財務データ抽出システム

SKWAD

関西学院大学 商学部
地道 正行

2022 年 5 月 20 日

まえがき

2010 年を境に「データサイエンス」や「ビッグデータ」といった用語が日本においてもマスメディアで連日取り上げられるようになり, 10 年以上の歳月を経て, 「はやり言葉」(バズワード) から「日常用語」として定着したことに異存はないであろう. また, 近年では, ビジネスに関する意思決定においても「データ」によるエビデンスが要求され, 「ビジネス」,「データ」, 「サイエンス」を冠した専門書も出版されている (cf. Taddy (2019)).

これらの用語が出現した期を同じくして, 本学でも学内向け財務データ抽出システム (cf. 地道 (2010-a, b, 2012)) が構築・運用されて同様の年月が経過した. 開発当初のシステムは, 日経メディアマーケティング社 [1] から販売されている NEEDS 企業財務データ (一般事業会社, 金融関連会社) をデータベース化することによって設計・実装されていたが, 学内サービスの運用開始から 10 余年が経過し, 筆者が研究や講義・演習に利用する中で, 以下のような課題があることがわかった.

- システム名称の再考
- リレーショナル・データベース管理システムの見直し
- NEEDS 企業財務データの仕様変更への対応
- 抽出システムのインターフェースデザイン改良
- 財務データの拡充
- ダウンロード法の拡充

本稿 [2] は, これらの課題に対する対応策を検証し, 学内向けシステムとして再構築するための一連の研究 (地道 (2021-a, b, c, d), 地道, 阪 (2020-a, b, 2021-b, c), 地道ら (2021-a, b)) にもとづいて再構成したものである.

本稿は, 以下のような 3 部から構成されている. まず, 第 I 部では, システム構築の詳細を述べる. ここでは, NEEDS 企業財務データ (第 1 章), Osiris データ (第 2 章), Orbis データ (第 3 章) のデータベースと, データ抽出システム SKWAD (第 4 章) の設計と実装について扱う. 次に, 第 II 部では, 第 5 章においてデータ抽出システム SKWAD を利用した NEEDS 企業財務データ, Osiris データ, Orbis データ それぞれの抽出と可視化について実例と共に述べる. また, 第 III 部では, 資料編として, リレーショナル・データベースに関する基礎知識 (第 6 章) を与えるとともに, システムの開発環境 (第 7 章), 2020 年版 NEEDS 企業財務データ の仕様 (第 8 章), Osiris データ の仕様 (第 9 章), Orbis データ の仕様 (第 10 章) について述べる.

地道 (2010-b) の「まえがき」における以下の指摘を今一度引用しよう:

> 一般の科学の分野にとどまらず, 社会科学系の学問分野においても, 「安価」で「効率的」かつ「短時間」に「大量」のデータを入手する方法を検討することは, 現在でも重要なテーマであろう.

---

[1] `https://www.nikkeimm.co.jp/`

[2] 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.

この主張は, 現在でも, 再考するべき重要な課題といえる.

開発から 10 余年を経て, このシステムを利用した研究成果もいくつかでている (cf. 地道 (2014, 2022), Jimichi and Maeda (2014), 柳ら (2018-a, b, 2019), Shih *et al.* (2020), 田中ら (2021, 2022)) [3)]. 新システム SKWAD で提供されるサービスが本学における研究活動の活性化につながることが期待される.

2022 年 5 月

筆者

[3)] 筆者が担当している講義科目 (商学部開講科目「ビジネスデータ分析 I, II」) では, このシステムを利用し, 構造化参照言語を用いた財務データの抽出から, 前処理・可視化・統計モデリングを実行することを実施している. 柳ら (2018-a, b, 2019), 田中ら (2021, 2022) の研究成果は, この講義の課題レポートに端を発するものである.

謝辞

本研究の一部は以下の助成を得ている.

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〜2022 年), 課題番号: 19K02006

2017 年度〜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: jh171002-NWJ, jh181001-NWJ, jh191002-NWJ, jh201003-NWJ, 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, 東京大学 宮本大輔准教授, 日経メディアマーケティングの佐藤健吾氏, ビューロー・ヴァン・ダイクの増田 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, 関西学院大学図書館 藤澤 快氏, 中野容尚氏には様々なご足労を賜った. ここに感謝の意を表する.

目次

第 I 部

システム構築編

第 1 章

NEEDS 企業財務データ

本章では,「NEEDS 企業財務データ」にもとづくリレーショナル・データベース (以下単にデータベースと略す) の構築の詳細を与える [1]. なお, データベース関連の用語の解説を第 6 章に, NEEDS 企業財務データの仕様については第 8 章に与えているので参照されたい.

1.1 NEEDS 企業財務データの新仕様への対応

地道 (2010-b) で議論されている学内向けの財務データ抽出システムが構築された直後の 2010 年代初頭に NEEDS 企業財務データの仕様変更があり, その構造がそれまでのものと異なったものとなった. それ以前は, 連結決算, 単独決算でデータファイルが分かれており, 会計基準も一種類 (日本基準) であったが, 変更後は連結決算と単独決算が合併され, 会計基準が 3 種類 (日本基準, 米国会計基準, 国際会計基準) となり, 本決算に加えて, 四半期決算のデータも収録されるようになった. なお, これらの変更にともなって, データの一意性を保証するための主キー [2] が, 2 種類 (日経会社コード, 決算年月日) から, 5 種類 (日経会社コード, 連結基準フラグ, 決算年月, 決算種別フラグ, レコード種別) に増加した. データベースの構築には, これらの変更に対応する必要があるとともに, データ量の増加に伴って, 抽出時間を短縮することを考慮しつつ, データベースからデータを抽出する方法を再検討する必要があった.

これらの問題に対して, 以下のような方法でデータベースを構築した:

(DB1) データの前処理のためのスクリプトを新しいデータファイルの形式へ対応させるために修正 [3]
(DB2) MySQL [4] と PostgreSQL [5] の両方でデータベースを構築
(DB3) 全データを使って構築したデータベースから, 連結本決算と単独本決算のデータベースを分離・構築

対処法 (DB2) における MySQL と PostgreSQL は, オープンソース [6] の代表的なリレーショナル・データ

---

[1] 2022 年 4 月 4 日に東京証券取引所の市場構造の見直しが行われ, それまでの市場区分 (市場第一部, 市場第二部, マザーズ, JASDAQ(スタンダード, グロース)) から, 新市場区分 (プライム市場, スタンダード市場, グロース市場) に変更された. 今回構築した NEEDS 企業財務データを利用したデータベースは, 2020 年 6 月期決算までの財務情報が収録されているので, 旧区分であることを強調しておく. なお, これらの市場構造の変更への弊システムの対応は, 今後の課題といえる.

[2] 主キー (primary key) とは, リレーショナル・データベースのテーブル内でレコードを一意に識別することができるように指定される列のことである (IT 用語辞典 e-Words `https://e-words.jp/` 参照).

[3] 前処理には, GNU parallel(`https://www.gnu.org/software/parallel/`) を利用して並列処理により高速化することも試みている. GNU parallel については, Tange (2018) を参照されたい.

[4] `https://www.mysql.com/`

[5] `https://www.postgresql.org/`

[6] オープンソース (Open Source) とは, 人間が理解しやすいプログラミング言語で書かれたコンピュータプログラムであるソースコードを広く一般に公開し, 誰でも自由に扱ってよいとする考え方. また, そのような考えに基づいて公開されたソフトウェアのこと. オープン・ソース・ソフトウェア (Open Source Software: OSS) とも呼ばれる (IT 用語辞典 e-Words

ベース管理システム (Relational DataBase Management System; RDBMS) である．なお，リレーショナル・データベース管理システムについては，第 6 章を参照されたい．

データファイルとしては 2020 年に納品されたものを利用して，作業工程毎のスクリプトを作成したものを `Makefile` にまとめ，作業工程を `make` コマンド (cf. Mecklenburg (2005)) によって自動実行する仕様とした．このことは，地道 (2012) でも指摘されているが，データベースを迅速かつ正確に自動的に構築したり，サービスの拡大を行うために必要となったためである．

図 1.1 には，NEEDS 企業財務データ (一般事業会社) 2020 年版にもとづくデータベース構築のイメージを与えている．なお，MAMP, MAPP, LAMP, LAPP 環境とその構築の詳細については，第 7 章を参照されたい．

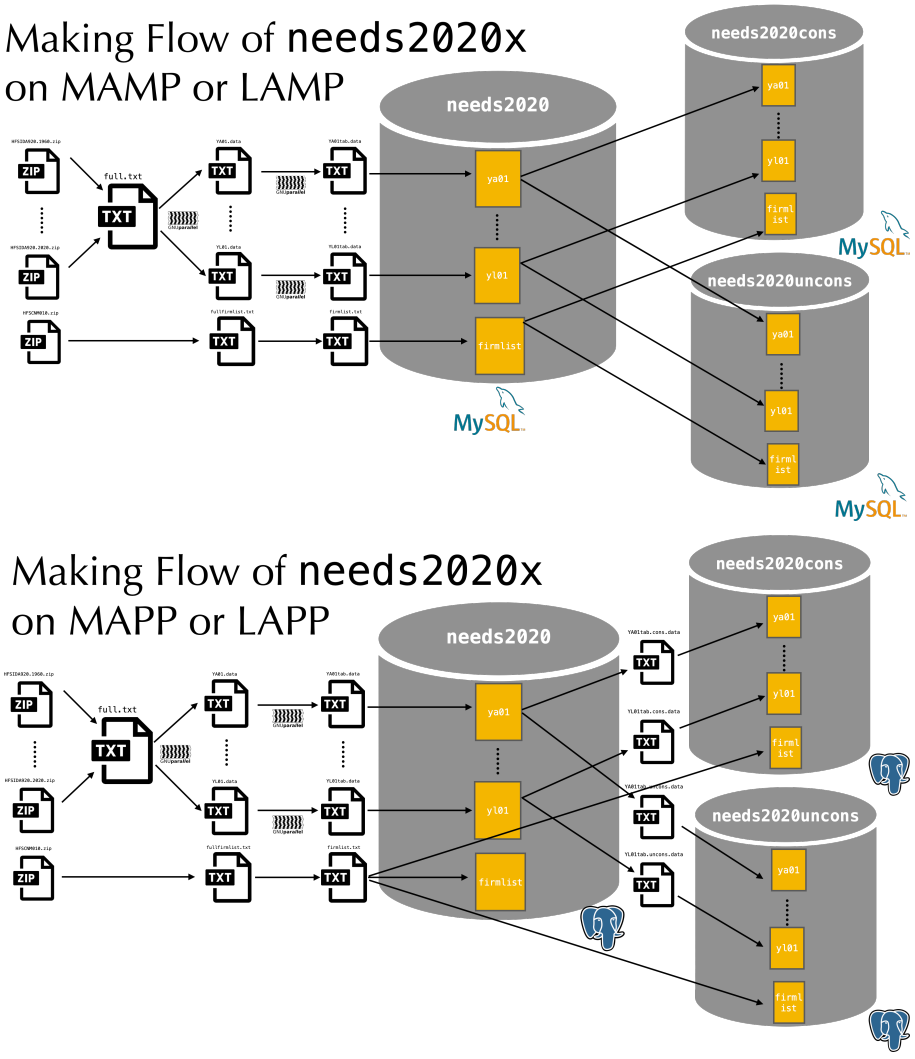


図 1.1: MAMP, MAPP, LAMP, LAPP 環境のもとで NEEDS 企業財務データ (一般事業会社) 2020 年版にもとづくデータベース構築のイメージ

これらのデータベースの構築に関する詳細を以下に述べる [7]．

`https://e-words.jp/` 参照)．

[7] 今回のシステム再構築に関しては，NEEDS 企業財務データ (一般事業会社) 2018, 2019 年版にもとづくデータベースも作成した

1.2 データベースの構築

1.1 節でも述べたが, この製品 (NEEDS 企業財務データ (一般事業会社) 2020 年版) に関しては [8], 2010 年代初めに以下のような仕様変更があった.

NEEDS 企業財務データの仕様変更

- 変更前は, 単独決算と連結決算のファイル群を分けて販売されており, 会計基準による分類もなかった.
- 仕様変更に伴って, 単独決算と連結決算のデータが合併され, 会計基準として「日本基準」,「米国会計基準」,「国際会計基準」のデータも同一のファイルに合併されて配布されるようになった.
- 本決算と中間決算以外に四半期決算の情報も収録されるようになった.
- データを一意化するためのキー (主キー) は, 従来のものが (日経会社コード, 決算年月日) の 2 個であったのに対して, 表 1.1 の 5 個に増加した.

表 1.1: NEEDS 企業財務データに関する主キー

キー順位	カラム名	形式	項目	収録内容
第１キー	a05	C7	日経会社コード	日経が定める会社コード
第２キー	a27	C1	連結基準フラグ	1: 日本基準，2: 米国会計基準， 3: 国際会計基準，0: 単独
第３キー	a02	C6	決算年月	YYYYMM 形式
第４キー	a10	C2	決算種別フラグ	10: 本決算，21: 第１四半期， 22: 第２四半期（中間決算），23: 第３四半期，24: 第４四半期，25: 第５四半期
第５キー	a01	C4	レコード種別	決算短信情報レコード: TA01〜TL01，有価証券報告書情報レコード：YA01〜YL01

これらの仕様変更から, データベースとデータ抽出環境を構築することに関して以下のような問題が生じた.

仕様変更に伴うデータベース及びデータ抽出環境構築に関する問題

(CP1) SQL [a] 問合せによるデータ抽出時間の問題
(CP2) 一意性の確保の問題

[a] 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. 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. 簡単な説明を 6.3 節に与えているので参照されたい. また, 詳細については, 例えば, 増永 (2017) を参照されたい.

問題 (CP1) については, 以前と同様の方法で実験的にデータベース環境を構築し, これまで利用していた適当な SQL 問合せ (ある時点での売上高や資産合計など 2, 3 の指標に対するデータを抽出するためのスクリプト) が, 方法は 2020 年版の場合と本質的にかわらない.

[8] 2010 年までは,「NEEDS 企業財務データ (一般事業会社), MT 版」という名称で販売されていた. ここで, MT とは配布媒体が Magnetic Tape であることを指していたが, 2008 年には DVD などのメディアに変更されていた.

による抽出を試みたところ, データの規模が大きくなったことから, 数十分 (場合によっては数時間) を要する場合があった. さらに, 問題 (CP2) に起因する問題から, 主キーをいくつも与えないと企業の複数の決算にもとづく結果が抽出されてしまい, 一意に抽出できない. これらの問題に対して, 以下のような解決策を講じた.

解決策

(S1) 主キーによるデータベースの分離
(S2) SQL 問合せの工夫

解決策 (S1) について説明する. なお, 主キーの説明については, 表 1.1 を参照されたい. まず, 今回利用してるデータファイルは, 第 5 キーの「レコード種別」における「有価証券報告書情報レコード」のみであり, 「決算短信情報レコード」のものは利用しない. このことから, 第 5 キーは検討の対象外となる. 次に, これまでに, 構築してきた財務データベースは,「本決算」かつ,「連結」または「単独」のものであるので, 第 4 キーである「決算種別フラグ」として「本決算」(10) を選択し, かつ, 第 2 キーである「連結基準フラグ」を「単独」(0) とそれ以外 (「日本基準」1, 「米国会計基準」2, 「国際会計基準」3) に分けて選択することによって, 主キーを第 1 キーである「日経会社コード」と第 3 キーである「決算年月」に絞り込むことができる. つまり, 「連結本決算」と「単独本決算」のレコードを抽出し, 別のデータベースへ分離することによって, サイズを縮小することが可能となる. なお, この方法は, ユーザに多くの主キーを指定させることなく一意性が確保できるという「副次的効果」(side effect) も与える. 以上のことから, 解決策 (S1) はサーバサイドの対応といえる.

次に, 解決策 (S2) について説明する. これまで運用してきた財務データベースは, 各テーブルのサイズが (今回のものよりも) 小さかったことから, JOIN を実行してテーブルをフルに結合しても, それにかかる時間は, それほど気になるものではなかったが, 今回実験的に構築したデータベース環境で試行してみたところ, 「現実的な時間」で結合することが難しいことがわかった. そこで, 結合する際に, テーブルから事前にカラムを指定したものをテーブル化して, 結合時のテーブルサイズを縮小するというアイデアを利用したところ, 現実的な時間で抽出することが可能であることがわかった. このアイデアは, データベースの分野ではよく知られた単純な方法であり, クライアント (ユーザ) サイドの解決策といえる.

サーバサイドの解決策 (P1) を考慮し, 以下のような手順でデータベースの構築を行った:

NEEDS 企業財務データ (一般事業会社) 2020 年版にもとづくデータベースの構築手順

(NS1) データファイル HFSIDA920.1960.zip〜HFSIDA920.2020.zip と収録会社情報のファイル HFSCNM010.zip の前処理
(NS2) データベース needs2020 を作成し, テーブル ya01〜yl01 (レコード種別 YA01〜YL01 に対応) を作成後, 前処理されたデータファイル YA01tab.data〜YL01tab.data からテーブル ya01〜yl01 へロード
(NS3) データベース needs2020 にテーブル firmlist を作成し, ファイル firmlist.txt からロード
(NS4) データベース needs2020 から, 連結本決算のデータベース needs2020cons と単独本決算のデータベース needs2020uncons を分離・作成

上記の手順は, OS として, macOS と Ubuntu(Linux), さらに, RDBMS として, MySQL と PostgreSQL の合計 4 種類の環境 (MAMP, MAPP, LAMP, LAPP) について実施する必要がある (表 7.1 と図 7.1 参照).

以下に, それぞれの環境のもとでの手順を確認していく. なお, macOS 環境と Ubuntu 環境で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, それぞれ, 図 1.2 と 図 1.3 を参照さ

れたい [9].

```
MT-general2020-macOS
├──── DBMaking
│       ├──── Makefile
│       ├──── dumpfromPostgreSQLgen-cons.sql
│       ├──── dumpfromPostgreSQLgen-uncons.sql
│       ├──── loadfirmlisttoMySQLgen.sql
│       ├──── loadfirmlisttoPostgreSQLgen-cons.sql
│       ├──── loadfirmlisttoPostgreSQLgen-uncons.sql
│       ├──── loadfirmlisttoPostgreSQLgen.sql
│       ├──── loadtoMySQLgen.sql
│       ├──── loadtoPostgreSQLgen-cons.sql
│       ├──── loadtoPostgreSQLgen-uncons.sql
│       ├──── loadtoPostgreSQLgen.sql
│       ├──── mkcons.sql
│       ├──── mkuncons.sql
│       ├──── replacedumpPostgreSQL-CDIR-cons.sh
│       ├──── replacedumpPostgreSQL-CDIR-uncons.sh
│       ├──── replaceloadCDIR-cons.sh
│       ├──── replaceloadCDIR-uncons.sh
│       ├──── replaceloadCDIR.sh
│       ├──── replaceloadPostgreSQL-CDIR-cons.sh
│       ├──── replaceloadPostgreSQL-CDIR-uncons.sh
│       ├──── replaceloadPostgreSQL-CDIR.sh
│       ├──── replaceloadfirmlist.sh
│       ├──── replaceloadfirmlistPostgreSQL-cons.sh
│       ├──── replaceloadfirmlistPostgreSQL-uncons.sh
│       ├──── replaceloadfirmlistPostgreSQL.sh
│       ├──── script-p.sh
│       ├──── scriptfirmlist.sh
│       ├──── separate.pl
│       ├──── separatefirmlist.pl
│       ├──── seqscript-p.sh
│       └──── unzip.sh
└──── datafiles
        ├──── HFSCNM010.zip
        ├──── HFSIDA920.1960.zip
        ├──── HFSIDA920.1970.zip
        ├──── HFSIDA920.1980.zip
        ├──── HFSIDA920.1990.zip
        ├──── HFSIDA920.2000.1.zip
        ├──── HFSIDA920.2000.2.zip
        ├──── HFSIDA920.2010.1.zip
        ├──── HFSIDA920.2010.2.zip
        └──── HFSIDA920.2020.zip
```

図 1.2: macOS 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

[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ことに注意が必要である.

```
MT-general2020-Ubuntu
├── DBMaking
│   ├── Makefile
│   ├── dumpfromPostgreSQLgen-cons.sql
│   ├── dumpfromPostgreSQLgen-uncons.sql
│   ├── loadfirmlisttoMySQLgen.sql
│   ├── loadfirmlisttoPostgreSQLgen-cons.sql
│   ├── loadfirmlisttoPostgreSQLgen-uncons.sql
│   ├── loadfirmlisttoPostgreSQLgen.sql
│   ├── loadtoMySQLgen.sql
│   ├── loadtoPostgreSQLgen-cons.sql
│   ├── loadtoPostgreSQLgen-uncons.sql
│   ├── loadtoPostgreSQLgen.sql
│   ├── mkcons.sql
│   ├── mkuncons.sql
│   ├── replacedumpPostgreSQL-CDIR-cons.sh
│   ├── replacedumpPostgreSQL-CDIR-uncons.sh
│   ├── replaceloadCDIR-cons.sh
│   ├── replaceloadCDIR-uncons.sh
│   ├── replaceloadCDIR.sh
│   ├── replaceloadPostgreSQL-CDIR-cons.sh
│   ├── replaceloadPostgreSQL-CDIR-uncons.sh
│   ├── replaceloadPostgreSQL-CDIR.sh
│   ├── replaceloadfirmlist.sh
│   ├── replaceloadfirmlistPostgreSQL-cons.sh
│   ├── replaceloadfirmlistPostgreSQL-uncons.sh
│   ├── replaceloadfirmlistPostgreSQL.sh
│   ├── script-p.sh
│   ├── scriptfirmlist.sh
│   ├── separate.pl
│   ├── separatefirmlist.pl
│   ├── seqscript-p.sh
│   └── unzip.sh
└── datafiles
    ├── HFSCNM010.zip
    ├── HFSIDA920.1960.zip
    ├── HFSIDA920.1970.zip
    ├── HFSIDA920.1980.zip
    ├── HFSIDA920.1990.zip
    ├── HFSIDA920.2000.1.zip
    ├── HFSIDA920.2000.2.zip
    ├── HFSIDA920.2010.1.zip
    ├── HFSIDA920.2010.2.zip
    └── HFSIDA920.2020.zip
```

図 1.3: 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

1.2.1 前処理

ファイルの前処理 (NS1) は, 4 種類の環境 (MAMP, MAPP, LAMP, LAPP) で共通のスクリプトで処理が可能である. 具体的には以下の手順で前処理が行われる:

前処理の手順

(PP1) データの ZIP ファイル HFSIDA920.1960.zip〜HFSIDA920.2020.zip を展開し, 一つのファイル full.txt へ結合

(PP2) ファイル full.txt をレコード種別ごとのファイル YA01.data〜YL01.data へ分離し, 固定長フォーマットをデータの仕様に応じて, 適切にレコード間に分割記号 (タブ区切り) を挿入後, 文字コードを CP932 から UTF-8 へ変換することによって, ファイル YA01.data〜YL01.data をファイル YA01tab.data〜YL01tab.data へ変換

(PP3) 会社リストの ZIP ファイル HFSCNM010.zip をファイル fullfirmlist.txt へ展開し, 定長フォーマットをデータの仕様に応じて, 適切にレコード間に分割記号 (タブ区切り) を挿入後, 文字コードを CP932 から UTF-8 へ変換することによって, ファイル fullfirmlist.txt をファイル firmlist.txt へ変換

これらの手順を実行するスクリプトを Makefile のターゲット preprocess に記述した (スクリプト 1.1 参照). スクリプト 1.1 において, 2, 6 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 1.1: Makfile: ターゲット preprocess

```
1 preprocess:
2       date > start-preprocess.txt
3       /bin/bash unzip.sh
4       /bin/bash seqscript-p.sh
5       /bin/bash scriptfirmlist.sh
6       date > end-preprocess.txt
```

スクリプト 1.1 の 3, 4, 5 行目の各シェルスクリプトが前処理の手順 (PP1), (PP2), (PP3) を実現するためのものである. これらのシェルスクリプトに関するファイルの流れを図 1.4 に与える.

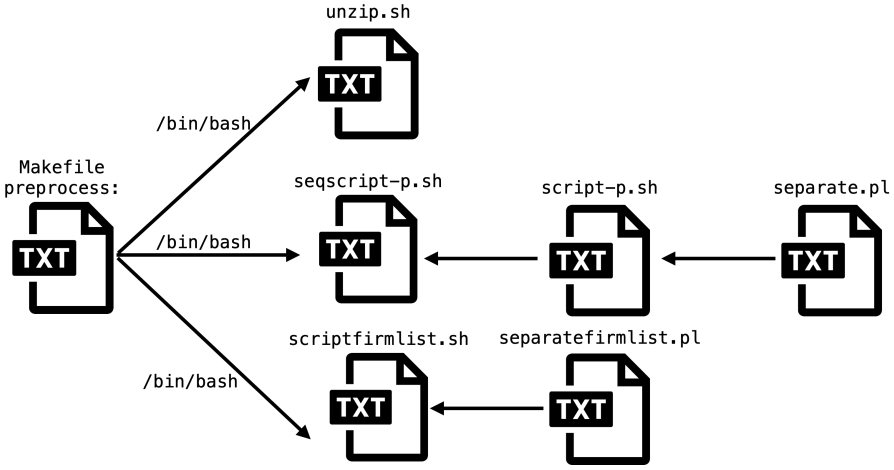


図 1.4: 前処理のシェルスクリプトに関するファイルの流れ

まず, 手順 (PP1) を実現するための unzip.sh はスクリプト 1.2 で与えられる.

スクリプト 1.2: unzip.sh

```
#!/bin/bash
echo Start unzip!
7z e -so ../datafiles/HFSIDA920.1960.zip >  full.txt
7z e -so ../datafiles/HFSIDA920.1970.zip >> full.txt
7z e -so ../datafiles/HFSIDA920.1980.zip >> full.txt
7z e -so ../datafiles/HFSIDA920.1990.zip >> full.txt
7z e -so ../datafiles/HFSIDA920.2000.1.zip >> full.txt
7z e -so ../datafiles/HFSIDA920.2000.2.zip >> full.txt
7z e -so ../datafiles/HFSIDA920.2010.1.zip >> full.txt
7z e -so ../datafiles/HFSIDA920.2010.2.zip >> full.txt
7z e -so ../datafiles/HFSIDA920.2020.zip >> full.txt
echo Complete unzip!
```

このシェルスクリプトでは, 7z コマンド [10] を用いて ZIP [11] ファイルを展開した後, リダイレクション (>, >>) 機能を使って, 一つのテキストファイル full.txt に結合している.

次に, 手順 (PP2) を実現するための seqscript-p.sh は, スクリプト 1.3 で与えられる.

スクリプト 1.3: seqscript-p.sh

```
#!/bin/bash
for i in {A..L}
do
echo "Y"$i"01"
/bin/bash script-p.sh "Y"$i"01"
done
```

このシェルスクリプトでは, 2 行目で変数 i を A から L まで変化させ, 3 行目から 6 行目で, 順次文字列 "Y"$i"01" を生成し, シェルスクリプト script-p.sh を繰り返し適用している. 例えば, i に A が代入されている場合は, (文字列) YA01 が生成され, スクリプト script-p.sh の引数として与えられる. 5 行目で使用されている処理手順を実行するスクリプト script-p.sh は, スクリプト 1.4 で与えられる.

スクリプト 1.4: script-p.sh

```
#!/bin/bash
export LC_ALL=C
#
parallel --pipepart -k --block 100M -a "full.txt" 'grep' $1 > $1".data"
#
echo Complete Separate!
#
perl separate.pl $1".data"
parallel --pipepart -k --block 100M -a "temp" "sed␣'/^$/d'" > $1".txt"
parallel --pipepart -k --block 100M -a $1".txt" "nkf␣--utf8" > $1"tab.data"
rm temp
#
echo Complete Tables!
echo Finish!
```

[10] https://www.7-zip.org/

[11] ZIP とは, 複数のファイルやフォルダ (ディレクトリ) を一つのファイルにまとめて格納するアーカイブファイルの標準的な形式の一つである. 1989 年に米 PKWARE 社の Phil Katz 氏によって考案された (IT 用語辞典 e-Words https://e-words.jp/ 参照).

このシェルスクリプトでは, 4 行目で全データを結合したテキストファイル full.txt から, 「親」スクリプトから受け取った引数 $1 (例えば, YA01) を含む行を grep コマンドを使って抽出し, ファイル (例えば, YA01.data) に書き出している. その際, この処理を GNU parallel で並列化することによって処理時間を短縮している. 次に, 8 行目で perl コマンドを利用して, 分離された個々のファイルの適切な箇所に分割記号を挿入している. その際, 引数として分離された個々のファイル名を与え, Perl [12] スクリプトファイル separate.pl (スクリプト 1.5 参照) を実行している.

スクリプト 1.5: separate.pl (一部抜粋)

```
1  #!/usr/bin/perl
2  #####
3  open(IN,"@ARGV");
4  open(OUT,">temp");
5  #####
6  do{
7  read(IN,$xx,3042);
8  $a[1]=substr($xx,0,4);
9  $a[2]=substr($xx,4,6);
10 : (中略)
11 $a[46]=substr($xx,97,3);
12 $b[1]=substr($xx,100,14);
13 $b[2]=substr($xx,114,14);
14 : (中略)
15 $b[210]=substr($xx,3026,14);
16 print OUT
17 "$a[1]\t$a[2]\t...\t$a[46]\t$b[1]\t$b[2]\t...\t$b[210]\n";
18 } while (eof(IN)!=1);
19 close (IN);
```

このスクリプトの, 3 行目では, 引数 ("@ARGV") として与えられたファイルを入力 (IN) とし, 1 行づつ読み取りながら, 適切に文字列を切り出したもの (8 行目から 15 行目) にタブコード (\t) を挿入し, 出力するということを最終行まで繰り返し, 最終的にファイル temp に出力している (4 行目). この工程で出力されたファイル temp を, スクリプト 1.4 の 9 行目で空行を取り除いた後, 10 行目で nkf [13] コマンド用いて文字コードを UTF-8 [14] に変換したものを最終的にファイル YA01tab.data〜YL01tab.data として出力している. なお, これらの工程は GNU parallel を用いて並列化することによって処理時間を短縮している.

前処理の最後の手順 (PP3) を実現するための scriptfirmlist.sh は, スクリプト 1.6 で与えられる.

スクリプト 1.6: scriptfirmlist.sh

```
1 export LC_ALL=C
2 #
3 echo Start unzip!
4 7z e -so ../datafiles/HFSCNM010.zip > fullfirmlist.txt
5 #
6 echo Complete unzip!
7 #
8 perl separatefirmlist.pl fullfirmlist.txt
```

[12] Perl (https://www.perl.org/) とは, 簡潔な記述や柔軟性, 拡張性の高さが特徴的な高水準のプログラミング言語の一つであり, Practical Extraction and Report Language の略である. Larry Wall 氏によって開発された (IT 用語辞典 e-Words https://e-words.jp/ 参照).

[13] 文字コードを変換するフィルタプログラムの一つ. Network Kanji Filter の略である. https://ja.osdn.net/projects/nkf/

[14] UTF-8 とは, Unicode/UCS で定義された文字集合を表現することができる文字コード (符号化方式) の一つ. UCS Transformation Format 8 または Unicode Transformation Format-8 の略 (IT 用語辞典 e-Words https://e-words.jp/ 参照).

```
 9 sed '/^$/d' temp | nkf --utf8 > firmlist.txt
10 rm temp;
11 #
12 echo Complete Tables!
13 #
14 rm  fullfirmlist.txt
15 #
16 echo Finish!
```

このシェルスクリプトの 4 行目で, 7z コマンドを用いて収録会社に関する情報が納められた ZIP ファイル HFSCNM010.zip を展開した後, リダイレクション (>) 機能を使って, テキストファイル fullfirmlist.txt に出力している. 8 行目で perl を利用して, 分離された個々のファイルの適切な箇所に分割記号を挿入している. その際, 引数としてファイル名 fullfirmlist.txt を与え, Perl スクリプトファイル separatefirmlist.pl (スクリプト 1.7 参照) を実行している.

スクリプト 1.7: separatefirmlist.pl

```
 1 #!/usr/bin/perl
 2 #####
 3 open(IN,"@ARGV");
 4 open(OUT,">temp");
 5 #####
 6 do{
 7 read(IN,$xx,302);
 8 $a[1]=substr($xx,0,4);
 9 $a[2]=substr($xx,4,7);
10 $a[3]=substr($xx,11,4);
11 $a[4]=substr($xx,15,1);
12 $a[5]=substr($xx,16,9);
13 $a[6]=substr($xx,25,4);
14 $a[7]=substr($xx,29,11);
15 $a[8]=substr($xx,40,30);
16 $a[9]=substr($xx,70,30);
17 $a[10]=substr($xx,100,50);
18 $a[11]=substr($xx,150,80);
19 $a[12]=substr($xx,230,7);
20 $a[13]=substr($xx,237,15);
21 $a[14]=substr($xx,252,27);
22 $a[15]=substr($xx,279,6);
23 $a[16]=substr($xx,285,13);
24 $a[17]=substr($xx,298,2);
25 #
26 print OUT
27 "$a[1]\t$a[2]\t$a[3]\t$a[4]\t$a[5]\t$a[6]\t$a[7]\t$a[8]\t$a[9]\t$a[10]\t$a
       [11]\t$a[12]\t$a[13]\t$a[14]\t$a[15]\t$a[16]\t$a[17]\n";
28 } while (eof(IN)!=1);
29 close (IN);
```

このスクリプトの, 3 行目では, 引数 ("@ARGV") として与えられたファイル fullfirmlist.txt を入力 (IN) とし, 1 行づつ読み取りながら, 適切に文字列を切り出したもの (8 行目から 24 行目) にタブコード (\t) を挿入し, 出力するということを最終行まで繰り返し (26 行目から 28 行目), 最終的にファイル temp に出力している (4 行目). この工程で出力されたファイル temp を, スクリプト 1.6 の 9 行目で sed コマンドを用いて空行を取り除いた後, さらに nkf コマンド用いて文字コードを UTF-8 に変換したものを最終的にファイル firmlist.txt として出力している.

以上の前処理におけるデータに関するファイルの流れを可視化したものを図 1.5 に与える.

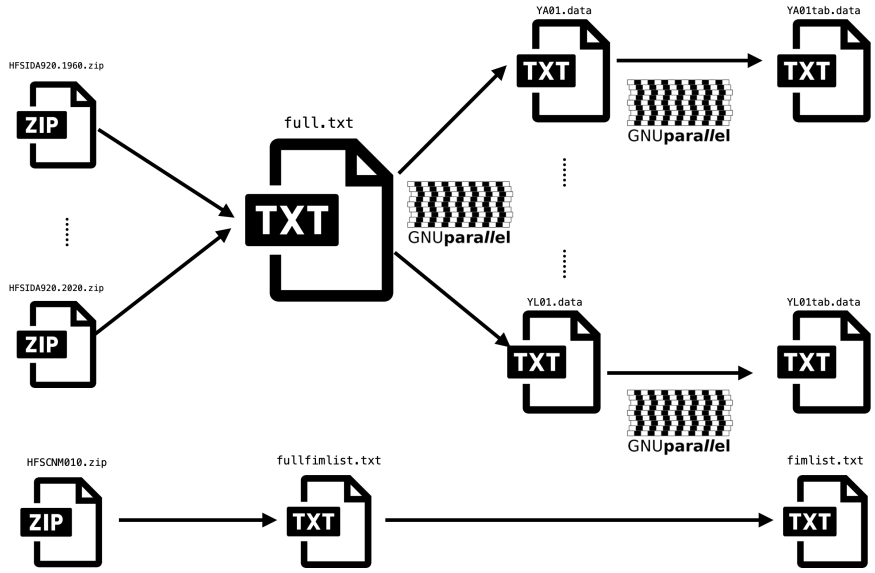


図 1.5: NEEDS 企業財務データ (一般事業会社) 2020 年版の前処理におけるデータに関するファイルの流れ

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 1.6 に与える.

