MANUAL DEL PROGRAMADOR
LENGUAJE ALGEBRAICO EN BASES DE DATOS

Octubre 2018

ACUERDO DE CONFIDENCIALIDAD

AL LEER ESTE DOCUMENTO EL LECTOR RECONOCE QUE EL MISMO CONTIENE INFORMACIÓN CONFIDENCIAL PROPIEDAD INTELECTUAL DE DO ANALYTICS LLC Y ACEPTA QUE LO MANTENDRÁ EN FORMA CONFIDENCIAL, GUARDÁNDOLO CONTRA INSPECCIÓN DE TERCERAS PERSONAS Y DE ORGANIZACIONES NO AUTORIZADAS EXPLÍCITAMENTE POR DO ANALYTICS.

EL LECTOR RECONOCE QUE LA METODOLOGÍA DE DESARROLLAR MODELOS DE PROGRAMACIÓN MATEMÁTICA BASADOS EN LA CONFIGURACIÓN DE UN SISTEMA DE INFORMACIÓN Y SU POSTERIOR PROCESAMIENTO ES PROPIA Y ORIGINAL DEL PRODUCTO OPTEX OPTIMIZATION EXPERT SYSTEM (OPTEX), QUE LA MISMA FUE DESARROLLADA INICIALMENTE POR DECISIONWARE LTDA. Y QUE ACTUALMENTE ES PROPIEDAD DE DO ANALYTICS LLC.

EL LECTOR ACEPTA QUE ÉL SABE QUE LEER Y/O ESTUDIAR (O FACILITAR QUE ALGUIEN LEA O ESTUDIE) ESTE DOCUMENTO CON LA INTENCIÓN DE COPIAR / CAMBIAR / MEJORAR / SIMPLIFICAR / DESINTEGRAR / INTEGRAR / ESPIAR (O CUALQUIER OTRA ACTIVIDAD SIMILAR)

i) LA METODOLOGÍA IMPLÍCITA EN OPTEX,
ii) LOS SISTEMAS DE INFORMACIÓN DE OPTEX,
iii) LOS PROGRAMAS DE COMPUTADOR GENERADOS POR OPTEX, Y/O
iv) LAS INTERFACES DE ACCESO ASOCIADAS A LOS PROGRAMAS QUE INTEGRAN OPTEX

CORRESPONDE A UNA VIOLACIÓN DE LOS DERECHOS DE AUTOR Y DE LA PROPIEDAD INTELECTUAL DE DO ANALYTICS Y ENTIENDE QUE DOA PODRÁ TOMAR LAS ACCIONES LEGALES PERTINENTES PARA PROTEGER SUS DERECHOS.

LA INFORMACIÓN CONTENIDA EN ESTE DOCUMENTO NO PODRÁ SER REVELADA A TERCEROS Y NO DEBERÁ SER COPIADA DIGITALMENTE NI FOTOCOPIADA, NI USADA NI REVELADA, EN SU TOTALIDAD O PARCIALMENTE, PARA NINGÚN OTRO PROPÓSITO DISTINTO AL USO INTERNO.

ESTA RESTRICCIÓN NO LIMITA EL DERECHO DEL LECTOR PARA UTILIZAR LA INFORMACIÓN CONTENIDA EN ESTE INFORME, QUE SEA DE DOMINIO PÚBLICO O SI ES OBTENIDA DE OTRAS FUENTES SIN RESTRICCIONES.

TODA LA INFORMACIÓN DEL TERCERO A LA QUE DO ANALYTICS TENGA ACCESO COMO RESULTADO DE ESTE PROCESO DE DIFUSIÓN DE LOS SERVICIOS Y DE LOS PRODUCTOS QUE OFRECE DO ANALYTICS SERÁ MANTENIDA EN FORMA ESTRICTAMENTE CONFIDENCIAL POR DO ANALYTICS Y POR LOS PROFESIONALES DE DO ANALYTICS QUE SE VINCULEN AL PROCESO.

LA FORMULACIÓN ALGEBRAICA PRESENTADA EN ESTE DOCUMENTO Y EN EL SOFTWARE QUE CONTIENE LA IMPLEMENTACIÓN DE LOS MODELOS MATEMÁTICOS EN OPTEX SOLO PUEDE SER UTILIZADA CON PROPÓSITOS ACADÉMICOS Y DE APRENDIZAJE EXCLUSIVAMENTE DE OPTEX; SI SE DESEA UTILIZAR LA FORMULACIÓN ALGEBRAICA Y/O LOS PROGRAMAS DE COMPUTADOR CON PROPÓSITOS COMERCIALES SE DEBE ADQUIRIR UNA LICENCIA FORMAL DEL SOFTWARE. PARA UTILIZAR ESTE MATERIAL COMO PARTE DE UN PROCESO LIBRE SE DEBE TENER UNA AUTORIZACIÓN ESCRITA Y FIRMADA POR DO ANALYTICS.

DO ANALYTICS MANTIENEN LA PROPIEDAD DE ESTE DOCUMENTO Y PODRÁ SOLICITAR SU DEVOLUCIÓN Y/O SU DESTRUCCIÓN EN CUALQUIER MOMENTO.

1. GENERALIDADES

El enfoque original de formulación de modelos de **OPTEX** está basado en un lenguaje algebraico que se almacena en tablas de la base de datos **SIMM**. Esta característica se convierte en una de las peculiaridades más importantes de **OPTEX**, que lo diferencian de la mayoría de sistemas desarrollados para implementar modelos de programación matemática, principalmente en lo que se refiere al desarrollo concurrente de modelos.

Sin embargo, con el ánimo de ampliar el campo de acción de **OPTEX, DW** desarrolló un lenguaje algebraico de programación de computadores para resolver problemas de programación matemática siguiendo los estándares utilizados para el desarrollo de modelos. Los principios del lenguaje algebraico **OPTEX** (**OPTEX Computer Language**) son similares a los de **GAMS**, **AMPL** o **GNU Mathprog**, pero se caracteriza por su orientación a la programación basada en objetos.

La representación algebraica de modelos en el lenguaje de computación de **OPTEX** puede ser fácilmente leída por los modeladores y procesada por los computadores; esto implica que en si un programa **OPTEX** contiene la documentación del modelo, lo que evita la necesidad de documentos separados uno para la descripción del modelo y otro para su implementación.

La representación de modelos en **OPTEX** es concisa y hace uso apropiado de la representación algebraica por medio de ecuaciones. Se trata de facilitar al usuario la definición de los elementos del modelo matemático en su forma más elemental, y su posterior integración por medio de los vínculos que existen entre los diferentes componentes. La idea es hacer de los modelos matemáticos mas asequibles, mas entendibles, mas verificables y por ende más creíbles.

El proceso de compilación trabaja como un compilador de doble pasada: en la primera analiza la sintaxis del programa y en la segunda la lógica de su contenido; si todo esta correcto se encadena con **OPTEX-MG**. Para la edición de los programas se utiliza el software libre **NOTEPAD++.**

A partir de un programa **OPTEX** es posible llenar las tablas del sistema de información de **OPTEX,** o sea formular el modelo en un lenguaje algebraico basado en tablas, lo que permite el modelamiento simultáneo por múltiples usuarios concurrentes. De esta forma un programa **OPTEX** se convierte en un medio de llenado de las tablas de **OPTEX**. De manera simétrica, **DW** ha desarrollado la exportación de modelos basados en tablas a su correspondiente versión en un programa en lenguaje algebraico.

La correspondencia en **OPTEX** entre el lenguaje computacional y el lenguaje algebraico basado en tablas es total, de tal forma que los elementos fundamentales del lenguaje computacional tienen un equivalente directo en el modelaje basado en tablas. De esta forma el Manual del Administrador de **OPTEX** se puede considerar complementario a este Manual.

Dado que se está en fase de desarrollo, este manual debe considerarse como un Documento de Trabajo que especifica el diseño de lo que se ha implementado, o de lo que se implementará en el futuro mediato.

2. ESTRUCTURA GENERAL

OPTEX Computer Lenguage está basado en el concepto de objetos y de atributos de dichos de dichos objetos. La elaboración de un programa **OPTEX** implica la definición de diferentes tipos de objetos y su encadenamiento por medio de sus atributos.

Como lenguaje de computadores se tiene una primera fase de declaración de objetos y una segunda fase de ejecución de comandos e instrucciones que utilizan los objetos previamente declarados. Se considera la definición de tres tipos de objetos:

- **Modelamiento de Datos (Data Modeling):** relacionados con la especificación de las fuentes de datos que se utilizan en los modelos

- **Modelamiento Matemático (Mathematical Modeling):** relacionados con la especificación de las componentes de los modelos matemáticos y de su interrelación
- **Datos (Data):** relacionado con los datos que se insertan en el programa

La siguiente tabla presenta las palabras clave asociadas a los objetos de cada tipo.

OPTEX Computer Lenguage – Objects		
Data Modeling	**Mathematical Modeling**	**Data**
entity field master-table server table	alias constraint decision-tree index model node objective-function parameter period problem scenario set solver time-horizon variable	data-table

Los atributos para cada entidad se presentan en la siguiente tabla

OPTEX Computer Language – Objects – Attributes	
Object	**Attributes**
Data Modeling Objects	
field	long, type, units
server	type, directory, dsn, password, user
table	field, server
Mathematical Modeling Objects	
alias	index, set, variable
constraint	index, formulae, lhs, rhs, set, type, units
decision-tree	node, variable
index	entity-type, sector, master-table, field, gis, data-table type,
model	model, problem, type
node	random-path, period-initial, period-final, probability
objective-function	constraint, parameter, sign, variable, variable2
parameter	data-table, default, field, form, formulae, function, index, series, table, type, units, validation
period	type-period, number-periods
problem	class, constraint, rol, variable
scenario	date-initial, model, objective-function, type-optimization
set	data-table, field-element, field-index, filter, index, index-element, operation, set, table
solver	library, lp-algorithm, mip-option, nlp-algorithm
time-horizon	delta-period, number-periods, period, time-horizon, type, type-period
variable	index, formulae, lower, set, type, units, upper
Data Objects	
data-table	index, field

Adicional a los anteriores atributos, todos los objetos tienen los atributos **description** y **comment**, los cuales se utilizan para describir los objetos y cuando se va a transferir informacion a la base de datos de **OPTEX**.

De acuerdo con la información que almacenan los atributos son de dos tipos: escalares o vectoriales. Los vectoriales permiten almacenar varios valores para el mismo atributo, como puede ser el caso del atributo **index** en las variables, restricciones y parámetros. En este caso el orden de los valores definidos es importante ya que puede relacionarse con otros atributos que también son vectoriales, como es el caso del atributo **set** en las variables y restricciones que está relacionado con el atributo **index**.

Los comandos de **OPTEX Computer Lenguage** se presentan en la siguiente tabla.

OPTEX Computer Lenguage - Comands
solve
recover

3. SINTAXIS

La escritura de programas en **OPTEX Computer Lenguage** debe seguir ciertas reglas las cuales se describen a continuación:

- **ESCRITURA**
  - El lenguaje se basa en cláusulas que se van especificando cada una en una línea, o múltiples cláusulas en una línea separándolas por punto y coma (**;**)
  - Se toma como inicio de comentarios en una cláusula el doble slash (**//**)
  - Para bloque de comentarios se utiliza al inicio flash asterisco (**/***) y para finalización asterisco flash (***/**).

