ライブツアーのセットリストを公演毎に横に並べる SQL動的生成□UNION利用編

2015/12/13
　某□UNION使う方法もどうせなんだし載せちゃおうぜ！』
　私『…はい』

どうしても横方向が可変の出力に対応したい場合は、動的なSQL組み立てを行って対応します。
昔いた職場のメインフレーム上のRDBMSへSQLを発行する時によく使ってた手です。

ライブツアーのセットリストを公演毎に横に並べるのデータを使って説明します。※テーブル events に項目 stage が追加されている状態のデータを使います。

基本的な考えかた

固定なら難しくないのですが、可変の場合□SQLの世界だけでやるのは困難かと思います。
いまなら□PivotやCrosstabの様な構文があるでしょうけど、それでも面倒で困難な事には変わらないかなと思われます。

そんなめんどうな事をするぐらいなら、自分に都合のよいSQLを動的生成し、それを発行してしまえばよいのです。
何項目を並べるかわからない？ 何レコードあるか不明□ HAHAHA□□SELECT COUNT(*)でカウントしちゃえばいいじゃないですか。目的のSQL実行前に事前確認の為のSQL実行しちゃいけないなんて決まりはないはずです。

簡単なSQLを発行すれば、項目をいくつ横に繋ぐかが判ります。データを取得する本命のSQLを生成する時に、取得個数に合わせて項目を増やせばいいんです。

プログラムを作る業界においては、困難より容易な方で対処するのが美徳でございます。ですので□SQLのみ」で解決するのではなく、対応が容易な「SQLを発行する側」で対処するのです。
※「容易だけど無駄にコスト(時間)のかかる」方法、の意ではないので間違えないでください。

実行結果とSQL生成ロジック

tour_id=227が指定されたものとして□SQLの生成ロジックと実行結果を示します。このidはライブツアーのセットリストを公演毎に横に並べるのb.plが受け取るCGIのパラメタに相当します。そしてうちの環境下で tour_id=227 は「NANA MIZUKI LIVE SENSATION 2003□に該当します。

実行結果

生成されたSQLの実行結果はこうなります。

ans.txt

```
                col0                |              col1               |
col2                |              col3               |              col4
```

