2013.11.08 Tokyo

PostgreSQL で オープンデータ XML / JSON を使いこなす

国府田 諭（埼玉大学環境科学研究センター 研究員）

発表内容

❶ まず使ってみよう（入門）

❷ どう活用できるか（事例）

❸ オープンデータ統計の今後

キーワード

- 統計 API
- XML / JSON
- Postgres 9.3
- メッシュデータ
- PostGIS
- R , PL/R

※ 紙数の都合で
本資料は❶のみ

❶ まず使ってみよう（次世代統計利用システム）

仮登録

↓

本登録

↓

アプリ ID 取得

↓

準備完了

国勢調査など、各種統計データを
XML / JSON でダウンロードできる

❶ まず使ってみよう（仮登録）

仮登録

↓

本登録

↓

アプリ ID 取得

↓

準備完了

http://statdb.nstac.go.jp/

> 利用登録・ログイン

> 仮登録

メールアドレスを送信

❶ まず使ってみよう（本登録）

仮登録

↓

本登録

↓

アプリ ID 取得

↓

準備完了

仮登録完了メールが来る

> 本登録用 Web ページアドレスあり

> 当日中にアクセス

必要事項を記入、送信

❶ まず使ってみよう（アプリケーション ID 取得）

仮登録

↓

本登録

↓

アプリ ID 取得

↓

準備完了

http://statdb.nstac.go.jp/

> 利用登録・ログイン

> ログイン

> 利用者情報変更／削除

> アプリケーション ID の取得

❶ まず使ってみよう（統計 API の URL）

http://statdb.nstac.go.jp/api/1.0b/app/

- getStatsList？パラメータ　統計表一覧（XML）
- getStatsData？パラメータ　XML データ
- json/getStatsData？パラメータ　JSON データ

