

D B A I D

**Sistema Experto
para el diseño de
Bases de Datos**

**Herman E. Dolder
Esteban Lubomirsky**

Primera Edición

© 1988 DATA S.A. - Buenos Aires - Argentina

Prólogo

Este libro es la cuarta obra de una serie sobre diseño de bases de datos que se iniciara en 1980 con "Análisis de Datos y Diseño de Bases de Datos", continuara en 1982 con "Diseño Conceptual e Implementación de Bases de Datos" y en 1986 con "Diseño de Bases de Datos utilizando conceptos y técnicas de Inteligencia Artificial".

La última de las obras mencionadas describe el funcionamiento y utilización de un sistema computarizado denominado DBAID, diseñado e implementado como sistema de ayuda inteligente para el diseño de Bases de Datos.

La descripción corresponde a la primer versión de DBAID a la que denominamos "versión cero" la cual fue implementada en un computador de gran envergadura.

Para la fecha de publicación del libro, realizada por la Editorial Universitaria de Buenos Aires, nos encontrábamos desarrollando una versión para computadores personales.

La presente obra describe las funciones y la utilización de la segunda versión para computadores personales, diseñada teniendo en cuenta la experiencia obtenida con las versiones anteriores así como el nuevo conocimiento disponible sobre el tema.

Es también un libro sobre diseño de Bases de Datos y contiene información útil para especialistas que no tienen acceso a DBAID. Creemos que su contenido conceptual, resultado del continuo trabajo en el tema durante más de 16 años, también lo hace apto como libro de texto.

Una de las novedades de esta obra es la aparición como coautor del Ing. Esteban Lubomirsky, quien es actualmente responsable del proyecto DBAID en DATA S.A..

Tanto el Ing. Lubomirsky como yo deseamos que los lectores disfruten de la lectura de esta obra y que su contenido les ayude a comprender mejor el proceso de diseño así como a diseñar bases de datos en forma más rápida y confiable.

Ing. Herman E. Dolder

Contenido

	Página
1. Introducción	8
1.1. . Contenido del libro	9
1.2. . Principales características del Sistema	10
1.3. . Ambiente Operativo requerido.	11
2. Modelos de Datos	12
2.1. . Sobrecarga Semántica.	13
2.2. . Potencia Semántica.	13
2.3. . El Modelo de Datos Relacional	14
2.3.1. . Dominios	17
2.4. . Extensión del modelo relacional para . captar más significado	18
2.5. . Modelización Conceptual Semántica . utilizando el sistema.	19
2.6. . Captación de conocimiento semántico . en el sistema.	19
2.7. . Utilización del conocimiento semántico . por el sistema	20
2.8. . Filosofía de la modelización conceptual	21
3. El Modelo de Datos Conceptual utilizado . en el sistema	24
3.1. . Representación del Conocimiento	24

3.2.	. Modelo de Datos Conceptual Semántico.	26
3.2.1.	. Estructura de Datos.	26
3.2.1.1.	. Conceptos Genéricos	26
3.2.2.	. Restricciones.	27
3.2.3.	. Operaciones.	27
3.3.	. Categorías Semánticas para los Conceptos.	27
3.3.1.	. Ejemplos de categorización de conceptos	29
3.4.	. Vinculaciones Primitivas entre Conceptos.	31
3.4.1.	. Cardinalidad de la Vinculación entre dos conceptos	32
3.5.	. Ejemplos de modelos conceptuales muy simples	33
3.6.	. Vinculaciones Lingüísticas.	34
3.7.	. Análisis de las Vinculaciones Primitivas.	37
3.7.1.	. Generalización	37
3.7.1.1.	. La Generalización vista desde el Modelo Relacional.	39
3.7.2.	. Agregación	39
3.7.3.	. Propiedad.	41
3.7.4.	. Circunstancias de los Eventos.	41

3.8.	. Modelización de vinculaciones con	
	. cardinalidad [M:N]	43
3.9.	. Concepto vinculado consigo mismo.	44
3.9.1.	. Cuando la cardinalidad es [M:N].	44
3.9.2.	. Cuando la cardinalidad es [1:N].	45
3.9.3.	. Cuando la cardinalidad es [1:1].	46
3.10.	. Precedencia de Conceptos.	46
3.11.	. Precedencia de Eventos.	47
3.12.	. Eventos Implícitos.	48
3.13.	. Grupos.	49
3.14.	. Elección de las Categorías Semánticas	49
3.14.1.	. La perspectiva de Aristóteles	
	. o Nada es nuevo bajo el sol	49
3.14.2.	. Desde la perspectiva de la	
	. Lingüística	50
3.14.3.	. Desde la perspectiva de la	
	. Inteligencia Artificial	51
3.14.4.	. Criterios utilizados en el diseño	
	. del sistema	52
4.	Modelización Conceptual.	54
4.1.	. Etapas en la modelización conceptual.	54
4.2.	. Vistas.	55
4.2.1.	. Vistas de Usuario.	56
4.2.2.	. Vistas de Contexto	56

4.3.	.	Obtención de Vistas	56
4.4.	.	Ejemplos de Vistas.	57
4.5.	.	Ejemplo de Observación (obtención de vistas). . .	57
4.6.	.	Ejemplos de Abstracción y Formalización	58
4.7.	.	Interacción con el sistema.	61
4.8.	.	Validación de la Información provista	
	.	en las microvistas	61
4.9.	.	Ejemplo de Modelización Conceptual.	62
5.		Verificación del Modelo.	67
5.1.	.	Funciones de Visualización.	68
5.2.	.	Funciones de Verificación	68
5.2.1.	.	Verificaciones de correspondencia	
	.	semántica	71
5.3.	.	El modelo en su Forma Canónica.	72
5.3.1.	.	Ventajas de la forma canónica.	73
5.3.2.	.	Desventajas de la forma canónica	73
5.4.	.	El Tiempo y las Bases de Datos.	74
5.4.1.	.	El tiempo y los datos en una	
	.	Base de Datos	74
5.4.2.	.	El tiempo y la representación de	
	.	Objetos en la Base de Datos	75
5.4.3.	.	Puntos e Intervalos de tiempo.	75
5.4.4.	.	Conceptos y Vinculaciones Temporales	76
5.4.5.	.	El tiempo y los DBMSs relacionales	77
5.4.6.	.	Evolución de los Esquemas en el tiempo	78

5.5.	. Introduciendo Futuro en el Modelo Conceptual. . .	78
6.	Diseño Lógico y Físico	81
6.1.	. El foco de atención cambia.	81
6.2.	. Generación del modelo relacional . normalizado.	81
6.3.	. Cardinalidad de los Eventos	83
6.4.	. Consideraciones sobre la identificación . unívoca de Tuplas.	84
6.5.	. Optimización Lógica	88
6.5.1.	. Optimización del espacio de . almacenamiento.	88
6.5.2.	. Optimización del tiempo de respuesta . para aplicaciones críticas.	89
6.5.2.1.	. Introduciendo atributos derivables.	90
6.5.2.2.	. Agregando vinculaciones redundantes	90
6.5.2.3.	. Desnormalizando relaciones.	91
6.5.3.	. Absorción de relaciones.	93
6.5.4.	. División de Relaciones	95
6.5.5.	. Facilidades de Optimización de los RDBMS . . .	95
6.6.	. Ventajas y desventajas de la . desnormalización	97
6.7.	. Facilidades de Optimización Lógica . provistas por el sistema	98
7.	Síntesis de las etapas de diseño	99

8.	Adaptación a manejadores de bases de datos	
	no relacionales100
8.1.	. Para DBMSs de tipo Jerárquico100
8.2.	. Para DBMSs de tipo Red.102
8.3.	. Para DBMSs Seudorelacionales.103
8.4.	. Para Archivos Secuenciales.103
9.	Resumen de las principales características	
	del sistema104
10.	Anexos:	
10.1.	. A: Lenguaje de consulta casi-natural.106
11.	Referencias Bibliográficas109

1. Introducción

La tecnología de bases de datos ha sido, sin lugar a duda, uno de los desarrollos claves en la evolución de la informática ya que ha aportado la capacidad de integrar la información permitiendo que múltiples usuarios y aplicaciones puedan compartirla en un ambiente que garantiza su seguridad e integridad.

Sin embargo, el completo aprovechamiento de las posibilidades que ofrece esta tecnología requiere un correcto diseño de las estructuras de datos que conformarán el contenido de las bases de datos.

Los datos de una base de datos se almacenan y se procesan en el computador por medio de sistemas administradores de bases de datos (DBMS), los cuales responden a diferentes "modelos de datos" (jerárquicos, de redes y relacionales).

En ese contexto un modelo de datos describe la forma de representación de la información utilizada por un DBMS en particular. Así se dice que un DBMS Relacional responde a un Modelo de Datos Relacional [Codd, 1970] (ver "2. Modelos de Datos").

Estos modelos de datos, que describen como se almacenan los datos en el computador, resultan inadecuados para representar el significado de los datos almacenados.

En los últimos años han surgido metodologías de diseño de bases de datos basadas en la utilización de modelos de datos orientados a la descripción de la realidad que estará representada en las bases de datos [Hull, 1987].

Estos modelos, que se denominan modelos "conceptuales" o "semánticos", están orientados a la descripción de la realidad captando en mayor medida el significado de los datos y de sus vinculaciones en el mundo real.

Los diferentes modelos de datos conceptuales propuestos proveen una variedad de niveles de abstracción y de recursos para la modelización. Existen modelos simples y concretos como el Modelo de Entidades-Relaciones [Chen, 1976] así como modelos más abstractos y con mayor poder semántico tales como los Modelos Semánticos desarrollados en el área de la Inteligencia Artificial.

Existen entonces dos tipos de modelos de datos, uno orientado al diseño de las estructuras de datos que compondrán las bases de datos y el otro orientado a describir cómo se almacenan los datos en el computador. Los modelos del primer tipo se denominan Modelos Conceptuales o Modelos Semánticos, mientras que los del segundo tipo están orientados a describir como se almacenan los datos en el computador, por lo que nos referiremos a ellos como Modelos de Almacenamiento.

En este libro describimos un sistema inteligente para el diseño de bases de datos que utiliza un modelo de datos conceptual de elevado nivel de abstracción y de gran potencia semántica para el diseño de las estructuras de datos.

Una vez construido el modelo conceptual el sistema obtiene automáticamente el modelo relacional equivalente, a partir del cual se implementa una bases de datos relacional o se traduce el modelo relacional a alguno de los otros modelos de almacenamiento (jerárquicos, de red y seudorelacionales).

El sistema descrito, denominado DBAID, utiliza conceptos y técnicas de inteligencia artificial conformando lo que en estas disciplinas se conoce como un Sistema Experto. DBAID es un sistema de diseño asistido por computadora que posee un elevado nivel de inteligencia.

El sistema releva al diseñador de procesos tales como el de "integración de vistas" y el de "normalización", comunes a la mayoría de las metodologías de diseño actuales. Estos procesos son realizados por el sistema en forma automática, requiriendo sólo la intervención del diseñador en la resolución de situaciones ambiguas o incompatibles.

El sistema encapsula un conjunto de ideas y un método de diseño conceptual perfeccionados a través de una prolongada y continua exposición al diseño de bases de datos para una muy amplia gama de aplicaciones [Dolder, 1984, 1987].

Se ha aprovechado asimismo el conocimiento sobre teoría de diseño de bases de datos y metodologías disponible en la bibliografía (si bien por simplicidad se han minimizado las referencias bibliográficas en este libro) y se ha incorporado el conocimiento que los diseñadores del sistema tienen en temas tales como interfases inteligentes, sistemas expertos y semántica computacional.

1.1. Contenido del Libro

Este libro describe la base conceptual, las principales funciones del sistema y las etapas de diseño. No se incluyen detalles de implementación, los cuales pueden visualizarse en una demostración del sistema (por un importe reducido puede obtenerse de DATA SA un diskette de demostración).

Asumimos en el tratamiento de los temas que el lector tiene conocimientos teóricos y prácticos, sobre bases de datos relacionales. De no ser así sugerimos la lectura previa de "An Introduction to Database Systems" de C.J.Date (Addison-Wesley).

Si el lector considera necesario profundizar en el tema de los Modelos de Datos le sugerimos el libro "Data Models" de D.C.Tsichritzis y F.H.Lochoovsky (Prentice-Hall). En este libro encontrará, además, una muy amplia referencia bibliográfica sobre Bases de Datos y Modelos de Datos.

1.2. Principales Características del Sistema

DBAID es un sistema experto que provee todas las facilidades necesarias para diseñar e implementar bases de datos cualquiera sea su envergadura y complejidad.

El sistema presenta cuatro características principales:

- Posibilita que diseñadores con reducida experiencia en diseño de bases de datos puedan diseñar bases de datos de gran envergadura y complejidad.
- Cubre todas las etapas del diseño normalizado hasta la implementación física de la base de datos.
- Provee al diseñador una interfase de diseño de alta productividad.
- Explota la tecnología de Inteligencia Artificial.

Posibilita un Diseño Rápido y Seguro con Mínima Experiencia Previa

El sistema es el resultado de la prolongada y exitosa experiencia de DATA S.A. en el diseño de bases de datos para las más diversas áreas de aplicación. Esta experiencia ha sido encapsulada en el sistema posibilitando que diseñadores con mínima experiencia previa puedan diseñar, en forma autónoma, fácilmente y con completa seguridad, bases de datos de cualquier envergadura y nivel de complejidad (por ejemplo, bases de datos relacionales con más de 100 relaciones y más de 1000 atributos diferentes).

Cubre Todas las Etapas de Diseño

El diseñador, a partir de requerimientos de información de los usuarios y descripciones de la organización o ambiente de aplicación para el cual desea diseñar la base de datos, construye y refina un modelo conceptual de datos con la cooperación del sistema. Una vez terminado y verificado el modelo conceptual el sistema obtiene automáticamente el modelo relacional equivalente generando el esquema relacional normalizado correspondiente, así como los comandos requeridos para la implementación física de la base de datos.

Provee una Interfase de Diseño de Elevada Productividad

La comunicación con el diseñador fue ideada para obtener la más alta productividad en todas las etapas de diseño. Se utilizan múltiples ventanas, menús "pull-down", comunicación en lenguaje casi-natural, representaciones gráficas y un vasto sistema de ayudas "en contexto".

Toda la información requerida por el diseñador para operar el sistema es accesible a través del sistema de ayudas en contexto.

Las facilidades disponibles permiten maximizar la productividad del diseñador, así como minimizar el costo de su entrenamiento inicial.

Utiliza Tecnología de Inteligencia Artificial

DBAID utiliza la avanzada tecnología de los Sistemas Expertos y la Semántica Computacional.

El sistema puede captar y comprender el significado de los conceptos del área de aplicación, así como de sus vinculaciones. Esta facilidad minimiza la probabilidad de cometer errores semánticos en el diseño.

Un error semántico tiene origen en una interpretación equivocada por parte del diseñador del significado de uno o varios conceptos, o de sus vinculaciones, durante el diseño.

Un error semántico se manifiesta fundamentalmente como una imposibilidad (total o parcial) de reflejar en el contenido de la base de datos un cambio de estado ocurrido en el mundo real que ésta representa.

Los errores semánticos generalmente se descubren recién en la etapa de implementación de la aplicación resultando su corrección de costo muy elevado.

La utilización de información semántica en el sistema constituye la ventaja comparativa más importante de este sistema en relación con otros sistemas de diseño, facilita además al diseñador la completa comprensión del área de aplicación.

1.3. Ambiente Operativo Requerido

El sistema opera sobre computadores personales compatibles con los IBM PS, PC/AT o PC/XT, operando bajo MS-DOS, PC-DOS, o compatible, con 512 K de memoria, monitor gráfico y disco rígido con 3 MB disponibles.

El sistema se distribuye en un conjunto de diskettes. Entre los archivos suministrados se incluye un archivo denominado READ.ME el cual puede contener ampliaciones o correcciones al contenido de este libro.

2. Modelos de Datos

Conocimiento es el entendimiento de la naturaleza, cualidades y relaciones de las cosas.

La constante adquisición de conocimiento constituye una de las características esenciales de nuestra naturaleza humana. Permanentemente buscamos aumentar nuestro conocimiento acerca de nosotros mismos y del ambiente que nos rodea.

Cuando percibimos u observamos un fenómeno, obtenemos una porción incremental de conocimiento. A estos incrementos de conocimiento denominamos "información" [Langefors, 1977].

La información valiosa tiene que ser registrada y también comunicada a otras personas en una forma simbólica que pueda ser entendida. Para estos fines resulta de importancia fundamental contar con una adecuada representación simbólica de la información.

La pieza más elemental de información es el "dato" (por ejemplo una cifra de sueldo). Un dato elemental no es de utilidad por sí mismo, es útil sólo cuando está vinculado con otros datos (por ejemplo la identificación de un empleado), por lo que el almacenamiento y la comunicación de información requieren la construcción de "estructuras de datos".

La estructura y el contexto (en primera aproximación) le dan "significado" a los datos posibilitando su entendimiento.

El lenguaje natural es nuestra forma primaria de representación y comunicación de información. Si bien es el medio más generalizado para representar y comunicar información, no siempre constituye el mejor medio.

En muchas situaciones resulta más útil establecer formas especializadas de representación de la información. Las fórmulas matemáticas, los mapas carreteros y las partituras musicales, constituyen ejemplos de "modelos" especializados de representación simbólica de la información.

El almacenamiento, procesamiento y distribución de información mediante computadores y redes de comunicación ha creado la necesidad de encontrar modelos de representación adecuados a estos fines. Se han explorado y desarrollado diferentes "Modelos de Datos" (Relacionales, Jerárquicos, de Red, Semánticos, etc.) para la representación de la información en los sistemas computarizados.

Hemos mencionado en el capítulo "1. Introducción" que además de los modelos de datos orientados a describir como se almacenan los datos en un computador, se han introducido modelos de datos de mayor nivel de abstracción y potencia semántica orientados al diseño de las estructuras de datos que compondrán las bases de datos. Este nuevo tipo de modelo de datos recibe la denominación de Modelo Conceptual o Semántico.

2.1. Sobrecarga Semántica

En la actualidad no existen DBMS que respondan a modelos de elevado nivel de abstracción como los que se utilizan durante el diseño por lo que es necesario traducir el modelo conceptual al modelo de almacenamiento que se utilizará (por ejemplo el modelo relacional).

Esta traducción se hace con pérdida de información semántica. Se dice que el modelo de datos de menor nivel de abstracción resulta "sobrecargado semánticamente" ya que al contar con una menor variedad de elementos de modelización cada elemento debe representar varios elementos del modelo de mayor nivel de abstracción y potencia semántica.

Los modelos de datos en general especifican:

- La estructura de los datos
- Restricciones sobre la estructura
- Operaciones sobre las estructuras

Un modelo de datos establece reglas para la estructuración de los Datos (Mediante Tablas, Arboles, Redes, etc.), define las operaciones permitidas sobre las estructuras (en general para extraer nuevas estructuras de las existentes), y permite establecer restricciones sobre las estructuras para que los datos mantengan su coherencia informativa o "integridad semántica". Estas restricciones pueden abarcar una gran cantidad de aspectos. Por ejemplo, pueden especificar restricciones provenientes del mundo real tales como:

- Que los empleados no puede ganar más que sus jefes.
- Que para inscribirse en un curso, un estudiante debe satisfacer ciertos prerequisites.
- Que la cantidad de pacientes internados en un hospital no debe ser mayor que la cantidad total de camas existente.

Además, de Restricciones explícitas, como las anteriores, un modelo de datos introduce restricciones implícitas como consecuencia de sus propias características y limitaciones de modelización.

2.2. Potencia Semántica

En las secciones anteriores introdujimos, sin definirlo, el concepto de "potencia semántica". La potencia semántica de un modelo está determinada por su capacidad de representar no sólo la estructura de los datos sino también "la interpretación o significado" de los mismos y de sus interrelaciones.

Por ejemplo, en la frase:

"Su altura es de 5 metros"

el dato es el número "5" y el significado es "altura en metros".

La utilización del computador para el procesamiento de información ha resultado, en general, en una separación entre los datos y su significado. En general los computadores procesan sólo los datos y no se registran explícitamente los significados, dejando al usuario (o a los programas de tratamiento) la tarea de interpretación.

A medida que la utilización de los computadores evoluciona hacia aplicaciones más sofisticadas resulta cada vez más necesario captar y utilizar activamente el significado de los datos (y de sus vinculaciones).

A continuación revisaremos el Modelo Relacional, concentrándonos en su capacidad de estructurar datos y en su potencia semántica, ya que ambos aspectos tienen relación directa con el diseño de una Base de Datos Relacional.

2.3. El Modelo de Datos Relacional

El Modelo de Datos Relacional es extremadamente simple. Es un modelo orientado al almacenamiento.