col5	col6	col7	col8	col9
2003-07-19	2003-07-21	2003-07-26	2003-07-27	2003-08-03	2003-08-22	2003-08-24	2003-08-25	2003-08-27	2003-08-31
宮城	北海道	大阪	福岡	東京	埼玉	愛知	京都	愛媛	東京
Zepp Sendai	Zepp Sapporo	Zepp Osaka	Zepp Fukuoka	Zepp Tokyo	川口**リリアメインホール**	愛知県勤労会館	京都会館第二**ホール**	松山市民会館中**ホール**	渋谷公会堂
TRANSMIGRATION	TRANSMIGRATION	TRANSMIGRATION	TRANSMIGRATION	TRANSMIGRATION	TRANSMIGRATION	TRANSMIGRATION	TRANSMIGRATION	TRANSMIGRATION	TRANSMIGRATION
アノネ〜まみむめ☆もがちょ〜	**アノネ〜まみむめ☆もがちょ〜**	**アノネ〜まみむめ☆もがちょ〜**	**アノネ〜まみむめ☆もがちょ〜**	**アノネ〜まみむめ☆もがちょ**	The place of happiness	The place of happiness	The place of happiness	The place of happiness	The place of happiness
Heaven Knows	Heaven Knows	Heaven Knows	Heaven Knows	Heaven Knows	**おんなになあれ**	**おんなになあれ**	**おんなになあれ**	**おんなになあれ**	**おんなになあれ**
リプレイマシン -custom-	**リプレイマシン -custom-**	**リプレイマシン -custom-**	**リプレイマシン -custom-**	**リプレイマシン -custom-**	**リプレイマシン -custom-**	**リプレイマシン -custom-**	**リプレイマシン -custom-**	**リプレイマシン -custom-**	**リプレイマシン -custom-**
HONEY FLOWER	HONEY FLOWER	HONEY FLOWER	HONEY FLOWER	HONEY FLOWER	HONEY FLOWER	HONEY FLOWER	HONEY FLOWER	HONEY FLOWER	HONEY FLOWER
恋してる…	**恋してる…**	**恋してる…**	**恋してる…**	**恋してる…**	**恋してる…**	**恋してる…**	**恋してる…**	**恋してる…**	**恋してる…**
二人のMemory	二人のMemory	二人のMemory	二人のMemory	二人のMemory	二人のMemory	二人のMemory	二人のMemory	二人のMemory	二人のMemory
想い	**想い**	**想い**	**想い**	**想い**	**想い**	**想い**	**想い**	**想い**	**想い**

```
| STAND                          | STAND                          | STAND
| STAND                          | STAND
 POWER GATE                          | POWER GATE                          |
POWER GATE                     | POWER GATE                     | POWER
GATE                    | POWER GATE                     | POWER GATE
| POWER GATE                     | POWER GATE                     | POWER
GATE
 What cheer□                         | What cheer□                         | What
cheer□                  | What cheer□                  | What cheer□
| What cheer□                         | What cheer□                         | What
cheer□                  | What cheer□                  | What cheer□
 Year□めっちゃホリディ(松浦亜弥) | JUST COMMUNICATION(TWO-MIX)  | Nothern
lights(林原めぐみ) ¦川の流れのように(美空ひばり) | VALENTI(BoA)
| 明日へのbrilliant road(angela) | Blue Water(森川美穂)   |残酷な天使のテー
ゼ(高橋洋子)|輪舞-revolution(奥井雅美) |天城越え(石川さゆり)
空と心と…           ¦空と心と…           ¦空と心と…           ¦空と心と…
¦空と心と…          ¦テルミドール          ¦テルミドール          ¦テルミドール
¦テルミドール          ¦テルミドール
フリースタイル          ¦フリースタイル          ¦フリースタイル          ¦フリースタイ
ル          ¦フリースタイル                | Heaven Knows                   |
Heaven Knows                | Heaven Knows                | Heaven
Knows                  | Heaven Knows
ジュリエット          ¦ジュリエット          ¦ジュリエット          ¦ジュリエット
¦ジュリエット          ¦想い             ¦想い             ¦想い             ¦想
い             ¦想い
 still in the groove             | still in the groove          |
still in the groove          | still in the groove          | still in
the groove          | still in the groove            | still in the
groove          | still in the groove          | still in the groove
| still in the groove
 supersonic girl                 | supersonic girl             |
supersonic girl             | supersonic girl             |
supersonic girl             | supersonic girl               |
supersonic girl             | supersonic girl             |
supersonic girl             | supersonic girl
 Suddenly□巡り合えて□          | Suddenly□巡り合えて□       | Suddenly□巡
り合えて□       | Suddenly□巡り合えて□       | Suddenly□巡り合えて□       |
Suddenly□巡り合えて□         | Suddenly□巡り合えて□       | Suddenly□巡り合
えて□       | Suddenly□巡り合えて□       | Suddenly□巡り合えて～
 New Sensation                   | New Sensation                | New
Sensation               | New Sensation               | New Sensation
| New Sensation                   | New Sensation                | New
Sensation               | New Sensation               | New Sensation
 refrain                         | refrain                         |
refrain                    | refrain                    | refrain
| refrain                    | refrain                    |
refrain                    | refrain                    | refrain
 ---                             | ---                             | ---
| ---                           | ---                           | ---
| ---                           | ---                           | ---
| ---
 The place of happiness          | The place of happiness       | The
```

```
place of happiness       | The place of happiness       | The place of
happiness       | アノネ〜まみむめ☆もがちょ〜 ¦アノネ〜まみむめ☆もがちょ〜¦ア
ノネ〜まみむめ☆もがちょ〜¦アノネ〜まみむめ☆もがちょ〜¦アノネ〜まみむめ☆もがちょ〜
 PROTECTION                          | PROTECTION                          |
PROTECTION                        | PROTECTION                        |
PROTECTION                        | PROTECTION                          |
PROTECTION                        | PROTECTION                        |
PROTECTION                        | PROTECTION
 ---                                 | ---                                 | ---
| ---                               | ---                                 | ---
| ---                               | ---                                 | ---
| ---
                                    | New Sensation                       |
|                                   | New Sensation                       |
| New Sensation                     | 六甲おろし          ¦天城越え(石川さゆり)
| New Sensation
(28 行)

nanamizuki=>
```

