Asian Journal of Information Technology 16 (9): 699-705, 2017
ISSN: 1682-3915

Analyzing and Improving the Performance of Decision Database with Enhanced Momentous Data Types

[1]M. Muthukumar, [2]P. Senthil Pandian and [3]K. Karthikeyan
[1]Delivery Project Lead, Mphasis Limited, Bangalore, Karnataka, India
[2]BIT Campus, University College of Engineering, Trichy, Tamil Nadu, India
[3]Department of Computer Applications, Anna University, Madurai, Tamil Nadu, India

Abstract: In the current digital age, volume of data is getting increased by 59% on every year worldwide. According to research, it is estimated that 70-85% of data is unstructured and more than 20% performance degradation happens due to improper database design. There are many factors which may influence the performance of the database among the factors a data type plays an important role. Choosing the right data type for each column is most essential in either heavy transaction or less transaction tables. Proper database design provides multiple benefits like minimized disk usage, less consumptions of CPU and resources and improved query performance. The proper usage of data types in the database improves the performance of the whole system. These data types are of different types and should be used in an intelligent way to achieve the expected behavior. This study demonstrates how to choose the correct data types to design efficient database and also identifies which data type should be used or not to be used.

Key words: Data decision types, database decision design, disk space utilization, bytes consumption, query execution time, CPU

INTRODUCTION

SQL server stores data in a special structure called data pages that are 8 KB (8192 bytes) in size. This means in 1MB, 128 data pages can be stored in SQL server. First, 96 bytes of the each page is used to store system information about the page, next 8060 bytes are used to store data of users and remaining 36 bytes are used for storing row off set details. Total number of bytes used in single page is 96+36+8060 = 8192 bytes. If one row size comprises 4040 bytes, then only one row can be placed on each data page. If the size of the row is decreased to 4030 bytes, two rows can be stored with in a single page. As a result, the tables can be designed in such a way to maximize the number of rows that can fit into one data page. For this purpose, the narrowest columns should be specified as much as possible. Due to narrower columns, the less data is stored and the SQL server will be faster. Hence, it is able to read and write data.

Currently, many approaches and techniques are available in the market to enrich the database performance. Nevertheless, various options available through data types are discussed in this study. The approach introduced in this study is generic and it does not depend on any particular database like SQL server, IBM DB2 Oracle, Sybase and etc. Before creating the database, one must make sure that what type of data will be stored in the particular column. This approach can be implemented in any database environments and every database developer should keep in mind about the factors that are going to be discussed in the forthcoming topics during designing the table schema. The proper design schema will result in faster query execution and provides an overall better performance. This will give considerable performance improvements in CPU execution, resource utilization and space reduction. Currently, there are many approaches and tools available in the market for troubleshooting, tuning and optimizing database performance. But using this proposed unique methodology, the usage of external tools can be avoided at some extent. When large amounts of data and transactions are involved, the maintenance cost of the storage system easily exceeds the cost of the data server. In real-time environment, small improvement/reduction also plays a very significant role in cost savings.

Choosing an appropriate data type is very important because the errors made in a table design can result in large performance degradation. These problems often surface later when a large amount of data is inserted. In every relational database system each column, local variable, expression and parameter have related data type and it specifies the type of data that the object can hold.

Corresponding Author: M. Muthukumar, Delivery Project Lead, Mphasis Limited, Bangalore, Karnataka, India

The examples are integer data, character data, monetary data, date and time data, binary strings and so on. All database system supplies a set of predefined data types that define all the types of data that can be used. Generally, data types are organized in multiple categories such as exact numbers, approximate numbers, date and time, character strings, Unicode character strings, binary string and other data types. Almost, all the databases provide the above data types but getting actual performance of these data types is based on how and where these data types are utilized. There might be minor differences among various databases in the way data types are promoted. Here, SQL server data types are discussed for experimental purpose. However, the same approaches can be implemented with any other relational database system as well.

