Query Optimization

The design of the database is one of the most important factors in the performance of the database and with a good database design you also need optimal queries to perform optimally. Everyone wants the performance of their database to be optimal but does not concentrate on designing a query. They just write the query depending on the only major factor. What I want. They don’t consider that even the same thing can be achieved with some alternate queries and in more efficient manner.

Let’s take a very simple 3 table example, Employee, Department and Sub-Department. Schema of

Employee table

1. Employee

a. EmpId
b. SDId
c. EmpName
d. EmpAdd
e. EmpPhone

Sub-Department table

2. SubDepartment

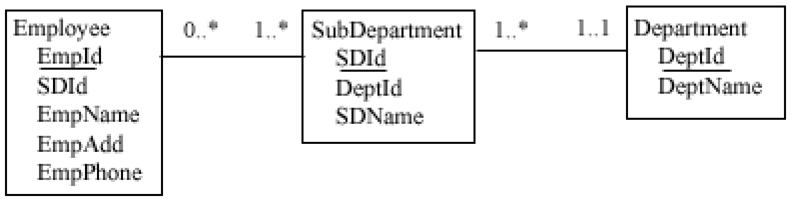
a. SDId
b. DeptId
c. SDName

Department table

3. Department

a. DeptId
b. DeptName

The relationship between the Employee and Sub-Department table is one or more sub-departments have zero or more employees or in other words zero or more employees work in one or more Sub-Departments. The relationship between Department and Sub-Department is one or more Sub-Department is within one and only one Department. The Entity Relationship diagram of the above scenario is



ER Diagram

ER Diagram

Now with the above Database schema lets have a very basic queries which can give us the same results and we will analyze as to which is faster and better performing than the other one. Let's take a very common example with the above database schema, to know all the Employees in a particular department. The common solution that comes to our mind is.

Code: sql

```sql
SELECT e.EmpId, e.SDId, e.EmpName, e.EmpAdd, e.EmpPhone
FROM Employee e, SubDepartment sd, Department d
WHERE e.SDId = sd.SDId
AND sd.DeptId = d.DeptId
AND d.DeptId = 'MyDepartment'
```

The above solution of joining the 3 tables to give us the output for the MyDepartment as the id of the department is a very common but is a very expensive one. Any organization have an employee to department ratio quite high and joining an Employee to a Department table can be quite a undesirable operation.

If we analyze the above solution then we are getting the output for one and only one Department i.e. MyDepartment. So our aim should be to find the Sub-Department's in our MyDepartment.

Code: sql

```sql
SELECT SDId
FROM SubDepartment
WHERE DeptId = 'MyDepartment'
```

The above query gives us the Sub-Department ID's of all the Sub department present in our concerned department 'MyDepartment'. Now if we can have the all the employee details present in each subdepartment returned by the above query we get the results that we had with the join from 3 tables.

Code: sql

```
SELECT e.EmpId, e.SDId, e.EmpName, e.EmpAdd, e.EmpPhone
FROM Employee e
WHERE e.SDId
IN (SELECT SDId
FROM SubDepartment
WHERE DeptId = ‘MyDepartment’)
```

With the above solution we have avoided 3 table join and also a query to the department table with introduction of a sub query. By avoiding the joins of a tables what we have achieved is no Department table in the query to retrieve the results but managed to provide the same output. Now we have some limitations with the above solutions.

If you are a web developer and if your client is using MySQL and that also there are chances that he might not be using version 5 then you are in trouble as you cannot use Sub queries. Then you can even write the above query using any programming language and avoid even the sub-query. Let’s take the simple example of PHP and MySQL.

Code: PHP

```
//SQL to give the sub-departments of the queries DeptId
$sql_sub_dept = "SELECT SDId FROM SubDepartment WHERE DeptId = $DeptID";
//Main SQL to give us the Employee of a particular department here $DeptID
$sql_main_dept = " SELECT e.EmpId, e.SDId, e.EmpName, e.EmpAdd, e.EmpPhone
FROM Employee e WHERE Employee. SDId IN (";

//Execute the SQL and concatenate each subdepartment ID
    $result = mysql_query ($sql_sub_dept);
    while($rec = mysql_fetch_array($result))
    {
        $ sql_main_dept.= $rec["SDId"] ;
        $ sql_main_dept.= ',';
    }
    //Remove the last “,” and end the bracket to complete the SQL.
    $sql_main_dept = trim($sql_main_dept,",");
    $sql_main_dept.= ")";
```

Now the variable $sql_main_dept has the required query for us to execute, retrieve the results and display them. The limitation is we can never know the name of the department as we don't have the department table involve. If we also need the Department table its better to go for a third query to retrieve the results by querying the department name and storing them in a variable to show against each row.

With the above example it looks like the situation is very uncommon but with experience I have found that the above situations occur in numerous real time databases and occasions and some of the example can be very easily taken. One of the examples is what I have chosen, the employee with Department / Subdepartment and other can be Category / Sub-Categor. The category / Sub-Category example applies to majority of users if not all so before going about writing the query just keep in mind "Can this can be done in any alternate and better way".

Source: http://www.go4expert.com/articles/query-optimization-t565/