他にもあるが、表一覧 XML と JSON データだけでも足りる

❶ まず使ってみよう（統計表一覧のパラメータ例）

全件（メッシュデータ等を除く）

…/getStatsList ? appId= アプリ ID

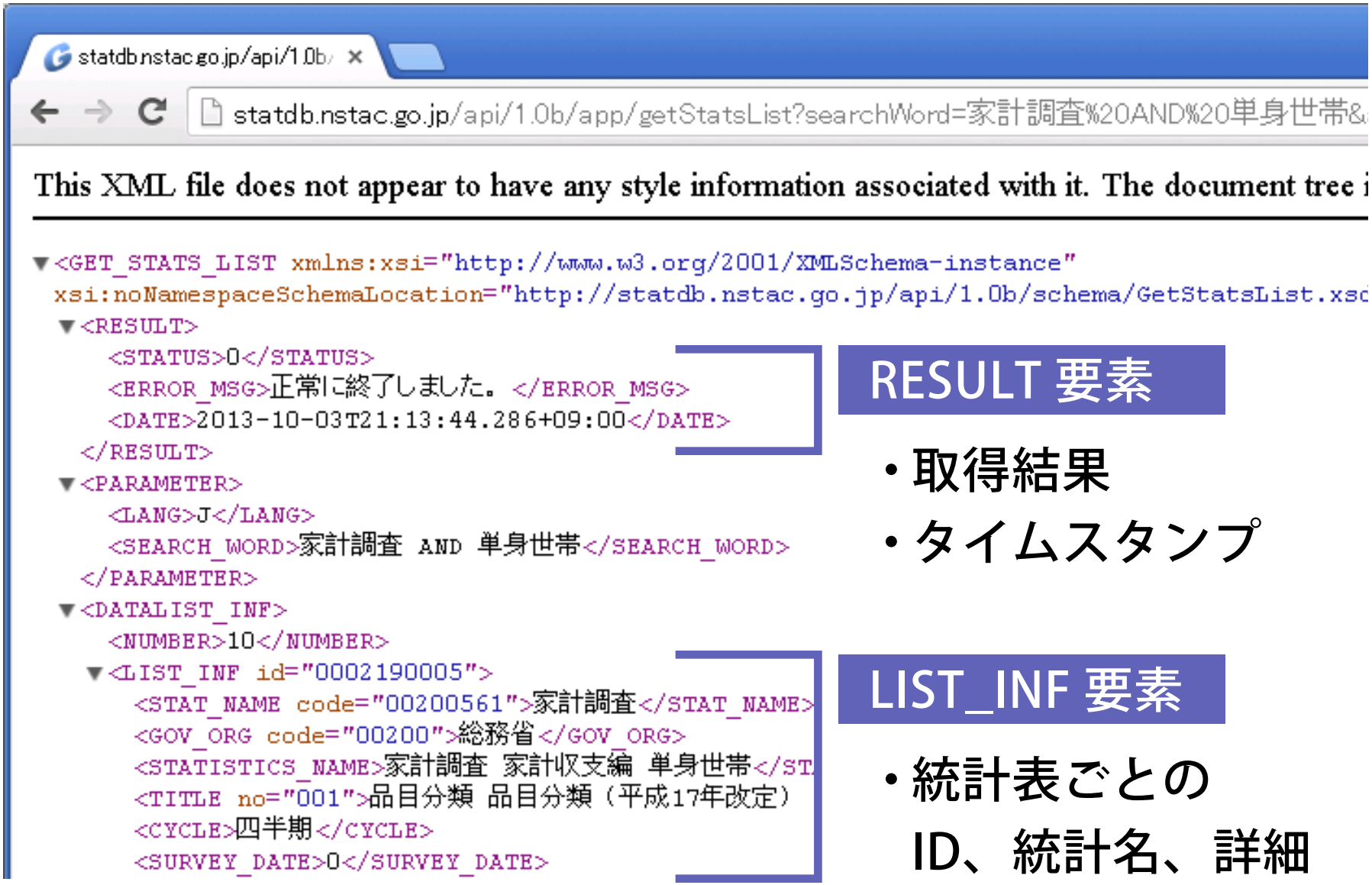
全件（メッシュ、小地域データ）

…/getStatsList ? appId= アプリ ID & searchKind=2

キーワード検索

…/getStatsList ? appId= アプリ ID
& searchWord= 家計調査 AND 単身世帯

❶ まず使ってみよう（統計表一覧 XML の例）

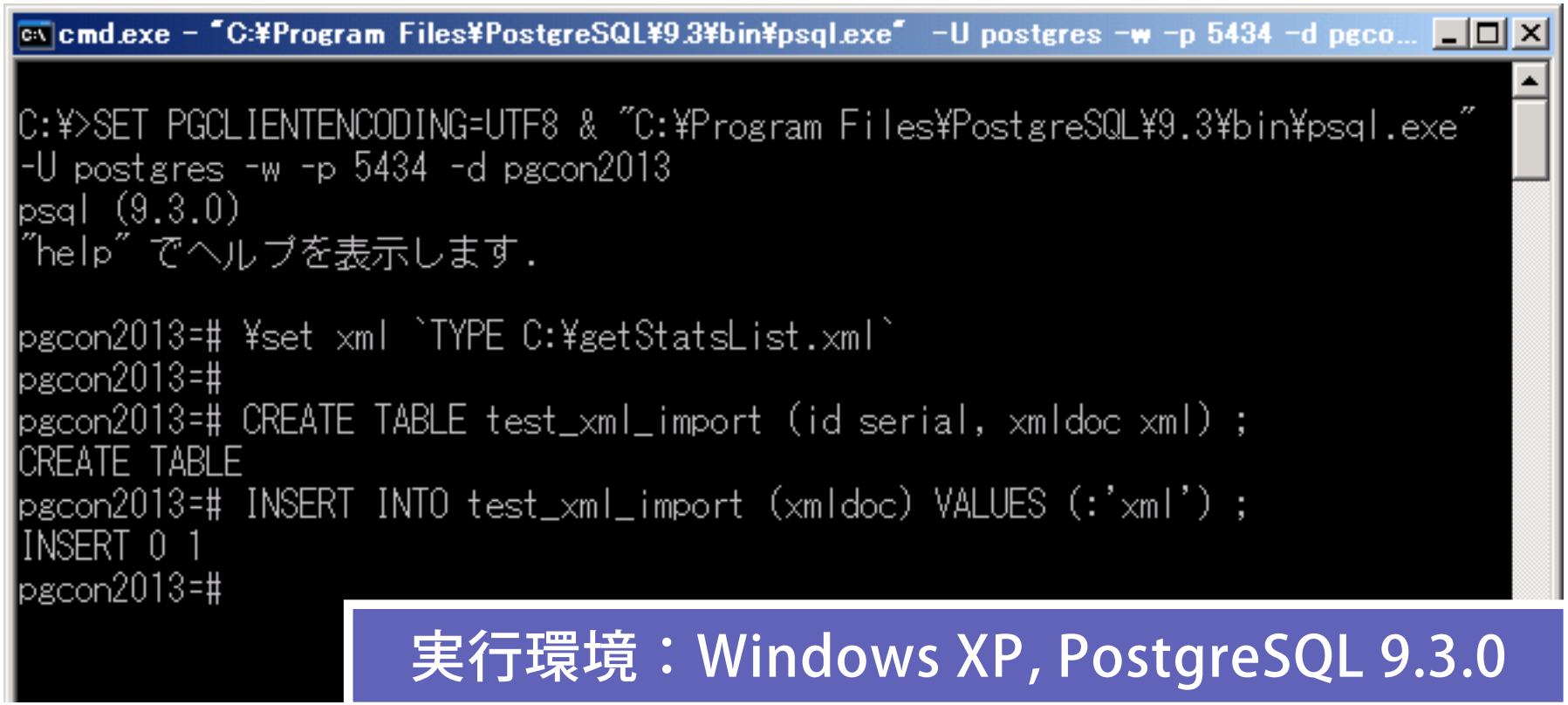


❶ まず使ってみよう（XML を Postgres に格納）

(1) XML を適当な場所に保存

(2) psql を起動、XML の中身を変数に入れる

(3) SQL の変数差し替えでテーブルに投入



実行環境：Windows XP, PostgreSQL 9.3.0

❶ まず使ってみよう（XML の内容を確認）

```
SELECT unnest(xpath('//LIST_INF', xmldoc))
FROM test_xml_import WHERE id = 1 ;
```

