Dumping the Mainframe: Migration Study from DB2 UDB to PostgreSQL

Michael Banck <michael.banck@credativ.de>

PGConf.EU 2015
Overview

- Introduction
- Differences Between DB2 and Postgres
- Schema Migration
- Data Migration
Introduction

- DB2 UDB is the z/OS mainframe edition of IBM’s DB2 database
- DB2 UDB central database and application server (“the Host”) in German regional government ministry
- Used by programs written in (mostly) Software AG Natural and Java (some PL/1)
- Natural (and PL/1) programs are directly executed on the mainframe, no network round-trip
- Business-critical, handles considerable payouts of EU subsidies
- Crunch-Time in spring when citizens apply for subsidies
Prior Postgres Usage

- Postgres introduced around 10 years ago due to geospatial requirements (PostGIS)
- Started using Postgres for smaller, non-critical projects around 5 years ago
- Modernized the software stack merging geospatial and business data around 3 years ago
- In-house code development of Java web applications (Tomcat/Hibernate/Wicket)
- Business-logic in the applications, almost no (DB-level) foreign keys, no stored procedures
- Some business data retrieved from DB2, either via a second JDBC connection, or via batch migrations
- Now migrating all Natural/DB2 programs to Java/Postgres
Application Migration Strategy

- **Java Applications**
  - Development environment switched to Postgres and errors fixed
  - Not a lot of problems if Hibernate is used
  - Potentially get migrated to modernized framework

- **PL/1 Applications**
  - Get rewritten in Java

- **Natural Application**
  - Automatic migration/transcription into (un-Java, but correct) Java on DB2
  - Migration from DB2 to Postgres in a second step (no application changes planned)
Current Setup

- **Postgres**
  - PostgreSQL-9.2/PostGIS-2.0 (upgrade to 9.4/2.1 planned for mid-December)
  - SLES11, 64 cores, 512 GB memory, SAN storage
  - HA 2-node setup using Pacemaker, two streaming standbys (one disaster recovery standby)
  - Roughly 550 GB data, 22 schemas, 440 tables, 180 views in PROD instance
  - Almost no stored procedures (around 10)

- **DB2**
  - DB2 UDB Version 10
  - Roughly 120 GB data, 70 schemata, 2000 tables, 200 views, 50 trigger in TEST instance
  - Roughly 550 GB data in PROD instance
  - Almost no stored procedures (around 20)
Current Status

- Proof-of-concept schema and data migration of TEST instance
- Migration of PROD instance originally planned for end of 2015
- Natural migration to Java delayed
  - So far no tests with migrated Java on Postgres
  - Planned for November 2015
- In-house Java developers difficult to reach, have not tested their applications so far
- Several Java projects maintained by external developers have been (mostly) successfully tested on local Postgres deployments
- First production migration of a java program and its schema planned for November 2015
SQL Differences

- Migration Guide in PostgreSQL wiki
  - Age and Author unknown

- Noticed SQL Differences
  - CURRENT_TIMESTAMP etc. (but CURRENT_TIMESTAMP is supported by DB2 as well)
  - Casts via scalar functions like INT(foo.id)
  - CURRENT_DATE + 21 DAYS
  - ‘2100-12-31 24.00.00.000000’ timestamp in data - year 2100/2101
Behavior Differences

- Noticed SQL Behavior Differences
  - Column names in DB2 are written in upper case by default, can lead to issues if they are quoted
  - ORDER BY returns differently sorted sets due to legacy(?) collation in DB2
  - GROUP BY implies sorting in DB2 so corresponding ORDER BY have been left out - Postgres does not guarantee sorted output

- Noticed JDBC Behavior Differences
  - SELECT COUNT(*) returns an int on DB2, but long in Postgres

- Other Behavior Differences
  - Statements error out in transactions in Postgres after first error
Evaluated Programs and Tools

- SQLWorkbench/J ([http://www.sql-workbench.net](http://www.sql-workbench.net))
  - Java-based, DB-agnostic workbench GUI
  - Heavily-used in-house already, installed on workstations
  - Allows for headless script/batch operation via various internal programs
  - Almost Open Source (Apache 2.0 with restriction of right-to-use to US/UK/China/Russia/Canada government agencies)
  - Provides a mostly usable console akin to psql

