FDW（外部データラッパ）のアセスメント

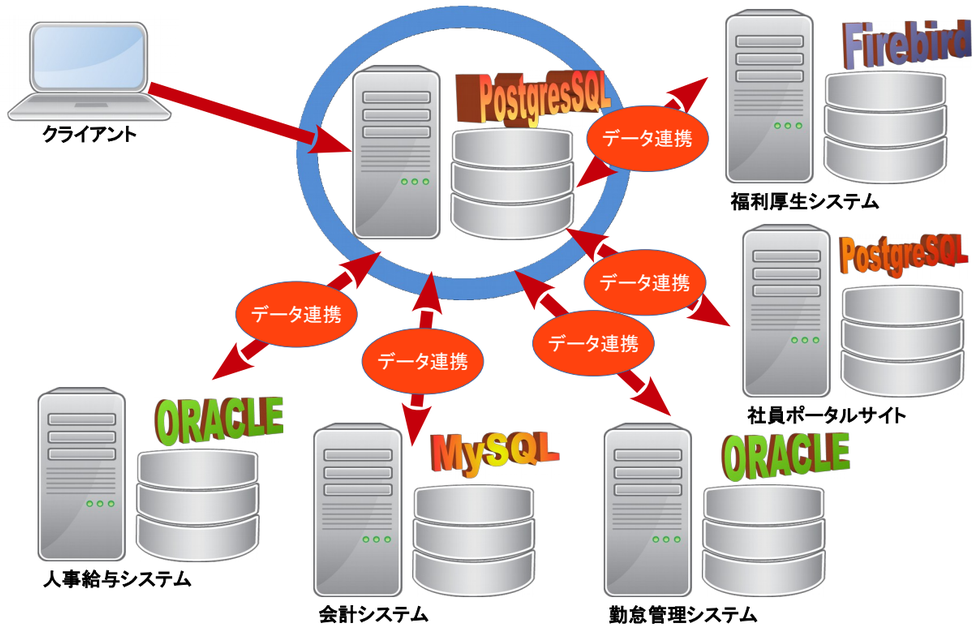
1. 概要

今回のアセスメントの前提として、以下のような状況を設定しています。

・社内システムの現状
　・勤怠管理、人事給与等、業務ごとにシステムが存在し、主管部署が運用している。
　・業務システムの利用者を個別に社員情報として保有している。
　・社員情報の変更は、変更の発生時点で各業務システムの管理者が独自に行っている。
　・業務システムごとに、様々な種類のDBMSを採用している。
・社内システムの課題
　・業務システムごとに、社員情報の変更実施のタイムラグが発生する。
　　例）・他部門から人事部への異動後、直ぐには人事給与システムを利用できない。
　　・退職者の利用権限が残ったまま、利用できる状態にある。
　・社員情報のメンテナンスの運用負荷が高い。
　　例）・所属異動が月3回ある。
・解決策
　・社員情報のメンテナンスを人事部で一元管理する。
・実現方法
　・PostrgreSQLの外部データラッパというデータ連携機能を利用し、
　各業務システムの社員情報をPostgreSQL自体のテーブルとして扱うことで、参照、更新等のメンテナンスを行う。

1.1. 検証環境

業務システム用DBサーバを5つと一元管理用のPostgreSQLサーバを1つ用意します。



1．2．検証データ

各社員は、所属部署やその役職により、以下の表に示した要領で、システムへのログイン可否や利用権限をあたえられていることとします。

・検証データ

部名	課名	ユーザID	ユーザ名	役職	人事給与システム Oracle		会計システム MySQL		勤怠管理システム Oracle		社員ポータルサイト PostgreSQL		福利厚生システム Firebird	
					Login	利用権限	Login	利用権限	Login	利用権限	Login	利用権限	Login	利用権限
経理部 100	―	52030	経理 太郎	部長	○	一般	○	管理者	○	一般	○	一般		
	経理課 10010	55040	経理 一郎	課長	○	一般	○	管理者	○	一般	○	一般		
		58020	経理 二郎	主任	○	一般	○	一般	○	一般	○	一般		
		61050	経理 三郎	社員	○	一般	○	一般	○	一般	○	一般		
	財務課 10020	53150	経理 一男	課長	○	一般	○	管理者	○	一般	○	一般		
		71080	経理 二男	主任	○	一般	○	一般	○	一般	○	一般		
		79010	経理 三男	社員	○	一般	○	一般	○	一般	○	一般		
人事部 200	―	51030	人事 太郎	部長	○	管理者	○	一般	○	管理者	○	一般	○	一般
	人事給与課 20010	52100	人事 一郎	課長	○	管理者	○	一般	○	管理者	○	一般	○	一般
		55090	人事 二郎	主任	○	管理者			○	管理者	○	一般	○	一般
		58060	人事 三郎	社員	○	一般			○	管理者	○	一般	○	一般
	福利厚生課 20020	62010	人事 一男	課長	○	管理者	○	一般	○	管理者	○	一般	○	管理者
		65030	人事 二男	主任	○	管理者	○	一般	○	管理者	○	一般	○	管理者
		76020	人事 三男	社員	×	一般			○	管理者	○	一般	○	管理者
営業部 300	―	52071	営業 太郎	部長					○	一般	○	一般		
	営業1課 30010	61011	営業 一郎	課長					○	一般	○	一般		
		62051	営業 二郎	主任					○	一般	○	一般		
		67071	営業 三郎	社員					○	一般	○	一般		
	営業2課 30020	72011	営業 一男	課長					○	一般	○	一般		
		73021	営業 二男	主任					○	一般	○	一般		
		74031	営業 三男	社員					○	一般	○	一般		
開発部 400	―	67022	開発 太郎	部長					○	一般	○	一般		
	開発1課 40010	73092	開発 一郎	課長					○	一般	○	一般		
		75082	開発 二郎	主任					○	一般	○	一般		
		77142	開発 三郎	社員					○	一般	○	一般		
	開発2課 40020	75222	開発 一男	課長					○	一般	○	一般		
		76892	開発 二男	主任					○	一般	○	一般		
		74322	開発 三男	社員					○	一般	○	一般		
情報 システム部 500	―	58023	情報 太郎	部長					○	システム管理者	○	管理者		
	システム課 50010	63043	情報 一郎	課長					○	システム管理者	○	管理者		
		67183	情報 二郎	社員					○	システム管理者	○	管理者		

1．4．検証シナリオ

検証データの状態から、以下のような異動が発生したこととします。

①人事太郎が休職
②営業三男を経理部財務課に異動
③人事三郎が主任に昇格
④人事二郎が営業部に異動
⑤開発次郎が退職

・検証シナリオ想定結果

部名	課名	ユーザID	ユーザ名	役職	人事給与システム Oracle		会計システム MySQL		勤怠管理システム Oracle		社員ポータルサイト Oracle		福利厚生システム Firebird	
					Login	利用権限	Login	利用権限	Login	利用権限	Login	利用権限	Login	利用権限
経理部 100	―	52030	経理 太郎	部長	○	一般	○	管理者	○	一般	○	一般		
	経理課 10010	55040	経理 一郎	課長	○	一般	○	管理者	○	一般	○	一般		
		58020	経理 二郎	主任	○	一般	○	一般	○	一般	○	一般		
		61050	経理 二郎	社員	○	一般	○	一般	○	一般	○	一般		
	財務課 10020	5315					○	管理者	○	一般	○	一般		
		71080	経理 [illegible]	主任	○	一般	○	一般	○	一般	○	一般		
		74031	営業 三男	社員	○	一般	○	一般	○	一般	○	一般		
		79010	経理 三男	社員	○	一般	○	一般	○	一般	○	一般		
人事部 200	―	51030	人事 太郎	部長	×	管理者	×	一般	×	管理者	×	一般	×	一般
	人事給与課 20010	52100	人事 一郎	課長	○	管理者	○	一般	○	管理者	○	一般	○	一般
		58060	人事 三郎	~~主任~~	○	~~管理者~~			○	管理者	○	一般	○	一般
	福利厚生課 20020	[illegible]	[illegible]	[illegible]	[illegible]	管理者	○	一般	○	管理者	○	一般	○	管理者
		[illegible]	[illegible]	[illegible]	[illegible]	管理者	○	一般	○	管理者	○	一般	○	管理者
		76020	人事 三男	社員	×	一般			○	管理者	○	一般	○	管理者
営業部 300	―	52071	営業 太郎	部長					○	一般	○	一般		
	営業1課 30010	61011	営業 一郎	課長					○	一般	○	一般		
		~~55090~~	~~人事 二郎~~	~~主任~~	~~○~~	~~管理者~~			○	~~一般~~	○	一般	~~○~~	~~一般~~
		6[illegible]							○	一般	○	一般		
		6[illegible]							○	一般	○	一般		
	営業2課 30020	7[illegible]							○	一般	○	一般		
		73021	営業 二男	主任					○	一般	○	一般		
開発部 400	―	67022	開発 太郎	部長					○	一般	○	一般		
	開発1課 40010	73092	開発 一郎	課長					○	一般	○	一般		
		~~75082~~	~~開発 二郎~~	~~主任~~					~~○~~	~~一般~~	~~○~~	~~一般~~		
		7[illegible]		[illegible]員					○	一般	○	一般		
	開発2課 40020	7[illegible]		[illegible]長					○	一般	○	一般		
		76892	開発 二男	主任					○	一般	○	一般		
		74322	開発 三男	社員					○	一般	○	一般		
情報 システム部 500	―	58023	情報 太郎	部長					○	システム管理者	○	管理者		
	システム課 50010	63043	情報 一郎	課長					○	システム管理者	○	管理者		
		67183	情報 二郎	社員					○	システム管理者	○	管理者		

検証シナリオ①
すべてのシステムをログイン不可に

検証シナリオ②
所属変更
人事給与システムと会計システムにユーザ追加

検証シナリオ③
人事給与システムに管理者権限を付与

検証シナリオ④
所属変更
人事給与システム、福利厚生システム登録
勤怠管理システムの利用権限を一般ユーザに

検証シナリオ⑤
レコード削除

・所属部署と業務システムの利用権限対応表

システム	利用権限				
	経理部	人事部	営業部	開発部	情報システム部
人事給与システム	・一般（必須）	・一般（必須） ・管理者（主任以上）	・不可	・不可	・不可
会計システム	・一般（必須） ・管理者（課長以上）	・一般（任意）	・不可	・不可	・不可
勤怠管理システム	・一般（必須）	・管理者（必須）	・一般（必須）	・一般（必須）	・システム管理者（必須）
社員ポータルサイト	・一般（必須）	・一般（必須）	・一般（必須）	・一般（必須）	・管理者（必須）
福利厚生システム	・不可	・一般（必須） ・管理者（福利厚生）	・不可	・不可	・不可

2. 検証作業

概要に示した内容で検証作業を行います。

2.1. 外部データラッパの導入検証

今回導入検証用に入手した以下のFDWの導入から外部テーブルの参照までを検証しました。
※FDWと同名のシートに詳細を記載してあります。

種類	対象	FDW
RDBMS	ORACLE	Oracle_fdw
	MySQL	MySQL_fdw
	PostgreSQL	postgres_fdw
	Firebird	firebird_fdw
	SQLite	SQLite_fdw
非RDBMS	LDAP	LDAP_fdw(python)
	CSV	file_fdw

FDWの入手先：PostgreSQL wiki： https://wiki.postgresql.org/wiki/Foreign_data_wrappers を参照してください。

2.2. 検証シナリオの実施

検証シナリオに従い実施して、1.4の表「検証シナリオ想定結果」の通りになることを検証しました。
※別シート「検証シナリオ①～⑤」及び「検証結果」に実施結果を記載してあります。

