Innovate with you

OSS-DB研究会で得られた技術者必見の知見とは？

KS Solutions

2013年6月7日
関電システムソリューションズ株式会社
経営改革推進本部　技術部　システム技術グループ
松添　隆康

Innovate with you

アジェンダ

・会社紹介
・OracleからPostgreSQLへのデータ移行の主な課題
・Ora2Pgの仕様確認
・Ora2Pgを利用したデータ移行の課題解決策検討
・検証環境
・検証結果総括
・今後の課題と取り組み

【付録】
-検証結果
-手作りSQL
-Ora2Pgのパラメータについて

KS Solutions

アジェンダ

・会社紹介
・OracleからPostgreSQLへのデータ移行の主な課題
・Ora2Pgの仕様確認
・Ora2Pgを利用したデータ移行の課題解決策検討
・検証環境
・検証結果総括
・今後の課題と取り組み

【付録】
-検証結果
-手作りSQL
-Ora2Pgのパラメータについて

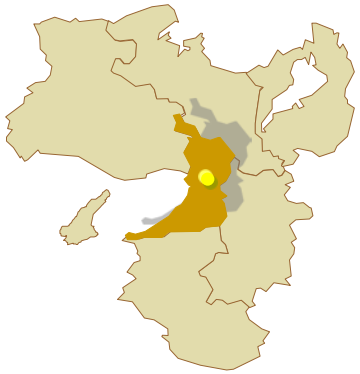
会社紹介　概要

関電システムソリューションズ株式会社
（Kanden System Solutions Co.,Inc. 略称 KS Solutions)

・ 設立	：2004年10月1日 ※関電情報システム株式会社（1967年4月設立）と株式会社関西テレコムテクノロジー（1986年5月設立）は2004年10月1日に合併し関電システムソリューションズ株式会社となりました。
・ 資本金	：9,000万円
・ 売上高	：380億（2011年度実績）
・ 株主構成	：関西電力株式会社100%出資
・ 従業員数	：1,166名（2013年4月1日現在）
・ 主な営業品目	：情報通信システムのコンサルティング、情報化戦略の立案 情報通信システムの計画、設計、構築、保守、運用管理 情報通信アプリケーションサービスの開発、提供 情報通信システム設備管理・運用のアウトソーシング
・ 主な受注先	：関西電力(株)、関電関係会社、法人 、地方自治体
・ グループ会社	：関西コンピュータサービス株式会社（KCS） 関西レコードマネジメント株式会社（KRM） 中央コンピューター株式会社（CCC）

KS Solutions

（参考）都市型の新データセンター



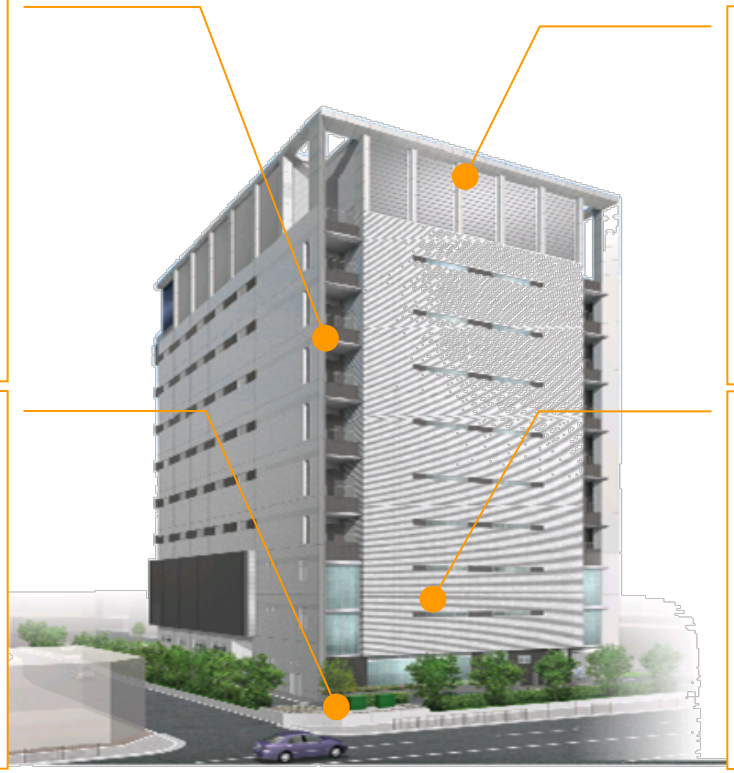
大阪第１データセンター
（大阪市北区　2001年10月～）

大阪第２データセンター
（大阪市福島区　2002年10月～）

大阪第３データセンター
（大阪市北区　2011年12月～）

監視ルーム
24時間365日お預かりしたシステムの稼働状況を有人監視。万が一のトラブルにも技術者が即座に対応いたします。

免震システム
通常の耐震基準を超える大地震でも、基準内の揺れに抑制するビル免震システムを採用。



グリーンIT設備
太陽光発電や外気冷却による自然エネルギーを利用。高効率空調システムで国内最高レベルのPUEを実現。

電源設備
3スポット給電、非常用発電機(EG)、無停電電源装置(CVCF)の冗長化による止まらない受電設備。

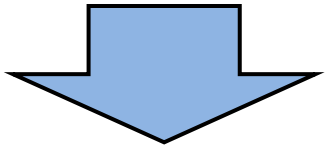
アジェンダ

・会社紹介
・OracleからPostgreSQLへのデータ移行の主な課題
・Ora2Pgの仕様確認
・Ora2Pgを利用したデータ移行の課題解決策検討
・検証環境
・検証結果総括
・今後の課題と取り組み

【付録】
-検証結果
-手作りSQL
-Ora2Pgのパラメータについて

OracleからPostgreSQLへのデータ移行の主な課題

・移行工数を削減したい
・文字コードを変換したい
・複数回に分けて実施したい
・既存環境に手を加えず実施したい
・移行漏れを防ぎたい



デファクトスタンダードの Ora2Pg で
どこまで解決できるか？

本日は、OSS-DB研究会での検証をご紹介する。

アジェンダ

・会社紹介
・OracleからPostgreSQLへのデータ移行の主な課題
・Ora2Pgの仕様確認
・Ora2Pgを利用したデータ移行の課題解決策検討
・検証環境
・検証結果総括
・今後の課題と取り組み

【付録】
-検証結果
-手作りSQL
-Ora2Pgのパラメータについて

Ora2Pgの仕様確認

・機能（＝パラメータ）の抽出 → 81機能を確認（man ora2pgの実行結果から）
・機能（＝パラメータ）の理解 → 一覧表を作成

カテゴリ	パラメータ	デフォルト値	設定値	パラメータの意味
Oracle database connection				
1	ORACLE_HOME	/usr/local/oracle/10g	Oracle Clientをインストールする際に使用したのORACLE_HOMEを設定する。	OracleClientソフトウェアがインストールされているディレクトリを指定する。 DBD.pmで使用される。
2	ORACLE_DSN	dbi:Oracle:host=mydb.mydom.fr;sid=SIDNAME	既存リスナーに接続するためのtnsnames.oraを作成し、dbi:Oracle:<TNS接続識別子>を設定する。	Oracleデータベースへの接続情報を記述する。 ■ポート番号：1521へ接続する場合 dbi:Oracle:host=<ホスト名>;sid=<オラクルSID> ■ポート番号：1521以外へ接続する場合 dbi:Oracle:<TNS接続識別子> ※tnsnames.oraの設定が必要
途中省略				
Limiting object to export				
24	ALLOW	未設定	オブジェクト名	エクスポートしたいオブジェクトを限定したい場合に指定する。 正規表現を用いることができる。 スペースで区切って、複数指定することができる。
25	EXCLUDE	未設定	オブジェクト名	ALLOWパラメータの逆の意味を持つ。
途中省略				
81	IMPORT	なし	全システム共通で使用するパラメータ値を設定したファイル名を指定	ora2pgのパラメータを別ファイルに記述し、使用したい場合はIMPORTパラメータに別ファイルの名前を指定する。

アジェンダ

・会社紹介
・OracleからPostgreSQLへのデータ移行の主な課題
・Ora2Pgの仕様確認
・Ora2Pgを利用したデータ移行の課題解決策検討
・検証環境
・検証結果総括
・今後の課題と取り組み

【付録】
-検証結果
-手作りSQL
-Ora2Pgのパラメータについて

Ora2Pgを利用したデータ移行の課題解決策検討

主な課題	方針	具体策（案）
移行工数を削減したい	パラメータを標準化する	IMPORTを使用する
	手動作業を減らす	PG_NUMERIC_TYPE、PG_INTEGER_TYPE、DEFAULT_NUMERIC、NULL_EQUAL_EMPTYを使用する
文字コードを変換したい	Ora2Pgで対応する	NLS_LANG、BINMODE、CLIENT_ENCODINGを使用する
複数回に分けて実施したい	テーブルを絞る	ALLOW、EXCLUDEを使用する
	レコードを絞る	WHEREを使用する
	スキーマを絞る	-nオプションを使用する
既存環境に手を加えず実施したい	Oracleクライアント側の設定を工夫する	tnsnames.oraを作成し、ORACLE_DSNを使用する
移行漏れを防ぎたい	移行前後で設定を一致させる	KEEP_PKEY_NAMES、FKEY_DEFERRABLEを使用する
	手動で対応しないといけない作業を明確化する	Oracle側で実行するSQLを作成する

