JACKIE's Forum

Encoded Vector Indexes (EVI)

Jackie Jansen

Talking to a group of iSeries customers recently I was disappointed at how few of them had implemented Encoded Vector Indexes (EVIs). I decided that it was time I wrote another column on this subject. It appears that no-one paid much attention when I wrote about EVIs exactly five years ago this month.

EVIs, introduced with V4R3, were designed to assist in the performance of many SQL queries. They do not replace our traditional binary radix indexes. They are an additional method that the optimizer can use to implement a query. The optimizer can use EVIs in one of two ways. It may choose to use the EVIs during execution of a query or it may simply use the EVIs to get statistics about the data to assist in building the access plan.

SYMBOL TABLE

Key Value	Code	First Row	Last Row	Count
Alberta	1	3	64002	4033
British Columbia	2	2	150432	78762
....	...			
Quebec	8	1	875943	223989
Saskatchewan	9	41532	643022	20384

VECTOR

CODE	RRN
8	1
2	2
1	3
2	4
3	5
5	6
8	7
...	

EVIs don't fit every query. They are not used for grouping or ordering. Where they are appropriate is with columns that have fairly low cardinality or a relatively low number of distinct values. If your query is going to select somewhere between 20% and 70% of the rows then EVIs are often considered. If your query has complex ANDing and ORing EVIs may be used. On the other hand if your query is only going to select a few records then a binary radix index may be a very good fit. If the query is going to select almost all the records in a table then the system probably won't use an index at all, it may simply scan the entire table.

The good news is that you don't need to tell the optimizer which index to use. You simply create the indexes ahead of time and the optimizer will choose the appropriate implementation method.

To create an EVI you can either use the SQL "CREATE ENCODED VECTOR INDEX" statement or you can use iSeries Navigator. When you create an EVI the system will generate a symbol table and a vector. The symbol table has one row for each distinct key value in your table. It also specifies the relative record number (RRN) of the first row in a table that has that specific key value and the last row in the table that uses that same key. If you look at the example, the system would know that if you were searching WHERE PROVINCE='British Columbia' it would start looking at record 2 but it could stop searching at record 150,432 even if the table had a million records. In addition the symbol table contains a count of all the rows containing each distinct key value. In V5R3 the system may choose to use only the EVI and not even access the data for queries that want either a count or a list of distinct values. If your application currently fills a drop-down list with distinct values for a user to choose from you may see a nice performance gain.

The vector contains one element for every row in the table. Element 1 in the vector represents RRN 1 or row 1 in the base table. Element 2 in the vector represents row 2 in your table etc. The symbol table contains a unique code for every distinct key value. This allows for data compression. This code is stored in the vector. In the example shown here row 1 or relative record 1 in your table would contain the key 'Quebec'. When searching for PROVINCE = 'QUEBEC' the system will create a bitmap or an array of 1's and 0's where the first element of the array references relative record 1 etc. The array will contain a 0 if the code doesn't match your selection criteria and a 1 if it does. If you have a query WHERE PROVINCE='QUEBEC' and PRODUCT = 'COLA' the optimizer can choose to look at the EVI you have previously created for PROVINCE and the EVI you have previously created for PRODUCT and then "AND" the two EVIs together. Any element in the array or bitmap that contains a '1' after the ANDing will point to a relative record that matches your search criteria.

Customers who implemented EVIs saw queries that took hours go down to minutes and queries that took many, many minutes go down to seconds.

By the time you read this column Service Pack 3 for iSeries Access for V5R3 will be available. You can run the V5R3 client for iSeries Access even if your server is running V5R2. Service Pack 3 includes new functionality to indicate when an index has been used either during query execution or to gather column statistics during query optimization. This same functionality is being added to V5R2 via PTF SI16313. Update iSeries Access, evaluate your queries, create EVIs and then analyze if the EVIs are being used and if your performance has improved. Good luck and happy querying. TUG

***Jackie Jansen** is a Senior Consulting IT Specialist. She currently works in the IBM Americas Advanced Technical Support Solutions Centre. Jackie is a frequent speaker at iSeries Technical Conferences and User Group meetings. Contact her at jjansen@ca.ibm.com.*