Timesheet Database Management System using Indexes and Joins

Lalitha Madhuri Putchala, Anusha Chanumolu, SriVidhya Gundarapu
School of Computer Science & Mathematics,
University of Central Missouri

***Abstract*-- This project develops the Database Management System for the Time-sheet entries of employees at University of Central Missouri (UCM). Nearly 5000 people are working in this university both full-time and part-time as well as both technical and non-technical. It requires an efficient database system to log their work entries and time-sheet. The UCM Office has a desktop application which is used by the employees to log their working hours in a time-sheet. It is built in JAVA using Eclipse IDE. The application contains numerous modules based on categories of work and employee types. There can be a normal employee, an employer who bills the employee and an administrator who deals with managing both. Everyone has their account which contain their information. They have their respective levels of access to make changes in their account. For an example, an employer can add or remove an employee from the application whereas employee can only edit his own details. Administrative assistant assigns a work to an employee suggested by the employer.**

We have developed such database application using JAVA and the database connectivity to a database like SQL using Eclipse IDE. Predominantly, the concepts of joins and indexes are used to build the database application. The software design based on the requirements is tested with the relevant test-cases which will be designed by our team. Upon the satisfactory results, we achieved a robust database application using indexes and joins.

Keywords- Database Management System, Timesheet entries, SQL, Indexes, Joins.

I. INTRODUCTION

The timesheet entry is an important task to trace the activities of employees in an organization. In this project report, we have started with motivation of the project in the chapter 2. In chapter 3, we discussed in detail about the design trade-offs, tools used, modules developed and the requirements of this project. In the chapter 4, we have provided the in-depth explanation of the database implementation that includes primary hash, unique constraints, joins and indexes along with the screenshots.

II. Motivation

This kind of application is necessary for any kind of organization to track the employees and their performance. Higher end commercial time-sheet applications like PPM and salesforce are followed by corporate companies. Having inspired from them, we are developing this application, even though, not at higher scale when compared to tools like salesforce and PPM. This application would suit medium scale organizations like UCM where employees in hundreds of numbers are involved.

III. Design and Architecture

We have followed a systematic procedure while designing the architecture of the system. Initially, we have developed the modules, later the implemented the database queries accordingly and the trade-offs have been tested efficiently.

A. Modules:

There are three types of users that exist in the database application. Employee, Employer and the Administrative assistant. Every user has their own duties defined as below.

Employee: In this module, we have developed the duties of employee. He will have the below actions to access:

- Clock_in
- Clock_out
- Set-Pin
- View Task

Employer: This module sets the duties of the employer. He/she will have the below actions to access:

- Removing an employee
- Adding an employee

- Displaying the prompt messages
- Tracing the employee with respect to their job positions.

Administrative Assistant: This module, we have concentrated to set the duties of the administrative assistant. He will have the below actions to access:

- Checking the applications used by employees
- Validating the software
- Upgrading the software
- Responding to employee issues

B. Tools:

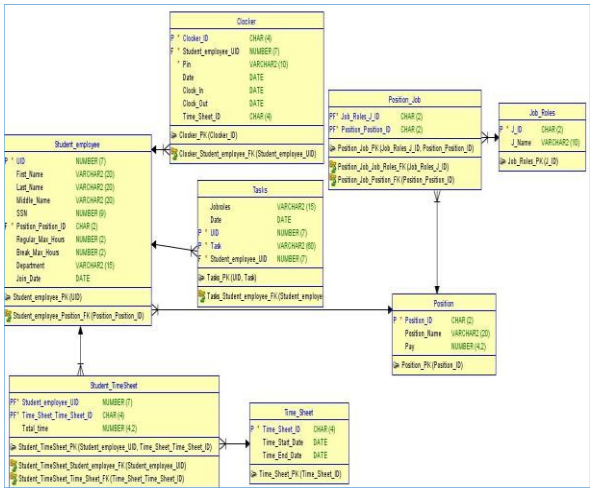
Software requirements:

- Technology: JAVA
- Database: MySQL
- Developing Tool: Eclipse JAVA IDE (NEON)
- JDBC Tool Kit
- JRE 7

