BY CRAIG S. MULLINS

From index processing to access-path selection, IBM has retooled its strategic DBMS

DB2 v. 2.2: A Few More Bells & Whistles

THE MOST TOUTED aspect of the latest version of DB2, officially released by IBM last fall, has been its support of distributed database. But that's not the only difference; other important modifications include changes to the way data is accessed, an increase in the information provided by the EXPLAIN command, and an increase in the statistics stored for query optimization. These changes have the potential to significantly enhance query performance.

First, DB2 v. 2.2 lets you use multiple indexes for an access path. While the need for multiple indexes to satisfy specific portions of a query might seem obvious, that feature wasn't supported until this release.

To analyze a query, the DB2 optimizer breaks it down into query subsets known as query blocks. Each query block can access one new table. Multiple indexes can now be used, a feature unavailable prior to v. 2.2. When using multiple indexes to satisfy a query block, DB2 takes one of two approaches: row-identifier intersection or RID union. (An RID identifies a row within a page to DB2.) When multiple indexes are used to satisfy an AND condition in the WHERE clause of a query, RID intersection is used. When multiple indexes satisfy an OR condition in the WHERE clause, RID union is used.

To clarify these concepts, let's consider an example: a table (Relation1) with two indexes, IDX1 on Column1 and Column2 and IDX2 on Column6 and Column4. Note the queries in Listing 1. With v. 2.2, query 1 could have used either IDX1 or IDX2, but not both. With an earlier version, this decision would have been based on the statistics in the DB2 catalog (such as the cluster ratio, cardinality, and so forth). Now DB2 can use both indexes and RID intersection. It can also satisfy the Column1 portion of the WHERE clause using IDX1 and the Column6 portion using IDX2. The RIDs would then be intersected and the data retrieved.

Query 2, on the other hand, would not have used any index under v. 2.1 because of the OR in the WHERE clause. Now both indexes can be used; DB2 can satisfy the Column1 portion of the clause using IDX1 and the Column6 portion using IDX2. The RIDs that were retrieved would then be appended via UNION, and the data would be retrieved.

Of course, the DB2 optimization process still takes other information into account when deciding on the best way to satisfy the query. Also, DB2 doesn't use multiple indexes when certain thresholds are reached. For RID intersection, the threshold occurs when fewer than 32 RIDs qualify for the query block. For RID union, the threshold occurs when more than one quarter of the RIDs in the table qualify for the query block. This limit prohibits DB2 from using multiple indexes when it's unnecessary or when it could adversely affect performance.

IBM has released figures showing a decrease of 10:1 elapsed time and 5:1 CPU time between v. 2.1 and v. 2.2 when multiple indexes were used. However, this feature may also increase the cost of your query. Multiple indexes usually enhance performance, but beware when converting: You must test *all* of your critical production queries with the new release of DB2 before moving to production.

DISTRIBUTION STATISTICS

Version 2.1 and all prior releases work on the assumption that all data in all tables is uniformly distributed. This implies that a table with 100 rows and 10 unique values has 10 occurrences of each unique value. Version 2.2 corrects this problem.

The RUNSTATS utility has been modified to gather statistics for the first column of multicolumn indexes and all nonunique, single-column indexes. This data is stored in the catalog for use during query optimization. Distribution statistics give the optimizer better information about the nature of the data needed to satisfy the query, resulting in better access-path selection.

DB2 can't always use distribution statistics. Queries with the IN and LIKE predicates won't use them; nor will queries with host variables. DB2 still assumes uniform distribution in these cases.

The performance implications of this enhancement appear to be as impressive as those for multiple-index usage. IBM's figures indicate the possibility of a 7:1 decrease in elapsed time and a 2:1 decrease in CPU time. Once again, these numbers are the results of lab benchmarks and may not accurately reflect the performance gain (or loss) your shop will experience.

LIST PREFETCH

Another new facility is list prefetch. Similar to sequential prefetch, it's used to enable more efficient processing of nonclustering indexes. At this point, a short refresher on clustering may be useful. The term refers to the process of physically ordering table data by specified column values. Each index has an associated cluster ratio that indicates the extent to which the table is in physical order by that index's keys. The higher the cluster ratio of data in DB2 tables, the better queries against the tables will perform.

