UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE CIENCIAS

Desarrollo de una herramienta para
normalizar bases de datos relacionales

REPORTE DE ACTIVIDAD DOCENTE

QUE PARA OBTENER EL TÍTULO DE:

LICENCIADO EN CIENCIAS DE LA COMPUTACIÓN

P R E S E N T A :

DENIS OMAR VERDUGA PALENCIA

DIRECTOR DE TESIS:

DRA. AMPARO LÓPEZ GAONA

2008

Hoja de Datos del Jurado

Datos del alumno	
Apellido paterno	Verduga
Apellido materno	Palencia
Nombre(s)	Denis Omar
Teléfono	56968411
Universidad	Universidad Nacional Autónoma de México
Facultad	Facultad de Ciencias
Carrera	Ciencias de la Computación
Número de cuenta	40006784-1
2. Datos del tutor	
Grado	Dra.
Nombre(s)	Amparo
Apellido paterno	López
Apellido materno	Gaona
3. Datos del sinodal 1	
Grado	Mat.
Nombre(s)	Salvador
Apellido paterno	López
Apellido materno	Mendoza
4. Datos del sinodal 2	
Grado	M. en C.
Nombre(s)	Egar Arturo
Apellido paterno	García
Apellido materno	Cárdenas
5. Datos del sinodal 3	
Grado	Mat.
Nombre(s)	María Concepción Ana Luisa
Apellido paterno	Solís
Apellido materno	González-Cosío
6. Datos del sinodal 4	
Grado	M. en C.
Nombre(s)	María Guadalupe Elena
Apellido paterno	Ibargüengoitia
Apellido materno	González
7.Datos del trabajo escrito.	
Título	Desarrollo de una herramienta para normalizar bases de datos relacionales.
Número de páginas	91
Año	2008

Índice

Introducción

Este trabajo se plantea como la investigación, análisis e implementación de un sistema normalizador de relaciones para una situación planteada en el curso de bases de datos para estudiantes del curso, por ende se planea que esta herramienta sirva de apoyo para próximos cursos de Bases de Datos de la Facultad de Ciencias.

Dado que las Bases de Datos se han consolidado como el motor de las llamadas tecnologías de la información y la mayoría de ellas usa el modelo relacional como su fundamento teórico, no es difícil observar que la comprensión de este modelo tiene una importancia superlativa en el diseño y/o mejora de una base de datos y sus tablas.

El objetivo de este trabajo es mostrar el desarrollo de una herramienta que facilite la comprensión del tema de normalización de bases de datos relacionales, tema generalmente difícil de comprender en cursos introductorios de Bases de Datos.

Esta herramienta fue planeada para mostrar en pantalla los pasos importantes de las normalizaciones, implementando algoritmos de la teoría del modelo relacional, de tal forma que los alumnos puedan comprender con ejemplos los pasos detrás de las normalizaciones, así como que puedan tener acceso a la implementación de la herramienta para extender la funcionalidad de la misma.

Para fines de hacer legible el siguiente trabajo, se presenta la siguiente estructura:

Capítulo 1 Marco Teórico: En este capítulo se muestra una selección de definiciones, algoritmos y resultados útiles para la sección de las normalizaciones del modelo relacional.

Capítulo 2 Trabajos relacionados: Aquí se presentan herramientas actuales que buscan resolver la problemática de las normalizaciones en el área de las bases de datos relacionales.

Capítulo 3 Metodología y Tecnologías: Durante este capítulo se fundamenta la elección de tecnologías, arquitecturas y metodologías para el desarrollo de la herramienta.

Capítulo 4 Implementación: A lo largo de este capítulo se describe el proceso de implementación del sistema, así como una crónica de la retroalimentación recibida durante este proceso.

Posteriormente se termina el trabajo con las conclusiones e ideas para trabajos futuros,así como apartados de glosario y bibliografía.

1. Marco Teórico

1.1 Introducción.

El modelo relacional fue formalmente introducido por Codd en 1970 y desde entonces ha evolucionado a través de los años.
El modelo relacional provee un concepto sencillo, más rigurosamente definido, de cómo los usuarios perciben los datos. Dentro de este modelo, los datos son representados en tablas bidimensionales. Cada tabla representa algún objeto del mundo real y por tanto, una base de datos relacional es una colección de tablas bidimensionales.
La organización de los datos dentro de tablas relacionales es conocida como la vista lógica de una base de datos, esto es, la forma en la cual una base de datos relacional presenta datos al usuario y al programador.

1.2 Propiedades de las tablas relacionales.

Toda tabla relacional debe cumplir las siguientes propiedades:

Atomicidad de los valores: Esta propiedad implica que cada columna de cada renglón de una tabla relacional solo puede tener un valor, y no un conjunto de valores. Estas tablas se dice que están en Primera Forma Normal (ver apéndice de otras formas normales). Esta propiedad es uno de los pilares del modelo, y su principal beneficio es que esto simplifica la lógica de manipulación de los datos.

Los valores de las columnas son del mismo tipo: Esta propiedad simplifica el acceso a los datos ya que uno tiene la certeza del tipo de datos de una columna en particular. Esta propiedad también simplifica la validación de datos de las columnas.

Unicidad de renglones: Esta propiedad implica que cada renglón en una tabla relacional es significativo y que un renglón especifico puede ser identificado al especificar el valor de su llave primaria.

Irrelevancia de la secuencia de las columnas: Esta propiedad garantiza que la ordenación de las columnas no tiene importancia, puesto que el acceso a las columnas puede ser en cualquier orden o secuencia. Esto permite que la estructura física de la base de datos pueda cambiar sin que se afecte la base de datos relacional (la estructura lógica).

Irrelevancia de la secuencia de los renglones: Esta propiedad es análoga a la anterior, pero basándose en renglones. Debido a esta propiedad se pueden obtener renglones en diferente orden, así como ingresar renglones de manera más simple.

Unicidad del nombre de columna: Dado que la secuencia de las columnas no tiene importancia, las mismas deben ser ubicadas por nombre y no por posición, esta unicidad del nombre se limita solo a la tabla a la que pertenece dicha columna.

1.3 Definiciones

Antes de pasar a los algoritmos frecuentemente usados en el modelo relacional, es importante incluir algunas definiciones para el pleno entendimiento del trabajo.
Se prefirió usar el termino llave en lugar de clave, y por ende, todas las definiciones usan esta palabra.

Atributo: En el modelo relacional, un atributo es una característica presente en una relación. Para fines prácticos, se puede ver como cada columna(campo) de una tabla relacional.

Llave: Una llave es un atributo o conjunto de atributos que identifica de manera única a una tupla. La importancia de las llaves es que permiten identificar registros individuales en una relación. De hecho las llaves son una de las restricciones a especificar más importantes al momento de generar un esquema de base de datos.
En alguna literatura el termino se encuentra como superllave.

Superllave mínima (llave candidata): Una superllave mínima es una superllave $a_1,a_2..a_n$ a la cual, si se le sustrae un elemento a_i a dicho conjunto $a_1,a_2..a_n$ entonces el conjunto resultante $a_1,a_2..a_n- a_i$ no sigue siendo llave.

Llave primaria: Una llave primaria es una columna (o columnas) cuyos valores identifican plenamente cada renglón en una tabla.

Llave foránea: Una llave foránea (o externa) es una columna (o columnas) cuyos valores son idénticos a la llave primaria de otra tabla.

Tupla: Una tupla es un conjunto de valores de atributos sin un orden especifico.

Relación : Una relación es una asociación entre dos o más tablas, las relaciones son generalmente expresadas por los valores de las llaves primarias y foráneas.

Las llaves son fundamentales en el concepto de las bases de datos relacionales ya que permiten a las tablas estar relacionadas entre sí. Además, la navegación en una base de datos depende de la habilidad de la llave primaria para identificar sin ambigüedad renglones específicos de una tabla. La navegación entre tablas requiere que la llave externa sea capaz de referenciar los valores de una llave primaria en otra tabla de una manera correcta y consistente.

1.4 Conceptos básicos de las dependencias funcionales.

Una dependencia funcional es un tipo de restricción sobre el conjunto de atributos de una tabla en una base de datos. Las dependencias funcionales permiten expresar hechos provenientes del mundo real, el cual se esta modelando. La noción generaliza la idea de una llave según se muestra a continuación.

Sean $\alpha \subseteq \mathbf{R}$ y $\beta \subseteq \mathbf{R}$ dos conjuntos de atributos pertenecientes a la relación **R.** Entonces α determina a β, denotado por $\alpha \rightarrow \beta$ se mantiene en **R** si cada valor de α tiene asociado un solo valor de β. Usando esta notación, puede decirse que **K** es una llave de la relación R si $\mathbf{K} \rightarrow R$.

Si cualquier relación legal *r*(**R**), para todos los pares de tuplas $\mathbf{t_1}$ y $\mathbf{t_2}$ en *r* tales que $\mathbf{t_1}[\alpha]=\mathbf{t_2}[\alpha]$,también será el caso de $\mathbf{t_1}[\beta]=\mathbf{t_2}[\beta]$.

- En otras palabras, **K** es una *llave de* R si, para cualquier $\mathbf{t_1}[\mathbf{K}]=\mathbf{t_2}[\mathbf{K}]$, entonces $\mathbf{t_1}[R]=\mathbf{t_2}[\mathbf{R}]$ (y así $\mathbf{t_1}=\mathbf{t_2}$).

En resumen, se puede decir que una dependencia funcional es una relación entre atributos de una misma relación (tabla).

La dependencia funcional es una noción semántica. Si hay o no dependencias funcionales entre atributos no lo determina una serie abstracta de reglas, sino los modelos mentales del usuario y las reglas de negocio del cliente para el que se desarrolla el sistema de información.

Dependencia funcional trivial: Una dependencia funcional trivial es una dependencia funcional $\alpha \rightarrow \beta$, donde $\beta \subseteq \alpha$.

1.5 Cerradura del conjunto de dependencias funcionales

La importancia de la cerradura del conjunto de dependencias funcionales radica en el hecho de que la cerradura del conjunto de dependencias funcionales es el conjunto de dependencias funcionales lógicamente implicadas por el conjunto original de éstas.
Es por esto que una vez calculada la cerradura del conjunto, se tiene el universo de dependencias funcionales válidas para la relación.
La certeza de que se trata de dependencias funcionales válidas se basa en que todas las dependencias funcionales presentes en la cerradura son creadas a partir del conjunto original de dependencias funcionales mediante un conjunto de reglas de inferencia llamadas Axiomas de Armstrong[1,3].
Para obtener la cerradura es necesario considerar todas las dependencias funcionales que se presenten. Dado un conjunto F de dependencias funcionales, se puede probar que algunas otras también estarán presentes, en este caso se dice que son lógicamente implicadas por F.

Si se tiene $\mathbf{A \rightarrow B}$ y $\mathbf{B \rightarrow C}$ se puede concluir que $\mathbf{A \rightarrow C}$ es lógicamente implicada ya que si $\mathbf{t_1}$ y $\mathbf{t_2}$ son tuplas tales que

$$\mathbf{t_1}[\mathbf{A}] = \mathbf{t_2}[\mathbf{A}].$$

Como está dado que $\mathbf{A \rightarrow B}$, se sigue que también se debe tener

$$\mathbf{t_1}[\mathbf{B}] = \mathbf{t_2}[\mathbf{B}].$$

Mas aún, ya que se tiene $\mathbf{B \rightarrow C}$, se obtiene que

$$\mathbf{t_1}[\mathbf{C}] = \mathbf{t_2}[\mathbf{C}].$$

Por lo tanto, cualesquiera dos tuplas que tengan el mismo valor en $\mathbf{A}$, deberán tener el mismo valor $\mathbf{t_1}[\mathbf{A}] = \mathbf{t_2}[\mathbf{A}]$ en $\mathbf{C}$ y se dice que $\mathbf{A \rightarrow C}$.

La cerradura del conjunto de dependencias funcionales es el conjunto de todas las dependencias funcionales lógicamente implicadas por **F**, se denota por $\mathbf{F}^+$.

1.6 Axiomas de Armstrong.

Para calcular $\mathbf{F}^+$, se hace uso de unas reglas de inferencia llamadas Axiomas de Armstrong:

- **Reflexividad:** Si β es un conjunto de atributos y $\beta \subseteq \alpha$ entonces $\alpha \rightarrow \beta$ se cumple.
- **Aumentatividad:** Si $\alpha \rightarrow \beta$ se cumple, y γ es un conjunto de atributos entonces $\alpha\gamma \rightarrow \beta\gamma$ se cumple.
- **Transitividad:** Si $\alpha \rightarrow \beta$ se cumple, y $\beta \rightarrow \gamma$ también se cumple, entonces $\alpha \rightarrow \gamma$ se cumple.

Estas reglas no generan dependencias funcionales incorrectas y ellas generan todo $\mathbf{F}^+$.

Para auxiliar en el proceso del cálculo, se pueden usar unas reglas adicionales directamente derivadas de los axiomas de Armstrong [2].

Unión: Si $\alpha \rightarrow \beta$ y $\alpha \rightarrow \gamma$, entonces $\alpha \rightarrow \beta\gamma$ se cumple.

Descomposición: Si $\alpha \rightarrow \beta\gamma$ se cumple, entonces $\alpha \rightarrow \beta$ y $\alpha \rightarrow \gamma$ se cumplen.

Pseudotransitividad: Si $\alpha \rightarrow \beta$ se cumple, y $\beta\gamma \rightarrow \delta$ se cumple, entonces $\alpha\gamma \rightarrow \delta$ se cumple.

1.7 Cerradura del conjunto de atributos

Para probar si un conjunto dado de atributos α es una llave tenemos que encontrar el conjunto de atributos determinados funcionalmente, esto es, atributos que son lógicamente implicados por α ,y comparar el mismo con la totalidad de atributos presentes en la relación **R**.

Para calcular el conjunto de atributos determinados por un subconjunto de atributos α, se sigue el siguiente proceso.

" Sea α un subconjunto de atributos de **R**.Al conjunto de todos los atributos determinados funcionalmente por α bajo un conjunto **F** de dependencias funcionales se le llama cerradura de α bajo **F** y se representa por α^+_F.

" El algoritmo que calcula α^+_F, la cerradura de α bajo F, tiene como entrada el conjunto F de dependencias funcionales y el conjunto α de atributos.

Algoritmo para calcular la cerradura de α bajo F

```
resultado := α;
while (cambios en resultado) hacer
       for each  Dependencia Funcional  β → γ en F hacer
           begin
              if  β ⊆ resultado  then
                  resultado := resultado ∪ γ;
           end;
     α+F := resultado;
```

Una vez que el algoritmo termina, la salida del mismo se encuentra almacenada en la variable **resultado**.

1.8 Conjuntos de dependencias funcionales equivalentes y mínimas.

" Dados dos conjuntos de dependencias funcionales, F y G,se dice que son equivalentes si $F^+ = G^+$.

Equivalencia mínima.

Para explicar mejor el tema, es necesario introducir las definiciones de los siguientes términos.

Dependencia redundante: Una dependencia funcional **f** de un conjunto **F**, se dice que es redundante si puede ser deducida de **{ F-f }**; es decir, la cerradura del conjunto de dependencias funcionales de F es idéntica a la cerradura del conjunto de dependencias funcionales de **{F-f}**, esto es, si $\mathbf{F^+ = \{ F\text{-}f \}^+}$.

Atributo extraño (o superfluo): Dada la dependencia funcional $\alpha \rightarrow \beta$ en **F**, se dice que un atributo **A** es extraño o superfluo en α si $\mathbf{A} \in \alpha$ y además **F** implica lógicamente $\mathbf{F - \{ \alpha \rightarrow \beta \} \cup \{(\alpha - A) \rightarrow \beta \}}$. Además, se dice que un atributo **B** es extraño o superfluo en β si $\mathbf{B} \in \beta$, y el conjunto de dependencias funcionales $(\mathbf{F - \{ \alpha \rightarrow \beta \}}) \cup \mathbf{\{ \alpha \rightarrow (\beta \ B)\}}$implica lógicamente a **F**.

Dependencia elemental: Dada la dependencia funcional $\mathbf{\alpha \rightarrow \beta} \in \mathbf{F}$, se dice que es elemental si no tiene atributos extraños(superfluos).

Equivalente mínimo: Un conjunto de dependencias funcionales, **Fc**, es un equivalente mínimo de **F** si:

- $\mathbf{F^+ = Fc^+}$.
- Ninguna dependencia funcional de **Fc** tiene atributos superfluos.
- Cada lado izquierdo de las dependencias funcionales de **Fc** es único.

La importancia del equivalente mínimo (en algunas bibliografías se conoce como cobertura canónica) es que incluye una menor cantidad de dependencias funcionales y por tanto, las comprobaciones sobre una relación se simplifican.

A continuación cito textualmente el algoritmo mostrado en el libro de Silberschatz Database System Concepts[2].

Algoritmo para encontrar la cobertura canónica

Fc =F

do

- Usar la regla de unión para reemplazar dependencias de la forma

  $\alpha \rightarrow \beta_1$ y

  $\alpha \rightarrow \beta_2$ con

  $\alpha \rightarrow \beta 1 \beta 2$.

- Encontrar una dependencia funcional $\alpha \rightarrow \beta$ con un atributo superfluo a en β.
- Si un atributo superfluo es encontrado, borrarlo de $\alpha \rightarrow \beta$.

while Fc *cambie.*

1.9 Dependencias Multivaluadas

Las dependencias funcionales impiden que ciertas tuplas puedan aparecer en una relación. Si $\alpha \rightarrow \beta$, nosotros no podemos tener dos tuplas con el mismo valor α y distinto β.

Esta restricción no está presente dentro de otro tipo de dependencias llamadas dependencias multivaluadas. Las dependencias multivaluadas no impiden la existencia de esas tuplas,al contrario, requieren que dichas tuplas estén en la relación.

