a) TOXLINE, SMOKING AND HEALTH, CHEMICAL SAFETY NEWSBASE are extremely useful to acquire or collect the information in occupational safety and health.
b) AIDSLINE and Canderlit which covers literature on the specific disease are very useful for the information.
c) EMBASE, POLLUTION ABSTRACTS, MEDLINE are also useful for getting them.

Table 7. The databases strongly related to occupational safety and health

rank	fileno	tim.	file name in DIALOG
1	161	9	OCCUPATIONAL SAFETY & HEALTH (NIOSH)_73-91
2	156	7	TOXLINE _ 1965-1992
2	160	7	SMOKING AND HEALTH _ 70-89
2	317	7	CHEMICAL SAFETY NEWSBASE_1981-92
3	157	6	AIDSLINE _ 1980-91
3	159	6	Cancerlit_1963-92
4	73	5	EMBASE (EXCERPTA MEDICA)_74-92
5	41	4	POLLUTION ABSTRACTS _ 70-92
5	155	4	MEDLINE _ 66-92

fileno : file number in DIALOG
tim. : occurrence times of databases appeared in Table 6

CONCLUSION

It was concretely proved in our study which databases are effective to occupational safety and health. The authors concluded that the procedure used in present investigation could be taken as applicable and useful for other themes on general interest of interdisciplinary field.

REFERENCES

1. These data, quantity of records of the databases were extracted by means of online searching for BLUESHEET database which was conducted in July 1992.

2. Online searching for DIALINDEX file was conducted in July 1992

3. The data of updating file in DIALOG is different with that of BLUESHEET. Therefore these values are not accurate.

Elsevier Science B.V.
© FID, 1994

Reglas de Producción para una Base de Conocimientos en la Construcción de Tesauros

Rodríguez Muñoz J.V.[a]; Díaz Ortuño P.[b]; Moya Martínez G.[b]; Martínez Mendez J.[b]

[a] Departamento de Información y Documentación. Universidad de Murcia. España

[b] Departamento de Informática y Automática. Universidad de Murcia. España

Abstract

The times are gone when the software for the handling of large amounts of data was sheer craftmanship. The last two decades have witnessed the rise of models aimed at setting up a formal theoretical frame to afford coherence to data design. At present such models can be seen to have become well established, even commercially, as is the case with the Relational Model put forward by Codd in 1970. Currently, owing to substantial progress in IA and allied issues, a new developmental stage can be said to have been reached in so called deductive data bases, where not only text, numeric or factual data are stored but also knowledge in the form of inference rules. In this contribution we present a knowledge base which wil enable us, through the relevant facts and rules, to construct a thesaurus.

Keywords data model. data base. artificial intelligence. relational model. deductive data base. logic programing. thesaurus.

INTRODUCCION

Un aspecto a considerar de los tesauros [AIT82,83,87], dejando aparte su validez como lenguaje documental destinado al control del vocabulario; es el de la gran cantidad de información implícita que nos proporcionan. Esta información viene detallada en base a la lista de términos constituyente del tesauro y todas las relaciones existentes entre ellos [VAN87]. Todo este conjunto de relaciones entre términos y su diversa tipología conforman un sistema de información muy útil a la hora de generar una base de conocimientos para la gestión documental.

En anteriores trabajos [MAR92] [ROD90] [ROD92] hemos introducido la idea del diseño a nivel abstracto de un tesauro utilizando un Modelo de Datos. Un modelo de datos es una herramienta utilizada para el diseño lógico-conceptual de sistemas de información; identificando los objetos y las relaciones que se establecen entre ellos.

Uno de los Modelos de Datos de mayor relevancia y posteriores desarrollos es el Modelo Entidad Relación [CHE76,77a.77b,77c,80], [DIA89]. Este modelo de datos está constituido por dos elementos fundamentales:

a) Las Entidades o conjunto de objetos individuales que se distinguen unos de otros por medio de sus atributos y ..

b) Las Relaciones o asociaciones que se establecen entre las entidades.

Un Tesauro se presenta como un sistema de información cuya estructura es fácilmente modelizable utilizando este tipo de herramienta conceptual. Los términos descriptores son objetos operacionales distinguibles unos de otros y además, se establecen entre ellos una serie de relaciones de naturaleza semántica que resultan de sencilla implementación en un esquema lógico-conceptual.