2.3. Oracle_fdwの基本動作検証

FDWの基本動作検証の一環で、Oracle_fdwに限定して実施した結果を紹介します。
※別シート「Oracle_fdw動作検証」に詳細を記載してあります。

①検証環境（Oracle）のテーブル定義とデータ

カラム物理名	カラム論理名	制約
user_id	ユーザID	主キー
user_name	ユーザ名	–
password	パスワード	–
user_group	組織	–

user_id	user_name	password	user_group
1	David_Hume	pass_hume	thought
2	Immanuel_Kant	pass_kant	philosophy
3	Gilles_Deleuze	pass_gilles	philosophy
4	Hitoshi_Nagai	pass_hitoshi	philosophy
5	Kitaro_Nishida	pass_kitaro	philosophy

②検証内容

・CRUDの検証

PostgreSQL上の外部テーブルへのCRUD操作を行い、PostgreSQL及びOracle上で操作の結果を確認しました。

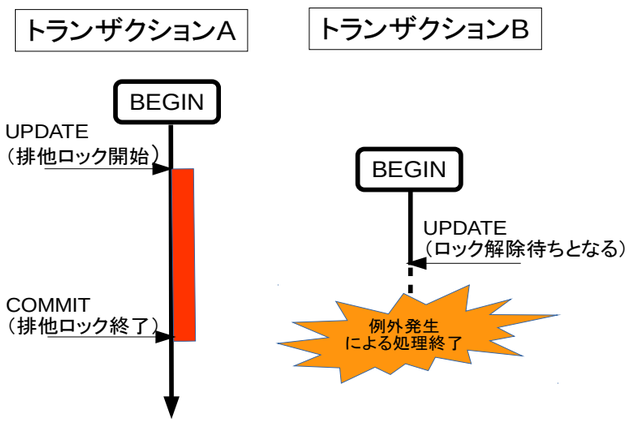
・トランザクション管理の検証

PostgreSQL上への更新処理の確定（COMMIT）、更新処理の破棄（ROLLBACK）操作を行い、PostgreSQL及びOracle上で操作の結果を確認しました。

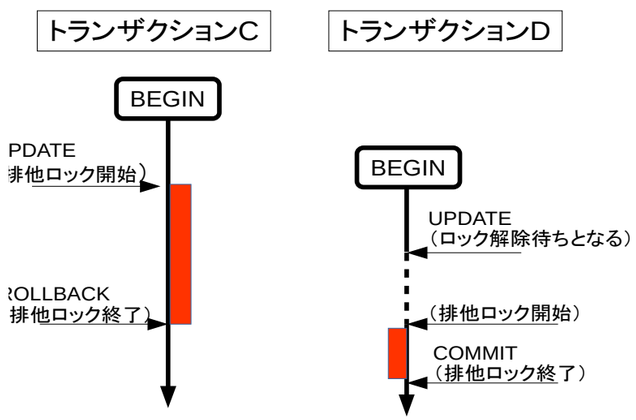
・同時実行制御の検証

oracle_fdwを経由した複数のトランザクションによる、同一データの同時更新時のロック制御を検証しました。
結果として、以下の図のようになることを確認しました。

・先にロックしたトランザクションAをCOMMITした場合



・先にロックしたトランザクションCをROLLBACKした場合



※上記結果は、「oracle_fdw経由のトランザクション分離レベルはすべてSerializableになる」という仕様によるものです。

・ロック解除待ち回避の検証

同時実行制御検証のトランザクションBのように、トランザクションを受け付けた上で、最後に異常終了してしまう挙動は、業務用アプリケーションとしては致命的です。
これを避けるために、SELECT句による明示的ロック（SELECT ~ FOR UPDATE NOWAIT）を行い、トランザクションBの実行が受け付けられない（リソース・ビジーエラー）ようにする方法があります。
そこで、同時実効制御検証のトランザクションB及びトランザクションDにおいて、UPDATEの前に明示的ロックを行いましたが、上記結果と同様にロック解除待ち状態となりました。

・同時実行制御検証結果の考察

複数トランザクションを用いる場合、ロック解除待ちをした上で、直列化失敗によるエラーが発生し、処理が失敗してしまう可能性があります。
同時実行制御検証のトランザクションBのような事態を回避する為に、以下の点に留意する必要があると思われます。

①:同一データを複数トランザクションで同時に更新しないようにする。
②:Oracle定義例外　ORA-08177: can't serialize access for this transaction　が発生した場合、例外が発生したトランザクションをROLLBACKし、再処理する。

以上

No	概要	対象	ユーザ	コマンド	確認（確認コマンド等）
1	Oracleクライアントの入手	PostgreSQLサーバ	root	# cd /usr/local/src	# ls /usr/local/src Oracle-instantclient11.2-basic-11.2.0.1.0-1.x86_64.rpm oracle-instantclient11.2-devel-11.2.0.1.0-1.x86_64.rpm oracle-instantclient11.2-sqlplus-11.2.0.1.0-1.x86_64.rpm （参考） 通常の入手方法 1.OTN（Oralce Technorogy Network）にユーザー登録する 2.ダウンロードサイトからrpmファイルをローカルPCにダウンロードする 3.rpmファイルをPostgreSQLサーバへアップロードする ※ダウンロードサイト（2016/2/16現在） http://www.oracle.com/technetwork/topics/linuxx86-64soft-092277.html ※クライアント接続ライブラリ（sqlplus）は接続確認用であり、FDWに必須ではない。
2	Oracleクライアント基本パッケージのインストール	PostgreSQLサーバ	root	# rpm -ivh oracle-instantclient11.2-basic-11.2.0.1.0-1.x86_64.rpm	# rpm -ivh oracle-instantclient11.2-basic-11.2.0.1.0-1.x86_64.rpm 準備しています... ################################## [100%] 更新中 / インストール中... 1:oracle-instantclient11.2-basic- 11################################## [100%]
3	OracleクライアントSDKパッケージのインストール	PostgreSQLサーバ	root	# rpm -ivh oracle-instantclient11.2-devel-11.2.0.1.0-1.x86_64.rpm	# rpm -ivh oracle-instantclient11.2-devel-11.2.0.1.0-1.x86_64.rpm 準備しています... ################################## [100%] 更新中 / インストール中... 1:oracle-instantclient11.2-devel- 11################################## [100%]
4	OracleクライアントSQL*Plusパッケージのインストール	PostgreSQLサーバ	root	# rpm -ivh oracle-instantclient11.2-sqlplus-11.2.0.1.0-1.x86_64.rpm	# rpm -ivh oracle-instantclient11.2-sqlplus-11.2.0.1.0-1.x86_64.rpm 準備しています... ################################## [100%] 更新中 / インストール中... 1:oracle-instantclient11.2-sqlplus- ################################## [100%]
5	インストール済みOracleクライアントパッケージの確認	PostgreSQLサーバ	root	# pm -qa | grep -i oracle	# rpm -qa | grep -i oracle oracle-instantclient11.2-devel-11.2.0.1.0-1.x86_64 oracle-instantclient11.2-sqlplus-11.2.0.1.0-1.x86_64 oracle-instantclient11.2-basic-11.2.0.1.0-1.x86_64
6	Oracle用環境変数の設定	PostgreSQLサーバ	root	# export LD_LIBRARY_PATH=/usr/lib/oracle/11.2/client64/lib:$LD_LIBRARY_PATH # export PATH=/usr/lib/oracle/11.2/client64/bin:$PATH	# echo $LD_LIBRARY_PATH /usr/lib/oracle/11.2/client64/lib: # echo $PATH /usr/lib/oracle/11.2/client64/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/root/bin:/usr/pgsql-9.4/bin

7	Oracleサーバへの接続	PostgreSQLサーバ	root	# sqlplus [ユーザ]/[パスワード]@[サーバ名]/[データベース名]	# sqlplus system/manager@133.227.221.136:1521/xe SQL*Plus: Release 11.2.0.1.0 Production on Wed Feb 17 00:26:47 2016 Copyright (c) 1982, 2009, Oracle. All rights reserved. Connected to: Oracle Database 11g Express Edition Release 11.2.0.2.0 – 64bit Production SQL>
8	環境設定 (~postgres/.bash_profile)	PostgreSQLサーバ	postgres	$ vi ~postgres/.bash_profile (以下の値を追加) export LD_LIBRARY_PATH=/usr/lib/oracle/11.2/client64/lib: $LD_LIBRARY_PATH	$ source .bash_profile $ echo $LD_LIBRARY_PATH /usr/lib/oracle/11.2/client64/lib:

9	PostgreSQLの再起動	PostgreSQLサーバ	postgres	$ pg_ctl stop –m fast
				
				$ pg_ctl start –w
					$ ps –ef | grep postgres
					Root 4645 2319 0 00:32 pts/0 00:00:00 su – postgres
					postgres 4646 4645 0 00:32 pts/0 00:00:00 –bash
					postgres 4889 1 0 00:40 pts/0 00:00:00 /usr/pgsql–9.4/bin/postgres
					postgres 4890 4889 0 00:40 ? 00:00:00 postgres: logger process
					postgres 4892 4889 0 00:40 ? 00:00:00 postgres: checkpointer process
					postgres 4893 4889 0 00:40 ? 00:00:00 postgres: writer process
					postgres 4894 4889 0 00:40 ? 00:00:00 postgres: wal writer process
					postgres 4895 4889 0 00:40 ? 00:00:00 postgres: autovacuum launcher process
					postgres 4896 4889 0 00:40 ? 00:00:00 postgres: stats collector process
					postgres 4926 4646 0 00:43 pts/0 00:00:00 ps –ef
					postgres 4927 4646 0 00:43 pts/0 00:00:00 grep ––color=auto postgres

10	Oracle_fdwの入手	PostgreSQLサーバ	root	# cd /usr/local/src/ # git clone https://github.com/laurenz/oracle_fdw.git # cd oracle_fdw	$ ls -l /usr/local/src/oracle_fdw 合計 372 -rw-r--r--. 1 root root 10079 2月 17 00:47 CHANGELOG -rw-r--r--. 1 root root 1009 2月 17 00:47 LICENSE -rw-r--r--. 1 root root 1751 2月 17 00:47 Makefile lrwxrwxrwx. 1 root root 17 2月 17 00:47 README -> README.oracle_fdw -rw-r--r--. 1 root root 31878 2月 17 00:47 README.oracle_fdw -rw-r--r--. 1 root root 156 2月 17 00:47 TODO drwxr-xr-x. 2 root root 72 2月 17 00:47 expected -rw-r--r--. 1 root root 231 2月 17 00:47 oracle_fdw--1.0--1.1.sql -rw-r--r--. 1 root root 1003 2月 17 00:47 oracle_fdw--1.1.sql -rw-r--r--. 1 root root 156006 2月 17 00:47 oracle_fdw.c -rw-r--r--. 1 root root 133 2月 17 00:47 oracle_fdw.control -rw-r--r--. 1 root root 7959 2月 17 00:47 oracle_fdw.h -rw-r--r--. 1 root root 44463 2月 17 00:47 oracle_gis.c -rw-r--r--. 1 root root 96260 2月 17 00:47 oracle_utils.c
11	コンパイルの実行	PostgreSQLサーバ	root	# export PATH=${PATH}:/usr/pgsql-9.4/bin # make USE_PGXS=1	エラーが表示されないこと.
12	インストールの実行	PostgreSQLサーバ	root	# make USE_PGXS=1 install	エラーが表示されないこと. # ls /usr/pgsql-9.4/lib/oracle_fdw.so /usr/pgsql-9.4/lib/oracle_fdw.so

