TEMA 2. MODELOS DE DATOS

1. Introducción

El objetivo de cualquier método de construcción de sistemas software debe ser construir sistemas robustos, fiables, portables y fáciles de mantener. Estos criterios coinciden completamente con lo que se considera una buena práctica de ingeniería. Por tanto, la producción de software debe ser vista como una disciplina de ingeniería en sí misma. En cualquier disciplina de ingeniería se establecen los principios teóricos que derivan en métodos estructurales de análisis, diseño e implementación. Las características genéricas de estas metodologías se han llevado al desarrollo de software mediante la introducción de un conjunto sistemático de procedimientos que van desde la concepción de un sistema mediante el análisis, pasando por su especificación, diseño, implementación, operación y mantenimiento.

Dentro de las metodologías convencionales de desarrollo de software, la noción de cómo evoluciona un sistema se representa por el concepto de *modelo de ciclo de vida* (MCV). El MCV indica el orden en el que las actividades de desarrollo deben ser realizadas. Las fases son, típicamente, especificación de requerimientos, diseño estructural, diseño detallado, implementación, integración y prueba. Cada fase se define en términos de las salidas que produce, las cuales (desde el punto de vista de la dirección) constituyen el criterio objetivo de progreso del desarrollo.

2. Enfoque metodológico

Construir un sistema de software de tipo medio o grande es una tarea que requiere planificación, organización y control. Requiere, además, usar técnicas formalizadas y estructuradas que guíen la realización. Requiere, en tercer lugar, aproximarse al desarrollo de software considerando el proceso completo de producción desde un punto de vista de ingeniería, estructurando el proceso en un conjunto de fases. Requiere, en definitiva, utilizar una metodología que aporte sistemática al propio proceso de construcción.

Una metodología añade estructura al proceso de desarrollo mediante tres componentes:

1. **Modelo de ciclo de vida** (MCV). Un MCV es una abstracción de cómo discurren el desarrollo general de un proyecto así como, internamente, cada una de las fases típicas en que este se divide.
2. **Métodos y técnicas**. Una metodología debe aportar soluciones para afrontar cada fase del ciclo de vida. Estas soluciones consistirán en técnicas y métodos de análisis, de diseño y programación. Si la metodología se basa en el uso de modelos, aportará primitivas para su construcción así como normas y guías de buena modelización.
3. **Criterios de progreso**. Toda metodología debe aportar criterios objetivos y tangibles de avance del proyecto estableciendo hitos claros de control, especificando los resultados que se deben obtener en cada fase en forma de documentos y/o productos software.

Al más alto nivel, todo ciclo de vida de desarrollo engloba actividades de análisis, diseño, implementación, instalación, utilización y mantenimiento, que modernamente se suelen recorrer mediante aproximación iterativa.

Las actividades de análisis son, quizá, las más importantes. El objetivo principal del análisis es construir un modelo correcto y preciso de esa parcela del mundo real que se pretende llevar al computador. Ese modelo debe dejar muy claro:

a) Cómo es el mudo que se modela; es decir, de qué cosas entenderá el sistema (plantas, animales, empresas, coches, etc.).
b) Qué cosas ocurren (o se desea que ocurran) en el mundo; es decir, qué funciones sabrá hacer el sistema (confeccionar pedidos, resolver ecuaciones, diagnosticar enfermedades, etc.).
c) Cuándo ocurren las cosas y a quién le ocurren; es decir, cómo se realizan las funciones en el tiempo y bajo qué condiciones, quién las pone en marcha y quiénes se ven afectados por su realización.

Las facetas anteriores toman forma mediante sus correspondientes modelos. Respectivamente, modelo de datos, modelo funcional y modelo dinámico.

3. Conceptos y técnicas del modelado

A través de la fase de análisis, el usuario del futuro sistema y el que desarrolla acuerdan de forma precisa lo que el sistema debe saber hacer.

¿Cómo se hace el análisis?
Se practica un tipo de análisis denominado "guiado por modelos". Esto significa que para hacer correctamente el análisis han de construirse una serie de modelos, cada uno de ellos capturando una parte del problema que se pretende resolver.

3.1. Concepto de modelo

La realidad del mundo que nos rodea es demasiado rica y compleja para la limitada capacidad de percepción y entendimiento del hombre. Ante esta situación, el hombre viene acercándose a la verdad objetiva (la realidad) por vía de simplificarla, eliminando todo lo que en un momento dado pueda considerarse superfluo y capturando lo esencial para sus propósitos. Estamos muy acostumbrados a escuchar frases como:

> *"En condiciones ideales de presión y temperatura..."*
> *"En ausencia de rozamiento..."*
> *"Asumiendo ausencia de monopolios en el mercado..."*

Por ejemplo, si se evalúa la solvencia de una persona para concederle un crédito, interesará saber su nivel de ingresos, su patrimonio, etc. pero, con toda probabilidad, no se considerará relevante el tamaño o color de sus ojos (salvo excepciones), ni si su nariz es aguileña o respingona.

Ese esfuerzo en capturar lo relevante y eliminar los detalles no significativos para nuestros objetivos, es lo que se conoce como **abstracción**. El conjunto de abstracciones que conforman una visión parcial, pero útil, de una parte del mundo es lo que se conoce como **modelo**. Se puede decir, por consiguiente, que el hombre se acerca a la realidad del mundo circundante a base de construir modelos que representan aspectos parciales de la misma. Esto también es aplicable para los productos que el hombre fabrica. Así, antes de la fabricación real de un automóvil, el ingeniero industrial diseñó multitud de modelos del mismo. Otro tanto podría decirse del arquitecto que construye casas, el ingeniero naval que construye barcos y, por supuesto, el ingeniero de software que construye sistemas de información.

