DISEÑO DE UN LENGUAJE PARA BASES DE DATOS FUNCIONALES

José de Jesús Pérez Alcázar
Alberto Henrique Frade Laender
Roberto da Silva Bigonha
Departamento de Ciência da Computação
Minas Gerais - *Brasil*

RESUMEN

Este artículo describe las principales características de un lenguaje llamado LBF (Lenguaje para Bases de Datos Funcionales) el cual es un lenguaje auto-contenido para el manejo de bases de datos funcionales. LBF es una extensión del lenguaje DAPLEX definido por Shipman. Entre las extensiones podemos enumerar la inclusión de comandos de entrada y salida, comandos condicionales, operandos aritméticos y lógicos y comandos para modificar el esquema.

1. INTRODUCCION.

En los últimos años, varios modelos de datos han sido propuestos con el propósito de capturar mas semántica en los datos y remover varias de las restricciones impuestas por los llamados modelos clásicos (jerárquico, de red y relacional). Estos modelos, denominados modelos semanticos [4,15], fueron diseñados para ofrecer mecanismos de modelaje mas ricos y expresivos, permitiendo la descripción de la estructura de una base de datos de una forma más clara y precisa [3,7,8,18,21].

Entre los modelos de datos semánticos descritos en la literatura se destaca, por su simplicidad, el modelo funcional [6,9,12,19,27]. De las propuestas sobre el modelo funcional existentes, algunas incorporan lenguajes de manejo de datos a través del uso de funciones y conjuntos de operadores. e estas, solamente las propuestas de Shipman [19] y Buneman & Frankel [6] integran operaciones de manejo de datos con operaciones de propósito general en un solo lenguaje. Esta es una característica importante ya que en general los modelos de datos semánticos propuestos en la literatura facilitan solamente el modelaje de las propiedades estáticas de las aplicaciones (estructuras), prestando poca atención a los aspectos dinámicos (operaciones) [15].

Seguramente, el modelo de Shipman [19] es el más destacado de los modelos funcionales. Este modelo utiliza varios de los recientes resultados provenientes de estudios en el area de modelaje de datos, lo cual convierte a DAPLEX, su lenguaje de definición y manipulación de datos, en un lenguaje bastante rico semanticamente.

El propósito de este artículo es describir un lenguaje del tipo de DAPLEX que está siendo desarrollado en el Departamento de Ciencias de la Computación de la Universidad Federal de Minas Gerais. Este lenguaje, llamado LBF (Lenguaje para Bases de Datos Funcionales), es una extensión de DAPLEX, en el cual fueron incluidas algunas construcciones propuestas por KulKarni y utilizadas en EFDM [13], además de facilidades para operaciones de E/S y comandos condicionales.

A seguir el artículo se divide en cuatro secciones. En la sección 2 describimos el modelo de datos funcional y los motivos de su selección para el desarrollo de este trabajo. En esta sección también presentamos una descripción de la propuesta de Shipman. En la sección 3 describimos LBF, sus estructuras y operaciones y damos un pequeño ejemplo de su utilización. En la sección 4 describimos el estado actual del trabajo y mostramos algunas de las orientaciones para estudios futuros. En la sección 5 presentamos algunas conclusiones. Por último anexamos un apéndice con la sintaxis de LBF.

2. EL MODELO DE DATOS FUNCIONAL.

El modelo funcional nace como resultado de la busqueda de nuevas formas de modelaje que sirvan de base común o formalismo en la descripción y comparación de los tres modelos clásicos [20]. Este formalismo debería ofrecer una base unuficada para el diseño de esquemas tanto relacionales como de red, sin anomalías de actualización.

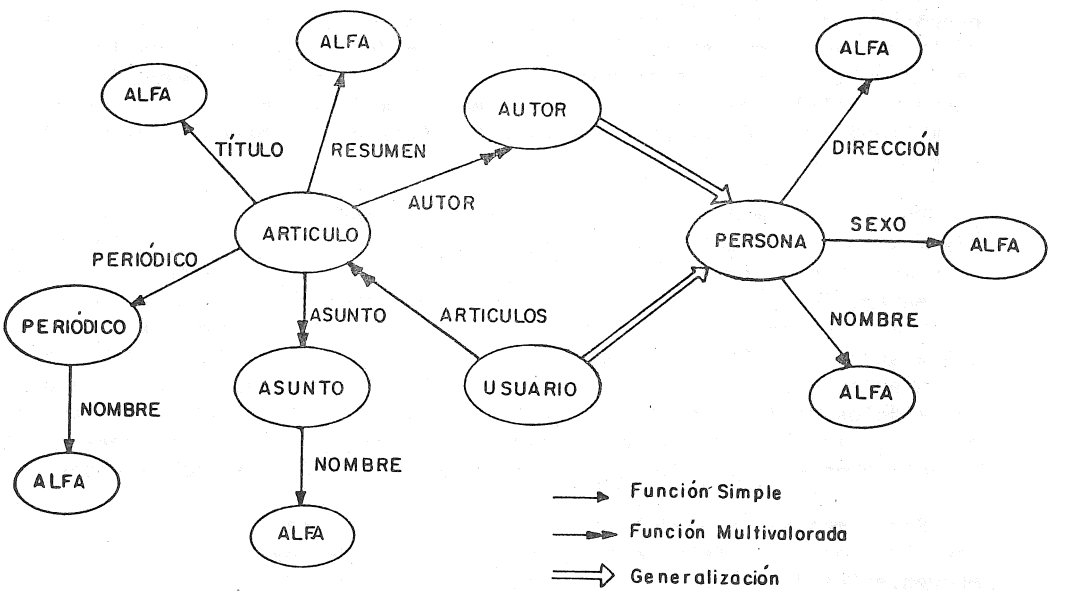

Fig. 1

El modelo funcional tiene sus bases en el concepto matemático de función y utiliza dos tipos de primitivas (estructurales) para el modelaje del mundo real [13]: los conjuntos de objetos y las funciones que son aplicadas a estos para relacionarlos entre si. Los conjuntos de objetos estan divididos en conjuntos de entidades y conjuntos de valores. La principal diferencia entre un conjunto de valores y un conjunto de entidades es que el primero

