

Databases and Application Development CW 2 James Frew 21808660

by James Frew

Submission date: 08-Jan-2023 04:57PM (UTC+0000)

Submission ID: 193961723

File name:

31653_James_Frew_Databases_and_Application_Development_CW_2_James_Frew_21808660_464571_1398189412.docx
(164.26K)

Word count: 2232

Character count: 12489

Database Development Case Study

James Frew
Student University of Wales Trinity St
David
Swansea, Wales
2108660@student.uwtsd.ac.uk

Abstract— This report covers performance tuning – queries, clustered and non-clustered indexes, and implementing secondary indexes for the Health is Wealth Hospital database. It then goes on to advise them on distributed database solutions (fragmentation) – horizontal, vertical, mixed, and derived horizontal, as well as how to reconstruct them.

Keywords—database, performance tuning, distributed database solutions

I. CONTENTS

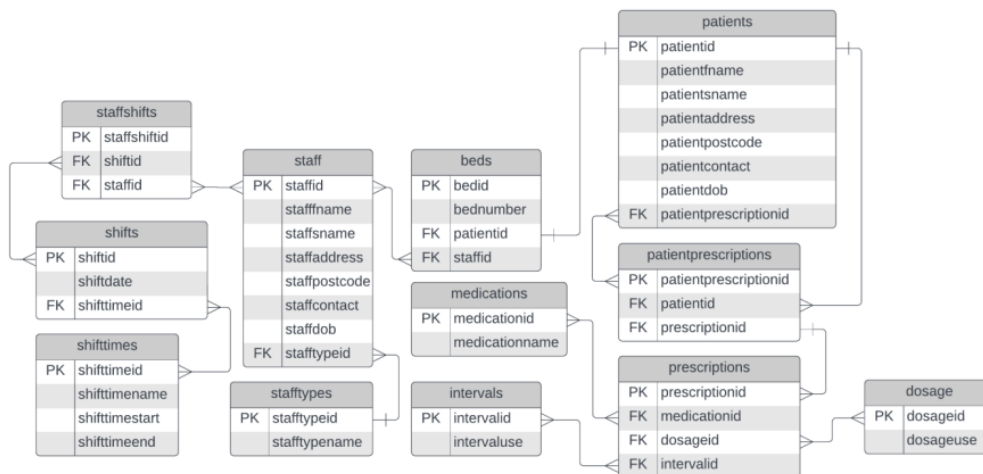
I. Contents	1
II. Introduction	1
III. Entity Relationship Diagram	1
IV. Performance Tuning	1
A. Queries.....	2
B. Clustered Indexes.....	2
C. Non-Clustered Indexes	2
D. Implementing secondary indexes.....	2
V. Distributed Database Solutions.....	2
A. Horizontal Fragmentation	2

B. Vertical Fragmentation	2
C. Mixed Fragmentation	3
D. Derived Horizontal Fragmentation	3
E. Reconstruction	3
VI. Conclusion	3
VII. Additional tools/sites used	3
VIII. References.....	3

II. INTRODUCTION

This report will cover the methods of performance tuning and distributed database solutions that the Health is Wealth Hospital could go about applying to their database. It will go on to discuss queries, and indexes – both clustered and non-clustered, as well as how they are implemented. The distributed database solutions section will cover different methods of fragmentation – horizontal, vertical, mixed, and derived horizontal. Then advise on how they can be reconstructed.

III. ENTITY RELATIONSHIP DIAGRAM



IV. PERFORMANCE TUNING

When attempting to increase the performance of a database, it is important to look at which aspects are most

commonly used. The most used features of the database will benefit more so when they are optimized correctly. Usually,

data queries are the most commonly performed actions in a database.

A. Queries

A query is a request from the user, asking the database to perform a specific action. This can include adding, removing, editing, viewing records, etc. It must therefore be ascertained which of these actions the Health is Wealth Hospital database system users are most likely to frequent. The database has two main purposes of which users interact with it – assigning shifts to staff and assigning prescriptions to patients.

In regard to assigning shifts to staff, both time and date, as well as the relevant staff member must be set using the database. The date and time of each shift should be automated as there are lots of repeat data entries to be performed – which don't require administration. Assigning staff to those shifts can also be automated, but in the event that it is not, an administrator will have to select either each shift and assign a staff member to it or select each staff member and assign shifts to them.

It would be more effective to first retrieve the list of shifts available, then select staff to cover them – as this would enable a more thorough overview of the staff timetable. This creates the first main query – viewing/editing the staff shift table.

The second main query comes from assigning prescriptions to patients. Doctors will be frequently assigning prescriptions to patients they treat so will need to be able to quickly look up that the patient's records to be able to assign them their medication. The creation of new prescription records would also need to be performed, but not as frequently as assigning patients prescriptions – There is a more limited number of variations of prescription than there are patients to treat. The second query would therefore be viewing /editing the patient's table.