Las Bases de Datos Relacionales están basadas en este modelo de datos introducido por E.F.Codd en 1970 (ver [Codd, 1970] y [Codd, 1982]).

En este modelo de datos, la estructura de datos básica es la Relación normalizada.

Una Relación es básicamente una tabla de valores "plana" (de dos dimensiones: Filas y Columnas), en la que las columnas reciben el nombre de "Atributos" y las filas el nombre de "Tuplas".

Cada Tupla de una Relación es diferente de las demás y está identificada unívocamente mediante una Clave Primaria (subconjunto de Atributos de una Relación cuyos Valores identifican unívocamente una Tupla en la Relación).

Tanto las filas (Tuplas) como las columnas (Atributos) pueden ser consideradas en cualquier momento en cualquier secuencia sin que ello afecte el contenido de información de la tabla ni el significado de sus datos.

El esquema anterior se representa también mediante la siguiente notación:

HOSPITALES(IdHospital, NombreDelHospital,
CantidadDeCamas)

PABELLONES(IdPabellón, IdHospital, NombreDelPabellón,
CantidadDeCamas)

MEDICOS(IdMédico, IdHospital, NombreDelMédico,
Especialidad)

Las Restricciones en el Modelo Relacional son básicamente:

- De Clave. En una Relación cada Tupla debe tener valor diferente en su Clave Primaria. Los valores de la Clave Primaria no pueden ser modificados.
- De Integridad Referencial. Las que especifican, por ejemplo, que para insertar los datos de un nuevo médico en la Relación MEDICOS es prerequisite que el hospital correspondiente exista en la Relación HOSPITALES.

O que el prerequisite para eliminar los datos de un determinado Hospital en la Relación HOSPITALES, es que no haya médicos asignados a dicho hospital en la Relación MEDICOS ni pabellones de dicho hospital en la Relación PABELLONES.

Los Operadores del Modelo Relacional permiten obtener nuevas Relaciones a partir de un conjunto de Relaciones dadas.

E.F.Codd definió tanto un "Algebra Relacional" como un "Cálculo Relacional". El Algebra Relacional incluye las siguientes operaciones:

SELECCION	-	PROYECCION	-	UNION	-
INTERSECCION	-	FUSION (JOIN)	-	SUSTRACCION	-
PRODUCTO CARTESIANO			-		-

La potencia del Algebra Relacional surge no sólo de los operadores en sí, sino de la forma en que éstos se pueden combinar. El resultado de aplicar un operador a una relación es otra relación, por lo que, mediante la aplicación secuencial de operadores, se puede obtener el resultado buscado.

Los operadores del Algebra Relacional están definidos para la recuperación (y no para la actualización) de la información contenida en Relaciones normalizadas (ver los temas "Dependencia Funcional" y "Formas normales" en [Date, 1981], [Codd, 1982], [Kent, 1983] y [Smith, 1985]).

El Algebra Relacional está basada en el Algebra Matemática. El Cálculo Relacional, de mayor nivel conceptual que el Algebra, está basado en la Lógica Matemática, en particular en la Lógica del Cálculo de Predicados.

Lenguajes tales como el QUEL y el ahora standard SQL están basados en el Cálculo Relacional y contienen extensiones para la actualización de los datos y de los esquemas de las bases de datos.

Codd demostró que el Algebra y el Cálculo Relacionales son equivalentes en el sentido de que permiten obtener las mismas respuestas cuando se aplican al mismo conjunto de Relaciones.

El Modelo Relacional no impone ninguna disciplina sobre el significado (la semántica) de las Tablas y de sus Atributos. Por ejemplo, en la tabla de MEDICOS anterior, podría incluirse un atributo "SABOR" que puede tomar valores tales como "Frutilla", "Vainilla" y "Chocolate", lo cual no corresponde a ninguna situación observable en el mundo real.

La responsabilidad de establecer una disciplina semántica en el Modelo Relacional queda en manos de diseñador lo cual no resulta fácil debido a que los únicos "recipientes" provistos para el almacenamiento del significado de los datos y sus vinculaciones son los nombres de las Relaciones y de los Atributos (y sus características computacionales).

Una de las limitaciones importantes del Modelo Relacional es que la modelización se realiza a un nivel de abstracción más bajo del cual uno piensa al diseñar una aplicación. El Modelo Relacional obliga a modelizar en términos de Relaciones, Atributos, Tuplas, etc., que si bien es mejor que modelizar en términos de Archivos, Bytes y Bits, no es lo mismo que modelizar en términos del mundo real, por ejemplo, de Personas, Organizaciones, Lugares, que es lo que uno generalmente quisiera al desarrollar una aplicación.

2.3.1. Dominios

El Modelo Relacional básico permite especificar DOMINIOS para los Atributos. En el ejemplo anterior, es posible especificar que el atributo CantidadDeCamas sólo puede tomar valores correspondientes al dominio de los números enteros positivos dentro de un cierto rango.

Dos o más atributos pueden compartir un mismo dominio de valores.

Para aumentar la potencia semántica del modelo relacional, en diversas implementaciones se han definido extensiones al modelo relacional básico. Algunas de estas extensiones están relacionadas con el concepto de Dominio.

Por ejemplo, en SYSTEM R (IBM) se han introducido diferentes tipos de dominios tales como:

- Dominios de Alcance ("scope"): Permiten restringir los posibles valores que puede tomar un atributo.
- Dominios de Comparabilidad: Permiten determinar si valores pertenecientes a diferentes atributos (de una o más relaciones) son comparables. Definiendo diferentes dominios se puede impedir la comparación o asociación de dos atributos con diferente significado, por ejemplo IdMedico con IdHospital.
- Dominios de Unidades de Medida: Permiten impedir que se realicen operaciones aritméticas si dos cantidades no tienen unidades de medida compatibles.

Estas facilidades permiten definir Dominios con contenido semántico (también llamados "Dominios Interpretados"), proveyendo un medio para captar significado.

Por definición, si dos atributos de diferentes relaciones pertenecen al mismo dominio ellos representan el mismo tipo de concepto. Sólo si dos atributos pertenecen al mismo dominio, pueden ser relacionados (por ejemplo, comparados por "=", ">", etc).

Los Dominios proveen un medio para asegurar la calidad de la información almacenada y procesada.

La facilidad de Dominios, así como la de restricción de la Integridad Referencial, si bien son componentes muy importantes del Modelo Relacional, aún no se han implementado en la mayoría de los manejadores de bases de datos relacionales (RDBMSs) disponibles comercialmente.

2.4. Extensión del Modelo Relacional para Captar más Significado

Hemos caracterizado al modelo relacional, en términos de nivel de abstracción y potencia semántica, como un modelo de bajo nivel, por lo que es previsible que evolucionará hacia (o será en el futuro sustituido por) un modelo de más alto nivel.

E. F. Codd en su artículo "Extending the database relational model to capture more meaning" (ACM Trans. Database Syst. 4, 1979) propone un nuevo modelo relacional denominado "RM/T", que incorpora esencialmente la tipificación de las Relaciones (Property graph relations, Association graph relations, Characteristic graph relations, etc.) y la representación y especificación de varios tipos de restricciones sobre las Relaciones y sus vinculaciones lógicas. Hasta el momento esta propuesta no se ha materializado en un manejador de bases de datos.

2.5. Modelización Conceptual Semántica Utilizando el Sistema

Debido a las limitaciones del Modelo de Datos Relacional (reducida potencia semántica y bajo nivel de abstracción) para modelizar situaciones del mundo real, en DBAID se emplea un Modelo Conceptual Semántico, el cual es descrito en el capítulo "3. El Modelo de Datos Conceptual utilizado en el sistema".

Durante el diseño con DBAID el diseñador utiliza su conocimiento del significado de los conceptos de la aplicación, si bien este conocimiento no aparecerá representado en el esquema de la base de datos relacional, debido a las limitaciones semánticas del modelo relacional. De esta manera el conocimiento semántico actúa principalmente como "catalizador" en el proceso de diseño.

Sin embargo, la disponibilidad de facilidades de modelización de elevado nivel de abstracción y potencia semántica hace al diseñador más cómoda su tarea, permitiéndole expresar en forma más amplia y completa su conocimiento del ambiente de la aplicación para el cual diseña la base de datos.

Un objetivo principal en el diseño del sistema ha sido brindar al diseñador facilidades de modelización que le permitan utilizar durante el diseño su conocimiento semántico del "mundo de la aplicación" de la manera más completa y eficaz posible, para asegurar la calidad del mismo y automatizar en lo posible todas sus etapas.

2.6. Captación de Conocimiento Semántico en el Sistema

Durante la construcción del modelo conceptual el diseñador codifica la información semántica utilizando un conjunto de "Categorías Semánticas".

Este conjunto de categorías constituye un núcleo de conocimiento preestablecido fijo, conocido por el sistema y el diseñador. Utilizando este "acuerdo semántico", diseñador y sistema pueden comunicarse e interactuar cooperativamente.

El sistema, teniendo la capacidad de comprender el significado de los conceptos manejados por el diseñador, puede realizar tareas que demandan normalmente un alto grado de inteligencia.

La adopción de categorías semánticas que corresponden a conceptos del mundo real (Personas, Organizaciones, Objetos Físicos, Dispositivos, Tiempos, Lugares, etc.) permiten al sistema realizar verificaciones semánticas, efectuar transformaciones en el modelo automáticamente sin pérdida de significado y aportar elementos o estructuras predefinidas (Templates). También posibilita la interacción mediante una comunicación en lenguaje casi-natural.

2.7. Utilización del Conocimiento Semántico por el Sistema

Mencionamos previamente que el sistema utiliza activamente el conocimiento semántico disponible para asegurar el correcto diseño de la base de datos y para automatizar en lo posible todas las etapas del mismo.

Como ejemplo introductorio, consideremos las facilidades provistas por los Templates, suponiendo que, por ejemplo, el diseñador informa al sistema que "Empleado" es una "Persona" (una de las categorías semánticas). En esta situación el sistema ofrece al diseñador la incorporación de una estructura conceptual predefinida (Template) que contiene entre otros los siguientes elementos:

Nombre y Apellido
Fecha de Nacimiento
Documento de Identidad
Domicilio
.....

pudiendo el diseñador seleccionar aquellos elementos que desea incorporar al modelo.

Más adelante veremos otros importantes usos del conocimiento semántico provisto por el diseñador al sistema.

2.8. Filosofía de la Modelización Conceptual

En nuestra metodología consideramos como punto de partida las siguientes ideas:

1. Una base de datos es una colección de hechos registrados que reflejan el estado de ciertos aspectos de interés del mundo real. En todo momento el contenido ("extensión") de la base de datos representa una instantánea del estado del sistema real. Cada cambio en un valor almacenado en la base de datos refleja un cambio ocurrido o planificado en el sistema real.
2. Un modelo de datos expresa la estructura y las reglas de funcionamiento de un sistema. Expresa, por ejemplo, la estructura organizativa y las reglas de funcionamiento de un hospital.
3. Los modelos de datos resultan definidos por el sistema que representan.
4. Los modelos de datos permiten diseñar la estructura (esquema o "intensión") de una base de datos de manera de que ésta sea un espejo de la estructura del sistema que la misma modeliza.
5. Como consecuencia de las consideraciones anteriores se puede inferir que la estructura y las reglas de funcionamiento del sistema real determinan objetivamente la estructura de la base de datos, con independencia del diseñador. El diseñador se limita a decidir el alcance del modelo (ver punto siguiente) y a modelizar conceptualmente el sistema real, del modelo resultante se deriva objetivamente la estructura para la base de datos.
6. Los aspectos del mundo real reflejados en una base de datos definen su "alcance" (amplitud) y "granularidad" (grado de detalle). La decisión sobre cuáles aspectos reflejar resulta de importancia fundamental.
7. Un ambiente de aplicación (banco, hospital, universidad, industria, etc.) es un sistema realimentado por información que evoluciona con el tiempo. Es posible identificar en él múltiples flujos (dinero, materiales, personas, equipos, órdenes, etc.) siendo el flujo de información el que interconecta y posibilita el funcionamiento del sistema completo.

8. Dentro de un ambiente de aplicación, cada grupo, actividad, proceso, persona, función y sistema tiene su propia percepción dinámica de la realidad, su propio vocabulario y su propia valoración de lo que es importante. La empresa típica es una composición de estas visiones.
9. El resultado del diseño conceptual es un modelo de datos conceptual construido desde el punto de vista de los usuarios y de la aplicación, con independencia de como los datos van a ser almacenados y procesados en el computador.

En una etapa posterior de "diseño lógico" se traduce el modelo conceptual al modelo de almacenamiento que corresponde al manejador de bases de datos (DBMS) a utilizar, se realiza la adaptación a dicho manejador, y a los aspectos de procesamiento (procesos, frecuencias, tiempos de respuesta, prioridades, etc) resultando un esquema lógico.

Finalmente en la etapa de "diseño físico" se realiza la adaptación al computador contemplando las características y limitaciones del mismo.

3. El Modelo de Datos Conceptual Utilizado en el Sistema

3.1. Representación del Conocimiento

Previamente definimos al conocimiento como el entendimiento de la naturaleza, cualidades y relaciones de las cosas.

Podemos pensar en el conocimiento como una red de conceptos vinculados entre sí. De esta manera se puede representar conocimiento mediante grafos orientados.

Este principio de representación es el utilizado en las Redes Semánticas desarrolladas en el área de la Inteligencia Artificial [Charniak, 1976].

Las redes semánticas simples permiten representar conjuntos de predicados binarios (que relacionan dos conceptos) tales como:

"Juan es humano"

"María es humano"

"Juan tiene 23 años de edad"

"María posee a Ramsés"

"Ramsés es un gato"

Una Red Semántica es un gráfico compuesto de nodos y de arcos "orientados", donde los nodos representan conceptos de un dominio de conocimiento y los arcos orientados permiten representar las vinculaciones existentes entre los distintos conceptos. Cada nodo y cada arco tienen un significado por lo que la red se dice que es "semántica".

En los predicados anteriores encontramos vinculaciones tales como "agente", "destinatario", "acción" y "tema". Estas vinculaciones explicitan el "rol" que cada uno de los conceptos (Juan, libro, María, etc.) juegan en el evento descripto.

El primero de los predicados anteriores define al concepto artificialmente introducido como del tipo "evento".

Dejamos a cargo del lector la representación gráfica en forma de red de la conjunción de predicados anterior.

3.2. Modelo de Datos Conceptual Semántico

El Modelo de Datos utilizado en DBAID, es un Modelo de Datos Conceptual Semántico organizado como una red orientada en la cual los nodos representan Conceptos y los arcos orientados representan Vinculaciones entre conceptos.

3.2.1. Estructura de Datos

La estructura de datos de nuestro Modelo Conceptual está organizada como una red compuesta de:

- Conceptos genéricos.
- Vinculaciones entre conceptos genéricos.

3.2.1.1. Conceptos Genéricos

Los conceptos genéricos son abstracciones que representan a conjuntos de elementos del ambiente de la aplicación que presentan las mismas características.

Por ejemplo el concepto genérico EMPLEADO es una abstracción que representa al conjunto de personas concretas (conceptos individuales, particulares o específicos) empleadas en una organización.

Ejemplos de conceptos genéricos son:

HOSPITAL, CURSO, DOCTOR, REGION, VENDEDOR

La metodología requiere que los conceptos genéricos se denominen por un nombre genérico en singular.

Cuando el sistema detecta la presencia de un nombre en plural lo convierte automáticamente al singular. Por ejemplo, EMPLEADOS se convierte en EMPLEADO, SECRETARIAS en SECRETARIA, CIUDADES en CIUDAD, etc.

3.2.2. Restricciones

Nuestro Modelo Conceptual define restricciones bajo la forma de:

- * Categorías Semánticas para los Conceptos que conforman un conjunto de dominios a los que deben pertenecer los conceptos genéricos. Así, por ejemplo, el concepto genérico EMPLEADO pertenece a la categoría PERSONA.
- * Un conjunto predefinido de Vinculaciones Primitivas para conectar los conceptos genéricos.

Existen además restricciones semánticas sobre las estructuras modelizadas, expresadas como reglas lógicas, las cuales son aplicadas por el sistema para asegurar el correcto diseño de la base de datos (ver "5.2. Funciones de Verificación" y "5.2.1. Verificaciones Semánticas").

3.2.3. Operaciones

El Modelo define operaciones de modelización mediante las cuales se construye el modelo conceptual. Las mismas están implementadas en el sistema.

Como veremos, el Modelo de Datos así definido presenta una gran potencia representacional y semántica, siendo su arquitectura muy simple.

3.3. Categorías Semánticas para los Conceptos

En la sección "3.14. Elección de las Categorías Semánticas" se mencionan antecedentes históricos, se analizan alternativas y se presentan los criterios que se utilizaron para la elección de las categorías utilizadas en el sistema.

Para la categorización de los Conceptos Genéricos se han definido las siguientes categorías:

categoría	significado	Notas
EVT	EVENTO	[1]
PERS	PERSONA	
ORG	ORGANIZACION	
OBJF	OBJETO FISICO	
DISP	DISPOSITIVO	
LUG	LUGAR	
PTIM	TIEMPO (Punto de tiempo)	
FECH	FECHA (caso particular de PTIM)	
CTIM	CANTIDAD DE TIEMPO (Intervalo)	[3]
CAT	CATEGORIA	[2]
IMPO	IMPORTE	
PRC	PRECIO	
CANT	CANTIDAD (diferente de IMPO y CTIM)	[3]
RELA	RATIO (cociente)	[3]
ID	IDENTIFICADOR (código de identificación sin significado y sin estructura)	
NOMB	NOMBRE	
TEXT	TEXTO	

Notas:

1. Un EVENTO es un concepto complejo. Es posible definir un evento como "Algo que sucede" o "Algo que se realiza" en determinadas circunstancias de tiempo, lugar, etc. "a", o "por", determinadas personas, organizaciones, medios, etc. Más adelante definiremos un evento con más precisión.
2. Denota en sí un esquema de clasificación arbitrario. Esquemas tales como: Estado Civil, Sexo, Color, Tipo de Documento, etc.
3. Para los conceptos categorizados como CANT (Cantidad) , CTIM (Cantidad de tiempo) y RELA (Ratio) el sistema solicita al diseñador que especifique la Unidad de Medida correspondiente. Esta especificación es libre, es manejada arbitrariamente por el diseñador, y sólo sirve para fines documentacionales.

3.3.1. Ejemplos de Categorización de Conceptos

A continuación mostraremos algunos ejemplos de categorización de conceptos con la idea de proveer una idea intuitiva del alcance y aplicación de cada Categoría.

Normalmente no existe dificultad en asignar un concepto a una categoría. Sin embargo algunas veces se dan situaciones de ambigüedad en que a un concepto le podría corresponder más de una categoría, y el sistema, como veremos en la sección "3.14. Elección de las Categorías Semánticas", admite sólo una categorización para cada concepto.

Existe un "Principio de Relativismo" que expresa que un concepto puede tener diferente significado para diferentes observadores.

Así, por ejemplo, un barco puede ser visto como un objeto físico, un dispositivo, un lugar, una categoría, etc. por diferentes observadores.

Cada observador, en general, asigna a un objeto un significado relacionado con la función que dicho objeto cumple para él.

En nuestra metodología utilizaremos el criterio de categorizar los conceptos desde la perspectiva más global o general posible, tratando de ver el ambiente de aplicación en su totalidad.

A continuación se incluye un listado de ejemplos de categorización:

DENOMINACION	SIGNIFICADO	EJEMPLOS
EVT	EVENTO	Contratación Curso Inscripción Término Proyecto Suscripción Competencia Visita
PERS	PERSONA	Piloto Empleado Doctor
ORG	ORGANIZACION	Departamento Banco Hospital Proveedor
OBJF	OBJETO FISICO	Producto Combustible

DISP	DISPOSITIVO	Barco Avión Computador Modelo Programa (software) Lenguaje Archivo Relación Plano Contenedor Motor
LUG	LUGAR	Ciudad País Domicilio Provincia Puerto
PTIM	TIEMPO	Mes Año Hora
FECH	FECHA	FechaDeNacimiento FechaDeIngreso
CTIM	CANTIDAD DE TIEMPO	DuraciónDelViaje DuraciónDelProyecto DuraciónDelCurso
CAT	CATEGORIA	* Sistema de categorización: Dominio Color Sexo EstadoCivil TipoDeDocumento (de identidad por ejemplo) TipoDeCuentaBancaria * Posiciones (vertical, horizontal, etc.) * Estados (sólido, líquido, etc.)
IMPO	IMPORTE	* Sueldos
PRC	PRECIO	* Precios

CANT	CANTIDAD	CantidadSolicitada * Dimensiones tales como: Longitud Altura Peso Superficie Volumen etc.
RELA	RATIO	* Porcentajes * Tasas de interés * Velocidades
ID	IDENTIFICADOR	* Códigos: IdDeDepartamento IdDeVuelo NroDocIdentidad NroCuentaBancaria CódigoPostal * Ordinales: NroDeHabitación NumeroDeCalle NumeroDePiso IdentificadorDel Procedimiento
NOMB	NOMBRE	NombreDeEmpleado NombreDeLaCiudad
TEXT	TEXTO	DescripciónDelProducto DescripciónDelProyecto DescripciónDelProceso

3.4. Vinculaciones Primitivas entre Conceptos

Mencionamos previamente que nuestro Modelo Conceptual define restricciones para conectar los conceptos genéricos que componen un modelo conceptual. Dos conceptos genéricos sólo se pueden conectar mediante una vinculación perteneciente a un conjunto predefinido de Vinculaciones Primitivas cuyo significado es conocido por el sistema.