Sea **R** el esquema de la relación y sean $\alpha \subseteq$ **R** y $\beta \subseteq$ **R** dos conjuntos de

atributos.
La dependencia multivaluada

$$\alpha \twoheadrightarrow \beta$$

se conserva en **R** si cualquier relación legal *r*(**R**), para todos los pares de tuplas $\mathbf{t_1}$ y $\mathbf{t_2}$ en *r* tales que $\mathbf{t_1}[\alpha]=\mathbf{t_2}[\alpha]$, existen tuplas $\mathbf{t_3}$ y $\mathbf{t_4}$ en *r* tales que:

$\mathbf{t_1}[\alpha]=\mathbf{t_2}[\alpha]=\mathbf{t_3}[\alpha]=\mathbf{t_4}[\alpha]$
$\mathbf{t_3}[\beta]=\mathbf{t_1}[\beta]$
$\mathbf{t_3}[\mathbf{R}-\beta]=\mathbf{t_2}[\mathbf{R}-\beta]$
$\mathbf{t_4}[\beta]=\mathbf{t_2}[\beta]$
$\mathbf{t_4}[\mathbf{R}-\beta]=\mathbf{t_1}[\mathbf{R}-\beta]$

Esto también se puede definir como si para cada valor de α , hay un conjunto de valores de β, independientemente de los valores restantes de { **R** - α -β }.

Dependencia multivaluada trivial: Una dependencia multivaluada α **-»** β en **R** se dice trivial si se cumple que $\beta \subseteq \alpha$ ó β ? α = **R**.

1.10 Teoría de las dependencias multivaluadas

En algunos casos es necesario obtener todas las dependencias multivaluadas que son lógicamente implicadas por un conjunto dado de dependencias multivaluadas. En esos casos se sigue el siguiente procedimiento:

- Sea **D** el conjunto de dependencias funcionales y multivaluadas
- La cerradura $\mathbf{D}^+$ de **D** es el conjunto de todas las dependencias funcionales y multivaluadas lógicamente implicadas por **D**.

- Se puede calcular **D^+** usando definiciones, pero es más fácil usando el siguiente conjunto de reglas de inferencia, siendo importante mencionar que estas reglas solo se aplican a dependencias multivaluadas[2].

    - **Complementación:** Si **α -» β** se cumple, entonces **α -» R β - α** se cumple.
    - **Aumento multivaluado:** Si **α -» β** se cumple,y **γ ⊆ R** y **δ ⊆ γ** , entonces **α γ -» βδ** se cumple.
    - **Transitividad Multivaluada:** Si **α -» β** se cumple, y **β -» γ** se cumple, entonces **α -» γ - β** se cumple.
    - **Replicación:** Si **α → β** se cumple, entonces **α -» β**.
    - **Unión:** Si **α -» β** se cumple, y **γ ⊆ β** , y existe una **δ** tal que **δ ⊆ R** y **δ ∩ β = Φ** y **δ → γ**, se tiene que **α → γ** se cumple.

1.11 Normalización

La normalizacaión es una técnica para diseñar tablas en bases de datos relacionales que minimicen la duplicidad de información y por extensión, para cuidar la base de datos contra ciertos tipos de problemas en la lógica o en la estructura llamados anomalía de datos.

Por ejemplo, cuando múltiples instancias de una pieza de información están presentes en una tabla, existe la posibilidad de que esas instancias no puedan ser mantenidas consistentes cuando los datos de la tabla sean actualizados, derivando en una perdida de la integridad de los datos. Una tabla que esta normalizada es menos vulnerable a problemas de ese estilo, porque su estructura refleja las asunciones básicas para que cuando múltiples instancias de la misma información

deban estar en una tabla puedan ser representadas por una única instancia.

En caso de que la base de datos presente un diseño pobre (es decir, que se presenten anomalías al efectuar operaciones sobre la misma) seria conveniente pensar en la normalización. Cuando se quiere normalizar el diseño de una base de datos se debe tener cuidado con la descomposición, es decir, refinar una relación grande descomponiéndola en un conjunto de varias relaciones más pequeñas que cumplen un conjunto de propiedades determinadas por la forma normal en la que se desea tener la base de datos.

La función de la normalización es eliminar la redundancia, ganando al mismo tiempo espacio dentro de la base de datos. Al hacer uso de la normalización también se desea que se eliminen las anomalías al borrar y actualizar datos dentro de las distintas tablas de la base de datos.

Ejemplo:
En esta tabla sin normalizar se presentan las siguientes anomalías.

Redundancia: La información se repite innecesariamente en muchas tuplas, en esta tabla, las columnas Duración y Tipo.

Anomalías de actualización: Esto es, si cambiamos la información de una tupla no se efectúa la actualización en la otra. En este caso podríamos actualizar la columna Duración de la primera tupla de Star Wars de 124 a 125. Dicho cambio tendría que ser efectuado en cada tupla de esa película.

Anomalías de eliminación: Si eliminamos al Actor Emilio Estevez, perderemos toda la información de la tupla asociada a dicho actor.

Película	Año	Duración	Tipo	Estudio	Actor
Star Wars	1977	124	color	Fox	Carrie Fisher
Star Wars	1977	124	color	Fox	Mark Hamill
Star Wars	1977	124	color	Fox	Harrison Ford
Mighty Ducks	1991	104	color	Disney	Emilio Estevez
Wayne's World	1992	95	color	Paramount	Dana Carvey
Wayne's World	1992	95	color	Paramount	Mike Meyers

Tabla 1. Relación de películas.

La normalización puede ser tanto correctiva como preventiva, ya que si bien puede ser aplicada desde el diseño de la base de datos como parte de un aseguramiento de calidad de la misma, también se usa para optimizar el tamaño de la base de datos en una etapa posterior al diseño.

1.12 Descomposición

Para normalizar una base de datos se tiene que aplicar un algoritmo de descomposición sobre cada relación (tabla) presente en dicha base de datos.

Para que dicha descomposición sea útil, se deben buscar las siguientes propiedades.

- **Que los esquemas resultantes no tengan perdida de información en una reunión:** Esto es, que las nuevas tablas resultantes de la descomposición puedan ser reunidas mediante una reunión natural (natural join) en una nueva tabla sin que se generen datos incorrectos o alterados en la misma. Una reunión natural, según Silberschatz , es una simplificación de productos cartesianos que requieren operaciones de selección sobre el resultado[2], es decir, es la combinación de dos operaciones , un producto cartesiano y una selección, sobre tablas relacionales.
  Por ejemplo, un producto cartesiano de la siguiente tabla T1

Atributo1	Atributo2
a	c
b	d

con la siguiente tabla T2

Atributo2	Atributo3
a	x
c	y

arrojaría la siguiente tabla

Atributo1	T1.Atributo2	T2.Atributo2	Atributo3
a	c	a	x
a	c	c	y
b	d	a	x
b	d	c	y

mientras que la reunión natural se limita al siguiente resultado

Atributo1	Atributo2	Atributo3
a	c	y

puesto que ambas tablas coinciden en un atributo (el atributo c sirve como pivote para hacer la operación de selección de la reunión)

- **Que se conserven las dependencias:** Esto es, que en cada actualización de la base de datos, se debe comprobar que no se generen relaciones ilegales, es decir, relaciones que no cumplan las dependencias funcionales o multivaluadas dadas.
- **Eliminación de repetición de información**: Una de las funciones

principales de la normalización es no repetir datos (la no redundancia es deseable) pero el grado hasta el que se desee eliminar la redundancia es dependiente de cada forma normal.

Para el caso de la descomposición se cuenta con distintos algoritmos dependiendo de la forma normal deseada.

1.13 Formas Normales

Por el alcance de la herramienta, las formas que se cubrirán en la siguiente sección son tres, dos formas para dependencias funcionales (la Forma Normal Boyce-Codd y la Tercera Forma Normal) y una para dependencias multivaluadas (Cuarta Forma Normal).

1.13.1 Formas normales para dependencias funcionales

a) Forma Normal Boyce-Codd:

Un esquema relacional se dice que está en **Forma Normal Boyce-Codd** (FNBC) con respecto a un conjunto **F** de dependencias funcionales si para todas las dependencias funcionales en **F** de la forma **α → β** donde **α ⊆ R** y **β⊆ R** al menos una de las siguientes condiciones se cumple:

- **α → β** es una dependencia funcional trivial(es decir **β ⊆ α**).
- **α** es una superllave de **R**.

Una vez dicho esto, se dice que una base de datos está en FNBC si cada uno de los esquemas están en FNBC.

El algoritmo que se utiliza para una descomposición a FNBC es el siguiente:

Algoritmo de descomposición a Forma Normal Boyce-Codd

```
resultado := [R] ;
done := false;
calcular F+ ;
while ( not done) do
     if ( existe algún esquema Ri en resultado que aun no se
          encuentre en FNBC)
      then begin
          sea α → β una dependencia funcional no trivial que se
          cumple en Ri tal que α → Ri no este en F+, y α ∩ β = Φ;
             resultado = (resultado  Ri)∪( Ri - β )∪{ α , β };
   end
 else done = true;
```

Este proceso produce una descomposición de reunión sin perdida,ya que si se reemplaza a R_i con $R_i - \beta$ y $\{\alpha, \beta\}$ tenemos que la dependencia $\alpha \rightarrow \beta$ permanece cumpliéndose en R_i y $(R_i - \beta) \cap \{\alpha, \beta\} = \alpha$.

Por otra parte, García-Molina y Ullman[1] proponen una heurística donde basan la descomposición de las dependencias funcionales que violan la FNBC.
El proceso se basa en tomar cada dependencia funcional $\alpha \rightarrow \beta$ que viole la FNBC en el esquema **R.**
Dicha dependencia se convierte en la nueva dependencia $\alpha \rightarrow \beta \cup (\alpha)^+$, con lo cual e forman dos subesquemas: el primer subesquema contiene los atributos presentes en la nueva dependencia, esto es, los atributos $\beta \cup (\alpha)^+$.

El segundo subesquema contiene los atributos restantes del esquema original, menos los atributos presentes en el lado derecho de la dependencia creada, es decir **R − (β ∪ (α)⁺) ∪ α.**

Con este procedimiento, el primer subesquema está en FNBC, y contendrá las dependencias cuyos atributos estén presentes entre los atributos de la partición.

El segundo subesquema tendrá el resto de los atributos y dependencias que no estén en el primer subesquema.

Este procedimiento se sigue hasta que todas las dependencias se encuentren en un subesquema normalizado.

Pese a los algoritmos aquí presentados para la descomposición a FNBC hay que tener en mente que no siempre se podrá dar el caso de tener una reunión sin perdida, una preservación de dependencias y el esquema en FNBC.

b) Tercera Forma Normal

Cuando no se pueden hacer coincidir los tres criterios anteriores, se debe pensar en aceptar una forma con menos restricciones llamada **Tercera Forma Normal (3FN)**.

Una relación esta en **3FN** con respecto al conjunto F de dependencias funcionales si para todas las dependencias funcionales en $\mathbf{F^+}$ de la forma $\boldsymbol{\alpha \rightarrow \beta}$, donde $\boldsymbol{\alpha \subseteq R}$ y $\boldsymbol{\beta \subseteq R}$, al menos una de las siguientes condiciones se cumple:

- $\boldsymbol{\alpha \rightarrow \beta}$ es una dependencia funcional trivial.
- $\boldsymbol{\alpha}$ es una superllave.
- Cada atributo **A** en $\boldsymbol{\beta - \alpha}$ esta contenido en una llave candidata para **R**.

De tal forma que una base de datos se dice que está en Tercera Forma Normal si cada relación de la base de datos está en Tercera Forma Normal.

Con esta nueva forma normal, se permiten dependencias que satisfagan

solo la tercera condición. Estas dependencias son llamadas dependencias transitivas y no son permitidas en FNBC.
Como se puede ver, si una relación esta en FNBC también lo estará en 3FN. La Forma Normal Boyce Codd es más restrictiva que la 3FN.
Su ventaja sobre la Forma Normal Boyce Codd es que siempre es posible encontrar una descomposición en 3FN que preserve dependencias y no pierda datos durante la reunión.
El procedimiento para normalizar una relación a 3FN es el siguiente:

Algoritmo de descomposición a Tercera Forma Normal

Sea **F** el conjunto de dependencias funcionales en forma canónica.

Sea $\mathbf{F_C}$ la cobertura canónica para **F**;

```
i := 0;

for each dependencia funcional α → β , donde α ⊆ Fc do
if ninguno de los esquemas contiene αβ
 then begin
 i := i + 1;
 Ri := αβ
 end
 if ninguno de los esquemas Rj , 1 ≤ j ≤ i
  contiene una llave candidata para R then
  begin
      i := i + 1;
      Ri := cualquiera llave candidata para R
  end
  return ( R 1 ,R 2 ...,R i )
```

Como observaciones finales, al tener que elegir entre FNBC o

preservación de dependencias, es mejor optar por la 3FN, ya que si no se tiene cuidado en la preservación de las dependencias uno tiene que pagar un precio en el rendimiento del sistema o en la integridad de los datos con lo que la cantidad de redundancia que existe en la 3FN es siempre un mal menor.

Obviamente, siempre se desea que la base de datos permita la Forma Normal Boyce Codd pero, si no se puede, al menos se cuenta con la Tercera Forma Normal que permite reuniones sin pérdida y que conserva todas las dependencias.

Ahora bien, la descomposición tiene un costo, ya que para unir la información descompuesta en piezas, hay que unir los datos en base a reuniones naturales o productos cartesianos, lo cual, por supuesto, toma tiempo computacional.

1.13.2 Forma Normal para dependencias multivaluadas

Cuarta Forma Normal

Una relación R está en Cuarta Forma Normal (4FN) con respecto a un conjunto **D** de dependencias tanto funcionales como multivaluadas si

para todas las dependencias multivaluadas en $\mathbf{D}^{+}$ de la forma

α -» β donde **α ⊆ R** y **β ⊆ R**, al menos una de las siguientes condiciones se cumple:

- **α -» β** es una dependencia multivaluada trivial
- **α** es superllave para el esquema R

Es importante recordar que la definición de dependencia multivaluada trivial difiere de la definición de dependencia funcional trivial.

Con esto, una base de datos se dirá que está en 4FN si cada relación del conjunto de relaciones está en 4FN.
Como se puede ver, la diferencia entre las definiciones de la Cuarta

Forma Normal y la Forma Normal Boyce Codd difiere en que la primera usa a las dependencias multivaluadas. Por ende, todo esquema en 4FN esta en FNBC.

Si un esquema no está en FNBC, hay alguna dependencia funcional no trivial **α → β** que permanece en *R*, donde **α** no es una superllave.

Ya que **α → β** implica **α -» β**, por la regla de replicación mostrada previamente, **R** no puede estar en 4FN.

Por su similitud con FNBC, se tiene un algoritmo similar para la descomposición de un esquema a 4FN.

Algoritmo de descomposición a Cuarta Forma Normal

resultado := [**R**];
done := false;
calcular $\mathbf{D}^+$;
if (existe un esquema $\mathbf{R_i}$ en resultado que no este en 4FN)
while (not done) **do**
then begin
Sea **α -» β** una dependencia multivaluada no trivial que se cumple en $\mathbf{R_i}$ tal que **α → $\mathbf{R_i}$** no esta en $\mathbf{D}^+$, y **α ∩ β = Φ**;
resultado = **(** resultado **$\mathbf{R_i}$)∪($\mathbf{R_i}$ - β)∪(α , β)** ;
end
else done = true;

Este algoritmo genera solo descomposiciones con reuniones sin perdida

Sea **R** una relación y **D** un conjunto de dependencias, tanto funcionales como multivaluadas sobre *R*.

Sean tanto $\mathbf{R_1}$ como $\mathbf{R_2}$ la descomposición de **R**.

Esta descomposición no tiene perdida en la reunión si y solo si al menos una de las siguientes dependencias multivaluadas esta en D^+:

a) $R_1 \cap R_2 \twoheadrightarrow R_1$

b) $R_1 \cap R_2 \twoheadrightarrow R_2$

Sin embargo, la preservación de dependencias es no tan simple como con las dependencias funcionales.

Sea **R** una relación dentro del esquema.

Sean $R_1, R2, \ldots R_n$ una descomposición de **R**.

Sea **D** el conjunto de dependencias funcionales y multivaluadas que se cumplen en **R**.

La restricción de **D** a R_i es el conjunto D_i consistente de:

Todas las dependencias funcionales D^+ que incluyen solo atributos de R_i.

- Todas las dependencias multivaluadas de la forma $\alpha \twoheadrightarrow \beta \cap R_i$ donde $\alpha \subseteq R_i$ y $\alpha \twoheadrightarrow \beta$ está en D^+.

Una descomposición del esquema R preserva dependencias con respecto al conjunto D de dependencias funcionales y multivaluadassi por cada conjunto de relaciones **r1(R1),r2(R2)...rn(Rn)** tales que para toda i, **ri** que satisface **Di**, existe una relación **r(R)** que satisface **D** y para la cual **ri = ΠR1(r)** para toda i, donde **ΠR1(r)** es la proyección de r.

Conclusión.

Como se mencionó previamente, estos algoritmos, si bien no son los únicos presentes dentro de la teoría del modelo relacional para la normalización de las bases de datos, fueron los seleccionados para ser incluidos en la herramienta, puesto que son los que usualmente se cubren dentro del temario del curso.

2. Trabajos relacionados

2.1 Antecedentes

Durante la carrera de Ciencias de la Computación de la Facultad de Ciencias de la Universidad Nacional Autónoma de México es obligatorio cursar la asignatura de Bases de Datos, en dicho curso existe un tema llamado normalización de Bases de Datos.

La teoría tras las dependencias funcionales y las formas normales es fundamental para el trabajo de normalización.

El modelo relacional fue propuesto en 1970 por Codd,y su principal característica es almacenar los datos de la base de datos como relaciones. [1]

2.2 Importancia práctica.

Las relaciones que violan las formas normales pueden ser modificadas por un procedimiento llamado normalización.

La normalización es un conjunto de heurísticas y algoritmos que tiene, como algunos de sus fines, eliminar la redundancia en sus registros, ahorrar espacio, facilitar la modificación de los datos, simplificar las restricciones de integridad referencial y evitar las anomalías por modificación. Como resultado se debe tener una representación de los datos más cercana al mundo real. En resumen, al hablar de normalización se habla del proceso de crear una estructura relacional apropiada, eficiente, confiable y flexible para almacenar información.

