**Part 03 - Optimization**

**Experiment 1: To calculate how many employees currently works in each department**

This experiment intend to show how the count query was performed faster after the use of a secondary index to get a derived attribute.

Based on Connolly, T. (pg 499) “Attributes whose value can be found by examining the values of other attributes are known as derived or calculated attributes”.

In this case, we will use a count statement to check how many employees works for each department. But it gets more complex as on the table we want to run the search (dept\_emp) there are only the department numbers and not the department name. So we have to access columns from two differents tables, department and dept\_emp, to be able to compare (with where clause) and complete the query successfully.

Other point is that analysing the table dept\_emp, the column emp\_no is already a index as primary key, so there is no point make it as an index for optimize the system. We need to find a index candidate attribute, based on what we want search. As we want access that relation based on the employee that are currently working for the department.

According Connolly, T (pg 509) “secondary indexes provide a mechanism for specifying an additional key for a base relation that can be used to retrieve data more efficiently”.

That is where the secondary index comes into play. By applying a secondary index to the dep\_emp table, we will be able to reduce the number of records the query will search and consequently, reduce the time to process the query.

**SQL query**

**SELECT SQL\_NO\_CACHE** dept\_name

, count(emp\_no) AS numberOfEmployees

**FROM** dept\_emp

**LEFT JOIN** departments **ON** departments.dept\_no = dept\_emp.dept\_no

**WHERE** dept\_emp.to\_date = "9999-01-01"

**GROUP BY** dept\_name

**ORDER BY** numberOfEmployees;

**Transaction Analysis**

Transaction: List the Department name and number of employee current working in each department, ordered by number of employee.

Transaction volume: Average - 10 per hour

Peak - 20 per hour (between 15:00 and 17:00 on Thursday)

SELECT SQL\_NO\_CACHE dept\_name, count(emp\_no) AS numberOfEmployees

FROM dept\_emp

LEFT JOIN departments ON departments.dept\_no = dept\_emp.dept\_no

WHERE dept\_emp.to\_date = "9999-01-01"

GROUP BY dept\_name

ORDER BY numberOfEmployees;

Predicates: dept\_emp.to\_date = "9999-01-01"

Join attributes: departments.dept\_no = dept\_emp.dept\_no

Grouping attribute: dept\_name

Ordering attribute: numberOfEmployees



|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Access** | **Entity** | **Type of access** | **N. of References** | | |
| **Per transaction** | **Avg Per Hour** | **Peak Per Hour** |
| 1 | Dep\_emp | R | 36,844 | 368,440 | 736,880 |
| 2 | Department | R | 9 | 90 | 180 |
| Total References | | | 36,853 | 368,550 | 737,060 |

**Full SQL Audit Trail**

|  |  |
| --- | --- |
| SELECT SQL\_NO\_CACHE dept\_name, count(emp\_no) AS numberOfEmployees | **Select the columns we want on the result** |
| FROM dept\_emp | **Stating the table from where the data will be retrieved** |
| LEFT JOIN departments | **Stating the second table for join** |
| ON departments.dept\_no = dept\_emp.dept\_no | **Giving the comparable column to join** |
| WHERE dept\_emp.to\_date = "9999-01-01" | **Condition to a record be considered** |
| GROUP BY dept\_name | **Group the result based on** |
| ORDER BY numberOfEmployees; | **Order the result based on** |

**Explain Analisys**

EXPLAIN

SELECT SQL\_NO\_CACHE dept\_name, count(emp\_no) AS numberOfEmployees

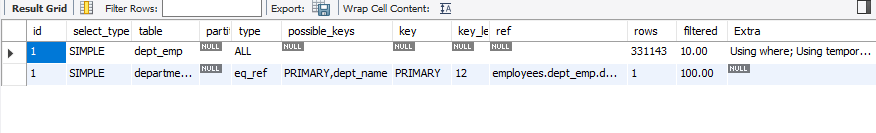
from dept\_emp

left join departments on departments.dept\_no = dept\_emp.dept\_no

where dept\_emp.to\_date = "9999-01-01"

group by dept\_name

order by numberOfEmployees;



CREATE index idx\_dept\_to\_date on dept\_emp(to\_date desc);

EXPLAIN

SELECT SQL\_NO\_CACHE dept\_name, count(emp\_no) AS numberOfEmployees

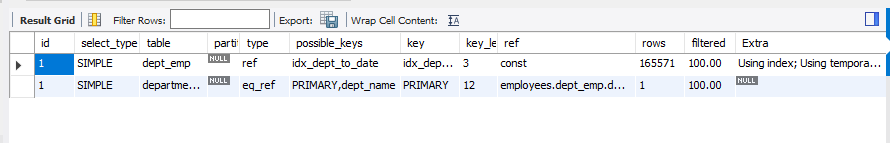
from dept\_emp

left join departments on departments.dept\_no = dept\_emp.dept\_no

WHEREdept\_emp.to\_date = "9999-01-01"

GROUP BY dept\_name

ORDER BY numberOfEmployees;



**Query analysis**

Machine Specification: Processor Intel(R) Core(TM) i5-6200U CPU @ 2.30GHz, 2400 Mhz, 2 Core(s), 4 Logical Processor(s)

Based on the output of Explain, when this query is called, the system should get the 331143 records from tables dept\_emp.

Also the EXPLAIN command says the type of emp\_no column is **const**, that means that there is only one matching row and the values from this column can be accessed very fast because they are read only once.

For that reason, we decided to create an index on dep\_emp table, because that is the table which we will get more data from.

The output of show create table command, reveals that emp\_no and dept\_no are primary keys and contains the data from employee and department table respectivement, as foreign keys. We decided to use to\_date column as a secondary index, after analyzing that some records is dated 9999-01-01, that means that the employee current works for the company.

The output gives us extra information, that a temporary file was created to organize the data collect from the WHERE clause.

After the creation of the index idx\_dept\_to\_date, we could notice that the system accessed only 165571 records to figure out the result for our query and also reduced the time lapse for complete the query.

**Time taken for query executions**

State of DB before change

EXPLAIN output

Time Taken for Query Execution: 1.734sec

State of DB after change

EXPLAIN output

Time Taken for Query Execution: 0.421sec

**Experiment 02: How many employees work in more than one department**

**SELECT CONCAT(**first\_name, ' ', last\_name**) AS** Employee, **COUNT(**\***) AS** 'N. of Departments'

**FROM** dept\_emp

**INNER JOIN** employees

**ON** employees.emp\_no = dept\_emp.emp\_no

**GROUP BY** dept\_emp.emp\_no

**HAVING COUNT(**\***) <>** 1

**LIMIT 7;**

****

**Experiment 03: Average Salary of Female and Male employees**

**SELECT** gender, **AVG(**salary**) AS** 'Average Salary'

**FROM** salaries

**INNER JOIN** employees

**ON** salaries.emp\_no = employees.emp\_no

**WHERE** gender = 'F'**;**



**SELECT** gender, **AVG(**salary**) AS** 'Average Salary'

**FROM** salaries

**INNER JOIN** employees

**ON** salaries.emp\_no = employees.emp\_no

**WHERE** gender = 'M'**;**

