VII.5 SISTEMA EVOLUTIVO GENERADOR DE ESQUEMAS LÓGICOS DE BASES DE DATOS

Elsa Berruecos Rodríguez *

Resumen

La construcción del Sistemas Evolutivo Generador de Esquemas Lógicos de Bases de Datos se basa en el método de "Programación Dirigida por Sintaxis", el cual, nos permite construir sistemas con la capacidad de inducir reglas generales a partir de ejemplos específicos para posteriormente aplicarlas a la solución de problemas particulares. Tal es el caso del sistema que se presenta en este trabajo, el cual a partir de ejemplos significativos del lenguaje del usuario (Lenguaje Natural), induce su gramática generativa para posteriormente poder interpretar oraciones que describan un ambiente y generar a partir de dicha descripción el esquema lógico deseado.

Palabras clave: Sistemas Evolutivos, Base de Datos, Lenguaje Natural, Programación Dirigida por Sintaxis, Gramáticas, Gramática con Atributos, Semántica, Forma Canónica, Recursividad

INTRODUCCIÓN

A raíz de la aceptación que han tenido los Sistemas Manejadores de Base de Datos (SMBD o DBMS por sus siglas en inglés), las compañías que desarrollan estos sistemas invierten cada año gran cantidad de recursos humanos y materiales para perfeccionar su funcionamiento, sin embargo, poco se ha hecho para facilitarle al usuario de estas herramientas la difícil tarea de diseñar las bases de datos que se implantarán en esos sistemas.

---

* Elsa Berruecos Rodríguez realizó este trabajo en su seminario de titulación en la Unidad Profesional Interdisciplinaria de Ingeniería y Ciencias Sociales y Administrativas (UPIICSA) del Instituto Politécnico Nacional (IPN) el 5 septiembre de 1988.

Existen definidas actualmente varias metodologías para elaborar el diseño lógico y físico de una base de datos, sin embargo, a pesar de hacer uso de ellas, el tiempo que se debe dedicar a esta tarea continua siendo elevado y no puede llevarse a cabo sin tener un conocimiento profundo sobre diseño de base de datos.

La herramienta que se propone para ayudar a solucionar este problema es un Sistema Evolutivo que genera o actualiza esquemas lógicos de bases de datos a partir de la descripción en lenguaje natural que el usuario haga del ambiente que desea manejar en dicha base de datos.

La construcción de este sistema se basa en uno de los métodos más innovadores que en materia de desarrollo de sistemas se haya conceptualizado, la "Programación Dirigida por Sintaxis", este método permite construir sistemas que tienen la capacidad de inducir reglas generales a partir de ejemplos específicos para posteriormente aplicarlas a la solución de problemas particulares. Tal es el caso del Sistema Evolutivo Generador de Esquemas Lógicos de Bases de Datos, el cual a partir de ejemplos significativos del lenguaje del usuario induce su gramática generativa para posteriormente interpretar oraciones que describen el ambiente del que se obtiene el esquema de base de datos requerido.

El objetivo de este sistema es generar el esquema lógico de una Base de Datos que refleje el ambiente que el usuario le describa mediante oraciones en lenguaje natural. Pretende cubrir las necesidades de aquellas organizaciones que ya tienen instalado algún Sistema Manejador de Bases de Datos (SMBD, DBMS por sus siglas en inglés) y que ahora se enfrentan al problema de tener que capacitar a sus analistas para diseñar bases de datos.

La idea es que este sistema pueda ser utilizado por personas que aún sin tener un conocimiento profundo sobre Bases de Datos puedan hacer uso de ellas para aprovechar las facilidades que esta tecnología les brinda.

El sistema también se plantea el objetivo de ayudar a los diseñadores de bases de datos a definir y modificar esquemas fácilmente y sin que esto les consuma demasiado tiempo en el análisis.

Por el momento se tiene considerado que el sistema produzca esquemas de tipo relacional y que le proporcione al usuario su descripción en términos de las tablas, columnas, llaves primarias y llaves foráneas que constituyen su base de datos, además siguiendo los lineamientos que se plantean en este trabajo se pueden implantar otros modelos de base de datos como el orientado a objetos.