Until list prefetch was implemented, additional I/O was likely to be incurred when tables with low cluster ratios were accessed. The list prefetch facility reduces the I/O for tables accessed via indexes with a low cluster ratio. It works by sorting the RIDs retrieved in sequence and invoking a single I/O for multiple pages containing those RIDs. With list prefetch, up to 32 pages can be retrieved at a time, and multiple I/Os for the same page are eliminated. Previously, pages were retrieved one by one based on RIDs in the index.

When invoking list prefetch, you may notice a lower elapsed time. It is important to realize, though, that overhead is involved. To sort RIDs, you must obtain sort work space from the DB2 buffer pool. An RID pool is formed from this space. IBM has stated that rigorous checks are used to ensure effective use of the RID pool. Up to 50% of the buffer pool can be used (to a maximum of 200 MB).

STAGE 1 PROCESSING

Evaluating a predicate in the Data Manager portion of DB2 is more

Another useful addition is the ability to create a table ALIAS

efficient than evaluating it in the Relational Data Services portion. The Data Manager acts upon data sets, whereas the RDS acts upon rows and columns. Any predicate evaluated in the Data Manager is said to be in stage 1; any predicate evaluated in the RDS is said to be in stage 2. DB2 2.2 evaluates IN, NOT IN, = ANY, and ∧ = ALL at stage 1. (IBM's *DB2 System and Database Administration Guide*, SC26-4374-1, lists the stage 1 predicates.) Previously, these predicates were evaluated at stage 2.

In addition, column functions (such as SUM) are also evaluated at stage 1 if the query uses only stage 1 predicates, is requested for a single column, and is not involved in a sort for GROUP BY. For joins, the function must be on the innermost table.

Both of these enhancements may improve the performance of queries that meet those conditions. IBM has released figures that indicate a possible 4:1 decrease in both elapsed time and CPU time when DB2 uses these enhancements.

EXPLAIN ENHANCEMENTS

EXPLAIN is the DB2 utility that examines SQL and reports on the access path chosen by the optimizer. The columns in PLAN_TABLE are listed in Table 1. Three new columns have been added to EXPLAIN's PLAN_TABLE output, and new values were added to an existing column.

Two welcome additions to PLAN_TABLE are the COLUMN_FN_EVAL and PREFETCH columns. The first reveals whether or not the column function for this query block will be evaluated during stage 1 or stage 2 processing to further aid query performance analysis. The PREFETCH column indicates whether sequential prefetch, list prefetch, or no prefetch was performed for this query block. Before this enhancement to EXPLAIN, only a performance monitor report could indicate whether or not sequential prefetch was invoked.

EXPLAIN has also been changed to show more than one index for an access path. The ACCESSTYPE column, in conjunction with the new column MIXOPSEQ (multiple-index operation sequence), provides information on each index in an index path. The ACCESSTYPE column has been enhanced to allow for codes indicating multiple indexes and the manner in which those indexes are being used (scan, RID intersection, or RID union). The MIXOPSEQ column contains a sequence number for the multiple-index indicators (or zero for any other ACCESSTYPE). To use these new columns, DBAs will have to change all PLAN_TABLEs before implementing v. 2.2.

MISCELLANEOUS CHANGES

Several other changes are worth noting. First, the performance of correlated subqueries has been modified in certain instances. The MAX and MIN column functions no longer require an index scan (if an index exists) but now perform only one fetch. This one-fetch index scan is also shown in the ACCESSTYPE column of PLAN_TABLE. Range comparisons using ANY and ALL can now use an index. Finally, the new version modifies merge-scan join performance by avoiding a row sort for the inner table in certain cases.

