Building a GentiaDB Database

(ref: gdb05)

http://extranet.gentia.com

Global Hotline / Helpdesk -

ACCURACY

Every effort has been made to ensure the accuracy of the features and techniques presented in this publication. However, Gentia Software plc accepts no responsibility, and offers no warranty, whether expressed or implied, for the accuracy of this publication.

RESTRICTED RIGHTS

Contents

Building a GentiaDB Database

GENTIA™

Objectives

- To understand the concepts of GentiaDB
- To learn how to build models and load data
- To understand how to use the various components of GentiaDB
- To gain an appreciation of the efficiency issues

GENTIA™

The objectives will be achieved through the use of demonstrations, examples and practical exercises.

This course is aimed at developers involved in the building of Gentia applications, who have attended that *Introduction to Gentia* training course, or have equivalent experience.

Before we get started...

- Timings
  - End of the day
  - Lunch
  - Breaks
- Toilets
- Fire
- Messages
- Evaluation form

GENTIA™

Agenda

- Overview of GentiaDB
- Build a simple database
  - Focus on concepts and necessary steps
    - Define business model
    - Create base models
    - Load data & view
- A more complex example
  - To cover further features and issues
- Efficiency issues

GENTIA™

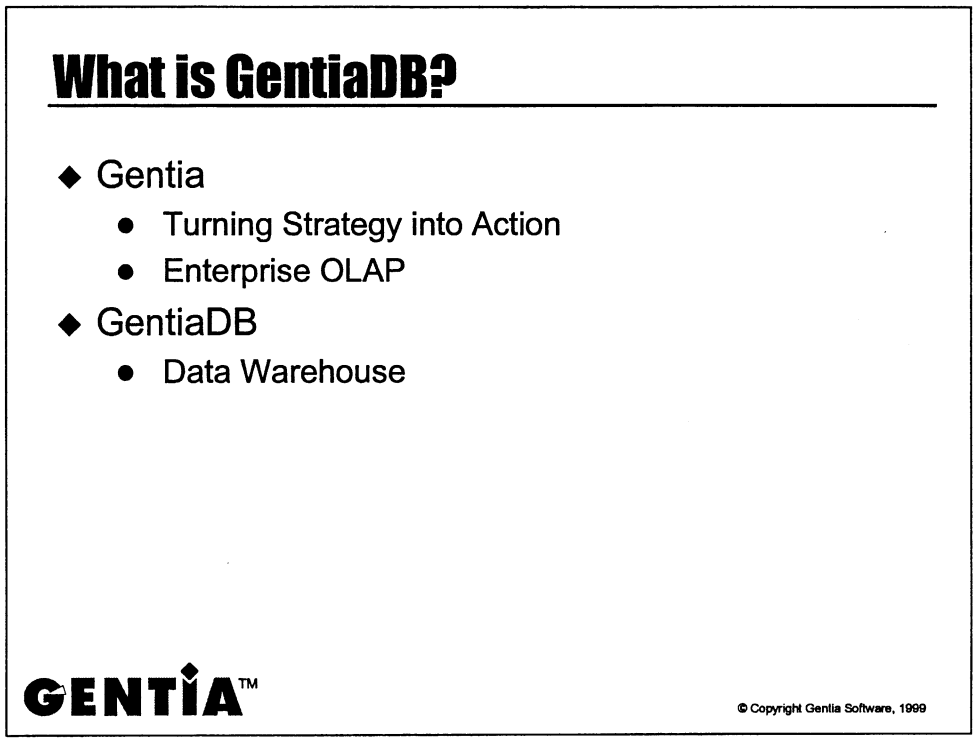
Gentia's Data Sources

- In addition to GentiaDB:
- Analytical Models (GADB)
  - Complex multi-dimensional modelling
  - Smaller than GentiaDB
- Scenarios
  - Ad-hoc analysis, forecasting, identifying the effect of change
  - Held in memory

GENTIA™

All three data sources can easily be utilised in a Gentia application as appropriate.

Analytical models are discussed in greater depth on the *Building Analytical Models & Scenarios.*



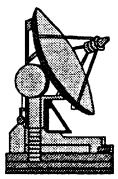
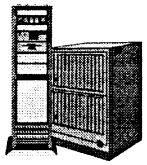
GentiaDB is a multi-dimensional database that extends database storage capacity from gigabytes to multiple terabytes. A wide range of warehouse issues are addressed by GentiaDB by providing extraction, cleansing and transformation tools, all accessed using a visual development interface. Combined with Gentia's advanced agent-based architecture, this provides the mechanism for automated and scheduled extraction, transformation and loading of data and metadata from production systems into GentiaDB.

What is Data Warehousing?

- Set of tools to query, analyse and present information
- Gives access to corporate data
- The data is consistent
- The data can be separated and combined by every possible means in the business

GENTIA™

Data Warehouse Stages

Extract	Cleanse / Transform	Store	Stage "OLAP"	Analyse Present
				*Ad-hoc* *Data mining* *Slice & dice* *EIS* 

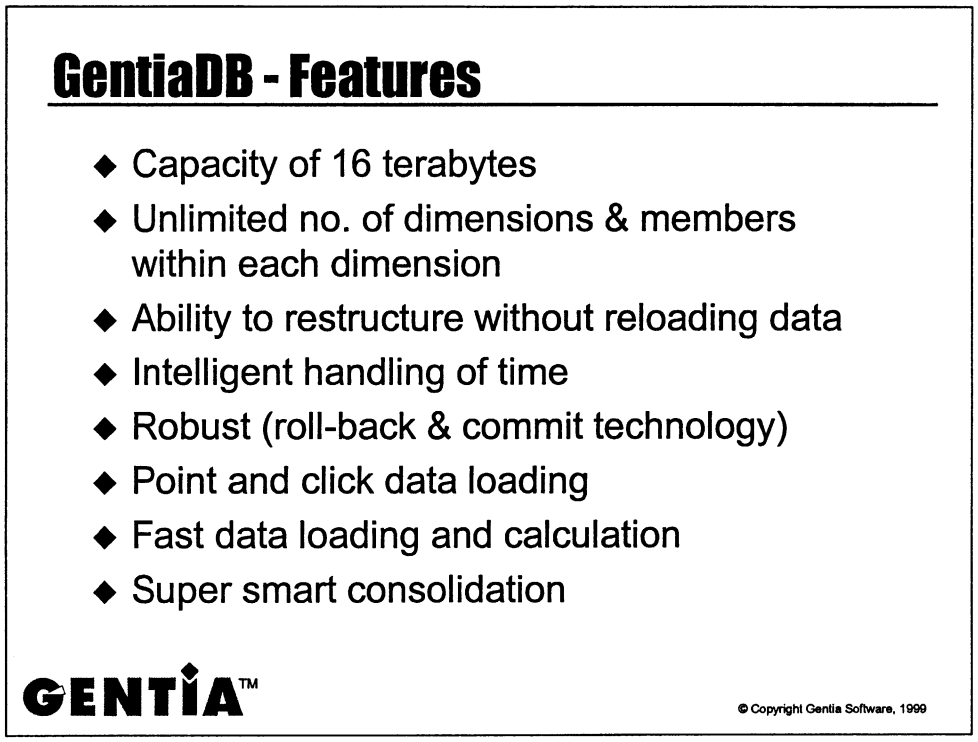
GENTIA - Enterprise OLAP & Decision Support

GENTIA™

Examples of Gentia Applications

- Financial reporting/analysis
  - Profitability analysis
  - Budgeting,
  - Planning
- Sales & Marketing
  - Promotion tracking
  - Customer profiling
- Customer Applications
  - Client satisfaction
  - Retention management
- Operations
  - Order fulfillment
  - Process Modelling

GENTIA™



GentiaDB has the following key features:

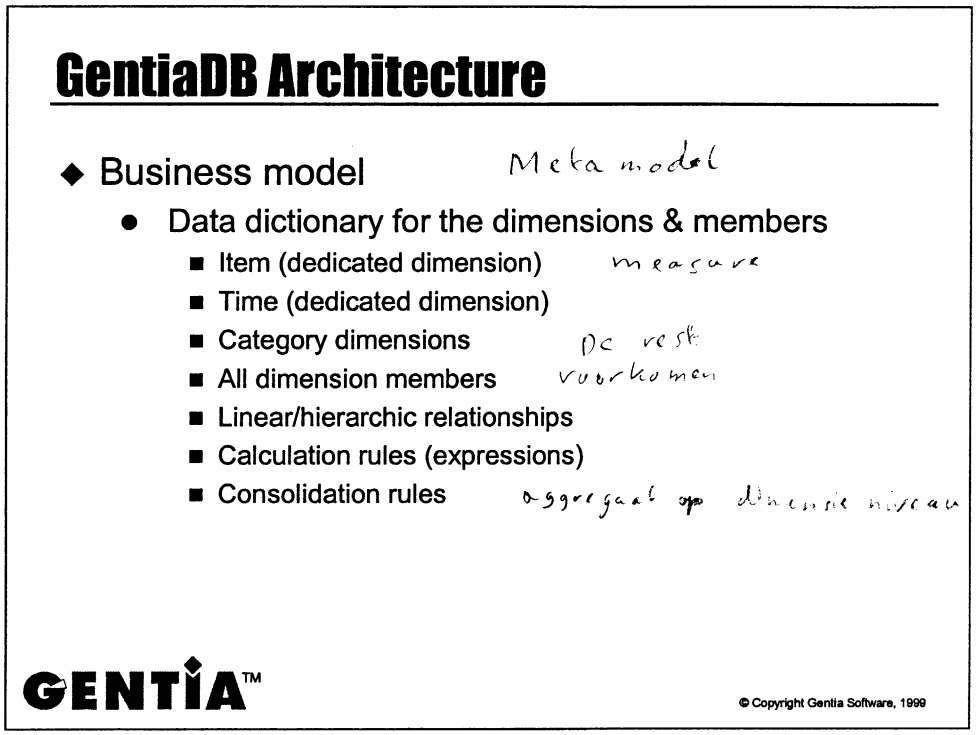
· **Huge data storage capabilities** - A GentiaDB database can be up to 16 terabytes in size.

· **Fast data loading and consolidation** - GentiaDB performs its fast consolidation by analysing the data being loaded with data that is already held in the database. GentiaDB then determines which cells in the database require adjusting or re-calculating. The other cells in the database are left untouched.

· **Intelligent consolidation** - The hierarchic consolidation is performed by a process of adjustments. A change to base level data triggers adjustment to its hierarchic parents. The effect is to increase efficiency because re-calculating parent level cells based on all their children's values would take longer to process.

· **Efficient model restructuring without reloading data** - Database restructuring is supported without the need to re-load the data or re-calculate and re-consolidate any cells which are not affected by the change. This restructuring may include adding, deleting, or moving members within a dimension. Also, old data can be removed from the database without reloading or reconsolidating.

· **Database Recovery** - GentiaDB constructs a list of change records during the load and consolidate process. The main database is kept separate from the change records while the update is in progress, but remains accessible. Only when the update has completed are the change records incorporated into the main database and committed. The result is a change over to the new integrated data which appears to be immediate. GentiaDB will not commit an incomplete set of change records, instead it will roll back to the secure state it was in before the update process began.

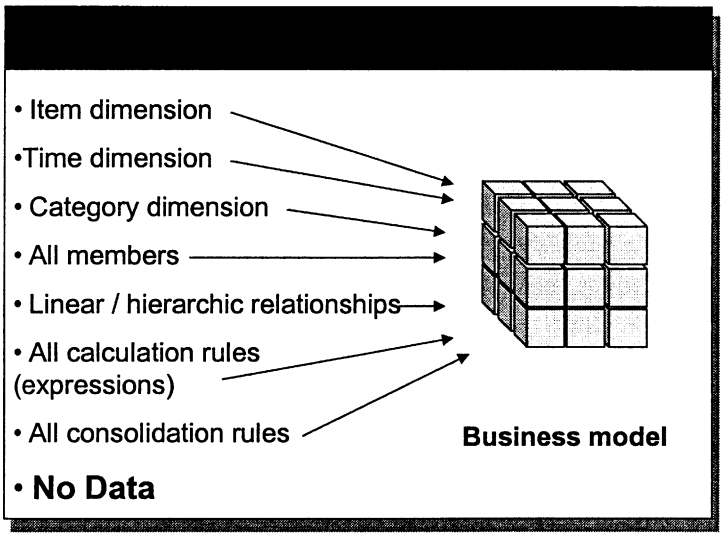


Your organisation can be split up into a number of dimensions, for example Time, Product and Customer. These dimensions will contain many members. Every dimension that your organisation is likely to need can be defined in a business model. This model gives you a Data Dictionary which is centrally held allowing easy maintenance.

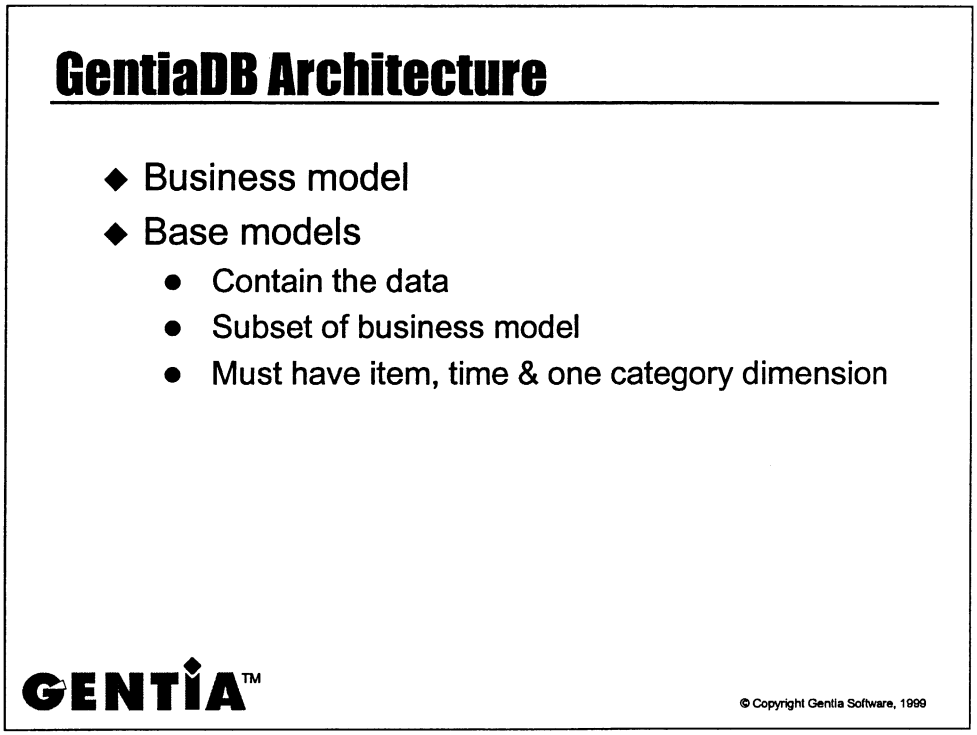
The business model is the framework into which you models that contain the data. It also holds the calculation rules for non-input members and hierarchical structure information. The model must have a Time and an Item dimension, the other dimensions are defined under the title of Category dimensions. But note that at no time is the business model used to hold any data.

Creating a business model is the first stage of building any application.

Business Models



GENTIA™



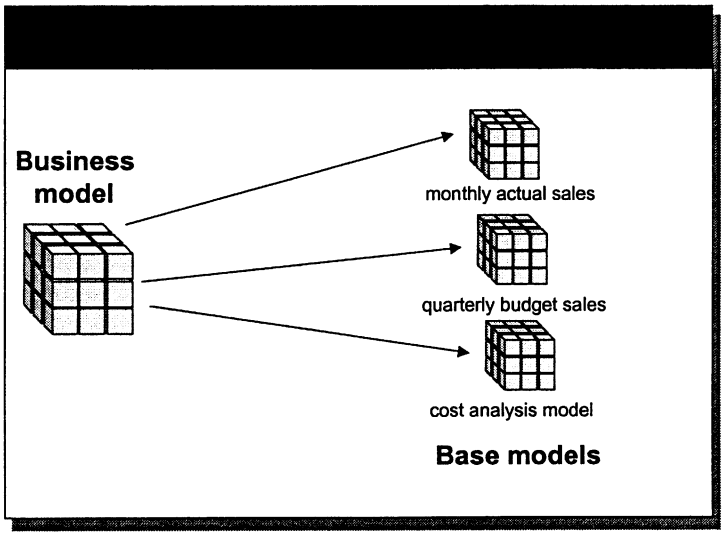
A base model is a sub-set of the business model. A base model must have at least a Time and Item dimension, after that you can add as many dimensions and members as you want from the business model.

Data is loaded into the base model once you have defined the dimensions and members for the base model.

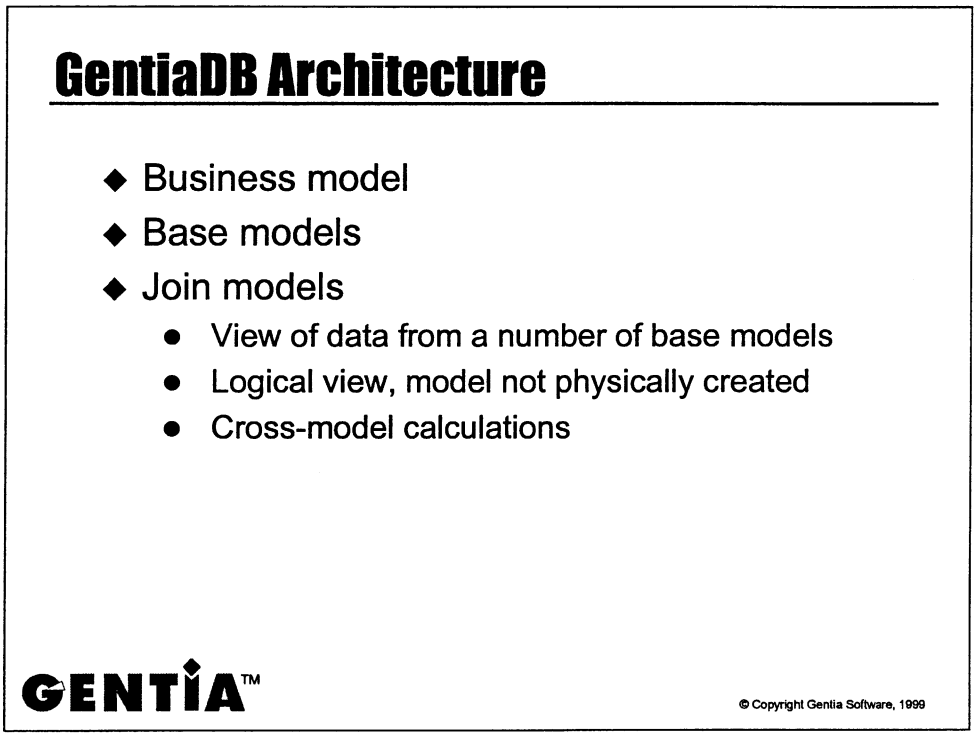
Different base models will therefore show different views of the business model. The advantage of this is that not all parts of your business are going to be interested in all parts of the business model, so a base model can show a user just the information they are interested in. Some dimensions will not directly relate to one another, for example a clothes product dimension will not relate to an electrical product dimension. A base model can be used to split a dimension or display different dimensions to show just the parts of the business model that a user wants to see.

Typically, there will be many base models within a GentiaDB application.

Base Models



GENTIA™



If you wanted to see data from two base models, you could merge the base models together into a join model, rather than having to create a new base model from scratch. This has the following advantages:

• The join model is not a physical model; it is merely a view of the data held in the base model and therefore does not take up a significant amount of disk space.

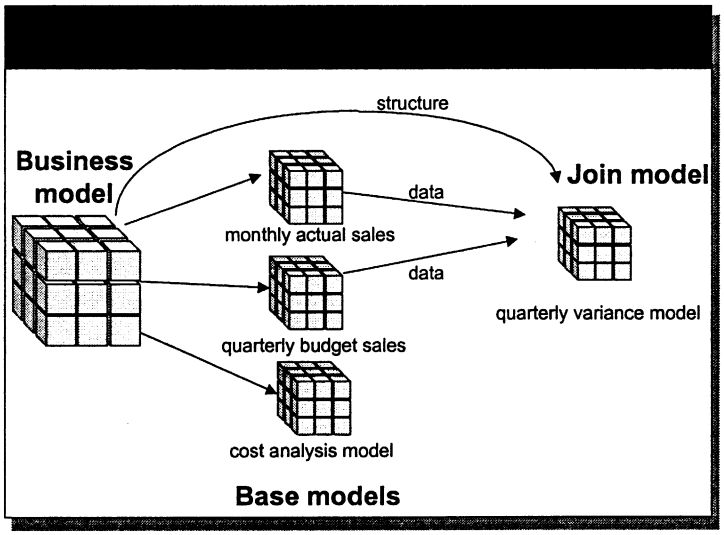
• Cross-model calculations can be carried out, which are performed in-flight and therefore not held in the database.

It is only the data from the base models that is used by the join model. The join model is defined from the business model in terms of its dimensions and members. The specified base model data is then viewed through the join model.

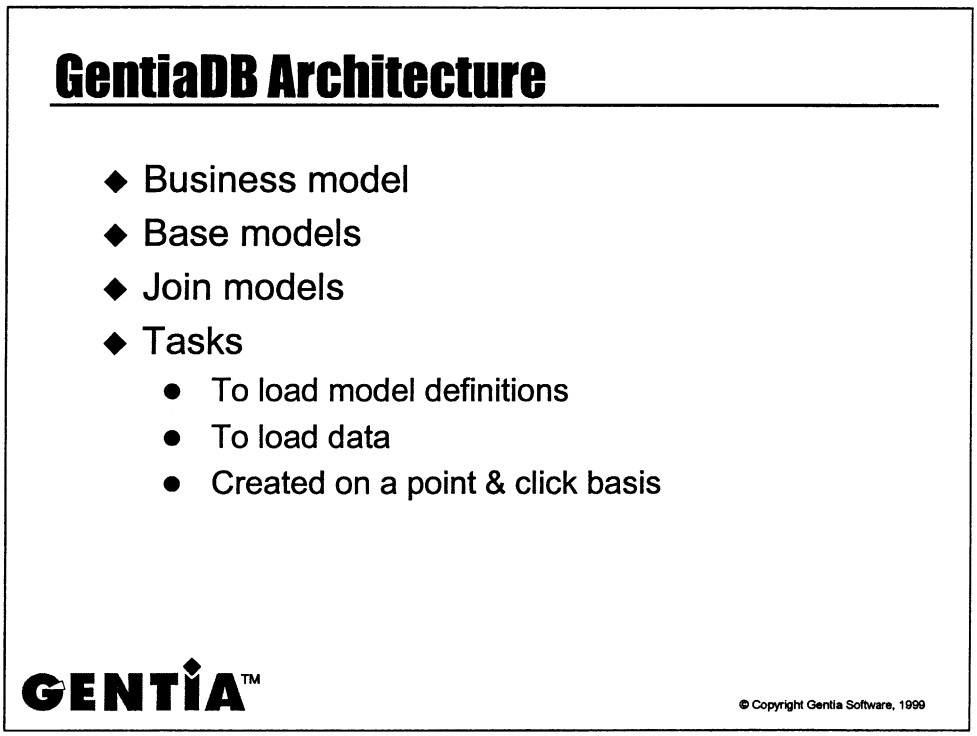
Building base and join models in this way is very effective in terms of data storage, since a 'relational' approach is adopted. Base models can be built, which are based upon your data in order to avoid sparsity.

As long as the base models share one or more common dimensions (other than Time and Item), they can be joined to form a join model. It should be remembered that only base models within the same business model can be joined.

Join Models



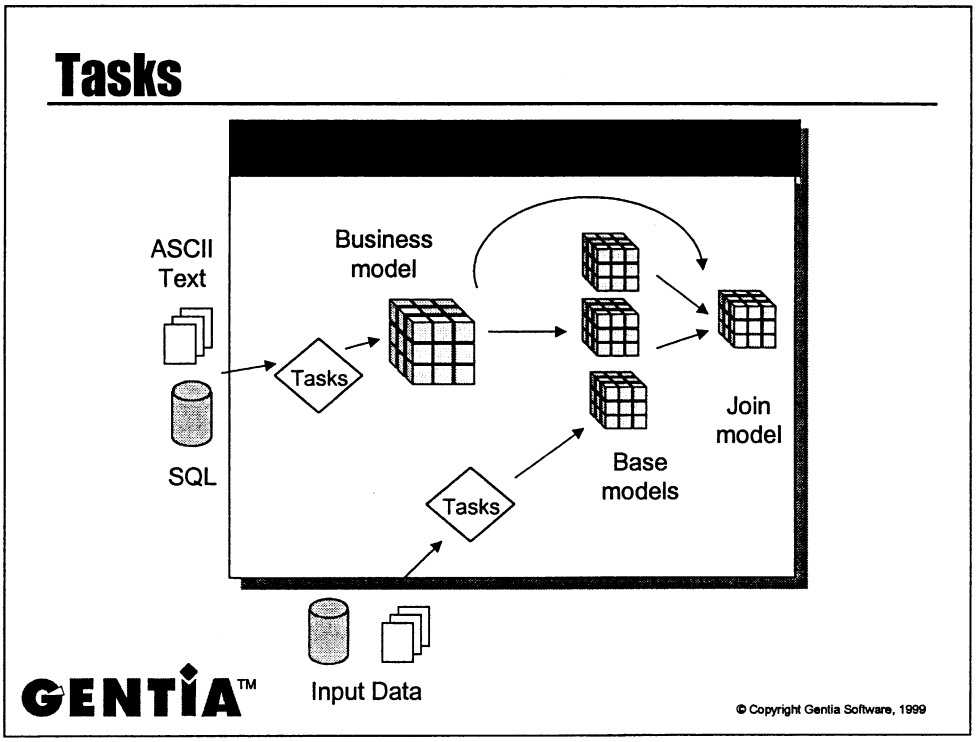
GENTIA™



Tasks are primarily used to perform 2 GentiaDB-related functions.

1. Defining dimension members in the business model.
2. Loading data into base models.

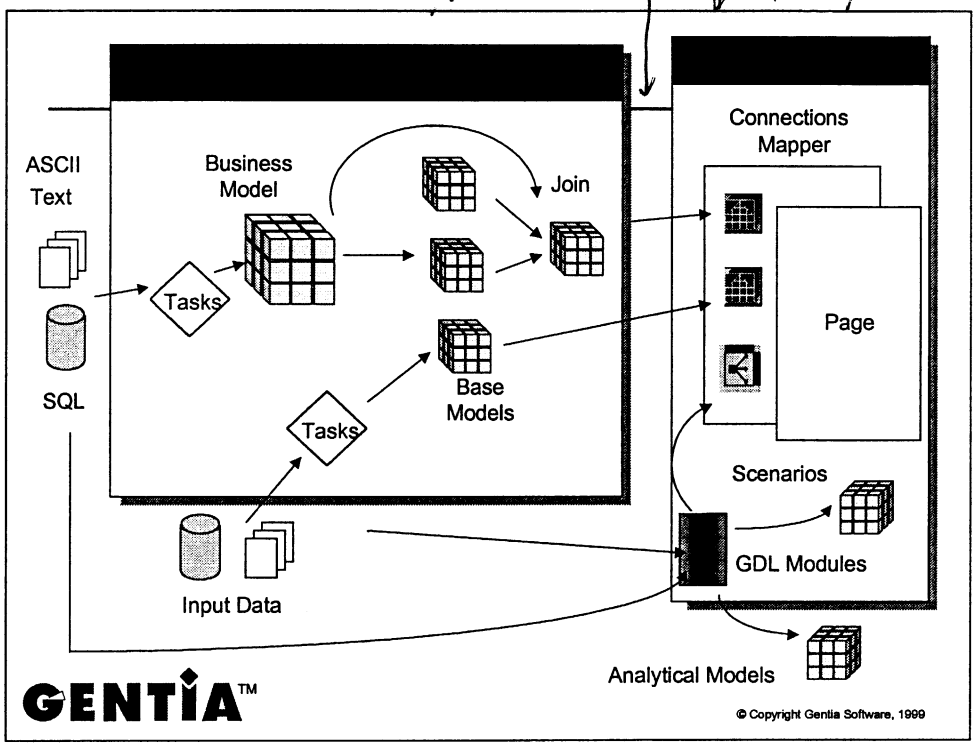
Each task will involve a series of actions. You specify these actions using TRANSFORMERS, linked together on a mapping area. Transformers allow you to break down a task into smaller easy to define units, for example the ASCII extractor transformer is used to specify details of an ASCII file (CSV format) from which data is to be extracted. This data, once extracted is cleansed and prepared for the business or base model and finally placed into a holder, ready to be loaded.


Tasks
ASCII
Text
SQL
Business
model
Tasks
Tasks
Base
models
Join
model
Input Data
GENTIA™
© Copyright Gentia Software, 1999

GentiaDB Architecture

- Business model
- Base models
- Join models
- Tasks
- Interaction with object store, analytical models & scenarios

GENTIA™


ASCII
Text
SQL
Business
Model
Tasks
Join
Base
Models
Tasks
Input Data
Connections
Mapper
Page
Scenarios
GDL Modules
Analytical Models
GENTIA™
© Copyright Gentia Software, 1999

Time Dimension

- Dedicated time dimension in business model
- Hierarchic structure
  - Periods (lowest level)
  - Span levels (intermediate levels)
  - Cycles (top level)
- Can split time into 2 dimensions
- Dynamic spans
  - YTD's, rolling totals
- Can 'drop' historic data

GENTIA™

Item Dimension

- Dedicated item dimension in business model
- Items can be computed or non-computed
- Items have an attribute to control how they consolidate
  - across all dimensions
  - no consolidation
  - across all non-time dimensions

GENTIA™

Category Dimensions

- Any further dimensions are held as category dimensions
- Very flexible
  - Add, move or delete members
  - All base and join models will be automatically updated
  - No need to reload data
  - Can choose to restate or retain historical data ← base model installing

GENTIA™

Tasks

- Used for:
  - Dimension definitions & structural updates
  - Data loading - from ASCII, SQL
- Made up of a series of linked transformers
- No coding - all drag & drop, point & click
- Can automate execution of tasks
  - Using jobs and agents
- Can run GDL and SQL queries from tasks

GENTIA™

Business Model Files

- The entire GentiaDB (business model, base & join models, tasks, etc.) held in **.bm** files
- Each file - up to 2Gb
- All files do not have to be held on same server; can be distributed across mapped drives
- Order of files must be maintained; cannot reorder or delete files
- Referenced in configuration file via route files

GENTIA™

Route File

- Used to list all the business model files
- Must have one route file for each business model.
- Read as an ASCII file; therefore use any text editor to create it.
- Example: **fred.rte**

```
c:\gdb\fred1.bm
c:\gdb\fred2.bm
c:\gdb\ fred3.bm
:
c:\gdb\ fred24.bm
```

GENTIA™

Configuration File Keywords

- **MEMORY** 8000000
  - Amount (in bytes) of server memory to be used by GentiaDB for the cache
  - Shared by all the business models on the server
  - Gentia will grab more if it needs to, rather than fail
  - 8Mb is the default and minimum
- **BUSINESS** salemodel
  - Defines the alias name for the business model
- **ROUTE_FILE** c:\...\fred.rte
  - Points to route file which contains list of **.bm** files
  - Must follow the BUSINESS line
- **LOG_FILE** c:\...\fred.log
  - Points to file which contains debug information about the business models

GENTIA™

Example Configuration File

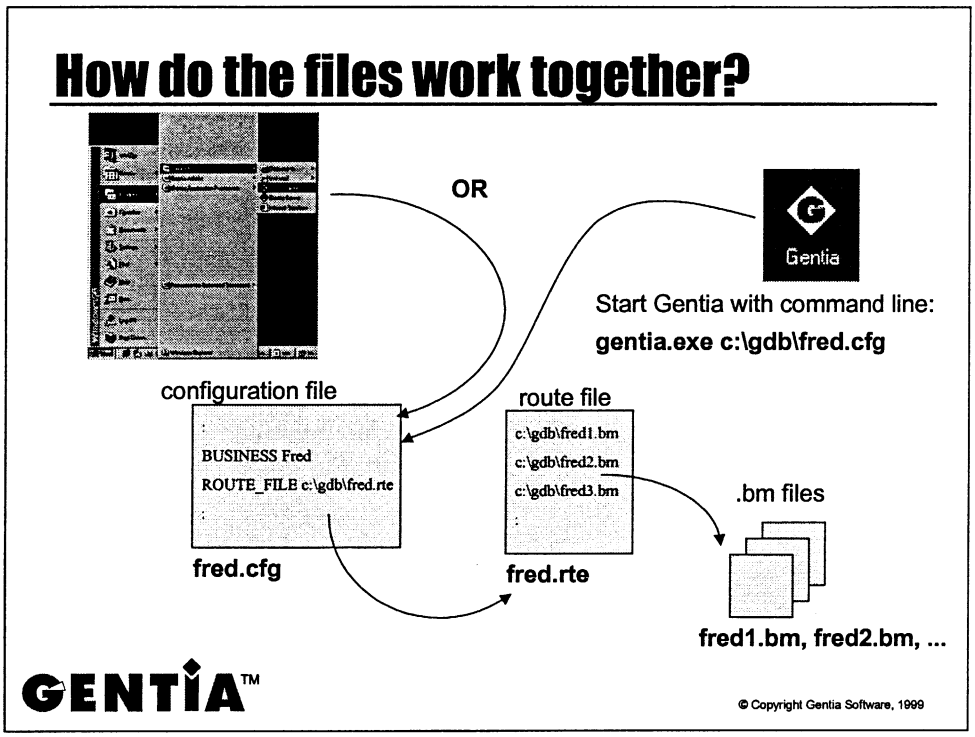
```
MEMORY 8000000
STORE Profit c:\gentia\gentiadb\profit.gos

BUSINESS Fred
ROUTE_FILE c:\gentia\gentiadb\fred.rte

BUSINESS Profit
ROUTE_FILE c:\gentia\gentiadb\profit.rte

LOG_FILE c:\gentia\gentiadb\data\profit.log
TIMEOUT 60
```

GENTIA™


How do the files work together?
OR
Gentia
Start Gentia with command line:
gentia.exe c:\gdb\fred.cfg
configuration file
BUSINESS Fred
ROUTE_FILE c:\gdb\fred.rte
fred.cfg
route file
c:\gdb\fred1.bm
c:\gdb\fred2.bm
c:\gdb\fred3.bm
fred.rte
.bm files
fred1.bm, fred2.bm, ...
GENTIA™
© Copyright Gentia Software, 1999

Summary

- Architecture of GentiaDB
  - Business model, base & join models, tasks
  - Interaction with object store, analytical models
- Features
- Item, time, category dimensions, tasks
- Files
  - Configuration, route, **.bm** files

GENTIA™

Getting Started...

- Start Gentia with configuration file which:
  - refers to a business model name
  - points to route file - does not need to exist
- Need to initialise new .bm files in services manager
  - enter .bm file names
  - initialise files
  - .bm file names will be written to route file
- Open business model manager
  - by creating and opening a model spec object

GENTIA™

1. Creating a Business Model

For the first part of this training course, we will be focusing on the following example business model which has five dimensions: Time, Item, Version, Location and Product which are structured as follows:

Time Dimension

Year	Total Year	
Qtr1	Quarter 1	
Jan	January	
Feb	February	
Mar	March	
Qtr2	Quarter 2	
Apr	April	
May	May	
Jun	June	
Qtr3	Quarter 3	
Jul	July	
Aug	August	
Sep	September	
Qtr4	Quarter 4	
Oct	October	
Nov	November	
Dec	December	

Item Dimension

UnC	Unit Cost	
UnP	Unit Price	
UnS	Units Sold	
Rev	Revenue	(UnS*UnP)

Version Dimension

Actual		
Budget		
Variance		(Actual-Budget)
VarPC	Variance Pct	(Variance % Budget)

Location Dimension

World	World
Eur	Europe
Lon	London
Ips	Ipswich
Utr	Utrecht
NthAm	North America
NY	New York
Atl	Atlanta
Den	Denver
SthPac	South Pacific
Syd	Sydney
HK	Hong Kong
Sin	Singapore
Afr	Africa
Joe	Johannesburg
Cap	Cape Town

Product Dimension

TotProd	Total Products
Whi	White Goods
Was	Washing Machines
Fri	Fridges
Bro	Brown Goods
Tel	Televisions
Vid	Videos
Hi	Hi-Fi's

When defining a business model initially, you need to complete a number of steps, which are as follows:

1. Open up a new business model
2. Define the time dimension
3. Define the item dimension
4. Define the category dimensions
5. Enter the dynamic spans
6. Commit the model

Getting Started

You first need to ensure that your configuration file and route file are pointing to the business model.

