

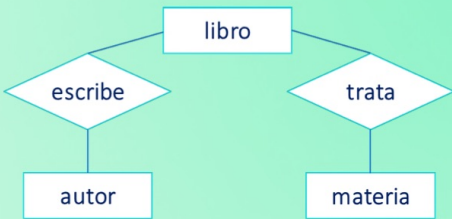
FUND. BASES DE DATOS - TEMA 3

MODELOS DE DATOS.

EL MODELO DE DATOS RELACIONAL

DEFINICIÓN DE MODELO DE DATOS

Un modelo de datos establece un proceso de transformación. Antes de todo, presentamos un ejemplo de este proceso:

a) Mundo real	<ul style="list-style-type: none"> • Delimitación objetivos • Selección de datos • Hipótesis semánticas 	<ul style="list-style-type: none"> • Biblioteca
b) Datos operativos	<ul style="list-style-type: none"> • Atributos • Conexiones • Restricciones 	<ul style="list-style-type: none"> • <u>Libro</u>: título, isbn, editorial.... • <u>Autor</u>: nombre, nacionalidad,.... • <u>Materia</u>: código, descripción....
c) Esquema conceptual		

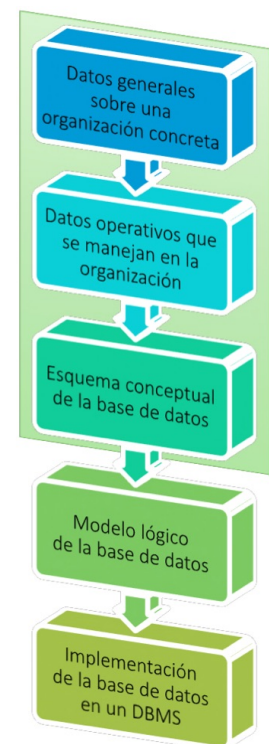


Tabla LIBROS

Modelado lógico

En esta etapa, trasladamos a una estructura implementable el esquema conceptual de la BD al que llegamos en la etapa anterior.

Título	ISBN	Editorial	...
Introducción a las BD	1234-1234	Thomson	...
Cálculo para todos	4321-4321	Delta	...
...

Implementación en un SGBD

```
CREATE TABLE LIBROS (
    titulo char(45) NOT NULL,
    isbn char(10) PRIMARY KEY,
    editorial char(30) REFERENCES ...
);
```

Modelo de datos: mecanismo formal para representar y manipular información de manera general y sistemática. Debe constar de:

- Notación para describir datos
- Notación para describir operaciones
- Notación para describir restricciones de integridad

En relación con la arquitectura ANSI/SPARC, tenemos el modelo de datos externo (NE), el modelo de datos conceptual (NC) y el modelo de datos interno (NI).

A lo largo de la historia, ha habido distintas propuestas: modelo relacional, modelos basados en grafos, modelo E/R, modelos orientados a objetos y modelos lógicos.

Necesidad de modelos de datos: nace de que cada esquema se describe usando un lenguaje de definición de datos que es de muy bajo nivel (muy ligado al SGBD). Hacen falta otros mecanismos de más alto nivel que permitan describir los datos de una forma no ambigua y entendible por los usuarios implicados en cada paso del proceso de implantación.

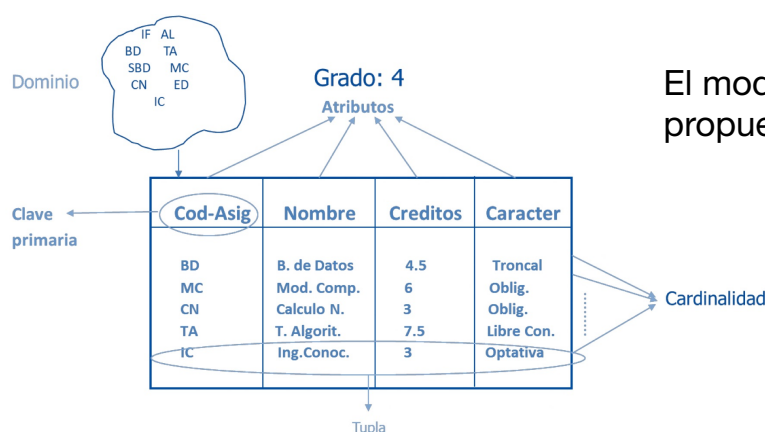
Objetivo: describir modelos que representen los datos y los describan de una forma entendible y manipulable.