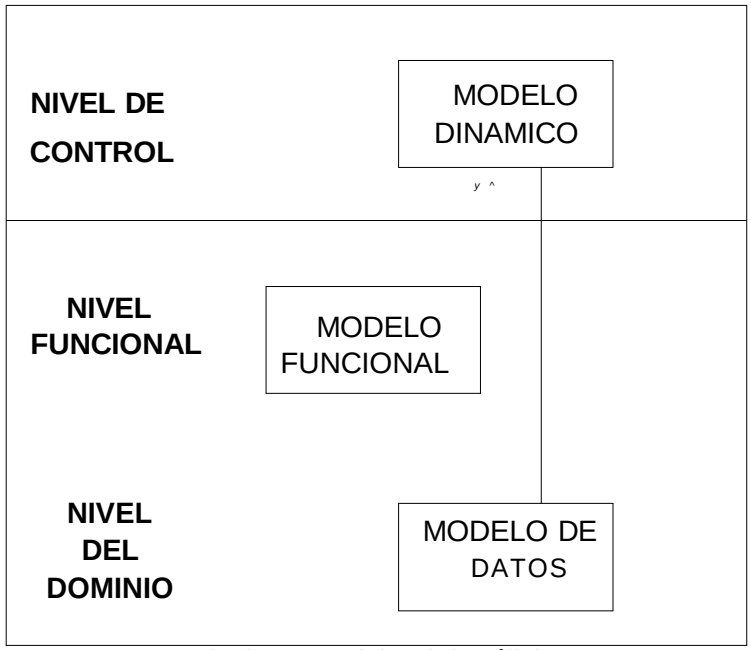
Todo ingeniero, cuando construye un modelo, utiliza un lenguaje (vocabulario) determinado y unas reglas de construcción de las que no puede salirse. Los elementos que pueden verse en un modelo, junto con las reglas de construcción, constituyen las **primitivas de modelización** (u ontología). El conjunto de primitivas que uno defina para construir modelos puede ser en cierta forma

arbitraria, en todo caso conviene que sea reducido. Al fin y al cabo, sea cual fuere la ontología que se defina, los modelos que a partir de ella se construyan serán siempre incompletos e imprecisos. Lo importante es que sean adecuados y útiles a nuestros objetivos.

3.2. Modelos del análisis.

El ingeniero puede llegar a construir tres modelos alineados en tres niveles. Estos modelos, evidentemente, no son independientes ya que representan enfoques distintos de una misma realidad.

La siguiente figura muestra la estructura de niveles y modelos. Cada nivel se ocupa de un tipo de conocimiento entre los que se necesitan para responder a las grandes preguntas del análisis: ¿que hay en el mundo (dominio)?, ¿Qué ocurre en el mundo (funcional)?, ¿Cuándo cambia el mundo (control)?



Niveles y modelos del análisis.

3.2.1. Modelo a nivel del dominio

El nivel del dominio busca modelar el conocimiento de tipo estático identificando, en el mundo del que va a entender el sistema, verdades de tipo universal (axiomas) referentes a los objetos presentes en el dominio, sus características relevantes y las relaciones y dependencias entre ellas. En el siguiente apartado se estudiará la forma de modelar a nivel del dominio. Baste, por ahora, con algunos ejemplos de afirmaciones típicas en diferentes dominios, expresados en lenguaje natural:

- "(en el dominio de la empresa) hay empleados, funciones y departamentos. Todo empleado tiene asignada alguna función y pertenece a un departamento. Toda función ha de ser desempeñada por alguien. A veces, incluso, por más de un empleado. Hay distintas categorías de empleado y todo empleado tiene derecho a percibir una remuneración acorde con su categoría..."
- "(en el dominio de las casas distribuidoras de vehículos) hay vehículos, de distintas marcas, modelos y niveles de equipamiento. Cada vehículo, según los parámetros anteriores, tiene asociado un precio base, que puede variar según la formula de pago elegida por el cliente y la estrategia comercial -descuentos, ofertas, etc.- del distribuidor. No toda formula de pago es

admisible para cualquier cliente, pues ello depende de si es un cliente antiguo, su capacidad económica, etc."

- "(en el dominio de la física dinámica) hay cuerpos en movimiento. Un cuerpo se caracteriza por tener masa y velocidad y ocupar una posición en el espacio. La cantidad de movimiento de un cuerpo es proporcional a su masa y su velocidad, y es constante siempre que el movimiento sea no acelerado..."
- "(en el dominio de la banca) hay clientes que solicitan créditos. Conceder un crédito es una operación con riesgo, que requiere estudio antes de su aprobación. Esto depende en gran medida de la cuantía de la operación y de ciertas características del cliente. Hay muchos perfiles de cliente, unos más adecuados que otros de cara a asumir riesgos. En todo caso, no es fácil decidir cual es el perfil de un cliente, ya que este puede poseer características de varios de los perfiles considerados..."
- "(en el dominio de una cuenca hidrológica) hay zonas de lluvia, que son superficies donde se asume comportamiento meteorológico uniforme. Asimismo, hay áreas receptoras de lluvia, que son superficies con un único punto de drenaje a una red de transporte de aguas. Las redes de transporte se componen de cauces de muchos tipos pero, básicamente, los hay de régimen rápido y régimen lento. Los primeros tienen pendientes altas y alta velocidad de transporte del agua; los segundos tienen bajas pendientes y baja velocidad de transporte, se sitúan en los tramos finales de las cuencas y, ante un caudal importante, tienen mayor riesgo de desbordamientos..."

