TECNOLÓGICO NACIONAL DE MÉXICO
Instituto Tecnológico de Ciudad Madero

INSTITUTO TECNOLÓGICO DE CIUDAD MADERO
DIVISIÓN DE ESTUDIOS DE POSGRADO E INVESTIGACIÓN

IMPLEMENTACIÓN DE UN TRADUCTOR DE CONSULTAS DE LENGUAJE NATURAL A SQL QUE INVOLUCRAN FUNCIONES DE AGREGACIÓN, AGRUPAMIENTO Y SUBCONSULTAS

OPCIÓN I
TESIS PROFESIONAL

Que para obtener el título de:
Maestro en Ciencias de la Computación

Presenta:
Andrés Adolfo Verástegui Ollervides
G13071952

Asesor:
Dr. José Antonio Martínez Flores
Dr. Rodolfo Abraham Pazos Rangel

Cd. Madero, Tamps. Octubre 2015

SEP
SECRETARÍA DE EDUCACIÓN PÚBLICA

TECNOLÓGICO NACIONAL DE MÉXICO
Instituto Tecnológico de Ciudad Madero

"2015, Año del Generalísimo José María Morelos y Pavón"

Cd. Madero, Tamps; a 21 de Septiembre de 2015.

OFICIO No.: U5.177/15
AREA: DIVISIÓN DE ESTUDIOS
DE POSGRADO E INVESTIGACIÓN
ASUNTO: AUTORIZACIÓN DE IMPRESIÓN DE TESIS

ING. ANDRÉS ADOLFO VERÁSTEGUI OLLERVIDES
NO. DE CONTROL G13071952
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.	JUAN JAVIER GONZÁLEZ BARBOSA
SECRETARIO :	DR.	ARTURO HERNÁNDEZ RAMÍREZ
VOCAL :	DR.	JOSÉ ANTONIO MARTÍNEZ FLORES
SUPLENTE	DR.	RODOLFO ABRAHAM PAZOS RANGEL
DIRECTOR DE TESIS :	DR.	JOSÉ ANTONIO MARTÍNEZ FLORES
CO-DIRECTOR DE TESIS:	DR.	RODOLFO ABRAHAM PAZOS RANGEL

Se acordó autorizar la impresión de su tesis titulada:

"IMPLEMENTACIÓN DE UN TRADUCTOR DE CONSULTAS DE LENGUAJE NATURAL A SQL QUE INVOLUCRAN FUNCIONES DE AGREGACIÓN, AGRUPAMIENTO Y SUBCONSULTAS"

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"®

Ma. Yolanda Chávez Cinco

M. P. MARÍA YOLANDA CHÁVEZ CINCO
JEFA DE LA DIVISIÓN

ESTADOS UNIDOS MEXICANOS
S.E.P.
DIVISION DE ESTUDIOS
DE POSGRADO E
INVESTIGACION
I T C M

c.c.p.- Archivo
Minuta

MYCHC 'NDCO ' jar

CONTENIDO

Lista de tablas

Lista de figuras

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.

Andrés Adolfo Verástegui Ollervides

AGRADECIMIENTOS

A los doctores integrantes de mi comité tutorial:

Dr. Arturo Hernández quien durante sus clases fue ejemplo del compromiso y entusiasmo que se debe tener hacia la profesión.

Dr. José Martínez por su confianza, colaboración y disposición que permitieron concluir este proyecto.

Dr. Javier González por sus consejos, ánimo y guía como mi tutor.

Dr. Rodolfo Pazos ya que gracias a su conocimiento, apoyo, visión y paciencia se logró materializar este proyecto.

A los alumnos de doctorado y maestría con los que conviví, en especial a mis compañeros de generación Javier, Daniel, Jorge y Manuel por su amistad, horas de convivencia y risas que compartimos durante la maestría. A Rafael Ortega por su ayuda durante mis años de estudiante y por su amistad.

Finalmente a mi familia, por su apoyo incondicional.

RESUMEN

Las interfaces de lenguaje natural para bases de datos buscan facilitar el proceso de obtener información de una base de datos al eliminar la labor de traducción de lenguaje natural a SQL, el avance de la tecnología ha permitido que el hardware no sea un impedimento para alcanzar dicho objetivo; sin embargo, todavía existen dificultades en la traducción de lenguaje natural a SQL inherentes al manejo del lenguaje natural y su conversión a SQL. Un ejemplo de esta problemática es el tratamiento de funciones de agregación, agrupamiento y subconsultas.

