GENERADOR AUTOMATICO DE APLICACIONES DE BASES DE DATOS MEDIANTE LA DESCRIPCION GRAFICA DEL DIAGRAMA DE FLUJO DE DATOS Y UN MODELADO ORIENTADO A OBJETOS DE LOS MISMOS

TESIS

Que para obtener el grado de

MAESTRO EN INGENIERÍA ELÉCTRICA

presenta

Cesar Jerónimo López Camacho

MC Luis Rubén Rusiles Zamora

Director de Tesis

Universidad Michoacana de San Nicolás de Hidalgo

Octubre 2007

Agradezco a

A Dios, por tener la familia que tengo

A mis padres

Arnulfo López Peña, Yolanda Camacho Mateo

y hermanas

Perla Yolanda y Silvia Naty

por su constante e infinito cariño,

A mis maestros. Especialmente al MC. Luis R. Rusiles Z. por su valiosa asesoría,

A la Universidad Michoacana de San Nicolás de Hidalgo por el apoyo otorgado para realizar estos estudios.

Resumen

El sistema “GAABAD: Generador Automático de Aplicaciones de Bases de Datos mediante la descripción gráfica del diagrama de flujo de datos y un modelo orientado a objetos de los mismos” es el resultado de un proyecto cuyo propósito es satisfacer la necesidad de construir aplicaciones de bases de datos por usuarios sin conocimientos o habilidades en programación y bases de datos.

El GAABAD programa automáticamente las aplicaciones de bases de datos en una forma sencilla y fácil. El usuario introduce los diagramas DFD (de las siglas en inglés, Data Flow Diagram: Diagrama de Flujo de Datos) y OODM (de las siglas en inglés, Object Oriented Data Model: Modelo de Datos Orientado a Objetos) a partir de una GUI (de sus siglas en inglés, Graphical User Interface: Interfaz Gráfica de Usuario). Los elementos de ambos diagramas son líneas, cuadros y círculos que representan objetos y sus relaciones; de esta forma es más fácil para el usuario describir las aplicaciones que desea construir.

La información de los diagramas tanto del DFD como del OODM se almacenan en archivos de texto para su posterior uso. Así, esta información puede también ser consultada o actualizada a través de un editor de texto.

El GAABAD ha sido programado en Java y construye aplicaciones para los sistemas operativos Linux y Windows. Para las aplicaciones en el Sistema Operativo Linux se generan programas en Php que usan el DBMS (Sistema Manejador de Bases de Datos) MySQL, y para el Sistema Operativo Windows se generan programas en Asp que usan el DBMS SQL Server.

Abstract

"GAABAD: Generador Automático de Aplicaciones de Bases de Datos mediante la descripción gráfica del diagrama de flujo de datos y un modelo orientado a objetos de los mismos" is the outcome of a project aiming to satisfy the need of building databases application by users who do not have programming skills or databases knowledge.

GAABAD builds automatically databases applications programs in a straighforward and simple way. The user introduces throught a GUI (Graphical User Interface) the DFD (Data Flow Diagram) and the OODM (Object Oriented Data Model). The elements of both diagrams are lines, squares and circles representing objects and their relations, in this way it is easier for the user to describe the applications that he wants to build.

The information of the DFD and OODM diagrams is stored in text files for handling it afterwords. So, this information may also be consulted or updated using a text editor.

GAABAD has been programmed in Java and builds applications for Linux and Windows Operating Systems. For Linux OS applications the programs are generated in Php for using MySQL DBMS (Database Base Management Systems); for Windows OS in Asp for using SQL Server DBMS.

Contenido

Dedicatoria III
Resumen V
Abstract VII
Contenido VIII
Lista de Figuras XI
Lista de Tablas XIII

1. Introducción 1
   1.1. Motivación 1
   1.2. Objetivo de la tesis 2
      1.2.1. Alcances 2
      1.2.2. Retos 3
      1.2.3. Contribuciones 3
   1.3. Descripción de la tesis 4

2. Antecedentes 5
   2.1. Toolkit for Conceptual Modelling (TCM) 6
   2.2. Visual Query Language (VQL) 6
   2.3. Proyecto Tioga 7
   2.4. DeZign for databases 7
   2.5. Unified Modelling Language (UML) 8
   2.6. Conclusiones 8

3. Modelos Formales Utilizados 11
   3.1. Los modelos 11
   3.2. Modelo del Diagrama de Flujo de Datos 12
      3.2.1. Definición 12
      3.2.2. Características 13
      3.2.3. Modelo Formal 14
      3.2.4. Síntesis 15
   3.3. Modelo de las Bases de Datos Relacionales 15
      3.3.1. Definición 16
      3.3.2. Características 16
      3.3.3. Modelado Formal 18

3.3.4. Síntesis 22
3.4. Modelo de las Bases de Datos Orientadas a Objetos 22
3.4.1. Definición 23
3.4.2. Caraterísticas 23
3.4.3. Modelado Formal 28
3.4.4. Síntesis 31
3.5. Sistema 31
3.6. Conclusiones 34

4. Modelo de la Interfaz Visual 35
4.1. Ejemplo 35
4.2. El sistema 36
4.3. La Interfaz Visual 36
4.3.1. La pantalla principal 37
4.3.2. Bases de datos 38
4.3.3. Flujo de la información 40
4.4. El archivo de texto 40
4.4.1. Bases de datos 41
4.4.2. Flujo de la información 43
4.5. Modelado 45
4.6. Conclusiones 46

5. Generación automática de Código 47
5.1. Bases de Datos 47
5.2. Lenguaje de Consulta 49
5.3. Scripts 51
5.4. Conclusiones 56

6. Integración y pruebas del Sistema 57
6.1. Bases de Datos 57
6.2. Consulta 58

7. Conclusiones 61

Referencias 63

Glosario 67

Lista de Figuras

3.1. Modelos utilizados: OODM, DFD y RDB 12

4.1. Personas en ambiente escolar: Profesores y Trabajadores 36
4.2. La pantalla principal 37
4.3. Selección entre OODM (izq) y DFD (der) 37
4.4. Creación de las bases de datos 38
4.5. Creación de los flujos de información 38
4.6. Atribución de las bases de datos 39
4.7. Composición por agregación 39
4.8. Diseño del nombre del proceso 40

6.1. Atribución 57
6.2. Inclusión por especialización 58
6.3. Composición por asociación 59
6.4. Generación de consultas en scripts 60
6.5. Petición de una consulta 60
6.6. Resultado de una consulta 60

Lista de Tablas

3.1. Atribución 32
3.2. Inclusión por especialización 32
3.3. Inclusión por partición 32
3.4. Composición por Asociación (Conjunto) 33
3.5. Composición por Ordenamiento (lista) 33
3.6. Composición por Agregación (tupla) 34

4.1. Representación de la aplicación 45
4.2. Representación del texto 45
4.3. Coordenadas 46

Capítulo 1

Introducción

Las bases de datos son importantes puesto que permiten almacenar y explotar grandes cantidades de información. Su uso se extiende en diferentes áreas de aplicación.

La necesidad del manejo de bases de datos se satisface creando aplicaciones que las manipulen. Sin embargo, la construcción de estas aplicaciones generalmente es realizada por expertos en programación y bases de datos. Existe, por tanto, la necesidad de la construcción de aplicaciones por usuarios inexpertos. Debido a esta necesidad se desarrolló el "GAABAD: Generador Automático de Aplicaciones de Bases de Datos mediante la descripción gráfica del diagrama de flujo de datos y un modelo orientado a objetos de los mismos". Se desarrolló con el fin de proveer al usuario de modelos conceptuales simples que le permitan describir fácilmente sus aplicaciones; así como de una herramienta amigable para construir dichas aplicaciones.

En este capítulo se presentan las razones por las que se desarrolló el GAABAD, el objetivo que se persiguió, los retos que se enfrentaron así como las contribuciones aportadas por su desarrollo.

1.1. Motivación

Debido a la necesidad del manejo de bases de datos por usuarios sin conocimientos suficientes en la construcción de aplicaciones, surgió la idea de realizar un generador automático para facilitar el trabajo. El sistema debería de ser simple y de fácil manejo para el usuario inexperto. Por lo tanto, fue necesario buscar modelos sencillos que fueran entendibles y suficientemente intuitivos para este tipo de usuarios. Los modelos elegidos

fueron modelos matemáticos conceptuales, intuitivamente simples, a través de los cuales el usuario puede diseñar su sistema de información.

Otro aspecto importante era que la herramienta pudiera ser utilizada en diferentes plataformas sin necesidad de realizar cambios significativos en ella.

Además, que la especificación de los sistemas definidos por el usuario pudieran ser modificados tanto en un ambiente gráfico como en archivos de texto correspondientes.

Puesto que se pretendía que el usuario modelara fácilmente sus aplicaciones se usará un modelo semántico y no un modelo estructural como el de las bases de datos relacionales. Por lo que se implementó el uso de un modelo propio de bases de datos orientadas a objetos basado en el propuesto por Adolfo Antunes [Antunes, 1995].

Finalmente, se esperaba que las aplicaciones construidas por el sistema fueran fácilmente accesibles, por lo que se construye el sistema para que estas aplicaciones fueran manejadas por un navegador de Internet.

1.2. Objetivo de la tesis

Desarrollar una herramienta de programación automática de aplicaciones de bases de datos, donde el usuario describe su sistema de información usando un modelo intuitivamente simple y el sistema genera la aplicación para Web.

1.2.1. Alcances

Los alcances del GAABAD son los siguientes:

- La programación automática de aplicaciones básicas de bases de datos; es decir, prototipos de sistemas de información.
- El desarrollo de una interfaz gráfica de manejo intuitivo y fácil para el usuario, que permita introducir el modelo del sistema de información de manera simple.
- La implementación del modelo semántico en términos de bases de datos relacionales, dado que éstas son un estándar y, por lo tanto, las más comúnmente utilizadas.
- El uso de las aplicaciones producidas por el GAABAD en Web.

1.2.2. Retos

Para poder desarrollar el GAABAD, se enfrentaron los siguientes retos:

- Lograr integrar el modelo del diagrama de flujo de datos y el de las bases de datos orientadas a objetos para modelar las aplicaciones.
- Permitir al usuario construir el modelo de las aplicaciones de manera fácil e intuitiva a partir de una interfaz gráfica.
- Encontrar la manera de traducir el modelo orientado a objetos de las bases de datos propuesto al modelo relacional, usado por los gestores de bases de datos relacionales.
- Generar automáticamente los programas Web de las aplicaciones modeladas, permitiendo de esta manera que un usuario sin conocimientos previos de programación sea capaz de contruir una aplicación.

1.2.3. Contribuciones

Las contribuciones esenciales del GAABAD son:

- Una herramienta para el desarrollo rápido y seguro de soluciones básicas a sistemas de información por usuarios sin conocimientos de programación.
- Una interfaz gráfica de fácil manejo por el usuario que permite la descripción de aplicaciones en forma amigable e intuitiva, y el uso de los modelos de la aplicación en forma gráfica y en archivos de texto.
- El modelo propuesto orientado a objetos que es traducido a las bases de datos relacionales.
- El modelo del diagrama de flujos de datos que describe las acciones que realiza el sistema.
- La generación de sistemas de información para los sistemas operativos Windows y Linux.

1.3. Descripción de la tesis

La tesis está organizada de la siguiente manera: en el capítulo 2 se presentan proyectos relacionados con el GAABAD, tanto comerciales como de investigación, mostrando sus semejanzas y sus diferencias; en el capítulo 3 se describen los modelos utilizados para la definición de los sistemas de información, que son el modelo de Flujo de Datos y el modelo propuesto de las Bases de Datos Orientadas a Objetos, así como su interrelación; en el capítulo 4 se trata el modelo de la interfaz gráfica del sistema desarrollado; en el capítulo 5 se presenta la generación de código; en el capítulo 6 se documentan las pruebas realizadas; y finalmente, se presentan las conclusiones.

Capítulo 2

Antecedentes

Las aplicaciones de bases de datos requieren de un conjunto de habilidades y conocimientos para su construcción, es necesario conocer el lenguaje de programación de las aplicaciones y los lenguajes de definición, manipulación y control de las bases de datos. Además, el desarrollo de aplicaciones se vuelve una tarea que requiere mucho tiempo. Por estas situaciones, se han buscado soluciones a estos problemas, lo cual ha llevado a realizar sistemas que ayuden al ahorro tanto de tiempo como de esfuerzo.

En este capítulo se presentarán sistemas relacionados con el GAABAD, donde se describen y se analizan sus semejanzas y diferencias. Estos productos han sido desarrollados tanto con objetivos comerciales como de investigación. El sistema desarrollado en el Imperial College, es una herramienta CASE (de las siglas en inglés, Computer Aided Software Engineering: Ingeniería de Software Asistido por Computadora), llamado "Toolkit for Conceptual Modelling", y es utilizada para el modelado de DFD. En el sistema de la Universidad de Pennsylvania, llamado "Visual Query Language", se hacen consultas a bases de datos médicas usando una interfaz visual y fue programado en Java. El sistema de la Universidad de California en Berkeley, llamado "Tioga", es un ambiente de visualización de base de datos que combina un modelo de navegación sofisticado con una interfaz gráfica, donde los usuarios pueden crear sus propias visualizaciones. El sistema llamado "DeZign for Databases", es un desarrollo comercial donde se describe cómo modelar bases de datos utilizando los ERD (de las siglas en inglés, Entity Relationship Diagram: Diagramas Entidad Relación. El UML (de las siglas en inglés, Unified Modelling Language: Lenguaje Unificado de Modelado), es un lenguaje de modelado utilizado en la actualidad que consiste de notaciones y diagramas para modelar sistemas orientados a objetos.

