財務データ抽出システムの再構築
—Osiris 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）で扱われた学内向け財務データ抽出システム SKWAD（スクワッド）に対して，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，世界の上場企業の財務情報を収録したデータベース Osiris から抽出されたデータである．SKWAD からデータを抽出する方法と，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），Osiris 財務データ（Osiris 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 Osiris から抽出されたデータを地道，阪（2020-a, b），地道（2021）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]．

本稿の構成は以下のようなものである．まず，Ⅱ節では，システム構築を行うための環境について紹介する．次に，Ⅲ節では，今回利用するデータと

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，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．

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．また，Ⅳ節では，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[3]．さらに，Ⅴ節では，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，Ⅵ節で，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．最後に，Ⅶ節で今後の展望について述べる．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

地道，阪（2020-a, b），地道（2021）でも議論したが，ネットワーク上でデータベースを利用したオーソドックスなシステム構成としては，Ubuntu (Linux), macOS といった 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 Database 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[4] を利用したものである．OS として，macOS, Ubuntu を利用し，RDBMS として MySQL と PostgreSQL を選択した場合のシステム構成は以下のような表にまとめられる：

表 1 ：MAMP, MAPP, LAMP, LAPP 環境

OS \ RDBMS	MySQL	PostgreSQL
macOS	MAMP	MAPP
Ubuntu	LAMP	LAPP

例えば，LAPP の場合は，OS として Ubuntu（Linux 系 OS）上で，Web サーバ Apache HTTP Server（`httpd`）と RDBMS である PostgreSQL[5] を

3） 抽出システム SKWAD の Osiris 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．
4） `https://www.php.net/`
5） `https://www.postgresql.org/`

導入し，PHP でそれらを連携しながら運用することを意味している．各システム構成の概念図を図 1 に与える．

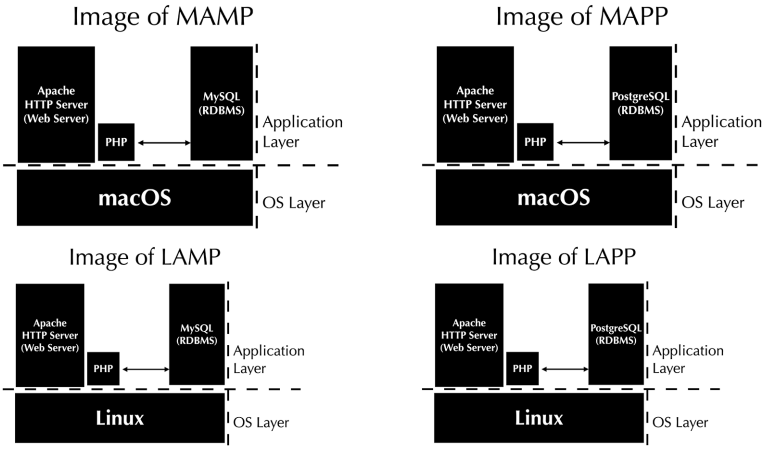

図 1 ：MAMP，MAPP，LAMP，LAPP 環境のイメージ

これらのシステムに対応する具体的な構成として，本稿では表 2 のような開発環境を用意した[6]：

表 2 ：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 Osiris（オシリスまたはオサイリス）

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．

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 Osiris は，世界の「上場企業」（データ抽出時点で9万社以上を収録）の情報が国際比較可能な統一のフォームで納められたデータベースである．

本稿では，Osiris から，データセットとして，主要財務情報を最長30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96,377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96,377社（以後，「非連結ベース」と略す）

データセットのファイルサイズは，1.6 GB 程度の TSV ファイル[7]（2セット）からなり，表3，表4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．

表3 ：Osiris データセット：仕様

データセット名	年月版	データベース名	上場情報	抽出主体	抽出期間	抽出指標数
DS-Osiris-C-2020	2020年3月版	Osiris	上場（上場廃止企業含む）	連結優先	30年	91
DS-Osiris-U-2020	2020年3月版	Osiris	上場（上場廃止企業含む）	非連結優先	30年	91

表4 ：Osiris データセット：サイズ

データセット名	社数	行数	粗データファイル	トータルサイズ
DS-Osiris-C-2020	96,377	2,987,687	SJ_Project_2020_OS_C_96377.asc	約 1.6 GB
DS-Osiris-U-2020	96,377	2,987,687	SJ_Project_2020_OS_U_96377.asc	約 1.6 GB

地道，阪（2021）では，連結優先で抽出したデータセット DS-Osiris-C-2020 を GNU parallel（Tange, 2018 参照）によって並列化することにより前処理の工程を効率的に行うことが議論されている[8]．なお，処理後のファイル

7) TSV（Tab Sepelated Values）ファイルとは，項目（カラム）間がタブ区切りのテキスト形式のファイルである．

8) 2018年に Osiris から抽出したデータセットに対して前処理を実行し，探索的データ解析を再現可能研究の観点から実行することに関しては，地道（2018-a, b）を参照されたい。

（CSV[9] 形式）のサイズは，DS-Osiris-C-2020（連結ベース），DS-Osiris-U-2020（非連結ベース）のそれぞれの場合に対して表 5 のようなものである．

表 5 ：前処理後の Osiris 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-Osiris-C-2020	96,377	2,891,311	94	`firmfinC2020.csv`	約 1.5 GB
DS-Osiris-U-2020	96,377	2,891,311	94	`firmfinU2020.csv`	約 1.5 GB

なお，抽出時の指標数が91に対して，処理後のファイルの列数が94になっているのは，前処理の工程で社名（`firm`），社名＋BvD ID（`firmID`），通貨換算年情報（`year`）を追加したためである．

Ⅳ　Osiris データによるデータベースの構築

1. Osiris 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2020年 3 月版から抽出されたデータセット DS-Osiris-C-2020（連結ベース），DSOsiris-U-2020（非連結ベース）を前処理した，それぞれのファイル `firmfinC2020.csv`, `firmfinU2020.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96,377社の約10％にあたる 1 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ような仕様とした．

