DESARROLLO DE HERRAMIENTA DE SOFTWARE PARA DETERMINAR LA CALIDAD DE LOS DATOS DE UNA BASE DE DATOS RELACIONAL

UNIVERSIDAD PONTIFICIA BOLIVARIANA

VICERRECTORÍA ACADÉMICA

SISTEMA DE BIBLIOTECAS

2013

DESARROLLO DE HERRAMIENTA DE SOFTWARE PARA DETERMINAR LA CALIDAD DE LOS DATOS DE UNA BASE DE DATOS RELACIONAL

DAVID SANTIAGO PALACIO COLORADO

DANIEL RODRÍGUEZ GONZÁLEZ

UNIVERSIDAD PONTIFICIA BOLIVARIANA

ESCUELA DE INGENIERÍAS

FACULTAD DE INGENIERÍA INFORMÁTICA Y TELECOMUNICACIONES

MEDELLÍN

2013

DESARROLLO DE HERRAMIENTA DE SOFTWARE PARA DETERMINAR LA CALIDAD DE LOS DATOS DE UNA BASE DE DATOS RELACIONAL

DAVID SANTIAGO PALACIO COLORADO

DANIEL RODRÍGUEZ GONZÁLEZ

UNIVERSIDAD PONTIFICIA BOLIVARIANA

ESCUELA DE INGENIERÍAS

FACULTAD DE INGENIERÍA INFORMÁTICA Y TELECOMUNICACIONES

MEDELLÍN

2013

DESARROLLO DE HERRAMIENTA DE SOFTWARE PARA DETERMINAR LA CALIDAD DE LOS DATOS DE UNA BASE DE DATOS RELACIONAL

DAVID SANTIAGO PALACIO COLORADO

DANIEL RODRÍGUEZ GONZÁLEZ

Trabajo de grado para optar al título de Ingeniero de Sistemas e Informática

Asesor

IVAN AMÓN URIBE

Magister en Ingeniería de Sistemas

UNIVERSIDAD PONTIFICIA BOLIVARIANA

ESCUELA DE INGENIERÍAS

FACULTAD DE INGENIERÍA INFORMÁTICA Y TELECOMUNICACIONES

MEDELLÍN

2013

NOTA DE ACEPTACIÓN

_____

_____

_____

_____

_____

Firma
Andrés Felipe Muñetón Lopera
Jurado

_____

Firma
Javier Emilio Sierra Carrillo
Jurado

_____

Firma
Iván Amón Uribe
Jurado

Medellín, 18 de Febrero de 2013

AGRADECIMIENTOS

Los autores de este trabajo quieren expresar su agradecimiento primeramente a nuestros padres por el enorme apoyo recibido durante las largas jornadas de estudio dedicadas a la adquisición de los conocimientos aquí materializados.

También queremos extender nuestro agradecimiento al director de la tesis, el ingeniero Iván Amón Uribe, quien nos acompañó y guio durante la ejecución de la misma.

Y por último agradecemos a la Universidad Pontificia Bolivariana que nos formó humana, académicamente y nos brindó a través de sus maestros los conocimientos que nos permitieron llevar a cabo este proyecto.

CONTENIDO

LISTA DE FIGURAS

LISTA DE TABLAS

LISTA DE DIAGRAMAS

GLOSARIO

Atributos: En bases de datos son el conjunto de propiedades de una entidad, esto está representado por las columnas de una tabla.

Bodega de datos: Concentración de información orientada a su análisis posterior para apoyar la toma de decisiones. Previo a su utilización deben aplicarse procesos de análisis, selección y transferencia de datos desde las fuentes originales.

Calidad de datos: Es el valor agregado que adquieren los datos al ser más confiables.

ETL: Extract, Transform and Load por sus siglas en inglés. Es un proceso usado especialmente en la construcción de bodegas de datos que implica la extracción de la fuente de datos, su transformación según las necesidades del negocio y su carga en el modelo de datos objetivo.

Herencia: Concepto de la programación orientada a objetos en el cual un clase deriva de otra extendiendo su funcionalidad.

Hosting: Es un servicio en el cual un tercero brinda la posibilidad de utilizar su hardware para que los usuarios ejecuten sus aplicaciones.

Ingeniería de software: Es la disciplina o área de la Ingeniería que ofrece métodos y técnicas para desarrollar y mantener software con el fin de acotar el riesgo de fracaso en la consecución del objetivo creativo.

Interfaces: Estructura que define un conjunto de métodos sin implementación a manera de protocolo de comportamiento para las clases que la implementen.

Linq To SQL: Es un componente de .NET framework el cual permite convertir una base de datos relacional en un modelo orientado a objetos, el cual es más cercano al lenguaje del desarrollador. Este componente se encarga de realizar las conversiones entre objetos y métodos a sentencias SQL y viceversa.

Linq To XML: interfaz que permite utilizar las funcionalidades de LINQ para el procesamiento en memoria de XML.

Patrón Factory: Patrón de diseño en el cual existe una clase encargada de retornar instancias de otros objetos.

Patrón Singleton: Patrón de diseño en el cual se garantiza que sólo existe una instancia y que puede ser accedida globalmente.

Perfilamiento: Es el proceso de examinar los datos en una base de datos para recopilar estadísticas e información sobre el estado de los mismos.

Polimorfismo: Es la capacidad de las clases hijas de sobrescribir el comportamiento del padre.

Registro huérfano: Registro de una base de datos que no referencia a otro registro donde la definición del modelo determina su obligatoriedad, violando así la integridad de los datos.

Registros padres sin hijos: Registro de una base de datos que no tiene asociado uno o más registros donde la definición del modelo determina su obligatoriedad, violando así la integridad de los datos.

Versionamiento: Gestión de las diferentes versiones generadas de un producto a medida que este se modifica, actualiza o mejora.

Taxonomía: Es el proceso de clasificar los elementos de un sistema por medio de una estructura jerárquica.

Tuplas: Es una secuencia ordenada de elementos. Para el caso de las bases de datos nos referimos a los registros de una tabla.

Wizard: También conocido como asistente de software o asistente de configuración. Es un tipo de interfaz de usuario que presenta una secuencia de cuadros de diálogo que lo llevan a través de una serie de pasos bien definidos con el fin de llevar a cabo una tarea. Las tareas complejas, con frecuencia son más fáciles de realizar a cabo mediante un asistente.

RESUMEN

En este trabajo se dará primeramente una explicación al lector sobre qué es la calidad de datos, algunas clasificaciones de los problemas de esta y de la importancia que tiene para las empresas durante la toma de decisiones y generación de estrategias.

Posteriormente se presenta una breve descripción de algunas aplicaciones existentes en el mercado, algunas de sus características y los tipos de errores que corrigen y se realiza una comparación entre las mismas.

Se plantea al lector el problema al que se pretende dar solución y el enfoque que se usó para lograrlo, cómo se llevó a cabo la elaboración de la aplicación y las pruebas realizadas para constatar el buen funcionamiento de la herramienta.

PALABRAS CLAVE: CALIDAD DE DATOS, PROBLEMAS DE CALIDAD DE DATOS, BASES DE DATOS, ESCALABILIDAD, MÓDULOS, PERFILAMIENTO DE DATOS, MÉTRICAS.

INTRODUCCIÓN

La calidad de los datos almacenados por las empresas constituye un factor fundamental para la toma de sus decisiones ya que puede afectar en gran medida la implementación de proyectos que dependen de este factor, por ejemplo, los proyectos de bodegas de datos o proyectos de inteligencia de negocio.

Es por esto que la calidad de datos debe ser monitoreada con el fin de tomar medidas correctivas y servir de estimador del esfuerzo a invertir en la tarea, labor que en bases de datos de gran tamaño puede ser dispendiosa y poco eficiente realizarla sin ayuda de una herramienta especializada. Es aquí donde entra en escena nuestra aplicación, con el fin de permitir a un usuario con un nivel intermedio de conocimiento en la materia realizar la configuración de un perfilamiento completo a múltiples bases de datos en busca de problemas de calidad a través de una interfaz amigable e intuitiva que lo guiará por los pasos requeridos, desde la conexión a la base de datos hasta la presentación de los resultados. De este modo el usuario sólo tendrá que sentarse y esperar para poder observar un reporte con los resultados del análisis para los campos, tablas o bases de datos seleccionados.

CAPÍTULO 1
OBJETIVOS

1.1. Objetivo general

Desarrollar una herramienta de software que determine la calidad de los datos de una base de datos relacional, que permita la detección de tres problemas de calidad de datos y elabore un perfil de múltiples atributos simultáneamente entregando resultados particulares y globales acerca de la base de datos.

1.2. Objetivos específicos

- Identificar los principales problemas en calidad de datos.
- Identificar metodologías existentes para determinar el estado de la calidad de los datos de una base de datos.
- Identificar algoritmos para detección de errores e inconsistencias en los datos.
- Realizar el análisis, diseño e implementación de un prototipo de software para determinar el estado de los datos en relación con al menos tres problemas de calidad de los datos.
- Probar la herramienta sobre una base de datos.

CAPÍTULO 2
MARCO TEÓRICO Y ESTADO DEL ARTE

2.1. Introducción

En la actualidad el manejo de la información es un tema de vital importancia para las empresas, el gobierno, las investigaciones y cualquier entidad que requiera el manejo de la misma. Sin embargo, ya no es suficiente tener un lugar para almacenar grandes volúmenes de datos, hoy en día se requiere su procesamiento para extraer patrones y generar información que permitan crear estrategias de mercadeo en las empresas, generar conocimiento en las investigaciones o políticas gubernamentales [1], es decir, los datos se convierten en una herramienta para la toma de decisiones.

2.2. Qué es calidad de los datos

Los datos a procesar adquieren un valor agregado, lo que impone como requisito que su contenido sea fiable y corresponda a la realidad. Este valor agregado se conoce como calidad [1], la cual genera un nivel de confianza sobre la información arrojada, permitiendo una mejor toma de decisiones y colaborando en el cumplimiento de los objetivos de las organizaciones que hacen uso de ella. El mejoramiento de dicho nivel implica la detección y corrección de los errores de calidad, sin embargo es engorroso definir qué es y qué no es un error ya que estos son altamente específicos de la aplicación [1], dificultando determinar a priori cuales problemas son principales y requieren mayor atención en las bases de datos del cliente.

2.3. Problemas en las bases de datos

Existen diferentes tipos de errores en las bases de datos y su clasificación es muy diversa ya que varía de autor en autor, sin embargo, es posible identificar ciertos

tipos comunes en las investigaciones de este tema. A continuación se mostrará una taxonomía que se considera como una de las más completas encontradas basada en [2].

2.3.1. Tipos de errores

Errores en una sola relación

Estos errores se dan en una tabla de la base de datos. En este se encuentran los siguientes errores:

- Errores en un sólo campo:
    - Campos vacíos
    - Transgresión de sintaxis
    - Valores desactualizados
    - Valor fuera del rango permitido
    - Valor no perteneciente al conjunto
    - Errores de ortografía
    - Valor no perteneciente al contexto
    - Sobrepaso del contexto
    - Valores sin sentido
    - Valores imprecisos que pueden adquirir un doble significado
    - Violación de las restricciones de un dominio
- Valores en una columna
    - Violación de clave primaria
    - Existencia de sinónimos
    - Violación de las restricciones de un dominio
- Valores de atributo de una tupla
    - Tupla semi-vacía
    - Inconsistencia entre valores de atributos
    - Violación de las restricciones de un dominio

- Valores de atributo en múltiples tuplas
    - Redundancia de una entidad
    - Inconsistencia entre la entidad
    - Violación de las restricciones de un dominio
- Interrelación entre múltiples relaciones
    - Violación de la integridad referencial
    - Referencias desactualizadas
    - Inconsistencia de sintaxis
    - Inconsistencia entre valores de atributos relacionados
    - Circularidad entre tulas en una relación
    - Violación de las restricciones de un dominio

Múltiples fuentes de datos

Estos tipos de errores se encuentran en múltiples bases de datos cuando éstas son analizadas como un todo.

- Inconsistencias de sintaxis
- Diferentes unidades de medición
- Inconsistencia en la representación
- Diferentes niveles de agregación
- Existencia de sinónimos
- Existencia de homónimos
- Redundancia sobre una entidad
- Inconsistencia sobre una entidad
- Violación de las restricciones de un dominio

2.4. Importancia de la calidad de los datos

Según Harrington [3], la calidad de los datos es importante ya que es necesario tener confianza de que lo que se obtiene de una base de datos es fiable. Las organizaciones toman decisiones tanto operacionales como estratégicas basadas

en lo que se tiene almacenado en las bases de datos. La calidad de estas decisiones está directamente relacionada con la calidad de los datos que subyace bajo estos.

La falta de conocimiento de la naturaleza de los datos puede acarrear sobrecostos o llevar a fracasar procesos de bodegaje de datos como el presentado en [4] donde una compañía diseñó su modelo de datos y sus scripts de extracción, transformación y carga (ETL por sus siglas en inglés) alrededor de una base de datos de 11.5 millones de clientes sólo para darse cuenta posteriormente que de hecho contenía únicamente 1.5 millones de registros de clientes distintos. En [5] se presenta una serie de casos reales donde la falta de la calidad de la información ha llevado a multimillonarias pérdidas que según cálculos del autor llegan a totalizar USD 44.589’450.000.

2.5. Perfilamiento de datos

Siguiendo esta línea es necesario realizar un estudio y análisis de los datos para determinar su calidad. Este proceso se llama perfilamiento de datos (*data profiling*), el cual consiste en el uso de una serie de algoritmos para el análisis y la evaluación de la calidad de un conjunto de datos [6].

Este análisis se puede realizar de dos formas: manual o automática. El método manual requiere que los administradores generen sentencias SQL para verificar así su calidad. El problema de este método es que requiere conocimientos técnicos y es muy dispendioso. Por lo tanto, se plantea el método automático, en el cual se usan aplicaciones que analizan la base de datos en su totalidad y generan reportes que ofrecen un mejor entendimiento del estado de los datos [4].

2.5.1. Características de una herramienta de perfilamiento

A continuación se mencionan algunos de los aspectos deseables en una herramienta de perfilamiento, según Eckerson [4].

- Fácil uso: esto es que la herramienta pueda ser usada por personas que no sean necesariamente expertos en bases de datos ni en estadística.
- Facilitar la colaboración: Generar reportes en diferentes formatos que sean accesibles a todos.
- Conectividad directa a fuente de datos: Esto garantiza que los datos con los cuales se está trabajando están actualizados.
- Copia de datos a un repositorio: Hacer una copia de los datos permite trabajar con mayor tranquilidad ya que no se verá impactado el rendimiento del sistema ni se verán afectados los datos al realizar operaciones complejas.
- Generar reglas de limpieza: los analistas deben poder crear reglas con la herramienta de perfilamiento, las cuales son pasadas a la herramienta de limpieza para su procesamiento, esto ayuda a que el tiempo empleado en el análisis sea más eficiente.
- Integración con herramientas de terceros: con el fin de automatizar la rutina de migración o automatización como la carga a bodegas de datos (data *warehouse*), las herramientas de perfilamiento y limpieza deben estar integradas y permitir su inicialización por parte de aplicativos de terceros.
- Ofrecer múltiples funcionalidades: brindar diferentes funcionalidades para analizar las estructuras de datos, generando estadísticas sobre los valores de una columna y mapeando las dependencias entre tablas.
- Precio: hoy en día se hace necesario ofrecer un conjunto de herramientas que realicen no sólo un proceso de análisis, sino también de limpieza y de ETL, para que el costo de la inversión sea más atractivo.

2.6. Qué es limpieza de datos

Teniendo claro los tipos de errores y la importancia de tener una base de datos fiable para la toma de decisiones, un paso posterior al diagnóstico consiste en la depuración de acuerdo con los objetivos de la compañía. Por lo tanto, la limpieza

de datos se encarga de “remover los errores e inconsistencias de los datos con el fin de mejorar la calidad de los mismos” [7]. Esta limpieza puede llevarse a cabo empleando diferentes técnicas para detectar errores, como las descritas en [8] donde también es posible encontrar fundamentos matemáticos para medir cuantitativamente el grado de limpieza de los datos.

2.7. Implicaciones de la limpieza de datos

- Requiere un experto en los datos para tomar decisiones acertadas al momento de decidir cómo corregir los errores.
- Se requiere mucho procesamiento para aplicar los algoritmos de limpieza y de identificación de errores.
- El costo de los programas encargados de este proceso generalmente es alto.
- El proceso de identificación y limpieza toma mucho tiempo.
- En muchas ocasiones no se pueden reparar todos los tipos de errores que se identificaron anteriormente ya que la mayoría de herramientas que existen sólo cubren un rango de problemas, por lo que una limpieza integral requeriría el uso de distintas herramientas. La tabla 3 muestra una comparación de algunas herramientas existentes y de los problemas que solucionan, tomada de [1].

	AJAX	FraQL	Potter's Wheel	ARKTOS	IntelliClean
Lexical Error					
Domain Format Error	User Defined	User Defined	Pattern learning	User Defined	User Defined
Irregularities	User Defined	User Defined			User Defined
Constraint Violation	Filter violating tuples			Three types of constraints	Alert and Update rule
Missing Values	User Defined	Statistical Values			Update rule
Missing Tuples					
Duplicates	Match, Cluster and Merge	Union, Join, and Reconciliation			Merge/Purge rule
Invalid Tuple		Statistical Methods			

Tabla 1 - Errores VS. Aplicación

2.8. Formas de corregir los errores

A continuación se muestran algunos de los métodos implementados para hacer limpieza de datos:

- Análisis (*Parsing*): Este consiste en identificar si el contenido de una cadena pertenece a una gramática G y encontrar posibles correcciones usando la distancia entre cadenas.

- Transformación de los datos (*Data Transformation*): pretende mapear los datos de su formato de origen hacia un formato esperado por la aplicación. Las transformaciones afectan tanto los esquemas de las tuplas como el dominio de sus valores. Los datos de diferentes fuentes son mapeados en un esquema común que se ajusta mejor a las necesidades de la aplicación [1].
- Garantizar el cumplimiento de las restricciones (*Integrity Constraint Enforcement*): este consiste en la identificación o corrección de posibles valores que al momento de hacer una eliminación, edición o borrado, puedan afectar la integridad referencial de la base de datos.
- Eliminación de duplicados (*Duplicate elimination*): Este método consiste en identificar cuando dos tuplas corresponden a la misma identidad.
- Métodos estadísticos: Analizan los datos usando valores de medias, desviaciones estándar, rangos, entre otros, para encontrar anomalías en los datos.

2.9. Aplicaciones

A continuación se hace una breve descripción de algunas aplicaciones existentes en el mercado, algunas de sus características y los tipos de errores que corrigen.

2.9.1. Data Match 2011 [9]

- Carece de un asistente (*wizard)* para todo el proceso, sólo para la inserción de la fuente de datos.
- Usan video tutoriales para la explicación del funcionamiento de la aplicación.
- Realiza análisis y limpieza por columnas.
- Opciones de limpieza:
    - Intercambio de mayúsculas por minúsculas y viceversa
    - Cambio a mayúsculas

  - Cambio a minúsculas
  - Inicio con mayúsculas y el resto con minúsculas Ej: Aeropuerto Internacional
  - Eliminar caracteres no imprimibles
  - Eliminar espacios al inicio y final del contenido
  - Eliminar caracteres en específico
  - Reemplazar caracteres en específico
  - Eliminar espacios entre palabras
  - Eliminar letras
  - Eliminar números
  - Reemplazar cero con la letra O
  - Reemplazar letra O con ceros
- Limpieza de emails.
- Definir tipo de campo (fecha, nombre, dirección, Zip, empresa, etc.) y auto detectar el tipo de campo.
- Word Smith: Permite modificar texto que aparece varias veces para estandarizarlo Ej: Inc aparece 20 veces, puede cambiarse por *Incorporated* y se actualizará en todos los registros que lo tengan, puede crear una columna donde se escriba la nueva palabra en los registros que tiene la antigua. Esto se realiza por columnas.

2.9.2. Data Cleaner [10]

- Permite conexiones a diferentes bases de datos instalando sus controladores:
  - Oracle
  - Postgress
  - MySQL
  - SQL Server
  - SQLite

  - Derby
  - Archivos de excel
  - Access
  - XML
- Posee diccionario al cual se le pueden agregar nuevas palabras para validaciones.
- Permite el uso de expresiones regulares para reglas de validación.
- Optimización de consultas.
- Posee una interfaz intuitiva, pero carece de asistente para todo el proceso de análisis.
- Es *OpenSource.*
- Carece de documentación detallada acerca de todas las funcionalidades.
- Permite hacer análisis de:
  - Valores booleanos
  - Fechas
  - Buscar coincidencias según patrones y diccionarios
  - Valores numéricos
  - Busca coincidencias entre valores fonéticamente similares
  - Cadenas, proporcionando diferentes estadísticas de las mismas como cantidad de números, espacios en blanco, cantidad de palabras, etc.
  - Distribución de los valores
  - Distribución de días de la semana en los campos de fecha

2.9.3. Datiris [11]

- Permite hacer perfilamiento a bases de datos:
  - Oracle
  - SQL Server
  - MS Access

  - Excel
  - Archivos de texto
  - FoxPro
  - ODBC
- Permite crear reglas de dominio para validar los datos.
- Realiza análisis de patrones para identificar problemas de calidad en grandes cantidades de datos.
- Permite perfilamiento en *batch* para automatizar el análisis.
- Interfaz en línea de comandos.
- Permite especificar filtros para hacer un perfilamiento condicional, es decir a un conjunto de datos específico.
- Permite guardar la configuración de los perfilamientos que se hacen para uso posterior.
- Permite exportar resultados a hojas de cálculo de Excel.
- Tags inteligentes: Permite centralizar los resultados de los análisis para compartirlos con todo el equipo de trabajo.
- Sistema de seguridad para controlar el acceso a la información que contiene la aplicación.
- Interfaz intuitiva.
- Carece de asistente.

2.9.4. DbGlobal [12]

Esta aplicación provee sensibilidad a mayúsculas y minúsculas en los nombres, la determinación de género, manejo de *null,* manejo de cadenas vacías, concordancia de índices similares, entre otras.

2.9.5. MatchIT v5 Data Quality Suite [13]

Esta herramienta cubre una amplia gama de errores e inconsistencias en los datos, tales como:

- Mala digitación y variaciones de abreviación de los datos.
- Previene la adición de registros duplicados.
- Remueve registros duplicados.
- Encuentra concordancias entre sistemas.
- Corrige mayúsculas y minúsculas en nombres y direcciones.
- Mejora la estructura y reacomoda los datos para corregir los campos.
- Detecta el género de los nombres de pila.
- Escanea por basura y entradas ofensivas.
- Concordancia fonética, mala digitación y variaciones de abreviaciones.
- Estandariza los títulos y calificadores (qualifications).
- Inteligentemente divide los nombres entre las partes que lo componen.

2.9.6. Trillium [14]

Permite la detección y corrección de errores en los datos, utilizada por grandes compañías como Fedex [8]. Esta aplicación es capaz de analizar los campos en busca de los siguientes errores:

- Corrección de escritura
- Datos incorrectos
- Datos faltantes
- Anomalías en los datos
- Datos mal clasificados
- Valores Nulos

2.9.7. Comparación de las aplicaciones

La Tabla 4 resume los tipos de errores que las diferentes aplicaciones pueden trabajar.

	Aplicación					
Tipo de aplicación	Perfilamiento Limpieza	Perfilamiento Limpieza	Perfilamiento Limpieza	Perfilamiento Limpieza	Perfilamiento	Limpieza
Tipos de errores	Trillium	Data Match 2011	Data Cleaner	MatchIT v5 Data Quality Suite	Datiris	DbQuality
Errores léxicos	x			x		x
Errores de formato de dominio	x	x	x	x	X	x
Irregularidades	x	x	x	x		
Violación de restricciones	x		x	x	X	
Valores faltantes	x	x		x	X	
Tuplas faltantes						
Duplicados	x	x		x		
Tuplas inválidas	x					

Tabla 2: Aplicaciones vs Tipos de errores

2.10. Antecedentes

Como trabajos previos mediante los cuales se maduró la necesidad de este proyecto, se encuentran:

1. Tesis de Maestría: Guía metodológica para la selección de técnicas de depuración de datos (Autor: Iván Amón Uribe) [15].
2. Tesis de pregrado Ingeniería Informática UPB: Diagnóstico de la calidad de la base de datos de la Clínica Universitaria Bolivariana (Autores: James Paniagua y Juan Felipe Mira) [16].
3. Tesis de pregrado Ingeniería Informática UPB: Caracterización de técnicas para detección de strings duplicados (Autor: Ricardo Patiño) [17].
4. Tesis de pregrado Ingeniería Informática UPB: Caracterización de técnicas para detección de valores faltantes (Autor: Carolina Muñoz) [18].

5. Tesis de pregrado Ingeniería Informática UPB: Caracterización de técnicas para detección de valores extremos (Autor: Christine Michele) [19].
6. Tesis de pregrado Ingeniería Informática UPB: Algoritmo fonético para detección de cadenas de texto duplicadas (Autor: Juan David Zuluaga) [20].
7. Tesis de pregrado Ingeniería Informática UPB: Caracterización de metodologías para limpieza de datos (Autor: Juan David Velásquez) [21].

Estos trabajos han dado lugar a dos publicaciones nacionales [22] [15], una ponencia internacional [23] y dos ponencias nacionales [22] [15].

El trabajo a realizar en este proyecto de grado es la continuación de los esfuerzos realizados en la facultad en la temática de calidad de datos. Así, varias de las tesis de grado mencionadas anteriormente [15], [17], [18], [19], establecieron caracterizaciones y metodologías que servirán de insumo para diagnosticar correctamente algunos problemas comunes de los datos. De otra parte, el trabajo realizado en [21], servirá para determinar la metodología general del proceso a seguir para realizar el diagnóstico de las bases de datos. Por último, la experiencia real de determinar el estado de los datos de la Clínica Bolivariana, permitió constatar la dificultad de realizar un proceso real de diagnóstico a una base de datos con las herramientas existentes y la necesidad de contar con conocimientos en estadística y limpieza de datos.

Contar con una herramienta poderosa para establecer rápidamente la situación de una base de datos cualquiera, es un aporte valioso que permitirá realizar trabajos de consultoría organizacional en este aspecto. Cabe anotar que este diagnóstico es de gran importancia para empresas que estén pensando en proyectos de Inteligencia de Negocios (*Business Intelligence*) ya que la calidad real de los datos, determina en buena medida la duración total del proyecto.

CAPÍTULO 3
DESCRIPCIÓN DEL PROBLEMA

Comúnmente, la depuración de los datos de una base de datos es llevada a cabo por personal del área de tecnología informática como DBAs (administradores de bases de datos) o analistas, quienes con frecuencia no están preparados para realizar esa labor. Saber qué pasos seguir para determinar la calidad de los datos de una base de datos, unido a la variedad de posibles problemas de calidad, la multitud de técnicas existentes para cada uno de ellos y las condiciones que deben cumplirse para que las técnicas puedan aplicarse exitosamente, hacen que diagnosticar la situación de calidad de los datos de una base de datos sea una tarea difícil que, para realizarla exitosamente, requiere conocimientos tanto estadísticos como de metodologías y técnicas para limpieza de datos.