2.1. Toolkit for Conceptual Modelling (TCM)

El TCM [Eisenback,2003] es una herramienta para representar especificaciones de sistemas de software en forma de diagramas, tablas y árboles. Ofrece distintos tipos de editores para tareas específicas, entre ellos, editores para diagramas entidad relación y diagramas de flujo de datos. El editor de diagramas entidad relación (TERD, por sus siglas en inglés) para el modelado de diagramas entidad relación y el editor de diagramas de flujo de datos (TDFD, por sus siglas en inglés) para el modelado de diagramas de flujo de datos. Estos editores usan el modelo y la representación gráfica estándar para generar los modelos conceptuales correspondientes. En el GAABAD el modelo del DFD es muy similar, a diferencia que sólo acepta cuatro tipo de acciones y no utiliza el modelo entidad relación sino el modelo orientado a objetos bajo una propuesta en específico. Por otra parte, el TCM no genera código para los distintos sistemas de software que se modelan, algo que sí hace el GAABAD.

Esta herramienta está diseñada desde el punto de vista del diseñador/programador, el cual debe tener conocimientos acerca de C++, X/Motif y Unix; a diferencia de GAABAD, donde el usuario final no requiere de conocimientos acerca de algún sistema en especial. El sistema está diseñado para poder utilizarse bajo cualquier sistema Unix con X Windows y Motif; el GAABAD, dado que su implementación no requiere de algún conocimiento acerca de algún lenguaje, es fácilmente instalable en sistemas operativos tales como Windows y Linux; en Linux requiere que se tenga instalado X Windows.

2.2. Visual Query Language (VQL)

El VQL [Davidson,2005] es un lenguaje de consultas visuales que provee herramientas intuitivas para el usuario; tal como ocurre con el GAABAD. Este sistema ha desarrollado un lenguaje de consulta llamado CPL (Collection Programming Language) que se utiliza para hacer consultas a bases de datos biomédicas, pero su uso se puede extender a entradas de datos diferentes. El esquema gráfico se traduce a este lenguaje de consulta para obtener las consultas deseadas, en el GAABAD se traduce al lenguaje SQL.

El desarrollo de este sistema ha sido implementado utilizando la biblioteca Java Foundation Class. Esta biblioteca ha sido utilizado también por el GAABAD.

2.3. Proyecto Tioga

El proyecto Tioga [Tioga, 1999] desarrolló un sistema usando el paradigma de "cajas y flechas", donde le permite al usuario crear y navegar de forma visual las tablas de las bases de datos; de forma similar, el GAABAD utiliza información visual pero exclusivamente en el diseño de las bases de datos, mas no propiamente en la información. Consta de dos componentes: el espacio de trabajo en el cual puede ser utilizado un gráfico relativo al sistema que se quiere desarrollar, dentro del cual se pueden hacer acercamientos para un desarrollo en detalle; y el sistema de navegación propiamente. El GAABAD utiliza un sistema de navegación similar a éste, no existe la variación del gráfico de fondo. Además de los iconos para el modelo de las bases de datos, también cuenta con iconos para agregar título y leyendas; el GAABAD no utiliza iconos para agegar títulos ni leyendas, sólo los referentes al modelado de bases de datos.

Tioga fue desarrollado e implementado en Unix con conexiones a bases de datos Postgress; por lo tanto, desarrolla sistemas para bases de datos relacionales y realiza las consultas a ellos desde el mismo sistema. El GAABAD modela los sistemas utilizando el paradigma orientado a objetos y lo traduce al modelo relacional; no se conecta a la base de datos; genera los programas para su posterior utilización en el gestor de bases de datos correspondientes, ya sea MySQL o SQL Server.

2.4. DeZign for databases

Dezign for Databases [Datanamic, 1999] es una herramienta de desarrollo de bases de datos que usa el diagrama entidad relación. Soporta esquemas de entidades y relaciones y genera automáticamente esquemas SQL para la mayoría de las aplicaciones comerciales de bases de datos. La idea de la generación de esquemas SQL de este producto ha sido implementada en el GAABAD.

El Dezign soporta las siguientes bases de datos: Oracle, Interbase, IBM DB2, Sybase, MS Access, MS SQL Server, Paradox, dBase, Informix, SQL-Anyware, MySQL y PostgreSQL. A diferencia del Dezing, el GAABAD soporta MySQL y SQL Server.

Dezign cuenta con un editor de Diagramas Entidad Relación y exporta los resultados a archivos de texto; funcionamiento similar al GAABAD. El editor de Dezign utiliza las representaciones gráficas propias del modelo entidad relación; en cambio, el editor del

GAABAD utiliza representaciones gráficas del modelo orientado a objetos que se propone en este trabajo de tesis.

2.5. Unified Modelling Language (UML)

El UML [UML,2002] es un lenguaje de modelado de sistemas de software que cuenta con diferentes modelos, los cuales muestran diferentes aspectos de las entidades representadas. Utiliza un conjunto de notaciones y diagramas estándar para modelar sistemas orientados a objetos; el GAABAD, dado que su modelado es orientado a objetos pero se traduce al modelo relacional no sigue esta notación.

Organiza el proceso de diseño de tal forma que los analistas, clientes, desarrolladores y otras personas involucradas en el desarrollo del sistema lo comprendan.

Está compuesto por diversos elementos gráficos que se combinan para conformar diagramas. Debido a que es un lenguaje, cuenta con reglas para combinar esos elementos.

La finalidad de los diagramas es presentar diversas perspectivas de un sistema, a los cuales se les conoce como modelo, un modelo UML describe lo que hará el sistema, pero no dice cómo implementar dicho sistema.

El UML estandariza la notación orientada a objetos con el fin de tener una única notación, pero, debido a que el GAABAD no modela todos los elementos del modelo orientado a objetos y, además, se basa en el modelo de Adolfo Antunes [Antunes, 1995] no es factible adoptar este modelado para su desarrollo. El UML se puede usar para modelar dintintos tipos de sistemas: sistemas de software, sistemas de hardware y organizaciones del mundo real; de ellos el GAABAD modela sistemas de software, en específico, sistemas de información. El UML no construye la solución del sistema, la elección de qué metodología o proceso a usar queda abierta a que el modelador lo decida; el GAABAD genera la solución del sistemas a través de la generación de programas para ser utilizado desde la Web.

Cabe destacar que GAABAD, a difencia de UML, genera código para el sistema; esto es, no nada mas lo modela, sino genera el código de implementación.

2.6. Conclusiones

Se han presentado algunas soluciones al diseño de bases de datos, y el cómo han ido evolucionando a través del tiempo. Estas soluciones se han desarrollado por distintas univer-

sidades y empresas. También se presentaron las ideas que fueron integradas al GAABAD. En posteriores capítulos se detallarán el diseño del GAABAD y el cómo estas ideas fueron integradas.

Capítulo 3

Modelos Formales Utilizados

El GAABAD permite que el usuario modele conceptualmente las aplicaciones de bases de datos a partir de su descripción usando el DFD y el OODM propuestos, y posteriormente mapea el OODM al RDBM (de las siglas en inglés, Relational Data Bases Modelling, Modelo de Bases de Datos Relacionales).

En este capítulo se presentarán estos modelos y la manera en que han sido implementados.

3.1. Los modelos

El OODM propuesto es una conceptualización parcial del modelo propuesto por Adolfo Antunes en su tesis doctoral "An axiomatic approach to Deductive Object-Oriented Databases" [Antunes, 1995], que toma solamente algunos conceptos que consideramos suficientes para construir prototipos de una gran por ciento de aplicaciones de bases de datos.

El DFD permite el modelado de los programas que accederán a la base de datos, a los elementos que estos lo conforman así como las restricciones que tendrán para su acceso; este modelo del DFD también está simplificado, ya que solamente usa conceptos con los cuales se puede insertar, borrar, modificar y consultar las bases de datos; esto es, toma las bases del DFD estándar, pero no realiza todas las funciones de éste, es decir, un modelo DFD propuesto específicamente para el desarrollo de GAABAD.

Las bases de datos en las que se utilizan los programas generados por el GAABAD son RDB (de las siglas en inglés, Relational Data Bases: Bases de Datos Relacionales); tales como MySQL en el Sistema Operativo Linux y SQL Server en el Sistema Operativo

Windows; por lo tanto, el modelo conceptual diseñado en el GAABAD es necesario que sea traducido al modelo relacional.

La traducción al RDB es transparente para el usuario, la forma de realizarlo se ejemplica en la figura 3.1.

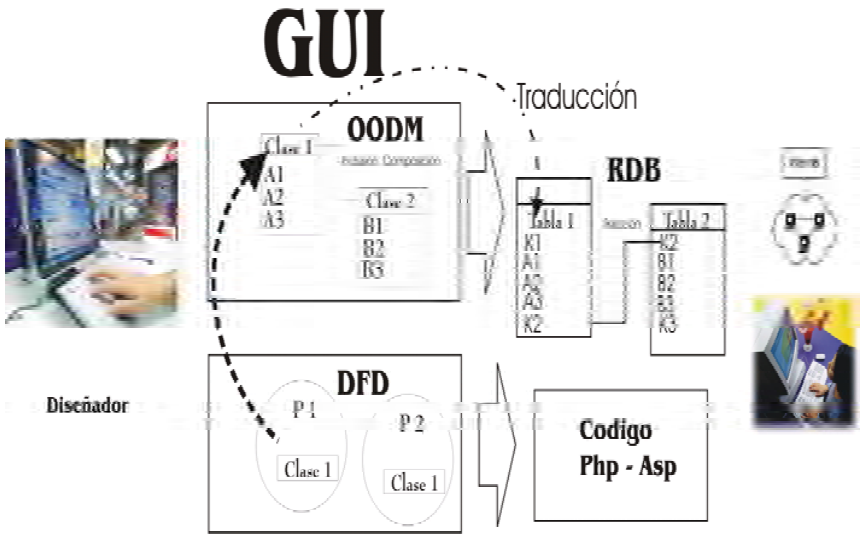

Figura 3.1: Modelos utilizados: OODM, DFD y RDB

3.2. Modelo del Diagrama de Flujo de Datos

El DFD tiene una utilidad especial, que es la generación del ambiente gráfico a partir del cual se maneja la información, tanto de la generación de las bases de datos como del sistema de información. Se utiliza este modelo debido a su facilidad y frecuente uso en el ambiente de los sistemas de información. Se presentarán los símbolos utilizados y su significado; así como la manera en que se implementa en GAABAD y la fusión que se hace entre el modelo de las bases de datos y del sistema de información.

3.2.1. Definición

El DFD estándar es un modelo que describe los flujos de datos, los procesos que cambian o transforman los datos en un sistema, las entidades externas que son fuente

o destino de los datos (y en consecuencia los límites del sistema) y los almacenamientos o depósitos de datos a los cuales tiene acceso el sistema, permitiendo así describir el movimiento de los datos a través del sistema. El DFD es una representación de todo el sistema. El DFD contempla el sistema de términos de sus componentes indicando los enlaces entre los componentes.

La técnica del diagrama de flujo de datos es una representación gráfica que permite definir entradas, procedimientos y salidas de la información, permitiendo así comprender los procedimientos existentes con la finalidad de optimizarlos, reflejándolos en el sistema propuesto [Berrios,2002]. La información se transforma a medida que fluye por un sistema basado en computadora. El sistema acepta entradas en una gran variedad de formas; aplica elementos de hardware, software y humanos para transformar la entrada en salida, y produce la salida en una gran variedad de formas.

En el DFD propuesto para el GAABAD la entrada es ingresada por el usuario y la salida son los programas generados tanto para la creación de las bases de datos como para su manipulación.

El DFD propuesto para el GAABAD se comporta de la siguiente manera:

- La entrada es el modelado tanto de las bases de datos como del flujo de la información a partir de la interfaz gráfica o de un archivo de texto.
- Los procedimientos son los diagramas del modelo orientado a objetos para modelado de aplicaciones de bases de datos.
- La salida son los archivos de programas para la manipulación de las bases de datos.

3.2.2. Características

El DFD tiene por **objetivo** representar gráficamente el sistema a nivel lógico y conceptual [Univ. Valladolid, 2000], ilustrando los componentes esenciales de un proceso y la forma en que interactuan. Esta técnica del DFD es útil porque representa gráficamente los límites del sistema de estudio, muestra el movimiento de los datos y la transformación de los mismos a través del sistema y facilita el mantenimiento del sistema.

Los elementos utilizados en el GAABAD para el uso de los DFD son los siguientes:

- **Proceso:** representado por un círculo que indica la acción aplicada a las bases de datos. El proceso muestra la acción realizada a partir de un verbo.

- **Flujo:** representado por una flecha, indica el movimiento de la información de las bases de datos hacia los procesos.

- **Almacenamiento:** representado por un rectángulo con dos líneas, es propiamente la base de datos construida.

- **Terminadores:** representado por un rectángulo simple, son las entidades externas que permiten que fluya la información; esto es, los privilegios de los usuarios finales de las aplicaciones desarrolladas.

3.2.3. Modelo Formal

A continuación se presenta el modelo formal de los DFDs propuesto para GAABAD. Este modelo es aportación del sistema, puesto que muestra las características de estos diagramas en función del sistema, para la implementación de las aplicaciones de bases de datos básicas.

Un DFD utilizado en GAABAD cuenta con los siguientes elementos:

A - acciones

T - terminadores

S - almacenamientos

P - procesos

F - flujos

Los terminadores tienen los privilegios para ejecutar las acciones, procesos.

Los almacenamientos se refieren a las bases de datos que contienen la información; estas bases de datos son las diseñadas bajo el modelo orientado a objetos (OODB). La fusión entre los dos diagramas se realiza en este elemento.