The route file

This contains the name of the files to which the business model and all the other components of the GentiaDB database (i.e. the base models, join models, all the data) will be saved. GentiaDB can span across several different files. For example, a route file could look something like this:

```
c:\sales\busmod\file1.bm
c:\sales\busmod\file2.bm
c:\sales\busmod\file3.bm
c:\sales\busmod\file4.bm
c:\sales\busmod\file5.bm
```

Each business model file can be up to 2Gb in size. Gentia will write to each of the files listed in the route file, in turn. If the files are likely to reach maximum size, new files can be added to the route file. Gentia will then add to all of the files in turn until the original files become full.

The files do not have to be located in the same directory or drive, but their order in the route file must always remain the same.

The configuration file

This contains the parameters that Gentia sets as it starts up, including the location of the route file. It could look something like this:

```
MEMORY 8000000
STORE "Store 1"
BUSINESS "Sales"
ROUTE_FILE "c:\sales\route.rte"
LOG_FILE "c:\gentia3\gentdb.mod"
```

where:

- MEMORY specifies how much of the memory available on the server is to be used by GentiaDB for the cache. This is shared by all business models on the server. 8000000 is the default value and also the minimum. If this is not enough Gentia will grab more if it can, rather than fail. However, if more memory is available, it is worth increasing the memory parameter, so that Gentia will make use of more memory, rather than only using the minimum that it requires.
- STORE defines the Gentia object store, where all application details, apart form GentiaDB are held (i.e. the pages and other objects).
- BUSINESS defines the business model, in terms of its alias name.
- ROUTE_FILE is the file which contains the list of files over which the business model is to be spread. This must come immediately after the BUSINESS keyword.

- LOG_FILE is the file to which Gentia writes debug information about the database. If it doesn't exist, Gentia will create it.

Creating a New Business Model

By referring to a new business model in the configuration file, Gentia will automatically create a new, empty business model. The route file and path name must also appear in the configuration file, after the business model. If the file itself doesn't exist, Gentia will create it. In terms of the business model files to which Gentia saves the GentiaDB (the **.bm** files), these can be created and initialised using the services option from the warehouse manager menu on the builder keypad.

Model Spec Object

This is a Gentia object that allows you to specify a business model and corresponding base model to view on a page. Since the model spec is an object, you can define access rights for it, in terms of the 0-7 author and user access levels. In this way, your application may have a business model with many base models, where different users are only allowed to view certain base models.

The model spec is also used to establish other business model settings, as you will see later.

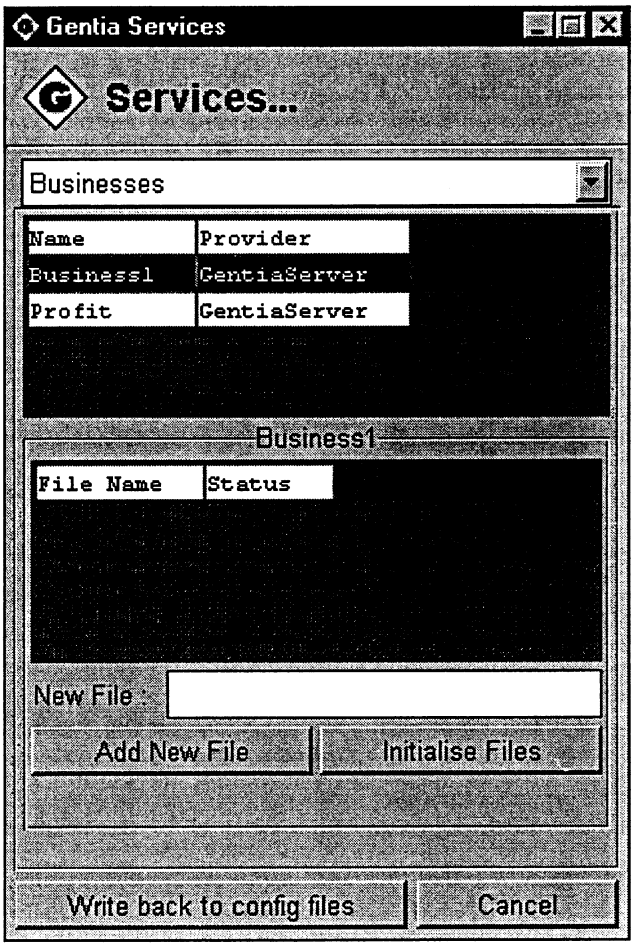
An important feature of the model spec at this stage is that it allows you to access the business model manager.

Practical 1.1 - Starting a new Business Model

- Start Gentia by double clicking on the GentiaDB icon. From the Framework, switch into *Author* mode by holding down the Ctrl key and clicking the mouse on the page background.

A configuration file **gentiadb.cfg** is specified which refers to a business model called **Business1**. The route file is called **busmod1.rte**. No .bm files exist yet.

- From the warehouse manager menu on the builder keypad, select services...
- From the services manager menu, select *Businesses*, and click on the Business1 model name.



- At the *New File* field enter the name of the first .bm file: **c:\gentia\gentiadb\file1.bm**
- Click on the **Add New File** button.
- Repeat, to add file2.bm and file3.bm with the same path as file1.bm

The files appear in red and the *Status* reads *fail*.

File Name	Status
c:\gentia\gentiadb\file1.bm	fail
c:\gentia\gentiadb\file2.bm	fail
c:\gentia\gentiadb\file3.bm	fail

- Click on the **Initialise Files** button, to initialise the .bm files

They should now appear in black, and the Status changes to OK:

File Name	Status
c:\gentia\gentiadb\file1.bm	OK
c:\gentia\gentiadb\file2.bm	OK
c:\gentia\gentiadb\file3.bm	OK

The .bm files have now been created. You can check this by using Windows Explorer to view the contents of the c:\gentia\gentiadb directory.

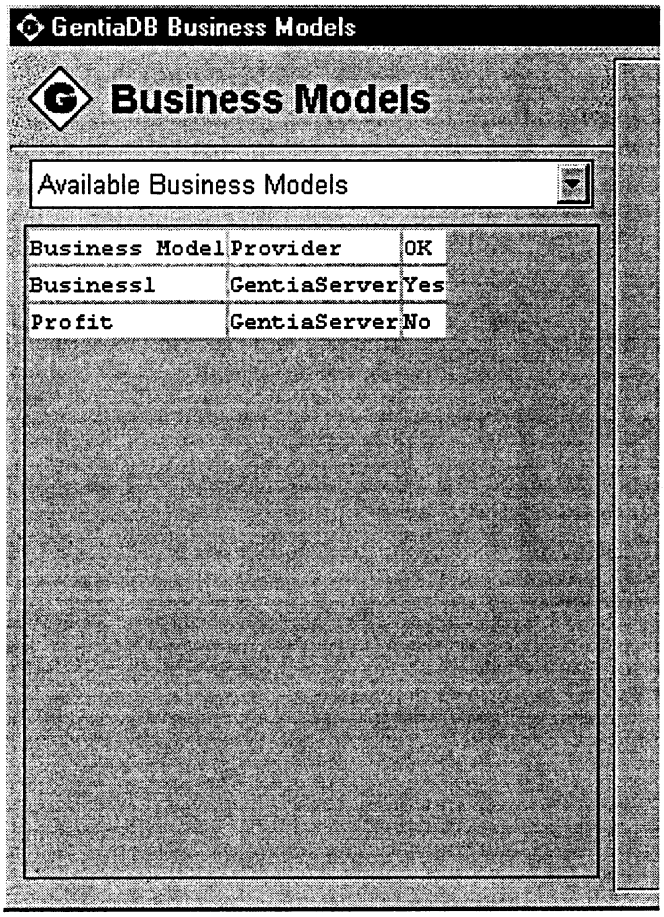
- Close the services window.
- Open book manager.
- Select *Model Spec* from the object pick list.
- Drag a new model spec object into the *GDB* user work area.
- Name it **Example1**

The model spec window will open.

- Click on the **View Business Model Manager Inspector** button at the bottom of the model spec window, to open the business model.

The available business models specified within the configuration file will be displayed as follows:

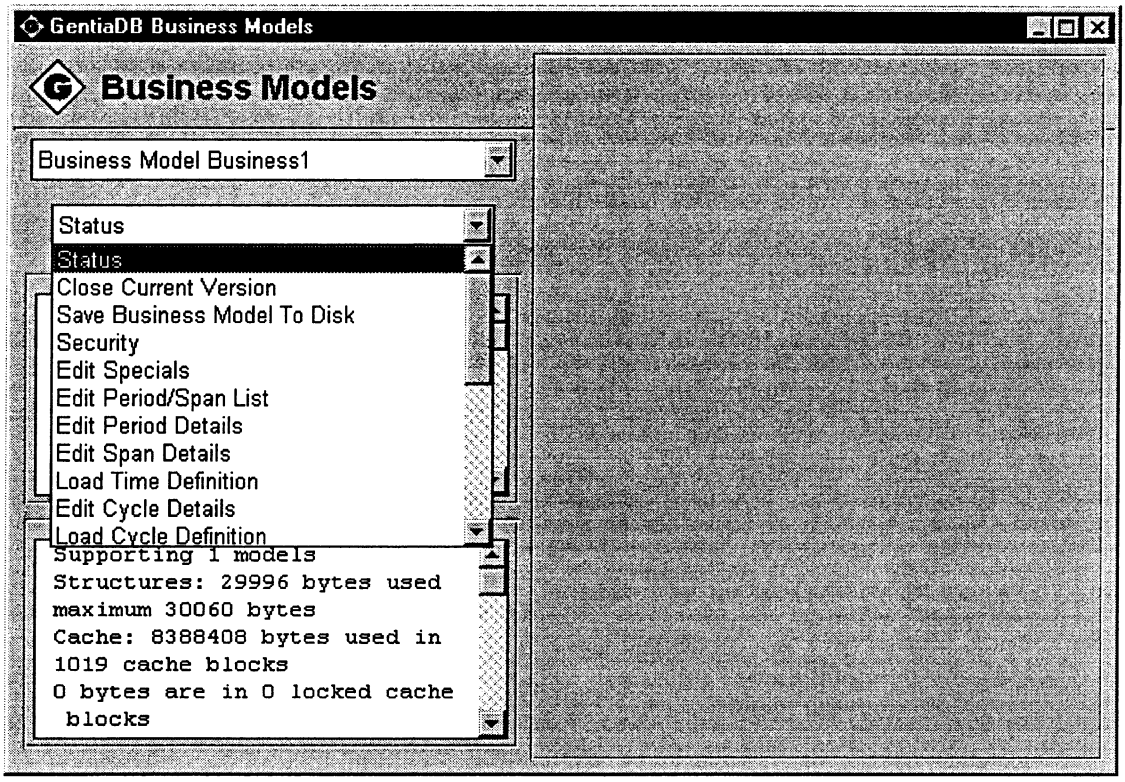
(see next page)



- Click on the name Business1 to open this particular business model.

The options in the pull-down menus are used to define the business model.

(see next page)



2. The Time Dimension

A business model must have one and only one time dimension. If the application is not time based, a dummy time dimension can be introduced with one member.

The time dimension is by default hierarchic; the levels within the hierarchy are defined as follows:

Period These are the base level members of the dimension, at the bottom of the hierarchy.

Cycle This is the top level member of the hierarchy, typically the year total.

Span These are the intermediate levels of the hierarchy

For example:

Cycle	1997
Span	Qtr 1, Qtr 2, Qtr 3, Qtr 4
Span	Jan, Feb, Mar, Apr, May, Jun, Jul, Aug, Sep, Oct, Nov, Dec
Span	Wk1, Wk2, ...Wk52
Period	Day1, Day2, ...Day365

As many cycles as necessary can be defined. Typically, an application might stretch over a number of years, in which case a cycle would be set up for each year's worth of data.

Do note that once a time dimension has been defined, its structure cannot be altered. Care should be taken to ensure that the time dimension has been designed appropriately. However, new cycles can be added or removed when required. Gentia has facilities to control how data can be 'dropped off' after a certain amount of time. This can vary for different levels of data within the hierarchy.

By default the time dimension is called **Time**. This can be changed by using the *Edit Specials* option.

Display Sets

In Gentia it is possible to create descriptions for each of the members for display purposes. This is done by defining a display set, which contains a display string (description) for each member. Any number of display sets can be defined to hold different versions of descriptions, for different languages, say. Display sets **can** be changed after the time dimension has been created.

For example,

Display Set	Display String
English	January, February, ...
French	Janvier, Fevrier, ...

Synonyms

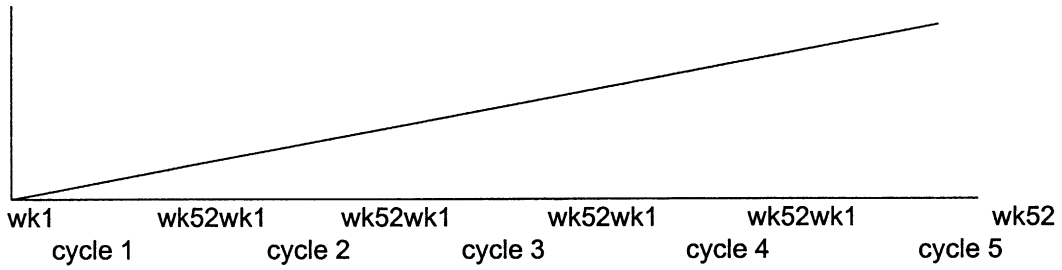
Synonyms are used mainly for data loading purposes when a reference to a dimension member is different from the defined code name. For example, when a data file refers to the time members by numbers, e.g. 01=Jan, 02=Feb, ...etc. a synonym set can be set up to store the cross-references for each member.

Splitting Cycles

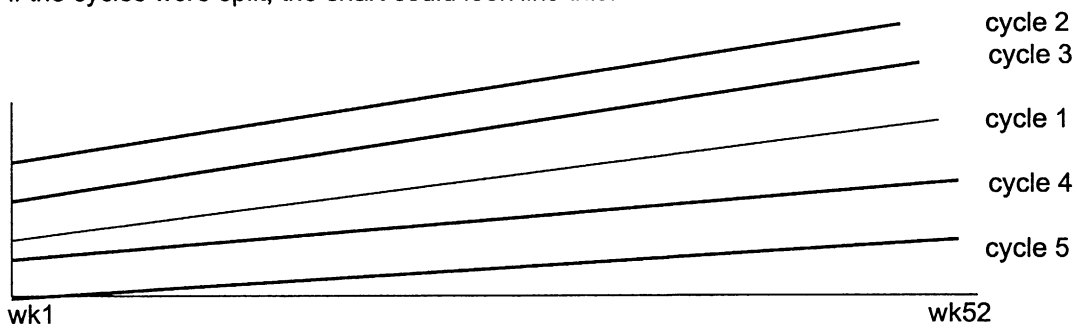
For display purposes the time dimension can be split into cycles, in order to compare data in individual years, say. This option is available in the model spec object.

For example:

A time based chart could look like this if the cycles were not split:



Or, if the cycles were split, the chart could look like this:



Edit Specials

Since cycles can be split within the time dimension, as above, Gentia will allow you to define a name for the cycles part of the time dimension, as though the cycles actually form a dimension in their own right. This is know as the **Cycle Dimension.** Within the cycle dimension, a name can also be defined for the components; this is known as the **In-Cycle Dimension**. The **Cycle Sum Member In-Cycle Dimension** is the name that you can give to the top parent within each cycle.

For instance, a model with 2 years, 24 months could have the following defined:

Cycle Dimension:	Year
In Cycle Dimension:	Period
Cycle Sum Member In-Cycle Dimension:	Total Year

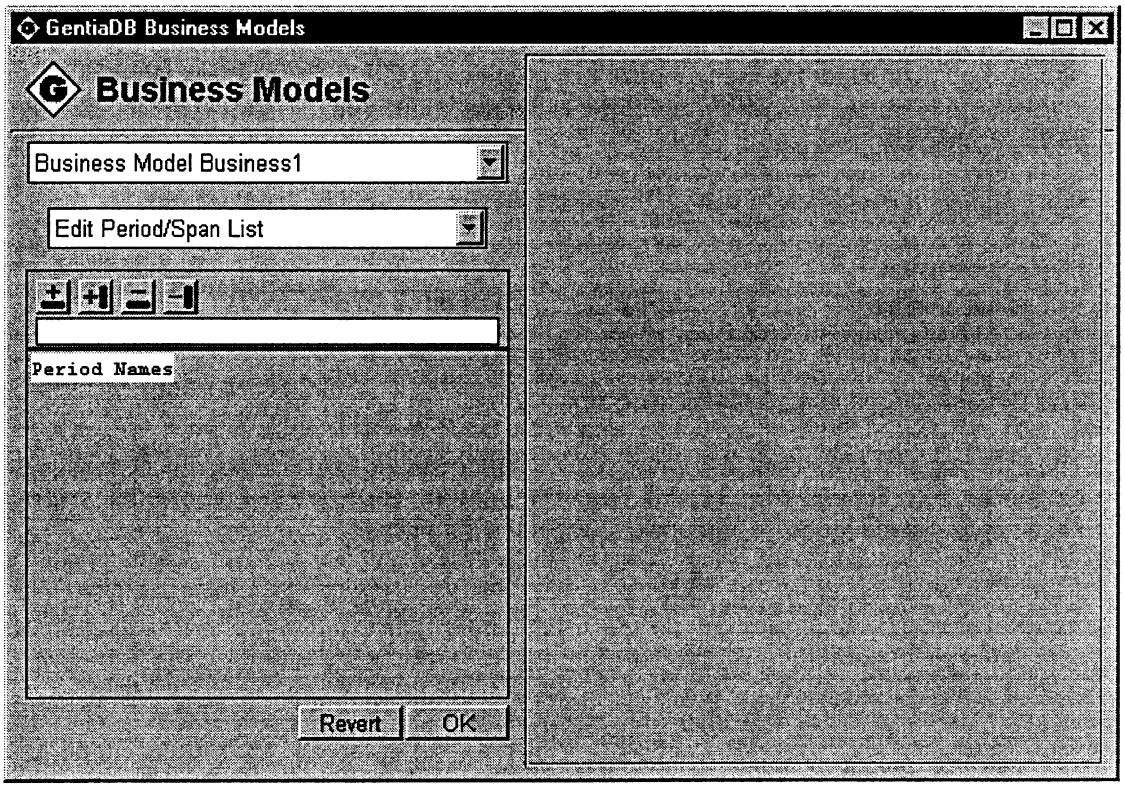
This will be covered in more detail in section 11.

The *Edit Specials* option will also allow you to change the default names of the item and time dimension from **Item Dimension** and **Time Dimension**, respectively. For instance, you may not want to include the word, dimension, within the name itself.

Practical 2.1 - Defining the Time Dimension

- From the pull-down menu, select *Edit Period/Span List*.

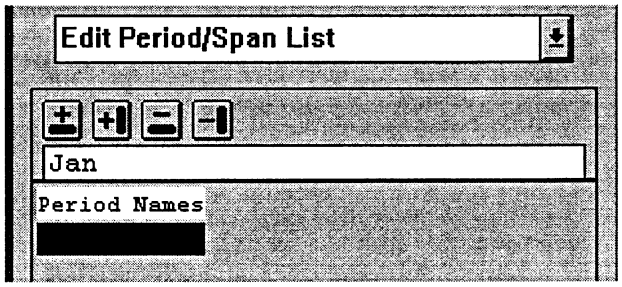
We will be specifying the members of the time dimension from here.



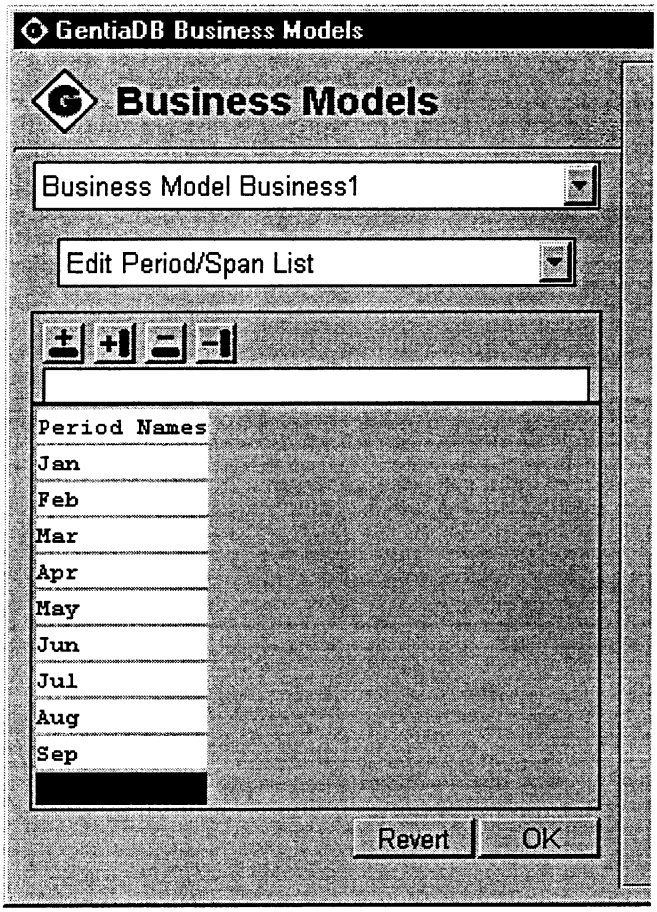
Periods

- Click on the  button, which will place a highlighted field under the *Period Names* heading.

- Click on the highlighted field, and then on the blank field above the *Period Names* heading, and type in the first *Period Name*, **Jan**.



- Click on the empty field under Jan, and repeat the process to define the remaining *Period Names*, **Feb** through to **Dec**.
- Ensure that each new period is created under the previous one. If necessary, use the  button to create a new blank field underneath the selected one.

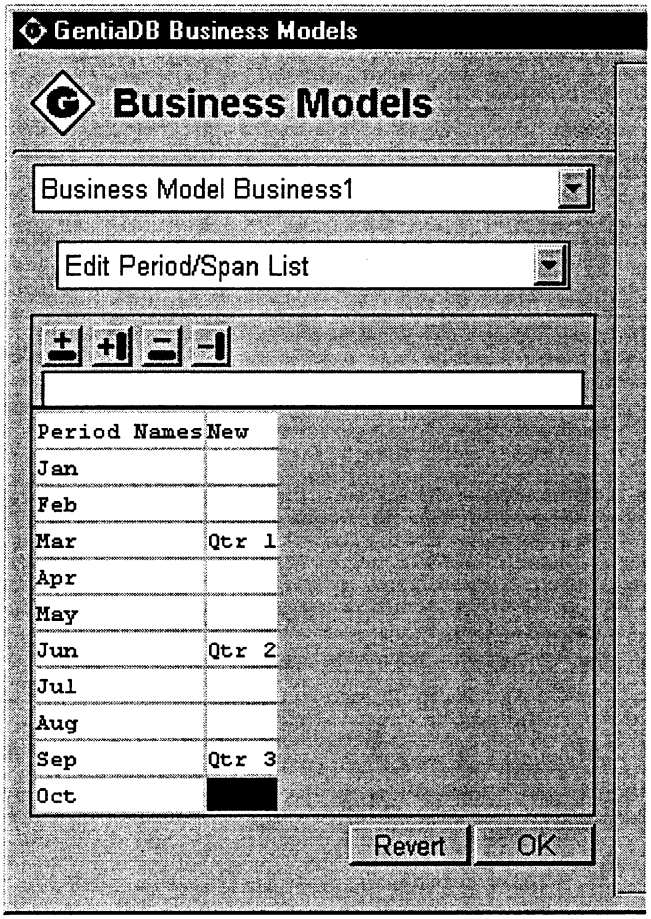


- Use the button to delete any unwanted *Period Names*, including blank ones.
- Click on OK to save the periods

If you have any errors you will be informed of them at this stage. Ensure that you amend any errors before continuing.

Spans

- Click on the button to enter the spans.
- Enter the spans in the same way as you entered the period, as below:



- Click on OK to save.

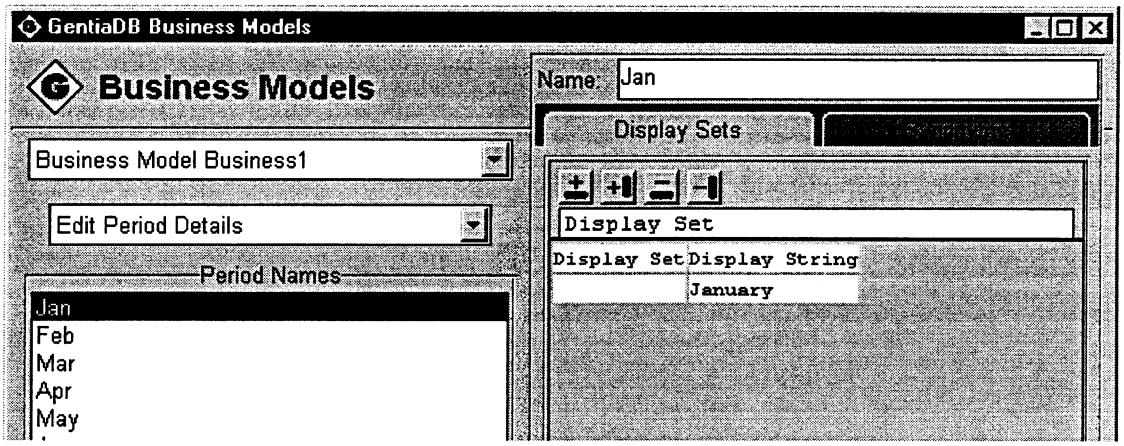
So far, the time dimension structure looks like this:

Span	Qtr1	Qtr2	Qtr3	Qtr4
Period	Jan Feb Mar	Apr May Jun	Jul Aug Sep	Oct Nov Dec

Display Sets

The names that have been entered for the dimension members are the unique code names. The next step is to enter the descriptions.

- From the pull-down menu, select *Edit Period Details*
- Click on the first period name, Jan. This opens up the display sets area of the screen where you can specify many display strings for each member.



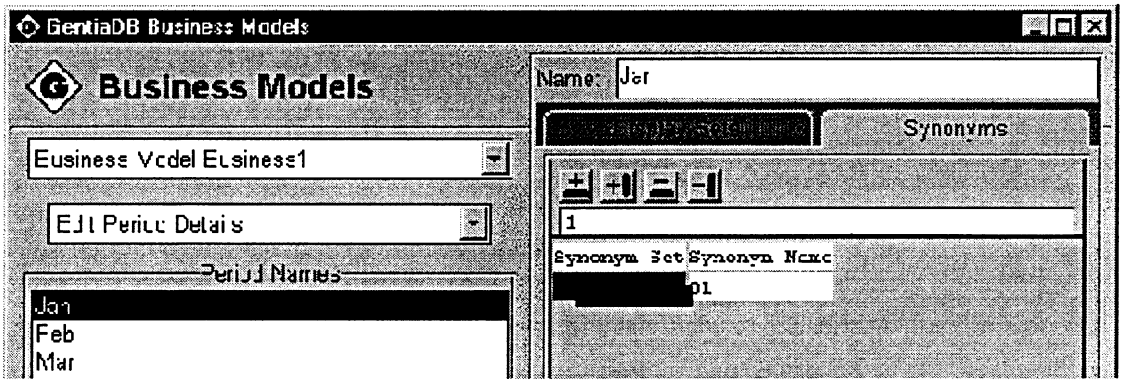
- Click on the  button and then on the blank field below *Display String*.
- Enter the description, **January**, into the field below the four buttons.

There is no need to enter the name of a display set; by leaving it blank, this display set will effectively be the default descriptions.

- Click on OK to save.
- Click on Feb, and repeat the above steps to define display strings for each of the periods, **February - December**.

Synonyms

- To set up a synonym, first select the period, Jan.
- Click on the synonyms tab.
- Enter 01 as a synonym for Jan, as below:



- Don't forget to click on OK to save.
- Repeat, to enter synonyms for the other months
- Use the *Edit Span Details* options in the pull-down menu to define descriptions for the spans, as follows:

Qtr1 Quarter 1

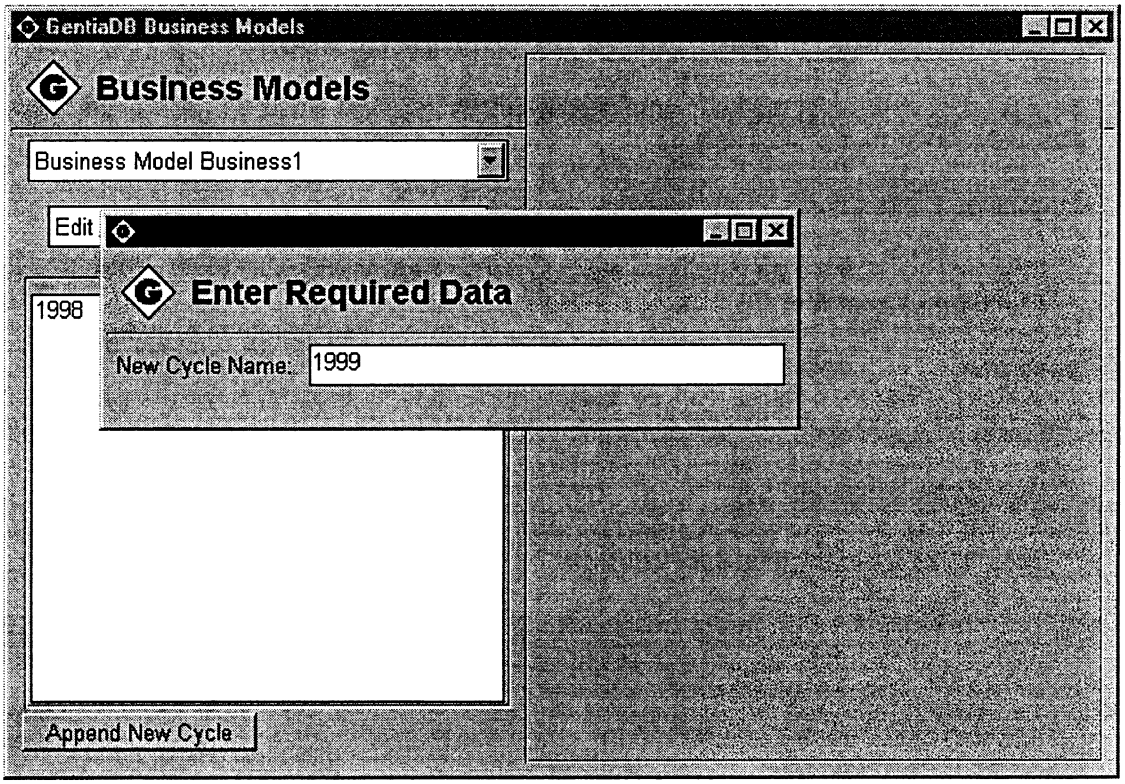
Qtr2 **Quarter 2**
Qtr3 **Quarter 3**
Qtr4 **Quarter 4**

Cycles

Select the *Edit Cycle Details* option for the pull-down menu. Cycles can be entered and altered once a model has been committed, even though the other aspects of the time dimension cannot be changed.

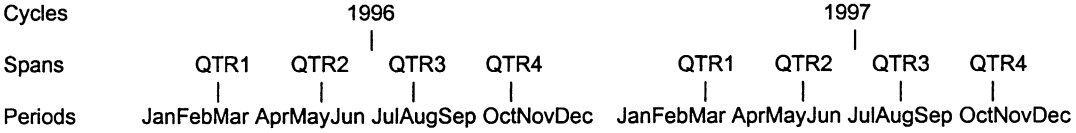
Each cycle that is created will adopt the underlying periods and spans already defined.

Click on *Append New Cycle* at the bottom of the screen and enter both **1996** and **1997** as cycles:



A description for each cycle can be defined by clicking on the cycle name and entering a display string as with previous elements. Enter descriptions if you wish.

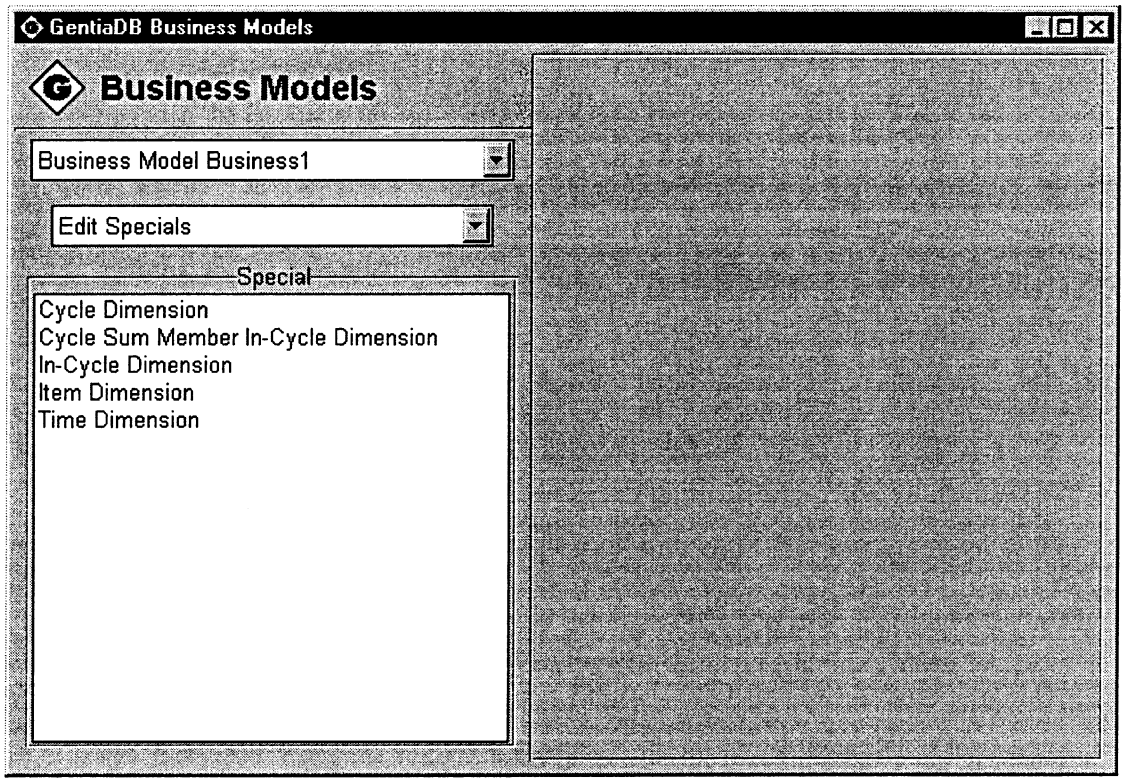
The time dimension will now have the following structure:



As new cycles are entered the structure will grow, adding the spans and period to each cycle.

Edit Specials

From the pull-down menu, select *Edit Specials*.



- Use the *Item Dimension* and *Time Dimension* options to change the names of the dimensions to **Item** and **Time**, respectively.

- Don't forget to click on *OK* each time, otherwise, the details will not have been saved.

The time dimension is now complete. Once we commit the model, the time dimension cannot be altered, except for cycle details and the display strings.

- From the menu, select

to write to the .bm files, so that your work is saved.

3. The Item Dimension

As with the time dimension, a business model must have a dimension reserved for items. The item dimension contains two types of member: computed items which are calculated within GentiaDB from rules defined within the model, and non-computed items which have no rules associated with them.

For example, you might have data for the price of an article and the number of units sold, in which case these item members would not be computed items. But you may wish to calculate the revenue generated by the sale of the articles in which case the revenue member would be a computed item, with a rule (known in Gentia as the expression) such as:

revenue = price * units sold

The two types are defined separately, using the *Edit Items* and *Edit Computed Items* options in the pull-down menu.

Note that item members cannot be deleted once created, however existing members can be changed and new members can be added.

Consolidation Rules

Consolidation attributes need to be set for each item to control the way in which Gentia will handle the consolidation of individual items. There are four options, as follows:

- *Full Consolidation* - This consolidates across all dimensions. An example of this might be the revenue for electrical items, which is computed on a monthly basis. In the example below, the three electrical items are consolidated into the final electrical figure.