I ARQUITECTURA DEL SISTEMA

En la figura 1 se muestra el Diagrama de Flujo de Datos del sistema, los procesos que lo componen se describen en forma general en las secciones de este capítulo.

1 DETERMINACIÓN DE UNIDADES LÉXICAS

Este proceso se encarga fundamentalmente de pasar cada oración proporcionada por el usuario en lenguaje natural a su correspondiente forma canónica, sustituyendo cada una de las palabras por su tipo de unidad léxica. Las unidades léxicas que maneja el sistema son las que se muestran en la figura 2.

Debido a las restricciones que tiene el proceso de comunicación con el sistema, es necesario que el usuario tome en cuenta los siguientes lineamientos:

1) *Las palabras que definan una sola entidad o atributo, deben unirse por guiones*, ejemplos: cuenta-de-cheques, saldo-mensual

2) *Las palabras que indiquen el numero de ocurrencias de una entidad que esta siendo descrita, también deberán estar unidas por guiones*, ejemplos: un-sólo, un-único, uno-o-varios

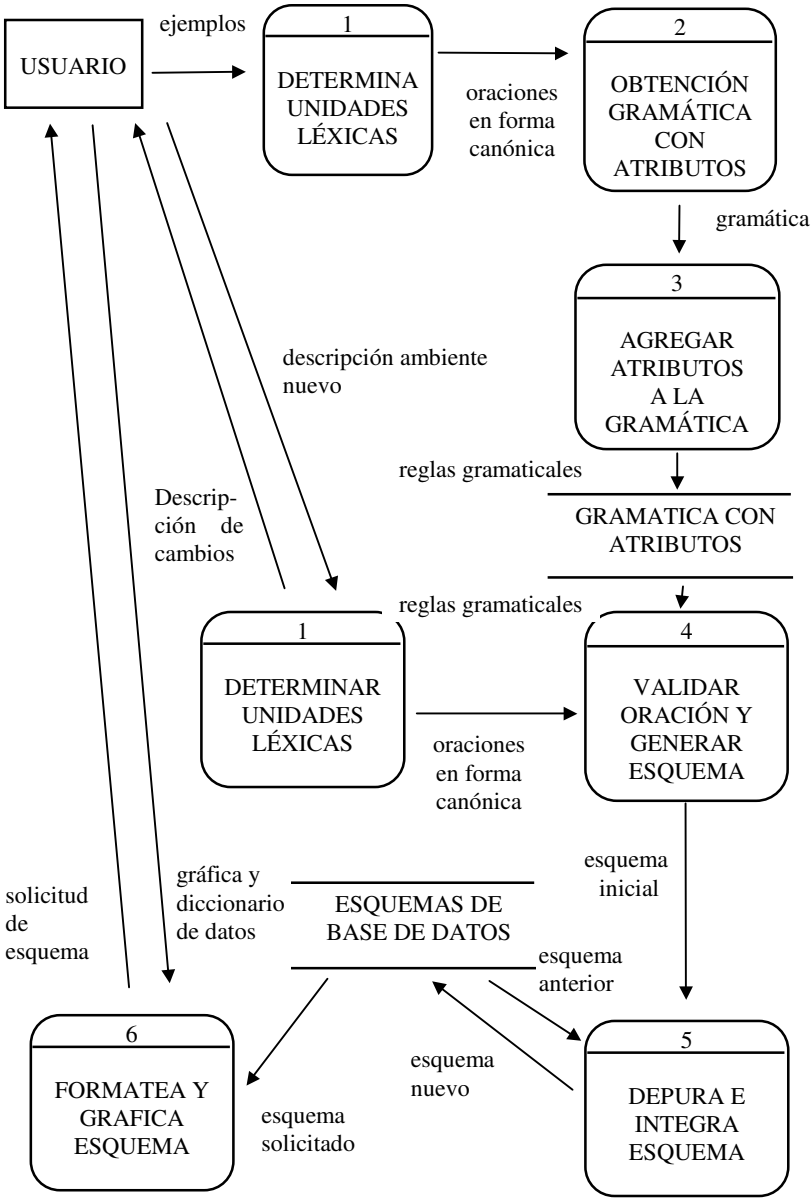

Figura 1. Arquitectura del Sistema Evolutivo Generador de Esquemas Lógicos de Base de Datos

