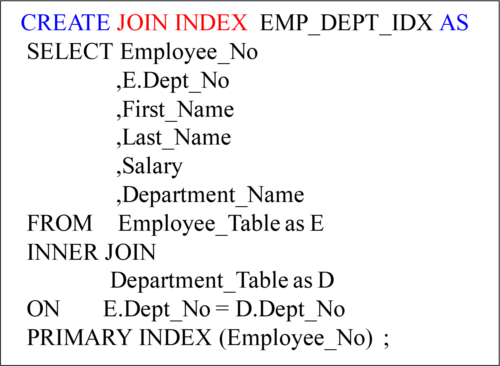
Chapter 9 - Join Indexes

“Every sunrise is a second chance.”

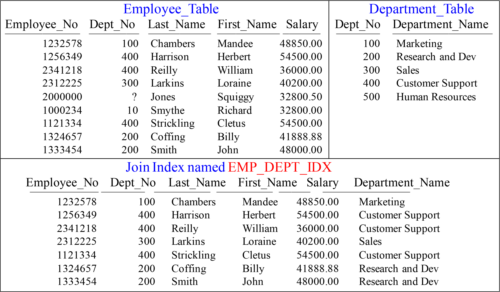
– Unknown

Creatinga Multi-Table Join Index



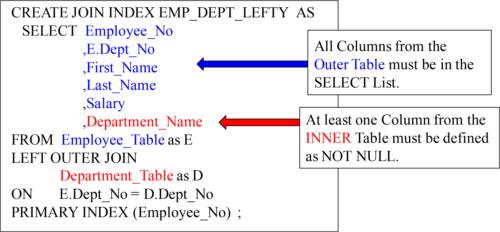
The Syntax above will create a Multi-Table Join Index with a NUPI on Employee\_No. The next slide will illustrate a visual so you can see the data in the Join Index. Join Indexes are created so data doesn’t have to move to satisfy the join. The Join Index essentially pre-joins the table and keeps it hidden for the Parsing Engine to utilize. Remember, the user cannotselect **directly** from the **Join Index**. The PE decides when it is to be used whenever the user queries the base table or the views of the base table.

Visualof a Join Index



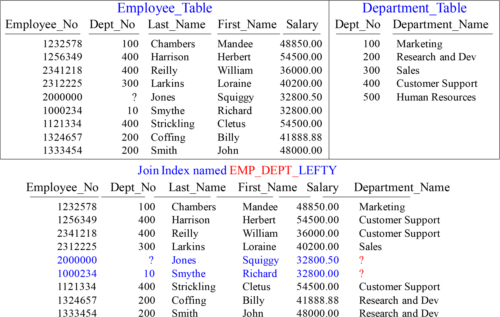
The Join Index looks like an Answer Set, but each row is stored like a normal table in that the rows of the Join Index are spread amongst the AMPs. Users can’t query the Join Index, but the Parsing Engine gets data from the Join Index when it chooses.

OuterJoin Multi-Table Join Index



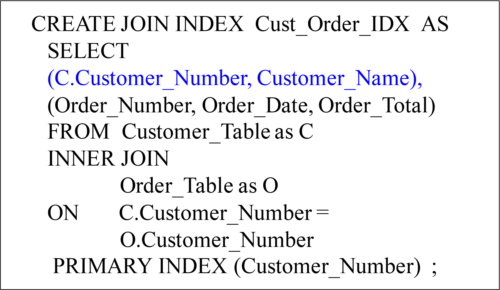
A Multi-Table Outer Join Index has some very specific rules above to remember. Turn the page to see a visual of the data.

Visualof a Left Outer Join Index



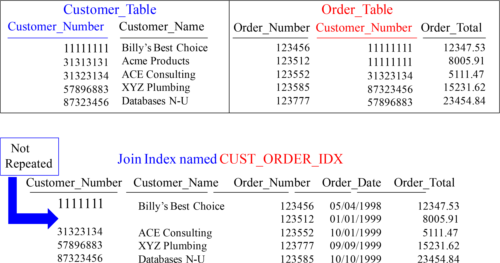
The Outer Join Index has the additional rows that did NOT match.

CompressedMulti-Table Join Index



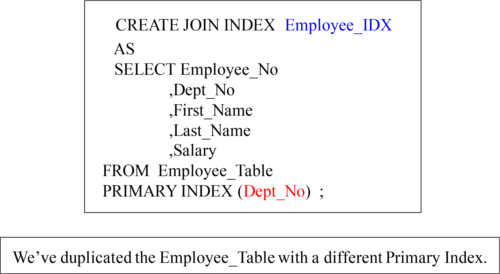
A Compressed Multi-Table Join Index won’t keep repeating the same Customer\_Number and Customer\_Name, but only list it once. A visual example follows on the next page. Remember, a **compressed Join Index** (such as the example above) is an example of **Row compression** becuase it eliminates space by **not duplicating** the Customer\_Number or Customer\_Name. Some rows are essentially smaller and this is a form of **compression**!