13	PostgreSQL エクステンションの作成	PostgreSQLサーバ	postgres	$ psql <DB名>
				
				CREATE EXTENSION oracle_fdw;
					$ psql testdb
					
					testdb=# CREATE EXTENSION oracle_fdw;
					CREATE EXTENSION
					
					testdb=> ¥dx
					インストール済みの拡張の一覧
					名前 | バージョン | スキーマ | 説明
					-------------+-------------+------------+---
					oracle_fdw | 1.1 | public | foreign data wrapper for Oracle access
					plpgsql | 1.0 | pg_catalog | PL/pgSQL procedural language
					(2 行)

14	PostgreSQL 外部サーバの作成	PostgreSQLサーバ	postgres	CREATE SERVER oracle_server FOREIGN DATA WRAPPER oracle_fdw OPTIONS (dbserver '//[サーバ名]/[データベース名]');	testdb=# CREATE SERVER oracle_server testdb-# FOREIGN DATA WRAPPER oracle_fdw testdb-# OPTIONS (dbserver '//133.227.221.136/XE'); CREATE SERVER testdb=> ¥des 外部サーバー一覧 名前 | 所有者 | 外部データラッパー ---+---+--- oracle_server | postgres | oracle_fdw (1 行)
15	PostgreSQL 外部サーバの利用権限	PostgreSQLサーバ	postgres	GRANT USAGE ON FOREIGN SERVER oracle_server TO [ユーザ名];	testdb=# GRANT USAGE ON FOREIGN SERVER oracle_server TO jip; GRANT testdb=# ¥c - jip データベース "testdb" にユーザ"jip"として接続しました。
16	PostgreSQLユーザーとの関連付け	PostgreSQLサーバ	postgres	CREATE USER MAPPING FOR ［ユーザ名］ SERVER oracle_server OPTIONS (user '<ORACLEユーザ>', password '<ORACLEパスワード>');	testdb=# CREATE USER MAPPING FOR jip testdb-# SERVER oracle_server testdb-# OPTIONS (user 'scott', password 'tiger'); CREATE USER MAPPING
17	PostgreSQL 外部表の作成	PostgreSQLサーバ	postgres	CREATE FOREIGN TABLE foreign_oracle_user_tbl (no int, id text, pass text, level int) SERVER oracle_server OPTIONS (schema '<スキーマ所有者>', table '<テーブル名>');	testdb=> CREATE FOREIGN TABLE foreign_oracle_user_tbl testdb-> (testdb(> no int, testdb(> id text, testdb(> pass text, testdb(> level int testdb(>) testdb-> SERVER oracle_server testdb-> OPTIONS (schema 'JIP', table 'USER_TBL'); CREATE FOREIGN TABLE
18	PostgreSQL 外部表の検索	PostgreSQLサーバ	postgres	select * from foreign_oracle_user_tbl;	testdb=> select * from foreign_oracle_user_tbl; no | id | pass | level ---+---+---+--- 1 | O_user1 | O_pass1 | 3 1 | O_user2 | O_pass1 | 2 1 | O_user3 | O_pass1 | 2 1 | O_user4 | O_pass1 | 1 1 | O_user5 | O_pass1 | 1 (5 行)

No	概要	対象	ユーザ	コマンド	確認（確認コマンド等）
1	MySQLライブラリの入手	PostgreSQLサーバ	root	# cd /usr/local/src # wget http://ftp.jaist.ac.jp/pub/mysql/Downloads/MySQL-5.6/MySQL-shared-5.6.28-1.el7.x86_64.rpm # wget http://ftp.jaist.ac.jp/pub/mysql/Downloads/MySQL-5.6/MySQL-devel-5.6.28-1.el7.x86_64.rpm # wget http://ftp.jaist.ac.jp/pub/mysql/Downloads/MySQL-5.6/MySQL-client-5.6.28-1.el7.x86_64.rpm	# ls /usr/local/src/MySQL* /usr/local/src/MySQL-client-5.6.28-1.el7.x86_64.rpm /usr/local/src/MySQL-devel-5.6.28-1.el7.x86_64.rpm /usr/local/src/MySQL-shared-5.6.28-1.el7.x86_64.rpm ※クライアント接続ライブラリ（MySQL-client）は接続確認用であり、FDWに必須ではない。
2	MySQL-shredライブラリのインストール	PostgreSQLサーバ	root	# rpm -Uvh MySQL-shared-5.6.28-1.el7.x86_64.rpm	# rpm -Uvh MySQL-shared-5.6.28-1.el7.x86_64.rpm 警告: MySQL-shared-5.6.28-1.el7.x86_64.rpm: ヘッダー V3 DSA/SHA1 Signature、鍵 ID 5072e1f5: NOKEY 準備しています... ################################## [100%] 更新中 / インストール中... 1:MySQL-shared-5.6.28-1.el7 ################################## [100%]
3	MySQL-devel-ライブラリのインストール	PostgreSQLサーバ	root	# rpm -Uvh MySQL-devel-5.6.28-1.el7.x86_64.rpm	# rpm -Uvh MySQL-devel-5.6.28-1.el7.x86_64.rpm 警告: MySQL-devel-5.6.28-1.el7.x86_64.rpm: ヘッダー V3 DSA/SHA1 Signature、鍵 ID 5072e1f5: NOKEY 準備しています... ################################## [100%] 更新中 / インストール中... 1:MySQL-devel-5.6.28-1.el7 ################################## [100%]
4	MySQL-client-ライブラリのインストール	PostgreSQLサーバ	root	# rpm -Uvh MySQL-client-5.6.28-1.el7.x86_64.rpm	# rpm -Uvh MySQL-client-5.6.28-1.el7.x86_64.rpm 警告: MySQL-client-5.6.28-1.el7.x86_64.rpm: ヘッダー V3 DSA/SHA1 Signature、鍵 ID 5072e1f5: NOKEY 準備しています... ################################## [100%] 更新中 / インストール中... 1:MySQL-client-5.6.28-1.el7 ################################## [100%]
5	インストール済みMySQLライブラリの確認	PostgreSQLサーバ	root	# rpm -qa | grep -i mysql	# rpm -qa | grep -i mysql MySQL-devel-5.6.28-1.el7.x86_64 MySQL-client-5.6.28-1.el7.x86_64 MySQL-shared-5.6.28-1.el7.x86_64

6	MySQLサーバへの接続	PostgreSQLサーバ	root	# mysql -u [ユーザ] -h [サーバ名] -p[パスワード] [データベース名]	# mysql -u pguser -h 133.227.221.136 -ppguser mysqldb Warning: Using a password on the command line interface can be insecure. Reading table information for completion of table and column names You can turn off this feature to get a quicker startup with -A Welcome to the MySQL monitor. Commands end with ; or ¥g. Your MySQL connection id is 8 Server version: 5.6.28 MySQL Community Server (GPL) Copyright (c) 2000, 2015, Oracle and/or its affiliates. All rights reserved. Oracle is a registered trademark of Oracle Corporation and/or its affiliates. Other names may be trademarks of their respective owners. Type 'help;' or '¥h' for help. Type '¥c' to clear the current input statement. Mysql>
7	MySQL_fdwの入手	PostgreSQLサーバ	root	# cd /usr/local/src/ # git clone https://github.com/EnterpriseDB/mysql_fdw.git # cd mysql_fdw	# ls /usr/local/src/mysql_fdw CONTRIBUTING.md README.md deparse.o mysql_fdw.control mysql_init.sh option.c LICENSE connection.c expected mysql_fdw.h mysql_query.c option.o META.json connection.o mysql_fdw--1.0.sql mysql_fdw.o mysql_query.h sql Makefile deparse.c mysql_fdw.c mysql_fdw.so mysql_query.o [root@gp-platform03 mysql_fdw]#
8	コンパイルの実行	PostgreSQLサーバ	root	# export PATH=${PATH}:/usr/pgsql-9.4/bin # make USE_PGXS=1	エラーが表示されないこと.
9	インストールの実行	PostgreSQLサーバ	root	# make USE_PGXS=1 install	エラーが表示されないこと. # ls /usr/pgsql-9.4/lib/mysql_fdw.so /usr/pgsql-9.4/lib/mysql_fdw.so

10	PostgreSQL エクステンションの作成	PostgreSQLサーバ	postgres	CREATE EXTENSION mysql_fdw;	$ psql testdb
					
					testdb=# CREATE EXTENSION mysql_fdw;
					CREATE EXTENSION
					
					testdb=# ¥dx
					インストール済みの拡張の一覧
					名前 | バージョン | スキーマ | 説明
					-----------+-----------+-----------
					+---
					mysql_fdw | 1.0 | public | Foreign data wrapper for querying a MySQL server
					oracle_fdw | 1.1 | public | foreign data wrapper for Oracle access
					plpgsql | 1.0 | pg_catalog | PL/pgSQL procedural language
					(3 行)

11	PostgreSQL 外部サーバの作成	PostgreSQLサーバ	postgres	CREATE SERVER mysql_server FOREIGN DATA WRAPPER mysql_fdw OPTIONS (host '[サーバ名]', port '[ポート番号]');	testdb=# CREATE SERVER mysql_server testdb-# FOREIGN DATA WRAPPER mysql_fdw testdb-# OPTIONS (host '133.227.221.136', port '3306'); CREATE SERVER testdb=# ¥des 外部サーバー一覧 名前 | 所有者 | 外部データラッパー ---------------+----------+--------------------- mysql_server | postgres | mysql_fdw oracle_server | postgres | oracle_fdw (2 行)
12	PostgreSQL 外部サーバの利用権限	PostgreSQLサーバ	postgres	GRANT USAGE ON FOREIGN SERVER mysql_server TO [ユーザ名];	testdb=# GRANT USAGE ON FOREIGN SERVER mysql_server TO jip; GRANT testdb=# ¥c - jip データベース "testdb" にユーザ"jip"として接続しました。
13	PostgreSQLユーザーの関連付け	PostgreSQLサーバ	postgres	CREATE USER MAPPING FOR ［ユーザ名］ SERVER mysql_server OPTIONS (username '<MySQLユーザ>', password '<MySQLパスワード>');	testdb=> CREATE USER MAPPING FOR jip testdb-> SERVER mysql_server testdb-> OPTIONS (username 'pguser', password 'pguser'); CREATE USER MAPPING

14	PostgreSQL 外部表の作成	PostgreSQLサーバ	postgres	CREATE FOREIGN TABLE foreign_mysql_user_tbl (no int, id text, pass text, level int) SERVER mysql_server OPTIONS (dbname '<データベース名>', table_name '<テーブル名>');	testdb=> CREATE FOREIGN TABLE foreign_mysql_user_tbl testdb-> (testdb(> no int, testdb(> id text, testdb(> pass text, testdb(> level int testdb(>) testdb-> SERVER mysql_server testdb-> OPTIONS (dbname 'mysqldb', table_name 'user_tbl'); CREATE FOREIGN TABLE
15	PostgreSQL 外部表の検索	PostgreSQLサーバ	postgres	select * from foreign_mysql_user_tbl;	testdb=> select * from foreign_mysql_user_tbl; INFO: Successfully connected to MySQL database mysqldb at server 133.227.221.136 via TCP/IP with cipher <none> (server version: 5.6.28, protocol version: 10) no | id | pass | level ----+---------+----------+------- 1 | M_user1 | M_pass1 | 3 1 | M_user2 | M_pass1 | 2 1 | M_user3 | M_pass1 | 2 1 | M_user4 | M_pass1 | 1 1 | M_user5 | M_pass1 | 1 (5 行)

