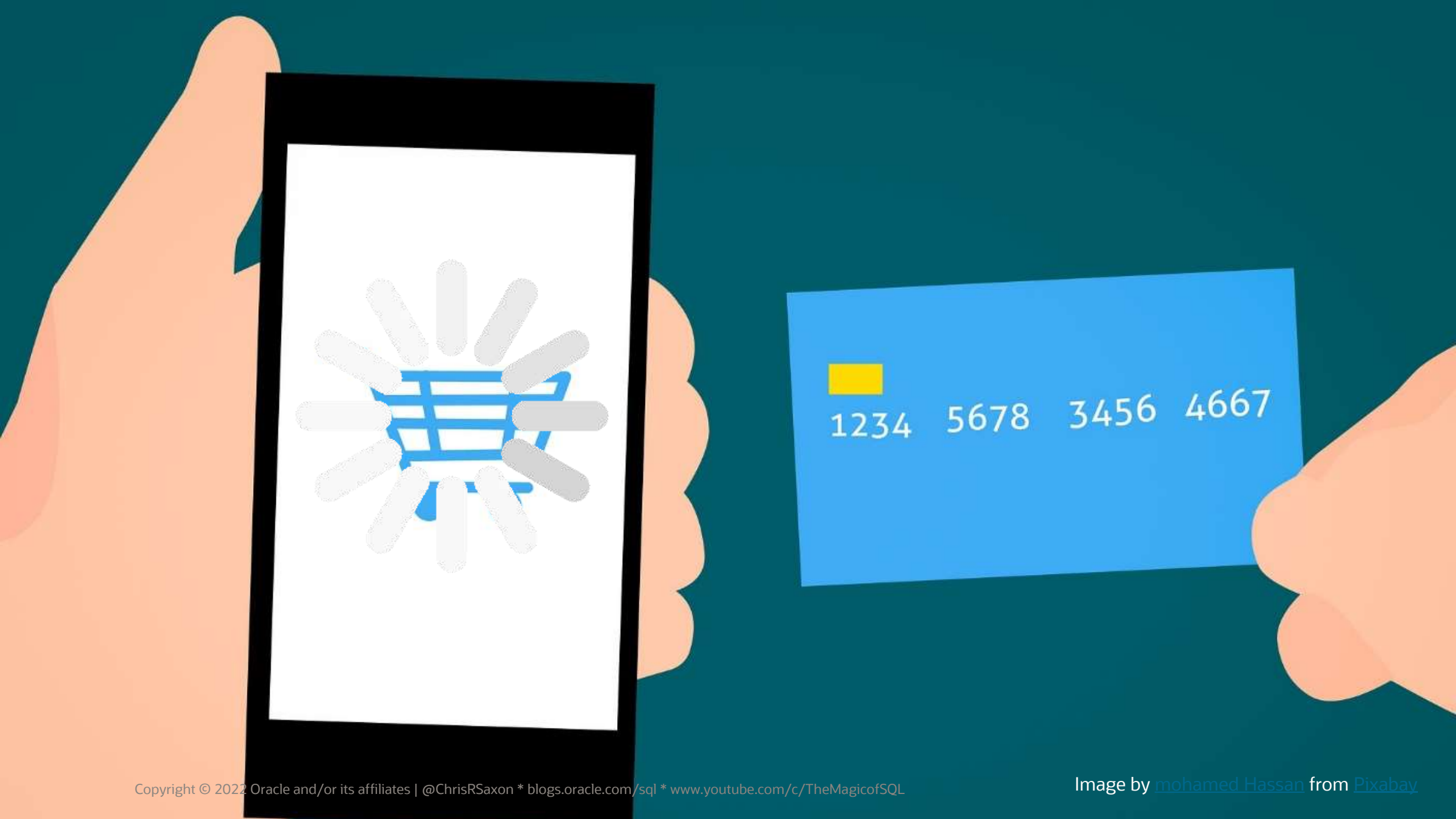
