DB/IQ QA
Automated Quality Assurance for DB2 SQL

Domination of SQL and DB2

Since their introduction in 1983, IBM's DB2 and SQL have become the dominant DBMS and data management language respectively on mainframes running under OS/390. Performance increases in importance every year as more applications use SQL, and more data is managed by DB2.

DB2 based applications cover the entire spectrum. Batch applications using SQL access millions of records, while some online transactions are used by hundreds of users concurrently and need to execute in fractions of a second. IBM has supported the challenge created by such wide usage by steadily adding flexibility to SQL and performance enhancing features to DB2. *The result is a very rich system which is easy to use....but difficult to use consistently well.*

The non-procedural nature of SQL makes it easy to access DB2 data but difficult to understand its internal operation. Apparently slight changes in the SQL can increase resource utilization by several magnitudes. Production code often contains "time bombs" which result in future application failure or increasingly severe performance problems.

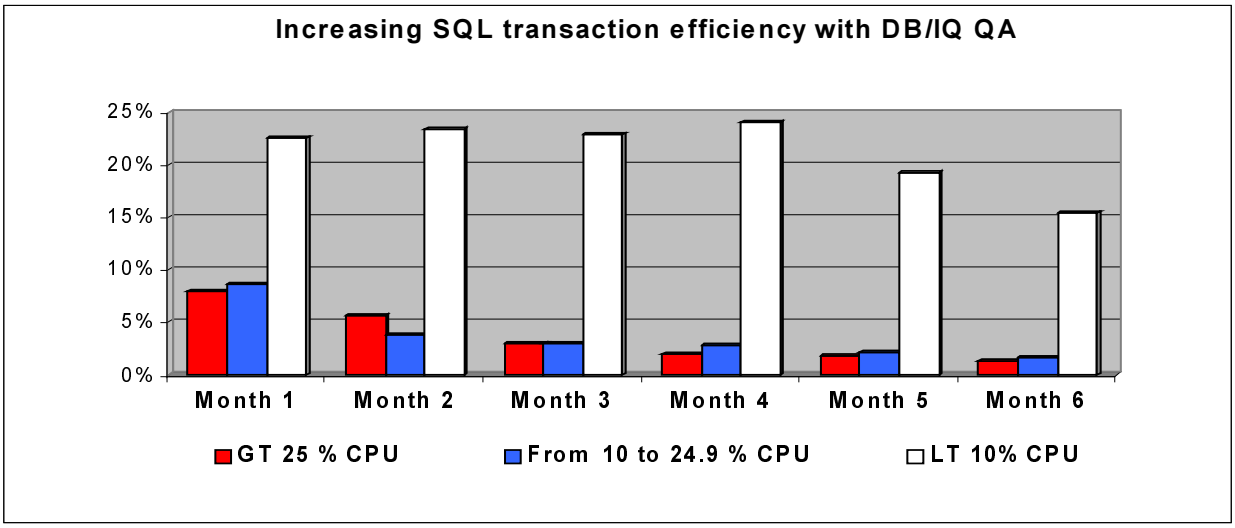
The importance of standards

Fixing a problem that occurs in production can cost hundreds of times more than fixing it during development. Consequently, many organizations have made a commitment to submit only high quality code into production. Typically, this involves trying to minimize DB2 performance and reliability problems by using SQL coding standards which avoid the most common pitfalls. To be effective, however, standards must be continuously monitored to ensure consistent application. Many organizations have discovered during their Y2K project that "standards" are the exception rather than the rule.

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 them check all the new code produced 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 unexpectedly fails 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.



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 code in existing libraries as well as evaluate new code dynamically, as it is written.

DB/IQ QA is very easy to use. Comprehensive, well-structured ISPF screens prompt DBAs to set the organizations own standards for "clean" SQL (easy to understand and maintain) and "efficient" SQL (avoiding pitfalls that, as applications grow, could choke the system through excessive resource utilization).

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 execute the code, the statistics that influence its choice, and areas where performance and reliability could be improved. Tuning facilities anticipate production level data volumes to forecast the application's behavior and select the access paths which will be most efficient for those volumes. 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.

