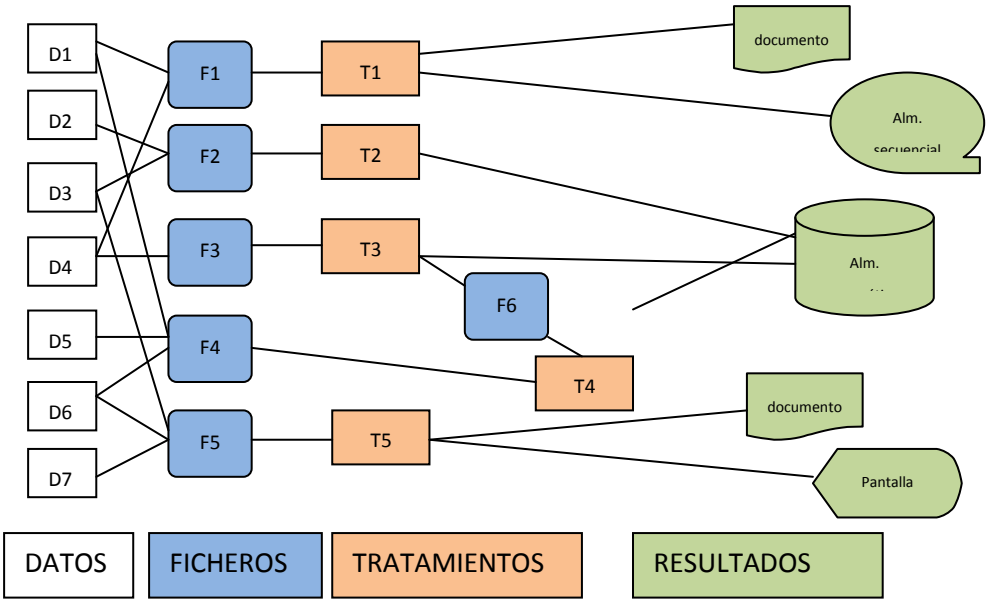
UNIDAD DE TRABAJO 1: ALMACENAMIENTO DE LA INFORMACIÓN

TEMA 2: BASES DE DATOS. SISTEMAS GESTORES DE BASES DE DATOS

2.1. CONCEPTO Y VENTAJAS DE LOS SISTEMAS DE BASES DE DATOS

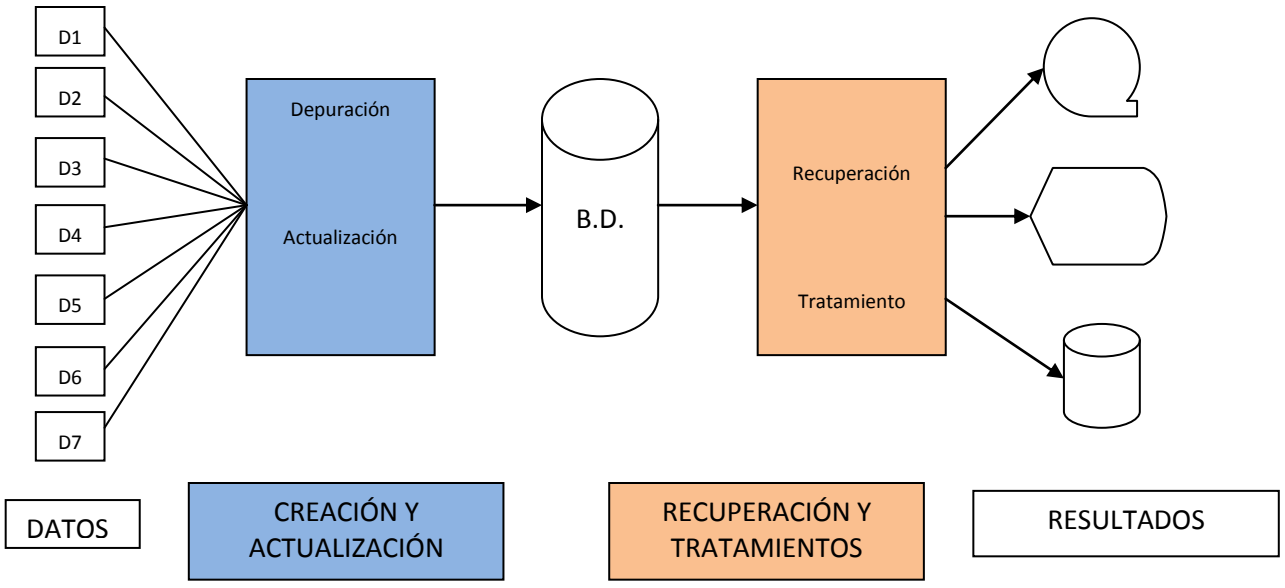
Los sistemas informáticos tradicionales se denominan sistemas **orientados hacia procesos**, debido a que, en ellos, se pone el énfasis en los tratamientos que reciben los datos, los cuales se almacenan en ficheros que son diseñados para una **determinada** aplicación.



Este planteamiento produce, además de una ocupación inútil de memoria secundaria, un aumento de los tiempos de proceso, al repetirse los mismos controles y operaciones en los distintos ficheros. Pero más graves todavía son las inconsistencias que a menudo se presentan en estos sistemas, debido a que la actualización de los mismos datos, cuando éstos se encuentran en más de un fichero, no se suele realizar de forma simultánea en todos ellos.

Por otra parte, la dependencia de los datos respecto al soporte físico y a los programas da lugar a una falta de flexibilidad y de adaptabilidad frente a los cambios que repercute muy negativamente en el rendimiento conjunto del sistema informático.

Con el fin de resolver estos problemas y de lograr una gestión más racional del conjunto de datos, surge un nuevo enfoque que se apoya sobre una "base de datos" en la cual los datos son recogidos y almacenados, al menos lógicamente, una sola vez, con independencia de los tratamientos.



CONCEPTO DE BASE DE DATOS:

"Una base de datos es una colección o depósito de datos integrados, con redundancia controlada y con una estructura que refleja las interrelaciones y restricciones existentes en el mundo real. Los datos, que han de ser compartidos por diferentes usuarios y aplicaciones, deben mantenerse independientes de éstas, y su definición y descripción únicas para cada tipo de datos, han de estar almacenadas junto con los mismos. Los procedimientos de actualización y recuperación, comunes y bien determinados, habrán de ser capaces de conservar la seguridad (integridad, confidencialidad y disponibilidad) del conjunto de datos".