Como ya se ha mencionado, las acciones a las que se limita el proyecto son las siguientes: Consultar, Insertar, Modificar y Borrar. Formalmente, tenemos que:

$$A \in \{Consultar, Insertar, Modificar, Borrar\} \tag{3.1}$$

Para que se ejecute un proceso, es necesario saber quien tiene el privilegio para poder realizarlo así como saber de donde proviene la información; es decir, a partir de qué medio de almacenamiento se obtiene la información. Formalmente, podemos decir que:

$$P \subseteq A \times S \tag{3.2}$$

Los flujos unen los terminadores, con los procesos de los cuales tiene el privilegio de ejecutar. En esta propuesta los flujos representan los privilegios. Formalmente,

$$F \subseteq T \times P \tag{3.3}$$

Por lo tanto, el DFD se compone de Procesos, Flujos, Almancenamientos y Terminadores.

$$DFD = (P, F, S, T) \tag{3.4}$$

3.2.4. Síntesis

El GAABAD utiliza los DFD para modelar tanto los procesos como las bases de datos para la aplicación de bases de datos deseada. Los DFD han sido sintetizados en función del GAABAD; esto es, se basa en los DFD estándar, pero con características propias.

Los diagramas utilizan procesos, flujos y terminadores. Los *procesos* se representan por un círculo; realiza acciones específicas: insertar, consultar, borrar y modificar.

El *almacenamiento* está incluido en el proceso; esto es, al definir el proceso, se define también de donde proviene la información. Esta es la parte de vinculación entre el DFD y el OODB.

Todo proceso tiene un *flujo* de entrada; la entrada es el terminador. Una flecha liga los procesos con el terminador; no hay flechas de proceso a proceso.

Los *terminadores* definen los propietarios de los procesos; esto es, cada proceso pertenece a un terminador, y sólo se pueden ejecutar si pertenece a uno de ellos.

Una vez diseñado el modelo en su forma gráfica, se generan las bases de datos correspondientes, utilizando el modelo relacional el cual se verá en el siguiente apartado.

3.3. Modelo de las Bases de Datos Relacionales

Se presentará el modelo relacional utilizado para la creación de la solución de los sistemas modelados. Éste es usado a partir del modelo orientado a objetos que más adelante se verá. Se presentará su definición, características y el modelo.

3.3.1. Definición

Las bases de datos han tomado varias vertientes a lo largo de los años de su existencia; muchos conceptos de bases de datos se han definido, siendo los siguientes algunos de gran relevancia [Piattini, 1993]:

- Colección de datos interrelacionados almacenados en conjunto sin redundancias perjudiciales o innecesarias; su finalidad es servir a una aplicación o más, de la mejor manera posible; los datos se almacenan de modo que resulten independientes de los programas que los usan; se emplean métodos bien determinados para incluir nuevos datos y para modificar o extraer los datos almacenados.
- Colección o depósito de datos, donde los datos están lógicamente relacionados entre sí, tienen una definición y descripción comunes y están estructurados de una forma particular. Una base de datos es, también, un modelo del mundo real y, como tal, debe poder servir para toda una gama de usos y aplicaciones.
- Conjunto de datos de la empresa memorizado por un ordenador, que es utilizado por numerosas personas y cuya organización está regida por un modelo de datos.
- Conjunto estructurado de datos registrados sobre soportes accesibles por el ordenador para satisfacer simultáneamente a varios usuarios de forma selectiva y en tiempo oportuno.
- Colección no redundante de datos compartibles entre diferentes sistemas de aplicación.
- Colección integrada y generalizada de datos, estructurada atendiendo a las relaciones naturales de modo que suministre todos los caminos de acceso necesarios a cada unidad de datos con objeto de poder atender todas la necesidades de los diferentes usuarios.
- Conjunto de ficheros maestros, organizados y administrados de una manera flexible de modo que los ficheros puedan ser fácilmente adaptados a nuevas tareas imprevisibles.
- Colección de datos interrelacionados.

3.3.2. Características

De acuerdo con Grassmann [Grassmann, 1997] un modelo de datos relacional incluye las siguientes partes:

- Una estructura de datos, por ejemplo, una relación o tabla en el modelo relacional.
- Un lenguaje para la definición de la estructura de datos.
- Un lenguaje para la manipulación de los datos, por ejemplo, álgebra relacional en el modelo relacional.
- Relaciones para la seguridad e integridad.

Cada una de estas partes tiene asociado unos términos especiales.

Una *relación* se corresponde informalmente con una tabla. La definición de una relación se especifica madiante un esquema, que es esencialmente una declaración de plantilla (o tipo) para una familia de posibles tablas concretas. Un *esquema* incluye lo siguiente:

- Nombres de las tablas.
- Conjunto de nombres de atributos.
- Dominio de cada atributo correspondiente.

Los dominios de los atributos no necesitan ser distintos, es decir, los atributos pueden compartir dominios. La mayoría de los sistemas de bases de datos no admiten *formalmente* la definición de dominios. Cada fila o registro del cuerpo de una tabla es una *tupla*. El número de columnas (atributos) de una tabla es el *grado* de una relación. Un *dominio* de un atributo es, esencialmente, un tipo de datos o conjunto de valores a partir del cual se seleccionan los valores del atributo. La *cardinalidad* de una relación es simplemente el número de tuplas que hay en la relación o el número de filas que hay en la tabla. La cardinalidad de una relación es un valor que varía con el tiempo debido a que las tuplas (o registros) se pueden añadir o borrar de la relación con el paso del tiempo.

La *clave primaria* es un atributo (o combinación de atributos) que identifica de manera única a un registro o tupla de una relación. Por consiguiente, una relación o tabla no puede contener registros o tuplas duplicadas. Aunque las entradas o registros de una tabla parecen enumerarse en forma ascendente, no se exige ese orden. Desde un punto de vista matemático, las tuplas de una relación forman un conjunto que no está ordenado.

Las tuplas de una tabla dada sólo se ordenan a efectos de su presentación. Sin embargo, ese orden no es un requisito, ya que las columnas de la tabla son rótulos. Como

consecuencia de este etiquetado, se pueden permutar fácilmente las columnas (es decir, componentes) de una tupla sin causar ninguna ambigüedad.

Una entidad (u objeto) tiene ciertas características o propiedades. Las propiedades seleccionadas para caracterizar una entidad, dentro del contexto de una aplicación particular, se denominan sus *atributos*. Los valores de los atributos se seleccionan dentro de un dominio. Un *dominio* D es un conjunto de valores del mismo tipo de datos. Todos los valores de un dominio son homogéneos. Un dominio puede ser atómico (simple) o compuesto. Un *dominio atómico* tiene valores que no pueden ser descompuestos sin pérdida de algún significado dentro de algún contexto de aplicación dado. Ejemplos de dominios atómicos son el conjunto de los enteros, reales y booleanos. Un *dominio compuesto* contiene valores no atómicos. Por ejemplo, el dominio del atributo *dirección* en la relación PERSONA es un ejemplo de dominio compuesto que especifica el nombre y número de la calle. Los atributos definidos para un mismo dominio se dice que son comparables porque los valores de estos atributos se seleccionan a partir de un mismo conjunto.

De acuerdo al modelo de bases de datos que se utiliza en el GAABAD, tanto el lenguaje de definición de datos como el de manipulación de los mismos se lleva a cabo por el OODM propuesto que lo mapea al RDBM; y la relaciones de seguridad son llevadas a cabo por el flujo del DFD propuesto en la relación terminador-proceso.

3.3.3. Modelado Formal

Una parte importante del modelo relacional que tiene que ver con la manipulación de las relaciones es el uso de las operaciones básicas [Grassmann, 1997]. Puesto que las relaciones son conjuntos, todos los operadores de conjuntos están disponibles para las relaciones. Para que una operación relacional sea significativa, los dos operandos (relaciones) deben tener el mismo tipo de tuplas; esto es, sus grados deben ser iguales. Además la parte de la cabecera de los dos esquemas relacionales para ambos operandos debe ser idéntica (o al menos compatible). Esto significa que ambas relaciones deben ser *compatibles por unión*. Esto implica lo siguiente:

1. El mismo conjunto de nombres de atributos.

2. Atributos con el mismo nombre definidos en el mismo dominio (o en uno compatible).

Unión (∪)

Dadas las relaciones compatibles por unión P y Q, su unión relacional, **P** ∪ **Q**, es una relación con la misma cabecera que P (o Q) y un cuerpo compuesto por las tuplas que pertecen a P, Q o a ambos. Las tuplas duplicadas se descartan.

Intersección (∩)

Dadas las relaciones compatibles por unión P y Q, su intersección relacional **P** ∩ **Q**, es una relación con la misma cabecera que P (o Q) y un cuerpo compuesto por las tuplas que pertenecen a P y a Q.

Diferencia (-)

Dadas las relaciones compatibles por unión P y Q, su diferecia relacional, **P - Q**, es una relación con la misma cabecera que P (o Q) y un cuerpo compuesto por las tuplas que pertencen a P y no a Q.

Producto (X)

El producto cartesiano de dos conjuntos **P** × **Q** es el conjunto de todos los pares ordenados tales que el primer elemento de cada par se selecciona en el primer conjunto y el segundo elemento se selecciona en el segundo conjunto. En el caso del producto de dos relaciones, sin embargo, el resultado deseado es un conjunto de tuplas etiquetadas, y no de tuplas ordenadas. Más formalmente, sean

$$P = \{(A1 : a1), (A2 : a2), ..., (An : an)\} \tag{3.5}$$

y

$$Q = \{(B1 : b1), (B2 : b2), ..., (Bm : bm)\} \tag{3.6}$$

tuplas no ordenadas etiquetadas en la relación P y Q, respectivamente. Entonces la tupla no ordenada etiquetada en P x Q es

$$P \times Q = \{(A1 : a1), (A2 : a2), ..., (An : an), (B1 : b1), (B2 : b2), ..., (Bm : bm)\} \tag{3.7}$$

El grado de esta tupla es m + n. Cuando las relaciones tiene un nombre de atributo común surge un problema en la formación del producto cartesiano de dos relaciones. En tal caso, algunos atributos deben cambiarse de nombre de manera que las cabeceras de las dos relaciones se conviertan en disjuntas o *compatibles por producto*.

Los operadores relacionales $\cup$, $\cap$ y x son todos asociativos y conmutativos. El operador diferencia, sin embargo, no es ni asociativo ni conmutativo.

Las operaciones relacionales básicas se complementan por cuatro operaciones adicionales: proyección, restricción, unión y división.

Proyección Π

La operación proyección selecciona atributos específicos de una relación indicada. Esos atributos corresponden a las columnas de una relación. La operación proyección también se puede ver como una operación de subdivisión vertical. Más formalmente, la proyección de una relación R sobre los atributos A1, A2,...Ai se escribe como:

$$\Pi_{A_1,A_2,\ldots,A_i}(R) \tag{3.8}$$

y denota una extracción de una relación de R compuesta de la cabecera de cada atributo espeficado y los valores de columnas para cada columna asociada en la relación. La cardinalidad de la relación resultante puede ser más pequeña que la de R ya que las tuplas duplicadas serán eliminadas.

Selección σ

Esta operación subdivide horizontalmente una relación especificada seleccionando un subconjunto de sus tuplas. Un formato de esta operación es:

$$\sigma_{A\,\theta\,OPER}(R) \tag{3.9}$$

A: nombre de atributo

θ: $=,\neq,\leq,\geq$

OPER: denota un nombre de atributo o constante.

Combinación (*Join*) $\bowtie$

La operación combinación es muy importante en las bases de datos porque permite la combinación de dos relaciones, las cuales comparten algunos atributos comunes, para producir una nueva relación única como resultado. La operación combinación tiene un número de variantes. Una variante llamada *combinación natural (natural join)*, es ampliamanente conocida como la operación más importante en bases de datos.

Formalizando la idea de unión natural:

Asumimos que,

$$\begin{aligned} P &= a_1, ..., a_n \\ Q &= b_1, ..., b_n \\ W &= d_1, ..., d_n \\ W &\notin P, Q \end{aligned} \tag{3.10}$$

Entonces la operación $P \bowtie Q$ se lleva a cabo de la siguiente manera:

Se renombran los atributos comunes en Q,

$$\begin{aligned} Q &= Q' \\ Q' &= \rho_{d1/b1}(...\rho_d m/bm(Q)...) \end{aligned} \tag{3.11}$$

Se toma el producto cartesiano y selecciona las tuplas las cuales se van a combinar,

$$R = \sigma_{b1=d1}(...\sigma_{bm=dm}(P \times Q)...) \tag{3.12}$$

Finalmente, se hace la proyección para liberar los atributos renombrados,

$$S = \Pi_{a1,...,an,b1,...,bm,c1,...,cm}(R) \tag{3.13}$$

División /

La *división* de P por Q, se denota como P / Q. Existe un dividendo y un divisor como en álgebra normal, los cuales están relacionados, y expresan débilmente que el dividendo puede contener varias veces al divisor, donde las veces significan el cociente.

Formalmente, asumimos que,

$$P = \{a_1, ..., a_n\}$$
$$Q = \{b_1, ..., b_n\} \tag{3.14}$$

Entonces la operación P/Q se lleva a cabo de la siguiente manera:

Se proyecta P a sus nombres de atributos únicos y se construyen todas las combinaciones con las tuplas en Q,

$$R = \Pi_{a_1,...,a_n}(P) \times Q \tag{3.15}$$

Se sustrae P de la relación,

$$S = R - P \tag{3.16}$$

En S se tienen todas las combinaciones que deberían haber estado en P pero que no estuvieron. Así, si hace la proyección de los atributos únicos de P para tener las restricciones de las tuplas en P para el cual no todas las combinaciones con las tuplas en Q estén presentes en P,

$$T = \Pi_{a_1,...,a_n}(S) \tag{3.17}$$

Se toma la proyección de P y sus atributos únicos y se sustraen los de T,

$$W = \Pi_{a_1,...,a_n}(P) - T \tag{3.18}$$

3.3.4. Síntesis

Se describieron las operaciones del modelo relacional, con el objeto de describir el mapeo del OODM al RDB.