AVisual of a Compressed Multi-Table Join Index



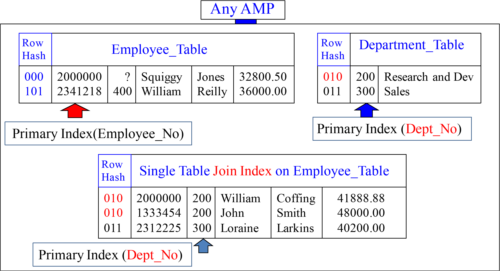
Billy’s Best Choice is Customer\_Number 11111111 and they have placed two orders, but the Customer\_Number and Customer\_Name don’t repeat unnecessarily.

Creatinga Single-Table Join Index



If a USER queries with the Dept\_No in the WHERE clause this will be a Single-AMP retrieve. If the USER joins the Employee and Department Tables together then Teradata won’t need to Redistribute or Duplicate to get the data AMP local. The next page will give you a visual of how that looks on a particular AMP.

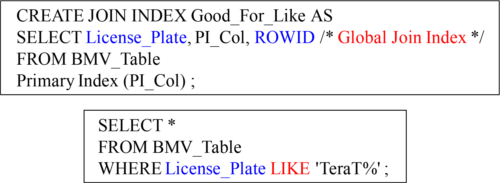
Conceptualof a Single Table Join Index on an AMP



Notice the Primary Indexes on both tables and the Single Table Join Index. The Join Index gives the Parsing Engine options. If a query is run against the Employee\_Table with Employee\_No in the WHERE clause, it will use the normal table, but if a user Uses Dept\_No in the WHERE clause, it will use the Join Index. If a user needs to join the Department\_Table to the Employee\_Table, the Join Index is used so no data moves.

SingleTable Join Index Great For LIKE Clause

Build a STJI with column that contains three columns. They are the LIKE column being queried, the primary index column of the base table and the keyword ROWID!

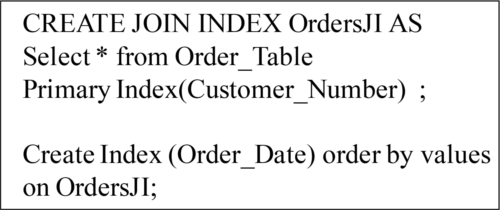


The PE will choose to scan the narrow table (the Join Index) and qualify all rows that qualify with a car license LIKE 'TeraT%', then the PE uses the ROWID to get data from the BMV\_Table where row is on the same AMP because both the base table and join index are on the same AMP because they both have the same Primary Index. This can save enormous time for queries using the LIKE command.

A LIKE command on a base table will never use a Non-Unique Secondary Index (NUSI). The above technique should be tested and only used if a lot of users are utilizing the LIKE command on a large table. If that is the case a lot of time can be saved.

SingleTable Join Index with Value Ordered NUSI

A Value Ordered NUSI can only be done on columns that are 4-byte integers. Dates qualify because they are stored internally in Teradata as 4-byte integers.



A value ordered index has been expanded from 16 to 64 columns.

Indexes are always sorted by their hash, but a Value Ordered index is sorted on each AMP by values and not hash. This is a great technique for you to run trials on.

AggregateJoin Indexes

Aggregate Join Indexes may be defined on:

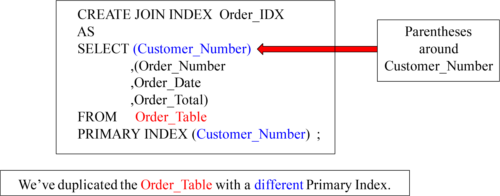
Single Tables - A columnar subset of a base table with aggregates automatically maintained by Teradata.

Multiple Tables -A columnar subset of as many as 64 base tables with aggregate columns automatically maintained by Teradata.

Sparse Join Indexes are defined with a WHERE clause that limits the number of base table rows included and the space required to store them.

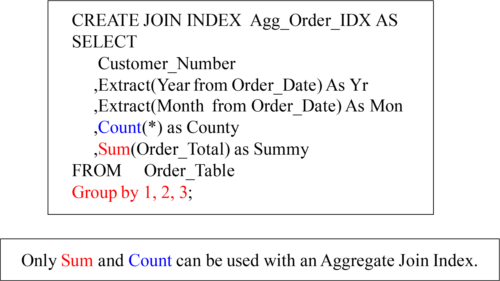
Aggregate Join Indexes can only include SUM and COUNT values. You can calculate Averages from these two columns though.