- pgloader ([http://pgloader.io](http://pgloader.io))
  - Lisp-based Postgres bulk loading and migration tool
  - Open Source (PostgreSQL license)
  - Written and maintained by Dimitri Fontaine (PostgreSQL major contributor)
General Approach

- Dump schema objects into an XML representation
- Transform XML into Postgres DDL via XSLT
- Provide compatibility environment for functions called in views and triggers
- Post-process SQL DDL to remove/work-around remaining issues
- Handle trigger separately
- Ignore functions/stored procedures (out-of-scope)
Schema-Migration, Encountered Problems

- wbreport2pg.xslt stylesheet sorting column numbers alphabetically, leading to problems when bulk-loading data
  - Patched to sort columns numerically
- Column name with keywords like USER
  - Use (new) XSLT parameter (quoteAllNames=true)
- Sequences having same name as corresponding table, leading to namespace violations
  - Need to be renamed
- DB2 View definitions are stored including CREATE VIEW in XML, leading to duplicated CREATE VIEW in generated DDL
  - Removed in post-processing
- DB2 got upgraded to version 10, but not using new system catalogs - View definition export errors out
  - Put old XML in ~/SQLWorkbench/ViewSourceStatements.xml

Michael Banck <michael.banck@credativ.de>
Schema-Migration, Details

- SQLWorkbench/J WbSchemaExport program
  - Writes an XML file of the schema
  - `WbSchemReport -schemas=${SCHEMA} -file=${XMLFILE} -includeSequences=true -includeTriggers=true -writeFullSource=true -types=table,view,sequence,constraint,trigger`

- SQLWorkbench/J WbXslT program
  - Transforms XML to Postgres DDL via `wbreport2pg.xslt` script
  - `WbXslT inputfile=${XMLFILE} -xsltoutput=${DDLFILE} -xsltParameters="quoteAllNames=true -xsltParameters="makeLowerCase=true" -xsltParameters="commitAfterEachTable=false" -stylesheet=wbreport2pg.xslt`
Convert charset of generated DDL from iso-8859-15 to UTF-8

- Remove WITH [LOCAL] CHECK OPTION for views for now (supported in 9.4)

- Rewrite CURRENT DATE to CURRENT_DATE etc.

- Rewrite SELECT CURRENT_DATE + 10 DAYS to SELECT CURRENT_DATE + INTERVAL '10 DAYS'

- Explicitly schema-qualify DEC()/DECIMAL()/CHAR()/INT() functions to db2.FOO() (see next)

- Other SQL functions (in views and triggers) not supported by Postgres provided by compatibility layer
DB2 Compatibility Layer (db2fce)

- Similar (in spirit) to orafce, only SQL-functions so far
- https://github.com/credativ/db2fce, PostgreSQL license
- SYSIBM.SYSDUMMY1 view (similar to Oracle’s DUAL table)
  - SELECT 1 FROM SYSIBM.SYSDUMMY1;
- db2 Schema:
  - Time/Date:
    MICROSECOND() / SECOND() / MINUTE() / HOUR() / DAY() / MONTH() / YEAR() / DAYS() / MONTHS_BETWEEN()
  - String: LOCATE() / TRANSLATE()
  - Casts: CHAR() / INTEGER() / INT() / DOUBLE() / DECIMAL() / DEC()
  - Aliases: VALUE() (for coalesce()), DOUBLE (for DOUBLE PRECISION type), ^= (for <> / != operators), !!(for || operator)
The BLOB column is converted to BYTEA by the schema migration.

Tables with BLOBs have an additional column `DB2_GENERATED_ROWID_FOR_LOBS`.

