Oracle to PostgreSQL replication and migration

Tomasz Rybak

TeraData
tomasz.rybak@teradata.com

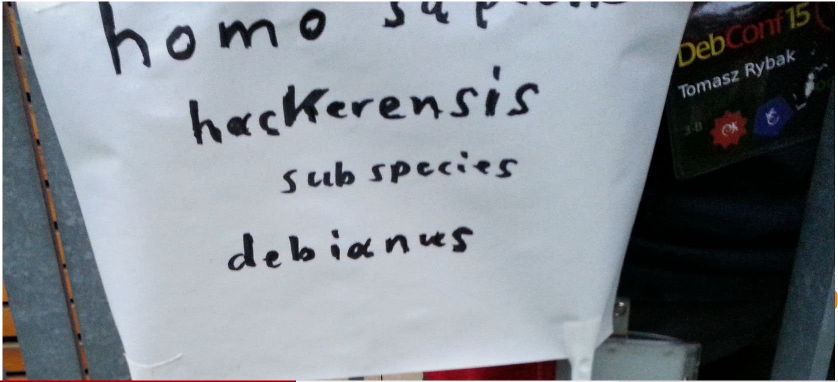
PgConfEU, 2015-10-29

Outline

1. Our system
2. ORQ
3. Problems and solutions
   - Performance
4. Our changes
5. Process
   - Lesson learned
6. Summary

Who am I

- Professionally
  - DBA at Teradata in Muenchen
- FLOSS
  - Debian Maintainer (with open NM process)
  - Python
  - GPGPU (PyOpenCL, PyCUDA)



Disclaimer

All opinions presented here are my own, and my employer is not responsible for them.

Big picture

- Many databases
  - PostgreSQL
  - Oracle
- Databases created over many years
- Oracle
  - First database; initial schema
  - Quite old (10g), currently unsupported version
- Decision
  - Unification on PostgreSQL
  - Unification of schema

Oracle

- Over 700 tables

  widest table over 600 columns

  largest table 5.5G rows

  database size almost 1.5TB
- Our first database
- Old hardware
  - Magnetic hard drives
  - High IO
  - Limited disk space

Production

- Live systems
- Clients' jobs constantly running
  - Cannot stop and migrate data offline
- Cannot add much load
  - Quite high IO
- We are already using Londiste on PostgreSQL side

Solution

- ORQ from 2ndQuadrant
  - Similar to londiste
  - . . . but not identical
- Mostly Python
  - cx_Oracle
  - psycopg2
- Quite a bit of PL/SQL

Architecture

- Code on Oracle
  - Triggers
  - Table with events
  - Metadata tables: subscribers, etc.
- PostgreSQL
  - Extension
  - Just information about tables
- Daemons

Daemons

orqd ticker, allowing for determining when to apply transaction changes

orqrep replicator; only one (unlike in londiste)

orqrep-admin management tool

Oracle

- orq user and schema
- triggers
- configuration tables
  - ticks
  - queues
  - daemon status
  - trigger names
  - and not the list of replicated tables

PostgreSQL

- user
- extension
- only one table: list of replicated tables

Queues

- For storing events
  - Insert, Update, Delete
  - Also adding and removing table
- Three (not one) tables for queue
  - Tables rolling
  - Helps with cleaning old events
- Need to prevent replication from rolling tables with long transactions
- Need to determine from which queues to read events

Multi-table queues

- Helps with changes to event tables on live system
- More difficult debugging
- More complicated event queries
- orqd
    - Determines when to rotate based on ticks seen by orqrep
    - Deletes old ticks
- Lack of rotation suggests some stuck transaction

Encoding

- Everything should be in UTF8
- Two ways of inserting data on PostgreSQL side
  - initial copy
  - event applying
- Problems for PostgreSQL: Not always valid UTF8 characters
- Difficult debugging
  - cx_Oracle uses Oracle libraries
- Force Oracle to give use UTF8
- Both as parameter to cx_Oracle and as environment variable
- Took lot of checking and searching
- Still not always perfect
  - Additional code to check and fix text fields

Object names

- Oracle size limit: 30 characters
- Triggers: table name and suffix
- Need to manage triggers
- Additional table with mapping
- table name ⇒ shortened trigger name

Triggers

- Initially one trigger for all operations
- Generated per table, using list of columns
- For many tables it was bigger than 32k
    - We had to move to CLOB from varchar
- For some of tables (500 columns) generated 560kB of code
- Much larger than limit
    - Documentation: 32kB
    - Tests: 90kB

Triggers - cont.