Ventajas:

- **Independencia de los datos respecto a los tratamientos y viceversa:**
  Quiere decir que aunque varíe el tratamiento (proceso) que vamos a realizar sobre los datos no es necesario modificar el diseño lógico o físico de la base de datos. De igual forma el hecho de que aparezcan nuevos datos o desaparezcan otros, no implica tener que modificar los programas que actúan sobre la base de datos.
  La independencia entre datos y procesos es muy importante a la hora de plantearse adaptaciones del sistema a los cambios que sufren los sistemas de información.
- **Coherencia de los resultados**:
  Con lo que se elimina el inconveniente de las divergencias en los resultados debidas a actualizaciones no simultáneas en todos los ficheros.
- **Mejor disponibilidad de los datos:**
  Con una mayor "transparencia" respecto a la información existente para todo el conjunto de usuarios.
- **Mayor valor informativo:**
  Todos los elementos están interrelacionados, por lo tanto la información que se nos ofrece siempre es completa y veraz.
- **Documentación de la información mejor y más normalizada.**
- **Mayor eficiencia en la recogida, validación y entrada de los datos al sistema**

2.2. CARACTERÍSTICAS DE LAS BASES DE DATOS

1. **Posibilidad de ser utilizada por varios usuarios:**

   Las bases de datos deben permitir el acceso a los datos a uno o varios usuarios asegurando un tiempo de respuesta adecuado en la comunicación hombre-máquina.

2. **Tener mínima redundancia:**

   La existencia de redundancias tienen como efecto la inconsistencia en la información almacenada, esto quiere decir que en un momento dado si un dato está almacenado en diferentes sitios puede presentar valores distintos. Las bases de datos no eliminan por completo la redundancia ya que deben representar las interrelaciones que existen entre las entidades que forman parte del sistema, pero en las bases de datos se establecen procedimientos que garantizan la consistencia de los datos.

3. **Simplicidad:**

   Las bases de datos deben estar basadas en representaciones lógicas simples que permitan la verificación de la representación del problema y la modificación de los requerimientos sin una dificultad excesiva.

4. **Integridad.**

   La integridad se refiere a la veracidad de los datos almacenados y las relaciones entre ellos, respecto al dominio del problema, por tanto se deben establecer procedimientos para verificar que los valores de los datos se ajustan a los requerimientos y restricciones extraídas del análisis del problema.

5. **Seguridad y privacidad de los datos:**

   La seguridad se refiere a la capacidad de la base de datos para proteger la información contra su pérdida total o parcial por fallos del sistema.
   La privacidad se refiere a la posibilidad de reservar la información a personas no autorizadas.

6. **Mayor eficiencia en la recogida, validación y entrada de los datos al sistema:**

   Al no existir redundancias, los datos se recogen y se validan una sola vez, aumentando el rendimiento de todo proceso de almacenamiento.

7. **Reducción del espacio de almacenamiento:**

   La inexistencia de redundancias implica una menor ocupación en los sistemas de almacenamiento secundario.

2.3. MODELOS DE DATOS

La parte esencial de la estructura de base de datos es el *modelo de datos*: una colección de herramientas conceptuales para describir los datos, las relaciones de datos, la semántica de los datos y las ligaduras de consistencia. Los diferentes modelos de datos que se han propuesto se clasifican en tres grupos diferentes: modelo conceptual (modelo Entidad/Relación), modelo orientado a objetos y modelo tradicional (modelo lógicos basado en registros).

2.3.1 MODELOS CONCEPTUALES

Se usan para describir datos en el nivel global. Representamos los datos de forma muy parecida a como nosotros los captamos en el mundo real. Este tipo de modelos tienen una capacidad de estructuración bastante flexible y permiten especificar restricciones de datos explícitamente. Existen diferentes modelos de este tipo, pero el más utilizado por su sencillez y eficiencia es el modelo Entidad-Relación.

El modelo entidad-relación ha conseguido gran aceptación en el diseño de bases de datos y se usa ampliamente en la práctica.

Modelo Entidad-Relación

Está basado en una percepción del mundo real que consta de una colección de objetos básicos, llamados entidades, y de relaciones entre estos objetos. Una entidad es una cosa u objeto en el mundo real distinguible de otros objetos. Por ejemplo, una persona es una entidad. Las entidades se describen en una base de datos mediante un conjunto de atributos. Por ejemplo, el nombre, o el dni describen a una persona en particular. Por ejemplo, una relación "tiene" asocia a un cliente con una determinada cuenta bancaria (que será otra entidad).

2.3.2. MODELOS LÓGICOS BASADOS EN OBJETOS

Los modelos lógicos basados en objetos se usan para describir datos en los niveles lógico y de vistas. Se caracterizan por el hecho de que proporcionan capacidades estructurales muy flexibles.

El modelo orientado a objetos incluye tanto código ejecutable (métodos) como datos (propiedades).

Como el modelo E-R el modelo orientado a objetos está basado en una colección de objetos. Tendremos, como veremos más adelante, instancias de ese objeto. Un objeto también contiene fragmentos de código que operan dentro de ese objeto, son los *métodos* del objeto.

Los objetos que tienen las mismas características y los mismos métodos se agrupan en *clases.*

La única manera que un objeto tiene para acceder a los datos de otro objeto es mediante la invocación de un método de ese otro objeto. Esta acción se llama *paso de mensaje* al otro objeto.

2.3.3. MODELOS LÓGICOS TRADICIONALES (Modelos lógicos basados en registros)

Se usan para describir datos en el nivel global, pero de un modo más cercano a la máquina. En contraste con los modelos basados en objetos, se usan tanto para especificar la estructura lógica completa de la base de datos como para proporcionar una descripción de alto nivel de la implementación (nivel físico).

Los modelos basados en registros se llaman así debido a que la base de datos se estructura en registros de formato fijo de diferentes tipos. En cada registro se define un número fijo de campos o atributos y cada campo tiene normalmente una longitud fija. El uso de registros de longitud fija simplifica la implementación en el nivel físico de la base de datos.