En una primera fase [ROD90], aportábamos la idea de implementar un soporte para un tesauro basado en un esquema relacional. El punto de partida, tal como indicamos en el párrafo anterior, es el diseño a nivel lógico-conceptual de un esquema Entidad-Relación apropiado para un Tesauro.

En el mismo, se identifican una serie de objetos individuales (entidades), distinguibles por medio de unas características particulares de cada objeto (atributos); sobre estos objetos individuales se definen una serie de operaciones (las relaciones).

Por medio de un proceso de reducción aplicado sobre el esquema E-R [MOY89], obtenemos el esquema relacional apropiado para el almacenamiento en diversas tablas de los términos constituyentes del tesauro y las diversas relaciones existentes entre ellos.

La ampliación y modificación posterior de esta propuesta inicial viene a sustituir el original modelo E-R por el modelo de datos Entidad-Categoría-Relación [WEE80] [ROD92]. Este nuevo modelo de datos nos proporciona el concepto de Categoría que nos permite la modelización de un sistema de información a un nivel abstracto superior, agrupando conjuntos de entidades.

Este modelo nos permite la utilización del lenguaje de definición GORDAS [ELM85] para considerar los problemas de integridad de los datos y de las estructuras.

A continuación, a este nuevo esquema ECR se le puede aplicar el proceso de reducción a modelo relacional [COD70,71,74,79,82] para obtener el esquema relacional definitivo. Tomando este esquema relacional como punto de partida, podemos construir una base de datos relacional adaptada específicamente para la generación y el mantenimiento de un tesauro.

Esta base de datos va a tener dos tipos de información: información explícita e implícita. La información explícita se corresponde con los hechos depositados en la base de datos relacional (el contenido de las tablas); y la información implícita que no aparece dentro de una vista relacional de los datos, pero que se puede deducir por medio de una serie de reglas de inferencia lógica.

Llegados a este punto, introducimos el concepto de Base de Datos Deductiva [FRO89], que podemos definir sucintamente como una base de datos de la que podemos extraer información más allá de la que se encuentra explícita para cualquier usuario de la misma.

Con más detalle, en una Base de Datos deductiva se aplican una serie de reglas de inferencia lógica sobre los hechos explícitos de la misma; estas reglas van a producir una nueva información como resultado de su aplicación.

Esta nueva tipología de base de datos viene a proporcionarnos una nueva dimensión al anterior soporte relacional de los datos que hemos diseñado específicamente para un tesauro. Si utilizamos la base de datos relacional como punto de partida de un proceso de aplicación de un conjunto de reglas de inferencia lógica, generaremos una serie de nuevas tablas con la información implícita [MAR92].

Por lo tanto, obtenemos un sistema de información más completo que el original; ya que en el mismo recogemos una serie de relaciones entre los objetos que no aparecen en el primero.

No basta sólo con extraer información adicional en base a unos hechos previamente definidos, sino que tenemos que asegurarnos de la validez y consistencia de la misma.

En cuanto al aspecto de la validez de los nuevos datos, debemos de asegurarnos que la información que éstos proporcionan no se contradiga en ningún momento con la ya existente en la base de datos.

Por lo tanto, centramos nuestra línea de trabajo en el desarrollo de un motor de inferencia (en base a las reglas), que aplicado sobre la base de datos relacional (configurada por los hechos recogidos en el tesauro origen), nos produzca un nuevo tesauro donde tenga cabida también la información implícita y se verifiquen todas las propiedades de integridad de los datos.

DISEÑO LOGICO-CONCEPTUAL

En todo sistema de información hay dos aspectos bien diferenciados, aunque haya entre ellos una fuerte interrelación: los datos y los tratamientos. Si bien la concepción del sistema de datos y la del conjunto de tratamientos no puede realizarse de forma totalmente independiente, los problemas a resolver son de naturaleza distinta.

A medida que los diseñadores de sistemas de información se van convenciendo de la transcendencia que la gestión racional de los datos tiene para conseguir el desarrollo coherente y eficaz de estos sistemas, las bases de datos empiezan a ocupar un primer plano en las áreas de interés de los informáticos.

