Modelos de datos

PID_00271453

Blas Torregrosa García

Tiempo mínimo de dedicación recomendado: 2 horas

Universitat Oberta de Catalunya

Blas Torregrosa García

Ingeniero en Informática y máster universitario en Seguridad de las Tecnologías de la Información y de las Comunicaciones (MISTIC) por la Universitat Oberta de Catalunya (UOC). Especializado en ciberseguridad. Profesor colaborador en el máster de Ciencia de Datos de la UOC y profesor asociado en la Universidad de Valladolid (UVA).

El encargo y la creación de este recurso de aprendizaje UOC han sido coordinados por el profesor: Ferran Prados Carrasco (2020)

Primera edición: febrero 2020

Av. Tibidabo, 39-43, 08035 Barcelona
Realización editorial: FUOC

Índice

Introducción..... 5

1. Definiendo modelo de datos..... 7
1.1. Estructuras de los modelos de datos 7
1.1.1. Datos estructurados 7
1.1.2. Datos no estructurados 9
1.1.3. Datos semiestructurados 10
1.2. Operaciones con datos 10
1.2.1. Subconjunto 10
1.2.2. Unión 11
1.2.3. Proyección 11
1.2.4. Conexión (*join*) 12
1.3. Restricciones de los datos 12

2. Niveles de modelado de datos..... 14
2.1. Modelado conceptual de datos 14
2.2. Modelado lógico de datos 15
2.3. Modelado físico de datos 16

3. Tipos de modelos de datos..... 18
3.1. Modelo jerárquico 18
3.2. Modelo relacional 18
3.3. Modelo en red 20
3.4. Modelo orientado a objetos 20

Bibliografía..... 23

Introducción

Los **modelos de datos** son una manera de estructurar y organizar los datos para que se puedan utilizar más fácilmente. Introduciremos diferentes estructuras de datos, las operaciones con datos y las distintas restricciones que podemos imponer a los datos. Asimismo, veremos los diferentes niveles de modelado de datos y, finalmente, los tipos de modelos de datos.

1. Definiendo modelo de datos

Un **modelo de datos** determina la manera en la que se organizan y estructuran los datos. Los modelos de datos son el núcleo del almacenamiento, análisis y procesamiento de los sistemas que gestionan datos.

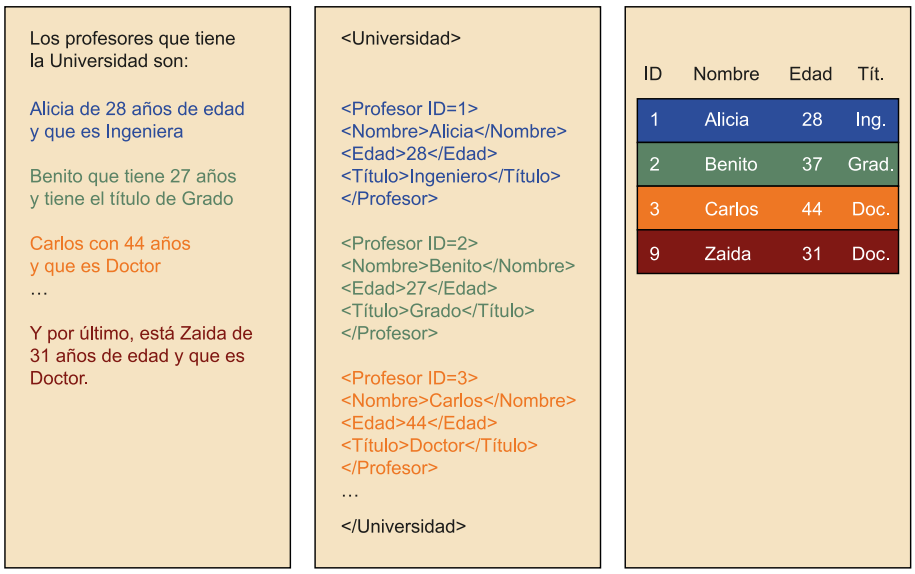
Los principales tipos de estructuras son datos estructurados, semiestructurados y no estructurados. También examinaremos diferentes técnicas de modelado aplicadas a estos tipos de datos.

1.1. Estructuras de los modelos de datos

Los modelos de datos tratan y describen una gran variedad de características de los datos. Por su origen, hay tres principales tipos de datos:

- Datos estructurados
- Datos no estructurados
- Datos semiestructurados

Figura 1. Diferencia entre datos no estructurados, semiestructurados y estructurados



