UNIVERSIDAD AUTÓNOMA DE CIUDAD JUÁREZ

Instituto de Ingeniería y Tecnología

Departamento de Ingeniería Eléctrica y Computación

Prototipo de Herramienta Normalizadora Mediante el Algoritmo de Síntesis de Bernstein Basado en Dependencias Funcionales a Partir de una Tabla para Bases de Datos con Sintaxis Oracle

Reporte Técnico de investigación presentado por:

Irving Abad Castillo Fierro 113044

Requisito para la obtención del título de:

INGENIERO EN SISTEMAS COMPUTACIONALES

Asesor: M.C. Arnulfo Castro Vázquez

Ciudad Juárez, Chihuahua Noviembre de 2017

Ciudad Juárez, Chihuahua a 6 de noviembre de 2017

Asunto: Liberación de Asesoría

Ing. Jesús Armando Gándara Fernández

Jefe del Departamento de Ingeniería

Eléctrica y Computación

Presente.-

Por medio de la presente me permito comunicarle que después de haber realizado las asesorías correspondientes al reporte técnico **Prototipo de herramienta normalizadora mediante el algoritmo de síntesis de bernstein basado en dependencias funcionales a partir de una tabla para bases de datos con sintaxis Oracle**, del alumno **Irving Abad Castillo Fierro** de la Licenciatura en Ingeniería en Sistemas Computacionales, considero que lo ha concluído satisfactoriamente, por lo que pueden continuar con los trámites de titulación intracurricular.

Sin más por el momento, reciba un cordial saludo.

Atentamente

M.C. Arnulfo Castro Vázquez

Profesor Investigador IIT

Ccp. Cynthia Vanessa Esquivel Rivera

Irving Abad Castillo Fierro

Archivo

Ciudad Juárez, Chihuahua a 6 de noviembre de 2017

Asunto: Autorización de impresión

C. Irving Abad Castillo Fierro

Presente.-

En virtud de que cumple satisfactoriamente los requisitos solicitados, informo a usted que se autoriza la impresión del proyecto de **Prototipo de herramienta normalizadora mediante el algoritmo de síntesis de bernstein basado en dependencias funcionales a partir de una tabla para bases de datos con sintaxis Oracle**, para presentar los resultados del proyecto de titulación con el propósito de obtener el título de Licenciado en Ingeniería en Sistemas Computacionales.

Sin otro particular, reciba un cordial saludo.

M. en C. Estrada Saldaña Fernando

Declaración de Originalidad

Yo, Irving Abad Castillo Fierro, declaro que el material contenido en esta publicación fue generado con la revisión de los documentos que se mencionan en la sección de Referencias y que no ha sido utilizado para obtener otro título o reconocimiento en otra Institución de Educación Superior.

Irving Abad Castillo Fierro

Agradecimientos

Agradezco a mi familia por apoyarme a lo largo de mi carrera profesional, a la Universidad Autónoma de Ciudad Juárez por brindarme una gran variedad de herramientas para mejorar mis estudios universitarios. Agradezco a dios por darme la oportunidad de culminar exitosamente un objetivo más en mi vida.

Dedicatoria

Dedico el presente reporte técnico de investigación a aquellas personas que he conocido a lo largo de mi vida y que han influido en mi persona para ser mejor día a día

Índice de Contenido

Índice de Tablas

Índice de Figuras

Introducción

El empleo de sistemas cada día se hace más frecuente abarcando desde la modernización de las empresas hasta los proyectos académicos, habitualmente el sistema viene acompañado de una base de datos para manejar la información. Para el correcto funcionamiento de un sistema se requiere primordialmente un buen funcionamiento en la base de datos.

El rendimiento de la base de datos al igual que un sistema, será proporcional al diseño y esquematización con la que se habrá de programar. Es por ello que se debe contar con un buen diseño y arquitectura antes de desarrollar la base de datos, un buen diseño indica que se han establecido los estándares para el funcionamiento adecuado [1] .

La normalización surge a raíz de la antigua manera de manejar los atributos ya que se solían agrupar todos en una sola tabla. La normalización optimiza los datos en grupos lógicos de tal manera que se minimizan la cantidad de datos repetidos almacenados [2].

Cuando se tiene una base de datos estandarizada se minimiza la redundancia de datos ya que se agrupan los atributos en relaciones. Esto implica un mejor rendimiento así como un número mínimo de operaciones, reduciendo las posibilidades de que aparezcan anomalías en los datos almacenados [3] .

Entre las anomalías que se pueden llegar a presentar están:

- ✓ Duplicación de datos.
- ✓ Asociación incorrecta de dos atributos de tal manera que imposibilite la eliminación de alguno de ellos.
- ✓ Problemas al insertar datos nuevos.

En la terminología de la base de datos estas cuestiones se denominan anomalías. Una anomalía es un eufemismo para “problema” [4].

El problema se presenta ante la falta del conocimiento o tiempo de los programadores para realizar la normalización ya que dicho concepto se considera de suma importancia para evitar las anomalías dentro de la base de datos [2], tales como se han mencionado duplicación de registros , eliminación sin consentimiento así como problemas para insertar los nuevos registros. Experiencias propias del autor del presente documento fueron motivo para realizar

el prototipo de herramienta. Aunado a esto sin normalización las consultas en la base de datos pueden tomar más tiempo de ejecución por lo que si se manejan grandes cantidades de registros, el tiempo de ejecución de una sentencia requerirá más recursos.

La solución propuesta ante esta problemática fue el desarrollo de un prototipo de herramienta normalizadora por medio del algoritmo de síntesis de Bernstein. El prototipo es capaz de normalizar una tabla a partir de su sentencia de creación de tabla así como sus dependencias funcionales. Posteriormente aplicando el algoritmo se obtiene un *schema* nuevo normalizado en la tercera forma normal a partir de las dependencias funcionales dadas.

I Planeamiento del problema

En el primer capítulo se describirá el problema que se ha encontrado, definiéndolo de tal manera que se podrá apreciar la magnitud de la situación la cual se plantea mejorar. Además se podrán observar un par de antecedentes así como los objetivos para este prototipo de herramienta. Finalmente, la justificación podrá ser visualizada así como las limitaciones y delimitantes que presenta el proyecto.

Actualmente un sistema relacional no implica una base de datos estandarizada, por lo que se debe crear la arquitectura y el diseño a la base de datos como se hace en un sistema [5]. La etapa de diseño en el desarrollo de una base de datos es fundamental para el desarrollo óptimo de la misma.

El problema radica cuando no se crea o no se toma el tiempo suficiente para esta fase del desarrollo. Esto puede ser debido a la falta de conocimiento sobre bases de datos, lo cual conlleva a una serie de anomalías que se pueden presentar al manejar los registros en nuestras bases de datos. Una serie de casos los cuales podrían presentarse son problemas al actualizar un registro o eliminarlo así como su modificación. Estos son problemas comunes cuando se han programado la base de datos sin la debida planeación [6, 7]. Si no se cuenta con una normalización adecuada se pueden llegar a presentar anomalías las cuales son:

- ✓ de Inserción**:** Cada vez que se inserta un registro es necesario repetir los datos en otros atributos.
- ✓ de Modificación: Cada vez se actualiza un registro dentro de un atributo es necesario hacer los cambios en cada una de las tuplas correspondientes a ese atributo.
- ✓ de Eliminación: Si se elimina un atributo asociado a otro se pierde la información del atributo el cual no debería ser eliminado.

Se le llama atributo a cada columna de la tabla y registro al valor de cada uno de los campos en las columnas de una tabla. La problemática continúa al momento de presentarse anomalías debido a que el tiempo de respuesta de la base de datos es mayor. Esto implica la utilizacion extra de recursos de hardware que a su vez es más tiempo requerido en la ejecución. Todos estos factores implican mayor gasto económico.

1.1 Antecedentes

De acuerdo a la investigación referencial, se han realizado sistemas en los cuales se muestra la intensión automatizar el proceso de la normalización. A continuación se presentarán trabajos relacionados que han tratado de solucionar la necesidad de normalizar las bases de datos de una manera automática.

1.1.1 SINORGES

En la Universidad de Ciencias Iinformáticas en la Habana Cuba se ha desarrollado el Sistema para la Integración del Proceso de Normalización de Bases de Datos Relacionales con Gestores de Dases de Datos (SINORGES, por sus siglas en español). El sistema permite la normalización de bases de datos de manera automática [8].

Es un software que permite la normalización por medio de análisis de descomposición y de síntesis, llegando a obtenerse una base de datos normalizada hasta FNBC. Aunque es un nivel aceptable de normalización, la mayoría de las bases de datos requieren una normalización estándar la cual es suficiente con la Tercera Forma Normal [4].

El sistema SINORGES realiza una normalización que podría ser contraproducente con respecto a una base de datos tradicional. Hacer la normalización hasta la Cuarta Forma Normal no garantiza la disminución de las anomalías, incluso aumenta el riesgo de perdida de información [9].

1.1.2 Algoritmo matemático

En el artículo [10] se muestra como problema la falta de normalización dentro de una base de datos en específico. Sin embargo el algoritmo trata de buscar las anomalías por medio de cálculos matemáticos, lo que finalmente resulta con una detección de dichas anomlías de aproximadamente el 67% de ellas.

El algoritmo matemático intenta solventar la falta de normalización por medio de cálculos matemáticos. Sin embargo, no se garantiza su total funcionamiento. El prototipo que se desarrolla se enfocará en la estandarización hasta la Tercera Forma Normal, el cual permite evitar anomalías en las bases de datos.

Uno de los mayores problemas que se tiene cuando no se ha implementado un diseño adecuado en las bases de datos es la redundancia, ya que es más que un solo lugar en la base de datos. Esto puede causar anomalías tanto en la actualización, inserción y borrado de datos [11].

1.1.3 Normalization Tool

La Universidad Griffith en Australia ha desarrollado una herramienta de normalización automática, donde una tabla puede ser normalizada hasta la la forma FBNC. La herramienta solicita los atributos de las tablas así como las dependecias funcionales [12].

1.1.4 Table Analizer

Se trata de una herramienta que sólo puede ser utilizada en bases de datos Access, dicha herramienta busca los atributos únicos de una tabla. Estos valores únicos se etiquetan como claves principales. Si no se identifica ningún valor único, la herramienta crea una clave principal [13].

1.2 Definición del Problema

En el desarrollo de un sistema una de las fases iniciales es el diseño de la base de datos, lo cual a su vez implica el análisis así como el diseño para evitar las anomalías que se han mencionado. Tal como se explica en [2] la normalización es parte fundamental para un diseño exitoso de la base de datos.

Los programadores que no aplican la normalización necesaria son generalmente desarrolladores con pocos conocimientos sobre bases de datos, lo cual hace difícil mantener un nivel óptimo de normalización.

Cuando se manejan los registros en un sistema, cuya base de datos no tuvo una normalización adecuada las anomalías afectan el rendimiento. Esto se convierte en un problema el cual no es generado por la lógica de la interfaz sino del diseño de la base de datos.

Finalmente cabe destacar que los problemas de anomalías que pudiesen presentar las bases de datos pueden afectar a la empresa, institución o cual sea el caso, con pérdida de tiempo en la espera de resultados así como perdidas económicas.

1.3 Objetivos

Dentro de este capítulo se explicarán tanto los objetivos específicos así como el objetivo general del prototipo.

1.3.1 Objetivo General

Normalizar un esquema físico de base de datos oracle por medio de un prototipo de herramienta desarrollado a base del algoritmo de síntesis de Bernstein.

1.3.2 Objetivo Específicos

Los objetivos específicos del protitipo se enlistan a continuación:

- ✓ Desarrollo de analizador sintáctico.
- ✓ Creación de *schema* en 3FN a partir del *schema* original desnormalizado.

1.4 Justificación

El prototipo que se realizó tiene un gran potencial debido a que es una herramienta que podrá ser utilizada por los programadores con pocos conocimientos de bases de datos lo cual ayudará a la arquitectura de sus bases. El prototipo toma un archivo *SQL* con el esquema creado. Éste es analizado para posteriormente aplicar algoritmos de normalización entregando un nuevo archivo con un nuevo esquema el cual se encuentra normalizado, evitando así anomalías futuras.

Este proceso evita los problemas de rendimiento dentro de la base de datos, haciendo así más eficaz el tiempo de respuesta, ahorrando tiempo lo cual a su vez es ahorro de dinero. El prototipo de herramienta cuenta con un nivel máximo de normalización el cual es el recomendado para una base de datos de una pequeña y mediana empresa así como instituciones educativas. [14, 2]

Además de la normalización automática de esquemas por medio de un analizador lexicográfico el usuario será capaz de definir *constraints* nuevos para la base de datos, tales como enteros mayores a *x* número positivo o negativo Esto con el fin de crear y manipular los datos directamente en la base de datos y evitar la programación de scripts, métodos o funciones al momento de validar el dato que se ingresa a la base.

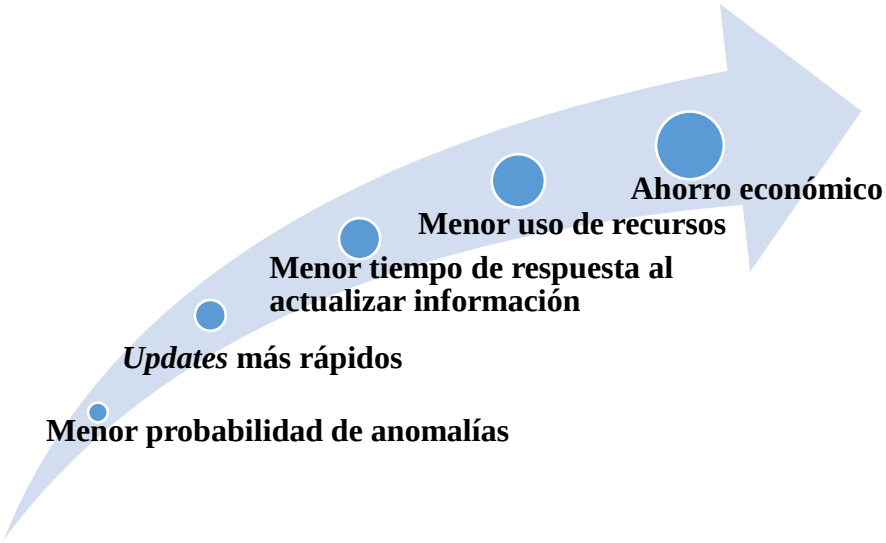


Figura 1 Ventajas de una base de datos normalizada

El prototipo de sistema se encargará de la normalización de manera automática con ayuda de las dependencias funcionales dadas por el usuario. Una dependencia funcional constituye una generación del concepto clave [15, 4]. Dicho concepto clave es resultado de las conexiones entre uno o más atributos. Sin embargo, no se descarta la posibilidad de continuar en un futuro para la creación automática de relaciones entre tablas.

El motivo de este proyecto está orientado a ayudar aquellos desarrolladores que han previamente creado las tablas en su base de datos sin el debido cuidado en el diseño. Así como proporcionar un prototipo normalizador necesario para ayudar a los programadores con pocos conocimientos sobre las bases de datos. Por lo que evitará los problemas que suelen suceder por una mala conceptualización.

II Marco Referencial

En este capítulo se verán los conceptos clave dentro del desarrollo de este prototipo, los cuales ayudarán a un mejor entendimiento sobre el proyecto. Anteriormente se han definido algunos conceptos los cuales serán retomados para su mejor explicación.

Cuando se tiene un buen software para la administración de la información no es suficiente contar con una base de datos. Hay que normalizar dicha base para evitar redundancia. Se debe contar con un diseño que logré manejar y reducir las anomalías [4] que surgen entre los datos. Al proceso de evaluación y correción de estructuras de tablas a fin de minimizar la posibilidad de anomalías en las bases de datos se le llama normalización. El proceso de normalización se divide en las siguientes etapas [16, 17, 4, 15].

- ✓ Primera Forma Normal *FN1*
- ✓ Segunda Forma Normal *FN2*
- ✓ Tercera Forma Normal *FN3*
- ✓ Forma Normal de Boyce-Codd *FNBC*
- ✓ Cuarta Forma Normal *FN4*
- ✓ Quinta Forma Normal *FN5*

Mientras más alto sea el nivel de normalización más uniones se requieren para producir un resultado especifico por lo que el sistema de bases de datos reaccionará más lentamente ante las demandas del usuario [5].

2.1 Marco Teórico

Dentro del marco teórico se podrá conocer más sobre aquellos conceptos que se han estado presentando en capítulos anteriores. Es importante definir los conceptos que se describirán a continuación para entender el propósito del prototipo.

2.1.1 Normalización

El concepto principal de este proyecto es el que se describe a continuación, ya que normalizar es la acción que realiza el prototipo de herramienta sobre la tabla *SQL*. A continuación se muestran un par de conceptos sobre la normalización.

- ✓ Es el optimizar los atributos de una tabla de una manera lógicamente similar, así evitar repeticiones de datos innecesarios y prevenir problemas de incosistencia [18].
- ✓ Es el proceso de organización de datos en tablas de tal manera en la que los resultados de la base de datos sean siempre inequívocos y como se esperaban [19].
- ✓ Proceso en el cual se designan y aplican una serie de reglas denominadas formas normales. Si se cumplen las tres primeras reglas , la base de datos se considera en el máximo nivel necesario de reglas aplicadas a un *schema* para la mayor parte de las aplicaciones [20].

2.1.2 Desnormalización

El concepto desnormalización es una practica común en el desarrollo de una base de datos y un concepto que se necesita entender para dar seguimiento a la lectura del presente documento.

El proceso de tomar un *schema* de antemano normalizado y hacerlo no normalizado se denomina como desnormalización. Diseñadores de bases de datos utilizan la desnormalizacion para ajustar el rendimiento de los sistemas en operaiones criticas en el tiempo.La penalización de mejorar el rendimiento de la base de datos por medio del proceso de la desnormalizacion es el trabajo extra, basándose en tiempo de codificación y tiempo de ejecución, al mantener los datos redundantes como consistentes [15, 1].

2.1.3 Formas Normales

Edgar Frank Codd científico informático fue el creador de las bases de datos relacionales , y en 1970 definió las tres primeras formas normales [18].La normalización funciona por medio de una serie de etapas llamadas formas normales. Estas proporcionan criterios para determinar el grado de vunerabilidad de una tabla a las anomalías lógicas [4, 5]. La normalización abarca niveles donde el más estandarizado es la Quinta Forma Normal (5FN). Las formas normales puede ser observadas en la figura 2.

