# RDB サーバと CLI ツールで帳票作成（PSQL）

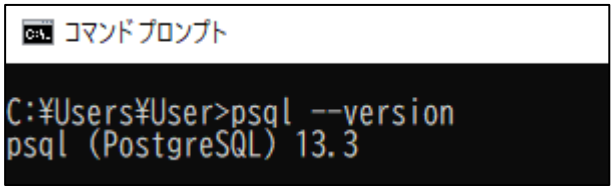
## 目次

# RDB サーバと CLI ツールで帳票作成（PSQL）

## はじめに

近年 情報系という言葉をあまり聞かなくなったように感じます。情報系システムは基幹系のデータやログをデータウェアハウス（DWH）に投入してツールで分析したり、専用のシステムを開発して運用していましたが基幹系のデータを RDB で管理するようになってからは利用者が SQL クエリで直接データを抽出・加工できるようになり専用のアプリケーションを必要としなくなったのが理由の一つと推測します。SQL は宣言型言語でデータの抽出と多段階の加工を制御処理を書かずに実現できるので、アプリケーションでレコードを 1 件づつ処理するのに比べて品質や性能、保守性が向上します。要件全体をクエリ化できない場合でも開発が必要なコードの量を小さくできる可能性があります。

この資料では“情報系システム”で使えそうな機能を SQL 構文と関数だけで実装してみます。但し、SQL は ANSI（米国国家規格協会）が 1986 年に SQL 86 として出して以降 現在も標準化が続けられていますが RDBMS 個々の機能拡張や歴史的経緯により完全な互換はありません。ここでは下図の環境で ANSI SQL に沿った構文と多くの RDBMS に用意されていそうな関数を使いますが関数名が異なっていたり機能が多少異なっている可能性もあります。




## 1. 使用ツールと出力形式

RDBMS のパッケージには通常 サーバを操作するためのクライアントアプリが添付されています。コマンドラインからサービスに接続してクエリを実行するためのクライアントとして PostgreSQL の場合は psql が付いています。クライアントアプリで使えるコマンドは RDBMS 間で標準化はされていませんが、必要な機能が同じなのでコマンドの種類も似通っています。特に PostgreSQL は開発の初期（～ver.7）に Oracle 互換に言及[1]していたこともあり今でも API 等が似ています[2]。

クライアントアプリに作表機能はありませんが、クエリの実行結果を html 形式で出力することができるのでこれを使って帳票イメージを作成できます。

[1] Oracle 互換性タスク https://wiki.postgresql.org/wiki/Oracle_Compatibility_Tasks

[2] PosgreSQL をベースにした RDB の商用製品、オープンソース、移行ツール等が開発されています
IvorySQL オープンソースの Oracle 互換 PostgreSQL：https://github.com/IvorySQL/IvorySQL
移行ツール Ora2Pg： https://ora2pg.darold.net/

# RDB サーバと CLI ツールで帳票作成（PSQL）

## 2. 多段階集計の出力例

帳票化するテーブルと出力の例を以下に示します。

### （1） 帳票イメージ

N 件の科目コードとペアで登録された金額から対象帳票区分が「損益」の科目だけを選択し、大分類、中分類、小分類、細分類毎に集計して帳票化します。最下位の分類は“金額”、金額を集計した上位の分類は“計”として出力します。

<コード体系とデータ>

① 科目コードのコード体系は以下のようになっています。

・科目コード：６桁（有意な桁以外は０詰め…〔例〕大分類コード=x00000）
・大分類：先頭１桁
・中分類：大分類＋１桁（科目コードの先頭から２桁）
・小分類：中分類＋２桁（科目コードの先頭から４桁）
・細分類：小分類＋２桁…科目によっては小分類までで細分類が存在しない場合があり、存在しない場合は末尾２桁は常に 00
・勘定科目マスタには大分類、中分類、小分類のレコードも登録されている