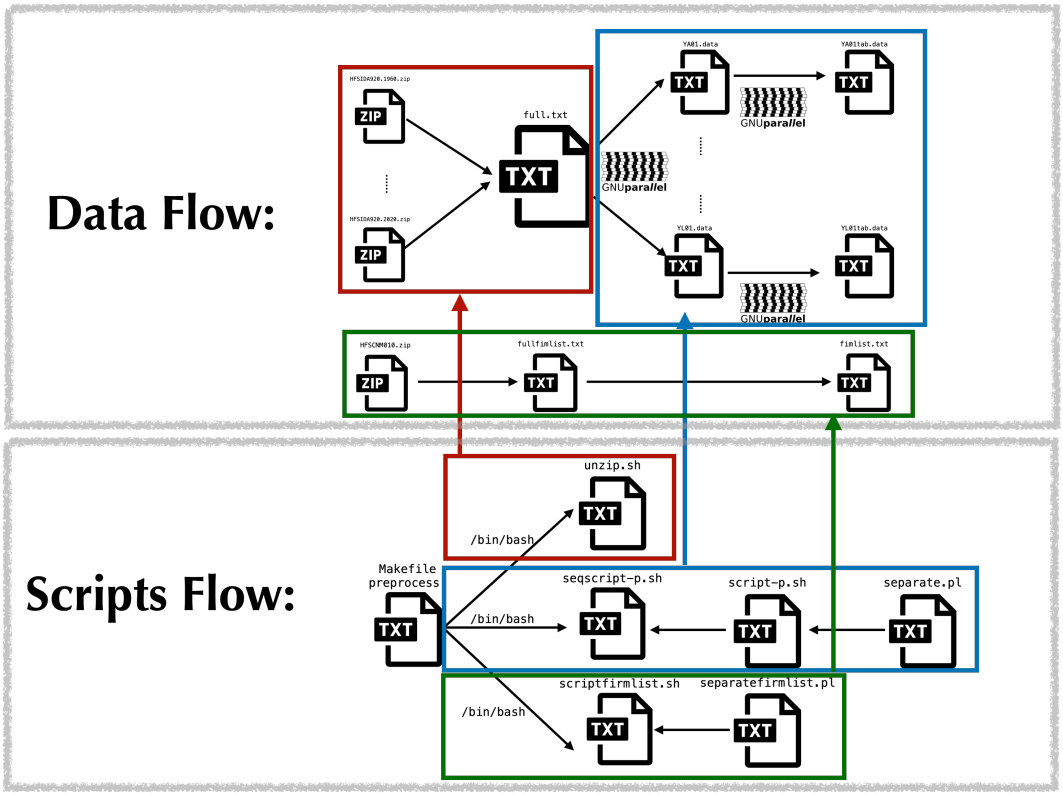


図 1.6: NEEDS 企業財務データ (一般事業会社) 2020 年版の前処理におけるシェルスクリプトとデータに関するファイルの流れ

なお, 前処理は, ディレクトリ `DBMaking` (図 1.2 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 1.1 において, 2, 6 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
Sat Jan 16 13:30:45 JST 2021
masa@aule DBMaking % cat end-preprocess.txt
Sat Jan 16 13:36:25 JST 2021
```

この結果から, 5 分 40 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 preprocess の処理時間の計測

```
masa@balin:~/data/NEEDS/MT-general2020-Ubuntu/DBMaking$ cat start-preprocess.txt
Thu Jan 28 13:11:59 JST 2021
masa@balin:~/data/NEEDS/MT-general2020-Ubuntu/DBMaking$ cat end-preprocess.txt
Thu Jan 28 13:15:28 JST 2021
```

この結果から, 3 分 29 秒である [16].

1.2.2 データベース needs2020, needs2020cons, needs2020uncons の構築

ここでは, 構築手順 (NS2), (NS3), (NS4) におけるデータベース needs2020 (全体), needs2020cons (連結本決算), needs2020uncons (単独本決算) の構築工程を詳細に述べる. その際, RDBMS として, MySQL と PostgreSQL を利用する場合に分けて考え, さらに macOS と Ubuntu の間で差異がある場合については適宜場合わけする.

なお, 最終的に構築したデータベースに対しては, 適当なユーザを作成し, アクセスする権限を与える必要があるが, セキュリティの観点から, それらの設定に関する詳細は割愛する.

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　構築手順 (NS2), (NS3), (NS4) 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 1.8 参照). このスクリプトおける, 2 行目と 9 行

[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 コマンド (Homebrew) と apt コマンド (Advanced Packaging Tool) 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様が異なっているためである.

目は処理時間を計測するための指定である.

スクリプト 1.8: Makfile: ターゲット MSDB (macOS)

```
1 MSDB:
2       date > start-MSDB.txt
3       /bin/bash replaceloadCDIR.sh
4       mysql -u root  --local_infile=1 -p******** < ./loadtoMySQL.sql
5       /bin/bash replaceloadfirmlist.sh
6       mysql -u root  --local_infile=1 -p******** < ./loadfirmlisttoMySQL.
            sql
7       mysql -u root --local_infile=1 -p******** < ./mkcons.sql
8       mysql -u root --local_infile=1 -p******** < ./mkuncons.sql
9       date > end-MSDB.txt
```

スクリプト 1.8 の 3, 4 行目が構築手順 (NS2) を実現するためのものであり, 5, 6 行目によって構築手順 (NS3) が, さらに 7, 8 行目の指定で構築手順 (NS4) が実現する. 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 1.7 に与える.

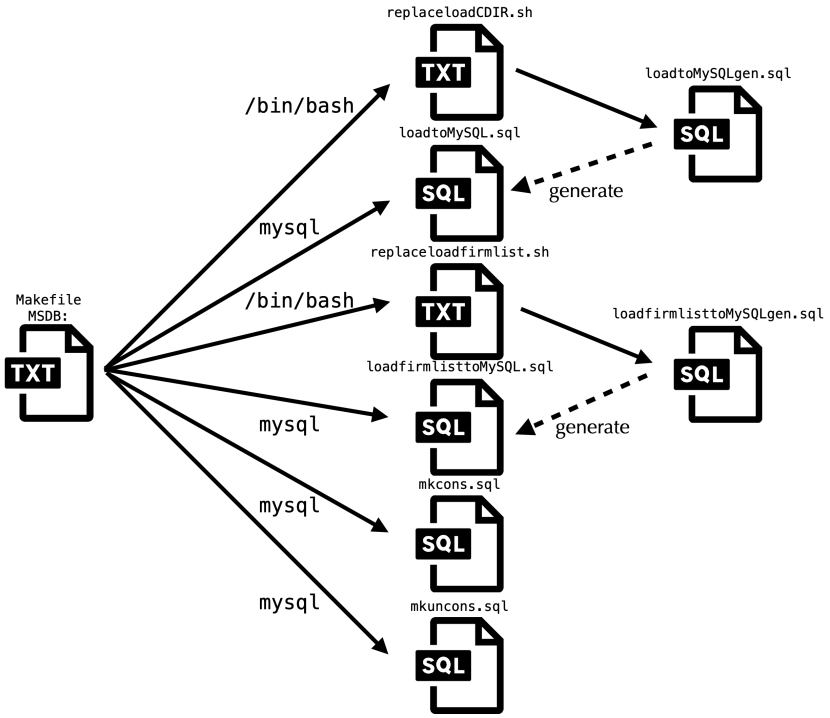


図 1.7: ターゲット MSDB の実行に伴うスクリプトファイルの流れ

まず, 構築手順 (NS2) 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 replaceloadCDIR.sh の内容 (スクリプト 1.9) 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.sql (スクリプト 1.10) 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 1.10 の 20〜22 行目) し, その結果を SQL スクリプトファイル loadtoMySQL.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.sql は, シェルスクリプト replaceloadCDIR.sh によって SQL スクリプトファイル loadtoMySQLgen.sql から生成 (generate) される (図 1.7 参照).

スクリプト 1.9: replaceloadCDIR.sh

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

スクリプト 1.10: loadtoMySQLgen.sql(一部抜粋)

```
1  DROP DATABASE IF EXISTS needs2020;
2  CREATE DATABASE needs2020;
3  USE needs2020
4  DROP TABLE IF EXISTS ya01;
5  : (中略)
6  DROP TABLE IF EXISTS yl01;
7  CREATE TABLE ya01 (
8  a01 VARCHAR(4),
9  a02 INT(6),
10 : (中略)
11 a46 VARCHAR(3),
12 b001 VARCHAR(14),
13 b002 VARCHAR(14),
14 : (中略)
15 b210 VARCHAR(14)
16 );
17 CREATE TABLE yb01 LIKE ya01;
18 : (中略)
19 CREATE TABLE yl01 LIKE ya01;
20 LOAD DATA LOCAL INFILE 'CDIR/YA01tab.data' INTO TABLE ya01 FIELDS
       TERMINATED BY '\t';
21 : (中略)
22 LOAD DATA LOCAL INFILE 'CDIR/YL01tab.data' INTO TABLE yl01 FIELDS
       TERMINATED BY '\t';
```

生成された SQL スクリプトファイル loadtoMySQL.sql は, データベース needs2020 と, テーブル ya01〜yl01 を作成 (2〜19 行目) した後, データファイル YA01tab.data〜YL01tab.data から, テーブル ya01〜yl01 へデータをロードするためのものであり, 実際にターゲット MSDB (スクリプト 1.8) の 4 行目で実行される.

次に, 構築手順 (NS3) 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 1.8) の 5 行目で実行されるスクリプトファイル replaceloadfirmlist.sh (スクリプト 1.11) は, replaceloadCDIR.sh (スクリプト 1.9) と同様の役割を果たす. つまり, 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, 3 行目で sed コマンドを利用して, SQL スクリプトファイル loadfirmlisttoMySQLgen.sql (スクリプト 1.12) 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 1.12 の 22 行目) し, その結果を SQL スクリプトファイル loadfirmlisttoMySQL.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 loadfirmlisttoMySQL.sql は, シェルスクリプト replaceloadfirmlist.sh によって SQL スクリプトファイル loadfirmlisttoMySQLgen.sql から生成される (図 1.7 参照).

スクリプト 1.11: replaceloadfirmlist.sh

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

スクリプト 1.12: loadfirmlisttoMySQLgen.sql

```
1  USE needs2020;
2  DROP TABLE IF EXISTS firmlist;
3  CREATE TABLE firmlist (
4  record_type VARCHAR(4),
5  nikkei_corp_code VARCHAR(7),
6  stock_code VARCHAR(4),
7  etc1 VARCHAR(1),
8  etc2 VARCHAR(9),
9  finance_code VARCHAR(4),
10 etc3 VARCHAR(11),
11 firmname VARCHAR(30),
12 firmname_jp VARCHAR(30),
13 firmname_jpk VARCHAR(50),
14 address VARCHAR(80),
15 zip_code VARCHAR(7),
16 phone_number VARCHAR(15),
17 etc4 VARCHAR(27),
18 nikkei_ind_code VARCHAR(6),
19 corp_number VARCHAR(13),
20 etc5 VARCHAR(2)
21 );
22 LOAD DATA LOCAL INFILE 'CDIR/firmlist.txt' INTO TABLE firmlist FIELDS
       TERMINATED BY '\t';
```

生成された SQL スクリプトファイル loadfirmlisttoMySQL.sql は, データベース needs2020 においてテーブル firmlist を作成 (3〜21 行目) した後, 収録企業に関する情報が収められたファイル firmlist.txt から, テーブル firmlist へデータをロードするためのものであり, 実際にターゲット MSDB (スクリプト 1.8) の 6 行目で実行される.

最後に, 構築手順 (NS4) を実現するためのスクリプトをみる. Makefile のターゲット MSDB (スクリプト 1.8) の 7 行目で指定されている SQL スクリプトファイル mkcons.sql (スクリプト 1.13) は, データベース needs2020 から, 連結本決算のデータベース needs2020cons を分離するためのものである. また, 8 行目で指定されている SQL スクリプトファイル mkuncons.sql (スクリプト 1.14) は, データベース needs2020 から, 連結本決算のデータベース needs2020uncons を分離するためのものである. これらのスクリプトは, ほぼ機能が同じであるので, SQL スクリプトファイル mkcons.sql (スクリプト 1.13) を主に説明する. まず, 2 行目で連結本決算のデータベース needs2020cons を作成した後, テーブル ya01〜yl01 を作成している (7 行目から 19 行目). さらに, 収録企業のリストのテーブル firmlist を作成 (22 行目から 40 行目) し, 41 行目で既存のデータベースのテーブル needs2020.ya01 の全ての列 (*) から, SQL 問合せに関する WHERE 句に連結本決算を満たす条件を与え, 制約付きでレコードを抽出し, 新しいデータベースのテーブル needs2020cons.ya01 へ挿入している. ここで,

(WHERE **句条件** 1)　　WHERE a09='2' AND a10='10' AND a27 !='0'

は, ヘッダーパートの仕様 (表 8.3) から, 「連結・単独フラグ」a09 として 2 (連結) を選択し, かつ「決算別フラグ」a10 として 10 (本決算), さらに, 「連結基準フラグ」a27 として 0 (単独) でないものを選択していることがわかる.

スクリプト 1.13: mkcons.sql

```
1  DROP DATABASE IF EXISTS needs2020cons;
2  CREATE DATABASE needs2020cons;
3  USE needs2020cons
4  DROP TABLE IF EXISTS ya01;
```

```
5  : (中略)
6  DROP TABLE IF EXISTS yl01;
7  CREATE TABLE ya01 (
8  a01 VARCHAR(4),
9  a02 INT(6),
10 : (中略)
11 a46 VARCHAR(3),
12 b001 VARCHAR(14),
13 b002 VARCHAR(14),
14 : (中略)
15 b210 VARCHAR(14)
16 );
17 CREATE TABLE yb01 LIKE ya01;
18 : (中略)
19 CREATE TABLE yl01 LIKE ya01;
20 #
21 DROP TABLE IF EXISTS firmlist;
22 CREATE TABLE firmlist (
23 record_type VARCHAR(4),
24 nikkei_corp_code VARCHAR(7),
25 stock_code VARCHAR(4),
26 etc1 VARCHAR(1),
27 etc2 VARCHAR(9),
28 finance_code VARCHAR(4),
29 etc3 VARCHAR(11),
30 firmname VARCHAR(30),
31 firmname_jp VARCHAR(30),
32 firmname_jpk VARCHAR(50),
33 address VARCHAR(80),
34 zip_code VARCHAR(7),
35 phone_number VARCHAR(15),
36 etc4 VARCHAR(27),
37 nikkei_ind_code VARCHAR(6),
38 corp_number VARCHAR(13),
39 etc5 VARCHAR(2)
40 );
41 INSERT INTO needs2020cons.ya01 SELECT * FROM needs2020.ya01 WHERE a09='2'
       AND a10='10' AND a27!='0';
42 : (中略)
43 INSERT INTO needs2020cons.yl01 SELECT * FROM needs2020.yl01 WHERE a09='2'
       AND a10='10' AND a27!='0';
44 INSERT INTO needs2020cons.firmlist SELECT * FROM needs2020.firmlist;
```

同様の処理をテーブル yl01 まで行い (43 行目), 最後に firmlist については全く同じテーブルをデータベース needs2020 から needs2020cons へ挿入している.

スクリプト 1.14: mkuncons.sql

```
1  DROP DATABASE IF EXISTS needs2020uncons;
2  CREATE DATABASE needs2020uncons;
3  USE needs2020uncons
4  DROP TABLE IF EXISTS ya01;
5  : (中略)
6  DROP TABLE IF EXISTS yl01;
7  CREATE TABLE ya01 (
8  a01 VARCHAR(4),
9  a02 INT(6),
10 : (中略)
11 a46 VARCHAR(3),
12 b001 VARCHAR(14),
```

```
b002 VARCHAR(14),
: (中略)
b210 VARCHAR(14)
);
CREATE TABLE yb01 LIKE ya01;
: (中略)
CREATE TABLE yl01 LIKE ya01;
#
DROP TABLE IF EXISTS firmlist;
CREATE TABLE firmlist (
record_type VARCHAR(4),
nikkei_corp_code VARCHAR(7),
stock_code VARCHAR(4),
etc1 VARCHAR(1),
etc2 VARCHAR(9),
finance_code VARCHAR(4),
etc3 VARCHAR(11),
firmname VARCHAR(30),
firmname_jp VARCHAR(30),
firmname_jpk VARCHAR(50),
address VARCHAR(80),
zip_code VARCHAR(7),
phone_number VARCHAR(15),
etc4 VARCHAR(27),
nikkei_ind_code VARCHAR(6),
corp_number VARCHAR(13),
etc5 VARCHAR(2)
);
INSERT INTO needs2020uncons.ya01 SELECT * FROM needs2020.ya01 WHERE a09='1'
    AND a10='10' AND a27='0';
: (中略)
INSERT INTO needs2020uncons.yl01 SELECT * FROM needs2020.yl01 WHERE a09='1'
    AND a10='10' AND a27='0';
INSERT INTO needs2020uncons.firmlist SELECT * FROM needs2020.firmlist;
```

なお, SQL スクリプト 1.13 と 1.14 の (データベース名以外の) 異なっている箇所としては,

```
(WHERE 句条件 2)                WHERE a09='1' AND a10='10' AND a27 =='0'
```

であるが, ヘッダーパートの仕様 (表 8.3) から,「連結・単独フラグ」a09 として 1 (単独) を選択し, かつ「決算別フラグ」 a10 として 10 (本決算), さらに,「連結基準フラグ」 a27 として 0 (単独) を選択している点にあり, この指定によって単独本決算のデータベース needs2020uncons を構築できる.

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 1.8 に与える.

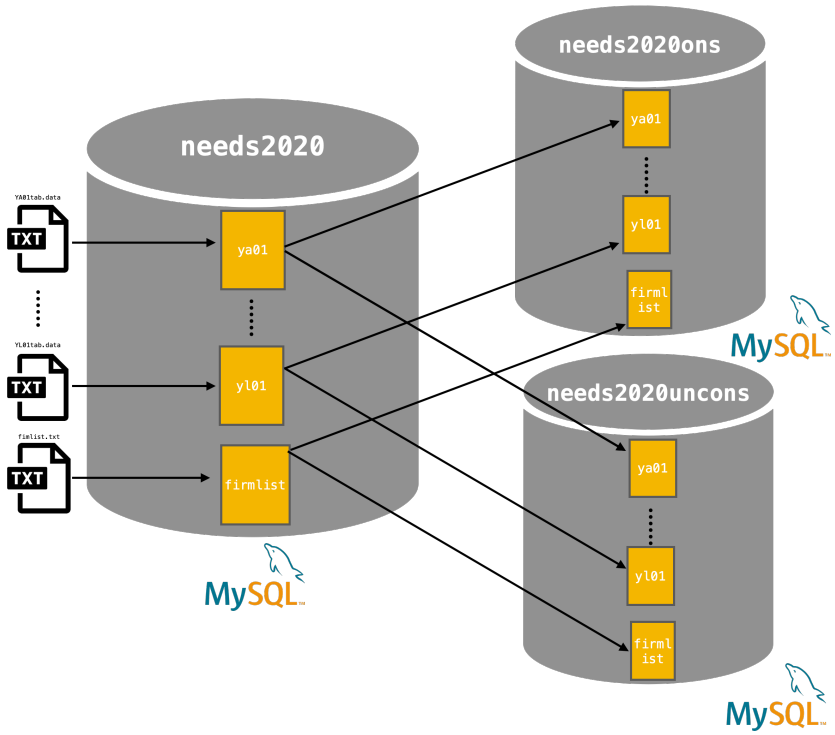


図 1.8: MySQL によるデータベース needs2020, needs2020cons, needs2020uncons の構築

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 1.9 に与える.

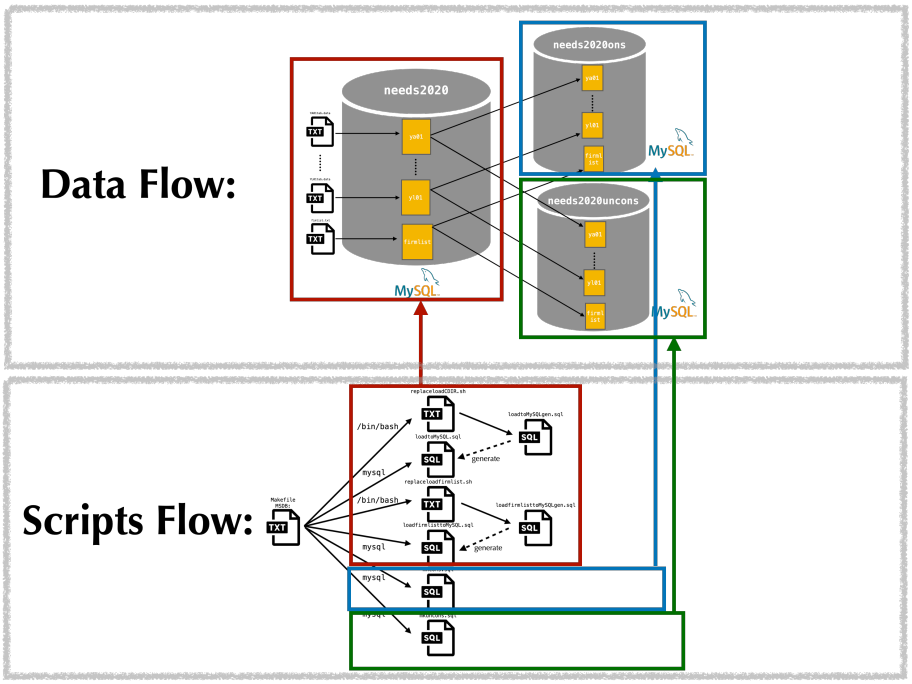


図 1.9: ターゲット MSDB の実行に伴うシェルスクリプトとデータに関するファイルの流れ

図 1.8, 1.9 は macOS 上でのデータベースの構築の流れを表しているが, Ubuntu 上でも全く同じことがい

えることに注意が必要である.

なお, ターゲット MSDB はディレクトリ DBMaking (図 1.2 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 1.8 において, 2, 9 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
Sat Jan 16 13:37:37 JST 2021
masa@aule DBMaking % cat end-MSDB.txt
Sat Jan 16 13:42:41 JST 2021
```

この結果から, 5 分 4 秒であることがわかる [18].

■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 1.15: Makfile: ターゲット MSDB (Ubuntu)

```
1 MSDB:
2       date > start-MSDB.txt
3       /bin/bash replaceloadCDIR.sh
4       sudo mysql --local_infile=1 < ./loadtoMySQL.sql
5       /bin/bash replaceloadfirmlist.sh
6       sudo mysql --local_infile=1 < ./loadfirmlisttoMySQL.sql
7       sudo mysql --local_infile=1 < ./mkcons.sql
8       sudo mysql --local_infile=1 < ./mkuncons.sql
9       date > end-MSDB.txt
```

macOS 用のターゲット MSDB (スクリプト 1.8) と, Ubuntu 用のもの (スクリプト 1.15) 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 PostgreSQL のバージョンと仕様に差異があるためである [19].

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 1.3 のディレクトリ MT-general2020-Ubuntu において, 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 1.15 において, 2, 19 行目の実行結果を比較することによってわかる.

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

[19] macOS と Ubuntu の MySQL の設定を詳細に行うことによって, 同じスクリプトを利用することも可能と思われるが, ここではスクリプトを OS 毎に変更することによって対応した.

Ubuntu 上でターゲット MSDB の処理時間の計測

```
masa@balin:~/data/NEEDS/MT-general2020-Ubuntu/DBMaking$ cat start-MSDB.txt
Thu Jan 28 13:15:36 JST 2021
masa@balin:~/data/NEEDS/MT-general2020-Ubuntu/DBMaking$ cat end-MSDB.txt
Thu Jan 28 13:54:22 JST 2021
```

この結果から, 38 分 46 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, それぞれの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　構築手順 (NS2), (NS3), (NS4) 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 1.16 参照). スクリプト 1.16 において, 2 行目と 19 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 1.16: Makfile: ターゲット PGDB (macOS)

```
1  PGDB:
2        date > start-PGDB.txt
3        /bin/bash replaceloadPostgreSQL-CDIR.sh
4        psql postgres < ./loadtoPostgreSQL.sql
5        /bin/bash replaceloadfirmlistPostgreSQL.sh
6        psql postgres < ./loadfirmlisttoPostgreSQL.sql
7        /bin/bash replacedumpPostgreSQL-CDIR-cons.sh
8        psql postgres < ./dumpfromPostgreSQL-cons.sql
9        /bin/bash replaceloadPostgreSQL-CDIR-cons.sh
10       psql postgres < ./loadtoPostgreSQL-cons.sql
11       /bin/bash replaceloadfirmlistPostgreSQL-cons.sh
12       psql postgres < ./loadfirmlisttoPostgreSQL-cons.sql
13       /bin/bash replacedumpPostgreSQL-CDIR-uncons.sh
14       psql postgres < ./dumpfromPostgreSQL-uncons.sql
15       /bin/bash replaceloadPostgreSQL-CDIR-uncons.sh
16       psql postgres < ./loadtoPostgreSQL-uncons.sql
17       /bin/bash replaceloadfirmlistPostgreSQL-uncons.sh
18       psql postgres < ./loadfirmlisttoPostgreSQL-uncons.sql
19       date > end-PGDB.txt
```

スクリプト 1.16 の 3, 4 行目が構築手順 (NS2) を実現するためのものであり, 5, 6 行目によって構築手順 (NS3) が, さらに 7〜18 行目の指定で構築手順 (NS4) が実現する. 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 1.10 に与える.

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

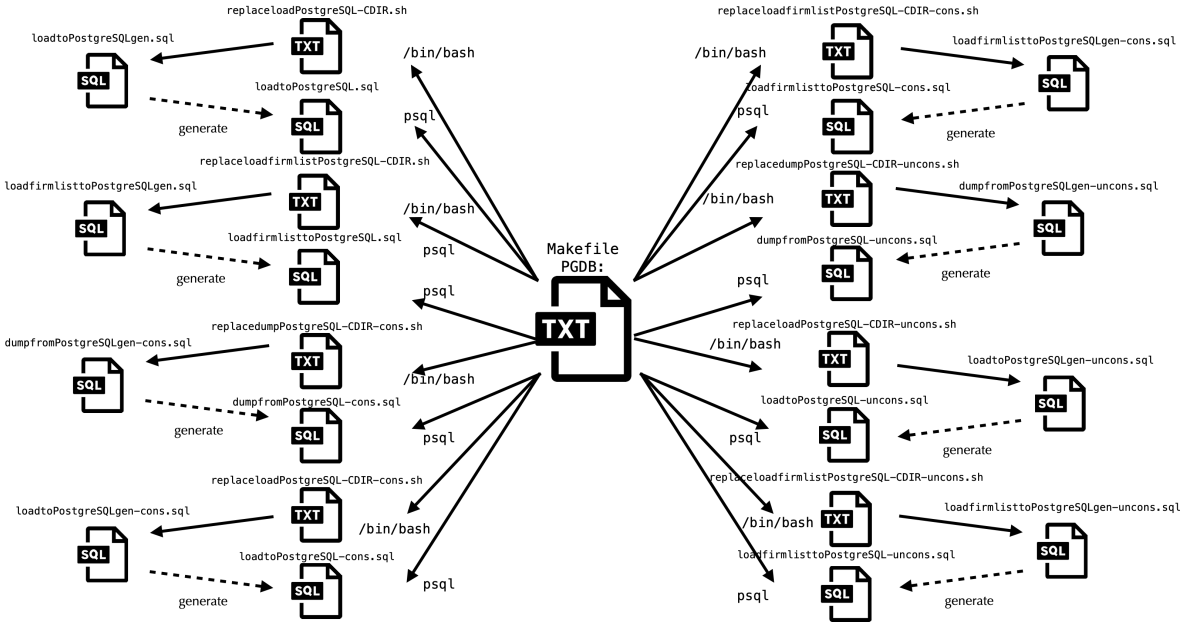


図 1.10: ターゲット PGDB の実行に伴うスクリプトファイルの流れ

まず, 構築手順 (NS2) 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.sh の内容 (スクリプト 1.17) 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.sql (スクリプト 1.18) 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 1.18 の 20〜22 行目) し, その結果を SQL スクリプトファイル loadtoPostgreSQL.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.sql は, シェルスクリプト replaceloadPostgreSQL-CDIR.sh によって SQL スクリプトファイル loadtoPostgreSQLgen.sql から生成 (generate) される (図 1.10 参照).

スクリプト 1.17: replaceloadPostgreSQL-CDIR.sh

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

スクリプト 1.18: loadtoPostgreSQLgen.sql(一部抜粋)

```
1  DROP DATABASE IF EXISTS needs2020;
2  CREATE DATABASE needs2020;
3  \c needs2020
4  DROP TABLE IF EXISTS ya01;
5  : (中略)
6  DROP TABLE IF EXISTS yl01;
7  CREATE TABLE ya01 (
8  a01 VARCHAR(4),
9  a02 BIGINT,
10 : (中略)
11 a46 VARCHAR(3),
12 b001 VARCHAR(14),
13 b002 VARCHAR(14),
14 : (中略)
15 b210 VARCHAR(14)
```

```
16 );
17 CREATE TABLE yb01 (LIKE ya01);
18 : (中略)
19 CREATE TABLE yl01 (LIKE ya01);
20 COPY public.ya01 FROM 'CDIR/YA01tab.data';
21 : (中略)
22 COPY public.yl01 FROM 'CDIR/YL01tab.data';
```

生成された SQL スクリプトファイル loadtoPostgreSQL.sql は, データベース needs2020 と, テーブル ya01〜yl01 を作成 (2〜19 行目) した後, データファイル YA01tab.data〜YL01tab.data から, テーブル ya01〜yl01 へデータをロードするためのものであり, 実際にターゲット PGDB (スクリプト 1.16) の 4 行目で実行される.

次に, 構築手順 (NS3) 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 1.16) の 5 行目で実行されるスクリプトファイル replaceloadfirmlistPostgreSQL.sh (スクリプト 1.19) は, replaceloadPostgreSQL-CDIR.sh (スクリプト 1.17) と同様の役割を果たす. つまり, 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, 3 行目で sed コマンドを利用して, SQL スクリプトファイル loadfirmlisttoPostgreSQLgen.sql (スクリプト 1.19) 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 1.20 の 22 行目) し, その結果を SQL スクリプトファイル loadfirmlisttoPostgreSQL.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 loadfirmlisttoPostgreSQL.sql は, シェルスクリプト replaceloadfirmlistPostgreSQL.sh によって SQL スクリプトファイル loadfirmlisttoPostgreSQLgen.sql から生成される (図 1.10 参照).

スクリプト 1.19: replaceloadfirmlistPostgreSQL.sh

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

スクリプト 1.20: loadfirmlisttoPostgreSQLgen.sql

```
1 \c needs2020
2 DROP TABLE IF EXISTS firmlist;
3 CREATE TABLE firmlist (
4 record_type VARCHAR(4),
5 nikkei_corp_code VARCHAR(7),
6 stock_code VARCHAR(4),
7 etc1 VARCHAR(1),
8 etc2 VARCHAR(9),
9 finance_code VARCHAR(4),
10 etc3 VARCHAR(11),
11 firmname VARCHAR(30),
12 firmname_jp VARCHAR(30),
13 firmname_jpk VARCHAR(50),
14 address VARCHAR(80),
15 zip_code VARCHAR(7),
16 phone_number VARCHAR(15),
17 etc4 VARCHAR(27),
18 nikkei_ind_code VARCHAR(6),
19 corp_number VARCHAR(13),
20 etc5 VARCHAR(2)
21 );
22 COPY public.firmlist FROM 'CDIR/firmlist.txt';
```

生成された SQL スクリプトファイル loadfirmlisttoPostgreSQL.sql は, データベース needs2020 においてテーブル firmlist を作成 (3〜21 行目) した後, 収録企業に関する情報が収められたファイル firmlist.txt から, テーブル firmlist へデータをロードするためのものであり, 実際にターゲット PGDB (スクリプト 1.16) の 6 行目で実行される.

最後に, 構築手順 (NS4) を実現するためのスクリプトを解説する. ここで, 注意を要する事項としては, 今回利用した PostgreSQL のバージョンでは, MySQL のようにデータベースからデータベースへテーブルの情報を「流し込む」機能がデフォルトで用意されていないということである. すなわち, 新たなデータベースを作るためには, 一旦データをデータベースからダンプした後, 改めてデータをロードする必要がある. このため, PostgreSQL 用のターゲットは MySQL のものに比べて冗長な記述となっている.

Makefile のターゲット PGDB (スクリプト 1.16) における 7〜12 行目が連結本決算のデータベースを構築するためのものである. まず, 7 行目で指定されている replacedumpPostgreSQL-CDIR-cons.sh (スクリプト 1.21) は, replaceloadPostgreSQL-CDIR.sh (スクリプト 1.17) と同様の役割を果たす. つまり, 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, 3 行目で sed コマンドを利用して, SQL スクリプトファイル dumpfromPostgreSQLgen-cons.sql (スクリプト 1.22) 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 1.22 の 2〜4 行目) し, その結果を SQL スクリプトファイル dumpfromPostgreSQL-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 dumpfromPostgreSQL-cons.sql は, シェルスクリプト replacedumpPostgreSQL-CDIR-cons.sh によって SQL スクリプトファイル dumpfromPostgreSQLgen-cons.sql から生成される (図 1.10 参照).

スクリプト 1.21: replacedumpPostgreSQL-CDIR-cons.sh

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

スクリプト 1.22: dumpfromPostgreSQLgen-cons.sql (一部抜粋)

```
1 \c needs2020
2 \COPY (SELECT * FROM public.ya01 WHERE a09='2' AND a10='10' AND a27!='0')
      TO 'CDIR/YA01tab.cons.data' WITH CSV DELIMITER E'\t';
3 : (中略)
4 \COPY (SELECT * FROM public.yl01 WHERE a09='2' AND a10='10' AND a27!='0')
      TO 'CDIR/YL01tab.cons.data' WITH CSV DELIMITER E'\t';
```

生成された SQL スクリプトファイル dumpfromPostgreSQL-cons.sql は, 既存のデータベースのテーブル needs2020.ya01 の全ての列 (*) から, SQL 問合せに関する WHERE 句に連結本決算を満たす条件 (WHERE **句条件 1**) 参照) を与え, 制約付きでレコードを抽出し, カレントディレクトリへタブ区切りファイル YA01tab.cons.data として出力している (2 行目). これを同様に繰り返し, カレントディレクトリへタブ区切りファイル YL01tab.cons.data として出力している (4 行目). このスクリプトは, 実際にターゲット PGDB (スクリプト 1.16) の 8 行目で実行される. この工程で出力された連結本決算のデータファイル YA01tab.cons.data〜YL01tab.cons.data にもとづくデータベースを構築するスクリプトがターゲット PGDB (スクリプト 1.16) の 9, 10 行目である. 9 行目で指定されている replaceloadPostgreSQL-CDIR-cons.sh (スクリプト 1.23) は, replaceloadPostgreSQL-CDIR.sh (スクリプト 1.17) と同様の役割を果たす. つまり, 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, 3 行目で 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 1.24) 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 1.24 の 22 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.

スクリプト 1.23: replaceloadPostgreSQL-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 1.24: loadtoPostgreSQLgen-cons.sql (一部抜粋)

```
1  DROP DATABASE IF EXISTS needs2020cons;
2  CREATE DATABASE needs2020cons;
3  \c needs2020cons
4  DROP TABLE IF EXISTS ya01;
5  : (中略)
6  DROP TABLE IF EXISTS yl01;
7  CREATE TABLE ya01 (
8  a01 VARCHAR(4),
9  a02 BIGINT,
10 : (中略)
11 a46 VARCHAR(3),
12 b001 VARCHAR(14),
13 b002 VARCHAR(14),
14 : (中略)
15 b210 VARCHAR(14)
16 );
17 CREATE TABLE yb01 (LIKE ya01);
18 : (中略)
19 CREATE TABLE yl01 (LIKE ya01);
20 COPY public.ya01 FROM 'CDIR/YA01tab.cons.data';
21 : (中略)
22 COPY public.yl01 FROM 'CDIR/YL01tab.cons.data';
```

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 needs2020cons と, テーブル ya01〜yl01 を作成 (2〜19 行目) した後, データファイル YA01tab.cons.data〜YL01tab.cons.data から, テーブル ya01〜yl01 へデータをロードするためのものであり, 実際にターゲット PGDB (スクリプト 1.16) の 10 行目で実行される. 連結本決算のデータベース needs2020cons の仕上げとして, 収録企業に関する情報が収められたテーブル firmlist を作成するためのスクリプトについて説明する. ターゲット PGDB (スクリプト 1.16) の 11 行目で実行される replaceloadfirmlistPostgresSQL-cons.sh (スクリプト 1.25) の 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, 3 行目で sed コマンドを利用して, SQL スクリプトファイル loadfirmlisttoPostgreSQLgen-cons.sql (スクリプト 1.26) 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 1.26 の 22 行目) し, その結果を SQL スクリプトファイル loadfirmlisttoPostgreSQL-cons.sql にリダイレクション (>) 機能を使って出力している. よって, SQL スクリプトファイル loadfirmlisttoPostgreSQL-cons.sql は, シェルスクリプト replaceloadfirmlistPostgreSQL.sh から生成される。

スクリプト 1.25: replaceloadfirmlistPostgreSQL-cons.sh

```
1 #!/bin/bash
```

```
2 CDIR=$PWD
3 sed -e "s|CDIR|$CDIR|g" loadfirmlisttoPostgreSQLgen-cons.sql >
      loadfirmlisttoPostgreSQL-cons.sql
```

スクリプト 1.26: loadfirmlisttoPostgreSQLgen-cons.sql

```
1 \c needs2020cons;
2 DROP TABLE IF EXISTS firmlist;
3 CREATE TABLE firmlist (
4 record_type VARCHAR(4),
5 nikkei_corp_code VARCHAR(7),
6 stock_code VARCHAR(4),
7 etc1 VARCHAR(1),
8 etc2 VARCHAR(9),
9 finance_code VARCHAR(4),
10 etc3 VARCHAR(11),
11 firmname VARCHAR(30),
12 firmname_jp VARCHAR(30),
13 firmname_jpk VARCHAR(50),
14 address VARCHAR(80),
15 zip_code VARCHAR(7),
16 phone_number VARCHAR(15),
17 etc4 VARCHAR(27),
18 nikkei_ind_code VARCHAR(6),
19 corp_number VARCHAR(13),
20 etc5 VARCHAR(2)
21 );
22 \COPY public.firmlist FROM 'CDIR/firmlist.txt';
```

生成された SQL スクリプトファイル loadfirmlisttoPostgreSQL-cons.sql は, データベース needs2020cons においてテーブル firmlist を作成 (3〜21 行目) した後, 収録企業に関する情報が収められたファイル firmlist.txt から, テーブル firmlist へデータをロードするためのものであり, 実際にターゲット PGDB (スクリプト 1.16) の 12 行目で実行される.

以上の処理 (Makefile のターゲット PGDB (スクリプト 1.16) の 7〜12 行目の実行) によって, 連結本決算のデータベースが構築される. ターゲット PGDB (スクリプト 1.16) の残りの処理 (13〜18 行目) は, 単独本決算のデータベースを構築するためのもので, 連結本決算のデータベースを構築するための処理との違いは, スクリプト中で文字列 cons を uncons に置き換えることと, 既存のデータベースのテーブル needs2020.* から単独本決算のデータを抽出する際の条件が (WHERE **句条件** 2)) に変更されている点以外は, 本質的には変わらない. よって, 冗長となるため説明は割愛する.

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 1.11 に与える.

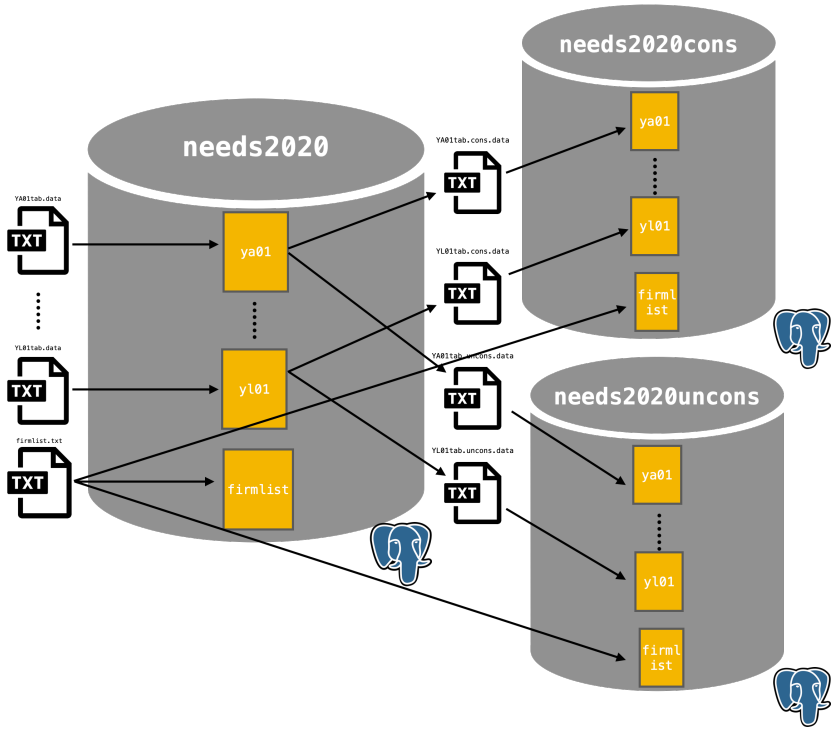


図 1.11:　macOS 環境のもとでの PostgreSQL によるデータベース needs2020, needs2020cons, needs2020uncons の構築 (MAPP)

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 1.12 に与える.

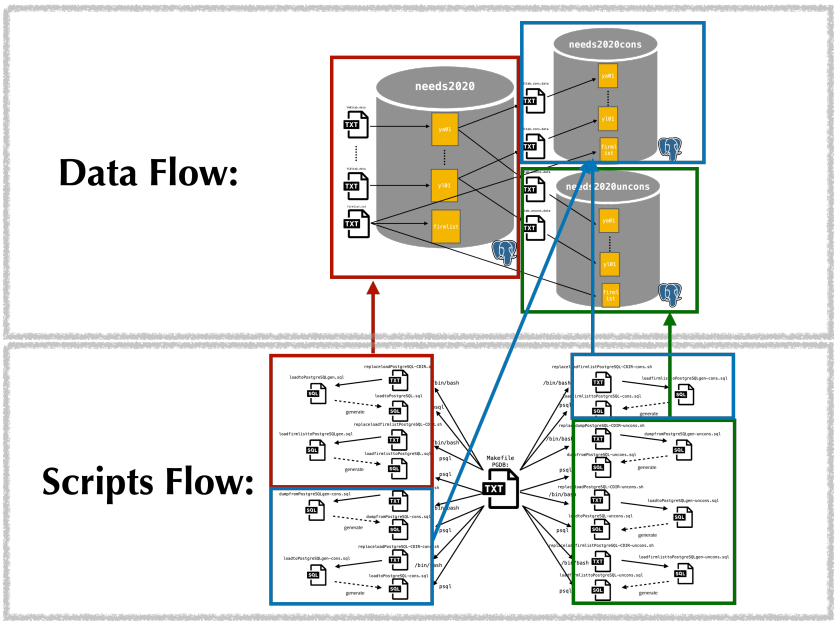


図 1.12: ターゲット PGDB の実行に伴うシェルスクリプトとデータに関するファイルの流れ

図 1.11, 1.12 は macOS 上でのデータベースの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ことに注意が必要である.

なお, ターゲット PGDB は, ディレクトリ DBMaking (図 1.2 参照) において, ターミナル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
```

この処理時間は, スクリプト 1.16 において, 2, 19 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
Sat Jan 16 13:49:16 JST 2021
masa@aule DBMaking % cat end-PGDB.txt
Sat Jan 16 13:54:21 JST 2021
```

この結果から, 5 分 5 秒であることがわかる [21].

■Ubuntu (LAPP) 環境のもとでのデータベース構築　MySQL の場合と同様に, macOS 環境 (MAPP) と Ubuntu 環境 (LAPP) 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 1.27: Makfile: ターゲット PGDB (Ubuntu)

```
1  PGDB:
2        date > start-PGDB.txt
3        /bin/bash replaceloadPostgreSQL-CDIR.sh
4        sudo -u postgres psql < ./loadtoPostgreSQL.sql
5        /bin/bash replaceloadfirmlistPostgreSQL.sh
6        sudo -u postgres psql < ./loadfirmlisttoPostgreSQL.sql
7        /bin/bash replacedumpPostgreSQL-CDIR-cons.sh
8        sudo -u postgres psql < ./dumpfromPostgreSQL-cons.sql
9        /bin/bash replaceloadPostgreSQL-CDIR-cons.sh
10       sudo -u postgres psql < ./loadtoPostgreSQL-cons.sql
11       /bin/bash replaceloadfirmlistPostgreSQL-cons.sh
12       sudo -u postgres psql < ./loadfirmlisttoPostgreSQL-cons.sql
13       /bin/bash replacedumpPostgreSQL-CDIR-uncons.sh
14       sudo -u postgres psql < ./dumpfromPostgreSQL-uncons.sql
15       /bin/bash replaceloadPostgreSQL-CDIR-uncons.sh
16       sudo -u postgres psql < ./loadtoPostgreSQL-uncons.sql
17       /bin/bash replaceloadfirmlistPostgreSQL-uncons.sh
18       sudo -u postgres psql < ./loadfirmlisttoPostgreSQL-uncons.sql
19       date > end-PGDB.txt
```

macOS 用のターゲット PGDB (スクリプト 1.16) と, Ubuntu 用のもの (スクリプト 1.27) 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 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 1.3 の MT-general2020-Ubuntu をカレントディレクトリとして, 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.