- **DECLARACIÓN DE OBJETOS**
  - Para definir un objeto se abre un proceso de declaración el cual comienza con el nombre del objeto que se está definiendo y termina con la cláusula **end**
  - El nombre dado al objeto se establece a continuación, del nombre del tipo de objeto
  - Los atributos del objeto se definen entre la declaración del objeto y la cláusula **end**. Cada atributo debe comenzar con un punto (**.**) continuando con el nombre del atributo y finalizando con el signo igual (**=**). El valor del atributo será la expresión que siga a continuación, del igual atributo **index** puede definirse en la declaración del objeto especificando entre paréntesis los componentes de **index**.
  - Cuando el atributo es vectorial se dispone de dos alternativas. La primera es la definición secuencial de los atributos en múltiples cláusulas, la segunda es la definición de los valores en una sola cláusula encerrando los valores entre corchetes (**{}**).
  - Cuando se declara un objeto en algunos casos se asignan valores de inicialización por defecto, hay atributos que no se inicializan.
  - Cuando un atributo corresponde a un objeto, este debe definirse como tal. No es necesario que cuando se asocia un objeto a un atributo esta ya haya sido definido, en otras palabras, el orden de declaración no importa.
  - Cuando los atributos no están asociados a objetos, en varios casos existen valores pre-definidos que determinan la validez del atributo.

- **ESPECIFICACIÓN DE COMANDOS**
  - Los comandos están asociados a acciones específicas que realiza el procesador de OPTEX. Normalmente incluyen parámetros que están asociados a los objetos que previamente se han declarado

4. OBJETOS

4.1. DATA MODELING OBJECTS

Los **Data Modeling Objects** están relacionados con la especificación de la conectividad a servidores de datos en los que residen las tablas que aportan los datos que definen la topología y las especificaciones técnico-económicas del sistema que se está modelando. La declaración de este tipo de objetos implica definir datos para:

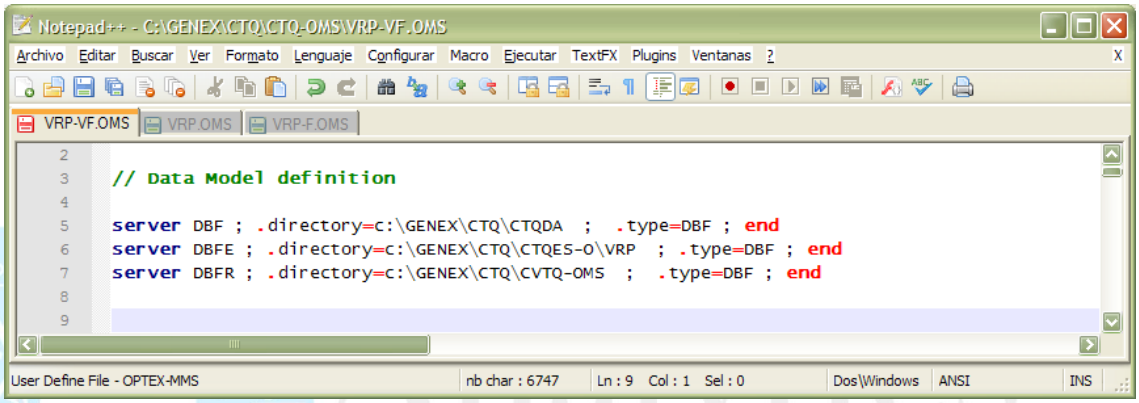
- Servidores de datos
- Tablas de datos
- Campos de las tablas de datos

4.1.1. SERVIDORES

Los servidores especifican fuentes de datos que se requieren como input de los modelos, para ello se utiliza el objeto **server**, para el cual hay definir los siguientes atributos:

- **name:** nombre dado al servidor
- **directory:** directorio donde están las tablas de datos, aplica para tablas tipo **dBase** o hojas **EXCEL**
- **type**: tipo de archivo cuando se accede a archivos en un directorio
- **dsn**: Data Source Name para acceso a servidores tipo **SQL** vía **ODBC**s.
- **user:** usuario para acceso al servidor
- **password:** palabra clave para acceso al servidor

La siguiente pantalla presenta ejemplo de declaración de **server**s.

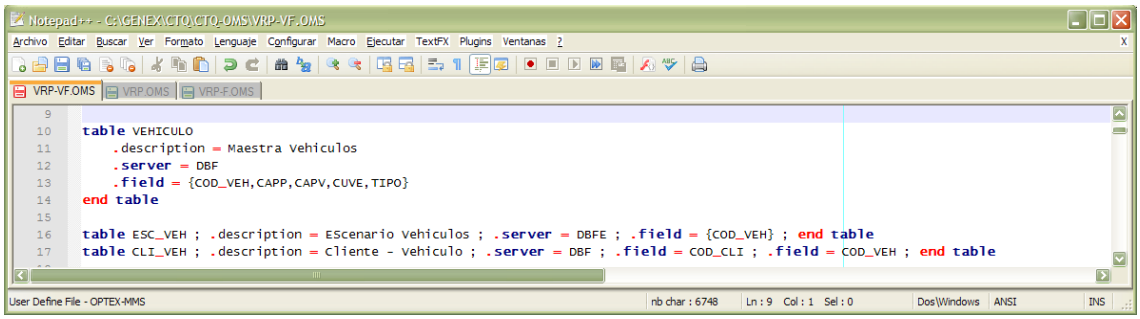


4.1.2. TABLAS DE DATOS

Para acceder aun tabla de datos, en un servidor o en un directorio, se debe declarar la tabla utilizando el objeto **table**. Los atributos a definir son:

- **name:** nombre de la tabla, debe coincidir con el nombre real
- **server**: servidor a utilizar para acceder a la tabla, debe ser del tipo **server**
- **field**: atributo vectorial que almacena los campos de la tabla, solo se requiere para tablas output

La siguiente pantalla presenta ejemplo de declaración de **table**s.



4.2. MATHEMATICAL MODELING OBJECTS

Los **Mathematical Modeling Objects** están relacionados con el lenguaje algebraico basado en tablas con el proceso de desarrollar el **SIMM** (**Sistema de Informacion de Modelos Matemáticos**) que implica configurar los siguientes objetos para:

- Definiciones básicas;
- Configuración de Modelos; y
- Configuración de Escenarios.

Las definiciones básicas están relacionadas con aspectos generales de modelaje que se pueden utilizar en todos los modelos y problemas que se configuren, y por lo tanto no están asociadas a un modelo o problema en particular. Se deben definir los siguientes aspectos:

- Índices
- Conjuntos
- Parámetros
- Variables
- Restricciones
- Alias
- Función Objetivo

El proceso de configuración de modelos tiene dos etapas en las que se debe definir:

- Problemas y
- Modelos.

De acuerdo a la teoría de gran escala un modelo está integrado por múltiples problemas; y cada problema cumple un rol dentro del modelo integrado. Cuando no se aplican técnicas de gran escala el modelo coincide con el problema básico.

4.2.1. ÍNDICES

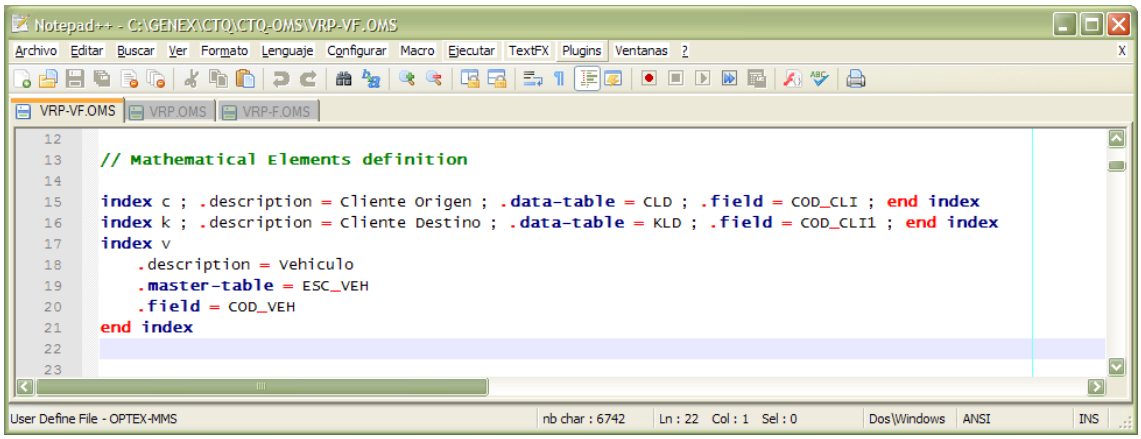
Los índices asociados a las variables, ecuaciones y parámetros que se manejan en los modelos deben declararse para ello se utiliza el objeto **index**. Están asociados directamente a entidades que se manejan en el sistema que se está modelando. Los subíndices **t** y **q** deben asociarse al tiempo. El proceso de configuración de un índice implica la definición de los siguientes atributos:

- **name:** corresponde a una letra minúscula que se utiliza para referenciar en los modelos matemáticos la entidad asociada al índice.
- **description:** descripción, o nombre, de la entidad asociada al índice.
- **master-table:** cuando un índice es de tipo alfanumérico y el universo de posibles valores esta almacenado en una tabla de datos asociada se debe especificar este atributo que debe ser del tipo **table**.
- **field:** define el campo en la tabla de datos, cuyo contenido servirá para establecer el universo de posibles valores para el índice. Se requiere Normalmente estará asociado al código relacional de la entidad relacionada con el índice. En el caso de que no exista tabla de datos asociada al contenido de este campo se ignora.
- **sector:** se utiliza únicamente en casos de modelaje con partición y descomposición y asocia la entidad que representa el índice a un sector o a una área intersectorial.
- **type:** señala si el índice es:
  - **N** Numérico: índice con contenido numérico como: etapas de proyectos de inversión, secuencias de producción y similares
  - **T** Tiempo: índice las divisiones del período de planificación
  - **I** Incertidumbre: índice asociado con posibilidades aleatorias
  - **A** Alfanumérico: índice asociado a entidades que se asocian con un código alfanumérico.

  Se debe notar que el índice **t** siempre debe estar asociado al tiempo.
- **entity-type:** se utiliza para especificar el tipo de entidad a la que asociado el índice, esta caracterización se utiliza para visualizar los resultados de los modelos. Se consideran los siguientes tipos de entidades:
  - **I** Instalación
  - **F** Flujo
  - **T** Tiempo
  - **I** Escenario aleatorio

- **gis:** indica si la entidad esta georeferenciada en un sistema **GIS**. Valores validos TRUE o FALSE

La siguiente pantalla presenta ejemplo de declaración de **index**s.



4.2.2 CONJUNTOS

Los conjuntos que determinan las condiciones de existencia de las variables, de las restricciones y de los problemas se deben declarar, para ello se utiliza el objeto **set**. Se consideren dos tipos de conjuntos:

- **CONJUNTOS PRIMARIOS:** se definen directamente a partir de las fuentes de datos
- **CONJUNTOS SECUNDARIOS:** resultan de operaciones entre conjuntos

Los conjuntos contienen valores para el índice dependiente con base en el valor de otros índices que actúan como índices independientes.

OPTEX genera un conjunto de elementos asociados al índice dependiente, a partir de la búsqueda en una fuente de datos que puede ser una **table** o en una **data-table**. Adicionalmente, es posible establecer una condición sobre un campo de la **table** para que actúe como condición de selección de elementos; en caso de que esta condición no se defina, todos los valores existentes en la **table** de datos conformarán el conjunto.