	Iron	Heater	Kettle	Electrical
Revenue	£10,000	£3,000	£8,000	£21,000

- *Full Consolidation: Time Latest Value* - This will consolidate normally across all non-time dimensions, but for the time dimension members the consolidated value will be the latest value. This is very useful for cumulative totals and balance sheet items, e.g. Fixed Assets, where data is required at a particular point in time. The field Revenue has *Full Consolidation* whereas Cumulative Revenue has the *Full Consolidation: Time Latest Value*. In Quarter 1 the Time Latest Value is March's value.

	Jan	Feb	Mar	Quarter 1
Revenue	£10,000	£14,000	£11,000	£35,000
Cumulative Revenue	£10,000	£24,000	£35,000	£35,000

- *Dimension Only Consolidation* - Will consolidate across all dimensions except the time dimension. An example might be the average headcount in any given period, which would require more than one step to be calculated. In the example below, the dummy headcount contains a consolidation of three month's data and a yearly total. This figure is of no value until it is divided by the number of periods, which then provide the average headcount. The Dummy Headcount and the Number of Periods fields would be set to Full Consolidation.

	Quarter 1	Quarter 2	Quarter 3	Quarter 4	Year
Dummy Headcount	55	54	57	60	226
Number of Periods	3	3	3	3	12
Average Headcount	18.33	18	19	20	18.83

- *No Consolidation* - Consolidation will not be carried out at all, e.g. Unit Price, Contribution% which might have a rule to calculate items above the base level of the hierarchy, if necessary.

	January	February	March	Quarter 1
Unit Price	£76.00	£73.60	£76.80	N/A
Contribution	5	10	8	23
Revenue	10	30	32	72
Contribution %	50%	33.3%	25%	**31.9%**

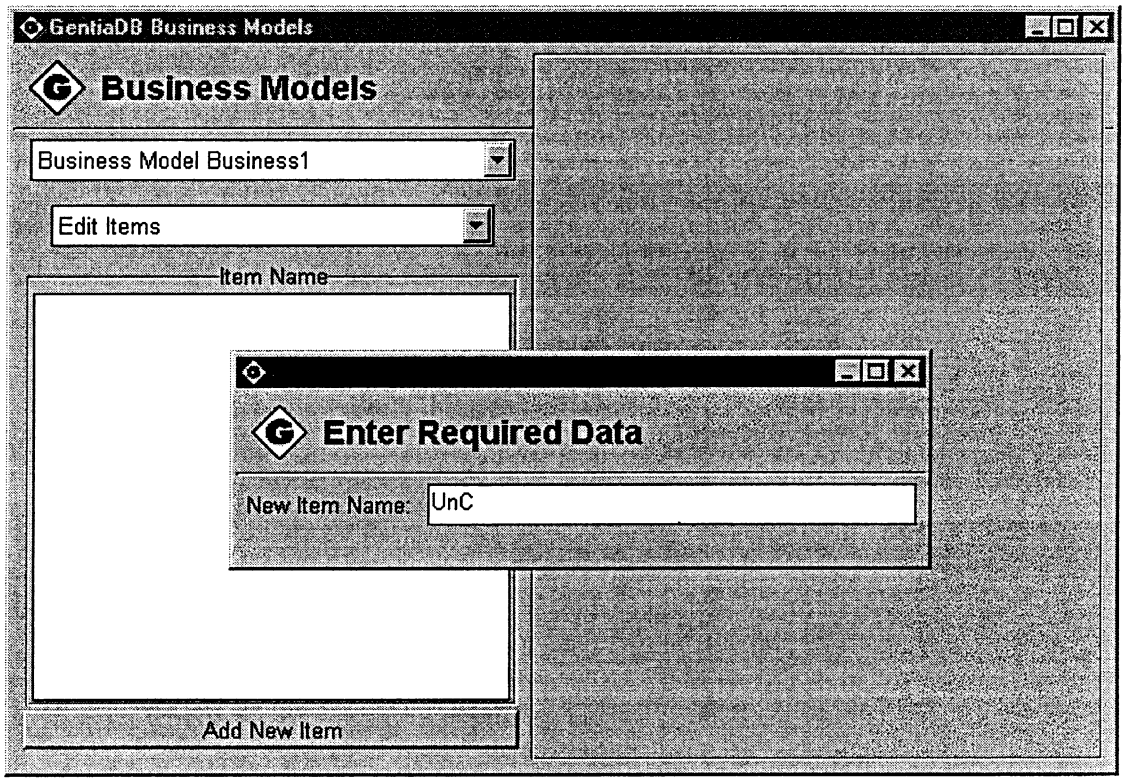
In the above example for unit price, there is no reason to consolidate the monthly figures in the Quarter 1 cell because the new figure would serve no purpose.

In the example for Contribution% there is a calculation which applies. Contribution% is calculated as Contribution / Revenue * 100. Although Contribution and Revenue might be consolidated into Quarter 1, Contribution% is not. Adding percentage figures together does not produce valid results. This figure is a calculated number, based on the consolidated figures in the cells above.

Practical 3.1 - Defining the Item Dimension

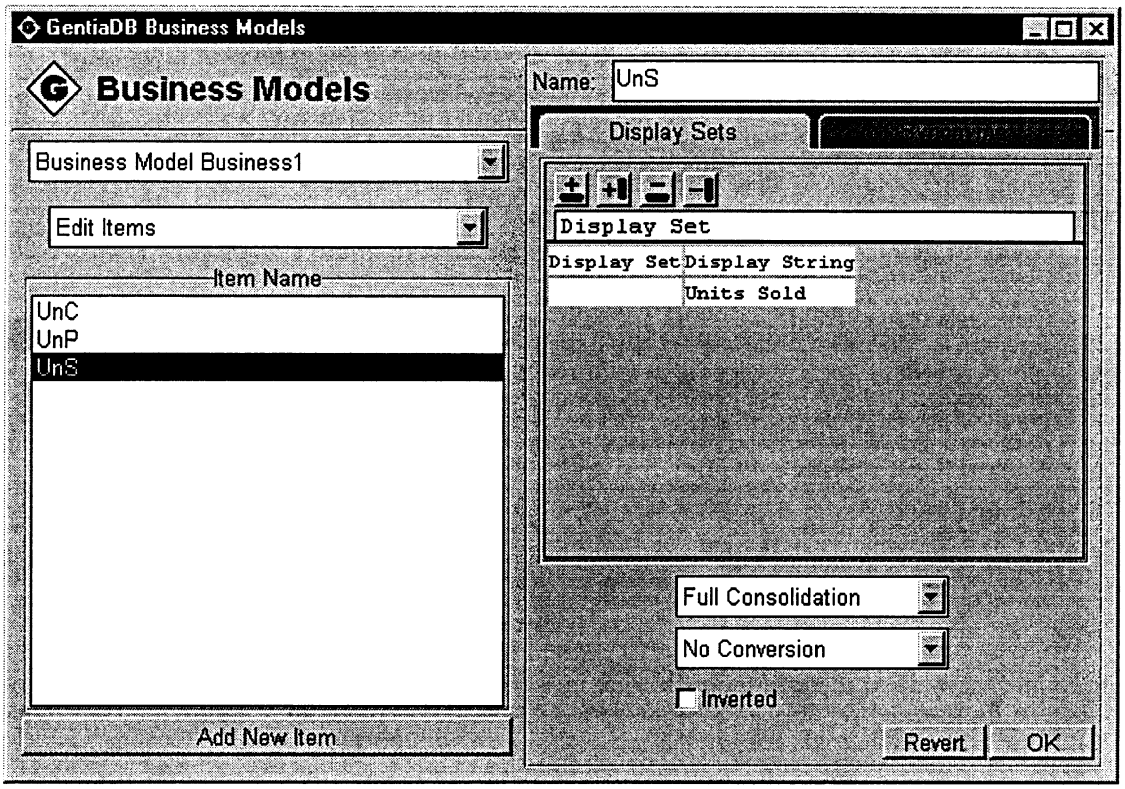
Non-Computed Items

- From the pull-down menu, select *Edit Items*.
- Click on *Add New Item*, at the bottom of the screen:

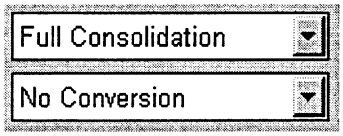


Enter in the following item member names and descriptions (by clicking on the item name, once defined), as you did for the time dimension members:

Name	Description
UnC	Unit Cost
UnP	Unit Price
UnS	Units Sold



- Select each of the items in turn and use the *Consolidation* pull-down menu to set the following consolidation attributes:

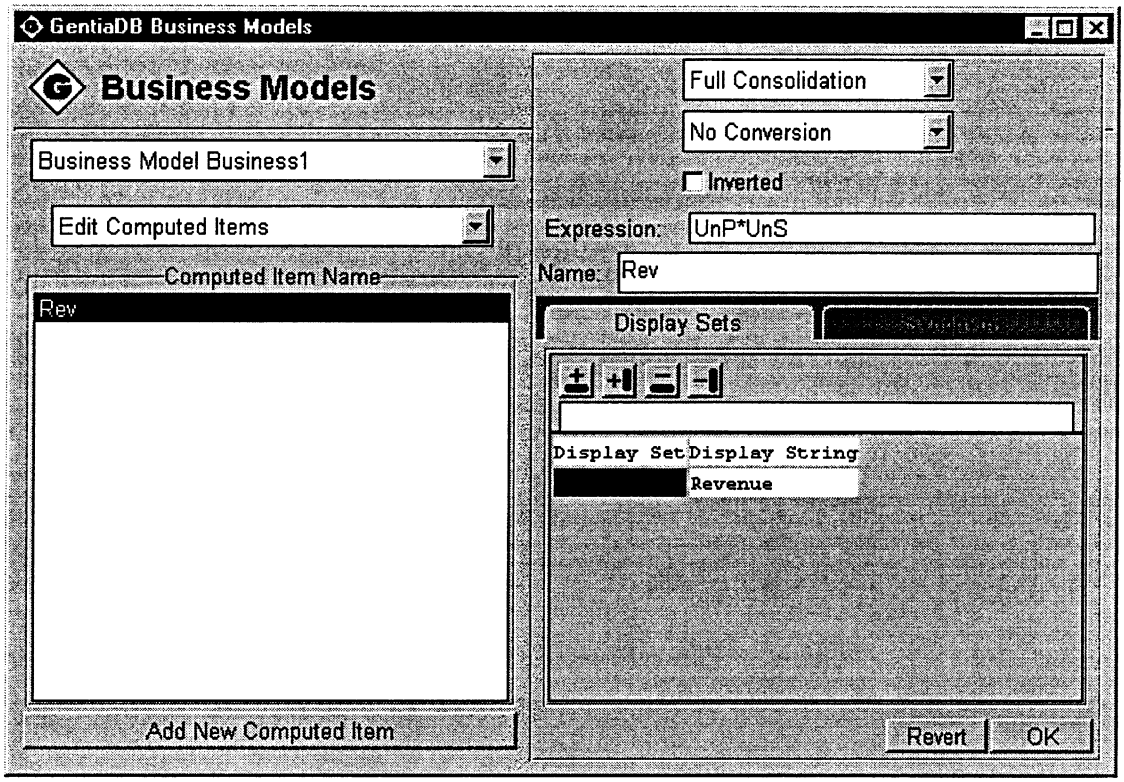


UnC - No Consolidation
UnP - No Consolidation
UnS - Full Consolidation

- Don't forget to click on *OK* for each item.

Computed Items

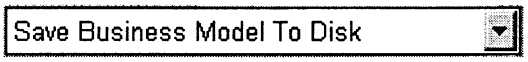
- From the pull-down menu, select *Edit Computed Items*.
- Click on the *Add New Computed Item* button.
- Enter the member, **Rev**.
- Once defined, enter the description **Revenue**, by clicking on the name Rev and entering a display string.



- In the *Expression* field, define the calculation for Rev as follows:

The consolidation attribute should remain as *Full Consolidation*.

- Click on *OK* to save.
- From the menu, select



to write to the .bm files, so that your work is saved.

4. The Category Dimensions

Once the time and item dimensions have been defined, any further dimensions are held as category dimensions. Category dimensions are flexible in that you can add, remove, or move members.

Category dimensions can be created by typing in the members or by reading in the members from a file. Both methods will be covered on this course. As with the item dimension, computed members and members with data are defined separately.

A category dimension can be either **Linear** or **Hierarchic**. If a hierarchic relationship is defined, there are a number of consolidation options:

Hierarchic and Additive
Hierarchic and Subtractive
Hierarchic only
Additive only
Subtractive only

Use one of the hierarchic options if you want Gentia to display the positions of a member within the hierarchy.

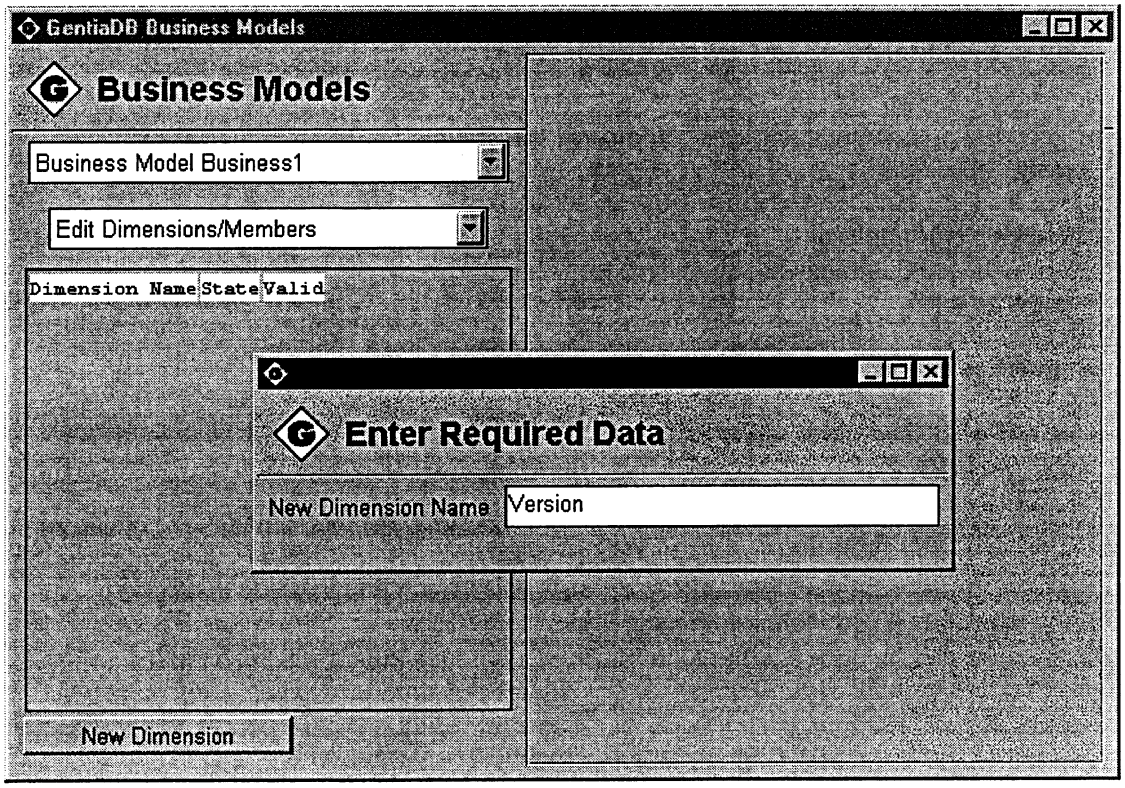
Use an additive option to consolidate a member by adding together the value of its children.

Use a subtractive option if you wish to subtract the value of that member during the consolidation process, rather than adding it.

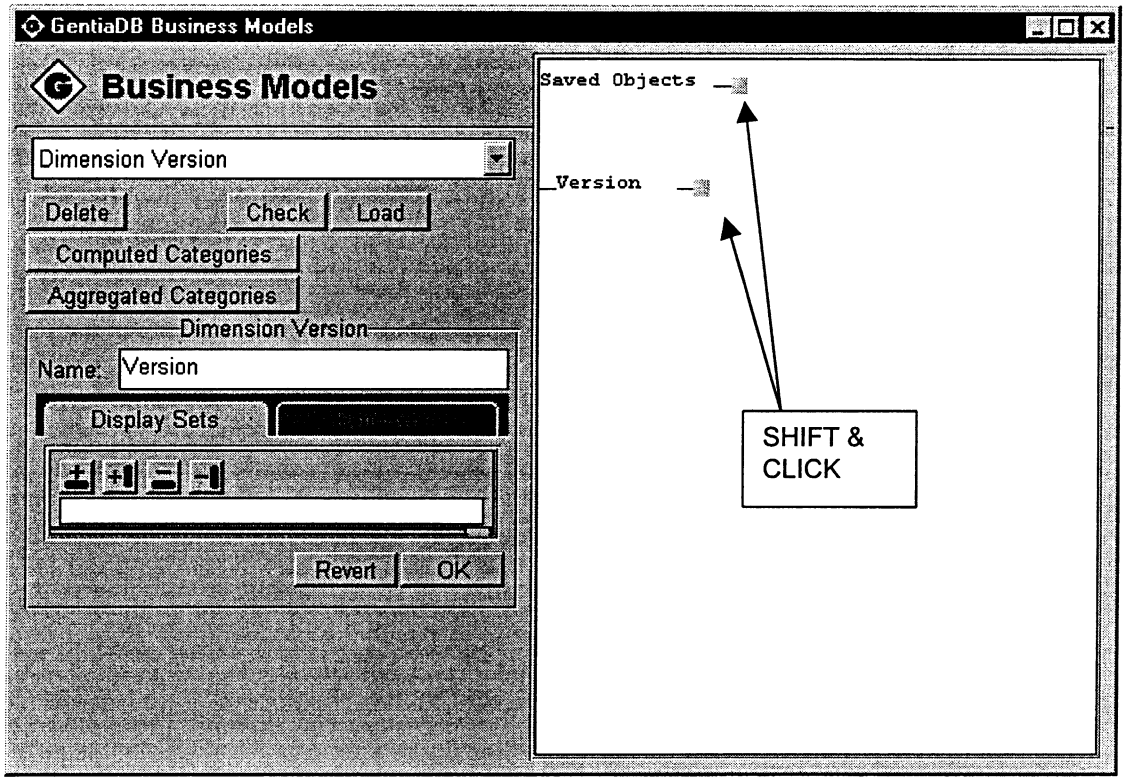
Practical 4.1 - Defining the Category Dimensions

The Version Dimension

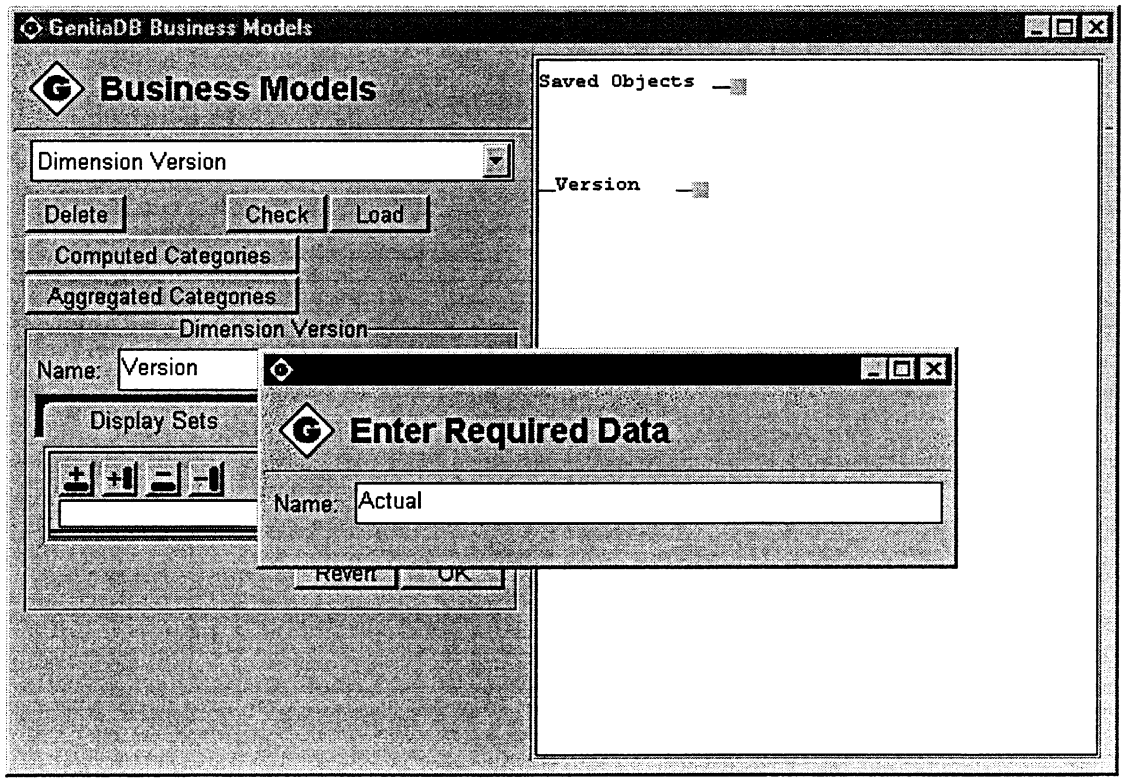
- Select *Edit Dimensions/Members* from the pull-down menu.
- Click on the *New Dimension* button at the bottom, and enter **Version** at the prompt.



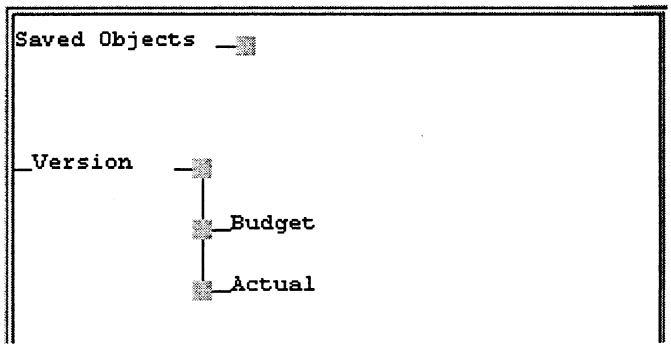
- Press the Enter key to open the hierarchy editor, where the members can be defined:



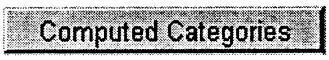
- Hold down the Shift key and click on the dimension name Version (or on the grey box to the right of the name).
- Enter the member name **Actual** at the prompt and press enter.



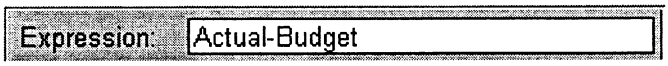
- Repeat, to define Budget as a member. The hierarchy editor displays the defined members of the version dimension:



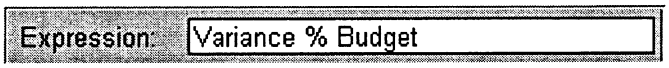
Actual and Budget will have data supplied for them. The other members, Variance and %Variance will be calculated.

- Click on the  button.

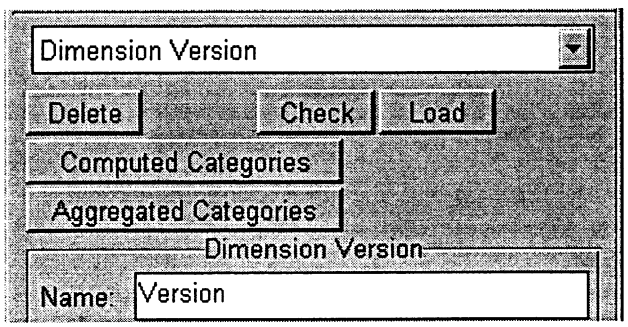
- Click on the *Add New Computed Member* button at the bottom.
- At the prompt, enter the member name **Variance**.
- Repeat, to enter the member name **VarPC**.
- Click on the member name Variance.
- Click on the *Expression* field and enter the rule for Variance:



- Click on OK.
- Click on VarPC, and enter the rule:

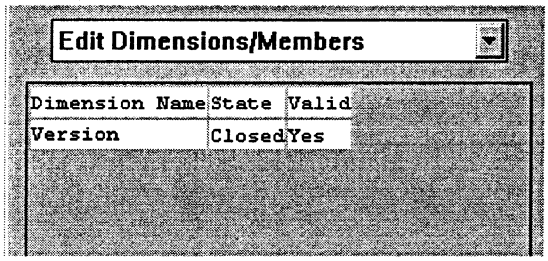


- Define a display string for VarPC, e.g. Variance Pct.
- Click on OK.
- Return to the previous window, by selecting *Dimension Version* from the menu.



- Click on the  Button. If there are any errors, these will be displayed in a pop-up window. Amend any errors before you continue.

- Return to the *Edit Dimension/Members* window, by first selecting Business model Business1 from the pull-down menu, and then *Edit Dimensions/Members* from the lower menu.

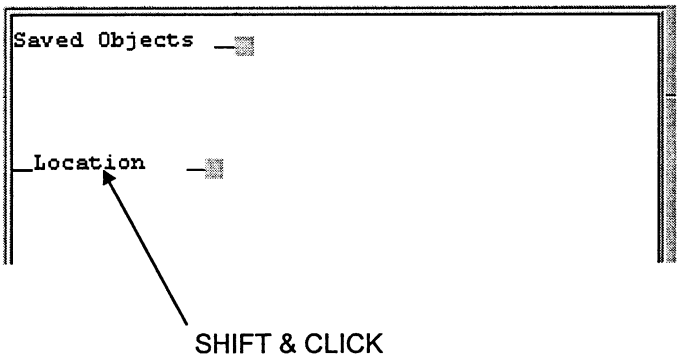


The *Valid* field now reads Yes to indicate that it has been checked and is free of errors.

The Location Dimension

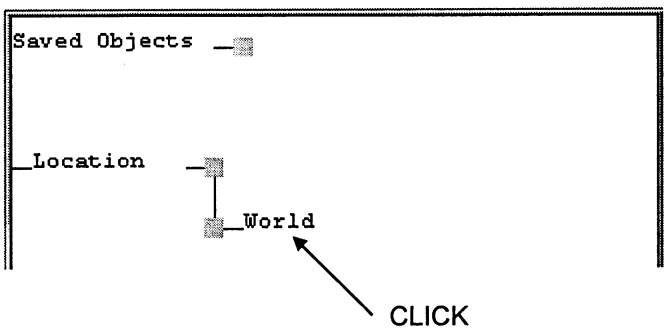
This will be a hierarchic dimension, whereas Version was linear.

- Click on the *New Dimension* button at the bottom, and enter **Location**.



- To enter the members for the location dimension, hold down the Shift key and click on the name Location.
- Enter **World**.

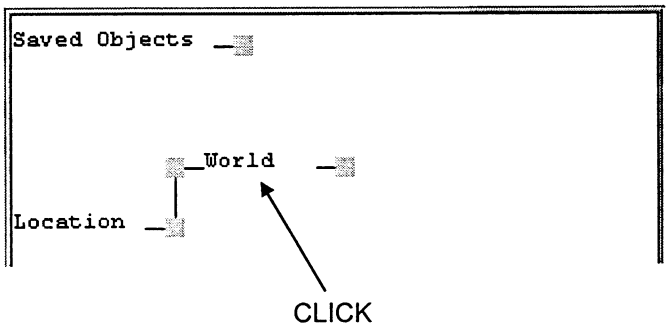
This will be the top parent in the location dimension



To enter a child member for World, a new level needs to be created.

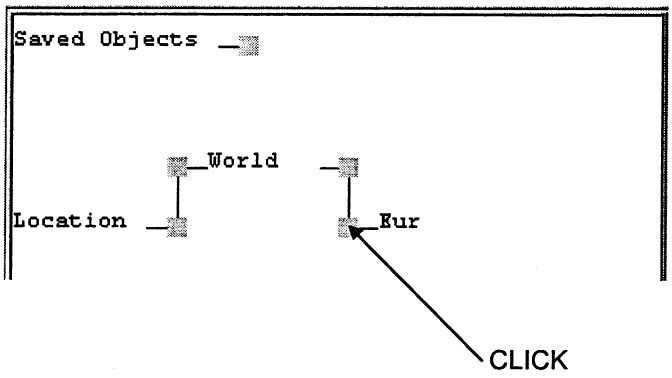
- Click on the name World.

This will add a new level to the hierarchy:

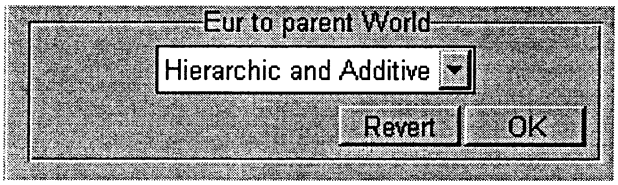


Now you can enter the children of World.

- Hold down the Shift key and click on the name, World.
- At the prompt, enter **Eur** and press enter.



- Click on the grey box to the left of Eur, which opens up the following area, to allow you to set the hierarchic consolidation rules for the member Eur:



We will stick with the default, Hierarchic and Additive.

- Click on Eur.

This creates a new level, to allow you to define children for Europe. But first, we'll define a description for Eur.

- Move to the left-hand area of the window and define a display string for Eur: **Europe**.
- Complete the location dimension to match the structure outlined below. You may name each member as you wish, but ensure that the base level names match exactly with the following:

Name	**Display String**
World	World
Eur	Europe
Lon	London
Ips	Ipswich
Utr	Utrecht
NthAm	North America
NY	New York
Atl	Atlanta
Den	Denver
SthPac	South Pacific
Syd	Sydney

HK	Hong Kong
Sin	Singapore
Afr	Africa
Joe	Johannesburg
Cap	Cape Town

Define display strings if you have time.

Remember:

To define children for a member, select the member to create a new level.

To define new members within a level, hold down the Shift key and click on the parent name.

To delete an unwanted member, hold down both the Shift and Ctrl keys and click on the name.

- When you have defined all the members, click on the Check button. If you have any errors, you will be presented with them at this stage. Ensure that you amend any errors before continuing.

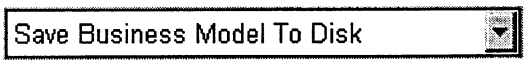
The Product Dimension

- Add the product dimension and define members for it. Again, you may name each member as you wish, but ensure that the base level names match exactly with the following:

Name	Display String
TotProd	Total Products
Whi	White Goods
Was	Washing Machines
Fri	Fridges
Bro	Brown Goods
Tel	Televisions
Vid	Videos
Hi	Hi-Fi's

- Define descriptions if you have time.
- Click on the 

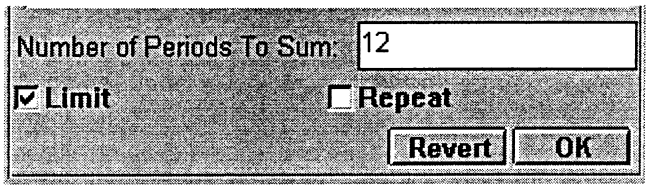
button to ensure that the product dimension is free of errors.
- Return to the Business model *Business1*.
- From the menu, select



to write to the .bm files, so that your work is saved.

5. Dynamic Spans

Dynamic spans are calculated elements within the time dimension, that vary over time, like year-to-date figures and rolling totals.



For each dynamic span, you set the number of periods across which it will span. For example, YTD and rolling 12 month figures would span across 12 periods.

The *Limit* check box is for dynamic spans that are limited to the current cycle, e.g. YTD. For a rolling 12 month span, *Limit* should not be checked.

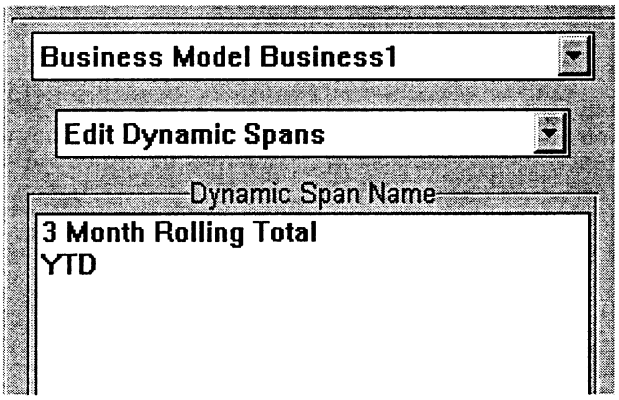
The *Repeat* check box allows rolling figures to be repeated so that you can view figures for each of the dynamic spans prior to the current period.

N.B. Rolling totals will roll backwards from the current period. Thus, a 3 month rolling total, if the current period is May, will be as follows;

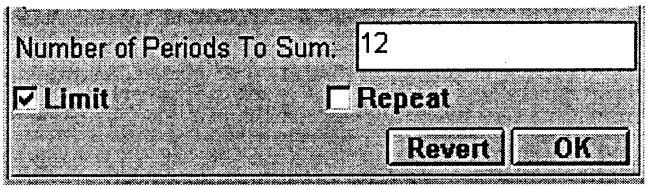
	Jan	Feb	3 month rolling total	Mar	Apr	May	3 month rolling total
Sales	15	24	39	21	13	22	56

Practical 5.1 - Defining Dynamic Spans

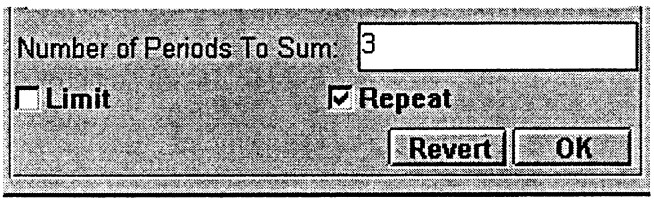
- Select *Edit Dynamic Spans* from the pull-down menu.
- Click on the *Add New Dynamic Span* button at the bottom.
- Define two dynamic spans: YTD and 3 Month Rolling Total.



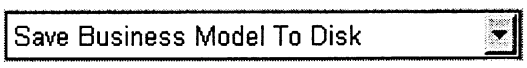
- Click on YTD to open the right-hand side of the window.
- Enter the number of periods to span, and check the *Limit* box.



- Click on OK to save.
- For the 3 Month Rolling Total, set the following:



- Click on OK to save.
- From the menu, select



to write to the .bm files, so that your work is saved.

6. Committing the Business Model

The business model has now been defined. The next stage is to close the model and then commit it. The current version of the business model is 0; this will become version 1 once the model is committed. It is not possible to retrieve earlier versions, unless of course, you make a backup copy of the .bm files; the version numbers are for identification purposes.

Changes in a business model will not affect the base models until it has been committed.

The business model holds a marker identifying which of the periods in the time dimension is the current period. This allows you to use the current period as the default period for data loading. GentiaDB will also use the current period when calculating dynamic spans.

When committing the first version of the business model, you must specify the current period at that time. Thereafter, you will need to advance the current period each time you move into a new period. Once you have advanced the period, you **cannot** revert the current period to an earlier period.

Once the Model has been committed, the structure of the time dimension **cannot** be changed.

Practical 6.1 - Committing the Model

- From the lower pull-down menu, select *Close Current Version*. No further editing can take place, unless you were to *Reopen Current Version*.

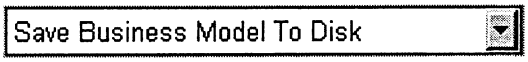
At this stage, if your category dimensions had not been checked, you would get a message telling you so. A business model cannot close until all category dimensions have been checked.

- Select *Commit Version Zero*.
- You will then be prompted to set the current period.
- Select *May 1997*.
- Click on the

button.

The model status will now refer to version 1.

- From the menu, select



to write to the .bm files, so that your work is saved.

7. Base Models

Base models are defined in terms of the business model definition. You specify a sub-set of dimensions and members you require: all the members or a number of generations in a hierarchy or specific branch.

Data is held at the base model level, so you will need to analyse the business model to determine how it should be broken down to form a suite of base models for data loading and access.

Each base model must have a time, an item and at least one category dimension; you specify which of the category dimensions are to be included. Within each dimension, any number of members can be selected. The maximum number of item members within a base model is 1000, providing no other dimensions are non-keyed (see later).

Any changes to the business model e.g. restructuring of dimensions, or changes to dimension and member descriptions will automatically be reflected in the base models.

When identifying members of the item dimension to be included in the base model, for computed items, you specify whether these are to be stored, or calculated on-the-fly when accessed.

For the time dimension, you can identify when data is to be dropped. For instance, you may want to discard monthly data after two years, quarterly data after three, and yearly data after five years.

