ÁLGEBRA RELACIONAL COMO LENGUAJE DE ACCESO A BASES DE DATOS RELACIONALES

Ginés Moreno Valverde
Tomás Rojo Guillén
Pascual González López

Ginés Moreno Valverde, Tomás Rojo Guillén y Pascual González López están en el Departamento de Informática. Esc. Univ. Politécnica de Albacete, Universidad de Castilla-La Mancha. Campus Universitario, s/n, 02071 Albacete, España. Tel: (967) 228551. Fax: 34 67 506650 Email: gmoreno@ info-ab.uclm.es.

RESUMEN

En este artículo proponemos una notación y un nuevo lenguaje de manipulación de bases de datos relacionales, que equivalen a una traducción fiel y directa (tanto sintáctica como semánticamente) del álgebra relacional de conjuntos. Presentamos una crítica comparativa con otros lenguajes del mismo ámbito, y desarrollamos las técnicas aplicadas al diseño e implementación de un lenguaje para la manipulación de datos en modo inmerso en el seno de un sistema de bases de datos relacional. Se detallan las características del lenguaje inmerso y del anfitrión de la base de datos, así como el interfaz de comunicación que se establece entre ambos extremos. La discusión se plantea sobre un sistema relacional conocido dentro del entorno de los ordenadores personales como es DB3 (de Ashton Tate), y se introduce el lenguaje que actuará en modo inmerso dentro del mismo: BAREL.

PALABRAS CLAVES: Bases de Datos, Lenguajes de manipulación de datos, Lenguajes inmersos, Álgebra Relacional.

1. INTRODUCCIÓN

PRIMERO fueron las bases de datos jerárquicas, después el modelo en red, ahora se investiga en proyectos aún experimentales de bases de datos deductivas y orientadas al objeto, pero es indudable que el modelo de base de datos más implantado y difundido en la actualidad y el que presumiblemente mayor expansión continuará teniendo en el futuro es el relacional. Las razones se deben a la simplicidad de sus sistema de almacenamiento de información, así como a la comodidad de

acceso y recuperación de datos que caracterizan a sus estructuras, ya que, como es sabido, el modelo relacional pretende minimizar la cantidad de memoria necesaria para almacenar datos (poca redundancia de la información), así como maximizar la velocidad de las operaciones de acceso que se permiten en dicho modelo [ULL89].

Una de las principales causas del éxito que caracteriza al modelo relacional de datos hay que buscarla en su fuerte fundamentación teórica. La base matemática que sirve de base a este esquema de tratamiento de la información es bastante clásica, bien conocida y probada: la teoría de conjuntos y bolsas, el cálculo matricial y de predicados... Pero es sobre todo en el álgebra relacional donde se encuentra la incidencia más directa (puede decirse que inmediata) sobre el mundo informático de las bases de datos relacionales. No es de extrañar que ante una base teórica bien afianzada y reconocida (incluso mucho antes de que aparecieran los primeros computadores) el proyecto relacional haya madurado tan rápida como felizmente [FER87].

Aquí se plantea una discusión sobre la traducción que se ha hecho en el campo de la informática de las fuentes teóricas que inspiran el modelo relacional. En particular, creemos que algunos de los lenguajes de manipulación de datos que se han diseñado en la actualidad sobre bases de datos relacionales, se han distanciado sensiblemente de su inspiración teórica, y en consecuencia han perdido cierta propiedades.

Por todo ello, lo que aquí proponemos es una nueva notación para expresar el potencial que la traducción directa del álgebra relacional alcanza dentro del ámbito de las bases de datos relacionales. Y además ofrecemos una primera implantación de la misma (en forma de lenguaje inmerso) sobre un conocidísimo sistema relacional de gestión de datos registrado por Ashton-Tate: DB3 ó DBASE III.

2. BASES DE DATOS RELACIONALES Y DBASE III

Una Base de Datos es un conjunto de información útil organizada de una forma específica. Este concepto no es precisamente igual que el de gestor de ficheros, con el que nunca debe confundirse, ya que un programa gestor de ficheros únicamente se limita a abrir ficheros de datos y a realizar informes con ellos. No suelen crecer a sistemas complejos y no suelen tener lenguaje de programación. La potencia y flexibilidad que encierra un verdadero sistema de datos, como el propio DB3, es lo que hace que éste se convierta en una herramienta cada vez más utilizada.

La información almacenada en una base de datos puede ser estructurada o visualizada en varias formas. El modelo relacional organiza una base de datos como un conjunto de relaciones o tablas de dos dimensiones que consisten en filas y columnas. Cada fila contiene infor-

mación correspondiente a distinta ocurrencias de una entidad que describe a un objeto. Esta entidad tiene a su vez asociados una serie de atributos correspondientes a cada una de las columnas de la relación [GAR89]. Por ejemplo, una relación o tabla de una base de datos relacional puede tener el formato que muestra la figura 1.