② 金額は細分類の科目コードで、細分類が存在しない場合は科目は中分類で登録されている

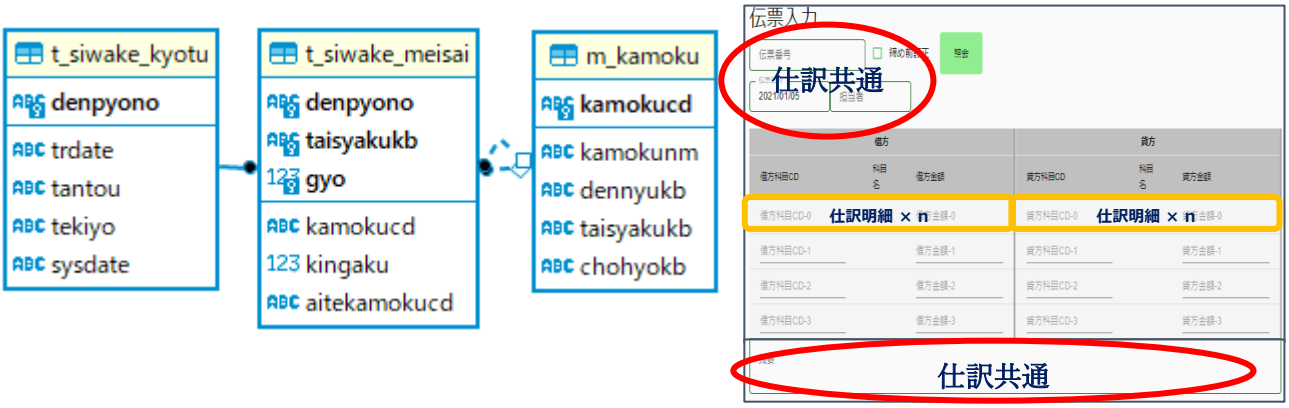

損益計算書

| 科目コード | 科目名 | 計 | 金額 |
|---|---|---|---|
| 800000 | 経常損益 | 990000000 | |
| 810000 | ｜売上高 | 10990000000 | |
| 810100 | ｜｜売上高 | 9990000000 | |
| 810101 | ｜｜｜売上高（売上控除高を除く） | | 10000000000 |
| 810109 | ｜｜｜売上控除高 | | -10000000 |
| 810200 | ｜｜役務収益 | | 1000000000 |
| 820000 | ｜売上原価 | -10000000000 | |
| 820100 | ｜｜期首商品・製品たな卸高 | | -1000000000 |
| 820200 | ｜｜当期仕入高 | -9900000000 | |
| 820201 | ｜｜｜当期商品仕入高（仕入控除高を除く） | | -10000000000 |
| 820209 | ｜｜｜仕入控除高 | | 100000000 |
| 820300 | ｜｜当期製品製造原価 | | -100000000 |
| 820400 | ｜｜期末商品・製品たな卸高 | | 1000000000 |
| 830000 | ｜販売費及び一般管理費 | 0 | |

# RDB サーバと CLI ツールで帳票作成（PSQL）

（2） 使用するテーブルの ER 図

テーブルは伝票入力を分解したイメージで、仕訳データ（共通）と仕訳データ（明細）が１：Ｎで仕訳データ（明細）の科目コードが勘定科目マスタに１：１対応しています。




## テーブル仕様書

| テーブル名 | 仕訳データ(共通) | 作成 | | 担当 | | 項目数 |
|---|---|---|---|---|---|---|
| TABLE_NAME | t_siwake_kyotu | 修正 | | 担当 | | 5 |

〔備考〕
会計取引(伝票入力等)の内容を仕訳し、共通的な情報を管理する。

| インデックス | キー名 | 一意 | 外部キー | 制約名 | 参照先 |
|---|---|---|---|---|---|
| p | pk_t_siwake_kyotu | ○ | 1 | fk_tantou | m_syain(sy |
| s1 | | | 2 | | |
| s2 | | | 3 | | |
| s3 | | | 4 | | |

| No. | FIELD_NAME | 日本語名 | 型 | 日本語 | 桁数 | 小数 | 必須 | IDX P | IDX s1 | IDX s2 | IDX s3 | 外部キー | 備考 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | tr_date | 取引日付 | CHAR | | 8 | | ○ | | | | | | 取引日付 |
| 2 | denpyono | 伝票番号 | CHAR | | 5 | | | 1 | | | | | 伝票種類2桁＋連番3桁 |
| 3 | tantou | 担当者 | CHAR | | 4 | | | | | | | 1 | 取引担当者の社員番号 |
| 4 | tekiyo | 摘要 | VARCHAR | ○ | 50 | | | | | | | | |
| 5 | sys_date | システム処理日 | CHAR | | 8 | | ○ | | | | | | システム投入日 |
| 6 | | | | | | | | | | | | | |

