Universidad de las Ciencias Informáticas

Facultad 3

Título: Sistema Integrado de Gestión Estadística (SIGE).

Rol diseñador de la base de datos.

Trabajo para optar por el título de

Ingeniero en Ciencias Informáticas

Oficina Nacional de Estadísticas

Autor: Amado Alburquerque Arias.

Tutor: Lic. Ridosbey Milian Iglesias.

Junio 2007

Año 49 de la Revolución

Declaración de autoría

Declaro que soy el único autor de este trabajo y autorizo a la Facultad 3 de la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio.

Para que así conste firmo la presente a los _23_ días del mes de _junio_ del año _2007_.

_____	_____
Amado Alburquerque Arias	Ridosbey Milian Iglesias

Agradecimientos

Con este trabajo quiero agradecerle:

A mis padres por ayudarme en todo momento con sus consejos y apoyo a salir de los momentos difíciles.

Le agradezco a mi novia por estar junto a mi, comprenderme y amarme tanto.

Agradezco a mi tutor por guiarme en el desarrollo de este trabajo.

Le agradezco a mis compañeros y amigos que han compartido conmigo tantos momentos durante estos años y me han extendido su mano cuando me ha hecho falta.

Agradezco a los estudiantes del proyecto ONE por ayudarme en lo que necesitaba.

A todo el claustro de profesores que de una forma u otra haya influido en mi preparación como profesional y que me haya orientado durante toda la carrera.

A Fidel Castro y a la Universidad de las Ciencias Informáticas por permitir que yo me forme como ingeniero en esta carrera durante cinco años.

A todas aquellas personas que de alguna forma ha hecho posible la realización de este trabajo.

Dedicatoria

Dedico mi tesis, trabajo donde he puesto todo mi esfuerzo y dedicación, a:

Mis padres por haber depositado toda su confianza en mí y ser mi razón de ser.

A mi novia por ocupar un lugar tan importante en mi corazón y vida.

A mi familia.

Y a todas aquellas personas que se preocupan por mí.

Resumen

La gestión de información estadística es de gran importancia para cualquier país puesto que a través de ella permite tener una visión general y un control de muchos factores desde varios puntos de vista.

En nuestro país la Oficina Nacional de Estadísticas (ONE) constituye el órgano rector de las estadísticas. La ONE trabaja actualmente con el software Microset NT para gestionar la información estadística, pero la información la almacena en ficheros. Por esta razón se hace necesario el desarrollo de una base de datos para gestionar la información en un sistema estadístico.

Este trabajo se centra en el diseño y la implementación de una base de datos que contará de dos módulos: el módulo Gestión de Usuarios que permite gestionar la seguridad del sistema, y el módulo Sistema Estadístico que permite gestionar la información estadística. Además se realiza el diseño de la capa de acceso a datos para el módulo de Entrada de Datos del Sistema Integrado de Gestión Estadística.

Para hacer posible los objetivos de este trabajo, primeramente se realizó un estudio de los principales gestores de bases de datos que existen, las herramientas más utilizadas para diseñar bases de datos y las herramientas que se emplean para el diseño del acceso a los datos. Basado en el estudio anterior seleccionamos las herramientas que más se ajustaban a las características del trabajo que se iba a desarrollar. Luego de realizar la elección se describe la solución propuesta y finalmente se hace una valoración del trabajo realizado. Seguidamente se enuncian las conclusiones que se obtuvieron en este trabajo así como las recomendaciones del trabajo.

Índice de contenidos

Índice de figuras

Índice de tablas

Introducción

La evolución y uso de las Nuevas Tecnologías de la Informática y las Comunicaciones es uno de los principales aspectos que está marcando la diferencia entre las naciones del mundo, puesto que el desarrollo de los países depende en cierto punto del nivel de automatización de la industria y la sociedad.

Cuba no está ajena de tal situación y es por esto que se ha trazado como política, la informatización de la sociedad cubana, reto al que está dedicando gran cantidad de recursos y esfuerzos para preparar al capital humano que contribuirá al logro del objetivo antes mencionado. Un ejemplo fehaciente es la creación y desarrollo de la Universidad de las Ciencias Informáticas (UCI), universidad de nuevo tipo, donde sus estudiantes dedican tiempo a sus estudios y de igual manera trabajan en el desarrollo de proyectos productivos. La UCI constituye una ciudad experimental donde se aplican y desarrollan software para varios sectores de la sociedad, los cuales se prueban y de ser exitosos, se aplican en la sociedad cubana.

Uno de los software que se desarrollan en la UCI es el Sistema Integrado de Gestión Estadística (SIGE) para la Oficina Nacional de Estadísticas (ONE). La ONE constituye el órgano rector de las estadísticas, es la institución gubernamental que organiza, dirige, regula y controla la actividad estadística en Cuba. Esta información se analiza y el resultado de dicho estudio sirve de apoyo a la dirección del país para la toma de decisiones.

La información se recoge de forma escalonada, o por niveles. A la ONE se le subordinan y rinden parte un conjunto de Oficinas Territoriales de Estadísticas (OTE) que se encuentran situadas en cada provincia y rigen la actividad estadística territorial. A su vez las OTE tiene subordinadas y le rinden parte un conjunto de Oficinas Municipales de Estadística (OME) que realizan las labores estadísticas en los municipios, y recogen la información que les brindan directamente los Centros Informantes (CI), que no son más que todas las entidades del país.

La información se recoge a través de modelos estadísticos que contienen un conjunto de indicadores que expresan el comportamiento de la entidad a lo largo de un período de tiempo. Estos modelos son de tipo tabla y se utilizan para captar la información continuamente cada cierto período.

Para que las Oficinas de Estadística almacenen la información de forma segura y puedan realizar adecuadamente los análisis y estudios que se llevan a cabo con dicha información en el país, es necesario el desarrollo de una aplicación que cuente con una base de datos, debido a que el software MicroSet NT, que actualmente se utiliza para este proceso en el país, guarda la información en ficheros.

El trabajo con ficheros trae consigo varios problemas como son:

- Existe la posibilidad de que la información con que se trabaja no sea íntegra, es decir, que se trabaja con información no válida.
- Una misma información no pueden tenerla compartida todas las personas que la necesitan.
- Existe repetición en la información almacenada.
- No hay presencia de un orden y una organización en la información con la que se está trabajando.
- La información no se encuentra centralizada, es decir, la información no es considerada como un recurso corporativo que carece de dueños específicos.

Todos estos problemas traen como resultado que no estemos en presencia de un software con calidad que haga una eficiente gestión de la información y que los reportes que se dan puedan tener errores, lo cual no será confiable para una futura toma de decisiones en la dirección de nuestro país.

Problema Científico:

La no existencia de una estructura correcta y eficiente para el almacenamiento de la información estadística, compromete el correcto desempeño del Sistema Integrado de Gestión Estadística y afecta el proceso de desarrollo dentro del proyecto de la Oficina Nacional de Estadística.

Objeto de Estudio:

Diseño de bases de datos.

Campo de acción:

Diseño de bases de datos para un sistema de gestión de información estadística.

Objetivo General:

Desarrollar la base de datos para el Sistema Integrado de Gestión Estadística y el acceso a los datos para uno de sus módulos.

Objetivos Específicos:

- Diseñar la base de datos para los módulos Gestión de Usuarios y Sistema Estadístico del Sistema Integrado de Gestión Estadística.
- Implementar la base de datos para los módulos Gestión de Usuarios y Sistema Estadístico del Sistema Integrado de Gestión Estadística.
- Diseñar el acceso a datos para el módulo Entrada de Datos del Sistema Integrado de Gestión Estadística.

Hipótesis:

Un diseño adecuado de la base de datos para el Sistema Integrado de Gestión Estadística facilita el desempeño del sistema y ayuda al proceso de desarrollo del proyecto de la Oficina Nacional de Estadística.

Estructuración del trabajo:

Capítulo 1: En este capítulo se realiza un estudio del arte del tema de las bases de datos, se analizan las herramientas que existen para el diseño de las bases de datos, los principales gestores de bases de datos y las herramientas más usadas para el diseño de la capa de acceso a datos, entre otros aspectos.

Capítulo 2: En este capítulo, basado en el estudio realizado en el capítulo anterior, se seleccionan y justifican cuáles herramientas se emplearán en el desarrollo de la solución. Además se explica la solución propuesta.

Capítulo 3: En este capítulo se realiza la validación del diseño de la base de datos y se analiza el nivel de normalización en que se encuentra el esquema de relación de la base de datos.

Capítulo 1: Fundamentación teórica

Los Sistemas de Bases de Datos ocupan hoy en día un lugar muy importante en todos los sistemas de información, ya que prácticamente todas las aplicaciones o soluciones a problemas utilizando informática hacen uso de las bases de datos. Este capitulo encierra varios aspectos relacionados con las bases de datos, entre los que están: el estudio de los principales gestores de base de datos, las herramientas para el diseño de las base de datos, se enuncian varios conceptos de base de datos definidos por diferentes personalidades a nivel mundial. Se realiza un estudio de las herramientas utilizadas para la generación de la capa de acceso a datos en las aplicaciones. Se exponen varias clasificaciones de las mismas atendiendo a diferentes criterios. Se explican además las diferentes fases por las que se debe transitar mediante el diseño de una base de datos, las cuales son: diseño conceptual, diseño lógico y diseño físico. También se explican dos de los modelos más utilizados en la actualidad para diseñar base de datos, estos son el modelo relacional y el modelo orientado a objetos; así como los problemas de incompatibilidad que existen entre ambos modelos y dos de las soluciones que hay para resolver dichos problemas.

1.1 Marco teórico.

Antes del surgimiento de las bases de datos, la manera en que se almacenaba y trabajaba con la información era en los sistemas de ficheros. Un sistema de ficheros es un grupo de programas con los que trabajan los usuarios finales, donde cada programa controla sus propios datos, es decir, que no hay un lugar físico centralizado donde almacenen los datos y al cual puedan acceder todos los que lo necesiten. Esto trae consigo dificultades tales como: separación y aislamiento de los datos, duplicación de datos, dependencia de datos, formatos de ficheros incompatibles, consultas fijas y proliferación de programas de aplicación. [1]

Luego aparece la necesidad de realizar un trabajo más efectivo con la información y es que surgen las bases de datos y los sistemas de gestión de bases de datos. Existen varios criterios o definiciones de las bases de datos enunciadas por diferentes personalidades en el mundo entre las que se encuentran:

"Un sistema de bases de datos es básicamente un sistema computarizado cuya finalidad general es almacenar información y permitir a los usuarios recuperar y actualizar esa información con base en peticiones". [2]

"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". [3]

"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". [3]

"Conjunto estructurado de datos registrados sobre soportes accesibles por ordenador para satisfacer simultáneamente a varios usuarios de forma selectiva y en tiempo oportuno". [3]

"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 atender todas las necesidades de los diferentes usuarios". [3]

A modo de resumen las bases de datos son una colección de datos aparentes usados por el sistema de aplicaciones de una determinada entidad, son un conjunto de información relacionada que se encuentra agrupada o estructurada. Un archivo por sí mismo no constituye una base de datos, sino más bien la forma en que está organizada la información es la que da origen a la base de datos. Pueden definirse también como una colección de datos organizada para dar servicio a muchas aplicaciones al mismo tiempo, al combinar los datos de manera que parezcan estar en una sola ubicación.

En el proceso de desarrollo de un software actualmente juega un papel fundamental ***el rol de diseñador de base de datos***, entre sus principales responsabilidades se encuentran:

- Tener una total comprensión del funcionamiento de la entidad para la cual se desarrolla el software.
- Tener conocimiento de las personas que intervienen en el negocio de dicha entidad, para con esta información poder hacer el diseño lógico
- Identificar en el diseño lógico los datos que se van a almacenar en cada tabla, la forma en que se van a relacionar y las restricciones que van a tener en los mismos.
- Definir cuáles son los usuarios que tendrá la base de datos, así como el nivel de acceso que tendrá cada uno de ellos según el rol que desempeñen.

1.2 Clasificaciones de las bases de datos.

De acuerdo a varios criterios, las bases de datos pueden ser clasificadas de diferentes formas:

1.2.1 Según la forma en que cambia la información almacenada:

Bases de datos estáticas: Son aquellas bases de datos en las que no se puede modificar la información que está guardada, es decir, solo se puede consultar, son utilizadas preferentemente para guardar datos históricos con los que luego se podrán realizar estudios sobre su comportamiento en el transcurso del tiempo, lo cual ayudará a tomar decisiones.

Bases de datos dinámicas: Son aquellas bases de datos en las que la información que está almacenada está en constante cambio, ya que sobre ella se pueden realizar operaciones tales como: adicionar, actualizar, eliminar, etc., así como las operaciones de consultas. [4]

1.2.2 Según la información que almacenan:

Bases de datos bibliográficas: Almacenan la información necesaria para poder localizar la fuente primaria. Un registro típico de una base de datos bibliográfica contiene información sobre el título, fecha de publicación, autor, editorial, de una determinada publicación, etc. Puede contener un fragmento de la publicación original, pero nunca el texto completo.

Bases de datos de texto completo: Almacenan las fuentes primarias, es decir, guardan toda la información íntegra de las fuentes primarias, no fragmentos, ni referencias.

Directorios: Este es el caso de directorios digitales como los que edita la Empresa de Telecomunicaciones de Cuba (ETECSA) con los números telefónicos de sus usuarios, ya sean privados o estatales, donde se puede encontrar el nombre al que pertenece dicho número y la dirección dónde se encuentra.

Banco: Estas según el formato de la información que almacena puede ser: de audio, video, imágenes, multimedia, etc. [4]

1.3 Fases del diseño de base de datos.

El proceso de diseño de una base de datos cuenta con tres fases fundamentales: Diseño Conceptual, Diseño Lógico y Diseño Físico.

1.3.1 Diseño Conceptual:

El objetivo del diseño conceptual, también conocido como modelo conceptual, y que constituye la primera fase de diseño, es obtener una buena representación de los recursos de información, con

independencia de usuarios y aplicaciones en particular y sin considerar aspectos como eficiencia de la computadora. Esta primera fase consta de dos momentos: análisis de requisitos, donde se centra el trabajo en definir qué es lo que se va a representar, y la conceptualización, donde se piensa en cómo se va a proceder para representar lo antes definido.

1.3.2 Diseño Lógico:

El diseño lógico parte del esquema conceptual y genera el esquema lógico. Un esquema lógico es la descripción de la estructura de la base de datos que puede procesarse por Sistema Gestor de Base de Datos. El objetivo principal del diseño lógico es obtener un esquema lógico eficiente en cuanto a operaciones de consulta y actualización. El modelo relacional es el único modelo que ha permitido abordar la fase de diseño lógico aplicando una teoría formal.

1.3.3 Diseño Físico:

El objetivo del diseño físico es conseguir una instrumentación lo más eficiente posible del esquema lógico. Para esto se analizan aspectos como las características del Sistema Operativo, el Sistema Gestor de Base de Datos, la Herramienta para realizar el diseño, aspectos relacionados con el rendimiento y los requisitos de procesos así como las características del hardware, en fin, cualquier factor cercano con la computadora, para con ello lograr optimizar el consumo de recursos, minimizar el espacio de almacenamiento, proporcionar la seguridad máxima, disminuir los tiempos de respuesta y evitar las reorganizaciones.

1.4 Modelos para el diseño de bases de datos.

Existen varios modelos que con frecuencia son utilizados en la actualidad para el diseño de las bases de datos. Entre los modelos más utilizados se encuentran: Modelo Orientado a Objetos y Modelo Relacional.

1.4.1 EL Modelo Orientado a Objetos.

El modelo de objetos conjuga de forma centralizada los conceptos de abstracción, jerarquía, encapsulación, modularidad, persistencia, tipos y concurrencia, donde los primeros cuatro conceptos son fundamentales, lo cual significa que si un modelo carece de estos elementos pues no es orientado a objetos. Además se basa en los principios de la ingeniería. Este modelo refleja una nueva forma de pensar acerca de la de descomposición, haciendo que el diseño orientado a objetos sea diferente a la forma que existía de diseño estructurado hasta el momento, y que por consiguiente su arquitectura sea diferente también. Esto ocurre sencillamente porque los métodos de diseño orientado a objetos utilizan la programación orientada a objetos, mientras que el diseño estructurado se basa en la programación estructurada. El desarrollo del diseño orientado a objetos no niega las ideas existentes, sino que se basa en ellas y las mejora. Un modelo orientado a objetos tiene obligatoriamente que cumplir con lo siguiente: la estructura básica de trabajo son los objetos, no algoritmos; donde cada objeto no es más que una instancia de una clase ya definida y que dichas clases estarán relacionadas únicamente por relaciones de herencia. En caso de que no se cumpla uno de las condiciones antes mencionadas, pues no estaríamos en presencia de un modelo orientado a objetos. [5]

1.4.2 EL Modelo Relacional.

El modelo relacional de datos tiene varios aspectos que lo caracterizan, desde el punto de vista de estructural podemos plantear que este modelo organiza la información en forma de tablas y de esta manera es que sus usuarios la perciben y no de otra; desde el punto de vista de la integridad podemos decir que el modelo establece varias restricciones de integridad, las cuales deben ser cumplidas por todas sus tablas; y desde el punto de vista de la manipulación, el modelo relacional tiene definidos un grupo de operadores a través de los cuales interactúa con la información almacenada en las tablas de la base de datos y sus resultados los devuelven en tablas. Entre estos operadores se encuentran proyectar, restringir y juntar. Debido a que la información consultada en cada operación se encuentra en forma de tabla, al igual que la que devuelven, pues existe la posibilidad de concatenar operadores unos con otros. Este modelo además está capacitado para hacer procesamiento de conjunto. Esto significa que al devolver el resultado de cualquier operación, el resultado lo devuelve en tablas que están conformadas por un conjunto de filas, no por filas individuales. El modelo relacional está compuesto por 5 componentes que lo

identifican, los cuales menciono a continuación: un conjunto abierto de tipos escalares, un generador de tipos de relación y una interpretación propuesta de dichos tipos, herramientas para definir variables de dichos tipos de relación generados, una operación asignación relacional para asignar valores de relación a las variables de relación mencionadas, un conjunto abierto de operadores relacionales genéricos para derivar valores de relación de otros valores de relación. [2]

1.5 Principales gestores de bases de datos.

No es posible hablar de las bases de datos sin mencionar a los Sistemas de Gestión de Bases de Datos (SGBD). Los SGBD son un conjunto de programas que permiten la implementación, acceso y mantenimiento de la base de datos. El SGBD, junto con la base de datos y los usuarios, constituyen el Sistema de Base de Datos. Los SGBD se pueden agrupar en tres grandes grupos: SGBD Relacionales, SGBD Orientado a Objetos y SGBD Objeto/Relacional.