No	概要	対象	ユーザ	コマンド	確認（確認コマンド等）

1	PostgreSQL エクステンションの作成	PostgreSQLサーバ	postgres	CREATE EXTENSION postgres_fdw;	$ psql testdb
					
					testdb=# CREATE EXTENSION postgres_fdw;
					CREATE EXTENSION
					
					testdb=# ¥dx
					インストール済みの拡張の一覧
					名前 ｜バージョン｜ スキーマ ｜ 説明
					---+---+---+---
					file_fdw ｜ 1.0 ｜ public ｜ foreign-data wrapper for flat file access
					firebird_fdw ｜ 0.2.5 ｜ public ｜ foreign data wrapper for Firebird
					multicorn ｜ 1.3.1 ｜ public ｜ Multicorn Python bindings for Postgres 9.2.* Foreign Data Wrapper
					mysql_fdw ｜ 1.0 ｜ public ｜ Foreign data wrapper for querying a MySQL server
					oracle_fdw ｜ 1.1 ｜ public ｜ foreign data wrapper for Oracle access
					plpgsql ｜ 1.0 ｜ pg_catalog ｜ PL/pgSQL procedural language
					postgres_fdw ｜ 1.0 ｜ public ｜ foreign-data wrapper for remote PostgreSQL servers
					sqlite_fdw ｜ 0.0.1 ｜ public ｜ SQLite Foreign Data Wrapper
					(8 行)

2	PostgreSQL 外部サーバの作成	PostgreSQLサーバ	postgres	CREATE SERVER postgres_server FOREIGN DATA WRAPPER postgres_fdw OPTIONS (host '[サーバ名]' , port '[ポート番号]' , dbname '[サーバ名]');	testdb=# CREATE SERVER postgres_server testdb-# FOREIGN DATA WRAPPER postgres_fdw testdb-# OPTIONS(host'133.227.221.136',port'5432',dbname'testdb'); CREATE SERVER testdb=# ¥des 外部サーバー一覧 名前 ｜ 所有者 ｜外部データラッパー ---+---+--- csv_server ｜ postgres ｜ file_fdw firebird_server ｜ postgres ｜ firebird ldap_server ｜ postgres ｜ multicorn mysql_server ｜ postgres ｜ mysql_fdw oracle_server ｜ postgres ｜ oracle_fdw postgres_server ｜ postgres ｜ postgres_fdw sqlite_server ｜ postgres ｜ sqlite_fdw (7 行)
3	PostgreSQL 外部サーバの利用権限	PostgreSQLサーバ	postgres	GRANT USAGE ON FOREIGN SERVER postgres_server TO [ユーザ名];	testdb=# GRANT USAGE ON FOREIGN SERVER postgres_server TO jip; GRANT testdb=# ¥c - jip データベース "testdb" にユーザ"jip"として接続しました。
4	PostgreSQLユーザーの関連付け	PostgreSQLサーバ	postgres	CREATE USER MAPPING FOR ［ユーザ名］ SERVER postgres_server OPTIONS (user '<Postgresユーザ>', password '<Postgresパスワード>');	testdb=> CREATE USER MAPPING FOR jip SERVER postgres_server OPTIONS(user'postgres',password'postgres'); CREATE USER MAPPING

5	PostgreSQL 外部表の作成	PostgreSQLサーバ	postgres	CREATE FOREIGN TABLE postgres_user_tbl (　uidnumber　　int, 　uid　　text, 　cn　　text, 　gidnumber　　int, 　Usernumber　text, 　loginlevel　int, 　uselevel　　int) 　SERVER postgres_server 　OPTIONS (schema_name'<スキーマ所有者>',table_name'<テーブル名>');	testdb=> CREATE FOREIGN TABLE postgres_user_tbl testdb-> (testdb(> uidnumber int, testdb(> uid text, testdb(> cn text, testdb(> gidnumber int, testdb(> usernumber text, testdb(> loginlevel text, testdb(> uselevel text testdb(>) testdb-> SERVER postgres_server testdb-> OPTIONS(schema_name'public',table_name'portal_tbl'); CREATE FOREIGN TABLE
6	PostgreSQL 外部表の検索	PostgreSQLサーバ	postgres	select * from postgres_user_tbl;	testdb=> select * from postgres_user_tbl; 　uidnumber	uid	cn	gidnumber	usernumber	loginlev el	uselevel -----------+----------------+----------------+-----------+------------- +--------- ---+---------- 　　52030	taro.keiri	Taro Keiri	100	pass	1 	0 　　55040	ichiro.keiri	Ichiro Keiri	10010	pass	1 	0 (中略) 　　58020	jiro.keiri	Jiro Keiri	10010	pass	1 	0 　　55090	jiro.jinji	Jiro Jinji	30010	pass	1 	0 　　75082	jiro.kaihatu	Jiro Kaihatu	40010	pass	1 	0 (31 行)

No	概要	対象	ユーザ	コマンド	確認（確認コマンド等）
1	firebird-libfbclientのインストール	PostgreSQLサーバ	root	# yum -y install firebird-libfbclient	
2	firebird--develのインストール	PostgreSQLサーバ	root	# yum -y install firebird-devel	# rpm -qa | grep -i firebird firebird-libfbclient-2.5.5.26952.0-2.el7.x86_64 firebird-devel-2.5.5.26952.0-2.el7.x86_64 firebird-libfbembed-2.5.5.26952.0-2.el7.x86_64
3	libfqの入手	PostgreSQLサーバ	root	# cd /usr/local/src/ # git clone https://github.com/ibarwick/libfq.git # cd libfq	
4	libfqのコンパイル＆インストール	PostgreSQLサーバ	root	./configure make make install	# ls /usr/local/lib libfq.a libfq.la libfq.so libfq.so.0 libfq.so.0.1.4 libyajl.so libyajl.so.2 libyajl.so.2.0.1 libyajl_s.a
5	環境設定 (~postgres/.bash_profile)	PostgreSQLサーバ	postgres	$ vi ~postgres/.bash_profile (以下の値を追加) export LD_LIBRARY_PATH=/usr/local/lib:$LD_LIBRARY_PATH	$ source .bash_profile $ echo $LD_LIBRARY_PATH /usr/local/lib:

6	PostgreSQLの再起動	PostgreSQLサーバ	postgres	$ pg_ctl stop -m fast
				
				$ pg_ctl start -w
					$ ps -ef | grep postgres
					Root 4645 2319 0 00:32 pts/0 00:00:00 su - postgres
					postgres 4646 4645 0 00:32 pts/0 00:00:00 -bash
					postgres 4889 1 0 00:40 pts/0 00:00:00 /usr/pgsql-9.4/bin/postgres
					postgres 4890 4889 0 00:40 ? 00:00:00 postgres: logger process
					postgres 4892 4889 0 00:40 ? 00:00:00 postgres: checkpointer process
					postgres 4893 4889 0 00:40 ? 00:00:00 postgres: writer process
					postgres 4894 4889 0 00:40 ? 00:00:00 postgres: wal writer process
					postgres 4895 4889 0 00:40 ? 00:00:00 postgres: autovacuum launcher process
					postgres 4896 4889 0 00:40 ? 00:00:00 postgres: stats collector process
					postgres 4926 4646 0 00:43 pts/0 00:00:00 ps -ef
					postgres 4927 4646 0 00:43 pts/0 00:00:00 grep --color=auto postgres

7	Oracle_fdwの入手	PostgreSQLサーバ	root	# cd /usr/local/src/ # git clone https://github.com/ibarwick/firebird_fdw.git # cd firebird_fdw	# ls /usr/local/src/firebird_fdw BUGS CHANGELOG License META.json Makefile README.md firebird_fdw.control firebird_fdw.so packaging sql src test
8	コンパイルの実行	PostgreSQLサーバ	root	# export PATH=${PATH}:/usr/pgsql-9.4/bin # make	エラーが表示されないこと.
9	インストールの実行	PostgreSQLサーバ	root	# make install	エラーが表示されないこと. # ls /usr/pgsql-9.4/lib/firebird_fdw.so /usr/pgsql-9.4/lib/firebird_fdw.so

10	PostgreSQL エクステンションの作成	PostgreSQLサーバ	postgres	$ psql <DB名>
				
				CREATE EXTENSION firebird_fdw;
					$ psql testdb
					
					testdb=# CREATE EXTENSION firebird_fdw;
					CREATE EXTENSION
					
					Testdb=# ¥dx
					インストール済みの拡張の一覧
					名前 | バージョン | スキーマ | 説明
					-------------+------------+------------
					+---
					---------- 	
					file_fdw | 1.0 | public | foreign-data wrapper for flat file
					access
					firebird_fdw | 0.2.5 | public | foreign data wrapper for Firebird
					multicorn | 1.3.1 | public | Multicorn Python bindings for
					Postgres 9.2.* Foreign Data Wrapper
					mysql_fdw | 1.0 | public | Foreign data wrapper for querying
					a MySQL server
					oracle_fdw | 1.1 | public | foreign data wrapper for Oracle
					access
					plpgsql | 1.0 | pg_catalog | PL/pgSQL procedural language
					postgres_fdw | 1.0 | public | foreign-data wrapper for remote
					PostgreSQL servers
					sqlite_fdw | 0.0.1 | public | SQLite Foreign Data Wrapper
					(8 行)

11	PostgreSQL 外部ラッパーの設定	PostgreSQLサーバ	postgres	CREATE FOREIGN DATA WRAPPER firebird HANDLER firebird_fdw_handler VALIDATOR firebird_fdw_validator;	testdb=# CREATE FOREIGN DATA WRAPPER firebird testdb-# HANDLER firebird_fdw_handler testdb-# VALIDATOR firebird_fdw_validator; CREATE FOREIGN DATA WRAPPER
12	PostgreSQL 外部サーバの作成	PostgreSQLサーバ	postgres	CREATE SERVER firebird_server FOREIGN DATA WRAPPER firebird OPTIONS (address '[サーバ名]', database '[データベース名]');	testdb=# CREATE SERVER firebird_server testdb-# FOREIGN DATA WRAPPER firebird testdb-# OPTIONS (testdb(# address '133.227.221.136', testdb(# database '/data/firebird/firedb.fdb' testdb(#); CREATE SERVER
13	PostgreSQLユーザーとの関連付け	PostgreSQLサーバ	postgres	CREATE USER MAPPING FOR ［ユーザ名］ SERVER oracle_server OPTIONS (username '<ユーザ>', password '<パスワード>');	testdb=# CREATE USER MAPPING FOR CURRENT_USER SERVER firebird_server testdb-# OPTIONS(username 'jip', password 'jip'); CREATE USER MAPPING
14	PostgreSQL 外部表の作成	PostgreSQLサーバ	postgres	CREATE FOREIGN TABLE fb_test(usercd integer, usernm varchar(100), bumoncd integer, loginflg char(1), levelno char(1)) SERVER firebird_server OPTIONS(table_name '<テーブル名>');	testdb=# CREATE FOREIGN TABLE fb_test(testdb(# usercd integer, testdb(# usernm varchar(100), testdb(# bumoncd integer, testdb(# loginflg char(1), testdb(# levelno char(1) testdb(#) testdb-# SERVER firebird_server testdb-# OPTIONS(testdb(# table_name 'usertbl' testdb(#); CREATE FOREIGN TABLE

