SEP
SECRETARÍA DE EDUCACIÓN PÚBLICA

TECNOLÓGICO NACIONAL DE MÉXICO

Instituto Tecnológico de Ciudad Madero

INSTITUTO TECNOLÓGICO DE CIUDAD MADERO

División de Estudios de Posgrado e Investigación

TRATAMIENTO DE SUBCONSULTAS COMBINADAS CON FUNCIONES DE AGREGACIÓN EN UNA INTERFAZ DE LENGUAJE NATURAL PARA CONSULTA A BASES DE DATOS

Opción I

Tesis Profesional

Que para obtener el grado de:

Maestra en Ciencias de la Computación

Presenta

L.I. Juana Gaspar Hernández

G06071547

Director de Tesis

Dr. Rodolfo Abraham Pazos Rangel

Cd. Madero, Tamaulipas, México Junio 2017

"Año del Centenario de la Promulgación de la Constitución Política de los Estados Unidos Mexicanos"

Cd. Madero, Tamps; a 12 de Mayo de 2017.

OFICIO No.: U5.064/17
AREA: DIVISIÓN DE ESTUDI
DE POSGRADO E INVESTIGACIÓN
ASUNTO: AUTORIZACIÓN DE IMPRESIÓN DE TESIS

ING. JUANA GASPAR HERNÁNDEZ
NO. DE CONTROL G06071547
P R E S E N T E

Me es grato comunicarle que después de la revisión realizada por el Jurado designado para su examen de grado de Maestría en Ciencias de la Computación, el cual está integrado por los siguientes catedráticos:

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

Se acordó autorizar la impresión de su tesis titulada:

"TRATAMIENTO DE SUBCONSULTAS COMBINADAS CON FUNCIONES DE AGREGACIÓN EN UNA INTERFAZ DE LENGUAJE NATURAL PARA CONSULTA A BASES DE DATOS"

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.

A T E N T A M E N T E
"POR MI PATRIA Y POR MI BIEN"®

DRA. ADRIANA ISABEL REYES DE LA TORRE
JEFA DE LA DIVISIÓN

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

AIRT'NACO 'jar

Contenido

Lista de Figuras

Lista de Tablas

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 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 publicaciones.

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

_____

Juana Gaspar Hernández

Agradecimientos

Agradezco al CONACYT por haberme brindado el apoyo necesario para concluir este proyecto de investigación. Asimismo, agradezco al Dr. Rodolfo A. Pazos Rangel por dar seguimiento atento a este proyecto y además haber aportado su paciencia y sabiduría para llevar a buen término este trabajo.

También agradezco a mi familia por haberme apoyado en esta etapa muy importante en mi vida; en especial a mi esposo Alan G. Aguirre Lam y a mi hija Hannah J. Aguirre Gaspar. A mis padres Severiano Gaspar Gaspar y Enedina Hernández Tenorio por el ánimo que me dieron en el transcurso de este proyecto.

Finalmente, agradezco a los miembros del comité tutorial asignado a este proyecto por sus puntuales observaciones para llevar por buen camino este proyecto.

Resumen

Las interfaces de lenguaje natural a bases de datos (ILNBDs) han sido herramientas que facilitan a usuarios inexpertos la obtención de información de bases de datos. A pesar de ser herramientas útiles, el desarrollo de éstas ha sido complejo.

El presente proyecto se implementó en una ILNBD desarrollada en el Instituto Tecnológico de Ciudad Madero. Este proyecto brinda a la ILNBD la capacidad de procesar un mayor número de consultas, al permitir procesar consultas que involucren un mayor número de subconsultas que la versión anterior de la ILNBD.

Para tal fin, se realizó un análisis de los corpus de Geobase y ATIS para identificar las consultas que involucraran el uso de reuniones, funciones de agregación y agrupamiento. Una vez que se identificaron los posibles casos, se diseñaron algoritmos para construir consultas con subconsultas en lugar de reuniones (como en la versión anterior).

Posteriormente, se realizó un análisis de los algoritmos implementados en la versión anterior. Dicho análisis fue usado para modificar el núcleo de la ILNBD de tal manera que ésta pudiera construir consultas que involucren funciones de agregación y agrupamiento, combinados con subconsultas en lugar de reuniones.

La modificación descrita en el párrafo anterior permite a la ILNBD construir consultas con mayor número de subconsultas que el obtenido en la versión anterior de la ILNBD.

Capítulo 1. Introducción

Las primeras interfaces de lenguaje natural para bases de datos (ILNBDs) eran básicamente interfaces hechas a la medida para un dominio en específico, algunas de las más famosas son Baseball [Green, 1961] y Lunar [Woods, 1972]. Dichas ILNBDs funcionaban con una base de datos (BD) en particular.

La mayoría de las ILNBDs fueron construidas teniendo en mente una base de datos particular. Debido a esto no podían ser fácilmente modificadas para ser usadas con diferentes bases de datos o eran difíciles de portar a diferentes dominios de aplicaciones (diferentes gramáticas, conocimiento específico o reglas de mapeo que tuvieron que ser desarrolladas). Las fases de configuración eran tediosas y requerían un largo tiempo. Estos sistemas tenían una base de datos o un sistema de conocimiento creados manualmente por expertos del dominio. Algunas de las primeras ILNDBs se basaban en técnicas de comparación de patrones, pero lo superficial de la técnica frecuentemente ocasionaba errores.

Las ILNBDs desarrolladas posteriormente usaban un lenguaje intermedio de representación, el cual expresa el significado de las consultas planteadas por el usuario en función de conceptos de alto nivel independientemente de la estructura de la base de datos [Androutsopoulos, 1995].

Las ILNBDs son herramientas importantes, ya que permiten a los usuarios acceder a la información en una base de datos mediante una consulta formulada en lenguaje natural (LN). La interfaz interpreta la consulta y la traduce a una instrucción de lenguaje de consulta de BDs (como SQL), la cual es enviada por el ILNBD a un sistema de administración de BDs para obtener la información solicitada. Conviene destacar que sólo usuarios con experiencia pueden formular consultas en este tipo de lenguajes (como SQL).

El propósito de este proyecto es ampliar la funcionalidad de una ILNBD para que ésta pueda contestar consultas que involucren subconsultas combinadas con funciones de agregación.

1.1. Objetivos

Objetivos generales:

OG1. Modificar el proceso de traducción de LN a SQL de la ILNBD desarrollada en el ITCM [Aguirre, 2014], de tal manera que ahora genere expresiones en SQL que incluyan subconsultas en lugar de expresiones que incluyan reuniones (*joins*) en los casos donde sea posible.

OG2. Procesar consultas que involucren funciones de agregación y agrupamiento con más niveles de subconsultas que el obtenido en [Verástegui, 2015].

Objetivos específicos:

OE1. Sustituir la subcapa *Determinación de reuniones implícitas* de la ILNBD (Figura 1.1) por otra que genera expresiones en SQL con subconsultas en lugar de expresiones con reuniones en los casos donde sea posible.

OE2. Analizar las consultas que involucran subconsultas combinadas con funciones de agregación e identificar los problemas de este tipo de consultas.

OE3. Adaptar el código implementado en [Verástegui, 2015] a la modificación de la ILNBD (mencionada en OE1) con el fin de hacer posible el procesamiento de consultas que involucren funciones de agregación y agrupamiento con más niveles de subconsultas que el obtenido previamente.

1.2. Justificación y beneficios

La cantidad de información disponible en las BDs se encuentra en constante crecimiento; debido a esto, surge la necesidad de crear sistemas de consulta que sean eficaces. Las dos alternativas más usadas para que los usuarios accedan a BDs son lenguajes de consulta a BDs (como SQL) o formularios gráficos. Los lenguajes de consulta a BDs permiten consultar BDs con una gran flexibilidad, pero son complicados de usar para usuarios que no son profesionales de la computación; por otra parte, los formularios son fáciles de usar, pero tienen limitaciones en cuanto a flexibilidad para consultar BDs. Las ILNBDs constituyen una alternativa que combina las ventajas de las dos alternativas anteriores: son fáciles de usar y poseen gran flexibilidad para consultar BDs. Por lo tanto, los beneficios de este proyecto se derivan de los beneficios de las ILNBDs.

Particularmente, en lo que respecta a este proyecto, conviene mencionar que la ILNBD sobre la que se va a trabajar en este proyecto [Aguirre, 2014], al traducir una consulta en LN que involucra dos o más tablas, genera una expresión en SQL que incluye reuniones (*joins*).

De acuerdo a la mayoría de la literatura, el desempeño de consultas con reuniones en algunos casos es mejor que el desempeño de las consultas con subconsultas. Sin embargo, es muy difícil saber cuál de los dos tipos de consultas tiene mejor desempeño, esto se debe a que cada SABD (sistema administrador de bases de datos) tiene un plan de ejecución de consultas, el cual especifica cómo debe tratarse cada consulta formulada al SABD, p. ej. Algunos SABD como MySQL antes de ejecutar una consulta, convierten las consultas con subconsultas en consultas con reuniones. Para este proyecto se utilizó MS Access como SABD. Desafortunadamente, no existe documentación que especifique cuál tipo de consulta tiene mejor desempeño (subconsultas o reuniones).

Por lo tanto, un beneficio específico de este proyecto es reducir el consumo de recursos de tiempo y memoria de la ILNBD para consultas que involucren dos o más tablas. Esto se verá reflejado en futuras versiones de la ILNBD que utilicen un SABD diferente al de la versión actual. Adicionalmente, el procesamiento de consultas involucrando subconsultas se combinará

con la funcionalidad desarrollada en [Verástegui, 2015] para hacer posible el procesamiento de consultas que involucren funciones de agregación y agrupamiento con más niveles de subconsultas que el obtenido en [Verástegui, 2015].

1.3. Planteamiento del problema

El problema a resolver puede ser descrito como una caja negra, la cual tiene como finalidad producir una salida dada cierta entrada. De manera más específica, la entrada hace referencia a una consulta formulada en lenguaje natural escrito, la cual puede involucrar funciones de agregación, agrupamiento y subconsultas. En el lado de la salida se debe obtener una consulta traducida a SQL que sea semánticamente equivalente a la consulta especificada en la entrada. A continuación se presentan algunos ejemplos de consultas que pueden ser procesadas por el módulo desarrollado en este proyecto de tesis.

Consulta que involucra varias tablas:

Consulta en LN:

¿A través de qué estados corre el río Delaware?
(What states does the Delaware river run through?)

Traducción a SQL (empleando reuniones):

```
SELECT state.state_name
FROM state, river, riverstate
WHERE river.river_name = "Delaware"
    AND state.abbreviation = riverstate.state_abbreviation
    AND riverstate.river_id = river.river_id
```

Traducción a SQL (empleando subconsultas):

```
SELECT state_name
FROM state
WHERE state.abbreviation IN
    (SELECT riverstate.state_abbreviation
    FROM riverstate
    WHERE riverstate.river_id =
        (SELECT river.river_id
        FROM river
        WHERE river.river_name = "Delaware"))
```

Consulta que involucra subconsultas y funciones de agregación:
Consulta en LN:

¿Cuál es el punto más alto del estado con la mayor área?
(What is the highest point of the state with the largest area?)

Traducción a SQL:

```
SELECT highest_point
FROM highlow
WHERE highest_elevation =
    (SELECT MAX(highest_elevation)
    FROM highlow
    WHERE highlow.state_name =
        (SELECT state.state_name
        FROM state
        WHERE state.area =
            (SELECT MAX(area)
            FROM state)))
```

El presente trabajo añade funcionalidad a una ILNBD [Aguirre, 2014]. Para tal fin, se desarrolló un módulo que permite procesar consultas que involucren más de una subconsulta y funciones de agregación. Además, se usan subconsultas (siempre que esto sea posible) en lugar de reuniones para enlazar las tablas involucradas en la consulta. Para tal efecto, se sustituye el modulo *Determinación de reuniones implícitas* por un módulo llamado *Generación de consultas en SQL* (Figura 1.1).

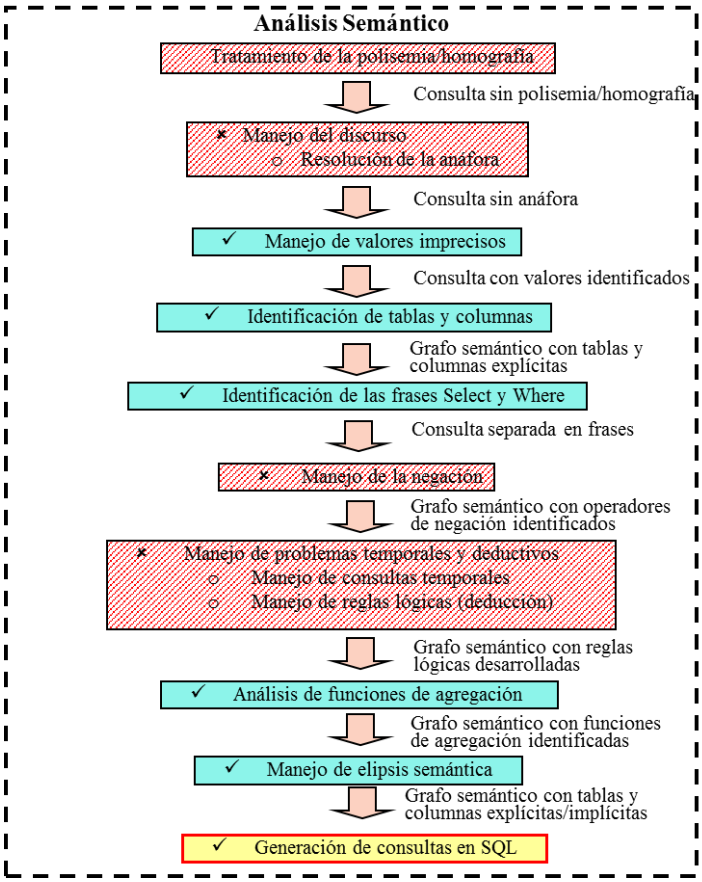

Figura 1.1. Modificación de arquitectura propuesta.

1.3.1. Descripción de la complejidad del problema

A continuación se muestran algunos ejemplos de consultas, las cuales se abordan en este proyecto; además, se muestran los resultados obtenidos con la interfaz comercial ELF para dichas consultas. Esto se realiza con el objetivo de demostrar que algunas ILNBDs son incapaces de responder correctamente las consultas que se contestan con la realización de este proyecto.

Consulta 1 en LN:

¿A través de qué estados corre el río Mississippi?
(Which states does the Mississippi river run through?)

Consulta en SQL con subconsultas:

```
SELECT state.state_name
FROM state
WHERE state.abbreviation IN
    (SELECT riverstate.state_abbreviation
    FROM riverstate
    WHERE riverstate.river_id IN
        (SELECT river.river_id
        FROM river
        WHERE river.river_name = "Mississippi"))
```

Resultado obtenido por ELF:

En la Figura 1.2 se muestra la interfaz ELF respondiendo la consulta *¿A través de qué estados corre el río Mississippi?* Como puede verse ELF no es capaz de responder correctamente la consulta, y retorna una consulta en SQL cuyo resultado no es el esperado. Esto se debe a que ELF busca en la BD el valor de búsqueda *Mississippi river*, encontrándolo en la columna *lowest_point* de la tabla *highlow*. Esto hace que la interfaz construya la consulta usando dicha tabla.

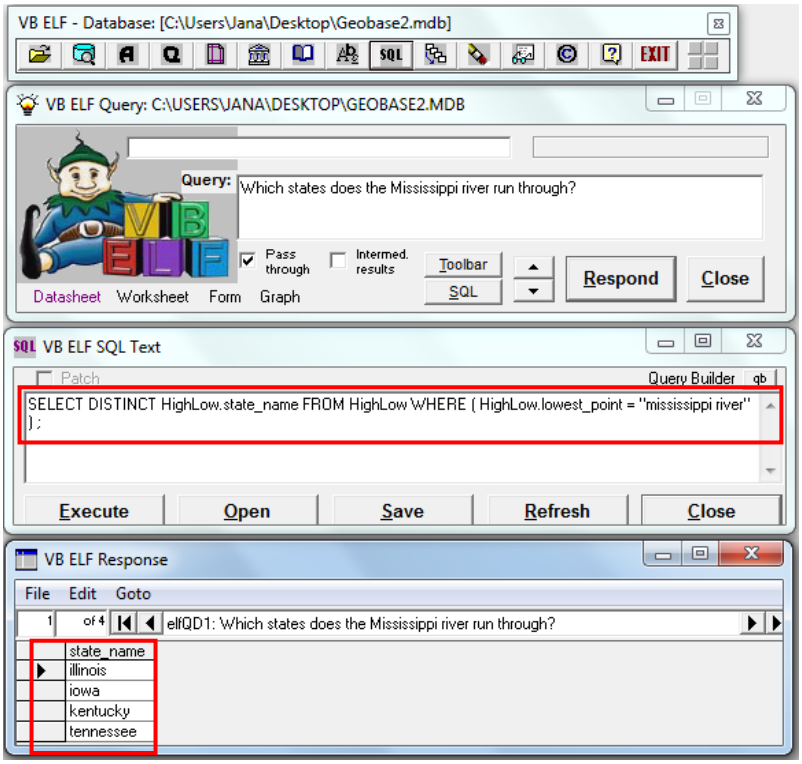

Figura 1.2. Resultado de ELF para la consulta 1.

Consulta 2 en LN:
¿Por cuántos estados corre el río Mississippi?
(How many states does the Mississippi river run through?)

Consulta en SQL con subconsultas:

SELECT COUNT (*state.state_name*)
FROM *state*
WHERE state.abbreviation IN
    (SELECT *riverstate.state_abbreviation*
    FROM *riverstate*
    WHERE *riverstate.river_id* IN
        (SELECT *river.river_id*
        FROM *river*
        WHERE *river.river_name* = "Mississippi"))

Resultado obtenido por ELF:

En la Figura 1.3 se muestra la interfaz ELF respondiendo la consulta *¿Por cuántos estados corre el río Mississippi?* Como puede verse ELF no es capaz de responder correctamente la consulta, y retorna una consulta en SQL cuyo resultado no es el esperado. Esto se debe a que ELF busca en la BD el valor de búsqueda *Mississippi river*, encontrándolo en la columna *lowest_point* de la tabla *highlow*. Posteriormente, ELF aplica la función de agregación COUNT

sobre la consulta ya mencionada, por lo que la consulta compuesta no está relacionada con la tabla *river*, que es la tabla de interés.

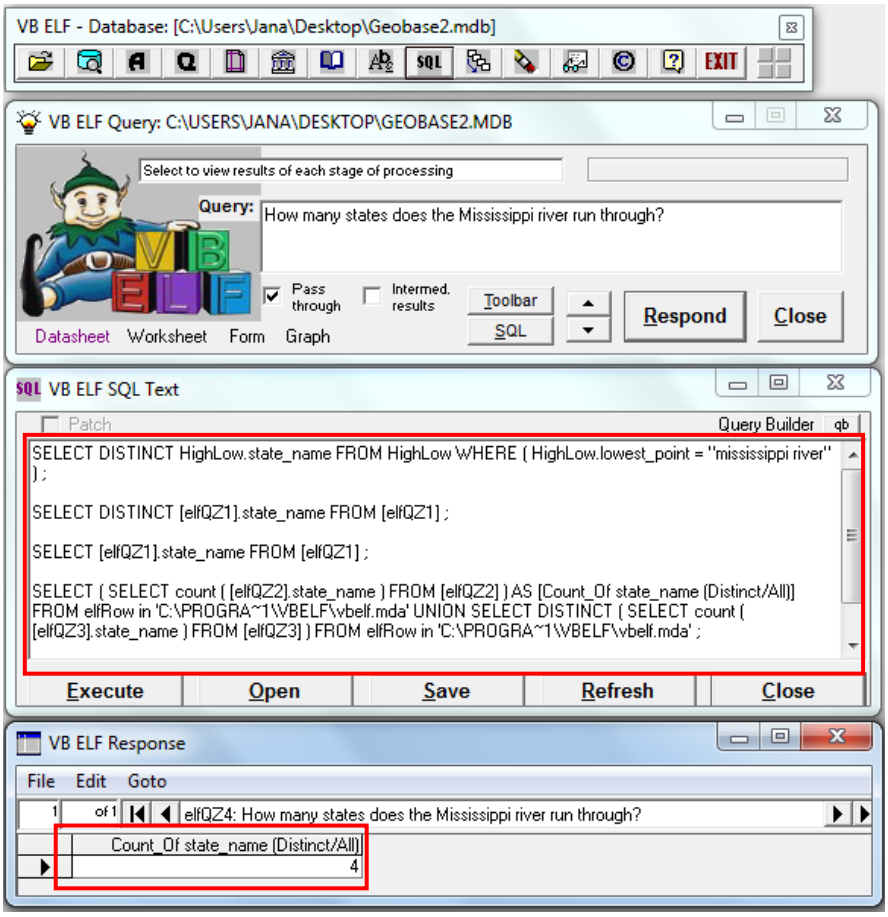

Figura 1.3. Resultado de ELF para la consulta 2.

Consulta con varias subconsultas
Consulta 3 en LN:

¿Cuál es la capital del estado con el punto más alto?
(What is the capital of the state with the highest point?)

Consulta en SQL:

SELECT *state.state_name, state.capital*
FROM *state*
WHERE *state.abbreviation* IN
    (SELECT *highlow.state_abbreviation*
    FROM *highLow*
    WHERE *highlow.highest_elevation* =
        (SELECT MAX (h*ighlow.highest_elevation*)
        FROM *highlow*))

Resultado obtenido por ELF:

En la Figura 1.4 se muestra la interfaz ELF respondiendo la consulta *¿Cuál es la capital del estado con el punto más alto?* Como puede verse ELF no es capaz de responder correctamente la consulta, y retorna una consulta en SQL cuyo resultado no es el esperado. Esto se debe a que ELF no es capaz de detectar que *highest point* es un valor de búsqueda que requiere aplicar una función de agregación (MAX) con subconsulta.

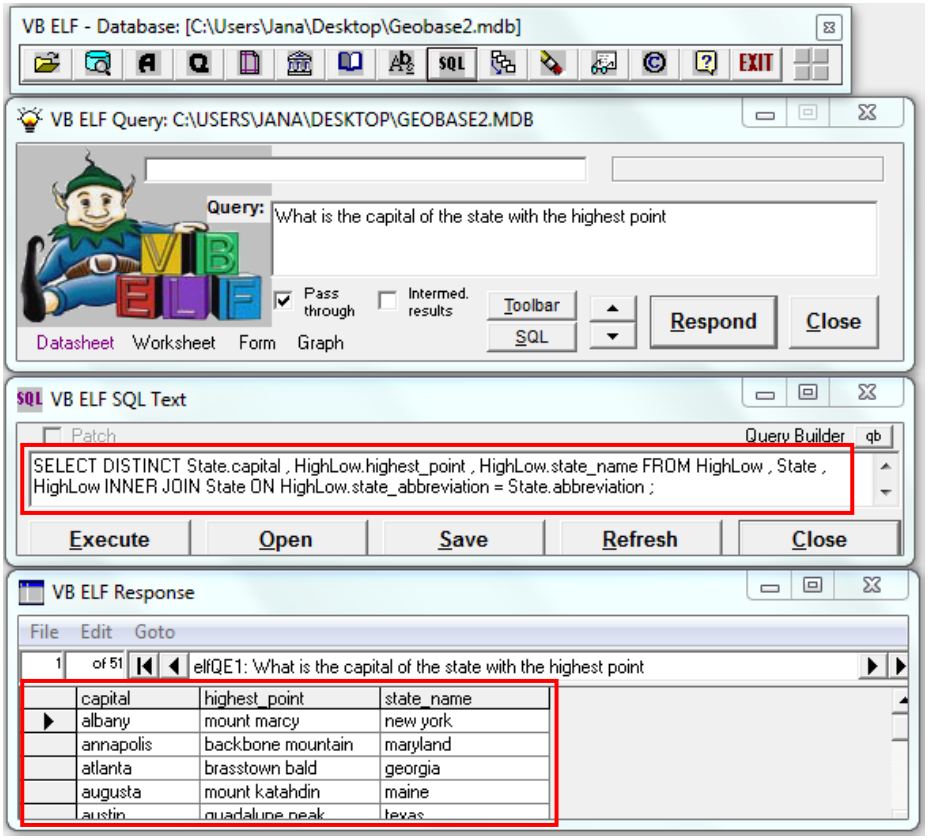

Figura 1.4. Resultado de ELF para la consulta 3

Con relación al código programado para la implementación de la ILNBD [Aguirre, 2014], para tener una idea de la complejidad de los procesos implementados en los módulos principales, es conveniente mencionar que el traductor de LN a SQL tiene aproximadamente 8,000 líneas de código, el editor para configurar el DIS tiene aproximadamente 7,500 líneas de código, y finalmente, el Wizard tiene aproximadamente 4,800 líneas de código. En total la ILNBD consta de aproximadamente 20,000 líneas de código.