nunca es parte del dominio de una función. Los conjuntos de valores son llamados por Kulkarni [13] conjuntos de entidades predefinidas (ej: conjuntos de enteros, reales, etc). Las funciones pueden ser totales o parciales, y la distinción corresponde a una restricción [11] sobre el número de elementos del conjunto (dominio) que deben participar del relacionamento funcional.

La figura 1 muestra la descripción gráfica, de acuerdo con el modelo funcional, del esquema conceptual de una base de datos para una biblioteca personal de artículos publicados en revistas (periódicos). Esta base de datos será usada en la mayoría de nuestros ejemplos.

2.1. El Modelo de Datos Funcional de Shipman

Como fue dicho anteriormente, el modelo funcional mas destacado es el modelo propuesto por Shipman [19], el cual esta incluido en el lenguaje DAPLEX. Su objetivo primordial es ofrecer un lenguaje conceptualmente natural que sirva de interfaz en la utilización de una base de datos. Las construcciones básicas del lenguaje DAPLEX son las entidades que son usadas para modelar los objetos de la aplicación y las funciones que son usadas para modelar sus propiedades. El aspecto mas interesante del lenguaje DAPLEX es su simplicidad aliada a construcciones de modelaje semanticamente poderosas.

Entre las características principales del lenguaje DAPLEX podemos enumerar las siguientes:

1) Las funciones pueden tener cero o más argumentos. Las funciones sin argumentos son utilizadas para modelar entidades o objetos del mundo real y las funciones con argumentos son utilizadas para modelar atributos de entidades y relacionamentos entre estas.

2) Las funciones pueden ser clasificadas en: simples (retornan como resultado una entidad) y multivaloradas (retornan un conjunto de entidades).

3) Existe una diferencia entre entidad y su identificación externa o llave (orientación a objetos).

4) Las entidades puedem ser organizadas en una jerarquía de tipos [21,25], donde atributos y relacionamentos de los supertipos son automáticamente heredados por cada uno de los subtipos correspondientes.

5) A partir de funciones básicas es posible crear funciones derivadas, es decir, el concepto de información derivada [25] es soportado de forma simple y natural.

2.2. Una Validación del Modelo Funcional

Pero porqué el modelo funcional es una herramienta atrayente para el modelaje conceptual ?. Kerschberg & Pacheco [12], Kulkarni [13] y Orman [16] enumeran algunas de estas razones:

1) Los modelos funcionales ofrecen un ambiente de modelaje semanticamente rico. El hecho de no diferenciar entre datos y programas (información derivada) facilita bastante el modelaje conceptual.

2) Los lenguajes de consulta pueden ser especificados de una forma más natural. Además, facilidades de manejo de datos pueden ser integradas naturalmente con operaciones de propósito general (FQL, el lenguaje definido por Buneman & Frankel [6], es un ejemplo).

3) Existen algoritmos para mapear la estructura de una base de datos de red (CODASYL) o relacional para una funcional [12]. Como consecuencia, el modelo de datos funcional puede ser usado como mecanismo de integración de bases de datos hetereogeneas [22].

4) El modelo funcional representa la información a través de hechos atómicos (funciones) [16], eliminando algunos problemas de redundancia y evitando anomalias en la actualización.

5) La teoría matemática de funciones puede ser explorada para ofrecer una base teórica sólida para bases de datos.

6) La función segun Orman [16], es una estructura más primitiva y poderosa que otras comunmente usadas por los modelos ya conocidos, como relaciones, archivos, conjuntos DBTG y arreglos. A partir de funciones, consultas y restricciones son naturalmente implementadas, ya que consultas son funciones aplicadas a valores de entrada y que retornan valores de salida, al mismo tiempo que restricciones son predicados que corresponden a funciones cuyo codominio es el conjunto de valores lógicos.

3. DESCRIPCION DEL LENGUAJE LBF

LBF es un lenguaje interactivo, con el cual el usuario puede crear, operar y actualizar su base de datos. Este lenguaje, como el lenguaje EFDM [13], es auto-contenido, es decir no necesita ser enbutido en un lenguaje de propósito general como sugiere Shipman [19] al definir DAPLEX. LBF es básicamente una extensión del lenguaje DAPLEX, al cual le fueron adicionados algunas construcciones utilizadas en EFDM además de comandos condicionales y de E/S. Originalmente estos comandos fueron definidos en portuguès, pero versiones en otros idiomas pueden ser implementadas via procedimientos de instalación de software. En este articulo adoptaremos una versión con palavras claves en Español. Una descripción detallada de la sintaxis de LBF se encuentra en el apendice y ejemplos sobre su utilización lo mismo que una descripción de su semantica puede ser encontrada en [17].