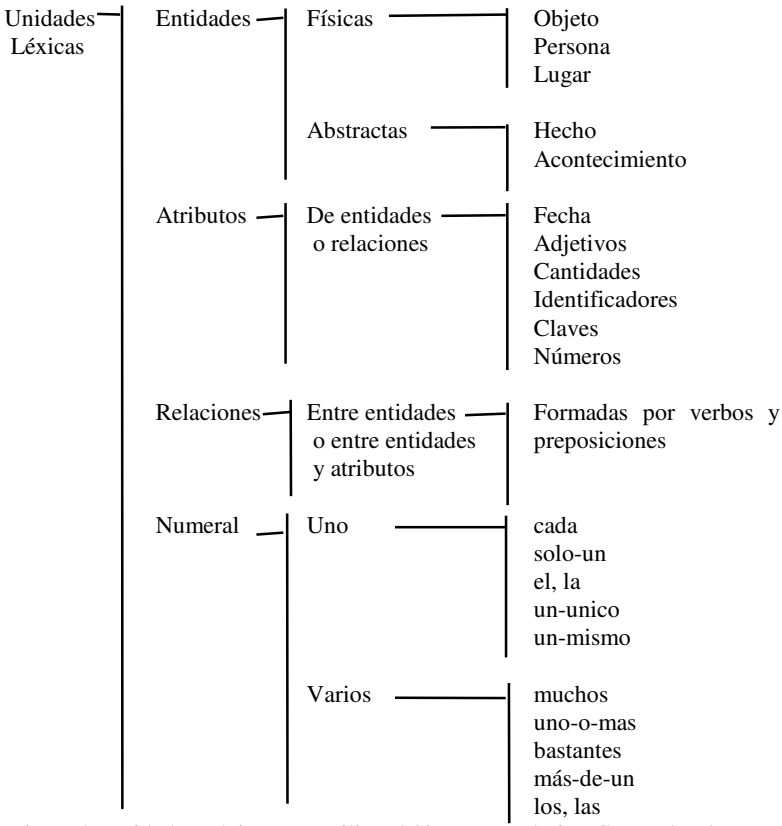

Figura 2. Unidades Léxicas que utiliza el Sistema Evolutivo Generador de Esquemas Lógicos de Base de Datos

Hay dos tipos de oraciones de entrada

1) *Oraciones que definen alguna relación*, ejemplo: *Cada empleado es asignado a un-sólo departamento*

En base a este tipo de oraciones se induce la gramática del lenguaje del usuario.

2) *Oraciones que ayuden al sistema a identificar el tipo de unidad léxica que le corresponde a un palabra que se haya mencionado en alguna oración*, por ejemplo:

Sistema: *¿Qué es un empleado?*
Usuario: *es una persona*

“persona" es una palabra clave que se considera en este ejemplo como sinónimo de Entidad.

2 Obtención de la Gramática Generativa

El objetivo de este proceso es obtener la gramática de las oraciones de ejemplo proporcionadas por el usuario. Esta gramática se obtiene.

1) *Detectando estructuras recursivas*. Para lograrlo se hace un barrido de cada oración para determinar cuantas veces dentro de una oración ocurre en forma contigua una unidad léxica. Las unidades que aparezcan mas de dos veces en forma contigua se consideraran como estructuras recursivas.

2) *Factorizando oraciones detectando estructuras intermedias*. Para lograrlo se recorren las oraciones por pares de unidades léxicas. Se marcan las parejas que se repiten más veces a lo largo de todas las oraciones. Luego se recorren las oraciones pero ahora para detectar tercias de unidades léxicas. Se hace lo mismo que en el inciso anterior pero ahora con cuartetos. Los de mayor ocurrencia se sustituyen por variables no terminales y se define una regla de producción para ella. Se hace esto desde los cuartetos hasta los pares mientras quede alguno.

3 Agregar Atributos a la Gramática para Incluirle la Semántica

En este proceso se agregan las rutinas semánticas a la gramática que se obtuvo en el proceso anterior, en este caso las rutinas están relacionadas con el manejo de una pila.

