Xlate Function

Description

Returns data from the specified element in a table that has not been opened. Typically used in symbolic fields to bring back column data from a related table.

Syntax

status = **Xlate** *(tablename, key, field, control [,expression])*

Parameters

The Xlate function accepts arguments for the following parameters.

Parameter	Description
Tablename	The name of the table from which the record is to be read. *tablename* must yield the literal name of a table. It is not a table variable (the result of an Open statement). The table must be currently attached.
Key	The key or keys of the row(s) containing the column values to be returned. Can be multiple keys. Separate multiple keys with a system delimiter. Each key will return the value of that row in the specified column. These values will be separated by whatever system delimiter you use to separate the keys during the call to Xlate. Any values (@VM-delimited) or subvalues (@SVM-delimited) will be delimited with their own system delimiters.
Field	The number of the field or name of the field in quotes (example: "COMPANY") to be extracted. If a null is specified, the entire record is returned. If a zero is specified, the key is returned.
Control	Must be either of the literal values X or C. These keywords indicate what the function should return if the record does not exist. An X indicates that a null should be returned, while a C will cause Xlate to return the key value.
Expression	Indicates the number of times the system delimiters will be converted to the next lower delimiter. The default is 1. The column returned from the row can contain system delimiters. If the column is a multi-valued string (@VM-delimited), it becomes a subvalue string (@SVM-delimited) after one conversion. This feature becomes important when key yields a multi-valued string. After the conversion, each value contains a string of subvalues. A text mark (ASCII character 251) is the lowest delimiter.

Let's presume you pass three keys to Xlate, and you specify the contents of the code column. The value of key could be:

```
key1:@FM:key2:@FM:key3
```

where you use @FM to delimit the keys. If the code column contains multi-values (as code in sample_invoices does), then the results could be:

```
mv1-1:@VM:mv1-2:@FM:mv2-1:@VM:mv2-2:@FM:mv3-1:@VM:mv3-2
```

@FM separates values from each column.

If the keys had been passed using any other system delimiter than @FM, the column values would have been returned separated by that delimiter (including @VM).

In the case of @VM, all returned values would be then delimited by @VM, and you could no longer tell which values belonged to which field.

See also

Extract

Example

```
/* The following stored procedure function returns multi-value data from the two rows specified in the Xlate
call.
In its code column, sample_invoices has multi-value data, so this is returned in RetVal. */

Compile Function test_Xlate (void)

tablename = "SAMPLE_INVOICES"     ;* previously unopened table
keys  = 8:@FM:11           ;* two rows to access
field  = "CODE"            ;* column to access
control  = "X"             ;* return null, if not found

RetVal = Xlate(tablename, keys, field, control)


Return RetVal
```