El siguiente cuadro muestra el conjunto de Vinculaciones Primitivas disponibles:

primitiva	significado	Cardinalidad
		[1]
GENERALIZ	Generalización	[1:1]
AGREGAC	Agregación	[1:N]
PROPIEDAD	Propiedad	[1:1]
RECURSIVO	Vinculación consigo mismo [3]	[1:1],[1:N] o [M:N]
Para las circunstancias de EVENTOS [2]		
AGENTE	Agente	[1:N]
COAGENTE	Coagente	[1:N]
DESTINAT	Destinatario	[1:N]
TIEMPO	Tiempo de realización	[1:N]
D_TIEMPO	Tiempo de comienzo	[1:N]
A_TIEMPO	Tiempo de finalización	[1:N]
LUGAR	Lugar de realización	[1:N]
D_LUGAR	Lugar de comienzo	[1:N]
A_LUGAR	Lugar de finalización	[1:N]
ESTADO	Estado	[1:N]
D_ESTADO	Estado de comienzo	[1:N]
A_ESTADO	Estado de finalización	[1:N]
TEMA	Tema	[1:N]
INSTRUM	Instrumento	[1:N]
MODO	Modo de realización	[1:N]
ORIGEN	Origen	[1:N]
DESTINO	Destino	[1:N]

Notas: [1] Se define a continuación (sección 3.4.1.).

[2] Se analiza en la sección 3.7.4.

[3] Se analiza en la sección 3.9.

3.4.1. Cardinalidad de la Vinculación entre dos Conceptos

Al comenzar este capítulo diferenciamos entre conceptos genéricos y conceptos individuales y establecimos que cada concepto genérico representa a un conjunto de conceptos individuales.

Hemos adoptado para denotar a los conceptos genéricos su denominación en mayúsculas. Así el término EMPLEADO denota al conjunto de las personas individuales empleadas en una organización.

Utilizaremos el término <EMPLEADO> para referirnos a un empleado individual cualquiera del conjunto representado por el término EMPLEADO.

Nuestro modelo conceptual semántico reconoce conceptos genéricos y vinculaciones entre conceptos genéricos. Dichas vinculaciones en el plano genérico representan al conjunto de vinculaciones individuales subyacentes (entre conceptos individuales).

La idea de cardinalidad tiene que ver con las vinculaciones entre conceptos individuales. Estas vinculaciones pueden ser:

- "uno a uno", que denotaremos como [1:1].
- "uno a varios", que denotaremos como [1:N].
- "varios a varios", que denotaremos como [M:N].

Consideremos la situación siguiente en que:

- Un <EMPLEADO> tiene un único <NOMBRE_EMPLEADO>.
- Un <NOMBRE_EMPLEADO> pertenece a un único <EMPLEADO>.

En esta situación la cardinalidad entre EMPLEADO y NOMBRE_EMPLEADO es "uno a uno" [1:1]. Veremos otros ejemplos en las siguientes secciones.

El sistema puede en algunos casos determinar automáticamente la cardinalidad de una Vinculación. Cuando ello no es posible el sistema solicita al diseñador que indique cual es la cardinalidad correspondiente.

3.5. Ejemplos de Modelos Conceptuales muy Simples

El siguiente ejemplo muestra un modelo conceptual muy simple de la descripción de las características de un automóvil:

Categorización de los conceptos:

AUTOMOVIL	:	DISP
MARCA	:	NOMB
COLOR	:	CAT
AÑO	:	PTIM
PRECIO	:	PRC

Modelo Conceptual (estructura):

```
AUTOMOVIL  --PROPIEDAD--->  MARCA
AUTOMOVIL  --PROPIEDAD--->  COLOR
AUTOMOVIL  --PROPIEDAD--->  AÑO
AUTOMOVIL  --PROPIEDAD--->  PRECIO
```

Supongamos ahora un ejemplo algo más complicado, que incluye un evento y describe la asignación de un empleado a un departamento:

Categorización de los conceptos:

```
EMPLEADO           : PERS
DEPARTAMENTO        : ORG
ASIGNACION          : EVT
FECHA_ASIGNACION   : FECH
```

Estructura:

```
ASIGNACION  ---DESTINATARIO--->  EMPLEADO
ASIGNACION  ---LUGAR----->     DEPARTAMENTO
ASIGNACION  ---D_TIEMPO----->   FECHA_ASIGNACION
```

La siguiente es una variante notacional que utilizaremos en los ejemplos subsiguientes:

```
ASIGNACION [EVT ]  --DESTINATARIO-->  [PERS] EMPLEADO
ASIGNACION [EVT ]  --LUGAR----->    [ORG ] DEPARTAMENTO
ASIGNACION [EVT ]  --D_TIEMPO----->  [FECH] FECHA_ASIGNACION
```

El siguiente ejemplo describe la visita de un médico a un paciente en un determinado turno:

```
VISITA [EVT ]  ---AGENTE----->    [PERS] DOCTOR
VISITA [EVT ]  ---DESTINATARIO-->   [PERS] PACIENTE
VISITA [EVT ]  ---TIEMPO----->    [EVT ] TURNO
```

3.6. Vinculaciones Lingüísticas

Hemos mencionado previamente que el sistema provee interfaces de comunicación en lenguaje casi-natural.

Durante la modelización conceptual el sistema permite especificar las vinculaciones entre conceptos utilizando elementos del lenguaje natural castellano.

El diseñador no especifica las vinculaciones utilizando las primitivas mostradas en el cuadro de la sección "3.4. Vinculaciones primitivas entre conceptos", sino que utiliza las siguientes "Vinculaciones Lingüísticas":

-----<VINCULACION>----->
LINGUISTICA
+-----+
A
+-----+
CON
+-----+
DE
+-----+
DESDE
+-----+
EN
+-----+
ES
+-----+
HASTA
+-----+
PARA
+-----+
POR
+-----+
SEGUN
+-----+

A partir de la vinculación lingüística utilizada por el diseñador el sistema infiere la correspondiente vinculación primitiva. En caso de ambigüedad presenta al diseñador las vinculaciones primitivas alternativas para que éste indique cuál es la correcta a utilizar.

En determinados casos el sistema requiere que el diseñador especifique la cardinalidad de la vinculación para poder eliminar ambigüedades.

Así, por ejemplo, de las siguientes definiciones:

```

VISITA [EVT ] ---DE-----> [PERS] DOCTOR
VISITA [EVT ] ---A-----> [PERS] PACIENTE
VISITA [EVT ] ---EN-----> [EVT ] TURNO

```

El sistema infiere sin ambigüedades que en el evento "VISITA":

DOCTOR	es el	AGENTE	(quien realiza la visita)
PACIENTE	es el	DESTINATARIO	(quien recibe la visita)
TURNO	es el	TIEMPO	(momento de la visita)

e incorpora al modelo las siguientes definiciones formales:

VISITA	[EVT]	---AGENTE----->	[PERS] DOCTOR
VISITA	[EVT]	---DESTINATARIO-->	[PERS] PACIENTE
VISITA	[EVT]	---TIEMPO----->	[EVT] TURNO

En el ejemplo siguiente, que fue introducido previamente, el sistema requiere que el diseñador especifique la cardinalidad de las vinculaciones:

AUTOMOVIL	---CON---	MARCA
	[1:1]	
AUTOMOVIL	---CON---	COLOR
	[1:1]	
AUTOMOVIL	---CON---	AÑO
	[1:1]	
AUTOMOVIL	---CON---	PRECIO
	[1:1]	

Los siguientes ejemplos son de comprensión intuitiva:

SECRETARIA	[PERS]	---ES---	[PERS] EMPLEADO
VELOCIDAD_			
TIPEO	[RELA]	---DE---	[PERS] SECRETARIA
		[1:1]	
INGENIERO	[PERS]	---ES---	[PERS] EMPLEADO
ESPECIALIDAD	[CAT]	---DE---	[PERS] INGENIERO
		[1:1]	
NOMBRE	[NOMB]	---DE---	[PERS] EMPLEADO
		[1:1]	
DOMICILIO	[LUG]	---DE---	[PERS] EMPLEADO
		[1:1]	
SUELDO	[IMPO]	---DE---	[PERS] EMPLEADO
		[1:1]	

```

ORDEN_COMPRA[EVT ] ---A-----> [ORG ] PROVEEDOR
ORDEN_COMPRA[EVT ] ---EN-----> [FECH] FECHA_EMISION
ITEM_OC      [EVT ] ---DE-----> [EVT ] ORDEN_COMPRA
              [N:1]
ITEM_OC      [EVT ] ---DE-----> [ID  ] MATERIAL
              [1:1]
ITEM_OC      [EVT ] ---CON-----> [TEXT] DESCRIPCION
              [1:1]
ITEM_OC      [EVT ] ---POR-----> [CANT] CANTIDAD
              [1:1]
ITEM_OC      [EVT ] ---POR-----> [PRC ] PRECIO
              [1:1]
ITEM_OC      [EVT ] ---PARA----> [FECH] FECHA_ENTREGA
              [1:1]
ITEM_OC      [EVT ] ---SEGUN--> [ID  ] NORMA_NRO
              [1:1]

```

3.7. Análisis de las Vinculaciones Primitivas

3.7.1. Generalización

Consideremos el siguiente ejemplo, que fue introducido en una sección anterior:

```

SECRETARIA [PERS] ---ES---> [PERS] EMPLEADO
VELOCIDAD_
  TIPO     [RELA] ---DE---> [PERS] SECRETARIA
              [1:1]
INGENIERO [PERS] ---ES---> [PERS] EMPLEADO
ESPECIALIDAD[CAT ] ---DE---> [PERS] INGENIERO
              [1:1]
NOMBRE    [NOMB] ---DE---> [PERS] EMPLEADO
              [1:1]
DOMICILIO [LUG ] ---DE---> [PERS] EMPLEADO
              [1:1]
SUELDO    [IMPO] ---DE---> [PERS] EMPLEADO
              [1:1]

```

Este ejemplo, describe que:

* EMPLEADO es una "Generalización" de SECRETARIA y de INGENIERO (o que SECRETARIA e INGENIERO constituyen "especializaciones" de EMPLEADO).

La vinculación ---ES----> expresa que la vinculación es una vinculación de generalización.

* Que una SECRETARIA "posee" una VELOCIDAD_TIPEO, y que un INGENIERO "posee" una ESPECIALIDAD.

* Que un EMPLEADO "posee" un NOMBRE, un DOMICILIO y un SUELDO, los cuales son aplicables tanto a SECRETARIA como a INGENIERO. Es decir que los conceptos SECRETARIA e INGENIERO, "heredan" las propiedades o atributos del concepto EMPLEADO.

Una vinculación de Generalización establece una "taxonomía jerárquica de generalización" entre un concepto más específico (ej: SECRETARIA) y un concepto más general (ej: EMPLEADO) siendo la cardinalidad de la vinculación "uno a uno" [1:1].

Las propiedades y las vinculaciones del concepto más general son heredadas por el concepto más específico (ej: Nombre, Domicilio, Sexo, Fecha de Nacimiento, etc.).

IMPORTANTE: De las definiciones anteriores surge que las dos generalizaciones mostradas son independientes entre sí, aunque conceptualmente ambas pertenecen a un mismo "grupo de generalización".

3.7.1.1. La Generalización vista desde el Modelo Relacional

Consideremos el ejemplo anterior. Desde la perspectiva del Modelo Relacional podemos imaginar la situación del ejemplo, como un esquema compuesto por tres Relaciones: EMPLEADOS, SECRETARIAS e INGENIEROS.

SECRETARIAS		EMPLEADOS			
secretaria	velocidad_	empleado	nombre	domicilio	sueldo
#	tipeo	#			
[1]					

INGENIEROS	
ingeniero	especialidad
#	
[1]	

En este ejemplo podemos observar que la relación EMPLEADO puede considerarse una extensión virtual de las relaciones SECRETARIA e INGENIERO, y viceversa.

Nota [1]:

A cada tupla de las Relaciones SECRETARIA e INGENIERO le corresponde una tupla en la relación EMPLEADO, por lo que es posible utilizar el mismo valor para identificarlas (ver "6.4. Consideraciones sobre la identificación unívoca de tuplas").

El sistema por consiguiente cambia los atributos de identificación [1] por "empleado#".

3.7.2. Agregación

Una vinculación de agregación establece una vinculación con cardinalidad "uno a varios", o [1:N] entre dos conceptos.

La vinculación de Agregación establece una jerarquía en la que el concepto de mayor nivel tiene como "agregado" un concepto de menor nivel.

Por ejemplo, la siguiente definición:

```
TELEFONO      [DISP]  ---DE--->  [PERS] EMPLEADO
                               [N:1]
```

que expresa que:

- * Un <EMPLEADO> tiene (o puede tener) varios <TELEFONO> y
- * Un <TELEFONO> pertenece sólo a un <EMPLEADO>.

establece una vinculación "uno a varios" [1:N] entre ambos conceptos, es decir establece una Vinculación de Agregación (TELEFONO es un agregado de EMPLEADO).

Dos eventos también pueden estar vinculados por una vinculación de Agregación. El siguiente ejemplo, que fuera introducido en una sección anterior, ilustra esta situación (ver vinculación [2]).

```
ORDEN_COMPRA[EVT ]  ---A----->  [ORG ] PROVEEDOR
ORDEN_COMPRA[EVT ]  ---EN----->  [FECH] FECHA_EMISION
ITEM_OC      [EVT ]  ---DE----->  [EVT ] ORDEN_COMPRA [2]
                               [N:1]
ITEM_OC      [EVT ]  ---DE----->  [ID  ] MATERIAL
                               [1:1]
ITEM_OC      [EVT ]  ---CON----->  [TEXT] DESCRIPCION
                               [1:1]
ITEM_OC      [EVT ]  ---POR----->  [CANT] CANTIDAD
                               [1:1]
ITEM_OC      [EVT ]  ---POR----->  [PRC ] PRECIO
                               [1:1]
ITEM_OC      [EVT ]  ---PARA---->  [FECH] FECHA_ENTREGA
                               [1:1]
ITEM_OC      [EVT ]  ---SEGUN-->  [ID  ] NORMA_NRO
                               [1:1]
```

En la vinculación [2]:

- * Una <ORDEN_COMPRA> tiene (o puede tener) varios <ITEM_OC> y
- * Un <ITEM_OC> pertenece sólo a una <ORDEN_COMPRA>.

3.7.3. Propiedad

Una vinculación de Propiedad establece una vinculación con cardinalidad "uno a uno" [1:1] entre dos conceptos (un concepto que tiene una propiedad con el concepto que denota dicha propiedad).

Por ejemplo, la siguiente definición:

```
NOMBRE_EMPLEADO [NOMB] ---DE----> [PERS] EMPLEADO
[ 1 : 1 ]
```

establece una Vinculación de Propiedad ya que:

- * Un <EMPLEADO> tiene un <NOMBRE_EMPLEADO>.... y...
- * Un <NOMBRE_EMPLEADO> pertenece a un <EMPLEADO>.

3.7.4. Circunstancias de los Eventos

Hemos definido previamente a los EVENTOS como conceptos complejos, que representan "Algo que sucede" o "Algo que se realiza" en determinadas circunstancias de tiempo, lugar, etc., "a", o "por", determinadas personas, organizaciones, medios, etc..

Los conceptos categorizados como Eventos son conceptos complejos, que representan la asociación de varios otros conceptos cada uno de los cuales desempeña un "ROL" en el mismo.

Invirtiendo la perspectiva podemos decir que UN EVENTO SE REALIZA EN LAS CIRCUNSTANCIAS DEFINIDAS POR LOS CONCEPTOS ASOCIADOS.

En general, un evento representa la ocurrencia de un suceso en el mundo real que pretendemos modelizar. Este suceso ocurre en determinado momento o período de tiempo, por lo que, salvo excepciones que veremos a continuación, todo Concepto categorizado como Evento debería estar vinculado, en el modelo conceptual, con Conceptos categorizados como "PTIM" o "FECH", por vinculaciones lingüísticas del tipo "EN", "POR", "DESDE" o "HASTA".

La capacidad semántica del sistema le permite detectar, y notificar al diseñador, la ausencia de estas vinculaciones.

Como mencionáramos previamente existen también excepciones constituidas por eventos que denotan la existencia de asociaciones aparentemente permanentes, atemporales, o para los que interesa sólo el momento actual (ver "5.4.2. El Tiempo y la representación de objetos en la Base de Datos") y por consiguiente no presentan vinculaciones temporales. Este es el caso que se da, por ejemplo, cuando se desea representar las fronteras actuales entre países, el matrimonio actual entre personas, etc. (ver "3.9. Concepto vinculado consigo mismo").

3.8. Modelización de Vinculaciones con Cardinalidad [M:N]

En determinadas situaciones de modelización pueden aparecer conceptos vinculados con vinculaciones con cardinalidad "varios a varios" [M:N]. Por ejemplo, la siguiente definición:

```
TELEFONO [DISP] ---DE----> [PERS] EMPLEADO
[ M : N ]
```

expresa que:

- Un <EMPLEADO> tiene (o puede tener) varios <TELEFONO> y
- Un <TELEFONO> pertenece (o puede pertenecer) a varios <EMPLEADO>.

Para la modelización de vinculaciones con cardinalidad [M:N] es necesario introducir "eventos de asociación", por lo que cuando el diseñador especifica una vinculación "varios a varios" entre dos conceptos el sistema propone automáticamente la inclusión de un evento de asociación, ofreciendo para el mismo una denominación por defecto.

Si en el caso del ejemplo anterior el diseñador adopta como denominación para el evento de asociación "ASIGN_TEL" ("asignación de teléfono"), la vinculación [M:N] queda modelizada finalmente como:

```
ASIGN_TEL [EVT ] ---A----> EMPLEADO [PERS]
ASIGN_TEL [EVT ] ---DE----> TELEFONO [DISP]
```

3.9. Concepto Vinculado Consigo Mismo

En algunas situaciones de modelización un determinado concepto puede estar vinculado consigo mismo. Situaciones de esta naturaleza se dan cuando se desea modelizar:

- Estructuras de productos.
- Jerarquías Organizacionales.
- Fronteras entre Países.
- Matrimonios.

Cuando el diseñador especifica una vinculación de un concepto consigo mismo el sistema propone automáticamente la inclusión de un evento de asociación o de un concepto de asociación, según sea la cardinalidad de dicha vinculación.

Para cardinalidad:	se requiere:
[M:N]	Evento de asociación
[1:N]	Concepto de asociación
[1:1]	Concepto de asociación

3.9.1. Cuando la Cardinalidad es [M:N]

Ejemplos:

Estructuras de Producto

MATERIAL [OBJF] ---EN---> [OBJF] MATERIAL
[M : N]

Fronteras entre Países

PAIS [LUG] ---CON---> [LUG] PAIS
[M : N]

Acción:

El sistema propone la introducción de un evento de asociación, solicitando un nombre para el mismo (ofrece un nombre por defecto).

Por ejemplo, para el caso de Estructuras de Producto la situación final (suponiendo que el diseñador indica ESTRUCTURA como nombre para el evento de asociación) es:

```
ESTRUCTURA [EVT ] ----DE----> [OBJF] MATERIAL
ESTRUCTURA [EVT ] ----CON----> [OBJF] MATERIAL
```

3.9.2. Cuando la Cardinalidad es [1:N]

Ejemplo:

Organigrama Jerárquico

```
+-----+
| EMPLEADO [PERS] ---DE----> [PERS] EMPLEADO |
|                               [ N : 1 ]       |
+-----+
```

Acción:

El sistema propone la introducción de un Concepto de Asociación vinculado al concepto "EMPLEADO" y solicita una denominación para el concepto de asociación. Si el diseñador especifica ejemplo JEFE_DEL_EMPLEADO el modelo queda:

```
+-----+
| JEFE_DEL_EMPLEADO [ID ] ---DE----> [PERS] EMPLEADO |
|                               [ 1 : N ]       |
+-----+
```

3.9.3. Cuando la Cardinalidad es [1:1]

Ejemplo:

Matrimonio

```

+-----+
| PERSONA [PERS] ---CON---> [PERS] PERSONA |
|                                     [ 1 : 1 ] |
+-----+

```

Acción:

El sistema propone la introducción de un Concepto de Asociación para el concepto "PERSONA". Solicita una denominación para el concepto de asociación. Si el diseñador especifica por ejemplo CONYUGE el modelo queda:

```

+-----+
| CONYUGE [ID ] ---DE----> [PERS] PERSONA |
|                                     [ 1 : 1 ] |
+-----+

```

3.10. Precedencia de Conceptos

Las vinculaciones entre conceptos genéricos llevan asociada la idea de precedencia. Retomando el ejemplo que describe la visita de un médico a un paciente en un determinado turno:

```

VISITA [EVT ] ---DE-----> [PERS] DOCTOR
VISITA [EVT ] ---A-----> [PERS] PACIENTE
VISITA [EVT ] ---EN-----> [EVT ] TURNO

```

En este ejemplo podemos ver que una Visita no puede realizarse si no existen Doctores, Pacientes y Turnos. Desde esta perspectiva los conceptos genéricos DOCTOR, PACIENTE y TURNO son conceptos precedentes del concepto genérico VISITA (las mismas consideraciones resultan especialmente válidas a nivel de conceptos individuales).