4 Se Valida la Oración y se Genera el Esquema de Base de Datos Inicial

Este proceso está constituido por dos partes: la primera parte usa las reglas gramaticales obtenidas en el proceso descrito en C y en base a ellas se determina si una oración de entrada es valida y

entendible para el sistema. La segunda parte, a partir de cada oración de entrada empieza a formar o modificar el esquema lógico de la Base de Datos que refleja el ambiente que el usuario le este describiendo.

5 Depuración e Integración del Esquema Definitivo

Si el usuario esta solicitando al sistema una modificación a un esquema ya almacenado, en este proceso se integraran las nuevas entidades, atributos y relaciones que se encontraron en el proceso descrito en D. Después de esto, se pasa por un proceso de normalización que permite depurar el esquema eliminando las anomalías que llegase a tener. Por otra parte, si el usuario esta solicitando la generación de un esquema por primera vez este proceso solo se encarga de normalizar el esquema inicial y de almacenarlo ya normalizado en el archivo de esquemas reconocidos por el sistema para que pueda ser consultado y/o modificado posteriormente por el usuario.

6 Formateo del Esquema y Graficación

En este proceso se accesa el archivo que tiene el esquema que desea ver el usuario y se formatea en un vocabulario entendible para el. La salida que se presenta al usuario esta compuesta básicamente por lo siguiente:

Un listado del esquema (entidades y relaciones) describiendo los objetos que lo constituyen.

ESQUEMA: nombre de la base de datos
ENTIDAD 1: nombre de la entidad
  nombre de la columna 1
  nombre de la columna 2
  . . .
  nombre de la columna n
LLAVE PRIMARIA: lista de columnas
LLAVES FORANEAS:
  lista 1 de columnas
  lista 2 de columnas
  …

lista m de columnas
ENTIDAD 2: nombre de la entidad
columna1 nombre de la columna
columna2 nombre de la columna
. . .
columna n nombre de la columna
LLAVE PRIMARIA: lista de columnas
LLAVES FORANEAS:
lista 1 de columnas
lista 2 de columnas
…
lista m de columnas

Otra salida que se obtiene es la gráfica del modelo entidad-relación formado con rombos y cuadrados correspondientes a relaciones y entidades respectivamente.

II Obtención de la Gramática con Atributos

El problema fundamental del Lenguaje Natural es la complejidad de las oraciones que puede llegar a recibir el sistema, puesto que a pesar de tratarse de oraciones meramente declarativas presentan peculiaridades tales como:

1) *Símbolos iguales que tienen diferente significado en función de la posición en que se encuentren*, por ejemplo:

Cada empleado tiene número de empleado, dirección y teléfono y es asignado a varios proyectos

en este caso la primera aparición de la letra "y" tiene la función de enlazar dos atributos que corresponden a una misma entidad, mientras que la segunda aparición de esta letra cumple la función de enlazar una oración con otra para proporcionar información adicional al respecto a la entidad principal de la oración.

2) *Símbolos que a pesar de ser diferentes pueden llegar a significar lo mismo*, por ejemplo:

Cada empleado tiene número de empleado, dirección y teléfono

como se puede observar la unidad léxica "," y la unidad léxica "y" en el ejemplo cumplen la misma función, ambas actúan

como nexos entre varios atributos que corresponden a una misma entidad.

3) *Recursividad a varios niveles*, en donde el primer nivel se puede ver en el ejemplo siguiente:

ORACIÓN EN LENGUAJE NATURAL	**ORACIÓN CANÓNICA**
Cada empleado tiene número-de-empleado.	c e r a .
Cada empleado tiene número-de-empleado, dirección.	c e r a , a
Cada empleado tiene número-de-empleado, dirección, edad.	c e r a , a , a
Cada empleado tiene número-de-empleado, dirección, edad, estado civil.	c e r a , a , a , a

donde: a = atributo, e = entidad, r = relación, c =cuantificador

La aparición consecutiva de la unidad léxica "a" separada por "," puede considerarse recursiva y por lo tanto ser representada con las siguientes reglas de producción.

