# Array

Firebird/InterBase® allows a column to be defined as an array of elements, i.e. data information can be stored in so-called arrays. An array is a range of values determined by setting a lower and an upper limit. An array consists of any amount of information that can be split into different dimensions. The array can be managed as a whole, as a series of elements in one dimension of the array, or as individual elements.

Arrays should be used with caution. Database normalization usually supplies an alternative format for storing such data, so that normal table structures are just as suitable, and also preferable. There are however occasionally exceptions, for example for measurement value logging, when arrays are the preferred option.

The array data type is used relatively seldom, as it is not very simple to process, and does not really conform to the typical demands of an SQL database (usually one or more detail tables would be created, and not an array).

Arrays can be declared as a domain or directly in the table definition following the data type definition. Array data can be of any type except blob. Between 1 and 16 dimensions can be specified; each dimension can store as many elements as can be fitted into the database. The values are stored as a blob and are therefore almost unlimited in scope.

The only difference compared to the normal data type definition is the specification of the dimensions in square brackets, each dimension being separated by commas. By default, the lower bounds ID number is 1 and the upper bounds ID number is the maximum of that dimension. Alternate bounds IDs can be specified in place of the array size by separating them with a colon. For example, an array with 5 measurements with 2 dimensions starting at the default value 1 is defined as follows:

```
[2,5]
```

Counting begins at 1 and ends at the value entered by the user. In this case 2 x 5 = 10 measurements can be logged. If counting is to begin at, for example, 0, the array definition is as follows:

```
[0:2, 0:5]
```

# One-dimensional arrays

*Definition*: NAME DATATYPE [LOWER_DIMENSION:UPPER_DIMENSION]

*Example*: LANGUAGE_REQ VARCHAR(15) [1:5]

In this field 5 data entries of the VARCHAR(15) type can be stored. LANGUAGE_REQ[1] up to LANGUAGE_REQ[5] can be accessed.

# Multi-dimensional arrays

Definition: NAME DATATYPE [LOWER_DIMENSION1:UPPER_DIMENSION1] [LOWER_DIMENSION2:UPPER_DIMENSION2]

Example: DAILY_MEASUREMENTS NUMERIC(18,2) [1:24][1:365] When using arrays, it is important to be aware of the advantages and limitations.

# Advantages of arrays

1. InterBase® operations can be performed upon the total data type as a single element. Alternatively operations can be executed on part of an array only for certain values of a dimension. An array can also be broken down into each single element.
2. Following operations are supported:

- SELECT statement from array data.
- Insertion of data in an array.
- Updating data in an array slice.
- Selecting data from an array slice.
- Examination of an array element in a SELECT statement.

# Array limitations

1. A user-defined function can only access one element in an array.

2. The following operations are not supported:

- Dynamically referencing array dimensions using SQL statements.
- Inserting data into an array slice.
- Setting individual array elements to null.
- Using aggregate functions such an MIN(), MAX(), SUM(), AVG() and COUNT() on arrays.
- Referencing an array in the GROUP BY clause in a SELECT query.
- Creating a view, which selects from array slices.

3. The data stored in this way cannot be selected per index; each query always accesses the fields unindexed.