Dadas las ventajas que se adquieren al contar con una base de datos eficiente se han desarrollado herramientas que permiten detectar si una base de datos se encuentra normalizada o presenta anomalías, e incluso herramientas que proponen esquemas alternativos que solucionen dichas anomalías. Más dichos sistemas no son presentados en este trabajo dado que la idea tras el sistema presentado en el trabajo actual es crear la herramienta que desglose conceptos

mencionados en el tema del modelo relacional, lo cual conlleva a que dentro de la investigación se buscaron sistemas parecidos enfocados a una problemática semejante, es decir, la búsqueda de herramientas similares fue realizada dentro del ámbito académico mundial.

2.3 Herramientas educativas para temas del modelo relacional de Codd

Dentro del conjunto de herramientas presentadas, se considera pertinente mencionar algunas herramientas que son creadas por alumnos de universidades con la misma problemática, dichas herramientas son:

FPLUS

Fplus es una aplicación desarrollada por Germán Viscuso.
Este programa implementa algoritmos de la teoría de las bases de datos, como normalización, cerradura de atributos y calculo de claves.
Se desarrollo como un applet y se puede ejecutar desde un navegador web con tecnología java 1.4.
Este programa se encuentra en el repositorio de sourceforge, en la dirección electrónica:

http://fplus.sourceforge.net

DataBase Normalization Tool

DataBase Normalization Tool (en lo sucesivo denotado como DBNT), creado en la Universidad de Cornell. Este proyecto fue desarrollado por Scott Selikoff.
Dentro de las metas de este proyecto estaba el desarrollar una herramienta para reforzar el entendimiento en áreas problemáticas para otros estudiantes.
Este programa sólo trabaja sobre dependencias funcionales.
La herramienta encuentra la forma normal de la relación introducida,

así como llaves y la cobertura mínima.
Durante la navegación dentro del sistema se despliega información teórica acerca de los resultados mostrados en pantalla.
Este programa fue realizado sobre la tecnología j2ee de Java Server Pages (JSP).
Esta herramienta mantiene el siguiente sitio en internet:

http://dbtools.cs.cornell.edu/norm_index.html

NORMIT

NORMIT es un sistema de tutoría inteligente, desarrollada con el fin de que los estudiantes del curso introductorio a bases de datos de la Universidad de Canterbury en Nueva Zelanda mejoraran sus habilidades en el tema de normalización de datos.
El sistema NORMIT presenta las siguientes funcionalidades.

- Cálculo de claves candidatas.
- Cálculo de cerradura de conjuntos de atributos.
- Cálculo de atributos primos.
- Simplificación de dependencias funcionales.
- Determinación de la forma normal de una tabla.
- En ciertos casos, puede descomponer una tabla dada en un conjunto de tablas que se encuentren en Forma Normal Boyce Codd.
- Este sistema no menciona manejo de dependencias multivaluadas.

El sistema fue desarrollado en Allegro Common Lisp por Antonija Mitrovic.
La dirección del sitio en Internet del sistema es:

http://www.cosc.canterbury.ac.nz/tanja.mitrovic/normit.html

Normalizer

Normalizer, desarrollado por Giacomo del Rio en la Universidad de Pisa es un programa que permite probar algunos algoritmos de las bases de datos relacionales. Sus principales funciones son probar si una relación satisface las Formas Normales Tercera y Boyce-Codd,

descomponer una relación en relaciones que cumplan las formas normales previas, encontrar todas las llaves y los atributos primos de una relación dada, calcular la cerradura de un conjunto de atributos, calcular proyecciones de conjuntos de dependencias y encontrar coberturas canónicas de un conjunto de dependencias.

Esta herramienta se diferencia de las anteriores herramientas en que ya maneja el concepto de la Tercera Forma Normal, y es un sistema de escritorio, mientras las anteriores se trataban de aplicaciones Web.

Este programa está desarrollado en Java y se puede descargar desde el siguiente sitio:

http://www.cli.di.unipi.it/~delrio/normalizer/normalizer.html

DataBase Normalizer Application

Database Normalizer Application es un sistema que trabaja con dependencias funcionales para calcular las normalizaciones de un esquema de una base de datos relacional. El sistema puede determinar la forma normal de un esquema dado y calcular llaves candidatas y tuplas equivalentes. Otra característica presente en este programa es la implementación del algoritmo de síntesis que puede crear esquemas relacionales, garantizando que estén en Tercera Normal y contengan un conjunto mínimo de relaciones.

Esta herramienta se distingue de las herramientas previas en que si bien tiene una utilidad académica, el creador espera que en la práctica ayude así mismo a creadores de bases de datos para que comprueben su esquema con respecto a la normalización. Este sistema fue creado por Elmar Jurgens , en la Universidad de Munich, usando el lenguaje C# del 2002 al 2004 y se puede descargar del sitio:
http://home.in.tum.de/~juergens/html-data/DatabaseNormalizer/index.htm

Normal Forms: Schema Refinement
Esta versión fue desarrollada en Prolog por Anthony Aaby. Fue desarrollado en Diciembre del 2004.
Se encuentra en la dirección:

http://moonbase.wwc.edu/~aabyan/415/Normal.html

Dentro del sitio el creador pone a disposición el código fuente en Prolog, así como algunos ejercicios para demostrar la funcionalidad de la aplicación.

2.4 Conclusiones de la investigación de antecedentes.

Después de analizar la funcionalidad proporcionada por estas herramientas, así como el temario del curso de bases de datos se decidió que la funcionalidad mínima requerida por el sistema debería incluir:

- Cálculo de claves
- Cálculo de cerraduras transitivas
- Cálculo de cobertura mínima
- Determinación de forma normal de un esquema dado
- Descomposición de esquemas a esquemas normalizados en alguna forma normal (1FN, 2FN, 3FN,4FN, 5FN y FNBC)

Los dos últimos puntos se limitan a dos formas normales, la Forma Normal Boyce Codd y la Tercera Forma Normal, pues a pesar de que estas herramientas ya lo presentan, al momento de seguir el temario de la clase de Bases de Datos se observa que los mismos no son cubiertos durante el curso.
También se percibió que no se hacia uso de las dependencias multivaluadas dentro de las herramientas presentadas, sin embargo, dentro de la herramienta propuesta en el presente trabajo se decidió incluirla, puesto que es una forma normal presente dentro del temario del curso de Bases de Datos.

3. Metodología y tecnologías

3.1 Pasos a seguir.

En este capítulo se describen los pasos que se siguieron para desarrollar el Sistema Normalizador de Relaciones. La secuencia que se siguió fue:

a) Elegir la arquitectura de la aplicación, para saber qué tecnología era posible usar.
b) Definir el lenguaje de etiquetas adecuado, siendo siempre un lenguaje parecido a SGML.
c) Definir en qué lenguaje de programación sería conveniente la codificación del sistema.
d) Definir bajo qué orientación sería el análisis de los documentos.
   (Dependiente del lenguaje de etiquetas del inciso b)
e) Definir la implementación del analizador sintáctico.
f) Definir las etiquetas que forman la definición del documento.
g) Definir el alcance de la aplicación, es decir, sus casos de uso.

3.2 Elegir la arquitectura de la aplicación.

Se eligió que la aplicación fuera una aplicación de escritorio, puesto que el sistema no presenta persistencia en sus resultados y el punto fino de la misma es cómo se modelan los esquemas para trabajar con los datos. La elección del formato XML fue tomada en base a la sencillez del mismo.

3.3 Lenguaje de etiquetas.

Se decidió usar el lenguaje XML, el cual es una derivación de SGML. En los últimos años XML ha experimentado un auge debido a que facilita estructurar los datos de una manera auto explicativa.

Por esto se definió que el modo de obtener la información de las relaciones sería analizar documentos XML de forma que se pudieran definir etiquetas de acuerdo a las necesidades del sistema[4].
En resumen, la elección del lenguaje de etiquetas XML fue tomada en base a:

- XML es independiente de sistemas y plataformas.
- En XML se pueden generar etiquetas específicas para las necesidades de cada caso.
- Las etiquetas son auto descriptivas, es decir, hacen claro el contenido de su información para un actor externo.
- Existen gran cantidad de analizadores sintácticos.

3.4 Lenguaje de Programación.

Una vez definida la arquitectura de la aplicación, el siguiente paso fue la elección del lenguaje para manipular esos datos, en este caso la elección fue el lenguaje orientado a objetos Java.

Por el enfoque de los algoritmos también se pensó en crear el sistema en Prolog y hacer una interfaz en Java o C (incluso en alguna biblioteca gráfica del mismo lenguaje), pero debido a la experiencia en desarrollos en Java, la elección fue hacer todo el sistema en este lenguaje.

Después de fijar a Java como el lenguaje para hacer la implementación de los algoritmos, el siguiente paso fue definir que tipo de analizador sintáctico sería el adecuado para el sistema.
(En el capitulo anterior se vio que los esfuerzos previos usaban lenguajes de programación como Lisp o Prolog, que se podría pensar que son mas adaptables al problema, pero aquí entra en juego la idea de que el Sistema Normalizador debería ofrecer el código para la comunidad de la Facultad de Ciencias, y en cursos de introducción el

lenguaje que se ve es Java.)

3.5 Tipos de analizadores sintácticos

Un analizador sintáctico procesa el documento XML y verifica que esté bien formado (y/o válido), es decir , que cumpla con la sintaxis de este lenguaje de etiquetas y si aplica, con una serie de restricciones para el documento en cuestión.

Un analizador sintáctico es el núcleo de cualquier aplicación XML, ya que a través de él se puede trabajar con documentos XML.
Es necesario en la herramienta presentada debido a que se requiere recorrer el documento XML constatando que contenga las etiquetas previamente definidas y en el caso de este sistema, trasladar los datos entre las etiquetas a instancias de clases en Java para que puedan ser manipuladas por los algoritmos.

Existen dos grandes API (Application Programming Interface) para el análisis de documentos XML: DOM y SAX, cada una con un enfoque distinto que se resume a continuación en la siguiente tabla[7].

SAX (Simple API for XML)	**DOM** (Document Object Model)
Lee el documento XML secuencialmente de principio a fin, sin cargar todo el documento en memoria	El documento se carga totalmente en memoria en una estructura de árbol
Ventajas • Eficiencia en cuanto al tiempo y la memoria empleados en el análisis • Mayor facilidad para recorrer de manera	Ventajas - Fácil acceso a datos en función de la jerarquía de elementos - Modificación del contenido de los documentos

SAX (Simple API for XML)	**DOM** (Document Object Model)
secuencial un documento	
Desventaja No se dispone de la estructura en árbol de los documentos	Desventaja El costo en tiempo y memoria que conlleva construir el árbol
Es el estándar de facto para procesamiento de XML basado en eventos	

Comparativa SAX-DOM

3.6 Elección de API e implementación

Una vez que se investigaron las fortalezas y debilidades de SAX y DOM, se procedió a hacer un análisis de qué tanta funcionalidad requería el sistema.

Se planeó que el analizador debería pasar los datos a objetos en java conforme los fuera recorriendo, dejando el documento XML intacto, lo que era más simple de hacer en la API SAX.

Una vez definido esto, se observó que los documentos XML nada mas tendrían que ser recorridos una vez, sin necesidad de manipularlos directamente a través del analizador, con lo que finalmente la API SAX quedo confirmada como la API a usar dentro del sistema, puesto que era la API más sencilla y adecuada para el trabajo requerido.

SAX viene en distintas implementaciones para Java, siendo las mas conocidas Crimson y Xerces, ambas de Apache Software Foundation, en este caso no hubo mucho análisis y se eligió Xerces por ser el

analizador que contiene la distribución de java 1.5.

3.7 Representación del esquema relacional: Definición de etiquetas XML

Se decidió que la herramienta se basaría en convertir representaciones de modelos relacionales previamente generados en formato XML a instancias de clases en lenguaje java.

Se eligió el formato XML por ser un metalenguaje de etiquetas, ampliamente usado actualmente, ya sea en combinación de HTML, o, como en este caso, por ser una forma de generar documentos auto descriptivos facilitando el intercambio de datos.

Para llegar a su representación, se ubicaron dos componentes mínimos de una Relación presente en cualquier ejercicio del tema de Normalización de Bases de Datos en la asignatura de Bases de Datos de la Facultad de Ciencias, los cuales eran un conjunto de atributos y un conjunto de dependencias.

Así mismo, las dependencias tenían tres componentes : un lado izquierdo o el antecedente de la dependencia, un lado derecho o consecuente de la dependencia y un elemento que determinaba si se trata de una dependencia funcional o multivaluada.
Todo antecedente o consecuente tiene una restricción : que solo puede ser formada por miembros del conjunto de atributos excepto el elemento vacío.

Una vez que se contaba con una representación conceptual, se definió que los documentos XML que contienen la información útil para el sistema siempre deberán contener las siguientes etiquetas:

Etiqueta	Descripción
<RELACION>	Etiqueta que engloba la definición del esquema de la base de datos
<ATRIBUTOS>	Indica que a continuación se encontrara una lista de atributos presentes en las relaciones del modelo
<ATS>	Encerrado entre esta etiqueta se encuentra el nombre de un atributo
<DF>	Indica que a continuación se encontrara una lista de dependencias, ya sea funcionales o multivaluadas que se presentan en el esquema
<LADO_IZQUIERDO>	Indica que inicia la descripción de atributos presentes en el lado izquierdo de la dependencia
<MULTIVALUADO valor=X /> con X := true | false	Etiqueta que determina si la dependencia es multivaluada o funcional, dependiendo de la cadena asignada a valor
<LADO_DERECHO>	Indica que inicia la descripción de atributos presentes en el lado derecho de la dependencia
<ATT>	Atributo presente en alguno de los dos lados de la dependencia (lado izquierdo o derecho).

Etiquetas propuestas.

Obviamente, todas etiquetas tienen sus respectivas etiquetas de final:
</RELACION>, </ATRIBUTOS>, </ATS>, </DF>, </LADO_IZQUIERDO>, </LADO_DERECHO>, </ATT>.

Un caso especial es la etiqueta <MULTIVALUADO ...../>, pues esta lo

incluye en una sola definición.

Ahora bien, se decidió hacer uso de validaciones para los esquemas introducidos mediante un documento DTD (Document Type Definition), el cual es un documento que describe la estructura de los datos presentes en el archivo XML, así como la sintaxis de los mismos.

A continuación, para clarificar el análisis del uso de las etiquetas en un documento XML, así como la DTD que se definió, se muestra a continuación un ejemplo.

Ejemplo:

El esquema de la relación es el siguiente:

```
    R(a,b,c,d,e) con
    dependencias funcionales
    ab -> c ,
    de -> c ,
    b  -> d
```

Se vera en el formato XML previamente definido como:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE RELACION SYSTEM "tesisDTD.dtd">
<RELACION>
 <ATRIBUTOS>
     <ATS>a</ATS>
     <ATS>b</ATS>
     <ATS>c</ATS>
     <ATS>d</ATS>
```

```
    <ATS>e</ATS>
 </ATRIBUTOS>
 <DF>
           <LADO_IZQUIERDO>
          <ATT>a</ATT>
          <ATT>b</ATT>
    </LADO_IZQUIERDO>
    <MULTIVALUADO valor="false"/>
    <LADO_DERECHO>
          <ATT>c</ATT>
    </LADO_DERECHO>
    <LADO_IZQUIERDO>
          <ATT>d</ATT>
          <ATT>e</ATT>
    </LADO_IZQUIERDO>
    <MULTIVALUADO valor="false"/>
    <LADO_DERECHO>
          <ATT>c</ATT>
    </LADO_DERECHO>
    <LADO_IZQUIERDO>
          <ATT>b</ATT>
    </LADO_IZQUIERDO>
    <MULTIVALUADO valor="false"/>
    <LADO_DERECHO>
          <ATT>d</ATT>
    </LADO_DERECHO>
 </DF>
</RELACION>
```

donde tesisDTD.dtd contiene la siguiente información:

```
<?xml version='1.0' encoding='UTF-8'?>
<!ELEMENT RELACION (DF|ATRIBUTOS)*>
```