La configuración de un conjunto implica la definición de los siguientes atributos:

- **name:** código del conjunto
- **description:** descripción del contenido del conjunto
- **index**: atributo vectorial que almacena los índices que actúan como independientes.
- **field-index**: este atributo almacena información correspondiente al campo que contiene el código de los **index** cuando la fuente de datos del índice es una **table**.
- **index-element**: corresponde al índice dependiente que define el tipo de entidad contenida en el conjunto.
- **table**: nombre de la tabla de datos que se utilizará para construir el conjunto. Corresponde a un objeto tipo **table**.
- **field-index:** corresponde al campo de la **table** que contiene los elementos del conjunto.
- **filter:** este campo corresponde a una condición de existencia para los elementos del conjunto. Para establecer la condición se debe especificar el campo de la **table** sobre el cual se establece la condición, la condición y el valor de referencia. Las condiciones validas son :

=	igual que
<=	menor o igual que
<	menor que
>=	mayor o igual que
>	mayor que
<>	diferente a

NV No vacío (el contenido del campo no debe ser vacío)

El valor de referencia se establece de acuerdo al tipo de campo sobre el cual se establece la condición: En el caso de campos numéricos se debe especificar un número, para cadenas de caracteres se debe especificar la cadena y para campos lógicos los posibles valores son:

.T. (True) Verdadero
.F. (False) Falso

- **data-table**: nombre de la **data-table** que se utilizará para construir el conjunto cuando las componentes se especifican dentro del programa **OPTEX**. Los índices de la **data-table** de coincidir con los **index** del conjunto, no necesariamente en el mismo orden.
- **operation:** Cuando el conjunto no se lee de una tabla se debe establecer la operación del conjunto que se debe utilizar. La operación se define sobre uno o dos conjuntos que se deben especificar secuencialmente en el atributo **set**.

Es posible realizar los siguientes tipos de operaciones entre conjuntos:

C **Complemento:** la operación complemento se realiza con relación a un conjunto universal que se define para el índice dependiente y que corresponde a todos los elementos encontrados en la Tabla Maestra del Índice.

X **Indexación:** es idéntica a la superunión, pero el conjunto que se utiliza para la suma solo tiene un elemento para cada índice independiente, es así que la suma que se hace en la superunión, es de un solo elemento.

I **Intersección:** se realiza entre conjuntos cuyos índices dependientes sean iguales.

S **Super Unión:** el índice dependiente de un conjunto C1 y el independiente del conjunto C2 deben ser iguales. Esta operación suma sobre dicho índice común, obteniendo un conjunto con índice dependiente igual al dependiente del conjunto C2. En la suma, el índice sobre el cual se está sumando, desaparece del conjunto resultado.

U **Unión:** se realiza entre conjuntos cuyos índices dependientes sean iguales

En la especificación de los índices independientes de un conjunto que sea el resultado de la operación de dos conjuntos se deben tener en cuenta los índices dependientes de los conjuntos que intervienen en la operación (operandos). Como regla se debe cumplir que los índices dependientes del conjunto resultado serán igual a la unión de los índices dependientes de los operandos.

Ejemplos:

Complemento (C): Sea B el conjunto universal de bebidas producidas en una planta. El conjunto de bebidas no alcohólicas BNA es el complemento del conjunto BA de bebidas alcohólicas respecto del conjunto universal.

$$\overline{\mathbf{BNA}} = \mathbf{BA}$$

Intersección (I): Sea el conjunto **CTR(c,g)** agrupa las marcas transportables por cervecería y es el resultante de la siguiente operación

Marcas por cervecerías ∩ Marcas transportables

lo que equivale a:

CC(c) ∩ PTR()

Obsérvese que para poder hacer una intersección, es indispensable que los dos índices dependientes sean iguales. El conjunto PTR() no tiene índice independiente porque únicamente denota las marcas que son transportables. El resultante de la operación toma los índices de CC.

SuperUnión (S): Sea PL el conjunto de líneas según producto, esto es, PL(l/p) y sea EL el conjunto de envasadoras según líneas, LE(e/l). Si se desea obtener

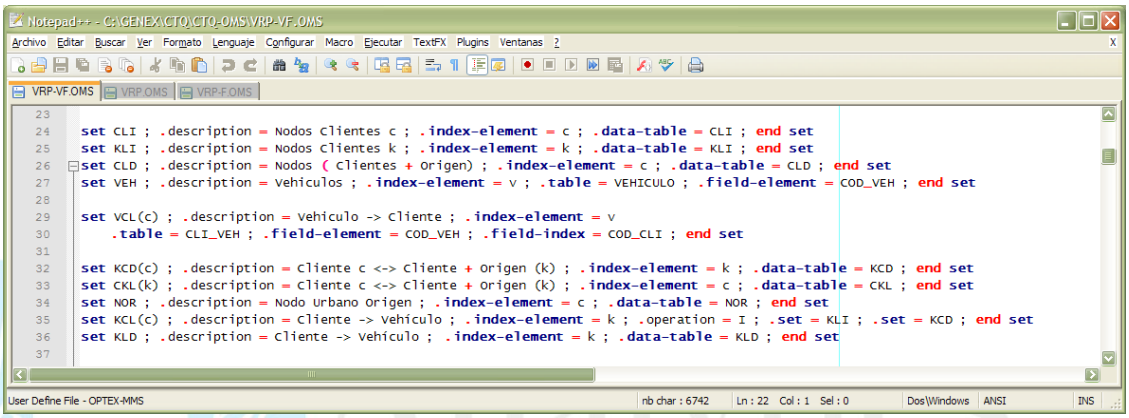
envasadoras según producto (PE), se puede hacer la superunión de estos dos conjuntos

$$PE(e/p) = S\ [PL(l/p)]\ LE(e/l)$$

Unión (U): para los conjuntos PG de plantas productoras de bebidas gaseosas asociado al índice p, y PA de plantas productoras de bebidas alcohólicas asociado al mismo índice p, la unión de los dos conjuntos da como resultado la suma de todas las plantas que producen bebidas

$$PGA = PG \cup PA$$

La siguiente pantalla presenta ejemplo de declaración de **sets**.



4.2.3 PARÁMETROS

Se deben declarar los parámetros de los modelos, asociándoles un nombre que se utiliza en las ecuaciones del modelo matemático, para ello se utiliza el objeto **parameter**. El valor de un parámetro puede establecerse por dos vías:

- A partir del contenido de un campo de una **table** o en una **data-table**; y
- Como resultado de la evaluación de una ecuación que involucra otros parámetros.

Los atributos que se deben especificar suministrar para declarar un parámetro son los siguientes:

- **name:** código asociado al parámetro.
- **description:** descripción del parámetro.
- **index**: atributo vectorial que almacena los índices que determinan la variabilidad del parametro
- **type:** indica el formato en el cual se maneja la tabla de datos que se utilizará para cargar los parámetros. Se consideran los siguientes tipos de formatos:
  - **C** Parámetro Calculado
  - **R** Relacional
  - **S** Serie de Tiempo
  - **T** Tabla de Datos
- **type-period:** para el formato tipo tabla (**T**) indica la unidad de tiempo bajo la cual esta organiza la tabla. Son válidos los siguientes valores:
  - **A** Años
  - **M** Meses
  - **Q** Quincenas
  - **S** Semanas
  - **D** Días
  - **H** Horas
- **form:** ca la forma como se deben interpretar las series de datos temporales que se cargan de la tabla. Las formas de interpretación son:

E Escalón
I Impulso
P Polilínea

- **form-value:** indica la forma como se debe calcular el parámetro en el proceso de generación de las matrices y vectores del modelo matemático. Las formas de cálculo posibles son:
  I **Integral:** Durante el período asociado.
  M **Media:** Durante el período asociado
  V **Valor:** Al comienzo del período.
- **table:** corresponde al nombre de la **table** en la que se encuentran almacenados los valores del parámetro.
- **data-table:** corresponde al nombre de la **data-table** en la que se encuentran almacenados los valores del parámetro.
- **field:** corresponde al **field** asociado a la **table** o a la **data-table** en que se encuentran almacenados los valores del parámetro.
- **function:** información correspondiente a la función de proyección asociada al parámetro, y puede ser el nombre de una función programada por el administrador, o valores numéricos constantes.
- **default:** es el valor asignado a un parámetro cuando es cargado de una **table** o de una **data-table** y no se encuentra especificado para determinado valor de los índices.
- **validation:** especifica la condición de validez que debe cumplir el parámetro para que se acepte como válido. Cuando esta condición no se cumple el compilador reporta error.
- **formulae:** fórmula de cálculo del parámetro.

En caso que el parámetro este definido con base en una ecuación, esta debe expresarse acuerdo con las normas especificadas en el numeral correspondiente al lenguaje algebraico La ecuación podrá contener otros parámetros, funciones de parámetros y/o valores numéricos.

En el proceso de determinación del valor de los parámetros se deben tener en cuenta tres aspectos fundamentales que se analizan detalladamente:

- Carga de los datos a partir de **table** o **data-table**
- Calculo de parámetros como resultados de operaciones sobre otros parámetros
- Proyección de parámetros dinámicos por medio de las funciones de proyección.

Cuando el parámetro es cargado a partir de una **table** los campos asociados a los **index** (índices) que definan la variabilidad de los parámetros se determinan con base en el **field** especificados en la declaración del objeto **index**.

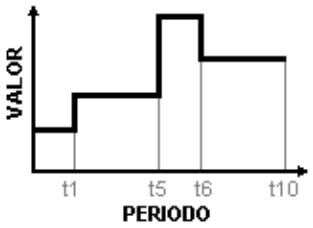
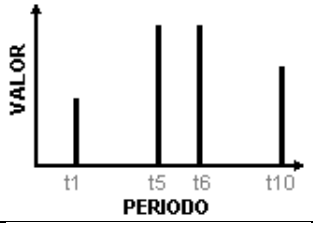
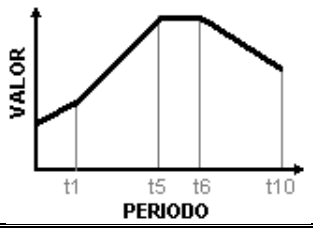
- **TIPOS DE TABLAS**

Las tablas de parámetros pueden ser de diferentes tipos, tal como se presenta en la siguiente tabla:

TIPOS DE TABLAS	
TIPO DE TABLA	**DESCRIPCIÓN**
R **RELACIONAL**	Estas tablas almacenan datos en forma relacional. En el momento de generar las matrices y vectores del problema, se buscará en la tabla un registro asociado a los subíndices del parámetro, y se determinará con base en el contenido del campo especificado. Se asume que este valor corresponde al valor del parámetro en el tiempo cero (t=0) y después será proyectado en el tiempo de acuerdo con la función de proyección temporal. Las tablas de este tipo están compuestas por al menos un campo de relación que actúa como llave y un campo que contiene el valor del parámetro.
S **SERIE**	En esta tabla se almacenan los valores de los parámetros definiendo el valor de la serie de datos en momentos específicos en el tiempo. La estructura de las tablas incluye el campo FECHA y los campos correspondientes a los valores de los parámetros del modelo. Los valores de los parámetros se determinan ubicándolos en el tiempo de acuerdo al valor del campo FECHA y asociándolos de acuerdo con el valor de los índices del parámetro.
T **TABLA**	En este caso se almacenan en un mismo registro varios datos correspondientes a diferentes momentos en el tiempo dentro de un período. Las filas de la tabla se clasifican con base en claves relacionales y una fecha que identifica el período al cual corresponden los datos almacenados. Las columnas corresponden a diferentes momentos en el tiempo dentro del período identificado por la fecha. La estructura de la tabla depende de la resolución temporal a la que esta asociada. Los niveles de agregación son: ▪ horario: veinticuatro horas para un día ▪ diario: treinta y un días para un mes ▪ mensual: doce meses para un año A partir de los datos leídos se conforma una serie de tiempo que posteriormente será interpretada de acuerdo al tipo de serie de tiempo especificada.