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

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 1.27 において, 2, 19 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:~/data/NEEDS/MT-general2020-Ubuntu/DBMaking$ cat start-PGDB.txt
Thu Jan 28 15:23:50 JST 2021
masa@balin:~/data/NEEDS/MT-general2020-Ubuntu/DBMaking$ cat end-PGDB.txt
Thu Jan 28 15:29:51 JST 2021
```

この結果から, 6 分 1 秒であることがわかる [22].

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

第 2 章

Osiris データ

本研究では, ビューロー・ヴァン・ダイク [1] のデータベース Osiris (オシリスまたはオサイリス) 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種類を拡充する [2]. なお, ここで利用する Osiris に関するデータの仕様の詳細は, 第 9 章を参照されたい.

2.1 データベースとデータセット

データベース Osiris は, 全世界の上場・上場廃止企業 [3] 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. ここでは, Osiris から, 金融・保険会社を除く上場及び上場廃止企業 96,377 社の連結 (Consolidated) 財務諸表 (以後, 「連結ベース」と略す), および非連結（単体：Un-consolidated）財務諸表 (以後, 「非連結ベース」と略す), それぞれを優先的に表示する設定のもとで抽出されたデータを利用する. 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 [4] (2 セット) からなり, 表 2.1, 表 2.2 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 2.1: 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 2.2: 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 によって並列化

[1] Bureau van Dijk `https://www.bvdinfo.com/`

[2] SKWAD から提供される Osiris データの抽出サービスの利用は, 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.

[3] データ抽出時点で金融機関を除いた 9 万社以上を収録されている.

[4] 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.

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 [5]. なお, 処理後のファイル (CSV [6] ファイル) 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 2.3 のようなものである.

表 2.3: 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.

2.2 Osiris データによるデータベースの構築

2.2.1 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), DS-Osiris-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 2.1 (macOS 環境と Ubuntu 環境で共通) を参照されたい [7].

```
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 2.1: 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:

[5] 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。

[6] 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.

[7] 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.

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 7.1 と図 **??** 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.

2.2.2 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 [8]. 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 2.1 参照).

スクリプト 2.1: 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 2.1 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 2.2) が指定されている.

スクリプト 2.2: script-preprocess.sh

```
1 #!/bin/bash
2 sed -f sedbs ../CSV/firmfinC2020.csv > ./firmfinC2020.csv
3 sed -f sedbs ../CSV/firmfinU2020.csv > ./firmfinU2020.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 2.3) に定義しておき, sed を利用して置換している.

スクリプト 2.3: 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 2.2 に与える.

[8] 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.

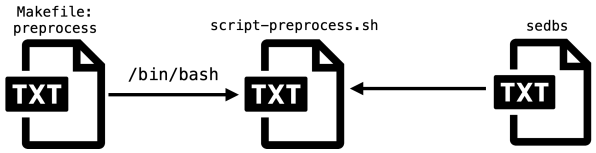


図 2.2: Osiris データファイルの前処理に関するシェルスクリプトの流れ

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 2.3 に与える.

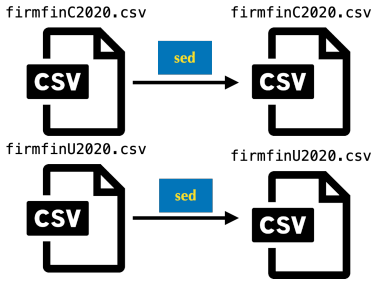


図 2.3: Osiris データの前処理におけるデータファイルの流れ

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 2.4 に与える.

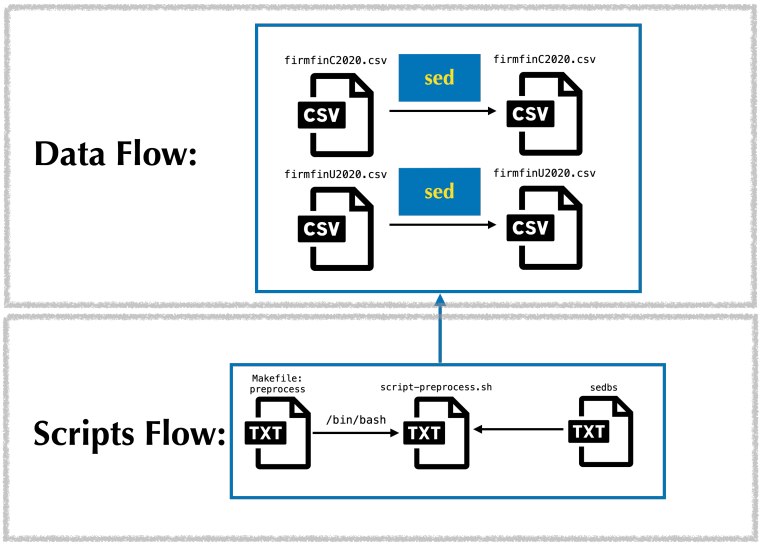


図 2.4: Osiris データの前処理におけるシェルスクリプトとデータに関するファイルの流れ

なお, 前処理は, ディレクトリ `DBMaking` (図 2.1 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 (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 2.1 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 [9].

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 [10].

2.2.3 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 2.4 参照).

スクリプト 2.4: Makefile: 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 2.4 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 2.5) 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.

スクリプト 2.5: rs10000-osiris2020.R

```
1 library(readr)
2 library(dplyr)
3 #
4 x <- read_csv("firmfinC2020.csv")
5 idx <- unique(x$ID)
6 set.seed(12345)
7 idx10000 <- sample(idx, 10000, replace = FALSE)
8 x %>% filter(ID %in% idx10000) %>%
9   write_csv(file = "firmfinC2020-10000.csv")
```

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

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

```
10 #
11 y <- read_csv("firmfinU2020.csv")
12 idy <- unique(y$ID)
13 set.seed(12345)
14 idy10000 <- sample(idy, 10000, replace = FALSE)
15 y %>% filter(ID %in% idy10000) %>%
16   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 2.5 に与える.

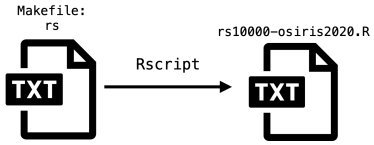


図 2.5: Osiris データファイルのランダムサンプリングに関するシェルスクリプトの流れ

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 2.6 に与える.

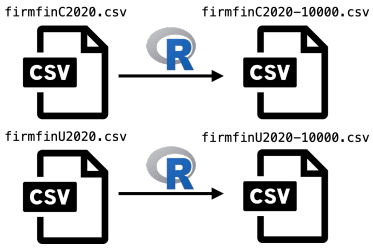


図 2.6: Osiris データのランダムサンプリングによるデータファイルの流れ

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 2.7 に与える.

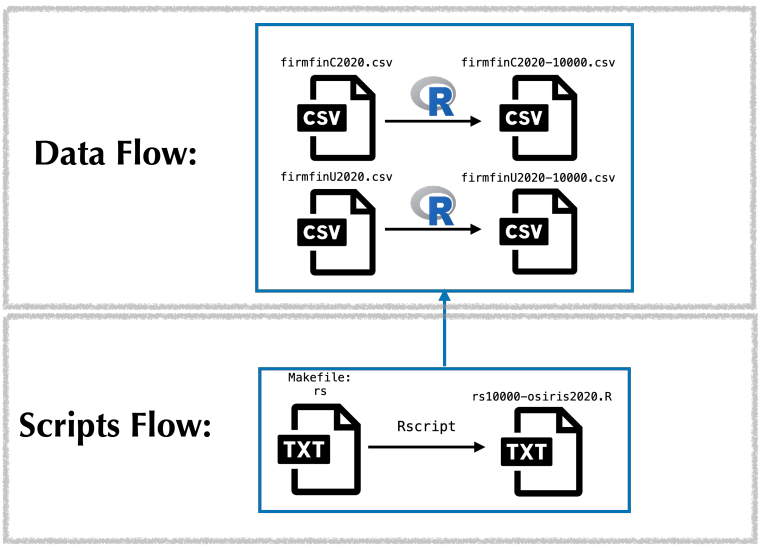


図 2.7: Osiris データのランダムサンプリングよるシェルスクリプトとデータに関するファイルの流れ

なお, この処理は, ディレクトリ DBMaking (図 2.1 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: macOS, Ubuntu 共通

```
% make rs
```

この処理時間は, スクリプト 2.4 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 [11].

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 [12].

2.2.4 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 [13].

■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 2.6 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 2.6: Makfile: 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
```

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

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

[13] 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.

```
6       mysql -u root -p********* < ./loadtoMySQL-uncons.sql
7       date > end-MSDB.txt
```

スクリプト 2.6 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 2.8 に与える.

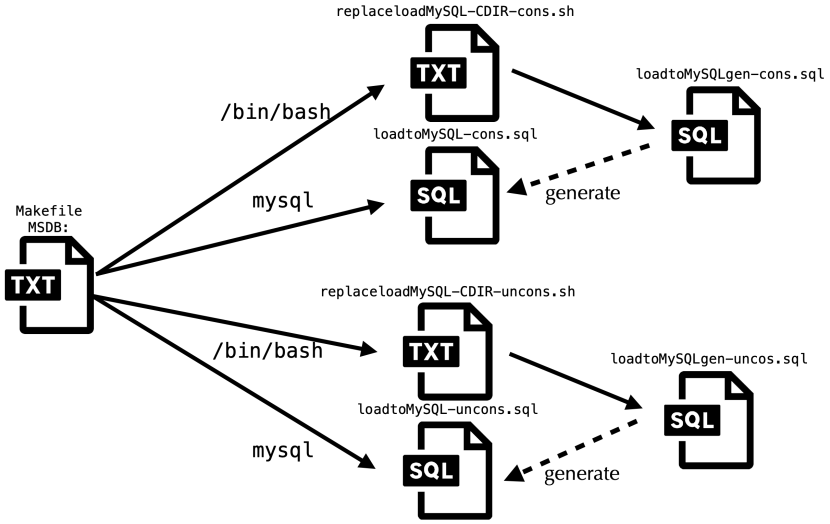


図 2.8: ターゲット MSDB の実行にともなうスクリプトファイルの流れ

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 2.7) 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-cons.sql (スクリプト 2.8) 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 2.8 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 2.8 参照).

スクリプト 2.7: 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 2.8: 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 2.6) 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 MSDB (スクリプト 2.6) 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 2.9) 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 2.10) 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 2.10 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 2.8 参照).

スクリプト 2.9: 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 2.10: 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 2.6) 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 2.9 に与える.

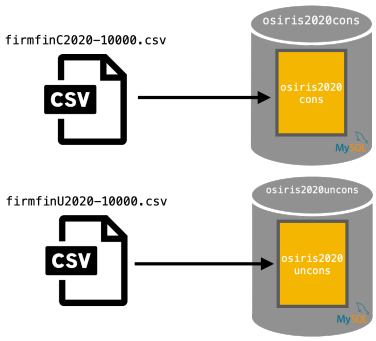


図 2.9: MySQL よる Osiris データにもとづくデータベース osiris2020cons, osiris2020uncons の構築

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 2.10 に与える.

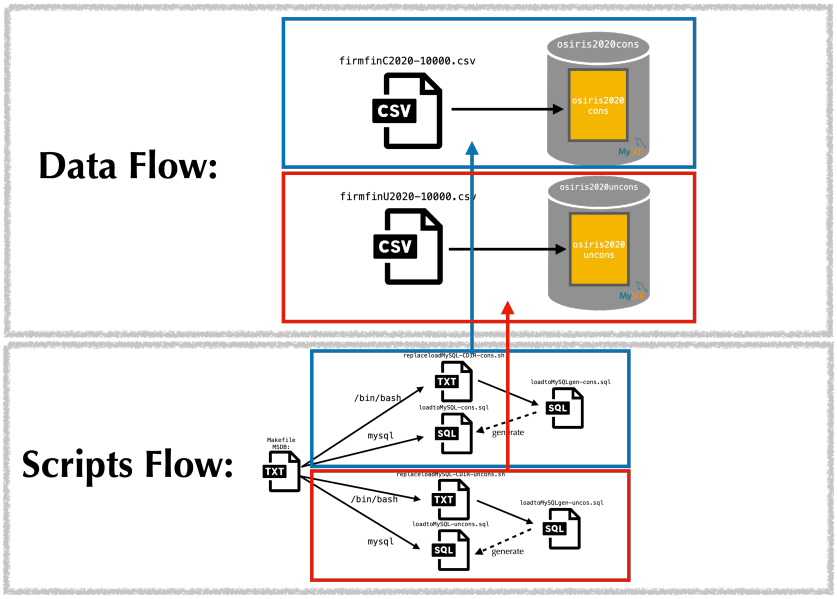


図 2.10: Osiris データに関するデータベース構築のためのターゲット MSDB の実行にともなうシェルスクリプトとデータに関するファイルの流れ

図 2.9, 2.10 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 2.1 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 2.6 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 [14].

■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 2.11: 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 2.6) と, Ubuntu 用のもの (スクリプト 2.11) 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 2.1 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 2.11 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.

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 [15].

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.

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

■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 2.12 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 2.12: 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 2.12 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化したものを図 2.11 に与える.

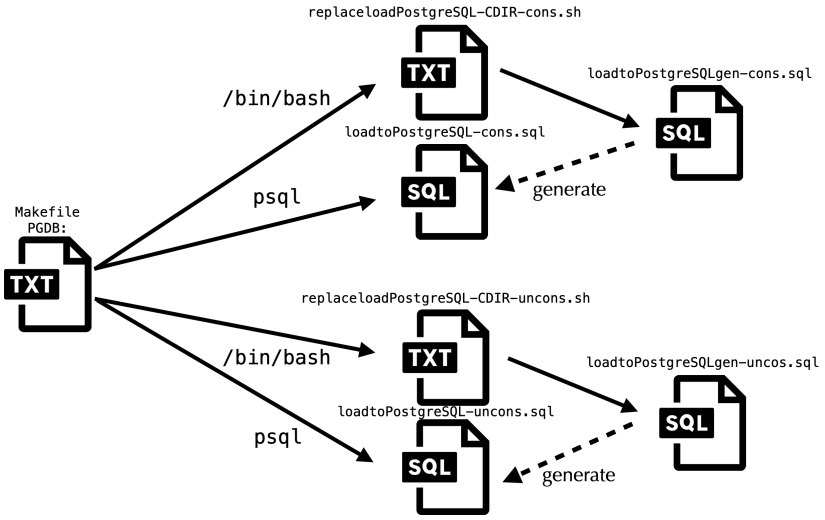


図 2.11: ターゲット PGDB の実行にともなうスクリプトファイルの流れ

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 2.13) 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 2.14) 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 2.14 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 2.11 参照).

スクリプト 2.13: 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 2.14: 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 2.12) 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 2.12) 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 2.15) 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 2.16) 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 2.16 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 2.11 参照).

スクリプト 2.15: 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 2.16: 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 2.12) 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 2.12 に与える.

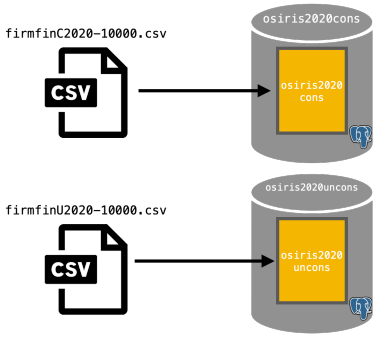


図 2.12: PostgreSQL よる Osiris データにもとづくデータベース osiris2020cons, osiris2020unccons の構築

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 2.13 に与える.

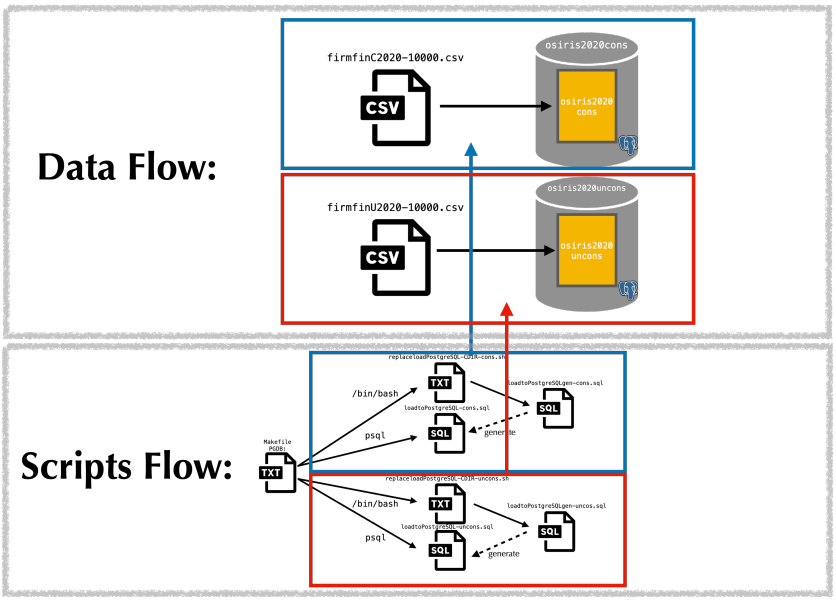


図 2.13: Osiris データに関するデータベース構築のためのターゲット PGDB の実行にともなうシェルスクリプトとデータに関するファイルの流れ

図 2.12, 2.13 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 2.1 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 2.12 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 [16].

■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 2.17: 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 2.12) と, Ubuntu 用のもの (スクリプト 2.17) 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 2.1 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 2.17 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 [17].

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

第 3 章

Orbis データ

本研究では, ビューロー・ヴァン・ダイクのデータベース 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. なお, ここで利用する Orbis に関するデータの仕様の詳細は, 第 10 章を参照されたい.

3.1 データベースとデータセット

ここでは, データベース Orbis から, 全世界の上場・非上場企業 (金融機関除く) [1] を対象に国際比較可能な統一のフォームで納められている財務情報から, 以下のように最長 10 年分抽出したデータセットを利用する:

(1) 金融・保険会社を除く上場及び非上場企業の連結財務諸表 (Consolidated) を優先的に抽出した 26,353,934 社のデータ (以後, 「連結ベース」と略す)
(2) 金融・保険会社を除く上場及び非上場企業の非連結財務諸表 (単体: Unconsolidated) を優先的に抽出した 26,352,382 社のデータ (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 5 GB 程度の 27 個の TSV ファイル (2 セット) からなり, 表 3.1, 表 3.2 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 3.1: 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	0rbis	上場・非上場	連結優先	10 年	85
DS-Orbis-U-2019	2019 年 12 月	0rbis	上場・非上場	非連結優先	10 年	85

表 3.2: 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

[1] 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.

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 [2]. 環境でデータベース管理システム PostgreSQL と **PG-Strom** [3] 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 3.3 のようなものである.

表 3.3: 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

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.

3.2 Orbis データによるデータベースの構築

3.2.1 Orbis データのデータベース構築手順

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 3.1 (macOS 環境と Ubuntu 環境で共通) を参照されたい [4].

```
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 3.1: 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

[2] 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 e-Words `https://e-words.jp/w/GPGPU.html` 参照)

[3] `https://heterodb.github.io/pg-strom/ja/`

[4] 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.

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.

3.2.2 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 140 GB 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 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 [5]. 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 [6].

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 3.1 参照).

スクリプト 3.1: Makfile: ターゲット rs

```
1  rs:
2          date > start-rs.txt
3          Rscript orbis2019-rs1per.R
4          date > end-rs.txt
5          date > start-load.txt
6          psql postgres < ./loadfirmIDs1p.sql
7          date > end-load.txt
8          date > start-dump-csv.txt
9          Rscript dumpfirmIDs1p-csv.R
10         date > end-dump-csv.txt
11         date > start-tar-csv.txt
12         /bin/bash script-tar-csv.sh
13         date > end-tar-csv.txt
```

まず, スクリプト 3.1 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 3.2 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:

[5] **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.

[6] 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.

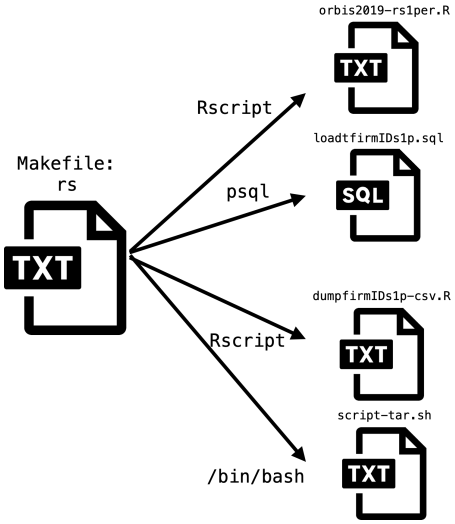


図 3.2: Orbis データファイルのランダムサンプリングに関するシェルスクリプトの流れ

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 3.3 に与える.

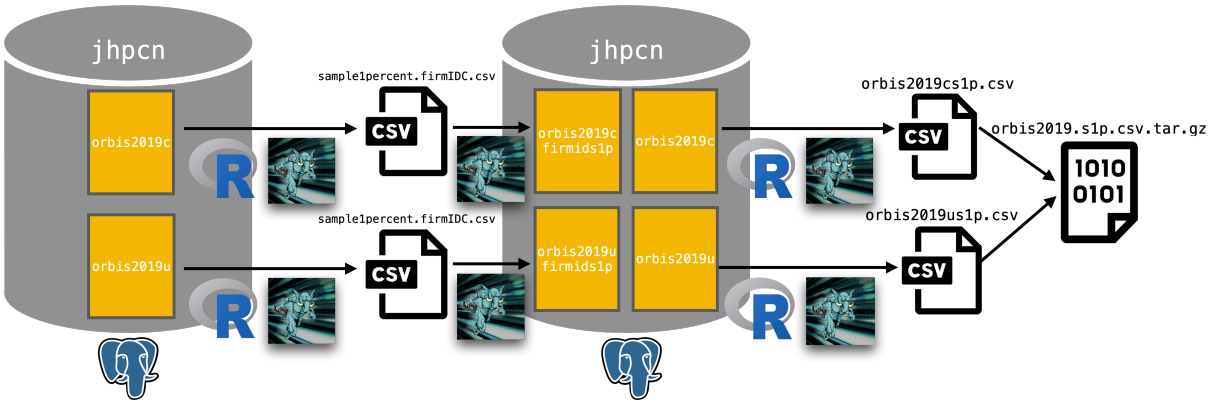


図 3.3: FENNEL 環境のもとでの Orbis データのランダムサンプリングにともなうデータファイルの流れ

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 3.1) 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 3.2 を指定することによって行われる.

スクリプト 3.2: `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 3.2 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 sample1percent.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 3.1) 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 3.3 を指定することによって行われる.

スクリプト 3.3: loadfirmIDs1p.sql

```
1  \c jhpcn
2  DROP TABLE IF EXISTS orbis2019cfirmids1p;
3  DROP TABLE IF EXISTS orbis2019ufirmids1p;
4  CREATE TABLE orbis2019cfirmids1p (
5  firmid VARCHAR(300)
6  );
7  CREATE TABLE orbis2019ufirmids1p (
8  firmid VARCHAR(300)
9  );
10 \COPY public.orbis2019cfirmids1p FROM '~/Orbis2019Sampling/sample1percent.
       firmIDC.csv' (format csv, header true);
11 \COPY public.orbis2019ufirmids1p FROM '~/Orbis2019Sampling/sample1percent.
       firmIDU.csv' (format csv, header true);
```

スクリプト 3.3 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 orbis2019ufirmids1p 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 3.1) 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 3.4 を指定することによって行われる.

スクリプト 3.4: 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 3.4 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 3.1) 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 3.5 を指定することによって行われる.

スクリプト 3.5: CSV ファイルを圧縮するための シェルスクリプト

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

スクリプト 3.5 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 3.4 に与える.

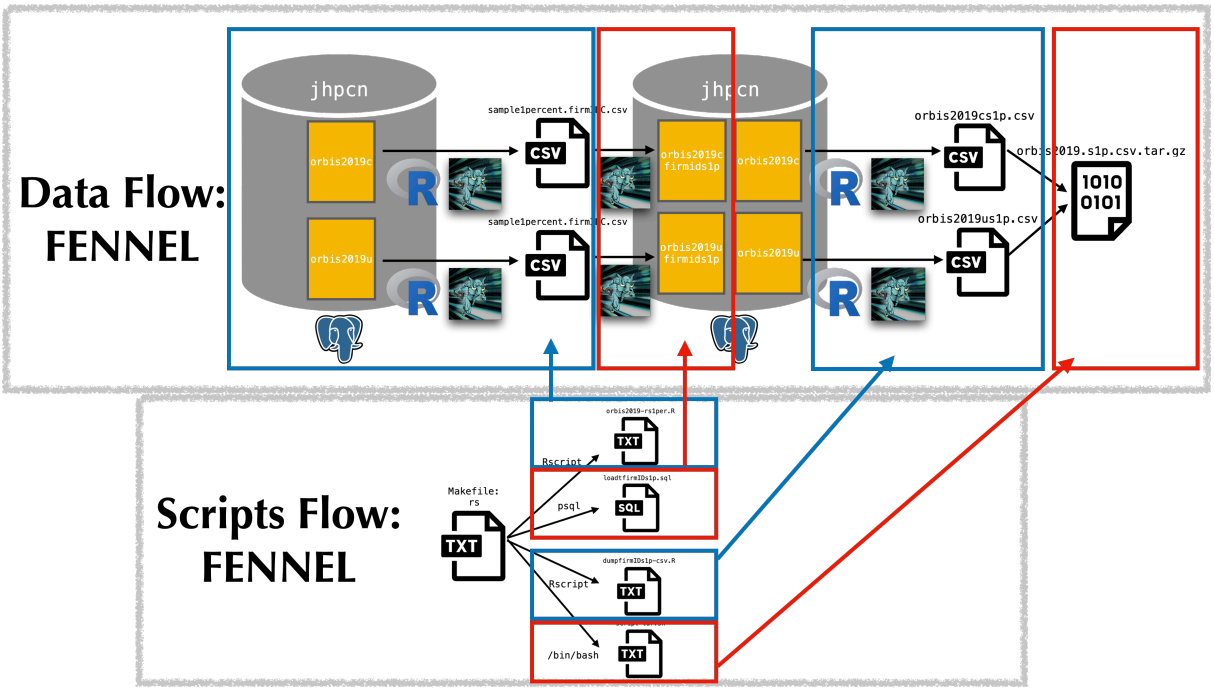


図 3.4: Orbis データのランダムサンプリングに利用されるシェルス・クリプト・ファイルとデータファイルの処理の対応関係: FENNEL 環境を利用

なお, この処理は, FENNEL 環境におけるディレクトリ Orbis2019Sampling (図 ?? 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 3.1 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 [7].

次に, 手順 (Or-RS2) の処理に要した時間は以下である:

[7] 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 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 [8].

また, 手順 (Or-RS3) 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 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 [9].

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 3.5 参照).

[8] この結果も **PG-Strom** と FENNEL 環境の恩恵といえる.

[9] 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.

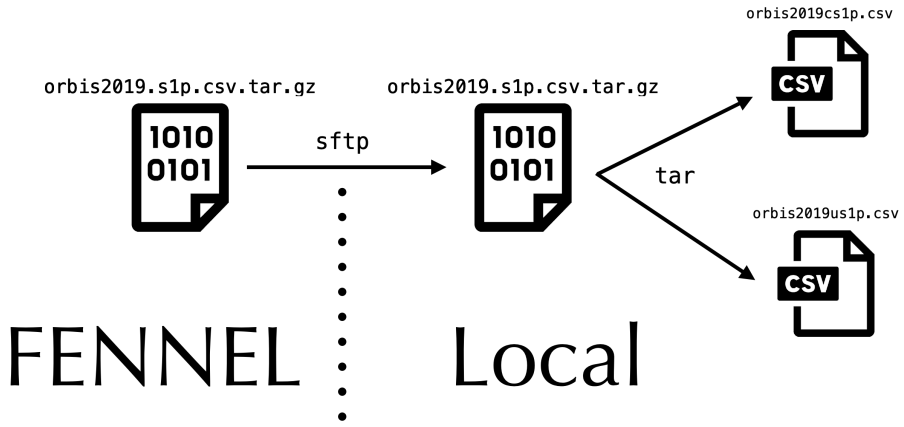


図 3.5: FENNEL 環境からローカル環境へファイルの転送と展開

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 3.1 参照).

3.2.3 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 [10]. 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 3.6 参照).

スクリプト 3.6: Makfile: 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 3.6 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 3.7) が指定されている.

スクリプト 3.7: 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 3.8) に定義しておき, sed [11] を利用して置換している.

スクリプト 3.8: sedbs

[10] 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.

[11] 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.

```
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 3.6 に与える.

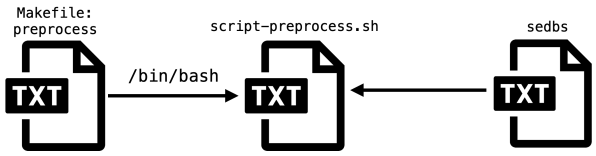


図 3.6: Orbis データファイルの前処理に関するシェルスクリプトの流れ

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 3.7 に与える.

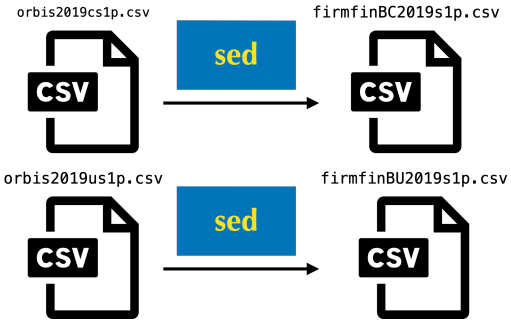


図 3.7: Orbis データの前処理におけるデータファイルの流れ

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 3.8 に与える.

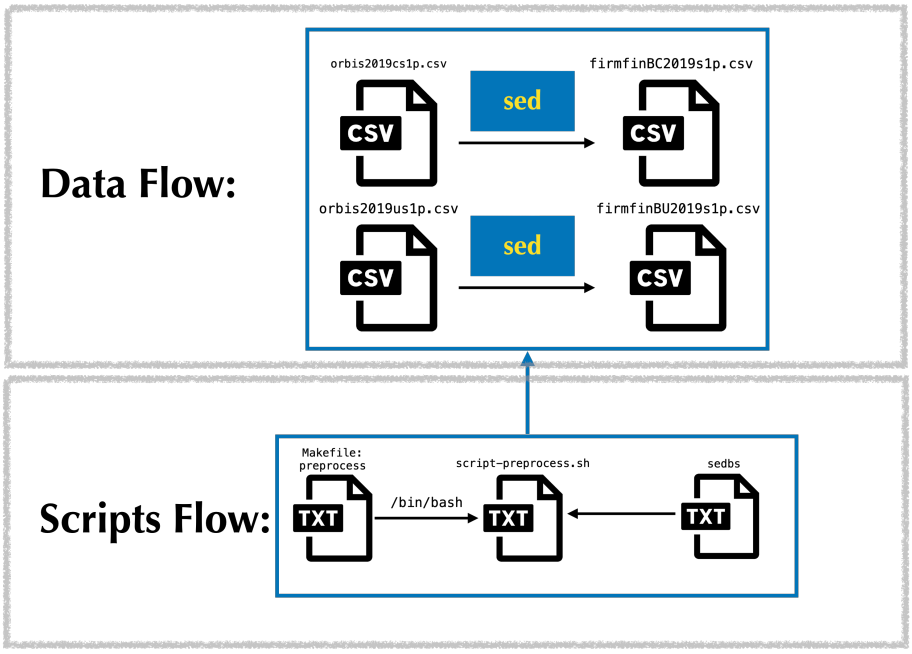


図 3.8: Orbis データの前処理におけるシェルスクリプトとデータに関するファイルの流れ

なお, 前処理は, ディレクトリ DBMaking (図 3.1 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 (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 3.6 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 [12].

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 [13].

3.2.4 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.

■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 3.9 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 3.9: Makfile: 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
```

[12] この結果は, iMac Pro 2018 (macOS Big Sur) で計測したものである.
[13] 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 3.9 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 3.9 に与える.

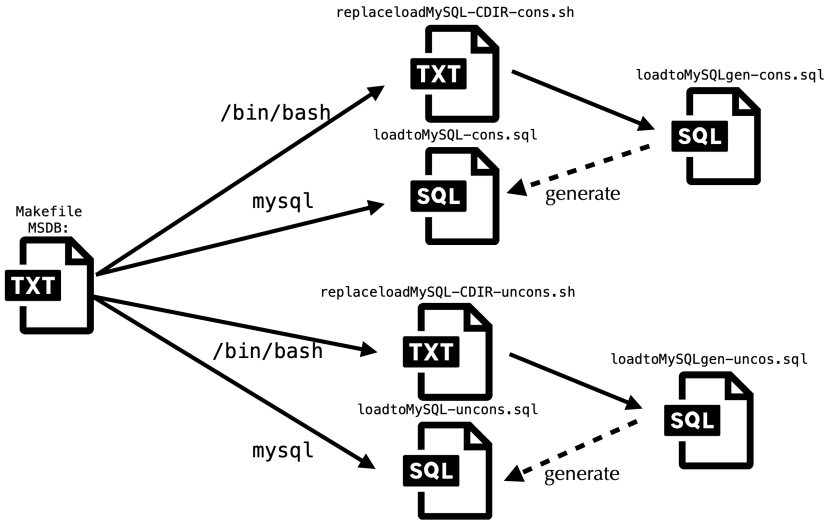


図 3.9: ターゲット MSDB の実行にともなうスクリプトファイルの流れ

まず, 手順 (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-cons.sh の内容 (スクリプト 3.10) 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[14] スクリプトファイル loadtoMySQLgen-cons.sql (スクリプト 3.11) 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 3.11 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 3.9 参照).

スクリプト 3.10: 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 3.11: 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 :
```

[14] 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.

```
year VARCHAR(4)
);
LOAD DATA LOCAL INFILE 'CDIR/firmfinBC2019s1p.csv'
INTO TABLE orbis2019cons
FIELDS TERMINATED BY ','
ENCLOSED BY '"'
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 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 3.9) 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 3.9) 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 3.12) 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 3.13) 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 3.13 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 3.9 参照).

スクリプト 3.12: replaceloadMySLQ-CDIR-uncons.sh

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

スクリプト 3.13: loadtoMySQLgen-uncons.sql(一部抜粋)

```
DROP DATABASE IF EXISTS orbis2019uncons;
CREATE DATABASE orbis2019uncons;
USE orbis2019uncons;
DROP TABLE IF EXISTS orbis2019uncons;
CREATE TABLE orbis2019uncons (
firm VARCHAR(350),
:
: (中略)
:
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 3.9) 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 3.10 に与える.

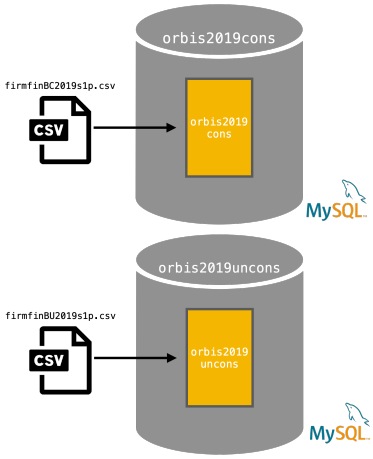


図 3.10: MySQL よる Orbis データにもとづくデータベース orbis2019cons, orbis2019uncons の構築

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 3.11 に与える.

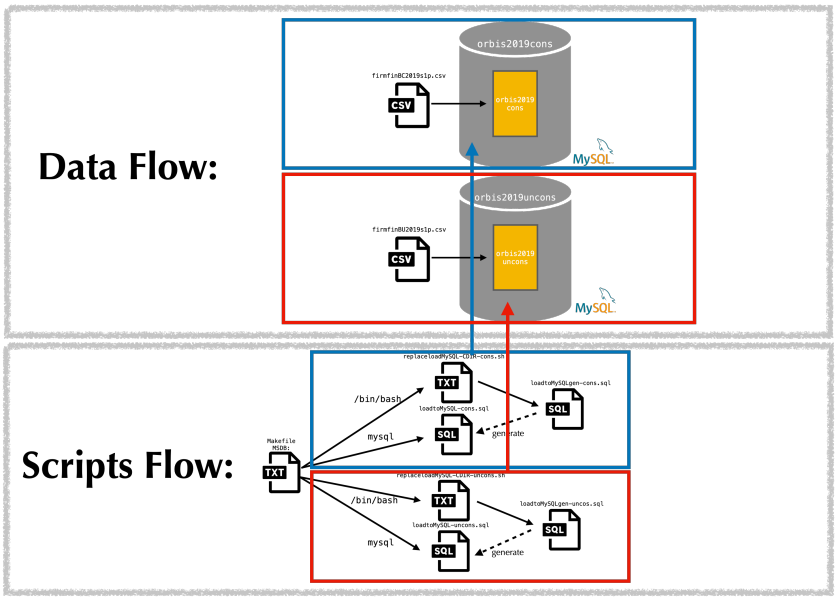


図 3.11: Orbis データに関するデータベース構築のためのターゲット MSDB の実行にともなうシェルスクリプトとデータに関するファイルの流れ

図 3.10, 3.11 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.

ターゲット MSDB はディレクトリ DBMaking (図 3.1 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 3.9 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 [15].

■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 3.14: Makefile: 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 3.9) と, Ubuntu 用のもの (スクリプト 3.14) 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 3.1 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 3.14 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
```

この結果から, 3 分 59 秒であることがわかる [16].

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.

[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.

■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 3.15 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 3.15: 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 3.15 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 3.12 に与える.

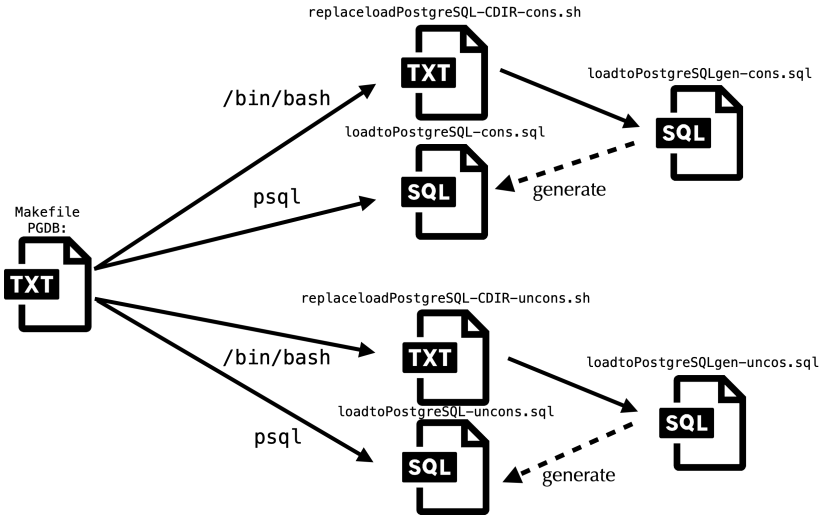


図 3.12: ターゲット PGDB の実行にともなうスクリプトファイルの流れ

まず, 手順 (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 3.16) 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 3.17) 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 3.17 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 loadtoPostgreSQLgen-cons.sql から生成される (図 3.12 参照).

スクリプト 3.16: 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 3.17: 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 3.15) 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 3.15) 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 3.18) 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 3.19) 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 3.19 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 3.12 参照).

スクリプト 3.18: 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 3.19: 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 3.15) 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 3.13 に与える.

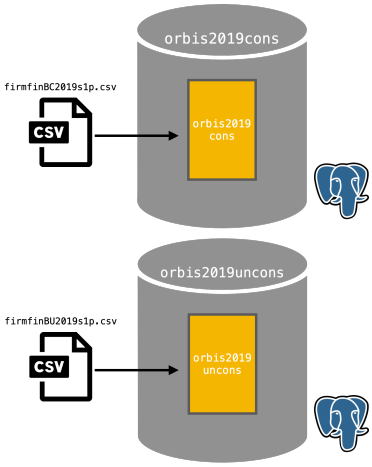


図 3.13: PostgreSQL よる Orbis データにもとづくデータベース orbis2019cons, orbis2019uncons の構築

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 3.14 に与える.

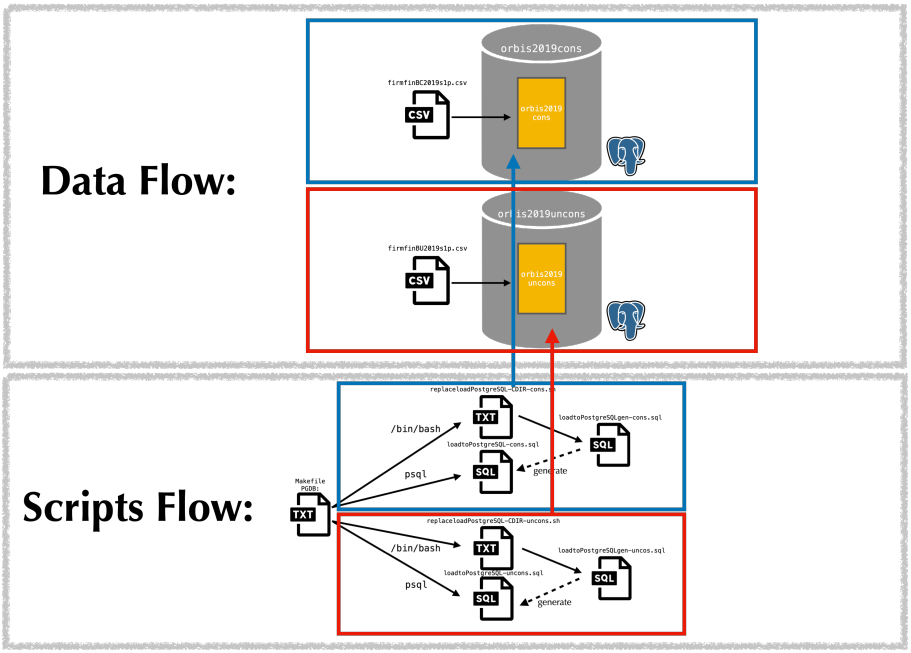


図 3.14: Orbis データに関するデータベース構築のためのターゲット PGDB の実行にともなうシェルスクリプトとデータに関するファイルの流れ

図 3.13, 3.14 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.