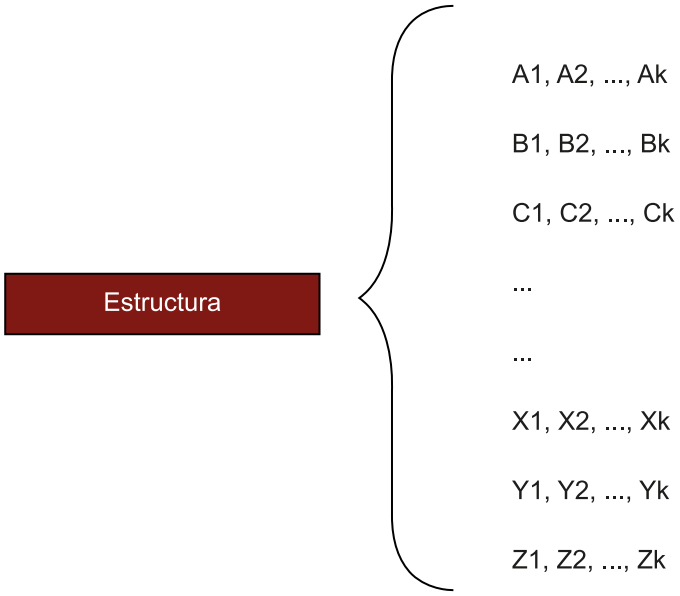
1.1.1. Datos estructurados

Los datos estructurados son datos que tienen una determinada longitud y un determinado formato.

Algunos ejemplos de tipos de datos estructurados incluyen los números, fechas y grupos de palabras y números denominados **cadenas**.[1] En general, los datos estructurados siguen un patrón como el de la figura 2.

[1] En inglés, *strings*.

Figura 2. Forma general de datos estructurados



Habitualmente, los **datos estructurados** tienen predefinido el número de columnas, es decir, el número *k* es fijo. Algunas de esas columnas podrían no siempre tener datos, siendo en este caso todavía datos estructurados.

La mayor parte de los datos estructurados se puede clasificar en datos generados por máquinas y datos generados por personas. Entre los datos estructurados generados por máquinas se incluyen:

- Datos de sensores, por ejemplo, las tarjetas de identificación por radiofrecuencia (RFID), medidores inteligentes, dispositivos médicos, sensores en relojes inteligentes.
- Datos del sistema de posicionamiento global (GPS).
- Datos de registro de actividad (*logs*).
- Datos financieros.
- Datos de puntos de venta.

Los datos generados por el ser humano incluyen actividades y eventos en las redes sociales:

- **Datos de entrada**, puede ser cualquier dato que una persona pueda introducir en un sistema informático, como su nombre, teléfono, edad, correo

electrónico, ingresos, direcciones físicas o respuestas de encuestas. Son útiles para entender el comportamiento de los usuarios.

- **Datos de *clickstream*** (tráfico), cuando los usuarios navegan a través de sitios web o redes sociales generan una gran cantidad de datos. Estos datos se pueden registrar y analizar para determinar el comportamiento de los clientes o los patrones de compra, o para descubrir defectos en los procesos.

- **Datos de videojuegos** que incluyen la actividad del usuario mientras juegan por Internet a videojuegos en una variedad de plataformas, como ordenadores, móviles y consolas.

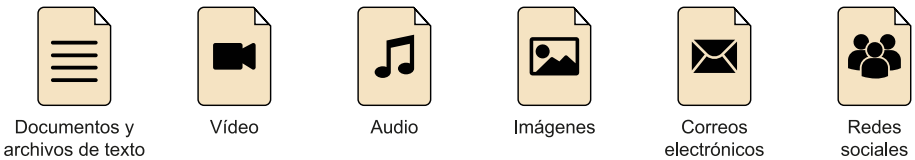
1.1.2. Datos no estructurados

Los datos que no se ajustan a un modelo de datos o a un esquema de datos se conocen como **datos no estructurados**.

Cualquier documento que se componga principalmente de texto, con poca o ninguna estructura que describa el contenido del documento, se puede clasificar como datos no estructurados. Los datos no estructurados se diferencian de los datos estructurados en el sentido de que su estructura no es predecible.

Algunos ejemplos de datos no estructurados son documentos, correos electrónicos, blogs, informes, notas, imágenes digitales, vídeos e imágenes por satélite. También pueden ser datos no estructurados algunos datos generados por máquinas o sensores (figura 3). De hecho, los datos no estructurados representan la mayoría de los datos (se estima que en torno al 80 %) tanto dentro como fuera de las organizaciones, así como en la web (LinkedIn, Twitter, Snapchat, Instagram, YouTube, Facebook, etc.).

Figura 3. Varios tipos de datos no estructurados



Los datos no estructurados también se pueden clasificar como generados por máquina o generados por personas. La mayoría de los conjuntos de datos **generados por máquinas** son imágenes de satélites, datos científicos, fotografías y vídeos. La mayoría de los conjuntos de datos no estructurados **generados por personas** son documentos, datos de redes sociales, datos de dispositivos móviles o sitios web.

Técnicamente, tanto los archivos de texto como los archivos de audio o vídeo tienen una estructura definida por el propio formato de archivo, pero este aspecto no es importante aquí. La idea de «no estructurado» depende del contenido del archivo y no del formato.