```
<!ELEMENT ATRIBUTOS (ATT)*>
<!ELEMENT ATT (#PCDATA)>
<!ELEMENT DF (LADO_DERECHO|MULTIVALUADO|LADO_IZQUIERDO)*>
<!ELEMENT LADO_IZQUIERDO (ATT)*>
<!ELEMENT MULTIVALUADO EMPTY>
<!ATTLIST MULTIVALUADO
    valor CDATA #IMPLIED
  >
<!ELEMENT LADO_DERECHO (ATT)*>
```

Ejemplo de una relación en el formato XML propuesto.

3.8 Funcionalidad deseada. Casos de Uso

Debido a que la herramienta se planeó para una sección bien limitada del temario del curso de Bases de Datos que se imparte en la Facultad de Ciencias, UNAM, los casos de uso eran limitados a los vistos en el temario.

El sistema cubre las normalizaciones Boyce Codd, Tercera Forma Normal y Cuarta Forma Normal, y, puesto que la teoría que se usó es la indicada en el primer capítulo, también era útil indicar a modo de opción el resultado de otros algoritmos. Los tres algoritmos que se consideran usualmente dentro del tema del curso son:

a) Llaves
b) Cerradura Transitiva
c) Cobertura Canónica

Puesto que son parte fundamental de la teoría del modelo relacional y, por consecuencia, del tema de normalizaciones.

A continuación se presenta el diagrama UML de casos de uso propuesto para la herramienta:

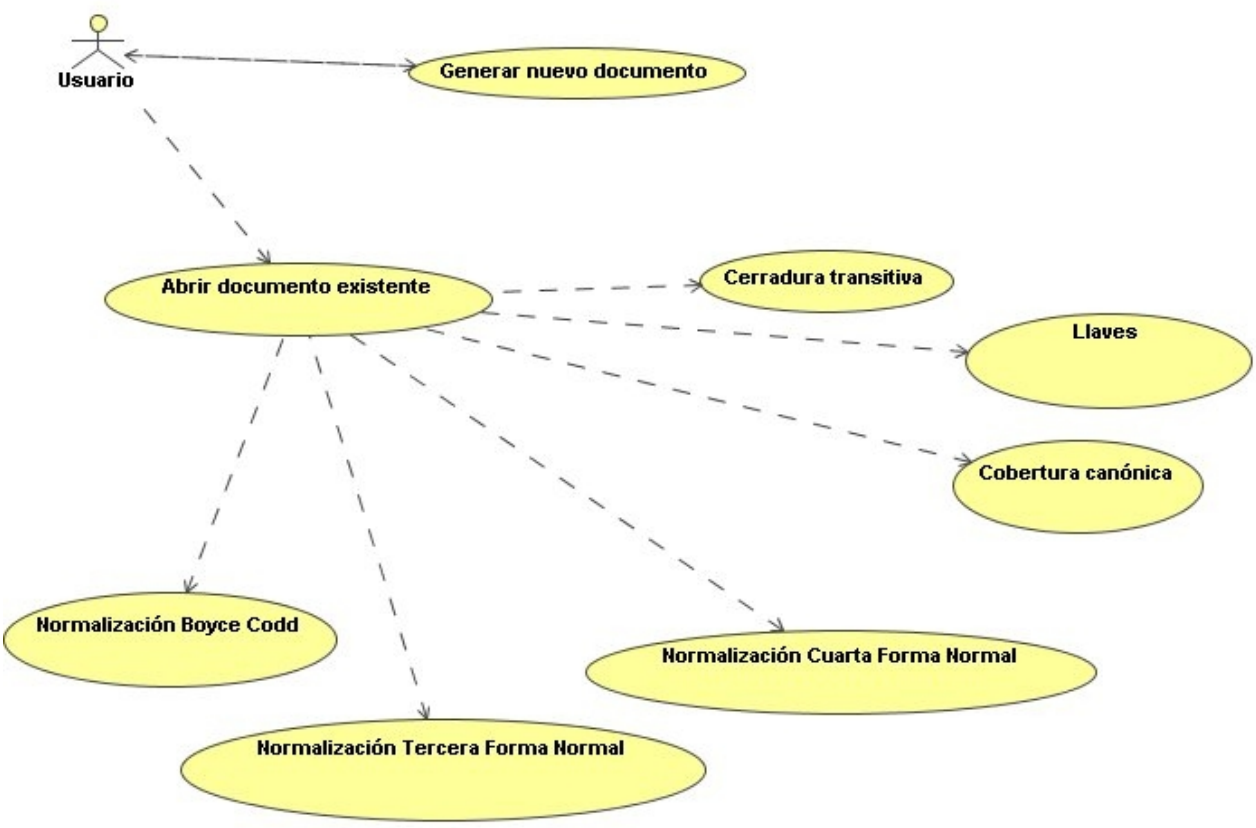


Figura 1: Diagrama de casos de uso

3.8.1 Caso 1: Generación de un nuevo documento.

Se despliega una pantalla para definir atributos de la relación, dependencias entre atributos y si las mismas son multivaluadas o funcionales.

El usuario elige el nombre y guarda el documento.

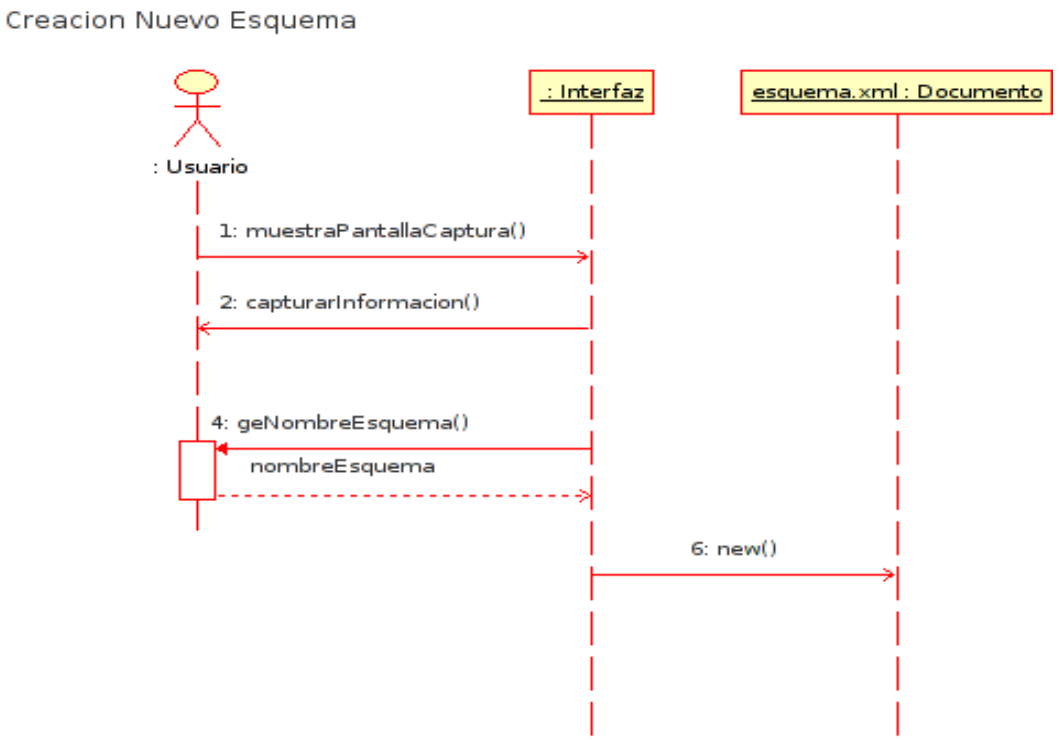


Figura 2: Diagrama de secuencia, caso de creación de nuevo esquema

3.8.2 Caso 2: Abrir documento existente.

En este caso, el sistema deberá mostrar los documentos XML presentes en el directorio actual, el usuario carga el documento y el mismo es presentado en pantalla.

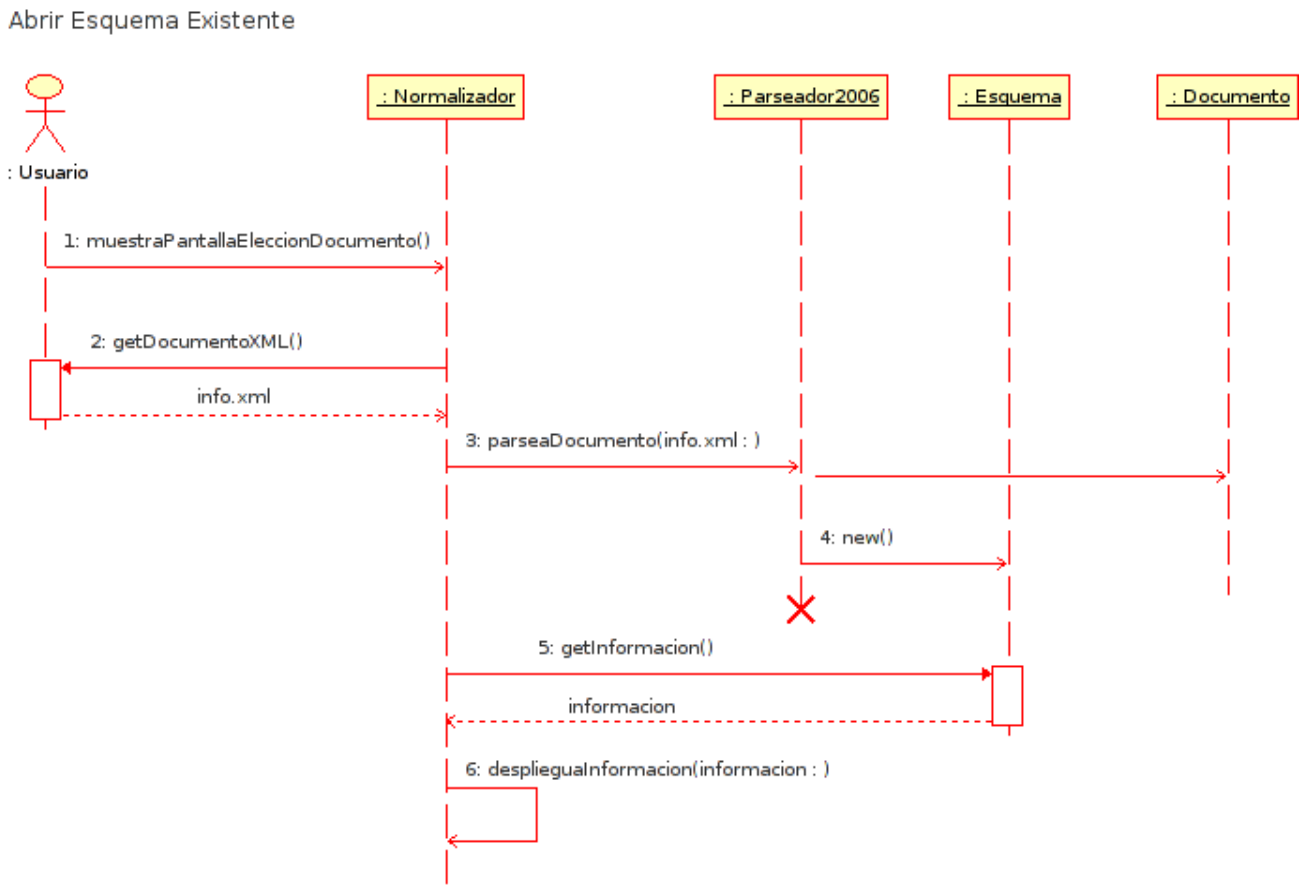


Figura 3: Diagrama de secuencia, cargar al sistema un documento existente

3.9 Funcionamiento con documento cargado en el sistema.

Una vez que el esquema se carga en pantalla el usuario deberá elegir cual de las seis opciones presentadas dentro de los casos de uso será ejecutada.

Restricciones de las opciones.

Por la teoría de la normalización, se analizó que si un esquema incluye solamente dependencias funcionales, no tiene caso sugerirle al usuario que dicho esquema puede ser normalizado a Cuarta Forma Normal.

Con un razonamiento análogo, si el esquema incluye alguna dependencia multivaluada, no se presentaran Formas Normales distintas a la Cuarta Forma Normal.

Sin importar lo anterior, los casos de Cerradura Transitiva, llaves y

Cobertura Canónica siempre se podrán elegir.
Se presenta el siguiente diagrama para exponer la bifurcación que se presenta al elegir un documento.

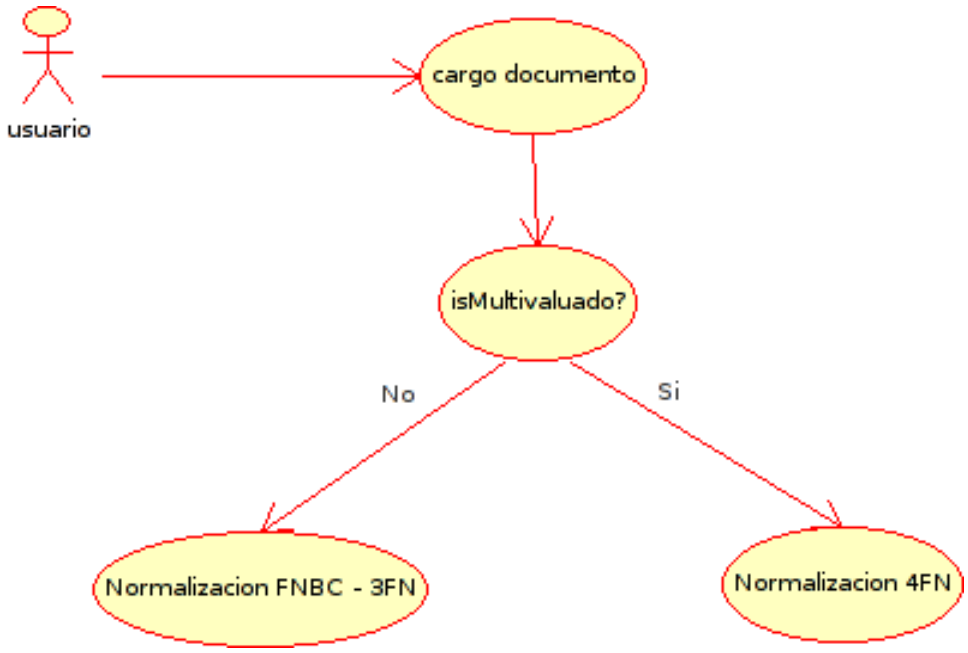


Figura 4: Diagrama de secuencia, decisión de normalizaciones a mostrar

3.10 Análisis de Funcionalidades del Sistema

El sistema presenta las seis funcionalidades presentadas en la figura 1.
A continuación se analiza el proceso de cada una de las 6 funcionalidades.

3.10.1 Cerradura Transitiva

Como base para la funcionalidad del sistema fue preciso incluir, aun sin ser mostrado en pantalla, los axiomas de Armstrong, dichos axiomas son la base de las cerraduras transitivas de conjuntos de atributos, el proceso es el mismo que se definió en el capítulo de teoría.

3.10.2 Llaves.

En esta opción, el sistema genera subconjuntos X_R de todos los atributos **A** presentes en la relación **R**, posteriormente, a cada subconjunto X_R se le aplica la cerradura transitiva Cerr(X_R). Si el conjunto de atributos Cerr(X_R) es el mismo que el conjunto de atributos **A,** entonces X_R es una llave.

Este proceso se explica en el siguiente pseudocódigo:

```
 Sea llaves el conjunto de llaves de la relación R.
 Sea A el conjunto de atributos de R.
  llaves := A;
  subconjuntos := generaSubconjuntos( A );
/* En la variable subconjuntos tenemos el conjunto potencia de A*/
        foreach  ( subconjunto in subconjuntos )  do
          begin
             cerradura:= cerraduraTransitiva( subconjunto );
              if ( cerradura equals  A ) then
                  llaves := llaves ∪ subconjunto;
          end;
       end;
```

Proceso de obtención de llaves.

Por otra parte, el sistema también muestra las llaves mínimas (o superllaves) de la relación, este proceso se muestra a continuación.

```
Sea llaves el conjunto de llaves de la relación R.
Sea llaves_minimas  el conjunto de llaves mínimas de la relación
```

```
R.
 numero_atributos_llave_minima := numero_atributos( A );
 llaves_minimas := A;
  foreach  ( llave_actual in llaves ) do
  begin
   numero_atributos_actual:=  numero_atributos(  llave_actual);
    if ( numero_atributos_llave_actual  <
       numero_atributos_minima  ) then
      borrar_llaves_minimas() ;
      llaves_minimas := llave_actual;
    else if  (numero_atributos_actual equals
numero_atributos_llave_minima  ) then
        llaves_minimas := llaves_minimas  ∪ llave_actual;
   end;
 end;
```

Proceso obtención de llaves mínimas

3.10.3 Cobertura canónica.

El sistema presenta la relación en su cobertura canónica, (equivalencia mínima), este proceso usa el algoritmo mostrado en la sección de teoría.

3.11 Normalizaciones

Si bien los algoritmos descritos en la sección de teoría son los

usados por el sistema, esta sección tratará del contexto en el que los mismos son invocados por el sistema.

3.11.1 Normalización a Forma Normal Boyce-Codd

El proceso consiste en recorrer cada dependencia funcional del esquema, y ver si cumple la forma normal Boyce-Codd.
En caso de que alguna dependencia no cumpla la forma, se procede a la normalización tomando dicha dependencia como pivote para las particiones. Por el contrario, si cada dependencia dentro del esquema cumple la forma normal Boyce-Codd, se dice que la relación esta en Forma Normal Boyce-Codd.
El proceso se muestra en el siguiente recuadro:

```
Sea deps el conjunto de dependencias funcionales de la relación R.

foreach ( dep_actual in deps)do
     if ( dep_actual not cumpleFNBC ) then
          NormalizacionBCNF( dep_actual, R) ;
      else continue;
end;
en_Forma_Normal_Boyce-Codd := true;
```

Procedimiento de comprobación de esquema en forma normal Boyce-Codd.

3.11.2 Normalización a Tercera Forma Normal

El proceso consiste en recorrer cada dependencia funcional del esquema, y ver si cumple las condiciones de la Tercera Forma Normal presentadas previamente. En caso de que alguna dependencia no cumpla

la forma, se procede a la normalización generando la cobertura canónica del esquema actual y pasando a normalizar por el algoritmo especificado en la sección de teoría. Por el contrario, si cada dependencia dentro del esquema cumple la Tercera Forma Normal, se dice que la relación esta en Tercera Forma Normal.
El proceso se muestra en el siguiente recuadro:

```
Sea deps el conjunto de dependencias funcionales de la relación R.
 foreach ( dep_actual in deps) do
     if ( dep_actual not cumple3FN ) then
          minimal := cobertura_canonica( deps );
          NormalizacionTerceraFormaNormal( minimal, R);
    else en_Tercera_Forma_Normal := true;
end;
```

Procedimiento de comprobación de esquema en tercera forma normal.

3.11.3 Normalización a Cuarta Forma Normal

El proceso que se sigue es bastante parecido al correspondiente a la Forma Normal Boyce-Codd.
El proceso se muestra en el siguiente recuadro:

```
Sea deps el conjunto de dependencias multivaluadas de la relación R.
 foreach ( dep_actual in deps) do
     if ( dep_actual not cumple4FN ) then
          Normalizacion4FN( dep_actual, R) ;
```