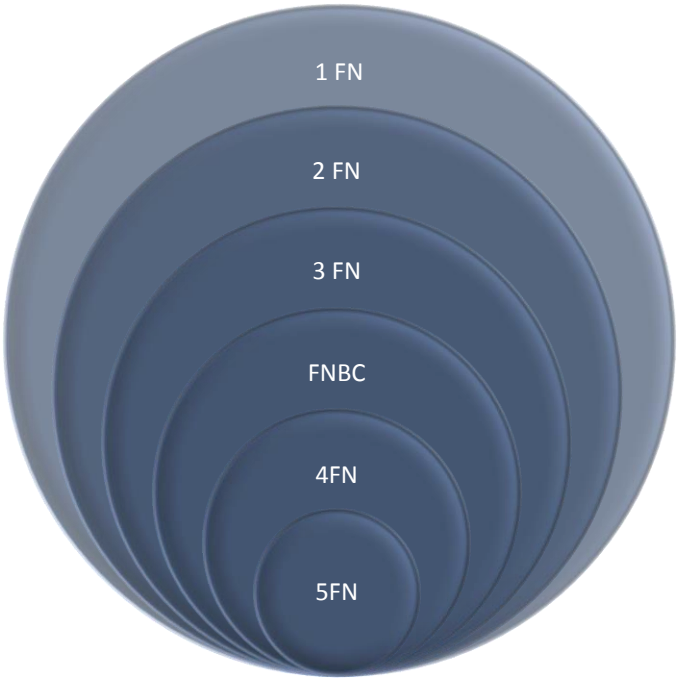


Figura 2 Formas Normales

2.1.3.1 Primera Forma Normal FN1

La Primera Forma Normal indica que la tabla debe contar con sus atributos atómicos o simples, es decir que dichos atributos ya están definidos y no pueden contener otros registros [17, 21].

Cada atributo debe tener un registro único así como un tipo de dato único. Un ejemplo que viola la 1FN es cuando el atributo nombre en una base de datos contiene el apellido y nombre en el mismo registro.

2.1.3.2 Segunda Forma Normal FN2

Para satisfacer la Segunda Forma Normal primeramente se debe satisfacer la Primera Forma Normal y cada atributo debe ser dependiente completamente de la clave primaria [4]. Si no es totalmente dependiente se debe realizar un proceso de descomposición [21, 5].

2.1.3.3 Tercera Forma Normal FN3

Para satisfacer la tercera forma normal primeramente se debe satisfacer la segunda forma normal y no contiene dependencias transitivas. Una dependencia transitiva es aquella donde se tiene un atributo por medio de otro indirectamente [5].

Un ejemplo serían los atributos X , Y , Z con las siguientes dependencias:

Z es funcionalmente dependiente de X	$X \rightarrow Z$	
Y es funcionalmente dependiente de X	$X \rightarrow Y$	La 3FN se encarga de eliminar la dependencia transitiva en este ejemplo es $X \rightarrow Y$ [22, 23].
Y es funcionalmente dependiente de Z	$Z \rightarrow Y$	

2.1.3.4 Forma Normal de Boyce-Codd FNBC

De acuerdo con [23] una tabla se encuentra en la Forma Normal de Boyce-Codd solamente si cada atributo determina completamente a otro.

2.1.3.5 Cuarta Forma Normal FN4

La 4FN implica que la tabla se encuentra en Forma Normal de Boyce Codd y todas las dependencias multivaluadas son una dependencia funcional [23].

2.1.3.6 Quinta Forma Normal FN5

Una base de datos se ecunetra normalizada en la Quinta Forma Normal si está en 4FN y no existen relaciones de dependencias no triviales [23].

2.1.4 Dependencias Funcionales

Las dependencias funcionales son restricciones inherentes a la semántica de los atributos que permiten expresar hechos sobre de la realidad que se está modelando con la base de datos [4]. Las dependencias funcionales que se involucran de primera instancia en el prototipo desarrollado son las dependencias triviales , implícitas así como las transitivas.

Un ejemplo de como funcionan las dependencias funcionales serían dados los conjuntos de atributos X y Y dentro de una relación se establece que Y depende funcionalmente de X. Dicho de otra manera X determina o implica Y solamente si cada valor asociado tiene un valor único [24].

En este ejemplo la dependencia funcional se representa como $X \rightarrow Y$.

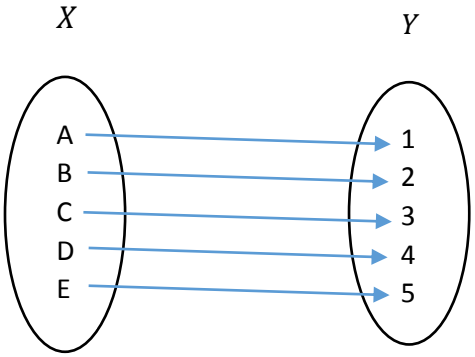


Figura 3 Dependencia Funcional

- ✓ Dependencia Funcional Elemental: Dados dos atributos X y Y forman una dependencia completa y Y como único atributo [23].

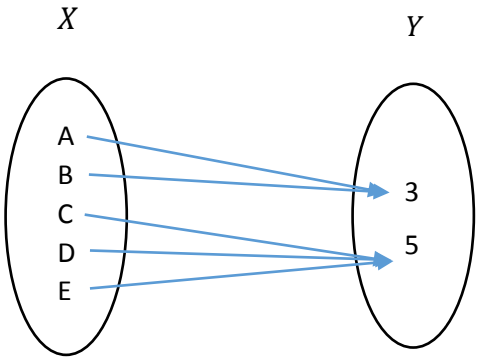


Figura 4 Dependencia Funcional Elemental

- ✓ Dependencia Funcional Transitiva: Sean X, Y, Z tres atributos, de la misma entidad. Si Y depende funcionalmente de X y Z de Y, pero X no depende funcionalmente de Y, se dice entonces que Z depende transitivamente de X [23, 24].

$$X \rightarrow Y \rightarrow Z \ \therefore \ X \rightarrow Y$$

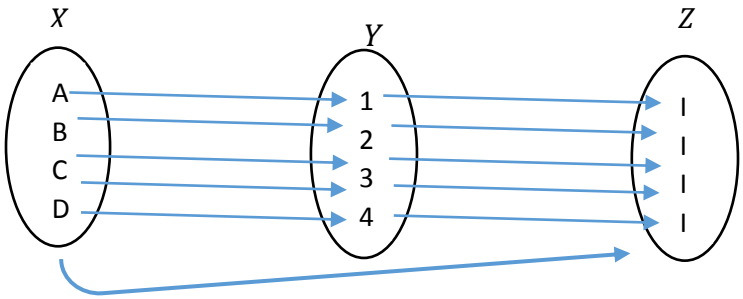


Figura 5 Dependencia Funcional Transitiva

✓ Dependencia Funcional Trivial: Una dependencia $X \rightarrow Y$ es trivial si Y es un subconjunto de X [23, 24]. El ejemplo de la figura 6 muestra al atributo Y siendo parte del subconjunto de X.

$$id\_cliente \rightarrow id\_cliente$$

$$id\ , nombre\_cliente \rightarrow nombre\_cliente$$

Figura 6 Dependencia Funcional Trivial

✓ Dependencia Funcional Implicita: Una dependencia es implícita si dado un conjunto de dependencias un atributo puede ser deducido por otra dependencia funcional [25]. En la figura 7 se puede observar que $nombre\_cliente$ depende funcionalmente de $id\_cliente$ y a su vez $id\_cliente\ , nombre\_cliente$ dependen funcionalmente de id_venta ya que de $id\_cliente$ es deducible $nombre\_cliente$ por lo que no se requiere este último atributo en la primer dependencia funcional ya que se encuentra implícitamente.

$$id\_venta \rightarrow id\_cliente\ , nombre\_cliente$$

$$id\_cliente \rightarrow nombre\_cliente$$

Figura 7 Dependencia Funcional Implicita

✓ Dependencia Funcional Completa: Se presenta cuando un conjunto del atributo X contiene dependencia funcional de Y y además no se puede obtener más atributos de Y de tal manera que se consiga obtener otra dependencia funcional de X. Una dependencia funcional completa se denota por $X \rightarrow Y$ [23, 24]

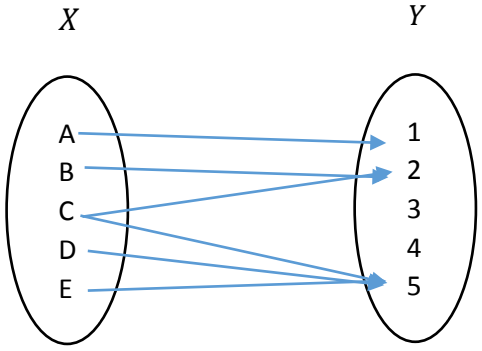


Figura 8 Dependencia Funcional Completa

- ✓ Dependencia Multivaluada: Las dependencias anteriores son basadas en los primeros niveles de normalización hasta la Forma Boyce Codd. Una dependencia multivaluada es una con atributos X, Y, Z en la cual los valores de Y se encuentran sobre cualquier valor de X y Z, el cual los hace más independientes [23, 24].

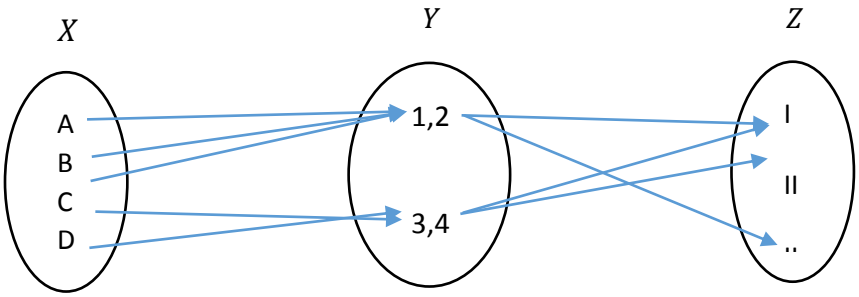


Figura 9 Dependencia Multivaluada

2.1.5 Axiomas de Armstrong

El algoritmo que fue programado para la normalización de datos sigue las reglas de las formas normales . Para la aplicación de las formas normales dentro de un algoritmo se basa en los Axiomas de Armstrong

Los Axiomas de Armstrong son reglas de inferencia las cuales permiten deducir todas las dependencias funcionales que se encuentran en un conjunto de atributos [16]. Los Axiomas de Armstrong son los siguientes [1, 15].

- ✓ Reflexividad:

  Si α se interpreta como un conjunto de atributos y $\beta \subseteq \alpha$, se cumple $\alpha \rightarrow \beta$.
- ✓ Aumentatividad:

  Si $\alpha \rightarrow \beta$ y γ se interpretan como un conjunto de atributos, entonces se cumple $\gamma\ \alpha \rightarrow \gamma\ \beta$.
- ✓ Transitividad:

  Si $\alpha \rightarrow \beta$ y $\beta \rightarrow \gamma$ se interpretan como un conjunto de atributos, entonces se cumple $\alpha \rightarrow \gamma$.

Reglas adicionales derivadas de los axiomas

- ✓ Regla de unión:

  Si $\alpha \rightarrow \beta$ y $\alpha \rightarrow \gamma$ por lo tanto se cumple $\alpha \rightarrow \beta\ \gamma$.
- ✓ Regla de descomposición:

  Si $\alpha \rightarrow \beta\ \gamma$ por lo tanto se cumple $\alpha \rightarrow \beta$ y $\alpha \rightarrow \gamma$.

2.1.7 Covertura mínima de dependencias funcionales

Una cobertura de dependencias funcionales se obtiene cuando todas las dependencias son elementales preservando la misma información. Se denomina cobertura mínima de dependencias funcionales al conjunto mínimo entre un conjunto de dependencias dado del cual a partir de ellas se pueden generar todas las dependencias dadas en un inicio [26].

En la Figura 10 se puede ver el proceso del algoritmo para obtener la cobertura mínima de de pendencias funcionales. [27, 26]

$F_{mín} \leftarrow F$
//Poner las DF en forma canónica:
Sustituir en $F_{mín}$ cada $X \rightarrow \{A_1,...,A_n\}$ por $X \rightarrow A_1, ..., X \rightarrow A_n$
//Eliminar atributos redundantes de la izda
Para cada DF $X \rightarrow A$ de $F_{mín}$
  Para cada atributo B X
    Si $F_{mín} - \{X \rightarrow A\}$ $\{X - \{B\} \rightarrow A\}$ equivalente a $F_{mín}$
    entonces Sustituir en $F_{mín}$ $X \rightarrow A$ por $X - \{B\} \rightarrow A$
//Eliminar las DF que se pueden inferir
Para cada DF $X \rightarrow A$ de $F_{mín}$
  Si $F_{mín} - \{X \rightarrow A\}$ equivalente a $F_{mín}$ **entonces**
    $F_{mín} \leftarrow F_{mín} - \{X \rightarrow A\}$

Figura 10 Algoritmo de cálculo de una cobertura mínima

A continuación se muestra un pequeño ejemplo de la aplicación de este algoritmo contando con los atributos E, G, S y las siguientes dependencias funcionales $E \rightarrow G, G \rightarrow S, E \rightarrow S$.

- ✓ El primero paso a seguir es aplicar a las dependencias funcionales la reglar LHS ,*Left Hand Side,* La cual indica tener atributos atómicos en el lado izquiero. En este caso ya se encuentran valores atómicos en el lado izquierdo. $E \rightarrow G, G \rightarrow S, E \rightarrow S$
- ✓ El siguiente paso es eliminar aquellas dependencias funcionales redundantes con ayuda de los axiomas de Armstrong. Basandose en los axiomas se puede descartar $E \rightarrow S$ por transitividad.

- ✓ Finalmente se unen por llave primaria las dependencias funcionales resultando en este caso $E \rightarrow G, G \rightarrow S$.

2.1.8 Algoritmo de síntesis de Bernstein

Philip A. Bernstein es un cientifico en computacion especializado en la investigacion de base de datos en el grupo de investigacion Microsoft [28].Bernstein en 1976 desarrollo un algoritmo para normalizar a la tercera forma normal a partir de las dependencias funcionales y atributos [29]. En la figura 11 se puede apreciar el proceso del algoritmo mencionado.

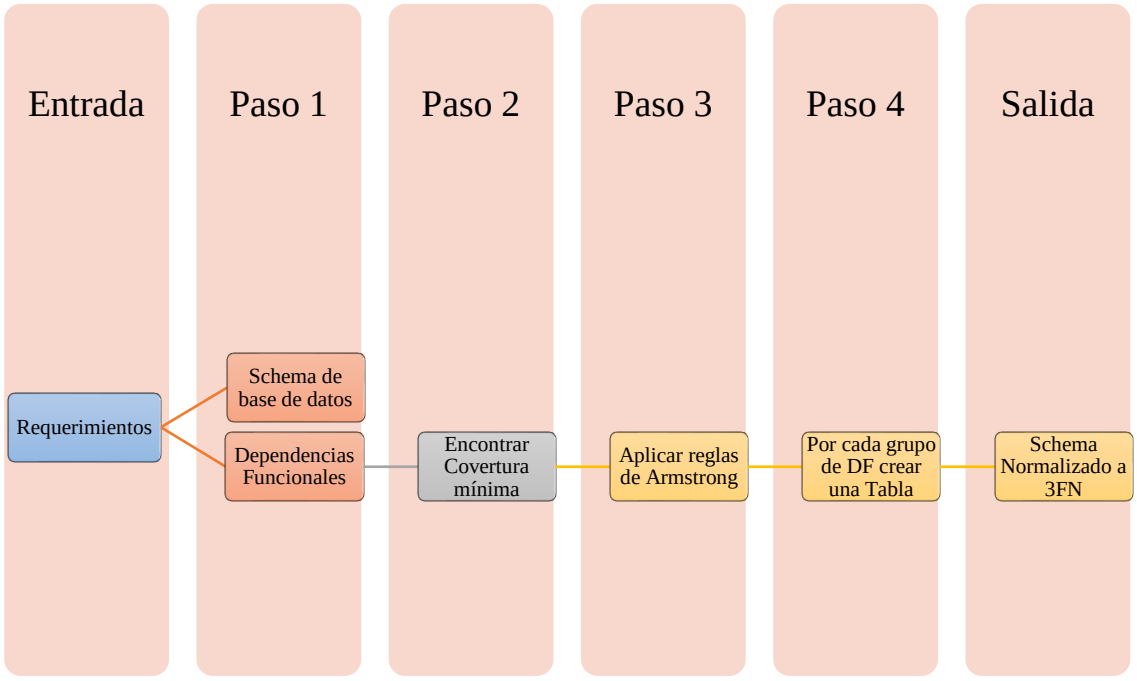


Figura 11 Algoritmo de síntesis de Bernstein

- ✓ Paso 1: El algoritmo de bernstein define un universo en el cual se incluyen los atributos de una tabla y así como sus dependencias funcionales.
- ✓ Paso 2: Se aplica el proceso para obtener la cobertura mínima de dependencias funcionales.
- ✓ Paso 3: Reducción del lado derecho de la dependencia funcional por medio de axiomas de Armstrong.
- ✓ Paso 4: Por cada dependencia funcionale resultantes se creará una tabla nueva.
- ✓ El nuevo *schema* estará normalizado en la tercera forma normal

2.1.9 Analizador léxico

La principal función es leer aquellos caracteres de entrada y elaborar como salida una secuencia de componentes léxicos utilizables en algún lenguaje de programación. [30]

2.1.10 Base de Datos

Es un conjunto de datos almacenados sin redundancias innecesarias accesible simultáneamente por distintos usuarios y aplicaciones [21].

2.2 Marco Tecnologico

Dentro del marco tecnológico se verán todos aquellos conceptos necesarios para entender el como se desarrollo el prototipo de herramienta.

2.2.1 NetBeans

Netbeans es un Compilador gratuito en el cual se pueden desarrollar proyectos en distintos lenguages de programación tales como JAVA ,C , C++ y más. *NetBeans IDE* es el compilador oficial para JAVA 8 con los analizadores de código y convertidores se pueden actualizar las aplicaciones con rapidez y sin problemas para utilizar nuevos constructores de lenguaje Java 8 tales como lambdas operaciones funcionales y referencias de métodos [31].

2.2.1 Versiones Java

Para programar el prototipo de herramienta normalizadora se utilizó el lenguaje de programación Java en su edición estándar 8.