生成されたSQL

生成されたSQLは以下になります。

```
$ perl gensql.pl
-- 公演日の出力
SELECT CAST(MAX(CASE stage WHEN 1 THEN event_date ELSE NULL END) AS VARCHAR)
AS "col0"
,CAST(MAX(CASE stage WHEN 2 THEN event_date ELSE NULL END) AS VARCHAR) AS
"col1"
,CAST(MAX(CASE stage WHEN 3 THEN event_date ELSE NULL END) AS VARCHAR) AS
"col2"
,CAST(MAX(CASE stage WHEN 4 THEN event_date ELSE NULL END) AS VARCHAR) AS
"col3"
,CAST(MAX(CASE stage WHEN 5 THEN event_date ELSE NULL END) AS VARCHAR) AS
"col4"
,CAST(MAX(CASE stage WHEN 6 THEN event_date ELSE NULL END) AS VARCHAR) AS
"col5"
,CAST(MAX(CASE stage WHEN 7 THEN event_date ELSE NULL END) AS VARCHAR) AS
"col6"
,CAST(MAX(CASE stage WHEN 8 THEN event_date ELSE NULL END) AS VARCHAR) AS
"col7"
,CAST(MAX(CASE stage WHEN 9 THEN event_date ELSE NULL END) AS VARCHAR) AS
"col8"
,CAST(MAX(CASE stage WHEN 10 THEN event_date ELSE NULL END) AS VARCHAR) AS
"col9"
 FROM events WHERE tour_id=227
-- 開催場所の出力
 UNION ALL
```

```
 SELECT MAX(CASE t1.stage WHEN 1 THEN t2.place ELSE NULL END)
,MAX(CASE t1.stage WHEN 2 THEN t2.place ELSE NULL END)
,MAX(CASE t1.stage WHEN 3 THEN t2.place ELSE NULL END)
,MAX(CASE t1.stage WHEN 4 THEN t2.place ELSE NULL END)
,MAX(CASE t1.stage WHEN 5 THEN t2.place ELSE NULL END)
,MAX(CASE t1.stage WHEN 6 THEN t2.place ELSE NULL END)
,MAX(CASE t1.stage WHEN 7 THEN t2.place ELSE NULL END)
,MAX(CASE t1.stage WHEN 8 THEN t2.place ELSE NULL END)
,MAX(CASE t1.stage WHEN 9 THEN t2.place ELSE NULL END)
,MAX(CASE t1.stage WHEN 10 THEN t2.place ELSE NULL END)
 FROM events t1 LEFT OUTER JOIN places t2 ON (t1.place_id=t2.id) WHERE
tour_id=227
-- 会場の出力
 UNION ALL
 SELECT MAX(CASE t1.stage WHEN 1 THEN t2.venue ELSE NULL END)
,MAX(CASE t1.stage WHEN 2 THEN t2.venue ELSE NULL END)
,MAX(CASE t1.stage WHEN 3 THEN t2.venue ELSE NULL END)
,MAX(CASE t1.stage WHEN 4 THEN t2.venue ELSE NULL END)
,MAX(CASE t1.stage WHEN 5 THEN t2.venue ELSE NULL END)
,MAX(CASE t1.stage WHEN 6 THEN t2.venue ELSE NULL END)
,MAX(CASE t1.stage WHEN 7 THEN t2.venue ELSE NULL END)
,MAX(CASE t1.stage WHEN 8 THEN t2.venue ELSE NULL END)
,MAX(CASE t1.stage WHEN 9 THEN t2.venue ELSE NULL END)
,MAX(CASE t1.stage WHEN 10 THEN t2.venue ELSE NULL END)
 FROM events t1 LEFT OUTER JOIN venues t2 ON (t1.venue_id=t2.id) WHERE
tour_id=227
 UNION ALL
 SELECT '---','---','---','---','---','---','---','---','---','---'
-- セットリストの出力
 UNION ALL
 SELECT s1.col0
,s1.col1
,s1.col2
,s1.col3
,s1.col4
,s1.col5
,s1.col6
,s1.col7
,s1.col8
,s1.col9
 FROM (SELECT MAX(CASE t1.stage WHEN 1 THEN t3.song_title ELSE NULL END) AS
"col0"
,MAX(CASE t1.stage WHEN 2 THEN t3.song_title ELSE NULL END) AS "col1"
,MAX(CASE t1.stage WHEN 3 THEN t3.song_title ELSE NULL END) AS "col2"
,MAX(CASE t1.stage WHEN 4 THEN t3.song_title ELSE NULL END) AS "col3"
,MAX(CASE t1.stage WHEN 5 THEN t3.song_title ELSE NULL END) AS "col4"
,MAX(CASE t1.stage WHEN 6 THEN t3.song_title ELSE NULL END) AS "col5"
,MAX(CASE t1.stage WHEN 7 THEN t3.song_title ELSE NULL END) AS "col6"
,MAX(CASE t1.stage WHEN 8 THEN t3.song_title ELSE NULL END) AS "col7"