La situación de años atrás, de falta de metodologías y de instrumentos informáticos de ayuda al diseño ha ido mejorando; sin embargo, en el momento actual, los métodos usados en el diseño de bases de datos muchas veces no están soportados ni por fundamentos científicos ni por una disciplina de ingeniería, sino sólo en la experiencia e intuición del diseñador.

En el proceso de diseño de bases de datos es necesario distinguir, a grandes rasgos, tres etapas:

Diseño conceptual: cuyo objetivo es obtener una buena representación de los recursos de información de las organizaciones, con independencia de usuarios o aplicaciones en particular y fuera de consideraciones sobre eficiencia del ordenador.

Diseño lógico: cuyo objetivo es transformar el esquema conceptual obtenido en la etapa anterior, adaptándolo al modelo de datos -Jerárquico, Codasyl o Relacional- en el que se apoya el Sistema Gestor de Bases de Datos (SGBD) que se va a utilizar.

Diseño físico: cuyo objetivo es conseguir una instrumentación lo más eficiente posible del esquema lógico.

Estas son las tres etapas fundamentales que aparecen en esta metodología, la cual no pretende presentarse como original ni muy diferente de otras existentes, pero si consideramos coherente y práctica.

En la fase de diseño conceptual -crítica para el desarrollo de todo el sistema- proponemos la utilización del modelo Entidad- Relación, como instrumento adecuado y muy extendido para representar el "universo del discurso", es decir, el fenómeno real que tratamos de plasmar en nuestra bases de datos, expresando así, de manera formal, los resultados del proceso de análisis de la organización.

Para el diseño lógico se ha elegido el modelo relacional, creemos que la elección de este modelo está plenamente justificada en el momento actual, en el que la tecnología relacional va imponiéndose, no sólo a nivel teórico sino en sistemas comerciales. Por otro lado los fundamentos matemáticos del modelo relacional permiten introducir una formalización en el diseño de la que se carece en otros modelos, lo que no significa en absoluto ni que el diseño relacional exija profundos conocimientos matemáticos ni tampoco que esté alejado de práctica.

La realidad es que en el modelo relacional se formalizan ideas sencillas que eran ya aplicadas por cualquier diseñador que tuviese una cierta experiencia, pero la teoría del diseño relacional hace comprender la razón de que ciertas estructuras de datos sean más adecuadas que otras y presenten menos problemas, además de enseñar a buscar estas estructuras de una forma más sistemática.

Teniendo en cuenta estas consideraciones, se desarrolló [ROD92] un esquema conceptual para definir el tesauro como un concepto en si mismo, obteniendo así una herramienta que nos permitiera a partir de esta la creación de bases de datos de tesauros específicos, pero ya bajo un esquema capaz de detectar las posibles incoherencias, lo que nos proporciona la deseada integridad de los datos.

BASES DE DATOS DEDUCTIVAS

Una base de datos deductiva es una base de datos en la que podemos derivar información a partir de la que se encuentra almacenada explícitamente. Como elementos constitutivos de una base de datos deductiva nos encontramos con los Hechos, Reglas de Inferencia y las Restricciones de Integridad.

Los Hechos representan la información que se almacena explícitamente; en el diseño e implementación de las Reglas de Inferencia, que definen como se deriva nueva información a partir de la que está almacenada explícitamente, se toma como base la lógica de primer orden y las Restricciones de Integridad que determinan las propiedades semánticas que la base de datos debe cumplir, son de la misma tipología que en el modelo relacional.

Como quiera que hemos señalado que tomamos como base la lógica de primer orden y dado que esta estudia las interrelaciones de implicación entre premisas y conclusiones, podemos pues, usarla para expresar proposiciones, relaciones entre proposiciones y como inferencia válida para pasar de unas proposiciones a otras. Por otro lado desde el punto de vista lógico, un sistema gestor de bases de datos se puede asimilar como un sistema de pregunta-respuesta de propósito general, en el cual el conjunto de hechos necesarios para la contestación de preguntas puede verse como axiomas de un teorema, y las cuestiones verlas como resolución del teorema.

Con esta última visión, parece lógico pensar que puesto que un tesauro es una estructura basada en un esquema de relaciones semánticas [ROD92], es factible abordarlo a través de una axiomatización mediante programación lógica, obteniendo un sistema basado en conocimiento capaz de inferir información no almacenada de forma explícita.

