

INSTITUTO TECNOLÓGICO DE CIUDAD MADERO
DIVISION DE ESTUDIOS DE POSGRADO E INVESTIGACION
MAESTRIA EN CIENCIAS DE LA COMPUTACIÓN



TESIS

IMPLEMENTACIÓN DE UN WIZARD PARA UNA INTERFAZ DE LENGUAJE
NATURAL A BASES DE DATOS PARA SU CONSULTA MEDIANTE INTERNET

Que para obtener el Grado de
Maestro en Ciencias de la Computación

Presenta
Ing. Gerardo González Hernández
G10070257

Director de Tesis
Dr. Rodolfo Abraham Pazos Rangel

Co-director de Tesis
Dr. Marco Antonio Aguirre Lam

Cd. Madero, Tams., a **04 de Diciembre de 2018**

OFICIO No.: U5.108/18
ÁREA: DIVISIÓN DE ESTUDIOS
DE POSGRADO E INVESTIGACIÓN
ASUNTO: AUTORIZACIÓN DE IMPRESIÓN
DE TESIS.

ING. GERARDO GONZÁLEZ HERNÁNDEZ
No. DE CONTROL G 10070257
PRESENTE

Me es grato comunicarle que después de la revisión realizada por el Jurado designado para su Examen de Grado de Maestro en Ciencias de la Computación, se acordó autorizar la impresión de su tesis titulada:

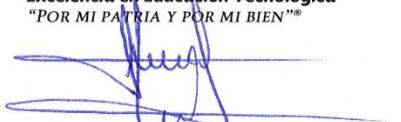
**“IMPLEMENTACIÓN DE UN WIZARD PARA UNA INTERFAZ DE LENGUAJE NATURAL A BASES DE DATOS
PARA SU CONSULTA MEDIANTE INTERNET “**

El Jurado está integrado por los siguientes catedráticos:

PRESIDENTE :	DR.	JOSÉ ANTONIO MARTÍNEZ FLORES
SECRETARIO:	DR.	JUAN JAVIER GONZÁLEZ BARBOSA
VOCAL:	DR.	RODOLFO ABRAHAM PAZOS RANGEL
SUPLENTE:	DR.	MARCO ANTONIO AGUIRRE LAM
DIRECTOR DE TESIS :	DR.	RODOLFO ABRAHAM PAZOS RANGEL
CO-DIRECTOR DE TESIS:	DR.	MARCO ANTONIO AGUIRRE LAM

Es muy satisfactorio para la División de Estudios de Posgrado e Investigación compartir con Usted el logro de esta meta. Espero que continúe con éxito su desarrollo profesional y dedique su experiencia e inteligencia en beneficio de México.

ATENTAMENTE
Excelencia en Educación Tecnológica®
“POR MI PATRIA Y POR MI BIEN”®



DR. JOSÉ AARÓN MELO BANDA
**JEFE DE LA DIVISIÓN DE ESTUDIOS
DE POSGRADO E INVESTIGACIÓN**



SECRETARÍA DE EDUCACIÓN PÚBLICA
TECNOLÓGICO NACIONAL
DE MÉXICO
INSTITUTO TECNOLÓGICO DE CIUDAD MADERO
DIVISIÓN DE ESTUDIOS DE POSGRADO
E INVESTIGACIÓN

c.c.p.- Archivo
Minuta

JAMB 'JAMF 'mdcoa*



Tabla de contenido

Capítulo 1 Introducción.....	12
1.1 Objetivos.....	12
1.2 Justificación y beneficios.....	13
1.3 Descripción del problema.....	13
1.4 Alcances y limitaciones.....	14
Capítulo 2 Marco teórico y trabajos relacionados.....	15
2.1 Marco teórico.....	15
2.1.1 Procesamiento de lenguaje natural.....	15
2.1.2 Bases de datos.....	16
2.1.3 Sistemas de administración de bases de datos.....	16
2.1.4 Structured Query Language (SQL).....	16
2.1.5 Lenguaje de definición de datos.....	17
2.1.6 Lenguaje de manipulación de datos.....	17
2.1.7 Administrador de bases de datos.....	17
2.1.8 Interfaces de lenguaje natural.....	17
2.1.9 Interfaces de lenguaje natural para bases de datos.....	18
2.1.10 Aplicación de web.....	19
2.1.11 Lenguaje de programación para web.....	19
2.2 Trabajos relacionados.....	21
2.2.1 Microsoft English Query.....	21
2.2.2 BirdQuest.....	22
2.2.5 Conclusiones sobre trabajos relacionados.....	26
Capítulo 3 Análisis y solución conceptual del problema.....	30
3.1 Descripción de la arquitectura de la versión anterior de la ILNBD.....	30
3.2 Descripción de la arquitectura de la versión actual para web.....	32
3.3 Requisitos de la nueva versión de la ILNBD.....	33
3.4 Descripción de la nueva arquitectura de la ILNBD.....	33
3.5 Descripción de la arquitectura del control de usuarios.....	34
Capítulo 4 Desarrollo de la ILNBD para web.....	36
4.1 Servidor de web.....	36
4.2 Sistema de administración de bases de datos para la ILNDB para web.....	37

4.3 Migración de bases de datos de Microsoft Access a PostgreSQL	37
4.4 Control de usuarios de la ILNBD	40
4.4.1 Tipos de usuarios.....	42
4.5 Implementación del control de usuarios	43
4.5.1 Implementación del registro de un nuevo usuario	43
4.5.2 Implementación del acceso de un usuario identificado.....	45
4.5.3 Implementación del acceso de un usuario invitado (no identificado).....	46
4.6 Implementación de la página de consulta	47
4.7 Implementación del editor de dominio	50
4.8 Implementación para la generación de dominio	51
4.9 Implementación de la página del wizard	53
4.9.1 Implementación del evento del botón procesar.....	56
4.9.2 Implementación del evento de la lista referencias	60
4.9.3 Implementación del evento de la lista tablas.....	60
4.9.4 Implementación de los eventos de los botones para asociar componente anterior y posterior.....	60
4.9.5 Implementación del evento de la opción valor.....	61
4.9.6 Implementación del evento del botón actualizar.....	62
4.9.7 Implementación del evento del botón eliminar	62
4.9.8 Implementación del evento del botón guardar	63
4.10 Cambios a la interfaz de consulta	63
4.11 Arquitectura de la ILNBD para web.....	65
4.11.1 Diagrama de la interfaz hombre-máquina.....	65
4.11.2 Estructura de navegación	66
4.11.3 Jerarquía de clases.....	66
Capítulo 5 Pruebas de la ILNBD	72
5.1 Objetivos de las pruebas	72
5.2 Descripción del escenario de pruebas	72
5.2.1 Hardware y software utilizado	73
5.2.2 Corpus de consultas para prueba.....	74
5.3 Casos de prueba para la ILNBD para web.....	74
5.4 Resultados de las pruebas	99
Capítulo 6 Conclusiones.....	100

6.1 Contribución del Proyecto	100
6.2 Conclusiones generales.....	100
6.3 Resultados obtenidos	101
6.4 Trabajos futuros	102
Referencias.....	103

Lista de Figuras.

Figura 2.1 Arquitectura general de una ILN.....	18
Figura 2.2 Esquema básico de una aplicación de web	19
Figura 2.3 Arquitectura cliente/servidor.....	21
Figura 2.4 Una parte de la ontología integrada que representa las conceptualizaciones tanto de la enciclopedia de aves como de los usuarios [Eriksson, 2003]	23
Figura 2.5 Nombramiento de elementos de la base de datos sobre el esquema [Minock, 2009].....	24
Figura 2.6 El sistema Cindi con la interfaz NLPQC [Stratica, 2002]	25
Figura 2.7 Arquitectura general de la ILNBD [Aguirre, 2014].....	27
Figura 2.8 Capas de funcionalidad [Aguirre, 2014]	28
Figura 3.1 Diagrama conceptual de la ILNBD versión de escritorio	31
Figura 3.2 Arquitectura de la ILNBD versión para web (<i>demo</i>).....	32
Figura 3.3 Arquitectura de la ILNBD versión para web	34
Figura 3.4 Arquitectura del proceso de registro de un nuevo usuario	35
Figura 3.5 Arquitectura del proceso de acceso de usuarios.....	35
Figura 4.1 Selección de base de datos de origen	38
Figura 4.2 Configuración del destino de la base de datos	39
Figura 4.3 Selección de tablas a emigrar	39
Figura 4.4 Opciones de transferencia	40
Figura 4.5 Diagrama conceptual de la asignación de acceso de un usuario a su diccionario de información semántica.....	42
Figura 4.6 Página de consulta para un usuario invitado.....	49
Figura 4.7 Página de consulta para un usuario identificado con opciones de configuración.....	50
Figura 4.8 Página de consulta para un usuario identificado con opciones de herramientas.....	50
Figura 4.9 Interfaz del wizard en la ILNBD de escritorio.....	54
Figura 4.10 Página del wizard en la ILNBD para web.....	55
Figura 4.11 Algoritmo para corregir problemas de columnas o tablas mal asociadas [Aguirre, 2014]	57
Figura 4.12 Algoritmo para corregir problemas de columnas o tablas no asociadas [Aguirre, 2014].....	57
Figura 4.13 Algoritmo para corregir problemas de valores imprecisos o alias no asociados [Aguirre, 2014]	58
Figura 4.14 Diagrama conceptual del proceso para resolver problemas de palabras no relacionadas	59
Figura 4.15 Interfaz de consulta de la ILNBD de escritorio	64
Figura 4.16 Interfaz de consulta de la ILNBD para web.....	64
Figura 4.17 Diagrama de la IHM de la ILNBD para web	65
Figura 4.18 Estructura de navegación de la IHM de la ILNBD para web.	66
Figura 4.19 Jerarquía de clases de la ILNBD para web	67

Figura 5.1 Escenario de pruebas <i>número uno</i>	73
Figura 5.2 Escenario de pruebas <i>número dos</i>	73
Figura 5.3 Resultado de una consulta en lenguaje natural	76
Figura 5.4 Resultado del procesamiento por lotes de consultas de ATIS	77
Figura 5.5 Resultado del procesamiento por lotes de consultas de Geobase	77
Figura 5.6 Selección del diccionario de información semántica	78
Figura 5.7 Resultado de una consulta en lenguaje natural	79
Figura 5.8 Diálogo para eliminar un <i>DIS</i>	79
Figura 5.9 Eliminación del <i>DIS Prueba y Prueba2</i>	80
Figura 5.10 <i>DIS</i> disponibles después de la eliminación	80
Figura 5.11 Editor de dominio.....	81
Figura 5.12 Resultado de una consulta en LN con problema de columnas no asociadas	82
Figura 5.13 Consulta introducida por el DBA.....	83
Figura 5.14 Sugiere asociar a la palabra <i>número</i> a la columna <i>flight.flight_number</i>	83
Figura 5.15 Componentes de la consulta.....	84
Figura 5.16 Asociar la palabra <i>número</i> con la palabra <i>vuelo</i>	84
Figura 5.17 El sintagma formado es <i>vuelo número</i>	85
Figura 5.18 Cambio realizado en el componente de la consulta	85
Figura 5.19 Actualización de los descriptores de la palabra <i>tarifa</i>	86
Figura 5.20 Resultado de la consulta en LN después de utilizar el wizard	86
Figura 5.21 Resultado de una consulta en LN con problema de columnas mal asociadas ..	87
Figura 5.22 Sugiere actualizar la referencia de la palabra <i>a</i> de la columna <i>flight.to_airport</i> a la columna <i>fare.to_airport</i>	88
Figura 5.23 Sugiere actualizar la referencia de la palabra <i>desde</i> de la columna <i>flight.from_airport</i> a la columna <i>fare.from_airport</i>	89
Figura 5.24 Componentes actualizados de la consulta	89
Figura 5.25 Resultado de la consulta en LN después de utilizar el wizard	90
Figura 5.26 Resultado de una consulta en LN con problema de valor impreciso no asociado	91
Figura 5.27 Consulta introducida por el DBA.....	92
Figura 5.28 Sugiere asociar a la palabra <i>mañana</i> a la columna <i>flight.departure_time</i>	92
Figura 5.29 Sugiere asociar a la palabra <i>a</i> a la columna <i>flight.from_airport</i>	93
Figura 5.30 Componentes actualizados de la consulta	93
Figura 5.31 Actualización de las palabras <i>desde</i> y <i>a</i> para los descriptores de las columnas <i>flight.from_airport</i> y <i>flight.to_airport</i>	94
Figura 5.32 Resultado de la consulta en LN después de utilizar el wizard	94
Figura 5.33 Resultado de una consulta en LN con problema de valor alias no asociado.....	95
Figura 5.34 Consulta introducida por el DBA.....	96
Figura 5.35 Sugiere asociar la palabra <i>mediodía</i> a la columna <i>flight.departure_time</i>	96
Figura 5.36 Componentes actualizados de la consulta	97
Figura 5.37 Actualización de la palabra <i>vuelos</i> para los descriptores de la columna <i>flight.flight_number</i>	97

Figura 5.38 Actualización de la palabra <i>salgan</i> para los descriptores de la columna <i>flight.departure_time</i>	98
Figura 5.39 Resultado de una consulta en LN después de utilizar el wizard	98

Lista de Tablas.

Tabla 2.1 Características de configuración vía web de ILNBDs	27
Tabla 4.1 Columnas de la tabla usuarios	41
Tabla 4.2 Lista de elementos de HTML que se utilizaron en la página del wizard	55
Tabla 4.3 Lista de objetos o variables utilizadas en la clase Control_usuario	67
Tabla 4.4 Lista de objetos o variables utilizadas en la clase Consulta	68
Tabla 4.5 Lista de objetos o variables utilizadas en la clase Metadatos.....	69
Tabla 4.6 Lista de objetos o variables utilizadas en la clase Wizard.....	70
Tabla 5.1 Especificaciones de hardware y software de los escenarios de pruebas	74
Tabla 5.2 Comparación de resultados obtenidos en las configuraciones del DIS	99

Declaración de Originalidad

Declaro y prometo que este documento de tesis es producto de mi trabajo original y que no infringe los derechos de terceros, tales como derechos de publicación, derechos de autor, patentes y similares.

Además, declaro que en las citas textuales que he incluido (las cuales aparecen entre comillas) y en los resúmenes que he realizado de publicaciones ajenas, indico explícitamente los datos de los autores y las publicaciones.

Además, en caso de infracción de los derechos de terceros derivados de este documento de tesis, acepto la responsabilidad de la infracción y relevo de ésta a mi director y codirectores de tesis, así como al Instituto Tecnológico de Cd. Madero y sus autoridades.

12 de diciembre de 2018, Cd. Madero, Tamps.



Ing. Gerardo González Hernández

Agradecimientos

Agradezco al Dr. Rodolfo A. Pazos Rangel por dar seguimiento atento a este proyecto y además haber aportado sus conocimientos, experiencia, apoyo y tiempo para el desarrollo de este trabajo.

Mi profundo agradecimiento a los miembros del comité tutorial de esta tesis, Dr. Rodolfo Pazos, Dr José A. Martínez, Dr. Juan J. González y Dr. Marco A. Aguirre por las sugerencias brindadas durante el desarrollo de esta tesis.

Gracias al Instituto Tecnológico de Cd. Madero (ITCM) por darme la oportunidad de seguir adelante con mis estudios, y al Consejo Nacional de Ciencia y Tecnología (CONACYT) por el apoyo económico que me otorgó para la realización de mis estudios.

Agradezco a mis compañeros por brindarme su apoyo, amistad y confianza, principalmente a Jorge A. Castro Rivera, Moisés I. Herrera Ramos y Jesús E. Morán Escalante.

Gracias totales.

Dedicatorias

Dedico esta tesis a mis padres Fidel González Hernández y Benita Hernández Hernández por todas sus enseñanzas, el cariño que me brindan y por apoyarme en los momentos más decisivos de mi vida.

A mi hermano José Angel González Hernández por todas sus enseñanzas, su apoyo y cariño.

Resumen

Las grandes empresas utilizan computadoras para almacenar información, la cual se encuentra almacenada generalmente en Bases de Datos (BDs). Para facilitar la obtención de los datos contenidos en estas bases de datos, se han creado Interfaces de Lenguaje Natural para Bases de Datos (ILNBDs), que facilitan a los usuarios inexpertos la obtención de información de las bases de datos.

El presente proyecto de tesis consiste en desarrollar una arquitectura modular de una ILNBD para que pueda ser usada mediante un navegador de web y de esta manera, formular consultas en Lenguaje Natural (LN) y afinar la configuración del Diccionario de Información Semántica (DIS) mediante el uso de un wizard (asistente); es decir, convertir una versión anterior de escritorio a una nueva versión para web.

El punto de partida para la implementación de este proyecto fue el uso de una ILNBD de escritorio que fue implementada en el Instituto Tecnológico de Cd. Madero, para la cual se propuso el uso de un DIS para mejorar el desempeño de la ILNBD. Usualmente, el desempeño de una ILNBD depende mucho de su configuración, es decir, cuanto mejor esté configurada, mayor será el porcentaje de consultas correctamente contestadas por la ILNBD. Con el fin de mejorar el desempeño de la ILNBD, utiliza un wizard para afinar la configuración del DIS.

Durante el desarrollo del proyecto, se realizó un análisis de los algoritmos en la versión anterior. Dicho análisis fue usado para determinar qué tecnologías de programación de aplicaciones de web se utilizarían, y así proponer una nueva arquitectura para el desarrollo de la ILNBD para web.

Una aportación de este proyecto es la incorporación de tres algoritmos que permite al wizard corregir tres tipos de problemas: problemas de tablas o columnas mal asociadas, problemas de tablas o columnas no asociadas y problemas de valores alias o valores imprecisos no asociados.

Para demostrar que la ILNBD para web tiene el mismo desempeño que la ILNBD de escritorio respecto al porcentaje de consultas correctamente traducidas, se realizaron pruebas en ambas interfaces. Para dichas pruebas se utilizó un corpus de 70 consultas para la base de datos ATIS.

El desempeño de la ILNBD de escritorio, utilizando la configuración automática de la interfaz, es en promedio 24.28% de consultas correctamente traducidas, al igual que la ILNBD para web. Se realizó otra prueba en ambas interfaces utilizando el wizard para afinar la configuración del DIS, obteniendo en ambas interfaces un 85% de consultas correctamente contestadas. De esta manera se demuestra que el wizard de la ILNBD para web funciona correctamente; es decir, de la misma manera que el wizard de la ILNBD de escritorio.

Capítulo 1

Introducción

Actualmente el uso de la computadora ha ido creciendo rápidamente a medida que la tecnología va avanzando. En las grandes empresas, como en la mayoría de los hogares, cuentan con equipos de cómputo para realizar tareas y almacenar información. La gran mayoría de los datos son almacenados en bases de datos (BDs). Sin embargo, se requiere que el usuario tenga una capacitación apropiada para tener amplio acceso a los datos de una forma correcta. La gran mayoría de los usuarios no tienen esta preparación.

Para facilitar la obtención de la información contenida en las bases de datos, se propuso una forma en la que los usuarios inexpertos puedan hacer consultas a estas bases de datos sin la necesidad de aprender un lenguaje de consulta por ejemplo, el lenguaje de consulta estructurada (en inglés SQL, Structured Query Language), de tal suerte que puedan formular sus preguntas de una manera más fácil. Por lo tanto, surge la idea de utilizar el lenguaje natural (LN) como medio para consultar las bases de datos. Sin embargo, las computadoras no tienen la capacidad de comprender directamente este lenguaje. De esta manera surgen lo que hoy se conocen como interfaces de lenguaje natural para bases de datos (ILNBDs), las cuales permiten a los usuarios acceder a la información que se encuentra almacenada en la base de datos por medio de una consulta en lenguaje natural.

La principal razón de este proyecto es la necesidad de facilitar el acceso a una ILNBD [Aguirre, 2014] para que pueda ser utilizada desde un navegador de web. Esto permitiría que la ILNBD pudiera ser consultada y configurada de forma remota (es decir desde Internet) por más de un usuario, lo cual ayudará a que la configuración de la ILNBD pueda ser afinada adecuadamente para aumentar su tasa de consultas correctamente contestadas.

1.1 Objetivos

El objetivo general consiste en desarrollar una arquitectura modular de la ILNBD para que ésta pueda ser usada mediante un navegador de web y de esta manera, pueda ser consultada y afinada mediante una conexión de internet; es decir, convertir la aplicación de escritorio a una aplicación de web.

Para alcanzar el objetivo general, es necesario alcanzar los siguientes objetivos específicos:

OE.1) Realizar un análisis de requerimientos de la nueva ILNBD.

OE.2) Realizar una investigación sobre las diferentes tecnologías de web existentes y su compatibilidad con el estado actual de la ILNBD.

OE.3) Diseñar una nueva arquitectura modular.

OE.4) Rediseñar los algoritmos de la ILNBD actual para que sean compatibles con la nueva versión para web.

OE.5) Proponer un sistema de administración de bases de datos que permita su consulta mediante internet y además sea lo más apegado posible a los estándares de SQL.

OE.6) Diseñar e implementar un núcleo de la interfaz que permita la incorporación de nuevos módulos.

OE.7) Incorporar el wizard en la ILNBD para su consulta mediante un navegador de web.

1.2 Justificación y beneficios

La principal justificación es que actualmente existe una aplicación que fue desarrollada para escritorio (*desktop*) de una ILNBD. Esta ILNBD cuenta con un asistente (wizard) que sirve para afinar la configuración de la ILBD, lo cual permite traducir mejor las consultas que son formuladas por los usuarios. Desafortunadamente, el wizard únicamente puede ser usado por un usuario, ya que está instalado en una computadora aislada.

Por lo tanto, el beneficio que se obtendrá con este proyecto es que la ILNBD permitirá ser consultada y configurada de forma remota usando una conexión de internet por uno o hasta cinco usuarios simultáneos.

Otro beneficio muy importante es que la ILNBD permitirá ser configurada para una o más bases de datos y así tener una mayor aplicabilidad.

1.3 Descripción del problema

El problema es que la versión de ILNBD de Aguirre está implementada para una computadora aislada; es decir, que la aplicación es monousuario. Esta situación ocasiona que únicamente un solo usuario puede utilizar esa aplicación para formular consultas en lenguaje natural y ser capaz de configurar la ILNBD con ayuda del asistente (wizard). Esto a su vez ocasiona que únicamente permite administrar una base de datos a la vez para hacer las pruebas de la interfaz, aunque la ILNBD está configurada para dos BDs.

Uno de los problemas en la implementación de la nueva versión de la ILNBD consistió en que debería funcionar en la internet por medio de un navegador de web. También, debería permitir que la ILNBD trabaje con varias bases de datos, y además, ser accedida por uno o hasta cinco usuarios simultáneamente para formular consultas en lenguaje natural. Además, la nueva versión debería permitir configurar la ILNBD con ayuda del asistente (wizard), de tal manera que permita a los usuarios modificar el diccionario de información semántica de la ILNBD de una manera independiente para cada uno (sin interferir las configuraciones de unos con las de otros).

1.4 Alcances y limitaciones

Alcance

- Desarrollar y diseñar una arquitectura para que la ILNBD que fue implementada en el Instituto Tecnológico de Ciudad Madero (ITCM) [Aguirre, 2014] pueda ser accedida mediante la web.
- Que la ILNBD pueda hacer consultas a las bases de datos utilizando otro sistema de administración de bases de datos que permita ser consultada mediante la web.
- Que más de un usuario pueda configurar el diccionario de datos de la ILNBD a través de la web utilizando el wizard.
- El nuevo núcleo de la interfaz debe permitir incorporar actualizaciones a la ILNBD sin que éstas afecten el desempeño de los módulos que no hayan sido modificados directamente por la actualización.

Limitaciones

- La ILNBD únicamente puede manejar hasta 5 usuarios simultáneamente para la formulación de consultas y para la configuración del diccionario de datos.
- No se desarrollaron mejoras en cuanto a desempeño y funcionalidad en el funcionamiento de la ILNBD, adicionales a las indicadas en el alcance.

Capítulo 2

Marco teórico y trabajos relacionados

En este capítulo se presenta el marco teórico y los trabajos relacionados que conforma este proyecto de tesis. En el marco teórico destacan los conceptos más importantes utilizados durante el desarrollo de este proyecto. En los trabajos relacionados se presentarán los trabajos que se investigaron en la realización de este proyecto de tesis.

2.1 Marco teórico

En el marco teórico que se presenta a continuación, permite conocer los conceptos básicos necesarios para el entendimiento del desarrollo de este proyecto de tesis.

2.1.1 Procesamiento de lenguaje natural

Lenguaje. Conjunto de sonidos articulados con que el hombre manifiesta lo que piensa o siente [RAE, 2017].

Cuando se habla de lenguajes se pueden diferenciar dos clases muy bien definidas:

1. Los lenguajes naturales como el español, inglés, francés, etc.
2. Los lenguajes formales como los lenguajes de programación, el lenguaje de la lógica matemática, etc [Cortez, 2009].

Lenguaje formal. Un lenguaje formal es un lenguaje artificial creado por el hombre, el cual está formado por símbolos y fórmulas, y tiene como objetivo fundamental formalizar la programación de computadoras o representar simbólicamente un conocimiento [Rojas, 2009].

En matemáticas, lógica y ciencias de la computación, un lenguaje formal es un lenguaje cuyos símbolos primitivos y reglas para unir esos símbolos están formalmente especificados [Honderich, 2005].

Lenguaje natural. Lenguaje hablado o escrito por humanos, opuesto a un lenguaje de programación utilizado para programar o comunicarse con computadoras [Farlex, 2003].

Otra de las consideraciones en la comprensión del lenguaje natural escrito, además del entendimiento de las palabras por separado, es “entender la estructura de la oración”, que también es difícil. Este tema se puede dividir en tres análisis diferentes [Andrade, 1997]:

1. Análisis sintáctico de la oración.
2. Análisis semántico.
3. Análisis interpretativo.

Procesamiento de lenguaje natural (PLN). Es el campo que combina las tecnologías de la ciencia computacional (como la inteligencia artificial, el aprendizaje automático o la inferencia estadística) con la lingüística aplicada, con el objetivo de hacer posible la comprensión y el procesamiento asistidos por computadora de información expresada en lenguaje humano para determinadas tareas, como la traducción automática, los sistemas de diálogo interactivos, el análisis de opiniones, etc. [Vicomtech, 2017].

2.1.2 Bases de datos

Un sistema de base de datos es básicamente un sistema computarizado para guardar registros; es decir, es un sistema computarizado cuya finalidad general es almacenar información y permitir a los usuarios recuperar y actualizar esa información con base en peticiones [Faudón, 2001].

2.1.3 Sistemas de administración de bases de datos

Un sistema de administración de bases de datos (en español SABD, en inglés DBMS, Database Management System) es un conjunto de elementos interrelacionados y una serie de programas que permiten a varios usuarios tener acceso a estos archivos ya sea para consultarlos o actualizarlos [Faudón, 2001].

2.1.4 Structured Query Language (SQL)

SQL (lenguaje de consultas estructurado) fue desarrollado por IBM. SQL es una combinación de constructores de álgebra relacional y del cálculo racional. Usando SQL se puede definir la estructura de los datos, además de poder modificar los datos de la base de datos y especificar restricciones de seguridad. Actualmente SQL es el estándar de uso en la inmensa mayoría de los SABDs comerciales [Osorio, 2008].

2.1.5 Lenguaje de definición de datos

Un lenguaje de definición de datos (en inglés DDL, Data Definition Language) es un lenguaje especial que permite la especificación de un esquema de base de datos por medio de una serie de definiciones.

El resultado de la compilación de las instrucciones en DDL es un conjunto de tablas que se almacenan en un archivo especial conocido como diccionario de datos [Osorio, 2008].

2.1.6 Lenguaje de manipulación de datos

Un lenguaje de manipulación de datos (en inglés DML, Data Manipulation Language) permite a los usuarios manejar o tener acceso a los datos que estén organizados por medio del modelo apropiado [Osorio, 2008]. Existen dos tipos de DML:

- **Procedimentales**, los cuales requieren que el usuario especifique qué datos quiere y cómo obtenerlos.

No procedimentales, que requieren que el usuario especifique qué datos quiere sin especificar cómo obtenerlos.

2.1.7 Administrador de bases de datos

A la persona que tiene un control centralizado sobre un DBMS se le conoce como el Administrador de Base de Datos (en inglés DBA, Database Administrator) [Osorio, 2008]. Sus funciones principales son:

1. **Definición de esquema.** Es el creador del esquema original de la base de datos en un DDL.
2. **Modificación del esquema y de su organización física.**
3. **Administración de la seguridad.** Concede autorizaciones para el acceso a los datos a los distintos usuarios de la base de datos. Garantiza la permanencia de los datos mediante copias de seguridad.
4. **Especificación de las limitaciones de consistencia.**

2.1.8 Interfaces de lenguaje natural

Mecanismos de comunicación entre una persona y una máquina a través de lenguaje natural. En la mayoría de las ocasiones esta comunicación es bidireccional, siendo de tipo pregunta-

respuesta [Aguirre, 2014]. La arquitectura general de una interfaz de lenguaje natural se muestra en la Figura 2.1.

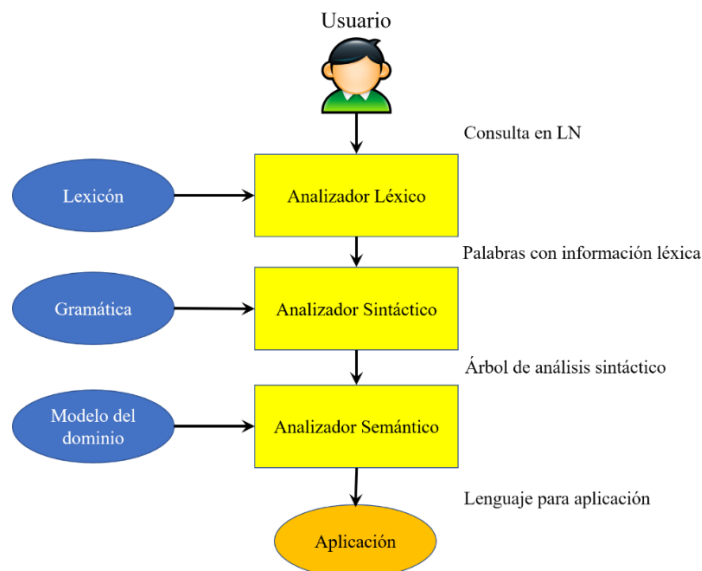


Figura 2.1 Arquitectura general de una ILN

2.1.9 Interfaces de lenguaje natural para bases de datos

Una ILNBD es un sistema que permite a un usuario acceder a la información de una base de datos, mediante una solicitud en lenguaje natural [Aguirre, 2014].

También puede ser definida como un sistema que permite al usuario acceder a la información almacenada en una base de datos escribiendo peticiones expresadas en algún lenguaje natural [Androutsopoulos, 1995].

2.1.9.1 ILNBD de dominio específico

Son aquellas ILNBDs que únicamente pueden consultar una sola base de datos [Pazos, 2012].

2.1.9.2 ILNBD independiente de dominio

Son aquellas ILNBDs que pueden usarse para consultar diferentes bases de datos [Pazos, 2012].

Las ILNBDs que ofrecen independencia de dominio necesitan un proceso inicial de configuración antes de ser utilizadas para hacer consultas por los usuarios finales. La configuración consiste en proporcionar al sistema las palabras y conceptos necesarios para el dominio, los cuales están relacionados con la información almacenada en la base de datos [Pazos, 2012].

2.1.10 Aplicación de web

Una aplicación de web es un tipo especial de aplicación cliente/servidor, donde tanto el cliente como el servidor y el protocolo mediante el que se comunican están estandarizados y no han de ser creados por el programador de aplicaciones [Mora, 2002]. El esquema básico de una aplicación de web se muestra en la Figura 2.2.

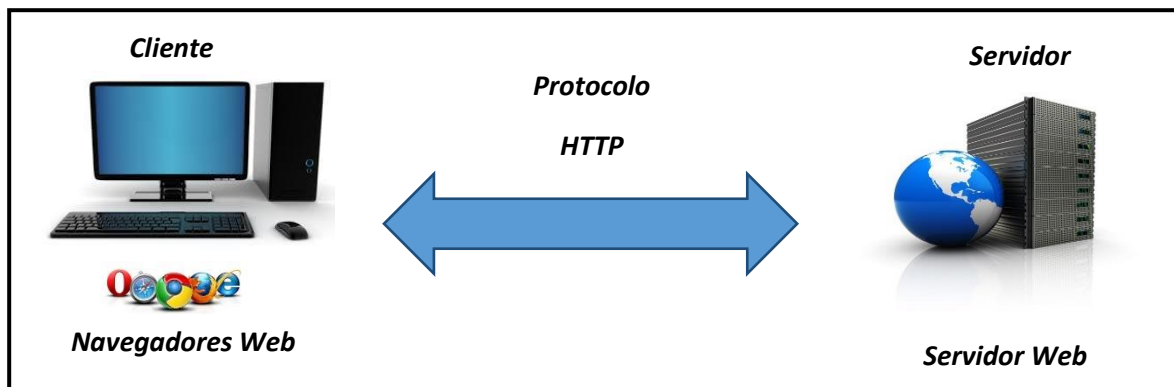


Figura 2.2 Esquema básico de una aplicación de web

Cliente de web. Es un programa con el que interacciona el usuario para solicitar a un servidor de web el envío de los recursos que desea obtener mediante el Hypertext Transfer Protocol (HTTP) [Mora, 2002].

Las tecnologías que se suelen emplear para programar el cliente de web son:

- Hypertext Markup Language (HTML).
- Cascading Style Sheets (CSS).
- Lenguajes de script: JavaScript.
- Applets programados en Java.

Servidor de web. Es un software que está esperando permanentemente las solicitudes de conexión mediante el protocolo HTTP por parte de los clientes de web [Mora, 2002].

2.1.11 Lenguaje de programación para web

Los lenguajes de programación para web han ido surgiendo según las necesidades de las plataformas, intentando facilitar el trabajo a los desarrolladores de aplicaciones. Se clasifican en lenguajes del lado cliente y lenguajes del lado servidor [EcuRed, 2017].

2.1.11.1 Lenguaje del lado cliente

Son aquellos lenguajes que son asimilados directamente por el navegador y no necesitan pretratamiento.

2.1.11.1.1 Lenguaje HTML

El HyperText Markup Language (HTML) es un lenguaje de marcado que se diseñó con el objetivo de estructurar documentos y mostrarlos en forma de hipertexto [W3C Superseded Recommendation, 2017].

HTML puede ser definido de una manera más sencilla como un lenguaje de descripción de hipertexto compuesto por una serie de comandos, marcas o etiquetas, también denominadas *tags*, las cuales permiten definir la estructura lógica de un documento de web y establecer los atributos del mismo [Cobo, 2005].

2.1.11.1.2 JavaScript

Es un lenguaje de programación que se utiliza principalmente para crear páginas web dinámicas. Una página web dinámica es aquella que incorpora efectos como texto que aparece y desaparece, animaciones, acciones que se activan al pulsar botones y ventanas con mensajes de aviso al usuario. JavaScript es un lenguaje de programación interpretado, por lo que no es necesario compilar los programas para ejecutarlos [Pérez, 2009].

2.1.11.2 Lenguaje del lado servidor

Son aquellos lenguajes que se ejecutan en el propio servidor y son enviados al cliente en un formato claro para él.

2.1.11.2.1 PHP

PHP (Hypertext Preprocessor) es un lenguaje de código abierto muy popular especialmente adecuado para el desarrollo para web y que puede ser inmerso en HTML [The PHP Group, 2017].

La diferencia de PHP con los lenguajes que se utilizan en el lado cliente como JavaScript es que el código es ejecutado en el lado servidor, generando HTML y enviándolo al cliente, que lo solicita, lo que permite ocultarle al cliente el código que es ejecutado.

2.1.11.3 Arquitectura cliente/servidor

Cliente/servidor es una arquitectura de red en la que cada ordenador o proceso en la red es cliente o servidor [Mora, 2002]. En la Figura 2.3 se muestra la arquitectura cliente/servidor.

Servidor. Son ordenadores poderosos dedicados a gestionar unidades de disco (servidor de disco), tráfico de red (servidor de red), datos (servidor de base de datos) o de aplicaciones (servidor de aplicaciones).

Cliente. Son máquinas menos poderosas que usan los recursos que ofrecen los servidores.

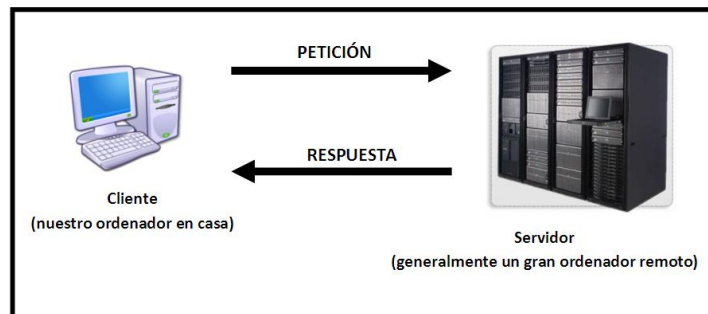


Figura 2.3 Arquitectura cliente/servidor

2.2 Trabajos relacionados

En esta sección se presentan los trabajos relacionados que se investigaron para la realización del proyecto de tesis. En donde se describen cuatro ILNBDs que se encontraron en la literatura, en donde se destacan las siguientes ILNBDs: Microsoft English Query, BirdQuest, C-Phrase y Cindi The Virtual Library.