De manera general, para el modelaje del mundo real son necesarios dos tipos de mecanismos [25]: un mecanismo que permita la descripción de las propiedades estáticas del mundo real, es decir, sus estructuras y las restricciones definidas sobre ellas; y un mecanismo que permita la descripción de las propiedades dinámicas, es decir, las operaciones aplicadas a las estructuras. A continuación describimos como estos mecanismos son implementados en LBF.

3.1 Estructuras

Al igual que DAPLEX, LBF modela las entidades del mundo real y sus propiedades a través de funciones. La declaración de estas funciones es realizada por medio del comando DECLARE. Las funciones pueden tener cero o mas argumentos, siendo que funciones sin argumentos son utilizadas para modelar entidades, mientras que funciones con argumentos son utilizadas para modelar propiedades y relacionamientos. Por ejemplo, considerando la base de datos de la Figura 1,

```
DECLARE ARTICULO () ->> ENTIDAD
```

declara un conjunto de entidades con propiedades comunes (tipo de entidad) llamado ARTICULO y

```
DECLARE TITULO(ARTICULO) -> ALFA(20)
DECLARE DETALLES(ARTICULO) -> ALFA(100)
```

declara sus propiedades (atributos) TITULO como una cadena de 20 caracteres y DETALLES como una cadena de 100 caracteres.

LBF como otros lenguajes basados en el modelo funcional, maneja conjuntos de entidades y conjuntos de valores. Estos ultimos, también llamados de entidades predefinidas, pueden ser clasificados en conjuntos de valores alfanuméricos, numéricos (los cuales pueden tener parte decimal) y lógicos. Los conjuntos de entidades representan objetos del mundo real, pero no son identificadores, es decir, números o nombres que identifican estos objetos externamente, por este motivo no pueden ser utilizados en comandos de E/S. Las funciones son también clasificadas de acuerdo con su resultado, si retornan un conjunto de entidades son llamadas de funciones multivaloradas y si retornan una única entidad son llamadas de funciones simples.

Otra característica de LBF heredada de DAPLEX es la utilización de jerarquías de generalización [21,25]. De esta manera, por ejemplo,

```
DECLARE AUTOR() ->> PERSONA
```

declara un AUTOR como un subtipo de PERSONA, lo que indica que cualquier entidad del tipo AUTOR es también una entidad del tipo PERSONA, y que todas las propiedades y relacionamientos definidos para el tipo PERSONA son auto-

maticamente heredados por el tipo AUTOR.

LBF también permite la definición de funciones derivadas [19] a partir de funciones básicas declaradas a través del comando DECLARE. Las funciones derivadas son definidas por medio del comando DEFINA. Por ejemplo, dada la función

AUTOR(ARTICULO) ->> AUTOR

podemos definir la función derivada (inversa)

DEFINA ARTICULO(AUTOR) ->> INVERSA DE AUTOR(ARTICULO)

que retorna como resultado la lista de articulos publicados por un determinado autor.

Desde el punto de vista estructural, LBF presenta las siguientes diferencias con respecto a DAPLEX:

- Al contrário de la propuesta de Shipman, las funciones simples son por definición consideradas parciales. Esto genera dos ventajas: es posible introducir en la base de datos objetos cuyos datos son incompletos y las funciones inversas pueden ser definidas libremente.

- Shipman permite que argumentos de funciones sean expresiones arbitrárias. La razón para esto radica en el problema de manejar funciones con multiples argumentos. En este caso, si una función utiliza tipos de entidad como argumentos, para algunas combinaciones de estos argumentos la función puede no estar definida. Por ejemplo, en la declaración

DECLARE NUMARTICULOS(AUTOR, PERIODICO) -> NUMERICO(2)

pueden existir pares autor-periódico para los cuales la función no estaría definida, violando el hecho de que las funciones deben ser totales. Como en LBF esta restricción no existe, todos los argumentos deben ser del tipo entidad, lo que simplifica la sintaxis de la definición de funciones.

3.2. Operaciones

Desde el punto de vista de las operaciones, LBF presenta diferencias sensibles en relación a DAPLEX, que son tratadas a continuación.

3.2.1. Operaciones de Selección y Recuperación de Datos

DAPLEX utiliza expresiones y comandos como construcciones básicas para el manejo de datos. Expresiones aparecen siempre dentro de comandos y representan tanto un conjunto de entidades como una sola entidad, lo que permite

clasificarlas como expresiones de conjunto y expresiones simples respectivamente. Las expresiones en general tienen tres caracteristicas: un valor, un papel y un orden. El valor de la expresión es el conjunto de entidades (o la entidad) retornado como resultado de su evaluación. El papel de una expresión está relacionado con el tipo en el cual las entidades del conjunto son interpretadas. El orden está asociado con expresiones de conjunto y representa el orden establecido entre las entidades del conjunto. LBF utiliza la sintaxis de DAPLEX para la especificación de un orden parcial de un conjunto (Vea apêndice).