REGLAS DE PRODUCCION

Para la realización de la implementación de las reglas de producción de nuestro tesauro, hemos elegido LISP, como lenguaje de programación, más concretamente la versión GOLDEN COMMON LISP 1.1 (GCLISP), ya que en su utilidad EXPLORER tenemos las funciones necesarias para hacer la base de datos relacional que nos sirve de punto de partida. Dicha utilidad cuenta con las siguientes instrucciones, ya previamente implementadas:

- DB-MAKE-RELATION: Construye las relaciones (Tablas).
- DB-SHOW: Nos muestra las relaciones en forma de tablas.
- DB-PROJECT: Transforma una relación en otra con menor numero de campos.
- DB-SELECT: Transforma una relación en otra consistente en tuplas que cumplan una determinada condición.
- DB-EXTRACT-VALUE: Devuelve el valor de un campo particular; Si hay más de un campo o registro que pueden cumplir este valor, esta función imprime un "warning".
- DB-UNION: Combina los registros en dos relaciones para hacer una tercera relación.
- DB-JOIN: Combina dos relaciones, añadiendo los campos de una relación a los campos de otra, pero sólo cuando un predicado especificado se encuentre entre los valores de los campos especificados.

En realidad, GCLISP trabaja con listas, y las tablas tienen la representación interna de una lista compuesta por listas, donde la cabecera es una lista con los nombres de los campos y el resto son listas, donde cada una de ellas contiene la información de una tupla.

Como podemos ver GCLISP nos permite construir mediante funciones los elementos básicos de un álgebra relacional que tomamos como base a partir del esquema conceptual ECR del tesauro.

A partir de estas funciones podemos construir las tablas que van a contener los hechos, esto es la información explícita, tendremos pues que pasar a construir las reglas de producción que generen nueva información a partir de los hechos contenidos en la base de datos.

Así, basándonos en esta estructura hemos implementado las funciones que nos permiten realizar la relación *JERARQ** [ROD92], la cual contiene todas las relaciones jerárquicas descendientes para cada uno de los componentes de la relación *JERARQ (NT)* [ROD92].

```
(defun nt1 ()
        (setq l (cdr jerarq))
        (setq z nil)
        (nt l)
        (setq jerarq* (append (list (car jerarq)) l)))
```

Esta función crea la lista *jerarq** donde estan contenidos los códigos de los descriptores relacionados mediante *jerarq* junto con sus descendientes. Nos servimos de dos listas auxiliares, *l* y z. En *l* tendremos la lista formada por las tuplas de la lista *jerarq*, excepto su cabecera, que es la lista formada por los nombres de los campos de la misma. La lista z, que inicialmente estará vacía (nil), es donde se irán incorporando los elementos que se derivan a partir de la lista *jerarq*.

Finalmente la unión de ambas listas formará la lista *jerarq** juntos con los nombres de los campos.

Esta función llama a la siguiente función, denominada *nt*.

```
(defun nt (x)
        (nt-aux x)
        (nt-aux (reverse x))
        (cond ((equal z nil) l)
                (t (setq l (append z l)) (setq z nil)(nt l))))
```

Esta función llama a su vez a la función *nt-aux*, la cual realiza los procesos de comparación para comprobar si se debe añadir o no un nuevo elemento a la lista *l*. Hacemos una busqueda hacia delante *(nt-aux x)* y una búsqueda hacia detrás *(nt-aux (reverse x))* para comprobar con total seguridad que todas las relaciones jerárquicas se van a incorporar a *jerarq**.

```
(defun nt-aux (x)
        (cond ((eq (cdr x) nil) nil)
                ((eq (second (car x)) (car (car (cdr x))))
                        (setq j (list (first(car x))
                        (second (first (cdr x))))) (poner j z))
                (t (nt-aux(append(list(car x))(cdr(cdr x))))(nt-aux(cdr x))))))
```

Aquí comprobamos para cada elemento de la lista (por ello la función es recursiva), si el segundo elemento de una tupla es igual al primer elemento de la siguiente tupla. Si esto

ocurre, creamos una nueva lista *(j)* de dos elementos que contiene como primer elemento al primer elemento de la primera tupla y al segundo elemento de la segunda tupla, enviándola a la función *poner*. En el caso de que la comparación no sea satisfactoria, realizamos una nueva lista con los elementos de la anterior, excepto el elemento (tupla) que no satisfacía la comparación, y volvemos a realizar de nuevo la comparación.

```
(defun poner (x y)
        (cond ((or (ver x l) (ver x y)) (setq z z))
                (t (setq z (append (list x) z)))))
```