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[10]．

9）CSV（Comma Sepelated Values）ファイルとは，項目（カラム）間がコンマ区切りのテキスト形式のファイルである．

10）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．

```
osiris2020
├──── CSV
│     ├──── firmfinC2020.csv
│     └──── firmfinU2020.csv
└──── DBMaking
      ├──── Makefile
      ├──── loadtoMySQLgen-cons.sql
      ├──── loadtoMySQLgen-uncons.sql
      ├──── loadtoPostgreSQLgen-cons.sql
      ├──── loadtoPostgreSQLgen-uncons.sql
      ├──── replaceloadMySQL-CDIR-cons.sh
      ├──── replaceloadMySQL-CDIR-uncons.sh
      ├──── replaceloadPostgreSQL-CDIR-cons.sh
      ├──── replaceloadPostgreSQL-CDIR-uncons.sh
      ├──── rs10000-osiris2020.R
      └──── 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：

Osiris データセットにもとづくデータベース構築手順

（Os-S1）前処理
　（1）ファイル firmfinC2020.csv におけるバックスラッシュ（\）処理
　（2）ファイル firmfinU2020.csv におけるバックスラッシュ（\）処理
（Os-S2）ランダムサンプリング
　（1）10,000社の企業データをランダムサンプリングし，ファイル firmfinC2020-10000.csv へ出力
　（2）10,000社の企業データをランダムサンプリングし，ファイル firmfinU2020-10000.csv へ出力
（Os-S3）データベース構築
　（1）データベース osiris2020cons 及びテーブル作成 osiris2020cons の作成と，データファイル firmfinC2020-10000.csv をテーブル osiris2020cons へロード
　（2）データベース osiris2020uncons 及びテーブル作成 osiris2020uncons の作成と，データファイル firmfinU2020-10000.csv をテーブル osiris2020uncons へロード

上記の手順は，合計 4 種類の環境（MAMP, MAPP, LAMP, LAPP）について実施する必要がある（表 1 と図 1 参照）.

以下に手順（Os-S1），（Os-S2），（Os-S3）について詳しく述べる．なお，手順（Os-S1），（Os-S2）は，macOS と Ubuntu の両方の環境で共通であり，差異が生じるのは手順（Os-S3）であることに注意しよう.

2. Osiris データの前処理

ここでの前処理は，ファイル firmfinC2020.csv, firmfinU2020.csv

のバックスラッシュ（\）を二重バックスラッシュ（\\）へ置換することである[11]．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（Os-S1）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 1 参照）．

スクリプト 1 ：Makefile: ターゲット preprocess

```
1 preprocess:
2       date > start-preprocess.txt
3       sed -f sedbs ../CSV/firmfinC2020.csv > ./firmfinC2020.csv
4       sed -f sedbs ../CSV/firmfinU2020.csv > ./firmfinU2020.csv
5       date > end-preprocess.txt
```

まず，スクリプト 1 における 2 行目と 5 行目は処理時間を計測するための指定である．また，置換のためのルールをファイル sedbs（スクリプト 2 ）に定義しておき， 3 行目と 4 行目で sed を利用して置換を行っている[12]．

スクリプト 2 ：sed による置換のためのルールファイル 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 3 に与える．

11）データベース Osiris から抽出されたデータセットの前処理については，地道（2018-a）や地道，阪（2021）を参照されたい．

12）ここでの処理は，sed コマンドの引数に直接与える方法もあるが，このような処理では往々にして複数の置換が必要にある場合があるため，本稿では，置換のためのルールをファイル sedbs に定義する方法をとった．

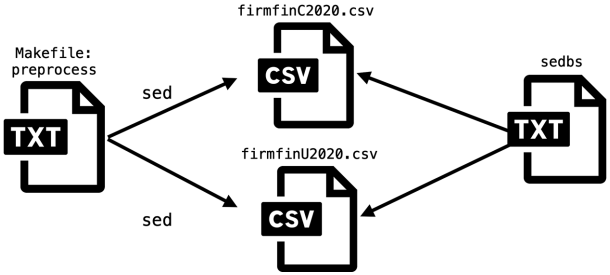

図 3 ：Osiris データファイルの前処理に関するシェルスクリプトの流れ

また，Osiris 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 4 に与える．

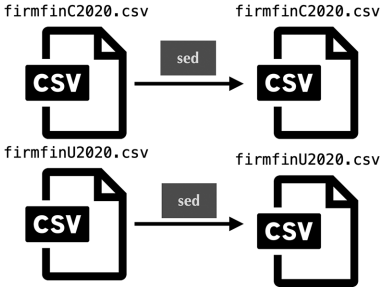

図 4 ：Osiris データの前処理におけるデータファイルの流れ

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 5 に与える．

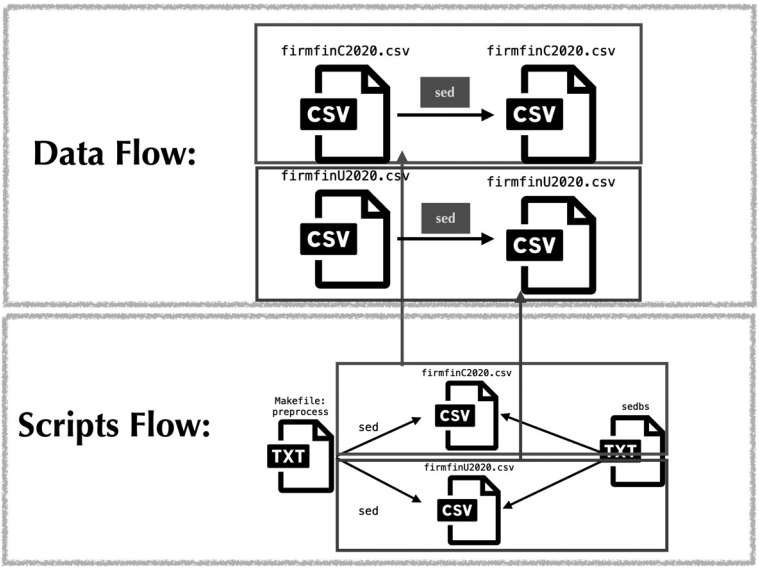

図 5 ：Osiris データの前処理におけるシェルスクリプトとデータに関するファイルの流れ

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 1 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
Fri Mar 19 13:43:37 JST 2021
masa@aule DBMaking % cat end-preprocess.txt
Fri Mar 19 13:43:44 JST 2021
```

この結果から，7秒であることがわかる[13]．

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
Mon Mar 15 14:35:47 JST 2021
masa@balin: DBMaking$ cat end-preprocess.txt
Mon Mar 15 14:35:59 JST 2021
```

この結果から，12秒である[14]．

3. Osiris データからのランダムサンプリング

ここでの処理は，ファイル firmfinC2020.csv, firmfinU2020.csv に収録されている企業からランダムサンプリングした10,000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．この処理，すなわち手順（Os-S2）を行うための Makefile におけるターゲットが rs（Random Sampling）である（スクリプト 3 参照）．

スクリプト 3 ：Makfile: ターゲット rs