In terms of the category dimensions, you can specify the members you wish to include in a number of different ways:

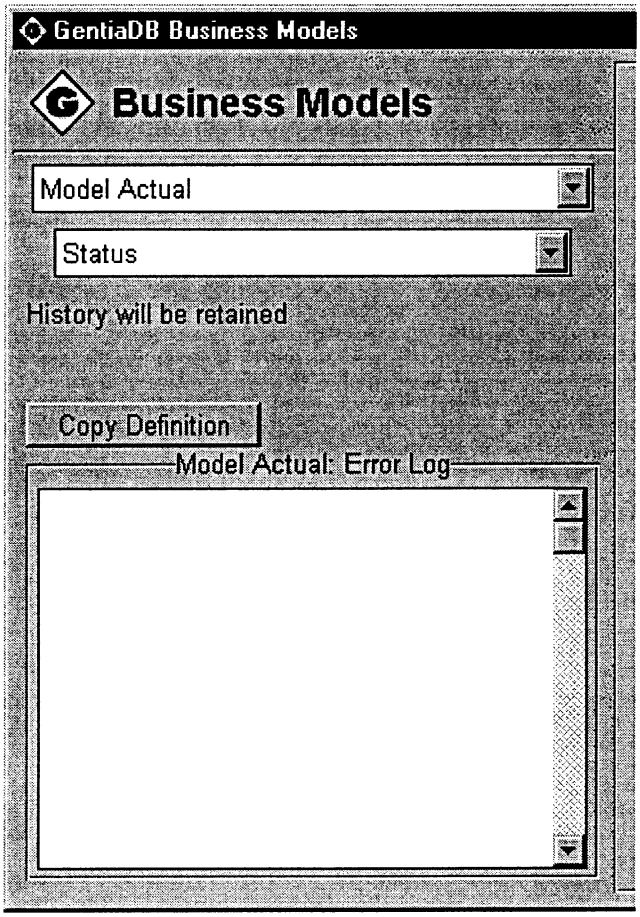
- All members.
- Choose a point in a hierarchy and specify how many generations below that point are to be included.
- Members below a specific level.
- A combination of level and a number of generations.
- You can specify individual points within the hierarchy, but in this case, automatic updating would not happen if the business model were to change.

Calculation and consolidation rules are taken from the business model. No additional dimensions, members, or rules can be added to the base model.

Base models effectively consolidate by aggregating a series of adjustments to the database. A change to a base level data cell triggers a series of adjustments to its hierarchic parents. GentiaDB will also identify which data cells need to be updated following the data load, but cannot be calculated by hierarchic adjustment, e.g. ratios. These cells are recalculated using the pre-defined rules. The unaffected cells in the database are left untouched. This process means that GentiaDB keeps track of changes and only consolidates on the impacted values, resulting in only a small amount of work. This approach provides a fast and intelligent consolidation.

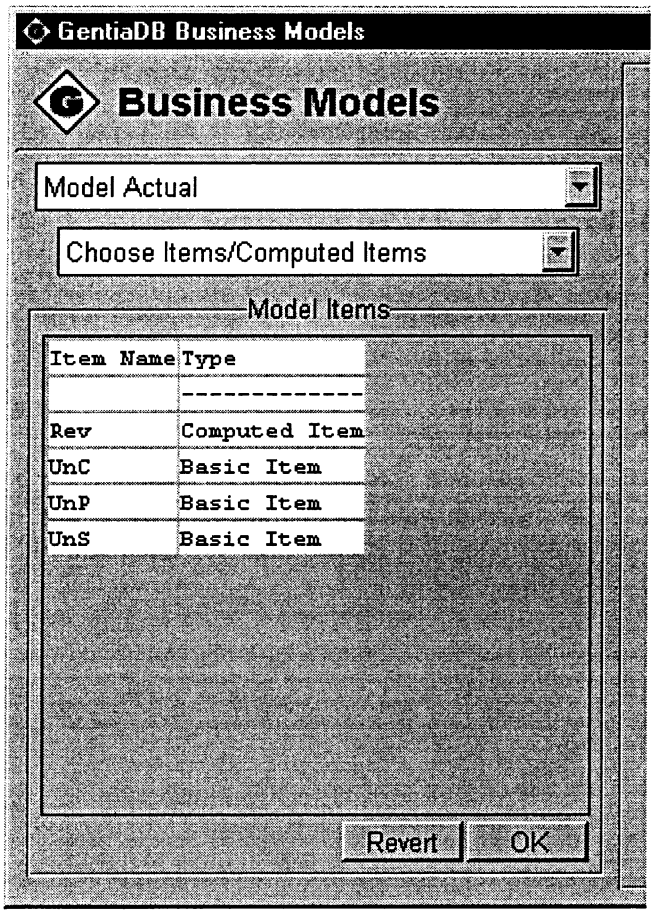
Practical 7.1 - Building a Base Model

- Select *Edit Models* form the lower pull-down menu.
- Click on the **New Model** button and enter the name of the base model, **Actual**.
- When you press enter, you should see the following:



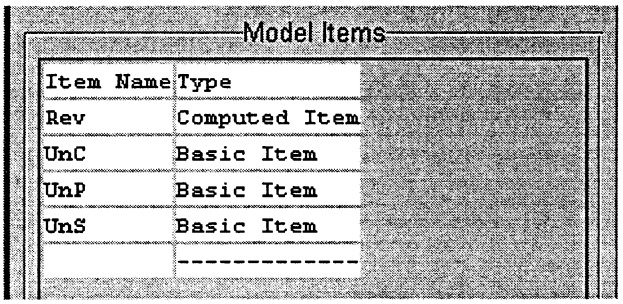
Specifying the Items

- From the lower pull-down menu, select *Choose Items/Computed Items*.



The Actual base model is to have all items. This is done by holding down the Ctrl key and dragging an item upwards so that it appears above the dotted line. All items above the dotted line will then be included in the base model. Alternatively, the dotted line can be moved downwards, by dragging its name field with the Ctrl key held down.

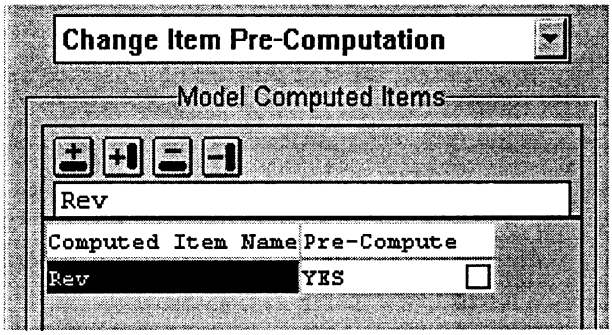
- Click on the item name field to the left of the dotted line, hold down the Ctrl key, and drag it down so that all the items appear above the dotted line.



- Click on OK to save.

Pre-Computed Items

- Select *Change Item Pre-Computation*.



This allows you to specify whether a computed value will be stored or whether it's to be calculated in-flight.

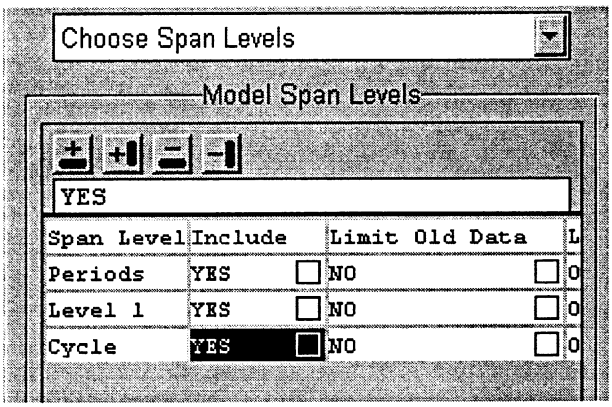
The default is NO, i.e. the item is not to be pre-computed, and would therefore be calculated on-the-fly; its value would not be stored.

YES means that the value would be pre-computed and therefore stored in the GentiaDB database.

- Change the *Pre-Compute* option to YES by clicking on the toggle box.
- Click on OK to save.

Choose Span Levels

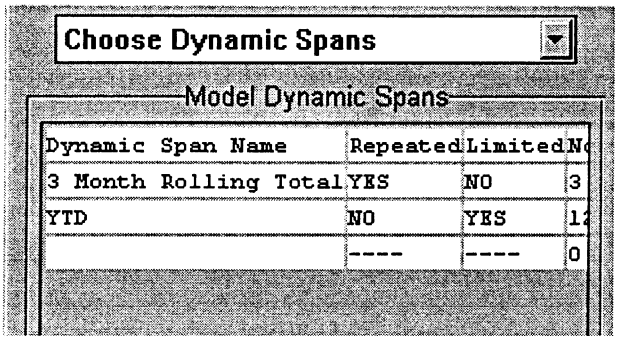
- To specify the time dimension members, select *Choose Span Levels* from the pull-down menu.
- Set the *Include* field to YES for all three span levels.



- Click on OK to save.

Dynamic Spans

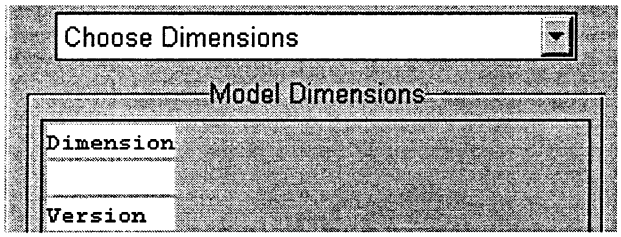
- Select *Choose Dynamic Spans* from the pull-down menu.
- Using the Ctrl key, drag the dotted line down so both dynamic spans appear above it, and will therefore be included in the Actual base model.



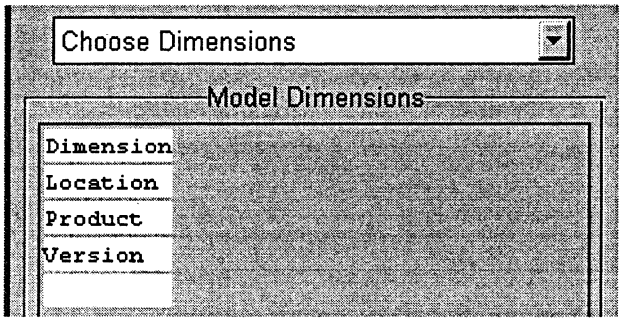
- Click on OK to save.

Choose Dimensions

- Select *Choose Dimensions* from the menu.



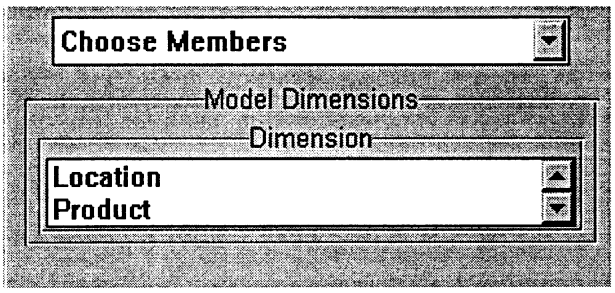
- Select all the dimensions by using the Ctrl key to move the blank row downwards so that all three dimensions appear above it.



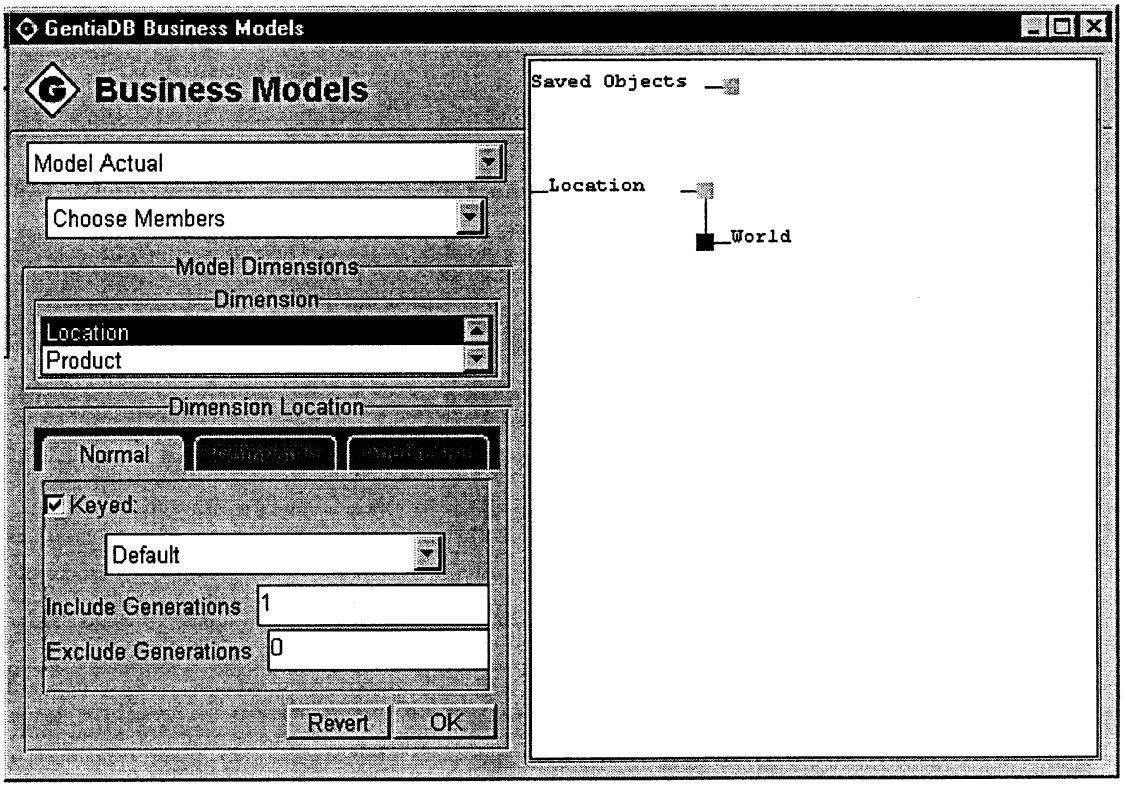
- Click on OK to save.

Choose Members

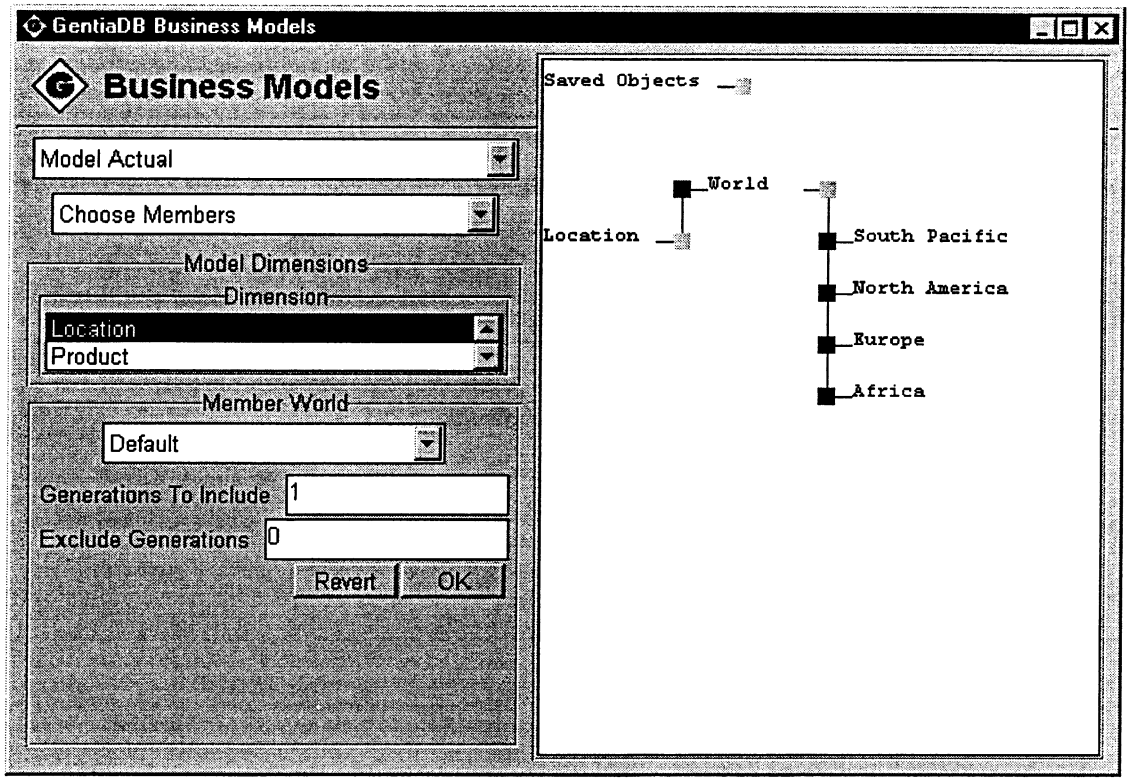
- Select *Choose Members* from the menu.



- Click on Location, which will display the structure of the dimension.

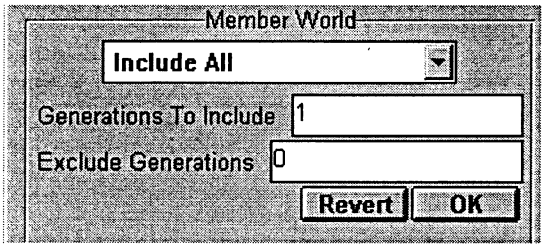


- Click on the name World.



In the hierarchy editor on the right-hand side, you should see that each member has a blue box, which was previously grey.

- In the bottom left-hand area of the window, select *Include All* from the pull-down menu.



- Click on OK.

The colour of the boxes by each member should change to green for World, and yellow for the other locations. The colour coding is as follows:

Yellow:	**member is included**
Green:	**start of a branch for which members are included**
Blue:	**excluded, or not currently selected member**
Black:	**start of a branch for which members are excluded**
Red:	**shows members included using the Include Only option**

- Repeat for the product dimension, using the *Include All* option.

- Repeat for the version dimension selecting *Actual* as the only member.

Validating the Base Model

The base model needs to be checked to ensure that it is valid.

- Click on the *Check Model* option in the pull-down menu.

You will then be presented with the status of the base model.

Model Name	State	Valid	Committed
Actual	Closed	Yes	No

If the *Valid* field does not say Yes, you have errors which need to be amended before proceeding.

- Select the *Actual* base model, by clicking on its name. This will give you any error log which may prove helpful if you do have errors.

Committing the Base Model

- Select the *Commit Model* option in the pull-down menu.

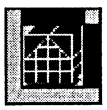
Viewing the Base Model

Even though the Actual Model does not yet contain data, it can still be viewed.

- Open book manager and select *Model Spec* from the object list.
- Open the Example1 model spec that you created earlier.

The business model and base model now need to be defined.

- From the *Business Model Service* menu, select the Business1 business model.
- Select the Actual base model from the *Model Menu* and close this window.
- Open the page called Example1, in the book called GentiaDB in builder mode.
- Open the connections mapper.

A business source connector  called **Business 1** has been positioned on the mapper area.

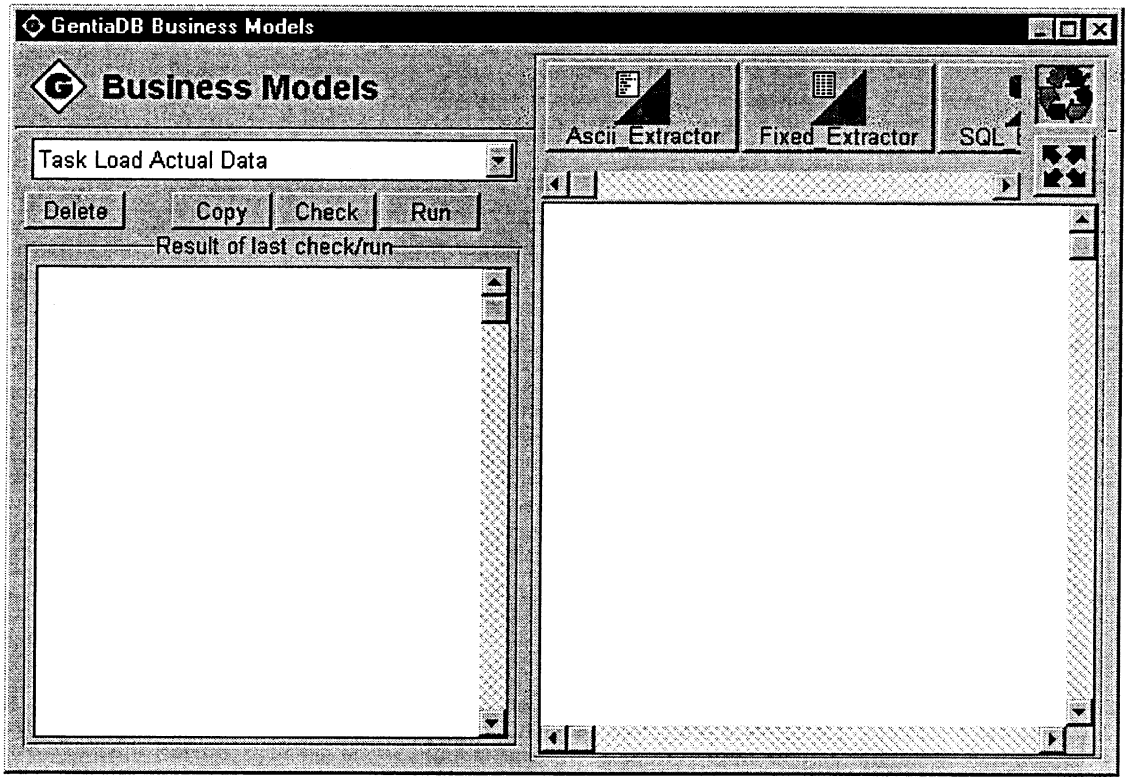
- Open the inspector for the business source.
- Specify the name of the model spec, **Example1**
- Connect the business source to the selector.
- Switch into user mode to view the structure of the base model and spend some time examining the structure of each dimension, to check that all the members have been defined properly.

8. Tasks - Data Loading

A task is used to perform a variety of GentiaDB-related functions, like loading dimension member details from a file and loading data into a base model. We will be using a task to load data to the Actual base model that has just been built.

A task is made up of a series of actions represented in GentiaDB by transformers. The actions are linked together, using the task mapper, which is rather like the connections mapper.

The task mapper looks like this:



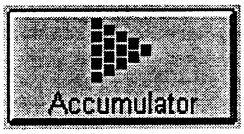
The transformers appear in the top right-hand part of the Window, and they can be dragged to the area underneath, where they can be linked together.

Below is a list and a brief description of each transformer.

Used to specify details of an ASCII file (CSV format) from which data is to be extracted.

Used to specify details of a fixed position text file from which data is to be extracted.

To specify details to extract from an SQL database.

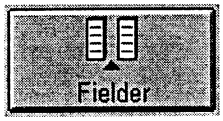


To aggregate values when incoming data contains records with multiple values for one or more coordinates

Used to perform currency conversions on data being imported.

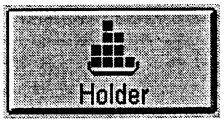
To add a text string to a field, either before or after the existing value.

To copy data from one field to another. The new field must be created beforehand.

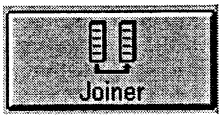


To add fields to incoming records.

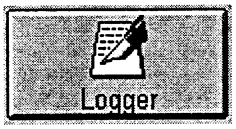
To link to a GDL module to perform transformations other than those provided by the standard transformers



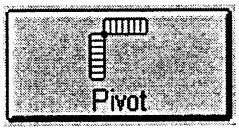
The Holder is the final transformer of any task. It is the object within the Business Model that holds the transformed data ready for loading.



Used to join data from two separate files.

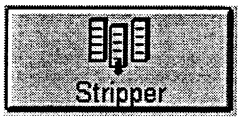


A logger can be placed at any point in the task, in order to view the output from another transformer, for checking purposes.

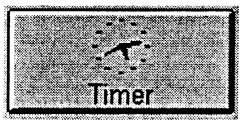


To switch data into correct format when across values in data file are not for the item members.

Used to validate data against an existing dimension.



Removes fields from incoming data records.



Used to validate details against the time dimension.

Practical 8.1 - Defining a Task to Load Actual Data

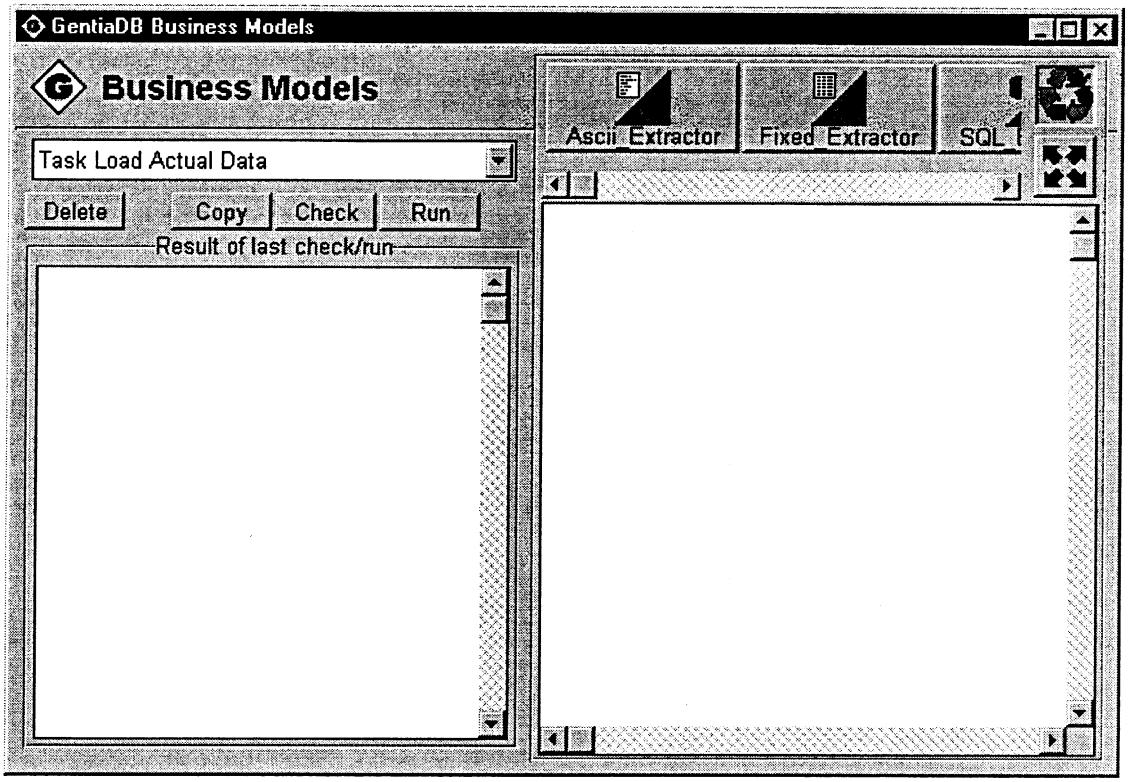
The data to be loaded is held in a csv file, and is specifically for the version dimension member Actual. The file, **c:\gentia\gentiadb\data\main.csv**, is structured in the following way:

```
Fri,Lon,Jan,504,694,38
Fri,Lon,Feb,504,694,36
Fri,Lon,Mar,504,694,50
Fri,Lon,Apr,504,694,27
Fri,Lon,May,504,694,13
Fri,Lon,Jun,504,694,18
Fri,Lon,Jul,504,694,19
Fri,Lon,Aug,504,717,19
Fri,Lon,Sep,504,750,22
Fri,Lon,Oct,504,750,26
Fri,Lon,Nov,504,750,41
Fri,Lon,Dec,504,750,45
```

Where: field 1 contains product information
field 2 contains location information
field 3 contains time information
field 4 contains UnC (unit cost) data
field 5 contains UnP (unit price) data
field 6 contains UnS (units sold) data

- Within the business model Business1, select *Edit Tasks* from the pull-down menu.
- Click on the 

button.
- Enter the name for the task, **Load Actual Data**, at the prompt, and press enter.

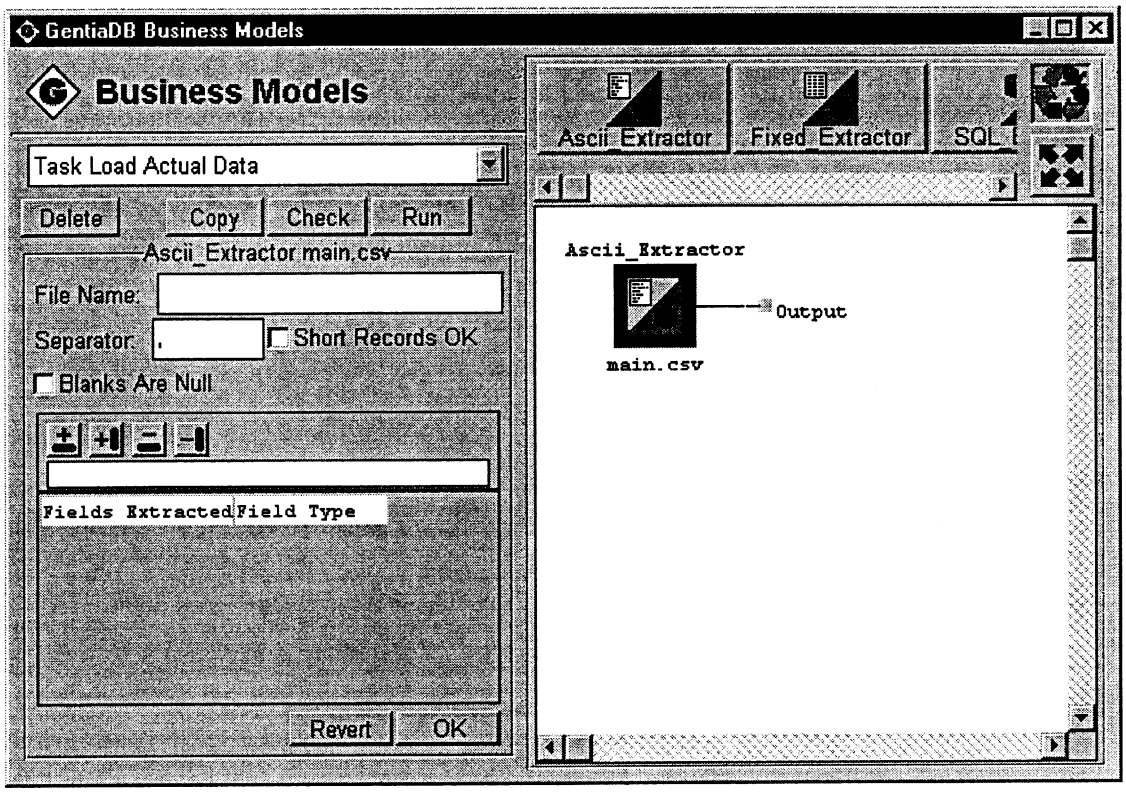


Specifying the data file

The first stage is to use an ASCII Extractor transformer to open the data file.

- From the scroll bar, drag the ASCII Extractor onto the connections area.
- At the name prompt, enter **main.csv**.

This is the name of the transformer, rather than the name of the data file. It makes sense to give the transformers meaningful names that reflect the role of particular transformers, at a glance. Hence, the name of the data file is an appropriate name for the transformer in this case.



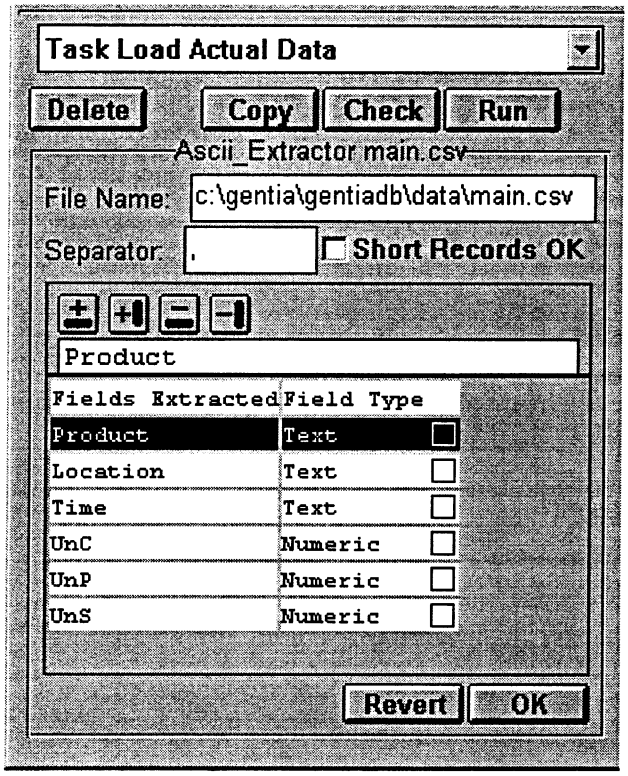
The area on the left-hand side is where the data file is identified.

- Enter the following details:

  File Name: c:\gentia\gentiadb\data\main.csv
  Separator: , (i.e. a comma)
  Short Records: leave unchecked

The Short Records option is used if the data file does not contain enough data for all the fields defined in the extractor. Checking this option will allow the data fields to be blank filled.

- Use the  button to specify the data fields in the same order as they appear in the data file, as follows:



- Click on the toggle box in the *Field Type* field 

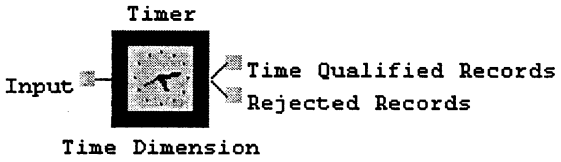
to switch between Text and Numeric.

- Click on OK to save the ASCII Extractor details.

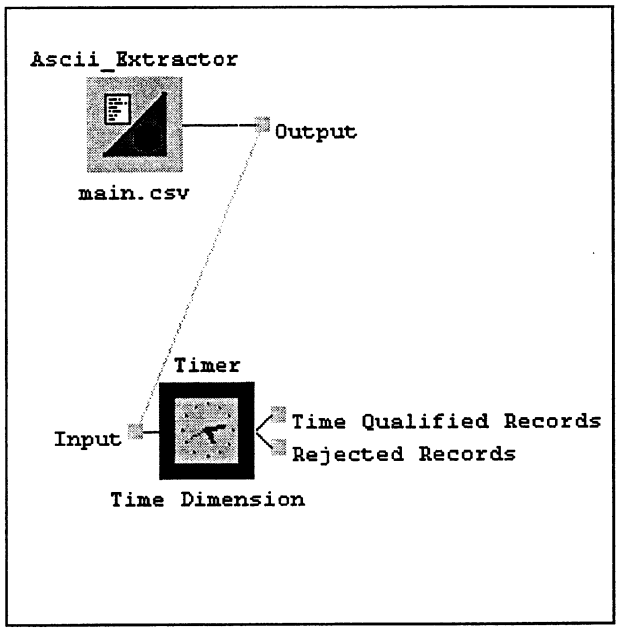
Validating the Time Dimension

The next stage is to validate the data in the file against the dimensions. We'll start with the time dimension, for which we need to use a Timer transformer. For the other dimensions, a Qualifier transformer will be used.

- Drag a Timer transformer onto the mapper area, and name it **Time Dimension**, or something similar.

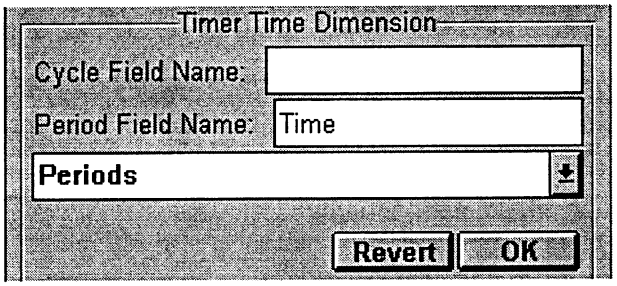


- Link the two transformers together by dragging the mouse to draw a line from the ASCII Extractor output to the Timer input, rather like making a connection in the connections mapper.



A link can be broken by re-drawing it.

- On the left-hand side of the window, fill in the following details:



Cycle Field Name: The name of the field in the data file whose details are to be validated against the defined cycles. If left blank, GentiaDB will add a field to each data record to identify the cycle as the current one.

