

Modelo de datos jerárquico y
en red. Estructuras.
Operaciones.

TEMA 37

ABACUS NT

Oposiciones 2021

Índice

- 1. Introducción**
- 2. Tipos de bases de datos**
- 3. Tipos de bases de datos**
 - 3.1. Bases de datos relacionales**
 - 3.2. Sistemas NoSQL.**
 - 3.3. Sistemas en la Nube.**
 - 3.4. El modelo jerárquico**
 - 3.4.1. Características de la estructura jerárquica
 - 3.4.2. Clasificación de los árboles
 - 3.4.3. Esquema y ocurrencia de árbol
 - 3.4.4. Definición del Modelo Jerárquico
 - 3.4.5. Problemas del modelo Jerárquico
 - 3.4.6. Función de manipulación de datos en el modelo jerárquico
 - 3.5. El modelo en red**
 - 3.5.1. Presentación de un modelo en red general
 - 3.5.2. Objetivos del modelo Codasyl
 - 3.5.3. Elementos del modelo de datos (definiciones)
 - 3.5.4. Definición formal del modelo Codasyl
- 4. Conclusión**
 - 4.1. Relación del tema con el sistema educativo actual**
- 5. Bibliografía**

1. Introducción

Los SGBD organizan y estructuran los datos de tal forma que puedan ser recuperados y manipulados por usuarios y programas de aplicación. Las estructuras de los datos y las técnicas de acceso proporcionadas por un SGBD particular se denominan su Modelo de Datos. El modelo de datos determina la “personalidad” de un SGBD, y las aplicaciones para las cuales está particularmente bien conformado.

Es normal presentarle al usuario una vista de los datos en la que deliberadamente se omiten los detalles sobre la forma en que están representados esos datos en el almacenamiento. Ésta es la denominada vista externa (subesquema).

En la mayoría de los sistemas actuales, la vista externa y la vista conceptual son muy semejantes.

El rango de estructuras de datos soportadas al nivel de usuario (ya sea externo o conceptual), es un factor que afecta de manera decisiva a muchos componentes del sistema.

En particular, impone el diseño del lenguaje de manipulación de datos correspondiente, ya que, cada operación de DML debe definirse en términos de su efecto sobre esas estructuras de datos.

Así pues, la pregunta: ¿qué estructuras de datos y operadores asociados debe soportar el sistema?, es crucial. Es conveniente clasificar a los diferentes sistemas de bases de datos de acuerdo con el enfoque que adoptan para dar respuesta a dicha pregunta.

Los tres enfoques mejor conocidos y más extendidos son los siguientes:

- El enfoque relacional
- El enfoque jerárquico
- El enfoque en red

En la *estructura relacional*, nos encontramos con los datos organizados en tablas. Cada una de esas tablas se asemeja mucho a un fichero secuencial convencional, donde las filas de la tabla corresponderían a los registros del fichero, y las columnas a los campos (ítems) de los registros. Cada una de estas tablas en realidad es un caso especial de la construcción conocida en matemáticas como relación.

En el *enfoque jerárquico*, los datos se representan por una sencilla estructura de árbol. El registro principal de un árbol es el registro raíz, que puede tener cualquier número de registros dependientes, y cada uno de éstos, también puede tener cualquier número de registros dependientes, y así sucesivamente.

Las consultas son simétricas en el caso relacional pero no en el jerárquico. La pérdida de simetría es una consecuencia directa de la vista, que en sí misma es asimétrica. Esta asimetría es un enorme inconveniente del enfoque jerárquico, porque conduce a complicaciones innecesarias para el usuario.

En términos específicos, el usuario está obligado a dedicar tiempo y esfuerzo a resolver los problemas introducidos por la estructura de datos jerárquica, y que no son intrínsecos a

las preguntas formuladas. Lo que induce a pensar que cuantos más registros y más complicados sean, más, a su vez, se complicará la estructura, obteniéndose una depuración y mantenimiento en alto nivel y continuamente.

En el *enfoque en red*, al igual que en el jerárquico, los datos se representan por registros y enlaces, pero una ocurrencia de registro puede tener cualquier número de superiores inmediatos, es decir, no está limitado a un máximo de uno, como ocurre en la jerárquica.

De esta manera, el enfoque en red permite modelar una correspondencia de muchos a muchos, de manera más directa que en el enfoque jerárquico. Además, se introduce un nuevo tipo de registro denominado conector.

