EMu Documentation

Range Indexing

Document Version 1

EMu Version 4.0

Contents

Range Indexing

Overview

KE EMu 3.1.01 saw the addition of new indexing methods to the database engine. In particular support for NULL indexing (whether a field is empty or not) and PARTIAL indexing (fast searching for leading characters, e.g. `a*`) was added. Tools were provided that allowed System Administrators to configure, via the EMu Registry, which fields required the new indexing methods. The new indexing facilities did not provide a mechanism for adjusting or tuning range indexes.

KE EMu 4.0.01 provides new tools that permit System Administrators to tune the range indexing used by EMu. Support for automatic optimisation of range indexes has also been added. Using these tools EMu can now provide optimal range indexes with significantly faster range based searches.

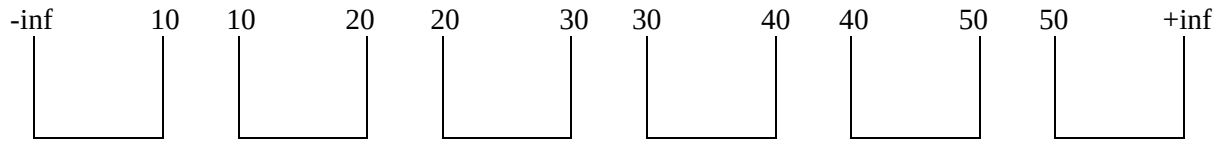
How range indexing works

The range indexing employed by the EMu database server is designed to work with the *Two Level Superimposed Coding Scheme for Partial Match Retrieval* system used for all searching. As it is an extension of the standard indexing EMu still requires only one set of index files, thus avoiding the need for costly "join" based queries.

Range indexing is really a series of "mini" indexes on a per field basis. Unlike the *Two Level* scheme, where all search terms are placed in the one index, range terms are placed in per field indexes that are then concatenated to form one range index. Each field index consists of a number of *range buckets*. These buckets are used to indicate whether a given value falls within the bucket or not.
There are two related considerations when establishing range buckets:

1. Data distribution: as best as possible data should be distributed evenly between the buckets.
2. Logical query ranges. The aim is to minimise the necessity to check a bucket for a value as checking whether records in a bucket match the query takes time; therefore if users are likely to search on particular ranges (decades for instance: 1/1/1910 to 1/1/1920) it makes sense to configure range buckets appropriately.

An example may make things clearer. Consider *Width*: (*Measurement Details*) in the Catalogue module. Let's say that the width of photographs is generally from 10cm to 50cm so we establish the following range buckets:

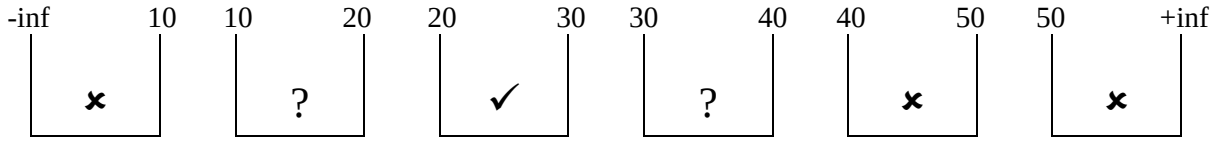


Note:

1. -infinity and +infinity will capture any values outside the specified ranges.

When a range search is performed these buckets are used to determine possible matches. Consider the following searches for photographs with a width of between:

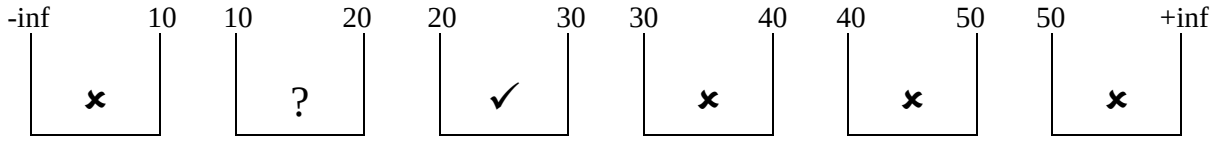
1. 15cm to 35cm



In this case:

1. Three buckets can be safely ignored.
2. One bucket does not need to be checked and is definitely included in the search.
3. Two buckets need to be checked for a match.