xpath で LIST_INF 要素を全て取得
> 配列として返る
> unnest 関数で一行ずつに

Data Output | Explain | Messages | Hi

	unnest xml
1	<LIST_INF id="0002190005"> <STAT_NAME code="00200561">家計調査</STAT_NAME> <GOV_ORG code="00200">総務省</GOV_ORG> <STATISTICS_NAME>家計調査 家計収支編 単身世帯</STATISTICS_NAME> <TITLE no="001">品目分類 品目分類（平成17年改定）（金額）</TITLE> (...)
2	<LIST_INF id="0002190006"> <STAT_NAME code="00200561">家計調査</STAT_NAME> <GOV_ORG code="00200">総務省</GOV_ORG> <STATISTICS_NAME>家計調査 家計収支編 単身世帯</STATISTICS_NAME>
3	<LIST_INF id="0003000773">
4	<LIST_INF id="0003000797">
5	<LIST_INF id="0003015134">
6	<LIST_INF id="0003016656">
7	<LIST_INF id="0002190004">

❶ まず使ってみよう（XML から統計表一覧を作成）

```
SELECT xpath('@id', unnest) id
        , xpath('//STATISTICS_NAME/text()', unnest) sname
        , xpath('//TITLE/text()', unnest) title
        , xpath('//CYCLE/text()', unnest) "cycle"
FROM (
        SELECT unnest(xpath('//LIST_INF', xmldoc))
        FROM test_xml_import WHERE id = 1
) foo ;
```

Data Output | Explain | Messages | History

	id xml[]	sname xml[]	title xml[]	cycle xml[]
1	{0002190005}	{"家計調査 家計収支編 単身世帯"}	{"品目分類 品目分類（平成17年改定）（金額）"}	{四半期}
2	{0002190006}	{"家計調査 家計収支編 単身世帯"}	{"品目分類 品目分類（平成17年改定）（金額）"}	{年次}
3	{0003000773}	{"家計調査 家計収支編 単身世帯"}	{"用途分類 用途分類（総数） 全国"}	{四半期}
4	{0003000797}	{"家計調査 家計収支編 単身世帯"}	{"用途分類 用途分類（総数） 全国"}	{年次}
5	{0003015134}	{"家計調査 家計収支編 単身世帯"}	{"品目分類 品目分類（平成22年改定）（金額）"}	{四半期}
6	{0003016656}	{"家計調査 家計収支編 単身世帯"}	{"品目分類 品目分類（平成22年改定）（金額）"}	{年次}
7	{0002190004}	{"家計調査 家計収支編 単身世帯"}	{"用途分類 用途分類（年齢階級別）"}	{年次}
8	{0003000795}	{"家計調査 家計収支編 単身世帯"}	{"用途分類 用途分類（年齢階級別）"}	{四半期}
9	{0003000798}	{"家計調査 家計収支編 単身世帯"}	{"用途分類 用途分類（総数） 都市階級・地方"}	{年次}
10	{0003005368}	{"家計調査 家計収支編 単身世帯"}	{"用途分類 用途分類（総数） 都市階級・地方"}	{四半期}

❶ まず使ってみよう（統計データ JSON を取得）

http://statdb.nstac.go.jp/api/1.0b/app/json/
getStatsData ? appId= アプリ ID
&statsDataId=

統計表一覧 XML
→ LIST_INF 要素 → id 属性

Data Output | Explain | Messages | History

	id xml[]	sname xml[]	title xml[]	cycle xml[]
1	{0002190005}	{"家計調査　家計収支編　単身世帯"}	{"品目分類　品目分類（平成17年改定）（金額）"}	{四半期}
2	{0002190006}	{"家計調査　家計収支編　単身世帯"}	{"品目分類　品目分類（平成17年改定）（金額）"}	{年次}
3	{0003000773}	{"家計調査　家計収支編　単身世帯"}	{"用途分類　用途分類（総数）　全国"}	{四半期}
4	{0003000797}	{"家計調査　家計収支編　単身世帯"}	{"用途分類　用途分類（総数）　全国"}	{年次}
5	{0003015134}	{"家計調査　家計収支編　単身世帯"}	{"品目分類　品目分類（平成22年改定）（金額）"}	{四半期}
6	{0003016656}	{"家計調査　家計収支編　単身世帯"}	{"品目分類　品目分類（平成22年改定）（金額）"}	{年次}
7	{0002190004}	{"家計調査　家計収支編　単身世帯"}	{"用途分類　用途分類（年齢階級別）"}	{年次}
8	{0003000795}	{"家計調査　家計収支編　単身世帯"}	{"用途分類　用途分類（年齢階級別）"}	{四半期}
9	{0003000798}	{"家計調査　家計収支編　単身世帯"}	{"用途分類　用途分類（総数）　都市階級・地方"}	{年次}
10	{0003005368}	{"家計調査　家計収支編　単身世帯"}	{"用途分類　用途分類（総数）　都市階級・地方"}	{四半期}