El presente proyecto se centra en la tercera capa de funcionalidad de la ILNBD, llamada Análisis Semántico, la cual consta aproximadamente con 3,000 líneas de código. El principal obstáculo para implementar la sustitución de reuniones implícitas por subconsultas, implica desarrollar un algoritmo con la capacidad de distinguir si una consulta que involucra varias tablas puede ser expresada por medio de subconsultas. De ser posible expresar dicha consulta con subconsultas, se requerirá de otro algoritmo para construir las subconsultas a partir del esquema

de la BD, cuya dificultad sería equivalente a desarrollar un nuevo módulo de determinación de reuniones implícitas.

Además, para la incorporación de los algoritmos desarrollados en [Verástegui, 2015] es necesario un análisis a profundidad y rediseño para adaptar dichos algoritmos al código correspondiente de la arquitectura modificada de la ILNBD, así como agregar la funcionalidad para incluir varios niveles de subconsultas y varias tablas en una consulta.

Los algoritmos para tratamiento de FA y agrupamiento se muestran en el Apéndice A. Algunos de estos algoritmos fueron modificados en este proyecto con el fin de agregarles mayor funcionalidad: método para detección de funciones de agregación y agrupamiento, asociación de columna para Group by, método para recopilar elementos de la subconsulta y asociación de cláusula Where.

1.4. Alcance y limitaciones

1.4.1. Alcance

De acuerdo a los objetivos mencionados anteriormente, el alcance de este proyecto se detalla en los siguientes puntos:

- La sustitución del módulo *Determinación de reuniones implícitas* debe permitir a la ILNBD [Aguirre, 2014] la construcción de consultas en SQL usando subconsultas en lugar de reuniones.

- La modificación de los algoritmos desarrollados en [Verástegui, 2015] debe permitir procesar consultas con funciones de agregación y agrupamiento cuyos niveles de subconsultas sean mayores a los obtenidos anteriormente.

- La tipificación de consultas que involucren subconsultas combinadas con funciones de agregación debe permitir resolver problemas de traducción en consultas más complejas.

1.4.2. Limitaciones

- Al igual que las versiones previas de la ILNBD [Aguirre, 2014], esta versión funciona únicamente para el lenguaje español.

- Sólo se maneja un subconjunto de las consultas de SQL de la versión ISO/IEC 9075:1989(E) (SQL 1). Entre las funciones que no están implementadas se encuentran: Order by, Desc, Asc y todas las funciones que se relacionen con operaciones de inserción, borrado o actualización de información.

- No se trata con consultas que involucren información que no esté explícita en la BD, es decir, consultas para BDs deductivas.

- No se considera el problema de transformar una consulta a su equivalente optimizada.
- Las consultas introducidas en LN a la ILNBD deben ser léxica y sintácticamente correctas.
- No se resuelven consultas que contengan los siguientes problemas: deducción, anáfora, negación y otros.
- Las pruebas realizadas son únicamente funcionales. Se realizan pruebas para las bases de datos de ATIS y Geobase.

Capítulo 2. Marco teórico y trabajos relacionados

2.1. Marco teórico

A continuación se explicaran conceptos que servirán de base para entender los procesos detallados en desarrollo de este proceso de tesis.

2.1.1. Procesamiento del lenguaje natural

El procesamiento de lenguaje natural (PLN o NLP en inglés) es un conjunto de técnicas computacionales para analizar y representar naturalmente textos en uno o más niveles de análisis lingüístico, con el fin de llevar a cabo el procesamiento del lenguaje como un humano para un rango de tareas y aplicaciones [Liddy, 2001].

2.1.2. Lenguaje

Conjunto de señales que dan a entender algo [RAE, 2017]. Cuando se habla de lenguajes se pueden diferenciar dos clases muy bien definidas:

- Los lenguajes naturales como el español, inglés, francés, etc.
- Los lenguajes formales como los lenguajes de programación, el lenguaje de la lógica matemática, etc.

2.1.3. Lenguaje formal

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 [Mellema, 2009].

2.1.4. Lenguaje natural

Lenguaje hablado o escrito por humanos, opuesto a un lenguaje de programación utilizado para programar o comunicarse con computadoras. Existen dos campos en el estudio del entendimiento del lenguaje natural [Andrade, 1997]:

- Entendimiento del lenguaje escrito, que utiliza el conocimiento léxico, sintáctico y semántico del lenguaje, unido a la información o conocimiento del dominio.

- Entendimiento del lenguaje oral, que comprende todo lo del campo anterior junto con toda la fonología.

2.1.5. Sistema administrador de bases de datos

Un sistema administrador de bases de datos (SABD, en inglés, *database management system: DMBS*) es un conjunto de programas que permite a los usuarios crear y mantener una base de datos. Por tanto, el SABD es un sistema de software de propósito general que facilita el proceso de definir, construir y manipular bases de datos para diversas aplicaciones [Kroenke, 2003].

2.1.6. Lenguaje de manipulación de datos

El lenguaje de manipulación de datos permite crear, modificar, recuperar y eliminar registros (entidades en un modelo lógico) de una base de datos [Fernández, 2006]. Dicho lenguaje permite las siguientes operaciones:

- Añadir, eliminar y modificar tablas.
- Visualizar el contenido de las tablas.
- Bloquear tablas.

Las instrucciones de este lenguaje son: INSERT, UPDATE, DELETE, SELECT, EXPLAIN, PLAN, LOCK y MERGE.

2.1.7. Funciones de agregación

Llamadas funciones de agregación o funciones de columna, aumentan la capacidad de obtener información. Las funciones de agregación básicas son: COUNT, SUM, AVG, MAX y MIN. Cada una de estas funciones opera en la colección de valores escalares en una columna de alguna tabla [Date, 1989]. A continuación se muestran los resultados de dichas funciones acompañados de un breve ejemplo en SQL. Los ejemplos están basados en la BD Geobase (véase Apéndice C):

COUNT: número de valores en la columna.
Por ejemplo:
SELECT COUNT (*city.city_name*)
FROM *city*
WHERE *city.state_name* = "Alabama"

SUM: suma de los valores en la columna.
Por ejemplo:
SELECT SUM (*city.population*)
FROM *city*
WHER*E city.state_name* = "Alabama"

AVG: promedio de los valores en la columna.
Por ejemplo:
SELECT AVG (*city.population*)
FROM *city*
WHERE *city.state_name* = "Alabama"
MAX: el mayor valor de la columna.
Por ejemplo:
SELECT MAX (c*ity.population*)
FROM *city*
WHERE *city.state_name* = "Alabama"

MIN: el menor valor de la columna.
Por ejemplo:
SELECT MIN (*city.population*)
FROM *city*
WHERE *city.state_name* = "Alabama"

2.1.8. Cláusula Group by

La cláusula Group by reacomoda la tabla representada por la cláusula From en grupos; de tal manera que cada grupo contiene el mismo valor de la columna a la que se aplica Group by. Se debe tomar en cuenta que Group by no ordena los valores, esto se debe realizar con la cláusula Order by [Date, 1989].

2.1.9. Subconsultas

Una subconsulta es una instrucción Select que aparece dentro de otra instrucción Select. Normalmente se utilizan para filtrar una cláusula Where o Having con el conjunto de resultados de la subconsulta [Date, 2008].

Ejemplos:
Dame la ciudad con mayor población del estado de Alabama.

```
SELECT city.city_name, city.population
FROM city
WHERE city.state_name = "Alabama"
   AND city.population =
  (SELECT MAX (city.population)
  FROM city
  WHERE city.state_name = "Alabama")
```

Dame el estado con la mayor área.

```
SELECT state.state_name
FROM state
```

WHERE *state.area* =
SELECT MAX (*state.area*)
FROM *state*

2.2. Trabajos relacionados

2.2.1. MASQUE/SQL

Fue desarrollada por la Universidad de Edimburgo. MASQUE (Modular Answering System for Queries in English) en un principio contestaba consultas elaboradas en Prolog que se ejecutaban en su BD [Androustsopoulos, 1993]. Posteriormente se desarrolló una versión para ser utilizada en SQL y cualquier base de datos relacional que lo soporte. Su principal característica es su portabilidad de dominio, la cual se consigue mediante un editor integrado. Esta nueva versión incluía la capacidad de manejar consultas con funciones de agregación y agrupamiento. Se presenta la arquitectura de MASQUE/SQL en la Figura 2.1

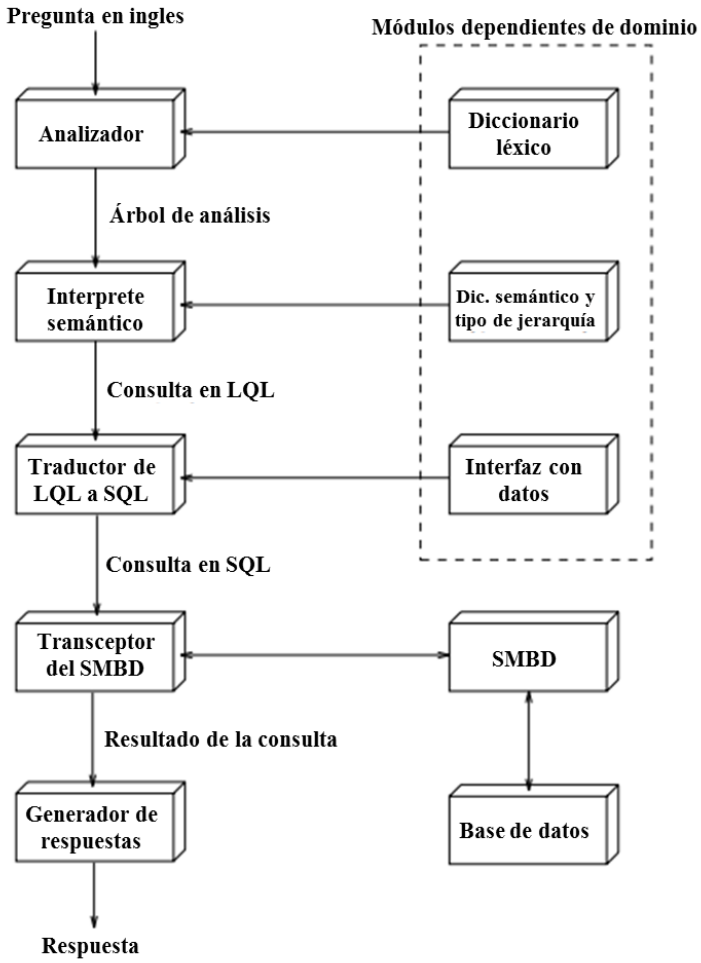

Figura 2.1. Arquitectura de MASQUE/SQL

A continuación se presenta una consulta que involucra funciones de agregación y agrupamiento para la interfaz MASQUE.

What is the average area of the countries in each continent?

Consulta en SQL:
SELECT DISTINCT *rel1.arg1*, AVG(*rel2.arg1*)
FROM *continent#1 rel1, area#2 rel2, country#1 rel3, in#2 rel4*
WHERE *rel3.arg1 = rel2.arg2*
AND *rel4.arg1 = rel2.arg2*
AND *rel4.arg2 = rel1.arg1*
GROUP BY *rel1.arg1*

2.2.2. OWDA

Interfaz desarrollada por la Universidad de Manchester. Esta interfaz maneja un enfoque diferente al utilizar agentes inteligentes y árboles de conocimiento [Owda, 2007]. Este enfoque permite una traducción más precisa al utilizar un diálogo con el usuario mediante los agentes inteligentes. De esta manera se va detallando desde lo más general hasta la consulta concreta.

La utilización de árboles de conocimiento facilita la estructuración de la información, ya que se asemeja a la forma de representar conocimiento por parte de los humanos. Los árboles de conocimiento también facilitan al administrador el mantenimiento y revisión de la información, lo que ofrece una mayor ventaja en su administración, ya que no requiere de conocimiento de ningún lenguaje de programación.

La parte medular de esta interfaz es el administrador de conversación, el cual se encarga de administrar la interacción entre los diversos componentes: los árboles de conocimiento, agentes inteligentes y la base de datos.

Se presenta un ejemplo de la interfaz OWDA, donde se observa la utilización de funciones de agregación y agrupamiento en la Figura 2.2. Esta figura muestra un ejemplo que involucra diálogo, donde CB-NLI-RDB es una abreviatura para denotar a la interfaz, y USR es una abreviatura de usuario.

1	CB-NLI-RDB	Hi I am here to help you explore our company database. What do you want to know? Use the options if that accommodate you! - sales - profitability - products - customers
2	USR	Aha, I want to know about sales.
3	CB-NLI-RDB	What do you want to know about sales? - sales for products - products families - product classes
4	USR	I want sales for all products in the current month.
5	CB-NLI-RDB	**SQL Generated**: SELECT A.ITEM_DSC as "Item description" , sum(C.UNITS) AS "Number of items" FROM PROTOTYPE.PRODUCT_DIM A , PROTOTYPE.UNITS_UPDATE_FACT C WHERE(C.ITEM_ID = A.ITEM_ID) group by A.ITEM_DSC **The result set:** Item description Number of items 1.44MB External 3.5 Diskette 964 Deluxe Mouse 5797 Envoy Ambassador 179

Figura 2.2. Ejemplos de un diálogo de la interfaz de OWDA.

2.2.3. C-Phrase

Esta interfaz se enfoca en técnicas de procesamiento de lenguaje natural [Minock, 2010]. Para tal efecto utiliza cuatro componentes: dominio léxico, lenguaje de procesamiento de lenguaje natural, definición de equivalencias y definición de relaciones funcionales para definir un dominio específico. Como medio de comprobación utiliza diálogos de aclaración. El enfoque de esta interfaz se basa en el uso de un lenguaje de razonamiento.

En su proceso de traducción modela las consultas en una versión del cálculo de tuplas de Codd de alto orden. Esto debido a la dificultad de trabajar directamente con expresiones en SQL. Para complementar utiliza gramáticas extendidas sincrónicas con funciones lambda para representar las gramáticas semánticas. Estas gramáticas semánticas son modeladas como gramáticas sincrónicas independientes de contexto, incrementadas con las expresiones de cálculo lambda (λ-SCFG).

La utilización de las reglas SCGF define dos árboles sincrónicos. El producto del primer árbol es lenguaje natural y el producto del segundo es lenguaje formal que expresa las semánticas del primer árbol. Como se puede apreciar la complejidad de este diseño complica la identificación de que un producto es para el segundo árbol, ya que utiliza variables en las fórmulas semánticas.

Para el procesamiento de consultas que involucran funciones de agregación, C-Phrase hace uso de cláusulas y operaciones que sustituyen la función de agregación. Con este

procedimiento C-Phrase es capaz de obtener resultados similares a los obtenidos con el uso de las funciones de agregación. Sin embargo, la consulta traducida en SQL es más compleja y conlleva un mayor uso de cláusulas y operaciones detalladas. A continuación se presenta un ejemplo en el cual se puede observar el uso de la función de agregación MAX y su traducción a SQL utilizando el método de C-Phrase.

Consulta en LN:

Cities of over 100,000 people in the largest area mid-western state.
(Ciudades de más de 100,000 personas en el estado más grande de la zona del medio oeste.)

Consulta en SQL:

```
SELECT *
FROM city AS x
WHERE x.population > 100000
AND EXISTS
   (SELECT *
   FROM
      (SELECT *
      FROM state AS z
      WHERE z.name = 'Indiana'
      OR ...
      OR z.name = 'Wisconsin'
      ORDER BY area DESC LIMIT 1)
      AS y
      WHERE x.state = y.name)
```

2.2.4. ELF

Es una de las interfaces comerciales para traducir lenguaje natural a una consulta estructurada en cualquier base de datos de Access [Conlon, 2004]. Es la interfaz que mejor se desempeña en el ámbito comercial. Cuenta con tres características principales:

- Independencia de dominio: una vez que ELF es instalado en una computadora, es capaz de funcionar con cualquier base de datos [ELF, 2009].
- Estructura de la BD: ELF es capaz de obtener automáticamente la estructura de la BD; por lo tanto, su configuración inicial es muy rápida y sencilla.
- Semántica de la BD: durante el proceso de análisis ELF examina los términos utilizados para definir las columnas y tablas, y utiliza su diccionario integrado para tratar de predecir los sinónimos utilizados en las consultas. En caso de que ELF no pueda asignar un término, existe una ventana de búsqueda en la cual el usuario ingresa el término y selecciona la columna correcta, a partir de los datos asociados. Por ejemplo: al ingresar López, se muestra

la columna TRABAJADOR_APELLIDOS donde TRABAJADOR es el nombre de la tabla y APELLIDOS el nombre de la columna donde se encuentra el valor [ELF, 2009].

2.2.5. SNL2SQL

La interfaz de lenguaje natural español a SQL es una iniciativa por parte del Gobierno Federal Mexicano [Esquivel, 2013], la cual tiene por objeto la generación automática de informes, mediante un sistema de traducción de peticiones hechas en lenguaje natural a instrucciones en SQL. El sistema en la actualidad ha logrado resultados en los siguientes aspectos:

- Manejo de expresiones temporales (fechas).
- Traducción de términos de agrupamiento.
- Tratamiento de expresiones estadísticas.

Debido a la complejidad inherente, aún no se han creado módulos para el filtrado de grupos, manejo de subpreguntas y procesamiento de expresiones horarias.

Como parte del desarrollo de la interfaz, se descubrió que es necesario traducir primero la consulta en SQL a lenguaje natural para verificar si el sistema "ha entendido" la petición correctamente.

El módulo cuenta con dos diccionarios: el principal llamado *de dominio* y el secundario llamado *de contenido*. El principal se construyó a partir de un diccionario de sinónimos y de metadatos de la base de datos.

Los metadatos se extraen de cada tabla, los cuales consisten de nombre, descripción de la misma y los datos relativos a la columna (nombre, tipo de dato, tamaño, etc.).

Utilizando este procesamiento se pueden obtener los sustantivos asociados con las columnas y tablas que en su descripción los contengan a ellos o sus sinónimos.

Para el manejo de funciones de agregación y agrupamiento, se crearon diccionarios de expresiones homólogas para enriquecer el contenido de las oraciones. El módulo en cada conversión elige aleatoriamente una expresión distinta del correspondiente diccionario.

Se diseñaron una serie de pasos para lograr la traducción:

- Se determinan las cláusulas y subcláusulas presentes mediante un arreglo de términos base.
- Se detectan las variables para su búsqueda en el diccionario de dominio.
- Al encontrar los términos correspondientes a datos, comúnmente asociados a la cláusula Where o Having, se verifica que existan en el diccionario de contenido.

- Ya convertidos todos los componentes, se integra la oración completa para su presentación.

Este procedimiento permite traducir una consulta de lenguaje natural a SQL. Posteriormente traduce el comando SQL obtenido a una pregunta en lenguaje natural antes de obtener la información para así verificar la exactitud de la traducción, como se muestra a continuación.

SELECT MAX (*salary*)
WHERE *job* = 'designer'

Se traduce a:
Muestra máximo de salario donde puesto sea igual a diseñador

A continuación se presenta una tabla de las interfaces mencionadas y sus principales características.

Tabla 2.1. ILNBDs y características.

Interfaz	Funciones de agregación	Consultas con varias tablas y funciones de agregación	Consultas con varias subconsultas	Independencia de dominio	Año
MASQUE	✓	No reportado	No reportado	✓	1992
OWDA	✓	No reportado	No reportado	✓	2007
C-PHRASE	✓	Parcialmente	Parcialmente	✓	2008
ELF	✓	Parcialmente	Parcialmente	✓	2010
SNL2SQL	Uso limitado	No reportado	No reportado	No reportado	2010

Como se puede observar en la Tabla 2.1, la mayoría de las interfaces mencionadas en esta sección no reportan si su enfoque permite responder consultas que involucren consultas con varias tablas y funciones de agregación y consultas con varias subconsultas. Esto se debe a que los enfoques empleados en dichas interfaces se centran en otros aspectos más generales del procesamiento de LN, dejando de lado aspectos más específicos tales como los mencionados.

2.3. Antecedentes

2.3.1. Resumen de la tesis de Marco Aguirre

En 2014 Aguirre desarrolló una ILNBD con una arquitectura basada en capas (Figura 2.3) y un diccionario de información semántica (DIS) basado en un modelo de BDs semánticamente enriquecido [Aguirre, 2014].

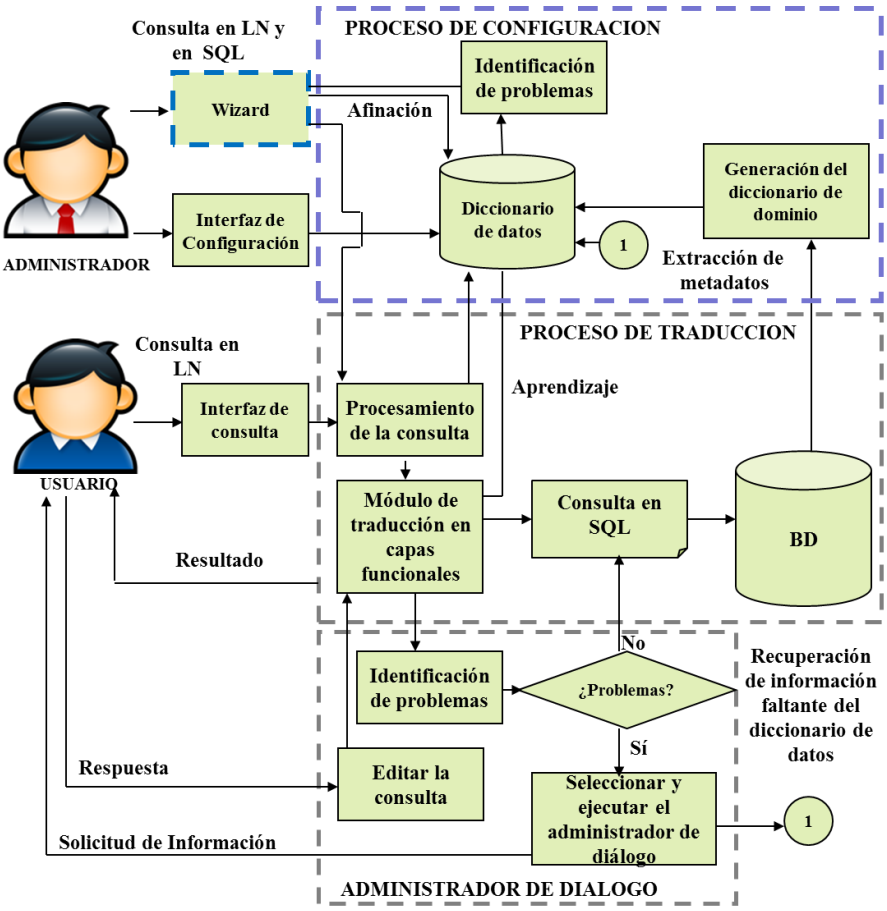

Figura 2.3. Arquitectura general de la ILNBD.

El núcleo de la interfaz propuesta por Aguirre y desarrollada en el ITCM consta de tres capas principales (Figura 2.4). La primera capa se denomina Análisis Léxico y tiene como finalidad etiquetar todas las palabras que se encuentren en el lexicón con su categoría gramatical. En caso de que alguna palabra no sea localizada dentro del lexicón, se supone que es un valor de búsqueda. La siguiente capa llamada Análisis Sintáctico está compuesta por una heurística, la cual realiza un análisis para obtener una sola categoría gramatical por palabra, ya que existen palabras que tienen más de una categoría gramatical, también se ignoran palabras irrelevantes. La tercera capa, denominada Análisis Semántico, se encarga de identificar tablas y columnas en base a la información almacenada en el DIS.