A third, less important query would be that of assigning patients to beds – almost all patients will need a prescription, but not all patients will need a bed (but will also need a prescription if they are assigned a bed). This query would involve viewing/editing the bed records to find which are not currently in use.

Now that these core queries have been found, secondary indexes should be created to increase the speed at which they can be performed. There are two kinds of secondary index – clustered, and non-clustered indexes.

B. Clustered Indexes

In a clustered index, the rows of the table are sorted and stored based on their specified values (dictated by the index). This index sets what the order should be. This is done so that users can pick what order they want the data presented to them in (alphabetical, numerical, etc.) – regardless of how it has been saved to the database. There can only be one clustered index per table (as this sets the order and any others would overwrite the first order). [1]

C. Non-Clustered Indexes

A non-clustered index differs to clustered indexes in that they don't actually reorder the data they use. Non-clustered indexes effectively create a new table that only visualizes the data that they use – using pointers in respect to the original

data positions. This means that multiple non-clustered indexes can be used on any table (up to 999 per column). [2]

D. Implementing secondary indexes

The Health is Wealth Hospital database would benefit greatly from implementing secondary indexes. Based on the queries discussed above, the best implementation would be for clustered indexes to be created for the staff shifts, patients, and beds.

The staff shifts should be organized by shift id. This will sort in descending order to show the newest registered shifts first. The shift id can then be matched to whichever staff id should be working that shift.

The patients table should be sorted by alphabetical order of the patients surname. This is because when doctors, (etc.) are searching for a patient, the common practice is to first search by surname, then first name, then date of birth.

The beds table should be sorted by which beds are not currently occupied (patient id), with all free beds first, and occupied beds after. This decision is based upon the idea that an empty bed might need to be found more quickly than an occupied bed in an emergency. [3]

V. DISTRIBUTED DATABASE SOLUTIONS

A distributed database solution is a system in which different files on the in the same database are located on different networks. These different networks can be in the same building or split across different sites – potentially all over the world. A distributed database solution primarily involves fragmentation of database tables. Tables are split into horizontal, vertical, mixed, and horizontal derived fragmentation. [4]

A. Horizontal Fragmentation

Horizontal fragmentation involves splitting up a table by its rows. Only the rows that are wanted are split and used with the others left out. The rows selected are based upon the values that the row contains – a WHERE clause. In the Health is Wealth Hospital database there are a number of uses for horizontal fragmentation. The beds table would benefit greatly from a clause to find which beds are currently unoccupied – in an emergency, finding a spare bed is urgent. The staff table could also check to find which staff member has which staff type (role) in the hospital. This would help with scheduling staff shifts to ensure that the numerical requirement of each staff type is met (ensure there are at least two doctors on each shift, etc.). Finally, the prescription table would use a clause to find all the information about a specific prescription id. This would allow staff to easily check what the dosage is of already made prescription records.

B. Vertical Fragmentation

Vertical fragmentation is similar to horizontal fragmentation, except it splits tables by column instead of by row. Again though, only the columns that are selected will be used and only those whose values meet the requirements set. There are many vertical fragmentations for the Health is Wealth Hospital database. When checking the timings of shifts, a select clause to find the shift names, start times, and end times would greatly benefit any new staff who are trying to understand when their next shift is – e.g., the start and end times of the morning shift, etc. Being able to view the patient

and prescription ids from the patient prescription table would allow nurses to check which prescription a patient has been designated.

The next four tables possessing vertical fragmentation would only slightly benefit the Health is Wealth Hospital database. This is due to the fact that each of them only contains their table id and a single other column with a string name (to better identify the records in that table). The four are as follows:

- Dosage use from the dosages table
- Interval use from the intervals table
- Medication name from the medications table
- And staff type from the staff types table.

The use for these vertical fragmentations is to reduce the amount of information displayed to the user when they access each of these tables – e.g., users will only see the medication name, not its id number. This would only be used to check the full record list for each table to see if a new record needs to be entered.

C. Mixed Fragmentation

Mixed fragmentation is a combination of the two methods – horizontal and vertical (a SELECT and a WHERE clause). The caveat being that one method must be carried out first, then the other – e.g., horizontal then vertical. Either direction of fragmentation can occur first, but it must only occur first – it is not possible to fragment horizontally, then vertically, then horizontally again. The order chosen is based upon which would be best for the specific table in the database. The only table that would benefit from a mixed fragmentation, would be that of the patient table. The Health is Wealth Hospital database users (primarily nurses) will need to check what prescription a patient has. The simplest clause would be to select the patient's prescription id where the patient's first name, surname, and date of birth are equal to the details of the patient they are checking. If needed, this clause could also check against the patients address and postcode (depending on how specific Health is Wealth Hospital feel they want to be).

D. Derived Horizontal Fragmentation

A derived horizontal fragmentation takes a horizontal fragmentation and creates a child fragmentation from it. It bases a constraint on a table that has already had a constraint placed upon it – a WHERE clause inside a WHERE clause. The shift table is one use for derived horizontal fragmentation

in the Health is Wealth Hospital database. The clause would be to select all values from the shifts table where the shift date is equal to the date specified, where the shift name is equal to the shift name specified. This would allow users to find all the dates of the morning shifts.