```
      else continue;
end;
en_Cuarta_Forma_Normal := true;
```

Proceso de comprobación de esquema en cuarta forma normal.

3.12 Conclusiones.

En este capítulo se presentaron los algoritmos que forman el núcleo del Sistema Normalizador.

Si bien dichos algoritmos podían ser hechos de otra forma siguiendo bibliografías distintas, únicamente se exponen las versiones de los algoritmos que fueron codificados para la herramienta.

Este capítulo también trató de las opciones prácticas para desarrollar la herramienta, así como las justificaciones de cada elección realizada con el fin de que el cuerpo estudiantil que siga el curso de Bases de Datos pudiera proponer y realizar modificaciones.

4. Implementación

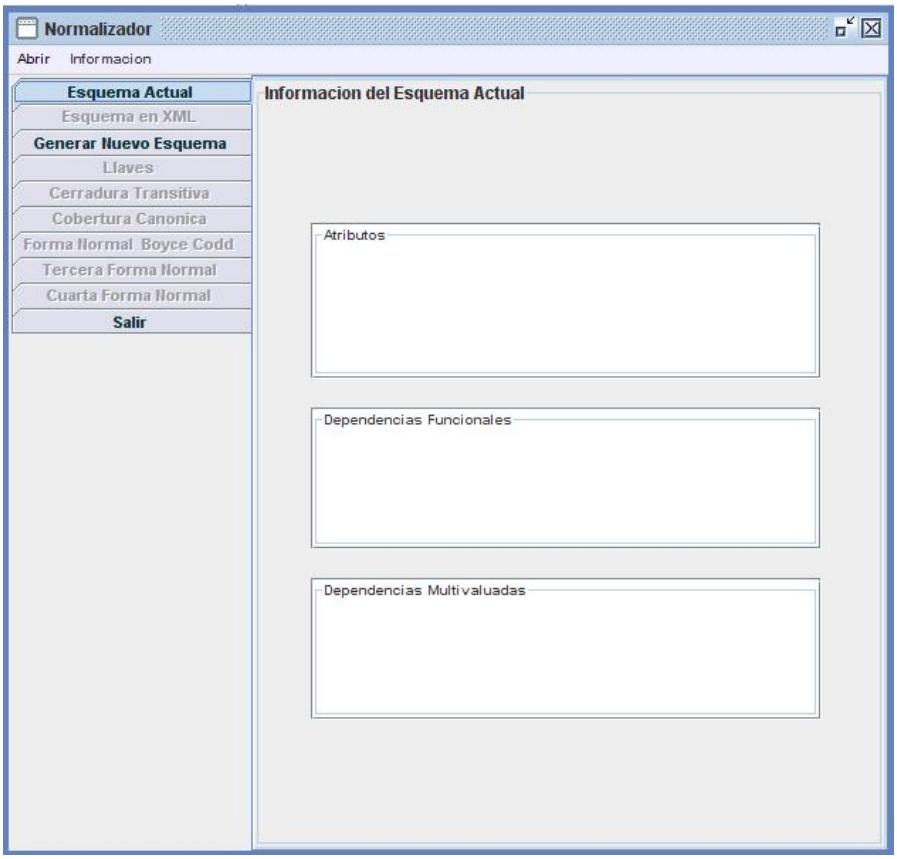
Una vez definido que procesos tenía que hacer el sistema, se procedió a desarrollar cómo se iban a efectuar dichos procesos.
Este capítulo trata acerca de los pasos llevados para la implementación del Sistema Normalizador.

4.1 Elección de la arquitectura de la aplicación.

Actualmente la herramienta posee una arquitectura cliente-servidor, es decir, consta de una interfaz que consulta un conjunto de clases que le sirven información dentro de la funcionalidad necesaria.
La herramienta fue planeada para que cada alumno del curso de Bases de Datos, el usuario final del sistema, genere sus propios ejemplos y los guarde en su máquina.
Esto sin necesidad de que el sistema tenga que procesar dichos ejemplos en algún servidor que administre una base de datos de ejemplos del curso, simplemente, el alumno podrá crear su esquema, y posteriormente procesarlo a través de la herramienta para ver la salida de distintos algoritmos.

4.2 Interfaz de usuario.

Se decidió generar una interfaz de usuario para la herramienta educativa, de tal forma que fuera mas sencillo para el usuario hacer uso de la funcionalidad definida para el sistema.



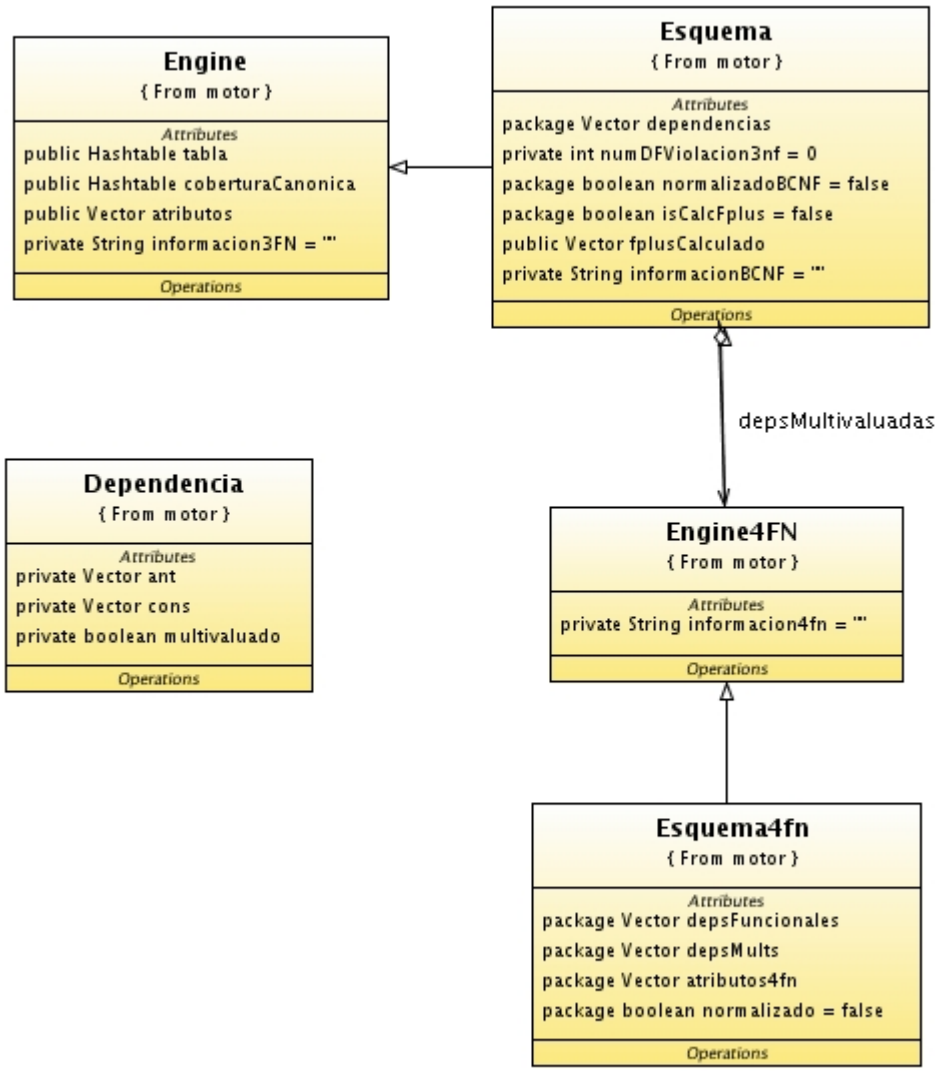
Captura de la ventana inicial del Sistema Normalizador.

4.3 Paquetes

La funcionalidad del sistema se agrupó en 5 paquetes que tienen funciones especificadas a continuación.

4.3.1 Paquete motor

Paquete que agrupa tanto los algoritmos inherentes a la normalización de los esquemas, como a la representación de los mismos en objetos de Java. Contiene cinco clases: Dependencia,Engine, Engine4FN, Esquema y Esquema4FN.

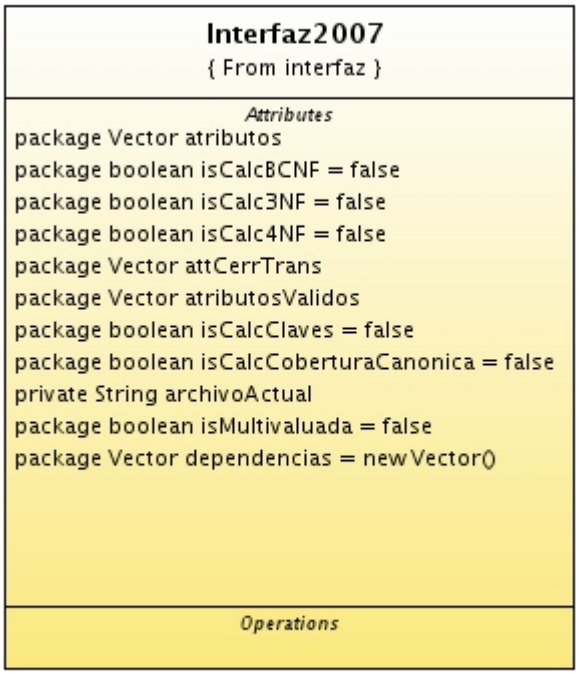


Clases del paquete motor.

4.3.2 Paquete interfaz

Paquete encargado de la interacción del sistema y el usuario.

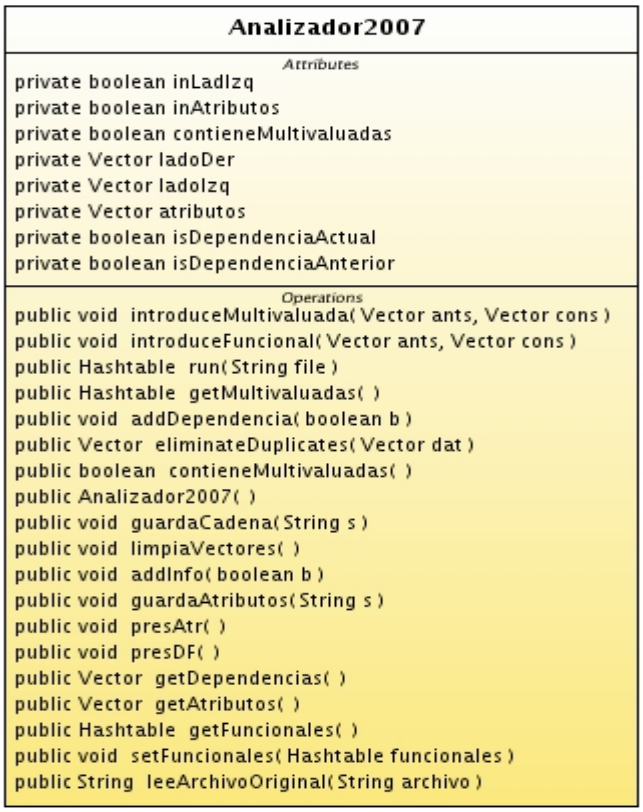
Presenta una única clase Interfaz.



Clase del paquete interfaz.

4.3.3 Paquete analizador

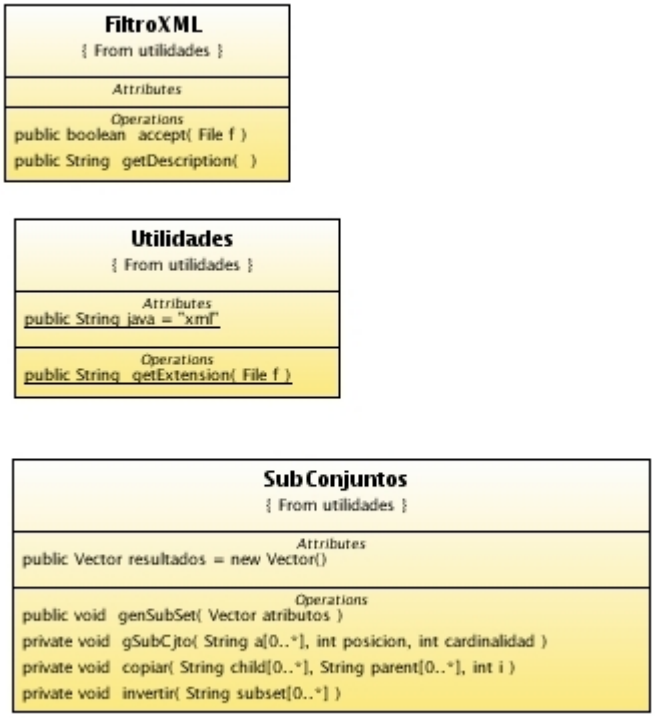
Paquete que contiene el analizador sintáctico del sistema, el cual hereda de Xerces. La clase es llamada Analizador2007.



Clase del paquete analizador.

4.3.4 Paquete utilidades

Paquete que contiene tres clases tanto para la parte gráfica como para los algoritmos. Sus clases son Utils,SubSet e ImageFilter.



Clases del paquete utilidades.

4.3.5 Paquete pruebas

Paquete sin utilidad en la versión final del sistema, pero contiene los ejemplos con los que se fue haciendo la depuración de los algoritmos. Lo mantuve en la distribución para utilidad de futuros desarrollos. Este conjunto cubre cada algoritmo presente en el sistema, aunque corren en modo consola, puesto que la interfaz fue la fase final del desarrollo.

4.4 Bibliotecas auxiliares.

Para implementar la funcionalidad del sistema, la herramienta hace uso de tres bibliotecas:

AbsoluteLayout.jar: Biblioteca para la interfaz gráfica.

swing-layout-1.0.1.jar: Biblioteca para la interfaz gráfica.

xercesImpl.jar : Biblioteca para analizar documentos XML.

4.5 Manual de Usuario

4.5.1 Pantalla principal

En la figura 1 se muestra la primera pantalla del sistema, la cual aparece al ejecutar la aplicación.

Desde esta pantalla se decide que opción ejecutar.

Se ve que en primera instancia, las opciones de Llaves, Cerradura Transitiva, Cobertura Canónica y las 3 Formas Normales no están activadas, esto es debido a que no hay ningún esquema relacional cargado previamente por la aplicación.

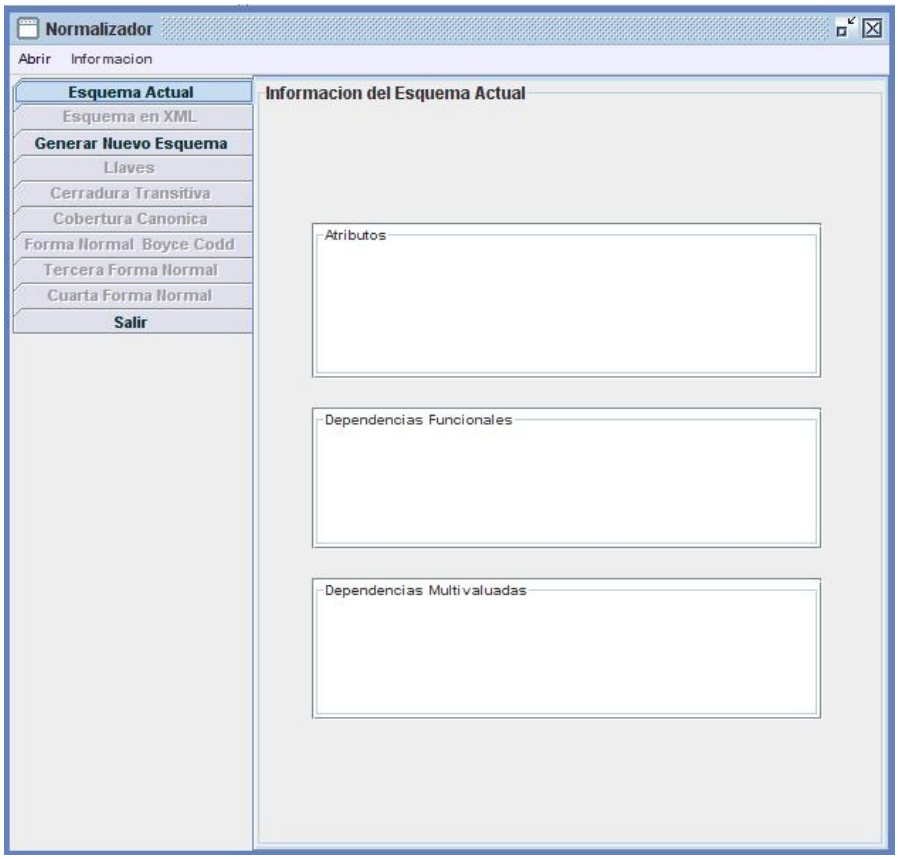


Figura 1. Pantalla principal.

4.5.2 Creación de un nuevo esquema.

Si se decide generar un nuevo esquema se desplegará la siguiente pantalla, donde se introducirá en cada campo la información requerida para la creación de un esquema relacional.

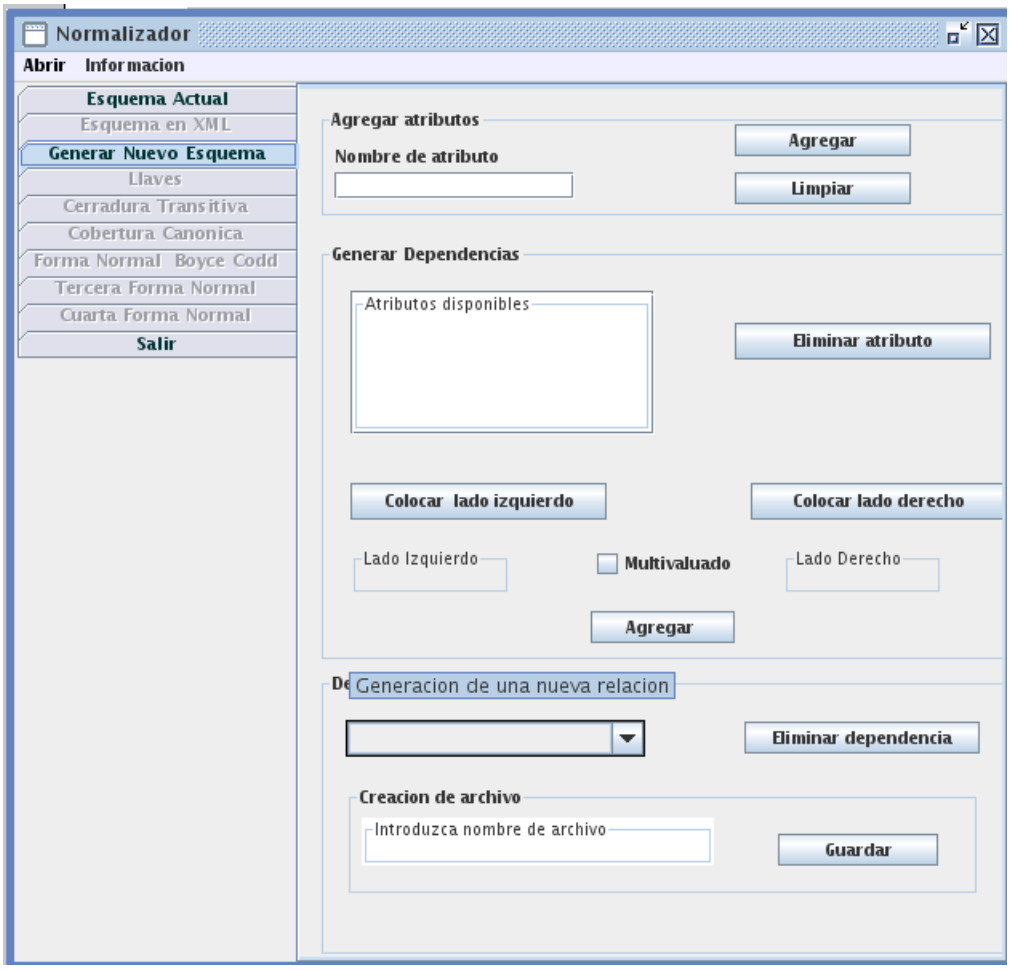


Figura 2. Pantalla de generación de nuevo esquema.

4.5.2.1 Introducir atributos

Dentro del campo Nombre de atributo se pueden agregar uno a uno los atributos de la relación, o introducirlos todos separados por el carácter ',', como se muestra en la figura.

