財務データ抽出システムの再構築
—Orbis データの利用—

地　道　正　行

要　旨

本稿では，地道，阪（2020-a, b），地道（2021-a, b）で議論された学内向け財務データ抽出システム SKWAD（スクワッド）において，抽出対象となるデータの拡充について検討される．追加されるものは，世界の全企業を対象として収集された財務情報を収録したデータベース Orbis から抽出されたデータである．SKWAD からのデータ抽出の実際と，その可視化についても紹介される．

キーワード：リレーショナルデータベース管理システム（Relational Database Management System），Orbis 財務データ（Orbis Financial Data），データ抽出システム（Data Extraction System），前処理（Preprocessing），再現可能性（Reproducibility）

I　はじめに

本稿では，Bureau van Dijk[1]（ビューロー・ヴァン・ダイク，以下 BvD と略）社のデータベース Orbis から抽出されたデータを地道，阪（2020-a, b），地道（2021-a, b）で議論された財務データ抽出システム SKWAD から提供することを検討する[2]．本稿で扱う内容は，地道（2021-b）における Osiris デー

1）　BvD Web Page: https://www.bvdinfo.com/en-gb/

2）　今回提供されるサービスの利用は，地道（2021-b）で言及されたように，筆者の研究グループとの共同研究を行う際に利用することを前提としているため，**不特定多数のユーザを対象としたものではなく，学内限定であり，かつアクセス制限をかけた仕様**となっていることに注意が必要である．

タから Orbis データにデータソースが変更されたということのみが相違点であるが，「規模」の観点からこれらのデータの間には大きな差があり[3]，この点に対する工夫が必要となる．

本稿の構成は以下のようなものである．まず，今回のシステム構築を行うための環境（II 節）と，利用するデータとその抽出もとになるデータベースについての情報を与える（III 節）．次に，データベースの構築についての手順を確認したあと，実際のデータベース構築について詳細に述べる[4]（IV 節）．さらに，構築されたシステムから実際にデータを抽出する方法について述べた後（V 節），そのデータを可視化する方法について，再現性の観点から述べる（VI 節）．最後に，今後の展望について述べる（VII 節）．なお，付録 A には R に関する環境の情報を，付録 B には，本稿で構築されたシステムを利用して抽出可能なデータのカラム名（財務指標等）の一覧も与える．

II 開発環境

本稿でも，地道，阪（2020-a, b），地道（2021-a, b）で議論した Ubuntu（Linux），macOS 等の UNIX 系オペレーティングシステム（Operating System: OS）上でリレーショナル・データベース管理システム（Relational Dababase Management System: RDBMS）と Web サーバとして Apache HTTP Server, 汎用スクリプト言語である PHP[5] を組み合わせた，いわゆる，MAMP, MAPP，LAMP，LAPP 環境を利用する．具体的なシステム構成としては，表 1 のような開発環境を用意した[6]：

3） 大まかな比較であるが，Orbis データは Osiris データの100倍近い規模となっている．
4） 抽出システム SKWAD の Orbis 財務データの抽出に関する仕様についてはセキュリティの観点から割愛する．
5） https://www.php.net/
6） 開発環境における，iMac Pro は2018年仕様のものであり，Dell Precision は Tower 7910 である．

表１：開発環境

Specification	MAMP	MAPP	LAMP	LAPP
Hardware	iMac Pro	iMac Pro	Dell Precision	Dell Precision
OS	macOS Big Sur（11.4）	macOS Big Sur（11.4）	Ubuntu（20.04）	Ubuntu（20.04）
CPU cores	36	36	48	48
Main Memory	128 GB	128 GB	128 GB	128 GB
RDBMS	MySQL 5.6.50	PostgreSQL 13.1	MySQL 8.0.22	PostgreSQL 12.5

Ⅲ　データベースとデータセット

本研究では，BvD 社のデータベース Orbis（オービス）から抽出されたデータを利用してシステムから抽出できるデータの種類を拡充する．データベース Orbis には，世界の全企業[7] を対象に収集された財務情報が国際比較可能な統一のフォームで納められている．

本稿では，Orbis から，データセットとして，主要財務情報を最長10年分抽出した以下のようなものを利用する：

（1）連結（Consolidated）財務諸表を優先的に抽出した26,353,934社（以後，「連結ベース」と略す）
（2）非連結（Un-consolidated）財務諸表（単体財務諸表）を優先的に抽出した26,352,382社（以後，「非連結ベース」と略す）

データセットのファイルは，１個のサイズが５GB 程度の27個の TSV ファイル（２セット）からなり，表２，表３には，それぞれ，データセットの仕様とサイズを与えている．

7）　本稿で利用している2019年12月版 Orbis には，約３億社以上が収録されている．

表 2 ：Orbis データセット：仕様

データセット名	年版	データベース	上場情報	抽出主体	抽出期間	抽出指標数
DS-Orbis-C-2019	2019年12月版	Orbis	上場・非上場	連結優先	10年	85
DS-Orbis-U-2019	2019年12月版	Orbis	上場・非上場	非連結優先	10年	85

表 3 ：Orbis データセット：サイズ

データセット名	社数	総行数	粗データファイル	トータルサイズ
DS-Orbis-C-2019	26,353,934	289,893,274	SJ_OB_2019_1_1000000.asc,...,SJ_OB_2019_26000001_26353934.asc	約 142 GB
DS-Orbis-U-2019	26,352,382	289,876,202	SJ_OB_2019_U_1_1000000.asc,...,SJ_OB_2019_U_26000001_26352382.asc	約 142 GB

地道（2020-a）では，2018年に抽出された Orbis データセットを GNU parallel を用いて並列化することによって効率的に前処理することを議論している．また，地道（2020-b）では，前処理された Orbis データセットを東京大学情報基盤センターに設置された専有利用型リアルタイムデータ解析ノード（以後，FENNEL と略す）と GPGPU 環境でデータベース管理システム PostgreSQL と **PG-Strom**[8] を利用することによって，ラングリングを高速化することが検討されている．さらに，地道ら（2020-a, b, c）では，2020年に新しく抽出された Orbis データセット DS-Orbis-C-2019（連結ベース），DS-Orbis-U-2019（非連結ベース）の前処理の並列化と **PG-Strom** によってラングリングを効率化することが議論されている．なお，処理後のファイル（CSV 形式）のサイズは，これらのデータセットのそれぞれに対して表 4 のようなものである．

表 4 ：前処理後のデータセット：サイズ

データセット名	社数	行数	列数	CSV ファイル	サイズ
DS-Orbis-C-2019	26,353,934	263,539,341	88	firmfinBC2019.csv	約 140.5 GB
DS-Orbis-U-2019	26,352,382	263,523,821	88	firmfinBU2019.csv	約 140.5 GB

8） https://heterodb.github.io/pg-strom/ja/

なお，抽出時の指標数が85に対して，処理後のファイルの列数が88になっているのは，前処理の工程で社名（firm），社名＋BvD ID（firmID），会計年度（year）を追加したためである．

Ⅳ　Orbis データによるデータベースの構築

1. Orbis データのデータベース構築手順

本節では，前節で説明した2019年12月版から抽出されたデータセット DS-Orbis-C-2019（連結ベース），DS-Orbis-U-2019（非連結ベース）を前処理した，それぞれのファイル firmfinBC2019.csv，firmfinBU2019.csv を利用してデータベースを構築する．データベース構築にあたっては，データベースから抽出する際のパフォーマンス等の関係から，これらのファイルをそのまま利用せずに，一部のものに制限することを考える．このことを実現するために，抽出された企業約2,600万社の約１％にあたる26万社をランダムサンプリングにより抽出し，それらの企業のデータをデータベース化する仕様とした．ただし，Orbis データの規模が大きく，通常のコンピュータ環境ではランダムサンプリングを実行すること自体が困難であるため，東京大学の FENNEL 環境を利用した．この環境においてランダムサンプリングに使用したスクリプトファイルを格納したディレクトリ構成を図１に与える．

```
Orbis2019Sampling
├──── Makefile
├──── dumpfirmIDs1p-csv.R
├──── loadfirmIDs1p.sql
├──── orbis2019-rs1per.R
└──── script-tar-csv.sh
```

図１：FENNEL 環境用スクリプトファイルのディレクトリ構成

また，ローカル環境において構築に利用したデータファイル，スクリプトファイルのディレクトリ構成については，図２（macOS 環境と Ubuntu 環境で共通）を参照されたい[9]．

```
orbis2019
├── CSV
│   ├── orbis2019cs1p.csv
│   └── orbis2019us1p.csv
└── DBMaking
    ├── Makefile
    ├── loadtoMySQL-cons.sql
    ├── loadtoMySQL-uncons.sql
    ├── loadtoMySQLgen-cons.sql
    ├── loadtoMySQLgen-uncons.sql
    ├── loadtoPostgreSQL-cons.sql
    ├── loadtoPostgreSQL-uncons.sql
    ├── loadtoPostgreSQLgen-cons.sql
    ├── loadtoPostgreSQLgen-uncons.sql
    ├── mkindexMS.sql
    ├── mkindexPG.sql
    ├── replaceloadMySQL-CDIR-cons.sh
    ├── replaceloadMySQL-CDIR-uncons.sh
    ├── replaceloadPostgreSQL-CDIR-cons.sh
    ├── replaceloadPostgreSQL-CDIR-uncons.sh
    └── sedbs
```

図 2 ：macOS 環境及び Ubuntu 環境用のデータファイル，スクリプトファイルのディレクトリ構成

データベースの構築手順は以下のようなものである：

Orbis データセットにもとづくデータベース構築手順

（Or-S1）ランダムサンプリング
（Or-S2）前処理
（Or-S3）データベース構築
　　（1）連結ベースのデータベース orbis2019cons 作成
　　（2）単体ベースのデータベース orbis2019uncons 作成

以下に手順（Or-S1），（Or-S2），（Or-S3）について詳しく述べる．

2. Orbis データからのランダムサンプリング

ここでの処理は，データセット DS-Orbis-C-2019（連結ベース），DS-Orbis-C-2019（非連結ベース）を前処理することによって得られたファイル firmfinBC2019.csv，firmfinBU2019.csv に収録されている企業からラン

9） ディレクトリとファイルの構成は同じであるが，スクリプトの内容は異なることに注意が必要である．

ダムサンプリングした26万社の企業に対する情報を新たなCSVファイルとして保存することである．これらのファイルはそれぞれ140GBを超えるものであり，サイズが大きいことから通常のコンピュータ環境では扱うことは難しい．この問題に対して，東京大学のFENNEL環境に用意されている**PG-Strom**環境を利用することを考える．**PG-Strom**は，GPGPU[10]のコアを並列で利用してPostgreSQL環境下でデータを高速に抽出することを可能にするシステムである[11]．上記のデータはFENNEL環境において，既にPostgreSQLのデータベース`jhpcn`（テーブル`orbis2019c`，`orbis2019u`）から抽出できるように用意されている[12]．