El sentido de la vinculación entre dos conceptos especifica el orden de precedencia de los mismos.

En las representaciones gráficas provistas por el sistema los conceptos con mayor precedencia se grafican más arriba que los de menor precedencia. Así para este ejemplo tenemos:

```

DOCTOR      PACIENTE      TURNO
  |          |            |
  |          |            |
  |          |            |
+-----VISITA-----+

```


El sistema tiene la capacidad de identificar potenciales precedencias entre eventos que no fueron vinculados explícitamente por el diseñador y de proporcionar información para una eventual explicitación.

3.12. Eventos Implícitos

Consideremos el siguiente ejemplo:

```
[1] IMPORTE_
    VENDIDO      [IMPO] ---POR---> [PERS] VENDEDOR
                        [1:1]

[1] IMPORTE_
    VENDIDO      [IMPO] ---EN----> [LUG ] ZONA
                        [1:1]

[1] IMPORTE_
    VENDIDO      [IMPO] ---EN----> [PTIM] MES
                        [1:1]
    VENDEDOR     [PERS] ---ES----> [PERS] EMPLEADO
    ZONA         [LUG ] ---EN----> [LUG ] REGION
                        [N:1]
```

Las vinculaciones [1] consideradas en conjunto están definiendo un evento implícito que asocia a VENDEDOR, ZONA y MES (IMPORTE_VENDIDO es un dato de la asociación de VENDEDOR, ZONA y MES). En cambio, consideradas separadamente tienen un significado completamente distinto ya que estarían denotando la presencia de conceptos "homónimos", es decir de conceptos diferentes con igual denominación.

IMPORTANTE: En nuestro Modelo Conceptual diferentes conceptos (que representan a diferentes cosas y por consiguiente tienen diferentes significados) deben tener diferentes denominaciones.

Cuando el sistema detecta mediante sus funciones de verificación una situación de este tipo solicita al diseñador la explicitación del evento, solicitándole una denominación para el mismo, proponiéndole una denominación por defecto compuesta por la concatenación de las tres primeras letras del nombre de los conceptos asociados (en este caso: VEN_ZON_MES).

Las vinculaciones [1] se transforman en:

```
VEN_ZON_MES [EVT ] ---POR---> [PERS] VENDEDOR
VEN_ZON_MES [EVT ] ---EN----> [LUG ] ZONA
VEN_ZON_MES [EVT ] ---EN----> [PTIM] MES
IMPORTE_
    VENDIDO     [IMPO] ---DE----> [EVT ] VEN_ZON_MES
                        [1:1]
```

3.13. Grupos

En la sección "3.7.1. Generalización" introdujimos la idea de "grupo de generalización". La idea de Grupo es también aplicable a las vinculaciones de Agregación y de Propiedad.

Por ejemplo, los conceptos DIA, MES y AÑO conforman un grupo asociado con FECHA, y los conceptos CALLE, NUMERO, PISO, DEPARTAMENTO, LOCALIDAD, PROVINCIA conforman un grupo asociado con DOMICILIO.

Actualmente el sistema no provee mecanismos para la representación y el tratamiento de Grupos.

3.14. Elección de las Categorías Semánticas

Trataremos en esta sección el tema de la elección de las categorías semánticas, recordaremos algunos antecedentes históricos importantes, examinaremos las diferentes alternativas de esquemas de categorización y presentaremos los criterios que se han utilizado en la elección de las categorías para DBAID.

3.14.1. La Perspectiva de Aristóteles o Nada es Nuevo Bajo el Sol

Hace 2300 años Platón introdujo el concepto de "abstracción" separando "la idea" de un objeto del conjunto de objetos concretos representados por dicha idea.

Separaba, por ejemplo, la idea de "caballo" (a la cual se refería como "universal caballo" o "concepto genérico caballo") de los caballos concretos representados por la idea.

Platón planteó la posibilidad de razonar en términos de ideas abstractas.

Aristóteles, discípulo de Platón, introdujo las ideas de:

- Establecer categorías semánticas para las abstracciones, es decir propuso la idea de realizar abstracciones utilizando un conjunto de categorías preestablecidas.
- Establecer múltiples niveles de abstracción. Por ejemplo la idea de animal puede abstraerse de la idea de caballo.
- Creó la Lógica de Proposiciones (denominada comunmente Lógica Aristotélica).

y planteó asociar las tres tratando de crear una lógica basada en categorías semánticas.

El problema planteado por Aristóteles aún no tiene una solución formal. En esencia su planteo es completamente análogo al que surge al diseñar un sistema de las características de DBAID. La solución en DBAID es una solución ad-hoc que utiliza categorías semánticas, múltiples niveles de abstracción y Lógica de Predicados de Primer Orden.

En el discurso humano, afirmaba Aristóteles, se pueden distinguir diez "categorías fundamentales" (el número 10 era un "número mágico" en aquella época) la primera era la Sustancia (el sujeto), en la cual se centra toda afirmación y las nueve restantes eran los nueve tipos de afirmaciones que se podían hacer de ella (en la lengua griega).

Por ejemplo sobre Sócrates (la Sustancia) se podía decir su Cantidad (mide 7 palmos), su Cualidad (es filósofo), que está en un Tiempo (al mediodía) y en un Lugar (en el ágora), cual es su Acción (dialoga) y su Pasión (es juzgado), cual es su Relación (es maestro de Platón), y en que Estado (pobre) y Posición se halla (de pié).

El sistema de categorías de Aristóteles está orientado a describir las circunstancias que rodean a un sujeto y fue determinado por la gramática de la lengua griega.

La Lógica Aristotélica y su Análisis de las Categorías reinó hasta el siglo XIX, y hoy en día se dice que Aristóteles dió el paso definitivo en el desarrollo de los procesos de abstracción.

3.14.2. Desde la Perspectiva de la Lingüística

Como dijéramos al comenzar el libro, el lenguaje natural es nuestra forma primaria de representación y comunicación de la información, siendo la Lingüística la disciplina que lo estudia formalmente.

Una de las ramas de la Lingüística es la Semántica, la cual estudia el significado de las palabras y de las oraciones.

En el campo de la Semántica Lingüística se han propuesto diferentes sistemas para asignar significado a las palabras. Entre los más utilizados están los sistemas de "composición de rasgos".

Por ejemplo el significado de la palabra "mujer" se puede definir conceptualmente mediante la composición de varios "rasgos contrastantes" tales como:

- + HUMANO
- MASCULINO
- + ADULTO

En este enfoque se busca que los rasgos contrastantes (categorías semánticas según nuestra nomenclatura) sean lo más universales y estables posible. Los rasgos utilizados en la tipificación anterior son intuitivamente más universales que los siguientes (siempre para definir el significado de la palabra "mujer"):

- + CONVERSADOR
- + EXPERTO EN COCINA
- + USA FALDAS

C.J.Fillmore en 1968 [Fillmore, 1968] publica un trabajo titulado "The Case for Case" en el cual analiza el significado profundo de las oraciones estudiando las circunstancias en que se realizan las acciones y los roles que desempeñan las cosas intervinientes.

Fillmore identifica entre otros las siguientes circunstancias y roles universales: Tiempo, Lugar, Agente, Instrumento, Beneficiario, Origen, Destino, Tema y Objeto.

3.14.3. Desde la Perspectiva de la Inteligencia Artificial

Estas ideas desarrolladas en el campo de la Lingüística son utilizadas por la Lingüística Computacional y la Semántica Computacional, las cuales son ramas de la Inteligencia Artificial.

En la década de los 60 cobra impulso el desarrollo de la Inteligencia Artificial, siendo algunas de sus principales preocupaciones:

- La representación formal del conocimiento.
- La semántica computacional (como lograr que un computador entienda el significado de los datos que procesa).
- El razonamiento automático.

Uno de los sistemas de representación propuesto es el de Red Semántica, el cual según vimos es el sistema de representación utilizado en DBAID.

En el área de la Semántica Computacional se utilizan sistemas de tipificación semántica de dos tipos:

- De composición de rasgos, en el cual cada concepto se define en términos de múltiples rasgos o categorías.
- De asignación a una categoría, donde cada concepto se define como perteneciente a una única categoría.

En ambos sistemas las categorías pueden conformar una estructura (jerárquica o en red) con múltiples niveles, si bien normalmente se utilizan en la tipificación de un concepto solo las categorías ubicadas en la base de estas estructuras (categorías "terminales").

Es importante destacar que todo sistema de representación del conocimiento, para resultar de utilidad, debe tener asociado un mecanismo de tratamiento automático, por ejemplo un mecanismo de razonamiento automático que permita inferir nuevo conocimiento a partir del conocimiento disponible, verificar la consistencia del conocimiento disponible o responder interrogantes utilizando dicho conocimiento.

La mayoría de los sistemas de representación utilizan mecanismos de inferencia basados en lógicas "ad hoc" como en el caso de DBAID.

Cuando se introducen categorías semánticas en las inferencias, la complejidad del mecanismo de inferencias crece muy rápidamente a medida que crece la complejidad del sistema semántico.

En general el sistema de representación, el sistema de categorías semánticas y el mecanismo de tratamiento automático asociado están fuertemente interrelacionados e integrados, resultando prácticamente imposible modificar uno de ellos sin modificar los demás.

3.14.4. Criterios Utilizados en el Diseño del Sistema

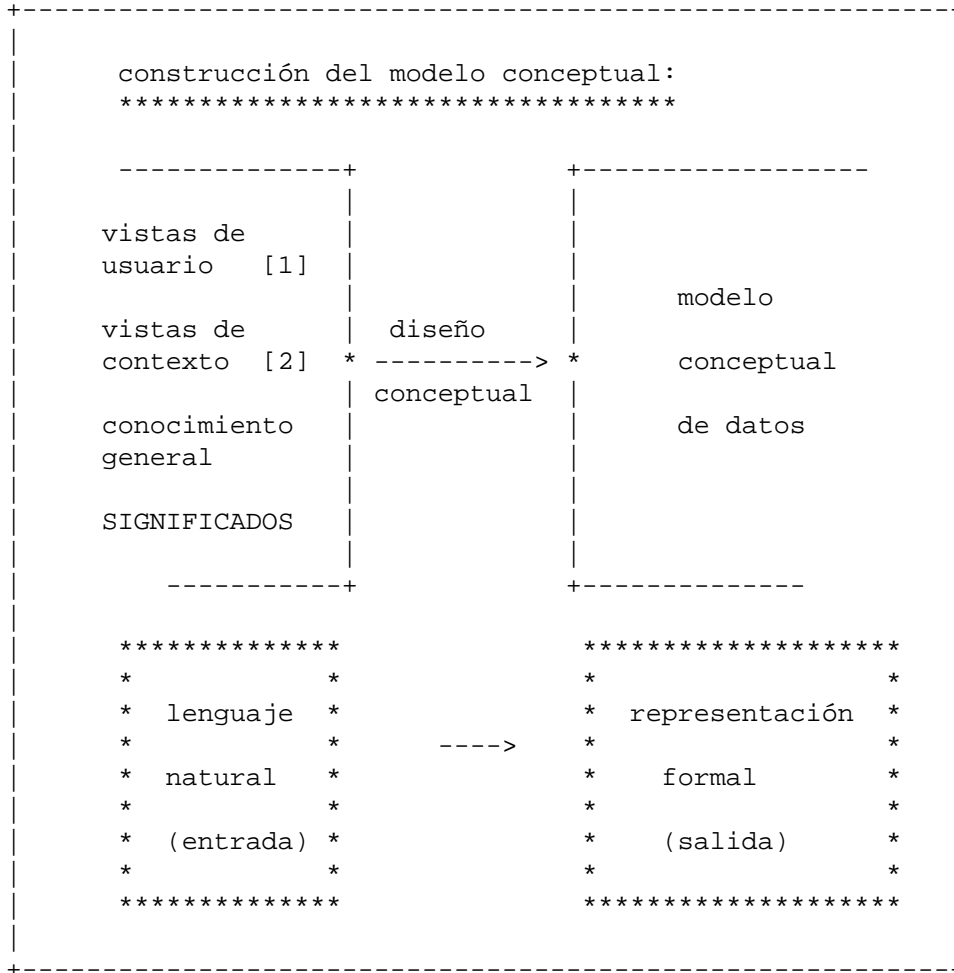
Los criterios utilizados para el diseño del sistema semántico de DBAID fueron los siguientes:

- El conjunto de Categorías Semánticas debe poseer "adecuación expresiva" e "independencia de contexto" (universalidad), es decir, debe tener la capacidad de expresar adecuadamente el significado de los conceptos y sus vinculaciones en múltiples áreas de aplicación.
- Debe promover una "economía notacional", para lo cual el conjunto debe ser lo más reducido posible.
- Debe ser de "comprensión intuitiva" y "uso natural" tanto para el diseñador como para los usuarios que participan en el diseño
- El conjunto debe ser "coherente", las diferentes categorías no deben superponerse entre sí, ni deben quedar huecos que impidiesen determinadas categorizaciones.
- Debe permitir implementar un mecanismo de inferencias semánticas que cubra las necesidades del diseño conceptual semántico y cuyo funcionamiento sea eficiente aún para los modelos muy grandes.

El sistema semántico adoptado por DBAID es de "asignación a una categoría", en el cual cada concepto se define como perteneciente a una única categoría. La estructura de categorías es de un solo nivel. Las vinculaciones entre conceptos se realizan mediante un conjunto de vinculaciones primitivas predefinido.

El mecanismo de inferencias asociado resulta de esta manera simple, satisfaciendo además adecuadamente todos los requerimientos de la metodología de diseño.

El sistema de categorías y vinculaciones primitivas de DBAID, presentado al comienzo de este capítulo, sigue los lineamientos del trabajo de Fillmore en sus aspectos fundamentales. El sistema se ha perfeccionado en base a las experiencias de diseño realizadas con las sucesivas versiones (releases) del sistema.



[1] Requerimientos de información de los usuarios.

[2] Descripciones del ambiente de aplicación necesarias para obtener conocimiento no provisto por las vistas de usuario que permita definir, integrar y correlacionar conceptos abstraídos de las vistas de usuario.

4.2. Vistas

El diseñador obtiene información para comenzar el proceso de modelización de las "vistas de usuario" y las "vistas de contexto", así como de su propio bagaje de "conocimiento general" sobre el ambiente de la aplicación. La información obtenida se expresa mediante oraciones en lenguaje castellano.

4.2.1. Vistas de Usuario

Son descripciones de la información requerida por los usuarios del sistema de información para contestar una pregunta, tomar una decisión, realizar una tarea, y/o proveer información a otras personas o sistemas.

4.2.2. Vistas de Contexto

Son un conjunto de oraciones que describen la empresa, su organización y su modo de operación, complementando la información provista por las "vistas de usuarios", aportando fundamentalmente conocimiento sobre la estructura del ambiente de aplicación, la que frecuentemente no puede abstraerse completamente a partir de las "vistas de usuario" solamente.

Podemos imaginar la estructura de una empresa como un objeto sólido, una escultura por ejemplo, y que cada usuario lo ve parcialmente desde una perspectiva particular (vista de usuario). Suponiendo que cada vista de usuario pueda plasmarse en una fotografía, con una cantidad suficiente de estas fotografías sería posible visualizar el objeto completo.

Cuando la cantidad de fotografías no es suficiente, lo cual ocurre en la mayoría de los casos, resulta necesario recurrir a fotografías adicionales (vistas de contexto).

4.3. Obtención de Vistas

Las vistas pueden obtenerse de una variedad de fuentes como las siguientes:

- entrevistas con los usuarios.
- entrevistas con funcionarios no usuarios.
- conversaciones informales con integrantes de la organización.
- analizando informes en uso emitidos:
 - periódicamente
 - a pedido
 - por excepción
- analizando pantallas de procesamientos interactivos.
- analizando bases de datos ya implementadas.
- analizando registros de archivos convencionales y transferencia de datos entre programas de aplicación ya implementados.

De una fuente cualquiera pueden surgir múltiples vistas.

4.4. Ejemplos de Vistas

Las siguientes vistas corresponden al ambiente de aplicación de las competencias automovilísticas de Fórmula 1 Internacional. La elección de este ejemplo se ha realizado buscando un sistema real relativamente simple y conocido por la mayoría de los lectores, de manera de poder aprovechar al máximo sus "conocimientos generales" relevándonos de detalladas explicaciones.

A continuación se muestran ejemplos de algunas vistas de usuario y de contexto las que supondremos fueron obtenidas en un proceso de observación (relevamiento) realizado previamente.

4.5. Ejemplo de Observación (Obtención de Vistas)

Vistas de Usuario

Vista número	Origen	Descripción
1	Director Comercial	Con qué Patrocinantes cuenta la Escudería "aaa" en la Temporada actual?.
5	Director Deportivo	Cuántos Puntos obtuvo cada Piloto en la Temporada actual?.
9	Director de Mecánica	Qué Chassis fueron utilizados en el Circuito "ccc" por las diferentes Escuderías la Temporada pasada y cuáles fueron los Ordenes de llegada de cada uno?.

Vistas de Contexto

vista número	origen	descripción
1		Se realiza una Temporada de Competencias por Año.
2		Se realiza un número variable de Competencias, en diferentes lugares (Circuitos), cada Temporada.
3		En cada Competencia participa una Cantidad variable de Pilotos. La cantidad depende del Circuito donde se realiza.

4.6. Ejemplos de Abstracción y Formalización

Para cada una de las vistas anteriores, extraeremos a continuación Conceptos y Vinculaciones entre conceptos, los tipificaremos utilizando las Categorías Semánticas del sistema, y especificaremos las definiciones ("microvistas") que suministraremos al sistema.

Realizamos la Abstracción y Formalización (Extracción y Tipificación) de Conceptos y Vinculaciones, respondiendo para cada vista el interrogante: Qué describe la vista?, o bien expresando la vista como oración en la forma objetiva (que se define más abajo):

Así, para la Vista de Usuario 1:

"Con qué Patrocinantes cuenta la Escudería "aaa" en la Temporada actual?".
--

Qué describe esta vista?

La vista que estamos considerando describe:

El patrocinio de patrocinantes
a escuderías
en una temporada.

lo cual se expresa mediante las siguientes definiciones:

```

+-----+
| PATROCINIO [EVT] --DE-----> [ORG] PATROCINANTE |
+-----+
| PATROCINIO [EVT] --A-----> [ORG] ESCUDERIA |
+-----+
| DENOMINACION_ [NOMB] --DE-----> [ORG] ESCUDERIA |
| ESCUDERIA [ 1 : 1 ] |
+-----+
| PATROCINIO [EVT] --EN-----> [PTIM] TEMPORADA |
+-----+

```

El ejemplo anterior nos permite observar que existe correspondencia entre elementos del lenguaje natural y elementos del modelo conceptual. Es posible ver que:

- Los SUSTANTIVOS o nombres genéricos por lo general denotan Conceptos.
- Los VERBOS denotan la presencia de EVENTOS.
- Los NOMBRES PROPIOS y las CANTIDADES denotan Valores de Atributos, lo mismo que los ADJETIVOS (por ejemplo "actual" en el caso anterior).
- Las PREPOSICIONES denotan Vinculaciones entre Conceptos.

Las preposiciones tienen un gran poder informativo. Obsérvese en ese sentido que la mayoría de las Vinculaciones Lingüísticas en DBAID (ver "3.6. Vinculaciones Lingüísticas") coinciden con las preposiciones más utilizadas en el lenguaje diario.

En el lenguaje natural las Preposiciones relacionan las palabras entre las que van colocadas, y sus significados dependen del significado de las palabras que vinculan.

Obsérvese que las vistas de usuario se expresan en general mediante oraciones interrogativas, mientras que las vistas de contexto se expresan mediante oraciones descriptivas o declarativas en la voz activa (el sujeto de la oración es quien ejecuta la acción) y en tercera persona.