Hardware requirements:

- Hard Disk Space: 1 TB
- Processor: Intel Core I7

C. Relational Database Modelling with ERD:

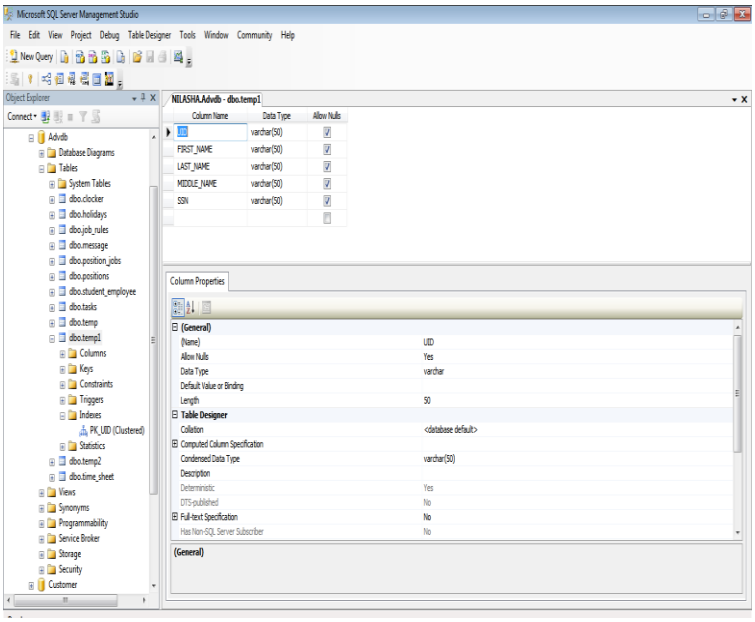


IV. Database Implementation

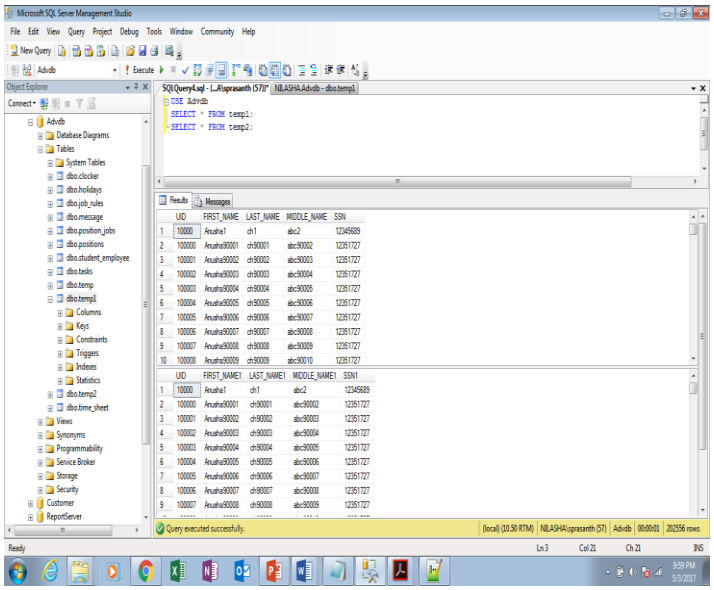
A. Creating Table for student_employee:

- CREATE TABLE `student_employee` (
- `uid` int(11) NOT NULL DEFAULT '0',
- `First_Name` varchar(20) DEFAULT NULL,
- `Last_Name` varchar(20) DEFAULT NULL,
- `Middle_Name` varchar(20) DEFAULT NULL,
- `SSN` int(9) NOT NULL DEFAULT '0',
- `Position_id` varchar(2) DEFAULT NULL,
- `Regular_Max_Hours` int(11) DEFAULT NULL,
- `Break_Max_Hours` int(11) DEFAULT NULL,
- `Department` varchar(15) DEFAULT NULL,
- `join_Date` date DEFAULT NULL,
- PRIMARY KEY (`uid`,`SSN`))

Created table and checking the Constraints in the below slide. We are taking two tables to analyze all the scenarios of creating indexes and joins.