Dentro del Análisis Semántico existen subcapas que realizan procesos necesarios para poder traducir la consulta. Una de esas subcapas consiste en la identificación de las frases Select y Where. Una vez que se tienen identificadas las tablas, columnas y valores de búsqueda, la consulta en LN se divide en frases Select y Where. Para tal efecto, cada valor de búsqueda es asociado a una columna de acuerdo con su cercanía y tipo de dato. Los pares columna-valor de búsqueda constituyen la frase Where y el resto de las columnas constituyen la frase Select. Por último, se determinan reuniones (*joins*) implícitas, ya que para construir la consulta en SQL es necesario que el grafo (constituido por las tablas) y las condiciones de reunión (condiciones de búsqueda constituidas por una columna de una tabla y otra columna de otra tabla) sea un grafo conectado. Para realizar esta acción se utiliza una heurística para determinar el camino más corto

entre dos tablas. Esta ILNBD logra obtener un porcentaje de acierto de 90% con la base de datos ATIS.

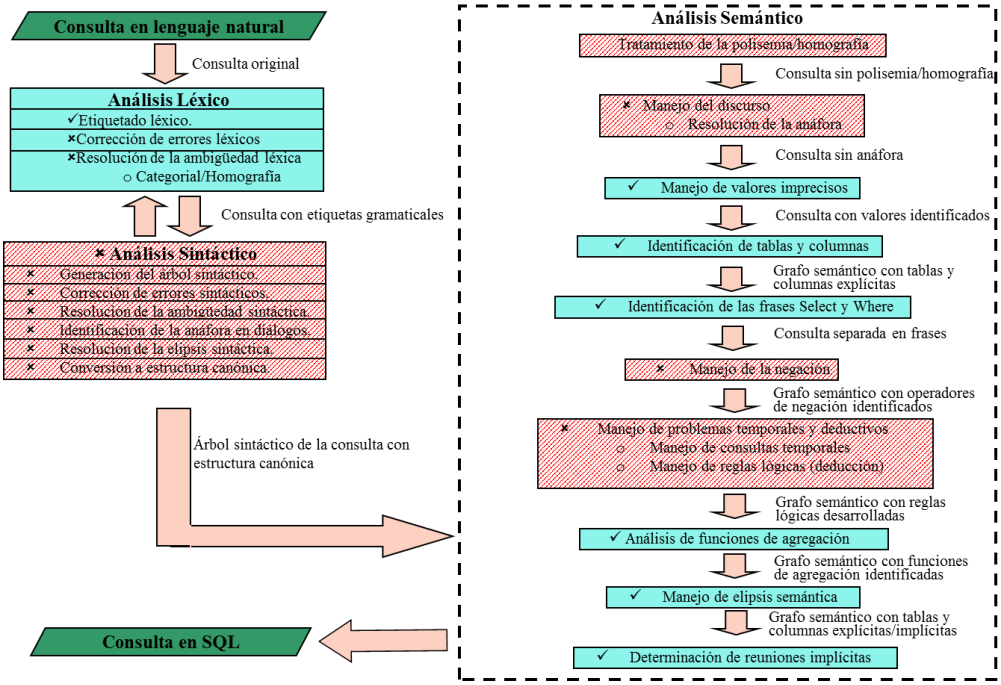

Figura 2.4. Capas de funcionalidad de la ILNBD.

2.3.2. Resumen de la tesis de Andrés Verástegui

En 2015 Verástegui realizó una mejora a la interfaz, añadiendo un módulo para el tratamiento de consultas que involucran funciones de agregación, agrupamiento y subconsultas.

Este problema puede ser descrito como una caja negra, la cual tiene como finalidad producir una salida dada cierta entrada. De manera más específica, la entrada hace referencia a una consulta formulada en lenguaje natural escrito, la cual puede involucrar funciones de agregación, agrupamiento y subconsultas. En el lado de la salida se debe obtener una consulta traducida a SQL que sea semánticamente equivalente a la consulta especificada en la entrada. A continuación se presentan algunos ejemplos de consultas que pueden ser procesadas correctamente por el módulo ya mencionado:

Consulta con función de agregación:
¿Cuál es el área del estado más grande?
SELECT MAX (*area*)
FROM *state*

Consulta con función de agrupamiento y agregación:
Dame el área total de los lagos por estado
SELECT SUM (*area*)
FROM *lake*

GROUP BY (*state*)

Consulta con funciones de agregación y subconsulta:
Dame el nombre del estado con la mayor área
SELECT *state_name*
FROM *state*
WHERE *area* = (SELECT MAX *(area)* FROM *state*)

Dicho módulo se ubica en la tercera capa de funcionalidad (Análisis Semántico), específicamente después de la identificación de las cláusulas Select y Where (Figura 2.5). Su posición dentro del procesamiento de la consulta es esencial, ya que el módulo utiliza la información recabada por la interfaz para poder realizar el tratamiento de funciones de agregación, agrupamiento y subconsultas (ver algoritmos del Apéndice A). Para el tratamiento de funciones de agregación y la cláusula Group by se creó una tabla dentro del DIS, la cual contiene palabras que hacen referencia a funciones de agregación y agrupamiento y su respectiva equivalencia en SQL.

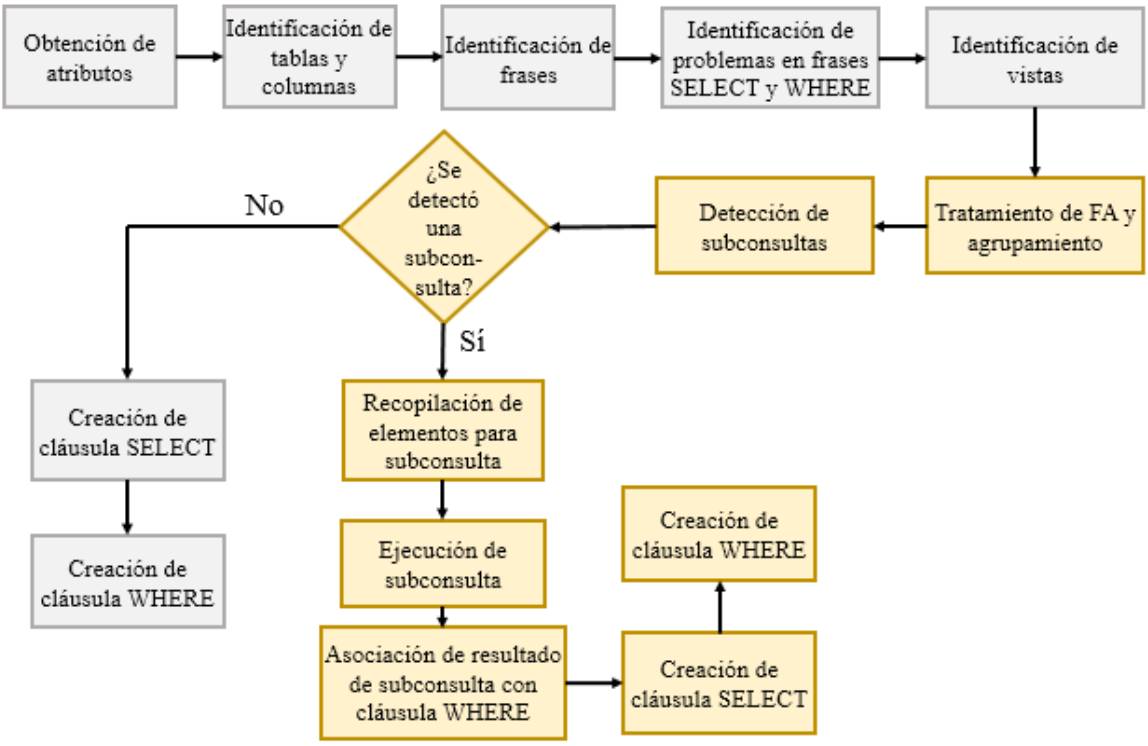

Figura 2.5. Procesos del análisis semántico.

Capítulo 3. Metodología de solución

Uno de los objetivos de este proyecto es cambiar las reuniones implícitas por subconsultas. Sin embargo, en algunos casos donde la cláusula Select involucra columnas de dos o más tablas diferentes, no se puede construir la consulta con subconsultas y forzosamente tendrá que construirse con reuniones. Para demostrar lo anterior se presenta el siguiente ejemplo:

Consulta en LN:
Costo de vuelo del vuelo número 144165.

Consulta en SQL:
SELECT *fare.one_way_cost, fare.rnd_trip_cost, flight.flight_number*
FROM *flight, fare, flight_fare*
WHERE *flight.flight_code = flight_fare.flight_code*
AND *flight_fare.fare_code = fare.fare_code*
AND *flight.flight_number* = 144165

Como se puede observar en el ejemplo anterior, la consulta en SQL tiene una cláusula Select que involucra columnas de dos tablas diferentes (*fare* y *flight*). Por lo tanto, no se puede representar con subconsultas, ya que al realizar las reuniones entre tablas usando subconsultas, las subconsultas anidadas referentes a cada reunión sólo pueden realizarse sobre una tabla a la vez.

Debido a lo anterior, en esos casos se utilizaron reuniones en lugar de subconsultas. En caso de que la consulta involucre únicamente una tabla en la cláusula Select, se procedió a modificar el proceso de traducción a fin de generar expresiones de SQL con subconsultas.

A continuación se presenta un ejemplo de una consulta que se puede construir con subconsultas.

Consulta en LN:
Me gustaría la lista de precios del vuelo número 3.

Consulta en SQL con reuniones:
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* = 3

Consulta en SQL con subconsultas:
SELECT *fare.one_way_cost*
FROM *fare*
WHERE *fare.fare_code* IN

```
(SELECT flight_fare.fare_code
  FROM flight_fare
  WHERE flight_fare.flight_code IN
     (SELECT flight.flight_code
       FROM flight
       WHERE flight.flight_number = 3))
```

Como puede observarse en el ejemplo anterior, la consulta puede construirse con subconsultas, ya que ésta involucra únicamente una tabla (*fare*) en la cláusula Select.

Para esta consulta, las tablas involucradas son *flight* y *fare*. Para determinar las reuniones implícitas entre estas dos tablas, la interfaz utiliza una estructura llamado grafo semántico, el cual representa las tablas y conexiones entre ellas.

En el grafo semántico (Figura 3.1) se localizan las tablas involucradas en la consulta y se emplea una heurística para determinar la ruta más corta entre las tablas *flight* y *fare*, dando como resultado una ruta. En este caso la ruta indica el conjunto de tablas y sus reuniones involucradas en la consulta.

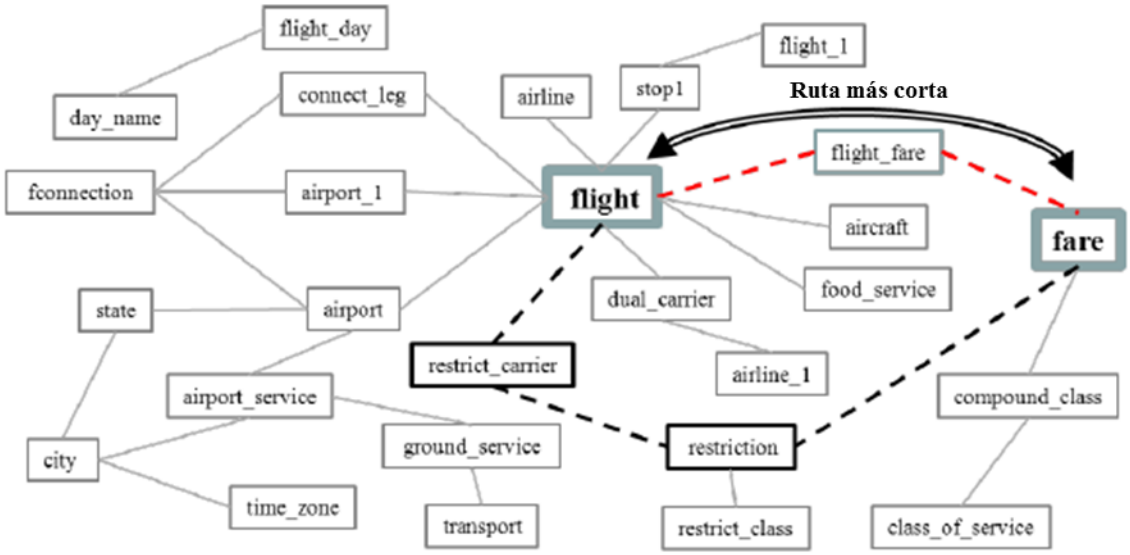

Figura 3.1. Ejemplo de determinación de reuniones implícitas.

Como ya se mencionó en la Sección 1.1, este proyecto debe proporcionar a la interfaz la capacidad de construir las consultas que involucren varias tablas (como la ya mencionada) con subconsultas en lugar de reuniones. Para tal efecto, se puede tomar el grafo conectado que representa las reuniones y construir una consulta que realice una subconsulta anidada por cada arista en el grafo.

Con el presente proyecto se agrega funcionalidad a la ILNBD [Aguirre, 2014] mejorando el módulo descrito en [Verástegui, 2015]. Para tal efecto, se modificaron los algoritmos relacionados con el tratamiento de funciones de agregación de tal manera que dichos algoritmos

permitan procesar consultas en LN tales que involucren más de una tabla y varias funciones de agregación.

3.1. Resumen del trabajo previo sobre FAs, agrupamiento y subconsultas

El módulo para el tratamiento de funciones de agregación, agrupamiento y subconsultas [Verástegui, 2015] realiza modificaciones en la subcapa de identificación de las frases Select y Where y en la estructura utilizada para contener la información de la traducción de la consulta. El módulo para funciones de agregación, agrupamiento y subconsultas funciona analizando la estructura que contiene los componentes léxicos de la consulta en lenguaje natural.

La clase que efectúa el análisis semántico (*analisisSemantico*) contiene los métodos utilizados para realizar dicho análisis. Dentro de la clase análisis semántico se llaman a los métodos del módulo para el tratamiento de funciones de agregación, agrupamiento y subconsultas. Los métodos de dicho módulo forman parte de una sola clase. Los métodos mencionados se listan a continuación:

- El primer paso del análisis semántico es la obtención de atributos (*obtenerAtributos*), los cuales se utilizan para ignorar palabras irrelevantes como las conjunciones.

- El siguiente método realiza la identificación de tablas y columnas (*identificacionTablasyColumnas*), mediante la búsqueda sistemática en el diccionario de información semántica (DIS).

- Una vez detectadas tablas y columnas se realiza la identificación de frases Select y Where (*identificacionFrases*); su etiquetado se realiza en base a las columnas detectadas en el método anterior.

- El tercer método identifica problemas en las frases Select y Where (llamado *identificaProblemasFraseSelect_Where*), y realiza como su nombre lo indica la identificación de problemas en las columnas etiquetadas como frases Select o Where.

- La identificación de vistas (*identificacionVistas*) permite inferir relaciones implícitas entre tablas de manera que se puedan establecer tablas o columnas intermedias para realizar la conexión entre dos tablas.

- El siguiente método efectúa el tratamiento de funciones de agregación y agrupamiento (*agrega_agrupa_subcons*). Este método es el encargado de detectar, clasificar y marcar las columnas y palabras que se refieren a funciones de agregación y agrupamiento de forma que la interfaz pueda detectarlas y utilizarlas en la construcción de la instrucción en SQL.

Una vez detectadas las funciones de agregación y agrupamiento, el método de detección de subconsultas (*validacion_patron*) determina si la consulta cumple con el patrón de una consulta que involucra subconsultas. En caso afirmativo el método para crear la cláusula Select

(*creaClausulaSelect*) y el método para crear la cláusula Where (*creaClausulaWhere*) inician la construcción del comando en SQL para la subconsulta.

Una vez recopilados los elementos se ejecuta la subconsulta, la cual arroja un valor como resultado. Dicho valor se utiliza en el método de asociación en cláusula Where (*asociacion_clausula_where*), el cual, como su nombre lo indica, asocia el valor obtenido con la cláusula Where que se integra en la consulta con el fin de ser usado posteriormente. Una vez que se ejecuta dicho método, los métodos para crear las cláusulas Select (*creaClausulaSelect*) y Where (*creaClausulaWhere*) son usados para iniciar la construcción de la expresión en SQL. Al finalizar, el análisis semántico continúa con su proceso hasta que construye y ejecuta el comando en SQL para así retornar la información solicitada.

En [Verástegui, 2015] se implementó el módulo para el tratamiento de funciones de agregación, agrupamiento y subconsultas en una clase (*agregaAgrupaSubcons*) separada de los procesos de la interfaz. Al realizar esto se puede asegurar la integridad del núcleo de la interfaz, ya que los llamados al módulo se hacen de manera específica dentro del proceso, lo cual modifica los resultados de la interfaz sin necesidad de modificar los procesos internos del núcleo de la ILNBD.

Además, en ese trabajo se realizaron modificaciones a la estructura que almacena los componentes léxicos e información de la interfaz. Esto es necesario ya que dicha estructura contiene toda la información necesaria para construir la instrucción en SQL; por lo tanto, su modificación permite integrar la información necesaria para manejar funciones de agregación, agrupamiento y subconsultas.

Para la clasificación del tipo de frase, se complementó el código ya existente en la interfaz con cláusulas que permiten detectar y etiquetar funciones de agregación, agrupamiento y subconsultas.

3.1.1. Problemas en consultas

A continuación se presentan los problemas que fueron detectados en [Verástegui, 2015], los cuales pueden estar presentes en consultas que involucran funciones de agregación, agrupamiento y subconsultas; mientras que en las siguientes subsecciones, se presentan los problemas exclusivos para cada uno de los tres aspectos. Los problemas generales que existen en el corpus de Geobase se describen en la Tabla 3.1.

Tabla 3.1. Problemas existentes en el corpus de Geobase.

Tipos de consultas	Descripción
Deductivas	Están basadas en deducción por uso de inferencia.
Con negación	Las consultas que involucran negación usualmente incluyen las palabras *no*, *sin*, *excepto*, *nada*, *ningún*, etc.
Elipsis semántica	Es la omisión de elementos que, aunque no son obligatorios sintácticamente, son requeridos para una completa interpretación semántica de un texto.
Anáfora	Tienen la repetición de un elemento de la oración utilizando pronombres indicativos para referirse a algo mencionado previamente en la oración.
Ambigüedad en columna	Se presenta cuando existe una columna con el mismo nombre en más de una tabla y ambas tablas son detectadas en la consulta o cuando dos columnas tienen el mismo descriptor.
Ambigüedad COUNT/SUM	Se presenta cuando hay palabras que no pueden ser discriminadas entre las funciones de agregación COUNT y SUM.

3.1.2. Problemas en funciones de agregación

Entre las consultas que contienen funciones de agregación, se detectaron los problemas descritos en la Tabla 3.2.

Tabla 3.2. Problemas relacionados con FA.

Consultas que contienen funciones de agregación	Descripción
Validación de mayor que/menor que	Este problema surge debido al proceso de la interfaz en el cual sólo detecta la palabra mayor/menor y le asocia un signo de desigualdad sin verificar si dicha palabra se refiere a una función de agregación MAX o MIN.
Elipsis en FA	Consiste en la omisión de la columna a la cual se le aplica la función de agregación.

3.1.3. Problemas en agrupamiento

Los problemas detectados en las consultas con agrupamiento se describen en la Tabla 3.3.

Tabla 3.3. Problemas detectados en consultas con agrupamiento.

Consultas con agrupamiento	Descripción
Concatenación de palabras en cláusula Group by	Es el uso de una frase (con dos o más palabras) para referirse a una columna de la base de datos.
Doble agrupamiento	Caso en el cual se presenta un agrupamiento por dos columnas.
La palabra *por* no se refiere a Group by	Este caso ocurre cuando la consulta incluye la palabra *por*, pero no se refiere a la cláusula de agrupamiento Group by.
Elipsis en Group by	La elipsis en Group by se presenta cuando no se especifica una columna para realizar el agrupamiento.
Uso de condiciones	El uso de condiciones como Having dentro del agrupamiento complica la traducción de lenguaje natural a SQL, ya que añade más parámetros y procesos a la consulta.
Group by implícito en comando de SQL	Se presenta cuando la consulta no requiere un agrupamiento de manera explícita, pero al momento de analizar su traducción a SQL, se puede observar que se requiere un agrupamiento para obtener la información deseada.
Múltiples tablas	Se presenta cuando la consulta involucra más de una tabla.

3.1.4. Problemas en subconsultas

En el análisis sistematizado del corpus de Geobase realizado en [Verástegui, 2015], se detectó que existen al menos dos patrones en las consultas que involucran subconsultas. En este proyecto se manejaron sólo consultas con el patrón uno: columna | columna interna | FA. El patrón dos es cuando la consulta no tiene un patrón, y por lo tanto, complica la recopilación de los elementos necesarios para construir la subconsulta. Para el funcionamiento del tratamiento para subconsultas, es necesario que el tratamiento para funciones de agregación haya sido ejecutado previamente, esto es necesario ya que como se observa en el patrón uno, es necesario tener identificada la función de agregación para construir la subconsulta.

Se detectaron los problemas relacionado con subconsultas descritos en la Tabla 3.4.

Tabla 3.4. Problemas relacionados con subconsultas.

Consultas con subconsultas	Descripción
Ambigüedad de patrón	Se presenta cuando la consulta tiene el mismo patrón (columna | columna interna | función de agregación) que las consultas que contienen subconsultas; sin embargo, dicha consulta no involucra una subconsulta.
Más de una subconsulta	Este problema se presenta cuando una consulta tiene más de una subconsulta; es decir, se requieren dos o más subconsultas para construir la instrucción en SQL.
Uso de condiciones	El uso de condiciones como Having dentro de las subconsultas es un problema que complica la traducción de lenguaje natural a SQL.
Múltiples tablas	Consiste en el uso de más de una tabla en la consulta y subconsulta.
Uso de operadores de comparación	En las subconsultas se presentan casos en que la subconsulta se asocia a la consulta principal mediante el uso de operadores de comparación.

3.2. Análisis del procesamiento de consultas de la ILNBD

Para comprender mejor el proceso de análisis semántico de una consulta en lenguaje natural, es conveniente describir primero una estructura de datos que se usa para guardar información de las unidades léxicas (*tokens*) de una consulta, la cual se muestra en la Tabla 3.5.

Para efectuar la traducción de una consulta, se usa una estructura de datos donde se va registrando la información resultante del proceso de traducción. La Tabla 3.5 muestra la estructura de datos donde se registra la información generada para cada unidad léxica de una consulta. Una unidad léxica (*token*) puede ser tratada como un elemento independiente, que es una estructura que contiene un conjunto de atributos que son utilizados para procesar la consulta. Los atributos contenidos en cada estructura de las unidades léxicas son: *Componente léxico*, *Lema*, *Categoría* (indica la categoría gramatical), *Frase* (secuencia de palabras que contiene la unidad léxica), *Id. Frase* (almacena un conjunto de números, donde cada número representa la posición del componente léxico que pertenece a una frase), *Tipo de frase* (indica el tipo de frase de cada componente léxico; por ejemplo, Frase Select, Frase Where), *Etiqueta de columna* (indica la columna a la que se refiere el componente léxico), *Etiqueta final* (almacena el equivalente semántico en SQL del componente léxico) y *Marcado* (indica si el componente léxico es relevante para la traducción).

3.2.1. Procesamiento de consultas

En la capa Análisis Semántico (Figura 2.4) existen subcapas que realizan procesos necesarios para traducir la consulta. Una de esas subcapas consiste en la identificación de las frases Select y Where. Una vez que se tienen identificadas las tablas, columnas y valores de búsqueda, la consulta se divide en las frases Select y Where. Para tal efecto cada valor de búsqueda se asocia a una columna de acuerdo con su cercanía y tipo de dato. Los pares columna-valor de búsqueda constituyen la parte Where, y el resto de las columnas pertenecen a la cláusula Select.

A continuación se presenta un ejemplo del funcionamiento utilizando una consulta que incluye subconsulta.

3.2.2. Estado inicial

El proceso de traducción de una consulta inicia con la información que se muestra en las celdas resaltadas de la Tabla 3.5.

Tabla 3.5. Estructura de datos para almacenar información de unidades léxicas.