En el mercado existen herramientas de software, tanto estadísticas como de perfilamiento de datos (*Data profiling*), con diferentes tipos de licenciamiento que determinan la calidad de los datos almacenados en una base de datos. En general, estas herramientas adolecen de lo siguiente:

- No incorporan una metodología que oriente al usuario para seguir un proceso ordenado en el diagnóstico.
- La selección de la técnica durante la elaboración del perfil de la calidad de los datos requiere tener en cuenta la naturaleza de los mismos, es decir, las características específicas que presentan los datos a evaluar, ya que una técnica incorrecta puede no encontrar ningún problema o encontrar más de los existentes. Esto hace que sea necesario determinar la naturaleza de los datos para seleccionar una técnica acorde. A modo de ejemplo, para realizar una detección de cadenas de texto duplicadas, si los datos presentan problemas como abreviaturas (Juan Alberto López vs Juan A

López), la técnica más adecuada es Brecha Afín. Si en este caso se utiliza Distancia de Edición, las diferencias entre las cadenas son muy grandes y no detectará que se trata de un duplicado de la cadena en cuestión.

- El análisis consolidado del estado de la base de datos es difícil de realizar ya que las herramientas entregan resultados atributo por atributo. Aunque tratar los atributos individualmente es necesario para poder tomar acción sobre los problemas encontrados, es altamente deseable poder contar con informes resumidos del estado general de la base de datos para darse una idea del esfuerzo que conllevará su limpieza y los efectos sobre un posible proyecto de Inteligencia de Negocios. Por ejemplo, no se tiene conocimiento de alguna herramienta que informe el porcentaje total de datos con problemas en una base de datos lo cual sería un indicador útil.

Se pretende entonces, realizar una investigación aplicada cuyos resultados sean plasmados en una herramienta de software que supere las dificultades enunciadas. Esta aplicación tendrá la finalidad de determinar el estado de una base de datos en cuanto a calidad de datos se refiere, esto es, se construirá un software que realice en forma semiautomática la detección de los posibles problemas de calidad de los datos incorporando una metodología de evaluación, elaborando el perfil de múltiples campos simultáneamente sin necesidad de contar con expertos en estadística o en calidad de datos y facilitando el análisis global de los resultados.

CAPÍTULO 4
METODOLOGÍA DE PERFILAMIENTO

Para el análisis de la calidad y la limpieza de datos es recomendable la implementación de una metodología que sirva las veces de guía y proceso racional para la evaluación y corrección de los errores en los datos [24]. Por tal motivo este trabajo se apoya en una metodología existente para que el análisis realizado brinde información útil al usuario, la cual podrá utilizar en un proceso integral de mejoramiento y retroalimentación.

Dentro de la bibliografía se pudo encontrar varias metodologías referentes al proceso de evaluación y mejoramiento de la calidad de datos, de las cuales se eligió una como base ya que se podía aplicar más fácilmente al enfoque de nuestro trabajo, las demás metodologías se descartaron ya que no encontró el detalle esperado, no compartían nuestro enfoque o atacaban los problemas de calidad de datos desde ámbitos diferentes al nuestro.

A continuación describiremos las metodologías propuestas en los artículos leídos y la razón por la cual no se seleccionaron.

4.1. A Data Quality Methodology for Heterogeneous Data [24]

Este artículo propone una metodología que consiste en 3 fases:

1. Reconstruir las unidades organizacionales, procesos, recursos y entidades conceptuales relacionadas con la organización. Una vez se han identificado los recursos relacionados con los problemas en los datos, se pasa a abstraerlos en entidades conceptuales y a relacionarlos entre sí. De igual forma esta fase tiene como tarea la identificación cualitativa de los principales problemas de la base de datos, para luego darles una evaluación cuantitativa.

2. La segunda fase se encarga de evaluar y obtener calificaciones cuantitativas de los problemas de una base de datos. Esta fase se divide en dos pasos:
    a. Ranking de los recursos: en esta fase se usan los conocimientos de los expertos para identificar qué recursos tendrían mayor impacto en la mejora de la calidad de los datos. Se asignan pesos entre 0 y 1.
    b. Medición de la calidad de datos: Se eligen dimensiones de calidad de datos y sus métricas correspondientes y se aplican a los recursos seleccionados.
3. En esta última fase se busca implementar actividades de mejoramiento, esta fase consta de 3 pasos:
    a. Definir los valores que se desean alcanzar en el proceso de mejoramiento.
    b. Seleccionar las actividades candidatas para el proceso de mejoramiento.
    c. Seleccionar las actividades que puedan mejorar cada una de las dimensiones medidas en la fase 2.

Este artículo también propone la medición de dos métricas:

- La precisión: esta se refiere a la cercanía que hay entre dos valores, en nuestro caso la cercanía que tiene un valor A, a su representación en la vida real dada por un valor B.
- Razón entre que tan pronta es actualizada la información respecto al tiempo que necesita el usuario según sus necesidades o las restricciones del dominio.

La metodología propuesta por estos autores es válida para nuestro propósito, sin embargo la descartamos al encontrar una metodología más detallada, que tiene embebido los pasos descritos anteriormente.

4.2. Data Quality Assessment [25]

Este documento propone la implementación de 3 fases para la evaluación de la calidad de datos, los cuales son:

1. Realizar una evaluación objetiva y subjetiva de la calidad de los datos.
2. Comparar los resultados de la evaluación identificando discrepancias e identificando las causas principales de las mismas.
3. Determinar y tomar las acciones necesarias para mejorar.

De igual forma describe un conjunto de métricas para aplicar a los datos, entre las cuales están:

- Accesibilidad
- Cantidad adecuada de datos
- Credibilidad
- Completitud
- Representación concisa
- Fácil de manipular
- Libre de error
- Interpretabilidad (idioma, unidades y símbolos correctos)
- Objetividad
- Relevancia
- Reputación
- Seguridad
- Oportuna
- Entendible
- Valor agregado

Los autores proponen tres formas funcionales para realizar las medidas estas son:

- Taza simple: Mide la razón de las salidas deseadas respecto al total de salidas.

- Operadores Max o Min: es útil cuando se manejan dimensiones que requieren de múltiples variables.
- Promedio ponderado: para casos donde se manejan varias variables, es una alternativa al operador Min.

La propuesta de los autores se descartó ya que no tiene gran profundidad en el proceso de perfilamiento de datos, y los pasos que describen están contenidos en otras metodologías más detalladas.

4.3. El Procedimiento de Diagnóstico de la Calidad de los Datos con Enfoque de Riesgos [26]

La propuesta del autor es priorizar el análisis de los datos según una jerarquización respecto al riesgo e impacto que tienen algunos datos en el negocio. Esta propone los siguientes pasos para el proceso de medición y mejoramiento de calidad de datos:

1. Selección y ordenamiento de datos críticos.
2. Selección de las dimensiones e indicadores por medio de una metodología Delphi.
3. Se determina la cantidad de errores en cada dato, se clasifican los errores y se determina cual error es más grave.0
4. En esta etapa se analizan los problemas y se les da una prioridad, se buscan las causas de los problemas y las posibles soluciones.
5. Esta etapa consiste en monitorear constantemente la calidad de los datos, desde el momento en que se capturan, se procesan y salen.
6. Esta etapa es transversal a todo el proceso y consiste en capacitar a todo el personal para que comprenda la importancia de la calidad en los datos.

Este documento presenta una metodología muy similar a la seleccionada ya que es elaborada por el mismo autor. Como se mostrará más adelante, se tomarán algunos procedimientos descritos en los pasos de este método ya que apoyan y complementan la metodología seleccionada.

4.4. Measuring Data Quality [27]

Propone a GQM, un *Framework* para definir y usar métricas de medición de la calidad de los datos mediante los siguientes pasos:

- Especificar metas
- Caracterizarlas por significado apuntando a los atributos relevantes
- Dar medidas que tal vez resuelvan el problema

En este marco de trabajo los objetivos están definidos para objetos con un propósito desde una perspectiva en un ambiente. Cuando se sigue GQM para definir métricas se han identificado dos objetos principales a ser medidos:

- Conjunto de datos: datos actualmente guardados en la base de datos
- Modelo de datos: como están estructurados los datos, desde la implementación o desde un punto de vista lógico

Cada objetivo es definido en un ambiente en particular, cuyos elementos pueden ser

- Un conjunto predeterminado de datos
- Un modelo predeterminado
- Un conjunto de atributos temporales: tienen que ser actualizados para ser precisos. Es necesario identificar estos subconjuntos y darles un procedimiento para calcular o estimar cuando los datos están desactualizados.

Definición de métricas

Primero se introducen múltiples objetivos identificados para una dimensión en particular. Segundo se incluyeren preguntas y métricas para cada objetivo y tercero se brinda una técnica operacional para calcular el valor asociado. En algunos casos no hay una manera operacional de medir ítems específicos o no se ha identificado uno todavía o el identificado es muy complejo de usar, en tales casos, la técnica debe ser redefinida.

4.5. Daquincis architecture [28]

La presente metodología está enmarcada en ambientes empresariales que pueden llegar a tener múltiples fuentes de datos para albergar información similar. Esta metodología presenta una arquitectura peer2peer diseñada para compartir los datos de mejor calidad mediante la implementación de un motor publicador y suscriptor.

El componente fundamental de dicha arquitectura es el Data Quality Broker, el cual funciona como un servicio distribuido peer-to-peer realizando dos funciones principalmente: procesamiento de consultas y mejoramiento de la calidad.

El Data Quality Broker notifica a las organizaciones con datos de baja calidad sobre la disponibilidad de unos de mayor calidad para que esta decida si actualiza o no sus datos actuales con los nuevos valores. Este componente consulta las diferentes fuentes de datos y compara los resultados obtenidos con el fin de retornar únicamente los datos de mejor calidad

Relevancia y pertinencia de la medición

En el documento también se aborda la relevancia y pertinencia de una medición, ya que al medir un factor de calidad para un atributo pueden darse las siguientes situaciones:

- “Resulta interesante medir dicho factor de calidad sobre dicho atributo, para lo cual es necesario diseñar un procedimiento”.
- “Resulta interesante medir dicho factor de calidad sobre dicho atributo pero no es necesario utilizar un procedimiento de medición, dado que es posible deducir las medidas a partir de los metadatos de las bases fuente. Por ejemplo, sabemos que ciertos atributos, por construcción, siempre van a cumplir ciertas restricciones de formato”.

- “No resulta interesante medir dicho factor de calidad sobre dicho atributo. Esto puede ser, por ejemplo, porque constituye un caso análogo a otros”.
- No tiene sentido medir dicho factor de calidad sobre dicho atributo. Por ejemplo, un identificador autogenerado que no tiene ningún significado para los usuarios, por lo que no tiene sentido medir si los valores de dicho atributo corresponden con la realidad.

4.6. Metodología seleccionada

A continuación se describe la metodología propuesta en [29], la cual está enfocada a compañías productoras o de servicios, dándole un carácter muy general.

Procedimiento para el diagnóstico de la calidad de los datos

El proceso de diagnóstico puede ser definido como: “el proceso de estudio de los síntomas, especulación sobre las causas, ensayo de las teorías y descubrimiento de las causas.” Basado en dichos conceptos se proponen los siguientes pasos:

1. Encontrar la causa de los síntomas
2. Encontrar la cura a las causas

Etapa 1. Identificación y ordenamiento de los datos críticos de clientes

Ya que existen un gran volumen de datos en las organizaciones es necesario identificar qué información tiene mayor impacto negativo en las operaciones del negocio (datos críticos). Por el volumen de los datos que se maneja esta tarea puede ser difícil por esta razón se divide en dos pasos:

1. *Identificación a partir de los clientes*

En esta etapa se hacen encuestas a los clientes para identificar, según la percepción de ellos, qué datos tienen errores. De ser necesario se puede obviar este paso, ya que no todos los clientes tienen acceso a información de la empresa, por lo tanto sólo se debe aplicar en casos donde estos acceden a servicios que se pueden relacionar fácilmente con una fuente de datos.

2. *Identificación a partir de los especialistas*

En esta etapa se toma en cuenta la opinión de los especialistas en la organización, los cuales tendrán la tarea de aportar otros datos críticos y darle un orden al listado de datos proporcionados por ellos y por los clientes.

Ordenamiento de los datos críticos.

A continuación proponemos una forma de ordenar los datos según su criticidad, este procedimiento es propuesto en [26] y consideramos apoya este procedimiento.

a. Una vez seleccionado el listado de datos críticos, será trabajo de los especialistas crear una matriz de clasificación de errores, esta matriz permitirá determinar la función de ponderación del dato y el índice de gravedad. En esta matriz se ubica cada tipo de dato [D] según la gravedad que representa la presencia de un error en el mismo como se muestra en la Tabla 1, donde [D1], [D2], [D3] y [D4] son los tipos de datos. Cada dato se le da una calificación entre Grave y Menos Grave por cada impacto.

Gravedad / Impacto	Error Grave	Error de Gravedad Media	Error Menos Grave
1-Funcionalidad del dato	[D1] [D3]	[D2]	[D4]
2-Impacto Económico	[D1]	[D3] [D4]	[D2]
3-Afecta la salud	[D3]	[D2]	[D1] [D4]
4-Toma Decisiones	[D1]	[D2]	[D3] [D4]
5-Dependencia de otros datos	[D2]	[D1]	[D3] [D4]
6-Quejas y reclamaciones	[D4]	[D1]	[D2] [D3]

Tabla 3

b. Según la anterior clasificación calculamos la función de ponderación la cual está dada por:

$$f(d) = c * p$$

Dónde $f(d)$: Es la función de ponderación del dato.

c: Es la cantidad de veces que cada tipo de dato se ubica en la clasificación de error.

p: El peso en cada clasificación de error, este es 1 para menos grave, 2 para medianamente grave y 3 para grave.

Según la tabla anterior obtenemos los siguientes resultados:

Dato No.	Error Grave	Error de Gravedad Media	Error Menos Grave
	(c) x (3) =f (d)	(c) x (2) =f (d)	(c) x (1) =f (d)
[D1]	(3) x (3) = 9	(2) x (2) = 4	(1) x (1) = 1
[D2]	(1) x (3) = 3	(3) x (2) = 6	(2) x (1) = 2
[D3]	(2) x (3) = 6	(1) x (2) = 2	(3) x (1) = 3
[D4]	(1) x (3) = 3	(1) x (2) = 2	(4) x (1) = 4

Tabla 4

La clasificación final del dato está dada por el mayor valor en la función, por ejemplo el dato 4 tendrá una clasificación menos grave.

c. Según la clasificación anterior asignaremos un índice de gravedad (IG), este índice estará dado por la siguiente tabla:

Elemento	Clasificación	Puntuación
Gravedad	Grave (A)	10
	Gravedad Media (M)	5
	Menos Grave (B)	1

Tabla 5

Por ejemplo el IG del dato 4, el cual es Menos Grave, sería 1.

d. Se asigna un índice de frecuencia, este indica que tan probable es la aparición de errores en los datos, este valor se calcula por la experiencia que se tenga, o por información disponible que permita definirlo. Este índice está dado por los siguientes valores:

Elemento	Clasificación	Puntuación
Frecuencia	Alta (A)	10
	Media (M)	5
	Baja (B)	1

Tabla 6

e. Para ordenar y seleccionar el dato según el riesgo que represente un error en este se calcula el Índice de Riesgo para el Dato (IRD), este valor es el producto del IG y el IF, se ordenan los datos según el IRC y se seleccionan los de mayor riesgo (con IRC igual a 100 o 50). Según el ejemplo desarrollado los IRD de los datos serían los siguientes:

Tipo de dato	Gravedad (IG)	Frecuencia (IF)	IRD = IG x IF
[D1]	10	10	100
[D3]	10	5	50
[D2]	5	5	25
[D4]	1	1	1

Tabla 7

f. Por último el autor propone mostrar los datos en un Mapa de Riesgos de Datos, esta herramienta permite visualizar en un conjunto de regiones dadas por los IF e IG, que datos son más riesgosos.

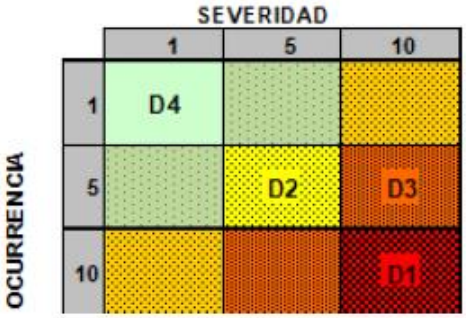

Tabla 8

Donde el extremo superior izquierdo contendría los datos menos riesgosos y el extremo inferior derecho los más riesgosos y por ende, los cuales se debe priorizar.

Etapa 2. Selección del tipo de datos crítico a diagnosticar según el listado ordenado

Esta etapa consiste en elegir el dato con mayor índice de criticidad, lo cual se obtiene en el ordenamiento realizado por los expertos en el paso anterior.

Etapa 3. Definir un responsable de calidad del dato objeto de estudio

Durante todo el proceso existirá una persona encargada de hacer cumplir el conjunto de actividades relacionadas con el proceso de diagnóstico y mejora de la calidad de los datos, esta persona también tendrá la responsabilidad de realizar los cálculos de los indicadores.

Etapa 4. Analizar las dimensiones de calidad del Dato

Para identificar si un dato es de calidad se requiere su comparación con algún criterio o dimensión, esta etapa se encarga de identificar estos parámetros para garantizar que el dato sea íntegro, consistente, puntual, exclusivo, válido. Se definen dos pasos:

1. *Determinar las dimensiones de calidad asociadas al Dato*

Como se mencionaba anteriormente es necesario el uso de criterios y dimensiones de calidad de datos para medir y analizar la calidad de los datos. Por este motivo esta fase se encarga de identificar las dimensiones que más se acomoden al tipo de dato que se haya seleccionado en los pasos anteriores. Es de resaltar que en esta fase se consideró como un complemento adecuado definir una granularidad de los datos de manera similar a lo enunciado en el documento de [30]. Según dichos autores “Llamamos granularidad de la medición a la unidad básica de información a la que asociamos las medidas. La granularidad puede ser: fuente, conjunto de tablas, tabla, túpla, atributo o celda. Por ejemplo si estamos manejando granularidad tabla, las medidas que obtenemos están asociadas a la tabla entera (no a elementos contenidos en la tabla) y si manejamos granularidad

celda, las medidas que se obtienen corresponden a celdas concretas (un valor de un atributo de una tabla)". Otro concepto interesante del documento es la agregación de medidas: "Le llamamos agregación de las medidas al pasaje de una granularidad a otra granularidad mayor, es decir con menor nivel de detalle. Por ejemplo, realizamos una agregación cuando, a partir de medidas a nivel de las túplas de una tabla, calculamos medidas globales para dicha tabla".

Uno de los puntos en el que se debe tener cuidado es que en esta etapa se corre el riesgo de estar dirigiendo la atención hacia características o dimensiones que no necesariamente son aquellas que están entre las preferidas de los clientes. Es importante interpretar adecuadamente la información y que los indicadores sean comunicados de forma rápida y efectiva y que sean de fácil compresión.

Dado que hay tantas dimensiones en la literatura, algunas de ellas comunes a todos los datos, por ejemplo, la exactitud para garantizar la confianza en el servicio, el autor propone la siguiente forma para seleccionarlas:

"Ante la cantidad de dimensiones que existen y que referencia la literatura, se determinó utilizar la decisión multicriterio para de esta forma poder elegir las más relevantes y que aportarán mejor información a la investigación. Para esto se propone la utilización del software Decisión, el cual a partir de las alternativas y criterios selecciona cuál de las primeras es mejor según el peso que se le dé a los criterios y el peso que se le otorgue a la relación alternativa-criterio.

Para la selección de los criterios y los pesos a otorgar a cada uno se proponen algunas técnicas, como la tormenta de ideas con la participación de especialistas y personal calificado. Realmente las bases del método Electre que se ha propuesto, establecen que se debe realizar un método Delphi para de esta forma establecer los criterios y los pesos, pero cuando hay pocos especialistas, y sobre todo en etapas iniciales de estudio, esto puede obviarse.

A partir de la obtención de la función de utilidad se obtienen las mejores alternativas. Los resultados dados por la función de utilidad. Se propone también el uso de los ordenamientos y la matriz de dominancia como técnicas para seleccionar las mejores alternativas."

Dimensiones

Dentro del conjunto de dimensiones existentes a continuación se hablará sobre algunas de las más mencionadas en los documentos sobre evaluación de calidad de datos.

- *Precisión*: La precisión es la cercanía que hay entre un valor X y un valor Y, donde Y es un valor que representa un fenómeno real, y X un valor que trata de representarlo. Según [24] esta dimensión pude ser medido de la siguiente manera:

$$(1)\ Acc(t) = \sum_{i=1}^{|t|} \frac{acc(r_i, D(r_i))}{|t|}$$

$$(2)\ acc(r_i, D(r_i)) = \begin{cases} 1 & r_i \in D(r_i) \\ 1 - NED(r_i, D(r_i)) & en\ caso\ contrario \end{cases}$$

Dónde:

r_i: $es\ el\ i-esimo\ valor\ de\ la\ tupla\ t$
$|t|$: $es\ el\ número\ de\ valores\ de\ la\ tupla$

La función acc() retorna 1 si el valor $r_{i,}$ es igual a el valor más cercano del dominio D, en caso contrario se retorna la distancia de edición normalizada entre el valor r_i y el valor más cercano en el dominio D.

- *Completitud:* según [31] esta dimensión indica si los atributos de un registro deberían tener valores. Estas reglas de completitud pueden ser asignadas en tres niveles de restricciones:
    - Atributos cuyos valores son obligatorios

  - Atributos opcionales que pueden tomar valores bajo ciertas condiciones
  - Atributos que no deberían tener ningún valor.

- Razonabilidad: Esta métrica pretende definir si un atributo está dentro de los rangos o valores permitidos que puede tomar según los valores existentes [31].

- Integridad Referencial: para una mejor administración de la información se hace necesario la identificación de los registros de manera única y esto a su vez conlleva a que otros registros al referenciar estos objetos por medio de las llaves foráneas puedan garantizar su existencia.

2. *Identificar los indicadores asociados a las dimensiones de calidad del dato y determinación de la frecuencia de medición de cada indicador*

Una vez seleccionadas las dimensiones se hace necesario elegir los indicadores encargados de medir ya sea cuantitativamente o cualitativamente las mismas. Este punto es neurálgico para el proceso y la toma de estas medidas debe realizarse en el momento oportuno, es decir que no interrumpan el proceso pero que reflejen la realidad del mismo.

Etapa 5. Realizar las mediciones necesarias para analizar las dimensiones de calidad del Dato

En esta fase se realizan las mediciones de la calidad actual de los datos, utilizando las métricas anteriormente seleccionadas. Este paso nos ayudará a detectar donde se encuentran los mayores problemas y a priorizar los esfuerzos.

1. *Localizar la información asociada al dato y toma de muestras*

Localizar una fuente de información precisa asociada al dato. Se recomienda tener una única fuente de información.

Para tomar las muestras aleatorias primeramente se debe calcular el tamaño de la muestra.

Se deben realizar comprobaciones de errores evidentes a partir de criterios estandarizados.

“Después de esta primera revisión, donde se anotarán todas las ambigüedades e incongruencias que aparezcan, se procederá a utilizar un muestreo aleatorio simple, con el tamaño de muestra anteriormente calculado. Cuando se encuentren identificados los clientes a los que se les realizarán las verificaciones de sus datos, se harán estas según se decida, anotándose todos los errores que aparezcan.”

2. *Calcular los indicadores y detectar problemas*

Una vez se tienen anotados todos los errores resultantes de la subetapa anterior se realiza el cálculo de los indicadores seleccionados. Estos indicadores deben ser tales que permitan reflejar la situación existente, realizar las comparaciones necesarias y tomar las acciones correspondientes. Se deben reflejar todos los problemas detectados.

Etapa 6. Procesamiento de los indicadores de calidad asociados al dato

Análisis de los indicadores para la detección de errores.

1. ***Elegir las herramientas apropiadas para el análisis de los indicadores (herramientas de análisis y procesamiento, herramientas estadísticas y de gestión)***

Se pueden emplear herramientas sencillas como:

- Histogramas
- Gráfico Pareto
- Gráfica de comportamiento

Herramientas complejas como:

- Diagramas de distribución
- Diagramas de interrelaciones

2. *Desarrollo de la(s) herramienta(s) seleccionada(s)*

Se debe ir analizando la efectividad de las herramientas utilizadas para, llegado el caso, pasar a emplear otras que brinden una mejor compresión de la situación.

Etapa 7. Análisis de los problemas de calidad del dato

Se realiza una jerarquización de los problemas mediante las mediciones de los indicadores para determinar el impacto que tiene cada uno de ellos en el proceso basándose en las etapas cuarta (detección de problemas) y la sexta (análisis de los indicadores). Se recomienda un diagrama Pareto para determinar el peso de los problemas.

El peso de los problemas puede definirse basándose en la frecuencia de ocurrencia o el impacto en la empresa.

Esta etapa es fundamental antes de pasar a la etapa ocho (8) donde se detectarán las causas. “Para esto deben encontrarse presentes representaciones de todas las áreas, para en equipo determinar las áreas asociadas a cada problema.”

Etapa 8. Diagnóstico de la calidad del dato

Los problemas podrán ser detectados gracias a los indicadores. Se recomienda la utilización del diagrama *causa-efecto* donde las causas de los problemas se desglosan haciendo más evidentes sus causas (raíces).

Si no se detectan todas las causas de los problemas éstos nunca serán eliminados totalmente o incluso de manera parcial. Luego de detectar las posibles causas con ayuda de los jefes de departamento interesados por medio de una

lluvia de ideas se procede a trazar los planes de acción encaminados a mejorar los indicadores.

Etapa 9. Monitoreo de los indicadores por parte de la gerencia

Es de gran importancia la revisión periódica de los indicadores de la calidad de los datos para una mejora continua. Esta etapa permitirá revisar los indicadores y definir si se deben seguir evaluando, o si se deben eliminar o modificar. De igual forma se debe crear planes preventivos de los problemas encontrados en esta revisión y llevar un registro y documentación de los mismos.

Etapa 10. Garantizar la seguridad del dato

Los datos de una empresa tienen demasiado valor, por lo tanto es necesario protegerlos de pérdidas o utilización indebida.

1. *El nivel de seguridad en que se encuentra el dato*

Se establecen tres niveles de seguridad en función del tipo de datos personales y en consecuencia las medidas de seguridad a emplear.

Nivel básico: obliga a las empresas a disponer de un documento de seguridad, registrar las incidencias, hacer que los usuarios se identifiquen y autentifiquen, controlar el acceso a los sistemas donde se almacenan los datos, etc.

Nivel medio: incluye las medidas contempladas en el nivel básico, además de contar con un responsable de seguridad, realizar auditorías de seguridad cada dos (2) años, implantar medidas adicionales de autentificación e identificación y controlar el acceso físico a los datos.

Nivel alto: incluye las medidas contempladas en el nivel básico y medio. Se debe tener seguridad en la distribución de soportes informáticos, un registro en los accesos, medidas adicionales en copias de respaldo en un lugar diferente del equipo en el que se tratan los datos y cifrado de datos para la transmisión de datos mediante redes de telecomunicaciones. Además de esto, se debe emitir un informe mensual por parte del responsable de seguridad sobre las revisiones realizadas y los problemas encontrados.