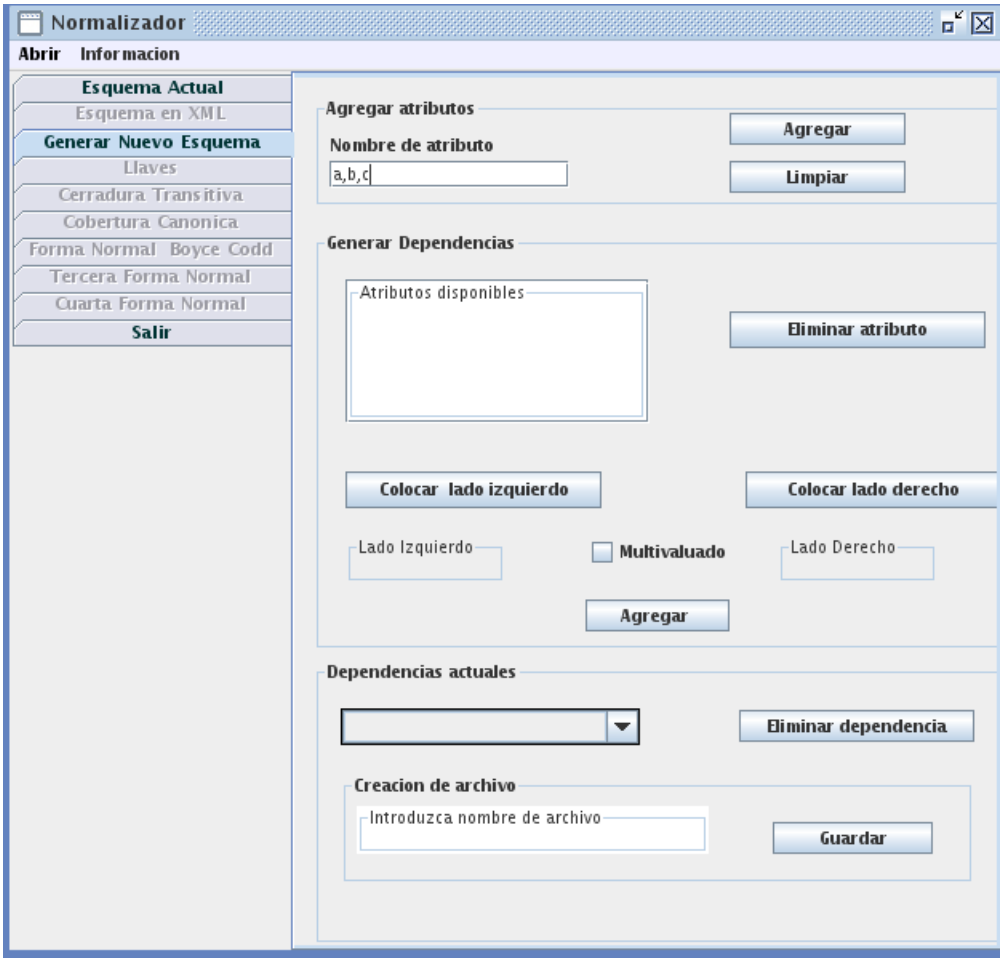


Figura 3. Agregando los atributos del esquema.

4.5.2.2 Generar dependencias

Dentro de la zona señalada en la figura, se pueden observar los atributos que se han definido previamente para ser parte del esquema. La creación de dependencias supone que el usuario elegirá un atributo, posteriormente elegirá si dicho atributo pertenece al lado izquierdo o derecho de la dependencia a crear, siguiendo este proceso hasta que la dependencia a crear tenga los atributos deseados. Se prosigue con la definición de la dependencia como multivaluada o funcional. Una vez definida la dependencia, se pulsa agregar y la dependencia pasará a mostrar dentro del despliegue de las dependencias actuales.

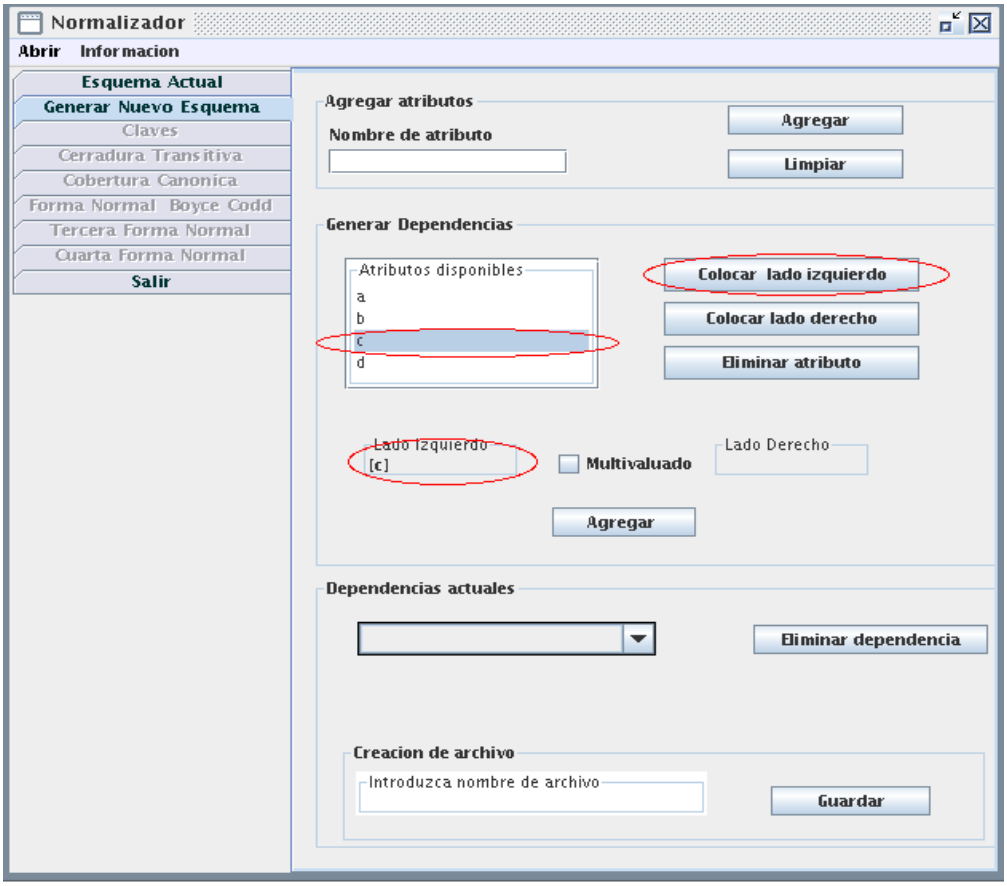


Figura 4. Creando dependencias del esquema.

4.5.2.3 Agregando una dependencia funcional

En la figura se muestra como el usuario elige el atributo b, pulsa el botón colocar debajo del mensaje de Lado Derecho y se muestra la dependencia funcional actual. Una vez que el usuario pulsa el botón agregar, la dependencia recién creada se agrega a la lista que se resalta en recuadro rojo. En esta parte el usuario puede ver las dependencias actuales y eliminar alguna si así lo desea.

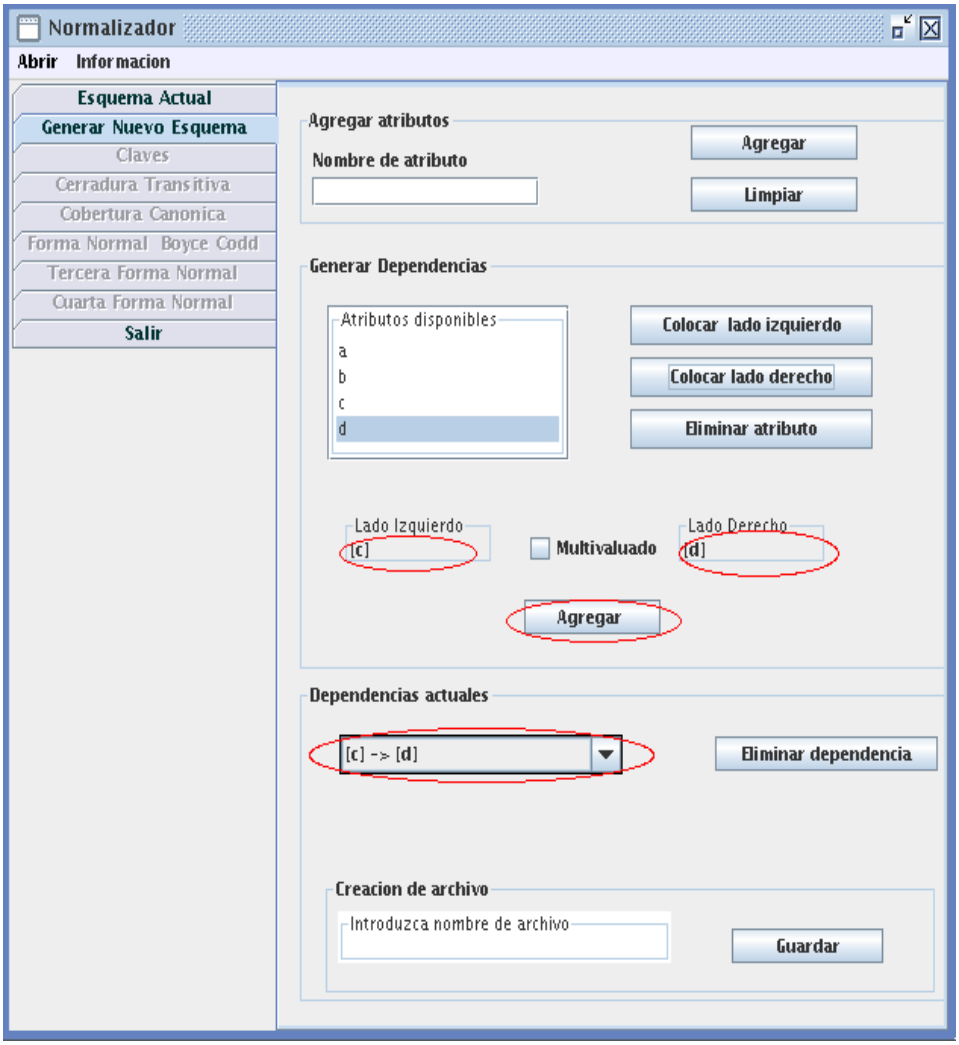


Figura 5. Agregando dependencias del esquema.

4.5.2.4 Creación de archivo

Para terminar la creación de un documento xml con la información perteneciente al esquema creado, el usuario deberá asignarle un nombre, es innecesario que el usuario introduzca la terminación .xml , puesto que el sistema lo hace de cualquier manera.

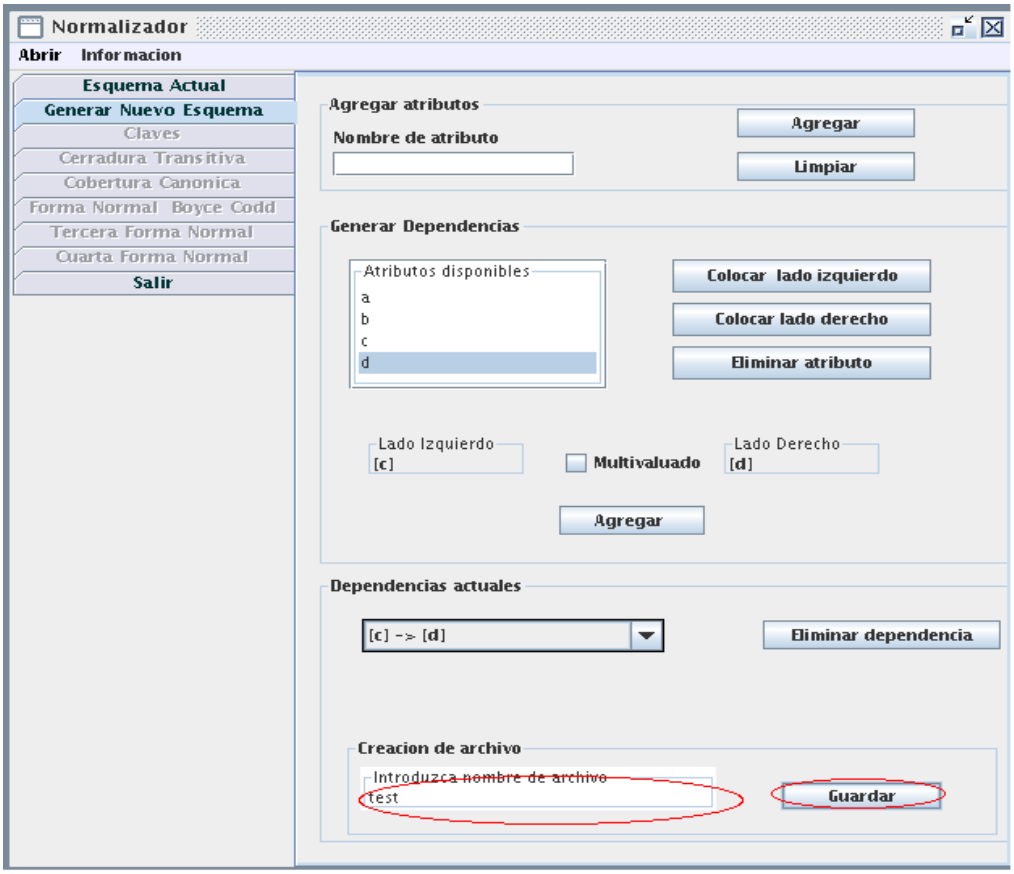


Figura 6.Creando el archivo.

Una vez que se ha creado el archivo, el sistema muestra el éxito de la operación.

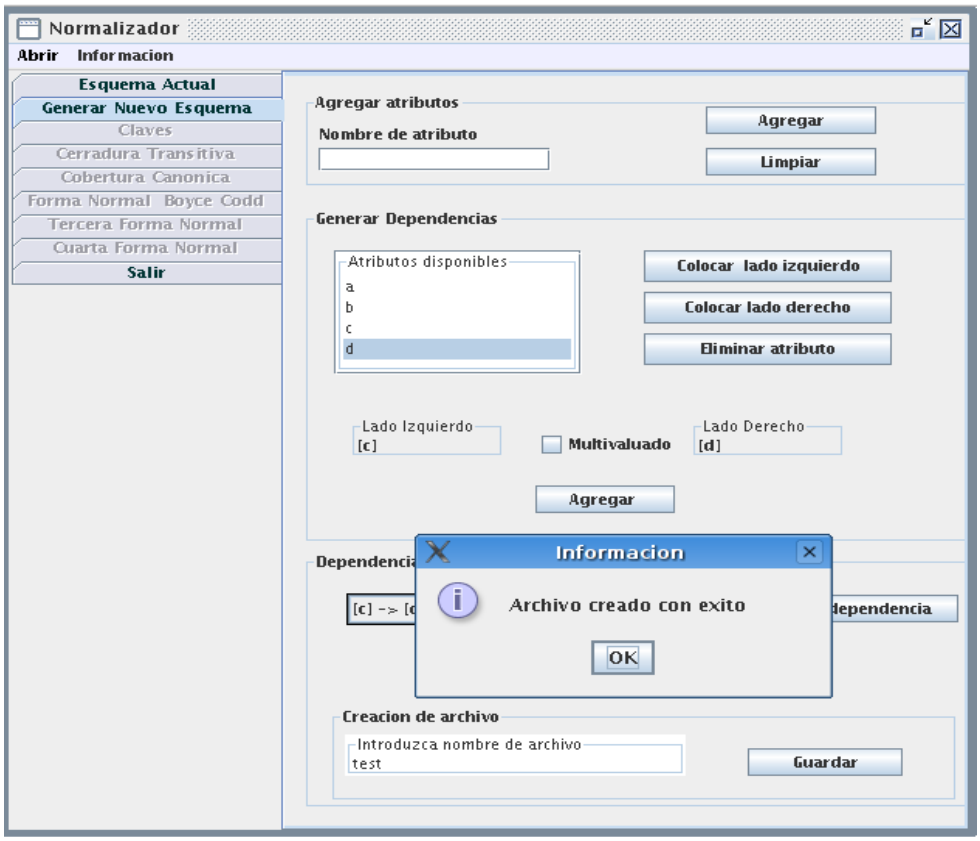


Figura 7. Archivo generado.

4.5.3 Carga de un esquema existente.

Se selecciona la opción abrir archivo del menú

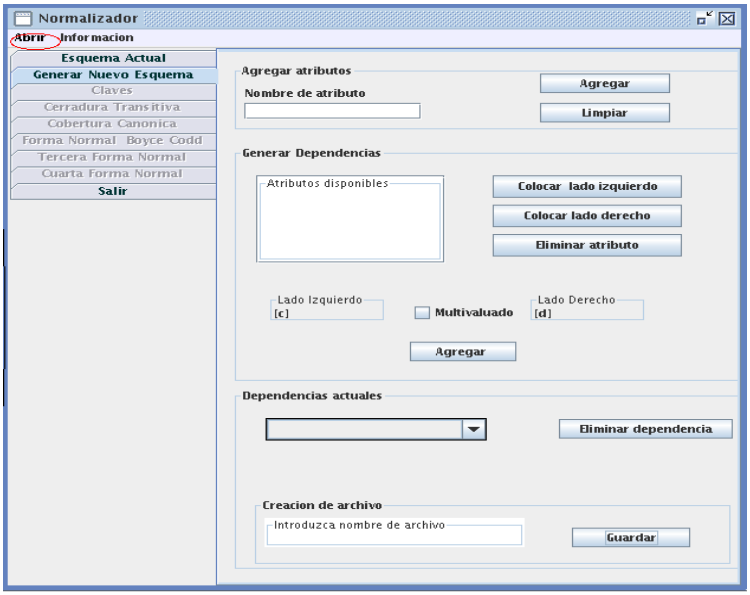


Figura 8.Usando el menú.

4.5.3.1 Seleccionar archivo

A continuación se desplegarán los documentos xml en el directorio de ejecución de la aplicación.

Una vez seleccionado el archivo, se pulsará el botón Open.

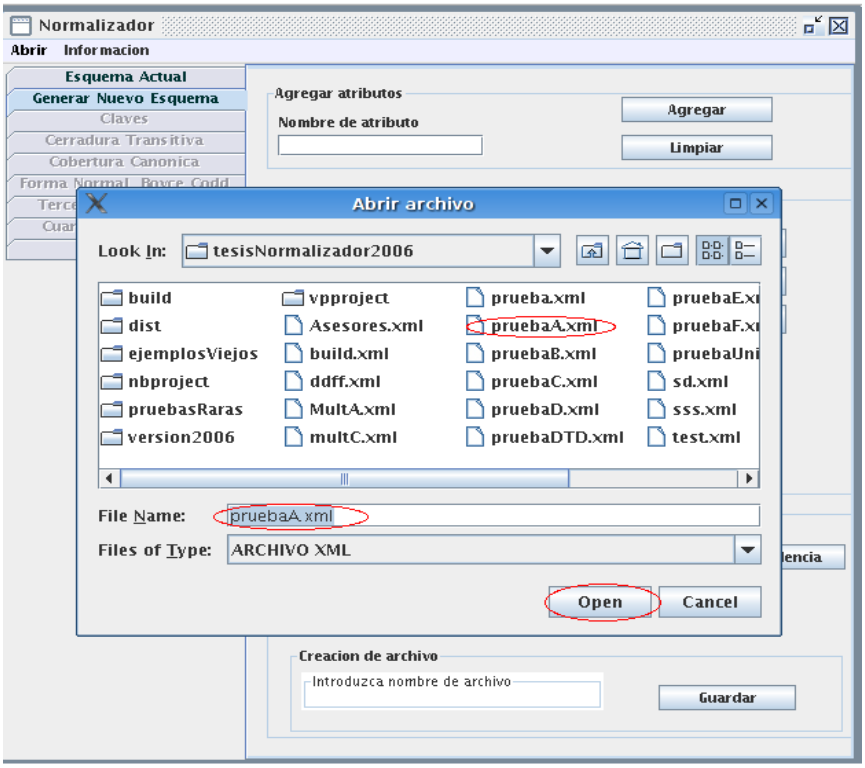


Figura 9.Seleccionando el archivo XML.

4.5.3.2 Mostrar información del archivo en pantalla

El sistema mostrará la información contenida en el documento elegido. Se puede observar que la opción de Cuarta Forma Normal esta desactivada debido a que el esquema actual no presenta dependencias multivaluadas.

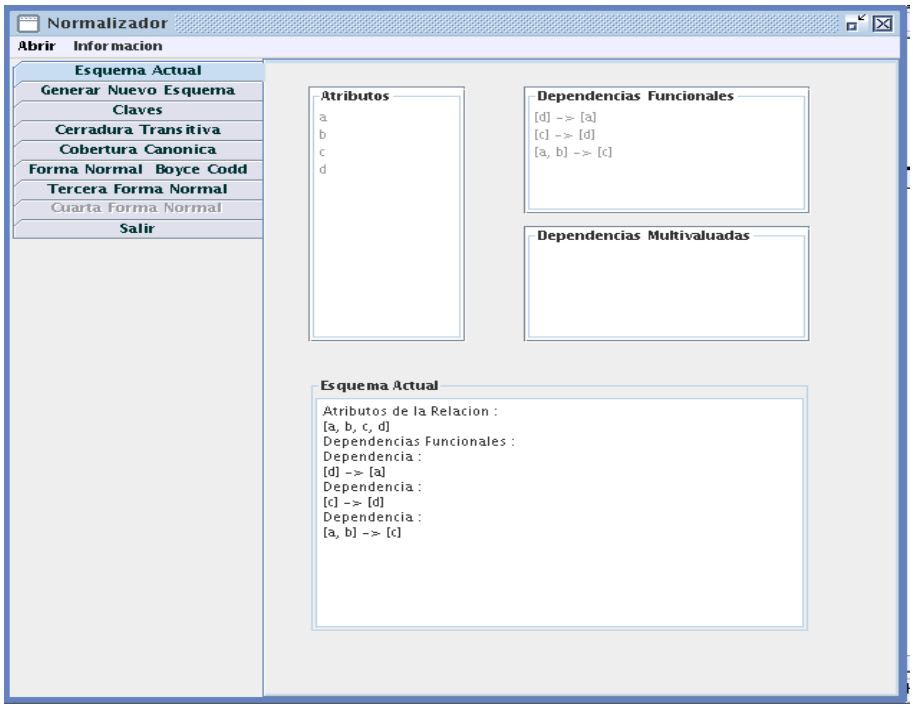


Figura 10. Normalización de dependencias multivaluadas desactivada.

4.5.3.3 Abriendo esquema con dependencias multivaluadas

En este caso se puede ver como las opciones de Forma Normal Boyce-Codd y Tercera Forma Normal se encuentran deshabilitados.

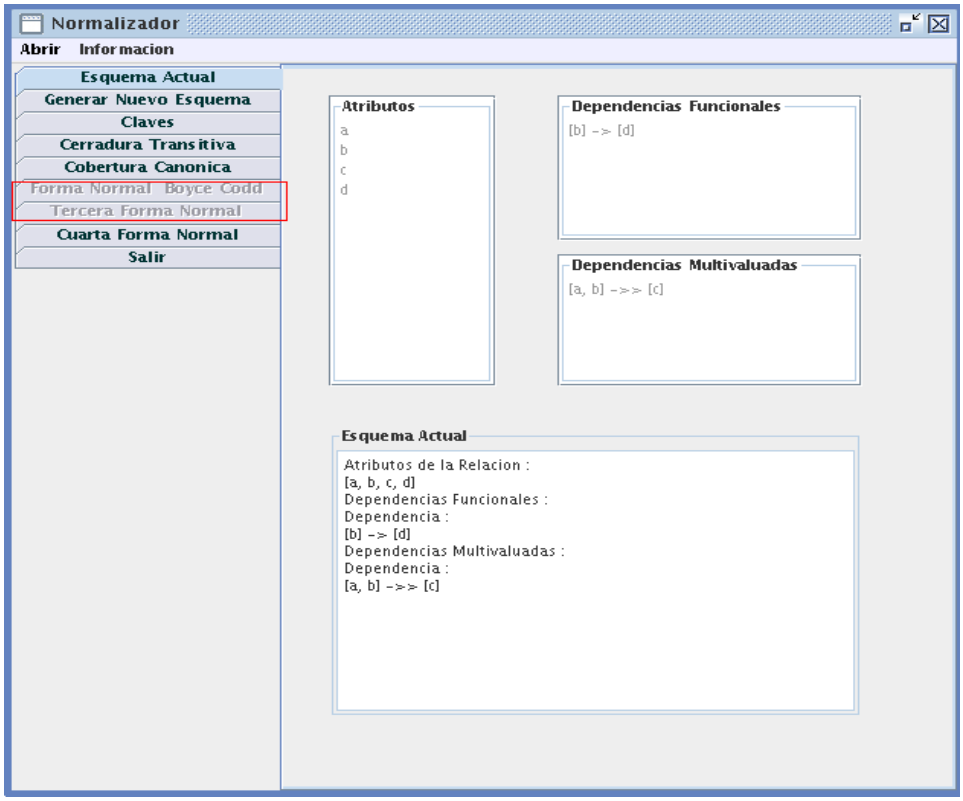


Figura 11. Normalización de dependencias multivaluadas activada.

4.5.4 Selección de llaves del esquema actual.

Con esta opción el sistema encuentra las llaves candidatas del esquema actual, así como encontrará las llaves mínimas del mismo esquema.

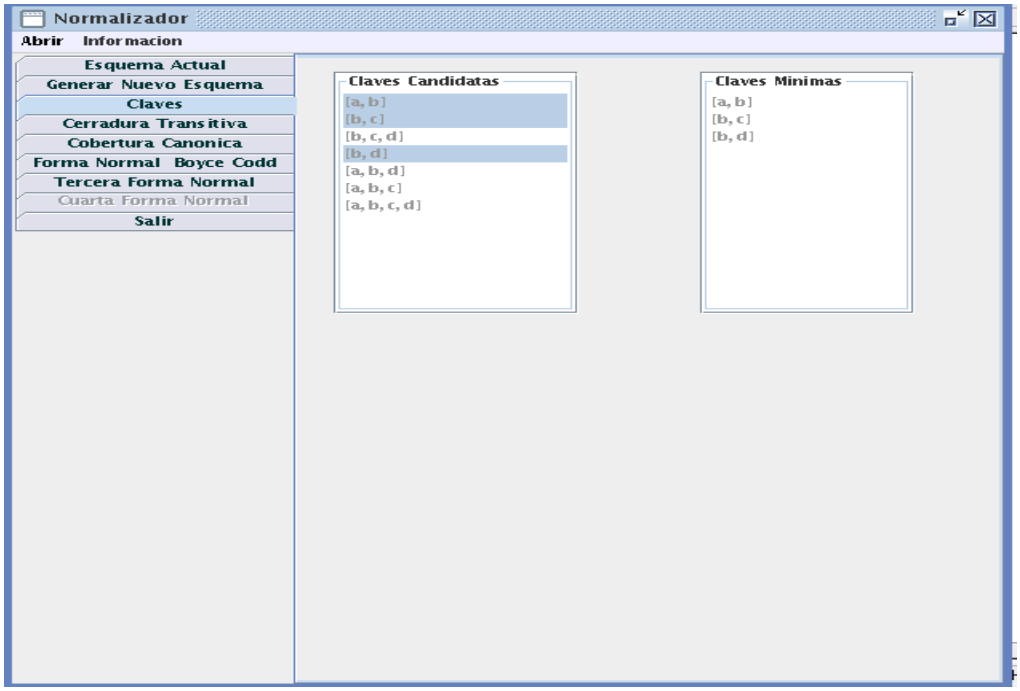


Figura 12. Obtención de llaves por el Sistema Normalizador.

4.5.5 Selección de cerradura transitiva

Al seleccionar la opción de cerradura transitiva, el usuario deberá introducir aquellos atributos a los que se les calculara la cerradura teniendo en cuenta las dependencias del esquema.

4.5.5.1 Introduciendo los atributos a calcular

En primera instancia, se piden los atributos. Igual que en la creación de un esquema, los atributos se pueden agregar uno por uno o en conjunto, separados por el carácter ','.

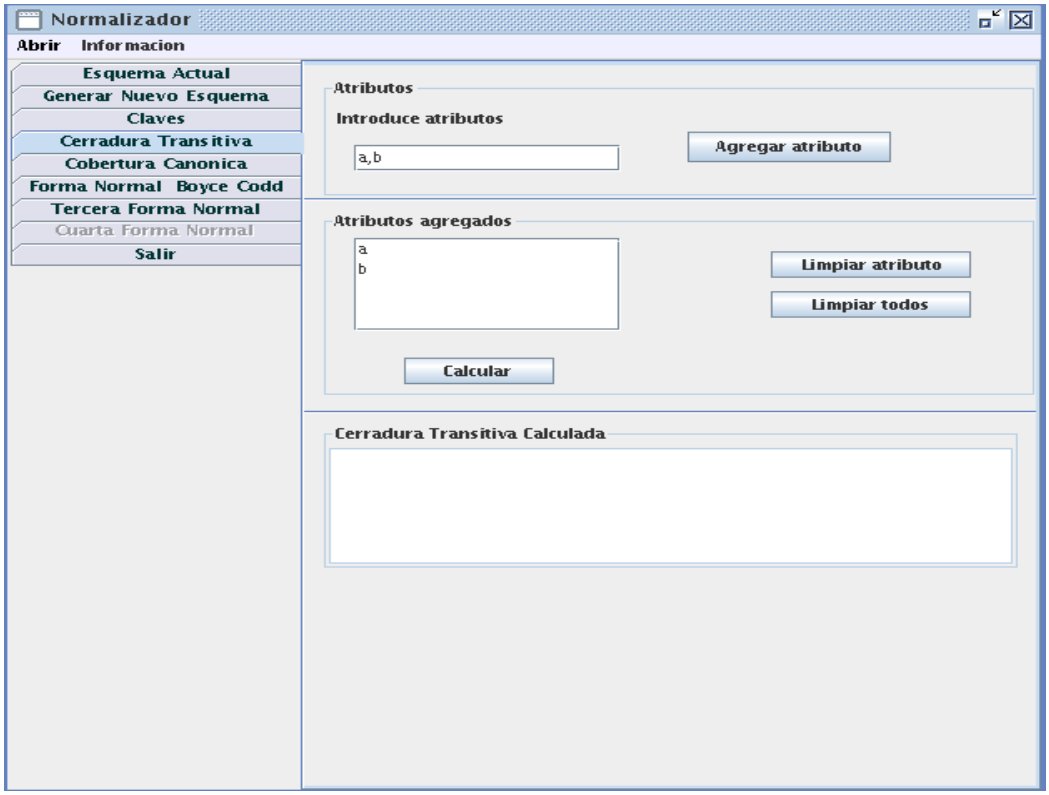


Figura 13. Cálculo de cerradura transitiva por el Sistema Normalizador.

4.5.5.2. Atributos agregados

Posteriormente, el sistema presenta la lista de atributos a calcular. En esta sección de la pantalla también podrá limpiar todos los atributos de la lista o solo el actualmente seleccionado.

4.5.5.3 Presionar botón de cerradura

Al presionar el botón Calcular Cerradura Transitiva el sistema se encargara de desplegar el resultado en la parte inferior de la pantalla del sistema.

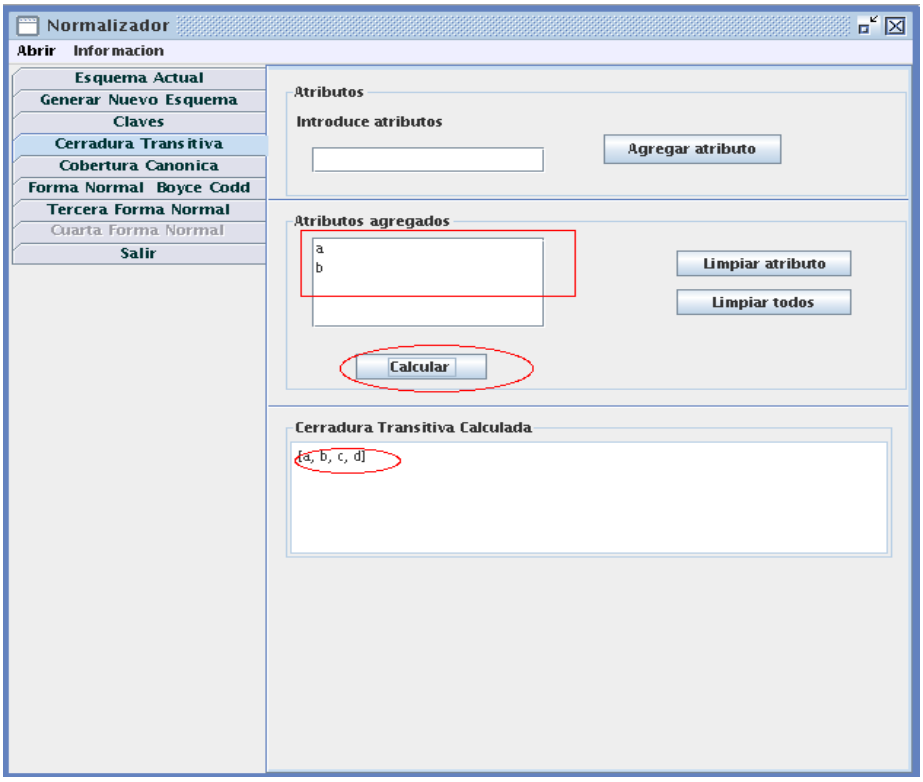


Figura 14. Cálculo de cerradura transitiva por el Sistema Normalizador.

4.5.6 Selección de cobertura canónica

En caso de que el usuario seleccione la opción Cobertura Canónica , el sistema mostrara las dependencias pertenecientes a la cobertura canónica calculada.

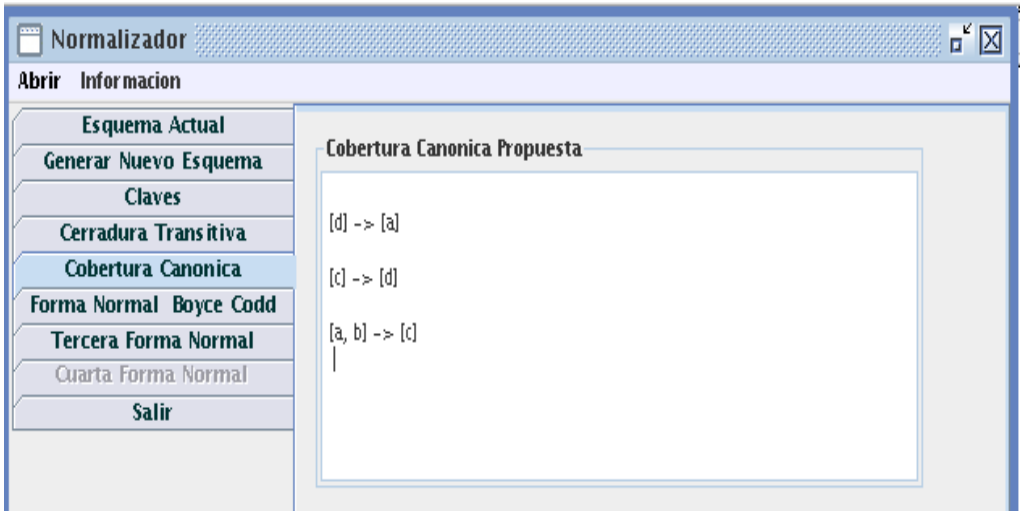


Figura 15. Cálculo de cobertura canónica por el Sistema Normalizador.

4.5.7 Selección de Forma Normal Boyce-Codd