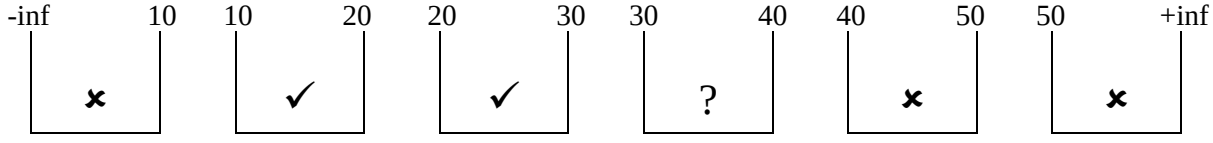
2. 15cm to 30cm



In this case:

1. Four buckets can be safely ignored.
2. One bucket does not need to be checked and is definitely included in the search.
3. One bucket needs to be checked.

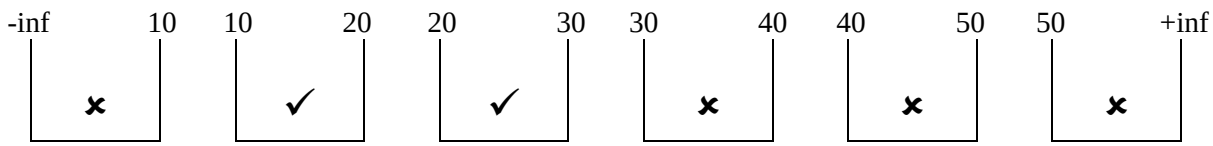
3. 10cm to 35cm



In this case:

1. Three buckets can be safely ignored.
2. Two buckets do not need to be checked and are definitely included in the search.
3. One bucket needs to be checked.

4. 10cm to 30cm



In this case:

1. Four buckets can be safely ignored.
2. Two bucket do not need to be checked and are included in the search.
3. No buckets need to be checked.

It should be clear from the last example that determining range buckets that match likely user searches has a direct influence on the speed of range based queries.

The problem with setting the range bucket values is that the bucket values depend on two variables. The first is the range of values entered into a field, in other words the *data distribution*. If the *Width* field contains a wide range of values, it makes sense to have a wide range of range buckets. If, however, the data is centred around one point, it may make sense to have a series of range buckets cover this period.

The second variable is the *query ranges* used to retrieve data. The problem with this variable is that for a given field it may be very hard to determine what sort of query ranges will be used without performing extensive analysis.

As these two variables cannot be known before data has been loaded EMu provides a default set of range buckets for each range searchable field. In general these buckets are satisfactory for a small numbers of records, however significant reductions in search times may be achieved by "tuning" the range buckets for large numbers of records.

Tuning the range indexes involves considering the two variables, data distribution and query ranges, and for each variable determining the set of range buckets that provides optimal performance. The objective in fine-tuning range indexing is to:

1. Minimise the number of buckets that need to be searched.
2. Minimise the number of records in buckets that need to be searched. Note that this objective will take precedence over the default distribution of records evenly between buckets.

EMu 4.0.01 provides a mechanism that allows System Administrators to have the range buckets tuned automatically based on the data distribution within a field. The facility does not take query ranges into account as this information requires subjective interpretation to determine which queries are important to have optimised and which queries can run a little slower.

The new facility does however provide data distribution information that may be used by a System Administrator to set the range buckets manually to achieve optimal performance where query ranges are known and can be weighted.

Hence the new facility provides automated range buckets, but allows System Administrators to override these settings and configure their own buckets manually.

Manual range bucket configuration

The setting of range bucket values is performed on a field basis. Each range indexed field may have a Registry entry that defines the range buckets for that field. If a Registry entry does not exist, the built-in values defined when the table was designed are used.