Los tres modelos basados en registros más conocidos son:

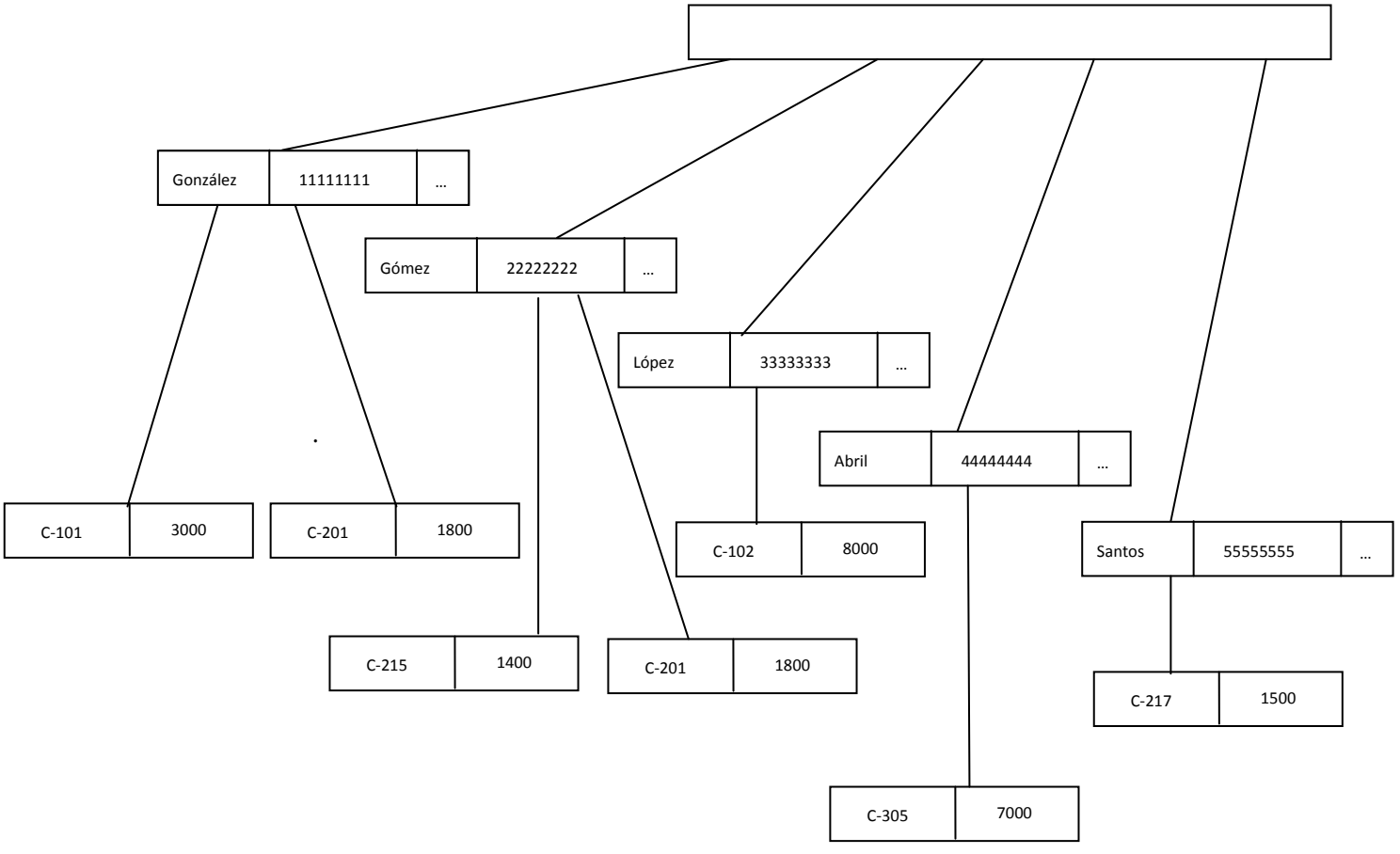
- El modelo jerárquico
- El modelo en red
- El modelo relacional

Tanto el modelo jerárquico como el modelo en red sólo se usan en las bases de datos más antiguas.

MODELO JERÁRQUICO

El modelo jerárquico a pesar de presentar importantes problemas tuvo una gran expansión por estar adoptado por IBM.

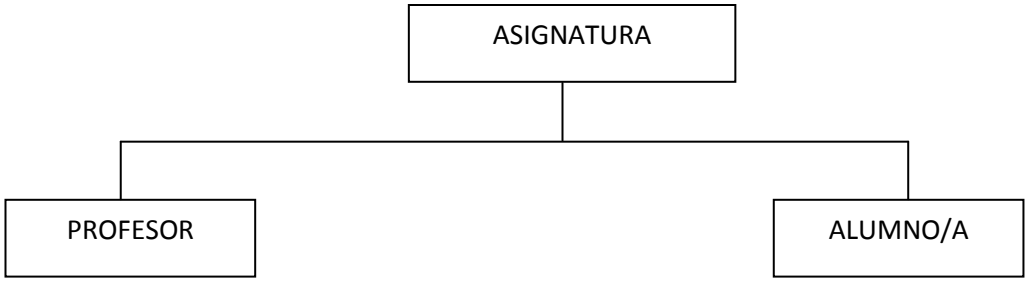
El esquema de un modelo jerárquico es una estructura en árbol que debe cumplir las siguientes características:



- El árbol se organiza en niveles, siendo el nodo más alto de la jerarquía el nivel cero que está formado por el nodo raíz.
- Un nodo padre puede tener cualquier número de hijos, pero a cada hijo solo puede corresponder un nodo padre, en caso de que corresponda a 2 padres, tendrá que repetirse dicho registro, provocando duplicidad de datos y, por lo tanto, más riesgo de incongruencias.
- Todo nodo salvo el nodo raíz debe tener obligatoriamente un nodo padre.
- Los hijos de un padre común están ordenados, de forma que el árbol se recorre de arriba abajo y de izquierda a derecha.