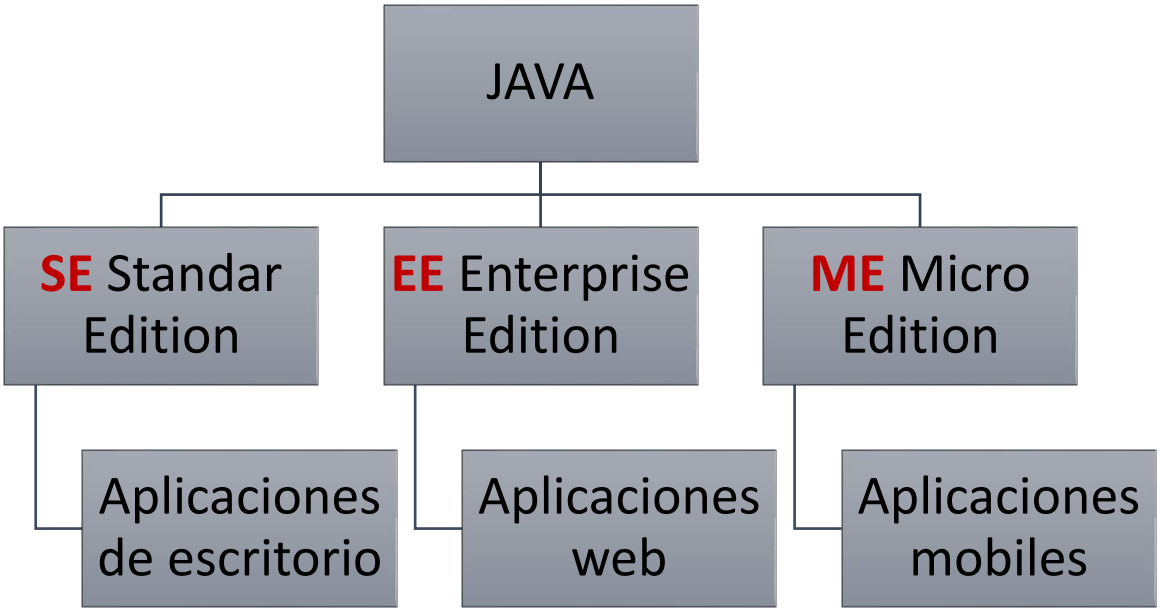


Figura 12 Versiones Java

Java *Stantar Edition (Java SE)* permite desarrollar e implementar aplicaciones de servidor y escritorio ofrece unua rica intefaz de usuario rendimiento, versatilidad portabilidad y seguridad que requieren las aplicaciones actuales.

Java *Enterprise Edition (Java EE)* es el estándar del software empresarial. Cada versión nueva integra nuevas características que se alinean con las necesidades de la industria. Mejora la portabilidad de las aplicaciones y aumenta la productividad de los desarrolladores.

Java *Micro Edition (Java ME)* proporciona un entorno robusto y flexible para aplicaciones que se ejecutan en dispositivos móviles y otros dispositivos integrados. Las aplicaciones basadas en JAVA ME son portátiles a través de muchos dispositivos, aprovechando las capacidades nativas de cada dispositivo [32].

Java Development Kit (*JDK)* es un ambiente de desarrollo para desarrollo de aplicaciones applets y componentes usando el lenguaje de programación Java [32]. Incluye herramientas útiles para el desarrollo y pruebas escritas en Java. En el desarrollo del prototipo de herramienta se utilizó el *JDK 8u111*.

Java Runtime Environment (*JRE*) es una máquina virtual de Java cuya función es comunicar la aplicación que se programa en java con el sistema operativo utilizado. De esta manera, cualquier aplicación puede ejecutarse [32]. El *JRE* es requisito indispensable para desarrollar aplicaciones Java así como para ejecutar el *JDK*.

III Desarrollo del Proyecto

Dentro del tercer capítulo se explica el producto desarrollado, definiendo a lo que se entiende como prototipo en este proyecto. Se darán detalles de las delimitaciones y limitaciones con las cuales el proyecto podría enfrentarse.

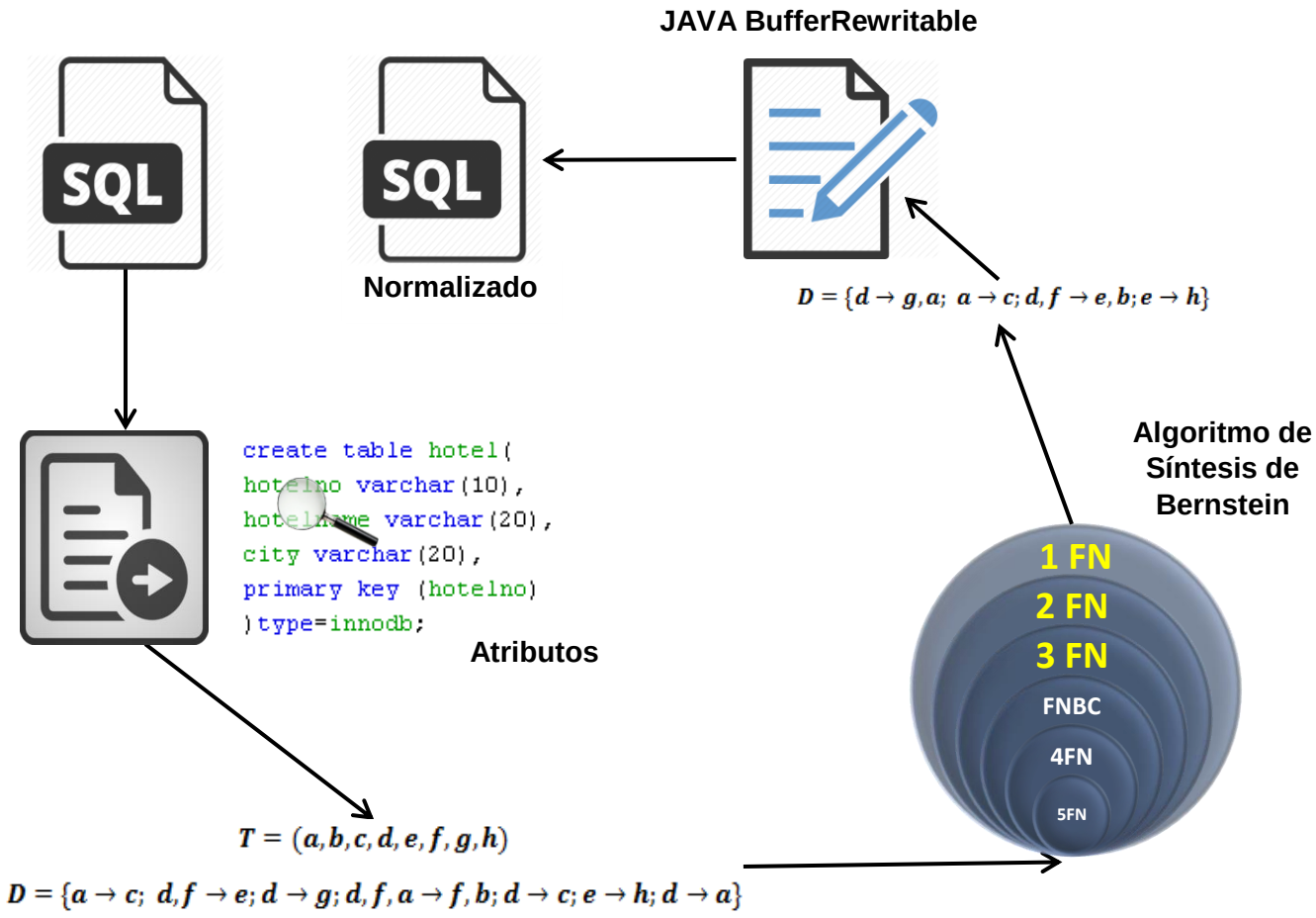


Figura 13 Funcionamiento de la Herramienta Normalizadora

La figura 13 muestra el proceso que se realiza en el prototipo de herramienta desarrollado. El algoritmo se lleva acabo con los atributos y dependencias funcionales como parámetros.

3.1 Descripción de la Solución

La solución al problema previamente planteado en el capítulo I se llevó a cabo mediante el desarrollo tecnológico con un prototipo de herramienta. Un prototipo de herramienta se entiende como una versión del proyecto con aspectos fundamentales para su funcionamiento dando hincapié al seguimiento del mismo.

El producto es un prototipo de herramienta normalizadora para bases de datos funcional dentro de lo que respecta a las delimitaciones que se comentan en el apartado 3.2 del presente documento. Fue desarrollado con el modelo evolutivo [33], con metodos programados basados en el algoritmo de Bernstein el cual a su vez cubre al algoritmo de cobertura mínima de dependencias funcionales. El analizador lexicográfico se desarrolló por medio de un método, utilizando funciones *Java Standar Edition* 7, se vio la posibilidad de usar de igual manera Bison y Flex [34], sin embargo se decidió por funciones Java SE 7.

3.2 Delimitaciones y Limitaciones

Este proyecto cuenta con una limitante potencial para continuar el desarrollo el cual es tiempo. El tiempo en el que se realizó este proyecto no fue suficiente para abarcar todo el trabajo a futuro así como atender las limitantes. Sin embargo en este prototipo se hará énfasis al desarrollo para una primera etapa funcional.

3.2.1 Delimitaciones

Las delimitaciones que se encuentran en este prototipo de herramienta, el cual se base en la normalización automática de bases de datos, son los siguientes.

- ✓ El prototipo ayudará a programadores a normalizar en 3FN bases de datos de manera automática.
- ✓ Obtendrá un archivo *SQL* normalizado a partir de otro archivo *SQL* previamente otorgado con un esquema sin normalización.
- ✓ Herramienta funcional con las especificaciones indicadas.

3.2.2 Limitaciones

Las delimitaciones que se encuentran en este prototipo de herramienta, el cual se base en la normalización automática de bases de datos, son los siguientes.

- ✓ Límite de tiempo.

- ✓ Tiempo límite para el desarrollo del analizador lexicográfico, así como para la programación de los algoritmos de las tres primeras formas normales.
- ✓ Obtención de atributos por medio del analizador lexicográfico de los datos que se encuentran en el archivo *SQL*.
- ✓ El prototipo es sensible a la sintaxis, por lo que si hay un carácter en blanco se tomará como diferente. Los registros debe de estar almacenados de manera limpia

3.3 Formas de validación

Las formas de validación se efectuaron por medio de pruebas en un entorno dentro de un gestor de bases de datos para ejecutar sentencias y registrar tiempos. Teniendo dos *schemas* con los mismos atributos sin embargo uno de ellos estaba normalizado.

Una forma principal de validación fue la reducción en el tiempo de ejecucción de un *query SQL*. La normalización agrupa conjuntos de atributos lógicamente similares, por lo que el tiempo de ejecución es menor al momento de realizar búsquedas dentro de un esquema normalizado que a uno sin normalizar. Otra forma de validación que se presenta es la comparación entre el esquema de base de datos anterior y posterior. Debe contar con una mínima cantidad de atributos por tabla sin perdida de los mismos, con respecto al conjunto de dependencias funcionales inicial. Lo que muestra reducción de redundancias [1, 2, 3].

La forma de validación es con la ayuda del prototipo desarrollado, ya que se realizaron pruebas con un archivo *SQL* piloto para que se obtenga un archivo nuevo normalizado. Se desarrolla el algoritmo de forma manual con los mismos atributos y dependencias funcionales para llegar a un esquema normalizado y comprobar que fue normalizado correctamente el archivo piloto.

Finalmente se generó un *stored procedure* el cual ejecuta un *query* iterativo con una misma sentencia para medir el tiempo que se ejecuta en ambos casos sea el archivo piloto normalizado y sin normalizar. En la figura 14 se puede mostrar el diagrama de flujo que se siguió en el proceso de las validaciones.

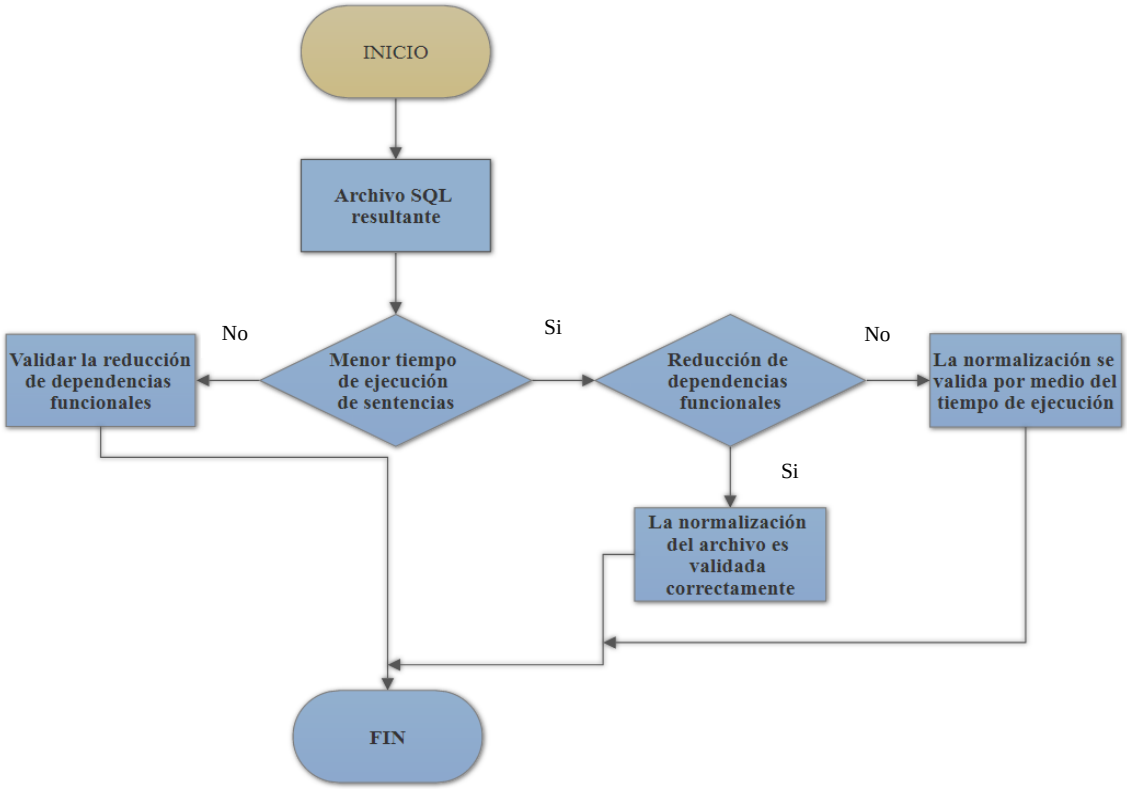


Figura 14 Diagrama de flujo en validaciones

3.4 Metodología

La metodología que se lleva acabo es el modelo evolutivo que se utiliza en desarrollo de software. El prototipo de herramienta fue desarrollado con métodos en el lenguaje de programación Java SE 8.

El modelo evolutivo es un modelo iterativo, la creación de versiones más completas a cada iteración es una de las caractiristicas de este modelo. [35] Las siguientes fases son secciones de este modelo.

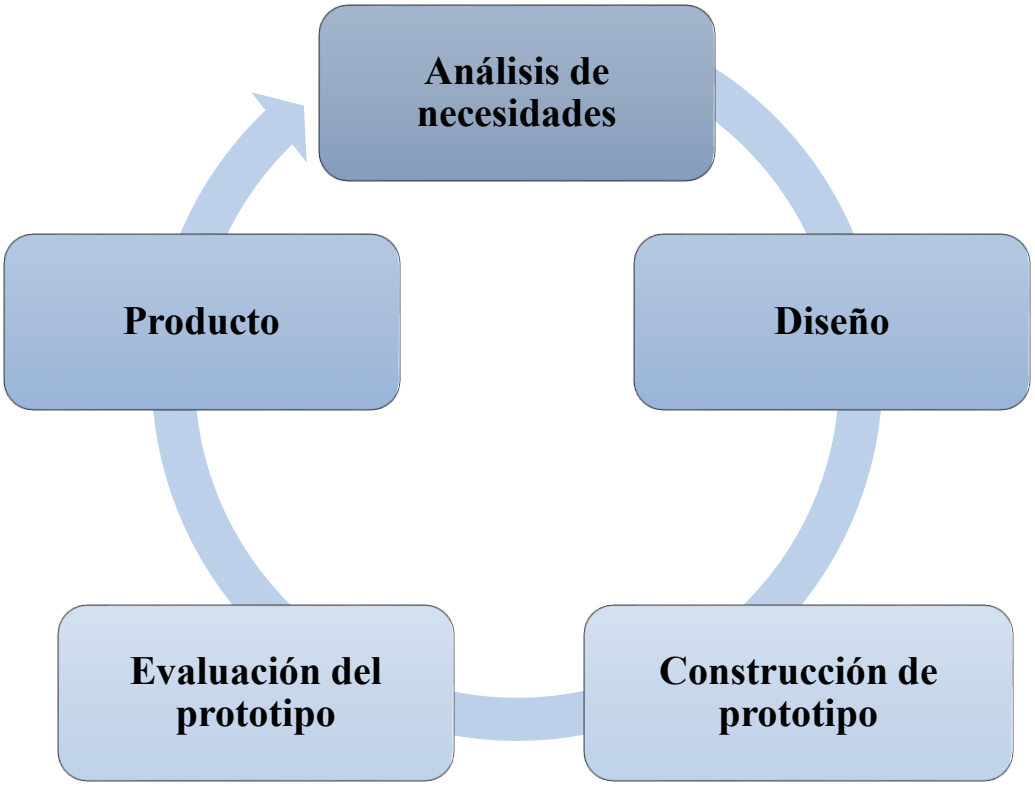


Figura 15 Modelo evolutivo

3.4.1 Análisis

En la cual se llevó a cabo una investigación sobre qué nivel de normalización se es recomendable, para posteriormente enfocarse en el nivel de optimización a alcanzar. Se basó dentro de la investigación referencial viendo hasta que punto de normalización se han realizado en los antecedentes.

3.4.2 Diseño

El prototipo es considerado como desarrollo tecnológico por lo que fue diseñado siguiendo una metodología de desarrollo de software, la cual es la evolutiva mostrada anteriormente. La figura 16 muestra la estructura del prototipo I de herramienta normalizadora el cual cuenta con 3 paquetes. El paquete normalizar contiene las clases FormTool la cual es una interfaz gráfica así como la clase principal que se encarga de llamar a *FoormTool.*

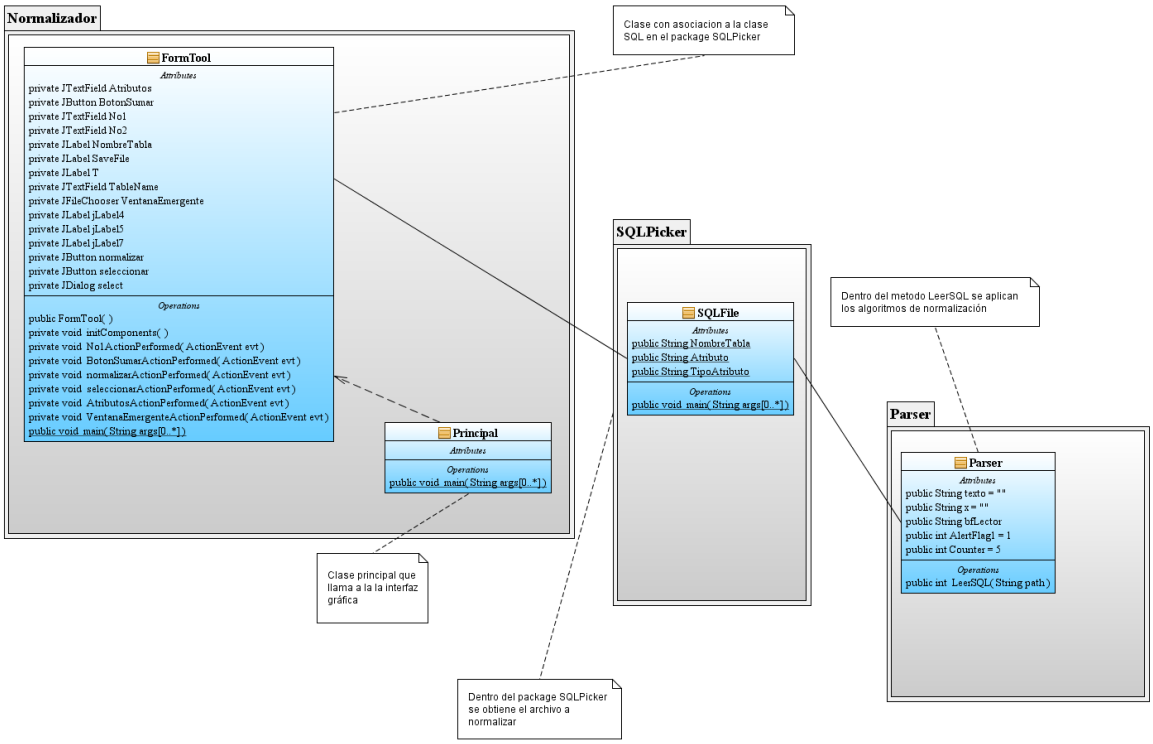


Figura 16 Diagrama de Clases del prototipo I

La clase SQLFile se encarga de preparar al archivo *SQL* para su procesamiento en la clase parser. El usuario puede seleccionar cualquier archivo SQL localizado en el disco duro de su computador.

