PGECons
PostgreSQL Enterprise Consortium

PostgreSQL エンタープライズ・コンソーシアム 技術部会 WG#2

アプリケーション移行実践編

製作者
NTT ソフトウェア株式会社

改訂履歴

版	改訂日	変更内容
1.0	2013/04/22	初版

はじめに

■本資料の概要と目的

本資料では、異種 DBMS で稼動しているアプリケーションを PostgreSQL 上に移行し、その移行手順と評価を記載します。

■本資料で扱う用語の定義

本資料では、幾つかの曖昧な意味としてとらえることができる用語を用いますが、次のような意味で記載がされています。

表 1: 用語定義

No.	用語	意味
1	DBMS	データベース管理システムを指します。ここでは、PostgreSQL および異種 DBMS の総称として利用します。
2	異種 DBMS	PostgreSQL ではない、データベース管理システムを指します。 本資料では、Oracle Database、MySQL、および Microsoft SQL Server が該当します。
3	Oracle	データベース管理システムの Oracle Database を指します。
4	MySQL	データベース管理システムの MySQL を指します。
5	SQL Server	データベース管理システムの Microsoft SQL Server を指します。

■本資料で扱うソフトウェア

本資料で登場する異種 DBMS、およびツールの一覧を記載します。

表 2: 異種 DBMS 一覧

No.	DBMS	バージョン
1	Oracle Database	11g R2
2	Microsoft SQL Server	2008 R2 Express
3	MySQL Server	5.5

表 3: ツール一覧

No.	ツール	バージョン	ライセンス	入手元(URI)	概要
1	Eclipse IDE for Java Developers	4.2.1	Eclipse Public License Version 1.0	http://www.eclipse.org/downloads/	統合開発環境。
2	Commander4j	3.98	GNU Lesser General Public License v2	http://www.commander4j.com/	生産記録と業界標準（EAN 128）GTIN バーコードラベルの作成のための Java ベースのアプリケーション。
3	db_syntax_diff	2.0	Th PostgreSQL License	https://github.com/db-syntax-diff	Oracle 上で稼動するアプリケーションを PostgreSQL へ移行する際に修正が必要となる項目の概要を報告する支援ツール。

目次

1.アプリケーション移行トライアル概要....6
1.1.目的....6
1.2.移行対象アプリケーション....6
1.3.対象データベース、JDBCドライバ....6
1.4.SQL 差分調査ツール....7
2.移行対象アプリケーションについて....8
2.1.Commander4j ディレクトリ構成....8
2.2.SQL 実行イメージ....9
3.移行準備....11
3.1.PostgreSQL サーバの構築....11
3.2.db_syntax_diff の適用....11
4.移行手順....14
4.1.移行対象の修正....14
4.2.移行修正の動作確認....19
5.移行実施結果....20
5.1.db_syntax_diff 実行結果....20
5.2.db_syntax_diff 未報告の事象....21
5.3.移行作業の各作業時間割合....22
6.まとめ....24
7.別紙・付録一覧....24

1. アプリケーション移行トライアル概要

1.1. 目的

PostgreSQL 未対応のアプリケーション「Commander4j」を PostgreSQL でも動作するように修正を行います。
本作業を通じて、アプリケーション移行に必要な手順の明確化、および SQL 抽出ツール適用結果の評価を行います。

1.2. 移行対象アプリケーション

Commander4j は生産記録と業界標準 (EAN 128) GTIN バーコードラベルの作成のための Java ベースのアプリケーションです。Commander4j は、SQL Server や Oracle 、MySQL データベースで使用することができます。
Commander4j の概要を以下に記載します。

表 1.1: 移行対象アプリケーション情報

項目	内容	備考
名称	Commander4j	
バージョン	3.98	
対象アプリケーション	デスクトップアプリケーション版	Web アプリケーション版も存在するが今回の移行作業では対象外とする。
記述言語	Java、XML	
DB へのインターフェース	JDBC	
対応 DB	Oracle Database 11g R2 Microsoft SQL Server 2008 R2 Express MySQL Server 5.5	
Java ファイル数	272 ファイル	
ステップ数	71.242Ks	
DB テーブル数	39 テーブル	
DB ビュー数	2 ビュー	
SQL 数	3390	
ストアドプロシージャ数	0	
入手元(URL)	http://www.commander4j.com/	
ライセンス	GNU Lesser General Public License v2	