```
1 rs:
2       date > start-rs.txt.
3       Rscript ./rs10000-osiris2020.R
4       date > end-rs.txt
```

まず，スクリプト 3 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，ランダムサンプリングを実行するための R スクリプトをファイル rs10000-osiris2020.R（スクリプト 4）に定義しておき，3 行目で Rscript コマンドを利用して実行している．

13）この結果は，iMac Pro 2018（macOS Big Sur）で計測したものである．
14）この結果は，Dell Precision Tower 7910（Ubuntu 20.04）で計測したものである．

スクリプト 4 ：**rs10000-osiris2020.R**

```
library(readr)
library(dplyr)
#
x <- read_csv("firmfinC2020.csv")
idx <- unique(x$ID)
set.seed(12345)
idx10000 <- sample(idx, 10000, replace = FALSE)
x %>% filter(ID %in% idx10000) %>%
  write_csv(file = "firmfinC2020-10000.csv")
#
y <- read_csv("firmfinU2020.csv")
idy <- unique(y$ID)
set.seed(12345)
idy10000 <- sample(idy, 10000, replace = FALSE)
y %>% filter(ID %in% idy10000) %>%
  write_csv(file = "firmfinU2020-10000.csv")
```

ターゲット 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 6 に与える．

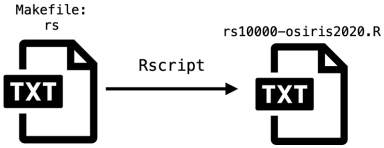

図 6 ：Osiris データファイルのランダムサンプリングに関する
シェルスクリプトの流れ

また，ランダムサンプリングにともなう Osiris データファイルの流れを可視化したものを図 7 に与える．

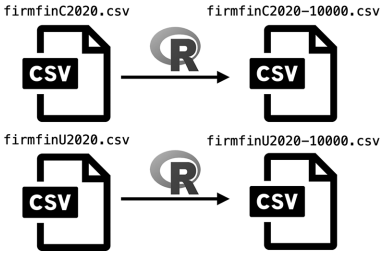

図 7 ：Osiris データのランダムサンプリングによるデータファイルの流れ

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 8 に与える．

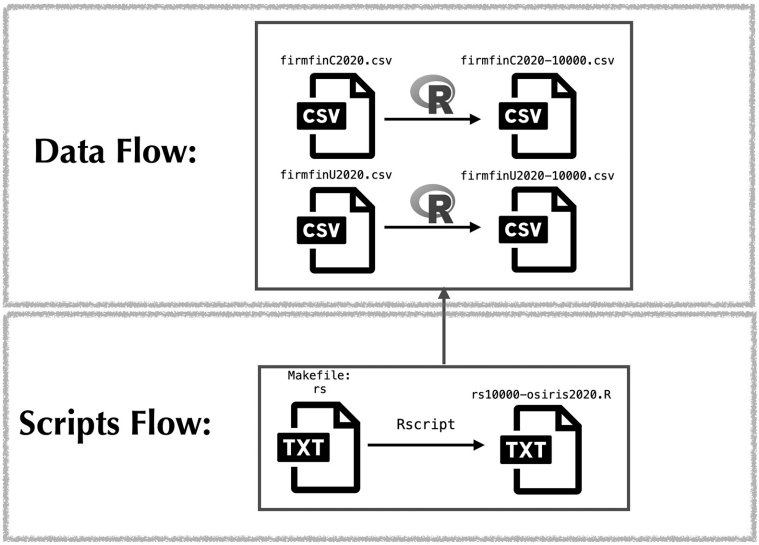

図 8 ：Osiris データのランダムサンプリングよるシェルスクリプトとデータに関するファイルの流れ

なお，この処理は，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．

ターゲット rs の実行: macOS, Ubuntu 共通

```
% make rs
```

この処理時間は，スクリプト 3 の 2 行目と 4 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 rs の処理時間の計測

```
masa@aule DBMaking % cat start-rs.txt
Sun Mar 21 11:44:08 JST 2021
masa@aule DBMaking % cat end-rs.txt
Sun Mar 21 11:45:06 JST 2021
```

この結果から，58秒であることがわかる[15]．

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 rs の処理時間の計測

```
masa@balin: DBMaking$ cat start-rs.txt
Mon Mar 15 14:35:59 JST 2021
masa@balin: DBMaking$ cat end-rs.txt
Mon Mar 15 14:37:14 JST 2021
```

この結果から，1分15秒である[16]．

4. Osiris 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[17]．

■macOS（MAMP）環境のもとでのデータベース構築 手順（Os-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 5 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．

15）この結果は，iMac Pro 2018（macOS Big Sur）で計測したものである．
16）この結果は，Dell Precision Tower 7910（Ubuntu 20.04）で計測したものである．
17）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方が異なっているためである．

スクリプト 5 ：**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 -p********* < ./loadtoMySQL-cons.sql
5       /bin/bash replaceloadMySQL-CDIR-uncons.sh
6       mysql -u root -p********* < ./loadtoMySQL-uncons.sql
7       date > end-MSDB.txt
```

スクリプト 5 の 3 ， 4 行目が手順（Os-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（Os-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 9 に与える．

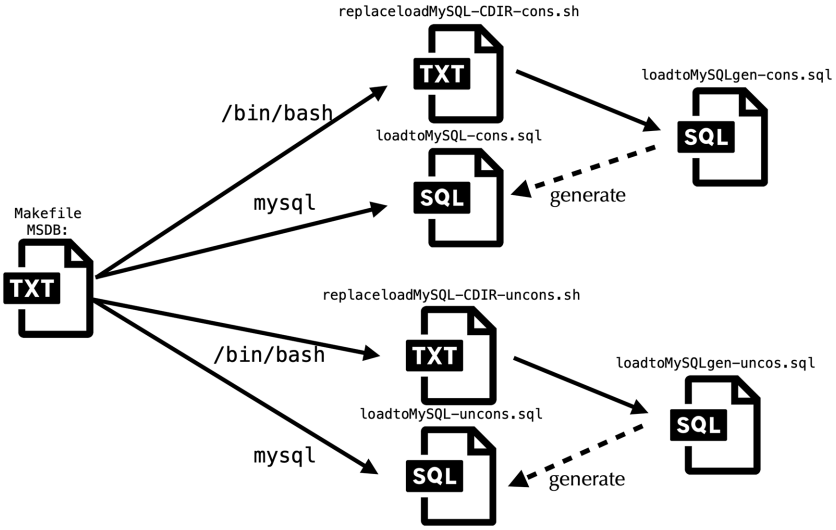

図 9 ：ターゲット **MSDB** の実行にともなうスクリプトファイルの流れ

まず，手順（Os-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-cons.sh の内容（スクリプト 6 ）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[18] スクリプトファイル loadtoMySQLgen-cons.sql