De un modo riguroso, y usando una nomenclatura matemática, que por otro lado es la originaria del modelo relacional, tenemos que definir una relación (ó tabla, ó archivos de datos de una base de datos relacional) de esta manera:

> *«Dada una serie de conjuntos $D_1, D_2, ..., D_n$ no necesariamente distintos), se dice que R es una relación sobre esos n conjuntoss si es un conjunto de n tuplas ordenadas $\langle d_1, d_2, ..., d_n \rangle$ tales que d_1 pertenece a D_1, d_2 pertenece a $D_2, ..., d_n$ pertenece a D_n. Los conjuntos $D_1, D_2, ..., D_n$ son los dominios de R. El valor n es el grado y el número de tuplas la cardinalidad de relación.»*

Un fichero de base de datos relacional (que es la implementación práctica de una relación teórica) consiste en dos partes fundamentales (y aquí está la diferencia con lo ficheros normales, que únicamente suelen presentar área de datos). Una parte define la estructura de los registros de datos y la otra contiene los datos propiamente dichos. En DBASE III estas dos regiones se guardan en los mismos ficheros de datos (con extensión **DBF**) aunque en otra bases de datos relacionales más generales no necesariamente se sigue este convenio, y en muchas ocasiones ambos bloques de información se almacenan en otra regiones como pueden ser tablas especiales de especificaciones, o incluso en el diccionario de datos del gestor del sistema. Y centrándonos ya en DBASE III, veamos como se construyen estas dos zonas dentro de un

	PRENDA	COLOR	TALLA	TEJIDO
1	pantalón	verde	40	algodón
2	camisa	azul	30	rayón
3	jersey	rojo	40	rayón
4	camisa	verde	30	algodón
5	pantalón	azul	20	lino
6	pantalón	rojo	41	rayón
7	camisa	rojo	30	rayón
8	jersey	verde	40	algodón
9	corbata	naranja	–	seda

FIGURA 1.
Ejemplo de tabla o relación.

fichero **DBF** (información que será necesaria para diseñar los operadores algebraicos de BAREL que introduciremos posteriormente).

a) **Cabecera de especificación.** Es la cabecera de especificación un área de información de control de tamaño variable (dependiendo del número de campos que tenga el registro patrón) que se localiza al principio de todo fichero DBF. Los primeros 32 bytes siempre aparecen con información general del archivo en el que se incluyen (fecha de creación, número y tamaño de los registros, etc.). A continuación se reservan 32 bytes para la descripción de cada uno de los campos o atributo que se inscriben en el patrón de registros (nombre y tipo del campo, longitud, número de decimales...). Un último byte nulo se usa para marcar el final de esta cabecera de especificación.

b) **Área de datos.** Los registros que componen el fichero «DBF» en cuestión se almacenan en el archivo uno detrás de otro, siguiendo el orden cronológico en el que se insertaron, y guardando entre ellos un único espacio en blanco. El primer registro introducido, se localiza justo a continuación de la cabecera de definición, mediando entre ellos, como es normal, un espacio en blanco. Por otro lado, los datos asociados a un registro se guardan en formato ASCII, independientemente de que su tipo sea numérico, fecha, lógico, etc.

3. ÁLGEBRA RELACIONAL

El álgebra Relacional es una disciplina matemática construida a partir de cinco operadores básicos sobre los que se apoyan una serie de axiomas y teoremas, siendo especialmente valioso el que implica su carácter de completitud [FER87]. Este resultado, demostrado por Codd, es de una relevancia especial, pues con él se puede demostrar que cualquier acceso a una base de datos relacional puede realizarse en término de estos operadores.

Los operandos del modelo se conocen como relaciones (constantes o variables y de grado fijo). Una relación es un subconjunto de un producto cartesiano de conjuntos (dominios). A cada uso particular de un dominio en una relación se le denomina atributo, de modo que un atributo, digamos A, está restringido a tomar sus valores en un dominio, que designaremos $Dom\ (A)$. Así la relación R definida sobre los atributos $A_1, A_2, ..., A_n$, es un subconjunto del producto cartesiano de los dominios correspondientes, es decir, R está incluida o coincide con: $Dom\ (A_1) * Dom\ (A_2) * \ldots Dom\ (A_n)$. Como puede verse, la similitud entre una relación del álgebra relacional y una tabla del modelo relacional o un fichero DBF de DBASE III, resulta más que evidente.

Los cinco operadores básicos y por otra parte necesarios para describir y definir el álgebra Relacional (con carácter relacionalmente completo, como ya dijimos) son:

- **Unión de conjuntos.** La unión de dos relaciones R y S, denotada por $R \cup S$, es el conjunto de tuplas que pertenecen a R, a S o a ambas. Solo se aplica este operador a relaciones del mismo grado y definidas sobre los mismos atributos.
- **Diferencia de conjuntos.** La diferencia de dos relaciones R y S, denotada por $R - S$, es el conjunto de tuplas de R que no pertenecen a S. Las mismas restricciones del caso anterior se verificarán sobre R y S.
- **Producto cartesiano.** Sean R y S, dos relaciones de grados m y n, respectivamente. El producto cartesiano $R * S$ es una relación de grado $m + n$, constituida por todas las posibles tuplas, en que los m primeros elementos constituyen una tupla de R, y los n últimos una tupla de S.
- **Proyección.** La proyección $\pi_x (R)$, donde R es una relación definida sobre J (esquema o conjunto de atributos posibles) y x está incluido o coincide con J, es una relación construida por las columnas de R correspondientes a los atributos de x.
- **Selección.** Sea F una fórmula que involucra:
  i) Operandos constantes, y que referencian a atributos de la relación implicada R.
  ii) Operadores aritméticos de comparación: <, >, =, <=, >=, <>.
  iii) Operadores lógicos: AND, OR.

  Entonces la selección $\sigma_F (R)$ es el conjunto de tuplas de R tales que una vez sustituida en la fórmula F, las ocurrencias de los atributos que en ella se referencian, resulta verdadera.