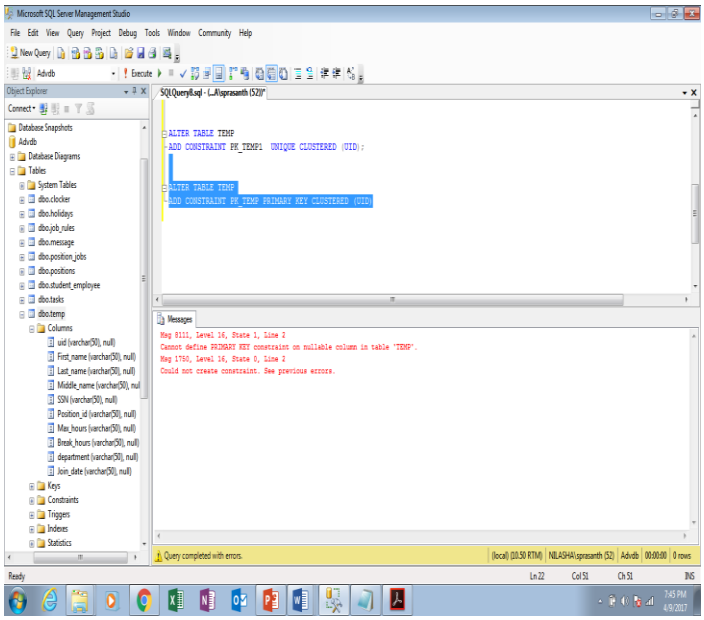


Screenshot of the table



B. Primary Key:

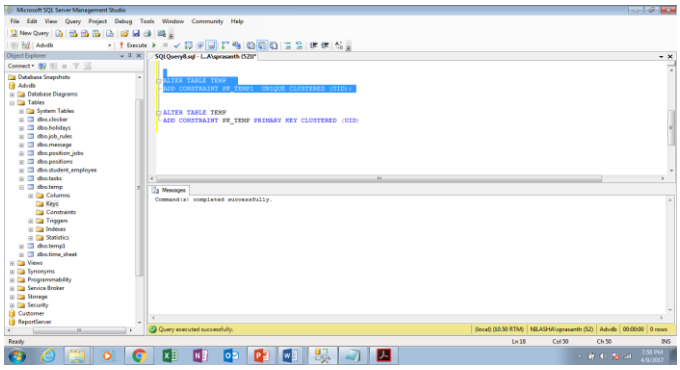
Primary key uniquely identifies each record and must contain unique values and cannot accept NULL Values. It is not possible to create primary key index on column which accepts null values. As seen in below figure, we could not able to give a primary key (Clustered) index to the field UID. UID is the field which accepts null Values. As it is accepting nullable values we cannot create Primary Key on the UID Field. To create a Primary index, we need to change that field to "Not Null", then we can create a Primary Key on that field.



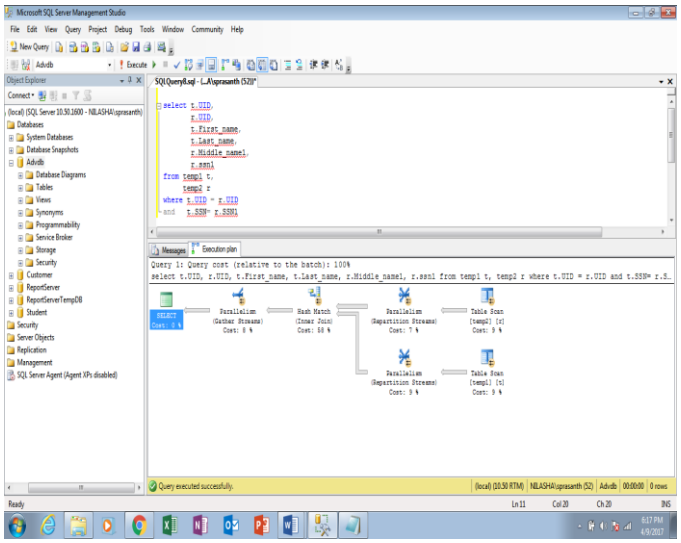
Primary key screenshot not accepting NULL values