（スクリプト7）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7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9参照）．

スクリプト6：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7：loadtoMySQLgen-cons.sql（一部抜粋）

```
1  DROP DATABASE IF EXISTS osiris2020cons;
2  CREATE DATABASE osiris2020cons;
3  USE osiris2020cons;
4  DROP TABLE IF EXISTS osiris2020cons;
5  CREATE TABLE osiris2020cons (
6  firm VARCHAR(350),
7  :
8  : (中略)
9  :
10 year VARCHAR(4)
11 );
12 LOAD DATA LOCAL INFILE 'CDIR/firmfinC2020-10000.csv'
13 INTO TABLE osiris2020cons
14 FIELDS TERMINATED BY ','
15 ENCLOSED BY '"'
16 IGNORE 1 LINES;
```

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 osiris2020cons と，テーブル osiris2020cons を作成（2～11行目）した後，データファイル firmfinC2020-1000.csv から，テーブ

18）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．

ル osiris2020cons 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 5）の 4 行目で実行される．

次に，手順（Os-S3）の（2）を実現するためのスクリプトとして，Make-file のターゲット MSDB（スクリプト 5）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 8）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 9）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 9 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 9 参照）．

スクリプト 8 ：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 9 ：loadtoMySQLgen-uncons.sql（一部抜粋）

```
1  DROP DATABASE IF EXISTS osiris2020uncons;
2  CREATE DATABASE osiris2020uncons;
3  USE osiris2020uncons;
4  DROP TABLE IF EXISTS osiris2020uncons;
5  CREATE TABLE osiris2020uncons (
6  firm VARCHAR(350),
7  :
8  : (中略)
9  :
10 year VARCHAR(4)
11 );
12 LOAD DATA LOCAL INFILE 'CDIR/firmfinU2020-10000.csv'
13 INTO TABLE osiris2020uncons
14 FIELDS TERMINATED BY ','
15 ENCLOSED BY '"'
16 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 osiris2020uncons と，テーブル osiris2020uncons を作成（2～11行目）した後，データファイル firmfinU2020-1000.csv から，テーブル osiris2020uncons 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5）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10に与える．

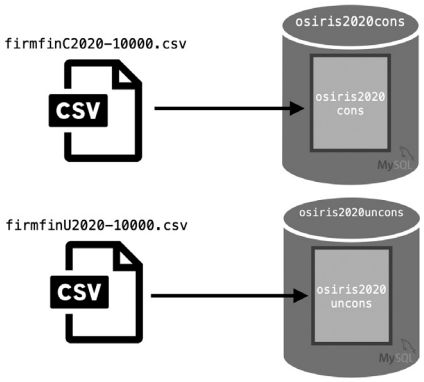

図10：MySQL よる Osiris データにもとづくデータベース osiris2020cons，osiris2020uncons の構築

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11に与える．

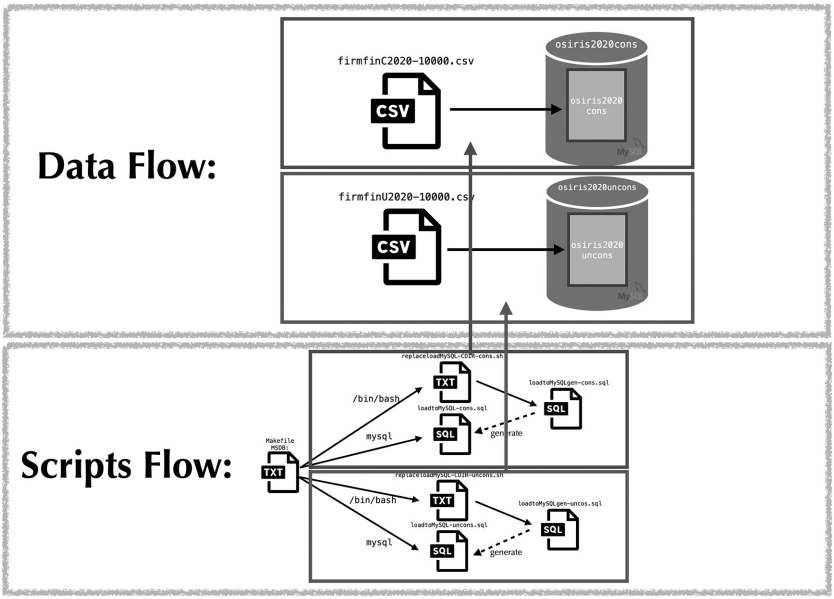

図11：Osiris データに関するデータベース構築のためのターゲット **MSDB** の実行にともなうシェルスクリプトとデータに関するファイルの流れ

図10，11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．

なお，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 5 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
Fri Mar 19 13:44:37 JST 2021
masa@aule DBMaking % cat end-MSDB.txt
Fri Mar 19 13:44:45 JST 2021
```

この結果から，8 秒であることがわかる[19]．

■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10：`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 5）と，Ubuntu 用のもの（スクリプト10）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
```

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

この処理時間は，スクリプト10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
Fri Mar 19 14:16:49 JST 2021
masa@balin: DBMaking$ cat end-MSDB.txt
Fri Mar 19 14:17:16 JST 2021
```

この結果から，27秒であることがわかる[20]．

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（Os-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11参照）．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11：Makfile: 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11の3，4行目が手順（Os-S3）の（1）を実現するためのものであり，5，6行目が手順（Os-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