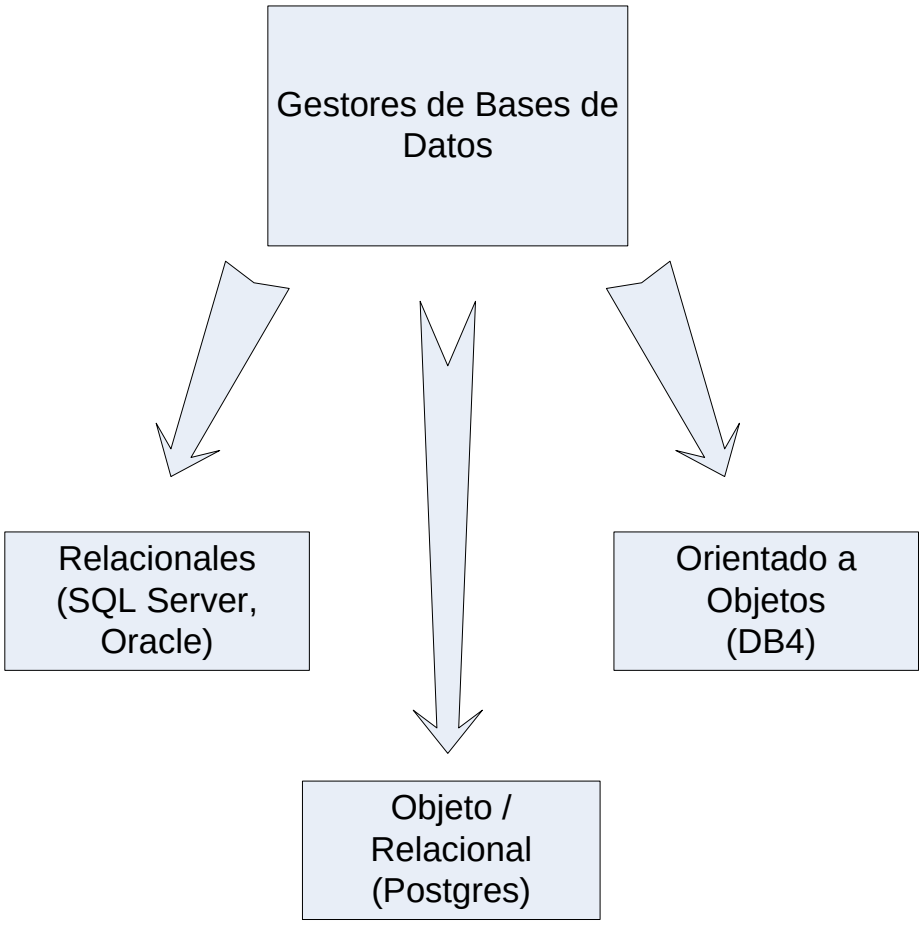

Figura 1 Clasificación de los Sistemas Gestores de Bases de Datos.

1.5.1 Sistemas Gestores de Bases de Datos Relacionales

Entre los SGBD Relacionales podemos encontrar el Lenguaje de Consulta Estructurado o Structure Query Lenguaje Server (SQL Server) y el Oracle.

SQL Server.

SQL Server fue desarrollado por Microsoft y permite la gestión de un entorno de base de datos relacional. A través de una interfaz cómoda y fácil de trabajar SQL Server abarca tanto el área de diseño como la de administración. Es nombrado SQL porque hace uso de este lenguaje para la definición y manejo de los datos, y se llama Server porque dispone de una parte servidora que es responsable de

atender a los procesos clientes, que son aquellos que realizan las peticiones al servidor, cumpliendo con una arquitectura cliente-servidor.

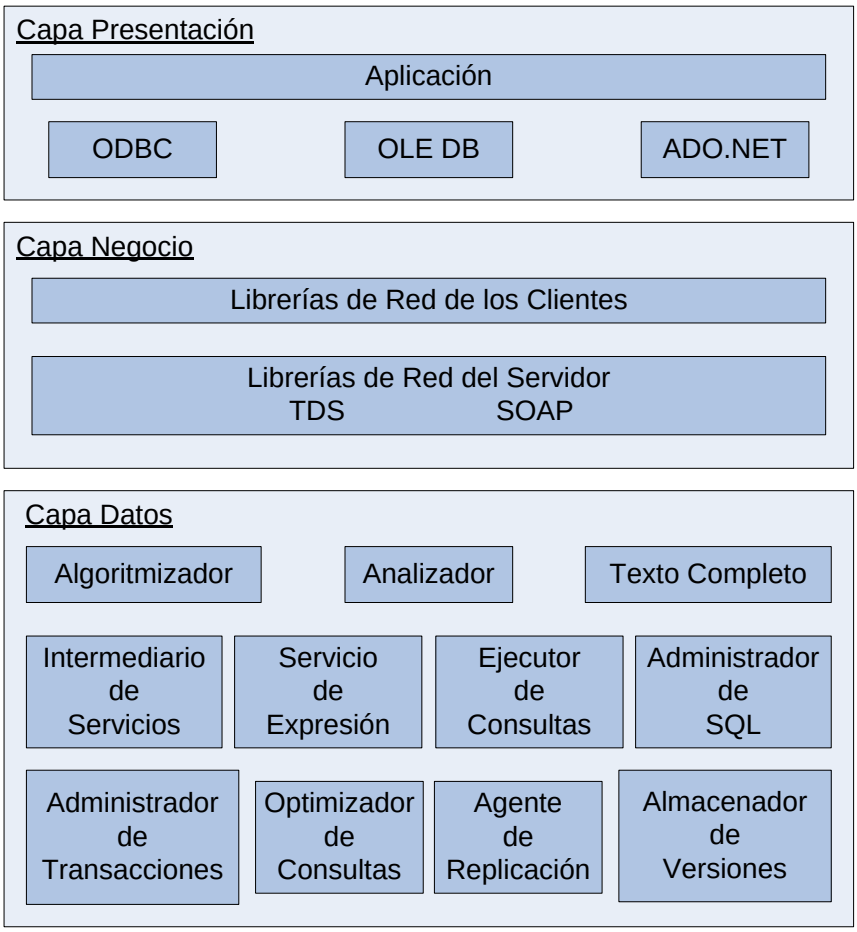

Figura 2 Arquitectura Cliente-Servidor del SQL Server.

Como podemos observar en la Figura 2, la arquitectura cliente-servidor organiza o estructura el sistema en tres capas: una capa de presentación o capa de los clientes, una capa de lógica del negocio o red y una capa de datos o servidora.

En la capa de presentación se encuentra una interfaz de usuario y los métodos que utiliza para la conexión con el servidor de datos. Las aplicaciones que se desarrollan haciendo uso de SQL Server

pueden utilizar diferentes métodos de conexión entre los que se encuentran ODBC (Open Database Connectivity o Conectividad Abierta de la Base de Datos), OLE DB (Object Linking and Embedding for Databases) y ADO.NET (ActiveX Database Objects).

El ODBC constituye uno de los métodos más populares y ampliamente utilizados de conexión y está compuesto por el nombre de una fuente de datos, un abastecedor y por opciones. El abastecedor es una biblioteca o librería provenida por la base de datos o por el desarrollador del software que esta permitiendo las conexiones a su software. El nombre de una fuente de datos es un grupo de opciones o ajustes que se encuentran almacenados en el registro de entrada de un usuario específico, lo cual será genérico para todo el sistema, o estarán almacenados en un archivo. Estas opciones o ajustes contienen el nombre del servidor, las credenciales de seguridad y el nombre de la base de datos. Las desventajas del ODBC son su velocidad y la necesidad de configurar a cada usuario que desee utilizarla.

Microsoft crea el OLE DB para tratar algunos defectos de ODBC y contemplar un ambiente computacional distribuido. OLE DB es básicamente un sistema de interfaces de componentes de modelos objetos para aplicaciones que acceden a funcionalidades y datasets de bases de datos. Con una conexión OLE DB, la máquina del usuario no necesita crear un nombre de fuentes de datos, y el desarrollador puede crear una conexión con el servidor, incluyendo en el código la localización y las credenciales de seguridad.

En la capa de red o del negocio es donde se definen y aplican las reglas del negocio y permite además la comunicación entre la capa de presentación y la capa de datos. Esta capa está compuesta por las librerías de red de los clientes y las librerías de red del servidor que permitirán el paso de información tanto de la capa de presentación hacia la capa de datos, como de la capa de datos para la capa de presentación. Las librerías de red, como es el caso del protocolo TCP/IP que tienen características, entre las que se encuentran el cifrado y la comprensión de los datos, entre otras, con la peculiaridad de que si se permiten en el cliente, tienen que estar presentes en el servidor. Entre las librerías de red del servidor se encuentran los TDS (Tabular Data Stream) y SOAP (Simple Object Access Protocol). TDS es un protocolo usado para entrar y extraer datos del servidor. SOAP es un protocolo que se utiliza para las conexiones a internet y permite que el SQL Server se comunique usando HTTP y XML y también proporciona los puntos finales que usa el SQL.

En la capa de datos o capa servidora se controlará el almacenamiento, cambio y mantenimiento de la información generada por las otras dos capas. Para ello esta capa está dividida en varios elementos. Entre estos elementos se encuentran los siguientes: Algoritmizador, Analizador, Texto Completo, Intermediario de Servicios, Servicio de Expresión, Administrador de SQL, Optimizador de Consultas, Ejecutor de Consultas, Administrador de Transacciones, Almacenador de Versiones y Agente de replicación, entre otros. El Algoritmizador es el que recibe la solicitud de información que se le hace al servidor, y reparte las tareas a diferentes operadores que están relacionados y que formarán un árbol de trabajo para procesar la respuesta. Determina las tareas que se pueden hacer de forma paralela y además determina la localización de los elementos en la memoria caché. Además determina la mejor manera de trabajar con las funciones de sub consultas y agregación que constituyen dos de las funciones más difíciles de las bases de datos. La información jerárquica se encuentra en un archivo XML Algoritmizador. El Analizador, cuando una solicitud al servidor se divide en varias tareas, se deshace de las tareas más pequeñas, incluso las más futuras. Al igual que el Algoritmizador, cuenta con un archivo XML para el proceso llamado XML Analizador. La función Texto Completo hace posible el análisis de la secuencia de texto, en caso de que la solicitud al servidor este relacionada con preguntas con texto completo. Intermediario de Servicios, este componente le permite a los desarrolladores desconectar aplicaciones, y luego manualmente volver a procesarlas cuando sea necesario procesarlas. Servicio de Expresión es el proceso que controla las variables y los parámetros con los que se tiene que trabajar en algunas consultas, cuando se encuentran implicados procedimientos almacenados que tienen que trabajar con variables. El Administrador de SQL se encarga de los procedimientos almacenados y de cómo se ejecutan. El Optimizador de Consultas es el proceso que determina los índices, la estadística y las trayectorias óptimas para dar respuesta a la solicitud en proceso. El Ejecutor de Consultas crea un plan final, y luego el plan final de ejecución, después de haberse realizado la comprobación de la sintaxis y se haya optimizado. Este plan se puede visualizar de forma gráfica o en texto en el SQL Server Management Studio. El Administrador de Transacciones es el proceso que maneja operaciones de registración de datos y la recuperación de las bases de datos. Este es el proceso que protege los datos móviles entre el registro de transacción y la base de datos y también los roles antes y después de iniciar. El Almacenador de Versiones es quién va a manejar las transacciones relacionadas con las diferentes versiones de una información almacenada en una misma base de datos, dando la posibilidad de que dos usuarios obtengan resultados diferentes de la misma base de datos. El Agente de Replicación es el encargado de manejar

las operaciones de réplica de la información. Durante las operaciones de réplica, SQL Server necesita mover grandes cantidades de información, ya sea internamente o del sistema a otro sistema. [6]

1.5.2 Sistemas Gestores de Bases de Datos Objeto/Relacional.

Entre los SGBD Relacionales podemos encontrar el Postgres.

Postgres.

Postgres es un SGBD identificado como Objeto/Relacionales aunque tiene algunas características pertenecientes a las bases de datos orientadas a objetos. Este SGBD se comenzó a implementar en el año 1986 y es comúnmente llamado PostgreSQL. Es considerado el gestor de bases de datos de código abierto más avanzado en la actualidad, por su gran compatibilidad con diferentes lenguajes de programación entre los que se encuentran: C++, Python, Java, C, Perl, entre otros. También por su capacidad de soportar la mayoría de las sintaxis SQL, como es el caso de: transacciones, subconsultas, además de funciones y tipos definidos por el usuario. Además por la propiedad que posee de ofrecer control de concurrencia multi-versión. Trabaja con conceptos básicos como: clases, tipos, herencias y funciones, con los que ofrece una gran potencia. PostgreSQL es el resultado del mejoramiento de versiones que lo anteceden. Posee características como integridad transaccional, reglas (rules), restricciones (constraints) y disparadores (triggers) que le aportan flexibilidad y potencia adicional. Entre las principales mejoras que brinda este gestor se encuentran las siguientes:

- Posibilita tener copias de seguridad en caliente mientras que la base de datos continua disponible para realizar consultas.
- Con el control de concurrencia multi-versión se sustituyeron los bloqueos de tablas, lo cual hace posible que los accesos de solo lectura puedan, durante la actualización de registro, leer datos consistentes.
- Se han agregado funcionalidades en línea con el estándar SQL92, incluyendo identificadores entrecomillados, claves primarias, forzado de tipos cadenas literales y conversión de tipos y entrada de enteros binarios y hexadecimales.
- Se han podido implementar importantes características del motor de datos, incluyendo subconsultas, valores por defecto, restricciones a valores en los campos (constraints) y disparadores (triggers).

- Se han mejorado además, los tipos internos, incluyendo nuevos tipos de fecha/hora de rango amplio y soporte para tipos geométricos adicionales.[7]

El Postgres cumple con una arquitectura cliente-servidor.

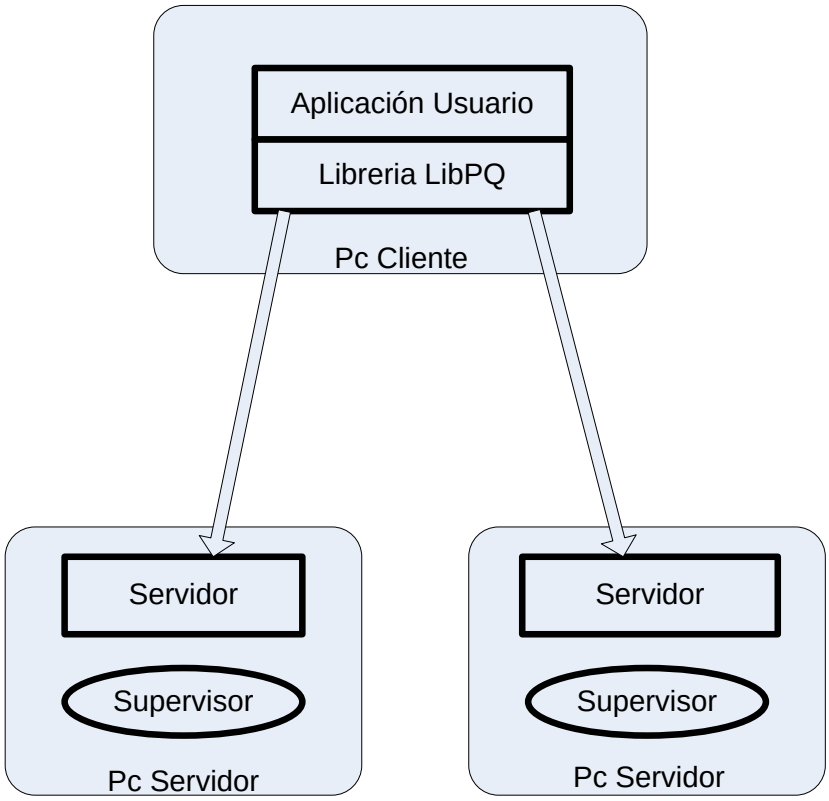

Figura 3 Arquitectura de Postgres.

Como podemos observar en la Figura 3, el Postgres cuenta con tres procesos fundamentales: la aplicación del usuario, el supervisor y uno o más servidores de bases de datos. La relación que existirá entre ellos funciona de la siguiente forma: una aplicación realiza una petición de conexión a una base de datos. Dicha petición es gestionada mediante el supervisor. La aplicación hace dicha petición a través de la librería libpq. El supervisor es el encargado de manipular una colección de base de datos sobre un host simple, también entre sus responsabilidades se encuentra la de hacer una conexión a la aplicación creando un nuevo proceso servidor. Luego de que el supervisor establece la conexión, la aplicación

cliente y el servidor de base de datos se comunicarán sin la presencia del mismo. Mediante la librería libpq, la aplicación del usuario puede establecer varias conexiones a servidores de bases de datos. En esta arquitectura se cumple que el servidor de bases de datos y el supervisor siempre se ejecutan en la misma computadora, mientras que la aplicación del usuario se ejecuta en cualquier lugar, debido a que esta última no soporta conexiones multihilos. [8]

1.5.3 Sistemas Gestores de Bases de Datos Orientado a Objetos.

Entre los SGBD Orientado a Objetos podemos encontrar el DB4.

DB4.

El DB4 es un gestor de base de datos de alto rendimiento que soporta aplicaciones distribuidas, con arquitectura Cliente/Servidor. Además puede ser utilizado en entornos Java o .NET. Entre sus principales características se encuentran: alto rendimiento, fácil de implementar, consume pocos recursos, brinda portabilidad y confiabilidad a los datos que almacena. En el DB4 el modelo de clases representa el esquema de base de datos, lo que posibilita que se elimine el proceso de diseño, implementación y mantenimiento de la base de datos. DB4 nos ofrece tres tipos de consultas, estas son: consultas dinámicas basadas en nodos, consultas por ejemplo o prototipo y consultas a datos con lenguaje nativo. El DB4 les permite a los usuarios trabajar con la base de datos de diferentes maneras: realizar una conexión distribuida entre servidores de bases de datos, realizar una conexión a base de datos desde un cliente (cliente/servidor), y trabajar con la base de datos incluida dentro de la propia aplicación cliente. [9]

De los Sistemas Gestores de Bases de Datos analizados anteriormente, el que utilizo para almacenar la información del SIGE es el SQL Server 2000.

1.6 Herramientas CASE que se utilizan para el diseño de bases de datos.

Las herramientas CASE son de gran ayuda para el desarrollo de un software. Estas son un grupo de programas que utilizan las personas que intervienen en el desarrollo de un software, como es el caso

de los diseñadores, desarrolladores, analistas, entre otros, durante las fases del desarrollo del software (análisis, diseño, implementación, etc.), para agilizar y facilitar su trabajo, ya que dichas herramientas proveen de métodos, técnicas y utilidades que ayudan al perfeccionamiento del desarrollo de sistemas de información, de forma total o parcialmente.

Las herramientas CASE, a lo largo de su desarrollo, han ido aumentando en complejidad. En los inicios era solo un procesador de palabras que se usaba para crear y manipular documentación. Luego fueron adquiriendo técnicas gráficas y diagramas de flujo de datos. En la actualidad, permiten la creación y modificación de diagramas con gran facilidad, además de automatizar varias actividades como generación de código para el desarrollo de software, lo cual mejora considerablemente la calidad, el rendimiento, la utilidad y fiabilidad de las herramientas CASE.