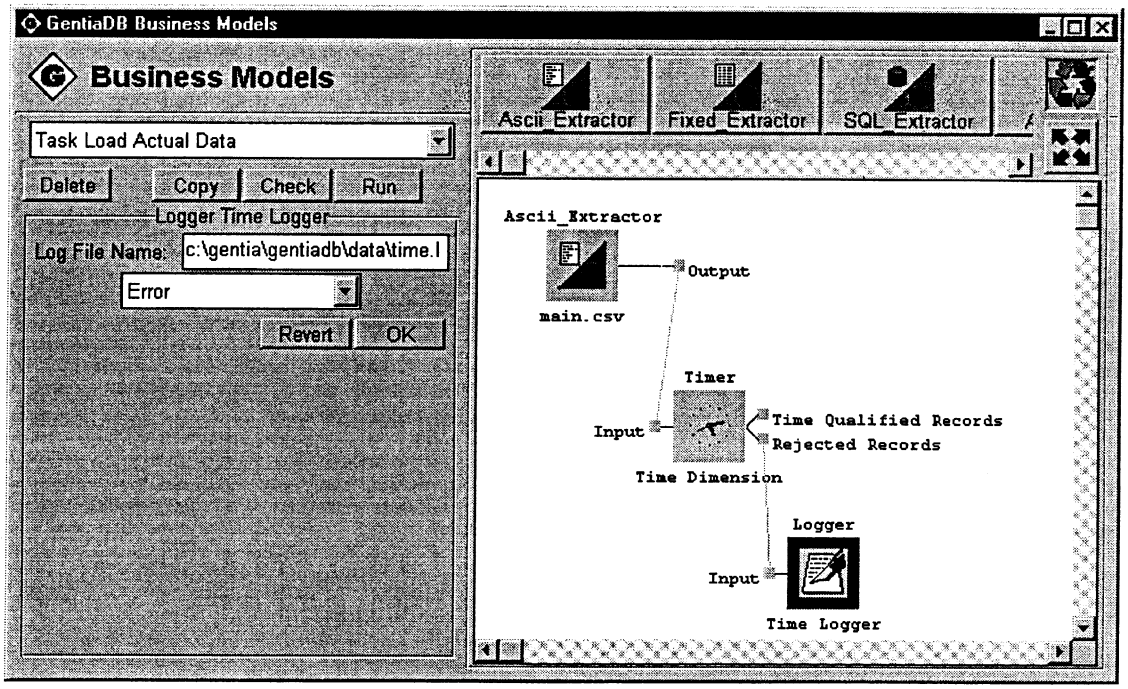
Period Field Name: The name of the field to be validated against the defined periods. If left blank, GentiaDB will add a field to each data record to identify the period as the current one.

List bar entry: To specify whether the data item in the *Period* should be validated against period names, span name, or cycle names.

- Click on OK to save the Timer transformer details.
- As you can see, the Timer transformer has two outputs: one for validated records, the other for rejected records.
- The rejected records can be sent to a Logger, in order to detect data failing to pass validation checks.

Logging rejected records

- Drag a Logger transformer onto the mapper area.
- Name it **Time Logger**.
- Connect the Timer and the Logger, and fill in the details on the left-hand side, as follows:



Log File Name: The name of the file in which details will be logged (time.log). This should include the path name.

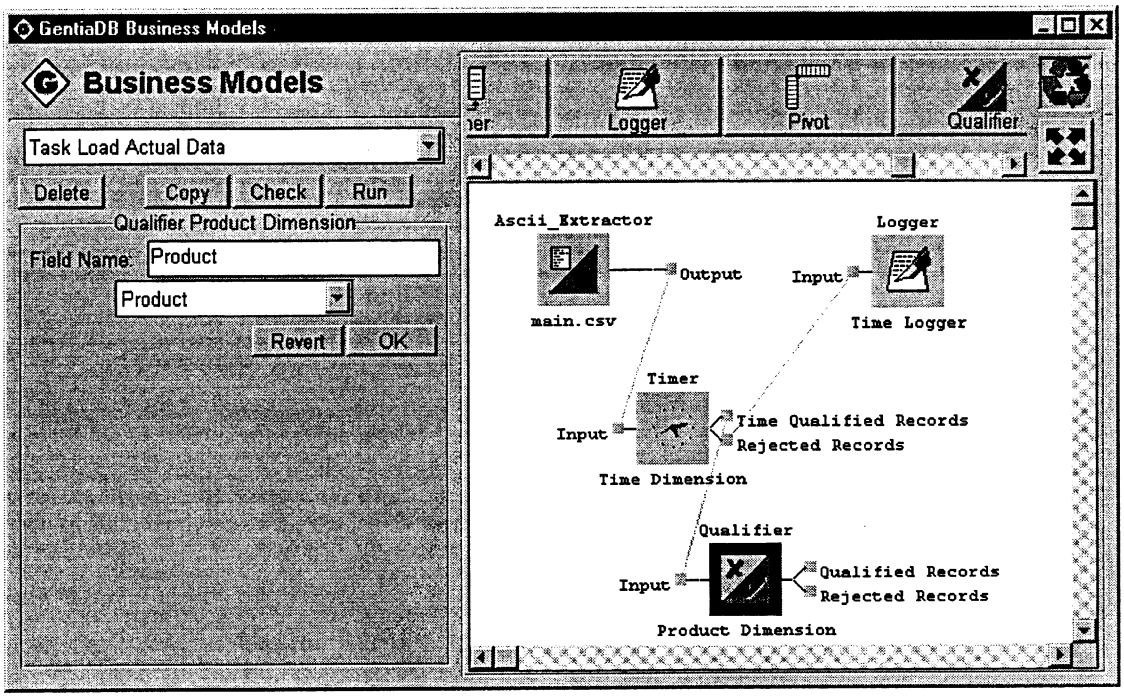
List bar entry: The logger type.
Error will halt execution of the task if an error is detected.
Warning will send a message to the Status panel, but will not halt the task.
Ignore will not interrupt processing, but will send a message to the log file.

- Click on OK to save the Logger details.

Validating the Product Dimension

The next stage in the task is to validate the data against members of the other dimensions. We'll start by validating the product dimension.

- Drag a Qualifier transformer onto the mapper area and name it **Product Dimension**, or something similar.
- Link the Timer (Time Qualified Records) output to the Qualifier Input, and fill in the details for the Qualifier as follows:



Field Name: This is the name of the field in the data file that requires validation.

List bar entry: The dimension against which the data is to be validated.

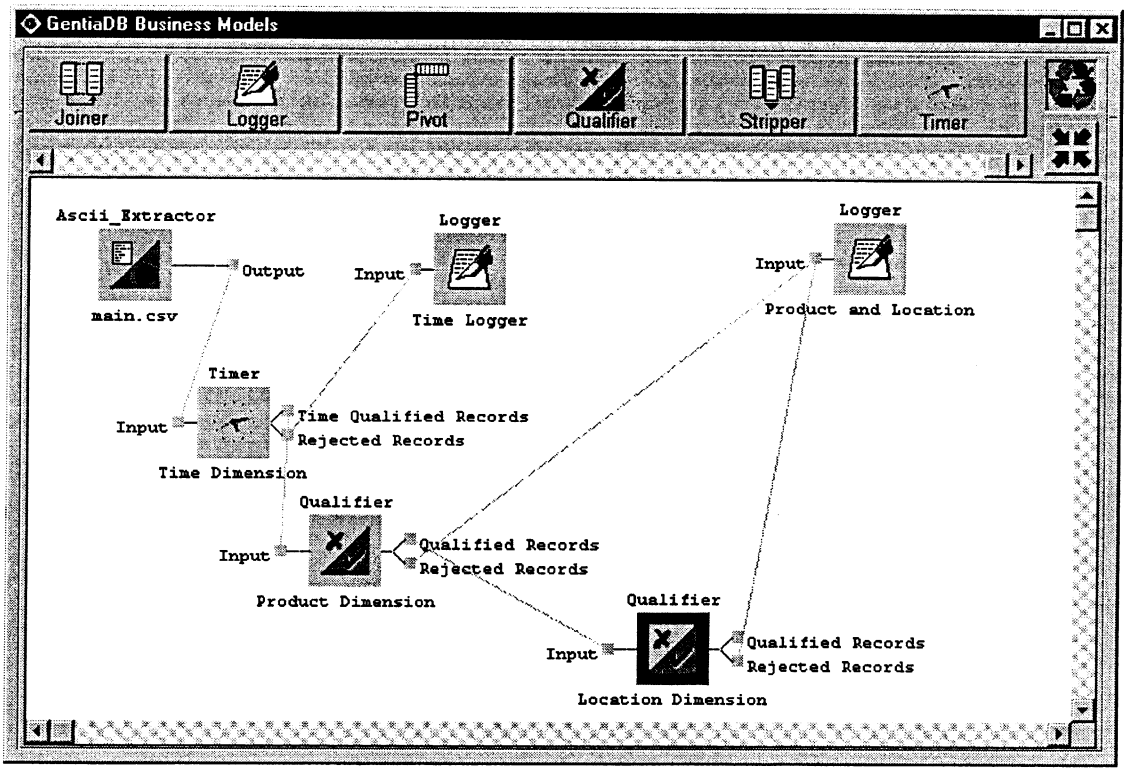
- Introduce a second Logger and connect it to the rejected records from the Qualifier

Location Validation

- Use another Qualifier to validate the data against the location dimension. You could also introduce a third Logger, for the rejected records; alternatively, you could send the output to either of the previous Loggers.

The mapper area can be expanded and restored by clicking on the  icon, in the top right-hand corner.

Your task should now look something like this:

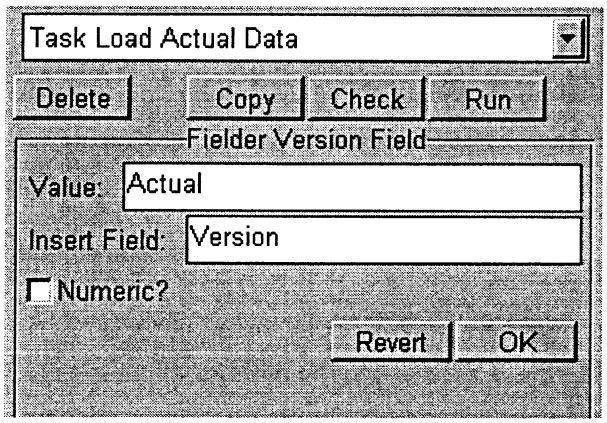


The task is now set up to validate the time, product and location dimensions.

The data file does not contain any data that relates to the version dimension, since the whole file represents Actual data. This information must be added to the task. This is done by adding a new field with a set value. The transformer that is used for this is called a Fielder.

- Drag a Fielder transformer onto the mapper area, give it a suitable name and connect it to the Qualifier for Location.

- Fill in the details for the Fielder (name of the field to be inserted and the value to be placed in that field for each record) as follows:

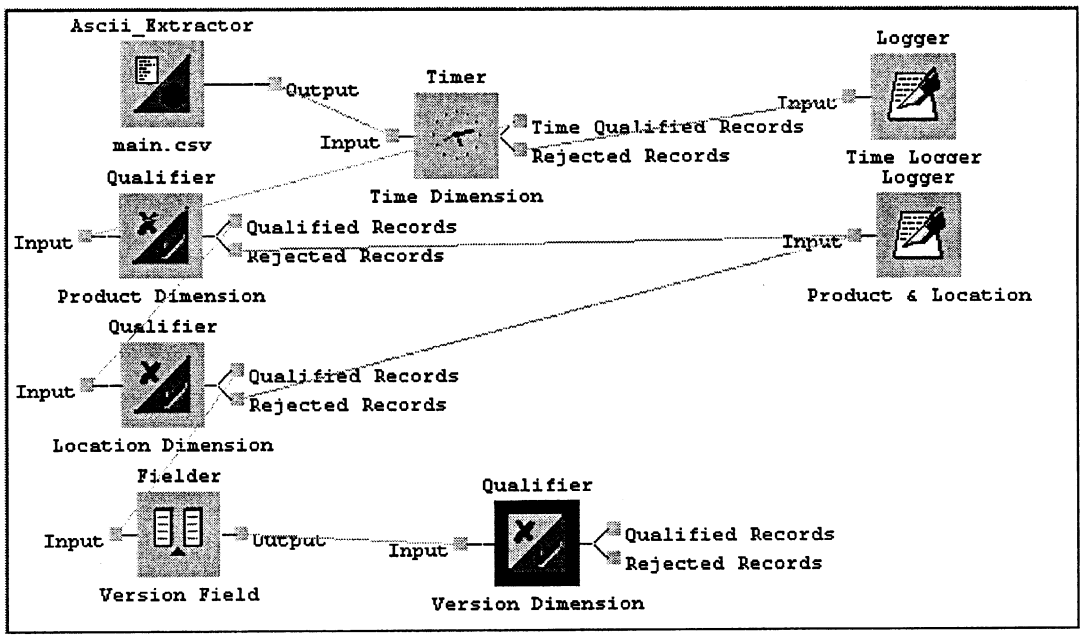


- Click on OK to save the details.

A Fielder must always be followed by a Qualifier to validate the new field.

- Introduce another Qualifier to follow the Fielder and then another Logger, or send the rejected records to an existing Logger.

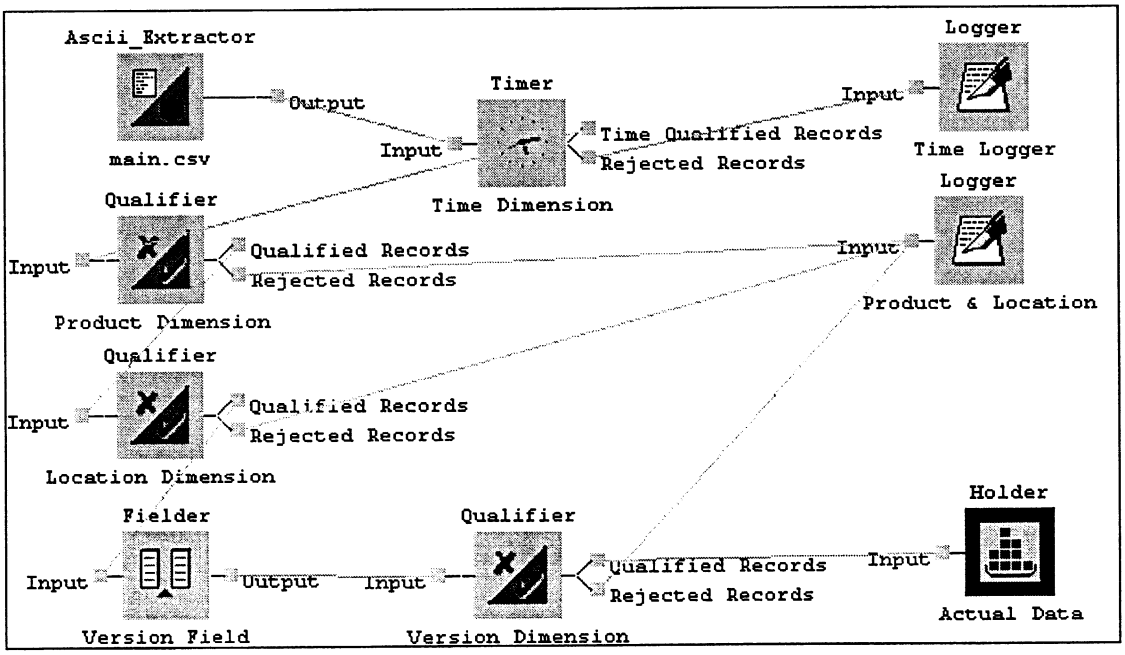
The task should now look something like this:



The final transformer is a Holder which will contain the final results. Every task must end with one and only one Holder. Once the task has run, the Holder will become populated with valid records.

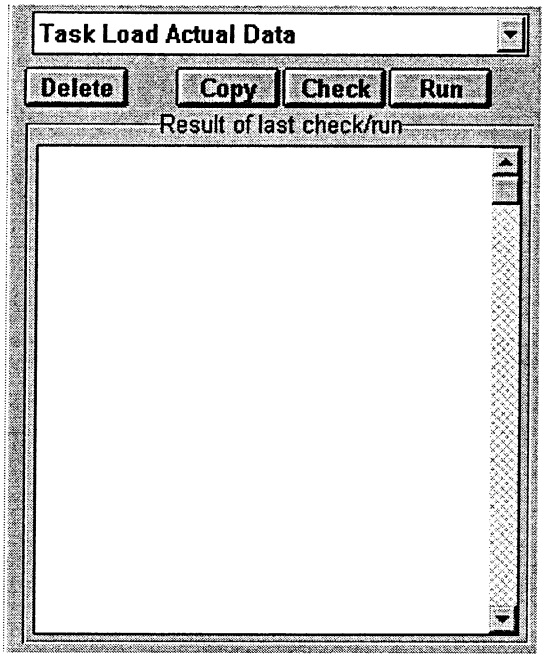
- Drag a Holder onto the mapper area and connect it to the last of the Qualifiers. No further information is required when specifying a Holder.

So, the completed task should now look like this:



Before the task can be executed, it must be checked for any errors.

Click on the Check button.

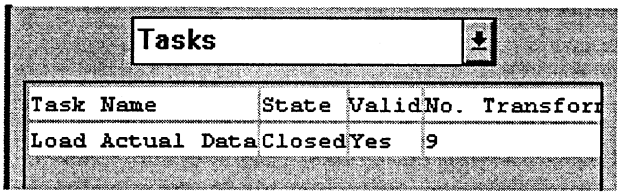


If there are any errors, they will be shown within the *Results of last check/run* window.

- If you have any errors, ensure that they are amended before continuing.
- Click on the Run button.

This will run the task and populate the Holder; the data will not yet be loaded into the base model.

Now that the task has been checked, its status should look like this:

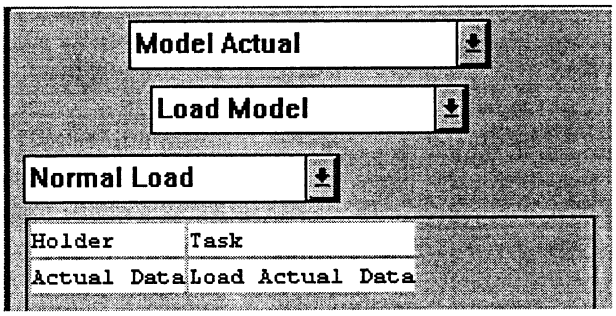


- Select the task again to check that no errors were produced during the run.
- If you have any errors, amend them now. If the error message refers to a Logger containing records, examine the log file specified for the Logger to try and determine where the error is.

Loading the Data

Since the data is to be loaded into a base model, the model needs to be opened.

- From the pull-down menu select *Business Model* Business1
- From the lower menu select *Edit Models* and then the model, Actual.
- Select *Load Model* from the menu, which will present you with a list of valid tasks and their associated Holders:



If you do not have any holders listed, it means that although your task has no syntax errors, it did not run properly. You will need to return to the task to correct the errors. Try using the Loggers to identify where things are going wrong.

- Select the appropriate Holder and click on the Start Load button to start the data load.
- To view the data, open the Example1 page as you did before and you should now see data.
- Spend some time examining the figures to ensure that all the data has been loaded properly.

9. Tasks - Defining Category Dimensions

We are now going to be focusing on a new example, to demonstrate some more features of Gentia. The business model is called Profit and is structured like the Business1 model. It has the same five dimensions (time, item, version, location and product), but the item and location dimensions have been changed and the time dimension is going to be split. In addition, we will look in more detail at tasks, jobs and transformers and also at rules.

Time Dimension (no change)

Year	Total Year
Qtr1	Quarter 1
Jan	January
Feb	February
Mar	March
Qtr2	Quarter 2
Apr	April
May	May
Jun	June
Qtr3	Quarter 3
Jul	July
Aug	August
Sep	September
Qtr4	Quarter 4
Oct	October
Nov	November
Dec	December

Item Dimension (new items; hierarchic display)

CostsTot		Total Costs	(Costs01+02+03+04)
Costs01		Staff Costs	(Costs011+012)
Costs011		Total Salaries	(Costs0111+0112+...+0117)
Costs0111		Directors Salary	
Costs0112		Senior Salary	
Costs0113		Middle Salary	
Costs0114		Junior Salary	
Costs0115		Clerical Salary	
Costs0116		Secretarial Salary	
Costs0117		Casual Salary	
Costs012		Overtime	
Costs02		Equipment	(Costs021+Costs023)
Costs021		Computer Hardware	
Costs022		Personal Computers	
Costs023		Photocopier	
Costs03		Admin Costs	
Costs04		Travel Costs	
Costs05		Salary PC of Costs	("Costs011" % "Costs Tot")
SalChg		Monthly Sales Changes	("UnS" - $TIME("UnS", -1))
CostsMean		Mean Cost Value	($MEAN($TIMES ("CostsTot", -12,12)))

CostsRound	Round Mean Costs	($ROUND("CostsMean",2))
CostsMax	Max Year Costs	($MAX($TIMES ("Costs Tot",-12,12))).
Prof	Profit	(Contrib-CostsTot)
Contrib	Contribution	(Rev-CoGS)
CoGS	Cost of Goods Sold	(UnC*UnS)
Rev	Revenue	(UnP*UnS)
UnC	Unit Cost	
UnP	Unit Price	
UnS	Units Sold	
RevPC	Revenue Percentage	(% VALUEOF("Rev", "World","TotProd"))
Staff01	Total Headcount	(Staff011+012+013+014+... ...+015+016+017)
Staff011	Directors	
Staff012	Senior Managers	
Staff013	Middle Managers	
Staff014	Junior Managers	
Staff015	Clerical	
Staff016	Secretarial	
Staff017	Casual Staff	

Version Dimension (one new dimension)

Actual	
Budget	
Variance (Actual-Budget)	
VarPC (Variance % Budget)	Variance Percentage
ActualYTD	Historical YTD

Location Dimension (extended to include a multiple hierarchy)

World	World
Eur	Europe
Lon	London
Ips	Ipswich
Utr	Utrecht
NthAm	North America
NY	New York
Atl	Atlanta
Den	Denver
SthPac	South Pacific
Syd	Sydney
HK	Hong Kong
Sin	Singapore
Afr	Africa
Joe	Johannesburg
Cap	Cape Town
Grp	Total Group
Sub	Total Subsidiaries

Lon	London
Ips	Ipswich
Utr	Utrecht
Atl	Atlanta
Den	Denver
Syd	Sydney
Joe	Johannesburg
Dis	Total Distributors
NY	New York
Sin	Singapore
HK	Hong Kong
Cap	Cape Town

NB. World and Total Group will both give total figures for the entire organisation.

Product Dimension (no change)

TotProd	Total Products
Whi	White Goods
Was	Washing Machines
Fri	Fridges
Bro	Brown Goods
Tel	Televisions
Vid	Videos
Hi	Hi-Fi's

Using a Task to Define Category Dimensions Members

Location Dimension

The location, product and version dimensions were defined in the Business1 model by simply typing in the members. This is not a very convenient method, since you could have hundreds, even thousands, of members for these dimensions. We will be using a task to define our dimensions instead, using data from a text input file.

It might be that you have access to a data file containing location information, which may have been produced from a particular database. Rather than typing in each of the member names, Gentia can use this information to generate the members for you. This is done by using a task.

In order to demonstrate how this works, we will focus on an example. The details of the location dimension are contained in a Fixed text file. We will use this data to load our dimension.

Gentia requires the information in the input text file to be structured in a particular way. The purpose of the task will therefore be to take the information in the text file and change its structure to suit Gentia's requirements before placing the information in a *holder*. The contents of the *holder* can then be loaded into the location dimension.

All fields must be named. Some fields have reserved names and some can have any appropriate name, such as Record Number. In addition to the list below, Expression, Consolidation (which is specific to Item) and Last Child Name (which is specific to time) are all reserved names and are mentioned in later sections.

The structure required by Gentia is as follows:

Name, [Parent Name], [Display Set, Display String], [Synonym Set, Synonym Name]

Name is mandatory
Parent Name is required for a hierarchic dimension (# represents the top of the hierarchy)
Display Set, Display String fields must both exist if Display Strings are to be loaded. To load default Display Strings, the Display Set field is left empty.
Synonym Set, Synonym Name fields must both exist if Synonym Names are to be loaded.

This input text file is a fixed file which means that the fields are not separated by commas but are in a specific position within the file. The width of the fields in the file is very important, as they will need to be input into the extractor to indicate where the data in each field is to be found. For instance, in the practical, the width of the Name field is 8; the Display String width is 24 and the Parent Name is 8.

In this example, we will be aiming to achieve the following structure:

Name, Parent Name, Display Set, Display String
The order in which these fields appear is not important.

The task will consist of three transformers; an extractor to take out the data, a fielder to add in the display set field (which is not present in the input file) and a *holder* to hold data until it can be loaded into the model.

Practical 9.1 - Task for Location Dimension

Objective

To create a task that will take information from the location input file and place it into a *holder*, ready to be loaded into the location dimension. Note that the information is stored in a fixed file.

The file is **(c:\gentia\gentiadb\data\LocDim.txt)** and is as follows:

World	World	#
Eur	Europe	World
NthAm	North America	World
SthPac	South Pacific	World
Afr	Africa	World
Lon	London	Eur
Ips	Ipswich	Eur
Utr	Utrecht	Eur
NY	New York	NthAm
Atl	Atlanta	NthAm
Den	Denver	NthAm
Syd	Sydney	SthPac
Sin	Singapore	SthPac
HK	Hong Kong	SthPac
Joe	Johannesburg	Afr
Cap	Cape Town	Afr
Grp	Total Group	#
Sub	Total Subsidiaries	Grp
Lon		Sub
Ips		Sub
Utr		Sub
Atl		Sub
Den		Sub
Syd		Sub
Joe		Sub
Dis	Total Distributors	Grp
NY		Dis
Sin		Dis
HK		Dis
Cap		Dis

i.e. the fields are **Name, Display String, Parent Name**

The Name field width is 8, the Display String width is 24 and the Parent Name width is 8.

The Display Strings are left out in the second branch of the dimension because they have already been defined in the first branch. Gentia will return an error if the Display Strings are repeated.

- The task will need to do the following:

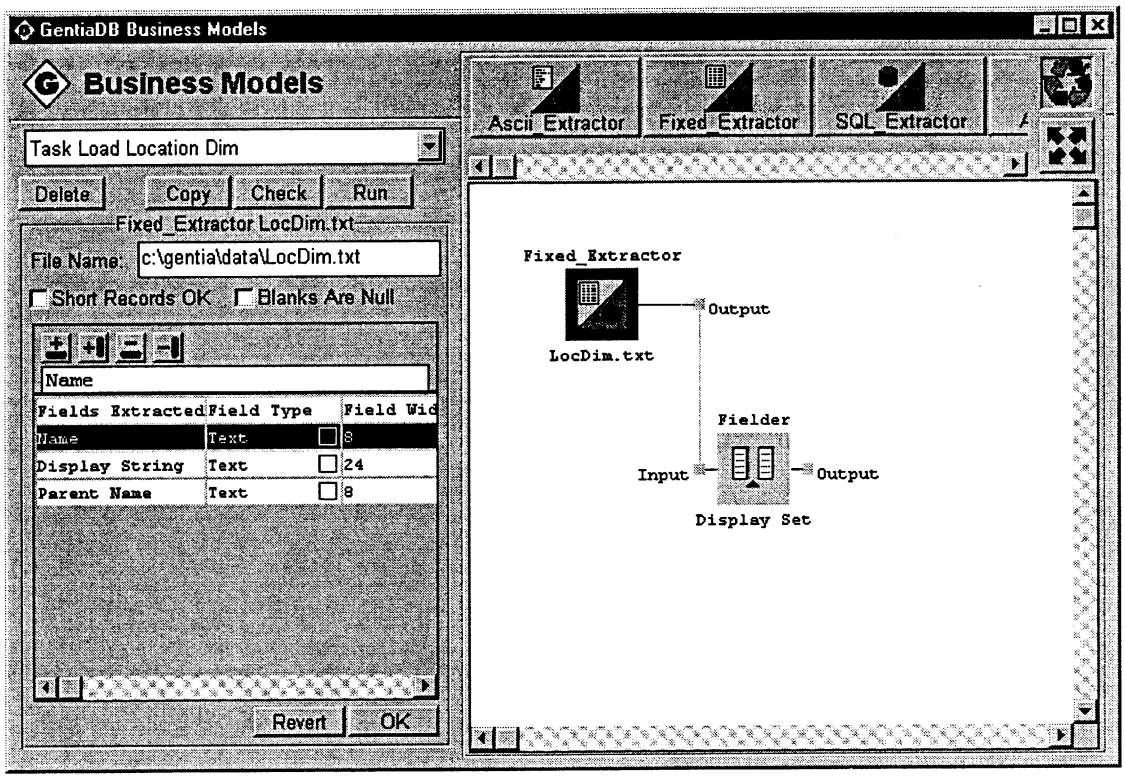
1. Open the Text file using a Fixed Extractor
2. Add the field Display Set using a fielder
3. Place the results into a *holder*.

To create a Task:

- Select the option *Edit Task.*
- Select the option *New Task.*
- Give the new task a meaningful name, for example **Load Location Dim.**

Once we have done this, we can choose the relevant transformer to carry out the relevant action within the task. The first transformer that is required is the Fixed Extractor. This will be set up with the text file name and the field structures.

- Drag the Fixed Extractor onto the mapper area, and give it the name: **LocDim.txt**
- Enter the file name: **c:\gentia\gentiadb\data\LocDim.txt.**
- Add **Name, Display String** and **Parent Name** in the *Fields Extracted* column. Ensure that all fields are Text type.
- Enter field widths according to the positions in the file.



A Fielder transformer is used to add the field **Display Set.**

- Drag a Fielder onto the mapper area.
- Enter **Display Set** into the *Insert Field* field.
- Leave the *Value* field blank so that the defined Display String will be the default.
- Connect the Extractor to the Fielder.

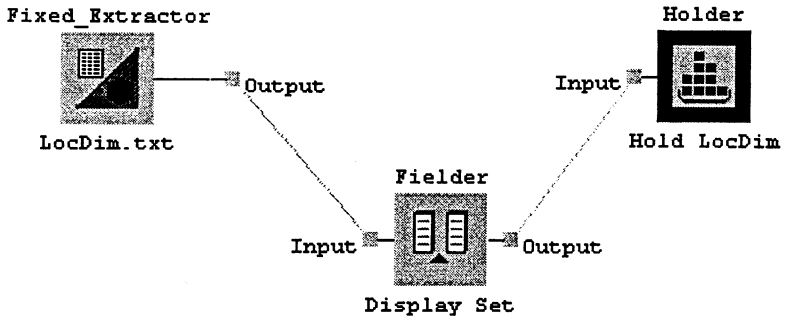
The final transformer that is required is the *holder.* This will contain the valid results.

- Drag a *holder* onto the mapper. This transformer requires no further input.
- Connect the Fielder to the *holder.*

The task now must be checked and then run to populate the *holder.*

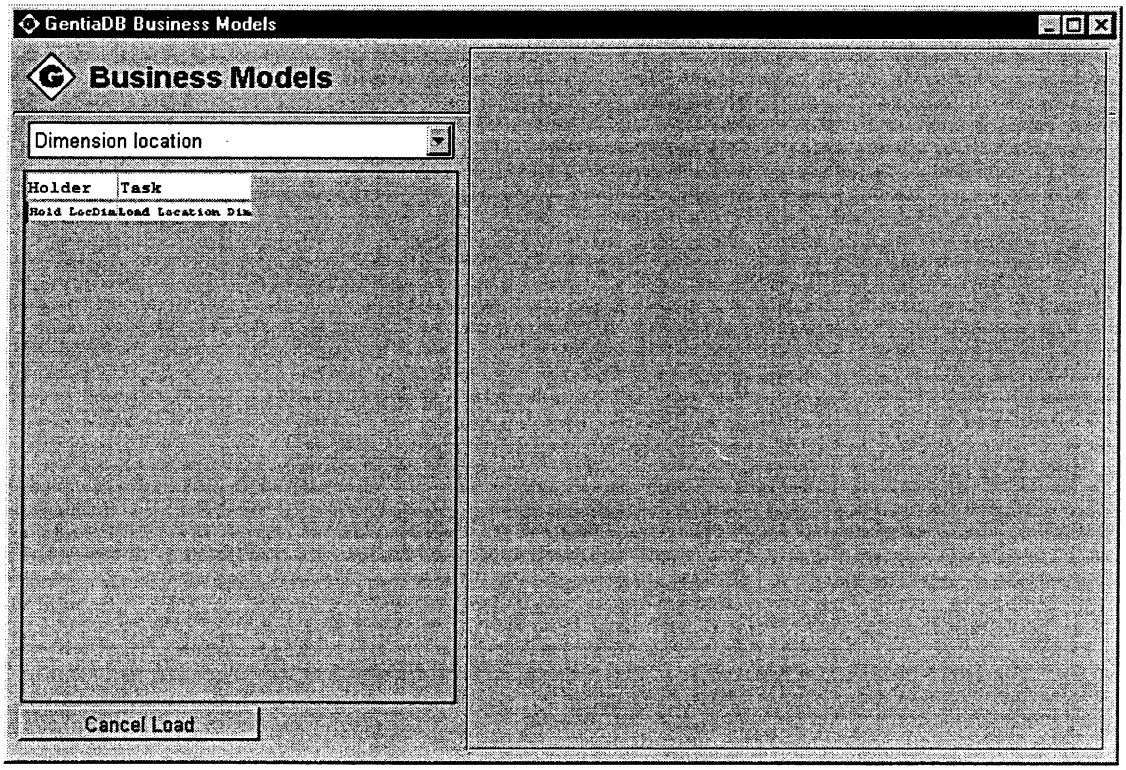
- Click on the *Check* button. If you have any errors, amend them before continuing.
- Click on the *Run* button. This will take you out of the task area.

Return to the task area in order to see if any error messages appeared as a result of running the task. Amend any errors before continuing.



Note. By running the task, the location dimension is not created. The data from the *holder* must be loaded into the location dimension from the *Edit Dimension/Member* option.

- From the drop down menu select *Business Model Profit.*
- Select *Edit Dimension/Member* from the bottom menu.
- Select *New Dimension* and enter **Location** in the *New Dimension Name* field.
- From within the location dimension screen, select the *Load* button.
- Click on the holder name for the *Load Location Dim* task.
- The location dimension should now be loaded.



- Spend some time examining the structure of the location dimension in the hierarchy editor, to check that it's as it should be.
- From the location dimension area, click on the Check button; Gentia will validate the dimension. You cannot commit a business model unless all dimensions have been checked.

Version Dimension

The next step is to create the version dimension. This will not be loaded by task because the dimension is very small and it contains a number of computed and aggregated members. **Computed or aggregated members cannot be loaded by task in category dimensions.**

Aggregated members

Aggregated members are very useful for calculating cumulative values, like year-to-date figures. We have already seen how you might use dynamic spans to achieve this. Let's consider the differences between the two methods, by looking at a year-to-date calculation for Actual data.

In the table below, the year-to-date calculation has been set up as a dynamic span, and is therefore presented within the Time dimension.

	January	February	March	April	YTD
Actual	27	36	36	35	134
Budget	25	30	38	38	131

Electrical ▽	World ▽

We can view YTD for Actual or Budget and any other members specified using the listbar. However, the YTD figures will always be in terms of the current period; we cannot see any historic YTD figures.

In this table, the YTD figures have been set up as aggregated category members of the Version dimension. This means that YTD figures will be shown for each period, therefore giving historic year-to-date data, which could be set up within any category dimension.

	January	February	March	April
Actual	27	36	36	35
Budget	25	30	38	38
Actual YTD	27	63	99	134
Budget YTD	25	55	93	131

Electrical ▽	World ▽

However, using aggregated category members will increase the size of the database since they cannot be calculated in-flight; the values must be held in the database.

The advantage of viewing the YTD in the time dimension is that it keeps the database from increasing unnecessarily, because no large amounts of data are being stored.

When defining an aggregated member, you will need to choose from one of three types of aggregation, as follows:

Normal aggregates over a specified span, e.g. moving 12 month total

Limited aggregation limited to current cycle, e.g. YTD

Average calculates average over specified span, e.g. moving 3 month average

You will also need to enter the number of spans over which the calculation will take place.

Inverted Calculations

In the case of a calculation such as Variance, the expression needs to be different, depending on the item for which Variance is being calculated.

For example:

Variance = Actual - Budget is appropriate for Revenue

but for costs, Variance = Budget - Actual is more appropriate.

To reflect this in Gentia, use the following expression: **Variance = Actual -? Budget**

The **?** will work in conjunction with items defined as *inverted,* to provide the correct result.