En realidad, aunque estos cinco operadores que acabamos de detallar hacen completa la estructura relacional recién definida, existen nuevos operadores que pueden expresarse en términos de los anteriores, pero que por su frecuente uso reciben nombre especial y a veces se usan como funciones primitivas. De hecho, en nuestra implementación pueden resultar más útiles que los primeros (potentes pero excesivamente simples), y por ello aparecerán construidos independientemente de los demás.

Estos son:

- **Intersección.** Es el inverso a la resta, y se resuelve por la tuplas comunes a las dos relaciones que se intersectan: $R \cap S = (R - (R - S))$
- **División o cociente.** Sean R y S relaciones de grado r y s, respectivamente, donde $r > s$ y $S <> \varnothing$. Entonces el cociente $R \div S$

es el conjunto de tuplas t de grado $(r - s)$, tales que para toda tupla u de S, la tupla tu está en R.
Como fácilmente puede comprobarse, este operador, se expresaría en función de los primitivos como:

$$R \div S = \pi_{1,2,\ldots,r-s}(R) - \pi_{1,2,\ldots,r-s}((\pi_{1,2,\ldots,r-s}(R) * S) - R).$$

Nos remitimos a las figuras de página atrás, donde se muestra en la tabla COCIENTE el resultado de aplicar este operador sobre las tablas DIVIDENDO y DIVISOR.

- **Reunión o Join.** La α-reunión de R y S sobre las columnas i y j, que se denota como $R_{(i \propto j)} \rhd\lhd S$, donde α es un operador aritmético de comparación, equivalente a: $\sigma_{i \propto (r+j)}(R * S)$, siendo r el grado de R.
- **Reunión natural o Njoin.** Sean R y S dos relaciones con los mismos atributos comunes $A_1, \ldots, A_k$, entonces la njoin simbolizada por $R \rhd\lhd S$ se calcula así:
  i) Se halla el producto cartesiano $R * S$.
  ii) Para cada atributo A_i se seleccionan las filas en las que el valor $R \cdot A_i$ coincide con el valor $S \cdot A_i$.
  iii) Realizada la selección, se eliminan las columnas $S \cdot A_i$.

  Esto dicho equivale a:

$$R \rhd\lhd S = \pi\, i_1, i_2, \ldots, i_m\, (\sigma_{R \cdot A_1 = S \cdot A_1 \text{ and } \ldots\ R \cdot A_k = S \cdot A_k}(R * S))$$

  donde $i_1, i_2, \ldots, i_m$ es la lista de todos los componentes de $R * S$ en orden, excepto los componentes comunes a R y S.

En la figura 2 mostramos la sintaxis de los operadores originales y la adoptada por el nuevo lenguaje BAREL (de BAse RELacional).

4. LENGUAJES INMERSOS

4.1. Conceptos básicos

El cada día más amplio y complejo mundo de los lenguajes y modelos de programación existentes, plantea en ocasiones la necesidad de ampliar un determinado lenguaje informático para abordar problemas de programación específicos. La continua especialización de los lenguajes de programación obliga al programador acostumbrado al uso de lenguajes estándar de propósito general, a conocer cada vez má un mayor número de éstos para poder llevar a cabo distintas facetas de su trabajo y de esta manera codificar cada una de las distintas áreas de que consta su algoritmo original con un lenguaje diferente (dependiendo de la afinidad y competencias del mismo con respecto a la porción y características del trabajo que pretende resolver), obteniendo así un siste-

OPERADOR RELACIONAL	GRAMÁTICA MATEMÁTICA	GRAMÁTICA BAREL
UNIÓN	$A \cup B$	A + B
RESTA	$A - B$	A - B
PRODUCTO CARTESIANO	$A \times B$	A * B
INTERSECCIÓN	$A \cap B$	A \ B
DIVISIÓN O COCIENTE	$A \div B$	A / B
REUNIÓN NATURAL	$A \rhd\lhd B$	A NJOIN B
REUNIÓN	$A_{i \propto j} \rhd\lhd B$	A JOIN[expresion] B
PROYECCIÓN	$\pi_x (A)$	PROJECT[esquema] A
SELECCIÓN	$\sigma_x (A)$	SELECT[expresión] A

FIGURA 2.
Los operadores algebraicos.

ma ejecutable global, que proviene de un conglomerado de módulos implementados con lenguajes heterogéneos. Y aunque el problema no es excesivamente grave en el mundo de los microordenadores (donde existen una gran variedad de dialectos de lenguajes estándar que permiten practicar –por ejemplo– al mismo tiempo la programación de sistemas y la de gestión en un mismo programa sin cambiar de lenguaje) si es cierto que en entornos de grandes sistemas informáticos (donde los diferentes lenguajes de programación son bastante independientes entre sí) el problema se desborda. Máxime si se dispone de escasas y poco flexibles herramientas de programación.

Por todo ello, lo que aquí pretendemos es introducir la noción de *lenguajes inmersos* como una posible solución al problema planteado, que permita ampliar y flexibilizar el campo de actuación de lenguajes especializados, hacia áreas de distinta vocación para las que fueron primeramente diseñados.

Las ventajas de este modelo consisten en que, si bien el lenguaje original queda intacto en su estructura y diseño primitivo, su potencia computacional crece tanto como se quiera gracias a las nuevas capacidades que le aportan los lenguajes inmersos. Pero además se puede conseguir con muy pocas modificaciones sobre el preprocesador o traductor que acompaña al lenguaje inmerso generado, que su diseño puede reutilizarse e incluirse dentro de otros lenguajes para aportarles

sus propias cualidades (aquellas con las que específicamente se construye).