❶ まず使ってみよう（cURL + psql でデータ取得）

cURL … Web データを取得できる CLI ツール

⑴ cURL を準備、psql 起動
⑵ cURL でデータ取得、変数に入れる
⑶ SQL の変数差し替えで INSERT（XML と同じ）

```
pgcon2013=# ¥set json `"C:/curl-7.32.0-devel-mingw32/bin/curl.exe" http://statdb
.nstac.go.jp/api/1.0b/app/json/getStatsData?appId=
^&statsDataId=0003000797`
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100  306k    0  306k    0     0   479k      0 --:--:-- --:--:-- --:--:--  479k
pgcon2013=# CREATE TABLE test_json_import (id serial, jsondoc json) ;
CREATE TABLE
pgcon2013=# INSERT INTO test_json_import (jsondoc) VALUES (:'json') ;
INSERT 0 1
pgcon2013=#
```

Windows のコマンドラインは & を ^ でエスケープ

❶ まず使ってみよう（PG 9.3 の COPY 新機能）

PostgreSQL 9.3 で追加された COPY … PROGRAM

＋

cURL など Web データを取得できる CLI ツール

⬇

SQL 一文で、統計データ JSON をテーブルに追加

```
COPY 投入先テーブル名 （列名）
FROM PROGRAM '……/curl "http://……' ;
```

```
pgcon2013=# COPY test_json_import (jsondoc) FROM PROGRAM '"R:/curl-7.32.0-devel-
mingw32/bin/curl.exe" http://statdb.nstac.go.jp/api/1.0b/app/json/getStatsData?a
ppId=                                    ^&statsDataId=0003000797' ;
COPY 1
pgcon2013=#
```

Windows のコマンドラインは & を ^ でエスケープ

❶ まず使ってみよう（DO 文でデーター括取得）

```
DO $$
DECLARE cmd text ; sid text ;
BEGIN
FOR sid IN
        SELECT unnest(xpath('//LIST_INF/@id', xmldoc)) :: text
        FROM test_xml_import WHERE id = 1
LOOP
        RAISE NOTICE '%', sid ;
        cmd = '"R:/curl-7.32.0-devel-mingw32/bin/curl.exe"'
                || ' http://statdb.nstac.go.jp/api/1.0b/app/json/getStatsData?appId='
                || '^&statsDataId=' || sid ;
        EXECUTE 'COPY test_json_import (jsondoc) FROM PROGRAM ''' || cmd  || '''' ;
END LOOP ;
END $$ ;
```

先に取得した統計表一覧 XML
＞各表の statsDataId でループ

Data Output | Explain | **Messages** | History

```
NOTICE:  0002190005
NOTICE:  0002190006
NOTICE:  0003000773
NOTICE:  0003000797
NOTICE:  0003015134
NOTICE:  0003016656
NOTICE:  0002190004
NOTICE:  0003000795
NOTICE:  0003000798
NOTICE:  0003005368
Query returned successfully with no result in 24547 ms.
```

COPY … FROM PROGRAM で
cURL 起動、JSON 取得

❶ まず使ってみよう（統計表データ JSON の内容）


```
{ "GET_STATS_DATA":{
    "RESULT":{"STATUS":0,"ERROR_MSG":"正常に終了しました。","DATE":"2013-10-04T16:24:41.185+09:00"},
    "PARAMETER":{"LANG":"J","STATS_DATA_ID":"0003000797","DATA_FORMAT":"J","START_POSITION":1,"METAGET_FLG":"Y"}
    "STATISTICAL_DATA":{
      "TABLE_INF":{"@id":"0003000797","STAT_NAME":{"@code":"00200561","$":"家計調査"}, ……},
      "CLASS_INF":{
        "CLASS_OBJ":[
          {"@id":"tab","@name":"表章項目",
            "CLASS":{"@code":"01","@name":"金額","@level":""}},
          {"@id":"cat01","@name":"用途分類",
            "CLASS":[
              {"@code":"001","@name":"世帯数分布（抽出率調整）","@level":"1","@unit":"一万分比"},
              {"@code":"002","@name":"集計世帯数","@level":"1","@unit":"世帯"}, ……
              ]}, ……
          }
        ]},
      "DATA_INF":{
        "NOTE":[
          {"@char":"***","$":"調査又は集計していないもの"},{"@char":"-","$":"該当数字がないもの"},……],
        "VALUE":[
          {"@tab":"01","@cat01":"001","@cat02":"21","@area":"00000","@time":"2012000000","@unit":"一万分比","$":"10000"},
          {"@tab":"01","@cat01":"001","@cat02":"21","@area":"00000","@time":"2011000000","@unit":"一万分比","$":"10000"}, ……
        ]}
} } }
```