- **INTERPRETACIÓN DE LAS SERIES**

Los parámetros almacenados en forma de series de tiempo se interpretan de acuerdo al tipo de serie seleccionada. Las posibilidades implementadas se presentan en la siguiente tabla:

SERIES		
TIPO DE SERIE	**DESCRIPCIÓN**	**INTERPRETACIÓN**
E **ESCALÓN**	Orientado al manejo de series de datos almacenadas en forma de escalón con base en un punto de quiebre de la serie: entre un punto y hasta el siguiente se asume un valor igual al punto inicial del tramo	
I **IMPULSO**	Orientado al manejo de series de datos almacenadas con base en los momentos (períodos) en el tiempo en que la serie es diferente de cero: en puntos no especificados en la tabla el parámetro se asume cero.	
P **POLILÍNEA**	Orientado al manejo de series de datos almacenadas con base en los puntos de quiebre de la pendiente de la serie: entre dos puntos de quiebre se asume una pendiente constante.	

- **CÁLCULO DE PARÁMETROS**

La determinación de los parámetros puede involucrar procesos de cálculo. Las posibles vías para calcular un parámetro son:

- Series temporales

- Fórmulas matemáticas

- **VÍA SERIES TEMPORALES**

A partir de la interpretación de la serie de tiempo se calculan los parámetros asociados. Las formas de cálculo posibles son:

SERIES TEMPORALES	
CALCULO DEL PARÁMETRO	**DESCRIPCIÓN**
I INTEGRAL	El valor del parámetro para el período t se calcula como la integral de la serie entre el instante asociado al período t y el instante asociado al período t+1.
M MEDIA	El valor del parámetro para el período t se calcula como el valor promedio de la serie en el período comprendido entre el instante asociado al período t y el instante asociado al período t+1.
V VALOR	El parámetro para el período t resulta de calcular el valor medio de la serie en el momento.

- **VÍA FORMULAS**

En caso de que el parámetro este definido con base en una ecuación, esta debe expresarse de acuerdo con las normas que se especifican en el numeral correspondiente a lenguaje algebraico. La ecuación solo podrá contener parámetros, y su evaluación corresponderá al valor del parámetro para el tiempo cero.

Ejemplo:
El costo unitario **AAI(c,g)** de producir una marca **g** en una planta **c** puede ser el resultado de sumar los costos de todos los insumos mas un costo base derivado de la planta y el producto:

$$\mathbf{AAI(c,g) = CUC(c,g) + \sum_{weINS(g)} URC(c,w)\ KTC(c,g,w)}$$

donde **CUC(c,g)** es el costo unitario de producir una marca por cervecería para recursos agregados, **URC(c,w)** es el costo unitario de los recursos desagregados, **KTC(c,g,w)** es el consumo de recursos desagregados por marca y **INS(g)** define el conjunto de insumos que se utilizan para producir la marca g. Todos los factores que intervienen en la estimación de **AAI** son datos de entrada al sistema.

- **FUNCIONES DE PROYECCIÓN**

La variación temporal de los parámetros se asume puede provenir de modelo dinámico. Para ello **OPTEX** incorpora una metodología, basada en la definición de funciones de proyección temporal. El proceso para el cálculo del valor del parámetro es:

- Inicialmente **OPTEX** evalúa el valor del parámetro sin tener en cuenta la función de proyección temporal. Este valor se asume que corresponde al valor del parámetro en el tiempo cero (**t=0**).
- Posteriormente, cada vez que se requiera evaluar el parámetro para un determinado período igual a **t**, él mismo es calculado como el resultado de multiplicar el valor del parámetro en el tiempo cero por el valor de la función de proyección temporal evaluada en la fecha asociada al período **t**, esto es:

parámetro(t) = parámetro(0) * FUNCION_DE_PROYECCION(t)

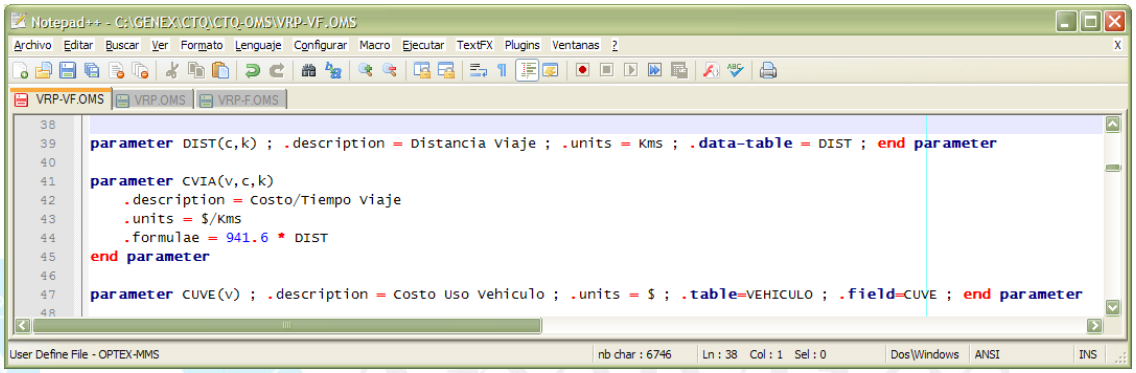
La función de proyección temporal da gran flexibilidad al manejo de los parámetros. A continuación, se citan ejemplos típicos en los que aplica:

- **Multiplicar un parámetro por factores que son variables en el tiempo**: Se recurre al uso de funciones de proyección para determinar factores que varían en el tiempo. A manera de ejemplo, se tiene la disponibilidad de horas ordinarias en una planta para un

período de tiempo, que depende del número de días del período. Otro caso es el factor de descuento para ingresos y egresos a valor presente. También se pueden considerar el caso de parámetros que son el resultado de un modelo dinámico como puede ser la proyección de un precio que tiene variaciones estacionales.

- **Multiplicar un parámetro por factores que no tienen una lógica sencilla:** Cuando un parámetro toma distintos valores a lo largo del tiempo y su lógica de cálculo rompe con los mecanismos estándares de cálculo del parámetro, el usuario puede programar funciones de proyección totalmente personalizadas de acuerdo con los requerimientos del problema. Estas funciones de proyección deben ser programas en C, de acuerdo con un protocolo definido posteriormente.

La siguiente pantalla presenta ejemplos de declaración de **parameter**s.

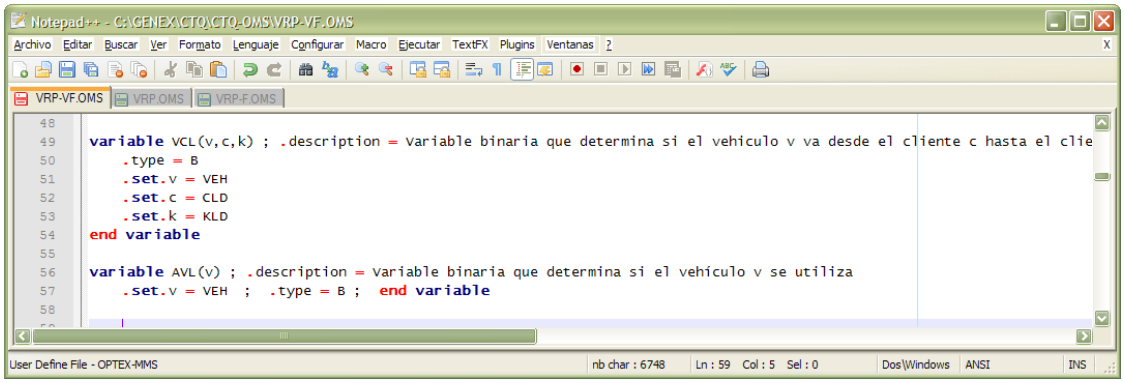


4.2.4 VARIABLES

Cada variable se debe declarar, para ello se utiliza el objeto **variable.** Los atributos que se deben definir para cada variable es:

- **name:** nombre asociado a la variable.
- **description:** descripción de la variable. **OPTEX** utiliza esta información en el despliegue de los resultados.
- **index**: atributo vectorial que almacena los índices que determinan la variabilidad de la variable
- **set**: atributo vectorial que almacena los conjuntos que determinan la existencia de la variable. Deben estar coordinados secuencialmente con los **index**.
- **type:** define el tipo de variable de acuerdo con:
  - **C** Continua
  - **B** Binaria
  - **E** Entera
- **upper**: puede contener uno de los siguientes valores:
  - Nombre del parámetro que define la cota
  - Valor numérico
  - INFINITO para variables no acotadas
- **lower:** puede contener uno de los siguientes valores:
  - Nombre del parámetro que define la cota
  - Valor numérico
  - Si no se define se asume cero.

La siguiente pantalla presenta ejemplos de declaración de **variable**s.



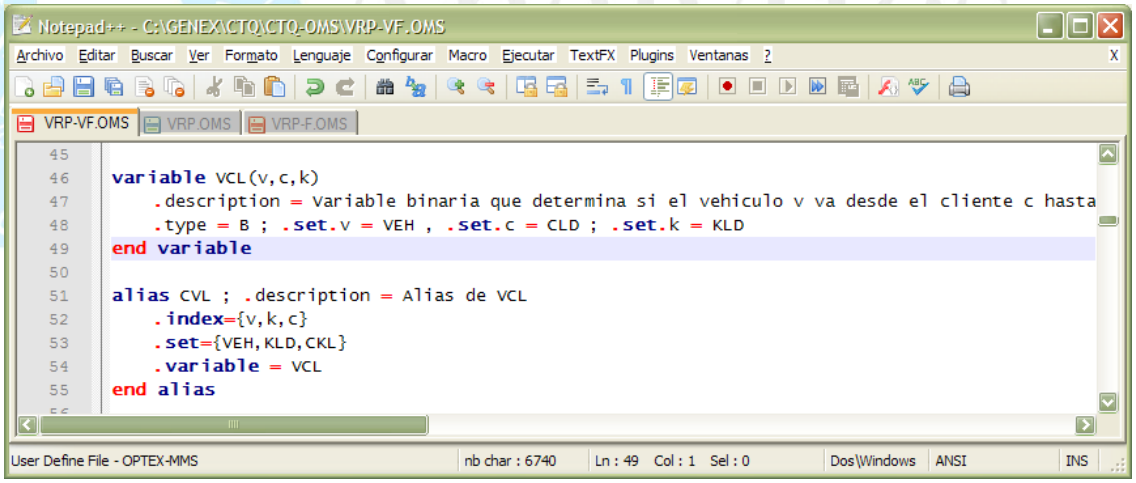
4.2.5 ALIAS

Para el caso de variables que estén relacionadas con enlaces entre entidades del mismo tipo, se requiere utilizar el concepto de alias de la variable original en la que intercambia el orden los índices. Para ello se utiliza el objeto **alias**.

Los atributos que se deben definir son:

- **name:** nombre asociado a la variable alias.
- **index**: atributo vectorial que almacena los índices que determinan la variabilidad del alias
- **variable**: nombre dado a la variable original

La siguiente pantalla presenta ejemplos de declaración de **alias**s y de **variable**s.



4.2.6 RESTRICCIONES

De manera similar a las variables y a los parámetros se debe declarar cada restricción para ello . Los valores correspondientes a las variables duales y a las de holgura se calculan por medio de la solución del problema de optimización. En caso de que se desee almacenar el valor de las restricciones en el **SIDI**, se deberá definir la conectividad correspondiente.