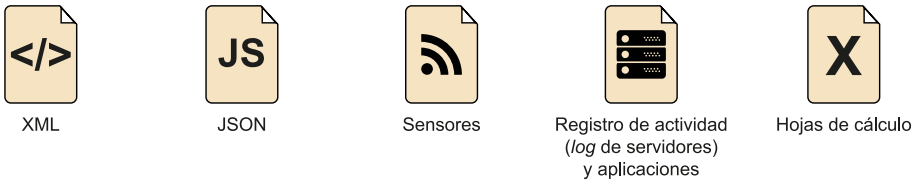
1.1.3. Datos semiestructurados

En ocasiones, los datos llevan adjuntos etiquetas de metadatos que proporcionan información y contexto sobre el contenido de los datos. Estos datos se consideran **datos semiestructurados**.

La línea divisoria entre datos no estructurados y semiestructurados no es fácil de establecer, aunque algunos autores consideran que incluso los datos no estructurados tienen cierto grado de estructura.

Los datos semiestructurados tienen un nivel definido de cierta estructura y coherencia, pero no son de naturaleza relacional. En su lugar, los datos semiestructurados suelen ser jerárquicos o grafos. Este tipo de dato se almacena habitualmente como archivo de texto. Los archivos XML o JSON son formas frecuentes de almacenamiento de datos semiestructurados (figura 4).

Figura 4. Algunos formatos de datos semiestructurados



1.2. Operaciones con datos

El segundo componente de un modelo de datos es un conjunto de operaciones que se pueden realizar en los datos.

Las **operaciones** indican los métodos para tratar los datos.

Puesto que los diferentes modelos de datos suelen estar asociados con estructuras diferentes, las operaciones también serán diferentes, aunque algunos tipos de operaciones se pueden realizan en todos los modelos de datos.

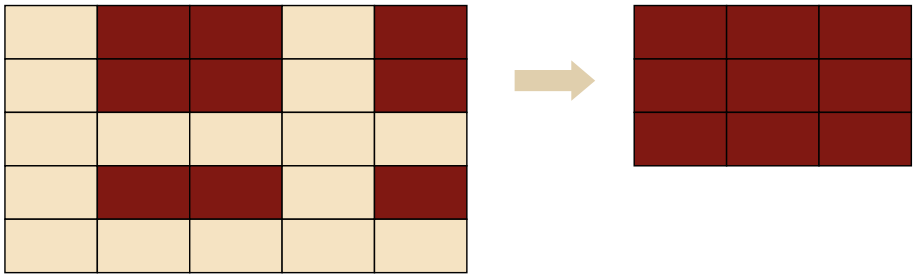
1.2.1. Subconjunto

A menudo, cuando estamos trabajando con un conjunto de datos grande, solo necesitaremos una parte de él para su análisis o tratamiento.

El proceso de generar **subconjuntos** (*subsetting*) consiste en extraer las variables y observaciones necesarias para una operación.

Dependiendo del contexto, también se denomina **selección** o **filtrado**. Un subconjunto puede estar formado por un número indeterminado de campos (columnas) y datos (filas).

Figura 5. Ejemplo de subconjuntos

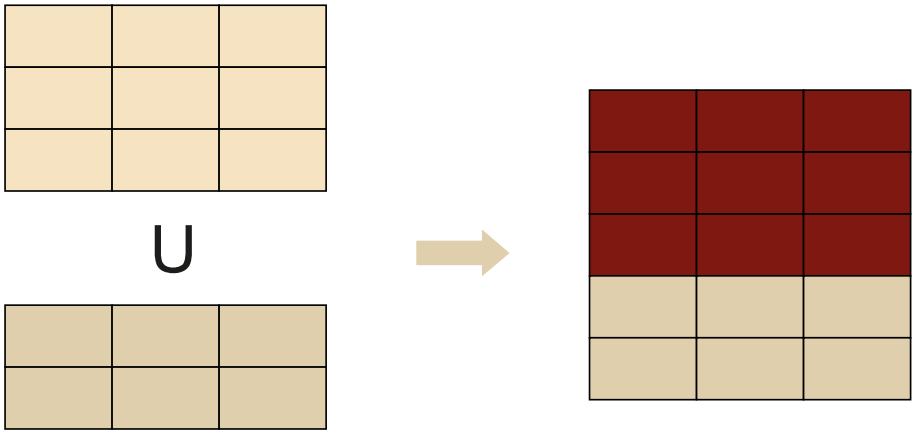


1.2.2. Unión

La suposición que subyace a la operación de **unión** es que los datos involucrados tienen la misma estructura.

Consiste en incluir todos los datos de los diferentes conjuntos de datos en uno solo. Esta operación también elimina los datos duplicados.

