ÑERANDU: Lenguaje de Soporte de Decisiones en la Industria

César Cabello
Sergio Aranda
Mercedes Rivaldi

Laboratorio de Electrónica Digital
Facultad de Ciencias y Tecnología
Universidad Católica "Ntra. Sra. de la Asunción"
Casilla de correo: 1718 Fax: (595) 21 - 31 05 87
E-mail: ccabello@ledip.py, saranda@venus.py
Asunción Paraguay

Resumen

Este trabajo presenta el lenguaje inmerso en un sistema integrado de Producción-Finanzas (*ARANDU*), desarrollado con el objeto de servir como soporte a las decisiones desde la definición de los productos hasta el planeamiento de producción, compras y el manejo financiero; que surgen en la administración de industrias sometidas a variabilidad en la demanda de productos, oferta de materias primas y disponibilidad financiera.

1 Introducción

En los países de producción fundamentalmente agropecuaria, la economía está fuertemente influenciada por los ciclos de la naturaleza; esto impone la necesidad de planificar las actividades productivas, y en particular la industrial, tomando en cuenta las complejas interacciones que resultan de las mencionadas variaciones. A este fin fue desarrollado *Arandú*, nombre del sistema original de Producción - Finanzas a cuya extensión se orienta *Ñerandú*. El objetivo es crear un ambiente *Arandú*, que manteniendo la interactividad y sencillez, pueda ser objeto de programación por parte del usuario.

2 Ambiente de aplicación

Para una adecuada comprersión de *Ñerandú*, es necesaria una descripción, al menos sucinta, de *Arandú* (en su versión original).

Tomando en cuenta las funciones de la labor industrial en que la informática puede brindar mayor apoyo, planificación y control, el ambiente *Arandú* se dividió en los siguiente módulos:

a) Planificación
b) Control de Producción
c) Control Financiero

La lógica de trabajo reside en que el usuario, sobre la base de la información disponible: estructura del producto, procesos realizados sobre el producto y recursos utilizados, estimaciones de ventas por periodos, estimaciones de costo de compra, estimaciones de costo de almacenamiento, estimaciones de costos fijos, estimaciones de disponibilidad de horas hombres y máquinas, y estimaciones de costos de horas hombres y máquinas; elabora un plan o planes alternativos en cuyo cumplimiento se generan productos y consumen recursos. El plan puede ser opcionalmente obtenido en base al soporte de la máquina que en todo su ambiente provee algoritmos para la toma de decisiones fundados en programación matemática, programación dinámica y heurísticas ad hoc, elegibles según el criterio del usuario.

El planeamiento de la producción, de las compras y de las finanzas pueden ser efectuados en un horizonte de *n* periodos, bajo diferentes criterios de optimización. Una vez aceptado un plan es objeto de un *lanzamiento*; a partir de este momento el cumplimiento del plan es objeto de control en los niveles productivos y financieros.

Este esquema de trabajo se ha demostrado eficaz, como lo prueba la experiencia en las empresas que utilizan *Arandú*, pero el análisis de alternativas y planes multiples puede facilitarse significativamente mediante la disponibilidad de recursos informáticos que permitan un control automático del flujo de dichos análisis, sin necesidad de la intervención directa ante cada resultado, de manera similar a como operan las macros en las planillas electrónicas. De esta necesidad nace ÑERANDU.

3 Descripción del Lenguaje

Los comandos definidos en Ñerandú permiten al usuario procesar la información contenida en las tablas definidas en el sistema. Los comandos se agrupan en tres conjuntos :

a) *Algebra relacional*, permiten el manejo eficiente de bases de datos relacionales.
Ejemplos : **CREAR_TABLA**, permite al usuario crear una tabla, ya sea para ser usada como introducción de datos, así como para obtener resultados de salida.

BORRAR_TABLA, permite al usuario eliminar una tabla previamente creada.

b) *Estructuras de control de flujo*, permiten al usuario avanzado crear programas que interactúan con las bases de datos.
Ejemplos: **SI-FINSI**, permite un salto condicional en la ejecución de un grupo de sentencias .

PARA-FINPARA, permite la iteración de un conjunto de sentencias.

c) *Operaciones matemáticas complejas*, incluyen la posibilidad de plantear y resolver un problema de programación lineal.

Ejemplos : **PRG_LINEAL**, soluciona un problema lineal.

CAMBIO_DIS, dada la solución de un problema lineal, permite resolver uno semejante partiendo de la solución anterior, pero con cambios en las restricciones de *disponibilidades*. Puede de esta manera evitarse el replanteo completo y el cálculo desde el comienzo de la solución,

CAMBIO_COST, semejante a la función anterior, pero los cambios permitidos son sobre los costos de las variables de decisión del problema lineal.

