

SQL Review

Thomas Schwarz, SJ

Selects from a single table

- Uses a From clause and a Where Clause
- Gives desired attribute names that can be renamed or subjected to arithmetic calculations

Example

- We will have to pay Value Added Taxes
 - Let's assume we leave it to the clients unless we have a presence in the country.

-

```
CREATE TABLE VAT
( country VARCHAR(30) KEY,
  vatrate FLOAT)
```

```
INSERT INTO VAT(country, vatrate)
VALUES
('France', 0.20),
('Japan', 0.08),
('UK', 0.20),
('Australia', 0.10);
```

Example

- We check the VAT-rate of our customers
 - Careful: we cannot simply join the VAT table with customers
 - We need an SQL IF condition
 - If (boolean, value if true, value if false)
 - We check for the presence of the customer's country in the VAT table

Example

```
SELECT
    customers.customerName, customers.country,
    IF (customers.country IN
        (SELECT DISTINCT VAT.country FROM VAT),
        VAT.vatrate,
        0) AS tax
FROM
    customers, VAT;
```

Example

- Easiest to embed the vat calculation into a function

```
CREATE FUNCTION `get_vatrate` (mycountry VARCHAR(30))
  RETURNS float
  READS SQL DATA
BEGIN
  RETURN
  IF (mycountry IN (SELECT VAT.country FROM VAT),
    (SELECT VAT.vatrate FROM VAT
      WHERE VAT.country = mycountry),
    0) ;
END
```

Example

- Maybe less opaque

```
CREATE FUNCTION "get_vat" (my_country VARCHAR(30))
    RETURNS float
    READS SQL DATA
BEGIN
    DECLARE myvatrate FLOAT;
    SELECT
        vatrate
    INTO myvatrate
    FROM VAT
    WHERE VAT.country = my_country;
    RETURN IF(myvatrate IS NULL, 0, myvatrate);
END
```

Example

- We can get now order summaries for a given time

```
SELECT
    customerName, country, orderNumber, orderDate,
    ROUND(SUM(quantityOrdered * priceEach) *
        (1+get_vatrate(country)),2) AS "Inclusive TAX",
    ROUND(SUM(quantityOrdered * priceEach),2 )
        AS "Exclusive Tax"
FROM
    orders JOIN orderdetails USING(orderNumber)
        JOIN customers USING(customerNumber)
WHERE orderDate BETWEEN '2003-01-01' AND '2003-10-01'
GROUP BY orderNumber
ORDER BY customerName;
```


Dealing with NULL

- Dealing with Null values is important
 - First way: Use the IFNULL function
 - IFNULL(X, Y) returns X, if X is not NULL
 - Otherwise returns Y

Example

- Dealing with VAT again
 - A join between customers and VAT on country needs to be an outer join
 - `customers LEFT JOIN VAT using(country)`
 - or
 - `VAT RIGHT JOIN customers using(country)`

Example

- The result of the outer join has NULL values for VAT rate

```
SELECT
    customers.customerName, customers.country,
    IFNULL(VAT.vatrate, 0) AS tax
FROM
    customers LEFT JOIN VAT using(country);
```

customerName	country	tax
Atelier graphique	France	0.2
Signal Gift Stores	USA	0
Australian Collectors, Co.	Australia	0.1
La Rochelle Gifts	France	0.2
Baane Mini Imports	Norway	0
Mini Gifts Distributors Ltd.	USA	0
Havel & Zbyszek Co	Poland	0
Blauer See Auto, Co.	Germany	0
Mini Wheels Co.	USA	0
Land of Toys Inc.	USA	0
Euro+ Shopping Channel	Spain	0
Volvo Model Replicas, Co	Sweden	0
Danish Wholesale Imports	Denmark	0

Dealing With NULLS

- Another way is to use the COALESCE function
 - Returns the first Non-null argument

Example

- Same result as before

```
SELECT
    customers.customerName, customers.country,
    COALESCE(VAT.vatrate, 0) AS tax
FROM
    customers LEFT JOIN VAT using(country);
```