ターゲット PGDB はディレクトリ DBMaking (図 3.1 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 3.15 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 [17].

■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 3.20: 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 3.15) と, Ubuntu 用のもの (スクリプト 3.20) 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 3.1 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 3.20 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.

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

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 [18].

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

第 4 章

財務データ抽出システム SKWAD の構築

4.1　システム名称

従来のシステム名称 “KGUSBADES” (Kwansei Gakuin University, School of Business Administration, Data Extraction System の略) は, 発音が難しいかったり, 綴りが長いため覚えにくいため改名することを検討した. 様々な検討の結果として, 新名称を “SKWAD” (スクワッド) とした. これは, 変則的ではあるが, School of business administration, KWAnsei gakuin university, Data extraction system の略である [1].

4.2　インターフェースデザイン

KGUSBADES によるサービスが 2010 年に開始されて以来, 2012 年にサービスの種類を増加させることに伴って, Web インターフェースのデザインを改良したが, 今回の再構築では, システム名称の変更と抽出できるデータの種類をさらに拡充するため, 図 4.1 のように Web インターフェースを改良した. [2].

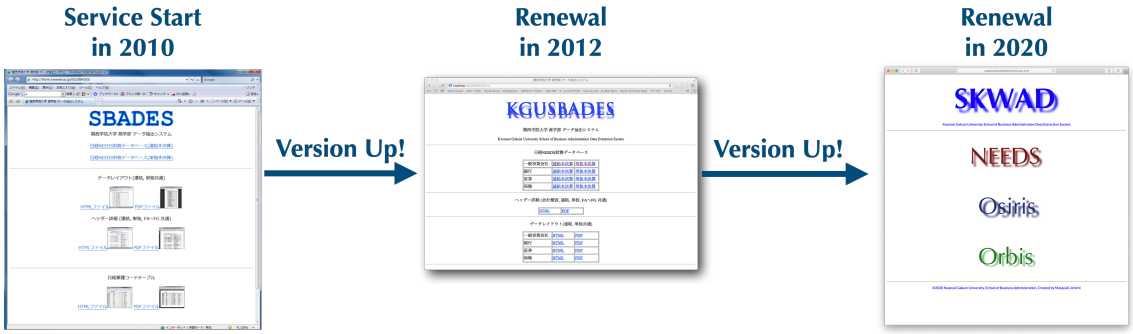


図 4.1: Web インターフェースの改良

本システムは, 不特定多数に対して公開するわけでなく, 学内限定であり, かつその利用に関しては, 教育・研究に利用されるものであることから, 極力余計な情報を提供することなく, データソース名をアイコン化することによってシンプルなものとした. 今回リニューアルが予定されているシステムのトップページは, (システムのロゴアイコン以外では) NEEDS, Osiris, Orbis という財務データの提供元が命名したデータベース名

[1] SKWAD (スクワッド) は, 精鋭 (特殊) 部隊という意味の単語 squad の発音 (skwɑ(:)d) と同じであり, 「選りすぐりの財務データ」の一団を表すという理解もできる.

[2] 本システムを公開・運用・メンテナンスする中で, 筆者が Web インターフェースをデザインする際, 重視してきたコンセプトは「シンプル」という一点のみである.

を表すアイコンが並んでいるだけのものであり, ユーザはファーストアクセス時には戸惑うかもしれないが, 簡単な説明を与えれば, それ以降は混乱はないものと思われる.

これらのアイコンは, 以下のようなデータを抽出するためのものである:

NEEDS: 日経メディアマーケティング株式会社から販売されている日本における一般事業会社 (1.6 万社超) の財務データ (NEEDS 企業財務データ [3])

Osiris: ビューロー・ヴァン・ダイクから販売されている世界の全上場会社 (金融機関を除く 9 万社超) の財務データ

Orbis: ビューロー・ヴァン・ダイクから販売されている世界の (財務データが収集されている) 全会社 (金融機関を除く 2,600 万社超) の財務データ

4.3　財務データ抽出システム SKWAD の仕様

ここでは, 財務データ抽出システム SKWAD の仕様について解説する (トップページは図 4.2 を参照). なお, セキュリティに関する観点から, システムのソースコードを部分的に割愛する.

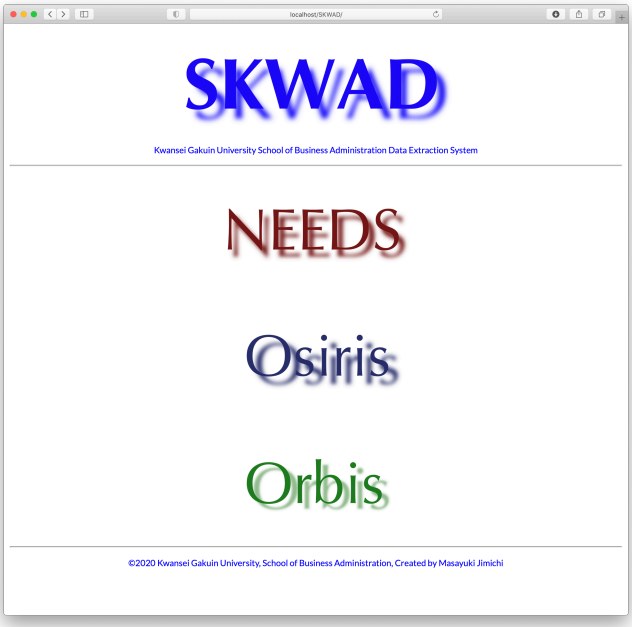


図 4.2: データ抽出システム SKWAD のトップページ

このシステムを構成するファイルのディレクトリ構成 (一部抜粋) については図 4.3 を参照されたい.

[3] ここで提供される NEEDS 企業財務データは, 2020 年 6 月期決算までの財務情報が収録されており, 東京証券取引所に上場している企業に関しては旧区分 (市場第一部, 市場第二部, マザーズ, JASDAQ(スタンダード, グロース)) であることに注意されたい.

```
SKWAD
├──── NEEDS
│       ├──── 2020
│       └──── index.html
├──── index.html
└──── logo
        ├──── NEEDS.png
        ├──── Orbis.png
        ├──── Osiris.png
        └──── SKWAD.png
```

図 4.3: データ抽出システム SKWAD を構成するファイルのディレクトリ構成 (一部抜粋)

ここで, システムのトップページの実体は, 図 4.3 におけるディレクトリ SKWAD [4] のサブディレクトリにある HTML ファイル index.html であり, このページに表示されるロゴの画像ファイルが logo ディレクトリに格納されている. このトップページの NEEDS ロゴ (アイコン NEEDS) をクリックすることによって, NEEDS 企業財務データのデータベースからデータを抽出するためのトップページ (図 4.4) に移動することができる.

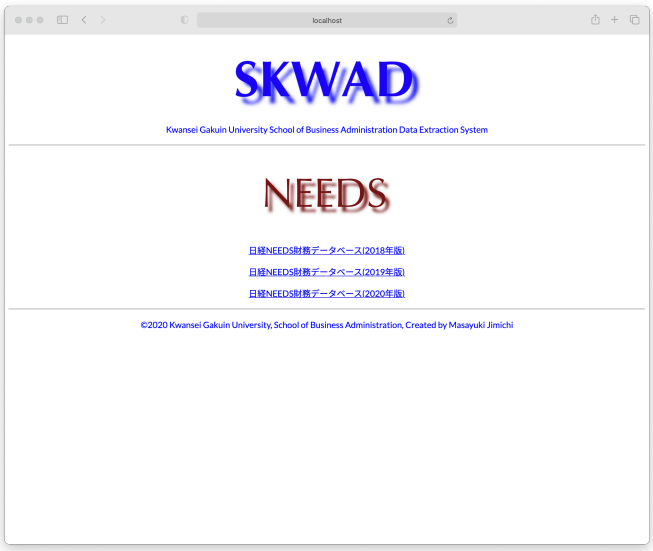


図 4.4: NEEDS 企業財務データを抽出するためのトップページ

なお, このページの実体は, 図 4.3 におけるサブディレクトリ NEEDS にある HTML ファイル index.html である. ここで, サブディレクトリ NEEDS のさらに下のディレクトリ 2020 の構造を図 4.5 に与える.

[4] 実際のシステム開発では, データが毎年バージョンアップすることを考慮して, SKWAD ディレクトリには, 開発年度をディレクトリ名に入れており (たとえば, SKWAD2020), そこから, SKWAD へ ln コマンドによって, リンクを張るような仕様としている. このことによって, 年度が変わってもユーザは同一の URL でシステムにアクセスできるようになる.

```
NEEDS
├──── 2020
│   ├──── codes
│   │   ├──── NIKKEIcode.fld
│   │   ├──── NIKKEIcode.htm
│   │   ├──── NIKKEIcode.pdf
│   │   ├──── datainformation.fld
│   │   └──── datainformation.htm
│   ├──── cons
│   │   ├──── NEEDS-general-cons-2020.html
│   │   └──── SQLtreat-general-cons-2020.php
│   ├──── index.html
│   └──── uncons
│       ├──── NEEDS-general-uncons-2020.html
│       └──── SQLtreat-general-uncons-2020.php
└──── index.html
```

図 4.5: NEEDS 企業財務データ (一般事業会社: 2020 年版) のデータベースからデータを抽出するシステムのディレクトリ構成 (一部抜粋)

NEEDS 企業財務データを抽出するためのページ (図 4.4) から, さらにリンク **日経 NEEDS 財務データベース (2020 年版)** をクリックすることによって, NEEDS 企業財務データ (一般事業会社) 2020 年版のデータベースからデータを抽出するページに移動することができる (図 4.6 参照). このページの実体は, 図 4.5 におけるディレクトリ 2020 の直下にある `index.html` ファイルである.

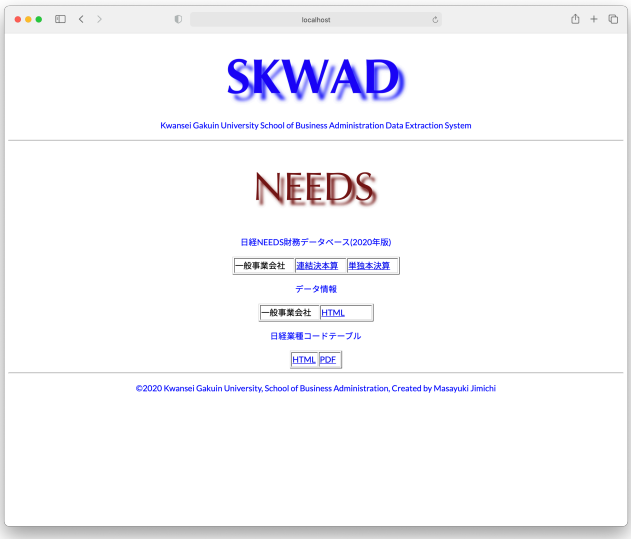


図 4.6: NEEDS 企業財務データ (一般事業会社) 2020 年版のデータベースからデータを抽出するためのトップページ

このページにおいて, リンク **連結本決算** とリンク **単独本決算** をそれぞれ選択することによって, NEEDS 企業財務データ (一般事業会社) 2020 年版のデータベースからそれぞれの財務データを抽出するページへ移動できる. これらのリンクの実体は, 図 4.5 のディレクトリ 2020 におけるサブディレクトリ `con` の `NEEDS-general-cons-2020.html` と, サブディレクトリ `uncon` の `NEEDS-general-uncons-2020.html` である.

また, 「データ情報」の一般事業会社のリンク **HTML** を選択することによって, データの仕様に関するページへ移動することができ, 「日経業種コードテーブル」のリンク **HTML** と **PDF** をそれぞれ選択することによって, 日経業種コードに関する情報を HTML 形式または PDF ファイルとして参照することができる. これらのリンクの実体は, 図 4.5 におけるディレクトリ 2020 のサブディレクトリ `codes` に格納されている.

NEEDS 企業財務データ (一般事業会社) 2020 年版のデータベースからデータを抽出するためのページ (図 4.6) のリンク **連結本決算** を選択することによって, 「日経 NEEDS 財務データ抽出システム (一般事業会社: 2020 年版, 連結本決算)」のページに移動することができる (図 4.7 参照).

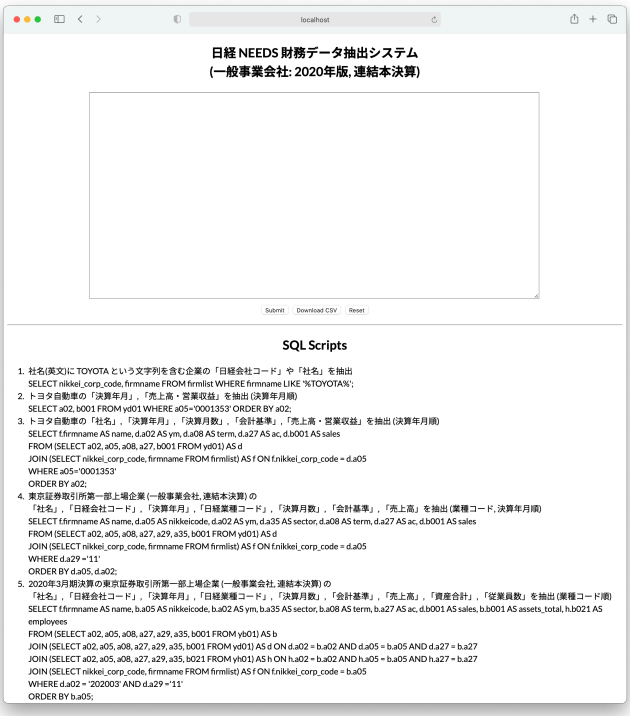


図 4.7: 「日経 NEEDS 財務データ抽出システム (一般事業会社: 2020 年版, 連結本決算)」のページ

?? 節でも説明したが, このシステムにおいて, `スクリプト入力ボックス` に SQL 問合せを入力し, `Submit` ボタンまたは `Download CSV` ボタンをクリックすることによって, HTML 形式, または CSV 形式でデータを抽出することができる.

このシステムの概要は図 4.8 に与えられる.

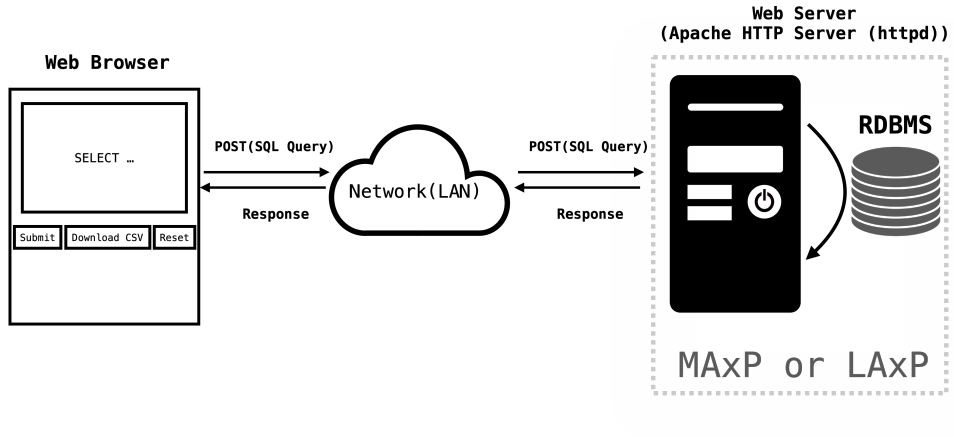


図 4.8: データ抽出システムの概要

図 4.8 が示すネットワーク経由でデータを抽出するための処理の流れは以下のようなものである:

データ抽出の流れ

(F-1) Web ブラウザ (クライアント) を利用して, SQL 問合せを送信
(F-2) サーバが命令を処理, 返信
(F-3) 抽出結果を受け取り Web ブラウザに表示, またはダウンロード

この処理の流れを実現するための仕様は以下のようなものである:

データ抽出システムの仕様

(SPEC-1) HTML の FORM タグを使って SQL 問合せをデータベースサーバへ送信
(SPEC-2) PHP スクリプトを使って, 受け取った SQL 問合せ (SQL スクリプト) にもとづきデータを抽出した結果を HTML ファイル, または CSV ファイルとして出力し, 返信
(SPEC-3) クライアントが結果の HTML ファイルを受信・表示, または CSV ファイルをダウンロード

ここで仕様 (SPEC-1) は, ディレクトリ 2020 (図 4.5 参照) のサブディレクトリ con における NEEDS-general-cons-2020.html ファイルが, その役割を果たし, 実際のコードはスクリプト 4.1 で与えられる.

スクリプト 4.1: NEEDS-general-cons-2020.html (一部抜粋)

```
1  <FORM ACTION='SQLtreat-general-cons-2020.php' METHOD='post'>
2    <div align='center'>
3        <p>
4          <TEXTAREA NAME='SQLcode' ROWS='30' COLS='120'></TEXTAREA>
5          <BR>
6          <INPUT TYPE='submit' VALUE='Submit' NAME = 'HTML'>
7          <INPUT TYPE='submit' VALUE='Download␣CSV' NAME = 'CSV'>
8          <INPUT TYPE='reset' VALUE='Reset'>
9        </p>
10   </div>
11 </FORM>
```

このスクリプトの役割を以下に与える:

1 行目: FORM タグを利用し, 属性 METHOD='post' と属性 ACTION='SQLtreat-general-cons-2020.php' でサーバ上の PHP ファイル SQLtreat-general-cons-2020.php へ入力されたスクリプトを送信する.

4 行目: TEXTAREA タグによって SQL 問合せを入力するためのボックス (スクロールテキストボックス) を作成し, 入力欄の名前を属性 NAME に SQLcode という文字列を与えており (NAME='SQLcode'), 属性 ROWS と COSL を使ってボックスの大きさを 30 行, 120 列と指定する (ROWS='30' COLS='120').

6 行目: INPUT タグの属性 TYPE='submit' を利用して, サーバへ提出するための Submit ボタンを作成する. この入力については, NAME = 'HTML' と指定することによって, HTML 形式で出力するための名称を設定する.

7 行目: INPUT タグの属性 TYPE='submit' を利用して, サーバへ提出するための Download CSV ボタンを作成する. この入力については, NAME = 'CSV' と指定することによって, CSV 形式で出力するための名称を設定する.

8 行目: INPUT タグの属性 TYPE='reset' を利用して, 入力したスリプトを消去するための Reset ボタンを作成する.

次に, 仕様 (SPEC-2) は, 図 4.5 で与えられるディレクトリ 2020 のサブディレクトリ cons の PHP ファイル SQLtreat-general-cons-2020.php が, その役割を果たし, 実際のコードはスクリプト 4.2 で与えられる.

スクリプト 4.2: SQLtreat-general-cons-2020.php (一部抜粋)

```
1  <?php
2  if(isset($_POST['HTML'])) {
3        $pid = exec('echo␣$$');
4        $sqlquery = "/tmp/sqlquery-general-2020".$pid.".sql";
5        $NEEDShtml = "/tmp/NEEDS2020cons".$pid.".html";
6        $fp = fopen($sqlquery, 'w');
7        fwrite($fp, $_POST['SQLcode']);
8        fclose($fp);
9        system("cat␣$sqlquery␣|␣PGPASSWORD=********␣psql␣-U␣*******␣-d␣
             needs2020cons␣-H␣>␣$NEEDShtml");
10       include($NEEDShtml);
11       exit;}
12 elseif(isset($_POST['CSV'])) {
13       $pid = exec('echo␣$$');
14       $sqlquery = "/tmp/sqlquery-general-2020".$pid.".sql";
15       $NEEDScsv = "/tmp/NEEDS2020cons".$pid.".csv";
16       $fp = fopen($sqlquery, 'w');
17       fwrite($fp, $_POST['SQLcode']);
18       fclose($fp);
19       system("cat␣$sqlquery␣|␣PGPASSWORD=********␣psql␣-U␣*******␣-d␣
             needs2020cons␣--csv␣>␣$NEEDScsv");
20         header("Content-Type:␣application/octet-stream");
21         header("Content-Disposition:␣attachment;␣filename=\"NEEDS2020cons.
             csv\"");
22         header("Content-Length:␣" . filesize($NEEDScsv));
23         readfile($NEEDScsv);
24       exit;}
25 ?>
```

このスクリプトの役割は以下のようなものである:

Submit ボタンが選択されたときへの対応 (2 行目〜11 行目):

2 行目: Submit ボタンが選択 (スクリプト 4.1 の 6 行目参照) されたかどうかを $_POST['HTML'] がセッ

トされたかどうかで判断する．TRUE (真) の場合は, 3 行目以降が実行される．

3 行目: シェルのプロセス ID を取得 (exec('echo $$')) し, 変数 $pid に代入する．

4 行目: SQL 問合せを格納するスクリプトファイル名をシェルのプロセス ID 付きで定義し, 変数 $sqlquery へ代入する．

5 行目: 抽出結果を保存する HTML ファイル名をシェルのプロセス ID 付きで定義し, 変数 $NEEDShtml へ代入する．

6 行目: PHP 関数 fopen を利用して, 変数 $sqlquery で定義されるファイルを書き込みモード 'w' で開く．

7 行目: スクリプト 4.1 の 4 行目で定義されたボックスの入力欄の名前 'SQLcode' から, その入力内容を変数 $sqlquery で定義された SQL スクリプトファイルへ変数 $fp を通じて入力内容を出力する．

8 行目: PHP 関数 fclose を利用して SQL スクリプトファイルを閉じる．

9 行目: 変数 $sqlquery で定義された SQL スクリプトファイルの内容を cat で展開後, パイプ機能 (|) を通じて, psql に引き渡し, オプション -d でデータベース needs2020cons を指定して, データを抽出後, オプション -H を指定することによって HTML 形式で変数 $NEEDShtml で指定されたファイルへ出力する．

10 行目: PHP 関数 include で結果として出力された HTML ファイルを読み込む．

Download CSV ボタンが選択されたときへの対応 (12 行目〜24 行目)

12 行目: Submit ボタンが選択 (スクリプト 4.1 の 7 行目参照) されたかどうかを $_POST['CSV'] がセットされたかどうかで判断する．TRUE (真) の場合は, 13 行目以降が実行される．

13 行目: シェルのプロセス ID を取得 (exec('echo $$')) し, 変数 $pid に代入する．

14 行目: SQL 問合せを格納するスクリプトファイル名をシェルのプロセス ID 付きで定義し, 変数 $sqlquery へ代入する．

15 行目: 抽出結果を保存する CSV ファイル名をシェルのプロセス ID 付きで定義し, 変数 $NEEDScsv へ代入する．

16 行目: PHP 関数 fopen を利用して, 変数 $sqlquery で定義されるファイルを書き込みモード 'w' で開く．

17 行目: スクリプト 4.1 の 4 行目で定義されたボックスの入力欄の名前 'SQLcode' から, その入力内容を変数 $sqlquery で定義された SQL スクリプトファイルへ変数 $fp を通じて入力内容を出力する．

18 行目: PHP 関数 fclose を利用して SQL スクリプトファイルを閉じる．

19 行目: 変数 $sqlquery で定義された SQL スクリプトファイルの内容を cat で展開後, パイプ機能 (|) を通じて, psql に引き渡し, オプション -d でデータベース needs2020cons を指定して, データを抽出後, オプション -csv を指定することによって CSV 形式で変数 $NEEDScsv で指定されたファイルへ出力する．

20 行目: PHP 関数 header で "Content-Type: application/octet-stream" と指定することによって, ファイル形式に関係なくダウンロードを開始する．

21 行目: PHP 関数 header で "Content-Disposition: attachment; filename=\"NEEDS2020cons.csv\"" と指定することによって, ファイル名を NEEDS2020cons.csv とする．

22 行目: PHP 関数 header で "Content-Length: " . filesize($NEEDScsv) と指定することによって, ファイルサイズを取得し, ダウンロードの進捗を表示する．

23 行目: PHP 関数 readfile で CSV ファイルを出力する．

なお, ここで与えたスクリプトには, セキュリィティの観点から, PostgreSQL のユーザ名やパスワード, さらに対話型インターフェース `psql` のパスなどは明記していない. もし, 本稿を参考にする際には, これらの情報を適切に設定されたい.

第 II 部

利用編

第 5 章

財務データ抽出システム SKWAD の利用

5.1 NEEDS データの抽出と可視化

NEEDS 企業財務データ (一般事業会社, 連結本決算) を抽出するための手順は以下のようなものである:

(NE1) SKWAD のトップページにアセスセスする.
(NE2) NEEDS のロゴ (アイコン NEEDS) をクリックする.
(NE3) 移動したページのリンク 日経 NEEDS 財務データベース (2020 年版) をクリックする.
(NE4) 移動したページのリンク 連結本決算をクリックする.

以上の手順によって, NEEDS 企業財務データ (一般事業会社, 連結本決算) を抽出するためのページに移動することができる (図 5.1).

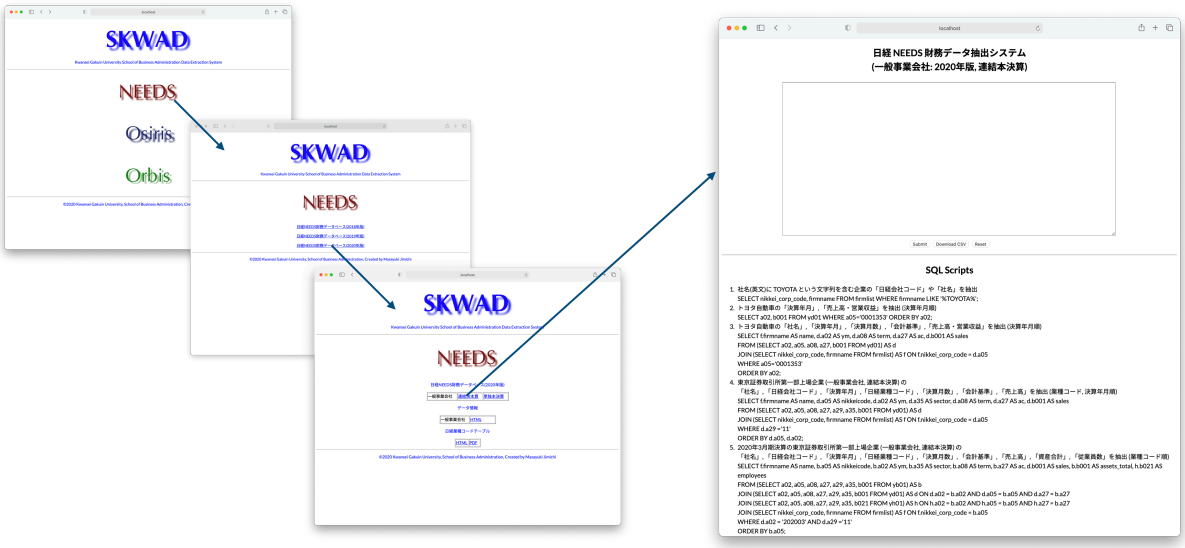


図 5.1: NEEDS 企業財務データ抽出システム (一般事業会社: 2020 年版, 連結本決算) へのリンク

このページの利用法を説明する. 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できる.

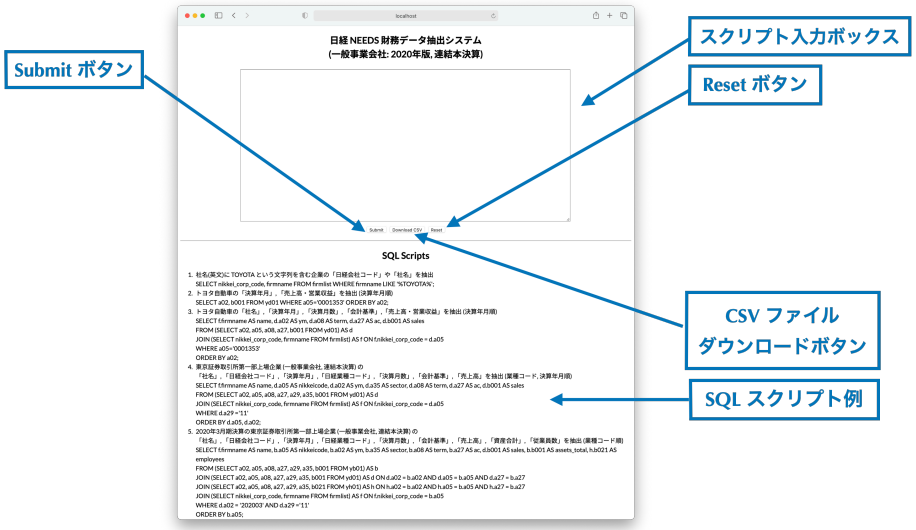


図 5.2: NEEDS 企業財務データ抽出システム (一般事業会社: 2020 年版, 連結本決算) のページ

なお, SQL のサンプルスクリプトも用意されている (図 5.2 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, 手軽にデータ抽出を試すことができる.

では, 以下に SQL 問合せの例を幾つか取り上げる.

5.1.1　トヨタ自動車の日経会社コードの検索

最初の例として与えられているトヨタ自動車の日経会社コードを検索するための操作を図 5.3 に与える.

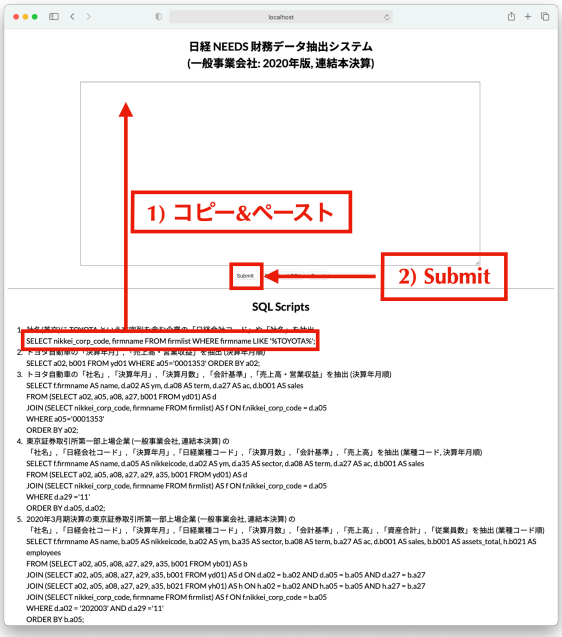


図 5.3: トヨタ自動車の日経会社コードの検索

この抽出に利用された SQL 問合せをスクリプト 5.1 に与える．テーブルと列名の意味については, 第 8 章か日本経済新聞社 (2020) を参照されたい．

スクリプト 5.1: 社名に 'TOYOTA' という文字列を含む企業の日経会社コードと社名を抽出

```
1 SELECT nikkei_corp_code, firmname
2       FROM firmlist
3             WHERE firmname LIKE '%TOYOTA%';
```

この SQL 問合せを以下に説明する．

1 行目: SELECT 句で日経会社コード (nikkei_corp_code) と社名 (firmname) の列を指定する．

2 行目: FROM 句で日経会社コード (nikkei_corp_code) や社名 (firmname) が納められているテーブル firmlist を指定する．

3 行目: WHERE 句で条件として英文表記の企業名 (firmname) の列において TOYOTA という文字列を含むような (LIKE) 行に限定する．

ここで, % は任意の文字列を表し, '(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．

5.1.2 トヨタ自動車の売上高の抽出

次に, 実際の企業の財務データの抽出例として, トヨタ自動車の売上高の推移を時系列プロットによって与える．実際の操作手順は以下のようなものである (図 5.4):

1. SQL のサンプルスクリプトの 2 番目を スクリプト入力ボックス にコピー・アンド・ペースト
2. Submit ボタンをクリック
3. 得られた抽出結果を, 全て選択し, Microsoft Excel (以下 Excel と略) にコピー・アンド・ペースト

4. セルを適切に選択し, 時系列プロット

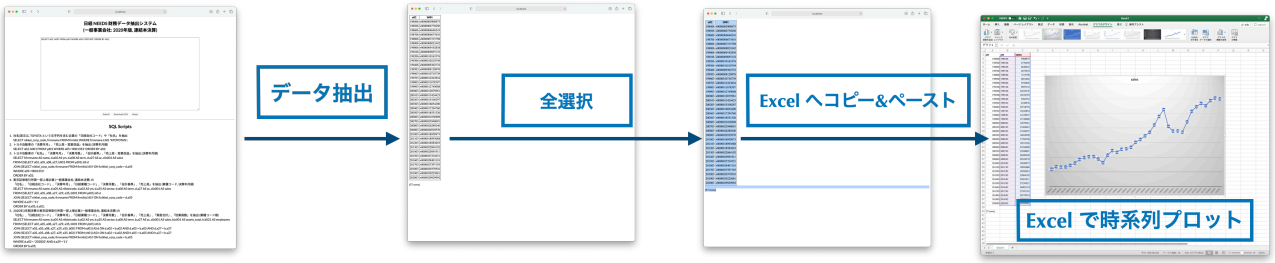


図 5.4: トヨタ自動車の売上高の推移 (時系列プロット) の作成工程

この抽出に利用された SQL 問合せは以下のようなものである:

スクリプト 5.2: トヨタ自動車の「決算年月」, 「売上高・営業収益」の抽出 (決算年月順)

```
1 SELECT a02, b001
2       FROM yd01
3             WHERE a05='0001353'
4                   ORDER BY a02;
```

この SQL 問合せを以下に説明する.

1 行目: SELECT 句で決算年月 (a02) と 売上高 (b001) の列を指定する.
2 行目: FROM 句で決算年月 (a02) と 売上高 (b001) がおさめられているテーブル yd01 を指定する.
3 行目: WHERE 句で列 a05 (日経会社コード) においてトヨタ自動車の日経会社コード 0001353 を含む行 (a05='0001353') を限定する.
4 行目: ORDER BY 句で決算年月 (a02) の順に並べ替える.

5.1.3 2020 年 3 月期決算の東京証券取引所第一部上場企業 (一般事業会社, 連結本決算) の財務データの抽出

さらに複雑な SQL 問合せの例として, 2020 年 3 月期決算の東京証券取引所第一部上場企業 (一般事業会社, 連結本決算) の売上高や資産合計, 従業員数を抽出する例を考える. データの抽出は, トヨタ自動車の売上高を抽出する手順と本質的には変わらない (図 5.5 参照).

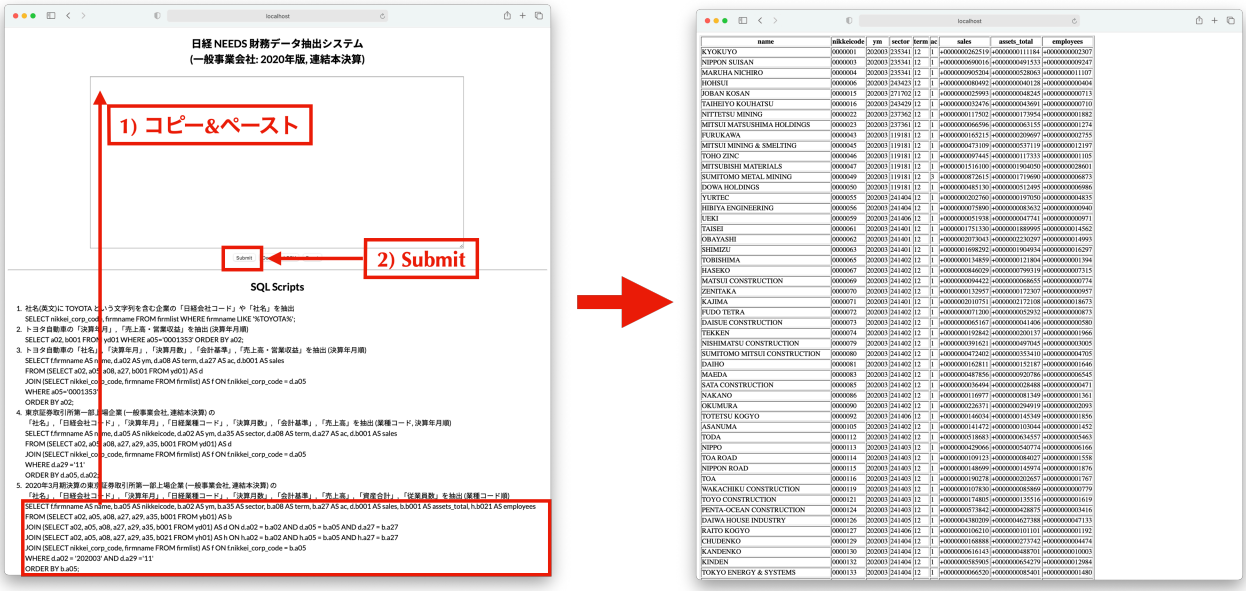


図 5.5: 2020 年 3 月期決算の東京証券取引所第一部上場企業 (一般事業会社, 連結本決算) の売上高, 資産合計, 従業員数等の抽出

この抽出に利用された SQL 問合せは以下のようなものである:

スクリプト 5.3: 2020 年 3 月期決算の東証 1 部上場企業 (連結本決算) の「売上高」, 「資産合計」, 「従業員数」等の抽出

```
SELECT
      f.firmname AS name,
      b.a05 AS nikkeicode,
      b.a02 AS ym,
      b.a35 AS sector,
      b.a08 AS term,
      b.a27 AS ac,
      d.b001 AS sales,
      b.b001 AS assets_total,
      h.b021 AS employees
      FROM (SELECT a02, a05, a08, a27, a29, a35, b001 FROM yb01) AS b
      JOIN (SELECT a02, a05, a08, a27, a29, a35, b001 FROM yd01) AS d
            ON d.a02 = b.a02 AND d.a05 = b.a05 AND d.a27 = b.a27
      JOIN (SELECT a02, a05, a08, a27, a29, a35, b021 FROM yh01) AS h
            ON h.a02 = b.a02 AND h.a05 = b.a05 AND h.a27 = b.a27
      JOIN (SELECT nikkei_corp_code, firmname FROM firmlist) AS f
            ON f.nikkei_corp_code = b.a05
            WHERE d.a02 = '202003' AND d.a29 ='11'
                  ORDER BY b.a05;
```

この SQL 問合せの説明を以下に与える. なお, 説明の都合上, 行に関して順不同となっている箇所がある.

11 行目: まず, 括弧内 (...) で SELECT 文によってデータを抽出している. ここでは, FROM 句に貸借対照表のテーブル yb01 を指定し, SELECT 句でテーブル yb01 のヘッダーパートから, 決算年月 (a02), 日経会社コード (a05), 決算月数 (a08), 会計基準 (a27), 上場場部 (a29), 日経業種コード (a35) を抽出し, 同じテーブル yb01 のデータパートから, 資産合計 (b001) の列を抽出し, AS 句でその結果をテーブル

名 b として定義している．次に，そのテーブル b を括弧外の FROM 句で指定する．

12 行目〜13 行目: まず，括弧内（...）で SELECT 文によってデータを抽出している．ここでは，FROM 句に損益計算書のテーブル yb01 を指定し，SELECT 句でヘッダーパートから，テーブル b と同じ列を抽出し，さらにデータパートから，売上高 (b001) の列を抽出した後，AS 句でそれらの結果をテーブル名 d と定義している．更に，ON 句で決算年月 (a02)，日経会社コード (a05)，会計基準 (a27) をキーとして指定 (ON d.a02 = b.a02 AND d.a05 = b.a05 AND d.a27 = b.a27) し，JOIN 句でテーブル b へ結合している．

14 行目〜15 行目: まず，括弧内（...）で SELECT 文によってデータを抽出している．ここでは，FROM 句に「その他・明細情報等」のテーブル yh01 を指定し，SELECT 句でヘッダーパートから，テーブル b と同じ列を抽出し，さらにデータパートから，(期末) 従業員数 (b021) を抽出した後，AS 句でそれらの結果をテーブル名 h として定義している．次に，ON 句で決算年月 (a02)，日経会社コード (a05)，会計基準 (a27) をキーとして指定 (ON d.a02 = b.a02 AND d.a05 = b.a05 AND d.a27 = b.a27) し，JOIN 句でテーブル b へ結合している．

16 行目〜17 行目: まず，括弧内（...）で SELECT 文によってデータを抽出している．ここでは，FROM 句に収録会社情報のテーブル firmlist を指定し，SELECT 句で日経会社コード (nikkei_corp_code) と社名 (firmname) を抽出し，AS 句でその結果をテーブル名 f として定義している．次に，ON 句で日経会社コードをキーとして指定 (ON f.nikkei_corp_code = b.a05) し，JOIN 句でテーブル b へ結合している．

18 行目: WHERE 句で決算年月が 2020 年 3 月 (d.a02 = '202003') と上場場部が東京証券取引所第一部 (d.a29 ='11') の行を抽出する条件を与えている．

19 行目: ORDER BY 句で日経会社コード (b.a05) の昇順で並べ替えている．

1 行目〜10 行目: AS 句を伴って，SELECT 句で社名 (f.firmname) を name，日経会社コード (b.a05) を nikkeicode，決算年月 (b.a02) を ym，日経業種コード (b.a35) を sector，決算月数 (b.a08) を term，会計基準 (b.a27) を ac，売上高 (d.b001) を sales，資産合計 (b.b001) を assets_total，従業員数 (h.b021) を employees として抽出している．

トヨタ自動車の売上高の例と同様に抽出結果を Excel へ コピー・アンド・ペーストし，可視化 (売上高と資産合計の散布図を作成) する工程を図 5.6 に与える．

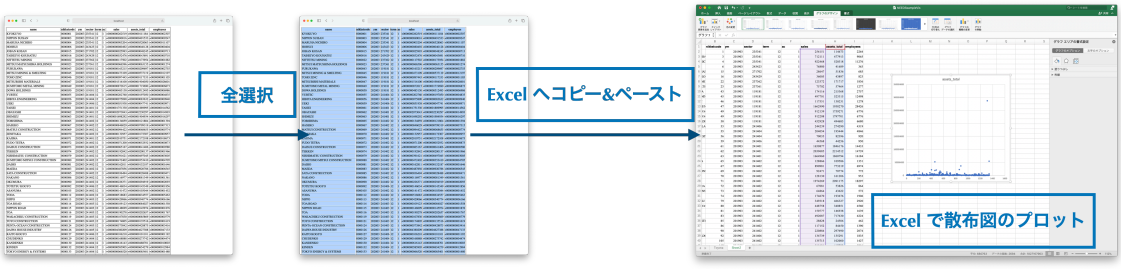


図 5.6: 2020 年 3 月期決算の東証一部上場企業 (連結本決算) の売上高 (x-軸) と資産合計 (y-軸) の散布図の作成

従来の財務データ抽出システム (KGUSBADES) では，ここで利用しているような，Web ブラウザから

Excel へコピー・アンド・ペーストを行うことによってデータを可視化・分析することを前提としてきた [1]. トヨタ自動車の売上高を多年度にわたって抽出する例では抽出されたデータは 37 行であったので, この手順は手軽さということでは適していると思われる. しかしながら, このような仕様は以下のような意味で限界があるように思われる.