実際にランダムサンプリングを実行するのが，`Makefile`におけるターゲット`rs`（Random Sampling）である（スクリプト1参照）．

スクリプト1：`Makefile`：ターゲット`rs`

```
rs:
	date > start-rs.txt
	Rscript orbis2019-rs1per.R
	date > end-rs.txt
	date > start-load.txt
	psql postgres < ./loadfirmIDs1p.sql
	date > end-load.txt
	date > start-dump-csv.txt
	Rscript dumpfirmIDs1p-csv.R
	date > end-dump-csv.txt
	date > start-tar-csv.txt
	/bin/bash script-tar-csv.sh
	date > end-tar-csv.txt
```

まず，スクリプト1における，それぞれ，（2，4）行目，（5，7）行目，（8，10）行目，（11，13）行目は処理時間を計測するための指定である．ターゲッ

10）GPGPUとは，General-Purpose computing on Graphics Processing Unitsの略語であり，画像処理を高速に実行するGPU（Graphics Processing Unit）の機能を，画像処理以外の用途に転用することである（IT用語辞典`https://e-words.jp/w/GPGPU.html`参照）

11）**PG-Strom**によってデータラングリングを効率化することについては，地道（2020-b），地道ら（2020-a, b, c）の一連の研究を参照されたい．

12）データファイル`firmfinBC2019.csv`，`firmfinBU2019.csv`のデータベース化は，東京大学の宮本大輔氏の協力を仰いだ．

ト rs を実行することにともなう，スクリプトファイルの流れを図 3 に与える．これらのスクリプトの実行は，以下のような手順を通じて行われる：

Orbis データセットからランダムサンプリングを行うための手順

（Or-RS1）26万社の企業の社名情報（firmID）をランダムサンプリング
（Or-RS2）抽出された26万社の firmID のテーブルをデータベース jhpcn に作成
（Or-RS3）データを CSV ファイルへ出力
（Or-RS4）データの CSV ファイルを単一のファイルへ圧縮

ランダムサンプリングにともなう Orbis データファイルの流れを可視化したものを図 4 に与える．

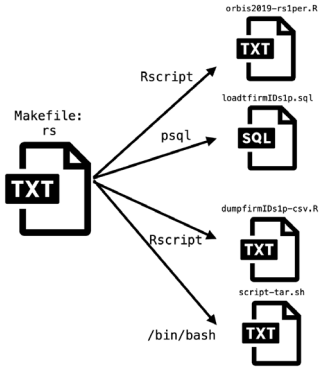

図 3 ：Orbis データファイルのランダムサンプリングに関するシェルスクリプトの流れ

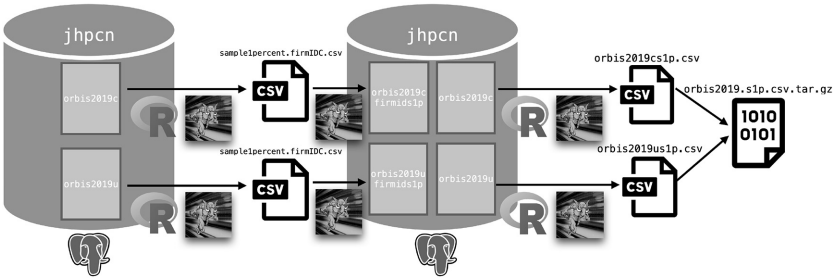

図 4 ：**FENNEL** 環境のもとでの Orbis データのランダムサンプリングにともなうデータファイルの流れ

まず，手順（Or-RS1）で行われる処理は，Makefile におけるターゲットが rs（スクリプト 1）の 3 行目で指定されているように，Rscript コマンドにスクリプト 2 を指定することによって行われる．

スクリプト 2 ：orbis2019-rs1per.R

```
 1 # environments setting
 2 library(RPostgreSQL)
 3 drv <- dbDriver("PostgreSQL")
 4 con <- dbConnect(drv, host="localhost", port=5432, user= "********",
       password="********", dbname="jhpcn")
 5 # fetch all firmID of orbis2019c
 6 sql.firmIDlistC <- "SELECT␣firmID␣FROM␣orbis2019c␣WHERE␣year␣=␣2009"
 7 rs.firmIDlistC <- dbSendQuery(con, sql.firmIDlistC)
 8 firmIDlistC <- fetch(rs.firmIDlistC, n=-1)
 9 # fetch all firmID list orbis2019u
10 sql.firmIDlistU <- "SELECT␣firmID␣FROM␣orbis2019u␣WHERE␣year␣=␣2009"
11 rs.firmIDlistU <- dbSendQuery(con, sql.firmIDlistU)
12 firmIDlistU <- fetch(rs.firmIDlistU, n=-1)
13 x <- as.vector(firmIDlistC$firmid)
14 y <- as.vector(firmIDlistU$firmid)
15 # random sampling
16 set.seed(12345)
17 sample1percent.firmIDC <- sample(x, 26*10^4)
18 sample1percent.firmIDU <- sample(y, 26*10^4)
19 # dump 1% firmIDs of orbis2019
20 library(readr)
21 write_csv(as.data.frame(sample1percent.firmIDC), "sample1percent.firmIDC.
       csv")
22 write_csv(as.data.frame(sample1percent.firmIDU), "sample1percent.firmIDU.
       csv")
```

スクリプト 2 によって行われる処理は以下のようなものである：

（R1）データベース jhpcn とのコネクション設定（1 〜 4 行目）

（R2）データベース jhpcn の連結ベースのテーブル orbis2019c と非連結ベースのテーブル orbis2019u から，それぞれ，単年（ここでは2009年）分の全ての企業の firmID（社名＋BvD ID のコード）をフェッチし，R のベクトルオブジェクト x，y として格納（5 〜14行目）

（R3）firmID のベクトルオブジェクト x，y から，26万社分をランダムサンプリング（15〜18行目）

（R4）ランダムサンプリングされたデータを CSV ファイル sample1per-

cent.firmIDC.csv（連結ベース），sample1percent.firmIDU.csv（非連結ベース）へ出力

次に，手順（Or-RS2）で行われる処理は，Makefile におけるターゲットが rs（スクリプト 1）の 6 行目で指定されているように，psql コマンドにスクリプト 3 を指定することによって行われる．

スクリプト 3：loadfirmIDs1p.sql

```
\c jhpcn
DROP TABLE IF EXISTS orbis2019cfirmids1p;
DROP TABLE IF EXISTS orbis2019ufirmids1p;
CREATE TABLE orbis2019cfirmids1p (
firmid VARCHAR(300)
);
CREATE TABLE orbis2019ufirmids1p (
firmid VARCHAR(300)
);
\COPY public.orbis2019cfirmids1p FROM '~/Orbis2019Sampling/sample1percent.
    firmIDC.csv' (format csv, header true);
\COPY public.orbis2019ufirmids1p FROM '~/Orbis2019Sampling/sample1percent.
    firmIDU.csv' (format csv, header true);
```

スクリプト 3 によって行われる処理は以下のようなものである：

（L1）データベース jhpcn を選択（1 行目）

（L2）もし古いテーブル orbis2019cfirmids1p，orbis2019ufirmids1p が存在するならばドロップ（2， 3 行目）

（L3）テーブル orbis2019cfirmids1p（連結ベース）の作成（4 〜 6 行目）

（L4）テーブル orbis2019ufirmids1p（非連結ベース）の作成（7 〜 9 行目）

（L5）連結ベースの抽出された26万社の企業の firmID の CSV ファイル sample1percent.firmIDC.csv からテーブル orbis2019cfirmids1p へデータをコピー（10行目）

（L6）非連結ベースの抽出された26万社の企業の firmID の CSV ファイル sample1percent.firmIDU.csv からテーブル orbis2019ufir-

mids1p へデータをコピー（10行目）

さらに，手順（Or-RS3）で行われる処理は，Makefile におけるターゲットが rs（スクリプト 1 ）の 9 行目で指定されているように，Rscript コマンドにスクリプト 4 を指定することによって行われる．

スクリプト 4 ：dumpfirmIDs1p-csv.R

```
 1 # environment setting
 2 library(RPostgreSQL)
 3 drv <- dbDriver("PostgreSQL")
 4 con <- dbConnect(drv, host="localhost", port=5432, user= "********",
       password="********", dbname="jhpcn")
 5 # fetch orbis2019cs1p
 6 sql.orbis2019cs1p <- "SELECT␣*␣FROM␣orbis2019c␣WHERE␣firmid␣IN␣(select␣
       firmid␣from␣orbis2019cfirmids1p)"
 7 rs.orbis2019cs1p <- dbSendQuery(con, sql.orbis2019cs1p)
 8 orbis2019cs1p <- fetch(rs.orbis2019cs1p , n=-1)
 9 # fetch orbis2019us1p
10 sql.orbis2019us1p <- "SELECT␣*␣FROM␣orbis2019u␣WHERE␣firmid␣IN␣(select␣
       firmid␣from␣orbis2019ufirmids1p)"
11 rs.orbis2019us1p <- dbSendQuery(con, sql.orbis2019us1p)
12 orbis2019us1p <- fetch(rs.orbis2019us1p , n=-1)
13 # dump CSV files
14 library(readr)
15 write_csv(orbis2019cs1p, "orbis2019cs1p.csv")
16 write_csv(orbis2019us1p, "orbis2019us1p.csv")
```

スクリプト 4 によって行われる処理は以下のようなものである：

（D1）データベース jhpcn とのコネクション設定（ 1 〜 4 行目）

（D2）ランダムサンプリングされた連結ベースの26万社の firmID のテーブル orbis2019cfirmids1p の企業を連結ベースの企業情報が納められたテーブル orbis2019c から抽出し，データフレームオブジェクト orbis2019cs1p に付値（ 5 〜 8 行目）

（D3）ランダムサンプリングされた連結ベースの26万社の firmID のテーブル orbis2019ufirmids1p の企業を非連結ベースの企業情報が納められたテーブル orbis2019u から抽出し，データフレームオブジェクト orbis2019us1p に付値（ 9 〜12行目）

（D4）データフレームオブジェクト orbis2019cs1p の内容を CSV ファイ

ル orbis2019cs1p.csv へ出力

(D5) データフレームオブジェクト orbis2019us1p の内容を CSV ファイル orbis2019us1p.csv へ出力

最後に，手順（Or-RS4）の処理は，Makefile におけるターゲットが rs（スクリプト 1 ）の12行目で指定されているように，シェル /bin/bash にスクリプト 5 を指定することによって行われる．

スクリプト 5 ：script-tar-csv.sh

```
1 #!/bin/bash
2 tar cvzf orbis2019.s1p.csv.tar.gz orbis2019*.csv
```

