Desarrollo de un prototipo de software para la automatización del diseño de un esquema de bases de datos relacional

Alexander Lozano Forero
Código: 20161099031

Yudy Lizeth Valbuena Montenegro
Código: 20161099044

Universidad Distrital Francisco José de Caldas
Facultad de Ingeniería
Especialización En Ingeniería De Software
Bogotá D.C. 2016

Índice general

I Descripción del Proyecto **13**

1. Contextualización de la Investigación **15**

1.1. Introducción 15
1.2. Definición del Problema 15
1.3. Título 16
1.4. Hipótesis 16
1.5. Objetivos 17
1.5.1. Objetivo General 17
1.5.2. Objetivos Específicos 17
1.6. Justificación de la investigación 17
1.7. Alcances 17
1.8. Restricciones del prototipo 18
1.9. Resultados esperados 18
1.10. Cronograma de actividades 19
1.11. Marco Referencial 19
1.11.1. Marco teórico 19
1.11.1.1. Bases de datos relacionales 19
1.11.1.2. Dependencias Funcionales 20
1.11.1.3. Relación 21
1.11.1.4. Integridad de Entidad 21
1.11.1.5. Axiomas de Armstrong 22
1.11.1.6. Algoritmo de síntesis de Bernstein 22
1.11.1.7. Normalización de base de datos 22
1.11.1.8. Clave Candidata 23
1.11.1.9. Atributo extraño 23
1.11.1.10. Dependencia redundante 24
1.11.1.11. Metodología de desarrollo Scrum 24

II Desarrollo de la investigación **27**

2. Desarrollo de la investigación **29**

2.1. Gestión del proyecto 29
2.1.1. Lista priorizada de tareas del producto 29
2.1.2. Planeación de sprints 29
2.1.3. Ejecución de sprints 30
2.2. Fase de análisis 30
2.2.1. Stakeholders 30
2.2.2. Requerimientos funcionales 34
2.2.3. Requerimientos no funcionales 34
2.3. Caracterización del prototipo 34
2.3.1. Dependencias funcionales 34
2.3.2. Cálculo del cierre (algoritmo de Ullman) 35
2.3.3. Cálculo de recubrimiento mínimo 35
2.3.4. Cálculo de llaves candidatas 36
2.3.5. Algoritmo de síntesis de Bernstein 36
2.4. Fase de Diseño 36
2.4.1. Resultados del análisis 36
2.4.1.1. Cálculo de cierre de un descriptor 37
2.4.1.2. Cálculo del recubrimiento mínimo 37
2.4.1.3. Cálculo de llaves candidatas 37
2.4.1.4. Algoritmo de síntesis de Bernstein 37
2.5. Fase de Desarrollo 38
2.5.1. Diagrama de clases 38
2.5.2. Diagrama de Clases-entidades 38
2.5.3. Diagrama de clases de la interface gráfica de la aplicación 39
2.5.4. Diagrama de Secuencia 40
2.5.5. Diagrama de clases de implementación de la lógica en la aplicación 41
2.5.6. Diagrama de la estructura general de la aplicación 41
2.5.7. Diagrama comportamiento de la aplicación 42
2.6. Implementación 42
2.6.1. Documentación del Código 42
2.6.2. Casos de Prueba 42

III Cierre de la investigación 47

3. Cierre de la investigación 49
3.1. Resultados de la investigación 49
3.1.1. Caso práctico de verificación 49
3.1.1.1. Planteamiento del problema 49
3.1.1.2. Dependencias funcionales 49
3.1.1.3. Forma Normal 50
3.1.1.4. Sintesis (Algoritmo de Sintesis de Bernstein) . . 51
3.1.2. Aplicación del Caso de Estudio en el prototipo 52
3.1.2.1. Carga de la relación 52

3.1.2.2. Calculo de particiones (Algoritmo de sintesis de Bernstein) 54
3.1.2.3. Verificación de resultados 55
3.2. Conclusiones 56
3.3. Evaluación de los objetivos 56
3.4. Prospectivo del prototipo 56
3.4.1. Trabajos de investigación futuros 56
3.4.2. Contraste con el modelo pseudomatemático 56

Índice de figuras

1.1. Escenarios de la ingeniería de Software [Londoño, 2016, Pág 4] . 16
1.2. Metodologías Scrum [Metodologias agiles, 2016] 24

2.1. Gráfica de trabajo pendiente Sprint 1 31
2.2. Gráfica de trabajo pendiente Sprint 2 31
2.3. Gráfica de trabajo pendiente Sprint 3 32
2.4. Gráfica de trabajo pendiente Sprint 4 32
2.5. Gráfica de trabajo pendiente Sprint 5 33
2.6. Gráfica de trabajo pendiente Sprint 6 33
2.7. Diagrama de clases Camaleon. 38
2.8. Diagrama de clases de las entidades de la aplicación. 38
2.9. Diagrama de clases de la interface. 39
2.10. Diagrama de clases de la interface. 40
2.11. Diagrama de clases de la interface. 41
2.12. Diagrama de la estructura general de la aplicación 41
2.13. Diagrama de comportamiento. 42

3.1. Pantalla inicial CAMALEON 53
3.2. Selección de archivo de relación para cargar 53
3.3. Carga exitosa del archivo 54
3.4. Pantalla inicial CAMALEON con relación cargada 54
3.5. Calculo de las particiones de la relación 55
3.6. Particiones de la relación 55
3.7. Dependencias funcionales externas definidas en el modelo pseudomatematico y dependencias internas planteadas por la logica de conjuntos 57

Índice de cuadros

2.1. Historias de usuario 29
2.2. Historias de Usuario 30
2.3. Historias de usuario y tiempos de ejecucion 30
2.4. Caso de prueba CPU-001 Ingreso de datos al sistema 43
2.5. Caso de prueba CPU-002 Recubrimiento mínimo 43
2.6. Caso de prueba CPU-003 Cálculo de llaves 44
2.7. Caso de prueba CPU-004 Cálculo de particiones 44
2.8. Caso de prueba CPU-005 Generación del Script 45

3.1. Planteamiento del caso de estudio 50
3.2. Caso de estudio: Llaves candidatas 51

Dedicatoria

A Dios por haberme permitido llegar hasta aquí y haberme dado salud para lograr mis objetivos. A mis padres por ser mi ejemplo en todo lo que soy, en toda mi educación, tanto en mis estudios, como en la vida, por su apoyo incondicional. Y por último y no por ello menos importante a Liz, mi apoyo siempre...

Alexander Lozano

A Dios por cuidar de mi siempre y ser mi ayuda incondicional, a mi familia por su ejemplo de vida y por último a Alex mi complice y apoyo en todo momento.

Lizeth Valbuena

Parte I

Descripción del Proyecto

CAPÍTULO 1

Contextualización de la Investigación

1.1. Introducción

En el proceso de construcción de los sistemas informáticos, el diseño de las bases de datos (BD) ocupa un lugar importante; puede verse como un proceso compuesto por una serie de etapas para su desarrollo, relativamente independiente dentro del diseño de un sistema, este proceso aumenta los tiempos de las fases de elaboración y construcción del mismo. Como respuesta a esta problemática, surgen procesos como la normalización que es el proceso de organizar los datos de una base de datos, que incluye la creación de tablas y el establecimiento de relaciones entre ellas, según reglas diseñadas tanto para proteger los datos como para hacer que la base de datos sea más flexible al eliminar la redundancia y las dependencias incoherentes, además de algoritmos y buenas prácticas que al hacer uso de ellos, permiten simplificar el diseño del modelo de la base de datos.

Este documento plantea la creación un prototipo de software que haga uso de estos procesos y algoritmos para el diseño de bases de datos relacionales a partir de la generación de modelos de bases de datos relacionales con el fin de automatizar el diseño del modelo de bases de datos relacional

1.2. Definición del Problema

El diseño de esquemas de bases de datos relacionales tiene ya definido un proceso, y generalmente en su implementación encontramos anomalías de inserción, eliminación, y actualización de datos en las bases de datos relacionales que crean restricciones artificiales en la estructura de los mismos, dependencia entre ellos, y también problemas en la gestión y obtención la información.

Actualmente no existe una metodología capaz de separar el sistema del software y el sistema de la base de datos, se confunde software con sistema y base de datos con sistema. De este modo, es imprescindible "Conocer y definir en esencia el contexto del sistema sobre el cual se desea construir el diseño de la

base de datos; es un impositivo entender que antes de diseñar la base de datos y antes de diseñar el software se debe conocer de manera clara, precisa y lógia el contexto del sistema."[Piattini, 1999, Pág 3]

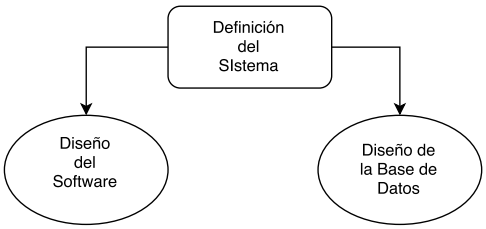


Figura 1.1: Escenarios de la ingeniería de Software [Londoño, 2016, Pág 4]

Teniendo en cuenta lo anterior este documento aborda el diseño de base de datos relacional partiendo del conocimiento absoluto del funcionamiento del sistema y la previa definición de dependencias funcionales para diseñar un prototipo de software que automatice la generación de un modelo de bases de datos relacional con base a los algoritmos existentes y que permita comprobar su validez esto con el fin de evitar problemas en la puesta en marcha de las diferentes funcionalidades del sistema y la estructura lógica de los datos almacenados.

¿Es posible construir un prototipo de software que permita automatizar el diseño de un modelo de bases de datos relacional a partir de dependencias funcionales?

¿Qué aspectos se deben tener en cuenta en la caracterización del prototipo para definir los datos de entrada con base utilizando dependencias funcionales que sirvan como base para el desarrollo de un prototipo de software?

¿Cómo contribuir a disminuir las inconsistencias, desde el inicio, en los diseños de bases de datos relacionales utilizando un prototipo de software?

1.3. Título

Desarrollo de un prototipo de software para la automatización del diseño de un esquema de bases de datos relacional

1.4. Hipótesis

Es posible automatizar el proceso de diseño de un esquema de base de relacional por medio de un prototipo de software.

1.5. Objetivos

1.5.1. Objetivo General

Desarrollar un prototipo de software que automatice el diseño de un esquema de base de datos relacional con base en el análisis de dependencias funcionales, algebra relacional y la aplicación del algoritmo de síntesis de Bernstein.

1.5.2. Objetivos Específicos

- Analizar los algoritmos de recubrimiento mínimo, llaves candidatas y síntesis de Bernstein ya que serán la base para generar un modelo de base de datos relacional
- Exportar un script de base de datos siguiendo el estándar SQL92 a partir del modelo relacional generado para implementarlo en un sistema gestor de base de datos compatible
- Construir un escenario experimental para verificar la validez del modelo generado.

1.6. Justificación de la investigación

Los sistemas informáticos en su proceso de construcción incluyen el diseño de bases de datos como un proceso imprescindible para llevar a cabo el funcionamiento de su sistema. Un modelo de bases de datos determina la estructura lógica para el almacenamiento, organización y manipulación de los datos, definir de manera incorrecta dicho modelo dará lugar a estructuras de datos redundantes que desperdician el espacio de disco y crean problemas de mantenimiento. Es por ello que resulta de interés el estudio de los problemas relacionados con el diseño de las bases de datos y la modelación de la información. La identificación y resolución de los tipos de dependencias funcionales (DFE, DFNE) [Londoño, 2016, Pág 15] garantiza el correcto diseño de los modelos de bases de datos, simplificando la aplicación de los métodos existentes, de este modo se disminuyen las inconsistencias actuales a la hora de diseñar un modelo relacional, mejorando los sistemas, procedimientos de las organizaciones.