(EP1) スクリプト 5.3 で与えられる抽出例では, 結果が 1000 行を超える (1340 行) ため,「コピー・アンド・ペースト」を「手作業」で行う段階で操作上の誤りをおかす可能性がある.

(EP2) Excel などのグラフィカル・ユーザ・インターフェース (Graphical User Interface: GUI) をベースとするソフトウェアで可視化を行った結果は, 手法の種類が限定されることやその再現性が乏しい.

問題 (EP1) は, 本稿の冒頭で述べた課題 (P6)「ダウンロード法の拡充」によってある程度対処することができる. 今回のシステム再構築では, `Download CSV` ボタンを用意することによって, CSV ファイルとしてデータをダウンロードするための機能を追加した. 例えば, 2020 年 3 月期決算の東京証券取引所第一部上場企業 (一般事業会社, 連結本決算) の売上高, 資産合計, 従業員数等の抽出した結果を CSV ファイルとしてダウンロードするためには, SQL 問合せのスクリプトを スクリプト入力ボックス にコピー・アンド・ペースト後, `Download CSV` ボタンをクリックすればよい (図 5.7 参照).

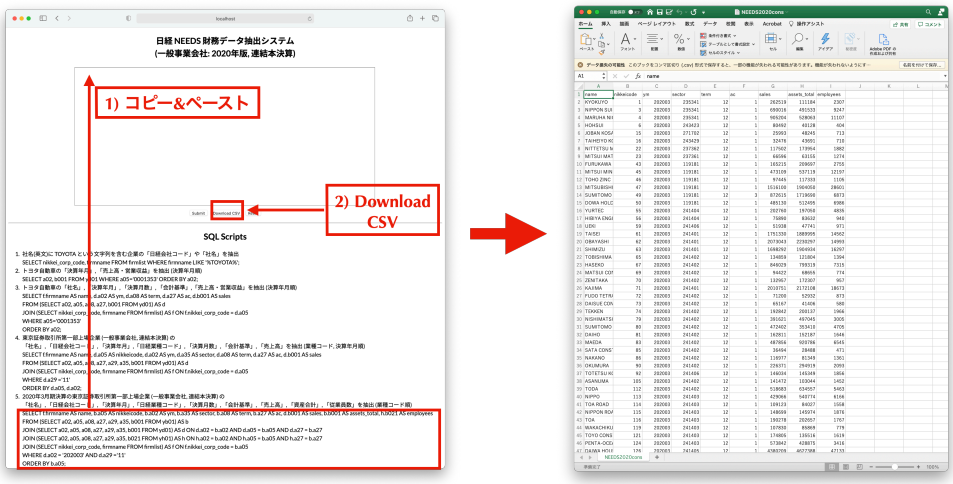


図 5.7: 2020 年 3 月期決算の東京証券取引所第一部上場企業 (一般事業会社, 連結本決算) の売上高, 資産合計, 従業員数等の抽出結果を CSV ファイル `NEEDS2020cons.csv` としてダウンロード

このようにダウンロードされた CSV ファイル `NEEDS2020cons.csv` は, Excel のみならず R [2] などのデータ解析環境に読み込んで可視化・分析・解析することも可能である. 例えば, 適当な場所 (作業ディレクトリ) に保存された CSV ファイル `NEEDS2020cons.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 [3].

---

[1] ここで, Web ブラウザから Excel へコピー・アンド・ペーストを行う場合でも, 結果を CSV ファイルに保存し, 他のデータ解析ソフトウェアに読み込んで可視化・分析することが可能である. 筆者が担当する講義や演習では, 高度な可視化・分析を行う際には, Excel 以外の専門的なソフトウェアで実施することを推奨してきた.

[2] `https://www.r-project.org/`

[3] 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.

R へのデータの読み込み

```
> x <- read.csv("NEEDS2020cons.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)
                        name nikkeicode     ym sector term ac  sales assets_total employees
1 KYOKUYO                              1 202003 235341   12  1 262519       111184      2307
2 NIPPON SUISAN                        3 202003 235341   12  1 690016       491533      9247
3 MARUHA NICHIRO                       4 202003 235341   12  1 905204       528063     11107
4 HOHSUI                               6 202003 243423   12  1  80492        40128       404
5 JOBAN KOSAN                         15 202003 271702   12  1  25993        48245       713
6 TAIHEIYO KOUHATSU                   16 202003 243429   12  1  32476        43691       710
```

ここで, 変数名は以下のようなものである:

name: 企業名
nikkeicode: 日経会社コード
ym: 決算年月
sector: 日経業種コード
term: 決算月数
ac: 会計基準 (1: 日本基準, 2: 米国会計基準, 3: 国際会計基準)
sales: 売上高 (単位: 百万円)
assets_total: 資産合計 (単位: 百万円)
employees: 従業員数 (単位: 人)

このデータオブジェクトの対散布図をプロットすることを考える．そのためには, 以下のようにまずオブジェクトを変換する必要がある．

データオブジェクトの変換

```
> library(dplyr)
> y <- x %>%
+   mutate(sales = na_if(sales, "-9999999999999"),
+          assets_total = na_if(assets_total, "-9999999999999"),
+          employees = na_if(employees, "-9999999999999"),
+          sector1 = as.factor(substring(sector,1,1)))
```

この R スクリプトでは, データ操作 (data manipulation) を行うための R パッケージ dplyr を読み込んでおり, このパッケージに付属の mutate 関数を利用して, 売上高 (sales), 資産合計 (assets_totala), 従業員数 (employees) に含まれる欠測値 (-9999999999999 [4]) を, 関数 na_if を利用して, 欠測値 (NA) として変換し, 変数を再定義している．また, 日経業種コード (sector) の 1 文字目が大分類 (1: 製造業, 2: 非製造業) を表すことから, 関数 substring を利用して切り出し, さらにそれを関数 as.facor を利用して, 因子型に変換

[4] NEEDS 企業財務データでは, 欠測値は -9999999999999 で表される．

したものを新しい列 sector1 として追加している.

以上の準備のもとで, 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: ggpair 関数を利用した場合

```
> library(GGally)
> y %>% select(sales, assets_total, employees, sector1) %>%
+         ggpairs(
+         mapping = aes(color = sector1),
+         upper = list(continuous = wrap("points", size = 0.5, alpha = 0.5)),
+         lower = list(continuous = wrap("cor", size = 3))
+         ) +
+         theme(
+         axis.text= element_text(size = 5),
+         axis.title = element_text(size = 3)
+         )
```

この R スクリプトでは, 関数 library をつかって GGally パッケージを呼び出し, そのパッケージに付属する ggpair 関数を利用して対散布図を描いている. パイプ演算子 %>% を利用して, パイプラインを構成することによって, データのオブジェクト y の列を選択するために select に引き渡し, さらに, 対散布図を描くために関数 ggpairs に引き渡している. なお, 日経業種分類 (大分類 sector1) で色分けするために, 審美的属性 (aesthetic attributes) を与えるための関数 aes を利用して引数 mapping に情報 (aes(color = sector1)) を与えており, 引数 upper と lower に対散布図の上三角ブロックと下三角ブロックに, 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. それ以外の指定は, 軸などに利用される文字の大きさを調整するものである. 詳細は, ggpairs のヘルプを参照されたい.

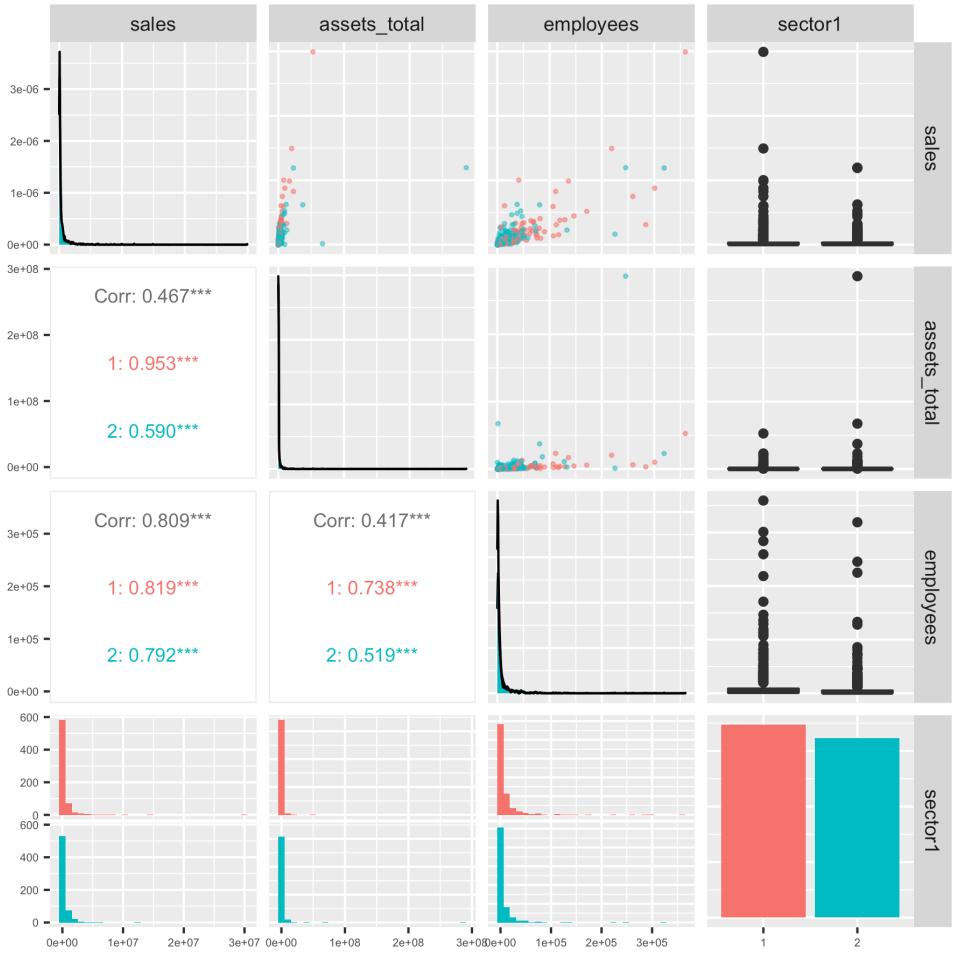


図 5.8: 対角ブロックには売上高 (sales), 資産合計 (assets_total), 従業員数 (employees) に関する推定された密度関数と, 日経産業分類 (sector1: 大分類) の頻度が描かれている. また, 上対角ブロックには, 売上高 (sales), 資産合計 (assets_total), 従業員数 (employees) の 2 つの組合せに関する散布図と最終列には, それぞれのデータのボックスプロット (1: 製造業, 2: 非製造業) が描かれている. さらに, 下対角ブロックには, 売上高 (sales), 資産合計 (assets_total), 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おり, 最終行には, それぞれのデータ頻度のバーチャート (1: 製造業, 2: 非製造業) が描かれている.

対散布図 (図 5.8) から, 業種によらず売上高 (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. さらに, 業種 (1: 製造業, 2: 非製造業) によって, 相関の強弱に違いがあることがわかる. 例えば, 売上高 (sales) と資産合計 (assets_total) の相関は, 製造業 (0.953) と非製造業 (0.59) で極端に異なっていることがわかる. 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. 以上のことから, 問題 (EP2) に対しては, ここで述べた方法が解決法の一つとなろう (図 5.9 も参照).

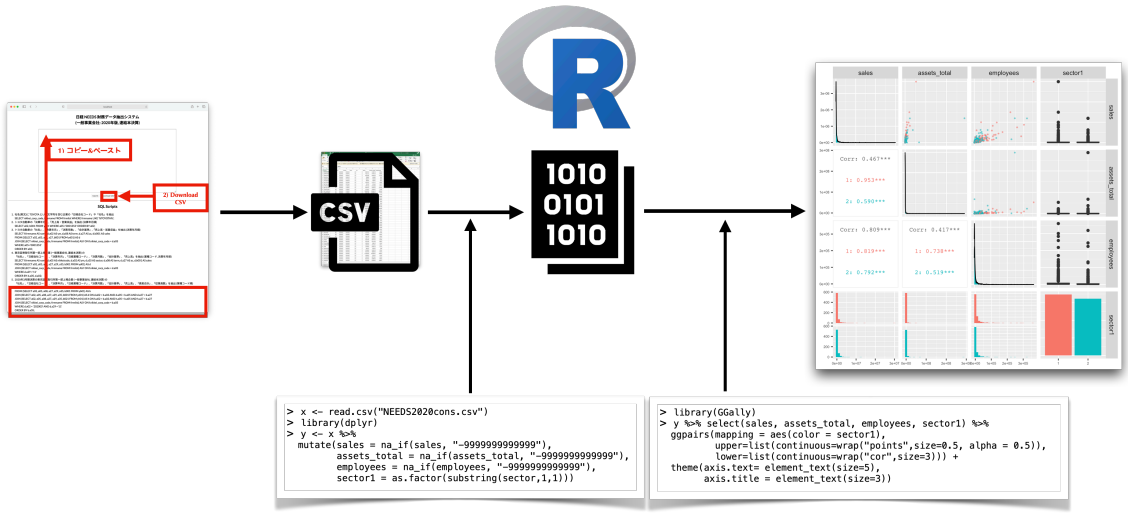


図 5.9: CSV ファイルとしてダウンロードされた NEEDS データファイルを R を利用して可視化する工程

5.2 Osiris データの抽出と可視化

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.

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 5.10 参照).

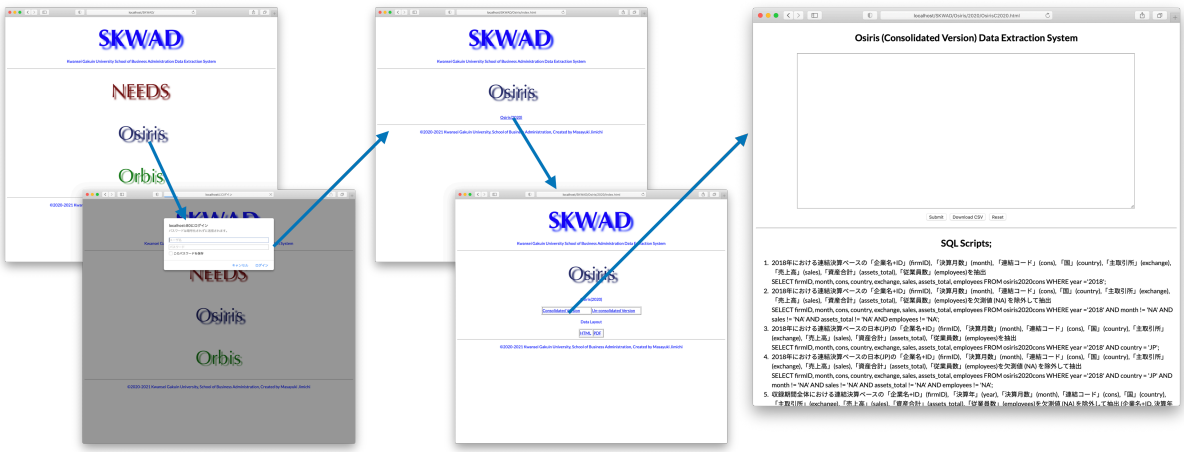


図 5.10: Osiris データ (連結ベース) の抽出ページへのアクセス

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 5.11 参照).

まず, SQL 問合せを 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.

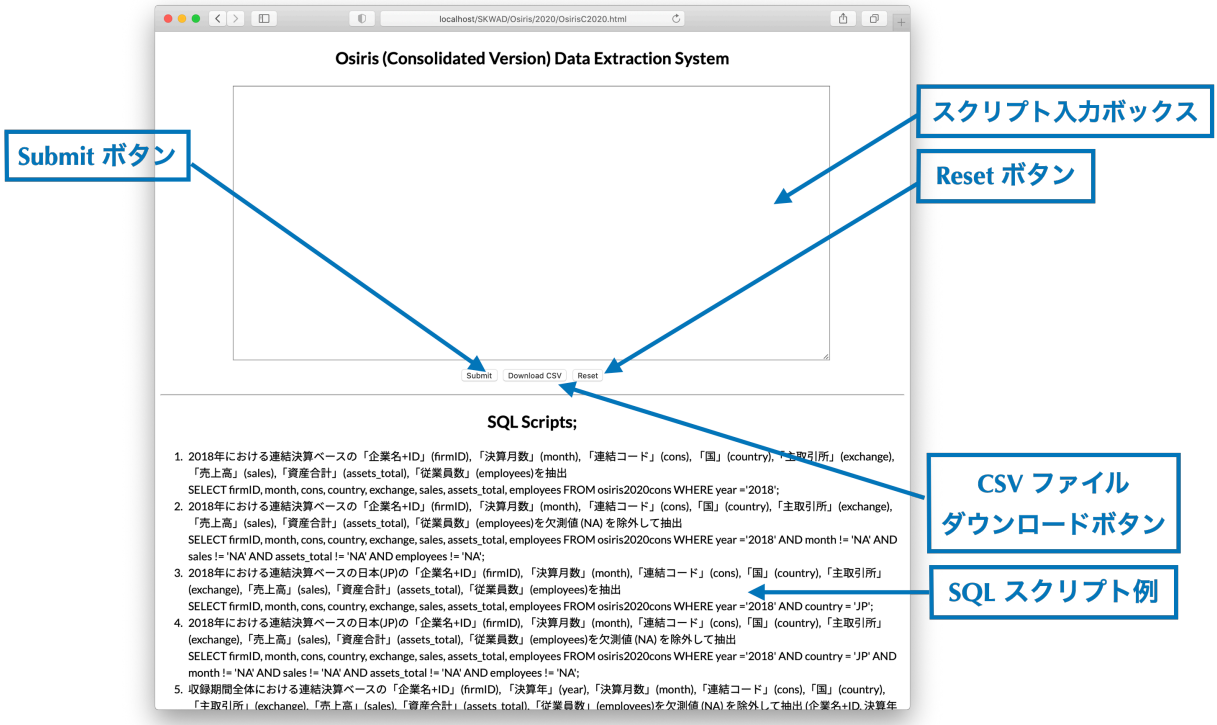


図 5.11: Osiris データ (連結ベース) 抽出システムのページ

なお, SQL のサンプルスクリプトも用意されている (図 5.11 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 5.4 に与える. なお, 列名の意味については, 第 9 章を参照されたい.

スクリプト 5.4: 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

```
1 SELECT firmID, month, cons, country, exchange, sales, assets_total,
    employees
2           FROM osiris2020cons
3                 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 [5] 」(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 osiris2020cons 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 5.12 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.

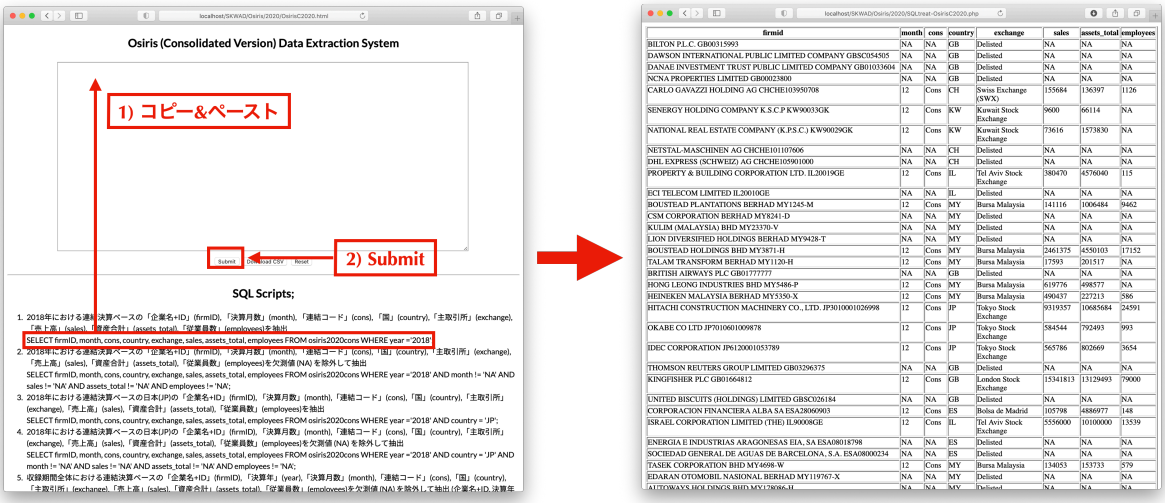


図 5.12: Osiris データ (連結ベース) の抽出

この手順を実行することによって, 図 5.12 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 [6].

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 5.13 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.

[5] BvD ID は BvD 社が各企業に一意に与えたコードである.
[6] 2.2 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.

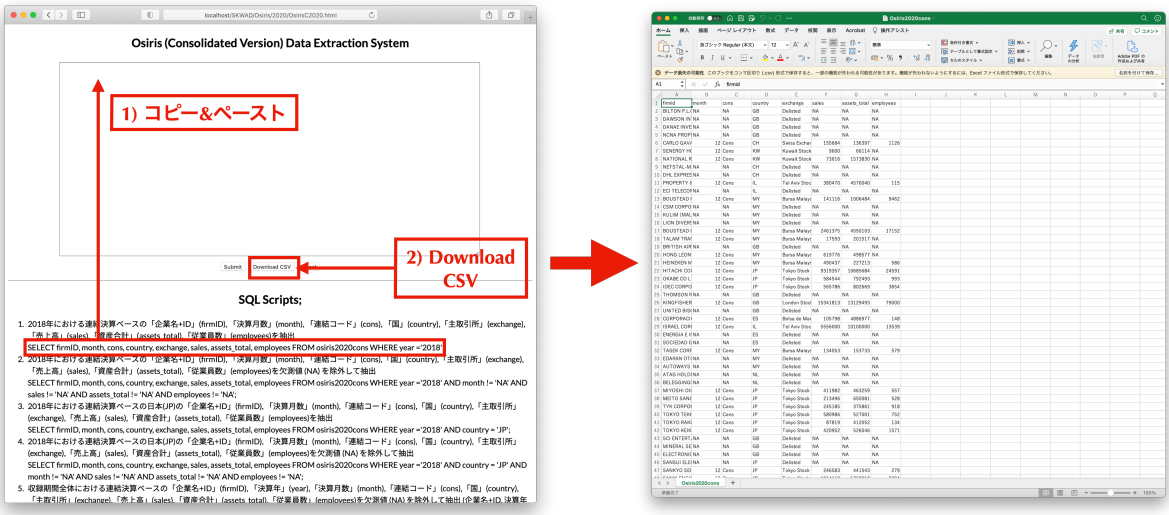


図 5.13: Osiris データ (連結ベース) の抽出結果を CSV ファイル (osiris2020cons.csv) としてダウンロード

次に, 抽出した Osiris データの CSV ファイル osiris2020cons.csv を R 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 read.csv 関数を利用して R に読み込む:

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 [7]:

firmid: 企業名 + BvD ID
month: 決算月数
cons: 連結コード (Cons: 連結)

[7] 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.

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.

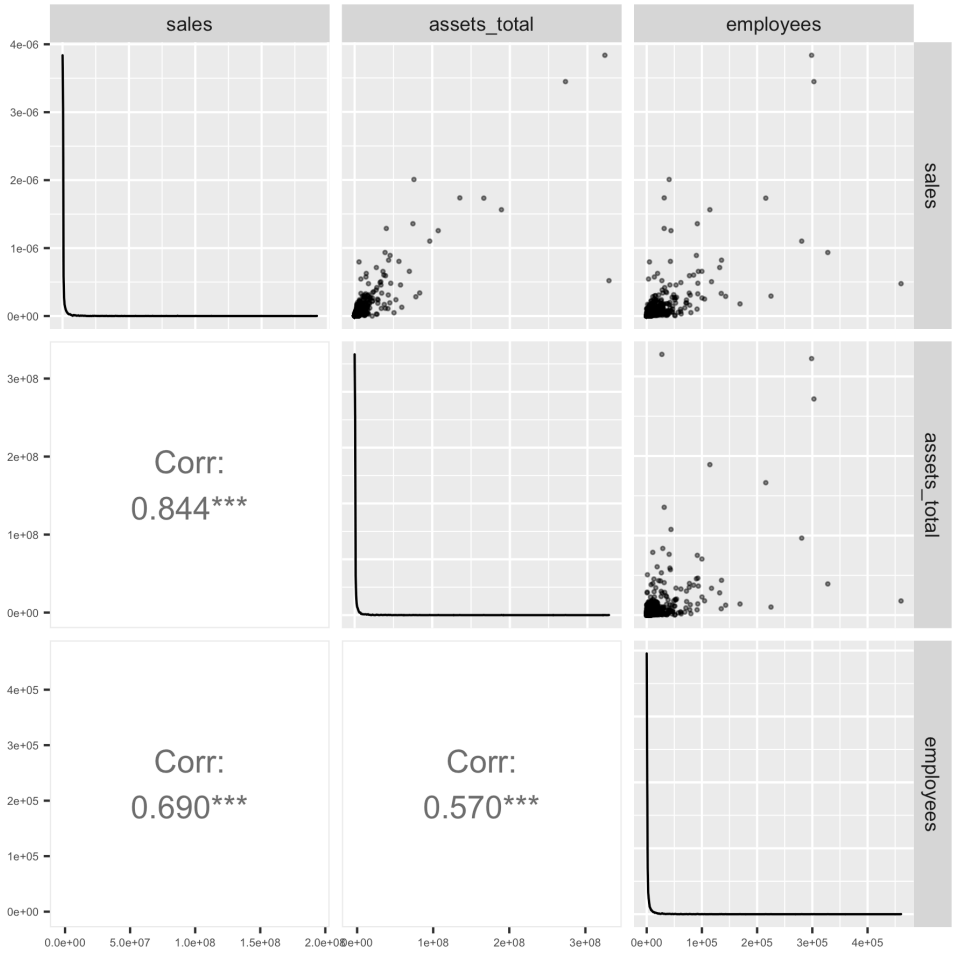


図 5.14: 対角ブロックには売上高 (sales), 資産合計 (assets_total), 従業員数 (employees) に関する推定された密度関数が描かれている．また, 上三角ブロックには, 売上高 (sales), 資産合計 (assets_total), 従業員数 (employees) の 2 つの組合せに関する散布図が描かれている．さらに, 下三角ブロックには, 売上高 (sales), 資産合計 (assets_total), 従業員数 (employees) の 2 つの組合せに関する相関係数が与えられている．

対散布図 (図 5.14) 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 (図 5.15 も参照).

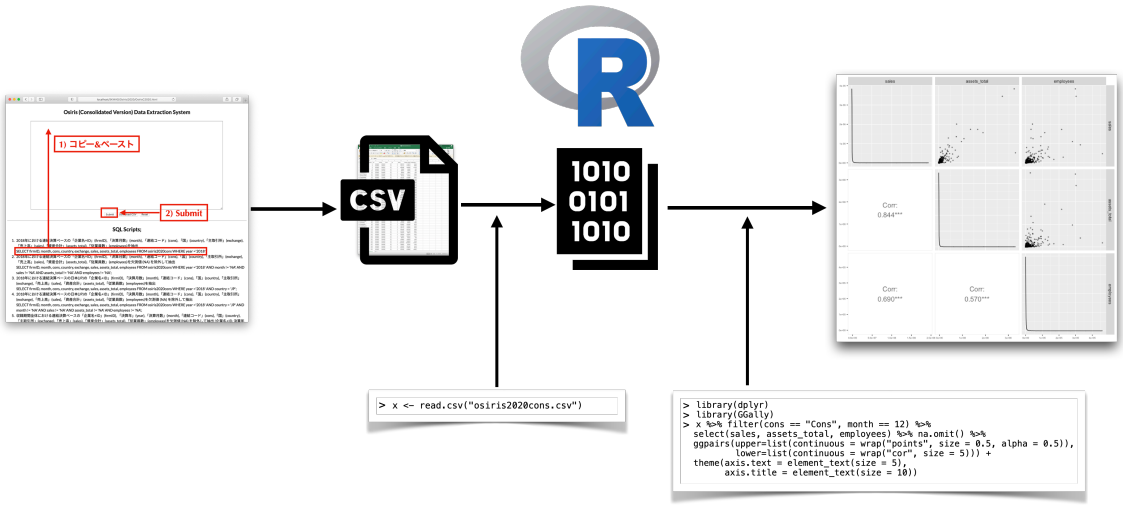


図 5.15: CSV ファイルとしてダウンロードされた Osiris データファイルを R を利用して可視化する工程

5.3 Orbis データの抽出と可視化

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ることができる (図 5.16 参照).

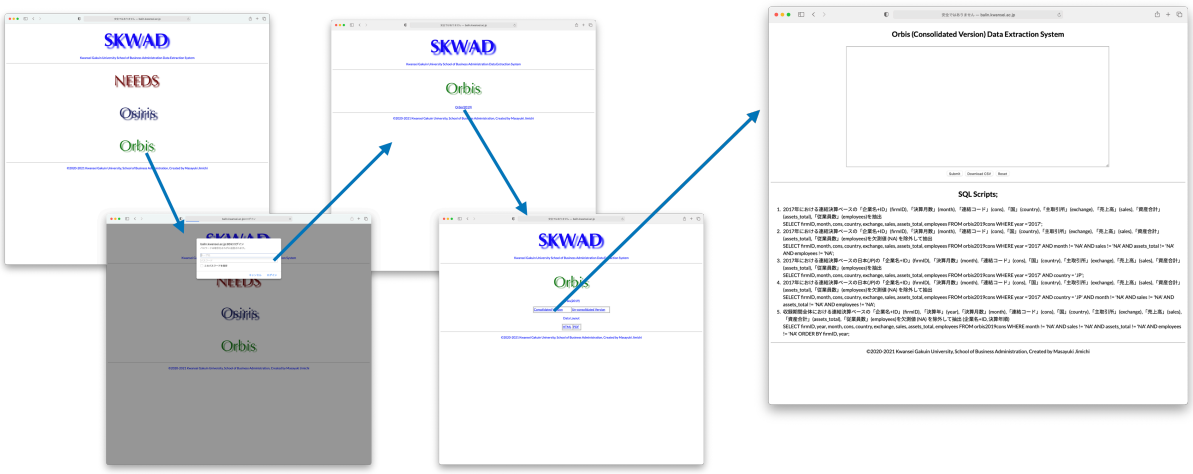


図 5.16: Orbis データ (連結ベース) の抽出ページへのアクセス

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 データの抽出と同

様であるが, 以下に簡単に説明する (図 5.17 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.

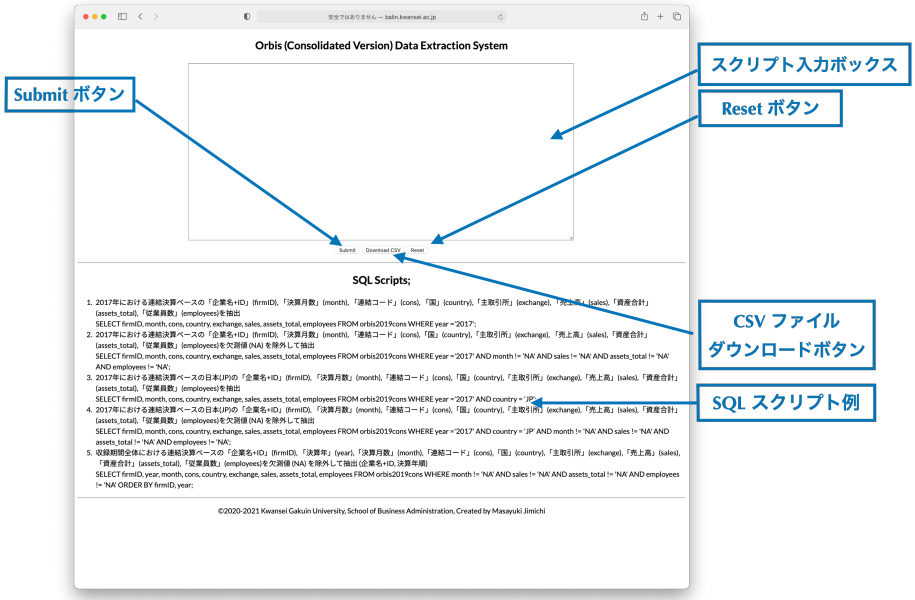


図 5.17: Orbis データ (連結ベース) 抽出システムのページ

なお, SQL のサンプルスクリプトも用意されている (図 5.17 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 5.5 に与える. なお, 列名の意味については, 第 10 章を参照されたい.

スクリプト 5.5: 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 [8] 」(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」

[8] BvD ID は BvD 社が各企業に与えた一意のコードである.

(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 5.5) 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 5.18 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.

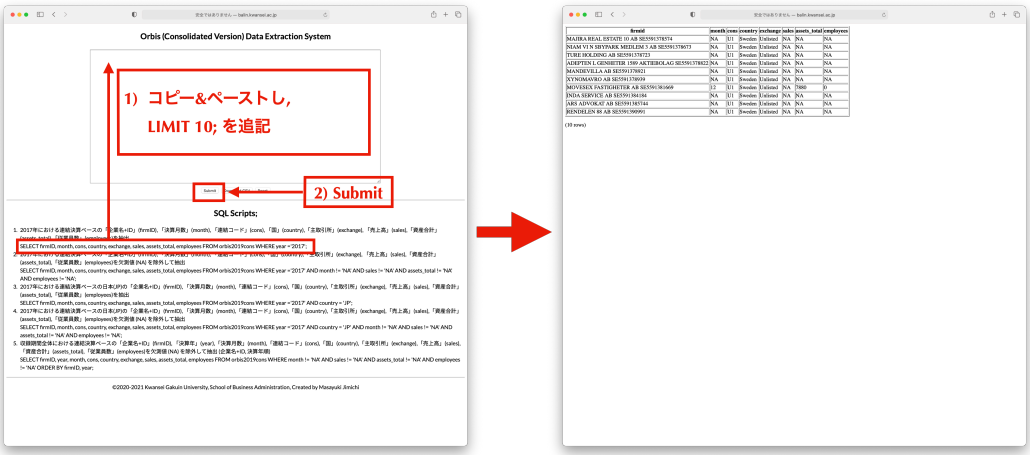


図 5.18: Orbis データ (連結ベース) の抽出

この手順を実行することによって, 図 5.18 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 5.19 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.

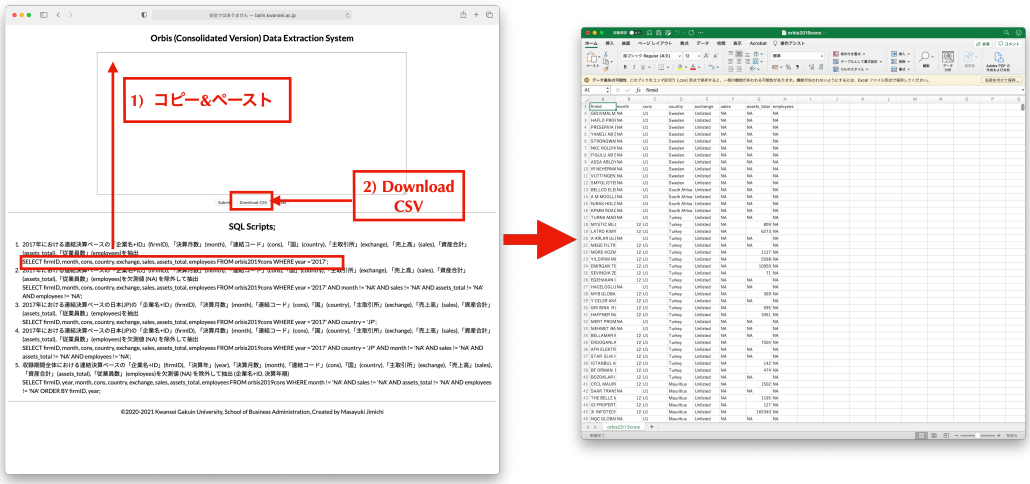


図 5.19: Orbis データ (連結ベース) の抽出結果を CSV ファイル (orbis2019cons.csv) としてダウンロード

では, Orbis データの CSV ファイル orbis2019cons.csv を R 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 read.csv 関数を利用して R に読み込む:

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 [9]:

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

[9] 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.

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 5.20) 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 (図 5.21 も参照).

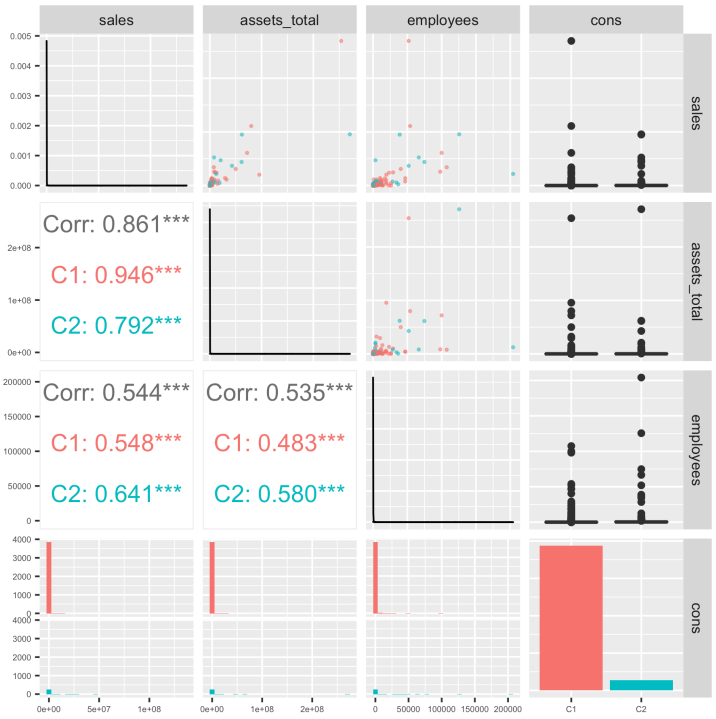


図 5.20: 対角ブロックには売上高 (sales), 資産合計 (assets_total), 従業員数 (employees) に関する推定された密度関数と連結コード (C1, C2) 別の企業数がバーチャートで描かれている．また，上三角ブロックには，売上高 (sales), 資産合計 (assets_total), 従業員数 (employees) の 2 つの組合せに関する散布図と連結コード別のボックスプロットが描かれている．，下三角ブロックには，売上高 (sales), 資産合計 (assets_total), 従業員数 (employees) の 2 つの組合せに関する相関係数と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．

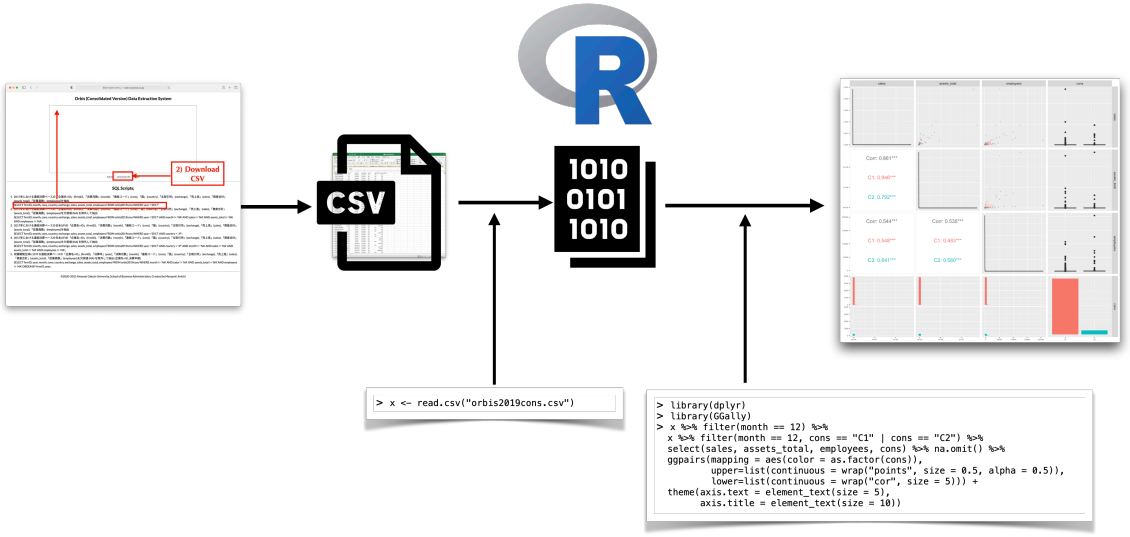


図 5.21: CSV ファイルとしてダウンロードされた Orbis データファイルを R を利用して可視化する工程

第 III 部

資料編

第 6 章

リレーショナル・データベースに関する基礎知識

ここでは, 地道 (2010) を参考に, データベースを理解する上で最も基本的な用語・知識を与える. なお, 詳しくは増永 (2017) などを参照されたい.

6.1 リレーショナル・データ・モデル

リレーショナル・データベースのもとになる概念であるリレーショナル・データ・モデルは Codd (1970) によるものである. ここでは, それに関連する用語をまとめる.

ドメイン (domain): 集合のこと. 記号として $D_1, \ldots, D_n$ で表される.
直積 (Cartesian product):

$$D_1 \times \cdots \times D_n := \{(d_1, \ldots, d_n) \mid d_i \in D_i, i = 1, \ldots, n\}$$

をドメイン $D_1, \ldots, D_n$ の直積 (集合) という.
タプル (tuple), レコード (record): ドメインの直積集合 $D_1 \times \cdots \times D_n$ の要素:

$$t := (d_1, \ldots, d_n) \in D_1 \times \cdots \times D_n$$

のこと.
リレーション (relation): ドメインの直積集合 $D_1 \times \cdots \times D_n$ の任意の有限部分集合:

$$R \subset D_1 \times \cdots \times D_n$$

のこと. すなわち, タプルを要素とする集合を表す. なお, リレーションが定義されているドメインの個数を次数 (degree) という.
リレーショナル・データベース (Relational DataBase; RDB): リレーショナル・データ・モデルにもとづいて設計, 開発されるデータベース
リレーショナル・データベース管理システム (Relational DataBase Management System; RDBMS): リレーショナル・データベースを管理するためのソフトウェア
属性 (attribute): タプルがあらわしている対象のもつ属性. 属性名を $A_1, \ldots, A_n$ で表す.
ドメイン関数: 属性 A_i からドメイン D_i への写像:

$$\mathrm{dom} : A_i \to D_i, \quad i = 1, \ldots, n$$

をドメイン関数 (domain function) という.

リレーションスキーマ (relation schema): リレーションの時間的に不変な性質
インスタンス (instance): 時間とともに変化するリレーションそのもの
シンプル (simple): ドメインが他のドメインの入れ子 (nest) や巾集合 (power set) 等であらわすことができないこと.
第 1 正規形 (the first Normal Form; 1NF): リレーションを定義するすべてのドメインがシンプルであること. 通常のリレーションには第 1 正規形であることが仮定されている.

注意 6.1.1. リレーションは 2 次元の単純なテーブル (table) と考えてもよく [1], タプルはその行 (row), テーブルの列 (column) は属性に対応し, ドメインは属性の取り得る値全体の集合といってもよい.

6.2　リレーショナル・データベース管理システム

リレーショナル・データベース (RDB) を管理するシステムがリレーショナル・データベース管理システム (Relational DataBase Management System; RDBMS) である. オープンソースとしては, MySQL, PostgreSQL, SQLite 等が存在し, さまざまな OS 上で稼働している.　リレーショナル・データベースは, テーブル (表, リレーション) から構成されており, テーブルにおける縦の並びをカラム (列, フィールド), 横の並びをレコード (行, タプル) と呼ぶ (図 6.1 も参照).

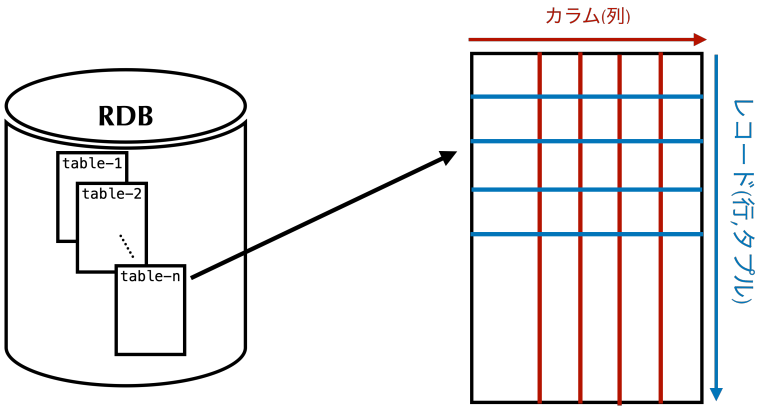


図 6.1: リレーショナル・データベース (RDB) の構造とテーブル

6.3　構造化照会言語

RDBMS とのインターフェース言語として構造化照会言語 (Structured Query Language; SQL) が国際標準として利用されている (図 6.2 参照). この言語は, 米国国家規格協会 (American National Standards Institute; ANSI) と国際標準化機構 (International Organization for Standardization; ISO) によって制定されたものである. SQL92 が ISO/IEC 9075 と ANSI X3.135-1992 で制定されており, 日本ではその邦訳が日本工業規格 (Japanese Industrial Standard; JIS) の JIS X 3005 として制定されている.

[1] ここでは,「ぶち抜き」や「入れ子」のないことを前提としている

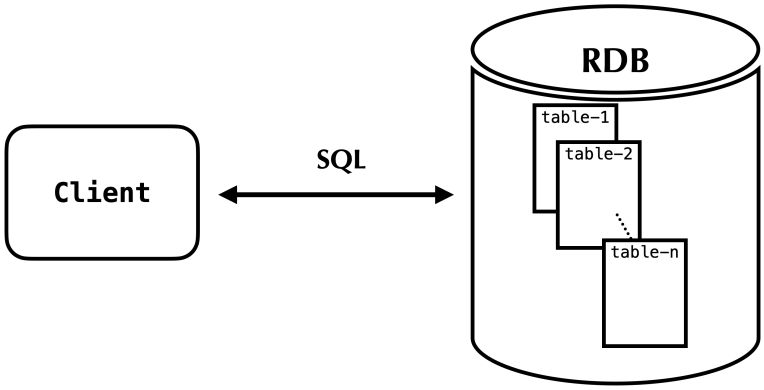


図 6.2: SQL の役割

リレーショナル・データ・モデルにおける用語と SQL における用語には以下の対応がある:

リレーショナル・データ・モデル	SQL
リレーション (relation)	テーブル (table)
属性 (attribute)	カラム, 列 (column)
タプル (tuple)	ロウ, 行 (row), レコード (record)

SQL によってテーブルからカラムを抽出するためは SELECT 文 (ステートメント) を利用する. 基本的な構文は以下のようなものである:

SELECT 文の基本構文