You can determine which fields have range indexing enabled by using the `emuindexing` command. For example, running `emuindexing eparties` will show all indexes available for the Parties module on a field by field basis:

```
Table "eparties"
        irn
                Type: Integer
                Indexing: Key
        SummaryData
                Type: Text
                Indexing: Word, Phonetic
        BioDeathEarliestDate
                Type: Date
                Indexing: Word, Range
        BioDeathLatestDate
                Type: Date
                Indexing: Word, Range
        ...
```

The word `Range` indicates fields that have range indexing enabled. Range indexing is available on fields of type:

- Integer
- Float
- Date
- Time
- Latitude
- Longitude

In general, all fields of the above types will have ranging already enabled. It is possible to enable and disable range indexing on fields of the above types via EMu Registry entries. The same Registry entry used to set range buckets can also "disable" ranging by setting the number of buckets to zero. Ranging is enabled by specifying one or more range bucket values on a field, provided the field is one of the above types. Range searching is not supported on *Text* or *String* based fields.

The Registry entry used to configure range buckets for a field is of the form:

`System|Setting|Table|`*`table`*`|Range Buckets|`*`colname`*`|`*`bucket;...`*

where *table* is the name of the EMu module that contains the field to be configured and *colname* is the name of the field (this can be determined by using the *What's this help?* facility in the EMu client, or via the `emuindexing` command). The *bucket* setting is a semi-colon separated list of values to be used for range buckets. The format of the value depends on the field type.

In our original example using the *Date Visited* field we had range buckets of 2000-01-01, 2003-01-01, 2006-01-01 and 2009-01-01. The following Registry entry may be used to set these buckets:

```
System|Setting|Table|ecollectionevents|Range
Buckets|ColDateVisited|2000-01-01;2003-01-02;2006-01-
01;2009-01-01
```

When using the `Range Buckets` Registry entry it is important to make sure that the values specified are all fully qualified. In particular, full date values are required. The table below shows what constitutes a fully qualified value for each field type:

Field Type	**Format**	**Examples**
Integer	n	-10, 12, 25
Float	n.*nnnn*	-23.0, -5, 2.125, 10
Date	yyyy-mm-dd	2003-10-23, 2008-03-17
Time	hh:mm:ss.*sss*	10:30:00, 18:00:00.000
Latitude	dd:mm:ss.*sss*:D	9:12:15.1:N,35:06:01:S
Longitude	ddd:mm:ss.*sss*:D	123:34:06.34:W

The Registry entry below could be used to set the range buckets on the *Latitude* field in the Collection Events module:

```
System|Setting|Table|ecollectionevents|Range
Buckets|LatLatitude_tab|16:00:00:S;18:00:00:S;20:00:00:S;
22:00:00:S
```

Using the `Range Buckets` Registry entry System Administrators can set the range buckets on any range based field manually. In order to help with this configuration a tool is provided that can analyse the values in a field and provide a data distribution table, as well as suggesting suitable bucket values.

emurangeupdate

The tool used to list, and optionally to set, suitable range buckets is called `emurangeupdate`. It is found on the EMu server. To use this facility it is necessary to log in to the EMu server as user **emu**. The tool is used to perform a number of activities:

- print out suitable range buckets for examination
- install EMu Registry entries to be used when updating the indexes
- print a table of data distribution allowing manual configuration

The `emurangeupdate` usage message is:

```
Usage: emurangeupdate [-dip] [-qrv] [-mmin:max] [-nrecords] [[dbname][:column] ...]
```

where:

```
        -d              use distribution based ranges [default]
        -i              use interval based ranges
        -mmin:max       minimum and maximum number of buckets to use [6:39]
        -nrecords       records per bucket for distribution ranges [5000]
        -p              use partition based ranges
        -q              quiet mode, do not output progress
        -r              update range Registry entries
        -v              output data distribution table
```

It is possible to analyse anything from:

- a single column in one table
- to all columns in a table
- to a single column in all tables
- to all columns in all tables

The format used to specify a column for analysis is `dbname:column`, where `dbname` is the name of the table to be analysed and `column` is the name of the column. The following combinations are allowed:

- `dbname` - all columns in the table `dbname` will be analysed
- `:column` - column `column` in all tables will be analysed
- `dbname:column` - column `column` in table `dbname` will be analysed

Any number of the above entries may be supplied to `emurangeupdate`. If an entry is not given, all columns in all tables are examined.