1.7. Alcances

En la ejecución del proyecto de investigación se busca desarrollar un prototipo de software que cumpla con la siguiente característica:

1. Aplicación para escritorio que permita generar un modelo de base de datos con base relacional en tercera forma normal y ser exportarlo en un script según el estándar SQL92

2. Desarrollo del prototipo planteado utilizando herramientas open Source para evitar sobrecosto que implica el uso de software licenciado.

1.8. Restricciones del prototipo

El prototipo de software que se busca desarrollar presenta las siguientes restricciones:

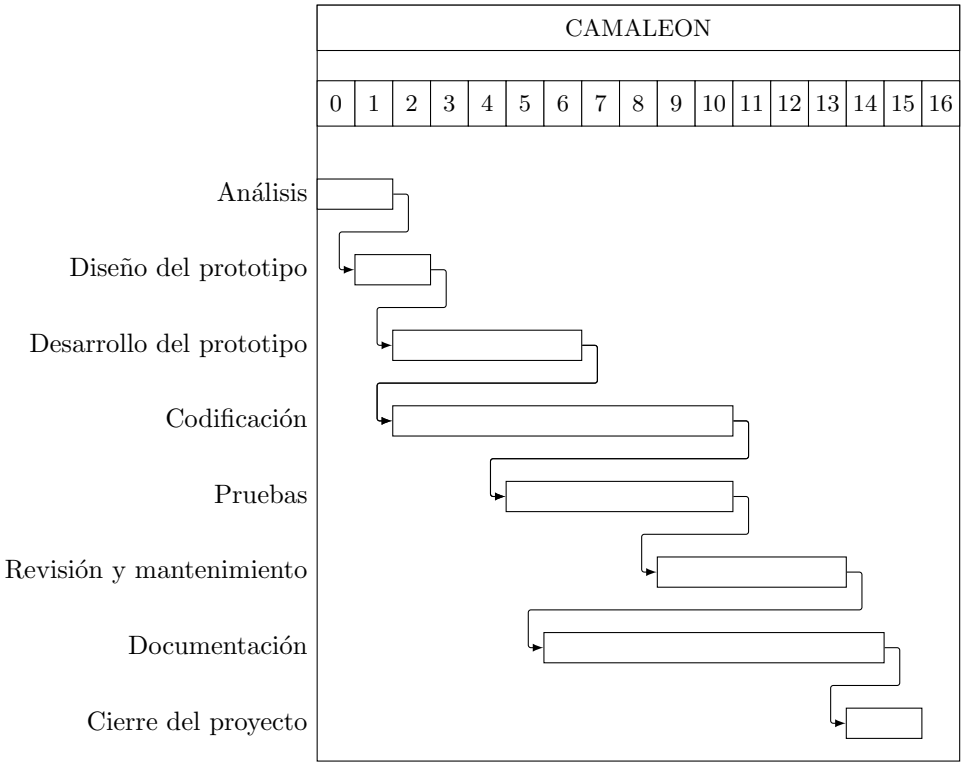
1. El script de base de datos generado a partir del modelo relacional podrá ser ejecutado en un sistema gestor de base de datos compatible con el estándar SQL92.

2. El prototipo construido tendrá una Interfaz mono-usuario.

1.9. Resultados esperados

El resultado esperado de la investigación es un prototipo que sirva como apoyo para el diseño bases de datos relacionales aplicando las teorías de dependencias funcionales, recubrimiento mínimo y síntesis de Bernstein, y como resultado la generación del modelo en un script que cumpla con el estándar SQL92 que permita ser ejecutado en motores de bases de datos afines.

1.10. Cronograma de actividades



1.11. Marco Referencial

1.11.1. Marco teórico

1.11.1.1. Bases de datos relacionales

Las bases de datos relacionales son un tipo de base de datos que pretenden representar los datos en forma de tablas y relaciones, es uno de los modelos más utilizados y populares, creado por Edgar Frank Codd a finales de los años 60. Según [Korth, 2004] las bases de datos relacionales consisten en un conjunto de tablas, a las cuales se les asigna un nombre exclusivo; tiene como estructura la selección de atributos para cada tabla y un dominio asociado a ese atributo. Estos modelos son basados en fundamentos matemáticos ya que las relaciones de las tablas se ven como subconjuntos del producto cartesiano en la lista de dominios. Se utilizan los términos matemáticos relación y tupla en lugar de los términos tabla y fila. Una variable tupla es una variable que representa a una tupla; en otras palabras, una tupla que representa al conjunto de todas las tuplas. [Korth, 2004]

1.11.1.2. Dependencias Funcionales

Según [Korth, 2004] "una dependencia funcional (abreviada DF) es un vínculo muchos a uno que va de un conjunto de atributos a otro dentro de una determinada varrel o tabla". Dada esta definición si tenemos una tabla X donde se encuentren subconjunto $A \rightarrow B$, se dice que hay una dependencia de B hacia A si se encuentran tuplas o registros con el mismo valor; es decir que los valores de A se encuentren también en B, siendo A el subconjunto determinante y B el subconjunto dependiente.

Las dependencias son propiedades inherentes al contenido semántico de los datos; forman parte de las restricciones de usuario del modelo relacional y se han de cumplir para cualquier extensión de un esquema de relación.

Puesto que las dependencias constituyen una parte importante de la semantica del Universo Discurso que se pretende modelar, son incluidas, en la definición de esquema relacional y de los esquemas de relación, que forman parte del mismo. [Piattini, 1999, Pág 61]

Sea el esquema de relación R(A,DF) y sean X e Y dos descriptores. Se dice que existe una dependencia funcional entre X e Y de forma que X determina a Y, si, y sólo si, se cumple que para cualesquiera dos tuplas de R, u y v tales que u[X] = v[X], entonces necesariamente u[Y] = v[Y] (la notación u[X] o v[X] indica la proyección de la tupla u o de la tupla v, respectivamente, sobre el descriptor X, cuyo resultado será un valor de X); análogamente u[Y] o v[Y] indica la proyección de u o de v, respectivamente, sobre el descriptor Y.

Un determinante o implicante es un conjunto de atributos del que depende funcionalmente otro conjunto de atributos al que llamamos, implicado.[Piattini, 1999, Pág 117]

- `Dependencia funcional trivial:` Una dependencia funcional $X \rightarrow Y$ se dice que es trivial si Y es un subconjunto de X. Por ejemplo serán triviales las siguientes dependencias:

  $CodEstudiante \rightarrow CodEstudiante$

  $CodCurso, CodEdicion \rightarrow CodCurso$

- `Dependencia funcional elemental` Decimos que una dependencia funcional $X \rightarrow Y$ es elemental si Y es un atributo único no incluido en X, y no existe X' incluido en X tal que $X^{'} \rightarrow Y$, es decir, la dependencia funcional elemental es una dependencia funcional completa, no trivial y en la que el implicado es un atributo único.

  No todas las dependencias funcionales son útiles para la teoría de la normalización, sino solamente las elementales, que acabamos de definir, y que son un subconjunto de las dependencias funcionales.

- `Dependencia funcional transitiva:` Dado un esquema de relación R(X,Y,Z) en las que existen las siguientes dependencias funcionales.
  $X \rightarrow Y$

$Y \rightarrow Z$
$Y \not\rightarrow X$
Se dice que Z tiene una dependencia transitiva respecto a X a través de Y, lo que se representa por
$X \rightarrow Z$
Si, además,$Z \not\rightarrow Y$ se dice que la dependencia transitiva es estricta.

1.11.1.3. Relación

Es un conjunto de m elementos denominados tuplas t_1 . Cada $tupla_j$ es conjunto de pares $(a_1 : v_1, a_2 : v_2, a_3 : v_3, a_4 : v_4...)$ donde cada a_i es el nombre de un atributo y v_{ij} es un valor correspondiente del dominio D_i sobre el que está definido un atributo: [Piattini, 1999, Pág 97]

1.11.1.4. Integridad de Entidad

Es una restricción inherente, debida asimismo a la necesidad de que todas las tuplas de una relación sean distintas, la cual establece que: "Ningún atributo que forme parte de la clave primaria puede tomar valores nulos". Valor nulo de un atributo representa información desconocida, inaplicable, etc. Por lo tanto, si alguno de los atributos que forman parte de la clave primaria tomase valores nulos, algunas de las tuplas de la relación no podrían ser identificadas, por lo que se violaría la condición de que el valor de la clave debe ser único para cada tupla.[Piattini, 1999]

- Las columnas de la tabla son los atributos que expresan las propiedades de la relación. El número de atributos se llama grado de la relación.
- Cada fila de la tabla, llamada tupla, es un elemento del conjunto que es la relación. El número de tuplas se llama cardinalidad de la relación. La cardinalidad varía en el transcurso del tiempo.

No se deben confundir los conceptos de tabla y de relación, puesto que:

- Una tabla es una forma de representar una relación.
- Una relación tiene unas propiedades intrínsecas que no tiene una tabla, y que se derivan de la misma definición matemática de relación, ya que, al tratarse de un conjunto, en una relación:
  - No puede haber dos tuplas iguales.
  - El orden de las tuplas no es significativo.
  - El orden de los atributos no es significativo.
  - Cada atributo sólo puede tomar un único valor del dominio
  - No se admiten grupos repetitivos (ni otro tipo de estructuras) como valores de los atributos de una tupla.

1.11.1.5. Axiomas de Armstrong

Sean los descriptores X, Y, Z, W, los axiomas de Armstrong, es decir, las reglas de derivación de dependencias, establecen:[Piattini, 1999].

- `Reflexibidad:`
  Si $Y \subseteq X$, $X \rightarrow Y$

- `Aumentatividad:`
  Si $X \rightarrow Y$, $Z \subseteq W$, Entonces $XW \rightarrow YZ$

- `Transitividad:`
  Si $X \rightarrow Y$, $Y \rightarrow Z$ entonces $X \rightarrow Z$

- `Proyectividad:`
  Si $X \rightarrow Y$, entonces $X \rightarrow Y$ si $Y^{'} \subset Y$

- `Union a aditividad:`
  Si $X \rightarrow Y$, $X \rightarrow Z$ entonces $X \rightarrow YZ$

- `Pseudotransitividad`
  Si $X \rightarrow Y$, $YW \rightarrow Z$ entonces $XW \rightarrow Z$

1.11.1.6. Algoritmo de síntesis de Bernstein

El proceso de síntesis se puede describir, de la forma siguiente: R (A, DF) donde A es un conjunto de atributos y DF el conjunto de dependencias funcionales que existen entre dichos atributos.[Piattini, 1999, Pág 172].

1. Se busca un recubrimiento minimal DF^m del conjunto de dependencias funcionales DF.

2. Se agrupan las dependencias de DF^m en particiones que tengan el mismo implicante.

3. Se forma un esquema de relación R, para cada partición, el cual tendrá como atributos todos los que aparezcan en la correspondiente partición así como las dependencias funcionales implicadas.

4. Si existen atributos que no son implicantes ni implicados en DF^m, se forma un esquema de relación con éstos sin dependencias funcionales.