Example

- A better way for the previous question is now

```
SELECT
    customerName, country, orderNumber, orderDate,
    ROUND(SUM(quantityOrdered * priceEach) *
        (1+ COALESCE(VAT.vatrate,0)),2) AS "Inclusive TAX",
    ROUND(SUM(quantityOrdered * priceEach),2 )
        AS "Exclusive Tax"
FROM
    orders JOIN orderdetails USING(orderNumber)
        JOIN customers USING(customerNumber)
        LEFT JOIN VAT USING(country)
WHERE orderDate BETWEEN '2003-01-01' AND '2003-10-01'
GROUP BY orderNumber
ORDER BY customerName;
```

customerName	country	orderNumber	orderDate	Inclusive TAX	Exclusive Tax
Alpha Cognac	France	10136	2003-07-04	17079.24	14232.70
Anna's Decorations, Ltd	Australia	10148	2003-09-11	45710.2	41554.73
Atelier graphique	France	10123	2003-05-20	17485.73	14571.44
Australian Collectors, Co.	Australia	10120	2003-04-29	50450.43	45864.03
Australian Collectors, Co.	Australia	10125	2003-05-21	8321.59	7565.08
Australian Gift Network, Co	Australia	10152	2003-09-25	10803.45	9821.32
AV Stores, Co.	UK	10110	2003-03-18	58110.83	48425.69
La Corne D'abondance, Co.	France	10114	2003-04-01	40059.77	33383.14
Lyon Souvenirs	France	10134	2003-07-01	28103.36	23419.47
Marseille Mini Autos	France	10122	2003-05-08	60989.59	50824.66
Reims Collectables	France	10121	2003-05-07	20040.56	16700.47

Select from Multiple Tables

- “Classic” SQL makes implicit joins
- “New” SQL has explicit joins
 - In general makes for more understandable statements
 - MySQL only has left and right joins, not outer joins

Inner vs. Outer Joins

- A tuple in an inner join on a set of attribute:
 - In both tables, these attributes have the same value
- Outer joins allow for missing values, in which case they become Nulls

Example

- Right Join: All tuples in the left table are represented

```
SELECT
    customerName, city, country, vatrate
FROM
    VAT RIGHT JOIN customers using(country);
```

	customerName	city	country	vatrate	
▶	Atelier graphique	Nantes	France	0.2	
	Signal Gift Stores	Las Vegas	USA	NULL	
	Australian Collectors, Co.	Melbourne	Australia	0.1	
	La Rochelle Gifts	Nantes	France	0.2	
	Baane Mini Imports	Stavern	Norway	NULL	
	Mini Gifts Distributors Ltd.	San Rafael	USA	NULL	
	Havel & Zbyszek Co	Warszawa	Poland	NULL	
	Blauer See Auto, Co.	Frankfurt	Germany	NULL	
	Mini Wheels Co.	San Francisco	USA	NULL	
	Land of Toys Inc	NYC	USA	NULL	

Example

- If we switch, we loose tuples

```
SELECT
    customerName, city, country, vatrate
FROM
    customers RIGHT JOIN VAT USING (country);
```

	customerName	city	country	vatrate
▶	Australian Collectables, Ltd	Glen Waverly	Australia	0.1
	Australian Gift Network, Co	South Brisbane	Australia	0.1
	Souvenirs And Things Co.	Chatswood	Australia	0.1
	Anna's Decorations, Ltd	North Sydney	Australia	0.1
	Australian Collectors, Co.	Melbourne	Australia	0.1
	Auto Canal+ Petit	Paris	France	0.2
	Reims Collectables	Reims	France	0.2
	Marseille Mini Autos	Marseille	France	0.2
	Auto Associés & Cie.	Versailles	France	0.2
	Lyon Souvenirs	Paris	France	0.2
	Alpha Cognac	Toulouse	France	0.2
	Mini Caravy	Strasbourg	France	0.2
	La Corne D'abondance, Co.	Paris	France	0.2
	Daedalus Designs Imports	Lille	France	0.2
	Saveley & Henriot, Co.	Lyon	France	0.2
	Le Daubelle Gifts	Nantes	France	0.2

Aggregate Queries

- Without aggregate function
 - GROUP BY has the effect of distinct
 - GROUP BY orders

```
SELECT status  
FROM orders  
GROUP BY status;
```

Result Grid	Filter R
status	
▶ Shipped	
Resolved	
Cancelled	
On Hold	
Disputed	
In Process	

Aggregate Queries

- With aggregate function







```
SELECT
    status, SUM(priceEach * quantityOrdered) AS VOLUME
FROM
    orders
    JOIN
    orderdetails USING (orderid)
GROUP BY status WITH ROLLUP;
```

Result Grid		Filter Rows:
status	VOLUME	
▶ Cancelled	238854.18	
Disputed	61158.78	
In Process	135271.52	
On Hold	169575.61	
Resolved	134235.88	
Shipped	8865094.64	
HULL	9604190.61	

Aggregate Queries

- With aggregate function






```
SELECT
    status, COUNT(*) AS Incidences
FROM
    orders
    JOIN
    orderdetails USING (orderid)
GROUP BY status WITH ROLLUP;
```

Result Grid			 Filter Row
	status	Incidences	
	Cancelled	79	
	Disputed	14	
	In Process	41	
	On Hold	44	
	Resolved	47	
	Shipped	2771	
	NULL	2996	

