Description of the different ranges in Dexterity for Microsoft Dynamics GP

Partner Only Article

Article ID : 922057
Last Review :
Revision : 1.0

INTRODUCTION

This article describes the different ranges in Dexterity for Microsoft Dynamics GP. This article also describes how to use some of these ranges.

MORE INFORMATION

Understanding Dexterity ranges

Dexterity uses the concept of ranges to limit the records that can be accessed in a table. There are two methods that you can use to create ranges:

- You can use an indexed range that limits a table.
- You can use a Range Where clause.

You can use a Range Where clause when the index does not contain the appropriate fields for your purpose. You can use an indexed range together with a range that you created by using a Range Where clause. Or, you can use either type of range independently.

For more information about the Range Where clause, click the following article number to view the article in the Microsoft Knowledge Base:

> 910129 How to write "Passthrough" SQL statements and "Range Where" clauses in Microsoft Great Plains Dexterity

Indexed ranges

For example, you can use an indexed range to find an invoice. An invoice contains information from a Header table and from a Line table. In this example, you might want to limit the lines that are visible from the Line table when a user examines a specific record from the Header table.

A range is created on a single instance of a table buffer. For a range to function after you define the range, you must perform all table actions by using the same index on which the range was defined. If you need to use the same range in a function or in a procedure, you must pass the table buffer as an *inout* table parameter.

To create a range, follow these steps.

Note These steps are for an invoice.

1. Clear any existing range for the table.
2. Set the key fields for the start record in the table buffer.
3. Apply the start of the range.
4. Set the key fields for the end record in the table buffer.
5. Apply the end of the range.

The following code sample shows these steps.

```
range clear table LINE; {Step 1}

clear table LINE;
'Invoice Number' of table LINE = 'Invoice Number' of table HDR; {Step 2}
range start table LINE by number 1; {Step 3}

fill table LINE;
'Invoice Number' of table LINE = 'Invoice Number' of table HDR; {Step 4}
range end table LINE by number 1; {Step 5}
```

The code in this range-setting example may differ from the range-setting code that you expect to see. However,

this code does not have to be changed when you add more fields because this code automatically handles additional fields that you add as new segments to the key. The code does not have to be changed because the code uses the "clear table" statement and the "fill table" statement. Therefore, the code does not have to specifically reference the other fields in the key.

Well-defined ranges

When you set a range, you may receive unexpected results. This usually occurs when a range is not well defined. When Dexterity was used with ISAM databases, the ranges that were created were always inclusive. You defined the start record in a key and the end record in a key. All the records between the start record and the end record were included in the range.

By default, ranges are exclusive in Microsoft SQL Server databases. Ranges are implemented by using an SQL Where clause. Each field must meet the limits that are defined. Consider the following scenario in which you have a Line table that contains the following information.

Invoice Number	Sequence Number
1	1
1	2
1	3
2	1
2	2

You set up the following range on the Line table.

- Start: Invoice Number = 1, Sequence Number = 1
- End: Invoice Number = 2, Sequence Number = 2

An ISAM database would produce all five records in the inclusive range. A SQL Server database would produce four records in the exclusive range because the following conditions are true:

- Record "1, 3" does not have an invoice number between 1 and 2.
- Record "1, 3" does not have a sequence number between 1 and 2.

A well-defined range is a range in which the inclusive range and the exclusive range always return the same record set. A well-defined range must meet the following criteria.

Note Dexterity evaluates the criteria from the first key field to the last key field in the key that you are using.

1. You can have any number of fields or no fields in which the start value and the end value are the same.
2. You can have one field or no fields in which the start value and the end value are different.
3. If any values remain in the table, you must clear the fields by using the "clear table" statement, and then you must use the "fill table" statement to populate all the values.

The range in this example is not well defined because the start value and the end value for the two fields are different. Therefore, the range does not meet the second criterion. To make the range well formed, you must set up the following range on the Line table:

- Start: Invoice Number = 1, Sequence Number = C
- End: Invoice Number = 2, Sequence Number = F

Descending keys and ranges

If you have a descending segment in your key, the descending segment in your key must be treated in reverse. When you set a start value and an end value, the start value must be the larger value. The end value must be the lower value. If you plan to clear and then fill this field in the key, you must actually fill for the start of the range and clear for the end of the range.

REFERENCES

For more information about indexed ranges or about the Range Where clause, see the Dexterity Help file.

APPLIES TO

- Dexterity, when used with:
  Microsoft Dynamics GP 9.0
  Microsoft Business Solutions–Great Plains 8.0
  Microsoft Business Solutions–Great Plains 7.5
  Microsoft Great Plains eEnterprise 7.0
  Microsoft Great Plains Dynamics 7.0

Keywords: kbexpertiseadvanced kbexpertiseinter kbmbspartner kbhowto kbinfo kbmbsmigrate KB922057