1.11.1.7. Normalización de base de datos

La normalización de bases de datos son una serie de reglas que se deben tener en cuenta para hacer el diseño de las mismas. Para que una tabla cumpla con estas reglas se deben tener en cuenta 5 formas normales y una denominada Boyce-Cood.

- `Primera forma normal (1FN):` "una tabla esta en 1FN si y solo si, en cada valor valido de esa tabla, todo el registro contiene exactamente un valor para cada atributo" [Londoño, 2016, Pág 357]
- `Segunda forma normal (2FN):` esta forma normal está definida por Date (2004) como: "una varrel está en 2FN si y sólo si está en 1FN y todo atributo que no sea clave es dependiente irreduciblemente de la clave primaria"[Korth, 2004, Pág 360].
- `Tercera forma normal (3FN):` "Una varrel está en 3FN si y sólo si está en 2FN y los atributos que no son clave son dependientes en forma no transitiva de la clave primaria". [Korth, 2004, Pág 362]
- `Forma normal de Boyce/Codd (FNBC):` "Una varrel está en FNBC si y solamente si toda DFi trivial, irreducible a la izquierda, tiene una clave candidata como su determinante" [Korth, 2004, Pág 362]
- `Cuarta forma normal (4FN):` "una relación está en 4FN cuando se tiene la forma FNBC y no se tienen dependencias multivaluadas" [Kroenke, 2003] .
- `Quinta forma normal (5FN):` "Una varrel R está en 5FN —también llamada forma normal de proyección-junta (FN/PJ) — si y solamente si cada dependencia de junta no trivial válida para R está implicada por las claves candidatas de R [Korth, 2004, Pág 398].

1.11.1.8. Clave Candidata

Una clave candidata de una relación es un conjunto no vacío de atributos que identifica unívoca y minímamente cada tupla de una realación. En toda relación siempre, hay al menos, una clave candidata, ya que el conjunto de atributos que constituye la relación gozará de la propiedad de unicidad (por la definición misma de relación no puede haber dos tuplas iguales)y, si no tuviese la de minimalidad se podría prescindir de aquellos atributos que lo impidieran, obteniendo así una clave candidata.[Piattini, 1999, Pág 103] Una relación puede tener mas de una clave candidata, entre ellas cabe distinguir:

- `Clave primaria:` Es una clave candidata que el usuario elegirá por condiciones ajenas al modelo relacional, para identificar las tuplas de la relación. Los atributos que forman la clave primaria no pueden ser valores nulos.[Piattini, 1999, Pág 103]
- `Clave Alternativas:` Son aquellas claves candidatas que no han sido elegidas como claves primarias de la relación.[Piattini, 1999, Pág 103]

1.11.1.9. Atributo extraño

Dada la dependencia $X \rightarrow Y$ de DF, un atributo A perteneciente a X se dice que es un atributo extraño en la dependencia si la dependencia $(X - A) \rightarrow Y$ se encuentra en DF^{+}.

1.11.1.10. Dependencia redundante

Una dependencia funcional f de DF se dice que es redundante si puede derivarse de DF - f mediante la aplicación de los axiomas de Armstrong.

1.11.1.11. Metodología de desarrollo Scrum

Scrum es una metodología de desarrollo de software, iterativa e incremental, basado en iteraciones de trabajo conocidos como sprints, los cuales se realizan en bloques de tiempo definidos de no más de cuatro semanas, durante las cuales el equipo realiza reuniones periódicas, a menudo diarias, para revisar el progreso de las tareas asignadas; la duración de los sprints es fija, por lo que cada uno termina en una fecha específica aunque no se hayan cumplido los objetivos propuestos. Para cada uno de los sprints, se definen los Elementos (requerimientos del cliente), que se van a trabajar, estos elementos no se pueden cambiar a lo largo del sprint. Al final de cada sprint, se realiza una reunión con los interesados del proyecto para la revisión del avance del producto, con el fin de obtener retroalimentación de los resultados del sprint y usarlo como insumo para el siguiente.[Metodologias agiles, 2016].

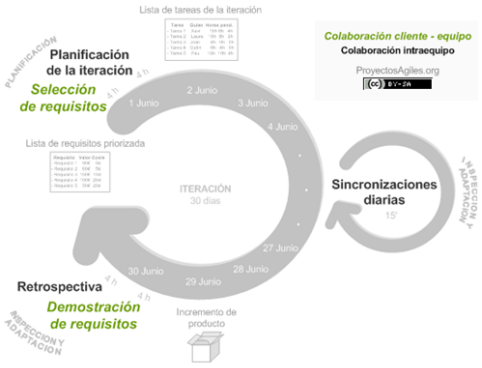


Figura 1.2: Metodologías Scrum [Metodologias agiles, 2016]

Cada una de las iteraciones se compone de 3 fases:

- `Planificación de la iteración` Se realiza el primer día de la iteración y se busca seleccionar los elementos a trabajar, se crea una lista de tareas y los miembros del equipo se auto asignan tareas de la lista.
- `Ejecución de la iteración:`Se realiza la ejecución de las tareas planeadas y se realizan reuniones de sincronización, en las cuales se valida el avance de todos los miembros del equipo.
- `Inspección y adaptación:` Se realiza reunión con los interesados, para mostrar el resultado obtenido en la iteración, se recibe retroalimentación;

además se realiza una reunión de verificación al interior del equipo para validar el trabajo durante la iteración.

[Metodologias agiles, 2016].

Parte II

Desarrollo de la investigación

CAPÍTULO 2

Desarrollo de la investigación

2.1. Gestión del proyecto

De acuerdo a la metodología ágil de desarrollo de software SCRUM, en un proyecto se deben tener en cuenta la lista priorizada de tareas del producto (product backlog), la planeación de sprints y los gráficos de esfuerzo pendiente, a continuación, se muestran estos artefactos.

2.1.1. Lista priorizada de tareas del producto

A continuación, se muestra las historias de usuario establecidas para el desarrollo del prototipo de software para la automatización del diseño de un esquema de bases de datos relacional, para ver la especificación de cada una y sus criterios de aceptación ir al anexo correspondiente.

ID de la Historia	**Característica - Funcionalidad**	**Anexo**
CAM-01	Necesito ingresar una relación compuesta por atributos y dependencias funcionales	Anexo1-HistoriasUsuario
CAM-02	Necesito poder calcular la clausura de un conjunto de atributos, con base en una relación compuesta de atributos y dependencias funcionales	Anexo1-HistoriasUsuario
CAM-03	Necesito calcular el recubrimiento mínimo de una relación ingresada previamente	Anexo1-HistoriasUsuario
CAM-04	Necesito poder calcular las llaves candidatas de la relación	Anexo1-HistoriasUsuario
CAM-05	Necesito calcular las particiones de la relación	Anexo1-HistoriasUsuario
CAM-06	Necesito generar el script del modelo relacional	Anexo1-HistoriasUsuario

Cuadro 2.1: Historias de usuario

2.1.2. Planeación de sprints

A continuación, se muestran los sprints utilizados para realizar el desarrollo de las historias de usuario, la duración elegida para cada sprint fue de dos semanas.

HISTORIAS DE USUARIO PARA EL PROTOTIPO DEL PROTOTIPO DE SOFTWARE PARA LA AUTOMATIZACIÓN DEL DISEÑO DE UN ESQUEMA DE BASES DE DATOS RELACIONAL			
ID de la historia	**Rol**	**Funcionalidad**	**Razón / Resultado**
CAM-01	Usuario	Necesito ingresar una relación compuesta por atributos y dependencias funcionales	Con la finalidad de poder hacer uso de la aplicación
CAM-01	Usuario	Necesito poder calcular la clausura de un conjunto de atributos, con base en una relación compuesta de atributos y dependencias funcionales	Con la finalidad de poder utilizar esta funcionalidad en el cálculo de recubrimiento mínimo y llaves candidatas de la relación
CAM-01	Usuario	Necesito calcular el recubrimiento mínimo de una relación ingresada previamente	Con la finalidad de obtener el recubrimiento mínimo, para posteriormente obtener el modelo relacional
CAM-01	Usuario	Necesito poder calcular las llaves candidatas de la relación	Con la finalidad de,obtener las llaves candidatas, para posteriormente obtener el modelo relacional
CAM-01	Usuario	Necesito calcular las particiones de la relación	Con la finalidad de obtener el modelo relacional
CAM-01	Usuario	Necesito generar el script del modelo relacional	Con la finalidad de obtener el script del modelo relacional

Cuadro 2.2: Historias de Usuario

2.1.3. Ejecución de sprints

A continuación se relacionan las tareas realizadas en cada uno de los sprints, junto con las gráficas de trabajo pendiente, en las cuales se evidencia que durante la ejecución del proyecto se cumplió con los objetivos propuestos para cada sprint.

ID Sprint	Descripción	Duración
1	Historia de usuario CAM-01 – Ingresar una relación compuesta por atributos y dependencias funcionales	2 Semanas
2	Historia de usuario CAM-02 – Calcular la clausura de un conjunto de atributos, con base en una relación compuesta de atributos y dependencias funcionales	2 Semanas
3	Historia de usuario CAM-03 – Calcular el recubrimiento mínimo de una relación ingresada previamente	2 Semanas
4	Historia de usuario CAM-04 – Calcular las llaves candidatas de la relación	2 Semanas
5	Historia de usuario CAM-05 – Calcular las particiones de la relación	2 Semanas
6	Historia de usuario CAM-06 – Generar el script del modelo relacional	2 Semanas
	TOTAL	**3 Meses**

Cuadro 2.3: Historias de usuario y tiempos de ejecucion

2.2. Fase de análisis

2.2.1. Stakeholders

- `Universidad Distrital Facultad de ingeniería Programa Especialización en ingeniería de Software:` Organización a la cual está dirigida la investigación.
- `Alexander Lozano:` Analista y desarrollador del proyecto
- `Lizeth Valbuena:` Analista y desarrollador del proyecto
- `Roberto A Pava:`Director del proyecto de investigación
- `John Jairo Londoño:` Revisor Proyecto de Investigación