La información a definir para cada restricción es:

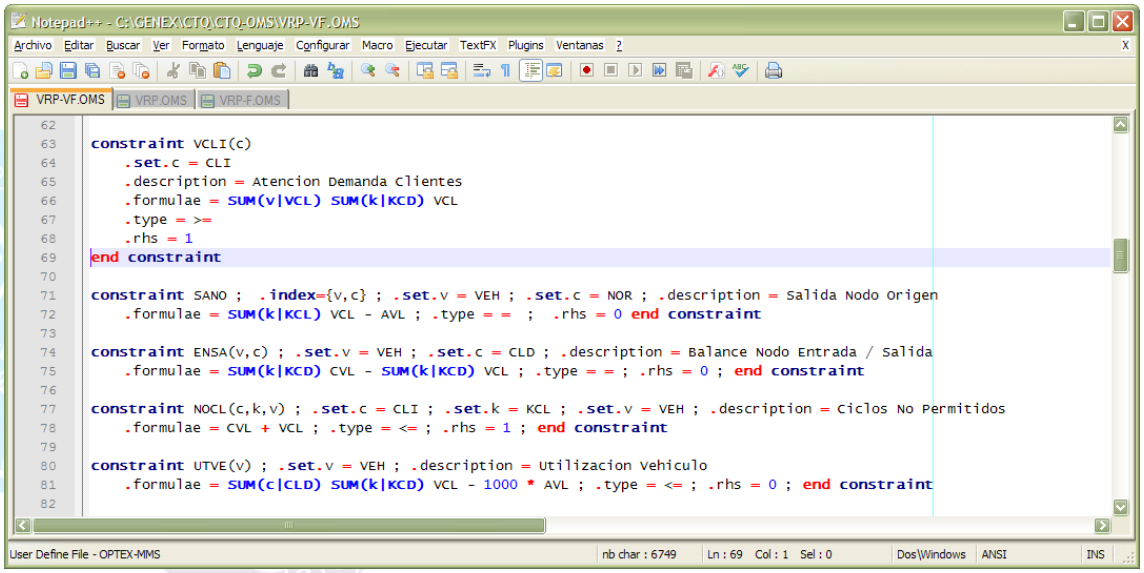
- **name:** código asociado a la restricción.
- **description:** descripción de la restricción. **OPTEX** utiliza esta información en el despliegue de los resultados.
- **index**: atributo vectorial que almacena los índices que determinan la variabilidad de la restricción
- **set**: atributo vectorial que almacena los conjuntos que determinan la existencia de la variable. Deben estar coordinados secuencialmente con los **index**.
- **type:** define el tipo de restricción de acuerdo con:

= Igual a
≤ Mayor o igual que
≥ Menor o igual que
: Menor o igual que y mayor o igual que

- **lhs:** puede contener uno de los siguientes valores:
  - Nombre del parámetro que define el coeficiente; o
  - Valor numérico.
- **rhs:** puede contener uno de los siguientes valores:
  - Nombre del parámetro que define el coeficiente; o
  - Valor numérico

  Aplica cuando el tipo es menor y mayor igual que (**:**) corresponde a la condición mayor o igual que.

La restricción deberá expresarse de acuerdo con las normas especificadas en el numeral correspondiente al lenguaje algebraico y deberá contener productos de parámetros, valores numéricos y variables.
La siguiente pantalla presenta ejemplos de declaración de **constraint**s.



4.2.7 FUNCIONES OBJETIVO

Se ha establecido independencia entre una variable y su costo en la función objetivo. Esta conceptualización permite que un modelo pueda utilizarse con múltiples funciones objetivo y en procesos de optimización multicriterio.

Las ventajas de este enfoque radican en la posibilidad de analizar el sistema desde diferentes puntos de vista, lo que es fundamental para el análisis multiobjetivo. Caso que se aplica en la optimización de sistemas multipropósito que se modelan con base en múltiples funciones objetivo que se ponderan en un solo objetivo o que se ordenan jerárquicamente.

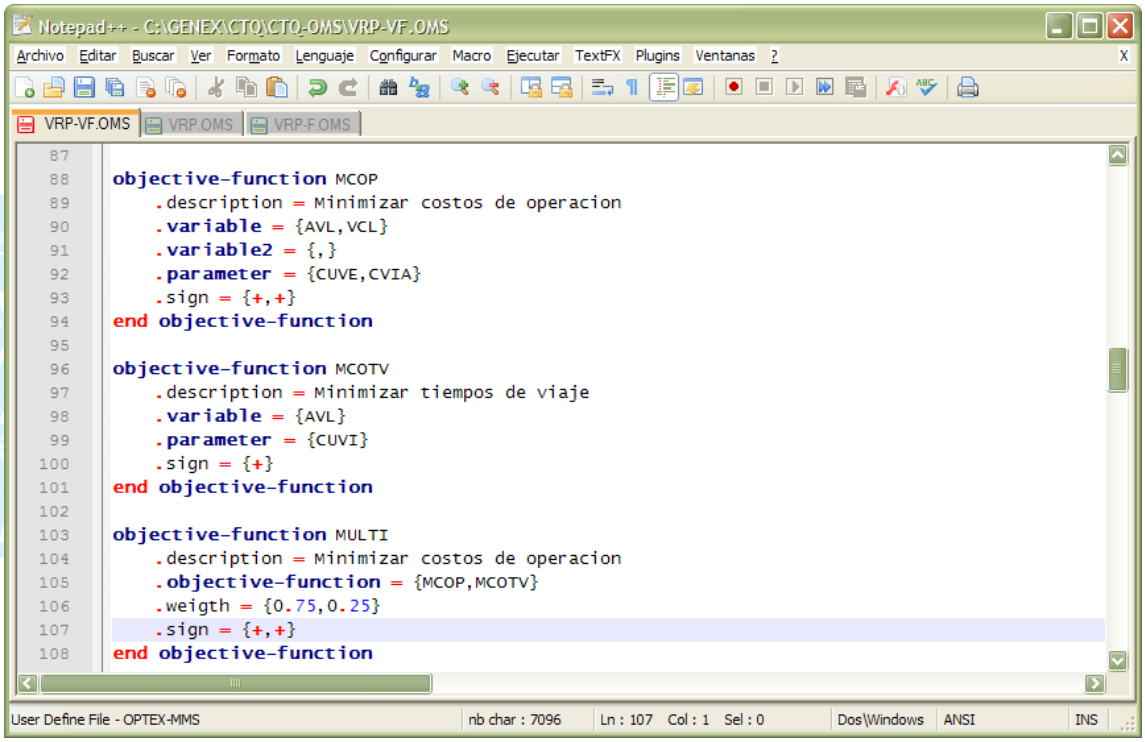
El usuario debe definir las Funciones Objetivo que desea vincular a los modelos, para ello debe utilizar el objeto **objetive-function**

Los atributos propios que se deben definir son;

- **name**: nombre dado a la función objetivo
- **variable**: atributo vectorial que almacena las variables que participan en la función objetivo con costo diferente de cero
- **variable2**: atributo vectorial que almacena la componente de las variables que participan en la función objetivo con costo diferente de cero. Cuando no hay componente cuadrática se debe dejar la posición en el vector en blanco

- **parameter**: atributo vectorial que almacena los parámetros o los valores numéricos asociados a las variables que participan en la función objetivo con costo diferente de cero
- **sign**: indica el signo con el cual se toma el parámetro asociado a la variable
- **constraint**: sustituye a los anteriores atributos e indica la ecuación que se utilizará como función objetivo
- **objetive-function**: atributo vectorial que almacena las funciones objetivo que participan en una función objetivo multicriterio ponderada
- **weigth**: atributo vectorial que almacena los pesos de las funciones objetivo que participan en una función objetivo multicriterio ponderada

Cuando se utilizan los atributos **variable**, **variable2**, **parameter** y **sign**, la expansión de la función objetivo es automática en **OPTEX**, de tal forma que **OPTEX** genera la sumatoria para todos los valores válidos para una variable, o una pareja de variables en el caso cuadrático, multiplicadas por el parámetro indicado. Cuando se utiliza la opción **constraint**, como **objetive-function** se utiliza la **formulae** asociada a la **constraint**.



4.2.8. LENGUAJE ALGEBRAICO

Para el manejo de las fórmulas matemáticas de las ecuaciones involucradas en un modelo se ha concebido un lenguaje algebraico por medio del cual se describen los elementos que integran la ecuación, este lenguaje es intuitivo ya que utiliza símbolos algebraicos convencionales y los nombres de los objetos que han sido declarados.

La actual implementación de **OPTEX Computer Lenguaje** maneja expresiones lineales. La sintaxis de la fórmula es la siguiente:

- La sumatoria de elementos matemáticos se asocia al símbolo **SUM**

- Los conjuntos sobre los que opera la sumatoria se deben expresar entre paréntesis a continuación, del símbolo **SUM**, así: inicialmente se indica el índice de la sumatoria el cual se se para del conjunto sobre el cual se desplaza el índice por medio de la barra vertical (**|**), que equivale al símbolo pertenece ($\in$), o sea que

SUM(c|CLD)

representa una sumatoria sobre el índice **c** para los valores contendidos en el conjunto **CLD**.

- Para límites numéricos en sumatorias se puede especificar los límites utilizando el operador igual (=) y separando los limites por una como (**,**), o sea que

SUM(t=t1,t2)

representa una sumatoria sobre el índice **t** para los valores comprendidos en el intervalo **t1** a **t2**.

- Si se requieren de múltiples sumatorias anidadas, simplemente se indican una a continuación, de la siguiente, por ejemplo:

SUM(c|CLD) SUM(k|KCD)

- Los términos sobre los cuales opera la sumatoria se escriben a continuación, de la ultima sumatoria, si la hay. Estos términos pueden corresponder a variables, a parámetros o a funciones de parámetros, dependiendo del caso. En la formula no es necesario escribir los subíndices de las variables o de los parámetros ya que estos se asocian a los definidos para los objetos referenciados. O sea que la siguiente expresión

$$\sum_{v \in VEH} \sum_{c \in CLI} \sum_{k \in KCL} CVIA_{c,k}\ VCL_{v,c,k}$$

se escribe como:

SUM(v|VEH) SUM(c|CLI) SUM(k|KCL) CVIA * VCL

Se debe notar que los índices del parámetro **CVIA(c,k)** y de la variable **VCL(v,c,k)** se pueden ignorar, ya que **OPTEX** los asume implícitos. El símbolo asterisco (*****) como operador de multiplicación es necesario parta diferenciar los multiplicandos.

- En el caso que existan operaciones sobre los índices, lo que aplica para índices asociados a períodos (**t**) o para índices con contenido numérico, las operaciones de los índices se deben incluir entre paréntesis para aquellas variables que se afectan.

- Diferentes sumandos se concatenan con el operador suma (**+**) o con el operador resta (**-**).

Ejemplos:

La ecuación

$$NF_{t,h,m} - NF_{t-1,h,m} - \sum_{c \in ARE(m)} HCE_{t,h,m,c} + \sum_{c \in ABE(m)} HEC_{t,h,m,c} + VE_{t,h,}$$

se escribe

NF - NF(t-1) - SUM(c|ARE) HCE + SUM(c|ABE) HEC + VE

En los límites de índices numéricos es posible utilizar operaciones referenciadas a parámetros a valores fijos. La ecuación

$$\sum_{t \in TEIN(p),TEFI(p)} SEZ_{t,p,z} + SDE_{p,z}$$

se escribe

SUM(t=TEIN,TEFI) SEZ+ SDE

La ecuación

$$\mathbf{OEW}_{t,e,w} - \mathbf{OMN}_{t\text{-}TES,e,w} + \mathbf{OMC}_{t\text{-}TES,e\ \forall MTC(e),w} + \mathbf{OMA}_{t,e,w}$$

se escribe

$$\mathbf{OEW - OMN\ (t\text{-}TES) + OMC(t\text{-}TES) + OMA}$$

4.2.9. PROBLEMAS

Un problema está asociado a un conjunto de restricciones que lo definen y a un conjunto de variables sobre las cuales tiene control. La declaración de un problema se realiza por medio del objeto **problem**.