La clase Parser es donde se aplican los algoritmos de normalización necesarios para llegar a la tercera forma normal. Dentro del método LeerSQL se genera el nuevo documento normalizado.

La clase Parser la cual se encuentra en el paquete Parser contine los atributos que se muestran en la tabla 1.

Clase Parser		
Atributo	Tipo	Descripción
Texto	String	Se almacena X a cada iteración generando el texto del nuevo *schema*.
X	String	Se almacena cada línea nueva del nuevo *schema*
bfLector	String	Maneja cadenas de texto las cuales serán parseadas.
AlertFlag1	int	Cambia de valor cuando un ciclo detecta una dependencia funcional para no volver a parsear.
Counter	int	Cambia de valor cuando un ciclo detecta el fin del texto en el archivo *SQL* lo cual permite iniciar con el proceso de normalización.
LeerSQL	Método	El método lee el archivo lo parsea y aplica los algoritmos para encontrar la cobertura mínima así como el algoritmo de síntesis de Bernstein.

Tabla 1 Descripción de la Clase Parser

En el prototipo II de la herramienta normalizadora se agregaron métodos para la aplicación del algoritmo de bernstein. Todos los métodos necesarios para la normalización se agregaron en el paquete llamado parser en una clase llamada cobertura mínima. El prototipo II acepta hasta tres dependencias funcionales con cuatro atributos permitiendo dos atributos dependientes de otros dos. En la figura 17 se puede observar el diagrama de clases así como en la tabla 2 la estructura de la clase cobertura mínima.

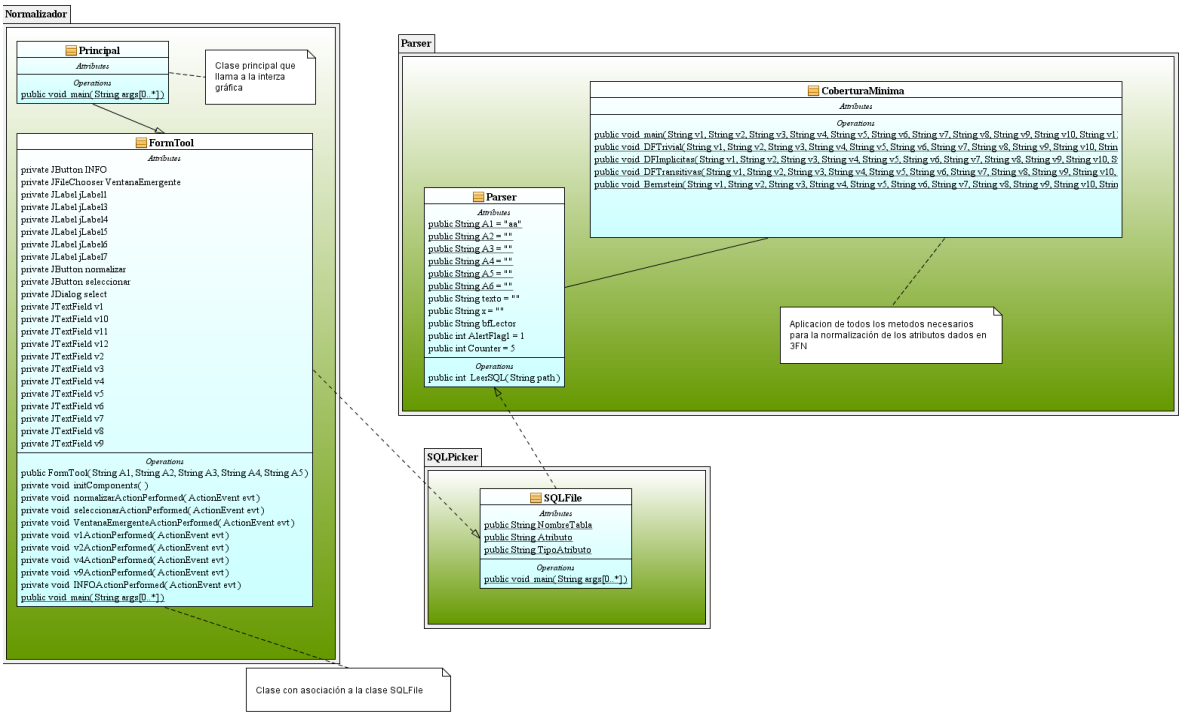


Figura 17 Diagrama de clases del prototipo II

Clase CoberturaMínima		
Atributo	Tipo	Descripción
DFTrivial	Método	El primer método donde las dependencias funcionales son procesadas para eliminar todos aquellos atributos triviales.
DFImplicitas	Método	Recibe los parámetros del método anterior y procesa las dependencias funcionales eliminando atributos deducibles de otras dependencias es decir implícitos.
DFTransitivas	Método	En el prototipo II la transitividad es resuelta de igualmanera en el algoritmo anterior.
Bernstein	Método	Finalmente se aplica la regla de unión por parte de los axiomas de Armstrong para obtener un nuevo *schema* en 3FN.

Tabla 2 Descripción de la Clase CoberturaMínima

3.4.3 Construcción de prototipos

Posteriormente al diseño se comenzó la programación para el desarrollo de la herramienta normalizadora hasta la tercera forma normal por medio de algoritmos tales como el algoritmo de síntesis de Bernstein y el algoritmo de cobertura mínima de dependencias funcionales.

El prototipo I se programó con la finalidad de recibir n cantidad de dependencias funcionales. Sin embargo la lógica que representaba la codificación dinámica del algoritmo de la cobertura mínima sobrepasaba el tiempo estimado dentro del desarrollo de este prototipo.

El prototipo II posteriormente siendo la metodología de programación el modelo evolutivo se replantearon las necesidades y se programó un algoritmo para el procesamiento de un *schema* de base de datos a partir de 3 dependencias funcionales con un máximo de 2 atributos dependientes de 2 atributos más. A continuación se muestra la descripción de los dos prototipos de herramienta normalizadora realizados.

- ✓ El prototipo I fue desarrollado con la finalidad de aplicar el algoritmo de la cobertura mínima y posteriormente el agoritmo de Bernstein teniendo un universo de atributos y una cantidad variable de dependencias funcionales.

  Sin embargo se requería más tiempo del previsto para programar los algoritmos, por lo que se optó por buscar otro método para programar el mismo algoritmo de una manera más rápida. El prototipo I es funcional para todo aquel que desease aplicar el algoritmo de cobertura mínima sobre un *schema* dado, ya que se logró definir en variables dinámicas en forma de arreglos aquellos atributos de todas las dependencias funcionales dadas por lo que se puede iniciar con la eliminación de dependencias triviales como lo indica el algoritmo de la cobertura minima.
- ✓ El prototipo II a diferencia del primero cuenta con un algoritmo estático ya que acepta un máximo de tres dependencias funcionales, teniendo cada una de ellas hasta dos atributos dependientes de otros dos, dando un total de cuatro atributos por dependencia funcional. El prototipo además de preparar las variables para la aplicación del algoritmo de la cobertura minima él mismo lo ejecuta iniciando con la eliminación de dependencias triviales , seguido de dependencias implícitas y finalmente dependencias de transitividad.

Terminando de aplicar el algoritmo de la cobertura mínima se aplica el algoritmo de Bernstein el cual como se comentó en el capítulo III se basa en la creación de una nueva tabla por cada dependencia funcional resultante de la aplicación de la cobertura mínima.

3.4.4 Evaluación del prototipo

Al finalizar con la programación del prototipo se realizaron pruebas con bases de datos no normalizadas para analizar los resultados posteriormente.

Fases	Descripción	Actividades
Análisis de necesidades	Investigación a profundidad de la problemática implicada al no normalizar bases de datos.	Evaluación de requerimientos. Investigación de formas normales. Investigacion de lenguajes de programación necesarios. Analizador de sintaxis
Diseño	Establecer los algoritmos a implementar.	Primera etapa de desarrollo. Diagrama UML
Construcción de prototipo	Obtener un prototipo inicial con funcionalidad.	Diseño de interfaces de usuario Diseño entradas/salidas Programación del analizador lexicográfico
Evaluación del prototipo	Refinación de los requisitos del software a desarrollar.	Pruebas de funcionalidad Notas sobre retroalimentación
Mejora de prototipo	Proceso evolutivo en el que el prototipo es refinado.	Sugerencia de modificaciones Correcciones y cambios
Producto	Desarrollo y entrega del producto terminado.	Versión final de prototipo

Tabla 3 Fases y Actividades

3.4.5 Diagrama de uso de casos

El diagrama de casos es el responsable de documentar los macro requisitros del sistema. La finalidad de un diagrama de caso de uso es el describir la manera en que se usará un sistema, así como describir sus finalidades esenciales de una manera visual [36]. En el diagrama mostrado en la figura 18 se puede apreciar la iteracción que el usuario tiene con respecto al prototipo de herramienta que se realizó.

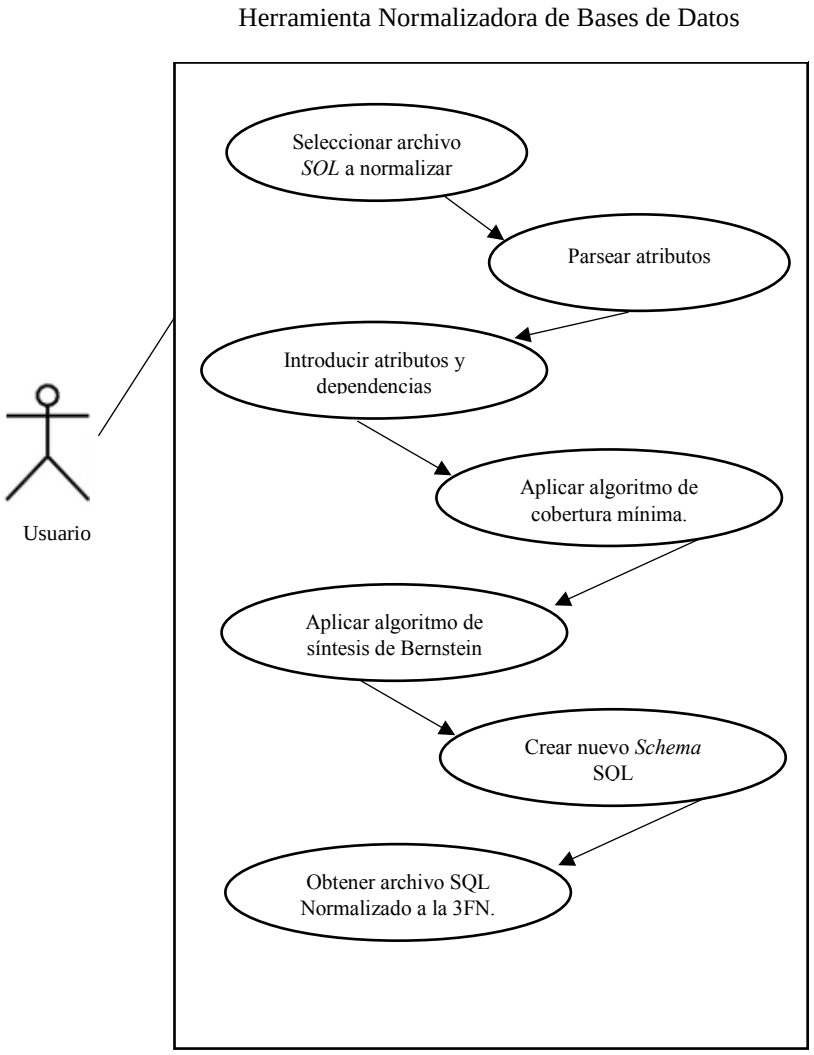


Figura 18 Diagrama de caso de uso

3.4.6 Diagrama de secuencia

Un diagrama de secuencia muestra un ordenamiento en el tiempo al seguir la secuencia de tareas de un sistema, mostrándolas de derecha a izquierda por medio de mensajes. El diagrama de secuencia del prototipo de herramienta normalizadora puede ser apreciado en la figura 19

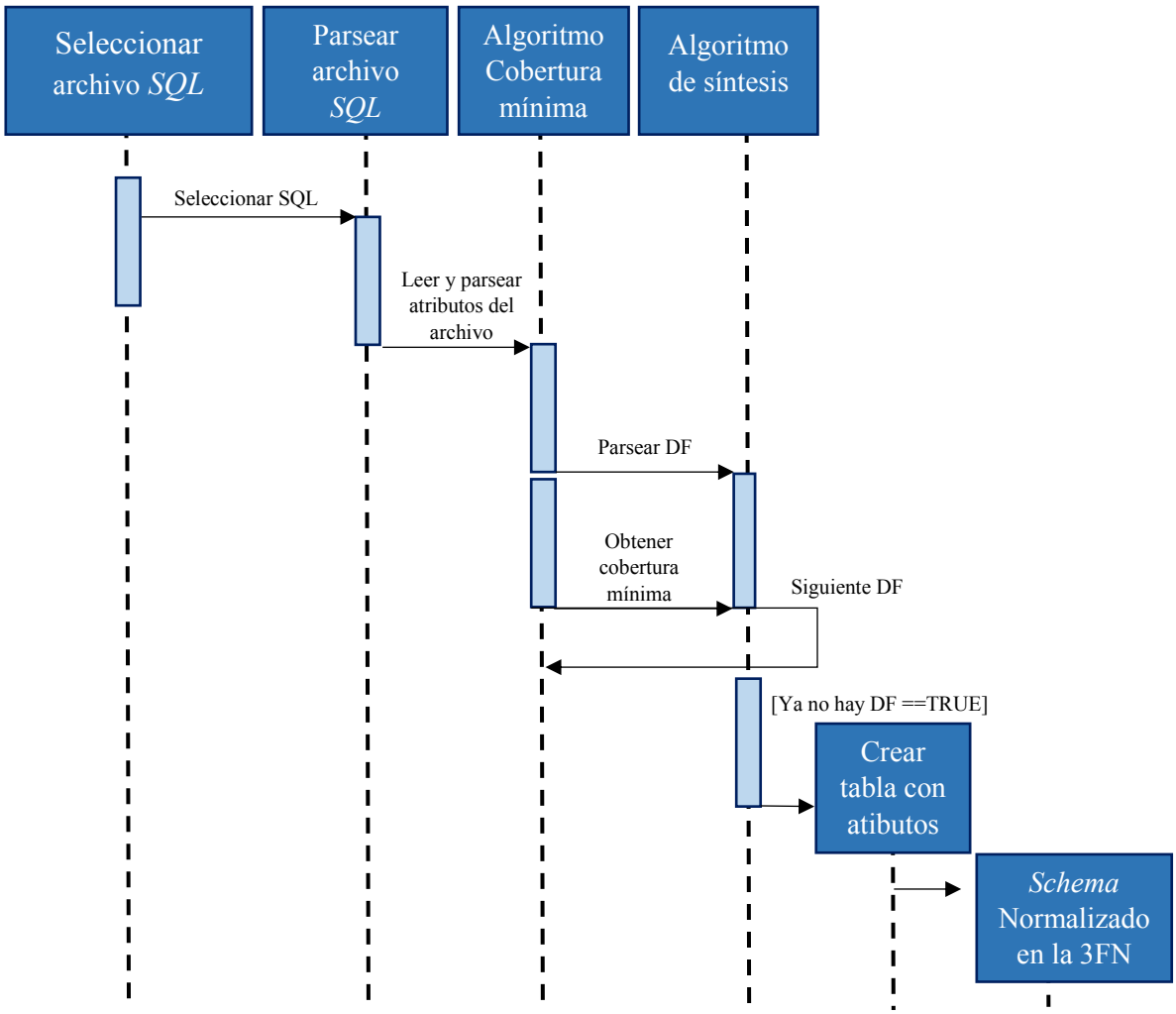


Figura 19 Diagrama de secuencia

IV Resultados y Discusiones

En el capítulo IV se verán los resultados obtenidos a lo largo del desarrollo de este proyecto. Se mostrarán los detalles de resultados obtenidos por el prototipo II el cual es una herramienta normalizadora funcional.

Los resultados son muestra de la ejecución de procedimientos almacenados los cuales llevaron a cabo determinadas instrucciones para poblar las tablas de datos dentro de dos *schemas* normalizado, obtenido con ayuda del prototipo II , y no normalizado.

El *schema* piloto a normalizar fue una tabla llamada ventas con los atributos siguientes:

- ✓ Id *INT*
- ✓ Id_cliente *INT*
- ✓ Nombre_cliente *VARCHAR*
- ✓ Fecha *DATETIME*
- ✓ Costo_total *INT*

El *schema* se demuestra a detalle en la sección de anexos.

Las dependencias funcionales en este *schema* son las siguientes:

$$Id \rightarrow Id\_cliente, Nombre\_cliente$$

$$Id\_cliente \rightarrow Nombre\_cliente\ , Id\_cliente$$

$$Id \rightarrow Fecha, Costo\_total$$

Seleccionando el archivo con el schema previamente mencionado el prototipo II por medio del algoritmo de Bernstein entregó dos tablas con los siguientes atributos.

- ✓ Id *INT*
- ✓ Fecha *DATETIME*
- ✓ Costo_total *INT*
- ✓ Id_cliente *INT*
- ✓ Nombre_cliente *VARCHAR*

Creando de esta manera dos tablas en lugar de una, las cuales pueden ser llamadas ventas lado izquierdo y clientes lado derecho.

4.1 Resultados de ejecución de un update

En la figura 20 se puede apreciar el tiempo de ejecución tomado para la ejecución de un *update* dentro de una tabla del *schema* normalizado. En la parte derecha se aprecia el tiempo total de la ejecución del *update*.