1.3. 対象データベース、JDBC ドライバ

移行先の PostgreSQL のバージョンと使用する JDBC ドライバを以下に記載します。

表 1.2: 移行先データベース情報

No.	名称	バージョン	入手元(URL)
1	PostgreSQL	9.2.2	http://www.postgresql.org/ftp/source/v9.2.2/
2	JDBC ドライバ	9.2 Build 1002 (Type4)	http://jdbc.postgresql.org/download.html

1.4. SQL 差分調査ツール

アプリケーション移行に使用する SQL 差分調査ツールを以下に記載します。

表 1.3: 移行先データベース情報

No.	名称	バージョン	入手元(URL)
1	db-syntax-diff	2.0	https://github.com/db-syntax-diff

2. 移行対象アプリケーションについて

Commander4j の仕様に関して移行に関連する箇所のみ記載します。

2.1. Commander4j ディレクトリ構成

移行に関連するディレクトリの構成を以下に記載します。

```
[./]
|
+--- lib/
|      |
|      +---- db/
|      +---- …
|
+--- src/
|
+--- xml/
|      |
|      +--- schema/
|      |      |
|      |      +--- com.microsoft.sqlserver.jdbc.SQLServerDriver/
|      |      +--- com.mysql.jdbc.Driver/
|      |      +--- oracle.jdbc.driver.OracleDriver/
|      |
|      +--- sql/
|      |      |
|      |      +--- sql.com.microsoft.sqlserver.jdbc.SQLServerDriver.xml
|      |      +--- sql.com.mysql.jdbc.Driver.xml
|      |      +--- sql.oracle.jdbc.driver.OracleDriver.xml
|      |
|      +--- view/
|      |      |
|      |     +--- view.com.microsoft.sqlserver.jdbc.SQLServerDriver.xml
|      |      +--- view.com.mysql.jdbc.Driver.xml
|      |      +--- view.oracle.jdbc.driver.OracleDriver.xml
|      |
|      +--- …
|
+--- …
```

lib/db・・・・・・・・・・各 DBMS 用 JDBC ドライバの jar ファイルを格納するディレクトリ
src・・・・・・・・・・・・ソースファイルの Java ファイルを格納するディレクトリ
xml/schema・・・・各 DBMS 毎に JDBC ドライバ名のディレクトリを作成し、環境構築時に実行される SQL 文を記載した、
XML ファイルを格納するディレクトリ
sql・・・・・・・・・・・・操作時に実行される SQL 文を記載した、各 DBMS 毎に JDBC ドライバ名で区別された XML
ファイルを格納するディレクトリ
view・・・・・・・・・・・上記 XML ファイルに記述された SQL 内のマクロを記載した、各 DBMS 毎に JDBC ドライバ名で
区別された XML ファイルを格納するディレクトリ

2.2. SQL 実行イメージ

Commander4j の SQL 実行イメージを以下に記載します。

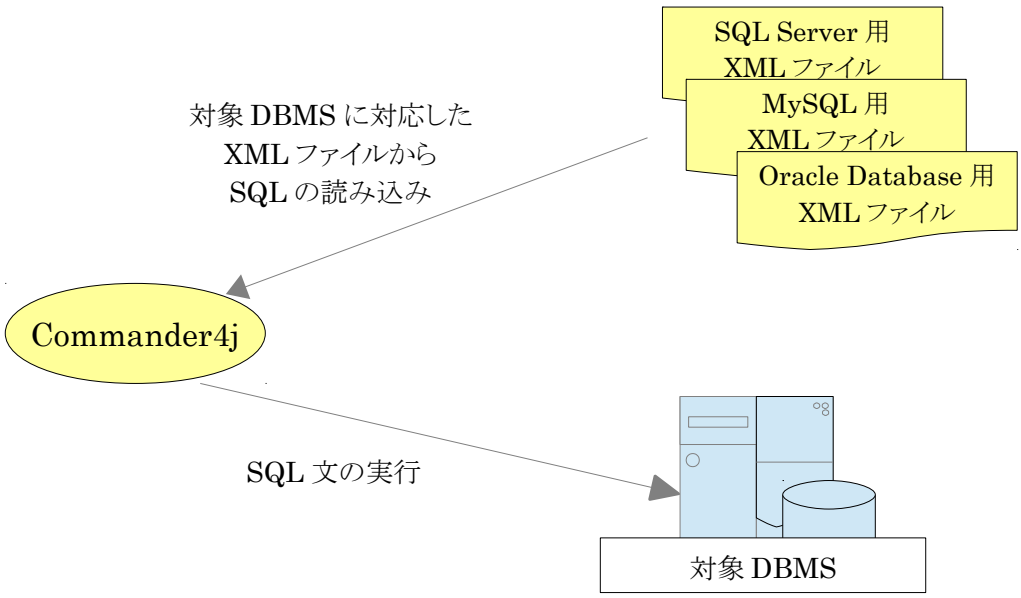

図 2.1SQL 実行イメージ

xml ディレクトリ配下に各 DBMS 毎の XML ファイルが準備されており、SQL 実行時は対象の DBMS に対応した XML ファイルから SQL 文を読み込んで SQL を実行します。ツールの環境構築時とツール実行時は SQL の読み込み方法が異なります。