## テーブル仕様書

| テーブル名 | 仕訳データ(明細) | 作成 | | 担当 | | 項目数 |
|---|---|---|---|---|---|---|
| TABLE_NAME | t_siwake_meisai | 修正 | | 担当 | | 6 |

〔備考〕
会計取引(伝票入力等)の内容を仕訳した状態で管理する

| インデックス | キー名 | 一意 | 外部キー | 制約名 | 参照先 |
|---|---|---|---|---|---|
| p | pk_t_siwake_meisai | ○ | 1 | fk_kamokucd | m_kamoku |
| s1 | | | 2 | fk_aite_kamokucd | m_kamoku |
| s2 | | | 3 | fk_denpyono | t_siwake_k |
| s3 | | | 4 | | |

| No. | FIELD_NAME | 日本語名 | 型 | 日本語 | 桁数 | 小数 | 必須 | IDX P | IDX s1 | IDX s2 | IDX s3 | 外部キー | 備考 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | denpyono | 伝票番号 | CHAR | | 5 | | ○ | 1 | | | | 3 | 伝票種類2桁＋連番3桁 |
| 2 | gyo | 行番号 | NUMERIC | | 1 | | ○ | 2 | | | | | |
| 3 | kamokucd | 科目コード | CHAR | | 6 | | ○ | | | | | 1 | |
| 4 | taisyakukb | 貸借区分 | CHAR | | 1 | | ○ | 3 | | | | | 発生方の貸借 |
| 5 | kingaku | 金額 | NUMERIC | | 13 | 0 | ○ | | | | | | |
| 6 | aite_kamokucd | 相手科目コード | CHAR | | 6 | | ○ | | | | | 2 | 複数の場合「諸口」を設定 |
| 7 | | | | | | | | | | | | | |

## テーブル仕様書

| テーブル名 | 勘定科目マスタ | 作成 | | 担当 | | 項目数 |
|---|---|---|---|---|---|---|
| TABLE_NAME | m_kamoku | 修正 | | 担当 | | 5 |

〔備考〕
勘定科目の情報を管理する

| インデックス | キー名 | 一意 | 外部キー | 制約名 | 参照先 |
|---|---|---|---|---|---|
| p | pk_m_kamoku | ○ | 1 | | |
| s1 | | | 2 | | |
| s2 | | | 3 | | |
| s3 | | | 4 | | |

| No. | FIELD_NAME | 日本語名 | 型 | 日本語 | 桁数 | 小数 | 必須 | IDX P | IDX s1 | IDX s2 | IDX s3 | 外部キー | 備考 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | kamokucd | 科目コード | CHAR | | 6 | | ○ | 1 | | | | | |
| 2 | kamokunm | 科目名 | VARCHAR | ○ | 30 | | ○ | | | | | | |
| 3 | dennyukb | 伝票入力可否区分 | CHAR | | 1 | | ○ | | | | | | 1:伝票入力有り、0:無し |
| 4 | taisyakukb | 貸借区分 | CHAR | | 1 | | ○ | | | | | | 1:借方、2:貸方 |
| 5 | chohyokb | 対象帳票区分 | CHAR | | 1 | | ○ | | | | | | 1:貸借、2:損益、3:利益処分 |
| 6 | | | | | | | | | | | | | |

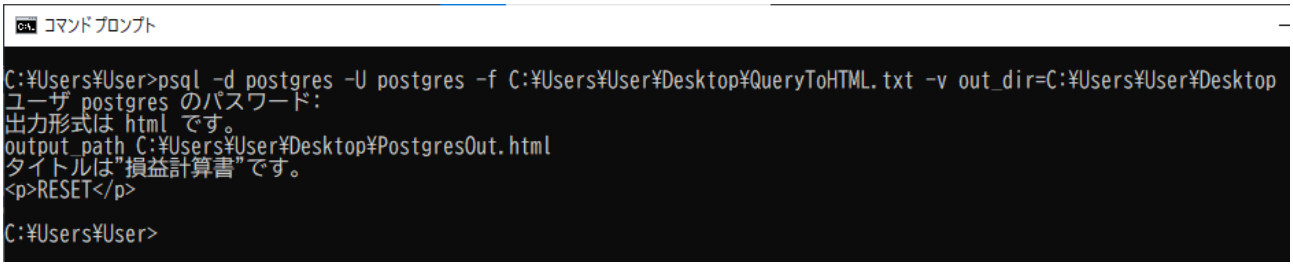
# RDB サーバと CLI ツールで帳票作成（PSQL）

## 2.1. クエリの実行

psql のコマンドと SQL のクエリはファイルに記述して psql の実行時パラメータとして指定します。

この例では出力先をパラメータとして追加している（-v パラメータ）ので、実行コマンドは以下のようになります。(実行時に postgres の利用者パスワードの問合せがあるので入力します)

```
psql -d <database> -U <username> -f <query.sql> -v out_dir=<出力先ディレクトリ>|省略
```