2. *Elaborar el manual de seguridad del dato*

En esta subetapa se debe:

- Redactar un documento de seguridad de obligatorio cumplimiento para el personal con acceso a los datos automatizados.
- Redactar unas cláusulas de protección de datos.
- Realizar auditorías cada cierto tiempo para comprobar la seguridad de los datos.
- Llevar a cabo las medidas precisas de seguridad y de tipo técnico y organizativo para garantizar la seguridad de los datos.
- Elaborar contratos, formularios y cláusulas para la recogida de datos, los tratamientos por parte de terceros y las sesiones o comunicaciones de los datos.

Todo el procedimiento anteriormente descrito es cíclico y una vez que sea empleado para el dato considerado más crítico se procederá a aplicársele al próximo dato crítico.

De igual forma durante todo el proceso se debe realizar capacitación de todo el personal, ya que no sólo los investigadores deben estar involucrados en el proceso de evaluación y mejoramiento de la calidad de los datos, sino también todos los trabajadores, ya que es un proceso que beneficiará a todos porque una buena calidad de datos implica un mejor desempeño y una disminución del tiempo de trabajo.

CAPÍTULO 5
METODOLOGÍA DE TRABAJO

Como metodología de Desarrollo se usará OpenUP. Esta es una versión liviana del Proceso Unificado que plantea un enfoque iterativo e incremental dentro de un ciclo de vida estructurado. Es una metodología ágil que se enfoca en la naturaleza colaborativa del desarrollo de software. En OpenUP sólo se incluye el contenido fundamental [32].

Los cuatro principios que se listan a continuación capturan las intenciones generales detrás de OpenUP [33]:

- Balanceo de prioridades para maximizar beneficios de los Interesados.
- Colaboración para fomentar un entendimiento común y alinear intereses.
- Enfoque temprano en la arquitectura para minimizar riesgos y organizar el desarrollo.
- Evolución para obtener retroalimentación continua y mejoras.

Si bien OpenUP es derivado de RUP, puede ser considerado como una metodología ágil puesto que sus principios claves coinciden con enunciados del manifiesto por el desarrollo ágil de software.

OpenUp consta de varias fases para la elaboración de un proyecto de software. Cada fase tiene un enfoque y un conjunto de objetivos que sirven para medir el estado del proyecto y a su vez determinar si se avanza a la siguiente fase o si se realiza una nueva iteración en la fase actual [34].

5.1. Fase de Inicio

Esta fase busca entender qué se quiere con el proyecto y obtener la mayor información posible, lo cual es necesario para determinar la viabilidad del proyecto.

El principal objetivo de esta fase es lograr consenso sobre los objetivos del proyecto. Dicho objetivo es alcanzado al proponer una visión que explique lo que se planea construir; identificar requisitos clave, determinar al menos una arquitectura candidata y su viabilidad y hacerse una idea sobre el caso general y los riesgos que implica el proyecto [35].

5.2. Fase de Elaboración

En la segunda fase es donde los riesgos arquitecturalmente significativos del proyecto son dirigidos.

El propósito de esta fase es establecer la línea base de la arquitectura del sistema y proporcionar una base estable para la siguiente fase de desarrollo. En esta fase se obtiene un entendimiento detallado sobre los requisitos, obteniendo mayor entendimiento de aquellos que van a ser validados por la arquitectura. Se implementa una arquitectura ejecutable, es decir, se diseña, implementa y prueba un esqueleto estructural del sistema. Se toman acciones de mitigación frente a los riesgos identificados al diseñar, implementar y probar la arquitectura ejecutable [35].

5.3. Fase de Construcción

Es de las fases con mayor número de iteraciones y donde se realiza mayor cantidad de trabajo. La construcción se enfoca en el diseño, implementación y pruebas de las funcionalidades para desarrollar el sistema. Como resultado de esta fase se obtiene un sistema basado en la arquitectura propuesta, un producto

completo. Es de crucial importancia que en esta fase se logre minimizar el costo del desarrollo [36].

5.4. Fase de Transición

La transición se enfoca en el despliegue del software a los usuarios y asegurar que sus expectativas hayan sido alcanzadas. En esta fase por lo general se depuran errores, se mejora el funcionamiento y la usabilidad, además, se debe lograr un consenso general por parte de todos los implicados en el proyecto sobre lo completo del desarrollo. El propósito en esta fase es tener seguridad que el software esté listo para entregarse a los usuarios [37].

5.5. Aplicación de la metodología

Durante la etapa inicial del proyecto se realizó el levantamiento de requisitos de la aplicación donde se determinó que ésta debería ser extensible y permitir agregar nuevos análisis fácilmente en el futuro debido a la imposibilidad de abarcar inicialmente todos los problemas y algoritmos existentes en el momento. Se planteó la necesidad de realizar conexiones a múltiples bases de datos de diferentes proveedores y de almacenar la configuración de análisis ingresada por el usuario para permitir realizar análisis futuros empleando la misma configuración. Con esto en mente el equipo de desarrollo se reunió para determinar la versión inicial de la arquitectura que permitiera alcanzar los requisitos planteados.

Luego de diseñar la arquitectura se procedió a implementar el modelo de la aplicación que serviría como soporte para la adición de las nuevas funcionales a medida que estas se fueran desarrollando. Estos incrementos eran revisados aproximadamente cada dos semanas por el equipo de desarrollo para realizar demostraciones de lo construido hasta el momento, determinar los avances siguientes y sugerir nuevas ideas que facilitarán la usabilidad del software. Luego

de fijar la meta propuesta se evaluaba el impacto generado al desarrollo actual, cuánto esfuerzo tomará realizar la tarea y quién del equipo debía llevarla a cabo. La división del trabajo se realizaba agrupando funcionalidades que compartieran rasgos comunes o que fueran independientes (modulares) respecto a las demás. Un ejemplo de esto es la implementación de los problemas soportados por la aplicación, en donde un desarrollador era encargado de diseñar, construir y probar uno o varios problemas que luego se integrarían a lo construido actualmente.

En algunos casos especiales se llegó a emplear la técnica de programación en parejas con el fin de tener una segunda revisión y le permitiera concentrarse a uno de los integrantes del equipo en lo que debería hacer el código y no en que escribir.

Según la metodología el siguiente paso a llevar a cabo es el despliegue y prueba de la aplicación en los equipos del cliente, pero en nuestro caso, el rol del cliente es llevado a cabo por el mismo equipo de desarrollo y el director de tesis quienes determinan los criterios de aceptación del software.

5.6. Ambiente de desarrollo

El entorno de desarrollo escogido fue el Visual Studio 2010 ya que es el IDE oficial para desarrollo en .NET, dispone de un entorno visual e intuitivo y un amplio número de herramientas integradas que facilitan el desarrollo de aplicaciones en C#.

El desarrollo de la aplicación se apoyó en diferentes estrategias y herramientas para ayudar a mitigar los problemas propios del trabajo en equipo cuando sus miembros no siempre pueden estar juntos al avanzar en la codificación. La estrategia más sobresaliente fue la de centralizar los datos que debían ser accedidos por los miembros del equipo en servidores de hosting ubicados en internet con el fin de que los cambios realizados por un colaborador se reflejaran

inmediatamente para los demás. Los casos concretos donde se aplicó la estrategia son los siguientes:

- La base de datos de la aplicación estaba ubicada en un hosting gratuito en internet [38], lo que evitó una constante restauración de back-ups para que los cambios en el modelo de datos pudieran ser visualizados por el otro integrante del equipo.
- Se empleó la herramienta Tortoise SVN [39] en conjunto con un servidor para el control de versiones [40] con el fin de facilitar la recuperación en caso de error y la replicación de los cambios realizados a la aplicación por el equipo de desarrollo.
- Debido al trabajo simultáneo en diferentes aspectos de la aplicación se dificultaba a ambos desarrolladores el conocer todos los detalles del software, y por lo tanto, su ajuste en caso de error. Esto generó la necesidad de emplear una herramienta para la administración de los errores encontrados. La herramienta escogida para la labor fue etraxis [41]. Esta aplicación web brindó al equipo de trabajo la posibilidad de gestionar los errores hallados y las mejoras pendientes por realizar.

CAPÌTULO 6
REQUISITOS

En este capítulo se enunciarán los requisitos de la aplicación:

Requisitos funcionales:

- Se necesita una aplicación que permita elaborar análisis de varios tipos de problema entregando resultados por campo, tabla y base de datos.
- En una sola configuración se debe poder realizar el análisis de varias bases de datos.
- El usuario debe poder escoger entre los algoritmos disponibles para cada problema y realizar las configuraciones pertinentes.
- La aplicación debe entregar reportes para cada problema entregando resultados por campo, tabla y base de datos.
- La aplicación debe guardar en una base de datos los resultados de todos los análisis.
- Inicialmente la aplicación debe soportar 3 tipos de bases de datos.
- Inicialmente la aplicación debe soportar 3 tipos de problemas diferentes.
- Por cada resultado del análisis se debe ver el detalle de registros que tienen errores.
- La aplicación debe guardar las configuraciones realizadas para retomarlas más adelante y realizar modificaciones o completar los pasos faltantes.

Requisitos no funcionales:

- La aplicación debe permitir conectarse a diferentes motores de bases de datos.

- La aplicación debe ser intuitiva, para que usuarios sin mucha experiencia sean capaces de utilizarla para lo cual se sugiere un funcionamiento tipo wizard para facilitar al usuario la configuración del perfilamiento.
- La aplicación debe permitir la inclusión de nuevos tipos de bases de datos a las cuales realizar análisis, algoritmos y problemas.
- Debe ser una aplicación de escritorio.
- La aplicación debe estar integrada a una metodología de análisis de datos.

CAPÍTULO 7
ARQUITECTURA

7.1. Modelo general de arquitectura

La arquitectura base Fig. 1 se planteó con el objetivo de ser escalable para cumplir con el requisito de poder agregar nuevas bases de datos, problemas y algoritmos. Esta arquitectura facilita además la modificación de las funcionalidades ya implementadas sin afectar los demás componentes.

Para agregar un nuevo problema o base de datos se requiere heredar de clases base, las cuales definen los comportamientos fundamentales que deben tener cada uno de los hijos, los cuales se encargan de implementar el comportamiento en específico según la base de datos o el problema.

7.2. Criterios de diseño de la arquitectura

Para la construcción de esta arquitectura se emplearon los siguientes criterios:

- Ya que cada módulo requiere unos tipos de datos y clases particulares para su funcionamiento crear un medio de comunicación directo entre ellas hubiese generado un alto acoplamiento, por tal motivo se optó por crear un medio de comunicación genérico entre los módulos que mitigara el problema permitiendo comunicar cualquier tipo de dato.
- El acceso a datos se planteó en un módulo independiente a los problemas ya que el agregar una nueva base de datos no debería afectar los problemas soportados por la aplicación. De igual forma se planteó el módulo de problemas en el cual el agregar uno nuevo no afecta el acceso a los datos.

- El módulo de reportes se planteó de una manera separada para permitirle al usuario visualizar los resultados de un perfilamiento sin necesidad de realizar el análisis nuevamente.
- Un problema puede tener diferentes técnicas de solución. Esto también es flexible dentro de esta arquitectura ya que se pueden agregar nuevas técnicas fácilmente.

A continuación se dará una explicación de cada uno de los módulos de la arquitectura.

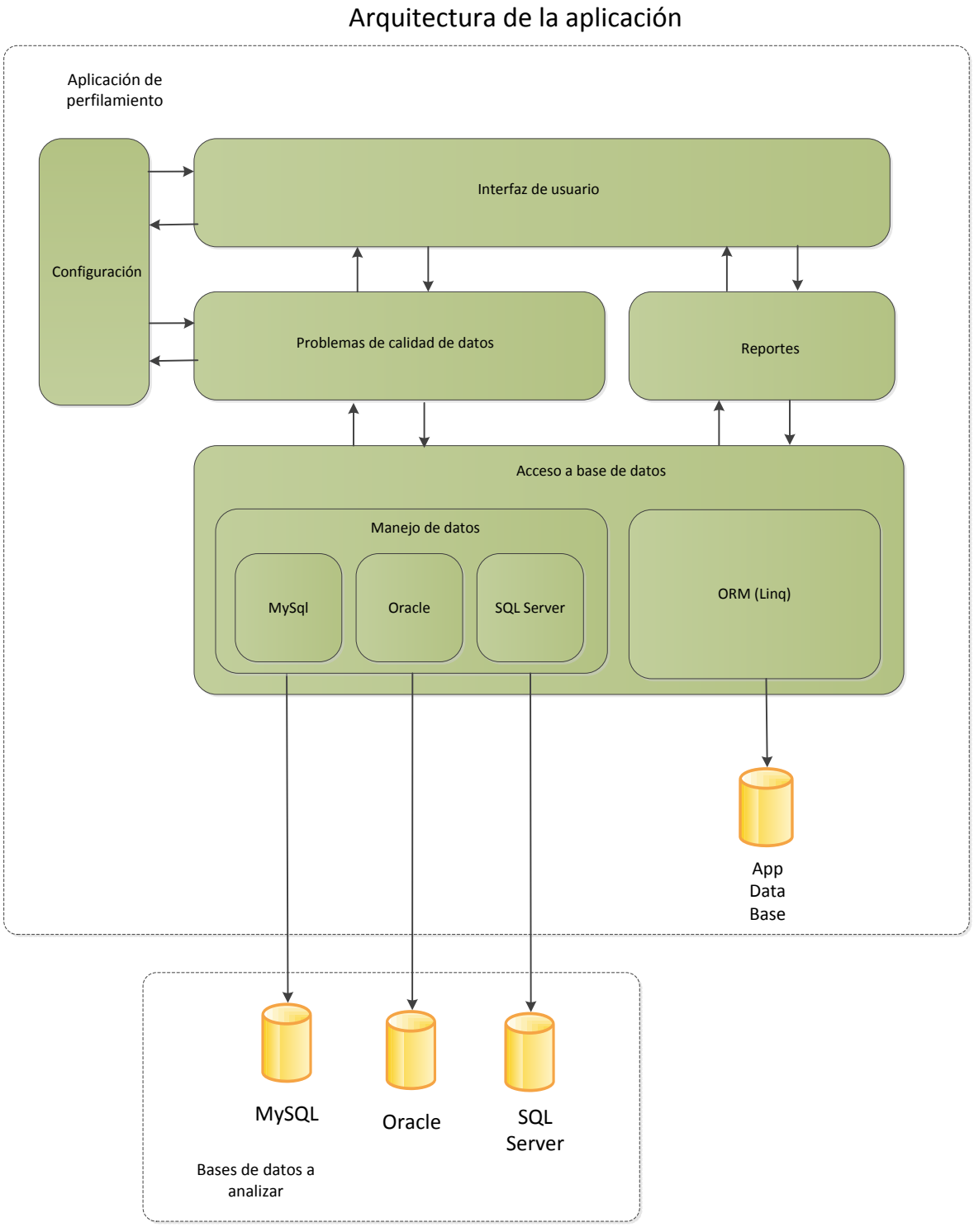
Arquitectura de la aplicación
Aplicación de perfilamiento
Configuración
Interfaz de usuario
Problemas de calidad de datos
Reportes
Acceso a base de datos
Manejo de datos
MySql
Oracle
SQL Server
ORM (Linq)
App Data Base
MySQL
Oracle
SQL Server
Bases de datos a analizar

Fig. 1

7.3. Casos de uso

A continuación se presentan los casos de uso que fueron identificados para la aplicación:

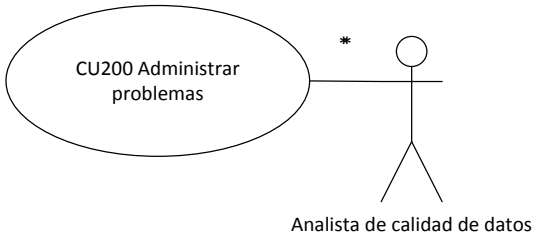

Fig. 2

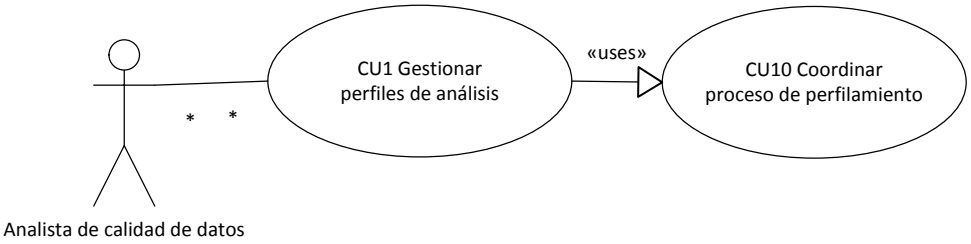

Fig. 3

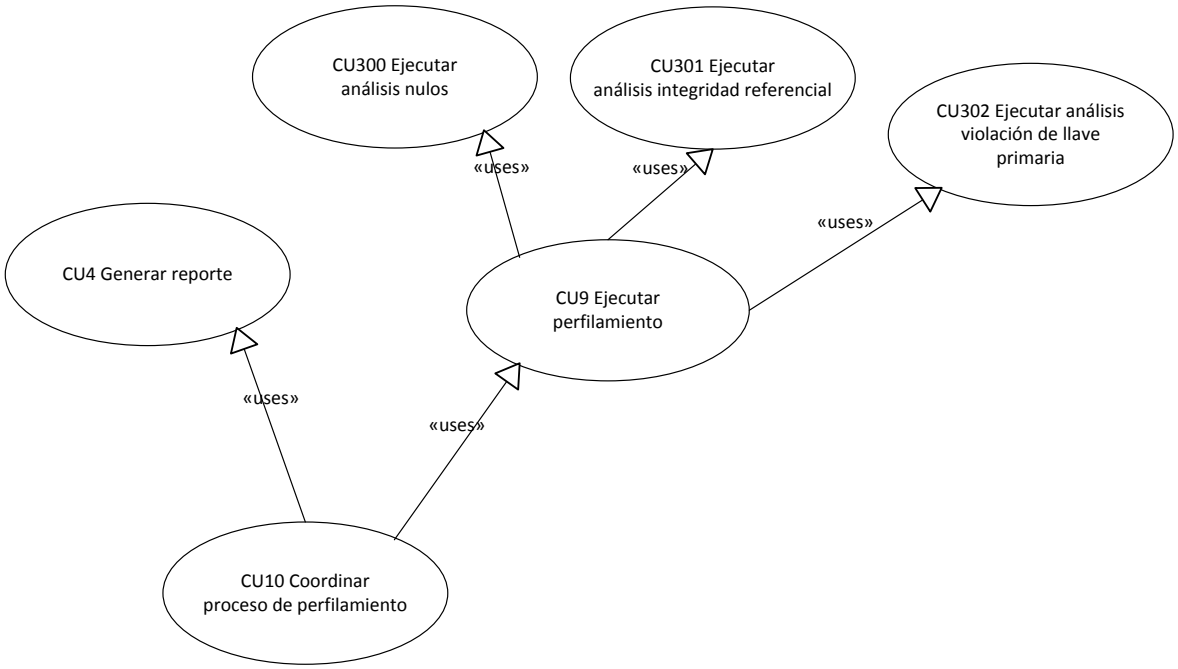
CU300 Ejecutar
análisis nulos
CU301 Ejecutar
análisis integridad referencial
CU302 Ejecutar análisis
violación de llave
primaria
«uses»
«uses»
«uses»
CU4 Generar reporte
CU9 Ejecutar
perfilamiento
«uses»
«uses»
CU10 Coordinar
proceso de perfilamiento

Fig. 4

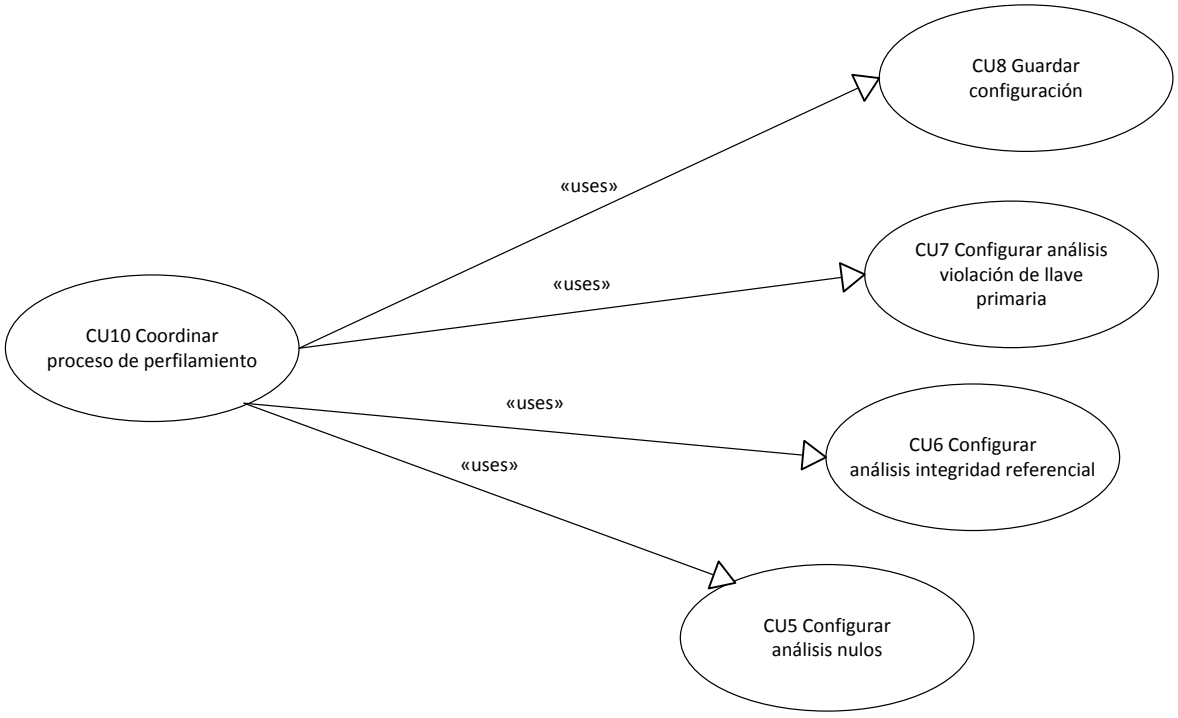
CU8 Guardar
configuración
«uses»
CU7 Configurar análisis
violación de llave
primaria
«uses»
CU10 Coordinar
proceso de perfilamiento
«uses»
CU6 Configurar
análisis integridad referencial
«uses»
CU5 Configurar
análisis nulos

Fig. 5

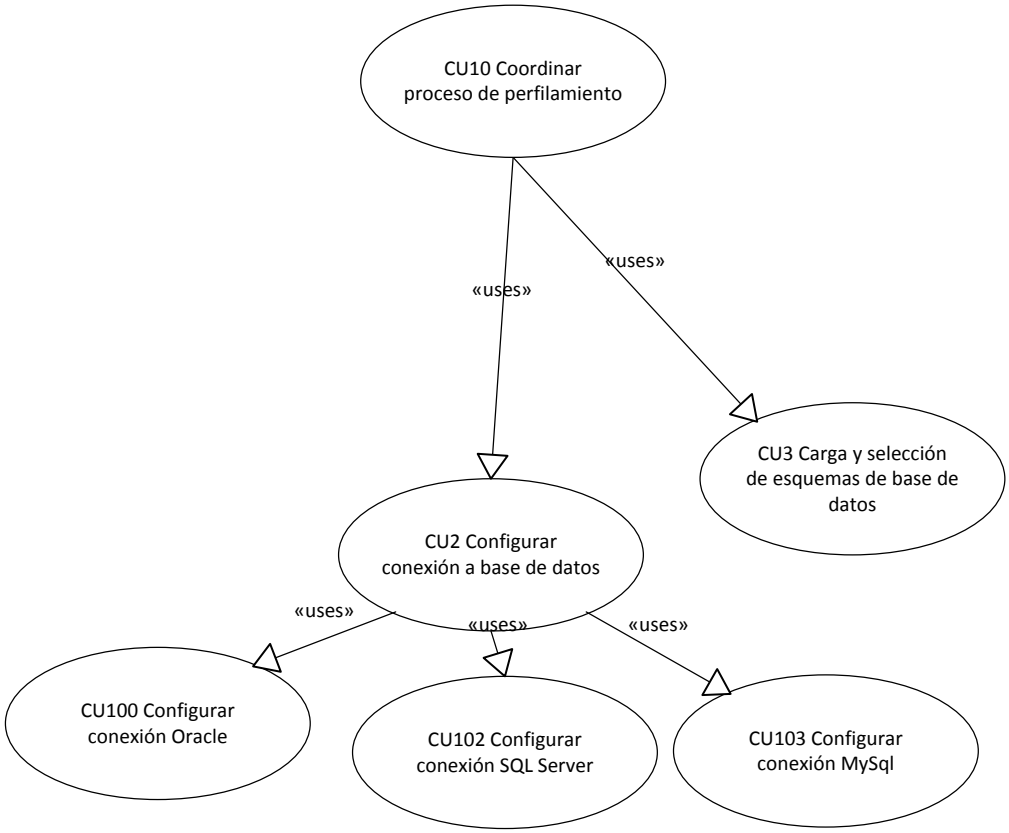

Fig. 6

7.4. Módulos

Los módulos en la arquitectura son los bloques de construcción en los cuales esta se divide. Este manejo permite la separación y agrupación de las funcionalidades de la aplicación y su actualización de manera que no impacte los demás componentes.

7.4.1. Acceso a base de datos

Este módulo se definió para dar abstracción a los detalles propios de cada motor de base de datos ya que uno de los requisitos de la aplicación es permitir la conexión a múltiples fuentes.

Para lograr dicha tarea se construyó una clase abstracta que define las funcionalidades básicas necesarias para el acceso a los datos, tales como: construcción del esquema de base de datos, conexión, desconexión y ejecución de queries. Estos detalles se abordaron creando clases que la heredaban y empleaban drivers específicos del proveedor además de encargarse de la construcción del string de conexión y aportar a las consultas elementos propios de la sintaxis de cada motor.

Para dar soporte a una nueva base de datos se deben seguir los siguientes pasos:

1. Crear una clase que herede e implemente la clase *DataManager* que define las funcionalidades básicas necesarias para el acceso a los datos, ejemplo: conexión, desconexión, ejecución de consultas, etc.
2. Crear una clase que herede e implemente la clase *DataBaseConnectionsUI* que define las funciones básicas para cada interfaz de configuración de conexión a una base de datos, tales como: validación de campos o generación de strings de conexión.
3. Se debe agregar el nuevo *DataManager* a la clase que realiza el Factory de los *DataManagers* disponibles en la aplicación.

7.4.2. Problemas de calidad de datos

Este componente agrupa los problemas soportados por la aplicación y los algoritmos de cada uno. Es el encargado de realizar el perfilamiento y análisis de los datos contenidos en las bases de datos, tablas y campos configurados por el usuario, generando los queries y ejecutándolos a través del módulo mencionado en 7.4.1.