環境構築時は JDBC ドライバ配下の全ての XML ファイルに記述された SQL を読み込み実行します。

ツール実行時は JDBC ドライバ名で区別された XML ファイルから実行する SQL を id タグを利用して読み込み実行します。

それぞれの XML ファイルのイメージを以下に示します。

環境構築時の XML ファイル例

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE DDL SYSTEM "schema.dtd">
<DDL>
   <statement>
      ALTER TABLE app_pallet_history MODIFY (user_id VARCHAR2(20))
   </statement>
   <statement>
      ALTER TABLE sys_user_group_membership MODIFY (user_id VARCHAR2(20))
   </statement>
   <statement>
      ALTER TABLE sys_users MODIFY (user_id VARCHAR2(20))
   </statement>
</DDL>
```

ツール実行時の XML ファイル例

```
<?xml version="1.0" encoding="UTF-8"?> <!DOCTYPE SQL SYSTEM "sql.dtd">
<SQL>
   <jdbcDriver Type="oracle.jdbc.driver.OracleDriver">
   …
   <statement>
      <id>JDBPalletHistory.selectWithExpiry</id>
      <text>select * from [view_pallet_history_expiry]</text>
   </statement>
   …
   </jdbcDriver>
</SQL>
```

3. 移行準備

移行準備として以下の手順を実行します。

1. PostgreSQL サーバの構築
2. db_syntax_diff の適用

それぞれの手順の詳細を以降に記載します。

3.1. PostgreSQL サーバの構築

1. PostgreSQL サーバの構築
   移行対象先 DBMS サーバとして、PostgresSQL サーバを構築します。

3.2. db_syntax_diff の適用

1. db_syntax_diff のインストール
   db_syntax_diff の利用マニュアルに従い db_syntax_diff のインストールを行います。
   インストール後のディレクトリ構成は以下のとおりです。

```
[./]
|
+--- bin/
|     |
|     +--- pg_sqlextract_wrapper.sh
|     +--- convert_utf8.pl
|
+--- config/
|     |
|     +--- wrapper.ctl.sample
|
+--- output/
|
+--- tmp/
```

2. db_syntax_diff 実行対象の選定
   db_syntax_diff の適用対象を選定します。
   選定基準は「Oracle Database 用の SQL が記載されている可能性があるファイル・ディレクトリ」となります。
   今回、以下のとおり実行対象を選定しました。

実行対象		ディレクトリ	フォーマット
Java ソースコード		/src 配下一式	Java ファイル
SQL	スキーマ定義	/xml/schema/oracle.jdbc.driver.OracleDriver 配下一式	XML ファイル
	ビュー定義	/xml/view/view.oracle.jdbc.driver.OracleDriver.xml	XML ファイル
	DML 定義	/xml/sqlsql.oracle.jdbc.driver.OracleDriver.xml	XML ファイル

3. 制御ファイルの作成
上記結果に基づき、db_syntax_diff の制御ファイルを作成します。
制御ファイルの内容は以下のとおりです。

実行対象		制御ファイル名	制御ファイルの内容
Java ソースコード		controlfile_java	TARGET_PATH=***Java ソースコード配置場所*** TARGET_EXT=**java** TARGET_MODE=**java** TARGET_CHAR=utf8 POSTGRESQL_VERSION=9.2 OUTPUT_DIR_NAME=java
SQL	スキーマ定義	controlfile_schema	TARGET_PATH=***スキーマ定義配置場所*** TARGET_EXT=**xml** TARGET_MODE=**sql** TARGET_CHAR=utf8 POSTGRESQL_VERSION=9.2 OUTPUT_DIR_NAME=schema
	ビュー定義	controlfile_view	TARGET_PATH=***ビュー定義配置場所*** TARGET_EXT=**xml** TARGET_MODE=**sql** TARGET_CHAR=utf8 POSTGRESQL_VERSION=9.2 OUTPUT_DIR_NAME=view
	DML 定義	controlfile_sql	TARGET_PATH=***DML 定義配置場所*** TARGET_EXT=**xml** TARGET_MODE=**sql** TARGET_CHAR=utf8 POSTGRESQL_VERSION=9.2 OUTPUT_DIR_NAME=sql

4. XML ファイルの変換
スキーマ定義、ビュー定義、DML 定義の 3 つについては XML ファイルとなっているため、制御ファイルを
TARGET_EXT=xml （抽出対象ファイルの拡張子）
TARGET_MODE=sql （抽出ツールの実行モード）
として作成しました。
実行モードが sql の場合、各 SQL の末尾に";"（セミコロン）がついていないと db_syntax_diff は SQL として認識しません。今回、XML ファイル中に定義されている SQL には末尾に";"（セミコロン）がついていなかったため、XML ファイルの変換を行い SQL ファイルの末尾に";"を付与しました。

XML ファイルの変換例（変換前）

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE DDL SYSTEM "schema.dtd">
<DDL>
  <statement>
    ALTER TABLE app_pallet_history MODIFY (user_id VARCHAR2(20))
  </statement>
  <statement>
    ALTER TABLE sys_user_group_membership MODIFY (user_id VARCHAR2(20))
  </statement>
  <statement>
    ALTER TABLE sys_users MODIFY (user_id VARCHAR2(20))
  </statement>
</DDL>
```