3.2.2. Modelo funcional

El modelo funcional describe aquellos aspectos del sistema relativos a las transformaciones que puede sufrir la información contenida en el dominio. Se trata de capturar lo que el sistema ha de hacer, sin considerar cuándo o cómo lo hará.

3.2.3. Modelo dinámico

El modelo dinámico describe los aspectos del sistema relativos al tiempo, capturando el control del sistema: cuándo ocurren los cambios en el sistema, sin especificar en qué consisten esos cambios (aspecto funcional) ni cómo son implementados.

3.2.4. Relaciones entre los modelos.

De todos los niveles del análisis, el nivel del dominio es el que recoge el conocimiento más profundo sobre un tema. Si nos propusiéramos realizar un análisis de todo el saber humano, una gran parte del conocimiento científico se recogería a este nivel. El resto de los niveles buscan sacar partido del modelo del dominio, de manera que nos sea útil (o sea, responder a la pregunta ¿qué hacemos con el conocimiento del dominio para resolver nuestro problema?).

Lo que aquí nos interesa es la forma de representar los datos en el modelo del dominio y concretamente qué modelos de datos disponemos para representar esos datos en un sistema de base de datos.

4. El nivel del dominio: Modelos de datos.

- **Modelo**. Def. (Diccionario de la Lengua Española) Esquema teórico, generalmente en forma matemática, de un de un sistema o de una realidad compleja (por ejemplo, la evolución económica de un país), que se elabora para facilitar su comprensión y el estudio de su comportamiento.
-

- "***Modelar***" consiste en definir un mundo abstracto y teórico tal que las conclusiones que se puedan sacar de él coinciden con las manifestaciones aparentes del mundo real. [1]
- "***Modelo***": conjunto de conceptos que permiten construir una representación organizacional de la empresa.
- "***Modelo de datos***" es un dispositivo de abstracción que nos permite ver el bosque (la información contenida en los datos) en oposición a los árboles (valores individuales de los datos).[2]

Modelo: Instrumento que se aplica a una parcela del mundo real (Universo del Discurso) para obtener una estructura de datos a la que denominamos esquema.

MODELO:	Instrumento
ESQUEMA:	Resultado de aplicar el instrumento

El primer paso en el diseño de una Base de Datos es definir el Universo del Discurso fijando una serie de objetivos sobre el mundo real.

Definición de Modelo de Datos: *Conjunto de conceptos, reglas y convenciones que nos permiten describir los datos del universo del discurso, constituyendo una herramienta que facilita su interpretación y su representación en forma de datos en nuestro sistema de información*.

Los Modelos de Datos son la base para los lenguajes de datos. El nivel de abstracción de los lenguajes de datos es menor, ya que son el modelo más una sintaxis:

L.D. = M.D. + Sintaxis

El lenguaje puede venir tanto del modelo como de la sintaxis:

SQL = Modelo Relacional + Sintaxis
DL/I = Modelo Jerárquico + Sintaxis

Objetivos de todo modelo de datos:

a) **FORMALIZACIÓN**: Permite definir formalmente las estructuras permitidas y sus restricciones a fin de representar los datos, y también porque establece las bases para un lenguaje de datos.

b) **DISEÑO**: El modelo de datos es uno de los elementos básicos en el diseño de una Base de Datos.

Propiedades del Universo del Discurso:
- Estáticas (estructuras)
- Dinámicas (datos o valores que se almacenan en las estructuras)

Componentes de todo Modelo de Datos:
Estática
Objetos permitidos
Restricciones
    Inherentes
    De usuario
Dinámica
    Selección <condición>
        Navegacional
        Especificación
    Acción <Objetivo>
        Recuperación (previa selección)
        Actualización
        Modificación
        Inserción
        Borrado

4.1. DEFINICIÓN FORMAL DE MODELO DE DATOS

Propiedades del Universo del Discurso:
- Invariantes en el tiempo (ESTÁTICAS) à Estructuras.
- Varían en el tiempo (DINÁMICAS). Datos o valores que se almacenan en esas estructuras.

$$MD = <S,O>$$

S = Conjunto de reglas de generación que permiten representar la componente estática, es decir las estructuras del Universo del Discurso. Se corresponde con el Lenguaje de Definición de Datos.

O = Conjunto de operaciones autorizadas sobre la estructura que permite representar la componente dinámica. Se corresponde con el Lenguaje de manipulación de datos.

4.1.1. COMPONENTE ESTÁTICA

Está compuesta por:
    I.- Objetos permitidos.
    II.- Restricciones.

I.- Varían mucho de unos modelos a otros. En general son:
    Entidades
    Atributos
    Dominios sobre los que se definen los atributos
    Interrelaciones (asociaciones entre los objetos)

II.- No todas las estructuras están permitidas. Existen restricciones que invalidan ciertos objetos o asociaciones entre ellos.

- **Inherentes:** Impuestas por la misma naturaleza del modelo que no admite ciertos objetos o asociaciones. Introduce rigideces a la hora de modelar.