＜query.sql ファイルの内容＞

```
--HTML 形式、文字化けするためフッターの行数を非表示で出力する
¥pset format html
--¥pset footer off

--出力するデータの文字コード…コメント化：デフォルトの SJIS で出力
--¥encoding UTF-8

--out_dir 変数（入力パラメータ）が設定されていたら出力先にする
¥if :{?out_dir}
   ¥cd :out_dir
   ¥set out_path :out_dir'¥¥PostgresOut.html'
¥else
   ¥set out_path 'c:¥¥tmp¥¥PostgresOut.html'
¥endif
¥echo output_path :out_path

--出力先ファイルを指定
¥out :out_path

-- 出力タイトル
¥C 損益計算書

-- ①＜細目：ない場合もある＞科目コード毎に益（貸方）－損（借方）で集計
WITH meisai AS(
    SELECT
          kamokucd
        , kamokunm
        ,(
          COALESCE((SELECT SUM(kingaku) * -1 FROM  t_siwake_meisai
                    WHERE kamokucd = k.kamokucd
                    AND   taisyakukb = '1'
          ),0)
          +  COALESCE((SELECT SUM(kingaku)      FROM  t_siwake_meisai
                    WHERE kamokucd = k.kamokucd
```

```
                        AND   taisyakukb = '2'
          ),0)
        ) AS kingaku
        , FALSE AS kei

    FROM   m_kamoku AS k
    WHERE  kamokucd NOT LIKE ('%00') --細分類のコードを選択
      AND  chohyokb = '2'
)

-- ②<小分類>益（貸方）－損（借方）を集計した作業テーブルを作成する
, L3 AS(
    SELECT
          kamokucd
        , kamokunm
        , CASE
            WHEN EXISTS (
                SELECT kamokucd FROM m_kamoku
                WHERE  SUBSTR(kamokucd, 1, 4) = SUBSTR(k.kamokucd, 1, 4)
                  AND  kamokucd <> k.kamokucd

            ) THEN (  --小分類が存在する場合は、meisai から集計
                SELECT SUM(kingaku) FROM meisai
                WHERE  SUBSTR(kamokucd, 1, 4) = SUBSTR(k.kamokucd, 1, 4)

            ) ELSE (  --小分類が存在しない場合は、仕訳明細から集計
                 COALESCE((SELECT SUM(kingaku) * -1 FROM  t_siwake_meisai
                           WHERE  kamokucd = k.kamokucd
                             AND  taisyakukb = '1'
                 ),0)
              +  COALESCE((SELECT SUM(kingaku)      FROM  t_siwake_meisai
                           WHERE  kamokucd = k.kamokucd
                             AND  taisyakukb = '2'
                 ),0)
            )
            END AS kingaku
        , CASE
            WHEN EXISTS (
                SELECT kamokucd FROM m_kamoku
                WHERE  SUBSTR(kamokucd, 1, 4) = SUBSTR(k.kamokucd, 1, 4)
                  AND  kamokucd <> k.kamokucd
            ) THEN TRUE ELSE FALSE END AS kei

    FROM   m_kamoku AS k
    WHERE  kamokucd LIKE ('%00')
      AND  kamokucd NOT LIKE ('%0000') --小分類のコード以外を除外
      AND  chohyokb = '2'
)

-- ③小分類の作業テーブルから中分類に集計した L2 作業テーブルを作る
, L2 AS(
```

```
    SELECT kamokucd
         , kamokunm
         , (
              SELECT SUM(kingaku) FROM L3
              WHERE  SUBSTR(L3.kamokucd, 1, 2) = SUBSTR(k.kamokucd, 1, 2)
           ) AS kingaku
         , TRUE AS kei
    FROM   m_kamoku AS k
    WHERE  kamokucd LIKE ('%0000')
      AND  kamokucd NOT LIKE ('%00000')  --中分類のコード以外を除外
      AND  chohyokb = '2'
)

-- ④大分類～小分類を連合して出力形式に項目を並べる
SELECT kamokucd AS 科目コード
     , kamokunm AS 科目名
     , CASE  allrec.kei WHEN  TRUE
         THEN kingaku ELSE null
       END AS 計
     , CASE  allrec.kei WHEN  TRUE
         THEN null ELSE kingaku
       END AS 金額
FROM(
-- ⑤中分類の L2 作業テーブルから大分類に集計する
    SELECT kamokucd
         , kamokunm
         , (
              SELECT SUM(kingaku) FROM L2
              WHERE  SUBSTR(kamokucd, 1, 1) = SUBSTR(k.kamokucd, 1, 1)
           ) AS kingaku
         , TRUE AS kei
    FROM   m_kamoku AS k
    WHERE  kamokucd LIKE ('%00000')
           AND  chohyokb = '2'

    -- 明細と中分類の作業テーブルも出力して科目コード順に並べる
    UNION ALL
    SELECT kamokucd, CONCAT('|', kamokunm)     AS kamokunm, kingaku, kei FROM L2
    UNION ALL
    SELECT kamokucd, CONCAT('||', kamokunm)    AS kamokunm, kingaku, kei FROM L3
    UNION ALL
    SELECT kamokucd, CONCAT('|||', kamokunm) AS kamokunm, kingaku, kei FROM meisai
    ORDER BY kamokucd
) AS allrec
;

--出力先をデフォルト（ターミナル）に戻す
¥out
--出力するデータの文字コードをデフォルトに変更
reset client_encoding;
```

