Migration From DB2 in a Large Public Setting: Lessons Learned

Balázs Bárány and Michael Banck

PGConf.EU 2017

Introduction

- Federate state ministry in Germany
- Hosting by state's central IT service centre
- Michael worked as an external consultant for both
- Balázs took over DBA role and migration lead at ministry
- Michael continues to support the service centre's Postgres operations

Introduction

- Proof-of-Concept version of this talk presented at pgconf.eu 2015
  - Slides still available on `https://wiki.postgresql.org/`
- 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/I)
  - Natural (and PL/I) programs 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

Prior Postgres Usage

- Postgres introduced about 12 years ago due to geospatial requirements (PostGIS, nothing comparable for DB2 at the time)
- Started using Postgres for smaller, non-critical projects about 7 years ago
- Modernized the software stack merging geospatial and business data about 5 years ago
- In-house code development of Java web applications (Tomcat/Hibernate)
- 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
- Migrated all Natural and PL/I 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 migrated to modernized framework
- PL/I Applications
  - Rewritten in Java (only a few)
- Natural Applications
  - Automatic migration/transcription into (un-Java, but correct) Java on DB2
  - Test of migrated "Java" application on the original data
  - Test on schema migrated to PostgreSQL
  - Multi-year project facilitated by an external consultancy

Setup Before the Migration

- Postgres
  - Postgres-9.4/PostGIS-2.1 (upgrade to 9.6/2.3 planned in late 2017)
  - SLES11, 64 cores, 512 GB memory, SAN storage
  - HA 2-node setup using Pacemaker, two streaming standbys (one disaster recovery standby)
  - Roughly 1.3 TB data, 22 schemas, 440 tables, 180 views in PROD
  - Almost no stored procedures (around 10)
- DB2
  - DB2 UDB Version 10
  - Roughly 600 GB data in PROD instance
  - Almost no stored procedures (around 20, written in PL/I)

Steps towards migration

- Natural migration to Java delayed
  - Originally planned for November 2015, ready in July 2017
  - Gave us one year for testing the migration process
- Several Java projects maintained by external developers have been (mostly) successfully tested on local Postgres deployments
- First production migration of a complex Java program and its schema done in early 2016
  - Required daily migration of core tables (DB2 to PostgreSQL) starting at that point
- Separate DB2 database operated by the ministry migrated from mainframe to another Postgres instance successfully in Q1/2017
  - Just a data migration, schema was migrated by hand

Tools used for the migration

- SQLWorkbench/J (`http://www.sql-workbench.net`) v117.6
  - Java-based, DB-agnostic workbench GUI
  - Heavily-used in-house already, installed on workstations
  - Allows for headless script/batch operation via internal programs
  - Used for schema migration and data export from DB2
- pgloader (`http://pgloader.io`) 3.2.0
  - Postgres bulk loading and migration tool written in Lisp
  - Open Source (PostgreSQL license)
  - Written and maintained by Dimitri Fontaine (PostgreSQL major contributor)
  - Used for data import into Postgres

Schema-Migration

- 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 triggers separately
  - Ignore functions/stored procedures (out-of-scope)

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()/STRIP()`
  - Casts: `CHAR()/INTEGER()/INT()/DOUBLE()/DECIMAL()/DEC()`
  - Aliases: `VALUE()` (for `coalesce()`), `DOUBLE` (for `DOUBLE PRECISION` type), ˆ= (for <> / != operators), `!!` (for || operator)
- `search_path` changed to `'db2, public'` in database configuration

Data Migration, Encountered Problems

- Several tables had `\x00` values in them, resulting in `"invalid byte sequence for encoding UTF8: 0x00"` errors
- Exporting tables with a column `USER` resulted in `WbExport` writing the username of the person running it
- Default timestamp resolution was too coarse, leading to duplicate key violations
- `NUMERIC(X,Y)` columns were exported with a precision of 2 only
- Import of timestamps invalid in daylight saving change time rejected by PostgreSQL
  - Export them with `-Duser.timezone=GMT` despite being local (Central European) timestamps
- Objects in target DB with the same name as in the source, but different contents
  - Renamed in source system