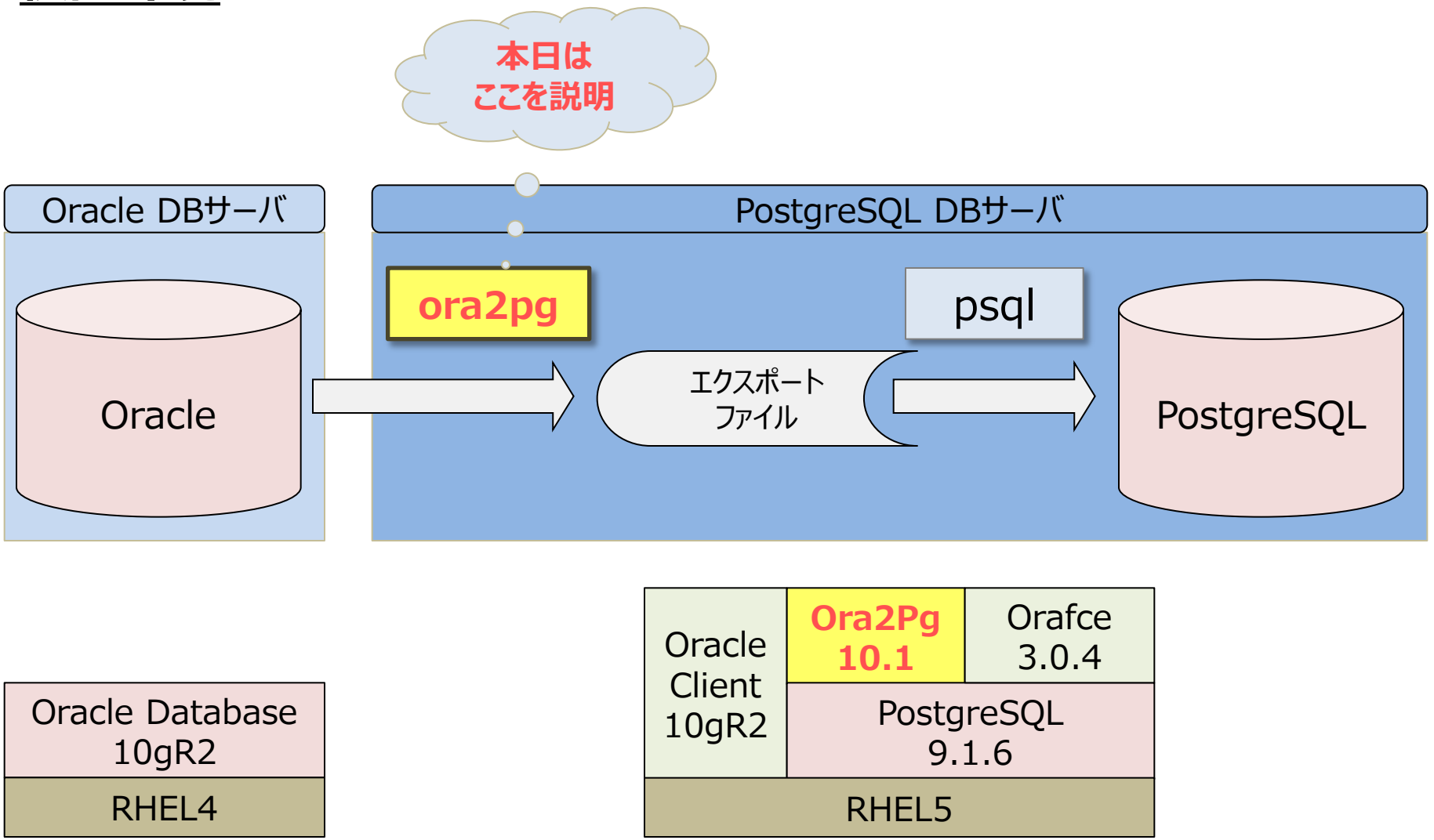
パラメータの詳細はP.56～P.60を参照

アジェンダ

・会社紹介
・OracleからPostgreSQLへのデータ移行の主な課題
・Ora2Pgの仕様確認
・Ora2Pgを利用したデータ移行の課題解決策検討
・検証環境
・検証結果総括
・今後の課題と取り組み

【付録】
-検証結果
-手作りSQL
-Ora2Pgのパラメータについて

検証環境



Oracle Database 10gR2
RHEL4

Oracle Client 10gR2	Ora2Pg 10.1	Orafce 3.0.4
	PostgreSQL 9.1.6	
RHEL5		

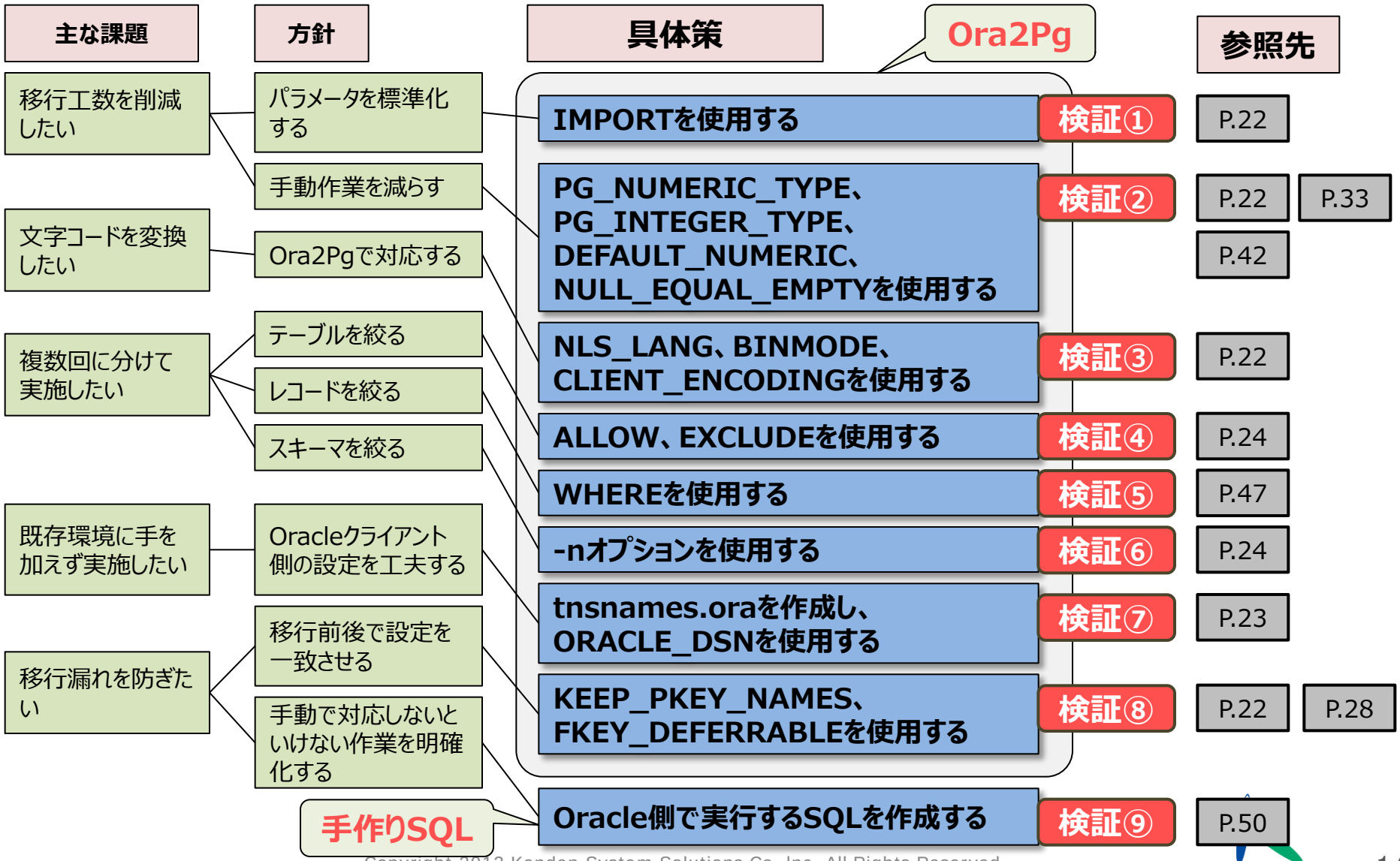
アジェンダ

- 会社紹介
- OracleからPostgreSQLへのデータ移行の主な課題
- Ora2Pgの仕様確認
- Ora2Pgを利用したデータ移行の課題解決策検討
- 検証環境
- **検証結果総括**
- 今後の課題と取り組み

【付録】
- -検証結果
- -手作りSQL
- -Ora2Pgのパラメータについて

検証結果総括

結論	Ora2Pgと手作りSQLを組み合わせることで、主な課題を解決できることができた。



アジェンダ

・会社紹介
・OracleからPostgreSQLへのデータ移行の主な課題
・Ora2Pgの仕様確認
・Ora2Pgを利用したデータ移行の課題解決策検討
・検証環境
・検証結果総括
・今後の課題と取り組み

【付録】
-検証結果
-手作りSQL
-Ora2Pgのパラメータについて

今後の課題と取り組み

- 今回の検証では、Ora2Pgと手作りSQLを組み合わせることで主な課題を解決することができた。
- しかし、時間と検証環境の制約により、エンタープライズ機能などの検証が充分でない。
- 今後は以下のような移行検証の実施や、移行方法の整理を行い、実案件に備えたい。