En lo que respecta a los comandos, existen algunas diferencias entre LBF y DAPLEX. la mas importante es la utilización por parte de LBF de un comando condicional. El comando condicional fue definido para facilitar operaciones (consultas o actualizaciones) dependientes de una condición, evitando asi, que estas sean desmembradas. Por ejemplo, considerando la base de datos de una empresa, la operación de actualización "Aumentar en 25% el salario de todas las personas del Departamento de 'Sistemas y computación' y en 15% el salario de los demás empleados" seria expresada en LBF de la siguiente manera:

```
PARA CADA EMPLEADO

    SI NOMBRE(DEPART(EMPLEADO)) = 'Sistemas y Computación' ENTONCES

      SEA SALARIO(EMPLEADO) = 1.25 * SALARIO(EMPLEADO)

    SINO

      SEA SALARIO(EMPLEADO) = 1.15 * SALARIO(EMPLEADO)

    FIN

FIN
```

El comando SI también es utilizado en la evaluación de expresiones simples y de conjunto. Esta forma de utilización es bastante útil en la definición de funciones derivadas. Por ejemplo,

```
DEFINA POTENCIA (X EM NUMERICO(5), J EM NUMERICO(1)) ->

                SI J = 0 ENTONCES 1

                SINO X * POTENCIA(X,J - 1)

                FIN
```

define la función POTENCIA que calcula el valor de X elevado a la J-ésima potencia.

El primer tipo de comando SI es siempre utilizado dentro de un comando

PARA. Por tanto, ninguna operación puede comenzar con el comando SI. Esta restricción fue impuesta para facilitar la especificación de las operaciones, sin la necesidad de disminuir el poder del lenguaje, ya que cualquier operación que es especificada a partir del comando SI puede igualmente ser definida con la utilización del comando PARA. Una descripción detallada del comando condicional puede ser encontrada en [17].

En lo que se relaciona a los operadores utilizados en las expresiones de conjunto, LBF, al contrario de DAPLEX, provee los operadores UNION, INTERSECCION y DIFERENCIA. Los cuales son utilizados en la unión, intersección y diferencia de expresiones de conjuntos respectivamente. En DAPLEX estos operadores son utilizados exclusivamente en la definición de funciones derivadas. Como operadores sobre expresiones simples LBF utiliza los operadores lógicos Y, O y NO, los operadores aritméticos "+", "-", "*", "/" y RES (módulo), y el operador "&" para la concatenación de cadenas de caracteres.

3.2.2. Operaciones de Entrada y Salida de Datos

Para las operaciones de entrada de dados, LBF utiliza el concepto de ENTRADA que equivale a una "entidad externa" cuya función es facilitar el almacenamiento de grandes volumenes de datos a partir de archivos de entrada. Por ejemplo, para modificar la dirección de algunos usuarios de la biblioteca personal (ver Figura 1), bastaria con crear una entidad externa INFORMACION (con sus respectivos atributos) de la siguiente manera,

```
DECLARE INFORMACION() ->> ENTRADA

DECLARE NOMBRE(INFORMACION) -> ALFA(20)

DECLARE DIRECCION(INFORMACION) -> ALFA(30)
```

y ejecutar los comandos

```
PARA CADA INFORMACION

    PARA EL USUARIO TAL QUE

                NOMBRE(USUARIO) = NOMBRE(INFORMACION)

        SEA DIRECCION(USUARIO) = DIRECCION(INFORMACION)

    FIN

FIN
```

Para la salida de datos, LBF utiliza el comando ESCRIBA (IMPRIMA para emitir reportes por la impresora), que posibilita la generación de reportes simples en un formato predefinido, pero con encabezados, rompimiento por diferentes campos y ordenados acendente o descendentemente. Por ejemplo,

dada la consulta, "Liste los artículos (y sus características) prestados a todos los usuarios que son también autores". En LBF esta consulta es expresada de la siguiente manera:

```
PARA CADA X EN (USUARIO INTERSECCION AUTOR)
     EN ORDEN ACENDENTE POR NOMBRE(X)
     PARA CADA ARTICULO(X) EN ORDEN ACENDENTE POR TITULO(ARTICULO(X))
        CONSIDERE
        ENCABEZADO "REPORTE DE ARTICULOS PRESTADOS".
                   "A USUARIOS QUE SON AUTORES"
        ROMPIMIENTO POR NOMBRE(X), "NOMBRE DEL USUARIO:", NOMBRE(X) 'AP'
        ESCRIBA(TITULO(ARTICULO(X)): "TITULO",
                AUTOR(ARTICULO(X)): "AUTORES",
                PERIODICO(ARTIGO(X)): "PERIODICO DONDE FUE PUBLICADO",
                ASUNTO(ARTICULO(X)): "ASUNTOS RELACIONADOS")
     FIN
FIM
```

Los campos que conforman el reporte siempre seran impresos en la forma de columnas sin importar si la salida pedidad exceda el ancho de la página. Por ejemplo, el reporte anterior sería presentado en el terminal de la siguiente manera:

```
12 Abr. 1982                                                    Pag.01

                        REPORTE DE ARTICULOS PRESTADOS

                          A USUARIOS QUE SON AUTORES

                                    TITULO

          AUTORES  PERIODICO_DONDE_FUE_PUBLICADO  ASUNTOS_RELACIONADOS

XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

          YYYYYY   ZZZZZZZZZZZZZZZZZZZZZZZZZZZZ     WWWWWWWWWWWWWWWWW
```

```
        YYYYYY                                          WWWWWWWWWWWWWWWWWWW

        YYYYYY

   XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

        YYYYYY   ZZZZZZZZZZZZZZZZZZZZZZZZZ   WWWWWWWWWWWWWWWWWWWW

                                             WWWWWWWWWWWWWWWWW

                                             WWWWWWWWWWWWWW

    .....  .......  ............................  ......................
```

Una discusión más detallada de los comandos ESCRIBA e IMPRIMA esta por fuera del alcance de este articulo y puede ser encontrada en [17].

