# Create a pivot result in mariadb/mysql

This relies heavily on https://stackoverflow.com/questions/15997090/crosstab-view-in-mysql and I even stole the data structure. I just worked to figure out what was going on, then wrote this up. NOTE: it is possible I made a mistake in my interpretation. If so, it is all on me.

A pivot query (aka crosstab) is a type of query where the top, where you normally have column names, is the result of the query. Some database engines have a special *PIVOT* function built in for this, but MySQL/MariaDB is not one of them.

Take this data structure (stolen from Taryn's answer in the above article).

create_test.sql

```
CREATE TABLE clients
    (`clID` INT, `clName` VARCHAR(5))
;

INSERT INTO clients
    (`clID`, `clName`)
VALUES
    (1, 'Chris'),
    (2, 'Gale'),
    (3, 'Donna')
;

CREATE TABLE scores
    (`ID` INT, `clID` INT, `PlayDate` datetime, `Score` NUMERIC(10,5))
;

INSERT INTO scores
    (`ID`, `clID`, `PlayDate`, `Score`)
VALUES
    (1, 2, '2012-01-23 00:00:00', -0.0125),
    (2, 2, '2012-01-24 00:00:00', 0.1011),
    (3, 3, '2012-01-24 00:00:00', 0.0002),
    (4, 3, '2012-01-26 00:00:00', -0.0056),
    (5, 3, '2012-01-27 00:00:00', 0.0001),
    (6, 1, '2012-01-12 00:00:00', 0.0122),
    (7, 1, '2012-01-13 00:00:00', 0.0053)
;
```

A simple query to find everyone's total score summarized by playdate would be

```
SELECT
   clName,
   PlayDate,
   SUM(Score)
```

```
FROM
   clients
   JOIN scores USING (clID)
GROUP BY
   PlayDate,
   clName
ORDER BY
   PlayDate,
   clName;
```

which would return

```
+--------+---------------------+------------+
| clName | PlayDate            | sum(Score) |
+--------+---------------------+------------+
| Chris  | 2012-01-12 00:00:00 |    0.01220 |
| Chris  | 2012-01-13 00:00:00 |    0.00530 |
| Gale   | 2012-01-23 00:00:00 |   -0.01250 |
| Donna  | 2012-01-24 00:00:00 |    0.00020 |
| Gale   | 2012-01-24 00:00:00 |    0.10110 |
| Donna  | 2012-01-26 00:00:00 |   -0.00560 |
| Donna  | 2012-01-27 00:00:00 |    0.00010 |
+--------+---------------------+------------+
```

But, in many cases, a pivot makes it more readable to humans. A pivot of the above data could be

```
+---------------------+--------+---------+---------+
| playdate            | Chris  | Gale    | Donna   |
+---------------------+--------+---------+---------+
| 2012-01-12 00:00:00 | 0.0122 |       0 |       0 |
| 2012-01-13 00:00:00 | 0.0053 |       0 |       0 |
| 2012-01-23 00:00:00 |      0 | -0.0125 |       0 |
| 2012-01-24 00:00:00 |      0 |  0.1011 |  0.0002 |
| 2012-01-26 00:00:00 |      0 |       0 | -0.0056 |
| 2012-01-27 00:00:00 |      0 |       0 |  0.0001 |
+---------------------+--------+---------+---------+
```

If you already know all the values for clName, you can create a static query, creating a column for each of them.

```
SELECT s.playdate,
  SUM(CASE WHEN clname = 'Chris' THEN score END) Chris,
  SUM(CASE WHEN clname = 'Gale' THEN score END) Gale,
  SUM(CASE WHEN clname = 'Donna' THEN score END) Donna
FROM clients c
INNER JOIN scores s
  ON c.clid = s.clid
GROUP BY s.playdate;
```

What this does is creates three columns for the actual score; one for each clName, then only adds the

ones where clName matches. The sql CASE is excellent.