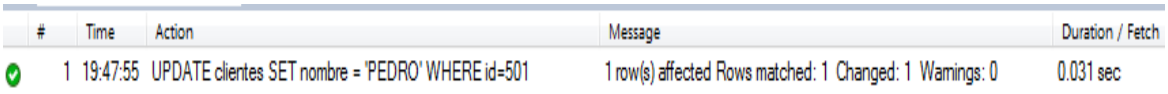


Figura 20 Resultados de un update en tabla normalizada

Se realizaron 100 corridas de esta misma sentencia dando como mínimo en tiempo de respuesta 0.031 segundos y un máximo de 0.164 segundos con un promedio de 0.064 segundos.

En la figura 21 se muestra el mismo update al mismo cliente aplicado en el *schema* antes de ser normalizado y dividido en las dos tablas previamente mostradas.

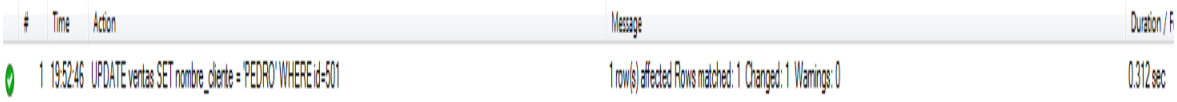


Figura 21 Resultados de un update en tabla no normalizada

Se realizaron 100 corridas de esta misma sentencia dando como mínimo en tiempo de respuesta 0.047 segundos y un máximo de 0.312 segundos con un promedio de 0.078 segundos.

En la tabla 4 se muestran los tiempos a detalle.

Comparación de tiempos		
Tiempos de ejecución de sentencia *update*	Tabla ventas no normalizada	Tabla de *schema* ventas normalizado
Tiempo mínimo de ejecución	0.047	0.031
Tiepo promedio de ejecución	0.078	0.064
Tiempo máximo de ejecución	0.312	0.164

Tabla 4 Tiempos de un update

En la figura 22 se muestra el gráfico representativo de la tabla 4. Como se muestra en el gráfico el update en el *schema* normalizado, color naranja, se muestra un menor tiempo de ejecución tanto en tiempo mínimo , promedio como máximo.

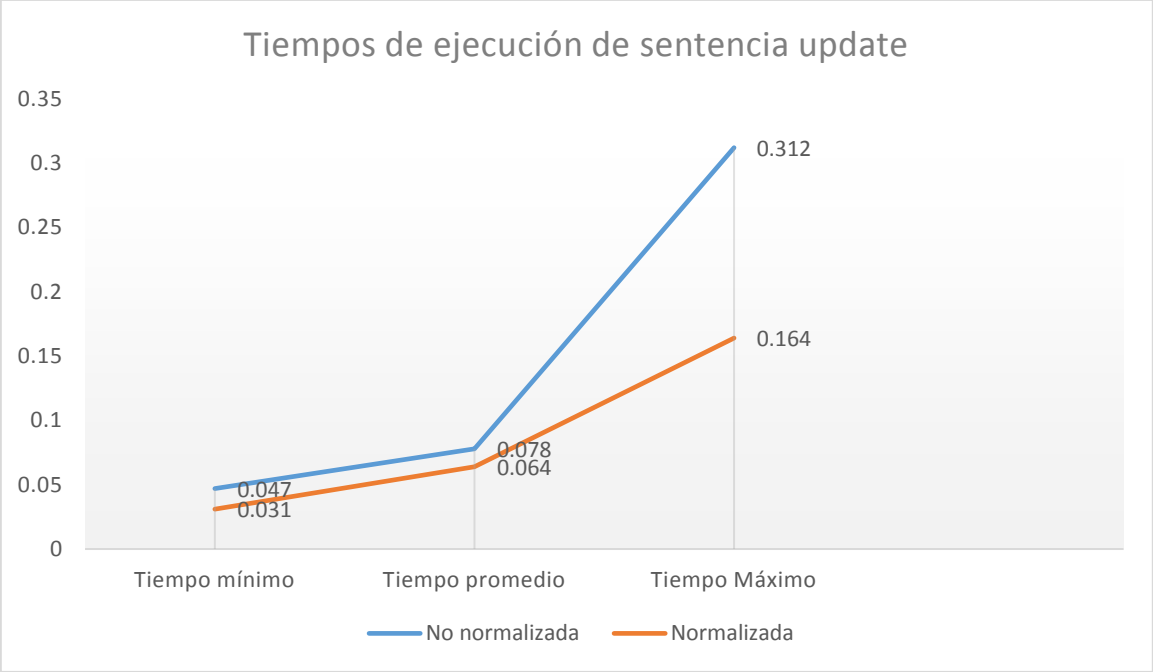


Figura 22 Gráfico comparativo tiempos en update

4.2 Resultados anomalías de eliminación

En el *schema* no normalizado mostrado en la tabla 5, lado izquierdo, se pueden observar los clientes en el mismo entorno que las ventas. En el *schema* mencionado se encuentran 500 clientes de los cuales han hecho al menos 2 compras. Sin embargo existe el cliente 501 el cual solamente registra una compra, si se elimina dicha venta correspondiente al cliente 501 sus datos se perderán sin consentimiento alguno.

Es imposible mantener las información del cliente si él nunca compró algo, ya que no se registraría la venta. Lo mismo acontecerá con aquellos clientes que aparten artículos o no registren el pago total de la venta.

4.3 Resultados redundancia de datos

A continuación se muestra en la tabla 5 los registros clientes del *schema* no normalizado, así como del *schema* normalizado. En el lado izquierdo de la tabla 5 se muestra que el nombre se repite cada vez que se registra una venta de igual manera si se analiza que cada cliente puede tener su dirección así como teléfono estos se repetirían de la misma manera en la que el nombre lo hace. Es así como la redundancia aparece en la tabla ventas antes de ser normalizada.

Tablas

Ventas no normalizada (Incluyendo clientes)

id	id_cliente	nombre_cliente	fecha	total
1	1	JUAN	2017-10-24 19:30:18	301
2	2	Nombre Cliente 2	2017-10-24 19:30:18	302
3	3	Nombre Cliente 3	2017-10-24 19:30:18	303
4	4	Nombre Cliente 4	2017-10-24 19:30:18	304
5	5	Nombre Cliente 5	2017-10-24 19:30:19	305
6	1	JUAN	2017-10-24 19:30:51	301
7	2	Nombre Cliente 2	2017-10-24 19:30:51	302
8	3	Nombre Cliente 3	2017-10-24 19:30:51	303
9	4	Nombre Cliente 4	2017-10-24 19:30:52	304
10	5	Nombre Cliente 5	2017-10-24 19:30:52	305
11	6	Nombre Cliente 6	2017-10-24 19:30:52	306
12	7	Nombre Cliente 7	2017-10-24 19:30:52	307
13	8	Nombre Cliente 8	2017-10-24 19:30:52	308
14	9	Nombre Cliente 9	2017-10-24 19:30:52	309
15	10	Nombre Cliente 10	2017-10-24 19:30:52	310
16	11	Nombre Cliente 11	2017-10-24 19:30:52	311
17	12	Nombre Cliente 12	2017-10-24 19:30:52	312
18	13	Nombre Cliente 13	2017-10-24 19:30:52	313
19	14	Nombre Cliente 14	2017-10-24 19:30:52	314
20	15	Nombre Cliente 15	2017-10-24 19:30:52	315
21	16	Nombre Cliente 16	2017-10-24 19:30:52	316
22	17	Nombre Cliente 17	2017-10-24 19:30:52	317
23	18	Nombre Cliente 18	2017-10-24 19:30:52	318
24	19	Nombre Cliente 19	2017-10-24 19:30:52	319
25	20	Nombre Cliente 20	2017-10-24 19:30:52	320
26	21	Nombre Cliente 21	2017-10-24 19:30:52	321
27	22	Nombre Cliente 22	2017-10-24 19:30:52	322
28	23	Nombre Cliente 23	2017-10-24 19:30:52	323
29	24	Nombre Cliente 24	2017-10-24 19:30:52	324
30	25	Nombre Cliente 25	2017-10-24 19:30:52	325
31	26	Nombre Cliente 26	2017-10-24 19:30:52	326
32	27	Nombre Cliente 27	2017-10-24 19:30:52	327
33	28	Nombre Cliente 28	2017-10-24 19:30:52	328
34	29	Nombre Cliente 29	2017-10-24 19:30:52	329
35	30	Nombre Cliente 30	2017-10-24 19:30:52	330
36	31	Nombre Cliente 31	2017-10-24 19:30:52	331

Clientes normalizada

id	nombre
1	JUAN
2	Nombre Cliente 2
3	Nombre Cliente 3
4	Nombre Cliente 4
5	Nombre Cliente 5
6	Nombre Cliente 6
7	Nombre Cliente 7
8	Nombre Cliente 8
9	Nombre Cliente 9
10	Nombre Cliente 10
11	Nombre Cliente 11
12	Nombre Cliente 12
13	Nombre Cliente 13
14	Nombre Cliente 14
15	Nombre Cliente 15
16	Nombre Cliente 16
17	Nombre Cliente 17
18	Nombre Cliente 18
19	Nombre Cliente 19
20	Nombre Cliente 20
21	Nombre Cliente 21
22	Nombre Cliente 22
23	Nombre Cliente 23
24	Nombre Cliente 24
25	Nombre Cliente 25
26	Nombre Cliente 26
27	Nombre Cliente 27
28	Nombre Cliente 28
29	Nombre Cliente 29
30	Nombre Cliente 30
31	Nombre Cliente 31
32	Nombre Cliente 32
33	Nombre Cliente 33
34	Nombre Cliente 34
35	Nombre Cliente 35
36	Nombre Cliente 36

Tabla 5 Comparación de tabla clientes

4.4 Resultados anomalías de actualización

Tomando como ejemplo la tabla 5, si en el *schema* no normalizado se actualiza el nombre de un cliente este debería ser actualizado en cada una de las ventas registradas de lo contrario los datos serán inconsistentes. En la figura 22 se mostró un gráfico comparativo de tiempos de ejecución de un *update. Donde* se puede apreciar que se toma más tiempo aplicar el *update* en un *schema* sin normalización ya que puede haber varios registros con las condiciones que acreditan el *update,* por el contrario en una tabla normalizada será un solo registro el que se deberá actualizar.

V Conclusiones

En el último capítulo se podrán observar todas aquellas conclusiones a las que se llegarón con el presente proyecto el cual fue desarrollado a lo largo de 10 meses. De la misma manera se explicarán las futuras investigaciones que se pueden realizar a partir de este proyecto para aquellos a quienes les interesa el área.

5.1 Conclusiones con respecto al objetivo general

El prototipo de herramienta normalizadora cumple con el objetivo principal de este proyecto el cual es normalizar una tabla de base de datos.

- ✓ A través del algoritmo de Bernstein se puede normalizar un *schema* en la tercera forma normal.
- ✓ Eliminar las dependencias funcionales implícitas dado a un conjunto de dependencias funcionales ayuda a obtener un *schema* en la segunda forma normal.
- ✓ Un *schema* puede ser normalizado a partir de sus dependencias.
- ✓ Una base de datos desnormalizada puede ejecutar sentencias *select incluso más rápido* , debido a que no se necesitan unir tablas con *joins* ya que se tienen los atributos en la misma tabla. El precio de dicha ventaja es pagado con la probabilidad de anomalías de actualización, eliminación e inserción. Y dicha ventaja en el tiempo de ejecución de *query* finalmente terminará consumiendo más recursos del sistema al momento de ejecutar senctencias de manipulación de datos *SQL* .

5.2 Trabajos futuros

El prototipo II , el cual es presentado para su evaluación, cumple con el objetivo el cual fue normalizar un *schema* a partir de una tabla de base de datos. Algunos de los proyectos e investigaciones que pueden seguir este proyecto se mencionan a continuación.

- ✓ Método de búsqueda de dependencias funcionales a partir de una tabla con datos. Lo cual permitiría evitar al usuario ingresar las dependencias funcionales de su *schema*.
- ✓ Definir automáticamente *constraints, index* dentro del método actual para la creación del nuevo schema.
- ✓ Definir relaciones entre las tablas previamente normalizadas.

Referencias

[1] H. Korth, A. Silberschatz y S. Sudarshan, Database System Concepts, Singapore: McGraw-Hill, 2006.

[2] M. Poolet, «Why You Need Database Normalization,» Febrero 1999.

[3] C. B. Thomas Connolly, Database Systems: A practical approach to Design, Implementation, and Management, United States of America: Pearson Education Inc, 2015.

[4] R. Stephens, «Diseño de Bases de Datos,» Madrid, Anaya Multimedia, 2009, pp. 187 -224.

[5] C. C. Peter Rob, Sistemas de bases de datos Diseño, implementación y administración, México D.F: THOMSON, 2005.

[6] M. A. Poolet, «Why You Need Database Normalization,» *SQL ServePror,* Febrero 1999.

[7] M. Piattini, M. Esperanza, C. Calero y B. Vela, Tecnología y diseño de Bases de Datos, D.F: Alfaomega, 2007.

[8] L. M. E. P. M. M. S. Yanet Espinal Martin, «Sistemas para la integración del proceso de normalización de bases de datos relacionales con gestores de bases de datos (SINORGES),» Octubre 2008. [En línea]. Available: http://www.redalyc.org/articulo.oa?id=133117498004.

[9] S. B. N. Ramez Elmasri, Fundamentos de sistemas de base de datos, Madrid España: Pearson Education S. A, 2007.

[10] M. B. Rodrigo Costas, «Algoritmos para solventar la falta de normalización de nombres de autores en los estudios bibliométricos,» Septiembre 2006. [En línea]. Available: http://www.scielo.org.mx/scielo.php?pid=S0187358X2007000100002&script=sci_arttext&tlng=pt.

[11] J. G. Raghu Ramakrishnan, Database Management Systems, New York: McGraw-Hill , 2003.

[12] J. Wang, 2015. [En línea]. Available: http://www.ict.griffith.edu.au/~jw/normalization/ind.php#editAttibutes.

[13] Microsoft, 2017. [En línea]. Available: https://support.microsoft.com/es-co/help/292799/how-the-table-analyzer-wizard-works.

[14] A. Fuller, «techrepublic,» Diciembre 2006. [En línea]. Available: http://www.techrepublic.com/article/normalization-how-far-is-far-enough/.

[15] A. Silberschatz, H. F. Korth y S. S. , Fundamentos de bases de datos, Aravanca Madrid España: McGraw-Hill, 2002.

[16] C. J. Date, de *Introduccion a Los Sistemas de Bases de Datos*, Mexico, Pearson, 2001.

[17] D. M. Kroenke, Procesamiento de bases de datos, México D.F: Prentice Hall, 2000.

[18] M. d. C. G. Fuentes, Bases de Datos, Mexico: Publidisa Mexicana S. A. de C.V., 2013.

[19] TechTarget, «Database Normalization,» 2016. [En línea]. Available: http://searchsqlserver.techtarget.com/definition/normalization.

[20] S. Microsoft, «Fundamentos de la normalización de bases de datos,» 2017. [En línea]. Available: https://support.microsoft.com/es-co/help/283878/description-of-the-database-normalization-basics.

[21] M. Á. G.-N. E. L. E. G. C. G. Irene Luque Ruiz, Bases de Datos Desde Chen hasta Codd con ORACLE, Madrid España: Alfaomega Grupo Editor, 2004.

[22] S. C. S. P. R. T. Paolo Atzeni, Database Systems concepts, languages and architectures, Berkshire Inglaterra : McGraw-Hill, 1999.

[23] A. Bueno, 2012. [En línea]. Available: http://www.abcsoftperu.com/descargas/curso02.html.

[24] J. M. Piñeiro Gómez, Diseño de bases de datos relacionales, España: Paraninfo, 2014.

[25] R. Godin, Systèmes de gestion de base de données par l'exemple, LOZE-DION, 2006.

[26] R. Soulé, «Functional Dependencies and Finding a Minimal Cover,» [En línea]. Available: http://www.inf.usi.ch/faculty/soule/teaching/2014-spring/cover.pdf. [Último acceso: 2017].

[27] C. D. A. Jaime, «Universidad de la Rioja,» 2005. [En línea]. Available: http://www.unirioja.es/cu/arjaime/Temas/05.Normalizacion.pdf.

[28] P. Bernstein, «researchgate,» Microsoft, 2016. [En línea]. Available: https://www.researchgate.net/profile/Philip_Bernstein.

[29] P. Bernstein, «National University of Singapure,» 1976. [En línea]. Available: https://personal.comp.nus.edu.sg/~lingtw/papers/bernstein.pdf.

[30] K. Daviana, 2011. [En línea]. Available: http://www.kramirez.net/RI/Material/Presentaciones/Analizador%20Lexico.pdf.

[31] O. Corporation, «netbeans.org,» 2017. [En línea]. Available: www.netbeans.org.

[32] Oracle, «oracle,» 2017. [En línea]. Available: www.oracle.com/technetwork/java/embedded/overview/index.html.

[33] R. S. Pressman, Ingeniería del Software un Enfoqie ´ráctico, México, D. F.: Mc Graw Hill, 2010.

[34] J. R. Levine, Flex & Bison, Sebastopol, CA: O´REILLY, 2009.

[35] R. S. Pressman, Ingeniería del software, 6ta ed., D.F: McGraw-Hill, 2005.

[36] P. Kimmel, Manual de UML, CDMX: Mc Graw Hill, 2008.