```
SELECT カラム名 1, カラム名 2,…
        FROM テーブル名
        JOIN テーブル名 ON 条件
                WHERE 条件 (LIKE 条件)
                        ORDER BY カラム名;
```

ここで, SELECT から FROM の間でカラム名を指定 (SELECT 句) し, FROM でテーブル名を選択するが, これらは必須である. また, それ以外の JOIN (テーブルの結合), WHERE (条件指定), ORDER BY (並べ替え) は, オプショナルである. なお, SELECT 句にはワイルドカードとしてアスタリスク (*) を与えることができ, そのテーブルの全てのカラム (列) が選択される.

第 7 章

開発環境

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 [1], macOS [2], 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 (RDBMS) と Web サーバとして Apahche HTTP Server [3], 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 7.1: 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 を導入し, 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 7.1 に与える.

従来のシステムでは RDBMS として MySQL を利用してきたが, MySQL を取り巻く最近の動向 [5] を勘案すると, データ抽出システムを PostgreSQL をベースに再構築することが, 環境の変化に対する頑健性を確保するために必要と考えられた [6]. このような理由から, 本稿では表 7.2 のような開発環境を用意した [7]:

---

[1] Ubuntu Linux(`https://ubuntu.com`) とは, 英カノニカル (Canonical) 社が開発を主催する Debian GNU/Linux 派生のフリーの Linux ディストリビューションのこと (IT 用語辞典 e-Words `https://e-words.jp/` 参照).

[2] macOS (`https://www.apple.com/jp/macos/monterey/`) とは、米アップル (Apple) 社がパーソナルコンピュータ「Mac」シリーズ向けに提供している OS 製品のこと (IT 用語辞典 e-Words `https://e-words.jp/` 参照).

[3] Apache HTTP Server (`https://httpd.apache.org`) とは, 世界的に最も普及している Web サーバサーバソフトウェアの一つ. Apache Software Foundation (`https://httpd.apache.org`) が開発しており, オープンソースソフトウェアとして公開している (IT 用語辞典 e-Words `https://e-words.jp/` 参照).

[4] PHP(`https://www.php.net/`) は, Web サーバの機能を拡張し, 動的に Web ページを生成するために用いられるプログラミング言語の一つ. PHP: Hypertext Preprocessor を再帰的に略したもの (IT 用語辞典 e-Words `https://e-words.jp/` 参照).

[5] Oracle 社 (`https://www.oracle.com/`) によって (2009 年から 2010 年に) Sun Microsystems 社が買収されたことにより, MySQL は Oracle 社によって開発が続けられている.

[6] RDBMS の選択を再検討したもう一つの理由が, MySQL のバージョン間での仕様変更である. 筆者だけかもしれないが, MySQL のバージョンアップ (例えば, バージョン 5.6 からバージョン 5.7) に伴う仕様変更には戸惑いを覚える. 筆者は利用したことがないが, MySQL からのフォークである MariaDB の導入も検討に値すると思われる. なお, MySQL については, たとえば, 西沢 (2017) を, PostgreSQL については, 鈴木 (2012) を参照されたい.

[7] 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.

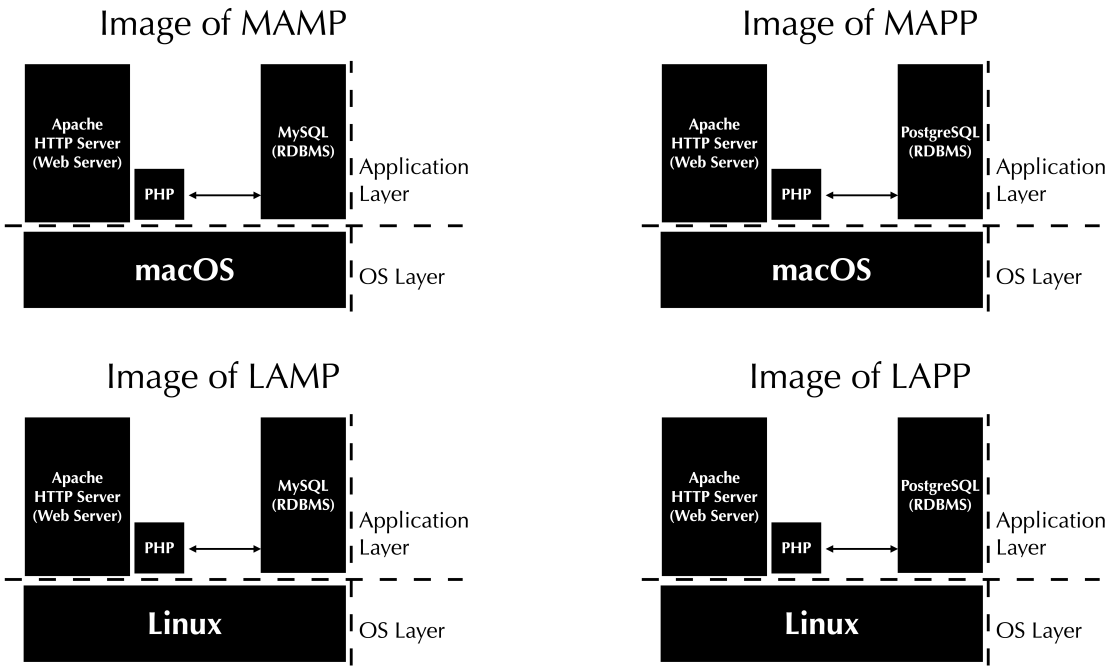


図 7.1: MAMP, MAPP, LAMP, LAPP 環境のイメージ

表 7.2: 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
Storage	SSD 4TB	SSD 4TB	SSD 4TB	SSD 4TB
RDBMS	MySQL 5.6.50	PostgreSQL 13.1	MySQL 8.0.22	PostgreSQL 12.5

第 8 章

2020 年版 NEEDS 企業財務データ (一般事業会社) の仕様

8.1 納品ファイル

ここでは, 日経メディアマーケティング社が販売している「NEEDS 企業財務データ (一般事業会社)」のファイルの 2020 年版に関する情報を与える (表 8.1 参照) [1].

表 8.1: NEEDS 企業財務データ (一般事業会社): 2020 年版

ファイル名	ファイルサイズ (Byte)	説明
HFSCNM010.zip	1316551	収録会社情報ファイル (圧縮済み)
HFSIDA920.1960.zip	9259287	1960 年代のデータファイル (圧縮済)
HFSIDA920.1970.zip	19375861	1970 年代のデータファイル (圧縮済)
HFSIDA920.1980.zip	28253657	1980 年代のデータファイル (圧縮済)
HFSIDA920.1990.zip	44890820	1990 年代のデータファイル (圧縮済)
HFSIDA920.2000.1.zip	72357847	2000 年代のデータファイル 1(圧縮済)
HFSIDA920.2000.2.zip	16454263	2000 年代のデータファイル 2(圧縮済)
HFSIDA920.2010.1.zip	10298255	2010 年代のデータファイル 1(圧縮済)
HFSIDA920.2010.2.zip	7843009	2010 年代のデータファイル 2(圧縮済)
HFSIDA920.2020.zip	7017406	2020 年代のデータファイル (圧縮済)

収録会社情報のファイル HFSCNM010.zip とデータファイル HFSIDA920.1960.zip 〜 HFSIDA920.2020.zip で構成されており, 全て ZIP 形式で圧縮されている. 以下にこれらのファイルを展開することによって得られるファイルの説明を行う.

[1] 本稿では 2018 年版, 2019 年版のデータも利用しているが, 構造は 2020 年版と同様である.

8.2　収録会社情報ファイル

収録会社情報のファイル HFSCNM010.zip を展開後, 文字コードを UTF-8 に変換し, 先頭の 1 行を表示した結果をリスト 8.1 に与える.

スクリプト 8.1: 収録会社情報のファイル HFSCNM010.zip の先頭 1 行

