STUMBLING STONES WHEN MIGRATING FROM ORACLE

BY LAURENZ ALBE
ABOUT ME AND MY COMPANY

- Who is Laurenz Albe?
- Who is CYBERTEC?
LAURENZ ALBE
SENIOR DATABASE CONSULTANT

- contributions to PostgreSQL and related projects since 2006
- maintainer of the Oracle Foreign Data Wrapper
- PostgreSQL support, training, consulting and development
- working for CYBERTEC since 2017

MAIL  laurenz.albe@cybertec.at
PHONE +43 2622 930 22-7
WEB    www.cybertec-postgresql.com
About CYBERTEC

- Inhouse development
- International team of developers
- Specialized in data services
- Owner-managed since 2000
DATABASE SERVICES

DATA Science

- Artificial Intelligence
- Machine Learning
- Big Data
- Business Intelligence
- Data Mining
- etc.

POSTGRESQL Services

- 24/7 Support
- Training
- Consulting
- Performance Tuning
- Clustering
- etc.
CLIENT SECTORS

- ICT
- University
- Government
- Automotive
- Industry
- Trade
- Finance
- etc.
AGENDA

- Overview
- Understanding open source and PostgreSQL
- Migrate the schema (DDL)
- Data migration
- Migrating stored code
- Migrating SQL
- Migrating the application
- Migration tools
MIGRATION STEPS

- understand open source software and PostgreSQL
- migrate the schema (DDL)
- migrate the data
- migrate stored code (PL/SQL, Java)
- migrate SQL
- migrate the application
MIGRATION STEPS
(ACTUAL SIZE)

- understand open source software and PostgreSQL
- migrate the schema (DDL)
- migrate the data
- migrate stored code (PL/SQL, Java)
- migrate SQL
- migrate the application
UNDERSTANDING OPEN SOURCE AND POSTGRESQL
THE SHIFT TO OPEN SOURCE

- This is written by some enthusiasts in their spare time, right?
- Is this "enterprise ready"?
- Where can I get support?
- Why do I have to install and integrate so many different pieces of software (PostgreSQL, PostGIS, backup software, extensions, GUI clients, monitoring,...)?
- What if open source software is no longer maintained?
- It's for free, so I don't have to invest anything, right?
TRANSACTIONS, UNDO, MULTIVERSIONING

- both Oracle and PostgreSQL use multiversioning, so concurrency and locking are similar (but not equal!)
- big transactions are no problem in PostgreSQL (but long transactions are), so less need to “batch” large transactions
- no UNDO tablespace in PostgreSQL, no “snapshot too old”, immediate rollback

But:
- **UPDATE**-heavy workloads are problematic in PostgreSQL (may need “HOT update” and autovacuum tuning)
- table size will grow (all that visibility information)
- I no statement-level rollback
SCHEMAS, USERS AND SYNONYMS

Oracle has a reduced metadata model:
- a schema is always tied to a user with the same name
- ownership is determined by the schema
- only objects in your own schema can be referenced without schema

Synonyms are there largely to overcome these limitations
- can often be replaced by an appropriate search_path setting
- for other uses, a view is usually just as good
VIEWS AND DEPENDENCIES

- Oracle tables can be dropped/modified even if views depend on them
  - views become “invalid” and cause an error when used

- PostgreSQL is stricter about data integrity

- Schema upgrade procedures more difficult in PostgreSQL
  - but to make up for it, we have transactional DDL

- Materialized View support much more sophisticated in Oracle
  - replace `ON COMMIT REFRESH` with triggers in PostgreSQL
TABLESPACES

- tablespaces are important in Oracle
  - Oracle essentially implements its own file system

- PostgreSQL uses the host file system
  - tablespaces are rarely necessary

- Resist the urge to create tablespaces during migration!
MIGRATE THE SCHEMA (DDL)
DATA TYPE TRANSLATION

- PostgreSQL has way more data types, so the problem is often which one to choose
- `DATE` to `date` or `timestamp(0)`?
- `NUMBER` to integer, bigint, double precision or numeric?
  - Oracle allows foreign keys from `NUMBER(5)` to `NUMBER`
  - must take care to migrate them to the same data type
- `BLOB` to `bytea` or Large Objects?
  - easy, use `bytea`
GENERAL CONSIDERATIONS

■ Oracle makes it hard to export data in clear text
  ■ probably on purpose to make migration harder

■ this is often the least complicated step, but the one that causes the most down time

■ reducing down time is difficult
  ■ run migration of table data in parallel
  ■ use “change data capture” for replication and switch-over with little down time (only available with commercial tools)
DATA MIGRATION PROBLEMS

- corrupted strings in Oracle (more common than you think!)
  invalid byte sequence for encoding "UTF8": 0x80