Entre las herramientas CASE más utilizadas para el diseño de base de datos se encuentran:

1.6.1 ERwin:

PLATINUM ERwin es una CASE empleada en el diseño de base de datos, con la cual se puede hacer un trabajo productivo por las facilidades que brinda para la generación y mantenimiento de aplicaciones de forma sencilla para el diseñador. ERwin permite realizar el diseño del modelo lógico de los requerimientos de información, así como el diseño del modelo físico, ya con nivel mayor, refinando las características de la base de datos diseñada. Con esta herramienta CASE es posible visualizar la estructura de la base de datos diseñada, lo cual tiene como ventaja que el diseñador pueda observar en su totalidad el trabajo realizado, para realizar un análisis y si fuera necesario hacer cambios en busca de optimizar el diseño final. Esta herramienta permite generar, de forma automática, las tablas y el código referente a los stored procedure (procedimientos almacenados) y triggers para los principales tipos de bases de datos. La migración automática garantiza la integridad referencial de la base de datos, es decir, evita la pérdida de información durante este proceso. ERwin puede hacer una conexión entre dos bases de datos, una diseñada y otra sin diseño, estableciendo una transferencia entre ellas y la aplicación de ingeniería reversa. Mediante esta conexión puede automáticamente generar índices, vistas, tablas, reglas de integridad referencial (llaves foráneas, llaves primarias), restricciones de dominios y campos y valores

por defecto. ERwin básicamente soporta bases de datos del tipo relacionales SQL, y es compatible con otras como: Oracle, Microsoft SQL Server, Sybase, Microsoft Access, dBase, FoxPro, Informix, SQL Base y SQL Anyware, entre otras. Con un mismo modelo es posible generar múltiples bases de datos o hacer una conversión de una aplicación de una base de datos a otra. Es compatible con los sistemas operativos de la familia Windows, como Windows 95, Windows 98, Windows XP y Windows NT. [10]

1.6.2 EasyCASE:

EasyCASE Profesional es un producto destinado a la generación de esquemas de base de datos e ingeniería reversa. Su objetivo es proporcionar a los usuarios una solución comprensible para el diseño, documentación y consistencia del sistema que se va a desarrollar. Brinda la posibilidad de automatizar las primeras fases del desarrollo de un software (análisis y diseño), para lograr el desarrollo de un trabajo óptimo y eficaz. Forma parte del grupo de herramientas multi-usuarios, permitiendo que varias personas trabajen en un proyecto al mismo tiempo. Esta herramienta permite la conexión de varias personas a un servidor donde está la información que necesitan los integrantes del proyecto para su trabajo. Esta conexión se hará de forma segura, ya que la herramienta controlará el nivel de acceso que podrá tener cada usuario, según el rol que desempeñe, a través del diagrama y diccionario de los datos que bloquean por niveles al registro, al archivo y al proyecto. EasyCASE soporta las siguientes base de datos: Oracle, Postgres, SQL Server, ANSI SQL, Fox Pro, SQL Base, Access, Paradox, Sybase, Clipper, entre otras. [10]

1.6.3 Oracle Designer:

Es un grupo de aplicaciones que automatizan la construcción de aplicaciones cliente/servidor y permiten además el almacenamiento de las definiciones que necesita el usuario de forma rápida y sencilla. Oracle Designer hace uso de un repositorio central para almacenar, en cualquier fase de desarrollo, toda la información generada por cualquier herramienta de Oracle Designer, posibilitando un trabajo fácil para la dirección del proyecto y su equipo de desarrollo. Esta herramienta CASE, puede en el lado del servidor definir, generar y capturar el diseño, por conexión Oracle, los siguientes tipos de bases de datos: Oracle8, Oracle7, Personal Oracle Lite, Microsoft SQL Server, Sybase y ANSI 92, entre otras. [10]

1.6.4 Rational Rose Data Modeler:

Es una herramienta que le permite al diseñador de base de datos realizar de manera fácil y rápida su trabajo, ya que permite una transformación flexible entre los modelos lógicos y físicos haciendo uso de un entorno de modelado sofisticado y de capacidades avanzadas de modelado visual para bases de datos. Además le permite a los desarrolladores visualizar la forma en que la aplicación o el software que se está desarrollando accederán a la información en la base de datos, para detectar y resolver posibles problemas que se pueden presentar en dicho proceso. Cuenta con una herramienta llamada Unified Modeling Languaje que permite la comunicación entre los diseñadores de base de datos que trabajan con el modelo relacional y el resto del equipo de desarrollo de la aplicación. Los sistemas operativos y plataformas con que es compatible son: Windows 2000, Windows NT y Windows XP.

La herramienta CASE que utilizo para realizar el diseño de las bases de datos del SIGE es el Erwin 7.0.

1.7 Herramientas que se utilizan para diseñar e implementar el acceso a los datos.

Cuando un sistema cuenta con una base de datos para almacenar la información, tiene que haber una capa de acceso a datos que les permita a los usuarios la interacción con dicha información. Para implementar la capa de acceso a datos existen varias herramientas como son el NHibernate, el TierDeveloper, entre otras. A continuación explicaremos algunas de las características de ambas herramientas.

1.7.1 TierDeveloper.

El TierDeveloper es una herramienta de generación de código que genera el negocio de .NET y los objetos de datos, ASP.NET apps y Windows Forms apps. TierDeveloper les permite a los usuarios definir los objetos de datos luego de haber mapeado una o más tablas de la base de datos, lo cual ayuda a extraer la información que se necesita en el menor tiempo posible. Se puede definir los objetos de datos

usando esquemas existentes en la base de datos, generar el código fuente de los objetos de datos así como su estructura de despliegue, y prueba los objetos de datos con una aplicación que se genera fácilmente. Para obtener el máximo de beneficios del TierDeveloper, es necesario trabajar con una arquitectura .NET, programando en lenguajes como C# y Visual Basic.NET, debe estar instalado el Internet Information Service y trabajar con gestores de base de datos como Oracle, DB2, SQL Server y Microsoft Access. [11]

1.7.2 NHibernate

La herramienta NHibernate ayuda a la comprensión entre los modelos relacional y orientado a objetos. Además facilita el trabajo con las asociaciones entre los objetos persistentes. Les permite a los usuarios hacer varias operaciones en un ambiente total orientado a objetos y lo abstrae de realizar las consultas SQL. Con el uso de NHibernate le damos la facilidad a nuestro cliente de, si le fuera necesario, hacer un cambio de un software propietario a un software libre y con ello el gestor de base de datos, pues esta herramienta puede cambiar de manera fácil y sin problemas de pérdida de información de un gestor a otro, haciendo el traspaso de información porque es compatible o soporta cualquier gestor de base de datos. Proporciona mantenibilidad a las aplicaciones, al automatizar la persistencia y contar con un almacenador (buffer) que funciona como intermediario entre los modelos relacional y orientado a objetos. Al tener automatizada la persistencia, permite mejorar la productividad en la aplicación ya que se le puede dedicar más tiempo al trabajo en otras capas como la del negocio. [12]

1.8 Análisis del problema de la incompatibilidad entre el modelo relacional y el modelo orientado a objetos.

Un elemento que refleja el problema de la incompatibilidad entre el modelo relacional y el modelo orientado a objetos, es el llamado paradigma mismatch. Este paradigma se explicará abordando diferentes problemas que existen como: el problema de la granularidad, el problema de los subtipos, el

problema de la identidad, el problema referente a las asociaciones y el problema de la navegación en el gráfico de objetos. El paradigma mismatch se refleja cuando en el desarrollo de una aplicación estamos en presencia de dos clases que se relacionan mediante una relación de 1 a muchos y que en esta relación podemos navegar en ambas direcciones y comenzamos a agregar nuevas entidades con sus relaciones a nuestra aplicación. El problema real está cuando en una de estas clases representamos un atributo como un simple string y este atributo puede ser desglosado en varios atributos, como por ejemplo en una clase estudiante, se encuentre un atributo sobre sus datos personales que se llame dirección, y este atributo puede desglosarse en calle, municipio, provincia, país, etc. En la mayoría de los sistemas actuales se utilizan estos atributos desglosados como hemos explicado anteriormente. Podríamos darle solución a este problema adicionando estos atributos en los que se ha desglosado el atributo dirección, directamente a la clase estudiante, pero cabe la posibilidad de pensar que otras clases del sistema utilicen también el atributo dirección. Seria razonable entonces crear una clase aparte para guardar la información referente a la dirección. Frecuentemente se trata la información de la dirección en la tabla estudiante como una columna más, lo cual nos evita el uso del operador JOIN en una simple consulta donde se quiere obtener la información de un estudiante con su dirección. Otra solución pudo haber sido crear un tipo de dato definido en el SQL para representar direcciones y así utilizaríamos una sola columna de este nuevo tipo en la tabla estudiante en lugar de usar varias nuevas columnas para ello.

1.8.1 El Problema de la Granularidad.

Granularidad se refiere al tamaño de los objetos con los que se está trabajando. Cuando estamos hablando de los objetos y las tablas de la base de datos, el problema de granularidad significa que los objetos persistentes pueden tener varios tipos de granularidad en tablas y columnas que son intrínsecamente limitadas en granularidad.

Supongamos que se adiciona un nuevo tipo de dato en las bases de datos para almacenar objetos. Cuando chequeamos el tipo de columna definida por el usuario (user-defined column types, UDT) en el Sistema Gestor de Bases de Datos SQL, pues nos damos cuenta que pueden surgir varios problemas. UDT soporta uno, del número de las llamadas extensiones objetos relacionales del SQL. El UDT es un concepto que aun no está claro en la mayoría de los sistemas gestores de bases de datos SQL y seguramente no es posible aplicarlos entre diferentes sistemas. El SQL soporta UDT pero de manera muy

deficiente. Por esta razón el uso de los UDT en la actualidad no es de común práctica en la industria. Es difícil encontrar sistemas que hagan un empleo extenso de los UDT. Por lo tanto llegamos a la conclusión que no es posible almacenar objetos en una simple columna como si fuera equivalente a algún tipo de dato definido en el SQL. La solución para este problema es almacenar la información en nuevas columnas de la base de datos con tipos de datos que admite el SQL, tales como: string, bolean, numeric, entre otros. Resulta que el problema de la granularidad no es difícil de resolver. Incluso no es un problema que se observa con frecuencia en los sistemas actuales. [12]

1.8.2 El Problema de los Subtipos.

El problema de los subtipos lo vemos presente en aplicaciones que utilizan la herencia en su programación (orientada a objetos). La herencia la implementamos utilizando clases y subclases. Dichas clases tienen propiedades y características generales que son similares o comunes para todas y cada una de las subclases que heredan de ella. Además cada una de estas subclases tiene características y métodos que las diferenciaran a una subclase de otras. Es válido destacar que en SQL no es posible implementar la herencia directamente. Pero no todo está perdido, mientras implementan como trabajar la herencia en el modelo orientado a objetos, existe la posibilidad de trabajar con el polimorfismo. En una relación entre dos clases donde dicha relación es de 1 a muchos, podemos decir que existe una asociación polimórfica. Con el término asociación polimórfica nos referimos al hecho de que una clase puede estar asociada a cualquiera de las subclases que heredan de una clase abstracta. Esta propiedad es llamada también consulta polimórfica. Podemos observar también que el SQL no es capaz tampoco de representar una asociación polimórfica, debido a que el SQL define que una llave foránea estándar hace referencia exactamente a una tabla, no es correcto definir que una llave haga referencia a varias tablas. Así pues, el paradigma mismatch de los subtipos es uno en el cual la estructura de la herencia en un modelo orientado a objetos debe persistir en una base de datos SQL que nos ofrezca una estrategia de herencia. [12]

1.8.3 El Problema de la Identidad.

El problema de la herencia lo podemos encontrar presente en muchas de las aplicaciones de la actualidad. Dicho problema surge cuando estamos frente a dos objetos, los cuales son idénticamente iguales. En el modelo orientado a objetos se tienen en cuenta dos criterios de igualdad:

-Identidad del objeto (atendiendo al criterio de la equivalencia de la posición en memoria, lo cual se comprueba con la condición a==b).

-Igualdad de objetos (atendiendo al criterio de que los objetos sean iguales por su valor, lo cual se comprueba con el método equals ()).

Es común encontrar objetos (no idénticos) representando la misma fila de la base de datos. Algunos de los problemas que se presentan vienen dados por la incorrecta implementación del método equals () como una clase persistente. En otros casos nos encontramos con la situación de que por ejemplo, a una tabla estudiante le asignan como llave primaria (la cual identifica la tabla), el atributo nombre del estudiante. En este caso, dicha llave primaria no es la correcta, puesto que podemos encontrar dos estudiantes con el mismo nombre. Otra razón por la que dicha llave no es correcta es que las llaves primarias de una tabla deben ser a su vez llaves sustituibles (surrogate keys). Esto significa que dichas llaves solo cumplen la función de identificar a la tabla, que no pueden tener otro tipo de significado para la tabla lo cual permite que pueda ser cambiado con facilidad si alterar la información almacenada en dicha tabla. Para cumplir con esto, cuando no tenemos un atributo que cumpla esta funcionalidad, como es el caso del número del solapín de un estudiante, pues SQL permite la creación asignar atributos o códigos identificadores, cuyo valor es generado por el sistema. [12]

1.8.4 El Problema referente a las Asociaciones.

Existen diferencias entre la manera de representar las asociaciones en un modelo orientado a objetos y en el modelo relacional. En el modelo orientado a objetos las asociaciones representan las relaciones entre las entidades. El mapeo de asociaciones y la gestión de las asociaciones de la entidad son conceptos centrales de cualquier solución de la persistencia en objetos. Los lenguajes orientados a objetos representan asociaciones usando referencias del objeto y colecciones de referencias del objeto. En el mundo relacional, una asociación se representa como una llave extranjera de una columna, haciendo copias de los valores de estas llaves en las tablas implicadas en la relación. Existen diferencias

entre estas dos formas de representar las asociaciones. Las referencias del objeto son intrínsecamente direccionales; la asociación es a partir de un objeto al otro. Si una asociación entre los objetos es navegable en ambas direcciones, se debe definir la asociación dos veces, una vez en cada uno de las clases asociadas. Sin embargo, en el mundo relacional, las llaves extranjeras de las asociaciones no tienen una naturaleza direccional. De hecho, la navegación no tiene ningún significado para un modelo relacional de datos, porque puedes crear datos arbitrarios asociados, a través de operadores como el join y el projection entre las tablas. Realmente, no es posible determinar la multiplicidad de una asociación unidireccional mirando solamente las clases orientadas a objetos. Las asociaciones de los lenguajes orientados a objetos pueden tener una multiplicidad múltiple. Las asociaciones de las tablas, por otra parte, son siempre de uno a muchos o de uno a uno. Puedes ver la multiplicidad inmediatamente mirando la definición de la llave extranjera. Si deseas representar una asociación múltiple o de muchos a muchos en una base de datos relacional, debes introducir una tabla nueva, llamada tabla de la relación. Esta tabla no aparece en el modelo de objeto. [12]

1.8.5 El Problema de la Navegación en el Gráfico de Objetos.

Existe una diferencia fundamental entre la forma en que se accede a objetos y la que se accede a una base de datos relacional. La forma normal que se emplea para acceder a la información almacenada en objetos es haciendo una llamada al método que tiene dicho objeto implementado para realizar dicha función. Navegas desde un objeto hacia otro, buscando asociaciones entre instancias. Esta no constituye una manera eficiente de recuperar los datos en una base de datos relacional. Una forma simple de mejorar el proceso de acceso a los datos es minimizar el número de peticiones a la base de datos, reduciendo la cantidad de consultas SQL. Otras vías de solución incluyen el uso de procedimientos almacenados. Por lo tanto, el acceso eficiente a los datos relacionales usando SQL, generalmente requiere el uso de operadores JOIN entre las tablas de interés. El número de las tablas incluidas en el JOIN determina la profundidad del gráfico del objeto que puedes navegar.

Es necesario saber la porción del gráfico de objeto que tenemos pensado acceder cuando recuperamos una información final, antes de que comencemos a recorrer gráfico de objeto. Por otra parte, cualquier solución de la persistencia de objeto brinda la funcionalidad para obtener los datos de los objetos asociados solo cuando primeramente el objeto es accedido. Sin embargo, este estilo fragmentario

del acceso de los datos es fundamentalmente ineficiente en un contexto de bases de datos relacionales, porque requiere la ejecución de un operador select para cada nodo del gráfico de objetos. Este es el n+1 temido problema de selección. El paradigma mismatch en la manera en que accedemos a los datos el mundo orientado a objetos y en las bases de datos relacionales es quizás la fuente más simple y común de los problemas de funcionamiento en aplicaciones orientadas a objetos. Aun, cuando ha publicado innumerables libros y revistas, artículos aconsejando el uso del StringBuffer para la concatenación de string, parece imposible encontrar cualquier consejo sobre las estrategias para evitar el n+1 problema de selección. [12]

1.9 Posibles soluciones al problema de la persistencia.

En este epígrafe abordaremos dos de las soluciones más factibles para eliminar los problemas de la incompatibilidad entre el modelo relacional y el modelo orientado a objetos, anteriormente explicados. Estas soluciones son: Uso de bases de datos orientadas a objetos y Uso de mapeadores objeto/relacional.

1.9.1 Uso de bases de datos orientadas a objetos:

Para cuando se va a trabajar con el modelo orientado a objetos, seria ideal que existiera una forma de almacenar los objetos en una base de datos sin tener que modificar el modelo objeto. A mediados de los 90, los nuevos sistemas de bases de datos orientados a objeto ganaron publicidad. Un sistema gestor de base de datos orientado a objeto (OODBMS) es más parecido a la extensión de un ambiente de una aplicación, que a una base de datos externa. Un OODBMS ofrece generalmente una implementación multi-hilos, con la base de datos, el analizador de objetos y una aplicación cliente, con los que interactúan a través de un protocolo de red propietario. El desarrollo de bases de datos orientados a objetos comienza con la definición descendente de lenguajes que reciben que se adicionan a una capa de persistencia en el lenguaje de programación. Por lo tanto las bases de datos orientadas a objetos ofrecen similar integración que en los ambientes de las aplicaciones orientadas a objetos. Esto es diferente del modelo utilizado por las base de datos relacionales de nuestros días, donde la interacción con la base de datos ocurre a través de un lenguaje intermediario (SQL). Análogamente a ANSI SQL, la interfaz estándar de consultas para bases de datos relacionales, constituye un estándar para los productos de bases de datos orientados a objetos. El Grupo Gestor de Datos de Objetos (ODMG) específicamente define un API, un lenguaje de consultas, un lenguaje de metadatos, y lenguajes host para C++, SmallTalk y Java. La mayoría de los

sistemas de bases de datos orientados a objetos proporcionan cierto nivel de soporte para los estándares de ODMG, pero sin embargo, esta es una implementación incompleta. El ODMG no es muy utilizado actualmente. Más reciente es la especificación del Java Data Objects (JDO), el cual trajo consigo nuevas posibilidades desde abril del 2002. El JDO fue asesorado por los miembros de la comunidad de bases de datos orientadas a objetos y esta siendo adoptado por los productos de bases de datos orientados a objetos como es el caso del API primario, a menudo adicionado al soporte extensible de ODMG. No emplearemos tiempo en pensar por qué la tecnología orientada a objetos de las bases de datos no ha sido más popular. Observen simplemente que las bases de datos objetos no se han adoptado de forma extensa y no aparecen como probablemente estarán en un futuro próximo. Podemos ver simplemente que la mayoría de los desarrolladores tienen más posibilidades de utilizar la tecnología relacional que la orientada a objetos, dadas las realidades políticas existentes. [12]

1.9.2 Uso de mapeadores objeto/relacional:

El Mapeador de Objeto Relacionales (ORM). Algunos lo llaman mapeador de objetos relacionales, otros prefieren simplemente mapeador de objetos. Utilizamos exclusivamente el termino objeto/relacional y sus siglas, ORM. El slash vertical representa el problema mismatch que ocurre cuando chocan los dos mundos. ORM no es más que la persistencia automática (y transparente) de los objetos, en una aplicación orientada a objetos, en tablas de una base de datos relacional, haciendo uso de metadatos que describen el mapeo entre los objetos y la base de datos. Cuando nos queremos referir al nivel de complejidad que tiene la puesta en práctica de ORM, podemos hacer la observación siguiente: la implementación de un ORM es más complejo que el framework de una aplicación web, pero sin embargo más sencillo que una aplicación de un servidor. El uso de mapeadores objeto/relacional constituye una solución beneficiosa para enfrentar los problemas de incompatibilidad entre el modelo relacional y el modelo orientado a objetos, puesto que nos proporciona ventajas en cuanto a mantenibilidad, productividad, independencia del vendedor y funcionamiento. [12]

Capítulo 2: Descripción de la solución.

En este capítulo se realiza una explicación y descripción de la solución propuesta para nuestro trabajo. Primeramente se enuncian de forma breve los requisitos funcionales y no funcionales relacionados con las bases de datos. Luego se realiza el diseño lógico y físico, así como la descripción de las tablas que lo conforman y las relaciones que existen entre ellas, de los módulos Gestión de Usuarios y Sistema Estadístico de la base de datos del Sistema Integrado de Gestión Estadística. Además se explican los procedimientos almacenados más importantes para estos módulos. También se explica el diseño de la capa de acceso a datos para el Módulo de Entrada de Datos. Finalmente se realiza la justificación de las herramientas utilizadas en el desarrollo de la solución propuesta.