Figura 6. Ejemplo de unión

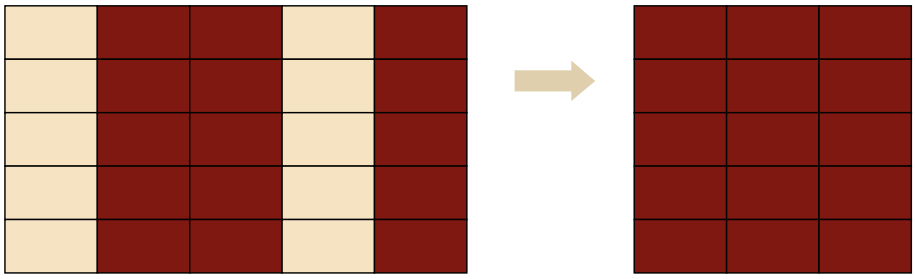


1.2.3. Proyección

Otra operación usual consiste en recuperar una parte concreta de los datos. En este caso, especificamos que estamos interesados solo en ciertos campos de una colección de datos. Esto produce una nueva colección de datos que

contiene exclusivamente esos campos. La diferencia con los subconjuntos es que en la **proyección** se incluyen todos los datos (filas) de cada uno de los campos seleccionados (columnas).

Figura 7. Ejemplo de proyección

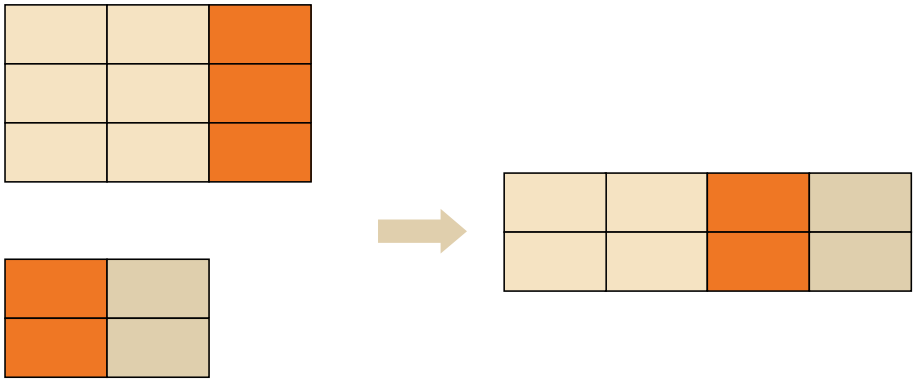


1.2.4. Conexión (*join*)

El segundo tipo de combinación, denominada **conexión**,[2] se puede realizar cuando las dos colecciones de datos tienen contenido diferente, pero tienen algunos elementos comunes (campos y datos).

(2)*Join*, en inglés, de uso muy habitual.

Figura 8. Ejemplo de conexión o *join*



1.3. Restricciones de los datos

El tercer factor de un modelo de datos son las restricciones sobre los datos.

Las **restricciones** son las instrucciones lógicas que pueden imponerse a los datos.

Hay diferentes tipos de restricciones y los diferentes modelos de datos tienen diferentes formas de expresar restricciones.

1) Restricciones de valor. Una restricción de valor es una declaración lógica sobre el valor que pueden tener los datos. Por ejemplo, pongamos que cierto valor de un atributo (edad) no puede ser negativo.

2) Restricciones de unicidad. Esta es una de las limitaciones más importantes. Esta restricción permite identificar de manera única cada elemento de la colección. Por ejemplo, al hacer que el correo electrónico sea único para poder acceder a sitios web. Es posible tener más de un atributo único en una colección.

3) Restricciones de cardinalidad. Requiere que se contabilice el número de valores asociados con cada objeto y que se compruebe si se encuentran entre unos límites superior e inferior.

4) Restricciones de tipo. Para evitar que se pueda poner cualquier valor a un atributo, es posible imponer un tipo de datos. Una restricción de tipo impone un tipo de datos a un atributo. Por ejemplo, podemos imponer que un atributo que contiene el apellido de una persona tenga que ser una cadena alfabética y no pueda ser un número o una fecha. Una restricción de este tipo es un caso particular de restricción de dominio.

5) Restricciones de dominio. El dominio de un atributo es el conjunto de posibles valores permitidos para ese atributo. Por ejemplo, los meses del año están entre 1 y 12, o que la puntuación de un examen en la UOC está entre 0 y 10.

6) Restricciones estructurales. Una restricción estructural impone restricciones a la estructura de los datos en lugar de a los valores de los datos en sí. Por ejemplo, podemos imponer que la estructura de los datos sea matricial y que el número de filas y columnas sea el mismo.

2. Niveles de modelado de datos

Un **modelo de datos** proporciona una representación visual de diversos aspectos funcionales, estructurales o de gestión de una organización.

Además, un modelo de datos actúa como una forma de comunicación entre las diferentes partes interesadas, tanto técnicas como no técnicas.

En ocasiones, un modelo de datos ilustra conceptos que deben comunicarse o acordarse, como si fuera un plano de obra. Estos planos se construyen con varios niveles de detalle distintos, que van desde unos requisitos de diseño básicos hasta especificaciones de diseño detalladas. Estos planos (o modelos) se tienen que construir con diferentes niveles de diseño, denominados **niveles de modelado de datos**.