$S \rightarrow c\ e\ r\ a\ A$
$A \rightarrow . \mid , a\ A$

El segundo nivel de recursión se visualiza el ejemplo siguiente:

ORACIÓN EN LENGUAJE NATURAL	**ORACIÓN CANÓNICA**
Cada empleado es-asignado-a un departamento.	c e r c e .
Cada empleado es-asignado-a un departamento y a un proyecto.	c e r c e , c e .
Cada empleado es-asignado-a un departamento, a un proyecto y a un grupo	c e r c e , c e , c e

$S \rightarrow A\ .\ |\ A\ ,\ B\ .\ |\ A\ ,\ B\ ,\ B\ .$
$A \rightarrow c\ e\ r\ c\ e$
$B \rightarrow c\ e$

simplificando

$S \rightarrow A\ X$
$X \rightarrow .\ |\ ,\ B\ X$
$A \rightarrow c\ e\ r\ c\ e$
$B \rightarrow c\ e$

A su vez dentro de A y B puede haber un primer nivel de recursividad.

El tercer nivel de recursión se ejemplifica a continuación:

ORACIÓN EN LENGUAJE NATURAL	**ORACIÓN CANÓNICA**
Cada empleado es-asignado-a un departamento; cada departamento maneja varios proyectos; cada proyecto tiene número-de-proyecto, fecha-de-inicio y fecha-de-terminación .	c e r c e ; c e r c e ; c e r a , a y a .

$S \rightarrow A\ ;\ A\ ;\ A\ .$
$A \rightarrow c\ e\ r\ c\ e\ |\ c\ e\ r\ c\ e\ |\ c\ e\ r\ a\ ,\ a\ y\ a$

simplificando

$S \rightarrow A\ X$
$X \rightarrow .\ |\ ;\ A\ X$

a su vez dentro de A puede presentarse el segundo nivel de recursividad.

III Módulos del Sistema Evolutivo Generador de Esquemas Lógicos de Base de Datos

Las funciones del Sistema Generador del Esquemas Lógicos de Base de Datos son fundamentalmente dos, aprender el lenguaje del usuario y utilizar el conocimiento adquirido para generar un

esquema de Base de Datos de acuerdo a los requerimientos del usuario.

Para aprender el lenguaje del usuario se usan los módulos:

1) *Módulo de transformación de oraciones a su forma canónica*

2) *Módulo de obtención de la gramática generativa*

3) *Módulo de obtención de la gramática con atributos*

Para obtener el esquema de base de datos se usa el módulo:

1) *Módulo generador de esquemas de base de datos*

1 Módulo de Transformación de las Oraciones a su Forma Canónica

El objetivo de este módulo es identificar las unidades léxicas de cada oración de entrada para obtener su forma canónica formateada de tal manera que pueda ser de utilidad para la inducción de la gramática como para la generación del esquema de base de datos.

Los pasos que sigue el para transformar la oración en lenguaje natural a su correspondiente forma canónica son:

1) *Generación de la oración canónica inicial*. Consiste en sustituir cada palabra de la oración en lenguaje natural por su tipo de unidad léxica correspondiente.

Evidentemente el sistema en un principio no puede reconocer todas las palabras que contiene la oración puesto que solo cuenta con el vocabulario mínimo indispensable para realizar sus funciones, sin embargo tiene la capacidad de ir incrementando ese vocabulario conforme se vayan utilizando nuevas palabras dentro de las oraciones.

Cuando el sistema no reconoce alguna palabra, le pregunta al usuario de que tipo de unidad léxica se trata y la almacena como tal para poder reconocerla la próxima vez que la utilice el usuario, por ejemplo:

En cada departamento se tienen varios empleados con nombre, edad, sexo, dirección y además tienen varios proyectos con clave, nombre, fecha-de-inicio y fecha-de-terminación.

UNIDAD LÉXICA	NOMBRE TIPO	TIPO
En	preposición	p
cada	cuantificador	c
departamento	entidad	e
se	pronombre	o
tienen	verbo	v
varios	cuantificador	c
empleados	entidad	e
con	preposición	p
nombre	atributo	a
,	coma	,
edad	atributo	a
,	coma	,
sexo	atributo	a
,	coma	,
dirección	atributo	a
y	nexo	y
además	adverbio	d
tienen	verbo	v
varios	cuantificador	c
proyectos	entidad	e
con	preposición	p
clave	atributo	a
,	coma	,
nombre	atributo	a
,	coma	,
fecha-de-inicio	atributo	a
y	nexo	y
fecha-de-terminación	atributo	a
.	punto	.