Esta función añade a la lista z el nuevo elemento que habíamos creado con la función *nt-aux*, comprobando antes, mediante la función *ver* que dicho elemento no estuviera anteriormente ni en la lista original -*l*- ni en la lista donde vamos añadiendo los nuevo elementos -z-. (Esto lo hacemos para asegurarnos que no vamos a tener elementos introducidos en *jerarq**).

```
(defun ver (x y)
        (cond ((null y) nil)
                ((equal x (car y)) t)
                (t (ver x (cdr y)))))
```

Como ya hemos mencionado anteriormente, la única misión de esta función en comprobar si un elemento pertenece a una lista o no, devolviendo el valor t (true) si pertenece el elemento a la lista o nil (false) en caso contrario.

Con *(nt1)* ponemos en ejecución la función *nt1* para que construya la lista *jerarq** a partir de la lista *jerarq*.

Dado que el carácter de las relaciones jerárquicas en un tesauro son de alcance inmediato, lo que ponemos de manifiesto con estas reglas es el concepto de *ascendiente* que tiene la relación NT (de forma inversa con la relación BT aplicaríamos el concepto de descendiente) y que se encuentra implícito en un tesauro, así de forma más genérica podríamos expresar estas reglas del siguiente modo:

i) Ascendiente(x,y) <-- NT(x,y)

ii) Ascendiente(x,y) <-- NT(x,z) ∧ Ascendiente(z,y)

Con las mismas, queda definida el concepto de *Ascendiente*, de la siguiente manera:

i) toda relación NT es Ascendiente.

ii) un término x es Ascendiente de otro y si existe u z de tal forma que x NT z y z sea a su vez Ascendiente de y.

Aunque las relaciones asociativas no son transitivas, podríamos hacer uso de las funciones anteriormente definidas para poder hacer un análisis de la transitividad implícita de las mismas en un determinado tesauro, que permitiría controlar de forma inmediata, gracias a las reglas de producción, estas relaciones tan difíciles de normalizar. Para ello definiríamos la siguiente función *rt*.

```
(defun rt ()
        (setq l (cdr asoc))
        (setq z nil)
        (nt l)
        (setq asoc* (append (list (car asoc)) l)))
```

Esta función crea la lista *asoc** [ROD92] donde estan contenidos los códigos de los descriptores que están relacionados mediante *asoc* [ROD92] junto con sus descendientes. Nos servimos de las dos listas auxiliares, *l* y z. En *l* tendremos la lista formada por las tuplas de la lista *asoc*, excepto su cabecera, que es la lista formada por los nombres de los campos de la misma. La lista z, que inicialmente estará vacía (nil), es donde iremos incorporando los elementos que se derivan a partir de la lista *asoc*. Finalmente la unión de ambas listas formará la lista *asoc** junto con los nombres de los campos. Como podemos observar, esta función llama a la función *nt1*, ya descrita anteriormente, debido a que el proceso de construcción de la lista *asoc** es idéntico al de la lista *jerarq**.

A continuación se presentan una serie de funciones que permiten mostrar las tablas de una manera más comprensible. Las mismas estan realizadas con las funciones de consulta que proporciona GCLISP.

```
(defun ver-cspdesc ()
        (db-show (db-project (db-join (db-project (db-join cs cspdesc
         where cod-cs eql cod-cs) over nom-cs and cod-cs cod-desc)
                 (db-project desc over nom-desc and cod-desc)
         where cod-desc eql cod-desc) over nom-cs and
         cod-cs cod-desc nom-desc)))
```

Esta función nos muestra la tabla *cspdesc* (relaciones de pertenecía [ROD92]), con los nombres y los códigos de los campos semánticos y descriptores que relaciona.

```
(defun ver-intraq ()
        (db-show (db-project (db-join (db-project (db-join desc intraq
         where cod-desc eql cod-desc) over nom-desc and cod-desc cod-nd)
                 (db-project nd over nom-nd and cod-nd)
         where cod-nd eql cod-nd) over nom-desc and cod-desc cod-nd nom-nd)))
```