XML ファイルの変換例（変換後）

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE DDL SYSTEM "schema.dtd">
<DDL>
  <statement>
    ALTER TABLE app_pallet_history MODIFY (user_id VARCHAR2(20));
  </statement>
  <statement>
    ALTER TABLE sys_user_group_membership MODIFY (user_id VARCHAR2(20));
  </statement>
  <statement>
    ALTER TABLE sys_users MODIFY (user_id VARCHAR2(20));
  </statement>
</DDL>
```

5. db_syntax_diff の実行
ここまでの結果を用いて db_syntax_diff を実行します。
実行方法は以下のとおりです。各制御ファイルごとに計 4 回実行します。

```
./bin/db_syntax_diff_wrapper.sh 制御ファイル名
```

6. 出力結果の確認
"./output"配下に実行結果が出力されていることを確認します。
実行結果の詳細については「付録 1：db_syntax_diff 実行結果」をご参照ください。

4. 移行手順

今回の移行手順は以下のとおりとなります。

1. PostgreSQL 用 JDBC ドライバファイルの追加
2. DBMS 別分岐処理への PostgreSQL 用処理の追加
3. PostgreSQL 用設定ファイルの追加
4. SQL の修正
5. PostgreSQL への JDBC 接続確認
6. SQL 修正内容の動作確認

それぞれの手順の詳細を以降に記載します。

4.1. 移行対象の修正

4.1.1. PostgreSQL 用 JDBC ドライバの追加

1. PostgreSQL 用 JDBC ドライバファイルの追加
   “./lib/db”配下に各 DBMS の JDBC ドライバファイルが格納されています。
   PostgreSQL 用 JDBC ドライバファイルを追加します。

4.1.2. DBMS 別分岐処理への PostgreSQL 用処理の追加

1. 修正箇所の特定
   移行対象アプリケーションは元々複数 DBMS に対応しているので、ソースコード上での修正箇所の特定は容易です。ソースファイルを「jdbc.driver」と「Oracle」のキーワードで grep して修正箇所の特定を行ないます。

2. Java ファイルの修正
   DBMS 別に JDBC ドライバ名で分岐している処理に PostgreSQL 用 JDBC ドライバ名での分岐と処理を追加します。
   修正例(src/com/commander4j/cfg/JFrameHostAdmin.java)を以下に記載します。

ソース修正例(修正前)

```
if (hst.getDatabaseParameters().getjdbcDriver().equals("com.mysql.jdbc.Driver")){
        jComboBoxjdbcDriver.setSelectedIndex(1);
}if (hst.getDatabaseParameters().getjdbcDriver().equals("oracle.jdbc.driver.OracleDriver")){
        jComboBoxjdbcDriver.setSelectedIndex(2);
}if (hst.getDatabaseParameters().getjdbcDriver().equals("com.microsoft.sqlserver.jdbc.SQLServerDriver")){
        jComboBoxjdbcDriver.setSelectedIndex(3);
}if (hst.getDatabaseParameters().getjdbcDriver().equals("http")){
        jComboBoxjdbcDriver.setSelectedIndex(4);
}
```

ソース修正例(修正後)