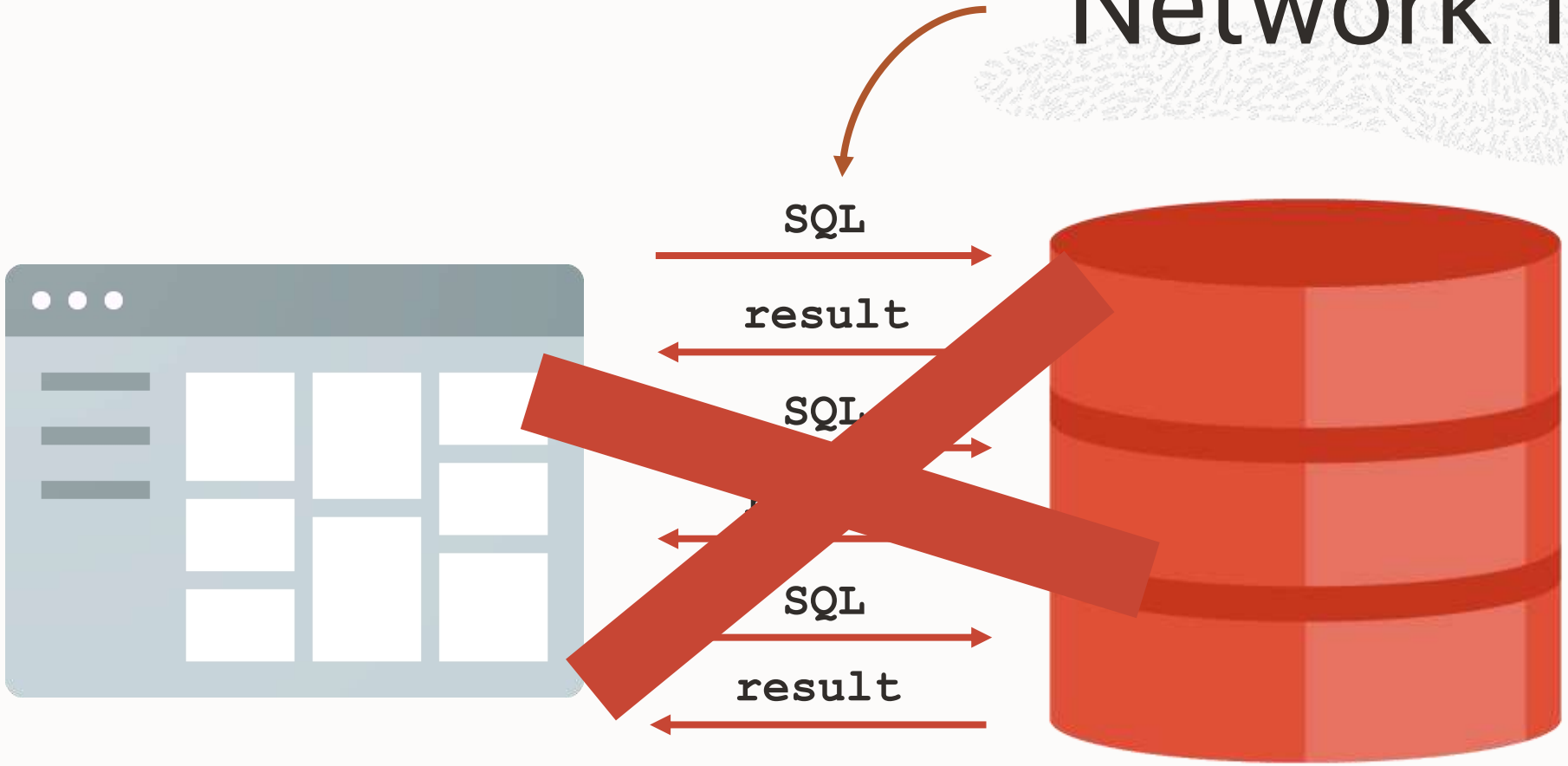


