BBj Data Server
Delivers Database Analysis

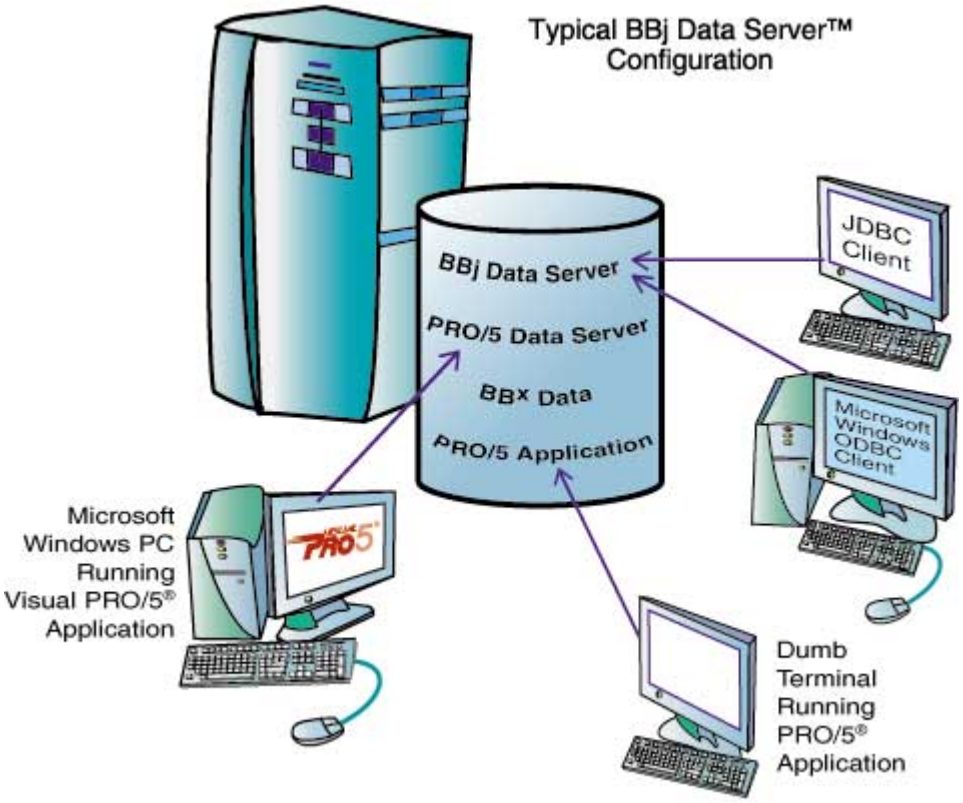
By Jeff Ash

Staying competitive in today's marketplace requires that businesses effectively and appropriately use available technologies. With the release of BBj™, BASIS Customers will have even greater access to cutting edge technologies. This article discusses one component of BBj-the BBj Data Server™. The BBj Data Server is a powerful file server and database management system rolled into one. In particular, this article discusses the new optimization features never before available in BASIS products.

OPTIMIZATION

One of the tremendous improvements made to the SQL engine in the BBj Data Server over previous BASIS SQL engines is the presence of a much more intelligent optimization system. To understand the BBj optimization system, we must first discuss indexes and how they are used to optimize SQL statements.

Indexes (or keys) are a part of the data file and allow the SQL engine to skip over records that could not possibly meet the criteria specified by a query. Without indexes, the SQL engine has no way to limit the number of records it must search, so it must search every record in the data file. Another important fact about the use of indexes is that they can only be used if the column (or columns) contained in the index have a filtering value specified in the WHERE clause of a query. For example, if there is an index on the STATE column and no index on the ZIP_CODE column, the statement `SELECT * FROM customer WHERE zip_code = '87002'` would still have to iterate through every record in the table to make sure it doesn't miss anything. The statement `SELECT * FROM customer WHERE state = 'NM'` would be able to skip over all states except "NM" because the index makes it possible. This kind of simple case optimization does not require much intelligence on the part of the SQL engine as it simply checks to see if one of the columns specified in the WHERE clause has a usable index, and if so, it uses it.



AMBIGUITY PROBLEMS

In the example above, how to optimize the statement is obvious. However, it is very common for a statement to include a number of expressions in the WHERE clause that could be used for optimization using different indexes. Because the SQL engine can only choose one index on each table, this can cause an ambiguous situation to arise. Previous BASIS SQL products have no way of knowing which one of the indexes is better or less repetitive in its values. Choosing the wrong index can make no noticeable difference in the speed of a query, or it can mean the difference between iterating over 1,000,000 records or 100 records.

THE BBj SOLUTION - DATABASE ANALYSIS

Because the BBj Data Server has a completely redesigned and redeveloped SQL engine, BASIS engineers have been able to significantly improve the optimization strategy used on SQL queries. Not only does the BBj SQL engine examine all possibilities before executing the statement, it now has the ability to take into account the uniqueness of indexes once the all-new "Analyze" feature is run on a database via the BBj Enterprise Manager. This feature only needs to be run one time on a database unless the table changes in a significant way or if indexes are added or dropped.

When the SQL engine goes through optimization of an SQL statement, it uses a process in which it scores each possible optimization strategy. This scoring process roughly attempts to determine how many records the SQL engine would have to search before returning the results to the user. The lower the score, the better. This scoring system uses the type of index (unique or non-unique), the operator (<, =, >=, LIKE, etc.) and the number of records in the table to determine a score. This is a very generalized process; one in which the SQL

engine makes a fair guess. In many cases, this guess is very accurate. However, when there is some ambiguity, such as "is LAST_NAME or ZIP_CODE more restrictive?", this guess may or may not be accurate, depending on the position of each expression in the WHERE clause.

This is where the analysis feature enters. When a table is analyzed, a new data dictionary file called FREQUENCY.DD is populated with information about the frequency of differing values within each index. For example, if there are 5,000 records in a customer table and there are 4,000 different last names, the analysis feature assumes that using the LAST_NAME index finds an average of 1.25 records for each value. Now, if there are only 20 different zip codes in the table, then there would be an average of 250 records for each value. Because this information is now stored in the data dictionary, the SQL engine can use it to decide that the LAST_NAME index is more restrictive than the ZIP_CODE index.

PERFORMANCE RESULTS

A real-world test case illustrates the dramatic difference between using analysis and not using analysis. One of our Customers had a table containing 1.5 million records. Because of the way the indexes were set up, a query had to iterate over all 1.5 million records to return approximately 250 results, taking a number of hours, because it was ambiguous as to which key would be more restrictive. In tests, once analysis was done, the query looked at only a few hundred records and returned results in a matter of seconds. We observed similar performance results testing our own in-house database, in which one query that took 164 seconds to return results without analysis took only 16 seconds to return results with analysis.

We're confident developers will appreciate the improvements in performance and stability that the BBj Data Server provides. With the all-new BBj Data Server, users of BASIS products have access to powerful new optimization built into BBj's SQL engine. The analysis feature takes BASIS' SQL engine to a whole new level of performance and reliability that improves the speed and reliability of users' data access.