7.4.3. Reportes

Una vez realizado el perfilamiento a los datos se hace necesario presentar al usuario los resultados de dicho análisis. Las disimilitudes propias de la naturaleza de cada problema y de las métricas generadas por estos, llevaron a que la implementación de un reporte genérico fuese compleja debido a que la interpretación de los resultados está fuertemente relacionada con el algoritmo que los generó. Esta situación se abordó creando una clase abstracta implementada por cada algoritmo en una clase concreta que está en capacidad de entender las métricas generadas. Cada subclase está encargada de consultar los resultados y definir los tipos de gráficas a utilizar (gráficos de torta, tablas o texto) con el fin de presentar los resultados del análisis.

7.4.1. Interfaz de usuario

La interfaz de usuario es el medio por el cual los componentes lógicos obtienen sus parámetros de funcionamiento y presentan los resultados del análisis. En este aspecto se dio a la aplicación un funcionamiento tipo wizard que brinda un manejo intuitivo y requiere una pequeña curva de aprendizaje para el usuario final debido a su extendido uso en las aplicaciones actuales. Este enfoque también permite a los desarrolladores agregar nuevos problemas a la aplicación de una manera sencilla y modular que genera poco impacto para lo construido hasta el momento.

Este módulo está compuesto por varias secciones tales como:

- Administrador de problemas: permite crear, actualizar y eliminar tipos de problemas a la aplicación, los cuales se utilizan para la carga de librerías.
- Landing de la aplicación: permite al usuario eliminar un perfil de análisis o escoger uno de los almacenados previamente para ser ejecutado.
- Conexión a bases de datos: en esta sección se permite al usuario seleccionar el motor de base de datos al cual se desea conectar y según su elección se carga la interfaz de usuario encargada de pedir y validar los parámetros de conexión necesarios.

- Configuración de problemas: esta sección se construye dinámicamente empleando los problemas agregados a través del Administrador de problemas. En una primera parte se le permite al usuario escoger cuales desea analizar en cada campo, tabla o base de datos. Luego de la selección, cada problema es el encargado de definir y retornar su interfaz de configuración mediante la cual se ingresarán los parámetros de análisis.
- Análisis: esta sección permitirá al usuario visualizar el avance en la ejecución del análisis de los problemas. Cada algoritmo está en la capacidad de reportar en pantalla el progreso de su ejecución por medio de eventos.
- Reportes: esta sección permite al usuario visualizar los resultados generados durante el análisis a la base de datos, la tabla o el campo según el algoritmo empleado para el perfilamiento.

7.4.2. Configuración

El módulo de configuración permite a los componentes de la aplicación brindar un contexto global a las variables locales que así lo requieran. Luego de que una variable alcanza este nivel, las demás páginas del wizard podrán leer, eliminar o editar su valor sin necesidad de estar acopladas con la entidad que la generó, brindando así a este módulo la capacidad de servir como medio de “comunicación” entre los componentes de la aplicación.

7.5. Modelo de datos

Para la aplicación se diseñó un modelo de datos flexible que permitiera el almacenamiento de los resultados generados durante el perfilamiento a las bases

de datos. La flexibilidad radica en brindar la capacidad de almacenar los valores generados por cada tipo de problema independientemente de la naturaleza o significado del dato. Este funcionamiento permitió una fácil integración de nuevos problemas a medida que se realizaba su construcción. A continuación se explicarán las entidades del modelo de datos.

TProblem: Esta tabla almacena el conjunto de problemas que soporta la aplicación. Los campos son los siguientes:

- TProblemId: Identificador único de la tabla.
- ProblemName: Contiene el nombre del problema (Nulos, Integridad Referencial, etc.)
- ClassName: Contiene el nombre de la librería que debe cargar la aplicación para el correcto funcionamiento del módulo.
- DefaultAlgorithm: Es el algoritmo por defecto que se utilizará al momento de configurar un análisis.

TProfile: Define una configuración de perfilamiento que se genera al seguir todos los pasos del wizard. En este proceso se obtienen las bases de datos a las cuales se conectará la aplicación, el esquemas de las mismas y las configuraciones propias de los algoritmos. Los campos son los siguientes:

- TProfileId: Identificador único de la tabla.
- ProfileName: Nombre que el usuario da al perfilamiento.
- Description: Información adicional que da el usuario acerca del perfilamiento que está guardando.
- ProfileConfiguration: Contiene toda la configuración creada en el wizard, serializada en formato binario.

TAnálisis: Define un análisis realizado en una fecha determinada con un perfil de análisis almacenado en *TProfile*. Un perfilamiento puede tener varios análisis asociados ya que se busca poder visualizar la evolución de la calidad de datos en las bases de datos, aunque esta funcionalidad no se implementó (visualizar el

histórico de análisis realizados), se deja la puerta abierta para su implementación. Los campos son los siguientes:

- TAnálisisId: Identificador único de la tabla.
- TProfileId: Identificador del perfilamiento al cual corresponde el análisis.
- AnalisysDate: Fecha en la cual se ejecutó el análisis.

TResult: Contiene los resultados generados por un algoritmo durante un análisis o perfilamiento. Por ejemplo si se cuenta la cantidad de nulos N en el campo C de la tabla T, el valor N se guardará en esta tabla. La principal caracterísitica de esta entidad es que permite almacenar cualquier métrica sin importar el algoritmo que la generó, brindando la posibilidad a futuros desarrollos de almacenar sus propias métricas. Esto se logró acompañando cada valor de un ResultMeaning (significado asociado a una métrica generada por un algoritmo) y de la base de datos, la tabla y el campo al cual aplica. Los campos son los siguientes:

- TResultId: Identificador único de la tabla.
- TAlgorithmId: Identificador del algoritmo al cual pertenece el resultado.
- TAnalisisId: Identificador del análisis al cual pertenece el resultado.
- DataBaseName: Nombre de la base de datos a la cual pertenece el resultado.
- TableName: Nombre de la tabla a la cual pertenece el resultado.
- ColumnName: Nombre de la columna a la cual pertenece el resultado.
- AuxiliarData: Contiene información adicional que sirve de ayuda para interpretar el resultado.
- ResultMeaning: Indentifica el significado de un resultado o valor, dado que un mismo algoritmo puede generar múltiples métricas y a que estas son almacenadas en una misma tabla. Por ejemplo: Total de nulos, Total de registros y Total de registros no nulos son valores numéricos que puden ser generados por un mismo algoritmo pero poseen interpretaciones diferentes.
- Value: Resultado del análisis.

DrillDown: Contiene el detalle de un resultado, esto es, los registros que tienen problemas de calidad de datos y fueron incluidos o contabilizados en el resultado. Por ejemplo si se contó la cantidad de nulos en el campo C de la tabla T, se almacenarán los registros que poseen valores nulos en ese campo. Dado que la información resultante puede ser de gran tamaño, en este campo se guarda únicamente el nombre del archivo en disco que contiene los registros reales. Los campos son los siguientes:

- DrillDownId: Identificador único de la tabla.
- TAnálisisId: Identificador del análisis al cual pertenece el detalle.
- TAlgorithmId: Identificador único del algoritmo, este identificador lo posee cada algoritmo en su librería. Permite identificar a cual de los algoritmos pertenece el detalle.
- DataBaseName: Nombre de la base de datos a la cual pertenece el resultado.
- TableName: Nombre de la tabla a la cual pertenece el resultado.
- ColumnName: Nombre de la columna a la cual pertenece el resultado.
- AuxiliarData: Información adicional que utiliza el algoritmo al almacenar los resultados para dar información adicional al momento de recuperar la información.
- FileName: Nombre del archivo que contiene el detalle de los registros.
- CreationDate: Fecha en la cual se almacenó el detalle.

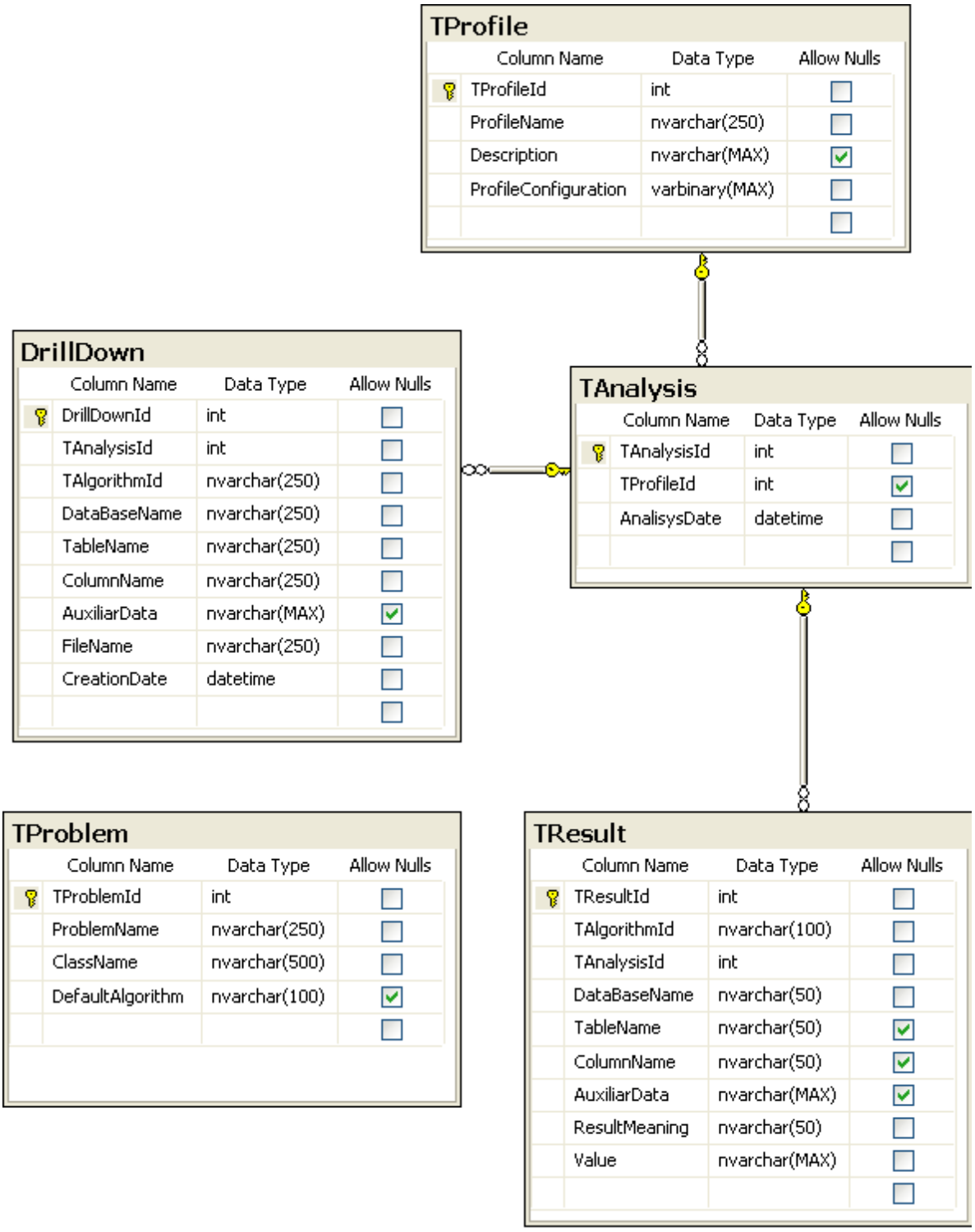
TProfile
Column Name
Data Type
Allow Nulls
TProfileId
int
ProfileName
nvarchar(250)
Description
nvarchar(MAX)
ProfileConfiguration
varbinary(MAX)
DrillDown
Column Name
Data Type
Allow Nulls
DrillDownId
int
TAnalysisId
int
TAlgorithmId
nvarchar(250)
DataBaseName
nvarchar(250)
TableName
nvarchar(250)
ColumnName
nvarchar(250)
AuxiliarData
nvarchar(MAX)
FileName
nvarchar(250)
CreationDate
datetime
TAnalysis
Column Name
Data Type
Allow Nulls
TAnalysisId
int
TProfileId
int
AnalisysDate
datetime
TProblem
Column Name
Data Type
Allow Nulls
TProblemId
int
ProblemName
nvarchar(250)
ClassName
nvarchar(500)
DefaultAlgorithm
nvarchar(100)
TResult
Column Name
Data Type
Allow Nulls
TResultId
int
TAlgorithmId
nvarchar(100)
TAnalysisId
int
DataBaseName
nvarchar(50)
TableName
nvarchar(50)
ColumnName
nvarchar(50)
AuxiliarData
nvarchar(MAX)
ResultMeaning
nvarchar(50)
Value
nvarchar(MAX)

Fig. 7

7.6. Tecnologías de desarrollo

A continuación se presentan las diferentes tecnologías y librerías utilizadas para el desarrollo de la aplicación.

7.6.1. Tecnologías Microsoft

Se seleccionó el .NET Framework 3.5 el cual incluye tecnologías como Reflection, Linq To SQL, Linq To XML y el lenguaje C# debido a la experiencia en el uso de éstas y a que brindan agilidad y rapidez en la construcción de aplicaciones, además el framework posee otras características como las presentadas en [42]:

- Proporcionar un entorno de programación orientado a objetos consistente ya sea que el código se almacene y se ejecute localmente, se almacene a nivel local, distribuya en Internet o se ejecute en forma remota.
- Proporcionar un entorno de ejecución de código que reduzca al mínimo la implementación de software y los conflictos de versiones.
- Proporcionar un entorno que promueve la ejecución segura del código, incluyendo el código creado por un tercero desconocido o de poca confianza.
- Proporcionar un entorno de ejecución de código que elimine los problemas de rendimiento de los entornos de secuencias de comandos o interpretado.
- Hacer la experiencia del desarrollador consistente a través de diversos tipos de aplicaciones, tales como aplicaciones basadas en Windows y aplicaciones basadas en Web.
- Construir todas las comunicaciones en estándares de la industria para asegurar que el código basado en. NET Framework se puede integrar con cualquier otro código.

En el aspecto de acceso a datos, se utilizó la tecnología Linq To SQL debido a que brinda una abstracción del modelo entidad-relación al permitir al desarrollador interactuar con los datos como si fueran objetos ordinarios. Esta tecnología se utilizó en conjunto con el motor de base de datos SQL Server 2008 Express incluido en el ambiente de desarrollo utilizado, Visual Studio 2010, y para la

manipulación del esquema de la base de datos se empleó SQL Server Management Studio 2008.

7.6.2. Librerías utilizadas

7.6.2.1. Acceso a datos

Uno de los requisitos de la aplicación es el de permitir la conexión a diferentes bases de datos, por lo que se emplearon los drivers propios de cada fabricante para mejorar el rendimiento. La aplicación actualmente soporta conexiones a SQL Server 2008, Oracle 11G Express y MySQL por lo que se requirió de las siguientes librerías:

- MySQL: Se utilizó el conector versión 6.4.4 para .NET, del cual se empleó la librería mysql.data.dll
- Oracle: Para este fabricante se requiere instalar el ODP .NET (Oracle Data Provider) el cual instala un conjunto de librerías y variables de entorno las cuales utiliza para el funcionamiento del cliente. Se utilizó el ODAC 11.2 Release 2(11.2.0.1.2) con herramientas de Oracle para Visual Studio. Para utilizar el driver se referenció la librería Oracle.Data.Access.dll
- SQL Server 2008: dado que este motor es propietario de Microsoft se utilizaron las librerías ya provistas en el .NET Framework

7.6.2.2. Lectura de metadatos

Para la construcción del esquema de las bases de datos a las cuales se conecta la aplicación para realizar el perfilamiento se utilizan las herramientas ofrecidas en la librería DatabaseSchemaReader.dll del proyecto Database Schema Reader [43] licenciado bajo la modalidad de Licencia Pública de Microsoft (Ms-PL) .

7.7. Diagrama de despliegue

Este diagrama muestra como se distribuye la aplicación en las diferentes máquinas, se puede apreciar que la aplicación funciona en un sólo servidor y se conecta bases de datos externas para realizar los análisis.

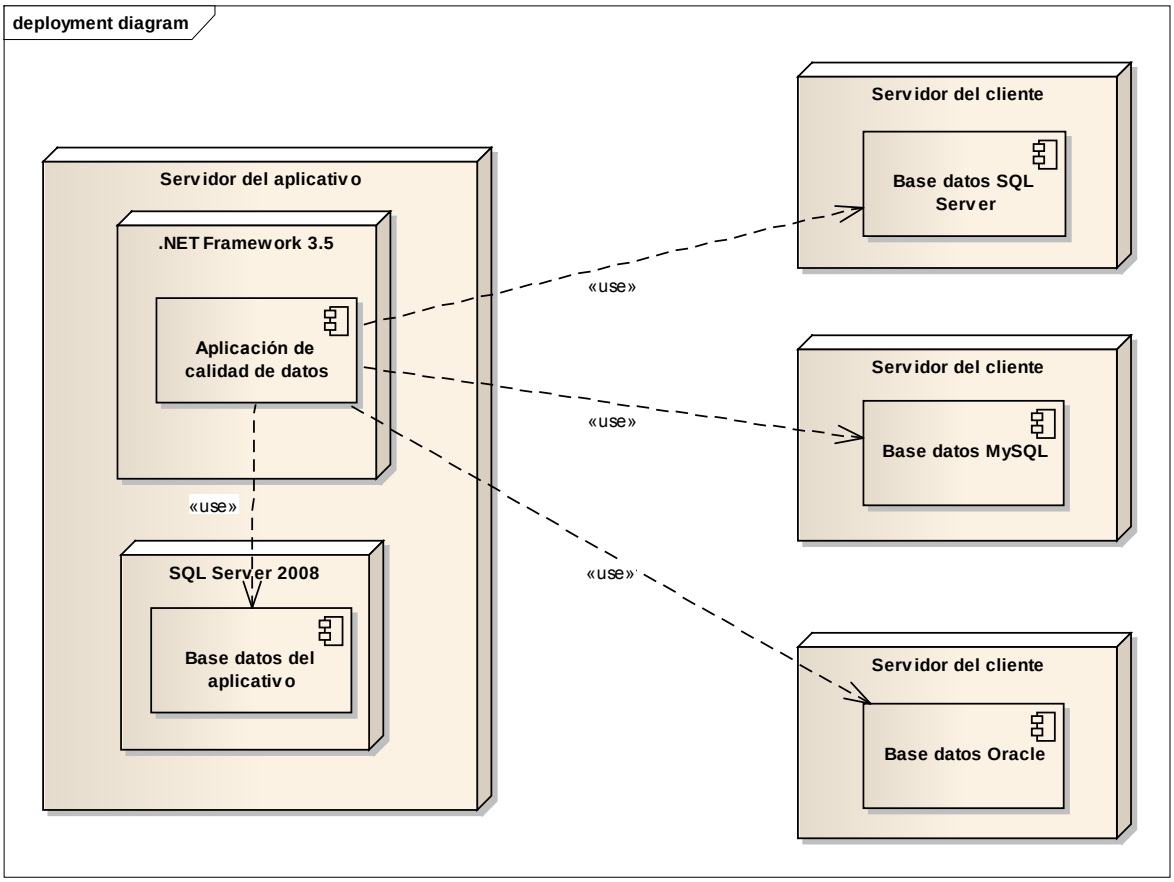

Fig. 8

7.8. Problemas soportados

El adicionar un nuevo problema a la aplicación consiste en agregar un nuevo módulo, realizar la implementación de las clases necesarias e integrarlo al wizard para mostrar la pantalla de configuración y las gráficas de resultados en los reportes. A continuación se explicarán los problemas implementados en la aplicación. Dado que se mostrarán sentencias T-SQL se utilizarán las siguientes convenciones:

Tn: Representa una tabla de la base de datos

Cn: Representa un campo de la base de datos

PKn: Campos que componen la llave primaria en la tabla n

FKn: Campos que componen la llave foránea en la tabla n

Los problemas soportados por la aplicación actualmente son los siguientes:

7.8.1. Nulos

Este problema es el encargado de buscar en las columnas seleccionadas por el usuario las celdas nulas. La ausencia de un valor, dependiendo de las definiciones del negocio, puede ser considerado un error, por ejemplo para campos definidos como llave primaria o cuyo valor es requerido para la entidad.

En este problema se calculan las siguientes métricas:

Cantidad de registros nulos: este valor puede ser visualizado a nivel campo, tabla y base de datos.

```
Cantidad de nulos = SELECT COUNT(*) FROM T WHERE C IS NULL
```

Cantidad de registros no nulos: este valor puede ser visualizado a nivel campo, tabla y base de datos.

```
Cantidad de no nulos = SELECT COUNT(*) FROM T – Cantidad de nulos
```

7.8.2. Integridad referencial

Este problema es el encargado de identificar los registros “huérfanos” y los “padres sin hijos” para las tablas seleccionadas por el usuario. Un análisis para este problema permitirá encontrar errores tales como ciudades que no estén asociadas a una región o facturas sin detalle. Para el análisis de este tipo de error

se creó la siguiente consulta sql en base al documento [44], el cual brinda un algoritmo para la detección de problemas de integridad referencial:

Si se tiene la tabla T1 cuya llave es PK1 y está siendo referenciada por la tabla T2 con clave foránea FK2 se calcularían las siguientes métricas:

Cantidad de Padres sin hijos =
```
SELECT COUNT(*) FROM T1 LEFT OUTER JOIN T2 ON PK1=FK2 WHERE FK2 IS NULL
```

Cantidad de Huérfanos =
```
SELECT COUNT(*) FROM T2 LEFT OUTER JOIN T1 ON PK1=FK2 WHERE PK1 IS NULL
```

7.8.3. Violación de llave primaria

Este problema es el encargado de identificar los registros en una entidad donde los valores en los campos definidos como llave primaria están repetidos en otra tupla de la tabla, por ejemplo, si se tienen dos números de factura idénticos o dos o más cédulas iguales. Las métricas calculadas son las siguientes:

Cantidad de registros cuyos valores en los campos C1, C2, C3…Cn definidos como llave primaria PK1están repetidos en la entidad T1

Cantidad de registros que violan la restricción de llave primaria
```
= SELECT PK1,
COUNT(*) AS total FROM T1 GROUP BY PK1 HAVING COUNT(*) > 1
```

Cantidad de registros válidos en la tabla = Tamaño de la tabla -
```
SELECT PK1,COUNT(*) AS total FROM T1 GROUP BY PK1 HAVING COUNT(*) > 1
```

CAPÌTULO 8
PRUEBAS

En este capítulo se presentan las pruebas realizadas para mostrar la eficacia de la aplicación, las pruebas realizadas para cada problema, los valores de control utilizados y los resultados obtenidos.

Para realizar las pruebas se creó la base de datos ‘Pruebas’ y en esta a su vez se crearon las tablas necesarias para realizarlas a cada problema.

8.1. Nulos

Para este problema se creó la tabla ‘Nulos’ con las columnas ‘IdProducto’, ‘Producto’, ‘Valor’ y ‘Descripción’ con los siguientes registros de pruebas:

IdProducto	Producto	Valor	Descripción
1	Porcelana	5000	Porcelana chiviada
2	NULL	1000	Carro último modelo
3	Computador	NULL	NULL
4	Libro	6000	Novela de terror
5	Teclado	50000	Qwerty
6	CD	1000	Regrabable
7	NULL	1500	DVD
8	Impresora	100000	Multifuncional
9	Parlante	20000	Parlantes Súper Boom
10	Microondas	200000	NULL
11	Portátil	30000	Marca Gato
12	Mouse óptico	20000	Láser rojo
13	Disco duro portable	100000	Sea Gate
14	Memoria USB	20000	8 GB
15	Procesador	NULL	NULL

Tabla 9

Como se puede observar, la tabla posee un total de 15 registros. La columna ‘IdProducto’ no posee valores nulos ya que es un auto-numérico, los campos ‘Producto’ y ‘Valor’ poseen 2 valores nulos cada uno y en ‘Descripción’ hay 3 valores nulos. Según esto, la aplicación deberá encontrar dichos valores para cada columna y un total de 7 valores nulos a nivel de tabla.