- **De usuario:** Permiten captar la semántica del Universo del Discurso que se quiere modelar. Facilitan la labor del diseñador.

La aplicación de la parte estática de un modelo a un UD. da como resultado un ESQUEMA o estructura de datos que representa dicho UD. en el correspondiente modelo.

4.1.2. COMPONENTE DINÁMICA

Los valores que toman los distintos objetos de un esquema en un momento determinado t_i se llaman OCURRENCIA del esquema o Base de datos en t_i (BD_i).
En otro momento t_j la ocurrencia será, en general, distinta. Se puede haber modificado la Base de Datos.
La componente dinámica del modelo consta de un conjunto de operaciones que se definen sobre la estructura del correspondiente modelo de datos (no todas las estructuras admiten las mismas operaciones).
La aplicación de una operación a una ocurrencia de un esquema transforma a ésta en una ocurrencia distinta:

$$O(BD_i) = BD_j$$

Tipos de operaciones:

SELECCIÓN
ACCIÓN

SELECCIÓN: Localizar una ocurrencia indicando un camino (sistema NAVEGACIONAL) o un conjunto de ocurrencias especificando una condición (ESPECIFICACIÓN).
ACCIÓN: sobre las ocurrencias previamente localizadas. Puede consistir en: Recuperación o Actualización.

Se puede expresar:

SELECCIÓN <CONDICIÓN>
ACCIÓN <OBJETIVO>

La distinción SELECCIÓN - ACCIÓN es de tipo formal, aunque algunos lenguajes, como DL/I, tienen verbos distintos para cada operación. SQL reúne ambas operaciones en una única sentencia.

Modelos

- Conceptual (Lógico)
  - ME/R
- Convencional (Físico)
  - Red
  - Jerárquico
  - Relacional

El presente apartado pretende dar una visión general de los modelos en red y jerárquico, los cuales, si bien están hoy ampliamente superados por el modelo relacional, han gozado de gran popularidad durante muchos años y han sido de gran importancia para el desarrollo de la tecnología de bases de datos.

5. ANTECEDENTES HISTÓRICOS.

Las redes constituyen una manera natural de representar las interrelaciones entre diferentes objetos. Se utilizan ampliamente en las matemáticas, la investigación operativa, la física, la química y otros campos. Como los objetos y sus interrelaciones constituyen maneras útiles de modelar muchos de los fenómenos que nos conciernen en los negocios, no es sorprendente que el modelo de datos en red se aplique también a la organización de bases de datos.

De manera general, las redes pueden representarse mediante una estructura matemática denominada grafo orientado. Estos grafos orientados tienen una estructura simple: se construyen mediante nodos conectados por arcos orientados o aristas. En el contexto de los modelos de datos, los nodos pueden considerarse como tipos de registros de datos y las aristas pueden considerarse como las interrelaciones entre ellos. La estructura de grafo facilita la representación simple de todo tipo de interrelaciones entre los datos (jerárquicas, de pertenencia, etc.). 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 desee. Además, una vez que se ha establecido una relación entre dos objetos, la recuperación y la manipulación de los datos asociados puede realizarse eficientemente.

El modelo en red general es muy flexible debido a la inexistencia de restricciones inherentes, pero por esa misma razón su implementación e instrumentación en el nivel físico resulta difícil y poco eficiente. Debido a esto, al llevar a la práctica modelos basados en red se introducen restricciones para facilitar su implementación que se hacen más restrictivas en el modelo jerárquico.

Como se verá más adelante, una jerarquía es un caso particular de una red. En consecuencia, el modelo de datos jerárquico es un caso particular del modelo en red. Aunque el modelo de datos jerárquico precede cronológicamente al modelo en red, nos parece adecuado discutir éste en primer lugar por ser más general.

La organización no oficial ***Conference on Data Systems Languages*** (**CODASYL**), formada por representantes de fabricantes de hardware, fabricantes de software y los principales usuarios (entre los que se encontraban miembros de varios departamentos del gobierno de Estados Unidos), inicialmente desarrolló y normalizó el lenguaje COBOL (*Common Business Oriented Language*) a principios de la década de los 60.

En 1965 se crea dentro de CODASYL el ***List Processing Task Force***, dedicado a estudiar las capacidades del proceso de listas para el lenguaje COBOL. Dos años más tarde, este grupo cambió su nombre por ***Data Base Task Group*** (**DBTG**) haciendo notar el creciente interés que despertó hacia mediados y finales de los años sesenta el desarrollo de normas y estándares para los sistemas de bases de datos. Este subgrupo estaba fuertemente influido por la arquitectura utilizada para el primero de los Sistemas de Gestión de Bases de Datos (SGBD): el ***Integrated Data Store*** (**IDS**) construido por uno de sus integrantes, C.W. Bachman, para la compañía General Electric. Bachman propuso la estructura de datos propia del modelo CODASYL y los diagramas que llevan su nombre. Esta influencia se aprecia en las recomendaciones para un modelo