La tabla 1 presenta los tres niveles de modelado de datos existentes, de menor a mayor nivel de detalle:

Tabla 1. Diferentes niveles de modelado de datos

Niveles	Propósito	Audiencia
Modelado conceptual de datos	Comunicación y definición de términos y reglas.	Gestores y personal interesado en el proyecto (*stakeholders*).
Modelado lógico de datos	Clarificación y detalle de estructuras de datos y reglas.	• Arquitectos de datos • Analistas de negocio
Modelado físico de datos	Implementación técnica en sistemas reales.	• Desarrolladores • Administradores de sistemas y de bases de datos

2.1. Modelado conceptual de datos

El modelado conceptual de datos es el primer paso en una perspectiva de menor a mayor nivel de detalle, en la que el objetivo es capturar desde una vista aérea los requisitos de datos de una organización.

La principal razón para el **modelado conceptual de datos** es capturar el panorama general y comprender el alcance de los requisitos de alto nivel del proyecto de datos en una organización.

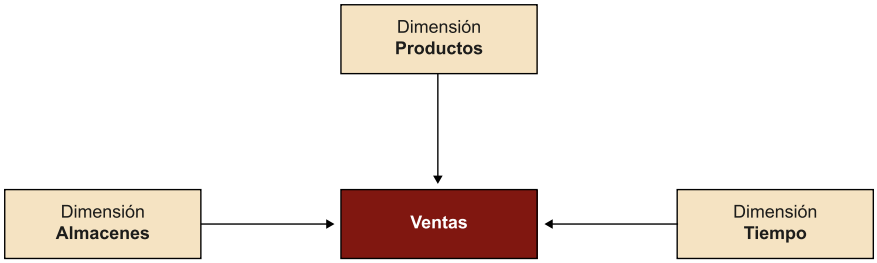
El modelado conceptual de datos intenta responder las siguientes preguntas:

- ¿Qué problema de datos incide en el negocio y necesita una solución?
- ¿Cuáles son los conceptos principales relativos a ese problema?
- ¿Cómo se relacionan estos conceptos entre sí?

El resultado de esto es un modelo conceptual de datos. El modelo generado tiene que ilustrar los conceptos clave necesarios para resolver el problema de datos en particular. Además, debería proporcionar una descripción del esfuerzo requerido por la organización para realizarlo.

Por ejemplo, supongamos un supermercado que quiere medir sus ventas. El supermercado pretende medir las ventas por producto, por almacén y por fecha.

Figura 9. Modelado conceptual de datos



El modelado conceptual de datos ayuda a obtener un análisis preliminar y las definiciones de los términos clave, ayudando a comprender los conceptos más importantes del proyecto y definiendo y documentando los requisitos para cada uno de esos conceptos. También permite explorar las relaciones más importantes entre los conceptos. Por lo tanto, este tipo de modelos suelen denominarse **modelos de dominio**.

2.2. Modelado lógico de datos

El **modelado lógico** se utiliza como paso previo al modelado físico y representa las estructuras detalladas, así como sus relaciones detalladas.

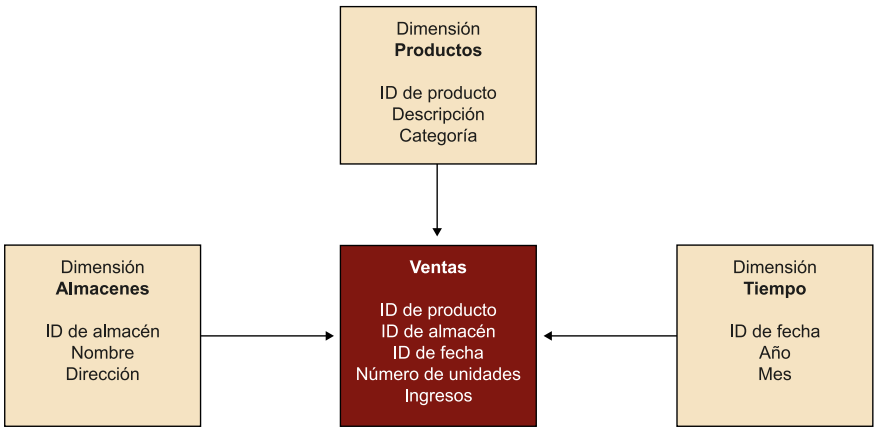
El modelo debería contener todas las entidades esenciales y los atributos, así como las relaciones que tienen entre sí. El modelo no depende de la implementación. Por lo tanto, los modelos lógicos deben ser flexibles y adaptables independientemente del sistema en el que se vaya a implementar.

Los principales **beneficios** de construir un modelo lógico son:

- Facilita la comprensión de los elementos de datos y sus requisitos.
- Ayuda a evitar la duplicidad e inconsistencia de los datos.
- Promueve la reutilización y el intercambio de datos.

En la figura 10 se representa el modelo lógico aplicado al ejemplo del supermecado.

