コードレビュー Vol.12

CODE Review

USP 研究所技術研究員
written by 大内智明

今回は、ユニケージ開発で扱うデータ形式について、ご紹介いたします。

データ形式の種類と用途について

ユニケージ開発は、リレーショナルデータベース（RDB）を使用せず、フラットなテキストファイルでデータを構成して、業務システムを作成します。テキストファイルには、フィールド形式・タグ形式・ネーム形式の3種類のデータ形式があり、用途に合わせてデータ形式を使い分けます（図2）。フィールド形式は、各フィールド（列）とレコード（行）から成るデータです。タグ形式は、フィールド形式のデータとヘッダー部分の項目名から成るデータです。ネーム形式は、一行が項目名と値で形成するデータです。

データ形式と使い方

（A）タグ形式は、フィールド形式＋ヘッダー部分に項目名の形式です。タグ形式は、項目名が多くて並び替えがよく発生する場合に有効です。

（B）フィールド形式は、各レコードが半角スペース区切りで並べた形式です。この形式はユニケージ開発の中でも最もよく使うデータ形式です。

（C）ネーム形式は、項目名＋値の形式です。ネーム形式は、HTML の画面に入力した値を取得する場合によく使用します。

図1　タグ形式

```
$ cat data
TAG1 TAG2 TAG3 TAG4 TAG5 TAG6 TAG7 TAG8 TAG9
0000000 浜地_____ 50 F 91 59 20 76 54
0000001 鈴田_____ 50 F 46 39 8  5  21
0000003 杉山_____ 26 F 30 50 71 36 30
0000004 白土_____ 40 M 58 71 20 10 6
0000005 崎村_____ 50 F 82 79 16 21 80
$ tagself TAG2 TAG4 data
TAG2 TAG4
浜地_____ F
鈴田_____ F
杉山_____ F
白土_____ M
崎村_____ F
```

データ＋項目名で構成

データ処理は、項目名を指定して実行します

図2　フィールド形式

```
$ cat data
0000000 浜地_____ 50 F 91 59 20 76 54
0000001 鈴田_____ 50 F 46 39 8  5  21
0000003 杉山_____ 26 F 30 50 71 36 30
0000004 白土_____ 40 M 58 71 20 10 6
0000005 崎村_____ 50 F 82 79 16 21 80
$ self 2 4 data
浜地_____ F
鈴田_____ F
杉山_____ F
白土_____ M
崎村_____ F
```

各レコードは
半角スペース区切り

データ処理は、
フィールドを指定して実行します

図3　ネーム形式

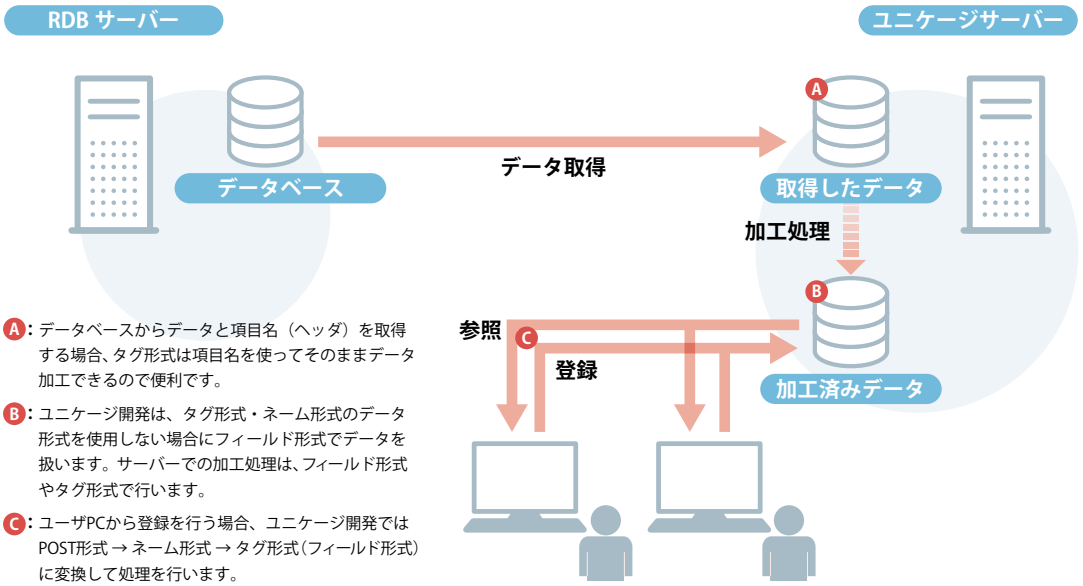
```
$ cat data
0000000 浜地_____
0000001 鈴田_____
0000003 杉山_____
0000004 白土_____
0000005 崎村_____
$ nameread 0000001 data
鈴田_____
```