Al final de esta sección se encuentran las conclusiones de la investigación realizada de los trabajos relacionados.

2.2.1 Microsoft English Query

English Query es un conjunto de herramientas que los administradores de bases de datos, desarrolladores de aplicaciones y profesionales de la web pueden utilizar para desarrollar una ILNBD. Con las aplicaciones de consulta en inglés, los usuarios pueden formular consultas a bases de datos ad hoc utilizando preguntas o declaraciones en inglés.

English Query proporciona un entorno robusto para desarrollar un modelo de consulta en inglés. Sin embargo, como las bases de datos tienden a ser únicas y los usuarios hacen

preguntas únicas, crear un modelo que responda a las preguntas de los usuarios puede ser un proceso complejo.

Por ejemplo, un usuario puede formular la siguiente pregunta:

Who wrote The Gourmet Microwave?

Microsoft English Query traduce automáticamente la pregunta en inglés a la siguiente instrucción en SQL [Spenik, 2003]:

```
SELECT DISTINCT dbo.authors.au_fname AS "First Name",  
dbo.authors.au_lname AS "Last Name"  
FROM dbo.titles, dbo.titleauthor, dbo.authors  
WHERE dbo.titles.title = 'The Gourmet Microwave'  
AND dbo.titles.title_id = dbo.titleauthor.title_id  
AND dbo.titleauthor.au_id = dbo.authors.au_id
```

En una aplicación que incluye English Query, el motor de English Query realiza la traducción. El motor utiliza un archivo de dominio (.eqd), que contiene un modelo de la base de datos. El modelo contiene información específica de la base de datos consultada por la aplicación, incluido el esquema de la base de datos, una capa de abstracción semántica construida sobre el esquema y el mapeo entre ellos. En el modelo, las tablas y los campos están representados por entidades y las reuniones (*joins*) están representadas por relaciones. Las entidades y relaciones definidas en un modelo permiten a English Query traducir el inglés a Transact-SQL [Microsoft Corporation, 2009].

El Asistente para Proyectos de English Query (English Query Project Wizard en inglés) puede definir automáticamente algunas entidades y relaciones basadas en la estructura de la base de datos. Los desarrolladores definen otras entidades y relaciones usando el entorno de desarrollo de English Query. Un modelo completo y bien diseñado permitirá a una aplicación de consulta en inglés hacer un mejor trabajo de traducir preguntas en LN a consultas en SQL [Microsoft Corporation, 2009].

Actualmente en la página de Microsoft dice que English Query ha sido discontinuado en las versiones recientes, Microsoft no ha dado información sobre por qué English Query no fue continuado.

2.2.2 BirdQuest

Es un sistema que combina la interacción del diálogo con la extracción de información, la cual apoya la interacción del diálogo para acceder a los datos textuales en una enciclopedia de aves. Los datos de origen se proporcionan inicialmente como texto no estructurado, pero se refinan con técnicas de extracción de información que se utilizarán dentro de un marco de sistema de diálogo. BirdQuest utiliza una ontología para representar muchas de las tareas en el dominio del sistema de conocimiento.

El componente de interacción en BirdQuest se basa en el marco MALIN. MALIN es un sistema de diálogo modularizado y separa la gestión del diálogo (DM) de la gestión del conocimiento del dominio (DKM). El primero maneja el diálogo mientras que el segundo maneja el acceso a varias fuentes de información de fondo. El modelo de diálogo MALIN clasifica los segmentos del discurso por categorías de actos de habla general, tales como la consulta (Q) y la respuesta (A). En cambio, el gestor de diálogo utiliza los parámetros focales para controlar la interacción. En BirdQuest los objetos son normalmente pájaros y la información del modelo de las propiedades sobre las aves, como la apariencia, el número de huevos y la alimentación.

Para el procesamiento de documentos y consultas de usuarios se utilizan varias fuentes de conocimiento. Estas fuentes de conocimiento compartidas comprenden léxico, gramática y ontologías de dominio. La ontología compartida para el sistema BirdQuest se desarrolló a partir del análisis de dos tipos diferentes de material empírico: una enciclopedia de aves y un corpus de consultas. El corpus consiste en más de 250 consultas sobre aves [Eriksson, 2003].

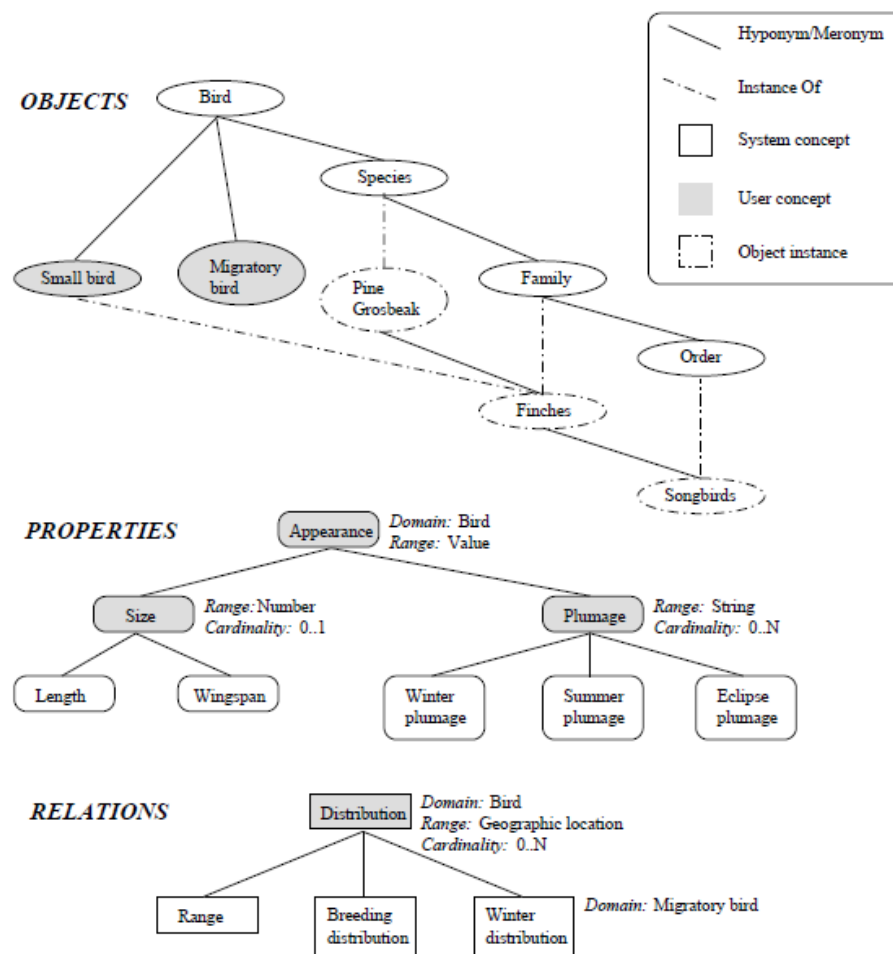


Figura 2.4 Una parte de la ontología integrada que representa las conceptualizaciones tanto de la enciclopedia de aves como de los usuarios [Eriksson, 2003]

2.2.3 C-Phrase

C-Phrase es un sistema de interfaz de lenguaje natural que puede ser configurado por equipos técnicos normales, no especializados y basados en la web. C-Phrase modela consultas en una versión extendida del cálculo de tuplas de Codd y utiliza gramáticas sincrónicas independientes de contexto con expresiones lambda para representar gramáticas semánticas.

El configurador de una ILN construye la gramática semántica a través de una serie de operaciones de nombrado, configuración y definición dentro de una GUI basada en web. C-Phrase tiene una herramienta de creación basada en AJAX que proporciona una GUI integrada a través de la cual un configurador puede importar un esquema desde cualquier base de datos accesible con ODBC y comenzar lo que se denomina ciclo nombrado-configuración-definición de la creación de una ILN. Las acciones de nomenclatura proporcionan nombres de texto simples para las relaciones, atributos y rutas de reunión (*join*) del esquema de la base de datos.

La Figura 2.5 muestra el navegador de esquemas de la herramienta donde el usuario tiene la opción de explorar y nombrar los diversos elementos de la base de datos. Los atributos y las relaciones con asteriscos ya han sido nombrados, y el usuario está en el proceso de nombrar la reunión de CITY a STATE. Las acciones de nomenclatura disparan la definición de reglas léxicas predeterminadas de cabeza, premodificador y postmodificador. Estas reglas predeterminadas se forman mediante el acoplamiento de expresiones lógicas sobre los elementos de la base de datos, con palabras y frases asociadas de la relación de nombrado (una tabla del diccionario de datos) [Minock, 2009].

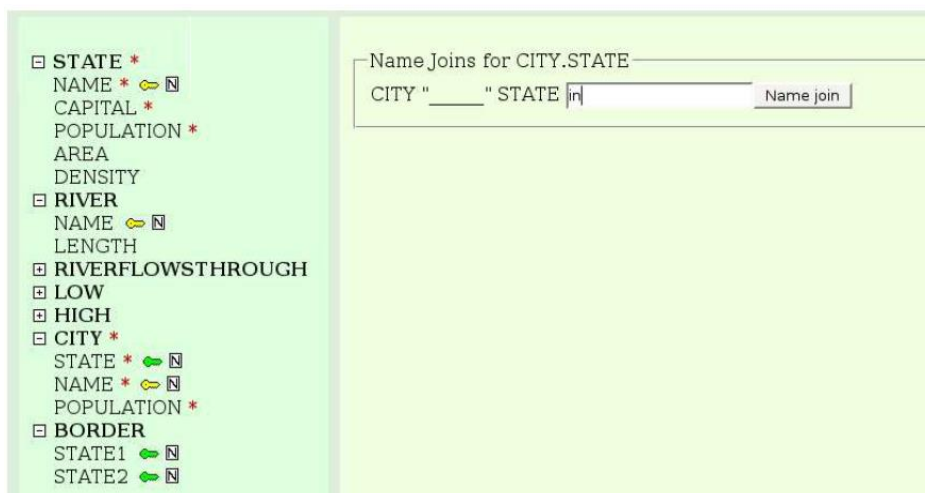


Figura 2.5 Nombramiento de elementos de la base de datos sobre el esquema [Minock, 2009]

Esta interfaz ha sido probada con la base de datos Geobase, con un conjunto de 100 consultas obtenidas del corpus de Geoquery.

2.2.4 Cindi The virtual library

Cindi es el nombre de un sistema de indexado y búsqueda vía internet de una biblioteca virtual de la Universidad de Concordia. El procesador de lenguaje natural, llamado NLPQC, analiza semánticamente las preguntas de lenguaje natural y genera consultas en SQL correspondientes para la base de datos. Esto hace que el sistema NLPQC sea una interfaz de lenguaje natural para bases de datos relacionales. El sistema NLPQC está diseñado para ser independiente de la plataforma y puede aceptar cualquier esquema de base de datos [Stratica, 2002].

El sistema NLPQC acepta frases de lenguaje natural del usuario, las analiza semánticamente y genera una consulta en SQL para la base de datos de Cindi. La funcionalidad principal del sistema se basa en reglas y plantillas. El administrador del sistema puede modificar o redefinir las reglas.

NLPQC está diseñado para aceptar la entrada del usuario en inglés a través de una página en HTTP y el servidor ASF Apache1. Genera una consulta en SQL para un motor de bases de datos relacional [Stratica, 2002].

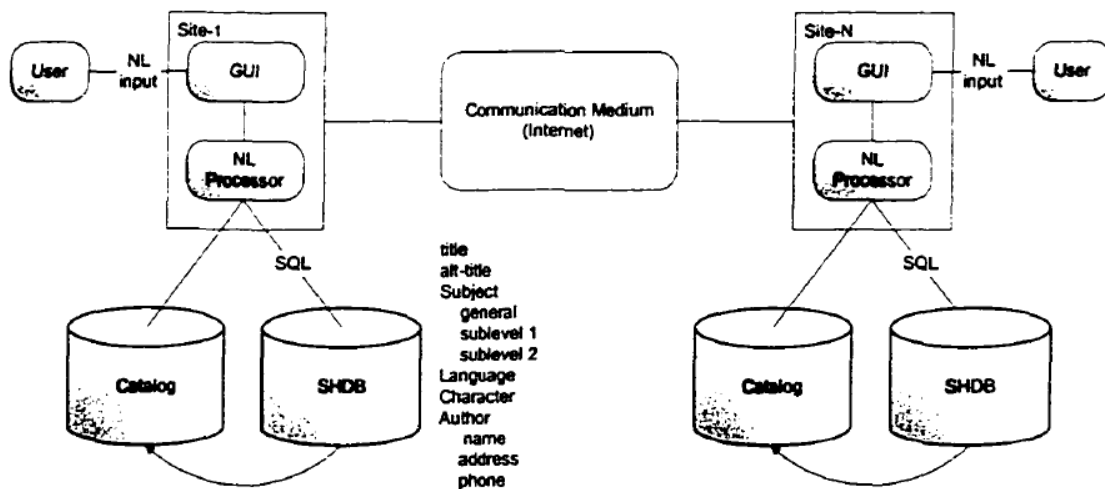


Figura 2.6 El sistema Cindi con la interfaz NLPQC [Stratica, 2002]

La funcionalidad del sistema se ha dividido en dos partes: primera, el preprocesador interpreta el esquema de la base de datos y construye las reglas del sistema, que posteriormente se utilizan en el análisis semántico; segunda, el módulo de tiempo de ejecución procesa la entrada del usuario y realiza el análisis semántico basado en las reglas establecidas por el preprocesador [Stratica, 2002].

NLPQC puede generar consultas en SQL que involucren una, dos o tres tablas. No puede resolver la información de fecha y lugar y no puede utilizar más de dos columnas por tabla.

2.2.5 Conclusiones sobre trabajos relacionados

La diferencia que existe entre English Query y la ILNBD de Aguirre es que English Query proporciona únicamente el asistente para hacer la configuración inicial, para que después los usuarios puedan formular consultas a una base de datos. En cambio, la ILNBD proporciona una herramienta de configuración inicial y además una afinación posterior: el wizard (ver Figura 2.7). Esta herramienta permite al DBA escribir una consulta en LN (que no ha sido contestada correctamente por la ILNBD) y la traducción correcta a SQL. El wizard compara las instrucciones en SQL correcta y errónea para identificar diferencias en ambas instrucciones, y así el asistente determina qué tipo de error se produjo y sugiere una modificación/adición de información semántica en el Diccionario de Información Semántica (Diccionario de datos en la Figura 2.7) con el fin de evitar que este error se repita.

BirdQuest es una ILNBD exclusiva para una base de datos; por lo tanto, se considera dependiente de dominio. Ya que BirdQuest es para una sola base de datos, entonces no necesita una herramienta de configuración como se requiere para ILNBDs independientes de dominio.

C-Phrase ha sido probada con varias bases de datos: Geobase, Northwind y Copestake; por lo tanto, se puede considerar independiente de dominio. En cuanto a configuración, C-Phrase permite importar información del esquema de la base de datos a consultar y cuenta con una herramienta de autoría semejante a la Interfaz de Configuración de nuestra ILNBD (Fig. 2.7). La configuración de C-Phrase es extremadamente complicada; por ejemplo, el archivo de configuración para Geobase consta de 1593 líneas de texto. Finalmente, su herramienta de configuración puede usarse vía web, pero no cuenta con un wizard para afinar la configuración.

No se ha encontrado evidencia documental de alguna aplicación de NLPQC a otra base de datos; por lo tanto, queda en duda la afirmación de que NLPQC sea independiente de dominio. En cuanto a configuración, NLPQC tiene un servicio de configuración inicial, el cual obtiene información del esquema de la base de datos a consultar. Esta configuración es parecida a la que efectúa el módulo Generación del diccionario de dominio de nuestra ILNBD (Fig. 2.7), con la diferencia de que NLPQC genera plantillas para las consultas.

Tampoco se ha encontrado evidencia de que NLPQC permita hacer la configuración vía web, ni tampoco que tenga un wizard para afinar la configuración.

Finalmente, se puede concluir que, de acuerdo con la revisión de documentación de otras ILNBDs, no existe ninguna otra interfaz que sea independiente de dominio y que posea un wizard para afinar la configuración para cualquier BD (como el que posee nuestra ILNBD) y que además sea accesible vía web, como se pudo hacer en este proyecto. En la Tabla 2.1 se presenta un resumen de las características de configuración vía web de las ILNBDs más importantes.

Tabla 2.1 Características de configuración vía web de ILNBDs

Interfaz	Independencia de dominio	Config. inicial	Afinación de config.	Config. vía web	Afinación con wizard
English Query	✓	✓	✓	✓	✗
BirdQuest	✗	✗	✗	✗	✗
C-Phrase	✓	✓	✓	✓	✗
NLPQC (Cindi)	?	✓	?	?	✗
Este proyecto	✓	✓	✓	✓	✓

2.2.6 Antecedentes

En el 2014 se terminó un proyecto de tesis denominado Modelo Semánticamente Enriquecido de Bases de Datos para Explotación por Interfaces de Lenguaje Natural en el que se propuso un Diccionario de Información Semántica (DIS) en una ILNBD para mejorar su desempeño [Aguirre, 2014].

En la Figura 2.7 se muestra la arquitectura general de la ILNBD propuesta en el proyecto de tesis mencionado anteriormente, en la cual se puede apreciar dos módulos principales: el proceso de traducción y el proceso de configuración.

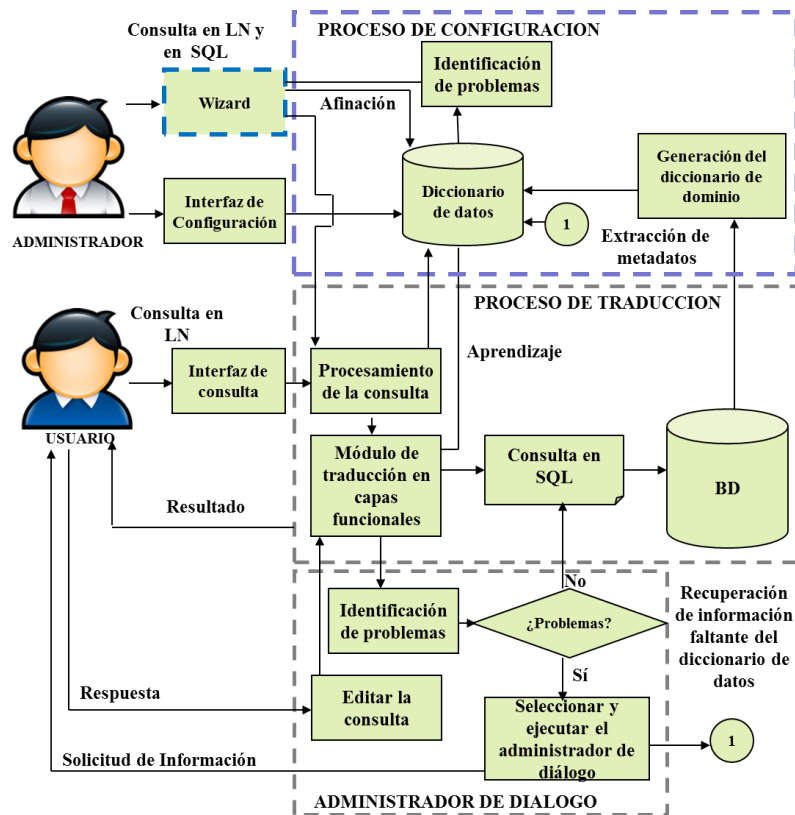


Figura 2.7 Arquitectura general de la ILNBD [Aguirre, 2014]

El proceso de traducción contiene los procesos encargados de realizar la traducción de una consulta en lenguaje natural a una consulta en SQL.

El proceso de configuración contiene los procesos encargados de realizar la configuración del DIS, en donde destacan la generación del diccionario de dominio y la afinación del DIS con ayuda del wizard.

La ILNBD funciona con tres capas que fueron propuestas por Aguirre: Análisis Léxico, Análisis Sintáctico y Análisis Semántico, como se muestra en la Figura 2.8.

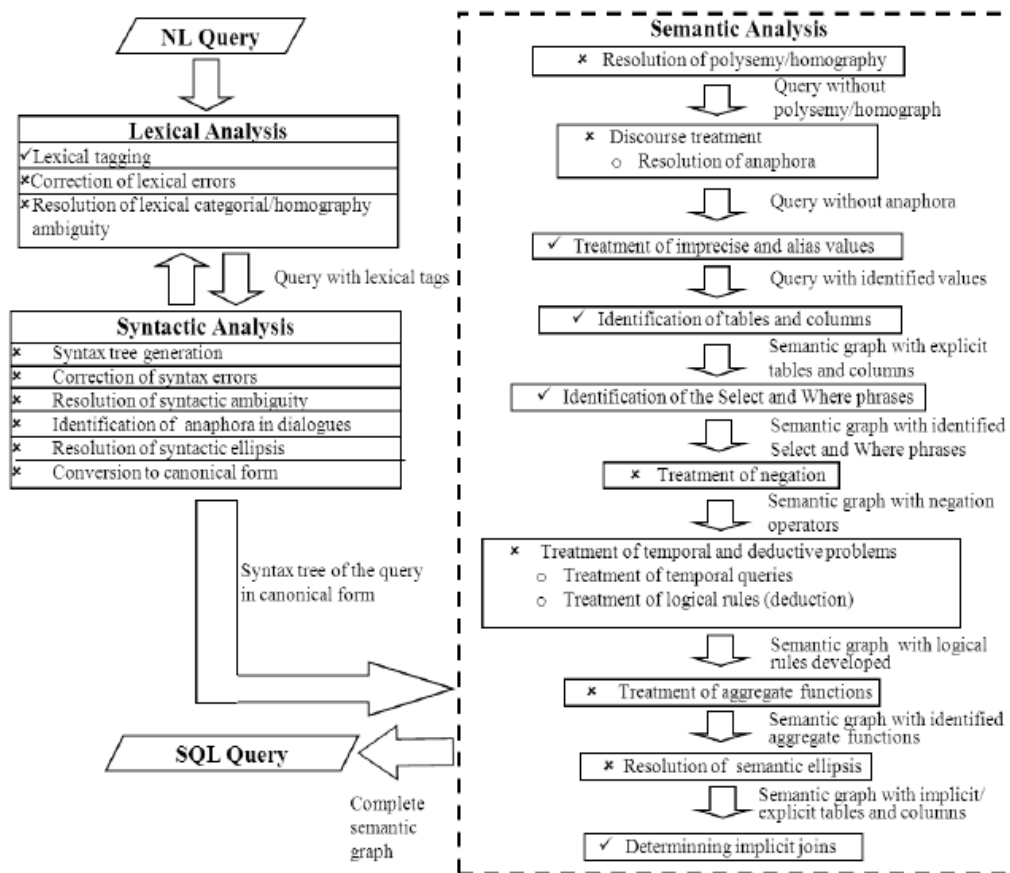


Figura 2.8 Capas de funcionalidad [Aguirre, 2014]

La capa Análisis Léxico permite etiquetar todas las palabras que se encuentren en el lexicón con su categoría gramatical. La capa Análisis Sintáctico permite reducir a una sola categoría gramatical por palabra, también permite ignorar palabras que son irrelevantes. La tercera capa, Análisis Semántico, se encarga de identificar tablas y columnas en base a la información almacenada en el diccionario de información semántica. Esta capa también se encarga de traducir la consulta a SQL.

En la tesis de Aguirre se menciona que “El desempeño de una ILNBD depende en gran medida de su configuración; es decir, cuanto mejor sea la información en el diccionario de datos que describe las relaciones entre los descriptores y las tablas/columnas de la BD y otras propiedades semánticas, mayor será el número de elementos identificados con precisión, lo que aumentará el porcentaje de consultas correctamente contestadas por la interfaz” [Aguirre, 2014]. Para una ILNBD se recomienda que sea independiente de dominio, para que permitan portar la interfaz a otros dominios. Por ende, es recomendable que la ILNBD tenga que ser configurada cada vez que se utilice una base de datos diferente. La persona responsable de configurar la base de datos es el administrador de base de datos (DBA), que es quien tendrá conocimiento de la base de datos y de los lenguajes de consulta de bases de datos.

La ILNBD cuenta con una configuración automática (realizada por el módulo Generación del diccionario de dominio, Figura 2.7), pero desafortunadamente no siempre proporciona una configuración eficaz que sea capaz de responder correctamente las consultas. Por este motivo se desarrolló un asistente (wizard) que permite afinar la configuración. Esto proporciona una gran ventaja, ya que el usuario que utilice este asistente no necesita tener conocimiento de conceptos especializados y mucho menos del funcionamiento de la misma interfaz. Únicamente requiere que el configurador proporcione la consulta en SQL correcta para cada consulta en lenguaje natural que la interfaz no haya podido traducir correctamente.

Capítulo 3

Análisis y solución conceptual del problema

En este capítulo se presentan las arquitecturas de la ILNBD de la versión anterior y de la versión actual, los requisitos de la nueva versión de la ILNBD y la descripción de la arquitectura del control de usuarios. La nueva versión implementada en este proyecto de tesis debe permitir a la ILNBD funcionar en la internet por medio de un navegador de web. También, debe permitir que la ILNBD trabaje con varias bases de datos y además ser accedida por uno o hasta cinco usuarios simultáneamente para formular consultas en lenguaje natural. Además, la nueva versión debe permitir configurar la ILNBD con ayuda del asistente (wizard) de tal manera que permita a los usuarios modificar el diccionario de información semántica de la ILNBD de una manera independiente para cada uno (sin interferir las configuraciones de unos con las de otros). Teniendo en cuenta lo mencionado anteriormente, este proyecto tiene 3 objetivos principales: modificar el wizard para ser usado con un navegador de web, diseñar una nueva arquitectura para la ILNBD e incluir nuevas funcionalidades a la ILNBD.

3.1 Descripción de la arquitectura de la versión anterior de la ILNBD

La ILNBD que se desarrolló en un proyecto de tesis por Aguirre en el 2014 funciona con la arquitectura que se muestra en la Figura 2.7. Esta ILNBD puede ser usada por un solo usuario a la vez, ya que es para una computadora aislada. El usuario tiene la opción de realizar dos procesos principales: formular consultas en lenguaje natural y configurar la ILNBD.

El proceso de consultas en lenguaje natural permite al usuario introducir una consulta en lenguaje natural (en español), donde una vez que el usuario introduzca la consulta en LN en la interfaz, la ILNBD hará el proceso de traducción y retorna un resultado al usuario.

El proceso de configuración que se muestra en la Figura 2.7, permite al usuario generar y editar un diccionario de dominio con ayuda de una interfaz de configuración, además de editar directamente el diccionario de información semántica (DIS) mediante el uso del asistente (wizard).

El proceso de configuración de la ILNBD se puede realizar de tres maneras diferentes: configuración automática (Generación del diccionario de dominio), afinación manual (Interfaz de Configuración) y afinación con el wizard, como se muestra en la Figura 3.1.

La configuración automática o generador de dominio permite generar un nuevo diccionario de información con ayuda de la información que se encuentra en la base de datos a la que se desea formular consultas en lenguaje natural. Para realizar la configuración automática, el usuario tiene que seleccionar la opción generar dominio, después la ILNBD empieza a extraer los metadatos de la base de datos, de donde extrae la información de las tablas, columnas y relaciones que existen.

La afinación manual o edición de dominio permite editar el diccionario de información semántica que la ILNBD esté utilizando. La ILNBD utiliza una interfaz donde muestra la información de las tablas, las columnas, las relaciones, valores alias, valores imprecisos, multicolumna, ordenes (lista, muéstrame y necesito), palabras inútiles, funciones de agregación y vistas. Para realizar la afinación manual el usuario únicamente tiene que seleccionar la opción edición de dominio, modificar la información que requiera y guardarla.

La afinación con el wizard permite editar y eliminar la información del diccionario de información semántica. Para realizar la afinación con el wizard el usuario necesita haber formulado una consulta en lenguaje natural previamente, en donde los datos de entrada que necesita el wizard son la consulta en lenguaje natural, la instrucción en SQL errónea (generada por la ILNBD) y la instrucción en SQL correcta, ésta última introducida por el usuario. Además, el wizard necesita consultar el diccionario de información semántica que esté utilizando la ILNBD en ese momento.

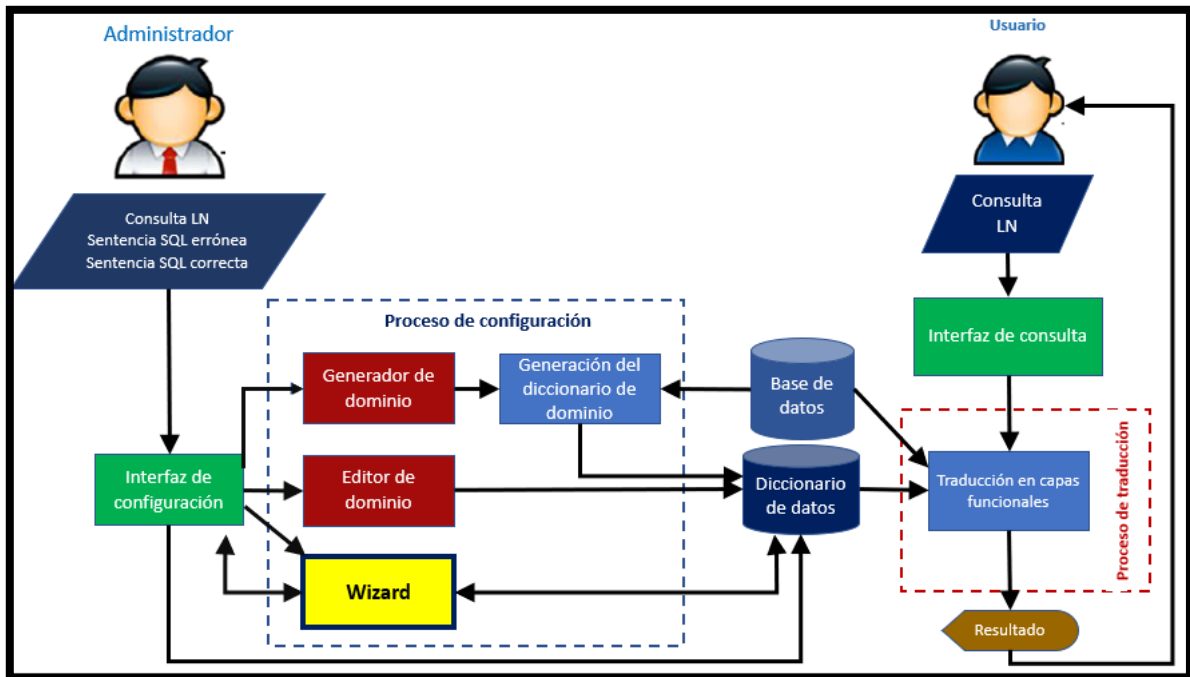


Figura 3.1 Diagrama conceptual de la ILNBD versión de escritorio

3.2 Descripción de la arquitectura de la versión actual para web

Actualmente existe un demo de la ILNBD que está disponible en la dirección <http://nlp.itcm.edu.mx:8080/nli/demo.html>. Esta ILNBD fue implementada utilizando la ILNBD de escritorio. En la Figura 3.2 se muestra la arquitectura que utiliza la ILNBD para web. Esta ILNBD cuenta con la Interfaz de configuración y la Interfaz de consulta. Además en esta arquitectura se encuentran separados los módulos que son de la interfaz hombre-máquina de los que son de procesamiento (cuyo código está en lenguaje de programación Java).

La Interfaz de consulta permite al usuario formular consultas en lenguaje natural de la misma manera que la ILNBD de escritorio, devolviendo una consulta en SQL y mostrando los datos al ejecutar esa consulta.

Cabe mencionar que en la Interfaz de configuración existen limitaciones, solamente permite ver las tablas, columnas, relaciones del diccionario de información semántica, pero no permite hacer cambios. Además, esta ILNBD no tiene disponible la configuración automática o generación de dominio y la afinación con el wizard, con que cuenta la ILNBD de escritorio descrita en la Sección 3.1. Además, no permite seleccionar una base de datos específica a utilizar, por tal motivo la ILNBD utiliza únicamente la base de datos ATIS.

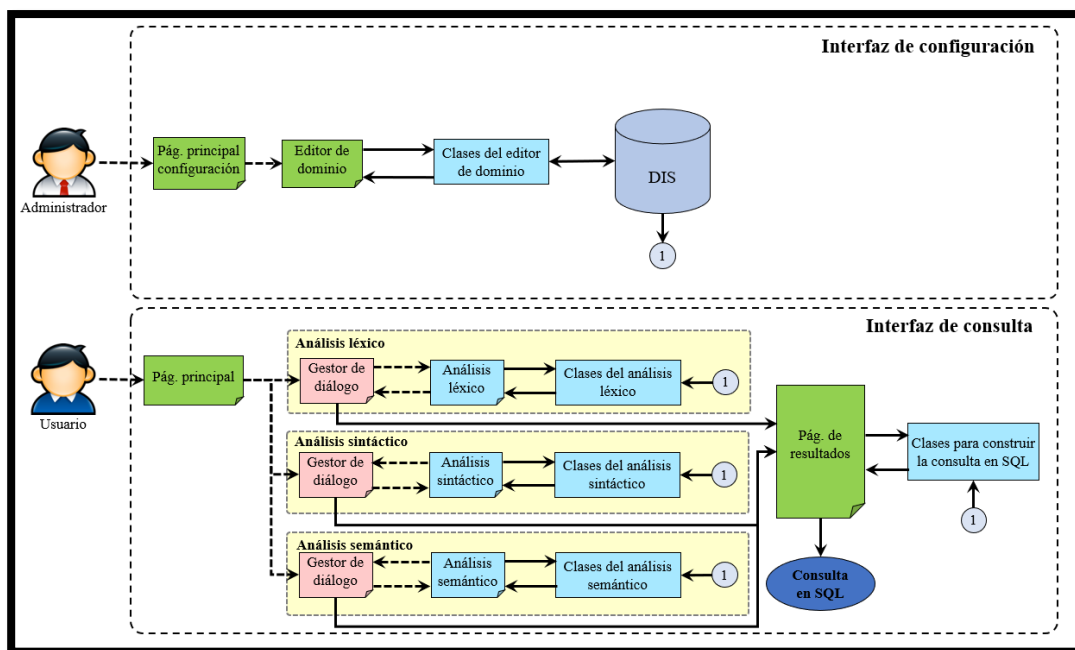


Figura 3.2 Arquitectura de la ILNBD versión para web (demo)

3.3 Requisitos de la nueva versión de la ILNBD

La nueva versión de la ILNBD debe satisfacer los siguientes requisitos funcionales:

- Que pueda ser accedida a través de Internet.
- Que tenga un sistema de administración de bases de datos que permita formular consultas a través de Internet.
- Que sea multiusuario.
- Que tenga control de usuarios (registro y sesión de usuarios).
- Que permita efectuar las siguientes configuraciones: seleccionar base de datos a utilizar, seleccionar/eliminar diccionario de información semántica a utilizar, configuración automática, afinación manual, afinación con el wizard.
- Que permita incluir nuevas funcionalidades a la ILNBD de una manera más fácil.

3.4 Descripción de la nueva arquitectura de la ILNBD

La nueva ILNBD para web tiene la arquitectura que se muestra en la Figura 3.3. Esta arquitectura tiene dividido los submódulos que corresponden a la interfaz hombre-máquina de los submódulos que realizan el procesamiento. Además, cuenta con los dos módulos principales que tiene la ILNBD de escritorio.

El primero módulo es la Interfaz de configuración en el cual se integraron los submódulos de control de usuarios, configuración automática (Generar dominio), afinación manual (Editor de dominio) y afinación con el wizard. A continuación se describe de manera abreviada cada uno de estos submódulos.

El submódulo de control de usuarios permite crear o iniciar sesión en la ILNBD y así poder utilizar todas las funciones que ofrece la interfaz de configuración.

El submódulo de configuración automática (Generar dominio) permite generar un nuevo diccionario de información semántica de acuerdo con la base de datos que el usuario esté utilizando en ese momento.

El submódulo de afinación manual (Editor de dominio) permite editar el diccionario de información semántica.

El submódulo de consulta permite formular consultas en lenguaje natural, además tiene la opción de utilizar la afinación con el wizard.

El segundo módulo, Interfaz de consulta, tiene la misma estructura que se presentó en la arquitectura de la ILNBD versión demo que se muestra en la Figura 3.2. Este módulo permite al usuario formular consultas en lenguaje natural.