2.1 Requerimientos funcionales de la base de datos.

- Adicionar usuarios: El administrador es el autorizado de adicionar un usuario. Cuando el administrador adiciona un usuario nuevo el sistema debe permitir que especifique el nombre de usuario, apellidos, el nick de usuario y la contraseña.
- Eliminar usuarios: El administrador es el autorizado de eliminar un usuario. Cuando el administrador elimina un usuario el sistema debe permitir que seleccione el usuario de la lista y comprobar si lo desea eliminar.
- Modificar usuarios: El administrador es el encargado de modificar el nivel de acceso y los permisos de cada usuario.
- Asignar roles: El administrador asigna el rol al usuario en correspondencia al nivel de acceso que debe tener el usuario dentro del sistema. La aplicación debe dar los permisos al sistema en función del rol que desempeñe.
- Adicionar modelos: El administrador es el encargado de adicionar los modelos aprobados para la captación de información estadística.
- Eliminar modelos: Cuando el digitador desea eliminar un modelo seleccionado el sistema debe comprobar que se desea realizar esa operación y advertir las consecuencias de la eliminación de un modelo.
- Modificar modelos: Al digitador solicitar actualización de un modelo, el sistema debe cargar de la base de datos el modelo.
- Autenticarse: La autenticación tiene alta importancia para el sistema, brinda una mayor seguridad y permite que cada usuario acceda solo a la parte del sistema que lo permita su rol y da los derechos para realizar las acciones que requiera su rol. Un mismo usuario puede interpretar varios roles. Solo el administrador otorga los permisos necesarios a cada usuario.
- Digitar datos estadísticos: El sistema debe de permitir cargar los modelos (entiéndase el formulario) desde un archivo y llenarlos según los datos, tanto desde una vista amigable diseñada para los CI, como desde una interfaz de formularios dedicada a las oficinas

estadísticas y adecuada a las características del digitador (similares a la de MicroSet). Este proceso será realizado por el digitador.

- Validar datos estadísticos: El sistema debe permitir la validación de los datos una vez realizado cada modelo, debe verificar que los datos entrados sean correctos, para ello es necesario que se haya ofrecido la ayuda correspondiente en el llenado de los modelos. Esta acción será realizada por el validador.
- Auditar el sistema: El sistema debe permitir tener un control detallado del manejo de información sobre las modificaciones realizadas. Debe brindar un reporte con la fecha, hora y modificación realizada, así como por la persona que fue realizada dicha transformación.
- Introducir datos del modelo en la base de datos una vez que estén validados: Cuando se culmina con las validaciones correspondientes el sistema debe permitir que se guarden los datos del modelo (la información estadística digitada) en la base de datos.
- Crear clasificador de un Centro Informante: El Especialista de Clasificadores es el encargado de crear un nuevo clasificador al cual le serán asociadas diferentes clasificaciones para clasificar a un Centro Informante.
- Eliminar clasificador de un Centro Informante: El Especialista de Clasificadores es el encargado de eliminar un clasificador que ya como acuerdo de los directivos no se va a utilizar más.
- Modificar clasificador de un Centro Informante: El Especialista de Clasificadores es el encargado de modificar un clasificador en caso de que existir algún cambio en la descripción del mismo.
- Crear clasificación asociada a un clasificador: El Especialista de Clasificadores y/o el Responsable de Clasificadores son los encargados de crear una clasificación que será asociada a un clasificador.
- Eliminar clasificación asociada a un clasificador: El Especialista de Clasificadores y/o el Responsable de Clasificadores son los encargados de eliminar una clasificación por la que ya no se regirá el clasificador correspondiente.

- Modificar clasificación asociada a un clasificador: El Especialista de Clasificadores y/o el Responsable de Clasificadores son los encargados de modificar una clasificación por los siguientes motivos: inserción de un error cuando se crea la clasificación o cuando se desea cambiar en cuanto a nombre o descripción.
- Listar clasificadores: La aplicación debe dar la posibilidad de listar los clasificadores existentes con sus clasificaciones asociadas.
- Dar alta a un Centro Informante: El Registrador es el encargado de clasificar a un Centro Informante utilizando los diferentes clasificadores y sus clasificaciones asociadas, cuando se le da de alta.
- Dar baja a un Centro Informante: El Registrador es el encargado de registrar la fecha en que se dio de baja al Centro Informante.
- Modificar clasificaciones de un Centro Informante: El Registrador es el encargado de modificar las clasificaciones asociadas al Centro Informante debido a algún cambio en el mismo.
- Crear Registro: El Registrador es el responsable de crear un registro nuevo al cual pertenecerán diferentes Centros Informantes.
- Eliminar Registro: El Registrador es el responsable de eliminar un registro así como también su contenido.
- Modificar Registro: El Registrador es el responsable de modificar las características o atributos del registro.

2.2 Requerimientos no funcionales de la base de datos.

- En la ONE se necesitan dos servidores con las siguientes características: deben tener un procesador Pentium 4 a una velocidad de 3.2 GHZ, con una memoria RAM de 6 Gb, 4 MB de caché y capacidad en el disco duro 1TB.

- En las OTE los servidores de datos deben tener un procesador P4 a 3.2 GHZ de velocidad en el procesador, con una memoria RAM de 2 GB y capacidad de almacenamiento en dos discos de 250 Gb. Estas características son las mínimas necesarias.
- En las OME los servidores de datos deben tener un procesador P4 a 3.2 GHZ de velocidad en el procesador, con una memoria RAM de 2 GB y capacidad de almacenamiento en dos discos de 250 Gb. Estas características son las mínimas necesarias.
- La conectividad desde las OME hacia las OTE, y de las OTE hacia la ONE debe tener una velocidad de 1 Mb/seg como mínimo.
- La conectividad desde la ONE hacia las OTE, y de las OTE hacia las OME debe tener una velocidad de 128 Kb/seg como mínimo.
- Debido a que los futuros usuarios del sistema no son expertos en la rama informática, la interfaz del sistema debe ser manipulable.
- La información manejada por el sistema debe ser protegida de accesos no autorizados y divulgación.
- La base de datos debe ser independiente de la aplicación.
- La base de datos debe ser protegida contra la corrupción y estados inconsistentes.
- Los usuarios para trabajar con el sistema deben previamente autenticarse.
- Las contraseñas del sistema deben estar encriptadas.
- Se debe planificar una estrategia para brindarle mantenimiento a la base de datos durante un periodo, una vez que el cliente este usando el producto.

2.3 Diseño de la base de datos.

Para el Sistema Integrado de Gestión Estadística se desarrolló el diseño de dos módulos de la base de datos: Gestión de Usuarios y Sistema Estadístico. El módulo Gestión de Usuarios de la base de datos es el que se va a encargar de la seguridad del sistema. El módulo Sistema Estadístico de la base de

datos es la que se va a encargar de gestionar toda la información estadística relacionada con los modelos. A continuación podemos encontrar la descripción del diseño de la base de datos para ambos módulos de la base de datos del Sistema Integrado de Gestión Estadística.

2.3.1 Módulo Gestión de Usuarios de la base de datos del SIGE.

Este módulo tiene como objetivos garantizar el almacenamiento de la información relacionada con los permisos de cada usuario en el sistema, y registrar todas y cada una de las acciones que realiza cada usuarios en el sistema.

El modelo lógico que se propone para el módulo Gestión de Usuarios es el siguiente:

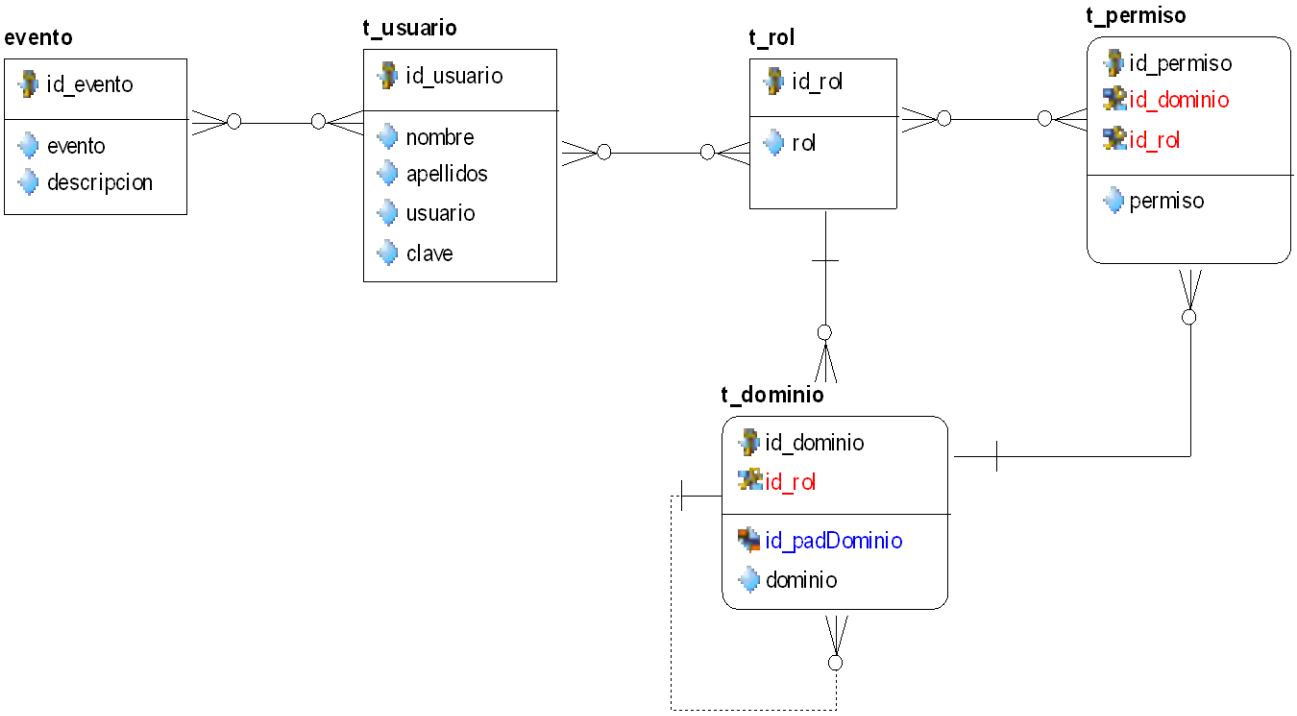

Figura 4 Modelo lógico del módulo Gestión de Usuarios de la base de datos del SIGE.

Descripción de las interrelaciones.

Las interrelaciones de modelo antes visto cumplen con las siguientes afirmaciones: un dominio puede contener varios subdominios. Un dominio es básicamente un conjunto de roles y permisos, donde cada rol interpreta un conjunto de permisos del dominio, y un permiso puede ser interpretado por varios roles. También ocurre que un usuario puede interpretar uno o más roles, y un rol puede ser interpretado por varios usuarios. Podemos plantear que en la aplicación ocurren eventos, pero cuando dichos eventos son provocados por un usuario pues se convierten en acciones, por lo que podemos decir que un usuario puede provocar varios eventos, es decir, puede realizar varias acciones, así como, un evento puede ser provocado por varios usuarios, es decir, una acción puede ser realizada por varios usuarios.

El modelo físico que se obtiene para el módulo Gestión de Usuarios es el siguiente:

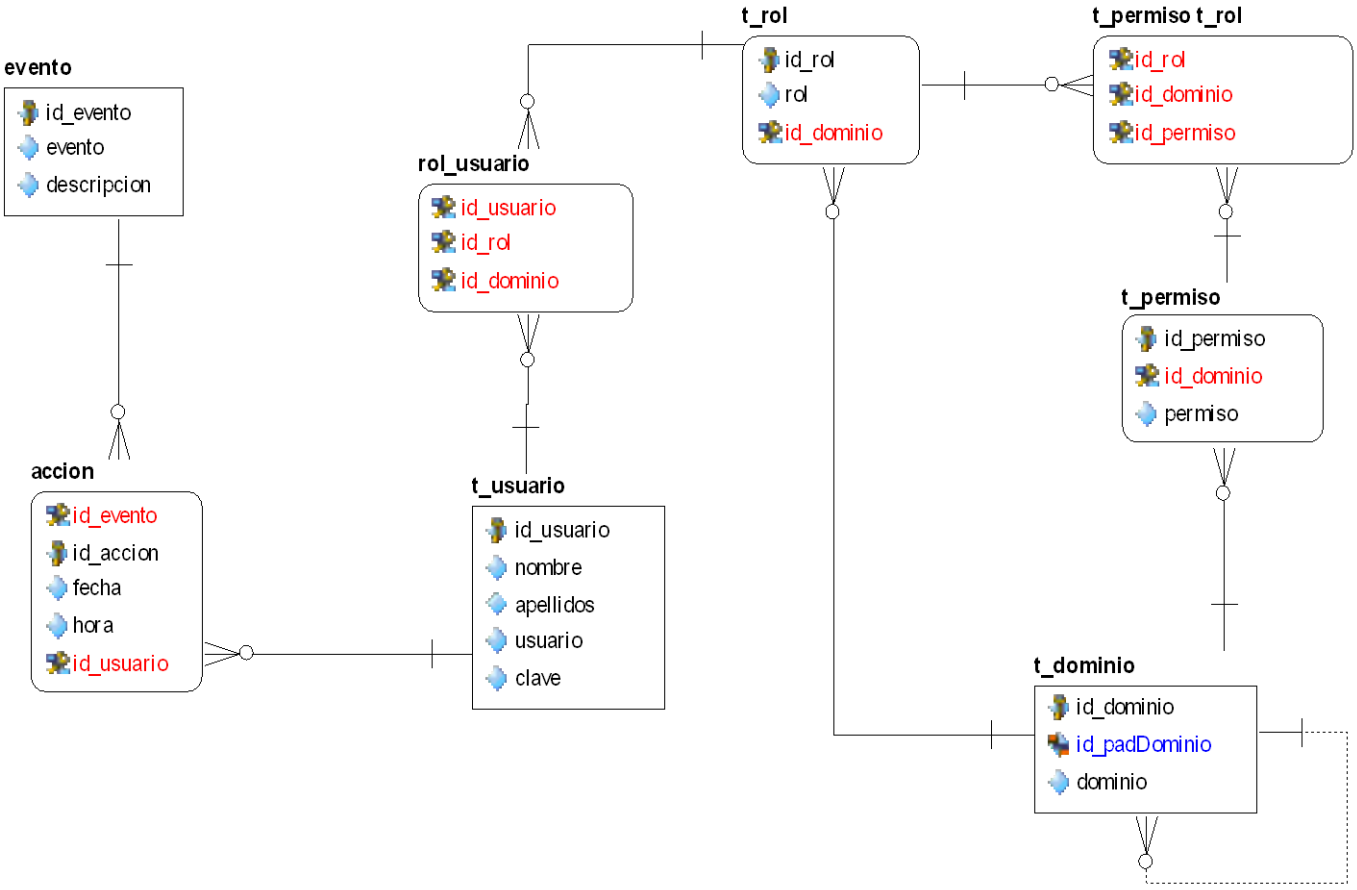

Figura 5 Modelo físico del módulo Gestión de Usuarios de la base de datos del SIGE.

Descripción de las tablas.

Tabla 1 Descripción de la tabla evento.

Nombre: evento					
Descripción: En esta tabla se almacenan los datos de los eventos que pueden ocurrir en la aplicación. Contiene los datos principales.					
Atributo	**Tipo**	**Descripción**	**Llave**		
			SI	**NO**	**Origen**
id_evento	int	**Código del evento:** Este código es un número que identifica a cada uno de los eventos que pueden ocurrir en la aplicación.	PK		evento
evento	char (255)	**Evento:** Este campo va a guardar el nombre de los posibles eventos que pueden ocurrir la aplicación.		X	
descripcion	text (16)	**Descripción:** Este campo va a guardar una breve explicación de las características del evento.		X	

Tabla 2 Descripción de la tabla t_usuario.

Nombre: t_usuario					
Descripción: En esta tabla se almacenan los datos de los usuarios de la aplicación. Contiene los datos principales.					
Atributo	**Tipo**	**Descripción**	**Llave**		
			SI	**NO**	**Origen**

id_usuario	int	**Código del usuario:** Este código es un número que identifica a cada uno de los usuarios que pueden acceder a la aplicación.	PK		t_usuario
nombre	char (30)	**Evento:** Este campo va a guardar el nombre real de la persona que es usuarios de la aplicación.		X	
apellidos	char (30)	**Descripción:** Este campo va a guardar los apellidos de los usuarios de la aplicación.		X	
usuario	char (30)	**Usuario:** Este campo almacena el nick del usuario en la aplicación.		X	
clave	char (30)	**Clave:** Este campo guarda el password de cada usuario en la aplicación.		X	

Tabla 3 Descripción de la tabla acción.

Nombre: accion					
Descripción: En esta tabla se almacenan los datos de las acciones que son realizadas por los usuarios en la aplicación. Contiene los datos principales.					
Atributo	**Tipo**	**Descripción**	**Llave**		
			SI	**NO**	**Origen**
id_evento	int	**Código del evento:** Este código es un número que identifica a cada uno de los eventos que pueden ocurrir en la aplicación.	PK		evento
id_usuario	int	**Código del usuario:** Este código es un número que identifica a cada uno de los usuarios que pueden acceder a la aplicación.	PK		t_usuario

id_accion	int	**Código de la acción:** Este código es un número que identifica a cada una de las acciones en la aplicación.	PK		accion
fecha	datetime	**Fecha:** Este campo va a guardar la fecha en que un usuario realizó una acción.		X	
hora	datetime	**Hora:** Este campo va a guardar la hora en que un usuario realizó una acción.		X	
Observaciones: Esta tabla es el resultado de la relación de muchos a mucho entre las tablas evento y t_usuario					

Tabla 4 Descripción de la tabla t_rol.

Nombre: t_rol					
Descripción: En esta tabla se almacenan los datos de los roles que pueden interpretar los usuarios en la aplicación. Contiene los datos principales.					
Atributo	**Tipo**	**Descripción**	**Llave**		
			SI	**NO**	**Origen**
id_rol	int	**Código del rol:** Este código es un número que identifica a cada rol que puede interpretar un usuario.	PK		t_rol
id_dominio	int	**Código del dominio:** Este código es un número que identifica a cada dominio que existe en la aplicación.	PK		t_dominio
rol	char (64)	**Descripción:** Este campo va a guardar los posibles roles que pueden existir en la aplicación.		X	

Tabla 5 Descripción de la tabla rol_usuario.

Nombre: rol_usuario					
Descripción: En esta tabla se almacenan los datos de los roles que interpretan los usuarios en la aplicación. Contiene los datos principales.					
Atributo	**Tipo**	**Descripción**	**Llave**		
			SI	**NO**	**Origen**
id_rol	int	**Código del rol:** Este código es un número que identifica a cada rol que puede interpretar un usuario.	PK		t_rol
id_dominio	int	**Código del dominio:** Este código es un número que identifica a cada dominio que existe en la aplicación.	PK		t_dominio
id_usuario	int	**Código del usuario:** Este código es un número que identifica a cada uno de los usuarios que pueden acceder a la aplicación.	PK		t_usuario
Observaciones: Esta tabla es el resultado de la relación mucho a mucho que existe entre las tablas t_usuario y t_rol.					

Tabla 6 Descripción de la tabla t_permiso.

Nombre: t_permiso					
Descripción: En esta tabla se almacenan los datos de los permisos que pueden ser interpretados por un rol en la aplicación. Contiene los datos principales.					
Atributo	**Tipo**	**Descripción**	**Llave**		
			SI	**NO**	**Origen**
id_permiso	int	**Código del permiso:** Este código es un número que identifica a cada permiso que puede interpretar un rol.	PK		t_permiso

id_dominio	int	**Código del dominio:** Este código es un número que identifica a cada dominio que existe en la aplicación.	PK		t_dominio
permiso	char (255)	**Descripción:** Este campo va a guardar los posibles permisos que puede interpretar un rol.		X	

Tabla 7 Descripción de la tabla t_permiso t_rol.

Nombre: t_permiso t_rol					
Descripción: En esta tabla se almacenan los datos de los eventos que pueden ocurrir en la aplicación. Contiene los datos principales.					
Atributo	**Tipo**	**Descripción**	**Llave**		
			SI	**NO**	**Origen**
id_permiso	int	**Código del permiso:** Este código es un número que identifica a cada permiso que puede interpretar un rol.	PK		t_permiso
id_dominio	int	**Código del dominio:** Este código es un número que identifica a cada dominio que existe en la aplicación.	PK		t_dominio
id_rol	int	**Código del rol:** Este código es un número que identifica a cada rol que puede interpretar un usuario.	PK		t_rol
Observaciones: Esta tabla surge como resultado de la relación de muchos a muchos entre las tablas t_permiso y t_rol.					