Tenemos que distinguir entre **esquema y ocurrencia.**

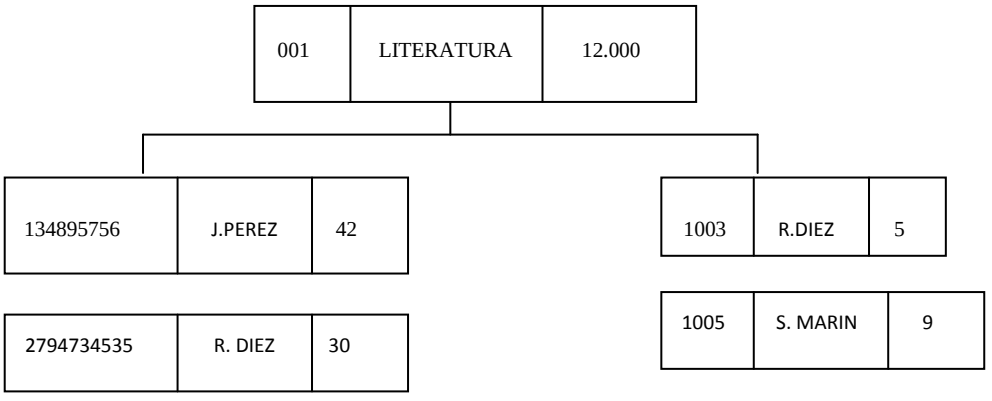
El **esquema** que es la descripción de lo que ocurre en nuestro sistema, representado por un árbol donde los nodos son las entidades y las líneas de unión representan las interrelaciones jerárquicas entre las entidades.



Las entidades representadas están formadas por los siguientes atributos:

ASIGNATURA = @NUM_ASIG + NOMBRE_ASIG
PROFESOR = @DNI + NOMBRE_PROF + EDAD
ALUMNO = @NUN_MATRI + NOMBRE_ALUM + NOTA

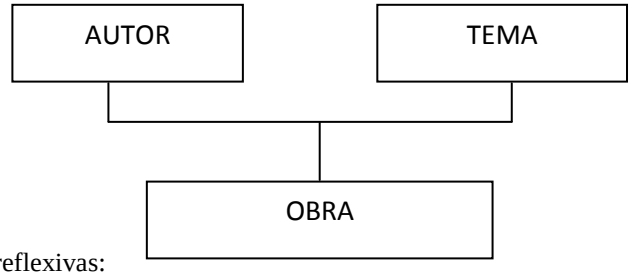
Y una **ocurrencia** del esquema también es un árbol pero los nodos son ocurrencias de las entidades y las líneas representan las interrelaciones jerárquicas entre esas ocurrencias, por tanto una base de datos jerárquica está formada por una colección de árboles disjuntos:



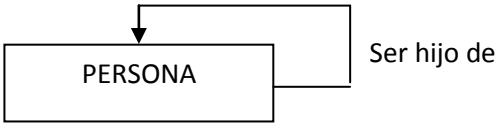
Problemas del modelo jerárquico

Los problemas del modelo jerárquico provienen de la imposibilidad de representar situaciones que aparecen en el mundo real como las siguientes:

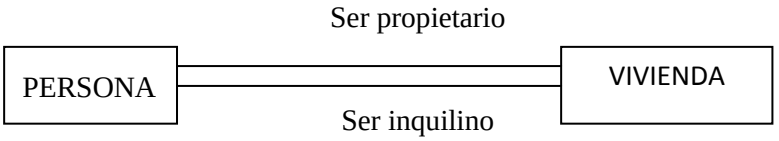
1. Hijos con más de un padre



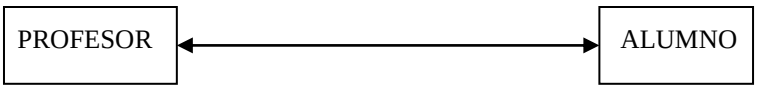
2. Interrelaciones reflexivas:



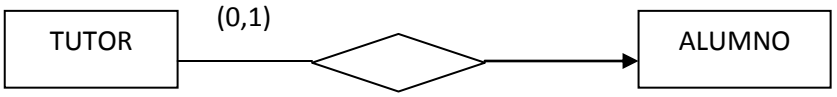
3. Más de una interrelación entre dos entidades:



4. Interrelaciones N:M



5. Hijos sin padre:



Ventajas de una base de datos jerárquica:

La principal ventaja es que las búsquedas son muy rápidas.

En la práctica, los motores de bases de datos más comunes no son jerárquicos, son relacionales, pero se ha aprovechado esta ventaja para soporte de ciertos servicios en entorno Linux como por ejemplo FTP que tiene la opción de validar usuarios sobre una base de datos de tipo LDAP.

LDAP (*Lightweight Directory Access Protocol, Protocolo ligero de acceso a directorios*)**.**

Otro ejemplo típico es el Active Directory de Windows, que es el término que usa Microsoft para referirse a su implementación de servicio de directorio en una red distribuida de computadores. Utiliza distintos protocolos (principalmente LDAP, DNS, DHCP, Kerberos...). Su estructura jerárquica permite mantener una serie de objetos relacionados con componentes de una red, como usuarios, grupos de usuarios, permisos y asignación de recursos y políticas de acceso.

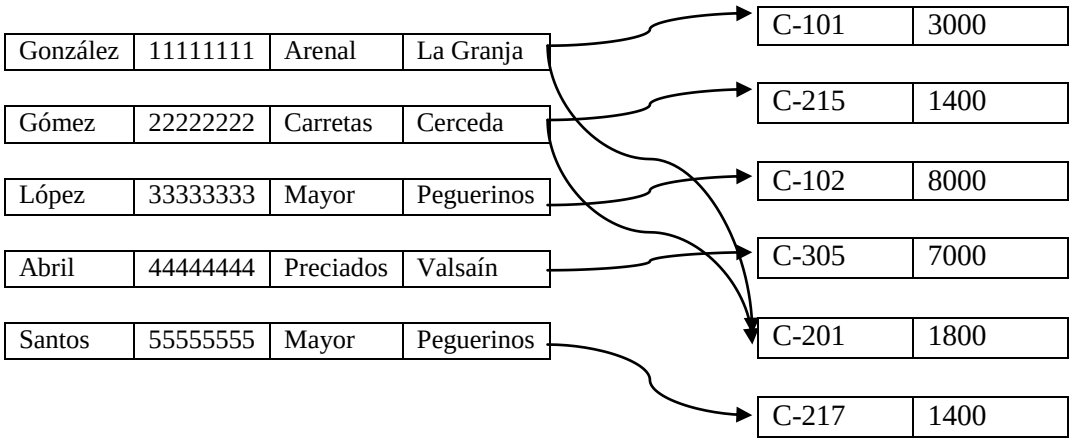
Los servidores de aplicaciones (por ejemplo, Oracle Weblogic Server) tienen su propia implementación de acceso a metadatos organizados de una forma jerárquica, se denomina JMI (Java Metadata Interface).