Functions

A member such as Percentage Variance can be calculated as follows:

Percentage Variance = Variance % Budget

or

Percentage Variance = (Actual-Budget) % Budget

The %% function provides a short-cut to the second expression.

i.e. Percentage Variance = Actual %% Budget since x %% y is equivalent to (x - y) % y

Practical 9.2 - Defining the Version Dimension

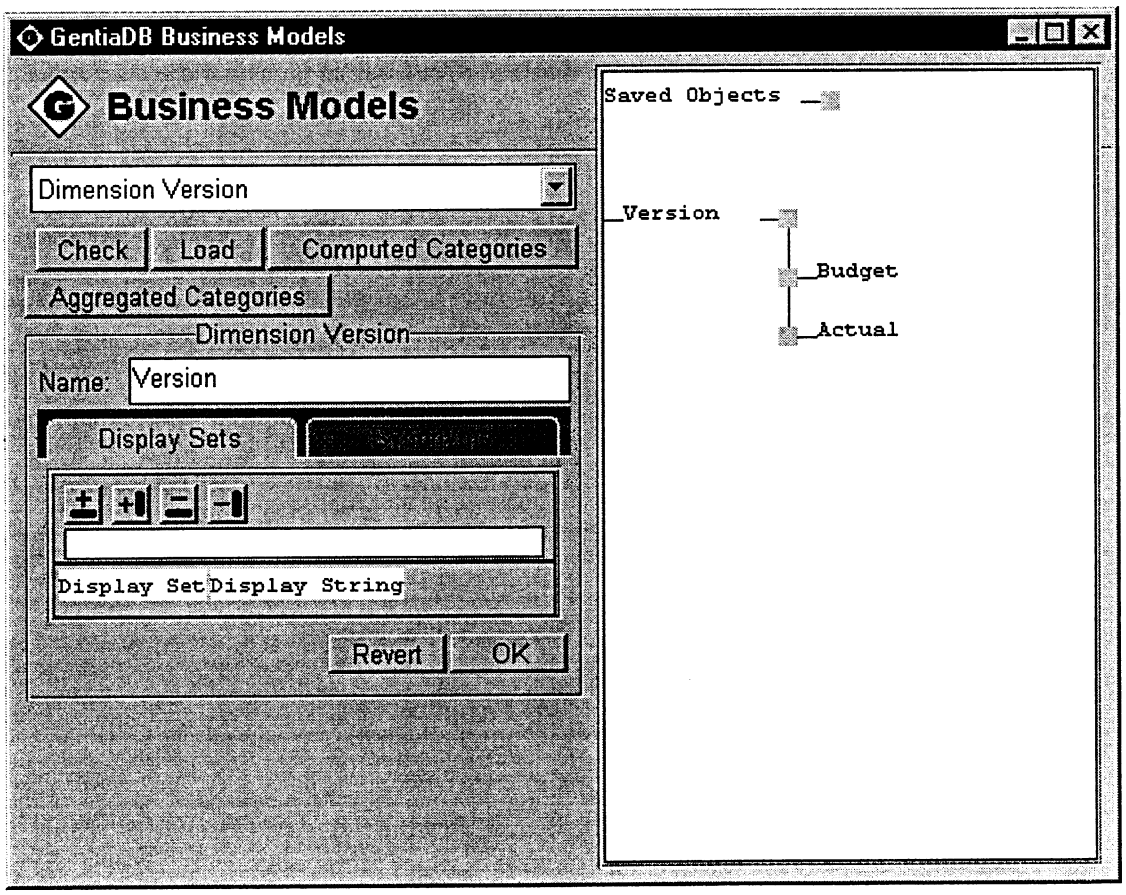
Objective

To define the version dimension, including the computed items and an Actual YTD member.

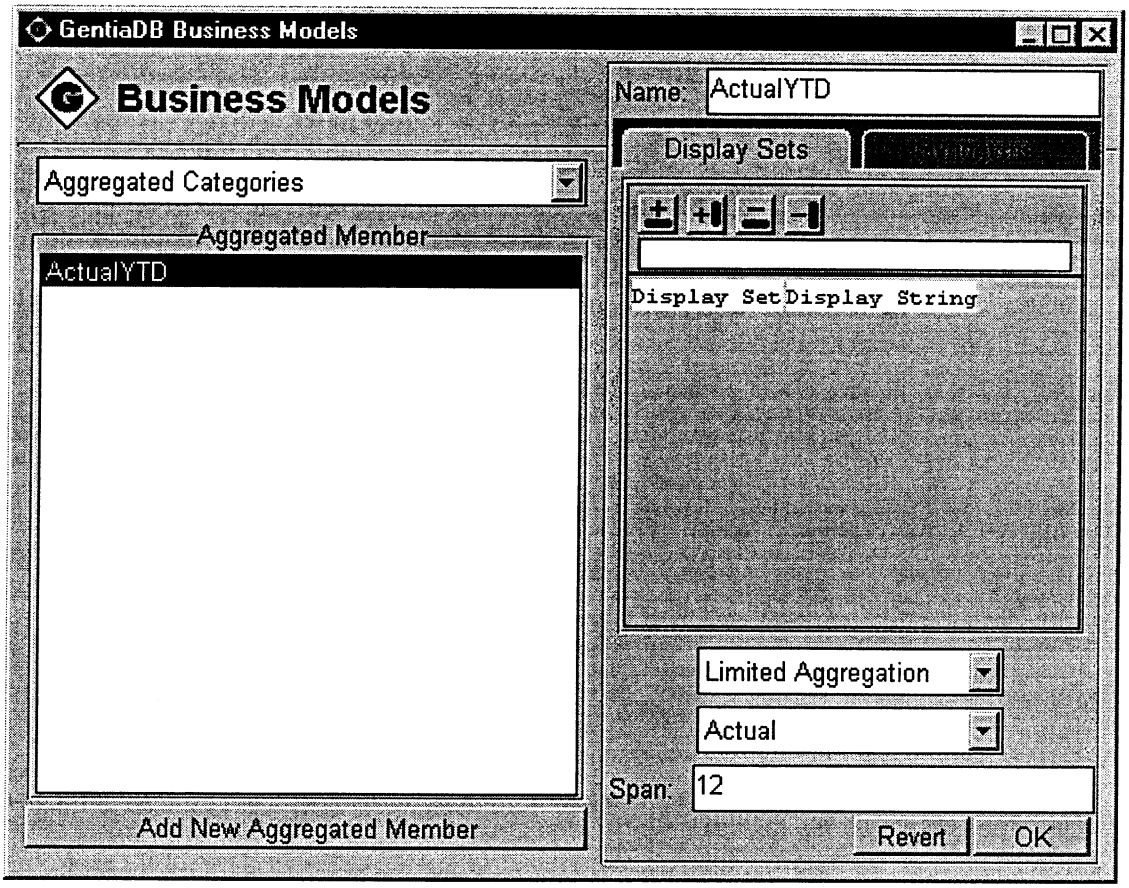
The version dimension will look as follows:

Name	Display String	Expression
Actual	Actual	
Budget	Budget	
Variance	Variance	Actual-?Budget
VarPC	Variance %	Actual%%?Budget
ActualYTD	ActualYTD	See below

- Using the information gained in the section 4, create the Version dimension adding in the computed categories.
- Create the Actual YTD member as follows:
  - Select the *Aggregated Categories* button



- Select the *Add New Aggregated Member* button, fill in the details of the member, setting the aggregation as limited, the target member as actual and the span as 12.



- Click on OK to save the changes.
- From the version dimension area, click on the *Check* button; Gentia will validate the dimension.

Product Dimension

The final category stage is the product dimension. The details of the product dimension are contained within two CSV text files. This is often the case when data is loaded from a database.

The first transformers that are required are the ASCII Extractors. There will be two of these, one for each input file. These will be set up with the text file names and the field structures. It is because there are two input files for product, that we will need to use a *joiner* transformer. Once we have the two ASCII Extractors on the mapper, we will need to add the *joiner* to merge the two files together. No new file is created at this stage but the merged data is ready to be used. In this instance we will be using the matched records from the *joiner* to go into the *holder*.

A *joiner* matches a field from one input file with a corresponding field from a second input file. It is important which file is matched first. For example:

Match This	***Match Against This***	***Match***
File 1	**File 2**	**Output**
Pen,Pens	Pen,Station,298,223	Pen,Pens,Station,298,223
Rul,Rulers	Pen,Station,887,745	Rul,Rulers,Station,552,967
Sta,Staples	Pen,Station,256,87	Sta,Staples,Station,88,22
Pri,Printer	Rul,Station,552,967	Pri,Printer,Electric,125,823
Rub,Station	Rul,Station,756,123	
	Sta,Station,88,22	
	Pri,Electric,125,823	

In the above example, file 1 is matched against file 2. The first 4 records in file 1 have been matched, resulting in 4 records in the output match. Now consider this variation:

Match This	***Match Against This***	***Match***
File 2	**File 1**	**Output**
Pen,Station,298,223	Pen,Pens	Pen,Pens,Station,298,223
Pen,Station,887,745	Rul,Rulers	Pen,Pens,Station,887,745
Pen,Station,256,87	Sta,Staples	Pen,Pens,Station,256,87
Rul,Station,552,967	Pri,Printer	Rul,Rulers,Station,552,967
Rul,Station,756,123		Rul,Rulers,Station,756,123
Sta,Station,88,22		Sta,Staples,Station,88,22
Pri,Electric,125,823		Pri,Printer,Electric,125,823
Rub,Station,645,774		

In this variation, each record in file 2 (apart from *Rub*) has been matched, resulting in a match of 7 records. The number of records in the first file to be matched, determines the number of records which are output. All records which do not find a match, will be output to the Unmatched records branch of the transformer. In the above two examples, Rub,Station would be output into Unmatched records.

There are five fields being input into the *joiner*. The Name field appears twice, however the *joiner* recognises this and automatically removes one of them.

The Rec No field is not needed by Gentia and must be removed. A *stripper* transformer is required. It simply contains a list of the fields which are to be removed.

Practical 9.3 - Task for Product Dimension

Objective

To create a task to join the data from 2 input files and populate a *holder*, ready to load the product dimension.

The first file is **(c:\gentia\gentiadb\data\Prod1Dim.csv)** and is as follows:

1,TotProd,#
2,Bro,TotProd
3,Whi,TotProd
4,Tel,Bro
5,Vid,Bro
6,Hi,Bro
7,Fri,Whi
8,Was,Whi

The fields in this text file are: **Record Number, Name, Parent Name**

The second file is **(c:\gentia\gentiadb\data\Prod2Dim.csv)** and is as follows:

TotProd,Total Products
Bro,Brown Goods
Whi,White Goods
Tel,Televisions
Vid,Videos
Hi,Hi-Fi's
Fri,Fridges
Was,Washing Machines

The fields in this text file are: **Name, Display String**

- Create a new task.
- Drag the ASCII Extractor onto the mapper area, and give it the name: **prod1.csv**
- Enter the file name: **c:\gentia\gentiadb\data\Prod1Dim.csv**.
- Add **Rec No, Name, Parent Name,** in the fields extracted column. The field containing the record number can have any name. Ensure that fields are Text type as necessary.

We must now choose another ASCII Extractor for the second input text file. This too will be set up with the text file names and the field structures.

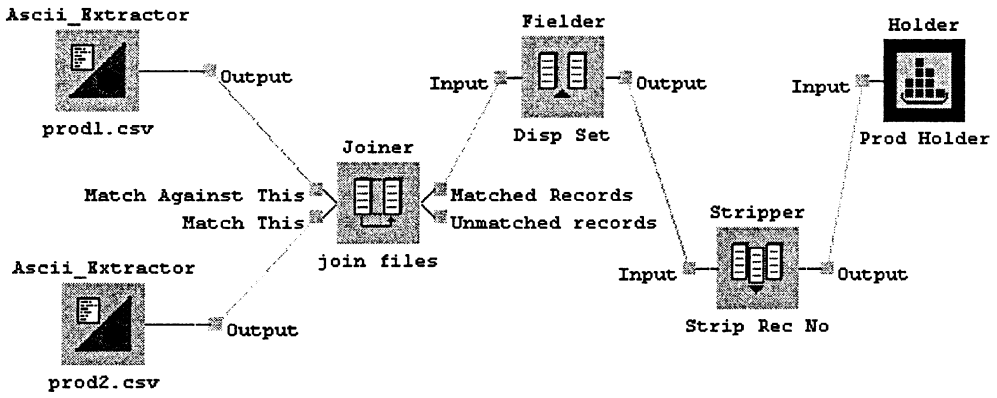
- Drag another ASCII Extractor onto the mapper area, and give it the name: **prod2.csv**
- Enter the file name: **c:\gentia\gentiadb\data\Prod2Dim.csv**.
- Add **Name, Display String,** in the fields extracted column. Ensure that all fields are Text type.

- Drag a *joiner* onto the mapper area.
- Link the two input files with the *joiner*.

- Add the Display Set to the task using a fielder.

- Use a *stripper* transformer to remove **Rec No**.
- Finish the task with a *holder*.

The task now must be checked and then run to populate the *holder*.



- Create the product dimension and then load the contents of the *holder* in order to generate the dimension members.
- From the product dimension area, click on the Check button; Gentia will validate the dimension.
- Spend some time examining the structure of the product dimension, to check that it's as it should be.

10. Tasks - Defining Item Dimension

Using a Task to Define Item Dimension

The Item Dimension

The item dimension members will be loaded by a task in linear format and then changed to hierarchy format at a later stage. The following members are computed as expressions in the input file; Costs01, Costs011, Costs02, Costs05, CostsTot, SalChg, CostsMean, CostsRound and CostsMax from the Costs section; CoGS, Prof, Rev, Contrib and RevPC from the Income section and Staff01 from the Personnel section. The calculations are as follows:

CostsTot is the total costs.
Costs01 is the total Staff Costs.
Costs011 is the total Salaries.
Costs02 is the total cost of equipment.
Costs05 is calculating the Total Salaries as a percentage of Total Costs.
SalChg shows the change in number of units sold over a 1 month period.
CostsMean is the average Total Costs for the last 12 months.
CostsRound is the CostsMean figure rounded to 2 decimal places.
CostsMax is the maximum Total Costs amount in the last 12 months.

CoGS is the unit cost multiplied by the units sold.
Revenue is the unit price multiplied by the units sold.
RevPC is the percentage of actual revenue for all products sold world-wide.
Contribution is revenue minus cost of goods sold.
Profit is the contribution less costs

Staff01 is the total number of working staff.

The consolidation attributes are set as follows:

Total and staff headcounts (staff011...etc.) to Full Consolidation: Time Latest Value.

Costs05, SalChg, CostsMean, CostsRound, CostsMax,RevPC, UnC and UnP to No Consolidation.

All the other items to Full Consolidation.

Expressions are rules or calculations and are used when loading computed members into the item dimension. They must be enclosed in double quotes if they contain member names which have spaces. The field is *Expression* which is a reserved name and must always be used.

The item dimension must be loaded as a linear dimension and then changed into a hierarchy for presentation purposes only, at a later stage, if so desired. Computed Items can also be loaded at this stage.

In addition, computed items must also have a field which explains their consolidation rules. This field is *Consolidation* which is a reserved word and is only associated with the item

dimension. The consolidation rule for an input file consists of a single letter. Use **F** for Full Consolidation, **L** for Full Consolidation: Time Latest Value, **H** for Dimension Only Consolidation and **N** for No Consolidation. The rules for consolidation are explained in Section 4.

There is also an inverted field. The word *invert* is a reserved word. Consider the following example.

Item Dimension: Revenue
Costs (inverted)

Version Dimension: Actual
Budget
Variance (Actual -? Budget)

	Actual	Budget	Variance
Revenue	70	50	20
Costs	50	40	-10

In the case of Revenue, the Variance is Actual - Budget as we would normally expect. However, with Costs the Variance is Budget - Actual. This is because if the actual costs exceed what was budgeted for costs then an adverse variance has been incurred. In the item input file the invert field is present and contains a **'Y'** if it is inverted. In the version file, the calculations have been made accordingly. When the field is referred to in an expression, it must include a question mark. When setting the inverted fields by hand, there is an inverted check box to tick.

The hierarchy is optional for the item dimension and is for presentational purposes only. Creating a hierarchy will have no effect on consolidations or computed items, it only affects the way in which the data is viewed. The hierarchy is applied when the base model is being set up, using the *Change Item Hierarchy* option. (See Base Model section later).

The file used for the practical is a csv file, however, the delimiter has been changed from a comma (,) to a pipe (|) symbol. This is because the expressions in the input file contain commas. If the delimiter was a comma then the number of fields would not be clear. The ASCII Extractor must be told that the separator is a pipe character and not a comma as expected.

Practical 10.1 - Task for Item Dimension

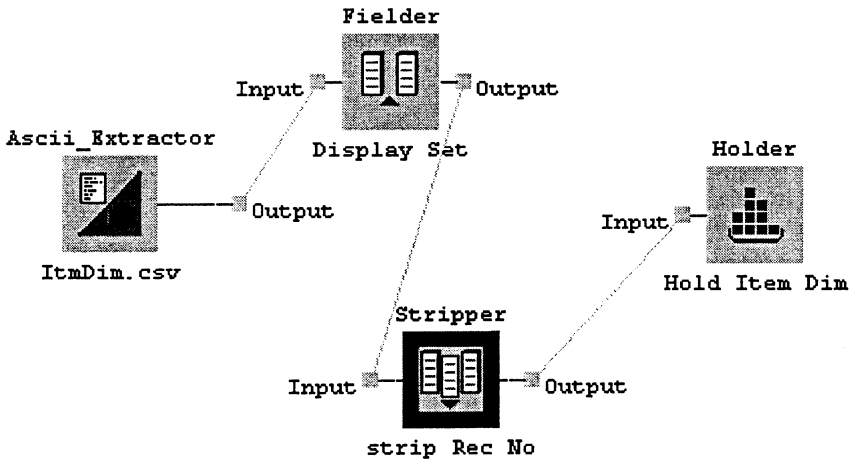
The Item text input file is **c:\gentia\gentiadb\data\ItmDim.csv** and is as follows:

```
5|Costs0111|Directors Salary|||Y
6|Costs0112|Senior Salary|||Y
7|Costs0113|Middle Salary|||Y
8|Costs0114|Junior Salary|||Y
9|Costs0115|Clerical Salary|||Y
10|Costs0116|Secretarial Salary|||Y
11|Costs0117|Casual Salary|||Y
12|Costs012|Overtime|||Y
14|Costs021|Computer Hardware|||Y
15|Costs022|Personal Computers|||Y
16|Costs023|Photocopier|||Y
17|Costs03|Admin Costs|||Y
18|Costs04|Travel Costs|||Y
29|Staff011|Directors||L|
30|Staff012|Senior Managers||L|
31|Staff013|Middle Managers||L|
32|Staff014|Junior Managers||L|
33|Staff015|Clerical||L|
34|Staff016|Secretarial||L|
35|Staff017|Casual Staff||L|
28|Staff01|Total Headcount|"Staff011+Staff012+Staff013+Staff014+Staff015+Staff016+
                                                    Staff017"|L|
4|Costs011|Total Salaries|"Costs0111+Costs0112+Costs0113+Costs0114+Costs0115+
                                                    Costs0116+Costs0117"||Y
3|Costs01|Staff Costs|"Costs011+Costs012"||Y
13|Costs02|Equipment|"Costs021+Costs022+Costs023"||Y
19|CostsTot|Total Costs|"Costs01+Costs02+Costs03+Costs04"||Y
36|Costs05|Salary PC of Costs|"Costs011 % CostsTot"|N|
21|UnC|Unit Cost||N|Y
22|UnP|Unit Price||N|
23|UnS|Units Sold|||
24|Rev|Revenue|"UnP*UnS"||
25|CoGS|Cost of Goods Sold|"UnC*UnS"||Y
1|Contrib|Contribution|"Rev-CoGS"||
26|Prof|Profit|"Contrib-CostsTot"||
37|SalChg|Monthly Sales Changes|"UnS"-$TIME("UnS",-1)"|N|
38|CostsMean|Mean Cost Value|$MEAN($TIMES("CostsTot",-12,12))|N|Y
39|CostsRound|Round Mean Costs|$ROUND("CostsMean",2)|N|Y
40|CostsMax|Max Year Costs|$MAX($TIMES("CostsTot",-12,12))|N|Y
41|RevPC|Income Value|$VALUEOF("Rev","World","TotProd")|N|
```

i.e. the fields are **Rec No, Name, Display String, Expression, Consolidation, Invert**

- Create a task that will generate the item dimension; it will need to do the following:

1. Open the Text file using a ASCII Extractor
    - *Short Records OK* must be selected. This is because it is acceptable for data records to have blank fields but not for dimension records. Switching on *Short Records OK* stops blank fields reporting an error.
    - Change the Separator to a | (pipe) sign.
2. Add a *fielder* to include the field Display Set.
3. Add a *stripper* to remove the Rec No. field
4. Add results to a Holder.



- Check and run the task. It would be wise to return to the task to check that the run did not produce any errors.

- Use the *Load Item Definition* option to load the contents of the holder in order to generate the item dimension members.

- Use the *Edit Items* and *Edit Computed Items* options to browse through the item members, to check the expressions, the consolidation rules, and whether or not the items are inverted.

11. Tasks - Defining Time Dimension

Using a Task to Define Time Dimensions Members

The next step is to create the time dimension. This will also be created by using a task. The time dimension, once committed cannot be changed, however new cycles can be added. In this example we will load the cycles by hand.

The time dimension file is set up differently from the other input files in that the # symbol is used to determine the *Last Child Name* as opposed to the highest level in the hierarchy. This is because the time dimension, unlike all other dimensions, is constructed from the bottom up. There is a far more rigid structure to the dimension, which does not support a multiple hierarchy. The previous business model saw us creating periods and spans manually by having two lists as shown below.

Period Names	Span Level 1
Jan	
Feb	
Mar	Qtr1
Apr	
May	
Jun	Qtr2
Jul	
Aug	
Sep	Qtr3
Oct	
Nov	
Dec	Qtr4

The span level 1 name appears alongside the last period to be included in the span (known as the *Last Child Name*); Qtr1 next to Mar, Qtr2 next to Jun etc. When periods and spans are being created from the data in an input file, one field must contain the *Last Child Name*. If there is no *Last Child Name* (i.e. for the periods in a span) then the # symbol is used.

i.e.
Jan, January, #
Feb, February, #
Mar, March, #
Qtr1, Quarter 1, Mar

When loading the Time dimension, the required fields are:

Name, Display String, Last Child Name and Display Set

Cycles

In the base practical, we will be loading data into two cycles; 1996 and 1997. The first, 1996, is not the current cycle and we must therefore use a *fielder* to specify the cycle value. The data being loaded will be at span level (i.e. Qtr's) for budget data and period level (i.e. month's) for actual data. The second cycle, 1997, is the current cycle and will have data

loaded at period level for actual data and span level for budget data. No *fielder* is necessary as the default cycle is the current one.

It is perhaps worth mentioning that although in this example periods are months, spans are quarters and cycles are years, this is not always the case. The time structure is very much dependant on the needs of the individual company. In certain circumstances these members can change to hold virtually any time measurement.

Splitting the Time Dimension

It is possible to split the time dimension as mentioned in Section 2. By default, the dimension is in linear format as in the example below, where the 1996 data is displayed first, followed by the 1997 data. When displayed as one dimension, time looks like any other hierarchy. Notice that each dimension has its own area on a listbar. As there are five dimensions, there are five areas on the listbars.

Item ▽ Time ▽

Item	J	F	M	Q1	A	M	J	Q2	J	A	S	Q3	O	N	D	Q4	1996	J	F
X	2	1	3	6	2	1	4	7	2	2	3	7	2	6	1	9	29	2	4
Y	3	3	3	9	1	4	5	10	2	2	6	10	2	3	2	7	36	2	2
Z	1	1	1	3	2	2	3	7	2	4	6	12	1	2	1	4	26	2	5

Total Prod ▽ World ▽ Actual ▽

Now consider splitting time so that it becomes two dimensions. If there are two dimensions, then the dimension which has been split from the remaining elements is known as the Cycle dimension. In this example, the Cycle Dimension is years. The remaining spans and periods are called the In-Cycle dimensions (in this instance, months and quarters). When splitting the time dimension, the listbar provides us with an extra dimension reference to access this aspect of the dimension. The Cycle dimension forms the rows and the In-Cycle Dimension forms the columns.

Cycle D ▽ In-Cycle D ▽

		J	A	S	Q3	O	N	D	Q4	Cycle Sum Member In-Cycle Dimension	J	F
1995	..	2	2	3	7	2	6	1	9	31	2	6
1996	..	4	3	6	13	3	4	6	13	44	5	2
1997	..	2	2	2	6	1	1	1	3	23	3	2

Total Prod ▽ World ▽ Actual ▽ UnS ▽

Additionally, if you split the time dimension the new member at the top of the In-Cycle dimension should be given a name. By default, it is the Cycle Sum Member In-Cycle Dimension. It is perhaps a better idea to rename the split dimension parts from within the *Edit Specials* option.

In the above example, Cycle Dimension might be Year, In-Cycle Dimension might be Periods and Cycle Sum Member In-Cycle Dimension might be Year Total.

Practical 11.1 - Task for Time Dimension

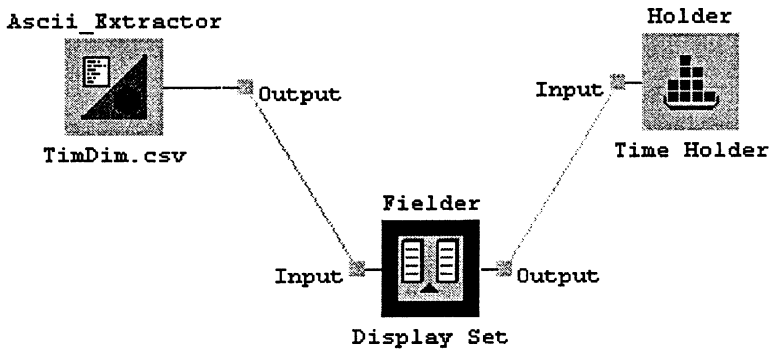
The time text input file is **c:\gentia\gentiadb\data\TimDim.csv** and is as follows :

Jan,January,#
Feb,February,#
Mar,March,#
Qtr1,Quarter 1,Mar
Apr,April,#
May,May,#
Jun,June,#
Qtr2,Quarter 2,Jun
Jul,July,#
Aug,August,#
Sep,September,#
Qtr3,Quarter 3,Sep
Oct,October,#
Nov,November,#
Dec,December,#
Qtr4,Quarter 4,Dec

i.e. the fields are **Name, Display String, Last Child Name**

- Create a task that will generate the time dimension; it will need to do the following:

1. Open the Text file using an ASCII Extractor
2. Add the field Display Set
3. Add results to a *holder*.



- Check and run the task. It would be wise to return to the task to check that the run did not produce any errors.

- Select LoadTime Definition and load the contents of the holder in order to generate the dimension members.

- Add 2 cycles to the business model, **1996** and **1997**.

- Add the same dynamic spans as in the Business1 practicals, i.e. *3 Month Rolling Total* and *YTD* (see page 5-2).

- From the *Edit Specials* option, change the display string for the 3 cycle names to more suitable alternatives.

- Assign new display strings to Item Dimension and Time Dimension, if you wish.

The business model is now complete and can be committed Once committed, it cannot be changed. However, cycles must be added, by choosing the *Edit Cycle Details* from the *GentiaDB Business Models Manager*. This can be done by creating a task. In addition, Dynamic spans can be added at this stage, if they are wanted.

- Select the option *Close Current Version*.
- Select *Commit version Zero*.
- You will be prompted for the current period; set this to **May 1997**.

You have now created a business model.

12. Creating Another Base Model

Copying Tasks

The next stage is to create two base models: one to store monthly actual data for 1997 and 1996, the second to hold quarterly budget data, also for both years. Four tasks need to be created to load data into each of the models. To save time, the ASCII Extractor already exists for the first task. Once you have fully created this task, use the Copy Task facility to duplicate the task for the other three loads. A task is simply copied by selecting the Copy Task button from within the current task. Gentia then prompts you to give the new task a name. The task for the first load is called actual 96. Do not forget to change the transformers and holders to reflect the names of the new input files.

Item Dimension Hierarchy

Once the models have been finished, the Item dimension hierarchy can be established if it is wanted. The item dimension is the only dimension whose hierarchy cannot be established at creation time. Therefore, it can be done now. The way that the hierarchy is created is very similar to the way that the version dimension was created in Business 1 model.

Having set up the base models, the following practical will consist of setting up two model specs and having the option to split the time dimension switched on in one spec and switched off in the other. By creating two tables and two list bars on a page, (one from either spec) we will be able to view the difference.

Pivot Transformer

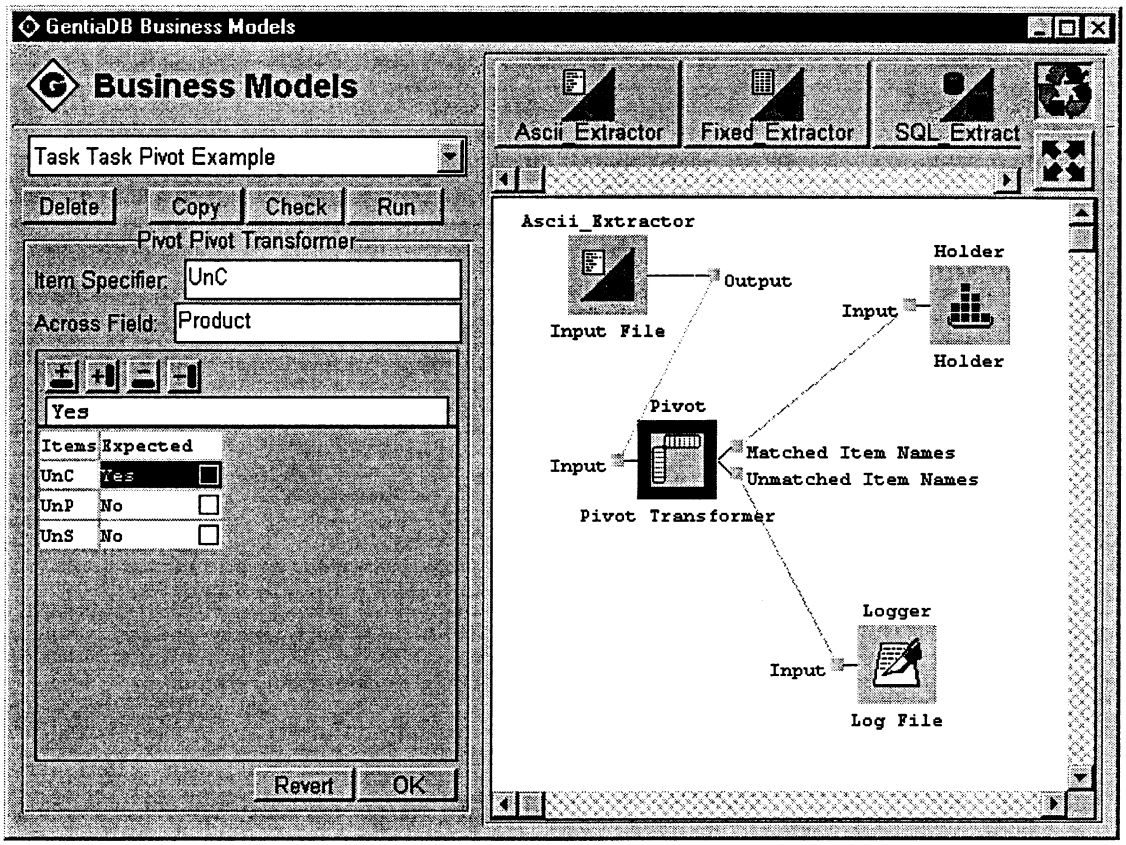
The information which is being input must be in a certain order for the item dimension. The across values given, must be the numeric item dimension data. For example, we would expect the input data to look as follows:

					Item		
	Prod	**Loc**	**Time**	**Vers**	**UnC**	**UnP**	**UnS**
Data	Fri	Lon	Jan	Actual	694	736	38
	Was	NY	Jan	Actual	668	755	33
	Vid	Sin	Jan	Actual	661	798	54

However, occasionally the data might be input in a different format where the numeric data belonged to another dimension:

					Prod		
	Item	**Loc**	**Time**	**Vers**	**Fri**	**Was**	**Vid**
Data	UnC	Lon	Jan	Actual	694	668	661
	...	...	...	...	...	...	...
	UnP	NY	Jan	Actual	736	755	798
	...	...	...	...	...	...	...
	UnS	Sin	Jan	Actual	38	33	54
	...	...	...	...	...	...	...

If the information was in the above format, we would use a *pivot* transformer. This would take the information and swap it around so that the numeric data was item information. The *pivot* transformer does not change the data itself, simply the order in which it arrives from the input file. An example of the above information might be as follows:



The *Item Specifier* box contains the name of the incoming field that holds the item name. The *Across Field* contains the name of the dimension against which the item dimension is to be pivoted.

The *Expected* toggle box indicates which item members are expected in the incoming data.

Practical 12.1 - Creating Base Models

Objectives

The aim of this practical is to create two base models and edit the hierarchy of the item dimension. The first base model will contain the actual member from the version dimension, and all members from the remaining dimensions and the second will contain the budget member from the version dimension and will be populated with quarterly data for both years.

Creating the Actual base model

- Create a new base model.
- Manually select the items required.
- Select all item members.
- Make all computed items pre-computed (as opposed to calculated in-flight), except for RevPC which cannot be pre-computed.
- Select span levels.
- Select some dynamic spans.
- Select dimensions (Select all dimensions).
- Select members from each dimension, but in the case of the Version dimension, select only Actual and Actual YTD.
- Edit the item hierarchy to group the items into a suitable hierachy for ease of use (see below and do this now).
- Check the model.
- Commit the model.

Creating the budget base model

Once the Actual model has been created you will be required to create a budget model to hold the data in quarters. You can make a copy of the Actual model, rather than creating a completely new model. This works in the following way:

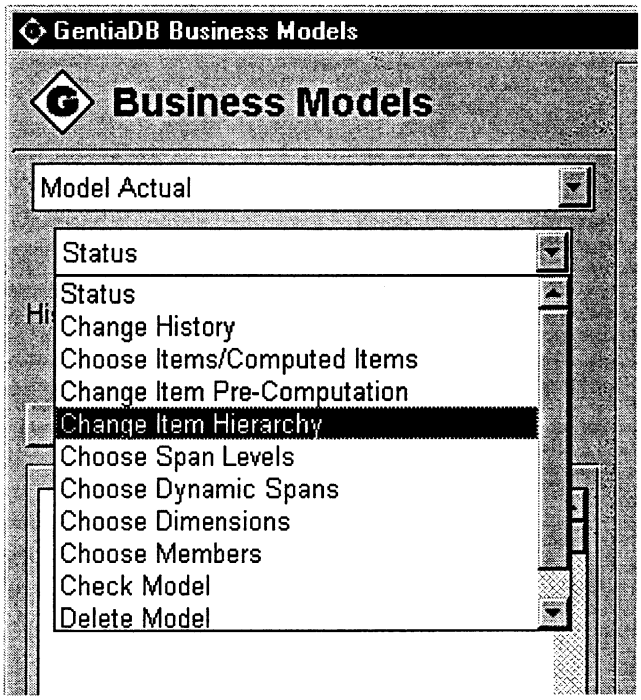
- With the actual model open, click on the *Copy Definition* button, which prompts for a model name.
- Enter the name, Budget; a new model has now been created which you will need to edit (see below).

