

DB / IQ - QA - Automated Quality Assurance for DB2 SQL

Domination of SQL and DB2

Since DB2's introduction in 1983, it has become the dominant DBMS on mainframes running under z/OS. Performance issues increase as more applications use SQL and more data is DB2 managed.

DB2 applications cover the entire spectrum. SQL in batch accessing millions of rows and online transactions needing to execute in fractions of a second. IBM has steadily added flexibility to SQL and performance enhancements to support the challenge. *The result is a very rich system which is easy to use.... but difficult to use consistently well.*

Access to DB2 is easy, understanding internals is not. Minor changes in the SQL may increase resource utilization by magnitudes. Production versions containing "time bombs", often result in performance issues.

The importance of standards

Fixing a problem in production is far more expensive than fixing it during development. Many organizations have therefore committed themselves to submit only high quality code into production. Typically, trying to minimize DB2 performance and reliability problems by using SQL coding standards, avoiding the common pitfalls. To be effective standards must be continuously monitored to ensure consistency.

Only the most experienced DBAs and application programmers have the skills to monitor new code manually to see if it meets standards. Few organizations can afford to have all new code checked daily - often amounting to hundreds of statements. Applying new standards manually to thousands of existing SQL statements is almost impossible.

Frequently, quality control is limited to the investigation of production SQL which has failed or which has major performance problems. Such work can be very frustrating to specialists who prefer to work on new database designs or on tuning databases **before** problems occur.

The best way to monitor SQL standards continuously and automatically is with DB/IQ QA

Sample rules

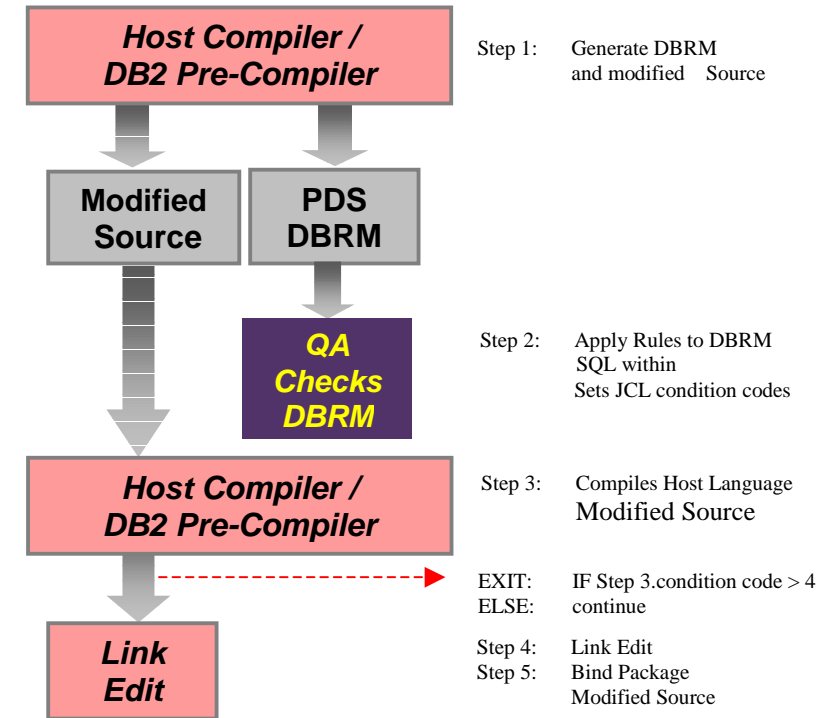
Currently, more than 200 different rules may be applied. Many with qualitative and quantitative parameters to provide the flexibility to accommodate different size databases. Individual combinations can be saved in named sets where each rule can be defined with a severity level when violated - to just inform, to issue a warning, or to set a condition code that will cause rejection of subsequent steps such as linking or binding. Some examples: -

- Enforcing index access
- Restricting the number of joins, sorts, UNIONS, or sub-selects
- Documenting Index, Trigger and Referential Integrity maintenance involved with DML
- Rejecting violations of standards such as the use of object qualifiers, superfluous cursors and objects, view materializations, Cartesian Products, or DELETES or UPDATES without WHERE predicates
- Rejecting programming violations such as the use of DDL, SELECT *, arithmetic in predicates, host variables with attributes that differ from the predicate column,...and many more.