取得時の全情報
- ステータス、日時
- 指定した統計表 ID
- パラメータ

CLASS_INF 要素

メタデータ（データ項目名、単位など）
＝ 統計表の内容・構造を集約

DATA_INF → VALUE 要素

統計表データ全体の配列

❶ まず使ってみよう（統計表一覧 XML と結合）

```
SELECT bar.id json_id
        , xpath('@id', foo.x) id
        , xpath('//STATISTICS_NAME/text()', foo.x) sname
        , xpath('//TITLE/text()', foo.x) title
        , xpath('//CYCLE/text()', foo.x) "cycle"
FROM (
        SELECT unnest(xpath('//LIST_INF', xmldoc)) x
        FROM test_xml_import WHERE id = 1
) foo
LEFT JOIN test_json_import bar
ON (xpath('@id', foo.x))[1] :: text = jsondoc #>> '{GET_STATS_DATA, STATISTICAL_DATA, TABLE_INF, @id}'
ORDER BY 1 ;
```

演算子 #>> '{ key, …, … }'

先頭からキーをたどり、
値を文字型で返す

	json_id integer	id xml[]	sname xml[]	title xml[]	cycle xml[]
1	1	{0002190005}	{"家計調査 家計収支編 単身世帯"}	{"品目分類 品目分類（平成17年改定） （金額）"}	{四半期}
2	2	{0002190006}	{"家計調査 家計収支編 単身世帯"}	{"品目分類 品目分類（平成17年改定） （金額）"}	{年次}
3	3	{0003000773}	{"家計調査 家計収支編 単身世帯"}	{"用途分類 用途分類（総数） 全国"}	{四半期}
4	4	{0003000797}	{"家計調査 家計収支編 単身世帯"}	{"用途分類 用途分類（総数） 全国"}	{年次}
5	5	{0003015134}	{"家計調査 家計収支編 単身世帯"}	{"品目分類 品目分類（平成22年改定） （金額）"}	{四半期}
6	6	{0003016656}	{"家計調査 家計収支編 単身世帯"}	{"品目分類 品目分類（平成22年改定） （金額）"}	{年次}
7	7	{0002190004}	{"家計調査 家計収支編 単身世帯"}	{"用途分類 用途分類（年齢階級別）"}	{年次}
8	9	{0003000798}	{"家計調査 家計収支編 単身世帯"}	{"用途分類 用途分類（総数） 都市階級・地方"}	{年次}
9		{0003005368}	{"家計調査 家計収支編 単身世帯"}	{"用途分類 用途分類（総数） 都市階級・地方"}	{四半期}
10		{0003000795}	{"家計調査 家計収支編 単身世帯"}	{"用途分類 用途分類（年齢階級別）"}	{四半期}

未結合 = JSON 取得失敗（データ件数 10 万超の統計表）

❶ まず使ってみよう（データ本体のテーブル化）

```
SELECT json_array_elements(
        jsondoc #> '{GET_STATS_DATA, STATISTICAL_DATA, DATA_INF, VALUE}'
)
FROM test_json_import
WHERE id = 4
LIMIT 1 ;
```

テーブル化する統計表を選択、VALUE のキー名を見る

	json_array_elements json
1	{"@tab":"01","@cat01":"001","@cat02":"21","@area":"00000","@time":"2012000000","@unit":"一万分比","$":"10000"}

```
CREATE TYPE test_json AS (
        "@tab" text,"@cat01" text,"@cat02" text,"@area" text, "@time" text
        ,"@unit" text, "$" text) ;

CREATE TABLE test_json_values (
        tab text, cat01 text, cat02 text, area text, "time" text
        , unit text, val text) ;

INSERT INTO test_json_values
SELECT (json_populate_recordset(NULL :: test_json,
        jsondoc #> '{GET_STATS_DATA, STATISTICAL_DATA, DATA_INF, VALUE}'
)).*
FROM test_json_import WHERE id = 4 ;
```