15	PostgreSQL 外部表の検索	PostgreSQLサーバ	postgres	select * from fb_test;	testdb=# select * from fb_test; usercd | usernm | bumoncd | loginflg | levelno ---+---+---+---+--- 51030 | Taro Jinji | 200 | 1 | 0 52100 | Ichiro Jinji | 20010 | 1 | 0 55090 | jiro Jinji | 20010 | 1 | 0 58060 | saburo Jinji | 20010 | 1 | 0 62010 | kazuo Jinji | 20020 | 1 | 1 65030 | fusao Jinji | 20020 | 1 | 1 76020 | mituo Jinji | 20020 | 1 | 1 (7 行)

No	概要	対象	ユーザ	コマンド	確認（確認コマンド等）
1	SQLiteライブラリのインストール	PostgreSQLサーバ	root	# yum -y install sqlite-devel	# yum -y install sqlite-devel 読み込んだプラグイン:fastestmirror, langpacks Loading mirror speeds from cached hostfile ＊ base: centos.usonyx.net ＊ epel: epel.mirror.srv.co.ge ＊ extras: centos.usonyx.net ＊ nux-dextop: li.nux.ro ＊ updates: centos.usonyx.net 依存性の解決をしています --> トランザクションの確認を実行しています。 ---> パッケージ sqlite-devel.x86_64 0:3.7.17-8.el7 を インストール --> 依存性の処理をしています: sqlite = 3.7.17-8.el7 のパッケージ: sqlite-devel-3.7.17-8.el7.x86_64 --> トランザクションの確認を実行しています。 ---> パッケージ sqlite.x86_64 0:3.7.17-4.el7 を 更新 ---> パッケージ sqlite.x86_64 0:3.7.17-8.el7 を アップデート --> 依存性解決を終了しました。 依存性を解決しました
2	インストール済みSQLiteライブラリの確認	PostgreSQLサーバ	root	# yum list installed | grep sqlite-devel	# yum list installed | grep sqlite-devel sqlite-devel.x86_64 3.7.17-8.el7 @base
3	SQLite_fdwの入手	PostgreSQLサーバ	root	# cd /usr/local/src/ # git clone https://github.com/gleu/sqlite_fdw.git # cd sqlite_fdw	# ls /usr/local/src/sqlite_fdw License README.md doc sqlite_fdw.control src Makefile THANKS.md sql sqlite_fdw.so
4	コンパイルの実行	PostgreSQLサーバ	root	# export PATH=${PATH}:/usr/pgsql-9.4/bin # make	エラーが表示されないこと.
5	インストールの実行	PostgreSQLサーバ	root	# make install	エラーが表示されないこと. # ls /usr/pgsql-9.4/lib/sqlite_fdw.so /usr/pgsql-9.4/lib/sqlite_fdw.so

6	PostgreSQL エクステンションの作成	PostgreSQLサーバ	postgres	CREATE EXTENSION sqlite_fdw;	$ psql testdb
					
					testdb=# CREATE EXTENSION sqlite_fdw;
					CREATE EXTENSION
					
					testdb=# ¥dx
					インストール済みの拡張の一覧
					名前 | バージョン | スキーマ | 説明
					-----------+-----------+-----------
					+-----------
					multicorn | 1.3.1 | public | Multicorn Python bindings for Postgres 9.2.* Foreign Data Wrapper
					mysql_fdw | 1.0 | public | Foreign data wrapper for querying a MySQL server
					oracle_fdw | 1.1 | public | foreign data wrapper for Oracle access
					plpgsql | 1.0 | pg_catalog | PL/pgSQL procedural language
					sqlite_fdw | 0.0.1 | public | SQLite Foreign Data Wrapper
					(5 行)

7	PostgreSQL 外部サーバの作成	PostgreSQLサーバ	postgres	CREATE SERVER sqlite_server
				  FOREIGN DATA WRAPPER sqlite_fdw
				  OPTIONS (database [リモートファイル名]'');
				
					testdb=# CREATE SERVER sqlite_server
					testdb-# FOREIGN DATA WRAPPER sqlite_fdw
					testdb-# OPTIONS (database '/mnt/data/litedb');
					CREATE SERVER
					
					testdb=# ¥des
					外部サーバー一覧
					名前 ｜ 所有者 ｜外部データラッパー
					―――――――――――+――――――――+――――――――――――――――
					ldap_server ｜postgres｜multicorn
					mysql_server ｜postgres｜mysql_fdw
					oracle_server｜postgres｜oracle_fdw
					sqlite_server｜postgres｜sqlite_fdw
					(4 行)
					
					※NFSによるリモートサーバのファイル共有が設定済み
					・/mnt/data

8	PostgreSQL 外部表の作成	PostgreSQLサーバ	postgres	CREATE FOREIGN TABLE sqlite_user_tbl
				(
				  userid int,
				  name varchar(100),
				  bumon int,
				  login char(1),
				  level char(1)
				)
				  SERVER sqlite_server
				  OPTIONS (table 'usertbl');
				
					testdb=# CREATE FOREIGN TABLE sqlite_user_tbl
					testdb-# (
					testdb(# userid int,
					testdb(# name varchar(100),
					testdb(# bumon int,
					testdb(# login char(1),
					testdb(# level char(1)
					testdb(#)
					testdb-# SERVER sqlite_server
					testdb-# OPTIONS (table 'usertbl');
					CREATE FOREIGN TABLE

9	PostgreSQL 外部表の検索	PostgreSQLサーバ	postgres	select * from sqlite_user_tbl; 	testdb=# select * from sqlite_user_tbl; WARNING: The sqlite3 database has not been analyzed. HINT: Run ANALYZE on table ″usertbl″, database ″/mnt/data/litedb″. userid｜ name ｜bumon｜login｜level ―――――――+――――――――+―――――+―――――+――――― 51030｜人事 太郎｜ 200｜1 ｜0 52100｜人事 一郎｜20010｜1 ｜0 55090｜人事 二郎｜20010｜1 ｜0 58060｜人事 三郎｜20010｜1 ｜0 62010｜人事 一男｜20020｜1 ｜1 65030｜人事 二男｜20020｜1 ｜1 76020｜人事 三男｜20020｜1 ｜1 (7 行

No	概要	対象	ユーザ	コマンド	確認（確認コマンド等）
1	python-develのインストール	PostgreSQLサーバ	root	# yum -y install python-devel	※Python 2.7.5が事前にインストール済み # yum -y install python-devel 読み込んだプラグイン:fastestmirror, langpacks Loading mirror speeds from cached hostfile ＊base: centos.usonyx.net ＊epel: epel.mirror.net.in ＊extras: ftp.iij.ad.jp ＊nux-dextop: li.nux.ro ＊updates: ftp.iij.ad.jp 依存性の解決をしています --> トランザクションの確認を実行しています。 ---> パッケージ python-devel.x86_64 0:2.7.5-34.el7 を インストール --> 依存性の処理をしています: python(x86-64) = 2.7.5-34.el7 のパッケージ: python-devel-2.7.5-34.el7.x86_64 --> トランザクションの確認を実行しています。 ---> パッケージ python.x86_64 0:2.7.5-16.el7 を 更新 ---> パッケージ python.x86_64 0:2.7.5-34.el7 を アップデート --> 依存性の処理をしています: python-libs(x86-64) = 2.7.5-34.el7 のパッケージ: python-2.7.5-34.el7.x86_64 --> トランザクションの確認を実行しています。 ---> パッケージ python-libs.x86_64 0:2.7.5-16.el7 を 更新 ---> パッケージ python-libs.x86_64 0:2.7.5-34.el7 を アップデート --> 依存性解決を終了しました。 依存性を解決しました ==
2	python-pipのインストール	PostgreSQLサーバ	root	# yum -y install python-pip	# yum -y install python-pip 読み込んだプラグイン:fastestmirror, langpacks Loading mirror speeds from cached hostfile ＊base: centos.usonyx.net ＊epel: epel.mirror.net.in ＊extras: ftp.iij.ad.jp ＊nux-dextop: li.nux.ro ＊updates: ftp.iij.ad.jp 依存性の解決をしています --> トランザクションの確認を実行しています。 ---> パッケージ python-pip.noarch 0:7.1.0-1.el7 を インストール --> 依存性解決を終了しました。
3	pgxnclientのインストール	PostgreSQLサーバ	root	# pip install pgxnclient	# pip install pgxnclient /usr/lib/python2.7/site-packages/pip/_vendor/requests/packages/urllib3/util/ssl_.py:90: InsecurePlatformWarning: A true SSLContext object is not available. This prevents urllib3 from configuring SSL appropriately and may cause certain SSL connections to fail. For more information, see https://urllib3.readthedocs.org/en/latest/security.html#insecureplatformwarning. InsecurePlatformWarning You are using pip version 7.1.0, however version 8.0.2 is available.

4	Python ldap3 libraryのインストール	PostgreSQLサーバ	root	# pip install ldap3	# pip install ldap3
					You are using pip version 7.1.0, however version 8.0.2 is available.
					You should consider upgrading via the 'pip install --upgrade pip' command.
					Collecting ldap3
					  Downloading ldap3-1.0.4-py2.py3-none-any.whl (327kB)
					  100% |????????????????????????????????| 327kB 890kB/s
					Collecting pyasn1>=0.1.8 (from ldap3)
					  Downloading pyasn1-0.1.9-py2.py3-none-any.whl

5	Python 汎用FDW（multicorn）のインストール	PostgreSQLサーバ	root	# export PATH=${PATH}:/usr/pgsql-9.4/bin # pgxn install multicorn	# pgxn install multicorn INFO: best version: multicorn 1.3.1 INFO: saving /tmp/tmpU5HGqJ/multicorn-1.3.1.zip INFO: unpacking: /tmp/tmpU5HGqJ/multicorn-1.3.1.zip INFO: building extension Python version is 2.7 [-d sql] || mkdir sql [-d src] || mkdir src touch directories.stamp gcc -Wall -Wmissing-prototypes -Wpointer-arith -Wdeclaration-after-statement

6	PostgreSQL エクステンションの作成	PostgreSQLサーバ	postgres	$ psql <DB名>
				
				CREATE EXTENSION multicorn;
					$ psql testdb
					
					testdb=# CREATE EXTENSION multicorn;
					CREATE EXTENSION
					
					testdb-# ¥dx
					インストール済みの拡張の一覧
					名前 | バージョン | スキーマ | 説明
					-----------+-----------+-----------
					+---
					multicorn | 1.3.1 | public | Multicorn Python bindings for Postgres 9.2.* Foreign Data Wrapper
					mysql_fdw | 1.0 | public | Foreign data wrapper for querying a MySQL server
					oracle_fdw | 1.1 | public | foreign data wrapper for Oracle access
					plpgsql | 1.0 | pg_catalog | PL/pgSQL procedural language
					(4 行)

7	PostgreSQL 外部サーバの作成	PostgreSQLサーバ	postgres	CREATE SERVER ldap_server foreign data wrapper multicorn options (wrapper 'multicorn.ldapfdw.LdapFdw');	testdb=# CREATE SERVER ldap_server foreign data wrapper multicorn options (testdb(# wrapper 'multicorn.ldapfdw.LdapFdw' testdb(#); CREATE SERVER testdb=# ¥des 外部サーバー一覧 名前 | 所有者 | 外部データラッパー -------------+----------+--------------------- ldap_server | postgres | multicorn mysql_server | postgres | mysql_fdw oracle_server | postgres | oracle_fdw (3 行)
8	PostgreSQL 外部表の作成	PostgreSQLサーバ	postgres	CREATE FOREIGN TABLE ldap_user_tbl (uidNumber character varying, uid character varying, cn character varying, userPassword character varying) server ldap_server options (uri 'ldap://<サーバ名>', path '<検索ベース>', scope '<検索階層>', binddn '<バインドDN >', bindpwd '<バインドDNパスワード >', objectClass '検索属性');	testdb=> CREATE FOREIGN TABLE ldap_user_tbl (testdb(> uidNumber character varying, testdb(> uid character varying, testdb(> cn character varying, testdb(> userPassword character varying testdb(>) server ldap_server options (testdb(> uri 'ldap://133.227.221.136', testdb(> path 'ou=Users,dc=example,dc=com', testdb(> scope 'sub', testdb(> binddn 'cn=manager,dc=example,dc=com', testdb(> bindpwd 'admin', testdb(> objectClass '*' testdb(>); CREATE FOREIGN TABLE