Atributo	**Unidad léxica de la consulta**								
Componente léxico	Puedes	decirme	la	tarifa	para	el	vuelo	número	16
Lema	puedes	decirme	el	tarifa	para	el	vuelo	número	16
Categoría	orden	orden	articulo	sustantivo	preposición	articulo	sustantivo	sustantivo	número
Frase									
Id. frase									
Tipo de frase									
Etiqueta de columna									
Etiqueta de tabla									
Etiqueta final									
Marcado	true	true	true	true	true	true	true	true	true

3.2.3. Análisis superficial

En el primer paso del proceso de traducción, se realiza un análisis superficial para identificar las categorías gramaticales de cada unidad léxica, tal como se ilustra en la Figura 3.2. La información resultante de este paso se muestra en las celdas de la Tabla 3.6 resaltadas con gris.

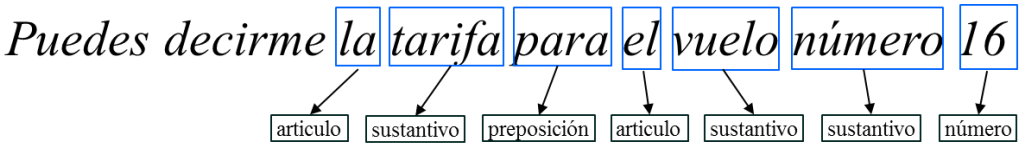

Figura 3.2. Análisis superficial de la consulta

Tabla 3.6. Ejemplo de la información generada durante el análisis léxico.

Atributo	**Unidad léxica de la consulta**								
Componente léxico	Puedes	decirme	la	tarifa	para	el	vuelo	número	16
Lema	puedes	decirme	el	tarifa	para	el	vuelo	número	16
Categoría	orden	orden	articulo	sustantivo	preposición	articulo	sustantivo	sustantivo	número
Frase									
Id. frase									
Tipo de frase									
Etiqueta de columna									
Etiqueta de tabla									
Etiqueta final									
Marcado	false	false	false	true	false	false	true	true	true

3.2.4. Identificación de tablas y columnas

En este paso del proceso se efectúa la identificación de tablas y columnas de la BD involucradas en la consulta. Para tal efecto, se buscan en el diccionario de información semántica (DIS) las unidades léxicas marcadas de la consulta como palabras individuales o frases de acuerdo a su categoría gramatical (nominales, verbales, adjetivales o preposicionales), tal como se ilustra en la Figura 3.3. Las celdas resaltadas de la Tabla 3.7 muestran la información resultante de este paso.

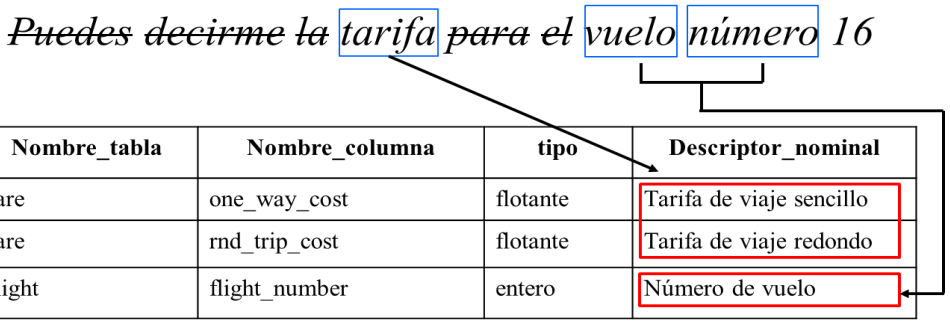

Figura 3.3. Identificación de tablas y columnas usando el DIS

Tabla 3.7. Ejemplo de la información generada durante la identificación de tablas y columnas.

Atributo	**Unidad léxica de la consulta**								
Componente léxico	Puedes	decirme	la	tarifa	para	el	vuelo	número	16
Lema	puedes	decirme	el	tarifa	para	el	vuelo	número	16
Categoría	orden	orden	articulo	sustantivo	preposición	articulo	sustantivo	sustantivo	número
Frase				tarifa			vuelo número	vuelo número	
Id. frase				3			6 - 7	6 - 7	
Tipo de frase									
Etiqueta de columna				fare.one_ way_cost, fare.rnd_ trip_cost			flight. flight_ number	flight. flight_ number	
Etiqueta de tabla									
Etiqueta final									16
Marcado	false	false	false	true	false	false	true	false	true

3.2.5. Separación de las frases Select y Where

En este paso del proceso, la identificación de las frases Select y Where se puede llevar a cabo después de haber identificado las tablas, columnas y valores de búsqueda involucrados en la consulta. Para ello, cada valor de búsqueda presente en la consulta es asociado con una columna considerando su cercanía y similitud de tipo de dato. El conjunto de pares de columnas y valores de búsqueda constituyen la frase Where. El resto de las unidades léxicas etiquetadas como columna son consideradas para la frase Select. La Figura 3.4 ilustra el proceso de separación; mientras que las celdas resaltadas de la Tabla 3.8 muestran la información generada en esta parte del proceso.

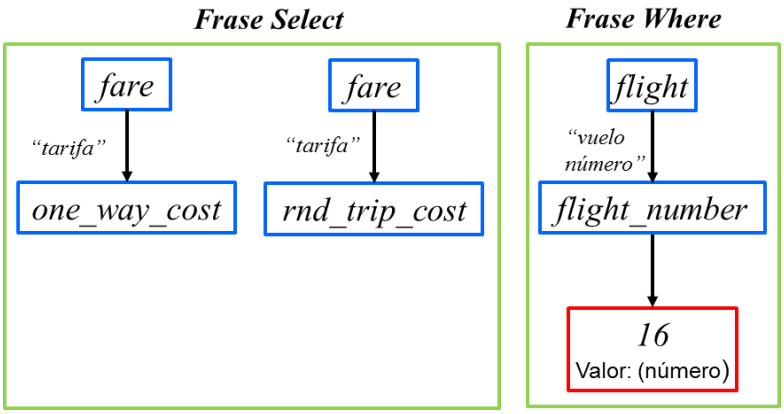

Figura 3.4. Separación de las frases Select y Where

Tabla 3.8. Ejemplo de la información generada durante la identificación de las frases Select y Where.

Atributo	**Unidad léxica de la consulta**								
Componente léxico	Puedes	decirme	la	tarifa	para	el	vuelo	número	16
Lema	puedes	decirme	el	tarifa	para	el	vuelo	número	16
Categoría	orden	orden	articulo	sustantivo	preposición	articulo	sustantivo	sustantivo	número
Frase				tarifa			vuelo número	vuelo número	
Id. frase				3			6 - 7	6 - 7	
Tipo de frase				frase Select			frase Where		frase Where
Etiqueta de columna				fare.one_ way_cost, fare.rnd_ trip_cost			flight**.** flight_ number	flight**.** flight_ number	
Etiqueta de tabla									
Etiqueta final							flight**.** flight_ number		16
Marcado	false	false	false	true	false	false	true	false	true

3.2.6. Determinación de reuniones implícitas

Una vez que se han identificado todas las tablas y columnas involucradas en la consulta, algunas veces la consulta no puede ser traducida a SQL, porque el grafo que representa estas tablas y sus relaciones no es conexo. En este caso se necesita incluir tablas adicionales (no mencionadas en la consulta en lenguaje natural) y arcos (relaciones entre tablas) con el fin de generar un grafo conexo.

La Figura 3.5 muestra el caso del grafo correspondiente a la consulta cuya información aparece en la Tabla 3.8. La información de esta tabla solamente menciona a las tablas *fare* y *flight*, para las cuales no existe una relación directa entre ellas. Por lo tanto, para traducir la consulta a SQL, es necesario incluir una o varias tablas, que en este caso es la tabla *flight_fare*. Para resolver este problema, se usa un método heurístico, el cual consiste en realizar un recorrido en amplitud del grafo semántico que representa todas las tablas de la base de datos y relaciones entre éstas con el fin de encontrar el conjunto más pequeño de tablas para generar un subgrafo conexo que incluya todas las tablas identificadas previamente. La Figura 3.6 muestra la información que se obtiene al realizar este proceso para la consulta cuya información se encuentra en la Tabla 3.8.

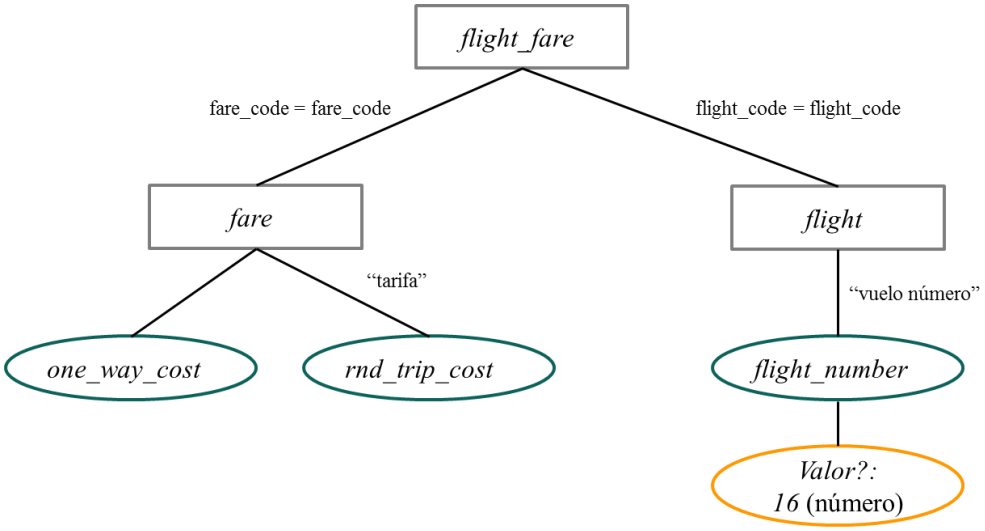

Figura 3.5. Ejemplo de un grafo semántico.

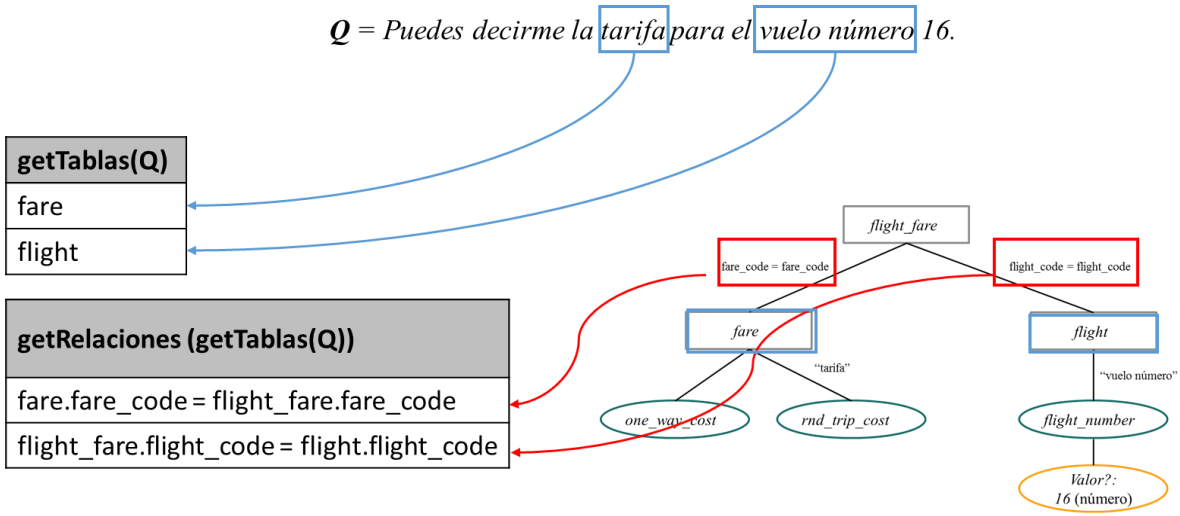

Figura 3.6. Ejemplo de un grafo semántico con sus relaciones entre tablas.

3.2.7. Generación de la instrucción en SQL

A continuación se presentan los pseudocódigos de los procesos que se efectúan en el módulo de análisis semántico para generar consultas en SQL.

El Algoritmo 3.1 es usado para crear la cláusula Select de la consulta en SQL. Para ello, se concatenan las columnas de la BD a las que hacen referencia las unidades léxicas marcadas para la frase Select. Además, se añaden las tablas involucradas en la consulta en la cláusula From.

Algoritmo 3.1. Pseudocódigo del método crearClausulaSelect(Q)

```
1:    if Q ≠ Ø
2:        return "SELECT " + obtenerColumnasSelect(Q) + " FROM " +
3:                    obtenerTablas(Q)
4:    endif
5:    return Ø
```

El Algoritmo 3.2 presenta el funcionamiento del método *crearClausulaWhere*, donde primero se verifica si hay valores de búsqueda y unidades léxicas marcados para la frase Where (línea 3). Posteriormente, se verifica si existen más de una tabla involucradas en la consulta, y en caso afirmativo se generan las reuniones entre dichas tablas (líneas 5 a 9).

Algoritmo 3.2. Pseudocódigo del método crearClausulaWhere(Q)

```
1:    W ← Ø
2:    //Si existen unidades léxicas marcados para la frase Where y valores de búsqueda
3:    if obtenerTokensWhere(Q) > 0 and hayValoresDeBusqueda(Q)
4:        W ← concatenarColumnasyValores(Q)
5:        if obtenerTablas(Q) > 1 //Si la consulta tiene más de una tabla diferente
6:            if W = Ø  //Si no es necesario concatenar columnas con valores de búsqueda
7:                W = " WHERE " + crearJoins(obtenerTablas(Q)) //crear reuniones
8:            else  //Si es necesario concatenar columnas con valores de búsqueda
9:                W = " WHERE " + crearJoins(obtenerTablas(Q)) + " AND " + W
10:       else //Si la consulta involucra una tabla solamente
11:           if W = Ø
12:               return W
13:           else //Si la consulta involucra una tabla y contiene valores de búsqueda
14:               return " WHERE " + W
15:   else obtenerTablas(Q) > 1 //Si la consulta tiene más de una tabla en la cláusula Select
16:                             //y sin valores de búsqueda
17:       return " WHERE " + crearJoins(obtenerTablas(Q))
18:   endif
19:   return Ø
```

El Algoritmo 3.3 muestra el método *evaluaFrases*, el cual es usado por la interfaz para construir la consulta en SQL. En este método se ejecutan los métodos *crearClausulaWhere()* y *crearClausulaSelect()*, los cuales son empleados para generar las cláusulas Where y Select respectivamente.

Algoritmo 3.3. Pseudocódigo del método convertirClausulaSelectWhere (Q)

```
1:    W ← crearClausulaWhere(Q)
2:    S ← crearClausulaSelect(Q)
3:    SQL ← Ø
4:    if obtenerTablas(Q) > 1
5:        SQL ← S + obtenerTablasTotales(Q) + W
6:    else
7:        SQL ← S + W
8:    endif
9:    return SQL
```

La Tabla 3.9 muestra algunas funciones que existen en una versión anterior de la ILNBD [Aguirre, 2014], las cuales permiten obtener de la estructura de datos de la consulta (Tabla 3.10), las tablas, columnas y valores de búsqueda de la consulta.

Tabla 3.9 Funciones para extraer información de la estructura de datos de la consulta.

Función	Descripción
getTablas(*consulta*) o addTablas(*consulta*)	Obtiene las tablas involucradas en la consulta
addColumnas(*consulta*)	Obtiene las columnas marcadas como frase Select de la consulta
addColumnasValores(*consulta*)	Obtiene las columnas con valores marcados como frase Where de la consulta. Formato *columna operador valor*.
getRelaciones(this.getTablas(*consulta*))	Obtiene las relaciones entre tablas. Formato: *primera_tabla.llave_foranea = segunda_tabla.llave_foranea*.

3.3. Procesamiento de consultas complejas que requieren subconsultas

Para comprender mejor el proceso de consultas complejas (tal como se encuentra implementado desde una versión anterior de la ILNBD [Aguirre, 2014]), es necesario entender el grafo semántico que genera la ILNBD. En dicho grafo los nodos representen a las tablas involucradas en la consulta y los arcos representen relaciones entre tablas (llaves foráneas).

En el grafo semántico se debe marcar con etiqueta Select el nodo o nodos que correspondan a las tablas que pertenezcan a la frase Select y marcar con etiqueta Where el nodo o nodos que correspondan a las tablas que pertenecen a la frase Where.

Si el grafo contiene un nodo en la etiqueta Select y un nodo con etiqueta Where, se trata de una consulta simple y se debe utilizar el método para generar consultas con subconsultas (Algoritmo 3.4). Si el grafo contiene un nodo con etiqueta Select y dos o más nodos con etiqueta Where, se trata de una consulta compleja y el Algoritmo 3.4 puede ser usado para tratar este tipo de consulta. Si el grafo contiene dos o más nodos con etiqueta Select y dos o más nodos con etiqueta Where, se debe utilizar el procedimiento para generar la expresión en SQL con reuniones (Algoritmos 3.1 a 3.3).

Para explicar cómo se genera la expresión en SQL primero con reuniones y después con subconsultas, se considera la siguiente consulta en LN:

Muéstrame el costo de clase Business para el vuelo número 1.

La interfaz al procesar la consulta obtiene los datos contenidos en la Tabla 3.10. Dichos datos son utilizados por la interfaz para determinar reuniones implícitas entre tablas involucradas en la consulta (según la versión [Aguirre, 2014]). Asimismo, se construye la consulta en SQL

determinando la cláusula Select y la cláusula Where, la cual se compone de las reuniones y los valores de búsqueda.

Tabla 3.10. Estructura de datos con la información generada para las unidades léxicas.

Atributo	**Unidad léxica de la consulta**										
Componente léxico	Muéstrame	el	costo	de	clase	business	para	el	vuelo	número	1
Lema	muéstrame	el	costo	de	clase	business	para	el	vuelo	número	1
Categoría	orden	articulo	sustantivo	preposición	sustantivo	nombre	preposición	articulo	sustantivo	sustantivo	numero
Frase			costo		clase				vuelo número	vuelo número	
Id. frase			2		4				8-9	8-9	
Tipo de frase			frase Select		frase Where	frase Where			frase Where		frase Where
Etiqueta de columna			fare.one_way_cost, fare.rnd_trip_cost		compound_class.class_type				flight. flight_number	flight. flight_number	
Etiqueta de tabla											
Etiqueta final					compound_class. class_type	Business			flight. flight_number	flight. flight_number	1
Marcado	false	false	true	false	true	true	false	false	true	true	true

El grafo semántico generado por la interfaz a partir de la Tabla 3.10 para dicha consulta se muestra en la Figura 3.7. Como se puede observar en el grafo, la consulta tiene un nodo con etiqueta Select (*fare*) y dos nodos con etiqueta Where (*compound_class* y *flight*); por lo tanto la consulta es compleja.

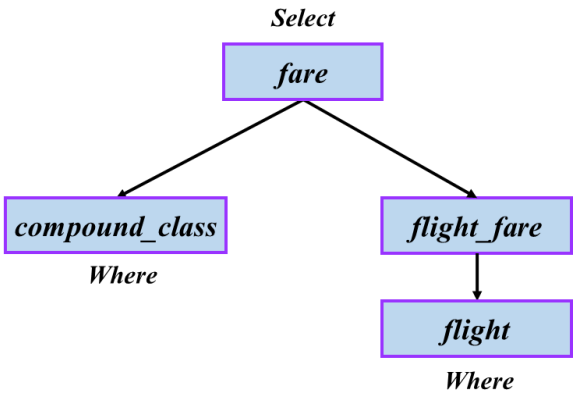

Figura 3.7. Grafo semántico en forma de árbol.

A continuación se muestran dos versiones de la consulta en SQL una con reuniones y otra con subconsultas. En la versión anterior de la ILNBD, la expresión en SQL se generaba con reuniones y en la versión actual, con subconsultas.

Consulta traducida a SQL con reuniones:

```
SELECT fare.one_way_cost, fare.rnd_trip_cost
FROM fare, compound_class, flight, flight_fare
WHERE compound_class.fare_class = fare.fare_class
AND fare.fare_code = flight_fare.fare_code
AND flight_fare.flight_code = flight.flight_code
AND flight.flight_number = 1
AND compound_class.class_type LIKE 'Business'
```

Consulta traducida a SQL con subconsultas:

```
SELECT fare.one_way_cost, fare.rnd_trip_cost
FROM fare
WHERE fare.fare_class IN (
        SELECT compound_class.fare_class
         FROM compound_class
          WHERE compound_class.class_type LIKE 'Business')
           AND fare.fare_code IN (
                SELECT flight_fare.fare_code
                FROM flight_fare
                WHERE flight_fare.flight_code IN (
                        SELECT flight.flight_code
                        FROM flight
                        WHERE flight.flight_number = 1))
```

Al observar el grafo semántico (Figura 3.7) y la consulta traducida a SQL, se observa que la consulta más externa corresponde al nodo raíz del árbol (el cual representa la tabla *fare*). Esta consulta tiene una cláusula Where, la cual contiene dos condiciones de búsqueda (combinadas con un operador AND): una corresponde al subárbol izquierdo, y la otra corresponde al derecho. Además, la subconsulta encerrada en el rectángulo con línea continua corresponde al subárbol izquierdo e involucra a la tabla *compound_class* que se encuentra en este subárbol; mientras que la doble subconsulta encerrada en el rectángulo con línea segmentada corresponde al subárbol derecho e involucra a las tablas *flight_fare* y *flight* que se localizan en el subárbol derecho.

En la consulta con subconsultas se puede apreciar el recorrido que debe realizar el proceso de la traducción de la interfaz para poder cubrir las reuniones entre tablas. Dicho recorrido se encuentra dividido en dos partes. Esto ocurre debido a que el nodo marcado con etiqueta Select se encuentra conectado directa o indirectamente con dos nodos marcados con etiqueta Where. Esto hace necesaria la implementación de un módulo recursivo, el cual se explica a continuación.

A continuación se describe conceptualmente el proceso para generar una expresion en SQL con subconsultas. Para tal efecto, considérese el conjunto *Pend* y una pila *Temp* (del tipo primero en entrar y primero en salir), los cuales se usan para almacenar nodos del árbol.

1. Hacer *Pend* igual al conjunto de nodos del árbol e insertar en la pila *Temp* el nodo con etiqueta Select.

2. Extraer un nodo *v* de la pila *Temp*. Hacer *ExpresiónSQL = CláusulaSelect(v) + CláusulaFrom(v)* correspondientes al nodo *v*. La información para estas cláusulas puede obtenerse de la estructura de datos del análisis semántico de la consulta.

3. Encontrar el conjunto *Hijos* de los nodos que sean hijos de *v* y que pertenezcan a *Pend - Temp*.

4. En caso de que |*Hijos*| ≥ 1, obtener *Subexpresion1* para el subárbol cuya raíz es el 1er. hijo de *v*. Hacer *ExpresionSQL = ExpresionSQL* + C*lausulaWhere*(*v, 1*) + 'IN ('+ *Subexpresion1* + ')'; obtener *Subexpresion2* para el subárbol cuya raíz es el 2do. hijo de *v*. Hacer *ExpresionSQL = EspresionSQL* + ' AND ' + *ClausulaWhere*(*v, 2*) + ' IN (' + *Subexpresion2* + ')' ; y así sucesivamente con el resto de los hijos de *v*.

5. En caso de que | Hijos | = 0, hacer *ExpresionSQL = ExpresionSQL + ClausulaWhere*(*v*).

6. Hacer *Pend = Pend – v* e insertar en *Temp* los nodos en *Hijos*.

7. Se repiten los pasos 3, 4, 5 y 6 hasta que *Pend* = Ø.

El seudocódigo mostrado en el Algoritmo 3.4 se implementó en la ILNBD para procesar consultas que involucran FA, agrupamiento y subconsultas. En la línea 1 se recolecta la información necesaria para detectar, solucionar problemas y asociar información de los elementos de la consulta que requieran FA o agrupamiento mediante el método *agrega_agrupa_subcons* (ver Apéndice A). En las líneas 2 a la 5 se inicializan las variables para construir las cláusulas Select y Where. Posteriormente, se obtienen los datos de la FA y agrupamiento que se hayan detectado en las primeras líneas del algoritmo (líneas 6 y 7). En la variable *SQL* se almacena el resultado en SQL de la consulta que se va construyendo (línea 8). En las líneas 9 y 10 se obtiene la información sobre las tablas involucradas en la consulta (incluyendo las tablas de las reuniones implícitas). En las líneas 11 a 15 se construyen las cláusulas Select y From del primer nivel de las subconsultas. Después, se inicializa la variable *pend* con las tablas involucradas en la consulta (líneas 16 a 19). Por último, se inicia el proceso de construcción de subconsultas ejecutando el código del Algoritmo 3.5.