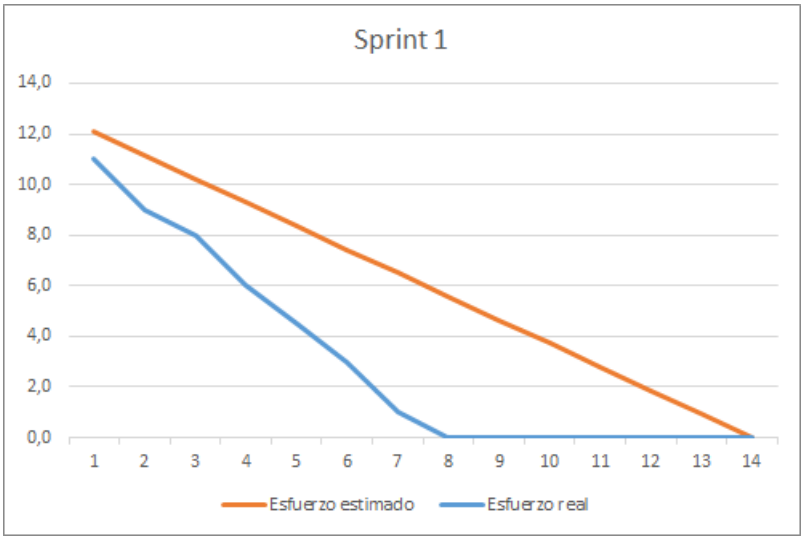


Figura 2.1: Gráfica de trabajo pendiente Sprint 1

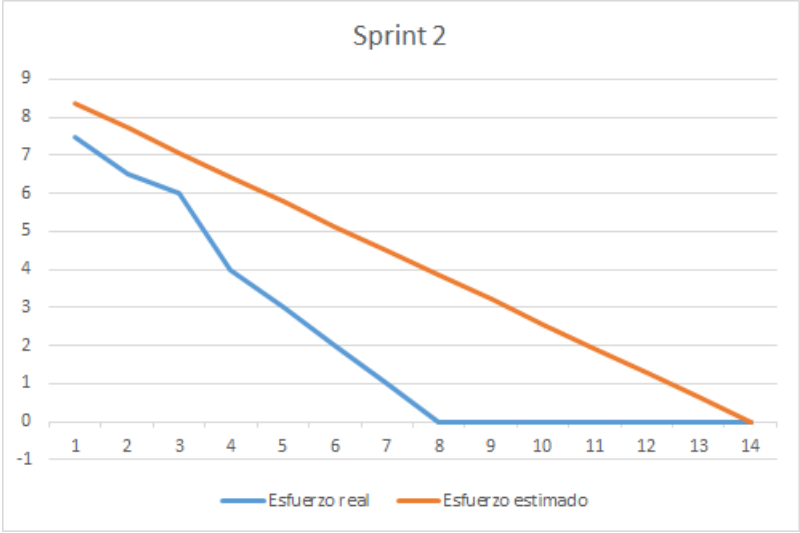


Figura 2.2: Gráfica de trabajo pendiente Sprint 2

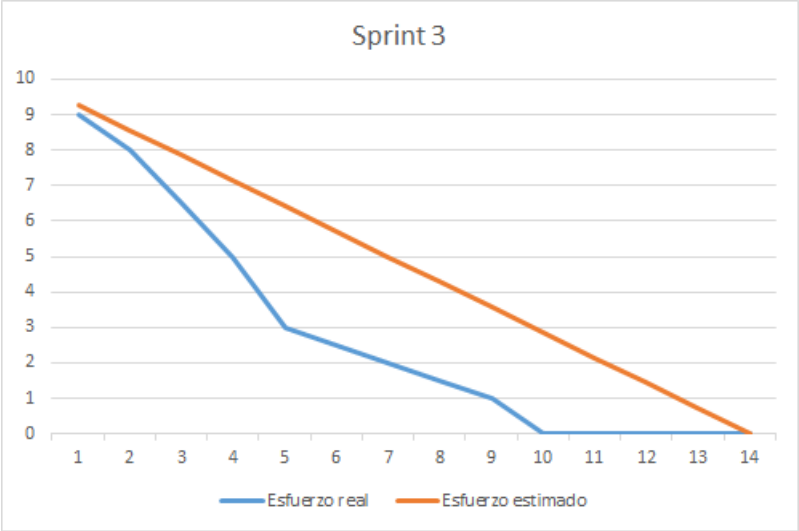


Figura 2.3: Gráfica de trabajo pendiente Sprint 3

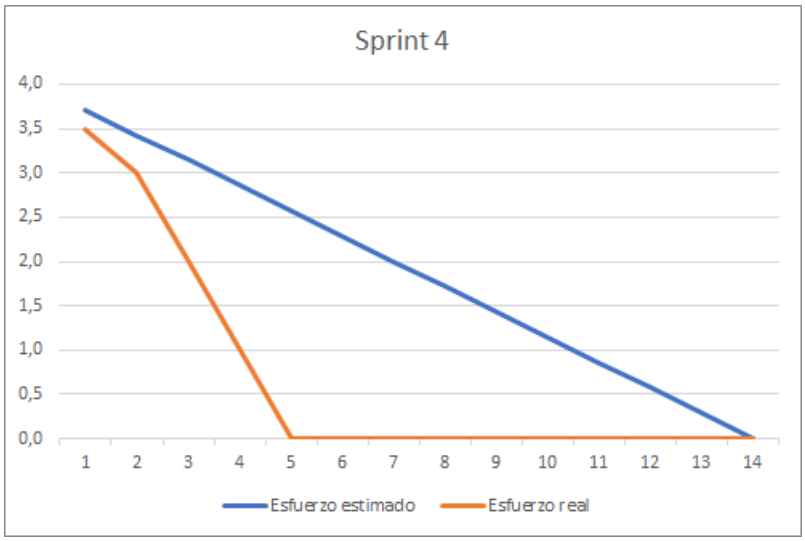


Figura 2.4: Gráfica de trabajo pendiente Sprint 4

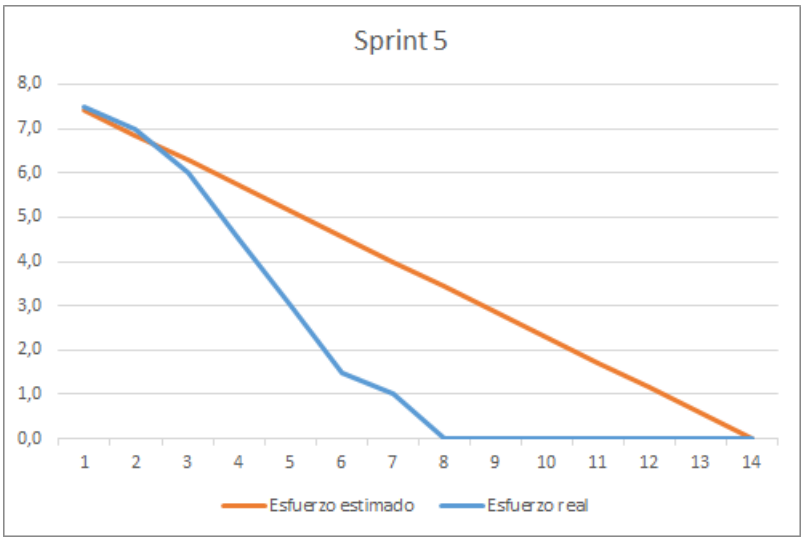


Figura 2.5: Gráfica de trabajo pendiente Sprint 5

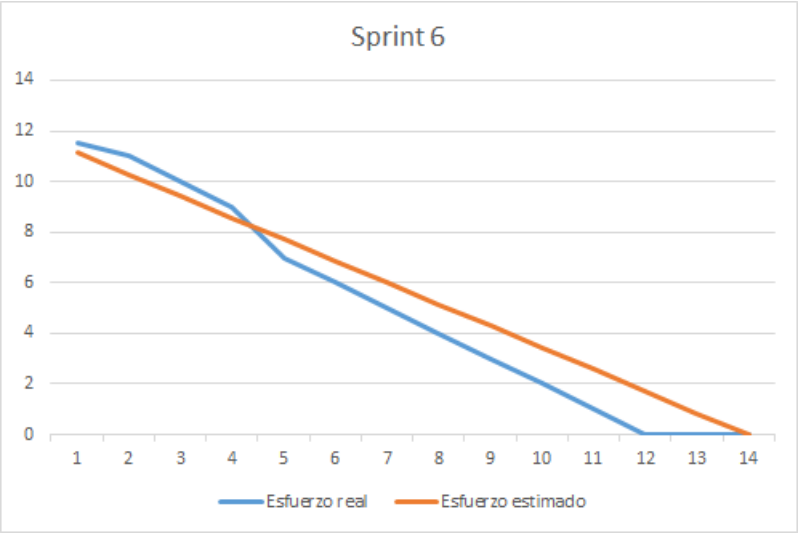


Figura 2.6: Gráfica de trabajo pendiente Sprint 6

2.2.2. Requerimientos funcionales

- Desarrollar un prototipo de software que permita ingresar dependencias funcionales de un sistema y que automatice la generación del modelo de base de datos relacional en 3FN en un script de base de datos.
- El prototipo de software deberá validar si todos los atributos que hacen parte de las dependencias funcionales están definidos en el conjunto de atributos.
- El prototipo de software deberá aplicar las reglas de recubrimiento mínimo
- El prototipo de software deberá realizar el cálculo de la(s) llave(s) candidatas.
- El prototipo de software aplicará los algoritmos necesarios para generar un modelo de bases de datos relacional en 3FN.

2.2.3. Requerimientos no funcionales

- El prototipo de software deberá implementar una interfaz gráfica que permita al usuario el ingreso de los datos y del mismo modo presentar al usuario los resultados obtenidos.
- El prototipo de software deberá considerar en la implantación de la automatización de los algoritmos el aprovechamiento de los recursos con el fin de optimizar los tiempos de respuesta.

2.3. Caracterización del prototipo

El primer paso en la concepción de una base de datos es definir el universo discurso, fijando para ello una serie de objetivos sobre el mundo real que se va a analizar.

Un universo discurso (UD) de una base de datos relacional está compuesto por un conjunto de dominios (D) y de relaciones (R) definidas sobre los dominios.

Un dominio es un conjunto nominado, infinito y homogeneo de valores atómicos.[Piattini, 1999, Pág 61]

En este sentido cuando hablamos de un universo discurso no es otra cosa que el marco conceptual y teórico del sistema, a lo que también podríamos denominar Cadena lógica del negocio del sistema (CLNS). [Londoño, 2016]

2.3.1. Dependencias funcionales

La automatización de la generación del modelo de base de datos realcional propuesta en este documento parte de la correcta definición de atributos y dependencias funcionales del sistema según el dominio del sistema que se quiere modelar como insumo fundamental para su funcionamiento.

2.3.2. Cálculo del cierre (algoritmo de Ullman)

- Nos permite determinar si una dependencia se puede derivar de un conjunto de dependencias, es decir si una dependencia puede ser deducida a partir de otra(s).
- Por otro lado es útil para obtener las claves de un esquema y saber si un descriptor es clave.
- También será útil para deducir la equivalencia entre dos conjutos de dependencias y por ende obtener el recubrimiento mínimo a partir de un conjuto de dependencias funcionales.

`E`jemplo:
Tenemos un esquema de relación `R=(T,L)` donde:
`T`=A,B,C,D,E,F
`L`=$AB \rightarrow C$, $D \rightarrow EF$, $C \rightarrow A$, $BE \rightarrow C$, $BC \rightarrow D$, $CF \rightarrow BD$, $ACD \rightarrow B$, $CE \rightarrow AF$

Vamos a calcular el cierre del descriptor `BD` con respecto a `L`

- `Paso 1:`
$$BD^{(0)} = BD$$
- `Paso 2:` (Por la segunda dependencia funcional)
$$BD^{(1)} = BDEF$$
- `Paso 3:` (Por la cuarta dependencia funcional)
$$BD^{(2)} = BDEFCA$$
- `Paso 4:` (Por la tercera dependencia funcional)
$$BD^{(3)} = BDEFCA$$

Dado que el cierre del descriptor BD contiene todos los atributos de T se da por finalizado el proceso.

2.3.3. Cálculo de recubrimiento mínimo

El cálculo del recubrimiento mínimo se refiere a la obtención del mínimo conjunto de dependencias funcionales que definen un sistema también se conoce como recubrimiento irredundante. Para llevar a cabo este procedimiento haremos uso del algoritmo utilizado para el cálculo del cierre de un descriptor conocido como algoritmo de Ullman y los demás procedimientos definidos para el calculo del recubrimiento mínimo.

De este modo un conjunto de dependencias funcionales se dice que es mínimo cuando cumple:

- Todas sus dependencias son elementales.
- No existe en el conjunto de dependencias ninguna redundante, es decir que se pueda deducir del resto aplicando Axiomas de Armstrong.