キー名に合わせ、一時的にデータ型を定義

テーブル作成

json_populate_recordset で投入

❶ まず使ってみよう（データ本体の内容）

	tab text	cat01 text	cat02 text	area text	time text	unit text	val text
1	01	001	21	00000	2012000000	一万分比	10000
2	01	001	21	00000	2011000000	一万分比	10000
3	01	001	21	00000	2010000000	一万分比	10000
4	01	001	21	00000	2009000000	一万分比	10000
5	01	001	21	00000	2008000000	一万分比	10000
6	01	001	21	00000	2007000000	一万分比	10000
7	01	001	22	00000	2012000000	一万分比	4200
8	01	001	22	00000	2011000000	一万分比	4419
9	01	001	22	00000	2010000000	一万分比	4358
10	01	001	22	00000	2009000000	一万分比	4637
11	01	001	22	00000	2008000000	一万分比	4824
12	01	001	22	00000	2007000000	一万分比	4727
13	01	001	23	00000	2012000000	一万分比	5800
14	01	001	23	00000	2011000000	一万分比	5581
15	01	001	23	00000	2010000000	一万分比	5642
16	01	001	23	00000	2009000000	一万分比	5363
17	01	001	23	00000	2008000000	一万分比	5176
18	01	001	23	00000	2007000000	一万分比	5273
19	01	002	21	00000	2012000000	世帯	702
20	01	002	21	00000	2011000000	世帯	689
21	01	002	21	00000	2010000000	世帯	708

Unix | Ln 38, Col 1, Ch 1121 | 32 chars | 2835 rows.

cat01, 02, … データ項目（最重要）

tab　表章項目
area　地域
time　時点
unit　単位
（JSON のキー "$"）数値

- データ本体の一行が一つの数値を示す
- データ項目、地域等の結合に工夫が必要

❶ まず使ってみよう（CLASS_OBJ 要素を見る）

```
WITH a AS (
        SELECT json_array_elements(jsondoc #>
                '{ GET_STATS_DATA, STATISTICAL_DATA, CLASS_INF
                , CLASS_OBJ }') j
        FROM test_json_import WHERE id = 4
), b AS (
        SELECT j ->> '@id' class_obj_id
                , j ->> '@name' class_obj_name
                , j #>> '{ CLASS }' j2
        FROM a
)
SELECT * FROM b ;
```

CLASS_OBJ 要素の ID

⬇

VALUE 要素のキー名

=

データテーブルの各列

	class_obj_id text	class_obj_name text	j2 text
1	tab	表章項目	{"@code":"01","@name":"金額","@level"
2	cat01	用途分類	[{"@code":"001","@name":"世帯数分布（
3	cat02	世帯区分	[{"@code":"21","@name":"単身世帯","@l
4	area	地域区分	{"@code":"00000","@name":"全国","@lev
5	time	時間軸（年次）	[{"@code":"2012000000","@name":"2012年

CLASS 要素の処理、やや面倒

❶ まず使ってみよう（CLASS 要素を展開）

```
WITH a AS (
        SELECT json_array_elements(jsondoc #>
                '{GET_STATS_DATA, STATISTICAL_DATA, CLASS_INF, CLASS_OBJ}') j
        FROM test_json_import WHERE id = 4
), b AS (
        SELECT j ->> '@id' class_obj_id
                , j ->> '@name' class_obj_name
                , j #>> '{ CLASS }' j2
        FROM a
), c AS (
        SELECT class_obj_id, class_obj_name
                , CASE WHEN j2 LIKE '{%' THEN '[' || j2 || ']'
                        ELSE j2 END :: json
        FROM b
), d AS (
        SELECT class_obj_id, class_obj_name, json_array_elements(j2) j3
        FROM c
)
SELECT class_obj_id, class_obj_name
        , j3 ->> '@code' class_code, j3 ->> '@name' class_name
FROM d ;
```

ここまで、前頁と同じ

配列でない JSON を配列化し、
json_array_elements で共通処理

❶ まず使ってみよう（メタデータのテーブル化）

前頁クエリの結果

class_obj_id text	class_obj_name text	class_code text	class_name text
tab	表章項目	01	金額
cat01	用途分類	001	世帯数分布（抽出率
cat01	用途分類	002	集計世帯数
cat01	用途分類	009	世帯主の年齢
cat01	用途分類	010	有業者比率
cat01	用途分類	011	持家率
cat01	用途分類	015	家賃・地代を支払
cat01	用途分類	018	受取
cat01	用途分類	019	実収入
cat01	用途分類	020	経常収入
cat01	用途分類	021	勤め先収入
cat01	用途分類	024	定期収入
cat01	用途分類	025	臨時収入
cat01	用途分類	026	賞与
cat01	用途分類	030	事業・内職収入
cat01	用途分類	031	家賃収入
cat01	用途分類	032	他の事業収入
cat01	用途分類	033	内職収入
cat01	用途分類	034	農林漁業収入
cat01	用途分類	035	他の経常収入
cat01	用途分類	036	財産収入
cat01	用途分類	037	社会保障給付