3.2.3. Operaciones de Actualización de la Base de Datos

En lo que respecta a las operaciones de actualización, LBF posee las siguientes características:

- Debido a que las funciones son por definición parciales, una función simple no necesita ser inicializada en el momento de su creación. Sin embargo, para facilitar la identificación de una entidad externamente, por lo menos una de sus funciones (o un grupo) debe ser inicializada en el momento de su creación.

- En LBF es posible incluir una entidad existente en la base de datos dentro del conjunto de entidades de un determinado tipo. Para esto, es utilizada la misma construcción que es usada para la inclusión de entidades en el conjunto correspondiente al codominio de una función multivalorada. Por ejemplo,

```
INCLUYA USUARIO = (EL AUTOR TAL QUE
                          NOMBRE(AUTOR) = "Shipman")
```

incluye el autor de nombre "Shipman" en el conjunto de usuarios de la biblioteca personal.

- De la misma forma es posible excluir una entidad del conjunto correspondiente a un determinado tipo, lo que resulta en la exclusión de todas las referencias a esta entidad por parte de los conjuntos correspondientes a todos sus subtipos. Por ejemplo,

```
EXCLUYA USUARIO = (EL AUTOR TAL QUE
```

NOMBRE(AUTOR) = "Shipman")

excluye el autor de nombre "Shipman" del conjunto de usuarios de la biblioteca personal, no obstante el permanece como autor.

- Entidades pueden ser removidas de la base de datos a través del comando REMUEVA. De esta manera,

REMUEVA EL AUTOR TAL QUE NOMBRE(AUTOR) = "Shipman"

resulta en la completa eliminación del autor de nombre "Shipman", de la base de datos y de todas sus referencias en los conjuntos correspondientes a sus subtipos y supertipos (es decir son eliminadas todas sus instancias).

3.2.4. Operaciones sobre el Esquema

De acuerdo con la propuesta de Shipman, sobre un esquema DAPLEX solamente se permite la adición de nuevas funciones. LBF aprovecha las ideas de Kulkarni [13] y ofrece un nuevo operador, REMOVA, para la eliminación de funciones que el usuario ya no desea mantener en la base de datos. Por ejemplo,

REMUEVA DIRECCION(PERSONA)

causa la eliminación de la función DIRECCION y de todas las funciones definidas a partir de ella.

3.3. Almacenamiento y Encapsulamiento de Consultas

En el lenguaje LBF, las consultas pueden ser encapsuladas, a través de la palabra clave CONSULTA, e identificadas por un nombre para una referencia posterior. Por ejemplo, la consulta

```
CONSULTA MUJERES:

  PARA CADA PERSONA TAL QUE SEXO(PERSONA) = "F"

       ESCRIBA(NOMBRE(PERSONA))

  FIN

FIN
```

puede ser referenciada y ejecutada en cualquier momento a través de la invocación de su nombre. Además, la consulta puede también ser eliminada con la utilización del comando REMUEVA.

El encapsulamiento es muy útil en el caso de consultas muy usadas y tiene la ventaja de una mayor velocidad de ejecución, ya que el código

generado puede ser almacenado, evitando su compilación cada vez que la consulta es ejecutada.

3.4. Restricciones

La propuesta de Shipman para la especificación de restricciones en DAPLEX incluye el uso de dos construcciones CONSTRAINT y TRIGGER [19]. Estas construcciones, no obstante, no son completas. LBF no soporta estos mecanismos de restricción. Un trabajo posterior deverá introducir la definición de um mecanismo más general para la especificación de restricciones, que pueda ser adicionado posteriormente al lenguaje. No obstante, LBF permite la especificación de restricciones básicas, tales como:

Restricciones de totalidad. Son utilizadas en el caso en que todo objeto perteneciente a un tipo de entidad este siempre asociado a otro objeto en la base de datos. Por ejemplo,

```
DECLARE AUTOR(ARTICULO) ->> AUTOR TOTAL
```

indica que un articulo debe siempre estar relacionado a un autor.

Restricciones sobre la cardinalidad. Las palabras MAXIMO y MINIMO son utilizadas para restringir el número de elementos de los tipos de entidad y de las funciones multivaloradas. Por ejemplo,

```
DECLARE ARTICULOS(USUARIO) ->> ARTICULO MAXIMO 3
```

indica que el número máximo de articulos prestados a un usuario de la biblioteca personal no puede ser mayor que 3.

Restricciones sobre los valores de las funciones. Ciertas funciones pueden ser restringidas de manera que sus valores no puedan ser alterados. Por ejemplo,

```
DECLARE TITULO(ARTICULO) -> ALFA(30) FIJO
```

indica que después de ser atribuido algún valor a este campo (título) por la primera vez este no puede ser alterado nuevamente.

Restricciones sobre la identificación de entidades. En una base de datos los usuarios pueden estar interesados en diferenciar entidades individuales de forma que ellas puedan ser identificadas sin ambiguedad. Por ejemplo,

```
DECLARE NOMBRE(PERSONA) -> ALFA(20) UNICO
```

indica que dos personas no pueden tener el mismo nombre.

4. ESTADO ACTUAL DEL TRABAJO Y ORIENTACIONES PARA ESTUDIOS FUTUROS

LBF está siendo en este momento implementada para microcomputadores compatibles con el IBM PC, teniendo como soporte para el manejo de archivos el sistema Btrieve [23]. Para la implementación del traductor de LBF será utilizado el sistema de implementación de compiladores SIC [1].