9	PostgreSQL 外部表の検索	PostgreSQLサーバ	postgres	select * from ldap_user_tbl;	testdb=> select * from ldap_user_tbl; uidnumber | uid | cn | userpassword ----------+----------+----------+---------- 1001 | takahiko.takeda | Takahiko Takeda | takeda 1002 | shinya.yokoyama | Shinya Yokoyama | yokoyama 1003 | osamu.inoue | Osamu Inoue | inoue 1004 | naoki.ishikawa | Naoki Ishikawa | ishikawa 1005 | kazuo.tamura | Kazuo Tamura | tamura (6 行)

No	概要	対象	ユーザ	コマンド	確認（確認コマンド等）

1	PostgreSQL エクステンションの作成	PostgreSQLサーバ	postgres	CREATE EXTENSION file_fdw;	$ psql testdb
					
					testdb=# CREATE EXTENSION file_fdw;
					CREATE EXTENSION
					
					testdb=# ¥dx
					インストール済みの拡張の一覧
					名前 | バージョン | スキーマ | 説明
					-----------+-----------+-----------
					+---
					----------- 	
					file_fdw | 1.0 | public | foreign-data wrapper for flat file
					access
					multicorn | 1.3.1 | public | Multicorn Python bindings for
					Postgres 9.2.* Foreign Data Wrapper
					mysql_fdw | 1.0 | public | Foreign data wrapper for querying
					a MySQL server
					oracle_fdw | 1.1 | public | foreign data wrapper for Oracle
					access
					plpgsql | 1.0 | pg_catalog | PL/pgSQL procedural language
					sqlite_fdw | 0.0.1 | public | SQLite Foreign Data Wrapper
					(6 行)

2	PostgreSQL 外部サーバの作成	PostgreSQLサーバ	postgres	CREATE SERVER csv_server FOREIGN DATA WRAPPER file_fdw;	testdb=# CREATE SERVER csv_server testdb-# FOREIGN DATA WRAPPER file_fdw; CREATE SERVER testdb=# ¥des 外部サーバー一覧 名前 | 所有者 | 外部データラッパー -------------+----------+--------------------- csv_server | postgres | file_fdw ldap_server | postgres | multicorn mysql_server | postgres | mysql_fdw oracle_server | postgres | oracle_fdw sqlite_server | postgres | sqlite_fdw (5 行)
3	PostgreSQL 外部表の作成	PostgreSQLサーバ	postgres	CREATE FOREIGN TABLE csv_user_tbl (uidnumber int, uid text, cn text, gidnumber int, userpassword text, loginlevel int, uselevel int) SERVER csv_server OPTIONS (filename '/mnt/data/user.csv', format 'csv');	testdb=# CREATE FOREIGN TABLE csv_user_tbl testdb-# (testdb(# uidnumber int, testdb(# uid text, testdb(# cn text, testdb(# gidnumber int, testdb(# userpassword text, testdb(# loginlevel int, testdb(# uselevel int testdb(#) testdb-# SERVER csv_server testdb-# OPTIONS (filename '/mnt/data/user.csv', format 'csv'); CREATE FOREIGN TABLE NFSによるリモートサーバのファイル共有が設定済み ・/mnt/data

4	PostgreSQL 外部表の検索	PostgreSQLサーバ	postgres	select * from csv_user_tbl;	testdb=# select * from cvs_user_tbl; uidnumber | uid | cn | gidnumber | userpassword | loginlevel | uselevel ---+---+---+---+---+---+--- 52030 | taro.keiri | Taro Keiri | 100 | pass | 1 | 0 55040 | ichiro.keiri | Ichiro Keiri | 10010 | pass | 1 | 0 58020 | jiro.keiri | Jiro Keiri | 10010 | pass | 1 | 0 61050 | saburo.keiri | Saburo Keiri | 10010 | pass | 1 | 0 53150 | kazuo.keiri | Kazuo Keiri | 10020 | pass | 1 | 0 71080 | fusao.keiri | Fusao Keiri | 10020 | pass | 1 | 0 79010 | mituo.keiri | Mituo Keiri | 10020 | pass | 1 | 0 51030 | taro.jinji | Taro Jinji | 200 | pass | 1 | 0

検証シナリオ①

◆シナリオ検証①：人事太郎が休職

・業務要件：すべてのシステムのログイン許可を不可状態に変更する

実行する処理	実行クエリ

すべてのシステムのログイン
を不可状態に変更する
	testdb=# UPDATE mysql_user_tbl SET login = 0 WHERE userid = 51030;
	UPDATE oracle_kintai_tbl SET loginkyoka = 0 WHERE uidnumber = 51030;
	UPDATE 1
	testdb=# UPDATE oracle_kintai_tbl SET loginkyoka = 0 WHERE uidnumber = 51030;
	UPDATE 1
	testdb=# UPDATE postgres_user_tbl SET loginlevel = 0 WHERE uidnumber = 51030;
	UPDATE fb_user_tbl SET loginflg = 0 WHERE usercd = 51030;
	UPDATE 1
	testdb=# UPDATE oracle_user_tbl SET loginkyoka = 0 WHERE userid = 51030;
	UPDATE 1

◆業務前

ユーザ名	ユーザID	部署ID	人事給与login	人事給与利用権限	会計login	会計利用権限	勤怠管理login	勤怠管理利用権限	社員ポータルlogin	社員ポータル利用権限	福利厚生login	福利厚生利用権限
Taro Jinji	51030	200	1	1	1	0	1	0	1	0	1	0

◆業務完了後（赤字部分が変更箇所）

ユーザ名	ユーザID	部署ID	人事給与login	人事給与利用権限	会計login	会計利用権限	勤怠管理login	勤怠管理利用権限	社員ポータルlogin	社員ポータル利用権限	福利厚生login	福利厚生利用権限
Taro Jinji	51030	200	0	1	0	0	0	0	0	0	1	0

◆シナリオ検証②：営業三男が経理部財務課に異動

・業務要件：
　人事給与システムに一般ユーザとして新規ユーザ登録
　会計システムに一般ユーザとして新規ユーザ登録
　勤怠管理システムの所属組織を経理部に変更
　社員ポータルサイト上の所属組織を経理部に変更

実行する処理	実行クエリ
人事給与システムに一般ユーザとしてユーザ登録	testdb=# INSERT INTO oracle_user_tbl VALUES testdb-# (testdb(# 74031,'営業 三男',10020,'1','0' testdb(#); INSERT 0 1
会計システムに一般ユーザとしてユーザ登録	testdb=# INSERT INTO mysql_user_tbl VALUES testdb-# (testdb(# 74031,'営業 三男',10020,'1','0' testdb(#); INSERT 0 1
勤怠管理システムの所属組織を経理部に変更	testdb=# UPDATE oracle_kintai_tbl SET gidnumber=10020 WHERE uidnumber = 74031; UPDATE 1
社員ポータルサイト上の所属組織を経理部に変更	testdb=# UPDATE postgres_user_tbl SET gidnumber=10020 WHERE uidnumber = 74031; UPDATE 1

◆業務前

ユーザ名	ユーザID	部署ID	人事給与login	人事給与利用権限	会計login	会計利用権限	勤怠管理login	勤怠管理利用権限	社員ポータルlogin	社員ポータル利用権限	福利厚生login	福利厚生利用権限
Mituo Eigyo	74031	30020					1	0	1	0		

◆業務完了後（赤字は変更した箇所）

ユーザ名	ユーザID	部署ID	人事給与login	人事給与利用権限	会計login	会計利用権限	勤怠管理login	勤怠管理利用権限	社員ポータルlogin	社員ポータル利用権限	福利厚生login	福利厚生利用権限
Mituo Eigyo	74031	10020	1	0	1	0	1	0	1	0		

検証シナリオ③

◆シナリオ検証③：人事三郎が主任に昇格

・業務要件：人事給与システムの権限を管理者権限に変更

実行する処理	実行クエリ
人事給与システムの権限を管理者権限に変更	testdb=# UPDATE oracle_user_tbl SET kengen = 1 WHERE userid = 58060; UPDATE 1

◆業務前

ユーザ名	ユーザID	部署ID	人事給与login	人事給与利用権限	会計login	会計利用権限	勤怠管理login	勤怠管理利用権限	社員ポータルlogin	社員ポータル利用権限	福利厚生login	福利厚生利用権限
Saburo Jinji	58060	20010	1	0			1	0	1	0	1	0

◆業務完了後（赤字は変更した箇所）

ユーザ名	ユーザID	部署ID	人事給与login	人事給与利用権限	会計login	会計利用権限	勤怠管理login	勤怠管理利用権限	社員ポータルlogin	社員ポータル利用権限	福利厚生login	福利厚生利用権限
Saburo Jinji	58060	20010	1	1			1	0	1	0	1	0

◆シナリオ検証④：人事二郎が営業部に異動

・業務要件：

人事給与システムのユーザ登録を抹消する
福利厚生システムのユーザ登録を抹消する
勤怠管理システムの権限を一般ユーザ権限に変更
勤怠管理システム上の所属組織を営業部に変更
社員ポータルサイト上の所属組織を営業部に変更

実行する処理	実行クエリ
人事給与システムのユーザ登録を抹消する	testdb=# DELETE FROM oracle_user_tbl WHERE userid = 55090;
	DELETE 1
福利厚生システムのユーザ登録を抹消する	testdb=# DELETE FROM fb_user_tbl WHERE usercd = 55090;
	DELETE 1
勤怠管理システムの権限を一般ユーザ権限に変更	testdb=# UPDATE oracle_kintai_tbl SET loginkyoka = 0,gidnumber = 30010
	WHERE uidnumber = 55090;
勤怠管理システム上の所属組織を営業部に変更	UPDATE 1
社員ポータルサイト上の所属組織を営業部に変更	testdb=# UPDATE postgres_user_tbl SET gidnumber = 30010
	WHERE uidnumber = 55090;
	UPDATE 1

◆業務前

ユーザ名	ユーザID	部署ID	人事給与login	人事給与利用権限	会計login	会計利用権限	勤怠管理login	勤怠管理利用権限	社員ポータルlogin	社員ポータル利用権限	福利厚生login	福利厚生利用権限
Jiro Jinji	55090	20010	1	1			0	0	1	0	1	0

◆業務完了後（赤字は変更した箇所）

ユーザ名	ユーザID	部署ID	人事給与login	人事給与利用権限	会計login	会計利用権限	勤怠管理login	勤怠管理利用権限	社員ポータルlogin	社員ポータル利用権限	福利厚生login	福利厚生利用権限
Jiro Jinji	55090	30010	（削除）	（削除）			0	0	1	0	（削除）	（削除）

◆検証シナリオ⑤：開発二郎が退職

・業務要件：

勤怠管理システムのユーザ登録を抹消する

社員ポータルサイト上のユーザ登録を抹消する

実行する処理	実行クエリ
勤怠管理システムのユーザ登録を抹消する	testdb=# DELETE FROM oracle_kintai_tbl WHERE uidnumber = 75082;
	DELETE 1
社員ポータルサイト上のユーザ登録を抹消する	testdb=# DELETE FROM postgres_user_tbl WHERE uidnumber = 75082;
	DELETE 1