項目名

値

データ処理は、項目名を指定して実行します

図4　データ形式と使う場面



Ⓐ：データベースからデータと項目名（ヘッダ）を取得する場合、タグ形式は項目名を使ってそのままデータ加工できるので便利です。

Ⓑ：ユニケージ開発は、タグ形式・ネーム形式のデータ形式を使用しない場合にフィールド形式でデータを扱います。サーバーでの加工処理は、フィールド形式やタグ形式で行います。

Ⓒ：ユーザPCから登録を行う場合、ユニケージ開発ではPOST形式 → ネーム形式 → タグ形式（フィールド形式）に変換して処理を行います。
サーバーからユーザPCにHTMLとして出力する場合は、タグ形式（フィールド形式）→ ネーム形式に変換を行います。

リスト1　LV3データからLV4を作成する

```
#!/bin/bash -xv
#
# LV4MAKE.DAY.HAC_SYS >>> 発注システムを作成
#
# Usage : LV4MAKE.DAY.HAC_SYS_HAC
#
# 説明  :LV3データからPOMPA(LV4)データを作成する
#
# Written by xxxx /Date: 20xx/xx/xx

# 走行ログの記録
logfile="/home/usp/LOG/LOG.$(basename $0).$(date +%Y%m%d)_$(date +%H%M%S)_$$"
exec 2> ${logfile}

#//////////////////////////////////////////////////////////////////////////
# 初期設定
#//////////////////////////////////////////////////////////////////////////
# パスの定義
export PATH=/home/UTL:/home/TOOL:/usr/local/bin:${PATH}
export LANG=ja_JP.UTF-8

# 変数の定義
tmp=/tmp/$$-$(basename $0)_$(date +%Y%m%d%H%M%S)
logd=${HOME}/LOG
semd=${HOME}/SEMAPHORE
lv3d=/LV3/HAC_SYS

# データの受け取り方は現状日単位(の予定)
today=$(date +%Y%m%d)
```

```
[ -z $sday ] && sday=${today}

# エラー時の終了処理定義
ERROR_CHECK(){
  [ $(plus ${PIPESTATUS[@]}) -eq 0 ] && return 0
  touch ${semd}/$(basename $0).${HOSTNAME}.ERROR.${sday}
  exit 1
}

# 開始時刻の記録
echo "${HOSTNAME} $(basename $0) START $(date +%Y%m%d%H%M%S)" >> ${logd}/UPCNT
touch ${semd}/$(basename $0).${HOSTNAME}.START.${sday}

#############################################################################
#  処理部
#############################################################################
# TBLパラメータ作成及び変更
#
# 1:大分類コード   2:大分類名
tagself DAIBUNRUI_CODE DAIBUNRUI_NAME ${lv3d}/DAI_BUNRUI    > $tmp-DAIBUNRUI_NAME
ERROR_CHECK
# 1:中分類コード   2:中分類名
tagself CHUBUNRUI_CODE CHUBUNRUI_NAME ${lv3d}/CHU_BUNRUI    > $tmp-CHUBUNRUI_NAME
ERROR_CHECK
# 1:売場コード     2:売場名
tagself URIBA_CODE URIBA_NAME ${lv3d}/URIBA                 > $tmp-URIBA_NAME
ERROR_CHECK
# 1:センタコード   2:センタ名
tagself CENT_CODE CENT_NAME ${lv3d}/CENT                    > $tmp-CENT_NAME
ERROR_CHECK
# 1:発注方法コード 2:発注方法名
tagself HAC_HOUHOU HAC_HOUHOU_NAME ${lv3d}/HOUHOU           > $tmp-HAC_HOUHOU_NAME
ERROR_CHECK
# 1:アイテムコード 2:アイテム名
tagself ITEM_CODE ITEM_NAME ${lv3d}/ITEM                    > $tmp-ITEM_NAME
ERROR_CHECK


# /_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_
# /_/_/_/_/_/_/_/_/_ タグ付きファイル処理             /_/_/_/_/_/_/_/_/_
# /_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_
# 発注帳票ヘッダ部
# ${lv3d}/HAC_CHOHYO_HEAD
#  1:発注帳票ＮＯ   2:発注日    3:売場コード 4:大分類コード 5:中分類コード
#  6:センターコード 7:発注方法
#
# 発注帳票ボディ部
# ${lv3d}/HAC_CHOHYO_BODY |
#  1:発注帳票ＮＯ       2:発注順          3:アイテムコード 4:アイテム数量     5:アイテム発注単位
#  6:アイテム原価       7:アイテム原価税込 8:アイテム売価    9:アイテム売価税込 10:アイテム生産国
#  11:アイテムメーカー
#
# 発注のボディーとヘッダを結合
tagjoin1 key=HAC_CHOHYO_NO ${lv3d}/HAC_CHOHYO_HEAD ${lv3d}/HAC_CHOHYO_BODY |
#  1:発注帳票ＮＯ       2:発注日          3:売場コード        4:大分類コード   5:中分類コード
#  6:センターコード     7:発注方法        8:発注順            9:アイテムコード 10:アイテム数量
#  11:アイテム発注単位 12:アイテム原価    13:アイテム原価税込 14:アイテム売価  15:アイテム売価税込
#  16:アイテム生産国   17:アイテムメーカー
#
```