Figura 10. Modelado lógico de datos



En este nivel de modelado, las preguntas que deberían responderse serían:

- ¿Cómo dar respuesta a cuestiones relativas a los datos con la mayor brevedad posible?
- ¿Cómo es la forma óptima de organizar la información?
- ¿Cómo almacenar los datos históricos?
- ¿Cómo hacer que la información sea segura?

El modelado lógico proporciona un mapa general que puede incluir múltiples tecnologías. Aunque el modelo se entiende desde una perspectiva independiente de la tecnología, el modelado lógico implica normalizar y abstraer para obtener las siguientes características:

- Todas las entidades y relaciones entre ellas.
- Todos los atributos de todas las entidades.
- Las restricciones sobre los datos (de valor, unicidad, cardinalidad, tipo y dominio).

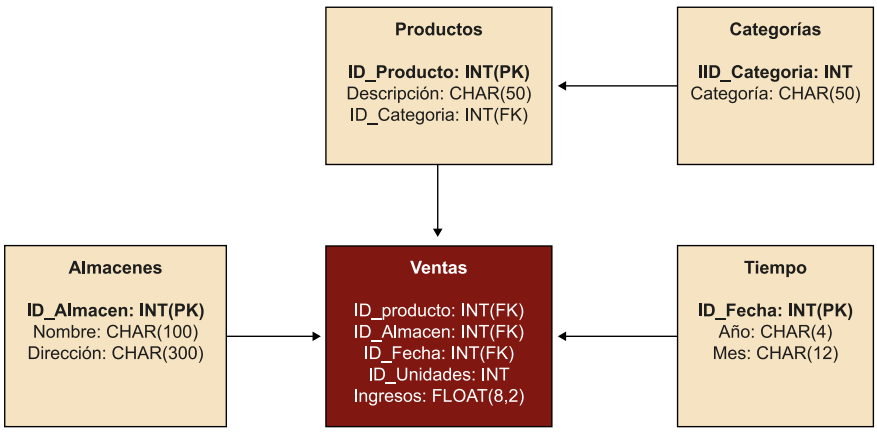
2.3. Modelado físico de datos

El **modelado físico de datos** ilustra cómo se construirán los diferentes elementos de datos, según los requisitos proporcionados por el modelo lógico.

Facilita una visión de cómo es la estructura física del conjunto de datos en la implementación. También se pueden incluir cuestiones de optimización o de mejora de rendimiento. El modelo físico contiene todas las estructuras de datos, incluidos los nombres de los atributos, los tipos de datos, las restricciones de los atributos y las relaciones entre estas estructuras.

En la figura 11 se representa el modelo físico aplicado al ejemplo del supermecado.

Figura 11. Modelado físico de datos



El diseño de un modelo físico de datos se determina por la tecnología con la que se implementará y estará optimizado según los requisitos de esa tecnología. Algunas **tecnologías de implementación** frecuentes serían:

- Bases de datos relacionales
- Bases de datos no relacionales o NoSQL
- Esquemas XML
- Sistemas legados (*legacy*)

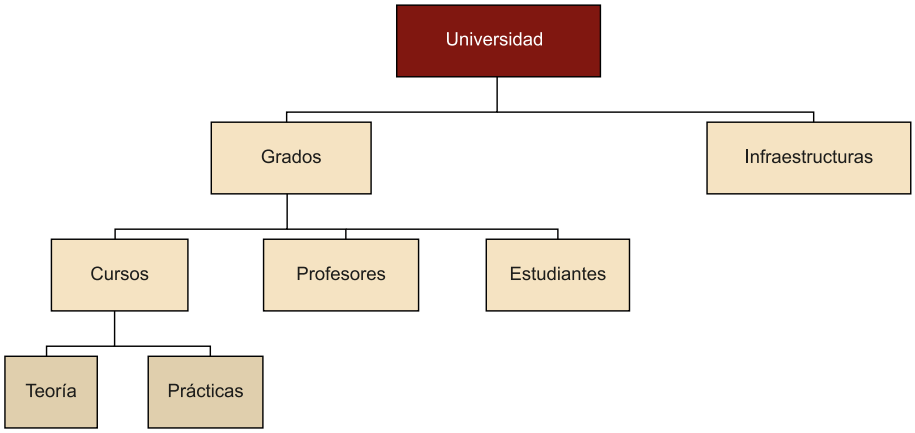
3. Tipos de modelos de datos

Una vez conocido que los modelos de datos son muy importantes en la construcción de una estructura de gestión de datos, estos modelos se pueden clasificar en diferentes tipos, en función de las estructuras que utilizan. En esta sección veremos las más comunes.

3.1. Modelo jerárquico

Un modelo jerárquico utiliza una estructura en forma de árbol para representar los datos, con un solo origen (denominado **raíz**) para cada registro. El registro contiene información sobre un tema en particular y está conectado con otros registros mediante enlaces. Hay un orden concreto de organización de los registros dentro de cada nivel del árbol (figura 12).