El problema que se aborda en este trabajo es manejar consultas que contengan funciones de agregación, agrupamiento y subconsultas en la interfaz de lenguaje natural para bases de datos desarrollada en el Instituto Tecnológico de Ciudad Madero.

Este proyecto de tesis se enfoca en el diseño e implementación de un módulo que dé tratamiento a funciones de agregación, agrupamiento y subconsultas. El módulo está integrado en la interfaz de lenguaje natural a bases de datos desarrollado en el Instituto Tecnológico de Ciudad Madero, para así mejorar el desempeño de la interfaz, aumentado su capacidad de traducción.

CAPÍTULO 1: INTRODUCCIÓN

El procesamiento de lenguaje natural para realizar consultas tiene su origen en la década de los 60s. Las primeras interfaces de lenguaje natural para bases de datos (ILNBDs) eran básicamente interfaces hechas a la medida para un dominio en específico. Estas interfaces tenían un sistema de conocimiento creado manualmente por expertos en la materia. Algunas de las primeras interfaces utilizaban técnicas de coincidencia de patrones.

En la actualidad y gracias a la evolución de la tecnología, las ILNBDs cuentan con una base tecnológica fuerte, lo que facilita su implementación; sin embargo, el reto original continúa presente, el cual consiste en contestar todas las consultas de manera correcta y que el proceso sea transparente para el usuario.

El presente trabajo sobre ILNBDs se encuentra dentro de las ciencias de la computación, en la rama de inteligencia artificial, la cual busca simular la capacidad de razonamiento en las máquinas. Finalmente, el procesamiento de lenguaje natural (PLN) es una de las subáreas de la inteligencia artificial, y las ILNBDs son una rama del primero. Por otra parte, el PLN (también conocido como lingüística computacional) es un área de la lingüística. El procesamiento de lenguaje natural, como su nombre lo indica, busca un resultado práctico.

1.1 Objetivos

Objetivo general:

> El objetivo general consiste en desarrollar un módulo para el manejo de consultas que involucren funciones de agregación, agrupamiento (cláusula GROUP BY) y subconsultas para la ILNBD desarrollada en el ITCM [Aguirre, 2014].

Objetivos específicos:

- Diseñar un método para el tratamiento de consultas que involucren funciones de agregación, agrupamiento y subconsultas.
- Implementar el método mencionado e integrarlo en el núcleo de la ILNBD [Aguirre, 2014].

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 opción 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.

Conviene mencionar que, en bases de datos de aplicaciones reales, es frecuente la necesidad de formular consultas que involucren funciones de agregación (COUNT, SUM, AVG, MIN y MAX), agrupamiento (cláusula GROUP BY) y subconsultas. Por lo tanto, para considerarse mínimamente completa, una ILNBD debe ser capaz de contestar este tipo de consultas. Desafortunadamente la ILNBD desarrollada en el ITCM [Aguirre, 2014] no ofrecía esta funcionalidad. Particularmente, en lo que respecta a este proyecto su beneficio consiste en que la nueva versión de la ILNBD ahora ofrece esta funcionalidad, con lo cual se amplía su utilidad.

1.3 Descripción del problema

El 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 consiste de 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 contenga las características especificadas en la entrada. A continuación se presentan algunos ejemplos:

Consulta con función de agregación:
¿Cuál es el área del estado más grande?
What is the area of the largest state?
SELECT MAX (area)
FROM (state)

Consulta con función de agrupamiento y agregación:
Dame el área total de los lagos por estado.
Give me the total area of the lakes by state.
SELECT SUM (area)
FROM (lake)
GROUP BY (state)

Consulta con funciones de agregación y subconsulta:
Dame el nombre del estado con el área más grande.
Give me the name of the state with the biggest area.
SELECT (state_name)
FROM (state)
WHERE area= (SELECT MAX (area) FROM (state))

Se realizó una tipificación sistemática del corpus de Geobase. La tipificación se hizo con el fin de descubrir los problemas que se encuentran en las diversas consultas del corpus. Entre los problemas encontrados se presentan los siguientes: elipsis semántica, consultas que incluyen subconsultas, consultas deductivas, con negación y otros. Se presenta en la tabla 1.1 una muestra representativa de consultas de Geobase y sus problemas (nota: se crearon algunas consultas, que no existen el corpus de Geobase, con fines ilustrativos de los problemas).

