

CONCEPTOS DE BASES DE DATOS*

Las actividades que realiza el hombre hacen necesario tener información acerca de cosas importantes.

Las “cosas” sobre las cuales se almacena información se conocen como *entidades*, pudiendo ser objetos tangibles (un estudiante, una computadora, un diskette) o tratarse de objetos intangibles (un suceso, una tarea, un estudio).

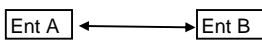
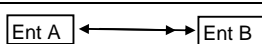
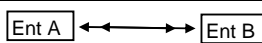
Toda entidad posee propiedades, algunas de las cuales conviene registrar eventualmente. Las propiedades sobre las que se registran datos se conocen como *atributos* de la entidad, conociéndose también como *ítems de datos*, cuando se hace referencia a ellos en forma independiente.

Los datos de la entidad se almacenan juntos, de modo que se facilite la obtención de información asociada. Los datos de una instancia de la entidad se almacenan en una secuencia fija de *campos* constituyendo un *registro* de un archivo.

Podemos ahora, por ejemplo, referirnos a una entidad **MAPA** con los atributos **Título**, **Escala** y **Proyección**. Para esta entidad podemos tener dos instancias o mapas específicos, cuyos valores están almacenados en dos registros (por ejemplo, “Mapa de Topografía”, “1:10000” y “UTM” para el primero, y , “Mapa Hidrográfico”, “1:15000” y “Mercator”, el segundo).

Las necesidades de información pueden comprender varias entidades relacionadas lógicamente entre sí. Así, podríamos requerir información sobre las entidades “Mapa”, “Itinerario” e “Investigador” dentro de un proyecto de investigación que comprende una zona geográfica.

Dependiendo del número de elementos (o registros) que pueden darse en cada entidad cuando se relaciona con otra, las relaciones que se dan entre las entidades pueden ser de tres tipos, los cuales se muestran en el cuadro siguiente:

Tipo de relación	Representación	Ejemplos
Uno a uno		Departamento – Jefe de Departamento
Uno a muchos		Departamento --- Docente
Muchos a muchos		Investigación --- Docente Curso --- Alumno

Con respecto al tipo de relación *uno a uno*, es fácil comprender que sólo puede haber un Jefe de Departamento para un Departamento; es decir, para una instancia de la entidad Departamento existe sólo una instancia posible de Jefe de Departamento, y viceversa. El tipo de relación más común es *uno a muchos*, también conocido como “padre-hijo”. Un Departamento puede tener varios Docente registrados, mientras que un Docente sólo puede pertenecer a un Departamento. El primer ejemplo mostrado para la relación *muchos a muchos* es claro: un proyecto de investigación (una instancia) puede incluir varios docentes (varias instancias), y un docente dado puede participar en varias investigaciones. Puede fácilmente derivarse una aproximación similar para el segundo ejemplo.

Las necesidades de almacenaje de datos de una determinada *aplicación* (solución de un problema de gestión mediante un sistema de información) se satisfacen mediante Bases de Datos.

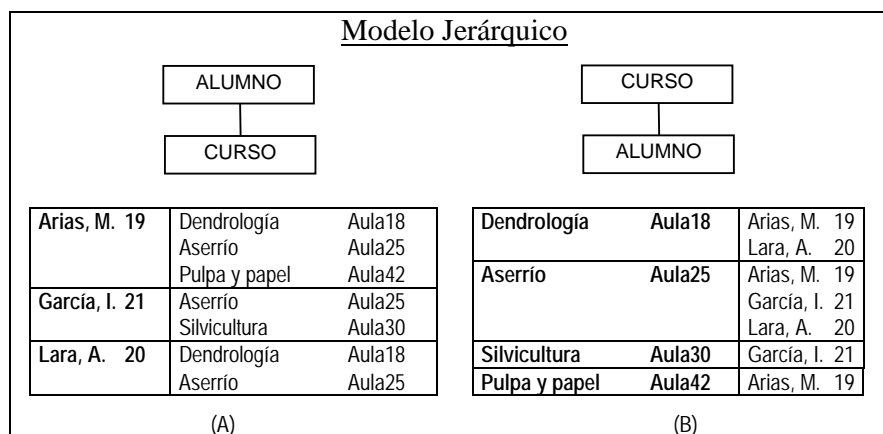
Para atender cabalmente las necesidades de almacenamiento que plantean los sistemas de información, se cuenta con Sistemas de Gestión de Bases de Datos, o SGBD, que son herramientas de software para organizar y manipular datos dentro de un sistema computarizado.

