Database modelling for vision impaired indoor navigation systems: Extended Abstract

J.A.D.C.Anuradha Jayakody[#1], Iain Murray[#2], Johannes Herrmann[#3]

Department of Electrical and Computer Engineering, Curtin University, Perth, Western Australia
[1] j.c.jayakody@postgrad.curtin.edu.au
[2] i.murray@curtin.edu.au
[*] Department of Computing, Curtin University, Perth, Western Australia
[3] hannes.herrmann@curtin.edu.au

***Keywords—Data* Model; indoor navigation; AccessBIM; Database Optimization.**

Cooperative research has been conducted to improve navigation services for vision impaired when they are moving through an indoor environment. Hence, to facilitate vision impaired individual in indoor environment requires a formal modelling approach for map generation and decision making about navigation pathways avoiding obstacles.

The proposed data model consists "AccessBIM" database (DB) and API functions. The "AccessBIM" features such as, database connection initiation, function call, function return and database connection termination are organized as a series of function objects that meet the various needs of the database to maintain the relational schema.

The proposed API primarily consists of two components, namely; DB connector and the API functions. Proposed "AccessBIM" DB will be implemented using real-time database such as "PostgreSQL". The weighted focus will be given to the areas such as Queries, Indexes and Transactions in relation to tuning the DB and the queries. The performance of the proposed API will be evaluated based on the time required to parse, and Data insert rate and retrieval rates.

A. Queries

Within this phase focus will be given to practice the following techniques in order to optimize the performance of the existing queries:

- Return only the rows and columns needed.
- Avoid expensive operators such as NOT LIKE.
- Avoid explicit or implicit functions in WHERE clauses.
- Use locking and isolation level hints to minimize locking.
- Use temporary tables and table variables appropriately.

B. Indexes

Indexes are vibrant to efficient data access. But as a drawback, there is a cost associated with creating and maintaining an index structure. Having a large number of indexes on a table may result in faster select statements, but slower insert, update and delete statements. Therefore, attention will be given to balance between the above trade-off while creating efficient indexes. Expecting to use the following guidelines to implement indexes efficiently:

- Remove unused indexes.
- Use the Index Tuning Wizard.
- Create indexes based on use.
- Create an index on all foreign keys.
- Consider a covering index for often-used, high-impact queries
- Use multiple narrow indexes rather than a few wide indexes.
- Create composite indexes with the most restrictive column first.

C. Transactions

With a vision of enhancing the scalability of the DB efficient handling of the transactions will be given a higher consideration. Transactions hold locks on resources that can block other transactions. The following techniques will be implemented to create efficient transactions:

- Avoid long-running transactions.
- Avoid transactions that require user input to commit.
- Try to access resources in the same order.
- Use isolation level hints to minimize locking.
- Ensure that explicit transactions commit or rollback.

D. Proposed Performance Evaluation

The performance of the proposed API will be evaluated based on the following metrics.

- Time required to parse, compile, and execute all the queries $(\mathbf{Q_{Total}})$ in milliseconds.
- Number of queries used within the API: n Time is taken to parse, compile, and execute a given query n (in milliseconds): $\mathbf{Q_{n_T}}$

Therefore, Q_{Total} can be defined as $Q_{Total}=\sum_{i=1}^{n} Q_{n_T}$

*Q_{Total} is calculated for a scenario where all the ***n*** queries are executed simultaneously.

*Q_{n_T} can be measured using SET STATISTICS TIME { ON | OFF } in SQL Server

Data insert rate and retrieval rates

- Data insert rate: Amount of data (in KB) insertion per second
- Data retrieval rate: Amount of data (in KB) retrieval per second.

This paper presents a data model with novel functions of proposed API can be used to either insert, update or delete sensor data to/ within the DB. Each function contains a specialized query which can connect to the DB tables in order to perform the necessary transactions to assist vision impaired in an indoor navigation environment, so as to create an indoor map that can support the vision impaired people to navigate using their smart phones. This model proposed a database optimization and evaluation methods to measure the performance. The key idea is to build an open, participatory and collaborative system through which users can contribute environment information using their mobile devices.