Una ocurrencia del conector representa la asociación entre dos o más tipos de registros y contiene datos de esa asociación. Todas las ocurrencias de un conector para un tipo de registro dado, se colocan en una cadena que parte de ese registro y retorna a él.

2. Tipos de bases de datos

Tradicionalmente, las bases de datos se clasifican en tres grupos: **jerárquicas, en red y relacionales**. Las bases de datos jerárquicas fueron las primeras en aparecer. La información se representa en forma de árbol. El problema con este tipo de estructura es que no todas las bases de datos se adaptaban a la **estructura en árbol**. En un intento de eliminar la rigidez de las bases de datos jerárquicas, se desarrolló la **estructura en red**, en la cual se permitían todo tipo de relaciones. Cualquier conjunto de información es representable mediante una base de datos en red. Por último, aparecieron las **bases de datos relacionales**, que pretendían obtener una mayor flexibilidad y rigor en el tratamiento de datos. Nacieron en los años 70 de la mano de E. F. Codd, quién planteó una alternativa a las bases de datos jerárquicas y en red existentes hasta el momento.

2.1. Bases de datos relacionales

Una base de datos relacional está formada por **tablas**. Una tabla es una estructura bidimensional formada por una sucesión de registros del mismo tipo. Si se imponen ciertas restricciones a las tablas, se pueden tratar como relaciones matemáticas. De ahí el nombre de este tipo de base de datos y el hecho de que a las tablas de una base de datos relacional se les llame relaciones.

El modelo relacional es uno de los más utilizados en el diseño y gestión de bases de datos, debido fundamentalmente a dos razones: por una parte, se basa en un reducido número de conceptos, que hacen de este sistema uno de los más fáciles de entender, diseñar, cambiar, administrar y acceder, y, por otra parte, posee un lenguaje de definición y manipulación de datos muy potente y flexible.

2.2. Sistemas NoSQL.

Típicamente las bases de datos relacionales modernas han mostrado poca eficiencia en determinadas aplicaciones que usan los datos de forma intensiva, incluyendo el indexado de un gran número de documentos, la presentación de páginas en sitios que tienen gran

tráfico, y en sitios de streaming audiovisual. Las implementaciones típicas de SGBDR se han afinado o bien para una cantidad pequeña pero frecuente de lecturas y escrituras o para un gran conjunto de transacciones que tiene pocos accesos de escritura. Por otro lado, NoSQL puede servir gran cantidad de carga de lecturas y escrituras.

2.3. Sistemas en la Nube.

Una base de datos en la nube es una colección de contenido, estructurado o no estructurado, que reside en una plataforma de infraestructura de computación en la nube privada, pública o híbrida.

Existen dos modelos de entorno de base de datos en nube: tradicional y base de datos como servicio (DBaaS).

2.4. El modelo jerárquico

A diferencia del modelo de datos relacional, que se fundamenta firmemente en las matemáticas, y del modelo de datos en red, que se desarrolló a partir de un esfuerzo para establecer estándares detallados, el modelo de datos jerárquico se ha desarrollado a través de la práctica.

No existen documentos originales que definan el modelo jerárquico, como los hay para los otros dos modelos. Afortunadamente, las implementaciones de bases de datos jerárquicas están dominadas por un sistema, **IMS (Sistema de Gestión de Información de IBM)**, desarrollado en los años 60 como soporte al proyecto lunar Apolo. Un factor clave en su desarrollo fue la necesidad de manipular millones de piezas que se relacionaban unas con otras de manera jerárquica.

Una de las aplicaciones más importantes en los Sistemas de Gestión de Bases de Datos primitivos era el planteamiento de la producción para empresas de facturación.

Si un fabricante de automóviles decidía producir 10000 unidades de un modelo de coche y 5000 unidades de otro modelo, necesitaba saber cuántas piezas pedir a sus suministradores. Para responder a la cuestión, el producto (un coche) tenía que descomponerse en ensamblajes (motor, cuerpo, chasis ...) que a su vez se descomponían en subensamblajes (válvulas, cilindros, bujías ...) y luego en subsubensamblajes, etc. El manejo de estas listas de piezas, conocido como una "Cuenta de materiales", era un trabajo a medida para los ordenadores. La cuenta de materiales para un producto tenía una estructura jerárquica natural. Para almacenar estos datos, se desarrolló el modelo de datos jerárquico.