Clasificación:

- **Basados en registros** (NE+NC), donde encontramos:
 - + Modelo de datos jerárquico
 - + Modelo de datos en red
 - + Modelo de datos relacional
- **Basados en objetos** (NE+NC)
- **Físicos** (NI)

EL MODELO DE DATOS RELACIONAL: Estructura de datos

El modelo de datos relacional está basado en registros y organiza y representa los datos en forma de tablas o relaciones. Así, una **base de datos relacional** es una colección de tablas cada una de las cuales tiene un nombre único. Veamos un **ejemplo: asignaturas de una titulación**



El modelo de datos relacional fue propuesto por Codd en 1970.

Este modelo abarca tres ámbitos distintos de los datos:

- **Estructuras para almacenarlos:** el usuario percibe la información de la base de datos estructurada en tablas
- **Integridad:** las tablas deben satisfacer ciertas condiciones que preservan la integridad y la coherencia de la información que contienen
- **Consulta y manipulación :** los operadores empleados por el modelo se aplican sobre tablas y devuelven tablas (seminario 4)

La tabla es la estructura lógica de un sistema relacional, por lo que se encuentra en el nivel conceptual. A nivel interno/físico, el sistema es libre de almacenar los datos en el formato más adecuado (archivo secuencial, archivo indexado...).

A continuación, definimos algunos **conceptos** relacionados con este modelo (y que han aparecido en el ejemplo):

Atributo: cualquier elemento de información susceptible de tomar valores - $A_i, i=1,2,\dots$

Dominio: rango finito de valores donde toma sus datos un atributo - $D_i, i=1,2,\dots$

Relación: dados los atributos A_i ($i=1..n$) con dominios D_i no necesariamente distintos, definimos relación asociada a A_1, \dots, A_n , y lo notaremos por $R(A_1, \dots, A_n)$, a cualquier subconjunto del producto cartesiano $D_1 \times \dots \times D_n$ - $R_i, i=1,2,\dots$

Tupla: cada una de las filas de una relación - $t_1, \dots, t_n \in r$

Valor de un atributo A_i en una tupla t_j : $t_j[A_i]$ ó A_{ij}

Cardinalidad de una relación: número de tuplas que contiene (es variable en el tiempo)

Esquema de una relación R : atributos de la relación junto con su dominio - $A_1:D_1, \dots, A_n:D_n$

Grado de una relación: número de atributos de su esquema (invariable en el tiempo)

Instancia de una relación: conjunto de tuplas $\{(x_1, \dots, x_n)\} \subseteq D_1 \times \dots \times D_n$ que la componen en cada momento - $R_1:r_1, \dots$

Esquema de una base de datos relacional: colección de esquemas de relaciones junto con restricciones de integridad

Instancia o estado de una base de datos: colección de instancias de relaciones que verifican las restricciones de integridad

Base de datos relacional: instancia de una base de datos junto con su esquema

En verde la notación que se usa con cada uno de los conceptos.

Vistos estos conceptos, pasamos a estudiar algunas **propiedades** del modelo de datos relacional:

Condición de normalización: todos los valores de los atributos de una relación son atómicos, es decir, no estructurados (no se puede descomponer). Cuando una relación cumple esta condición se dice que está en Primera Forma Normal.

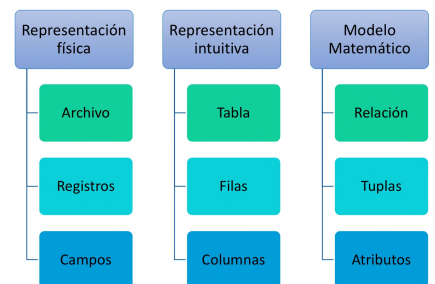
Como consecuencia, no hay valores de tipo conjunto, registro o tablas.

El problema que esto genera es que todas las representaciones tendrán que ser extensivas: no se puede representar directamente información del tipo “**Asignaturas de alumnoX: (FBD, SCD, MC)**”, sino “**Asignatura de alumnoX: FBD**”, “**Asignatura de alumnoX: SCD**”, “**Asignatura de alumnoX: MC**”. Esto desperdicia espacio (aunque no es redundante).

Consecuencias de la definición:

- Por la definición conjuntista de relación, no hay tuplas duplicadas
- No hay orden en las filas ni en los atributos. Así, el acceso es por Nombre de Atributo y Valor. No pido “**columna 4 de las filas 5-7**”, sino “**columna D de las filas con valor a1 en la columna A**”. La ventaja de esto es que conseguimos independencia física.