## 2.2. 出力編集

数値のカンマ編集を行う場合は以下の部分で文字列編集の関数を使います。

```
-- ④大分類～小分類を連合して出力形式に項目を並べる
SELECT kamokucd AS 科目コード
     , kamokunm AS 科目名
     , CASE  allrec.kei WHEN  TRUE
         THEN kingaku ELSE null
       END AS 計
     , CASE  allrec.kei WHEN  TRUE
         THEN null ELSE kingaku
       END AS 金額
FROM(
↓
SELECT kamokucd AS 科目コード
     , kamokunm AS 科目名
     , CASE  allrec.kei WHEN  TRUE
         THEN TO_CHAR(kingaku, '999,999,999,999') ELSE ''
       END AS 計
     , CASE  allrec.kei WHEN  TRUE
         THEN '' ELSE TO_CHAR(kingaku, '999,999,999,999')
       END AS 金額
FROM(
```

【注意】数値に対して出力編集を行うと文字列の扱いになって左寄せになります。編集時に左側に空白が詰められますが、一般的な日本語フォントだと空白の間隔が詰まって綺麗に揃いません

ファイル | C:/Users/User/Desktop/PostgresOut.html

損益計算書

| 科目コード | 科目名 | 計 | 金額 |
|---|---|---|---|
| 800000 | 経常損益 | 990,000,000 | |
| 810000 | ｜売上高 | 10,990,000,000 | |
| 810100 | ｜｜売上高 | 9,990,000,000 | |
| 810101 | ｜｜｜売上高（売上控除高を除く） | | 10,000,000,000 |
| 810109 | ｜｜｜売上控除高 | | -10,000,000 |
| 810200 | ｜｜役務収益 | | 1,000,000,000 |
| 820000 | ｜売上原価 | -10,000,000,000 | |
| 820100 | ｜｜期首商品・製品たな卸高 | | -1,000,000,000 |
| 820200 | ｜｜当期仕入高 | -9,900,000,000 | |
| 820201 | ｜｜｜当期商品仕入高（仕入控除高を除く） | | -10,000,000,000 |
| 820209 | ｜｜｜仕入控除高 | | 100,000,000 |
| 820300 | ｜｜当期製品製造原価 | | -100,000,000 |
| 820400 | ｜｜期末商品・製品たな卸高 | | 1,000,000,000 |
| 830000 | ｜販売費及び一般管理費 | 0 | |

＜生成された html＞

```
<table border="1">
  <caption>損益計算書</caption>
  <tr>
    <th align="center">科目コード</th>
    <th align="center">科目名</th>
    <th align="center">計</th>
    <th align="center">金額</th>
  </tr>
  <tr valign="top">
    <td align="left">800000</td>
    <td align="left">経常損益</td>
    <td align="left">     990,000,000</td>
    <td align="left">  </td>
  </tr>
  <tr valign="top">
    <td align="left">810000</td>
    <td align="left">｜売上高</td>
    <td align="left">  10,990,000,000</td>
    <td align="left">  </td>
  </tr>
```