メタデータで必要な
項目を確定すれば
⬇
データテーブルから
抽出・集計できる

データテーブル

tab text	cat01 text	cat02 text	area text	time text	unit text	val text
01	001	21	00000	2012000000	一万分比	10000
01	001	21	00000	2011000000	一万分比	10000
01	001	21	00000	2010000000	一万分比	10000
01	001	21	00000	2009000000	一万分比	10000
01	001	21	00000	2008000000	一万分比	10000
01	001	21	00000	2007000000	一万分比	10000
01	001	22	00000	2012000000	一万分比	4200
01	001	22	00000	2011000000	一万分比	4419
01	001	22	00000	2010000000	一万分比	4358
01	001	22	00000	2009000000	一万分比	4637
01	001	22	00000	2008000000	一万分比	4824

❶ まず使ってみよう（統計データの抽出例）

```
SELECT left("time", 4) "year", val, unit FROM test_json_values
WHERE
cat01 = (SELECT class_code
         FROM test_json_meta
         WHERE (class_obj_id, class_name) = ('cat01', '食料'))
AND cat02 = (SELECT class_code
         FROM test_json_meta
         WHERE (class_obj_id, class_name) = ('cat02', '（単身）勤労者世帯')) ;
```

単身・勤労者の
1ヶ月の食料費
（全国平均）

	year text	val text	unit text
1	2012	42291	円
2	2011	42930	円
3	2010	43905	円
4	2009	42628	円
5	2008	45808	円
6	2007	43595	円

CLASS_OBJ 別の
テーブルを作れば
通常の JOIN に

'（単身）勤労者以外の世帯'

2012	34417	円
2011	33617	円
2010	32328	円
2009	33184	円
2008	33405	円
2007	33181	円

❶ まず使ってみよう（統計データの計算例）

```
WITH m1 AS (
        SELECT * FROM test_json_meta
        WHERE class_obj_id = 'cat01'
                AND class_name IN ('消費支出', '食料')
), m2 AS (
        SELECT * FROM test_json_meta
        WHERE class_obj_id = 'cat02'
                AND class_name IN ('（単身）勤労者世帯', '（単身）勤労者以外の世帯')
), d0 AS (
        SELECT m1.class_name cat01, m2.class_name cat02
                , left(time, 4) "year", val :: float, unit
        FROM m1, m2, test_json_values v
        WHERE (m1.class_code, m2.class_code) = (v.cat01, v.cat02)
), d1 AS (
        SELECT * FROM d0 WHERE cat01 = '消費支出'
), d2 AS (
        SELECT * FROM d0 WHERE cat01 = '食料'
)
SELECT "year", cat02
        , round((d2.val / d1.val * 100) :: numeric, 1) " 消費支出に占める食料(%)"
FROM d1 JOIN d2 USING (cat02, "year") ;
```

前者に占める、後者の割合を

この世帯別に出す

データ本体を結合、抽出

割合の分母、分子

世帯と年で JOIN、計算

❶ まず使ってみよう（統計データの計算例の結果）

	year text	cat02 text	消費支出に占める食料(%) numeric
1	2012	（単身）勤労者世帯	24.9
2	2011	（単身）勤労者世帯	23.5
3	2010	（単身）勤労者世帯	24.1
4	2009	（単身）勤労者世帯	23.0
5	2008	（単身）勤労者世帯	23.5
6	2007	（単身）勤労者世帯	22.8
7	2012	（単身）勤労者以外の世帯	23.4
8	2011	（単身）勤労者以外の世帯	23.4
9	2010	（単身）勤労者以外の世帯	22.0
10	2009	（単身）勤労者以外の世帯	23.1
11	2008	（単身）勤労者以外の世帯	22.3
12	2007	（単身）勤労者以外の世帯	22.2

時系列がもっと多いデータも、前頁のクエリだけで対応可

例えば家計調査では：1985 年以降の月次、四半期データあり

❶ まず使ってみよう（前頁のクエリを単純化）

```
WITH m1 AS (
        SELECT * FROM test_json_meta WHERE class_obj_id = 'cat01'
), m2 AS (
        SELECT * FROM test_json_meta WHERE class_obj_id = 'cat02'
), d0 AS (
        SELECT m1.class_name cat01, m2.class_name cat02
                , left(time, 4) "year", val :: float, unit
        FROM m1, m2, test_json_values v
        WHERE (m1.class_code, m2.class_code) = (v.cat01, v.cat02)
), d1 AS (
        SELECT * FROM d0 WHERE cat01 = '消費支出'
), d2 AS (
        SELECT * FROM d0 WHERE cat01 = '食料'
)
SELECT "year", cat02
        , round((d2.val / d1.val * 100) :: numeric, 1) " 消費支 | に占める食料(%)"
FROM d1 JOIN d2 USING (cat02, "year")
ORDER BY cat02 DESC, year DESC ;
```

ここを指定するだけ

cat02（世帯の種類）と
time（年）ごとに
必要な数値を算出できる

❶ まず使ってみよう（データ項目には階層がある）

```
SELECT class_obj_id, class_obj_name
        , j3 ->> '@code' class_code, j3 ->> '@name' class_name
        , j3 ->> '@level' class_level
FROM d ;
```