En este modelo, cada registro de la Base de Datos representa una pieza específica. Los registros tenían "relaciones Padre/Hijo", que ligaba cada pieza a sus subpiezas, y así sucesivamente.

Para acceder a los datos en la Base de Datos, un programa podría:

- Hallar una pieza particular mediante su número (Ej. puerta izquierda).

- Descender al primer hijo (Ej. el tirador de la puerta).
- Ascender hasta su padre (Ej. el cuerpo),
- Moverse de lado hasta el siguiente hijo (Ej. la puerta derecha).

La recuperación de los datos en una Base de Datos jerárquica requería por tanto "navegar" a través de registros, moviéndose hacia arriba, hacia abajo y hacia los lados de un registro cada vez.

Existen ciertas estructuras de datos que no son planas y reciben el nombre de archivos jerárquicos, estructuras ramificadas o **árboles**, estas estructuras son el soporte de las Bases de Datos Jerárquicas. Los árboles no son el mejor método de representación lógica de la Base de Datos.

Todo árbol puede ser descrito como una jerarquía de nudos con relaciones binodales de tal modo que:

- El más alto nivel de la jerarquía tiene un solo nudo llamado raíz.
- Los nudos restantes se reparten en $m \geq 0$ conjuntos disjuntos (es decir, no conectados) T_1, T_2, \dots, T_m y cada uno de estos conjuntos constituye a su vez un árbol. Los árboles T_1, T_2, \dots, T_m se llaman **subárboles** de la raíz.

Los árboles, como instrumentos para la representación de estructuras de datos, presentan problemas por su poca flexibilidad, lo que da origen a una falta de adaptación a muchas organizaciones reales.

Además, no se ha llegado a una formulación matemática del modelo y de sus lenguajes como ha ocurrido en el caso relacional; ni tampoco se ha intentado su estandarización, como en Codasyl, a pesar de lo cual los productos jerárquicos (IMS, DL/1 de IBM) consiguieron altas cuotas de mercado.

Las ventajas del IMS y su modelo jerárquico son las siguientes:

- Estructura simple. La organización de una Base de Datos IMS era fácil de entender. La jerarquía de la Base de Datos se asemejaba al diagrama de organización de una empresa o un árbol familiar.
- Organización Padre/Hijo. Una Base de Datos IMS era excelente para representar relaciones Padre/Hijo, tales como "A es una pieza de B" o "A es propiedad de B".
- Rendimiento. IMS almacenaba las relaciones Padre/Hijo como punteros físicos de un registro de datos a otro, de modo que el movimiento a través de la Base de Datos era rápido. Puesto que la estructura era sencilla, IMS podía colocar los registros Padre e Hijo cercanos unos de otros en el disco, minimizando la Entrada/Salida de disco.

IMS sigue siendo el SGBD más ampliamente instalado en los grandes ordenadores IBM. Se utiliza en más del 25% de las instalaciones. Existen importantes aplicaciones soportadas en estos productos que están trabajando, por su eficiente respuesta, a satisfacción de sus usuarios.

2.4.1. Características de la estructura jerárquica

En este modelo, el esquema es una estructura arborescente compuesta de nodos, que representan las entidades, enlazadas por arcos, que representan las asociaciones o interrelaciones entre dichas entidades.

La estructura del modelo de datos jerárquico es un caso particular de la del modelo en Red, con fuertes restricciones adicionales derivadas de que las asociaciones del modelo jerárquico deben formar un árbol ordenado, es decir, un árbol en el que el orden de los nodos es importante.

2.4.2. Clasificación de los árboles

Los árboles se pueden clasificar atendiendo a varios criterios. Así, podemos distinguir:

- Árbol *balanceado*: es aquel en el que todos los nodos tienen el mismo número de hijos, excepto en el último en el preorden
 - Árbol *no balanceado*: es aquel en el que los nodos pueden tener diferente número de hijos.
 - Árbol *binario*: es aquel en el que cada nodo tiene, cero, uno o dos hijos.
 - Árbol *estrictamente binario*: es aquel en el que cada nodo tiene cero o dos hijos.
- Las estructuras jerárquicas se clasifican también como:
- *Lineales*: es un caso particular y simple en el que cada tipo de registro padre sólo puede tener un tipo de registro hijo.
 - *Arborescente* propiamente dicha: un tipo de registro padre puede tener varios tipos de registro descendientes.

Las organizaciones jerárquicas pueden servir para describir tanto estructuras lógicas como físicas.