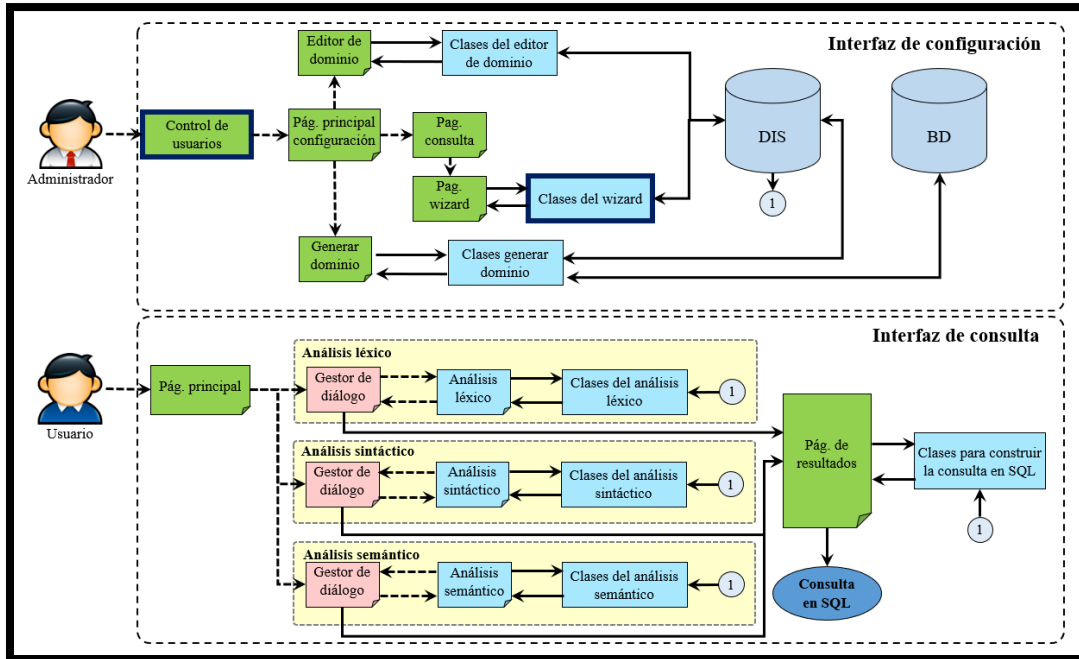


Figura 3.3 Arquitectura de la ILNBD versión para web

3.5 Descripción de la arquitectura del control de usuarios

Para que la ILNBD tenga un control de los usuarios que acceden a ella, se propuso un submódulo de control de usuarios. En la arquitectura presentada en la Figura 3.3, el módulo Interfaz de configuración cuenta con el submódulo Control de usuarios. En este submódulo se realizan dos procesos importantes para controlar el acceso de los usuarios, los cuales se describen enseguida.

El proceso de registro de un nuevo usuario el cual permite a un nuevo usuario registrarse y obtener una cuenta de acceso a la ILNBD. Para realizar el registro de un nuevo usuario, cada usuario que desee acceder a la ILNBD debe registrarse en la página de registro. Una vez llenados los campos requeridos, se envían los datos para su validación, en donde se comprueban si son válidos. En caso afirmativo, se almacenan en una tabla *usuarios* de una base de datos y regresa a la página principal para iniciar una sesión; en caso contrario, la página de registro retornará un mensaje de error al usuario.

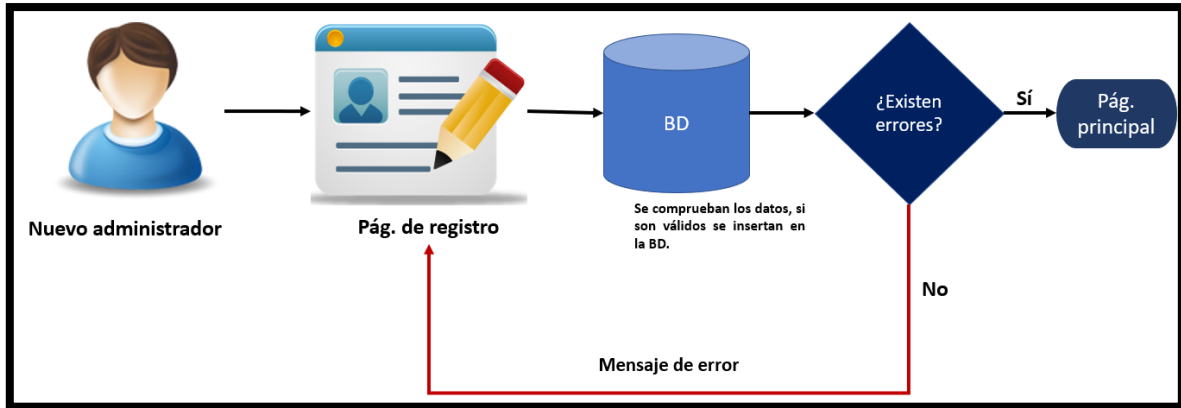


Figura 3.4 Arquitectura del proceso de registro de un nuevo usuario

El proceso de acceso de usuarios permite a un usuario iniciar una sesión en la ILNBD, para utilizar todas sus funciones disponibles.

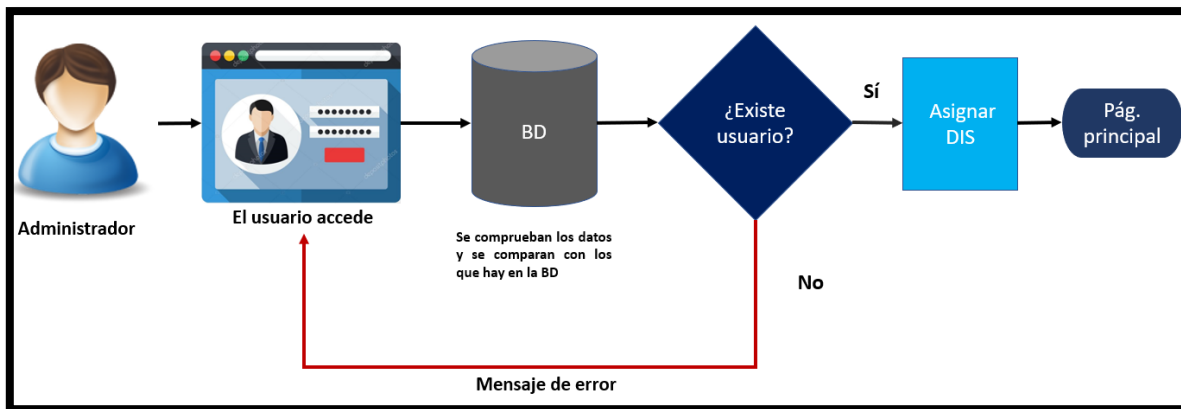


Figura 3.5 Arquitectura del proceso de acceso de usuarios

Para realizar el acceso de usuarios, los usuarios clientes que deseen utilizar la ILNBD de manera completa (es decir, utilizar todas las funciones disponibles con que cuenta la ILNBD), deberán registrarse previamente, con lo cual los usuarios obtendrán sus claves de acceso (usuario y contraseña). Para cada usuario, al iniciar sesión y llenar los campos correspondientes para el acceso, la interfaz comprobará si este usuario existe. En caso afirmativo, se le asignará el acceso al diccionario de información semántica (DIS) correspondiente, y se direccionará a la página principal de la ILNBD; en caso contrario, retornará un mensaje de error y regresará a la página de iniciar sesión.

Capítulo 4

Desarrollo de la ILNBD para web

En este capítulo se describe lo más importante del desarrollo de la ILNBD para web. Específicamente, el capítulo contiene una descripción del servidor de web y del sistema de administración de bases de datos (SABD), así como del procedimiento que se utilizó para migrar las bases de datos de un SABD a otro. Además, se incluye la descripción del control de los usuarios para el acceso a la ILNBD para web, y la descripción de la implementación de la interfaz de configuración, en donde destaca la configuración automática, afinación manual y la afinación con el wizard. El capítulo contiene además, la descripción de la implementación de la página de consulta, así como los códigos de programa utilizados. Por último, se presenta la jerarquía de las clases con las que está implementada la nueva ILNBD para web.

4.1 Servidor de web

Para que la ILNBD para web funcione, es necesario que la aplicación de web esté hospedada en un servidor de web que permita tener acceso a través de internet o de una intranet mediante un navegador. Para la elección del servidor fue importante tomar en cuenta los dos antecedentes de este proyecto: una ILNBD que se implementó en Java y una ILNBD para web versión demo, la cual se implementó utilizando código en Java, JavaScript y HTML en donde destaca el uso de la tecnología JSP (Java Server Pages). Para seleccionar el servidor que utilizaría la ILNBD para web, y teniendo como antecedente la ILNBD de escritorio que se desarrolló en el ITCM por Aguirre, se llegó a la conclusión de utilizar el servidor de web Apache Tomcat. Considerando los antecedentes mencionados anteriormente, a continuación se muestran las ventajas que proporciona utilizar Apache Tomcat en este proyecto:

- La reutilización del código en Java con el que cuenta la ILNBD de escritorio.
- Minimizar tiempo de desarrollo de la ILNBD para web.
- Usar como guía el código con que cuenta la ILNBD para web versión demo.

Tomando en cuenta lo anterior se instaló el servidor de web Apache Tomcat 7.0.77 para Windows con una arquitectura de 64 bits. Éste se descargó de la página *The Apache*

software foundation, a la cual se puede acceder en la siguiente dirección <http://tomcat.apache.org/>.

La instalación de Apache Tomcat en Windows se hace usando el instalador de Windows que nos proporciona al descargarlo de la página. Su interfaz y funcionalidad es similar a la de otros instaladores basados en asistentes.

4.2 Sistema de administración de bases de datos para la ILNDB para web

Con el fin de que la ILNDB para web conteste las consultas, necesita extraer información de una base de datos. La ILNDB de escritorio utilizaba el SABD Microsoft Access. Este SABD no proporciona un buen soporte en la formulación de consultas por internet. Por lo tanto, se optó por seleccionar otro SABD que permita formular consultas a través de internet, y además, que sea lo más apegado a los estándares de SQL. Considerando estos objetivos, se decidió utilizar el SABD PostgreSQL, el cual satisface esos dos requisitos fundamentales que hicieron posible a la implementación de la ILNDB para web.

Se instaló el SABD PostgreSQL 9.6 para Windows de 64 bits. Éste se descargó de la página oficial de PostgreSQL, a la cual se puede acceder en la siguiente dirección <https://www.postgresql.org/download/>.

La instalación de PostgreSQL se efectuó usando el instalador **.exe** para Windows de 64 bits. Este instalador nos proporciona un asistente para llevar a cabo una serie de pasos para realizar la instalación. Durante el proceso se hace una configuración de usuario, contraseña y el número de puerto.

4.3 Migración de bases de datos de Microsoft Access a PostgreSQL

En esta sección se describe la migración de las bases de datos que son utilizadas por la ILNDB de escritorio. La migración se hizo del SABD Microsoft Access al SABD PostgreSQL. Para realizar esta migración se descargó el software llamado Access to PostgreSQL. Dicho software es un pequeño programa que convierte las bases de datos de Microsoft Access a PostgreSQL, y además, permite enviar las tablas de Access directamente a una base de datos PostgreSQL con ayuda del conector ODBC para la base de datos PostgreSQL. Access to PostgreSQL se puede descargar de la siguiente dirección <http://www.bullzip.com/products/a2p/info.php>.

A continuación, se mostrará un ejemplo de la migración de la base de datos BDATIS.mdb, utilizando el software Access to PostgreSQL. En cada figura se muestran mediante números las opciones que se utilizaron para la migración. Al ejecutar el software nos muestra una interfaz con algunas ventanas.

1. **File name:** En la primera ventana (Figura 4.1), se selecciona la dirección en donde se encuentra el archivo que se desea emigrar. Este archivo debe ser de tipo **.mdb**, ya que

es el tipo de archivo usado para las bases de datos de Microsoft Access. En este caso se utilizó el archivo BDATIS.mdb.

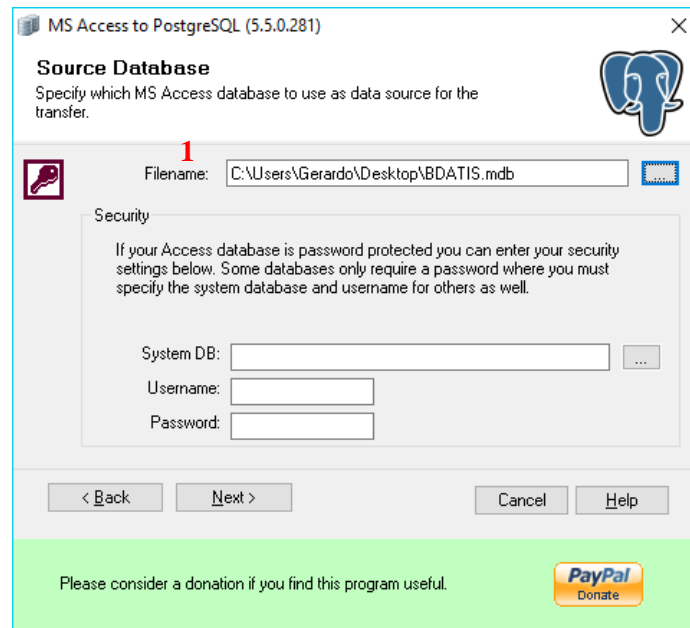


Figura 4.1 Selección de base de datos de origen

En la siguiente ventana (Figura 4.2) se especifica la configuración de la base de datos de destino.

1. **Direct transfer:** Se selecciona esta opción para enviar las tablas de Access directamente a una base de datos PostgreSQL.
2. **PostgreSQL Connection Options:** Se configura la conexión de PostgreSQL, para lo cual se tiene que ingresar el servidor, el puerto, el mantenimiento de la BD, el usuario y la contraseña.
3. **PostgreSQL Destination Database:** Se ingresa el nombre de la base de datos de destino, en este caso la base de datos ATIS.

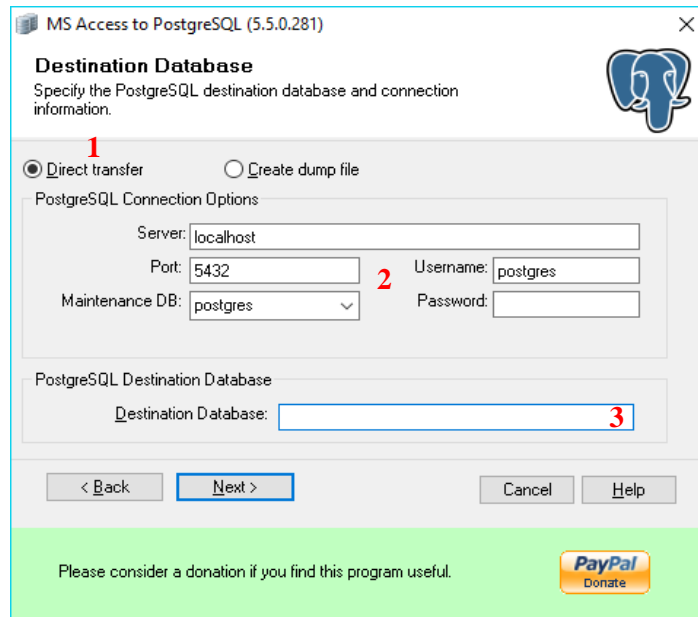


Figura 4.2 Configuración del destino de la base de datos

En la ventana de la Figura 4.3 se seleccionan las tablas que se desean emigrar a la base de datos de destino.

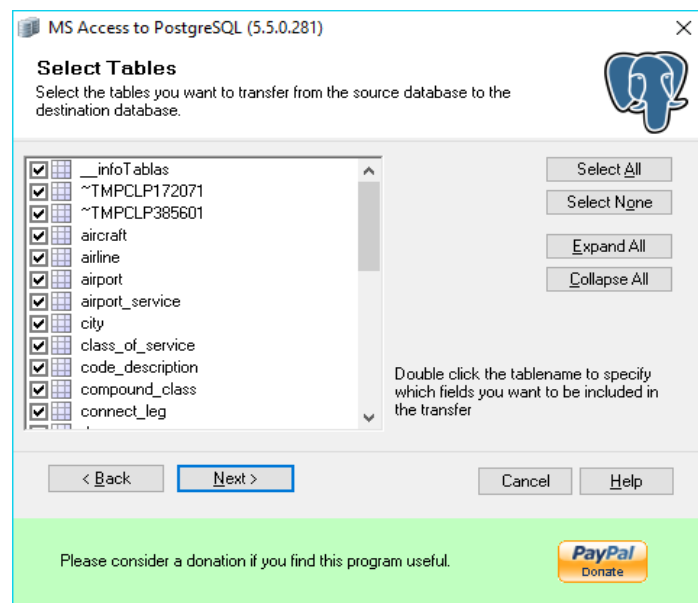


Figura 4.3 Selección de tablas a emigrar

En la Figura 4.4 se muestran algunas de las opciones que pueden afectar la transferencia a la base de datos de destino.

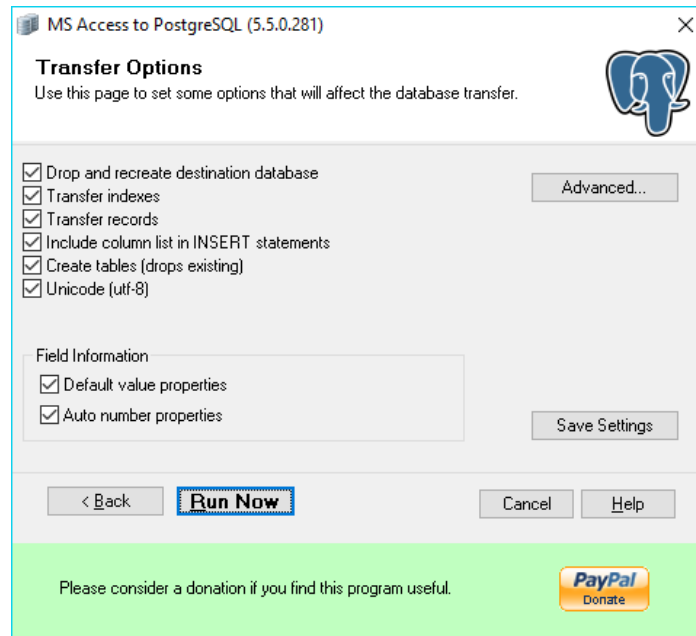


Figura 4.4 Opciones de transferencia

4.4 Control de usuarios de la ILNBD

En esta sección se describe la lógica general de una de las funciones principales con la que debe contar la ILNBD para web. Esta función es que sea multiusuario, y que cada usuario tenga la opción de generar y editar el diccionario de información semántica de manera independiente, ya sea usando la configuración automática, afinación manual o la afinación con el wizard.

La ILNBD para web debe permitir a varios usuarios usarla (inclusive modificar el diccionario) de manera simultánea, sin interferencia de unos con otros.

Se propusieron dos tipos de usuarios: usuario invitado (no identificado) y usuario identificado. El módulo de control de usuarios está enfocado a los usuarios identificados; es decir, aquellos usuarios que deseen utilizar todas las funciones que ofrece la ILNBD para web.

Para realizar el control de los usuarios invitados dentro de la ILNBD para web, se decidió utilizar una base de datos auxiliar, en donde se creó una tabla con el nombre *usuarios*. Esta tabla se utiliza en el proceso de registro de un nuevo usuario y en el proceso de inicio de sesión. En la Tabla 4.1 se muestra la información de las columnas de esta tabla.

Tabla 4.1 Columnas de la tabla usuarios

Columnas	Tipo de dato	Descripción
usuario	Carácter variable	Usuario de acceso
contraseña	Carácter variable	Contraseña de acceso
nombre	Carácter variable	Nombre del usuario
primer_apellido	Carácter variable	Apellido del usuario
segundo_apellido	Carácter variable	Apellido del usuario
bds	Carácter variable	Nombres de bases de datos
correo_electronico	Carácter variable	E-mail del usuario
fecha_conexión	Carácter variable	Última fecha de conexión
PRIMARY KEY: usuario		

La tabla *usuarios* contiene la columna *bds*, la cual sirve para almacenar todos los nombres de las bases de datos que corresponden a los diccionarios de información semántica que el usuario ha generado en la ILNBD para web. Cada vez que el usuario genera un nuevo DIS, esta columna se actualiza. Para realizar la actualización y no perder información, se decidió que la columna *bds* almacenara una cadena que tenga la estructura de un objeto JSON.

A continuación se muestra un ejemplo del contenido de la columna *bds*:

```
{"BDATIS":["DIS_Experimentacion1","DIS_Config_Wizard","Diccionario_Datos_p"],"Geobase":[]}
```

En el ejemplo anterior se muestra que el usuario al que le corresponde este registro ha creado tres diccionarios de información semántica para la base de datos BDATIS, y ningún diccionario de información semántica para la base de datos Geobase.

La Figura 4.5 muestra el proceso que se efectúa para validar el acceso de un usuario a la ILNBD para web, además de la forma en que se asigna la base de datos y su respectivo diccionario de información semántica.

Para los usuarios invitados (no identificados), únicamente acceden directamente a la página principal de la ILNBD para web, pero con funciones limitadas.

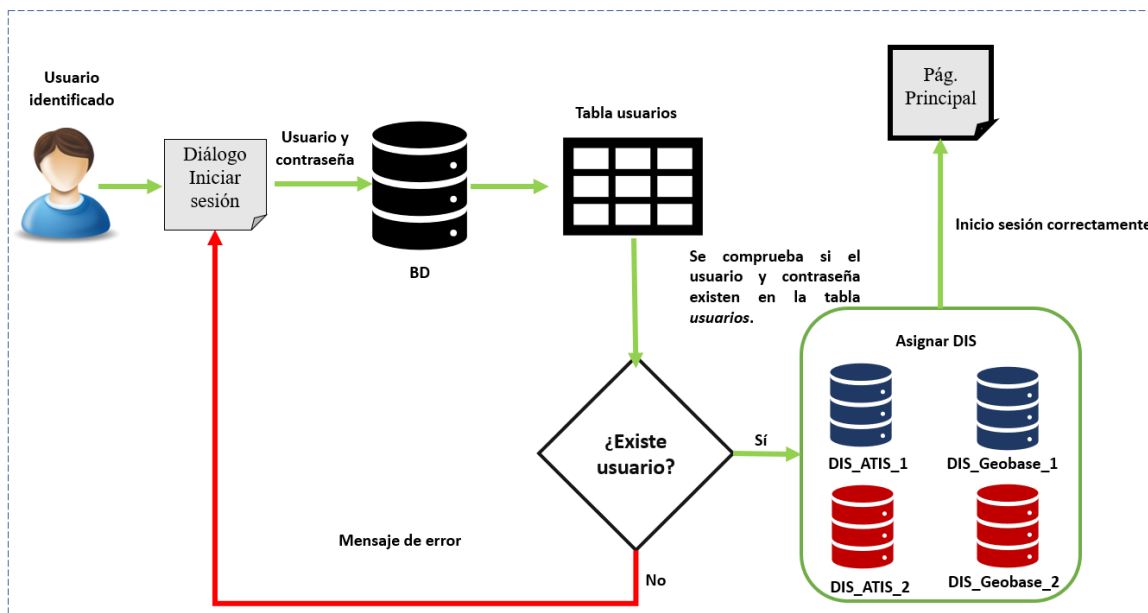


Figura 4.5 Diagrama conceptual de la asignación de acceso de un usuario a su diccionario de información semántica

4.4.1 Tipos de usuarios

Como se describió al principio de la Sección 4.4, la ILNBD para web utiliza un control de usuarios para manejar el acceso a la ILNBD para web y las funciones con las que cuenta. Se menciona que la ILNBD para web cuenta con dos tipos de usuarios: usuario invitado y usuario identificado. A continuación, se describe de manera detallada qué funciones ofrece la ILNBD a cada tipo de usuarios.

Usuario invitado (no identificado): Es el permiso que tiene cualquier persona que accede sin mediar identificación dentro de la ILNBD para web. Estos tipos de usuarios únicamente tienen permitido formular consultas en LN, y seleccionar qué base de datos van a utilizar para formular consultas en LN.

Usuario identificado: Este usuario tiene permitido utilizar todas las funciones con las que cuenta la ILNBD para web: configuración automática, afinación manual, afinación con el wizard, formulación de consultas en LN, selección de la base de datos con la que desea formular consultas en LN, selección y eliminación de un diccionario de información semántica, y procesamiento por lote de consultas.

4.5 Implementación del control de usuarios

En la Sección 4.4 se describe de forma general cómo se hizo el control de los usuarios dentro de la ILNBD para web, además se mencionaron dos tipos de usuarios. A continuación, se describe la manera en que se implementó el acceso para cada uno de estos tipos de usuarios.

4.5.1 Implementación del registro de un nuevo usuario

Para el proceso de registro de un nuevo usuario, éste tiene que acceder al formulario de registro, en donde deberá proporcionar los siguientes datos: usuario, contraseña, nombre, primer apellido, segundo apellido, correo electrónico. Una vez que se active el evento del botón "Registrar", llama a una función *\$.ajax({})* de la biblioteca JQuery, que forma parte de la página *Login_Registrar.jsp*. Esta función envía una solicitud al servidor de web. A continuación se presenta el código para esta función.

Inicio

```
$.ajax({
    // Envía datos para ser procesados a un recurso específico
    type ← "Post"
    // url para la petición
    url ← "../Registro_usuario/Registro_nuevo_invitado.jsp"
    // Datos enviados al servidor
    data ← { usuario←usuario,
            contraseña←contraseña,
            correo←correo,
            nombre←nombre,
            primer_apellido←prim_apellido,
            segundo_apellido←seg_apellido
          }
    // Función que se ejecuta cuando la función es exitosa
    success: function(data){
        //Muestra mensaje
        mensaje←Se registro exitosamente.
    }
    // Función que se ejecuta cuando la función no es exitosa
    Error: function(){
        mensaje ←Error al registrar.
    }
}
```

Fin

Los datos enviados son recibidos en el servidor de web por medio de la página *Registro_nuevo_invitado.jsp*. Esta página contiene código en Java para almacenar la información en la tabla *usuarios* de la base de datos en PostgreSQL, siempre y cuando no exista el usuario.

Inicio

```
Usuario ← getParameter("usuario")
Contraseña ← getParameter("contraseña")
Correo_electronico ← getParameter("correo")
Nombre ← getParameter("nombre")
primer_apellido ← getParameter("primer_apellido")
segundo_apellido ← getParameter("segundo_apellido")
DBS ← ""
int año ← fecha.get(Calendar.YEAR);
int mes ← fecha.get(Calendar.MONTH);
int dia ← fecha.get(Calendar.DAY_OF_MONTH);
int hora ← fecha.get(Calendar.HOUR);
int min ← fecha.get(Calendar.MINUTE);
int seg ← fecha.get(Calendar.SECOND);

fecha_conexion ← java.sql.Timestamp.valueOf(año+"-"+mes+"-"+dia+"
"+hora+": "+min+": "+seg);
```

Try:

```
conexión ← con.cargarBD(BD,usuarioBD,contraseña, 127.0.0.1)
pst ← conexión.prepareStatement("INSERT INTO usuarios
VALUES(?,MD5(?,?,?,?,?,?))")
```

```
    pst.setString(1,usuario);
    pst.setString(2,contraseña);
    pst.setString(3,nombre);
    pst.setString(4,primer_apellido);
    pst.setString(5,segundo_apellido);
    pst.setString(6,DBs);
    pst.setString(7,correo_electronico);
    pst.setTimestamp(8,fecha_conexion);
    pst.executeUpdate();
    conexión.close()
```

```
    return "Su registro fue un exito."
```

Catch: Exception e

```
    return "Error de registro."
```

Fin

En el código anterior, primero se obtienen los valores de los parámetros de la solicitud y cada uno de ellos se asigna a su variable correspondiente. Enseguida, se obtiene la fecha y hora del sistema, y por último, se hace una conexión a la base de datos y se ejecuta una instrucción INSERT en la tabla *usuarios* almacenando los valores de las variables. Si los

valores se almacenan correctamente, se envía un mensaje de registro exitoso; en caso contrario, se envía un mensaje de error de registro.

4.5.2 Implementación del acceso de un usuario identificado

Para que un usuario identificado pueda acceder a la ILNBD para web debe registrarse previamente. Una vez que el usuario identificado tenga su usuario y contraseña, tendrá que iniciar sesión, y al presionar el botón "Aceptar" procede a ejecutar una petición, la cual se envía con ayuda de una función `$.ajax({})` de la biblioteca JQuery. A esta función se le asignaron los siguientes parámetros: *type*, *url*, *data*, *success* y *Error*, los cuales se muestran en el código que se describe a continuación.

Inicio

```
$.ajax({
  // Envía datos para ser procesados a un recurso específico
  type ← "Post"
  // url para la petición
  url ← ../Registro_usuario/Logear_usuario.jsp"
  // Datos enviados al servidor
  data ← { Usuario←usuario , Contraseña←contraseña }
  // Función que se ejecuta cuando la función es exitosa
  success: function(data){
    //Muestra mensaje de bienvenida
    mensaje←Bienvenido
    direccionar ← ../Traductor/index.html
  }
  // Función que se ejecuta cuando la función no es exitosa
  Error: function(){
    mensaje ←Error al iniciar sesión
  }
}
```

Fin

El usuario y contraseña se envían junto con la solicitud, utilizando la URL `../Registro_usuario/Logear_usuario.jsp` que apunta a una página JSP que se encuentra en el servidor. A continuación, se muestra lo más importante de la página `Logear_usuario.jsp`.

Inicio

```
Usuario ← getParameter("usuario")
Contraseña ← getParameter("contraseña")
Try:
  obj_control_usuario cu ← new obj_control_usuario()
  obj_usuario us ← cu.logear (Usuario,Contraseña)
  if (us != null)

    HttpSession sesión ← getSession(true)
    sesión.setAttribute ← us.getUsuario()
    sesión.setAttribute ← us.getContraseña()
    sesión.setAttribute ← us.getNombre()+" "+us.getPrimer_apellido()+"
    "+us.getSegundo_apellido()
    sesión.setAttribute ← us.getCorreoelectronico()

    JSONObject jsusuario ← new JSONObject()
    jsusuario.put ← ("USER", us.getUsuario())
    jsusuario.put ← ("CONTRASEÑA", us.getContraseña())
    jsusuario.put ← ("Nombre_completo", us.getNombre()+"
    "+us.getPrimer_apellido()+" "+us.getSegundo_apellido())
    jsusuario.put ← ("Correo", us.getCorreo_electronico())

    return jsusuario
  else
    return "No se pudo iniciar sesión error en usuario o/ contraseña."

Catch: Exception e

  return "No se pudo iniciar sesión error en usuario o/ contraseña."
```

Fin

Si el usuario y contraseña son válidos, se le asigna la base de datos BDATIS y su correspondiente diccionario de información semántica con el nombre *diccionario_de_datos_BDATIS_v0*. Este DIS se le asigna a todos los tipos de usuarios para que utilicen la ILNBD para web.

4.5.3 Implementación del acceso de un usuario invitado (no identificado)

Para el acceso de un usuario invitado, se manejó una página principal en donde se aprecian dos opciones principales: "Interfaz de lenguaje natural v1" e "Interfaz de lenguaje natural v2". El usuario invitado deberá seleccionar la opción "Interfaz de lenguaje natural v1". Esta opción direccionará a la página inicial de la ILNBD. En esta página inicial no será necesario iniciar sesión como cuando se elige la opción "Interfaz de lenguaje natural v2".

4.6 Implementación de la página de consulta

En esta sección se describe la implementación de la página de consulta, en donde destaca la implementación del proceso de consulta y el desarrollo del diseño de la página de consulta.

Para implementar el proceso de consulta de la ILNBD para web, primeramente fue necesario encontrar todas las funciones que son necesarias para llevar a cabo el proceso de consulta en la ILNBD de escritorio. El método principal que ejecuta el proceso de consulta dentro de la ILNBD de escritorio es el método *botonConsultarActionPerformed()*. Este método maneja el evento del botón "Consultar", y dentro de este método se localizaron las variables, métodos que hacen conexiones a bases de datos y otros métodos que se utilizan para este proceso. Se creó una clase *Conexión* en donde se incorporó el método principal *consultar()* con unas ligeras modificaciones para que funcione en la nueva versión de la ILNBD. A continuación se describe cada una de las modificaciones que se hicieron.

La primera modificación que se efectuó fue que el método principal retornara un objeto de JSON. Se utilizó el objeto JSON ya que es un formato ligero de intercambio de datos; es muy utilizado para enviar y recibir información dentro de aplicaciones de web. Enseguida se muestra el código del objeto.

```
JSONObject js = new JSONObject();
js.put("consulta",consultaSQL + "<br><br>");
js.put("resultado",res);
double tiempoFinalDatos = (System.currentTimeMillis() - tiempoInicialDatos) * 0.001;
js.put("tiempo1", "<b>Translation time:</b> <i>" + String.format("%.3f",
tiempoTotalFinal) + "</i> seconds.<br><br>");
js.put("tiempo2", "<b>Database server response time:</b> <i>" + String.format("%.3f",
tiempoFinalDatos) + "</i> seconds.<br><br>");
js.put("objconsulta",tmp_consulta);
js.put("objproblema1",tmp_problemas1);

js.put("objproblema2",tmp_problemas2);
js.put("Stringconsultagenerada",consultaGenerada);
js.put("consultado",Consultado);
```

Uno de los inconvenientes que se presentó en el envío del objeto JSON, es que al momento de almacenar un objeto y ser enviado, este objeto se enviaba de manera incorrecta. Para resolver este problema se utilizó la biblioteca xstream 1.4.10. Esta biblioteca convierte un objeto a formato XML retornando en una cadena. Esta biblioteca se utilizó para convertir a XML el objeto *consulta* y el objeto *problema*.

A continuación se muestra un ejemplo del uso de la biblioteca XStream para convertir el objeto *consulta* a XML:

```
XStream xstream_consulta=new XStream(new JettisonMappedXmlDriver() );  
String tmp_consulta=xstream_consulta.toXML(consulta);
```

En el ejemplo anterior se muestra el uso de la biblioteca XStream para convertir un objeto *consulta* a XML. El ejemplo muestra la creación de un objeto XStream llamado *stream_consulta*, el cual se inicializa con el parámetro para convertir a XML *new JettisonMappedXmlDriver()*. La cadena *tmp_consulta* almacena la cadena que retorna el método *toXML(consulta)*.

Otro de los métodos que fue necesario modificar es el método *cargaBD()*, que permite iniciar una conexión a una base de datos. Únicamente se modificaron dos líneas que corresponden a la dirección de conexión y el conector del SABD. Enseguida se muestran las dos líneas mencionadas:

```
String url = "jdbc:postgresql://127.0.0.1/";  
Class.forName("org.postgresql.Driver");
```

Se creó una página *Consulta.jsp* con el cual primero se obtienen los valores de los parámetros de la solicitud, y después se crea un objeto de la clase *Conexión* que hace el llamado del método *consultar()*. Este método retorna un objeto JSON donde se encuentra almacenado el resultado del proceso de consulta. Enseguida se muestra el código para el proceso de consulta.

Inicio

```
consultaLN←request.getParameter("consultaLN");  
BD ←request.getParameter("BD");  
DIS←request.getParameter("DIS_");  
Conexión con ← new Conexión(BD,DIS);  
JSONObject js ←con.consultar(consultaLN);  
return js
```

Fin

La página *Consulta.jsp* se usa cuando el usuario presiona el botón "Consultar" dentro de la página de consulta, lo cual genera una petición a esta página. Esta petición se envía con ayuda de una función *\$.ajax({})* que se encuentra dentro del evento del botón "Consultar". A esta función se le asignaron los parámetros (*type, url, data, success, Error*) que se encuentran en el código que se muestra a continuación.

Inicio

```
$.ajax({  
  // Envía datos para ser procesados a un recurso específico  
  type ← "Post"  
  // url para la petición  
  url ← Consulta.jsp"  
  // Datos enviados al servidor  
  data ← { consultaLN ← consulta, BD ← BASE_DATOS,  
          DIS_ ← DICCIONARIO_INFO_SEMANTICA }  
  // Función que se ejecuta cuando la función es exitosa  
  success: function(data){  
    //Se muestra el resultado en la página de consulta.  
  }  
  // Función que se ejecuta cuando la función no es exitosa  
  Error: function(){  
    mensaje ← Error  
  }  
}
```

Fin

Para el diseño de la página de consulta se utilizaron archivos de HTML conservando el mismo diseño que tiene la ILNBD de escritorio. Además, se realizaron dos diseños diferentes para la página de consulta de acuerdo con el tipo de usuario. Se realizó un diseño para un usuario invitado (ver Figura 4.6) y uno para un usuario identificado (ver Figura 4.7 y Figura 4.8).