Aggregate Queries





- We use a WHERE clause to sub-select before grouping

```
SELECT
    status, COUNT(*) AS Incidences
FROM
    orders
    JOIN
    orderdetails USING (orderid)
WHERE YEAR(orderDate) = 2004
GROUP BY status WITH ROLLUP;
```

Result Grid			 Filter Row
	status	Incidences	
	Cancelled	54	
	On Hold	6	
	Resolved	8	
	Shipped	1353	
	NULL	1421	

Aggregate Queries

```
SELECT
    YEAR(orderDate) as YEAR,
    SUM(priceEach * quantityOrdered) AS "Cancelled Volume"
FROM
    orders
    JOIN
    orderdetails USING (orderid)
WHERE status = 'cancelled'
GROUP BY YEAR(orderDate) WITH ROLLUP;
```

Result Grid			 Filter Rows
	YEAR	Cancelled Volume	
	2003	67130.69	
	2004	171723.49	
	HULL	238854.18	

Aggregate Queries

- A MySQL specialty not in SQL:
 - You can have an alias in Group By:

```
SELECT
    YEAR(orderDate) as year, SUM(priceEach *
quantityOrdered) AS "Cancelled Volume"
FROM
    orders
    JOIN
    orderdetails USING (ordernumber)
WHERE status = 'cancelled'
GROUP BY year WITH ROLLUP;
```

Alias

Result Grid		Filter Row
year	Cancelled Volume	
▶ 2003	67130.69	
2004	171723.49	
NULL	238854.18	

Aggregate Queries

- WHERE
 - Filters Records
- HAVING
 - Filters Groups

```
SELECT
    select_list
FROM
    table_name
WHERE
    search_condition
GROUP BY
    group_by_expression
HAVING
    group_condition;
```



Aggregate Queries

- Example:
 - Overview of orders in January 2003

```
SELECT
  ordernumber,
  shippedDate,
  SUM(quantityOrdered) AS itemCount,
  SUM(priceeach*quantityOrdered) AS total
FROM
  orderdetails JOIN orders USING(ordernumber)
WHERE shippedDate between "2003-01-01" AND "2003-01-31"
GROUP BY ordernumber;
```

	ordernumb...	shippedDate	itemsCount	total
▶	10100	2003-01-10	151	10223.83
	10101	2003-01-11	142	10549.01
	10102	2003-01-14	80	5494.78

Aggregate Queries

- What about big orders only?

```
SELECT
  ordernumber,
  shippedDate,
  SUM(quantityOrdered) AS itemCount,
  SUM(priceeach*quantityOrdered) AS total
FROM
  orderdetails JOIN orders USING(ordernumber)
WHERE shippedDate between "2003-01-01" AND "2003-01-31"
GROUP BY ordernumber
HAVING total > 10000.00;
```

	ordernumb...	shippedDate	itemsCount	total
▶	10100	2003-01-10	151	10223.83
	10101	2003-01-11	142	10549.01

Common Table Expressions

- Creates a named, temporary table to simplify queries
- Defined with a WITH clause

```
WITH cte_name (column_list) AS (  
    query  
)
```

Example

```
WITH customers_in_asia AS (  
  SELECT *  
  FROM customers  
  WHERE country in  
    ('Japan', 'India', 'Singapore', 'Hong Kong',  
     'Philippines')  
)  
SELECT * FROM customers_in_Asia;
```

Example

- Finding the top sales people in 2004

```
WITH topsales AS (  
    SELECT employeeNumber, firstName, lastName,  
    SUM(quantityOrdered*priceEach) AS sales, officeCode  
    FROM orderdetails JOIN orders USING(orderNumber)  
    JOIN customers USING(customerNumber)  
    JOIN employees ON  
salesRepEmployeeNumber = employeeNumber  
    WHERE YEAR(shippedDate) = 2004  
    GROUP BY salesRepEmployeeNumber  
    ORDER BY sales DESC  
    LIMIT 5)  
SELECT * FROM topsales;
```

Example

- Which is the most successful office? Only top salespeople count:

Example

```
WITH topsales AS (  
    SELECT employeeNumber, firstName, lastName,  
    SUM(quantityOrdered*priceEach) AS sales, officeCode  
    FROM orderdetails JOIN orders USING(orderNumber)  
    JOIN customers USING(customerNumber)  
    JOIN employees ON  
    salesRepEmployeeNumber = employeeNumber  
    WHERE YEAR(shippedDate) = 2004  
    GROUP BY salesRepEmployeeNumber  
    ORDER BY sales DESC  
    LIMIT 5)  
SELECT SUM(sales) as volume, city, country  
FROM topsales JOIN offices USING(officeCode)  
GROUP BY officeCode  
ORDER BY volume DESC;
```