（例）

- パーティションテーブル
- マテリアライズドビュー
- DBリンク
- BFILE型、BLOB型データ
- 外部表

etc

アジェンダ

・会社紹介
・OracleからPostgreSQLへのデータ移行の主な課題
・Ora2Pgの仕様確認
・Ora2Pgを利用したデータ移行の課題解決策検討
・検証環境
・検証結果総括
・今後の課題と取り組み

【付録】
-検証結果
-手作りSQL
-Ora2Pgのパラメータについて

検証準備(1 of 3)

テーブル（1 of 2）

スキーマ	テーブル名	コメント	制約		DEFAULT句	インデックス		エクスポート対象
			種類	名前		種類	名前	
user1	commenttbl001	○						○
	consttbl001		PK	tbl001_pk				○
	consttbl002		FK	tbl002_fk				○
	consttbl003		チェック	tbl003_chk1	○			○
	consttbl004		ユニーク	tbl004_uq				○
	consttbl005		PK	自動付与				○
	consttbl006		FK	自動付与				○
	consttbl007		チェック	自動付与				○
	datatypetbl001							○
	datatypetbl002							○
	datatypetbl003							×

検証準備(2 of 3)

テーブル（2 of 2）

スキーマ	テーブル名	コメント	制約		DEFAULT句	インデックス		エクスポート対象
			種類	名前		種類	名前	
user1	excludetbl001							×
	excludetbl002							×
	excludetbl003							×
	indextbl001					ファンクション	tbl001_idx1	○
	indextbl002					昇順	tbl002_idx1	○
	indextbl003					降順	tbl003_idx1	○
	indextbl004					ユニーク	tbl004_idx1	○
	copytbl001							○
	copytbl002							○

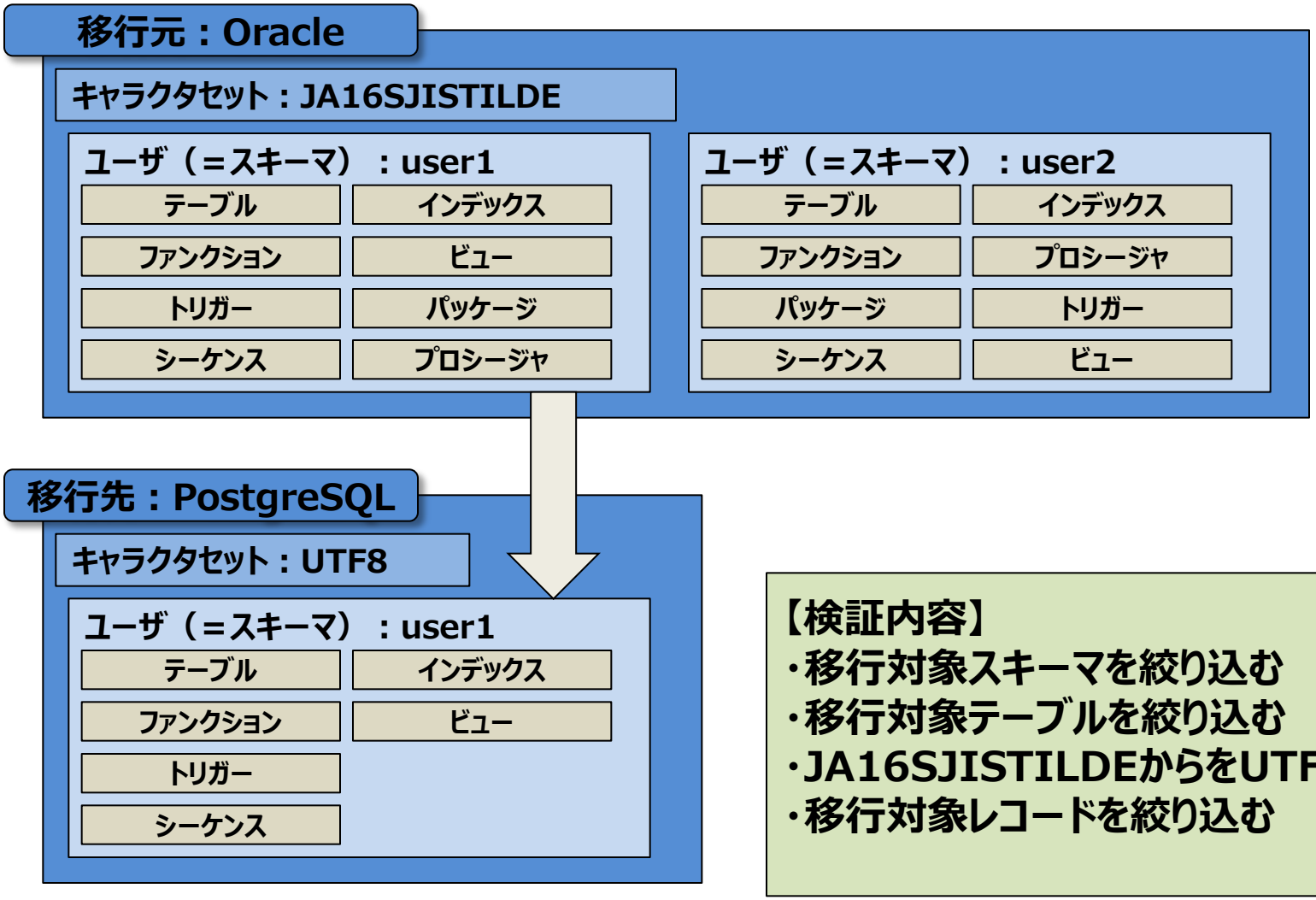
検証準備(3 of 3)

ビュー、ファンクション、トリガー、パッケージ、プロシージャ

※以下は全てエクスポート対象である。

スキーマ	オブジェクト名	オブジェクトタイプ				
		ビュー	ファンクション	トリガー	パッケージ	プロシージャ
user1	view001	○				
	view002	○				
	func001		○			
	trig001			○		
	pkg001				○	
	proc001					○

検証内容



【検証内容】
・移行対象スキーマを絞り込む
・移行対象テーブルを絞り込む
・JA16SJISTILDEからをUTF8へ移行
・移行対象レコードを絞り込む

検証結果（パラメータファイルの作成）

１．パラメータファイルの作成

■共通パラメータファイル（/etc/ora2pg/common.conf）の作成

```
ORACLE_USER          system  # Oracleデータベースにsystemユーザで接続する
KEEP_PKEY_NAMES      1       # Oracle側でユーザ指定しているPK名に合わせる
FKEY_DEFERRABLE      0       # FKの遅延制約をOracleに合わせる
DROP_FKEY            1       # インポート時にFKの再作成を行う
DROP_INDEXES         1       # インポート時にインデックスの再作成を行う
DISABLE_TRIGGERS     1       # インポート時にトリガーを無効にする
PG_NUMERIC_TYPE      1       # number(p,s)をrealまたはdouble precisionに変換する
PG_INTEGER_TYPE      1       # number(p)をsmallint、integer、bigintのいずれかに変換する
DEFAULT_NUMERIC      bigint  # number(桁数指定なし)をbigintに変換する
NLS_LANG             japanese_japan.al32utf8
                             # エクスポートデータの文字コードをUTF8に設定する
BINMODE              utf8    # クライアント側の文字コードをUTF8に設定する
CLIENT_ENCODING      utf8    # クライアント側の文字コードをUTF8に設定する
NULL_EQUAL_EMPTY     1       # NULLと空文字を同一とみなす
DEBUG                1       # Ora2Pgの詳細な実行結果を出力する
```

検証⑧　検証②　検証③　検証②

■パラメータファイル（/etc/ora2pg/ora2pg.conf）の作成

```
IMPORT    /etc/ora2pg/common.conf    # 共通パラメータファイル名を指定する
```

検証①

（注意）/etc/ora2pg/ora2pg.confの最終行に記述すること

パラメータの詳細はP.56～P.60を参照

検証結果（Oracle接続環境設定）

2．Oracle接続環境設定　検証⑦

■$ORACLE_HOME/network/admin/tnsnames.oraの作成

```
tns_test =
  (DESCRIPTION =
    (ADDRESS_LIST =
      (ADDRESS = (PROTOCOL = TCP)(HOST = <ホスト名>)(PORT = <リスナーのポート番号>))
    )
    (CONNECT_DATA =
      (SERVICE_NAME = <SID>)
    )
  )
```

■/etc/ora2pg/ora2pg.confを編集

```
ORACLE_DSN      dbi:Oracle:tns_test
```

※ORACLE_USERは共通ファイルで、ORACLE_PWDはコマンドライン「-w <systemユーザのパスワード>」で設定する。

検証結果（テーブルのエクスポート(1 of 13)）

３．テーブルのエクスポート

検証④

正規表現可能

■パラメータファイル（/etc/ora2pg/ora2pg.conf）の編集
ALLOW　commenttbl.* consttbl.* datatypetbl.* indextbl.*
EXCLUDE　excludetbl.* datatypetbl003

■ora2pgの実行
ora2pg -t table -w <systemユーザのパスワード> -n user1

検証⑥

■エクスポート結果

次頁参照

検証結果（テーブルのエクスポート(2 of 13)）

テーブルのエクスポート実施結果（1 of 2）

KEEP_PKEY_NAMESに1を設定しているため、Oracleと同じPK名でエクスポートされた。
ただし、Oracle側で自動付与しているPK名はPostgreSQL側で自動付与されるため異なる。