Tabla 1.1 Problemas detectados en consultas

Consultas con funciones de agregación	Elipsis	Subconsulta	Elipsis en función de agregación	Deductiva	Negación	Otros problemas
¿Cuál es el estado con la menor densidad de población?				✓*densidad de población* debe calcularse con población y área		

¿Cuál es el río más corto en US?			✓*corto* no especifica a qué columna se refiere			✓se presenta un problema con *US*, ya que no existe en la base de datos
¿Cuál es la población total de los estados colindantes con Texas?	✓					
¿Cuál es la población de la capital del estado más chico?		✓	✓*chico* no especifica a qué columna se refiere			
¿Qué ríos no corren por Tennessee?	✓				✓	

Como se puede apreciar se encuentran diversos problemas en las consultas, entre los cuales *Otros problemas* se refiere a problemas descubiertos que son diferentes de los principales. Por ejemplo: conversiones entre sistemas de medida, cálculos matemáticos y datos inexistentes.

En las funciones de agregación que involucran subconsultas y agrupamiento, se presentan los problemas ya expuestos, más algunos que son exclusivos de ellas. Al realizar la tipificación se descubrió que existen consultas con funciones de agregación que incluyen un caso de elipsis diferente a los estudiados en [Ortega, 2014]. Por ejemplo: *¿Cuál es la población de la capital del estado más chico?* Esta consulta aparentemente no presenta un problema de elipsis, pero al analizar con detenimiento se puede apreciar que involucra una función de agregación (MIN); sin embargo, no especifica sobre cuál columna pide el mínimo; puede ser el mínimo de área o de población.

Por otra parte, en las funciones de agregación se encuentra otro problema: el gran número de palabras que pueden usarse para hacer referencia a una misma función de agregación.

Un ejemplo sencillo de este problema es el siguiente: las funciones MAX y MIN que hacen referencia al valor máximo y mínimo de una columna. Algunas consultas que involucren MAX y MIN pueden ser relacionadas con precio, salario, tiempo, etc. Estas consultas se pueden formular utilizando palabras distintas, las que en términos prácticos para la traducción a SQL se refieren a las funciones MAX y MIN. Algunos ejemplos de lo que se mencionó se muestran a continucación.

1) ¿Cuál es el salario más alto del departamento de personal?
2) ¿Cuál es el salario más bajo del departamento de personal?
3) Dame el perfume más caro de la marca Kenzo.
4) Dame el perfume más barato de la marca Kenzo.
5) ¿A qué hora sale el último vuelo de hoy a la Ciudad de México?
6) Dame la hora del primer vuelo a Bogotá.

En las seis consultas mostradas se puede observar que todas hacen referencia a MAX y MIN. El problema radica en que cada consulta formula la petición utilizando palabras distintas. En la consulta 1 la palabra *alto* hace referencia a la función MAX, mientras que *bajo* en la consulta 2 hace refierencia a la función MIN. De la misma manera, *caro* y *barato* se refieren a las funciones MAX y MIN respectivamente. En el caso de consultas que manejan tiempo, las funciones MAX y MIN se expresan sin hacer referencia a un tiempo específico, por lo cual, al especificar *último* en la consulta 5, se refiere a la función MAX y contrariamente, en la consulta 6 el término *primer* se refiere a la función MIN. De esta manera se puede apreciar que algunas funciones de agregación presentan problemas en cuanto a su interpretación debido a la variedad de palabras que pueden hacer referencia a una misma función.

En las consultas que involucran la cláusula de agrupamiento GROUP BY, se detectaron problemas que complican su tratamiento. Los principales problemas son los siguientes: concatenación de palabras en la cláusula GROUP BY; ocurre cuando un

descriptor de columna usado en el agrupamiento contiene más de una palabra, por ejemplo: agrupar por número de motores. Doble agrupamiento, *por* no se refiere a GROUP BY, elipsis en GROUP BY, GROUP BY implícito en una sentencia SQL; se presenta cuando el agrupamiento no se encuentra explícito en la consulta en lenguaje natural, uso de condiciones y múltiples tablas. Como se puede apreciar, la complejidad del problema en consultas que involucran la cláusula GROUP BY radica en el gran número de problemas que se pueden encontrar al momento de realizar la traducción a SQL. Sin embargo, al requerir una función de agregación para aplicar el agrupamiento su complejidad aumenta. Esto se debe a que algunos problemas de las consultas que involucran funciones de agregación también están presentes en las consultas con agrupamiento. Entre dichos problemas se encuentran los siguientes: ambigüedad en columna y ambigüedad COUNT/SUM.