MATERIALS AND METHODS

Importance in opting right decision data types: Relational databases were developed in 1970's as a technology to store structured data organized as tables and with its own query language model called Structured Query Language (SQL). Very soon, the databases have become vital parts of organizations and used all over the globe. However with the constant growth of data, relational databases exhibit a variety of limitations (Pons, 2003; Anonymous, 2014). Data querying loses efficiency due to the large volumes of data and that the same time, the storage and management of larger databases become challenging. SQL server databases are developed to provide a set of new data management features while overcoming some limitations of currently used relational databases. In comparison to relational databases, SQL server databases are more flexible and very user friendly and have robust UI to manage all the activity (Corlatan *et al*., 2014).

Modern RDBMs support more flexible and elegant data types. Generally, there are two different categories of data types in high level on every database. They are fixed length and variable length data types. There are few data types which come under fixed length and few come under variable length. Another consideration is when summing a precision based value, the resulting summed values data type will be the same as the column definition. That way, the sum result will have more space and perhaps some casting can be avoided. From a constraining perspective, a decimal (5.2) might be more appropriate but maybe constraints requirements shouldn't be mixed up with data type decisions (Cioloca and Georgescu, 2011). A decimal (25.5) value of 999.999 takes 13 bytes, not 5 bytes. Even a NULL value will take 13 bytes.

Fixed length type: Fixed length type is used when the sizes of the column data are consistent. For fixed-length data types, the provider will allocate enough space to hold one item of that type (England, 2001). Especially, fixed length data types must be handled cautiously. Because space taken by value entries of this type will have always same size. It's the column definition that defines how much space each entry takes and not the size of the value. Fixed length will allocate memory of the specified size whether it is filled or not. If it is known how long the data will use character data type. This will save the memory in bytes. The benefits on using fixed length data types are CPU-wise, there is a performance gain when working with fixed-length data. Processing fixed-length columns are faster than processing variable-length columns.

Variable length type: Variable length will allocate memory based on the number of characters in it. VARCHAR (10) with "Yes" will take 3 bytes of storage. For variable-length data types, the provider must rely on the consumer to specify how much memory must be allocated for the data. Database designer has to be very vigilant during selecting the data types to avoid the performance issues. The designer should be proactive and plan properly during the time of designing the database. She/he should consider all the aspects of the data to accomplish the need of the customers as well as minimize the storage space and maximize the query processing (Kumari, 2012). By following certain standards during the database design, the data storage can be easily reduced in database.

Significant reasons in selecting right decision data types: While designing real time database, almost more challenges will be faced to build proper database (Kalaiselvi and Kavitha, 2013). Approximately, complex application database will have minimum of 30 and more tables. If the proper standards are followed, definitely some considerable benefits will be achieved. This will not only result only space savings but also will improve the overall performance.

Use fixed length data types if truly need it and when the values are mostly about the same size. Use a variable length to store large strings of data. This can greatly reduce I/O read cache memory used to hold data and improves overall query performance. Another advantage of using variable length type over fixed length is that sorting operation performed on variable length columns is generally faster than fixed length columns. This is because the entire width of a fixed length column needs to be sorted. Don't use TEXT/NTEXT as it has been deprecated. VARCHAR(max)/NVARCHAR(max) provides

the same functionality. But these fields cannot be indexed. TEXT/NTEXT data types have extra overheads that drag down the performance. Use Unicode data type, only when working with an international and multilingual application. This will take space twice as VARCHAR data type. This will increase server I/O and waste unnecessary space in the buffer cache. Use the smallest data type wherever is possible and also, make sure, it can accommodate the largest possible value as well. Avoid using FLOAT or REAL data types for primary keys as they add unnecessary overhead that hurts performance. Instead use one of the integer data types.