P.22 クエリの最後に追加

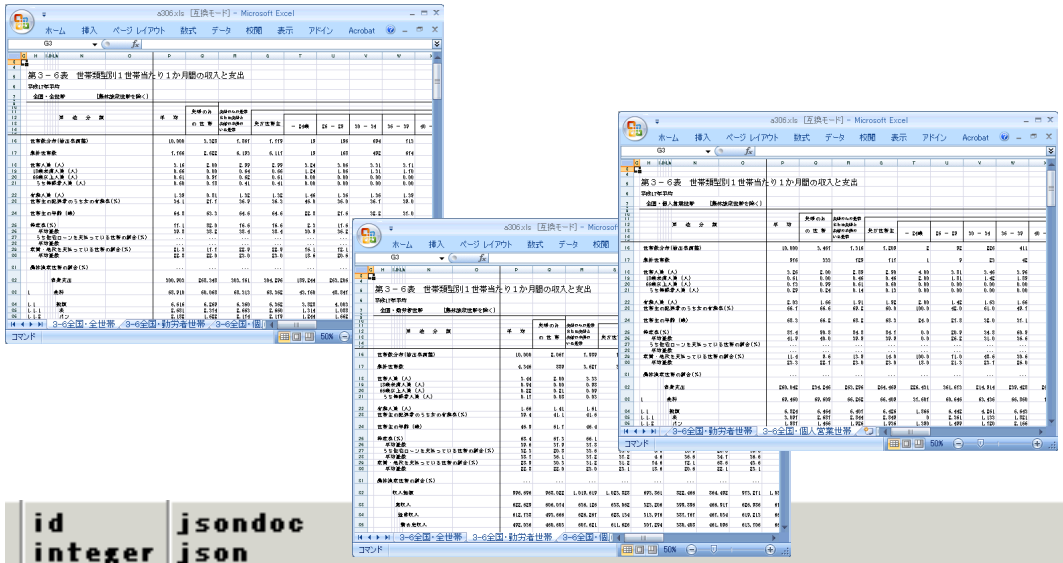
	class text	class_obj_na text	class_cod text	class_name text	class_level text
40	cat01	用途分類	055	実収入以外の受取りと	3
41	cat01	用途分類	056	繰入金	2
42	cat01	用途分類	057	支払	1
43	cat01	用途分類	058	実支出	2
44	cat01	用途分類	059	消費支出	3
45	cat01	用途分類	060	食料	4
46	cat01	用途分類	061	穀類	5
47	cat01	用途分類	062	米	6
48	cat01	用途分類	063	パン	6
49	cat01	用途分類	064	めん類	6
50	cat01	用途分類	065	他の穀類	6
51	cat01	用途分類	066	魚介類	5
52	cat01	用途分類	067	生鮮魚介	6
53	cat01	用途分類	068	塩干魚介	6
54	cat01	用途分類	069	魚肉練製品	6

先の例は

食料（レベル 4）

消費支出（レベル 3）

❶ まず使ってみよう（まとめ）



Excel 統計表のグループがテーブルの各 1 行になる

id integer	jsondoc json
1	{"GET_STATS_DATA":{"RESULT":{"STATUS":0,"ERROR_MSG":"正常に終了しました。","DATE":"2013-10-04T16:24:34.554
2	{"GET_STATS_DATA":{"RESULT":{"STATUS":0,"ERROR_MSG":"正常に終了しました。","DATE":"2013-10-04T16:24:37.314
3	{"GET_STATS_DATA":{"RESULT":{"STATUS":0,"ERROR_MSG":"正常に終了しました。","DATE":"2013-10-04T16:24:39.425
4	{"GET_STATS_DATA":{"RESULT":{"STATUS":0,"ERROR_MSG":"正常に終了しました。","DATE":"2013-10-04T16:24:41.185
5	{"GET_STATS_DATA":{"RESULT":{"STATUS":0,"ERROR_MSG":"正常に終了しました。","DATE":"2013-10-04T16:24:45.444

- 取得時ステータス、メタデータ、データ本体が一つ
  - ➡ 管理が容易。使用時は別途テーブル化
- メタデータから統計の内容・構造を読み取る
  - ➡ 項目間の階層に注意。一般的な DB とは違う

❶ まず使ってみよう（落ち穂拾い：XML と JSON）

- 現状、統計表一覧は XML のみ提供
- 10 万件超のデータをリクエストすると
  - ➡ JSON：エラーだけ返す
  - ➡ XML　：10 万件まで返し、
    続きのリクエスト用パラメータ NEXT_KEY を付加
- XML は改行あり。JSON は改行・タブなし
  - ➡ JSON：COPY コマンドでのインポートが簡単

紙数の都合で資料はここまで。❷❸ はスライドで！

本発表に関する問合せ先　satkouda@gmail.com
個人ブログ「研究に使うポスグレ」http://kenpg.seesaa.net/