Los seudocódigos de los pasos explicados anteriormente se muestran a continuación:

Algoritmo 3.4. Método general para procesar consultas con FA, agrupamiento y subconsultas

```
1:  consulta ← agrega_agrupa_subcons(consulta) //Llena la tabla consulta con la información
                                                //requerida para el tratamiento de funciones de agregación
2:  TS ← addTablas(consulta) //Se obtienen tablas involucradas en la consulta
3:  CS ← addColumnas(consulta) //Se obtienen tablas y columnas de la cláusula Select
4:  CV ← addColumnaValores(consulta) //Contiene las columnas de la cláusula Where con sus
                                     //valores de búsqueda
5:  TG ← getTablas(consulta) //Se obtienen todas las tablas de las cláusulas Select y Where
6:  FA ← getFA(consulta) //Se obtiene la función de agregación con su respectiva columna
```

```
7:  GB ← getGroupBy(consulta) //Se obtiene la columna de la cláusula Group by
8:  SQL ← "SELECT"
9:  relaciones ← getRelaciones(TG) //Se obtienen las relaciones entre pares de tablas
10: relaciones ← convertirRelaciones(relaciones)
11: pend ← Ø
12: for i = 0 to |CS| //Se construye cláusula Select
13:     SQL ← SQL+ CS[i] + ","
14: endfor
15: SQL ← SQL+ "FROM" + TS[0]  //Se construye la cláusula From
16: for i=0 to |relaciones|  //Se añaden todas las tablas de la consulta a pend
17:     pend ← relaciones[i]
18: endfor
19: actual ← pend[0]  //Se toma el primer elemento de pend
20: pend ← pend - pend[0] //Se remueve el primer elemento de pend
21: construirSubconsultas(actual) // Se inicia el proceso de construcción de subconsultas
```

El Algoritmo 3.5 muestra el seudocódigo que describe la construcción de subconsultas a partir de un nodo (tabla). En la variable *subexpresión* se almacena la subexpresión (parte de la subconsulta) en SQL correspondiente al nodo que se está procesando. En la variable *expresión* se va agregando cada subexpresión correspondiente a cada nodo involucrado en la consulta. Primero, se realiza una verificación de nodos pendientes para saber si ya no hay nodos por procesar (línea 2). Posteriormente, se obtienen las conexiones entre el nodo actual y los nodos que se encuentran conectados con éste y que además están en *pend* (líneas 3 a 8), lo anterior se realiza para depurar la lista de hijos del nodo actual, ya que éste tiene conexión con otros nodos que no se utilizarán en la consulta (nodos que no están en *pend*). Por cada conexión entre nodos, se obtienen las llaves foránea y primaria que conectan dichos nodos y se construye la subexpresión correspondiente a la parte de la subconsulta correspondiente (líneas 10 a 24). Si al procesar un nodo no existen más hijos (conexiones con otros nodos), entonces se añaden los valores de búsqueda para ese nodo y se añaden a la subexpresión (líneas 25 a 41). Ya que no existen más hijos para el nodo, se añade la subexpresión a la expresión y se cierran los paréntesis correspondientes a los niveles de la subconsulta (línea 42). Por último, cuando ya no hay nodos por procesar termina el proceso de construcción de subconsultas añadiendo los valores de búsqueda restantes y añadiendo la última subexpresión a *expresión* (líneas 43 a 52).

Algoritmo 3.5. Método para construir subconsultas (*nodo actual*)

```
1:  subexpresión ← Ø
2:  if |pend| ≠ Ø //Mientras que existan nodos pendientes
3:      hijos ← obtenerHijos(actual) //Se obtiene un vector de las tablas conectadas con el nodo actual
4:      for i = 0 to |hijos| //Elimina los elementos de hijos que no se utilizarán en la consulta
5:          if !contains(pend, hijos[i]) //Si el elemento hijo[i] no está en pend, se elimina
6:              hijos ← hijos - hijos[i]
7:          endif
8:      endfor
9:      if |hijos| ≥ 1 //Si hay por lo menos un nodo válido
10:         for i = 0 to |hijos|
11:             ruta ← obtenerFKs(actual, hijos[i])  //Se obtienen las llaves foráneas entre el nodo
                                                     //actual y su hijo
12:             FK1 ← ruta[0] //Tabla y columna de 1ra. parte de llave foránea
13:             FK2 ← ruta[1] //Tabla y columna de 2da. parte de llave foránea
```

```
14:             temp ← split(FK2, ".") //Se separan la tabla y la columna
15:             tablaFrom ← temp[0] //Se obtiene la tabla a la que pertenece el hijo
16:             if i = 0 //Si es el inicio de la subconsulta
17:                 subExpresion ← "WHERE" + FK1 + "IN(SELECT" + FK2 + "FROM" + tablaFrom)
18:             else //Si es continuación de otra subconsulta
19:                 subExpresion ← "AND" + FK1 + "IN(SELECT" + FK2 + "FROM" + tablaFrom)
20:             endif
21:             expresion ← expresion + SubExpresion //Concatena las subexpresiones
22:             pend = pend - hijos[i] //Elimina el nodo de pend
23:             construirSubconsultas(hijos[i]) //Realizar las subconsultas correspondientes a partir del
                                                //hijo i
24:         endfor
25:     else //Si es el último nodo del recorrido
26:         k ← 0
27:         for j = 0 to |CV| //Encuentra el valor de búsqueda que concuerde con el nodo actual
28:             temp ← split(CV[j], ".")
29:             tablaCV ← temp[0]
30:             if actual = tablaCV
31:                 if k > 0   //Si no es el primer valor de búsqueda a insertar incluirlo con AND
32:                     subExpresion ← " AND " + CV[j]
33:                     expresion ← expresion + subExpresion
34:                 else   //Si es el primer valor de búsqueda a insertar incluirlo con cláusula Where
35:                     subExpresion ← " WHERE " + CV[j]
36:                     expresion ← expresion + subExpresion
37:                 endif
38:                 k ← k + 1
39:             endif
40:         endfor
41:     endif
42:     expresion ← expresion + " ) " //Cerrar los paréntesis de cada subconsulta
43: else
44:     for j = 0 to |CV| //Insertar los valores de búsqueda del último nodo
45:         temp ← split(CV[j], ".")
46:         tablaCV ← temp[0]
47:         if actual = tablaCV
48:             subExpresion ← " WHERE " + CV[j]
49:             expresion ← expresion + subExpresion
50:         endif
51:     endfor
52: endif
```

3.4. Incorporación de algoritmos para tratamiento de FA y agrupamiento

Se adaptó el código [Verástegui, 2015] y se incorporó el módulo para el tratamiento de funciones de agregación y agrupamiento a la nueva versión de la interfaz.

Dentro del proceso Análisis Semántico (Figura 1.1) se llaman a los métodos del módulo Identificación de tablas y columnas, Identificación de frases y Análisis de funciones de

agregación y agrupamiento. Los métodos que se mencionaron anteriormente forman parte de una sola clase, la cual contiene los módulos completos.

El primer paso del análisis semántico es la obtención de atributos, los cuales se utilizan para ignorar palabras irrelevantes. El siguiente paso que se realiza es la identificación de tablas y columnas, mediante la búsqueda sistemática en el diccionario de información semántica (DIS). Una vez detectadas las tablas y columnas involucradas en la consulta, se efectúa la identificación de frases, las cuales pueden ser Select o Where. Su etiquetado se realiza en base a las columnas detectadas.

En el módulo de análisis de funciones de agregación [Verástegui, 2015] se encuentran varios métodos que son los siguientes: *validacionMayorqueMenorque()*, *deteccionFAAgrupamiento()*, *solucionAmbiguedad()*, *deteccionElipsisFA()*, y *asociacionColumnaGroupBy()*, los cuales fueron incorporados a la nueva versión de la interfaz. En la Tabla 3.11 se muestra una explicación general de cada uno de los métodos mencionados.

Estos métodos son utilizados para efectuar el tratamiento de funciones de agregación y agrupamiento, los cuales son los encargado de detectar, clasificar y marcar las columnas y palabras que se refieren a funciones de agregación y agrupamiento de forma que la interfaz pueda detectarlas y utilizarlas en la construcción de la instrucción en SQL.

Una vez detectadas las funciones de agregación y agrupamiento, el método de detección de subconsultas determina si la consulta cumple con un patrón para detección de valores de búsqueda que requieren subconsultas. En caso de que la consulta no cumpla con el patrón, el método para crear la cláusula Select (*creaClausulaSelect*) y el método para crear la cláusula Where (*creaClausulaWhere*) inician la construcción de la instrucción en SQL.

Tabla 3.11. Explicación de los algoritmos presentados en [Verástegui, 2015].

Algoritmo	Explicación
Algoritmos para tratamiento funciones de agregación y Group by.	
1. Validación mayor que, menor que	Tiene como objetivo discriminar entre las funciones de agregación MAX y MIN y las desigualdades (>, <).
2. Método *búsquedaFraseSID*	Tiene como finalidad extraer la función de agregación *fa* del diccionario de información semántica (DIS).
3. Solución de ambigüedad	Resuelve la ambigüedad existente entre las funciones de agregación COUNT y SUM.
4. Solución de elipsis en funciones de agregación	Tiene como propósito evitar que una consulta sea traducida a SQL sin haber identificado una columna a la cual aplicar la función de agregación.
5. Detección de funciones de agregación y agrupamiento	Analiza cada componente léxico que aún no haya sido identificado en la consulta. Al detectar una posible referencia, corrobora si efectivamente se trata de una función de agregación o cláusula Group by.
6. Asociación de columna para Group by	Detecta y modifica la columna que debe ser usada para realizar el agrupamiento en caso de que la consulta requiera agrupamiento.
Algoritmos para tratamiento de subconsultas	
7. Validación de patrón	Verifica si la consulta requiere el uso de subconsultas de acuerdo a un patrón.
8. Recopilación de elementos de la subconsulta	Tiene como función detectar los elementos necesarios para construir la subconsulta en SQL.
9. Asociación de la cláusula Where	Tiene como finalidad localizar la columna que debe ser usada para asociar el valor obtenido por la subconsulta.

3.5. Procesamiento de subconsultas combinadas con FA y agrupamiento

Para ejemplificar lo expuesto en los párrafos anteriores, se considera la siguiente consulta en LN:

Cuántos ríos corren a través del estado de Texas

La interfaz al procesar la consulta obtiene los datos contenidos en la Tabla 3.12. Dichos datos son utilizados por la interfaz para determinar funciones de agregación y agrupamiento, reuniones implícitas entre tablas involucradas en la consulta; asimismo, se construye la consulta en SQL determinando la cláusula Select, la cláusula Where donde se compone con subconsultas y valores de búsqueda.

Tabla 3.12. Estructura de datos con la información generada para las unidades léxicas.

Atributo	**Unidad léxica de la consulta**								
Componente léxico	Cuántos	ríos	corren	a	través	del	estado	de	Texas
Lema	cuánto	río	correr	a	través	del	estado	de	Texas
Categoría	pronombre	sustantivo	verbo	preposición	sustantivo	pronombre	sustantivo	preposición	nombre
rase		río					estado		
Id. frase		1					6		
Tipo de frase	frase Select	frase Select					frase Where		frase Where
Etiqueta de columna		river**.** river_name					state**.** state_name		
Etiqueta de tabla									
Etiqueta final	COUNT	river**.** river_name					state**.** state_name		Texas
Marcado	true	true	false	false	false	false	true	false	true

Es importante mencionar que para realizar el procesamiento de la consulta mencionada se pueden aplicar los algoritmos 3.4 y 3.5, donde se procesan las funciones de agregación y agrupamiento y se construyen las subconsultas correspondientes. Cabe mencionar que los algoritmos mencionados en las líneas 1 a 5 del Algoritmo 3.4 son parte del trabajo desarrollado en [Verástegui, 2015].

En la Figura 3.8 se muestra la arquitectura de la interfaz, donde los módulos con línea discontinua son usados para obtener los datos mostrados en la Tabla 3.12. Dichos datos posteriormente son usados para construir la consulta en SQL (módulos con línea continua) previa verificación del número de tablas detectadas para la cláusula Select.

Cuando se detecta que la consulta involucra una sola tabla en la cláusula Select, entra en funcionamiento el módulo *Generación de subconsultas*, el cual se encarga de construir las subconsultas que se requieran. Dentro del proceso recursivo ejecutado en el módulo ya mencionado, se añaden los valores de búsqueda detectados con anterioridad.

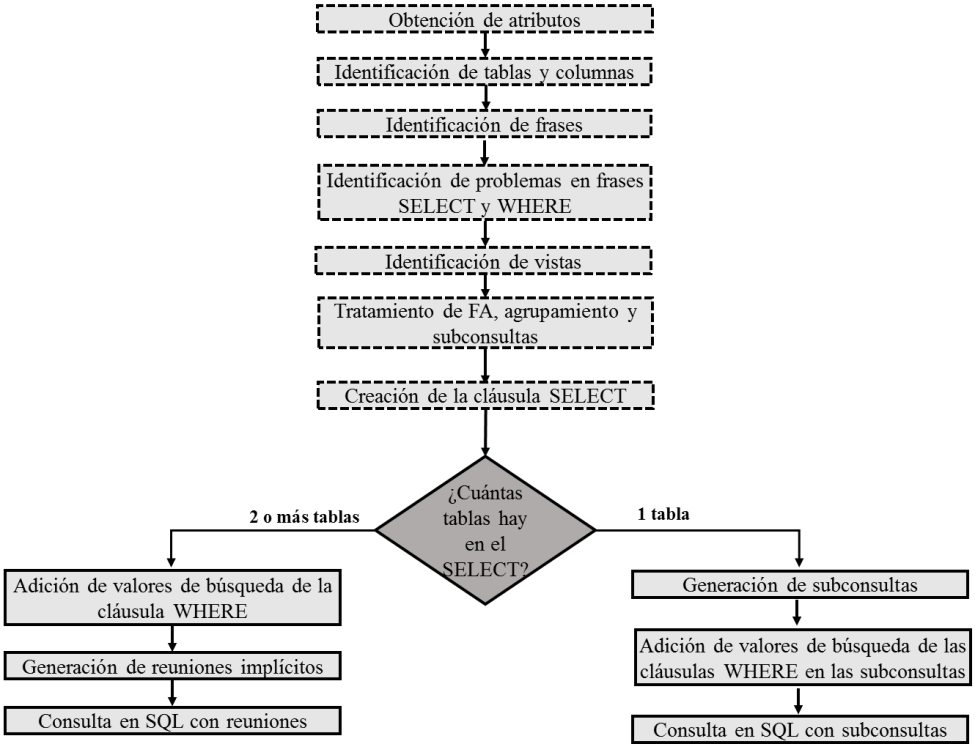

Figura 3.8. Diagrama de flujo para generación de una instrucción en SQL con subconsultas.

Capítulo 4. Experimentación

4.1. Descripción del hardware y software del equipo.

A continuación se presentan las características del hardware y software del equipo utilizado para realizar las pruebas funcionales.

Las pruebas fueron realizadas con el hardware especificado en la Tabla 4.1.

Tabla 4.1. Características del hardware

Características	Especificaciones
Procesador	Pentium(R) Dual-Core @ 2 GHz
Memoria	3 GB
Sistema operativo	Windows 7 Ultimate

Las pruebas fueron realizadas con el software especificado en la tabla 4.2.

Tabla 4.2. Características del software

Características	Especificaciones
Entorno	Netbeans IDE 8.1
Lenguaje	Java 1.7

4.2. Pruebas funcionales con consultas simples

Se realizaron pruebas funcionales con la versión anterior de la ILNBD y la nueva versión con consultas simples. En este tipo de consultas su grafo semántico tiene sólo un nodo con etiqueta Select y sólo un nodo con etiqueta Where. El propósito de estas pruebas fue verificar que la nueva versión generara las expresiones en SQL con subconsultas, de tal manera que éstas fueran semánticamente equivalentes a las generadas por la versión anterior. Es conveniente aclarar que estas consultas no involucran funciones de agregación ni agrupamiento. Para tal efecto, se listan enseguida cinco consultas de prueba, cuyos resultados se muestran en las Figuras 4.1.a, 4.1.b, hasta 4.5.a, 4.5.b.

- Consulta 1: *¿Puedes decirme la tarifa para el vuelo número 16?*
- Consulta 2: *¿Cuánto cuestan los vuelos número 1, 2, 3, 4 y 5?*
- Consulta 3: *¿Cuánto cuesta volar desde BOS a OAK sencillo?*
- Consulta 4: *Lista tarifas de viaje redondo desde DFW a ATL.*
- Consulta 5: *Desde OAK a BOS ¿qué tarifa es?*

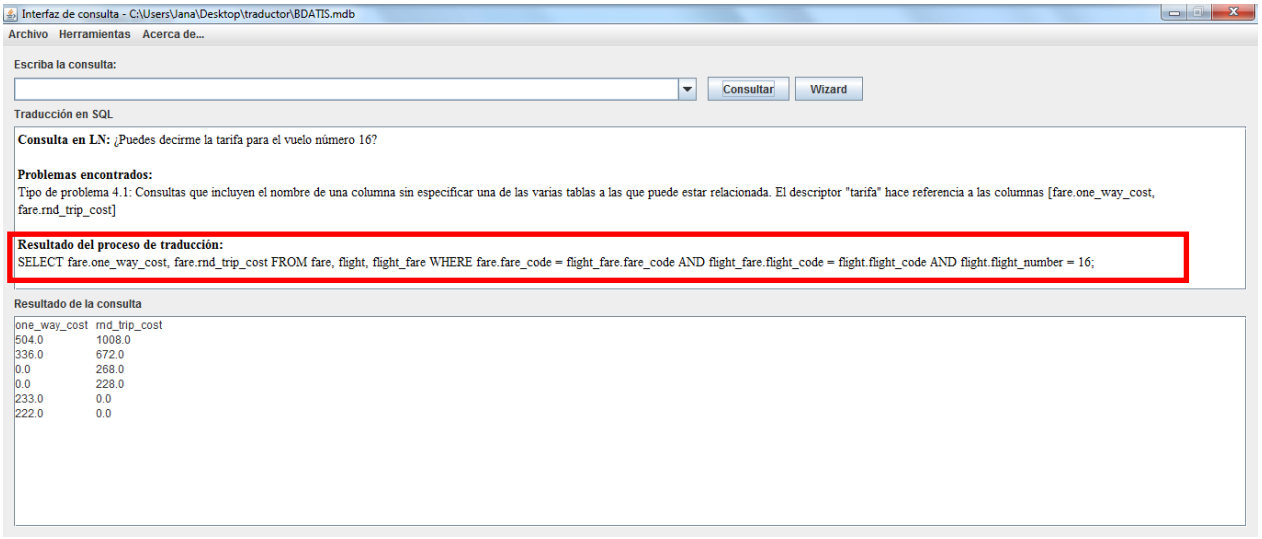

Figura 4.1(a). Respuesta a la consulta 1 (reuniones).

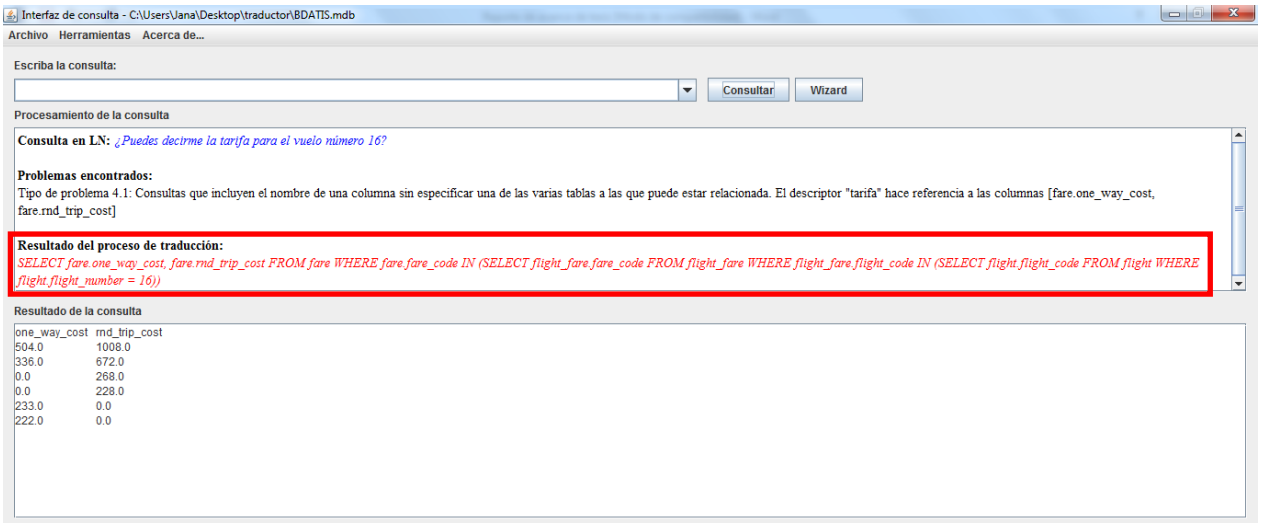

Figura 4.1(b). Respuesta a la consulta 1 (subconsultas).

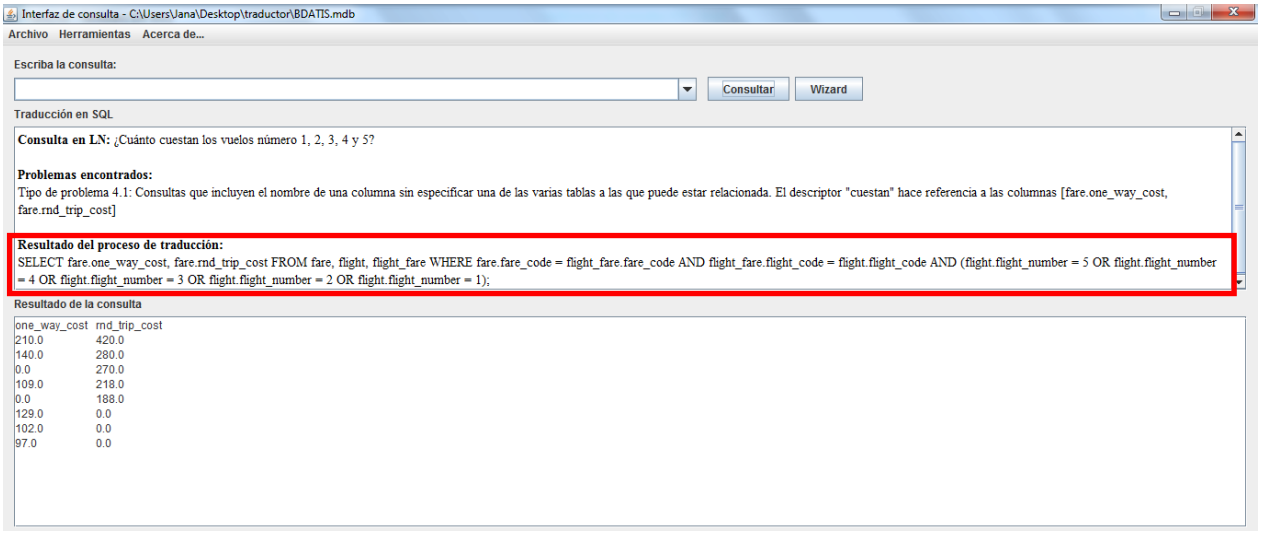

Figura 4.2(a). Respuesta a la consulta 2 (reuniones).