Figura 12. Modelo jerárquico de datos



Como se muestra en la figura 12, cada rama de la jerarquía representa una serie de registros relacionados. Una relación en un modelo jerárquico es una relación padre/hijo. Para acceder a los datos dentro de este modelo hay que comenzar en la raíz y descender por el árbol hasta los datos de destino. Es necesario conocer la estructura del modelo para poder acceder a los datos.

Acceder a los datos

Por ejemplo, en el modelo jerárquico que muestra la figura 12, para acceder a los datos de «Estudiantes», hay que conocer todo el modelo.

3.2. Modelo relacional

Los modelos relacionales organizan los datos en tablas denominadas **relaciones**. Las relaciones se componen de filas y columnas. Cada fila tiene un valor único, llamado *tupla*, y cada columna alberga un valor, llamado *atributo*:

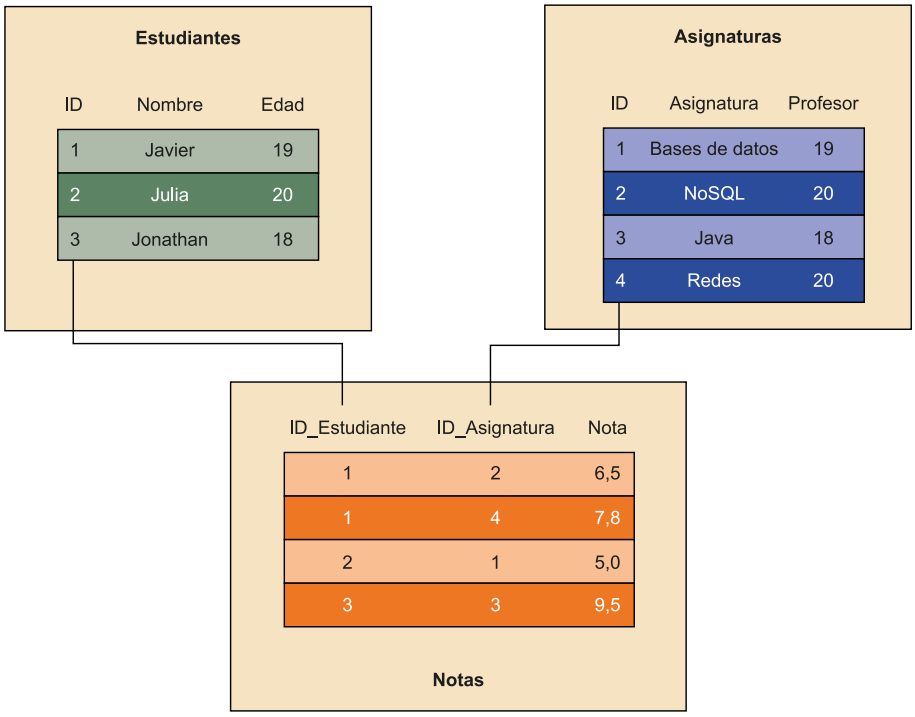
- Una **clave primaria** (*primary key*) es un atributo o una combinación de atributos que tienen la restricción de unicidad y siempre tienen valores.

- Una clave primaria en otra tabla se denomina **clave externa** (*foreign key*).
- Una **tupla** incluye los datos propios de una entidad.
- El **grado** de una relación es el número de atributos en la relación.
- La **cardinalidad** es el número de tuplas de una relación.

Un **modelo relacional** es un método declarativo para especificar tanto datos como consultas.

Esto implica que hay que declarar directamente los datos existentes y, con ello, se posibilita que el software defina las estructuras de datos para gestionarlos y recuperarlos de forma eficiente.

Figura 13. Ejemplo de modelo relacional



El modelo relacional es el más popular y extendido de todos los modelos de datos por ser el más utilizado en el ecosistema de sistemas de gestión de bases de datos. Los **beneficios** de usar un modelo de datos relacional son los siguientes:

- La principal ventaja es su capacidad para describir datos de forma sencilla.
- El proceso de recuperación de registros se simplifica usando los atributos clave.

- Es posible representar diferentes tipos de relación con este modelo.

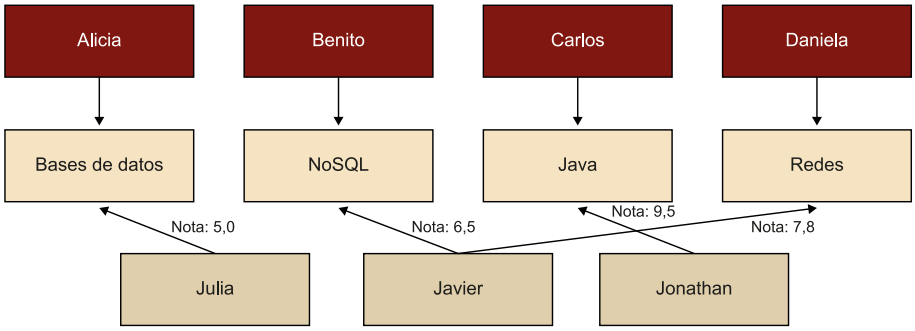
3.3. Modelo en red

Un **modelo en red** está diseñado como un enfoque flexible para representar datos y sus relaciones. En un modelo de datos en red, los datos se representan en términos de **nodos** y de **enlaces**:

- Un **nodo** representa un registro, que es una colección de atributos.
- Un **enlace** representa la relación entre dos nodos.