- Complexity matters more than size
- We had to split trigger to 3, to decrease number of statements
  - It allowed for simplifying of trigger code
- Static list of columns
  - Causes problems with schema migration

Events

- Need to group transaction statements
- Need to order transactions
- ora_rowscn
  - Virtual column
  - Similar to txid
  - . . . but not really
- Each event had ora_rowscn
  - Order by it
- Ticks (from orqd) determine which SCN to use

System Change Number

- ora_rowscn is virtual column
    - No index!
    - Full Table Scan for all events
- Problematic when more than few hundred thousands of events
- Decrease time of rolling event tables
- Did not help

Index

- Need to use index
    - Cannot on ora_rowscn
- Create ordinary column and fill in
    - SCN given at the end of transaction
    - Let's use dbms_fallback.get_current_change_number!
- Big mistake: It's changing during transaction!
- Caused problems more than once

Index - cont.

- Keep the column but update it on batch
  - 10g: cannot use ora_rowscn in non-select
  - Loop over transaction
- Not fast enough for our needs
- Tried to use during tick generation but it didn't work

Back to blackboard

- SCN issued at the end of transaction
- Before that column is empty
- Problem
- Two problems?
  - Grouping events into transactions
  - Transactions ordering
- Solution: separate grouping and ordering
  - Events use local_transaction_id
  - Not ordered, using implementation details
  - File, slot, transaction inside slot

Changes

- Additional table for transactions
- Additional trigger, per statement
- Check whether transaction exists, to avoid bloat
- One row per transaction — we can use SCN
- Problem with joining tables and rotating
- Use CTE/WITH
- local_transaction_id exists only after DML
- Problem when adding or removing table

Initial copy

- Not very fast: less than 2M rows per minute
    - Similar speed on sqlplus
- Tried different approaches
- Finally using Java
    - Copy done in separate process
- Java was fast for narrow tables, but slower for wide ones
    - JDBC vs. cx_Oracle
    - Different GC

Schema differences

- Oracle uses schema per user
- Tables are prefixed when read by different user
- Not in PostgreSQL
- So we use “different” table names for Oracle and PostgreSQL

Logging

- Error in trigger
- Log it, and otherwise ignore
- Do not rollback parent transaction!
- EXCEPTION WHEN OTHERS THEN . . .
- Just like “try: except:” in Python
- MERGE (UPSERT) to log number of errors
- Ignore errors during logging

Trigger compilation

- By default trigger gets compiled on first usage
  - Slowing first transaction
  - Catching problems on production
- We forced compilation of trigger during creation
  - Logging any problems
  - Also removing triggers and table from replication in such a case

Performance

- Dropping indices before initial copy
- Recreating them later
    - Similarly for FKs, constraints, etc.
- Already done in Londiste
- Problem with restoring after errors

Security

- Separate user
  - Although with some special privileges
- Separate tablespace
- Logging errors

Schema change

- Cleaning schema on Oracle
- Unifying as much as possible with PostgreSQL

Live systems

- Schema migrations on active databases
- Need to be applied on replica
- Regeneration of triggers

First attempt

- Missing tick on day of switch
  - Removed during table roll
  - Caused by dbms_flashback.get_current_system_change
- orqrep did not know what to do
- I was at DebConf
- To many unprocessed events when discovered
- Led us to discover difference between PostgreSQL and Oracle

Interlude

- Solve SCN problem
  - Many tried solutions
- Filled up tablespace
- Slow MERGE
  - Lock contention

Second attempt

- Intensive work on ORQ
  - Java for copy
  - Usage of transaction id
- Direct connection between Postgres and Oracle
  - To avoid problems during network changes
- Quite fast
- Failed
  - Broken storage
  - Logging (and ignoring) errors save us here

Truisms

- Documentation helps
  - When it exists
- Google helps
  - When it points to right answers
  - Oracle versions
  - Oracle vs. MySQL

Tailored solutions

- Existing system
  - We changed it for our needs
- We used quick rolls, to avoid keeping many events
- Limited look-back — only one table back

Quirks

- SELECT AS OF SCN
  - Interesting way of looking into near past
- MERGE
  - UPSERT
  - Did nor work correctly
  - Maybe I made some errors
- clob vs. varchar
  - Artificial limitation
  - Different code to deal with it

Hints

- Let the flamewar begin!
- Did not work for us
  - Neither for copy
  - Nor for event queries
- Parallel copy did not work
  - All tables of degree 1
- Suggestion of index usage ignored
- prefetch (array size in cx_Oracle) did not have much effect

Many attempts

- Practice makes perfect
- Monitoring of databases
  - . . . and replication
- Transaction size statistics
  - Many changed only one row
  - Some few dozen thousand changes
  - One changed 1.1M of rows

Result

We did it! (On Sunday)

- Many thanks:
    - Teradata team
    - 2ndQuadrant team

QA

- Thank you for your attention
- Questions?
- Contacts to me
    - Blog: http://tomaszrybak.wordpress.com/
    - Email:
        - tomasz.rybak@teradata.com
        - tomasz.rybak@post.pl
    - Skype: rybakthomas
    - GTalk: tomasz.rybak@gmail.com