MODELO EN RED

Éste es un modelo ligeramente distinto del jerárquico; su diferencia fundamental es la modificación del concepto de nodo: se permite que un mismo nodo tenga varios padres (posibilidad no permitida en el modelo jerárquico).

Fue una gran mejora con respecto al modelo jerárquico, ya que ofrecía una solución eficiente al problema de redundancia de datos; pero, aun así, la dificultad que significa administrar la información en una base de datos de red ha significado que sea un modelo desestimado para su uso comercial.

Los datos se representan mediante colecciones de registros y las relaciones entre los datos se representan mediante enlaces, que se pueden ver como punteros. Los registros en la base de datos se organizan como colecciones de grafos dirigidos.



MODELO RELACIONAL

El modelo relacional fue introducido por Codd a finales de los años 60 y se basa en la teoría de las relaciones, estructurando los datos en forma de relaciones (tablas). El objetivo fundamental del modelo es mantener independientes la estructura lógica y el modo de almacenamiento físico.

El elemento básico del modelo relacional es la relación que se puede representar con una tabla que lleva asociado un nombre

Clientes

nombreCliente	Dni	calleCliente	ciudadCliente	numCuenta
González	**11111111**	**Arenal**	**La Granja**	**C-101**
Gómez	**22222222**	**Carretas**	**Cerceda**	**C-215**
López	**33333333**	**Mayor**	**Peguerinos**	**C-102**
Abril	**44444444**	**Preciados**	**Valsaín**	**C-305**
Santos	**55555555**	**Mayor**	**Peguerinos**	**C-217**

Cuentas

numeroCta	Saldo
C-101	**3000**
C-215	**1400**
C-102	**8000**
C-305	**7000**
C-201	**1800**
C-217	**1500**
C-222	**1400**

Atributos: las columnas representan las propiedades de la relación, tienen asignado un nombre
Tuplas: son las ocurrencias de la relación
Cardinalidad: es el número de filas en la tabla.
Grado: es el número de columnas en la tabla.
Dominio: se establece para cada atributo y es el conjunto de valores que puede tomar cada atributo.

Aunque una relación puede representarse mediante una tabla debe cumplir una serie de requisitos como son:

- No puede haber filas duplicadas, todas las tupías deben ser diferentes.
- El orden de las filas es irrelevante.
- El contenido de cada celda (cruce entre líneas y columnas) solamente puede tener un valor, es decir no admite atributos multivaluados.

ALUMNOS

NOMBRE	EDAD	SEXO
Jesús	20	V
María	22	M
Ana	19	M
Rosa	24	M
Pedro	30	V

El esquema de la relación: ALUMNOS **(NOMBRE, EDAD, SEXO)** define la estructura de la tabla y el contenido o extensión de la tabla que estará dado por las tuplas, que varían con el tiempo.

Claves de las relaciones:

- **Clave candidata:** es un atributo o conjunto de atributos que definen de forma unívoca y mínima cada una de las tupías.

- **Clave primaria:** se denomina de esta forma a una clave candidata elegida por el diseñador para identificar las tupías.

- **Claves alternativas:** las claves candidatas que no han sido elegidas como claves primarias.

- **Claves ajenas:** es un atributo o un conjunto de atributos cuyos valores (y dominios) deben coincidir con los valores de la clave primaria de otra relación.

- **Regla de la integridad de entidad:** ningún atributo que forme parte de la clave primaria de una relación puede tomar un valor nulo, es decir un valor desconocido.

- **Regla de la integridad referencial:** los valores que puede tomar una clave ajena en una relación, o coinciden con los valores de la clave primaria que referencian o son nulos.

2.4. EL SISTEMA DE GESTIÓN DE LA BASE DE DATOS

Se denomina Sistema de Gestión de la Base de Datos a un conjunto de programas, procedimientos, lenguajes, etc, que suministra, tanto a los usuarios no informáticos como a los analistas, programadores, o al administrador, los medios necesarios para describir, recuperar y manipular los datos almacenados en la base, manteniendo su seguridad.

Hay que tener en cuenta que en una base de datos existe una gran variedad de usuarios con necesidades diversas que son susceptibles de trabajar simultáneamente con subconjuntos de esta colección de datos y cuyas necesidades varían a lo largo del tiempo. Por tanto, se pone de manifiesto que es imprescindible dotar al sistema de la adecuada flexibilidad para que pueda atender las exigencias de todos los usuarios y sea capaz de responder a los cambios a un coste no excesivo.

El objetivo principal de un SGBD es proporcionar a los usuarios una visión *abstracta* de los datos, es decir, el sistema esconderá ciertos detalles de cómo se almacenan y mantienen los datos.

2.5. NIVELES DE ABSTRACCIÓN DE DATOS EN UN SGBD

Para que un SGBD sea útil, debe recuperar los datos eficientemente. Como muchos usuarios de SGBD no están familiarizados con la informática a un nivel demasiado profundo, los desarrolladores esconden la complejidad a través de **niveles de abstracción** para simplificar la interacción de los usuarios con el sistema:

- **NIVEL FÍSICO:** es el nivel más bajo de abstracción, describe cómo se almacenan realmente los datos. En el nivel físico se describen en detalle las estructuras de datos complejas de bajo nivel.