En resumen el procedimiento a seguir se basa en los conceptos de atributos extraños y dependencias redundantes.(Vea numeral 1.10)

2.3.4. Cálculo de llaves candidatas

Una llave candidata es un conjunto de elementos que pertenecen al universo discurso y por medio de los cuales es posible determinar todos los atributos del sistema como consecuencia lógica de las dependencias funcionales.

A partir del recubrimiento mínimo obtenido y haciendo uso del cálculo del cierre llegaremos a definir las claves candidatas de un sistema, a partir de un universo discurso específico.

2.3.5. Algoritmo de síntesis de Bernstein

Los algoritmos de síntesis se caracterizan por agrupar atributos afines, con el fin de generar esquemas en tercera forma normal.

Para aplicar el algoritmo de Bernstein partimos del resultado obtenido en el procedimiento de Cálculo de recubrimiento mínimo, este será nuestro conjunto de dependencias funcionales y atributos, también llamado Relación universal.

En primer lugar se agrupan las dependencias funcionales que tengan el mismo determinante.Estas serán las particiones.

Para cada partición se genera un esquema de relación R la cual tendrá como clave primaria el determinante y como atributos los implicantes de las dependencias funcionales de la partición

En caso de que existan atributos que no son implicantes ni implicados, se forma un esquema de relación con ellos.

2.4. Fase de Diseño

2.4.1. Resultados del análisis

Con el fin de realizar el proceso de generación de un modelo relacional en 3FN es necesario tener como insumo una relación R compuesta por un conjunto de atributos y dependencias funcionales de la siguiente manera R = (T,DF). Donde T es una lista de atributos finitos y unicos, como por ejemplo T=A,B,C,D,E y DF es una lista de dependencias funcionales las cuales están compuestas por un implicante y un implicado como DF=$AB \rightarrow C$, $D \rightarrow EF$, $C \rightarrow A$, $BE \rightarrow C$, $BC \rightarrow D$, $CF \rightarrow BD$, $ACD \rightarrow B$, $CE \rightarrow AF$.

2.4.1.1. Cálculo de cierre de un descriptor

Para la automatización de este procedimiento podemos concluir que:

- `Insumos` Es necesario definir un conjunto de atributos y relaciones.
- `Resultados esperados:` un conjunto de atributos.
- El procedimiento de calculo de cierre debe ser reutilizable.

2.4.1.2. Cálculo del recubrimiento mínimo

Para la automatización de este procedimiento podemos concluir que:

- `Insumos:` Es necesario definir un conjunto de atributos y dependencias funcionales.
- `Resultados esperados:` un conjunto de atributos y dependencias funcionales.
- La estructura definida del procedimiento debe estar en la capacidad de intercambiar información con el procedimiento de Calculo de cierre con el fin de facilitar la reutlización de procesos.

2.4.1.3. Cálculo de llaves candidatas

Para la automatización de este procedimiento podemos concluir que:

- `Insumos` Es necesario definir un conjunto de atributos y dependencias funcionales.
- `Resultados esperados:` 1 o mas conjuntos de atributos.
- La estructura definida del procedimiento debe estar en la capacidad de intercambiar información con el procedimiento de Calculo de cierre con el fin de facilitar la reutlización de procesos.
- Es necesario que este procedimiento tenga definida una estructura que facilite su reutilización en caso de ser solicitado por otras funcionalidades.

2.4.1.4. Algoritmo de síntesis de Bernstein

Para la automatización de este procedimiento podemos concluir que:

- `Insumos` Es necesario definir un conjunto de atributos y dependencias funcionales.

- `Resultados esperados:` Particiones definidas y llaves primarias.
- La estructura definida del procedimiento debe estar en la capacidad de intercambiar información todos los procedimientos previamente definidos.

2.5. Fase de Desarrollo

2.5.1. Diagrama de clases

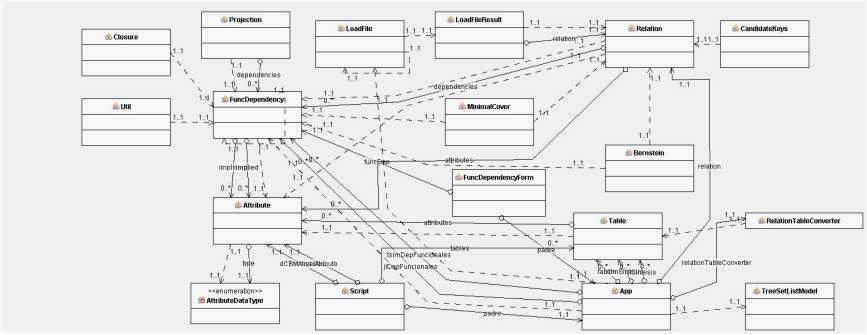


Figura 2.7: Diagrama de clases Camaleon.

Para mayor comprensión ver “Anexo-2 Diagrama de Clases”.

2.5.2. Diagrama de Clases-entidades

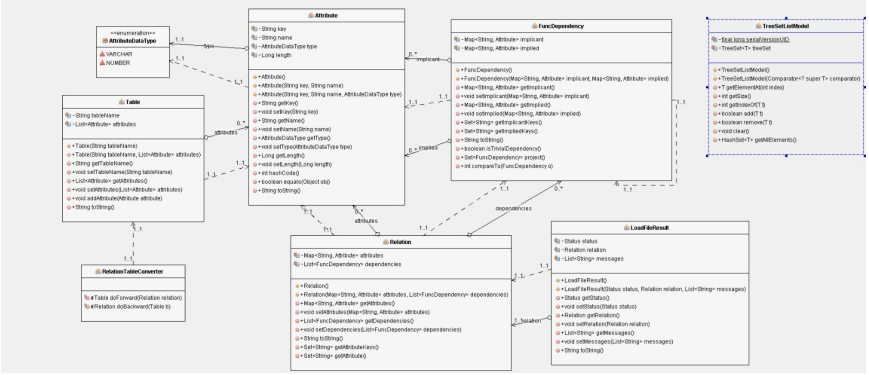


Figura 2.8: Diagrama de clases de las entidades de la aplicación.

2.5.3. Diagrama de clases de la interface gráfica de la aplicación

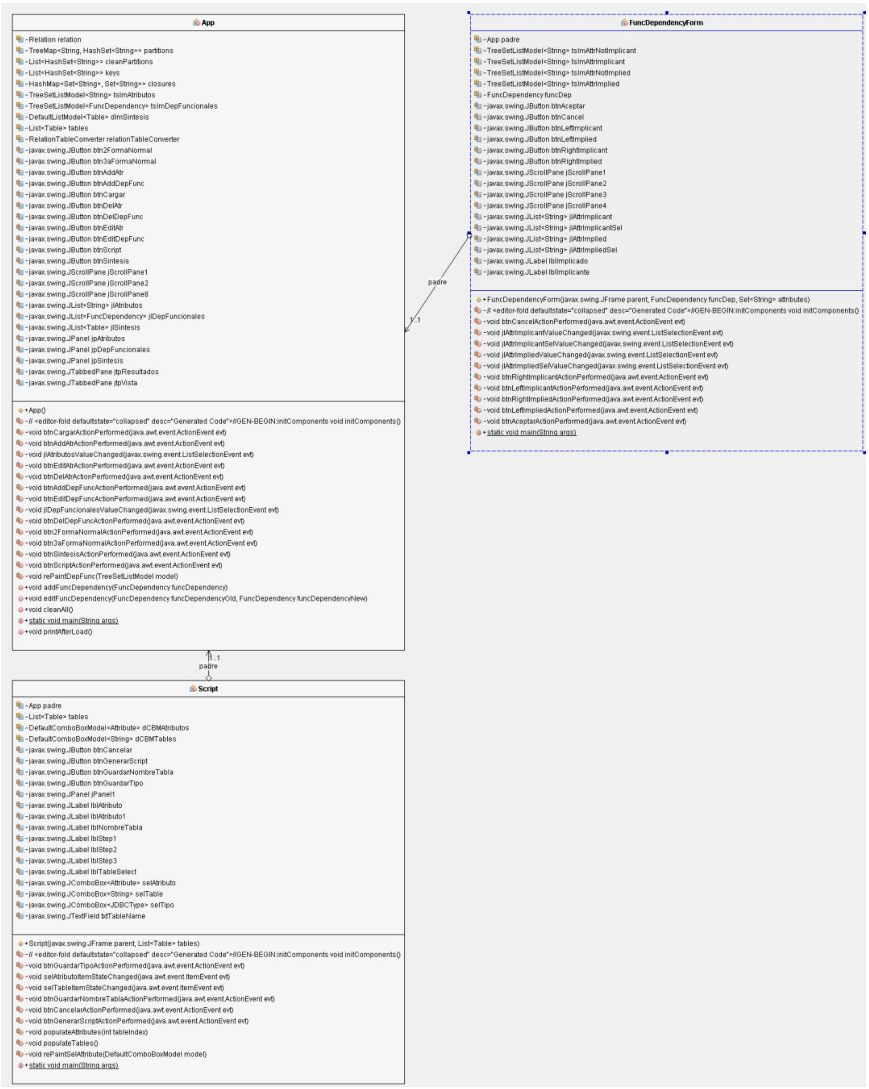


Figura 2.9: Diagrama de clases de la interface.

2.5.4. Diagrama de Secuencia

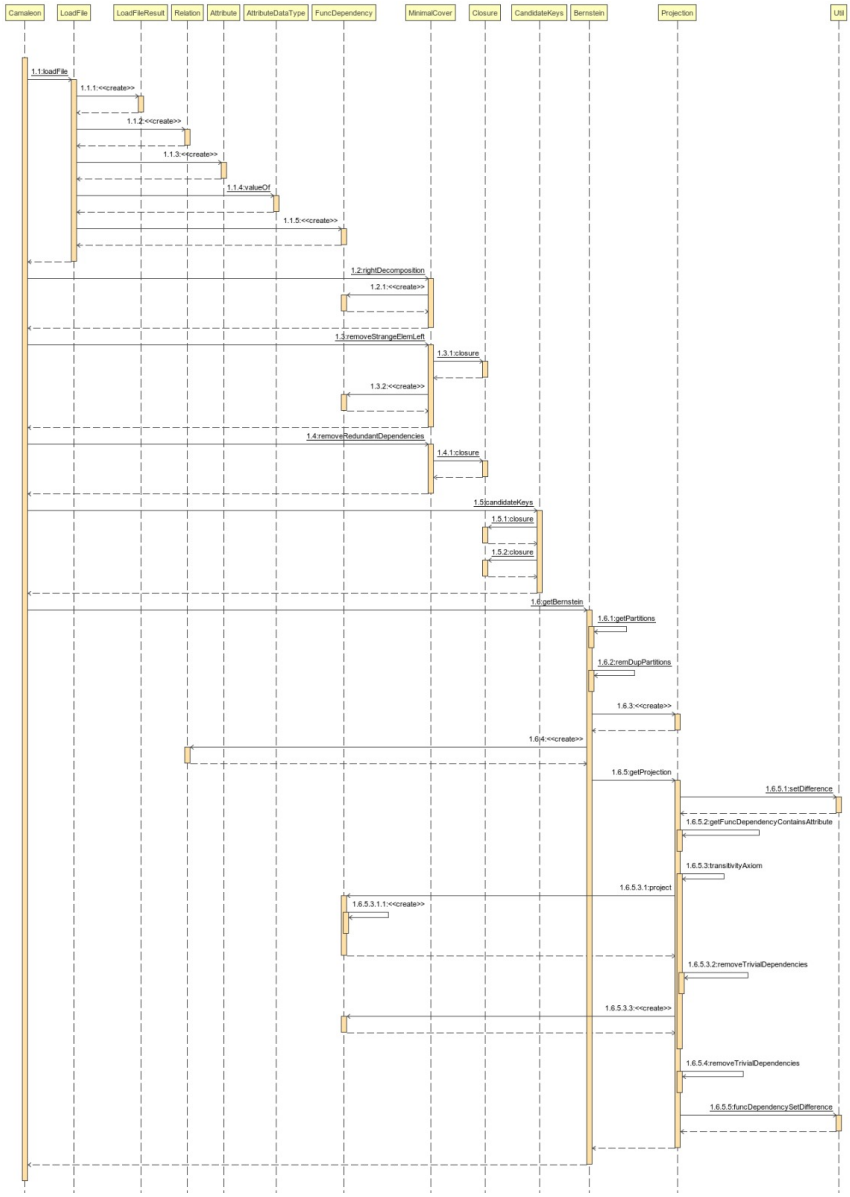


Figura 2.10: Diagrama de clases de la interface.