How is DB/IQ QA applied

The goal of Quality Assurance is to maximize quality with minimum disruption. DB/IQ QA establishes this by automatically checking the SQL at all stages of application development. Checks will also analyze the access paths and provide detailed explanations.

- QA against existing applications.** Check existing SQL to uncover any ticking time bombs. DBRMs, Views, Plans or Packages including their static and dynamic SQL can be checked. Typically, hundreds of rule violations will be identified. Uncovering just one major problem before it occurs can justify DB/IQ QA.
- QA for ERP and other dynamic applications.** Analyze all SQL, including the dynamic SQL of ERP packages. Using the Trace and Extract Monitor, QA captures all actual resource usage - CPU time, elapsed time, page activity etc., thus enhancing its standard checking facilities to adapt to the rapidly growing and changing application environment.
- QA while programs are being developed.** The earlier a problem is detected the lower the cost of correcting it. DB/IQ QA provides an intelligent ISPF edit macro which allows SQL statements to be evaluated as programs are being developed. Simply "mark" the code to be evaluated and invoke DB/IQ QA.
- QA as programs are compiled.** Use DB/IQ QA as an early warning tool or as a filter to ensure that no sub-standard SQL enters production. Standard compile jobs can be modified with the insertion of a DB/IQ QA step to evaluate all the SQL in the program. Subsequent Link edit and BIND steps can be flushed subject to condition codes set by DB/IQ QA rule violations.



The efficient but illuminating Explain Facility

DB2 explains all SQL "under the covers" in order to build an access strategy. QA's Explain Facility shows exactly how DB2 will process the SQL in easy-to-read output together with all relevant Catalog statistics. The following example based on the Catalog table SYSCOLUMNS locates columns with a column length greater than the average column length for its column type. The SYSCOLUMNS table had 20992 rows.

```
SELECT TBCREATOR,TBNAME,NAME,COLTYPE,LENGTH FROM SYSIBM.SYSCOLUMNS OT
WHERE OT.LENGTH > (SELECT AVG(LENGTH) FROM SYSIBM.SYSCOLUMNS IT
WHERE IT.COLTYPE=OT.COLTYPE)
```

■ *QA found the cost factor (DB2 timeron) equal to 221848320 and produced the following EXPLAIN.*

The "correlated subselect" must re-execute the internal sub select for each row in the OT table (outer table), even when the column type is repeated, in order to satisfy the ">" operator. Drilling down with the Explain Facility we discovered that Step-1/01 was executed 20992 times with a cost factor = **221837136**.

■ *The newer syntax below reduced the cost factor to 1799828 – a reduction of 92 percent!!*

```
SELECT TBCREATOR,TBNAME,NAME,C.COLTYPE,LENGTH FROM SYSIBM.SYSCOLUMNS C INNER JOIN (SELECT COLTYPE,AVG(LENGTH) AS ALEN FROM SYSIBM.SYSCOLUMNS GROUP BY
COLTYPE ) AS TEMP ON TEMP.COLTYPE=C.COLTYPE WHERE C.LENGTH > TEMP.ALEN
```

/Step	Method	Object - Creator / Name	FN	SORTs	Prefetch	Access in step
1/01	0,first table accessed TABLE stats: TSpace stats	SYSIBM SYSCOLUMNS(01) NPAGES 2107 CARDF 20992 PARTITIONS 0 SEGSIZE 0 NACTIVE 4860		No	Seq	Table space scan
2/01	0,first table Accessed TABLE stats: TSpace stats	SYSIBM SYSCOLUMNS(02) NPAGES 2107 CARDF 20992 PARTITIONS 0 SEGSIZE 0 NACTIVE 4860	R	No	Seq	Table space scan Correlated

Track changes in quality using the History Database

DB2 may execute the same SQL differently, this could be for many reasons. One example is rebinding a package after statistics have changed since the prior bind. Monitoring such differences can uncover problems before they occur, DB/IQ QA facilitates such comparisons. You can even compare future impact by simulating higher RUNSTATS.

