Giuseppe Maxia

Pivot Tables in MySQL 5
About me

http://datacharmer.org
Agenda

- Introducing pivot tables
- Common solutions
- The SQL way
- Pivot tables with stored procedures
- The OO approach
Introducing pivot tables (1)

• Pivot tables, or CROSSTABS
• Statistical reports
• Spreadsheets tools
• Usually on the client side
• On servers, dedicated tools
• Some vendors have language extensions
• Can be done in standard SQL
Introducing pivot tables (2)

Some online resources

- MySQL wizardry (2001)
- DBIx::SQLCrosstab (2004)
- See the addresses on my site (http://datacharmer.org)
Introducing pivot tables (3)

An example (raw data)

<table>
<thead>
<tr>
<th></th>
<th>person</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>name</td>
<td>gender</td>
<td>dept</td>
</tr>
<tr>
<td>1</td>
<td>John</td>
<td>m</td>
<td>pers</td>
</tr>
<tr>
<td>2</td>
<td>Mario</td>
<td>m</td>
<td>pers</td>
</tr>
<tr>
<td>7</td>
<td>Mary</td>
<td>f</td>
<td>pers</td>
</tr>
<tr>
<td>8</td>
<td>Bill</td>
<td>m</td>
<td>pers</td>
</tr>
<tr>
<td>3</td>
<td>Frank</td>
<td>m</td>
<td>sales</td>
</tr>
<tr>
<td>5</td>
<td>Susan</td>
<td>f</td>
<td>sales</td>
</tr>
<tr>
<td>6</td>
<td>Martin</td>
<td>m</td>
<td>sales</td>
</tr>
<tr>
<td>4</td>
<td>Otto</td>
<td>m</td>
<td>dev</td>
</tr>
<tr>
<td>9</td>
<td>June</td>
<td>f</td>
<td>dev</td>
</tr>
</tbody>
</table>
### Introducing pivot tables (4)

#### An example (crosstab)

<table>
<thead>
<tr>
<th>dept by gender</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>dept</td>
<td>m</td>
</tr>
<tr>
<td>----------------------</td>
<td>--</td>
</tr>
<tr>
<td>dev</td>
<td>1</td>
</tr>
<tr>
<td>pers</td>
<td>3</td>
</tr>
<tr>
<td>sales</td>
<td>2</td>
</tr>
</tbody>
</table>
Introducing pivot tables (5)

Definitions

<table>
<thead>
<tr>
<th>Row headers</th>
<th>Column headers</th>
</tr>
</thead>
<tbody>
<tr>
<td>dept by gender</td>
<td>dept</td>
</tr>
<tr>
<td>dev</td>
<td>1</td>
</tr>
<tr>
<td>pers</td>
<td>3</td>
</tr>
<tr>
<td>sales</td>
<td>2</td>
</tr>
</tbody>
</table>
**Introducing pivot tables (6)**

### Definitions

<table>
<thead>
<tr>
<th>dept by gender</th>
<th>m</th>
<th>f</th>
<th>total</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>dept</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>dev</strong></td>
<td>1</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td><strong>pers</strong></td>
<td>3</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td><strong>sales</strong></td>
<td>2</td>
<td>1</td>
<td>3</td>
</tr>
</tbody>
</table>

- **Row headers**: dev, pers, sales
- **Column headers**: dept
- **Distinct values**
# Introducing pivot tables (7)

## Definitions

<table>
<thead>
<tr>
<th>dept by gender</th>
<th>m</th>
<th>f</th>
<th>total</th>
</tr>
</thead>
<tbody>
<tr>
<td>dev</td>
<td>1</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>pers</td>
<td>3</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>sales</td>
<td>2</td>
<td>1</td>
<td>3</td>
</tr>
</tbody>
</table>

- **Row headers**: dept
- **Column headers**: by gender
- **Calculated values**: m, f, total
- **Distinct values**
Introducing pivot tables (8)

Some important points

- **Row headers** are defined in one go
- **Column headers** are defined in two separate steps.
Common solutions