* Responsable: Carlos R. Vargas Salas

Casi todos los software efectúan algo de la función de administración de datos, pero el beneficio de un SGBD es que provee el soporte necesario para organizar los datos de modo que posibilite la manipulación y uso de una colección de datos por varios usuarios o programas (acceso concurrente). Igualmente, permite reducir la redundancia de datos, lograr su independencia respecto de los programas, otorgarles seguridad, y ofrece servicios para gestionar la base de datos.

Un *modelo de datos* es el método usado por un SGBD para organizar y recuperar los datos. Existen tres categorías básicas de modelos de datos: (a) de red, (b) jerárquica y (c) relacional. Mientras que las entidades son representadas en la misma forma en los tres modelos, las relaciones son representadas de manera distinta por cada uno.

El *modelo jerárquico* es el más antiguo de los tres y conlleva el ordenamiento jerárquico de los archivos que contienen a las entidades. En un diagrama de este tipo, aplicado a las entidades **CURSO** y **ALUMNO**, podemos organizar los datos agrupados según **CURSO** (alumnos que pertenecen a cada curso) o, alternativamente, agrupados por **ALUMNO** (cursos que pertenecen a cada alumno).



La búsqueda de cualquier dato sólo será eficiente siguiendo el orden jerárquico, de arriba hacia abajo. En el primer caso (A), se puede averiguar fácilmente cuáles cursos lleva Lara, pero difícilmente quién está llevando Dendrología. En el segundo caso (B) es fácil averiguar quién lleva Dendrología pero más difícil resulta averiguar qué cursos lleva Lara. Con un modelo jerárquico usted no puede tener las dos formas de agrupar^φ, debiendo decidir correctamente al escoger la manera de acceder a los datos.

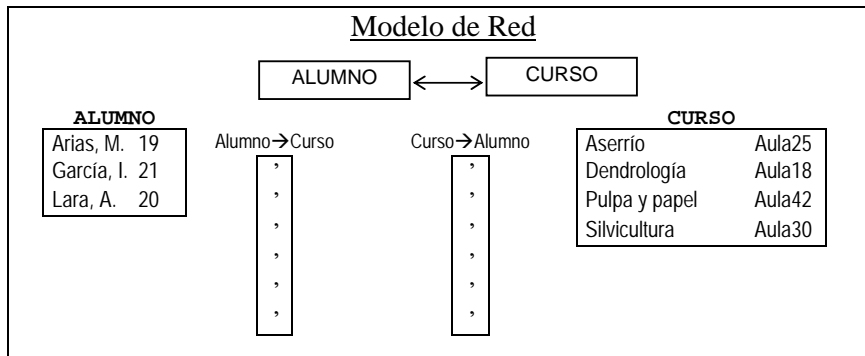
La falta de flexibilidad para la recuperación de datos es sólo una de las propiedades indeseables del modelo jerárquico. También es ineficiente en el uso del espacio porque genera redundancia. En el ejemplo A, los tres estudiantes están registrados en Aserrio. Los datos asociados con Aserrio están presentes en tres oportunidades, uno por alumno. La redundancia es ineficiente no sólo en cuanto al uso del espacio, sino también por la posibilidad que existe de que sea cambiado el dato de Aula en un/unos registro(s) y no en el/los otro(s), quedando la base de datos desactualizada e inconsistente.

El problema de redundancia mostrado surge debido a que la relación curso-alumno es del tipo muchos a muchos. El modelo jerárquico trabaja mal con este tipo de relaciones pero sí es muy eficiente para relaciones uno a muchos.

