Maintaining Index Sequential Sets in DMSII – On-Line Garbage Collection

This article is in two parts, the first part covers Garbage Collection without going into technical details of the structure of I/S sets. The second part provides a deeper look at I/S sets in technical detail for those who are interested.

The first part of this article explains what I/S Set Garbage Collection is, why it needs to be done and what to look at to determine the sets that need it.

Index Sequential (I/S) Sets as implemented in DMSII require some attention. The nature of the architecture of the I/S set can cause both performance issues and/or result in LimitErrors. (See **Index Sequential Sets Technical Details).**

However, today's DMSII includes tools to prevent problems with I/S sets by making it easy to garbage collect on-line with the database in use. Garbage collection is a process that "cleans up" an I/S set by rearranging the entries in the sets to eliminate wasted space.

Wasted space in a set can cause three problems:

LimitErrors because of wasted space in the Set

When space is wasted in a set, the set grows faster and becomes larger that it should resulting in unexpected LimitErrors.

Performance problems because each read of a set contains wasted space

DMSII reads blocks of data from databases. In I/S sets the blocks are called tables. The blocksize of the I/S set determines the size of the tables and is the number entries in a table. Tables can have entries that are not used (empty) within a table. This is caused by the way sets are created and maintained. For example if a table has 50% of its entries empty, when DMSII reads the table 50% of the read is wasted.

Performance problems because the number of Levels in an I/S structure is too high

Internally I/S sets are tree structures and contain levels. Each level points to entries in the next level, with the exception of the bottom level. The bottom level entries point into the dataset which is using the set. For best overall performance it is best to keep I/S sets at 3 levels if possible.

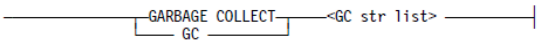
On-Line Garbage Collection

On-line garbage collection can fix or improve all of the above problems. It gathers the entries into the existing tables and eliminates empty tables.

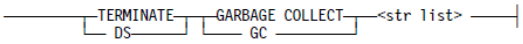
On-line garbage collection is started using a Visible DBS Command. The command consist of the structure to be collected and the loadfactor to be used during the collection.

Syntax

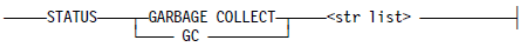
Initiating Garbage Collection



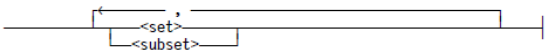
Terminating Garbage Collection



Monitoring Garbage Collection:



<str list>



On-line Garbage Collection can be used to garbage collect for a maximum of 10 sets at a time. Additionally the Independenttrans option must be set for the database.

On-line Garbage Collection can't run at the same time as:

- An online or offline dump
- A database reorganization
- A reconstruction on any structure in the database
- A Remote Database Backup takeover

With dbaTOOLS, there are three reports that are used to determine which I/S Sets that need to be garbage collected. The Set LimitError Projection report, the Set Loading Report and Level Analysis Report.

The first thing to consider is whether the set is close to a LimitError, but be aware, that an on-line garbage collection may or may not fix this problem. It depends upon how much wasted space is in the set, because of the set doesn't have much wasted space, a garbage collection will not help to extend the life of the set.

This is where the Loading information becomes important. Don't confuse loading with LoadFactor, LoadFactor is used by DMSII to attempt to control loading. The lower the actual loading of the set, the more wasted space it contains, and more that can be gained by a garbage collect. If for example the actual loading of the set is around 50%, that means half the set is empty.

The Level information is more of a performance consideration, for best overall performance it good to keep the number of levels at 3 or less.

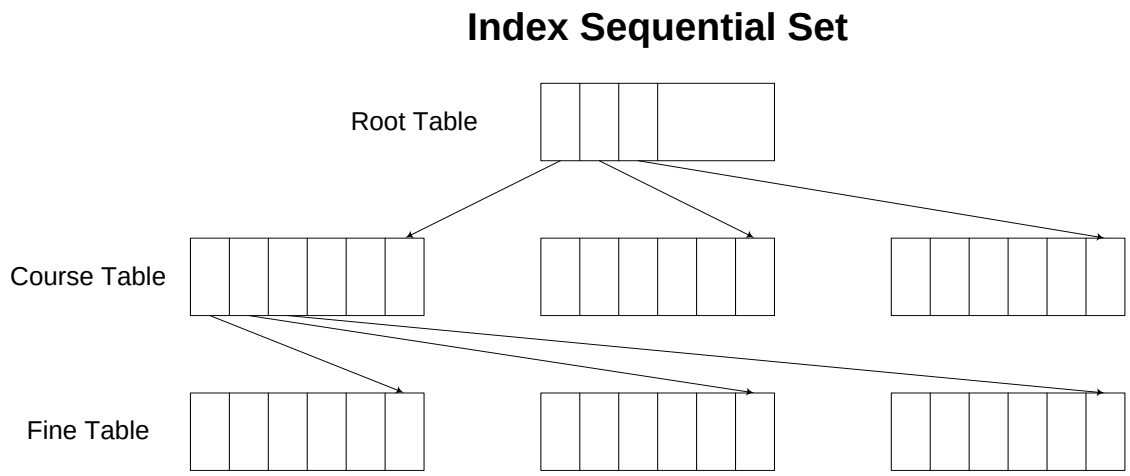
Index Sequential Sets Technical Details

I/S sets grow differently than the datasets they span. While dataset growth is based largely on population (in records), set growth is measured in areas (portions of disk). Datasets can be allowed to grow to a thousand areas or can be limited to the DASDL population. Sets always are allowed to grow to a thousand areas.

The set tablesize, areasize and number of areas (always 1,000) determine the number of maximum entries in a set. The sets initial tablesize, areasize are determined by the datasets population. If every entry in a table was full a set would hold at least the population of the dataset, if not more. As discussed below, this is usually not the case, because most tables are not full, which can result in a set getting a limiterror with half the population of the dataset. That is why in dbaTOOLS, for example, there are two different LimitError reports, one for datasets and another for sets.

I/S sets are organized as a binary tree. Because of this structure, the number of entries per table determines the number of levels in the binary tree and the potential number of i/o's required to retrieve a record.

The binary tree consists of a root record at the top of the tree, coarse tables in the middle (we can have multiple levels of coarse tables) and at the bottom of the structure fine tables. Entries in the root table and the coarse tables point back into the set, while the entries in the fine table pointto a record in the dataset. To access a record we first access the root table, then the coarse tables then the fine table entry and then the dataset record. The number of levels in the table will determine the number of i/o's required to access a record. It is highly desirable to keep the total number of levels at three or less. The number of minimum levels is ultimately determined by tablesize (number of entries in the table) and the population of the set. (See appendix C of the DASDL manual for the formula used to determine number of levels versus the tablesize.)



Loadfactor is set in the DASDL. It determines how records are moved when a table is split. When a Table (block) is full and a new record needs to be added to the table the table must be split into two tables. Part of the entries stay in the existing table and part are moved to the new table. How this is done is determine by the LoadFactor and where the new entry is placed in the table. This process is what can cause wasted space in a table. The default LoadFactor for I/S sets is 66 percent. This is not a bad default for most sets. If records are added to the set in random order (based upon the key) then a

lower LoadFactor will result is better table loading. If records tend to be added sequentially (by the key) then the LoadFactor should be higher.