Anexos

Anexo A Clase principal

```
/* /*  2017
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package Normalizador;

import Parser.Parser;

/**
 *
 * @author Irving
 */
public class Principal {
    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        //leer l = new leer();
        //System.out.println(l.LeerSQL("C:\\Users\\Irving\\Desktop\\Archivo.txt"));
        FormTool v = new FormTool("id","id_cliente","Nombre_Cliente","Fecha","Total");
        v.setTitle("Herramienta Normalizadora por Tabla de BD");
        v.setVisible(true);
        v.setSize(550, 300);

        System.out.println("Impresion desde el Main traido el A1 "+Parser.A1);
    }
}
```

Anexo B Formulario principal

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package Normalizador;

import Parser.Parser;
import Parser.CoberturaMinima;
import java.awt.Dimension;
import java.io.BufferedWriter;
import java.io.File;
import java.io.FileWriter;
import java.io.IOException;
import java.io.PrintWriter;
import javax.swing.JFileChooser;
import javax.swing.JOptionPane;

/**
 *
 * @author Irving
 */
public class FormTool extends javax.swing.JFrame {

    /**
     * Creates new form suma
     */
    public FormTool(String A1,String A2,String A3,String A4,String A5) {
        initComponents();
    }
```

```
/**
 * This method is called from within the constructor to initialize the form.
 * WARNING: Do NOT modify this code. The content of this method is always
 * regenerated by the Form Editor.
 */
@SuppressWarnings("unchecked")
// <editor-fold defaultstate="collapsed" desc="Generated Code">
private void initComponents() {

    select = new javax.swing.JDialog();
    VentanaEmergente = new javax.swing.JFileChooser();
    normalizar = new javax.swing.JButton();
    seleccionar = new javax.swing.JButton();
    v1 = new javax.swing.JTextField();
    v2 = new javax.swing.JTextField();
    v3 = new javax.swing.JTextField();
    v4 = new javax.swing.JTextField();
    v5 = new javax.swing.JTextField();
    v6 = new javax.swing.JTextField();
    v7 = new javax.swing.JTextField();
    v8 = new javax.swing.JTextField();
    v9 = new javax.swing.JTextField();
    v10 = new javax.swing.JTextField();
    v11 = new javax.swing.JTextField();
    v12 = new javax.swing.JTextField();
    jLabel3 = new javax.swing.JLabel();
    jLabel4 = new javax.swing.JLabel();
    jLabel5 = new javax.swing.JLabel();
    jLabel1 = new javax.swing.JLabel();
    jLabel6 = new javax.swing.JLabel();
    jLabel7 = new javax.swing.JLabel();
    INFO = new javax.swing.JButton();
```

```
        VentanaEmergente.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                VentanaEmergenteActionPerformed(evt);
            }
        });

        javax.swing.GroupLayout selectLayout = new
javax.swing.GroupLayout(select.getContentPane());
        select.getContentPane().setLayout(selectLayout);
        selectLayout.setHorizontalGroup(
            selectLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGap(0, 400, Short.MAX_VALUE)
            .addGroup(selectLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                .addGroup(selectLayout.createSequentialGroup()
                    .addGap(0, 0, Short.MAX_VALUE)
                    .addComponent(VentanaEmergente, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addGap(0, 0, Short.MAX_VALUE)))
        );
        selectLayout.setVerticalGroup(
            selectLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGap(0, 300, Short.MAX_VALUE)
            .addGroup(selectLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                .addGroup(selectLayout.createSequentialGroup()
                    .addGap(0, 0, Short.MAX_VALUE)
                    .addComponent(VentanaEmergente, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addGap(0, 0, Short.MAX_VALUE)))
        );

        setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
        setPreferredSize(new java.awt.Dimension(500, 300));
        getContentPane().setLayout(new org.netbeans.lib.awtextra.AbsoluteLayout());
```

```
        normalizar.setText("NORMALIZAR");
        normalizar.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                normalizarActionPerformed(evt);
            }
        });
        getContentPane().add(normalizar, new org.netbeans.lib.awtextra.AbsoluteConstraints(170,
190, 140, -1));

        seleccionar.setText("SELECCIONAR");
        seleccionar.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                seleccionarActionPerformed(evt);
            }
        });
        getContentPane().add(seleccionar, new org.netbeans.lib.awtextra.AbsoluteConstraints(20,
190, 140, -1));

        v1.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                v1ActionPerformed(evt);
            }
        });
        getContentPane().add(v1, new org.netbeans.lib.awtextra.AbsoluteConstraints(90, 90, 70, -1));

        v2.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                v2ActionPerformed(evt);
            }
        });
        getContentPane().add(v2, new org.netbeans.lib.awtextra.AbsoluteConstraints(170, 90, 60, -
1));
        getContentPane().add(v3, new org.netbeans.lib.awtextra.AbsoluteConstraints(280, 90, 60, -
1));
```

```
        v4.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                v4ActionPerformed(evt);
            }
        });
        getContentPane().add(v4, new org.netbeans.lib.awtextra.AbsoluteConstraints(350, 90, 60, -
1));
        getContentPane().add(v5, new org.netbeans.lib.awtextra.AbsoluteConstraints(90, 120, 70, -
1));
        getContentPane().add(v6, new org.netbeans.lib.awtextra.AbsoluteConstraints(170, 120, 60, -
1));
        getContentPane().add(v7, new org.netbeans.lib.awtextra.AbsoluteConstraints(280, 120, 60, -
1));
        getContentPane().add(v8, new org.netbeans.lib.awtextra.AbsoluteConstraints(350, 120, 60, -
1));

        v9.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                v9ActionPerformed(evt);
            }
        });
        getContentPane().add(v9, new org.netbeans.lib.awtextra.AbsoluteConstraints(90, 150, 70, -
1));
        getContentPane().add(v10, new org.netbeans.lib.awtextra.AbsoluteConstraints(170, 150, 60, -
1));
        getContentPane().add(v11, new org.netbeans.lib.awtextra.AbsoluteConstraints(280, 150, 60, -
1));
        getContentPane().add(v12, new org.netbeans.lib.awtextra.AbsoluteConstraints(350, 150, 60, -
1));

        jLabel3.setText("→");
        getContentPane().add(jLabel3, new org.netbeans.lib.awtextra.AbsoluteConstraints(250, 150,
20, 20));

        jLabel4.setText("→");
```

```
        getContentPane().add(jLabel4, new org.netbeans.lib.awtextra.AbsoluteConstraints(250, 120, -
1, -1));

        jLabel5.setText("→");
        getContentPane().add(jLabel5, new org.netbeans.lib.awtextra.AbsoluteConstraints(250, 90, -
1, -1));

        jLabel1.setText("Dependencias Funcionals Para Bases de Datos a Aartir de Una Tabla con
Sitaxis Oracle");
        getContentPane().add(jLabel1, new org.netbeans.lib.awtextra.AbsoluteConstraints(20, 20, -1,
30));

        jLabel6.setText("Introducir las dependencias funcionales");
        getContentPane().add(jLabel6, new org.netbeans.lib.awtextra.AbsoluteConstraints(30, 60, -1,
-1));

        jLabel7.setText("Prototipo de Herramienta Normalizadora Mediante el Algoritmo de Bernstein
Basado en");
        getContentPane().add(jLabel7, new org.netbeans.lib.awtextra.AbsoluteConstraints(20, 0, -1,
30));

        INFO.setText("INFORMACIÓN");
        INFO.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                INFOActionPerformed(evt);
            }
        });
        getContentPane().add(INFO, new org.netbeans.lib.awtextra.AbsoluteConstraints(320, 190,
140, -1));

        pack();
    }// </editor-fold>

    private void normalizarActionPerformed(java.awt.event.ActionEvent evt) {
```

```
        String valor1 =(v1.getText());
        String valor2 =(v2.getText());
        String valor3 =(v3.getText());
        String valor4 =(v4.getText());
        String valor5 =(v5.getText());
        String valor6 =(v6.getText());
        String valor7 =(v7.getText());
        String valor8 =(v8.getText());
        String valor9 =(v9.getText());
        String valor10 =(v10.getText());
        String valor11 =(v11.getText());
        String valor12 =(v12.getText());

       String TipoAtributo;
       String NombreTabla;
       NombreTabla = "Alumnos";
       CoberturaMinima.main(valor1, valor2, valor3, valor4, valor5, valor6, valor7, valor8, valor9,
valor10, valor11, valor12);
      // CoberturaMinima.DFTransitivas(valor1, valor2, valor3, valor4, valor5, valor6, valor7, valor8,
valor9, valor10, valor11, valor12);
       //NombreTabla = TableName.getText();




    }


    private void seleccionarActionPerformed(java.awt.event.ActionEvent evt) {
       select.setSize(new Dimension(550, 350));
       select.setResizable(false);
       select.setVisible(true);
    }
```

```
    private void VentanaEmergenteActionPerformed(java.awt.event.ActionEvent evt) {
        JFileChooser SelectedFile = (JFileChooser)evt.getSource();
        String flag = evt.getActionCommand();
        if (flag.equals(JFileChooser.APPROVE_SELECTION)){
        File CurrentPath  = SelectedFile.getSelectedFile();
        String Ruta = CurrentPath.toString();
            Parser l2 = new Parser();
            System.out.println(l2.LeerSQL(Ruta));
            //System.out.println("Esta es la ruta del archivo: "+Ruta);

        //JOptionPane.showMessageDialog(this, "Ruta: "+ CurrentPath.getAbsolutePath()+"\n Archivo
"+ CurrentPath.getName());
        }else if(flag.equals(JFileChooser.CANCEL_SELECTION)){
         JOptionPane.showMessageDialog(this, "Seleccion un Archivo... ");
        }
    }



    private void v1ActionPerformed(java.awt.event.ActionEvent evt) {




    }

    private void v2ActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
    }

    private void v4ActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
    }
```

```
    private void v9ActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
    }

    private void INFOActionPerformed(java.awt.event.ActionEvent evt) {

                JOptionPane.showMessageDialog(null," Prototipo de Herramienta Normalizadora desarrollado por Irving Castillo 2017 \n Versión 2.0");

    }

    /**
     * @param args the command line arguments
     */
    public static void main(String args[]) {
        /* Set the Nimbus look and feel */
        //<editor-fold defaultstate="collapsed" desc=" Look and feel setting code (optional) ">
        /* If Nimbus (introduced in Java SE 6) is not available, stay with the default look and feel.
         * For details see http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html
         */
        try {
            for (javax.swing.UIManager.LookAndFeelInfo info : javax.swing.UIManager.getInstalledLookAndFeels()) {
                if ("Nimbus".equals(info.getName())) {
                    javax.swing.UIManager.setLookAndFeel(info.getClassName());
                    break;
                }
            }
        } catch (ClassNotFoundException ex) {
            java.util.logging.Logger.getLogger(FormTool.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
        } catch (InstantiationException ex) {
```

```
java.util.logging.Logger.getLogger(FormTool.class.getName()).log(java.util.logging.Level.SEVERE,
null, ex);
        } catch (IllegalAccessException ex) {

java.util.logging.Logger.getLogger(FormTool.class.getName()).log(java.util.logging.Level.SEVERE,
null, ex);
        } catch (javax.swing.UnsupportedLookAndFeelException ex) {

java.util.logging.Logger.getLogger(FormTool.class.getName()).log(java.util.logging.Level.SEVERE,
null, ex);
        }
        //</editor-fold>
        //</editor-fold>

        /* Create and display the form */
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new FormTool( "A1", "A2", "A3", "A4", "A5").setVisible(true);
            }
        });
    }

    // Variables declaration - do not modify
    private javax.swing.JButton INFO;
    private javax.swing.JFileChooser VentanaEmergente;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JLabel jLabel3;
    private javax.swing.JLabel jLabel4;
    private javax.swing.JLabel jLabel5;
    private javax.swing.JLabel jLabel6;
    private javax.swing.JLabel jLabel7;
    private javax.swing.JButton normalizar;
    private javax.swing.JButton seleccionar;
    private javax.swing.JDialog select;
```

```
    private javax.swing.JTextField v1;
    private javax.swing.JTextField v10;
    private javax.swing.JTextField v11;
    private javax.swing.JTextField v12;
    private javax.swing.JTextField v2;
    private javax.swing.JTextField v3;
    private javax.swing.JTextField v4;
    private javax.swing.JTextField v5;
    private javax.swing.JTextField v6;
    private javax.swing.JTextField v7;
    private javax.swing.JTextField v8;
    private javax.swing.JTextField v9;
    // End of variables declaration
}
```

Anexo C Paquete parser clase parser

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package Parser;

import com.sun.org.apache.xalan.internal.xsltc.compiler.Template;
import java.io.*;
import java.util.StringTokenizer;
import java.util.regex.Matcher;
import java.util.regex.Pattern;

/**
 *
 * @author Irving Castillo
 */
public class Parser {
 //void

    public String texto="";
    public String x="";
    public String bfLector;
    public int AlertFlag1=1;
    public int Counter = 5;
    public int LeerSQL(String path){


     String Direccion = path;
        try{
            //BufferedReader maneja cadenas de texto
            //FileReader lee el archivo
```

```
BufferedReader bf = new BufferedReader(new FileReader(Direccion));
//Un ciclo para ver las lineas de texto deteniendose hasta que la ultima no tiene nada solo NULO
while((bfLector = bf.readLine())!=null){
  // String BToken1=""; //Indica una Bandera Token para la palabra reservada CREATE
  // String BToken2 = ""; //Indica una Bandera Token para la palabra reservada TABLE
   StringTokenizer token  = new StringTokenizer(bfLector," ");//Token usado para evaluar palabras separadas por espacio
   StringTokenizer token2  = new StringTokenizer(bfLector,",");//Token usado para evaluar los atributos despues de saber que se inicio un CREATE TABLE
   Pattern patron = Pattern.compile(".*CREATE TABLE.*", Pattern.CASE_INSENSITIVE);
     Matcher mat = patron.matcher(bfLector); // Iteracion con "CREATE TABLE Alumnos"
          if(mat.matches()|| AlertFlag1!=1){
                //System.out.println("Contiene un CREATE esta linea Acontinuación la linea: "+bfLector);
              if(AlertFlag1==3 || Counter == 5){
                 while(token.hasMoreTokens()){
                 String PsWord1="TABLE";
                 String Flag1= token.nextToken();
                 if(Flag1.equalsIgnoreCase(PsWord1)){
                    String  flag2= token.nextToken();
                       if(flag2.endsWith("(")){
                       flag2 = flag2.substring(0, flag2.length() - 1);
                       System.out.println("La tabla se llama: "+flag2);
                     AlertFlag1=0;
                       }
                 }
                 //System.out.println("TOKEN de la linea: "+Flag1);
                Counter = Counter+5;                    }
               }else{
```

```
                        System.out.println("Entro al else indica que está listo para sacar los
atributos");
                        while (token2.hasMoreTokens()) {
                        String Tflag1=token2.nextToken();
                        System.out.println("TOKEN ATRIBUTO "+Tflag1);
                    }
                  }
                }else{
               AlertFlag1=0;
                        //System.out.println("No contiene un CREATE esta linea es un else:
"+bfLector);
                }
      }
         //System.out.println("Variable saliendo de whiles: "+bfLector);
         texto = x;
         System.out.println("es texto: "+texto);
      }catch(Exception ex){System.err.println("No se encontro el Archivo");}
   return 0;
   }
}
```