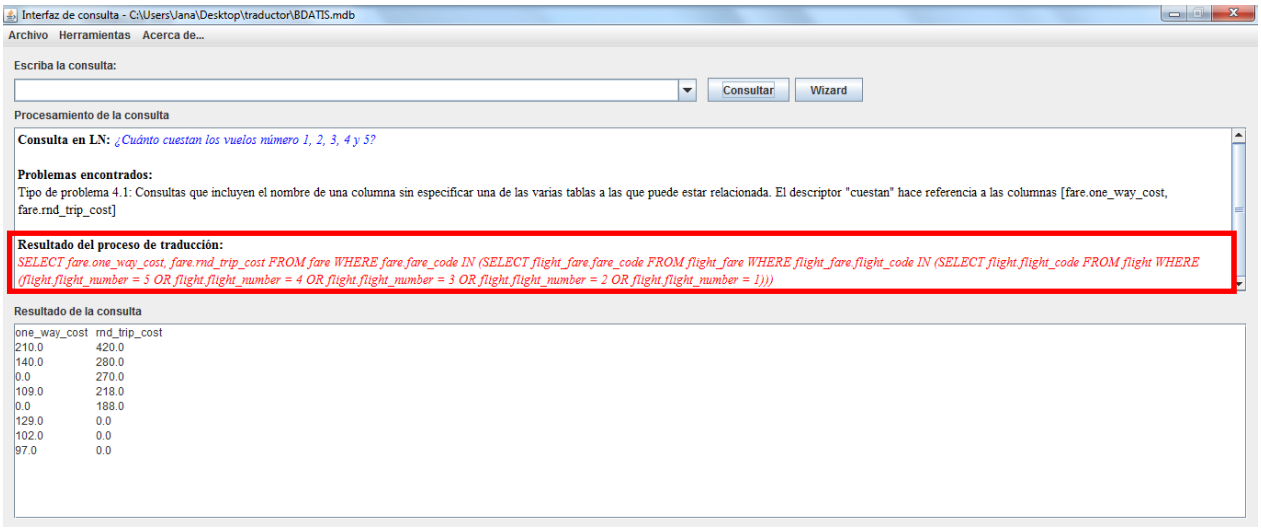

Figura 4.2(b). Respuesta a la consulta 2 (subconsultas).

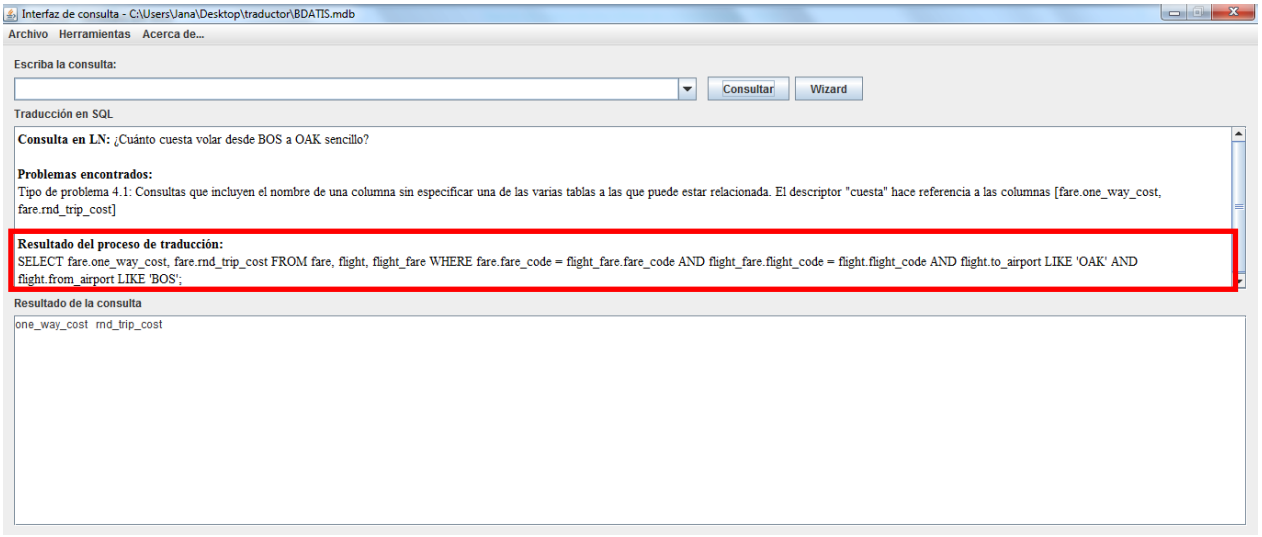

Figura 4.3(a). Respuesta a la consulta 3 (reuniones).

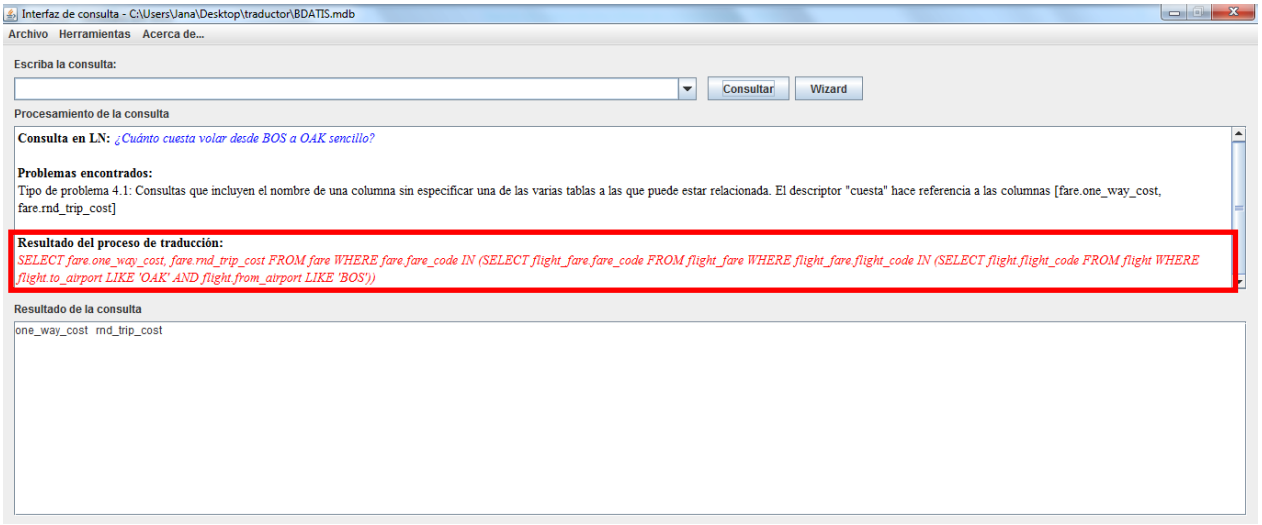

Figura 4.3(b). Respuesta a la consulta 3 (subconsultas).

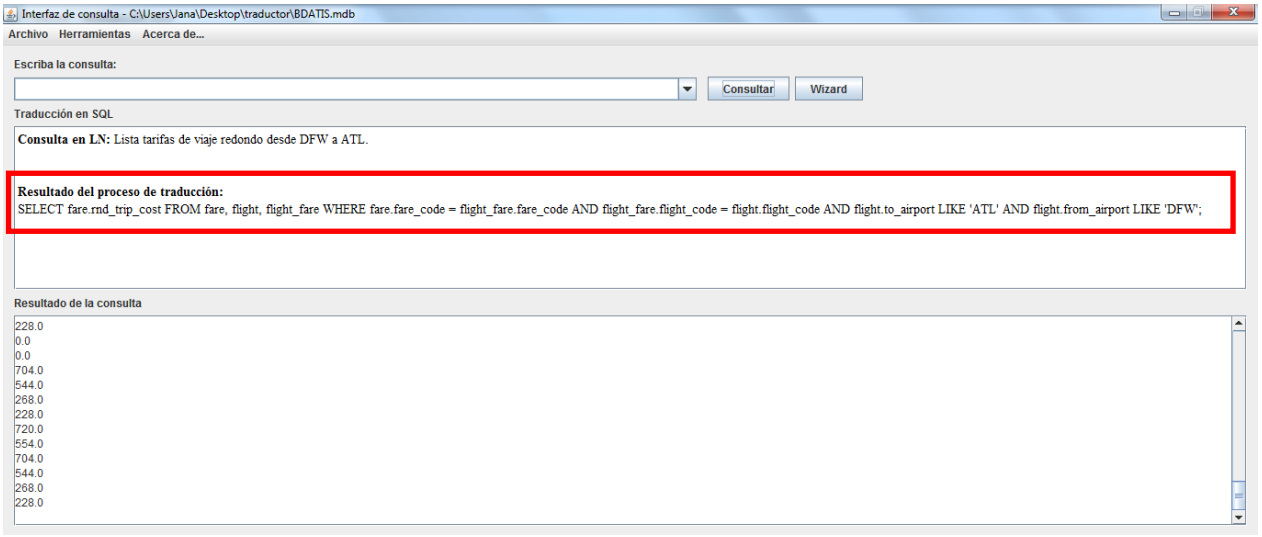

Figura 4.4(a). Respuesta a la consulta 4 (reuniones).

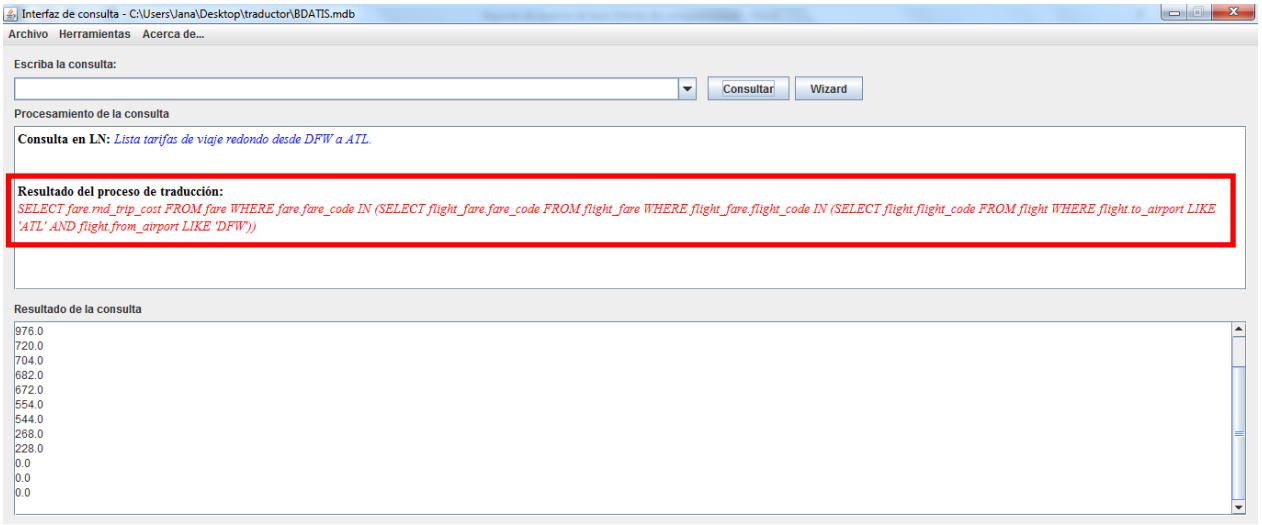

Figura 4.4(b). Respuesta a la consulta 4 (subconsultas).

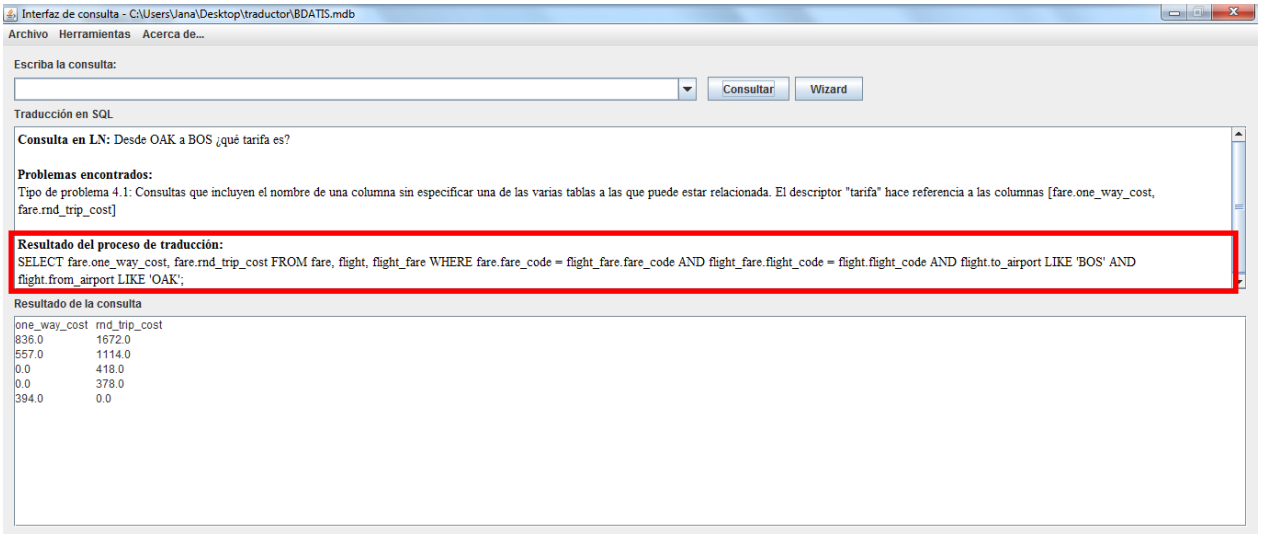

Figura 4.5(a). Respuesta a la consulta 5 (reuniones).

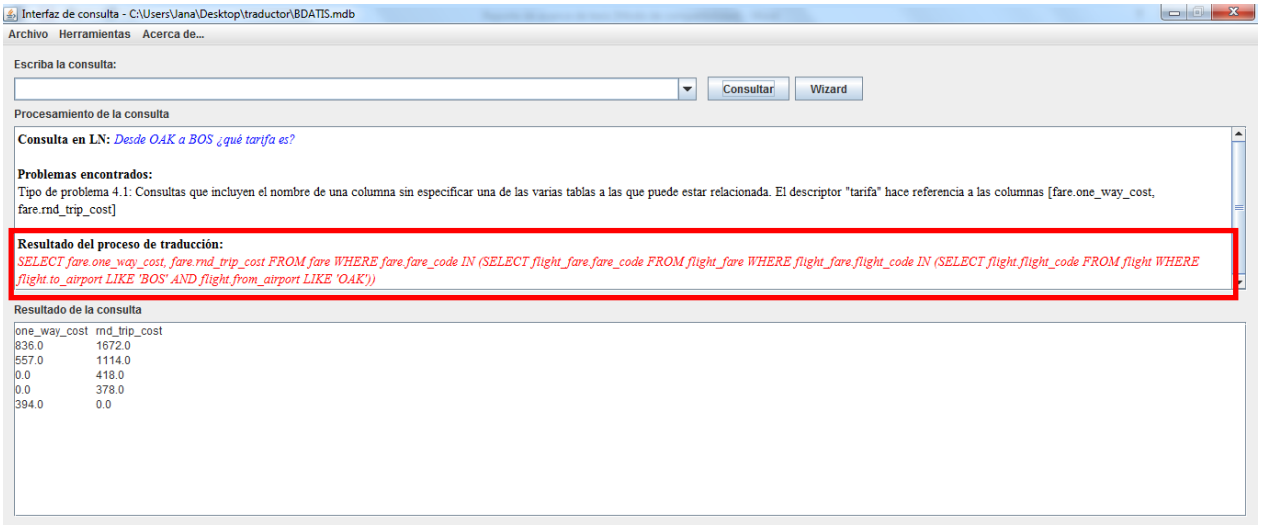

Figura 4.5(b). Respuesta a la consulta 5 (subconsultas).

Como se puede observar, las consultas construidas por la versión anterior de la interfaz tienen reuniones, mientras que las consultas construidas con la nueva versión tienen subconsultas. Sin embargo, los resultados obtenidos por ambas versiones son iguales.

4.3. Pruebas funcionales con consultas complejas

Se realizaron pruebas funcionales con la nueva versión con consultas complejas. En este tipo de consultas su grafo semántico tiene sólo un nodo con etiqueta Select y dos o más nodos con etiqueta Where. El propósito de estas pruebas fue verificar que la nueva versión generara las expresiones en SQL con subconsultas, de tal manera que éstas fueran semánticamente equivalentes a las generadas por la versión anterior. Es conveniente aclarar que estas consultas no involucran funciones de agregación ni agrupamiento. Para tal efecto, se listan enseguida dos consultas de prueba, cuyos resultados se muestran en las Figuras 4.6.a, 4.6.b, hasta 4.7.a, 4.7.b.

- Consulta 6: *Muéstrame el costo de clase Business para el vuelo número 1.*
- Consulta 7: *Dame la tarifa en clase Q desde DFW a ATL.*

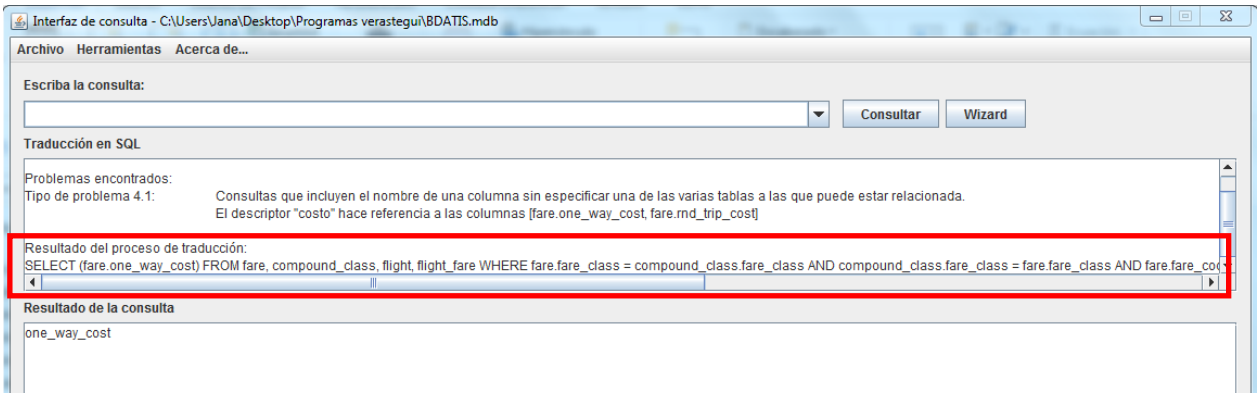

Figura 4.6 (a). Resultado a la consulta 6 (reuniones).

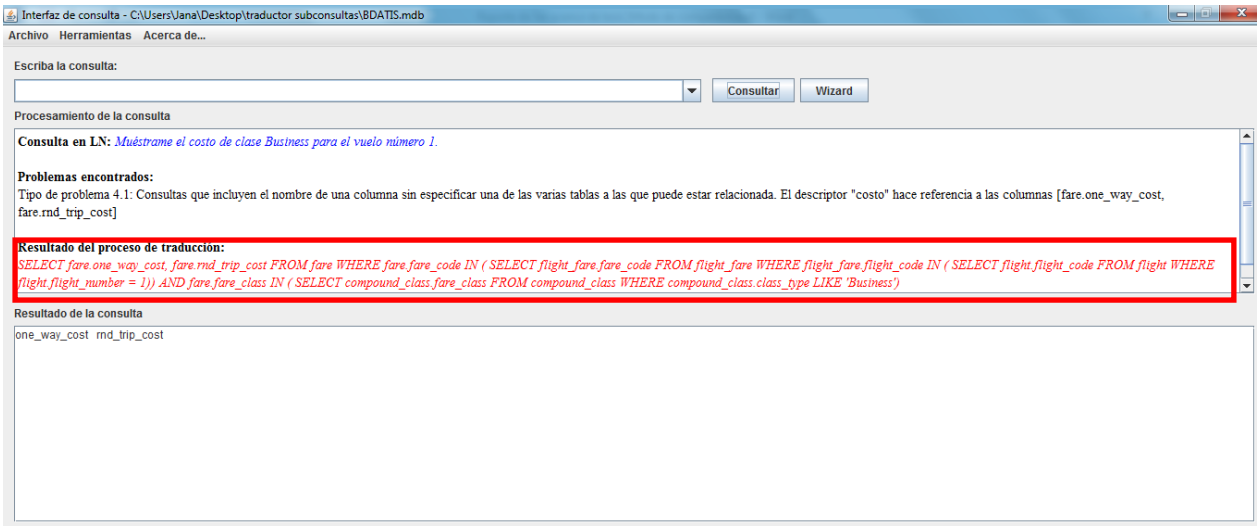

Figura 4.6 (b). Resultado de la consulta 6 (subconsultas).

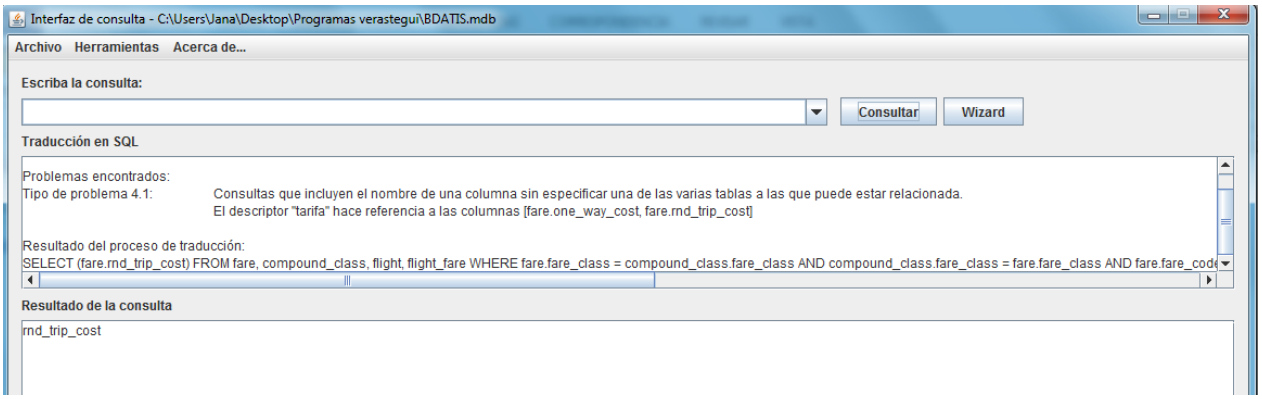

Figura 4.7 (a). Resultado a la consulta 7 (reuniones).

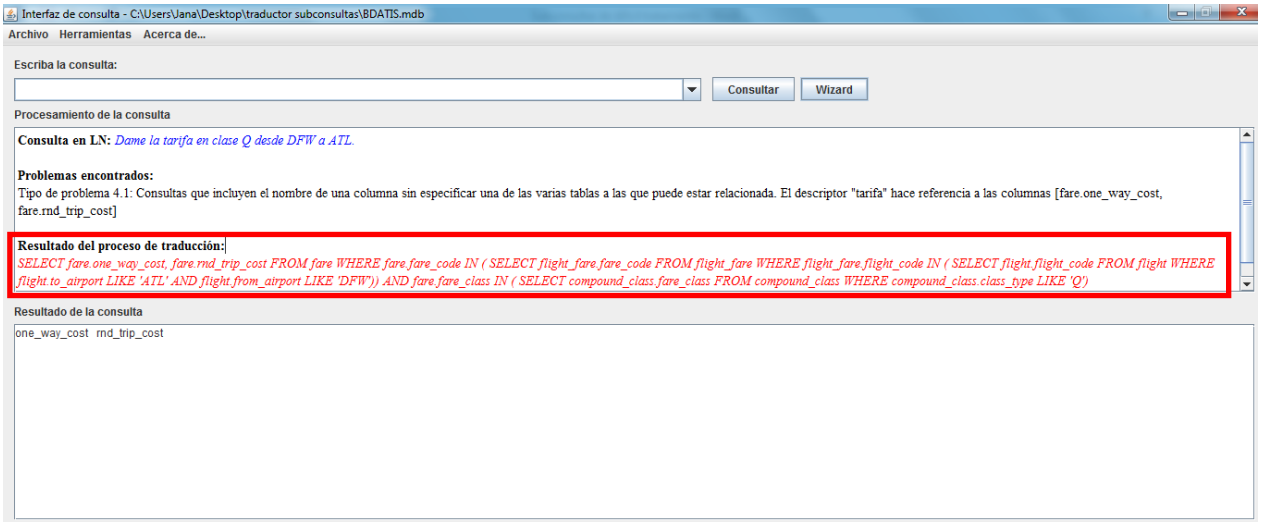

Figura 4.7 (b). Resultado a la consulta 7 (subonsultas).

