Sybase IQ Multiplex – Designed For Analytics

Roger MacNicol Blaine French

Sybase Inc
561 Virginia Road
Concord, MA 01742

rdm@sybase.com blainef@sybase.com

Abstract

The internal design of database systems has traditionally given primacy to the needs of transactional data. A radical re-evaluation of the internal design giving primacy to the needs of complex analytics shows clear benefits in large databases for both single servers and in multi-node shared-disk grid computing. This design supports the trend to keep more years of more finely grained data online by ameliorating the data explosion problem.

1. Background

Historically, databases have been designed around the requirement to support large numbers of small concurrent updates. The primary design criterion has been to minimize the portion of stored data that must be locked for exclusive access and the length of time that locks are held. Consequently, when the writer of data has primacy, the internal design generally adheres to the following rules

1. Data should be stored in rows to enable a single disk i/o to write the modified data out to the table.
2. Data should be stored on small page sizes to minimize the i/o cost and portion of disk locked for exclusive access.
3. The database must support the imperfect notion of isolation levels to enable reports to run in acceptable time while negotiating held locks.
4. Few columns should be indexed because locks on tree structures may deny access to more rows than row-page locks and increase the time locks are held.
5. Data pages should typically not be compressed because of poor amortization of the compression cost. Rows typically do not compress well because of the mix of data types stored adjacently.
6. Adding or dropping a column or index may be expensive since page storage may need to be updated for all rows.
7. Searched update statements may be relatively expensive since the entire row must be read and written for a single column to be updated.

Given these consequences, IQ's designers started by asking one fundamental question "What would a database look like internally if it were designed from the ground up for complex analytics on massive amounts of data"? It was understood that a reader-friendly analytics server must also support many concurrent update streams and, increasingly, it would be an operational data store rather than a store for cleansed and summarized data. But, data update streams in such a server are likely to be well-bounded. Once the primary criterion for the internal design becomes the performance of complex analytics, rather than row-oriented updates, a radical *volte-face* in design becomes self-obvious.

2. Designed for analytics

Typical analytical queries access relatively few columns of the storage-dominating fact tables and may access a notable proportion of the rows stored in the fact tables. While CPU performance and available cache memory has increased dramatically with 64-bit servers and lower memory prices, disk performance has not kept up and, consequently, disk-bound performance is typical for many analytics. So, were design primacy to be given to complex analytics, many of the previous rules are reversed as follows:

**Proceedings of the 30th VLDB Conference,
Toronto, Canada, 2004**

1. Data would be stored in columns, not rows, so only the columns required to answer a query need be read – in effect every possible ad-hoc query could have performance akin to a covering index. Since data is only written once but read many times any increase in load time would be more than offset by improved query performance. Column storage would enhance main cache efficiency since commonly used columns would be more likely to be cached.
2. Data would be stored in a large page size so that many cells of a column can be retrieved in a single read and the larger page size plays to technological changes in how physical disk reads are structured. Traditional DBMS cannot use large pages sizes since each read drags along unneeded columns in the row.
3. The database would use page-level snapshot versioning so that data modification happens without interfering with running reports. Consequently, there would be no need for the notion of isolation levels – every query would see an internally consistent database state while navigating a nearly lockless environment.
4. Every column could be indexed: data will be written once and read many times so the cost will be amortized. Column storage at the physical layer greatly simplifies parallel index updates and adding an index requires only that column to be read, not the entire row.
5. Data could be compressed on disk. The compression cost would be well amortized and the homogenous datatype of a stored column would offer optimal compression.
6. Adding or dropping a column or index to reflect changing business requirements would be cheap, as no other data would be accessed.
7. Searched updates would be relatively cheap since only the columns being modified would need to be read and written.

From these design principles, IQ was developed from the ground up to be a 64-bit DBMS using kernel threads giving design primacy to the needs of complex analytics without compromising load performance.

3. Designed for scalability (Multiplex)

Sybase IQ optimizes workloads across multiple servers through Sybase IQ-M, which is a multi-node, shared-storage, parallel database system whose design focus is on large-scale data warehousing workloads. IQ-M offers grid-computing through multiple IQ instances running on multiple server nodes connected to a single shared IQ data store, which can comprise multiple disk arrays. Each node (instance) sees the entire database and has direct physical access to it, unlike the horizontally partitioned databases (such as MPP (massively parallel processors) with shared-nothing architecture). There are two types of IQ-M nodes, writer (one per Multiplex) and reader (all other nodes in the Multiplex).

1. The Writer node owns all database locks, performs DBA tasks, and updates the database. The Writer is tuned for data loading efficiency and speed.
2. Reader nodes have read only access to the shared database and perform tasks such as ad-hoc queries and reporting. The reader access data without the need for acquiring locks or for a distributed lock manager.