Las definiciones de entrada al sistema tienen el carácter de declarativas por lo que antes de realizar la abstracción de conceptos de las vistas de usuario se deben transformar estas vistas de la forma interrogativa a la forma declarativa.

Además, para las vistas que denotan la realización de eventos, tal como la vista del ejemplo anterior, resulta útil transformar la oración de manera que la acción se convierta en el objeto de la oración. La oración así transformada diremos que se encuentra en la forma objetiva.

Para la vista de usuario anterior las transformaciones sucesivas son:

1. A la forma declarativa:

"Patrocinantes Patrocinan a Escuderías en cada Temporada".

2. A la forma objetiva:

"Se realizan Patrocinios de Patrocinantes a Escuderías en cada Temporada".

Consideremos ahora la Vista de Usuario 5:

```
+-----+
| "Cuántos Puntos obtuvo cada Piloto en la Temporada |
| actual?                                             |
+-----+
```

Llevando la vista a la forma declarativa queda:

"Los Pilotos obtienen Puntos en las Temporadas".

lo cual se expresa mediante las siguientes definiciones:

```
+-----+
| PUNTOS_OBTENIDOS [CANT] ---DE-----> [PERS] PILOTO |
|                                     [ N : 1 ]         |
+-----+
| PUNTOS_OBTENIDOS [CANT] ---EN-----> [EVT ] TEMPORADA |
|                                     [ N : 1 ]         |
+-----+
```

Las dos definiciones consideradas en conjunto definen un evento implícito (ver tratamiento en "Eventos Implícitos").

Pasemos ahora a considerar las Vistas de Contexto.

Consideremos ahora la Vista de Contexto 1:

```
+-----+
| Se realiza una Temporada de Competencias por Año. |
+-----+
```

Esta vista ya está expresada en la forma objetiva y genera las siguientes definiciones:

```
+-----+
| TEMPORADA [EVT ] ---DE-----> [EVT ] COMPETENCIA |
| TEMPORADA [EVT ] ---POR-----> [PTIM] AÑO        |
+-----+
```

Consideremos ahora la Vista de Contexto 2:

```
+-----+
| Se realiza un número variable de Competencias, en |
| diferentes lugares (Circuitos), en cada Temporada. |
+-----+
```

Expresando la vista en la forma objetiva tenemos:

"Se realizan Competencias en Circuitos en cada Temporada".

Generándose las siguientes definiciones de entrada al sistema:

```
+-----+
| COMPETENCIA [EVT ] ---EN-----> [LUG ] CIRCUITO |
| COMPETENCIA [EVT ] ---EN-----> [EVT ] TEMPORADA |
+-----+
```

4.7. Interacción con el Sistema

La modelización conceptual se realiza interactuando con el sistema a través de múltiples ventanas y menús. A diferencia de la notación utilizada en este libro para las microvistas, el sistema no requiere repetir la tipificación de cada concepto cada vez que éste se menciona. Un concepto se tipifica semánticamente sólo la primera vez que es mencionado.

4.8. Validación de la Información Provista en las Microvistas

A medida que se interactúa con el sistema durante el proceso de modelización, el diseñador descubre que el sistema realiza determinadas validaciones en tiempo de Input de las microvistas, mientras que otras validaciones son realizadas posteriormente mediante Funciones de Verificación.

Si bien es posible realizar la mayoría de las validaciones en tiempo de entrada, ello puede provocar dos efectos indeseados:

- Una menor fluidez en el ingreso de información debido al tiempo requerido por el sistema para validar la coherencia de una microvista contra la totalidad del modelo construido hasta ese momento.

- Una menor posibilidad de adaptación del modelo a cambios evolutivos en los mecanismos mentales de abstracción del diseñador (aunque el sistema busca objetivizar el proceso de abstracción, debemos mencionar que inicialmente existe una componente subjetiva que va desapareciendo a medida que se avanza en un diseño particular).

Si por el contrario, diferimos la mayoría de las validaciones al momento de Verificación, también se presentan efectos secundarios tales como:

- La posibilidad de una acumulación relativamente alta de errores y de ambigüedades, que para ser corregidas demanden un esfuerzo comparativamente elevado.

La solución al problema de la validación es necesariamente una solución de compromiso. Encontrar la mejor solución de compromiso es una de las tareas de experimentación permanente de los diseñadores de DBAID.

Aún en los casos de validación en tiempo de entrada, determinados juicios deben ser forzosamente diferidos hasta el momento en que estén dados todos los elementos de juicio (microvistas) necesarios para analizar una situación específica.

Una verificación anticipada, basada en aceptar que la información previamente suministrada al sistema es correcta, podría conducir a una situación en que "antiguas verdades, que ya no lo son, impidan la aceptación de nuevas verdades", es decir conducir a una situación que impida la evolución del modelo a lo largo del diseño.

4.9. Ejemplo de Modelización Conceptual

Como ejemplo de un proceso de diseño, plantearemos un caso muy simple pero ilustrativo, al cual nos referiremos de ahora en más como el "Caso Talleres".

Debido a la simplicidad del ejemplo no consideraremos vistas de usuario y de contexto separadamente. El caso es el siguiente:

Una gran empresa de reparación de automóviles, que tiene varios talleres en diferentes lugares, desea implementar una base de datos sobre los mecánicos de cada taller, sus especializaciones y su nivel de capacitación.

Cada taller tiene un gerente y varios mecánicos.

La empresa tiene un programa de capacitación para los mecánicos que consta de varias asignaturas. Para cada asignatura se realizan cursos idénticos en forma periódica. La duración de los cursos depende de la asignatura a la que corresponden.

Las especializaciones de cada mecánico están dadas por los correspondientes cursos de capacitación que ha tomado y su nivel de capacitación por la calificación obtenida en dichos cursos.

Interesa conocer además de los datos personales de los empleados (gerente y mecánicos) la fecha en que cada mecánico tomó sus cursos de capacitación.

Abstracción y Formalización de Conceptos y Vinculaciones:

Como primer paso, informaremos al sistema que "los MECANICOS y los GERENTES son EMPLEADOS":

```
+-----+
| MECANICO      [PERS] ---ES----> [PERS] EMPLEADO |
+-----+
```

Para la vinculación ES (vinculación de Generalización) el sistema verifica que los dos conceptos vinculados pertenezcan a categorías compatibles (en este caso [PERS]).

Cuando se define un concepto como [PERS] el sistema ofrece un Template bajo la forma de menú de opciones múltiples conteniendo un conjunto de atributos tales como Fecha de Nacimiento, Nombre, Domicilio, Sexo, etc., pudiendo el diseñador elegir aquellos conceptos que desea incluir en el modelo.

Suponiendo que seleccionamos Nombre y Domicilio, para EMPLEADO solamente, ya que MECANICO heredará estos atributos, el sistema genera automáticamente las dos microvistas siguientes:

```
+-----+
| NOMBRE_
|   EMPLEADO [NOMB] ---DE----> [PERS] EMPLEADO
|                                     [ 1 : 1 ]
+-----+
| DOMICILIO_
|   EMPLEADO [LUG ] ---DE----> [PERS] EMPLEADO
|                                     [ 1 : 1 ]
+-----+
```

Completamos la información introduciendo entonces la siguiente microvista y no aceptamos los atributos del Template correspondiente a PERS ofrecidos para GERENTE, ya que dicho concepto heredará los atributos asignados previamente a EMPLEADO.

```
+-----+
| GERENTE      [PERS] ---ES----> [PERS] EMPLEADO |
+-----+
```

A continuación informamos al sistema que "cada taller tiene un gerente" (y un gerente administra un sólo taller).

```

+-----+
| GERENTE      [PERS]  ---DE----> [ORG ] TALLER |
|                                     [ 1 : 1 ] |
+-----+

```

Cuando se define un concepto como [ORG] el sistema ofrece un Template con un conjunto de atributos tales como Denominación, Domicilio, etc., En este caso supongamos que seleccionamos Denominación y Domicilio, el sistema genera entonces:

```

+-----+
| DENOMINACION_ |
| TALLER [NOMB]  ---DE----> [ORG ] TALLER |
|                                     [ 1 : 1 ] |
+-----+
| DOMICILIO_    |
| TALLER [LUG ] ---DE----> [ORG ] TALLER |
|                                     [ 1 : 1 ] |
+-----+

```

A continuación informamos al sistema que "cada taller tiene varios mecánicos" (y un mecánico trabaja en un sólo taller).

```

+-----+
| MECANICO      [PERS]  ---DE----> [ORG ] TALLER |
|                                     [ N : 1 ] |
+-----+

```

A continuación informamos que "se realiza un curso para una determinada asignatura en una fecha dada". ASIGNATURA es el "tema" del evento CURSO.

Esta vista está expresada en la forma objetiva y define:

```

+-----+
| CURSO         [EVT ]  --PARA---> [CAT ] ASIGNATURA |
+-----+

```

Para los conceptos tipificados como [CAT] el sistema ofrece un Template con el atributo Denominación. En este caso nosotros seleccionamos este atributo y el sistema genera:

```

+-----+
| DENOMINACION_ |
| ASIGNATURA [NOMB] ---DE----> [CAT ] ASIGNATURA |
|                                     [ 1 : 1 ] |
+-----+

```

Continuando con la definición del evento CURSO, informamos:

```
+-----+
| CURSO      [EVT ] ---EN----> [FECH] FECHA_CURSO |
+-----+
```

Informamos que "cada asignatura tiene una duración de curso determinada":

```
+-----+
| DURACION
|          CURSO [CANT] ---DE----> [CAT ] ASIGNATURA |
|                                [ 1 : 1 ]             |
+-----+
```

Para los conceptos tipificados como [CANT] el sistema solicita la unidad de medida correspondiente (arbitrariamente definida por el diseñador con fines de documentación solamente). Informamos: DIAS.

Informamos a continuación que "la especialización de cada mecánico está definida por los cursos que ha tomado y las calificaciones obtenidas en cada curso".

Expresando esta vista en la forma objetiva:

"Se realizan Especializaciones de Mecánicos en Cursos
obteniéndose Calificaciones"

Lo que se expresa como:

```
+-----+
| ESPECIALIZACION [EVT ] ---DE----> [PERS] MECANICO |
+-----+
| ESPECIALIZACION [EVT ] ---EN----> [EVT ] CURSO |
+-----+
| CALIFICACION   [CANT] ---EN----> [EVT ] ESPECIALIZACION |
|                                [ 1 : 1 ]             |
+-----+
```

El sistema solicita la unidad de medida correspondiente a CALIFICACION. Respondemos: PUNTAJE.

Como veremos en la sección "6.2. Generación del modelo relacional normalizado" a partir de las microvistas anteriores el sistema genera el siguiente modelo relacional:

TALLERES (TALLER, DENOMINACION_TALLER, DOMICILIO_TALLER)

EMPLEADOS (EMPLEADO, NOMBRE_EMPLEADO,
DOMICILIO_EMPLEADO)

GERENTES (EMPLEADO, TALLER)

MECANICOS (EMPLEADO, TALLER)

ASIGNATURAS (ASIGNATURA, DENOMINACION_ASIGNATURA,
DURACION_CURSO)

CURSOS (CURSO, ASIGNATURA, FECHA_CURSO)

ESPECIALIZACIONES (ESPECIALIZACION, MECANICO, CURSO,
CALIFICACION)

5 . Verificación del Modelo

Una vez realizados los procesos de Abstracción y Formalización es indispensable realizar una validación del contenido del modelo conceptual, de manera de detectar errores y ambigüedades.

DBAID provee un conjunto de funciones para la visualización y verificación del modelo. También provee facilidades de consulta y descripciones del modelo conceptual en lenguaje casi-natural castellano.

Estas funciones pueden aplicarse en cualquier momento durante la construcción del modelo y especialmente cuando el diseñador cree haber concluido la etapa de formalización.

En realidad el diseñador debería utilizar las funciones de verificación en forma frecuente durante la modelización, de manera de evitar una gran acumulación de errores y ambigüedades.

Además, una vez concluida la modelización y habiendo verificado la corrección del modelo, el diseñador debe comprobar que todas las "vistas de usuario" pueden ser satisfechas con la información descrita por el modelo.

Esta comprobación final se puede realizar de dos maneras:

- Utilizando las funciones de Visualización.
- Implementando la base de datos relacional, y cargándola con datos ficticios. Es decir creando una base de datos prototipo y comprobando, mediante la utilización de un lenguaje de consulta, que estén todos los datos requeridos.

Esta verificación final puede demostrar que se han introducido conceptos que no son requeridos por ninguna vista de usuario. Estos conceptos deberían ser eliminados del modelo verificando previamente que no sean necesarios para establecer, a través de sus vinculaciones, el cierre (la unicidad) de la estructura del modelo.

El objetivo de esta sección es mostrar algunas de las verificaciones realizadas por el sistema. El sistema de menús y de Ayuda en Contexto del sistema posibilita el acceso y describe el conjunto completo de funciones, así como las acciones a tomar como resultado de la detección de errores por parte del sistema.

5.1. Funciones de Visualización

El sistema provee un conjunto completo de funciones de visualización que a través de menús y un lenguaje de consulta casi-natural permiten obtener en cualquier momento representaciones tabulares y gráficas del contenido del modelo.

Por ejemplo, permiten obtener:

- Para un concepto determinado una descripción en lenguaje casi-natural de su tipificación semántica y sus vinculaciones con otros conceptos.
- Gráficos del modelo relacional generado a partir del modelo conceptual siguiendo las reglas de generación detalladas más adelante en la sección "6.2. Generación del modelo relacional normalizado".
- Información sobre conceptos, microvistas, eventos, vinculaciones, etc. , que satisfacen determinadas condiciones (Ver "10.1. Lenguaje de consulta casi-natural").

NOTA: A cada microvista cargada en el sistema, el sistema le asigna un código interno de identificación, el cual aparece en los diferentes listados obtenibles mediante el sistema. este código interno puede ser referenciado por el diseñador cuando necesita "editar" (modificar) el modelo conceptual.

5.2. Funciones de Verificación

Las funciones de verificación provistas permiten detectar errores y ambigüedades existentes en el modelo, en cualquier momento durante el diseño.

El sistema produce mensajes de error y de advertencia, sugiriendo determinadas acciones para corregir los errores y ambigüedades detectados.

La mayor parte de los mensajes son mensajes de advertencia. El sistema no detiene al diseñador si éste decide continuar avanzando en el diseño ignorando las advertencias del sistema. El diseñador puede continuar hasta la implementación de la base de datos.

Recordemos que la estructura básica de representación del modelo de datos conceptual es una red orientada (red semántica). Las funciones de Verificación permiten, por ejemplo:

* detectar lazos cerrados ("loops") en la red, del tipo:

```
AAA [    ] --- --> [    ] BBB
BBB [    ] --- --> [    ] CCC
CCC [    ] --- --> [    ] AAA
```

* detectar conceptos o subredes de conceptos (submodelos) desconectados entre sí.

Además de verificaciones sintácticas sobre la red como las anteriores, el sistema puede detectar (entre otras):

- La existencia de homónimos (conceptos diferentes con igual denominación) y sinónimos (el mismo concepto con diferente denominación).
- Eventos potencialmente idénticos (sinónimos).
- Eventos Implícitos (ver "3.12. Eventos Implícitos").
- Eventos sin tiempo de ocurrencia, sin agente explícito, etc.
- Falta de nombres, descripciones, etc. para conceptos tipificados como [PERS], [ORG], [CAT], etc.
- Potenciales precedencias entre eventos (ver "3.11. Precedencia de Eventos").
- Diferentes tipos de vinculaciones potencialmente erróneas como en los casos siguientes.

Caso 1

```
+-----+
| ZONA      [LUG ] ---DE-----> [LUG ] REGION |
|                                     [ N : 1 ] |
+-----+
| SUBZONA   [LUG ] ---DE-----> [LUG ] ZONA   |
|                                     [ N : 1 ] |
+-----+
| SUBZONA   [LUG ] ---DE-----> [LUG ] REGION |
|                                     [ N : 1 ] |
+-----+
```


verifica que "xxxx" sea uno de las siguientes categorías:

```
LUG      (Lugar)
PTIM     (Tiempo)
EVT      (Evento)
```

y en la siguiente vinculación de Generalización:

```
+-----+
|  XXXXXXXX  [xxxx]  ---ES---->  [yyyy]  YYYYYYYY  |
+-----+
```

verifica que "xxxx" e "yyyy" sean categorías semánticamente compatibles.

5.3. El Modelo en su Forma Canónica

Las funciones de Visualización y Verificación pueden también ser utilizadas para llevar el modelo conceptual a su "forma canónica" o forma más simple.

Decimos que un modelo está en su forma canónica cuando el mismo no contiene conceptos que representen "atributos derivables" (atributos cuyo valor puede obtenerse por cálculo a partir de otros atributos) ni vinculaciones redundantes.

Denominamos "atributos fundamentales" a aquellos que no pueden ser calculados a partir de otros atributos.

El tema de las vinculaciones redundantes fue tratado en la sección "5.2. Funciones de Verificación". Analizaremos ahora el tema de los Atributos Derivables.

Por ejemplo en una estructura de Zonas en Regiones, las Ventas de una Región son el agregado o sumarización de las Ventas de las Zonas que le corresponden.

O el Puntaje Acumulado por una Escudería en una Temporada de Competencias es la suma del Puntaje Acumulado por sus Pilotos en esa misma temporada.

En muchos casos resulta conveniente durante la modelización conceptual mantener el modelo en su forma canónica, eliminando atributos derivables.

En una etapa posterior, para la optimización de la performance en el procesamiento de los datos, puede resultar conveniente introducir en el modelo algunos atributos derivables. Esta inclusión representa también una forma de "desnormalización" de la base de datos y puede requerir la inclusión de conceptos (eventos) y vinculaciones adicionales en el modelo, de manera de establecer un concepto "propietario" para dichos atributos.

A continuación analizaremos ventajas y desventajas de la implementación de bases de datos en su forma canónica.

5.3.1. Ventajas de la Forma Canónica

La implementación de una base de datos en su forma canónica (relaciones normalizadas, sin atributos derivables, y sin vinculaciones redundantes) presenta las siguientes ventajas:

- simplifica los programas de actualización de los datos (la actualización se realiza en un sólo lugar).
- facilita a los usuarios la comprensión del contenido de la base de datos.
- simplifica la reestructuración de la base de datos en respuesta a cambios en el ambiente de aplicación.
- contribuye a preservar un mayor grado de "independencia de datos" en la aplicación reduciendo errores y costos de mantenimiento de los programas, al reducir el impacto de cambios en la estructura de datos de la base de datos.
- contribuye a una performance global más estable a lo largo de la vida útil de la aplicación. Reduce el impacto sobre la performance de procesamiento por cambios en la mezcla de procesos o por el agregado de nuevos procesos.

La experiencia muestra que las bases de datos implementadas inicialmente en forma no canónica, cuando deben ser ampliadas para la incorporación de datos o procesos adicionales, evolucionan hacia su forma canónica.

La forma canónica permite la evolución de las estructuras de datos oponiendo la mínima resistencia al cambio, lo cual es sumamente importante si aceptamos que una de las necesidades más dinámicas de cualquier organización es la necesidad de información.

5.3.2. Desventajas de la Forma Canónica

Según vimos en la sección anterior, son importantes las ventajas de la forma canónica. Su principal desventaja es una menor performance en consultas a la base de datos que involucran a atributos derivados, ya que estos deberán ser calculados nuevamente en cada consulta efectuada.

La incorporación de atributos derivables en una base de datos será por consiguiente una solución de compromiso determinada por la Volatilidad (frecuencia de actualización) y Actividad (frecuencia de acceso) de estos atributos para una mezcla de procesos determinada.

Con la introducción de desnormalizaciones en el esquema de una base de datos, éste "se orienta" de manera de maximizar la performance de la aplicación para una mezcla de procesos dada.

Sin embargo, debe recordarse que cuando se agreguen nuevos procesos la mezcla original cambiará así como la performance global resultante puede dejar de ser la óptima.

Mediante la implementación de esquemas en su forma canónica se logra una performance global más estable frente a cambios en la mezcla de procesamiento.

Por las razones anteriormente expuestas, el alejamiento de la forma canónica, en caso de resultar indispensable, debería ser el menor compatible con los requerimientos de performance.

Más adelante, en la sección "6. Diseño Lógico y Diseño Físico" analizaremos posibles maneras de desnormalizar el esquema relacional, así como sus ventajas y desventajas, cuando es necesario orientar el esquema a determinadas situaciones de procesamiento.

5.4. El Tiempo y las Bases de Datos

Mencionamos previamente que una base de datos constituye un modelo de algún ambiente de aplicación y que en cada momento el contenido de la base de datos representa una "instantánea" del estado de dicho ambiente. Expresamos también que cada cambio en la base de datos refleja un cambio ocurrido en el ambiente de aplicación.