◆業務前

ユーザ名	ユーザID	部署ID	人事給与login	人事給与利用権限	会計login	会計利用権限	勤怠管理login	勤怠管理利用権限	社員ポータルlogin	社員ポータル利用権限	福利厚生login	福利厚生利用権限
Jiro Kaihatu	75082	40010					1	0	1	0		

◆業務完了後（赤字は変更した箇所）

（レコード削除）

検証結果

◆シナリオで想定した業務の実行状態を比較したものです

以下の表は、全システムの外部表を結合し、その中からログイン許可情報、システム利用権限をそれぞれ取り出したものです。

	ユーザ名	ユーザID	部署ID	人事給与login	人事給与利用権限	会計login	会計利用権限	勤怠管理login	勤怠管理利用権限	社員ポータルlogin	社員ポータル利用権限	福利厚生login	福利厚生利用権限
	Taro Keiri	52030	100	1	0	1	1	1	0	1	0		
業務①で変更	Taro Jinji	51030	200	1	1	1	0	1	0	1	0	1	0
	Taro Eigyo	52071	300					1	0	1	0		
	Taro Kaihatu	67022	400					1	0	1	0		
	Taro System	58023	500					1	1	1	1		
	Ichiro Keiri	55040	10010	1	0	1	1	1	0	1	0		
	Saburo Keiri	61050	10010	1	0	1	0	1	0	1	0		
	Jiro Keiri	58020	10010	1	0	1	0	1	0	1	0		
	Mituo Keiri	79010	10020	1	0	1	0	1	0	1	0		
	Kazuo Keiri	53150	10020	1	0	1	1	1	0	1	0		
	Fusao Keiri	71080	10020	1	0	1	0	1	0	1	0		
	Ichiro Jinji	52100	20010	1	1	1	0	1	0	1	0	1	0
業務③で変更	Saburo Jinji	58060	20010	1	0			1	0	1	0	1	0
業務④で変更	Jiro Jinji	55090	20010	1	1			0	0	1	0	1	0
	Kazuo Jinji	62010	20020	1	1	1	0	1	0	1	0	1	1
	Mituo Jinji	76020	20020	0	0			1	0	1	0	1	1
	Fusao Jinji	65030	20020	1	1	1	0	1	0	1	0	1	1
	Ichiro Eigyo	61011	30010					1	0	1	0		
	Jiro Eigyo	62051	30010					1	0	1	0		
	Saburo Eigyo	67071	30010					1	0	1	0		
業務②で変更	Mituo Eigyo	74031	30020					1	0	1	0		
	Kazuo Eigyo	72011	30020					1	0	1	0		
	Fusao Eigyo	73021	30020					1	0	1	0		
業務⑤で変更	Jiro Kaihatu	75082	40010					1	0	1	0		
	Ichiro Kaihatu	73092	40010					1	0	1	0		
	Saburo Kaihatu	77142	40010					1	0	1	0		
	Kazuo Kaihatu	75222	40020					1	0	1	0		
	Fusao Kaihatu	76892	40020					1	0	1	0		
	Mituo Kaihatu	74322	40020					1	0	1	0		
	Jiro System	67183	50010					1	1	1	1		
	Ichiro System	63043	50010					1	1	1	1		

(31 行)

検証結果

	ユーザ名	ユーザID	部署ID	人事給与login	人事給与利用権限	会計login	会計利用権限	勤怠管理login	勤怠管理利用権限	社員ポータルlogin	社員ポータル利用権限	福利厚生login	福利厚生利用権限
	Taro Keiri	52030	100	1	0	1	1	1	0	1	0		
業務①で変更	Taro Jinji	51030	200	0	1	0	0	0	0	0	0	1	0
	Taro Eigyo	52071	300					1	0	1	0		
	Taro Kaihatu	67022	400					1	0	1	0		
	Taro System	58023	500					1	1	1	1		
	Ichiro Keiri	55040	10010	1	0	1	1	1	0	1	0		
	Saburo Keiri	61050	10010	1	0	1	0	1	0	1	0		
	Jiro Keiri	58020	10010	1	0	1	0	1	0	1	0		
	Mituo Keiri	79010	10020	1	0	1	0	1	0	1	0		
	Kazuo Keiri	53150	10020	1	0	1	1	1	0	1	0		
	Fusao Keiri	71080	10020	1	0	1	0	1	0	1	0		
業務②で変更	Mituo Eigyo	74031	10020	1	0	1	0	1	0	1	0		
	Ichiro Jinji	52100	20010	1	1	1	0	1	0	1	0	1	0
業務③で変更	Saburo Jinji	58060	20010	1	1			1	0	1	0	1	0
	Kazuo Jinji	62010	20020	1	1	1	0	1	0	1	0	1	1
	Fusao Jinji	65030	20020	1	1	1	0	1	0	1	0	1	1
	Mituo Jinji	76020	20020	0	0			1	0	1	0	1	1
	Saburo Eigyo	67071	30010					1	0	1	0		
	Jiro Eigyo	62051	30010					1	0	1	0		
	Ichiro Eigyo	61011	30010					1	0	1	0		
業務④で変更	Jiro Jinji	55090	30010	（削除）	（削除）			0	0	1	0	（削除）	（削除）
	Fusao Eigyo	73021	30020					1	0	1	0		
	Kazuo Eigyo	72011	30020					1	0	1	0		
	Ichiro Kaihatu	73092	40010					1	0	1	0		
	Saburo Kaihatu	77142	40010					1	0	1	0		
	Kazuo Kaihatu	75222	40020					1	0	1	0		
	Fusao Kaihatu	76892	40020					1	0	1	0		
	Mituo Kaihatu	74322	40020					1	0	1	0		
	Jiro System	67183	50010					1	1	1	1		
	Ichiro System	63043	50010					1	1	1	1		

業務⑤によって
レコード一件減

（30 行）

◆検証環境準備

–	操作	コマンド・結果【Postgres】	コマンド・結果【Oracle】
環境準備	Oracleに検証用のテーブルを作成	–	SQL> CREATE TABLE JIP.USER_TABLE (user_id NUMBER, user_name VARCHAR2(20), password VARCHAR2(20), user_group VARCHAR2(20), PRIMARY KEY(user_id)); 表が作成されました。
	検証用テーブルにデータ挿入	–	省略（本文参照）
	外部テーブルの作成	testdb=> CREATE FOREIGN TABLE FDW_TEST_PSQL_ORACLE testdb-> (testdb(> user_id INTEGER OPTIONS(KEY'YES'), testdb(> user_name TEXT, testdb(> password TEXT, testdb(> user_group TEXT, testdb(>) testdb-> SERVER oracle_server testdb-> OPTIONS(SCHEMA'JIP',TABLE'USER_TABLE'); CREATE FOREIGN TABLE	–
	検証用テーブルの一覧	testdb=> SELECT * FROM FDW_TEST_PSQL_ORACLE; user_id | user_name | password | user_group ---------+---------------+-------------+------------ 1 | David_Hume | pass_hume | thought 2 | Immanuel_Kant | pass_kant | philosophy 3 | Gilles_Deleuze | pass_gilles | philosophy 4 | Hitoshi_Nagai | pass_hitoshi | philosophy (4 行)	SQL> SELECT * FROM JIP.USER_TABLE; USER_ID USER_NAME PASSWORD USER_GROUP ---------- -------------------- -------------------- -------------------- 1 David_Hume pass_hume thought 2 Immanuel_Kant pass_kant philosophy 3 Gilles_Deleuze pass_gilles philosophy 4 Hitoshi_Nagai pass_hitoshi philosophy

◆基本動作（CRUD）の検証

検証内容	操作	コマンド・結果【Postgres】	コマンド・結果【Oracle】
任意のレコードを更新する 任意のレコードを削除する レコードを新規登録する	更新前テーブルの確認	testdb=> SELECT * FROM FDW_TEST_PSQL_ORACLE; user_id | user_name | password | user_group ---------+----------------+-------------+------------ 1 | David_Hume | pass_hume | thought 2 | Immanuel_Kant | pass_kant | philosophy 3 | Gilles_Deleuze | pass_gilles | philosophy 4 | Hitoshi_Nagai | pass_hitoshi | philosophy (4 行)	SQL> SELECT * FROM JIP.USER_TABLE; USER_ID USER_NAME PASSWORD USER_GROUP ---------- -------------------- -------------------- -------------------- 1 David_Hume pass_hume thought 2 Immanuel_Kant pass_kant philosophy 3 Gilles_Deleuze pass_gilles philosophy 4 Hitoshi_Nagai pass_hitoshi philosophy
	uesr_id = 5 のレコードを新規登録する	testdb=> INSERT INTO FDW_TEST_PSQL_ORACLE VALUES testdb-> (testdb(> 5, testdb(> 'Kitaro_Nishida', testdb(> 'pass_kitaro', testdb(> 'philosophy' testdb(>); INSERT 0 1	–
	user_jd = 2 のレコードのuser_groupを、'philosophy' から 'thought'に更新する	testdb=> UPDATE FDW_TEST_PSQL_ORACLE SET user_group = 'thought' WHERE user_id = 2; UPDATE 1	–
	user_jd = 4 のレコードを削除する	testdb=> DELETE FROM FDW_TEST_PSQL_ORACLE WHERE user_id = 4; DELETE 1	–
	更新後テーブルの確認	testdb=> SELECT * FROM FDW_TEST_PSQL_ORACLE; user_id | user_name | password | user_group ---------+----------------+-------------+------------ 1 | David_Hume | pass_hume | thought 2 | Immanuel_Kant | pass_kant | thought 3 | Gilles_Deleuze | pass_gilles | philosophy 5 | Kitaro_Nishida | pass_kitaro | philosophy (4 行)	SQL> SELECT * FROM JIP.USER_TABLE; USER_ID USER_NAME PASSWORD USER_GROUP ---------- -------------------- -------------------- -------------------- 1 David_Hume pass_hume thought 2 Immanuel_Kant pass_kant thought 3 Gilles_Deleuze pass_gilles philosophy 5 Kitaro_Nishida pass_kitaro philosophy