Image by mohamed Hassan from Pixabay

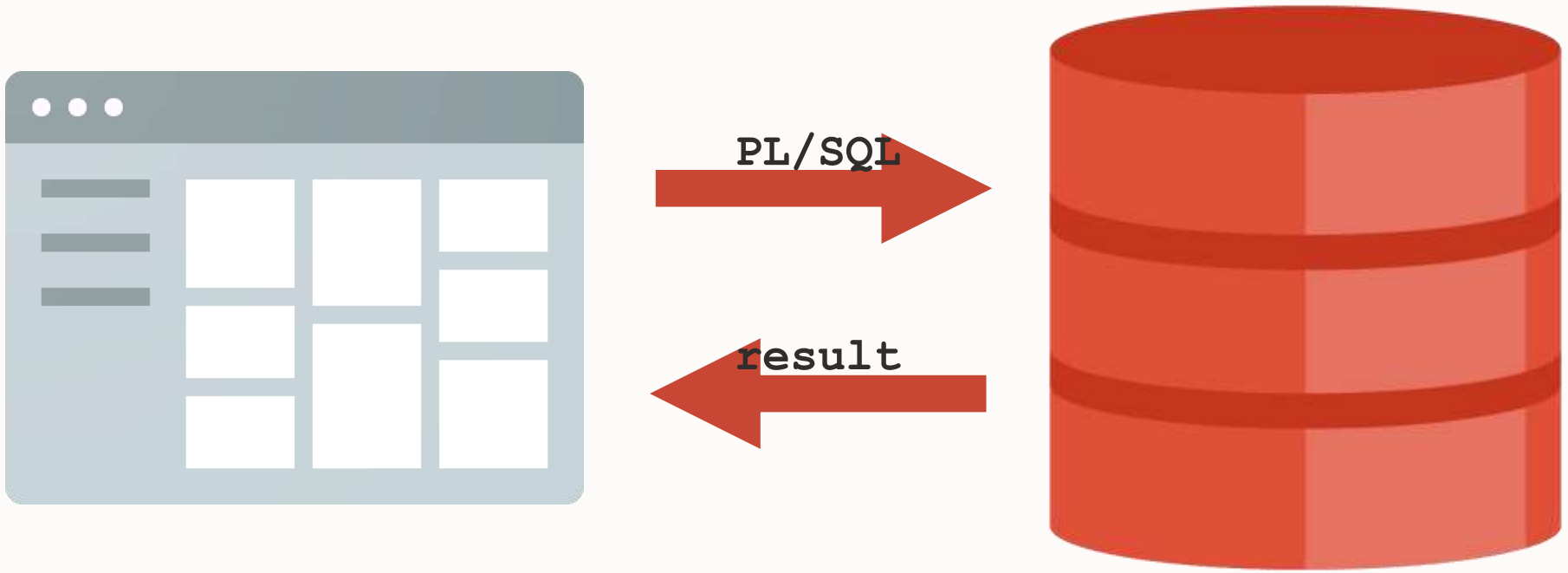
Ryan McGuire / Gratisography

```
for ( p = 0; p < products.length; p++ ) {

  PreparedStatement ins = conn.prepareStatement (
    "insert into order_items values ( ?, ?, … )";
  );

  ins.setInt ( 1, orderId );
  ins.setInt ( 2, products[p]… );
   …

  int row = ins.executeUpdate();
}
```

Network Time



Ryan McGuire / Gratisography



```
create procedure create_order (
  products product_array_type, …
) as
begin

  …

end;
```


```
for p in 1 .. products.count loop
```

Sparse arrays?

1 row/exec

```
end loop;
```


ORACLE

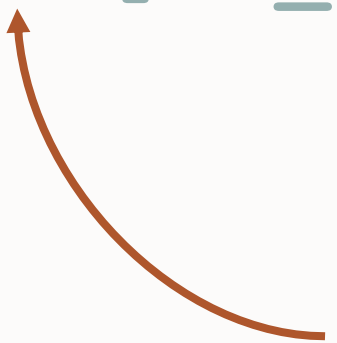
Supercharge data processing with PL/SQL

Better SQL with PL/SQL

Chris Saxon, Oracle Developer Advocate
@ChrisRSaxon & @SQLDaily
https://blogs.oracle.com/sql
https://www.youtube.com/c/TheMagicofSQL

Ryan McGuire / Gratisography

```
for n in 1 .. upper_bound loop

  if mod ( n, 3 * 5 ) = 0 then
    dbms_output.put_line ( n );
  end if;

end loop;
```