Sample rules

Currently, more than 160 different rules may be applied. Many are qualitative. Others have quantitative parameters to provide the flexibility to accommodate different size databases, types of transaction, and different groups of users. Different combinations can be saved in named sets where each rule can be defined with a severity level when violated - just to provide information, to issue a warning, or to set a condition code that will cause rejection of subsequent steps such as compilation, binding or loading into production libraries. Some examples:-

- Enforcing index access
- Restricting the number of joins, sorts, UNIONs, or sub-selects
- Documenting index 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 *, scalar functions or arithmetic in predicates, program variables with lengths that differ from database variables,...and many more.

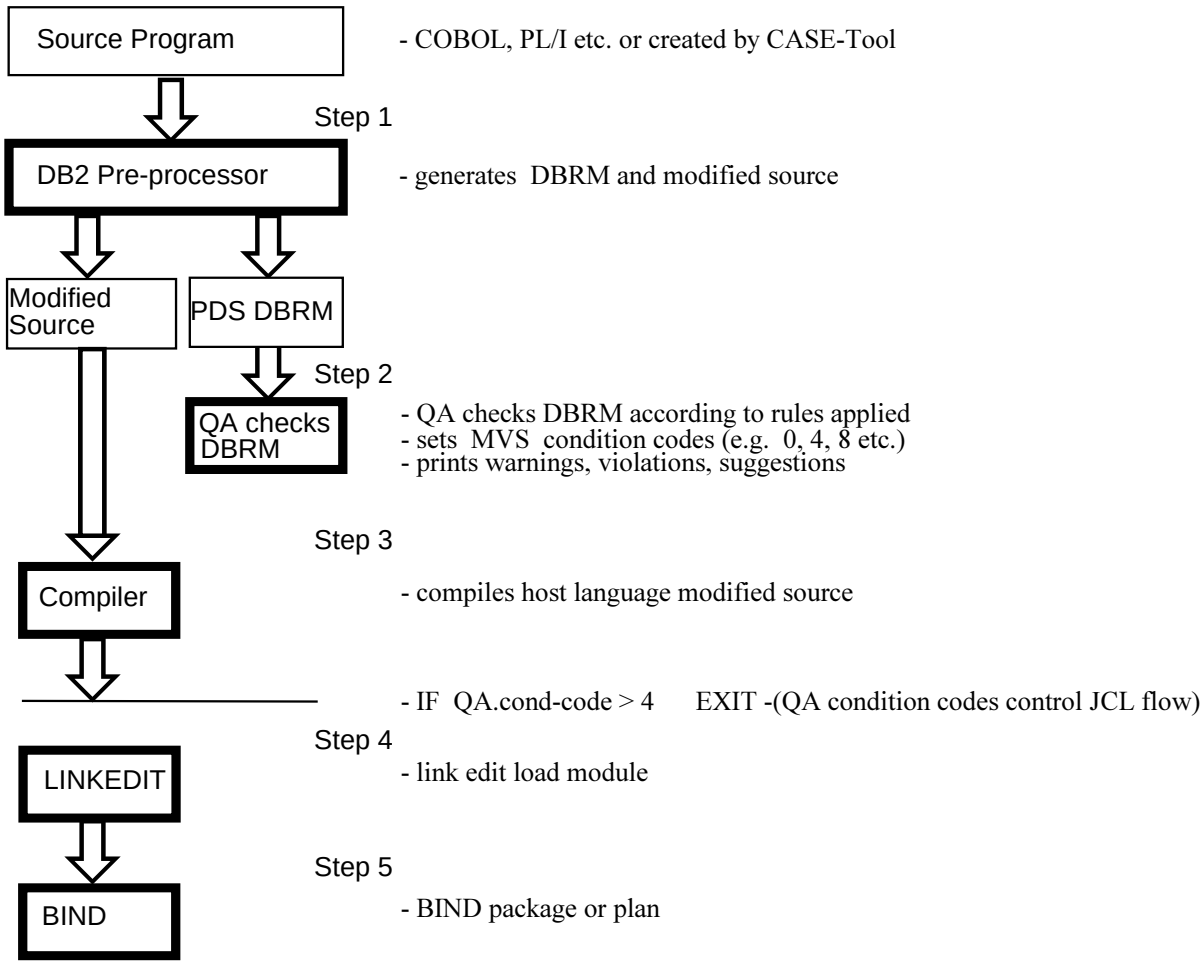
How is DB/IQ QA applied?