  Aunque el contenido del esquema interno es muy dependiente de cada SGBD, podemos distinguir tres clases de aspectos que deben especificarse en él:

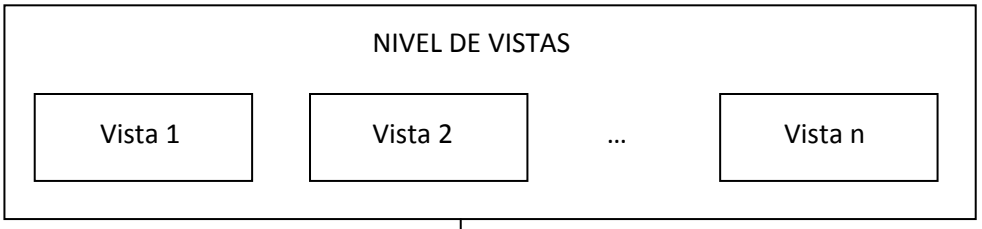
  En este apartado se consideran:
  - La estrategia de almacenamiento: asignación de espacio para los datos y sus interrelaciones.
  - Caminos de acceso: claves primarias o secundarias que se utilicen, índices, claves de ordenación, etc.
  - Además de los aspectos citados habría que incluir otros varios como pueden ser: técnicas de compresión de datos, de criptografiado de los mismos, la traslación o correspondencia del esquema interno al esquema conceptual, técnicas de ajuste y optimización (tuning: configuración de la memoria compartida, ajustar los ficheros temporales…).

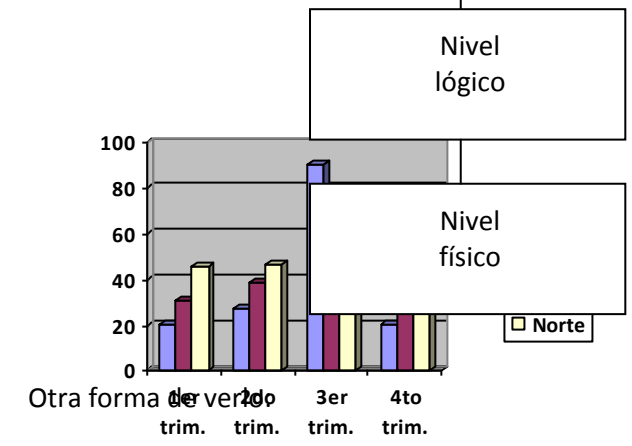
- **NIVEL LÓGICO:** es el siguiente nivel de abstracción, describe qué datos se almacenan en la base de datos y qué relaciones existen entre esos datos. Aunque la implementación de estructuras simples del nivel lógico puede involucrar estructuras complejas del nivel físico, los usuarios del nivel lógico no necesitan preocuparse de esta complejidad.

  Es la visión global de todos los datos, independientemente de cómo se almacenan físicamente y de las distintas visiones de los usuarios y de sus interrelaciones así como de las restricciones de integridad y confidencialidad.

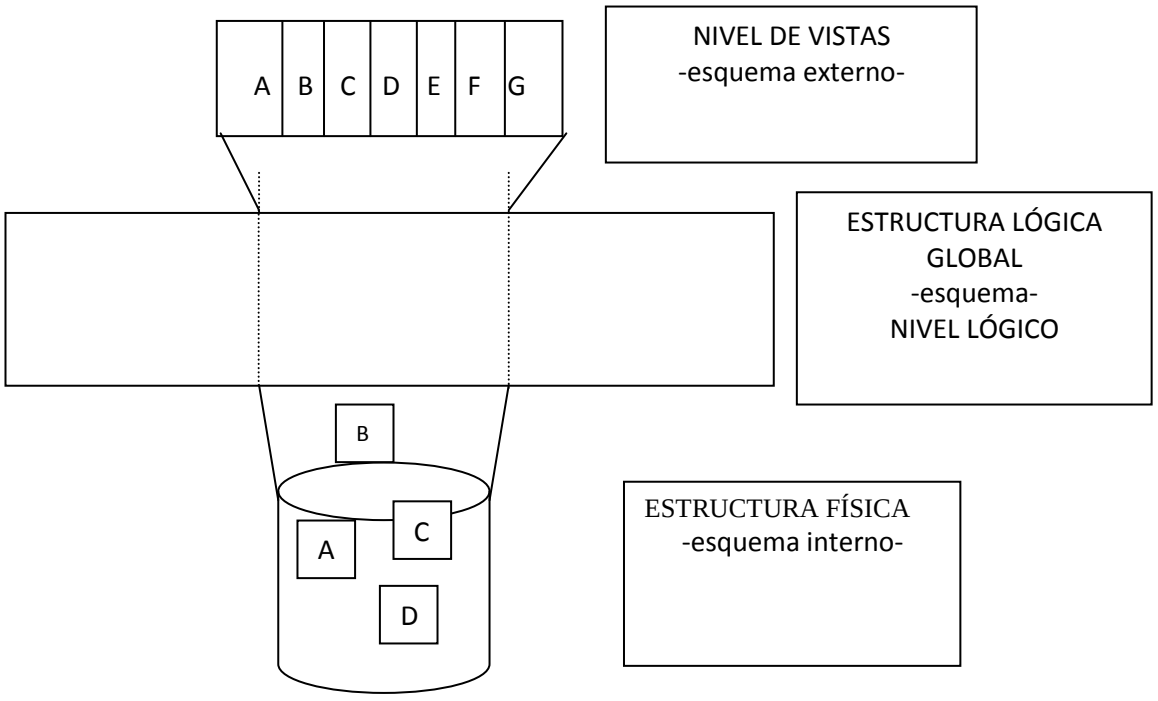
- **NIVEL DE VISTAS:** es el nivel más alto de abstracción. Los usuarios no necesitan acceder a toda la información de todo el SGBD, para que su interacción con el sistema se simplifique se define la abstracción del nivel de vistas. Un SGBD puede proporcionar muchas vistas para la misma base de datos.

  El esquema externo es la visión que tiene el usuario de la base de datos, por tanto en este esquema tendremos reflejados los datos e interrelaciones que necesite dicho usuario, así como las restricciones de utilización de la información de ese usuario, esto es si tiene derecho a insertar, modificar o borrar los datos que él observa. Hay tantos esquemas externos como usuarios.