Process every value

Better Loops

Powerful iterators in PL/SQL in 21c

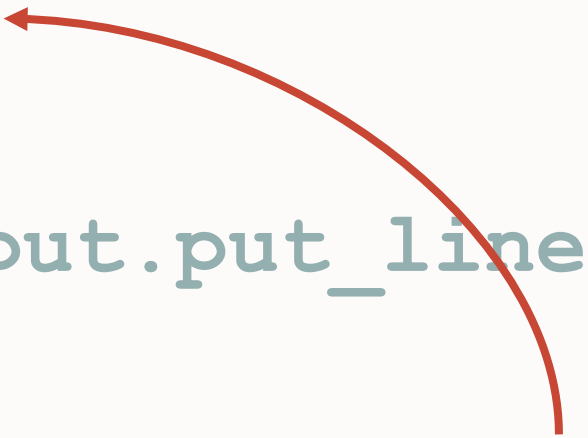
Increment

```
for n in 3 * 5 .. upper_bound
 by 3 * 5
loop
  dbms_output.put_line ( n );
end loop;
```

(21c)

Ryan McGuire / Gratisography

```
for n in 1 .. upper_bound
  by n * 2
loop
  dbms_output.put_line ( n );
end loop;
```



Needs fixed step value

(21c)

Enter loop when true

```
for n in 1 .. upper_bound
  when bitand ( n, n - 1 ) = 0
loop
   dbms_output.put_line ( n );
end loop;
```

Process every value

(21c)

do this

```
for powers in 1,
  repeat powers * 2
    while powers < 100
loop
  dbms_output.put_line ( powers );
end loop;
```

until false

(21c)

Change inside loop!

```
for powers mutable in 1 .. 100
loop
  dbms_output.put_line ( powers );
  powers := ( powers * 2 ) - 1;
end loop;
```

Still incremented!

(21c)

Ryan McGuire / Gratisography

```
for p in 1 .. products.count loop

   if debugging then
      log(...);
   end if;
   insert into ...

end loop;
```

ORA-01400

no data found

```
prod := products.first;
while prod is not null loop

  if debugging then
    log ( … );
  end if;
  insert into …
  prod := products.next ( prod );

end loop;
```

```
for p in indices of products loop

  if debugging then
    log ( … );
  end if;
  insert into …

end loop;
```

Array drives loop

1 row/exec

(21c)

```
if debugging then
  for p in indices of products loop
    log ( … );
  end loop;
end if;
```

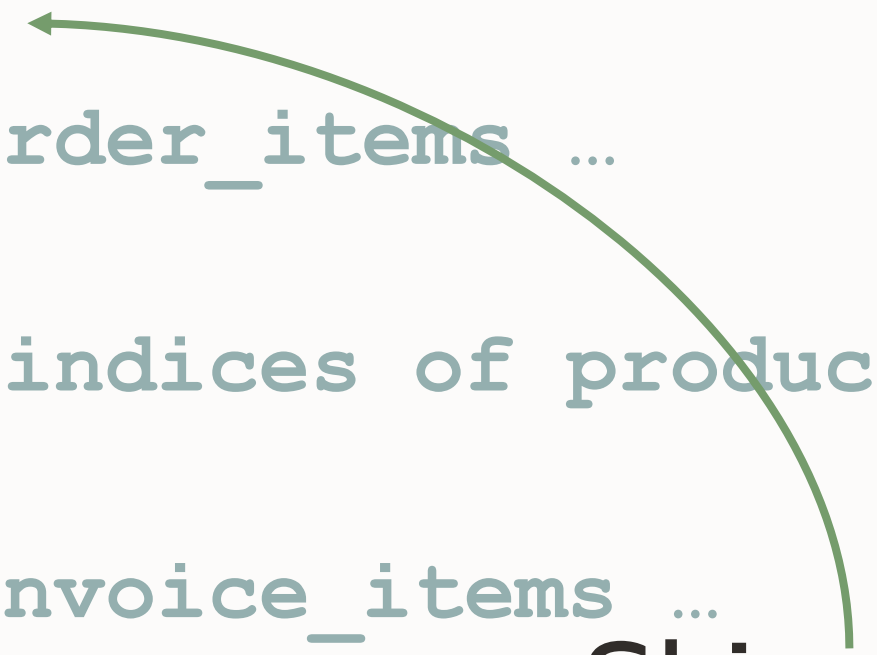
```
forall p in indices of products
  insert into …
```