◆トランザクション特性検証（COMMIT機能）

検証内容	操作	コマンド・結果【Postgres】	コマンド・結果【Oracle】
COMMIT機能の確認	更新前テーブルの確認	testdb=> SELECT * FROM FDW_TEST_PSQL_ORACLE; user_id | user_name | password | user_group ---------+---------------+-------------+------------ 1 | David_Hume | pass_hume | thought 2 | Immanuel_Kant | pass_kant | philosophy 3 | Gilles_Deleuze | pass_gilles | philosophy 4 | Hitoshi_Nagai | pass_hitoshi | philosophy (4 行)	SQL> SELECT * FROM JIP.USER_TABLE; USER_ID USER_NAME PASSWORD USER_GROUP ---------- -------------------- -------------------- -------------------- 1 David_Hume pass_hume thought 2 Immanuel_Kant pass_kant philosophy 3 Gilles_Deleuze pass_gilles philosophy 4 Hitoshi_Nagai pass_hitoshi philosophy
	トランザクション開始	testdb=> BEGIN; BEGIN	-
	外部テーブルに user_id = 5のレコードを挿入	testdb=> INSERT INTO FDW_TEST_PSQL_ORACLE VALUES(testdb(> 5,'Kitaro_Nishida','pass_kitaro','philosophy' testdb(>); INSERT 0 1	-
	外部テーブルのuser_id = 2のレコードの user_groupを'thought'に更新	testdb=> UPDATE FDW_TEST_PSQL_ORACLE SET user_group = 'thought' WHERE user_id = 2; UPDATE 1	-
	トランザクション内にのみ 更新がされていることを確認	testdb=> SELECT * FROM FDW_TEST_PSQL_ORACLE; user_id | user_name | password | user_group ---------+---------------+-------------+------------ 1 | David_Hume | pass_hume | thought 2 | Immanuel_Kant | pass_kant | thought 3 | Gilles_Deleuze | pass_gilles | philosophy 4 | Hitoshi_Nagai | pass_hitoshi | philosophy 5 | Kitaro_Nishida | pass_kitaro | philosophy (5 行)	SQL> SELECT * FROM JIP.USER_TABLE; USER_ID USER_NAME PASSWORD USER_GROUP ---------- -------------------- -------------------- -------------------- 1 David_Hume pass_hume thought 2 Immanuel_Kant pass_kant philosophy 3 Gilles_Deleuze pass_gilles philosophy 4 Hitoshi_Nagai pass_hitoshi philosophy
	トランザクションのコミット	testdb=> COMMIT; COMMIT	-
	外部環境にコミット内容が 反映されていることを確認	testdb=> SELECT * FROM FDW_TEST_PSQL_ORACLE; user_id | user_name | password | user_group ---------+---------------+-------------+------------ 1 | David_Hume | pass_hume | thought 2 | Immanuel_Kant | pass_kant | thought 3 | Gilles_Deleuze | pass_gilles | philosophy 4 | Hitoshi_Nagai | pass_hitoshi | philosophy 5 | Kitaro_Nishida | pass_kitaro | philosophy (5 行)	SQL> SELECT * FROM JIP.USER_TABLE; USER_ID USER_NAME PASSWORD USER_GROUP ---------- -------------------- -------------------- -------------------- 1 David_Hume pass_hume thought 2 Immanuel_Kant pass_kant thought 3 Gilles_Deleuze pass_gilles philosophy 4 Hitoshi_Nagai pass_hitoshi philosophy 5 Kitaro_Nishida pass_kitaro philosophy

◆トランザクション特性検証(ROLLBACK機能)

検証内容	操作	コマンド・結果【Postgres】	コマンド・結果【Oracle】
ROLLBACK機能の確認	更新前テーブルの確認	省略(COMMIT機能の確認部分参照)	
	トランザクション開始	testdb=> BEGIN; BEGIN	–
	外部テーブルに user_id = 5のレコードを挿入	省略(COMMIT機能の確認部分参照)	–
	外部テーブルのuser_id = 2のレコードの user_groupを'thought'に更新	省略(COMMIT機能の確認部分参照)	–
	トランザクション内にのみ 更新がされていることを確認	testdb=> SELECT * FROM FDW_TEST_PSQL_ORACLE; user_id | user_name | password | user_group ---------+---------------+--------------+------------ 1 | David_Hume | pass_hume | thought 2 | Immanuel_Kant | pass_kant | thought 3 | Gilles_Deleuze | pass_gilles | philosophy 4 | Hitoshi_Nagai | pass_hitoshi | philosophy 5 | Kitaro_Nishida | pass_kitaro | philosophy (5 行)	SQL> SELECT * FROM JIP.USER_TABLE; USER_ID USER_NAME PASSWORD USER_GROUP ---------- -------------------- -------------------- -------------------- 1 David_Hume pass_hume thought 2 Immanuel_Kant pass_kant philosophy 3 Gilles_Deleuze pass_gilles philosophy 4 Hitoshi_Nagai pass_hitoshi philosophy
	トランザクションのロールバック	testdb=> ROLLBACK; ROLLBACK	–
	更新前の状態に戻ることを確認	testdb=> SELECT * FROM FDW_TEST_PSQL_ORACLE; user_id | user_name | password | user_group ---------+---------------+--------------+------------ 1 | David_Hume | pass_hume | thought 2 | Immanuel_Kant | pass_kant | philosophy 3 | Gilles_Deleuze | pass_gilles | philosophy 4 | Hitoshi_Nagai | pass_hitoshi | philosophy (4 行)	SQL> SELECT * FROM JIP.USER_TABLE; USER_ID USER_NAME PASSWORD USER_GROUP ---------- -------------------- -------------------- -------------------- 1 David_Hume pass_hume thought 2 Immanuel_Kant pass_kant philosophy 3 Gilles_Deleuze pass_gilles philosophy 4 Hitoshi_Nagai pass_hitoshi philosophy

・検証パターン1 （別トランザクションによる同一データの更新、トランザクションAがCOMMITし、トランザクションBの処理が失敗するパターン）

検証内容	操作	コマンド・結果【Postgres（トランザクションA）】	コマンド・結果【Postgres（トランザクションB）】
トランザクションAで 外部表の1つのレコードを UPDATEする	【Postgres】 トランザクションA開始	testdb=> BEGIN; BEGIN	–
	【Postgres】 user_id = 2 のレコードのuser_groupを、 'philosophy' から 'thought'に更新する	testdb=> UPDATE fdw_test_psql_oracle SET user_group = 'thought' WHERE user_id = 2; UPDATE 1	–
トランザクションBで 外部表の同一キーのレコードを UPDATEする	【Postgres】 トランザクションB開始	–	testdb=> BEGIN; BEGIN
	【Postgres】 user_id = 2 のレコードのuser_groupを、 'philosophy' から 'thought'に更新する	–	testdb=> UPDATE fdw_test_psql_oracle SET user_group = 'thought' WHERE user_id = 2; （ロック待ち状態となる）
検証内容	操作	–	コマンド・結果【Oracle】
ロック情報を取得する	【Oracle】 ロック情報を取得する	–	SQL> SELECT a.SID sid,a.USERNAME username,a.SERIAL# serialno,a.PROGRAM program, b.REQUEST request,b.TYPE type,b.BLOCK block,b.LMODE lmode FROM V$SESSION a,V$LOCK b WHERE a.SID = b.SID AND b.TYPE IN ('TX','TM'); SID USERNAME SERIALNO ---------- ------------------------------ ---------- PROGRAM REQUEST TY BLOCK -- ---------- -- ---------- LMODE ---------- 34 SCOTT 31429 0 TX 1 6 38 SCOTT 24973 6 TX 0 0 38 SCOTT 24973 0 TM 0 3 34 SCOTT 31429 0 TM 0 3 **（2セッションのうち1つが行排他ロックのリクエストを行っていること、 もう1つのセッションが別の行排他ロックの妨げとなっていることが確認できる）**
検証内容	操作	コマンド・結果【Postgres（トランザクションA）】	コマンド・結果【Postgres（トランザクションB）】
トランザクションAをコミットすると エラーが発生する	【Postgres】 トランザクションAのコミット	testdb=> COMMIT; COMMIT	ERROR: error executing query: OCIStmtExecute failed to execute remote query DETAIL: ORA-08177: can't serialize access for this transaction

・検証パターン1　（別トランザクションによる同一データの更新、トランザクションAがROLLBACKし、トランザクションBの処理が即座に開始されるパターン）

検証内容	操作	コマンド・結果【Postgres（トランザクションA）】	コマンド・結果【Postgres（トランザクションB）】
外部表の1つのレコードを UPDATEする	【Postgres】 user_id = 2 のレコードのuser_groupを、 'philosophy' から 'thought'に更新する	省略（トランザクションをCOMMITした場合と同様）	-
外部表の同一キーのレコードを UPDATEする	【Postgres】 user_id = 2 のレコードのuser_groupを、 'philosophy' から 'thought'に更新する	省略（トランザクションをCOMMITした場合と同様）	-
ロック情報を取得する	【Oracle】 ロック情報を取得する	省略（トランザクションをCOMMITした場合と同様）	-
トランザクションAをロールバック するとトランザクションBが直ちに 開始される	【Postgres】 トランザクションAのロールバック	testdb=> ROLLBACK; ROLLBACK	UPDATE 1

・検証パターン2（別トランザクションによる別データの更新）

検証内容	操作	コマンド・結果【Postgres（トランザクションA）】	コマンド・結果【Postgres（トランザクションB）】
トランザクションAで 外部表の1つのレコードを UPDATEする	【Postgres】 トランザクションA開始	testdb=> BEGIN; BEGIN	–
	【Postgres】 user_jd = 2 のレコードのuser_groupを、 'philosophy' から 'thought'に更新する	testdb=> UPDATE fdw_test_psql_oracle SET user_group = 'thought' WHERE user_id = 2; UPDATE 1	–
トランザクションBで 外部表のロックされていない レコードをUPDATEする	【Postgres】 トランザクションB開始	–	testdb=> BEGIN; BEGIN
	【Postgres】 user_jd = 5 のレコードのuser_groupを、 'philosophy' から 'thought'に更新する	–	testdb=> update fdw_test_psql_oracle set user_group = 'philosophy' Where user_id = 5; UPDATE 1
検証内容	操作	–	コマンド・結果【Oracle】
ロック情報を取得する	【Oracle】 ロック情報を取得する	–	SQL> SELECT a.SID sid,a.USERNAME username,a.SERIAL# serialno,a.PROGRAM program, b.REQUEST request,b.TYPE type,b.BLOCK block,b.LMODE lmode FROM V$SESSION a,V$LOCK b WHERE a.SID = b.SID AND b.TYPE IN ('TX','TM'); SID USERNAME SERIALNO ---------- ------------------------------ ---------- PROGRAM REQUEST TY BLOCK -- ---------- -- ---------- LMODE ---------- 34 SCOTT 31429 0 TX 0 6 38 SCOTT 24973 0 TX 0 6 38 SCOTT 24973 0 TM 0 3 34 SCOTT 31429 0 TM 0 3 **（2セッションが表共有ロック及び行排他ロックを行っていることが確認できる）**

・検証パターン3（NOWAITオプションの動作確認）

検証内容	操作	コマンド・結果【Postgres（トランザクションA）】	コマンド・結果【Postgres（トランザクションB）】
トランザクションAで 外部表の1つのレコードを UPDATEする	【Postgres】 トランザクションA開始	testdb=> BEGIN; BEGIN	–
	【Postgres】 user_id = 2 のレコードのuser_groupを、 'philosophy' から 'thought'に更新する	testdb=> UPDATE fdw_test_psql_oracle SET user_group = 'thought' WHERE user_id = 2; UPDATE 1	–
検証内容	操作	–	コマンド・結果【Oracle】
Oracle側でNOWAIT オプションが機能するかどうか 確認する	【Oracle】 user_id = 2のレコードを NOWAITオプションを付けて SELECT FOR UPDATEする	–	SQL> SELECT * FROM JIP.USER_TABLE WHERE user_id = 2 FOR UPDATE NOWAIT; SELECT * FROM JIP.USER_TABLE WHERE user_id = 2 FOR UPDATE NOWAIT * 行1でエラーが発生しました。: ORA-00054: リソース・ビジー。NOWAITが指定されているか、タイムアウトしました
検証内容	操作	コマンド・結果【Postgres（トランザクションA）】	コマンド・結果【Postgres（トランザクションB）】
oracle_fdw経由で リソースビジーが返るかどうか 確認	【Postgres】 user_id = 2のレコードを NOWAITオプションを付けて SELECT FOR UPDATEする	–	testdb=> SELECT * FROM fdw_test_psql_oracle FOR UPDATE NOWAIT; （ロック解除待ちとなる）
トランザクションAをコミットすると エラーが発生する	【Postgres】 トランザクションAのコミット	testdb=> COMMIT; COMMIT	ERROR: error executing query: OCIStmtExecute failed to execute remote query DETAIL: ORA-08177: can't serialize access for this transaction