2) *Aprendizaje de nuevas relaciones*. En esta etapa la oración no sufre ningún cambio puesto que el sistema sólo analiza la palabra o conjunto de palabras que el usuario haya marcado explícitamente como relaciones. Este análisis está encaminado a descubrir la estructura interna de las unidades léxicas Relación para darles de alta en la tabla de gramática de relaciones para que el usuario en futuras oraciones no tenga

que unir con guiones aquellas palabras que conforman una relación, por ejemplo:

Cada empleado es-asignado-a un proyecto

UNIDAD LÉXICA	NOMBRE TIPO	TIPO
Cada	cuantificador	c
empleado	entidad	e
es-asignado-a	relación	r
un	cuantificador	c
proyecto	entidad	e

Si un usuario marca explícitamente la unidad léxica Es-asignado-a como una relación, se hace el análisis siguiente:

i) *Se separa la relación eliminando lo guiones*

es
asignado
a

ii) *Se determina el tipo de unidad léxica de cada palabra de la relación*:

es	verbo	v
asignado	verbo	v
a	preposición	p

UNIDAD LÉXICA	**NOMBRE TIPO**	**TIPO**
es	verbo	v
asignado	verbo	v
a	preposición	p

iii) *Se busca en la tabla TipRel la combinación* ***v v p*** *y si no se encuentra, se inserta esa combinación como un renglón nuevo de esa tabla.*

TipRel (antes)

v	v		
v			

TipRel (después)

v	v	p	
v	v		
v			

De esa manera la próxima vez que el usuario inserte una relación del tipo **v v p**, ya no será necesario que una las tres palabras con guiones.

3) *Sustitución de relaciones*. Esta etapa consiste en aplicar el conocimiento adquirido en la etapa anterior respecto a la estructura interna que guardan las unidades léxicas Relación. Para ello el sistema va recorriendo la oración canónica tratando de encontrar varias unidades léxicas contiguas que puedan agruparse para formar una Relación, de ser así sustituye las unidades léxicas correspondientes por la unidad léxica Relación, por ejemplo:

Tomando como base la oración canónica que se obtuvo en el punto 1), las unidades **o**, **v** y la unidad **v** fueron sustituidas por una unidad **r**.

4) *Depuración de la oración*. Una vez que han sido detectadas las oraciones dentro de la oración canónica, el siguiente paso es eliminar las preposiciones, artículos y demás unidades léxicas que se considera no serán determinantes durante la generación del esquema de lógico que corresponda a esa oración.

2 MÓDULO DE OBTENCIÓN DE LA GRAMÁTICA GENERALIZADA MEDIANTE INFERENCIA GRAMATICAL

Este módulo se encarga de inferir la gramática generativa del lenguaje del usuario a partir de un conjunto de oraciones que se alimentan al sistema durante su etapa de aprendizaje. Para lograr esto, efectúa el siguiente proceso sobre las oraciones canónicas que se obtuvieron como salida del modulo descrito en A.

1) *Fraccionar oraciones complejas en oraciones mas simples*. Inicialmente el sistema toma cada oración canónica y la desglosa en oraciones mas simples pero completas, sustituye en la oración original cada oración simple por un símbolo no terminal y graba cada fracción de oración como una opción de la regla de producción que corresponde a ese símbolo no terminal. Por ejemplo, considérense las oraciones siguientes:

ORACIÓN EN LENGUAJE NATURAL	ORACIÓN CANÓNICA
Cada cliente tiene una cuenta-de-cheques; cada cuenta pertenece a una sucursal; cada sucursal pertenece a una plaza.	c e r c e # @ ; c e r c e # @ ; c e r c e # @ .
Un cliente puede tener varias cuentas-de-cheques y varias cuentas-de—vpf; cada cuenta-de-vpf está asignada a un cliente.	c e r c e # y c e # @ ; c e r c e # @ .
Un cliente tiene una cuenta-de-cheques, varias cuentas-de-cartera y varias cuentas-de-bursatil.	c e r c e # y c e # y c e # y c e # @ .
Un cliente con nombre, dirección, teléfono y edad puede tener una cuenta-de-cheques y una cuenta-de-bursatil.	c e a , a , a , a r c e # y c e # @ .
Cada cliente tiene nombre y dirección	c e r a , a # @ .