Las consultas que contienen subconsultas también tienen un grado de complejidad mayor al esperado, ya que al realizar el análisis se descubrieron problemas que complican el tratamiento de las subconsultas. Entre dichos problemas se encuentran los siguientes: validación de patrón, más de una subconsulta, uso de condiciones, múltiples tablas y uso de operadores de comparación. El principal problema del tratamiento de subconsultas es la falta de estandarización en la sintaxis de la consulta en lenguaje natural. La falta de un analizador sintáctico reduce la funcionalidad del módulo para el tratamiento de subconsultas, ya que es virtualmente imposible que este módulo pueda analizar, detectar y organizar la información de una subconsulta sin tener un patrón estandarizado que permita detectar que dicha consulta contiene una subconsulta, y que además, permita recopilar la información para construirla.

En conclusión, el tratamiento de consultas que involucran funciones de agregación, agrupamiento y subconsultas tiene una complejidad mayor a la que se presumía antes de realizar el análisis sistematizado del corpus.

Es oportuno mencionar que algunos de los problemas mencionados en esta sección no habían sido detectados en la tesis “Traducción de consultas de lenguaje natural español a SQL que involucran agrupamiento” [Bautista, 2014].

1.4 Alcances y limitaciones

Se realizó una tipificación del corpus de Geobase con la finalidad de delimitar el alcance de este proyecto. A continuación se muestran los alcances:
Se consideran consultas que involucran los siguientes elementos:

1. Funciones de agregación.
2. Funciones de agrupamiento (cláusula GROUP BY).
3. Subconsultas.

En cuanto a las limitaciones se encuentran los siguientes puntos:

1. El único idioma soportado es el español.
2. El medio de comunicación es en lenguaje escrito.
3. 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.
4. No proporciona información que no esté explícita en la BD.
5. No se considera el problema de transformar una consulta a su equivalente optimizada.
6. Las consultas deben ser léxica y sintácticamente correctas.
7. La información solicitada en la consulta a la base de datos debe expresarse antes que la condición.
8. Las consultas sólo serán en forma interrogativa e imperativa.
9. Si la consulta contiene un valor compuesto por dos o más palabras, el cual corresponde a una columna de la base de datos, deberá estar escrito entre comillas dobles para detectar que es un único valor en la base de datos. Por ejemplo: nombres de personas (“Juan Pérez Fernández”), domicilios, etc.
10. El formato para representar fechas debe ser: dd/mm/aaaa.

11. No se resuelven consultas que contengan los siguientes problemas: deducción, anáfora, negación y otros (problemas nuevos encontrados en el corpus).
12. En el caso de consultas con agrupamiento (GROUP BY), no se resuelven los siguientes problemas: GROUP BY implícito en sentencia SQL, uso de condición HAVING, elipsis en GROUP BY, agrupamiento en subconsultas, uso de más de una tabla.
13. En el caso de subconsultas, no se resuelven los siguientes problemas: más de una subconsulta, uso de condición HAVING, uso de operadores de comparación, uso de más de una tabla.
14. Las pruebas sólo se realizan para las bases de datos de Geobase y ATIS.

1.5 Organización del documento

Capítulo 2: Este capítulo presenta los conceptos principales relacionados con este trabajo de tesis, incluyendo también un resumen de algunos trabajos relacionados y antecedentes del proyecto relativos al manejo de consultas de lenguaje natural para bases de datos.

Capítulo 3: En este capítulo se presentan las etapas realizadas para el desarrollo del módulo que da tratamiento a las funciones de agregación, agrupamiento y subconsultas. Estas etapas son: el análisis sistematizado de problemas en el corpus de consultas, la tipificación de los problemas, el estudio del núcleo de la ILNBD y el tratamiento de problemas para su resolución por medio de la implementación del módulo, incluyendo un ejemplo de cada tratamiento.