スクリプト 5 によって行われる処理は，データの CSV ファイル orbis2019cs1p.csv，orbis2019us1p.csv を tar コマンドを利用して圧縮し，ファイル orbis2019s1p.csv.tar.gz へ出力することである．

さらに，Makefile におけるターゲット rs から実行されるシェル・スクリプト・ファイルとそれによって処理されるデータファイルの対応関係を可視化したものを図 5 に与える．

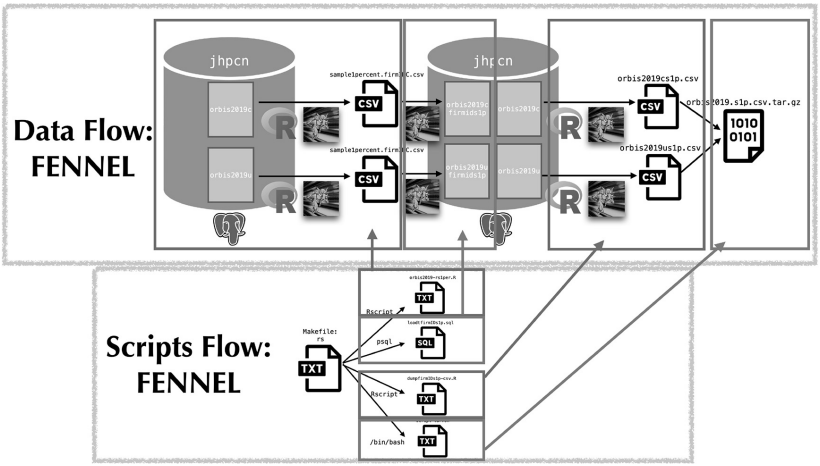

図 5 ：Orbis データのランダムサンプリングに利用されるシェルス・クリプト・ファイルとデータファイルの処理の対応関係：FENNEL 環境を利用

なお，この処理は，FENNEL 環境におけるディレクトリ Orbis2019Sampling（図 1 参照）をカレント（ディレクトリ）とし，ターミナル（コマンドライン）上で以下のように入力することによって実行できる．

ターゲット rs の実行

```
$ make rs
```

この処理時間は，スクリプト 1 から実行される手順（Or-RS1）～（Or-RS4）の 4 段階の開始・終了時刻を比較することによってわかる．まず，手順（Or-RS1）の処理に要した時間は以下のようなものである：

ターゲット rs における orbis2019-rs1per.R の処理時間の計測

```
$ cat start-rs.txt
2021 年 4 月 28 日 水曜日 00:35:58 JST
$ cat end-rs.txt
2021 年 4 月 28 日 水曜日 00:39:43 JST
```

この結果から，3 分45秒であることがわかる[13]．

次に，手順（Or-RS2）の処理に要した時間は以下である：

ターゲット rs における loadfirmIDs1p.sql の処理時間の計測

```
$ cat start-load.txt
2021 年 4 月 28 日 水曜日 00:39:43 JST
$ cat end-load.txt
2021 年 4 月 28 日 水曜日 00:39:43 JST
```

この結果から，1 秒以下であることがわかる[14]．

また，手順（Or-RS3）の処理に要した時間は以下である：

13）今回，**PG-Strom** を利用した場合しか計測していないが，このような短時間で企業のランダムサンプリングが可能となったのは，**PG-Strom** と FENNEL 環境の恩恵である．

ターゲット rs における dumpfirmIDs1p-csv.R の処理時間の計測

```
$ cat start-dump-csv.txt
2021 年 4 月 28 日 水曜日 00:39:43 JST
$ cat end-dump-csv.txt
2021 年 4 月 28 日 水曜日 00:48:06 JST
```

この結果から，8 分23秒であることがわかる．

最後に，手順（Or-RS4）の処理に要した時間は以下である：

ターゲット rs における script-tar-csv.sh の処理時間の計測

```
$ cat start-tar-csv.txt
2021 年 4 月 28 日 水曜日 00:48:06 JST
$ cat end-tar-csv.txt
2021 年 4 月 28 日 水曜日 00:48:52 JST
```

この結果から，46秒であることがわかる．

以上の結果を総合すると，合計時間は，12分54秒（＝ 3 分45秒＋ 0 秒＋ 8 分23秒＋46秒）である[15]．

以上のランダムサンプリングの処理によって得られたファイルのサイズは以下のようなものである：

ランダムサンプリングによって得られた CSV ファイルとその圧縮ファイル

```
$ ls -l orbis2019*.csv
-rw-r--r--@ 1 masa  staff  1391466575  4 28 00:47 orbis2019cs1p.csv
-rw-r--r--@ 1 masa  staff  1391830837  4 28 00:48 orbis2019us1p.csv
$ ls -l orbis2019.s1p.csv.tar.gz
-rw-r--r--@ 1 masa  staff  241291994  4 28 00:48 orbis2019.s1p.csv.tar.gz
```

この結果から，CSV ファイル orbis2019cs1p.csv，orbis2019us1p.csv は，それぞれ，1.4 GB 程度であり，それらを圧縮したファイル orbis2019.s1p.csv.tar.gz は 240 MB 程度であり，サイズが 1/10以下に圧縮

14）この結果も **PG-Strom** と FENNEL 環境の恩恵といえる．

15）この処理は頻繁に実行するものではないので，この時間で実行できることは十分許容されるものであるが，今回の実際の処理をおこなった経験から，ボトルネックは R から CSV ファイルを出力する段階にあるように思われる．

されていることがわかる.

これらのファイルを利用したデータベースの作成は，ローカル環境で行うため，FENNEL 環境から sftp コマンドを利用して転送し，tar コマンドを利用してローカル側で展開した（図 6 参照）.

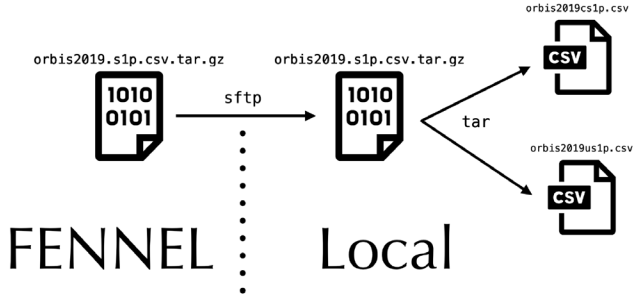

図 6 ：FENNEL 環境からローカル環境へファイルの転送と展開

展開後の CSV ファイル orbis2019cs1p.csv，orbis2019us1p.csv は，ローカル環境におけるディレクトリ orbis2019 のサブディレクトリ CSV に保存している（図 2 参照）.

3. Orbis データの前処理

ここでの前処理は，ファイル orbis2019cs1p.csv，orbis2019us1p.csv のバックスラッシュ（\）を二重バックスラッシュ（\\）へ置換し，新たなファイル firmfinBC2019s1p.csv，firmfinBU2019s1p.csv へ変換することである[16]．これは，MySQL においてファイルからデータをインポートする際に，エラーとなることを防ぐための処理である.

この処理，すなわち手順（Or-S2）を行うための Makefile におけるターゲットが preprocess である（スクリプト 6 参照）.

16）データベース Orbis から抽出された粗データファイルの前処理については，地道（2020-a）を参照されたい.

スクリプト 6 ：Makefile: ターゲット preprocess

```
1 preprocess:
2       date > start-preprocess.txt
3       /bin/bash script-preprocess.sh
4       date > end-preprocess.txt
```

スクリプト 6 における 2 行目と 4 行目は処理時間を計測するための指定である．また，3 行目では，シェル・スクリプト・ファイル script-preprocess.sh（スクリプト 7 ）が指定されている．

スクリプト 7 ：script-preprocess.sh

```
1 #!/bin/bash
2 sed -f sedbs ../CSV/orbis2019cs1p.csv > ./firmfinBC2019s1p.csv
3 sed -f sedbs ../CSV/orbis2019us1p.csv > ./firmfinBU2019s1p.csv
```

このスクリプトでは，置換を行うためのルールをファイル sedbs（スクリプト 8 ）に定義しておき，sed[17] を利用して置換している．

スクリプト 8 ：sedbs

```
1 # 置換 \ -> \\
2 s/\\/\\\\/g
```

ターゲット preprocess を実行することにともなう，スクリプトファイルの流れを図 7 に与える．

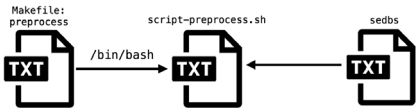

図 7 ：Orbis データファイルの前処理に関するシェルスクリプトの流れ

また，Orbis データファイルの前処理における流れを可視化したものを図 8 に与える．

17）sed は，ストリームエディタ（stream editor）の略であり，UNIX 環境でファイルなどの文字列置換に標準的に利用されるコマンドである．

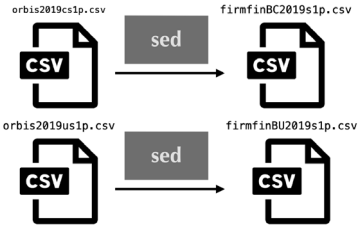

図 8 ：Orbis データの前処理におけるデータファイルの流れ

さらに，Makefile におけるターゲット preprocess から実行されるシェルスクリプトとデータに関するファイルの流れの対応を可視化したものを図 9 に与える．

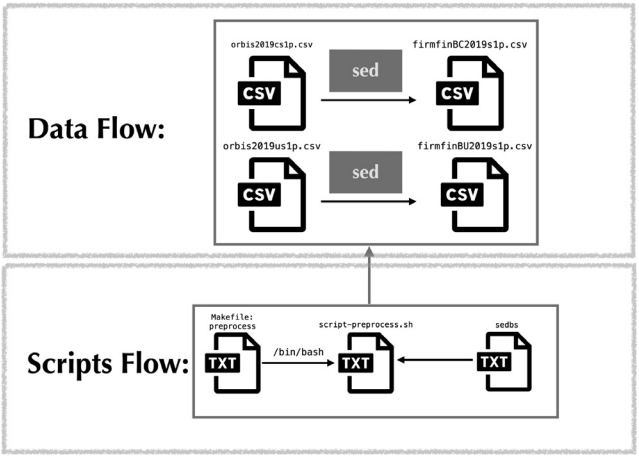

図 9 ：Orbis データの前処理におけるシェルスクリプトとデータに関するファイルの流れ

なお，前処理は，ディレクトリ DBMaking（図 2 参照）をカレント（ディレクトリ）とし，ターミナル（コマンドライン）上で以下のように入力することによって実行できる（ただし，%，$ はシェルプロンプトである）．