```
% 7z e -so HFSCNM010.zip | nkf -u | head -n 1
CN0000000011301 9851662019999            KYOKUYO                                極洋
                キヨクヨウ                          東京都港区赤坂３－３－５　住友
    生命山王ビル                      107005203-5545-0701
                                  2353411010401033225
```

展開後のファイルは, 1 行 300 バイト (Byte) であり, フィールドとフィールドを分ける区切り文字 [2] は無く, いわゆる, 固定長フォーマット (fixed length format) のファイルである. ファイルに含まれる項目の仕様は, 表 8.2 を参照されたい. なお, このファイルをデータベースのテーブルとして利用するためには, 表 8.2 に与えられているように, 項目の「位置」と「長さ」の情報をもとに適切に区切り文字を入れたり,「カラム名」を設定するなどの処理が必要となる.

表 8.2: 収録会社情報のファイル HFSCNM010 の仕様

項番	項目名	説明	カラム名	位置	長さ
1	レコード種別	CN00：固定	record_type	1	4
2	日経会社コード	日経が定める会社コード	nikkei_corp_code	5	7
3	株式コード		stock_code	12	4
4	予備		etc1	16	1
5	予備		etc2	17	9
6	金融機関コード	証券コード協議会が定める４桁の会社コード	finance_code	26	4
7	予備		etc3	30	11
8	英文略称	英文による会社名の略称（但し英文商号のない時はローマ字）	firmname	41	30
9	漢字略称	会社名の漢字略称	firmname_jp	71	30
10	カナ社名	カタカナによる会社名の略称（漢字モードのカタカナ表記）	firmname_jpk	101	50
11	本社事務所所在地・住所	本社事務所所在地の住所（漢字	address	151	80
12	本社事務所郵便番号	本社事務所所在地の郵便番号	zip_code	231	7
13	本社事務所電話番号	本社事務所所在地に対応する電話番号	phone_number	238	15
14	予備		etc4	253	27
15	日経業種コード	日経が定める業種コード	nikkei_ind_code	280	6
16	法人番号	未収録時スペース	corp_number	286	13
17	予備		etc5	299	2

[2] 区切り文字 (separator) とは, テキストデータ中で複数の要素を並べて記述する際に, 要素の区切りを表す記号や特殊な文字 (の並び) のことである. デリミタ, セパレータ, 分離記号, 分離文字などとも呼ばれる. 列挙された項目の区切りを表すものと、範囲の始まりと終わりを表すものがある (IT 用語辞典 e-Words https://e-words.jp/ 参照).

8.3　データファイル

データファイル HFSIDA920.1960.zip〜HFSIDA920.2020.zip を展開してできるファイルは, 一行 3040 バイトからなる固定長フォーマットである. そのうち, 先頭から 100 バイトは「ヘッダーパート」と呼ばれ, 残りの 2940 バイトは「データパート」と呼ばれる (図 **??** 参照).

ヘッダーパート	＋	データパート	＝	一行
100 バイト	＋	2940 バイト	＝	3040 バイト

なお, ヘッダーパートの「項目 (名)」は, 各行全てについて「共通」であることに注意しよう.

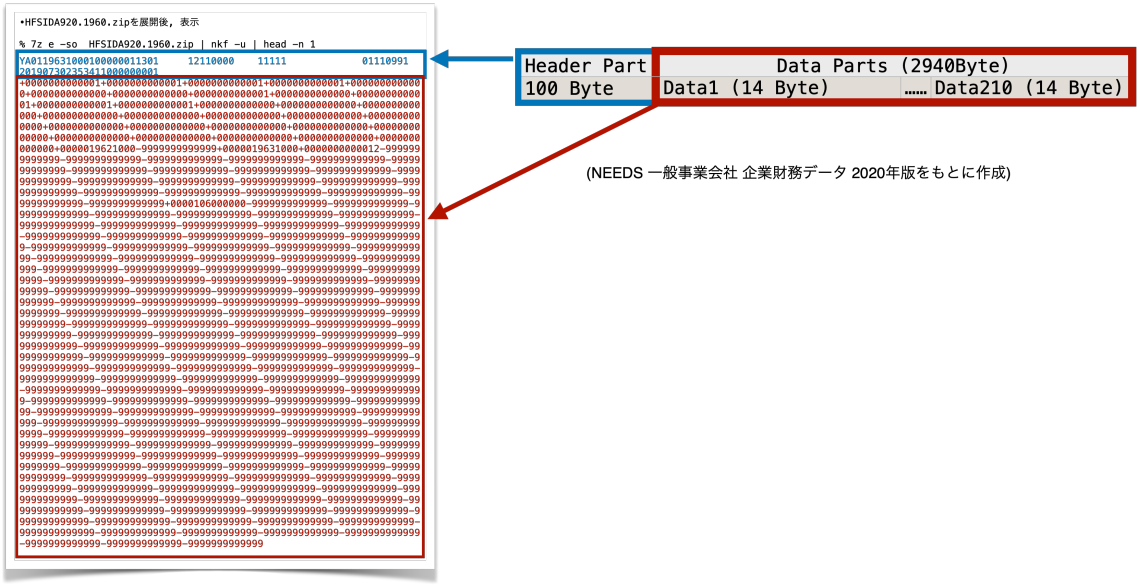
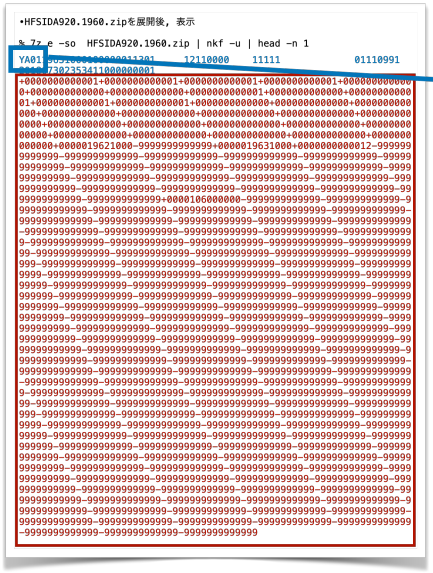


図 8.1: データファイル HFSIDA920.1960.zip 〜 HFSIDA920.2020.zip の仕様

ヘッダーパートに含まれる先頭 4 文字 (例えば, YA01) は「レコード種別」とよばれ, YA01〜 YL01 の 12 種類のものが用意されている. それぞれの意味については, 図 8.2 を参照されたい.



レコード種別	収録内容
YA01	基本情報（有価証券報告書）
YB01	貸借対照表
YC01	貸借対照表
YD01	損益計算書（累計）
YE01	損益計算書（3ヵ月）
YF01	キャッシュフロー計算書
YG01	株主資本等変動計算書
YH01	その他・明細情報等
YI01	その他・明細情報等
YJ01	その他・明細情報等
YK01	その他・明細情報等
YL01	更新停止項目

図 8.2: データファイル HFSIDA920.1960.zip 〜 HFSIDA920.2020.zip のレコード種別

ヘッダーパートの仕様の詳細を表 8.3 に与える．この表における「形式」は「型」と「長さ」の情報を与える．例えば，“C4” の場合は，長さ 4 の文字列 (character の頭文字 C) を表す．収録会社情報のファイル HFSCNM010.zip と同様に，データファイルをデータベースのテーブルとして利用するために，この情報と「カラム名」を利用する．

表 8.3: ヘッダーパートの仕様

項番	カラム名	形式	項目名	内容	Key 順位
1	a01	C4	レコード種別	YAO： （有価証券報告書）基本情報 YBO： （有価証券報告書）貸借対照表 YCO： （有価証券報告書）貸借対照表 YDO： （有価証券報告書）損益計算書（累計） YEO： （有価証券報告書）損益計算書（3 ヵ月） YFO： （有価証券報告書）キャッシュフロー計算書 YGO： （有価証券報告書）株主資本等変動計算書 YHO： （有価証券報告書）その他・明細情報 YIO： （有価証券報告書）その他・明細情報 YJO： （有価証券報告書）その他・明細情報 YKO： （有価証券報告書）その他・明細情報 YLO： （有価証券報告書）更新停止項目	第 5Key
2	a02	C6	決算年月	YYYYMM 形式	第 3Key
3	a03	C2	予備		
4	a04	C1	識別フラグ	1：新規・修正 9：削除	
5	a05	C7	日経会社コード	日経が定める会社コード	第 1Key
6	a06	C4	株式コード	証券コード協議会が定める 4 桁の会社コード	
7	a07	C5	予備		
8	a08	C2	決算月数	決算月数（累計会計期間の月数を収録）	
9	a09	C1	連結・単独フラグ	1：単独 2：連結	
10	a10	C2	決算種別フラグ	10：本決算 21：第 1 四半期 24：第 4 四半期 22：第 2 四半期（中間決算） 25：第 5 四半期 23：第 3 四半期	第 4Key
11	a11	C1	決算短信情報収録フラグ	基本情報（1）収録フラグ 1：収録 0：未収録	
12	a12	C1	決算短信情報収録フラグ	基本情報（2）収録フラグ 1：収録 0：未収録	
13	a13	C1	決算短信情報収録フラグ	明細情報（1），（2）収録フラグ 1：収録 0：未収録	
14	a14	C1	予備		
15	a15	C1	予備		
16	a16	C1	予備		
17	a17	C1	予備		
18	a18	C1	有価証券報告書情報収録フラグ	基本情報（1）収録フラグ 1：収録 0：未収録	
19	a19	C1	有価証券報告書情報収録フラグ	基本情報（2）収録フラグ 1：収録 0：未収録	
20	a20	C1	有価証券報告書情報収録フラグ	明細情報（1），（2），（3）収録フラグ 1：収録 0：未収録	
21	a21	C1	有価証券報告書情報収録終了フラグ（明細情報）	1：有価証券報告書からのデータ収録のうち、明細情報（1），（2）の収録まで終了 0：未終了	
22	a22	C1	有価証券報告書情報収録終了フラグ（基本情報）	1：有価証券報告書からのデータ収録のうち、基本情報（1），（2）の収録が終了 0：未終了	
23	a23	C1	予備		
24	a24	C1	予備		
25	a25	C1	予備		
26	a26	C10	予備		
27	a27	C1	連結基準フラグ	1：日本基準 2：米国会計基準 3：国際会計基準 0：単独	第 2Key
28	a28	C1	上場フラグ	1：上場中 0：未上場・上場廃止	
29	a29	C2	上場場部	11：東証 1 部 12：東証 2 部（TOKYO PRO Market 内国を含む）13：東証マザーズ 21：大証 1 部 22：大証 2 部 31：名証 1 部 32：名証 2 部（セントレックスを含む） 41：京都 51：広島 61：福岡（Q ボードを含む） 71：新潟 81：札幌（アンビシャスを含む） 91：ヘラクレス・スタンダード 94：ヘラクレス・グロース 99：未上場 ※上場廃止会社は、廃止時の場部	
30	a30	C1	ジャスダックフラグ	1：ジャスダック上場 0：ジャスダック未上場	
31	a31	C2	ジャスダック市場	11：ジャスダック上場 99：ジャスダック未上場 ※上場廃止会社は、廃止時の市場 ※ジャスダック取引所化以前は下記の通りに収録。 1 バイト目 1:東京 2:大阪 3:名古屋の市場（1989 年 8 月以降は"1"のみ） 2 バイト目 1:上場（第一号基準銘柄、スタンダード、グロース） 2:管理	
32	a32	C1	有報フラグ	1：有報提出会社 0：非有報提出会社	
33	a33	C6	予備		
34	a34	C8	データ作成日	YYYYMMDD 形式 ピーク運用時は 24 時を超え翌日が設定される場合あり	
35	a35	C6	日経業種コード	ABBCCC 形式 A: 製造業;1，非製造業;2 B: 日経業種中分類コード C: 日経業種小分類コード	
36	a36	C1	上場情報 ：東京	1：1 部上場 2：2 部上場（TOKYO PRO Market 内国を含む）3：マザーズ 0：未上場	
37	a37	C1	〃 ：大阪	1：1 部上場 2：2 部上場 0：未上場	
38	a38	C1	〃 ：名古屋	1：1 部上場 2：2 部上場（セントレックスを含む） 0：未上場	
39	a39	C1	〃 ：京都	1：1 部上場 0：未上場	
40	a40	C1	〃 ：広島	1：1 部上場 0：未上場	
41	a41	C1	〃 ：福岡	1：1 部上場（Q ボードを含む） 0：未上場	
42	a42	C1	〃 ：新潟	1：1 部上場 0：未上場	
43	a43	C1	〃 ：札幌	1：1 部上場（アンビシャスを含む） 0：未上場	
44	a44	C1	〃 ：ヘラクレス	1：スダンダード 4：グロース 0：未上場	
45	a45	C1	レコード変更フラグ	1：変更あり 0：変更なし（今回変更データの有無をレコード種別毎に収録）	
46	a46	C3	予備		

(NEEDS 企業財務データ 一般事業会社, 2020 年版をもとに筆者作成)

次に, データパートのサイズは,

$$\begin{aligned} \text{「データパート」サイズ} &= 210(\text{項目}) \times 14(\text{バイト}) \\ &= 2940 \text{ バイト} \end{aligned}$$

であり, これはレコード種別 YA01〜YL01 によって変化しない. 一方, 項目は, レコード種別によって変化する.

レコード種別 YA01〜YL01 毎の収録項目の説明を表 8.4〜8.15 に与える. なお, サイズは項目毎に 14 バイトに固定されており, 「カラム名」はデータベースのテーブルとして利用するときに利用する.

YA01 基本情報 (有価証券報告書)

表 8.4: 基本情報 (有価証券報告書)

項番	カラム名	YA01 基本情報（有価証券報告書）
1	b001	レコードＴＡ０*/ＹＡ０*収録フラグ
2	b002	レコードＴＢ０*/ＹＢ０*収録フラグ
3	b003	レコードＴＣ０*/ＹＣ０*収録フラグ
4	b004	レコードＴＤ０*/ＹＤ０*収録フラグ
5	b005	レコードＴＥ０*/ＹＥ０*収録フラグ
6	b006	レコードＴＦ０*/ＹＦ０*収録フラグ
7	b007	レコードＴＧ０*/ＹＧ０*収録フラグ
8	b008	レコードＴＨ０*/ＹＨ０*収録フラグ
9	b009	レコードＴＩ０*/ＹＩ０*収録フラグ
10	b010	レコードＴＪ０*/ＹＪ０*収録フラグ
11	b011	レコードＴＫ０*/ＹＫ０*収録フラグ
12	b012	レコードＴＬ０*/ＹＬ０*収録フラグ
13	b013	予備
14	b014	予備
15	b015	予備
16	b016	予備
17	b017	予備
18	b018	予備
19	b019	予備
20	b020	予備
21	b021	予備
22	b022	予備
23	b023	予備
24	b024	予備
25	b025	予備
26	b026	予備
27	b027	予備
28	b028	予備
29	b029	予備
30	b030	予備
31	b031	事業年度開始年月日≪累計≫
32	b032	事業年度開始年月日≪３カ月≫
33	b033	事業年度終了年月日
34	b034	決算月数
35	b035	決算日数

項番	カラム名	YA01 基本情報（有価証券報告書）
36	b036	決算発表日
37	b037	定時株主総会開催日
38	b038	配当支払開始日
39	b039	会計方針の変更
40	b040	持分法適用範囲変更フラグ
41	b041	連結範囲変更フラグ
42	b042	連結子会社数
43	b043	（うち上場会社数）
44	b044	非連結子会社数
45	b045	関連会社数
46	b046	持分法適用の非連結子会社・関連会社数
47	b047	持分法適用非連結子会社数
48	b048	持分法適用関連会社数
49	b049	持分法非適用の非連結子会社・関連会社数
50	b050	親会社コード（日経会社コード）
51	b051	親会社コード（株式コード）
52	b052	実質親会社コード（日経会社コード）
53	b053	実質親会社コード（株式コード）
54	b054	親会社における議決権所有割合
55	b055	期中平均株式数≪累計≫
56	b056	期中平均株式数≪３カ月≫
57	b057	期末発行済株式総数
58	b058	自己株式数
59	b059	一株当たり利益≪累計≫
60	b060	一株当たり利益≪３カ月≫
61	b061	一株当たり純資産
62	b062	潜在株式調整後一株当たり利益≪累計≫
63	b063	潜在株式調整後一株当たり利益≪３カ月≫
64	b064	予備
65	b065	配当性向≪累計≫
66	b066	予備
67	b067	純資産配当率≪累計≫
68	b068	予備
69	b069	潜在株式数（普通株式増加数）
70	b070	純資産減少割合
71	b071	報告書提出予定日・提出日
72	b072	持分法を適用した場合の投資損益≪累計≫
73	b073	持分法を適用した場合の投資損益≪３カ月≫
74	b074	利益剰余金からの配当総額（1Q）

項番	カラム名	YA01 基本情報（有価証券報告書）
75	b075	うち普通株配当総額
76	b076	うち種類株配当総額
77	b077	利益剰余金からの配当総額（2Q）
78	b078	うち普通株配当総額
79	b079	うち種類株配当総額
80	b080	利益剰余金からの配当総額（3Q）
81	b081	うち普通株配当総額
82	b082	うち種類株配当総額
83	b083	利益剰余金からの配当総額（4Q）
84	b084	うち普通株配当総額
85	b085	うち種類株配当総額
86	b086	利益剰余金からの配当総額（5Q）
87	b087	うち普通株配当総額
88	b088	うち種類株配当総額
89	b089	利益剰余金からの配当総額（各期末）
90	b090	うち普通株配当総額
91	b091	うち種類株配当総額
92	b092	利益剰余金からの配当金総額（累計）
93	b093	うち普通株配当総額
94	b094	うち種類株配当総額
95	b095	資本剰余金からの配当総額（1Q）
96	b096	うち普通株配当総額
97	b097	うち種類株配当総額
98	b098	資本剰余金からの配当総額（2Q）
99	b099	うち普通株配当総額
100	b100	うち種類株配当総額
101	b101	資本剰余金からの配当総額（3Q）
102	b102	うち普通株配当総額
103	b103	うち種類株配当総額
104	b104	資本剰余金からの配当総額（4Q）
105	b105	うち普通株配当総額
106	b106	うち種類株配当総額
107	b107	資本剰余金からの配当総額（5Q）
108	b108	うち普通株配当総額
109	b109	うち種類株配当総額
110	b110	資本剰余金からの配当総額（各期末）
111	b111	うち普通株配当総額
112	b112	うち種類株配当総額
113	b113	資本剰余金からの配当総額（累計）

項番	カラム名	YA01 基本情報（有価証券報告書）
114	b114	うち普通株配当総額
115	b115	うち種類株配当総額
116	b116	1株当たり配当金（1Q）：普通株
117	b117	うち普通配当
118	b118	うち記念配当
119	b119	うち資本剰余金を原資とする配当
120	b120	1株当たり配当金（2Q・中間）
121	b121	うち普通配当
122	b122	うち記念配当
123	b123	うち資本剰余金を原資とする配当
124	b124	1株当たり配当金（3Q）
125	b125	うち普通配当
126	b126	うち記念配当
127	b127	うち資本剰余金を原資とする配当
128	b128	1株当たり配当金（4Q）
129	b129	うち普通配当
130	b130	うち記念配当
131	b131	うち資本剰余金を原資とする配当
132	b132	1株当たり配当金（5Q）
133	b133	うち普通配当
134	b134	うち記念配当
135	b135	うち資本剰余金を原資とする配当
136	b136	1株当たり配当金（各期末）
137	b137	うち普通配当
138	b138	うち記念配当
139	b139	うち資本剰余金を原資とする配当
140	b140	1株当たり配当金（累計）
141	b141	うち普通配当
142	b142	うち記念配当
143	b143	うち資本剰余金を原資とする配当
144	b144	1株当たり配当金（上場種類株）（1Q）
145	b145	1株当たり配当金（上場種類株）（2Q）
146	b146	1株当たり配当金（上場種類株）（3Q）
147	b147	1株当たり配当金（上場種類株）（4Q）
148	b148	1株当たり配当金（上場種類株）（5Q）
149	b149	1株当たり配当金（上場種類株）（期末）
150	b150	1株当たり配当金（上場種類株）（累計）
151	b151	うち特別配当（1Q）
152	b152	うち特別配当（2Q）

項番	カラム名	YA01 基本情報（有価証券報告書）
153	b153	うち特別配当（3Q）
154	b154	うち特別配当（4Q）
155	b155	うち特別配当（5Q）
156	b156	うち特別配当（各期末）
157	b157	うち特別配当（累計）
158	b158	潜在株式数（普通株式増加数）≪３カ月≫
159	b159	期末発行済株式総数：遡及調整後
160	b160	自己株式数：遡及調整後
161	b161	一株当たり利益≪累計≫：遡及調整後
162	b162	一株当たり利益≪３カ月≫：遡及調整後
163	b163	一株当たり純資産：遡及調整後
164	b164	潜在株式調整後一株当たり利益≪累計≫：遡及調整後
165	b165	潜在株式調整後一株当たり利益≪３カ月≫：遡及調整後
166	b166	期中平均株式数≪累計≫：遡及調整後
167	b167	期中平均株式数≪３カ月≫：遡及調整後
168	b168	潜在株式数（普通株式増加数）≪累計≫：遡及調整後
169	b169	潜在株式数（普通株式増加数）≪３カ月≫：遡及調整後
170	b170	【単位フラグ】期中平均株式数≪累計≫
171	b171	【単位フラグ】期中平均株式数≪３カ月≫
172	b172	【単位フラグ】期末発行済株式総数
173	b173	【単位フラグ】期中平均株式数≪累計≫：遡及調整後
174	b174	【単位フラグ】期中平均株式数≪３カ月≫：遡及調整後
175	b175	【単位フラグ】期末発行済株式総数：遡及調整後
176	b176	予備
177	b177	予備
178	b178	予備
179	b179	予備
180	b180	予備
181	b181	予備
182	b182	予備
183	b183	予備
184	b184	予備
185	b185	予備
186	b186	予備
187	b187	予備
188	b188	予備
189	b189	予備
190	b190	予備
191	b191	予備

項番	カラム名	YA01 基本情報（有価証券報告書）
192	b192	予備
193	b193	予備
194	b194	予備
195	b195	予備
196	b196	予備
197	b197	予備
198	b198	予備
199	b199	予備
200	b200	予備
201	b201	予備
202	b202	予備
203	b203	予備
204	b204	予備
205	b205	予備
206	b206	予備
207	b207	予備
208	b208	予備
209	b209	予備
210	b210	予備

YB01 (貸借対照表)

表 8.5: YB01 (貸借対照表)

項番	カラム名	YB01 （貸借対照表）
1	b001	資産合計
2	b002	予備
3	b003	予備
4	b004	予備
5	b005	予備
6	b006	予備
7	b007	予備
8	b008	予備
9	b009	予備
10	b010	予備
11	b011	予備
12	b012	予備
13	b013	予備
14	b014	予備
15	b015	予備
16	b016	予備
17	b017	予備
18	b018	予備
19	b019	予備
20	b020	予備
21	b021	流動資産
22	b022	現金・預金
23	b023	受取手形・売掛金
24	b024	受取手形
25	b025	売掛金
26	b026	非連結子会社関連会社　受取手形・売掛金
27	b027	リース債権及びリース投資資産
28	b028	リース債権
29	b029	リース投資資産
30	b030	営業貸付金・営業投資有価証券
31	b031	営業貸付金
32	b032	営業投資有価証券
33	b033	有価証券
34	b034	棚卸資産
35	b035	商品・製品

項番	カラム名	YB01（貸借対照表））
36	b036	商品
37	b037	製品
38	b038	販売用不動産
39	b039	半製品・仕掛品
40	b040	半製品
41	b041	仕掛品・未成工事支出金
42	b042	うち未成工事支出金
43	b043	原材料・貯蔵品
44	b044	原材料
45	b045	貯蔵品
46	b046	その他棚卸資産
47	b047	前渡金
48	b048	前払費用
49	b049	未収入金
50	b050	未収収益
51	b051	短期貸付金
52	b052	短期貸付金
53	b053	役員・従業員短期貸付金
54	b054	繰延税金資産
55	b055	金銭の信託
56	b056	デリバティブ債権
57	b057	繰延ヘッジ損失
58	b058	自己株式
59	b059	その他流動資産
60	b060	貸倒引当金（▲）
61	b061	その他引当金（▲）
62	b062	固定資産
63	b063	有形固定資産
64	b064	償却対象有形固定資産
65	b065	建物・構築物
66	b066	建物
67	b067	構築物
68	b068	機械装置及び運搬具
69	b069	機械及び装置
70	b070	船舶・車両・運搬具
71	b071	工具・器具及び備品
72	b072	リース資産
73	b073	その他償却対象有形固定資産
74	b074	建設仮勘定

項番	カラム名	YB01（貸借対照表)）
75	b075	土地・その他非償却対象有形固定資産
76	b076	無形固定資産
77	b077	特許権・実用新案権
78	b078	ソフトウェア
79	b079	のれん
80	b080	リース資産
81	b081	商標権
82	b082	意匠権
83	b083	その他無形固定資産
84	b084	投資・その他の資産合計
85	b085	投資有価証券・関係会社株式・出資金
86	b086	投資有価証券
87	b087	関係会社有価証券
88	b088	出資金
89	b089	関係会社出資金
90	b090	非連結子会社関連会社株式・社債・出資金
91	b091	長期貸付金
92	b092	非連結子会社関連会社　長期貸付金
93	b093	破産債権・更生債権・長期債権
94	b094	長期前払費用
95	b095	退職給付に係る資産（前払年金費用）
96	b096	投資不動産
97	b097	敷金・差入保証金
98	b098	繰延税金資産
99	b099	再評価に係る繰延税金資産
100	b100	デリバティブ債権
101	b101	その他の投資・その他の資産
102	b102	貸倒引当金（▲）
103	b103	投資損失引当金・その他引当金（▲）
104	b104	繰延資産
105	b105	社債発行差金
106	b106	開発費（旧：開発費・試験研究費）
107	b107	その他繰延資産
108	b108	連結調整勘定
109	b109	為替換算調整勘定
110	b110	資産合計
111	b111	営業権
112	b112	売却目的保有資産
113	b113	うち未収入金

項番	カラム名	YB01（貸借対照表))
114	b114	預け金・関係会社預け金
115	b115	うち受取手形・売掛金
116	b116	うち貸付金
117	b117	うち貸倒引当金
118	b118	流動・非流動区分フラグ
119	b119	貸倒引当金
120	b120	受取手形・売掛金
121	b121	受取手形
122	b122	売掛金
123	b123	未収入金
124	b124	未収収益
125	b125	前渡金
126	b126	前払費用
127	b127	出資金
128	b128	破産債権・更生債権
129	b129	その他長期債権・長期未収入金
130	b130	予備
131	b131	予備
132	b132	予備
133	b133	予備
134	b134	予備
135	b135	予備
136	b136	予備
137	b137	予備
138	b138	予備
139	b139	予備
140	b140	予備
141	b141	予備
142	b142	予備
143	b143	予備
144	b144	予備
145	b145	予備
146	b146	予備
147	b147	予備
148	b148	予備
149	b149	予備
150	b150	予備
151	b151	予備
152	b152	予備

項番	カラム名	YB01 （貸借対照表））
153	b153	予備
154	b154	予備
155	b155	予備
156	b156	予備
157	b157	予備
158	b158	予備
159	b159	予備
160	b160	予備
161	b161	予備
162	b162	予備
163	b163	予備
164	b164	予備
165	b165	予備
166	b166	予備
167	b167	予備
168	b168	予備
169	b169	予備
170	b170	予備
171	b171	予備
172	b172	予備
173	b173	予備
174	b174	予備
175	b175	予備
176	b176	予備
177	b177	予備
178	b178	予備
179	b179	予備
180	b180	予備
181	b181	予備
182	b182	予備
183	b183	予備
184	b184	予備
185	b185	予備
186	b186	予備
187	b187	予備
188	b188	予備
189	b189	予備
190	b190	予備
191	b191	予備

項番	カラム名	YB01 （貸借対照表)）
192	b192	予備
193	b193	予備
194	b194	予備
195	b195	予備
196	b196	予備
197	b197	予備
198	b198	予備
199	b199	予備
200	b200	予備
201	b201	予備
202	b202	予備
203	b203	予備
204	b204	予備
205	b205	予備
206	b206	予備
207	b207	予備
208	b208	予備
209	b209	予備
210	b210	予備

YC01 (貸借対照表)

表 8.6: YC01 (貸借対照表)

項番	カラム名	YC01 （貸借対照表）
1	b001	負債
2	b002	純資産
3	b003	資本金
4	b004	自己資本
5	b005	予備
6	b006	予備
7	b007	予備
8	b008	予備
9	b009	予備
10	b010	予備
11	b011	予備
12	b012	予備
13	b013	予備
14	b014	予備
15	b015	予備

項番	カラム名	YC01（貸借対照表））
16	b016	予備
17	b017	予備
18	b018	予備
19	b019	予備
20	b020	予備
21	b021	流動負債
22	b022	支払手形・買掛金
23	b023	支払手形
24	b024	買掛金
25	b025	非連結子会社関連会社　支払手形・買掛金
26	b026	短期借入金・社債合計
27	b027	１年内返済の借入金
28	b028	短期借入金
29	b029	役員・従業員短期借入金
30	b030	コマーシャル・ペーパー
31	b031	１年内返済の長期借入金
32	b032	１年内償還の社債・転換社債
33	b033	１年内償還の社債
34	b034	１年内償還の転換社債
35	b035	リース債務
36	b036	未払金・未払費用
37	b037	未払金
38	b038	未払費用
39	b039	未払賞与・給与
40	b040	未払法人税等
41	b041	未払事業所税
42	b042	未払消費税等
43	b043	デリバティブ債務
44	b044	繰延税金負債
45	b045	前受金
46	b046	預り金
47	b047	従業員預り金
48	b048	前受収益
49	b049	割賦販売未実現利益
50	b050	賞与引当金
51	b051	役員賞与引当金（未払役員報酬・賞与）
52	b052	債務保証損失引当金
53	b053	その他短期引当金
54	b054	設備関係支払手形

項番	カラム名	YC01（貸借対照表)）
55	b055	設備関係未払金
56	b056	その他流動負債
57	b057	固定負債
58	b058	長期借入金・社債・転換社債
59	b059	社債・転換社債
60	b060	社債
61	b061	転換社債
62	b062	長期借入金
63	b063	非連結子会社関連会社　長期借入金
64	b064	リース債務
65	b065	長期支払手形
66	b066	長期未払金
67	b067	引当金合計
68	b068	退職給付に係る負債（退職給付引当金）
69	b069	役員退職慰労引当金
70	b070	債務保証損失引当金
71	b071	その他長期引当金
72	b072	負ののれん
73	b073	繰延税金負債
74	b074	再評価に係る繰延税金負債
75	b075	デリバティブ債務
76	b076	繰延ヘッジ利益
77	b077	その他固定負債
78	b078	特別法上の準備金・引当金
79	b079	連結調整勘定
80	b080	為替換算調整勘定
81	b081	少数株主持分（米国会計基準）
82	b082	負債合計
83	b083	純資産
84	b084	株主資本
85	b085	資本金
86	b086	新株式申込証拠金
87	b087	資本剰余金
88	b088	資本準備金
89	b089	その他資本剰余金
90	b090	資本金及び資本準備金減少差益
91	b091	自己株式処分差益
92	b092	利益剰余金
93	b093	利益準備金

項番	カラム名	YC01（貸借対照表)）
94	b094	その他利益剰余金
95	b095	任意積立金
96	b096	繰越利益剰余金
97	b097	自己株式（▲）
98	b098	その他の包括利益累計額
99	b099	その他有価証券評価差額金
100	b100	繰延ヘッジ損益
101	b101	土地再評価差額金
102	b102	為替換算調整勘定
103	b103	新株予約権
104	b104	非支配株主持分
105	b105	負債・純資産
106	b106	自己資本
107	b107	その他の剰余金
108	b108	その他の包括利益累計額
109	b109	年金負債調整額
110	b110	資産除去債務
111	b111	資産除去債務
112	b112	未成工事受入金
113	b113	退職給付に係る調整累計額
114	b114	未払有給休暇
115	b115	繰延収益
116	b116	引当金合計
117	b117	うち有給休暇引当金
118	b118	流動・非流動区分フラグ
119	b119	買掛金
120	b120	支払手形
121	b121	前受金
122	b122	預り金
123	b123	従業員預り金
124	b124	前受収益
125	b125	未払費用
126	b126	予備
127	b127	予備
128	b128	予備
129	b129	予備
130	b130	予備
131	b131	繰延収益
132	b132	うち新株予約権

項番	カラム名	YC01（貸借対照表））
133	b133	うち為替換算調整勘定の移行日調整額
134	b134	新株予約権
135	b135	その他
136	b136	償還可能非支配持分
137	b137	うち支払手形・買掛金
138	b138	支払手形・買掛金
139	b139	中間資本
140	b140	予備
141	b141	予備
142	b142	予備
143	b143	予備
144	b144	予備
145	b145	予備
146	b146	予備
147	b147	予備
148	b148	予備
149	b149	予備
150	b150	予備
151	b151	予備
152	b152	予備
153	b153	予備
154	b154	予備
155	b155	予備
156	b156	予備
157	b157	予備
158	b158	予備
159	b159	予備
160	b160	予備
161	b161	予備
162	b162	予備
163	b163	予備
164	b164	予備
165	b165	予備
166	b166	予備
167	b167	予備
168	b168	予備
169	b169	予備
170	b170	予備
171	b171	予備

項番	カラム名	YC01（貸借対照表））
172	b172	予備
173	b173	予備
174	b174	予備
175	b175	予備
176	b176	予備
177	b177	予備
178	b178	予備
179	b179	予備
180	b180	予備
181	b181	予備
182	b182	予備
183	b183	予備
184	b184	予備
185	b185	予備
186	b186	予備
187	b187	予備
188	b188	予備
189	b189	予備
190	b190	予備
191	b191	予備
192	b192	予備
193	b193	予備
194	b194	予備
195	b195	予備
196	b196	予備
197	b197	予備
198	b198	予備
199	b199	予備
200	b200	予備
201	b201	予備
202	b202	予備
203	b203	予備
204	b204	予備
205	b205	予備
206	b206	予備
207	b207	予備
208	b208	予備
209	b209	予備
210	b210	予備

YD01 (損益計算書 (累計))

表 8.7: YD01 (損益計算書 (累計))

項番	カラム名	YD01 (損益計算書 (累計))
1	b001	売上高・営業収益
2	b002	営業利益
3	b003	経常利益
4	b004	親会社株主に帰属する当期純利益 (連結) ／当期利益 (単独)
5	b005	売上高・営業収益 (短信サマリー)
6	b006	予備
7	b007	予備
8	b008	予備
9	b009	予備
10	b010	予備
11	b011	予備
12	b012	予備
13	b013	予備
14	b014	予備
15	b015	予備
16	b016	予備
17	b017	予備
18	b018	予備
19	b019	予備
20	b020	予備
21	b021	売上高・営業収益
22	b022	うち金融収益
23	b023	営業費用
24	b024	売上原価・営業原価
25	b025	割賦販売未実現利益・返品調整引当金差額
26	b026	売上総利益
27	b027	販売費および一般管理費
28	b028	金融費用
29	b029	営業利益
30	b030	営業外収益
31	b031	受取利息・配当金
32	b032	受取利息・割引料・有価証券利息
33	b033	受取配当金
34	b034	有価証券売却益
35	b035	有価証券評価益
36	b036	デリバティブ評価益

項番	カラム名	YD01 （損益計算書（累計））
37	b037	その他資産処分益・評価益
38	b038	その他資産処分益
39	b039	その他資産評価益
40	b040	為替差益
41	b041	負ののれん償却額
42	b042	持分法による投資利益
43	b043	投資事業組合・匿名組合関連利益
44	b044	賃貸料収入
45	b045	その他営業外収益
46	b046	営業外費用
47	b047	支払利息・割引料
48	b048	うち社債利息
49	b049	うちコマーシャル・ペーパー利息
50	b050	うち手形売却損
51	b051	うち売上割引
52	b052	社債発行費・社債発行差金償却
53	b053	有価証券売却損
54	b054	有価証券評価損
55	b055	デリバティブ評価損
56	b056	その他資産処分損・評価損
57	b057	その他資産処分損
58	b058	その他資産評価損
59	b059	為替差損
60	b060	租税公課
61	b061	のれん償却額
62	b062	持分法による投資損失
63	b063	投資事業組合・匿名組合関連損失
64	b064	賃貸収入原価
65	b065	その他営業外費用
66	b066	経常利益
67	b067	特別利益
68	b068	事業・組織再編関連利益
69	b069	有価証券売却益
70	b070	有価証券評価益
71	b071	その他資産処分益・評価益
72	b072	有形固定資産処分益・評価益
73	b073	うち不動産処分益・評価益
74	b074	有形固定資産以外のその他資産処分益・評価益
75	b075	為替差益

項番	カラム名	YD01（損益計算書（累計））
76	b076	退職給付関連利益
77	b077	持分変動利益
78	b078	債務免除益
79	b079	引当金・準備金戻入・取崩額
80	b080	その他特別利益
81	b081	特別損失
82	b082	事業・組織再編関連損
83	b083	減損損失
84	b084	有価証券売却損
85	b085	有価証券評価損
86	b086	その他資産処分損・評価損
87	b087	有形固定資産処分損・評価損
88	b088	うち不動産処分損・評価損
89	b089	有形固定資産以外のその他資産処分損・評価損
90	b090	為替差損
91	b091	減価償却費
92	b092	退職給付関連費用
93	b093	持分変動損失
94	b094	引当金・準備金繰入額
95	b095	その他特別損失
96	b096	匿名組合損益分配額
97	b097	税金等調整前当期純利益
98	b098	特別法上の準備金・引当金取崩額
99	b099	特別法上の準備金・引当金繰入額
100	b100	税金等調整前当期利益
101	b101	法人税等
102	b102	法人税・住民税及び事業税合計
103	b103	法人税等調整額
104	b104	過年度法人税等追徴・還付額
105	b105	非支配株主に帰属する当期純利益
106	b106	旧：連結調整勘定償却額
107	b107	持分法投資損益（米国会計基準）
108	b108	為替換算調整
109	b109	その他の当期利益影響額
110	b110	親会社株主に帰属する当期純利益（連結）／当期利益（単独）
111	b111	売上高・営業収益（米国基準/IFRS 基準）
112	b112	売上高・営業収益（日本基準）
113	b113	負ののれん発生益
114	b114	当期純利益

項番	カラム名	YD01（損益計算書（累計））
115	b115	その他有価証券評価差額金：当期発生額
116	b116	その他有価証券評価差額金：組替調整額
117	b117	その他有価証券評価差額金：税効果調整前
118	b118	その他有価証券評価差額金：税効果額
119	b119	その他有価証券評価差額金：税効果調整後
120	b120	繰延ヘッジ損益：当期発生額
121	b121	繰延ヘッジ損益：組替調整額
122	b122	繰延ヘッジ損益：税効果調整前
123	b123	繰延ヘッジ損益：税効果額
124	b124	繰延ヘッジ損益：税効果調整後
125	b125	為替換算調整勘定：当期発生額
126	b126	為替換算調整勘定：組替調整額
127	b127	為替換算調整勘定：税効果調整前
128	b128	為替換算調整勘定：税効果額
129	b129	為替換算調整勘定：税効果調整後
130	b130	年金負債調整額：税効果調整前
131	b131	年金負債調整額：税効果額
132	b132	年金負債調整額：税効果調整後
133	b133	固定資産再評価差額金：税効果調整前
134	b134	固定資産再評価差額金：税効果額
135	b135	固定資産再評価差額金：税効果調整後
136	b136	持分法適用会社に対する持分相当額：当期発生額
137	b137	その他の包括利益合計：税効果調整前
138	b138	その他の包括利益合計：税効果額
139	b139	その他の包括利益合計：税効果調整後
140	b140	包括利益
141	b141	親会社株主に係る包括利益
142	b142	非支配株主に係る包括利益
143	b143	収益合計（性質別分類）
144	b144	費用合計（性質別分類）
145	b145	継続事業からの純利益
146	b146	非継続事業からの純利益
147	b147	機能別/性質別分類フラグ
148	b148	研究開発費（機能別分類）
149	b149	金融収益
150	b150	金融費用
151	b151	その他の収益
152	b152	その他の費用
153	b153	その他の収益

項番	カラム名	YD01（損益計算書（累計））
154	b154	その他の費用
155	b155	持分法による投資利益
156	b156	持分法による投資損失
157	b157	為替差益
158	b158	為替差損
159	b159	その他
160	b160	その他
161	b161	その他金融収益
162	b162	その他金融費用
163	b163	受取利息・配当金
164	b164	受取利息及び割引料
165	b165	受取配当金
166	b166	支払利息・割引料
167	b167	うち売上割引
168	b168	為替差益
169	b169	為替差損
170	b170	その他
171	b171	その他
172	b172	予備
173	b173	予備
174	b174	予備
175	b175	予備
176	b176	予備
177	b177	退職給付に係る調整額：税効果調整前
178	b178	退職給付に係る調整額：税効果額
179	b179	退職給付に係る調整額：税効果調整後
180	b180	退職給付に係る純利息収益
181	b181	退職給付に係る純利息費用
182	b182	予備
183	b183	予備
184	b184	予備
185	b185	予備
186	b186	予備
187	b187	予備
188	b188	予備
189	b189	予備
190	b190	予備
191	b191	予備
192	b192	予備

項番	カラム名	YD01 （損益計算書（累計））
193	b193	予備
194	b194	予備
195	b195	予備
196	b196	予備
197	b197	予備
198	b198	予備
199	b199	予備
200	b200	予備
201	b201	予備
202	b202	予備
203	b203	予備
204	b204	予備
205	b205	予備
206	b206	予備
207	b207	予備
208	b208	予備
209	b209	予備
210	b210	予備

YE01 (損益計算書 (3 ヵ月))

表 8.8: YE01 (損益計算書 (3 ヵ月))

項番	カラム名	YE01 （損益計算書（３ヵ月））
1	b001	売上高・営業収益
2	b002	営業利益
3	b003	経常利益
4	b004	親会社株主に帰属する当期純利益（連結）／当期利益（単独）
5	b005	予備
6	b006	予備
7	b007	予備
8	b008	予備
9	b009	予備
10	b010	予備
11	b011	予備
12	b012	予備
13	b013	予備
14	b014	予備
15	b015	予備
16	b016	予備

項番	カラム名	YE01（損益計算書（３ヵ月））
17	b017	予備
18	b018	予備
19	b019	予備
20	b020	予備
21	b021	売上高・営業収益
22	b022	うち金融収益
23	b023	営業費用
24	b024	売上原価・営業原価
25	b025	割賦販売未実現利益・返品調整引当金差額
26	b026	売上総利益
27	b027	販売費および一般管理費
28	b028	金融費用
29	b029	営業利益
30	b030	営業外収益
31	b031	受取利息・配当金
32	b032	受取利息・割引料・有価証券利息
33	b033	受取配当金
34	b034	有価証券売却益
35	b035	有価証券評価益
36	b036	デリバティブ評価益
37	b037	その他資産処分益・評価益
38	b038	その他資産処分益
39	b039	その他資産評価益
40	b040	為替差益
41	b041	負ののれん償却額
42	b042	持分法による投資利益
43	b043	投資事業組合・匿名組合関連利益
44	b044	賃貸料収入
45	b045	その他営業外収益
46	b046	営業外費用
47	b047	支払利息・割引料
48	b048	うち社債利息
49	b049	うちコマーシャル・ペーパー利息
50	b050	うち手形売却損
51	b051	うち売上割引
52	b052	社債発行費・社債発行差金償却
53	b053	有価証券売却損
54	b054	有価証券評価損
55	b055	デリバティブ評価損

項番	カラム名	YE01（損益計算書（３ヵ月））
56	b056	その他資産処分損・評価損
57	b057	その他資産処分損
58	b058	その他資産評価損
59	b059	為替差損
60	b060	租税公課
61	b061	のれん償却額
62	b062	持分法による投資損失
63	b063	投資事業組合・匿名組合関連損失
64	b064	賃貸収入原価
65	b065	その他営業外費用
66	b066	経常利益
67	b067	特別利益
68	b068	事業・組織再編関連利益
69	b069	有価証券売却益
70	b070	有価証券評価益
71	b071	その他資産処分益・評価益
72	b072	有形固定資産処分益・評価益
73	b073	うち不動産処分益・評価益
74	b074	有形固定資産以外のその他資産処分益・評価益
75	b075	為替差益
76	b076	退職給付関連利益
77	b077	持分変動利益
78	b078	債務免除益
79	b079	引当金・準備金戻入・取崩額
80	b080	その他特別利益
81	b081	特別損失
82	b082	事業・組織再編関連損
83	b083	減損損失
84	b084	有価証券売却損
85	b085	有価証券評価損
86	b086	その他資産処分損・評価損
87	b087	有形固定資産処分損・評価損
88	b088	うち不動産処分損・評価損
89	b089	有形固定資産以外のその他資産処分損・評価損
90	b090	為替差損
91	b091	減価償却費
92	b092	退職給付関連費用
93	b093	持分変動損失
94	b094	引当金・準備金繰入額

項番	カラム名	YE01（損益計算書（３ヵ月））
95	b095	その他特別損失
96	b096	匿名組合損益分配額
97	b097	税金等調整前当期純利益
98	b098	特別法上の準備金・引当金取崩額
99	b099	特別法上の準備金・引当金繰入額
100	b100	税金等調整前当期利益
101	b101	法人税等
102	b102	法人税・住民税及び事業税合計
103	b103	法人税等調整額
104	b104	過年度法人税等追徴・還付額
105	b105	非支配株主に帰属する当期純利益
106	b106	旧：連結調整勘定償却額
107	b107	持分法投資損益（米国会計基準）
108	b108	為替換算調整
109	b109	その他の当期利益影響額
110	b110	親会社株主に帰属する当期純利益（連結）／当期利益（単独）
111	b111	売上高・営業収益（米国基準/IFRS 基準）
112	b112	売上高・営業収益（日本基準）
113	b113	負ののれん発生益
114	b114	当期純利益
115	b115	その他有価証券評価差額金：当期発生額
116	b116	その他有価証券評価差額金：組替調整額
117	b117	その他有価証券評価差額金：税効果調整前
118	b118	その他有価証券評価差額金：税効果額
119	b119	その他有価証券評価差額金：税効果調整後
120	b120	繰延ヘッジ損益：当期発生額
121	b121	繰延ヘッジ損益：組替調整額
122	b122	繰延ヘッジ損益：税効果調整前
123	b123	繰延ヘッジ損益：税効果額
124	b124	繰延ヘッジ損益：税効果調整後
125	b125	為替換算調整勘定：当期発生額
126	b126	為替換算調整勘定：組替調整額
127	b127	為替換算調整勘定：税効果調整前
128	b128	為替換算調整勘定：税効果額
129	b129	為替換算調整勘定：税効果調整後
130	b130	年金負債調整額：税効果調整前
131	b131	年金負債調整額：税効果額
132	b132	年金負債調整額：税効果調整後
133	b133	固定資産再評価差額金：税効果調整前

項番	カラム名	YE01（損益計算書（３ヵ月））
134	b134	固定資産再評価差額金：税効果額
135	b135	固定資産再評価差額金：税効果調整後
136	b136	持分法適用会社に対する持分相当額：当期発生額
137	b137	その他の包括利益合計：税効果調整前
138	b138	その他の包括利益合計：税効果額
139	b139	その他の包括利益合計：税効果調整後
140	b140	包括利益
141	b141	親会社株主に係る包括利益
142	b142	非支配株主に係る包括利益
143	b143	収益合計（性質別分類）
144	b144	費用合計（性質別分類）
145	b145	継続事業からの純利益
146	b146	非継続事業からの純利益
147	b147	機能別/性質別分類フラグ
148	b148	研究開発費（機能別分類）
149	b149	金融収益
150	b150	金融費用
151	b151	その他の収益
152	b152	その他の費用
153	b153	その他の収益
154	b154	その他の費用
155	b155	持分法による投資利益
156	b156	持分法による投資損失
157	b157	為替差益
158	b158	為替差損
159	b159	その他
160	b160	その他
161	b161	その他金融収益
162	b162	その他金融費用
163	b163	受取利息・配当金
164	b164	受取利息及び割引料
165	b165	受取配当金
166	b166	支払利息・割引料
167	b167	うち売上割引
168	b168	為替差益
169	b169	為替差損
170	b170	その他
171	b171	その他
172	b172	予備

項番	カラム名	YE01（損益計算書（３ヵ月））
173	b173	予備
174	b174	予備
175	b175	予備
176	b176	予備
177	b177	退職給付に係る調整額：税効果調整前
178	b178	退職給付に係る調整額：税効果額
179	b179	退職給付に係る調整額：税効果調整後
180	b180	退職給付に係る純利息収益
181	b181	退職給付に係る純利息費用
182	b182	予備
183	b183	予備
184	b184	予備
185	b185	予備
186	b186	予備
187	b187	予備
188	b188	予備
189	b189	予備
190	b190	予備
191	b191	予備
192	b192	予備
193	b193	予備
194	b194	予備
195	b195	予備
196	b196	予備
197	b197	予備
198	b198	予備
199	b199	予備
200	b200	予備
201	b201	予備
202	b202	予備
203	b203	予備
204	b204	予備
205	b205	予備
206	b206	予備
207	b207	予備
208	b208	予備
209	b209	予備
210	b210	予備

YF01 (キャッシュフロー計算書)

表 8.9: YF01 (キャッシュフロー計算書)

項番	カラム名	YF01（キャッシュフロー計算書）
1	b001	営業活動によるキャッシュフロー
2	b002	投資活動によるキャッシュフロー
3	b003	財務活動によるキャッシュ・フロー
4	b004	現金および現金同等物の期末残高
5	b005	予備
6	b006	予備
7	b007	予備
8	b008	予備
9	b009	予備
10	b010	予備
11	b011	予備
12	b012	予備
13	b013	予備
14	b014	予備
15	b015	予備
16	b016	予備
17	b017	予備
18	b018	予備
19	b019	予備
20	b020	予備
21	b021	税金等調整前当期純利益
22	b022	減価償却費
23	b023	その他の償却費
24	b024	減損損失
25	b025	繰延税金
26	b026	有価証券および投資有価証券売却損益（▲売却益）
27	b027	有価証券評価損益（▲評価益）
28	b028	デリバティブ評価損益（▲評価益）
29	b029	固定資産売却損益（▲売却益）
30	b030	固定資産除却損益（▲除却益）
31	b031	固定資産評価損益（▲評価益）
32	b032	その他の評価損益（▲評価益）
33	b033	社債発行差金償却額
34	b034	のれん・負ののれん償却額
35	b035	貸倒引当金の増加額（▲減少額）
36	b036	退職給付及び役員退職慰労引当金の増減額（▲減少額）

項番	カラム名	YF01（キャッシュフロー計算書）
37	b037	賞与引当金の増加額（▲減少額）
38	b038	その他の引当金の増加額（▲減少額）
39	b039	受取利息および受取配当金（▲）
40	b040	支払利息
41	b041	為替差損（▲為替差益）
42	b042	非支配株主損益
43	b043	持分法による投資損益（▲利益）
44	b044	損害賠償損失（▲利益）
45	b045	事業・組織再編関連損益（▲利益）
46	b046	売上債権の減少額（▲増加額）
47	b047	棚卸資産の減少額（▲増加額）
48	b048	仕入債務の増加額（▲減少額）
49	b049	未払い消費税等の増加額（▲減少額）
50	b050	割引手形の増加額（▲減少額）
51	b051	その他の流動資産の減少額（▲増加額）
52	b052	その他の流動負債の増加額（▲減少額）
53	b053	役員賞与の支払額（▲）
54	b054	その他の小計欄より上のキャッシュフロー
55	b055	営業収入
56	b056	原材料及び商品の仕入支出（▲）
57	b057	人件費支出（▲）
58	b058	その他の営業支出（▲）
59	b059	小計
60	b060	利息及び配当金の受取額
61	b061	利息の支払額（▲）
62	b062	法人税等の支払額（▲）
63	b063	損害賠償金の支払額（▲）
64	b064	その他の小計欄以降の営業キャッシュフロー
65	b065	営業活動によるキャッシュフロー
66	b066	日経調整営業キャッシュフロー
67	b067	定期預金の預入による支出（▲）
68	b068	定期預金の払戻による収入
69	b069	固定資産の取得による支出（▲）
70	b070	うち有形固定資産の取得による支出（▲）
71	b071	うち無形固定資産の取得による支出（▲）
72	b072	うち投資その他の資産の取得による支出（▲）
73	b073	固定資産の売却による収入
74	b074	うち有形固定資産の売却による収入
75	b075	うち無形固定資産の売却による収入

項番	カラム名	YF01（キャッシュフロー計算書）
76	b076	うち投資その他の資産の売却による収入
77	b077	有価証券の取得による支出（▲）
78	b078	有価証券の売却による収入
79	b079	投資有価証券の取得による支出（▲）
80	b080	投資有価証券の売却による収入
81	b081	子会社・関係会社株式取得（▲）
82	b082	子会社・関係会社株式売却
83	b083	貸付金の増加による支出（▲）
84	b084	貸付金の回収による収入
85	b085	利息及び配当金の受取額
86	b086	その他の投資活動によるキャッシュフロー
87	b087	投資活動によるキャッシュフロー
88	b088	日経調整投資キャッシュフロー
89	b089	短期借入金による収入
90	b090	短期借入金の返済による支出（▲）
91	b091	リース債務の返済（▲）
92	b092	コマーシャルペーパーによる調達額
93	b093	コマーシャルペーパー返済額（▲）
94	b094	長期借入金による収入
95	b095	長期借入金の返済による支出（▲）
96	b096	社債の発行による収入
97	b097	社債の償還による支出（▲）
98	b098	株式の発行による収入
99	b099	自己株式の取得による支出（▲）
100	b100	自己株式の処分による収入
101	b101	利息の支払金額（▲）
102	b102	配当金の支払金額（▲）
103	b103	非支配株主からの払い込みによる収入
104	b104	非支配株主への配当金の支払額（▲）
105	b105	その他の財務活動によるキャッシュフロー
106	b106	財務活動によるキャッシュ・フロー
107	b107	日経調整財務キャッシュフロー
108	b108	現金および現金同等物に係る換算差額
109	b109	現金および現金同等物の増加額（▲減少額）
110	b110	現金および現金同等物の期首残高
111	b111	その他の変更による影響額
112	b112	現金および現金同等物の期末残高
113	b113	現金および預金
114	b114	預入期間が３カ月を超える定期預金（▲）

項番	カラム名	YF01（キャッシュフロー計算書）
115	b115	コマーシャルペーパー
116	b116	譲渡性預金
117	b117	売り戻し条件付き現先
118	b118	公社債投信
119	b119	その他現金および現金同等物
120	b120	直接法間接法区分フラグ
121	b121	予備
122	b122	利息・配当金記載フラグ
123	b123	予備
124	b124	予備
125	b125	予備
126	b126	予備
127	b127	予備
128	b128	予備
129	b129	予備
130	b130	予備
131	b131	予備
132	b132	予備
133	b133	予備
134	b134	予備
135	b135	予備
136	b136	負ののれん発生益
137	b137	予備
138	b138	予備
139	b139	予備
140	b140	予備
141	b141	予備
142	b142	予備
143	b143	予備
144	b144	予備
145	b145	予備
146	b146	予備
147	b147	予備
148	b148	予備
149	b149	予備
150	b150	予備
151	b151	予備
152	b152	予備
153	b153	予備

項番	カラム名	YF01（キャッシュフロー計算書）
154	b154	予備
155	b155	予備
156	b156	予備
157	b157	予備
158	b158	予備
159	b159	予備
160	b160	予備
161	b161	予備
162	b162	予備
163	b163	予備
164	b164	予備
165	b165	予備
166	b166	予備
167	b167	予備
168	b168	予備
169	b169	予備
170	b170	予備
171	b171	予備
172	b172	予備
173	b173	予備
174	b174	予備
175	b175	予備
176	b176	予備
177	b177	予備
178	b178	予備
179	b179	予備
180	b180	予備
181	b181	予備
182	b182	予備
183	b183	予備
184	b184	予備
185	b185	予備
186	b186	予備
187	b187	予備
188	b188	予備
189	b189	予備
190	b190	予備
191	b191	予備
192	b192	予備

項番	カラム名	YF01（キャッシュフロー計算書）
193	b193	予備
194	b194	予備
195	b195	予備
196	b196	予備
197	b197	予備
198	b198	予備
199	b199	予備
200	b200	予備
201	b201	予備
202	b202	予備
203	b203	予備
204	b204	予備
205	b205	予備
206	b206	予備
207	b207	予備
208	b208	予備
209	b209	予備
210	b210	予備

YG01 (株主資本等変動計算書)

表 8.10: YG01 (株主資本等変動計算書)

項番	カラム名	YG01（株主資本等変動計算書）
1	b001	【資本金】当期首残高
2	b002	新株の発行
3	b003	資本金から資本準備金またはその他資本剰余金への振替
4	b004	資本準備金から資本金への振替
5	b005	その他資本剰余金から資本金への振替
6	b006	企業結合または会社分割による増減
7	b007	連結範囲または持分法適用範囲の変動による増減
8	b008	その他の資本金増減
9	b009	当期変動額合計
10	b010	当期末残高
11	b011	【資本準備金】当期首残高
12	b012	新株の発行
13	b013	資本金から資本準備金への振替
14	b014	資本準備金から資本金またはその他資本剰余金への振替
15	b015	その他資本剰余金から資本準備金への振替
16	b016	企業結合または会社分割による増減

項番	カラム名	YG01（株主資本等変動計算書）
17	b017	その他の資本準備金増減
18	b018	当期変動額合計
19	b019	当期末残高
20	b020	【その他資本剰余金】当期首残高
21	b021	剰余金の配当
22	b022	自己株式の処分
23	b023	自己株式の消却
24	b024	資本金からその他資本剰余金への振替
25	b025	資本準備金からその他資本剰余金への振替
26	b026	その他資本剰余金から資本金または資本準備金への振替
27	b027	資本剰余金から利益剰余金への振替
28	b028	企業結合または会社分割による増減
29	b029	その他のその他資本剰余金増減
30	b030	当期変動額合計
31	b031	当期末残高
32	b032	【資本剰余金合計】当期首残高