- The spreadsheets approach
- Using dedicated tools
The spreadsheet approach (1)

- Copy the raw data from the server to a spreadsheet
- Use the Pivot Table feature
The spreadsheet approach (2)
The spreadsheet approach (3)

<table>
<thead>
<tr>
<th></th>
<th>id</th>
<th>name</th>
<th>gender</th>
<th>dept</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>John</td>
<td>m</td>
<td>pers</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Mario</td>
<td>m</td>
<td>pers</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Mary</td>
<td>f</td>
<td>pers</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Bill</td>
<td>m</td>
<td>pers</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Frank</td>
<td>m</td>
<td>sales</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>Susan</td>
<td>f</td>
<td>sales</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>Martin</td>
<td>m</td>
<td>sales</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>Otto</td>
<td>m</td>
<td>dev</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>June</td>
<td>f</td>
<td>dev</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>gender</th>
<th>dept</th>
<th>Total Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>m</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>m</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>3</td>
<td>f</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td>9</td>
</tr>
</tbody>
</table>
The spreadsheet problems (4)

• Copying the raw data could be a long operation

• The spreadsheet can handle a limited number of records

• The results could be different in different clients

• It is not easy to scale
Dedicated tools

**PRO**

- Easy to use
- Rich in features
- Well integrated to BI packages
Dedicated tools

CON

- Client side tools are resource hogs
- Server side tools are either complicated to install and maintain or expensive (or both)
The query for that crosstab

SELECT dept,
    COUNT(CASE WHEN gender = 'm' THEN id ELSE NULL END) AS m,
    COUNT(CASE WHEN gender = 'f' THEN id ELSE NULL END) AS f,
    COUNT(*) AS total
FROM person
GROUP BY dept
The SQL way (2)

The query for that crosstab

```
SELECT dept,
    COUNT(CASE WHEN gender = 'm' THEN id ELSE NULL END) AS m,
    COUNT(CASE WHEN gender = 'f' THEN id ELSE NULL END) AS f,
    COUNT(*) AS total
FROM person
GROUP BY dept
```
The SQL way (3)

Two passes

```sql
SELECT dept,
      COUNT(CASE WHEN gender = 'm' THEN id ELSE NULL END) AS m,
      COUNT(CASE WHEN gender = 'f' THEN id ELSE NULL END) AS f,
      COUNT(*) AS total
FROM person
GROUP BY dept
```

Calculated in this query

Previously determined
The SQL way (4)

Two passes

# 1
SELECT DISTINCT gender FROM person;

<table>
<thead>
<tr>
<th>gender</th>
</tr>
</thead>
<tbody>
<tr>
<td>m</td>
</tr>
<tr>
<td>f</td>
</tr>
</tbody>
</table>

# build the second query

# 2
CROSSTAB QUERY
The SQL way (5)

PROBLEMS

• The two passes are not coordinated
• External languages needed (Perl, PHP, Java, etc.)
• For each language, you need appropriate routines
Pivot tables and stored routines (1)

**PRO**

- Bundling the two steps for crosstab creation
- Adding features easily
- Same interface from different host languages
Pivot tables and stored routines (2)

**CON**

- Less expressive than most languages
- More burden for the server
Methods to gather column headers (1)

- GROUP_CONCAT
- Cursor + temporary table
- Cursor + dedicated routines
GROUP_CONCAT (1)

SELECT GROUP_CONCAT(DISTINCT gender) AS column_list FROM person;

+----------------------+
| column_list          |
+----------------------+
| m,f                  |
+----------------------+
GROUP_CONCAT (2)

SELECT GROUP_CONCAT(DISTINCT CONCAT('nCOUNT(CASE WHEN gender= ",", gender, "," THEN id ELSE NULL END) AS ",", gender)) AS column_list
FROM person

column_list:
COUNT(CASE WHEN gender= "m" THEN id ELSE NULL END) AS m,
COUNT(CASE WHEN gender= "f" THEN id ELSE NULL END) AS f
GROUP_CONCAT (3)