- zero bytes in Oracle
  invalid byte sequence for encoding "UTF8": 0x00
  - can be filtered out during migration

- infinite numbers (~ and --)
  - can be mapped to Infinity in double precision, problematic otherwise

Most of these problems have to be solved in Oracle before migration.
MIGRATING STORED CODE
MIGRATING PL/SQL

- PL/pgSQL is a clone of PL/SQL, but sufficiently different (e.g., `RETURNS` vs. `RETURN`)
- some tools provide automated translation, but a lot of manual work may remain
- no `COMMIT/ROLLBACK` in PostgreSQL functions, limited support in procedures
  - often in “batched deletes”, → can be omitted
- no `PRAGMA AUTONOMOUS_TRANSACTION` in PostgreSQL
  - can sometimes be worked around with `dblink`
- no `BULK COLLECT` with arrays
  - process row by row

Shift transaction management to the application.
MIGRATING PL/SQL PACKAGES

- option to use closed source fork from EDB
- workaround: creating a schema with functions
  - no “package global variables” and types
- no large PL/SQL library in PostgreSQL
  - move code to the application
  - re-implement code in PL/Python or PL/PerlU
  - extension “orafce” provides some compatibility
MIGRATING PL/SQL TRIGGERS

- has to be split in two parts: trigger function and trigger
  - benefit: easier code reuse

- auto-increment triggers fetching from a sequence can be simplified to column `DEFAULT`

- no “logon triggers” in PostgreSQL
  - avoid or shift code to the application
MIGRATING SQL
WHERE DOES SQL OCCUR?

- application code
  - ORMs and other abstraction layers reduce this
- views
- PL/SQL code
- column `DEFAULT` clauses
- index definitions

Usually requires manual intervention; migration tools may help.
SQL: JOIN SYNTAX

```
SELECT b.col1, a.col2
FROM base_table b, attributes a
WHERE b.id=a.b_id(+);
```

has to be translated to

```
SELECT b.col1, a.col2
FROM base_table b
    LEFT JOIN attributes a ON b.id = a.b_id;
```

Always simple, but annoying!
SQL: EMPTY STRINGS

- Oracle treats empty strings as NULL
- as a consequence, 'hello' || NULL is not NULL in Oracle
- translate into concat('hello', NULL) or use "coalesce(strcol, '')"

This is a very frequent problem.
SQL: CURRENT DATE/TIME

- most Oracle code uses proprietary functions:
  - SYSDATE
  - SYSTIMESTAMP

- has to be translated:
  - the literal translation would be `clock_timestamp()`
  - sometimes `current_date` or `current_timestamp` is better

- easy with search and replace
Oracle code to fetch the next sequence value:
asequence.NEXTVAL

PostgreSQL code to fetch the next sequence value:
nextval('asequence')

both don't support the SQL standard way:
NEXT VALUE FOR asequence
MIGRATING THE APPLICATION
MIGRATING THE APPLICATION

- can be hard
  - hard coded dynamically composed SQL everywhere

- can be almost trivial
  - use an ORM that supports both Oracle and PostgreSQL

- requires **thorough testing**

- some differences (transaction handling, concurrency) may cause problems only during testing
MIGRATION TOOLS
some PostgreSQL forks (for example EDB) provide good compatibility
  but believe no claim of “drop-in replacement”
  carefully consider if you want to end up with closed source

consider using “orafce” for more compatibility
  open source, but still another dependency

it may be worth the effort to invest a little more and end up with free
standard PostgreSQL
ORA2PG

- the most widely used open source migration tool
  - time-tested and proven, but not 100% bug free
- generates a DDL script, exports and imports data
  - universally usable, but takes its time
- attempts to translate PL/SQL
  - simple search/replace, quality limited
**ORA_MIGRATOR**

- open source, uses the Oracle Foreign Data Wrapper
- directly migrates data into the target database
  - no export/import, therefore faster
- requires `oracle_fdw` in the target database
  - usually not an option with hosted databases
- no attempt to migrate PL/SQL
- provides a simple replication solution using triggers to reduce down time
CYBERTEC MIGRATOR

THE EASIEST WAY FROM ORACLE TO POSTGRESQL

SAVE HIGH LICENCE COSTS
VISUAL MONITORING
SIMPLE & FAST MIGRATION

COMPLETE SCHEMA CONVERSION
AUTOMATIC CODE REWRITE
CLOSE-TO ZERO DOWNTIME
CYBERTEC MIGRATOR

- commercial
- comfortable GUI driven migration
- fast, highly parallelized data migration
- high-quality PL/SQL conversion
- close-to zero downtime with change data capture under development

More information:
QUESTIONS?