2.5.5. Diagrama de clases de implementación de la lógica en la aplicación

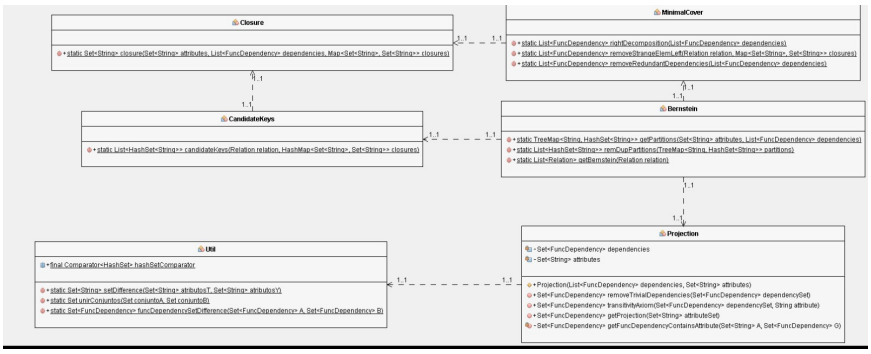


Figura 2.11: Diagrama de clases de la interface.

2.5.6. Diagrama de la estructura general de la aplicación

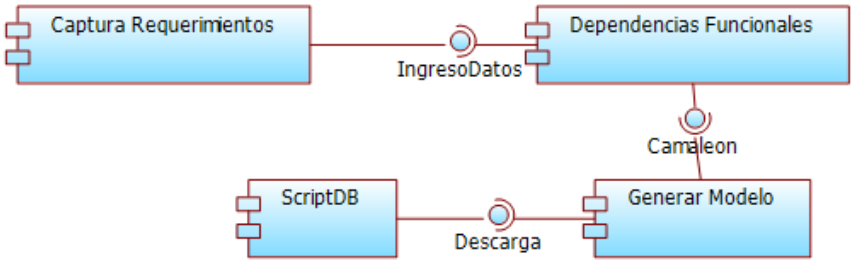


Figura 2.12: Diagrama de la estructura general de la aplicación

2.5.7. Diagrama comportamiento de la aplicación

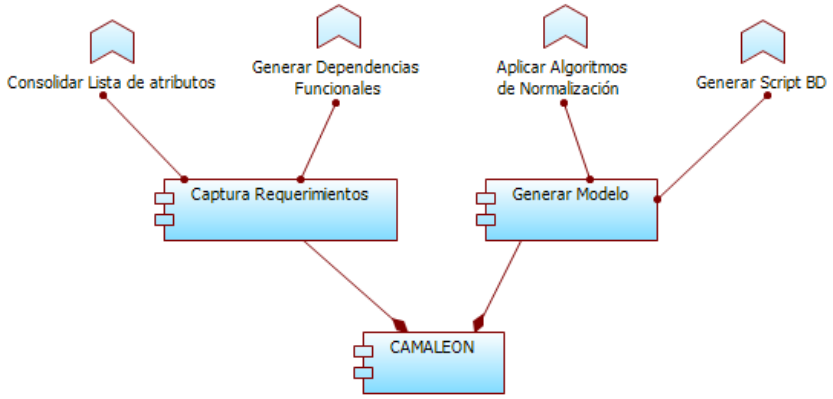


Figura 2.13: Diagrama de comportamiento.

2.6. Implementación

2.6.1. Documentación del Código

Ver anexo Documentación del código

2.6.2. Casos de Prueba

De acuerdo a la historias de usuario previamente definidas se establecen los siguientes casos de prueba:

Identificador (ID) de la Historia	**ID Caso Prueba**
CAM-01	CPU-001
Nombre Caso de Prueba	Ingreso y validación de datos al sistema
Descripción	Permitir ingresar una relación compuesta por atributos y dependencias funcionales
Condiciones de ejecución	Al iniciar la aplicación debe permitir el ingreso de atributos al sistema. Para el ingreso de una dependencia funcional deben existir mas de dos atributos previamente definidos. Al ingresar dos atributos repetidos el sistema debe ser capaz de eliminar la duplicidad existente
Entradas	Modo Gráfico: - El usuario por medio del boton añadir agrega atributos individuales hasta completar la lista de atributos del sistema. - Cuando exista una lista de 2 o mas atributos el prototipo debe permitir la creación de dependencias funcionales entre dichos atributos. Carga Archivo: Se carga un archivo .JSON, con la definición de atributos y dependencias funcionales, los cuales seran cargados al sistema.
Resultado esperado	El sistema muestra en pantalla los atributos y dependencias funcionales cargadas, eliminando atributos duplicados y advirtiendo al usuario en caso de alguna inconsistencia
RESULTADO DE LA EVALUACIÓN	**ACEPTADO**

Cuadro 2.4: Caso de prueba CPU-001 Ingreso de datos al sistema

Identificador (ID) de la Historia	**ID Caso Prueba**
CAM-03	CPU-003
Nombre Caso de Prueba	Comprobar funcionamieno del recubrimiento mínimo
Descripción	Comprobar la correcta automatización del procedimiento de recubirmiento mínimo.
Condiciones de ejecución	Luego de ingresar la lista de atributos y dependencias funcionales al oprimir el boton recubrimiento mínimo"se deben eliminar calcular las dependencias funcionales elementales, eliminar elementos extraños y eliminar las dependencias funcionales redundantes .
Entradas	Una relación previamente definida por el usuario y capturada por el sistema; la que contiene un conjunto de atributos y dependencias funcioales.
Resultado esperado	Conjunto de dependencias funcionales minima o irredundantes.
RESULTADO DE LA EVALUACIÓN	**ACEPTADO**

Cuadro 2.5: Caso de prueba CPU-002 Recubrimiento mínimo

Identificador (ID) de la Historia	**ID Caso Prueba**
CAM-04	CPU-004
Nombre Caso de Prueba	Verificación del cálculo de llaves candidatas
Descripción	A partir del recubrimiento mínimo previamente calculado, se definen las llaves candidatas de la relación
Condiciones de ejecución	Se parte de una relación irredundante y al accionar el boton calcular se deben generar las llaves candidatas
Entradas	Los datos del sistema han sido ingresados previamente. La relación sobre la cual se van a calcular las llaves primarias no tiene dependencias redudants
Resultado esperado	lista de atributos que corresponden a las llaves candidatas
RESULTADO DE LA EVALUACIÓN	**ACEPTADO**

Cuadro 2.6: Caso de prueba CPU-003 Cálculo de llaves

Identificador (ID) de la Historia	**ID Caso Prueba**
CAM-05	CPU-005
Nombre Caso de Prueba	Comprobar las particiones creadas
Descripción	En base a una relación irredundate, se deben calcular las particiones de un conjunto de dependencias funcionales
Condiciones de ejecución	Datos previamente ingresados al sistema, a la relación se le aplica el procedimiento de recubrimiento mínimo, y a partir del nuevo conjunto de dependencias funcionales se calculan las particiones que tendrá el sistema, al accionar el botón calcular
Entradas	Conjunto de dependencias funcionales irredundantes cargadas al sistema.
Resultado esperado	Una lista de dependencias funcionales que corresponden a cada una de las particiones del sistema
RESULTADO DE LA EVALUACIÓN	**ACEPTADO**

Cuadro 2.7: Caso de prueba CPU-004 Cálculo de particiones

Identificador (ID) de la Historia	**ID Caso Prueba**
CAM-06	CPU-006
Nombre Caso de Prueba	Generación del Script del modelo relacional
Descripción	Luego de generar el modelo relacional a partir de los algoritmos, se genera un Script de base de datos siguiendo los lineamientos sql92
Condiciones de ejecución	Oprimir el boton generar Script
Entradas	modelo de base de datos relacional
Resultado esperado	Script de base de datos del modelo de base de datos relacional generado
RESULTADO DE LA EVALUACIÓN	**ACEPTADO**

Cuadro 2.8: Caso de prueba CPU-005 Generación del Script

Parte III

Cierre de la investigación

CAPÍTULO 3

Cierre de la investigación

3.1. Resultados de la investigación

3.1.1. Caso práctico de verificación

Con el fin de llevar a cabo el proceso de verificación del prototipo de software para la automatización del diseño de un esquema de bases de datos relacionales, se ha ingresado al sistema el siguiente caso de estudio, el cual fue tomado de la universidad nacional del sur y plantea Ün ejemplo de aplicación con dependencias funcionales y descomposiciones para un problema real..[E1] documento original se encuentra disponible en: http://cs.uns.edu.ar/ gis/ebd/Archivos/-Clases/Ejemplo

3.1.1.1. Planteamiento del problema

Supongamos que deseamos mantener información sobre los cursados de alumnos de una universidad. Los datos que se desean mantener sobre los cursados incluyen:

3.1.1.2. Dependencias funcionales

Para el esquema del inciso anterior se plantearon las siguientes restricciones o dependencias funcionales que se espera satisfagan todas las instancias de R:

1. $MateriaCodigo$ →MateriaNombre, DptoCódigo, DptoNombre
2. $DptoCodigo$ →DptoNombre
3. $AlumnoLU$ →AlumnoApellido, AlumnoNombre, AlumnoDomicilio, AlumnoTipoDoc, AlumnoNroDoc
4. $AlumnoTipoDoc, AlumnoNroDoc$ →AlumnoApellido, AlumnoNombre, AlumnoDomicilio, AlumnoLU
5. $AlumnoLU, MateriaCodigo, DptoCodigo, AnioAcademico, Cuatrimestre$ →CursadoResultado

Atributo	**Descripción**	**Mapeo a Letra**
MateriaCódigo	Código de materia. Es el atributo que identifica a una materia.	A
MateriaNombre	Nombre de la materia	B
DptoCódigo	Código de departamento académico. Cada departamento tiene su propio código	C
DptoNombre	Nombre del departamento.	D
AlumnoLU	Número de libreta de un alumno. A cada alumno se le asigna un único numero de libreta	E
AlumnoApellido	Apellido del alumno.	F
AlumnoNombre	Nombre del alumno	G
AlumnoDomicilio	Domicilio del alumno	H
AlumnoTipoDoc	Tipo de documento. Puede asumir uno de los siguientes valores: DNI,LC,CI o Pasaporte.	I
AlumnoNroDoc	Numero de documento.	J
Cuatrimestre	Cuatrimestre lectivo. Asume los valores: 1 y 2.	K
AñoAcadémico	Año académico.	L
CursadoResultado	Resultado con que se califica el cursado. Puede tener asociado uno de lossiguientes valores: Aprobado, Desaprobado, Ausente, Sin Calificar	M

