



Dumping the Mainframe: Migration Study from DB2 UDB to PostgreSQL

Michael Banck <michael.banck@credativ.de>

PGConf.DE 2015

- ▶ DB2 UDB is the z/OS mainframe edition of IBM's DB2 database
- ▶ DB2 UDB central database and application server ("the Host") in German state 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 users apply for subsidies

- ▶ 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

- ▶ 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)

- ▶ Postgres
 - ▶ PostgreSQL-9.2/PostGIS-2.0 (upgrade to 9.4/2.1 planned for January)
 - ▶ 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
 - ▶ 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)

- ▶ 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 end of 2015

- ▶ Migration Guide in PostgreSQL wiki
 - ▶ <https://wiki.postgresql.org/wiki/File:DB2UDB-to-PG.pdf>
 - ▶ 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

- ▶ 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 data compared to EBCDIC 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
 - ▶ Transactions abort in Postgres after first error, unlike DB2
 - ▶ Github issue for pgjdbc driver opened

- ▶ SQLWorkbench/J (<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>)
 - ▶ 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)

- ▶ `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`

- ▶ 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

- ▶ 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)
- ▶ `search_path` changed to 'db2, public' for application roles

- ▶ 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

- ▶ 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
- ▶ SQLWorkench/J so far does not retrieve NEXT_VALUE
 - ▶ Current sequence value in SYSIBM.SYSSEQUENCES.
MAXASSIGNEDVAL system table column
 - ▶ DB2 SQL query outputs those values which are then set via ALTER SEQUENCE [...] RESTART in Postgres

- ▶ 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`

- ▶ DB2 UNLOAD
 - ▶ 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
 - ▶ http://www-01.ibm.com/support/knowledgecenter/SSXJAV_13.1.0/com.ibm.filemanager.doc_13.1/db2/exportcmd.htm
 - ▶ Wrapper around DB2 UNLOAD?
 - ▶ Able to write CSV format, allows for batch exports
- ▶ No direct access, data transfer unclear, DBA not helping
- ▶ Looked at JDBC as an alternative: Ispirer (commercial), SQLWorkbench/J, Pentaho Data Integration, others

- ▶ Possible Formats: COPY CSV, COPY TEXT, SQL
- ▶ SQL-format cannot be bulk loaded
- ▶ Separate values for DELIMITER, NULL, QUOTE (CSV only) possible
- ▶ SQLWorkbench/J can export all three if proper options are used
- ▶ Postgres/psql can import any
- ▶ pgloader is/was more restrictive
 - ▶ No custom NULL value for COPY CSV format (back then)
 - ▶ No format options whatsoever for COPY TEXT format
 - ▶ COPY TEXT format requires list of column names (CSV doesn't)
- ▶ Settled on COPY TEXT format, massaging the SQLWorkbench/J CSV so that pgloader thinks it is COPY TEXT
 - ▶ Delimiter is \t
 - ▶ Null string is "\N"
 - ▶ Trailing whitespace in text columns gets trimmed

- ▶ 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
- ▶ Nightly dump/restore of required tables
- ▶ Syncing via migration jobs
 - ▶ in-house written Java programs similar to existing ones
 - ▶ Pentaho Data Integration 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 multicore_fdw
 - ▶ Works for simple queries (2x slower)
 - ▶ Really bad for complex joins (60x slower)
 - ▶ Dedicated DB2 CLI FDW (to be written)

- ▶ 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.

- ▶ 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
- ▶ `psql` just aborts on the first error
- ▶ `SQLWorkbench/J WbImport` program just hangs on errors without user-visible error message

- ▶ 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.SSSSSS"`
- ▶ `NUMERIC(X,Y)` columns were exported with a precision of 2 only
 - ▶ Add `workbench.gui.display.maxfractiondigits=0` to `~/SQLWorkbench/workbench.settings`

- ▶ 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
- ▶ Create grants

- ▶ 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)

- ▶ 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

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

- ▶ 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`

► WbExport options:

```
-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"
```

► Dumps data into *SCHEMA*/TABLE.txt

- ▶ pgloader load script:
LOAD COPY FROM copy:///\${SCHEMA}/\${TABLE}.txt
\$COLUMNS
INTO postgresql:///\${SCHEMA}.\${TABLE} WITH truncate
SET client_encoding to 'utf8', work_mem to '14MB',
standard_conforming_strings to 'on'
BEFORE LOAD DO \\${\$} SET ROLE \${SCHEMA}_owner; \\${\$};
- ▶ \${COLUMNS} extracted from WbExport's COPY import script

- ▶ SQL for sequences:

```
SELECT SEQ.SCHEMA, SEQ.NAME, SEQ.MAXASSIGNEDVAL FROM  
SYSIBM.SYSSEQUENCES SEQ WHERE SEQ.SEQTYPE = 'S' AND  
SEQ.MAXASSIGNEDVAL IS NOT NULL AND  
SEQ.MAXASSIGNEDVAL > 1 AND SEQ.SCHEMA =  
UPPER('${SCHEMA}');
```

- ▶ SQL for serials:

```
SELECT SEQ.SCHEMA, SEQ.NAME, SEQ.MAXASSIGNEDVAL,  
SEQDEP.DNAME, SEQDEP.DCOLNAME FROM  
SYSIBM.SYSSEQUENCES SEQ LEFT JOIN  
SYSIBM.SYSSEQUENCESDEP SEQDEP ON SEQ.SEQUENCEID =  
SEQDEP.BSEQUENCEID WHERE SEQ.SEQTYPE = 'I' AND  
SEQ.MAXASSIGNEDVAL IS NOT NULL AND  
SEQ.MAXASSIGNEDVAL > 1 AND SEQ.SCHEMA =  
UPPER('${SCHEMA}');
```