Tabla 8 Descripción de la tabla t_dominio.

Nombre: t_dominio
Descripción: En esta tabla se almacenan los datos de los eventos que pueden ocurrir

en la aplicación. Contiene los datos principales.					
Atributo	**Tipo**	**Descripción**	**Llave**		
			SI	**NO**	**Origen**
id_dominio	int	**Código del dominio:** Este código es un número que identifica a cada dominio que existe en la aplicación.	PK		t_dominio
id_padDominio	int	**Código del dominio padre:** Este código es un número que identifica a cada dominio que contenga subdominios en la aplicación.		X	
dominio	char (64)	**Dominio:** Este campo almacena los nombres de los posibles dominios que existen en la aplicación.		X	
Observaciones: El código del domino padre puede ser nulo en caso de que el dominio al que estemos haciendo referencia no tenga padre, es decir, que no sea subdominio de ningún otro dominio.					

Descripción de los principales procedimientos almacenados.

Tabla 9 Descripción del procedimiento almacenado sprol_usuarioDelete.

Nombre del procedimiento almacenado:	sprol_usuarioDelete	
Descripción:	Este procedimiento almacenado elimina un usuario con el rol que interpreta el mismo en el sistema.	
Parámetros	**Tipo**	**Descripción**
id_rol	int	Este es el código que identifica a un rol.
id_dominio	int	Este es el código que identifica a un dominio.
id_usuario	int	Este es el código que identifica a un usuario.
Resultado:	Queda eliminado el usuario con el rol que interpreta.	

Tabla 10 Descripción del procedimiento almacenado spt_dominioInsert.

Nombre del procedimiento almacenado:	spt_dominioInsert	
Descripción:	Este procedimiento almacenado inserta un nuevo dominio en la tabla t_dominio.	
Parámetros	**Tipo**	**Descripción**
id_padDominio	int	Este es el código del dominio padre.
dominio	char (64)	Este representa el nombre del dominio que se insertará.
Resultado:	Queda insertado un nuevo dominio en la aplicación.	

Tabla 11 Descripción del procedimiento almacenado spt_rolUpdate.

Nombre del procedimiento almacenado:	spt_rolUpdate	
Descripción:	Este procedimiento almacenado actualiza un rol en la base de datos.	
Parámetros	**Tipo**	**Descripción**
rol	char (64)	Este parámetro representa el nombre de un rol.
id_rol	int	Este representa el código de un rol.
id_dominio	int	Este representa el código de un dominio
Resultado:	Queda actualizado un rol en la base de datos.	

Tabla 12 Descripción del procedimiento almacenado t_usuarioSelProc.

Nombre del procedimiento almacenado:	t_usuarioSelProc
Descripción:	Este procedimiento selecciona las características de un usuario dado el código del mismo.

Parámetros	Tipo	Descripción
id_usuario	int	Este representa el código de un usuario
Resultado:	Se obtiene el nombre, los apellidos, usuario y la clave de un usuario determinado.	

2.3.2 Módulo Sistema Estadístico de la base de datos del SIGE.

Este módulo tiene como objetivos garantizar que se almacene toda la información relacionada con todas y cada una de las Oficinas Territoriales de Estadísticas, Oficinas Municipales de Estadísticas y Centros Informantes del país, así como gestionar los modelos que se utilizarán en la captación de la información a los diferentes niveles.

El modelo lógico que se propone para el módulo Sistema Estadístico es el siguiente:

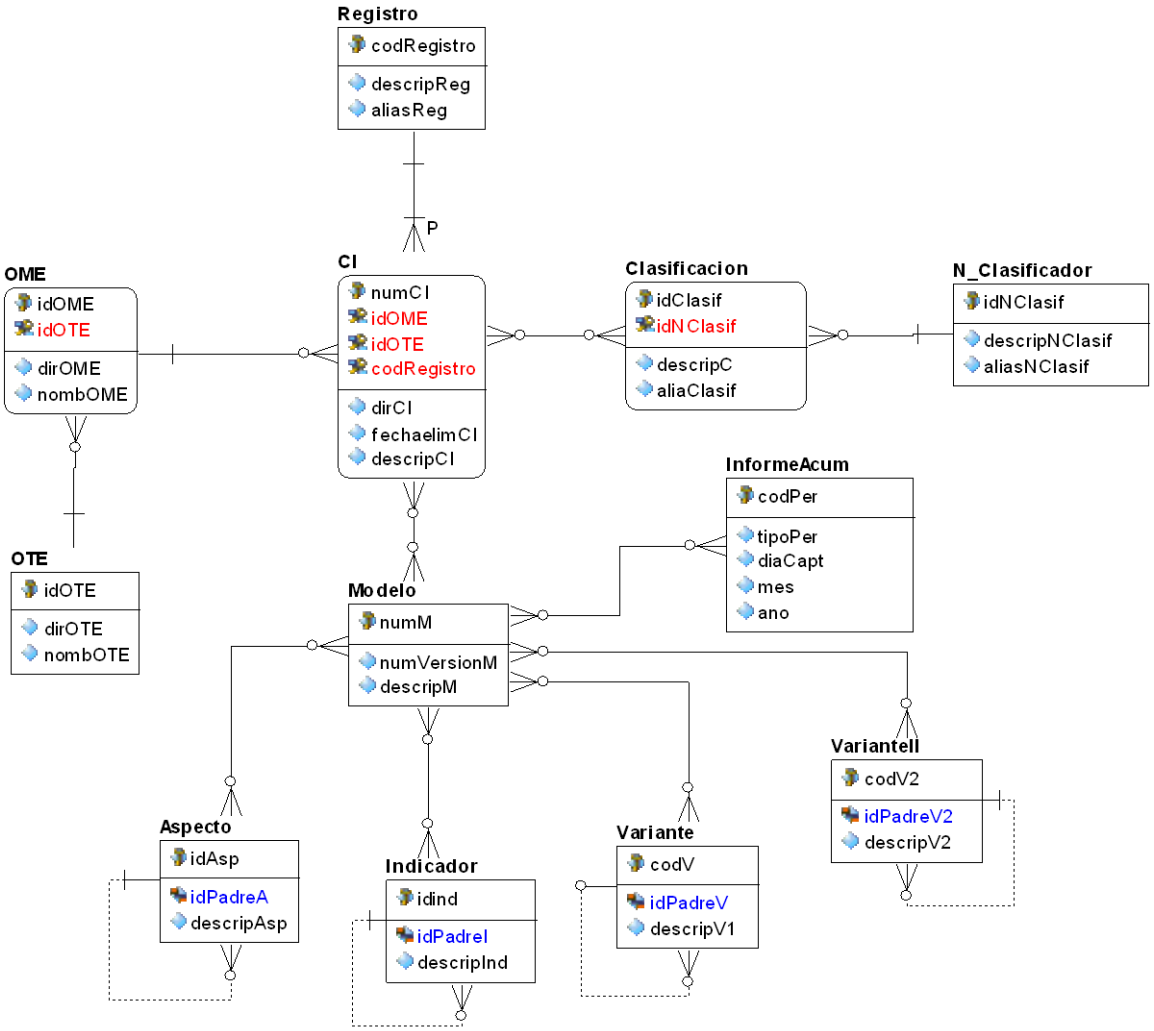

Figura 6 Modelo lógico del módulo Sistema Estadístico de la base de datos del SIGE.

Descripción de las interrelaciones.

Las interrelaciones de modelo antes visto cumplen con las siguientes afirmaciones: a una OTE pertenecen varias OME, pero una OME pertenece a una única OTE. Un CI pertenece a una única OME y análogamente a una única OTE, de manera que a una OTE y a una OME se le hacen corresponder muchos CI. Un CI está asociado a un solo Registro, mientras que en un Registro pueden agruparse varios

CI. A un CI se le asocian muchas clasificaciones, las cuales a su vez se relacionan con muchos CI. De ahí que una clasificación se encuentra relacionada con un único clasificador, el cual a su vez tiene asociado muchas clasificaciones.

Un CI capta información a través de varios modelos, así como un modelo puede ser utilizado en varios CI para captar información. Un modelo puede contener en su estructura varios indicadores, aspectos e informes acumulados, así como una variante y una variante II. De la misma forma en que un indicador, un informe acumulado, un aspecto, una variante y una variante II pueden formar parte de la estructura de varios modelos. Un indicador puede tener sub-indicadores. Un aspecto puede tener sub-aspectos. Una variante puede tener sub-variantes, así como también las tiene la variante II.

El modelo físico que se obtiene para el módulo Sistema Estadístico es el siguiente:

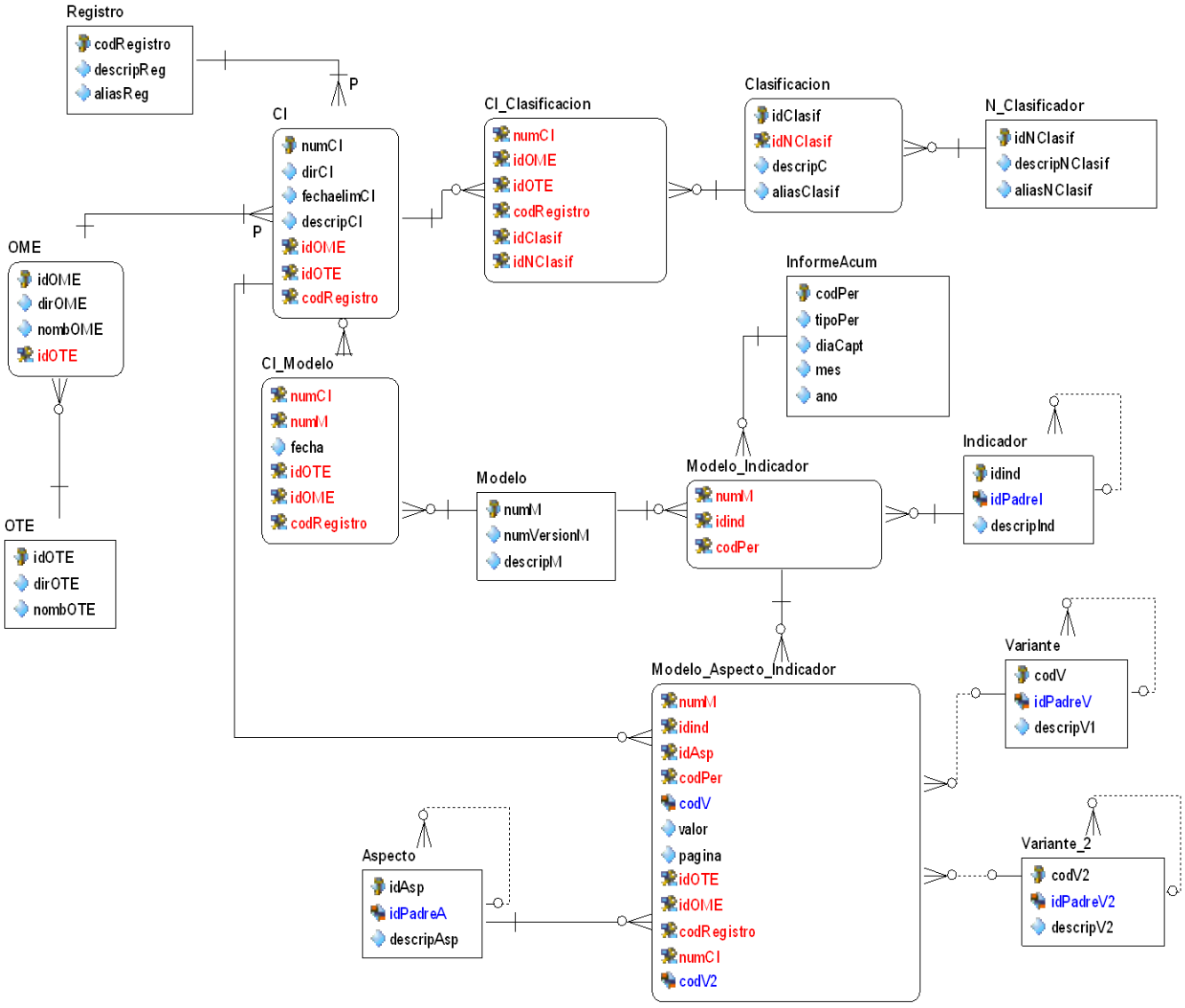

Figura 7 Modelo físico del módulo Sistema Estadístico de la base de datos del SIGE.

Descripción de las tablas.

Tabla 13 Descripción de la tabla OTE.

Nombre: OTE					
Descripción: En esta tabla se almacenan los datos de las 15 oficinas territoriales de estadística del país. Contiene los datos principales.					
Atributo	**Tipo**	**Descripción**	**Llave**		
			SI	**NO**	**Origen**
idOTE	int	**Código de la OTE:** este código es un número que identifica a cada una de las 15 oficinas territoriales de estadística dentro del país	PK		OTE
dirOTE	nvarchar (30)	**Dirección de la OTE:** Este campo va a guardar la dirección física real de cada una de las 15 oficinas territoriales de estadísticas ubicadas en cada una de las provincias de nuestro país.		X	
nomOTE	nvarchar (30)	**Nombre de la OTE:** Este campo va a guardar el nombre de cada una de las 15 oficinas territoriales de estadísticas ubicadas en cada una de las provincias de nuestro país.		X	
Observaciones: De este tipo de oficina existe una por cada provincia incluyendo el municipio especial Isla de la Juventud por lo que en el país hay un total de 15 OTE.					

Tabla 14 Descripción de la tabla OME.

Nombre: OME
Descripción: En esta tabla se almacenan los datos de las oficinas municipales de estadística del país. Contiene los datos principales.

Atributo	Tipo	Descripción	Llave		
			SI	NO	Origen
idOME	bigint	**Código de la OME:** este código es un número que identifica a cada una de las oficinas municipales de estadística dentro de un territorio.	PK		OME
dirOME	nvarchar (30)	**Dirección de la OME:** Este campo va a guardar la dirección física real de cada una de las oficinas municipales de estadísticas ubicadas en cada territorio.		X	
nomOME	nvarchar (30)	**Nombre de la OME:** Este campo va a guardar el nombre de cada una de las oficinas municipales de estadísticas ubicadas en cada territorio.		X	
idOTE	int	**Código de la OTE:** este código es un número que identifica a cada una de las oficinas municipales de estadística dentro un territorio.	PKF		OTE
Observaciones: De este tipo de oficina existe una por cada municipio en un territorio o provincia por lo que en el país hay un total de 169 OME.					

Tabla 15 Descripción de la tabla CI.

Nombre: CI					
Descripción: En esta tabla se almacenan los datos de los centros informantes del país. Contiene los datos principales.					
Atributo	**Tipo**	**Descripción**	**Llave**		
			SI	**NO**	**Origen**

numCI	bigint	**Código del Centro Informante:** Este campo va a guardar un número que identifica a cada centro informante dentro de un municipio.	PK		CI
codRegistro	char (18)	**Código del Registro:** Este campo va a guardar un número que identifica a un Registro.	PKF		Registro
descripCI	nvarchar (30)	**Descripción de los CI:** Este campo va a guardar las características que definen al centro informante dentro de un territorio.		X	
dirCI	nvarchar (50)	**Dirección de CI:** Este campo va a guardar la dirección física de cada uno de los Centros Informantes.		X	
fechaelimCI	datetime	**Fecha de eliminación de CI:** Este campo va a guardar la fecha en que se elimina un Centro Informante.		X	
idOME	bigint	**Código de la OME:** este código es un número que identifica a cada una de las oficinas municipales de estadística dentro de un territorio.	PKF		OME
idOTE	int	**Código de la OTE:** este código es un número que identifica a cada una de las 15 oficinas territoriales de estadística dentro del país.	PKF		OTE
Observaciones: Los CI son los responsables de captar la información, significa que son los encargados llenar los modelos estadísticos. Existen varios CI dentro de un municipio, es decir, existen varios CI relacionados con una OME.					

Tabla 16 Descripción de la tabla Registro.

Nombre: Registro
Descripción: Esta tabla surge por el hecho de que un conjunto de Centros Informantes

van a pertenecer a uno y solo un determinado Registro.

Atributo	Tipo	Descripción	Llave		
			SI	NO	Origen
codRegistro	char (18)	**Código del Registro:** Este campo va a guardar un número que identifica a un Registro.	PK		Registro
descripReg	nvarchar (30)	**Descripción del Registro:** Este campo va a guardar la descripción de las características del Registro.		X	
aliasReg	char (10)	**Alias del Registro:** Este campo va a guardar las siglas que identifican el Registro.		X	

Observaciones: En esta tabla se almacena la información necesaria de un Registro, al cual hay asociados varios CI.

Tabla 17 Descripción de la tabla Clasificación.

Nombre: Clasificación

Descripción: En esta tabla se almacenan las distintas clasificaciones que tienen los clasificadores. En esta tabla aparecerán por ejemplo, el nombre de todas las provincias del país, los cuales serán los posibles valores que tendrá el clasificador DPA (División Política Administrativa), y que tendrán como código que las identifica también el código de la DPA, y así sucesivamente aparecerán todas las clasificaciones que existen con sus respectivos identificadores o códigos de sus clasificadores.

Atributo	Tipo	Descripción	Llave		
			SI	NO	Origen
idClasif	char (10)	**Código de la Clasificación:** Este campo va a guardar el número que identifica a la clasificación.	PK		Clasificación

idNClasif	char (10)	**Código del Clasificador:** Este campo va a guardar el número que identifica al clasificador que se va a usar para clasificar.	PKF		N_Clasificador
descripC	nvarchar (20)	**Descripción de la Clasificación:** Este campo va a guardar la descripción de las principales características de la clasificación.		X	
aliasClasif	char (10)	**Alias de la Clasificación:** Este campo va a guardar las siglas que identifican al clasificador.		X	
Observaciones: En esta tabla se almacenan las diferentes clasificaciones que están asociadas a los diferentes clasificadores.					

Tabla 18 Descripción de la tabla CI_Clasificacion.

Nombre: CI_Clasificacion					
Descripción: Esta tabla surge de la relación mucho a mucho que existe entre las tablas CI y Clasificación. Un centro informante puede tener varias clasificaciones asociadas a el, así como una clasificación puede pertenecer a varios centros informantes. Contiene los datos principales.					
Atributo	**Tipo**	**Descripción**	**Llave**		
			SI	**NO**	**Origen**
numCI	bigint	**Código del Centro Informante:** Este campo va a guardar un número que identifica a cada centro informante dentro de un municipio.	PK		CI
codRegistro	char (18)	**Código del Registro:** Este campo va a guardar un número que identifica a un Registro.	PK		Registro

idOME	bigint	**Código de la OME:** este código es un número que identifica a cada una de las 15 oficinas municipales de estadística dentro de un territorio.	PK		OME
idOTE	int	**Código de la OTE:** este código es un número que identifica a cada una de las 15 oficinas territoriales de estadística dentro del país.	PK		OTE
idClasif	char (10)	**Código de la Clasificación:** Este campo va a guardar el número que identifica a la clasificación.	PK		Clasificacion
idNClasif	char (10)	**Código del Clasificador:** Este campo va a guardar el número que identifica al clasificador que se va a usar para clasificar.	PK		N_Clasificador
Observaciones: Esta tabla está constituida por las llaves primarias de las tablas CI, OTE, OME, Registro, Clasificación y N_Clasificador.					

Tabla 19 Descripción de la tabla N_Clasificador.

Nombre: N_Clasificador					
Descripción: Tabla nomencladora de clasificadores. Esta tabla almacena el universo de clasificadores, entre los que se encuentran: ORG, CAE, NAE, DPA, etc., con sus datos principales.					
Atributo	**Tipo**	**Descripción**	**Llave**		
			SI	**NO**	**Origen**
idNClasif	char (10)	**Código del Clasificador:** Este campo va a guardar el número que identifica al clasificador que se va a usar para clasificar.	PK		N_Clasificador
aliasNClasif	char (10)	**Alias del Clasificador:** Este campo va a guardar las siglas que		X	

		identifican al clasificador.			
descripNClasif	nvarchar (5)	**Descripción del Clasificador:** Este campo va a guardar la descripción de las características de un clasificador determinado.		X	
Observaciones: En esta tabla se almacenan todos los clasificadores con su respectivo código.					