Cuadro 3.1: Planteamiento del caso de estudio

6. AlumnoTipoDoc, AlumnoNroDoc, MateriaCódigo, DptoCódigo, AñoAcadémico, *Cuatrimestre* →CursadoResultado

Siguiendo el mapeo de letras las dependencias anteriores podrían reescribirse de la siguiente manera:

1. A →BCD
2. C →D
3. E →FGHIJ
4. IJ →EFGH
5. $ACEKL$ →M
6. $IJACKL$ →M

Esto es, ahora es posible redefinir el problema como:

Dado R un esquema de relación R(A,B,C,D,E,F,G,H,I,J,K,L,M) y sea el siguiente conjunto de dependencias funcionales definido sobre R
F=A →BCD, C →D, E →FGHIJ, IJ →EFGH, $ACEKL$ →M, $IJACKL$ →M

Queda propuesto como ejercicio al lector comprobar que el siguiente es un cubrimiento mínimo reducido para F:

$F_{minimoreducido}$ = A →BC, C →D, E →FGHIJ, IJ →E, $AEKL$ →M

3.1.1.3. Forma Normal

¿Qué forma normal respeta R?. Para poder responder esta consulta es necesario conocer las llaves de R

Simepre	Nunca	Tal vez
AKL	BDFGHM	CEIJ

Cuadro 3.2: Caso de estudio: Llaves candidatas

3.1.1.4. Sintesis (Algoritmo de Sintesis de Bernstein)

De acuerdo al algoritmo de sintesis de Bernstein, se deben realizar los siguientes pasos, para obtener las diferentes tablas del modelo relacional:

1. Agrupar dependencias funcionales en particiones que contengan el mismo implicante:

   - $P_1(A) = A \rightarrow BC$
   - $P_2(C) = C \rightarrow D$
   - $P_3(E) = E \rightarrow FGHIJ$
   - $P_4(IJ) = IJ \rightarrow E$
   - $P_5(AEKL) = AEKL \rightarrow M$

2. Verificar si existen particiones que contengan los mismos atributos que otras ó un subconjunto de estos y agruparlas, para el ejemplo P_4 contiene un subconjunto de los atributos de P_3, por lo que se pueden unir quedando de esta manera:

   - $P_1(A) = A \rightarrow BC$
   - $P_2(C) = C \rightarrow D$
   - $P_3(E) = \mathbf{IJ} \rightarrow FGH\mathbf{IJ}$
   - $P_4(IJ) = \mathbf{IJ} \rightarrow \mathbf{E}$
   - $P_5(AEKL) = AEKL \rightarrow M$

   Uniendo las particiones P_3 y P_4 quedaría de esta manera:

   - $P_1(A) = A \rightarrow BC$
   - $P_2(C) = C \rightarrow D$
   - $P_3(E) = E \rightarrow FGHIJ, IJ \rightarrow E$
   - $P_4(AEKL) = AEKL \rightarrow M$

   Que tambien podría expresarse más claramente en términos de relación $R = \{T, DF\}$ siendo T el conjunto de atributos de la relación y DF el conjunto de dependencias funcionales así:

   - $R_1 : T = \{A, B, C\}$ y $DF = \{A \rightarrow BC\}$
   - $R_2 : T = \{C, D\}$ y $DF = \{C \rightarrow D\}$
   - $R_3 : T = \{E, F, G, H, I, J\}$ y $DF = \{E \rightarrow FGHIJ, IJ \rightarrow E\}$

- $R_4 : T = \{A, E, K, L, M\}$ y $DF = \{AEKL \rightarrow M\}$

3. Finalmente, reemplazando los mapeos de letra, el esquema obtenido quedaría de la siguiente manera:

   - $R_1 =$ Esquema_Materias ($\underline{MateriaCodigo}, MateriaNombre, DptoCodigo$)
   - $R_2 =$ Esquema_Departamentos ($\underline{DptoCodigo}, DptoNombre$)
   - $R_3 =$ Esquema_Alumnos ($\underline{AlumnoLU}, AlumnoApellido, AlumnoNombre, AlumnoDomicilio, AlumnoTipoDoc, AlumnoNroDoc$)
   - $R_4 =$ Esquema_Cursados ($\underline{MateriaCodigo, AlumnoLU, Cuatrimestre, AnoAcademico}, CursadoResultado$)

3.1.2. Aplicación del Caso de Estudio en el prototipo

3.1.2.1. Carga de la relación

El prototipo diseñado permite el ingreso de los detalles de la relación, a través de la interfaz gráfica y/o mediante la carga de un archivo json con la siguiente estructura:

```
1  {
2    "attributes": [
3      "A",
4      "B",
5      "C"
6    ],
7    "functionalDependencies": [
8      {
9        "implicant": [
10         "A",
11         "B"
12       ],
13       "implied": [
14         "C"
15       ]
16     }
17   ]
18 }
```

Para este ejemplo, se hará uso de la carga de archivo json, para lo cual se hace lo siguiente:

1. Al abrir la aplicación se muestra el botón de Çargar Archivo", hacer clic en él.

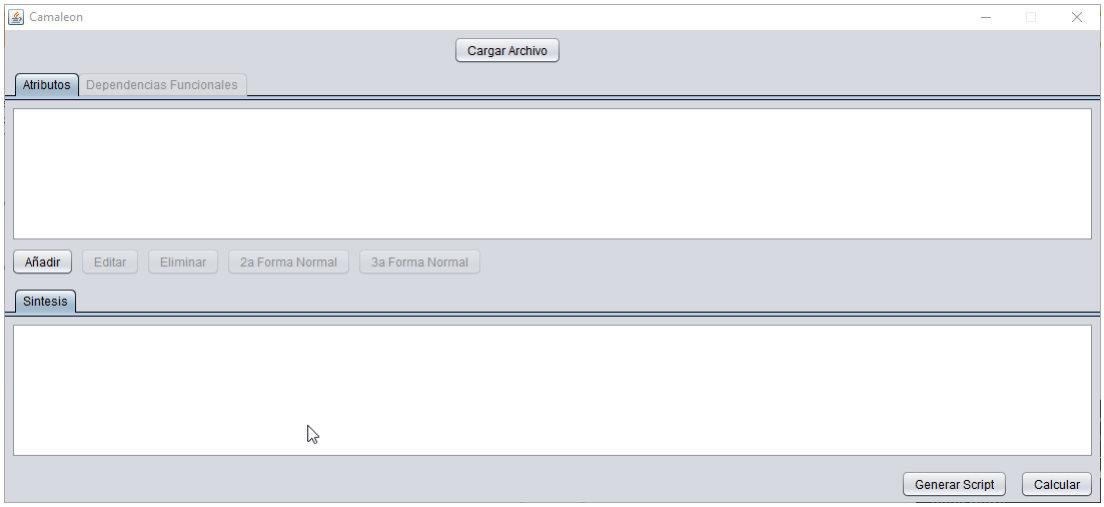


Figura 3.1: Pantalla inicial CAMALEON

2. Se desplegará una ventana para seleccionar el archivo a cargar, para este caso el archivo test.json y se presiona aceptar.

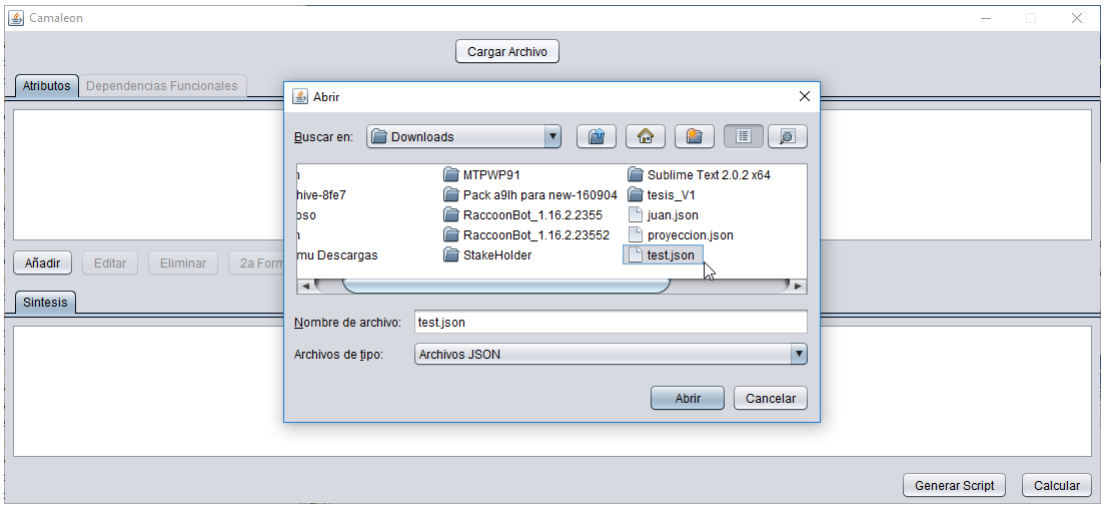


Figura 3.2: Selección de archivo de relación para cargar

3. La aplicación muestra una ventana indicado el resultado del proceso de carga y muestra en pantalla los atributos y dependencias funcionales cargadas.

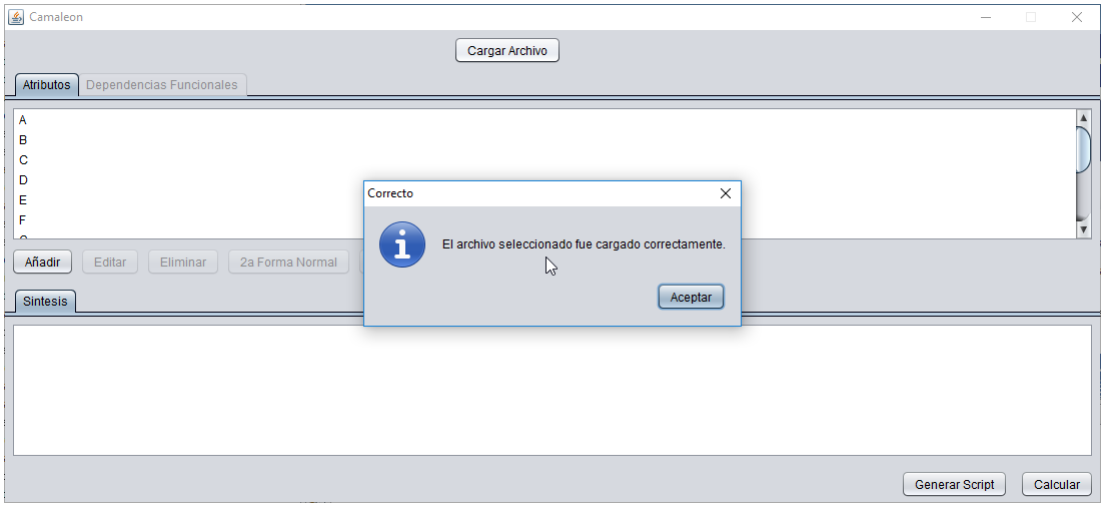


Figura 3.3: Carga exitosa del archivo

3.1.2.2. Calculo de particiones (Algoritmo de sintesis de Bernstein)

Una vez cargada la relación en la aplicación, se habilitará el botón calcular, que permitirá ejecutar los algoritmos de recubrimiento minimo, calculo de llaves candidatas y sintesis de Bernstein, para hallar las particiones del problema.