ターゲット preprocess の実行：macOS，Ubuntu 共通

```
% make preprocess
```

この処理時間は，スクリプト 6 において，2 行目と 5 行目の実行結果を比較することによってわかる．macOS 上で実行した結果を以下に与える．

macOS 上でターゲット preprocess の処理時間の計測

```
masa@aule DBMaking % cat start-preprocess.txt
Mon May 10 08:36:18 JST 2021
masa@aule DBMaking % cat end-preprocess.txt
Mon May 10 08:36:25 JST 2021
```

この結果から，7 秒であることがわかる[18]．

一方，Ubuntu 上で実行した結果も以下に与える．

Ubuntu 上でターゲット preprocess の処理時間の計測

```
masa@balin: DBMaking$ cat start-preprocess.txt
Sun May  2 11:53:13 JST 2021
masa@balin: DBMaking$ cat end-preprocess.txt
Sun May  2 11:53:24 JST 2021
```

この結果から，11秒である[19]．

4. Orbis データによるデータベース構築

MySQL の場合

RDBMS として，MySQL を利用する場合，macOS と Ubuntu（すなわち，MAPP と LAPP）の間で構築するためのスクリプトをそれぞれ準備する必要があるため，それぞれの場合に分けて述べる[20]．

18）この結果は，iMac Pro 2018（macOS Big Sur）で計測したものである．
19）この結果は，Dell Precision Tower 7910（Ubuntu 20.04）で計測したものである．
20）今回構築した環境は，RDBMS を OS にインストールするために，macOS と Ubuntu 上のパッケージ管理システム（Package Management System: PMS）を，それぞれ，brew（Homebrew）と apt を用いて MySQL をインストールしている．データベー

■macOS（MAMP）環境のもとでのデータベース構築　手順（Or-S3）を実行するスクリプトを Makefile のターゲット MSDB に記述した（スクリプト 9 参照）．このスクリプトおける，2 行目と 7 行目は処理時間を計測するための指定である．

スクリプト 9：Makefile: ターゲット MSDB（macOS）

```
1 MSDB:
2       date > start-MSDB.txt
3       /bin/bash replaceloadMySQL-CDIR-cons.sh
4       mysql -u root -psinx=0 < ./loadtoMySQL-cons.sql
5       /bin/bash replaceloadMySQL-CDIR-uncons.sh
6       mysql -u root -psinx=0 < ./loadtoMySQL-uncons.sql
7       date > end-MSDB.txt
```

スクリプト 9 の 3，4 行目が手順（Or-S3）の（1）を実現するためのものであり，5，6 行目によって手順（Or-S3）の（2）を実現する．なお，ターゲット MSDB によって実行されるスクリプトファイルの流れを可視化したものを図10に与える．

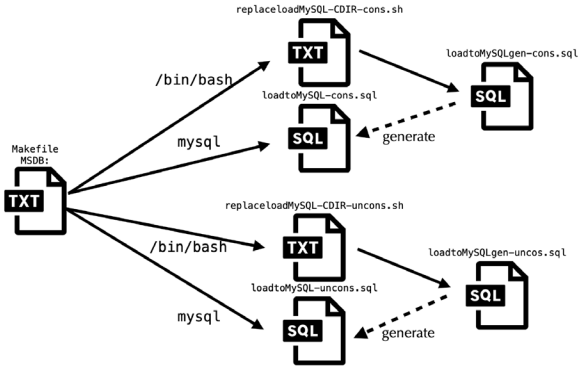

図10：ターゲット MSDB の実行にともなうスクリプトファイルの流れ

まず，手順（Or-S3）の（1）を実現するためのスクリプトについてみる．3 行目で利用されているシェルスクリプト replaceloadMySQL-CDIR-

スを構築するためのスクリプトに差異があるのは，これらの PMS を用いてインストールした際に，RDBMS のルート権限などの付与の仕様が異なっているためである．

cons.sh の内容（スクリプト10）をみると，2行目でカレントディレクトリの情報を環境変数に代入（CDIR=$PWD）しており，3行目では sed コマンドを利用して，SQL[21] スクリプトファイル loadtoMySQLgen-cons.sql（スクリプト11）における文字列 CDIR をカレントディレクトリの情報で置換（スクリプト11の12行目）し，その結果を SQL スクリプトファイル loadtoMySQL-cons.sql にリダイレクション（>）機能を使って出力している．この仕様から，SQL スクリプトファイル loadtoMySQL-cons.sql は，シェルスクリプト replaceloadMySQL-CDIR-cons.sh によって SQL スクリプトファイル loadtoMySQLgen-cons.sql から生成（generate）される（図10参照）．

スクリプト10：replaceloadMySLQ-CDIR-cons.sh

```
1 #!/bin/bash
2 CDIR=$PWD
3 sed -e "s|CDIR|$CDIR|g" loadtoMySQLgen-cons.sql > loadtoMySQL-cons.sql
```

スクリプト11：loadtoMySQLgen-cons.sql（一部抜粋）

```
 1 DROP DATABASE IF EXISTS orbis2019cons;
 2 CREATE DATABASE orbis2019cons;
 3 USE orbis2019cons;
 4 DROP TABLE IF EXISTS orbis2019cons;
 5 CREATE TABLE orbis2019cons (
 6 firm VARCHAR(350),
 7 :
 8 : (中略)
 9 :
10 year VARCHAR(4)
11 );
12 LOAD DATA LOCAL INFILE 'CDIR/firmfinBC2019s1p.csv'
13 INTO TABLE orbis2019cons
14 FIELDS TERMINATED BY ','
15 ENCLOSED BY '"'
16 IGNORE 1 LINES;
```

21）SQL とは，Structured Query Language の略であり，構造化照会言語と訳されることがある．リレーショナル・データベース管理システム（Relational DataBase Management System; RDBMS）とのインターフェース言語として国際標準として利用されている．詳細については，例えば，増永（2017）を参照されたい．

生成されたSQLスクリプトファイルloadtoMySQL-cons.sqlは，データベースorbis2019consと，テーブルorbis2019consを作成（2～11行目）した後，データファイルfirmfinBC2019s1p.csvから，テーブルorbis2019consへデータをロード（12～16行目）するためのものであり，実際にターゲットMSDB（スクリプト9）の4行目で実行される．

次に，手順（Or-S3）の（2）を実現するためのスクリプトとして，Makefileのターゲット MSDB（スクリプト9）の5行目で実行されるスクリプトファイルreplaceloadMySQL-CDIR-uncons.shの内容（スクリプト12）をみると，2行目でカレントディレクトリの情報を環境変数に代入（CDIR=$PWD）しており，3行目ではsedコマンドを利用して，SQLスクリプトファイルloadtoMySQLgen-uncons.sql（スクリプト13）における文字列CDIRをカレントディレクトリの情報で置換（スクリプト13の12行目）し，その結果をSQLスクリプトファイルloadtoMySQL-uncons.sqlにリダイレクション（>）機能を使って出力している．この仕様から，SQLスクリプトファイルloadtoMySQL-uncons.sqlは，シェルスクリプトreplaceloadMySQL-CDIR-uncons.shによってSQLスクリプトファイルloadtoMySQLgen-uncons.sqlから生成される（図10参照）．

スクリプト12：replaceloadMySLQ-CDIR-uncons.sh

```
1 #!/bin/bash
2 CDIR=$PWD
3 sed -e "s|CDIR|$CDIR|g" loadtoMySQLgen-uncons.sql > loadtoMySQL-uncons.sql
```

スクリプト13：loadtoMySQLgen-uncons.sql（一部抜粋）

```
1  DROP DATABASE IF EXISTS orbis2019uncons;
2  CREATE DATABASE orbis2019uncons;
3  USE orbis2019uncons;
4  DROP TABLE IF EXISTS orbis2019uncons;
5  CREATE TABLE orbis2019uncons (
6  firm VARCHAR(350),
7  :
8  : （中略）
9  :
10 year VARCHAR(4)
```

```
);
LOAD DATA LOCAL INFILE 'CDIR/firmfinBU2019s1p.csv'
INTO TABLE orbis2019uncons
FIELDS TERMINATED BY ','
ENCLOSED BY '"'
IGNORE 1 LINES;
```

生成された SQL スクリプトファイル loadtoMySQL-uncons.sql は，データベース orbis2019uncons と，テーブル orbis2019uncons を作成（2〜11行目）した後，データファイル firmfinBU2019s1p.csv から，テーブル orbis2019uncons へデータをロード（12〜16行目）するためのものであり，実際にターゲット MSDB（スクリプト 9）の 6 行目で実行される．

以上のデータベース構築の流れを簡略化して可視化したものを図11に与える．

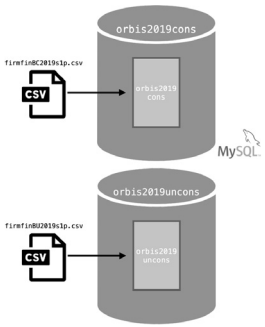

図11：MySQL よる Orbis データにもとづくデータベース orbis2019cons，orbis2019uncons の構築

さらに，Makefile におけるターゲット MSDB から実行されるシェルスクリプトとデータに関するファイルの流れの対応を可視化したものを図12に与える．

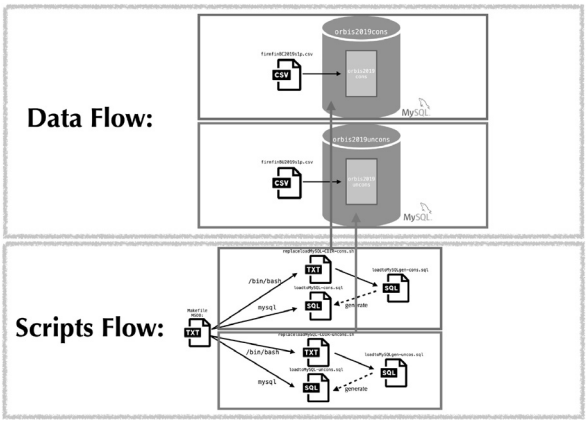

図12：Orbis データに関するデータベース構築のためのターゲット MSDB の実行にともなうシェルスクリプトとデータに関するファイルの流れ

図11，12は macOS におけるデータベース構築の流れを表しているが，Ubuntu 上でも全く同じことがいえる．

ターゲット MSDB はディレクトリ DBMaking（図 2 参照）をカレントとし，ターミナル上で以下のように入力することによって実行できる．

macOS 上でターゲット MSDB の実行

```
% make MSDB
```

macOS 上での，この処理時間は，スクリプト 9 において，2，7 行目の実行結果を比較することによってわかる．

macOS 上でターゲット MSDB の処理時間の計測

```
masa@aule DBMaking % cat start-MSDB.txt
Mon May 10 08:37:01 JST 2021
masa@aule DBMaking % cat end-MSDB.txt
Mon May 10 08:38:17 JST 2021
```