,MAX(CASE t1.stage WHEN 9 THEN t3.song_title ELSE NULL END) AS "col8"
,MAX(CASE t1.stage WHEN 10 THEN t3.song_title ELSE NULL END) AS "col9"
```

```
 FROM events t1 LEFT       JOIN setlists t2 ON (t1.id     =t2.event_id)
                                 LEFT OUTER JOIN songs    t3 ON
(t2.song_id=t3.id)
                   WHERE tour_id=227
                     AND t2.list_type=1
                   GROUP BY t2.order_index
                   ORDER BY t2.order_index) s1
 UNION ALL
 SELECT '---','---','---','---','---','---','---','---','---','---'
-- アンコールの出力
 UNION ALL
 SELECT s1.col0
,s1.col1
,s1.col2
,s1.col3
,s1.col4
,s1.col5
,s1.col6
,s1.col7
,s1.col8
,s1.col9
 FROM (SELECT MAX(CASE t1.stage WHEN 1 THEN t3.song_title ELSE NULL END) AS
"col0"
,MAX(CASE t1.stage WHEN 2 THEN t3.song_title ELSE NULL END) AS "col1"
,MAX(CASE t1.stage WHEN 3 THEN t3.song_title ELSE NULL END) AS "col2"
,MAX(CASE t1.stage WHEN 4 THEN t3.song_title ELSE NULL END) AS "col3"
,MAX(CASE t1.stage WHEN 5 THEN t3.song_title ELSE NULL END) AS "col4"
,MAX(CASE t1.stage WHEN 6 THEN t3.song_title ELSE NULL END) AS "col5"
,MAX(CASE t1.stage WHEN 7 THEN t3.song_title ELSE NULL END) AS "col6"
,MAX(CASE t1.stage WHEN 8 THEN t3.song_title ELSE NULL END) AS "col7"
,MAX(CASE t1.stage WHEN 9 THEN t3.song_title ELSE NULL END) AS "col8"
,MAX(CASE t1.stage WHEN 10 THEN t3.song_title ELSE NULL END) AS "col9"
 FROM events t1 LEFT       JOIN setlists t2 ON (t1.id     =t2.event_id)
                                 LEFT OUTER JOIN songs    t3 ON
(t2.song_id=t3.id)
                   WHERE tour_id=227
                     AND t2.list_type=2
                   GROUP BY t2.order_index
                   ORDER BY t2.order_index) s1
 UNION ALL
 SELECT '---','---','---','---','---','---','---','---','---','---'
-- Wアンコールの出力
 UNION ALL
 SELECT s1.col0
,s1.col1
,s1.col2
,s1.col3
,s1.col4
,s1.col5
,s1.col6
,s1.col7
```

```
,s1.col8
,s1.col9
 FROM (SELECT MAX(CASE t1.stage WHEN 1 THEN t3.song_title ELSE NULL END) AS
"col0"
,MAX(CASE t1.stage WHEN 2 THEN t3.song_title ELSE NULL END) AS "col1"
,MAX(CASE t1.stage WHEN 3 THEN t3.song_title ELSE NULL END) AS "col2"
,MAX(CASE t1.stage WHEN 4 THEN t3.song_title ELSE NULL END) AS "col3"
,MAX(CASE t1.stage WHEN 5 THEN t3.song_title ELSE NULL END) AS "col4"
,MAX(CASE t1.stage WHEN 6 THEN t3.song_title ELSE NULL END) AS "col5"
,MAX(CASE t1.stage WHEN 7 THEN t3.song_title ELSE NULL END) AS "col6"
,MAX(CASE t1.stage WHEN 8 THEN t3.song_title ELSE NULL END) AS "col7"
,MAX(CASE t1.stage WHEN 9 THEN t3.song_title ELSE NULL END) AS "col8"
,MAX(CASE t1.stage WHEN 10 THEN t3.song_title ELSE NULL END) AS "col9"
 FROM events t1 LEFT       JOIN setlists t2 ON (t1.id     =t2.event_id)
                                  LEFT OUTER JOIN songs    t3 ON
(t2.song_id=t3.id)
                   WHERE tour_id=227
                     AND t2.list_type=3
                   GROUP BY t2.order_index
                   ORDER BY t2.order_index) s1
$
```