tagself は、「データ形式と使い方」参照

❶

tagjoin1 は、図5参照

❷

```
 89  # 売場付加
 90  tagcjoin1 key=URIBA_CODE $tmp-URIBA_NAME -              |
 91  # 大分類名付加
 92  tagcjoin1 key=DAIBUNRUI_CODE $tmp-DAIBUNRUI_NAME -      |
 93  # 中分類名付加
 94  tagcjoin1 key=CHUBUNRUI_CODE $tmp-CHUBUNRUI_NAME -      |
 95  # センタ付加
 96  tagcjoin1 key=CENT_CODE $tmp-CENT_NAME -                |
 97  # 発注方法付加
 98  tagcjoin1 key=HAC_HOUHOU $tmp-HAC_HOUHOU_NAME -         |
 99  # アイテム名称付加
100  tagcjoin1 key=ITEM_CODE $tmp-ITEM_NAME -                |
101  #  1:発注帳票ＮＯ      2:発注日            3:売場コード       4:売場名            5:大分類コード
102  #  6:大分類名          7:中分類コード      8:中分類名         9:センターコード     10:センターコード名
103  #  11:発注方法         12:発注方法名       13:発注順          14:アイテムコード    15:アイテム名
104  #  16:アイテム数量     17:アイテム発注単位 18:アイテム原価    19:アイテム原価税込 20:アイテム売価
105  #  21:アイテム売価税込 22:アイテム生産国   23:アイテムメーカー
106  #
107  # 不要項目のフィールドを削除
108  tagdelf ITEM_COUNTRY ITEN_GENKA_INTAX ITEN_BAIKA_INTAX ITEN_MAKER |
109  #  1:発注帳票ＮＯ  2:発注日            3:売場コード     4:売場名称         5:大分類コード
110  #  6:大分類名称    7:中分類コード      8:中分類名称     9:センターコード   10:センター名称
111  #  11:発注方法     12:発注方法名       13:発注順        14:アイテムコード  15:アイテム名称
112  #  16:アイテム数量 17:アイテム発注単位 18:アイテム原価  19:アイテム売価
113  #
114  # 発注日でソート
115  tagmsort key=HAC_DATE                                   |
116  # 発注日の年月単位でファイル分割
117  tagkeycut ${HOME}/HAC_SYS/POMPA/ZYU_HAC_CHOHYO_MENTE/%HAC_DATE.1.6
118  ERROR_CHECK
119
120  #////////////////////////////////////////////////////////////////////////
121  # 終了処理
122  #////////////////////////////////////////////////////////////////////////
123  # 終了時刻の記録
124  echo "${HOSTNAME} $(basename $0) END $(date +%Y%m%d%H%M%S)" >> ${logd}/UPCNT
125  touch ${semd}/$(basename $0).${HOSTNAME}.END.${sday}
126
127  # 終了
128  rm -f ${tmp}-* 2>/dev/null
129  exit 0
```