Otra forma de verlo:



2.6. INDEPENDENCIA DE DATOS

La capacidad para modificar una definición de esquema en un nivel sin que afecte a una definición de esquema en el siguiente nivel más alto se llama *independencia de datos.* Hay dos niveles de independencia de datos:

1. **Independencia física de datos.** Es la capacidad para modificar el esquema físico sin provocar que los programas de aplicación tengan que reescribirse. Las modificaciones en el nivel físico son ocasionalmente necesarias para mejorar el funcionamiento.
2. **Independencia lógica de datos.** Es la capacidad para modificar el esquema lógico sin causar que los programas de aplicación tengan que reescribirse. Las modificaciones en el nivel lógico son necesarias siempre que la estructura lógica de la base de datos se altere (por ejemplo, cuando en un sistema de gestión de educación se añaden nuevas titulaciones).

2.7. LENGUAJES DE BASES DE DATOS

Un sistema de bases de datos proporciona dos tipos de lenguajes diferentes: uno para especificar el esquema de base de datos y el otro para expresar las consultas y actualizaciones de la base de datos.

2.7.1. LENGUAJE DE DEFINICICIÓN DE DATOS

Un esquema de base de datos se especifica mediante un conjunto de definiciones expresadas mediante un lenguaje especial llamado *lenguaje de definición de datos (LDD o DDL)*. El resultado de la compilación de las instrucciones del DDL es un conjunto de tablas que se almacenan en un archivo especial llamado *diccionario de datos o directorio de datos (repositorio).*

El **repositorio o diccionario de datos o directorio de datos** es un archivo que contiene *metadatos*; es decir, datos acerca de los datos. Este archivo se consulta antes de leer o modificar los datos reales del sistema de base de datos.

2.7.2. LENGUAJE DE MANIPULACIÓN DE DATOS

Los niveles de abstracción que se han visto en el apartado 2 se aplican no sólo a la definición o estructuración de los datos, sino también a la manipulación de datos. Por *manipulación de datos* se quiere decir:

- Recuperación de información almacenada en la base de datos.
- La inserción de información nueva en la base de datos.
- El borrado de información de la base de datos.
- La modificación de información almacenada en la base de datos.

Mientras que en el nivel físico se deben definir algoritmos que permitan un acceso eficiente a los datos, en el nivel lógico (niveles más altos de la abstracción) se enfatiza la facilidad de uso. El objetivo es proporcionar una interacción humana eficiente con el sistema.

El lenguaje de *manipulación de datos (LMD o DML)* es un lenguaje que permite a los usuarios acceder o manipular los datos organizados mediante el modelo de datos apropiado. Hay dos tipos:

- **LMD procedimentales:** requieren que el usuario especifique qué datos se necesitan y cómo obtener esos datos.
- **LMD no procedimentales:** requieren que el usuario especifique qué datos se necesitan sin especificar cómo obtener esos datos.

Los LMD no procedimentales son más fáciles de aprender y usar que los procedimentales. Sin embargo, estos lenguajes pueden quedarse "cortos" para realizar consultas complejas.

Una consulta es una instrucción de solicitud para recuperar información. La parte de un LMD que implica recuperación de información se llama *lenguaje de consultas.* Aunque técnicamente sea incorrecto, en la práctica se usa como sinónimos *lenguaje de consultas y lenguaje de manipulación de datos.*

2.8. FUNCIONES DE UN SGBD

Las funciones esenciales de un SGBD son **la descripción, manipulación y utilización**.

- **Función de descripción o definición:** esta función debe permitir al administrador de la base especificar los elementos de datos que la integran, su estructura y las relaciones que existen entre ellos, las reglas de integridad, los controles de acceso antes de autorizar el acceso a la base, etc. así como también las características de tipo físico y las vistas lógicas de los usuarios. Esta función se realiza mediante el lenguaje de definición de datos (LDD o DDL), que permite definir las tres estructuras de datos: externa, lógica e interna.
  Por ejemplo, para crear una tabla en una base de datos relacional, podría utilizarse la sentencia siguiente:

```
CREATE TABLE Documento
(Cod_Doc CHAR(4),
Titulo CHAR(25) NOT NULL,
Idioma CHAR(30),
Año INTEGER(4),
PRIMARY KEY Cod_Doc);
```

- **Función de manipulación de datos:** permite a los usuarios añadir, modificar, borrar o consultar los datos que forman la base de datos, siempre de acuerdo a las especificaciones y las normas de seguridad dictadas por el administrador. Esta función se lleva a cabo mediante el lenguaje de manipulación de datos (LMD) que facilita los instrumentos necesarios para la realización de estas tareas. Por ejemplo, para seleccionar en la tabla creada anteriormente el título y el idioma de los documentos creados en el año 2003, la sentencia sería:

```
SELECT Titulo, Idioma
FROM Documento
WHERE Año=2003;
```

- **Función de utilización:** reúne todas las interfaces que necesitan los diferentes usuarios para comunicarse con la base y proporciona un conjunto de procedimientos para el administrador, entre los que se encuentra el Lenguaje de Control de Datos (LCD). En la mayoría de los SGBD existen funciones de servicio, como cambiar la capacidad de los ficheros, obtener estadísticas de utilización, cargar archivos, etc.; principalmente las relacionadas con la seguridad física –copias de seguridad, rearranque en caso de caída del sistema, etc.—y de protección frente a accesos no autorizados, las cuales se encuentran comprendidas en la función de utilización. Para conceder el privilegio de consulta en una tabla al usuario Juan se haría así:

```
GRAN SELECT
ON Documento
TO Elena;
```

2.9. EL ADMINISTRADOR DE LA BASE DE DATOS (DBA)

El administrador de la base de datos (DBA) es la persona o grupo de personas responsables del diseño, control y administración de la base de datos.