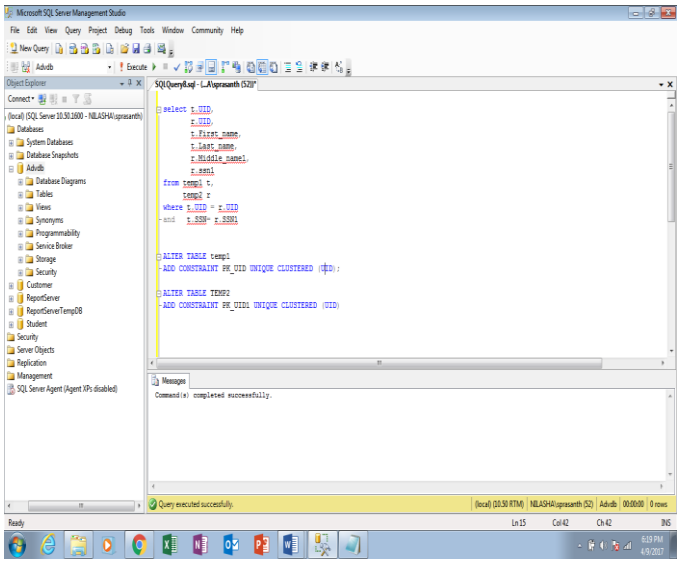
C.Unique Constraint:

It acts same like Primary key but it will accept null Values. Refer to Figure 3 below, where we have created Unique Constraint on the same column as it is accepting null values we can create Unique Constraint. It acts like Primary Key.



D. Hash Join:



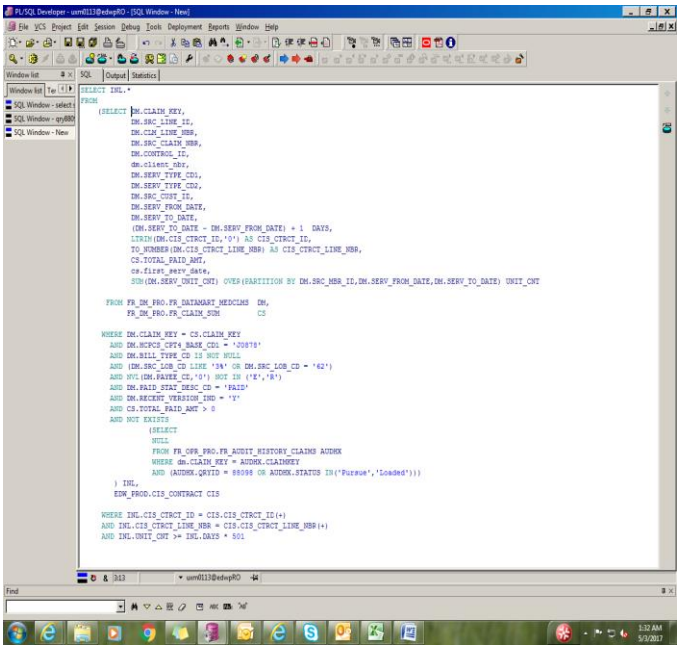


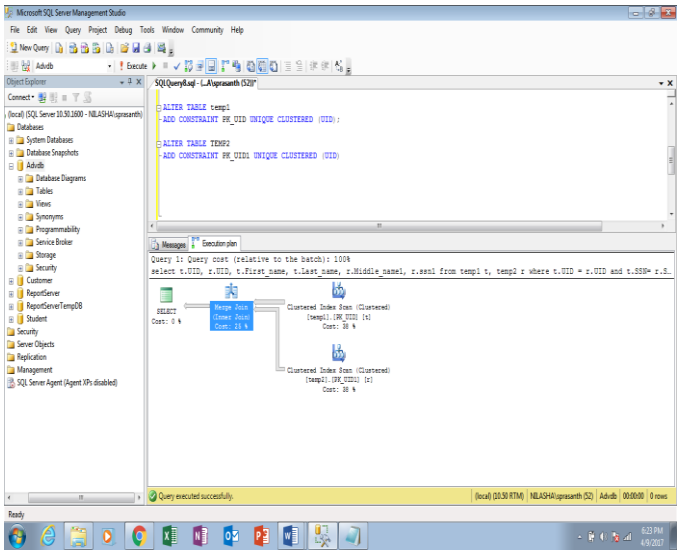
Clustered Index:

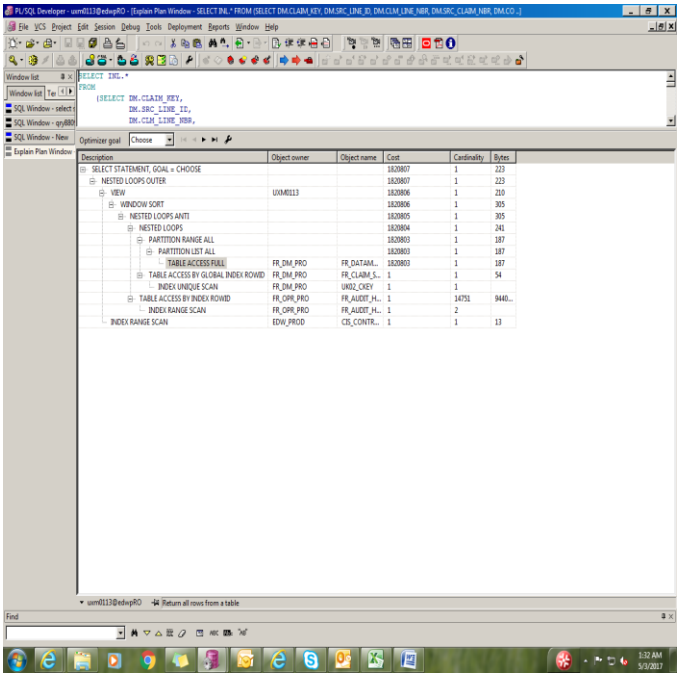
- A clustered index is a special type of index that reorders the way records in the table are physically stored. Therefore, table can have only one clustered index. By default, SQL Server takes primary key as clustered Index.
- So we have created clustered index on the two tables that we have shown in the previous slide. In the below slide we can see the cost consumption on that table.

Merge Join:

- Merge join is possible for the tables have an index on the join column. The index either clustered or covering non-clustered index.
- After applying clustered index on the above two tables it will take merge join instead of Hash join.
- Merge Join does not behave as hash join (it will take full table scan) whereas merge won't read the entire data. Clustered Index, sorts and stores the data in the table based on the key value. If it is not sorted in order, then we will call it as heap. Clustered table contains data in sorted order and retrieves data faster when we are retrieving small amount of data from huge tables.







HASH JOIN: -

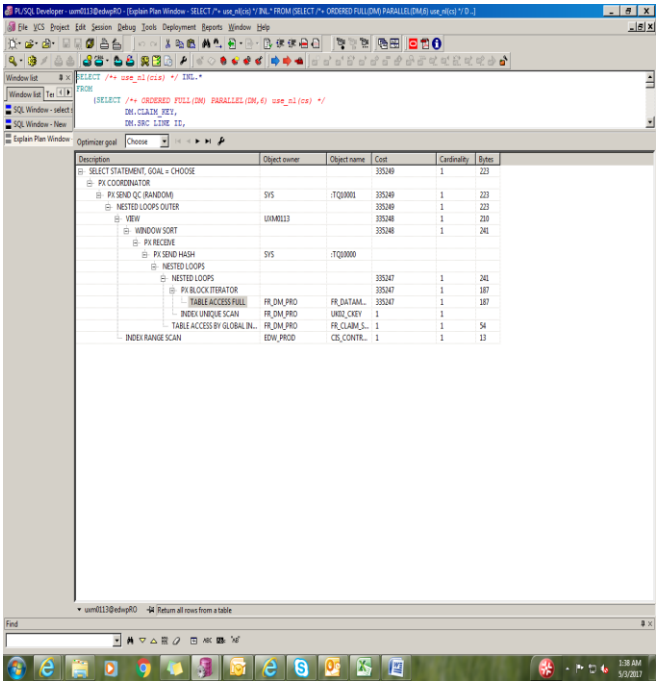
Hash joins is possible for tables with no index (or) either of the big tables has indexed. It's best join for this circumstance. Why it's best join? Because it's worked great for big tables with no index and run the query parallel (more than one processor) and give the best performance. Most of folk says its heavy lifter join.