In addition, an AUXILIARY TABLE exists for every table with BLOBS, which has 3 columns:

- AUXID VARCHAR(17)
- AUXVALUE BLOB
- AUXVER SMALLINT

The name of the AUXILIARY TABLE appears to be the name of the main table with the last char replaced with an L.
Schema-Migration, Sequences

- Two types, normal `CREATE SEQUENCE` and implicit `IDENTITY GENERATED` (SERIAL-like)
- Normal sequences `SEQTYPE = 'S'`
  - migrated without post-processing
- Implicit sequences `SEQTYPE = 'I'`
  - `INTEGER DEFAULT IDENTITY GENERATED [ALWAYS|BY DEFAULT]`
  - Implicitly created sequence named `SEQ + 12 random chars`
  - Corresponding column registered in `SYSIBM.SYSSEQUENCESDEP`
  - Ignore implicit sequence and rewrite to SERIAL in post-processing
- Current sequence value in `SYSIBM.SYSSEQUENCES.MAXASSIGNEDVAL` system table column
DB2 trigger functions are inline, i.e. directly attached to the `CREATE TRIGGER` SQL

Triggers are included in the XML schema dump, but not treated by the XSLT script

Custom XSLT extracts triggers from XML and a Perl script migrates trigger
  - Creates trigger and corresponding trigger function
  - Reverts `REFERENCING (NEW|OLD) AS` aliasing

Trigger function body is post-processed like views (`CURRENT DATE` etc.)
  - Additionally removes `BEGIN ATOMIC ... END`
Exporting Data from DB2 UDB

▶ DB2 UNLOAD
  ▶ http://www-01.ibm.com/support/knowledgecenter/SSEPEK_10.0.0/com.ibm.db2z10.doc.ugref/src/tpc/db2z_utl_unload.dita
  ▶ Able to write CSV format, see http://www-01.ibm.com/support/knowledgecenter/SSEPEK_10.0.0/com.ibm.db2z10.doc.ugref/src/tpc/db2z_unloaddelimitedfiles.dita

▶ JCL CSV BATCH EXPORT via FM/DB2 File Manager for z/OS
  ▶ Wrapper around DB2 UNLOAD?
  ▶ Able to write CSV format, allows for batch exports
Exporting Data from DB2 UDB

- Via JDBC or ODBC (DB2CLI)
  - Ispirer
  - SQLWorkbench/J
  - Other Tools
Export/Import Format

- Possible Formats: COPY CSV, COPY TEXT
- Separate values for DELIMITER, NULL, QUOTE (CSV only) possible
- Postgres/psql can import any
- pgloader is/was more restrictive
Exporting Data with SQLWorkbench/J

- WbCopy program can directly copy between source and target DB
  - Cannot use Postgres’ COPY interface for loading
  - Not very flexible regarding encoding, data formatting and other issues either
  - Discarded

- WbExport
  - can write various formats: CSV, SQL, XML, JSON, XLS, ...
  - Can export data for a whole schema in one run, or for specific tables
  - CSV (-type=text) can be coerced to write both COPY CSV and COPY TEXT
Exporting Data, Details

- WbExport -type=text -schema="$SCHEMA" -sourceTable="*" -types=TABLE -outputDir="$SCHEMA" -showProgress -encoding="UTF-8" -escapeText="pgcopy" -formatFile=postgres -timestampFormat="yyyy-MM-dd HH:mm:ss.SSSSSS" -decimalDigits=0 -delimiter=\t -trimCharData -nullString="\N" -header=false -blobType="pghex"
Data-Migration, Sequences

SQLWorkench/J so far does not retrieve NEXT_VALUE for sequences, so direct SQL queries are used

SQL for sequences:
SELECT SQ.SCHEMA, SQ.NAME, SQ.MAXASSIGNEDVAL FROM SYSIBM.SYSSEQUENCES SQ WHERE SQ.SEQTYPE = 'S' AND SQ.MAXASSIGNEDVAL IS NOT NULL AND SQ.MAXASSIGNEDVAL > 1 AND SQ.SCHEMA = UPPER('${SCHEMA}');

SQL for serials:
SELECT SQ.SCHEMA, SQ.NAME, SQ.MAXASSIGNEDVAL, SQDEP.DNAME, SQDEP.DCOLNAME FROM SYSIBM.SYSSEQUENCES SQ LEFT JOIN SYSIBM.SYSSEQUENCESDEP SQDEP ON SQ.SEQUENCEID = SQDEP.BSEQUENCEID WHERE SQ.SEQTYPE = 'I' AND SQ.MAXASSIGNEDVAL IS NOT NULL AND SQ.MAXASSIGNEDVAL > 1 AND SQ.SCHEMA = UPPER('${SCHEMA}');