Another useful addition is the ability to create a table ALIAS. This feature reduces the name length of tables participating in distributed systems. (Data distribution adds a high-level qualifier LOCATION to the table name.) Regardless of its intended use, the ALIAS can be used as a global SYNONYM. Available in previous releases of DB2, SYNONYMs are accessible only by their creators. An ALIAS

```
Query 1:                              Query 2:

SELECT *                              SELECT *
FROM Relation1                        FROM Relation1
WHERE Column1 = 'VALUE1'              WHERE Column1 = 'VALUE1'
AND  Column6 = 'VALUE6';              OR   Column6 = 'VALUE6';
```

LISTING 1. *Sample queries to clarify index use.*

Column Name		Column Usage
QUERYNO		Integer to identify EXPLAIN statements
QBLOCKNO		Integer to identify subselects within an SQL statement
APPLNAME		Plan name (spaces for dynamic EXPLAIN statements)
PROGNAME		Program name that issued the EXPLAIN (spaces if not called from a program)
PLANNO		Integer to identify steps within the plan; shows the order in which DB2 processes each step
METHOD		0 . . . First table accessed
		1 . . . Nested-loop join
		2 . . . Merge-scan join
		3 . . . Sort for ORDER BY, GROUP BY, or DISTINCT
CREATOR		Table creator for this step's new table
TNAME		Name of this step's new table
TABNO		Integer to identify separate references to same table
ACCESSTYPE		Method of accessing the new table for this step:
		I . . . . Index
	+	I1 . . . One-fetch index scan
		R . . . Sequential page scan
		N . . . Index (for predicate using IN)
	+	M . . . Multiple-index scan
	+	MX. . Multiple-index scan on ACCESSNAME
	+	MI . . Multiple-index intersection
	+	MU . Multiple-index union
MATCHCOLS		Number of index columns used in an index scan
ACCESSCREATOR		Index creator for this step's index
ACCESSNAME		Name of index used in this step (blank if none)
INDEXONLY		Y . . . Index alone used to satisfy request
		N . . . Data must also be accessed
SORTN_UNIQ SORTN_JOIN SORTN_ORDERBY SORTN_GROUPBY		Whether a sort is done on the new table for uniqueness, join, ORDER BY, or GROUP BY (Y/N)
SORTC_UNIQ SORTC_JOIN SORTC_ORDERBY SORTC_GROUPBY		Whether a sort is done on the composite table for uniqueness, join, ORDER BY, or GROUP BY (Y/N)
TSLOCKMODE		Lock mode for the table space that holds the new table for this step (IS, IX, S, X)
TIMESTAMP		Time the EXPLAIN statement was executed
REMARKS		Comments
PREFETCH*		S . . . Sequential prefetch
		L . . . List prefetch
		" " . . No prefetch
COLUMN_FN_EVAL*		R . . . Data retrieval time
		S . . . Sort time
		" " . . Data manipulation time (or unknown)
MIXOPSEQ*		Integer indicating sequence for multiple-index scans

TABLE 1. *PLAN_TABLE columns. New columns are marked with an asterisk; new column values are marked with a plus.*

can be accessed by anyone granted the proper authority. ALIASes are maintained using the CREATE and DROP SQL verbs and are recorded in the SYSIBM.SYSTABLES table in the DB2 catalog. ALIASes could be used as replacements for VIEWs on base tables where all data and columns are accessible.

Don't forget that IBM's performance claims are based on best-case scenarios—simply moving from one release to the next won't necessarily improve performance. Each enhancement addresses specific changes made for specific types of queries. If the nature of your queries is such that they can benefit from these changes, you may see a significant improvement in performance.

Craig S. Mullins is a database and system administrator specializing in DB2 at Mellon Bank in Pittsburgh, Pa. He is cofounder and vice president of ASSET Inc., a customized software and technical consulting firm, and a DB2 instructor.

PUT YOUR READER SERVICE CARD TO WORK FOR YOU!

DATABASE PROGRAMMING & DESIGN knows you're busy, and sometimes there isn't enough time to call an advertiser for information. That's why we have provided Reader Service cards for you to obtain information on the products and services advertised in this issue.

The Reader Service card (located in the back of this issue) is an efficient, effective and cost free way for you to get the information you need. Simply circle the corresponding numbers of the products you are interested in, fill out your name and address, and drop the postage-paid card in the mail. We will take care of the rest!

We'll make sure that your request is passed on directly to the advertiser, who will send free details of their products and services back to you.

SEND IT TODAY!