De esta manera, la inmersión de un lenguaje dentro de otro genera un lenguaje mixto de mayor capacidad expresiva. En realidad y en la mayoría de los casos (siempre hay excepciones como después detallaremos), los lenguajes inmersos son más bien pseudolenguajes que no tienen existencia propia fuera del contexto de los lenguajes sobre los que se «hospedan», en base a que dependen de los traductores de estos últimos, así como de los preprocesadores que para ellos se diseñan. Además, no interesa que estos lenguajes inmersos sean complicados y de aplicaciones múltiples: deben ser sencillos y breves, concretos y específicos, en el sentido de poder aportar nuevas propiedades a los lenguajes anfitriones, sin complicar la mezcla de ambos. Un buen diseño de estos pseudolenguajes *ampliadores* (y de sus preprocesadores-traductores) deberá permitir la inclusión de varios de ellos dentro de un mismo texto fuente, rodeando las sentencias del lenguaje estándar con el que se combinan, de tal modo que el programador pueda *completar* sus programas escritos con un lenguaje de propósito general, utilizando todos los lenguajes inmersos de que disponga y considere oportunos. Asimismo, y en el otro extremo, siempre será deseable que un mismo lenguaje inmerso sea transportable a nuevos entornos (lenguajes anfitriones) de inmersión, como ya dijimos anteriormente.

En definitiva, la gran aportación de los lenguajes inmersos es la de permitir al diseñador de aplicaciones hacer uso de potentes herramientas dentro de sus programas, sin tener que recurrir a mecanismos de inclusión de librerías, implementación de nuevas funciones, etc. ya que los preprocesadores de estos pseudolenguajes lo hacen por él, y además le permiten expresarse a un nivel más alto. Esta última característica, la de permitir al programador expresarse de una manera más clara y directa en la resolución de un problema específico dentro de un programa global, probablemente sea la principal aportación de los lenguajes inmersos especializados.

4.2. Características de los lenguajes anfitrión e inmerso

El concepto de lenguaje inmerso es bastante difuso y sobre todo muy amplio: se mueve en un extenso rango de posibilidades dependiendo fundamentalmente del grado de dependencia (en su sintaxis, interfaces de comunicación, traductores...) del lenguaje inmerso con respecto al anfitrión (o los anfitriones) para el (los) que se diseña. Para tomar posiciones sobre nuestra visión particular del tema, definimos de una manera genérica el concepto de lenguaje inmerso como aquel que tiene *estructura de lenguaje y se apoya, de alguna manera, sobre otro.* Es decir, exigimos la presencia de un analizador sintáctico y un deter-

minado traductor (llámese preprocesador, intérprete, compilador o como sea conveniente en cada caso) que resuelva las sentencias inmersas en código ejecutable dependiente del programa global escrito en lenguaje anfitrión. No nos interesa pues, en principio, que el lenguaje con el que se implementa el inmerso sea el mismo que el anfitrión, ni que las referencias entre ambos se resuelvan en tiempo de compilación, interpretación, linkage o ejecución, siempre y cuando, el programador posea las herramienta necesarias (y a poder ser, lo menos complicadas posible) para convertir en programas ejecutables aquellos que fueron escritos con este tipo de lenguajes.

Un modelo muy extendido de diseño de lenguajes inmersos es aquel que impone las restricciones que pasamos a comentar. Para que un lenguaje de programación admita la inmersión de otro de distinta naturaleza en su seno debe de cumplir una serie de requisitos mínimos:

- Poder invocar a procedimientos de librería (independientemente de que éstos se hayan escrito en este mismo lenguaje o en cualquier otro) y cederles el control.
- Permitir un paso de información (parámetros) a dichos procedimientos.

Por su parte, el lenguaje inmerso deberá implementarse con otro que permita comunicarse con el anterior permitiendo la toma y cesión de control, así como la comunicación de datos. Al mismo tiempo, deberá cuidarse en su implementación la separación de la parte específica propia del lenguaje inmerso, de la dependiente del lenguaje anfitrión para, de esta manera, facilitar la inmersión del lenguaje y su reutilización en otros entornos, con el menor número posible de modificaciones. En una palabra y en líneas generales, para el diseño de un lenguaje inmerso es necesario que se pueda establecer un interfaz de comunicación entre el lenguaje anfitrión y aquel con el que se codifica el inmerso.

Una vez desarrolladas estas cuestiones, pasemos a centrarnos en nuestro modelo de lenguaje inmerso. En nuestro caso, partimos del conocido sistema de bases de datos relacional DB3 al que pretendemos dotar de un conjunto de comandos de mayor potencia a los que posee originariamente [DOA92]. Para justificar el diseño de un lenguaje inmerso dentro de DB3, digamos que lo que se pretende es dotar a DB3 de una serie de comandos de inspiración profundamente relacional que permitan trabajar con tablas completas en vez de con registro individuales, como normalmente actúan los operadores de DB3. Estos comandos se pueden combinar entre sí adoptando una estructura de lenguaje, y tanto sintáctica como semánticamente se basan en el **álgebra relacional** de conjuntos [FER87], disciplina ésta que fundamenta teóricamente el modelo relacional de datos [ULL89] [GAR89], y que, con más bien poca fortuna, ha sido utilizada en la implantación de DB3.