Relacionando todos los puntos de vista, podemos llegar a:



En algunas ocasiones, el valor de un atributo para una determinada tupla no se conoce. En esos casos, al atributo en cuestión se le asigna un valor nulo (que puede ser, o bien un valor desconocido, o bien un atributo no aplicable). Dicho valor será otro más en cualquiera de los dominios de la BD.

Aunque es cierto que a veces no representan información desconocida como tal, sino que cabe la posibilidad de que una tupla no requiera de un campo en concreto (por ejemplo, una persona genérica no necesita el atributo **Rango_Militar**), a ojos de la BD representan información que debería tener y no tengo.

Cualquier BD está plagada de nulos, y esto supone un gran problema.

EL MODELO DE DATOS RELACIONAL : Restricciones de integridad

Las restricciones o reglas de integridad son condiciones para preservar la corrección semántica de una BD. Las hay de **dos tipos**:

- **Específicas del problema:** dependen del escenario que estemos modelando, y algunos ejemplos son $0 \leq \text{edad} \leq 100$, $\text{color_ojos} = \text{marron, verde, azul}$.

- Propias del papel de los atributos en el esquema:

+ El código de una entidad que participa en relación ha de ser el código de una entidad existente. Por ejemplo, un profesor inexistente no puede impartir una asignatura: $\{\text{imparte.NPR}\} \subseteq \{\text{prof.NPR}\}$.

+ Una clave primaria nunca puede ser un nulo. Por ejemplo, una asignatura siempre ha de tener un código conocido.

A continuación, vemos algunos **conceptos** que necesitamos conocer para poder entender los siguientes puntos de este apartado.

Superclave: cualquier conjunto de atributos que identifica unívocamente a cada tupla de una relación (ej para una entidad alumno: DNI, Fecha_nac)

\subseteq

Clave: superclave minimal (clave \subseteq superclave siempre) (ej para una entidad alumno: DNI)

En una relación puede haber más de un conjunto de atributos que cumplan las condiciones para ser clave. Estos conjuntos de atributos se llaman **claves candidatas (CC)**. De entre las claves candidatas, el diseñador elige una como principal, que se hará llamar **clave primaria (CP)**. Para seleccionar, se aplicanw distintos criterios: tamaño, significado, fusión de tablas... Formalmente, definimos:

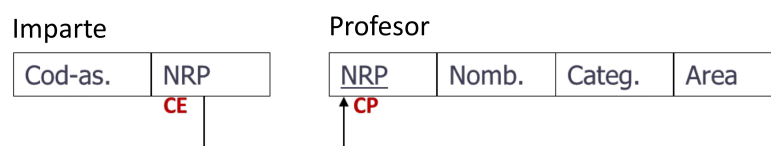
- Sea $R(A_1, A_2, \dots, A_n)$; $CC \subseteq \{A_1, A_2, \dots, A_n\}$ se denomina clave candidata si y solo si:
 - Unicidad: $\forall r$ instancia de R y $\forall t_1, t_2 \in r, t_1 \neq t_2 \Rightarrow t_1[CC] \neq t_2[CC]$
 - Minimalidad: No existe $CC' \subset CC$ que verifique la unicidad.

Es decir, una clave candidata es un atributo o conjunto de atributos que identifica a cada tupla en la relación para el que, además, no existe un subconjunto propio del mismo que también identifique a cada tupla de la relación. Si CC verifica la unicidad y no la minimalidad, entonces sólo es superclave.

Por otro lado, consideramos una relación R que referencia, un subconjunto de atributos de su esquema que denominaremos **clave externa (CE)** y otra relación S que es referenciada, cuya CP coincide con CE . R y S pueden coincidir. Formalmente:

- Si CE es una clave externa de R que referencia a CP en S , entonces:
 - $\forall t \in r, \exists t' \in s$ tal que $t[CE] = s[CP]$, donde r y s son las instancias de R y S en la base de datos.

Gráficamente:



Notar que puede haber más de una CE en una relación, y que la CE puede referenciar a la CP de la propia relación (en el caso $R=S$).

Por último, definimos el **dominio activo**, que será un subconjunto de valores del dominio de atributo de una relación que está presente en la instancia de la relación. Se puede extender la definición a un conjunto de atributos.