Tabla 20 Descripción de la tabla Modelo.

Nombre: Modelo					
Descripción: Tabla en la que se guardarán los tipos de modelos en los que se recogerá la información de los CI.					
Atributo	**Tipo**	**Descripción**	**Llave**		
			SI	**NO**	**Origen**
numM	bigint	**Código del Modelo:** Este campo va a guardar el número que identifica a cada modelo que se utilizará en la captación de información.	PK		Modelo
numVersionM	bigint	**Número de la Versión del Modelo:** Este campo va a guardar el número que identifica la versión de cada modelo.		X	Modelo
descripM	nvarchar (30)	**Descripción del Modelo:** Este campo va a guardar la descripción de las características de un modelo determinado.		X	
Observaciones: Los modelos son el soporte donde los CI van a registrar toda la información estadística que le soliciten.					

Tabla 21 Descripción de la tabla CI_Modelo.

Nombre: CI_Modelo					
Descripción: Esta tabla surge de la relación mucho a mucho que existe entre CI y Modelo. Un centro informante puede tener varios modelos asociados a el, así como un modelo puede pertenecer a varios centros informantes. Contiene los datos principales.					
Atributo	**Tipo**	**Descripción**	**Llave**		
			SI	**NO**	**Origen**
numCI	bigint	**Código del Centro Informante:** Este campo va a guardar un número que identifica a cada centro informante dentro de un municipio.	PK		CI
numM	bigint	**Código del Modelo:** Este campo va a guardar el número que identifica a cada modelo que se utilizará en la captación de información.	PK		Modelo
idOME	bigint	**Código de la OME:** este código es un número que identifica a cada una de las 15 oficinas municipales de estadística dentro de un territorio.	PK		OME
idOTE	int	**Código de la OTE:** este código es un número que identifica a cada una de las 15 oficinas territoriales de estadística dentro del país.	PK		OTE
codRegistro	char (18)	**Código del Registro:** Este campo va a guardar un número que identifica a un Registro.	PK		Registro
fecha	datetime	**Fecha:** Esta es la fecha con la que se registra el llenado con los datos estadísticos del modelo.		X	
Observaciones: Esta tabla está constituida por el atributo fecha, que es la fecha en que					

se realiza el modelo en un CI, y además por la llaves primarias de las tablas Modelo, CI, OME, OTE y Registro.

Tabla 22 Descripción de la tabla InformeAcum.

Nombre: InformeAcum					
Descripción: Esta tabla refleja la información respecto al informe acumulado hasta: como así aparecen en los modelos para llamarle de alguna manera a esta tabla. La misma tendrá la información referente a la fecha de captación, en dependencia por supuesto del tipo de periodicidad, entre otros atributos que abajo se describen. En sentido general lo que se quiere reflejar es un universo finito de todos los tipos posibles de informes acumulados que existen para después asociarlos a la relación modelo indicador.					
Atributo	**Tipo**	**Descripción**	**Llave**		
			SI	**NO**	**Origen**
codPer	bigint	**Código de Periodicidad:** Este campo va a guardar el código que identifica una determinada periodicidad que se haya definido para el llenado del modelo.	PK		InformeAcum
tipoPer	nvarchar (20)	**Tipo de Periodicidad:** Este campo especifica el tipo de periodicidad que va a estar definido por el tiempo en que se realiza el llenado del modelo. Puede ser: mensual, trimestral, anual, etc.		X	
diaCapt	int	**Día de Captación:** Este campo va a guardar el día en que se registran los datos del informe acumulado.		X	
mes	int	**Mes:** Este campo va a guardar el mes en que se registran los datos del informe acumulado.		X	

ano	int	**Año:** Este campo va a guardar el año en que se registran los datos del informe acumulado.		X	
Observaciones: En esta tabla se registran los datos de un informe acumulado que lleva un determinado modelo.					

Tabla 23 Descripción de la tabla Indicador.

Nombre: Indicador					
Descripción: Tabla asociada con los indicadores que debe tener un modelo. Contiene los datos principales de un indicador, con la característica de que el universo de indicadores será único por lo que su código no será repetible en más de un modelo.					
Atributo	**Tipo**	**Descripción**	**Llave**		
			SI	**NO**	**Origen**
idind	bigint	**Código del Indicador:** Este campo va a guardar el código que identifica a un indicador de otro.	PK		Indicador
idPadreI	bigint	**Código Padre Indicador:** Este campo va a guardar el código de un indicador del cual se desglosan otros.	FK		Indicador
descripInd	nvarchar (30)	**Descripción del Indicador:** Este campo va a guardar la descripción de las características de un indicador determinado.		X	
Observaciones: En esta tabla se almacenan todos los indicadores estadísticos existentes.					

Tabla 24 Descripción de la tabla Modelo_Indicador.

Nombre: Modelo_Indicador

Descripción: Esta tabla surge de la relación mucho a mucho que existe entre un modelo y un indicador que también se realiza con una determinada periodicidad. En esta tabla se relacionan el modelo con el indicador, partiendo del principio de que un modelo puede contener varios indicadores y un indicador puede aparecer independientemente en varios modelos.

Atributo	**Tipo**	**Descripción**	**Llave**		
			SI	**NO**	**Origen**
numM	bigint	**Código del Modelo:** Este campo va a guardar el número que identifica a cada modelo que se utilizará en la captación de información.	PK		Modelo
idind	bigint	**Código del Indicador:** Este campo va a guardar el código que identifica a un indicador de otro.	PK		Indicador
codPer	bigint	**Código de Periodicidad:** Este campo va a guardar el código que identifica una determinada periodicidad que se haya definido para el llenado del modelo.	PK		InformeAcum

Observaciones: En esta tabla se almacena el código de la periodicidad con el objetivo de reflejar que un modelo tiene sus propias características en cuanto a día y mes de captación, tipo de periodicidad (mensual, trimestral o anual). Todas estas características se encuentran presentes en la tabla informeAcum que tiene como llave que la identifica el cod_period.

Tabla 25 Descripción de la tabla Aspecto.

Nombre: Aspecto			
Descripción: Esta tabla almacena todos los posibles aspectos que pueden existir en un modelo.			
Atributo	**Tipo**	**Descripción**	**Llave**

			SI	NO	Origen
idAsp	bigint	**Código del Aspecto:** Este campo va a guardar el número que identifica a cada aspecto que se encontrará en cada modelo.	PK		Aspecto
idPadreA	bigint	**Código Padre de Aspecto:** Este campo va a guardar el número que identifica a un aspecto que este compuesto por sub-aspectos. Este campo puede ser nulo.	FK		Aspecto
descripAsp	nvarchar (30)	**Descripción del Aspecto:** Este campo va a guardar la descripción de las características de un aspecto determinado.		X	
Observaciones: En esta tabla se almacenan el universo de aspectos y sub-aspectos existentes en la estadística.					

Tabla 26 Descripción de la tabla Variante.

Nombre: Variante					
Descripción: En esta tabla se almacenan todas las posibles variantes que pueden existir en un modelo. Contiene los datos principales.					
Atributo	**Tipo**	**Descripción**	**Llave**		
			SI	**NO**	**Origen**
codV	bigint	**Código de la Variante:** Este campo va a guardar el número que identifica a la variante que tendrá un modelo.	PK		Variante
idPadreV	bigint	**Código Padre de Variante:** Este campo va a guardar el número que identifica a una variante en caso de que esta esté compuesta por sub-variantes. Este campo puede ser	FK		Variante

		nulo.			
descripV1	nvarchar (30)	**Descripción de la Variante 1:** Este campo va a guardar la descripción de la variante.		X	
Observaciones: En esta tabla se almacenan el universo de variantes y sub- variantes existentes en un determinado modelo.					

Tabla 27 Descripción de la tabla Variante_2.

Nombre: Variante_2					
Descripción: En esta tabla se almacenan todas las posibles variantes 2 que pueden existir en un modelo. Contiene los datos principales.					
Atributo	**Tipo**	**Descripción**	**Llave**		
			SI	**NO**	**Origen**
codV2	bigint	**Código de la Variante 2:** Este campo va a guardar el número que identifica a la variante 2 que tendrá un modelo.	PK		Variante
idPadreV2	bigint	**Código Padre de Variante 2:** Este campo va a guardar el número que identifica a una variante 2 en caso de que esta esté compuesta por sub-variantes. Este campo puede ser nulo.	FK		Variante
descripV2	nvarchar (30)	**Descripción de la Variante 2:** Este campo va a guardar la descripción de la variante 2.		X	
Observaciones: En esta tabla se almacenan el universo de variantes y sub- variantes existentes en un determinado modelo.					

Tabla 28 Descripción de la tabla Modelo_Aspecto_Indicador.

Nombre: Modelo_Aspecto_Indicador					
Descripción: Esta tabla surge de la relación mucho a mucho que existe entre un indicador y un modelo, con un aspecto y una variante determinada. Viene representando un escaque en un modelo.					
Atributo	**Tipo**	**Descripción**	**Llave**		
			SI	**NO**	**Origen**
numM	bigint	**Código del Modelo:** Este campo va a guardar el número que identifica a cada modelo que se utilizará en la captación de información.	PK		Modelo
idind	bigint	**Código del Indicador:** Este campo va a guardar el código que identifica a un indicador de otro.	PK		Indicador
codPer	bigint	**Código de Periodicidad:** Este campo va a guardar el código que identifica una determinada periodicidad que se haya definido para el llenado del modelo.	PK		InformeAcum
idAsp	bigint	**Código del Aspecto:** Este campo va a guardar el número que identifica a cada aspecto que se encontrará en cada modelo.	PK		Aspecto
codV	bigint	**Código de la Variante:** Este campo va a guardar el número que identifica a la variante que tendrá un modelo.	FK		Variante
codV2	bigint	**Código de la Variante:** Este campo va a guardar el número que identifica a la variante que tendrá un modelo.	FK		Variante2

idOTE	int	**Código de la OTE:** este código es un número que identifica a cada una de las 15 oficinas territoriales de estadística dentro del país	PK		OTE
idOME	bigint	**Código de la OME:** este código es un número que identifica a cada una de las 15 oficinas municipales de estadística dentro de un territorio.	PK		OME
numCI	bigint	**Código del Centro Informante:** Este campo va a guardar un número que identifica a cada centro informante dentro de un municipio.	PK		CI
codRegistro	char (18)	**Código del Registro:** Este campo va a guardar un número que identifica a un Registro.	PK		Registro
valor	float	**Valor:** Este campo va a guardar el valor que contiene una determinada celda.		X	
pagina	int	**Página:** Este campo va a guardar el número de la página del modelo		X	
Observaciones: Con esta tabla y su contenido podemos insertar, localizar, modificar, eliminar, etc. un determinado valor de un modelo. Constituye una de las tabla más importantes, pues contendrá cada uno de los valores contenidos en las celdas de los modelos y de esa manera cada tupla o fila de esta tabla le corresponde además del valor real de la celda, el código del modelo, del indicador asociado, del aspecto asociado, y de las posibles variantes que tendrá el modelo así como el código de la periodicidad (se explica claramente en la tabla InformeAcum).					

Descripción de los principales procedimientos almacenados.

Tabla 29 Descripción del procedimiento almacenado IndicadorDelProc.

Nombre del procedimiento almacenado:	IndicadorDelProc

Descripción:	Este procedimiento almacenado elimina un indicador de un modelo.	
Parámetros	**Tipo**	**Descripción**
idind	bigint	Este representa el código del indicador.
Resultado:	Se elimina un indicador de la tabla Indicador.	

Tabla 30 Descripción del procedimiento almacenado ModeloInsProc.

Nombre del procedimiento almacenado:	ModeloInsProc	
Descripción:	Este procedimiento almacenado inserta un nuevo modelo estadístico.	
Parámetros	**Tipo**	**Descripción**
numM	bigint	Este representa al número que identifica a un modelo.
numVersionM	bigint	Este representa al número de la versión del modelo.
descripM	nvarchar (30)	Este representa la descripción de un modelo.
Resultado:	Se inserta un modelo.	

Tabla 31 Descripción del procedimiento almacenado AspectoUpdProc.

Nombre del procedimiento almacenado:	AspectoUpdProc	
Descripción:	Este procedimiento almacenado actualiza un aspecto.	
Parámetros	**Tipo**	**Descripción**
idAsp	bigint	Este representa el código que identifica a un aspecto.
idPadreA	bigint	Este representa el código padre de un aspecto.
descripAsp	nvarchar (30)	Este representa la descripción de un aspecto.

Resultado:	Se obtiene el aspecto modificado.

Tabla 32 Descripción del procedimiento almacenado ListarCIdadoRegistro.

Nombre del procedimiento almacenado:	ListarCIdadoRegistro	
Descripción:	Este procedimiento almacenado me devuelve una lista de Centros Informantes dado el código del Registro al que pertenecen.	
Parámetros	**Tipo**	**Descripción**
codReg	int	Este representa el código que identifica a un registro.
Resultado:	Se obtiene una lista de CI.	

Tabla 33 Descripción del procedimiento almacenado N_ClasificadorInsProc.

Nombre del procedimiento almacenado:	N_ClasificadorInsProc	
Descripción:	Este procedimiento almacenado inserta un nuevo clasificador.	
Parámetros	**Tipo**	**Descripción**
idNClasif	char (10)	Este representa el código que identifica a un clasificador.
descripNClasif	nvarchar (5)	Este representa la descripción de un clasificador.
aliasNClasif	char (10)	Este representa las siglas del clasificador.
Resultado:	Se inserta un nuevo clasificador.	

2.4 Diseño e implementación de la capa de acceso a datos para el Módulo Entrada de Datos.

La persistencia no es más que el almacenamiento de los datos en memoria, para una posterior recuperación de los mismos, por lo que para el desarrollo de las aplicaciones en la actualidad es muy importante la implementación de una capa de persistencia que permita la comunicación entre la capa del negocio y la capa de los datos sin que haya pérdida de información.

2.4.1 Diseño de la capa de acceso a datos para el Módulo de Entrada de Datos.

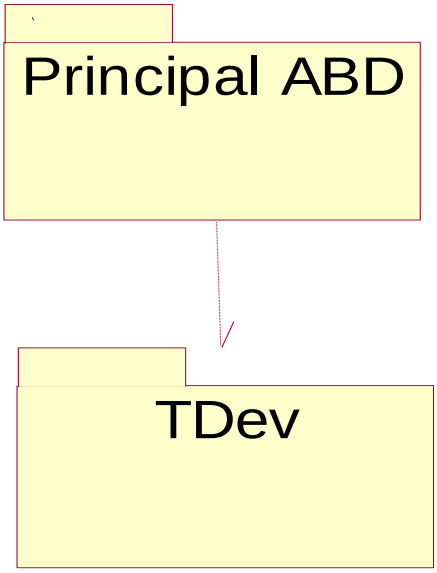

Figura 8 Diseño de la capa de acceso a datos para el módulo Entrada de Datos.

En la figura se representa el diseño de la capa de acceso a datos para el Módulo Entrada de Datos, representando el mismo a través de dos paquetes fundamentales, el Paquete Principal ABD y el paquete TDev.

En el paquete Principal ABD se encuentran agrupadas las clases GestorModeloAccesoBD, la interfaz IGestorModeloAccesoBD y las clases entidades que se utilizan cuando se implementan los métodos que permiten el acceso a los datos.

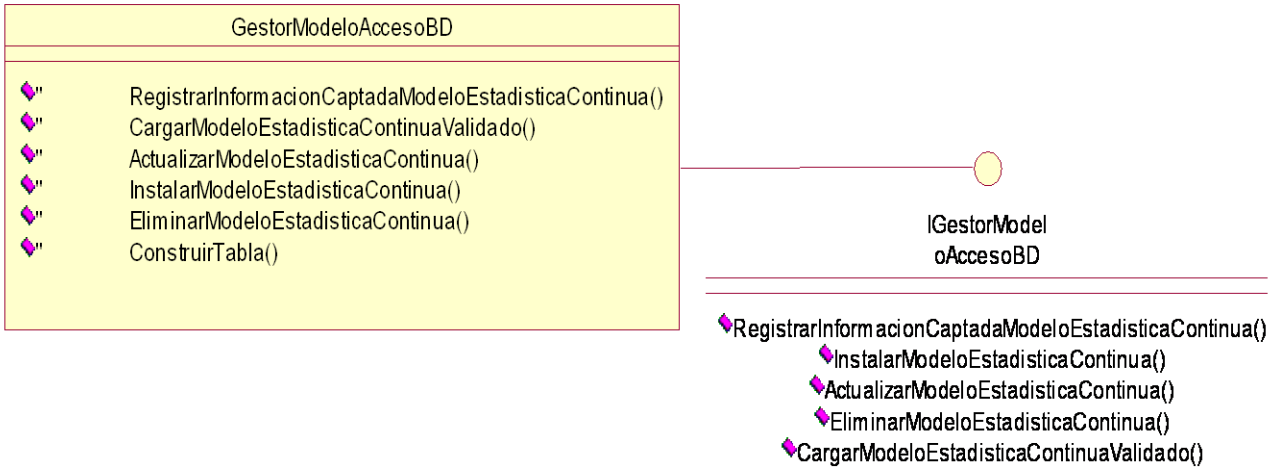

Figura 9 Diseño de clases: GestorModeloAccesoBD e IGestorModeloAccesoBD.

Tabla 34 Descripción de la clase GestorModeloAccesoBD.

Nombre	GestorModeloAccesoBD
Descripción	Esta clase es la encargada de gestionar el acceso a los datos.
Método	RegistrarInformacionCaptadaModeloEstadisticaContinua ()
Descripción	Este método registra la información que se obtiene como resultado de la captación de un modelo estadístico.
Método	InstalarModeloEstadisticaContinua ()
Descripción	Este método proporciona los elementos necesarios para crear un modelo.
Método	ActualizarModeloEstadisticaContinua ()

Descripción	Este método actualiza los cambios realizados en un modelo estadístico.
Método	EliminarModeloEstadisticaContinua ()
Descripción	Este método elimina un modelo estadístico.
Método	ConstruirTabla ()
Descripción	Permite estructurar la información que devuelve la base de datos en forma de tabla.
Método	CargarModeloEstadisticaContinuaValidado ()
Descripción	Este método es el responsable de cargar todos los datos de un modelo estadístico.

La interfaz IGestorModeloAccesoBD tiene la responsabilidad de brindar los servicios a la capa del negocio y de realizar las transformaciones de objetos del tipo del negocio a objetos del tipo de la base de datos y viceversa. Además define cómo se va a comunicar la capa del negocio con la capa de acceso a datos.

En el paquete TDev encontraremos las clases que genera el TierDeveloper por cada una de las tablas y las vistas que mapea de la base de datos, las cuales son:

Figura 10 Clase Aspecto.

Tabla 35 Descripción de la clase Aspecto.

Nombre	Aspecto		
Descripción	Esta clase contiene los posibles aspectos que puede contener un modelo.		
Atributos		Tipos	Descripción
idAsp		Int64	Este atributo representa el código del aspecto y guarda el número que identifica a cada aspecto que se encontrará en cada modelo.
idPadreA		Int64	Este atributo representa el código del padre de aspecto y guarda el número que identifica a un aspecto que este compuesto por sub-aspectos.
descripAsp		String	Este atributo guarda la descripción de las características de un aspecto determinado.
Método	Insert ()		
Descripción	Este método es el encargado de insertar un aspecto.		
Método	Load ()		
Descripción	Este método es el encargado de cargar un aspecto.		
Método	Update ()		
Descripción	Este método es el encargado de actualizar un aspecto.		
Método	Delete ()		
Descripción	Este método es el encargado de eliminar un aspecto.		

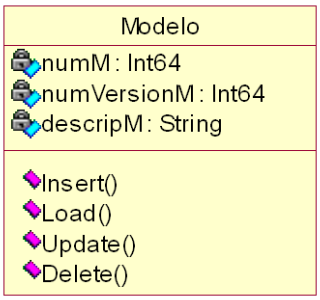

Figura 11 Clase Modelo.

Tabla 36 Descripción de la clase Modelo.