8.1.1. Resultados de la aplicación

Resultados de valores a nivel de la tabla:

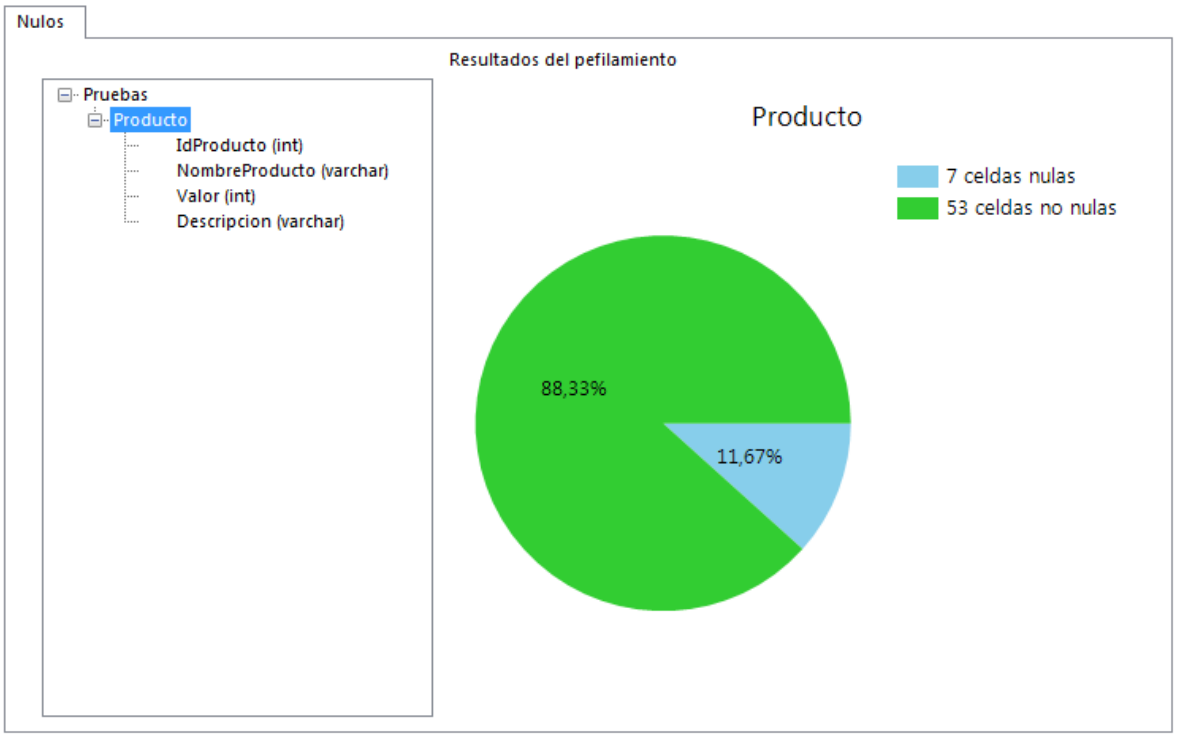

Fig. 9

Resultados para la columna ‘IdProducto’:

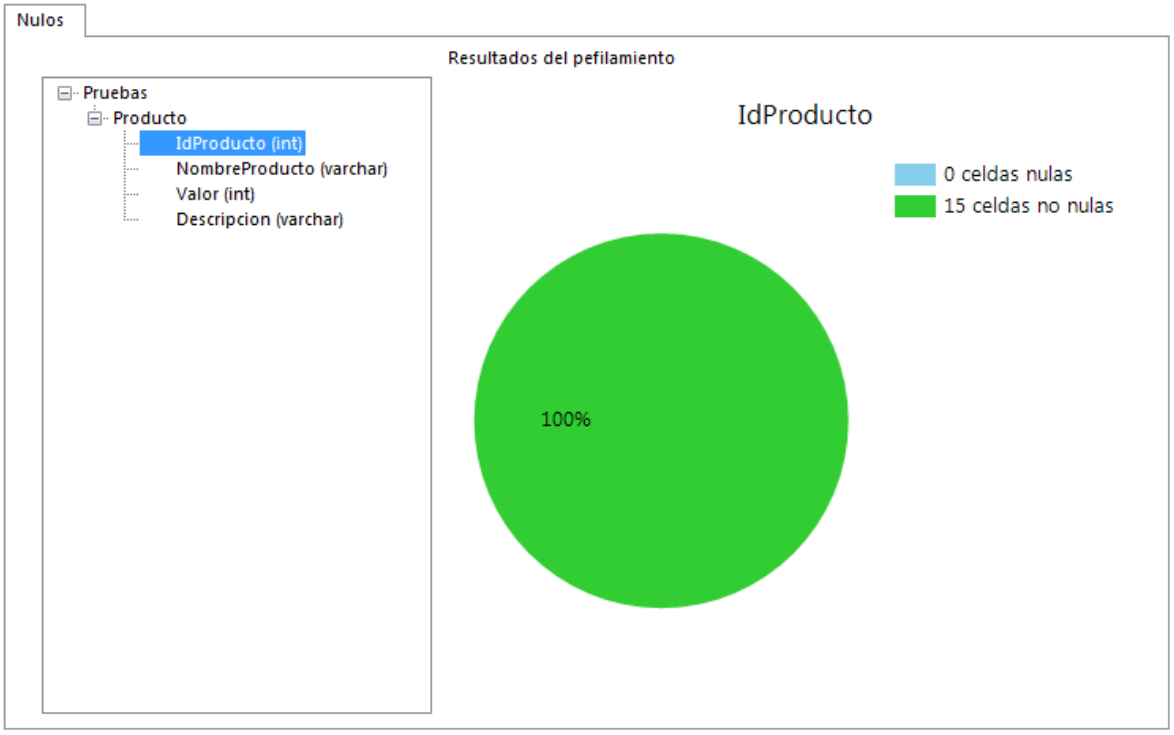

Fig. 10

Resultados para la columna ‘NombreProducto’:

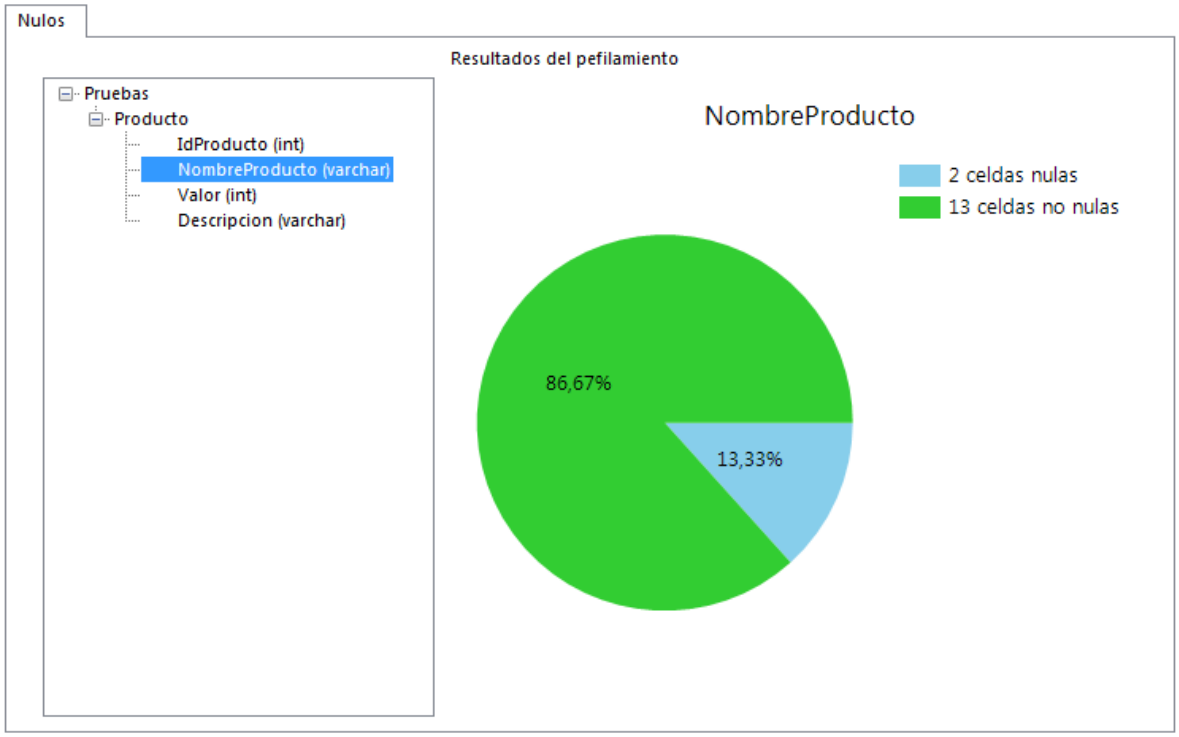

Fig. 11

Resultados para la columna ‘Valor’:

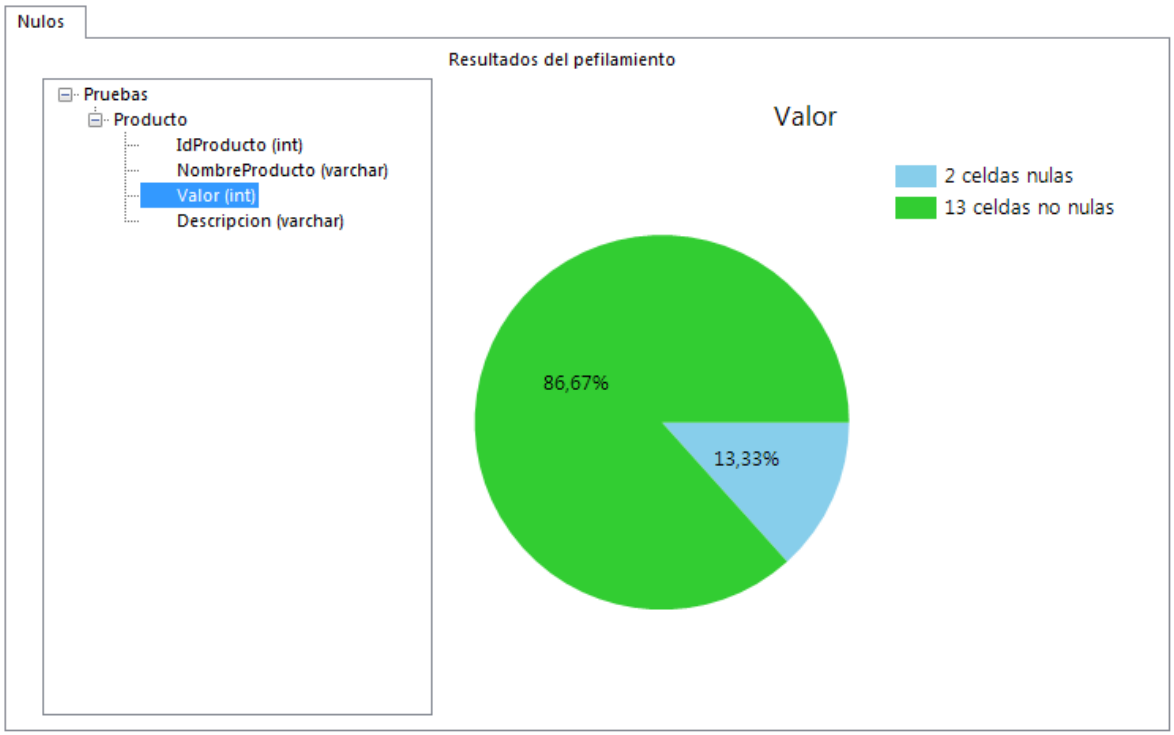

Fig. 12

Resultados para la columna ‘Descripcion’:

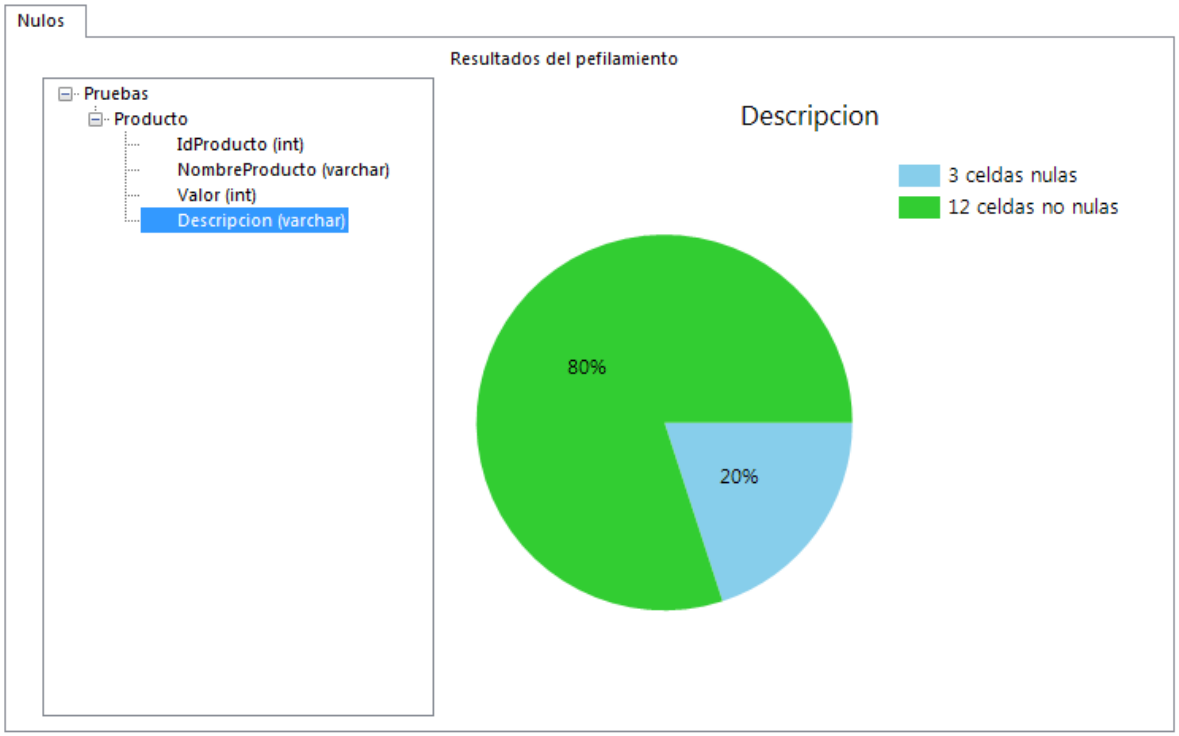

Fig. 13

Los resultados anteriores muestran que la aplicación identificó todos los registros que presentan valores nulos, brindando un 100% de eficacia.

8.2. Violación de llave primaria

Para este problema se creó la tabla ‘Persona’, con los campos ‘TipoDocumento’, ‘NumeroDocumento’ y ‘Genero’. En la tabla se crearon los siguientes registros:

TipoDocumento	NumeroDocumento	Genero
Cedula	123456	
Cedula	123456	Femenino
Cedula	100200300	Masculino
NIT	45678941	
NIT	100200300	NULL
Tarjeta identidad	45679216	1
Cedula	455743384	NULL

Cedula	100200300	NULL
NIT	3642178	NULL
Tarjeta identidad	541348843	Niño
NIT	NULL	4
Cedula	7980134	Masculino
Cedula	976465	Femenino
Cedula	74178234	Femenino
Cedula	74178234	Masculino

Tabla 10

Como se puede ver, si se define que la llave primaria de la tabla son los campos 'TipoDocumento' y 'NumeroDocumento', esto llevaría a que existiesen 6 registros con problemas de violación de llave primaria. Basados en esta premisa se configura la aplicación indicando que la llave primaria de la tabla es una llave compuesta por los campos 'TipoDocumento' y 'NumeroDocumento'. Esto se indicó de esa forma al configurar el problema. Ver Fig. 14

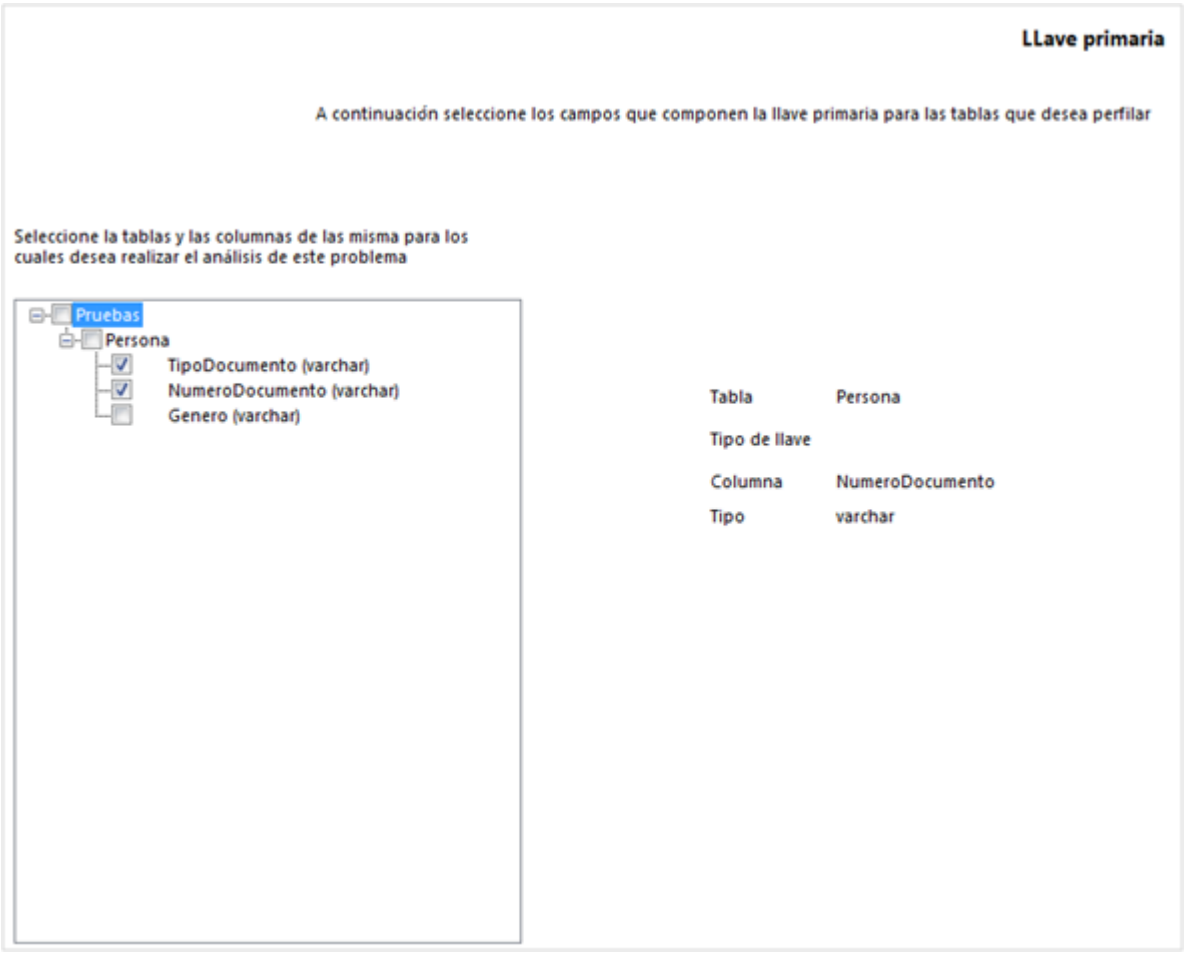

Fig. 14

Resultados para la tabla ‘Persona’:

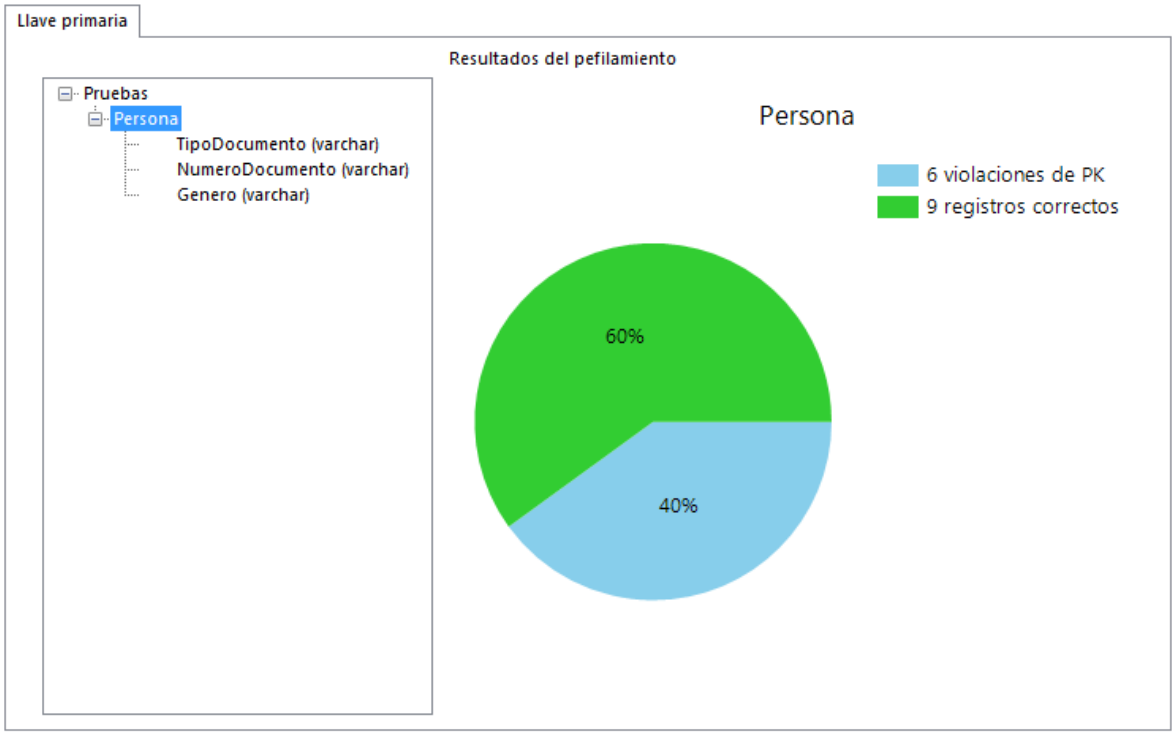

Fig. 15

Como se puede ver, la aplicación fue capaz de detectar los 6 registros que presentan problemas de violación de llave primaria.

8.3. Integridad referencial

Para las pruebas en este tipo de problema se utilizarán las tablas de los problemas anteriores y se agregarán dos adicionales: Factura y ProductoFactura. Para esta prueba se suponen las siguientes relaciones.

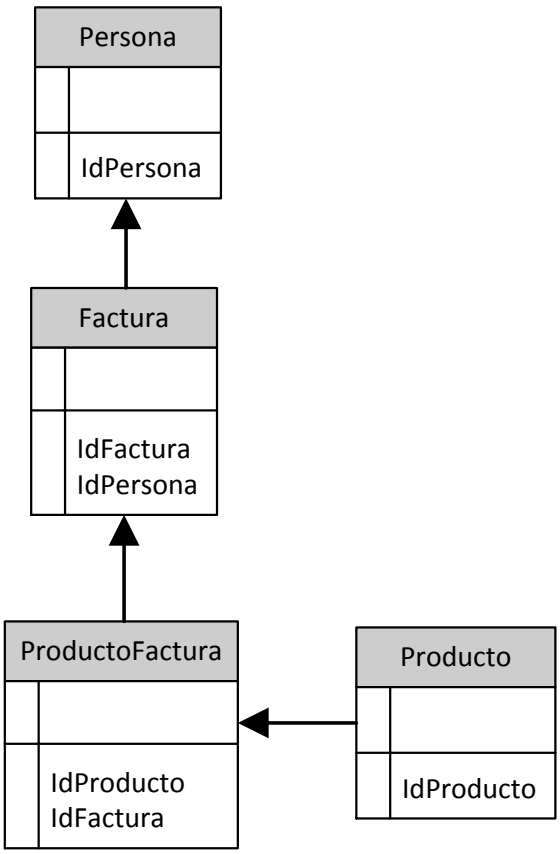

Fig. 16

Existe la entidad persona la cual tiene asociado un conjunto de facturas, las cuales a su vez tienen varios productos como detalle. Para efectos de la prueba no se crearon relaciones entre las tablas ya que esto impediría agregar errores intencionales en las llaves primarias y foráneas. Por tal motivo se supone que los campos mostrados en las tablas anteriores corresponden a las llaves primarias y foráneas según correspondan los nombres, por ejemplo el campo IdPersona corresponde a la llave primaria de la tabla Persona, y corresponde a la llave foránea en la tabla Factura, de igual forma con los demás campos.

Las tablas descritas poseen los siguientes registros:

Persona			
IdPersona	TipoDocumento	NúmeroDocumento	Género
1	Cedula	123456	
2	Cedula	123456	Femenino
3	Cedula	100200300	Masculino
4	NIT	45678941	
4	NIT	100200300	NULL
5	Tarjeta identidad	45679216	1
6	Cedula	455743384	NULL
7	Cedula	100200300	NULL
8	NIT	3642178	NULL
10	Tarjeta identidad	541348843	Niño
11	NIT	NULL	4
20	Cedula	976465	Femenino
21	Cedula	74178234	Femenino
NULL	Cedula	7980134	Masculino
23	Cedula	74178234	Masculino

Tabla 11

Factura				
IdFactura	IdPersona	SerieFactura	ValorFactura	Descripcion
1	2	1231425465	200000	Ninguna
2	23	23524536	235650	NULL

Tabla 12

ProductoFactura		
IdProductoFactura	IdProducto	IdFactura
NULL	1	1
NULL	4	1
NULL	5	1
NULL	3	1
NULL	6	2
NULL	10	2
NULL	300	2
NULL	6	3
NULL	7	3
NULL	21	3

Tabla 13

Producto			
IdProducto	NombreProducto	Valor	Descripcion
1	Porcelana	5000	Porcelana chiviada
2	NULL	1000	Carro último modelo
3	Computador	NULL	NULL
4	Libro	6000	Novela de terror
5	Teclado	50000	Qwerty

6	CD	1000	Regrabable
7	NULL	1500	DVD
8	Impresora	100000	Multifuncional
9	Parlante	20000	Parlantes Súper Boom
10	Microondas	200000	NULL
11	Portátil	30000	Marca Gato
12	Mouse óptico	20000	Láser rojo
13	Disco duro portable	100000	Sea Gate
14	Memoria USB	20000	8 GB
15	Procesador	NULL	NULL

Tabla 14

En la tabla Persona se hallan 13 llaves primarias que no se encuentran referenciadas por ningún registro hijo (llave foránea) lo cual se puede evidenciar en la Fig. 17

.

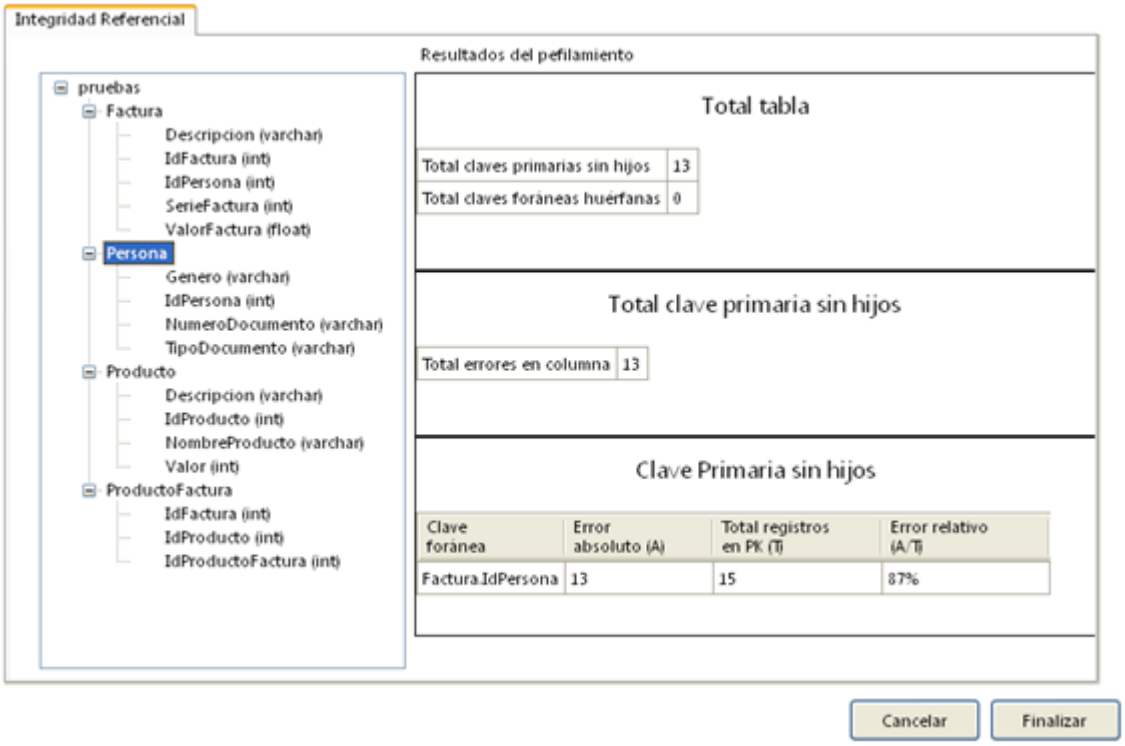

Fig. 17

Para la tabla Factura no tenemos errores de ningún tipo ya que todas las personas que referencia existen en la tabla Persona y cada clave primaria de ella está siendo asociada en la tabla ProductoFactura. Esto se puede observar en la figura Fig. 18

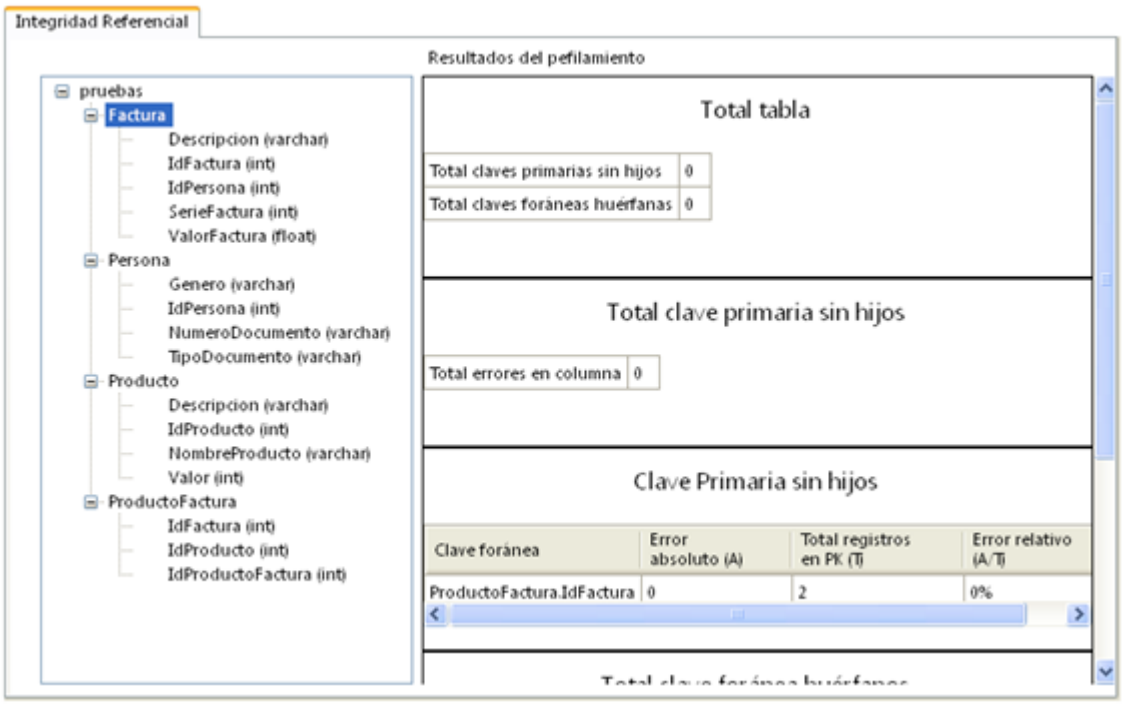

Fig. 18

En la tabla ProductoFactura encontramos 5 errores, 2 para el campo IdProducto ya que no existen estas entidades y 3 en el campo IdFactura, ya que la respectiva factura no existe. Estos resultados coinciden con los reportes generados por la aplicación como lo muestra Fig. 19

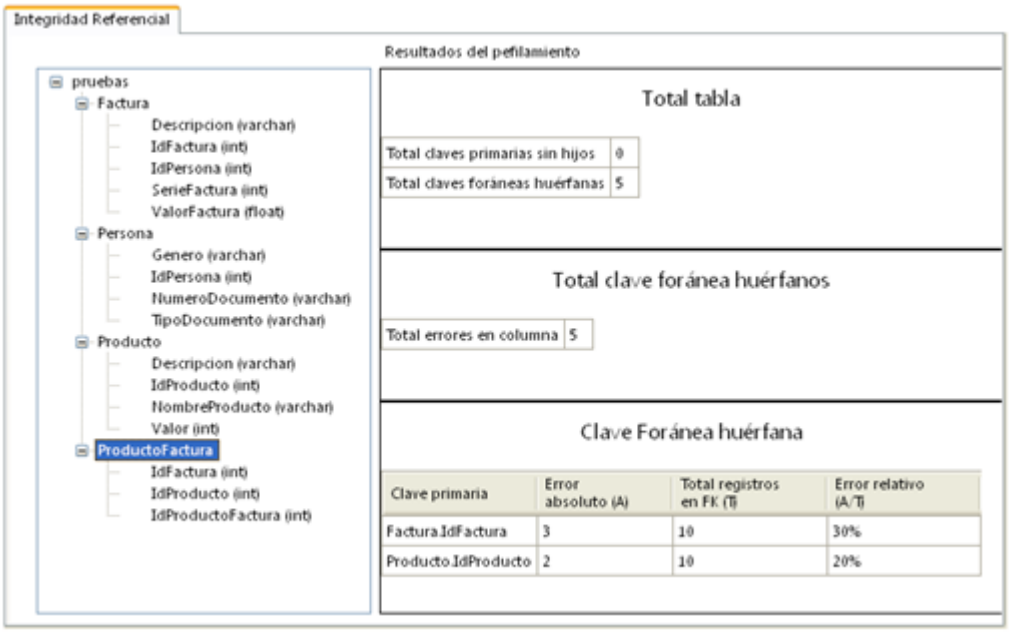

Fig. 19

Por último, en la tabla encontramos 8 errores de registros que no están siendo referenciados por ninguna otra tabla, en este caso ProductoFactura.