en red que aparecen publicadas en un informe preliminar en 1969. El informe recogía la especificación de la sintaxis y semántica de un lenguaje de definición de datos (LDD) que permitía la descripción de una base de datos y un lenguaje de manipulación de datos (LMD) que se configuró como extensión de un lenguaje anfitrión para el manejo (recuperación y actualización) de los datos almacenados en la base de datos. Este lenguaje de manipulación se proponía dar respuesta a las necesidades de gestión de datos para cualquier tipo de lenguaje anfitrión (en teoría) pero en la práctica, su orientación a COBOL era evidente. En las propuestas de este informe, se definía una arquitectura de base de datos a dos niveles (esquema y subesquemas), lo que dificultaba conseguir la independencia física/lógica deseable en un sistema de base de datos. Este aspecto, junto con la orientación al COBOL, fueron los más criticados. Este primer informe dio lugar a numerosas sugerencias para su perfeccionamiento y se publicó un informe oficial revisado en 1971[3] que se sometió a la consideración del *American National Standards Institute* (ANSI) para su posible adopción como norma nacional para los SGBDs. La organización ANSI no realizó acción alguna al respecto pues estaba desarrollando su propia recomendación,[4] y los informes modificados publicados en 1978 y 1981 sucedieron al informe de 1971. El informe de 1978 propone una nueva arquitectura a tres niveles que consigue una mayor independencia física/lógica, coincidiendo con la publicación de los resultados de los trabajos del comité ANSI para bases de datos que proponían la conocida arquitectura a tres niveles.[5] A pesar de su carácter no oficial, los informes y normas de CODASYL obtuvieron una amplia difusión y respaldo debido en gran medida al apoyo que obtuvieron por parte del Departamento de Defensa de los Estados Unidos.

El grupo CODASYL se disolvió en el año 1983 debido a que la organización oficial ANSI estaba trabajando sobre una propuesta de lenguaje de datos para modelos en red, culminando éstos con su recomendación NDL/ANS de un lenguaje normalizado de datos en red.

La existencia de varias versiones de los lenguajes CODASYL ha ocasionado que las diferencias entre ellas se vieran reflejadas en los diferentes productos que, en general, se desviaban en mayor o menor medida de las recomendaciones, así como en la bibliografía al respecto, que no coincide al presentar la sintaxis de los lenguajes CODASYL.

Sin embargo, el documento de 1971 permanece como la proposición fundamental del modelo en red, que se convirtió en el modelo CODASYL, de DBTG y que ha servido de base para el desarrollo de los SGBD en red de diferentes fabricantes, entre los que destacan el IDS (Honeywell) y el IDMS (Computer Associates) como dos de las implementaciones comerciales más conocidas.

Si bien es verdad que en la actualidad los SGBD basados en los modelos de tipo red ya no dominan el mercado como ocurría hace unos años, también es cierto que todavía hay bastantes instalaciones con sistemas jerárquicos y CODASYL que están respondiendo con eficiencia a las necesidades de los usuarios

6. EL MODELO EN RED CODASYL

Como en todo modelo de datos, distinguiremos sus componentes estática y dinámica.

En el esquema se describen los aspectos estáticos, es decir, la parte estructural de los datos (tipos de entidades, tipos de interrelaciones, etc.), representadas, como ya se ha indicado, en forma de grafo.

En cuanto a la dinámica, los modelos en red se caracterizan por ser navegacionales, es decir, la recuperación y la actualización de la base de datos se lleva a cabo registro a registro, y procedimentales, es decir, asumen el conocimiento de la estructura interna de la base de datos por parte del programador.

6.1. Correspondencia del modelo en red con la arquitectura de tres niveles de ANSI/X3/SPARC

La correspondencia de la arquitectura de tres niveles del grupo ANSI/X3/SPARC con el modelo en red de CODASYL es como se indica a continuación:

- El nivel conceptual se denomina **esquema**.
- El nivel externo se denomina **subesquema**.
- El nivel interno está implícito en la implementación.

6.2. COMPONENTE ESTÁTICA. REGISTROS Y CONJUNTOS.

Hay dos estructuras de datos fundamentales en el modelo en red: los **tipos de registros** y los **conjuntos**. Los tipos de registros se definen habitualmente como colecciones de elementos de los datos lógicamente relacionados. Por ejemplo, el tipo de registro de un cliente podría incluir los elementos de datos siguientes: Identificador del cliente, Nombre del cliente, Dirección. El registro está a su vez compuesto de **campos** o **elementos de datos** que es la unidad de datos más pequeña a la que se puede hacer referencia en el modelo CODASYL. Un tipo especial de campo es el **agregado de datos**, que consiste en un vector con un número fijo de elementos, como por ejemplo la fecha, compuesta de día, mes y año.

Un **conjunto** expresa una interrelación uno a muchos, o uno a uno entre dos tipos de registros. En el modelo CODASYL, el conjunto constituye el elemento básico para la representación de las interrelaciones. Por medio de los conjuntos se establecen asociaciones 1:N (las asociaciones 1:1 son un caso particular de las 1:N) a dos niveles, en las que el nodo raíz se denomina **propietario** y los nodos dependientes se denominan **miembros**.

Puede definirse un tipo de conjunto especial denominado **conjunto singular** en el que el propietario sería ficticio (el sistema) y el miembro el tipo de registro cuyas ocurrencias se quieren encadenar. De esta forma, tendríamos las ocurrencias de un registro como si se tratara n de un fichero tradicional con estructura de lista circular.