Preferable advantages: SQL server has a plenty of data types. When there are more options, more confusion arises. Mostly, confusion arises from data type limitations rather than functionality. Always specify the low space consume data type to store less amount of data. Read and write operation will be faster when small data types are used and in addition if any sorts need to be performed on the column that operation is also be faster. For example, to store the column from the numbers 1 through 10, the TINYINT data type is more appropriate than the INT data type. The same thing happens for CHAR and VARCHAR data types. More characters should not be specified in character columns. This allows storing more rows in data and index pages by reducing the amount of I/O needed to read them. It also, reduces the amount of data moved from the server to the client by reducing network traffic and latency. Finally, it reduces the amount of waste space in the buffer cache. Choose the data types TINYINT/SMALLINT instead of INT/BIGINT, bit instead of VARCHAR/INT, DATE/TIME instead of DATE TIME, SMALL MONEY instead of MONEY and VARCHAR instead of TEXT wherever is possible. Few of the best scenarios are given below for reference.

State 1: If a column that is designed to hold only numbers, use a numeric data type such as INTEGER instead of a VARCHAR or CHAR data type. Numeric data types generally require less space to hold the same numeric value as does a character data type (Dam and Fritchey, 2009). This helps to reduce the size of the columns and can boost performance when the columns use WHERE clause ORDER BY and JOIN with other tables.

Use TINYINT data type instead of INT when designing the tables for "department, role, job and country". Mostly, these column (ex: department ID and country ID) will store values ranging from 1-100 and will save three bytes per record. The columns with Tinyint will use only one byte to store their values. Most of the organization won't have the department more than maximum value of TINYINT data type of 255. For 100,000 records, 300,000 bytes will be saved. If there are indexes containing this columnand if these indexes take less memory, database engine will process this index much more efficientlyin every "JOIN" and "WHERE" clauses (Vimala *et al*., 2013). So, queries wil lperform faster and system resources use less memory and CPU. This will make the whole server to perform better as there will be more resources available for other things.

State 2: BIT data type can be used instead of INT for "Sex and Status" columns. It means that for storing the values like 1/0, TRUE/FALSE, ACTIVE/INACTIVE, YES/NO and ON/OFF. BIT data type will research similar to Boolean type. This will use only one byte to store their values and save three bytes per record as compared with INT data type. Most of the employee/user tables have employees "Sex or Status" column. This indicates the employee's sex and current status. Every large scale organization is employee in flow and out flows are not retractable. They should maintain the records for their existing employee as well. Over the period, the size of this table will get increased and eventually space consumption will also get increased accordingly.

State 3: Use DATE data type instead of DATETIME for "date of birth, date of joining and date of relieving, approved date, travel date, return date" columns. DATE data types use only 3 bytes to store their values, so, each of the above columns will save 5 bytes per record. Almost, all these columns will be available in employee/user table to store their respective dates. According to the business requirement, few more date columns also might exist in the tables. By considering all these columns, significant amount of bytes can be saved per record. For example, if three DATETIME columns are used 15 bytes can be saved per records. With 100,000 records, 1500,000 bytes will be saved easily.

State 4: If fixed length columns (CHAR, NCHAR) are used in the table, consider it avoid storing NULLs in them. If it is done, the entire amount of space dedicated to the column will be used up. For example for a fixed length column of 255 characters a NULL is placed in it and then 255 characters have to be stored in the database. This is a large waste of space and that will cause SQL server to perform extra disk I/O to read data pages. It also wastes space in the data cache buffer. Both of these contribute to reduce the performance of SQL server.

Really, to use NULLs use a variable length column instead of a fixed length column. Variable length columns use only a very small amount of space to store a NULL. A

CHAR (10) means, reserving 10 characters for every single row whether they are used or not. Use this for something that shouldn't change lengths like EMPID, ManagerID, Zip Code or SSN. VARCHAR data types use only 1 byte to store 1 character. The NVARCHAR value uses 2 bytes to store 1 character. Hence, the VARCHAR columns use 2 times less space to store data compared to NVARCHAR columns. NVARCHAR data type can be used for designing database for multilingual.

State 5: Use SMALL MONEY and data type instead of using MONEY data type to store employee monthly salary incentive and allowances. SMALLMONEY and data types use only 4 bytes to store their values but MONEY data types will use eight bytes. By avoiding MONEY data type on every amount columns, significant amount of bytes per record can be saved.