（以下略）

編集後は文字列として左寄せ（align=”left”）の設定になるので、これを align=”right”に編集すれば右寄せで出力ができます。

```
<table border="1">↵
  <caption>損益計算書</caption>↵
  <tr>↵
    <th align="center">科目コード</th>↵
    <th align="center">科目名</th>↵
    <th align="center">計</th>↵
    <th align="center">金額</th>↵
  </tr>↵
  <tr valign="top">↵
    <td align="left">800000</td>↵
    <td align="left">経常損益</td>↵
    <td align="right">     990,000,000</td>↵
    <td align="right">  </td>↵
  </tr>↵
  <tr valign="top">↵
    <td align="left">810000</td>↵
    <td align="left">｜売上高</td>↵
    <td align="right">  10,990,000,000</td>↵
    <td align="right">  </td>↵
  </tr>↵
```

損益計算書

| 科目コード | 科目名 | 計 | 金額 |
|---|---|---|---|
| 800000 | 経常損益 | 990,000,000 | |
| 810000 | ｜売上高 | 10,990,000,000 | |
| 810100 | ｜｜売上高 | 9,990,000,000 | |
| 810101 | ｜｜｜売上高（売上控除高を除く） | | 10,000,000,000 |
| 810109 | ｜｜｜売上控除高 | | -10,000,000 |
| 810200 | ｜｜役務収益 | | 1,000,000,000 |
| 820000 | ｜売上原価 | -10,000,000,000 | |
| 820100 | ｜｜期首商品・製品たな卸高 | | -1,000,000,000 |
| 820200 | ｜｜当期仕入高 | -9,900,000,000 | |
| 820201 | ｜｜｜当期商品仕入高（仕入控除高を除く） | | -10,000,000,000 |

# RDB サーバと CLI ツールで帳票作成（PSQL）

## 3. 集約関数を利用した１：N の関連付け出力例

集約関数はグルーピングした複数行から集約した一つの値を求める関数で SUM, COUNT, MIN, MAX 等の数値計算がよく使われます。同様に文字列を一つの値（列）に集約する関数もあり、SQL:2016 で「LISTAGG: 行のグループの値を区切り文字で区切られた文字列に変換する関数」が定義されています。これも製品の実装が標準化に先行したため RDBMS によって関数名が異なっている場合があり、PostgreSQL の場合は STRING_AGG です。下図は仕訳共通に関係する n 件の仕訳明細を string_agg(借)と string_agg(貸)のセルに集約しています

伝票（String_agg）

| trdate | denpyono | tantou | tekiyo | sysdate | string_agg（借） | string_agg（貸） |
|---|---|---|---|---|---|---|
| 20230601 | 00001 | 1000 | 摘要 | 20230811 | 1,0,110101,1000000<br>1,1,110102,10000000<br>1,2,120100,500000000 | 2,0,120100,1000000<br>2,1,120200,10000000 |
| 20230602 | 00002 | 2000 | 摘要 | 20230811 | 1,0,110101,2000000<br>1,1,110102,20000000 | 2,0,120100,2000000<br>2,1,120200,20000000 |
| 20230801 | 00003 | 1000 | 損益計算書,出力用 | 20230802 | 1,0,810109,10000000<br>1,1,820100,1000000000<br>1,2,820201,10000000000<br>1,3,820300,100000000 | 2,0,810101,100000000000<br>2,1,810200,1000000000<br>2,2,820209,100000000<br>2,3,820400,1000000000 |

(3 行)

＜クエリの内容＞ クエリの実行方法は前例の項番 2.1 と同様です。

```
-- 出力タイトル
¥C 伝票（String_agg）
select k.*
, (select STRING_AGG(
           taisyakukb||','||gyo||','||kamokucd||','||kingaku,chr(10)
           order by taisyakukb, gyo
          )
     from t_siwake_meisai m
     where m.denpyono=k.denpyono
       and taisyakukb='1'
  )  as STRING_AGG（借）
, (select STRING_AGG(
           taisyakukb||','||gyo||','||kamokucd||','||kingaku,chr(10)
           order by taisyakukb, gyo
          )
     from t_siwake_meisai m
     where m.denpyono=k.denpyono
       and taisyakukb='2'
  ) as STRING_AGG（貸）
 from t_siwake_kyotu k
;
```