Al igual que la anterior, nos muestra la tabla *intraq* (relaciones de equivalencia intralingüisticas [ROD92]), junto con los nombres y los códigos de los descriptores y no descriptores que relaciona.

```
(defun ver-jerarq ()
        (db-show (db-join (db-project (db-join desc jerarq where
                 cod-desc eql cod-desc) over nom-desc and cod-desc cod-desc1)
                 (db-project desc over nom-desc and cod-desc)
                  where cod-desc1 eql cod-desc)))
```

Aquí, implementamos una función para que nos muestre la tabla *jerarq*, con los nombres y códigos de los descriptores implicados en la misma.


```
(defun ver-jerarq* ()
        (db-show (db-join (db-project (db-join desc jerarq* where cod-desc eql
                 cod-desc) over nom-desc and cod-desc cod-desc1)
                 (db-project desc over nom-desc and cod-desc)
                  where cod-desc1 eql cod-desc)))
```

Idéntica a la anterior, solo que trabajamos con la tabla *jerarq** en lugar de *jerarq*.

```
(defun ver-asoc ()
        (db-show (db-join (db-project (db-join desc asoc where cod-desc eql
                 cod-desc) over nom-desc and cod-desc cod-desc1)
                 (db-project desc over nom-desc and cod-desc)
                  where cod-desc1 eql cod-desc)))
```

Esta función nos muestra la tabla *asoc*, con los nombres y códigos de los descriptores implicados en la misma.

```
(defun ver-asoc* ()
        (db-show (db-join (db-project (db-join desc asoc* where cod-desc eql
                 cod-desc) over nom-desc and cod-desc cod-desc1)
                 (db-project desc over nom-desc and cod-desc)
                  where cod-desc1 eql cod-desc)))
```

Al igual que la función anterior, nos muestra la tabla *asoc**, generada por la función *rt* a partir de *asoc*, con los nombres y códigos de los descriptores que están relacionados mediante dicha relación.

CONCLUSIONES

En los últimos años se han desarrollado numerosos sistemas de gestión automática de tesauros, tal y como podemos ver en [LAG89] [GAN90] [RAD90] [RIT90] [FUG90] [SCH90]. Sin embargo el desarrollo de la inmensa mayoría está basado en técnicas tradicionales de construcción de software, sin llegar a plantearse la necesidad de un modelado del sistema. En [PER86] se introduce el concepto de un lenguaje pivote en base al cual se produciría un total desarrollo de un tesauro multilingüe. En [MAR89] se sigue trabajando bajo el mismo enfoque tradicional, pero se hace especial hincapié en la necesidad de identificar las entidades y las posibles asociaciones que se pueden establecer entre ellas en todos los niveles de jerarquía de un tesauro.

Siguiendo con las etapas de diseño [ROD92], trasladamos el esquema al modelo relacional con un doble objetivo, de un lado, conseguir la implementación (cuestión que se esta desarrollando mediante un SGBD relacional), y de otro utilizar la normalización como medio de validación de nuestro diseño ya transformado a un esquema relacional.

Es evidente que existen otras posibilidades de abordar la construcción de tesauros en algunos casos complementarias, y en otras alternativas al desarrollo tradicional de los mismos. Nos referimos a las aportaciones de la inteligencia artificial, de forma que se alcance mayor grado de interactividad con el usuario. Si bien esto es en si un buen objetivo, creemos que antes de

abordarlo era necesario conocer y tener la experiencia tanto en la estructura de los tesauro como en la respuesta que iban a dar estos aplicándoles un modelo de datos para obtener un esquema conceptual. Con esta experiencia y haciendo uso de la lógica como mecanismo de representación del conocimiento que nos proporciona un tesauro podemos comprobar que solo son realmente tratables las ya clásicas estructuras jerárquicas y que por tanto podemos concluir, con un mayor grado de seguridad, de la pobreza semántica de los tesauros, si bien aún hoy son herramientas útiles como medios de representación del conocimiento.

BIBLIOGRAFIA

[AIT82] AITCHINSON, J. **Indexing languajes: classifications schemes and thesauri**. Handbook of special librarian and information work, Anthony, L.J. (ed), 5th ed. London: Aslib, 1982

[AIT83] AITCHINSON, T.M. and HARDING, P. **Automatic indexing and classification for mechanised information retrieval**. Information Management Research in Europe. Proceedings of the EURIM 5 Conference, Versailles, France, May 1982. London: Aslib, 1983, p. 47-55.