Las características generales del modelo en red CODASYL pueden resumirse en las siguientes:

- Un **conjunto** es una colección nominada de dos o más tipos de registros que representa una interrelación 1:N (recuérdese que las interrelaciones 1:1 son un caso particular de las 1:N)
- Cada conjunto debe tener obligatoriamente un tipo de registro propietario y uno o más registros miembros.
- Pueden existir conjuntos singulares en los que el propietario es el sistema.
- No existe ninguna limitación en cuanto al número de conjuntos que pueden definirse en el esquema.
- Cualquier registro puede ser declarado propietario de uno o varios conjuntos.
- Cualquier registro puede ser declarado miembro de uno o varios conjuntos.
- Cualquier registro puede ser declarado propietario en un conjunto y miembro en otr o conjunto distinto.
- Aunque un tipo de registro se haya declarado miembro de un conjunto, existe la posibilidad de no encadenar en el conjunto ciertas ocurrencias del mismo, las cuales no pertenecerán a ningún propietario, es decir, quedarán sueltas respecto a ese conjunto.

6.3. RESTRICCIONES INHERENTES AL MODELO CODASYL

Están vinculadas al concepto de conjunto y son las siguientes:

- Sólo se admiten tipos de interrelaciones jerárquicas a dos niveles: nivel propietario y nivel miembro. Las jerarquías multinivel y las estructuras en red se consiguen combinando varios conjuntos.
- En el nivel propietario de un conjunto sólo se permite un tipo de registro.
- Una misma ocurrencia de un registro miembro no puede pertenecer en un conjunto a más de un propietario.

6.4. DINÁMICA DEL MODELO CODASYL

La componente dinámica de un modelo de datos es el conjunto de operaciones que, aplicadas a un estado de la base de datos, la transforman en otro estado distinto. El lenguaje de manipulación de datos de CODASYL es de tipo navegacional (opera registro a registro), procedimental (requiere el conocimiento de la estructura física de la base de datos por parte del programador) y tiene que estar embebido en un lenguaje de programación anfitrión.

Como característica importante, destacar que en él se distingue la localización o selección de la acción (recuperación o actualización). La navegación por la base de datos se lleva a cabo apoyándose en los indicadores de registro activo, concepto fundamental en la componente dinámica del modelo CODASYL.

7. EL MODELO JERÁRQUICO

Veremos ahora el modelo jerárquico como un caso particular del modelo en red. Aunque los productos basados en este modelo han perdido cuota de mercado y se consideran altamente superados por la tecnología relacional, aún persisten muchas aplicaciones basadas en este modelo

Al igual que los modelos en red, los modelos jerárquicos se encuentran entre los primeros modelos utilizados en SGBD comerciales, desarrollados a principios de la década de los 60.

Las estructuras en árbol, muy extendidas en la informática para representar estructuras de datos, tienen como característica su poca flexibilidad, lo que origina la falta de adaptación a muchas situaciones reales.

Aunque nunca se ha llegado a una formalización matemática del modelo ni de sus lenguajes, como ocurre con el modelo relacional, ni tampoco se ha pretendido su estandarización como sucedió con CODASYL, los productos basados en el modelo jerárquico alcanzaron altas cuotas de mercado y aunque actualmente están superados por la tecnología relacional, persisten importantes aplicaciones soportadas en estos productos trabajando muy eficientemente en grandes sistemas de transacciones dirigidas que requieren respuestas rápidas, siempre que las aplicaciones desarrolladas sobre ellos se mantengan sin apenas cambios. Una razón complementaria para la persistencia de los sistemas basados en el modelo jerárquico es que muchas estructuras son inherentemente jerárquicas. Por ejemplo, una universidad puede tener departamentos (primer nivel) los departamentos tienen profesores (segundo nivel) y los profesores imparten asignaturas (tercer nivel). Aunque la anterior estructura de datos podría representarse mediante un modelo en red, más robusto en cuanto a la capacidad de representación, la complejidad del sistema sería muy superior a la necesaria. Esta fue una de las razones que movió al desarrollo del modelo jerárquico, como se describe más adelante.

Debido a la inexistencia de formalización y estándares, su estudio requiere el examen de los SGBD empleados en la práctica. Afortunadamente, las implementaciones de sistemas de bases de datos jerárquicas están dominadas por un sistema: el IMS (*Information Management System*) de IBM, con su lenguaje de datos DL/I (*Data Language/One*). De hecho, es el único que ha sobrevivido hasta nuestros días.

El sistema IMS se desarrolló como fruto de un esfuerzo conjunto entre IBM y North American Aviation (posteriormente convertida en Rockwell) para desarrollar un SGBD para dar soporte al proyecto Apolo (uno de los mayores proyectos de ingeniería acometidos en aquellos tiempos). Un factor clave en el desarrollo de IMS fue la necesidad de manipular millones de piezas que se relacionaban unas con otras según una estructura jerárquica: las piezas más pequeñas se usaban para construir montajes más grandes que a su vez se empleaban como componentes de módulos más grandes y así sucesivamente. A su vez, esta estructura jerárquica permitía conocer con exactitud y rapidez la estructura de una pieza.

En la bibliografía, las exposiciones sobre el modelo jerárquico invariablemente incorporan la terminología y las convenciones propias de IMS, de modo que nosotros no nos alejaremos de esta tendencia.