Capítulo 4: En este capítulo se describe el escenario de pruebas, el cual consiste en las características de software y hardware utilizado, la experimentación realizada y los resultados obtenidos.

Capítulo 5: En este capítulo se presentan las conclusiones obtenidas a partir de este proyecto de tesis, así como los trabajos futuros que se podrían realizar a partir de los resultados de este proyecto.

CAPÍTULO 2: MARCO TEÓRICO Y TRABAJOS RELACIONADOS

En el presente capítulo se presentan las definciones más importantes necesarias para tener una mejor comprensión del proyecto. Por otra parte, también se incluyen los trabajos relacionados con este proyecto, los cuales tienen objetivos similiares a los de este proyecto. Su estudio ayuda a tener un panorama más amplio de los métodos ya utilizados para dar solución a los problemas que se presentan en el procesamiento de lenguaje natural.

2.1 Marco teórico

Lenguaje natural

Lenguaje hablado o escrito por seres humanos, el cual se divide en dos ramas [Bautista, 2014]:

- Lenguaje escrito: utiliza el conocimiento léxico, sintáctico y semántico del lenguaje.
- Lenguaje hablado: se compone de los campos del lenguaje escrito más la fonología.

Procesamiento de lenguaje natural

El procesamiento de lenguaje natural (PLN, NLP en inglés) es un conjunto de técnicas computacionales diseñadas para analizar y representar textos en uno o más niveles del análisis lingüístico, con el fin de procesar el lenguaje como un humano [Aguirre, 2014].

Interfaz de lenguaje natural para bases de datos

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

Sistema administrador de bases de datos

Consiste en una colección de datos interrelacionados y un conjunto de programas para accederlos. Esta colección de datos se denomina comúnmente base de datos (BD), y contiene información relevante para una organización. El objetivo de un SABD es proporcionar una forma de almacenar y recuperar la información de una BD de manera que sea lo más práctica y eficiente [Rojas, 2009].

Lenguaje de manipulación de datos

Tiene como función permitir crear, modificar, recuperar y eliminar registros (entidades en un modelo lógico) de una base de datos [Fernández, 2006]. Específicamente permite las siguientes operaciones:

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

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

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]. Sus resultados son los siguientes:

- COUNT: número de valores en la columna (o en una tabla).
- SUM: suma de los valores en la columna.
- AVG: promedio de los valores en la columna.
- MAX: el mayor valor de la columna.
- MIN: el menor valor de la columna.

Cláusula GROUP BY

GROUP BY reacomoda la tabla especificada en 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].

Subconsultas

Una subconsulta es una sentencia SELECT que se utiliza dentro de otra sentencia SELECT. La subconsulta obtiene unos resultados intermedios (uno o más datos) que se emplean en la consulta principal [Quintana, 2008].

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.