Hints for the budget model build

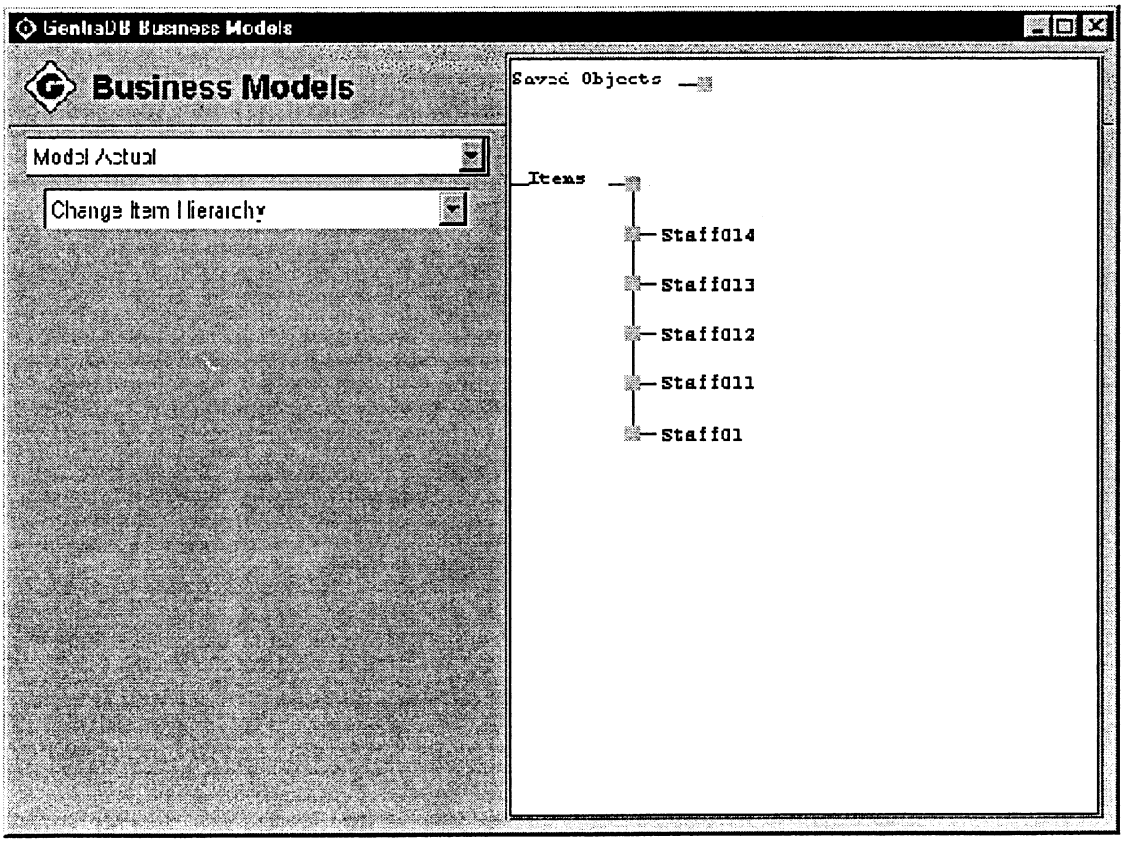
- When selecting the span for the model only include Level 1 (the quarters) and Cycle.
- From the version dimension only include the Budget member.
- Remember to deselect dynamic spans.

Changing the hierarchy for the Item dimension.

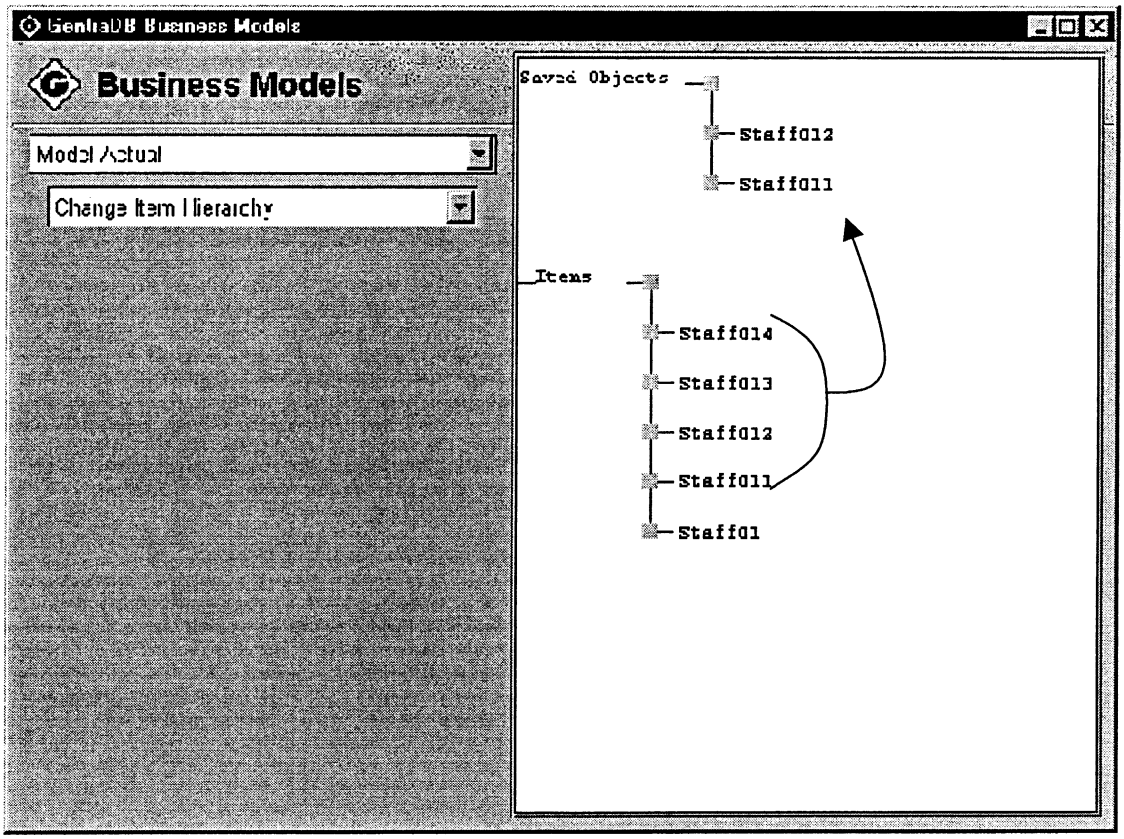
- First, select the *Change Item Hierarchy* option from the drop down menu.



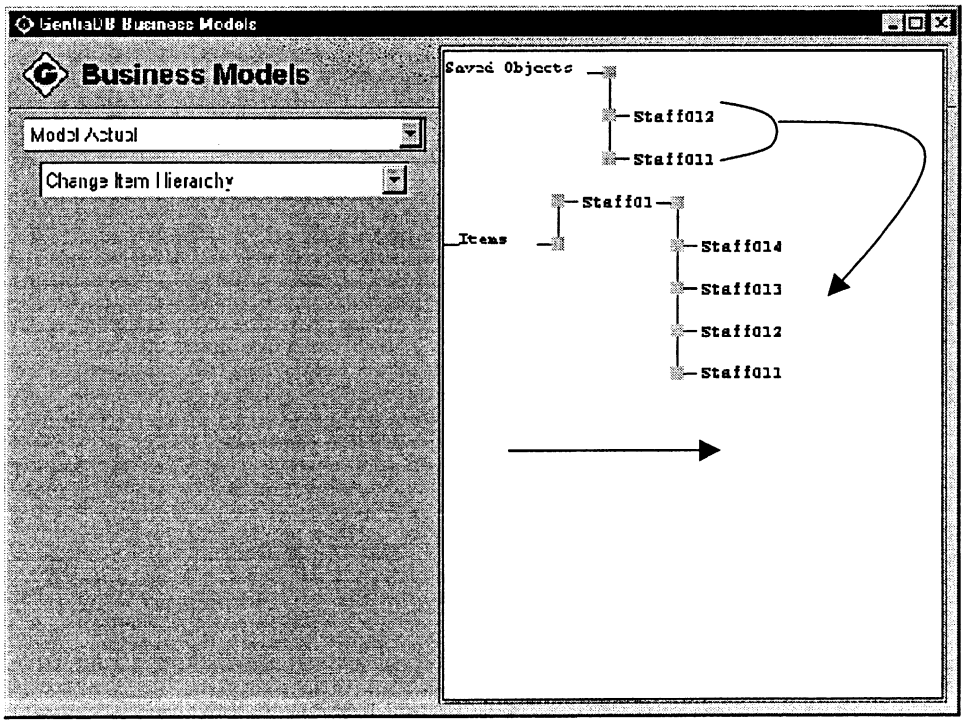
The screen which appears will show in linear format, all the members of your item dimension in one long list.



- Re-arrange the hierarchy by dragging the item members to *Saved Objects* for temporary storage.



- Drill down into the relevant branch and drag the item member from *Saved Objects* into its new location. To move all the item members at any one time, click and drag on the uppermost grey box to the right of the *Saved Objects* title.



Note that the item remains in the original list **and** in the *Saved Objects* list until they are deleted. To delete from the *Saved Objects* list hold down the Ctrl key and click on the item member. One click (with Ctrl key) on the uppermost grey box, deletes all members in the *Saved Item* area. To delete from the original list hold down the Ctrl + Shift keys and click on the grey box next to the item member.

Now that both base models have been created, checked and committed, the tasks which will load the data can be set up and run to populate the holders and subsequently the base models.

Practical 12.2 - Loading Base Model Data

Creating two tasks to load Actual Data

Task 1

The ASCII Extractor for the first task exists already and is called actual 1997. The data file for actual 1997 **(c:\gentia\gentiadb\data\Act97.csv)** is as follows:

```
Fri,Lon,Jan,100000,200000,300000,200000,40000,63000,24000,10000,10000
,20000,5000,2000,25000,489,673,37,1,4,10,10,5,7,4
Fri,Lon,Feb,100000,200000,300000,300000,40000,63000,18000,10500,10500
,23000,4000,2415,26374,489,673,35,1,4,10,15,5,7,3
Fri,Lon,Mar,100000,200000,300000,300000,40000,63000,18000,10500,11000
,21034,3243,2551,21423,489,673,49,1,4,10,15,5,7,3
Fri,Lon,Apr,120000,200000,300000,300000,40000,63000,18000,10500,14000
,23013,5342,1267,24132,489,673,26,1,4,10,15,5,7,3
Fri,Lon,May,120000,200000,300000,300000,40000,63000,18000,10500,10000
,19283,4231,1242,28795,489,673,13,1,4,10,15,5,7,3
... ...
```

i.e. the fields are: Product, Location, Time, Costs0111, Costs0112, Costs0113, Costs0114, Costs0115, Costs0116, Costs0117, Costs012, Costs021, Costs022, Costs023, Costs03, Costs04, UnC, UnP, UnS, Staff011, Staff012, Staff013, Staff014, Staff015, Staff016, Staff017

Transformers required

ASCII_Extractor (already exists)
Timer
Fielder (to add the Version dimension)
3 *Qualifiers*
Holder

Task 2

The data file for actual 1996 **(c:\gentia\gentiadb\data\Act96.csv)** is as follows:

```
Fri,Lon,Jan,504,694,38,80000,160000,170000,195000,35000,56000,24000,9
000,9000,16000,4000,1500,20000,489,673,37,1,4,10,15,5,7,4
Fri,Lon,Feb,504,694,36,80000,160000,170000,195000,35000,56000,24000,9
474,9263,15937,4001,1324,20988,489,673,35,1,4,10,15,5,7,4
Fri,Lon,Mar,504,694,50,80000,160000,170000,195000,35000,56000,24000,9
366,9253,16473,4352,1325,21764,489,673,49,1,4,10,15,5,7,4
Fri,Lon,Apr,504,694,27,80000,160000,170000,195000,35000,56000,24000,9
263,8696,15384,4098,908,27453,489,673,26,1,4,10,15,5,7,4
Fri,Lon,May,504,694,13,80000,160000,170000,195000,35000,56000,24000,9
253,8978,15243,3098,963,29758,489,673,13,1,4,10,15,5,7,4
... ...
```

i.e. the fields are: Product, Location, Time, Item X, Item Y, Item Z, Costs0111, Costs0112, Costs0113, Costs0114, Costs0115, Costs0116, Costs0117, Costs012, Costs021, Costs022, Costs023, Costs03, Costs04, UnC, UnP, UnS, Staff011, Staff012, Staff013, Staff014, Staff015, Staff016, Staff017.

The fields Item X, Item Y and Item Z contain data which is not needed. This data can be stripped out of the file using a *stripper*.

The data is to be loaded into the year 1996 which is not the current cycle. So, a *fielder* will need to be used, to include a field that contains the year. Also, the *timer* transformer will need to validate the year field. Remember to use the *Copy Task* button from the completed task (actual 1997) and name the new task actual 1996.

Transformers required

ASCII_Extractor
Stripper
2 *Fielders*
Timer
3 *Qualifiers*
Holder - since this must have a different name you will need to delete this and create a new holder.

- Check and run the tasks.
- Check on errors from the run by re-opening the task.
- Use the loggers to help de-bug any problems.
- To load the data, select the model created and click on the *Load* button. Select each of the tasks in turn.

Creating two tasks to load Budget data

Hints for the data load.

- The 1997 budget data is held in the following file:

c:\gentia\gentiadb\data\Bud97.csv which is structured as follows :

Fri,Lon,Qtr1,504,694,45,100000,200000,300000,200000,40000,63000,24000,13242,9000,200
00,5464,2132,24352,458,631,41,1,4,10,10,5,7,4
Fri,Lon,Qtr2,504,694,40,100000,200000,300000,300000,40000,63000,18000,10293,11000,23
000,3546,2435,23142,458,631,36,1,4,10,15,5,7,3
Fri,Lon,Qtr3,504,694,45,100000,200000,300000,300000,40000,63000,18000,10283,12000,21
034,5534,2235,23254,458,631,41,1,4,10,15,5,7,3
Fri,Lon,Qtr4,504,694,30,100000,200000,300000,300000,40000,63000,18000,10462,23013,23
013,4534,1553,26453,458,631,27,1,4,10,15,5,7,3
Fri,Ips,Qtr1,504,694,20,100000,200000,300000,300000,40000,63000,30000,10024,19283,19
283,6435,2143,27736,458,631,18,1,4,10,15,5,7,5...

i.e. the fields are: Product, Location, Time, Item X, Item Y, Item Z, Costs0111, Costs0112, Costs0113, Costs0114, Costs0115, Costs0116, Costs0117, Costs012, Costs021, Costs022, Costs023, Costs03, Costs04, UnC, UnP, UnS, Staff011, Staff012, Staff013, Staff014, Staff015, Staff016, Staff017.

- Data in positions 4, 5, and 6 is not needed. A *stripper* transformer can be used to remove any unwanted data.

- The 1996 budget data is held in the following file:

c:\gentia\gentiadb\data\Bud96 which is structured as follows:

Fri,Lon,Qtr1,240000,480000,510000,585000,105000,168000,71800,9000,9000,15000,3242,1234,21029,504,694,45,1,4,10,15,5,7,4
Fri,Lon,Qtr2,240000,480000,510000,585000,105000,168000,89800,9474,8837,16252,5243,1322,23142,504,694,40,1,4,10,15,5,7,5
Fri,Lon,Qtr3,240000,480000,510000,585000,105000,168000,53800,9366,9938,15000,4232,5142,28736,504,694,45,1,4,10,15,5,7,3
Fri,Lon,Qtr4,240000,480000,510000,585000,105000,168000,71800,9263,8383,16252,5342,726,26354,504,694,30,1,4,10,15,5,7,4
Fri,Ips,Qtr1,240000,480000,510000,585000,105000,168000,89800,9111,8838,17625,6352,938,21874,504,694,20,1,4,10,15,5,7,5...

i.e. the fields are: Product, Location, Time, Costs0111, Costs0112, Costs0113, Costs0114, Costs0115, Costs0116, Costs0117, Costs012, Costs021, Costs022, Costs023, Costs03, Costs04, UnC, UnP, UnS, Staff011, Staff012, Staff013, Staff014, Staff015, Staff016, Staff017.

- ASCII Transformers do not exist for either of these tasks, however, you can use the copy task button, to use the tasks which already exist.
- When setting up the *timer* transformer within the data loading task, remember that the data is not being loaded at period level but is being loaded at Level 1 of the time dimension.

You have now created a business model with two base models, both of which contain data.

Viewing the Data

A page has been set up to allow you to view the data in each of the models.

- Start by creating two model spec objects that point to each of the base models, Actual and Budget in the Profit business model.
- Open the **ActualBudget** page in the Profit chapter in the GentiaDB book.
- You will need to drag two business source connectors onto the connections mapper, specify a model spec for each source, and connect them accordingly.
- Switch into user mode and spend some time checking the data.

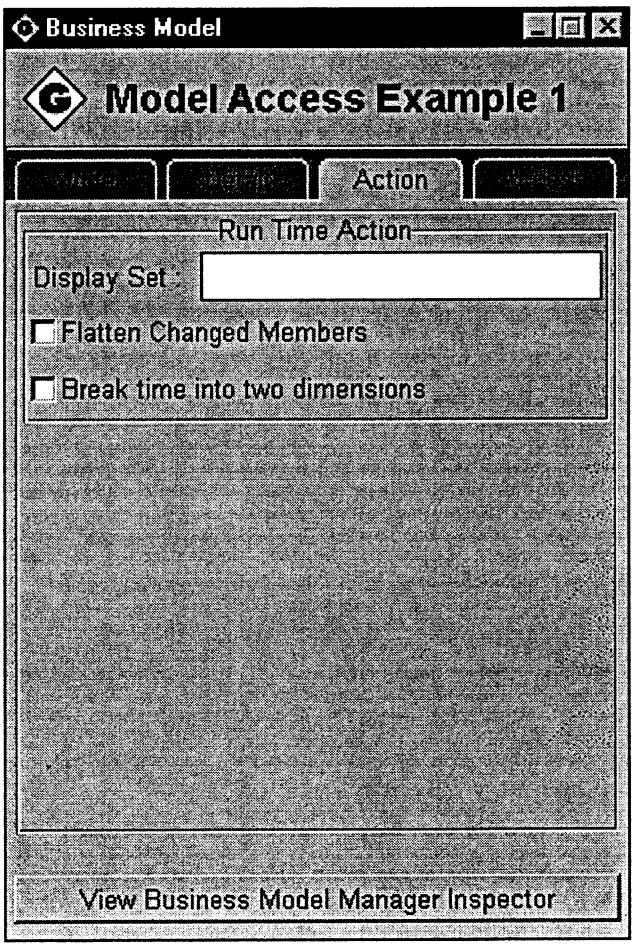
Practical 12.3 - Splitting the Time Dimension

Objective

The aim of this exercise is to split the Time dimension. We will be creating two model specs and setting the options accordingly.

This is an option which only affects the way that the data is viewed. In order to split the time dimension you must create a model spec from the book manager and then check the *Break time into two dimensions* box under the *Action* tab.

- From the book manager, select *Model Spec* from the drop down list.
- Create two new model specs and name them, something like TimeSplit and NotTimeSplit
- For each of the model specs, set the *Business Model Service* name to Profit and the *Model* to your Actual base model.
- On the TimeSplit model spec click on the *Action* tab and select the *Break time into two dimensions*.



A page has been set up with two tables to allow you to examine the effect of splitting the Time dimension.

- Open the **TimeSplit** page in the Profit chapter in the GentiaDB book.

- You will need to use two business source connectors on the connections mapper, one for each model spec, and connect them accordingly.

- Switch into user mode and spend some time viewing the effect of splitting the Time dimension.

13. Join Models

Join Model Example

We now have two models, as follows:

- An actual model with monthly data
- A budget model with quarterly data

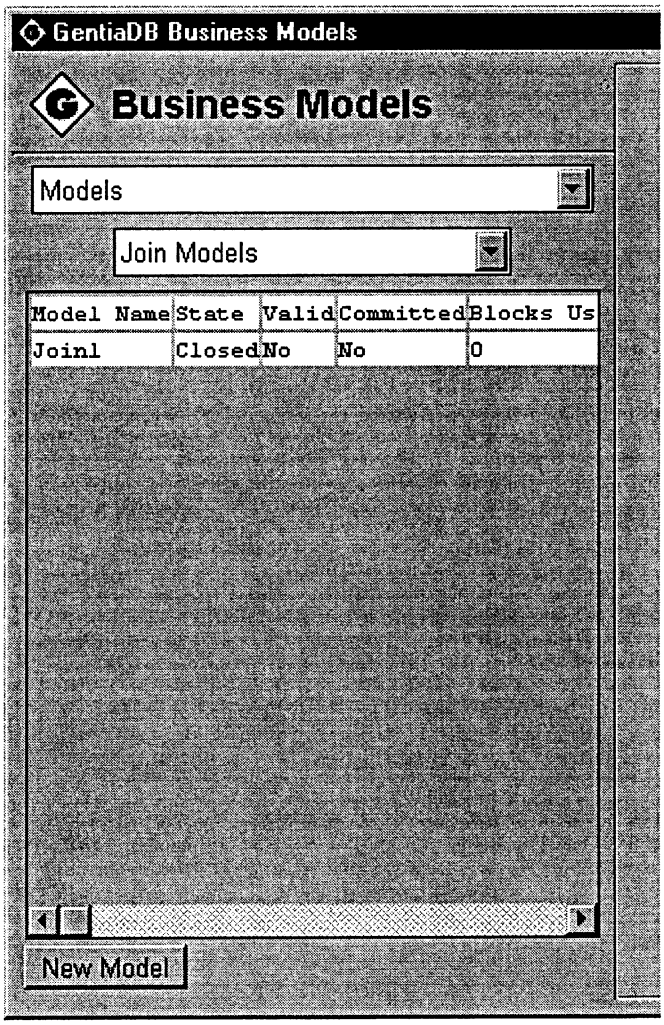
In order to view data on the same chart or table, a **Join Model** can be created. A join model is effectively a virtual view of the data, giving the impression that only one model is being viewed. Data is not loaded into a join model; the data is obtained from the base models. This means that duplication is not an issue, and the size of the database does not therefore increase unnecessarily.

It is not possible to load all the data files into one base model, as Gentia will only allow data to be loaded at the base level. In this instance the actual data is monthly whereas the budget data is quarterly. We must therefore create a join model to view all the data together.

Practical 13.1 - Creating a Join Model

Creating a Join model

Once the data has been loaded into the actual and budget models, they can be joined.



- Select *Edit Models*.
- From the list bar that says *Base Models*, select *Join Models*.
- Click on the button *New Model* and give the model a meaningful name.
- Choose all the items
- Choose span level 1 (quarters), but not any periods.
- Choose all dimensions and all members from each dimension. Don't forget about the computed and aggregated members of the Version dimension.
- Select the *Change Computed Item Source* option and change *From Base* to be YES for all the items.
- Select *Choose Joined Models* to specify from which base models the data is to come.
- Using the Ctrl key, drag the blank line below the base models to be used to populate the join models.
- Check the model.
- Commit the model.

When selecting the calculated members to be included in the join model, you must decide if they are to be taken from the base model or whether they are to be calculated when the join model is accessed. Choose **No** to calculate the value at run-time, i.e. when the join model is accessed. All components making up the calculation must be pre-computed in the base models. Choose **YES** if the value is already in the base model and is to be taken from there.

Once the model has been committed, it can be used as a data source for application pages.

- View the contents of the join model, by setting it as the data source for a page. A page has already been set up, called **Join**, but you will need to create a model spec that points to your join model and introduce a new business source connector.

- Spend some time checking the calculated values on the page.

Practical 13.2 - Optional Join Model Exercise

For discussion purposes

Join models are very powerful, and the use of join models can be exploited to achieve solutions to rather complicated applications. Let's consider the following example:

Item dimension		**Version dimension**
totalcost	(computed)	Actual
subcost1	(computed)	Budget
cost11	(non-computed)	Variance = Budget - Actual
cost12	(non-computed)	
cost13	(non-computed)	
subcost2	(computed)	
cost21	(non-computed)	
cost22	(non-computed)	

The requirement is for a model with the following:

- actual data loaded at the base item level (cost11, cost12, cost13, cost21, cost22)
- budget data loaded at the sub-total level (subcost1, subcost2).

Initially, this might seem reasonably straightforward; you might create two base models, each with the required items. However, the complication is that you cannot load data into computed items.

The object of the exercise is to reach the required solution, by the creative use of join models.

The Solution

- Create 3 sets of items in the item dimension, one set for each of actual, budget and total (sum of actual and budget). The set of items for budget does not need to include the base level members.

atotalcost	(computed)	= asubcost1 + asubcost2
asubcost1	(computed)	= acost11 + acost12 + acost13
acost11		
acost12		
acost13		
asubcost2	(computed)	= acost21 + acost22
acost21		
acost22		
btotalcost	(computed)	= bsubcost1 + bsubcost2
bsubcost1		
bsubcost2		
ttotalcost	(computed)	= atotalcost + btotalcoat
tsubcost1	(computed)	= asubcost1 + bsubcost1
tcost11	(computed)	= acost11
tcost12	(computed)	= acost12
tcost13	(computed)	= acost13
tsubcost2	(computed)	= asubcost2 + bsubcost2
tcost21	(computed)	= acost21
tcost22	(computed)	= acost22

- Create 2 base models each with one member of the Version dimension, for Actual and Budget. Both models should have all the item members, but with data loaded as follows:

Item dimension	Actual base model	Budget base model
atotalcost	computed	-
asubcost1	computed	-
acost11	data	-
acost12	data	-
acost13	data	-
asubcost2	computed	-
acost21	data	-
acost22	data	-
btotalcost	-	computed
bsubcost1	-	data
bsubcost2	-	data
ttotalcost	computed	computed
tsubcost1	computed	computed
tcost11	computed	computed
tcost12	computed	computed
tcost13	computed	computed
tsubcost2	computed	computed
tcost21	computed	computed
tcost22	computed	computed

The 'total' items are required so that both actual and budget base models have corresponding items populated with values; otherwise, a join model would be no use in calculating variance.

All computed items will need to be pre-computed, since the join model will not contain the elements of all the computed items.

- Create a join model, with only the 'total' set of items included. By including budget, actual and variance version members, variance will then be calculated according to the expression defined in the business models (e.g. variance = budget - actual).

- Two further join models can be created, for actual and budget, which are subsets of the base models, but only contain the applicable set of items. The reason for this is that you then have smaller, more manageable models to work with on a page, that only contain the items that are required.

14. Some Model Considerations and Jobs

In this section we will be looking at two topics. The effect of changes to business and base models and jobs, which allow us to run a number of tasks and actions in one go.

The best way to understand the effects of changes to business and base models is to look at examples of what changes are often introduced and to see the effects. A practical then follows using our model examples.

Description of change	Where change is made	Effects of change	Notes
Add or delete a new category dimension member	Business model	Filters into base model	Provided the dimension is non-keyed.
Move member within same dimension i.e. one branch of product to another	Business model	Filters into base model	Need to decide whether to restate or retain
Add a completely new item	Business models	Filters into base model	
Add new Item	Base model	Recalculates if necessary	New feature in 3.1
Delete an Item	Base Model	Recalculates if necessary	
Decommission base	Base model	All data is lost	
Advance period	Business model	All base, joins and dynamic spans	Cannot reverse
Change time dimension	Not possible		Can add cycles, display strings, display sets and dynamic spans

To see this in more detail, in the next practical we will move

NY New York

from the distributor branch into the subsidiary branch of the location dimension. According to the table above, moving a dimension member within the dimension does not need a data reload. The business changes filter down into the base model. Other software products require a reload of data after structural changes. Gentia does not need to do this, so much time can be saved.

Retain and Restate

If information is changed due to the restructuring of a dimension, then a decision needs to be made as to whether the changes should be shown or whether they should be removed. The restate history and retain history options allow us to do just that. There are many different options to consider when deciding between restating/retaining history. Consider one example

below, where Product Z has moved from one Product Group to another Product Group within the same model.

	Jan	Feb	Mar	Apr	May	Jun
Product Group 1						
Prod X	10	54	84	64	52	85
Prod Y	35	38	62	44	75	34
Prod Z	12	53	-	-	-	-
	57	145	146	108	127	119
Product Group 2						
Prod A	42	65	45	35	55	43
Prod B	34	36	30	82	34	77
Prod Z			64	36	65	43
	76	101	139	153	154	163

Product Z moved Groups at the start of March. The information has been retained, in that although for March onwards (in Product Group 1) there are no figures for Product Z, the product is still visible. Consider the alternative, which is to restate the information.

	Jan	Feb	Mar	Apr	May	Jun
Product Group 1						
Prod X	10	54	84	64	52	85
Prod Y	35	38	62	44	75	34
	45	92	146	108	127	119
Product Group 2						
Prod A	42	65	45	35	55	43
Prod B	34	36	30	82	34	77
Prod Z	12	53	64	36	65	43
	88	154	139	153	154	163

Now the information for Product Z in Group 1, has been removed. As no information was to be added it seemed appropriate to remove the product. The example now shows no trace of Product Z within Group 1, giving the appearance that Product Z has always been in Group 2.

The above examples show only one of the effects of retaining/restating history. In addition, if the retain history option is chosen, it is possible to use the **Flatten Changed Members** option as well. This enables us to flatten the Product Z line in Group 1 because it is no longer necessary to view it.

	Jan	Feb	Mar	Apr	May	Jun
Product Group 1						
Prod X	10	54	84	64	52	85
Prod Y	35	38	62	44	75	34
	57	**145**	146	108	127	119
Product Group 2						
Prod A	42	65	45	35	55	43
Prod B	34	36	30	82	34	77
Prod Z			64	36	65	43
	76	101	139	153	154	163

The product has disappeared from Branch 1, however, the totals (in bold) do not tally for the visible products showing that something has been removed. The advantage is that product z does not show lots of empty cells containing no information.

Jobs

Instead of going through all the various stages of loading dimensions and data into a model in order to view new data, for this next practical we will be running a job to change the information.

As mentioned earlier, a job is a list of commands and tasks, which perform repetitive actions. Jobs can be set to run one at a time or run automatically at specified times, by defining an agent. As this is a one-off job, we will be running it manually.

There will be 4 stages to the job as shown below:

1. Open a new version of the business model.
2. Run a task to reload the dimension. (This task already exists)
3. Load the location dimension.
4. Commit the new version.

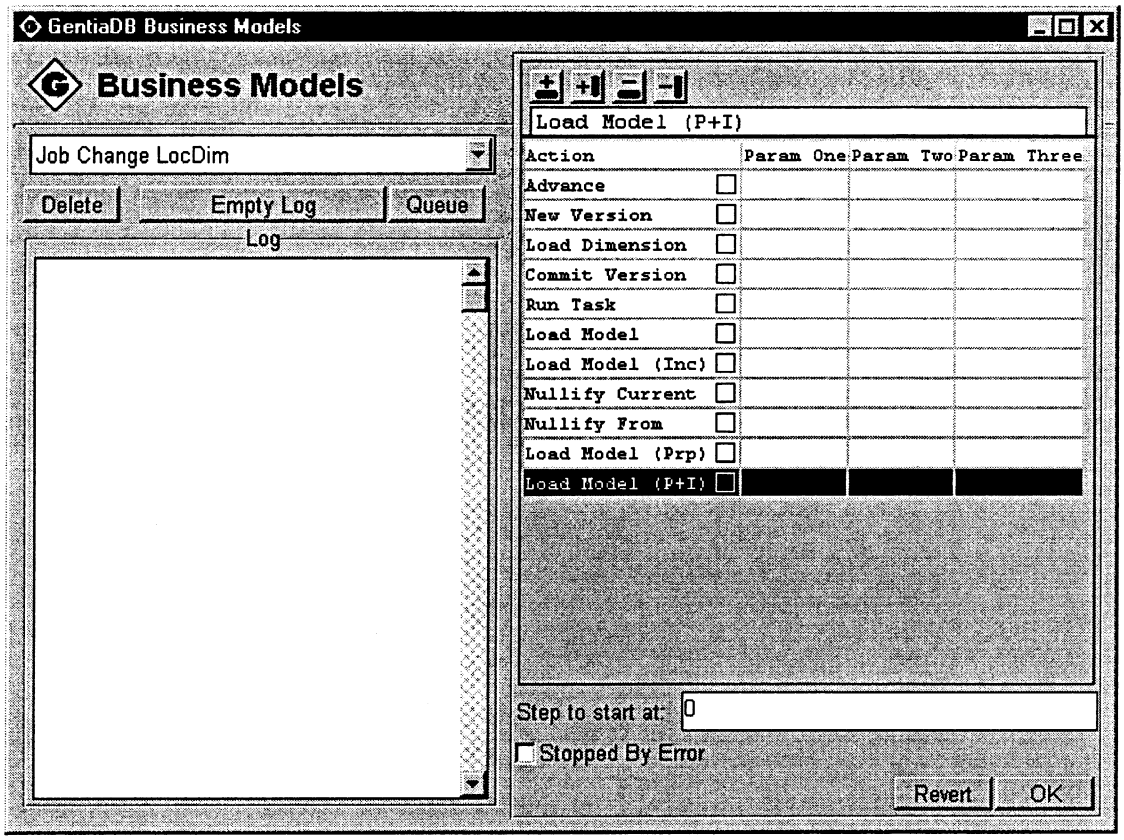
The job can consist of a variety of actions depending on what is required. The actions available can be any of the following:

a) Advance Period
b) New Versions
c) Load Dimension
c) Commit Version
d) Run Task
e) Load Model
f) Load Model Incrementally
g) Nullify Current
h) Nullify From

For the various options, certain parameters must be set as follows:

Action	Param One	Param Two	Param Three
Run Task	*task name*		
Load Dimension	*dimension name*	*task name*	*data holder name*
Load Model	*base model name*	*data holder name*	*task name*

All the other options have no parameter settings.



Once the tasks have been created and checked and the actions have been set up correctly for the job, the job itself must be sent to the queue from where it is run. The queue will contain all the jobs that have been submitted. All the actions in the job must complete successfully before the job is removed from the queue.

The queue button in the job dialogue window allows you to submit a job to the queue. The queue itself can be started and stopped from the drop-down options in the business model pick list. The log panel displays the status of jobs and allows you to track errors. The Empty Log button allows you to clear this log down. The delete button allows you to delete a job.

If the job falls over at any point, the number of the task or action where it fell is logged in the log panel. To restart the job (having first fixed the fault) you must:

- set the *Step to start at* number, to the number of the action which fell over, bearing in mind that the list starts at number 0.
- Uncheck the *Stopped By Error* box.
- start the job queue again

Practical 14.1 - Using a Job

The task to load the location dimension already exists, however, the file which it uses must be changed. The NY member must be moved to the subsidiary branch. Gentia recognises existing members and maintains the data associated with them. It should be noted that dimensions must be loaded in their entirety, not just one or two members at a time.

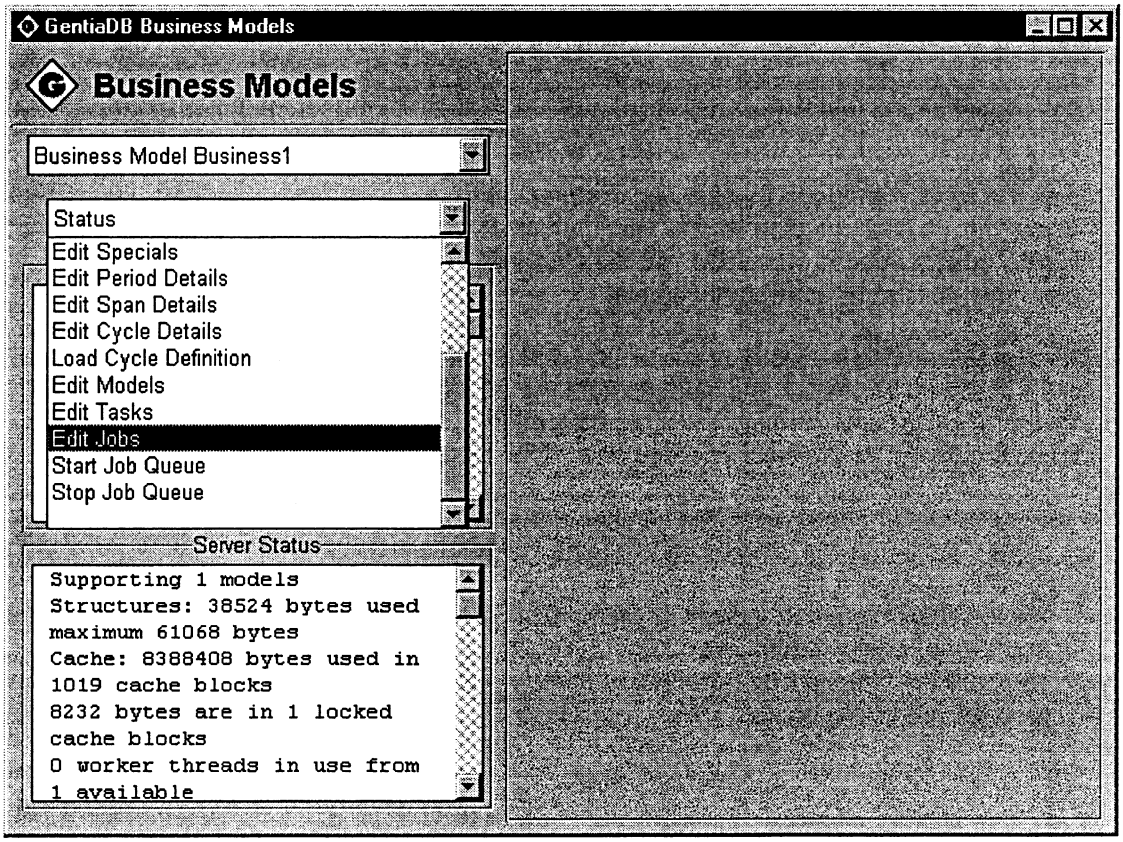
- Open the file c:\gentia\gentiadb\data\LocDim.txt
- Change **NY Dis** to read **NY Sub**

Once the input file is changed, the task can be run from within the job queue and the new version of the location dimension will be loaded.

