Dictionary Encoding

INTRODUCTION

Encoded data for reduced storage space and faster processing.

DESCRIPTION

Most databases and big data SQL data stores employ columnar encoding to achieve data compression by storing small integer numbers (surrogate values) instead of full string values. However, almost all existing databases and data stores divide the data into row groups containing anywhere from few thousand to a million rows and employ dictionary encoding only within each row group. Hence, the same column value can have different surrogate values in different row groups. So, while reading the data, conversion from surrogate value to actual value needs to be done immediately after the data is read from the disk. But CarbonData employs global surrogate key which means that a common dictionary is maintained for the full store on one machine/node. S o CarbonData can perform all the query processing work such as grouping/aggregation, sorting etc on light weight surrogate values. The conversion from surrogate to actual values needs to be done only on the final result. This procedure improves performance on two aspects. Conversion from surrogate values to actual values is done only for the final result rows which are much less than the actual rows read from the store. All query processing and computation such as grouping/aggregation, sorting, and so on is done on lightweight surrogate values which requires less memory and CPU time compared to actual values.

ENCODING TECHNIQUE

I) Original Data :

blocked URL

Figure 1: The data at time of loading

II) Dictionary Generation :

All the Multi Dimensional Keys(MDK)* are compressed to some lightweight(surrogate) values, which results in less memory usage. This encoding is used to achieve data compression by storing small integer numbers (surrogate values) instead of full string values. All nulls have a default value 0, Others are assigned values accordingly.

blocked URL

Figure 2 : Surrogate Keys

All query processing and computation such as grouping/aggregation, sorting, and so on is done on lightweight surrogate values which requires less memory and CPU time compared to actual values.

III) Dictionary Encoding :

After generating the dictionary(the surrogate values for column values), the table data is updated accordingly with the new surrogate values.

blocked URL

Figure 3 : Data in form of surrogate keys

IV) Sorting(on MDK : Multi Dimensional Keys) :

The multi dimensional keys are then sorted, and table data is arranged accordingly.

blocked URL

Figure 4 : Data in Sorted Order

V) Blocklet Logical View :

blocked URL

Figure 5 : The final column wise sorted data

Conversion from surrogate values to actual values is done only for the final result rows which are much less than the actual rows read from the store.

*(MDK)Multi Dimensional Keys are the columns which represent dimensions(the keys to analyse data) of the table(ex: Location, Months etc)