SQL COURSE

Sednove

Presented by
Xavier Bonifay

2015-08-07

TABLE OF CONTENT

SELECT STRUCTURE

STRUCTURED QUERY LANGUAGE

I want to select row data and calculated data
From different data sources
Where some conditions are met
Regrouping rows for calculation purposes
Ordering the output in a certain way.

```
SELECT columns, functions(columns)
FROM tables, views, (sub-queries)
WHERE conditions
GROUP BY columns
ORDER BY columns, functions(columns);
```

IF, IFNULL, STRING, DATE, NUMBER

Operators: = != > < <> LIKE IN BETWEEN IS NULL

IF (Expression, THEN, ELSE): IF(A=B,C,D)
IFNULL(column A, column B): if column A is null then replace the value with column B

String functions: https://mariadb.com/kb/en/mariadb/string-functions/
Cast, Concat, Instr, Length, Lower, Lpad, Replace, Rpad, Substr, Upper

Date functions: https://mariadb.com/kb/en/mariadb/date-and-time-functions/
Adddate, Date_format, Dayofweek, Last_day, Sysdate, Week, Weekday, Year

Number functions: https://mariadb.com/kb/en/mariadb/numeric-functions/
Mathematical functions, Ceil, Floor, Greatest, Least, Sign, Round, Truncate

ORDER BY, GROUP BY

ORDER BY column_name, function(column_name), 1 DESC/ASCE

GROUP BY: To use only if we have group by functions in the SELECT!

- Group By functions: SUM, MAX, MIN, COUNT, AVG, STD
- In the group by, ONLY put the columns which appear in the SELECT and not used within a group by function.

JOINS

Simple join: data must exists in the 2 tables:

```
FROM tableA
JOIN tableB on tableA.PK = tableB.FK
```

Outer join: data could not exists in the table on the left (or on the right)

```
FROM tableA
LEFT OUTER JOIN tableB on tableA.PK = tableB.FK
```

UNION, INTERSECT, MINUS

Combine two or more selects in one result. Used to ADD the results, get the common part or remove the results of the 2nd select from the first one.

```
SELECT colA as R1, colB as R2 From TableA
Where conditionA
UNION
SELECT colC as R1, colD as R2 From TableB
Where conditionB
ORDER BY 2, 1
```

Same number of columns, same types, give same aliases
ORDER BY at the end.
Note: UNION gives a unique result, UNION ALL gives all rows even if they are duplicated

SUB-QUERIES

A sub-query is a Select inside another select

1) **At the SELECT level:** Identical to call a function that returns one value for each row:
- SELECT (Select max(sale_date) from sales), employee_name from employee;

2) **At the FROM level:** Identical to call a View that returns multiple rows:
- SELECT employee_name, sales_date
  FROM employee, (select sales_date, emp_code from sales)
- WHERE employee.emp_code = sales.emp_code;

3) **At the WHERE level:** to validate a condition from another table:
- SELECT employee_name
  FROM employee
  WHERE emp_code in (select emp_code from sales);

VIEWS

Define a Select, store the select in the database and use it as a table
The result of the view is calculated when we use the View.

```
CREATE VIEW ViewName AS SELECT columns FROM tables
    WHERE conditions;
```

```
Select columns
FROM tables, ViewName
WHERE conditions
```

We can only use INSERT, DELETE, UPDATE on a view made on a single table.

INDEXES

Indexes are used to accelerate queries.

Indexes reduce all other transactions: INSERT, UPDATE, DELETE

Wrong indexes can slow down queries.

Indexes use a lot of disk space.

Only create indexes based on the needs of the queries.

```
CREATE INDEX IND1 ON TABLE1 (COL1, COL2, COL3);

1)SELECT * FROM TABLE1 WHERE COL1 = xxx AND COL2 = yyy
2)SELECT * FROM TABLE1 WHERE COL2 = xxx AND COL3 = yyy
3)SELECT * FROM TABLE1 WHERE substr(COL1,1,2) = xxx AND COL2
  = yyy
```

1) can use IND1 but 2) and 3) no

CONSTRAINTS

- NOT NULL
- PRIMARY KEY:

  CREATE TABLE Table_1 (column_1 SMALLINT, column_2 VARCHAR(5), CONSTRAINT constraint_1 PRIMARY KEY(column_1,column_2) NOT   DEFERRABLE);
- FOREIGN KEY: CREATE TABLE Table_2 (column_1 SMALLINT CONSTRAINT constraint_1 FOREIGN KEY REFERENCES Table_1 NOT DEFERRABLE, column_2 CHAR(5));
- CHECK: CREATE TABLE Table_1 (column_1 DATE CHECK (column_1 = CURRENT_DATE));

OPTIMIZING QUERIES

In the FROM:

- put the big tables first and the small tables after

In the WHERE:

- Follow the conditions based on your table list;
- Put the more restrictive condition at the end;
- Do not use OR: prefer UNION;
- Do not use IN: prefer EXISTS (NOT EXISTS);
- Try not to use subqueries;
- If you need subqueries, use subqueries returning one row and use =, not IN;
- Use DISTINCT instead of GROUP BY;
- Use only GROUP BY if you use GROUP functions;
- Do not return columns that you do not need (Subqueries or in the main select);
- Do not use functions on indexed columns;
- Be sure to have indexes on primary keys and foreign keys (or constraints);
- Index all significant columns used in your query and try to combine columns in one INDEX