There is no node-to-node interference since a single query is limited to a single node. The use of snapshot versioning requires minimal communication between the IQ-M nodes, consequently there is no requirement for an expensive high-speed interconnect between nodes.

The design offers a better approach to analytics scalability. Adding additional applications and users to a database can be solved simply by adding additional cheaper reader nodes. The impact of additional nodes on existing nodes is minimal until the disk array is saturated. The non-traditional storage model offered by IQ's design enables more user requests to be satisfied before for a given disk configuration is saturated than traditional DBMS clusters since each node typically accesses fewer pages to satisfy each query.

4. The role of compression

Vertical storage enables IQ to use a multi-tiered compression strategy:

1. Domains that have less than 65,535 distinct values are stored in enumerated form. A column store can use enumeration directly as the stored form rather than in an data structure.
2. Each column is written to disk using a compression algorithm specifically tailored to the patterns typical of its datatype.
3. Index record lists are stored in compressed form as segmented bitmaps. Bitmap index pages utilize a range of internal representations depending on the density of bits in any given segment.

A consistent emphasis on data compression has advantages. **First**, it requires less disk space and a smaller storage system footprint than other DBMS, reducing both system purchase and administration costs. **Second**, the required disk I/O bandwidth is typically reduced by over 90% allowing better CPU utilization. Sybase IQ systems are typically CPU bound unlike conventional DBMS, which are typically I/O bound. **Third**, as a result of using a large I/O request size, Sybase IQ systems can exploit

denser disk configurations using fewer larger disks than other DBMS without sacrificing performance. **Fourth**, because Sybase IQ reads only the columns referenced, it has a smaller memory footprint.

Businesses and government regulations are requiring: more data to be kept online; that it be kept highly available; and for that data to be more finely grained. Increasingly, a DW must keep operational data and not summarized data to meet reporting requirements. A column store enables more operational data to be kept online on a given configuration through the combination of enumerated storage and better compression thus ameliorating the inherent data explosion problem.

5. Query efficiency

The combination of column storage and compression enables every column in a Sybase IQ database to be indexed, often with more than one index type. For a column store, the results of the Where clause needs to be a mask of rows to be projected that can be applied to each column and virtual rows created as cells are read in parallel from each column and combined. This mask takes the form of a bitmap termed the "foundset".

The IQ query engine aims to make maximal use of available indices and push all sensible predicates (including predicates on functions and expressions) into column indexes in parallel. The result of each predicate is a bitmap of rows. The resulting bitmaps are combined through Boolean operators to solve the Where clause. Having bitmap indexes on all columns means that IQ does not need or support "update statistics" – meta-data required by the optimizer is obtained directly from the indexes and, by definition, is never stale.

6. Sizing rules

Sizing rules for IQ: I/O efficiency and comparison with typical RDBMS

DBsize/raw data size:	0.4-0.9 vs 2-8
Storage:	RAID_5_ovhd B vs RAID_1_ovhd
DB page size:	256K-512K vs 2K-32K
IO Bandwidth per CPU:	8 MB/s vs 10-200
Typical io/sec per CPU:	<100 vs >1000
Optimal disk size:	100GB+ vs 18-36GB

7. Results on Multi-Node system

A recent benchmark on a Sun Fire 6800 (24 750MHz US-III CPUs, 48GB RAM) of the current shipping version of IQ Multiplex showed 64-bit addressing by using 40 GB RAM for the IQ cache. Storage was on Sun StorEdge T3 arrays. 253 million rows were loaded in 1h35min for a loading speed of 2.6 million rows/min or 160 million rows/hour. 79 GB of raw input data was compressed to 33 GB (41% of its raw size).

Several queries were run first on a single domain, and then on four multiplex domains simultaneously against a single IQ-M data store. Each query test consisted of running 10 queries concurrently on one domain. As more domains were added to the Multiplex to run additional query streams, the overall query response times stayed the same (one would have expected degradation in performance as more query streams were being run against the same database on the same shared storage), and demonstrated excellent scalability of the described design. The following table gives time in seconds for a single domain and the range of times for four domains:

Query	Single domain	4 Multiplex domains
1a	2434	2372 - 2419
1b	2705	2660 - 2811
2a	1992	1936 - 1978
2b	2324	2446 - 2305
3a	2853	2806 - 2849
3b	2326	2287 - 2446
4a	731	665 - 693
4b	725	662 – 690
5a	672	797 - 832
5b	692	684 - 778

8. Summary

Sybase IQ-M demonstrates that complex analytics on very large databases benefit from a radical re-evaluation of many aspects of internal database design. By basing the primary design assumptions on the nature of analytic workloads, the combination of column storage, bitmaps, aggressive compression, and shared disk grid-computing through IQ Multiplex offers significant performance benefits for multi-terabyte databases.