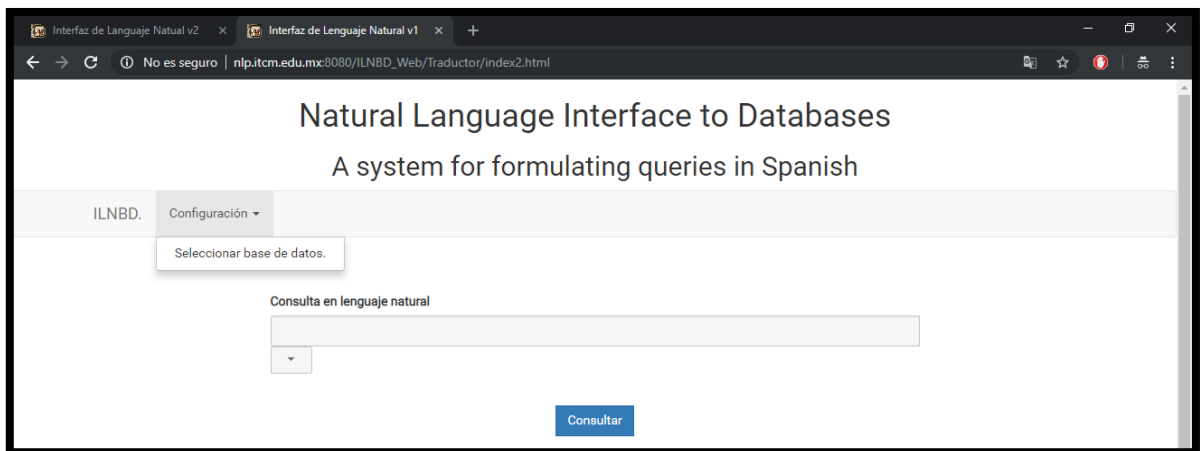


Figura 4.6 Página de consulta para un usuario invitado.

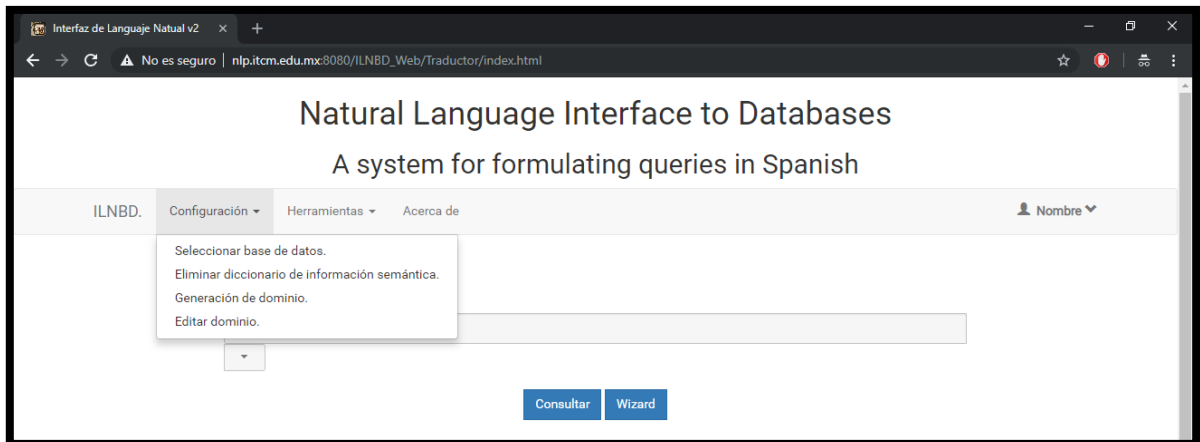


Figura 4.7 Página de consulta para un usuario identificado con opciones de configuración

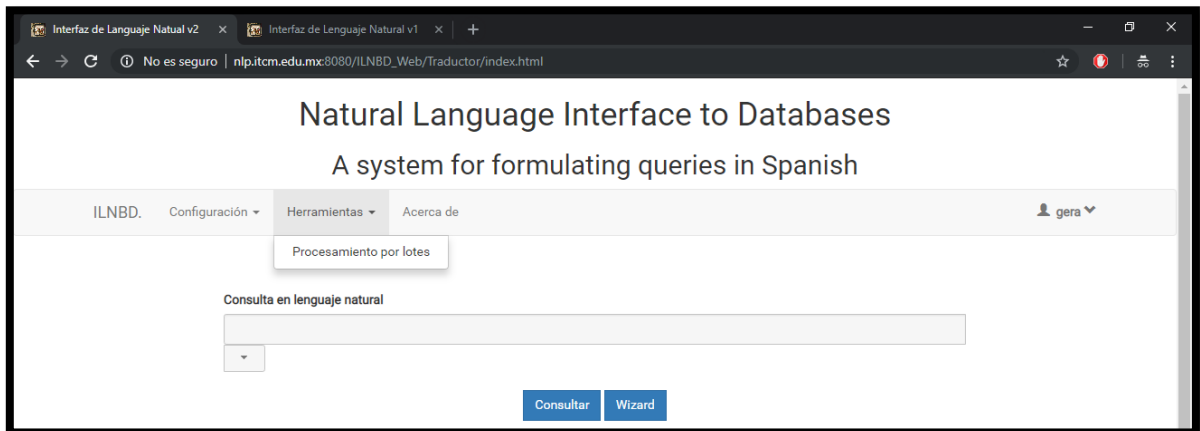


Figura 4.8 Página de consulta para un usuario identificado con opciones de herramientas

4.7 Implementación del editor de dominio

Para la implementación del editor de dominio, se comenzó por localizar el código en la ILNBD de escritorio que permite realizar este proceso. Dentro de la ILNBD para web, este proceso es llamado en la página de consulta, en el menú de configuración. La opción "Editar dominio" ejecuta un evento *click()*. Este método abre una nueva ventana de HTML en donde se carga una tabla de HTML con pestañas de navegación que contiene información de las tablas, columnas, relaciones, valores alias y valores imprecisos del diccionario de información semántica. Esta ventana se abre con ayuda del método *Window.open()* que proporciona el lenguaje JavaScript. Este método recibe como parámetro la URL de la página *editorDominio.jsp*. Esta página JSP ejecuta el código de HTML y el código en Java para cargar la información de las tablas.

Para obtener la información de las tablas correspondientes del diccionario de información semántica, se utilizó la clase *meta.java* en donde se crearon los métodos encargados de extraer la información de las tablas. Estos métodos retornan un objeto

ResultSet, el cual almacena la información extraída de la ejecución de la consulta que realiza cada método. El siguiente código muestra el proceso para obtener la información para cada una de las tablas del DIS, que se muestran en la página del editor de dominio.

Inicio

```
1 meta m ← new meta("postgres","nlipostgres","127.0.0.1");
2 Refactor r ← new Refactor();
3 ResultSet rs ← m.getTablas(urlDicBD);
4 nRows ← m.getNumRows(rs)
5 Num_cols ← getMetaData().getColumnCount()
6 Nombre ← ""
7 Valor ← ""
8 rNombre ← ""
9 For i ← 1 hasta i ≤ Num_cols hacer
10   Nombre ← rs.getMetaData().getColumnName(i)
11   rNombre ← r.renameColumn(Nombre);
12   If getNumRows(rs) > 0
13     Valor ← r.renameValue(rs.getString(Nombre))
14   Else
15     Valor ← ""
16
17   Mostrar la información en una tabla
18 Fin For
```

Fin

4.8 Implementación para la generación de dominio

En esta sección se describe la implementación de la opción que permite generar un nuevo diccionario de información semántica, llamada configuración automática. Esta opción forma parte de la interfaz de configuración de la ILNBD para web.

Para realizar la implementación se procedió de la misma manera que se menciona en la Sección 4.6; es decir, se comenzó encontrando las funciones que se utilizan en la ILNBD de escritorio que permiten realizar el proceso de configuración automática.

En este proceso se comenzó con la codificación para validar si existe algún diccionario de información semántica disponible para la base de datos seleccionada. Esto se hace por medio del evento *click()* que se encuentra en el script *Evento.js*. Este evento ejecuta una función *\$.ajax({})*, la cual llama a la página *Generar_dominio_dialogo.jsp*. Esta página contiene código para efectuar una consulta al catálogo *pg_database* del SABD. El catálogo *pg_databse* almacena información sobre las bases de datos disponibles. A continuación se muestra la consulta que se utiliza para obtener las bases de datos disponibles.

```
SELECT datname FROM pg_database WHERE datistemplate = false
```

Si existe un diccionario de información semántica disponible para la base de datos actual, se muestra un mensaje preguntando si se desea nuevamente generar un diccionario de información semántica para la base de datos. Si el usuario elige la opción "Sí", se ejecuta un diálogo (*JDialog*) para escribir el nombre del diccionario de información semántica, después se presiona el botón "Aceptar". Este evento permite validar el nombre del diccionario de información semántica escrito por el usuario. Esto se realiza al ejecutar una función *\$.ajax({})*, la cual llama a la página *ValidarnombreDIS.jsp*. El código que contiene esta página es similar al de la página *Generar_dominio_dialogo.jsp*, con la diferencia de que se usó otra página para que se pueda identificar con la función que realiza.

Si no existe ningún diccionario de información semántica, se ejecuta otra función *\$.ajax({})*, la cual llama a la página *Generar_dominio.jsp*. En esta página se llama el método *generaDominio()* de la clase *Metodos_clase_Interfaz.java*. Dentro de este método se realizan las conexiones a la base de datos por medio del constructor de la clase *Metadatos.java*.

Además de la inicialización de otros objetos, una de las modificaciones que se realizó para este proceso se encuentra en este constructor; específicamente, se modificó la clase *CopiarArchivo.java*. Debido a que en la ILNBD de escritorio, esta clase permitía generar una copia del archivo del diccionario de información semántica que era una base de datos de Microsoft Access, se sobrescribió esta clase para que funcionara para crear una base de datos y restaurar el diccionario de información semántica.

El siguiente código permite restaurar el diccionario de información semántica *diccionario_de_datos_BDATIS.backup*.

Inicio

```
String path ← "C:\\diccionario_de_datos_BDATIS.backup";
String user ← "postgres";
String password ← "nlipostgres";
Process p;
ProcessBuilder pb;
    pb ← new ProcessBuilder(
        "C:\\Program Files\\PostgreSQL\\10\\bin\\pg_restore.exe",
        "-h",
        "localhost",
        "-U",
        user,
        "-d",
        nombreBD_Destino,
        "-v",
        path);

    pb.environment().put("PGPASSWORD", password);
    pb.redirectErrorStream(true);
```

```
p ← pb.start();
InputStream is ← p.getInputStream();
InputStreamReader isr ← new InputStreamReader(is);
BufferedReader br ← new BufferedReader(isr);
String ll;
while ((ll = br.readLine()) != null) {
    System.out.println("restore:"+ll);
}
```

Fin

También se modificaron los tipos de datos del método *getTipo()* que se encuentra en la clase *MetaDatos.java*. Los tipos de datos se cambiaron por los tipos de datos que maneja el SABD PostgreSQL.

Además, se agregó el método *guardarNombreDominio()*, el cual permite sobrescribir el valor de la columna *bds* de la tabla *usuarios* con el nombre del diccionario de información semántica generado.

4.9 Implementación de la página del wizard

En esta sección se describe la implementación de la página del wizard en donde destaca el proceso para realizar la afinación de la configuración del diccionario de información semántica, el cual se utiliza en el procesamiento de consultas de LN. La afinación se hace con ayuda del wizard. También se mostrará la incorporación de tres algoritmos que se agregaron en el proceso de afinación con el wizard para corregir algunos errores que ocurrieron en algunas consultas en SQL generadas.

Para realizar la implementación se procedió de la misma manera que se mencionó en la Sección 4.6; es decir, encontrando las funciones que se utilizan en la ILNBD de escritorio para realizar el proceso de afinación con el wizard. La interfaz que usa la ILNBD de escritorio se muestra en la Figura 4.9.

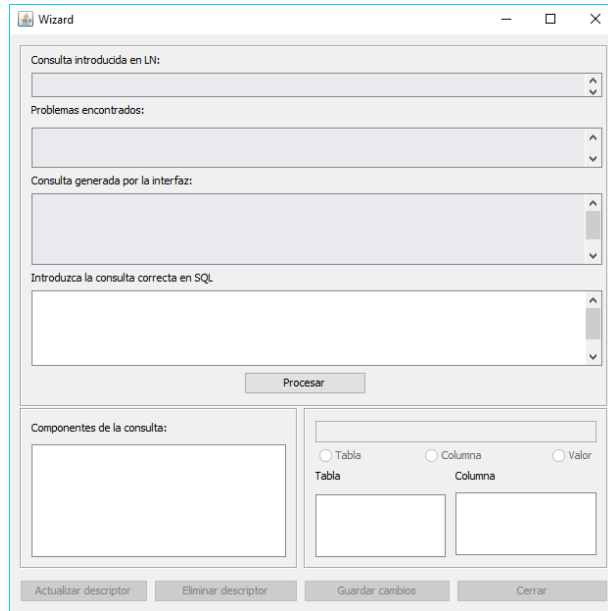


Figura 4.9 Interfaz del wizard en la ILNBD de escritorio

Como se muestra en la Figura 4.9, la interfaz del wizard de la ILNBD de escritorio permite mostrar varios componentes léxicos de la consulta en LN (en el campo "Componentes de la consulta"), los cuales permiten ejecutar eventos para llevar a cabo la afinación. A estos componentes se les denominará componentes de consulta. Para la codificación de estos eventos, se utilizó *javax.swing* que proporciona métodos independientes para cada uno de los componentes. Por este motivo se empezó por detectar el código que utiliza cada uno de estos eventos y así reutilizar la mayor cantidad de código posible.

Para la versión para web, se decidió implementar por separado cada uno de estos eventos. Esto se hizo para respetar la forma en que está desarrollada la ILNBD de escritorio. Para la nueva versión se codificó una página JSP por evento aproximadamente; sin embargo, hay algunos eventos que necesitan más de una página JSP para realizar su proceso.

Para crear la página del wizard de la ILNBD para web que se muestra en la Figura 4.10, se utilizó un diálogo (*JDialog*), el cual se muestra al presionar el botón "Wizard" que se encuentra en la página de consulta. El diálogo fue implementado en la página *Index.jsp*. Este diálogo está conformado por los elementos HTML mostrados en la Tabla 4.2.

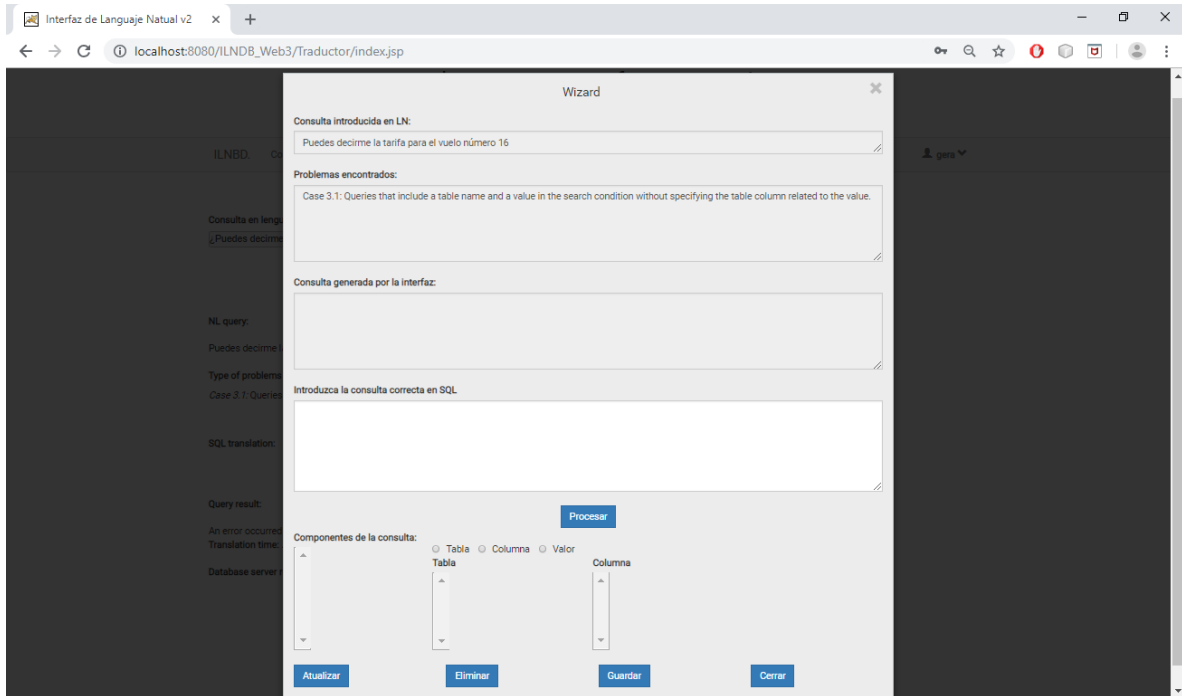


Figura 4.10 Página del wizard en la ILNBD para web

Tabla 4.2 Lista de elementos de HTML que se utilizaron en la página del wizard

Etiqueta	Descripción
<input type="text">	Define un campo de entrada de texto de una línea.
<input type="radio">	Define un botón de radio (para seleccionar una de varias opciones).
<textarea>	Define un control de entrada multilínea (área de texto).
<select>	Define una lista desplegable.
<button>	Define un botón pulsable.

A continuación, se explica lo más importante de la implementación de los eventos de cada uno de los elementos de HTML que permiten la afinación de la configuración utilizando el wizard.

Para implementar la operación de los eventos de la página del wizard, se utilizaron los eventos correspondientes para algunos de los elementos de HTML mostrados en la Tabla 4.2. Los elementos de HTML a los que se les implementaron los eventos son los siguientes: cinco <button> que corresponden a los botones “Procesar”, “Actualizar”, “Eliminar”, “Guardar” y “Cerrar”; tres <select> que corresponden a las listas “Componentes de consulta”, “Tablas” y “Columnas”; y tres <input type="radio"> que corresponden a los radio botones “Tabla”, “Columna” y “Valor”. La mayoría de estos eventos ejecutan una petición al servidor con ayuda de la función `$.ajax({})`, la cual se utilizó para generar la petición a

cada página JSP. Las funciones y los eventos se encuentran dentro de la página *Eventos Wizard.jsp*.

La clase principal para realizar el proceso de afinación en la ILNBD de escritorio es la clase *Wizard.java*. Esta clase se integró al proyecto de la ILNBD para web, además de otras clases que se utilizan dentro de ésta. Se modificaron y se agregaron nuevos métodos dentro de esta clase, y también se sobrecargó el constructor. Para algunos de los métodos fue necesario cambiar el valor de retorno, utilizando los objetos *JSONObject* y *JSONArray* para almacenar la información que retornan estos métodos y así ser enviados a través de la red. Se utilizó la biblioteca XStream para convertir objetos a formato XML, cuyo uso se explicó en la Sección 4.6.

4.9.1 Implementación del evento del botón procesar

Como se muestra en la Figura 4.10, se empezó codificando el evento del botón "Procesar". Este botón permite procesar la consulta correcta introducida por el usuario. Durante este proceso el wizard ofrece sugerencias por medio de un diálogo (*JDialog*), lo cual le permite al usuario asociar palabras con valores de columnas. Al final de este proceso el wizard muestra una lista con los componentes de consulta. El algoritmo principal para realizar este proceso es el siguiente.

Inicio

1. Analizar sintácticamente la consulta correcta introducida por el usuario.
2. Resolver problemas de tablas o columnas mal asociadas.
3. Resolver problemas de tablas o columnas no asociadas.
4. Resolver problemas de valores alias o valores imprecisos no asociados.
5. Resolver problemas de palabras no relacionadas.
6. Mostrar los componentes de consulta en la lista del diálogo (*JDialog*).

Fin

Para cada uno de los pasos del algoritmo anterior, se codificaron uno o más páginas JSP para realizar estos procesos, ya que muestran diálogos durante su ejecución. Los procesos 2, 3 y 4 del algoritmo anterior no estaban implementados en la ILNBD de escritorio. Una de las aportaciones en este proyecto de tesis es la implementación de estos algoritmos, para lo cual se utilizaron los pseudocódigos que fueron extraídos de la tesis de Aguirre.

A continuación, se describe el algoritmo principal del evento del botón "Procesar".

- 1. Analizar sintácticamente la consulta correcta introducida por el usuario:** Para este proceso se codificó la página *WizardJSP_procesar2.jsp*. Esta página es el primero que se ejecuta cuando el usuario utiliza el botón "Procesar". Dicho proceso permite obtener y extraer los componentes de consulta de las cláusulas WHERE y SELECT de la consulta generada por la ILNBD para web, así como los de la consulta correcta introducida por el usuario. Esta información se almacena y envía para ser utilizada en el siguiente proceso.

2. **Resolver problemas de tablas o columnas mal asociadas:** Para este proceso se codificó la página *WizardJSP_procesar_algoritmo6_3.jsp*. Esta página llama al algoritmo que permite resolver problemas de tablas o columnas mal asociadas. El método encargado de realizar este proceso es *tablasColumnasMalAsociadas()* que se encuentra en la clase *Wizard.java*. Este algoritmo (ver Figura 4.11) se implementó primero en la ILNBD de escritorio, con la finalidad de facilitar las pruebas y verificar que este algoritmo funcione adecuadamente, y posteriormente se integró a la ILNBD para web.

```

1:  if isEmpty(SCSQL) and isEmpty(SGSQL)
2:    for j = 0, ..., sizeOf(SGSQL)-1 do // For each element of SGSQL
3:      if compareDescriptors(Q, SGSQLj, SCSQL) // If there are similar descriptors
4:        if showConfirmDialog(Q, SGSQLj, SCSQL) // Show confirm dialog
5:          SGSQL ← remove(SGSQLj, SGSQL) // Remove SGSQLj
6:          SCSQL ← remove(SGSQLj, SCSQL)
7:          updateSID(Q, SCSQL) // Update the SID
8:        endif
9:      else // If there are not similar descriptors
10:       if showInputDialog(Q, SGSQLj, SCSQL) // Show input dialog
11:         SGSQL ← remove(SGSQLj, SGSQL) // Remove SGSQLj
12:         SCSQL ← remove(SGSQLj, SCSQL)
13:         updateSID(Q, SCSQL) // Update the SID
14:       endif
15:     endif
16:   endfor
17: endif

```

Figura 4.11 Algoritmo para corregir problemas de columnas o tablas mal asociadas [Aguirre, 2014]

3. **Resolver problemas de tablas o columnas no asociadas:** Para este proceso se codificó la página *WizardJSP_procesar_algoritmo6_4.jsp*. Dicha página llama al algoritmo (ver Figura 4.12) que permite resolver problemas de tablas o columnas no asociadas. El método encargado de realizar este proceso es *tablasColumnasNoAsociadas()* que se encuentra en la clase *Wizard.java*.

```

1:  if isEmpty(SCSQL) and isEmpty(SGSQL)
2:    for i = 0, ..., n-1 do // For each element of Q
3:      flag ← false
4:      if isNotTagged(Qi)
5:        for j = 0, ..., sizeOf(SCSQL)-1 do // For each element of SCSQL
6:          if compareSynonyms(SCSQLj, Qi) then
7:            if showConfirmDialog(Qi, SCSQLj) then // Show confirm dialog
8:              SCSQL ← remove(Qi, SCSQLj)
9:              updateSID(Qi, SCSQLj) // Update the SID
10:             flag ← true
11:           endif
12:         endfor
13:       endfor
14:       if not flag then // If Qi was not identified
15:         if showInputDialog(Qi, SCSQL) then // Show input dialog
16:           SCSQL ← remove(Qi, SCSQL)
17:           updateSID(Qi, SCSQL) // Update the SID
18:         endif
19:       endif
20:     endfor
21:   endfor
22: endif

```

Figura 4.12 Algoritmo para corregir problemas de columnas o tablas no asociadas [Aguirre, 2014]

- 4. Resolver problemas de valores alias o valores imprecisos no asociados:** Para este proceso se codificó la página *WizardJSP_procesar_algoritmo6_5.jsp*. Esta página llama al algoritmo (ver Figura 4.13) que permite resolver problemas de valores alias y valores imprecisos no asociados. El método encargado de realizar este proceso es *valoresAliasEImprecisosNoAsociados()* de la clase *Wizard.java*.

```

1:  if isEmpty(SCSQL) and isEmpty(SGSQL)
2:    for i= 0, ..., n-1 do // For each element of Q
3:      if isNotTagged(Qi) and isNounOrAdjective(Qi)
4:        if isABetweenStatementIn(SCSQL) // Imprecise value
5:          showConfirmDialog(Qi, SCSQL) // Show confirm dialog
6:          SCSQL ← remove(Qi, SCSQL)
7:          updateSID(Qi, SCSQL) // Update the SID
8:        else // Alias value
9:          if isANotTaggedValue(Qi, SCSQL)
10:            showConfirmDialog(Qi, SCSQL) // Show confirm dialog
11:            updateSID(Qi, SCSQL) // Update the SID
12:          endif
13:        endif
14:      endif
15:    endfor
16:  endif

```

Figura 4.13 Algoritmo para corregir problemas de valores imprecisos o alias no asociados [Aguirre, 2014]

- 5. Resolver palabras no relacionadas:** Para este proceso se codificaron cuatro páginas JSP. Estas páginas se utilizan para ejecutar y mostrar tres diálogos que se necesitan durante el proceso. Dichos diálogos incluyen sugerencias que se le ofrecen al usuario para asociar alguna palabra con alguna columna de la base de datos.

En el diagrama de la Figura 4.14 se muestra de forma muy general la lógica que se utilizó para tratar este proceso para que funcione en la nueva ILNBD.

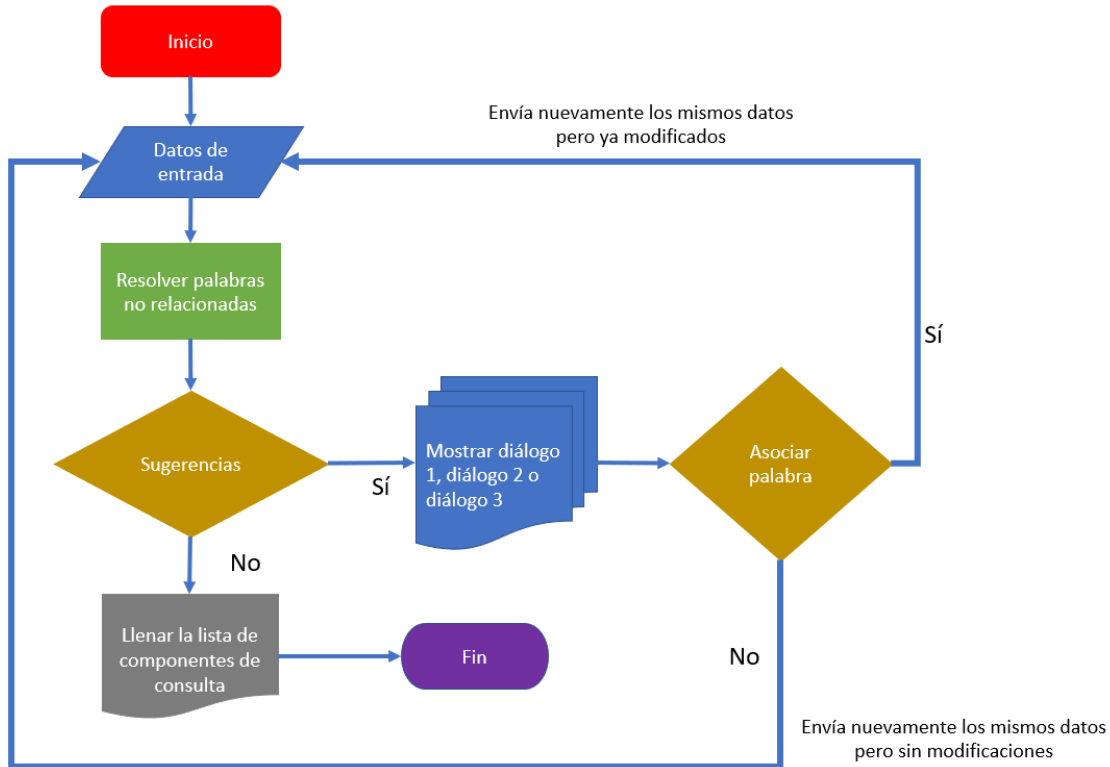


Figura 4.14 Diagrama conceptual del proceso para resolver problemas de palabras no relacionadas

El proceso recibe los datos de entrada que fueron generados por los procesos anteriores. Después se ejecuta el proceso para resolver el problema de palabras no relacionadas en el cual se buscan valores para cada una de las palabras que se encuentran en la consulta en lenguaje natural. Además, se buscan valores que acompañan a la columna de la consulta generada para encontrar columnas con las que existen coincidencias en la consulta correcta. Si se encuentra algún valor, este valor es sugerido por medio de uno de tres diálogos. Si el usuario acepta asociar el valor a la columna, se actualiza un objeto de la clase *Datos.java*, el cual se almacena dentro del objeto *JSONObject* junto con los demás datos que se utilizan para este proceso, los cuales se retornan como resultados de la petición. Después se vuelve a llamar este proceso, pero pasando a la siguiente palabra de la consulta en lenguaje natural. Este proceso se repite hasta que se haya comparado cada una de las palabras que se encuentran en la consulta en lenguaje natural. Nota: La clase *Datos.java* contiene arreglos unidimensionales que permiten almacenar la información de los componentes de consulta.

6. **Mostrar los componentes de consulta en la lista de diálogo (*JDialog*):** Al finalizar el proceso encargado de resolver problemas de palabras no relacionadas, se carga una lista `<select id="lista_referencias" size=8>` con los componentes de consulta correspondientes, seleccionando el primer componente de consulta de la lista.

4.9.2 Implementación del evento de la lista referencias

De acuerdo con lo mencionado en el paso 6 del algoritmo anterior, la lista `<select id="lista_referencias" size=8>` del diálogo (*JDialog*), mostrado en la Figura 4.10, contiene la información que retornó el evento del botón "Procesar". La lista `<select id="lista_referencias" size=8>` utiliza un evento `on("click")` en el script *Wizard.js*. Dicho evento permite capturar el índice del componente de consulta seleccionado dentro la lista `<select id="lista_referencias" size=8>`, ejecutando el método `getColumnasTablas()` que contiene el llamado a una función `$.ajax({})`, la cual llama a la página *WizardJSP_tablas_columnas.jsp*. Esta página llama al método `onclick()` de la clase *Wizard.java*, que devuelve un objeto de tipo *JSONObject*. Este objeto almacena dos objetos de tipo *JSONArray*; en uno almacena todas las tablas de la base de datos, y en el segundo almacena todas las columnas correspondiente al índice seleccionado. Al final, las tablas se muestran en la lista `<select id="lista_tablas" size=6>` y las columnas, en la lista `<select id="lista_columnas" size=6>`.

4.9.3 Implementación del evento de la lista tablas

La lista `<select id="lista_tablas" size=6>` se utiliza para que el usuario pueda asociar de manera manual un componente de consulta con una tabla de la base de datos. Esto se hace con ayuda del evento `on("click")`, con el cual se captura la tabla seleccionada, la cual se utiliza como parámetro para el método `listTablaAction()`. En este método se llama a una función `$.ajax({})`, la cual hace una petición a la página *WizardJSP_columnas.jsp*. En esta página se llama a un método `jListTablaAction()` de la clase *Wizard.java*, el cual devuelve un objeto de tipo *JSONObject* en donde se almacena otro objeto *JSONObject* con los nombres de las columnas de la tabla seleccionadas. Estas columnas se muestran en la lista `<select id=" lista_columnas" size=6>`.

4.9.4 Implementación de los eventos de los botones para asociar componente anterior y posterior

Estos botones permiten asociar un componente de consulta seleccionado con otro componente de consulta anterior o posterior. Esto se hace con los eventos `click()` de cada botón. En cada uno de estos eventos se captura el componente de consulta que se haya seleccionado en la lista `<select id="lista_referencias" size=8>`. Este componente de consulta es pasado como parámetro en el método `asociarComponentes_compruebafrase_dialogo()`. En este método se llama a una función `$.ajax({})`, la cual hace una petición a la página *WizardJSP_asociar_ant_post_compruebafrase_principal.jsp*, el cual llama al método `compruebaFrase()` de la clase *Wizard.java*, que devuelve un objeto de tipo *JSONObject*. Este objeto almacena el mensaje que se mostrará en un diálogo (*JDialog*). El diálogo se muestra en el resultado de la función `success` de la función `$.ajax({})`. Dicho mensaje muestra el sintagma formado, si el sintagma formado es aprobado por el usuario, se ejecuta el evento del botón "Sí", el cual hace una llamada al método `asociarComponentes_compruebafrase_modificar()`. En este método se llama a una función

\$.ajax({}), la cual hace una petición a la página *WizardJSP_asociar_ant_post_compruebafrase2.jsp*, el cual llama al método sobrecargado *compruebaFrase()*. Este método devuelve un objeto de tipo *JSONObject* en donde se almacena otro objeto *JSONObject* que contiene los componentes de consulta actualizados. Al final se actualiza la lista `<select id="lista_referencias" size=8>`.

4.9.5 Implementación del evento de la opción valor

Esta opción se implementó por medio de un botón de selección llamado "Valor". Esta opción muestra un diálogo (*JDialog*) en donde pregunta a qué tipo de valor se desea asociar el componente de consulta seleccionado, ya sea a un "Valor alias" o "Valor impreciso". Al seleccionar cualquiera de las dos opciones, se muestra otro diálogo.

Para el caso de valor alias, muestra el valor alias seleccionado y un campo de texto para asignarle un valor numérico. Este componente de consulta se actualiza por medio del evento del botón "Actualizar" que está en el diálogo. Este evento captura el componente de consulta considerado como valor alias y el valor asociado que representa el valor alias. El componente de consulta y el valor se pasan como parámetros en el método *actualizarvalorAlias()*. En este método se llama a una función *\$.ajax({})*, la cual hace una petición a la página *WizardJSP_ActualizarValorAlias.jsp*, el cual llama al método *actualizaValorAlias()* de la clase *Wizard.java*. Este método actualiza el objeto de la clase *Datos.java*. Este objeto se convierte a una cadena en formato XML por medio de la biblioteca *XStream*, la cual se almacena dentro de un objeto de tipo *JSONObject* para ser retornado.

Para el caso de valor impreciso, el diálogo (*JDialog*) muestra el valor impreciso seleccionado y dos campos de texto para asignar un valor superior y un valor inferior, además de una lista `<select id="VI_lista_tablas" size=8>` que contiene las tablas.

Esta lista se carga cuando se muestra este diálogo, lo cual ocurre cuando se selecciona la opción "Valor impreciso" dentro del diálogo en donde se muestran los dos tipos de valores (alias e impreciso). Este evento captura el componente de consulta que se desee asociar, el cual se pasa como parámetro en el método *getvalorImpreciso()*. En este método se llama a una función *\$.ajax({})*, la cual hace una petición a la página *WizardJSP_valoresimprecisos.jsp*, el cual llama al método *buttonValorAction2()* de la clase *Wizard.java*. Este método devuelve un objeto de tipo *JSONObject* en donde se almacena otro objeto *JSONObject* que contiene las tablas y columnas mencionadas. Además, si el componente de consulta seleccionado ya está asociado como valor impreciso, almacena también su información.

La lista `<select id="VI_lista_tablas" size=8>` (mencionada en el párrafo anterior) permite cargar las columnas al seleccionar una tabla de esta lista. Esto se hace con ayuda del evento *on("click")*, con el cual se captura la tabla seleccionada. Esta tabla se utiliza como parámetro para el método *listTablaActionValoresImprecisos()*. En este método se llama a una función *\$.ajax({})*, la cual hace una petición a la página *WizardJSP_valoresimprecisos_columnas.jsp*, el cual llama a un método *buttonValorAction3()* de la clase *Wizard.java*. Este método devuelve un objeto de tipo

JSONObject en donde se almacena otro objeto *JSONObject* con los nombres de las columnas de la tabla seleccionada.

Estas columnas se muestran en la lista `<select id="VI_lista_columnas" size=8 >` del diálogo (*JDialog*). Al final se implementó el evento del botón "Actualizar" en el diálogo (*JDialog*). Esto se hizo por medio del evento *click()*, con el cual se capturan todas las opciones que ofrece el diálogo para valores imprecisos, los cuales se utilizan como parámetros para el método *actualizarvalorImprecisos()*. En este método se llama a una función *\$.ajax({})*, la cual hace una petición a la página *WizardJSP_ActualizarValorImprecisos.jsp*, el cual llama a un método *actualizaValorImpreciso()* de la clase *Wizard.java*. Este método actualiza el objeto de la clase *Datos.java*. Este objeto se convierte a una cadena en formato XML por medio de la biblioteca *XStream*, la cual se almacena dentro de un objeto de tipo *JSONObject* para ser retornado. Al final se actualiza la lista `<select id="lista_referencias" size=8>`.

4.9.6 Implementación del evento del botón actualizar

Este botón permite actualizar el componente de consulta que haya sido asociado mediante la selección de una de las tablas de la lista `<select id="lista_tablas" size=6>` o la selección de una o más columnas en la lista `<select id=" lista_columnas" size=6>`.

Esto se hace por medio del evento *click()*, con el cual se captura si el componente de consulta se quiere asociar a un tabla, o bien a una o más columnas correspondientes de la tabla seleccionada. Estos valores se pasan como parámetros en el método *actualizarDescriptor()* del script *Wizard.js*. En este método se llama a una función *\$.ajax({})*, la cual hace una petición a la página *WizardJSP_actualizar_descriptor.jsp*, el cual llama al método *jButtonActualizarAction* de la clase *Wizard.java*. Este método devuelve un objeto de tipo *JSONObject*, en donde se almacena otro objeto *JSONObject* con los componentes de consulta actualizados, y un objeto de la clase *Datos.java* convertido a formato XML almacenado en una cadena por medio de la biblioteca *XStream*. Al final se actualiza la lista `<select id="lista_referencias" size=8>`.