Full Migration

- Closed databases for “normal” usage
  - Source DB switched to read only
  - PostgreSQL: removed USAGE on schemas from non-DBA users
  - Notified users with open connections
  - Deactivated HA watchdog, disaster recovery
- Scripted (automatic) migration process:
  - Dumped schema to XML, converted to DDL, post-processed
  - Dropped indexes, constraints and triggers
  - Exported data
  - Imported data
  - Set sequence values
  - Created indexes, constraints and triggers
  - Created grants

Full Migration, Results

- Full migration in 3 processes (different schemas) in 14 hours incl. index building
  - Database was gaining 1 GB every 2 minutes when 3 processes were writing
  - Up to 80 Mbit/sec both incoming and outgoing on the network interfaces
  - Up to 120 Mbit/sec when writing (4 CPUs at the limit)
- Data validation jobs started whenever a schema was ready
  - Minimal differences in floating point representation
  - Everything else identical, including binary data and sequence values
- Watching logs for errors while the applications start
  - A few schema or table permissions were missing
  - Tables missing
    - Dropped from source system before, application was not tested

SQL Differences

- 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
  - Operators like `!=` instead of <>
  - "Default default" values: attribute `INTEGER DEFAULT`
    - Like attribute `INTEGER` **DEFAULT 0** in PostgreSQL

Behaviour Differences

- DB2 sorts data by `GROUP BY` keys, no need for `ORDER BY`
  - PostgreSQL doesn't guarantee this
- Sorting differences
  - EBCDIC: numbers after characters (ASCII: before)
  - EBCDIC: special characters after characters and numbers
    - Similar behaviour with C collation in Postgres
  - Applications using ECBDIC order **inside** values
- Application got "duplicate key value" error
  - Tried to use `CURRENT_TIMESTAMP` as primary key
  - Postgres: Start of transaction. DB2: current time regardless of transaction.
- Application trying to insert `NULL` into field with `DEFAULT`
  - DB2 accepted the `NULL` and used the `DEFAULT`
  - For Postgres, we had to create a trigger to fix the `INSERT`s

Transaction handling

- Some queries in DB2 using `WITH UR`
  - `UR` = Uncommitted Read
  - Performance optimization to avoid locks (but getting inconsistent data)
  - No comparable built-in mechanism in PostgreSQL, but locking not a huge problem
- PostgreSQL cancels the entire transaction after an error, `ROLLBACK` necessary
  - Some program logic needed changes (bad error handling, errors used for branching logic)

Performance

- Most applications comparable or faster after migration
  - A few with mainframe access patterns slower
- Some smaller impacts: different indexing, JDBC oddities, . . .
- A few huge problems (application not usable)
  - Indexes correct, query is fast in SQL client
  - Not using "best" index when called from prepared query
  - Found the reason: ID (integer) field was queried with `NUMERIC` parameter

JDBC: defaultRowFetchSize

- Applications started crashing with Out of Memory errors
- Pattern: `SELECT * FROM <big table>`, read some rows for display
- PostgreSQL JDBC reads the whole data set by default
  - DB2 didn't, so the application was working fine
- Solution: setting `defaultRowFetchSize` to a reasonable value (e. g. 10000)
- No negative effects (negligible performance hit?)

JDBC: stringtype

- Errors with prepared statements, but query works in SQL client
  - This works (automatic casting to date):

```
SELECT * FROM t WHERE dat = '2017-08-01';
```

  - This doesn't (with `param1 = '2017-08-01'`):

```
SELECT * FROM t WHERE dat = ?;
```

  - Could affect date, timestamp, numeric and Boolean columns
- Comparison of date type with "forced" text type fails, no automatic cast
- Solution: `stringtype=unknown`
  - Fine for the affected applications
  - Might be wrong in some situations, e. g. garbled date format

Summary

- Customer is happy
- Big savings on mainframe costs
  - mainframe performance units, DB2, Natural runtime, . . .
- One modern database for business and GIS data, pleasant usage
- Better standards for DB roles and permissions, change management etc.

Contact

- DB2 compatibility extension: `https://github.com/credativ/db2fce`
- Michael Banck `<michael.banck@credativ.de>`
  - `http://www.credativ.de/postgresql-competence-center`
- Balázs Bárány `<balazs@tud.at>`
  - `https://datascientist.at/`

Questions?