2.4.3. Esquema y ocurrencia de árbol

Un esquema jerárquico consiste en una descripción de un determinado universo del discurso mediante un árbol en el que los nodos representan los tipos de registro (Entidades), y los arcos, los tipos de interrelaciones jerárquicas existentes entre los mismos. Una ocurrencia o instancia de dicho esquema será también un árbol, pero en él los nodos representan las ocurrencias de los registros, y los arcos, las interrelaciones jerárquicas entre dichas ocurrencias.

Una Base de Datos jerárquica está formada por una colección o bosque de árboles disjuntos.

2.4.4. Definición del Modelo Jerárquico

Se puede definir el modelo jerárquico como:

- Un conjunto de tipos de entidad (segmentos, grupos repetitivos, registros, etc.) $E_1, E_2 \dots E_n$ (nodos de un grafo).
- Un conjunto de interrelaciones o asociaciones nominadas R_{ij} que conectan tipos de entidad E_i y E_j (arcos del grafo).

- Un conjunto de restricciones inherentes que provienen de la estructura jerárquica.

2.4.5. Problemas del modelo Jerárquico

La poca flexibilidad de este modelo puede obligar a la introducción de redundancias cuando es preciso instrumentar situaciones del mundo real que no responden a una jerarquía. Se puede calcular un *índice de redundancia* mediante:

$$I_r = (\text{Núm. nodos extra} / \text{Núm. Total de nodos}) \times 100$$

Este índice de redundancia permite determinar hasta qué punto una estructura del mundo real se adapta más o menos a un árbol.

La redundancia que hemos calculado es lógica, y en general no coincide con la física. Esto es debido a que al almacenar los datos en el dispositivo físico no se suelen repetir los registros completos, sino sólo los identificadores, o bien se introducen punteros. Se trata ya de soluciones de instrumentación propias de cada suministrador. En general, los productos ofrecen algún tipo de solución a estos problemas.

Además del grave problema que provocan estas redundancias no controladas por el sistema, existe otro importante inconveniente en este tipo de solución como es la no conservación de las simetrías naturales en el mundo real.

Las actualizaciones en las Bases de Datos jerárquicas pueden originar problemas debido a las restricciones inherentes al modelo:

- Toda alta, a no ser que corresponda a un nodo raíz, debe tener un padre.
- La baja de un registro implica que desaparezca todo el subárbol que tiene dicho registro como nodo raíz, con lo que pueden desaparecer datos importantes que convendría conservar en la Base de Datos.

Los SGBD basados en el modelo jerárquico suelen facilitar instrumentos que los dotan de una mayor flexibilidad para representar estructuras no estrictamente jerárquicas. Sin embargo, puede ocurrir que dicha instrumentación no proporcione la debida independencia físico/lógica. Los sistemas jerárquicos IMS y DL/1 de IBM permiten definir lo que llaman "relaciones lógicas" y "bases de datos lógicas" para salvar este tipo de situaciones.

Las restricciones de usuario no se pueden definir en el modelo jerárquico.

En cuanto a las ventajas, cabe citar su sencillez de comprensión y la mayor facilidad de instrumentación en los soportes físicos, aunque esto depende en gran medida de los productos. Además, si se trata de estructuras del mundo real que sean de tipo jerárquico, este modelo puede ser muy idóneo. Por otra parte, los productos jerárquicos suelen ser muy eficientes, en especial IMS.

2.4.6. Función de manipulación de datos en el modelo jerárquico

La manipulación de datos jerárquicos, al igual que ocurre en todo modelo, necesita localizar (seleccionar) primero los datos sobre los que va a trabajar para realizar a continuación la acción de recuperación o actualización sobre dichos datos.

a) Localización o Selección

Esta operación es de tipo navegacional, es decir, trabaja registro a registro, en oposición a otros modelos como el Relacional en los que se seleccionan conjuntos de registros. Existen las siguientes formas básicas de búsqueda:

- Seleccionar un determinado conjunto de datos (como un registro) que cumpla una cierta condición. En el lenguaje DL/1 se realizará este tipo de selección mediante la instrucción **get unique** que activará (y también recuperará) el primer registro (segmento) que cumpla la condición especificada en un predicado.
- Seleccionar el siguiente conjunto de datos, que se encuentra perfectamente definido al existir un único camino jerárquico (preorden). Se puede imponer también una condición que ha de cumplir el registro para ser seleccionado. En DL/1 se utiliza la instrucción **get next**, con ella se selecciona el siguiente segmento en el preorden.
- Seleccionar el registro padre de otro dado (que ha sido activado previamente) se conoce como normalización jerárquica ascendente, mientras que la selección de descendientes se llama normalización jerárquica descendente.