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

化したものを図12に与える．

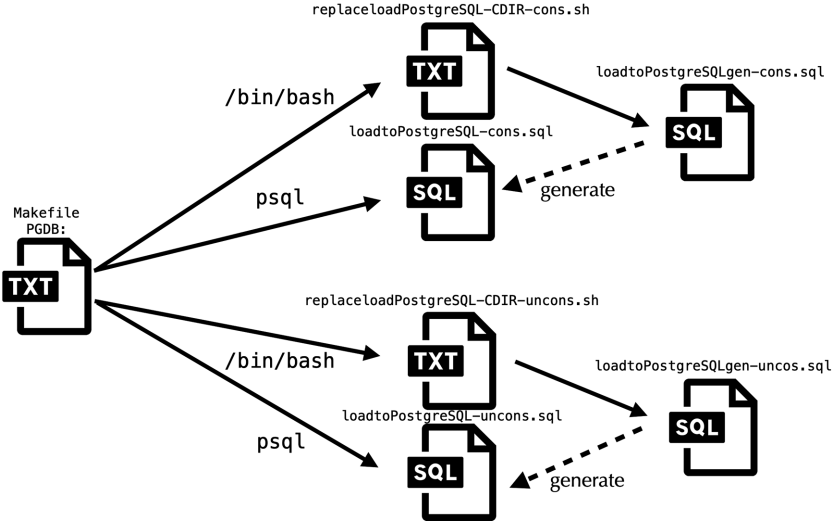

図12：ターゲット PGDB の実行にともなうスクリプトファイルの流れ

まず，手順（Os-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 loadtoPostgreSQLgen-cons.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 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 loadtoPostgreSQLgen-cons.sql から生成される（図12参照）．

スクリプト12：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13：loadtoPostgreSQLgen-cons.sql（一部抜粋）

```
1  DROP DATABASE IF EXISTS osiris2020cons;
2  CREATE DATABASE osiris2020cons;
3  \c osiris2020cons;
4  DROP TABLE IF EXISTS osiris2020cons;
5  CREATE TABLE osiris2020cons (
6  firm VARCHAR(350),
7  :
8  : (中略)
9  :
10 year VARCHAR(4)
11 );
12 COPY public.osiris2020cons  FROM 'CDIR/firmfinC2020-10000.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osiris2020consと，テーブルosiris2020consを作成（2～11行目）した後，データファイルfirmfinC2020-1000.csvから，テーブルosiris2020cons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11）の4行目で実行される．

次に，手順（Os-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11）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14）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15）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15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12参照）.

スクリプト14：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15：loadtoPostgreSQLgen-uncons.sql（一部抜粋）

```
1  DROP DATABASE IF EXISTS osiris2020uncons;
2  CREATE DATABASE osiris2020uncons;
3  \c osiris2020uncons;
4  DROP TABLE IF EXISTS osiris2020uncons;
5  CREATE TABLE osiris2020uncons (
6  firm VARCHAR(350),
7  :
8  : (中略)
9  :
10 year VARCHAR(4)
11 );
12 COPY public.osiris2020uncons  FROM 'CDIR/firmfinU2020-10000.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 osiris2020uncons と，テーブル osiris2020uncons を作成（2～11行目）した後，データファイル firmfinU2020-1000.csv から，テーブル osiris2020uncons 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11）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13に与える.

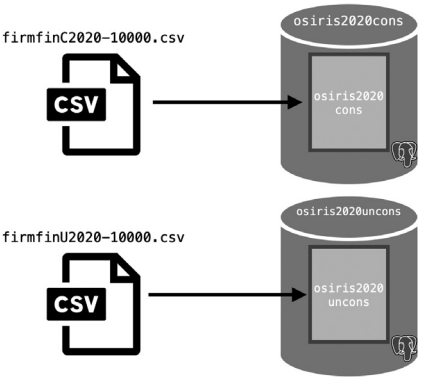

図13：PostgreSQL よる Osiris データにもとづくデータベース **osiris2020cons**，**osiris2020unccons** の構築

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14に与える．

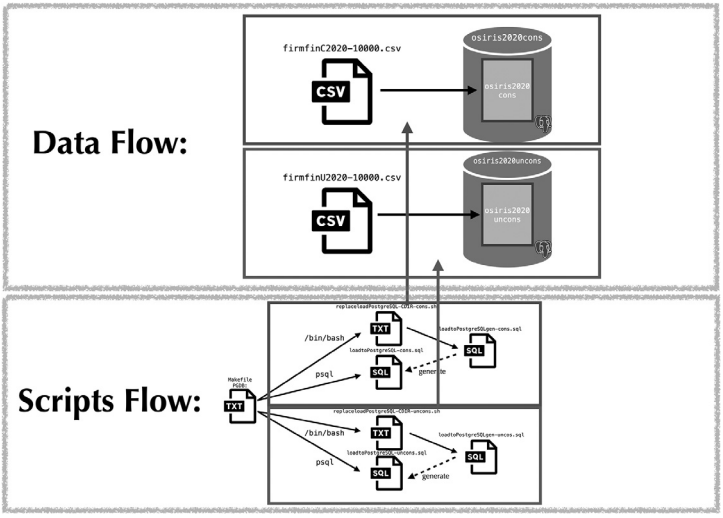

図14：Osiris データに関するデータベース構築のためのターゲット **PGDB** の実行にともなうシェルスクリプトとデータに関するファイルの流れ

図13，14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．

なお，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11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
Fri Mar 19 13:44:45 JST 2021
masa@aule DBMaking % cat end-PGDB.txt
Fri Mar 19 13:44:52 JST 2021
```

この結果から，7秒であることがわかる[21]．

■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16：Makefile: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11）と，Ubuntu用のもの（スクリプト16）を比較することによって，PostgreSQLとの対話型インター

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

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16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
Mon Mar 15 14:38:45 JST 2021
masa@balin: DBMaking$ cat end-PGDB.txt
Mon Mar 15 14:38:52 JST 2021
```

この結果から， 7 秒であることがわかる[22]．

V　Osiris データの抽出

ここでは，Osiris 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）Osiris のロゴ（アイコン Osiris）をクリックする．

（Os3）Osiris 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 Osiris2020 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．

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

以上の手順によって Osiris データ（連結ベース）の抽出ページへアクセスすることができる（図15参照）.