4.9.7 Implementación del evento del botón eliminar

Este botón permite eliminar un componente de consulta seleccionado de la lista `<select id="lista_referencias" size=8>`.

Esto se hace por medio del evento *click()*, con el cual se captura el índice del componente de consulta que se quiere eliminar. Este valor se pasa como parámetro en el método *eliminar_Descriptores()* del script *Wizard.js*. En este método se llama a una función *\$.ajax({})*, la cual hace una petición a la página *WizardJSP_EliminarDescriptor.jsp*, el cual llama al método *jButtonEliminarAction* de la clase *Wizard.java*. Este método devuelve un objeto de tipo *JSONObject*, en donde se almacena otro objeto *JSONObject* con un mensaje que se muestra en un diálogo (*JDialog*).

El diálogo se muestra en el resultado de la función *success* de la función *ajax*. El mensaje muestra el componente de consulta a eliminar, se ejecuta el evento del botón "Eliminar", en el cual se hace una llamada al método *eliminar_Descriptores_dialogo()*. En este método se llama a una función *\$.ajax({})*, la cual hace una petición a la página *WizardJSP_EliminarDescriptorDialogo.jsp*, el cual llama al método *actualizaDatos()*. Este método devuelve un objeto de tipo *JSONObject* en donde se almacena otro objeto *JSONObject* que contiene los componentes de consulta actualizados. Al final se actualiza la lista `<select id="lista_referencias" size=8>`.

4.9.8 Implementación del evento del botón guardar

Este botón permite guardar todos los componentes de consulta actualizados. Esto se hace por medio del evento *click()*, con el cual se captura el índice inicial para guardar. Este valor se pasa como parámetro en el método *guardarDescriptor()* del script *Wizard.js*. En este método se llama a una función *\$.ajax({})*, la cual hace una petición a la página *WizardJSP_GuardarDescriptor.jsp*, el cual llama al método *jButtonGuardarAction()* de la clase *Wizard.java*. Este método devuelve un objeto de tipo *JSONObject*, en donde se almacena un *JSONObject* con un mensaje que se muestra en un diálogo (*JDialog*) de confirmación.

El diálogo se muestra en el resultado de la función *success* de la función *ajax*. El mensaje muestra el o los componentes de consulta a guardar, se ejecuta el evento del botón "Sí" dentro de diálogo, cada vez que se muestre un mensaje y el usuario confirme guardar se ejecuta nuevamente el método *guardarDescriptor()*. Esto se repite hasta que cada componente de consulta se actualice.

4.10 Cambios a la interfaz de consulta

En la interfaz de consulta para la nueva ILNBD para web, no se efectuaron cambios en el proceso de consulta que utiliza esta interfaz; es decir, utiliza los mismos algoritmos con que cuenta la ILNBD de escritorio de Aguirre.

Los únicos cambios consistieron en el cambio de la conexión a la base de datos, de Microsoft Access a PostgreSQL, así como la forma de guardar el resultado del proceso de consulta. En cuanto al diseño de la interfaz de consulta, se decidió conservar el mismo diseño de la ILNBD de escritorio, como se muestra en la Figura 4.15, buscando los componentes equivalentes para el lenguaje HTML, como se muestra en la Figura 4.16, además de agregar nuevas opciones en el menú de la interfaz de consulta que se describió en la Sección 4.6.

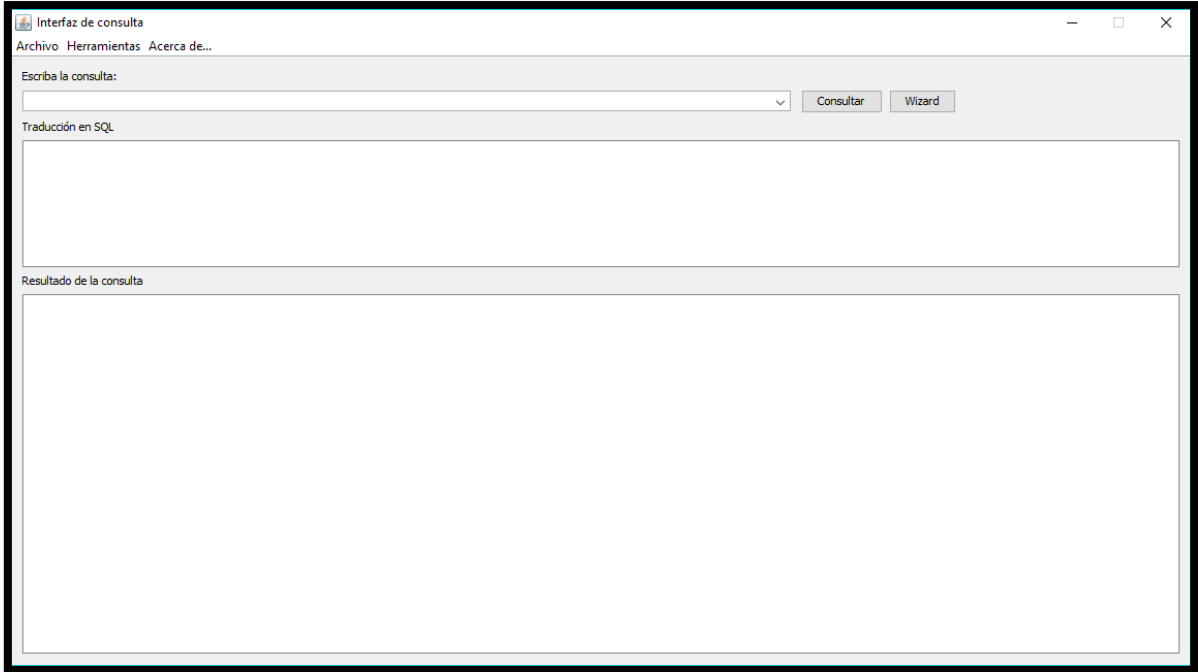


Figura 4.15 Interfaz de consulta de la ILNBD de escritorio

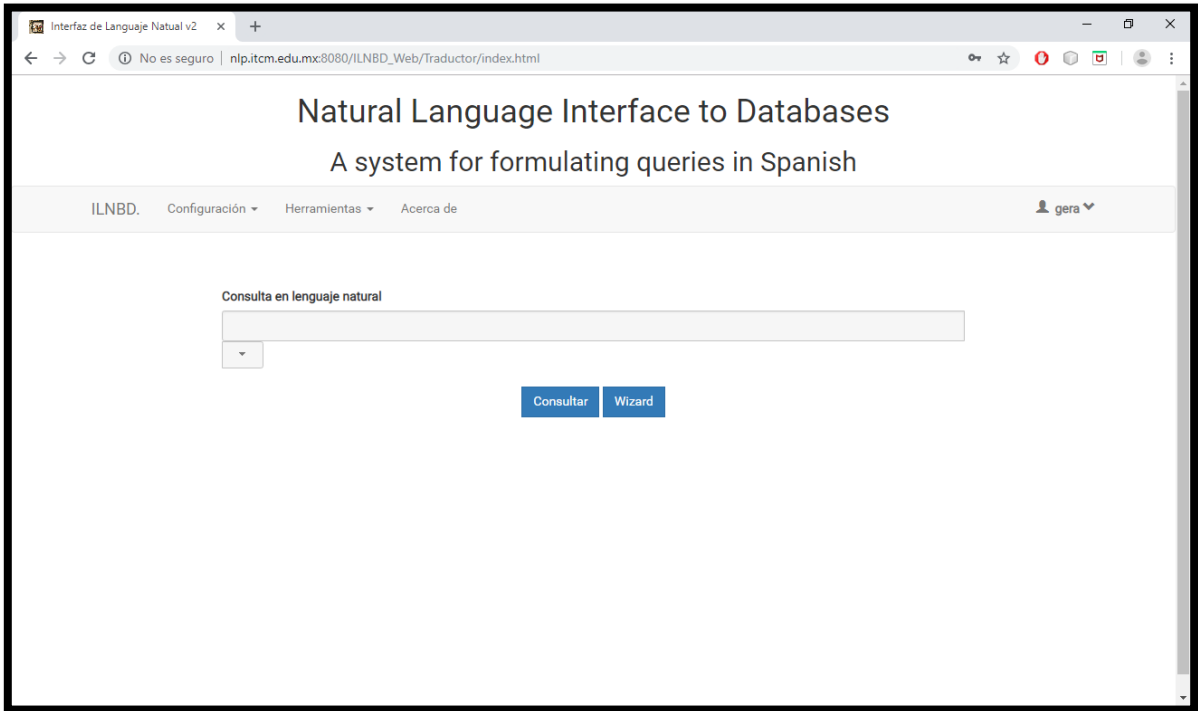


Figura 4.16 Interfaz de consulta de la ILNBD para web

4.11 Arquitectura de la ILNBD para web

4.11.1 Diagrama de la interfaz hombre-máquina

El diagrama de la interfaz hombre-máquina (IHM) de la ILNBD para web (ver Figura 4.17) permite describir la arquitectura de los módulos. En este diagrama se ubican los distintos módulos de manera jerárquica para que a simple vista se pueda determinar las ramas y los principales apartados en que se divide la IHM para web.

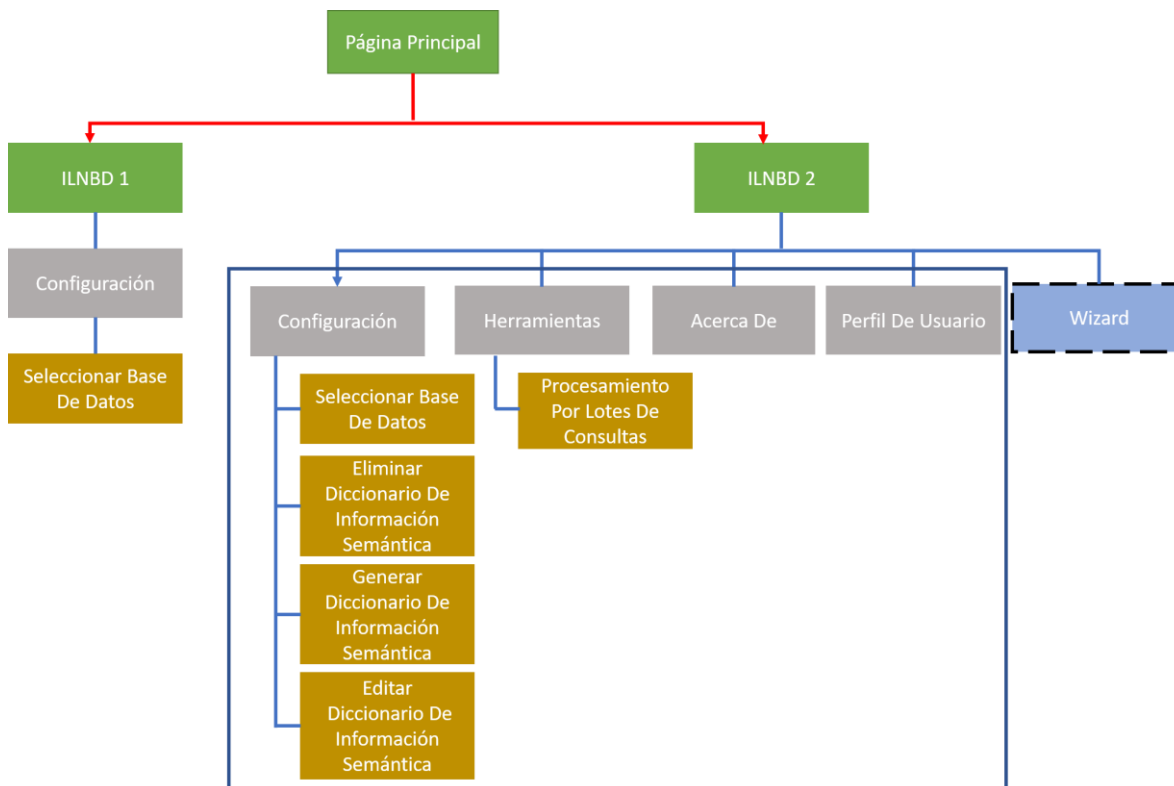


Figura 4.17 Diagrama de la IHM de la ILNBD para web

El diagrama mostrado en la Figura 4.17 principalmente se observan las opciones de la barra de navegación con la que cuenta cada una de las páginas que proceden de la página principal. En este caso la página ILNBD 1 únicamente incluye en su barra de navegación el menú "Configuración/Seleccionar Base de datos", en cambio para la página ILNBD 2 muestra en su barra de navegación cuatro menús disponibles: "Configuración", "Herramientas", "Acerca de" y "Perfil de usuario", en donde "Configuración" y "Herramientas" cuentan con submenús. Además de la barra de navegación para la ILNBD 2, también se muestra una opción "Wizard". Esta opción no forma parte de la barra de navegación; por tal motivo, se muestra fuera de las demás opciones, ya que esta opción se muestra como un botón dentro de la página ILNBD 2.

4.11.2 Estructura de navegación

En esta estructura se muestra la navegación de un usuario dentro de la IHM de la ILNBD para web (ver Figura 4.18), quien puede moverse de una página a otra dentro de la IHM. Esta estructura es de tipo jerárquico, ya que tiene una página principal de donde se accede a varias páginas y de éstas se pueden acceder a otras, y así sucesivamente creando distintos niveles.

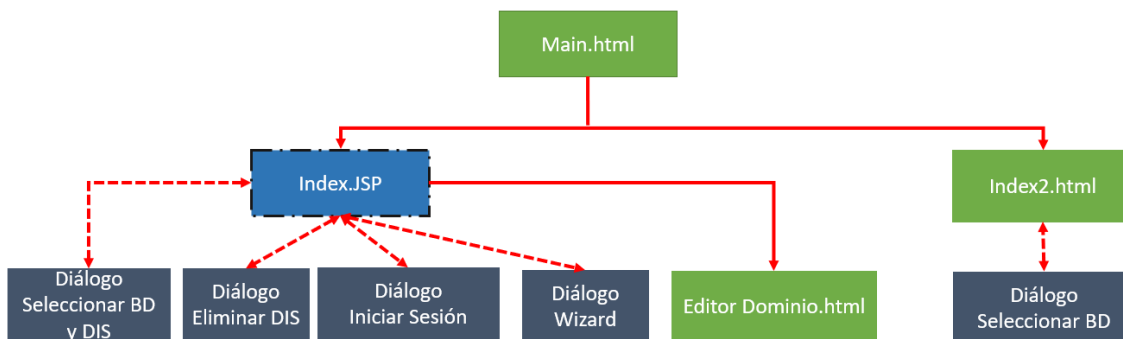


Figura 4.18 Estructura de navegación de la IHM de la ILNBD para web.

En la Fig. 4.18 se muestra la estructura de navegación general con la que cuenta la IHM de la ILNBD para web, en donde se puede apreciar que inicia en una página principal llamada *Main.html*. Esta página muestra dos opciones: una página de tipo JSP llamada *Index.jsp* y otra página de tipo HTML llamada *Index2.html*. La página *Index.jsp* está conectada mediante líneas segmentadas a cuatro rectángulos, los cuales representan los diálogos que se muestran dentro de la página *Index.jsp*; además, la página está conectada mediante línea continua a un rectángulo, el cual representa una página externa llamada *Editor Dominio.html*. La página *Index2.html* únicamente está conectada a un rectángulo que representa un diálogo que se muestra dentro de la misma página.

4.11.3 Jerarquía de clases

Debido a los antecedentes del proyecto, en donde la ILNBD de escritorio desarrollada por Aguirre estaba implementada en lenguaje Java, se decidió usar la tecnología JSP con ayuda de un servidor de web para generar las peticiones para la nueva ILNBD para web, reutilizando el código en Java de la ILNBD de escritorio.

La Figura 4.19 muestra la jerarquía de clases utilizadas para realizar todas las funciones de procesamiento de la ILNBD para web. En esta jerarquía no existe una clase principal, debido a que cada una de estas clases se llama de manera independiente, excepto las clases que crean instancias de otras clases dentro de ellas. Dichas clases se muestran unidas por una línea. En esta jerarquía de clases únicamente se muestran las clases que se modificaron en algunos de sus métodos, para que funcionaran con la nueva versión de la ILNBD. Algunos fragmentos de código en Java fueron escritos directamente en las páginas JSP correspondiente a cada función y éstos no se muestran en la jerarquía de clases.

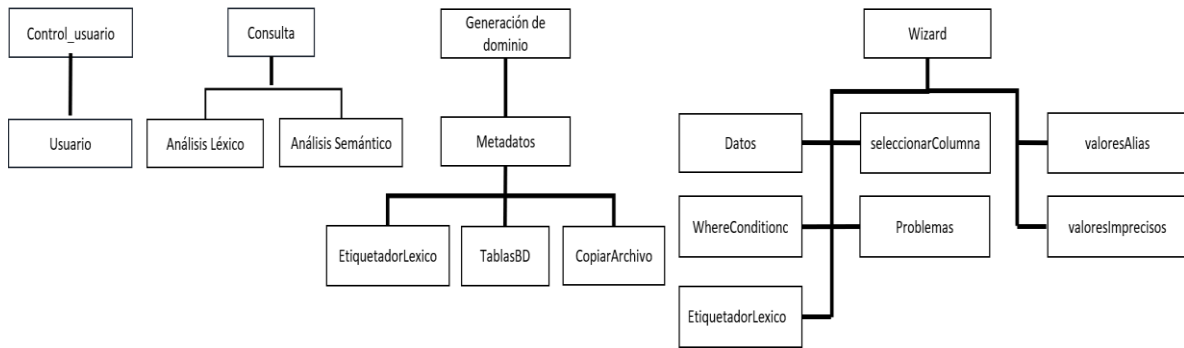


Figura 4.19 Jerarquía de clases de la ILNBD para web

4.11.3.1 Clase Control_usuario

La clase *control_usuario* permite verificar si un usuario puede acceder a la ILNBD para web.

Los objetos y variables que se utilizan en las modificaciones de esta clase se muestran en la Tabla 4.3.

Tabla 4.3 Lista de objetos o variables utilizadas en la clase Control_usuario

Nombre del objeto o variable	Tipo de clase	Descripción
Invitado	Usuario	Objeto utilizado para almacenar la información del usuario.

El método que integra esta clase es el siguiente:

- **Usuario logear(String usuario, String contraseña).** Verifica si el usuario y contraseña del cliente son válidos para acceder a la ILNBD para web.

4.11.3.2 Clase Consulta

Esta clase contiene todos los métodos utilizados para realizar el procesamiento de consultas en lenguaje natural, ejecutar una consulta SQL, el procesamiento por lotes de consultas, ejecutar una consulta SQL por lotes y la conexión a la base de datos.

Los objetos y variables que se utilizan en las modificaciones de esta clase se muestran en la Tabla 4.4.

Tabla 4.4 Lista de objetos o variables utilizadas en la clase Consulta

Nombre del objeto o variable	Tipo de clase	Descripción
pgUser	String	Cadena que almacena el usuario del conector del SABD.
pgPwd	String	Cadena que almacena el password del conector del SABD.
pgUrl	String	Cadena que almacena la dirección del conector del SABD.
consulta	Consulta	Objeto que almacena información sobre la consulta en LN.
conLexicon	Connection	Conector para la base de datos lexicón.
conVerbos	Connection	Conector para la base de datos Verbos.
conDD	Connection	Conector para la base de datos DIS.
conDB	Connection	Conector para la base de datos ATIS o Geobase.
sinonimos	Connection	Conector para la base de datos sinónimos.
capa1	AnalisisLexico	Objeto utilizado para el análisis léxico.
capa2	AnalisisSemantico	Objeto utilizado para el análisis semántico.

Los métodos modificados que integran esta clase son los siguientes:

- **Consulta(String BD, String DIS).** Constructor de la clase en donde se llama al método *cargaBDs*.
- **cargaBDs(String BD, String DIS).** Método encargado de llamar para cada conector el método *cargaBD*.
- **Connection cargaBD(String db, String pgUser, String pgPwd, String pgUrl).** Permite conectar a una base de datos.
- **JSONObject Consultar(String consultaLN).** Realiza el procesamiento de una consulta en lenguaje natural.
- **String ejecutaConsulta(String consulta).** Ejecuta una consulta en SQL.
- **JSONArray procesar_lotes_consulta(String BD, String DIS).** Realiza el procesamiento de un lote de consultas en lenguaje natural.
- **boolean ejecutaConsultaLote(String consulta).** Ejecuta una consulta en SQL.

4.11.3.3 Clase Generación de dominio

Esta clase se utiliza para realizar la generación de dominio.

El método que integra esta clase es el siguiente:

- **generaDominio(String url, String usuario, String DIS).** En este método se llaman a las clases que permiten realizar el proceso de generación de dominio.

4.11.3.4 Clase Metadatos

Los objetos y variables que se utilizan en las modificaciones de esta clase se muestra en la Tabla 4.5.

Tabla 4.5 Lista de objetos o variables utilizadas en la clase Metadatos

Nombre del objeto o variable	Tipo de clase	Descripción
conn	Connection	Conector para la base de datos.
conn2	Connection	Conector para la base de datos.
tabla	TablasBD	Objeto para almacenar la información de una tabla.

Los métodos modificados que integran esta clase son los siguientes:

- **Metadatos(String d, String usuario, String DIS).** En este constructor se crea una nueva base de datos y se restaura la base de datos que sirve como plantilla para un nuevo diccionario de información semántica.
- **Connection conectar(String direccion).** Realiza la conexión a una base de datos.
- **Connection cerrar(Connection conn1).** Cierra la conexión de una base de datos.
- **String getTipo(String tipo).** Permite obtener los tipos de datos de las columnas de una tabla.
- **generarDominio(String url, String usuario, String DIS).** En este método se realiza el llenado del diccionario de información semántica con información de las tablas, columnas y relaciones, además de actualizar el nombre del DIS generado en la columna del usuario correspondiente.

4.11.3.5 Clase CopiarArchivo

Esta clase realiza la creación de una base de datos con el nombre del diccionario de información semántica que se va a generar, así como la restauración de la base de datos del DIS que sirve como plantilla a la base de datos del nuevo DIS.

El método que integra esta clase es el siguiente:

- **restaurarBD_DIS(String nombreBD_Destino).** Realiza la creación y la restauración de un nuevo diccionario de información semántica.

4.11.3.6 Clase Wizard

Esta clase está conformada por los métodos encargados de realizar los procesos del wizard de la INLBD para web. Para esta clase fue necesario realizar varias modificaciones a sus métodos, además de agregar nuevos objetos y variables.

Los objetos y variables globales que se utilizan en las modificaciones de esta clase se muestran en la Tabla 4.6

Tabla 4.6 Lista de objetos o variables utilizadas en la clase Wizard

Nombre del objeto o variable	Tipo de clase	Descripción
pgUser	String	Cadena que almacena el usuario del conector del SABD.
pgPwd	String	Cadena que almacena el password del conector del SABD.
pgUrl	String	Cadena que almacena la dirección del conector del SABD.
consulta	Consulta	Objeto que almacena información sobre la consulta en LN.
conLexicon	Connection	Conector para la base de datos lexicón.
conVerbos	Connection	Conector para la base de datos Verbos.
conDD	Connection	Conector para la base de datos DIS.
sinonimos	Connection	Conector para la base de datos sinónimos.
jselementos	JSONObject	Objeto Json utilizado para almacenar todos los problemas generados por el proceso de consulta en LN.
jConsultaGenerada	String	Cadena que almacena la consulta generada.
jConsultaCorrecta	String	Cadena que almacena la consulta correcta introducida por el usuario.
jsListTabla	JSONArray	Objeto utilizado para almacenar todos los nombres de las tablas del DIS.
jsListColumna	JSONArray	Objeto utilizado para almacenar todos los nombres de las columnas del DIS.
datosConsulta	Datos	Este objeto almacena toda la información extraída del diccionario de información semántica.

Esta clase cuenta con cinco constructores sobrecargados, cada uno de los cuales se llama de acuerdo con el proceso solicitado dentro del wizard de la ILNBD para web.

1. Wizard(Consulta consultaOriginal, Problemas tipoDeProblemasFraseSelect, Problemas tipoDeProblemasFraseWhere, String consultaGenerada, String BD, String DIS)
2. Wizard(Consulta consultaOriginal, String consultaGenerada, String consultaCorrecta, String BD, String DIS)
3. Wizard(String BD, String DIS)
4. Wizard()
5. Wizard(Consulta consultaoriginal)

Algunos de los métodos modificados que integran esta clase son los siguientes:

- **JSONArray analizaSintacticamenteConsultasSQL().**
- **boolean esSinonimo(String palabra, String SCSQLj).**
- **ArrayList copiarArraylist(ArrayList al).**
- **JSONObject tablasColumnasMalAsociadas(int Estado, int indice_tcma, String tablasSQLcorrecto, String columWheresobrantes, String columSelectsobrantes, int indice_diag2).**
- **ArrayList compareDescriptors(String SGSQL).**
- **ArrayList existeDescriptor(String columnaQ, String tablaQ, String columna, String tabla, boolean esColumna, boolean esTabla).**
- **JSONObject tablasColumnasNoAsociadas(int Estado, int indice_tcna, String tablasSQLcorrecto, String columWheresobrantes, String columSelectsobrantes, int indice_diag2).**
- **JSONObject valoresAliasEImprecisosNoAsociados(int Estado, int indice_tcma, String columWheresobrantes, String columSelectsobrantes, String clausulaselectcorrecto, String clausulaWhereSQLcorrectocolumna, String clausulaWhereSQLcorrecto_valor, String clausulaWhereSQLcorrecto_operador).**
- **JSONObject resuelvePalabrasNoRelacionadas(int m, JSONArray jsreferencias, String consultaGenerada, String clausulaWhereSQLcorrectocolumna, String clausulaWhereSQLcorrecto_valor, String clausulaWhereSQLcorrecto_operador, String clausulaWhereSQLGenerado_valor, String clausulaWhereSQLGenerado_valor_columna, String clausulaWhereSQLGenerado_operador, String clausulaWhereSQLGenerado_columna, String tablasSQLcorrecto, String columWheresobrantes, String columSQLcorrecto, int Estado, String respseleccionarColumna).**
- **JSONObject cargaColumnas(String Columna).**
- **JSONObject actualizaDatos(int posicionLista).**
- **JSONObject cargaTablas(String Tabla).**
- **JSONObject jListTablaAction(String nombre_tabla_seleccionada, int ListTablaItemCount).**
- **JSONObject radioButtonTablaAction().**
- **JSONObject jButtonEliminarAction(int posicionLista).**
- **boolean insertarDescriptor(String palabra, String columna, String tabla, int posicionLista).**
- **JSONObject jButtonGuardarAction(int indice, int Estado).**
- **JSONObject jButtonActualizarAction(int posicionLista, boolean rdbuttonColumna, boolean rdbuttonTabla, int getSelectedIndexesListaColumna[], int getItemCountListColumna, String getSelectedItem_ListTabla, String getSelectedItem_ListColumna, JSONArray jsListColumna).**
- **JSONObject buttonValorAction(int selectedIndexReferencias).**
- **JSONObject buttonValorAction2(String DIS, int selectedIndexReferencias).**
- **JSONObject buttonValorAction3(String jListTablagetSelectedItem, int selectedIndexReferencias).**
- **cargaBDs(String BD, String DIS).**
- **Connection cargaBD(String db, String pgUser, String pgPwd, String pgUrl).**
- **JSONArray analizaSintacticamenteConsultasSQL().**

Capítulo 5

Pruebas de la ILNBD

En este capítulo se presentarán las pruebas realizadas a la ILNBD, con el fin de comprobar el correcto funcionamiento de ésta. También se presentará el objetivo de las pruebas, la descripción de los escenarios de pruebas y los casos de prueba utilizados con sus respectivos resultados. Se describirá el hardware y software utilizado para la ejecución de las pruebas.

5.1 Objetivos de las pruebas

El objetivo principal de las pruebas realizadas a la ILNBD para web consiste en verificar el funcionamiento de todas las funciones que se agregaron a esta nueva versión, que se ejecuten de manera correcta, y que los resultados sean los esperados en todas sus funciones; es decir, iguales a los de la ILNBD de escritorio.

Los objetivos de las pruebas de la ILNBD son los siguientes:

- 1 Verificar que la ILNBD para web permita formular consultas en lenguaje natural a través de internet.
- 2 Verificar que la ILNBD para web se pueda configurar a través de internet, en donde destacan las siguientes configuraciones: configuración automática o generación de dominio, afinación manual o edición de dominio y la más importante la afinación con el wizard.
- 3 Verificar que la ILNBD para web atienda satisfactoriamente las peticiones de cada uno de los clientes que realicen una conexión a la página web de la ILNBD.
- 4 Verificar que el usuario pueda seleccionar la base de datos y seleccionar o eliminar el diccionario de datos que utilizará la ILNBD para web a través de internet.

5.2 Descripción del escenario de pruebas

En esta sección se describen los escenarios utilizados para las pruebas de la ILNBD para web. Las pruebas se realizaron principalmente de manera local utilizando el equipo con el cual se implementó la ILNBD, así como utilizando internet. A continuación, se describen ambos escenarios.

En el escenario uno, mostrado en la Fig. 5.1, las computadoras se encuentran conectadas a través de una red local. En este caso únicamente se utilizaron dos computadoras. En la computadora A se encuentra instalado el servidor de web Apache Tomcat, el SABD PostgreSQL, así como todo el proyecto correspondiente a la codificación de la ILNBD para web.

La computadora B permite acceder a la página web principal de la ILNBD.

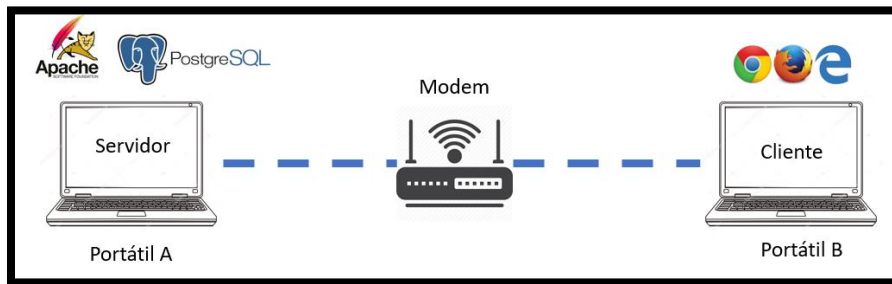


Figura 5.1 Escenario de pruebas número uno

El escenario de pruebas número dos, mostrado en la Fig. 5.2, es similar al escenario número uno; sin embargo, existen las siguientes diferencias:

- En el escenario de pruebas número dos, el acceso a la ILNBD para web es mediante el uso de internet; es decir, cualquier computadora tiene acceso siempre y cuando esté conectada a internet. Por otra parte, en el escenario número uno únicamente pueden acceder a la ILNBD de web las computadoras que se encuentren conectadas a la misma red local en donde se encuentra hospedada la ILNBD para web.

Para el escenario de pruebas número dos, la ILNBD para web se hospeda en uno de los servidores del ITCM.

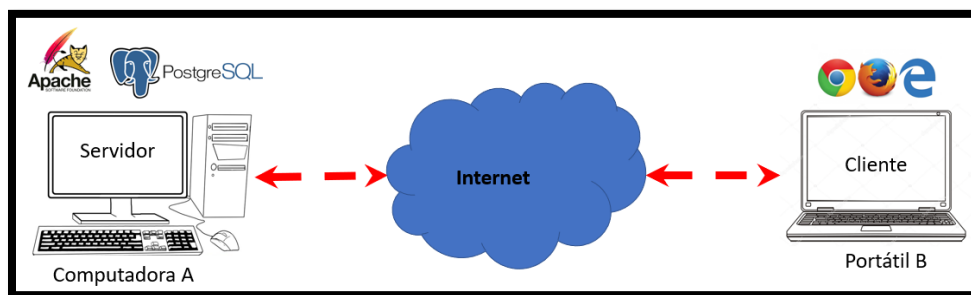


Figura 5.2 Escenario de pruebas número dos

5.2.1 Hardware y software utilizado

Para cada uno de los escenarios de pruebas presentados, se utilizó diferente tipo de hardware y software. A continuación en la Tabla 5.1 se presentan las especificaciones de hardware y

software que utiliza el equipo correspondiente al servidor donde se hospedó la aplicación de ILNBD de cada escenario de pruebas.

Tabla 5.1 Especificaciones de hardware y software de los escenarios de pruebas

Escenario número uno	Escenario número dos
<i>Hardware</i>	<i>Hardware</i>
Procesador: Intel (R) Core (TM) i5-6200, CPU @ 2.30GHz 2.40GHz	Procesador: Intel (R) Xeon(R), CPU E3-1225 v3 @ 3.20GHz 2.20GHz
Memoria instalada (RAM): 4.00 GB	Memoria instalada (RAM): 4.00 GB
<i>Software</i>	<i>Software</i>
Tipo de sistema: Sistema Operativo Windows 10 Home Single de 64 bits.	Tipo de sistema: Sistema Operativo Windows 7 Professional de 64 bits.
Servidor de web: Apache Tomcat 7.0.77	Servidor de web: Apache Tomcat 9.0.8
Netbeans IDE 8.2	---
Java versión 8	Java versión 8
SABD: PostgreSQL 9.6	SABD: PostgreSQL 10

Para los clientes de los dos escenarios, se utilizó un navegador de web para acceder a la ILNBD para web.

5.2.2 Corpus de consultas para prueba

Para realizar las pruebas aplicadas a la ILNBD para web para el proceso de consulta en lenguaje natural y para la afinación de la configuración de la ILNBD se utilizó principalmente la base de datos ATIS (Air Travel Information System). Esta base de datos tiene almacenada información de vuelos y suele usarse para pruebas. En estas pruebas se utilizó un corpus de 70 consultas que se obtuvieron del Apéndice B de la tesis de Aguirre.

Para algunos casos de pruebas se utilizó la base de datos Geobase, la cual almacena información geográfica de los Estados Unidos de América. Para las pruebas se utilizaron algunas consultas del corpus de 250 consultas que se obtuvieron del Apéndice D de la tesis de Aguirre.

5.3 Casos de prueba para la ILNBD para web

Se realizaron las pruebas funcionales a la ILNBD para web. Estas pruebas consistieron en verificar que todas las opciones con las que cuenta la ILNBD de escritorio que se implementaron para la nueva ILNBD para web cumplan su función adecuadamente, en donde destacan la página de consulta y la página de configuración. En la página de configuración destaca principalmente la afinación con el wizard, para la cual se realizaron pruebas a los nuevos algoritmos incorporados al wizard. Además, se efectuaron pruebas a los módulos adicionales que se integraron: control de usuario, eliminar o seleccionar un diccionario de información semántica y el procesamiento por lotes de consultas.

Prueba No. 1

Comprobar la concurrencia de peticiones realizadas al servidor de web por medio de la ILNBD para web.

Resultado:

Para realizar esta prueba se ejecutó la ILNBD para web en tres equipos. En dos de los equipos se accedió a ILNBD para web como usuario identificado y en el otro como usuario invitado, los cuales enviaron simultáneamente diferentes peticiones al servidor. Para los usuarios identificados, un usuario realizó la afinación con ayuda del wizard y el otro realizó la generación de un nuevo diccionario de información semántica, y para el usuario invitado únicamente se procesó una consulta en lenguaje natural. Dichas peticiones fueron satisfechas sin causar interferencia ni incongruencias. El orden en que arribaron las peticiones al servidor varió dependiendo de la velocidad de la red. Cada solicitud recibida fue procesada por sus procesos correspondientes, y éstos enviaron los resultados a sus respectivos equipos que generaron las solicitudes.

Prueba No. 2

Formulación de consultas en lenguaje natural.

Resultado:

Se accedió a la ILNBD para web en tres equipos clientes. En dos equipos se formularon consultas a la base de datos BDATIS y en el otro equipo se formularon consultas a la base de datos Geobase. Estos equipos enviaron simultáneamente diferentes peticiones al servidor. Dichas peticiones fueron satisfechas sin causar interferencia. El orden en que arribaron las peticiones al servidor varió dependiendo de la velocidad de la red. Cada solicitud recibida fue procesada por sus procesos correspondientes y éstos enviaron los resultados a sus respectivos equipos que generaron las solicitudes.

En la página de consulta se muestra la consulta en lenguaje natural, el tipo de problemas en la consulta, la traducción a SQL, el resultado de la consulta en SQL, el tiempo de traducción y el tiempo de respuesta del servidor.

En la Figura 5.3 se muestra el resultado de la consulta *¿Puedes decirme la tarifa para el vuelo número 16?*

NL query:
Puedes decirme la tarifa para el vuelo número 16

Type of problems in the query:
Case 4.1: Queries that include a column name without specifying one of the several tables to which it may be related.

SQL translation:
SELECT fare.md_trip_cost, fare.one_way_cost FROM fare, flight, flight_fare WHERE fare.fare_code = flight_fare.fare_code AND flight_fare.flight_code = flight.flight_code AND flight.flight_number = 16;

Query result:
Show 10 entries Search:

md_trip_cost	one_way_cost
0	233
0	222
228	0
268	0
672	336
1008	504

Showing 1 to 6 of 6 entries Previous 1 Next
Translation time: 0.437 seconds.
Database server response time: 0.000 seconds.