Fraccionando las oraciones originales en oraciones más simples:

S → A ; A ; A . | A ; A . | A . | A.
A → c e r c e # @ ; c e r c e # @ ; c e r c e # @ . |
c e r c e # y c e # @ ; c e r c e # @ . |
c e r c e # y c e # y c e # y c e # @ . |
c e a , a , a , a r c e # y c e # @ . |
c e r a , a # @ .

2) *Detección del primer nivel de recursividad.* Una vez desglosadas las oraciones, se analiza la primera regla de producción y se determina si el símbolo no terminal aparece suficientes veces como para considerar que puede expresarse con una regla recursiva, de ser así, la primer regla se sustituye por dos para expresar esa recursividad.

Después de esto, se analiza cada una de las oraciones simples que ya se separaron de su oración original y se verifica si aun pueden simplificarse mas por estar compuestas de segmentos

de oración; de ser así, se desglosa cada oración simple y se genera un nuevo símbolo no terminal para colocarlo en sustitución de cada segmento de oración.

En base a la gramática obtenida en el paso anterior se tiene:

S → A B
A → C @ | C y D @ | C y D y D y D @ | C y D @
B → ; A B | .
C → c e r c e # | c e a , a , a r c e # | c e r a , a #
D → c e #

3) *Detección del segundo nivel de recursividad*. Este paso consiste en determinar si el segundo símbolo, no terminal (D) que fue generado en el paso anterior aparece suficientes veces como para considerarlo recursivo, de ser así, la regla de producción en donde aparece se transforma en dos reglas que expresen esa recursividad.

A → C @ | C y D @ | C y D y D y D @ | C y D @
queda como:

A → C E
E → y D E | @

4) *Detección del tercer nivel de recursividad*. Debido a que el tercer nivel de recursividad se puede dar específicamente en el caso de la unidad léxica Atributo, y puesto que esta recursividad esta empotrada dentro de cada segmento de oración, si se detecta que aparece consecutivamente tres veces o mas, se asume que es recursiva y se extrae de la oración junto con la unidad léxica que le sucede para que sirva como símbolo de terminación de la recursividad y en el sitio done se extrajo el conjunto de atributos, se incluye un nuevo símbolo no terminal y se genera su regla de producción correspondiente.

C → c e r c e # | c e a , a , a , a r c e # | c e r a , a #
queda como:

C → c e r c e # | c e F c e # | c e r F#
F → a G
G → , a G | r | #

5) *Depuración de la gramática*. Finalmente se ordenan las alternativas de cada regla de producción y se eliminan aquellas que sean iguales para dejar simplificada la gramática.

La gramática resultante del ejemplo es:

S → A B
A → C E
B → ; A B | .
C → c e r c e # | c e F c e # | c e r F#
D → c e #
E → y D E | @
F → a G
G → , a G | r | #

3 MÓDULO PARA OBTENER LA GRAMÁTICA CON ATRIBUTOS

La gramática con atributos es una de las herramientas que nos permiten asignar y manejar la semántica de un lenguaje.

Este tipo de gramática se caracteriza por llevar intercalados en sus reglas de producción llamados a rutinas semánticas.

Las rutinas semánticas se representan como símbolos no terminales y su función es la de llevar a cabo alguna acción sobre los datos, por ejemplo:

S → c a r S1 c a . S2

donde S1 y S2 son rutinas semánticas.

El ejemplo indica que si se esta en el símbolo S y llega la señal **r**, entonces se manda a ejecutar la rutina S1 y se verifica si llega la señal **c**.

Las rutinas semánticas del sistema evolutivo en cuestión se basan en el manejo de una pila que se utiliza como entrada de la oración que se va a reconocer y en función del contenido de esta pila se van llenando las estructuras de datos de la figura 3.