Anexo D Método para crear nuevo *Schema SQL*

```
/*ESTA CLASE DIO INICIO A LA CLASE FINAL PARA CREAR EL SCHEMA
NORMALIZADO  SU HUBICACION FINAL SE ENCUENTRA EN LA CLASE
COBERTURAMININA*/
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package SQLPicker;

import java.io.BufferedWriter;
import java.io.*;
import java.io.File;
import java.io.FileWriter;
import java.io.IOException;
import java.text.SimpleDateFormat;
import java.util.Calendar;



/**
 *
 * @author Irving
 */
public class SQLFile {

   public static String NombreTabla;;
   public static String Atributo;
   public static String TipoAtributo;

    public static void main(String[] args) {
```

```
    NombreTabla = "Alumnos";
    Atributo = "Nombre";
    TipoAtributo = "VARCHAR";
        try {
            File file = new File("c:\\users\\irving\\desktop\\Normalizada.sql");

            if (file.createNewFile()){


              System.out.println("El archivo SQL ha sido creado!!");
            }else{

            FileWriter w = new FileWriter(file);
            BufferedWriter bw = new BufferedWriter(w);
            PrintWriter wr = new PrintWriter(bw);
            wr.write("-- Creación de Codigo Automatico \n \n");
            wr.append("CREATE TABLE Alumnos ( \n");
            wr.append( Atributo +" "+TipoAtributo);
            wr.append( "); \n");
            wr.close();
            bw.close();
              System.out.println("El archivo SQL ha sido Normalizado!!");

            }

        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

Anexo E Cobertura miníma y algoritmo de Bernstein

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package Parser;

import java.io.BufferedWriter;
import java.io.File;
import java.io.FileWriter;
import java.io.IOException;
import java.io.PrintWriter;
import javax.swing.JOptionPane;

/**
 *
 * @author Irving
 */
public class CoberturaMinima {
    /*    /************************* PRIMERA DEPENDENCIA FUNCIONAL
******************
   public static String vn1="";
   public static String  vn2="A";
   public static String  v33="B";
   public static String  v44="C";
 */
   /************************* PRIMERA DEPENDENCIA FUNCIONAL ******************
   public static String  v55="";
   public static String  v66="B";
   public static String  v77="C";
   public static String  v88="B";
 */
   /************************* TERCERA DEPENDENCIA FUNCIONAL ******************/
```

```
 /*  public static String  v99="";
   public static String  v100="A";
   public static String  v111="D";
   public static String  v112="E";
*/


  public static void main(String v1, String v2, String v3, String v4, String v5, String v6, String v7,
String v8, String v9, String v10, String v11, String v12)
 {
   /************************** Valores iniciales comparables al final para determinar si se
busca una dependencia funcional transitiva******************/


     System.out.println(v1);
     System.out.println(v2);
     System.out.println(v3);
     System.out.println(v4);
     System.out.println(v5);
     System.out.println(v6);
     System.out.println(v7);
     System.out.println(v8);
     System.out.println(v9);
     System.out.println(v10);
     System.out.println(v11);
     System.out.println(v12);

   final String Vn1=v1;
   final String  Vn2=v2;
   final String  V33=v3;
   final String  V44=v4;
   final String  V55=v5;
   final String  V66=v6;
   final String  V77=v7;
```

```
   final String  V88=v8;
   final String  V99=v9;
   final String  V100=v10;
   final String  V111=v11;
   final String  V112=v12;



DFTrivial(v1,v2,v3,v4,v5,v6,v7,v8,v9,v10,v11,v12,Vn1,Vn2,V33,V44,V55,V66,V77,V88,V99,V100,V1
11,V112);



 }

   public static void DFTrivial(String v1, String v2, String v3, String v4, String v5, String v6, String v7,
String v8, String v9, String v10, String v11, String v12, String Vn1, String Vn2, String V33, String V44,
String V55, String V66, String V77, String V88, String V99, String V100, String V111, String V112)
 {

    System.out.print("********ANTES DE APLICAR DF TRVIALES* \n");

    System.out.print("********PRIMERA DEPENDENCIA FUNCIONAL* \n");
        System.out.print("caractere 1: "+v1+"\n");
    System.out.print("caractere 2: "+v2+"\n");
    System.out.print("caractere 3: "+v3+"\n");
    System.out.print("caractere 4: "+v4+"\n");

    System.out.print("********SEGUNDA DEPENDENCIA FUNCIONAL* \n");
    System.out.print("caractere 5: "+v5+"\n");
    System.out.print("caractere 6: "+v6+"\n");
    System.out.print("caractere 7: "+v7+"\n");
    System.out.print("caractere 8: "+v8+"\n");

    System.out.print("********TERCERA DEPENDENCIA FUNCIONAL* \n");
```

```
    System.out.print("caractere 9: "+v9+"\n");
    System.out.print("caractere 10: "+v10+"\n");
    System.out.print("caractere 11: "+v11+"\n");
    System.out.print("caractere 12: "+v12+"\n");
   /*
    for(i=0;i<=11;i++){

      System.out.print("caractere 1: "+v1+"\n");
    }*/

    /************* PRIMERA DEPENDENCIA FUNCIONAL A APICAR ALGORITMO
TRIVIALIDAD************/
    if(v3.equals(v1)||v3.equals(v2)){
        v3="";
    }else if(v4.equals(v1)||v4.equals(v2)){
        v4="";
    }

    /************* SEGUNDA DEPENDENCIA FUNCIONAL A APICAR ALGORITMO
TRIVIALIDAD************/

    if(v7.equals(v5)||v7.equals(v6)){
        v7="";
    }else if(v8.equals(v5)||v8.equals(v6)){
        v8="";
    }

     /************* TERCERA DEPENDENCIA FUNCIONAL A APICAR ALGORITMO
TRIVIALIDAD************/

    if(v11.equals(v9)||v11.equals(v10)){
        v11="";
    }else if(v12.equals(v9)||v12.equals(v10)){
        v12="";
```

```
  }


  System.out.print("********DESPUES DE APLICAR DF TRVIALES* \n");

  System.out.print("********PRIMERA DEPENDENCIA FUNCIONAL* \n");
      System.out.print("caractere 1: "+v1+"\n");
  System.out.print("caractere 2: "+v2+"\n");
  System.out.print("caractere 3: "+v3+"\n");
  System.out.print("caractere 4: "+v4+"\n");

  System.out.print("********SEGUNDA DEPENDENCIA FUNCIONAL* \n");
  System.out.print("caractere 5: "+v5+"\n");
  System.out.print("caractere 6: "+v6+"\n");
  System.out.print("caractere 7: "+v7+"\n");
  System.out.print("caractere 8: "+v8+"\n");

  System.out.print("********TERCERA DEPENDENCIA FUNCIONAL* \n");
  System.out.print("caractere 9: "+v9+"\n");
  System.out.print("caractere 10: "+v10+"\n");
  System.out.print("caractere 11: "+v11+"\n");
  System.out.print("caractere 12: "+v12+"\n");



 DFImplicitas(v1,v2,v3,v4,v5,v6,v7,v8,v9,v10,v11,v12);


}
```

```
   public static void DFImplicitas(String v1, String v2, String v3, String v4, String v5, String v6, String
v7, String v8, String v9, String v10, String v11, String v12)
 {
    //Primera Dependencia Con Otras
    if(((v1.equals(v5) && !"".equals(v1)) || (v1.equals(v6) && !"".equals(v1)))&&((v2.equals(v7) &&
!"".equals(v2))||(v2.equals(v8) && !"".equals(v2)))){
    v2="";
    }else if(((v2.equals(v5) && !"".equals(v2)) || (v2.equals(v6) &&
!"".equals(v2)))&&((v1.equals(v7) && !"".equals(v1))||(v1.equals(v8) && !"".equals(v1)))){
    v1="";
    }else if(((v2.equals(v5) && !"".equals(v2)) || (v2.equals(v6) &&
!"".equals(v2)))&&((v3.equals(v7) && !"".equals(v3))||(v3.equals(v8) && !"".equals(v3)))){
    v3="";
    }else if(((v2.equals(v5) && !"".equals(v2)) || (v2.equals(v6) &&
!"".equals(v2)))&&((v4.equals(v7) && !"".equals(v4))||(v4.equals(v8) && !"".equals(v4)))){
    v4="";
    }else if(((v1.equals(v5) && !"".equals(v1)) || (v1.equals(v6) &&
!"".equals(v1)))&&((v3.equals(v7) && !"".equals(v3))||(v3.equals(v8) && !"".equals(v3)))){
      v3="";
    }else if(((v1.equals(v5) && !"".equals(v1)) || (v1.equals(v6) &&
!"".equals(v1)))&&((v4.equals(v7) && !"".equals(v4))||(v4.equals(v8) && !"".equals(v4)))){
      v4="";}
    /************************************************************
*************************************************************************************
***/
     else if(((v1.equals(v7) && !"".equals(v1)) || (v1.equals(v8) && !"".equals(v1)) || (v2.equals(v7)
&& !"".equals(v2)) || (v2.equals(v8) && !"".equals(v2)))&&((v7.equals(v3) &&
!"".equals(v7))||(v7.equals(v4) && !"".equals(v7)))){
      v7="";
    }else if(((v1.equals(v7) && !"".equals(v1)) || (v1.equals(v8) && !"".equals(v1)) || (v2.equals(v7)
&& !"".equals(v2)) || (v2.equals(v8) && !"".equals(v2)))&&((v8.equals(v3) &&
!"".equals(v8))||(v8.equals(v4) && !"".equals(v8)))){
      v8="";
    }


    if(((v1.equals(v9) && !"".equals(v1)) || (v1.equals(v10) && !"".equals(v1)))&&((v2.equals(v11)
&& !"".equals(v2))||(v2.equals(v12) && !"".equals(v2)))){
```

```
    v2="";
    }else if(((v2.equals(v9) && !"".equals(v2)) || (v2.equals(v10) &&
!"".equals(v2)))&&((v1.equals(v11) && !"".equals(v1))||(v1.equals(v12) && !"".equals(v1)))){
    v1="";
    }else if(((v2.equals(v9) && !"".equals(v2)) || (v2.equals(v10) &&
!"".equals(v2)))&&((v3.equals(v11) && !"".equals(v3))||(v3.equals(v12) && !"".equals(v3)))){
    v3="";
    }else if(((v2.equals(v9) && !"".equals(v2)) || (v2.equals(v10) &&
!"".equals(v2)))&&((v4.equals(v11) && !"".equals(v4))||(v4.equals(v12) && !"".equals(v4)))){
    v4="";
    }else if(((v1.equals(v9) && !"".equals(v1)) || (v1.equals(v10) &&
!"".equals(v1)))&&((v3.equals(v11) && !"".equals(v3))||(v3.equals(v12) && !"".equals(v3)))){
      v3="";
    }else if(((v1.equals(v9) && !"".equals(v1)) || (v1.equals(v10) &&
!"".equals(v1)))&&((v4.equals(v11) && !"".equals(v4))||(v4.equals(v12) && !"".equals(v4)))){
      v4="";}
    /*************************************************************
*****************************************************************************
***/
     else if(((v1.equals(v11) && !"".equals(v1)) || (v1.equals(v12) && !"".equals(v1)) ||
(v2.equals(v11) && !"".equals(v2)) || (v2.equals(v12) && !"".equals(v2)))&&((v11.equals(v3) &&
!"".equals(v11))||(v11.equals(v4) && !"".equals(v11)))){
      v11="";
    }else if(((v1.equals(v11) && !"".equals(v1)) || (v1.equals(v12) && !"".equals(v1)) ||
(v2.equals(v11) && !"".equals(v2)) || (v2.equals(v12) && !"".equals(v2)))&&((v12.equals(v3) &&
!"".equals(v12))||(v12.equals(v4) && !"".equals(v12)))){
      v12="";
    }



    //Segunda Dependencia Con Otras
    if(( (v5.equals(v1) && !v5.equals("")) || (v5.equals(v2) && !v5.equals("")))&&((v6.equals(v3)
&& !v6.equals(""))||(v6.equals(v4) && !v6.equals(""))){
    v6="";
    }else if(((v6.equals(v1) && !v6.equals("")) || (v6.equals(v2) &&
!v6.equals("")))&&((v5.equals(v3) && !"".equals(v5))||(v5.equals(v4) && !"".equals(v5)))){
    v5="";
```

```
    }else if(((v6.equals(v1) && !"".equals(v6)) || (v6.equals(v2) &&
!"".equals(v6)))&&((v7.equals(v3) && !"".equals(v7))||(v7.equals(v4) && !"".equals(v7)))){

    v7="";

    }else if(((v6.equals(v1) && !"".equals(v6)) || (v6.equals(v2) &&
!"".equals(v6)))&&((v8.equals(v3) && !"".equals(v8))||(v8.equals(v4) && !"".equals(v8)))){

    v8="";

    }else if(((v5.equals(v1) && !"".equals(v5)) || (v5.equals(v2) &&
!"".equals(v5)))&&((v7.equals(v3) && !"".equals(v7))||(v7.equals(v4) && !"".equals(v7)))){

     v7="";

    }else if(((v5.equals(v1) && !"".equals(v5)) || (v5.equals(v2) &&
!"".equals(v5)))&&((v8.equals(v3) && !"".equals(v8))||(v8.equals(v4) && !"".equals(v8)))){

     v8="";}

    /*************************************************************
*************************************************************************************
***/

     else if(((v5.equals(v3) && !"".equals(v5)) || (v5.equals(v4) && !"".equals(v5)) || (v6.equals(v3)
&& !"".equals(v6)) || (v6.equals(v4) && !"".equals(v6)))&&((v3.equals(v7) &&
!"".equals(v7))||(v3.equals(v8) && !"".equals(v8)))){

     v3="";

    }else if(((v5.equals(v3) && !"".equals(v5)) || (v5.equals(v4) && !"".equals(v5)) || (v6.equals(v3)
&& !"".equals(v6)) || (v6.equals(v4) && !"".equals(v6)))&&((v4.equals(v7) &&
!"".equals(v7))||(v4.equals(v8) && !"".equals(v8)))){

     v4="";}



    if(((v5.equals(v9) && !v5.equals("")) || (v5.equals(v10) && !v5.equals("")))&&( (v6.equals(v11)
&& !v6.equals(""))||(v6.equals(v12) && !v6.equals("")))){

    v6="";

    }else if(((v6.equals(v9) && !"".equals(v6)) || (v6.equals(v10) &&
!"".equals(v6)))&&((v5.equals(v11) && !"".equals(v5))|| (v5.equals(v12) && !"".equals(v6)))){

    v5="";

    }else if(((v6.equals(v9) && !"".equals(v6)) || (v6.equals(v10) &&
!"".equals(v6)))&&((v7.equals(v11) && !"".equals(v7))||(v7.equals(v12) && !"".equals(v7)))){

    v7="";

    }else if(((v6.equals(v9) && !"".equals(v6)) || (v6.equals(v10) &&
!"".equals(v6)))&&((v8.equals(v11) && !"".equals(v8))||(v8.equals(v12) && !"".equals(v8)))){

    v8="";
```

```
    }else if(((v5.equals(v9) && !"".equals(v5)) || (v5.equals(v10) &&
!"".equals(v5)))&&((v7.equals(v11) && !"".equals(v7))||(v7.equals(v12) && !"".equals(v7)))){
      v7="";
    }else if(((v5.equals(v9) && !"".equals(v5)) || (v5.equals(v10) &&
!"".equals(v5)))&&((v8.equals(v11) && !"".equals(v8))||(v8.equals(v12) && !"".equals(v8)))){
      v8="";}
      /*************************************************************
*****************************************************************************
***/
     else if(((v5.equals(v11) && !"".equals(v5)) || (v5.equals(v12) && !"".equals(v5)) ||
(v6.equals(v11) && !"".equals(v6)) || (v6.equals(v12) && !"".equals(v6)))&&((v3.equals(v11) &&
!"".equals(v3))||(v3.equals(v12) && !"".equals(v3)))){
      v3="";
    }else if(((v5.equals(v11) && !"".equals(v5)) || (v5.equals(v12) && !"".equals(v5)) ||
(v6.equals(v11) && !"".equals(v6)) || (v6.equals(v12) && !"".equals(v6)))&&((v4.equals(v11) &&
!"".equals(v4))||(v4.equals(v12) && !"".equals(v4)))){
      v4="";}



     //Tercera Dependencia Con Otras
    if(((v9.equals(v1) && !"".equals(v9)) || (v9.equals(v2) && !"".equals(v9)))&&((v10.equals(v3)
&& !"".equals(v10))||(v10.equals(v4) && !"".equals(v10)))){
    v10="";
    }else if(((v10.equals(v1) && !"".equals(v10)) || (v10.equals(v2) &&
!"".equals(v10)))&&((v9.equals(v3) && !"".equals(v9))||(v9.equals(v4) && !"".equals(v9)))){
    v9="";
    }else if(((v10.equals(v1) && !"".equals(v10)) || (v10.equals(v2) &&
!"".equals(v10)))&&((v11.equals(v3) && !"".equals(v11))||(v11.equals(v4) && !"".equals(v11)))){
    v11="";
    }else if(((v10.equals(v1) && !"".equals(v10)) || (v10.equals(v2) &&
!"".equals(v10)))&&((v12.equals(v3) && !"".equals(v12))||(v12.equals(v4) && !"".equals(v12)))){
    v12="";
    }else if(((v9.equals(v1) && !"".equals(v9)) || (v9.equals(v2) &&
!"".equals(v9)))&&((v11.equals(v3) && !"".equals(v11))||(v11.equals(v4) && !"".equals(v11)))){
      v11="";
```

```
    }else if(((v9.equals(v1) && !"".equals(v9)) || (v9.equals(v2) &&
!"".equals(v9)))&&((v12.equals(v3) && !"".equals(v12))||(v12.equals(v4) && !"".equals(v12)))){
      v12="";}
    /*************************************************************
*************************************************************************************
***/
     else if(((v9.equals(v3) && !"".equals(v9)) || (v9.equals(v4) && !"".equals(v9)) ||
(v10.equals(v3) && !"".equals(v10)) || (v10.equals(v4) && !"".equals(v10)))&&((v3.equals(v11) &&
!"".equals(v3))||(v3.equals(v12) && !"".equals(v3)))){
      v3="";
    }else if(((v9.equals(v3) && !"".equals(v9)) || (v9.equals(v4) && !"".equals(v9)) ||
(v10.equals(v3) && !"".equals(v10)) || (v10.equals(v4) && !"".equals(v10)))&&((v4.equals(v11) &&
!"".equals(v4))||(v4.equals(v12) && !"".equals(v4)))){
      v4="";}



    if(((v9.equals(v5) && !"".equals(v9)) || (v9.equals(v6) && !"".equals(v9)))&&((v10.equals(v7)
&& !"".equals(v10))||(v10.equals(v8) && !"".equals(v10)))){
    v10="";
    }else if(((v10.equals(v5) && !"".equals(v10)) || (v10.equals(v6) &&
!"".equals(v10)))&&((v9.equals(v7) && !"".equals(v9))||(v9.equals(v8) && !"".equals(v9)))){
    v9="";
    }else if(((v10.equals(v5) && !"".equals(v10)) || (v10.equals(v6) &&
!"".equals(v10)))&&((v11.equals(v7) && !"".equals(v11))||(v11.equals(v8) && !"".equals(v11)))){
    v11="";
    }else if(((v10.equals(v5) && !"".equals(v10)) || (v10.equals(v6) &&
!"".equals(v10)))&&((v12.equals(v7) && !"".equals(v12))||(v12.equals(v8) && !"".equals(v12)))){
    v12="";
    }else if(((v9.equals(v5) && !"".equals(v9)) || (v9.equals(v6) &&
!"".equals(v9)))&&((v11.equals(v7) && !"".equals(v11))||(v11.equals(v8) && !"".equals(v11)))){
      v11="";
    }else if(((v9.equals(v5) && !"".equals(v9)) || (v9.equals(v6) &&
!"".equals(v9)))&&((v12.equals(v7) && !"".equals(v12))||(v12.equals(v8) && !"".equals(v12)))){
      v12="";}
```

```
      /*************************************************************
*************************************************************************************
***/
     else if(((v9.equals(v7) && !"".equals(v9)) || (v9.equals(v8) && !"".equals(v9)) ||
(v10.equals(v7) && !"".equals(v10)) || (v10.equals(v8) && !"".equals(v10)))&&((v7.equals(v11) &&
!"".equals(v7))||(v7.equals(v12) && !"".equals(v7)))){
      v7="";
    }else if(((v9.equals(v7) && !"".equals(v9)) || (v9.equals(v8) && !"".equals(v9)) ||
(v10.equals(v7) && !"".equals(v10)) || (v10.equals(v8) && !"".equals(v10)))&&((v8.equals(v11) &&
!"".equals(v8))||(v8.equals(v12) && !"".equals(v8)))){
      v8="";}



   System.out.print("\n\n********DESPUES DE APLICAR DF IMPLICITAS* \n\n");

    System.out.print("*********PRIMERA DEPENDENCIA FUNCIONAL* \n");
        System.out.print("caractere 1: "+v1+"\n");
    System.out.print("caractere 2: "+v2+"\n");
    System.out.print("caractere 3: "+v3+"\n");
    System.out.print("caractere 4: "+v4+"\n");

    System.out.print("*********SEGUNDA DEPENDENCIA FUNCIONAL* \n");
    System.out.print("caractere 5: "+v5+"\n");
    System.out.print("caractere 6: "+v6+"\n");
    System.out.print("caractere 7: "+v7+"\n");
    System.out.print("caractere 8: "+v8+"\n");

    System.out.print("*********TERCERA DEPENDENCIA FUNCIONAL* \n");
    System.out.print("caractere 9: "+v9+"\n");
    System.out.print("caractere 10: "+v10+"\n");
    System.out.print("caractere 11: "+v11+"\n");
    System.out.print("caractere 12: "+v12+"\n");
```

```
   DFTransitivas(v1,v2,v3,v4,v5,v6,v7,v8,v9,v10,v11,v12);
  }

   public static void DFTransitivas(String v1, String v2, String v3, String v4, String v5, String v6,
String v7, String v8, String v9, String v10, String v11, String v12)
  {


       System.out.print("\n\n********ANTES DE APLICAR DF TRANSITIVIDAD* \n\n");
     System.out.print("********PRIMERA DEPENDENCIA FUNCIONAL* \n");
     System.out.print("caractere 1: "+v1+"\n");
     System.out.print("caractere 2: "+v2+"\n");
     System.out.print("caractere 3: "+v3+"\n");
     System.out.print("caractere 4: "+v4+"\n");

     System.out.print("********SEGUNDA DEPENDENCIA FUNCIONAL* \n");
     System.out.print("caractere 5: "+v5+"\n");
     System.out.print("caractere 6: "+v6+"\n");
     System.out.print("caractere 7: "+v7+"\n");
     System.out.print("caractere 8: "+v8+"\n");

     System.out.print("********TERCERA DEPENDENCIA FUNCIONAL* \n");
     System.out.print("caractere 9: "+v9+"\n");
     System.out.print("caractere 10: "+v10+"\n");
     System.out.print("caractere 11: "+v11+"\n");
     System.out.print("caractere 12: "+v12+"\n");

       // if(v1.equals(Vn1) && v2.equals(CoberturaMinima.Vn2) &&
v3.equals(CoberturaMinima.v33) && v4.equals(CoberturaMinima.v44) &&
v5.equals(CoberturaMinima.v55) && v6.equals(CoberturaMinima.v66) &&
v7.equals(CoberturaMinima.v77) && v8.equals(CoberturaMinima.v88) && v9.equals(
CoberturaMinima.v99)&& v10.equals(CoberturaMinima.v100) &&
v11.equals(CoberturaMinima.v111) && v12.equals(CoberturaMinima.v112)){
```

```
  System.out.print("\n\n********DESPUES DE APLICAR DF TRANSITIVIDAD* \n\n");

System.out.print("*********PRIMERA DEPENDENCIA FUNCIONAL* \n");
    System.out.print("caractere 1: "+v1+"\n");
System.out.print("caractere 2: "+v2+"\n");
System.out.print("caractere 3: "+v3+"\n");
System.out.print("caractere 4: "+v4+"\n");
//if(vE1 != ""){
//System.out.print("caractere Extra: "+vE1+"\n");}

System.out.print("*********SEGUNDA DEPENDENCIA FUNCIONAL* \n");
System.out.print("caractere 5: "+v5+"\n");
System.out.print("caractere 6: "+v6+"\n");
System.out.print("caractere 7: "+v7+"\n");
System.out.print("caractere 8: "+v8+"\n");

//if(vE2 != ""){
//System.out.print("caractere Extra : "+vE2+"\n");}

System.out.print("*********TERCERA DEPENDENCIA FUNCIONAL* \n");
System.out.print("caractere 9: "+v9+"\n");
System.out.print("caractere 10: "+v10+"\n");
System.out.print("caractere 11: "+v11+"\n");
System.out.print("caractere 12: "+v12+"\n");

//if(vE3 != ""){
// System.out.print("caractere Extra : "+vE3+"\n");
```

```
   Bernstein(v1,v2,v3,v4,v5,v6,v7,v8,v9,v10,v11,v12);

   }

   public static void Bernstein(String v1, String v2, String v3, String v4, String v5, String v6, String v7,
String v8, String v9, String v10, String v11, String v12){

   String vE1="";
   String vE2="";
   String vE3="";
   String vE4="";
   String vE5="";
   String vE6="";

    if(v1.equals(v5) && v2.equals(v6)){
     v5="";
     v6="";
     vE1=v7;
     vE2=v8;
     v7="";
     v8="";
    }

    else if(v1.equals(v9) && v2.equals(v10)){
     v9="";
     v10="";
     vE1=v11;
     vE2=v12;
     v11="";
     v12="";
```

```
    }

     else if(v5.equals(v9) && v6.equals(v6)){
      v9="";
      v10="";
      vE3=v11;
      vE4=v12;
      v11="";
      v12="";

    }

       System.out.print("\n\n\n\n********NUEVO SCHEMA DE BASE DE DATOS* \n\n\n\n");

    /*

if(((v1.equals(v5)||v1.equals(v6))&&(v2.equals(v5)||v2.equals(v6)))||((v1.equals(v9)||v1.equals(v
10))&&(v2.equals(v9)||v2.equals(v10)))){
      if((v1.equals(v5)||v1.equals(v6))&&(v2.equals(v5)||v2.equals(v6))){
      System.out.print("Nombre_Tabla( "+v1+" "+v2+" "+v3+" "+v4+" "+v7+" "+v8+" ) \n\n");
      }else{
      System.out.print("Nombre_Tabla( "+v1+" "+v2+" "+v3+" "+v4+" "+v11+" "+v12+" ) \n\n");}
    }else{
    System.out.print("Nombre_Tabla( "+v1+" "+v2+" "+v3+" "+v4+" ) \n\n");
    System.out.print("Nombre_Tabla( "+v5+" "+v6+" "+v7+" "+v8+" ) \n\n");
    }

if(((v5.equals(v9)||v5.equals(v10))&&(v6.equals(v9)||v6.equals(v10)))||((v5.equals(v1)||v5.equal
s(v2))&&(v6.equals(v1)||v6.equals(v2)))){
      if((v5.equals(v9)||v5.equals(v10))&&(v6.equals(v9)||v6.equals(v10))){
      System.out.print("Nombre_Tabla( "+v5+" "+v6+" "+v7+" "+v8+" "+v11+" "+v12+" ) \n\n");
      }else{
```

```
  System.out.print("Nombre_Tabla( "+v5+" "+v6+" "+v7+" "+v8+" "+v3+" "+v4+" ) \n\n");}
 }else{
 System.out.print("Nombre_Tabla( "+v1+" "+v2+" "+v3+" "+v4+" ) \n\n");
 System.out.print("Nombre_Tabla( "+v5+" "+v6+" "+v7+" "+v8+" ) \n\n");
 }

 */
 /*
 String DF[][]={{v1,v2,v3,v4},{v5,v6,v7,v8},{v9,v10,v11,v12}};

 for (int x=0; x < DF.length-2; x++) {
  for (int y=0; y < DF[x].length-2; y++) {
   if(DF[x][y] != "" ){
                System.out.print("Atributo "+DF[x][y]+"  \n\n");
  }
 }
 }

 */

       if(v1.equals("") && v2.equals("") && v3.equals("") && v4.equals("")){
}else{
 System.out.print("********TABLA RECOMENDADA********** \n");
 System.out.print("Nombre_Tabla( "+v1+" "+v2+" "+v3+" "+v4+" "+vE1+" "+vE2+" ) \n\n");
}

       if(v5.equals("") && v6.equals("") && v7.equals("") && v8.equals("")){
}else{
 System.out.print("********TABLA RECOMENDADA********** \n");
 System.out.print("Nombre_Tabla( "+v5+" "+v6+" "+v7+" "+v8+" "+vE3+" "+vE4+" ) \n\n");
}

       if(v9.equals("") && v10.equals("") && v11.equals("") && v12.equals("")){
```

```
}else{
 System.out.print("*********TABLA RECOMENDADA********** \n");
 System.out.print("Nombre_Tabla( "+v9+" "+v10+" "+v11+" "+v12+" ) \n\n");
}



    String Atributo;

        try {

            File file = new File("c:\\users\\irving\\desktop\\Normalizada.sql");

            if (file.createNewFile()){


           FileWriter w = new FileWriter(file);
           BufferedWriter bw = new BufferedWriter(w);
           PrintWriter wr = new PrintWriter(bw);

           wr.write("--    Prototipo de Herramienta Normalizadora Mediante el Algoritmo de
Bernstein Basado en  \n");
           wr.write("--    Dependencias Funcionals Para Bases de Datos a Aartir de Una Tabla con
Sitaxis Oracle \n");
            wr.write("-- Desarrollado por Irving Abad Castillo Fierro 2017 \n \n");
           wr.write("-- Creación de Codigo Automatico \n \n");
           wr.write("-- Se han agregado los create table necesarios y sus respectivos atributos \n ");
           wr.write("-- NOTA TIPO DE DATO ILUSTRATIVO CAMBIAR POR EL NECESARIO \n \n");


           if( !"".equals(v1) || !"".equals(v2) ){
           wr.append("CREATE TABLE Nombre_Tabla_ ( \n");
              if(!"".equals(v1)){
```

```
wr.append( v1 +" VARCHAR(65),\n ");}
   if(!"".equals(v2)){
wr.append( v2 +" VARCHAR(65) ,\n ");}
   if(!"".equals(v3)){
wr.append( v3 +" VARCHAR(65) ,\n ");}
   if(!"".equals(v4)){
wr.append( v4 +" VARCHAR(65) ,\n ");}
   if(!"".equals(vE1)){
wr.append( vE1 +" VARCHAR(65) ,\n ");}
   if(!"".equals(vE2)){
wr.append( vE2 +" VARCHAR(65) ,\n ");}
wr.append( "); \n");
}if(!"".equals(v5) || !"".equals(v6)  ){
wr.append("CREATE TABLE Nombre_Tabla ( \n");
   if(!"".equals(v5)){
wr.append( v5 +" VARCHAR(65),\n ");}
   if(!"".equals(v6)){
wr.append( v6 +" VARCHAR(65) ,\n ");}
   if(!"".equals(v7)){
wr.append( v7 +" VARCHAR(65) ,\n ");}
   if(!"".equals(v8)){
wr.append( v8 +" VARCHAR(65) ,\n ");}
   if(!"".equals(vE3)){
wr.append( vE3 +" VARCHAR(65) ,\n ");}
   if(!"".equals(vE4)){
wr.append( vE4 +" VARCHAR(65) ,\n ");}
wr.append( "); \n");
}if(!"".equals(v9) || !"".equals(v10) )
{
wr.append("CREATE TABLE Nombre_Tabla__ ( \n");
   if(!"".equals(v9)){
wr.append( v9 +" VARCHAR(65),\n ");}
   if(!"".equals(v10)){
```

```
wr.append( v10 +" VARCHAR(65) ,\n ");}
   if(!"".equals(v11)){
wr.append( v11 +" VARCHAR(65) ,\n ");}
   if(!"".equals(v12)){
wr.append( v12 +" VARCHAR(65) ,\n ");}
wr.append( "); \n");
}
wr.close();
bw.close();

JOptionPane.showMessageDialog(null,"El archivo SQL ha sido creado!! ");

  }else{


FileWriter w = new FileWriter(file);
BufferedWriter bw = new BufferedWriter(w);
PrintWriter wr = new PrintWriter(bw);

wr.write("--    Prototipo de Herramienta Normalizadora Mediante el Algoritmo de
Bernstein Basado en  \n");
wr.write("--    Dependencias Funcionals Para Bases de Datos a Aartir de Una Tabla con
Sitaxis Oracle \n");
 wr.write("-- Desarrollado por Irving Abad Castillo Fierro 2017 \n \n");
wr.write("-- Creación de Codigo Automatico \n \n");
wr.write("-- Se han agregado los create table necesarios y sus respectivos atributos \n ");
wr.write("-- NOTA TIPO DE DATO ILUSTRATIVO CAMBIAR POR EL NECESARIO \n \n");


if( !"".equals(v1) || !"".equals(v2) ){
wr.append("CREATE TABLE Nombre_Tabla_ ( \n");
   if(!"".equals(v1)){
wr.append( v1 +" VARCHAR(65),\n ");}
   if(!"".equals(v2)){
```

```
wr.append( v2 +" VARCHAR(65) ,\n ");}
   if(!"".equals(v3)){
wr.append( v3 +" VARCHAR(65) ,\n ");}
   if(!"".equals(v4)){
wr.append( v4 +" VARCHAR(65) ,\n ");}
   if(!"".equals(vE1)){
wr.append( vE1 +" VARCHAR(65) ,\n ");}
   if(!"".equals(vE2)){
wr.append( vE2 +" VARCHAR(65) ,\n ");}
wr.append( "); \n");
}if(!"".equals(v5) || !"".equals(v6)  ){
wr.append("CREATE TABLE Nombre_Tabla ( \n");
   if(!"".equals(v5)){
wr.append( v5 +" VARCHAR(65),\n ");}
   if(!"".equals(v6)){
wr.append( v6 +" VARCHAR(65) ,\n ");}
   if(!"".equals(v7)){
wr.append( v7 +" VARCHAR(65) ,\n ");}
   if(!"".equals(v8)){
wr.append( v8 +" VARCHAR(65) ,\n ");}
   if(!"".equals(vE3)){
wr.append( vE3 +" VARCHAR(65) ,\n ");}
   if(!"".equals(vE4)){
wr.append( vE4 +" VARCHAR(65) ,\n ");}
wr.append( "); \n");
}if(!"".equals(v9) || !"".equals(v10) )
{
wr.append("CREATE TABLE Nombre_Tabla__ ( \n");
   if(!"".equals(v9)){
wr.append( v9 +" VARCHAR(65),\n ");}
   if(!"".equals(v10)){
wr.append( v10 +" VARCHAR(65) ,\n ");}
   if(!"".equals(v11)){
```

```
            wr.append( v11 +" VARCHAR(65) ,\n ");}
               if(!"".equals(v12)){
            wr.append( v12 +" VARCHAR(65) ,\n ");}
            wr.append( "); \n");
            }
            wr.close();
            bw.close();


              }

         } catch (IOException e) {
             e.printStackTrace();
         }



    JOptionPane.showMessageDialog(null," El archivo SQL ha sido Normalizado!!");

  }
}
```