Figura 5.3 Resultado de una consulta en lenguaje natural

Prueba No. 3

Procesamiento de consultas por lotes.

Resultado:

Se accedió a la ILNBD para web utilizando dos usuarios identificados en dos equipos clientes. En ambos equipos se solicitó el procesamiento de consultas por lotes. En un equipo se utilizó la base de datos BDATIS y en el otro equipo, la base de datos Geobase. Dichas peticiones se enviaron de manera simultánea. Ambas peticiones se procesaron de manera correcta sin causar ninguna interferencia o error.

En la página de consulta se muestra la lista de las consultas procesadas, la cual incluye la consulta en lenguaje natural, el número de problema que se detectó en la consulta, la consulta en SQL, el tiempo de procesamiento y el resultado de la consulta en SQL.

En la Figura 5.4 se muestra el resultado del procesamiento por lotes de consultas de la base de datos BDATIS.

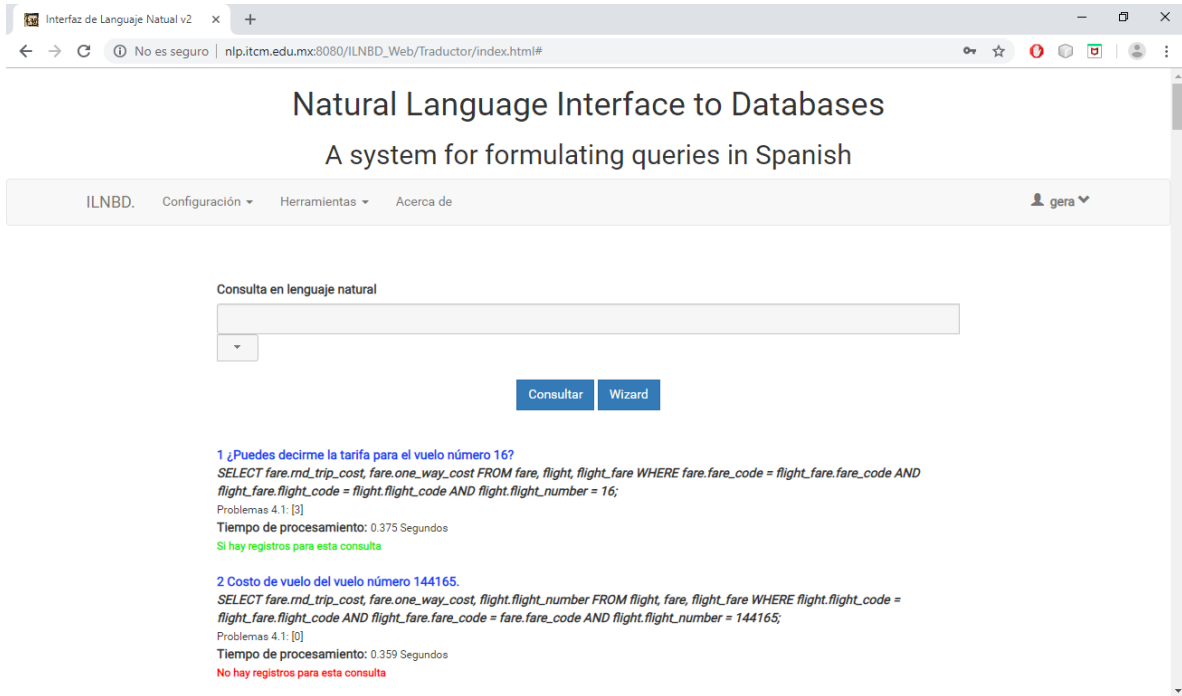


Figura 5.4 Resultado del procesamiento por lotes de consultas de ATIS

En la Figura 5.5 se muestra el resultado del procesamiento por lotes de consultas de la base de datos Geobase.

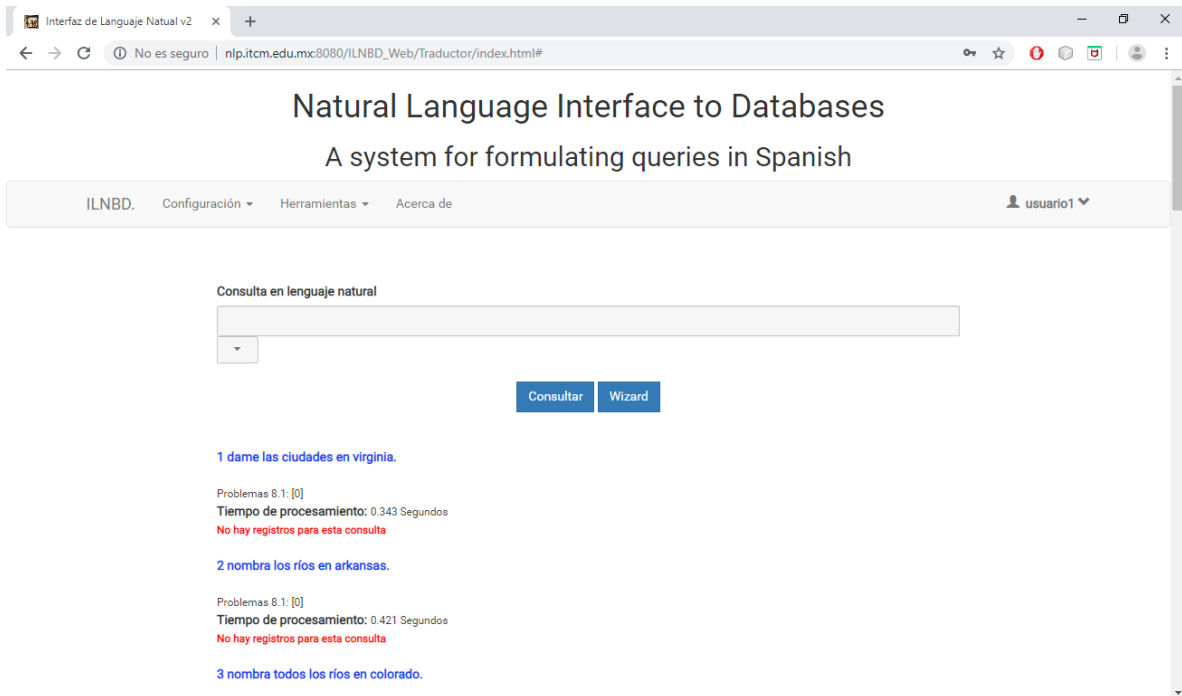


Figura 5.5 Resultado del procesamiento por lotes de consultas de Geobase

Prueba No. 4

Seleccionar una base de datos y un diccionario de información semántica.

Resultado:

En esta prueba se accedió a la ILNBD para web utilizando un usuario identificado. En el menú "Configuración/Seleccionar base de datos", esta opción muestra un diálogo en donde el usuario puede seleccionar una de las bases de datos disponibles: BDATIS o Geobase. Al seleccionar una de estas bases de datos, se carga el diccionario de información semántica correspondiente a dicha base de datos. En este ejemplo se seleccionó la base de datos BDATIS y el diccionario de información semántica Prueba, como se muestra en la Figura 5.6, y se procedió a formular algunas consultas (ver Figura 5.7). Estas peticiones se procesaron de manera satisfactoria.

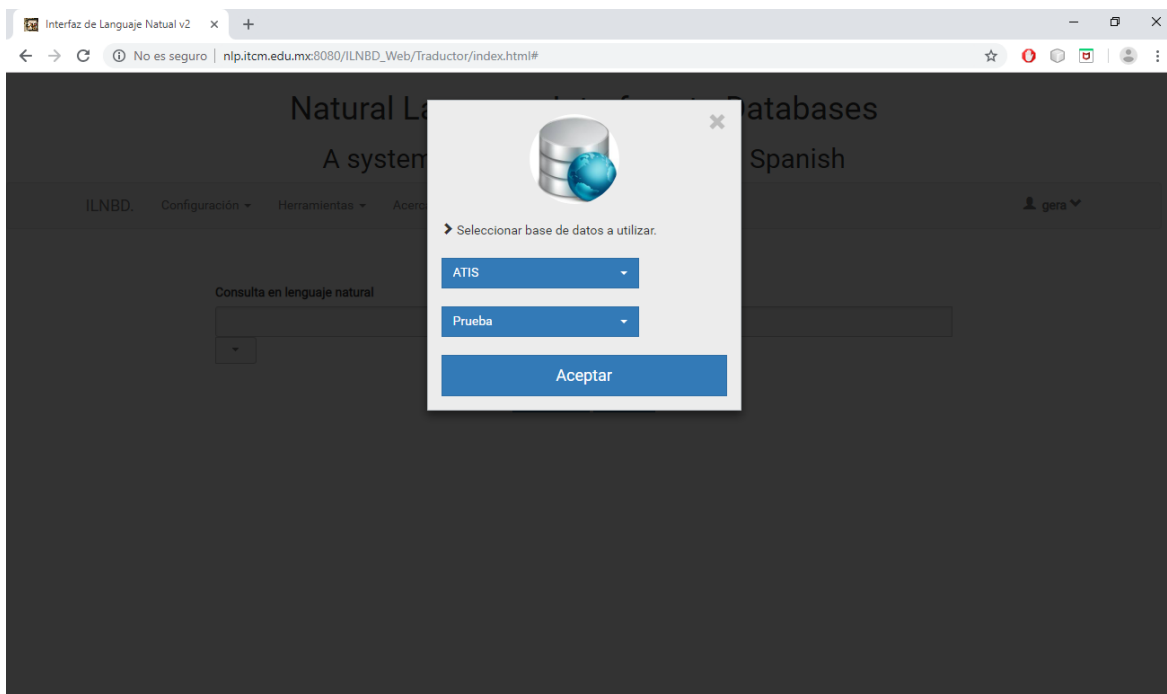


Figura 5.6 Selección del diccionario de información semántica

Natural Language Interface to Databases
A system for formulating queries in Spanish

Consulta en lenguaje natural
Lista todos los vuelos.

Consultar Wizard

NL query:
Lista todos los vuelos

Type of problems in the query:

SQL translation:
SELECT flight.* FROM flight;

Query result:
Show 10 entries

flight_code	flight_days	from_airport	to_airport	departure_time	arrival_time	airline_code	flight_number	class_string	aircraft_code
101908	1234567	ATL	BOS	636	1000	DL	296	FNYNBNMQ	72S
101909	1234567	ATL	BOS	641	855	DL	314	FNYNBNMQ	72S
101910	1234567	ATL	BOS	755	1019	EA	140	FYHQB	D9S
101911	1234567	ATL	BOS	920	1150	EA	534	FYHQB	D9S
101912	1234567	ATL	BOS	959	1215	DL	410	FYBMQ	757

Figura 5.7 Resultado de una consulta en lenguaje natural

Prueba No. 5

Eliminar un diccionario de información semántica.

Resultado:

En esta prueba se accedió a la ILNBD para web utilizando un usuario identificado. En el menú "Configuración/Eliminar diccionario de información semántica", esta opción muestra un diálogo en donde el usuario puede seleccionar una de las bases de datos disponibles: BDATIS o Geobase. Para esta prueba se utilizó la base de datos BDATIS (ver Figura 5.8), para la cual se eliminaron dos diccionarios Prueba y Prueba2 de diez disponibles (ver Figura 5.9).

Eliminar diccionario de información semántica.

Base de datos
BDATIS

Diccionario de datos disponibles

Show 10 entries Search:

Diccionario de Información Semántica	Base de datos	Delete
DIS_Experimentacion1	BDATIS	

Figura 5.8 Diálogo para eliminar un *DIS*

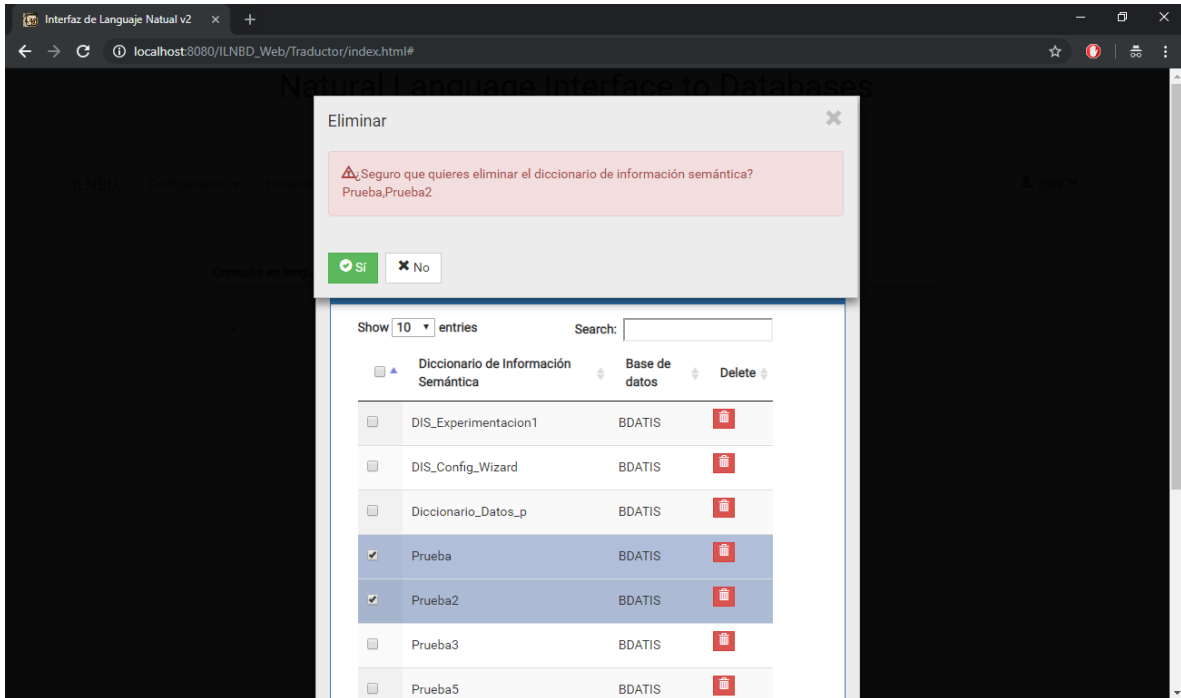


Figura 5.9 Eliminación de los DIS *Prueba* y *Prueba2*

En la Figura 5.10 se muestra el resultado obtenido después de eliminar los dos DIS, de los diez disponibles.

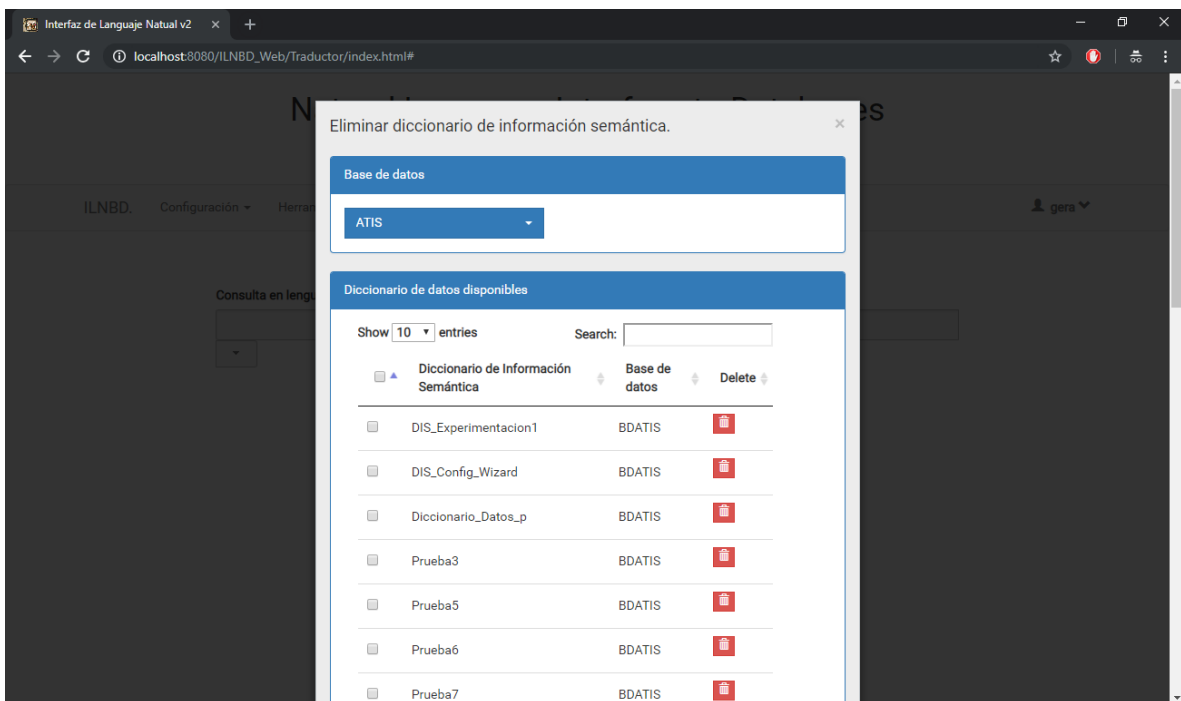


Figura 5.10 DIS disponibles después de la eliminación

Prueba No. 6

Afinación manual o edición de dominio.

Resultado:

En esta prueba únicamente se verificó que la página muestre la tabla con la información del diccionario de información semántica, como se muestra en la Figura 5.11, ya que se deshabilitó la opción de editar.

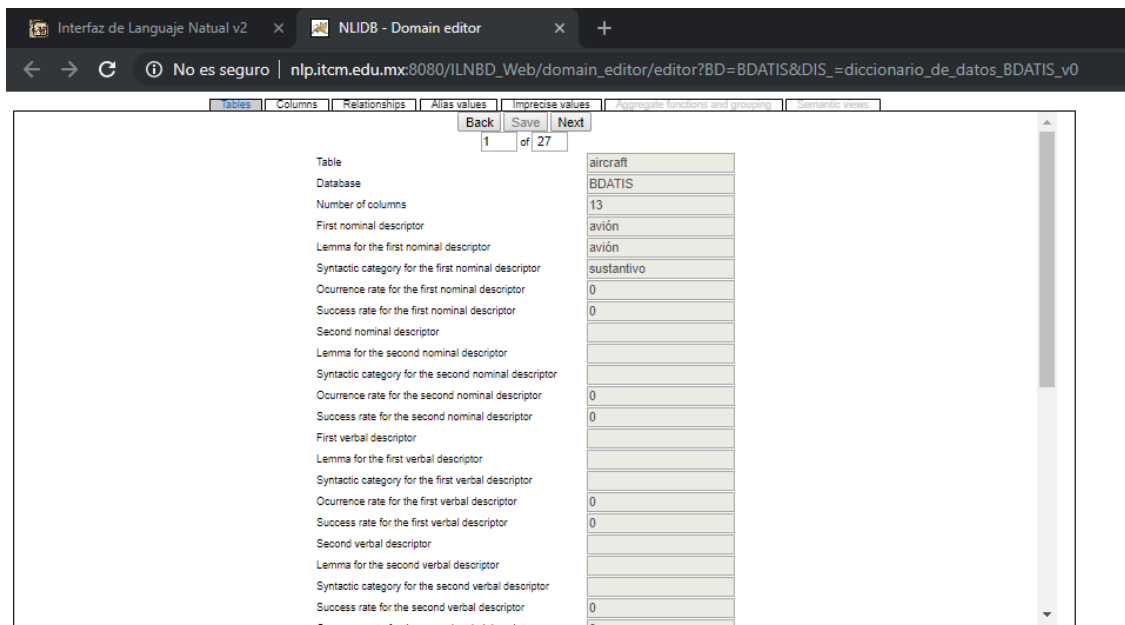


Figura 5.11 Editor de dominio

Las pruebas 7 y 8 corresponden a las pruebas funcionales de los nuevos algoritmos implementados que fueron incorporados en la afinación con el wizard. Para realizar las pruebas de los nuevos algoritmos, se utilizó una consulta para cada algoritmo de acuerdo con el problema que ocurre en dicha consulta. En el capítulo 6 de la tesis de Aguirre se pueden encontrar ejemplos de las consultas que presentan algunos de los problemas mencionados en la Subsección 4.9.1.

Prueba No. 7

Problemas de columnas o tablas no asociadas.

Resultado:

La primera consulta corresponde al **problema de columnas o tablas no asociadas**; es decir, cuando la consulta en lenguaje natural tiene palabras en posiciones donde se esperan nombres de **tablas** o **columnas**, pero durante el proceso de traducción no se puede asociar a ninguna tabla o columna [Aguirre, 2014]. Por ejemplo, la siguiente consulta:

¿Puedes decirme la tarifa para el vuelo número 16?

Al introducir esta consulta la ILNBD de escritorio muestra el resultado en la Figura 5.12.

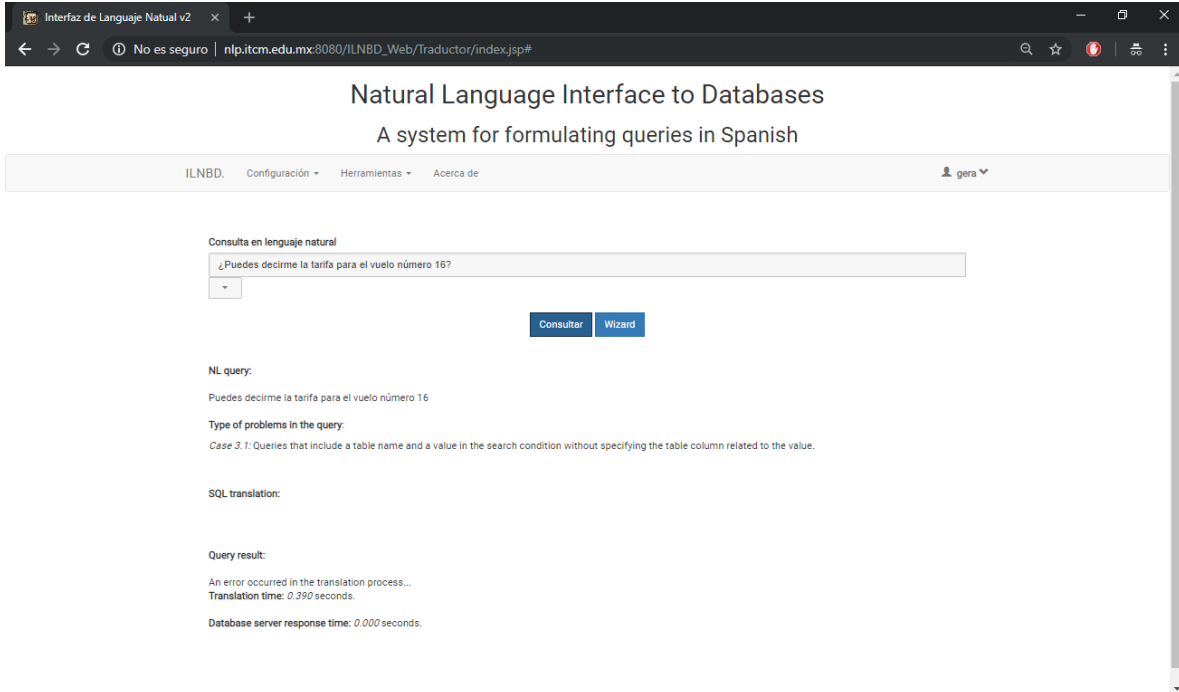


Figura 5.12 Resultado de una consulta en LN con problema de columnas no asociadas

Para corregir este error, el DBA debe utilizar el wizard para introducir la traducción a SQL correcta como se muestra en la Figura 5.13.

```
SELECT fare.one_way_cost, fare.rnd_trip_cost
FROM fare, flight, flight_fare
WHERE fare.fare_code= flight_fare.fare_code
AND flight_fare.flight_code = flight.flight_code AND flight.flight_number = 16.
```

El wizard compara la información recopilada durante el análisis de la consulta con la información de la expresión de SQL correcta, como se muestra en la Figura 5.14 y Figura 5.15. De esta manera supone que las palabras sobrantes en la consulta en LN (*vuelo número*) deben referirse a la columna sobrante en la expresión de SQL (*flight.flight_number*) que el DBA escribió. Por lo tanto, el wizard sugiere que el DBA asigne la frase *vuelo número* al descriptor de la columna *flight.flight_number* en el DIS [Aguirre, 2014], tal como se ilustra en la Figura 5.17.

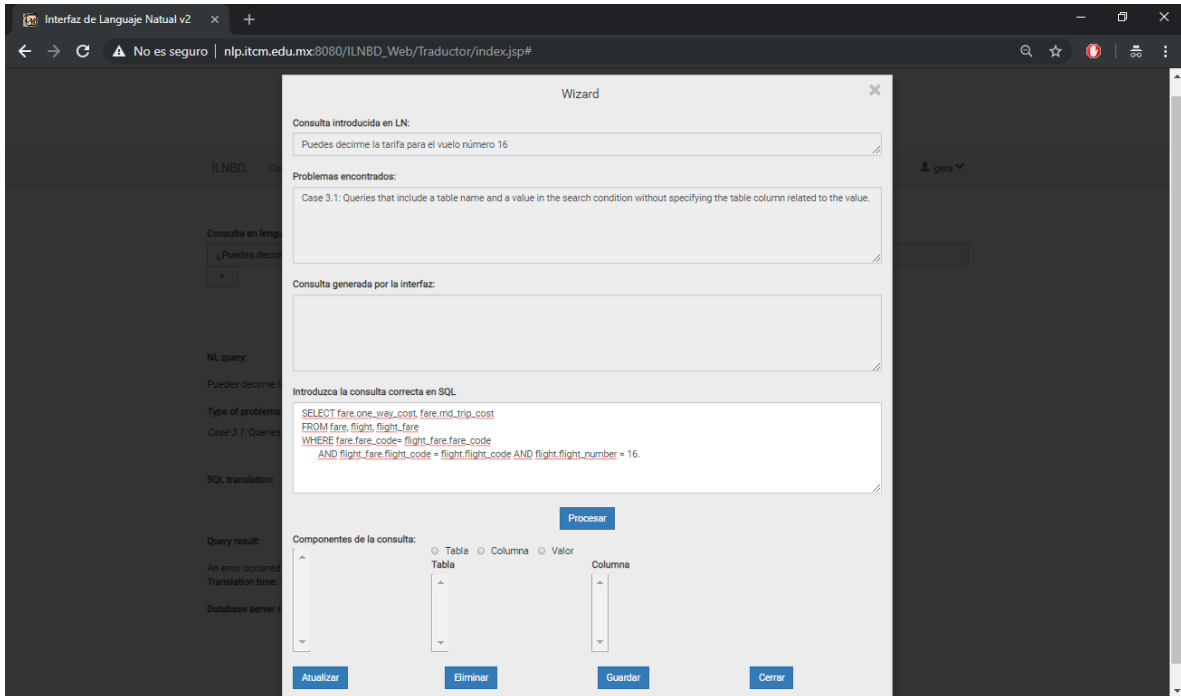


Figura 5.13 Consulta introducida por el DBA

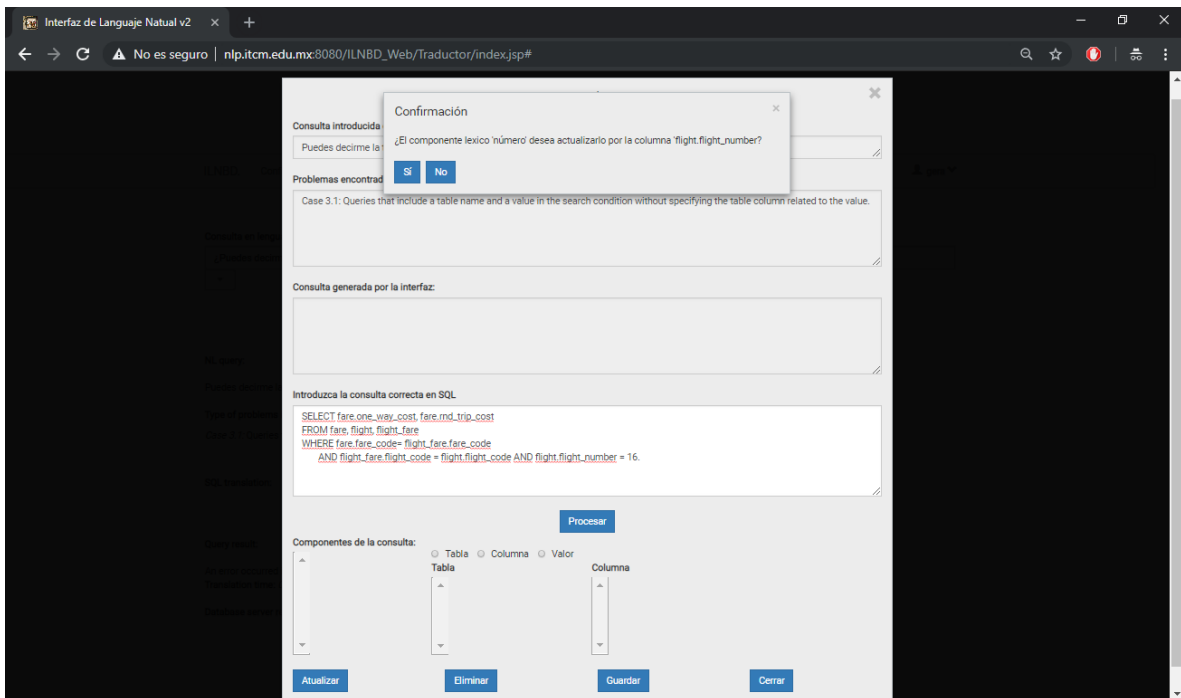


Figura 5.14 Sugiere asociar a la palabra *número* a la columna *flight.flight_number*

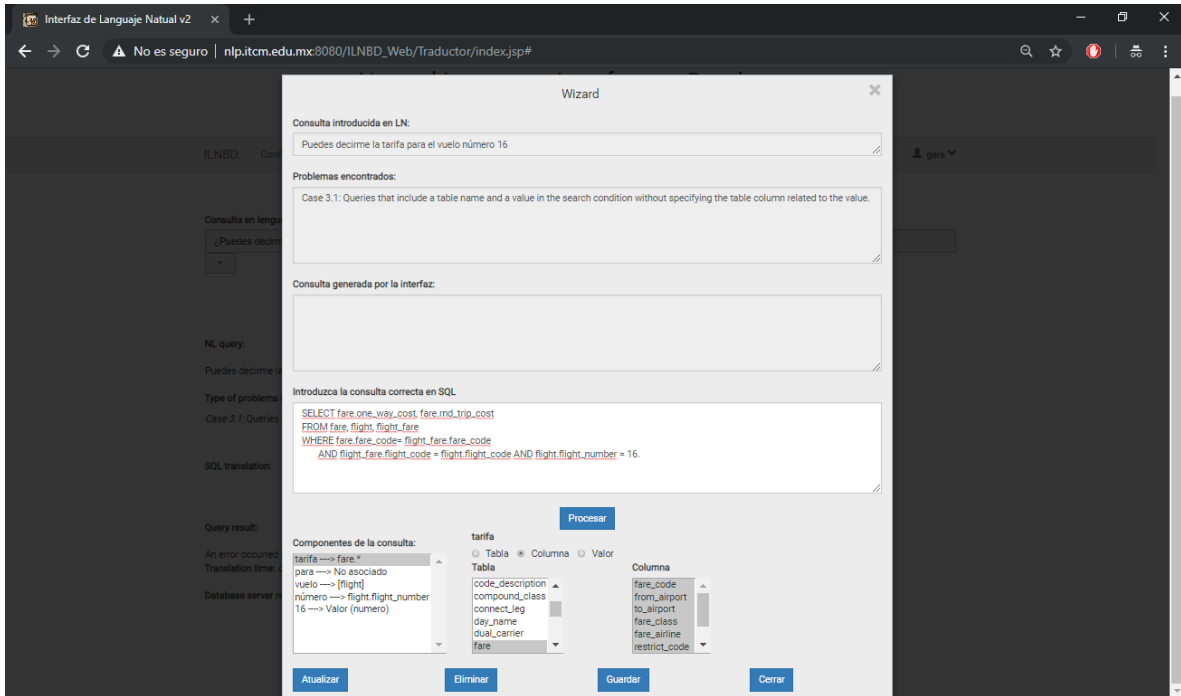


Figura 5.15 Componentes de la consulta

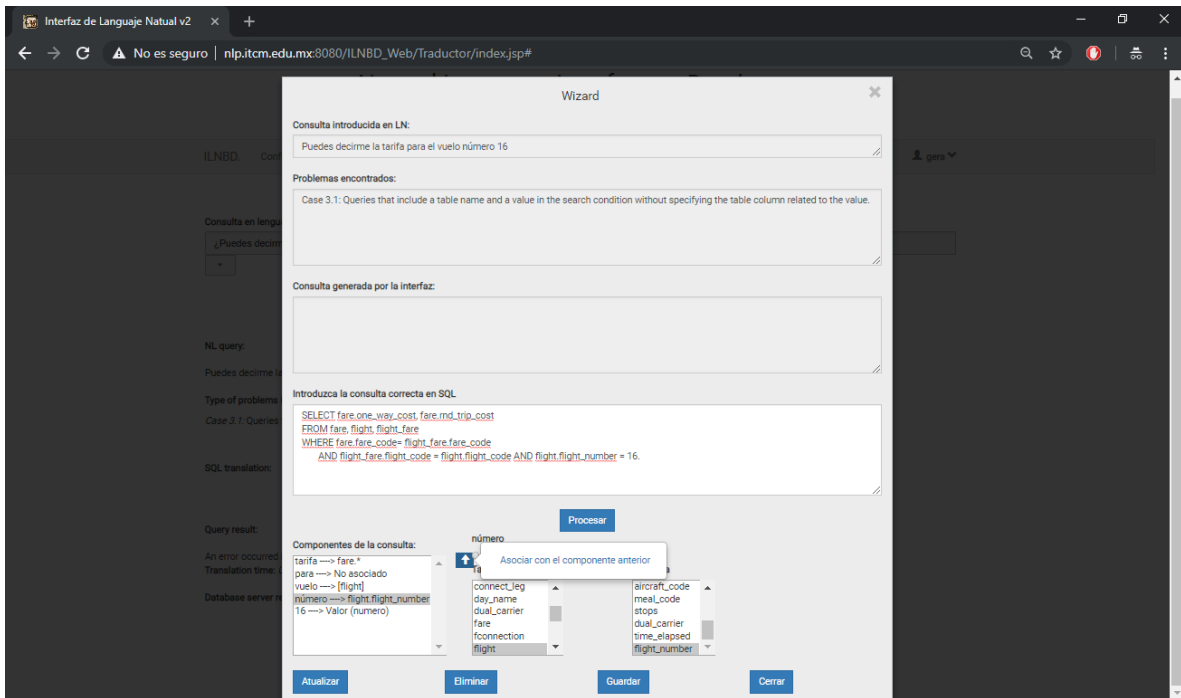


Figura 5.16 Asociar la palabra *número* con la palabra *vuelo*

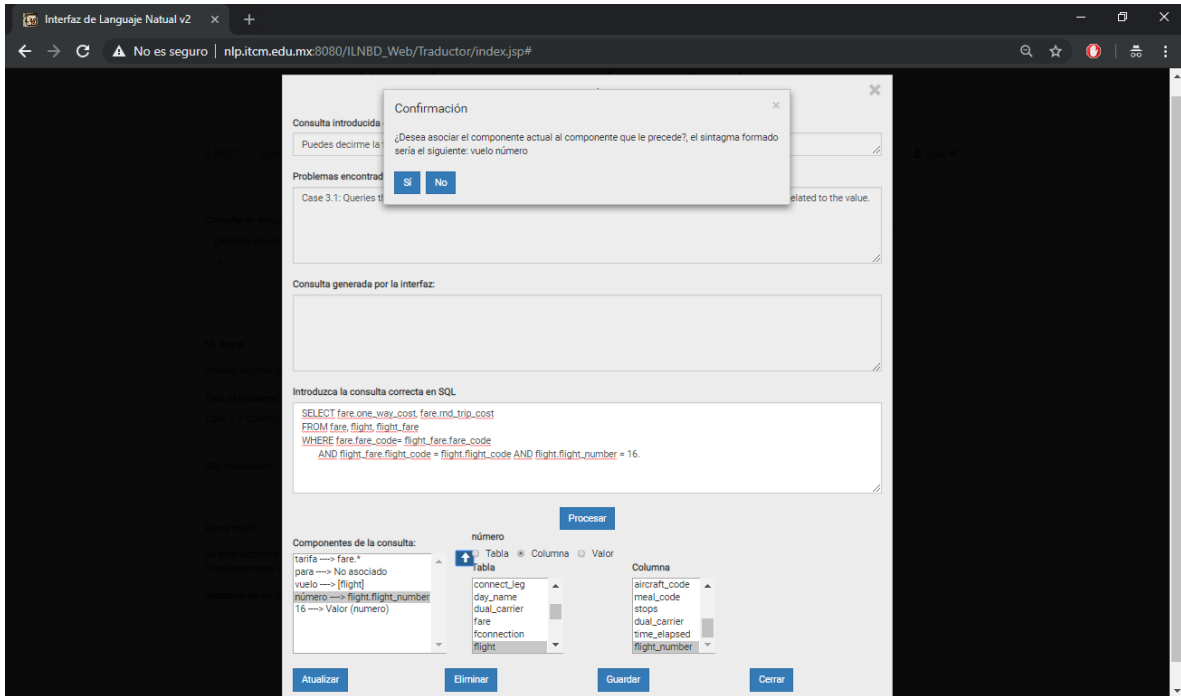


Figura 5.17 El sintagma formado es *vuelo número*

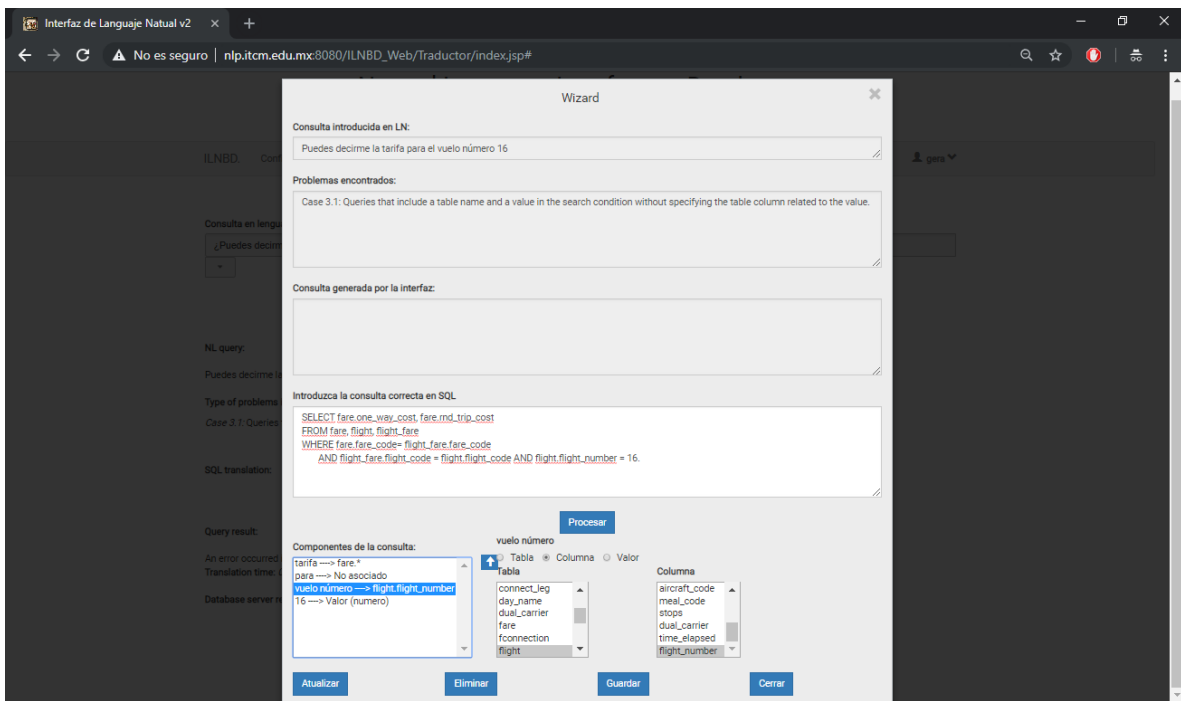


Figura 5.18 Cambio realizado en el componente de la consulta

Con respecto a la otra palabra sobrante, *tarifa*, la Figura 5.19 muestra el resultado del proceso de asociación para *tarifa*.