Cada comando de este lenguaje inmerso se corresponde con un operador del álgebra relacional, y está implementado como una función escrita en Microsoft C almacenada en una librería que se ligará al programa ejecutable final durante el proceso de linkage. En resumen, y para centrar el marco de actuación, digamos que partimos del lenguaje anfitrión DB3, sobre el que actuará en modo inmerso un nuevo lenguaje llamado BAREL que ha sido implementado en C, y que además de ser breve y sencillo, aporta una serie de novedades basadas en el álgebra relacional que amplían la capacidad expresiva del lenguaje anfitrión.

5. BAREL: UN NUEVO LENGUAJE DE PROGRAMACIÓN.

Una vez introducidos en la estructura de DBASE III, del álgebra relacional y conociendo el significado del lenguaje, estamos en disposición de aplicar estos conceptos al diseño del lenguaje del que es objeto este trabajo. Una sentencia BAREL no es más que una fórmula compuesta por operadores que hemos citado (implementados aquí como funciones C) compuestos sobre una serie de ficheros DBF que sirven como operandos. El resultado de la fórmula se asigna siempre a una tabla resultado, y la sintaxis general de cualquiera de estas fórmulas (y por extensión, de todo el lenguaje) se puede describir brevemente con las reglas BNF que muestra la figura 3.

Cualquier fórmula de consulta escrita en BAREL se podrá insertar en un programa para Dbase III, precediéndola del símbolo «$», consiguiéndose así trabajar en modo inmerso como se hace con otros muchos lenguajes de manipulación de datos (SQL sobre Cobol, C, etc.). Antes de que el texto fuente original (con BAREL inmerso) sea compilado (utilizando Clipper, de Nantucket Corporation) y linkado, será preprocesado por la herramienta que hemos desarrollado para conseguir una traducción correcta y eficiente del mismo.

El siguiente ejemplo nos puede ilustrar sobre el uso de BAREL.

- **Hipótesis:** Sea una base de datos relacional constituida por relaciones (ficheros DBF) definidas de la forma siguiente (con estos esquemas):

PROVEEDOR (CÓDIGOP, NOMBRE, CIUDAD, CALLE, NÚMERO)
ARTÍCULO (CÓDIGOA, NOMBREA, PRECIO, DESCRIPCIÓN)
PEDIDO (CÓDIGOP, CÓDIGOA, CANTIDAD)

- **Enunciado:** Obtener el nombre de los proveedores que suministran el artículo 'A20'.

FRASE ::= IDEN = FÓRMULA
FÓRMULA ::= PROJECT [ESQUEMA] FACTOR | FACTOR RESTO | SELECT [EXPCON] FACTOR
FACTOR := (FÓRMULA) | IDEN
RESTO ::= - FACTOR | + FACTOR | * FACTOR | \ FACTOR | FACTOR | NJOIN FACTOR | JOIN [EXPSIN] FACTOR \ λ
ESQUEMA ::= IDEN | IDEN , ESQUEMA
EXPSIN ::= (EXPSIN) AND (EXPSIN) | (EXPSIN) OR (EXPSIN) | (EXPSIN) | IDEN OP IDEN
EXPCON := (EXPCON) AND (EXPCON) | (EXPCON) OR (EXPCON) | (EXPCON) | IDEN OP IDEN | IDEN OP NUM | IDEM OP LOG | IDEN OP TEXT | NUM OP IDEN | LOG OP IDEN | TEXT OP IDEN
OP := < | > | = | >= | <= | <>

SÍMBOLOS TERMINALES : = PROJECT SELECT [ ] ( ) - + * / \ NJOIN JOIN AND OR < > ,
Los terminales IDEN NUM LOG TEXT refieren a los identificadores y tipos numéricos, lógicos y texto de DBASE III.
SÍMBOLOS NO TERMINALES: FRASE FÓRMULA FACTOR RESTO ESQUEMA EXPSIN EXPCON OP

FIGURA 3.
Reglas gramaticales de BAREL.

- **Resolución:** El programa FILE.PRG, escrito en DBASE III y BAREL, obtendrá tras ejecutarse el fichero EJEMPLO.DBF con la información deseada:

```
$ temporal= (SELECT [codigoa='A20']pedido) NJOIN proveedor
$ ejemplo= PROJECT[ nombre] temporal
erase temporal.dbf
```

Como puede verse en este programa hemos descompuesto la única fórmula necesaria para conseguir nuestro objetivo, en dos subfórmulas, debido a que solo se aceptan frases que no rebasen una línea de programa. Sin embargo el primer fichero que a nosotros no nos podría interesar puede borrarse (sentencia DBASE erase) una vez que no sea necesario.

Si al traducir este programa deseáramos guardar el fichero de llamadas intermedias FILE$.PRG hallaríamos un texto de este estilo:

```
CALL_selec WITH "PEDIDO.DBF", "CODIGOA='A20'","XY_$_UVA.DBF»"
CALL_njoin WITH "XY_$_UVA.DBF","PROVEEDOR.DBF","TEMPORAL.DBF"
CALL_borra
CALL_proje WITH "TEMPORAL.DBF","NOMBRE","EJEMPLO2.DBF"
erase temporal.dbf
```

Analicemos cada una de las cinco líneas de este programa:

- *línea 1:* Obtención de un fichero temporal con todos los pedidos cuyo código de artículo sea 'A20'.
- *Línea 2:* Creación de un fichero con esta estructura:

TEMPORAL (CODIGOP,CODIGOA,CANTIDAD,NOMBRE,CIUDAD,CALLE,NÚMERO)

...donde se detallan el nombre, la ciudad, la calle y el número, además del código (CODIGOP) de todos los proveedores que suministran el artículo de código CODIGOA (que será 'A20' según la primera fórmula) en un volumen CANTIDAD.