DEFAULT句とNOT NULLを同一カラムに設定している場合、NOT NULLがエクスポートされなかった。

PK名とFK名を自動付与している場合、FKがエクスポートされなかった。

チェック制約名を自動付与している場合、チェック制約がエクスポートされなかった。

nchar(n)は、charでエクスポートされた。

long、clob、nclobはtextでエクスポートされた。

PG_INTEGER_TYPEに1を設定しているため、以下のようにエクスポートされた。
number(1)～number(4) → smallint
number(5)～number(9) → integer
number(10)～ → bigint

PG_NUMERIC_TYPEに1を設定しているため、以下のようにエクスポートされた。
～number(6,任意) → real
number(7,任意)～ → double precision

DEFAULT_NUMERICにbigintを設定しているため、number(指定なし)はbigintでエクスポートされた。

binary_float、binary_doubleはdouble precisionでエクスポートされた。

検証結果（テーブルのエクスポート(3 of 13)）

テーブルのエクスポート実施結果（2 of 2）
dateはtimestampでエクスポートされた。
timestamp with time zone、timestamp with local time zoneはtimestamp with time zoneでエクスポートされた。
interval day to secondはinterval day(2) to second(6)でエクスポートされた。
raw、blob、bfile、long rawはbyteaでエクスポートされた。
ALLOWでテーブル名を記述した場合、対象テーブルに作成されているイデックスはエクスポートされなかった。

詳細な実行結果は次頁以降を参照

検証結果（テーブルのエクスポート(4 of 13)）

【エクスポート前】

```
create table commenttbl001 (
        col1 number,
        col2 date
);
comment on table commenttbl001 is 'コメントテーブル';
comment on column commenttbl001.col1 is 'コメントテーブルのカラム１';
comment on column commenttbl001.col1 is 'コメントテーブルのカラム２';
```

【エクスポート後】

```
create table commenttbl001 (
        col1 bigint,
        col2 timestamp
);
COMMENT ON TABLE commenttbl001 IS E'コメントテーブル';
COMMENT ON COLUMN commenttbl001.col1 IS E'コメントテーブルのカラム１';
COMMENT ON COLUMN commenttbl001.col2 IS E'コメントテーブルのカラム２';
```

検証結果（テーブルのエクスポート(5 of 13)）

【エクスポート前】

```
create table consttbl001 (
        col1 number,
        col2 varchar2(30)
);
alter table consttbl001 add constraint tbl001_pk primary key (col1);

create table consttbl002 (
        col1 number,
        col2 number,
        col3 varchar2(100)
);
alter table consttbl002 add constraint tbl002_fk foreign key (col2) references consttbl001 (col1) on
delete set null;
```

【エクスポート後】

検証⑧ KEEP_PKEY_NAMESに1を設定

```
CREATE TABLE consttbl001 (
        col1 bigint NOT NULL,
        col2 varchar(30)
);
ALTER TABLE consttbl001 ADD CONSTRAINT tbl001_pk PRIMARY KEY (col1);
CREATE TABLE consttbl002 (
        col1 bigint,
        col2 bigint,
        col3 varchar(100)
);
ALTER TABLE consttbl002 ADD CONSTRAINT tbl002_fk FOREIGN KEY (col2) REFERENCES consttbl001
(col1) ON DELETE SET NULL NOT DEFERRABLE INITIALLY IMMEDIATE;
```

FKEY_DEFERRABLEに0を設定 検証⑧

検証結果（テーブルのエクスポート(6 of 13)）

【エクスポート前】

```
create table consttbl003 (
        col1 number default 1 not null,
        col2 varchar2(10)
);
alter table consttbl003 add constraint tbl003_chk1 check (col1 in (0,1));
```

検証⑨

【エクスポート後】

```
CREATE TABLE consttbl003 (
        col1 bigint DEFAULT 1,
        col2 varchar(10)
);
ALTER TABLE consttbl003 ADD CONSTRAINT tbl003_chk1 CHECK (col1 in (0,1));
```

DEFAULT句と同時に指定した
NOT NULL制約がエクスポートされない

検証結果（テーブルのエクスポート(7 of 13)）

【エクスポート前】

```
create table consttbl004 (
        col1 number,
        col2 varchar2(30)
);
alter table consttbl004 add constraint tbl004_uq unique (col1);
```

【エクスポート後】

```
CREATE TABLE consttbl004 (
        col1 bigint,
        col2 varchar(30)
);
ALTER TABLE consttbl004 ADD CONSTRAINT tbl004_uq UNIQUE (col1);
```

検証結果（テーブルのエクスポート(8 of 13)）

【エクスポート前】

```
create table consttbl005 (
        col1 number,
        col2 varchar2(30)
);
alter table consttbl005 add primary key (col1);

create table consttbl006 (
        col1 number,
        col2 number,
        col2 varchar2(100)
);
alter table consttbl006 add foreign key (col2) references consttbl005(col1) on delete set null;
```

PK名を自動付与

FK名を自動付与

【エクスポート後】

```
CREATE TABLE consttbl005 (
        col1 bigint NOT NULL,
        col2 varchar(30)
);
ALTER TABLE consttbl005 ADD PRIMARY KEY (col1);

CREATE TABLE consttbl006 (
        col1 bigint,
        col2 bigint,
        col3 varchar(100)
);
```

検証⑨

PKはエクスポートされる

検証⑨

FKはエクスポートされない

検証結果（テーブルのエクスポート(9 of 13)）

【エクスポート前】

```
create table consttbl007 (
        col1 number,
        col2 varchar2(10)
);
alter table consttbl007 add check (col1 in (0,1));
```

チェック制約名を自動付与

【エクスポート後】

```
CREATE TABLE consttbl007 (
        col1 bigint,
        col2 varchar(10)
);
```

検証⑨

チェック制約がエクスポートされない

検証結果（テーブルのエクスポート(10 of 13)）

【エクスポート前】

```
create table datatypetbl001
(
        vc1     varchar2(10),
        nvc1    nvarchar2(20),
        ch1     char(20),
        nch1    nchar(30),
        long1   long,
        clob1   clob,
        nclob1  nclob,
        num1    number(1),
        num2    number(2),
        num3    number(3),
        num4    number(4),
        num5    number(5),
        num6    number(6),
        num7    number(7),
        num8    number(8),
        num9    number(9),
        num10   number(10),
        num11   number(11),
        num12   number(20),
        num13   number(30),
        num14   number,
        num15   number(2,1),
        num16   number(3,1),
        num17   number(3,2),
        num18   number(4,1),
```

検証⑨

numericでエクスポートされない（精度低下）

【エクスポート後】

```
CREATE TABLE datatypetbl001
(
        vc1 varchar(10),
        nvc1 varchar(20),
        ch1 char(20),
        nch1 char,
        long1 text,
        clob1 text,
        nclob1 text,
        num1 smallint,
        num2 smallint,
        num3 smallint,
        num4 smallint,
        num5 integer,
        num6 integer,
        num7 integer,
        num8 integer,
        num9 integer,
        num10 bigint,
        num11 bigint,
        num12 bigint,
        num13 bigint,
        num14 bigint,
        num15 real,
        num16 real,
        num17 real,
        num18 real,
```