En la siguiente sección se presenta el OODM propuesto, y posteriormente el mapeo del OODM al RDB.

3.4. Modelo de las Bases de Datos Orientadas a Objetos

Ya que se tiene el DFD propuesto y la relación que hay con el DRBM, a continuación se presenta el modelo del OODM propuesto, sus características y cómo se realiza el

mapeo al RDBM. El OODM propuesto se basó en la Tesis Doctoral "An Axiomatic Approach to Deductive Object-Oriented Databases" desarollada por Alvaro Adolfo Antunes Fernandes [Antunes, 1995].

3.4.1. Definición

Los OODBMS (de las siglas en inglés, Object Oriented Data Bases Management Systems, sistemas de manejo de bases de datos de orientadas a objetos) están diseñados para trabajar con lenguajes de programacion orientados a objetos, tales como C#, C++ y Java.

Cuando se integran las capacidades de bases de datos con capacidades de lenguajes de programación orientadas a objetos, el resultado es un OODMBS, que hace que los objetos de base de datos aparezcan como objetos de un lenguaje de programación en uno o mas lenguajes existentes. El OODBMS extiende el lenguaje con datos persistentes transparentemente, control de concurrencia, recuperación de datos, consultas asociativas y otras capacidades de las bases de datos.

3.4.2. Caraterísticas

A continuación se describen las generalidades del modelo orientado a objetos y posteriormente su formalización matemática. Los criterios que se toman para estos principios son los siguientes:

1. Las propiedades de las relaciones del modelo deben asegurar la construcción lógica de los OODM.

2. Las propiedades de las relaciones del modelo deben conformar lo más cerca posible el significado de los conceptos orientados a objetos.

El primer principio se aplica en los siguientes casos: en expresiones como *X es un conjunto de conjuntos de Y*, en su lugar es necesario nombrar todos los tipos intermedios, y separar las expresiones anidadas; por ejemplo, *X es un conjunto de Z* y *Z es un conjunto de Y*. Esto es necesario para asegurar que el lenguaje permanece libre de la función, dado que la estructura definida recursivamente se denota como un lenguaje de primer orden por un término definido recursivamente.

El segundo principio incluye las siguientes estipulaciones, entre otras. La herencia de las estructuras y su comportamiento está basado en una relación de inclusión reflexiva, transitiva y antisimétrica entre tipos de extensiones (típicamente leído como *es un* o *un tipo de*, por ejemplo un profesor es una persona). La relación de atribución entre tipos (típicamente leído como *tiene*), está entre otros afectado por la herencia (por ejemplo, si una persona tiene un nombre y un profesor es una persona, entonces un profesor tiene un nombre). La relación de composición entre tipos (típicamente leído como *hecho de* o *una parte de*) es asimétrica; por ejemplo, si un profesor está compuesto de materias entonces una materia no está formada de profesores.

Los conceptos se irán presentando a partir de un ejemplo. El ejemplo utilizado está basado en lo siguiente: suponga una universidad donde existen profesores y trabajadores; cada uno de ellos con características especiales y con característica comunes; es decir, parten de un mismo principio, pero se dividen en sus propias características.

Tipos básicos

Los tipos básicos utilizados para el diseño del modelo del GAABAD son las clases y los objetos.

La **clase** es el tipo más fundamental, porque el conocimiento de las entidades en dominios de aplicación, tanto de los aspectos estructural y de comportamiento, se define en las clases en lugar de en los objetos individuales que contienen. Ejemplos de clases son aquéllos denotados por nombre, apellido paterno, apellido materno, matrícula, sueldo, entero, caracter. Intuitivamente, un elemento de clases es una entidad en un dominio de aplicación.

Ejemplos de **objetos** son aquéllos denotados por 'Juan Carlos', 'Martínez', 'Pérez', 9100209F, 2300.00, 1, A. Intuitivamente, un elemento de objetos es una entidad.

Dimensiones de los tipos básicos

Las dimensiones dentro de las cuales estos tipos básicos se pueden organizar son:

1. En una dimensión **dependencia-independencia**, en donde cada uno de los tipos básicos se dividen en dos subgrupos, calificados en **específico** (en el sentido de *dependencia de la aplicación*) y **genérico** (en el sentido de *independencia de la aplicación*).

Por lo tanto, se definen los siguientes tipos: **clases específicas** y **clases genéricas**, **objetos específicos** y **objetos genéricos**.

Por ejemplo, nombre, apellido, matricula denotan elementos de clases específicas, mientras que entero denota un elemento de clases genéricas; 1, A denotan elementos de objetos específicos, mientras que 'Juan Carlos', 9100209F denotan elementos de objetos genéricos.

2. En una dimensión **esquema-instancia**, las clases son utilizados para modelarlos a nivel *esquema* (o *tipo* o *intensional*), mientras que los objetos métodos a nivel *instancia* (o *elemento* o *extensional*).

   Por ejemplo, nombre, apellido, matrícula, int denota el nivel esquema, mientras que 'Juan Carlos', 'Martínez', 9100209F, 1 denotan a nivel instancia.

Los objetos están organizados en clases, de los cuales entonces se dicen ser *instancias*. Por ejemplo, el objeto 9100209F es una instancia de la clase Matrícula.

Las clases están organizadas en jerarquía de herencia basado en una relación de inclusión sobre su *extensión*, por ejemplo, el conjunto de todas sus instancias. Por ejemplo, la clase profesor es una subclase de la clase persona, o, informalmente, un profesor es una persona. Esto implica que cada instancia de un profesor es también una instancia de una persona. Las jerarquías de herencia son algunas veces también referidas como una *jerarquía de inclusión*.

Las construcciones que modelan objetos estructuralmente son de dos tipos. Primero, un objeto es estructuralmente modelado por *propiedades* que son *atribuidos* a las clases de las cuales el objeto es una instancia. Por ejemplo, el nombre de una persona nombra (la extensión de) una propiedad que es un atributo de instancias de la clase persona. Segundo, un objeto complejo (i.e. una instancia de la clase compuesta) está modelado por los *componentes* de su *construcción*. Por ejemplo, una relación como impartido-por, entre un profesor y una materia, puede ser representado como una clase compuesta construida por agregación de las clases relacionadas. Por lo tanto, cada instancia de impartido-por tendrá en su construcción un par ordenado entre un autor y una materia. Por ejemplo, la instancia !101 de impartido-por puede ser el que haga la instancia de profesor cuyo nombre es 'Juan Carlos' es para la instancia de materia cuyo nombre es 'Matemáticas'. Además de las agregaciones, asociaciones y ordenamiento son también abastecidos.

Relaciones básicas

Las relaciones básicas seleccionadas para modelar los conceptos orientados a objetos son las siguientes:

1. **Inclusión**, que denota una relación binaria en **clases**, da lugar a las nociones de *superclase* y *subclase* de una clase dada. Como ejemplos, considere: un profesor es un tipo de persona (por ejemplo, la clase profesor es una subclase de la clase persona o, de forma equivalente, un persona es una superclase de la clase profesor), un trabajador es un tipo de persona, un alumno es un tipo de persona.

2. **Atribución**, que denota una relación binaria en **clases específicas** que da lugar a la noción de propiedades descriptivas, o *atributos* poseídos por todos los objetos de la clase. Por ejemplo, una personar tiene un nombre, apellido paterno y materno.

3. **Composición**, que denota una relación binaria en **clases específicas** que da lugar a la noción de *clases bulto* (o *agregar*, o *componer*), por ejemplo, clases cuyas instancias han construido objetos (o agregar, componer, complejar). Por ejemplo, una instancia de materias impartidas está construido como una colección sin orden de títulos de materias, una instancia de materias del tronco común está construida como una colección ordenada de materias.

Relaciones subsidiarias

Las relaciones básicas **inclusión**, **atribución**, **composición** abarcan todos los niveles esquemáticos de información que es específico al dominio que es modelado. En particular, ellos saltan la distinción entre información que está explícitamente declarado e información implícita que se obtiene por inferencia.

Sin embargo, esta distinción es esencial en el orden de hacer cumplir restricciones en herencia estructural y del comportamiento. Por ejemplo, para garantizar que siempre haya una mayor implementación específica de un método, el cual, por omisión, reemplaza a todos los otros, debe ser posible para diferenciar entre las clases distintas donde cada implementación candidata ha sido fijado por declaración implícita. En otras palabras, cuando una invocación a un método es ambiguo y el caso por omisión de último atascamiento se supone va a tomar lugar, el proceso de desambigüedad confía en una distinción entre operaciones declaradas y heredadas.

Para diferenciar entre información declarada e inferida, las siguientes relaciones subsidiarias son introducidas, una para cada una de las relaciones básicas a nivel esquema. Así, **inclusión***, **atribución***, **composición*** son relaciones subsidiarias correspondientes, respectivamente de las relaciones básicas **inclusión**, **atribución**, **composición**.

Además de las arriba mencionadas se necesita distinguir entre información declarada e inferida, también es el caso de la **inclusión*** que es una especificación más comprensiva que **inclusión**, en la medida en que como **inclusión*** especifica el modo de inclusión particular y la jerarquía de inclusión en la cual las dos clases están relacionadas, mientras que **inclusión** no especifica ninguna. Similarmente, **composición*** es una especificación más comprensiva que **composición**, en la medida que **composición*** especifica el modo de composición particular en el cual las dos clases están relacionadas, mientras que **composición** no.

Estas relaciones subsidiarias se dicen ser inducidas por las relaciones básicas.

- **Inclusión***, difiere de la **inclusión** llana en que tiene argumentos adicionales para determinar, respectivamente, el modo de inclusión (perteneciente a las tres disponibles, especialización, generalización y partición) que toman entre dos clases, y la jerarquía de inclusión en cuestión. Además, la **inclusión** está en el rango de información inferida y declarada, mientras que **inclusión*** solo sobre la declarada.

  Para ilustrar las diferencias entre los modos de inclusión posibles, considere lo siguiente. Si profesor, alumno y trabajador especializan la clase persona en su jerarquía categoría, éste se toma que significa que puede haber una instancia de la clase persona que no es una instancia de cualquiera de las subclases de la categoría de las anteriores. Si persona se generaliza en su jerarquía de formato las clases profesor y trabajador éste toma el significado que cada instancia de la clase persona es una instancia de al menos una de las subclases de formato anteriores. Finalmente, si se particiona la clase persona en su jerarquía origen, éste se toma que significa que cada instancia de la clase persona es una instancia de solamente una de las subclases anteriores.

- **Atribución***, difiere de la **atribución** llana en que el anterior solo relaciona una clase con sus propiedades declaradas, mientras que ésta adicionalmente relaciona una clase con sus propiedades heredadas.

  Por ejemplo, suponga que un profesor está declarado a ser una persona en algún

modo de inclusión y alguna jerarquía de inclusión. Suponga también que una persona es declarada para tener, entre otros, una propiedad nombre y que un profesor está declarado para tener, además, una propiedad especialización (ej, medicina, matemáticas, etc). Entonces, este par (profesor, nombre) está en la extensión de **inclusión** pero no en la de **inclusión***. Lo que esto quiere decir, nombre es una propiedad heredada, lo opuesto a asignada a profesor. En constraste, el par (especialización, profesor) está en la extensión de **inclusión*** (porque especialización fue asignada a profesor) y también en la extensión de **inclusión** (porque cada propiedad asignada a una clase es, trivialmente, heredada por esa clase).

- **Composición***, difiere de la **composición** llana en tener un tercer argumento que determina el modo de composición (entre las tres disponibles, asociación, ordenamiento y agregación) que toma entre dos clases. Además, **composición** está en el rango de información declarada e inferida, mientras que **composición*** sólo está en el rango de la última.

  Para ilustrar las diferencias entre los posibles modos de composición, considere lo siguiente. La **asociación** modela colecciones desordenadas de objetos de tipo homogéneo (ej, curricula es una asociación de materias); el **ordenamiento** modela colecciones ordenadas de objetos de tipo homogéneo (ej, la materia_tronco_comun de materias es un ordenamiento de materias impartidas); y la **agregación** modela colecciones de tuplas (ej, una materia impartida es una agregación de fecha_inicio, periodo, fecha_terminación).