^φ En realidad sí podría hacerlo, guardando por duplicado los datos, con todo el costo que ello puede significar en almacenaje, mantenimiento y posibles inconsistencias.

Si uno de los principales propósitos de un SGBD es permitir que varios usuarios, con distintas aplicaciones, vean la base de datos en forma diferente, entonces el modelo jerárquico carece de la flexibilidad requerida.

El *modelo de red* o *plex* ofrece varias mejoras respecto del modelo jerárquico. El modelo de red (en inglés, “network model”) resuelve el problema de la redundancia de datos y de las relaciones muchos a muchos. En este modelo, cada entidad es representada como un archivo independiente, y las relaciones entre archivos son representadas por *enlaces* o *vínculos* (en inglés, “links”). Usualmente, cada enlace tiene un nombre y provee una conexión entre dos archivos.



En el diagrama mostrado hay dos enlaces: el enlace Alumno → Curso, y el enlace Curso → Alumno. Los enlaces son usados como apuntadores o índices, y son especificados cuando la base de datos es diseñada, por lo que no pueden ser cambiados fácilmente.

El modelo de red tiene varias buenas características. Primero, no se almacenan datos redundantes. Segundo, puede accederse a los datos eficientemente de muchas formas distintas, limitadas sólo por lo especificado cuando se diseñó la base de datos. El modelo completo de red permite lograr enlaces entre todos los archivos de la base de datos. Ello hace posible que cualquier tipo de relación sea fácilmente representada dentro del modelo.

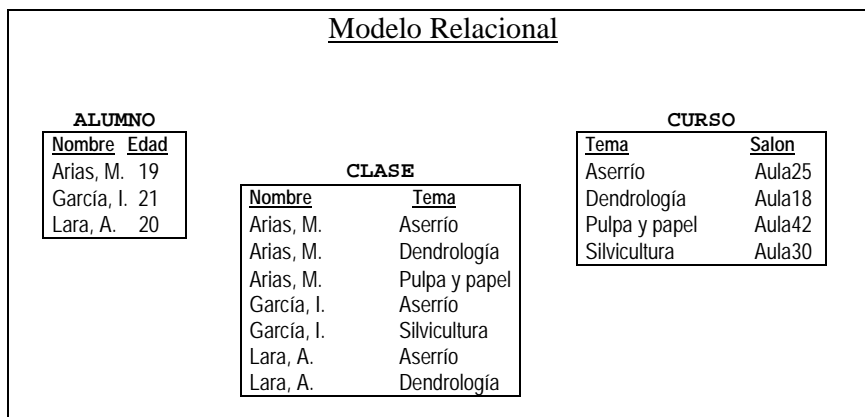
Aunque el modelo de red representa una significativa mejora sobre el modelo jerárquico, presenta algunas desventajas. Primero, aunque cualquier tipo de relación puede ser representado, es muy difícil agregar, cambiar o borrar relaciones de una base de datos de red. Tampoco es fácil agregar, cambiar o borrar archivos de la base de datos, debido al ajustado acoplamiento entre archivos y relaciones. Ello representa impedimento para el desarrollo progresivo de aplicaciones.

Otra desventaja es que en el momento de ser diseñada la base de datos los enlaces reciben un nombre, los cuales deben ser especificados por el usuario cuando realiza consultas en la base de datos. Ello hace que la búsqueda de información de distintas entidades relacionadas –que se lleva a cabo con el empleo de un *lenguaje procedimental*– sea compleja, ya que requiere definir los enlaces más convenientes para lograr eficiencia en la operación.

Una base de datos *relacional* es una base de datos construidas por medio de *relaciones*. Las relaciones se construyen organizando los ítems de datos como matrices bidimensionales, constituyendo así tablas o archivos planos bidimensionales. El *modelo relacional* involucra la *normalización*, proceso que sigue la construcción de estas tablas de datos de modo que se ajusten a algunas reglas que impedirán anomalías en la actualización de la base de datos, e inconsistencias dentro de ella.