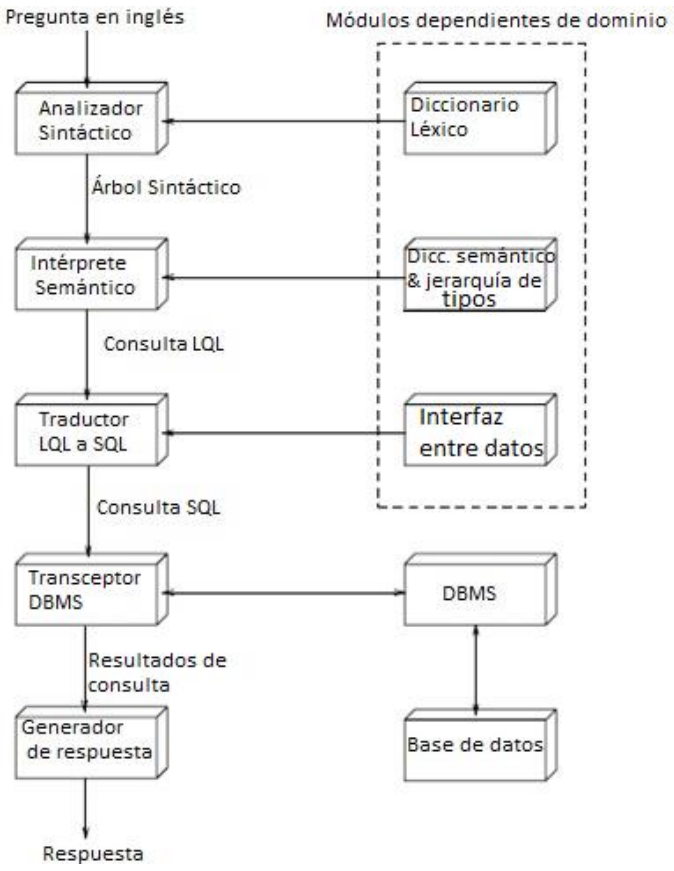

Figura 2.1 Arquitectura de MASQUE/SQL

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

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 en la interfaz OWDA

2.2.3 C-PHRASE

Esta interfaz se enfoca en técnicas de procesamiento de lenguaje natural [Aguirre, 2014]; para esto utiliza cinco 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 (TRL).

En su proceso de traducción modela las consultas en una versión de alto orden de cálculo de tuplas de Codd, 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 la resolución 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.

Cities of over 100,000 people in the largest area mid-western state.

```
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 [Baustisa, 2014]. 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, sólo es necesario crear una nueva interfaz para cada 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 los campos y tablas, y utiliza su diccionario integrado para tratar de predecir los sinónimos utilizados en las consultas, en caso en que ELF no pueda asignar un término existe una ventanta de búsqueda en la cual el usuario ingresa el término y selecciona el atributo correcto, a partir de los datos asociados. Por ejemplo: al ingresar López, se muestra el atributo TRABAJADOR_APELLIDOS donde TRABAJADOR es el nombre de la tabla y APELLIDOS el nombre del campo 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 campos:

- 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:

1. Se determinan las cláusulas y subcláusulas presentes mediante un arreglo de términos base.
2. Se detectan las variables para su búsqueda en el diccionario de dominio.
3. Al encontrar los términos correspondientes a datos, comúnmente asociados a la cláusula WHERE o HAVING, se verifica que existan en el diccionario de contenido.
4. 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 en la Figura 2.3.

Ejemplo:

```
SELECT MAX(SALARY)
WHERE JOB = 'DESIGNER'
```

Se traduce a:

Muestra máximo de salario donde puesto sea igual a diseñador

Figura 2.3 Ejemplo de consulta en SNL2SQL

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

Tabla 2.1 ILNBDs y características

Interfaz	Funciones de agregación	Función de agrupamiento	Independencia de dominio	Idioma	Año
MASQUE	✓	✓	✓	Inglés	1992
OWDA	✓	✓	✓	Inglés	2007
C-PHRASE	✓	✗	✓	Inglés	2008
ELF	✓	✓	✓	Inglés	2010
SNL2SQL	Uso limitado	Uso limitado	No reportado	Español	2010
Este proyecto	✓	✓	✓	Español	2015

La tabla 2.1 muestra la comparación de 5 interfaces con este proyecto, tomando en cuenta las características que se desarrollaron como puntos de comparación entre las interfaces. Como se puede observar sólo la interfaz SNL2SQL funciona en idioma español, lo que es una de las carateristicas principales de este proyecto; Aunque existen interfaces que cumplen con todos los puntos de comparación ninguna de ellas fue desarrollada para el idioma español.

2.3 Antecedentes

El proyecto titulado “Traductor de lenguaje natural español a SQL para un sistema de consultas a bases de datos” desarrollado en el CENIDET, tuvo como fin el desarrollo de un módulo traductor para una ILNBD [González, 2005]. Las principales características que destacan de dicho proyecto es que la técnica utilizada por el traductor es portable a cualquier dominio, realiza la creación automática de un diccionario de dominio utilizando metadatos de la base de datos a consultar, y utiliza las operaciones de unión e intersección en la técnica de traducción para dar tratamiento a la preposición *de* y la conjunción *y*.

El objetivo de dicha tesis fue desarrollar un método de traducción que aumentara el porcentaje de consultas contestadas correctamente, así como que el proceso fuera independiente de dominio. Los resultados obtenidos dan evidencia de que se logró el objetivo, al demostrar que la técnica utilizada ayuda a configurar la interfaz de una manera fácil y rápida, así como un aumento sustancial en el porcentaje de consultas traducidas correctamente el cual fue del 86%, sin dejar de lado su portabilidad a cualquier dominio.

La tesis desarrollada por Carlos Rojas tuvo como finalidad enfocarse al diseño de procesos de diálogo, con el fin de resolver uno de los problemas más difíciles que están presentes en el procesamiento de lenguaje natural: la elipsis semántica [Rojas, 2009]. La elipsis semántica es la omisión de elementos que, aunque no son obligatorios sintácticamente, son requeridos para una completa interpretación semántica de un texto. Un tipo de elipsis semántica es la resolución de referencia que depende de la reconstrucción de una categoría omitida semánticamente [Mcshane, 2004].

Para resolver el problema se desarrolló un administrador de diálogo que permite aumentar el porcentaje de consultas contestadas correctamente, el cual se basó en una tipificación de problemas en consultas. Una de las características más sobresalientes es la utilización de ciclos pregunta-respuesta, los cuales ayudan al proceso de aclarar consultas con elipsis semántica.

Los resultados obtenidos son una tipificación de problemas en consultas, el cual se puede aplicar a otros idiomas, y un administrador de diálogo basado en la tipificación, lo cual permite obtener independencia de dominio. Las pruebas experimentales demostraron que, al utilizar un administrador de diálogo, se logra incrementar el desempeño de la ILNBD en un rango del 30-35% de consultas correctamente contestadas.

El antecedente más cercano con el proyecto objeto de este proyecto es la tesis desarrollada en el ITCM titulada “Traducción de consultas de lenguaje natural español a SQL que involucran agrupamiento” [Bautista, 2014]. El objetivo de dicha tesis fue mejorar la interfaz desarrollada por el ITCM-CENIDET al agregarle la capacidad de traducción de consultas con agrupamiento. Esto se realizó mediante el reconocimiento de consultas que al traducirlas a SQL involucrasen la cláusula GROUP BY o funciones de agregación.