3.4.3. Modelado Formal

Los axiomas utilizados del modelo orientado objetos son los siguientes:

- **inclusión** está inducido por **inclusión***, por desatención al modo de **inclusión** y jerarquía de herencia. De forma mas detallada, la **inclusión** es una restricción de **inclusión*** al primer y tercer argumento de éste, el cual en el caso del modo de inclusión es **generalización**, intercambia posiciones en la **inclusión**

$$(is\_a*(x,y,u,v) \wedge (y = especializacion \vee y = particion)) \Rightarrow is\_a(x,u) \qquad (3.19)$$

- **inclusión** es reflexiva sobre **clases** y transitiva

$$class(x) \Rightarrow is\_a * (x, x)$$

$$(is\_a(x, y) \wedge is\_a(y, z)) \Rightarrow is\_a * (x, z) \tag{3.20}$$

- **atribución*** está incluida en **atribución**. De forma mas detallada, la **atribución** contiene **atribución***, pero no necesariamente viceversa.

$$has * (x, y) \Rightarrow has(x, y) \tag{3.21}$$

- **atribución** se propaga hacia la **inclusión**. De forma mas detallada, si x *tiene una* y y z *es_una* x, entonces z *está formada_de* y

$$(has(x, y) \wedge is\_a(z, x)) \Rightarrow has(z, y) \tag{3.22}$$

- **composición** se induce de **composición*** por desatención al modo de composición. De forma mas detallada, la **composición** es una restricción de **composición*** por el primer y tercer argumento de éste

$$made\_of * (x, y, z) \Rightarrow made\_of(x, z) \tag{3.23}$$

- **composición** se propaga hacia **inclusión**. De forma mas detallada, si x *está formada_de* y y z *es_una* x, entonces z *está formada_de* y

$$(made\_of(x, y) \wedge is\_a(z, x)) \Rightarrow made\_of(z, y) \tag{3.24}$$

Las relaciones básicas y subsidiarias del modelo orientado a objetos se resumen en las siguientes relaciones:

es un	inclusión	Especialización
es un	inclusión	Partición
tiene	atribución	
se compone de	composición	Asociación
se compone de	composición	Ordenamiento
se compone de	composición	Agregación

Que formalmente es,

$$Relaciones = \{R_i, R_a, R_c\} \tag{3.25}$$

donde,

R_i = Relación de inclusión.
R_a = Relación de atribución.
R_c = Relación de composición.

Cabe mencionar que cuando se habla de Clases genéricas nos referimos a las clases que representan la atribución, inclusión por especialización y partición y la composición por asociación, ordenamiento y agregación. Cuando se habla de Clases específicas se refiere a las generadas por el usuario y que cuentan con características específicas.

Por lo tanto, a la unión de las Clases Genéricas con Clases Específicas es el conjunto de las Clases, que se denota formalmente como,

$$C = Cg \bigcup Ce \tag{3.26}$$

La relación directa con el DFD propuesto se obtiene a partir de las clases específicas que, como se mencionó en el apartado 3.2.3, se refiere al almacenamiento (S), Así, tenemos que ,

$$S = Ce \tag{3.27}$$

La relación de inclusión pertenece el conjunto potencia las Clases con el Modo de Inclusión y las Clases. Formalmente,

$$R_i \in P(C \times IM \times C)$$
$$IM \in \{'especializacion','particion'\} \tag{3.28}$$

La relación de atribución pertence al conjunto potencia de clases genéricas con clases específicas, siendo estos previamente definidos con su propio nombre y un tipo de dato genérico. Formalmente,

$$R_a \in P(C \times C) \tag{3.29}$$

donde,

$$A = (A_1, A_2, ..., A_n)$$
$$A_i \in C$$
$$Cg = \{(tipo, long)\}$$
$$tipo = \{'char', 'int', 'date', 'float'\}$$
$$long \in \mathcal{N}$$

Las relaciones de compsición pertenecen al conjunto potencia de las clases con un modo de composición y las clases, Formalmente,

$$R_c \in P(C \times CM \times C)$$
$$CM \in \{Asociacion, Ordenamiento, Agregacion\} \tag{3.30}$$

3.4.4. Síntesis

Ya se tienen los modelos que se van a utilizar y la formalización de cada uno de ellos de manera independiente. Es necesario presentar la interrelación entre cada uno de ellos, el cual se verá en el siguiente apartado.

3.5. Sistema

A continuación se presenta el modelo propuesto en GAABAD para la solución al diseño de sistemas de información, el modelo orientado a objetos propuesto y su correspondencia en el modelo relacional.

La atribución en el modelo orientado a objetos se refiere a los atributos que tiene la clase, mapeado al modelo relacional se refiere a los campos que tiene la relación, incluyendo su llave; la Tabla 3.1 muestra el mapeo correspondiente.

La inclusión por especialización muestra que la clase B pertence a la clase A y que puede pertenece a alguna otra clase, donde, de acuerdo al modelo relacional se establece la proyección de los campos que los contienen seleccionándolos a partir de la unión de las relaciones A y B a partir de la llave que establece la unión entre ellos, como lo representa la Tabla 3.2.

Modelo OO		**Modelo Relacional**	**Verificación semántica**
Atribución		**Atributos**	
$A = (A_1, A_2, ..., A_n)$ has* (A, A_i)	$\longrightarrow$	M(A) = (k_A, A_1, A_2, .. , A_n) M(has *)(A, A_i) = $\Pi_{M(A_1),M(A_2),...,M(A_n)}$	

Tabla 3.1: Atribución

Modelo OO		**Modelo Relacional**	**Verificación semántica**
Inclusión Especialización			
A = (A_1, A_2, .. , A_n) is_a * (B,A)	$\longrightarrow$	M(A) = (k_A, A_1, A_2, .. , A_n) M(B) = (k_B, B_1, B_2, .. , B_n,k_A) is_a * (B,A especializacion)=$\Pi_{A_1,A_2,..,A_n,B_1,B_2,..,B_n}$ ($\sigma_{B'.k_A=A'.k_A}(B'XA')$)	$A = \bigcup B_i$ $B_j, B_k \in \{B_i\}$

Tabla 3.2: Inclusión por especialización

La inclusión por partición indica que la clase B pertenece a la clase A, pero no puede pertenecer a otra clase; el mapeo hacia el modelo relacional es similar al de la especialización a diferencia que en este caso es necesario checar que no exista una llave tal que relacione a la clase A con otra clase diferente a la clase B; la Tabla 3.3 muestra el mapeo correspondiente.

La composición por asociación indica que la clase A, además de sus atributos, cuenta con los atributos de la clase B, para establecer el mapeo hacia el modelo relacional

Modelo OO		**Modelo Relacional**	**Verificación semántica**
Inclusión Partición			
A = (A_1, A_2, .. , A_n) is _a * (B,A)	$\longrightarrow$	M(A) = (k_A, A_1, A_2, .. , A_n) M(B) = (k_B, B_1, B_2, .. , B_n,k_A) is_a*(B,A, particion)=$\Pi_{A_1,A_2,..,A_n,B_1,B_2,..,B_n}$ ($\sigma_{B'.k_A=A'.k_A}(B'XA')$)	$A = \bigcup B_i$ $B_j, B_k \in \{B_i\}$ $j \neq k \rightarrow B_j \bigcap B_k = \phi$

Tabla 3.3: Inclusión por partición

Modelo OO		**Modelo Relacional**	**Verificación semántica**
Composición Asociación			
A = (A_1, A_2, .. , A_n) made_of (A,B)	$\longrightarrow$	M(A) = (k_A, A_1, A_2, .. , A_n,k_C) M(B) = (k_B, B_1, B_2, .. , B_n) C' = (k_C, k_B) made_of(A,B)=$\Pi_{A_1,A_2,..,A_n,B_1,B_2,..,B_n}$ ($\sigma_{A'.k_C=C'.k_C}(A'XB')XC'$) $\sigma_{C'.k_B=B'.k_B}$	

Tabla 3.4: Composición por Asociación (Conjunto)

Modelo OO		**Modelo Relacional**	**Verificación semántica**
Composición Ordenamiento			
A = (A_1, A_2, .. , A_n) made_of (A,B)	$\longrightarrow$	M(A) = (k_A, A_1, A_2, .. , A_n,k_C) M(B) = (k_B, B_1, B_2, .. , B_n) C' = (k_C, k_B) made_of (A,B)=$\Pi_{A_1,A_2,..,A_n,B_1,B_2,..,B_n}$ ($\sigma_{A'.k_C=C'.k_C}(A'XB')XC'$) $\sigma_{C'.k_B=B'.k_B}$	M(B) ordenado por *op* por O $op \in \{\leq \geq\}$ $O \subseteq \{B_1, B_2, .., B_N\}$, $O \neq \phi$

Tabla 3.5: Composición por Ordenamiento (lista)

es necesario hacer la proyección de los campos de ambas relaciones y seleccionarlos a partir de las llaves de una tercera relación en las que se va almacenando la composición y así poder hacer la combinación entre ellas y obtener el resultado deseado; la Tabla 3.4 muestra el mapeo correspondiente.

La composición por ordenamiento es similar a la anterior, con la particularidad que se requiere hacer el ordenamiento del resultado deseado a partir de las operaciones de ordenamiento mayor que o menor que dependiendo de la información que se trate; la Tabla 3.5 lo representa.

La composición por agregación representa que la clase A se compone de la clase B, que a su vez está formada por un conjunto de clases; esto es, que al mapear al modelo relacional es necesario hacer la selección al conjunto de relaciones de las que se conforma y

Modelo OO		**Modelo Relacional**	**Verificación semántica**
Composición Agregación			
A = (A_1, A_2, .. , A_n) ↓ B = (B_{i1}, B_{i2}, .. , B_{im})	$\longrightarrow$	A' = (k_A, A_1, A_2, .. , A_n,k_B) B' = (k_B, B_1, B_2, .. , B_n) C' = (k_C, C_1, C_2, .. , C_n) M(B)=$\Pi_{A_1,A_2,..,A_n,B_1,B_2,..,B_n}$ ($\sigma_{B'.k_A=A'.k_A}(A'XB')$) M(C)=$\Pi_{A_1,A_2,..,A_n,C_1,C_2,..,C_n}$ ($\sigma_{C'.k_A=A'.k_A}(A'XC')$)	

Tabla 3.6: Composición por Agregación (tupla)

obtener por proyección todos los campos relacionados. Su representación se presenta en la Tabla 3.6.

3.6. Conclusiones

Ya se tienen los modelos formales utilizados, sus definiciones y características para el desarrollo del GAABAD y el mapeo del modelo relacional al modelo orientado a objetos propuesto. Se ha mostrado qué partes se han tomado de cada uno de los modelos y cómo se han adaptado al modelo propuesto, qué partes se tomaron y qué partes no, y cuáles ajustes se realizaron al modelo original. En el siguiente capítulo se verá como cada uno de estos elementos son utilizados en la interfaz visual.

Capítulo 4

Modelo de la Interfaz Visual

A continuación se presenta la interfaz que es utilizada para el diseño de las bases de datos. Se presentará la forma de utilizar la interfaz visual y la relación con los conceptos previamente vistos. Se explica el uso de cada uno de los elementos que lo componen y su representación de cada uno de los conceptos modelados. Se presentan los archivos donde se almacena la información y cómo es almacenada en relación con el modelo orientado a objetos.

4.1. Ejemplo

El ejemplo de la figura 4.1, para la representación gráfica de la interfaz visual sigue con el ejemplo del capítulo anterior de la universidad.

Se tiene el mundo de las persona, de las cuales podemos decir que tienen un nombre y un apellido paterno y materno. La atribución corresponde a estas características.

Las personas se dividen en profesores y trabajadores; esto es, existe una relación de inclusión por especialización. El profesor tiene una matrícula y el trabajador tiene un sueldo. Además, una profesor no puede ser trabajador y viceversa; esto es; existe una relación de inclusión por partición.

La composición por agregación indica que puede tener elementos que forman otros atributos; así, una materia tiene el par clave, nombre que forman parte de los atributos del profesor, existiendo así una relación de composición por agregación; esto es, el profesor tiene materias, relación de composición por asociación, y también tiene materias pero del tronco común; esto es, existe una relación de composición por ordenamiento.

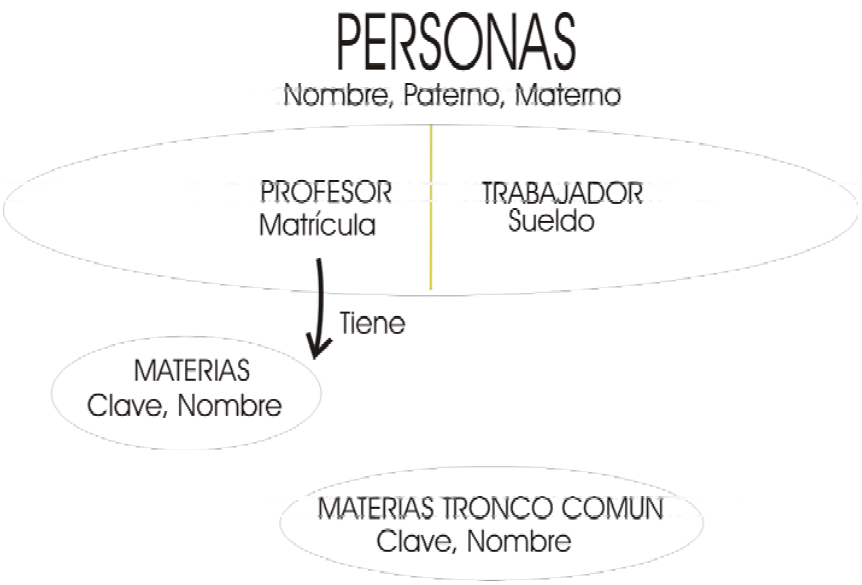

Figura 4.1: Personas en ambiente escolar: Profesores y Trabajadores

Se inicia modelando el OODM, para posteriomente modela el DFD; esto debido a que la función principal del DFD es la de realizar las acciones de las bases de datos; por lo tanto, si ésta no está definida, no se podrá realizar.

4.2. El sistema

Se ha desarrollado una interfaz visual para el diseño de sistemas de información utilizando un modelo orientado a objetos. Éste se almacena en archivos de texto. El archivo de texto se encuentra sincronizado con la interfaz visual; esto es, cualquier cambio en la interfaz visual se ve afectado en el archivo de texto, y viceversa. A continuación se presentan cada uno de ellos.

4.3. La Interfaz Visual

La interfaz visual está compuesta por una pantalla principal a partir de la cual se modelan conceptualmente las aplicaciones de bases de datos; ésta está dividida en dos áreas de trabajo; una para el OODM propuesto y otra para el DFD propuesto. Estos elementos

se detallan a continuación.