Nombre	Modelo	
Descripción	Esta clase contiene los posibles tipos de modelos que pueden utilizarse para la captación de información estadística.	
Atributos	Tipos	Descripción
numM	Int64	Este atributo representa el código del modelo y guarda el número que identifica a cada modelo que se utilizará en la captación de información.
numVersionM	Int64	Este atributo representa el número de la versión del modelo y guarda el número que identifica la versión de cada modelo.
descripM	String	Este atributo guarda la descripción de las características de un modelo determinado.
Método	Insert ()	
Descripción	Este método es el encargado de insertar un modelo.	

Método	Load ()
Descripción	Este método es el encargado de cargar un modelo.
Método	Update ()
Descripción	Este método es el encargado de actualizar un modelo.
Método	Delete ()
Descripción	Este método es el encargado de eliminar un modelo.

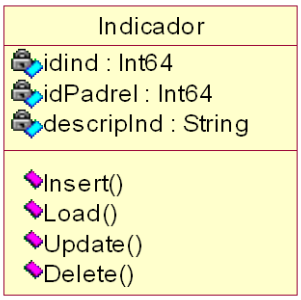

Figura 12 Clase Indicador.

Tabla 37 Descripción de la clase Indicador.

Nombre	Indicador	
Descripción	Esta clase representa los posibles indicadores que pueden conformar un modelo.	
Atributos	Tipos	Descripción
idind	Int64	Este atributo guarda el código que diferencia a un indicador de otro.

idPadreI	Int64	Este atributo representa el código del padre de un indicador del cual se desglosan otros. indicadores
descripInd	String	Este atributo guarda la descripción de las características de un indicador determinado.

Método	Insert ()
Descripción	Este método es el encargado de insertar un indicador.
Método	Load ()
Descripción	Este método es el encargado de cargar un indicador.
Método	Update ()
Descripción	Este método es el encargado de actualizar un indicador.
Método	Delete ()
Descripción	Este método es el encargado de eliminar un indicador.

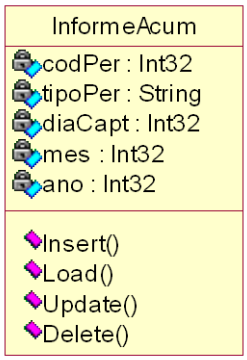

Figura 13 Clase InformeAcum.

Tabla 38 Descripción de la clase InformeAcum.

Nombre	InformeAcum		
Descripción	Esta clase representa los posibles Informes Acumulados que puede tener un modelo.		
Atributos		Tipos	Descripción
codPer		Int32	Este atributo representa el código de la periodicidad que se haya definido para el llenado del modelo.
tipoPer		String	Este atributo representa al tipo de periodicidad que va a estar definido por el tiempo en que se realiza el llenado del modelo. Puede ser: mensual, trimestral, anual, etc.
diaCapt		Int32	Este atributo representa el día en que se registran los datos del informe acumulado.
mes		Int32	Este atributo representa el mes en que se registran los datos del informe acumulado.
ano		Int32	Este atributo representa el año en que se registran los datos del informe acumulado.
Método	Insert ()		
Descripción	Este método es el encargado de insertar los datos que conforman el Informe Acumulado.		
Método	Load ()		
Descripción	Este método es el encargado de cargar los datos que conforman el Informe Acumulado.		
Método	Update ()		

Descripción	Este método es el encargado de actualizar los datos que conforman el Informe Acumulado.
Método	Delete ()
Descripción	Este método es el encargado de eliminar los datos que conforman el Informe Acumulado.

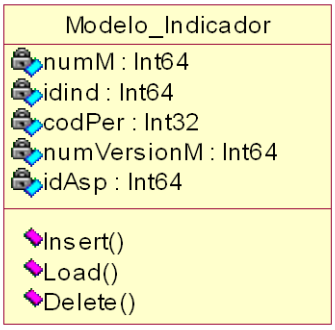

Figura 14 Clase Modelo_Indicador.

Tabla 39 Descripción de la clase Modelo_Indicador.

Nombre	Modelo_Indicador	
Descripción	Esta clase representa la relación entre los modelos y los indicadores que lo conforman.	
Atributos	Tipos	Descripción
numM	Int64	Este atributo representa el código del modelo y guarda el número que identifica a cada modelo que se utilizará en la captación de información.
idind	Int64	Este atributo guarda el código que diferencia a un

		indicador de otro.
codPer	Int32	Este atributo representa el código de la periodicidad que se haya definido para el llenado del modelo.
numVersionM	Int64	Este atributo representa el número de la versión del modelo y guarda el número que identifica la versión de cada modelo.
idAsp	Int64	Este atributo representa el código del aspecto y guarda el número que identifica a cada aspecto que se encontrará en cada modelo.

Método	Insert ()
Descripción	Este método es el encargado de insertar los datos de la clase Modelo_Indicador.
Método	Load ()
Descripción	Este método es el encargado de cargar los datos de la clase Modelo_Indicador.
Método	Delete ()
Descripción	Este método es el encargado de eliminar los datos de la clase Modelo_Indicador.

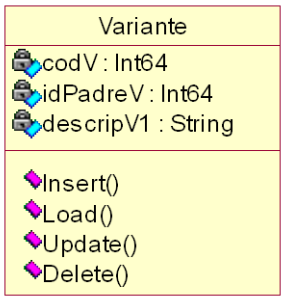

Figura 15 Clase Variante.

Tabla 40 Descripción de la clase Variante.

Nombre	Variante		
Descripción	Esta clase representa a las posibles variantes que puede tener un modelo.		
Atributos		Tipos	Descripción
codV		Int64	Este atributo representa el código de la variante y guarda el número que identifica a la variante que tendrá un modelo.
idPadreV		Int64	Este atributo representa el código padre de variante y guarda el número que identifica a una variante en caso de que esta esté compuesta por sub-variantes. Este campo puede ser nulo.
descripV1		String	Este atributo guarda la descripción de la variante.
Método	Insert ()		
Descripción	Este método es el encargado de insertar la variante.		
Método	Load ()		

Descripción	Este método es el encargado de cargar la variante.
Método	Update ()
Descripción	Este método es el encargado de actualizar la variante.
Método	Delete ()
Descripción	Este método es el encargado de eliminar la variante.

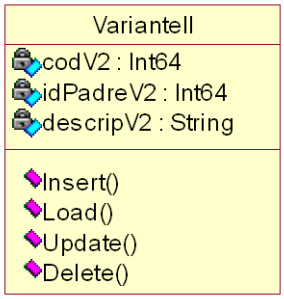

Figura 16 Clase VarianteII.

Tabla 41 Descripción de la clase VarianteII.

Nombre	VarianteII	
Descripción	Esta clase representa a las posibles variantes 2 que puede tener un modelo.	
Atributos	Tipos	Descripción
codV2	Int64	Este atributo representa el código de la variante 2 y guarda el número que identifica a la variante 2 que tendrá un modelo.
idPadreV2	Int64	Este atributo representa el código padre de variante 2 y

		guarda el número que identifica a una variante 2 en caso de que esta esté compuesta por sub-variantes. Este campo puede ser nulo.
descripV2	String	Este atributo guarda la descripción de la variante 2.

Método	Insert ()
Descripción	Este método es el encargado de insertar la variante 2.
Método	Load ()
Descripción	Este método es el encargado de cargar la variante 2.
Método	Update ()
Descripción	Este método es el encargado de actualizar la variante 2.
Método	Delete ()
Descripción	Este método es el encargado de eliminar la variante 2.

Figura 17 Clase Modelo_Aspecto_Indicador.

Tabla 42 Descripción de la clase Modelo_Aspecto_Indicador.

Nombre	Modelo_Aspecto_Indicador	
Descripción	Esta clase representa el valor que contiene cada una de las celdas en un modelo.	
Atributos	Tipos	Descripción
numM	Int64	Este atributo representa el código del modelo y guarda el número que identifica a cada modelo que se utilizará en la captación de información.
idind	Int64	Este atributo guarda el código que diferencia a un indicador de otro.
idAsp	Int64	Este atributo representa el código del aspecto y guarda el número que identifica a cada aspecto que se

		encontrará en cada modelo.
codPer	Int32	Este atributo representa el código de la periodicidad que se haya definido para el llenado del modelo.
codV	Int64	Este atributo representa el código de la variante y guarda el número que identifica a la variante que tendrá un modelo.
codV2	Int64	Este atributo representa el código de la variante 2 y guarda el número que identifica a la variante 2 que tendrá un modelo.
valor	Double	Este atributo representa el valor que contiene una determinada celda en el modelo.
pagina	Int32	Este atributo representa el número de página del modelo.
idOME	Int64	Este atributo representa al número que identifica a cada una de las Oficinas Municipales de Estadística dentro de un territorio.
codRegistro	Int64	Este atributo representa al código de un Registro y guarda un número que identifica al Registro.
numCI	Int64	Este atributo representa le código de un Centro Informante y guarda un número que identifica a cada centro informante dentro de un municipio.
idOTE	Int64	Este atributo representa el código de una Oficina Territorial de Estadísticas y es un número que identifica a cada una de las 15 Oficinas Territoriales de

		Estadística dentro del país
numVersionM	Int64	Este atributo representa el número de la versión del modelo y guarda el número que identifica la versión de cada modelo.

Método	Insert ()
Descripción	Este método es el encargado de insertar el valor de una celda determinada.
Método	Load ()
Descripción	Este método es el encargado de cargar el valor de una celda determinada.
Método	Delete ()
Descripción	Este método es el encargado de eliminar el valor de una celda determinada.

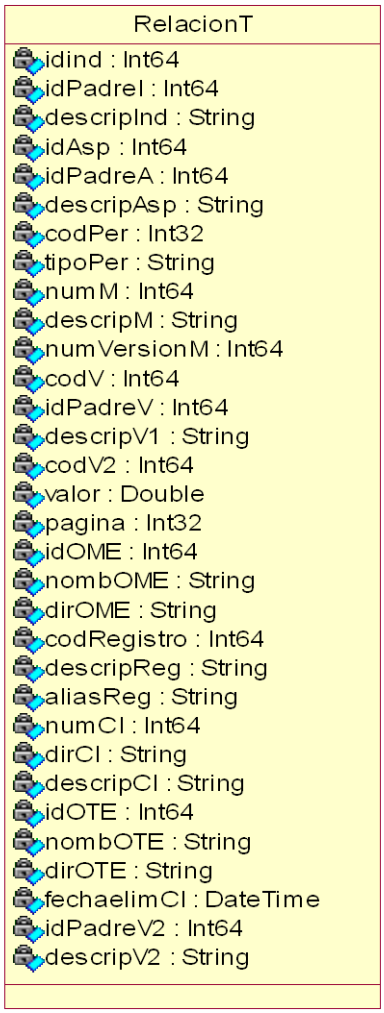

Figura 18 Clase RelacionT.

Tabla 43 Descripción de la clase RelacionT.

Nombre	RelacionT
Descripción	Esta clase es el resultado de una vista en la base de datos, donde agrupo un conjunto de tablas que necesito relacionadas para dar respuesta a determinadas consultas.

Atributos	Tipos	Descripción
idind	Int64	Este atributo guarda el código que diferencia a un indicador de otro.
idPadreI	Int64	Este atributo representa el código del padre de un indicador del cual se desglosan otros. indicadores
descripInd	String	Este atributo guarda la descripción de las características de un indicador determinado.
idAsp	Int64	Este atributo representa el código del aspecto y guarda el número que identifica a cada aspecto que se encontrará en cada modelo.
idPadreA	Int64	Este atributo representa el código del padre de aspecto y guarda el número que identifica a un aspecto que este compuesto por sub-aspectos.
descripAsp	String	Este atributo guarda la descripción de las características de un aspecto determinado.
codPer	Int32	Este atributo representa el código de la periodicidad que se haya definido para el llenado del modelo.
tipoPer	String	Este atributo representa al tipo de periodicidad que va a estar definido por el tiempo en que se realiza el llenado del modelo. Puede ser: mensual, trimestral, anual, etc.
numM	Int64	Este atributo representa el código del modelo y guarda el número que identifica a cada modelo que se utilizará en la captación de información.

descripM	String	Este atributo guarda la descripción de las características de un modelo determinado.
numVersionM	Int64	Este atributo representa el número de la versión del modelo y guarda el número que identifica la versión de cada modelo.
codV	Int64	Este atributo representa el código de la variante y guarda el número que identifica a la variante que tendrá un modelo.
idPadreV	Int64	Este atributo representa el código padre de variante y guarda el número que identifica a una variante en caso de que esta esté compuesta por sub-variantes. Este campo puede ser nulo.
descripV1	String	Este atributo guarda la descripción de la variante.
codV2	Int64	Este atributo representa el código de la variante 2 y guarda el número que identifica a la variante 2 que tendrá un modelo.
valor	Double	Este atributo representa el valor que contiene una determinada celda en el modelo.
pagina	Int32	Este atributo representa el número de página del modelo.
idOME	Int64	Este atributo representa al número que identifica a cada una de las Oficinas Municipales de Estadística dentro de un territorio.

nombOME	String	Este atributo guarda el nombre de cada una de las Oficinas Municipales de Estadísticas ubicadas en cada territorio.
dirOME	String	Este atributo guarda la dirección física real de cada una de las Oficinas Municipales de Estadísticas ubicadas en cada territorio.
codRegistro	Int64	Este atributo representa al código de un Registro y guarda un número que identifica al Registro.
descripReg	String	Este atributo guarda la descripción de las características del Registro.
aliasReg	String	Este atributo representa el alias del Registro y guarda las siglas que identifican el Registro.
numCI	Int64	Este atributo representa le código de un Centro Informante y guarda un número que identifica a cada Centro Informante dentro de un municipio.
dirCI	String	Este atributo representa la dirección de un Centro Informante y guarda la dirección física de cada uno de los Centros Informantes.
descripCI	String	Este atributo guarda las características que definen al Centro Informante dentro de un territorio.
idOTE	Int64	Este atributo representa el código de la Oficina Territorial de Estadísticas, el cual es un número que identifica a cada una de las 15 Oficinas Territoriales de Estadística dentro del país.
nombOTE	String	Este atributo guarda el nombre de cada una de las 15 Oficinas Territoriales de Estadísticas ubicadas en cada

		una de las provincias de nuestro país.
dirOTE	String	Este atributo guarda la dirección física real de cada una de las 15 Oficinas Territoriales de Estadísticas ubicadas en cada una de las provincias de nuestro país.
fechaelimCI	DateTime	Este atributo representa la fecha de eliminación de un Centro Informante y guarda la fecha en que se elimina un Centro Informante.
idPadreV2	Int64	Este atributo representa el código padre de variante 2 y guarda el número que identifica a una variante 2 en caso de que esta esté compuesta por sub-variantes. Este campo puede ser nulo.
descripV2	String	Este atributo guarda la descripción de la variante 2.

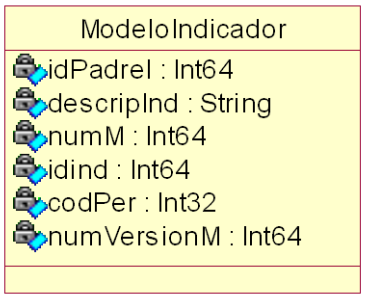

Figura 19 Clase ModeloIndicador.

Tabla 44 Descripción de la clase ModeloIndicador.

Nombre	ModeloIndicador
Descripción	Esta clase es el resultado de una vista en la base de datos, donde agrupo un conjunto

	de tablas que necesito relacionadas para dar respuesta a determinadas consultas.

Atributos	Tipos	Descripción
idPadreI	Int64	Este atributo representa el código del padre de un indicador del cual se desglosan otros. indicadores
descripInd	String	Este atributo guarda la descripción de las características de un indicador determinado.
numM	Int64	Este atributo representa el código del modelo y guarda el número que identifica a cada modelo que se utilizará en la captación de información.
idind	Int64	Este atributo guarda el código que diferencia a un indicador de otro.
codPer	Int32	Este atributo representa el código de la periodicidad que se haya definido para el llenado del modelo.
numVersionM	Int64	Este atributo representa el número de la versión del modelo y guarda el número que identifica la versión de cada modelo.

Además en este paquete TDev están las clases Factory, Entity y Collection que son propias del TierDeveloper y se utilizan en la implementación de los métodos que pertenecen a la clase GestorModeloAccesoDB para manipular la información.

2.4.2 Implementación de la Capa de acceso a datos del Módulo Entrada de Datos.

A continuación se muestra el diagrama de componentes para la capa de acceso a datos del Módulo Entrada de Datos.

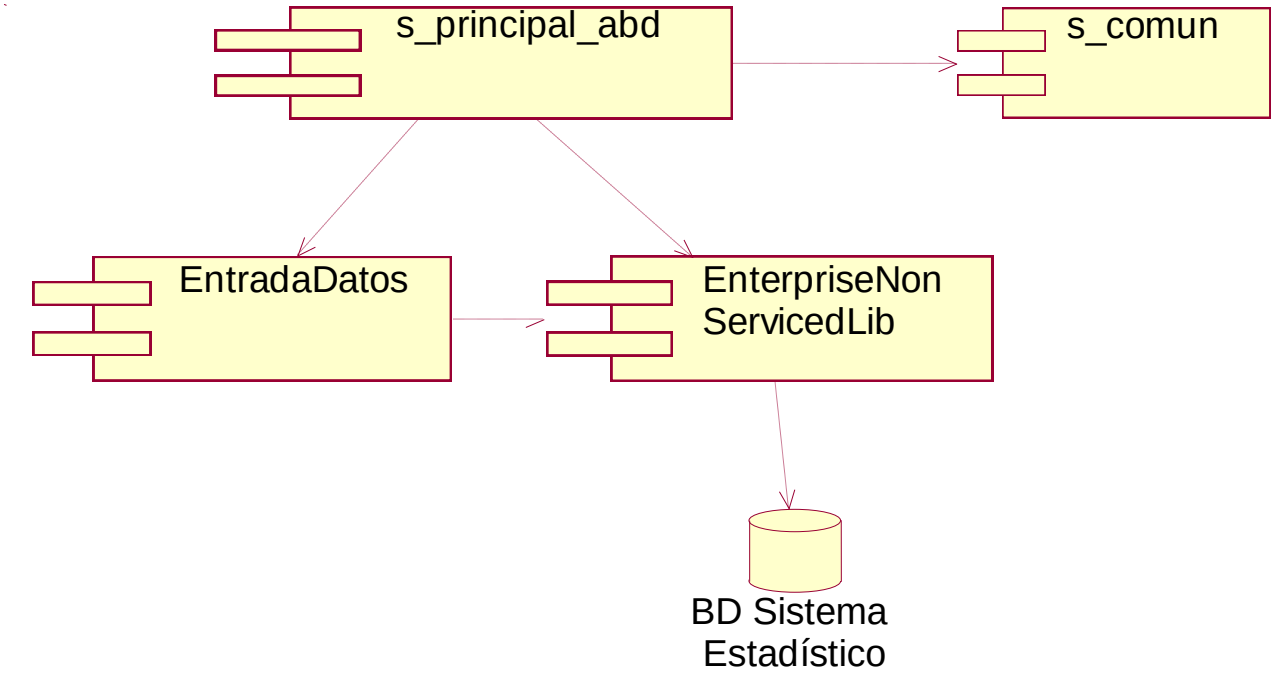

Figura 20 Diagrama de componentes para la capa de acceso a datos del módulo Entrada de Datos.

Los componentes s_principal_abd.dll y s_comun.dll se encuentran ubicados en el paquete Principal ABD antes mencionado. En el componente s_principal_abd.dll se encuentra implementada la clase GestorModeloAccesoBD y la interfaz IGestorModeloAccesoBD. En el componente s_comun.dll se encuentran implementadas las clases entidades. Los componentes EntradaDatos.dll y EnterpriseNonServicedLib.dll se encuentran ubicados en el paquete TDev. En el componente EntradaDatos.dll se encuentran implementadas las clases resultantes del mapeo de las tablas y vistas de la base de datos, y las clases Factory, Entity y Collection del TierDeveloper. El componente EnterpriseNonServicedLib.dll representa el Framework de TierDeveloper que utiliza para establecer la conexión con la base de datos Sistema Estadístico, entre otras funcionalidades. El componente BD

Sistema Estadístico representa la base de datos Sistema Estadístico en la cual se encuentra almacenada la información con la que trabaja el Módulo Entrada de Datos.