(21c)

```
forall prod in indices of products
  insert into order_items …

…

forall prod in indices of products
  insert into invoice_items …
```


Ryan McGuire / Gratisography

```
forall prod in indices of products
save exceptions
   insert into order_items …

forall prod in indices of products
save exceptions
   insert into invoice_items …
```



Skip over errors

```
exception
   when failure_in_forall then
      for inx in 1 .. sql%bulk_exceptions.count
      loop
         errors := … ;
      end loop;
end;
```

Error array

Ryan McGuire / Gratisography

```
for n in ( 3 * 5 ) .. upper_bound
  by ( 3 * 5 )
loop
  fizzbuzz ( fizzbuzz.count ) := n;
end loop;
```

(21c)

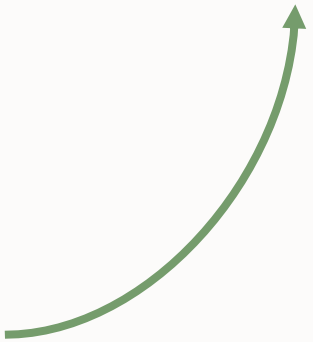
```
select level * 15 as n
bulk   collect
into   fizzbuzz
from   dual
connect by level < ( 100 / 15 )
```


Qualified Expressions

aka record and array constructors in 18c & 21c

```
declare
  fizzbuzz    dbms_sql.number_table;
begin

  fizzbuzz := dbms_sql.number_table (
    15, 30, 45, …
  );
```

Implicit constructor



(18c)

```
declare
  fizzbuzz    dbms_sql.number_table;
begin

  fizzbuzz := dbms_sql.number_table (
    for n in 15 .. upper_bound by 15
    sequence => n
  );
```

Index from 1

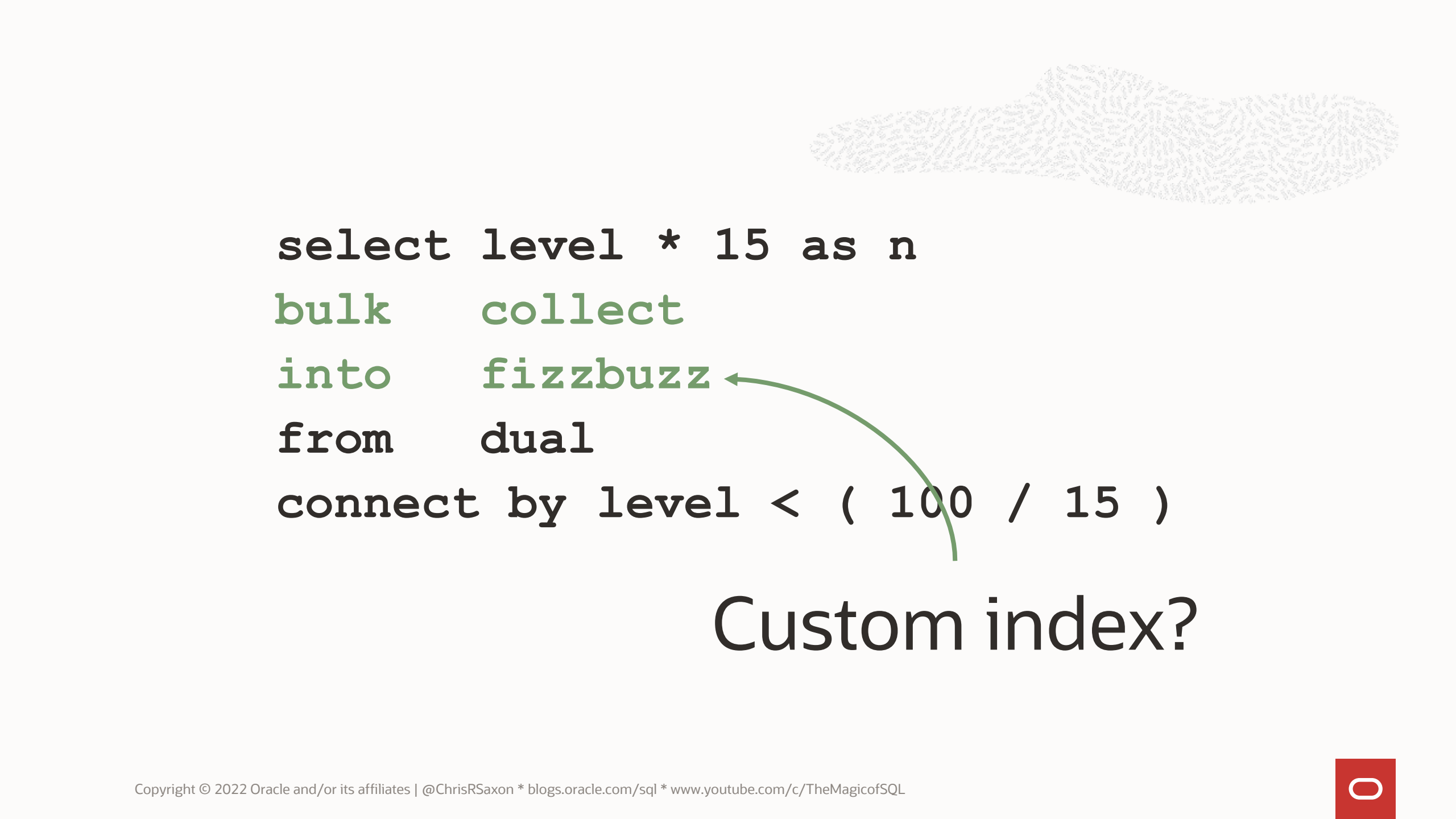
(21c)

```
declare
  fizzbuzz    dbms_sql.number_table;
begin

  fizzbuzz := dbms_sql.number_table (
    for n in 15 .. upper_bound by 15
    index n => n
  );
```

