


## Caching Query Results with QuickSelect Improves Db2 z/OS Workload Performance

By Craig S. Mullins

A decorative graphic on the left side of the page, consisting of a network of interconnected nodes and lines, resembling a molecular structure or a data network. The nodes are represented by small circles of varying shades of gray, and the lines are thin, light gray lines connecting the nodes.

In today's fast-paced digital landscape, the ability to access and process data as quickly as possible is of utmost importance for businesses striving to gain a competitive edge. As such, a significant trend in the database world is utilizing memory for more types of data management and processing functions. If you can bypass disk I/O, you can achieve tremendous performance gains. There are many reasons for this, but of course, the most important reason is that disk access is much slower than memory access.

Traditional disk-based data processing systems, while reliable, often struggle to meet the demands of modern applications that require real-time analytics, rapid transactions, and seamless scalability. Enter in-memory data processing, a transformative approach that harnesses the power of memory to revolutionize database systems.

In-memory data processing refers to the practice of storing and manipulating data primarily in main memory (RAM) rather than on disk. By leveraging the high-speed access time of memory, organizations can unlock a myriad of benefits that traditional disk-based systems simply cannot deliver.

Accessing data in memory is orders of magnitude faster and more efficient than reading data from disk. Memory access is usually measured in microseconds, whereas disk access is measured in milliseconds (1 millisecond equals 1,000 microseconds).

## Using Db2 Buffer Pools

To take advantage of memory, most relational database management systems, including Db2, utilize buffer pools to cache frequently accessed data in memory. This cache resides in RAM and helps reduce disk I/O by serving read requests directly from memory whenever possible. By intelligently managing the buffer pool, database systems can optimize memory utilization and minimize latency, thereby improving overall performance.

Db2 for z/OS uses buffer pools to store data pages retrieved from disk. Each buffer pool is configured with specific parameters such as size, number of pages, and page size. When Db2 needs to access data to execute a query, it first checks if the required data pages are in memory (buffer pool). If the pages are not in memory, it initiates a page fetch operation to read the required pages from disk into the buffer pool. So, buffer pools operate at the page level for Db2 data.

But utilizing buffer pools to reduce I/O is just one approach for leveraging in-memory data processing. Another useful technique is query result set caching.

## Db2 Query Result Set Caching

For purposes of this discussion, a unique query is defined as the combination of a specific SQL statement and specific host variables. The query's result set is thus unique to the query. As the name implies, a query result set cache stores query results delivered by Db2 and then serves these back to programs in response to subsequent requests for the same results. This saves Db2 the work of repetitively generating identical results from data within pages stored in a buffer pool. By caching and re-serving result sets for frequently run queries, you can significantly reduce application wait time and overall performance. When a query, whose result set has been cached, is run again, the caching application swiftly retrieves the corresponding result from memory. This bypasses not just the potential need for disk I/O but more importantly, Db2's costly computation of the same result set over and over. This translates into accelerated batch throughput, improved online application responsiveness and enhanced user satisfaction.

## QuickSelect for Db2 Workload Performance

Although query result set caching is not a built-in capability of Db2 for z/OS, you can implement a query result set cache by deploying [QuickSelect for Db2](#), an in-memory optimization product from [Log-On Software](#).

For cached result sets, QuickSelect will return the same result set as Db2 would. By caching the result sets of frequently run queries, QuickSelect can significantly speed up result set delivery. QuickSelect runs as a started task and automatically caches frequently requested query result sets in self-managed memory above the bar (64-bit).

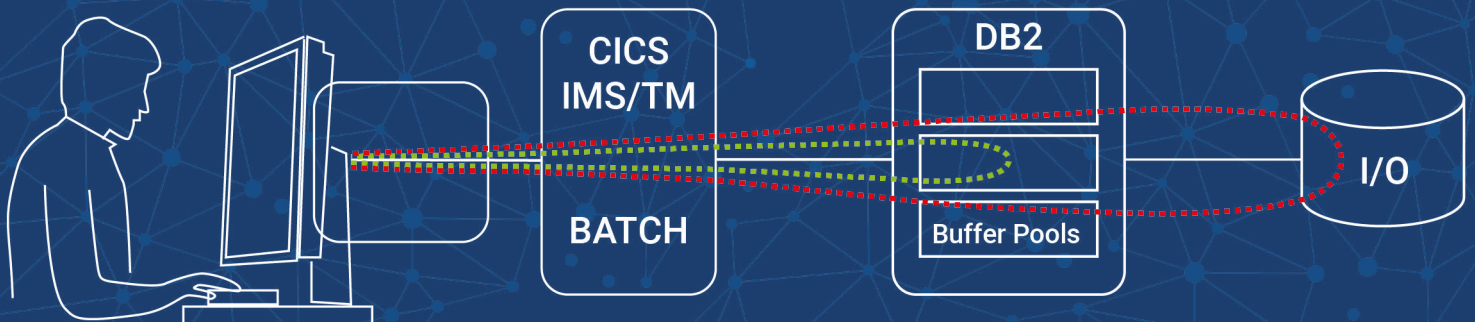
Since it makes no sense to cache a result set that won't be requested again, QuickSelect identifies and caches only those result sets meeting a 'must be requested x times before caching' threshold parameter. QuickSelect sits between your applications and Db2 and observes the Db2 workload for queries whose results are in cache as

well as for queries that have just met the frequency threshold.