以下のSQL記述、

```
 SELECT '---','---','---','---','---','---','---','---','---','---'
```

のSELECT文でFROMがないのは、不要ならFROM句がいらないPostgreSQLの仕様です。

Oracleだと FROM DUAL

```
 SELECT '---','---','---','---','---','---','---','---','---','---' FROM
DUAL
```

DB2だとFROM SYSIBM.DUALやFROM SYSIBM.SYSDUMMY1でしょうか。

```
 SELECT '---','---','---','---','---','---','---','---','---','---' FROM
SYSIBM.DUAL
```

生成ロジック

あいかわらず Perl のコードですみません。

SQLの生成までしかやっていないのでb.plの役割とするならHTML生成ロジックを追加することになります。

gensql.pl

```
use Encode;
use DBI;
```

```
my $dbname="nanamizuki";
my $dbhost="your.database.host.addr";
my $dbuser="db user id";
my $dbpass="db password";

my $dbh =
DBI->connect("dbi:Pg:dbname=$dbname;host=$dbhost",$dbuser,$dbpass) or
die  "cannot connect database.";

my $cgi_id = 227;

my $events_count = 0;

  $dbh->{AutoCommit}=0;

  ##指定ツアーの公演回数を取得
  my $sth0 = $dbh->prepare('SELECT count(*) from events WHERE
tour_id=?');
  $sth0->bind_param( 1, $cgi_id);
  $sth0->execute();
  $sth0->bind_col(  1, \$events_count   );
  $sth0->fetch();
  $sth0->finish();
  $dbh->commit();

  ##公演日を横につなげる
  my $sql = 'SELECT ';
  for(my $cnt=0; $cnt<$events_count; $cnt++)
  {
    $sql .=',' if ($cnt != 0);
    $sql .= sprintf("cast(max(case stage when %d then event_date else
null end) as varchar) as \"col%d\"\n", $cnt+1, $cnt);
  }
  $sql .= sprintf(" FROM events WHERE tour_id=%d\n", $cgi_id);

  ##開催場所を横につなげる
  $sql .= " UNION ALL\n SELECT ";
  for(my $cnt=0; $cnt<$events_count; $cnt++)
  {
    $sql .=',' if ($cnt != 0);
    $sql .= sprintf("max(case t1.stage when %d then t2.place else null
end)\n", $cnt+1);
  }
  $sql .= sprintf(" FROM events t1 left outer join places t2 on
(t1.place_id=t2.id) WHERE tour_id=%d\n", $cgi_id);

  ##会場を横につなげる
  $sql .= " UNION ALL\n SELECT ";
  for(my $cnt=0; $cnt<$events_count; $cnt++)
  {
    $sql .=',' if ($cnt != 0);
```

```
    $sql .= sprintf("max(case t1.stage when %d then t2.venue else null
end)\n", $cnt+1);
  }
  $sql .= sprintf(" FROM events t1 left outer join venues t2 on
(t1.venue_id=t2.id) WHERE tour_id=%d\n", $cgi_id);

  ##区切り
  $sql .= " UNION ALL\n SELECT ";
  for(my $cnt=0; $cnt<$events_count; $cnt++)
  {
    $sql .=',' if ($cnt != 0);
    $sql .= "'---'";
  }
  $sql .="\n";

  ##セットリストを横につなげる
  $sql .= " UNION ALL\n SELECT ";
  my $sql2 = "SELECT ";
  for(my $cnt=0; $cnt<$events_count; $cnt++)
  {
    $sql2 .=',' if ($cnt != 0);
    $sql2 .= sprintf("max(case t1.stage when %d then t3.song_title else