Custom index

(21c)

```
select level * 15 as n
bulk   collect
into   fizzbuzz
from   dual
connect by level < ( 100 / 15 )
```



```
declare

   fizzbuzz dbms_sql.number_table;

   cursor fibs_cur is
     select level * 15 as n
     from   dual connect by level < …;
```


```
begin
   fizzbuzz := dbms_sql.number_table (
      for f in fibs_cur
         index f.n
         => f.n
   );
```

Column as index

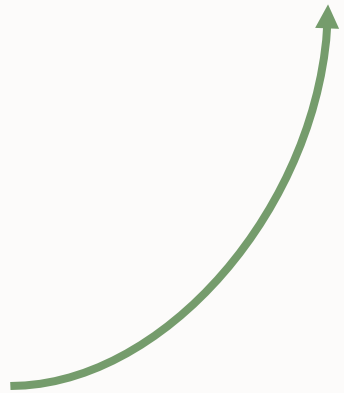
(21c)

Ryan McGuire / Gratisography

```
create table
  currencies (
  code varchar2(3)
    primary key,
  …
)
```

```
create table
  countries (
  code varchar2(2)
    primary key,
  …
)
```

```
curr_recs := curr_arr (
  for rws in curr_cur
    index rws.from_currency_code
    => rws
);
```



USD, GBP, EUR, ...

(21c)

```
eur_price :=
   usd_price *
   curr_recs ( 'USD' ).ex_rate;
```


Ryan McGuire / Gratisography

```
forall prod in indices of products
save exceptions
   insert into order_items …

forall prod in indices of products
save exceptions
   insert into invoice_items …
```

Get order error array

Get invoice error array

Combine!

```
order_errors := dbms_sql.number_table (
  for inx in 1 .. sql%bulk_exceptions.count
    => products (
      sql%bulk_exceptions (inx).error_index
    )
);
```

```
all_errors := dbms_sql.number_table (
  for error_prods in
    values of order_errors,
    values of invoice_errors
  index error_prods => error_prods
);
```

Ryan McGuire / Gratisography

DEMO

PL/SQL for better data access

PL/SQL

Reduce roundtrips

Loops

Stepped control
Skip values
Repeat <expr>
Array controls

Constructors

Init arrays with cursor loops

Merge arrays

https://cloud.oracle.com/free

https://www.oracle.com/xe/

Further reading

Better loops and qualified expressions in PL/SQL

https://blogs.oracle.com/plsql-and-ebr/better-loops-and-qualified-expressions-array-constructors-in-plsql

asktom.oracle.com

#AskTOMOfficeHours

Ryan McGuire / Gratisography