（補足）この例では区切り文字に改行コード chr(10)〔html では<br /> になります〕を使って複数行に見せていますがセル内は一つの値です。また「||」は Oracle と互換の文字列連結で CONCAT()を使っても同じ結果が得られます

## 4. 性能面を考慮したクエリ

SQL は同一の結果を複数の書き方で実現できますが、書き方が性能に影響します。
（実際はインデックスの状態と RDBMS が行うクエリの最適化で大きく変わります）

### （1） 多段階集計の例

この例では科目の SELECT 1 件に対して仕訳明細の全件 SELECT が貸方・借方の 2 回発生する可能性がありますが、テーブル結合にすると科目全件＋仕訳明細全件×貸借の 2 回になります。

```
-- ①<細目：ない場合もある>科目コード毎に益（貸方）－損（借方）で集計
WITH meisai AS(
    SELECT
           kamokucd
         , kamokunm
         ,(
           COALESCE((SELECT SUM(kingaku) * -1 FROM  t_siwake_meisai
                     WHERE kamokucd = k.kamokucd
                     AND   taisyakukb = '1'
           ),0)
           +  COALESCE((SELECT SUM(kingaku)      FROM  t_siwake_meisai
                     WHERE kamokucd = k.kamokucd
                     AND   taisyakukb = '2'
           ),0)
         ) AS kingaku
         , FALSE AS kei

    FROM   m_kamoku AS k
    WHERE  kamokucd NOT LIKE ('%00') --細分類のコードを選択
      AND  chohyokb = '2'
)
```

＜対策：仕訳明細を科目コード×貸借でグルーピングして集計し、科目マスタに結合する＞

```
-- ①<細目：ない場合もある>科目コード毎に益（貸方）－損（借方）で集計
WITH meisai AS(
    SELECT
        k.kamokucd
      , k.kamokunm
      , COALESCE(m2.kingaku, 0) - COALESCE(m1.kingaku, 0) AS kingaku
      , FALSE AS kei
    FROM   m_kamoku AS k
       LEFT OUTER JOIN (
         SELECT kamokucd, SUM(kingaku) AS kingaku from t_siwake_meisai
         WHERE taisyakukb='1'
         GROUP BY kamokucd
       ) AS m1 ON m1.kamokucd = k.kamokucd
       LEFT OUTER JOIN (
         SELECT kamokucd, SUM(kingaku) AS kingaku from t_siwake_meisai
         WHERE taisyakukb='2'
         GROUP BY kamokucd
       ) AS m2 ON m2.kamokucd = k.kamokucd
    WHERE  k.kamokucd NOT LIKE ('%00') --細分類のコードを選択
      AND  chohyokb = '2'
)
```

（2） 1:N の関連付けの例

仕訳共通 1 件に対して 仕訳明細のテーブルを貸方・借方の 2 回 全件 SELECT してしまい効率が悪くなる可能性があります。全てのデータを結合してから加工することで IO を減らすことができます。以下の例では仕訳共通に対して最初に借方の仕訳明細を結合し、その結果に貸方の仕訳明細を結合する 2 段階にしています。

＜クエリ部分の内容＞

```
-- 出力タイトル
¥C 伝票（String_agg）
select k2.*
    ,  STRING_AGG(
            m2.taisyakukb||','||m2.gyo||','||m2.kamokucd||','||m2.kingaku,chr(10)
            order by m2.taisyakukb, m2.gyo
       ) as ARRAY_AGG（貸）
from(
    select k.*
         , STRING_AGG(
            taisyakukb||','||gyo||','||kamokucd||','||kingaku,chr(10)
            order by taisyakukb, gyo
          ) as ARRAY_AGG（借）
    from t_siwake_kyotu k left outer join t_siwake_meisai m
         on m.denpyono=k.denpyono
    where taisyakukb='1'
    group by trdate,k.denpyono,tantou,tekiyo,sysdate
) as k2 left outer join t_siwake_meisai m2
          on m2.denpyono=k2.denpyono
    where m2.taisyakukb='2'
    group by k2.trdate,k2.denpyono,k2.tantou,k2.tekiyo,k2.sysdate,k2.ARRAY_AGG（借）
;
```

※こちらのは仕訳共通に対する仕訳明細の結合（貸、借の 2 回）を外部結合にすることで仕訳共通のデータが欠落する可能性を消しています

以上