The DB/IQ audit facility provides statistics on rule evaluations, which can be Analyzed to see how SQL quality overall, or just within a certain application, has changed over time. It is not rare to see the frequency of problems uncovered by DB/IQ cut by 50% in the first six months of use.

DB/IQ QA maximizes the benefits of your QA efforts

DB/IQ QA is a software product that automates Quality Assurance on SQL - applying rules tailored for your own site and applications. It can check large volumes of existent code (Dynamic and Static SQL) as well as evaluate new code dynamically, as it is written.

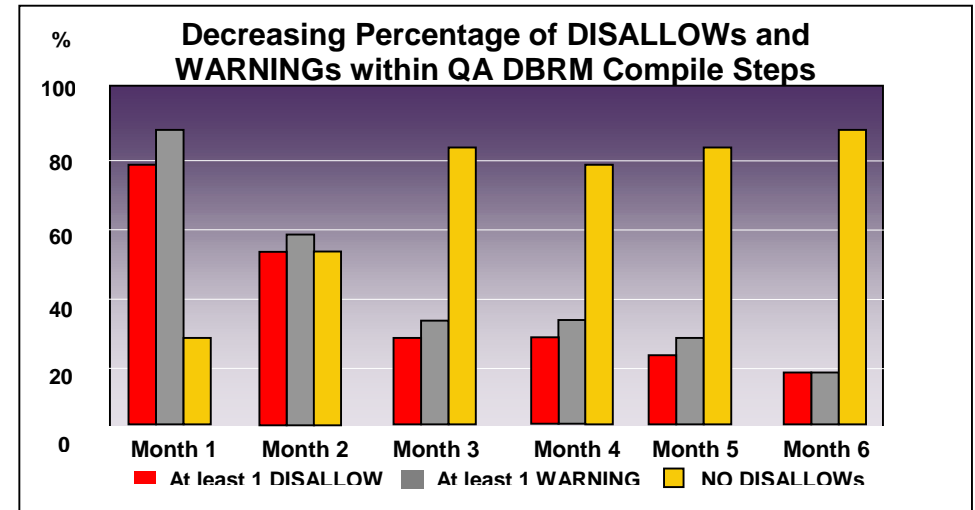
DB/IQ QA is easy to use. Comprehensive, well-structured screens prompt organizations to set their own standards for “clean” and “efficient” SQL - avoiding pitfalls which, as applications grow, choke the system through excessive resource utilization.

DB/IQ QA includes a Trace and Extract Monitor, which enhances QA functions by capturing all SQL for one or more applications.

- Runs as a stand-alone Batch job utilizing the standard DB2 Instrumentation Facility
- Can define the Trace Scope (filters, applications etc.) and duration time or conditions.
- Captures all SQL, including Dynamic, essential when tuning ERP applications such as SAP®, PeopleSoft®, Siebel® and other C/S applications. Captured data includes CPU and Elapsed time, execution frequencies, DB2 Row Level and GETPAGE statistics.
- Navigation Utilities guide the user to view the results found by the Extract Utility, yielding important information on expensive SQL statements.

Frequency Maintenance Utilities - set traced and predicted Execution Frequency values for SQL statements and for Plans and Packages, thus estimating future workloads.

DB/IQ QA takes a pro-active approach to quality control. It detects SQL coding errors or standards violations - and requires their correction *before* applications are moved into production. QA rules can be applied to SQL syntax; access paths; and even to cost factors provided by the DB2 optimizer. The QA Explain facility provides a comprehensive explanation of how DB2 will access the data, the statistics that influence its choice, and areas where performance and reliability could be improved. Tuning facilities anticipate production level data volumes and predicted execution frequency to forecast the application’s behavior and access paths most efficient. Regular use of DB/IQ QA should quickly help raise the quality of new code and save you significant amounts of both computer and human resources. An audit facility tracks the improvement in quality over time.



Other DB/IQ Utilities

QA is the centerpiece of DB/IQ but several other optional and easy-to-use utilities are also available.

DB/IQ WL+ is an add-on to QA. The "Workload Detector" is based on QA's SQL Monitor Trace facility. WL+ will analyze the extracted trace data and present results and / or reports to identify the most "costly" plans, packages and SQL statements, taking the execution frequency into account.