El usuario puede utilizar las siguientes macro funciones, que permiten aprovechar toda las prestaciones de *Arandú* :

A - **OPTIM_MEZ**(SemiElaborado, TablaResultado), que permite optimizar la composición de Semi-Elaborados.

B - **ALTER_PROD**(Alternativa, TablaResultado), genera los resultados correspondientes a una alternativa de producción.

C - **PLAN_PROD**(Estimación, Perspectiva, TablaResultado), genera un Plan de Producción, dada una estima y un perspectiva.

D - **PLAN_FINAN**(Estimación, Perspectiva, TablaResultado), genera un Plan Finaciero dada una estima y una perspectiva.

Por razones de consistencia de datos, los comandos definidos no pueden en ningún caso modificar la estructura de las tablas definidas en Arandú.

La descripción formal del lenguaje se presenta siguiendo la notación BNF para definición de sintáxis. Por convención, los términos en **negritas** representan los *símbolos terminales* del lenguaje.

Expresiones Booleanas

ccond -> ccond **.or.** cterm | cterm

cterm -> cterm **.and.** cfactor | cfactor

cfactor -> **.not.** cfactor | (ccond) | **VERDADERO** | **FALSO**

Operadores Matemáticos

op1 -> op1 + op2 | op1 - op2 | op2

op2 -> op2 * factor | op2 / factor | factor

factor -> factor ^ numero

numero -> - num

Números

num -> num digito | num . digito | digito

digito -> | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

Variables

var -> var alfanum | alfanum

alfanum -> a | b | c | d | . . . | z

Estructuras de Control

A - Estructura PARA

cicloPara -> PARA(inicializaciones ; ccond ; increment) sentencias FINPARA

inicializaciones -> inicializaciones, varInit

increment -> increment , varInit | varInit

varInit -> varInit, var:= op1 | var:= op1

B - Estructura SI

estrucSi -> SI ccond sentencia [SINO sentencia] FINSI

Funciones

stmt -> stmt agregT | stmt borrarT | stmt crearT | stmt nFilT | stmt LinealT | stmt unionT | stmt cicloPara | var:= stmt | stmt estrucSi | stmt cambioDisT | stmt cambioCostT

agregT -> AGREGAR(var , varInit) ;

borrarT -> BORRAR_TABLA(listVar); | BORRAR_TABLA(TODAS);

crearT -> CREAR_TABLA var CON listVar ;

nFilT -> NFILAS(var, ccond) ;

LinealT -> PRGLINEAL([MAX | MIN], var, var, var);

unionT -> UNION(var, var, ccond, [var,...], var);

cambioDisT -> CAMBIO_DIS(var, var, var);

cambioCostT -> CAMBIO_COST(var, var, var);

listVar -> listVar var | var

Inicio

Simbolo de Inicio -> sentencia

sentencia -> sentencia ; stmt | stmt

Símbolos y Convenciones

( ) Indica la lista de argumentos de una función.

[ ] Indica un item o lista opcional.

< > Indica un item que debe ser introducido por el usuario.

, Separador para lista de items.

; Fin de una sentencia.

Operadores y relaciones lógicas

. Indicador de campo.

[ ] Indicador número de fila

.+. Suma de todos los campos de dos tablas. (Alcance de conjunto)

.

.-. Diferencia entre todos los campos de dos tablas.(Alcance de conjunto) .

.*. Producto entre todos los campos de dos tablas.(Alcance de conjunto) .

./. División entre todos los campos de dos tablas.(Alcance de conjunto) .

+ Suma entre dos variables o campos.(Alcance simple) .

- Diferencia entre dos variables o campos. (Alcance simple).

* Producto entre dos variables o campos. (Alcance simple).

/ División entre dos variables o productos. (Alcance simple).

:= Asignación. (Alcance de conjunto o simple, dependiendo de las variables).

== Comparación de igualdad para conjuntos y variables simples.

<= Comparación menor o igual que, para conjuntos y variables simples.

>= Comparación mayor o igual que, para conjuntos y variables ismples.

< Comparación estrictamente menor que, para conjunto y variables simples.

> Comparación estrictamente mayor que, para conjuntos y variables simples.

<> Desigualdad, para conjuntos y variables simples.

.AND. Operación '*Y*' lógica.

.OR. OperaciÛn '*O*" lógica.

.NOT. Negación lógica.

A título ilustrativo se describe en detalle la función **PRGLINEAL** .

PRGLINEAL
Resuelve un programa lineal dado.