5.4.1. El Tiempo y los Datos en una Base de Datos

Cada dato en una base de datos relacional está definido en los siguientes términos:

- Nombre de la Relación
- Nombre del Atributo
- Valor del Atributo
- TIEMPO correspondiente al Valor

Si bien el tiempo constituye la piedra angular en el diseño y explotación de una base de datos, frecuentemente este aspecto no se explicita adecuadamente.

La mayoría de los diferentes modelos de datos asumen que el tiempo de un dato es "ahora", es decir que el valor actual de un dato en la base de datos corresponde al momento actual en el ambiente de la aplicación.

Cuando éste no es el caso, es decir cuando el valor del dato corresponde a algún momento del pasado o del futuro (valor planeado o esperado) es necesario identificar ese momento ya sea indicándolo en el Nombre del Atributo, ya sea asociándole otro dato que especifique el tiempo correspondiente.

En general la mayoría de los sistemas de información focaliza la atención en el "estado actual" del ambiente de aplicación, aunque en algunos casos también lo hace sobre valores correspondientes a estados del pasado y también valores previstos para el futuro de ciertos datos.

Para responder a consultas tales como:

- "Cuáles empleados ganaron más de 70.000 pesos en el año 1986 ?"
- "Cuándo se le incrementó el sueldo por última vez a Juan Perez ?".
- "Cuál es el nivel de ventas previsto para el primer trimestre del año próximo ?"

es necesario poder representar información de "estados distintos al actual, pero de interés actual".

En algunas aplicaciones específicas, tales como sistemas de historias clínicas, la representación del transcurso del tiempo resulta crítica.

5.4.2. El Tiempo y la Representación de Objetos en la Base de Datos

Un determinado elemento (una persona en particular, por ejemplo) puede existir en el ambiente de aplicación sin estar representado en la base de datos, o su representación en la misma puede ser transitoria.

En este último caso el elemento tiene un intervalo de tiempo en el que nos interesamos en él y durante ese intervalo mantenemos una representación suya en la base de datos.

En algunos casos la representación de un elemento en la base de datos es "permanente" (el elemento es de interés permanente).

5.4.3. Puntos e Intervalos de Tiempo

Los conceptos temporales pueden representar momentos, o "puntos de tiempo", o "intervalos de tiempo" (períodos de tiempo de una cierta duración).

DBAID ofrece las categorías PTIM y CTIM para representar ambos tipos de conceptos temporales. Además, debido a la elevada frecuencia de uso, el sistema ofrece también la categoría FECH la cual es un caso particular de la categoría PTIM.

Cuando un concepto se define como [PTIM] el sistema ofrece un menú de múltiples opciones (Template) conteniendo:

```
Año
Mes
Día
Hora
Minutos
Segundos
AA/MM/DD
DD/MM/AA
AA*10000+MM*100+DD (como número entero)
```

En la modelización del tiempo, por ejemplo al definir el tiempo de ocurrencia de un evento, es posible especificar que el evento se realiza "EN" determinado momento [PTIM] y "POR" un período de tiempo [CTIM], o especificar que el evento se realiza "DESDE" un determinado momento [PTIM] y "HASTA" un momento [PTIM] posterior.

5.4.4. Conceptos y Vinculaciones Temporales

Existen conceptos de naturaleza temporal los cuales pertenecen a las categorías semánticas "FECH" y "PTIM", por ejemplo: Fecha, Mes, Año, Hora, etc.

Los Eventos también pueden desempeñar un rol temporal en determinadas situaciones, por ejemplo en el siguiente caso:

```
COMPETENCIA [EVT ] --EN----> [EVT ] TEMPORADA
```

Las vinculaciones "EN", "DESDE", "HASTA", mencionadas previamente, así como la vinculación "POR", cuando se utilizan para establecer vinculaciones con conceptos temporales, representan vinculaciones temporales.

Si bien la vinculación "EN" tiene la connotación de "en algún momento de..." o "durante...", mientras que la vinculación "POR" tiene una connotación de simultaneidad, ambas vinculaciones reciben el mismo tratamiento por parte del sistema.

Ejemplos:

```
COMPETENCIA [EVT ] --EN----> [EVT ] TEMPORADA
```

```
CONTRATACION [EVT ] --POR----> [EVT ] TEMPORADA
```

Desde el punto de vista de la modelización conceptual puede ser importante recordar la siguiente regla de inferencia:

"Si algo existe, ocurre o vale durante un intervalo de tiempo t , también existe, ocurre o vale durante cualquier subintervalo de t ".

Por ejemplo, si un Piloto y una Escudería están asociados durante una Temporada, también lo están por cada Competencia de esa Temporada (Competencia es un "subevento" de Temporada).

5.4.5. El Tiempo y los Manejadores (DBMS) Relacionales

Mencionamos previamente que en algunas aplicaciones, tales como los sistemas de historias clínicas, la consideración del tiempo resulta de importancia fundamental.

Los manejadores de bases de datos (dbms) actuales no proveen facilidades de procesamiento temporal. Cuando estas resultan necesarias deben ser implementadas a nivel de los programas de la aplicación.

Existen actualmente desarrollos que tienen el propósito de extender el modelo relacional incorporándole facilidades de procesamiento temporal "nativas". Estos modelos extendidos reciben la denominación de "Time-Relational Models".

Las facilidades de procesamiento temporal permiten definir y procesar "time views" las cuales son vistas de la base de datos en un tiempo dado (presente, pasado o futuro).

Es posible, por ejemplo, realizar actualizaciones "retroactivas" a la base de datos (se actualiza una "time view" correspondiente al pasado, pudiendo la modificación eventualmente proyectarse sobre "time views" subsiguientes en el tiempo).

También se considera la posibilidad de realizar actualizaciones "proactivas", es decir actualizaciones sobre "time views" correspondientes al futuro.

Las facilidades de procesamiento temporal están orientadas también a proveer "independencia de tiempo" (como extensión del concepto de "independencia de datos") de manera de aislar a las aplicaciones en la mayor medida posible de los mecanismos de procesamiento temporal, de cambios evolutivos en los esquemas, etc.

5.4.6. Evolución de los Esquemas en el Tiempo

Expresamos previamente que la estructura de una base de datos refleja la estructura del sistema que la misma representa.

En general la estructura de los sistemas representados no es estática sino que evoluciona con el tiempo, determinando cambios en la estructura de la base de datos.

Los manejadores de bases de datos relacionales proveen mecanismos de independencia de datos, cuyo propósito es minimizar la necesidad de reestructurar las bases de datos y también minimizar el impacto que las reestructuraciones de las bases de datos producen sobre los programas de aplicación.

Sin embargo, al margen de las facilidades brindadas por el manejador de bases de datos, es posible diseñar la estructura de una base de datos, de manera de que no sea afectada (o lo sea en forma mínima) por cambios en la estructura del sistema modelizado.

5.5. Introduciendo Futuro en el Modelo Conceptual

Mencionamos al tratar "5.4. El Tiempo y las Bases de Datos" que la estructura de los sistemas modelizados no es estática sino que cambia con el transcurrir del tiempo, lo que puede obligar a modificar la estructura de la base de datos.

Expresamos también que, al margen de las facilidades de independencia de datos provistas por el manejador de bases de datos utilizado, es posible diseñar la estructura de la base de datos de manera de minimizar las necesidades de reestructuración debidas a cambios en el sistema modelizado.

La experiencia demuestra que ciertos tipos de cambios se presentan con mucha frecuencia, por lo que en principio es posible prever dichos cambios desde el momento mismo del diseño de la base de datos. Es conveniente en estos casos introducir modificaciones "a priori" en el modelo conceptual, operación que denominamos "introducción de futuro".

Los cambios en el sistema modelizado pueden responder a múltiples causas. Por ejemplo, si el sistema modelizado es una empresa, las causas pueden ser: cambios en la legislación, reorganizaciones internas, cambios en procesos, cambios en productos y servicios, etc.

Por ejemplo, en la mayoría de las empresas un empleado puede estar asignado a un único departamento en un momento dado y este hecho puede estar reflejado en la estructura del modelo conceptual con la siguiente vinculación:

```
EMPLEADO [PERS] ---DE----> [ORG ] DEPARTAMENTO [1]
[ N : 1 ]
```

Qué pasaría si un empleado cambia de departamento pero se desea mantener historia sobre su desempeño en el departamento anterior ?.

O bien, que pasaría si un empleado es asignado a dos departamentos simultáneamente, "part-time" por ejemplo ?.

Estas nuevas situaciones no estarían contempladas en el modelo conceptual, ni en el modelo relacional derivado y no podrían reflejarse en el contenido de la base de datos.

Cómo es posible aumentar la flexibilidad del modelo de manera de que responda, por un lado, a la estructura actual del sistema modelizado y, por otro lado, aumentar su capacidad de absorber cambios sin modificarse ?.

Existen tres maneras posibles [Wilmot, 1984] [Whitener,1988]:

1. La primera es: Transformando las vinculaciones [1:N] de Agregación entre conceptos en Eventos que asocien [M:N] a dichos conceptos.

El consejo es: "Desconfíe de las vinculaciones de Agregación, pueden ser temporarias o eventuales (no mantenerse a lo largo del tiempo)".

En el ejemplo anterior, la vinculación de agregación [1] podría eliminarse introduciendo al mismo tiempo el siguiente evento:

```
ASIGNACION [EVT ] ---DE-----> [PERS] EMPLEADO
ASIGNACION [EVT ] ---A-----> [ORG ] DEPARTAMENTO
ASIGNACION [EVT ] ---DESDE--> [FECH] ....
ASIGNACION [EVT ] ---HASTA--> [FECH] ....
```

Esta modificación en el modelo conceptual, impacta directamente a los programas de la aplicación, el acceso a los datos, la preservación de la integridad referencial, etc.

2. La segunda es: Transformando las vinculaciones [1:1] de Propiedad potencialmente inestables entre conceptos en vinculaciones que asocien [1:N] a dichos conceptos.

Consideremos como ejemplo el siguiente caso en el que cada empleado tiene (o puede tener) un único teléfono:

```
EMPLEADO [PERS] ---CON----> [DISP] TELEFONO
                        [1:1]
```

Qué pasaría si en el futuro cada empleado pudiese tener varios teléfonos ?.

La situación no estaría contemplada y no podría reflejarse en el contenido de la base de datos.

Si del análisis de estabilidad surgiese que esta situación es muy probable que se de en el futuro, sería conveniente transformar la vinculación [1:1] en [1:N].

3. La tercera tiene que ver con el Diseño Lógico de la base de datos (ver "6.4. Consideraciones sobre la Identificación Unívoca de Tuplas"): No incluir información (datos) en los atributos de identificación de las Relaciones (esto significa, por ejemplo, no incluir claves externas en la clave de identificación de una Relación).

EL ATRIBUTO DE IDENTIFICACION DEBE SOLO IDENTIFICAR Y NO DEBE CONTENER INFORMACION.

Consecuencias

El precio a pagar por aumentar la adaptabilidad del modelo, y su nivel de independencia de datos, será en mayor o menor medida una menor performance de la aplicación y una mayor utilización de almacenamiento externo, por lo que cada modificación debería evaluarse en función de costos y beneficios, contemplando también la probabilidad de que el cambio que se desea absorber se produzca efectivamente.

6. Diseño Lógico y Físico

El resultado de la modelización conceptual es un modelo de datos construido desde el punto de vista de los usuarios y de la aplicación con independencia de cómo los datos van a ser almacenados y procesados en el computador.

Durante el diseño lógico se convierte el modelo conceptual al modelo de almacenamiento equivalente (en este caso al modelo relacional equivalente).

También en esta etapa se contempla si la base de datos va a ser distribuída o centralizada, se define el manejador (DBMS) a utilizar, se analizan los aspectos de procesamiento: procesos, sus frecuencias, tiempos de respuesta requeridos, etc..

Se efectúan las adaptaciones necesarias en el esquema de la base de datos (introduciendo alguna desnormalización) obteniéndose el esquema lógico que se implementará finalmente.

En la etapa de "Diseño Físico" se realiza principalmente la adaptación al computador contemplando las características y restricciones del mismo: disponibilidad de espacio de almacenamiento, velocidades de procesamiento, tiempos de acceso, concurrencia con otras bases de datos y aplicaciones, etc..

6.1. El Foco de Atención Cambia

Desde el punto de vista del diseñador el comienzo del diseño lógico es un momento clave ya que debe, además de pensar en la "intensión" (esquema) de la base de datos, comenzar a pensar en la "extensión" (contenido de datos) de la misma: tuplas, identificadores unívocos, claves externas, características computacionales para los atributos, etc..

6.2. Generación del Modelo Relacional Normalizado

Una vez terminada la etapa de Diseño Conceptual, el sistema determina automáticamente cuál es el modelo relacional equivalente. Define las Relaciones y los Atributos correspondientes a cada Relación que se incluirá en el modelo relacional. Solicita las características computacionales para cada atributo (tipo, longitud, etc.) pudiendo asumir características por defecto.

El sistema realiza la conversión del Modelo Conceptual al Modelo Relacional de la siguiente manera:

```
+-----+
| El sistema genera una Relación para cada concepto que |
| tenga vinculaciones hacia otros conceptos.           |
+-----+
```

El nombre de la Relación Generada es el plural del nombre del concepto. Así el concepto EMPLEADO generará la Relación EMPLEADOS.

El sistema genera automáticamente un Atributo de Identificación para cada Relación. Así, a la Relación EMPLEADOS en el modelo relacional se le asigna un Atributo de Identificación para sus tuplas que se denomina "Empleado". (ver "6.4. Consideraciones sobre la identificación unívoca de Tuplas").

```
+-----+
| Para cada vinculación hacia otro concepto, el sistema |
| genera un atributo (considerándolo de descripción o de |
| vinculación según sea el caso).                       |
+-----+
```

Si la vinculación es hacia un concepto que a su vez no tiene vinculaciones hacia otros conceptos el atributo generado se considera un atributo de descripción. En los demás casos el atributo generado se considera un atributo de vinculación (o clave externa).

El sistema asigna también un conjunto de índices a cada Relación generada, de la siguiente manera:

- Para cada atributo de identificación genera un índice "unique" de manera de asegurar la unicidad de tuplas.
- Para cada atributo de vinculación genera un índice para proveer una "clave externa" para ese atributo (estos atributos son frecuentemente fusionados (JOINED) pues establecen las vinculaciones lógicas existentes entre las Relaciones).
- Para los eventos, en función de la Cardinalidad de sus circunstanciales (ver "6.3. Cardinalidad de los Eventos" a continuación), genera índices concatenando algunos de sus atributos de vinculación, de manera de introducir las restricciones de cardinalidad correspondientes.

Genera finalmente los comandos DDL (Data Definition Language) requeridos por el DBMS para la implementación física de la base de datos.

El proceso de diseño termina, de esta manera, con la base de datos lista para comenzar el almacenamiento y procesamiento de los datos.

6.3. Cardinalidad de los Eventos

Consideremos la vista siguiente:

```
+-----+
| Cada temporada las escuderías contratan pilotos. |
| La duración de cada contratación es la temporada |
| completa, no pudiendo un piloto pasar a otra escudería |
| en la mitad de una temporada. |
+-----+
```

La segunda parte de la vista "La duración de..." está especificando una Restricción de Cardinalidad sobre las circunstancias del evento.

IMPORTANTE: En este caso la cardinalidad también tiene que ver con conceptos individuales (ver "3.4.1. Cardinalidad de las vinculación entre dos conceptos"). Así en la vista anterior deben considerarse los conceptos individuales <TEMPORADA>, <PILOTO>, <ESCUADERIA> y <CONTRATO>. Esta consideración es válida para el análisis siguiente.

Habiendo suministrado al sistema las siguientes microvistas:

```
+-----+
| CONTRATACION [EVT ] ---DE-----> [ORG ] ESCUDERIA |
+-----+
| CONTRATACION [EVT ] ---A-----> [PERS] PILOTO |
+-----+
| CONTRATACION [EVT ] ---EN-----> [PTIM] TEMPORADA |
+-----+
```

El sistema determina la cardinalidad de las circunstancias del evento CONTRATACION efectuando al diseñador las siguientes preguntas, debiendo el diseñador contestar SI o NO a cada una de ellas:

```
+-----+
| UNA ESCUDERIA, UN PILOTO, VARIAS TEMPORADAS ? |
| Respuesta: SI |
+-----+
| UN PILOTO, UNA TEMPORADA, VARIAS ESCUDERIAS ? |
| Respuesta: NO |
| (Nota: una restricción de cardinalidad |
| resulta necesaria) |
+-----+
| UNA TEMPORADA, UNA ESCUDERIA, VARIOS PILOTOS ? |
| Respuesta: SI |
+-----+
```

En el esquema relacional, para la Relación CONTRATACIONES:

```
CONTRATACIONES(CONTRATACION, ESCUDERIA, PILOTO,
                TEMPORADA)
```

El sistema, para introducir la restricción de cardinalidad requerida hace uso de las facilidades de índices del manejador relacional, (RDBMS) generando un índice "unique" en la relación CONTRATACIONES para la concatenación de los atributos PILOTO y TEMPORADA.

De esta manera se asegura que en la relación CONTRATACIONES sólo habrá una tupla para un determinado piloto en una temporada dada.

6.4. Consideraciones sobre la Identificación Unívoca de Tuplas

En cada relación de una base de datos, cada tupla deberá contener en su Clave Primaria (ver "2.3. El Modelo de Datos Relacional") un valor que la distinga de las demás a los efectos del procesamiento de sus datos.

La Clave Primaria a utilizar debe satisfacer una condición muy importante para asegurar la conservación de la independencia de datos. Es de fundamental importancia satisfacer el siguiente criterio:

```

+-----+
|
| LA CLAVE PRIMARIA:
|
| ** NO DEBE CONTENER INFORMACION (datos).
|
| ** DEBE SOLO IDENTIFICAR.
|
+-----+

```

Los datos deben estar contenidos en los atributos dependientes de cada relación.

La Clave Primaria debe estar compuesta por un único atributo denominado Atributo de Identificación. Este identificador debe ser sólo un código (por ejemplo un número secuencial) CARENTE DE SIGNIFICADO.

Por otra parte, el valor que puede tomar el atributo de identificación de una tupla debe ser:

- UNIVOCO: se debe impedir que existan dos tuplas para un mismo elemento, por ejemplo que un determinado Empleado esté representado por dos tuplas.
- NO-AMBIGUO: se debe impedir que diferentes elementos tengan la misma identificación. Por ejemplo que dos Empleados tengan la misma identificación.
- INVARIANTE: el valor debe ser invariante en el tiempo, durante todo el intervalo de tiempo en que el elemento se mantiene representado en la base de datos.

Este último criterio es de gran importancia pues un cambio en el valor del identificador de una tupla puede requerir propagar dicho cambio sobre una gran cantidad de tuplas de otras relaciones en la base de datos, operación que puede llegar a ser de gran complejidad lógica.

En nuestra metodología suponemos que cada Relación tiene un Atributo de Identificación cuyo nombre es el mismo que el del concepto que le dió origen. Así, como mencionáramos previamente, la relación "EMPLEADOS" tendrá por defecto como identificador un atributo denominado "Empleado". Se satisface de esta manera el criterio de que el identificador no debe contener información (sólo identificar) y se logra conservar la independencia de datos del esquema.

En algunos casos puede ocurrir que las tuplas de una relación no puedan ser identificadas con independencia de otras tuplas de otras relaciones o no resulte natural hacerlo. Por ejemplo consideremos el caso de los ítems de una Orden de Compra, los cuales normalmente son identificados haciendo referencia a la identificación de la Orden de Compra correspondiente.

En estos casos se dice que existe una dependencia de identificación.

La solución más directa en el ejemplo anterior es utilizar como valor para el atributo de identificación en cada tupla, la concatenación de valores de los identificadores de la Orden de Compra y del ítem correspondiente. Así, el ítem 19 de la Orden de Compra 7243 podría tener como identificación unívoca el número 724319, o la secuencia de caracteres "7243-19".

Una solución de este tipo tiene la ventaja de no requerir cambios en el esquema relacional.