ENTIDADES

	entidad
1	cuenta-de-cheques
2	sucursal
3	
4	

ATRIBUTOS

atributo	entidad
saldo-mensual	1
numero-de-cuenta	1
numero-de-sucursal	2

RELACIONES

c1	e1	relación	c2	e2
u	1	es-abierta-en	u	2
u	2	maneja	m	1

donde c1=cuantificador1, c2=cuantificador2, e1=entidad1, e2=entidad2, u=uno, m =muchos

Figura 3. Estructuras de datos utilizadas para generar y modificar los esquemas de base de datos

Un ejemplo de las rutinas semánticas de este sistema es el siguiente:

S1: Rutina semántica que se ejecuta ante la entrada de la unidad léxica CUANTIFICADOR

```
ALGORITMO:
Si la pila esta vacía o en el tipo de la pila hay una relación o una entidad
entonces
   almacena el cuantificador en la pila
sino
  si en el tope de la pila hay una "y" entre oraciones
  entonces
      saca la "y" de la pila
      almacena el cuantificador en la pila
```

La complejidad del diseño de las rutinas semánticas de este sistema radica en el hecho de que cada atributo debe ser asociado con la entidad que le corresponde y las entidad que intervienen en la oración deben ser correctamente relacionadas unas con otras,

independientemente de la estructura que guarde la oración de entrada.

4 MÓDULO DE GENERACIÓN DEL ESQUEMA DE BASE DE DATOS

Este modulo cumple dos funciones básicas:

1) *Determinar si una oración de entrada corresponde al lenguaje que aprendió el sistema.*

2) *Interpretar cada oración de entrada y construir el esquema lógico que la representa*

Estas dos funciones se llevan a cabo simultáneamente gracias a la gramática con atributos que se obtuvo en el modulo anterior, puesto que al ir siguiéndola es posible determinar si la oración esta bien construida al mismo tiempo que se van ejecutando las rutinas semánticas que llenan las estructuras de datos que reflejan el esquema lógico correspondiente.

Al final de este modulo existe una rutina que se encarga de formatear el esquema obtenido, de manera que sea entendible para el usuario.

Ejemplo: Un usuario podría describir su ambiente problema de la manera siguiente:

> *Un cliente puede tener varias cuentas-de-cheques. Cada cuenta-de-cheques tiene número-de-cuenta, saldo, número-de-retiros y número-de-depósitos y puede pertenecer a uno-o-mas clientes. Un cliente es identificado por su nombre y RFC. Cada cliente tiene dirección y número-telefónico. Cada cuenta-de-cheques es abierta en una sucursal y cada sucursal puede abrir varias cuentas-de-cheques.*

El esquema lógico que propone el generador es el siguiente:

ENTIDADES

CLIENTE

nombre
RFC
dirección
número-telefónico
identificador-cliente

CUENTA-DE-CHEQUES
número-de-cuenta
saldo
número-de-retiros
número-de-depositos
identificador-cuenta-de-cheques

SUCURSAL
identificador-sucursal

RELACIONES

1 CLIENTE	PUEDE-TENER	M CUENTAS-DE-CHEQUES
1 CUENTA-DE-CHEQUES	PUEDE-PERTENECER-ES-ABIERTA-EN	M CLIENTES 1 SUCURSAL
1 SUCURSAL	PUEDE-ABRIR	M CUENTAS-DE-CHEQUES

Ejemplos de oraciones que se pueden utilizar para actualizar un esquema de base de datos son:

> *Desde mañana los clientes tienen derecho a darle una extensión de su cuenta de cheques a sus hijos. En el cálculo de impuestos se maneja una tabla de subsidio que tiene los mismos atributos que la tabla actual del cálculo del ISR.*

IV Obtención Automática de Llaves y Tipos de Datos (Propuesto por Jesús Manuel Olivares Ceja)

Las llaves de una entidad se pueden obtener en base al tipo de consultas que haga el usuario, por ejemplo si se solicita:

> *Dame las personas cuyo apellido inicia con L. Busca los datos del empleado Jiménez.*

el sistema genera un índice por apellido para facilitar la atención a estos requerimientos.

El tipo y la longitud de los atributos se puede obtener en base a ejemplos de datos que se maneja para cada uno, de donde el sistema infiere el tipo y longitud más apropiados. En un momento dado puede efectuar un cambio tanto en tipo como en longitud si los datos que llegan son diferentes:

Por ejemplo si se da: *Algunos nombres de personas son: Juan Pérez Herrera, Claudia Bracamontes.*

El sistema asigna un atributo de tipo alfabético de la longitud máxima del nombre que tenga registrado.

En otro ejemplo, cuando se da: *El sexo únicamente puede ser masculino o femenino.*

El sistema puede asignar una variable que tome dos valores como por ejemplo un bit.