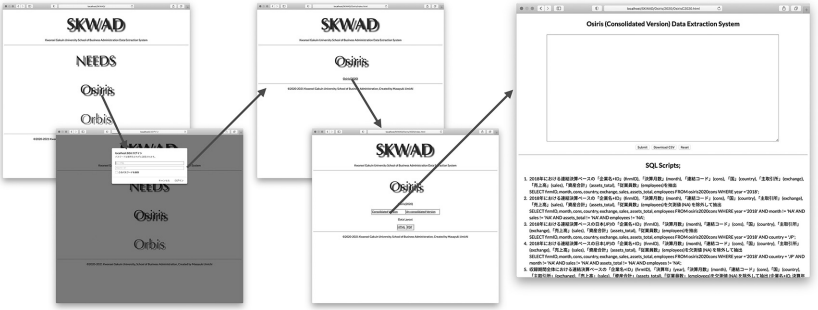

図15：Osiris データ（連結ベース）の抽出ページへのアクセス

この抽出ページの利用法は，地道（2021）で説明された NEEDS 企業財務データの抽出と同様であるが，以下に簡単に解説する（図16参照）.

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.

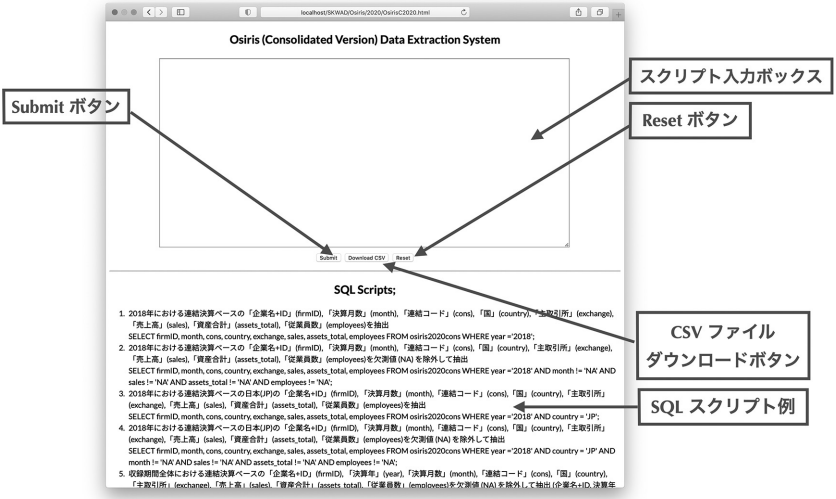

図16：Osiris データ（連結ベース）抽出システムのページ

なお，SQL のサンプルスクリプトも用意されている（図16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（osiris2020cons）から，2018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17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17：連結ベースのデータベース（osiris2020cons）から，2018年におけるデータの抽出

```
SELECT firmID, month, cons, country, exchange, sales, assets_total, employees
        FROM osiris2020cons
              WHERE year ='2018';
```

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[23]」（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osiris 2020consを指定する．

3行目：WHERE句で条件として2018年の情報（year = '2018'）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（図17も参照）：

（Os-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する．

（Os-E2）Submitボタンをクリックする．

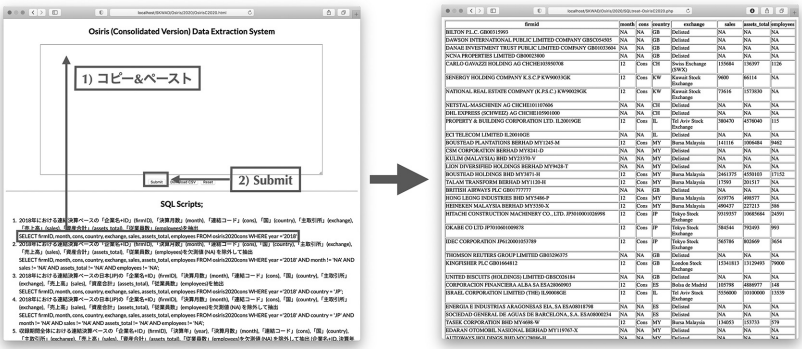

図17：Osirisデータ（連結ベース）の抽出

この手順を実行することによって，図17の右側の図にあるように抽出結果が表示される．なお，この結果として，10,000行のデータが得られる[24]．

23）BvD IDはBvD社が各企業に一意に与えたコードである．

24）Ⅳ節でも解説したが，今回構築したデータベースは，Osirisデータセットから1万社

この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可能であるが，行数が多いため，操作のミスを軽減したり，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18も参照）：

（Os-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．

（Os-C2）`Download CSV` ボタンをクリックする．

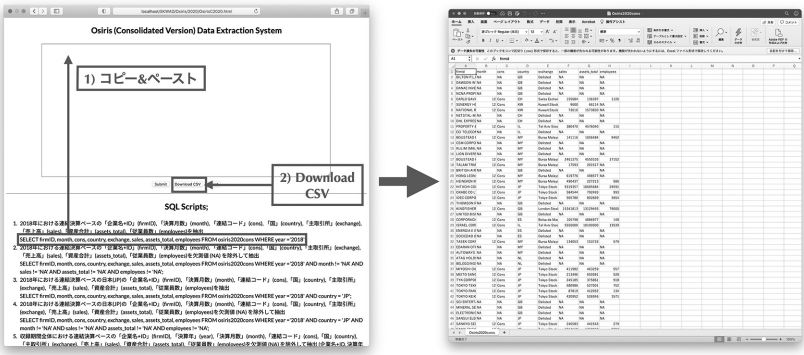

図18：Osirisデータ（連結ベース）の抽出結果をCSVファイル（**osiris2020cons.csv**）としてダウンロード

Ⅵ Osirisデータの可視化

ここでは，Ⅴ節で抽出したOsirisデータのCSVファイル `osiris2020cons.csv` をデータ解析環境R[25]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 `osiris2020cons.csv` をRに読み込むには，`read.csv` 関数を利用して以下のよ

をランダムサンプリングされたデータであることを思い出そう．

25）`https://www.r-project.org`

うに入力すればよい[26]．

Rへのデータの読み込み

```
> x <- read.csv("osiris2020cons.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                                BILTON P.L.C. GB00315993    NA <NA>      GB              Delisted     NA           NA        NA
2  DAWSON INTERNATIONAL PUBLIC LIMITED COMPANY GBSC054505    NA <NA>      GB              Delisted     NA           NA        NA
3 DANAE INVESTMENT TRUST PUBLIC LIMITED COMPANY GB01033604    NA <NA>      GB              Delisted     NA           NA        NA
4                      NCNA PROPERTIES LIMITED GB00023800    NA <NA>      GB              Delisted     NA           NA        NA
5                 CARLO GAVAZZI HOLDING AG CHCHE103950708    12 Cons      CH    Swiss Exchange(SWX) 155684       136397      1126
6               SENERGY HOLDING COMPANY K.S.C.P KW90033GK    12 Cons      KW  Kuwait Stock Exchange   9600        66114        NA
```