null end) as \"col%d\"\n", $cnt+1, $cnt);

    $sql .=',' if ($cnt != 0);
    $sql .= sprintf("s1.col%d\n", $cnt);
  }
  $sql2 .= sprintf(" FROM events t1 left       join setlists t2 on
(t1.id      = t2.event_id)
                                    left outer join songs    t3 on
(t2.song_id = t3.id)
                   WHERE tour_id=%d
                     AND t2.list_type=1
                   GROUP BY t2.order_index
                   ORDER BY t2.order_index", $cgi_id);
  $sql .= " FROM (" . $sql2 . ") s1\n";

  ##区切り
  $sql .= " UNION ALL\n SELECT ";
  for(my $cnt=0; $cnt<$events_count; $cnt++)
  {
    $sql .=',' if ($cnt != 0);
    $sql .= "'---'";
  }
  $sql .="\n";

  ##アンコールの演目を横につなげる
  $sql .= " UNION ALL\n SELECT ";
  $sql2 = "SELECT ";
  for(my $cnt=0; $cnt<$events_count; $cnt++)
  {
```

```
    $sql2 .=',' if ($cnt != 0);
    $sql2 .= sprintf("max(case t1.stage when %d then t3.song_title else
null end) as \"col%d\"\n", $cnt+1, $cnt);

    $sql .=',' if ($cnt != 0);
    $sql .= sprintf("s1.col%d\n", $cnt);
  }
  $sql2 .= sprintf(" FROM events t1 left       join setlists t2 on
(t1.id      = t2.event_id)
                                    left outer join songs    t3 on
(t2.song_id = t3.id)
                   WHERE tour_id=%d
                     AND t2.list_type=2
                   GROUP BY t2.order_index
                   ORDER BY t2.order_index", $cgi_id);
  $sql .= " FROM (" . $sql2 . ") s1\n";

  ##区切り
  $sql .= " UNION ALL\n SELECT ";
  for(my $cnt=0; $cnt<$events_count; $cnt++)
  {
    $sql .=',' if ($cnt != 0);
    $sql .= "'---'";
  }
  $sql .="\n";

  ## Wアンコールの演目を横に並べる
  $sql .= " UNION ALL\n SELECT ";
  $sql2 = "SELECT ";
  for(my $cnt=0; $cnt<$events_count; $cnt++)
  {
    $sql2 .=',' if ($cnt != 0);
    $sql2 .= sprintf("max(case t1.stage when %d then t3.song_title else
null end) as \"col%d\"\n", $cnt+1, $cnt);

    $sql .=',' if ($cnt != 0);
    $sql .= sprintf("s1.col%d\n", $cnt);
  }
  $sql2 .= sprintf(" FROM events t1 left       join setlists t2 on
(t1.id      = t2.event_id)
                                    left outer join songs    t3 on
(t2.song_id = t3.id)
                   WHERE tour_id=%d
                     AND t2.list_type=3
                   GROUP BY t2.order_index
                   ORDER BY t2.order_index", $cgi_id);
  $sql .= " FROM (" . $sql2 . ") s1\n";

  print $sql;
```

```
  $dbh->disconnect();
```