Puesto que una instrucción jerárquica selecciona un único registro, será preciso que el lenguaje de datos esté embebido en un lenguaje de programación mediante el cual se describirá el procedimiento necesario para ir recorriendo el árbol con el fin de buscar y manipular los datos.

b) Acción

Cuando se ha seleccionado un registro, se tendrá que realizar sobre él una acción, sea de recuperación o de actualización.

La **recuperación**, consiste en llevar al registro marcado como activo en la selección realizada previamente al área de E/S. En DL/1 se utiliza la sentencia **get** (Obtener) para esta función.

En cuanto a la **actualización**, es preciso distinguir entre:

- Reemplazar un conjunto de elementos de datos por otro.
- Borrar un conjunto de datos.
- Insertar un conjunto de datos.

Debido a la naturaleza jerárquica de las conexiones entre registros, las inserciones y borrados de registros requieren consideraciones especiales:

- Cuando un nuevo registro se inserta en una Base de Datos jerárquica, excepto para la raíz, tiene que ser conectado a un nodo padre previamente seleccionado mediante alguna sentencia de selección. El nuevo registro se inserta como hijo del registro padre seleccionado.
- Cuando un registro se borra en una Base de Datos jerárquica, excepto si se trata de una hoja, se han de borrar todos los registros descendientes de él.

2.5. El modelo en red

La estructura sencilla de una Base de Datos Jerárquica se convertía en una desventaja cuando los datos tenían una estructura más compleja. En una Base de Datos de procesamiento de pedidos, por ejemplo, un simple pedido podría participar en tres relaciones Padre/Hijo diferentes, ligando el pedido al cliente que lo remitió, al vendedor que lo aceptó y al producto ordenado.

La estructura de este tipo de datos simplemente no se ajustaría a la jerarquía estricta de IMS.

Para manejar aplicaciones tales como el procesamiento de pedidos, se desarrolló un nuevo modelo de datos en Red. El modelo de datos en Red extendía el modelo jerárquico permitiendo que un registro participara en múltiples relaciones Padre/Hijo.

Estas relaciones eran conocidas como "conjuntos" en el modelo en Red. En 1971 la Conferencia sobre Lenguajes de Sistemas de Datos publicó un **estándar oficial para Bases de Datos en Red**, que se hizo conocido como el modelo **Codasyl**. IBM nunca desarrolló un SGBD en Red por sí mismo, eligiendo en su lugar extender el IMS a lo largo de los años. Pero durante los años 70, Compañías de Software independientes se apresuraron a adoptar el modelo en Red, creando productos tales como el IDMS de Cullinet, el Total de Cincon y el SGBD Adabas que se hizo muy popular.

Para un programador, acceder a una Base de Datos en Red, era muy similar a acceder a una Base de Datos jerárquica. Un programa de aplicación podía:

- Hallar un registro Padre específico mediante clave (Ej. Un número de cliente).
- Descender al primer hijo en un conjunto particular (Ej. el primer pedido remitido por ese cliente).
- Moverse lateralmente de un Hijo al siguiente dentro del conjunto (Ej. a la orden siguiente remitida por el mismo cliente).
- Ascender desde un Hijo a su padre en otro conjunto (Ej. el vendedor que aceptó el pedido).

Una vez más el programador tenía que recorrer la Base de Datos registro a registro, especificando esta vez qué relación recorrer además de indicar la dirección.

Las Bases de Datos en Red tenían varias ventajas:

- **Flexibilidad.** Las múltiples relaciones Padre/hijo permitían a una Base de Datos en Red, representar datos que no tuvieran una estructura jerárquica sencilla.
- **Normalización.** El estándar CODASYL reforzó la popularidad del modelo en Red, y los vendedores de miniordenadores como Digital Equipment Corporation y Data General implementaron Bases de Datos en Red.
- **Rendimiento.** A pesar de su superior complejidad las Bases de Datos en Red, reforzaron el rendimiento aproximándolo al de las Bases de Datos Jerárquicas. Los conjuntos se representaron mediante punteros a registros de datos Físicos, y en algunos sistemas el Administrador de la Base de Datos podía especificar la agrupación de datos basada en una relación de conjunto.