El resultado obtenido de dichas pruebas demuestra que la interfaz es capaz de construir consultas en SQL usando subconsultas que involucran una tabla en la cláusula Select y dos tablas en la cláusula Where.

Por tal motivo, se concluye que la ILNBD puede componer consultas simples o complejas en SQL haciendo uso de reuniones o subconsultas sin afectar el resultado obtenido de las consultas.

4.4. Pruebas funcionales con consultas que involucran FA, agrupamiento y subconsultas

Se realizaron pruebas funcionales con la nueva versión de la interfaz con el fin de demostrar el correcto funcionamiento de la misma, al introducir consultas que involucren funciones de agregación, agrupamiento, valores de búsqueda que requieren subconsultas con F.A. y subconsultas.

El corpus de consultas utilizado en estas pruebas es el mismo que el utilizado en [Verástegui, 2015]. Dicho corpus de consultas se encuentra en el Apéndice D.

Las pruebas consistieron en ejecutar un conjunto de 100 consultas en la ILNBD, los resultados obtenidos demuestran que la nueva versión de la interfaz tiene un recall de 99% para dicho corpus al igual que la versión anterior.

La consulta número 17 no fue respondida satisfactoriamente por ninguna de las dos versiones, ya que contiene una ambigüedad léxica en la palabra río. La solución a dicha ambigüedad no es objeto de este tema de tesis por lo cual no fue contestada correctamente esta consulta.

Capítulo 5. Conclusiones y trabajos futuros

5.1. Conclusiones

Como resultado de esta tesis se ha obtenido un módulo llamado Generación de consultas en SQL (Figura 1.1), el cual sustituye al módulo Determinación de reuniones implícitas (Figura 2.4). Esta modificación al núcleo de la interfaz permite construir consultas con subconsultas en lugar de reuniones, en los casos donde sea posible.

Aunado a lo anterior, se ha integrado el módulo para tratamiento de funciones de agregación y agrupamiento con subconsultas. Dicha integración permite a la interfaz construir consultas con un mayor nivel de subconsultas que el obtenido en [Verástegui, 2015].

Se realizaron pruebas funcionales con consultas que involucran varias tablas. El resultado de dichas pruebas se presenta en la sección 4.2. En tales pruebas se puede observar que la nueva versión de la interfaz es capaz de construir consultas en SQL con subconsultas cuando la consulta en LN involucra varias tablas y requiere el uso de reuniones. Por lo tanto, se determina como cumplido el punto OG1 mencionado en la sección 1.1.

Además de lo anterior, se identificó que puede darse el caso en el que existan dos tablas diferentes en la cláusula Where y una tabla en la cláusula Select, llamadas consultas complejas. Este caso puede ser procesado con subconsultas en lugar de reuniones. Sin embargo, al realizar el proceso de construcción de las subconsultas en SQL, se notó que el procedimiento sería más complejo que al procesar consultas simples (consultas que involucran una tabla en el Select y una tabla en el Where). Dicho caso pudo ser resuelto por medio de la implementación de un algoritmo recursivo, el cual permite a la interfaz construir las subconsultas a partir del grafo semántico que representa una consulta.

También se realizaron pruebas funcionales con consultas complejas (sección 4.3). Estas pruebas demuestran el funcionamiento correcto de la interfaz al procesar consultas de este tipo.

Por último, se realizaron pruebas funcionales con el corpus empleado en [Verástegui, 2015]. Dichas pruebas se documentan en la sección 4.4, donde puede apreciarse que la ILNBD puede contestar correctamente cualquier consulta que conteste la ILNBD desarrollada en el mencionado trabajo. Además, el algoritmo recursivo implementado y la adaptación de los algoritmos desarrollado en [Verástegui, 2015] permiten que la nueva versión de la interfaz pueda contestar consultas con un mayor número de subconsultas que el que ya realizaba anteriormente. Por lo tanto, el punto OG2 queda cubierto.

5.2. Trabajos futuros

Los trabajos futuros que se pueden derivar del presente proyecto son los siguientes:

- Diseñar e implementar un wizard para palabras no identificadas en la consulta para incluirlas cuando se refieren a funciones de agregación.

- Dar tratamiento a los problemas detectados durante el análisis sistematizado de problemas en consultas, para de esta manera aumentar el desempeño del módulo de tratamiento de funciones de agregación, agrupamiento y subconsultas.

- Con el fin de enriquecer el diccionario de información semántica, un posible trabajo futuro sería la recopilación, organización y almacenamiento de palabras que no fueron incluidas en el diccionario de información semántica y que se refieren a funciones de agregación y agrupamiento.

- Diseñar e implementar un algoritmo para que la ILNBD pueda construir consultas con subconsultas cuando la consulta en LN involucre más de una tabla en la cláusula Select y más nodos en la cláusula Where.

Apéndices.

Apéndice A. Algoritmos para FA y agrupamiento

En esta sección se muestran los algoritmos desarrollados en [Verástegui, 2015]. Dichos algoritmos son usados en la ILNBD para dar tratamiento a las consultas que involucran funciones de agregación, agrupamiento y valores de búsqueda que requieren subconsultas. Estos algoritmos son ejecutados desde una clase llamada *agrega_agrupa_subcons*.

Algoritmo A.1. Método validación mayor que, menor que

```
1  for i=0,…,n-1 do //para cada token Qi de la consulta Q
2   if getLema(Qi)= "mayor" and getLema(Qi+1)= "que"
3    compactarTokens(i,i+1)
4    n ← n - 1
5    setEtiquetaFinal(Qi,">")
6    setMarcado(Qi,true)
7   endif
8   if getLema(Qi)= "menor" and getLema(Qi+1)= "que"
9    compactarTokens(i,i+1)
10   n ← n - 1
11   setEtiquetaFinal(Qi,"<")
12   setMarcado(Qi,true)
13  endif
14 endfor
```

Algoritmo A.2. Método BusquedaFraseEnSID

```
1 BusquedaFraseEnSID(frase, frases_SID) //método para
2 //encontrar frase en las cadenas extraídas del SID
3  for k=1,…,|frases_SID| do //para cada frase en frases extraídas
4   if BusquedaEnString(frase, frases_SID(k,0))= true
5    //frase es subcadena de una frase extraída
6    if frase = frases_SID(k,0)
7     //frase es igual a frase extraída
8     fa ← frases_SID(k,1)
9      return fa
10    else
11     return "+++" //frase contenida en una frase más grande
12    endif
13   endif
14 endfor
```

```
15 //frase no fue encontrada en frases extraídas
16 return "0"
```

Algoritmo A.3. Método para solución de ambigüedad

```
1 for i=0,…,i-1 do //para cada token Qi de la consulta Q
2   if getEtiquetaFinal(Qi)= "COUNT/SUM"
3     Diálogo para solucionar ambigüedad de (COUNT/SUM)
4     setEtiquetaFinal(Qi, opción_escogida_en_diálogo)
5   endif
6 endfor
```

Algoritmo A.4. Método para solución de elipsis en funciones de agregación

```
1 for i=0,…,n-1 do //para cada token Qi de la consulta Q
2   if getTipoFrase(Qi)="Frase Select" and getEtiquetaFinal(Qi)=" "
3     Diálogo para solución de elipsis
4     setEtiquetaFinal(Qi,opción_escogida_en_diálogo)
5   endif
6 endfor
```

Algoritmo A.5. Método para detección de funciones de agregación y agrupamiento.

```
0  boolean flag←false
1  int posición←0 //posición del token que indica GROUP BY
2  string frase //guarda 1 o más tokens de la consulta Q
3  list frases_SID //lista de frases del SID
4  string fa //indica función de agregación
5  for i=0,…,n-1 do //para cada token Qi de la consulta Q
6    if getMarcado(Qi)= false
7      frase ← getLema(Qi)
8      frases_SID ← getFraseDelSID(frase)
9      if |frases_SID|=0 //cero frases extraídas del SID
10       continue
11     else //para cada token Qi de la consulta Q
12     fa ← BusquedaFraseEnSID(frase, frases_SID)
13     endif
14     if fa = "0" //frase no fue encontrada en frases del SID
15       continue
16     endif
17     if fa ∉ {"0","+++"} //frase es igual a frase del SID
18       setEtiquetaFinal(Qi,fa)
```

```
19          setMarcado(Qi,true)
20      if fa ∈{"AVG","COUNT","MIN","MAX","SUM","COUNT/SUM"}
21              setTipoFrase(Qi,"Frase Select")
22      endif
23      if fa="GROUP BY"
24              setTipoFrase(Qi,"GROUP BY")
25              posición←i
26                  flag←true
27      endif
28      continue
29    endif
30    for k=1,…,n do  //frase es subcadena de una frase del SID
31      if getMarcado(Qi+1) = false
32        frase ← frase + " "+ lema(Qi+1)
33        fa ← BusquedaFraseEnSID(frase, frases_SID)
34        if fa = "0" //no fue encontrada en frases del SID
35          i ← i+1 //aumenta el índice de los tokens
36          break
37        endif
38        if fa ∉ {"0","+++"} //frase es igual a frase del SID
39          compactarTokens(i,i+k) //los tokens son compactados
40          n ← n - k //tamaño de la estructura es reducido
41          setEtiquetaFinal(Qi,fa)
42          setMarcado(Qi,true)
43          if fa ∈{"AVG","COUNT","MIN","MAX","SUM",
44          "COUNT/SUM"}
45            setTipoFrase(Qi,"Frase Select")
46          endif
47          if fa="GROUP BY"
48            setTipoFrase(Qi,"GROUP BY")
49                posición←i
50                    flag←true
51          endif
52          break
53        endif
54        continue //frase es subcadena de frase extraída
55      else
56              break
57      endif
58    endfor
```

```
59   endif
60 endfor
```

Algoritmo A.6. Asociación de columna para GROUP BY

```
1 int posición //posición del token que indica GROUP BY
2 if flag ← true
3   for i=posición+1,…,n-1 do //para cada token Qi de la consulta Q
4     if getTipoFrase(Qi)= "Frase Select" and
5     getEtiquetaColumna(Qi)<>" " and getEtiquetaFinal(Qi)<>" "
6       setTipoFrase(Qi,"GROUP BY")
7     endif
8   endfor
9 endif
```

Algoritmo A.7. Validación de patrón.

```
1 boolean flag ← false
2 string auxiliar //almacena la etiqueta final
3 string auxiliar2 //almacena el tipo de dato
4 list temporal //se utiliza para almacenar la columna y tabla
5 if(getEtiquetaFinal(Qn-1)∈{"AVG","COUNT","MIN","MAX","SUM")
6 //FA detectada
7   if(getEtiquetaFinal(Qn-2)<>" ") //identificó una columna
8     auxiliar←getEtiquetaFinal(Qn-2) //almacena la columna
9     temporal←split(auxiliar) //almacena resultado del split
10    auxiliar2←tipoDeDato(auxiliar) //ejecuta tipoDeDato
11    //guarda valor
12  endif
13  if(auxiliar2="entero") //si el dato fue numérico
14    flag←true
15  endif
16  else
17  if(auxiliar2<>"entero" and getEtiquetaFinal(Qn-1)= "COUNT")
18    //no fue numérico pero es la función de agregación "COUNT"
19    flag←true
20  endif
21  if(flag=true) //existe un patrón de subconsulta.
22    método_recopilación_elementos_subconsulta()
23  endif
24endif
```

Algoritmo A.8. Método para recopilar elementos de la subconsulta.

```
1 string fa //almacena la función de agregación de la subconsulta
2 string subcolumna //almacena la columna de la subconsulta
3 string subtabla //almacena la tabla de la subconsulta
4 string resultado //almacena el resultado de la subconsulta
5 int contador←0 //al llegar a 2 detiene ciclo for, se encontró la    6 //columna y FA
7  for i=n-1,…,0 do //para cada token Qi de la consulta Q
8   if getEtiquetaFinal(Qi)∈ {"MAX","MIN","SUM","AVG","COUNT"}
9   //se localizó la función de agregación para la subconsulta
10    fa←getEtiquetaFinal(Qi) //obtiene la función de agregación
11    contador←contador+1
12    setMarcado(Qi,true) //usado en subconsulta
13    setTipoFrase(Qi,"subconsulta")
14   endif
15   if getEtiquetaFinal(Qi)<>" " and getEtiquetaColumna(Qi)<>" "
16   and getTipoFrase(Qi) ="Frase Select" and getMarcado(Qi)= true
17   //se localizó la columna involucrada en la subconsulta
18    subcolumna←getEtiquetaFinal(Qi)
19    contador←contador+1
20    setTipoFrase(Qi,"Frase Where") //cambia para asociar valor
21    //de búsqueda
22   endif
23   if (contador=2) //encontró la columna y FA para subconsultas
24    break //se detiene el ciclo for
25   endif
26 enfor
27 subtabla←split(subcolumna) //obtiene subtabla de la subcolumna
28 resultado←SELECT fa(subcolumna) FROM subtabla //se ejecuta la subconsulta y se
    // almacena su valor en la variable resultado
```

Algoritmo A.9. Asociación de cláusula WHERE.

```
1 for i=0,…,<n-2 do //para cada token Qi de la consulta Q
2   if(getTipoFrase(Qi)="Frase Where" and
3   getEtiquetaColumna(Qi)<>" " and getMarcado(Qi)=true) //localizó
4   //columna que asocia al valor de búsqueda
5     addToken(Qi+1,"resultado_subconsulta")//agrega token resultado
6     //de subconsulta
7     setTipoFrase(Qi,"Frase Where")
8     setEtiquetaFinal(Qi,resultado)//etiqueta final es resultado
9     //de subconsulta, obtenido en el algoritmo 8
10    setMarcado(Qi,true)
11  endif
12endfor //se ejecuta la instrucción en SQL con el valor Where que representa el resultado de la
          //subconsulta
```

Apéndice B. Descripción de la BD ATIS

ATIS (Air Travel Information Service por sus siglas en inglés) es una base de datos relacional (BD) para almacenar información sobre vuelos. Este Apéndice presenta los detalles de las tablas de la BD y un diagrama del esquema de BD.

Para describir la base de datos ATIS, primero se presenta el nombre de las tablas, con la descripción usada para este trabajo, y a continuación, por cada columna en la tabla se muestran el nombre de la columna, el tipo de dato de la columna y por último una descripción de la columna.

Tabla: flight y flight_1 **Descripción**: Vuelo de un aeropuerto a otro

Columna	Tipo	Descripción
flight_code	Numérico	Código de vuelo
flight_days	Texto	Días hábiles
from_airport	Texto	Aeropuerto de origen
to_airport	Texto	Aeropuerto de destino
departure_time	Numérico	Hora de salida
arrrival_time	Numérico	Hora de llegada
airline_code	Texto	Código de aerolínea
flight_number	Numérico	Número de vuelo
class_string	Texto	Código de clase
aircraft_code	Texto	Código de avión

meal_code	Texto	Código de comida
stops	Numérico	Número de escalas
dual_carrier	Texto	Empresa dual
time_elapsed	Numérico	Tiempo de viaje

LLAVE PRIMARIA: flight_code
LLAVE FORÁNEA: flight (aircraft_code) – aircraft (aircraft_code)
flight (airline_code) – airline (airline_code)
flight (from_airport) – airport (airport_code)
flight (to_airport) – airport (airport_code)

Tabla: *airline* y *airline_1* **Descripción:** Aerolínea

Columna	Tipo	Descripción
airline_code	Texto	Código de aerolínea
airline_name	Texto	Nombre de aerolínea
notes	Textoo	Notas

LLAVE PRIMARIA: airline_code
LLAVE FORÁNEA: airline (airline_code) – restrict_carrier (airline_code)

Table: *fare* **Description:** Tarifa

Columna	Tipo	Descripción
fare_code	Texto	Código de tarifa
from_airport	Texto	Aeropuerto de origen
to_airport	Texto	Destination airport (Aeropuerto de destino)
fare_class	Texto	Clase de tarifa
fare_airline	Texto	Tarifa de aerolínea
restrict_code	Texto	Código de restricción
one_way_cost	Numérico	Tarifa de viaje sencillo
rnd_trip_cost	Numérico	Tarifa de viaje redondo

LLAVE PRIMARIA: fare_code
LLAVE FORÁNEA: fare (restrict_code) – restriction (restrict_code)
fare (fare_class) – compound_class (fare_class)

Tabla: *aircraft* **Descripción:** Avión

Columna	Tipo	Descripción

aircraft_code	Texto	Código de avión
aircraft_type	Texto	Tipo de avión
engines	Numérico	Número de motores
category	Texto	Categoría de avión
wide_body	Texto	Fuselaje ancho
wing_span	Numérico	Extensión de alas
length1	Numérico	Tamaño de equipo
weight	Numérico	Peso
capacity	Numérico	Número de asientos
pay_load	Numérico	Capacidad de carga
cruising_speed	Numérico	Velocidad
range_miles	Numérico	Longitud de vuelo
pressurized	Texto	Presurización
LLAVE PRIMARIA:	aircraft_code	

Tabla: *transport* **Descripción:** Servicio de transporte del aeropuerto

Columna	**Tipo**	**Descripción**
transport_code	Texto	Código de transporte
transport_desc	Texto	Descripción de transporte
LLAVE PRIMARIA:	transport_code	

Tabla: *compound_class* **Descripción:** Clase compuesta

Columna	**Tipo**	**Descripción**
fare_class	Texto	Código de clase de tarifa
base_class	Texto	Clase base
class_type	Texto	Tipo de clase
premium	Texto	Es clase premium?
economy	Texto	Es clase económica?
discounted	Texto	Tiene descuento?
night	Texto	Es vuelo nocturno?
season_fare	Texto	Tipo de tarifa de temporada
class_days	Texto	Días en que se aplica la clase

LLAVE PRIMARIA:	fare_class
LLAVE FORÁNEA: (class_code)	compound_class (base_class) – class_of_service

Tabla: *ground_service* **Descripción:** Servicio terrestre

Columna	**Tipo**	**Descripción**
city_code	Texto	Código de ciudad
airport_code	Texto	Código de aeropuerto
transport_code	Texto	Código de transporte
ground_fare	Numérico	Tarifa terrestre
LLAVE PRIMARIA:		city_code, airport_code, transport_code
LLAVE FORÁNEA: (transport_code)		ground_service (transport_code) – transport

Tabla: *food_service* **Descripción:** Servicio de comida

Columna	**Tipo**	**Descripción**
meal_code	Texto	Código de comida
meal_number	Numérico	Número de comida
meal_class	Texto	Clase de comida
meal_description	Texto	Descripción de comida
LLAVE FORÁNEA:		food_service (meal_code) – flight (meal_code)

Tabla: *restriction* **Descripción:** Restricción

Columna	**Tipo**	**Descripción**
restrict_code	Texto	Código de restricción
application	Texto	Aplicación
no_discounts	Texto	Descuentos no aplicables
reserve_ticket	Numérico	Boletos en reserva
stopovers	Texto	Escalas
return_min	Numérico	Mínimo de permanencia
return_max	Numérico	Máximo de permanencia
LLAVE PRIMARIA:		restrict_code

Tabla: *city* **Description:** Ciudad

Columna	Tipo	Descripción
city_code	Texto	Código de ciudad
city_name	Texto	Nombre de la ciudad
state_code	Texto	Código del estado
time_zone_code	Texto	Código de zona horaria
LLAVE PRIMARIA:	city_code	
LLAVE FORÁNEA:	city (state_code) – state (state_code)	

Tabla: *class_of_service* **Descripción:** Clase de servicio

Columna	Tipo	Descripción
class_code	Texto	Código de clase de servicio
rank	Numérico	Rango
class_description	Texto	Descripción de clase de servicio
LLAVE PRIMARIA:	class_code	

Tabla: *airport_service* **Descripción:** Servicio de aeropuerto

Columna	Typo	Descripción
city_code	Texto	Código de ciudad
airport_code	Texto	Código de aeropuerto
miles_distant	Numérico	Distancia en millas
direction	Texto	Dirección
minutes_distant	Numérico	Distancia en minutos
LLAVE PRIMARIA:	city_code, airport_code	
LLAVE FORÁNEA:	airport_service (airport_code) – airport (airport_code) airport_service (airport_code) – ground_service (airport_code) airport_service (city_code) – ground_service (airport_service) airport_service (city_code) – city (city_code)	

Tabla: *fconnection* **Descripción:** Conexión de vuelo

Column	Type	Description
connect_code	Numérico	Código de conexión
from_airport	Texto	Aeropuerto de origen
to_airport	Texto	Aeropuerto de destino

departure_time	Numérico	Hora de salida
arrival_time	Numérico	Hora de llegada
flight_days	Texto	Días de vuelo
stops	Numérico	Escalas
connections	Numérico	Conexiones

LLAVE PRIMARIA: connect_code
LLAVE FORÁNEA: fconnection (to_airport) – airport (airport_code)
fconnection (from_airport) – airport (airport_code)

Tabla: *connect_leg* **Descripción:** Segmento de conexión

Columna	Tipo	Descripción
connect_code	Numérico	Código de conexión
leg_number	Numérico	Número de segmento
flight_code	Numérico	Código de vuelo

LLAVE FORÁNEA: connect_leg (connect_code) – fconnection (connect_code)
connect_leg (flight_code) – flight (flight_code)

Table: *flight_day* **Descripción:** Días de vuelo

Column	Type	Description
day_mask	Texto	Máscara de días
day_code	Numérico	Código de día

LLAVE PRIMARIA: day_mask, day_code
LLAVE FORÁNEA: flight_day (day_code) – day_name (day_code)

Table: *day_name* **Description:** Días

Columna	Tipo	Descripción
day_code	Numérico	Código de día
day_name	Texto	Nombre de día

LLAVE PRIMARIA: day_code

Table: state **Descripción**: Estado

Columna	Tipo	Descripción

state_code	Texto	Código de estado
state_name	Texto	Nombre de estado
country_name	Texto	Nombre de país
LLAVE PRIMARIA:		state_code

Tabla: *dual_carrier* **Descripción:** Empresas dual

Columna	**Tipo**	**Descripción**
main_airline	Texto	Código de aerolínea principal
dual_airline	Texto	Código de empresa dual
low_flight	Numérico	Vuelo económico
high_flight	Numérico	Vuelo costoso
fconnection_name	Texto	Nombre de conexión
LLAVE FORÁNEA:		dual_carrier (dual_airline) – airline (airline_code) dual_carrier (main_airline) – airline (airline_code)

Tabla: *time_zone* **Descripción:** Zona horaria

Columna	**Tipo**	**Descripción**
time_zone_code	Texto	Código de zona horaria
time_zone_name	Texto	Nombre de zona horaria
LLAVE FORÁNEA:		time_zone (time_zone_code) – city (time_zone_code)

Tabla: *stop1* **Descripción:** Escalas

Columna	**Tipo**	**Descripción**
flight_code	Numérico	Código de vuelo
stop_number	Numérico	Número de escala
stop_flight	Numérico	Vuelo con escalas
LLAVE FORÁNEA:		stop1 (flight_code) – flight (flight_code) stop1 (stop_flight) – flight (flight_code)

Tabla: *flight_fare* **Descripción:** Tarifas de los vuelos

Columna	**Tipo**	**Descripción**
flight_code	Numérico	Código de vuelo
fare_code	Texto	Código de tarifa

LLAVE FORÁNEA: flight_fare (fare_code) – fare (fare_code) flight_fare (flight_code) – flight (flight_code)

Tabla: *restrict_carrier* **Descripción:** Restricciones de aerolinea

Columna	**Tipo**	**Descripción**
restrict_code	Texto	Código de restricción
airline_code	Texto	Código de aerolínea
LLAVE FORÁNEA: restrict_carrier (restrict_code) – restriction(restrict_code)		

Tabla: *restrict_class* **Descripción:** Restricción de clase

Column	**Type**	**Description**
restrict_code	Texto	Código de restricción
ex_fare_class	Texto	Tarifa de clase
LLAVE FORÁNEA: restrict_class (restrict_code) – restriction(restrict_code)		

El esquema de BD de ATIS se muestra en la Figura B.1. Esta base de datos cuenta con un total de 27 tablas y 123 columnas.