Se requere definir los siguientes atributos

- **name:** código o nombre asociado al problema.
- **description:** descripción corta del problema.
- **index**: atributo vectorial que almacena los índices que determinan la variabilidad del problema
- **set**: atributo vectorial que almacena los conjuntos que determinan la existencia del problema
- **class:** se define el tipo de problema con base en sus características matemáticas, de acuerdo a la siguiente nomenclatura:
  - **PL** Programación Lineal
  - **FR** Flujo en Redes
  - **BI** Binaria
  - **BM** Programación Mixta-Binaria
  - **PE** Programación Entera
  - **EM** Programación Entera-Mixta
  - **PC** Programación Cuadrática
  - **PN** Programación No-lineal (futuro)
  - **CL** Programación Cuasi-Lineal (futuro)
- **rol:** se define el rol del problema de acuerdo con las funciones que cumple dentro de un esquema de partición descomposición. Se define el rol de acuerdo con:
  - **CO** Coordinador
  - **PR** Primario
  - **IN** Integrado

  Un problema primario corresponde al último nivel de jerarquía en un esquema multinivel.
- **coordinator:** define el problema que actúa como coordinador del problema que se esta definiendo.
- **constraint:** atributo vectorial que almacena las restricciones que definen el problema (endógenas).
- **variable:** atributo vectorial que almacena las variables sobre las cuales tiene control el problema (endógenas). Solo aplica para modelos que utilizan metodologías de gran escala. Cualquier variable que aparezca en las restricciones del problema y que no pertenezca a este conjunto, será tomado como variable definida exógenamente. Para modelos integrados, todas las variables que aparecen en las ecuaciones son consideradas como endógenas.

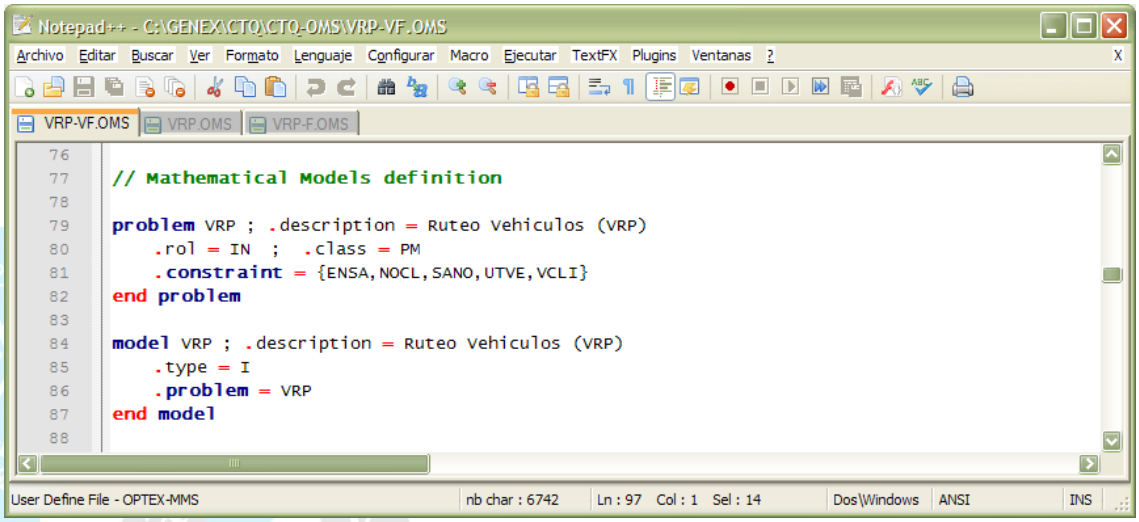
4.2.10 DE MODELOS

OPTEX soporta la definición de múltiples modelos en un mismo programa, para ello se utiliza el objeto **model**. Se declara un modelo con base en el conjunto de problemas que lo componen. El proceso de configuración de un modelo implica la definición de:

- **name:** nombre del modelo.
- **description:** descripción del modelo.

- **type:** Existen las siguientes opciones:
  - **E** Encadenado: Define una cadena de problemas encadenados en el tiempo.
  - **I** Integrado: Define un modelo integrado por un solo problema.
  - **P** Paralelo Estocástico: Define un conjunto de problemas que permiten analizar múltiples escenarios estudiados de forma paralela independientemente a cada escenario.
- **problem:** atributo vectorial que almacena los problemas que definen al modelo. El árbol jerárquico correspondiente será ensamblado automáticamente por **OPTEX** en el momento de generar la matriz asociada al problema.
- **model:** atributo vectorial que almacena los modelos encadenados secuencialmente en el tiempo que definen al modelo**,** en este caso se debe tener en cuenta el encadenamiento de horizontes de planificación asociados a cada problema.

La siguiente pantalla presenta ejemplos de declaración de **problem**s y **model**s.



4.2.11 HORIZONTES DE PLANIFICACIÓN

OPTEX permite definir modelos bajo un esquema flexible para la subdivisión del intervalo de planificación, y por lo tanto, no existe el concepto de horizonte de planificación asociado directamente a un modelo, sino que él mismo se integra al modelo en el momento de solucionarlo. Un horizonte de planificación se considera subdividido en un número de períodos definidos de acuerdo con el propósito del uso que vaya a dar al modelo. Para ello se utiliza el objeto **time-horizon**.

La configuración de un horizonte de planificación implica la definición de:

- **name**: código dado al horizonte de planificación
- **descripción:** descripción del horizonte.
- **type:** Tipo de horizonte. Puede ser:
  - **G** Genérico: implica que todos los períodos del horizonte son similares, con igual longitud de acuerdo a la unidad de tiempo especificada
  - **D** Detallado: implica que cada período del horizonte se especifica detalladamente en la tabla secundaria
  - **E** Encadenado: Indica que el horizonte está compuesto por un encadenamiento de horizontes. Este tipo de horizonte se requiere para la implementación de modelos encadenados.
- **type-period**: Aplica cuando el horizonte es genérico (**G**). Indica el tipo de período que se utiliza para manejar el horizonte de planificación, los cuales son todos del mismo tipo. Se consideran los siguientes tipos:
  - **A** Años
  - **M** Meses

D Días

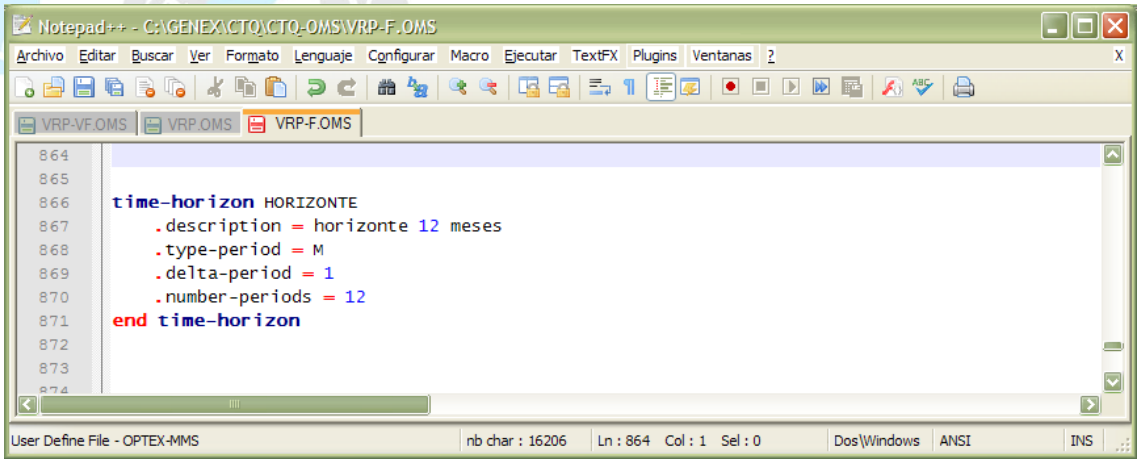
- **delta-period:** Aplica para horizontes uniformes e indica la longitud del período de acuerdo con la unidad de tiempo básica.
- **number-periods:** número de períodos que integran el horizonte de planificación.
- **periods:** atributo vectorial que contienen la secuencia de períodos que integran el horizonte de planificación cuando el horizonte es detallado (**D**)., corresponde a una serie de objetos del tipo **period**.
- **time-horizon:** atributo vectorial que contienen la secuencia de que integran un horizonte de planificación integrado por múltiples horizontes encadenados secuencialmente en el tiempo. Aplica para horizontes asociados a modelos encadenados.

Cuando el horizonte es detallado (**D**) se deben definir los diferentes subperíodos que integran el horizonte. De acuerdo con la unidad básica asumida para el manejo del tiempo se deben definir el número de unidades de tiempo para cada período. El horizonte no trabaja con fechas absolutas. **OPTEX** numera los períodos secuencialmente y especifica la cantidad de unidades de tiempo de cada subperíodo. De acuerdo con la fecha inicial que se defina para el modelo se determina la fecha cronológica asignada a cada período del horizonte de planificación, la que se define a nivel de la declaración del escenario.

Para cada el caso de horizontes detallados para cada **period** se debe declarar:

- **name:**
- **number-periods**
- **type-period:** indica el tipo de unidad de tiempo utilizada para el número de unidades de tiempo del subperíodo. El tipo de unidad de tiempo debe ser de orden superior o igual al tipo de unidad del horizonte. Se consideran los siguientes tipos:
  A Años
  M Meses
  D Días

La siguiente pantalla presenta ejemplos de declaración de **time-horizon**s.



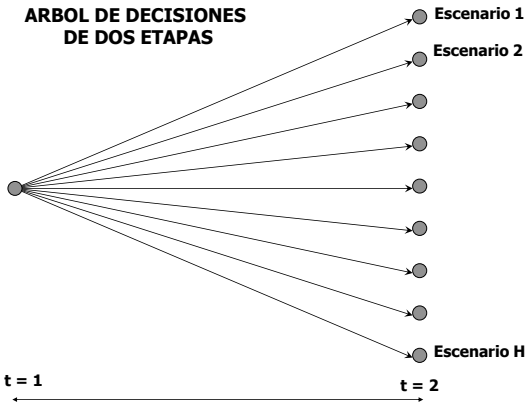
4.2.12. ÁRBOLES DE DECISIÓN

Los modelos de optimización estocástica no-anticipativa requieren de la definición de un árbol que represente la dinámica y la variabilidad del proceso estocástico asociado al modelo y lo relacione con el proceso de toma de decisiones que se está analizando. A continuación, se presenta de manera resumida el proceso matemático implícito.