The default action is for `emurangeupdate` to print out suitable range buckets after examining the data. The following is typical output after analysing the *Date Modified* field in the Parties table:

```
emurangeupdate eparties:AdmDateModified
Processing eparties...
        Determining range columns...
        Checking registry entries...
        Exporting range data...
        Processing AdmDateModified...
                Range Buckets (distribution)
                ============
                2000-11-22
                2001-5-10
                2003-7-22
                2003-8-6
                2005-10-3
                2006-2-21
```

As you can see the output contains recommended range bucket values. These values could be used with the `Range Buckets` Registry entry to set the range buckets for the *Date Modified* field. The required Registry entry would be:

```
System|Setting|Table|eparties|Range
Buckets|AdmDateModified|2000-11-22;2001-05-10;2003-07-22;200
3-08-06;2005-10-03;2006-02-21
```

In fact it is possible to have `emurangeupdate` add the Registry entry for you by specifying the `-r` option on the command line:

```
emurangeupdate -r eparties:AdmDateModified
Processing eparties...
        Determining range columns...
        Checking registry entries...
        Exporting range data...
        Processing AdmDateModified...
                Range Buckets (distribution)
                ============
                2000-11-22
                2001-5-10
                2003-7-22
                2003-8-6
                2005-10-3
                2006-2-21
        Registry entry updated...
```

If you want to perform some analysis of the data, you can use the `-v` option to have a data distribution table printed:

```
emurangeupdate -v eparties:AdmDateModified
Processing eparties...
        Determining range columns...
        Checking registry entries...
        Exporting range data...
        Processing AdmDateModified...
                Value                   Count
                =====                   =====
                2000:11:22               1507
                2001:5:10                   1
                2003:7:22                   2
                2003:8:6                    1
                2003:8:26                   1
                2003:9:4                    2
               ...
                2007:12:27                  1
                2008:1:3                    1
                Distinct                   74
                Total                    2328
                Range Buckets (distribution)
                ============
                2000-11-22
                2001-5-10
                2003-7-22
                2003-8-6
                2005-10-3
                2006-2-21
```

The `Value` column contains a sorted list of all values from the *Date Modified* field. The `Count` column indicates the number of occurrences of the value. At the end of the table the `Distinct` value provides the number of unique values and `Total` is the total number of values (including repeated values). With this information it is possible to perform some analysis (MS Excel may come in handy here!) and determine suitable range buckets.

Number of range buckets

When calculating the range bucket values another variable is the number of range buckets to use. The maximum number of buckets allowed is **39**. When determining the optimal number of buckets `emurangeupdate` uses two pieces of information. The first is the minimum number of buckets that may be used. The default value is **6**. You can use the `-mmin:max` option to alter the minimum and maximum number of buckets allocated. For example:

```
emurangeupdate -m2:15
```

will allocate a minimum of two and a maximum of fifteen buckets. To determine the number of buckets to use within this range `emurangeupdate` computes the number of values in the column (the `Total` value from the data distribution table). It then divides this number by the number of records per bucket value (default value of `5,000`) to give the number of buckets to allocate. For example, if a column has 40,000 values, 8 range buckets will be allocated. You can alter the number of records per bucket by specifying the `-nrecords` option. For example, if we use the following command:

```
emurangeupdate -m2:15 -n2000
```

and the column contains 40,000 values, fifteen buckets are allocated (40,000 / 2,000 = 20; however the maximum number of buckets allowed is fifteen via the `-m2:15` option).

There are two special cases concerning the allocation of buckets. The first is for columns that do not contain any values. In this case only **one** bucket is allocated. The bucket is used to provide fast searching since any range search specified will not match the indexes; however the indexes can be used as part of the search.

The second case is where a column is part of the server table but is not used in the EMu client. This can occur when clients decide to sub-class standard modules and remove columns they do not require. In this case zero range buckets are allocated.

The following output shows range buckets for an existing column without data and for a column not used by the client:

```
emurangeupdate eparties:BioMovementStartDate0 eparties:ImaEstimatedDate
Processing eparties...
        Determining range columns...
        Checking registry entries...
        Exporting range data...
        Processing BioMovementStartDate0...
                Range Buckets (distribution)
                ============
                1970-1-1
        Processing ImaEstimatedDate...
                Range Buckets (distribution)
                ============
```

Bucket selection methods

When determining what range buckets to use `emurangeupdate` provides three algorithms that offer different focuses on bucket allocation. These algorithms are known as the *distribution*, *interval* and *partition* methods.