[AIT87] AITCHINSON, T.M.; GILCHRIST, A. **Thesaurus construction. A Practical manual**. 2ª ed. Aslib, 1987. 173 p.

[CHA90] CHAKRAVARHY, U.S.; GRANT, J. and MINKER, J. **Logic-Based Approach to Semantic Query Optimization**. ACM Transactions on Databases Systems, vol. 15, nº 2, June 1990, p. 162-207.

[CHE76] CHEN, P.P. **The Entity-Relationship Model-Toward a Unified View of Data**. ACM Trans. Databases Systems, vol 1, nº 1, 1976. p. 9-36.

[CHE77a] CHEN, P.P. **The entity-relationship approach to logical data base design**. Wellesley, (Massachusetts): Q.E.D. Information Science, Inc., 1977. 73 p.

[CHE77b] CHEN, P.P. **The entity-relationaship model. A basis for the enterprise view of data**. Reprinted from AFIPS Conference Proceedings, vol. 46, 1977. p. 158-165.

[CHE77c] CHEN, P.P. and BING YAO, S. **Desing and performance tools for data base systems**. Proc. 3rt, IEEE International Conference on Very Large Database, 1977. p. 3-15.

[CHE80] CHEN, P. (ed.) **Entity-Relationship Approach to Systems Analysis and Design**, Amsterdam, 1980.

[COD70] CODD, E.F. **A Relational Model of Data for Large Shared Data Banks**. Communications of the ACM, vol 13, nº 6, jun 1970, pp. 377-387.

[COD71] CODD, E.F. **A data base sublanguage based on the relational calculus**. Proc. ACM SIGFIDET workshop on data description, access and control, 1971. p. 35-68.

[COD74] CODD, E.F. **Recent investigations in relational data base systems**. Proc. IFIP Conference, 1974. p. 1017-1021.

[COD79] CODD, E. F. **Extending the Database Relational Model to Capture More Meaning**. ACM Transactions on Database Systems, vol 4, nº 4, dec 1979, pp.397-434; IBM Research Report RJ2599, San Jose, (Calif.), Aug 1979.

[COD82] CODD, E. F. **Relational Database: A Practical Foundation for Productivity**. Communications of the ACM, vol. 25, nº 2, feb 1982.

[DIA89] DIAZ ORTUÑO, P.M. **Desarrollo del esquema conceptual mediante la utilización del modelo entidad-relación**. Escuela Universitaria de Informática. Universidad de Murcia, 1989. 114 p. (Trabajo inédito dirigido por el prof. J.V. Rodríguez).

[ELM85] ELMASRI, R.; WEELDREYER, J.; HEVNER, A. **The category concept: An extension to the entity-relationship model**. Data & Knowledge Engineering, vol. 1, 1985. p. 75-116.

[FUG90] FUGMANN, R. **An interactive Classaurus on the PC**. Int. Classif., vol. 17, nº 3/4, 1990. p. 133-137.

[FRO89] FROST, R. **Bases de datos y sistemas expertos**. Madrid: Díaz Santos, S.A., 1989. 769 p. (Trad. de la obra **Introduction to Knowledge Base Systems**. 1986).

[GAN90] GANZMANN, J. **Criteria for the evaluation of thesaurus software**. International Classification, vol. 17, nº 3/4, 1990. p. 148-157.

[LAG89] LAGUNA SERRANO, E.; IRAZAZABAL NERPELLL, A. de; VALLE BRACERO, A. **Confección Automática de Tesauros.** Revista Española de Documentación Científica, vol. 12, nº 2, 1989. p. 129-140.

[MAR89] MARTINEZ MENDEZ, J. **Aportaciones a la automatización de la normalización del tratamiento documental**. Escuela Universitaria de Informática. Universidd de Murcia, 1989. 116 p. (Trabajo inédito dirigido por el prof. J.V. Rodríguez).