A continuación incluimos el modelo relacional generado para el "Caso Talleres" cuya modelización conceptual fuera realizada en la sección "4.9. Ejemplo de Modelización Conceptual". Este modelo consta de las siguientes Relaciones y Atributos:

TALLERES (TALLER, DENOMINACION_TALLER, DOMICILIO_TALLER)

EMPLEADOS (EMPLEADO, NOMBRE_EMPLEADO,
DOMICILIO_EMPLEADO)

GERENTES (EMPLEADO, TALLER)

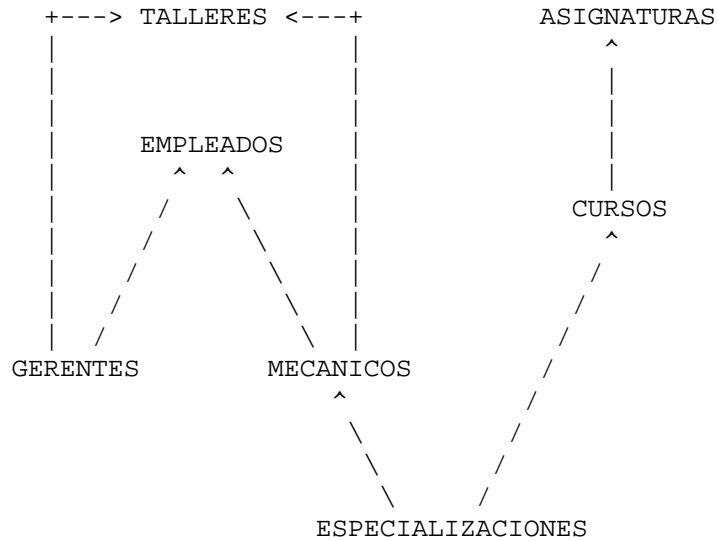
MECANICOS (EMPLEADO, TALLER)

ASIGNATURAS (ASIGNATURA, DENOMINACION_ASIGNATURA,
DURACION_CURSO)

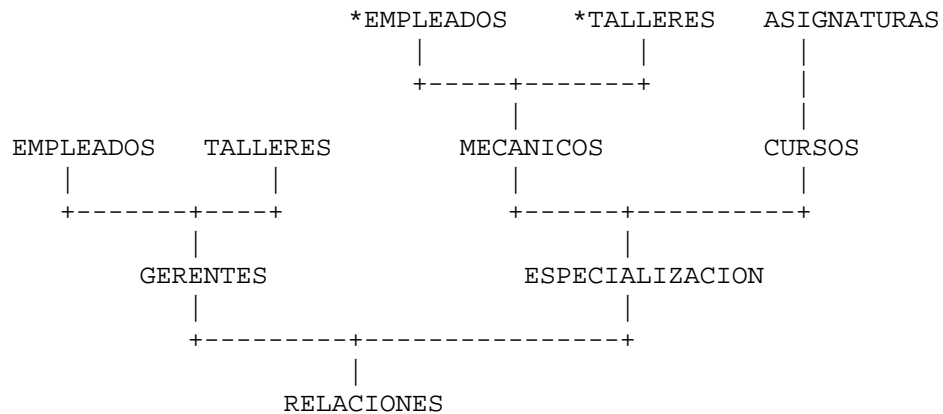
CURSOS (CURSO, ASIGNATURA, FECHA_CURSO)

ESPECIALIZACIONES (ESPECIALIZACION, MECANICO, CURSO,
CALIFICACION)

La figura siguiente muestra las vinculaciones lógicas entre las relaciones que componen el esquema:



Para facilitar la visualización de modelos complejos la función de graficación del sistema representa la red de relaciones que constituye un modelo en forma de árbol. Así la red anterior resulta graficada de la siguiente forma:



Los nodos cuya denominación está precedida por un asterisco, por ejemplo *EMPLEADOS y *TALLERES en el gráfico anterior, son nodos "fantasmas". El asterisco denota que el nodo "real" ha sido graficado previamente a la izquierda en el diagrama, juntamente con todas sus eventuales ramificaciones superiores (relaciones con mayor precedencia).

Para asegurar la unicidad de tuplas el sistema crea índices unique para las siguientes relaciones y atributos:

Relación	Atributo
TALLERES	TALLER
EMPLEADOS	EMPLEADO
GERENTES	EMPLEADO
MECANICOS	EMPLEADO
ASIGNATURAS	ASIGNATURA
CURSOS	CURSO
ESPECIALIZACIONES	ESPECIALIZACION

El sistema crea además índices para los atributos de vinculación de manera de proveer claves externas a las Relaciones:

Relación	Atributo de Vinculación
GERENTES	TALLER
MECANICOS	TALLER
CURSOS	ASIGNATURA
ESPECIALIZACIONES	MECANICO
ESPECIALIZACIONES	CURSO

Finalmente el sistema solicita la cardinalidad de los eventos, de manera de crear índices unique que provean las restricciones de cardinalidad requeridas.

En este ejemplo el sistema determina primero la cardinalidad de los circunstanciales del evento CURSO efectuando al diseñador las siguientes preguntas, debiendo el diseñador contestar SI o NO a cada una de ellas:

UNA ASIGNATURA, VARIAS FECHA_CURSO ? Respuesta: SI
UNA FECHA_CURSO, VARIAS ASIGNATURAS ? Respuesta: SI

Como resultado de las respuestas anteriores el sistema encuentra que no debe introducir restricciones de cardinalidad para la relación CURSOS.

6.5.2.1. Introduciendo Atributos Derivables

Si la introducción de atributos derivables no se realizó durante la modelización conceptual, puede hacerse en esta etapa.

La introducción de atributos derivables en la base de datos evita que los mismos tengan que ser calculados cada vez que se desea conocer su valor, pero implica que los mismos deben ser calculados y actualizados cada vez que cambian los atributos fundamentales de los que ellos se derivan.

De esta manera, la introducción de atributos derivables mejora el tiempo de acceso a la información pero empeora el tiempo de actualización requerido.

Los atributos derivables se justifican, por consiguiente, cuando aparecen en múltiples y/o frecuentes salidas y el costo de calcularlos cada vez es elevado en comparación con el costo de mantenerlos actualizados, es decir cuando tienen una alta "actividad" y baja "volatilidad".

Los atributos derivables se agregan a relaciones existentes o a relaciones creadas "ad-hoc".

6.5.2.2. Agregando Vinculaciones Redundantes

En la sección "5.2. Funciones de Verificación" mostramos un ejemplo (CASO 1) de vinculación redundante, la cual debía ser eliminada para llevar el modelo conceptual a su forma canónica.

A la inversa, dada la siguiente situación canónica:

```

+-----+
| ZONA      [LUG ] ---DE-----> [LUG ] REGION |
|                                     [ N : 1 ] |
+-----+
| SUBZONA   [LUG ] ---DE-----> [LUG ] ZONA   |
|                                     [ N : 1 ] |
+-----+

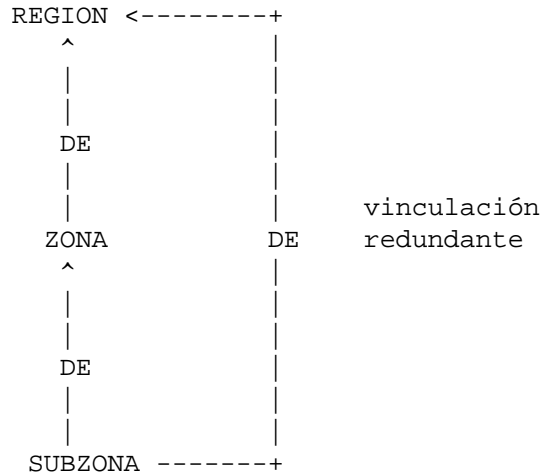
```

es posible optimizar el tiempo de acceso a expensas del tiempo de actualización, mediante el agregado de la vinculación redundante:

```

+-----+
| SUBZONA   [LUG ] ---DE-----> [LUG ] REGION |
|                                     [ N : 1 ] |
+-----+

```



En el caso que estamos analizando, mediante la introducción de la vinculación redundante (semánticamente correcta), dada una SUBZONA es posible acceder a la REGION de esa SUBZONA sin necesidad de acceder a la ZONA correspondiente, y viceversa.

Sin embargo, en el modelo relacional correspondiente, aparecerá en la relación SUBZONAS un atributo de vinculación adicional denominado REGION.

En caso de que una determinada ZONA sea cambiada de REGION deberán actualizarse los valores de los atributos REGION en ambas relaciones: ZONAS y SUBZONAS. La introducción de la vinculación redundante resultará beneficiosa cuando la cantidad de transferencias de ZONAS entre REGIONES sea poco frecuente en relación con la cantidad de accesos a través de la vinculación redundante.

Al considerar la posibilidad de introducir vinculaciones redundantes en un modelo debe verificarse, en cada caso, que ellas sean semánticamente correctas.

6.5.2.3. Desnormalizando Relaciones

Otra forma de optimizar el tiempo de acceso a expensas del tiempo de actualización de ciertos datos es incluyendo desnormalización en el esquema de la base de datos. Como toda desnormalización tiene efectos secundarios indeseados, es conveniente tener en cuenta la siguiente recomendación:

```

+-----IMPORTANTE-----+
| Antes de considerar la posibilidad de implementar un |
| esquema desnormalizado, deberían ensayarse sobre una |
| base de datos prototipo todas las facilidades de |
| optimización provistas por el manejador relacional |
| (RDBMS). |
+-----+
  
```

A continuación incluiremos algunos ejemplos de desnormalización, tratando de ver distintas posibilidades, pero también tratando de mostrar las "anormalidades" o efectos secundarios indeseados que aparecen en cada alternativa.

Como puede imaginarse aparecen múltiples alternativas de desnormalización resultando imposible recomendar "a priori" cual es la más conveniente de aplicar en cada caso (la que produce el efecto deseado con la menor cantidad de efectos colaterales indeseados).

Ejemplos

En la siguiente base de datos de un sistema de reservas de pasajes aéreos, supongamos que el esquema canónico es:

PERSONAS		VUELOS				
persona	datos de	vuelo	origen	destino	fecha	...
ID	la persona	ID				
UUUUUUUU		UUUUUU				
	v		v		UUUUU: clave	
	v	RESERVACIONES	v		única	
reservación	persona	vuelo	cantidad de	SSSSS: clave		
ID	ID	ID	plazas	secundaria		
UUUUUUUUUUUUU	SSSSSSSS	SSSSSS		o externa		

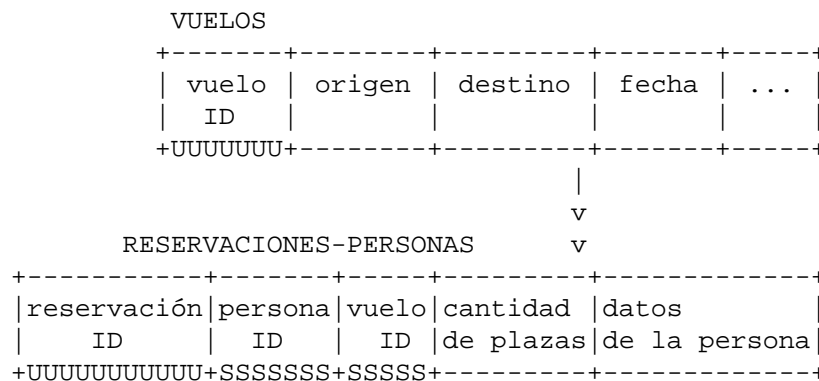
Si la mayor actividad de consultas es para conocer las reservaciones de una persona podríamos violar la PRIMERA FORMA NORMAL introduciendo grupos repetitivos conteniendo los datos de RESERVACIONES (lo cual no es aceptable por los manejadores de bases de datos relacionales basados en el Cálculo Relacional):

PERSONAS-RESERVACIONES		RESERVACIONES		
persona	datos de	reservación	vuelo	cantidad
ID	la persona	ID	ID	de plazas
UUUUUUUUUU		SSSSSSSSSSSSSS	SSSSSSSS	

VUELOS				
vuelo	origen	destino	fecha	...
ID				
UUUUUUUUU				

Puede verse que, en este ejemplo, al orientar el esquema lógico se pierde la posibilidad de acceder a todas las personas que tienen reservaciones para un determinado vuelo, salvo que sea posible introducir índices secundarios sobre grupos repetitivos, lo cual implica contar con un DBMS que permita crear índices secundarios sobre atributos incluidos en grupos repetitivos.

En cambio si la mayor actividad de consulta fuese por reservación podríamos violar la SEGUNDA FORMA NORMAL introduciendo una dependencia funcional incompleta (datos de la relación PERSONAS llevados a la relación RESERVACIONES, y eliminación de la relación PERSONAS).



En este caso los datos de las personas que tienen reservaciones aparecerán en la base de datos repetidos tantas veces como reservaciones tenga cada persona.

En general, cuando nos apartamos de las formas normales, nos encontramos con "anormalidades" en la actualización de la base de datos debido a la introducción de datos redundantes y a la evolución hacia estructuras de datos más complejas. Esto significa que para mejorar la performance de acceso el precio que inevitablemente hay que pagar es un desmejoramiento de la performance de actualización.

Por este motivo recomendamos que, antes de considerar la posibilidad de implementar una base de datos no normalizada, se ensayen sobre una base de datos prototipo normalizada todas las facilidades de optimización provistas por el manejador relacional que se utilizará.

6.5.3. Absorción de Relaciones

En nuestro "Caso Talleres" teniendo en cuenta que las relaciones GERENTES y MECANICOS tienen los mismos atributos, y que no tienen atributos específicos, es posible absorberlas (haciendo concesiones semánticas) en la relación EMPLEADOS (con la cual tienen una vinculación [1:1]) introduciendo en ésta los siguientes atributos:

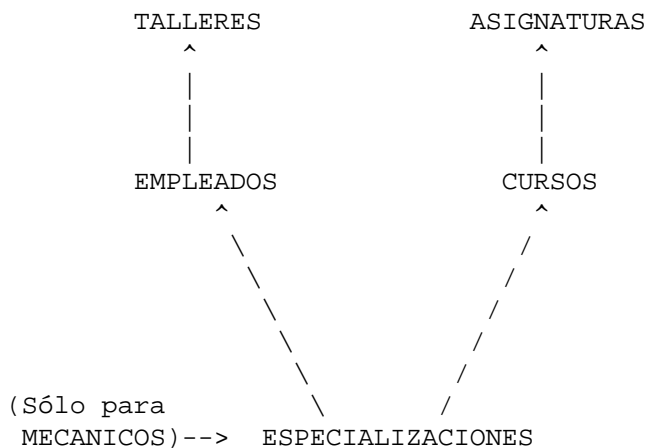
TIPO_EMPLEADO [CAT]	---DE---	[PERS] EMPLEADO
	[1 : 1]	
TALLER [ORG]	---DE---	[PERS] EMPLEADO
	[1 : 1]	

El atributo TIPO_EMPLEADO puede tomar los valores "GERENTE" y "MECANICO" (o bien códigos sustitutos tales como "G" y "M").

El esquema relacional queda entonces:

- TALLERES (TALLER, DENOMINACION_TALLER, DOMICILIO_TALLER)
- EMPLEADOS (EMPLEADO, NOMBRE_EMPLEADO, DOMICILIO_EMPLEADO, TIPO_EMPLEADO, TALLER)
- ASIGNATURAS (ASIGNATURA, DENOMINACION_ASIGNATURA, DURACION_CURSO)
- CURSOS (CURSO, ASIGNATURA, FECHA_CURSO)
- ESPECIALIZACIONES (ESPECIALIZACION, MECANICO, CURSO, CALIFICACION)

La siguiente figura muestra las vinculaciones lógicas entre las relaciones que componen el nuevo esquema:



6.5.4. División de Relaciones

Cuando en una relación existe una marcada diferenciación entre la actividad y volatilidad de sus atributos es posible optimizar la performance dividiendo dicha relación en dos (o más) partes.

Una parte conteniendo los atributos más activos y/o volátiles y otra(s) parte(s) conteniendo los restantes atributos.

La optimización se produce por reducción del volumen de información transferida entre los dispositivos de almacenamiento externo y el procesador.

6.5.5. Facilidades de Optimización de los RDBMS

La mayoría de los manejadores de bases de datos relacionales (RDBMS) proveen facilidades de optimización similares.

RDBMSs tales como SQL/DS, DB2, ORACLE, INGRES, etc. proveen facilidades de "clustering".

Las facilidades de "clustering" (agrupamiento físico en el almacenamiento en discos) de índices y de tuplas pertenecientes a diferentes relaciones de la base de datos permiten en muchos casos obtener resultados comparables a los de una desnormalización del esquema, sin necesidad de alterar el mismo.

Para RDBMSs como los mencionados, existen recomendaciones comunes respecto de cuando y cómo utilizar sus facilidades de optimización. Como ejemplo, mencionaremos aquí algunas de ellas (para información más amplia ver los manuales de cada producto).

Estas recomendaciones tienen que ver con:

- En qué casos y cómo utilizar índices.
- En qué momento crear los índices.
- Cómo disponer las relaciones en el almacenamiento en discos o en archivos virtuales ("dbspaces").
- Cómo dimensionar y disponer los "buffer pools".

Los dos últimos puntos pertenecen al dominio del diseño físico.

Algunas de las recomendaciones relacionadas con índices son, por ejemplo:

- Utilizar las facilidades de "clustering".
- Crear índices (preferiblemente "clustered") sobre atributos frecuentemente "joined".
- Crear índices para atributos mencionados frecuentemente en cláusulas "where".
- Crear índices sobre atributos para los que se calculan frecuentemente: count, min, max, sum, avg.
- NO CREAR INDICES para:
 - relaciones de bajo número de tuplas.
 - atributos con valores muy repetitivos.
 - atributos cuyos valores cambian muy frecuentemente (alta volatilidad).

Se recomienda que los índices sean creados posteriormente a la carga de datos en las relaciones.

Una gran cantidad de índices sobre relaciones con alta volatilidad de datos puede resultar contraproducente debido a la elevada actividad que la actualización de dichos índices demanda.

Como ejemplos de recomendaciones para el diseño físico tenemos:

- * Poner relaciones de dimensiones medianas o grandes en "dbspaces" separados.
- * Mantener "buffer pools" separados para índices y datos.

6.6. Ventajas y Desventajas de la Desnormalización

Agotadas las posibilidades de optimización vía sintonía (tuning) de la base de datos, si no se logran los tiempos de respuesta y performances requeridas en los procesos más críticos, no queda más alternativa que intentar lograr dichas performances mediante una desnormalización de la misma, combinando, por ejemplo, múltiples tablas que son utilizadas frecuentemente en conjunto ("joined") en una única tabla.

Ventajas

- menor cantidad de relaciones (con más atributos).
- menor cantidad de fusiones ("joins").
- menor cantidad de claves externas o secundarias ("foreign keys") para ser actualizadas.
- menor necesidad de actualización de índices.
- eliminación de problemas de "integridad referencial".

Desventajas

- mayor dificultad en la comprensión de la base de datos por los usuarios.
- mayor complejidad en los programas de tratamiento de la información.
- pérdida de independencia de datos.
- información redundante.
- mayor cantidad de entrada/salida para realizar una actualización global de la información redundante.

En síntesis, las relaciones se normalizan para facilitar las consultas a la base de datos y el mantenimiento de los programas de la aplicación.

Un pequeño aumento en el tiempo de procesamiento pareciera ser un precio reducido a pagar por el ahorro de meses-hombre de esfuerzo de desarrollo y mantenimiento de las aplicaciones.

Además, la desnormalización también paga un precio en cuanto a performance. Cada violación de la normalización afecta negativamente a cada uno de los procesos que cambian valores en los datos, y la lógica para el tratamiento de datos desnormalizados debe ser incluida en cada programa y por lo tanto está sujeta a mantenimiento.

6.7. Facilidades de Optimización Lógica Provistas por el Sistema

El sistema no provee ayudas para la optimización Lógica y Física de una Base de Datos relacional ya que se asume que se implementará el esquema normalizado (salida normal del diseño con DBAID), es decir que se asume que el esquema Lógico coincidirá con el esquema normalizado proveniente del diseño conceptual.

La implementación del esquema normalizado presenta importantes ventajas, que hacen que resulte conveniente implementar la gran mayoría de las bases de datos en su forma normal.

Sin embargo el diseñador puede, mediante el sistema, representar en el modelo conceptual las diferentes alternativas de optimización lógica que analizáramos previamente, las cuales una vez introducidas en el modelo conceptual serán traducidas al modelo relacional e incorporadas al esquema de la base de datos.

La introducción de grupos repetitivos no puede ser representada en el modelo conceptual, por lo que la misma debe hacerse en tiempo de generación del esquema final (para los DBMS no relacionales para los que DBAID tiene módulos de generación opcionales), o modificando el esquema generado por el sistema.

7. Síntesis de las Etapas de Diseño

Para el diseño de una base de datos relacional el diseñador debe seguir los pasos indicados a continuación, recordando que el proceso de diseño es iterativo e incremental, por lo que muchas veces deberá volver atrás algunos pasos para refinar el diseño.