Anexo F *Schema* desnormalizado de prueba

Para la elaboración del capítulo IV ,resultados , se crearon dos *schemas* con los mismos atributos sin embargo uno normalizado y otro desnormalizado los cuales fueron sometidos a los mismos *queries*.

```
CREATE TABLE `ventas` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `id_cliente` int(11) DEFAULT NULL,
  `nombre_cliente` varchar(35) DEFAULT NULL,
  `fecha` datetime DEFAULT NULL,
  `total` int(11) DEFAULT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=1007 DEFAULT CHARSET=latin1;

-- Stored procedure utilizado para poblar el schema. tabla ventas
DELIMITER $$
CREATE DEFINER=`root`@`localhost` PROCEDURE `Llenado_ventas`()
BEGIN
 DECLARE x  INT;
 DECLARE cliente_name varchar(35);
 SET x = 1;
 SET cliente_name = 'Cliente No. '+ 'convert(x,char)';
 WHILE x  <= 500 DO
        INSERT INTO ventas(id,id_cliente,nombre_cliente,fecha,total)
        values('',x,concat('Nombre Cliente ',x),now(),300+x);
 SET  x = x + 1;
 END WHILE;
END$$
DELIMITER ;
--Comando de ejecución para el Stored procedure.
CALL `nonormalizada`.`Llenado_ventas`();
```

Anexo G *Schema* normalizado de prueba

```
CREATE TABLE `clientes` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `nombre` varchar(45) DEFAULT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=502 DEFAULT CHARSET=latin1;

CREATE TABLE `ventas` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `cliente_id` int(11) NOT NULL,
  `fecha` datetime DEFAULT NULL,
  `total` int(11) DEFAULT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=506 DEFAULT CHARSET=latin1;

-- Stored procedure utilizado para poblar el schema. tabla clientes
DELIMITER $$
CREATE DEFINER=`root`@`localhost` PROCEDURE `llenado_clientes`()
BEGIN
DECLARE x  INT;
 DECLARE cliente_name varchar(35);
 SET x = 1;
 WHILE x  <= 501 DO
        INSERT INTO clientes(id,nombre) values('',concat('Nombre Cliente ',x));
 SET  x = x + 1;
 END WHILE;
END$$
DELIMITER ;

--Comando de ejecución para el Stored procedure.
CALL `normalizada`.`llenado_clientes`();
```

-- *Stored procedure* utilizado para poblar el *schema.* tabla ventas

```
DELIMITER $$
CREATE DEFINER=`root`@`localhost` PROCEDURE `llenado_ventas`()
BEGIN
 DECLARE x  INT;
 SET x = 1;
 WHILE x  <= 500 DO
        INSERT INTO ventas(id,cliente_id,fecha,total) values('',x,now(),300+x);
 SET  x = x + 1;
 END WHILE;
END$$
DELIMITER ;
```

--Comando de ejecución para el *Stored procedure.*

```
CALL `normalizada`.`llenado_ventas`();
```

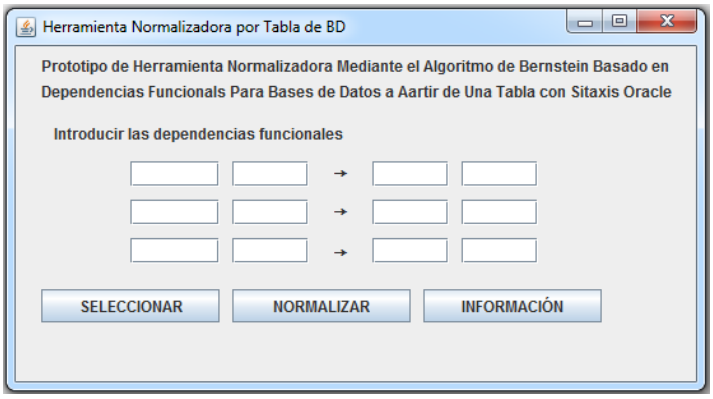
Sentencias update aplicados a schemas para pruebas

```
UPDATE nonormalizada.ventas
SET nombre_cliente = 'JUAN'
WHERE id_cliente=1;

UPDATE normalizada.clientes
SET nombre = 'JUAN'
WHERE id=501;

UPDATE nonormalizada.ventas
SET nombre_cliente = 'JUAN'
WHERE id_cliente=1;
```

Anexo H Manual de usuario

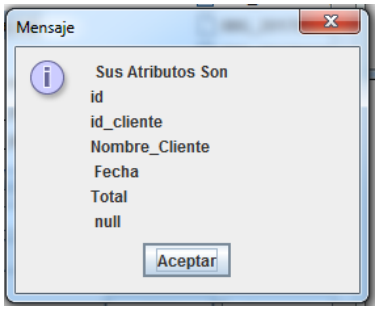


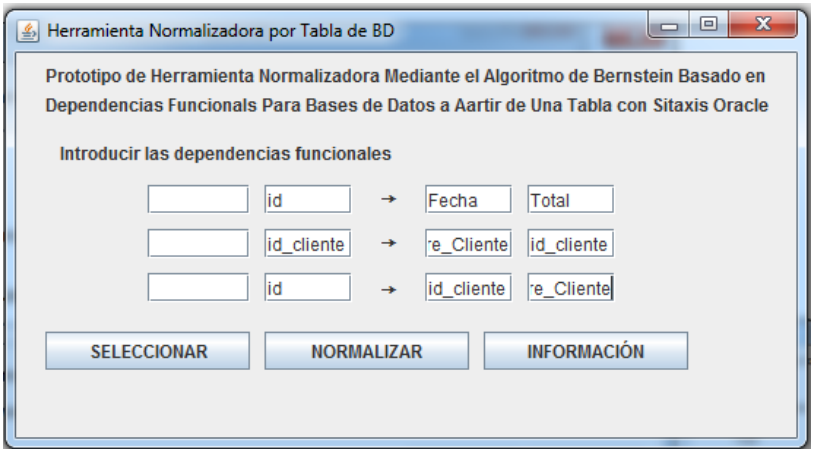
Ventana principal, es donde se selecciona el archivo *SQL* a normalizar, los cuadros de tesxto son campos para agregar las dependencias funcionales.

El botón seleccionar

Es donde se selecciona un archivo *SQL* del disco duro o unidad externa de almacenamiento.

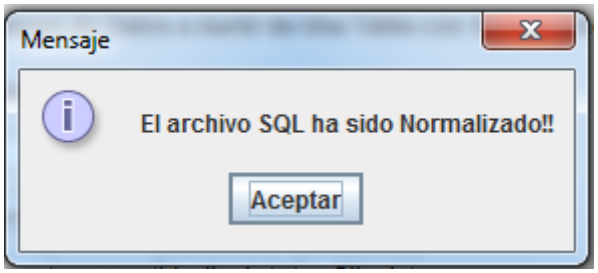
Al seleccionar un archivo aparecerá una ventana indicando los atributos del esquema físico encontrado. La palabra null indica que ya no se han localizado más atributos.





El siguiente paso es introducir las dependencias funcionales a partir de los atributos localizados en el paso anterior.

Finalmente se preciona el botón normalizar el cual mostrará la ventana siguiente.



Creando así un esquema físico como el mostrado a continuación

```
--    Prototipo de Herramienta Normalizadora Mediante el Algoritmo de Bernstein Basado en
--    Dependencias Funcionals Para Bases de Datos a Aartir de Una Tabla con Sitaxis Oracle
-- Desarollado por Irving Abad Castillo Fierro 2017

-- Creación de Codigo Automatico

-- Se han agregado los create table necesarios y sus respectivos atributos
-- NOTA TIPO DE DATO ILUSTRATIVO CAMBIAR POR EL NECESARIO

CREATE TABLE Nombre_Tabla_ (
id VARCHAR(65) ,
 Fecha VARCHAR(65) ,
 Total VARCHAR(65) ,
 id_cliente VARCHAR(65) ,
 Nombre_Cliente VARCHAR(65)
 );
CREATE TABLE Nombre_Tabla (
id_cliente VARCHAR(65) ,
 Nombre_Cliente VARCHAR(65)
 );
```

El botón información muestra detalles del desarrollador