The value of IBM DB2 for z/OS hash access for OLTP transactions

Improve response times and reduce the cost of queries

Highlights:

- Helps reduce costs and improve overall response time for access to single rows of data
- Can improve response times and reduce the cost of queries that retrieve single rows from a table containing unique keys
- Can yield database CPU usage reduction of up to 37 percent and reduction in get pages and synchronous I/Os, compared to traditional indexed data access
- Can reduce elapsed time and resources—including CPU time, memory and I/O—needed to access single rows of data
- Can provide significant benefits to many high-volume OLTP applications that need to efficiently and quickly retrieve a single row of data through a fully qualified primary key

The need to reduce costs and optimize business operations is among the top priorities faced by many worldwide businesses today. Access to single rows of data is essential to many high-volume online transaction processing (OLTP) applications. When used in appropriate situations, access to tables that are organized by hash has been demonstrated to result in database CPU usage reduction of up to 37 percent and reduction in get pages and synchronous I/Os, when compared to traditional indexed data access.

Why hash organization?

Hash organization can reduce the elapsed time and resources—including CPU time, memory and I/O—needed to access single rows of data, by replacing more costly access methods, such as index access.

The new IBM® DB2® 10 for z/OS® solution uses hash organization to improve response times and reduce the cost of queries that retrieve single rows from a table containing unique keys. DB2 for z/OS can use a unique value that identifies the data to efficiently compute the probable physical location of a single row of data in the data pages of the table space.

How it works

When you organize a table for hash access, DB2 applies a sophisticated hash algorithm to the unique key value for each row to determine the physical location of the row in the table space. The hash algorithm distributes data values in a random fashion throughout the table space for consistently optimal query performance.

Because different unique keys can have the same hash value, the same physical location is sometimes identified for more than one row of data. However, when DB2 encounters that situation, it accounts for that and places one of the rows where it can be efficiently accessed.

When queries specify equality predicates on the unique key value that DB2 used to organize the data, DB2 can apply the same hash algorithm to efficiently compute the physical location of the row of data and retrieve that row. For example, consider a bank's high-volume OLTP applications that routinely access the CUSTOMER_TABLE table to retrieve a single customer's row of information by using the following query:

SELECT * FROM CUSTOMER_TABLE WHERE CUSTOMER_NUMBER = 2378

The value of the CUSTOMER_NUMBER column is unique in the CUSTOMER_TABLE table.

If the CUSTOMER_TABLE table is organized for hash access, DB2 can apply the hash algorithm to the key value "2378" to compute the probable physical location of the row in a particular data page. Although some computation is required to apply the hash algorithm, the cost of retrieving the row is significantly reduced, and the bank benefits from the savings.

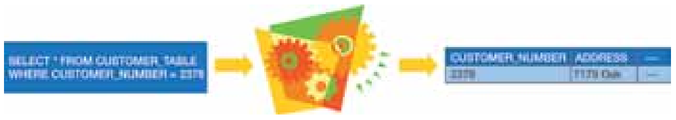

Figure 1: Single-row access through hash.

By comparison, if index access is used to determine the location of a row within the data pages, DB2 must traverse the index pages to find the index page that contains information about the location of the row. The result is greater CPU, memory and I/O usage, especially if the index contains many levels.

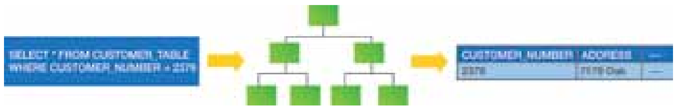

Figure 2: Single-row access through unique index.

Fast random access to rows

Hash access is particularly valuable for applications that randomly access rows in a table. When an application uses equality predicates on a unique key to locate a row, the savings from using hash access can be substantial, particularly in cases where the index access would have to traverse an index that is many levels deep. Some examples of these kinds of accesses are random lookups by bank or insurance policy account numbers. Because the base technology of hash access results in randomization of the rows within a fixed size hash data area, hash access is not intended for accesses that currently scan ranges of index keys, or that access tables varying drastically in size. When used appropriately, hash access provides very fast access to random individual rows in a table.

Proven cost reductions

IBM performance analysis has shown that a table organized by hash spaces can provide significant cost savings when applied in appropriate situations.

For example:

- Thirteen percent class 2 CPU usage reduction was observed for a SELECT statement that retrieved all columns of a single row from a table organized by hash, by using a fully qualified key, as compared to access through an index (with three index levels) on the same columns that defined the hash key.
- Thirty-seven percent class 2 CPU usage reduction was measured for 50,000 executions of a SELECT statement that retrieved all columns of a single row from a table organized by hash of the statement, as compared to access through an index (with three index levels) on the same columns that defined the hash key.

- Nine percent class 2 CPU usage reduction was measured for a SELECT statement that retrieved a subset of columns from a table organized by hash, when compared to index-only access (with three index levels) on all columns.

In each case, reduced get page and synchronous I/O operations were also observed.

Some hash organization best practices

Evaluate the applications and workload thoroughly before adopting hash organization. Hash-organized tables deliver the most reductions and response-time improvements in certain specific situations, which include the following:

- The table has a unique key.
- Queries that access the table specify equality predicates on unique values to return a single row of data.
- Most access to the data in the table is truly random. Applications that use range scans, or that depend on clustered data, do not perform optimally with hash-organized tables. You can use Instrumentation Facility Component Identifier (IFCID) 199 to verify that access is truly random.
- The size of the data in the table is relatively stable, or the maximum size of the data is known. The amount of space that must be dedicated to a hash-organized table is fixed.
- Many rows fit on a single data page. When too few rows fit within a single data page, additional space may be required to achieve the benefits of hash organization.
- The table contains rows of relatively uniform size.
- The benefits of hash access are greatest when an index on the table's unique key would have more than three levels.

After adopting hash organization, monitor real-time statistics to ensure that hash access is used, and tune the size of the hash space.

Conclusion

The hash organization capabilities of DB2 10 for z/OS can provide significant benefits to many high-volume OLTP applications that need to efficiently and quickly retrieve a single row of data through a fully qualified primary key. This specialized capability performs best under specific conditions which require evaluation before adoption. Hash access, when used appropriately, can provide significant cost savings over indexed data access.

For more information

To learn more about the IBM DB2 for z/OS hash access for online transaction processing (OLTP) transactions solution, please contact your IBM marketing representative or IBM Business Partner, or visit the following website: **ibm.com**/developerworks/data/library/dmmag/DMMag_2010_Issue2/idug/index.html

Authors

- Paul McWilliams, DB2 for z/OS Information Developer
- Kalpana Shyam, DB2 for z/OS Software Developer
- Karelle Cornwell, DB2 for z/OS Software Developer
- Rebecca Poole, DB2 for z/OS Business Development Executive
- Bob Lyle, DB2 for z/OS Software Developer, STSM

SWW14011-USEN-00