Las Bases de Datos en Red tenían sus desventajas también, igual que las Bases de Datos Jerárquicas resultaban muy rígidas. Las relaciones de conjunto y la estructura de los registros tenían que ser especificadas de antemano. Modificar la estructura de la Base de Datos requería típicamente la reconstrucción de la Base Datos completa.

Las dos Bases de Datos eran herramientas para programadores. Para responder a una cuestión tal como ¿Cuál es el producto más popular ordenado por ACME Manufacturing? un programador tenía que escribir un programa que recorriera su camino a través de la Base de Datos, la anotación de las peticiones para informes a medida duraba con frecuencia semanas o meses, y para el momento en que el programa estaba escrito la información que se entregaba con frecuencia, ya no merecía la pena.

2.5.1. Presentación de un modelo en red general

El modelo de datos en red general representa las entidades en forma de nodos de un grafo, y las asociaciones o interrelaciones entre éstas, mediante los arcos que unen dichos nodos.

Esta representación, que no impone en principio ninguna restricción ni al tipo ni al número de los arcos, permite el modelado de estructuras de datos tan complejas como se deseé.

Este modelo en red permitiría la representación de cualquier tipo de interrelación sin ninguna restricción inherente.

Podríamos definir el modelo en red general con una mayor formalización, como un conjunto finito de tipos de entidades:

$$\{E_1, E_2, \dots, E_n\}$$

Con sus respectivas propiedades (atributos):

$$\{A_{11}, A_{12}, \dots, A_{1n}, \dots, A_{n1}, A_{n2}, \dots, A_{nm}\}$$

Y un conjunto finito de tipos de interrelaciones (al igual que las entidades y que los atributos, tienen un nombre):

$$\{I^{h_j, k, \dots, n}\}$$

Con esta notación representamos la interrelación entre los elementos j, k, ..., n (bien sean entidades y/u otras interrelaciones), donde el superíndice h permite diferenciar dos interrelaciones distintas entre los mismos elementos, ya que se refiere al nombre de la interrelación.

Este modelo en red general es muy flexible debido a la inexistencia de restricciones inherentes, pero también por esta misma razón su instrumentación física resulta difícil y poco eficiente. Esta es la causa de que el modelo, tal como lo hemos presentado, sea teórico, y que al llevarlo a la práctica se introduzcan restricciones. El modelo Codasyl y jerárquico responden a estas estructuras de tipo de red, pero con restricciones bastante fuertes, en especial el modelo jerárquico.

Un modelo de datos de tipo red que introduce restricciones inherentes es el denominado modelo Codasyl. Este modelo constituye una simplificación del modelo en red general, en el que se admiten sólo determinados tipos de interrelaciones y se incluyen algunas restricciones adicionales que, sin embargo, no limitan excesivamente la flexibilidad que proporciona el modelo en red general, facilitando en cambio una instrumentación eficiente.

2.5.2. Objetivos del modelo Codasyl

El grupo Codasyl enunció una serie de características que en su opinión debía de perseguir cualquier sistema de gestión de Base de Datos. Las más importantes son:

- Flexibilidad para los usuarios.
- Uso concurrente.
- Estrategias de búsqueda diversas.
- Seguridad.
- Gestión centralizada de almacenamiento físico.
- Independencia de almacenamiento físico.
- Flexibilidad en el modelo de datos.
- Facilidad para el usuario.
- Independencia de los programas respecto de los datos.
- Descripción de datos independientes.
- Independencia respecto de los lenguajes.
- Interfaces con múltiples lenguajes.

Para que un SGBD pueda conseguir los objetivos anteriores, Codasyl propone los siguientes lenguajes:

- a) **Lenguaje de Definición del Esquema** (Schema DDL). Permite describir la estructura de la totalidad de los elementos que forman parte de la Base de Datos a nivel lógico (entidades, atributos, interrelaciones, etc.), permitiendo al usuario la elección de su estructura, nombres, etc. La independencia datos/aplicaciones en los sistemas Codasyl es relativa.
- b) **Lenguaje de Definición de Subesquemas** (Subschema DDL). Permite la descripción de subconjuntos del esquema para diferentes usuarios que no necesitan para su trabajo más que una parte determinada de la Base de Datos. Permite introducir ciertos cambios en el formato y en la estructura de los elementos del esquema que intervienen en el subesquema.
- c) **Lenguaje de manipulación de datos** (DML). Se ocupa de las operaciones de recuperación, inserción, borrado, modificación, etc., de los datos que contiene la Base de Datos.
- d) **Lenguaje de Control de Dispositivo/Soporte** (DMCL).
- e) **Lenguaje de Definición del Esquema de almacenamiento** (DSDL). Considera todos los aspectos físicos del almacenamiento de los datos recogidos en la Base de Datos. Permite describir cómo se organizan y almacenan los datos en los soportes, independientemente de los dispositivos y del Sistema Operativo.