```
if (hst.getDatabaseParameters().getjdbcDriver().equals("com.mysql.jdbc.Driver")){
        jComboBoxjdbcDriver.setSelectedIndex(1);
}if (hst.getDatabaseParameters().getjdbcDriver().equals("oracle.jdbc.driver.OracleDriver")){
        jComboBoxjdbcDriver.setSelectedIndex(2);
}if (hst.getDatabaseParameters().getjdbcDriver().equals("com.microsoft.sqlserver.jdbc.SQLServerDriver")){
        jComboBoxjdbcDriver.setSelectedIndex(3);
}if (hst.getDatabaseParameters().getjdbcDriver().equals("http")){
        jComboBoxjdbcDriver.setSelectedIndex(4);
}
//PG add ->
if (hst.getDatabaseParameters().getjdbcDriver().equals("org.postgresql.Driver")){
        jComboBoxjdbcDriver.setSelectedIndex(5);
}//PG add end
```

4.1.3. PostgreSQL 用設定ファイルの追加

1. PostgreSQL 用ディレクトリの追加
   XML ファイルは DBMS 別に JDBC ドライバ名で作成されています。
   Oracle 用 XML ファイルを複製して PostgreSQL の XML ファイルを作成します。
   具体的には以下に示す太字のファイルを新規作成します。

```
[./]
|
+--- xml/
     |
     +--- schema/
     |    |
     |    +--- com.microsoft.sqlserver.jdbc.SQLServerDriver/
     |    |    |
     |    |    +--- 000000.xml
     |    |    +--- …
     |    |    +--- 000060.xml
     |    |    +---schema.dtd
     |    |
     |    +--- com.mysql.jdbc.Driver/
     |    |    |
     |    |    +--- 000000.xml
     |    |    +--- …
     |    |    +--- 000060.xml
     |    |    +---schema.dtd
     |    |
     |    +--- oracle.jdbc.driver.OracleDriver/
     |    |    |
     |    |    +--- 000000.xml
     |    |    +--- …
     |    |    +--- 000060.xml
     |    |    +---schema.dtd
     |    |
     |    +--- org.postgresql.Driver/
     |    |    |
     |    |    +--- 000000.xml
     |    |    +--- …
     |    |    +--- 000060.xml
     |    |    +--- schema.dtd
     |
     +--- sql/
     |    |
     |    +--- sql.com.microsoft.sqlserver.jdbc.SQLServerDriver.xml
     |    +--- sql.com.mysql.jdbc.Driver.xml
     |    +--- sql.oracle.jdbc.driver.OracleDriver.xml
     |    +--- sql.org.postgresql.Driver.xml
     |    +--- sql.dtd
     |
     +--- view/
     |    |
     |    +--- view.com.microsoft.sqlserver.jdbc.SQLServerDriver.xmll
     |    +--- view.com.mysql.jdbc.Driver.xml
     |    +--- view.oracle.jdbc.driver.OracleDriver.xml
     |    +--- view.org.postgresql.Driver.xml
     |    +--- view.dtd
```

4.1.4. SQL の修正

1. db_syntax_diff 実行結果の確認
   SQL については db_syntax_diff 実行結果を元に修正を行ないます。
   db_syntax_diff 実行結果である"result.csv"ファイルを元に、SQL を確認し修正の必要有無を判定します。

2. SQL の修正
   修正が必要と判断した SQL を修正します。
   db_syntax_diff 実行結果と修正前後の SQL ファイルの例(xml/schema/org.postgresql.Driver/000054.xml)を以下に記載します。

db_syntax_diff 実行結果(result.csv の抜粋)

/XXX/xml/schema/org.postgresql.jdbc.driver.PostgreSQLDriver/000054.xml,1,206,TYP-006-001,LOW2,NUMBER 型はサポートされていません。NUMERIC 型に変更することで対応可能ですが、位取りの値に、負数および、精度より大きい値は指定できませんので注意が必要です。また、NUMERIC 型は性能上問題となる場合がありますので、厳密に精度を求める必要がない場合は REAL 型や INTEGER 型など、他の型の利用を検討してください。

SQL 修正例(修正前)