CompressedSingle-Table Join Index



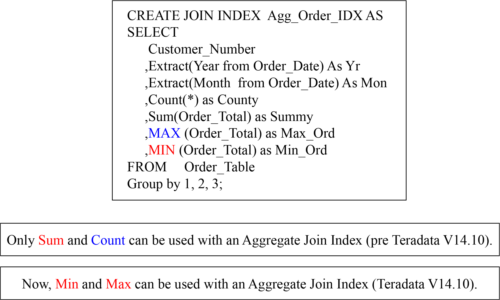
This is the compressed version of a Single-Table Join Index. Notice the parentheses around Customer\_Number in the SELECT list. A single row is used for each customer, with repeating orders per customer inside the Join Index. Remember, a **compressed Join Index** (such as the example above) is an example of **Row compression** becuase it eliminates space by **not duplicating** the Customer\_Number. Some rows are essentially smaller and this is a form of **compression**!

AggregateJoin Index



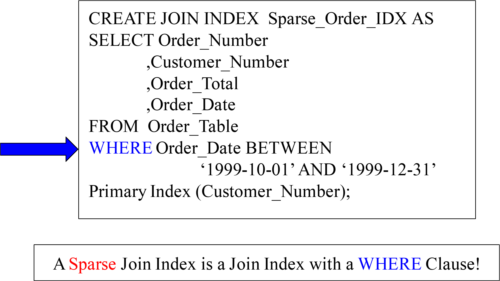
Users never query the Join Index directly. It is the Parsing Engine that commands the AMPs to pull the data from the Join Index. You can extract the Month and Year and calculations are updated as the Base Table changes. Count and Sum are required to be a data type of FLOAT. Why can you only have Count and Sum? Max and Min aren’t that important to know and Average isn’t all that important either. Business users want to know how much and how many so AVERAGE, MIN, and MAX are excluded!

NewAggregate Join New Index Aggregate ( Teradata Join V14.10Index) (Teradata V14.10)



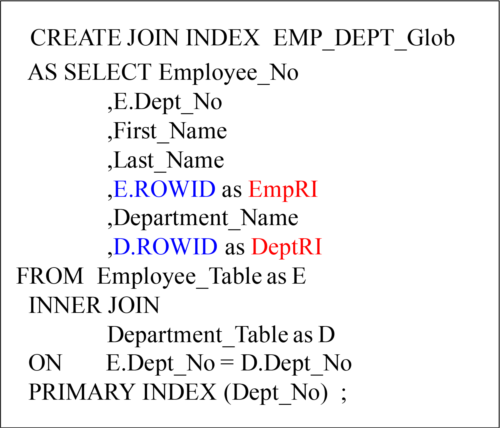
Aggregate Join Index (AJI) has always supported (SUM and COUNT), but with Teradata V14.10, there is support for MIN/MAX aggregates also.

SparseJoin Index



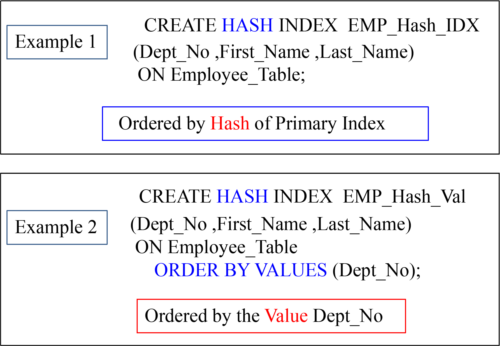
A Sparse Join Index has a WHERE clause so it doesn’t take all the rows in the table, but only a portion. This is a very effective way to save space and focus on the latest data.

AGlobal Multi-Table Join Index



With the ROWID inside the Join Index, the PE can get columns in the User’s SQL NOT specified in the Join Index directly from the Base Table by using the Row-ID.

Creatinga Hash Index



A Hash Index can be Ordered by Values or by Hash.

JoinIndex Details

•Join Indexes are physically stored exactly like normal Teradata tables.

•Users can’t query the Join Index directly, but PE will decide when to use.

•Join Indexes are automatically updated as base tables change.

•Join Indexes can have Non-Unique Primary Indexes (NUPI) only.

•Join Indexes can have Non-Unique Secondary (NUSI) Indexes.

•Max 64 Columns per Table per Join Index.

•BLOB and CLOB types cannot be defined.

•Triggers with Join Indexes allowed V2R6.2.

•After Restoring a Table, Drop and Recreate the Join Index.

•FastLoad/ MultiLoad won’t load to tables with a Join Index defined.