Workload checks are threshold-driven and can be set to alert when certain pre-defined situations occur. WL+ provides multi-dimensional views of the workload by plan, package, table and SQL giving all information required by application developers and DBAs to improve application performance.

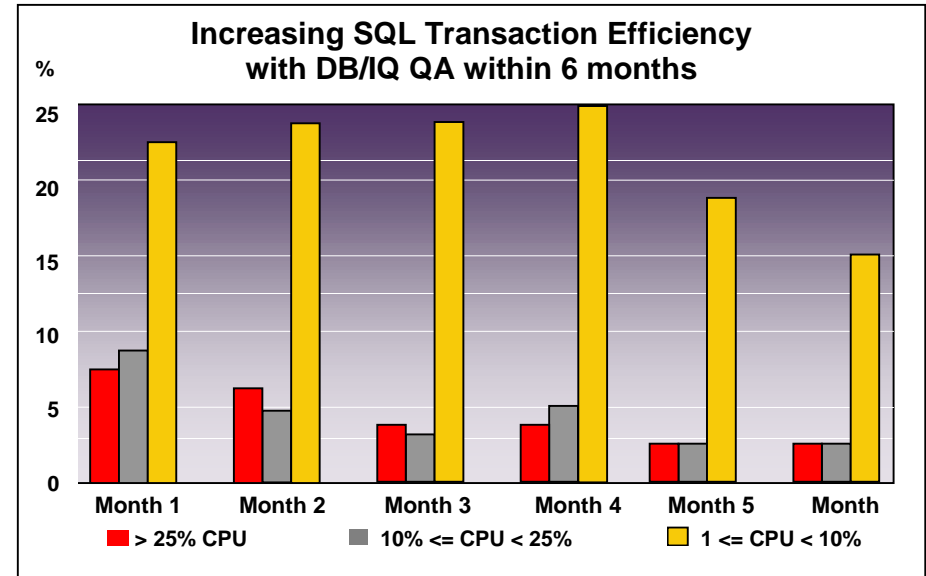
DB/IQ IA+ is an add-on to QA. The "Index Administrator" analyzes and ensures optimal indexes. It composes of a Repository database and facilities to perform index analysis, index advice, index design and index impact analysis. The Repository includes a unique cross-reference facility between plans, programs, tables, views, indexes and columns; originating from both static and dynamic SQL. It is able to drill down column usage as far as indicating when used as a predicate or in sargable or indexable functions - information not available in the DB2 Catalog. Future workloads can be predicted with execution frequency values for SQL statements and for plans and packages. IA's impact analysis will consider all SQL executed on a certain table, whether static or dynamic and executed just once a year or 1000 times a minute. Reports include cost- and workload-based evaluations, impact on applications affected, TS scans, internal sorts performed, non-used indexes ... and more.

DB/IQ MA+ is a further add-on to QA. The "Migration Aid and other Utilities" is a selection of tools to re-engineer DDL, automatically generate threshold-driven REORG jobs, maintain package versioning (FREE non-required versions) and more... MA+ provides all of the information required by application developers and DBAs to improve database administration.

DB2, OS/390, z/OS and ISPF are registered trademarks of IBM Corporation. All other company and product names may be trademarks or registered trademarks of their respective owners:



InSoft Software GmbH, Derendorfer Str. 70, 40479 Düsseldorf, Germany. Tel: +49-211-44 03 16 6, Fax: +49-211-48 80 33, www.insoft-software.de



DB/IQ QA Benefits

DB/IQ will typically uncover problems on the first day it is used, and will continue providing benefits for many years to come. For example, DB/IQ QA will: -

- Detect reliability or performance problems before they occur in production – resulting in major savings in machine, programmer, DBA, and user resources and cost.
- Validate Static / Dynamic SQL in ERP, purchased or "in-house" applications
- Automatically evaluate existing SQL against new standards.
- Check new code automatically before moving it into production.
- Track coding quality over time by application, development team or site.
- Estimate future performance using data volumes that anticipate growth and execution frequency values predicting future usage.
- Free up DBAs from routine QA for more beneficial design and optimization tasks.