Después de esta implementación inicial, nuestro trabajo consistirá en adicionar algunos mecanismos que hasta el momento no fueron considerados, tales como:

Definición de visiones. La propuesta de Shipman relacionada con la definición y actualización de visiones no es completa ya que en ella sólo se trata el caso en que una actualización sobre una visión ocasiona una única actualización sobre la base de datos. EFDM [13], por otro lado, ofrece un mecanismo diferente para definir visiones que no permite visiones actualizables. Nuestro propósito es extender posteriormente el lenguaje LBF, de manera que soporte visiones actualizables considerando el caso de múltiples actualizaciones sobre la base de datos.

El uso de tipos de datos en lenguajes para bases de datos. Un concepto bastante útil para el modelaje del mundo real es el de tipos de datos. Por ejemplo, en PASCAL [10] y EUCLID [14] podemos definir los siguientes tipos de datos

```
TYPE ALTURA = 1...100

TYPE PESO = 1...100
```

Consecuentemente, operaciones sin significado pueden ser efectuadas sobre estos tipos (por ejemplo, sumar pesos con alturas). Por tanto, es necesario que se haga una mejor verificación de tipos para obtener una integridad semántica mayor. Trabajos a este respecto han sido desarrollados en el contexto del modelo relacional [2,21]. Sería interesante extender LBF para que permita diferenciar dos tipos de entidades semanticamente diferentes.

El uso de tipos abstractos de datos. Tipos abstractos de datos han sido ampliamente estudiados en el contexto de lenguajes de programación. Ultimamente, se ha prestado bastante atención a la utilización de abstracción de datos en el contexto del modelaje semántico de datos [5]. En ciertas areas de aplicación, como en la utilización de bases de datos para el soporte a CAD ("Computer Aided Design") existen problemas serios en la representación y operación de objetos. Tipos abstractos de datos (TAD's) pueden ser una alternativa para la creación y manejo de objetos como puntos, líneas y polígonos. Recientemente, algunas implementaciones de TAD's han sido desarrolladas, entre ellas podemos citar el trabajo de Stonebraker en el contexto del sistema INGRES [24]. En este trabajo, fueron definidos TAD's sobre columnas de una relación, es decir, TAD's simples. Un trabajo interesante puede ser extender LBF no sólo para soportar TAD's simples como fue propuesto por Stonebraker [24], sino también para soportar otros tipos más comple-

jos que puedan ser utilizados en aplicaciones de CAD.

5. CONCLUSIONES

Nosotros hemos descrito brevemente los elementos básicos del lenguaje LBF, el cual está basado en el modelo funcional de Shipman. Este modelo no es el modelo semántico más completo pero como ya vimos es um modelo simple de utilizar. Este modelo consigue juntar conceptos como el de orientación a objetos y jerarquías de tipos (generalización) sin perder su simplicidad mientras que otros modelos semánticos como SDM [8] Y RM/T [7], son tan complejos que su implementación se vuelve difícil, además de ser, a veces, poco adecuados para la mayoría de usuarios. Modelos del tipo DAPLEX pueden facilitar, por tanto, el acceso de un mayor número de personas a la tecnología de bases de datos.

6. BIBLIOGRAFIA

[1] BIGONHA, M.A.S., "SIC : Sistema de Implementação de Compiladores". Belo Horizonte,Departamento de Ciência da Computação (DCC) ICEX-UFMG, Junho, 1985. (Tesis de Maestrado)

[2] BRODIE, M.L. "The Aplication of Data Types to Database Semantic Integrity". **Information Systems**, Pergamon Press , 5(4):287-296, 1980.

[3] BRODIE, M.L. & SILVA, A. "Active and Passive Component Modelling". In: OLLE, T. W. et al (eds.). "**Information Systems Design Methodologies: A Comparative Review**". North-Holland, Amsterdam, 1982, p. 41-91.

[4] BRODIE, M.L. "On the Development of Data Models". In: BRODIE, M.L., MYLOPOULOS, J. & SCHMIDT, J.W. (eds.). "**On Conceptual Modelling. Perspectives from Artificial Inteligence, Databases, and Programming Languages**". New York, Springer Verlag, 1984, p.21-47.

[5] BRODIE, M.L. & RIDJANOVIC, D. "On the Design and Specification of Database Transactions". In: BRODIE, M.L., MYLOPOULOS, J. & SCHMIDT, J.W. (eds.). "**On Conceptual Modelling. Perspectives from Artificial Inteligence, Databases, and Programming Languages**". New York, Springer Verlag, 1984, p.277-306.

[6] BUNEMAN, P. & FRANKEL, R.E. "FQL - A Functional Query Language". In: **PROCEEDING OF ACM SIGMOD CONF. ON MANAGEMENT OF DATA**, Boston, Mass., 1979, p. 52-58.

[7] CODD, E.F. "Extending the Relational Model of Data to Capture more Meaning". **ACM Tranactions on Database Systems**, New York, 4(4):397-434, 1979.

[8] HAMMER, M. & MCLEOD, D. "Database Description with SDM: a Modelling Mechanism for Database Aplications". **ACM Transactions on Database**

Systems, New York, 6(3):351-386, 1981.

[9] HOUSEL, B.C. , WADDLE, V. & YAO, S.B. "The Functional Dependency Model for Logical Database Design". In: **PROCEEDINGS OF INTERNATIONAL CONFERENCE ON VERY LARGE DATA BASES**, 5, Rio de Janeiro, Brazil, Out. 1979, p.194-208.