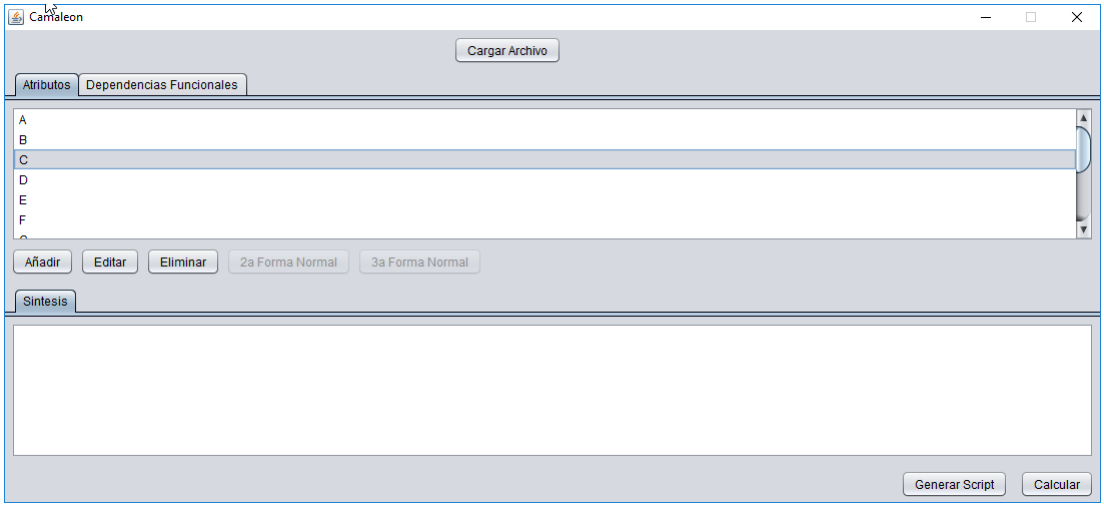


Figura 3.4: Pantalla inicial CAMALEON con relación cargada

Al hacer clic en el botón Çalcular", la aplicación hará el calculo del recubrimiento minimo, hallará las llaves candidatas y finalmente calculará las particiones de la relación y mostrará las particiones en la parte inferior.

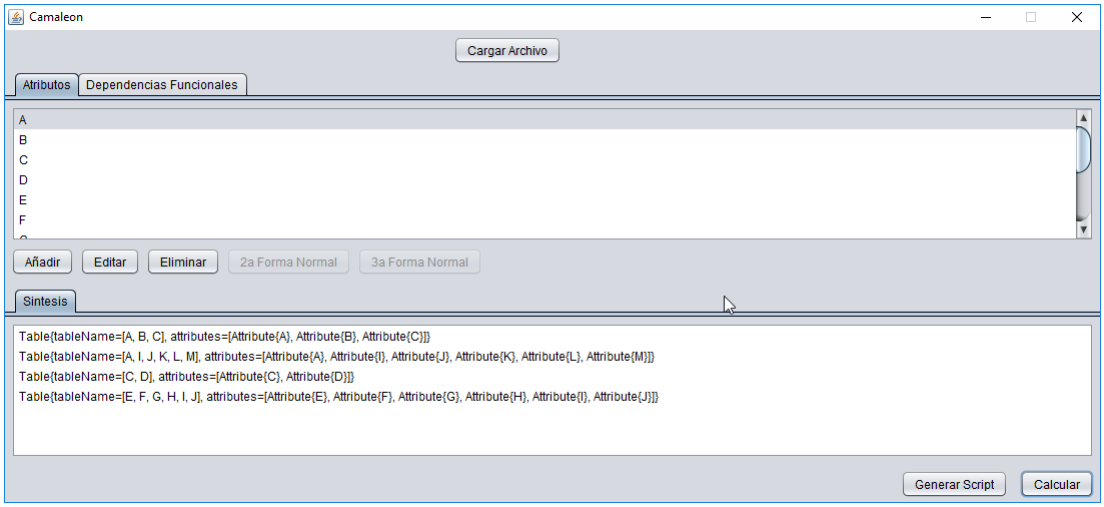


Figura 3.5: Calculo de las particiones de la relación

3.1.2.3. Verificación de resultados

Al comparar las particiones obtenidas con los algoritmos manualmente y los obtenidos por la aplicación, se evidencia que son los mismos.

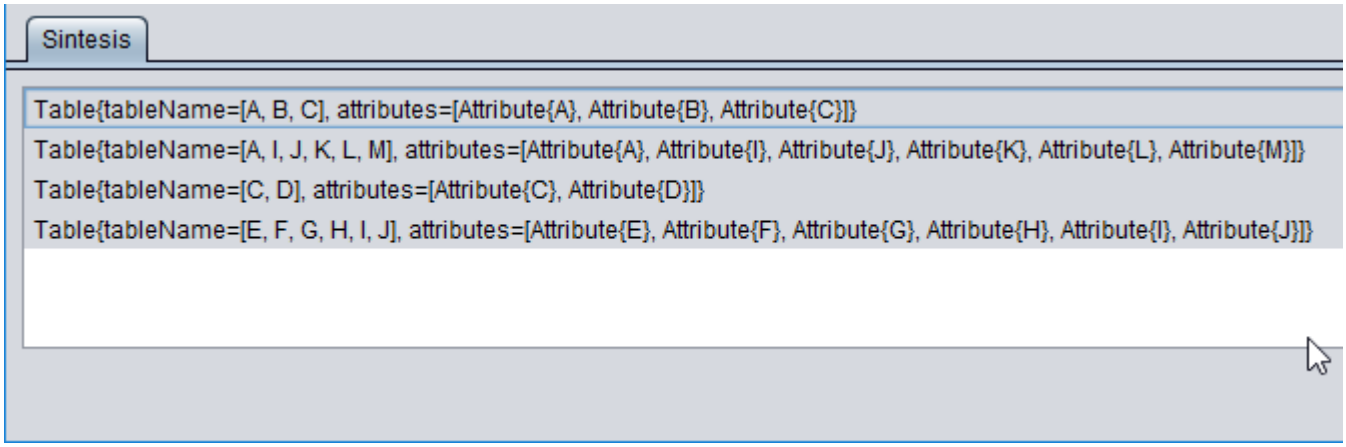


Figura 3.6: Particiones de la relación

- $R_1 : T = \{A, B, C\}$ y $DF = \{A \rightarrow BC\}$
- $R_2 : T = \{C, D\}$ y $DF = \{C \rightarrow D\}$
- $R_3 : T = \{E, F, G, H, I, J\}$ y $DF = \{E \rightarrow FGHIJ, IJ \rightarrow E\}$
- $R_4 : T = \{A, E, K, L, M\}$ y $DF = \{AEKL \rightarrow M\}$

3.2. Conclusiones

3.3. Evaluación de los objetivos

Por medio del análisis riguroso de los algoritmos de recubrimiento mínimo, llaves candidatas y Síntesis de Bernstein es posible generar un modelo de base de datos relacional en tercera forma normal y automatizarlo utilizando funciones programadas en un lenguaje formal computacional, en este caso java.

Teniendo como base el resultado de las funciones programadas que automatizan los procesos algorítmicos y de teoría de conjuntos mencionados anteriormente se logra la generación de un script de Base de datos que cumpla con el estandar SQL92 e implementarlo en un gestor de base de datos compatible.

Al realizar la verificación con un caso de estudio de reducidas características se comprueba la correcta funcionalidad del producto.

3.4. Prospectivo del prototipo

3.4.1. Trabajos de investigación futuros

Con el fin de mejorar el funcionamiento y operación del prototipo descrito en este documento se plantea el desarrollo de una funcionalidad que permita al usuario describir el funcionamiento del sistema por medio requerimientos funcionales, de este modo el prototipo será capaz de capturar dichos requerimientos y transformarlos en dependencias funcionales, lo que facilitará la utilización del prototipo para generar modelos de base de datos relacionales. Así mismo es posible proyectar este proyecto de investigación orientándolo a la aplicación de la ingeniería dirigida por modelos partiendo del modelo que actualmente se genera como resultado del prototipo.

Por otra parte se puede implementar un manual o tutorial en linea que especifique el funcionamiento paso a paso para la utilización de CAMALEON, con el fin de mejorar la experiencia del usuario con el prototipo.

3.4.2. Contraste con el modelo pseudomatemático

Este proyecto de investigación se enfoca en automatizar los algoritmos matemáticos hasta el momento existentes para la generación de un modelo de base de datos relacional, sustentados en la teoría de conjuntos y lógica matemática. Estos procedimientos parten de un conjunto de definiciones previas que se

alinean y organizan mediante una serie de pasos lógico-matemáticos con la finalidad de generar un modelo de base de datos relacional.

Cuando iniciamos el desarrollo del proyecto tuvimos la oportunidad de conocer medianamente la propuesta del ingeniero John Jairo Londoño a través de su libro "modelo seudomatemático para el diseño de las bases de datos relacionales" en el cual plantea una metodología que denomina "más amigable, parecida a las matemáticas" que se basa en dos preguntas fundamentales: ¿Qué se quiere controlar? y ¿Para quién?, de allí concibe nuevos conceptos como "Modelo de la cadena lógica de negocio del sistema" lo cual relaciona con dependencias funcionales, a las en base a una serie de condiciones denomina Dependencia funcional exclusiva (DFE) y dependencia funcional no exclusiva (DFNE) y de este modo desarrolla toda una metodología para el diseño de base de datos relacionales alineadas con el diseño de software.

Si observamos los dos modelos planteados anteriormente podemos llegar a la conclusión de que es necesario una investigación más a fondo acerca de los enfoques que cada uno plantea, lo cual nos puede conducir a una discusión muy interesante acerca de los postulados. Sin embargo sin llegar al detalle y en una exploración superficial, observamos que las dependencias funcionales planteadas por el profesor Londoño y el análisis correspondiente que realiza el modelo seudomatemático son dependencias funcionales externas ya que su análisis se basa en los componentes del sistema y las dependencias funcionales analizadas en los algoritmos tienen como objetivo de estudio los atributos contenidos dentro de los componentes del sistema a lo cual hemos denominado dependencias funcionales internas.

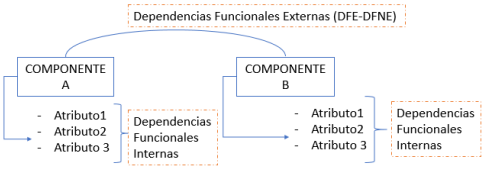


Figura 3.7: Dependencias funcionales externas definidas en el modelo pseudomatematico y dependencias internas planteadas por la logica de conjuntos

Bibliografía

[Korth, 2004] Korth, H. F. (2004). *Introducción a los sistemas de bases de datos.* Pearson Education, séptima edition.

[Kroenke, 2003] Kroenke (2003). *DM, Procesamiento de base de Datos.* Pearson Education, octava edition.

[Londoño, 2016] Londoño, I. J. J. (2016). *Modelo seudomatemático para el diseño de las bases de datos relacionales.* MAGA, primera edition.

[Metodologias agiles, 2016] Metodologias agiles (2016). Scrum. `https://proyectosagiles.org/que-es-scrum`.

[Piattini, 1999] Piattini, M. (1999). *Diseño de Base de datos relacionales.* Alfaomega, octava edition.