検証⑨

文字数がエクスポートされない

検証②

PG_INTEGER_TYPEに1を設定

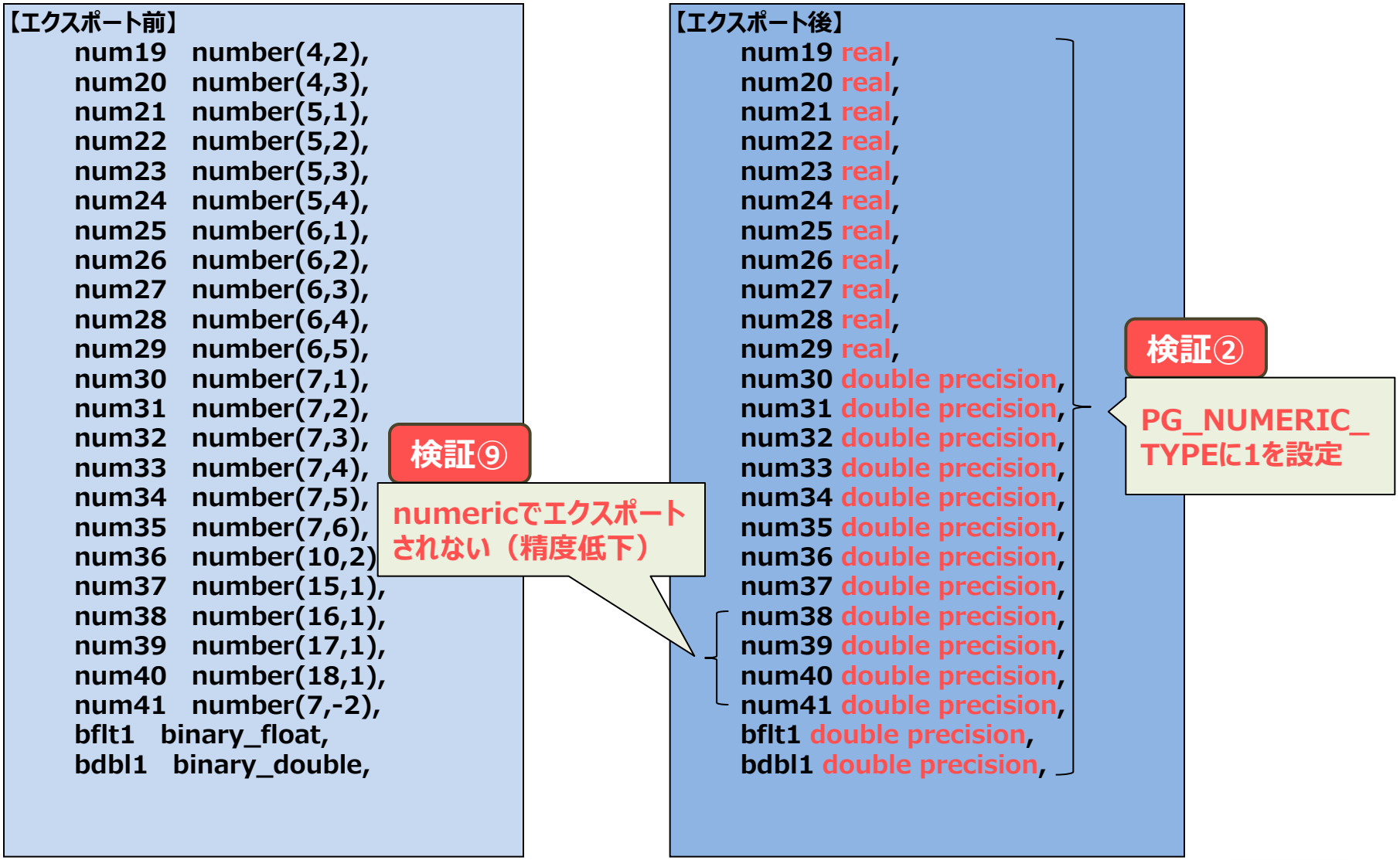
検証②

DEFAULT_NUMERICにbigintを設定

検証②

PG_NUMERIC_TYPEに1を設定

検証結果（テーブルのエクスポート(11 of 13)）



検証結果（テーブルのエクスポート(12 of 13)）

【エクスポート前】

```
        date1   date,
        tstmp1  timestamp,
        tstmp2  timestamp with time zone,
        tstmp3  timestamp with local time zone,
        iytom1  interval year(4) to month,
        idtos1  interval day to second,
        raw1    raw(100),
        blob1   blob,
        bfile1  bfile
);

create table datatypetbl002
(
        lraw1 long raw
);
```

【エクスポート後】

```
        date1 timestamp,
        tstmp1 timestamp,
        tstmp2 timestamp with time zone,
        tstmp3 timestamp with time zone,
        iytom1 INTERVAL YEAR(4) TO MONTH,
        idtos1 INTERVAL DAY(2) TO SECOND(6),
        raw1 bytea,
        blob1 bytea,
        bfile1 bytea
);

CREATE TABLE datatypetbl002
(
        lraw1 bytea
);
```

検証結果（テーブルのエクスポート(13 of 13)）

【エクスポート前】

```
create table indextbl001 (col1 char(10),col2 number);
create index tbl001_idx1 on indextbl001(upper(col1);

create table indextbl002 (col1 number,col2 varchar2(10),col3 date);
create index tbl002_idx1 on indextbl002(col1,col2 asc);

create table indextbl003 (col1 number,col2 varchar2(10),col3 date);
create index tbl003_idx1 on indextbl003(col1,col2 desc);

create table indextbl004 (col1 number,col2 varchar2(10),col3 date);
create unique index tbl004_idx1 on indextbl004(col1,col2 desc);
```

【エクスポート後】

```
CREATE TABLE indextbl001 (col1 char(10),col2 bigint);
CREATE TABLE indextbl002 (col1 bigint,col2 varchar(10),col3 timestamp);
CREATE TABLE indextbl003 (col1 bigint,col2 varchar(10),col3 timestamp);
CREATE TABLE indextbl004 (col1 bigint,col2 varchar(10),col3 timestamp);
```

検証⑨ テーブルに作成されているインデックスがエクスポートされない

検証結果（インデックスのエクスポート(1 of 2)）

４．インデックスのエクスポート

■パラメータファイル（/etc/ora2pg/ora2pg.conf）の編集
ALLOW　indextbl.* tbl.*idx.*
※indextbl.*はテーブル名、tbl.*idx*はインデックス名

■ora2pgの実行
ora2pg -t table -w <systemユーザのパスワード> -n user1

■エクスポート結果
以下参照

インデックスのエクスポート実施結果

ALLOWにテーブル名とインデックス名を指定した場合、テーブルとそこに作成されているインデックスがエクスポートされた。

全て昇順インデックスとしてエクスポートされた。

詳細な実行結果は次頁以降を参照

検証結果（インデックスのエクスポート(2 of 2)）

【エクスポート前】

```
create table indextbl001 (col1 char(10),col2 number);
create index tbl001_idx1 on indextbl001(upper(col1);

create table indextbl002 (col1 number,col2 varchar2(10),col3 date);
create index tbl002_idx1 on indextbl002(col1,col2 asc);

create table indextbl003 (col1 number,col2 varchar2(10),col3 date);
create index tbl003_idx1 on indextbl003(col1 desc,col2 desc);

create table indextbl004 (col1 number,col2 varchar2(10),col3 date);
create unique index tbl004_idx1 on indextbl004(col1,col2 desc);
```

【エクスポート後】

```
CREATE TABLE indextbl001 (col1 char(10),col2 bigint);
CREATE INDEX tbl001_idx1 ON indextbl001 (upper(col1));

CREATE TABLE indextbl002 (col1 bigint,col2 varchar(10),col3 timestamp);
CREATE INDEX tbl002_idx1 ON indextbl002 (col1,col2);

CREATE TABLE indextbl003 (col1 bigint,col2 varchar(10),col3 timestamp);
CREATE INDEX tbl003_idx1 ON indextbl003 (col1,col2);

CREATE TABLE indextbl004 (col1 bigint,col2 varchar(10),col3 timestamp);
CREATE UNIQUE INDEX tbl004_idx1 ON indextbl004 (col1,col2);
```

検証⑨

全て昇順インデックスとしてエクスポートされる

検証結果（シーケンスのエクスポート(1 of 2)）

５．シーケンスのエクスポート

■/etc/ora2pg/ora2pg.confを編集
※スキーマ（＝ユーザ）user1にあるシーケンスを全てエクスポートするため、ALLOWパラメータおよびEXCLUDEパラメータには何も設定しない。

■ora2pgの実行
ora2pg -t sequence -w <systemユーザのパスワード> -n user1

■エクスポート結果
以下参照

シーケンスのエクスポート実施結果

CACHE句は、そのままエクスポートされた。

PostgreSQL側のシーケンス最大値9223372036854775807に満たない場合でもNO MAXVALUEでエクスポートされた。

詳細な実行結果は次頁以降を参照

検証結果（シーケンスのエクスポート(2 of 2)）

【エクスポート前】
create sequence seq001 increment by 1 start with 1 nomaxvalue nominvalue nocycle order nocache;
create sequence seq002 increment by 1 start with 1 maxvalue 999999999999999999999999999 nominvalue cycle cache 20;
create sequence seq003 increment by 1 start with 1 maxvalue 9223372036854775807 nominvalue cycle cache 20;
create sequence seq004 increment by 1 start with 1 maxvalue 90000000000000000 nominvalue nocycle nocache;
create sequence seq005 increment by 1 start with 1 maxvalue 9223372036854775000 nominvalue nocycle nocache;
create sequence seq006 increment by 1 start with 1 maxvalue 922337203685477580 0 nominvalue nocycle nocache;
create sequence seq007 increment by 1 start with 1 maxvalue 10000 nominvalue cycle cache 20;
create sequence seq008 increment by 1 start with 1 maxvalue 10000 nominvalue cycle nocache;

【エクスポート後】
CREATE SEQUENCE seq001 INCREMENT 1 MINVALUE 1 NO MAXVALUE START 1 CACHE 1;
CREATE SEQUENCE seq002 INCREMENT 1 MINVALUE 1 NO MAXVALUE START 1 CACHE 20 CYCLE;
CREATE SEQUENCE seq003 INCREMENT 1 MINVALUE 1 NO MAXVALUE START 1 CACHE 20 CYCLE;
CREATE SEQUENCE seq004 INCREMENT 1 MINVALUE 1 MAXVALUE 90000000000000000 START 1 CACHE 1;
CREATE SEQUENCE seq005 INCREMENT 1 MINVALUE 1 MAXVALUE 9223372036854775000 START 1 CACHE 1;
CREATE SEQUENCE seq006 INCREMENT 1 MINVALUE 1 NO MAXVALUE START 1 CACHE 1;

最大値9223372036854775807に満たない場合でもNO MAXVALUEでエクスポートされた

CREATE SEQUENCE seq007 INCREMENT 1 MINVALUE 1 MAXVALUE 10000 START 1 CACHE 20 CYCLE;
CREATE SEQUENCE seq008 INCREMENT 1 MINVALUE 1 MAXVALUE 10000 START 1 CACHE 1 CYCLE;

検証結果（ファンクションのエクスポート）

６．ファンクションのエクスポート
■ora2pgの実行
ora2pg -t function -w <systemユーザのパスワード> -n user1

■エクスポート結果
以下参照

【エクスポート前】
```
create or replace function func001(
        v_dept_id in char)
        return varchar2 as
        v_dept_name varchar2(10);
begin
        select dept_name into v_dept_name from
dept
        where dept_id = v_dept_id;
        return v_dept_name;
end func001;
/
```

【エクスポート後】
```
CREATE OR REPLACE FUNCTION user1.func001 (
        v_dept_id in char)
         RETURNS varchar AS $body$
DECLARE
        v_dept_name varchar(10);
BEGIN
        select dept_name into v_dept_name from
dept
        where dept_id = v_dept_id;
        return v_dept_name;
END;
$body$
LANGUAGE PLPGSQL;
```