Creating/editing Job Details

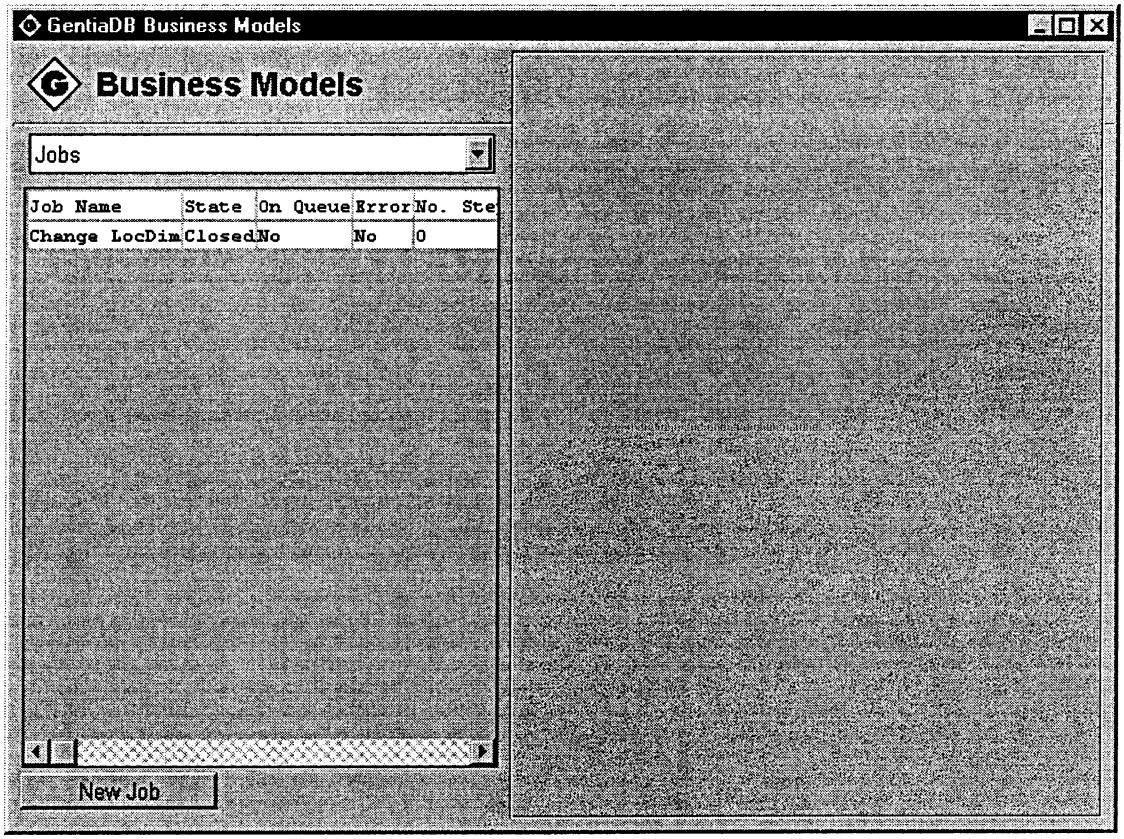
Once the above file is set up, we can create or edit our job by listing its actions.

- Select the *Edit Jobs* option from the business model list bar.

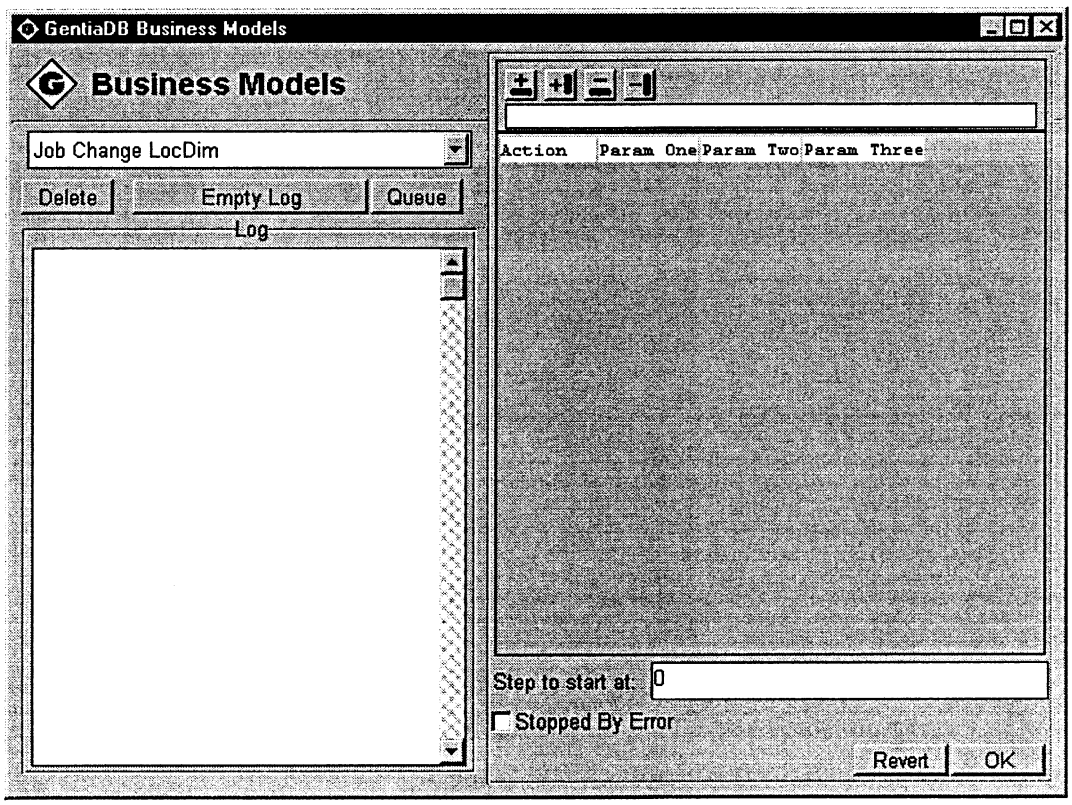


To add a new job, click on the *New Job* button.

- Give the job a name.

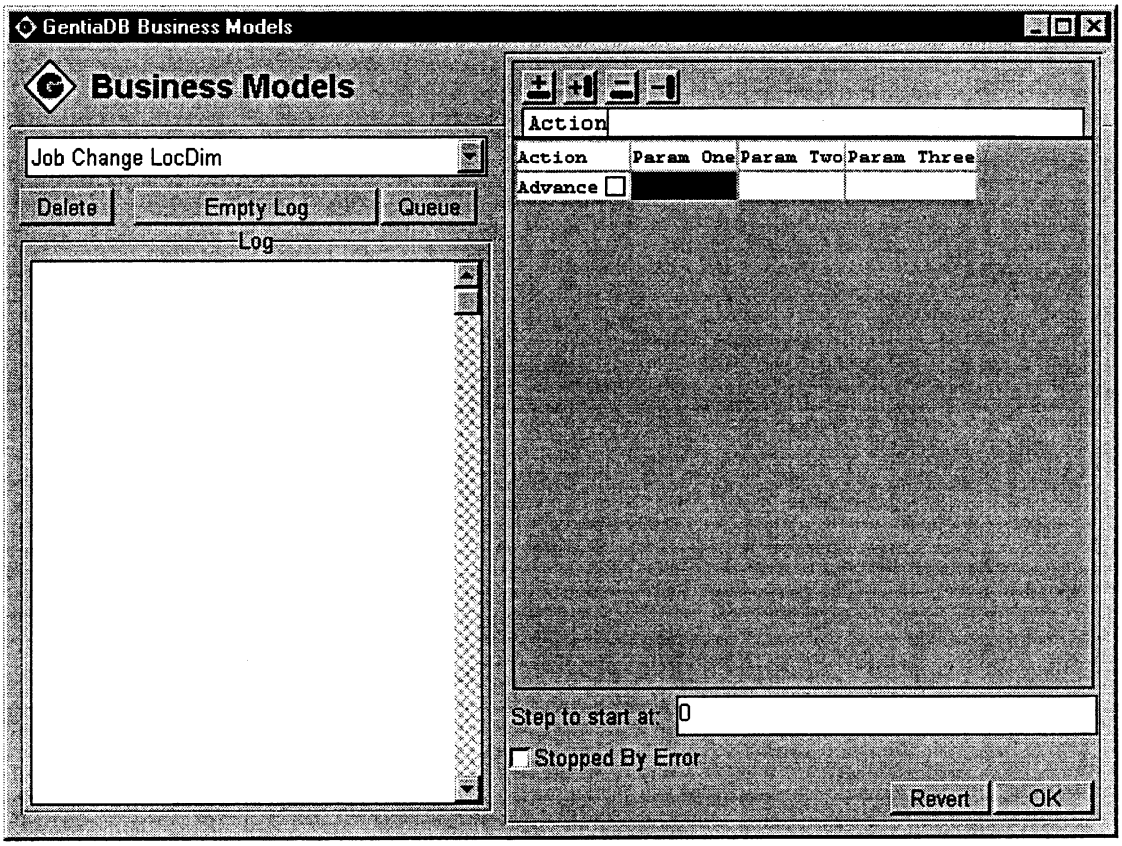


Alternatively, select an existing job by clicking on its job name on the left hand side.



- Adding tasks to the job is done by adding lines to the right hand side of the dialogue box. Click on the button to add a new row. 

This adds a row with an *action field*. By clicking on the box to the right, it is possible to toggle through the options so that all the steps of the job can be chosen.

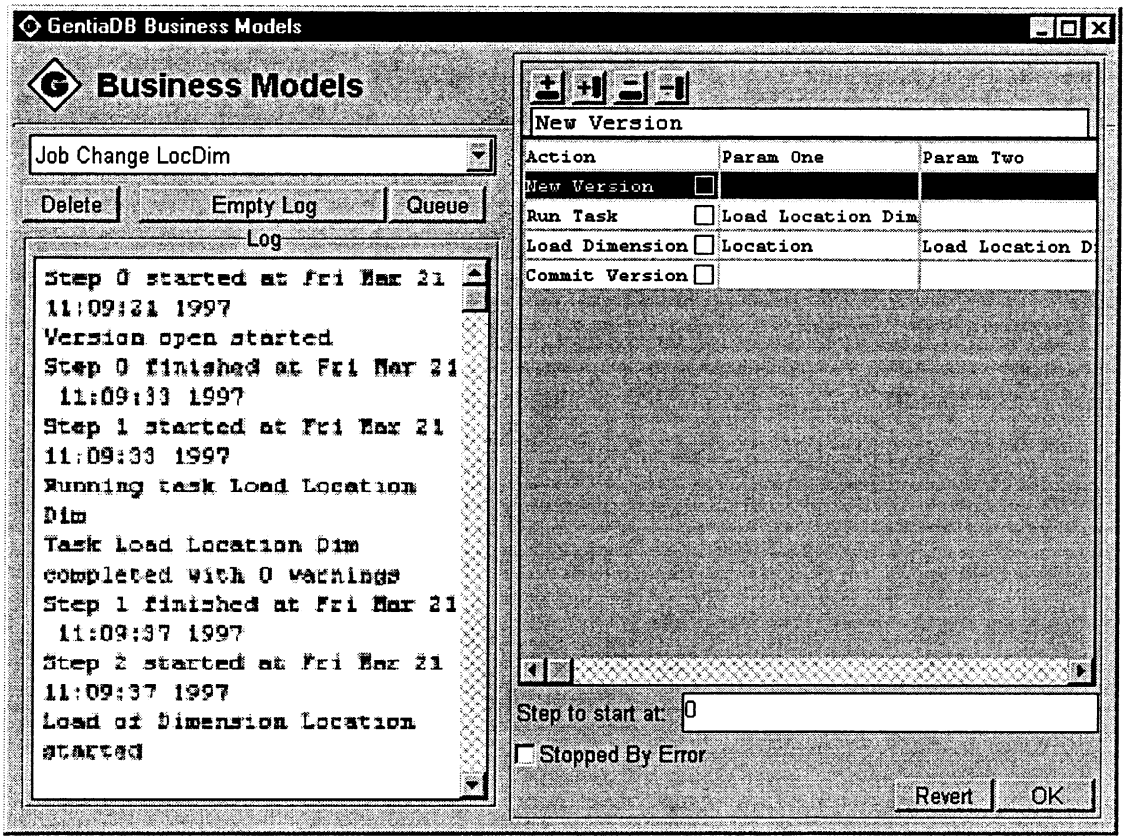


The 4 steps which are needed are as follows:

1. Open a new version of the business model.
2. Run the task to load the holder with the new structure.
3. Load the location dimension.
4. Commit the new version.

- Having selected the option you must then set the parameters accordingly.
- Once the job has been set up, place it on the queue.
- From the drop down list, select *Start Job Queue.*

When the job has completed successfully and there are no errors reported in the log, you can go and view the data to see the new location member and its appropriate data.



- View the effects of changing the NY member.

  In addition, ensure that the retain history option is on and experiment with the *Flatten Changed Members*. This option is located from the same place that we accessed the splitting of the time dimension.

In model spec
- Display Set : Taal Leuze / Werks omschrijving
  - breakdown into two dimensions
    span/cycles
    periods
  - Flatten changed members
    Bewaar de totalen ook als een member verwijderd is.
    vb. Filiaal x is verhuisd.
    De som van de regio klopt nog en Filiaal x is niet meer te zien.
    Dit is ongewijzigd voorde getrouwheid omdat de som op het scherm niet meer klopt met de data uit de database.

15. Efficiency Issues

Database efficiency can be affected by the design of the GentiaDB database. Therefore it is important to understand how good design can help to optimise your application.

We will be looking at how you can optimise your application in terms of:

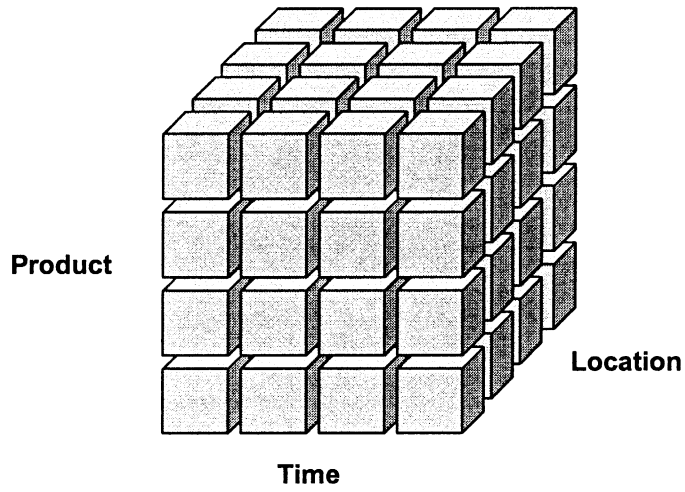
- controlling the sparsity
- optimising proportion of database containing data values (as opposed to overhead)
- calculation time
- in-flight calculations versus pre-computed values
- one large base model versus several smaller base models

Before we consider each issue we need to understand the concept of :

- multi-dimensionality
- **keyed** and **non-keyed** dimensions
- how data is stored in **records** within GentiaDB

Multi-dimensional Data Storage

A GentiaDB base model has a number of dimensions; at least three, but typically more. Space could potentially be reserved for every possible data combination. For instance, let's say that you had a three-dimensional model, as follows:



There is potentially room for a piece of data for every dimension combination, i.e. for every product sold in each of the locations, every month. However, there may be some products which are never sold in particular locations, or some products which do not sell at certain times of the year. If this is the case, there is no point in reserving space for data which will never exist, since each potential piece of data takes up **8 bytes** of disk space. This space, in which one value is stored, is known as a cell.

Reserving space for invalid data combinations will cause sparsity, which in turn will cause the database to grow in size unnecessarily.

GentiaDB handles this issue of sparsity, by allowing you to define whether a category dimension is keyed or non keyed.

Keyed and Non-keyed Dimensions

In GentiaDB base models, dimensions are defined as being either keyed or non-keyed, as follows:

Item dimension is always non-keyed
Time dimension is always keyed
Category dimensions can be either keyed or non-keyed; by default they are keyed.

A non-keyed dimension will always have space reserved for its data, whereas a keyed dimension will not. This means that if you have a keyed dimension that is sparsely populated with data, i.e. cells for which you do not have any data, GentiaDB will not waste disk space storing these empty cells. Since each cell occupies 8 bytes, this could be a useful saving.

However, if data was to be loaded into a keyed dimension, then space would immediately be found for it. The combination of keyed and non-keyed dimensions will determine the structure of the records.

Records

Within GentiaDB base models, data is held in records. The data is accessed by using a sequential search algorithm which provides extremely fast data retrieval. The record is defined by the keyed and non keyed dimensions. For the purpose of optimisation, we will need to consider the following:

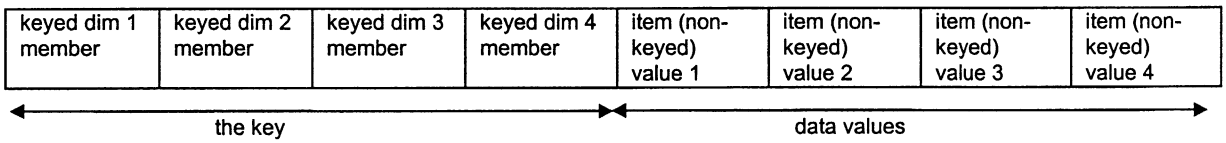
- structure of the records
- number of records in a base model
- size of each record
- use of space on disk

Structure of Data Record

As well as containing data values, each record also has a key, which acts as a label for the data values, to determine to which cells the data belongs. The key is created from the keyed dimensions.

The structure of a record in a base model with 5 dimensions, where the Item dimension has 4 members, and is the only non-keyed dimension, would be as follows:

A Record



Number of Records

Each record has a key formed by the combination of one member from each of the keyed dimensions. Consequently, the number of members in the keyed dimensions will determine the number of potential records in the base model.

Remember that a record will not exist for a combination of keyed dimension members for which there is no data.

Record Size

The size of the record can be an important consideration; the calculation is as follows:

record size = size of key + size of data values

size of key = no. keyed dimensions * 4 bytes (rounded up to a multiple of 8 bytes)

size of data values = no. data values * 8 bytes

Other Size Issues

It seems appropriate at this stage, to not only consider the amount of disk space used for data, but the other areas of GentiaDB that contribute to the size of the database. These are:

- The holder and change records
- The structural elements of the database, i.e. data dictionary, base/join model definitions, tasks, etc.

When a task has run, the holder becomes populated with data, which takes up disk space. The size of the holder is dependent upon the number of records; the size of each record is calculated on the following basis:

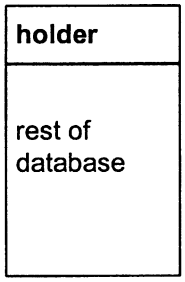
8 bytes for each numerical value
4 bytes for each member name, including members of the time dimension

When you delete a holder, the size of the database does not reduce, but the space taken up by the holder is allocated as reusable. If a large holder is replaced by a smaller one when running a particular task, the size of the database will not decrease. The space will be re-used by other elements in GentiaDB, but you cannot reclaim the space as such to reduce the size of the database.

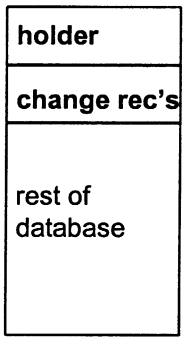
The same principle applies to change records. When a model is being loaded, change records are generated. It is only when the data load is complete with no errors that the change records are committed to the model. This is part of the commit/roll back technology, which ensures that any errors with data loading will not affect what was previously loaded. A large data load will produce large change records. If the next set of change records is smaller, the size of the database is not reduced; instead, the space would be allocated as being re-useable.

The following example demonstrates this:

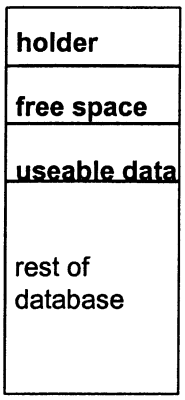
1. Run a task which populates a holder



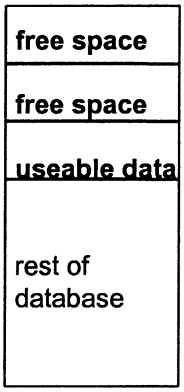
2. Load a model



3. The data is committed



4. Delete the holder



It is therefore recommended to avoid loading very large data files at one time, in order to prevent unnecessarily large holders and change records increasing the size of the database too much. For instance, it will be more efficient to separately load 12 data files for each of the periods, than one large data file containing data for all 12 periods.

The structural elements within GentiaDB will add to the size of the database. Obviously, the more members you have, the larger the database will be. But compared to the size of the data, the size of the structural elements is relatively small, and becomes increasingly more negligible as the whole database gets bigger.

In summary, the size of GentiaDB will **never** reduce. But, by understanding the principles, you can help ensure that your GentiaDB application is not unnecessarily large.

Example 1

Let's consider an example of a default base model with the following dimensions:

Item dimension	6 members	non-keyed
Time dimension	12 members	keyed
Location dimension	10 members	keyed
Product dimension	20 members	keyed
Version dimension	4 members	keyed

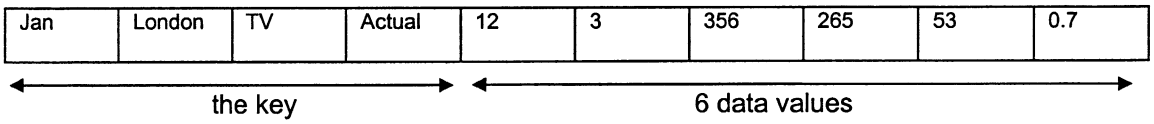
Since base models hold data multi-dimensionally, there could potentially be a data value for every combination of each of the dimensions, as follows:

Number of potential data cells = 6 * 12 * 10 * 20 * 4
= 57,600

Since the item dimension is the only non-keyed dimension, each record will contain data for the 6 item members,

i.e. number of values in each record = 6

A record would be structured as follows:



Each record will have a key made up of one member from each of the keyed dimensions.

i.e. the number of potential records = 12 * 10 * 20 * 4
= 9,600

Size of key = 4 * 4 bytes (rounded up to multiple of 8 bytes) = 16 bytes

Size of data values = 6 * 8 bytes = 48 bytes

Therefore, size of record = 64 bytes

From this, we can calculate the following:

No. records that would fit into one block = 8192 / 64 (rounded down to whole number)
= 128 records

No. blocks required = 9600 / 128 (rounded up to a whole number)
= 75 blocks

** Amount of disk space utilised by the data = 75 * 8192 bytes
= 0.614 Mb

**** Important note:**
Only the **potential** amount of disk space used for data can be calculated, since a fully populated database is assumed. A sparse database will obviously take up less space, since some records will not exist. Therefore, in practise, this can only be estimated, and would be determined by the degree of sparsity.

Example 2

If we change one of the category dimensions to be non-keyed, as follows:

Item dimension	6 members	non-keyed
Time dimension	12 members	keyed
Location dimension	10 members	keyed
Product dimension	20 members	keyed
Version dimension	**4 members**	**[redacted]keyed**

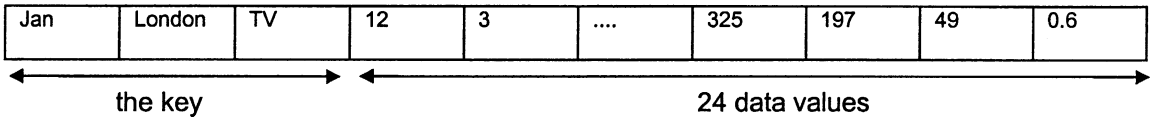
non

Be start niet wel als non-keyed.

Each record will contain data for the **item and version** members,

i.e. number of values in the record = 6 * 4
= 24

A record would be structured as follows:



And, each record will have a key made up of one member from each of the keyed dimensions.

i.e. the number of potential records = 12 * 10 * 20
= 2,400

Size of key = 3 * 4 bytes (rounded up to multiple of 8 bytes) = 16 bytes

Size of data values = 24 * 8 bytes = 192 bytes

Therefore, size of record = 208 bytes

From this, we can calculate the following:

No. records that would fit into one block = 8192 / 208 (rounded down to whole number)
= 39 records

No. blocks required = 2400 / 39 (rounded up to a whole number)
= 62 blocks

** Amount of disk space utilised by the data = 62 * 8192 bytes
= 0.508 Mb

**** Important note:**
Only the **potential** amount of disk space used for data can be calculated, since a fully populated database is assumed. A sparse database will obviously take up less space, since some records will not exist. Therefore, in practise, this can only be estimated, and would be determined by the degree of sparsity.

GentiaDB Optimisation

The following issues highlight how the design of the database can affect the efficiency of the application. They do not work in isolation, and therefore it is impossible to optimise an application in every respect (size, calculation time, and performance), but it should be possible, through testing, to find a balance to suit your specific application requirements.

For instance, your database might take 4 hours to load. (Do bear in mind that the load process also includes the calculation and consolidation processes.) You may be able to perform some fine tuning in order to reduce the loading time to 2.5 hours, but by doing so, on-line performance may not be as good. If the consolidation process runs over night or once a month, then it may not matter that it takes 4 hours, but it **will** matter to your users if the Gentia pages take a while to open, whilst many in-flight calculations are taking place.

We will be looking at how you can optimise your GentiaDB application in terms of the following:

- sparsity
- proportion of the database containing data values (as opposed to overhead)
- calculation time
- in-flight calculations versus pre-computed values
- one large base model versus several smaller base models

Minimising Sparsity

For keyed dimensions, a record will only exist if there is any data to be stored in it. For instance, you may never sell a certain product in a certain location, in which case the record for that key will not exist.

This means that you should only make densely populated category dimensions non-keyed, otherwise your base model will suffer from sparsity, since each record would have space reserved for data values which do not exist.

Important Note:

- Making a category dimension non-keyed can have serious implications on the way in which your application would work.

- If you have a category dimension with a frequently changing structure, e.g. Product, any change to the structure in the business model, will only ripple through to the base model if Product is keyed. Non-keyed category dimensions cannot be restructured in base models.

- So, if a category dimension is likely to vary, e.g. product groups changing, then it would make sense to keep the product dimension keyed.

- In the case of a changing non-keyed category dimension, the particular base model would need to be rebuilt.

Use of Disk Space

It is important to consider the amount of disk space used by numbers as opposed to overhead, i.e. the key and any empty parts of the 8K block.

By calculating the size of the record, and the potential number of records, you should be able to work out how many 8K blocks of data will be required. However, GentiaDB will provide you with a figure indicating the number of blocks used, and the percentage efficiency (space taken up by data values in relation to the space taken up by the entire 8K block). If this percentage is low, you may want to consider making a category dimension non-keyed, and then reloading the data to see if a better percentage can be achieved.

Another consideration is adding new non-keyed members. Adding a new member will increase the size of the slice. Even a small increase in slice size could drastically change the efficiency percentage.

Rule of Thumb for Small Records

For base models with very small records, the amount of empty space in the 8K blocks is negligible, and the efficiency can be considered more simply by calculating the ratio of non-keyed members to keyed members, as follows:

efficiency ratio = (n * 2) as a percentage of [(n * 2) + m]

where: n = number of data values in the non-keyed dimensions
m = number of dimensions

In **example 1**, the ratio = (6 * 2) as a percentage of (6 * 2 + 5)
= 12 % (12 + 5)
= 12 % 17
= 70.6%

In **example 2**, the ratio = (6 * 4 * 2) as a percentage of (6 * 4 * 2 + 5)
= 48 % (48 + 5)
= 48 % 53
= 90.5%

Since the ratio is representing the proportion of disk space used for data as opposed to the key, the larger the ratio, the better. Bear in mind, that this rule of thumb is only applicable for small records.

Calculation Time

Calculations in a non-keyed dimension are faster than in a keyed dimension. This is because Gentia will calculate records of data at a time. So, to minimise calculation time, category dimensions that contain calculations, other than consolidation, should be **non-keyed**.

e.g. **Variance = Actual - Budget,** where Variance is a computed member in the Version dimension

Summary: Guidelines for Keyed and Non-Keyed Dimensions

- Any category dimensions which have regularly changing structures should remain keyed.
- Category dimensions with calculations that are static should be non-keyed to minimise calculation time.
- Any sparsely populated dimensions should remain keyed.

Size and Performance: In-flight Calculations Vs. Pre-Computed Calculations

Calculating values on an in-flight basis, rather than pre-computing and therefore storing values, will save on disk space.

However, in-flight calculations could affect on-line performance, since the data will be calculated as it is required on a page, which could affect the time it takes for the page to open.

Pre-computed items will also increase the calculation time.

Size and Performance: One Base Model Vs. Many Base Models

The effect of having several smaller, normalised base models rather than one larger base model, is that vast savings on storage can be made, particularly if your database is sparse.

Join models can be created in order to analyse data spanning across more than one base model. Join models are not physical models, in that they do not hold any data; they are merely providing a virtual view of the data, as if it was held in one model. It is therefore possible to create a suite of 'normalised' base models which can then be joined as appropriate. This 'relational' approach to storing the data results in very effective use of disk space.

Performance may be affected by the time it takes for the join model to form, particularly if there are many cross-model calculations, e.g. Two base models, actual and budget, and a join model with a variance calculation.


USING JOBS

Jobs allow you to define a batch series of commands. You use jobs where you frequently want to perform a set sequence of actions. You can then simply initiate the job rather that having to select each element of the job individually.

The jobs you can use and their purpose are as follows:

- Advance Period - move the current period forward by one period.
- New Version - open a new version of the business model to allow updates.
- Load Dimension - load a new dimension structure.
- Commit Version - close and commit the version of the business model.
- Run Task - run a named task.
- Load Model - load a named base model with data from a specified holder.
- Load Model Incrementally - carry out an incremental data load on a named base model with data from a specified holder.

- Load Model Propagating – make the load carry forward values from a previous period if there is no data for the current period when calculating computed items.
- Load Model Incremental and Propagating - to combine an incremental load with a propagating load.
- Nullify Current - remove all values for the current period.
- Nullify From - remove from the named base model all data currently held in a specified holder.

➢ *You cannot use a job to run a task that modifies the item and time dimensions - for example, to load further cycles or change display strings.*

A typical sequence of actions for a job might be:

1 Open new version of business model

2 Run a task to load in dimension member names

3 Load the dimension

4 Commit the new version

5 Run a task to bring in data


6 Load data into a base model.

7 Advance Period

➢ *Each action within a job is a separate transaction. Completed actions will not roll-back if the job fails to complete.*

You can also initiate jobs using Gentia's agency system. This allows you to define Odd Job agents to run jobs automatically at set times.

Editing job details

Starting the job queue

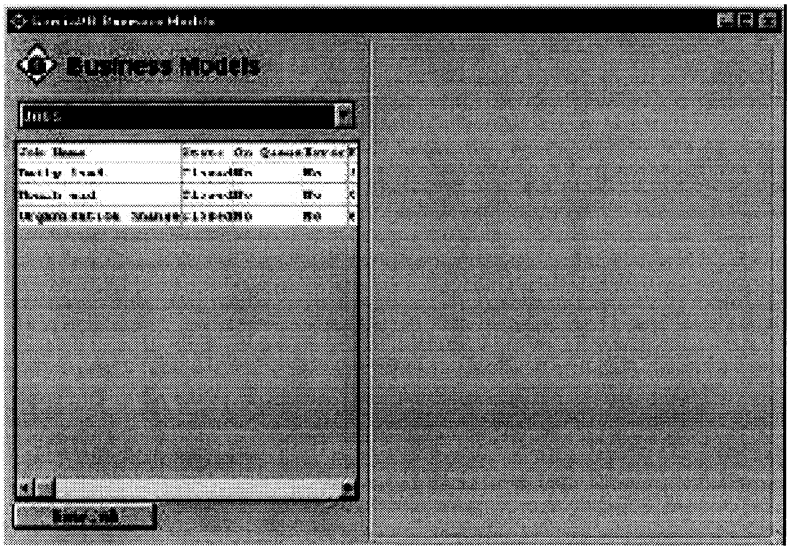
Stopping the job queue

Editing job details

To edit job details:

1 From the business model list bar select Edit Jobs.

2 To add a new job, click on the Add Job button and then enter the name of the job in the dialog box displayed. To select an existing job, click on the job name.



3 To add tasks to the job, add lines to the right hand side. Click on the Action box to toggle to the task type you wish to select.

4 Specify the appropriate parameters:

Run Task - the task name is the first parameter.

Load Dimension - the dimension name is the first parameter, the task name is the second, and the data holder name is the third.

Load Model - the base model name is the first parameter, the task name is the second, and the data holder name is the third.

Nullify Current - the base model name is the first parameter.

Nullify From - the base model name is the first parameter, the task name is the second, and the data holder name is the third.

5 To delete the selected job, click on the Delete button

6 The Log panel displays the log status of jobs as they are processed. This allows you to track any errors that may arise. To clear this log, click on the Empty Log button.

7 To send the job to the Job Queue, click on the Queue button.

➢ *If you will be using an odd job agent to run the job do not send it to the queue.*


8 If the job has been stopped by an error, the Stopped by error check box will be checked. You will need to identify and correct the error and then resubmit the job. When you resubmit, you must clear this check box and specify the step from which you wish to restart in the Step to start at box. (Note that the first step is identified as step 0.) For example, if the job initially advanced the period but failed on the data load, you would not wish to advance the period again when you resubmit the job.


Starting the job queue

Once started, the job queue will stay open and execute jobs until it is closed. To start the job queue:

1 From the business model options drop down list bar select Start Job Queue.

➢ *If you will be using an agent to run the job you need to ensure the queue has been started.*


Stopping the job queue

To stop the job queue and prevent jobs being processed:

1 From the business model options drop down list bar select Stop Job Queue.

- *Any jobs left in the queue when you stop it will remain in the queue.*
- *If the files of a GentiaDB go through recovery for any reason all job queues within the business model are stopped.*

```
routine startRun is
destroy $qry;
let $qry[1]= "sqlData1";
let $qry[2]= "sqlData2";

        let $file BE TEXT "ConnectionInfo";
        let $out be new file ".\\ppp.txt";
        $dataSource = $file nextLine;
        print on $out "Datasource is: "+$dataSource,nl;
        $service    = $file nextLine;
        print on $out "Service is: "+$service,nl;

        $file closeFile = YES;

        &msg = "OK";

        &sql = !_sqlServer newConnection;
        #&sql indefiniteTimeOut = $TimeOut;
        &sql dataSource = $dataSource;
        &sql namedService = $service;
        &sql connect = YES;

        if &sql failed then begin
                &msg = "Connection failed...."
        end
        else begin

                for each integer &t in $qry do begin
                        &sql storedSql = $qry[&t];
                        &sql executeSelect = YES;

                        if &sql failed then begin
                        &msg = "Select  failed....."
                        end
                        else begin

                                for each integer &i in &sql results rows do begin
                                        &results[1]      = STRIP(&sql results rows[&i] _Field_1);
                                        &results[2]              = STRIP(&sql results rows[&i] _Field_2);
                                   &results[3]           = STRIP(&sql results rows[&i] _Field_3);

                                        if $qry[&t] = "sqlData2" then
                                   &results[3]           = "P"+&results[3]
                                   fi;
                                        &date                    = STRIP(&sql results rows[&i] _Field_4);
                                   &num                          = split(&date,"-",$time);
                                        &results[4]              = "M" + $time[2];
                                   &results[5]           = "Y" + $time[1];
                                        &results[6]              = REAL(&sql results rows[&i] _Field_5);
                                       &results[7]               = REAL(&sql results rows[&i] _Field_6);
                                        &results[8]              = REAL(&sql results rows[&i] _Field_7);
                                        &results[9]              = REAL(&sql results rows[&i] _Field_8);
```

```
                                                print on $out "RETURN
"+&results[1]+";"+&results[2]+";"+&results[3]+";"+&results[4]+";"

+&results[5]+";"+&results[6]+";"+&results[7]+";"+&results[8]+";"+&results[9],nl;
                                CALL !_BM_outputStream(&stream=0, &data be &results)

                                end
                        end fi
                end
        end fi;
        print on $out &msg;
        let $out closeFile = YES;
        &sql connect = NO;

RETURN 0
end;
##################################################
ROUTINE eofReceived IS
##################################################
CALL !_BM_eofStream(&stream=0);
RETURN 0
end
END
```