Output then fed to a script updating sequence RESTART value
Cross-Database Data

- Most programs need to central customer data, kept in DB2
- Once a program is migrated, how does it access this data?
- Via a second JDBC connection to DB2
- Syncing via migration jobs
  - in-house written Java programs similar to existing ones
  - Pentaho Kettle or a similar ETL tool
  - SQLWorkbench/J can sync tables via WbCopy
    -mode=update,insert -syncDelete=true command
- Via a Foreign-Data-Wrapper
  - SQLAlchemy DB2 (db2_sa DB2 CLI ODBC python driver) via multicorn_fdw
    - DB2 CLI needs a different connection than JDBC
    - Querying foreign tables with multicorn results in errors so far
    - ibm_db_dbi::ProgrammingError: SQLNumResultCols failed: [IBM][CLI Driver][DB2] SQL0204N "$USER.$TABLE" is an undefined name. SQLSTATE=42704 SQLCODE=-204
**pgloader COPY TEXT Issues (fixed)**

- **Fixed:**
  - #218 “COPY TEXT format required enumeration of column names”
    - Contrary to the COPY CSV format, the COPY TEXT format option apparently requires that the column names are enumerated in the load file.
  - #222 “Does not properly decode Hex-encoded characters in COPY TEXT format”
    - If a dump contains hex-encoded characters like \x1a, pgloader inserts that as a literal \x1a string in the database, not as the hex value 0x1A

- **Open Wish List Issues**
  - #217 “custom NULL-value”
    - The COPY-Syntax allows for a custom NULL-value, would be good if that could be folded into the LOAD-File Syntax so CSV-Data with non-standard NULL values can be easily loaded.

Michael Banck <michael.banck@credativ.de>
If there are lots of errors, loading speed degrades rapidly

Table with 10192 rows:
  - 1005 errors: 00:04:58
  - 4 errors: 00:00:07

Table with 510576 rows:
  - 130095 errors: 16:05:31
  - 0 errors: 00:01:12

On the other psql just aborts on the first error

SQLWorkbench/J WbImport program just hangs on errors without user-visible error message
Data Migration, Encountered Problems

- Several tables had \x00 values in them, resulting in invalid byte sequence for encoding UTF8: 0x00 errors
  - Patched WbExport to drop \x00 for -escapeText="pgcopy"
- Exporting tables with a column USER resulted in WbExport writing the username of the person running it
  - Add USER to ~/SQLWorkbench/reserved_words.wb
- Default timestamp resolution was too coarse, leading to duplicate key violations
  - Worked around via -timestampFormat="yyyy-MM-dd HH:mm:ss.SSSSSSS"
- NUMERIC(X,Y) columns were exported with a precision of 2 only
  - Add workbench.gui.display.maxfractiondigits=0 to ~/SQLWorkbench/workbench.settings
Full Migration

- Dump schema to XML
- Convert XML to DDL and post-process
- Drop indexes, constraints and triggers
- Export data
- Import data
- Set sequences
- Create indexes, constraints and triggers
Full Migration, Timings

- First full migration of TEST instance took around 10 hours
- PROD instance is 4-5 times bigger
- Parallelized creation of indexes and constraints via parallel tool:
  
  ```
  cat $SQLFILE | parallel -j$NUM_JOBS "psql service=$SERVICE -c { }
  
  
  "
  ```
- Decoupled exporting/importing - exporting run in background and schema migration/data import wait for triggerfile before starting
- Parallelized import of data via parallel tool at table level
- Full migration of TEST instance down to 6 hours
  - Most time spent in biggest schema (2:15/3:00 for export/import)
- Data export could be further parallelized (if DB2 keeps up)

Michael Banck <michael.banck@credativ.de>
Summary

- Business-critical DB2 UDB being migrated to Postgres in a German regional government ministry
- First schema to be migrated in November 2015, full migration planned till mid-2016
- Proof-of-Concept automatic migration of TEST instance working
  - Schemas migrated via SQLWorkbench/J XML/XSLT and post-processing
  - Some DB2 compatibility provided by db2fce extension
  - Data exported by SQLWorkbench/J and imported with pgloader
- Some more performance tuning needed for PROD migration
Contact

- General Contact <info@credativ.de>
- Michael Banck <michael.banck@credativ.de>
- http://www.credativ.de/postgresql-competence-center
- https://github.com/credativ/db2fce
- Questions?