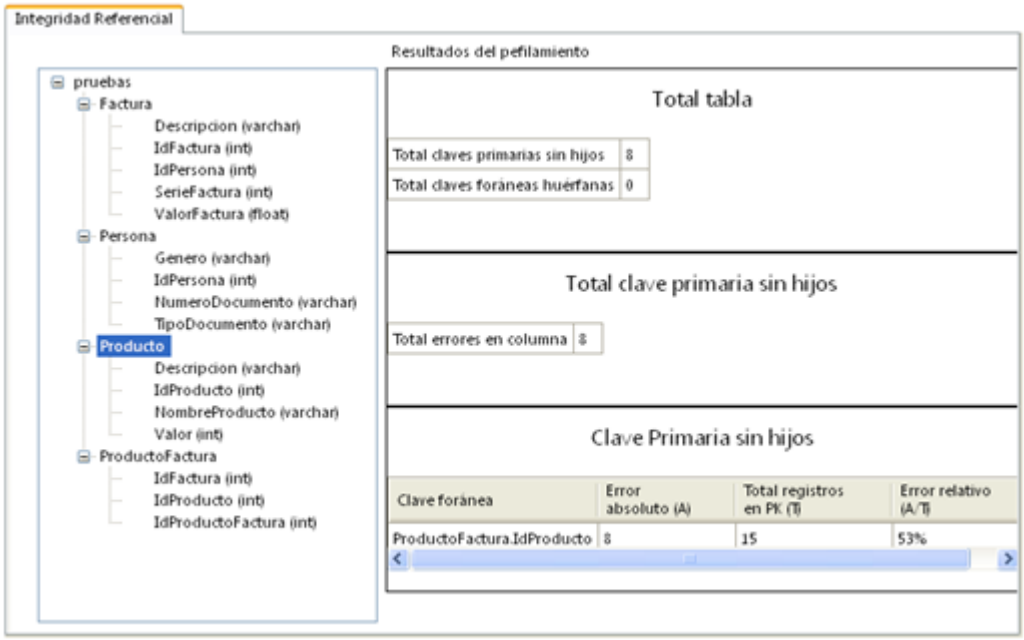

Fig. 20

CAPÌTULO 9
DIAGRAMAS

9.1. Diagrama de clases

9.1.1. Acceso BD

Este diagrama muestra las clases generadas por el DBML empleado para acceder a la base de datos de la aplicación

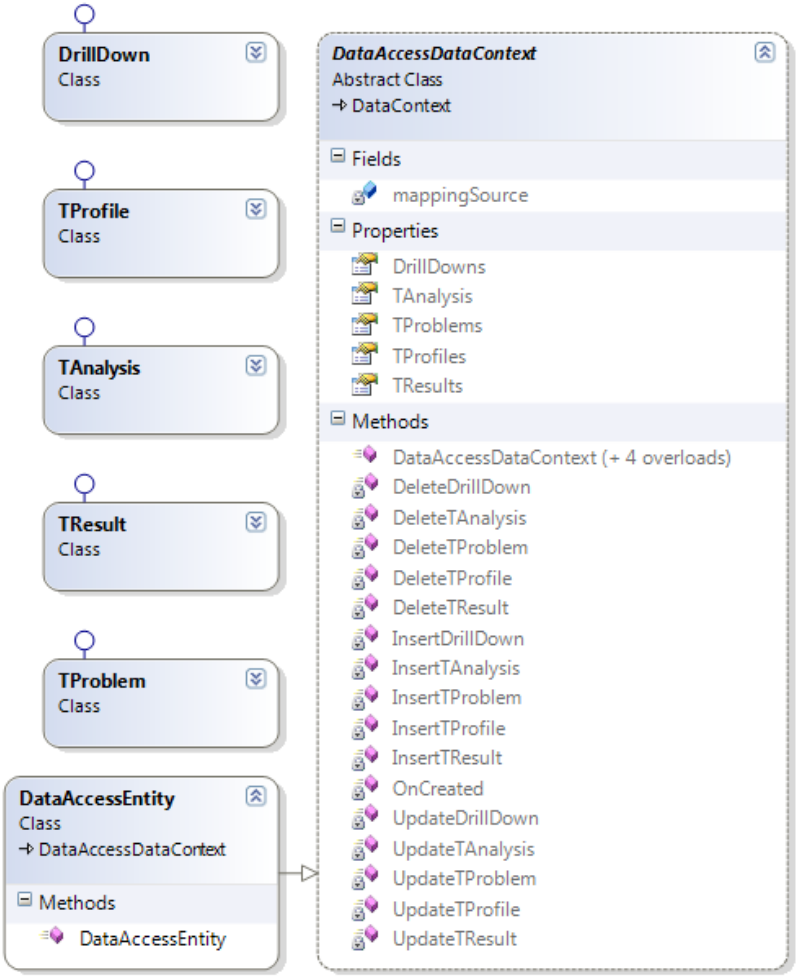

Diagrama 1

Este diagrama representa las clases que definen las operaciones que se pueden efectuar sobre las entidades en la base de datos de la aplicación.

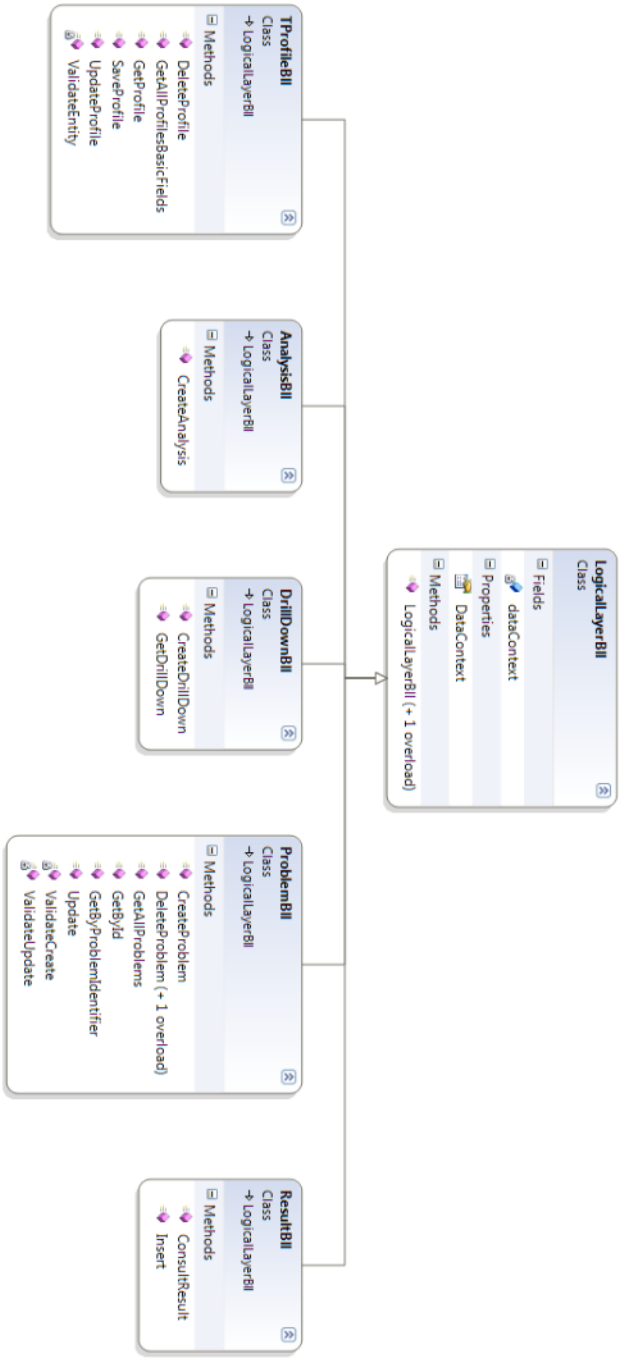

Diagrama 2

Este diagrama presenta la clase base de la cual se debe heredar para crear una nueva interface de configuración de los string de conexión a una base de datos y las interfaces gráficas actuales.

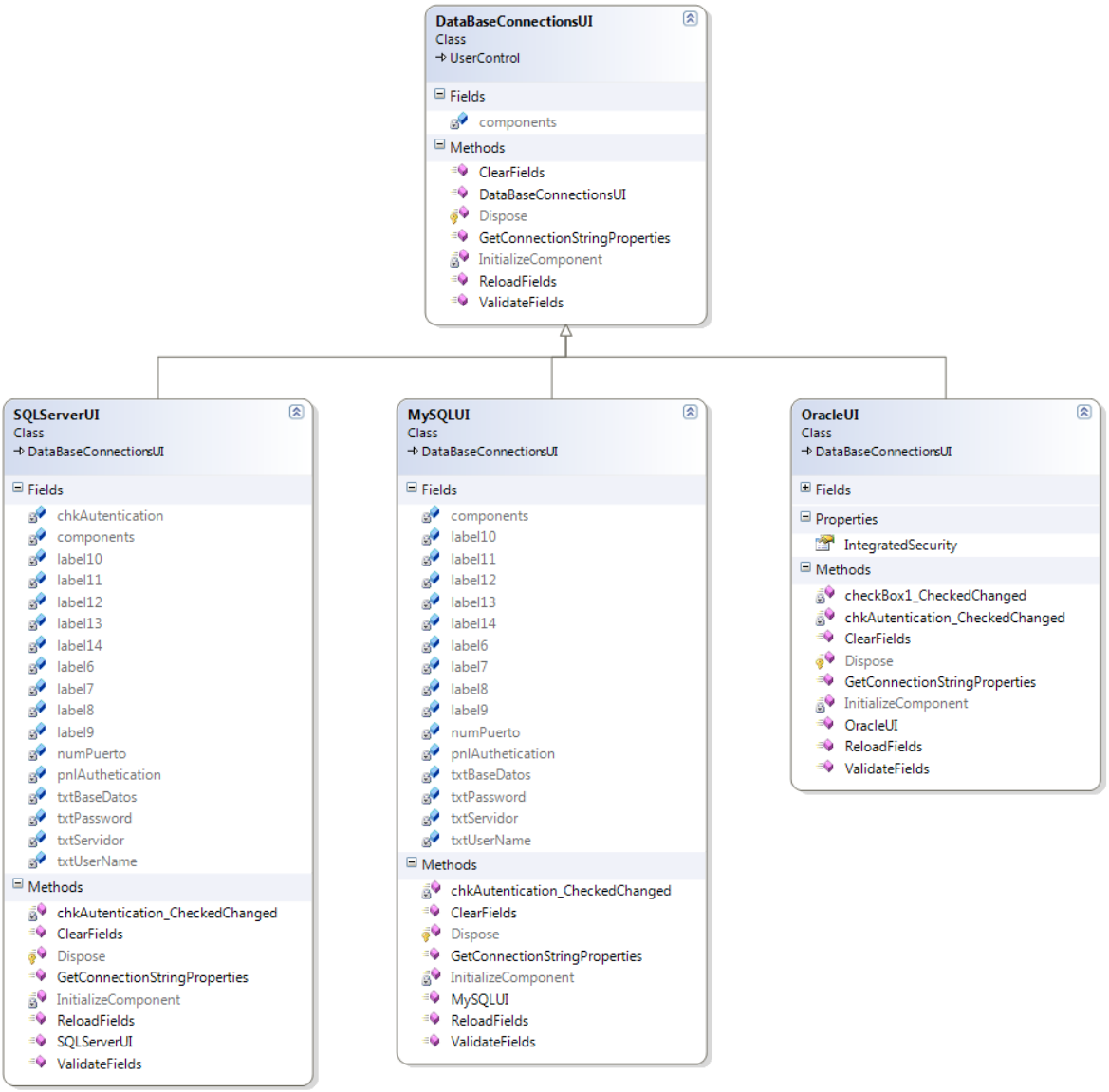

Diagrama 3

Este diagrama presenta las clases encargadas de administrar el acceso a las bases de datos de los clientes. También se muestra la clase Factory que retorna la instancia adecuada según el motor de base de datos recibido como parámetro.

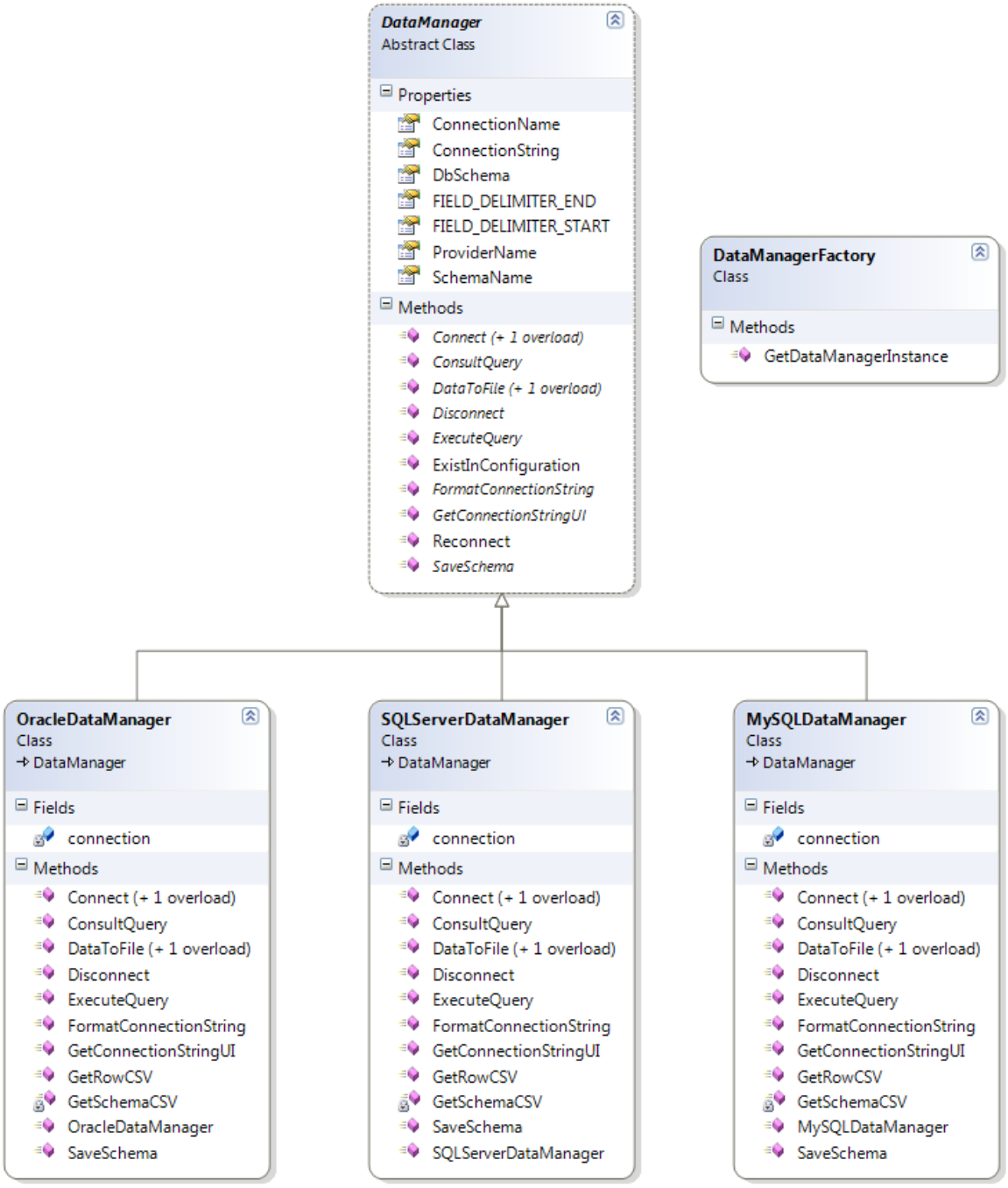

Diagrama 4

Este diagrama define las clases usadas por la aplicación para representar el esquema de una bases de datos.

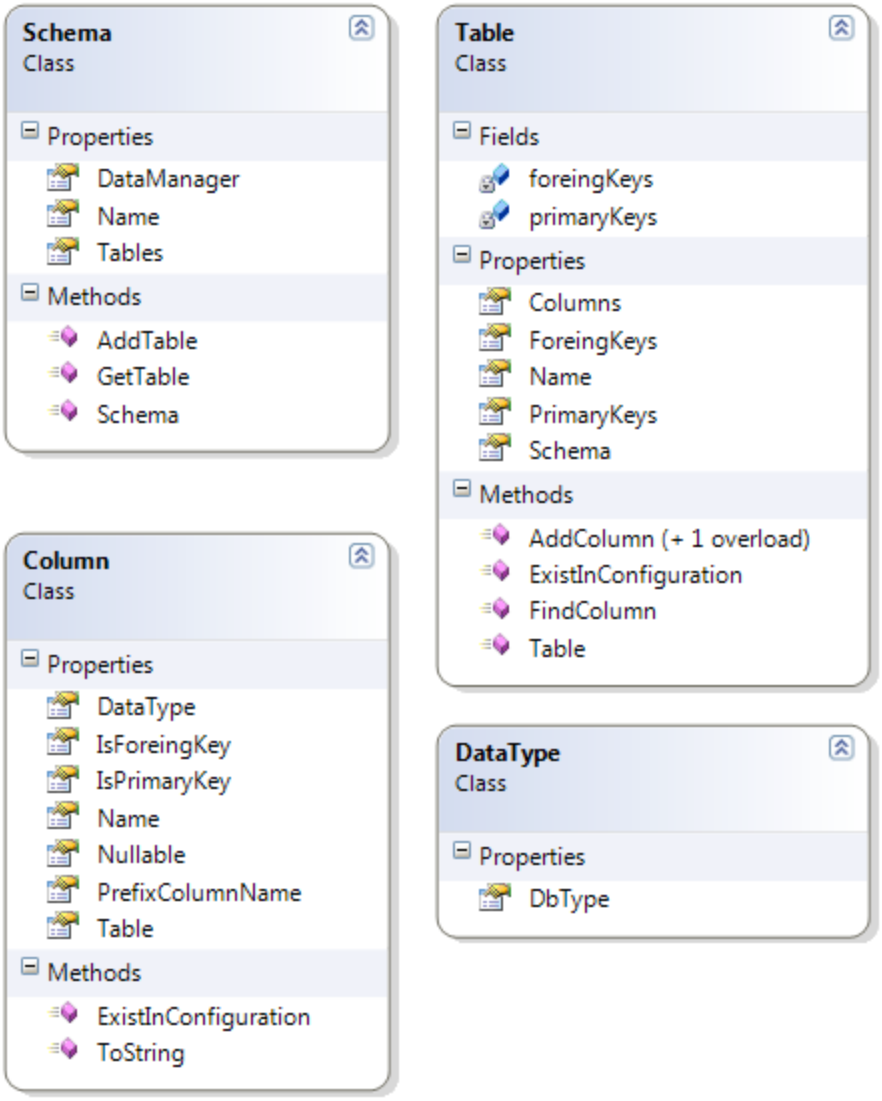

Diagrama 5

9.1.2. Analizador

Este diagrama contiene las clases base para los nuevos algoritmos, reportes y problemas que se deseen agregar.

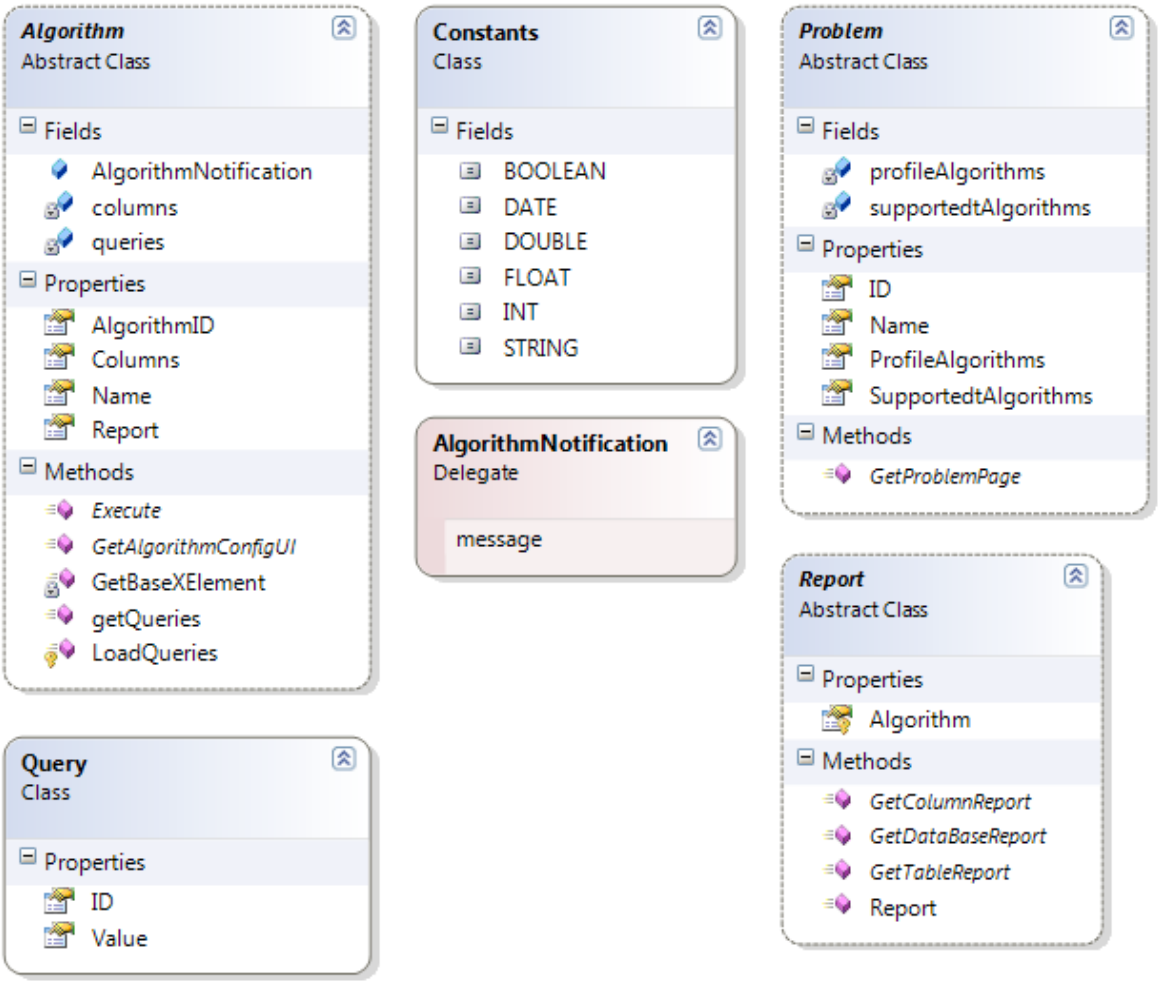

Diagrama 6

9.1.3. Configuración

Este diagrama presenta la clase que funciona como medio de comunicación entre los diferentes módulos.