Conviene repasar la función del operador NJOIN o reunión natural para comprender esta operación. Este operador produce un producto cartesiano entre los ficheros operandos (obteniéndose un archivo cuyo esquema sería CODIGOP, CODIGOA, CANTIDAD, NOMBRE, CODIGOP, CIUDAD, CALLE, NÚMERO), después una selección de registros en los que los campos repetidos coinciden (en este caso deben ser iguales las ocurrencias primera y segunda de los dos campos CODIGOP) y después elimina a base de proyecciones una sola columna (no las dos) correspondiente al campo (o los campos) repetidos (en este caso desaparece el segundo CODIGOP). El resultado pues, de esta operación es el que acabamos de describir.

- *Línea 3:* Eliminación del fichero XY_$_UVA.DBF (que no de TEMPORAL.DBF).
- *Línea 4:* La proyección del campo NOMBRE sobre el último fichero creado, generará el archivo con la información que deseamos.
- Línea 5: Borrando explícito mediante la sentencia erase de DB3 de un fichero no temporal (pues no es del tipo XY_$_UV.. como todos los intermedios generados y eliminados automáticamente por BAREL) que habíamos guardado anteriormente y que no nos es útil en este caso concreto (TEMPORAL.DBF).

6. EL PROCESO DE TRADUCCIÓN

El proceso global de traducción se describe en el diagrama de flujo de datos (DFD) de la figura 4 donde aparecen los procesos (círculos) y ficheros (entre líneas) involucrados en el mismo. La característica fundamental del esquema es la secuencialidad del proceso global, donde partiendo de un texto fuente, se alcanza un código ejecutable tras llevar a cabo estas cinco tareas y en ese orden: extracción, análisis, mezcla, compilación y linkage.

Ciñéndonos al gráfico de la figura 4, expliquemos uno por uno cada uno de los procesos que aparecen en el grafo y también las rutas que

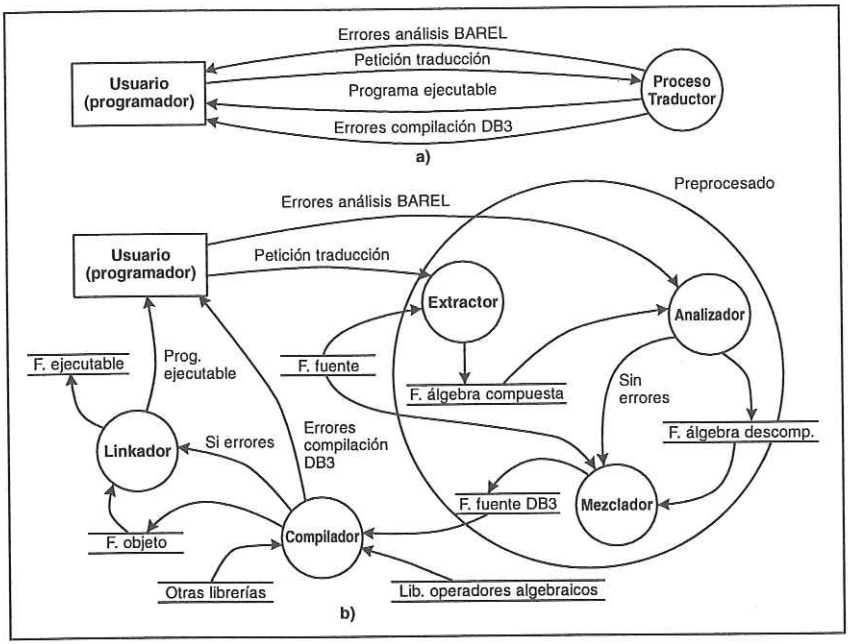

FIGURA 4.
Descripción del proceso de traducción. a) DFD de nivel 0; b) DFD de nivel 1.

los conectan. Primeramente se distinguen en el esquema cinco procesos y siete ficheros (de entre los cuales cinco de ellos son intermedios y dos como mínimo, siempre invisibles al usuario). En realidad existirá un proceso global (el lanzador, expresado como el rectángulo exterior que envuelve a toda la figura) que incluirá a todos los *subprocesos* y cuya misión será la de ir invocando secuencialmente a cada uno de ellos conforme les vaya tocando el turno.

6.1. Ficheros

El texto fuente.– Es un programa escrito en DB3+BAREL que, por tanto, presenta la particularidad de poder tener inmersas sentencias escritas en un lenguaje ajeno a DB3 (el lenguaje inmerso que ya hemos adelantado). Las frases en BAREL se distinguirán de las restantes (escritas en DBASE), aparte de por su discrepancia sintáctica con el resto del programa, por ir siempre precedidas de un símbolo especial: '$'.

El fichero intermedio con álgebra compuesta.– Donde tan solo existen frases escrita en BAREL, que pueden contener dentro de una misma fórmula varios operadores relacionales. Por ejemplo: $X =$

$U + (FICH - Y)$. Este fichero nunca será recuperable una vez terminada la traducción.