検証結果（ビューのエクスポート）

7．ビューのエクスポート

■ora2pgの実行

```
# ora2pg -t view -w <systemユーザのパスワード> -n user1
```

■エクスポート結果

【エクスポート前】

```
create or replace view view001 as select col1,col2 from viewtbl001 where col2 is null;
create or replace view view002 as select col1,col2 from viewtbl001 where col2 is not null;
```

【エクスポート後】

```
CREATE OR REPLACE VIEW view001 (col1, col2) AS SELECT col1,col2 FROM viewtbl001
where coalesce(col2::text, '') = '';
CREATE OR REPLACE VIEW view002 (col1, col2) AS SELECT col1,col2 FROM viewtbl001
where (col2 IS NOT NULL AND col2::text <> '');
```

検証②

NULL_EQUAL_EMPTYに1を設定

検証結果（トリガーのエクスポート）

８．トリガーをエクスポート
■ora2pgの実行
ora2pg -t trigger -w <systemユーザのパスワード> -n user1

■エクスポート結果
以下参照

【エクスポート前】
```
create or replace trigger trig001
  before insert or update or delete on
test_tbl for each row
begin
  if inserting then
    update cnt_tbl set i_cnt = i_cnt + 1;
  elsif updating then
    update cnt_tbl set u_cnt = u_cnt +
1;
  else
    update cnt_tbl set d_cnt = d_cnt +
1;
  end if;
end;
/
```

【エクスポート後】
```
DROP TRIGGER IF EXISTS test_trg ON test_tbl CASCADE;
CREATE OR REPLACE FUNCTION trigger_fct_trg001 ()
RETURNS trigger AS $BODY$
begin
  if TG_OP = 'INSERT' then
    update cnt_tbl set i_cnt = i_cnt + 1;
  elsif TG_OP = 'UPDATE' then
    update cnt_tbl set u_cnt = u_cnt + 1;
  else
    update cnt_tbl set d_cnt = d_cnt + 1;
  end if;
RETURN NEW;
end
$BODY$
 LANGUAGE 'plpgsql';

CREATE TRIGGER trig001
       BEFORE INSERT OR UPDATE OR DELETE ON test_tbl
FOR EACH ROW
       EXECUTE PROCEDURE trigger_fct_trig001();
```

検証結果（パッケージのエクスポート(1 of 2)）

９．パッケージをエクスポート

■ora2pgの実行
ora2pg -t package -w <systemユーザのパスワード> -n user1

■エクスポート結果
次頁参照

パッケージのエクスポート実施結果

ファンクションの中で使用される変数の定義がエクスポートされなかった。
→DECLARE var VARCHAR(50)がエクスポートされなかった。

詳細な実行結果は次頁以降を参照

検証結果（パッケージのエクスポート(2 of 2)）

【エクスポート前】

```
create or replace package pkg001 as
function f_kake (
  a in number,
  b in number
) return number;
procedure f_con (
  str1 in varchar,
  str2 in varchar
);
end;
/
create or replace package body pkg001
as
  var varchar2(50);
  function f_kake(a in number, b in
number) return number
  is
    d number;
  begin
    d := a * b;
    return d;
  end f_kake;
  procedure f_con(str1 in varchar, str2 in
varchar)
  is
  begin
    var := str1 || str2;
    dbms_output.put_line(var);
  end f_con;
end pkg001;
/
```

【エクスポート後】

```
DROP SCHEMA IF EXISTS pkg001 CASCADE;
CREATE SCHEMA pkg001;
var VARCHAR2(50);

CREATE OR REPLACE FUNCTION pkg001.f_kake (a IN
bigint, b IN bigint)  RETURNS bigint AS $body$
DECLARE
    d bigint;
BEGIN
    d := a * b;
    RETURN d;
  END;
$body$
LANGUAGE PLPGSQL;

CREATE OR REPLACE FUNCTION pkg001.f_con (str1 IN text,
str2 IN text)
   RETURNS VOID AS $body$

BEGIN
    var := str1 || str2;
    RAISE NOTICE '%', var;
  END;

$body$
LANGUAGE PLPGSQL;
```

変数varの定義がエクスポートされない

検証結果（プロシージャのエクスポート）

１０．プロシージャをエクスポート
■ora2pgの実行
ora2pg -t procedure -w <systemユーザのパスワード> -n user1

■エクスポート結果
以下参照

【エクスポート前】
```
CREATE OR REPLACE PROCEDURE
proc001(aa IN NUMBER,bb OUT VARCHAR)
IS
  CURSOR c1 IS SELECT * FROM tbtb001 WHERE
col1 = aa;
BEGIN
  DBMS_OUTPUT.PUT_LINE('start...');
  FOR rec IN c1 LOOP
    DBMS_OUTPUT.PUT_LINE(rec.col1);
  END LOOP;
  DBMS_OUTPUT.PUT_LINE('...end');
  bb := 'OK';
EXCEPTION
  WHEN others THEN
    DBMS_OUTPUT.PUT_LINE('..error..');
    bb := 'NG';
END;
/
```

【エクスポート後】
```
CREATE OR REPLACE FUNCTION
proc001 (aa IN bigint,bb OUT text)
 RETURNS text AS $body$
DECLARE
  c1 CURSOR FOR SELECT * FROM tbtb001 WHERE
col1 = aa;
BEGIN
  RAISE NOTICE 'start...';
  FOR rec IN c1 LOOP
    RAISE NOTICE '%', rec.col1;
  END LOOP;
  RAISE NOTICE '...end';
  bb := 'OK';
EXCEPTION
  WHEN others THEN
    RAISE NOTICE '..error..';
    bb := 'NG';
END;
$body$
LANGUAGE PLPGSQL;
```

検証結果（レコードのエクスポート）

１１．レコードのエクスポート

■/etc/ora2pg/ora2pg.confを編集

```
ALLOW copy.*
WHERE copytbl001[col1 > 5] copytbl002[col1 < 3 or col1 > 7]
```

検証⑤

■ora2pgの実行

```
# ora2pg -t copy -w <systemユーザのパスワード> -n user1
```

■エクスポート結果

```
COPY copytbl001 (col1,col2) FROM STDIN;
6    6ABCDE
7    7ABCDE
8    8ABCDE
9    9ABCDE
10   10ABCDE
¥.
COMMIT;

COPY copytbl002 (col1,col2) FROM STDIN;
1    1ABCDE
2    2ABCDE
8    8ABCDE
9    9ABCDE
10   10ABCDE
¥.
COMMIT;
```

検証結果（インプットファイルを変換）

１２．インプットファイルを変換
■インプットファイルの作成
　既に使用したファンクション、パッケージ、プロシージャを作成するためのPL/SQL文を
ファイルに保管
・ファンクション：create_func001.sql
・パッケージ：create_pkg001.sql
・プロシージャ：create_proc001.sql

■ora2pgの実行
ora2pg -t function –i create_func001.sql
ora2pg -t package –i create_pkg001.sql
ora2pg -t procedure –i create_proc001.sql

■エクスポート結果
既に検証済みのファンクション、パッケージ、プロシージャと同様のエクスポート結果が得られた。

アジェンダ

・会社紹介
・OracleからPostgreSQLへのデータ移行の主な課題
・Ora2Pgの仕様確認
・Ora2Pgを利用したデータ移行の課題解決策検討
・検証環境
・検証結果総括
・今後の課題と取り組み

【付録】

-検証結果
-手作りSQL
-Ora2Pgのパラメータについて

手作りSQL(1 of 5)

No	Oracle側で実行する手作りSQLの処理内容	
1	DEFAULT句とNOT NULLを同時指定している箇所を抽出	P.29
2	FK名を自動付与している箇所の抽出	P.31
3	チェック制約名を自動付与している箇所の抽出	P.32
4	ncharを使用しているカラム名と文字数の抽出	P.33
5	numeric型の検討が必要なカラム名の抽出	P.33～34
6	エクスポート対象テーブルに作成されているインデックス名の抽出	P.36
7	降順インデックスを使用しているカラム名の抽出	P.38

実際のSQLは次頁以降に記載

手作りSQL(2 of 5)

DEFAULT句とNOT NULLを同時指定している箇所の抽出

（必要な理由）
DEFAULT句とNOT NULLを同一カラムに指定している場合、NOT NULLがエクスポートされないため。

```
SQL> select table_name,column_name,nullable,data_default
     from user_tab_columns
     where nullable='N' and data_default is not null;
```

FK名を自動付与している箇所の抽出

（必要な理由）
FK名を自動付与している場合、FK定義がエクスポートされないため。

```
SQL> select * from user_constraints
     where constraint_type='R'
     and generated='GENERATED NAME';
```

手作りSQL(3 of 5)

チェック制約名を自動付与している箇所の抽出

（必要な理由）
　チェック制約名を自動付与している場合、チェック制約の定義がエクスポートされないため。チェック制約にはNOT NULL制約も含まれるが、それを除外した形で抽出するためにはsearch_condition列（LONG型）に抽出条件を指定する必要がある。

```
SQL> set serveroutput on
DECLARE
   v_tablename user_constraints.table_name%type;
   v_searchcondition VARCHAR2(4000);
BEGIN
   for c1 in (select table_name,search_condition from user_constraints
              where constraint_type='C' and generated='GENERATED NAME')
 LOOP
   v_tablename := c1.table_name;
   v_searchcondition := c1.search_condition;
   IF v_searchcondition like '%IS NOT NULL' THEN null;
   ELSE dbms_output.put_line(v_tablename||'   '||v_searchcondition);
   END IF;
 END LOOP;
END;
/
```