1. Obtenga las Vistas de Usuario y de Contexto para el ambiente de aplicación para el cual desea construir la base de datos.
2. Para cada vista extraiga y tipifique conceptos y vinculaciones utilizando el sistema.
3. Mediante las funciones de Verificación y Visualización determine errores y ambigüedades. Corrija el modelo de acuerdo con las indicaciones del sistema.
4. Mediante las funciones de Visualización verifique que se satisfagan todos los requerimientos de información expresados en las Vistas de Usuario.
5. Elimine del modelo todos los conceptos que no intervienen en la satisfacción de las Vistas de Usuarios ni contribuyen al cierre de la estructura de datos.
6. Realice el análisis de estabilidad del modelo e introduzca futuro en el mismo en todos los casos en que ello resulte conveniente.
7. Con la ayuda del sistema verifique y/o defina la Cardinalidad de los eventos del modelo.
8. Genere los comandos requeridos para la implementación física de la base de datos.
9. Someta los comandos al RDBMS y obtenga la base de datos física.
10. Introduzca datos de prueba en la base de datos. Verifique nuevamente que se puedan satisfacer todos los requerimientos de las Vistas de Usuario utilizando el lenguaje de consulta del RDBMS.
11. Ejercite la base de datos verificando que se satisfagan los requerimientos de performance. Utilizando las facilidades de optimización del RDBMS efectúe todos los ajustes necesarios.
12. En caso de que los requerimientos de performance no sean satisfechos, considere la conveniencia de introducir optimizaciones lógicas en el modelo conceptual (ver "6. Diseño Lógico y Físico").

8. Adaptación a Manejadores de Bases de Datos no Relacionales

Para la implementación física de las bases de datos diseñadas mediante DBAID, éste genera los comandos DDL (Data Definition Language) requeridos por el manejador de bases de datos (DBMS).

La implementación "nativa" de DBAID es para manejadores de bases de datos relacionales basados en el lenguaje SQL. El sistema incluye módulos opcionales para la generación de comandos DDL para otros manejadores de bases de datos (la lista de módulos disponibles se encuentra en el archivo READ.ME del sistema).

Para aquellos DBMSs para los que no existen actualmente módulos de generación se incluyen a continuación algunas pautas para la adaptación del esquema relacional.

Consideraremos la adaptación a :

- DBMSs de tipo jerárquico.
- DBMSs de tipo red.
- DBMSs seudorelacionales.
- Conjuntos de Archivos Secuenciales.

8.1. Para DBMS de Tipo Jerárquico

El esquema conceptual, cuya estructura es una red, debe ser descompuesto en un conjunto de árboles.

Esta transformación no es única (existen varias transformaciones posibles) por lo que es indispensable seleccionar la más conveniente para soportar una mezcla de procesos dada. El conjunto total de los accesos requeridos por la mezcla de procesos prevista determinará la descomposición óptima.

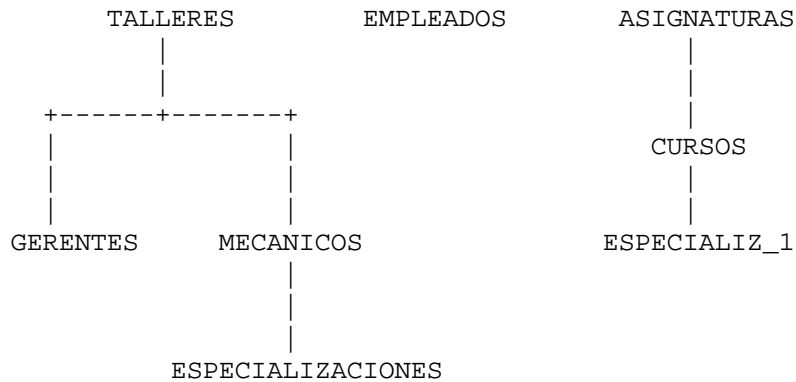
Se establecen las siguientes correspondencias entre elementos del modelo relacional y del modelo jerárquico:

Modelo Relacional	Modelo Jerárquico
Relaciones	Segmentos
Atributos	Campos (Fields)

Normalmente se define un tipo de segmento por cada Relación del modelo relacional (salvo que por razones de seguridad sea necesario "segmentar segmentos").

Para la construcción del modelo jerárquico resulta de gran utilidad la función de graficación del sistema que muestra las Relaciones en el modelo relacional, sus Atributos y las vinculaciones lógicas entre las Relaciones.

A continuación se incluye como ejemplo una transformación posible del esquema para el "Caso Talleres". La siguiente figura muestra las vinculaciones lógicas entre las relaciones que componen el esquema jerárquico:



Los campos de los diferentes segmentos son (la notación "[xxxxxx]" denota la presencia de un segmento cuyo contenido se describe más abajo):

TALLERES (TALLER, DENOMINACION_TALLER, DOMICILIO_TALLER
[GERENTES], [MECANICOS])

GERENTES (EMPLEADO)

MECANICOS (EMPLEADO, [ESPECIALIZACIONES])

ESPECIALIZACIONES (ESPECIALIZACION, CURSO,
CALIFICACION)

EMPLEADOS (EMPLEADO, NOMBRE_EMPLEADO,
DOMICILIO_EMPLEADO)

ASIGNATURAS (ASIGNATURA, DENOMINACION_ASIGNATURA,
DURACION_CURSO, [CURSOS])

CURSOS (CURSO, FECHA_CURSO, [ESPECIALIZ_1])

ESPECIALIZ_1 (ESPECIALIZACION)

En el caso de los DBMS de IBM denominados IMS y DL/1 el soporte de la estructura de datos está dado por las denominadas "bases de datos físicas".

Una base de datos jerárquica lógica es un superesquema, o vista, de un conjunto de bases de datos físicas (también jerárquicas). Las bases de datos físicas permiten representar jerarquías con precedencia posicional (los segmentos definidos "más arriba" o "más a la izquierda" en la estructura jerárquica son accedidos más rápidamente y más eficientemente).

Es posible "entrar" a las estructuras a través de los segmentos "raíz" (topes de las jerarquías) y de índices secundarios (supuesto asumido en el diagrama anterior). Se establecen tantas bases de datos físicas como segmentos raíz.

En caso de Relaciones que dependen lógicamente de más de una Relación (por ejemplo relaciones que representan eventos) el segmento correspondiente debe ubicarse en la base "física" que provea la vía de acceso más eficiente (siempre que esté unido al segmento raíz por una cadena de vinculaciones directas).

Las reglas de precedencia posicional hacen que las diferentes vías jerárquicas (trayectorias de acceso) no sean de performance equivalente, por lo que la ubicación de un tipo de segmento en la jerarquía resultará dictada por la frecuencia con que es accedido.

8.2. Para DBMS de Tipo Red

Para la construcción del modelo en red resulta de gran utilidad la función de graficación del sistema que muestra las Relaciones en el modelo relacional, sus Atributos y las Vinculaciones lógicas entre las Relaciones.

Los DBMS de tipo red permiten almacenar con eficiencia estructuras en la forma canónica, si bien las especificaciones del DBTG (Data Base Task Group) del CODASYL permite que los registros de las bases de datos contengan grupos repetitivos.

El soporte de la estructura de datos, en la mayoría los DBMS de este tipo, es a través de encadenamientos de registros denominados "SETS".

Se establecen las siguientes correspondencias entre elementos del modelo relacional y del modelo de red:

Modelo Relacional	Modelo de Red
Relaciones	Tipos de Registros
Atributos (vinculaciones lógicas entre relaciones)	Items de Dato Sets

Es posible establecer un ordenamiento (clasificación) de los registros dentro de una cadena (SET) de manera de facilitar el acceso a los datos para los procesos más frecuentes o más críticos. También es posible regular la "vecindad física" de los diferentes tipos de registros de manera de acortar las trayectorias de acceso más frecuentemente utilizadas.

Para nuestro "Caso Talleres", si aceptamos que cada relación es un tipo de registro y cada atributo un ítem de datos en el modelo de red, la adaptación del esquema relacional consiste en definir los diferentes sets a partir de las vinculaciones lógicas existentes entre las relaciones.

Cada set definido en el modelo de red recibe un nombre y especifica un tipo de registro como "DUEÑO" y otro tipo de registro como "MIEMBRO". Se indica para cada tipo de registro el ítem de dato que interviene en la vinculación.

Así en nuestro caso tendríamos:

Nombre del SET	DUEÑO	MIEMBRO
TALL_GERE	TALLERES (TALLER)	GERENTES (TALLER)
TALL_MECA	TALLERES (TALLER)	MECANICOS (TALLER)
EMPL_GERE	EMPLEADOS (EMPLEADO)	GERENTES (EMPLEADO)
EMPL_MECA	EMPLEADOS (EMPLEADO)	MECANICOS (EMPLEADO)
ASIG_CURS	ASIGNATURAS (ASIGNATURA)	CURSO (ASIGNATURA)
CURS_ESPE	CURSOS (CURSO)	ESPECIALIZACIONES (CURSO)
MECA_ESPE	MECANICO (EMPLEADO)	ESPECIALIZACIONES (MECANICO)

8.3. Para DBMS Seudorelacionales

Estos DBMSs reciben esta denominación por su parecido a los manejadores relacionales. La diferencia con estos últimos estriba en que admiten tablas que contienen grupos repetitivos, es decir tablas que no respetan la primera forma normal. De esta manera proveen un mecanismo de optimización lógica (ver "6.5.2.3. Desnormalizando Relaciones").

Para proveer acceso directo al contenido de los grupos repetitivos estos DBMSs permiten la creación de "índices secundarios" sobre atributos (y concatenaciones de atributos) de los grupos repetitivos.

Mediante manejadores de bases de datos es posible implementar el esquema relacional en la forma canónica y también introducir desnormalizaciones como las mostradas en la sección "6.5.2.3. Desnormalizando relaciones". También es posible implementar esquemas jerárquicos como el mostrado en la sección "8.1. Para DBMS Jerárquicos".

8.4. Para Archivos Secuenciales

El soporte de información puede ser también un conjunto de archivos secuenciales. Es posible almacenar estructuras estableciendo una correspondencia entre Relaciones y Archivos físicos.

Al igual que en los RDBMS las vinculaciones lógicas entre las relaciones residen en la lógica de los programas de tratamiento.

9. Resumen de las Principales Características del Sistema

Para finalizar deseamos hacer un resumen de las principales características del sistema y de la metodología utilizada. Algunas de las características importantes son:

FACILIDAD DE COMPRESION: la metodología está basada en un conjunto natural mínimo de elementos cuya comprensión es intuitiva. Los elementos semánticos, que distinguen a esta metodología de otras, contribuyen a facilitar su comprensión. La cantidad de reglas de diseño es mínima y se recurre a un mínimo de elementos artificiales, extraños al nivel conceptual, que es el nivel que prevalece en la metodología.

FACILIDAD DE UTILIZACION: la metodología parte de vistas expresadas en lenguaje natural, lo cual constituye la tendencia de evolución de las metodologías más modernas. El sistema de diseño es interactivo, y permite el diseño incremental del modelo de datos. Es muy fácil iterar en el diseño, refinando progresivamente el modelo. El diseño de modelos simples puede ser encarado directamente por el usuario final. DBAID posee un extenso sistema de ayuda en contexto. El diseñador recibe indicaciones sobre cómo actuar en aquellos casos en que el sistema detecta errores y ambigüedades durante el diseño.

TIPO SINTETICO: La metodología es de tipo sintético, la estrategia de modelización es "bottom-up" e incremental. Partiendo de conceptos y vinculaciones se sintetiza mediante el sistema una estructura conceptual y a partir de ella se obtiene automáticamente el modelo relacional. Es por consiguiente una metodología de dos niveles de abstracción.

Las metodologías de diseño de bases de datos "bottom-up" resultan convenientes para el diseño de bases de datos muy complejas debido a que, a diferencia de las metodologías "top-down", no requieren la realización de abstracciones (modelizaciones) mentales previas de mayor nivel.

ORIENTACION A LA INFORMACION: La perspectiva de información recibe prioridad frente a la perspectiva de procesamiento (ésta última aparece recién en la etapa de diseño lógico).

POTENCIA: El sistema permite manejar modelos de gran complejidad con facilidad y seguridad. El diseño asistido por computadora asegura la calidad de los resultados en tiempos menores. Además de cooperar en la modelización, el sistema ayuda a diagnosticar y corregir errores. La experiencia demuestra que las facilidades de verificación resultan de importancia fundamental.

ES COMPLETA: Tanto la metodología como el sistema cubren todos los pasos del diseño conceptual y del diseño lógico de la base de datos. El diseñador no se ve obligado a realizar extensiones "ad-hoc" a la metodología.

UTILIZA SEMANTICA: En toda metodología se debe partir de datos correctos. En caso contrario el diseño puede volverse extremadamente dificultoso. Los elementos semánticos requeridos por la metodología empleada permiten filtrar la incorporación de elementos erróneos en el modelo. El hecho de captar elementos semánticos minimiza la posibilidad de tener falsas percepciones de la realidad, que suelen ser consecuencia de la barrera de lenguaje entre usuarios legos en materia de bases de datos y diseñadores que desconocen el área de aplicación.

RIGUROSIDAD: Las reglas y etapas de diseño son suficientemente rigurosas como para asegurar diseños correctos y repetibles. La metodología promueve el tratamiento objetivo del problema por lo que diferentes diseñadores producirán resultados similares. El impacto de factores subjetivos resulta minimizado. Debido a que el sistema captura elementos semánticos, no existe ambigüedad en la interpretación de un modelo.

NORMALIZACION DE RELACIONES: Al incorporar incrementalmente conocimiento en la forma de predicados binarios (que representan la dependencia funcional entre conceptos) las Relaciones resultan generadas naturalmente en forma "normalizada" (las desnormalizaciones presentes en algunas bases de datos fueron introducidas artificialmente ya que no surgen en forma natural de la observación de la realidad).

VISIBILIDAD: El sistema tiene una extensa gama de facilidades de visualización del modelo de datos, que incluye un módulo de representación gráfica.

DOCUMENTACION AUTOMATICA: Uno de los beneficios del sistema es la posibilidad de obtener información consistente durante procesos de diseño que evolucionan muy rápidamente.

EXTENSIBILIDAD: La experiencia ha demostrado que, dada la naturaleza modular de la metodología y del sistema, es posible extender y refinar ambos con facilidad, sin que ello implique complicar su comprensión y utilización.

ADAPTABILIDAD: El sistema de ayuda en contexto de DBAID reside en un conjunto de archivos accesible por el diseñador permitiéndole introducir cambios para adecuarlo a sus necesidades.

INTELIGENCIA: El sistema con el fin de resolver adecuadamente determinadas situaciones de diseño puede admitir transitoriamente información ambigua. El manejo correcto de ambigüedades constituye una de las características distintivas de los sistemas inteligentes.

NATURALIDAD: El diseñador interactúa con el sistema en términos del ambiente de aplicación y del lenguaje natural.

10.1. ANEXO "A" Lenguaje de Consulta Casi-Natural

DBAID provee un lenguaje de consultas casi-natural (castellano restringido) para realizar consultas sobre el contenido del modelo conceptual.

El lenguaje reconoce un conjunto de "entidades" y de "asociaciones" entre dichas entidades.

Por ejemplo, en la consulta siguiente:

"Deme microvistas con concepto empleado".

"microvistas" y "concepto" son entidades reconocidas por el lenguaje, y "con" es una asociación también reconocida.

"empleado" es un concepto del modelo conceptual activo en ese momento.

Mediante el lenguaje de consulta es posible obtener información sobre las siguientes entidades y sus asociaciones:

ENTIDAD	SINONIMO UTILIZABLE
concepto(s)	
evento(s)	
rol(es)	
<roles tales como:> agente(s) beneficiario(s) objeto(s) tiempo(s) ..etc...	
dominio(s)	tipificación(es), definición(es), tipo(s).
vinculación(es)	
microvista(s)	
atributo(s)	propiedad(es)

Los siguientes son ejemplos de consultas válidas:

- conceptos.
- lísteme eventos.
- lísteme eventos con mecánico.
- lísteme eventos con beneficiario mecánico.
- conceptos con dominio persona.
- conceptos en la microvista U005.
- atributos de empleado.
- eventos con beneficiario.
- microvistas con eventos.
- beneficiario en evento especialización.
- beneficiario en especialización.
-

El esquema de consultas admisibles tiene la forma (más adelante se incluye la gramática expresada en notación BNF):

```
CONSULTA :- ENTIDAD
CONSULTA :- ENTIDAD ASOCIACION ENTIDAD CONCEPTO
CONSULTA :- ENTIDAD ASOCIACION CONCEPTO
CONSULTA :- ENTIDAD ASOCIACION ENTIDAD
```

donde la ASOCIACION ("con", "en", "tiene", "de", etc.) vincula una ENTIDAD ("evento", "concepto", etc.) con otra o con un CONCEPTO del modelo ("especialización", "empleado", etc.).

También es posible realizar "consultas anidadas" con cualquier nivel de profundidad tales como las siguientes:

```
CONSULTA :- ENTIDAD ASOCIACION CONSULTA
```

- atributos de beneficiario de especialización.
- agente en evento con beneficiario mecánico.
- dominio de conceptos con dominio persona.
- microvistas con conceptos con dominio organización.
- conceptos en microvistas con conceptos con dominio organización.
- dominio de conceptos en microvistas con conceptos con dominio organización.

También es posible realizar consultas de tipo "y/o" como las siguientes:

- beneficiarios en eventos o agentes en eventos.
- evento con mecanico y evento con concepto gerente.
- eventos con concepto mecánico y con concepto curso.
- eventos con concepto mecánico y concepto curso.
- eventos con concepto mecánico y con curso.
- eventos con concepto mecánico y curso.
- eventos con concepto mecánico y con curso y calificación.

Gramática del Lenguaje de Consulta Expresado en BNF

La sintaxis del lenguaje de consulta expresada en notación BNF es la siguiente:

```

<consulta>      ::- <cons_1>      |
                  <ent>           |
                  <cons_conj>     |

<cons_1>        ::- <ent> <asoc> [<ent>] <conc>  |
                  <ent> <asoc> <ent> [<conc>]  |
                  <ent> <asoc> <cons_1>        |

<ent>           ::- (entidad definida en el cuadro
                  anterior)

<asoc>          ::- con | en | tiene | de ....

<conc>          ::- (concepto incluido en el modelo
                  conceptual, por ejemplo "empleado"
                  y "especialización")

<cons_conj>     ::- <cons_1> <conj> <cons_2>

<conj>          ::- y | o

<cons_2>        ::- <cons_1>      |
                  <cons_conj>    |
                  <cons_conj_1>   |
                  <cons_3>        |

<cons_conj_1>   ::- <cons_3> <conj> <cons_3>      |
                  <cons_3> <conj> <cons_conj_1>  |

<cons_3>        ::- <asoc> <ent> <conc>          |
                  <asoc> <conc>                  |
                  <ent> <conc>                    |
                  <conc>                           |

```

11. Referencias Bibliográficas

- Charniak E., Wilks Y. [1976] "Computational Semantics".
Fundamental Studies in Computer Science, Vol.4,
North-Holland. Amsterdam. 1976.
- Chen. P.P. [1976] "The Entity-Relationship Model: Toward an
unified view of data". ACM. Trans. Database Syst. 1.
- Codd E.F. [1970]. "A Relational Model of Data for large
shared data banks". Commun. ACM. 13.
- Codd E.F. [1979]. "Extending the relational model to capture
more meaning". ACM Trans. Database Syst. 4.
- Codd E.F. [1982]. "Relational Database: A Practical
Foundation for Productivity". Commun. ACM 25.
- Date C.J. [1981]. "An introduction to Database Systems".
Addison-Wesley. Reading, MA.
- Dolder H. [1984]. "Diseño Conceptual e Implementación de
Bases de Datos". Data S.A.. Buenos Aires.
- Dolder H. [1987]. "Diseño de Bases de Datos Utilizando
Conceptos y Técnicas de Inteligencia Artificial". Eudeba.
Buenos Aires.
- Fillmore C.J. [1968]. "The case for case" en "Universals in
Linguistic Theory". Holt, Rinehart and Winston. New York.
- Hull R., King R. [1987] "Semantic Database Modeling: Survey,
Applications, and Research Issues". ACM Computer Surveys.
Vol 19. Number 3.
- Kent W. [1983] "A Simple Guide to Five Normal Forms in
Relational Database Theory", Commun. ACM 26.
- Lagenfors B. [1974] "Informations Systems", Proc. IFIP Congr.
74. North-Holland. Amsterdam.
- Wallace M. [1984]. "Communicating with Databases in Natural
Language". John Wiley & Sons. New York.
- Smith H.C. [1985] "Database Design: Composing Fully
Normalized Tables from a Rigorous Dependency Diagram",
Commun. ACM 28.

Smith J.M., Smith D.C.P. [1977]. "Database Abstractions: Aggregation and Generalization". ACM Trans. Database Syst. 2.

Tsichritzis D.C., Lochovsky F.H. [1982] "Data Models" Prentice-Hall. New Jersey.

Whitener T. [1988] "Building Database Stability". Database Programming & Design. June 1988.

Wilmot R.B. [1984] "Foreign Key Decrease Adaptability of Database Design", Commun. ACM 27.

www.holder.com