RESULTS AND DISCUSSION

Experimental configuration: To enable the better performance a good system configuration scheme and correct database software are needed. System configuration scheme depends on the system throughput, the system stability and user maintenance ability.

Software configuration: For the experimental analysis, SQL server 2008 R2 as database is used. It is widely used in all large scale industries and data warehousing. In order to evaluate the performance 2 databases with each 5 tables are used. In tables, first database contain the columns with legacy Data Types and second one contains correct data types as suggested in this study. Name of the first legacy database is called data types and second one is called Data Types-Well Defined. Number of tables, number of columns, schemas, records and data are exactly same in both data bases (Fig. 1 and 2). Only the difference is data types used for the columns in the legacy database are flexible and well defined databases are very narrow. For experimental purpose 5 tables are used and each table has multiple columns with the combination of various data types. Main table is called as employee and rest of the 4 tables (department, country, job and role) are used for holding the master data. Each master table has the primary key and relationship with main employee table. For experimental purpose 30,000, 50,000 and 1,00,000 records are loaded in the main table (employee) with various combinations of semantic data to evaluate the disk usage, CPU utilization and bytes occupied. Each record in main table is uniquely identified by a key composed by string "EmpID". Each field of the record contains the personal information about employee based on different data types.

Here, the SQL queries are used with relatively simple to fairly complex to measure the performance with various aspects. Each of the test scenario is called work load

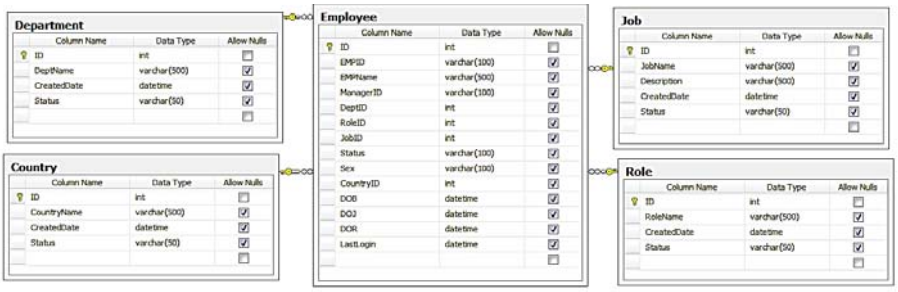

Fig. 1: Database diagram for legacy data types

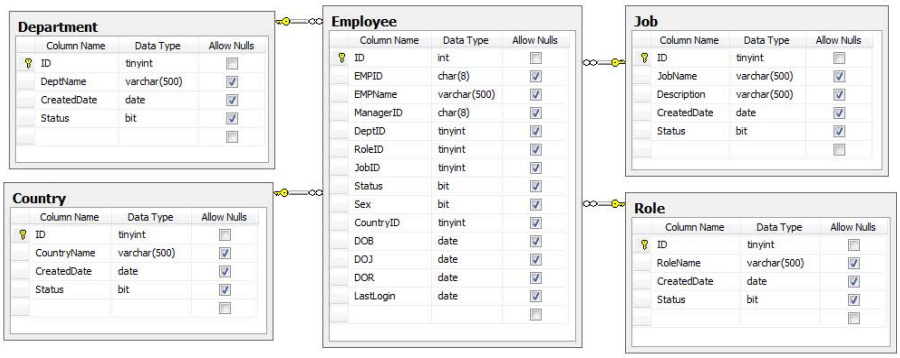

Fig. 2: Database diagram for well-defined data types

Table 1: Disk space utilization

	No. of records					
	30000		50000		100000	
DB data types	DRS* (KB)	SUSAD** (KB)	DRS* (KB)	SUSAD** (KB)	DRS* (KB)	SUSAD** (KB)
Legacy	3688	3448	6344	6152	11464	11384
Well defined	2096	1952	3272	3240	6536	6488
Space savings (KB)	1592	1496	3072	2912	4928	4896
Space savings (%)	43	43	48	47	42	43