Las principales funciones asignadas al DBA son:

- Determinar los datos que se deben almacenar en la base de datos después de haber analizado los requisitos de los distintos usuarios.
- Realizar la descripción conceptual y lógica de la base de datos: una vez especificados los requisitos de la información es preciso realizar el diseño conceptual de la base de datos para, después adecuar la estructura conceptual a un SGBD específico y, por tanto, también a un modelo convencional de datos concreto.
- Realizar la descripción física de la base de datos, encontrando una estructura interna que soporte el esquema lógico y los objetivos de diseño con la máxima eficiencia de los recursos de la máquina. Es una labor que se extiende a lo largo de la vida de la base de datos. El DBA tendrá que variar parámetros, reorganizar datos, modificar estructuras de almacenamiento, realizar nuevas distribuciones de ficheros en los soportes, etc.
- Realizar las especificaciones y vistas (o subesquemas) para los programas; del estudio de los requisitos de los usuarios se obtiene un conjunto de necesidades en cuanto a procesos a realizar sobre los datos; con esta información se tendrán que definir las vistas externas.
- Definir los estándares por los que se va a regir la organización en cuanto a documentación de la base de datos, metodologías de diseño de la misma, etc.
- Definir procedimientos de explotación y uso.
- Establecer y controlar todos los aspectos relativos a la seguridad de la información.
- Realizar el control y la interacción entre la red y la base de datos (para bases de datos accedidas a través de redes de telecomunicación o de bases de datos distribuidas).

2.10. TRANSACCIONES EN LOS SISTEMAS GESTORES DE BASES DE DATOS

Una transacción en un Sistema de Gestión de Bases de Datos (SGBD), es un conjunto de órdenes que se ejecutan formando una unidad de trabajo, es decir, en forma indivisible o atómica.

Un SGBD se dice que es transaccional si es capaz de mantener la integridad de los datos, haciendo que estas transacciones no puedan finalizar en un estado intermedio. Cuando por alguna causa el sistema debe cancelar la transacción, empieza a deshacer las órdenes ejecutadas hasta dejar la base de datos en su estado inicial (llamado punto de integridad), como si la orden de la transacción nunca se hubiese realizado.

Para esto, el lenguaje de consulta de datos SQL (Structured Query Language), provee los mecanismos para especificar que un conjunto de acciones deben constituir una transacción.

BEGIN TRAN: Especifica que va a empezar una transacción.

COMMIT TRAN: Le indica al motor que puede considerar la transacción completada con éxito.

ROLLBACK TRAN: Indica que se ha alcanzado un fallo y que debe restablecer la base al punto de integridad.

En un sistema ideal, las transacciones deberían garantizar todas las propiedades ACID (ACID es un acrónimo de Atomicity, Consistency, Isolation and Durability: Atomicidad, Consistencia, Aislamiento y Durabilidad), en la práctica, a veces alguna de estas propiedades se simplifica o debilita con vistas a obtener un mejor rendimiento.

EJEMPLO DE TRANSACCIÓN

Un ejemplo habitual de transacción es el traspaso de una cantidad de dinero entre cuentas bancarias. Normalmente se realiza mediante dos operaciones distintas, una en la que se decrementa el saldo de la cuenta origen y otra en la que incrementamos el saldo de la cuenta destino. Para garantizar la consistencia del sistema (es decir, para que no aparezca o desaparezca dinero), las dos operaciones deben ser atómicas, es decir, el sistema debe garantizar que, bajo cualquier circunstancia (incluso una caída del sistema), el resultado final es que, o bien se han realizado las dos operaciones, o bien no se ha realizado ninguna.

Mientras el SGBD maneja que la transacción sea atómica, el resto de usuarios puede estar realizando consultas. La forma en cómo el gestor maneja cómo ese resto de usuarios visualiza los datos mientras se está realizando la transacción, es lo que se conoce como nivel de aislamiento.

Los niveles de aislamiento se describen en función de los efectos secundarios de la simultaneidad que se permite, como las lecturas de datos sucios o lecturas fantasmas.

Los niveles de aislamiento de transacciones controlan lo siguiente:

- Si se realizan bloqueos cuando se leen los datos y qué tipos de bloqueos se solicitan
- Duración de los bloqueos de lectura
- Si una operación de lectura que hace referencia a filas modificadas por otra transacción:
    - Se bloquea hasta que se libera el bloqueo exclusivo de la fila
    - Recupera la versión confirmada de la fila que existía en el momento en el que se inició la instrucción o transacción
    - Lee la modificación de los datos no confirmada.

La selección de un nivel de aislamiento de transacción no afecta a los bloqueos adquiridos para proteger las modificaciones de datos. Siempre se obtiene un bloqueo exclusivo en los datos modificados en una transacción, bloqueo que se mantiene hasta que se completa la transacción, independientemente del nivel de aislamiento seleccionado para la misma. En el caso de las operaciones de lectura, los niveles de aislamiento de transacción definen básicamente el nivel de protección contra los efectos de las modificaciones que realizan otras transacciones.

Nivel de aislamiento	Lectura de datos sucios
Lectura no confirmada	Sí
Lectura confirmada	No

Oracle, que un SGBD de los más potentes, no permite visualizar cambios no validados. Hasta que no se haya hecho el **commit**, no se verán los datos actualizados.

MOTOR DE LA BASE DE DATOS

¿Qué es el motor de la base de datos?: es la forma en que almacena los datos, tablas, etc. físicamente y como los maneja.

MYSQL:

En un principio el motor que usaba era MyISAM: cuando se crea una tabla MyISAM se crean los siguientes ficheros:

.frm (definiciones)
.myd (datos)
.myi (índices)

Este motor es muy efectivo, ya que es realmente rápido, para lecturas, pero muy malo en gestión de transacciones. Cuando se realiza algún cambio en una tabla (un update, por ejemplo), bloquea TODA la tabla, con lo que esto conlleva…

Hace unos años, ORACLE creó un motor: InnoDB que sigue el paradigma "ACID" (ACID es un acrónimo de Atomicity, Consistency, Isolation and Durability: Atomicidad, Consistencia, Aislamiento y Durabilidad). MySQL lo adoptó, ya que las transacciones las realiza bloqueando únicamente el REGISTRO.

Cuando una tabla se crea InnoDB: se crea el fichero .frm (definición de la tabla) y los datos los almacena en un datafile (por defecto se llama ibdata1) (ver var/lib/mysql) al estilo de cómo lo hace oracle, que todos los datos los guarda en datafiles.

Cuando hice la aplicación creé alumnos y partes: InnoDB (que iban a ser las que se podían modificar continuamente) y las demás las dejé con el motor por defecto de MySQL (que si no lo cambiamos es MyISAM).

Para crear una tabla estableciendo nosotros el motor a utilizar:

Create Table kk (c1 char(1)) **ENGINE=InnoDB**

Para ver el motor por defecto que te coge, ver variables del sistema:

show variables like '%engine%' ;