ABOUT CYBERTEC

DATA Science
- Artificial Intelligence
- Machine learning
- BIG DATA
- Business Intelligence
- Data Mining
- Etc.

POSTGRESQL Services
- 24x7 Support
- Training
- Consulting
- Performance Tuning
- Clustering
- Etc.

Wiener Neustadt
AUSTRIA

Tallinn
ESTONIA

Zurich
SWITZERLAND

Montevideo
URUGUAY
Customer Sectors

- University
- Automotive
- Government
- Industry
- Electronic
- Finance
- Trade
- Etc...
Why migrate from Oracle to PostgreSQL?
REASONS FOR MIGRATION

- License terms can be a moving target
  - Virtualization is an issue
  - Entire infrastructure has to be licensed
  - License audits are a major financial risk

- Oracle tries to push clients to the cloud
  - Even issues with the SEC
  - Not everybody can put data into the cloud
MORE REASONS

- In the old days “costs” were the driving factor
- Technical reasons seem to become more frequent these days:
  - Simple administration
  - Flexible deployment
  - Small memory and storage footprint
- Ideological reasons become more frequent
“SPECIAL CUSTOMER RELATIONS”

- What if you car dealer inspected your garage?
- What if your car maker punished you for using 5th and 6th gear?
- License audits are an increasing issue
  - Unpredictable costs
We had to charge you every place because you could have parked anywhere.

REALLY!? € 5,000,000,- just to park my car?!
Moving to PostgreSQL - technical aspects
IMPORTANT ASPECTS TO CONSIDER

- SIMPLE STUFF:
  - Table definitions
  - Data
  - Indexes
  - Constraints and keys

- INTERESTING PARTS:
  - Stored procedures
  - Oracle Forms
  - Oracle APEX
MIGRATING TABLES

- Usually pretty simple
- What to keep in mind:
  - PostgreSQL knows more data types
  - “number” has many counterparts in PostgreSQL
- Constraints and foreign keys
  - Usually quite simple (SQL Standard)
MIGRATING DATA

- Data is usually simple
- The devil is in the details
- NULL Handling is special
  - PostgreSQL is ANSI SQL compliant
  - Oracle is not
  - Be aware of subtle bugs
  - " is not the same as undefined
MIGRATING DATA CHARACTER SETS

- Oracle is quite "relaxed" when dealing with broken data
- Oracle can store broken characters
  - PostgreSQL is extremely strict (= correct)
  - How shall one handle broken characters?
- Usually stuff has to be fixed on the Oracle side first
DUAL

- PostgreSQL does not need a FROM-clause
- DUAL can be faked easily

postgres=# CREATE TABLE dual AS SELECT 1 AS x;
SELECT 1
postgres=# SELECT * FROM dual;
x
---
1
(1 row)
ROWNUM:
Identifying rows in Oracle

- PostgreSQL does not have the quite same thing as Oracle
  - CTID is slightly different

- Migrating ROWNUM:
  
  `ROW_NUMBER() OVER (ORDER BY ...)`
SEQUENCES

- Fairly simple (unless buried deep inside your application)
  - Only small changes are needed

```python
sequence_name.nextval
vs.
nextval('sequence_name')
```
SUBQUERIES

- PostgreSQL needs an "AS"-clause

```sql
SELECT * FROM (SELECT * FROM table_a)
vs.
SELECT * FROM (SELECT * FROM table_a) AS foo
```
OUTER JOINS: FIXING “+”

```sql
SELECT a.field1, b.field2
FROM a, b
WHERE a.item_id = b.item_id(+)

vs.

SELECT a.field1, b.field2
FROM a
LEFT OUTER JOIN b
ON a.item_id = b.item_id;
```
DECODE: ORACLE SPECIFIC SYNTAX

- Make use of CASE / WHEN
  - or maybe use a small stored procedure to handle more complex scenarios

DECODE (x, NULL, 'null', 'else')

vs.

CASE x WHEN NULL THEN 'null' ELSE 'else' END
CONNECT BY vs. WITH RECURSIVE

- SQL is able to handle recursions
- CONNECT BY is Oracle specific
  - CONNECT BY has to be rewritten
- WITH RECURSIVE is ANSI standard
TOOLS AND UTILITIES
FUNCTIONS AND PROCEDURES

- Many functions are not available in PostgreSQL
- orafce can help to provide many of those
**EXAMPLES**

\[
\begin{align*}
\text{add_months(date '2005-05-31',1)} & \rightarrow 2005-06-30 \\
\text{trunc(date '2005-07-12', 'iw')} & \rightarrow 2005-07-11
\end{align*}
\]

vs.

\[
\begin{align*}
\text{oracle.sysdate()} & \rightarrow 2015-12-09 17:47:56 \\
\text{oracle.to_char(to_date('14-Jan08 11:44:49+05:30'))} & \rightarrow 14-Jan08 11:44:49
\end{align*}
\]

- "oracle" can be added to the search_path
VARIOUS COMMENTS

- NTT says: Using the means mentioned so far 80% of all code can be migrated directly