The goal of Quality Assurance is to maximize quality with minimum disruption. DB/IQ QA meets this goal by automatically checking the SQL at every stage of application development. All checks analyze the access paths and provide a detailed explanation of how DB2 will process the statement.

1. **QA against existing applications.** The first application of DB/IQ QA.is usually to check the existing inventory of SQL to uncover any ticking time bombs. Any combination of DBRMs, Views, Plans or Packages can be checked in batch. Typically, hundreds of rule exceptions will be identified. Uncovering just one major problem before it occurs can justify DB/IQ QA.

2. **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 editor which allows SQL statements to be evaluated as programs are being developed. Simply "mark" the code to be evaluated and press the appropriate PF Key to invoke the DB/IQ QA Explain facility or to get a cost estimate of executing the code. Corrections or improvements can be applied immediately.

3. **QA as programs are compiled.** DB/IQ QA can be used 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 are flushed *subject to condition codes* set by DB/IQ QA rule violations.



The efficient but illuminating Explain Facility

SQL, like all non-procedural languages, does a lot of work “under the covers” that the programmer does not see. This undercover work is critical to the performance of the code. Without understanding what takes place, the programmer has a difficult time recognizing problems and making improvements. Should the messages generated by the rules be insufficient to clarify the issues, The Explain Facility provides additional detail to clarify the problem.

Whether the SQL is evaluated interactively or in batch, the Explain Facility describes for the programmer exactly how DB2 has decided to process each SQL statement. It details what indexes (if any) have been used, when resource consuming table scans, table space scans, sorts, joins etc. have been performed, and summarizes with the cost factor of the statement. In addition, it provides output from the rules applied - either informational, warnings, or disallows as requested.

As this Explain Facility is provided for application developers as well as DBAs, high performance and clarity of output are essential. The output includes all relevant catalog statistics which are vital for performance analysis. These are retrieved from DB/IQ’s internal repository which is normally updated at night in batch to relieve online access to the DB2 catalog.

Let us consider the following example based on the DB2 Catalog table SYSCOLUMNS. It tries to locate columns with a column length greater than the average column length for this 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 costfactor (DB2 timeron) equal to 221848320 and produced the following EXPLAIN.

Block /Step	Method	Object Creator / Name	Col FN	Extra SORTs	Lock used	Pref etch	Access used in step
1/01	0,first table accessed	SYSIBM SYSCOLUMNS		No	IS	Seq	Table space scan
	TABLE stats :	NPAGES 2107	CARD 20992				
	TSpace stats:	PARTITIONS 0	SEGSIZE 0	NACTIVE 4860			
2/01	0,first table accessed	SYSIBM SYSCOLUMNS	R	No	IS	Seq	Table space scan
	TABLE stats :	NPAGES 2107	CARD 20992				
	TSpace stats:	PARTITIONS 0	SEGSIZE 0	NACTIVE 4860			

The major problem with the *"correlated subselect"* is, that for each row found in the OT table (OT=outer table), DB2 must re-execute the internal sub select, in order to compare the ">" operator. This is done even when the column type is repeated. Drilling down using the Explain Facility we discovered that Block/Step-1/01 was indeed executed 20992 times with a cost factor equal to **221837136.**

The DB2 4.1 syntax below reduced the cost factor to 1799828 – a reduction of 92 percent!!