El modelo de datos en red es parecido al modelo de datos jerárquicos, pero es más sencillo y su implementación más fácil.

Figura 14. Ejemplo de modelo en red



3.4. Modelo orientado a objetos

Un **modelo de datos orientado a objetos** es un modelo de datos que trata los conjuntos de datos como «objetos».

Estos objetos son entidades que incluyen tanto atributos como comportamiento, es decir, combinan datos y procedimientos para trabajar con los datos.

Los **elementos** de un modelo de datos orientado a objetos son:

- Los **objetos** que representan a entidades o situaciones del mundo real.

- Los **atributos** representan ciertas características de los objetos y los **métodos** representan el comportamiento de los objetos.

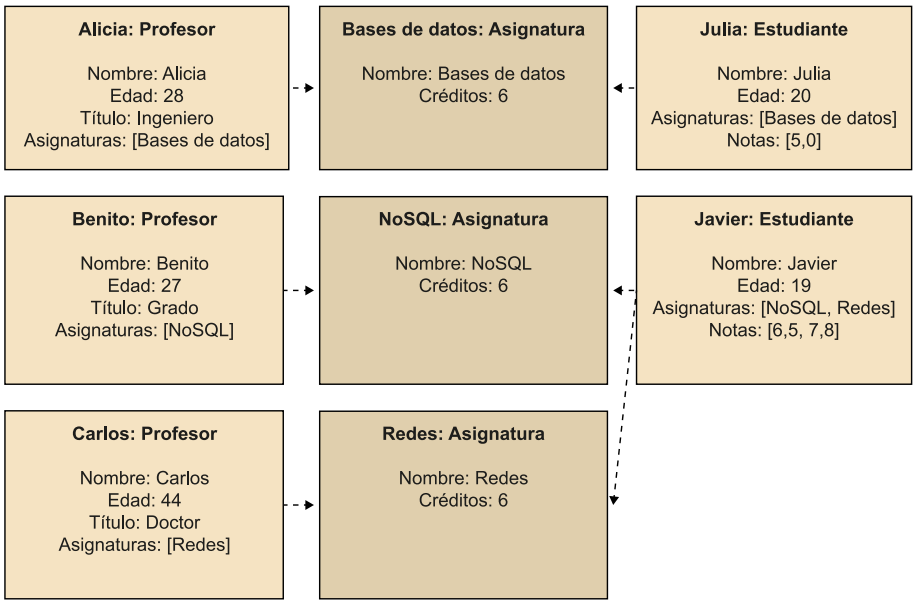
- Las **clases** son generalizaciones de objetos que comparten atributos y métodos. Un objeto se dice que es una **instancia** de una clase.

Con este tipo de modelo se pueden responder preguntas sobre los conjuntos de datos del tipo:

- ¿Cuántos de estos «objetos» se ajustan a un determinado formato?
- ¿Cuántos datos contienen cada uno de ellos?

La principal ventaja de este enfoque es que se adapta perfectamente a trabajar con lenguajes orientados a objetos. Sin embargo, el modelo relacional sigue siendo el más utilizado; así pues existe también un **modelo objeto-relacional**, un modelo híbrido que combina un modelo relacional con algunas funcionalidades superiores del modelo orientado a objetos. Es decir, permite un modelo de datos orientado a objetos virtualizado sobre un modelo relacional.

Figura 15. Ejemplo de modelo de datos orientado a objetos



Bibliografía

Abiteboul, S.; Buneman, P.; Suciu, D. (2000). *Data on the Web: From Relations to Semistructured Data and XML. The Morgan Kaufmann Series in Data Management Systems*.

Batini, C. (1991). *Conceptual Database Design: An Entity-Relationship Approach*. Londres: Pearson Education.

Google. «Comprende cómo funcionan los datos estructurados». Disponible en: https://developers.google.com/search/docs/guides/intro-structured-data

Inmon, W. H.; Linstedt, D. (2014). *Data Architecture: A Primer for the Data Scientist: Big Data, Data Warehouse, and Data Vault*. Massachusetts: Morgan Kaufmann.

Singh Birgi, J.; Khaire, M.; Hira, S. (2016). «Data Model: A Blueprint for Data Warehouse». *International Journal of Scientific and Research Publications* (vol. 6, n.º 1).

Wei, T.; Lee, J.; Kumar Mukhiya, S. (2018). *Hands-On Big Data Modeling*. Packt Publishing.

Yannakoudakis, E. J. (2013). *The Architectural Logic of Database Systems*. Berlín: Springer Verlag.