Distribution method

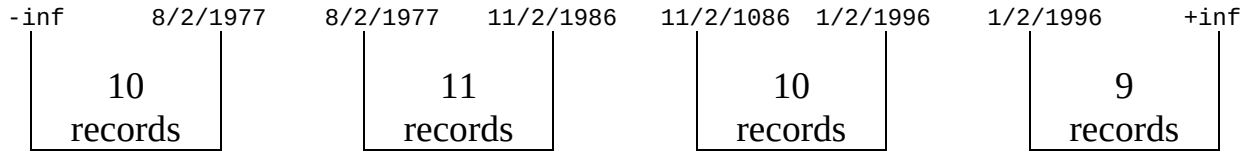
The *distribution* method tries to allocate the same number of values to each range bucket. The result is an even distribution of the values over the entire set of range bucket values. For example, if we have the following data distribution:

```
Value                Count
=====                =====
1959:1:20                1
1973:3:8                 1
1974:3:8                 1
1975:2:11                3
1977::                   3
1977:2:8                 1
1981::                   1
1982:11:8                1
1983:4:12                3
1983:5:9                 1
1984:9:11                1
1985::                   1
1985:6:12                1
1986:2:11                1
1986:6:17                1
1986:9:                  1
1988:6:28                2
1989:1:                  2
1995:6:6                 1
1995:9:                  1
1996:2:                  1
1996:2:28                4
1996:4:3                 1
1996:9:17                1
1997:8:20                1
2003:10:17               1
Distinct                26
Total                   37
```

we get the following range buckets calculated:

```
Range Buckets (distribution)
=============
1977-2-8
1986-2-11
1996-2-1
```

The graphic below shows the values per bucket:



The *distribution* method provides optimal searching where the query ranges are not known in advance and where a reasonable spread of query ranges is expected. As equal numbers of records are placed in each range bucket the index will generally provide reasonable performance for any given query. The distribution method is the default method used by `emurangeupdate`. The `-d` option may be used to select this ranging method.

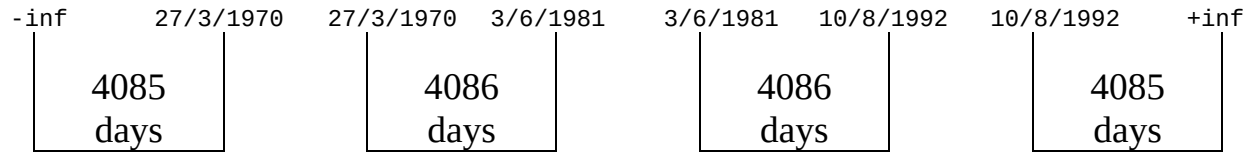
Interval method

The *interval* method takes a different approach to the *distribution* method. Rather than ensuring that an equal number of records is allocated to each range bucket, the *interval* method generates equal intervals between each bucket value. It does this by taking the difference between the minimum value and the maximum value and apportioning it between

the specified number of bucket intervals. So for the data distribution provided for the *distribution* method the generated intervals are:

```
Range Buckets (interval)
=============
1970-3-27
1981-6-3
1992-8-10
```

The graphic below gives the intervals:



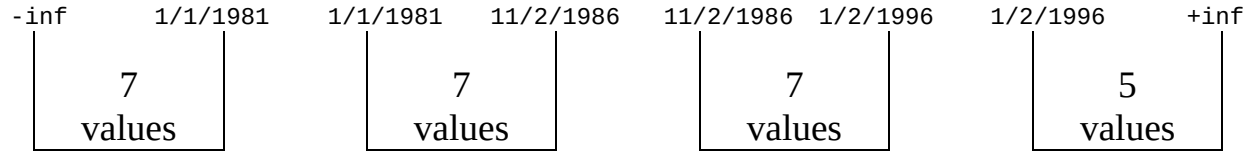
The advantage of the *interval* method is that it gives a decent set of range buckets when data is not available for a given field (generally because it has not been entered or imported). It also provides buckets that may be more in line with query ranges that correspond to fixed intervals (e.g. year ranges for date searches, hour intervals for time searches, etc.) and so gives better performance where query range intervals are commonly used. Interval ranges can be generated by specifying the `-i` option to `emurangeupdate`.