この結果から，1 分16秒であることがわかる[22)]．

■Ubuntu（LAMP）環境のもとでのデータベース構築 macOS 環境（MAMP）と Ubuntu 環境（LAMP）に対するデータベースを構築するためのスクリプトの唯一の違いは，Makefile におけるターゲット MSDB にある．

スクリプト14：Makfile：ターゲット MSDB（Ubuntu）

```
1 MSDB:
2       date > start-MSDB.txt
3       /bin/bash replaceloadMySQL-CDIR-cons.sh
4       sudo mysql --local_infile=1 < ./loadtoMySQL-cons.sql
5       /bin/bash replaceloadMySQL-CDIR-uncons.sh
6       sudo mysql --local_infile=1 < ./loadtoMySQL-uncons.sql
7       date > end-MSDB.txt
```

macOS 用のターゲット MSDB（スクリプト 9）と，Ubuntu 用のもの（スクリプト14）を比較することによって，MySQL との対話型インターフェース mysql を実行する権限の仕様が若干異なっていることがわかる．これは，macOS と Ubuntu のそれぞれの PMS（brew，apt）でインストールした MySQL のバージョンと仕様に差異があるためである．ただし，データベースの構築のためには，macOS と同様に，ディレクトリ DBMaking（図 2 参照）をカレントとして，Ubuntu のターミナルで以下のように make コマンドを実行すればよい．

Ubuntu 上でターゲット MSDB の実行

```
$ make MSDB
```

この処理時間は，スクリプト14において， 2， 7 行目の実行結果を比較することによってわかる．

Ubuntu 上でターゲット MSDB の処理時間の計測

```
masa@balin: DBMaking$ cat start-MSDB.txt
Sun May  2 11:30:48 JST 2021
masa@balin: DBMaking$ cat end-MSDB.txt
Sun May  2 11:34:47 JST 2021
```

22）この結果は，iMac Pro 2018（macOS Big Sur）で計測したものである．

この結果から，3分59秒であることがわかる[23]．

PostgreSQLの場合

RDBMSとして，PostgreSQLを利用する場合も，macOSとUbuntu（すなわち，MAPPとLAPP）の間で構築するためのスクリプトをそれぞれ準備する必要があるため，各場合に分けて述べる．

■macOS（MAPP）環境のもとでのデータベース構築 手順（Or-S3）を実行するスクリプトを`Makefile`のターゲット`PGDB`に記述した（スクリプト15参照）．このスクリプトにおいて，2行目と7行目は処理時間を計測するための指定である．

スクリプト15：`Makefile`：ターゲット`PGDB`(macOS)

```
1 PGDB:
2       date > start-PGDB.txt
3       /bin/bash replaceloadPostgreSQL-CDIR-cons.sh
4       psql postgres < ./loadtoPostgreSQL-cons.sql
5       /bin/bash replaceloadPostgreSQL-CDIR-uncons.sh
6       psql postgres < ./loadtoPostgreSQL-uncons.sql
7       date > end-PGDB.txt
```

スクリプト15の3，4行目が手順（Or-S3）の（1）を実現するためのものであり，5，6行目が手順（Or-S3）の（2）を実現するためのものである．なお，ターゲット`PGDB`によって実行されるスクリプトファイルの流れを可視化したものを図13に与える．

23）この結果は，Dell Precision Tower 7910（Ubuntu 20.04）で計測したものである．

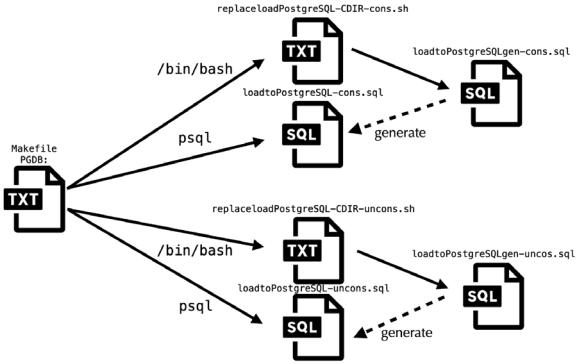

図13：ターゲット PGDB の実行にともなうスクリプトファイルの流れ

まず，手順（Or-S3）の（1）を実現するためのスクリプトについてみる．3行目で利用されているシェルスクリプト replaceloadPostgreSQL-CDIR-cons.sh の内容（スクリプト16）をみると，2行目でカレントディレクトリの情報を環境変数に代入（CDIR=$PWD）しており，3行目では sed コマンドを利用して，SQL スクリプトファイル loadtoPostgreSQLgen-cons.sql（スクリプト17）における文字列 CDIR をカレントディレクトリの情報で置換（スクリプト17の12行目）し，その結果を SQL スクリプトファイル loadtoPostgreSQL-cons.sql にリダイレクション（>）機能を使って出力している．この仕様から，SQL スクリプトファイル loadtoPostgreSQL-cons.sql は，シェルスクリプト replaceloadPostgreSQL-CDIR-cons.sh によって SQL スクリプトファイル loadtoPostgreSQL-gen-cons.sql から生成される（図13参照）．

スクリプト16：replaceloadPostgreSLQ-CDIR-cons.sh

```
1 #!/bin/bash
2 CDIR=$PWD
3 sed -e "s|CDIR|$CDIR|g" loadtoPostgreSQLgen-cons.sql > loadtoPostgreSQL-
    cons.sql
```

スクリプト17：loadtoPostgreSQLgen-cons.sql（一部抜粋）

```
 1 DROP DATABASE IF EXISTS orbis2019cons;
 2 CREATE DATABASE orbis2019cons;
 3 \c orbis2019cons;
 4 DROP TABLE IF EXISTS orbis2019cons;
 5 CREATE TABLE orbis2019cons (
 6 firm VARCHAR(350),
 7 :
 8 : (中略)
 9 :
10 year VARCHAR(4)
11 );
12 COPY public.orbis2019cons  FROM 'CDIR/firmfinBC2019s1p.csv' WITH csv HEADER
     ;
```

生成されたSQLスクリプトファイルloadtoPostgreSQL-cons.sqlは，データベースorbis2019consと，テーブルorbis2019consを作成（2〜11行目）した後，データファイルfirmfinBC2019s1p.csvから，テーブルorbis2019consへデータをロード（12行目）するためのものであり，実際にターゲットPGDB（スクリプト15）の4行目で実行される．

次に，手順（Or-S3）の（2）を実現するためのスクリプトとして，Makefileのターゲット PGDB（スクリプト15）の5行目で実行されるスクリプトファイル replaceloadPostgreSQL-CDIR-uncons.shの内容（スクリプト18）をみると，2行目でカレントディレクトリの情報を環境変数に代入（CDIR=$PWD）しており，3行目ではsedコマンドを利用して，SQLスクリプトファイル loadtoPostgreSQLgen-uncons.sql（スクリプト19）における文字列CDIRをカレントディレクトリの情報で置換（スクリプト19の12行目）し，その結果をSQLスクリプトファイル loadtoPostgreSQL-uncons.sqlにリダイレクション（>）機能を使って出力している．この仕様から，SQLスクリプトファイル loadtoPostgreSQL-uncons.sqlは，シェルスクリプト replaceloadPostgreSQL-CDIR-uncons.shによってSQLスクリプトファイル loadtoPostgreSQLgen-uncons.sqlから生成される（図13参照）．

スクリプト18：replaceloadPostgreSLQ-CDIR-uncons.sh

```
1 #!/bin/bash
2 CDIR=$PWD
3 sed -e "s|CDIR|$CDIR|g" loadtoPostgreSQLgen-uncons.sql > loadtoPostgreSQL-
    uncons.sql
```

スクリプト19：loadtoPostgreSQLgen-uncons.sql（一部抜粋）

```
 1 DROP DATABASE IF EXISTS orbis2019uncons;
 2 CREATE DATABASE orbis2019uncons;
 3 \c orbis2019uncons;
 4 DROP TABLE IF EXISTS orbis2019uncons;
 5 CREATE TABLE orbis2019uncons (
 6 firm VARCHAR(350),
 7 :
 8 : (中略)
 9 :
10 year VARCHAR(4)
11 );
12 COPY public.orbis2019uncons  FROM 'CDIR/firmfinBU2019s1p.csv' WITH csv
    HEADER;
```

生成されたSQLスクリプトファイルloadtoPostgreSQL-uncons.sqlは，データベースorbis2019unconsと，テーブルorbis2019unconsを作成（2～11行目）した後，データファイルfirmfinBU2019s1p.csvから，テーブルorbis2019unconsへデータをロード（12行目）するためのものであり，実際にターゲットPGDB（スクリプト15）の6行目で実行される．

以上のデータベース構築の流れを簡略化して可視化したものを図14に与える．

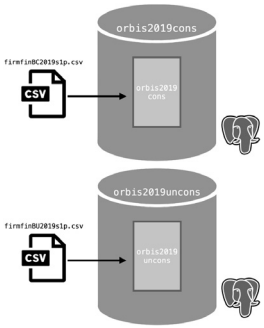

図14：PostgreSQL よる Orbis データにもとづくデータベース **orbis2019cons**, **orbis2019uncons** の構築

さらに，`Makefile` におけるターゲット `PGDB` から実行されるシェルスクリプトとデータに関するファイルの流れの対応を可視化したものを図15に与える．

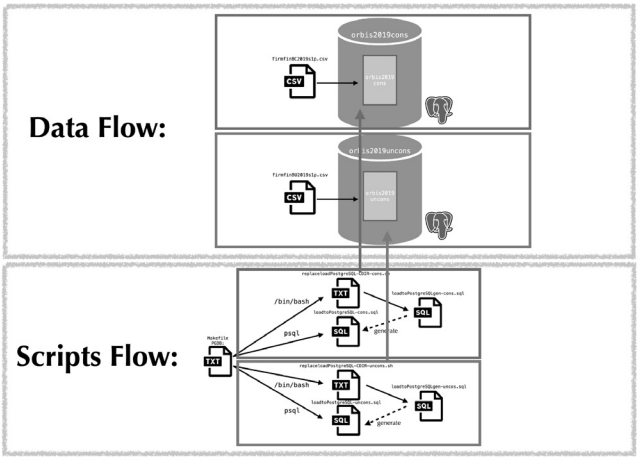

図15：Orbis データに関するデータベース構築のためのターゲット **PGDB** の実行にともなうシェルスクリプトとデータに関するファイルの流れ

図14，15は macOS におけるデータベース構築の流れを表しているが，Ubuntu 上でも全く同じことがいえる．

ターゲット PGDB はディレクトリ DBMaking（図 2 参照）をカレントとし，ターミナル上で以下のように入力することによって実行できる．

macOS 上でターゲット PGDB の実行

```
% make PGDB
```

macOS 上での，この処理時間は，スクリプト15において，2，7 行目の実行結果を比較することによってわかる．

macOS 上でターゲット PGDB の処理時間の計測

```
masa@aule DBMaking % cat start-PGDB.txt
Mon May 10 08:45:25 JST 2021
masa@aule DBMaking % cat end-PGDB.txt
Mon May 10 08:46:15 JST 2021
```