Consideremos un proceso decisorio de dos etapas. En la primera etapa, correspondiente al momento inicial, el comienzo del período **1** (aquí y ahora), se toma la decisión **y**, posteriormente a la toma de la decisión, durante el período **1**, ocurre una realización del proceso estocástico **w**, y posteriormente a que dicho proceso ocurre se debe tomar una nueva decisión **x**, al comienzo

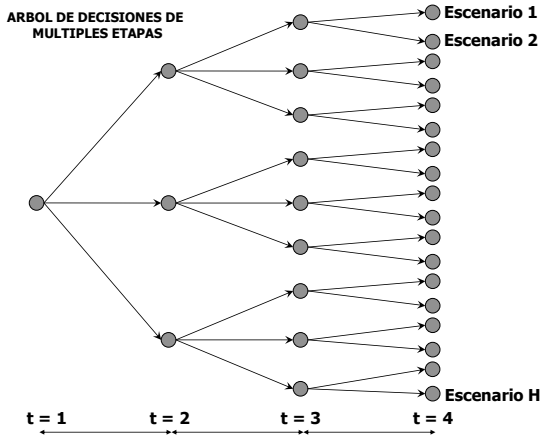
del período **2**. La decisión **y** es única e independiente del proceso estocásticos en tanto que la decisión **x** es dependiente, a posteriori, del proceso estocástico, y se simulan tantas decisiones **x** como escenarios aleatorios. EL carácter único de **y** hace que el proceso sea no-anticipativo, esto quiere decir que no se sabe que va a pasar, pero si se sabe que puede pasar, ya que se asume se conoce la probabilidad de ocurrencia de cada escenario, y los valores de los parámetros aleatorios en cada nodo.

El anterior proceso puede representarse por un árbol de decisiones de dos etapas



Consideremos ahora un proceso decisorio de múltiples etapas, el cual puede interpretarse como un encadenamiento de múltiples árboles de dos etapas. Al comienzo de la primera etapa, identificada como **t=1**, se toma la decisión $\mathbf{x_1}$, posteriormente, durante el período asociada a la duración de la etapa **1**, ocurre una realización del proceso estocástico $\mathbf{w_1}$; posteriormente a que dicho proceso ocurre se debe tomar una nueva decisión $\mathbf{x_2}$ y esperar a la realización $\mathbf{w_2}$ para posteriormente tomar la decisión $\mathbf{x_3}$ y así sucesivamente hasta llegar a la última etapa (**t=T**), último período del horizonte de planificación.

El anterior proceso puede representarse por un árbol de decisiones de múltiples etapas. En el árbol compuesto por ramas y nodos, cada rama corresponde a un escenario **h** y cada nodo **n** a un estado de toma de decisiones.

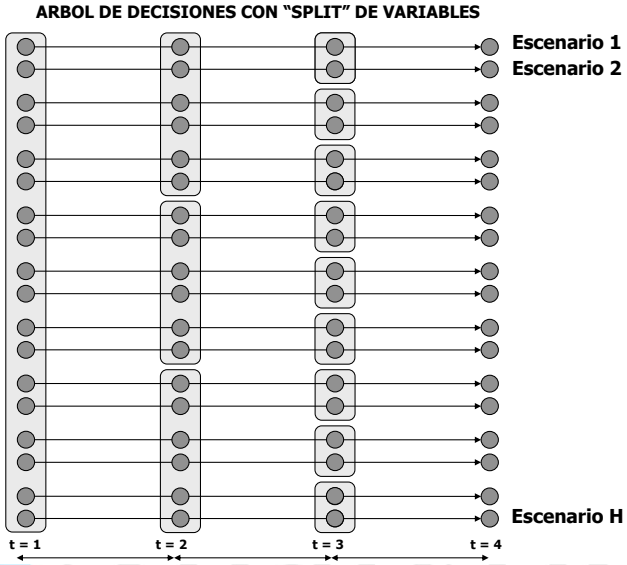


Para describir la topología del proceso decisorio se requiere definir los nodos, los períodos asociados a cada nodo (puede ser más de uno) y las ramas (condiciones aleatorias) que pasan por el nodo. Adicionalmente se deben conocer en cada nodo el valor de los parámetros que representan el proceso estocástico y la probabilidad de arribar a dicho nodo, dado el nodo que lo antecede.

Para garantizar la condición de no anticipatividad, es necesario definir las variables de control

sobre las que se aplicará la restricción que garantiza que la decisión tomada en cada nodo es igual para todas las variables no-antipativas para todas las ramas que pasan por el nodo. A continuación, se explica con mayor detalle matemático esta restricción.

Para comprender las restricciones de no-anticipatividad, se puede considerar que se dispone de tantas ramas paralelas como escenarios aleatorios y que en cada nodo se impone una restricción de igualdad que garantiza que todas las decisiones que se toman en un nodo, para las ramas que pasan por él, son iguales en su valor numérico.



Para visualizarlo matemáticamente, Definamos el vector de variables de decisión $\mathbf{x_{t,h}}$ como la decisión que se toma en la etapa **t** si ocurre el escenario **h** (transición por la rama **h** del árbol). Para garantizar la no-anticipatividad se requiere introducir una restricción que iguale las decisiones que se toman en el nodo **n**, lo que implica formular la restricción de no-anticipatividad como

$$\mathbf{z_n = x_{t,h}}$$
$$\mathbf{\forall\ n \in N\ \forall\ t \in e(n)\ \forall\ h \in B(n)}$$

donde **e(n)** representa al conjunto de ramas (escenarios) asociados al nodo **n**, **B(n)** al conjunto de ramas (escenarios) asociados al nodo **n** ("scenario bundle") y $\mathbf{z_n}$ una variable auxiliar que se conoce como variables de estado de la información ("information state vectors", Higle et al. 2002).

Para representar lo anterior, en **OPTEX** se utilizan los objetos **decision-tree** y **node.** Para declarar un **decisión-tree** se debe definir los siguientes atributos:

- **name:** nombre asociado al árbol
- **description:** descripción corta del árbol
- **node:** atributo vectorial que almacena los nodos que integran el árbol. Corresponden a objetos del tipo **node**.
- **variable:** atributo vectorial que almacena las variables no-anticipativas que se deben considerar en cada nodo.

Para declarar un **node** se debe definir:

- **name:** código o nombre asociado al nodo
- **node-previous:** nodo anterior al nodo en referencia
- **initial-stage:** período inicial para las decisiones asociadas al nodo
- **final-stage:** período final para las decisiones asociadas al nodo

- **probability:** probabilidad de condicionada de transición entre el nodo anterior y el nodo en regencia. La suma de todas las probabilidades que salen de un nodo deben ser igual a **1**.
- **branch:** atributo vectorial que almacena las ramas (escenarios, condiciones aleatorias) que pasan por el nodo.

4.2.13. ESCENARIOS

Para definir el caso que se desea analizar se requiere declarar un escenario para ello se debe utilizar el objeto **scenario**. Para ello el usuario debe determinar:

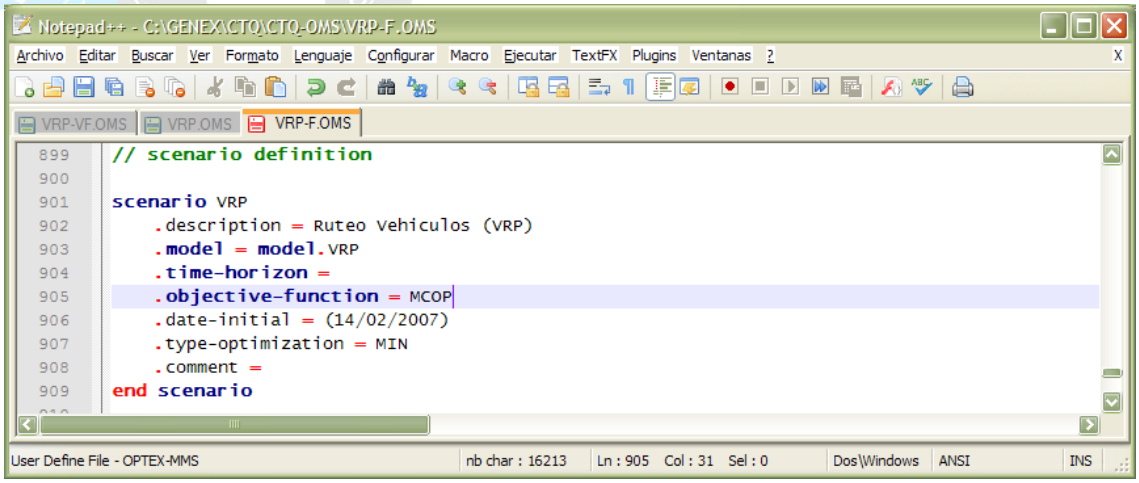
- Modelo a estudiar
- Función objetivo a optimizar
- Tipo de optimización
- Horizonte de planificación
- La fecha inicial de la planificación (si el modelo es dinámico)

Todas las anteriores especificaciones, con excepción de la definición de la fecha inicial y la función objetivo, deben realizarse a nivel de la familia de escenarios asociada al modelo.

Para declara un escenario se debe definir los siguientes atributos:

- **name:** nombre dado al escenario.
- **description:** descripción de la familia de escenarios.
- **model:** modelo asociado a la familia de escenarios
- **time-horizon:** nombre del horizonte de planificación
- **objetive-function:** corresponde al código de la función objetivo que se quiere utilizar como medida de optimalidad
- **type-optimization:** maximizar, minimizar, minimax o maximin.
- **date-initial**: para aquellos modelos en que la fecha inicial de modelaje es importante se debe definir el momento para el cual se considera el tiempo igual a cero. (**t=0**)
- **decision-tree:** nombre del árbol de decisiones, aplica para modelos de optimización estocástica no-anticipativa

La siguiente pantalla presenta ejemplos de declaración de **scenario**s.



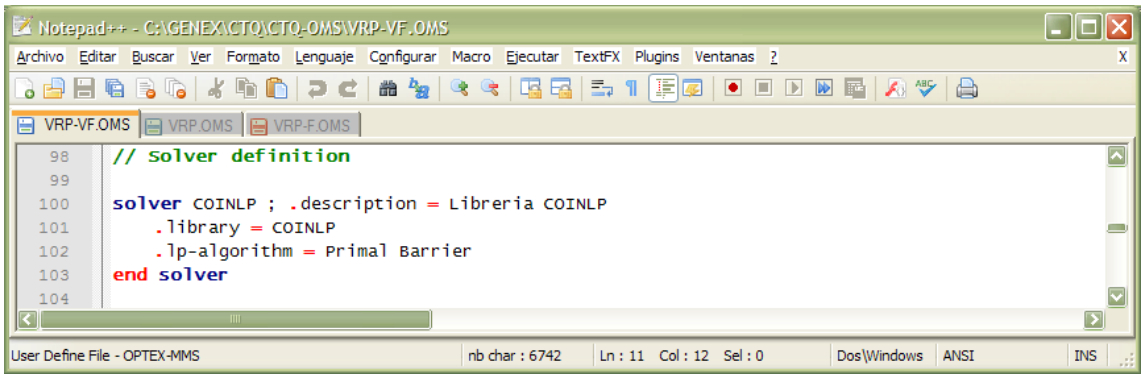
4.2.13. SOLVER

EL usuario puede declarar las características del "solver" que va a utilizar para resolver el problema, lo que se realiza por medio del objeto **solver**. Los atributos que se requieren son:

- **name:** nombre dado al solver
- **library**: librería
- **lp-algorithm**: algoritmo a utilizar para resolver el(los) problema(s) de programación lineal
- **mip-option**: opciones del proceso de programación entera

- **nlp-algorithm**: algoritmo a utilizar para resolver el(los) problema(s) de programación no-lineal

La siguiente pantalla presenta ejemplos de declaración de **solver**s.

