Oracle to PostgreSQL beyond the Syntax:

When DBMS Design Differences Matter

05/21/2026

Tino Engelbrecht, Joshua Steinmann

LUFTHANSA GROUP
BUSINESS SERVICES

Overview

1. Background
2. (Easy) Replacements
3. Fun With Numbers
4. Rethinking Updates
5. Key Takeaways

LUFTHANSA GROUP
BUSINESS SERVICES

Background

Who we are:

Software Developers, Ops, “own DBAs”
Department of Revenue Accounting
Customers are some of the largest airlines worldwide

Atlantic Joint Venture

Our experience:

In 2025: Migration project from Oracle to Postgres
Learned a lot from attendance at Postgres conference in Germany and Europe

(Easy) Replacements

Easy replacements

```
(SUBSTR(import_month, 1, 4) + 1 || '00') + NVL(late_coming_months, 0)
```

```
(SUBSTR(import_month, 1, 4) + 1 || '00') + COALESCE(late_coming_months, 0)
```

```
(SUBSTR(import_month, 1, 4)::BIGINT + 1 || '00') + COALESCE(late_coming_months, 0)
```

```
((SUBSTR(import_month, 1, 4)::BIGINT + 1) * 100) + COALESCE(late_coming_months, 0)
```

How to translate DECODE?

```
SELECT DECODE(n, null, 'NULL', 1, 'One', 'Other') FROM nums;
```

n
1
2
null

result
'One'
'Other'
'NULL'

How to translate DECODE?

```
SELECT CASE n
    WHEN null::INTEGER THEN 'NULL'
    WHEN 1 THEN 'One'
    ELSE 'Other' END
FROM nums;
```

n
1
2
null

result
'One'
'Other'
'Other'

How to translate DECODE?

```
SELECT CASE
    WHEN n IS NULL THEN 'NULL'
    WHEN n = 1 THEN 'One'
    ELSE 'Other' END
FROM nums;
```

n
1
2
null

result
'One'
'Other'
'NULL'

Join on DECODE

```sql
SELECT a.n, b.n FROM nums a JOIN nums b
    ON DECODE(a.n, b.n, 1, 0) = 1;
```

```sql
SELECT a.n, b.n FROM nums a JOIN nums b
    ON CASE
        WHEN a.n IS NULL AND b.n IS NULL THEN 1
        WHEN a.n = b.n THEN 1
        ELSE 0 END = 1;
```

```sql
SELECT a.n, b.n FROM nums a JOIN nums b
    ON a.n IS NOT DISTINCT FROM b.n;
```

n
1
2
null

a.n	b.n
1	1
2	2
null	null

IS NOT DISTINCT FROM

Expression	Result
1 IS NOT DISTINCT FROM 1	true
1 IS NOT DISTINCT FROM NULL	false
NULL IS NOT DISTINCT FROM NULL	true

```sql
SELECT * FROM tbl WHERE a IS NOT DISTINCT FROM NULL

-- rewritten to
SELECT * FROM tbl WHERE a IS NULL
```

```sql
SELECT * FROM tbl WHERE a IS NOT DISTINCT FROM 1

-- rewritten to
SELECT * FROM tbl WHERE a = 1
```

```sql
SELECT * FROM tbl_one LEFT JOIN tbl_two
  ON tbl_one.a IS NOT DISTINCT FROM tbl_two.a
```

Workaround with functions

n
1
2
null

a.n	b.n
1	1
2	2
null	null

```sql
SELECT a.n, b.n FROM nums a JOIN nums b
    ON COALESCE(a.n::text, '||NULL||') = COALESCE(b.n::text, '||NULL||');
```

```sql
CREATE INDEX idx_nums_coalesce ON nums (COALESCE(n::text, '||NULL||'));
```

Fun with numbers

Fun with numbers

Oracle

```
select 1/3;

=>  0.333333333333333333333333333333333333333 -- 39 digits
```

Postgres

```
select 1/3;

=>  0
```

Postgres

```
select 1.0/3;

=>  0.33333333333333333333 -- 20 digits
```

Postgres

```
select (1.0/3)::numeric(52,40);

=>  0.3333333333333333333300000000000000000000 -- 40 digits
```

Postgres

```
select 1::numeric(52,40)/3;

=>  0.3333333333333333333333333333333333333333 -- 40 digits
```

Exact Numeric Types

Oracle:

- Only one data type: NUMBER(Precision, Scale)
  - Stored as up to 20 base-100 digits → maximum precision of 39 or 40 digits depending on the position of the decimal point
  - INT, SMALLINT, INTEGER are type aliases for NUMBER(38,0)
- Always performs “normal” decimal division
- Always returns number with max precision as result of arithmetic operations

Postgres:

- Proper SMALLINT, INTEGER, BIGINT, NUMERIC (same as DECIMAL)
- Performs integer division on integer types
- Max precision is 1000
- Inexact arithmetic operations (division, sqrt, …) return at least 16 most significant digits
  - Scale of results is at least as big as the scale of both operands

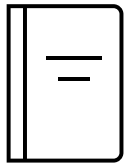
Rethinking Updates

LUFTHANSA GROUP
BUSINESS SERVICES

Oracle Updates

record	completed in
1	Jan ’26
2	Feb ’26
3	Mar ‘26
4	Mar ‘26
5	Mar ’26
6	Mar ‘26

record	completed in
1	Jan ’26
2	Feb ’26
3	Apr ’26
4	Apr ’26
5	Apr ’26
6	Mar ‘26



undo log

The Cost of Massive Updates

record	completed in	xmin	xmax
1	Jan ’26	100	0
2	Feb ’26	200	0
3	Mar ‘26	300	0
4	Mar ‘26	400	0
5	Mar ’26	500	0
6	Mar ‘26	600	0

record	completed in	xmin	xmax
1	Jan ’26	100	0
2	Feb ’26	200	0
3	Mar ‘26	300	700
4	Mar ‘26	400	700
5	Mar ’26	500	700
6	Mar ‘26	600	0
3	Apr ’26	700	0
4	Apr ’26	700	0
5	Apr ’26	700	0

The Cost of Massive Updates

record	completed in	xmin	xmax
1	Jan ’26	100	0
2	Feb ’26	200	0
3	Mar ‘26	300	0
4	Mar ‘26	400	0
5	Mar ’26	500	0
6	Mar ‘26	600	0

record	completed in	xmin	xmax
1	Jan ’26	100	0
2	Feb ’26	200	0
6	Mar ‘26	600	0
3	Apr ’26	700	0
4	Apr ’26	700	0
5	Apr ’26	700	0

The Cost of Massive Updates

record	completed in	xmin	xmax
1	Jan ’26	100	0
2	Feb ’26	200	0
3	<null>	300	0
4	<null>	400	0
5	<null>	500	0
6	<null>	600	0

record	completed in	xmin	xmax
1	Jan ’26	100	0
2	Feb ’26	200	0
3	<null>	300	0
4	<null>	400	0
5	<null>	500	0
6	<null>	600	700
6	Mar ’26	700	0

Key Takeaways for migrating

1. Make it run on Postgres ASAP
2. Find mistakes in translation, ensure correctness
3. Set goals for performance (benchmark old system?)
4. Optimize performance – Don’t shy away from hacks!
5. Find systematic issues
6. Go back to 4 until performance goals are met

There are bad practices in your system, you just got away with it.

Thank you for your attention!

LUFTHANSA GROUP
BUSINESS SERVICES