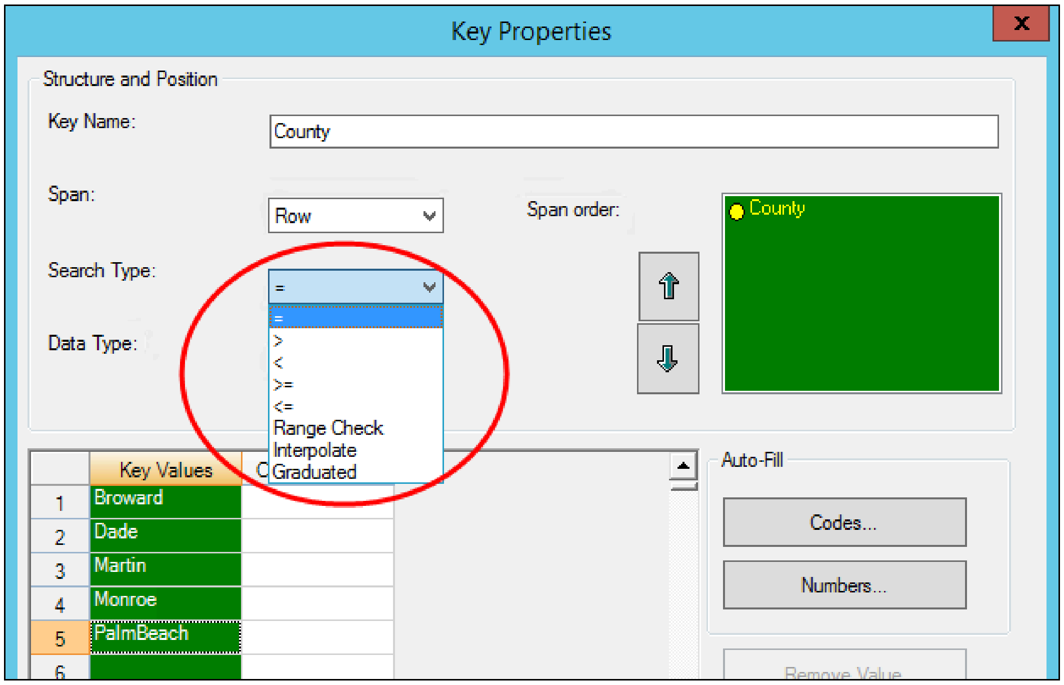
SEARCH TYPES

Overview

This section provides information on search types within Tables view.



Key Search Types

The **Search Type** of each key controls how the data is returned for each lookup made into the table. When a key is being searched, each key value is compared against the search value to find a match according to the search type being used. The default search method is = (equal).

Comparative Search Types

The simplest search types are the comparative search types:

- =
- < (less-than)
- > (greater-than)
- <= (less-than or equal to)
- >= (greater-than or equal to)

The comparative types search down a column until the comparison of the key value and the search value is true. The data type of the key is important because the comparison treats the values in the key list according to data type, comparing numbers, strings, or dates. When the less-than or greater-than types are used it is important to ensure the key value order make sense as a top-down search, from the most restrictive value to the least restrictive value.

Range Check Search Types

The Range Check search type searches the key to find if the search value is in any part of the range specified by each key value. The Range Check search type uses the hyphen (-) and the comma (,) to specify range values. A non-range key value can use either the less-than sign (<) or greater-than sign (>) to indicate the search value can be either less than or equal to, or greater than or equal to, the key value. Any combination of comma-delimited values and range values can be entered for each key item.

A hyphen indicates inclusive range (low to high) and a comma separates individual values. When both sides of a range phrase are numeric, the range is checked numerically. When non-numeric values are specified as a range, the comparison is done alphabetically. Each comma-delimited item is checked for enumeration using the same numeric or string logic. The following table provides examples of appropriate Range Check syntax.

KEY VALUE	VALID RESULTS
`<100`	100 or less
`12,16,20-30`	12, 16, or any value from 20 to 30 inclusive
`23-40, 50-61, >99`	23 to 40, or 50 to 61, or 99 and greater

Each Additional Support

Some tables list amounts for the most common values that do not fit into the interpolation process and then state something such as *for each additional 1000, add 5*. To support this kind of lookup, table keys need a method to indicate an *each additional* amount. Before the final key value in a list, place the text *[eachadd]* to indicate the value for an *each additional* phrase. An example of such a table appears below.