4.3.1. La pantalla principal

Al entrar al sistema, lo primero que encontramos es una pantalla dividida en dos partes (ver Figura 4.2):

- La parte izquierda es el área de trabajo.
- La parte derecha es el área de información.

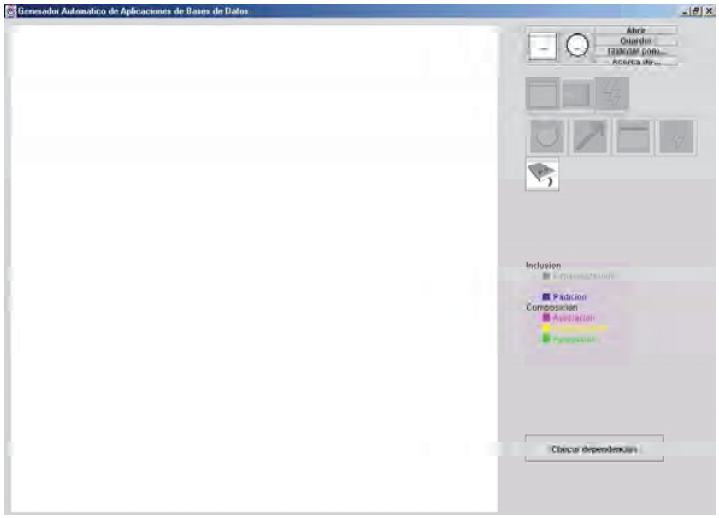

Figura 4.2: La pantalla principal

En la parte superior del área de información se encuentran dos botones que activan las funciones tanto del OODM como del DFD (Figura 4.3). El botón de la izquierda activa las operaciones del OODM y el de la derecha activa las operación del DFD.

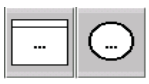

Figura 4.3: Selección entre OODM (izq) y DFD (der)

Para las bases de datos se cuentan con los botones para la creación de las clases y las tuplas y un botón para la generación de los scripts correspondientes. La Figura 4.4 muestra los botones asociados con el manejo de las bases de datos. El primer botón se encarga de generar las bases de datos, con él se establecen los conceptos de atribución, inclusión y composición. Para la composición por agregación en particular se utiliza el segundo botón; y para la generación de los scripts Sql se utiliza el tercer botón.

Figura 4.4: Creación de las bases de datos

La Figura 4.5 muestra los botones asociados con el manejo de los flujos de información. El primer botón crea los procesos con sus respectivas acciones, también en él se selecciona la base de datos a partir de la cual se van a crear las consultas. El siguiente botón crea los flujos necesarios de los terminadores a los procesos. El tercer botón crea los terminadores y el último botón genera los scripts necesarios para su manipulación.

Figura 4.5: Creación de los flujos de información

4.3.2. Bases de datos

Los elementos que se muestran a continuación, pertenecen a la interfaz visual para el ingreso del diseño de la base de datos.

La Figura 4.6 muestra una pantalla típica del sistema relacionado con las bases de datos. Las opciones de las bases de datos se encuentran separadas por pestañas, cada una con una tarea específica, y de acuerdo al modelo propuesto. La **clase**, es donde se ingresa el nombre de ésta; con las restricciones correspondientes en el modo de empleo. La **atribucion**, es donde se introducen los atributos de las bases de datos que, de acuerdo al modelo, pueden ser por atribución o por composición; en el caso de la composición es necesario seleccionar la base de datos a partir de la cual se va a incluir. Las siguientes dos

pestañas son relativas a la inclusión, la cual puede ser por **especialización** o **partición,** donde se selecciona(n) la(s) base(s) de datos de las cuales se incluyen. Para la composición por agregación, que utiliza tuplas se agregan las clases como lo muestra la Figura 4.7 donde se selecciona el nombre de la tupla que será utilizada, seguida de la(s) clase(s) de las cuales se va a conformar, que se seleccionan de la lista.

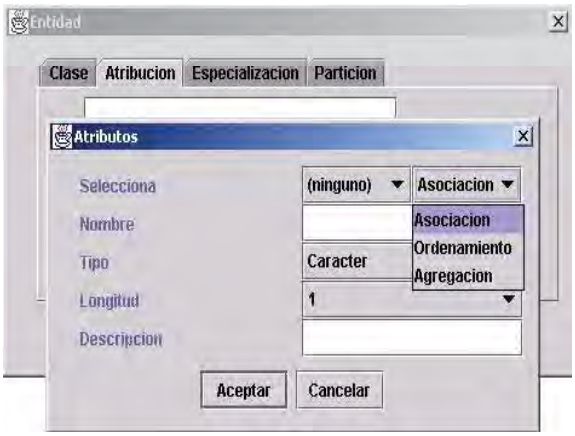

Figura 4.6: Atribución de las bases de datos

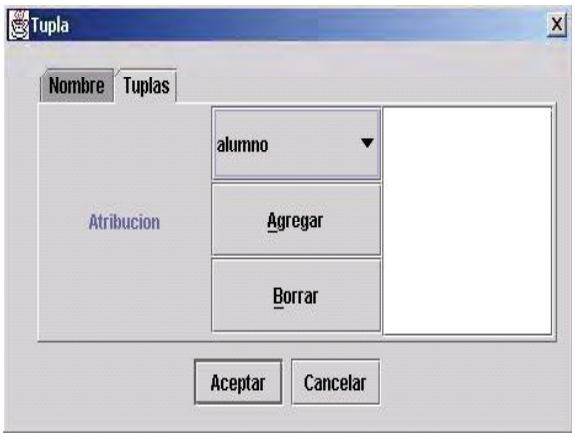

Figura 4.7: Composición por agregación

4.3.3. Flujo de la información

Al igual que para las bases de datos, se muestra la pantalla para el ingreso de la información del proceso (Figura 4.8) que contiene las pestañas. **Proceso** donde se ingresa el nombre de la clase. **Elemento** muestra las acciones que se definieron: Consultar, Insertar, Borrar o Modificar. **Bdd** selecciona la base de datos que se va a utilizar, siendo ésta la principal y, si ésta contiene atributos por inclusión o composición, en cualquiera de sus formas, todas ellas están disponibles para ser seleccionadas como atributos. **Campo**, selecciona los campos que se desean mostrar, estos pueden haber sido atribuidos, incluidos o compuestos. **Donde** sirve para seleccionar los campos a partir de los cuales se va a hacer la consulta o edición. **Ordenado por** permite seleccionar los campos a partir de los cuales se va a realizar el ordenamiento.

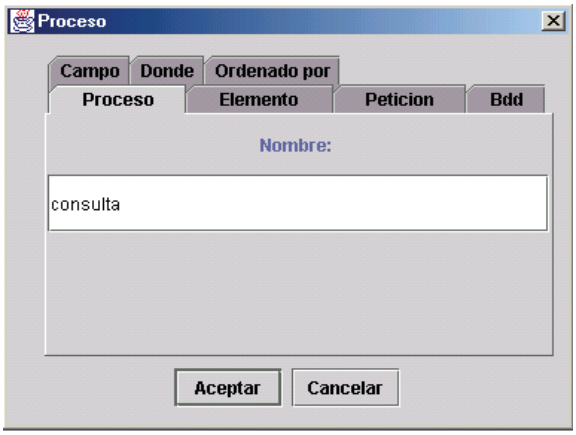

Figura 4.8: Diseño del nombre del proceso

4.4. El archivo de texto

La información se almacena en un archivo de texto el cual puede ser modificado en cualquier editor de texto. Existen archivos de texto tanto para las bases de datos como los flujos de información los cuales veremos a continuación.

4.4.1. Bases de datos

Consta de secciones para el ingreso de la información; en el orden en que aparecen se encuentra la clase, el número de atributos y los atributos y sus composiciones y, finalmente, sus inclusiones tanto por especialización, generalización y partición. Cada vez que se ingresa una clase perteneciente a la base de datos, en el archivo se almacena asignándole un número que corresponde a cada uno de estos y separados por un asterisco. Esta información es importante puesto que a partir de él, el sistema establece las coordenadas y las características de cada clase. Ejemplo de un archivo de texto es el siguiente:

```
Clase: persona
Atribucion: 3
nombre Caracter 20 nombres s
paterno Caracter 20 apellido paterno
materno Caracter 20 apellido materno
Especializacion: 0
Generalizacion: 0
Particion: 0
1
*
Clase: profesor
Atribucion: 2
matricula Entero 5 clave del profesor
Materia Composicion Asociacion
Materias Composicion Agregacion
Especializacion: 1
persona
Generalizacion: 0
Particion: 0
2
*
Clase: trabajador
Atribucion: 1
sueldo entero 5  Entero 5 Sueldo por trabajador
```

```
Especializacion: 1
persona
Generalizacion: 0
Particion: 1
profesor
3
*
Clase: Materia
Atribucion: 2
Clave Entero 5
Nombre Caracter 30 Nombre de la asignatura
Especializacion: 0
Generalizacion: 0
Particion: 0
4
*
Tupla: Curso
Atribucion: 1
Materia
*
```

Cabe destacar las palabras reservadas que se están utilizando. La palabra reservada **Clase** nos indica cómo se va a llamar la clase que se está diseñando. **Atribución** indica los atributos que contiene la clase. Los atributos que contiene pueden ser por **Composicion** por lo que el uso de esta palabra reservada le indica que tipo de composición se trata (Asociación, Composición, Agregación); el número de elementos, que son atributos, se indican a partir de un número que establece el número de atributos con los que cuenta; de la misma manera se encuentran las palabras reservadas **Especializacion**, **Generalizacion** y **Particion**. Para el caso de las tuplas se utiliza la palabra reservada **Tupla** y la palabra reservada **Atribucion** y el número de clases que conforman la tupla.

De igual manera, se cuenta con otro archivo donde se almacenan las coordenadas del sistema en diseño, muy importante debido a que a partir de él podemos hacer la representación gráfica del sistema.

```
Nombre: persona
coordenada: 193,126,193,126
1
*
Nombre: profesor
coordenada: 119,260,213,264
2
*
Nombre: trabajador
coordenada: 363,256,320,312
3
*
Nombre: Materia
coordenada: 225,484,146,504
4
*

Nombre: Curso
coordenada: 250,250,100,100
100
*
```

4.4.2. Flujo de la información

Para la creación de los diagramas de flujos de datos desde el modo texto, se requiere ingresar la siguiente información: el tipo de elemento que lo compone, el nombre del proceso, qué tipo de elemento se trata, a qué base de datos accede, los campos requeridos de la clase, a partir de qué campo van a ser las consultas, a partir de qué campo se va a ordenar, los datos y su tipo y el tipo de ordenamiento. Un ejemplo de esta construcción la podemos ver a continuación:

```
Tipo: Proceso
nombre: consAlumno
elemento: Consultar
```

```
bdd: Alumno
campo: 4
Alumno.Matricula
Persona.Nombre
Persona.Paterno
Persona.Materno
donde: 0
ordenado por: 0
NoDatos: 7
Alumno.a.Alumno
CostosCuotas.c.Alumno
Grupos.c.Alumno
Calificaciones.c.Alumno
Materias.c.Alumno
Maestro.i.Alumno
Persona.i.Alumno
orden: 0
1
*
```

Cabe describir las palabras reservadas que es están utilizando. **Tipo** nos indica si se trata de un proceso, flujo o terminador. **Nombre** es el nombre de la clase. **Elemento** nos indica si se trata de una consulta, inserción, actualización o borrado. **Bdd** indica la base de datos principal a partir de la cual se va a ejecutar la consulta. **Campo** indica los campos que se van a tomar en cuenta en la consulta; nótese que éstos pueden ser atributos directos, o los incluidos o compuestos, el uso de un número que le ayuda al sistema a saber cuántos son los campos a tomar en cuenta; donde indica los campos por los que se va a hacer la consulta, lo mismo con el número. **Ordenado por** indica los campos por los que se va a ordenar mas el número de campo por los que se va a ordenar. **NoDatos** indica los campos que se están utilizando y su descripción, esto si se accede a ellos como atributos propios de la base de datos, o bien por inclusión o composición. **Orden** que indica si el orden es ascendente (0) o descendente (1).

De igual manera, cuenta con un archivo de texto con las coordenadas, necesario para su representación visual. El archivo de coordenadas se presenta de la siguiente forma:

```
Nombre: Proceso
coordenada: 291,565,143,578
1
*
```

4.5. Modelado

El GAABAD se conforma de un modo visual y un archivo de texto, de los cuales podemos concluir que el archivo contiene la información tanto del diseño de las bases de datos como los datos de las coordenadas para establecer las posiciones del gráfico en la interfaz. Toda aplicación consta de un archivo de texto y un archivo que almacena las posiciones en el gráfico, para poder hacer una representación gráfica. La Tabla 4.1 la representa:

Aplicación = (texto, posiciones)
Aplicación $\Longrightarrow$ Visualización gráfica

Tabla 4.1: Representación de la aplicación

El texto corresponde a la representación de los diagramas de flujo de datos o de las bases de datos orientadas a objetos, como se puede ver en la Tabla 4.2.

texto = Dfd, Oodb

Tabla 4.2: Representación del texto

Las posiciones corresponden a las coordenadas iniciales y finales donde va a aparecer representado cada uno de los elementos que conforman el diseño. La Tabla 4.3 muestra la representación de las coordenadas.

posiciones = posiX, posiY, posfX, posfY
posiX = posición inicial en X
posiY = posición inicial en Y
posfX = posición final en X
posfY = posición final en Y

Tabla 4.3: Coordenadas

4.6. Conclusiones

Los modelos propuestos para el diseño de bases de datos orientados a objetos se han utilizado en la interfaz de usuario generada para el GAABAD y la introducción de datos para la obtención de los sistemas de información. Se presentaron las reglas que deben seguir para su correcto funcionamiento. En los siguientes capítulos se verá cómo se obtiene el código correspondiente a estos sistemas y las pruebas de su funcionamiento.