Las claves externas establecen relaciones de inclusión entre los dominios activos de la clave externa y la clave referenciada.

Tras introducir estas definiciones, pasamos ahora a ver **conceptos** más **generales**:

Condiciones de integridad: normas que mantienen la corrección semántica de una BD

Integridad genérica: depende del papel que juega un atributo en el diseño de la tabla. Se mantiene a través de metarreglas (generan reglas de integridad aplicadas a una BD concreta). Hay dos tipos:

- **Integridad de entidad:** no se debe permitir que una entidad sea representada en la BD si no se tiene una información completa de los atributos que son CP para la entidad (ningún campo de la CP puede ser nulo pues, en ese caso, dos tuplas con el mismo campo nulo y los otros campos con idénticos valores serían indistinguibles).

- **Integridad referencial:** se cumple si en una BD todos los valores no nulos de una CE referencian los de la clave referenciada en la otra relación. Si una relación incluye una CE conectada a una CP, el valor de la clave debe ser, bien igual a un valor ya existente en el dominio activo de la CP, o bien un nulo (si la semántica lo permite). Esta integridad mantiene las conexiones en las BD relacionales.

Para asegurar todo esto, el SGBD tiene que encargarse de mantener las siguientes restricciones:

- Unicidad de la CP y de las CC: a la hora de insertar/actualizar valores de atributos, debe evitar duplicados en las CP y CC

- Integridad de entidad: a la hora de insertar/actualizar valores de atributos, debe evitar que se asigne nulo a algún atributo de la CP

- Integridad referencial:

- + En *inserción*, debe rechazar a) tuplas cuya CE no concuerde en la relación que referencian para ninguno de los valores del dominio activo de la CP y b) tuplas cuya CE es nula y el diseño no lo permita.

- + En *actualización*, debe a) rechazar modificaciones que sitúen a la tupla en uno de los casos anteriores y b) actualizar en cadena las CE que referencien una CP que se quiere modificar (o, directamente, impedir la modificación si existen referencias).

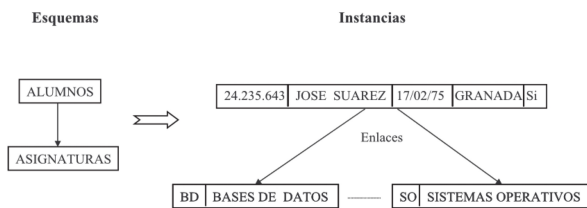
- + En *borrado*, borrado en cadena de las tuplas que referencian una CP que se quiere borrar, o poner valor nulo en las CE si es permitido.

OTROS MODELOS DE DATOS : Modelo jerárquico

Nota: todos los modelos que vemos son estructurados (los hay semi-estructurados también, pero no los estudiamos).

Fue el primero en implementarse físicamente. En el NE, disponía de aplicaciones Cobol (lenguaje de programación “para banqueros”). No era interactivo porque carecía de un lenguaje de consulta.

La **estructura de datos** era básica, en árbol con un registro maestro y registros secundarios. La **BD** era una colección de instancias de árboles.



Esta **estructura** plasma de forma muy directa relaciones N:1 y 1:1.

Sin embargo, para las N:M, habría que duplicar toda la información sobre las entidades involucradas (asig->alumnos).

Surgen **inconvenientes** como los siguientes:

- Almacenar árboles en ficheros es complejo (hay varios tipos de registros y hay punteros que mantener).
- DML difícil tanto de implementar como de usar.
- No se puede insertar un registro secundario mientras no exista el correspondiente raíz con el que enlazarlo (se genera una dependencia existencial).
- Redundancia necesaria en las relaciones N:M (la integridad de los datos es difícil de mantener)