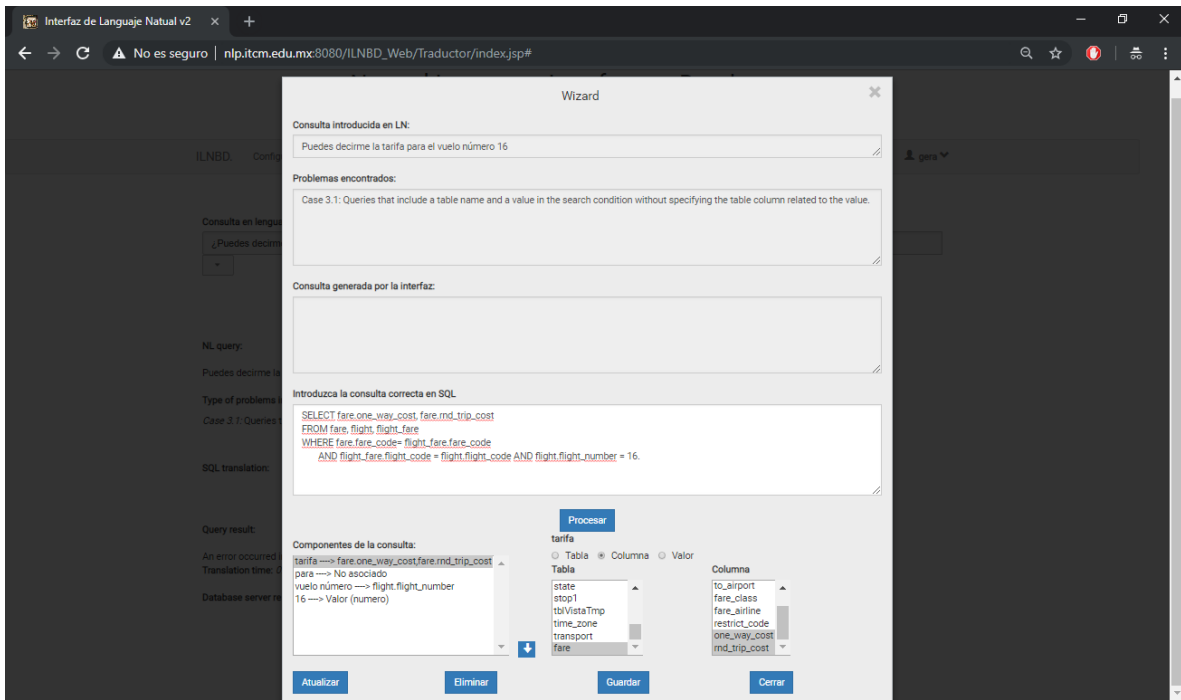


Figura 5.19 Actualización del componente de la palabra *tarifa*

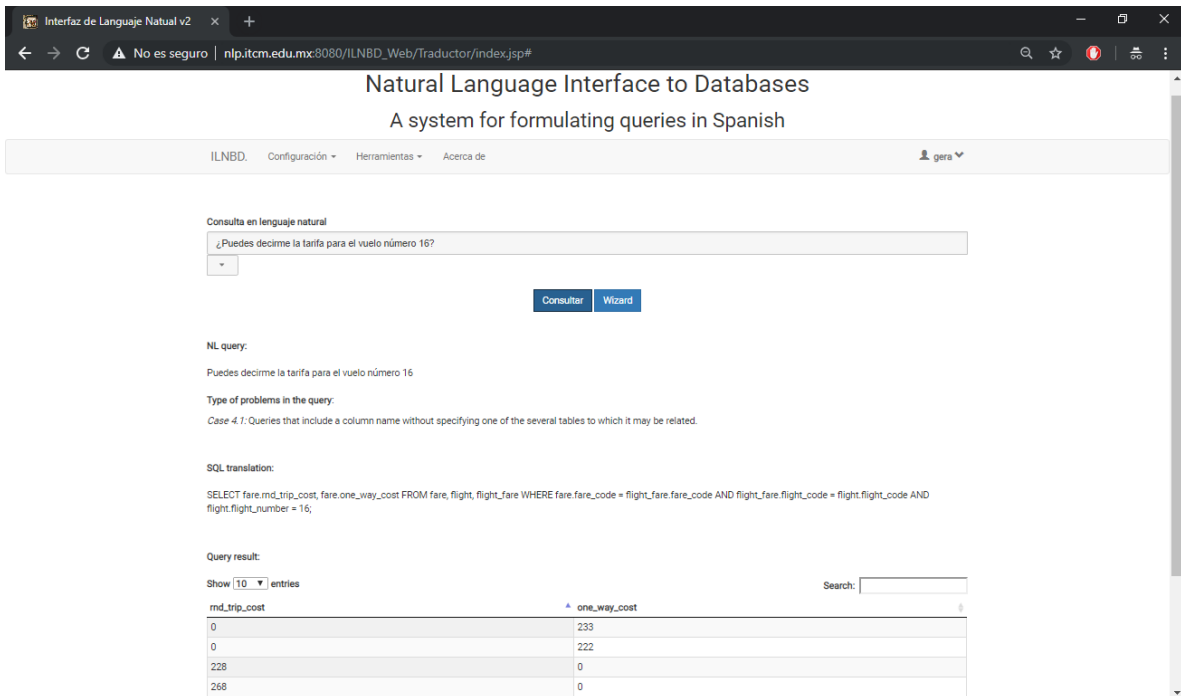


Figura 5.20 Resultado de la consulta en LN después de utilizar el wizard

Prueba No. 8

Problemas de tablas o columnas mal asociadas.

Resultado:

La siguiente consulta presenta problemas de tablas o columnas mal asociadas. Este tipo de error involucra aquellas palabras en la consulta en LN que durante su traducción a SQL fueron etiquetadas con un nombre de columna o tabla de base de datos incorrecto [Aguirre, 2014].

Un ejemplo de consulta que presenta problema de tablas o columnas mal asociadas es la siguiente consulta:

¿Cuánto cuesta un viaje redondo desde BOS a DFW?

Al introducir esta consulta la ILNBD de escritorio muestra el resultado en la Figura 5.21.

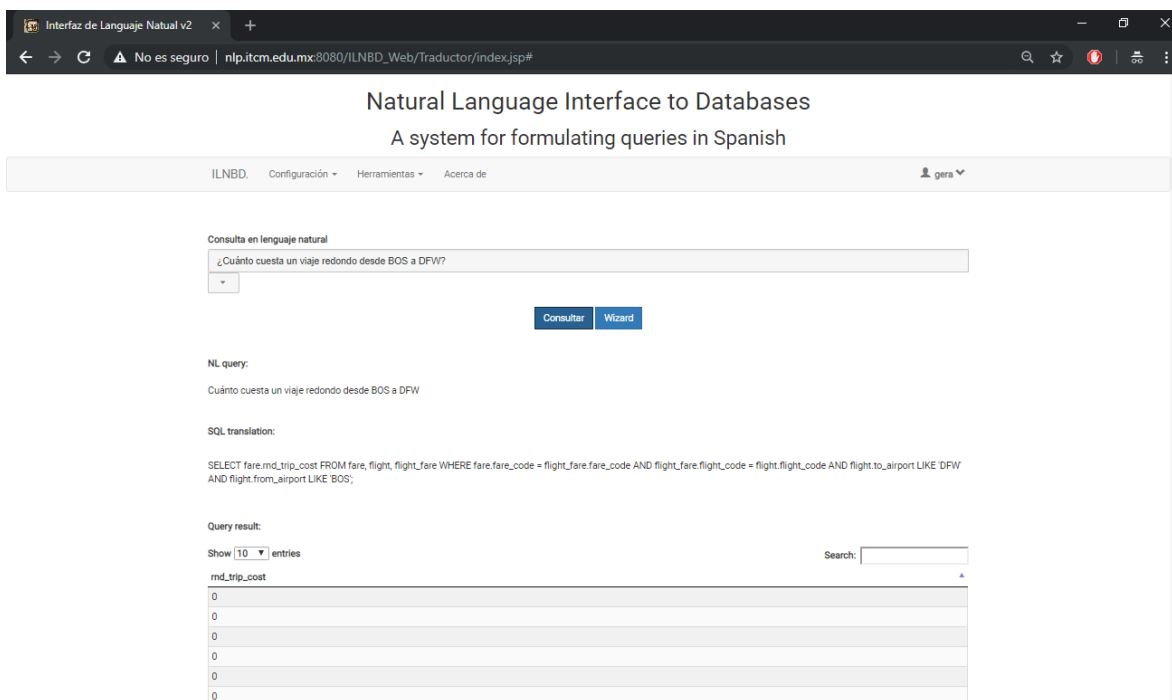


Figura 5.21 Resultado de una consulta en LN con problema de columnas mal asociadas

La ILNBD generó como resultado una consulta en SQL que no es la que el usuario esperaría como respuesta, debido a que el diccionario tenía un error (omisión) en su configuración.

```
SELECT fare.rnd_trip_cost
FROM fare, flight, flight_fare
WHERE fare.fare_code = flight_fare.fare_code
AND flight_fare.flight_code = flight.flight_code
```

AND flight.from_airport LIKE 'BOS'
AND flight.to_airport LIKE 'DFW'.

Para corregir este problema el DBA debe introducir la siguiente expresión en SQL (ver Figura 5.22).

SELECT fare.rnd_trip_cost
FROM fare
WHERE fare.from_airport LIKE 'BOS'
AND fare.to_airport LIKE 'DFW'.

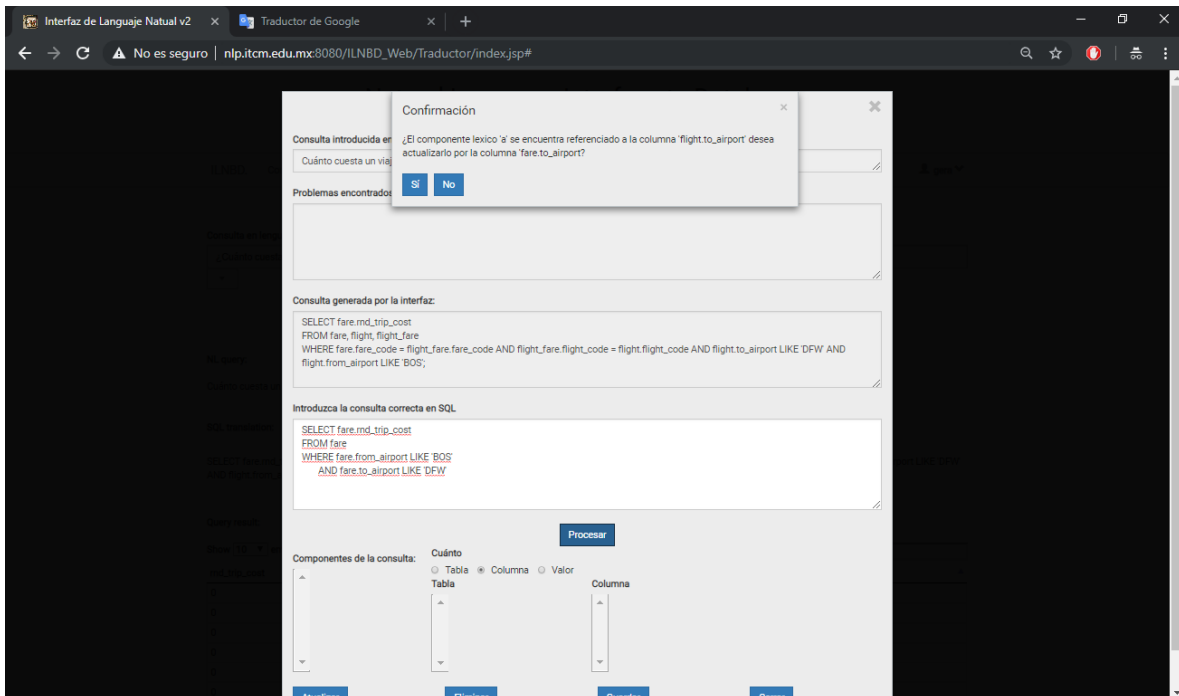


Figura 5.22 Sugiere actualizar la referencia de la palabra *a* de la columna *flight.to_airport* a la columna *fare.to_airport*

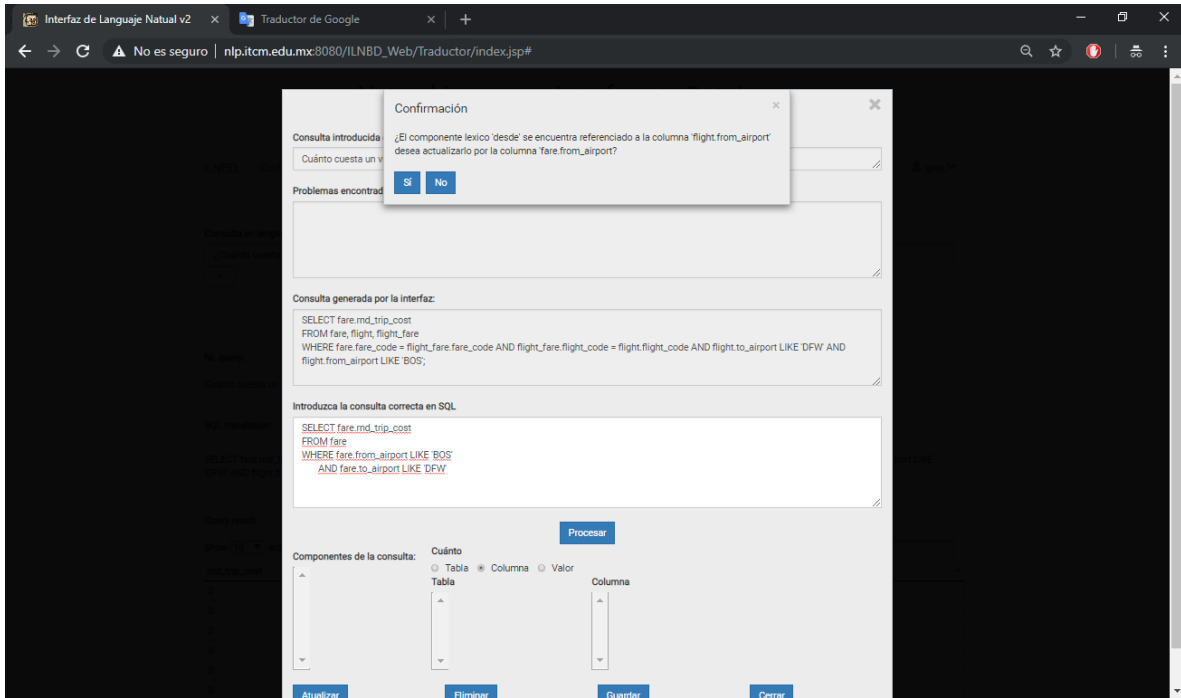


Figura 5.23 Sugiere actualizar la referencia de la palabra *desde* de la columna *flight.from_airport* a la columna *fare.from_airport*

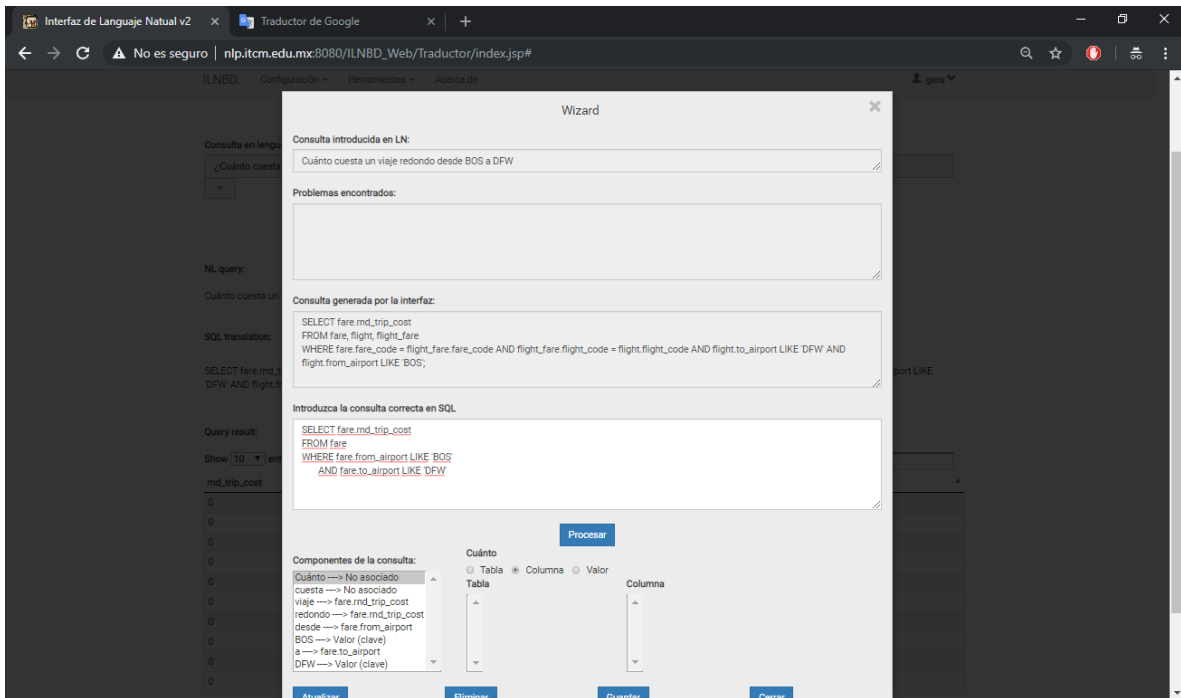


Figura 5.24 Componentes actualizados de la consulta

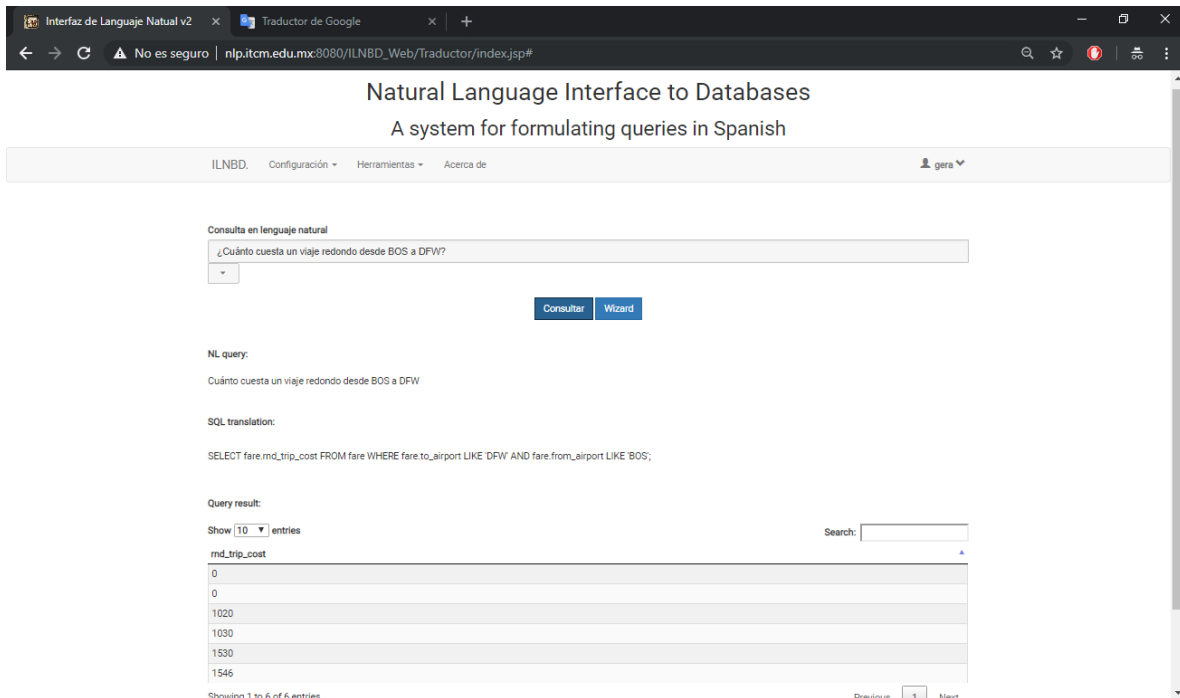


Figura 5.25 Resultado de la consulta en LN después de utilizar el wizard

Prueba No. 9

Problemas de valores imprecisos o alias no asociados.

Resultado:

Las siguientes consultas involucran problemas de valores imprecisos o alias no asociados. Un valor impreciso se refiere a un rango de valores; por ejemplo, mañana (que denota un rango de tiempo entre 0:00 y 11:59 h.), tarde, noche, etc. [Aguirre, 2014]. En cambio, un valor alias es una palabra (o frase) alternativa para referirse a algún valor que se espera que se encuentre en la base de datos; por ejemplo, medianoche (equivalente a 0:00 h), mediodía, docena, etc. [Aguirre, 2014].

Un ejemplo de consulta que involucra un problema de valor impreciso es la siguiente consulta:

Lista todos los vuelos desde DFW a BOS en la mañana

El problema con esta consulta es que, en la configuración inicial, la palabra *mañana* no se almacenó en el DIS y, por lo tanto, la interfaz la ignora [Aguirre, 2014].

Al introducir esta consulta la ILNBD de escritorio muestra el resultado en la Figura 5.26.

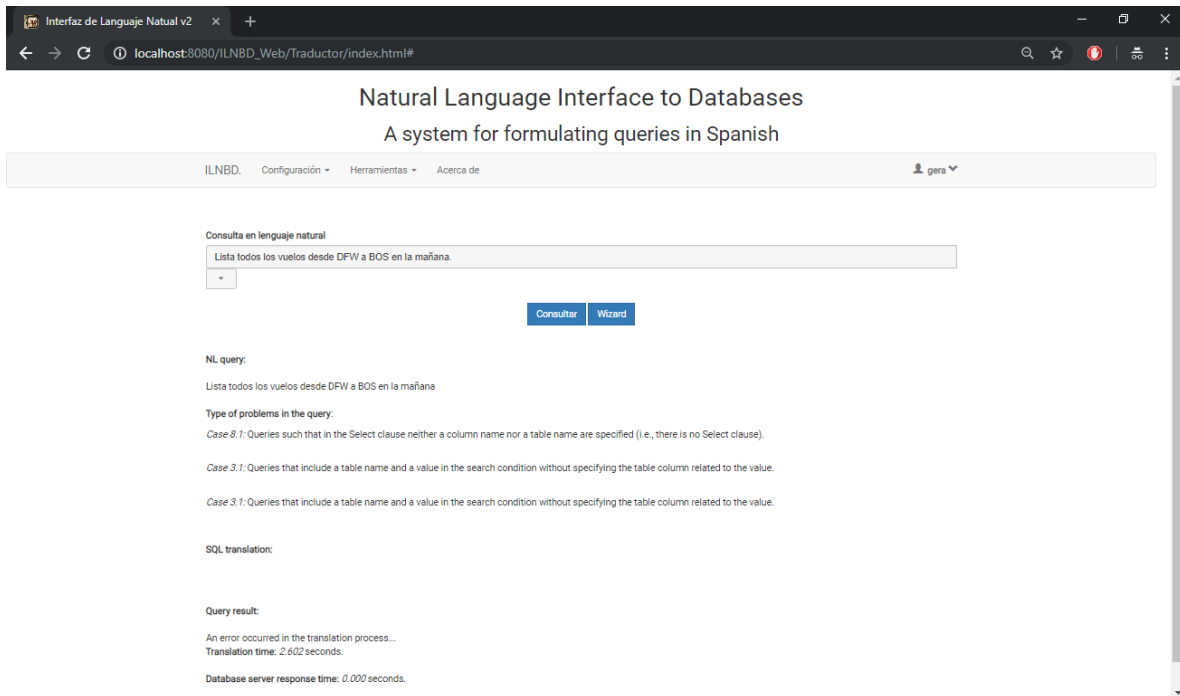


Figura 5.26 Resultado de una consulta en LN con problema de valor impreciso no asociado

Para corregir este problema, el DBA debe utilizar el wizard como se muestra en la Figura 5.27 para introducir la siguiente consulta:

```
SELECT flight.flight_number
FROM flight
WHERE flight.departure_time BETWEEN 0000 AND 1159
AND flight.to_airport LIKE 'BOS' AND flight.from_airport LIKE 'DFW'.
```

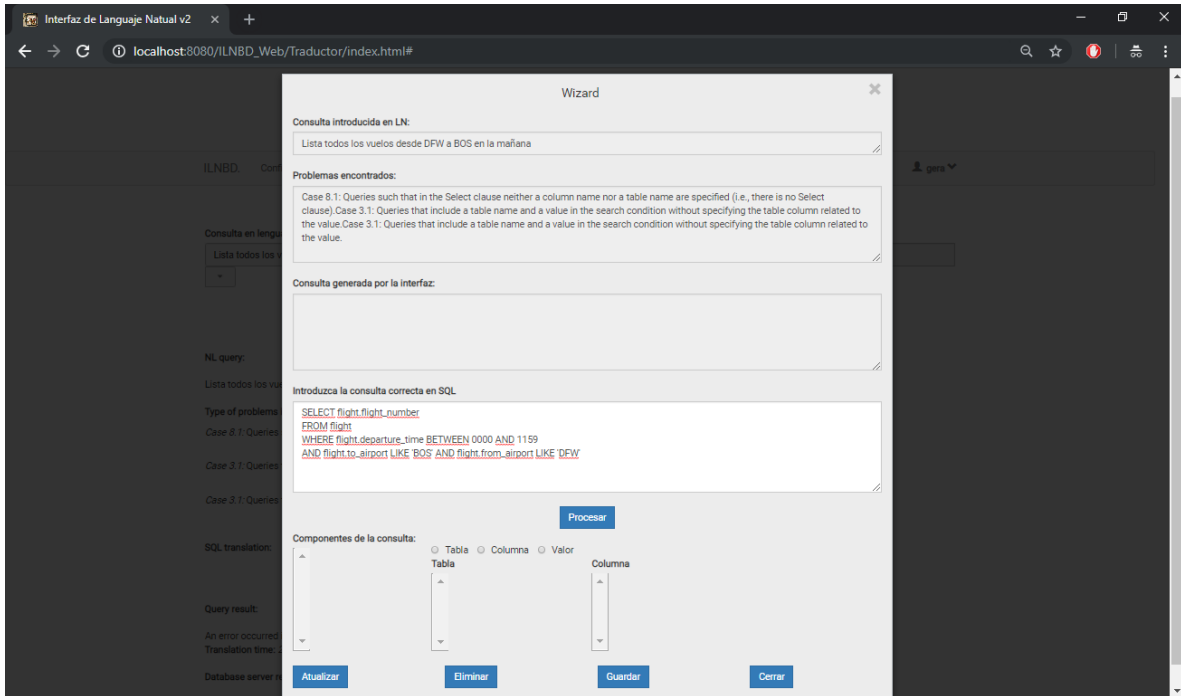


Figura 5.27 Consulta introducida por el DBA

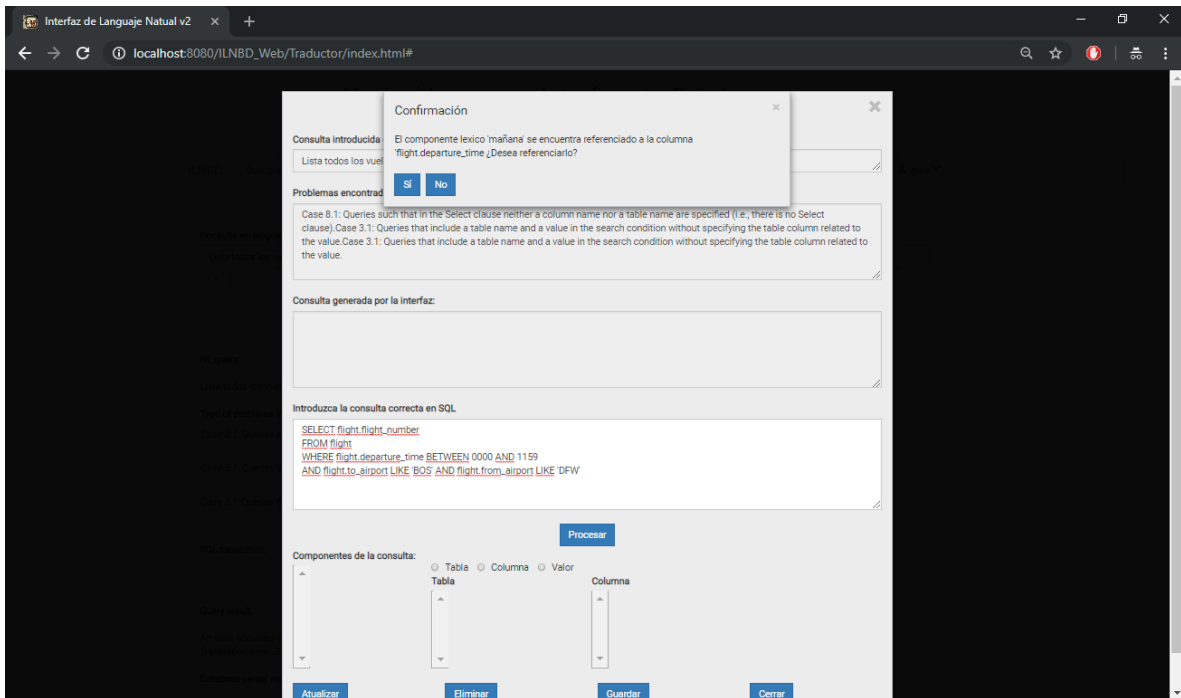


Figura 5.28 Sugiere asociar a la palabra *mañana* a la columna *flight.departure_time*

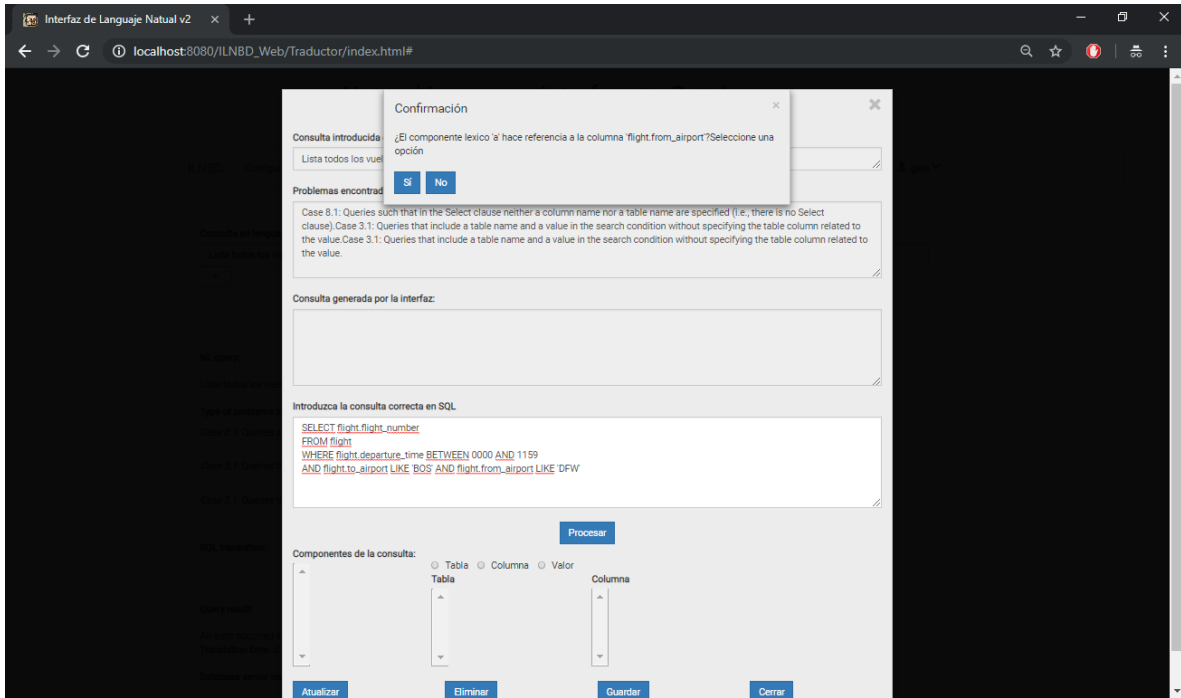


Figura 5.29 Sugiere asociar a la palabra *a* a la columna *flight.from_airport*

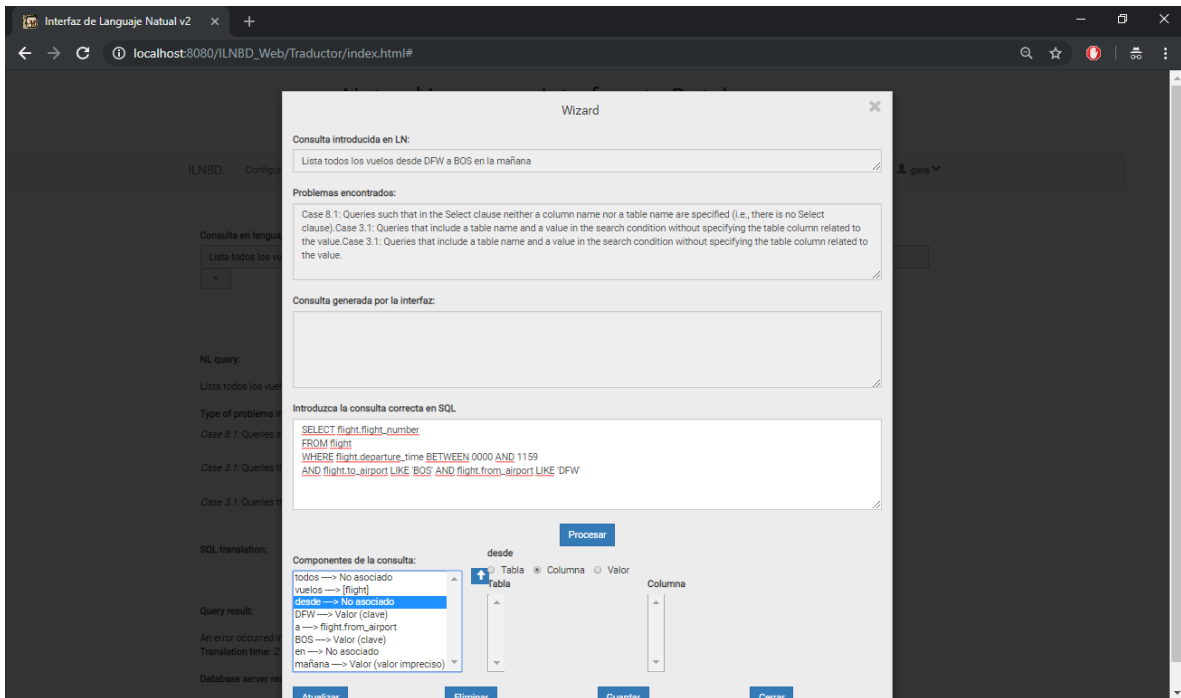


Figura 5.30 Componentes actualizados de la consulta

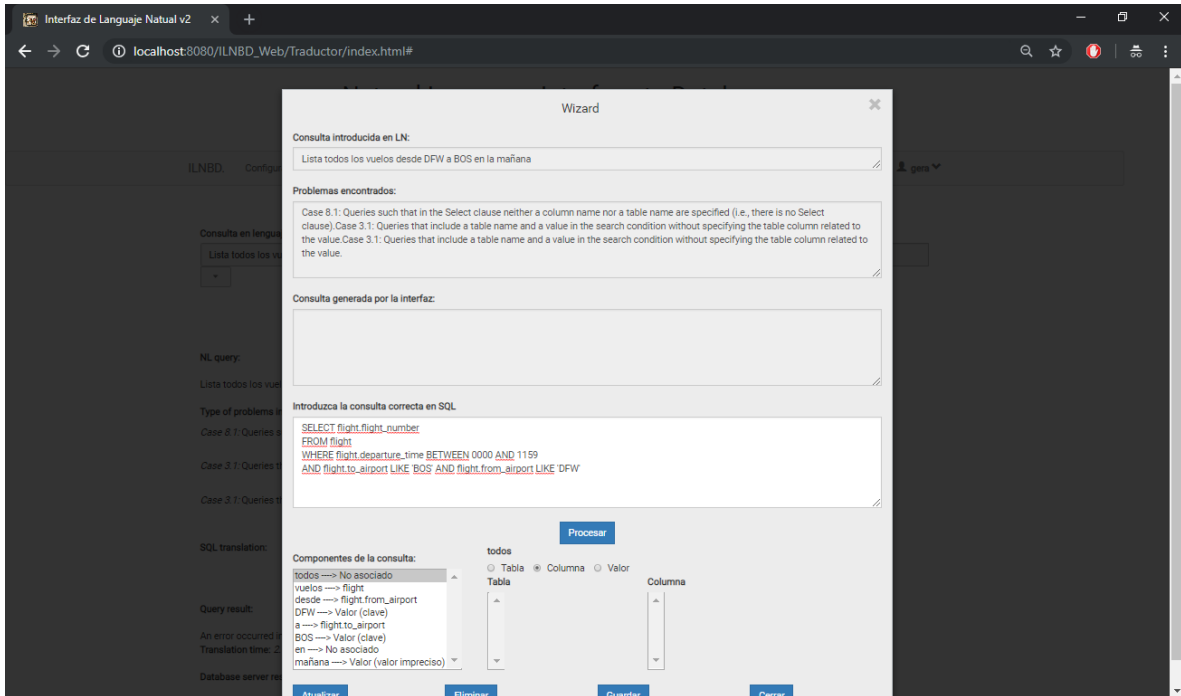


Figura 5.31 Actualización de las palabras *desde* y *a* para los descriptores de las columnas *flight.from_airport* y *flight.to_airport*