この結果から，50秒であることがわかる[24]．

■Ubuntu（LAMP）環境のもとでのデータベース構築　macOS 環境（MAMP）と Ubuntu 環境（LAMP）に対するデータベースを構築するためのスクリプトの唯一の違いは，Makefile におけるターゲット PGDB にある．

スクリプト20：Makfile：ターゲット PGDB（Ubuntu）

```
1 PGDB:
2       date > start-PGDB.txt
3       /bin/bash replaceloadPostgreSQL-CDIR-cons.sh
4       sudo -u postgres psql < ./loadtoPostgreSQL-cons.sql
5       /bin/bash replaceloadPostgreSQL-CDIR-uncons.sh
6       sudo -u postgres psql < ./loadtoPostgreSQL-uncons.sql
7       date > end-PGDB.txt
```

macOS 用のターゲット PGDB（スクリプト15）と，Ubuntu 用のもの（スクリプト20）を比較することによって，PostgreSQL との対話型インターフェース psql を実行する権限の仕様が若干異なっていることがわかる．これは，macOS と Ubuntu のそれぞれの PMS（brew，apt）でインストール

24）この結果は，iMac Pro 2018（macOS Big Sur）で計測したものである．

した PostgreSQL のバージョンと仕様に差異があるためである．ただし，データベースの構築のためには，macOS と同様に，ディレクトリ DBMaking（図 2 参照）をカレントとして，Ubuntu のターミナルで以下のように make コマンドを実行すればよい．

Ubuntu 上でターゲット PGDB の実行

```
$ make PGDB
```

この処理時間は，スクリプト20において，２，７行目の実行結果を比較することによってわかる．

Ubuntu 上でターゲット PGDB の処理時間の計測

```
masa@balin: DBMaking$ cat start-PGDB.txt
Sun  May 2 11:37:36 JST 2021
masa@balin: DBMaking$ cat end-PGDB.txt
Sun  May 2 11:38:28 JST 2021
```

この結果から，52秒であることがわかる[25]．

V　Orbis データの抽出

ここでは，Orbis データ（連結ベース）を実際に抽出する方法について述べる．まず，以下の手順によって抽出ページへアクセスする：

（Os1）SKWAD のトップページにアクセスする．
（Os2）Orbis のロゴ（アイコン Orbis ）をクリックする．
（Os3）Orbis データの抽出に関するアクセス認証に答える．
（Os4）移動したページのリンク Orbis2019 をクリックする．
（Os5）移動したページのリンク Consolidated Version をクリックする．

以上の手順によって Orbis データ（連結ベース）の抽出ページへアクセスす

25）この結果は，Dell Precision Tower 7910（Ubuntu 20.04）で計測したものである．

ることができる（図16参照）.

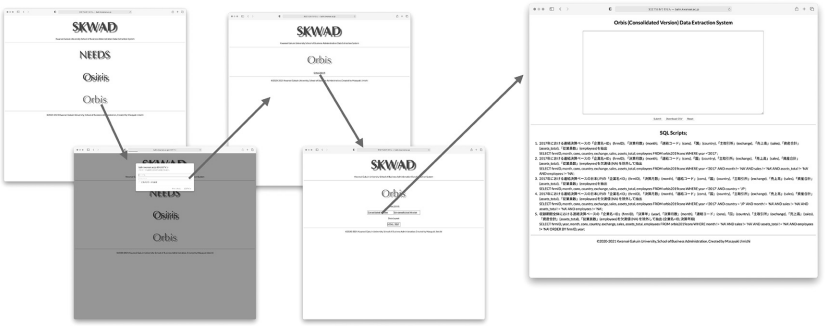

図16：Orbis データ（連結ベース）の抽出ページへのアクセス

この抽出ページの利用法は，地道（2021-a, b）における NEEDS 企業財務データや Osiris 財務データの抽出と同様であるが，以下に簡単に説明する（図17参照）.

まず，SQL 問合せのスクリプトを スクリプト入力ボックス に入力し，`Submit` ボタンをクリックすることによって，サーバに命令が送信され，結果が HTML 形式で返信される．次に，SQL 問合せのスクリプトを スクリプト入力ボックス に入力し，`Download CSV` ボタンをクリックすることによって，サーバに送信された命令の結果を CSV 形式でダウンロードすることができる．また，`Reset` ボタンをクリックすると スクリプト入力ボックス 内のスクリプトがクリアされる．なお， スクリプト入力ボックス の大きさはボックスの右隅にあるリサイズアイコン（//）を使って調整することが可能である.

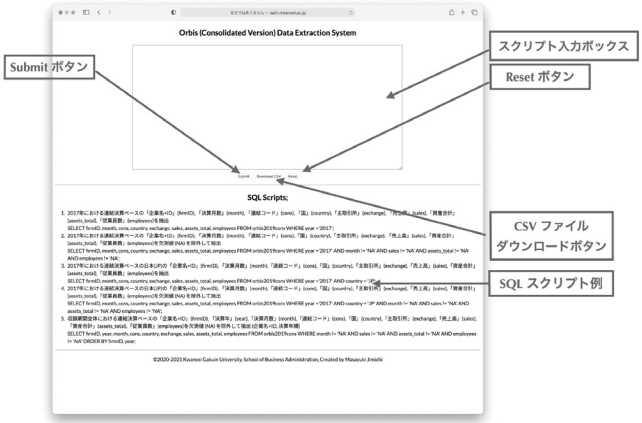

図17：Orbis データ（連結ベース）抽出システムのページ

なお，SQL のサンプルスクリプトも用意されている（図17の下部を参照）ので，コピー・アンド・ペーストすることによって，データ抽出を手軽に試すことができる．

例えば，SQL 問合せとして，連結ベースのデータベース（orbis2019cons）から，2017年における「売上高」（sales），「資産合計」（assets_total），「従業員数」（employees）などのデータを抽出することを考えよう．この抽出に利用される SQL 問合せをスクリプト21に与える．なお，列名の意味については，付録 B を参照されたい．

スクリプト21：連結ベースのデータベース（orbis2019cons）から，2017年におけるデータの抽出

```
1  SELECT firmID, month, cons, country, exchange, sales, assets_total,
       employees
2        FROM orbis2019cons
3        WHERE year ='2017';
```

この SQL 問合せを以下に説明する．

1 行目：SELECT 句で「企業名＋BvD ID[26]」（firmID），「決算月数」（month），

「連結コード」(cons),「国」(country),「主取引所」(exchange),「売上高」(sales),「資産合計」(assets_total),「従業員数」(employees)の列を指定する.

2行目：FROM句で連結ベースのデータが納められているテーブルorbis2019consを指定する.

3行目：WHERE句で条件として2017年の情報（year = '2017'）を与える.

ここで，シングルクォート（'）は条件の文字列を「包む」ための記号であり,「クォート処理」と呼ばれることがある.

次に，実際の抽出手順は以下のようなものである．ただし，SQL問合せ（スクリプト21）をそのまま実行すると，Webブラウザ上への表示に時間がかかるため，LIMIT 10; を追記することによって表示させる件数を10件に抑えていることに注意しよう（図18も参照）：

(Or-E1) SQL問合せ（最初の例）のスクリプトをコピーし，スクリプト入力ボックスへペーストし，さらにLIMIT 10; を追記する.

(Or-E2) Submit ボタンをクリックする.

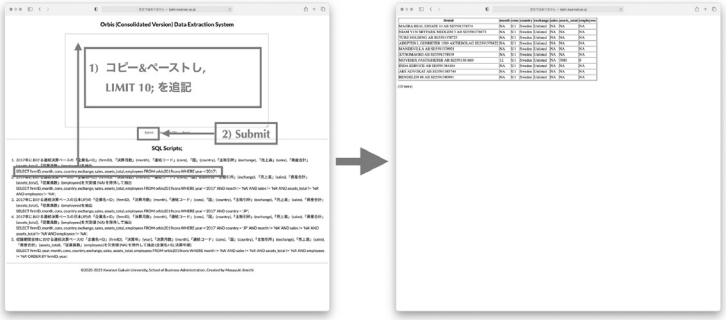

図18：Orbisデータ（連結ベース）の抽出

26) BvD IDはBvD社が各企業に与えた一意のコードである.

この手順を実行することによって，図18の右側の図にあるように抽出結果が表示される．

以上の抽出結果は10件分のデータのみであるが，SQL 問合せ自体には問題がないので，制限を外して実行することを考える．その際，結果全体を適当な表計算ソフトウェアへコピー＆ペーストすることによって可視化などを行うことも不可能ではないが，行数が多いため，表示に時間を要したり，操作のミスや結果の再現性を確保するために以下のように CSV ファイルとしてダウンロードする方法が推奨される（図19も参照）：

（Or-C1）SQL 問合せの最初の例のスクリプトをコピーし，スクリプト入力ボックスへペーストする．

（Or-C2）`Download CSV` ボタンをクリックする．

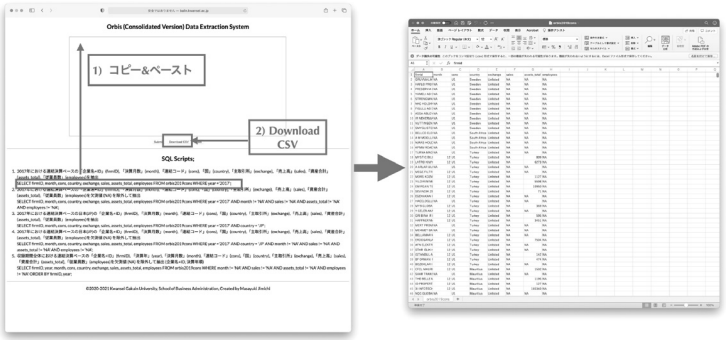

図19：Orbis データ（連結ベース）の抽出結果を CSV ファイル（**orbis2019cons.csv**）としてダウンロード

Ⅵ　Orbis データの可視化

ここでは，Ⅴ節で抽出した Orbis データの CSV ファイル `orbis2019cons.csv` をデータ解析環境 R[27] に読み込んだ後，可視化することを考える．

適当な場所（作業ディレクトリ）に保存された CSV ファイル orbis2019cons.csv を R に読み込むには，read.csv 関数を利用して以下のように入力すればよい[28]．