Sintáxis

PRGLINEAL(<maxMin>, <funcionObjTabla>, <restricListTablas>, <resultTabla>);

Argumentos

<maxMin> establece el tipo de optimización que se requiere. Así para maximizar un programa lineal se utiliza la palabra clave **MAX**, mientras que se utiliza **MIN** en el caso de minimizar.

<funcionObjTabla> es la tabla o relación en la que se encuentra definida la *función objetivo* del programa lineal dado.

<restricListTablas> es una lista de tablas. Las mismas definen los *requerimientos de recursos* por cada variable de decisión, y *restricciones de disponibilidades* del programa lineal. Cada elemento del vector es una terna formada por, una tabla de requerimientos de recursos, el símbolo <= ó =>, y la tabla de disponibilidades de recursos. Los símbolos de menor que y mayor que indican la relación existente entre las dos tablas anteriores.

<resultTabla> es el nombre de la tabla donde se guarda la *solución del programa lineal* planteado. Esta tabla se crea automáticamente si no existe siguiendo el formato definido en la tabla correspondiente a la función objetivo.

En caso de encontrarse ya definida la tabla, sólo se modifica su formato, si fuere necesario, y su contenido.

Advertencia : Toda información que previamente contenga la tabla solución se pierde.

Descripción

La función **PRGLINEAL** resuelve un problema de programación lineal, planteado en los términos de sus parámetros.

Ejemplo

/ Un fabricante de dos tipos de cerveza, desea saber cuales deben ser los niveles de producción semanal de cerveza clara y oscura que hagan máximo los ingreso por semana. Se sabe que 1000 lts de cerveza clara originan un ingreso semanal de $ 5000, utilizan 3 obreros para ser producidos y su producción cuesta $500. Se sabe también que 1000 lts de cerveza oscura originan un ingreso semanal de $3000, se precisa de 5 obreros y su producción cuesta $200. Se sabe además que la planta tiene un total de 15 obreros y no puede gastar más de $1000 semanales. */*

```
CREAR_TABLA     ingresos_semanales
CON  producto,  caracteres,     15,
     costos,          numérico, 10;
CREAR_TABLA     obreros_por_producto
CON  producto,  caracteres,     15,
     nro_obreros,     numérico, 10;
CREAR_TABLA     disponibilidad_de_obreros
CON  producto,  caracteres,     15,
     tot_obreros,     numérico, 10;
CREAR_TABLA     costos_por_producto
CON  producto,  caracteres,     15,
     costo_unitario, numérico, 10;
CREAR_TABLA     tot_gastos
CON  producto,  caracteres,     15,
     tot_costos,      numérico, 10
```

```
/* Se cargan los datos ya enunciados anteriormente */
AGREGAR(ingresos_semanales,
        "cerveza clara" : 5000,
        "cerveza oscura":1000);
AGREGAR(obreros_por_producto,
        "cerveza clara" : 3,
        "cerveza oscura":5);
. . .
. . .
/* Resolver el programa lineal formulado, guardando el resultado en
la nueva tabla cantidades_optimas */
PRGLINEAL(MAX , ingresos_semanales,
                obreros_por_producto,<= , disponibilidad_de_obreros,
                costos_por_productos, <= , tot_gastos,
                cantidades_optimas ) ;
```

4 Conclusión

Las facilidades que ofrece un lenguaje de soporte de decisiones son significativas respecto a los sistemas de acceso fijos, sin embargo aún queda camino por recorrer en la búsqueda de mayores facilidades para consultas y optimizaciones en sistemas de soporte de decisiones para la industria. Los avances en el procesamiento del lenguaje natural son promisorios para este campo.

Bibliografía

- Aho, Alfred - Sethi, Ravi - Ullman, Jeffrey
  Compilers: Principles, Techniques, and Tools.
  Addison-Wesley Publishing Company.
  1986

- Cabello, César
  Arandú: Soporte de Decisiones en la Industria
  Laboratorio de Electrónica Digital, Universidad Católica
  Asunción - Paraguay
  1991

Elmasri, Ramez - Navathe, Shamkant.
Fundamentals of Database Systems.
The Benjamin/Cummings Publishing Company, Inc.
California - USA
1989

Lee, Ronald M.
Epistemological aspects of Knowledge-based Decision Support Systems.
Processes and Tools para Decision Support
Norht-Holland Publishing Company
1983

Orchard Willian Hays
Advanced Linear - Programming Computing Techniques
McGraw Hill
1968

Prawda, Juan
Métodos y Modelos de Investigación de Operaciones Vol I.
Editorial Limusa.
Ciudad de México - MéxicoNorht-Holland Publishing Company
1989