La **arquitectura** de las especificaciones de Codasyl ha variado sustancialmente desde las primeras propuestas hasta las actuales, pasando de una arquitectura a dos niveles (esquema y subesquema) a otra en la que existen tres niveles, ya que se separan los aspectos físicos, en un nuevo esquema (llamado de almacenamiento) de las características de tipo lógico que continúan formando parte del esquema.

3. Operaciones en el modelo Codasyl

3.1.1. Definición formal del modelo Codasyl

Se puede definir el modelo Codasyl como un conjunto finito de tipos de registros: $\{R_1, R_2, \dots, R_n\}$.

Cada uno de ellos está compuesto por un conjunto finito de elementos de datos. Entre los tipos de registros se establecen interrelaciones, llamadas conjuntos: $\{C^k_{i,j}, \dots, h\}$. Este conjunto representa la interrelación entre los registros: R_i, R_j, \dots, R_h .

El tipo de registro R_i es el propietario y los demás son los miembros; el superíndice k indica que entre la misma colección de tipos de registro R_i puede haber más de un SET, al ser estos nominados. Cada tipo de registro tiene un conjunto finito de ocurrencias, entre las cuales existen las vinculaciones definidas por los SET del esquema; una ocurrencia de registro propietario encadenada con las correspondientes ocurrencias de registros miembro constituye una ocurrencia de SET.

3.2. Componentes

Los elementos básicos de la estructura de datos propia del modelo Codasyl son los siguientes:

- **Elementos de datos** (data ítem). Es la unidad de datos más pequeña a la que se pueda hacer referencia en el modelo Codasyl. Debe tener un nombre, y una ocurrencia del mismo contiene un valor (booleano, numérico, carácter, etc.). Además, puede definirse como dependiente de los valores de otros elementos (datos derivados).
- **Agregado de datos** (data aggregate). Puede ser un vector o un grupo repetitivo. El elemento y el agregado de datos se corresponden con los campos de los ficheros clásicos y con los atributos de otros modelos. En este modelo se admiten estructuras no planas como son los agregados.
- **Registro** (Record). Colección nominada de elementos de datos. Es la unidad básica de acceso y manipulación de la Base de Datos, y se corresponde con el concepto de registro en los ficheros clásicos y de entidad en otros modelos como el modelo E/R.
- **Conjunto** (SET o COSET). Es una colección nominada de dos o más tipos de registros que establece una vinculación entre ellos. Es un elemento clave y distintivo de este modelo de datos, siendo el origen de muchas de sus restricciones, tanto inherentes como opcionales.
- **Área**. Es una subdivisión nominada del espacio de almacenamiento direccionable de la Base de Datos que contiene ocurrencias de registros. Ha sido un elemento muy discutido por considerarse que atenta contra la independencia al ser una característica física. En las últimas versiones ha pasado a formar parte del esquema de almacenamiento.
- **Clave de Base de Datos** (Database key). Identificador interno único para cada ocurrencia de registro, que proporciona su dirección en la Base de Datos. Las

últimas versiones del modelo, aunque conservan este identificador interno, restringen mucho su uso por parte de los programadores, que no pueden guardarla de una unidad de ejecución (run unit) para utilizarla en otra.

Hay que distinguir entre tipos de registro y tipos de SET que se definen en el esquema y las ocurrencias de los mismos que constituyen la Base de Datos, la cual, en un instante dado, está formada por todas las ocurrencias de los registros y sus vinculaciones (SET) descritas en el correspondiente esquema.

4. Operaciones en el modelo en red

Para ver las operaciones nos fijaremos en el DML del modelo de red, el **DBTG** (Data Base Task Group). El lenguaje consta de varias órdenes que están incorporadas en un lenguaje principal, Pascal.