- CYBERTEC feels: Depends on the types of applications of course
A CLASSICAL TOOL: ORA2PG

- A Perl program capable of exporting Oracle databases

**PROS:**
- Has been around forever
- Widely adopted
- Many features

**CONS:**
- Quite buggy
- Tries to do things which are impossible
ora2pg --project_base /app/migration/
   --init_project test_project

Creating project test_project.
   /app/migration/test_project/
       schema/
           dblink/
           directories/
           functions/
           ...
           synonyms/
           tables/
           tablespaces/
           triggers/
           types/
           views/
       sources/
           functions/
           mviews/
           ...
       data/
       config/
       reports/
ORA2PG: EXECUTING THE IMPORT

Generating generic configuration file
Creating script export_schema.sh to automate all exports.
Creating script import_all.sh to automate all imports.
ORA2PG: PROCEDURES

- ora2pg tries to convert them
  - always fails
  - which is expected anyway
  - manual work needed
ORACLE_FDW & ORA_MIGRATOR

- Built as an alternative to ora2pg
  - we had to streamline the migration processes more
  - ora2pg is hard to automate
  - too many issues on import
ORAS_MIGRATOR: BASIC COMPONENTS

- **oracle_fdw** is the Oracle Foreign Data Wrapper
  - it is really good at reading data from Oracle
  - it does support IMPORT FOREIGN SCHEMA
  - excellent data type mapping

- **ora_migrator**: builds on top of oracle_fdw
ORA_MIGRATOR: WHAT IT DOES IT WORKS

- Connects to Oracle
- Clones the Oracle system catalog and creates copies on the PostgreSQL side
- You can now adjust things to your need
  - Fix types
  - Rewrite procedures
  - Exclude things, etc.
- “Compiles” those table definitions to PostgreSQL objects
- Imports the data
- Creates indexes, etc.
CREATE EXTENSION ora_migrator;

SELECT oracle_migrate(server => 'oracle', only_schemas => '{LAURENZ,SOCIAL}');

NOTICE: Creating staging schemas "ora_stage" and "pgsql_stage" ...
NOTICE: Creating Oracle metadata views in schema "ora_stage" ...
NOTICE: Copying definitions to PostgreSQL staging schema "pgsql_stage" ...
NOTICE: Creating schemas ...
NOTICE: Creating sequences ...
NOTICE: Creating foreign tables ...
NOTICE: Migrating table laurenz.log ...
NOTICE: Migrating table laurenz.ft_speed_sa ...
NOTICE: Migrating table laurenz.badstring ...
WARNING: Error loading table data for laurenz.badstring
DETAIL: invalid byte sequence for encoding "UTF8": 0x80:
NOTICE: Migrating table laurenz.datetest ...
NOTICE: Migrating table laurenz.department ...
NOTICE: Migrating table laurenz.hasnul ...
WARNING: Error loading table data for laurenz.hasnul
DETAIL: invalid byte sequence for encoding "UTF8": 0x00:
NOTICE: Migrating `table` social.blog ... 
NOTICE: Migrating `table` laurenz.employee ... 
NOTICE: Migrating `table` laurenz.identity ... 
NOTICE: Migrating `table` laurenz.req_lot ... 
NOTICE: Migrating `table` social.email ... 
NOTICE: Migrating `table` laurenz.numbers ... 
NOTICE: Creating **UNIQUE and PRIMARY KEY** constraints ... 
WARNING: Error creating `primary key or unique constraint` on table laurenz.badstring 
DETAIL: relation "laurenz.badstring" does not exist: 
WARNING: Error creating `primary key or unique constraint` on table laurenz.hasnul
ORA_MIGRATOR: HOW IT WORKS

DETAIL: relation "laurenz.hasnul" does not exist:
NOTICE: Creating FOREIGN KEY constraints ...
NOTICE: Creating CHECK constraints ...
NOTICE: Creating indexes ...
NOTICE: Setting column default values ...
NOTICE: Dropping staging schemas ...
NOTICE: Migration completed with 4 errors.
oracle_migrate
---------------------
4 (1 row)
ORA_MIGRATOR: WHAT IT DOES NOT

- we do not try to migrate procedures
- we simply extract the code
ORA_MIGRATOR: WHAT IT DOES NOT

- Oracle accepts “dirty” data
  - PostgreSQL does not
  - Oracle is fine with broken characters

- Data has to be verified and fixed on the Oracle side first
COMMON SHOWSTOPPERS

- Oracle accepts “dirty” data
  - PostgreSQL does not
  - Oracle is fine with broken characters

- Data has to be verified and fixed on the Oracle side first
ORA_MIGRATOR can perform those checks

- We convert data "back and forth" to see if it stays identical
- Only way to detect broken characters reliably
ORA_MIGRATOR: PROCEDURES

Procedures have to be rewritten manually

- No way to do automatic conversion
CEO
Hans-Jürgen Schönig

MAIL hs@cybertec.at
PHONE +43 2622 930 22-2
TWITTER @postgresql_007

WEB www.cybertec-postgresql.com
TWITTER @PostgresSupport

ANY QUESTIONS?