Capítulo 5

Generación automática de Código

En los capítulos anteriores se han presentado los modelos utilizados para el desarrollo y creación de bases de datos en GAABAD, también se ha presentado el ambiente de trabajo, así como los archivos a utilizar para su almacenamiento y el funcionamiento de ellos. En el presente capítulo se presentará la generación de código, tanto para las bases de datos como los scripts para la manipulación de la información.

Se presentarán los conceptos de atribución, inclusión y composición en cada una de sus formas a partir del ejemplo que se ha estado utilizando para su representación; para la generación del código de la creación de las bases de datos, del lenguaje de consulta (SQL) y los archivos de salida que serán utilizados por Internet, tanto en Php como en Asp.

5.1. Bases de Datos

La atribución del modelo orientado orientado a objetos que responde a "tiene", corresponde en el modelo relacional a cada uno de los atributos de la tabla; de acuerdo al ejemplo que se ha estado utilizando (3.4.2), para una persona que tiene nombre, apellidos (paterno y materno), mapeado al modelo relacional y de acuerdo a la Tabla 3.1, tenemos que:

```
CREATE TABLE Persona(
cvePersona int,
nombre varchar (30) not null default ' ' ,
Paterno varchar (20) not null default ' ' ,
Materno varchar (20) not null default ' ' );
```

Nota importante: Todas las relaciones tienen una llave - en este caso cvePersona - única para cada relación y que es con la que se hacen las uniones a otras tablas y así obtener el mapeo de acuerdo al modelo conceptual de la base de datos. También los valores generados por el script corresponden al modelo en la interfaz visual.

La inclusión del modelo orientado a objetos que responde a "es un" objeto de otra clase, corresponde a la proyección en el modelo relacional, como se explicó en las Tablas 3.2 y 3.3, dada por las claves en ambas relaciones; para poder realizar la proyección, como se especifica en el ejemplo utilizado (sección 4.1) un trabajador, que es una persona tenemos que:

```
CREATE TABLE Trabajador(
cveTrabajador int,
sueldo int not null default 0 ,
cvePersona int);
```

Es importante considerar que para la creación de las bases de datos se construye de forma similar para los dos modos de inclusión: especialización y partición (Tablas 3.2 y 3.3), la diferencia se establece al momento de diseñar el OODM.

En este caso, tiene su clave única que la representa - cveTrabajador - y la clave de la cual se hace la inclusión - cvePersona -, de esta manera se puede crear la relación de inclusión correspondiente.

La composición del modelo orientado a objetos que responde a "se compone de" otros objetos de otras clases, corresponde a la proyección de columnas y selección de filas en el modelo relacional, tal como se demostró en las Tablas 3.4, 3.5 y 3.6; de acuerdo a la especificación establecida en el ejemplo (sección 4.1), un profesor que es un trabajador "se compone de" materias, tenemos que:

```
CREATE TABLE Profesor(
cveProfesor int,
matricula int not null default 0 ,
cveTrabajador int,
cveMaterias int);
```

Caso similar a los modos de inclusión, los modos de composición: asociación, ordenamiento y agregación se establece la diferencia al diseñar el OODM.

Tiene su propia clave - cveProfesor - mas la de trabajador de la cual es miembro -cveTrabajador - y además las claves de las clases de las cuales se compone - en este caso, cveMaterias -.

Las materias, como no se incluyen, ni se compone de otro sólo cuenta con sus atributos.

```
CREATE TABLE Materias(
cveMaterias int,
Clave int not null default 0 ,
Nombre varchar (30) not null default ' ' );
```

5.2. Lenguaje de Consulta

El lenguaje de consulta es el código que se genera para la manipulación de la información correspondiente a las OODB previamente modeladas; este lenguaje de consulta es realizado por scripts en Asp y Php que contienen instrucciones SQL, que es el lenguaje utilizado en las RDB. Estos scripts corresponden a los conceptos introducidos en el modelo orientado a objetos, tales como atribución, inclusión y composición, como se demostraron anteriormente en las Tablas 3.2, 3.3, 3.1, 3.4, 3.5 y 3.6.

El lenguaje de consulta trabaja bajo las operaciones básicas consultar, insertar, borrar y modificar.

Para la acción **Consultar** se tiene que:

De acuerdo al ejemplo, una persona que "tiene" nombre y apellidos paterno y materno - atribución - se hace una proyección de los campos de la relación; esto es, traducido al lenguaje de consulta:

```
SELECT nombre,paterno,materno from Persona
```

Que corresponde a los atributos que contiene la clase.

Como es estableció en el ejemplo, Un trabajador que "es una" persona - inclusión - se hace la proyección de las columnas que se comparten entre relacciones de las cuales se hace una selección a partir de las claves que son iguales que, traducido al lenguaje de consulta queda de la siguiente manera:

```
SELECT profesor.matricula, persona.nombre, persona.paterno,
persona.materno from profesor left join persona on profesor.cvepersona
= persona.cvepersona
```

Para la composición el lenguaje de consulta se realiza de la proyección de columnas y selección de filas de las cuales se tiene la combinación; el ejemplo establece que un profesor "se compone de" materias, teniendo lo siguiente:

```
SELECT Materias.Clave, Materias.Nombre from profesor left join
Materia on profesor.cveMateria = Materia.cveMateria
```

Para la acción **insertar** en el caso de la atribución se hace una inserción de datos exclusivamente en la relación de la cual son atributos los campos; traducido al lenguaje de consulta:

```
insert into Persona(cvePersona,nombre,paterno,materno) values
(1,'nombre','paterno','materno')
```

Para la inclusión se inserta la información tanto a la relación padre como a la relación hija, estableciendo la clave para realizar las uniones correspondientes:

```
insert into profesor (cveProfesor, matricula) values (1,1)
insert into persona (cvePersona,cveProfesor,nombre,paterno,materno)
values (2,1,'nombre','paterno','materno')
```

Para la composición, el elemento compuesto ya existe; por lo tanto, solo se inserta a la relación que hace la consulta y su consiguiente clave para poder acceder a él cuando sea necesario.

```
insert into profesor(matricula, cveMateria, cveprofesor) values (1, 2, 2)
```

Para la acción **borrar** en el caso de la atribución sólo se borra la relación de la que son los campos; es decir, los atributos:

```
delete from Persona
```

En el caso de la inclusión primero se hace una selección de los campos comunes entre las relaciones implicadas para obtener las claves de los elementos que se van a borrar, para posteriormente pasar al borrado de los campos:

```
select persona.cvepersona from profesor left join persona on
profesor.cvepersona = persona.cvepersona
```

```
delete from persona where cvepersona = clave
```

En el caso de la composición se borran de la relación que llama a los registros que contienen la unión a esa relación de la que se compone, quedando únicamente la relación principal u otras relaciones de las que se componga:

```
delete from profesor where matricula = 1 and cveMateria = 2
```

Para la acción **modificar** en el caso de la atribución se modifican los campos de la relación exclusivamente:

```
update persona set nombre = 'otro nombre', paterno = 'otro paterno',
materno = 'otro materno'
```

En el caso de la inclusión primero se hace la selección de los campos comunes entre las relaciones implicadas para obtener las claves de los elementos al cual se le va a modificar información y a partir de éstos poder hacer las modificaciones a los campos requeridos.

```
select persona.cvepersona from profesor left join persona on
profesor.cvepersona = persona.cvepersona
```

```
update persona set nombe = 'nuevo nombre' where cvepersona = clave
```

En el caso de la composición no hay modificación, puesto que la modificación se hace a los campos relacionados de manera directa o por inclusión, debido a que estos son independientes del valor de la relación principal.

Cabe hacer mención que las claves para hacer las combinación cuando es inclusión se establecen en las relaciones que son las heredadas y cuando es composición se establecen de la que se inicia la composición.

5.3. Scripts

De acuerdo al modelo del DFD y del OODM propuestos y la generación de código, tanto de las bases de datos como del lenguaje de consulta, se generan los programas que se

utilizarán para la manejo de la información, con sus estructuras de manejo predefinidas y las conexiones a las RDB.

El ejemplo de creación de código que se presenta a continuación genera un archivo para consultar, de acuerdo al ejemplo que se ha venido utilizando, a un profesor y las materias que imparte; se demuestra el uso de los conceptos modelados; como un profesor es una persona, muestra los datos propios del profesor mas los de una persona del cual se incluye; además, el conjunto de materias de las que se compone.

En Php se genera el siguiente código:

```
<?php // Archivo: consulta1.php
 $bd = mysql_connect("localhost","root","");
 $exito = mysql_select_db("final",$bd);
if ($dondeprofesor_clave == "%=")
$dondeprofesor_clave = "like '$profesor_clave%'";
else if ($dondeprofesor_clave == "=%")
$dondeprofesor_clave = "like '%$profesor_clave'";
else if ($dondeprofesor_clave == "%=%")
$dondeprofesor_clave = "like '%$profesor_clave%'";
else
$dondeprofesor_clave = "$dondeprofesor_clave '$profesor_clave'";
$query = "SELECT profesor.clave, Materia.Clave, Materia.Nombre,
 persona.nombre, persona.paterno, persona.materno from
 profesor left join persona on
         profesor.cvepersona = persona.cvepersona
         left join Materia on profesor.cveMateria = Materia.cveMateria
         WHERE profesor.clave $dondeprofesor_clave  ";
muestra($query,$bd);
$query = "SELECT Materia.Clave, Materia.Nombre from profesor left join
         persona on profesor.cvepersona = persona.cvepersona  left
 join  Materia on profesor.cveMateria = Materia.cveMateria
         WHERE profesor.clave $dondeprofesor_clave  ";
muestra($query,$bd);
function muestra($query,$bd)
```

```
 {
$resultado = mysql_query($query, $bd);
$num_rows = mysql_num_rows($resultado);
if ($num_rows == 0)
    echo "<h3 align = center>No hay datos ...</h3>";
else
    {
     $num_fields = mysql_num_fields($resultado);
     echo "<table border = 1>";
     echo "<tr>";
     for ($i=0; $i<$num_fields; $i++)
         {
          $field = mysql_fetch_field($resultado);
          echo "<td><b>$field->name";
         }
     $resultado = mysql_query($query, $bd);
     while ($fila = mysql_fetch_row($resultado))
          {
      echo "<tr>";
           foreach($fila as $campo)
                { echo "<td>$campo"; }
          }
     echo "</table>";
    }
 }
mysql_close($bd);
?>
```

En Asp se genera el siguiente código:

```
<%
' Archivo: consulta.asp
 Dim rstRec,rstRec4Ins,rstRecIns
 Dim mConn
```

```
 Dim Cmd
 Set mConn = Server.CreateObject("ADODB.Connection")
 Set mCmd = Server.CreateObject("ADODB.Command")
 Set rstRec = Server.CreateObject("ADODB.Recordset")
 Set rstRec4Ins = Server.CreateObject("ADODB.Recordset")
 Set rstRecIns = Server.CreateObject("ADODB.Recordset")
 mConn.CommandTimeout = 1440
 mConn.CursorLocation = 1
 mConn.ConnectionString = "server=SERVIDOR;driver=SQL Server;db=final;uid=lcamacho;pwd=CHOlcam
 mConn.open
 mCmd.ActiveConnection = mConn
 mCmd.CommandType = 1
if (Request("dondeprofesor_clave") = "%=") then
    dondeprofesor_clave = "like '"&Request("profesor_clave")&"%'"
else
    if (Request("dondeprofesor_clave") = "=%") then
        dondeprofesor_clave = "like '%"&Request("profesor_clave")&"'"
    else
         if (Request("dondeprofesor_clave") = "%=%") then
             dondeprofesor_clave = "like '%"&Request("profesor_clave")&"%'"
         else
              dondeprofesor_clave = ""&Request("dondeprofesor_clave")&" '"&Request("profesor_c
   end if
    end if
end if
query = "SELECT profesor.clave, Materia.Clave, Materia.Nombre,persona.nombre,
persona.paterno, persona.materno from profesor left join persona on
profesor.cvepersona = persona.cvepersona  left join Materia on
profesor.cveMateria = Materia.cveMateria
WHERE profesor.clave "&dondeprofesor_clave&"  "
muestra(query)
query = "SELECT Materia.Clave, Materia.Nombre from profesor left join
persona on profesor.cvepersona = persona.cvepersona  left join Materia
```

```
on profesor.cveMateria = Materia.cveMateria
WHERE profesor.clave "&dondeprofesor_clave&"  "
muestra(query)
function muestra(query)
   mCmd.CommandText = query
   rstRec.Open mCmd,,3,3
   if rstRec.EOF then
      Response.Write("<h3 align = center>No hay datos ...</h3>")
   else
      Response.Write("<table border = 1>")
      Response.Write("<tr>")
      Dim campos()
      Dim dato()
      x = 0
      FOR EACH FIELD IN rstRec.FIELDS
          ReDim Preserve campos(x)
          campos(x) = FIELD.NAME
          x = x + 1
          Response.write("<td>"&FIELD.NAME)
      NEXT
      while not rstRec.EOF
         Response.Write("<tr>")
         for y = 0 to x - 1
             Response.write("<td>"&rstRec.Fields(y))
         next
         rstRec.moveNext
      wend
      Response.Write("</table>")
   end if
   rstRec.close
end function
 mConn.Close
%>
```

Así, en ambos casos los archivos de salida cuentan con los mismos principios, pero con las características propias para ser utilizados en diferentes sistemas operativos, la traducción del lenguaje es transparente para el usuario.