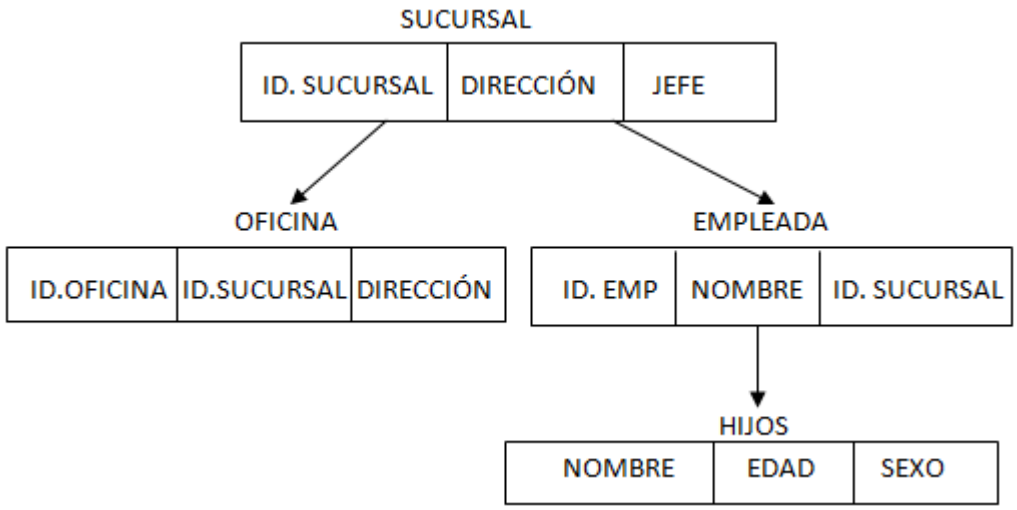
La implementación física del modelo jerárquico se basa en apuntadores. La estructura física (organización y tipo de punteros) varía de un producto a otro, incluso un mismo producto (IMS) proporciona diferentes organizaciones físicas con el fin de que el diseñador pueda obtener la mayor eficiencia en el diseño físico de la base de datos.

7.1. CARACTERÍSTICAS DE LA ESTRUCTURA JERÁRQUICA

En el modelo jerárquico, el esquema es una estructura en forma de árbol compuesta por nodos, que representan las entidades, conectados por arcos, que representan las interrelaciones entre esas entidades. En terminología IMS, los nodos se llaman **segmentos**.

Como ya se ha apuntado, la estructura de datos jerárquica es un caso particular de la del modelo en red, con importantes restricciones adicionales derivadas del hecho de que las asociaciones en el modelo jerárquico deben formar un árbol ordenado, es decir, un árbol en el que el orden de los nodos es importante.

La siguiente figura muestra un ejemplo de este tipo de organización:



Una estructura jerárquica tiene las siguientes características:

- El árbol se organiza en un conjunto de niveles.
- El más alto de la jerarquía es el nodo **raíz**.
- Los arcos que unen los nodos representan las asociaciones jerárquicas entre dos entidades y no tienen nombre (a diferencia del modelo en red CODASYL). No es necesario, ya que entre dos conjuntos de datos sólo puede existir un tipo de interrelación.
- Un nodo de nivel superior (nodo padre) puede tener un número ilimitado de nodos hijos, pero un nodo hijo sólo puede tener un único nodo padre.
- Todo nodo, a excepción del raíz, ha de tener obligatoriamente un padre.

- Llamamos **hojas** a los nodos que no tienen descendientes.
- Al número de niveles de la estructura jerárquica se le llama **altura** del árbol.
- Al número de nodos del árbol se le llama **momento**.
- Sólo están permitidas las estructuras de tipo 1:N (las de tipo 1:1 son un caso particular de las 1:N).
- Cada nodo no terminal y sus descendientes forman un subárbol, de modo que un árbol es una estructura recursiva.

En el modelo jerárquico el árbol se recorre en preorden: primero el subárbol izquierdo y después el subárbol derecho.

7.2. RESTRICCIONES INHERENTES AL MODELO JERÁRQUICO

Las restricciones en el modelo jerárquico son más severas que en el modelo en red por tratarse de un caso particular de éste:

- No pueden representarse directamente relaciones N:M ni reflexivas.
- No se admite más de una relación entre dos segmentos.
- No puede existir un segmento hijo con más de un padre.
- El árbol debe recorrerse siempre en preorden.
- Cualquier acceso a la base de datos debe comenzar necesariamente en el segmento raíz.

7.3. CORRESPONDENCIA DEL MODELO JERÁRQUICO DE IMS CON LA ARQUITECTURA DE TRES NIVELES DE ANSI/X3/SPARC

La correspondencia de la arquitectura de tres niveles del grupo ANSI/X3/SPARC con el modelo jerárquico de IMS es como se indica a continuación:

- El nivel conceptual se compone de un conjunto de árboles definidos mediante una estructura denominada ***Data Base Description*** (**DBD**).
- El nivel externo se compone de subconjuntos de los árboles definidos en el nivel conceptual.
- El nivel interno está implícito en la implementación.

IMS gestiona un conjunto de árboles, cada uno de ellos definido mediante una DBD. El conjunto de todas las DBDs definidas en una base de datos constituye el nivel conceptual.

Las diferentes aplicaciones accederán a una parte de la base de datos. Cada uno de los subesquemas del nivel externo en IMS se llama **base de datos lógica**. Cada una de ellas estará compuesta por determinados segmentos de uno o varios árboles. La parte de la base de datos lógica que afecta a cada uno de los árboles se define mediante una estructura denominada ***Program Control Block*** (**PCB**). El agrupamiento de los PBCs para formar una base de datos lógica forma un ***Program Specification Block*** (**PSB**). De esta forma, dentro de un PSB para definir una base de datos lógica habrá tantos PCBs como árboles del nivel físico participen en esa "vista" del nivel externo.