**PRO**

- Easy to use
- Dynamic
- Fast (built-in)
GROUP CONCAT (4)

CON

- Memory limit (1024 bytes)
- Can be increased by setting group_concat_max_len
- But you can't know the length in advance
# outline only
DECLARE dp char(10);
DECLARE column_list TEXT default ";
DECLARE CURSOR get_cols FOR
  SELECT DISTINCT dept FROM person;
OPEN get_cols;
LOOP
  FETCH get_cols INTO dp;
  SET column_list=CONCAT(column_list, 
',', dp);
END LOOP;
Cursor (2)

**PRO**

- No memory limits
- Fast
Cursor (3)

CON

- Not dynamic
-Verbose
Cursor + temp table (1)

# dynamic query
SET @q = CONCAT('CREATE TABLE temp temp
SELECT DISTINCT ', col_name, ', AS mycol
FROM ', table_name);
PREPARE d FROM @q
EXECUTE d;

# routine with cursor
DECLARE dp char(10);
DECLARE column_list TEXT default '
DECLARE CURSOR get_cols FOR
   SELECT mycol FROM temp;
Cursor + temp table (2)

**PRO**

- No memory limits
- dynamic
Cursor + temp table (2)

**CON**

- Requires a new table for each crosstab
- Data needs to be read twice
# Create a routine from a template

# use routine with cursor
DECLARE dp char(10);
DECLARE column_list TEXT default '';
DECLARE CURSOR get_cols FOR
    SELECT gender FROM person;
Cursor + dedicated routine (2)

**PRO**
- No memory limits
- data is read only once
- no temporary tables
Cursor + dedicated routine (3)

**CON**

- Not dynamic
- Dedicated routines need to be created by external language
Cursor + Higher Order Routines (1)

• Higher Order Routines are routines that create other routines

• It IS NOT STANDARD

• It is, actually, a hack
Cursor + Higher Order Routines (2)

• Very UNOFFICIALLY

• You can create a routine from a MySQL routine
Cursor + Higher Order Routines (3)

**PRO**

- Dynamic
- No memory limits
- No temporary tables
- Data is read only once
- Efficient
You need just ONE ROUTINE created with SUPER privileges (access to mysql.proc table)
DISCLAIMER

The **Very Unofficial** method described in this presentation:

- Is **not** recommended by MySQL AB
- Is **not** guaranteed to work in future releases of MySQL
- May result in any number of damages to your data, from bird flu to accidental explosions.
- You use it at your own risk
Cursor + Higher Order Routines (5)

**HOW**

- Download the code (all SQL)
- [http://datacharmer.org](http://datacharmer.org)
- Unpack and run

```
mysql -t < crosstab_install.mysql
```

- then use it

```
mysql> CALL crosstab_help()
```
Crosstab package (1)

CALL crosstab_help()

type "SELECT crosstab_usage()\G for general usage"

type "SELECT crosstab_example()\G" for a few examples

type "SELECT crosstab_env()\G" for modifiers and debugging
Crosstab package (2)

SELECT crosstab_usage()\G

== CROSSTAB USAGE ==

=================================

PROCEDURE get_crosstab_simple (row_name varchar(50),
-- the field that identifies each row
col_name varchar(50),
-- from which column we spread the values
op varchar(10),
-- which operation (COUNT, SUM, AVG)
op_col varchar(50),
-- to which column we operate
from_clause varchar(1000)
-- the data origin
    );
# A SIMPLE EXAMPLE

```sql
set @XTAB_FORMAT = "show";
call get_crosstab_simple (  
"department", -- row_name  
"gender", -- col_name  
"COUNT", -- op  
"person_id", -- op_col  
"person inner join departments using (dept_id)" -- from_clause
);
```
Crosstab package (4)