Partition method

The *partition* method is similar to the *distribution* method, however rather than ensuring that equal numbers of records are in each range bucket it ensures that equal numbers of *unique* values are in each range bucket (if ten values are the same, they are distributed as a single value). For the data distribution provided for the *distribution* method the generated partitions are:

```
Range Buckets (partition)
=============
1981-1-1
1986-2-11
1996-2-1
```

The graphic below shows the number of distinct values for each range bucket:



The *partition* method is useful when a good distribution of buckets is required that takes into account the values that have already been entered without giving undue influence to the weighting of each value (as is the case with the *distribution* method). In general this method will provide good "all-round" performance as it gives sensible range intervals based on the data already entered. Partition ranges can be generated by specifying the `-p` option to `emurangeupdate`.

Configuring emurangeupdate

When `emurangeupdate` is invoked it uses the EMu Registry to look for hints as to how many range buckets should be allocated and what type of distribution method should be used for any given column. Administrators may set Registry entries that can enable range searching on columns that do not have range support currently. It is also possible to disable range searching if a column does not require this type of indexing.

The format of the Registry entry is:

```
System|Setting|Table|table|Range Index|colname;...
```

where *`table`* is the name of the module containing the columns to be set. The *`colname`* setting is a list of semi-colon separated column names, listing the columns for which range searching is to be enabled (cf. `Null Index` and `Partial Index` Registry entries). It is possible to provide hints about the range index for the column by appending to the column name the number of range buckets required and the distribution method to be used. The format of the entry is:

```
colname=number:method
```

where *`number`* is the number of range buckets to allocate for the given *`colname`* and *`method`* is the distribution method to use. Allowable *`method`* values are:

- distribution
- interval
- partition

corresponding to the methods described above. A *`number`* setting of zero (0) will disable range indexing for the given field.

For example, if the *Date Modified* field is searched regularly using fairly small query ranges (e.g. monthly), you may want to increase the number of range buckets allocated and use the *partition* distribution method. You may also want to disable range indexing for the *Time Modified* field as users do not use it for range based searches (note that disabling range indexing **does not** mean you cannot perform range searches on a column, it just means that the search cannot use any indexing information to provide faster searching). The following Registry entry can be used to reflect these changes:

```
System|Setting|Table|eparties|Range
Index|AdmDateModified=15:partition;AdmTimeModified=0
```

Notice how all range column settings are specified in the one Registry entry. While this may seem confusing it is consistent with the `Null Index`, `Partial Index`, `Stem Index` and `Phonetic Index` Registry entries.

If you only want to change the distribution method for a column, you need not specify the number of buckets. For example, the entry:

```
System|Setting|Table|eparties|Range
Index|AdmDateModified=:partition
```

will ensure the *partition* method is used for the *Date Modified* field.

System Maintenance

The `emurangeupdate` utility has been designed not only to suggest suitable range buckets for range indexing, but also to update/create Registry entries that will result in the new range buckets being implemented the next time the database indexes are rebuilt (generally on the weekend).

Using the `-r` option with `emurangeupdate` will result in the creation of `Range Buckets` Registry entries reflecting the suggestions made by the utility. The Registry entries created differ slightly from the standard Registry entry in that the list of bucket values is prefixed with the string `auto:`. The string is used to indicate that the entry was made by `emurangeupdate` rather than by a System Administrator. Only Registry entries with the `auto:` prefix are updated by `emurangeupdate`; all other entries (that is, those created by other means) are not changed. This means that System Administrators may add their own entries manually and they will be preserved by `emurangeupdate`.

It is recommended that `emurangeudpate` is run on a regular basis to ensure that optimal range searching is available. The easiest way to provide automated configurations is to add an entry to user **emu**'s crontab (via **emucron**), similar to:

```
#
#  Calculate range buckets (first day of each month)
#
0 0 1 * * ${EMUPATH}/bin/emurun emurangeupdate -r 2>&1 | ${EMUPATH}/bin/emu
run emulogger -t "KE EMu Range Update Report" reindex
```

You may consider adding other options (e.g. `-m3:39` or `-p`) as required.