Las tablas de datos poseen las siguientes propiedades generales:

- Cada entrada de la tabla representa un ítem de dato; no hay grupos repetitivos.
- Son homogéneas por columna: es decir, todos los ítems de una columna son de la misma clase.
- Cada columna tiene nombre propio.
- Todas las filas son diferentes; no se admiten filas duplicadas.
- Tanto las filas como las columnas pueden ser consideradas en cualquier secuencia y en cualquier momento, sin afectar por ello ni el contenido de información ni la semántica de cualquier función que utilice la tabla.



Un verdadero SGBD es un sistema que usa el modelo relacional para representar las distintas relaciones, y el método de acceso relacional para la recuperación de datos.

En forma simplificada, un sistema relacional:

1. Está constituido de únicamente archivos planos (tablas).
2. Permite al usuario escoger cualquier campo de cualquier tabla para la manipulación de datos (operador *project*).
3. Permite al usuario seleccionar (conforme a algún criterio) cualquier conjunto de registros para manipulación (operador *select*).
4. Permite que dos o más archivos planos sean juntados en un archivo lógico donde coinciden campos pertenecientes a registros en cada campo (operador *join*).

A través del empleo de los operadores *select*, *project*, y *join*, cualquier pregunta arbitrariamente compleja puede ser respondida. El modelo relacional parece ser similar al modelo de red, con la clara excepción de que las relaciones lógicas, representadas en el modelo de red por enlaces (acoplamientos físicos), son representadas en el modelo relacional como otro archivo de datos plano (vea la tabla **CLASE** en el gráfico). Es decir, las relaciones entre las entidades son lógicas y se mantienen así hasta que queramos derivar una nueva entidad que la represente físicamente, y esta nueva entidad tiene la misma forma que la de aquellas de las que se deriva.

El hecho que se mantengan las relaciones en una forma lógica y no sean necesarios los acoplamientos físicos entre archivos, otorga una mayor libertad para la adición, eliminación y cambios en la base de datos. Nuevos archivos pueden agregarse, y nuevas relaciones pueden establecerse sin afectar a la base de datos existente.

La base de datos puede ser diseñada lenta y progresivamente, un archivo a la vez, o ser modificada según lo exija el ambiente cambiante. Así, también las aplicaciones pueden construirse y ajustarse progresivamente, según lo determinen las necesidades del usuario.

Otra ventaja del modelo relacional sobre el de redes es que el usuario no necesita especificar el enlace más eficiente para la recuperación de datos.

Dado que los datos se almacenan en tablas bidimensionales, tan familiares para el usuario, este puede crear sus propios archivos y manipular datos a través de *queries* (consultas) interactivas empleando un lenguaje de comandos parecido a órdenes escritas en inglés.

Para resumir, las principales ventajas del modelo relacional son:

- a) Todas las relaciones comunes pueden ser representadas en la forma que son vistas por el usuario, sin tener que distorsionar la relación para ajustarse al modelo.
- b) La flexibilidad para cambiar la estructura de la base de datos rápida y fácilmente permite reflejar las necesidades del usuario. Estos cambios a menudo tienen poco o ningún impacto sobre las aplicaciones que acceden a los datos.
- c) Los datos pueden accederse eficientemente en muchas formas distintas sin que previamente haya que haber diseñado en la base de datos todas las posibilidades.
- d) La habilidad para consultar a la base de datos en una forma *no-procedimental* hace posible desarrollar lenguajes de consulta (“query lenguajes”, en inglés) de alto nivel que un usuario inexperto puede usar para ver los datos en la forma que desea.
- e) La naturaleza plana de los archivos relacionales permite una gradual conversión a la tecnología relacional, minimizando el impacto sobre los sistemas existentes.

Adaptación de:

Martin, James. 1977. Organización de las bases de datos. 1ra. edición en español, Prentice-Hall, México. 544p

Hasling, Bill. 1985. Introduction to Relational Database Technology. //V The 1985 National Database and Fourth Generation Language Symposium: Proceedings