ここで，変数名（カラム名）は以下のようなものである[27]：

`firmid`：企業名＋ BvD ID

`month`：決算月数

`cons`：連結コード（`Cons`：連結）

`country`：国情報

`exchange`：主取引所

`sales`：売上高（単位：1,000 US ドル）

`assets_total`：資産合計（単位：1,000 US ドル）

`employees`：従業員数（単位：人）

このデータオブジェクト x（データフレーム）に対して，以下のようなス

26）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．

27）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．

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(cons == "Cons", month == 12) %>%
+   select(sales, assets_total, employees) %>% na.omit() %>%
+   ggpairs(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ヵ月だけのものに限定（cons == "Cons", month == 12）し，select 関数で列（sales, assets_total, employees）を選択している．さらに，関数 ggpairs によって対散布図を描いている．ここでは，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．

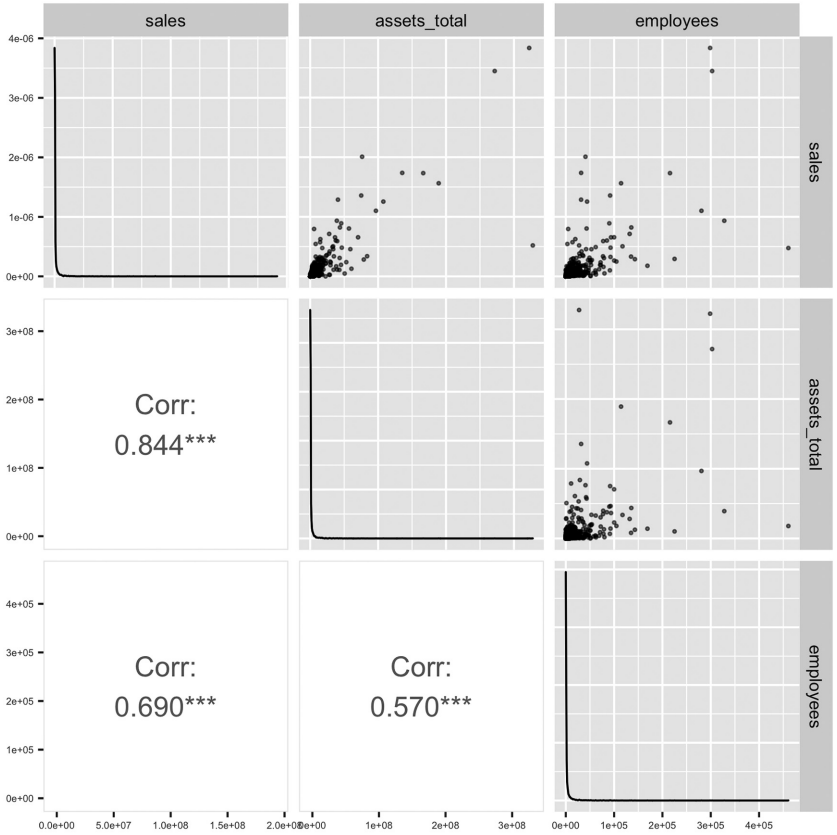

図19：対角ブロックには売上高（sales），資産合計（assets_total），従業員数（employees）に関する推定された密度関数が描かれている．また，上三角ブロックには，売上高（sales），資産合計（assets_total），従業員数（employees）の 2 つの組合せに関する散布図が描かれている．さらに，下三角ブロックには，売上高（sales），資産合計（assets_total），従業員数（employees）の 2 つの組合せに関する相関係数が与えられている．

対散布図（図19）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．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，R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[28]（図20も参照）．

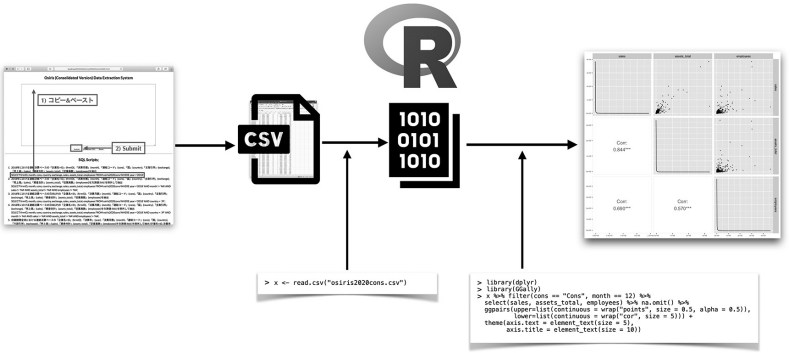

図20：CSV ファイルとしてダウンロードされた Osiris データファイルを R を利用して可視化する工程

Ⅶ おわりに

本稿では，世界の上場企業の財務情報を収録したデータベース Osiris 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 Osiris 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．

今後の課題として，BvD 社の世界の「全企業」を対象とするデータベース Orbis から抽出されたデータ[29] を利用した抽出システムの構築について検討する予定である．

（筆者は関西学院大学商学部教授）

28）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．

29）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

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）『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，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.

［10］ 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.

［11］ 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.

［12］ 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.

［13］ 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.

（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（FENNEL）と GPGPU 環境でデータベース管理システム PostgreSQL と **PG-Strom**（https://heterodb.github.io/pg-strom/ja/）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を検討している．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．

［14］ 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.
［15］ 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.
［16］ 西沢夢路（2017）『基礎からの MySQL 第 3 版』，SB クリエイティブ.
［17］ 鈴木啓修（2012）『PostgreSQL 全機能バイブル』，技術評論社.
［18］ Tange, Ole,（2018）*GNU Parallel 2018*, ISBN : 9781387509881, DOI : 10.5281/zenodo.1146014, URL: https://doi.org/10.5281/zenodo.1146014, Mar, 2018.
［19］ Wickham, H. and G. Grolemund（2016）*R for Data Science*, O'Reilly.

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 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，関西学院大学商学部 阪智香教授，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.