gensql.plのちょっとした説明

やっていることは単純明快で、

- 横に並べる項目数を求める
- 公演日を横に並べるSQL生成
- 開催場所を横に並べるSQL生成
- 会場を横に並べるSQL生成
- セットリストを横に並べるSQL生成
- アンコール演目を横に並べるSQL生成
- Wアンコール演目を横に並べるSQL生成

となっています。最初の項目数を求めたら、必要な項目数だけ追加したSQLを生成し、各SQLをUNION演算子で結合し1本のSQLに合成します。
このSQLの実行結果はあたかもSELECTの結果を縦につないだかのように見えます。

項目数を求める

最初に横に並べなきゃいけない項目数をSQLで求めています。
現在のデータだと、横に展開する時に使う項目stageの最大値と同じになるはずです。

```
  ## 指定ツアーの公演回数を取得
  my $sth0 = $dbh->prepare('SELECT count(*) from events WHERE tour_id=?');
  $sth0->bind_param( 1, $cgi_id);
  $sth0->execute();
  $sth0->bind_col(  1, \$events_count   );
  $sth0->fetch();
  $sth0->finish();
  $dbh->commit();
```

1行出力のSQL生成

出力項目を、求めた個数回だけ生成します。
例えば公演日抽出SQLの生成処理、

```
  ## 公演日を横につなげる
  my $sql = 'SELECT ';
  for(my $cnt=0; $cnt<$events_count; $cnt++)
  {
    $sql .=',' if ($cnt != 0);
    $sql .= sprintf("cast(max(case stage when %d then event_date else null
end) as varchar) as \"col%d\"\n", $cnt+1, $cnt);
  }
  $sql .= sprintf(" FROM events WHERE tour_id=%d\n", $cgi_id);
```

の部分で、以下の選択項目が$events_count個生成されることになります※MM,NNに数値が入ります。

```
CAST(MAX(CASE stage WHEN MM THEN event_date ELSE NULL END) AS VARCHAR) AS
"colNN"
```

他の部分も基本は同じです。

複数行の発生するSQL生成

セットリスト、アンコール演目Wアンコール演目の処理では、サブクエリをつかうSQLを生成しています。
以下はセットリストのSQL生成処理です。

```
  ## セットリストを横につなげる
  $sql .= " UNION ALL\n SELECT ";
  my $sql2 = "SELECT ";
  for(my $cnt=0; $cnt<$events_count; $cnt++)
  {
    $sql2 .=',' if ($cnt != 0);
    $sql2 .= sprintf("max(case t1.stage when %d then t3.song_title else null
end) as \"col%d\"\n", $cnt+1, $cnt);

    $sql .=',' if ($cnt != 0);
    $sql .= sprintf("s1.col%d\n", $cnt);
  }
  $sql2 .= sprintf(" FROM events t1 left       join setlists t2 on (t1.id
=t2.event_id)
                                  left outer join songs    t3 on
(t2.song_id=t3.id)
                   WHERE tour_id=%d
                     AND t2.list_type=1
                   GROUP BY t2.order_index
                   ORDER BY t2.order_index", $cgi_id);
  $sql .= " FROM (" . $sql2 . ") s1\n";
```

二つのSELECT文が生成されているのがわかると思います。これは、セットリストが公演日や公演場所の生成と異なり複数行生成される必要があるためです。 サブクエリでセットリストの演目順序でソートされたレコードセットを生成しています。このレコードセットをこの順番で適用します。

SQL, Perl