5.4. Conclusiones

En este capítulo se ha presentado las características del código generado por GAABAD, tanto para la creación de las bases de datos como de los scripts de consulta. Se ha presentado cómo se hace el mapeo del modelo orientado a objetos al modelo relacional, utilizando el lenguaje SQL. A continuación se presentará el sistema integrado.

Capítulo 6

Integración y pruebas del Sistema

Ya se han generado los códigos necesarios para el desarrollo de aplicaciones de bases de datos; en este capítulo se verá cómo estos archivos son utilizados para el manejo del sistema y su integración en una aplicación. En el capítulo anterior se mostró cómo se desarrollaba el sistema; ahora se verá cómo se interactúa con el sistema propiamente. Aquí se presenta la utilización de los scripts finales y en qué forma corresponden a cada uno de los conceptos del OODM.

6.1. Bases de Datos

En esta sección se introducirá en el manejo de la creación de las bases de datos utilizando el ejemplo presentado en la sección 4.1

De acuerdo con el ejemplo, que una persona "tiene" nombre y apellidos paterno y materno; la figura 6.1 presenta cómo se modela.

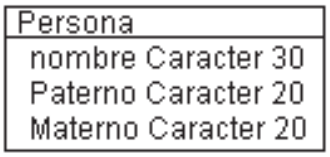

Figura 6.1: Atribución

De acuerdo al ejemplo, una trabajador "es una" persona y que además tiene sus propias características, por lo tanto, la inclusión por especialización la representa la figura

6.2, donde la línea representa que es una inclusión. La línea al principio tiene un punto que especifica la dirección de la relación, la inclusión por partición se especifica al ingresar a las propiedades de ella; pero gráficamente se representa de la misma forma.

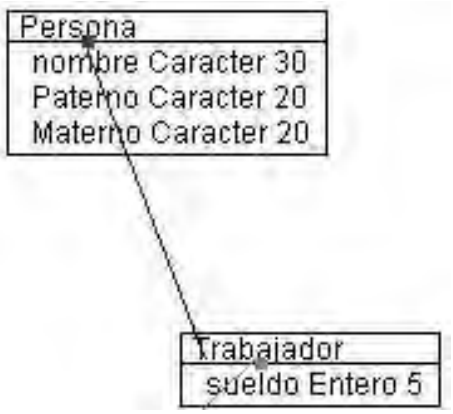

Figura 6.2: Inclusión por especialización

En el ejemplo, un profesor "es un" trabajador que tiene sus atributos y que además "se compone de" materias; gráficamente se representa por la figura 6.3.

De manera similar se presenta la composición que, en este caso es por asociación, las líneas también lo representan.

El uso de colores en el modelado permite representar cada uno de los elementos.

6.2. Consulta

Una vez que se tiene el OODM definido, se procede a la generación de los scripts, a partir de los cuales se va a utilizar en Internet, pudiendo acceder o agregar información a la base de datos crada a partir del OODM provisto.

La representación gráfica es similar en cada uno de los casos, cada uno de ellos selecciona la acción a realizar, las restricciones de acceso y la base de datos y sus campos, a partir de los cuales se va a llevar a cabo las acciones, como lo muestra la figura 6.4

Importante mencionar que, dado que el OODM propuesto propone conceptualmente acceder a los datos como un todo, independientemente que estos sean por atribución, inclusión o composición, el sistema, al ser modelado por el usuario bajo estos principios,

Figura 6.3: Composición por asociación

puede acceder a cualquier dato que en encuentre en la base de datos por cualquier modo de atribución, inclusión o composición; esto es, no es necesario que se lo especifique, el GAABAD se lo permite de manera transparente.

Finalmente, los scripts generan las pantallas para consultar, insertar, modificar o borrar información.

A continuación se presenta un ejemplo para hacer una consulta en Internet de acuerdo al script presentado en el sección 5.3, donde se utiliza la atribución, inclusión y composición como se especificó en el ejemplo de la sección 4.1.

Para consultar a un profesor con todas sus materias, se teclea la matrícula del profesor, para que haga la búsqueda y encuentre los datos, como lo muestra la Figura 6.5.

Éste, al ser una persona realiza la búsqueda como persona y también como profesor; además tiene materias, por lo tanto, también hace una búsqueda en materias. Esta búsqueda da como resultado, la pantalla que muestra la Figura 6.6.

Así, se presenta un caso típico de la generación de aplicaciones de bases de datos a partir de un OODM propuesto y un DFD propuesto.

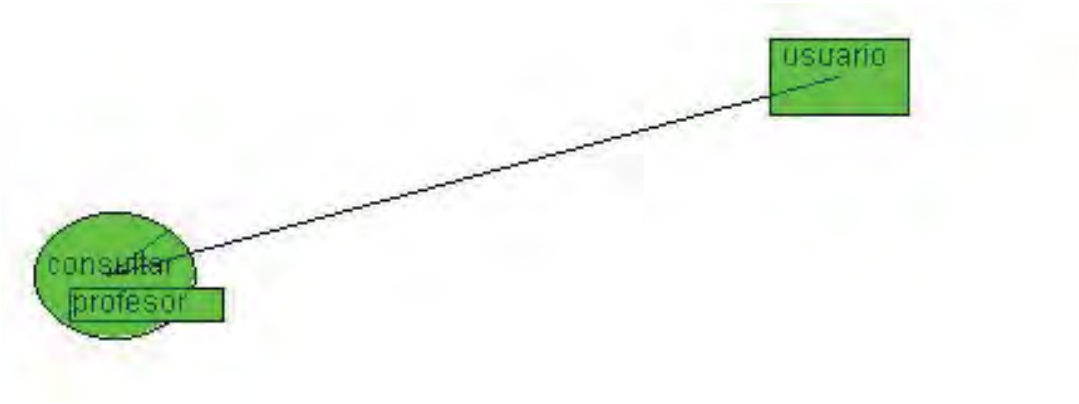

Figura 6.4: Generación de consultas en scripts

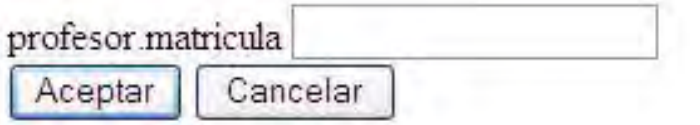

Figura 6.5: Petición de una consulta

matricula	nombre	paterno	materno
9100209F	Juan Carlos	Martinez	Perez

Clave	Nombre
123	Estadistica
321	Matematicas

Figura 6.6: Resultado de una consulta

Capítulo 7

Conclusiones

El sistema "GAABAD: Generador Automático de Aplicaciones de Bases de Datos mediante la descripción gráfica del diagrama de flujo de datos y un modelo orientado a objetos de los mismos" muestra un desarrollo de un modelo orientado a objetos propuesto y basado en la tesis doctoral "An axiomatic approach to Deductive Object-Oriented Databases" de Alvaro Adolfo Antunes Fernandes, donde presenta los conceptos utilizados en esta propuesta, tales como atribución, inclusión y composición y sus variantes; inclusión por especialización y partición y composición por asociación, ordenamiento y agregación.

Se usa para su implementación el modelo de bases de datos relaciones, donde es necesario mapear el modelo orientado a objetos a este modelo para su construcción, por lo que el sistema se limita a las capacidades del modelo relacional.

El resultado es un producto que satisface las necesidades básicas de un modelo orientado a objetos, teniendo como limitantes conceptos como la herencia múltiple y encapsulación que son parte del modelo, pero no son esenciales para la presentación de éste.

De esta manera, se deja para posteriores desarrollos la construcción de un sistema de bases de datos puramente orientado a objetos, donde se cubran en su totalidad todos los conceptos del modelo orientado a objetos.

Referencias

[Antunes, 1995] Heriot-Watt University
Department of Computing and Electrical Engineering Alvaro Adolfo Antunes Fernandes
An Axiomatic Approach to Deductive Object-Oriented Databases
1995

[Berrios,2002] Universidad Interamericana de Puerto Rico Profr. María de Jesús Berríos
Diagrama de Flujo de Datos
http://coqui.lce.org/mdejesus/CLAS2/tsld001.htm
2002

[Davidson,2005] Universidad de Pennsylvania
Grupo de Investigación en Bases de Datos
Susan Davison, Hartmut Liefke, Julie Herman, Xilin Li
Visual Query Language
http://www.seas.upenn.edu/g̃karvoun/dragon/research/vql.html
2005

[Datanamic, 1999] Datanamic
Dezign for Databases
http://www.datanamic.com/dezign/
1999

[Delaney, 2001] Kalen Delaney
A fondo Microsoft SQL Server 2000
McGraw-Hill, 2001

[DuBois, 2001] x Paul DuBois
MySQL Edición Especial
Prentice Hall, 2001

[Eisenback,2003] Imperial College
Faculty of Engineering
Department of Computing
Distribuited Software Engineering Group
Susan, Eisenback, Bashar Nuseibeh
Toolkit for Conceptual Modelling
http://wwwhome.cs.utwente.nl/~tcm
2003

[Grassmann, 1997] Winfried Karl GRASSMANN. Jean-Paul TREMBLAY
Matemática Discreta y Lógica. Una perspectiva desde la Ciencia de la Computación.
Prentice Hall, 1997

[INEI, 1997] Instituto Nacional de Estadística e Informática
Diagrama de Flujo de Datos
http://www.inei.gob.pe/web/metodologias/attach/lib604/cap3-2.htm
1997

[Jiménez, 2001] Profesora: Claudia Jiménez Quintana
Universidad de Concepción
Departamento de Ingeniería Informática y Ciencias de la Computación
Concepción, Chile
Bases de datos relacionales
http://www.inf.udec.cl/ basedato/apunte/capitulo1/capitulo1.html
2001

[Joyanes, 2001] Luis Joyanes Aguilar. Matilde Fernández Azuela
Java 2. Manual de Programacón.
McGraw-Hill, 2001

[Maslakowski, 2000] Mark Maslakowski

Aprendiendo MySQL en 21 días
Prentice Hall, 2000

[Piattini, 1993] Adoración de Miguel/Mario Piattini
Concepción y Diseño de Bases de Datos. Del Modelo E/R al Modelo Relacional
Addison-Wesley Iberoamericana, 1993

[Pressman,2002] Roger S. Pressman
Ingeniería del Software. Un enfoque práctico
Quinta Edición. McGraw-Hill, 2002

[Sanchez, 2001] Jesús Sánchez Allende. Gabriel Huecas Fernández-Toribio. Baltasar Fernández Manjón. Pilar Moreno Díaz
Java 2. Iniciación y Referencia
McGraw-Hill, 2001

[Tioga, 1999] University of California en Berkeley
El proyecto Tioga
http://tioga.cs.berkeley.edu/
1999

[Univ. Buenos Aires,1997] Facultad de Ciencias Económicas
Universidad de Buenos Aires
http://www.econ.uba.ar/www/departamentos/sistemas/plan97/
tecn_informac/cansler/cansler/DFD %20- %20teor %EDa.htm
1997

[Univ. Valladolid, 2000] Universidad de Valladolid
Informática de Gestión y de Sistemas
Diagrama de Flujo de Datos
http://usuarios.lycos.es/delegación
2000

[UML,2002] Modelado de Sistemas con UML
Popkin Software and Systems
http://es.tldp.org/Tutoriales/doc-modelado-sistemas-UML/multiple-html/
2002

Glosario

CASE Herramientas de Ingeniería de Software Asistido por Computadora; de las siglas en inglés Computer Aided Software Engineering.

CPL Collection Programming Language.

CRC Chequeo de Redundancia Cíclica; de las siglas en inglés Cyclic Redundancy Check; es una técnica poderosa pero fácilmente implementada para la obtención de datos confiables. Esta técnica es usada para proteger bloques de datos.

DDL Lenguaje de Definición de Datos; de las siglas en inglés Data Definition Language son aquéllas utilizadas para la creación de una base de datos y todos sus componentes: tablas; índices; relaciones; etc.

DFD Diagrama de Flujo de Datos; de las siglas en inglés Data Flow Diagram.

ERD Diagramas Entidad Relación para ilustrar la estructura de bases de datos; de las siglas en inglés Entity Relationship Diagrams.

GUI Interfaz Gráfica de Usuario; de las siglas en inglés Graphical User Interface; permite a los usuarios navegar e interactuar con las informaciones en la pantalla utilizando el mouse para señalar pulsar y desplazar iconos y otros datos en lugar de escribir palabras y frases.

HTML Lenguaje de Marcación de Hipertexto; de las siglas en inglés Hyper Text Markup Language; es un lenguaje de marcas diseñado para estructurar textos y presentarlos en forma de hipertexto que es el formato estándar de las páginas web.

Java Lenguaje de Programación orientado a objetos portable en distintas plataformas; tales como Windows y Linux.

OODB Base de Datos Orientada a Objetos; de las siglas en inglés Object Oriented Data Bases.

OODBMS Sistemas Manejadores de Bases de Datos Orientadas a Objetos; de las siglas en inglés Object Oriented Data Bases Management Systems.

OODM Modelo de Datos Orientado a Objetos; de las siglas en inglés Object Oriented Data Model.

OQL Lenguaje de Consulta de Objetos; de las siglas en inglés Object Query Language usado para transferir datos entre bases de datos.

RDB Bases de Datos Relacionales; de las siglas en inglés Relational Data Bases.

scripts Programas escritos mediante lenguajes interpretados tales como Asp y Php.

SQL Lenguaje de Consulta Estructurado; de las siglas en inglés Structured Query Language utilizado por los gestores de bases de datos como MySQL y SQL Server para crear; modificar; mantener y consultar una base de datos.

Web también llamado página de Internet que contiene información de un tema en particular que se encuentra almacenado en algún sistema de cómputo que se encuentre conectado en la red de Internet.