[10] JENSEN, K. & WIRTH, N. "PASCAL User Manual and Report", 2nd Ed., Lectures notes in Computer Science, Berlim, Springer-Verlag, 1974, p.18.

[11] KATZ, R.H. & WONG, E. "Resolving Conflicts in Global Storage Design through Replication". **ACM Transactions on Database Systems**, New York, 8(1):110-135, 1983.

[12] KERSHBERG, L. & PACHECO, J.E.S. **"A Functional Data Base Model"**. Rio de Janeiro, Brazil, Pontificia Universidade Católica, Fev. 1976. (Monograph in Computer Science 2/76).

[13] KULKARNI, G. K. **"Evaluation of functional data models for database design and use"**. Edinburgh, University of Edinburgh, 1983. 153p. (Tesis de Ph.D.).

[14] LAMPSON, B.W. & et al. "Report on the Programming Language EUCLID", **SIGPLAN Notices**, 12(2), 1977.

[15] McLEOD, D. & SMITH, J.M. "Abstractions in Databases". In: BRODIE, M.L. & ZILLES, S.N. (eds.) PROC. WORKSHOP ON DATA ABSTRACTION, DATABASIS AND CONCEPTUAL MODELLING. **SIGPLAN Notices**, New York, 16(1):19-25, 1981.

[16] ORMAN, L. "Functions in Information Systems". **Data Base**, 16(3):10-13, 1985.

[17] PEREZ, J.J.,LAENDER, A.H.F. & BIGONHA, R.S. **"Especificação da Linguagem para Banco de Dados Funcionais (LBF)"**. (Reporte técnico en elaboración).

[18] SCHIEL, U. "An Abstract Introduction to the Temporal Hierarchic Data Model (THM)", In: **PROCEEDINGS OF INTERNATIONAL CONFERENCE ON VERY LARGE DATA BASE**, 9, Florence, Italy, 1983.

[19] SHIPMAN, D. W. "The Functional Data Model and the Data Language DAPLEX". **ACM Transactions on Database Systems**, New York, 6(1): 140-173, 1981.

[20] SIBLEY, E.H. & KERSCHBERG, L. "Data Architecture and Data Model Considerations". In: **AFIPS CONFERENCE PROCEEDINGS, NATIONAL COMPUTER CONFERENCE**, Montvale, N. J., AFIPS Press, 1977, V.46, pp.85-96.

[21] SMITH, J. M. & SMITH D.C.P. "Database Abstractions: Aggregation and Generalization". **ACM Transactions on Database Systems**, New York, 2(2):105-133, 1977.

[22] SMITH, J.M. et al. "Multibase - Integrating Heterogeneous Distributed Database Systems". In: **AFIPS CONFERENCE PROCEEDINGS, NATIONAL COMPUTER CONFERENCE**, Montvale, N.J., AFIPS Press, 1981, V.50, p.487-499.

[23] SOFTCRAFT, Inc. **"Btrieve version 3.0 user's guide"**. Austin, Texas, 1984.

[24] STONEBRAKER M. "Adding Semantic Knowledge to a Relational Database System". In: BRODIE, M.L., MYLOPOULOS, J. & SCHMIDT, J.W. (eds.). **"On Conceptual Modelling. Perspectives from Artificial Inteligence, Databases, and Programming Languages"**. New York, Springer Verlag, 1984, p.333-353.

[25] TSCICHRITZIS, D. & LOCHOVSKY, F.H. **"Data Models"**. Prentice-Hall, Englewood Cliffs, N.J., 1982, 381p.

[26] WIRTH, N. "What can we do about the unnecesary diversity of notation for sintactic definition?". **Communications of ACM**, 20(11):822-823, 1977.

[27] WONG, E. & KATZ, R.H., "Logical Design and Schema Conversion for Relational and DBTG Databases". In : CHEN P. P.(ed.) **"Entity-Relationship Approach to Systems Analysis and Design"**. Amsterdam, North-Holland, 1980, p.344-383.

APENDICE

DEFINICION SINTACTICA DE LBF.

1. Notación para la descripción sintactica

Para describir la sintaxis de LBF usaremos el siguiente formalismo [26]: los símbolos no terminales de la gramática seran descritos en letras minúsculas y los terminales seran siempre colocados entre comillas. Cada producción tendrá la forma

S = E

donde S representa un símbolo no terminal y E las alternativas que definen S. La expresión E tiene la forma

T1 ¦ T2 ¦ ... ¦ Tn (n > 0)

donde Ti (que llamaremos término i) tiene la forma

F1 F2 ... Fn (n > 0)

donde cada Fi (que llamaremos factor i) puede ser:

(a) un símbolo terminal
(b) un símbolo no terminal
(c) {T}
(d) [T]
(e) la sentencia vacía.

Por tanto cada término pude generar una concatenación de factores que a su vez pueden generar los casos anteriores. Un término entre llaves representa una secuencia de ese mismo término, incluyendo la sentencia vacía. Un término entre corchetes representa ese mismo término o la sentencia vacía. Pares de paréntesis pueden ser usados libremente para indicar agrupamientos.