However, if you add a new user, you must change your query. You can do that via a script, by dynamically building the above query, or you can use a prepared statement to do build a dynamic SQL statement all within the database engine.

What we would like to do is create the three lines separating out the scores for Chris, Gale and Donna dynamically, then build the rest of the query around that. That query could look like:

```
SELECT
  GROUP_CONCAT(DISTINCT
    CONCAT(
      'sum(CASE WHEN clName = ''',
      clName,
      ''' THEN score else ''-'' END) AS `',
      clName, '`'
    )
  )
FROM clients;
```

At this point, you have built this part of the query

```
  sum(case when clname = 'Chris' then score end) Chris,
  sum(case when clname = 'Gale' then score end) Gale,
  sum(case when clname = 'Donna' then score end) Donna
```

We use the INTO clause to place that into a variable named @SQL

```
SET @SQL = NULL;
SELECT
  GROUP_CONCAT(DISTINCT
    CONCAT(
      'sum(CASE WHEN clName = ''',
      clName,
      ''' THEN score else ''-'' END) AS `',
      clName, '`'
    )
  ) INTO @SQL
FROM clients;
```

The variable @SQL now contains that information, so we build the query, inserting the value of @sql into it at the appropriate place, then run it.

pivot.sql

```
/* https://stackoverflow.com/questions/15997090/crosstab-view-in-mysql
*/

SET @SQL = NULL;
SELECT
  GROUP_CONCAT(DISTINCT
```

```
    CONCAT(
      'sum(CASE WHEN clName = ''',
      clName,
      ''' THEN score else ''-'' END) AS `',
      clName, '`'
    )
  ) INTO @SQL
FROM clients;

SET @SQL
  = CONCAT('SELECT s.playdate, ', @SQL, '
            from clients c
            inner join scores s
              on c.clid = s.clid
            group by s.playdate');

PREPARE stmt FROM @SQL;
EXECUTE stmt;
DEALLOCATE PREPARE stmt;
```

run the above and you'll see the same result. Now, let's add Mary to the mix and add a couple of her scores.

```
INSERT INTO clients VALUES (4, 'Mary');
INSERT INTO scores (`ID`, `clID`, `PlayDate`, `Score`)
   VALUES
      (8,4,20120124,0.159),
      (9,4,20120113,0.0567),
      (10,4,20120124,0.0125);
```

Running the static example, we still only calculate Chris, Gale and Donna. We would have to add a new clause to get Mary. But, if we run the dynamic query, Mary is automatically added to the report.

```
+---------------------+--------+---------+---------+--------+
| playdate            | Chris  | Gale    | Donna   | Mary   |
+---------------------+--------+---------+---------+--------+
| 2012-01-12 00:00:00 | 0.0122 |       0 |       0 |      0 |
| 2012-01-13 00:00:00 | 0.0053 |       0 |       0 | 0.0567 |
| 2012-01-23 00:00:00 |      0 | -0.0125 |       0 |      0 |
| 2012-01-24 00:00:00 |      0 |  0.1011 |  0.0002 | 0.1715 |
| 2012-01-26 00:00:00 |      0 |       0 | -0.0056 |      0 |
| 2012-01-27 00:00:00 |      0 |       0 |  0.0001 |      0 |
+---------------------+--------+---------+---------+--------+
```

For additional things you can do with pivot's, also see https://armantutorial.wordpress.com/2016/01/08/cross-tab-query-in-mysql/, which goes into calculating sums and multi-aggregate pivots

# Links

- https://stackoverflow.com/questions/15997090/crosstab-view-in-mysql
- https://armantutorial.wordpress.com/2016/01/08/cross-tab-query-in-mysql/

From:
https://kb.unixservertech.com/ - **Unix Server Tech Knowledge Base**

Permanent link:
**https://kb.unixservertech.com/software/mysql/pivots**

Last update: **2022/11/10 05:07**