手作りSQL(4 of 5)

ncharを使用しているカラム名と文字数の抽出

（必要な理由）
ncharをエクスポートした場合、サイズがエクスポートされないため。

```
SQL> select table_name,column_name,data_type,char_length
     from user_tab_columns
     where data_type = 'NCHAR'
     order by table_name;
```

※Oracleのncharはnls_length_semanticsの設定如何にかかわらず、キャラクターベースである。

numeric型の検討が必要なカラム名の抽出

（必要な理由）
精度低下を考慮せずにdouble precision、bigintにエクスポートされるため。
double precisionは15桁精度、bigintの範囲は-9223372036854775808から9223372036854775807
である。

```
SQL> select table_name,column_name,data_type,data_precision,data_scale
     from user_tab_columns
     where data_type='NUMBER'
     and ((nvl(data_precision,0) = 0 and nvl(data_scale,0) = 0)
          or (data_precision > 15 and nvl(data_scale,0) > 0)
          or (data_precision > 18 and nvl(data_scale,0) = 0));
```

手作りSQL(5 of 5)

エクスポート対象テーブルに作成されているインデックス名の抽出

（必要な理由）
　エクスポート時にテーブルとそこに作成されたインデックスをエクスポートする場合、インデックス名もALLOWで指定する必要があるため。

```
SQL> select table_name,index_name
     from user_indexes
     where table_name = 'エクスポート対象テーブル名';
```

降順インデックスを使用しているカラム名の抽出

（必要な理由）
　降順インデックスは全て昇順インデックスとしてエクスポートされないため。

```
SQL> select index_name,table_name,column_name,descend
     from user_ind_columns
     where descend='DESC'
     order by index_name,table_name;
```

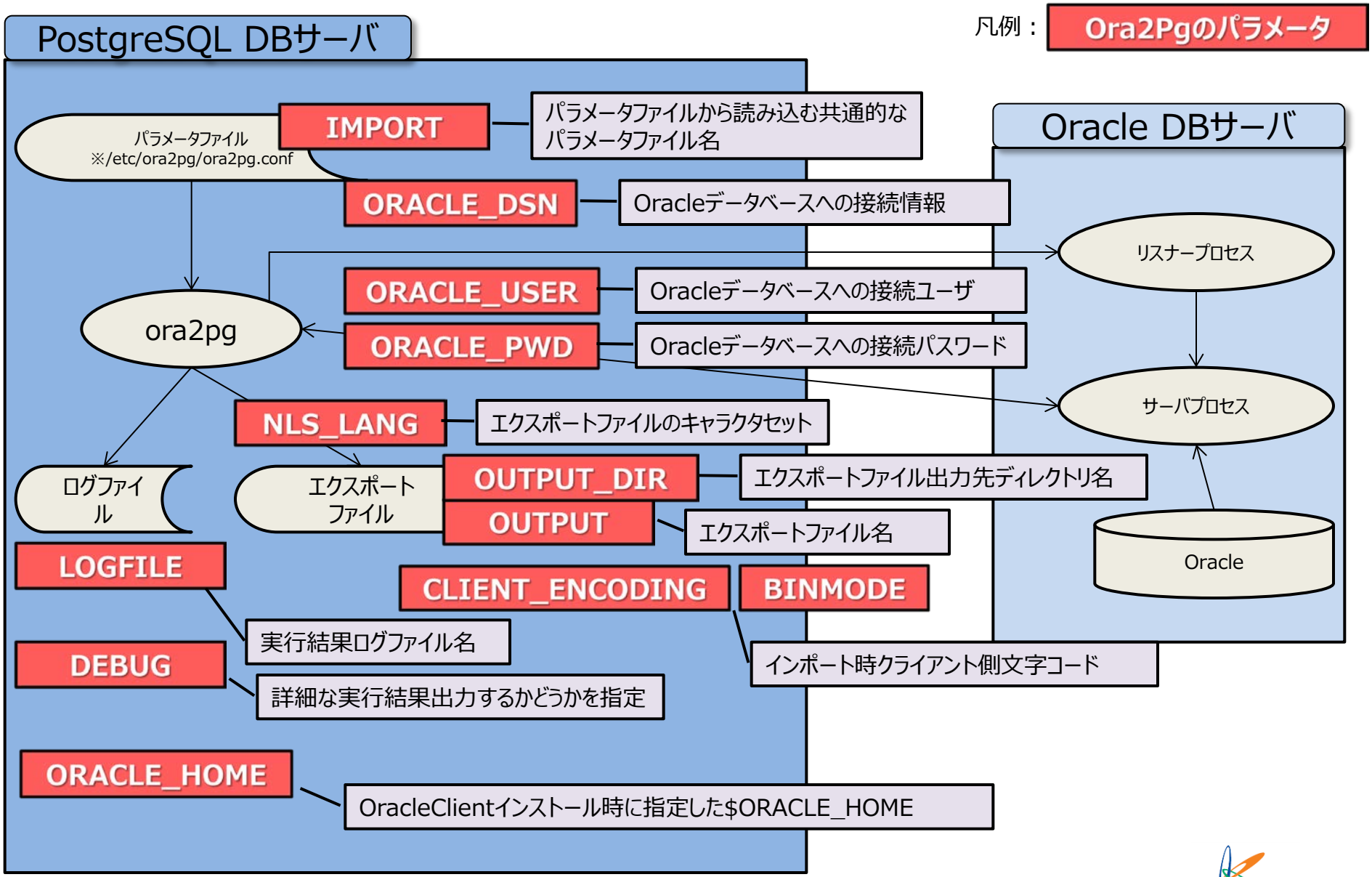
アジェンダ

- 会社紹介
- OracleからPostgreSQLへのデータ移行の主な課題
- Ora2Pgの仕様確認
- Ora2Pgを利用したデータ移行の課題解決策検討
- 検証環境
- 検証結果総括
- 今後の課題と取り組み