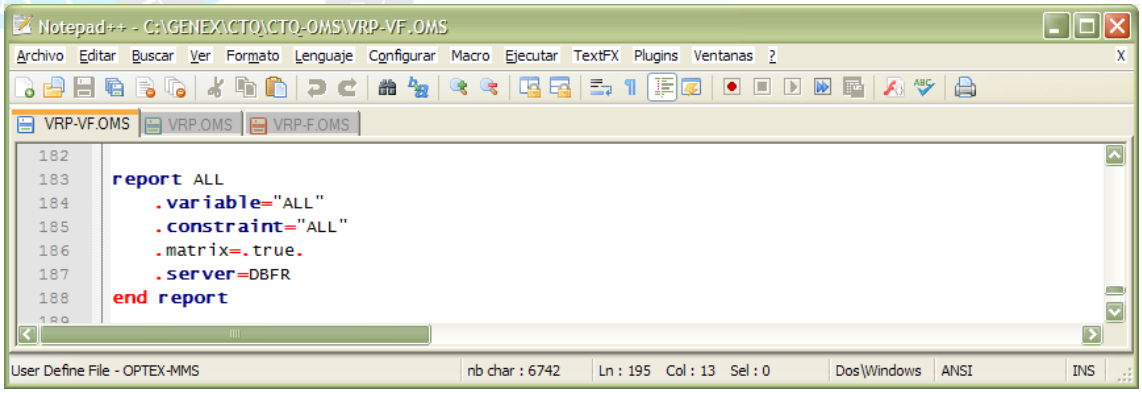


4.2.13. REPORTES

El usuario puede declarar las características del reporte que desea, para ello dispone del objeto **report**. Los atributos que se requieren son:

- **name:** nombre dado al reporte
- **variable:** atributo vectorial que almacena las variables que se incluirán en el reporte. Si se especifica "ALL" se incluyen todas.
- **constraint:** atributo vectorial que almacena las restricciones que se incluirán en el reporte. Si se especifica "ALL" se incluyen todas.
- **matriz:** indica si se desea el reporte de las estructuras matriciales.
- **mps:** indica si se desea el reporte en formato **MPS**.
- **server:** indica el servidor que se utilizará para ubicar las tablas de resultados.

La siguiente pantalla presenta ejemplos de declaración de **report**s.



4.1. DATA OBJECTS

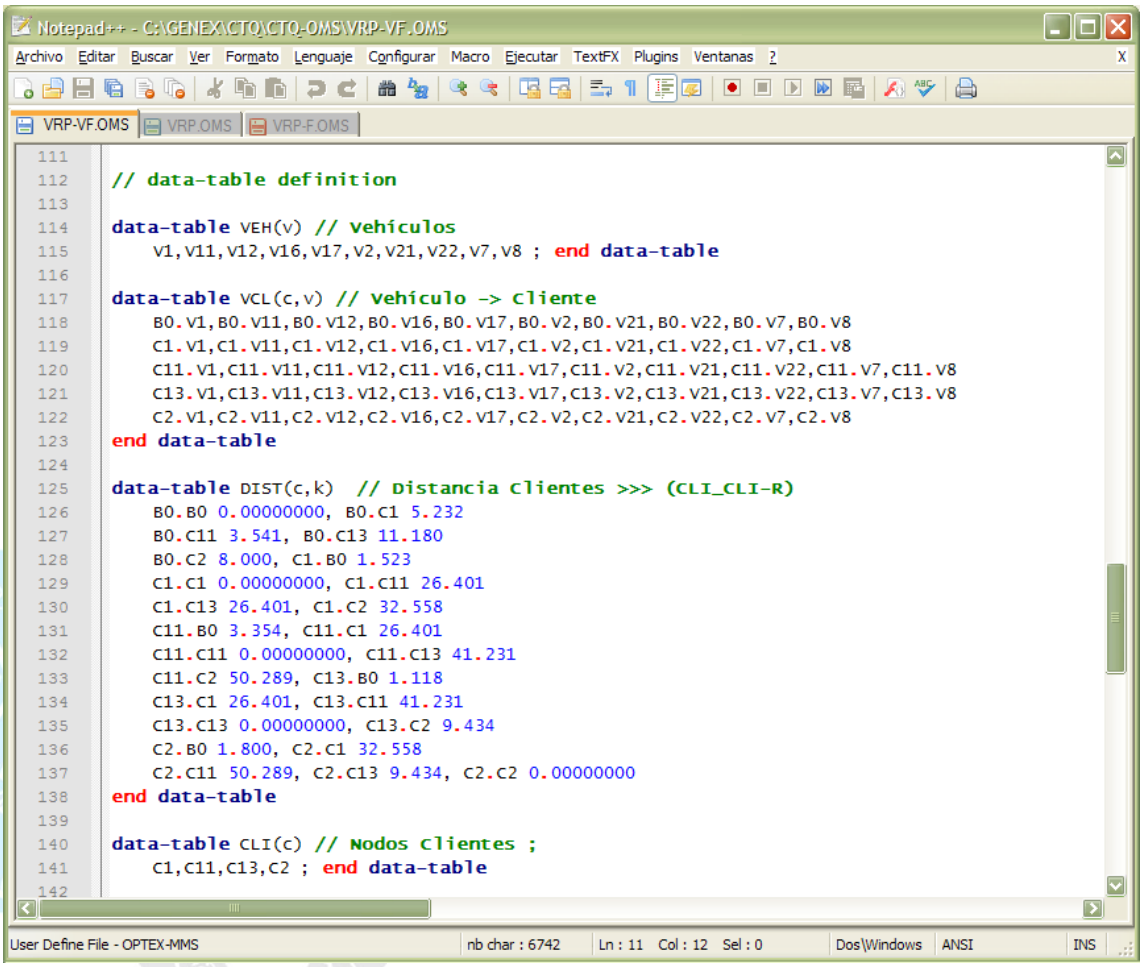
El usuario puede dentro del programa declarar datos para configurar conjuntos y establecer valores de los parámetros. Para ello debe utilizar el objeto **data-table**.

Los atributos de **data-table** son:

- **name:** nombre dado a la tabla de datos
- **index**: atributo vectorial que almacena los índices que determinan la variabilidad de la tabla de datos
- **field**: atributo vectorial que almacena los campos numéricos que contienen la tabla. Se requiere cuando una tabla de datos almacena más de un parámetro.

Los datos de la tabla se deben relacionar con los índices, separándolos por medio de puntos (**.**). Para introducir varios grupos de datos en una línea estos deben estar separados por comas (**,**).

La siguiente pantalla presenta ejemplos de declaración de **data-table**s.



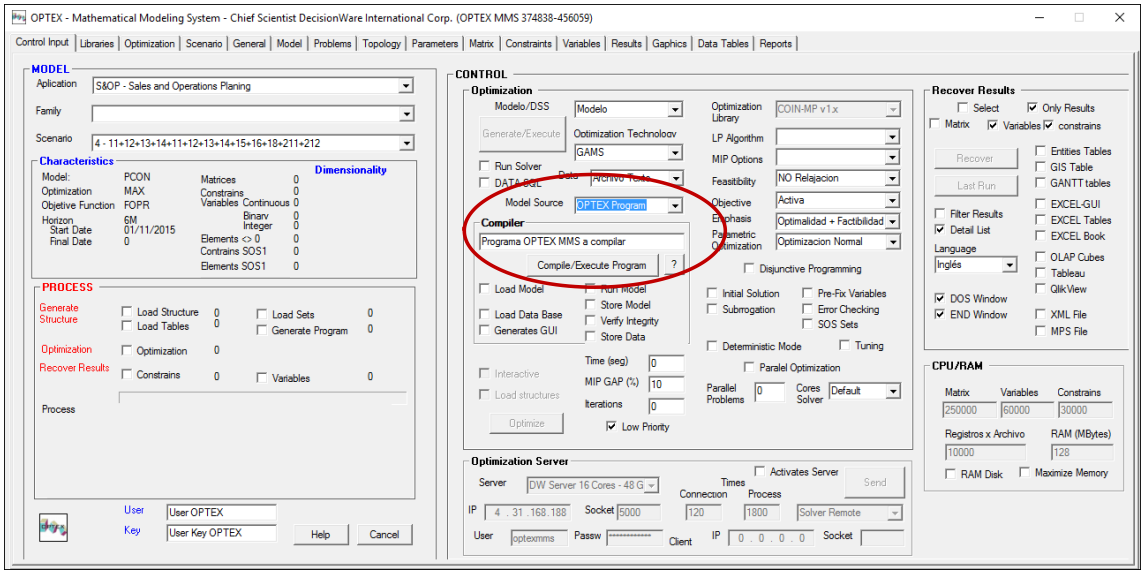
5. COMANDOS

A la fecha se han implementado los siguientes comandos

COMANDO	DESCRIPCIÓN
solve(scenario,solver)	Resuelve el problema definido para un **scenario** con un **solver** especificado.
recover(report)	Recupera los resultados de acuerdo con un **report** especificado.

6. SOLUCIÓN DE PROBLEMAS

Todos los aspectos relacionados con la solución de los modelos de optimización se manejan por medio del compilador el cual está conectado a la ventana de control de **OPTEX**. Cuando se compila un programa se tiene acceso parcial a los controles y pestañas de la ventana de control de **OPTEX-EXE**. El proceso de compilación se puede activar manualmente desde la ventana de control de **OPTEX** o automáticamente desde **NOTEPAD++.**


OPTEX - Mathematical Modeling System - Chief Scientist DecisionWare International Corp. (OPTEX MMS 374838-456059)
Control Input | Libraries | Optimization | Scenario | General | Model | Problems | Topology | Parameters | Matrix | Constraints | Variables | Results | Gaphics | Data Tables | Reports
MODEL
Aplication
S&OP - Sales and Operations Planing
Family
Scenario
4 - 11+12+13+14+11+12+13+14+15+16+18+211+212
Characteristics
Model: PCON
Optimization MAX
Objetive Function FOPR
Horizon 6M
Start Date 01/11/2015
Final Date 0
Matrices 0
Constrains 0
Variables Continuous 0
Binary 0
Integer 0
Elements <> 0 0
Contrains SOS1 0
Elements SOS1 0
Dimensionality
PROCESS
Generate Structure
Load Structure 0
Load Tables 0
Load Sets 0
Generate Program 0
Optimization
Optimization 0
Recover Results
Constrains 0
Variables 0
Process
User
User OPTEX
Key
User Key OPTEX
Help
Cancel
CONTROL
Optimization
Modelo/DSS
Modelo
Generate/Execute
Optimization Technology
GAMS
Run Solver
DATA-SQL
Data
Archivo Texto
Model Source
OPTEX Program
Compiler
Programa OPTEX MMS a compilar
Compile/Execute Program
?
Load Model
Run Model
Load Data Base
Store Model
Generates GUI
Verify Integrity
Store Data
Interactive
Load structures
Optimize
Time (seg) 0
MIP GAP (%) 10
Iterations 0
Low Priority
Optimization Library
COIN-MP v1.x
LP Algorithm
MIP Options
Feasibility
NO Relajacion
Objective
Activa
Emphasis
Optimalidad + Factibilidad
Parametric Optimization
Optimizacion Normal
Disjunctive Programming
Initial Solution
Pre-Fix Variables
Subrogation
Error Checking
SOS Sets
Deterministic Mode
Tuning
Paralel Optimization
Parallel Problems 0
Cores Solver
Default
Optimization Server
Server
DW Server 16 Cores - 48 G
Activates Server
Send
Times
Connection
Process
IP 4 . 31 . 168 . 188
Socket 5000
120
1800
Solver Remote
User optexmms
Passw
Client
IP 0 . 0 . 0 . 0
Socket
Recover Results
Select
Only Results
Matrix
Variables
constrains
Recover
Last Run
Filter Results
Detail List
Language
Inglés
DOS Window
END Window
Entities Tables
GIS Table
GANTT tables
EXCEL-GUI
EXCEL Tables
EXCEL Book
OLAP Cubes
Tableau
QlikView
XML File
MPS File
CPU/RAM
Matrix
Variables
Constrains
250000
60000
30000
Registros x Archivo
RAM (MBytes)
10000
128
RAM Disk
Maximize Memory