```
<statement>  CREATE TABLE "APP_QM_DICTIONARY"
  (       "TEST_ID" VARCHAR2(50 BYTE) NOT NULL ENABLE,
          "FIELD_ALIGNMENT" NUMBER,
          "DESCRIPTION" VARCHAR2(50 BYTE),
          "DATATYPE" VARCHAR2(15 BYTE),
          "UOM" VARCHAR2(10 BYTE),
          "REQUIRED" VARCHAR2(1 BYTE),
          "SELECT_LIST_ID" VARCHAR2(20 BYTE),
          "VISIBLE" VARCHAR2(1 BYTE),
          "EXTENSION_ID" NUMBER,
          "FIELD_WIDTH" NUMBER,
          "FIELD_MAX_CHARS" NUMBER,
           CONSTRAINT "APP_QM_DICTIONARY_PK" PRIMARY KEY ("TEST_ID")
  ENABLE   )</statement>
```

SQL 修正例(修正後)

```
<statement>  CREATE TABLE APP_QM_DICTIONARY
  (         TEST_ID VARCHAR(50) NOT NULL,
            FIELD_ALIGNMENT INT,
            DESCRIPTION VARCHAR(50),
            DATATYPE VARCHAR(15),
            UOM VARCHAR(10),
            REQUIRED VARCHAR(1),
            SELECT_LIST_ID VARCHAR(20),
            VISIBLE VARCHAR(1),
            EXTENSION_ID INT,
            FIELD_WIDTH INT,
            FIELD_MAX_CHARS INT,
             CONSTRAINT APP_QM_DICTIONARY_PK PRIMARY KEY (TEST_ID)
  )</statement>
```

4.2. 移行修正の動作確認

今回のアプリケーション移行では以下の観点で修正確認を行いました。

1. 修正有無に関わらず、ソースコードから実行される SQL は PostgreSQL 上で動作確認をする
   （データベース関連ロジック以外手が入らないため、ソースコードの機能性の確認は重要度を下げ、今回は SQL の動作確認をメインに確認を行った）

そのための確認手段は以下のとおりです。

1. ビデオチュートリアルに従い Commander4j の基本機能が正常実行できることを確認する
2. 上記 1 が完了しても未実行の SQL がある場合、個別に当該機能を実行して PostgreSQL 上で正常終了することを確認する

具体的には以下のとおり、今回の手順を実施しました。

1. DDL 文の確認
   「Commander4j」のビデオチュートリアル「Database Schema」の処理を実施し DDL 文の動作を確認します。
   本ビデオチュートリアルには、以下の内容が含まれています。
   データベース接続設定の作成
   データベースへの接続確認
   データベースへのスキーマ登録

2. DDL の修正
   上記 1 でエラーが発生した場合、該当する DDL 文を修正し、再度上記 1 の作業を実施します。

3. DML 文の確認
   「Commander4j」のビデオチュートリアルの処理を実施し各機能がエラーなく正常に実行できることを確認します。

4. DML 文の修正
   上記 3 でエラーが発生した場合、該当する DML 文を修正し、再度上記 3 の作業を実施します。

5. 未実行 SQL 文の動作確認
   上記 4 までの作業で未実行の SQL 文が存在した場合、個別に当該 SQL が含まれる機能を実行し、修正有無に関わらず SQL 文が PostgreSQL 上で正常に動作することを確認します。

6. トランザクション処理の修正
   MySQL や Oracle と異なり PostgreSQL ではトランザクション内でのエラー発生時、それ以降の同一トランザクション内の SQL がすべてエラーとなります。本問題に対処するため、PostgreSQL 用に例外処理の見直しを行いました。
   詳細は次章にご参照ください。

5. 移行実施結果

5.1. db_syntax_diff 実行結果