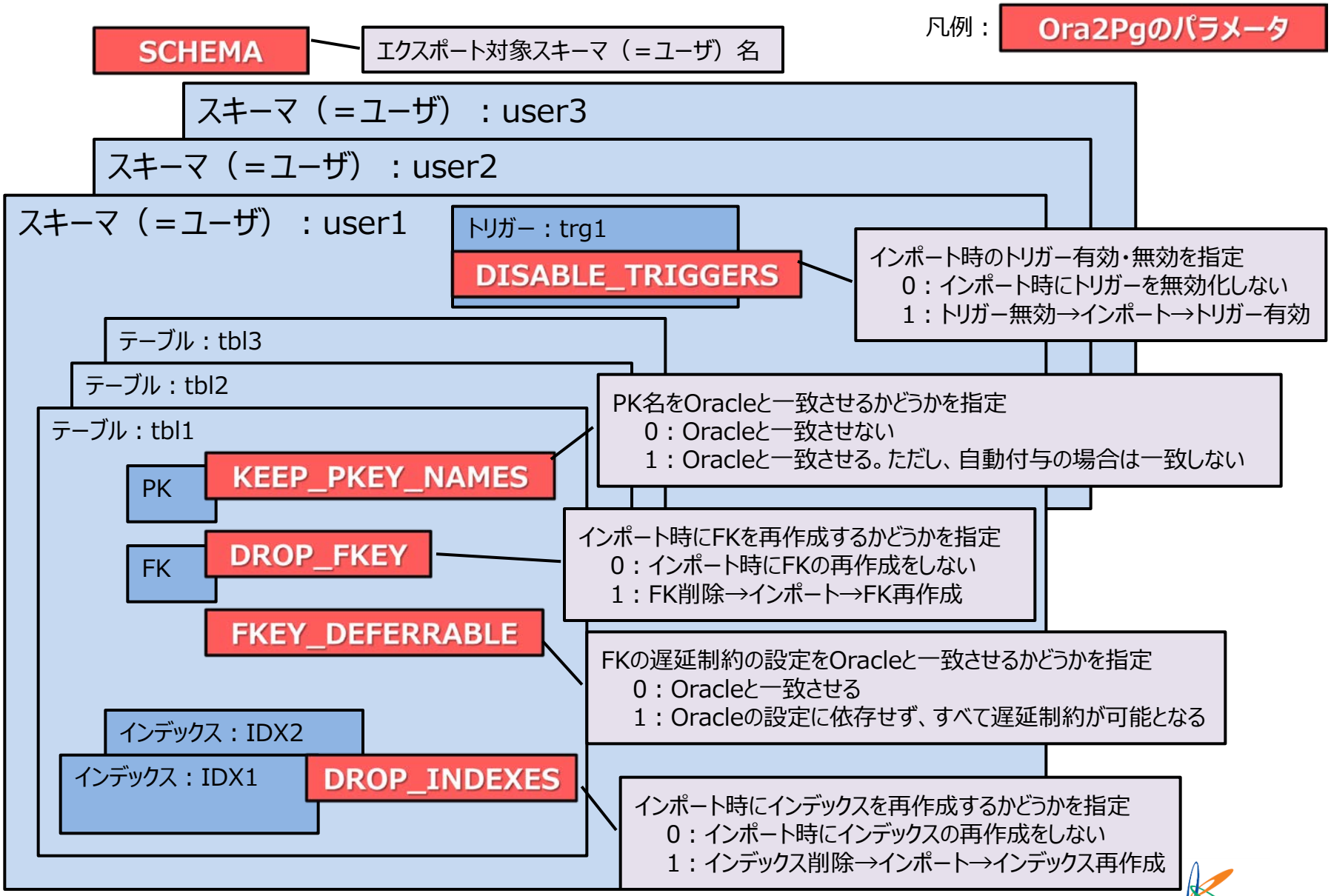
【付録】

- -検証結果
- -手作りSQL
- **-Ora2Pgのパラメータについて**

Ora2Pgのパラメータについて（1 of 5）

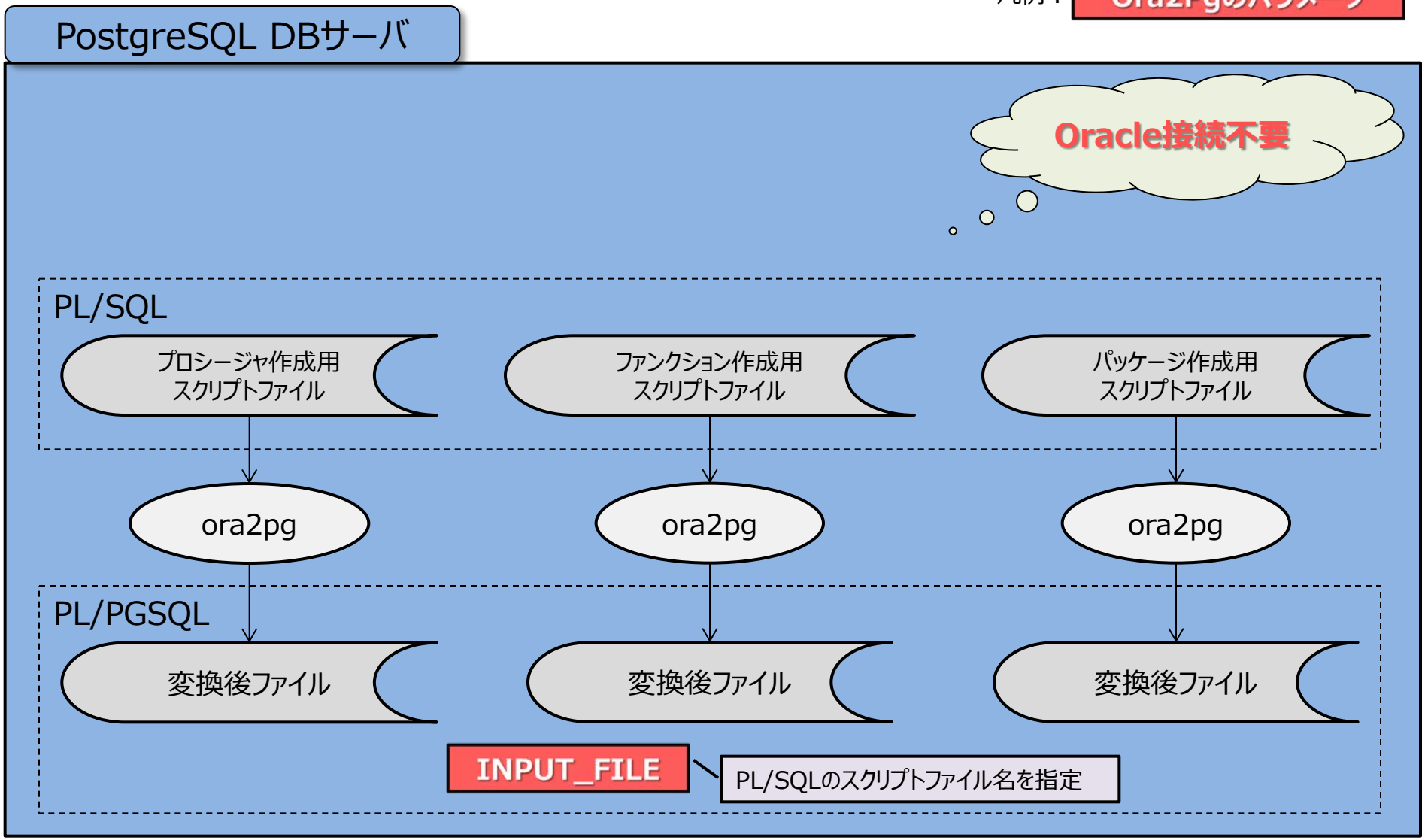


Ora2Pgのパラメータについて（2 of 5）



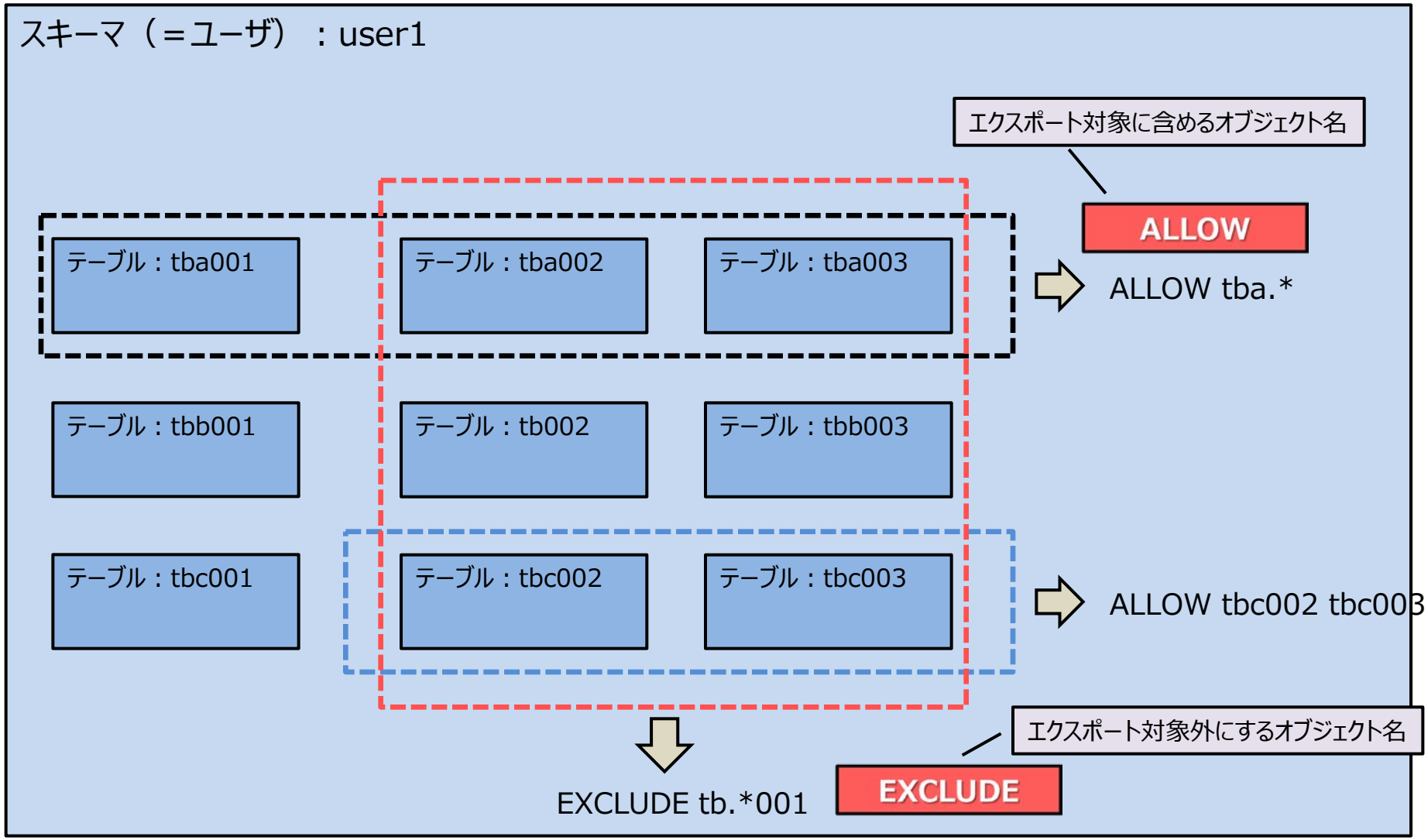
Ora2Pgのパラメータについて（3 of 5）

凡例：Ora2Pgのパラメータ



Ora2Pgのパラメータについて（4 of 5）

凡例：Ora2Pgのパラメータ



Ora2Pgのパラメータについて（5 of 5）

凡例：Ora2Pgのパラメータ

テーブル：tbl1

カラム名	データ型
col1	number(1)
col2	number(5,3)
col3	number

PG_INTEGER_TYPE

Oracleのnumber(p)をPostgreSQLの何型へ移行するかを指定
- 0：numeric(p)型へ移行
- 1：smallint型またはinteger型またはbigint型へ移行

PG_NUMERIC_TYPE

Oracleのnumber(p,s)をPostgreSQLの何型へ移行するかを指定
- 0：decimal(p,s)型へ移行
- 1：real型またはdouble precision型へ移行

DEFAULT_NUMERIC

PG_INTEGER_TYPEパラメータに1を設定している場合に有効となり、Oracleのnumber（桁数指定なし）をPostgreSQLの何型へ移行するかを指定
（例）bigint

WHERE

エクスポート対象データの絞り込み条件を指定
（例）tbl1[col1 > 100 and col3 = 10] tbl2[col1 > 1000]

ご清聴、ありがとうございました。