- For query results that are not found in QuickSelect cache, those queries are executed as usual by Db2.
- If a query has just met the execution frequency threshold, Db2 will, execute the query and return the result set to the application. QuickSelect will at the same time store the result set in its cache.
- For result sets already in cache, QuickSelect serves these directly to the application, saving both CPU and potential I/O, thus improving online application response times and batch job performance.

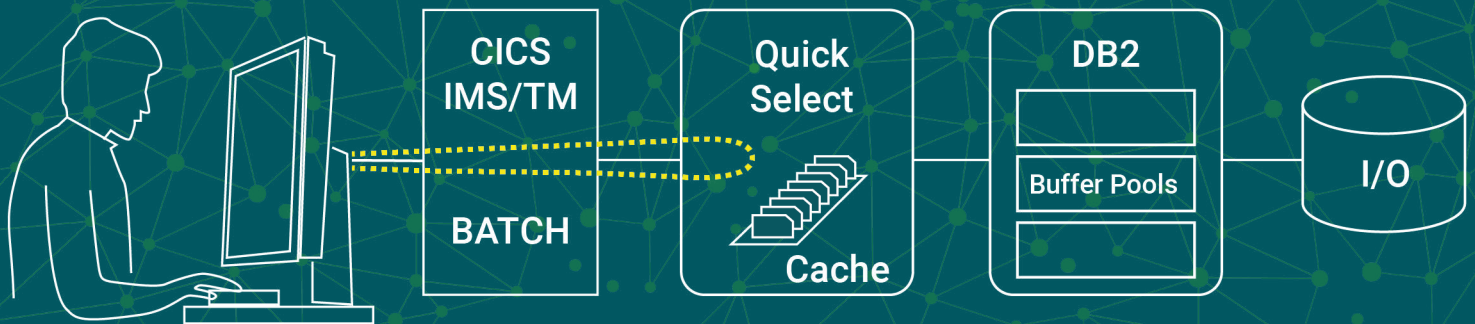
The following diagrams illustrate data flow without and then with QuickSelect. Without QuickSelect, all queries go directly to Db2. Db2 generates the results on the fly from pages already in buffer pools or if needed called from disk.

### Data Flow Without QuickSelect



Let's add QuickSelect to the picture. Results for eligible, frequently executed queries are stored in QuickSelect cache. When QuickSelect intercepts a query whose result set has been cached, QuickSelect very quickly returns it to the application, bypassing Db2 and its resource-intensive computations. It is important to note that QuickSelect is not a DBMS and performs no data processing. It does not cache entire tables or pages of data. It simply caches and returns frequently requested result sets as previously generated by Db2. QuickSelect's cache memory requirements are modest, generally no more than a few GB of above-the-bar memory per Db2 subsystem.

## Data Flow With QuickSelect



Think of all the queries that are repetitively running in your shop every day – and of all the redundant Db2 processing and I/O involved in repeatedly generating the exact same result sets.

This raises a question: Which queries should be cached?

## Survey Mode

QuickSelect Survey mode automatically identifies cache-eligible queries. These will be read-only static SELECT and DECLARE CURSOR statements. Remember that a query is the combination of a specific SQL statement and specific host variables. An SQL statement might be executed over time with millions of different host variables. Each execution of a query with different host variables generates a unique result set. One SQL statement can be the basis for millions of unique queries and corresponding result sets, with many repetitions over time. QuickSelect Survey profiles your Db2 SQL workload, identifying those cache-eligible SQL statements where the majority of queries are repetitively run. These are the high ROI SQL statements. At this point, the only additional knowledge required is the DBA's assessment of the relative stability of the tables underlying these cache-eligible SQL statements. These tables should be slow-changing during most, but not necessarily all, of the day; this avoids over-frequent cache invalidation as described below. The administrator then adds the names of suitable tables to a QuickSelect list, and QuickSelect starts caching the associated query results when placed in Save mode. No changes to the environment of any kind are required.

## Update Sensitivity

The next important question is: What happens if data in a Db2 table that underlies cached result sets changes? Even relatively stable tables experience change from time to time. If QuickSelect's cache contains result sets generated from a table that has just changed, the cached result sets may no longer be valid.

QuickSelect is sensitive to these table updates.