db_syntax_diff 実行結果、および実際の修正箇所数は以下のとおりでした。

表 5.1: db_syntax_diff 実行結果

実行対象	SQL 文総数	報告件数	修正箇所数
Java ソース	0	44	0
XML ファイル(環境構築用 SQL)	3115	384	307
XML ファイル(ツール実行時 SQL)	275	0	0

Java ソース対しての db_syntax_diff の報告数は 44 件でいずれも修正の必要はありませんでした。
内容としてはダイアログメッセージ文字列を SQL と誤認識した「DELETE 文の From 句省略」等でした。

誤認識の例

ダイアログメッセージに表示させる文字列を SQL 文と誤認識している。

DELETE 文の From 句省略 例(src/com/commander4j/app/JInternalFrameLocationAdmin.java)L239：
int n = JOptionPane.showConfirmDialog(Common.mainForm, "**Delete** Location " + llocation + " ?", "Confirm", JOptionPane.YES_NO_OPTION);

XML ファイルに対する db_syntax_diff の報告で過剰報告では、「FNC-226-003 TO_TIMESTAMP 関数の引数が 3」の誤検知が報告されていました。

過剰報告の例

TO_TIMESTAMP 関数の引数は実際には 2 つであるが、3 つと誤検知している。

TO_TIMESTAMP 関数の引数が 3 例(xml/schema/org.postgresql.Driver/000001.xml)L2114：
Insert into SYS_INTERFACE_LOG
(EVENT_TIME,INTERFACE_LOG_ID,MESSAGE_REF,INTERFACE_TYPE,MESSAGE_INFORMATION,INTERFACE_DIRECTION,ACTION,MESSAGE_DATE,MESSAGE_STATUS,MESSAGE_ERROR,WORKSTATION_ID) values (**to_timestamp('15-JUL-09 14.02.15.000000000','DD-MON-RR HH24.MI.SS.FF')**,2,'14','Production Declaration','SSCC=350001612200223186','Output','File Write',**to_timestamp('15-JUL-09 14.02.15.000000000','DD-MON-RR HH24.MI.SS.FF')**,'Warning','Active MQ unavailable, recovery file = xml¥interface¥recovery¥0000000014_ProductionDeclaration.xml','sony')

報告はほぼ型に関するもので大半が VARCHAR2 の報告でした。
今回はエラーレベル「WARNING」は修正の必要はありませんでした。

5.2. db_syntax_diff 未報告の事象

db_syntax_diff で検出できなかったものの、実際には修正が必要だった内容を以下に示します。
詳細については「別紙：db_syntax_diff 未検出事象一覧」をご参照ください。

表 5.2: db_syntax_diff 未報告事象

項番	事象
1	テーブル名を大文字小文字を識別する「”」(ダブルクォート)括りで宣言しているが、 問い合わせ時は括りの無しだとテーブルが見つからない。
2	1トランザクション内で SQL エラーが前提の確認処理が実行されるため、後続の SQL がエラー終了する。
3	CREATE OR REPLACE FORCE VIEW 文が実行された。
4	ALTER TABLE テーブル名 MODIFY 文が実行された。
5	「SYSDATE」を含む文が実行された。
6	「ROWNUM」を含む文が実行された。
7	サブクエリに alias(AS XX)が記述されてない。
8	「NOT NULL ENABLE」「ENABLE」を含む文が実行された。
9	DELETE 文で From 句が省略された。

修正箇所数は項番 1 が 583 箇所、項番 2 が 378 箇所と大半を占めていました。

項番 2 は、トランザクション内でエラーが発生した場合、それ以降の SQL が無条件にエラーとなる事象です。