33	b033	新株の発行
34	b034	剰余金の配当
35	b035	自己株式の処分
36	b036	自己株式の消却
37	b037	資本金から資本準備金またはその他資本剰余金への振替
38	b038	資本準備金から資本金への振替
39	b039	その他資本剰余金から資本金への振替
40	b040	資本剰余金から利益剰余金への振替
41	b041	企業結合または会社分割による増減
42	b042	連結範囲または持分法適用範囲の変動による増減
43	b043	その他の資本剰余金増減
44	b044	当期変動額合計
45	b045	当期末残高
46	b046	【利益準備金】当期首残高
47	b047	剰余金の配当
48	b048	利益準備金からその他利益剰余金への振替
49	b049	その他利益剰余金から利益準備金への振替
50	b050	企業結合または会社分割による増減
51	b051	その他の利益準備金増減
52	b052	当期変動額合計
53	b053	当期末残高
54	b054	【積立金】当期首残高
55	b055	剰余金の配当

項番	カラム名	YG01（株主資本等変動計算書）
56	b056	企業結合または会社分割による増減
57	b057	その他の積立金増減
58	b058	当期変動額合計
59	b059	当期末残高
60	b060	【繰越利益剰余金】当期首残高
61	b061	剰余金の配当
62	b062	親会社株主に帰属する当期純利益／当期純利益
63	b063	自己株式の処分
64	b064	自己株式の消却
65	b065	利益準備金からその他利益剰余金への振替
66	b066	その他利益剰余金から利益準備金への振替
67	b067	資本剰余金から利益剰余金への振替
68	b068	企業結合または会社分割による増減
69	b069	その他の繰越利益剰余金増減
70	b070	当期変動額合計
71	b071	当期末残高
72	b072	【利益剰余金合計】当期首残高
73	b073	剰余金の配当
74	b074	親会社株主に帰属する当期純利益／当期純利益
75	b075	自己株式の処分
76	b076	自己株式の消却
77	b077	資本剰余金から利益剰余金への振替
78	b078	企業結合または会社分割による増減
79	b079	連結範囲または持分法適用範囲の変動による増減
80	b080	その他の利益剰余金増減
81	b081	当期変動額合計
82	b082	当期末残高
83	b083	【その他の包括利益累計額】当期首残高
84	b084	当期変動額合計
85	b085	当期末残高
86	b086	【自己株式】当期首残高
87	b087	自己株式の取得
88	b088	自己株式の処分
89	b089	自己株式の消却
90	b090	企業結合または会社分割による増減
91	b091	連結範囲または持分法適用範囲の変動による増減
92	b092	その他の自己株式増減
93	b093	当期変動額合計
94	b094	当期末残高

項番	カラム名	YG01（株主資本等変動計算書）
95	b095	【株主資本合計】当期首残高
96	b096	新株の発行
97	b097	剰余金の配当
98	b098	親会社株主に帰属する当期純利益／当期純利益
99	b099	自己株式の取得
100	b100	自己株式の処分
101	b101	自己株式の消却
102	b102	企業結合または会社分割による増減
103	b103	連結範囲または持分法適用範囲の変動による増減
104	b104	その他の株主資本増減
105	b105	当期変動額合計
106	b106	当期末残高
107	b107	【その他有価証券評価差額金】当期首残高
108	b108	当期変動額合計
109	b109	当期末残高
110	b110	【繰延ヘッジ損益】当期首残高
111	b111	当期変動額合計
112	b112	当期末残高
113	b113	【為替換算調整勘定】当期首残高
114	b114	当期変動額合計
115	b115	当期末残高
116	b116	【土地再評価差額金】当期首残高
117	b117	当期変動額合計
118	b118	当期末残高
119	b119	【評価・換算差額等/その他の包括利益累計額】当期首残高
120	b120	当期変動額合計
121	b121	当期末残高
122	b122	【新株予約権】当期首残高
123	b123	当期変動額合計
124	b124	当期末残高
125	b125	【非支配株主持分】当期首残高
126	b126	当期変動額合計
127	b127	当期末残高
128	b128	【純資産合計】当期首残高
129	b129	新株の発行
130	b130	剰余金の配当
131	b131	親会社株主に帰属する当期純利益／当期純利益
132	b132	自己株式の取得
133	b133	自己株式の処分

項番	カラム名	YG01（株主資本等変動計算書）
134	b134	自己株式の消却
135	b135	企業結合または会社分割による増減
136	b136	連結範囲または持分法適用範囲の変動による増減
137	b137	その他の純資産増減
138	b138	当期変動額合計
139	b139	当期末残高
140	b140	【退職給付に係る調整累計額】当期首残高
141	b141	当期変動額合計
142	b142	当期末残高
143	b143	【資本剰余金合計】自己株式の取得
144	b144	【資本剰余金合計】自己株式取引による増減
145	b145	【利益剰余金合計】その他の包括利益
146	b146	【利益剰余金合計】自己株式の取得
147	b147	【利益剰余金合計】自己株式取引による増減
148	b148	【利益剰余金合計】利益剰余金・その他の資本の構成要素間の振替
149	b149	【自己株式】自己株式取引による増減
150	b150	【累積その他の包括利益】その他の包括利益
151	b151	自己株式の処分
152	b152	自己株式取引による増減
153	b153	利益剰余金・その他の資本の構成要素間の振替
154	b154	その他の増減
155	b155	【親会社の所有者に帰属する資本】その他の包括利益
156	b156	【親会社の所有者に帰属する資本】自己株式取引による増減
157	b157	【資本】その他の包括利益
158	b158	【資本】自己株式取引による増減
159	b159	予備
160	b160	予備
161	b161	予備
162	b162	予備
163	b163	予備
164	b164	予備
165	b165	予備
166	b166	予備
167	b167	予備
168	b168	予備
169	b169	予備
170	b170	予備
171	b171	予備
172	b172	予備

項番	カラム名	YG01（株主資本等変動計算書）
173	b173	予備
174	b174	予備
175	b175	予備
176	b176	予備
177	b177	予備
178	b178	予備
179	b179	予備
180	b180	予備
181	b181	予備
182	b182	予備
183	b183	予備
184	b184	予備
185	b185	予備
186	b186	予備
187	b187	予備
188	b188	予備
189	b189	予備
190	b190	予備
191	b191	予備
192	b192	予備
193	b193	予備
194	b194	予備
195	b195	予備
196	b196	予備
197	b197	予備
198	b198	予備
199	b199	予備
200	b200	予備
201	b201	予備
202	b202	予備
203	b203	予備
204	b204	予備
205	b205	予備
206	b206	予備
207	b207	予備
208	b208	予備
209	b209	予備
210	b210	予備

YH01 (その他・明細情報等)

表 8.11: YH01 (その他・明細情報等)

項番	カラム名	YH01（その他・明細情報等）
1	b001	受取手形割引高
2	b002	受取手形裏書譲渡高
3	b003	貸倒引当金（欄外注記分）
4	b004	税効果会計の対象となった繰越欠損金
5	b005	減価償却実施額（有形無形その他の合計）≪累計≫
6	b006	減価償却実施額（有形無形その他の合計）≪３カ月≫
7	b007	うち有形固定資産減価償却実施額≪累計≫
8	b008	うち有形固定資産減価償却実施額≪３カ月≫
9	b009	減損損失（有形無形その他の合計）
10	b010	うち有形固定資産減損損失
11	b011	減価償却累計額・減損損失累計額控除前（有形無形その他合計）
12	b012	うち有形固定資産減価償却累計額・減損損失累計額控除前
13	b013	減損損失累計額（有形無形その他の合計）
14	b014	うち有形固定資産減損損失累計額
15	b015	繰延資産償却額
16	b016	海外・輸出売上高≪累計≫
17	b017	海外・輸出売上高≪３カ月≫
18	b018	当期受注高≪累計≫
19	b019	当期受注高≪３カ月≫
20	b020	期末受注残高
21	b021	期末従業員数
22	b022	偶発債務合計
23	b023	うち保証債務等合計
24	b024	保証債務
25	b025	保証予約
26	b026	保証類似行為（経営指導念書等差入れ）
27	b027	会計方針の変更に伴う売上高修正額合計
28	b028	会計方針の変更に伴う経常損益修正額合計
29	b029	会計方針の変更に伴う税引前損益修正額合計
30	b030	継続企業の前提に関する注記（有無）
31	b031	継続企業の前提に関する注記（内容）
32	b032	監査意見フラグ
33	b033	研究開発費
34	b034	設備投資額
35	b035	主要な所有土地の簿価
36	b036	主要な所有土地の面積

項番	カラム名	YH01（その他・明細情報等）
37	b037	担保付借入金・社債合計
38	b038	担保資産合計
39	b039	役員賞与・役員賞与引当金繰入
40	b040	土地再評価後の簿価価額との差額
41	b041	研究開発費≪３カ月≫
42	b042	設備投資額≪３カ月≫
43	b043	予備
44	b044	予備
45	b045	予備
46	b046	予備
47	b047	予備
48	b048	予備
49	b049	予備
50	b050	予備
51	b051	予備
52	b052	予備
53	b053	予備
54	b054	予備
55	b055	予備
56	b056	予備
57	b057	予備
58	b058	予備
59	b059	予備
60	b060	予備
61	b061	予備
62	b062	予備
63	b063	予備
64	b064	予備
65	b065	予備
66	b066	予備
67	b067	予備
68	b068	予備
69	b069	予備
70	b070	予備
71	b071	予備
72	b072	予備
73	b073	予備
74	b074	予備
75	b075	予備

項番	カラム名	YH01（その他・明細情報等）
76	b076	予備
77	b077	予備
78	b078	予備
79	b079	予備
80	b080	予備
81	b081	予備
82	b082	予備
83	b083	予備
84	b084	予備
85	b085	予備
86	b086	予備
87	b087	予備
88	b088	予備
89	b089	予備
90	b090	予備
91	b091	予備
92	b092	予備
93	b093	予備
94	b094	予備
95	b095	予備
96	b096	予備
97	b097	予備
98	b098	予備
99	b099	予備
100	b100	予備
101	b101	予備
102	b102	予備
103	b103	予備
104	b104	予備
105	b105	予備
106	b106	予備
107	b107	予備
108	b108	予備
109	b109	予備
110	b110	予備
111	b111	予備
112	b112	予備
113	b113	予備
114	b114	予備

項番	カラム名	YH01（その他・明細情報等）
115	b115	予備
116	b116	予備
117	b117	予備
118	b118	予備
119	b119	予備
120	b120	予備
121	b121	予備
122	b122	予備
123	b123	予備
124	b124	予備
125	b125	予備
126	b126	予備
127	b127	予備
128	b128	予備
129	b129	予備
130	b130	予備
131	b131	予備
132	b132	予備
133	b133	予備
134	b134	予備
135	b135	予備
136	b136	予備
137	b137	予備
138	b138	予備
139	b139	予備
140	b140	予備
141	b141	予備
142	b142	予備
143	b143	予備
144	b144	予備
145	b145	予備
146	b146	予備
147	b147	予備
148	b148	予備
149	b149	予備
150	b150	予備
151	b151	予備
152	b152	予備
153	b153	予備

項番	カラム名	YH01（その他・明細情報等）
154	b154	予備
155	b155	予備
156	b156	予備
157	b157	予備
158	b158	予備
159	b159	予備
160	b160	予備
161	b161	予備
162	b162	予備
163	b163	予備
164	b164	予備
165	b165	内部統制報告書の評価フラグ
166	b166	賃貸等不動産残高
167	b167	賃貸等不動産時価
168	b168	ストックオプション未確定残
169	b169	ストックオプション未行使残
170	b170	平均臨時従業員数
171	b171	法人税等負担率
172	b172	監査報告書提出日
173	b173	連結納税制度採用フラグ
174	b174	のれん・負ののれん償却額
175	b175	契約総額
176	b176	うち当座貸越
177	b177	うちコミットメントライン
178	b178	借入実行残高総額
179	b179	うち当座貸越
180	b180	うちコミットメントライン
181	b181	借入未実行残高
182	b182	うち当座貸越
183	b183	うちコミットメントライン
184	b184	ファイナンス・リース 将来最低支払リース料の現在価値（借手側）合計額
185	b185	ファイナンス・リース 将来最低支払リース料の現在価値（借手側）1 年内
186	b186	ファイナンス・リース 将来最低支払リース料の現在価値（借手側）1 年超
187	b187	ファイナンス・リース 将来最低受取リース料の現在価値（貸手側）合計額
188	b188	ファイナンス・リース 将来最低受取リース料の現在価値（貸手側）1 年内
189	b189	ファイナンス・リース 将来最低受取リース料の現在価値（貸手側）1 年超
190	b190	期首における退職給付に係る負債
191	b191	退職給付費用
192	b192	退職給付の支払額

項番	カラム名	YH01（その他・明細情報等）
193	b193	制度への拠出額
194	b194	その他
195	b195	期末における退職給付に係る負債
196	b196	【年金資産の内訳】株式（金額）
197	b197	【年金資産の内訳】債券（金額）
198	b198	【年金資産の内訳】一般勘定（金額）
199	b199	【年金資産の内訳】その他（金額）
200	b200	【年金資産調整表】期首における年金資産
201	b201	【年金資産調整表】　期待運用収益
202	b202	【年金資産調整表】　数理計算上の差異の当期発生額
203	b203	【年金資産調整表】　事業主からの拠出額
204	b204	【年金資産調整表】　退職給付の額
205	b205	【年金資産調整表】　その他
206	b206	【年金資産調整表】期末における年金資産
207	b207	【年金資産の内訳】株式（割合）
208	b208	【年金資産の内訳】債券（割合）
209	b209	【年金資産の内訳】一般勘定（割合）
210	b210	【年金資産の内訳】その他（割合）

YI01 (その他・明細情報等)

表 8.12: YI01 (その他・明細情報等)

項番	カラム名	YI01（その他・明細情報等）
1	b001	売買目的有価証券　貸借対照表計上額 リース
2	b002	売買目的有価証券　損益に含まれた評価差額 リース
3	b003	満期保有目的債券合計　貸借対照表計上額 リース
4	b004	満期保有目的債券合計　時価 リース
5	b005	満期保有目的債券合計　差額 リース
6	b006	うち時価が計上額を超える分 リース
7	b007	うち時価が計上額を超えない分 リース
8	b008	子会社株式　貸借対照表計上額 リース
9	b009	子会社株式　時価 リース
10	b010	子会社株式　差額 リース
11	b011	関連会社株式　貸借対照表計上額 リース
12	b012	関連会社株式　時価 リース
13	b013	関連会社株式　差額 リース
14	b014	その他有価証券（株式）　取得原価 リース
15	b015	その他有価証券（株式）　貸借対照表計上額 リース
16	b016	その他有価証券（株式）　差額 リース

項番	カラム名	YI01（その他・明細情報等）
17	b017	うち計上額が取得原価を超える分　（株式）　リース
18	b018	うち計上額が取得原価を超えない分（株式）　リース
19	b019	その他有価証券（債券）　取得原価　リース
20	b020	その他有価証券（債券）　貸借対照表計上額　リース
21	b021	その他有価証券（債券）　差額　リース
22	b022	うち計上額が取得原価を超える分　（債券）　リース
23	b023	うち計上額が取得原価を超えない分（債券）　リース
24	b024	その他有価証券（その他）　取得原価　リース
25	b025	その他有価証券（その他）　貸借対照表計上額　リース
26	b026	その他有価証券（その他）　差額　リース
27	b027	うち計上額が取得原価を超える分　（その他）　リース
28	b028	うち計上額が取得原価を超えない分（その他）　リース
29	b029	その他有価証券合計　取得原価　リース
30	b030	その他有価証券合計　貸借対照表計上額　リース
31	b031	その他有価証券合計　差額　リース
32	b032	うち計上額が取得原価を超える分　リース
33	b033	うち計上額が取得原価を超えない分　リース
34	b034	当期に売却した満期保有債券合計　売却原価　リース
35	b035	当期に売却した満期保有債券合計　売却額　リース
36	b036	当期に売却した満期保有債券合計　売却損益　リース
37	b037	当期に売却したその他有価証券合計　売却額　リース
38	b038	当期に売却したその他有価証券合計　売却益　リース
39	b039	当期に売却したその他有価証券合計　売却損　リース
40	b040	時価評価されていない満期保有目的債券　リース
41	b041	時価評価されていない子会社・関連会社株式等　リース
42	b042	時価評価されていないその他有価証券　リース
43	b043	満期のある有価証券償還予定額　１年内　リース
44	b044	満期のある有価証券償還予定額　１～５年　リース
45	b045	満期のある有価証券償還予定額　５～１０年　リース
46	b046	満期のある有価証券償還予定額　１０年超　リース
47	b047	年金制度（連結子会社）リース
48	b048	その他有価証券の評価差額処理方法　リース
49	b049	予備　リース
50	b050	予備　リース
51	b051	（通貨）契約額　リース
52	b052	（通貨）時価　リース
53	b053	（通貨）評価損益　リース
54	b054	（金利）契約額　リース
55	b055	（金利）時価　リース

項番	カラム名	YI01（その他・明細情報等）
56	b056	（金利）評価損益 リース
57	b057	（株式）契約額 リース
58	b058	（株式）時価 リース
59	b059	（株式）評価損益 リース
60	b060	（債券）契約額 リース
61	b061	（債券）時価 リース
62	b062	（債券）評価損益 リース
63	b063	（商品）契約額 リース
64	b064	（商品）時価 リース
65	b065	（商品）評価損益 リース
66	b066	退職給付債務 リース
67	b067	年金資産額 リース
68	b068	その他の包括利益累計額計上額合計（未認識債務合計） リース
69	b069	未認識過去勤務費用（未認識過去勤務債務） リース
70	b070	未認識数理計算上の差異 リース
71	b071	その他（会計基準変更時差異の未処理額） リース
72	b072	前払年金費用 リース
73	b073	退職給付引当金 リース
74	b074	退職給付費用 リース
75	b075	勤務費用 リース
76	b076	利息費用 リース
77	b077	期待運用収益 リース
78	b078	未認識債務の償却費用合計 リース
79	b079	うち過去勤務債務の費用処理額 リース
80	b080	うち数理計算上の差異の費用処理額 リース
81	b081	うち会計基準変更時差異の費用処理額 リース
82	b082	うちその他未認識債務の償却費用 リース
83	b083	割引率 リース
84	b084	割引率（幅がある場合の下限） リース
85	b085	期待運用収益率 リース
86	b086	期待運用収益率（幅がある場合の下限） リース
87	b087	退職給付見込額の期間配分方法 リース
88	b088	過去勤務債務の償却処理年数 リース
89	b089	過去勤務債務の償却処理年数（幅がある場合の下限） リース
90	b090	数理計算上の差異の償却処理年数 リース
91	b091	数理計算上の差異の償却処理年数（幅がある場合の下限） リース
92	b092	会計基準変更時差異の償却処理年数 リース
93	b093	会計基準変更時差異の償却処理年数（幅がある場合の下限） リース
94	b094	退職給付信託額 リース

項番	カラム名	YI01（その他・明細情報等）
95	b095	退職給付信託期待運用収益率 リース
96	b096	原則法、簡便法区分フラグ リース
97	b097	複数事業主制度積立状況　年金資産額 リース
98	b098	複数事業主制度積立状況　年金財政計算上の給付債務額 リース
99	b099	複数事業主制度積立状況　差引額 リース
100	b100	複数事業主制度積立状況　制度全体に占める自社掛金拠出割合 リース
101	b101	取得価額相当額 リース
102	b102	減価償却累計額相当額 リース
103	b103	減損損失累計額相当額 リース
104	b104	期末残高相当額 リース
105	b105	未経過リース料期末残高相当額 リース
106	b106	１年内 リース
107	b107	１年超 リース
108	b108	リース資産減損勘定の残高 リース
109	b109	支払リース料 リース
110	b110	リース資産減損勘定の取崩額 リース
111	b111	減価償却費相当額 リース
112	b112	支払利息相当額 リース
113	b113	減損損失 リース
114	b114	減価償却費相当額の算定方法 リース
115	b115	支払利息相当額の算定方法 リース
116	b116	オペレーティング・リース未経過リース料 リース
117	b117	１年内 リース
118	b118	１年超 リース
119	b119	取得価額 リース
120	b120	減価償却累計額 リース
121	b121	期末残高 リース
122	b122	未経過リース料期末残高相当額 リース
123	b123	１年内 リース
124	b124	１年超 リース
125	b125	受取リース料 リース
126	b126	減価償却費相当額 リース
127	b127	受取利息相当額 リース
128	b128	受取利息相当額の算定方法 リース
129	b129	オペレーティング・リース未経過リース料 リース
130	b130	１年内 リース
131	b131	１年超 リース
132	b132	無議決権株式：株式数 リース
133	b133	議決権制限株式（自己株式等）：株式数 リース

項番	カラム名	YI01（その他・明細情報等）
134	b134	議決権制限株式（その他）：株式数 リース
135	b135	議決権制限株式（その他）：議決権数 リース
136	b136	完全議決権株式（自己株式等）：株式数 リース
137	b137	うち相互保有株式：株式数 リース
138	b138	完全議決権株式（その他）：株式数 リース
139	b139	完全議決権株式（その他）：議決権数 リース
140	b140	うち普通株式：株式数 リース
141	b141	うち普通株式：議決権数 リース
142	b142	うち種類株式：株式数 リース
143	b143	うち種類株式：議決権数 リース
144	b144	うち証券保管振替機構名義：株式数 リース
145	b145	うち証券保管振替機構名義：議決権数 リース
146	b146	うち自社名義失念株式：株式数 リース
147	b147	うち自社名義失念株式：議決権数 リース
148	b148	単元未満株式数 リース
149	b149	総株主の議決権数 リース
150	b150	失念・喪失株式：議決権数 リース
151	b151	【単位フラグ】議決権の状況 リース
152	b152	予備 リース
153	b153	予備 リース
154	b154	予備 リース
155	b155	予備 リース
156	b156	予備 リース
157	b157	予備 リース
158	b158	予備 リース
159	b159	予備 リース
160	b160	予備 リース
161	b161	予備 リース
162	b162	予備 リース
163	b163	予備 リース
164	b164	予備 リース
165	b165	予備 リース
166	b166	予備 リース
167	b167	予備 リース
168	b168	予備 リース
169	b169	予備 リース
170	b170	予備 リース
171	b171	予備 リース
172	b172	予備 リース

項番	カラム名	YI01（その他・明細情報等）
173	b173	予備 リース
174	b174	予備 リース
175	b175	その他退職給付費用 リース
176	b176	うち確定拠出年金への掛金支払額 リース
177	b177	年金制度 リース
178	b178	（通貨）契約額 リース
179	b179	（通貨）時価 リース
180	b180	（金利）契約額 リース
181	b181	（金利）時価 リース
182	b182	（株式）契約額 リース
183	b183	（株式）時価 リース
184	b184	（債券）契約額 リース
185	b185	（債券）時価 リース
186	b186	（商品）契約額 リース
187	b187	（商品）時価 リース
188	b188	金銭債権・満期有価証券の連結決算日後の償還予定額 1 年内 リース
189	b189	金銭債権・満期有価証券の連結決算日後の償還予定額 1 年超 5 年以内 リース
190	b190	金銭債権・満期有価証券の連結決算日後の償還予定額 5 年超 10 年以内 リース
191	b191	金銭債権・満期有価証券の連結決算日後の償還予定額 10 年超 リース
192	b192	積立制度の退職給付債務 リース
193	b193	非積立制度の退職給付債務 リース
194	b194	連結貸借対照表に計上された負債と資産の純額 リース
195	b195	【その他の包括利益計上額】過去勤務費用 リース
196	b196	【その他の包括利益計上額】 数理計算上の差異 リース
197	b197	【その他の包括利益計上額】 会計基準変更時差異 リース
198	b198	【その他の包括利益計上額】 その他 リース
199	b199	【その他の包括利益計上額】その他の包括利益額計上額 リース
200	b200	【その他の包括利益累計額計上額】 会計基準変更時差異の未処理額 リース
201	b201	【その他の包括利益累計額計上額】 その他 リース
202	b202	【退職給付債務調整表】期首における退職給付債務 リース
203	b203	【退職給付債務調整表】 勤務費用 リース
204	b204	【退職給付債務調整表】 利息費用 リース
205	b205	【退職給付債務調整表】 数理計算上の差異の当期発生額 リース
206	b206	【退職給付債務調整表】 退職給付の支払額 リース
207	b207	【退職給付債務調整表】 過去勤務費用の当期発生額 リース
208	b208	【退職給付債務調整表】 その他 リース
209	b209	【退職給付債務調整表】期末における退職給付債務 リース
210	b210	退職給付信託額の割合 リース

YJ01 (その他・明細情報等)

表 8.13: YJ01 (その他・明細情報等)

項番	カラム名	YJ01（その他・明細情報等）
1	b001	１単元の株数
2	b002	上位十大株主持株数
3	b003	少数特定者持株数
4	b004	浮動株数
5	b005	大株主・役員以外の株主数
6	b006	投信持株数
7	b007	年金持株数
8	b008	役員持株数
9	b009	従業員持株会持株数
10	b010	授権株式数
11	b011	政府公共団体所有株数
12	b012	金融機関所有株数
13	b013	金融商品取引業者所有株数
14	b014	その他法人所有株数
15	b015	外国法人等所有株数
16	b016	個人・その他所有株数
17	b017	単元未満株式数
18	b018	政府公共団体株主数
19	b019	金融機関株主数
20	b020	金融商品取引業者株主数
21	b021	その他法人株主数
22	b022	外国法人等株主数
23	b023	個人・その他株主数
24	b024	所有株式数合計
25	b025	単元株主数合計
26	b026	社債の１年以内償還予定額
27	b027	社債の２年以内償還予定額
28	b028	社債の３年以内償還予定額
29	b029	社債の４年以内償還予定額
30	b030	社債の５年以内償還予定額
31	b031	短期借入金の平均利率
32	b032	短期借入金の平均利率（幅がある時の下限）
33	b033	１年内返済予定の長期借入金の平均利率
34	b034	１年内返済予定の長期借入金の平均利率（幅がある時の下限）
35	b035	1 年以内に返済予定のリース債務の平均利率
36	b036	1 年以内に返済予定のリース債務の平均利率（幅がある時の下限）

項番	カラム名	YJ01（その他・明細情報等）
37	b037	長期借入金の平均利率（１年内返済を除く）
38	b038	長期借入金の平均利率（幅がある時の下限）
39	b039	リース債務の平均利率
40	b040	リース債務の平均利率（幅がある時の下限）
41	b041	コマーシャルペーパーの平均利率
42	b042	コマーシャルペーパーの平均利率（幅がある時の下限）
43	b043	借入金のうち無利子負債額
44	b044	長期借入金の１年超２年以内返済予定額
45	b045	長期借入金の２年超３年以内返済予定額
46	b046	長期借入金の３年超４年以内返済予定額
47	b047	長期借入金の４年超５年以内返済予定額
48	b048	リース債務１年超２年以内返済予定額
49	b049	リース債務２年超３年以内返済予定額
50	b050	リース債務３年超４年以内返済予定額
51	b051	リース債務４年超５年以内返済予定額
52	b052	その他有利子負債１年超２年以内返済予定額
53	b053	その他有利子負債２年超３年以内返済予定額
54	b054	その他有利子負債３年超４年以内返済予定額
55	b055	その他有利子負債４年超５年以内返済予定額
56	b056	受取手形
57	b057	売掛金
58	b058	短期貸付金
59	b059	株式
60	b060	その他有価証券
61	b061	出資金
62	b062	長期貸付金
63	b063	支払手形
64	b064	買掛金
65	b065	短期借入金
66	b066	長期借入金
67	b067	売上高・営業収益≪累計≫
68	b068	受取利息・割引料≪累計≫
69	b069	受取配当金≪累計≫
70	b070	仕入高≪累計≫
71	b071	原材料仕入高≪累計≫
72	b072	有形固定資産の減価償却方法
73	b073	無形固定資産の減価償却方法
74	b074	投資その他の資産の減価償却方法
75	b075	その他有価証券の評価方法

項番	カラム名	YJ01（その他・明細情報等）
76	b076	特定金銭信託の評価基準・評価方法
77	b077	棚卸資産（商品）の評価基準・評価方法
78	b078	棚卸資産（製品）の評価基準・評価方法
79	b079	棚卸資産（半製品）の評価基準・評価方法
80	b080	棚卸資産（仕掛品）の評価基準・評価方法
81	b081	棚卸資産（原材料）の評価基準・評価方法
82	b082	棚卸資産（貯蔵品）の評価基準・評価方法
83	b083	棚卸資産（その他）の評価基準・評価方法
84	b084	棚卸資産（合計）の評価基準・評価方法
85	b085	現金及び預金　貸借対照表計上額
86	b086	現金及び預金　時価
87	b087	受取手形及び売掛金　貸借対照表計上額
88	b088	受取手形及び売掛金　時価
89	b089	受取手形及び売掛金　差額
90	b090	有価証券及び投資有価証券　貸借対照表計上額
91	b091	有価証券及び投資有価証券　時価
92	b092	有価証券及び投資有価証券　差額
93	b093	長期貸付金　貸借対照表計上額
94	b094	貸倒引当金　貸借対照表計上額
95	b095	その他資産　貸借対照表計上額
96	b096	その他資産　時価
97	b097	その他資産　差額
98	b098	資産合計　貸借対照表計上額
99	b099	資産合計　時価
100	b100	資産合計　差額
101	b101	支払手形及び買掛金　貸借対照表計上額
102	b102	支払手形及び買掛金　時価
103	b103	支払手形及び買掛金　差額
104	b104	短期借入金　貸借対照表計上額
105	b105	短期借入金　時価
106	b106	短期借入金　差額
107	b107	社債　貸借対照表計上額
108	b108	社債　時価
109	b109	社債　差額
110	b110	長期借入金　貸借対照表計上額
111	b111	長期借入金　時価
112	b112	長期借入金　差額
113	b113	リース債務　貸借対照表計上額
114	b114	リース債務　時価

項番	カラム名	YJ01（その他・明細情報等）
115	b115	リース債務　差額
116	b116	その他負債　貸借対照表計上額
117	b117	その他負債　時価
118	b118	その他負債　差額
119	b119	負債合計　貸借対照表計上額
120	b120	負債合計　時価
121	b121	負債合計　差額
122	b122	デリバティブ取引計　貸借対照表計上額
123	b123	デリバティブ取引計　時価
124	b124	デリバティブ取引計　差額
125	b125	予備
126	b126	予備
127	b127	予備
128	b128	予備
129	b129	予備
130	b130	予備
131	b131	予備
132	b132	予備
133	b133	予備
134	b134	予備
135	b135	予備
136	b136	予備
137	b137	予備
138	b138	予備
139	b139	予備
140	b140	予備
141	b141	予備
142	b142	予備
143	b143	予備
144	b144	予備
145	b145	予備
146	b146	予備
147	b147	予備
148	b148	予備
149	b149	予備
150	b150	予備
151	b151	予備
152	b152	予備
153	b153	予備

項番	カラム名	YJ01（その他・明細情報等）
154	b154	予備
155	b155	予備
156	b156	予備
157	b157	予備
158	b158	予備
159	b159	予備
160	b160	予備
161	b161	予備
162	b162	予備
163	b163	予備
164	b164	予備
165	b165	予備
166	b166	予備
167	b167	予備
168	b168	予備
169	b169	予備
170	b170	予備
171	b171	予備
172	b172	予備
173	b173	予備
174	b174	予備
175	b175	予備
176	b176	予備
177	b177	予備
178	b178	予備
179	b179	予備
180	b180	予備
181	b181	予備
182	b182	予備
183	b183	予備
184	b184	予備
185	b185	予備
186	b186	予備
187	b187	予備
188	b188	予備
189	b189	予備
190	b190	予備
191	b191	予備
192	b192	予備

項番	カラム名	YJ01（その他・明細情報等）
193	b193	予備
194	b194	予備
195	b195	予備
196	b196	予備
197	b197	予備
198	b198	予備
199	b199	予備
200	b200	予備
201	b201	予備
202	b202	予備
203	b203	予備
204	b204	予備
205	b205	暫定会計処理フラグ（ＢＳ）
206	b206	暫定会計処理フラグ（ＰＬ）
207	b207	予備
208	b208	予備
209	b209	予備
210	b210	予備

YK01 (その他・明細情報等)

表 8.14: YK01 (その他・明細情報等)

項番	カラム名	YK01（その他・明細情報等）
1	b001	商品・製品売上高
2	b002	製品売上高
3	b003	商品売上高
4	b004	その他売上高・営業収益・営業収入
5	b005	（▲）売上値引・戻り高
6	b006	金融収益
7	b007	売上高・営業収益合計
8	b008	期首製品・商品棚卸高
9	b009	当期製品製造原価
10	b010	当期商品仕入高
11	b011	小計
12	b012	期末製品・商品棚卸高
13	b013	（▲）原価差額・他勘定振替高
14	b014	製品・商品評価損
15	b015	物品税・消費税
16	b016	その他営業原価

項番	カラム名	YK01（その他・明細情報等）
17	b017	売上原価・営業原価合計
18	b018	金融費用
19	b019	商品・製品売上高
20	b020	製品売上高
21	b021	商品売上高
22	b022	その他売上高・営業収益・営業収入
23	b023	（▲）売上値引・戻り高
24	b024	金融収益
25	b025	売上高・営業収益合計
26	b026	期首製品・商品棚卸高
27	b027	当期製品製造原価
28	b028	当期商品仕入高
29	b029	小計
30	b030	期末製品・商品棚卸高
31	b031	（▲）原価差額・他勘定振替高
32	b032	製品・商品評価損
33	b033	物品税・消費税
34	b034	その他営業原価
35	b035	売上原価・営業原価合計
36	b036	金融費用
37	b037	原材料費
38	b038	労務費・福利厚生費
39	b039	経費合計
40	b040	うち外注加工費
41	b041	うち動力・燃料・水道費
42	b042	うち荷造・運搬・保管費
43	b043	うち減価償却費
44	b044	うち賃借料
45	b045	うち租税公課
46	b046	うち支払特許料
47	b047	当期製造総費用
48	b048	期首仕掛品棚卸高
49	b049	期末仕掛品棚卸高
50	b050	（▲）他勘定振替高
51	b051	製造原価明細合計
52	b052	原材料費
53	b053	労務費・福利厚生費
54	b054	経費合計
55	b055	うち外注加工費

項番	カラム名	YK01（その他・明細情報等）
56	b056	うち動力・燃料・水道費
57	b057	うち荷造・運搬・保管費
58	b058	うち減価償却費
59	b059	うち賃借料
60	b060	うち租税公課
61	b061	うち支払特許料
62	b062	当期製造総費用
63	b063	期首仕掛品棚卸高
64	b064	期末仕掛品棚卸高
65	b065	（▲）他勘定振替高
66	b066	製造原価明細合計
67	b067	販売手数料
68	b068	荷造・運搬・保管費
69	b069	広告・宣伝費
70	b070	拡販費・その他販売費
71	b071	貸倒損失・貸倒引当金繰入額
72	b072	役員報酬
73	b073	役員退職慰労、役員退職慰労引当金繰入額
74	b074	役員賞与引当金繰入額
75	b075	人件費・福利厚生費
76	b076	退職給付費用・退職給付引当金繰入額
77	b077	減価償却費
78	b078	のれん償却額
79	b079	賃借料
80	b080	租税公課
81	b081	支払特許料
82	b082	開発費・試験研究費
83	b083	保証修理費
84	b084	その他販売費および一般管理費
85	b085	販売費および一般管理費明細合計
86	b086	販売手数料
87	b087	荷造・運搬・保管費
88	b088	広告・宣伝費
89	b089	拡販費・その他販売費
90	b090	貸倒損失・貸倒引当金繰入額
91	b091	役員報酬
92	b092	役員退職慰労、役員退職慰労引当金繰入額
93	b093	役員賞与引当金繰入額
94	b094	人件費・福利厚生費

項番	カラム名	YK01（その他・明細情報等）
95	b095	退職給付費用・退職給付引当金繰入額
96	b096	減価償却費
97	b097	のれん償却額
98	b098	賃借料
99	b099	租税公課
100	b100	支払特許料
101	b101	開発費・試験研究費
102	b102	保証修理費
103	b103	その他販売費および一般管理費
104	b104	販売費および一般管理費明細合計
105	b105	予備
106	b106	予備
107	b107	予備
108	b108	予備
109	b109	予備
110	b110	予備
111	b111	予備
112	b112	予備
113	b113	予備
114	b114	予備
115	b115	予備
116	b116	予備
117	b117	予備
118	b118	予備
119	b119	予備
120	b120	予備
121	b121	予備
122	b122	予備
123	b123	予備
124	b124	予備
125	b125	予備
126	b126	予備
127	b127	予備
128	b128	予備
129	b129	予備
130	b130	予備
131	b131	予備
132	b132	予備
133	b133	予備

項番	カラム名	YK01（その他・明細情報等）
134	b134	予備
135	b135	予備
136	b136	予備
137	b137	予備
138	b138	予備
139	b139	予備
140	b140	予備
141	b141	予備
142	b142	予備
143	b143	予備
144	b144	予備
145	b145	予備
146	b146	予備
147	b147	予備
148	b148	予備
149	b149	予備
150	b150	予備
151	b151	予備
152	b152	予備
153	b153	予備
154	b154	予備
155	b155	予備
156	b156	予備
157	b157	予備
158	b158	予備
159	b159	予備
160	b160	予備
161	b161	予備
162	b162	予備
163	b163	予備
164	b164	予備
165	b165	予備
166	b166	予備
167	b167	予備
168	b168	予備
169	b169	予備
170	b170	予備
171	b171	予備
172	b172	予備

項番	カラム名	YK01（その他・明細情報等）
173	b173	予備
174	b174	予備
175	b175	予備
176	b176	予備
177	b177	予備
178	b178	予備
179	b179	予備
180	b180	予備
181	b181	予備
182	b182	予備
183	b183	予備
184	b184	予備
185	b185	予備
186	b186	予備
187	b187	予備
188	b188	予備
189	b189	予備
190	b190	予備
191	b191	予備
192	b192	予備
193	b193	予備
194	b194	予備
195	b195	予備
196	b196	予備
197	b197	予備
198	b198	予備
199	b199	予備
200	b200	予備
201	b201	予備
202	b202	予備
203	b203	予備
204	b204	予備
205	b205	予備
206	b206	予備
207	b207	予備
208	b208	予備
209	b209	予備
210	b210	予備

YL01 (更新停止項目)

表 8.15: YL01 (更新停止項目)

項番	変数名	YL01 （更新停止項目）
1	b001	前期繰越利益
2	b002	うち合併引継未処分利益
3	b003	利益準備金取崩額
4	b004	諸任意積立金目的取崩額
5	b005	自己株式処分差損
6	b006	自己株式消却額
7	b007	普通株式中間配当額
8	b008	優先株式中間配当額
9	b009	中間配当に伴う利益準備金積立額
10	b010	その他諸任意積立金目的取崩額
11	b011	中間配当積立金取崩額
12	b012	当期未処分利益
13	b013	諸任意積立金・法定準備金取崩額
14	b014	利益準備金積立額
15	b015	普通株式配当金
16	b016	優先株式配当金
17	b017	資本金組入額
18	b018	諸任意積立金積立額
19	b019	中間配当積立金積立額
20	b020	その他の諸任意積立金積立額
21	b021	その他利益処分額
22	b022	次期繰越利益
23	b023	その他資本剰余金
24	b024	配当金
25	b025	その他のその他資本剰余金処分額
26	b026	その他資本剰余金次期繰越高
27	b027	資本剰余金期首残高
28	b028	資本剰余金増加高
29	b029	新株の発行
30	b030	自己株式処分差益
31	b031	その他の資本剰余金増加高
32	b032	資本剰余金減少高
33	b033	配当金
34	b034	優先株式配当金
35	b035	自己株式消却額
36	b036	その他の資本剰余金減少高

項番	変数名	YL01（更新停止項目）
37	b037	資本剰余金期末残高
38	b038	利益剰余金期首残高
39	b039	利益準備金期首残高
40	b040	その他の剰余金期首残高
41	b041	利益剰余金増加高
42	b042	当期純利益
43	b043	連結会社の異動に伴う剰余金増加高
44	b044	持分法適用会社の異動に伴う剰余金増加高
45	b045	その他の利益剰余金増加高
46	b046	利益剰余金減少高
47	b047	利益準備金繰入額
48	b048	当期純損失
49	b049	普通株式配当金
50	b050	優先株式配当金
51	b051	役員賞与
52	b052	資本金組入額
53	b053	自己株式消却額
54	b054	その他の利益剰余金減少高
55	b055	利益剰余金期末残高
56	b056	利益準備金期末残高　　（ＳＥＣ基準）
57	b057	その他の剰余金期末残高　　（ＳＥＣ基準）
58	b058	その他の包括利益期首残高
59	b059	未実現有価証券評価益期首残高
60	b060	為替換算調整額期首残高
61	b061	最小年金負債調整額期首残高
62	b062	その他の包括利益増減額
63	b063	未実現有価証券評価益
64	b064	為替換算調整額
65	b065	最小年金負債調整額
66	b066	その他の包括利益期末残高
67	b067	未実現有価証券評価益期末残高
68	b068	為替換算調整額期末残高
69	b069	最小年金負債調整額期末残高
70	b070	決算取締役会開催日
71	b071	新規連結子会社数
72	b072	除外連結子会社数
73	b073	新規持分法適用会社数
74	b074	除外持分法適用会社数
75	b075	従業員数　うち正社員

項番	変数名	YL01（更新停止項目）
76	b076	従業員数　うち常勤嘱託
77	b077	従業員数　うち受入社員
78	b078	従業員数　うち休職者等
79	b079	従業員数　うち出向者
80	b080	従業員数　うちその他
81	b081	予備
82	b082	普通社債当期発行額
83	b083	普通社債当期償還額
84	b084	債務履行引受契約による譲渡額
85	b085	為替予約による為替差額
86	b086	普通社債未償還残高
87	b087	新株引受権付社債当期発行額
88	b088	新株引受権付社債当期償還額
89	b089	債務履行引受契約による譲渡額
90	b090	為替予約による為替差額
91	b091	新株引受権付社債未償還残高
92	b092	転換社債当期発行額
93	b093	転換社債当期償還額
94	b094	転換社債未償還残高
95	b095	長期借入金当期借入額
96	b096	長期借入金当期返済額
97	b097	特定金銭信託の残高
98	b098	指定金外信託の残高
99	b099	流動資産に属する株式・簿価
100	b100	流動資産に属する株式・時価
101	b101	流動資産に属する株式・評価損益
102	b102	流動資産に属する債券・簿価
103	b103	流動資産に属する債券・時価
104	b104	流動資産に属する債券・評価損益
105	b105	流動資産に属するその他・簿価
106	b106	流動資産に属するその他・時価
107	b107	流動資産に属するその他・評価損益
108	b108	流動資産に属する小計・簿価
109	b109	流動資産に属する小計・時価
110	b110	流動資産に属する小計・評価損益
111	b111	固定資産に属する株式・簿価
112	b112	固定資産に属する株式・時価
113	b113	固定資産に属する株式・評価損益
114	b114	固定資産に属する債券・簿価

項番	変数名	YL01（更新停止項目）
115	b115	固定資産に属する債券・時価
116	b116	固定資産に属する債券・評価損益
117	b117	固定資産に属するその他・簿価
118	b118	固定資産に属するその他・時価
119	b119	固定資産に属するその他・評価損益
120	b120	固定資産に属する小計・簿価
121	b121	固定資産に属する小計・時価
122	b122	固定資産に属する小計・評価損益
123	b123	時価情報（旧）　合計・簿価
124	b124	時価情報（旧）　合計・時価
125	b125	時価情報（旧）　合計・評価損益
126	b126	単元株制度採用フラグ
127	b127	総株式数
128	b128	総株主数
129	b129	百万株以上所有株数
130	b130	百万株以上株主数
131	b131	五十万株以上所有株数
132	b132	五十万株以上株主数
133	b133	十万株以上所有株数
134	b134	十万株以上株主数
135	b135	五万株以上所有株数
136	b136	五万株以上株主数
137	b137	一万株以上所有株数
138	b138	一万株以上株主数
139	b139	五千株以上所有株数
140	b140	五千株以上株主数
141	b141	五百株以上所有株数
142	b142	五百株以上株主数
143	b143	五百株未満所有株数
144	b144	五百株未満株主数
145	b145	追加最小負債額
146	b146	予備
147	b147	予備
148	b148	予備
149	b149	予備
150	b150	予備
151	b151	予備
152	b152	予備
153	b153	予備

項番	変数名	YL01（更新停止項目）
154	b154	予備
155	b155	予備
156	b156	予備
157	b157	予備
158	b158	予備
159	b159	予備
160	b160	予備
161	b161	予備
162	b162	予備
163	b163	予備
164	b164	予備
165	b165	予備
166	b166	予備
167	b167	予備
168	b168	予備
169	b169	予備
170	b170	予備
171	b171	予備
172	b172	予備
173	b173	予備
174	b174	予備
175	b175	予備
176	b176	予備
177	b177	予備
178	b178	予備
179	b179	予備
180	b180	予備
181	b181	予備
182	b182	予備
183	b183	予備
184	b184	予備
185	b185	予備
186	b186	予備
187	b187	予備
188	b188	予備
189	b189	予備
190	b190	予備
191	b191	予備
192	b192	予備

項番	変数名	YL01（更新停止項目）
193	b193	予備
194	b194	予備
195	b195	予備
196	b196	予備
197	b197	予備
198	b198	予備
199	b199	予備
200	b200	予備
201	b201	予備
202	b202	予備
203	b203	予備
204	b204	予備
205	b205	予備
206	b206	予備
207	b207	予備
208	b208	予備
209	b209	予備
210	b210	予備

第 9 章

Osiris データ仕様

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 [1]:

表 9.1: 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

[1] 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.

No.	カラム名	説明	カラム名 (オリジナル)
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

No.	カラム名	説明	カラム名 (オリジナル)
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

第 10 章

Orbis データ仕様

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 [1]:

表 10.1: Orbis 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

[1] 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.

No.	カラム名	説明	カラム名 (オリジナル)
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

No.	カラム名	説明	カラム名 (オリジナル)
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

参考文献

[1] Codd, E. F. (1970) A relational model of data for large shared data banks, *Communications of the ACM*, Vol. 13, pp. 377-387.

[2] Janssens, J. (2014) *Data Science at the Command Line*, O'Reilly Media.
(太田満久, 下田倫大, 増田泰彦監訳, 長尾高弘訳 (2015) 『コマンドラインではじめるデータサイエンス: 分析プロセスを自在に進めるテクニック』, オライリー・ジャパン.)

[3] 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.

[4] 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

[5] 地道正行 (2012) 『金融関連会社に関する日経 NEEDS 財務データにもとづくデータベースの構築』, 商学論究, 第 59 巻, 第 3 号, pp.35–70, 関西学院大学商学研究会.

[6] 地道正行 (2014) 『R を利用した財務データの可視化と統計モデリング: 探索的データ解析の視点から』, 商学論究, 第 61 巻, 第 3 号, pp.241–295, 関西学院大学商学研究会.

[7] 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.

[8] 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.

[9] 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.

[10] 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.

[11] 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.

[12] 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 68 巻, 第 3 号, pp. 1–78, 関西学院大学商学研究会.

[13] 地道正行 (2021-b) 『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

[14] 地道正行 (2021-c) 『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 69 巻, 第 1 号, pp. 71–109, 関西学院大学商学研究会.

[15] 地道正行 (2021-d) 『財務データ抽出システムの再構築: Orbis データの利用』, 商学論究, 第 69 巻, 第 2

号, pp. 65–109, 関西学院大学商学研究会.

[16] 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.

[17] 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.

[18] 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.

[19] 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.

[20] 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), ポスター発表.

[21] 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), 発表資料.

[22] 地道正行, 阪 智香 (2021-a) 『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 68 巻, 第 3 号, pp. 79–116, 関西学院大学商学研究会.

[23] 地道正行, 阪 智香 (2021-b)『財務データ抽出システム SKWAD 』, 日本経営数学会第 43 回 (通算 63 回) 研究大会, オンライン開催 (専修大学神田キャンパス), 2021 年 6 月 12 日 (土), 報告要旨集, pp. 15–20.

[24] 地道正行, 阪 智香 (2021-c)『財務データ抽出システム SKWAD』, 国際数理科学協会, 2021 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, オンライン開催, 2021 年 8 月 22 日 (日), 発表資料.

[25] 地道正行, 宮本大輔, 阪 智香, 永田修一 (2021-a) 『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 13 回シンポジウム, オンライン開催, 2021 年 7 月 9 日 (木), ポスター発表.

[26] 地道正行, 宮本大輔, 阪 智香, 永田修一 (2021-b) 『財務データ抽出システム SKWAD: Osiris データと Orbis データの利用』, 日本計算機統計学会, 第 35 回シンポジウム, 慶應義塾大学三田キャンパス, ハイブリッド開催, 2021 年 11 月 27 日 (土), 講演論文集, pp. 39–42.

[27] Mecklenburg, R. (2005) *Managing Projects with GNU Make, Third Edition*, O'Reilly Media, Inc.

[28] 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.

[29] 日本経済新聞社 (2018) 『NIKKEI NEEDS 一般事業会社 企業財務データ 項目定義書 2018 年 12 月 17 日版』, 日本経済新聞社.

[30] 日本経済新聞社 (2019) 『NIKKEI NEEDS 一般事業会社 企業財務データ 2019 年 10 月 1 日版』, 日本経済新聞社.

[31] 日本経済新聞社 (2020) 『NIKKEI NEEDS 一般事業会社 企業財務データ 2020 年 6 月 24 日版』, 日本経済新聞社.

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

[33] 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.

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

[35] Taddy, M. (2019) *Business Data Science: Combining Machine Learning and Economics to Optimize, Automate, and Accelerate Business Decisions*, McGraw-Hill.
(上杉隼人, 井上毅郎 共訳 (2020) 『ビジネスデータサイエンスの教科書』, すばる舎.)

[36] 田中 凜, 阪 智香, 地道正行 (2021) 『日本の上場企業における企業価値に関する統計モデリング』, 国際数理科学協会, 2021 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, オンライン開催, 2021 年 8 月 22 日 (日), 発表資料.

[37] 田中 凜, 阪 智香, 地道正行 (2022) 『企業価値の統計モデリング』, 第 16 回日本統計学会春季集会, ポスター発表, 慶應義塾大学三田キャンパス, ハイブリッド開催, 2022 年 3 月 4 日 (土).

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

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

[40] 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).

[41] 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)

[42] 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).

索引

財務データ抽出システム SKWAD

2022 年 5 月 20 日 初版 発行

著 者: 地道 正行

ISBN: 978-4-9905530-1-2