En caso de que el usuario seleccione la opción Forma Normal Boyce Codd , el sistema calculara si el esquema esta en esta forma normal, en caso afirmativo, mostrara un mensaje informando al usuario que el esquema actual se encuentra en dicha forma, en caso contrario, la herramienta mostrara la información relevante al proceso de normalización a Forma Normal Boyce Codd

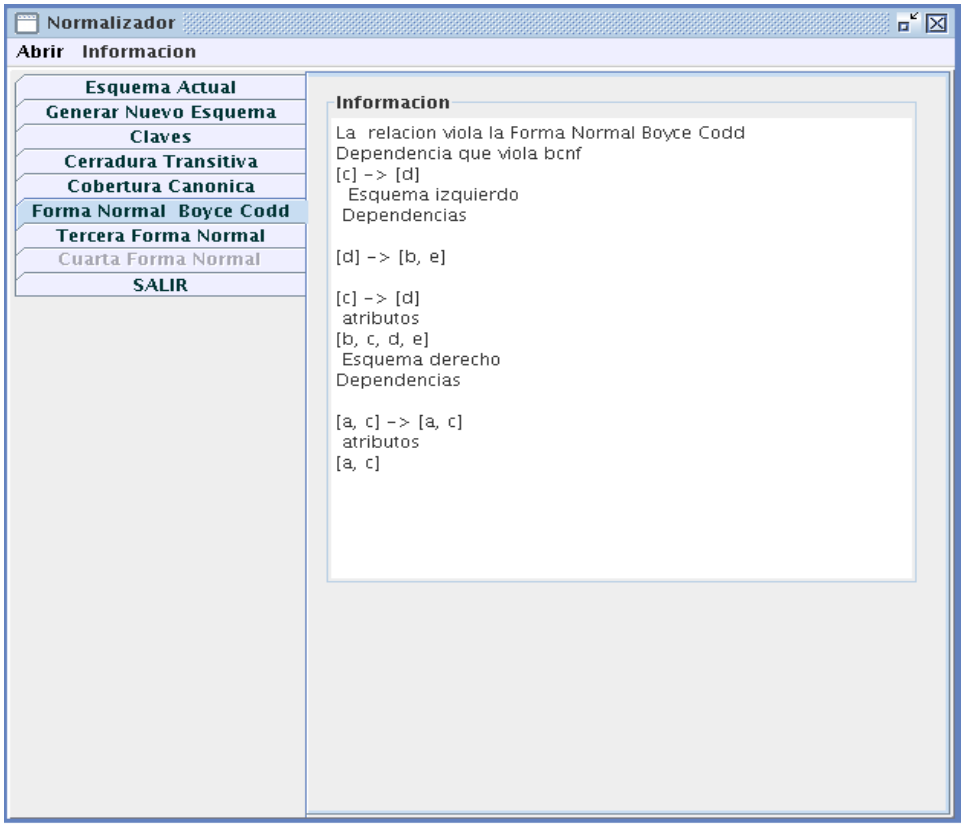


Figura 16. Resumen del proceso de normalización elegido.

4.5.8 Selección de Tercera Forma Normal

En caso de que el usuario seleccione la opción Tercera Forma Normal , el sistema calculará si el esquema esta en esta forma normal, en caso afirmativo, mostrara un mensaje informando al usuario que el esquema actual se encuentra en dicha forma, en caso contrario, la herramienta mostrara la información relevante al proceso de normalización a Tercera Forma Normal.

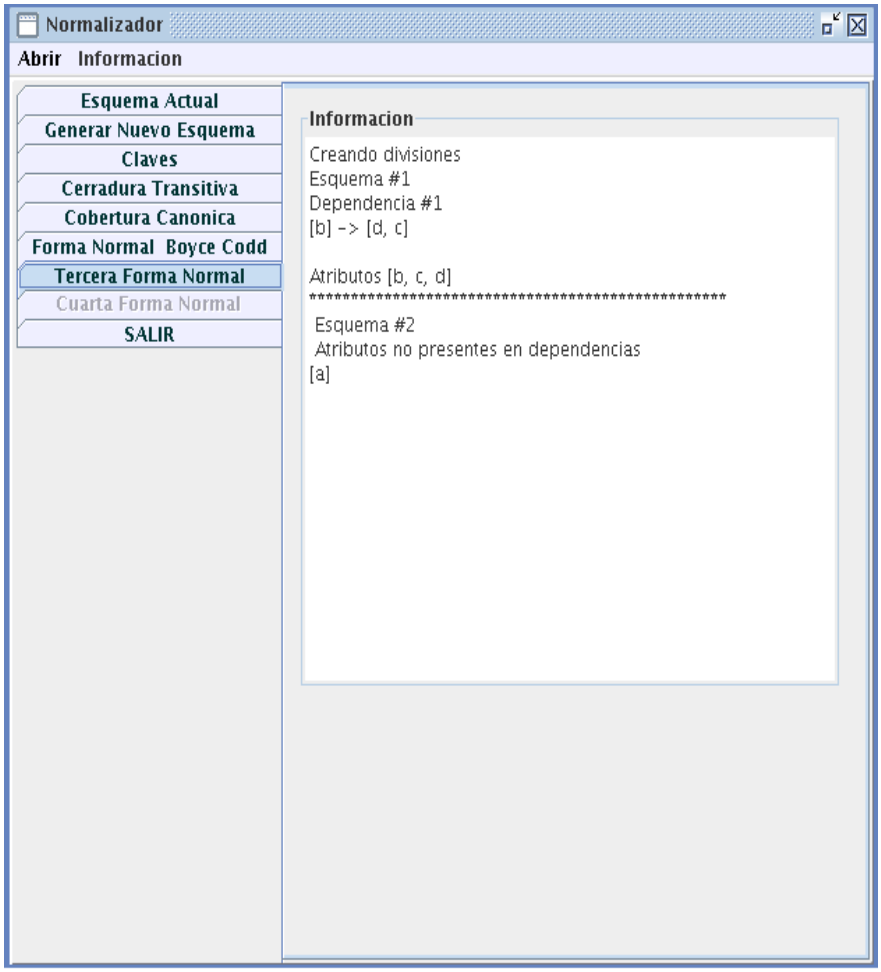


Figura 17. Resumen del proceso de normalización elegido.

4.5.9 Selección de Cuarta Forma Normal

En caso de que el usuario seleccione la opción Cuarta Forma Normal , el sistema calculara si el esquema esta en esta forma normal, en caso afirmativo, mostrara un mensaje informando al usuario que el esquema actual se encuentra en dicha forma, en caso contrario, la herramienta mostrará la información relevante al proceso de normalización a Cuarta Forma Normal.

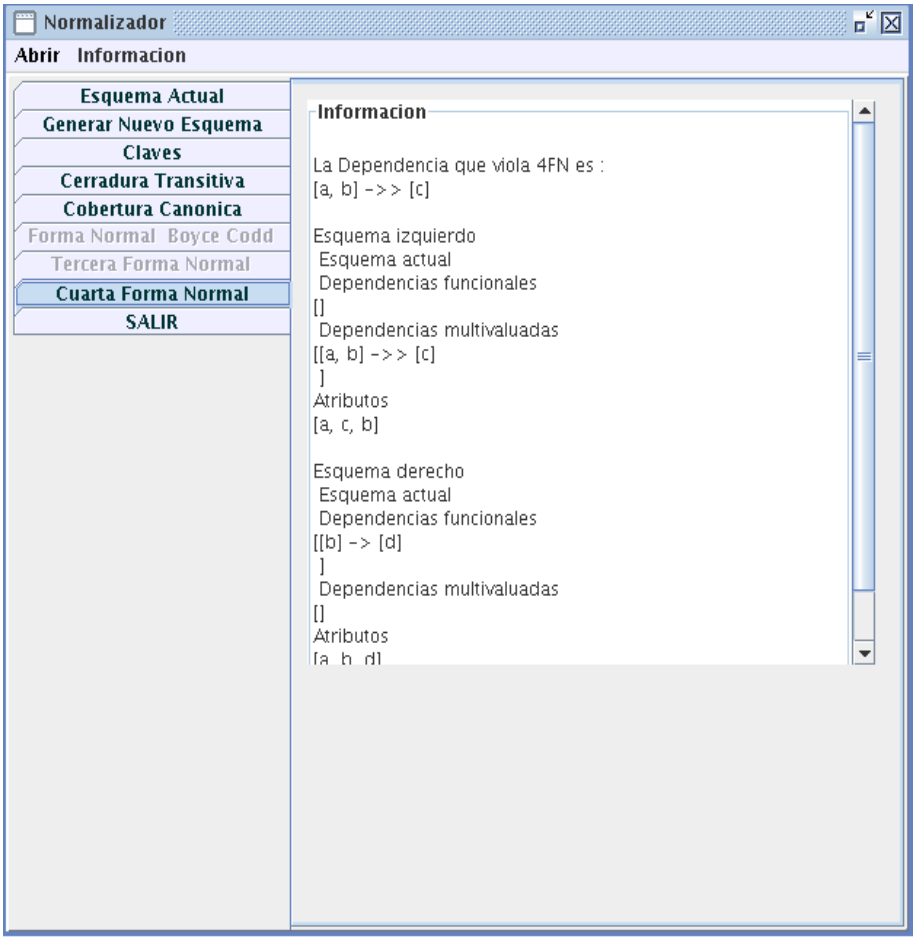


Figura 18. Resumen del proceso de normalización elegido.

4.6 Requerimientos del sistema normalizador.

Los algoritmos del sistema normalizador corren desde la versión 1.3 en adelante de Java Standard Edition (J2SE), sin embargo, la interfaz requiere ser usada a partir de la versión 1.4 en adelante.

En base a esto, si al usuario no le convence la interfaz, podrá generar una desde la versión 1.3 de Java y el núcleo de algoritmos correrá bien.

4.7 Distribución del Sistema Normalizador.

El sistema se distribuye como un archivo JAR, donde se encuentran presentes todas las bibliotecas necesarias, así mismo, se incluyen un par de scripts de uso tanto para el sistema operativo Windows como para sistemas operativos basados en Linux.

4.8 Conclusiones.

Durante este capítulo se ha mostrado la distribución de paquetes del Sistema Normalizador, así como cuestiones técnicas del ambiente sobre los que puede ser ejecutado.

Así mismo, en este capítulo se incluye el manual del usuario (sección 4.5 en adelante), donde se muestra mediante capturas de pantalla el uso de esta herramienta.

Conclusiones Generales.

1. Se ha construido una herramienta normalizadora de relaciones partiendo de algoritmos y definiciones vistas en el curso introductorio de bases de datos impartido en la Facultad de Ciencias de la UNAM.
2. Para ello se han identificado los principales casos de uso para la sección de normalización del modelo relacional del curso introductorio de bases de datos impartido en la Facultad de Ciencias de la UNAM.
3. Se ha creado un lenguaje de marcado en el metalenguaje XML que definen la información necesaria para describir una relación en el modelo relacional.
4. Se ha generado un DTD para validar los documentos que deseen mostrarse en el Sistema Normalizador.
5. Se han desarrollado algoritmos alternos para codificar de manera más simple algoritmos de la teoría relacional de bases de datos.
6. La herramienta mencionada podrá ser usada en el curso como un asistente para comprobar y afirmar los conocimientos adquiridos por los alumnos dentro del tema de normalizaciones del modelo relacional.
7. El código fuente de la herramienta está puesto a disposición de comunidad para que pueda observarse la implementación de los algoritmos en un lenguaje orientado a objetos.

Trabajos Futuros

- Se podría generar una versión Web que permitiera el uso de la interfaz desde la Internet, ya sea por medio de JNLP o

reconstruyendo la vista con algún framework tipo Java Server Faces (JSF).

- Se podría extender la funcionalidad a diversas formas normales como la Primera Forma Normal (1FN), Segunda Forma Normal (2FN), etc.
- Se podría mejorar la experiencia educativa generando un módulo para ver animaciones de cómo se sigue paso a paso los algoritmos presentados.
- Se podría generar una base de conocimiento de los ejercicios presentados para ser usada como un tutor de alumnos llevando un registro del avance de los mismos.

El Manual de usuario podría incluir vídeos grabados de su uso.

Glosario

Analizador Sintáctico (Parser)**:** Es un módulo, biblioteca o programa que se ocupa de transformar un archivo de texto en una representación interna, es decir un analizador de sintaxis para una determinada gramática.

API (Aplication Programming Interface): Es un conjunto de especificaciones de comunicación entre componentes software para conseguir abstracción en la programación entre diversas capas de software