The final table to benefit from derived fragmentation would be the staff shifts table. This would select all the information from the staff shifts table, where the shift id is equal to the specified value, where the staff type equals the desired value. This would allow users to see all the shifts taken by a specified staff type (job) – allowing shift planners to ensure that the correct number of staff are being assigned to each shift. This would also show if there were too many staff being assigned to the same shift.

E. Reconstruction

Reconstructing global relations is a fairly simple process. Vertical fragmentations are reconstructed using a “full outer join” combining the table columns back together. The horizontal reconstruction uses “union” to recollect the unused rows, and the mixed fragmentation uses both “full outer join” and “union”. [5]

VI. CONCLUSION

This report has discussed the use of queries, secondary indexes, and distributed database solutions. It discovered the best locations for implementing secondary indexes in the Health is Wealth Hospital database. These are the staff shifts, patients, and beds tables. It has also shown which tables would benefit best from horizontal fragmentation – beds, staff, prescriptions. The vertical fragmentation tables are shift times, patient prescriptions, dosages, intervals, medications, and staff types. The mixed fragmentation table is the patient table, and the derived horizontal fragmentation tables are staff shifts, and shifts.

VII. ADDITIONAL TOOLS/SITES USED

- Time syntax [6]
- Varchar minimum length syntax [7]
- List of medications [8]
- Patient and staff information generator [9]
- Unique non-clustered index – used to keep the patient number in each bed unique, but also allow multiple NULL values when beds are empty [10]
- Clustered ordered index syntax [11]

VIII. REFERENCES

[1] ("Nov 18, "). *Clustered and nonclustered indexes described - SQL Server | Microsoft Learn*. Available: <https://learn.microsoft.com/en-us/sql/relational-databases/indexes/clustered-and-nonclustered-indexes-described?view=sql-server-ver16> [Accessed Dec 30, 2022].

[2] (). *SQL Server non-clustered Indexes*. Available: <https://www.mssqltips.com/sqlservertutorial/9133/sql-server-non-clustered-indexes/> [Accessed Dec 30, 2022].

[3] A. Abbas and K. Ahmad, "An integrated indexing approach to improve query processing," in - *2021 International Conference on Electrical Engineering and Informatics (ICEEI)*, 2021, . DOI: 10.1109/ICEEI52609.2021.9611104.

[4] J. Herce-Zelaya *et al*, "Web platform for learning distributed databases' queries processing," *Procedia Computer Science*, vol. 162, pp. 827-834, 2019. Available: <https://www.sciencedirect.com.ezproxy.uwtsd.ac.uk/science/article/pii/S1877050919320678>. DOI: 10.1016/j.procs.2019.12.056.

[5] (). *Fragmentation in Distributed System*. Available:
<https://www.tutorialride.com/distributed-databases/fragmentation-in-distributed-system.htm>
[Accessed Dec 30, 2022].

[6] ("Nov 18, "). *Data types (Transact-SQL) - SQL Server | Microsoft Learn*. Available:
<https://learn.microsoft.com/en-us/sql/t-sql/data-types/data-types-transact-sql?view=sql-server-ver16> [Accessed Dec 30, 2022].

[7] (Oct 8.). *Varchar minimum length check - WebDeveloper.com Forums*. Available:
<https://forum.webdeveloper.com/d/162622-varchar-minimum-length-check> [Accessed Dec 30, 2022].

[8] (). *Medicines A-Z - NHS*. Available:
<https://www.nhs.uk/medicines/> [Accessed Dec 30, 2022].

[9] (). *Fake address generator UK - Random address generator , UK address list*. Available:
<https://www.fakexy.com/fake-address-generator-uk>
[Accessed Dec 30, 2022].

[10] ("Mar 4, "). *SQL Server: Allow Multiple NULL Values in UNIQUE Constraint*. Available:
<https://www.dbrnd.com/2018/03/sql-server-allow-multiple-null-values-in-unique-constraint/> [Accessed Dec 30, 2022].

[11] ("Jan 24, "). *Building SQL Server Indexes in Ascending vs Descending Order*. Available:
<https://www.mssqltips.com/sqlservertip/1337/building-sql-server-indexes-in-ascending-vs-descending-order/>
[Accessed Dec 30, 2022].

Databases and Application Development CW 2 James Frew

21808660

GRADEMARK REPORT

FINAL GRADE

88/100

GENERAL COMMENTS

Instructor

Feedback

Hi James,

You got 88% . Good work. However there is a key area that should have been included in the report as listed below:

1.
The steps showing the normalization process upto the 3NF could have been a good justification for the created tables.
2.
I would recommend you adding screen shots and specific steps showing your approach in the report.

I hope you will find these comments useful.

Should you have any questions let me know

All the Best

Regards

Divya

PAGE 1

PAGE 2

PAGE 3

PAGE 4