Oracle 等ではトランザクション内でエラーが発生した場合も、後続の SQL は実行する仕様となっており、さらに DDL 文は暗黙の commit が行なわれます。Commander4j では、環境構築時に、トランザクションの先頭で既存テーブルが無いことを確認するため SELECT 文を発行しています。Oracle 等では SELECT 文がエラーとなっても後続の SQL 文が実行されますが、PostgreSQL は以降すべての SQL 文がエラー終了してしまい、環境構築が実行できなくなっていました。そのため、PostgreSQL ではトランザクションでエラーが発生した場合はトランザクションを再構築する必要があります。
（参照 URL　http://www.oss-db.jp/measures/dojo_01.shtml）

5.3. 移行作業の各作業時間割合

今回の移行作業では、作業時間の大半が SQL 文の動作確認 - テストフェーズとなっており、全体の 90%以上を占めています。それは以下の要因によるものといえます。

- テストフェーズでは修正有無に関わらず、PostgreSQL 上で実行されうる SQL は動作確認を行う必要があること
- db_syntax_diff で修正が必要なことを検出できなかった問題については、テストフェーズで修正・動作確認を行う必要があること

今回の移行作業における各フェーズの所要時間比率を以下に示します。
ここに記載した情報は以下を前提にしています。

- 事前にソースコード等の構造は把握していること
- 今回はトライアル移行のため、通常の開発時には必要となるドキュメントの作成や品質確認プロセスはほとんど行っていないこと
- db_syntax_diff の使用方法について、事前に把握していること

なお、PostgreSQL への JDBC 接続確認は数分の作業だったので時間比率には含めていません。

移行対象アプリケーションの規模情報等は表 1.1: 移行対象アプリケーション情報をご参照ください。

表 5.3: 移行作業の所要時間比率

移行作業項番	移行作業項目	所要時間比率
#1	PostgreSQL 用 JDBC ドライバファイルの追加 DBMS 別分岐処理への PostgreSQL 用処理の追加 PostgreSQL 用設定ファイルの追加	2.8%
#2	SQL 文の修正	5.8%
#3	SQL 文の動作確認 # テスト中に発見したエラーの修正時間も含む	91.4%

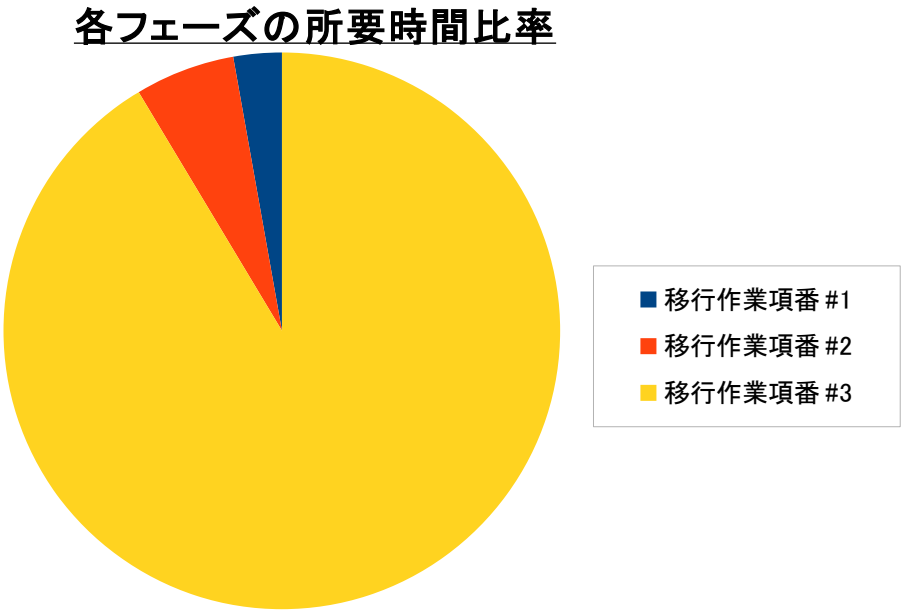

図 5.1: 移行作業の各作業時間割合

6. まとめ

移行対象が複数 DB 対応を考慮した Java アプリケーションだったため、移行の難易度は低かった思われます。
その理由は以下のとおりです。

- Java による JDBC アプリケーションであるため、共通の API で複数 DBMS に対応できた
- 複数 DBMS 対応済みアプリケーションであったため、DBMS 特有機能を使用した SQL を利用していなかった
- 複数 DBMS 対応済みアプリケーションであったため、ストアドプロシージャを利用していなかった
- SQL 文は XML ファイルにまとめられており、ソースでの SQL 構築が少なかったため、SQL 文の検出漏れ、修正時の修正箇所特定の工数が軽減できた
- db_syntax_diff のメッセージで SQL の修正方法に選択の余地がある場合に、MySQL の SQL 文を参考にできた
  例 1
  修正前　NUMBER(*,0)
  修正後　DECIMAL(22,0)

  例 2
  修正前　NUMBER(*,0)
  修正後　INT

移行作業で困難であった点は以下のとおりです。

- ソースから参照されない SQL や、ソースに実行されることのない処理が記述されており、修正した SQL を実行させる方法を特定することが困難であった
- トランザクション処理でのエラーに対する仕様が PostgreSQL と他 DBMS とは異なっていたため、エラー発生時の例外処理をすべて見直す必要があった

db_syntax_diff を使用した感想として以下が挙げられます。

■良い点

- ソースが膨大なため修正箇所を列挙されるだけでも作業工数を減少できる
- 検出漏れより誤検知の方が作業工数は減少できる

■悪い点

- 修正方法が複数ある場合、メッセージだけで修正方法を決定することが難しい

■改善点

- 実行結果が HTML 等で作成され、修正箇所へのリンクが表示されるとより作業が効率的になると思われる
- 修正はほぼ置換可能だが、db_syntax_diff に SQL の自動修正機能を追加する場合は、メッセージ文字列と SQL 文字列の判定が必要である

7. 別紙・付録一覧

- 別紙：db_syntax_diff 未検出事象一覧
- 付録 1：db_syntax_diff 実行結果
- 付録 2：PostgreSQL 対応版 Commander4j ソースコード

著者

版	所属企業・団体名	部署名	氏名
アプリケーション移行実践編 第 1.0 版 （2012 年度 WG#2）	NTT ソフトウェア株式会社		