tagdelf は、図6参照

tagmsort は、
タグ形式データ用のソートコマンド

tagkeycut は、図7参照

図5　tagjoin1

```
$ cat master
CODE NAME AGE SEX
0003 杉山 026 WOM
0005 崎村 050 MAN
0007 梶川 042 WOM
$ cat tran
SEQ CODE A1 A2 A3
001 0003 30 50 71
001 0004 58 71 20
001 0005 82 79 16
$ tagjoin1 key=CODE master tran
SEQ CODE NAME AGE SEX A1 A2 A3
001 0003 杉山 026 WOM 30 50 71
001 0005 崎村 050 MAN 82 79 16
```

masterとtranにおいて、keyで指定したフィールドで
マッチングを行い、一致した場合は、2ファイルを結合
して出力します

図6　tagdelf

```
$ cat master
CODE NAME AGE SEX
0003 杉山 026 WOM
0005 崎村 050 MAN
0007 梶川 042 WOM
$ tagdelf NAME AGE master
CODE SEX
0003 WOM
0005 MAN
0007 WOM
```

タグファイルから指定したフィールドを除いて出力します

図7　tagkeycut

```
$ cat data
ID KENNAME CITY CITYNAME TAG1 TAG2 TAG3 TAG4 TAG5
01 埼玉県 03 熊谷市 82 0 23 84 10
01 埼玉県 01 さいたま市 91 59 20 76 54
02 東京都 04 新宿区 30 50 71 36 30
02 東京都 05 中央区 78 13 44 28 51
03 千葉県 10 千葉市 52 91 44 9 0
03 千葉県 12 柏市 95 60 35 93 76
04 神奈川県 13 横浜市 92 56 83 96 75
04 神奈川県 16 小田原市 45 21 24 39 03
$ tagkeycut data.%ID data
$ ls -l data.*
-rw-rw-r-- 1 usp usp 137  1月 14 00:04 data.01
-rw-rw-r-- 1 usp usp 132  1月 14 00:04 data.02
-rw-rw-r-- 1 usp usp 127  1月 14 00:04 data.03
-rw-rw-r-- 1 usp usp 142  1月 14 00:04 data.04
$ cat data.01
ID KENNAME CITY CITYNAME TAG1 TAG2 TAG3 TAG4 TAG5
01 埼玉県 03 熊谷市 82 0 23 84 10
01 埼玉県 01 さいたま市 91 59 20 76 54
```

tagkeycutコマンドは、
指定したフィールドでファイルを分割します

コードの見どころ

コードについて、2つのセクションに分けてご説明します。

[1] コード＋名称のテーブルを作成します（❶46 ～ 65行目まで）。

[2] 発注帳票のヘッダ部ファイルとボディ部ファイルをキーで結合します（❷82 ～ 118行目まで）。

まとめ

ユニケージ開発では、使用するデータにより、フィールド形式・タグ形式・ネーム形式の3パターンで対応します。仮に、3パターン以外のデータが入力された場合は、変換処理を行い、3パターンの形にしてから処理を行います。