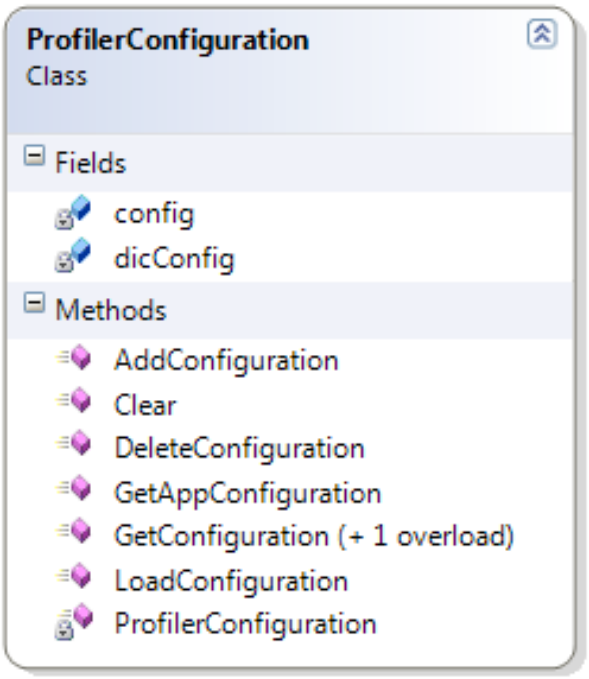

Diagrama 7

9.1.4. Gráficos

Este diagrama define un conjunto de controles utilizados en la presentación de resultados de análisis.

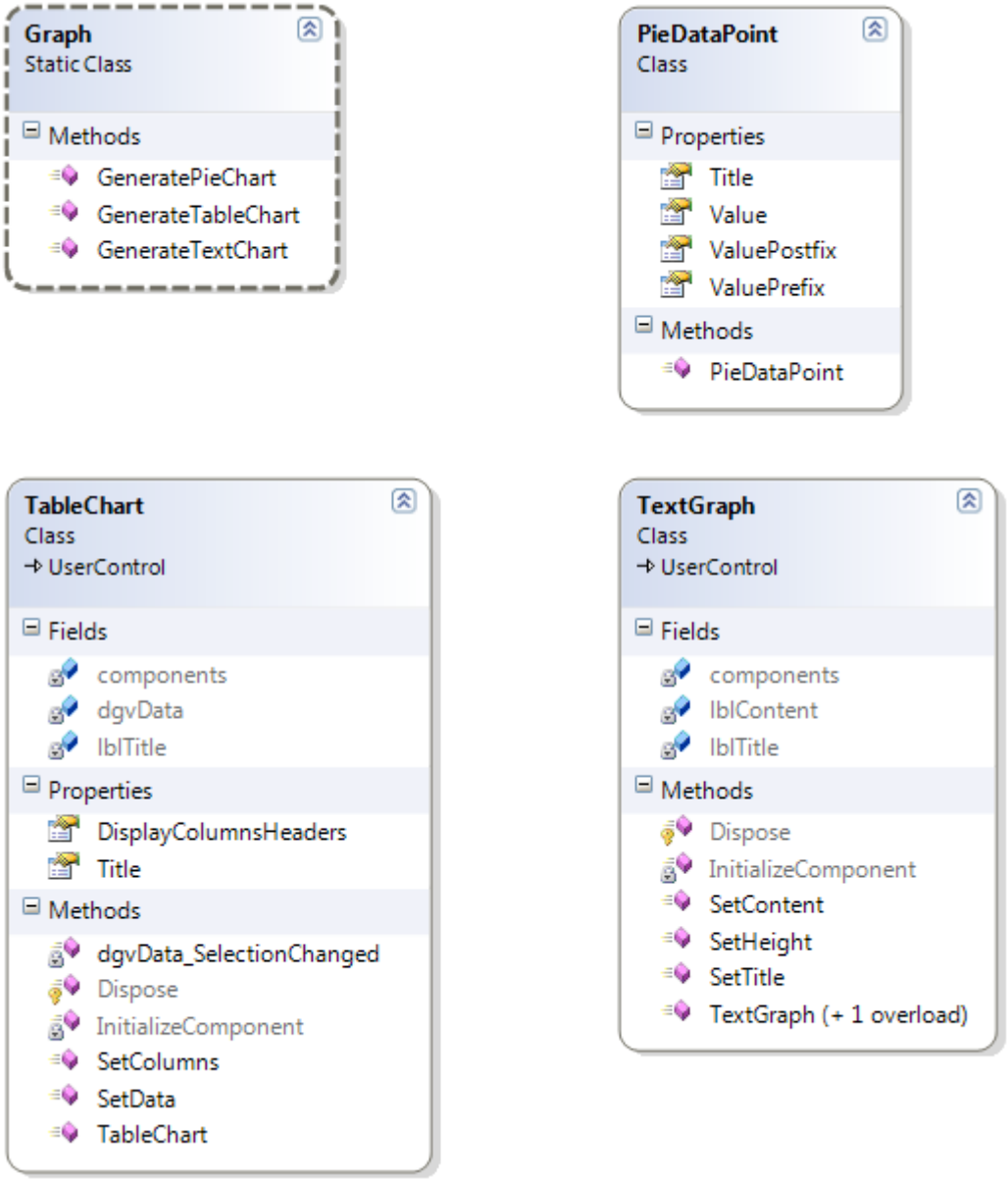

Diagrama 8

9.1.5. Utilidades

Este diagrama presenta clases con operaciones comunes y que no hacen parte necesariamente de la naturaleza de las demás clases.

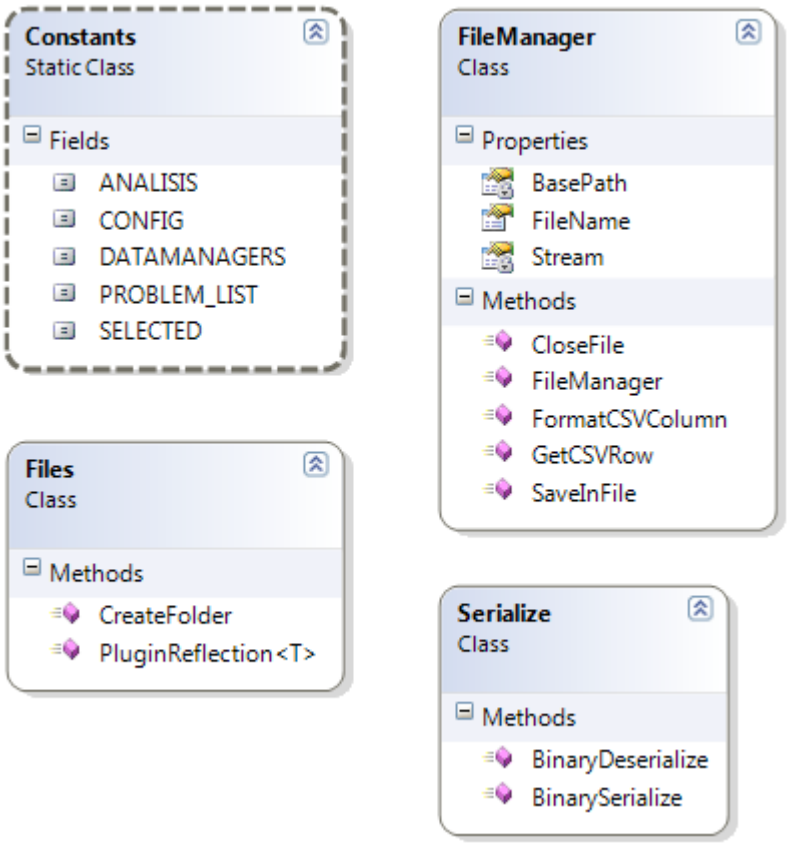

Diagrama 9

9.1.6. Nulos

Este diagrama presenta las clases involucradas en la implementación del problema de Nulos.

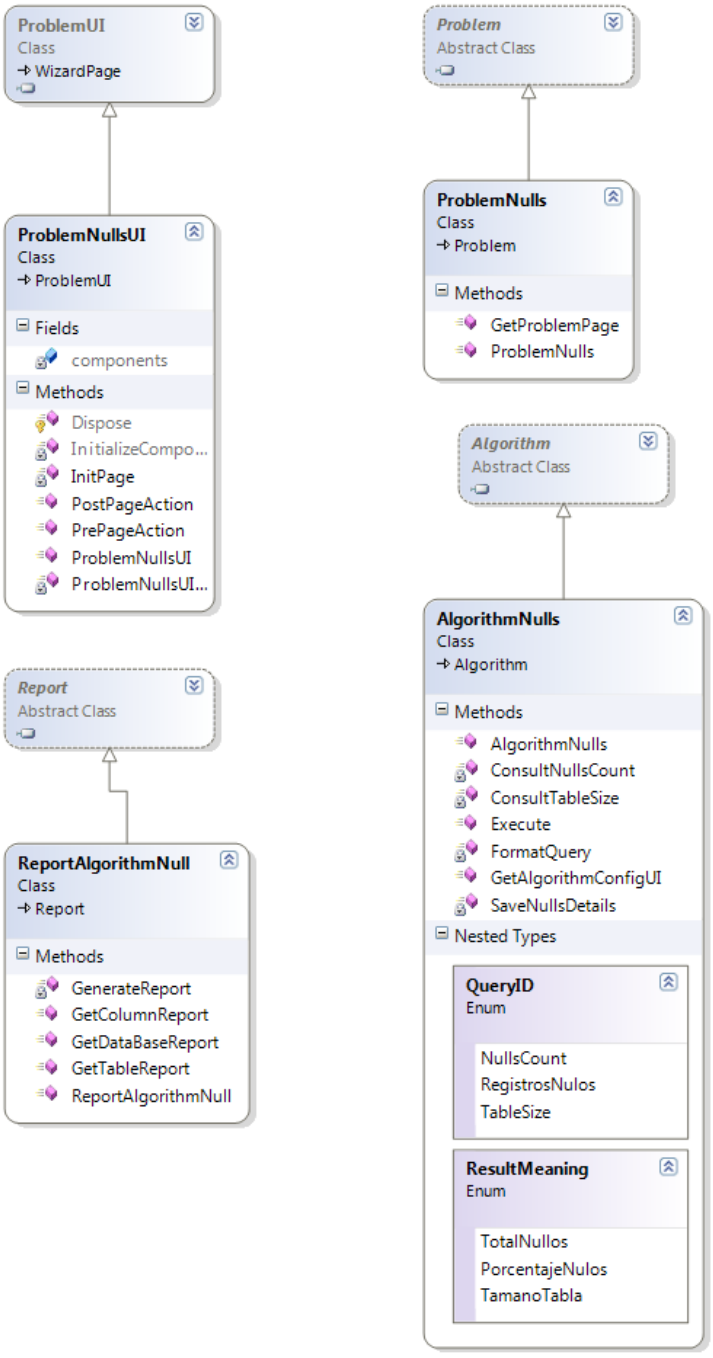

Diagrama 10

9.1.7. Llave primaria

Este diagrama presenta las clases involucradas en la implementación del problema de Llave primaria.

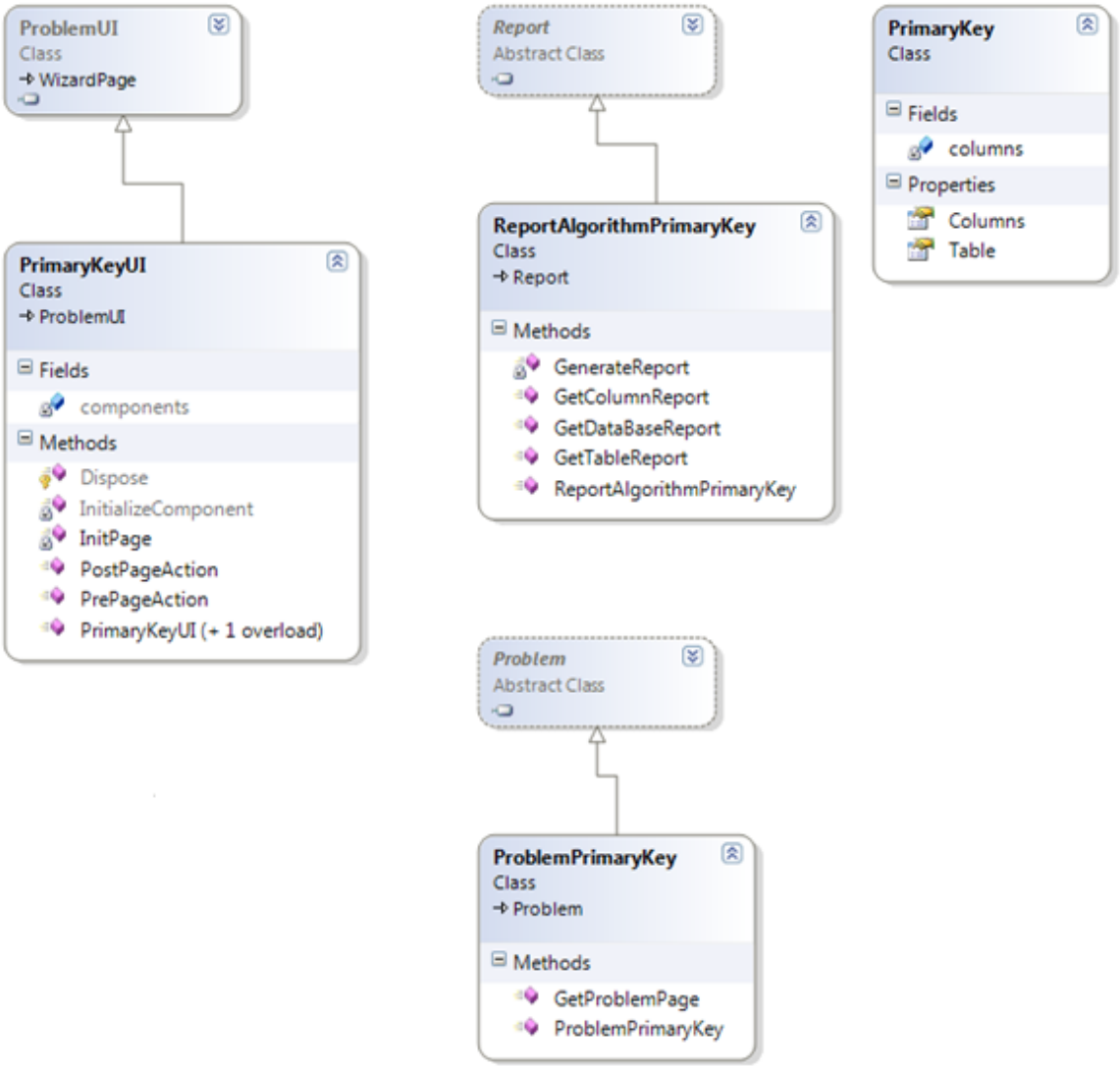

Diagrama 11

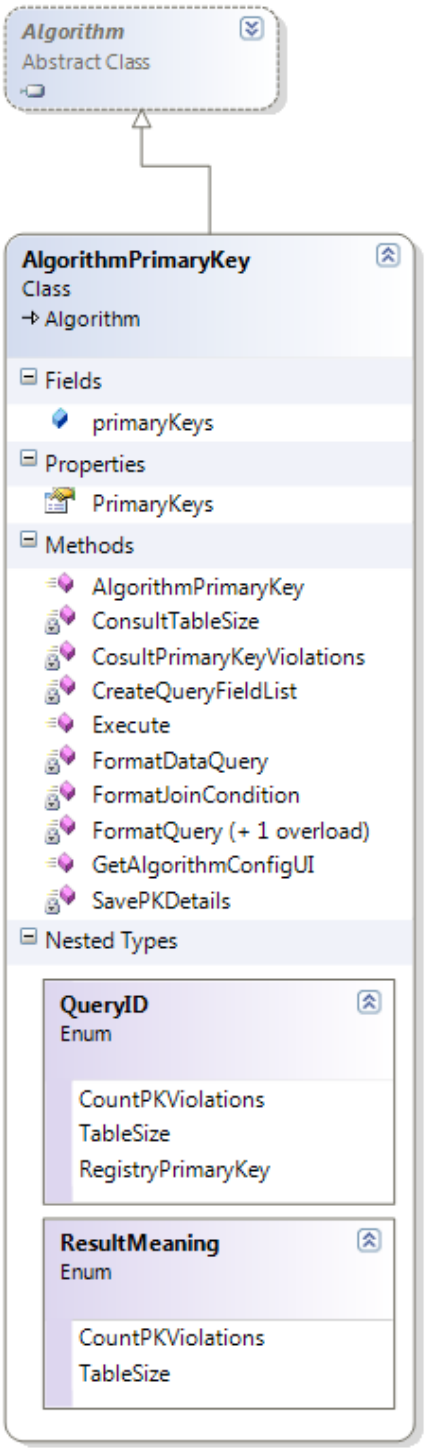
Algorithm
Abstract Class
AlgorithmPrimaryKey
Class
Algorithm
Fields
primaryKeys
Properties
PrimaryKeys
Methods
AlgorithmPrimaryKey
ConsultTableSize
CosultPrimaryKeyViolations
CreateQueryFieldList
Execute
FormatDataQuery
FormatJoinCondition
FormatQuery (+ 1 overload)
GetAlgorithmConfigUI
SavePKDetails
Nested Types
QueryID
Enum
CountPKViolations
TableSize
RegistryPrimaryKey
ResultMeaning
Enum
CountPKViolations
TableSize

Diagrama 12

9.1.8. Integridad referencial

Este diagrama presenta las clases involucradas en la implementación del problema de Integridad referencial.

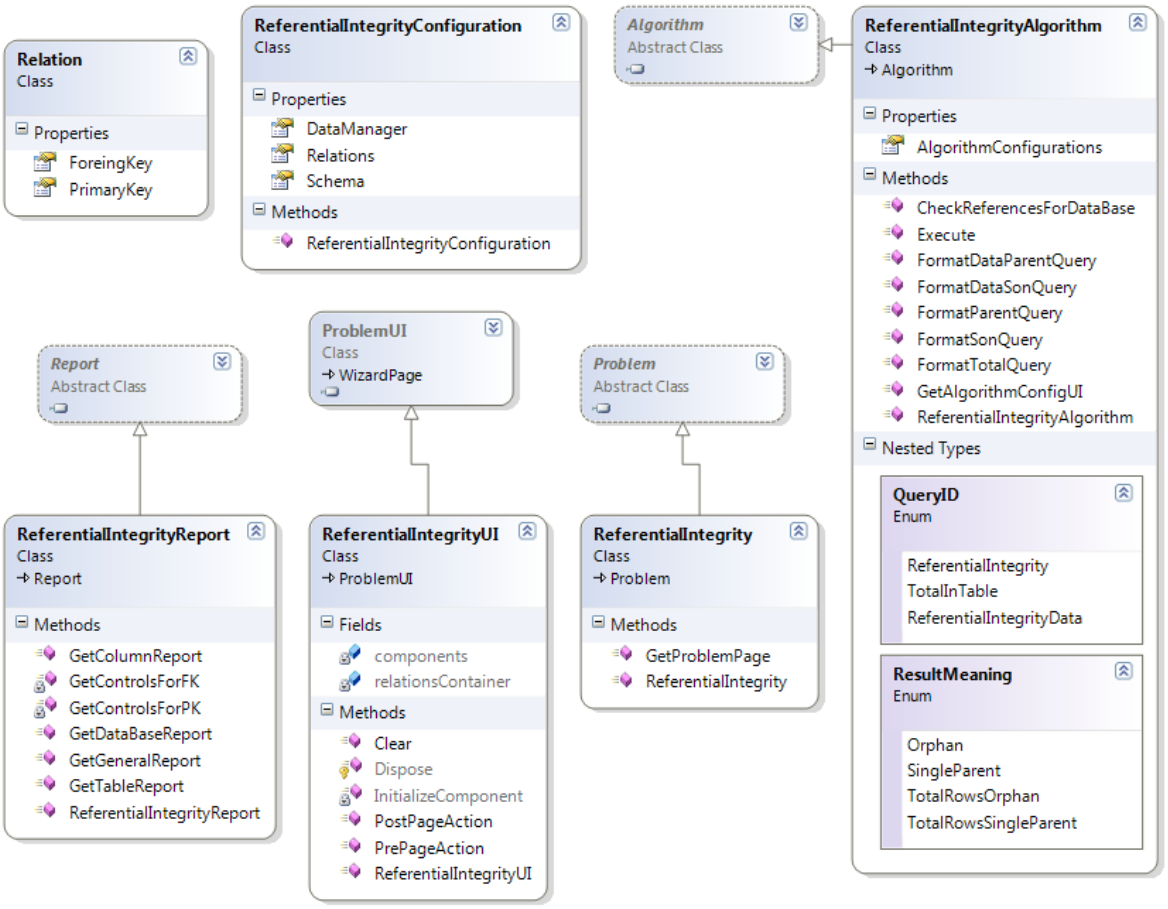

Diagrama 13

9.1.9. UI

Este diagrama presenta las clases que componen la interfaz gráfica del wizard.

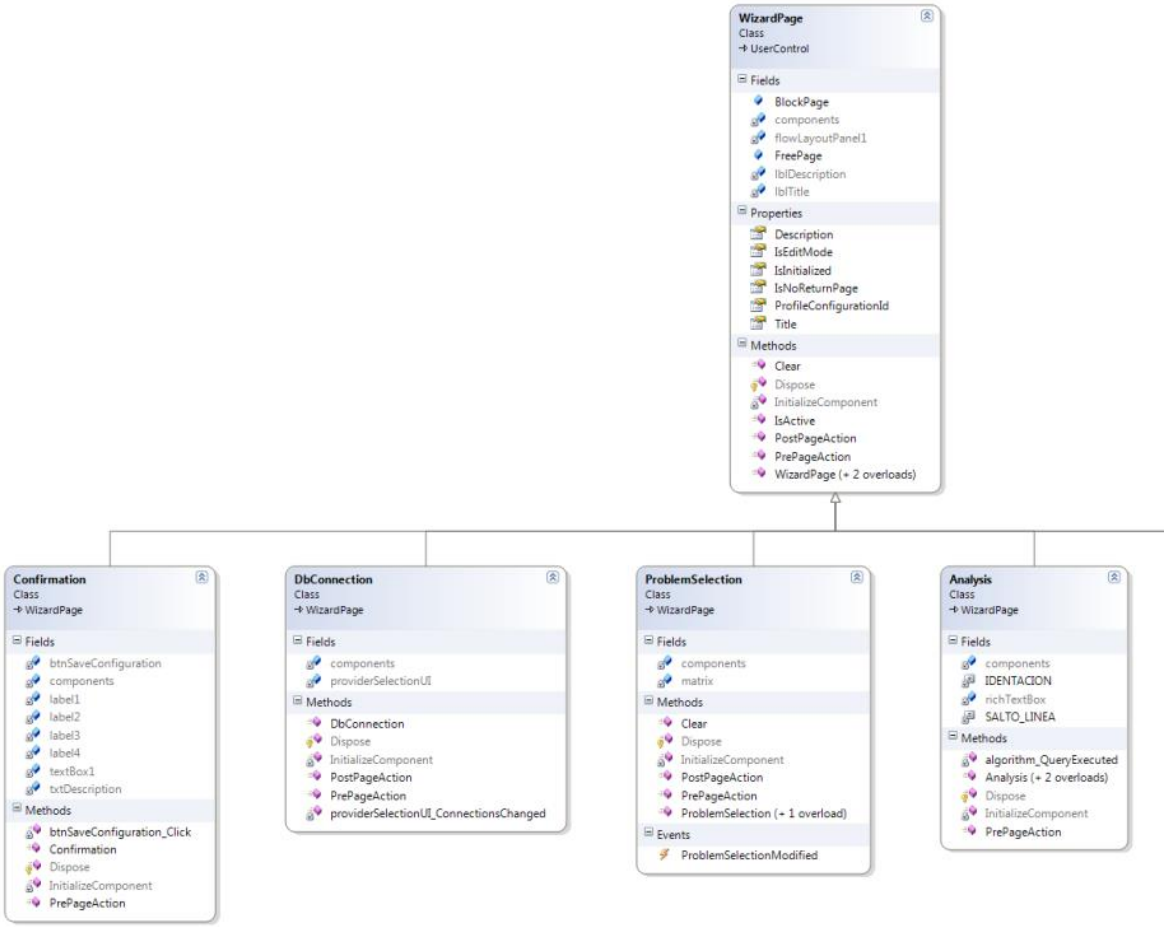

Diagrama 14

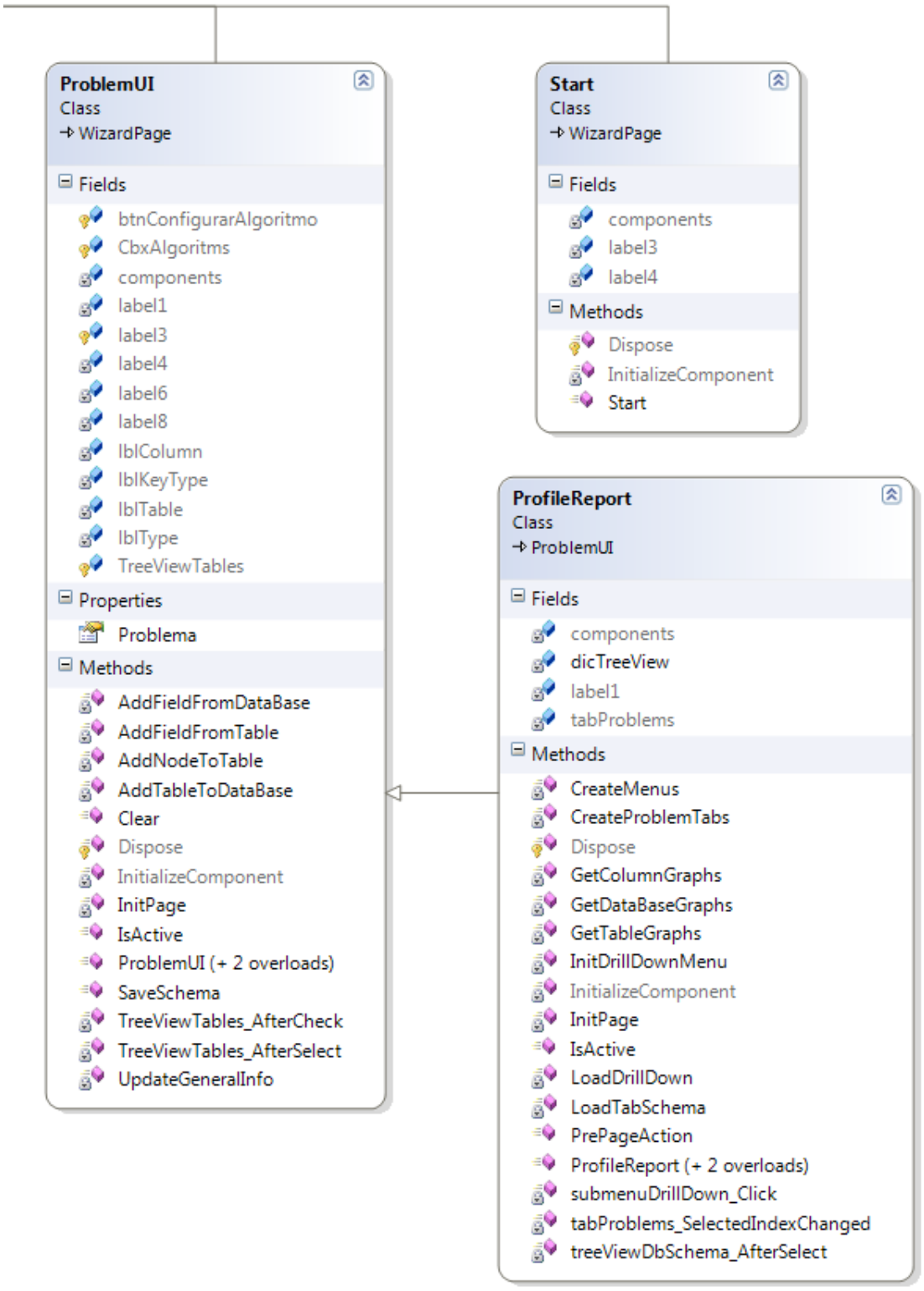
ProblemUI
Class
WizardPage
Fields
btnConfigurarAlgoritmo
CbxAlgoritms
components
label1
label3
label4
label6
label8
lblColumn
lblKeyType
lblTable
lblType
TreeViewTables
Properties
Problema
Methods
AddFieldFromDataBase
AddFieldFromTable
AddNodeToTable
AddTableToDataBase
Clear
Dispose
InitializeComponent
InitPage
IsActive
ProblemUI (+ 2 overloads)
SaveSchema
TreeViewTables_AfterCheck
TreeViewTables_AfterSelect
UpdateGeneralInfo
Start
Class
WizardPage
Fields
components
label3
label4
Methods
Dispose
InitializeComponent
Start
ProfileReport
Class
ProblemUI
Fields
components
dicTreeView
label1
tabProblems
Methods
CreateMenus
CreateProblemTabs
Dispose
GetColumnGraphs
GetDataBaseGraphs
GetTableGraphs
InitDrillDownMenu
InitializeComponent
InitPage
IsActive
LoadDrillDown
LoadTabSchema
PrePageAction
ProfileReport (+ 2 overloads)
submenuDrillDown_Click
tabProblems_SelectedIndexChanged
treeViewDbSchema_AfterSelect

Diagrama 15

9.2. Diagrama de actividades

9.2.1. Proceso general

Este diagrama presenta a modo general las actividades que realiza nuestra aplicación para efectuar un perfilamiento.