R へのデータの読み込み

```
> x <- read.csv("orbis2019cons.csv")
```

ここで > は R のプロンプトである．このように読み込まれたオブジェクトの先頭 6 行は関数 head を使って以下のように表示できる．

読み込んだデータオブジェクト x

```
> head(x)
                                firmid month cons country exchange sales assets_total employees
1           GRUVMALMEN GORM AB SE5591652499    NA   U1  Sweden Unlisted    NA           NA        NA
2         HAFLO PROPERTIES AB SE5591660740    NA   U1  Sweden Unlisted    NA           NA        NA
3 PRESERVIA BAALSTA HOLDING AB SE5591734040    NA   U1  Sweden Unlisted    NA           NA        NA
4                    YAMELI AB SE5591735153    NA   U1  Sweden Unlisted    NA           NA        NA
5               STRONGWALL AB SE5591741250    NA   U1  Sweden Unlisted    NA           NA        NA
6             NKC HOLDING AB SE5591766778    NA   U1  Sweden Unlisted    NA           NA        NA
```

ここで，変数名（カラム名）は以下のようなものである[29]：

firmid: 企業名＋BvD ID

month: 決算月数

cons: 連結コード

country: 国情報

exchange: 主取引所

sales: 売上高（単位：1,000 US ドル）

27）https://www.r-project.org

28）R を起動後，作業ディレクトリを適切に設定する必要がある．詳細は地道（2018-c）等を参照されたい．

29）一般に，リレーショナルデータベースは「表形式」のデータを扱い，その縦の並びはカラム（列）と呼ばれ，その名称としては，カラム名（column name）と呼ばれる．一方，R では，データフレーム（data frame）が表形式のデータを扱うデータ構造であり，カラムの名称は，変数（variable）と呼ばれる．なお，統計学的には（確率）変数は，変量（variate）であり，この意味から変量名（variable name）と呼ばれることもある．

assets_total: 資産合計（単位：1,000 US ドル）
employees: 従業員数（単位：人）

ここで，連結コードは以下のことを表している：

C1: 連結財務諸表のみを保有している企業
C2: 連結財務諸表を保有しており，何らかの理由で単体財務諸表も保有している企業
U1: 単体財務諸表のみを保有している企業
U2: 単体財務諸表を保有しており，何らかの理由で連結財務諸表も保有している企業

このオブジェクト x（データフレーム）に対して，以下のようなスクリプトを実行することによって対散布図をプロットする：

R による対散布図のプロット：ggpairs 関数を利用した場合

```
> library(dplyr)
> library(GGally)
> x %>% filter(month == 12, cons == "C1" | cons == "C2") %>%
+   select(sales, assets_total, employees, cons) %>% na.omit() %>%
+   ggpairs(mapping = aes(color = as.factor(cons)),
+           upper=list(continuous = wrap("points", size = 0.5, alpha = 0.5)),
+           lower=list(continuous = wrap("cor", size = 5))) +
+   theme(axis.text = element_text(size = 5),
+         axis.title = element_text(size = 10))
```

この R スクリプトでは，まず関数 library をつかって，dplyr パッケージと GGally パッケージを呼び出している．次に，データ・フレーム・オブジェクト x の行を filter 関数を利用して連結ベースと決算月数が12ヶ月だけのものに限定（month == 12, cons == "C1" | cons == "C2"）し，select 関数で列（sales, assets_total, employees, cons）を選択した後，欠測値を取り除いている（na.omit()）．さらに，関数 ggpairs によって対散布図を描いている．ここでは，連結コードで色分け（mapping = aes(color =as.factor(cons))）しており，引数 upper（上三角ブ

ロック）と lower（下三角ブロック）に，それぞれ，散布図の点（"point"）と相関係数の値（"cor"）を与えることを点と文字の大きさとともに指定している．これらの工程は，dplyr パッケージのパイプ演算子 %>% も利用して，パイプラインを構成することによって実現している．

対散布図（図20）から，売上高（sales），資産合計（assets_total），従業員数（employees）の全てが極端に右に歪んだものであることがわかる．また，これらの変数（変量）のペアの散布図も原点付近に密集しており，2次元の意味で歪んだものであることがわかる．さらに，連結コードで色分けすることによって，連結財務諸表と単体財務諸表の両方を保有している企業（C2）が少ないことがわかり，それぞれの企業によって相関係数の値が異なっているもの（売上高と資産合計）があることもわかる．

このように，R を利用して可視化することによって，ここで扱っている財務データの特性を詳細に捉えることができる．また，この結果はデータと SQL 問合せと R のスクリプト（コード）を適切に管理することによって，簡単に再現できることも利点といえる[30]（図21も参照）．

30）本稿は，Sweave（https://stat.ethz.ch/R-manual/R-devel/library/utils/doc/Sweave.pdf）を利用することによって，LaTeX に R のコードを埋め込み，自動実行することによって動的に文書を生成する方法で作成している．

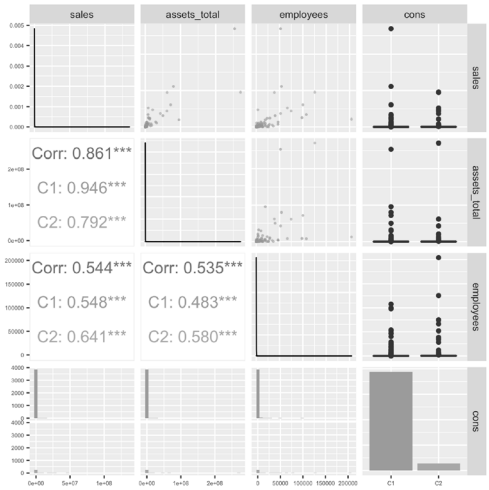

図20：対角ブロックには売上高（**sales**），資産合計（**assets_total**），従業員数（**employees**）に関する推定された密度関数と連結コード（**C1**，**C2**）別の企業数がバーチャートで描かれている．また，上三角ブロックには，売上高（**sales**），資産合計（**assets_total**），従業員数（**employees**）の２つの組合せに関する散布図と連結コード別のボックスプロットが描かれている．，下三角ブロックには，売上高（**sales**），資産合計（**assets_total**），従業員数（**employees**）の２つの組合せに関する相関係数と連結コード別の各指標のヒストグラムが与えられている．

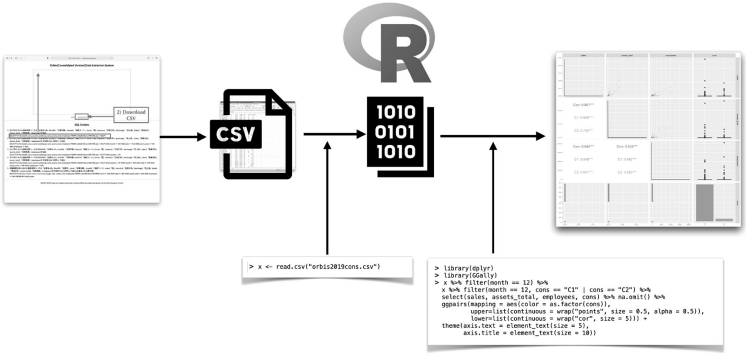

図21：CSV ファイルとしてダウンロードされた Orbis データファイルを R を利用して可視化する工程

Ⅶ おわりに

本稿では，世界の企業の財務情報を収録したデータベース Orbis から抽出したデータを利用した抽出システムの構築と利用について述べた．Ⅰ節の脚注でも言及したが，今回データ抽出システム SKWAD から提供される Orbis データを利用した抽出サービスは，筆者のグループが現在行っている研究の共同利用を目的としたものであることを強調しておきたい．前身のシステムを利用してこれまで行ってきた共同研究（Jimichi and Maeda, 2014, 柳ら, 2018-a, b, 2019, Shih *et al*., 2020 等）よりもまして，新システム SKWAD から開始されるサービスの充実が共同研究の活性化につながることが期待される．

（筆者は関西学院大学商学部教授）

参考文献

［1］ 地道正行（2010-a）『日経 NEEDS 財務データにもとづくデータベースサーバの構築』，商学論究，第57巻，第 4 号，pp. 23-80，関西学院大学商学研究会.
［2］ 地道正行（2010-b）『財務データベースサーバの構築』，関西学院大学リポジトリ，http://hdl.handle.net/10236/6013，ISBN: 9784990553005
［3］ 地道正行（2018-a）『探索的財務ビッグデータ解析－前処理，データラングリング，再現可能性－』，商学論究，第66巻，第 1 号，pp. 1-32，関西学院大学商学研究会.
［4］ 地道正行（2018-b）『探索的財務ビッグデータ解析－データ可視化，統計モデリング，モデル選択，モデル評価，動的文書生成，再現可能研究－』，商学論究，第66巻，第 2 号，pp. 1-41，関西学院大学商学研究会.
［5］ 地道正行（2018-c）『データサイエンスの基礎：R による統計学独習』，裳華房.
［6］ 地道正行（2020-a）『探索的財務ビッグデータ解析－前処理の並列化－』，商学論究，第67巻，第 3 号，pp. 1-19，関西学院大学商学研究会.
［7］ 地道正行（2020-b）『探索的財務ビッグデータ解析－PG-Strom によるデータラングリングの並列化－』，商学論究，第68巻，第 1 号，pp. 1-34，関西学院大学商学研究会.
［8］ 地道正行（2021-a）『財務データ抽出システムの再構築－NEEDS 企業財務データを中心に－』，商学論究，第69巻，第 3 号，pp. 1-78，関西学院大学商学研究会.
［9］ 地道正行（2021-b）『財務データ抽出システムの再構築－Osiris データの利用－』，商学論究，第70巻，第 1 号，pp. 71-109，関西学院大学商学研究会.
［10］ 地道正行（2021-c）『SKWAD ユーザマニュアル－NEEDS 企業財務データの抽出－』，Ver. 1.0，pp. 1-88，関西学院大学リポジトリ，http://hdl.handle.net/10236/

00029654

［11］ Jimichi, M. and S. Maeda（2014）Visualization and Statistical Modeling of Financial Data with R, Poster at *The R User Conference 2014*, University of California, Los Angeles, USA, July 1st, 2014.

［12］ 地道正行，阪智香（2020-a）『財務データ抽出システム KGUSBADES の再構築』，国際数理科学協会，2020年度年会「統計的推測と統計ファイナンス」分科会研究集会，大阪大学，オンライン開催，2020年 8 月22日（土），配付資料.

［13］ 地道正行，阪智香（2020-b）『学内向け財務データ抽出システムの再構築』，日本計算機統計学会，第34回シンポジウム，オンライン開催，2020年11月29日（日），講演論文集，pp. 123-126.

［14］ 地道正行，阪智香（2021）『財務データと ESG レーティングデータの前処理と結合』，商学論究，第69巻，第 3 号，pp. 79-116，関西学院大学商学研究会.

［15］ 地道正行，宮本大輔，阪智香，永田修一（2020-a）『探索的財務ビッグデータ解析－PG-Strom によるデータラングリングの並列化－』，日本計算機統計学会，第34回大会，オンライン開催，2020年 5 月31日（日），講演論文集，pp. 41-44.