El fichero intermedio con álgebra descompuesta.– Asociado con el anterior, este fichero contendrá las mismas fórmulas que el primero, pero descompuestas en subfrases con un solo operador cada una de ellas. Los resultados parciales de operaciones intermedias se guardarán en ficheros cuyo nombre será del tipo *"XY_$_UVi.DBF"* donde $'A' < i < 'Z'$, pues como máximo, cada frase original puede descomponerse en otras 24 subfrases (con los 24 ficheros intermedios que ello comporta). Así pues, la frase dada en el ejemplo anterior, se descompondrá de esta manera:

$$XY\_\$\_UVA = FICH - Y$$

$$X = U + XY\_\$\_UVA$$

Realmente las subfrases algebraicas que aparecen en este fichero (que al igual que el anterior será siempre totalmente invisible e irrecuperable por el usuario) no tendrán ese formato tan concreto, sino que estarán enmascaradas en *«quintentos»* reconocibles por el analizador sintáctico. De tal modo que si hiciésemos un listado del mismo (suponiendo que pudiésemos recuperarlo antes de que el traductor lo borre automáticamente cuando deje de utilizarlo), no hallaríamos información legible en él. Es únicamente un fichero temporal que siempre se pasará al mezclador para que éste, una vez haya interpretado los quintetos que se listan en él, pueda sustituir las frases algebraicas por sentencias CALL.

Librería de operadores algebraicos.– En ella aparecen procedimientos de dos categorías: operadores algebraicos y funciones de apoyo. Éstas últimas únicamente pueden ser referenciadas desde los procedimientos algebraicos y nunca desde el exterior aunque, en realidad una de ellas, la rutina _BORRA, si puede invocarse (y de hecho así se hace) desde los programas DB3, ya que su misión es la de eliminar en tiempo de ejecución todos los ficheros intermedios del tipo XY_$_UV que se hayan generado durante el proceso de actuación de los procedimientos algebraicos. Las funciones C que implementan todos y cada uno de los operadores algebraicos (unión, producto, proyección, ...) dependen fuertemente del formato de las tablas DB3 sobre las que actúan, de tal modo que en caso de que se quisiera introducir a BAREL en un nuevo entorno de inmersión tendríamos que remodelar todas ellas.

Fichero intermedio con sentencias en DB3 y sentencias CALL.– Posee las sentencias originales del texto fuente escritas en DB3, pero ya han desaparecido las fórmulas algebraicas escritas en BAREL, que han quedado sustituidas por llamadas a los procedimientos que ejecu-

tan las operaciones algebraicas comentados anteriormente (habrá una llamada por cada subfórmula simple que compone cada frase compuesta, y en el mismo orden que se indicó en los ejemplos anteriores) conservándose el orden de llamadas dentro del texto total, tal y como venía en el original. Asimismo, tras la secuencia de llamadas que genera una frase compuesta, se eliminarán (en tiempo de ejecución, que no de traducción, naturalmente) todos los archivos parciales del tipo "XY_$_UVi.DBF" que existan o que se hayan generado tras cada una de las llamadas CALL hacia los procedimientos algebraicos. Se consigue así eliminar *basura* en el disco, al tiempo que se niega el acceso al usuario hacia estos archivos.

Este fichero de llamadas puede conservarse si el usuario lo desea (en un texto ASCII idéntico a su programa original pero con las fórmulas BAREL resueltas) para lo cual únicamente tendrá que invocar al traductor de un modo concreto.

Fichero objeto.– Es el resultando de compilar el fichero anterior con CLIPPER y en este caso también se da la opción de guardar este archivo utilizando un nuevo conmutador al invocar al traductor de BAREL.

Fichero ejecutable.– Se obtiene tras el linkado del fichero anterior con la librería de operadores algebraicos y algunas otras (Clipper, LLIBCE, ...).

Los nombres de los tres últimos ficheros comentados, nunca son fijos ya que el usuario puede renombrarlos (o simplemente, por omisión, dejarlos como el fuente pero con distinta extensión) siempre que éste desee conservarlos.

6.2. Procesos

Extractor.– Este proceso tiene como misión extraer solo las frases BAREL del texto fuente (escrito en DB3+BAREL), precedidas por un "$", para reescribirlas en el primer fichero intermedio del que hablábamos en el apartado anterior (texto con álgebra compuesta).

Analizador o traductor.– Este programa recogerá una por una todas las frases que se listan en el primer fichero intermedio y les hará un análisis léxico-sintáctico para comprobar si se adecuan a las reglas gramaticales de la figura 3. Si es así, se genera código para ellas en forma de subfórmulas algebraicas con un único operador cada una de ellas con aspecto de quintetos (resultado, operando primero, operando segundo, operador y expresión o esquema). Estas tres operaciones (análisis léxico, sintáctico y generación de pseudocódigo en forma de quintetos) se llevan a cabo conjuntamente. Así pues, la generación de

código se simultanea con el análisis, y solo cesa cuando aparezca al menos un error en alguna fórmula.

Mezclador.– Genera un fichero con el mismo contenido que el texto fuente original, con la particularidad de que las frases algebraicas ya han sido sustituidas por sentencias CALL, permitidas en DB3 compilable.

Compilador CLIPPER.– Este es un programa no generado por nosotros (registrado por Nantucket Company como ya señalamos anteriormente) pero que evidentemente debe estar presente siempre que intentemos lanzar nuestro traductor para compilar el texto DB3 con sentencias CALL hacia los procedimientos algebraicos.