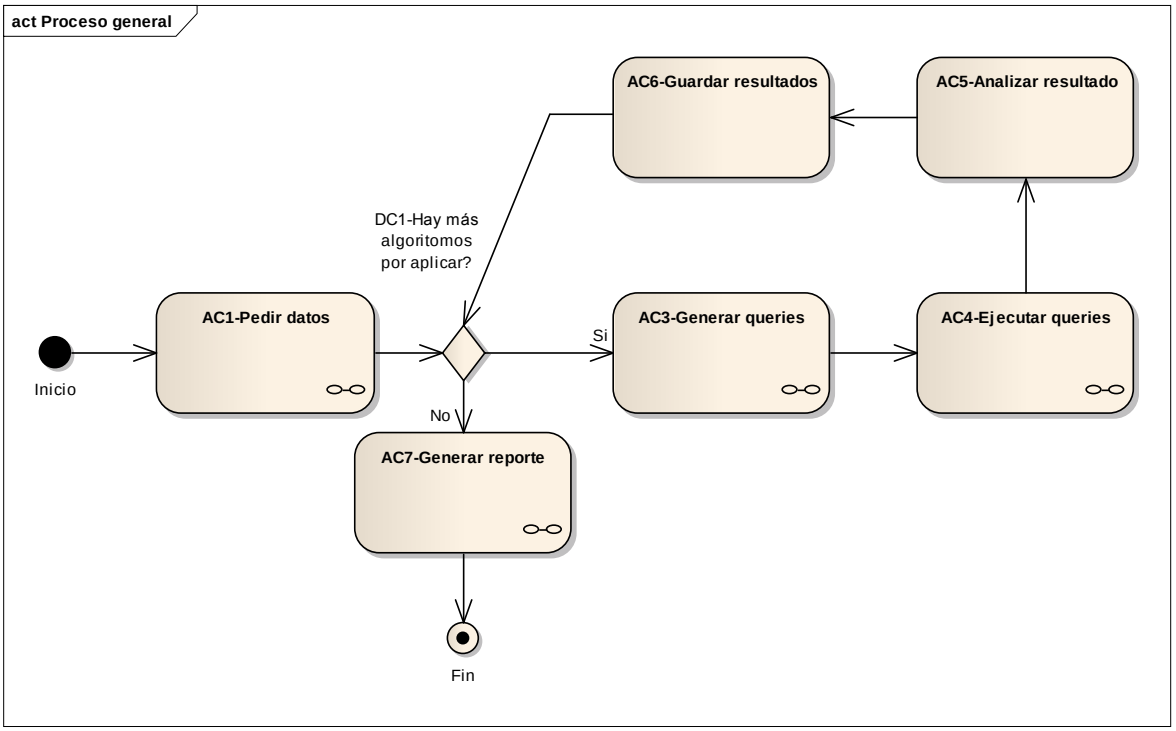

Diagrama 16

9.2.2. AC1-Pedir datos

Este diagrama presenta las actividades que realiza nuestra aplicación para efectuar la configuración de un perfilamiento.

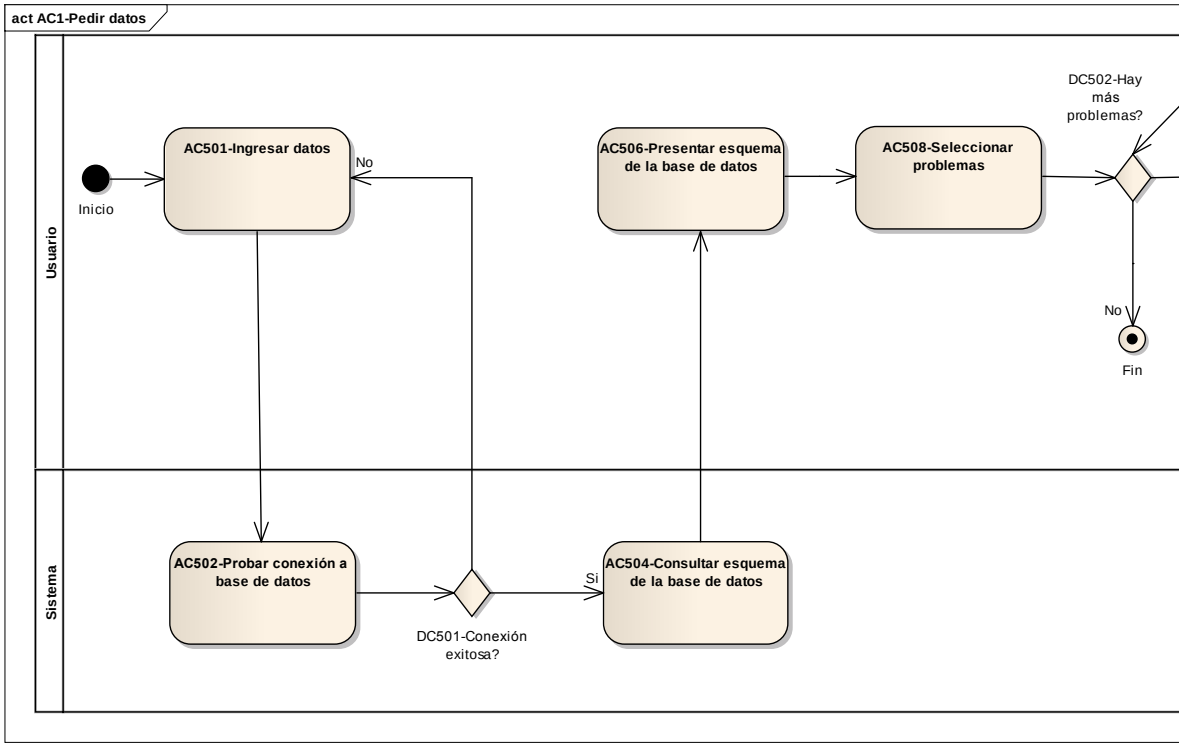

Diagrama 17

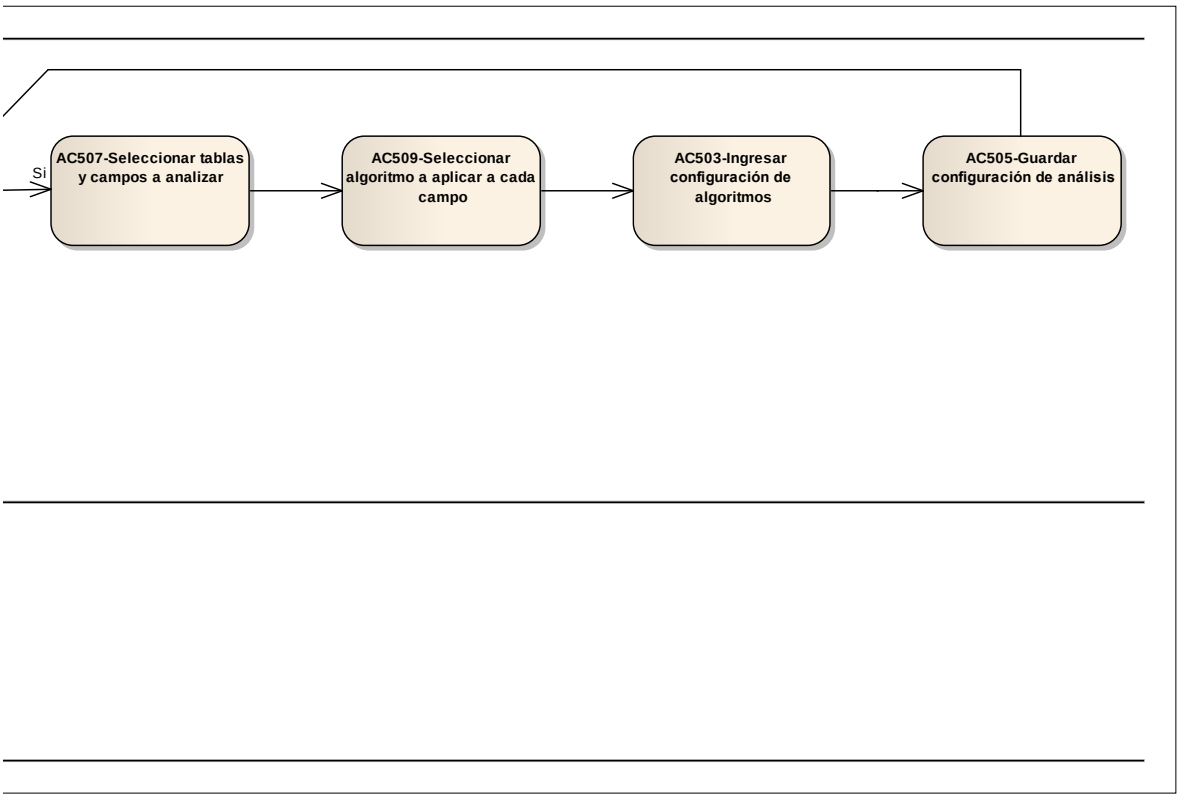

Diagrama 18

9.2.3. AC3-Generar queries

Este diagrama presenta las actividades incluidas en la actividad AC3-Generar queries del diagrama en la sección 9.2.1

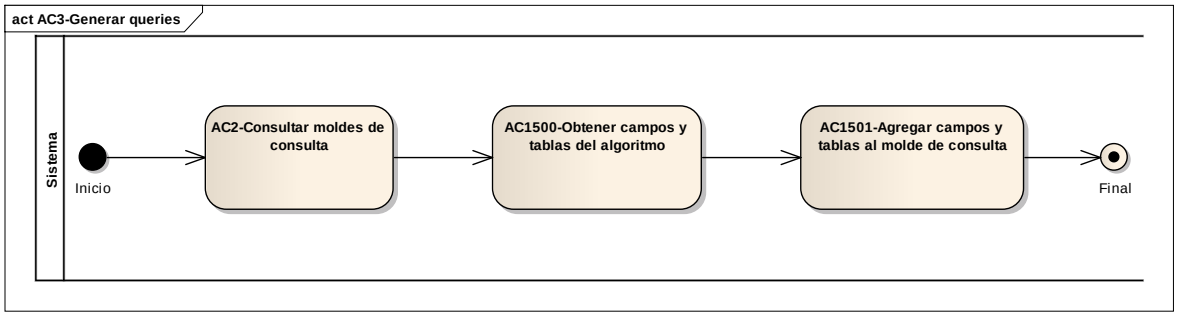

Diagrama 19

9.2.4. AC4-Ejecutar queries

Este diagrama presenta las actividades incluidas en la actividad AC4-Ejecutar queries del diagrama en la sección 9.2.1

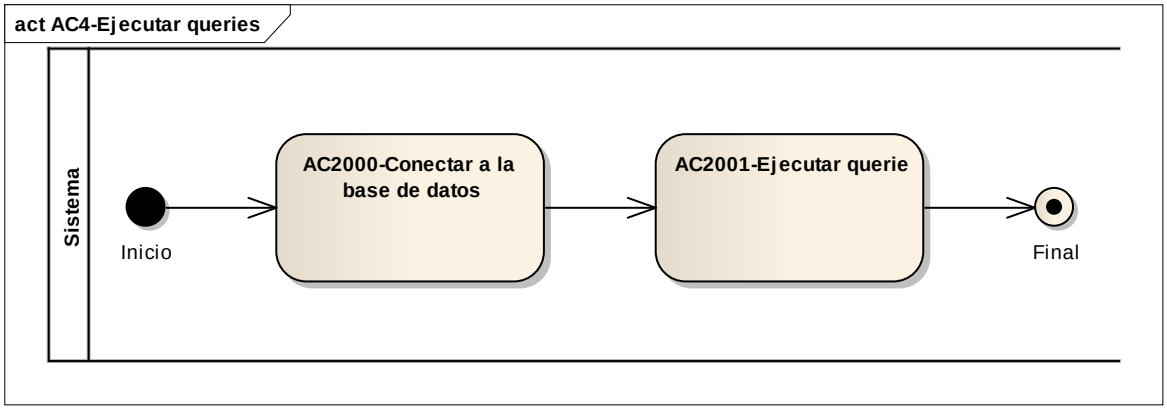

Diagrama 20

9.2.5. AC7-Generar reporte

Este diagrama presenta las actividades incluidas en la actividad AC7-Generar reporte del diagrama en la sección 9.2.1

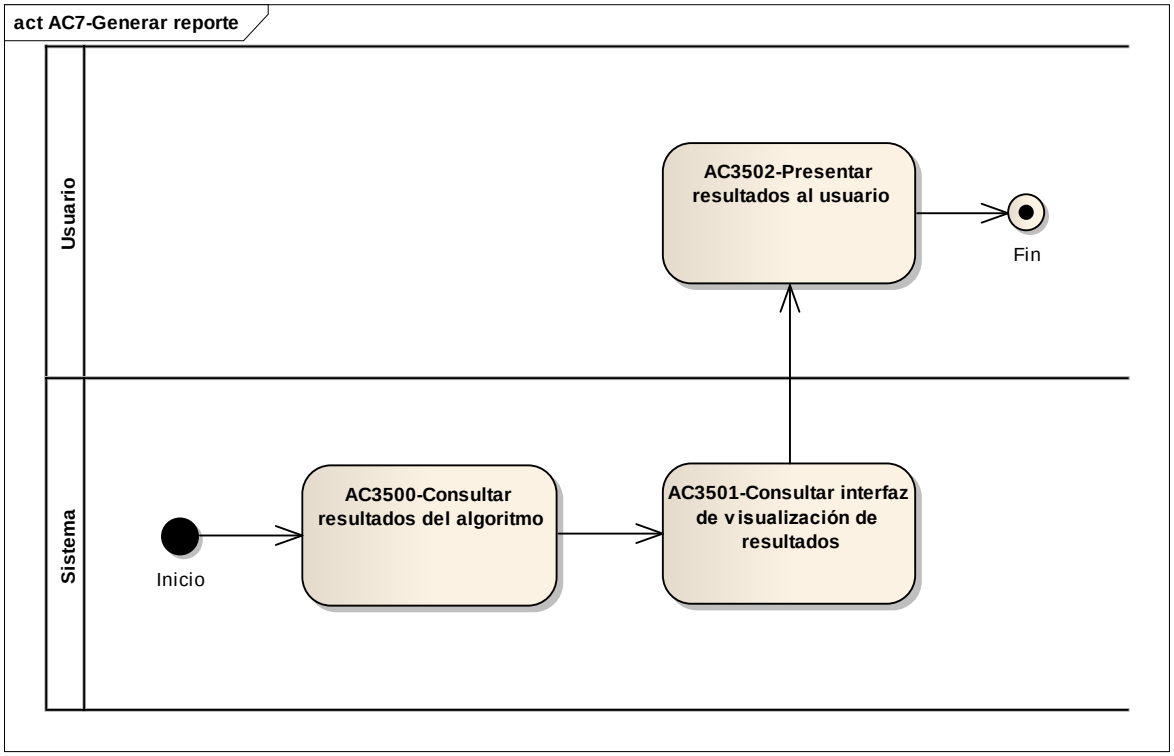

Diagrama 21

CAPÍTULO 10
CONCLUSIONES

Se logró desarrollar, diseñar y probar una herramienta de software que determina la calidad de los datos de una base de datos relacional (MySql, SQLServer y Oracle) y que además permite la detección de tres problemas de calidad de datos (falta de integridad referencial, violación llave primaria y nulos), elaborando un perfil de múltiples atributos simultáneamente entregando resultados particulares y globales acerca de la base de dato, como se puede constatar en los capítulos 7 Arquitectura, 8 Pruebas y 9 Diagramas. Esto es de gran importancia ya que puede hacer más eficiente un proceso de análisis de calidad, sobre todo cuando las fuentes poseen un gran número de entidades y atributos.

En cuanto a los objetivos específicos “Identificar los principales problemas en calidad de datos” e “Identificar algoritmos para detección de errores e inconsistencias en los datos”, fue posible detectar diferentes tipos de problemas que afectan la calidad de los datos como los presentados en el capítulo 2. 3 Problemas en las bases de datos, sin embargo, no fue posible identificar un conjunto de problemas prioritarios ya que esta importancia está determinada por los intereses del negocio y de las personas que usan dicha información para la toma de decisiones tal como se expone en [1]. Por tal motivo se seleccionaron tres problemas como foco de nuestro proyecto, a los que fue posible identificar los algoritmos de detección descritos en el capítulo 7.8 Problemas soportados, los cuales sirvieron de insumo para la construcción de la aplicación.

Se identificaron metodologías para determinar el estado de la calidad de los datos de una base de datos como se describe en el capítulo 4 Metodología de Perfilamiento. Con éstas fue posible desarrollar una herramienta que se enmarca

en la metodología de calidad de datos propuesta por Darián Pérez y José Alberto Vilalta enunciada en el capítulo mencionado, donde se propone una selección de campos a analizar, una etapa de toma de mediciones y por último el uso de herramientas gráficas para un posterior análisis. De ésta forma fue posible cumplir con el objetivo específico “Identificar metodologías existentes para determinar el estado de la calidad de los datos de una base de datos”.

BIBLIOGRAFÍA

[1] H. Müller and J.-C. Freytag, "Problems, Methods, and Challenges in Comprehensive Data Cleansing".

[2] P. Oliveira, F. Rodrigues, P. Henriques and H. Galhardas, "A Taxonomy of Data Quality Problems," Porto.

[3] J. L. Harrington, Relational Database Design and Implementation: Clearly Explained, Morgan Kaufmann, 2009.

[4] W. Eckerson, "Data Profiling: A Tool Worth Buying (Really!).," 2004.

[5] L. P. English, Information quality applied : best practices for improving business information, processes, and systems., 2009.

[6] D. Loshin, Master Data Management, Morgan Kaufmann, 2009.

[7] H. H. Do and E. Rahm, "Data Cleaning: Problems and Current Approaches," Germany.

[8] T. N. Herzog, W. E. Winkler and F. J. Scheuren, Data Quality and Record Linkage Techniques, Washington, EE.UU: Springer, 2007.

[9] "Data Ladder," [Online]. Available: http://www.dataladder.com/. [Accessed 4 Abril 2011].

[10] "Data Cleaner," [Online]. Available: http://datacleaner.eobjects.org/docs. [Accessed Abril 2011].

[11] "Datiris," [Online]. Available: http://www.datiris.com/features.shtml. [Accessed 4 Abril 2011].

[12] "DQ Global," [Online]. Available: http://www.dqglobal.com/deduplication-software#twoj_fragment1-2. [Accessed 4 Abril 2011].

[13] "Help It Systems," [Online]. Available: http://www.helpit.com/us/. [Accessed 4 Abril 2011].

[14] "Trillium Software," [Online]. Available: http://www.trilliumsoftware.com/home/solutions/enterprisedq/dataprofiling.aspx. [Accessed 4 Abril 2011].

[15] I. A. Uribe, "Guía Metodológica para la selección de técnicas de depuración de datos," Medellín, 2012.

[16] J. Paniagua and J. F. Mira, Diagnóstico de la calidad de la base de datos de la Clínica Universitaria Bolivariana, Medellín: Tesis. UPB. Facultad de Ingeniería Informática y Telecomunicaciones., 2011.

[17] R. Patiño, Caracterización de técnicas para detección de strings duplicados, Medellín, 2009.

[18] C. Muñoz, Caracterización de técnicas para detección de valores faltantes, Medellín: Tesis. UPB. Facultad de Ingeniería Informática, 2010.

[19] C. M. Escobar, Caracterización de técnicas para detección de valores extremos, Medellín : Tesis. UPB. Facultad de Ingeniería Informática., 2010.

[20] J. D. Zuluaga, Algoritmo fonético para detección de cadenas de texto duplicadas, Medellín: Tesis. UPB. Facultad de Ingeniería Informática, 2010.

[21] J. D. Velásquez, Caracterización de metodologías para limpieza de datos, Medellín : Tesis. UPB. Facultad de Ingeniería Informática, 2009.

[22] I. A. U. Esp. and C. J. Ramírez, "Hacia una Metodología para la Selección de," vol. 6, no. 1, 2009.

[23] I. Amón and C. Jiménez, "Funciones de Similitud sobre Cadenas de Texto: Una Comparación Basada en la Naturaleza de los Datos," 2010.

[24] C. Batini, D. Barone, F. Cabitza and S. Grega, "A Data Quality Methodology for Heterogenenus Data," *International Journal of Database Management Systems,* pp. 60-79, 2011.

[25] L. Pipino, Y. Lee and R. Wang, "Data Quality Assessment," *Communications of the ACM,* pp. 211-218, 2002.

[26] D. Pérez Rodríguez and J. A. Vilalta Alonso, "El Procedimiento de Diagnóstico de la Calidad de los Datos con Enfoque de Riesgos," Ciudad de

La Habana, 2010.

[27] M. Bobrowski, M. Marré and D. Yankelevich, "Measuring Data Quality," Buenos Aires.

[28] M. Scannapieco, A. Virgillito, C. Marchetti, M. Massimo and R. Baldoni, "The DaQuinCIS Architecture: a Platform for Exchanging and Improving Data Quality in Cooperative Information Systems," [Online]. Available: http://www.dis.uniroma1.it/~midlab/articoli/SVMMB_IS.pdf. [Accessed Julio 2011].

[29] J. A. Vilalta Alonso and M. Espinosa Álvarez Buylla, "Metodología para el diagnóstico de la calidad de los datos," 2008.

[30] L. Etcheverry, S. Tercia, A. Marotta and V. Peralta, *Medición de la Exactitud de Datos en Sistemas Fuentes: Un caso de Estudio,* 2007.

[31] D. Loshin, "Populating a Data Quality Scorecard with Relevant Metrics".

[32] "OPENUP. Introductionto OpenUP (Open UnifiedProcess).," [Online]. Available: http://www.eclipse.org/epf/general/OpenUP.pdf. [Accessed 14 Mayo 2011].

[33] "OPENUP. CorePrinciples.," [Online]. Available: http://epf.eclipse.org/wikis/openup/publish.openup.base/customcategories/core_principles_category_B7B55076.html?nodeId=e499564e. [Accessed 14 Mayo 2011].

[34] "OPENUP. Concept:Phase Milestones," [Online]. Available: http://epf.eclipse.org/wikis/openup/practice.mgmt.risk_value_lifecycle.base/guidances/concepts/phase_milestones_5678231E.html?nodeId=d37c00ba. [Accessed 14 Mayo 2011].

[35] "OPENUP. Concept:Elaboration Phase," [Online]. Available: http://epf.eclipse.org/wikis/openup/practice.mgmt.risk_value_lifecycle.base/guidances/concepts/elaboration_phase_BE880435.html?nodeId=2a75adb1. [Accessed 14 Mayo 2011].

[36] "OPENUP. Concept:Construction Phase," [Online]. Available: http://epf.eclipse.org/wikis/openup/practice.mgmt.risk_value_lifecycle.base/g

uidances/concepts/construction_phase_873B6559.html?nodeId=e0607818. [Accessed 14 Mayo 2011].

[37] "OPENUP. Concept: Transition Phase," [Online]. Available: http://epf.eclipse.org/wikis/openup/practice.mgmt.risk_value_lifecycle.base/guidances/concepts/transition_phase_DD5986E5.html?nodeId=2fb2aee9. [Accessed 14 Mayo 2011].

[38] [Online]. Available: http://somee.com/. [Accessed 30 Julio 2012].

[39] [Online]. Available: http://tortoisesvn.net/. [Accessed 30 Julio 2012].

[40] [Online]. Available: https://www.assembla.com/home. [Accessed 25 Agosto 2012].

[41] "etraxis," [Online]. Available: https://www.etraxis.com/. [Accessed 30 Julio 2012].

[42] "Microsoft," [Online]. Available: http://msdn.microsoft.com/en-us/library/zw4w595w. [Accessed 30 Julio 2012].

[43] "Codeplex," [Online]. Available: http://dbschemareader.codeplex.com/. [Accessed 25 Agosto 2012].

[44] C. Ordonez and J. García-García, "Referential integrity quality metrics," *Decision Support Systems,* vol. 44, p. 495–508, 2008.

[45] "OPENUP. Whatis OpenUP?," [Online]. Available: http://epf.eclipse.org/wikis/openup/index.htm. [Accessed 14 Mayo 2011].

[46] "OPENUP. Concept:InceptionPhase," [Online]. Available: http://epf.eclipse.org/wikis/openup/practice.mgmt.risk_value_lifecycle.base/guidances/concepts/inception_phase_C4456871.html?nodeId=d5fbbd8b. [Accessed 14 Mayo 2011].

[47] M. Gertz, T. Özsu, G. Saake and K.-U. Sattler, "Data Quality on the Web," 2003.

[48] "MSDN," [Online]. [Accessed 31 07 2012].

[49] Y. J. Kim, R. Kishore and G. L. Sanders, "From DQ to EQ: understanding data quality in the context of e-business systems," [Online]. Available: http://web.ebscohost.com/ehost/pdfviewer/pdfviewer?vid=3&hid=11&sid=76ffa912-8bd9-4c83-8983-5d0352490c14%40sessionmgr4. [Accessed 27 julio 2012].

[50] D. A. Correa, Desarrollo de software para administración de sistemas basados en controladores lógicos programables, Medellín, 2009.

[51] M. Scannapieco, M. Mecella, T. Catarci, C. Cappiello, B. Pernici, F. Mazzoleni and F. Stella, "Proposal, DL3: Comparative Analysis of the Proposed Methodologies for Measuring and Improving Data Quality and Description of an Integrated".