2.5 Justificación de las herramientas utilizadas para el desarrollo de la solución.

SQL Server 2000.

El uso de esta herramienta brinda varias ventajas como son:

- El SQL Server es un gestor de base de datos que soporta transacciones, lo que garantiza que en las Oficinas de Estadísticas no haya pérdida de información o cambios no deseados cuando se esté realizando alguna operación sobre la base de datos y ocurra algún fallo eléctrico o de cualquier otro tipo en el sistema. En caso de que ocurra esto, todas las operaciones que se estaban realizando vuelven a su estado inicial. Esto permite que el resultado del estudio de la información estadística sea lo más cercano a la realidad y sea considerada como una fuente confiable por la dirección del país para la toma de decisiones.
- El SQL Server tiene facilidad para manipular grandes volúmenes de información y este es un aspecto que hay que tener en cuenta en nuestro proyecto puesto que en la actividad estadística se maneja gran cantidad de datos.
- Como en nuestro proyecto se manipulan grandes volúmenes de información se hace necesario el uso de SQL Server puesto que este gestor soporta el trabajo con procedimientos almacenados. Esta característica brinda la posibilidad de ejecutar las consultas más probables en procedimientos almacenados desde el propio gestor, lo cual agiliza el trabajo con la base de datos.
- En las Oficinas de Estadísticas se realiza actividad diaria, por lo que se hace necesario el uso de SQL Server ya que permite la realización de un trabajo continuo y con pocas posibilidades de fallos.
- Como en las Oficinas de Estadística se trabaja con información clasificada, es muy importante almacenar la información en un gestor que brinde seguridad a los datos. Este es el caso del SQL Server puesto que desarrolla una política de trabajo que provee a sus bases de datos de buena

seguridad. Permite la definición de roles y la gestión del nivel de acceso de cada uno de ellos en el trabajo con la información almacenada, limitando el acceso de los usuarios según lo determine el administrador del sistema.

- En las Oficinas de Estadísticas se realiza un intercambio de información entre los diferentes niveles en que están estructuradas. Por esta razón se hace necesario el uso de SQL Server porque es un gestor que soporta réplica de información, permite la réplica de información entre varios servidores, posibilitando que se actualicen al realizar algún cambio en la información almacenada en cualquiera de los servidores. Este procedimiento lo hace de forma automática y segura, evitando la perdida o cambio no deseado de la información.
- No requiere muchos recursos y memoria de la máquina para un buen funcionamiento.
- Cuba no tiene que pagar licencia para utilizar este producto, ya que el producto que se va a desarrollar no se va a comercializar, sino que será un producto nacional.

Erwin:

El uso de esta herramienta brinda varias ventajas como son:

- En nuestro proyecto desarrollamos la base de datos en un gestor relacional, por lo que es conveniente el uso del Erwin puesto que esta herramienta modela el diseño de las bases de datos de modo relacional.
- En nuestro proyecto, para lograr un correcto diseño de base de datos, es necesario realizar un diseño lógico, y luego a partir de este se obtener un diseño físico de la base de datos. Esto te permite lograr que el diseño de la base de datos tenga en cuenta todos los detalles referentes al negocio que se esta modelando. La herramienta Erwin permite el diseño y construcción de bases de datos a nivel lógico y físico permitiendo la sincronización bidireccional entre ambos diseños, lo que posibilita que si se hace algún cambio en uno de los dos diseños antes mencionados, se refleje de forma automática en el otro.
- Como en nuestra base de datos se manipulan grandes volúmenes de información pues es conveniente utilizar la herramienta CASE Erwin debido a que esta permite la generación automática de procedimientos almacenados que ayudaran a agilizar el trabajo con la base de datos.

- Erwin está equipado para crear y manejar diseños de bases de datos funcionales y confiables.
- Ofrece la posibilidad de construir automáticamente bases de datos y documentación basada en HTML.
- Permite la creación de reportes de forma sencilla.
- Da la posibilidad de hacer reingeniería inversa de bases de datos.

TierDeveloper:

El trabajo con esta herramienta brinda varios beneficios entre los que encontramos los siguientes:

- En nuestro proyecto se trabaja con un lenguaje de programación orientado a objetos y se almacena la información en un gestor relacional, por lo que se hace necesario el uso de la herramienta TierDeveloper puesto que permite realizar el mapeo objeto relacional ayudando a atenuar los problemas de incompatibilidad entre el modelo relacional y el modelo orientado a objetos, pues genera un esquema de persistencia mapeando las tablas de la base de datos, para después trabajar orientado a objetos.
- En el proyecto es necesario haber terminado el producto final en un tiempo acordado y con calidad, por lo que se hace necesario el uso de esta herramienta puesto que nos permite generar el código de acceso a datos de forma automática y libre de errores, lo cual le brinda seguridad al sistema, que es un elemento que le aporta calidad al programa.
- Es una herramienta de 4^{ta} generación que permite generar capa de persistencia para varios gestores de base de datos.
- Se conecta en tiempo de ejecución al SQL y analiza la estructura de la base de datos, lo cual le permite la sincronización del código cuando se realizan modificaciones a la base de datos.

Capítulo 3 Validación del diseño realizado.

En este capítulo se realiza las principales validaciones de las tablas que conforman el diseño de los módulos Gestión de Usuarios y Sistema Estadístico de la base de datos del SIGE. Además se da una explicación de cómo se comprueba el nivel de normalización que tienen los esquemas relacionales de las tablas de los módulos antes mencionados.

3.1 Proceso de Normalización de la base de datos.

Cuando se va a realizar el diseño de una base de datos relacional, se tiene en cuenta que las tablas que la componen cumplan con las reglas de normalización, que no es más que el proceso que permite eliminar la redundancia de los datos y evitar los problemas al insertar, eliminar y actualizar los datos, es decir, optimizar el trabajo con la información almacenada. En este proceso existen varios niveles de normalización, pero los tres primeros son los más usados. Las reglas de normalización se relacionan entre ellas de forma dependiente, lo que indica que para que una base de datos se encuentre en una determinada forma normal, primeramente tiene que cumplir con las reglas de los niveles inferiores de normalización. Esto quiere decir que una base de datos esta en 2da Forma Normal si previamente cumple con las reglas de normalización del 1er nivel. Cada regla que se cumple aumenta el grado de normalización del esquema de relación. Si una regla no se cumple, el esquema se debe descomponer en varios esquemas de relación que si la cumplan por separado. Se dice que una base de datos se encuentra en determinada forma normal si cumple con las restricciones correspondientes a dicha forma normal.

Se puede afirmar que los diseños de los módulos de la base de datos propuestos se encuentran normalizados hasta Tercera Forma Normal. Se puede plantear que los esquemas de relación para ambos módulos están en 1ra Forma Normal puesto que podemos asegurar que cada tupla contiene exactamente un valor para cada atributo de las tablas de la base de datos, es decir, no existen campos multivaluados. También se cumple que los esquemas de relación están en 2da Forma Normal, porque primeramente se encuentran en 1ra Forma Normal, y además todos los atributos que no son claves en las tablas, dependen totalmente de la clave primaria. Finalmente se puede plantear que los esquemas de relación se encuentran en 3ra Forma Normal, porque primeramente se encuentran en 2da Forma Normal, y además que no existen dependencias transitivas entre llaves candidatas y atributos no primos.

3.2 Validación para las tablas del módulo Sistema Estadístico la base de datos del SIGE.

Tabla 45 Validaciones de la tabla OTE.

Tabla	OTE		
Atributo	**No Nulo**	**Llave**	**Tabla Foránea**
idOTE	X	PK	
dirOTE	X		
nombOTE	X		
Atributo unitario	idOTE		
Chequeo	idOTE > 0 AND idOTE < 17		
Descripción	El código idOTE tiene que ser mayor que 0 y menor que 17, para que garantice que exista un código por cada una de las 16 OTE.		

Tabla 46 Validaciones de la tabla OME.

Tabla	OME		
Atributo	**No Nulo**	**Llave**	**Tabla Foránea**
idOME	X	PK	
dirOME	X		
nombOTE	X		

idOTE	X	PKF	OTE
Atributo unitario	idOME, idOTE		

Tabla 47 Validaciones de la tabla Registro.

Tabla	Registro		
Atributo	**No Nulo**	**Llave**	**Tabla Foránea**
codRegistro	X	PK	
descripReg	X		
aliasReg	X		
Atributo unitario	codRegistro		

Tabla 48 Validaciones de la tabla CI.

Tabla	CI		
Atributo	**No Nulo**	**Llave**	**Tabla Foránea**
numCI	X	PK	
descripCI	X		
idOME	X	PKF	OME
idOTE	X	PKF	OTE
codRegistro	X	PKF	Registro

Atributo unitario	numCI, idOME, idOTE, codRegistro

Tabla 49 Validaciones de la tabla Clasificacion.

Tabla	Clasificacion		
Atributo	**No Nulo**	**Llave**	**Tabla Foránea**
idClasif	X	PK	
idNClasif	X	PKF	N_Clasificador
descripC	X		
aliasC			
Atributo unitario	idClasif, idNClasif		

Tabla 50 Validaciones de la tabla Modelo.

Tabla	Modelo		
Atributo	**No Nulo**	**Llave**	**Tabla Foránea**
numM	X	PK	
numVersionM	X		
descripM	X		
Atributo unitario	numM		

Tabla 51 Validaciones de la tabla N_Clasificador.

Tabla	N_Clasificador		
Atributo	**No Nulo**	**Llave**	**Tabla Foránea**
idNClasif	X	PK	
descripNClasif	X		
aliasNClasif	X		
Atributo unitario	idOTE		
Chequeo	idNClasif like '[1-9][0-9][0-9][0-9][0-9]'		
Descripción	EL código idNClasif tiene que estar conformado obligatoriamente por 5 dígitos		

Tabla 52 Validaciones de la tabla InformeAcum.

Tabla	InformeAcum		
Atributo	**No Nulo**	**Llave**	**Tabla Foránea**
codPer	X	PK	
tipoPer			
diaCapt	X		
mes	X		
ano	X		
Atributo unitario	codPer		

Chequeo	diaCapt >0 AND diaCapt < 32
Descripción	El atributo diaCapt representa un día que puede tomar un valor entre 1 y 31.
Chequeo	mes > 0 AND mes <13
Descripción	El atributo mes, representa un mes que puede tomar un valor entre 1 y 12.
Chequeo	ano > 1900 AND ano < 3000
Descripción	El atributo ano, representa un año que puede tomar un valor mayor que 1900 y menor que 3000.

Tabla 53 Validaciones de la tabla Indicador.

Tabla	Indicador		
Atributo	**No Nulo**	**Llave**	**Tabla Foránea**
idind	X	PK	
idPadreI		FK	Indicador
descripInd			
Atributo unitario	idind		

Tabla 54 Validaciones de la tabla Variante.

Tabla	Variante		
Atributo	**No Nulo**	**Llave**	**Tabla Foránea**

codV	X	PK	
idPadreV		FK	Variante
descripV1	X		
Atributo unitario	codV		

Tabla 55 Validaciones de la tabla Variante_2.

Tabla	Variante_2		
Atributo	**No Nulo**	**Llave**	**Tabla Foránea**
codV2	X	PK	
idPadreV2		FK	Variante_2
descripV2	X		
Atributo unitario	codV2		

Tabla 56 Validaciones de la tabla Aspecto.

Tabla	Aspecto		
Atributo	**No Nulo**	**Llave**	**Tabla Foránea**
idAsp	X	PK	
idPadreA		FK	Aspecto
descripAsp	X		

Atributo unitario	idAsp

3.3 Validación para las tablas del módulo Gestión de Usuarios de la base de datos del SIGE.

Tabla 57 Validaciones de la tabla evento.

Tabla	evento		
Atributo	**No Nulo**	**Llave**	**Tabla Foránea**
id_evento	X	PK	
evento	X		
descripcion			
Atributo unitario	id_evento		

Tabla 58 Validaciones de la tabla t_usuario.

Tabla	t_usuario		
Atributo	**No Nulo**	**Llave**	**Tabla Foránea**
id_usuario	X	PK	
nombre			
apellidos			

usuario			
clave			
Atributo unitario	id_usuario		

Tabla 59 Validaciones de la tabla accion.

Tabla	accion		
Atributo	**No Nulo**	**Llave**	**Tabla Foránea**
id_accion	X	PK	
id_evento	X	PKF	evento
id_usuario	X	PKF	T_usuario
fecha			
hora			
Atributo unitario	id_accion, id_evento, id_usuario		
Chequeo	fecha <= CURRENT_DATE		
Descripción	Verifica que la fecha en que se realice una accion no sea mayor que la fecha actual.		

Tabla 60 Validaciones de la tabla t_rol.

Tabla	t_rol

Atributo	No Nulo	Llave	Tabla Foránea
id_rol	X	PK	
rol			
id_dominio	X	PKF	t_dominio
Atributo unitario	id_rol, id_dominio		

Tabla 61 Validaciones de la tabla rol_usuario.

Tabla	rol_usuario		
Atributo	**No Nulo**	**Llave**	**Tabla Foránea**
id_usuario	X	PKF	t_usuario
id_rol	X	PKF	t_rol
id_dominio	X	PKF	t_dominio
Atributo unitario	id_usuario, id_rol, id_dominio		

Tabla 62 Validaciones de la tabla t_permiso t_rol.

Tabla	t_permiso t_rol		
Atributo	**No Nulo**	**Llave**	**Tabla Foránea**

id_rol	X	PKF	t_rol
id_dominio	X	PKF	t_dominio
id_permiso	X	PKF	t_permiso
Atributo unitario	id_rol, id_dominio, id_permiso		

Tabla 63 Validaciones de la tabla t_permiso.

Tabla	t_permiso		
Atributo	**No Nulo**	**Llave**	**Tabla Foránea**
id_permiso	X	PK	
id_dominio	X	PKF	t_dominio
permiso			
Atributo unitario	id_permiso, id_dominio		

Tabla 64 Validaciones de la tabla t_dominio.

Tabla	t_dominio		
Atributo	**No Nulo**	**Llave**	**Tabla Foránea**
id_dominio	X	PK	
id_padDominio	X	FK	t_dominio

dominio			
Atributo unitario	id_dominio		

Conclusiones

Con la terminación de este trabajo se puede plantear que se le ha dado cumplimiento a todos los objetivos del mismo.

Con el diseño e implementación del módulo Gestión de Usuarios de la base de datos se logró garantizar que la información estadística con que se trabaja esté protegida, ya que controla que en el Sistema Integrado de Gestión Estadística pueda autenticarse solamente el personal autorizado, y con el nivel de acceso que determinó el administrador de la aplicación, así como mantener el control y registro de las acciones que cada usuario realiza en el sistema.

Con el diseño e implementación del módulo Sistema Estadístico de la base de datos se obtuvo como resultado la posibilidad de gestionar la información estadística de forma eficiente, eliminando la posibilidad que se trabaje con información no válida, que no exista información redundante en la base de datos, que la información este organizada para que facilite el trabajo con los datos que se manejan, que los resultados obtenidos del estudio de la información estadística sean lo más cercano a la realidad, para que la dirección del país pueda basarse en ellos para tomar decisiones importantes.

Con el diseño de la capa de acceso a datos para el módulo Entrada de Datos se logró la implementación del acceso de dicho módulo de la aplicación a la base de datos para brindar las funcionalidades a los usuarios, eliminando la posibilidad de que exista pérdida de información, o se realicen cambios no deseados, producto de algún fallo eléctrico o de cualquier otro tipo en el sistema.

Recomendaciones

Con la realización del presente trabajo podemos recomendar:

Que se aplique el diseño del módulo Gestión de Usuarios de la base de datos del SIGE a las bases de datos de otros proyectos para que garanticen la seguridad de los sistemas.

Que se aplique el diseño del módulo Sistema Estadístico de la base de datos del SIGE a las bases de datos de otros proyectos que gestionen información estadística.

Que el diseño de la capa de acceso a datos del módulo de Entrada de Datos se implemente de igual forma en el resto de los módulos del proyecto, como son el módulo Generador de Reportes y el módulo de Registros y Clasificadores.

El empleo del gestor SQL Server 2000 para almacenar y gestionar la información en otros sistemas.

La utilización del TierDeveloper para el desarrollo automático de capas de acceso a datos en otros sistemas.

Bibliografía

[1] Marqués A. M. (2001). Electronic Source: *Apuntes de Ficheros y Bases de Datos.* <http://www3.uji.es/~mmarques/f47/apun/apun.html> [Consulta: 10 de enero del 2007].

[2] Date C. J. (2003). Introducción a los Sistemas de Bases de Datos. La Habana: Editorial Félix Varela.

[3] Electronic Source: *Sistema de Información y Base de Datos.* 8 de marzo del 2006. < http://kybele.escet.urjc.es/documentos/EI/T3_BD.pdf> [Consulta: 11 de enero del 2007].

[4] Electronic Source: *Base de datos.* <http://es.wikipedia.org/wiki/Base_de_datos#Tipos_de_bases_de_datos#Tipos_de_bases_de_datos> [Consulta: 13 de enero del 2007].

[5] Booch G. Electronic Source: *Diseño Orientado a Objetos con Aplicaciones.*

[6] Woody B. (2006) Electronic Source: *Administrator's Guide to SQL Server 2005.* Addison Wesley Professional.

[7] Lockhart T. (1996). Electronic Source: *Manual del usuario de PostgreSQL.*

[8] Martínez P. A. (2001). Electronic Source: *Un Sistema de Gestión de Bases de Datos Orientadas a Objetos sobre una Máquina Abstracta Persistente.* Oviedo.

[9] Fanjul A. (2005). Electronic Source: *db4objects.* < http://mhproject.org/media/blogs/mhpenlaces/Interno/Presentaciones/db4objects/db4objects.pdf> [Consulta: 2 de febrero del 2007].

[10] Valle J. (2005). Electronic Source: *Herramientas CASE para BD.* <http://www.monografias.com/trabajos24/herramientas-case/herramientas-case.shtml> [Consulta: 18 de febrero del 2007].

[11] Alachisoft. (2005). Electronic Source: *Fast Track Guide for .NET.*

[12] Gavin K. Ch. (2005). Electronic Source: *Hibernate in Action.* Greenwich: Manning Publications Co.

Glosario de Términos

ONE: Oficina Nacional de Estadísticas. Oficina encargada de las estadísticas a nivel nacional.

OTE: Oficina Territorial de Estadísticas. Oficina encargada de las estadísticas a nivel territorial. Las oficinas a nivel territorial son generalmente oficinas provinciales aunque en algunos casos existen OTE 's que realizan las estadísticas de un territorio que no se considera una provincia, como por ejemplo: La Isla de la Juventud.

OME: Oficina Municipal de Estadísticas. Oficina encargada de las estadísticas a nivel municipal.

MicroSet NT: Software utilizado actualmente por la ONE para procesar los modelos.

Centros Informantes (CI): Los Centros Informantes son las empresas u organismos que deben dar parte a las oficinas de estadísticas. En algunos casos existen establecimientos que pueden ser considerados Centros Informantes.

Aspecto: Definen los datos que van en las columnas de los modelos.

Modelo: Especie de planilla compuesta por tablas, cuadres e indicadores.

Indicador: se dice de una variable que puede tomar un valor de una determinada unidad de medida y de un determinado tipo de datos (generalmente numérico). Los indicadores de la ONE están bien definidos y tienen un código que los identifica.

Variante: Un modelo contiene variantes que especifican si el mismo pertenece a una empresa estatal o privada y a que organización. Aunque en otros caso permite especificar si el modelo tiene frecuencia mensual, trimestral o anual.

Periodicidad: Se refiere a la frecuencia con que se los centro informantes deben de entregar la información. Puede ser mensual, trimestral, anual.

SIGE: Sistema Integrado de Gestión Estadística.