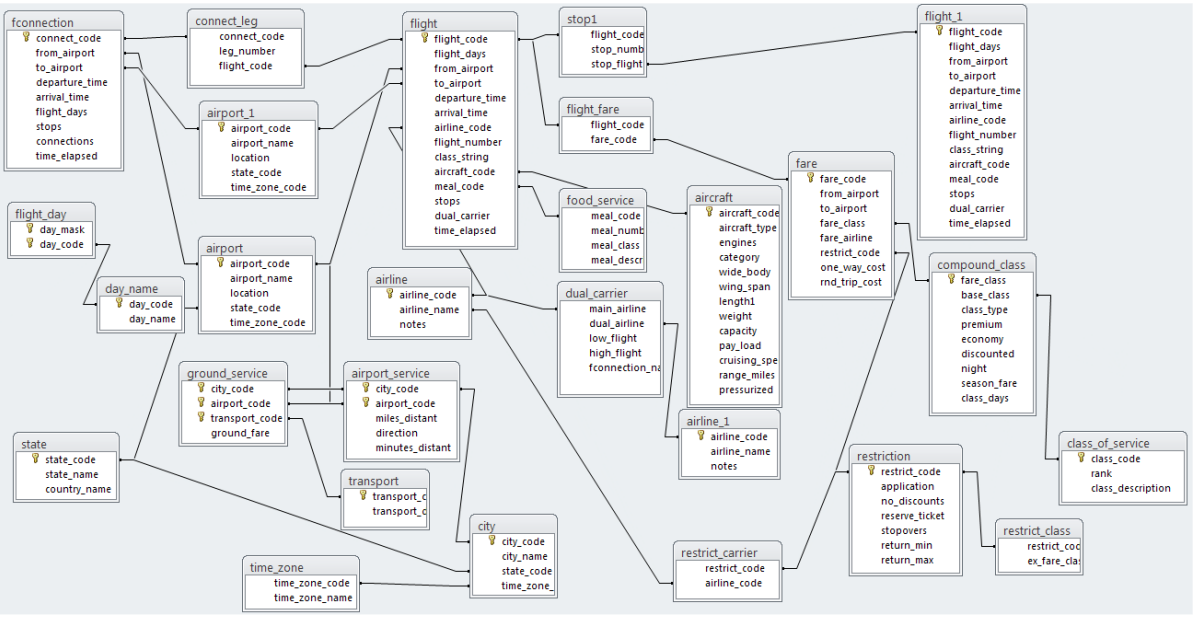

Figura B.1. Esquema de la base de datos ATIS.

Apéndice C. Descripción de la BD Geobase

Geobase es una base de datos para almacenar información geográfica de los Estados Unidos de Norte América, y es usada por muchas ILNBDs como BD de prueba desde los 90s. La razón por la que esta BD es usada como evaluación es porque se encuentra disponible y tiene una estructura simple. Sin embargo, muchos investigadores la han adaptado a un modelo de base de datos relacional para usarla como medio de evaluación de ILNBDs que trabajan con BDs relacionales.

Para describir la base de datos Geobase, primero se presenta el nombre de las tablas, con la descripción usada para este trabajo, y a continuación, por cada columna en la tabla se muestran el nombre de la columna, el tipo de dato de la columna y por último una descripción de la columna.

Tabla: *City* **Descripción:** Ciudad

Columna	**Tipo**	**Descripción**
state_abbreviation	Texto corto	abreviatura de estado
city_name	Texto corto	ciudad
state_name	Texto corto	nombre de estado
population	Número	población
LLAVE FORÁNEA: city (state_abbreviation) – state (abbreviation)		

Tabla: *HighLow* **Descripción:** Puntos Alto y Bajo

Columna	**Tipo**	**Descripción**
state_abbreviation	Texto corto	abreviatura de estado
state_name	Texto corto	nombre de estado
highest_point	Texto corto	punto más alto
highest_elevation	Número	elevación más alta
lowest_point	Texto corto	punto más bajo
lowest_elevation	Número	elevación más baja
LLAVE FORÁNEA: HighLow (state_abbreviation) – state (abbreviation)		

Tabla: *Border* **Descripción:** Frontera

Columna	**Tipo**	**Descripción**
state_abbreviation	Texto corto	abreviatura de estado
state_name	Texto corto	nombre de estado

bordering_state	Texto corto	colindancia
LLAVE FORÁNEA: Border (state_abbreviation) – state (abbreviation)		

Tabla: *State* **Descripción:** Estado

Columna	**Tipo**	**Descripción**
abbreviation	Texto corto	abreviatura
state_name	Texto corto	estado
capital	Texto corto	capital
population	Número	población
area	Número	área
state_number	Número	número de estado
city1	Texto corto	ciudad uno
city2	Texto corto	ciudad dos
city3	Texto corto	ciudad tres
city4	Texto corto	ciudad cuatro
LLAVE FORÁNEA: State (abbreviation) – RoadState (state_abbreviation)		

Tabla: *Mountain* **Descripción:** Montaña

Columna	**Tipo**	**Descripción**
mountain_id	Número	identificador de montaña
state_abbreviation	Texto corto	abreviatura de estado
state_name	Texto corto	nombre de estado
mountain_name	Texto corto	montaña
height	Número	altura
LLAVE FORÁNEA: Mountain (state_abbreviation) – state (abbreviation)		

Tabla: *RiverState* **Descripción:** Estado del río

Columna	**Tipo**	**Descripción**
river_id	Número	identificador de río
state_abbreviation	Texto corto	abreviatura de estado
LLAVE FORÁNEA: RiverState (state_abbreviation) – state (abbreviation)		

Tabla: *LakeState* **Descripción:** Estado del lago

Columna	Tipo	Descripción
lake_id	Número	identificador de lago
state_abbreviation	Texto corto	abreviatura de estado
LLAVE FORÁNEA: LakeState (state_abbreviation) – state (abbreviation)		

Tabla: *RoadState* **Descripción:** Estado de la carretera

Columna	Tipo	Descripción
number	Número	identificador de carretera
state_abbreviation	Texto corto	abreviatura de estado
LLAVE FORÁNEA: RoadState (state_abbreviation) – state (abbreviation)		

Tabla: *River* **Descripción:** Río

Columna	Tipo	Descripción
river_id	Número	identificador de río
river_name	Texto corto	río
length	Número	longitud
LLAVE FORÁNEA: River (river_id) – RiverState (river_id)		

Tabla: *Lake* **Descripción:** Lago

Columna	Tipo	Descripción
lake_id	Número	abreviatura de estado
lake_name	Texto corto	lago
area	Número	área
LLAVE FORÁNEA: Lake (lake_id) – LakeState (lake_id)		

Tabla: *Road* **Descripción:** Carretera

Columna	Tipo	Descripción
number	Número	carretera
LLAVE FORÁNEA: Road (number) – RoadState (number)		

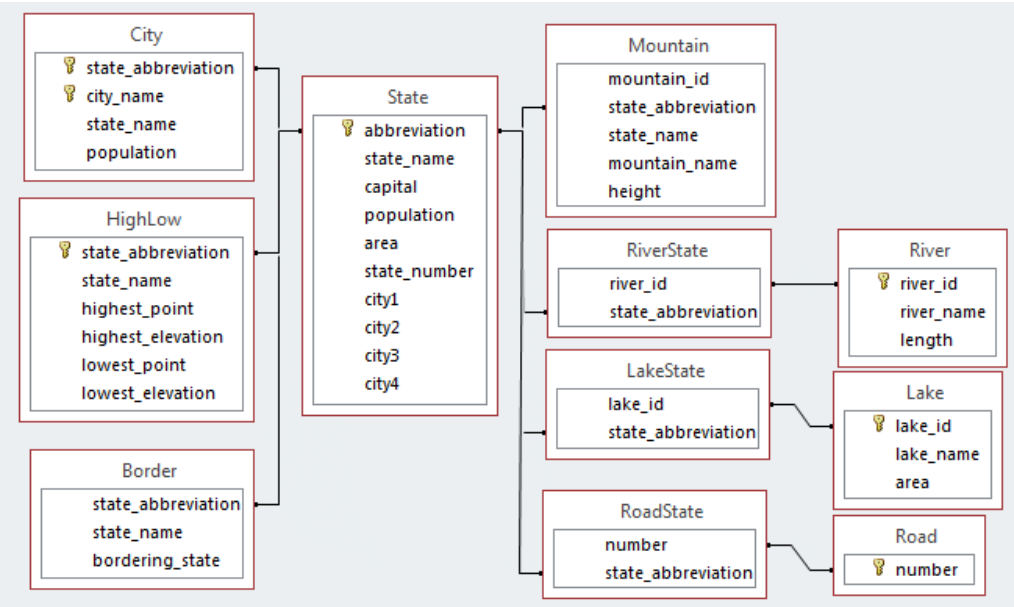

Figura C.1. Esquema de la base de datos Geobase.

Apéndice D. Corpus de consultas de pruebas funcionales.

En este Apéndice se muestra una tabla que contiene 100 consultas usadas en [Verástegui, 2015]. De igual forma, en este proyecto de tesis se utilizó dicho corpus para verificar el buen funcionamiento de la ILNBD.

Tabla D.1. Corpus de consultas de ATIS y Geobase

No.	Consulta	BD	Problemas Involucrados	Consultas contestadas con la nueva interfaz	Consultas contestadas con la interfaz de Verástegui
1	Cuántos ríos corren a través del estado de Texas	Geobase	FA, Ambigüedad COUNT/SUM	✓	✓
2	Combina la población de las ciudades del estado de Alabama	Geobase	FA, Ambigüedad en columna	✓	✓
3	Suma la altura de las montañas del estado de Washington	Geobase	FA	✓	✓
4	Cuál es la media de la longitud de los ríos en el estado de Texas	Geobase	FA	✓	✓
5	Cuál es el promedio de la altura de las montañas del estado de Alabama	Geobase	FA	✓	✓

6	Cuál es la longitud del río más largo del estado de Texas	Geobase	FA	✓	✓
7	Cuál es el área del estado más grande	Geobase	FA	✓	✓
8	Cuál es la longitud del río más pequeño en el estado de Oregon	Geobase	FA	✓	✓
9	Cuál es el área del estado más pequeño	Geobase	FA	✓	✓
10	Cuál es la longitud del río más grande del estado de Oregon	Geobase	FA	✓	✓
11	Dame la altura de la montaña más grande del estado de Colorado	Geobase	FA	✓	✓
12	Cuántas ciudades hay en el estado de California	Geobase	FA, Ambigüedad COUNT/SUM	✓	✓
13	Cuántos estados tienen colindancia con el estado de Florida	Geobase	FA, Ambigüedad COUNT/SUM	✓	✓
14	Dame la altura de la montaña más grande del estado de Alaska	Geobase	FA	✓	✓
15	Cuántos lagos hay en el estado de California	Geobase	FA, Ambigüedad COUNT/SUM	✓	✓
16	Suma las montañas del estado de Washington	Geobase	FA, Elipsis en FA	✓	✓
17	Cuál es el río más grande del estado de Oregon	Geobase	FA, Elipsis en FA, ambigüedad en río	✗	✗
18	Cuál es el estado más pequeño	Geobase	FA, Elipsis en FA	✓	✓
19	Cuál es la media de los ríos en el estado de Texas	Geobase	FA, Elipsis en FA	✓	✓
20	Cuál es el promedio de las montañas del estado de Colorado	Geobase	FA, Elipsis en FA	✓	✓
21	Cuál es el promedio de la longitud de los ríos del estado de Minnesota	Geobase	FA	✓	✓
22	Cuántas ciudades hay en el estado de Montana	Geobase	FA, Ambigüedad COUNT/SUM	✓	✓

23	Cuántos lagos hay en el estado de New York	Geobase	FA, Ambigüedad COUNT/SUM	✓	✓
24	Suma la longitud de los ríos del estado de Iowa	Geobase	FA	✓	✓
25	Cuántos ríos tiene el estado de North Dakota	Geobase	FA, Ambigüedad COUNT/SUM	✓	✓
26	Cuál es la ciudad más grande	Geobase	FA, Elipsis en FA	✓	✓
27	Cuál es el lago más pequeño	Geobase	FA, Elipsis en FA	✓	✓
28	Cuál es el lago más grande	Geobase	FA, Elipsis en FA	✓	✓
29	Cuál es la ciudad más pequeña	Geobase	FA, Elipsis en FA	✓	✓
30	Suma las montañas del estado de Alaska	Geobase	FA, Elipsis en FA	✓	✓
31	Cuál es la población de la ciudad más pequeña del estado de Texas	Geobase	FA, Ambigüedad en columna	✓	✓
32	Total de ciudades por estado	Geobase	Agrupamiento, Ambigüedad COUNT/SUM	✓	✓
33	Cuántas montañas hay por estado	Geobase	Agrupamiento, Ambigüedad COUNT/SUM	✓	✓
34	Total de ciudades por población	Geobase	Agrupamiento, Ambigüedad COUNT/SUM	✓	✓
35	Total de puntos más altos por estado	Geobase	Agrupamiento, Ambigüedad COUNT/SUM	✓	✓
36	Total de puntos más bajos por estado	Geobase	Agrupamiento, Ambigüedad COUNT/SUM	✓	✓
37	Cuántos lagos hay por área	Geobase	Agrupamiento, Ambigüedad COUNT/SUM	✓	✓
38	Suma la población de las ciudades por estado	Geobase	Agrupamiento	✓	✓
39	Cuántos ríos hay por longitud	Geobase	Agrupamiento, Ambigüedad COUNT/SUM	✓	✓
40	Cuántas ciudades hay por población por estado	Geobase	Agrupamiento, Ambigüedad COUNT/SUM, Ambigüedad en columna, Doble agrupamiento	✓	✓
41	Cuántas montañas	Geobase	Agrupamiento,	✓	✓

	hay por estado por altura		Ambigüedad COUNT/SUM, Doble agrupamiento		
42	Cuántas ciudades hay por estado	Geobase	Agrupamiento, Ambigüedad COUNT/SUM	✓	✓
43	Cuántas montañas hay por altura	Geobase	Agrupamiento, Ambigüedad COUNT/SUM	✓	✓
44	Cuántas ciudades hay por población	Geobase	Agrupamiento, Ambigüedad COUNT/SUM	✓	✓
45	Cuántos ríos por longitud	Geobase	Agrupamiento, Ambigüedad COUNT/SUM	✓	✓
46	Cuántos estados hay por población	Geobase	Agrupamiento, Ambigüedad COUNT/SUM	✓	✓
47	Cuántos estados hay por población por capital	Geobase	Agrupamiento, Ambigüedad COUNT/SUM, Doble agrupamiento	✓	✓
48	Cuántos estados hay por población por área	Geobase	Agrupamiento, Ambigüedad COUNT/SUM, Doble agrupamiento	✓	✓
49	Dame el estado con la población más grande	Geobase	Subconsulta	✓	✓
50	Cuál es la capital con la población más grande	Geobase	Subconsulta	✓	✓
51	Dame el estado con el área más pequeña	Geobase	Subconsulta	✓	✓
52	Cuál es la montaña con la altura más pequeña	Geobase	Subconsulta	✓	✓
53	Cuál es el estado con la población más pequeña	Geobase	Subconsulta	✓	✓
54	Dame la ciudad con la población más pequeña	Geobase	Subconsulta	✓	✓
55	Cuál es el río con la longitud más grande	Geobase	Subconsulta	✓	✓
56	Cuál es la ciudad con la población más grande	Geobase	Subconsulta	✓	✓
57	Cuál es la montaña con la altura más	Geobase	Subconsulta	✓	✓

	grande				
58	Cuál es el estado con el área más grande	Geobase	Subconsulta	✓	✓
59	Cuántos vuelos hay con origen en ATL y destino en BOS	ATIS	FA, Ambigüedad COUNT/SUM	✓	✓
60	Cuántas aerolíneas hay	ATIS	FA, Ambigüedad COUNT/SUM	✓	✓
61	Suma el costo del vuelo 148	ATIS	FA, Ambigüedad en columna	✓	✓
62	Suma el número de escalas del vuelo 106	ATIS	FA, Ambigüedad en columna	✓	✓
63	Cuál es la tarifa más cara con origen en ATL y destino en BOS	ATIS	FA, Ambigüedad en columna	✓	✓
64	Cuál es la más alta velocidad de todos los tipos de aeronave	ATIS	FA	✓	✓
65	Cuál es la tarifa más barata con origen en ATL y destino en BOS	ATIS	FA, Ambigüedad en columna	✓	✓
66	Cuál es la más baja velocidad de todos los tipos de aeronave	ATIS	FA	✓	✓
67	Cuál es la media de las tarifas con origen en ATL y destino en BOS	ATIS	FA, Ambigüedad en columna	✓	✓
68	Promedio de costo del vuelo 148	ATIS	FA, Ambigüedad en columna	✓	✓
69	Cuántos códigos de transporte hay	ATIS	FA, Ambigüedad COUNT/SUM, Ambigüedad de columna	✓	✓
70	Suma el tiempo de viaje del vuelo 148	ATIS	FA, Ambigüedad en columna	✓	✓
71	Cuántos aeropuertos hay	ATIS	FA, Ambigüedad COUNT/SUM	✓	✓
72	Cuántos tipos de aeronave hay	ATIS	FA, Ambigüedad COUNT/SUM	✓	✓
73	Cuántos códigos de aerolínea hay con origen en ATL y destino en BOS	ATIS	FA, Ambigüedad COUNT/SUM, Ambigüedad en columna	✓	✓
74	Promedio del peso del tipo de aeronave BOEING 737-300	ATIS	FA	✓	✓
75	Cuántas aerolíneas hay por notas	ATIS	Agrupamiento, Ambigüedad	✓	✓

			COUNT/SUM		
76	Cuántas velocidades hay por tipo de aeronave	ATIS	Agrupamiento, Ambigüedad COUNT/SUM	✓	✓
77	Cuántos tipos de aeronave hay por peso	ATIS	Agrupamiento, Ambigüedad COUNT/SUM, Doble agrupamiento	✓	✓
78	Cuántos aeropuertos hay por ubicación	ATIS	Agrupamiento, Ambigüedad COUNT/SUM	✓	✓
79	Cuántos tipos de aeronave hay por categoría de aeroplano	ATIS	Agrupamiento, Ambigüedad COUNT/SUM	✓	✓
80	Cuántos tipos de aeronave hay por número de motores	ATIS	Agrupamiento, Ambigüedad COUNT/SUM	✓	✓
81	Cuántos tipos de aeronave hay por peso por velocidad	ATIS	Agrupamiento, Ambigüedad COUNT/SUM	✓	✓
82	Cuántas escalas hay por conexiones	ATIS	Agrupamiento, Ambigüedad COUNT/SUM	✓	✓
83	Cuántos nombre de estado hay por nombre de país	ATIS	Agrupamiento, Ambigüedad COUNT/SUM	✓	✓
84	Cuántas categorías de aeroplano hay por longitud de vuelo por número de motores	ATIS	Agrupamiento, Ambigüedad COUNT/SUM, Doble agrupamiento	✓	✓
85	Cuántos vuelos hay por origen por destino	ATIS	Agrupamiento, Ambigüedad COUNT/SUM, Doble agrupamiento	✓	✓
86	Cuántos código de aerolínea hay por aerolínea por notas	ATIS	Agrupamiento, Ambigüedad COUNT/SUM, Ambigüedad en columna, Doble agrupamiento	✓	✓
87	Cuántos días hay por origen por destino	ATIS	Agrupamiento, Ambigüedad COUNT/SUM, Doble agrupamiento	✓	✓
88	Cuántos tipos de aeronave hay por peso por número de	ATIS	Agrupamiento, Ambigüedad COUNT/SUM,	✓	✓

	asientos		Doble agrupamiento		
89	Cuántas llegadas hay por origen por código de aerolínea	ATIS	Agrupamiento, Ambigüedad COUNT/SUM, Ambigüedad en columna, Doble agrupamiento	✓	✓
90	Cuenta los vuelos por código de aerolínea	ATIS	Agrupamiento, Ambigüedad COUNT/SUM	✓	✓
91	Cuál es el tipo de aeronave con el peso más grande	ATIS	Subconsulta	✓	✓
92	Cuál es la tarifa de aerolínea con tarifa de viaje redondo más cara	ATIS	Subconsulta	✓	✓
93	Cuál es la tarifa de aerolínea con tarifa de viaje sencillo más barata	ATIS	Subconsulta	✓	✓
94	Cuál es la categoría de aeroplano con el número de asientos más grande	ATIS	Subconsulta	✓	✓
95	Cuál es el tamaño de equipo con la carga más grande	ATIS	Subconsulta	✓	✓
96	Cuál es el vuelo con la salida más temprano	ATIS	Subconsulta	✓	✓
97	Cuál es el tipo de aeronave con la longitud de vuelo más grande	ATIS	Subconsulta	✓	✓
98	Cuál es el tipo de aeronave con el número de motores más alta	ATIS	Subconsulta	✓	✓
99	Cuál es el tipo de aeronave con el peso más pequeño	ATIS	Subconsulta	✓	✓
100	Cuál es el vuelo con la llegada más tarde	ATIS	Subconsulta	✓	✓

Referencias

[Aguirre, 2014] M. A. Aguirre, *Modelo Semánticamente Enriquecido de Bases de Datos para su Explotación por Interfaces de Lenguaje Natural*, tesis de doctorado, División de Estudios de Posgrado e Investigación, Instituto Tecnológico de Ciudad Madero, Cd. Madero, Tamps., México, 2014.

[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. Ritchie, P. Thanisch, "Natural Language Interface to Database: An Introduction", Journal of Natural Language Engineering, pp. 29-81, 1995.

[Bautista, 2014] A. Bautista, *Traducción de consultas de Lenguaje Natural Español a SQL que involucran agrupamiento*, tesis de maestría, Depto. de Posgrado e Investigación, Instituto Tecnológico de Cd. Madero, Cd. Madero,Tamps., México, 2014.

[Conlon, 2004] S. Conlon, J. Conlon, T. James, The economics of natural language interfaces: natural language processing technology as a scarce resource, *Decision Support Systems*, Vol. 38, No. 1, pp. 141-159, 2004.

[Date, 2008] C. J. Date, *The Relational Database Dictionary*, Apress, 2008.

[ELF, 2009] "ELF Software Documentation Series", http://www.elfsoft.com/help/accelf/Analyze.htm, 2009.

[Esquivel, 2013] I. Esquivel, R. Córdoba, D. González y E. López, "SNL2SQL: Conversión de consultas en SQL al idioma Español", *Congreso Internacional de Investigación*, México, 2013.

[Fernández, 2006] V. Fernández, *Desarrollo de sistemas de información: una metodología basada en el modelado*. Ediciones UPC, 2006.

[Green, 1961] B. Green, A. Wolf, C. Chomsky, K. Laughery, "Baseball: an automatic question-answerer", *Proc. western joint IRE-AIEE-ACM computer conference*, pp. 219–224, 1961.

[Kroenke, 2003] D. Kroenke, *Procesamiento de Bases de Datos*, 8va. Edición, Pearson Educación, 2003.

[Liddy 2001] D. Liddy, *Natural Language Processing for Information Retrieval & Knowledge Discovery*, School of Information Studies, Syracuse University, 2001.

[Mellema, 2009] G. Mellema, http://www.oxfordreference.com/views/ENTRY.html?subview=Main&entry=t116.e929, The Oxford Companion to Philosophy, 2009

[Minock, 2010] M. Minock, "C-phrase: A system for building robust natural language interfaces to databases*", Data & Knowledge Engineering*, Vol. 69, pp. 290–302, 2010.

[Owda, 2007] M. Owda, Z. Bandar y K. Crockett, "Conversation-Based Natural Language Interface to Relational Databases", The Intelligent Systems Group, Department of Computing and Mathematics, The Manchester Metropolitan University, Chester Street, Manchester, M1 5GD, UK, *IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology - Workshops*, 2007.

[RAE, 2017] http://dle.rae.es/?id=N7BnIFO, Real Academia Española, revisado el 14/03/2017, 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.

[Verástegui, 2015] A. A. Verástegui, *Implementación de un Traductor de Consultas de Lenguaje Natural a SQL que Involucran Funciones de Agregación, Agrupamiento y Subconsultas,* tesis de maestría, División de Estudios de Posgrado e Investigación, Instituto Tecnológico de Ciudad Madero, Cd. Madero, Tamps., México, 2015.

[Woods, 1972] W. Woods, R. Kaplan, B. Nash-Webber, "The Lunar Sciences Natural Language Information System: Final Report", BBN Report 2378, Bolt Beranek and Newman Inc., Cambridge, Mass., 1972.