［16］ 地道正行，宮本大輔，阪智香，永田修一（2020-b）『財務ビッグデータの可視化と統計モデリング』，学際大規模情報基盤共同利用・共同研究拠点（JHPCN），第12回シンポジウム，オンライン開催，2020年 7 月 9 日（木），発表用ポスター.

［17］ 地道正行，宮本大輔，阪智香，永田修一（2020-c）『探索的財務ビッグデータ解析－PG-Strom によるデータラングリングの並列化－』，国際数理科学協会，2020年度年会「統計的推測と統計ファイナンス」分科会研究集会，大阪大学，オンライン開催，2020年 8 月22日（土），配付資料.

［18］ 増永良文（2017）『リレーショナルデータベース入門－データモデル・SQL・管理システム・NoSQL－』，サイエンス社.

［19］ 西沢夢路（2017）『基礎からの MySQL 第 3 版』，SB クリエイティブ.

［20］ 鈴木啓修（2012）『PostgreSQL 全機能バイブル』，技術評論社.

［21］ Shih, J, T. Lin, M. Jimichi, and T. Emura（2020）Robust ridge M-estimators with pretest and Stein-rule shrinkage for an intercept term, *Japanese Journal of Statistics and Data Science*, DOI : 10.1007/s42081-020-00089-6.

［22］ Tange, Ole,（2018）*GNU Parallel 2018*, ISBN : 9781387509881, DOI : 10.5281/zenodo.1146014, URL: https://doi.org/10.5281/zenodo.1146014

［23］ Wickham, H. and G. Grolemund（2016）*R for Data Science*, O'Reilly.

［24］ 柳麻衣，阪智香，地道正行（2018-a）『配当金支払金額の探索的データ解析』，国際数理科学協会2018年度年会，「統計的推測と統計ファイナンス」分科会研究集会，関西学院大学梅田キャンパス1402号教室，2018年 8 月25日（土）.

［25］ 柳麻衣，阪智香，地道正行（2018-b）『配当金支払金額の探索的データ解析』，2018年度統計関連学会連合大会，中央大学後楽園キャンパス，2018年 9 月12日（水）.（発表要旨，http://www.jfssa.jp/taikai/2018/table/program_detail/pdf/51-

100/J10096.pdf）
［26］ 柳麻衣，阪智香，地道正行（2019）『配当金の探索的データ解析』，第13回日本統計学会春季集会，ポスター発表，日本大学経済学部本館，2019年３月10日（日）．

謝辞

本研究の一部は以下の助成を得ている．

科研費 KAKENHI 科学研究費基盤研究 C：「グラフィカル・データ・アナリシスによる格差研究と社会環境会計による解決方法の提案」（2016年～2019年），課題番号：16K04022

科研費 KAKENHI 科学研究費基盤研究 C：「共有価値創造（CSV）のための社会環境会計の構築」（2019年～2021年），課題番号：19K02006

JHPCN 2018年度学際大規模情報基盤共同利用・共同研究拠点（JHPCN）課題：「財務ビッグデータの可視化と統計モデリング」，課題番号：jh181001-NWJ

JHPCN 2019年度学際大規模情報基盤共同利用・共同研究拠点（JHPCN）課題：「財務ビッグデータの可視化と統計モデリング」，課題番号：jh191002-NWJ

JHPCN 2020年度学際大規模情報基盤共同利用・共同研究拠点（JHPCN）課題：「財務ビッグデータの可視化と統計モデリング」，課題番号：jh201003-NWJ

JHPCN 2021年度学際大規模情報基盤共同利用・共同研究拠点（JHPCN）課題：「財務ビッグデータの可視化と統計モデリング」，課題番号：jh211001-NWJ

関西学院大学図書館図書費 B，研究設備費（III），個人研究費

また，BvD 社増田歩氏，JHPCN プロジェクト[31] 共同研究メンバーである東京大学宮本大輔准教授，関西学院大学商学部阪智香教授，永田修一准教授，並びに，関西学院大学商学部研究資料室高瀬忍氏には様々なご足労を賜った．ここに感謝の意を表する．

31）https://jhpcn-kyoten.itc.u-tokyo.ac.jp/abstract/jh211001-NWJ

付録 A　R に関する環境

本稿を執筆する時点での R に関する情報を与える．

sessionInfo の実行結果

```
> sessionInfo()
R version 4.1.0 (2021-05-18)
Platform: x86_64-apple-darwin17.0 (64-bit)
Running under: macOS Big Sur 10.16

Matrix products: default
BLAS:   /Library/Frameworks/R.framework/Versions/4.1/Resources/lib/libRblas.dylib
LAPACK: /Library/Frameworks/R.framework/Versions/4.1/Resources/lib/libRlapack.dylib

locale:
[1] ja_JP.UTF-8/ja_JP.UTF-8/ja_JP.UTF-8/C/ja_JP.UTF-8/ja_JP.UTF-8

attached base packages:
[1] stats     graphics  grDevices utils     datasets  methods   base

other attached packages:
[1] GGally_2.1.1  ggplot2_3.3.3 dplyr_1.0.6

loaded via a namespace (and not attached):
 [1] Rcpp_1.0.6         magrittr_2.0.1     tidyselect_1.1.1   munsell_0.5.0
 [5] colorspace_2.0-1   R6_2.5.0           rlang_0.4.11       fansi_0.4.2
 [9] plyr_1.8.6         tools_4.1.0        grid_4.1.0         gtable_0.3.0
[13] utf8_1.2.1         DBI_1.1.1          withr_2.4.2        ellipsis_0.3.2
[17] digest_0.6.27      assertthat_0.2.1   tibble_3.1.2       lifecycle_1.0.0
[21] crayon_1.4.1       farver_2.1.0       RColorBrewer_1.1-2 purrr_0.3.4
[25] vctrs_0.3.8        glue_1.4.2         labeling_0.4.2     compiler_4.1.0
[29] pillar_1.6.1       generics_0.1.0     scales_1.1.1       reshape_0.8.8
[33] pkgconfig_2.0.3
```

付録 B　Orbis データ

本稿で構築したデータベース orbis2020cons，orbis2020uncons におけるテーブル orbis2020cons，orbis2020uncons のカラム名の一覧を以下に与える[32]：

32）PostgreSQL の仕様から，抽出時のカラム名は，全て小文字となることに注意が必要である．

表 5 ：Osiris データの仕様

No.	カラム名	説明	カラム名（オリジナル）
1	firm	企業名	NA
2	year_usd	年（通貨単位）	year（USD）
3	country	国	Country
4	city	市	City
5	postcode	郵便番号	Postcode
6	tel	電話番号	Telephone number
7	id	企業コード	BvD ID number
8	nid_num	法人企業ナンバー	National ID number
9	nid_lab	法人企業ラベル	National ID label
10	nid_type	法人企業タイプ	National ID type
11	ip_num	インフォメーションプロバイダーナンバー	IP identification number
12	ip_lab	インフォメーションプロバイダーラベル	IP identification label
13	isin_num	国際証券コード	ISIN number
14	tick_symb	国内証券コード	Ticker symbol
15	exchange	主取引所	Main exchange
16	listed	上場・非上場	Listed/Delisted/Unlisted
17	cons	連結・単体	Consolidation code
18	date	決算日	Closing date
19	month	月数	Number of months
20	audit	監査	Audit status
21	practice	会計基準	Accounting practice
22	source	データの出所	Source（for publicly quoted companies）
23	units	単位（金額）	Original units
24	currency	現地通貨	Original currency
25	exchange_rate	換算レート	Exchange rate from original currency
26	assets_fix	固定資産	Fixed assets
27	assets_int	無形固定資産	Intangible fixed assets
28	assets_tang	有形固定資産	Tangible fixed assets
29	assets_other_fix	その他の固定資産	Other fixed assets
30	assets_cur	流動資産	Current assets
31	stock	株式	Stock
32	debtors	売掛金	Debtors
33	assets_other_cur	その他の流動資産	Other current assets
34	cash	現金及び現金同等物	Cash & cash equivalent
35	assets_total	資産合計	Total assets
36	shareholders	株主資本	Shareholders funds
37	capital	資本	Capital
38	shareholders_other	その他の株主資本	Other shareholders funds
39	liabilities_non_cur	非流動負債	Non-current liabilities
40	debt_long	固定負債	Long term debt
41	liabilities_other_non_cur	その他の非流動負債	Other non-current liabilities
42	provisions	引当金	Provisions
43	liabilities_cur	流動負債	Current liabilities
44	loans	借入金	Loans
45	creditors	買掛金	Creditors
46	liabilities_other_cur	その他の流動負債	Other current liabilities
47	total_s_l	負債純資産合計	Total shareh. funds & liab.
48	capital_working	運転資本	Working capital
49	assets_net_cur	正味流動資産	Net current assets
50	enterprise_value	企業価値	Enterprise value

51	employees	従業員数	Number of employees
52	operating_revenue	営業収益	Operating revenue（Turnover）
53	sales	売上高	Sales
54	costs_goods	売上原価	Costs of goods sold
55	profit_gross	売上総利益	Gross profit
56	expenses_other	その他の営業費用	Other operating expenses
57	ebit	営業利益	Operating P/L［＝EBIT］
58	fin_revenue	金融収益	Financial revenue
59	fin_expenses	金融費用	Financial expenses
60	fin_pl	金融収益−金融費用	Financial P/L
61	pl_before_tax	税引前利益	P/L before tax
62	tax	税金	Taxation
63	pl_after_tax	税引後利益	P/L after tax
64	extr_rev	特別利益	Extr. and other revenue
65	extr_exp	特別損失	Extr. and other expenses
66	pl_extr	特別収支	Extr. and other P/L
67	net_income	純利益	P/L for period［＝Net income］
68	export_rev	海外売上高	Export revenue
69	costs_material	原材料費	Material costs
70	costs_employees	人件費	Costs of employees
71	depr_amor	減価償却及び減耗償却	Depreciation & Amortization
72	operating_items_other	その他営業項目	Other Operating Items
73	interest_paid	支払利息	Interest paid
74	r_d	研究開発費	Research & Development expenses
75	cash_flow	キャッシュフロー	Cash flow
76	add_value	付加価値	Added value
77	ebitda	EBITDA	EBITDA
78	sic_code1	SIC プライマリコード US	SIC, Primary code(s)
79	sic_text1	SIC プライマリコード（説明）	Ibid, text description
80	sic_code2	SIC セカンダリコード US	SIC, Secondary code(s)
81	sic_text2	SIC セカンダリコード（説明）	Ibid, text description
82	bvdmajor	BvD 主業種名	BvD major sector
83	infoprov	情報提供元	Information provider
84	date_inc	創業年月日	Date of Incorpotrate
85	date_ipo	上場年月日	IPO Date
86	date_delisted	上場廃止年月日	Delisted Date
87	firmid	企業名＋BvD ID	NA
88	year	年	NA