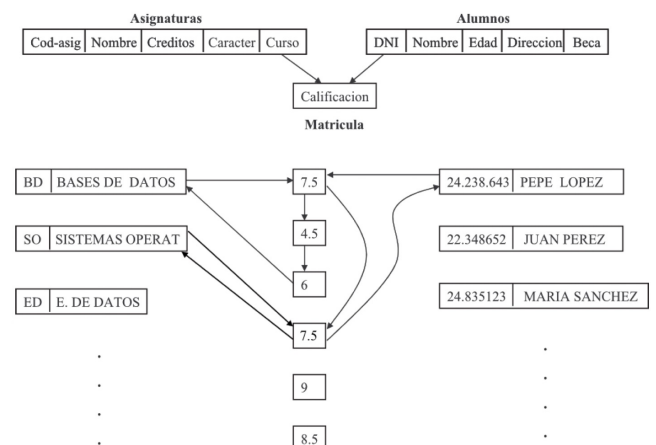
OTROS MODELOS DE DATOS : Modelo en Red

Para solucionar los problemas que se generan con el modelo anterior, se cambia el árbol grafo. Así, la **estructura de datos** de un modelo en red se basa en grafos cuya topología depende de las conexiones existentes entre las entidades. Los nodos son registros, los arcos enlaces entre ellos y las relaciones entre conjuntos de entidades se representan con conectores (registros especiales), siendo cada conector único para cada relación. La **BD** es una colección de instancias de grafos.

Se trata de una **estructura** muy genérica, que permite plasmar todo tipo de relaciones y que implementa directamente relaciones N:M.

Como **ventaja**, vemos que la estructura es más homogénea y que elimina la dependencia existencial que había antes.

Como **inconvenientes**, las operaciones del DDL y del DML siguen siendo complejas de implementar y utilizar debido a la existencia de enlaces entre registro.





OTROS MODELOS DE DATOS : Comparativa

Comparamos ahora las características de los distintos modelos que hemos estudiado:

Con respecto a la representación:

- Relacional
 - Un solo elemento para la representación (esencialidad).
 - Conexiones lógicas.
 - Representación relaciones n:m simétrica.
 - Identidad por valor.
- Basado en grafos
 - Dos elementos para la representación.
 - Conexiones en el modelo físico subyacente.
 - Representación conexiones n:m: imposible en modelos jerárquicos, difícil en modelos en red.
 - Identidad por posición.

Con respecto a la consulta:

- Relacional
 - Obtención de la consulta como resultado global.
 - Lenguajes declarativos (y también procedimentales).
- Basado en grafos
 - Mecanismo de navegación por punteros.
 - Lenguajes procedimentales.

Nota: un *lenguaje declarativo* permite definir muy bien lo que tenemos que buscar para que el sistema aplique un algoritmo que desarrolle el camino hacia lo que trato de encontrar, mientras que un *lenguaje procedimental* está diseñado para que le diga al sistema que operaciones tiene que hacer para buscar en la BD.

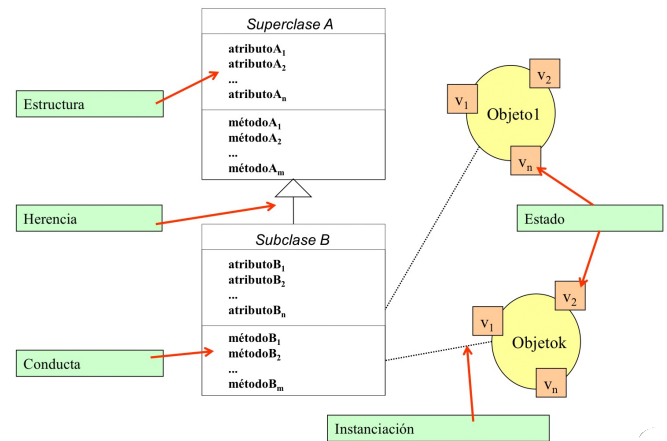
OTROS MODELOS DE DATOS : Orientación a objetos

Surge a partir de las debilidades de los SGBD relacionales, que son:

- Representan pobremente las entidades del mundo real
- Sobrecargan semánticamente la estructura básica (la relación)
- La estructura relacional en sí es muy estricta:
 - + Todas las tuplas han de tener los mismos atributos
 - + Los valores de un atributo pertenecen al mismo dominio
 - + Los atributos han de tener un valor atómico
- SQL, que es puramente relacional, permite un conjunto de operaciones limitado, con lo que no puede modelar el comportamiento de muchos objetos del mundo real
- Object-relational impedance mismatch

La filosofía de este nuevo modelo es abstraer al propio modelo de la realidad en forma de conjunto de objetos que interaccionan entre sí por medio de mensajes. Se introducen nuevos conceptos: estado/comportamiento, propiedades/métodos, encapsulamiento, herencia y polimorfismo.

Cuando en el mundo de la informática surge la orientación a objetos, se ofrece a las BD una herramienta gran capacidad de modelado, pero que supone dificultades en la implementación del SGBD. Más tarde, se idea el modelo objeto-relacional, que cuenta con la solidez de los SGBD relacionales y con la capacidad de modelado de la orientación a objetos.