Cliente-Servidor: Es un modelo para el desarrollo de sistemas de información, en el que las transacciones se dividen en procesos independientes que cooperan entre sí para intercambiar información, servicios o recursos. Se denomina cliente al proceso que inicia el diálogo o solicita los recursos y servidor, al proceso que responde a las solicitudes.

DTD (Document Type Definition): La definición de tipo de documento (DTD) es una descripción de estructura y sintaxis de un documento SGML. Su función básica es la descripción del formato de datos, para usar un formato común y mantener la consistencia entre todos los documentos que utilicen la misma DTD.

Framework: En el desarrollo de software, un framework es una estructura de soporte definida en la cual otro proyecto de software puede ser organizado y desarrollado. Representa una arquitectura de software que modela las relaciones generales de las entidades del dominio. Provee

una estructura y una metodología de trabajo la cual extiende o utiliza las aplicaciones del dominio.

LISP (LIst Processing)**:** Es un lenguaje funcional de alto nivel que trata los datos y al programa mismo como listas.

JNLP (Java **N**etworking **L**aunching **P**rotocol)**:** Es una especificación usada por Java para tener centralizado en un servidor Web un programa, evitando así problemas de distribución e instalación.

JSF (Java Server Faces)**:** Es un framework de desarrollo basado en el patrón MVC (Modelo Vista Controlador) que pretende normalizar y estandarizar el desarrollo de aplicaciones Web.

JSP (Java Server Pages): Es una tecnología Java que permite generar contenido dinámico e introducirlo en el contenido estático del documento Web.

PROLOG: Es un lenguaje de programación enmarcado dentro del paradigma de lenguajes lógicos, debido a estar compuesto por instrucciones tipo cláusulas de Horn, y tener reglas estilo Modus Ponens.

SGBD (Sistema de gestión de bases de datos): Es una interfaz entre la base de datos y el usuario, las aplicaciones que la utilizan.

SGML (Standard Generalized Markup Language): El lenguaje estandarizado de marcas es el estándar internacional para definir las descripciones de la estructura de diferentes tipos de documentos electrónicos.

UML (Unified Modelling Language): Es la especificación de un lenguaje sin propietario para el modelado de objetos.

XML (Extensible Markup Language): Metalenguaje que describe esencialmente el contenido de lo que etiqueta, identificando el contenido. Es un lenguaje derivado de SGML.

Bibliografía

1. Hector Garcia-Molina, Jeffrey D. Ullman, y Jennifer Widom.
   Database Systems: The Complete Book.
   Prentice Hall, 2002.
   ISBN: 0130319953.

2. Abraham Silberschatz, Henry F. Korth, y S. Sudarshan.
   Database Systems Concepts.
   McGraw-Hill, 1986.
   ISBN: 0071122680.

3. E. F. Codd.
   A relational model of data for large shared data bank's.
   Communications of the ACM, volumén 26. Edicion 25 aniversario.
   Pags: 64-69 ,original 1970 publicación 1983.
   ISBN: 0001-0782.

4. Graham, Ian S.; Quin, Liam.
   XML Specification Guide.
   Ed. John Wiley & Sons, 1999.
   ISBN: 0471327530.

5. Timothy Lethbridge, Robert Laganiere.
   Object-Oriented Software Engineering: Practical Software Development using UML and Java
   Mcgraw-Hill, 2da Edicion 2004.
   ISBN: 0077109082.

6. Bruce E. Wampler.
   The Essence of Object-Oriented Programming with Java and UML.

Addison-Wesley Professional; Ed.CD-Rom 2001.
ISBN: 0201734109

7. Andrei Cioroinu; Mohammad Akif; Steven Brodhead .
JAVA Y XML
Ed. ANAYA MULTIMEDIA,1ª edición,Mayo 2002.
ISBN: 8441513546.

8. John Zukowski.
JAVA 2. J2SE 1.4
Ed. ANAYA MULTIMEDIA, 1ª edición , Julio 2003.
ISBN: 844151559X. ISBN-13: 9788441515598.

9. Joshua Marinacci, Chris Adamson.
Swing Hacks Tips and Tools for Killer GUIs
Ed. O'Reilly. 1ª edición, Junio 2005.
ISBN 10: 0-596-00907-0.
ISBN 13: 9780596009076.

10. Bruno R. Preiss.
Data Structures and Algorithms with Object-Oriented Design Patterns in Java.
Editorial John Wiley & Sons 1999.
ISBN 0471-34613-6.

Recursos bibliográficos en la red

- Definición de XML en la red
  http://en.wikipedia.org/wiki/XML
  (Consultado en Diciembre 2006)

- Definición de UML en la red
  http://en.wikipedia.org/wiki/Unified_Modeling_Language
  (Consultado en Diciembre 2006)

- Definición de DTD en la red
  http://en.wikipedia.org/wiki/Document_Type_Definition
  (Consultado en Diciembre 2006)

- Definición de JSF en la red
  http://java.sun.com/javaee/javaserverfaces/
  (Consultado en Julio 2007)

- Definición de JSF en la red
  http://es.wikipedia.org/wiki/Prolog
  (Consultado en Julio 2007)

- Definición de JNLP en la red
  http://es.wikipedia.org/wiki/JNLP
  (Consultado en Julio 2007)