It is aware of all data changes, including those made by Db2 utilities like Load, Reorg, and Recover, and by applications issuing Insert, Update, Delete, Merge, and Truncate statements. Imminent table updates are detected by QuickSelect in real time. QuickSelect then immediately invalidates the dependent cached result sets, thus preserving data integrity. QuickSelect, at the same time, routes requests for the just-invalidated result sets directly to Db2 for execution. As repetitive requests for the just-invalidated result sets are identified by QuickSelect, those result sets are again added to cache and, in this fashion, the cache is quickly and dynamically rebuilt. QuickSelect Update Sensitivity is fully supported across Db2 data sharing and multiple-LPAR environments using XCF functions.

In other words, QuickSelect always returns the same answer as Db2 does... only faster.

Furthermore, QuickSelect delivers plug-and-play performance gains and CPU savings. It does not require any risky, time-consuming modifications of code, JCL, or database structures. Zero changes are required. No application testing is required. You don't even have to Rebind your programs to take advantage of QuickSelect.

Are you using the IBM Db2 Analytics Accelerator? QuickSelect and IDAA are fully compatible and play complementary roles in improving the performance of your Db2 workload. QuickSelect intercepts and caches result sets for frequently executed, static SQL. The remainder of the Db2 workload is passed to the Db2 Optimizer, which then passes IDAA-eligible SQL to IDAA for execution. There are no dependencies between the two solutions.

Finally, QuickSelect is non-intrusive. You can toggle QuickSelect on/off in your production

environment in real-time with zero impact on your operations other than a corresponding gain/loss in performance. When toggled off, all query processing reverts to native Db2 data access. When toggled on, QuickSelect dynamically rebuilds its cache of result sets and resumes delivering its benefits.

### **Key Features:**

- In-Memory Query Result Set Caching
- Zero-Change Implementation
- Automatic Cache Management
- Real-Time Update Sensitivity
- Minimal Memory Requirements
- Non-Intrusive Operation
- Customizable Caching Policies
- Highly Scalable

### **Key Benefits:**

1. Enhanced Online Application Responsiveness
2. Faster Batch Processing
3. Reduced Db2 Resource Consumption
4. Reduced I/O Operations
5. Immediate ROI
6. No Big Project; No Big Learning Curve
7. Set Up and 'Forget'

## **What Results Can You Expect with QuickSelect?**

QuickSelect can improve performance, save CPU, and reduce I/O operations for programs that repetitively request the same results. Benchmarks comparing QuickSelect to Db2 accessing data in buffer pools show up to a hundredfold increase in the speed at which SQL results are returned to the requesting program and up to an 85% reduction in CPU for that portion of the Db2 workload. In addition, QuickSelect frees up Db2 resources that can be more profitably used to service other requests, and reduces CICS Db2-related TCB switching for across-the-board improvements in CICS transaction performance.

One QuickSelect user, a large European commercial bank, was able to turn off an entire CPU after implementing QuickSelect. In this case, the customer saved more than 10 billion Db2 executions during a typical 12-hour window, executions replaced by rapid retrieval of result sets from QuickSelect cache.

Another customer, a large commercial bank running Db2 across a CICSplex, implemented QuickSelect for a 24% average reduction in CICS transaction response time and a 16% reduction in total CPU consumption.

Deploying QuickSelect can immediately improve online response times and reduce batch job elapsed times. QuickSelect delivers rapid ROI by virtue of its plug-and-play, no-changes-required implementation.



## The Bottom Line

QuickSelect for Db2 offers a novel plug-and-play approach to in-memory data management by caching query results. QuickSelect for Db2 significantly improves the performance of Db2-reliant online and batch programs while reducing CPU both on average and during peaks. Whether you subscribe to Rolling 4-Hour Average or Tailored Fit Pricing, QuickSelect can contribute to substantial savings in software and hardware costs while improving business performance.

The bottom line: In-memory processing can save CPU, improve performance, and save money. Wise organizations will look into multiple ways of exploiting memory to achieve results, including potentially looking at novel software solutions like [QuickSelect for Db2](#).

### About Craig Mullins

Craig S. Mullins is the president and principal consultant of Mullins Consulting, Inc., an independent consulting and strategy firm specializing in database management and mainframe systems. Craig has been involved with Db2 for z/OS since Version 1 and has extensive experience as an application developer, DBA, and instructor. He has been recognized as an IBM Gold Consultant and IBM Champion for Data and AI by IBM, and as an Influential Mainframer by Planet Mainframe. Craig is the author of several books, including the industry-leading “DB2 Developer’s Guide” on Db2 for z/OS and “Database Administration: The Complete Guide to DBA Practices and Procedures,” the only book on heterogeneous database administration. Craig can be reached at <http://www.MullinsConsulting.com>

Additional information on QuickSelect for Db2 can be found at <https://log-on.com/quickselect-for-db2-performance/> or by emailing to [ask@log-on.com](mailto:ask@log-on.com)