# A SIMPLE EXAMPLE (2)

<table>
<thead>
<tr>
<th>department</th>
<th>m</th>
<th>f</th>
<th>total</th>
</tr>
</thead>
<tbody>
<tr>
<td>dev</td>
<td>1</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>pers</td>
<td>3</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>sales</td>
<td>2</td>
<td>1</td>
<td>3</td>
</tr>
</tbody>
</table>
# Another EXAMPLE

set @XTAB_FORMAT = "show";
set @WITH_ROLLUP = 1;
call get_crosstab_simple (   "gender",            -- row_name   "department",       -- col_name   "COUNT",           -- op   "person_id",        -- op_col   "person inner join departments using (dept_id)"           -- from_clause );
# Another EXAMPLE (2)

<table>
<thead>
<tr>
<th>gender</th>
<th>pers</th>
<th>sales</th>
<th>dev</th>
<th>total</th>
</tr>
</thead>
<tbody>
<tr>
<td>f</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>m</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>6</td>
</tr>
<tr>
<td>NULL</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>9</td>
</tr>
</tbody>
</table>
# Some **input** modifiers

@XTAB_FORMAT={show|table|query}

@WITH_ROLLUP=1

@XTAB_ENGINE={MyISAM|memory}

@XTAB_TYPE=temporary
Crosstab package (8)

# Some output variables

@XTAB_QUERY
query to create the requested crosstab

@XTAB_TABLE
the table being created when requested
One step further

Object Oriented Crosstabs (1)

- Based on the Crosstab package
- Requires the MySQL Stored Routines Library (mysql-sr-lib) http://datacharmer.org
- It's a O-O like approach
Object Oriented crosstab (2)

HOW

• Download the code (all SQL)
  (http://datacharmer.org)
• Unpack and run

  mysql < oo_crosstab_install.mysql
• then use it

  mysql> CALL oo_crosstab_help()
SELECT oo_crosstab_usage()\G

== Object Oriented CROSSTAB USAGE ==

PROCEDURE xtab_new( crosstab_name varchar(50) );

PROCEDURE xtab_set( crosstab_name, p_param_name, p_param_value )

PROCEDURE xtab_check( crosstab_name )

PROCEDURE xtab_show( crosstab_name )

PROCEDURE xtab_exec( crosstab_name )

PROCEDURE xtab_drop( crosstab_name )
# An example

CALL xtab_new("dept_by_gender");

CALL xtab_set("dept_by_gender", "row_name", "department");
CALL xtab_set("dept_by_gender", "col_name", "gender");
CALL xtab_set("dept_by_gender", "op", "COUNT");
CALL xtab_set("dept_by_gender", "op_col", "person_id");

CALL xtab_check("dept_by_gender");

+-----------------------------------------------+-----------------+------------------+
| check result | error message |
|-----------------------------------------------+-----------------+------------------|
| 0 | dept_by_gender: parameter <from_clause> not set |
+-----------------------------------------------+-----------------+------------------+
# An example (continue)

CALL xtab_show("dept_by_gender");

<table>
<thead>
<tr>
<th>xtab_parameter</th>
<th>xtab_value</th>
<th>check</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>dept_by_gender</em></td>
<td>ok</td>
<td></td>
</tr>
<tr>
<td>row_name</td>
<td>department</td>
<td>ok</td>
</tr>
<tr>
<td>col_name</td>
<td>gender</td>
<td>ok</td>
</tr>
<tr>
<td>id_name</td>
<td>NULL</td>
<td>-</td>
</tr>
<tr>
<td>col_from</td>
<td>NULL</td>
<td>-</td>
</tr>
<tr>
<td>op</td>
<td>COUNT</td>
<td>ok</td>
</tr>
<tr>
<td>op_col</td>
<td>person_id</td>
<td>ok</td>
</tr>
<tr>
<td>from_clause</td>
<td>NULL</td>
<td>NOT OK</td>
</tr>
<tr>
<td>where_clause</td>
<td>NULL</td>
<td>-</td>
</tr>
<tr>
<td>wanted_result</td>
<td>NULL</td>
<td>-</td>
</tr>
</tbody>
</table>
# An example (continue)

```sql
CALL xtab_set("dept_by_gender", "from_clause", "person INNER JOIN departments using(dept_id)");

CALL xtab_check("dept_by_gender");
```