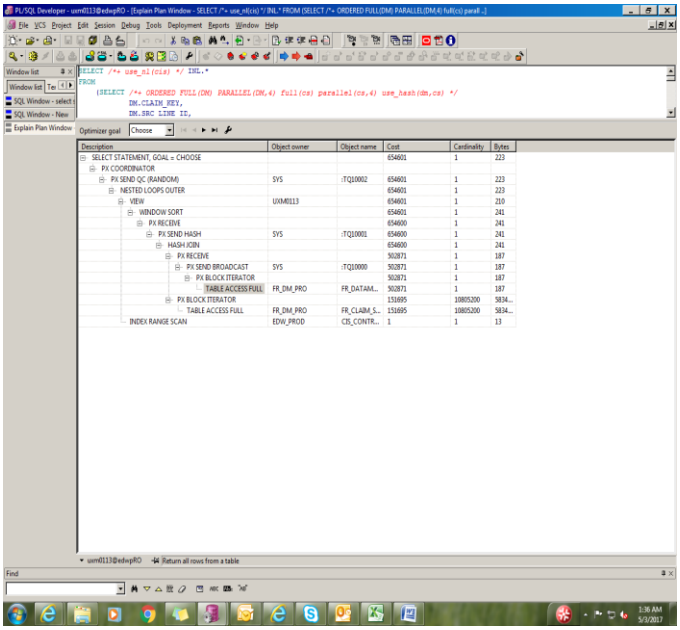
NESTED LOOP JOIN: -

Nested loop join is possible for small tables with index (or) either of the big tables have indexed. It's best join for this circumstance. Why it's best join? Because it works great for small tables like, compares each row from one table to each row from the other table 'looping'.

Analysed tables in real time scenario with applying all the indexes and joins. Checked Query optimization by giving Hash join and Merge Join and nested loop Join to analyse the cost optimization.

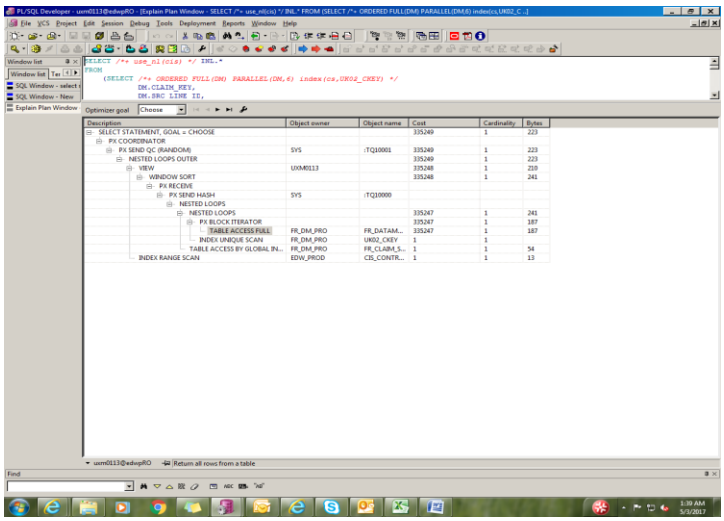
Sql Statements are used to retrieve data from the database. We can get same results by writing different sql queries. But use of the best query is important when performance is considered. So you need to sql query tuning based on the requirement. Here is the list of queries which we use regularly and how these sql queries can be optimized for better performance.





V.CONCLUSION

We have developed a robust database application for the timesheet management system. The tools used are SQL developer for the database server application and JAVA for building the UI. We have reduced Query Cost by verifying Query Optimization and fine tuning techniques. We have also analyzed few tables by applying all types of indexes and joins and verified how it works with Data. We have used the concept of primary index, unique constraint, joins and indexes. Later, we have tested the quality of the application using appropriate test cases.



REFERENCES

[1] DataBase System Concepts sixth edition by Avi Silberschartz, Henry F. Korth and S. Sudarshan