*DRS: Default Reserved Space by SQL server; **SUAD: Space Used to store Actual Data

Table 2: Bytes consumption

DB data types vs. No. of bytes used	No. of records 30000	50000	100000
Legacy	2570078	4226999	8437557
Well defined	1487851	2466400	4934067
Bytes savings	1082227	1760599	3503490
Bytes savings (%)	42	42	42

and is defined by a set of scenario such as simple query with single table, medium query with join condition and complex query with multiple tables. However, for better understanding the above scenario is executed in both databases with multiple times.

System configuration: All the tests are executed on a physical machine Intel(R) Core™ i5 CPU @ 2.5 GHz processer with 64 bit operating system, hosted on a computer with Windows 7 enterprise with service pack 1 with a total of 4 GB RAM. It is important to notice that microsoft SQL server 2008 R2 with 64 bit is used for the database.

Experimental evaluation: In the following sub section, the result of executing the SQL query with 3 sample set of records is analyzed. The SQL server is tested using a mix of real time examples to better understand the various data types and how the performance is affected internally by each data types. The usage of data types and the performance of the query execution are measured in terms of:

- Disk space utilization
- Bytes consumption
- Query execution time

Non unicode character based data types require 1 byte to store a character. Unicode character based data types require 2 bytes to store a character. Each and every type has set of unique features and limitations. These features and limitations must be thoroughly understood to design the proper database. By minimizing more than 20% disk space, better performance can be achieved based on the database design. If database design is not proper, the real performance cannot be achieved in application and it will become very difficult to satisfy the customers. Any organization must provide the service to the customers round the clock and should satisfy the customers highly. The time duration for window maintenance and system up gradation is really a challenging task for IT companies. Majority of the risk factors can be avoided by providing proper database design.

Disk space utilization: This provides detailed information about how much space has been consumed to store the data in a specific database. Disk space usage has been calculated based on the space used for data storage and reservation. Here SQL server tool is used to evaluate the space usage for the database design using legacy and well defined data types. The disk space used by each data file in the database is captured. From Table 1, space savings denote the actual space currently used by the file, excluding allocated space.

The above table describes the process of space savings based on different legacy and well defined data types which have different number of records. The results have been captured based on the tables which have 14 columns with various data types as mentioned in Fig. 2. Three different quantities of records like 30,000, 50,000 and 1,00,000 are used. Compared to legacy data types, the well defined data types achieve less storage space and the variance would be about 43-48% better. The equation given is used to calculate the space savings:

$$\text{Space saving} = \left(\frac{\text{Space different}}{\text{Space used in legacy DB}}\right)\times 100 \quad (1)$$

$$\text{Space differences} = \text{Space used in legacy DB-space used in well defined DB}$$

Bytes consumption: The performance tuning of database and solutions have been around for many year. The performance of SQL server database should be evaluated from several different perspectives (Table 2). The optimal performance approach is calculated based on all valuable system information including waits, queues and other non-SQL server performance information. There could be

Table 3: Query execution time

DB data types vs. CPU time (msec)	No. of records: 30,000 No. of trial 1	2	3	Avg.	50,000 No. of trial 1	2	3	Avg.	10,0000 No. of trial 1	2	3	Avg.
Legacy	251	279	265	265	452	468	472	464	889	874	908	890
Well defined	212	232	251	231	406	419	433	419	812	795	853	820
Time savings	34	-	-	-	45	-	-	-	70	-	-	-

multiple reasons for performance failure in every database. Trouble shooting the bottle necks is very difficult and more time is needed to investigate the root cause. However, the performance issue from the bottom level can be fixed using this new approach. From the experimental results, it is examined that the database designed using well defined types achieves better bytes savings in each column and also these data can be queried fast compared to legacy types. Table 2 contains the values of summation of bytes consumed for all 14 columns in main tables employee, employee1 and employee 2. From Table 2, it is clear that the new approach is superior in all aspects irrespective of data base. The bytes savings is increased by 42% and the consumption is lower than legacy type.