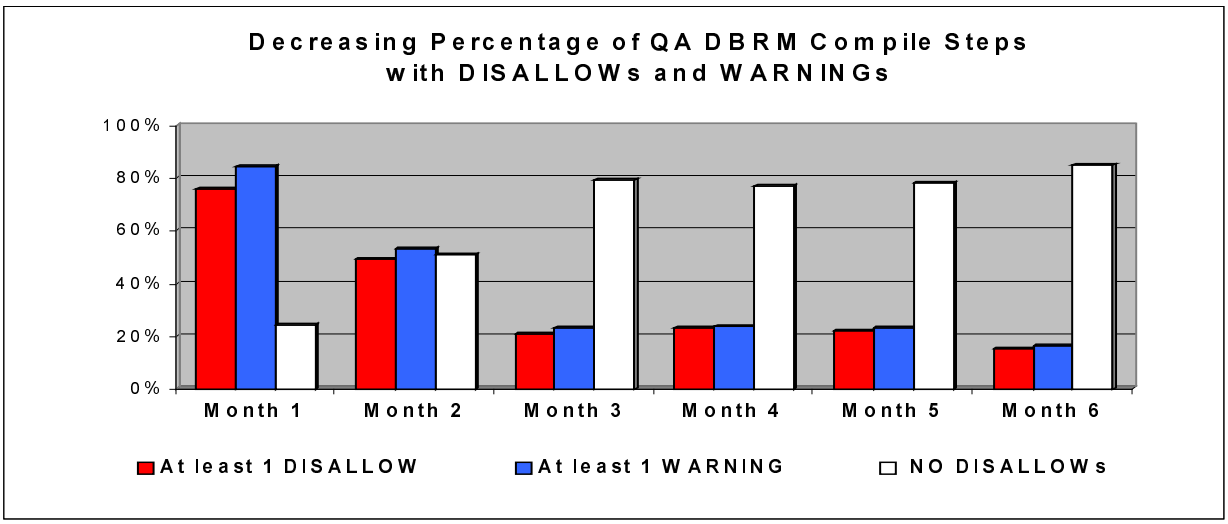
```
SELECT TBCREATOR,TBNAME,NAME,C.COLTYPE,LENGTH
        FROM SYSIBM.SYSCOLUMNS C
        INNER JOIN
        (SELECT COLTYPE,AVG(LENGTH) AS ALEN FROM SYSIBM.SYSCOLUMNS
                GROUP BY COLTYPE ) AS TEMP
                ON TEMP.COLTYPE=C.COLTYPE WHERE C.LENGTH > TEMP.ALEN
```

Tracking changes in quality using the History Database

DB2 may execute the same SQL differently in different situations. Monitoring such differences can uncover problems before they occur. The DB/IQ History Database facilitates these comparisons.

Execution may vary for many reasons. The same SQL used against regional or departmental databases running in different partitions is one example. Recompilation of unchanged SQL when the RUNSTATS have changed since the prior compilation is another. You can even compare future performance by implementing the RUNSTATS simulator. Differences are highlighted simply by selecting the versions to be compared from an ISPF screen which lists all stored versions.

The History Database is also used to provide an **audit** facility. Detailed statistics on rule evaluations are stored in the database whenever SQL is analyzed. These statistics can later be analyzed to see how overall SQL quality, or the quality of SQL used in a particular application or generated by a particular application development team, has changed over time. It is not uncommon to see the frequency of problems uncovered by DB/IQ cut by 50% or more in the first six months of its use.



DB/IQ QA Benefits

DB/IQ will typically uncover problems on the first day that 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, DBA, programmer, and user resources and cost.**
- **Automatically evaluate existing SQL libraries 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 database growth.**
- **Free up DBAs from routine QA for more beneficial design and optimization tasks.**

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 MA is an ISPF based migration aid. It helps move DB2 applications from one subsystem to another. Users select objects at any level of the catalog using full names or generics. DB/IQ MA generates the GRANT statements and DDL for the objects while selectively including or excluding certain types of objects, comments or labels, maintaining referential integrity and respecting all defined DB2 privileges.

DB/IQ ISPF provides DBAs and application developers fast and easy access to information needed to optimize system performance and understand their DB2 environment. It provides catalog navigation, the ability to analyze the benefits of possible Reorgs, plus a unique cross-reference facility between plans, programs, tables, views, indexes etc. able to drill down to individual column names and predicate usage.

DB/IQ - CICS provides many of the same features as DB/IQ ISPF in a CICS environment. In addition, it provides a a facility to execute a statement and analyze details of its timing, access paths and costs. Usage of these utilities by application developers can significantly reduce the necessity for DBAs to get involved in such optimization.

InSoft Software GmbH
Derendorfer Str. 70
40479 Dusseldorf
Germany
Tel: +49-211-48 80 31
Fax: +49-211-48 80 33
www.insoft-software.de