付録 A　R に関する環境

本稿を執筆した時点での R に関する情報を与える．

sessionInfo の実行結果

```
> sessionInfo()
R version 4.0.5 (2021-03-31)
Platform: x86_64-apple-darwin17.0 (64-bit)
Running under: macOS Big Sur 10.16

Matrix products: default
BLAS:   /Library/Frameworks/R.framework/Versions/4.0/Resources/lib/libRblas.dylib
LAPACK: /Library/Frameworks/R.framework/Versions/4.0/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_1.0.5

loaded via a namespace (and not attached):
 [1] Rcpp_1.0.6         magrittr_2.0.1     tidyselect_1.1.0   munsell_0.5.0
 [5] colorspace_2.0-0   R6_2.5.0           rlang_0.4.10       fansi_0.4.2
 [9] plyr_1.8.6         tools_4.0.5        grid_4.0.5         gtable_0.3.0
[13] utf8_1.2.1         DBI_1.1.1          withr_2.4.1        ellipsis_0.3.1
[17] digest_0.6.27      assertthat_0.2.1   tibble_3.1.0       lifecycle_1.0.0
[21] crayon_1.4.1       farver_2.1.0       RColorBrewer_1.1-2 purrr_0.3.4
[25] vctrs_0.3.6        glue_1.4.2         labeling_0.4.2     compiler_4.0.5
[29] pillar_1.5.1       generics_0.1.0     scales_1.1.1       reshape_0.8.8
[33] pkgconfig_2.0.3
```

付録 B　Osiris データ

本稿で構築したデータベース osiris2020cons，osiris2020uncons におけるテーブル osiris2020cons，osiris2020uncons のカラム名の一覧を以下に与える[30]：

30）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 6 ：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	ID	BvD ID（企業コード，一意）	BvD ID number
4	country	国	Address of incorp. -Country
5	SIC_code	SIC 業種コード	US SIC, Primary code(s)（M）
6	SIC_name	SIC 業種名	US SIC, primary code(s) description
7	exchange	主取引所	Main exchange
8	cons	連結・単体	Consolidation code
9	date	決算日	Closing date
10	month	月数	Number of months
11	audit	監査	Audit Status
12	practice	会計基準	Accounting standard
13	source	データの出所	Source
14	units	単位（金額）	Statement unit
15	currency	現地通貨	Currency of the statement
16	exchange_rate	換算レート	Exchange Rate from Local Currency
17	assets_fix	固定資産	Fixed Assets
18	assets_int	無形固定資産	Intangible Fixed Assets
19	assets_tang	有形固定資産	Tangible Fixed Assets
20	assets_other_fix	その他の固定資産	Other Fixed Assets
21	assets_cur	流動資産	Current Assets
22	stock	株式	Stock
23	debtors	売掛金	Debtors
24	assets_other_cur	その他の流動資産	Others
25	cash	現金及び現金同等物	Cash & Cash Equivalent
26	assets_total	資産合計	Total Assets
27	shareholders	株主資本	Shareholders Funds
28	capital	資本	Capital
29	shareholders_other	その他の株主資本	Other
30	liabilities_non_cur	非流動負債	Non Current Liabilities
31	debt_long	固定負債	Long Term Debt
32	liabilities_other_non_cur	その他の非流動負債	Other Non Current Liabilities
33	provisions	引当金	Provisions
34	liabilities_cur	流動負債	Current Liabilities
35	loans	借入金	Loans
36	creditors	買掛金	Creditors
37	liabilities_other_cur	その他の流動負債	Other
38	total_s_l	負債純資産合計	Total Shareh. Funds & Liab.
39	capital_working	運転資本	Working Capital
40	assets_net_cur	正味流動資産	Net Current Assets
41	enterprise_value	企業価値	Enterprise Value
42	employees	従業員数	Number of Employees
43	operating_revenue	営業収益	Operating Revenue / Turnover
44	sales	売上高	Sales
45	costs_goods	売上原価	Costs of Goods Sold
46	profit_gross	売上総利益	Gross Profit
47	expenses_other	その他の営業費用	Other Operating Items
48	EBIT	営業利益	Operating P/L
49	revenue_fin	金融収益	Financial Revenue
50	PL_fin	金融収支	Financial P/L

51	PL_other	臨時損益	Other non Oper./Financial Items
52	PL_before_tax	税引前利益	P/L before Tax
53	tax	税金	Taxation
54	PL_after_tax	税引後利益	P/L after Tax
55	PL_extr	特別収支	Extraord. & Oth. Items
56	net_income	純利益	P/L for Period
57	costs_material	原材料費	Material Costs
58	costs_employees	人件費	Costs of Employees
59	depr_amor	有形及び無形資産償却	Depreciation/Amortization
60	interest_paid	支払利息	Financial Expenses
61	R_D	研究開発費	Research & Development expenses（memo）
62	cash_flow	キャッシュフロー	Cash Flow
63	add_value	付加価値	Added Value
64	EBITDA	EBITDA	EBITDA
65	tax_income	法人税	Income taxes
66	tax_pay_income	未払法人税	Income Tax Payable
67	tax_deferred	繰延税金	Deferred Taxes
68	tax_credit	法人税等調整額及び投資税額控除	Def. Inc. Taxes & Invest. Tax Credit
69	date_cur_market_cap	株式時価総額の日付	Date of current Market capitalisation
70	market_cap_cur	時価総額（現在）	Current market capitalisation
71	year_mp	年（市場価格）	Market price -Year
72	market_price1	市場価格 1 月末	Market price -January
73	market_price2	市場価格 2 月末	Market price -February
74	market_price3	市場価格 3 月末	Market price -March
75	market_price4	市場価格 4 月末	Market price -April
76	market_price5	市場価格 5 月末	Market price -May
77	market_price6	市場価格 6 月末	Market price -June
78	market_price7	市場価格 7 月末	Market price -July
79	market_price8	市場価格 8 月末	Market price -August
80	market_price9	市場価格 9 月末	Market price -September
81	market_price10	市場価格10月末	Market price -October
82	market_price11	市場価格11月末	Market price -November
83	market_price12	市場価格12月末	Market price -December
84	market_cap	年度末の時価総額	Market Cap.
85	dividend	配当	Dividend Paid
86	date_IPO	上場年月日	IPO Date
87	date_delisted	上場廃止年月日	Delisted Date
88	date_incor	創業年	Date of Incorporation
89	ISIN	株式コード（全世界共通）	ISIN
90	ticker	株式コード（国内）	Ticker
91	GICS_code	GICS 業種コード	GICS code
92	GICS_name	GICS 業種名	GICS description
93	firmID	企業名＋BvD ID	NA
94	year	年	NA