Linker.– Qué podrá ser PLINK86 (el ligador que casi siempre acompaña a CLIPPER), o en su defecto el que proporcione el sistema. Con él se resolverán las referencias que se hagan en lo programas hacia las librerías algebraicas (aparte de otras que puedan aparecer en las librerías Clipper, LLIBCE, ...).

Lanzador o proceso traductor.– El usuario de esta aplicación solo tendrá que invocar a este programa general que ejecutará todas las fases de traducción, una por una, del modo que hemos descrito anteriormente. El programa acepta una serie de opciones (mediante *conmutadores* o *switches* de entrada) que le indicarán que ficheros intermedios deben conservarse (el fichero DB3 con llamadas, el texto objeto, y el archivo de mapeo) al tiempo que informan sobre si la *«ruta»* de compilación debe concluirse entera, o simplemente debe de cortarse en algún tramo específico (por ejemplo, a veces puede interesar llegar hasta el archivo objeto y no hasta el ejecutable). De esta manera si se desea llevar a cabo todo el proceso de traducción, y no se producen errores en ninguna de las fases que se ejecutan (por ejemplo, un fallo típico es la detección de un error sintáctico durante el análisis, o la pérdida de un fichero ejecutable que no podrá lanzarse) desde el exterior, el usuario obtendrá un fichero ejecutable que proviene de uno fuente, pero no podrá listar los diversos ficheros intermedios generados en cualquier fase del desarrollo de la traducción, porque habrán sido borrados previamente al no haberse especificado expresamente la conservación de alguno o algunos de ellos. Aunque se produzca algún error en cualquiera de las diferente etapas citadas, el usuario seguirá sin poder acceder a lo ficheros parciales, pues habrán sido borrados por el lanzador al tomar éste el control desde el subproceso que terminó erróneamente (adelantándose de esta manera a la voluntad del usuario que intente recuperarlos).

De este modo, el relativamente complicado proceso de traducción (que puede resumirse en preproceso, compilación y linkage) que hemos explicado será entendido como una herramienta que permite de

una sola pasada convertir textos fuente escritos en un lenguaje anfitrión (DB3) combinado con otro inmerso (BAREL), en programas ejecutables.

7. CONCLUSIONES Y TRABAJOS FUTUROS

En definitiva, con una vocación doblemente formal (científica) y pragmática (ingeniería), hemos pretendido diseñar un lenguaje para la gestión de sistemas de información, capaz de resolver problemas concretos con técnicas puramente teóricas. Hemos justificado la presencia en determinadas situaciones de los lenguajes inmersos como herramientas de manejo sencillo y alta capacidad computacional en aquellas áreas informáticas para las que se diseñan. BAREL cumple esta características y ataja de una manera rápida y eficaz ciertas dificultades de otro (DB3) en el contexto de las bases de datos relacionales.

El diseño y la implementación del lenguaje ha sido bastante independiente del lenguaje anfitrión sobre el que aparece inmerso. Indirectamente y gracias a esta característica de independencia, BAREL se nos presenta como un lenguaje potencialmente válido para ser aplicado (de nuevo como lenguaje inmerso) sobre entornos distintos a DB3. Actualmente estamos trabajando en un nuevo sistema de traducción que permita la inmersión de BAREL en el contexto del lenguaje C, con el objetivo de conseguir que éste último adquiera las propiedades que aporta BAREL (al introducir la potencia de los lenguajes basados en el álgebra relacional) en el terreno específico de la manipulación de datos, trabajando en este caso sobre ficheros C (que posean estructuras semejantes a las tablas de una base de datos relacional).

También es están abordando aspectos teóricos y prácticos para el desarrollo de una versión paralela de BAREL[mor93b]. En este sentido, se pretenden implementar algoritmos paralelos para todos y cada uno de los operadores algebraicos, así como dotar al lenguaje de herramientas que permitan al programador expresar composiciones paralelas de fórmulas BAREL.

BIBLIOGRAFÍA

[AHO86] AHO, A. V.; SESTHI, R.; ULLMAN, J. D.: *Compilers: principles, tecnique and tools*. Addison-Wesley (1986).

[DAT88] DATE, C. J.: *A guide to the SQL standard.* Addison-Wesley (1988).

[DOA92] DOAKE, J.; BRITTON, C.; MITCHELL, R.: *Strange bedfellows: abstract data types and Dbase III.* Information and Software technology, Febrero 1992, volumen 34, número 2.

[FER87] FERNÁNDEZ BAIZÁN, C.: *El modelo relacional de datos: de los fundamentos a los modelos deductivos.* Ediciones Díaz de Santos S.A. (1987).

[GAR89] GARDARIN, G.; VALDURIEZ, P.: *Relational databases and knowledge bases.* Addison-Wesley (1989).

[MOR93a] MORENO, G.; GÁMEZ, J. A.; CUARTERO, F.: *BAREL: Un lenguaje para la manipulación de datos basado en el Álgebra Relacional.* Segundas Jornadas de Ingeniería de Sistemas Informáticos y de Computación. Quito, 12-16 de Abril de 1993.

[MOR93b] MORENO, G.; GONZÁLEZ, P.; GÁMEZ, J. A.: *Parallel algorithm for table division in relational databases.* Euromicro'93. Barcelona, 6-9 de Septiembre de 1993.

[TRE85] TREMBLAY, J. P.; SORENSON, P. G.: *The theory and practice of compilar writting.* MacGraw-Hill, Inc. (1985).

[ULL89] ULLMAN, J. D.: *Principles of databae and knowledge-base systems,* volumen II. Computer Science Press (1989).