<table>
<thead>
<tr>
<th>check result</th>
<th>error message</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>NULL</td>
</tr>
</tbody>
</table>

---
# An example (continue)

```plaintext
CALL xtab_exec("dept_by_gender");

+----------+-------+-----+-------+
| department| m     | f   | total |
+----------+-------+-----+-------+
| dev      | 1     | 1   | 2     |
| pers     | 3     | 1   | 4     |
| sales    | 2     | 1   | 3     |
+----------+-------+-----+-------+
```

60
CALL xtab_copy_from('gender_by_dept', 'dept_by_gender', 'transpose');

CALL xtab_exec('gender_by_dept');

<table>
<thead>
<tr>
<th>gender</th>
<th>pers</th>
<th>sales</th>
<th>dev</th>
<th>total</th>
</tr>
</thead>
<tbody>
<tr>
<td>f</td>
<td>5500</td>
<td>5500</td>
<td>6000</td>
<td>17000</td>
</tr>
<tr>
<td>m</td>
<td>16000</td>
<td>10500</td>
<td>6000</td>
<td>32500</td>
</tr>
</tbody>
</table>
# Exporting crosstabs

```sql
SELECT xtab_export('gender_by_dept', 'compact')
```

******************** 1. row ****************

```sql
CALL xtab_new('gender_by_dept');
CALL xtab_fill('gender_by_dept',
'row_name' => 'gender;
'col_name' => 'department;
id_name' => '\N;
col_from' => '\N;
op' => 'sum;
op_col' => 'salary;
from_clause' => 'person inner join departments
using(dept_id);
where_clause' => '\N;
wanted_result' => '\N')
```
Object Oriented Crosstab (10)

# Exporting crosstabs

SELECT xtab_export('gender_by_dept', 'detailed')

****************** 1. row ********************
xtab_export('gender_by_dept', 'detailed'):
CALL xtab_new('gender_by_dept');
CALL xtab_set('gender_by_dept', 'row_name', 'gender');
CALL xtab_set('gender_by_dept', 'col_name', 'department');
CALL xtab_set('gender_by_dept', 'id_name', NULL);
CALL xtab_set('gender_by_dept', 'col_from', NULL);
CALL xtab_set('gender_by_dept', 'op', 'sum');
CALL xtab_set('gender_by_dept', 'op_col', 'salary');
CALL xtab_set('gender_by_dept', 'from_clause', 'person inner join departments using (dept_id)');
CALL xtab_set('gender_by_dept', 'where_clause', NULL);
CALL xtab_set('gender_by_dept', 'wanted_result', NULL);

G
CALL xtab_query('gender_by_dept')\G

query for crosstab:
SELECT gender,
    sum( CASE WHEN department = 'pers' THEN salary ELSE NULL END) AS `pers`,
    sum( CASE WHEN department = 'sales' THEN salary ELSE NULL END) AS `sales`,
    sum( CASE WHEN department = 'dev' THEN salary ELSE NULL END) AS `dev`,
    sum(salary) as total
FROM person inner join departments using(dept_id)
GROUP BY gender
Object Oriented Crosstab (12)

# Listing crosstabs

CALL xtab_list();

+---------------------------------+---------------------+
| CROSSTAB                        | executable          |
+---------------------------------+---------------------+
| category_by_rating              | ok                  |
| rating_by_category              | ok                  |
| location_by_dept                | ok                  |
| dept_by_location                | ok                  |
| gender_by_location              | ok                  |
| location_by_gender              | ok                  |
| dept_by_gender                  | ok                  |
| gender_by_dept                  | ok                  |
+---------------------------------+---------------------+
Parting thoughts

- Pivot tables can be made in SQL
- Pivot tables in normal SQL require external languages
- With stored routines, there are various techniques
- Efficient crosstab are feasible with well designed stored routines
THANKS

Question

time

http://datacharmer.org