Loading and Calculating Data

Creating a dimension from a database column

To read in the *products* dimension from a table products with fields from the ProductCode column:

```
LOAD WRITE
   READ TABLE Products
   FORMAT "column directed"
   CREATE DIMENSION products COLUMN &
   ProductCode VALUE
END LOAD
```

Creating a hierarchical dimension

To read from a table in the *products* dimension, products with fields from the *ProductCode* column and the Parents from the *Prodprnt* column:

```
LOAD WRITE
   FORMAT "column directed"
   CREATE DIMENSION products COLUMN ProductCode &
   HIERARCHICAL SUM ProdPrnt ORDER "normal scan"
   READ TABLE Products
END LOAD
```

To position the parents at the end of the dimension as the normal scan and to position the parents after the children use restart scan.

Adding new fields to dimension

```
LOAD WRITE
   FORMAT "column directed"
   ADD DIMENSION products COLUMN ProductCode &
   HIERARCHICAL SUM ProdPrnt ORDER "normal scan"
   READ TABLE Products
END LOAD
```

Loading time associated data

A dimension declared as a TIME dimension permits rules to reference the TIME fields using the SHIFT operator:

```
DIMENSION Weekly, TIME
   ...
END DIMENSION
```

Additionally, if a base date is defined and time offset applied to the fields, Holos will read rows from DATE columns and insert the values using the appropriate field:

```
DIMENSION Weekly, TIME, BASE DATE "5-Jan-1998"
   'WK01' "Week 1"  = 1Week
   'WK02' "Week 2"  = 1Week
   'WK03' "Week 3"  = 1Week
   ...
END DIMENSION
```

The database column must be a suitable date format and if many rows will insert values into the same fields, then the values will also need to be added.

```
LOAD WRITE
    FORMAT "column directed"
    WRITE STRUCTURE ty_store
    DIMENSION weekly COLUMN Salesdate VALUE DATE
    ....
    VALUES CREATE ADD COLUMN val_column
    READ TABLE datatable
END LOAD
```

Loading and Calculating Data (cont'd)

see also Help Load, Calculate

Writing to a structure from a database with a single values column

For any structure that is already loaded, the WRITE STRUCTURE statement specifies its name:

```
LOAD WRITE
   FORMAT "column directed"
   WRITE STRUCTURE ty_store
   DIMENSION ....
   ....
   VALUES CREATE COLUMN val_column
   READ ...
END LOAD
```

For a sparse structure, the CREATE keyword will ensure that cells are created before the value is inserted.

Creating a structure from a file

To create a structure from a file with fixed width columns of information:

```
LOAD
   FORMAT "Column directed"
   READ HOSTFILE"store.txt"
   COLUMN LOCATION = 1, WIDTH = 2, TYPE = TEXT
   COLUMN LOCATION = 5, WIDTH = 1, TYPE = TEXT
   COLUMN LOCATION = 10, WIDTH = 2, TYPE = NUMBER
   CREATE DIMENSION products COLUMN 1
   CREATE DIMENSION weekly COLUMN 2
   CREATE STRUCTURE ty_store
   VALUE COLUMN 3
END LOAD
```

If the file is comma separated do not specify the WIDTH in the column statement and specify the separator in the FORMAT clause:

```
LOAD
   FORMAT "Column directed" USING ","
   READ HOSTFILE"store.csv"
   COLUMN LOCATION = 1, TYPE = TEXT
   COLUMN LOCATION = 5, TYPE = TEXT
   COLUMN LOCATION = 10,TYPE = NUMBER
   CREATE DIMENSION products COLUMN 1
   CREATE DIMENSION weekly COLUMN 2
   CREATE STRUCTURE ty_store
   VALUE COLUMN 3
END LOAD
```

For help with quoting, see **HELP LOAD FORMAT CLAUSE.**

Structure Creation in the Load Block

The type of structure created by the LOAD block can be selected using the syntax below:

Private Memory	`CREATE STRUCTURE ty_store`
Non-sparse Disk	`CREATE SHARE STRUCTURE ty_store, "ty_store.shr"`
SQL structure	`CREATE ALIAS STRUCTURE ty_store`

Handling multiple values columns

Weekly data for weeks 1 to 3 in columns val1, val2, val3 in a table *new_table* can be loaded into a structure as follows:

```
LOAD WRITE
   FORMAT "column directed"
   WRITE STRUCTURE ty_store
   DIMENSION weekly FIELDS 'Wk01';'Wk02';'Wk03'
   ....
   VALUES CREATE COLUMNs val1,val2,val3
   READ new_table
END LOAD
```

Writing from a structure to a file

In order to write from a structure to a file, the load block iterates over the fields specified and writes out a record in the format specified:

```
 LOAD
   FORMAT "Column directed"
   READ STRUCTURE ty_store
   COLUMN LOCATION = 1, WIDTH = 2, TYPE = TEXT
   COLUMN LOCATION = 5, WIDTH = 1, TYPE = TEXT
   COLUMN LOCATION = 10, WIDTH = 2, TYPE = NUMBER
   DIMENSION products ITERATE products COLUMN 1
   DIMENSION weekly ITERATE * COLUMN 2
   VALUE COLUMN 3
   WRITE HOSTFILE "source.txt"
END LOAD
```

Consolidations and Calculations

A hierarchy field in a dimension is defined as a SUM, AVG or CALCULATE. For example:

```
DIMENSION b
   b1
   b2
   b3
   b4 = SUM  b1;b2
   b5 = CALCULATE <100> MIN (b2,b3)
END DIMENSION
```

Calculating values on the fly

```
DIMENSION b
   b1
   b2
   b3
   b4 = SUM  b1;b2
   b5 = CALCULATE <NO WRITE> MIN (b2,b3)
END DIMENSION
```

Partial Storage of Consolidated values

Use a prestore map to define which levels in a structure are stored and which ones are not stored but calculated when needed.

```
    CONSOLIDATE ty_store PRESTORE "2,4"
```

Consolidating structures

To consolidate a dense structure, ensure that the structure has a NO SPARSE attribute and then

```
CONSOLIDATE ty_store
```

There are different algorithms used when consolidating sparse and dense structures and the choice is partially dictated by the SPARSE attribute on the structure. To consolidate sparse structures, ensure that the structure has a SPARSE attribute, using `ALTER ty_store <SPARSE>`

Getting progress on the output

```
CONSOLIDATE ty_store LOG 100
```

The progress and algorithm used will be output to the command window.

Consolidating part of a structure

To consolidate only a known set of cells, the FROM clause to specify those cells. Only the totals affected by the known cells will then be reconsolidated. For example to consolidate this year's structure knowing that week41 sales figures have changed:

```
CONSOLIDATE ty_store FROM {'Sales','WK41',*,*}
```

Calculating models

The syntax for calculating follows the same rules as those for consolidating, except that you calculate a model not a structure. For example, to calculate the *ty_store* structure with rules from the *ty_store_rtb*:

```
MODEL store_mod ty_store,ty_store_rtb
CALCULATE store_mod
```

Loading and Calculating Data (cont'd)

see also Help Load, Calculate

Loading and Calculating Data (cont'd)

see also Help Load, Calculate

Note: The structure will by default be consolidated after the rules have been calculated.

Calculating a model without consolidating the structures

Apply a NO CONSOLIDATE attribute to the model to prevent the structures in the model being consolidated after the rules have been calculated.

```
MODEL store_mod <NO CONSOLIDATE> store, store_rtb
```