OTROS MODELOS DE DATOS : Sistemas multidimensionales (cubos de datos)

En una empresa, conviene disponer de dos modelos de BD:

- **BD operacionales** (modelo relacional o de objetos, OLTP), que soporte el funcionamiento diario de la empresa.
- **BD analíticas** (modelo multidimensional, OLAP), que ayude en la toma de decisiones. También se denominan como Data Warehouse (DW).

OLTP: On-line Transaction Processing

Las aplicaciones OLTP dan soporte a las operaciones diarias (estructuradas, repetitivas), y requieren de datos detallados y al día. Las transacciones afectan fundamentalmente a pocos registros, a los cuales se accede principalmente a través de la CP.

Es de vital importancia la consistencia (los datos se almacenan acorde a la empresa) y la fiabilidad (si esta BD se interrumpe, ¡la empresa también!). Además, el rendimiento es esencial, por lo que siempre se busca optimizar la gestión de las transacciones.

OLAP: On-line Analytical Processing

Las aplicaciones OLAP están orientadas al soporte de decisiones y no son tan previsibles como las OLTP. Recaban datos importantes: consolidados (pueden proceder de varias BD operacionales como fuente), resumidos y de tipo histórico (la dimensión tiempo juega un papel fundamental). **Por ejemplo, no nos interesa qué artículos se vendieron entre las 10 y las 13 del día 8 de enero de 2022, sino qué 5 artículos se vendieron más a lo largo de todo ese mes.**

Estos sistemas resuelven consultas complejas, en tiempo real (aunque también predefinidas) y que pueden involucrar millones de registros (lo que hace muy importante el tiempo de respuesta, más que el control de transacciones). Es por esto por lo que necesita estructuras de datos diferentes.

	Orientada a ...	Función	Procesamiento
BD Operacional	DBAs, Programadores de aplicaciones, Operadores ...	Soporte al funcionamiento diario de la organización.	De transacciones.
DW	Ejecutivos.	Soporte a la toma de decisiones.	Analítico.

Datos	Uso	Acceso	Crucial
Actuales, aislados, relacionales...	Repetitivo.	Lectura/escritura. Transacciones simples.	Gestión de transacciones. Consistencia de los datos.
Históricos, resumidos, integrados, multidimensionales...	"Ad hoc".	Lectura. Consultas complejas.	Gestión de consultas. Oportunidad de los datos.

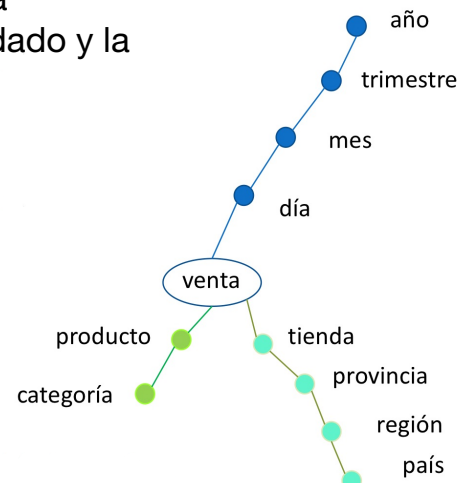
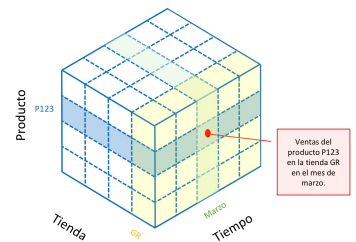
El **modelado multidimensional** es una técnica de modelado que permite organizar los datos como un conjunto de medidas que están descritas por aspectos comunes del negocio. Sirve para (des)agregar datos y reordenarlos para el análisis, y está enfocado para trabajar sobre datos numéricos.

Si se compara con otros modelos más propios de sistemas operacionales (**E/R**, **objeto-relacional...**), puede considerarse más fácil de entender y de usar, además de ser visualmente más atractivo de cara a la toma de decisiones.

Hace uso de cubos (o hipercubos) que representan un contexto caracterizado por n dimensiones.

Podemos jugar con el cubo a través de las operaciones OLAP (roll-up, drill-down, slice and dice, pivot) para consultar y analizar los datos.

Las dos primeras sirven si organizamos jerárquicamente (en 1d o en n dimensiones) los valores que le puedo dar a una dimensión. La tercera para “cortar una rebanada” del dado y la tercera para ver sus diferentes caras.



- FIN DEL TEMA 3 -