7.4. LA MANIPULACIÓN DE LOS DATOS

El acceso a los datos en IMS se realiza a través de su LMD, DL/I. A igual que ocurre con el sistema en red, es un lenguaje navegacional (un solo registro cada vez), procedimental (implica el conocimiento de la estructura jerárquica por parte del programador) y de tipo huésped (se embebe dentro de un lenguaje anfitrión).

8. LOS MODELOS EN RED Y EL MODELO RELACIONAL

Analizaremos, para finalizar, las principales diferencias entre los modelos en red y el modelo relacional, empezando por su origen.

Los modelos no relacionales no se desarrollaron con base en ningún modelo abstracto de datos predefinido. Por el contrario, los modelos que existen se definieron a posteriori por un proceso de abstracción o inducción a partir de las implementaciones ya existentes.

El modelo relacional fue el primer ejemplo de modelo de datos definido antes de cualquier implementación. Los sistemas relacionales fueron los primeros que se construyeron de acuerdo con los preceptos de un modelo de datos predefinido. El modelo en red fue el fruto del intento de establecer estándares detallados para sistemas de bases de datos. El modelo jerárquico surgió de la necesidad de resolver un problema concreto.

8.1. REPRESENTACIÓN DE LOS DATOS.

Una diferencia importante entre el modelo de datos en red y el relacional es la manera en que se representan las interrelaciones. En el modelo relacional, los enlaces entre dos relaciones se establecen incluyendo un atributo definido sobre el mismo dominio en ambas relaciones. Las filas que están lógicamente relacionadas tendrán en cada relación los mismos valores para ese atributo. En el modelo en red CODASYL la cardinalidad 1:N entre dos tipos de registros se establece mediante la definición explícita del tipo de conjunto. Es entonces cuando el SGBD conecta los registros en cada tipo de conjunto mediante punteros físicos.

Esto significa que los registros se conectan físicamente cuando participan en la misma ocurrencia de un conjunto. Esta representación explícita de los tipos de conjunto se ha considerado como una ventaja del modelo en red. Un argumento a tener en cuenta es que el modelo en red utiliza dos elementos de modelado: el tipo de registro y el tipo de conjunto, mientras que el modelo relacional utiliza un único concepto: la relación.

8.2. LENGUAJE DE MANIPULACIÓN DE DATOS.

Los sistemas no relacionales están en un nivel de abstracción mucho más bajo que los relacionales.

Las operaciones de navegación y de recuperación del LMD (Lenguaje de Manipulación de Datos) de los modelos en red se efectúan sobre registros individuales (lenguaje de tipo navegacional), en contraste con las operaciones del modelo relacional que se efectúan sobre relaciones completas (lenguaje de especificación). Estas operaciones deben estar inmersas en un lenguaje de programación anfitrión, es decir es un lenguaje de tipo huésped. Los LMD relacionales pueden utilizarse directamente o pueden estar embebidos en un lenguaje anfitrión. Como las operaciones de manipulación orientadas a los registros se basan en las operaciones tradicionales de procesamiento de archivos, el programador debe estar íntimamente familiarizado con las características de almacenamiento y los modos de acceso (es un lenguaje muy procedimental).

Los modelos de datos de los sistemas no relacionales pueden verse como abstracciones de algunas de las estructuras de almacenamiento subyacentes y sus operadores asociados, por lo que son, en general, sistemas de programación; el usuario primario es un programador que utiliza un lenguaje tipo COBOL y necesita navegar de manera manual en la base de datos. Cualquier optimización la debe realizar el programador, no el sistema

En el modelo relacional el LMD es muy poco procedimental, es decir, el usuario nada tiene que saber acerca de la organización interna de los datos ni de los modos de acceso. Esta ventaja del modelo relacional sobre el modelo en red se confirma por la adición al IDMS de un interfaz de usuario de tipo relacional, convirtiéndolo en IDMS/R. Esto dio lugar a la popularización de los interfaces de usuario relacionales en sistemas no relacionales lo que a su vez provocó la publicación de las doce reglas de Codd.[6]

8.3. RESTRICCIONES DE INTEGRIDAD

El modelo en red CODASYL proporciona un conjunto básico de restricciones de integridad, en particular para proteger la integridad de los conjuntos, que permiten al diseñador determinar cómo se deben comportar los registros propietarios respecto a los miembros y viceversa. Otros tipos de restricciones semánticas sólo pueden implementarse en el programa de aplicación, lo que dificulta gravemente el mantenimiento de dichas restricciones semánticas. En un sistema relacional, la mayoría de las restricciones semánticas pueden ser recogidas por el SGBD, asegurando su cumplimiento por cualquier programa de aplicación.

8.4. IMPLEMENTACIÓN

Los sistemas en red son especialmente convenientes para sistemas de bases de datos en los que se requieran características como:

- Gran tamaño.
- Consultas repetitivas bien definidas.
- Transacciones bien definidas.
- Aplicaciones bien definidas.

Si concurren todos estos factores, un sistema en red es una buena solución. Por el contrario, las futuras necesidades de información que requieran consultas no definidas a la base de datos pueden requerir una reorganización de la base de datos con un alto grado de dificultad. Los sistemas relacionales tienen clara ventaja en este aspecto, pues las demandas inesperadas de información pueden satisfacerse sin modificación alguna en la estructura interna de la base de datos, simplemente definiendo un nuevo subesquema en el nivel externo.