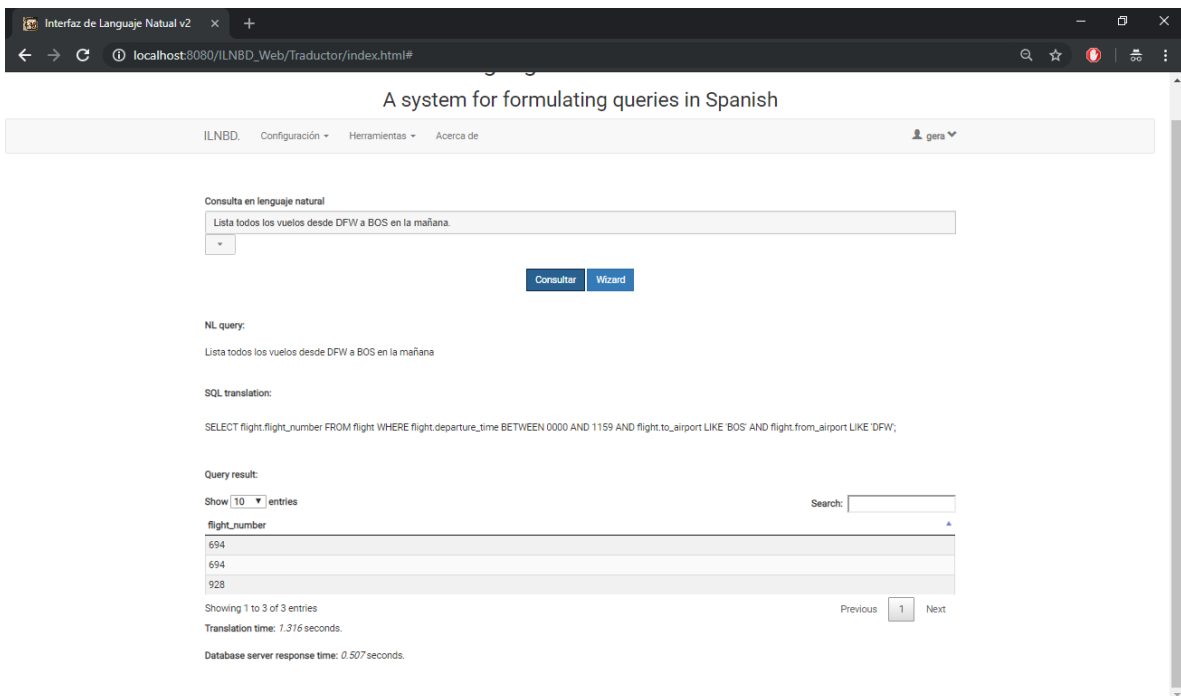


Figura 5.32 Resultado de la consulta en LN después de utilizar el wizard

Un ejemplo de consulta que involucra un problema de valor alias es la siguiente consulta:

Muéstrame vuelos que salgan después del mediodía.

El problema con esta consulta es que, en la configuración inicial, la palabra *mediodía* no se almacenó en el DIS y, por lo tanto, la interfaz la ignoró y generó la siguiente traducción errónea.

```
SELECT flight.* FROM flight;
```

Al introducir esta consulta la ILNBD de escritorio muestra el resultado en la Figura 5.33.

The screenshot shows the 'Interfaz de Lenguaje Natural v2' web application. The user has entered the natural language query: 'Muéstrame vuelos que salgan después del mediodía.' The system has translated this into the SQL query: 'SELECT flight.* FROM flight;'. Below the translation, the 'Query result' section displays a table with 10 entries. The table columns are: flight_code, flight_days, from_airport, to_airport, departure_time, arrival_time, airline_code, flight_number, class_string, aircraft_code, and meal_code. The results show flights from ATL to BOS with various departure and arrival times.

flight_code	flight_days	from_airport	to_airport	departure_time	arrival_time	airline_code	flight_number	class_string	aircraft_code	meal_code
101908	1234567	ATL	BOS	636	1000	DL	296	FVYBENMQ	72S	B
101909	1234567	ATL	BOS	641	855	DL	314	FVYBENMQ	72S	B
101910	1234567	ATL	BOS	755	1019	EA	140	FYHJK	D9S	B
101911	1234567	ATL	BOS	920	1150	EA	534	FYHJK	D9S	B
101912	1234567	ATL	BOS	959	1215	DL	410	FYBMQ	757	B
101913	1234567	ATL	BOS	1010	1355	DL	726	FYBMQ	72S	S

Figura 5.33 Resultado de una consulta en LN con problema de valor alias no asociado

La ILNBD generó como resultado una consulta en SQL; sin embargo, esta consulta en SQL no es la que el usuario esperaría como respuesta. Para corregir este problema el DBA debe utilizar el wizard como se muestra en la Figura 5.34 para introducir la siguiente expresión en SQL.

```
SELECT flight.flight_number  
FROM flight  
WHERE flight.departure_time > 1200.
```

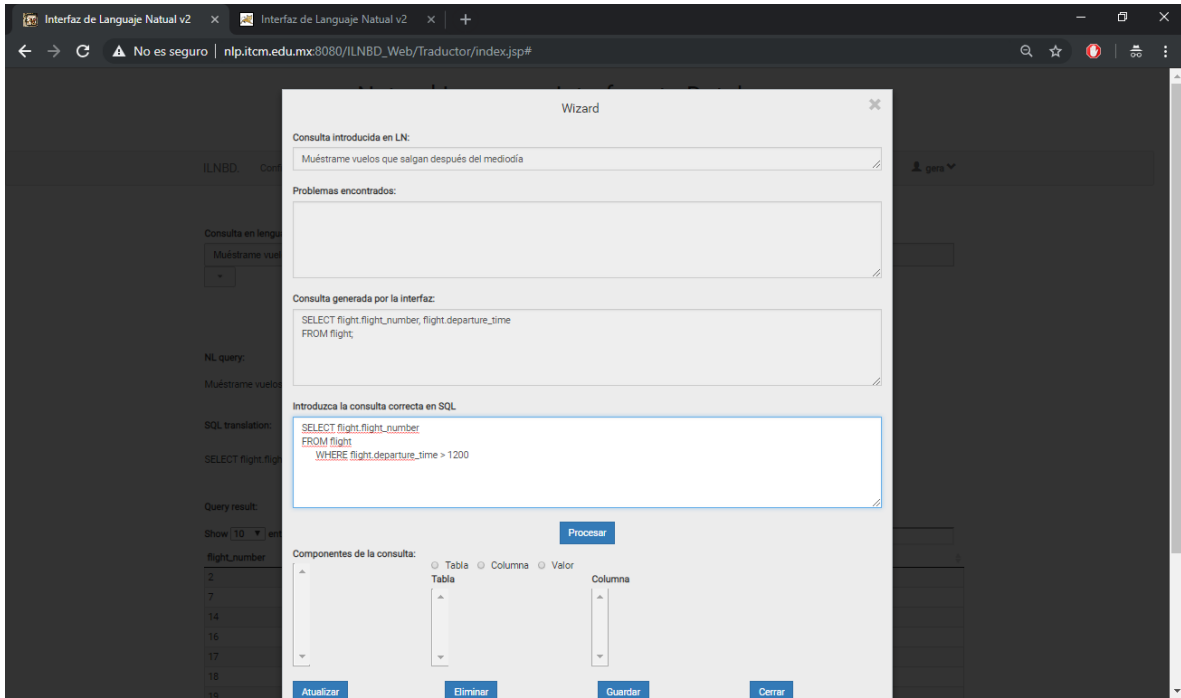


Figura 5.34 Consulta introducida por el DBA

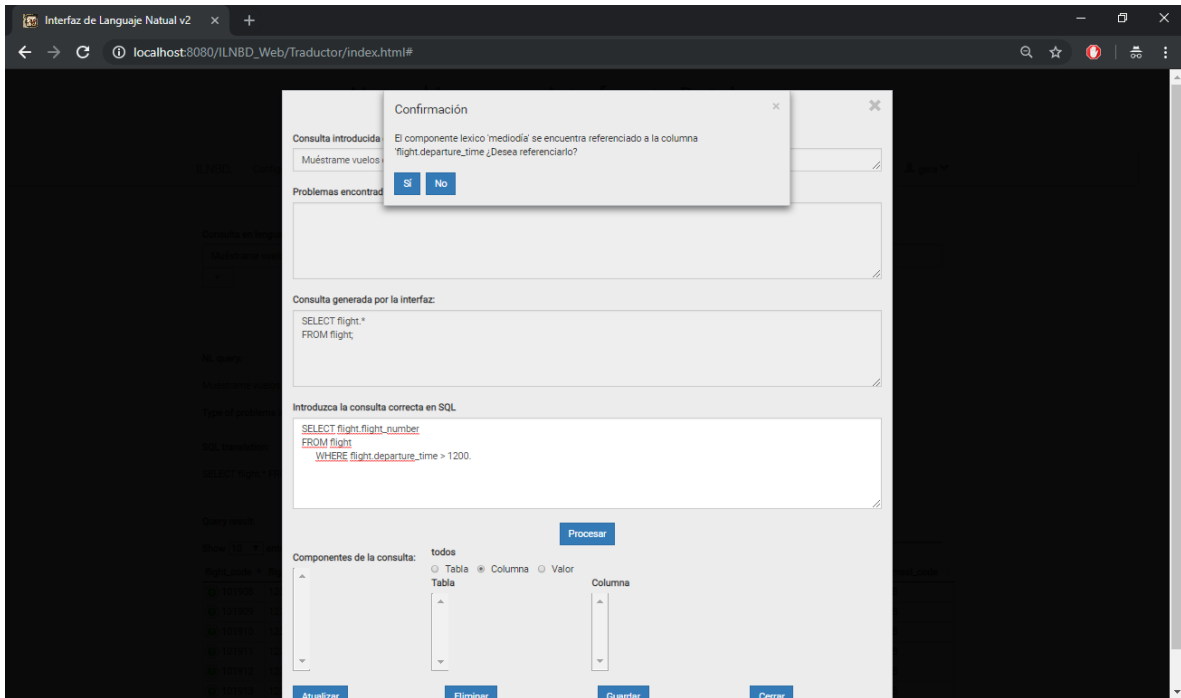


Figura 5.35 Sugiere asociar la palabra *mediodía* a la columna *flight.departure_time*

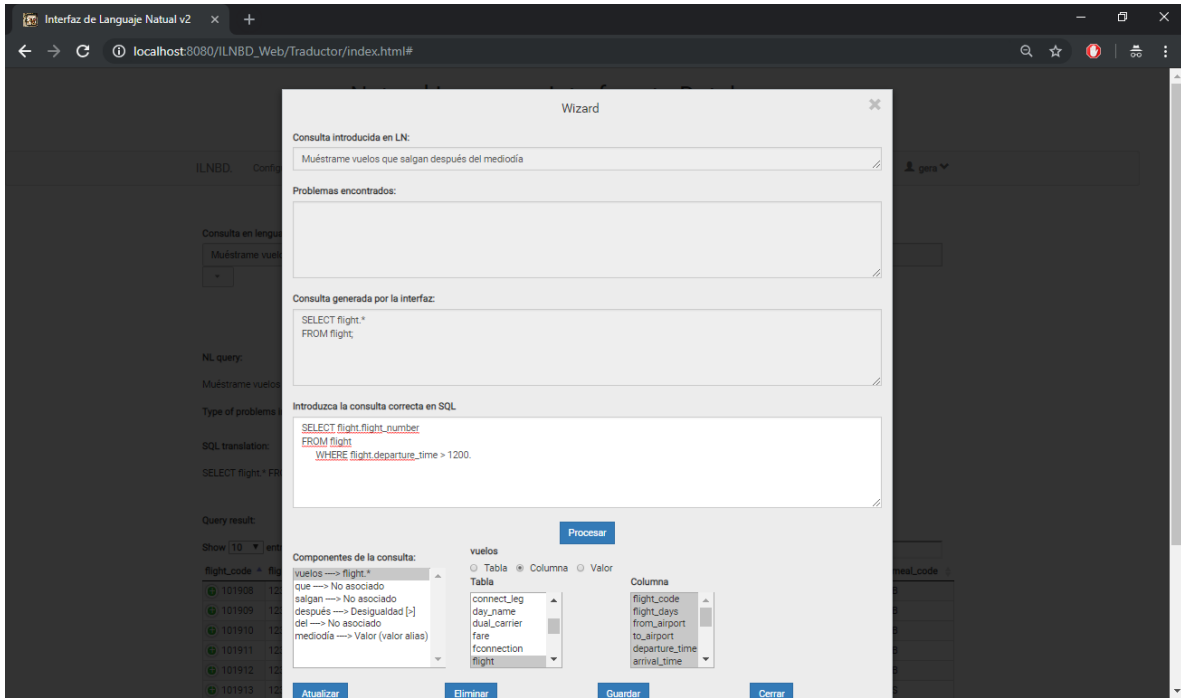


Figura 5.36 Componentes actualizados de la consulta

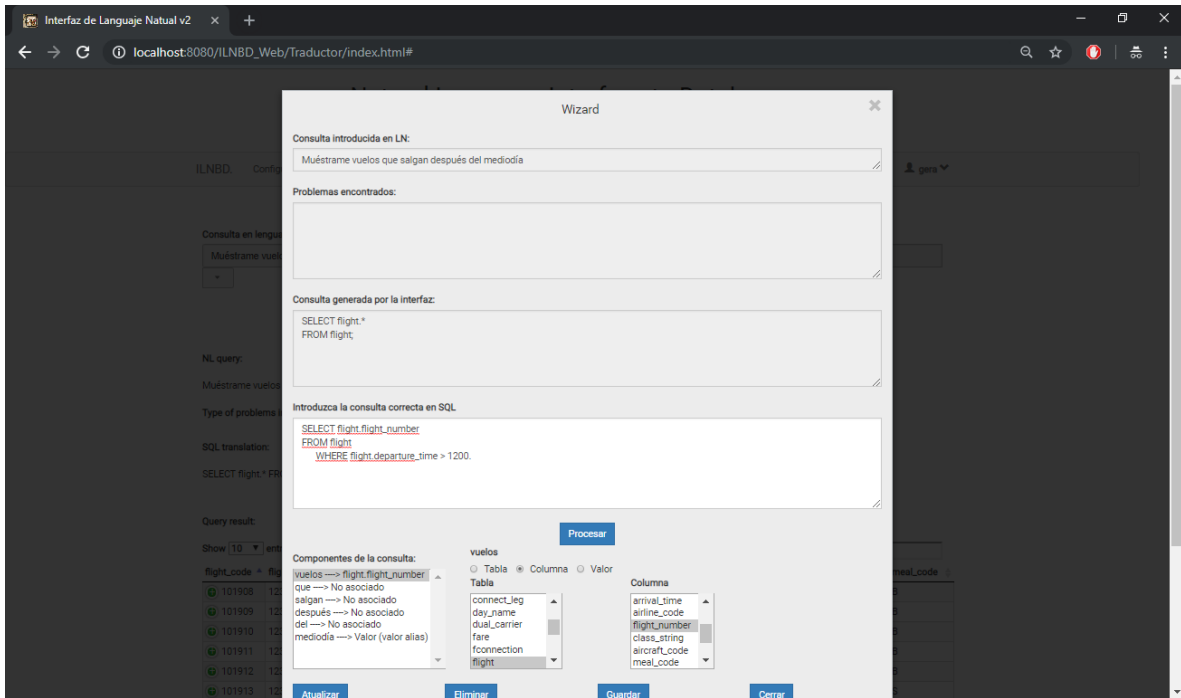


Figura 5.37 Actualización de la palabra *vuelos* para los descriptores de la columna *flight.flight_number*

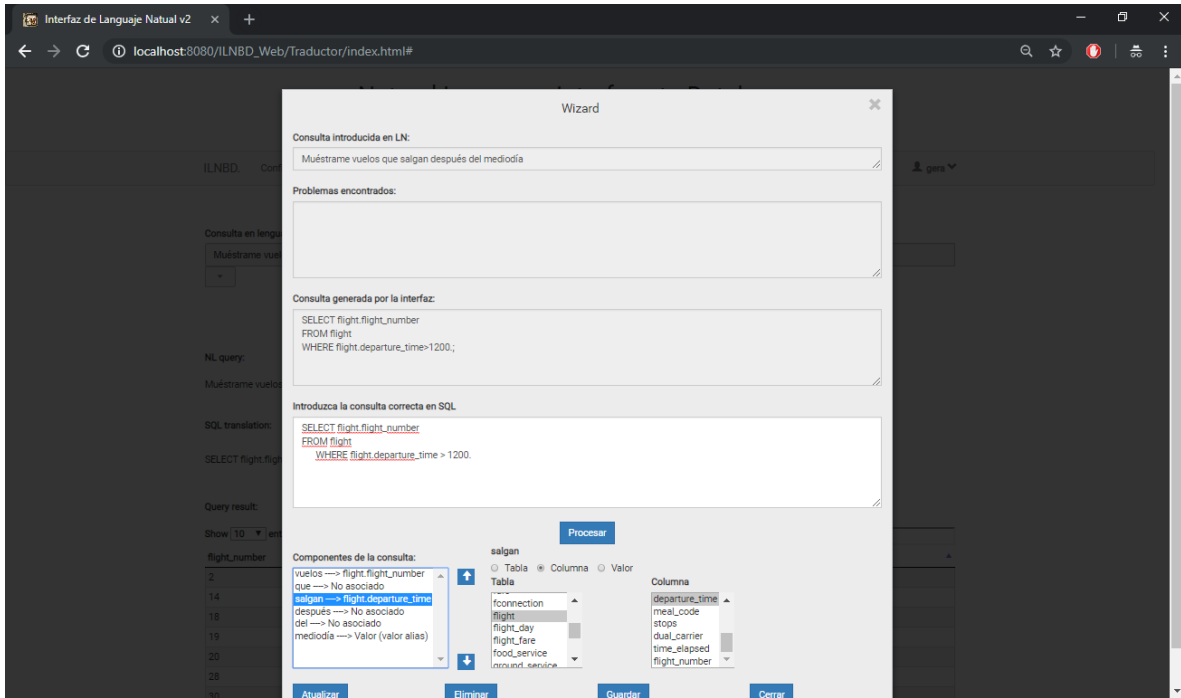


Figura 5.38 Actualización de la palabra *salgan* para los descriptores de la columna *flight.departure_time*

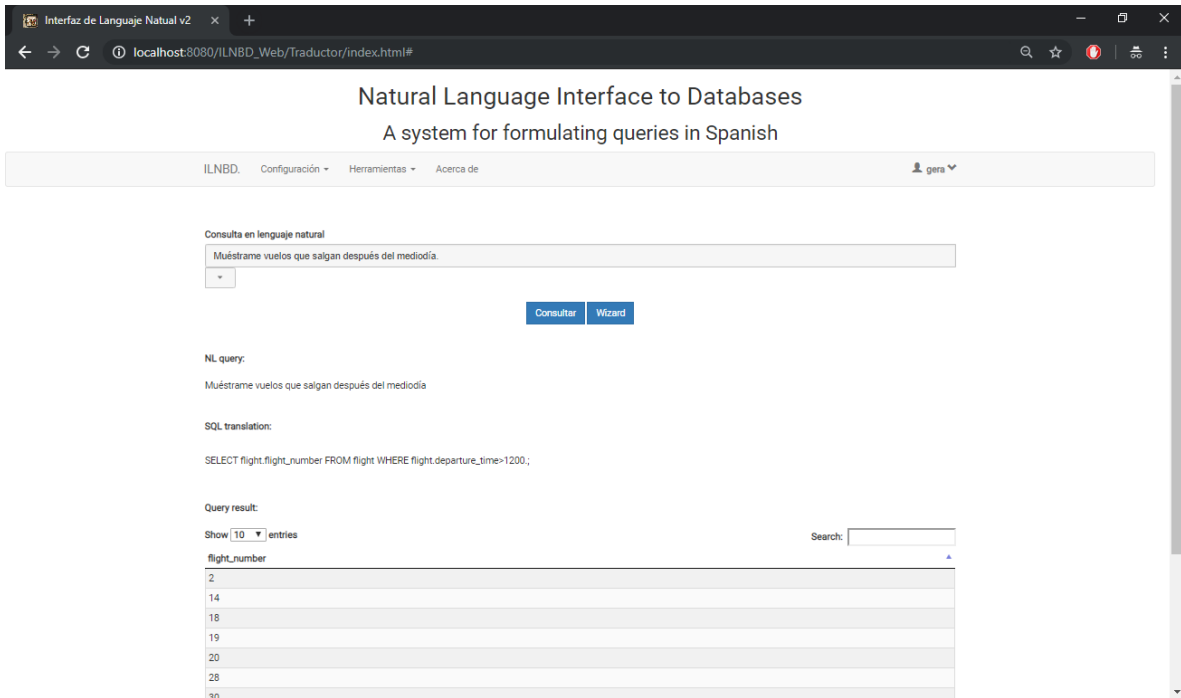


Figura 5.39 Resultado de una consulta en LN después de utilizar el wizard

Prueba No. 10

Configuraciones del diccionario de información semántica.

Resultado:

En esta prueba se realizó la configuración de un diccionario de información semántica generado por la ILNBD para web y por la ILNBD de escritorio de la base de datos ATIS. El diccionario de información semántica se configuró de dos maneras diferentes. Primero se realizó la configuración automática en ambas ILNBDs, la cual se realiza cuando se genera un diccionario de información semántica por primera vez. El resultado obtenido (ver Tabla 5.2) fue que en ambas ILNBDs dio un total de 17 consultas contestadas de un total de 70 consultas. Para la configuración con ayuda del wizard, la ILNBD para web obtuvo un resultado de 60 consultas correctamente contestadas de un total de 70 consultas. La ILNBD de escritorio, al ser configurada con ayuda del wizard, también obtuvo un resultado de 60 consultas correctamente contestadas de un total de 70 consultas.

Tabla 5.2 Comparación de resultados obtenidos en las configuraciones del DIS

ILNBD	BD	Total de consultas	Configuración inicial Consultas correctas	Afinación con wizard Consultas correctas	Porcentaje Configuración inicial	Porcentaje Afinación con wizard
ILNBD de escritorio	ATIS	70	17	60	24.28%	85%
ILNBD para web	ATIS	70	17	60	24.28%	85%

5.4 Resultados de las pruebas

De acuerdo con los casos de prueba mencionados anteriormente, los resultados de las pruebas efectuadas son satisfactorios, con lo cual se logró cumplir el objetivo mencionado en el Capítulo 1. Con estas pruebas se demuestra que la ILNBD para web puede ser accedida vía internet por más de un usuario, permitiendo realizar el procesamiento de consultas en lenguaje natural y configurar el diccionario de información semántica sin provocar interferencias entre varios usuarios que usen simultáneamente la ILNBD.

Debido a que el punto de partida para la implementación de ILNBD para web fue la ILNBD de escritorio, el funcionamiento debería ser el mismo. Esto se demostró con la prueba No. 10, en donde se muestran los resultados obtenidos al realizar la configuración del diccionario de información semántica creado en ambas ILNBDs. En lo que respecta a la operación del wizard, la ILNBD para web contesta correctamente 60 consultas de 70; mientras que la ILNBD de escritorio también contestó 60 consultas de 70.

La ILNBD para web se puede acceder a través de la siguiente dirección: http://nlp.itcm.edu.mx:8080/ILNBD_Web/Traductor/Main.html.

Capítulo 6

Conclusiones

En este capítulo se presentan las conclusiones generales, los resultados obtenidos, las contribuciones del proyecto y posibles trabajos futuros.

6.1 Contribución del Proyecto

El desarrollo de este proyecto de tesis tuvo como propósito implementar una nueva versión de una ILNBD que permita formular consultas en lenguaje natural y además afinar la configuración de la ILNBD con ayuda del wizard a través de internet. La ILNBD para web permite el fácil acceso al tipo de información que se guarda dentro de la base de datos, lo cual es una ventaja muy importante para un usuario casual de la red internet.

6.2 Conclusiones generales

Existen diferentes ILNBDs que permiten formular consultas en lenguaje natural, pero sólo una interfaz de lenguaje natural, aparte de la ILNBD de escritorio desarrollada por Aguirre, cuenta con un asistente (wizard). Esta ILNBD era English Query que proporcionaba un entorno para desarrollar un modelo de consultas en inglés, además de contar con un asistente para proyectos de consultas en inglés (English Query Project Wizard). Este asistente permitía definir automáticamente algunas entidades y relaciones basadas en la estructura de la base de datos; sin embargo, English Query fue quedando en el olvido por sus creadores, ya que actualmente no está integrada en la instalación de SQL Server.

Actualmente únicamente existe la ILNBD de escritorio desarrollada por Aguirre que cuenta con un asistente (wizard). El desarrollo de este proyecto de tesis tiene como antecedente la ILNBD de escritorio, la cual sirvió como elemento clave para la implementación de esta nueva versión.

Este proyecto consiste en la implementación de una ILNBD para web que permite formular consultas y además permite configurar el diccionario de información semántica con ayuda de un asistente (wizard) a través de internet. De acuerdo con los resultados obtenidos en las pruebas, la ILNBD para web permite realizar sus funciones de manera correcta al igual que la ILNBD de escritorio, sin provocar ninguna interferencia debido a que la ILNBD para

web permite el acceso a más de un usuario simultáneamente. Algunas de las ventajas de la ILNBD para web son las siguientes:

- **Fácil acceso:** La ILNBD para web puede ser accedida por cualquier usuario que cuente con un equipo que tenga un navegador de web y una conexión a internet.
- **Velocidad de procesamiento:** El tener todas las funciones encargadas del procesamiento de la ILNBD para web en el lado servidor, permite que se ejecuten las funciones de manera más rápida dependiendo de las especificaciones físicas con las que cuente el equipo que trabaja como servidor, además de la velocidad de la red en la que estén conectados tanto el servidor como el equipo cliente.

6.3 Resultados obtenidos

Los resultados que se obtuvieron al realizar este proyecto de tesis fueron los siguientes:

- Se cuenta con una nueva ILNBD que puede ser accedida a través de internet.
- La ILNBD para web ofrece independencia de plataformas y de sistemas operativos. El único requisito para un usuario es tener acceso a internet y que cuente con un navegador de web.
- La ILNBD para web tiene un núcleo o estructura que permitirá integrar nuevas funciones de manera fácil, ya que cada evento de cada componente se implementó en diferentes páginas JSP.
- La ILNBD para web permite formular consultas en lenguaje natural a través de internet, además de incorporar las funciones con las que cuenta la ILNBD de escritorio.
- La ILNBD para web ofrece un asistente (wizard) que permite modificar el diccionario de información semántica, el cual no estaba incorporado en una versión anterior para web.
- La ILNBD para web ofrece a usuarios más que sólo formular consultas en lenguaje natural, sino también la posibilidad de generar y modificar un diccionario de información semántica de manera más sencilla. Esta posibilidad está disponible para usuarios identificados.

6.4 Trabajos futuros

Es importante mencionar que la ILNBD para web desarrollada en este proyecto de tesis está dirigida principalmente a usuarios casuales e inexpertos, también para aquellos usuarios que estén interesados en interfaces de lenguaje natural. La ILNBD para web, en particular el diseño de la interfaz hombre-máquina, se puede mejorar en varios aspectos.

A continuación se presentan algunos trabajos orientados a mejorar la herramienta.

- La recuperación del usuario y contraseña a través del correo electrónico.
- Que un usuario sea eliminado automáticamente después de tener un tiempo determinado sin haber accedido a la ILNBD para web.
- Mejorar el diseño de la interfaz en general para que tenga una mejor presentación para el usuario.

Referencias

- [Aguirre, 2014] M. A. Aguirre, *Modelo Semánticamente Enriquecido de Bases de Datos para su Explotación por Interfaces de Lenguaje Natural*, tesis de doctorado, Depto. de Posgrado e Investigación, Instituto Tecnológico de Cd. Madero, CD. Madero, México: Tesis de doctorado.
- [Andrade, 1997] G. Andrade, *Supresión de Ambigüedades Léxicas en Galena mediante Métodos Estadísticos*, Tesis de Licenciatura, Universidade Da Coruña, 1997.
- [Androutsopoulos, 1995] I. Androutsopoulos, G.D. Ritchie, P. Thanisch, “Natural Language Interface to Database: An Introduction”, *Journal of Natural Language Engineering*, pp. 1, 1995.
- [bullzip, 2018] bullzip, “Access to PostgreSQL”, <http://www.bullzip.com/products/a2p/info.php>, 2018.
- [Chandra, 2006] Y. Chandra, *Natural Language Interfaces to Databases*, tesis de maestría, University of North Texas, Texas, EE.UU, 2009.
- [Cobo, 2005] A. Cobo, P. Gómez, D. Pérez, & R. Rocha, *PHP y MySQL: Tecnología para el desarrollo de aplicaciones web*, 2005.
- [Cortez, 2009] A. Cortez, H. Vega, J. Pariona, “Procesamiento de Lenguaje Natural”, *Revista de Ingeniería de Sistemas e Informática*, vol. 6, N.º 2, pp. 46, Julio-Diciembre 2009.
- [Dorado, 2005] M. D. Dorado, *Bases de datos en cliente con JavaScript DB*, Editorial Iberpresa, 2005.
- [EcuRed, 2017] EcuRed, “Lenguajes de programación web”, https://www.ecured.cu/Lenguaje_de_Programaci%C3%B3n_Web.
- [Eriksson, 2003] A. F. Eriksson, A. Jonsson, “Some empirical findings on dialogue management and domain ontologies in dialogue systems: Implications from an evaluation of BIRDQUEST”. 2003.
- [Farlex, 2003] Farlex, “TheFreeDictionary”, <http://encyclopedia2.thefreedictionary.com/natural+language>, 2003.

- [Faudón, 2001] S. R. Faudón. *Introducción a los Sistemas de bases de datos*, 7ª edición, Pearson Educación, 2001.
- [Honderich, 2005] M. Minock, *The Oxford Companion to Philosophy*, 2nd edición, editorial Oxford University Press, 2005.
- [Mateu, 2004] C. Mateu, *Desarrollo de aplicaciones web*, 1ª edición, marzo 2004.
- [Microsoft Corporation, 2009] Microsoft Corporation, *English Query Best Practices*, 2009.
- [Minock, 2009] M. Minock, “C-Phrase: A System for Building Robust Natural Language Interfaces to Databases”, Department of Computing Science Umea University, Sweden, 2009.
- [Mora, 2002] S. L. Mora, *Programación de aplicaciones web: Historia, principios básicos y clientes web*, editorial Club Universitario, España, 2002.
- [Oracle Technology Network, 2017] Oracle Technology Network, “JavaServer Pages Overview”, <http://www.oracle.com/technetwork/java/overview-138580.html>, 2017.
- [Osorio, 2008] F. R. Osorio, *Bases de Datos Relacionales. Teoría y Práctica*, 1ª edición, editorial Instituto Tecnológico Metropolitano, 2008.
- [Pazos, 2012] R. Pazos, M. A. Aguirre, “Interfaces de Lenguaje Natural para Consultar Bases de Datos en Español”, *Komputer Sapiens*, vol. 2, pp. 20-32, Diciembre 2012.
- [Pérez, 2009] J. E. Pérez, *Introducción a JavaScript*, pp. 5, 2009.
- [RAE, 2017] Real Academia Española, “Diccionario de lengua española”, <http://dle.rae.es/?id=N7BnIFO>, 2017.
- [Rojas, 2009] J. C. Rojas, *Administrador de Diálogo para una Interfaz de Lenguaje Natural a Bases de Datos*, tesis de doctorado, Depto. de Ciencias Computacionales, Centro Nacional de Investigación y Desarrollo Tecnológico, Cuernavaca, Mor., México, 2009.
- [Schulz, 2009] R. G. Schulz, *Diseño web con CSS*, 1ª edición, editorial Alfaomega, Marcombo, 2009.
- [Spenik, 2003] M. Spenik, O. Sledge, *Microsoft SQL Server 2000 DBA Survival Guide*, 2nd edición, 2003.

- [Stratica, 2002] N. Stratica, *A Natural Language Processor for Querying Cindi*, tesis de maestría, Universidad de Concordia Montreal, Québec, Canada, Septiembre 2002.
- [The Apache Software Foundation, 2017] The Apache Software Foundation, “Apache Tomcat”, <http://tomcat.apache.org/index.html>, 2017.
- [The PHP Group, 2017] The PHP Group, “Manual de PHP: Conceptos básicos”, <http://php.net/manual/es/intro-what-is.php>, 2017.
- [The PostgreSQL Global Development Group, 2017] The PostgreSQL Global Development Group, “Windows installer”, <https://www.postgresql.org/download/windows/>, 2017.
- [The PostgreSQL Global Development Group, 2017] The PostgreSQL Global Development Group, *PostgreSQL 9.6.5 Documentation*, 2017.
- [Vicomtech, 2017] Visual interaction & communication technologies, “Procesamiento de Lenguaje Natural”, <http://www.vicomtech.org/t4/e11/procesamiento-del-lenguaje-natural>, 2017.
- [W3C, 2017] W3C, “jQuery Introduction”, https://www.w3schools.com/jquery/jquery_intro.asp, 2017.
- [W3C Superseded Recommendation, 2017] W3C Superseded Recommendation, “Introduction to HTML 4”, <http://www.w3.org/TR/html401/intro/intro.html#h-2.2>, 2017.
- [Zhou, 1997] Y.Zhou, *Cindi: The Virtual Library Graphical User Interface*, tesis de maestría, Universidad de Concordia Montreal, Québec, Canada, Abril 1997.