2. Sintaxis de LBF

```
comando = declarativo ¦ imperativo
        ¦"CONSULTA" idconsulta ":" {imperativo} imperativo
         "FIN" [idconsulta]
        ¦"EJECUTE" idconsulta idarchivo
        ¦"REMUEVA" (especfun¦idconsulta)
        ¦idconsulta

declarativo = "DECLARE" especfun ("->"¦"->>")
```

```
                    (idtipo[orden] {restr}
                     | "ENTIDAD"
                     | "ENTRADA")
              |"DEFINA" especfun ("->"|"->>")
                    (expr|
                     "INVERSA DE" especfun|
                     "TRANSITIVA DE" expr|
                     "(" tupla ")"
                    ) [orden] {restr}

restr = "TOTAL"|"MAXIMO" int|"MINIMO" int|"FIJO"|"UNICO"

tipoprim  =  "ALFA" "(" int ")"| "NUMERICO" "(" int[","int]  ")|
             "LOGICO"

expr = expr_conjunto|expr_simple

tupla = expr {"," expr}

especfun = idfun "(" [idtipo {"," idtipo}] ")".

expr_conjunto = "SI" predicado "ENTONCES" expr_conjunto
                ["SINO" expr_conjunto] "FIN"
  | llamfunmv | idtipo | "{" [simple { "," simple } ] "}" | tipoprim
  |"("expr_conjunto {("UNION"|"INTERSECCION"|"DIFERENCIA") expr_conjunto|")"
  | identificador "EN" expr_conjunto | expr_conjunto "COMO" idtipo
  | expr_conjunto "TAL QUE" predicado
  | expr_conjunto comp (expr_simple | cuant expr_conjunto)

expr_simple = termolog [ "O" termolog ]

termolog = factorlog [ "Y" factorlog]

factorlog = [ "NO" ] exprlog

exprlog = expraritm [ comp exprearitm]

expraritm = [prefijo] termo [opsuma termo]

termo = factor [opmult factor]

factor = exprentid ["COMO" idtipo]

exprentid = constante | idvar | llamfuns | llamag | predicado
     | ("EL"|"LA") expr_conjunto | "UN NUEVO" idtipo
     | ("EL"|"LA") expr_conjunto ("QUE PRECEDE" | "QUE SIGUE") expr_simple
     | "SI" predicado "ENTONCES" expr_simple ["SINO" expr_simple] "FIN"
     | "(" expr_simple ")"

llamfuns = llamfun
```

```
llamfunmv = llamfun

llamfun = idfun "(" [tupla] ")"

llamag = ("CARDINAL"|"MAXIMO") "("expr_conjunto")"
       | ("TOTAL"|"MEDIA") "("["SOBRE"] expr_simple ")"

predicado = lógico
          | "PARA" (expr_simple | cuant expr_conjunto) predicado
          | expr_simple comp cuant expr_conjunto
          | cuant expr_conjunto comp (expr_simple|cuant expr_conjunto)
          | cuant expr_conjunto ("EXISTE" | "EXISTEN")

comp = ">" | "<" | "=" | ">=" | "<=" | "<>"

cuant = "ALGUN" | "TODO" | "NINGUN"
       | (("AL" "MENOS" | "ALO" "SUMO")) | "EXACTAMENTE") entero

entero = simple

lógico = simple

constante = ent | cad | log | num

ent = digito{digito}

num = [ent] "." int

cad = """" letra{letra}letra """"

log = "VERDADERO" | "FALSO"

imperativo = "PARA CADA" expr_conjunto[orden] cuerpo "FIN"
           |"PARA" simple cuerpo "FIN"
           |actualización
           |impresión

cuerpo = (imperativo|selección) {imperativo|selección}

selección = "SI" predicado "ENTONCES" cuerpo
              ["SINO" cuerpo] "FIN"

orden = "EN ORDEN" {[("ACENDENTE" | "DECENDENTE")] "POR" expr_simple}.

actualización = "SEA" (llamfuns|llamfummv|idvar) "=" expr
           |"INCLUYA" (llamfunmv|idtipo) "=" expr
           |"EXCLUYA" (llamfunmv|idtipo) "=" expr
           |"REMUEVA" expr_simple
           |"INSERTE" llamfunmv "=" (expr_simple | expr_conjunto[orden])
                                "QUE" ("PRECEDE" | "SIGUE") expr_simple
```

```
impresión = ["CONSIDERE" [titulo] [rompimiento] [total]]
     ("ESCRIBA"|"IMPRIMA") "("expr ":" cad {"," expr ":" cad")" idarchivo

titulo = "ENCABEZADO" texto {"." texto}

texto = (idvar | cad | expr_simple) {"," (idvar | cad | expr_simple)}

rompimiento = "ROMPIMIENTO POR" expr_simple ["," texto "'"opcionesq"'"]
                    {"," expr_simple ["," texto "'"opciones"'"]}

opcionesq = P : causa  un  salto de página después de la impresión de  los
                datos asociados al rompimiento en curso.

            A : causa que el texto sea emitido antes de los  detalles.  En
                el caso de que la opción no sea espicificada el texto será
                colocado después de los detalles.

total = "TOTAL POR" expr {"," expr } ["GRAN TOTAL" texto]

idvar = identificador

idtipo = identificador

idfun = identificador

idconsulta = identificador

idarchivo = identificador

prefijo = "+" | "-"

sumop = "+" | "-" | "&"

mulop = "*" | "/" | "RES"

identificador = letra{caracter_de_id}

caracter_de_id = letra
                | digito
                | "_"

letra = "A" | "a" | "B" | "b" | ... | "Z" | "z"

digito = "0" | "1" | ... | "9"
```

AGRADECIMIENTOS

Este trabajo está siendo desarrollado con el apoyo finaciero de la CAPES y del CNPQ (procesos número 302107/81-CC y 402170/85-CC).