[MAR92] MARTINEZ MENDEZ, J. et al. **Diseño lógico-conceptual de tesauros**. IV J. Catalanes de Documentació, 1992. p. 341-352

[MOY89] MOYA MARTINEZ, G. **Construcción del esquema conceptual para sistemas relacionales a partir de un modelo de datos**. Escuela Universitaria de Informática. Universidad de Murcia, 1989. 121 p. (Trabajo inédito dirigido por el prof. J.V. Rodríguez).

[PER86] PEREZ ALVAREZ-OSSORIO, J.R.; RIUDAVETS MONTES, A.; VALLE BRACERO, A. **Cambio automático de lenguaje pivote en un tesauro multilíngüe informatizado**. I Jornadas Españolas de Documentación Automatizada, 1986. p.601-620.

[RAD90] RADA, R. **Maintaining Thesauri and Metathesauri**. INt. Classif.,vol. 17, nº 3/4, 1990. p.158-164.

[RIT90] RITZLER, Cl. **Comparative Study of PC-supported Thesaurus Software**. Int. Classif., vol. 17, nº 3/4, 1990. p. 138-147.

[ROD90] RODRIGUEZ MUÑOZ, J.V. et al. **Los modelos de datos y modelo relacional como alternativa en la construcción de tesauros**. III J. Españolas de Documentación Automatizada, vol 2, 1990. p. 1145-1157.

[ROD92] RODRIGUEZ MUÑOZ, J.V. **Construcción del esquema conceptual del tesauro mediante un modelo de datos**. Universidad de Murcia, Tesis Doctoral inédita, 1992.

[SCH90] SCHMITZ-ESSER, W. **Thesauri facing new chachenges**. Int. Classif., vol. 17, nº 3/4, 1990. p. 129-132.

[VAN87] VAN SLYPE, G. **Les langages d'indexation: conception, construction et utilisation dans les systèmes documentaires**. Paris: Les Editions d'Organisation, 1987. (Traducción al castellano publicada por la Fundación Germán Sánchez Ruipérez, 1991. 198 p.).

[WEE80] WEELDREYER, S.A. **Structural aspects of the entity-category-relationship model of data**. Honeywell CCSC, technical report, HR-80-251. Bloomington, Minnesota. March, 1980.

J.R. Alvarez-Ossorio and B.G. Goedegebuure (Editors)
Elsevier Science B.V.

An Argentine Common Format for Documents (FOCAD).

T. Suter, A. Cassanello, C. Mabragaña, R. Monfasani, L. Revello & E. Zítara.

Grupo Formato Común, Centro Argentino de Información Científica y Tecnológica (CAICYT). Consejo Nacional de Investigaciones Científicas y Técnicas (CONICET). Moreno 431, (1091), Buenos Aires, Argentina.

Abstract

In 1988, the Format Group presented a common format for entering information in bibliographic databases. This format was adopted -sometimes modified- by different scientific and university institutions, in order to register the existing material in their libraries and documentation centers. The experience thus gathered, gave way to a complete revision of the common format, which is now presented as Argentine Common Format for Documents. Its Spanish acronym is FOCAD.

FOCAD features the following characteristics:
- 1. It is used both for entering and transferring data.
- 2. About 100 fields are defined, with data desegregated in subfields, in order to allow: a) conversion to other formats; b) greater flexibility for information retrieval.
- 3. FOCAD is conceived as a format generator. A system designer is able to select FOCAD fields, as well as subfields, required for a given database. It allows also to add other fields, out of format. However, a designer should stick to specifications -as setup for FOCAD fields- so that compatibility is secured while transferring. This methodology allows format to be used by a wide range of users.
- 4. Specifications have been set up so that a fine compatibility with UNESCO's CCF is achieved. At the same time, the possibility of transforming data to adjust it to the CEPAL format, has been taken in account (CEPAL is the Spanish acronym for the Economic Commission for Latin America - ECLA). Regarding cataloguing aspects not specifically considered, users should resort to AACR2.
- 5. Fields which may prove useful for library management are included.

Software needed to implement FOCAD should allow repeatable fields and subfields, all of them with a variable length.

In order to make use of FOCAD easier, a manual has been published with a field-to-field description, with related examples where MicroISIS characteristics have been used (since MicroISIS is a widely used software in Argentina). The manual includes several appendixes with code tables, rules, guidelines for FOCAD implementation onto MicroISIS, relation to CCF and a conversion table to CEPAL format. A diskette with FOCAD structure on MicroISIS is available.