Query execution time: When a query is submitted to SQL server to execute, it has to parse and compile for any syntax error and optimizer has to produce the optimal plan for the execution. SQL server parse and compile time refer to the time taken to complete this pre-execute steps (Table 3).

CPU time: CPU time is just the time for which the CPU is used by the query when it runs. It is measured in milliseconds. This time refers to the actual time spent on CPU and this is a base line for performance tuning. This is relatively independent of how busy the CPU is. The CPU time is relatively consistent and it can be used as a way to help to determine whether the changes made to the queries, during performance tuning are actually helping or hurting. The CPU time can be used to baseline the performance tuning. This value will not vary much from execution to execution, unless the query or data are modified.

Elapsed time: Elapsed time is the amount of time taken for the query to execute from start to completion. Of course, it would be larger than CPU time because it includes time spent during I/O operations required by query and network delay. Elapsed time may be different for same query syntax on the same server during different execution time because it depends upon the availability of other resources. The elapsed time will depend on many factors like load on the server, IO load network bandwidth between server and client:

$$\text{Elapsed time}(\text{total running time}) = \text{CPU time} + \text{Waits} \quad (2)$$

For experimental purpose, only few tables are used but while working in real-time which has more tables. The total performance of the system will get increased significantly. To capture accurate results SQL server instance has been restarted for multiple times. Average CPU time has been calculated based on three trails in each maintable (employee, emplyee 1 and 2). The proposed approaches achieve good performance in time saving and the variance would be about 34-70 mL/sec.

CONCLUSION

In this study, the performance of some of the most promising examples is analyzed and tested against research loads with simple and complex queries. By analyzing the performance, the results are interpreted in the light of the characteristics of the query performance. In particular, disk usage, CPU utilization and bytes consumption are discussed to reach evaluation analysis conclusion.

Varchar should be used for alphanumeric columns instead of using CHAR or TEXT. Data types such as TINYINT, SMALLINT, BIT and DATE are better to be used when designing table which hold a small amount of data. When these data types are used, the performance of the system is increased. Unmatched data type can dramatically affect the performance, query execution time and bytes used for the data store. Therefore, choosing the correct data type is an extremely important task.

REFERENCES

Anonymous, 2014. Mission critical performance and scale with SQL server and windows server technical white paper. Microsoft Corporation, Redmond, Washington, USA.

Cioloca, C. and M. Georgescu, 2011. Increasing database performance using indexes. Database Syst. J., 2: 13-22.

Corlatan, C.G., M.M. Lazar, L.U.C.A. Valentina and O.T. Petricica, 2014. Query optimization techniques in microsoft sql server. Database Syst. J., 5: 33-48.

Dam, S. and G. Fritchey, 2009. SQL Server 2008 Query Performance Tuning Distilled. 2nd Edn., Apress, New York, USA., ISBN: 978-1-4302-1902-6, Pages: 497.

England, K., 2001. Microsoft SQL Server 2000 Performance Optimization and Tuning Handbook. 1st Edn., Digital Press, Wisconsi, USA., ISBN:9780080479453, Pages: 320.

Kalaiselvi, R. and V. Kavitha, 2013. Efficient technique for authentication and integrity of data in grid using CSTA. Asian J. Inf. Technol., 12: 318-324.

Kumari, N., 2012. SQL server query optimization techniques-tips for writing efficient and faster queries. Intl. J. Sci. Res. Publ., 2: 1-4.

Pons, A.P., 2003. Database tuning and its role in information technology education. J. Inf. Syst. Educ., 14: 381-387.

Vimala, S., H.K. Nehemiah, R.S. Bhuvaneswaran and G. Saranya, 2013. Design methodology for relational databases: Issues related to ternary relationships in entity-relationship model and higher normal forms. Intl. J. Database Manage. Syst., 5: 15-37.