KEY	VALUE
`1-10000`	10
`10001-20000`	22
`20001-30000`	35
`[eachadd]5000`	5

Results of lookups into this table would include:

- If the search value is 15000, the returned lookup value would be 22.
- If the search value is 41000, the returned lookup value would be the next-to-last amount (35) plus the appropriate number of *each additional* amounts, computed by the number of increments between the request key and the highest defined key value. In this case that would be (41000 - 30000) / 5000 which is 3 increments, so the returned value for this example would be 50 (35 + (3 * 5)).

Interpolate Search Types

The Interpolate search type performs interpolation and extrapolation using key values. There can only be one key in a table that has the search type of Interpolate. There can be other keys, but only one can be an Interpolate key. The key values of an Interpolate search type must be numeric and the numeric order of the keys must be in ascending order. To enable extrapolation, the last key value should be the *each* *additional* amount, and must be smaller than the previous key value. The fact that the last key value is smaller than the previous value is the indicator to the system that extrapolation is desired of higher values than the next to the last value.

The system automatically adds an asterisk (*) after each additional key values as an indicator.

The interpolation/extrapolation lookup works by searching the key values to find the two entries that are closest to the search value. If the two key values are one lower and one higher, then the returned lookup value is the interpolated value of the associated data in the table of these two keys. If the key is set up to extrapolate (has an *each additional* value) and the search value is greater than the largest key value, the returned value is the extrapolated value.

The following table is defined with a single key with a search type of Interpolate.

KEY	VALUE
`0`	10
`20`	15
`30`	20
`40`	25
`50`	30
`60`	35
`70`	40
`80`	45
`10*`	5

Results of lookups into this table would include:

- If the search value is 34, the low key would be 30 and the high key would be 40. The interpolation between the two associated data values of 20 and 25 would result in the lookup returning 22.
- If the search value is 71, the low key would be 70 and the high key would be 80. The interpolation between the two associated data values of 40 and 45 would result in the lookup returning 40.5.
- If the search value is 102, the low key would be 80 and the high key would be 10. This example would use extrapolation to determine the value. The extrapolation would result in the lookup returning 56.

To prevent a lookup from returning a value with many decimal places, you can set a precision, which is the number of decimal places to round. For example, if you set a precision of 2, the lookup returns a value that has only two decimal places (9.99). Interpolation precision is only valid for table lookups that use the Interpolate search type.

Graduated Search Types

The Graduated search type is a special search type used for tables whose keys have a cumulative purpose. There can only be one key in a table that has the search type of Graduated. There can be other keys, but only one can be a Graduated key. The data in a Graduated table identifies a base value as the first key value and then each ensuing key value identifies an additional amount as either a factor or additive value to apply to the base. The returned lookup value is the cumulative total of all key values that apply to the search value.

- The Graduated search type uses a specific syntax for each key item that defines how the key is to be processed. The rules of the key values for the Graduated type are separated into the first key value, the last key value and the interior key values:
- **First key value** — Identifies the base amount. The data associated with the first key is used as the starting value. The numeric value of the first key defines the included amount, or the first threshold amount (for instance, the first 200 dollars). The syntax of the first key value is:

  `First` *IncludedValue*
- **Interior key values** — Identify the graduated steps of the value. The data associated with these keys are the additive amounts that are applied to the base amount. The numeric value of the interior keys identifies the increment amount (for instance, the next 100 dollars). Key values can also be defined with an optional *per* amount by including a comma and the *per* amount. If there is no per amount, the entire data value of the key is added to the base when the search value applies. If there is a *per* amount specified, the data value of the key is multiplied by the fractional number of per amounts the search value requires, interpolating the *per* value.

  All interim calculations are rounded to the dollar.

  The syntax of interior key values is:

  `Next` *IncrementAmount*, *PerAmount*
- **Last key value** — Identifies the additive amount if the search value is greater than all the graduated steps. Like the interior values, the last key value can have an optional *per* amount specified, and behaves the same as the interior keys. The syntax for the last key value is:

  `Excess` *IncrementAmount*, *PerAmount*

The following table is defined with a single key with a search type of Graduated.

KEY	VALUE
First 200	80
Next 100,25	12
Next 300,50	4
Excess, 100	5

Results of lookups into this table would include:

- If the search value is 125, it is less than or equal to the first amount, so the returned lookup value would be 80.
- If the search value is 350, the returned lookup value would be 132. This value is the base of 80, plus 48 (4 * 12) from the *Next 100,25* key, plus 4 for the first 50 from the *Next 300,50* key.
- If the search value is 650, the returned lookup value would be 155. This value is the base of 80, plus 48 (4 * 12) from the *Next 100,25* key, plus 24 (6 * 4) from the *Next 300,50* key, plus 3 from the fractional part of the first 100 from the *Excess, 100* key.