Find y get

Localiza un registro en la base de datos da valores a los punteros adecuados. Existen dos órdenes **find** diferentes para localizar registros individuales en la base de datos:

```
find any <tipo de registro> using <campo de registro>
```

Esta orden localiza un registro del tipo <tipo de registro> cuyo valor de <campo de registro> es el mismo que el valor <campo de registro> en la plantilla de <tipo de registro> en el área de trabajo del programa. Una vez que se encuentra ese registro se modifican los punteros para que apunten a ese registro.

La operación **get** copia el registro al que apunta el actual de unidad de ejecución de la base de datos a la plantilla adecuada en el área de trabajo de programa.

Pueden existir dos registros con el valor especificado, para ello utilizamos la orden:

```
find duplicate <tipo de registro> using <campo de registro>
```

Acceso a registros dentro de un conjunto con la operación **find**:

```
find first <tipo de registro> within <tipo de conjunto>
```

que localiza el primer registro de la base de datos del tipo especificado que pertenezca al conjunto <tipo de conjunto> actual. Para localizar los demás miembros del conjunto se utilizaría:

```
find next <tipo de registro> within <tipo de conjunto>
```

que encuentra el siguiente elemento del conjunto.

Store

Se utiliza para crear un nuevo registro:

```
Store <tipo de registro>
```

Más tarde se puede utilizar el operador connect (lo vemos más adelante)

Modify

Para modificar un registro existente (previamente localizado con find):

```
modify <tipo de registro>
```

Erase

Para eliminar un registro existente (previamente localizado con find):

```
erase <tipo de registro>
```

Connect

Para insertar un nuevo registro (previamente creado) en un conjunto ya existente:

```
Connect <tipo de registro> to <tipo de conjunto>
```

Disconnect

Operación contraria a connect

```
disconnect <tipo de registro> from <tipo de conjunto>
```

Reconnect

Cambiar el registro de un conjunto a otro (se especifica el nuevo conjunto)

```
reconnect <tipo de registro> to <tipo de conjunto>
```

5. Conclusión

Un sistema gestor de base de datos (SGBD) es un conjunto coordinado de programas, procedimientos y lenguajes que permite describir, recuperar y manipular la información almacenada en la base de datos, manteniendo su integridad, confidencialidad y seguridad.

Los sistemas jerárquico y de red constituyen la primera generación de los SGBD. El modelo jerárquico estructura los datos de manera escalonada, existiendo una relación del tipo padre-hijo entre sus registros. La representación de la información es similar a una estructura en árbol. El modelo en red, elimina la restricción del modelo jerárquico, en el sentido de que un registro puede tener más de un parente, por lo que resulta más flexible, dando lugar a una estructura de grafo. Pero estos sistemas presentan algunos inconvenientes: Es necesario escribir complejos programas de aplicación para responder a cualquier tipo de consulta de datos, por simple que ésta sea.

La independencia de datos es mínima ya carecen tienen un fundamento teórico y matemático profundo.

5.1. Relación del tema con el sistema educativo actual

Este tema es aplicado en el aula en los módulos profesionales siguientes, con las atribuciones docentes indicadas (PES/SAI):

Formación profesional básica

- Operaciones auxiliares para la configuración y la explotación(TPB en Informática de Oficina/ TPB en informática y Comunicaciones) (PES/SAI)
- Ofimática y archivo de documentos (TPB en Informática de Oficina) (PES/SAI)

Grado Medio

- Aplicaciones ofimáticas (GM de SMR) (PES/SAI)

Grado Superior

- Gestión de bases de datos (ASIR) (PES)
- Bases de Datos (DAW/DAM) (PES)

Bachillerato:

- 4º ESO – Tecnología de la Información y la comunicación (PES)
- Bachillerato – Tecnologías de la Información y la Comunicación (PES)

6. Bibliografía

- C.J. Date: **Introducción a los sistemas de bases de datos** Pearson, 2001.
- Elmasri, R.A. y Navathe S.B: "**Fundamentos de Sistemas de Bases de Datos**". Addison-Wesley, 3^a Edic, 2002.

- Olga Pons, J M Medina, M.A. Vila. **Introducción a los Sistemas de Bases de Datos** Edt Paraninfo (2005)
- Korth, H.F. y Silberschatz: "**Fundamentos de Bases de Datos**". McGraw -Hill, 4^a Edic., 2002.
- Garcia-Molina, H.; Ullman, J.D.; Widom, J. **Database systems: the complete book** - Pearson Education Limited, 2013.
- Abraham Silberschatz, Henry F. Korth, y S. Sudarshan, **Fundamentos de bases de datos** Edt. Mc Graw-Hill (2014)