Se realizó un análisis de 5 corpus para tipificarlos; de esta manera se pudo categorizar el tipo de consulta con respecto a la función de agregación que involucra y el uso de GROUP BY. Su funcionamiento consiste en utilizar el análisis léxico para etiquetar palabras o frases que involucren funciones de agregación y/o agrupamiento. También se actualizó el módulo “Identificación de las frases SELECT y GROUP BY” durante el análisis semántico para así incorporar la semántica de las funciones de agregación y agrupamiento. En consecuencia, se obtuvo como resultado entre un 83-92% de consultas correctamente contestadas, de los 5 corpus utilizados en las pruebas.

El objetivo del proyecto objeto de esta tesis consiste en el desarrollo de un módulo de funciones de agregación, agrupamiento y subconsultas. El módulo para funciones de agregación se puede apreciar en la Figura 2.4. Este módulo se encuentra dentro de las capas funcionales del análisis semántico. El módulo fue incrustado en el núcleo desarrollado en el ITCM en el proyecto denominado "Modelo Semánticamente Enriquecido de Bases de Datos para su Explotación por Interfaces de Lenguaje Natural" [Aguirre, 2014]; por lo tanto, se describirá a continuación con más detalle el funcionamiento y capacidad del núcleo.

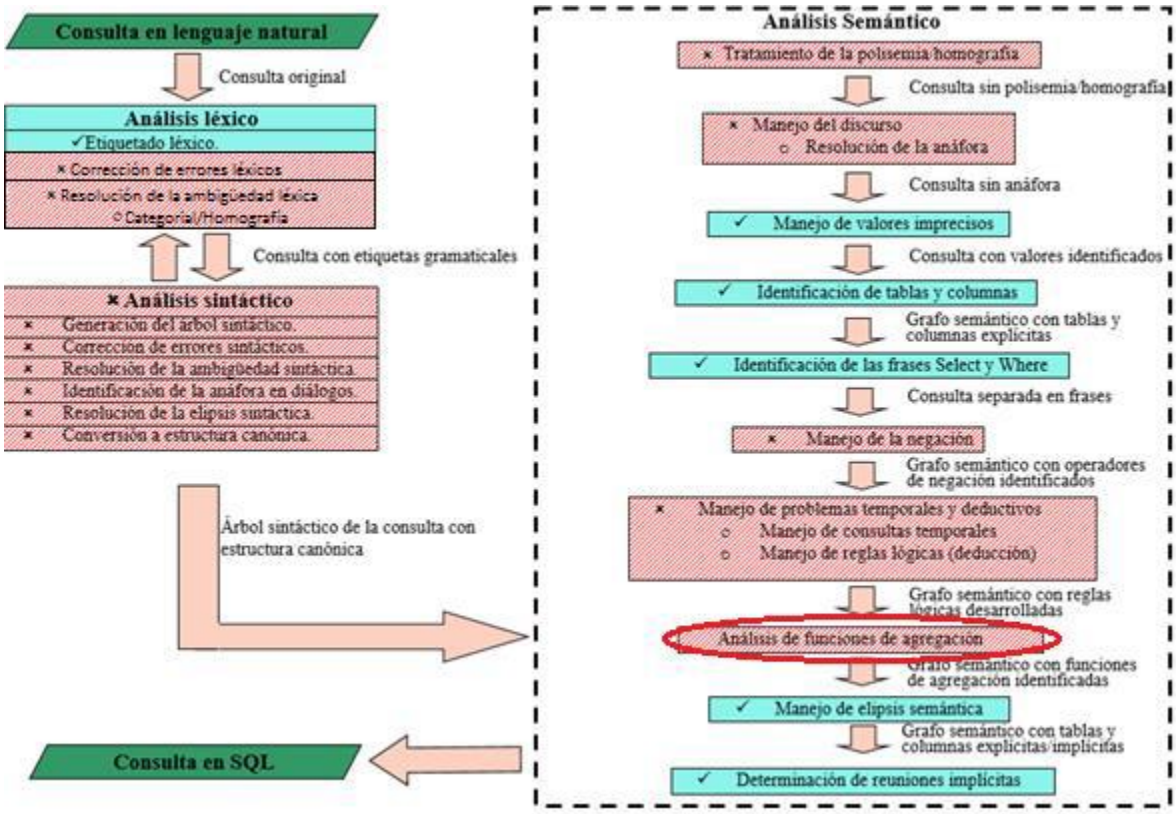

Figura 2.4 Capas funcionales de la ILNBD

El núcleo de la ILNBD consta de tres módulos principales para su funcionamiento: el análisis léxico, el análisis sintáctico y el análisis semántico. La arquitectura de los tres módulos se presenta en la Figura 2.4. Su modo de operación permite ir procesando la consulta desde una expresión en lenguaje natural hasta obtener una expresión en SQL. El primer paso consiste en recibir la consulta en lenguaje natural para posteriormente realizar

el análisis léxico, el cual se encarga de etiquetar cada palabra de la consulta de acuerdo a su categoría gramatical. Esto se realiza buscando cada palabra en el lexicón que contiene todas las palabras del idioma. En dado caso de que alguna palabra no sea encontrada en el lexicón y por ende no sea etiquetada, queda marcada con un signo de interrogación y se consiera como un posible valor de búsqueda.

Después, en lugar de realizar un análisis sintáctico, se realiza un análisis superficial a la consulta con el fin de obtener una sola categoría gramatical por palabra, ya que algunas veces las palabras poseen varias categorías gramaticales. Este análisis consiste en algunas reglas heurísticas, adicionalmente se eliminan palabras que no sean importantes para la traducción de la consulta, como artículos y preposiciones (a excepción de las que hacen referencia a una columna).

Dentro del análisis semántico se encuentra una subcapa que se encarga de manejar valores imprecisos o alias, éstos son valores que no se pudieron categorizar en el análisis léxico. Este proceso se realiza buscando en el diccionario de información semántica (DIS) el valor que se encontró para así obtener la palabra a la que hace referencia. Esto permite reducir el número de palabras sin categorizar y delimita los valores de búsqueda.

La siguiente subcapa identifica las tablas y columnas en la base de datos a las que se hace referencia en la consulta. Esta identificación se efectúa mediante una búsqueda en el DIS, localizando ya sea palabras o frases de diferente categoría sintáctica; si encuentra una palabra o frase que se encuentre en la consulta, entonces se etiqueta con la tabla o columna a la que hace referencia.

La siguiente subcapa se encarga de identificar las frases SELECT y WHERE dentro de la consulta. Una vez que se han logrado identificar las tablas, columnas y los valores de búsqueda, resulta posible dividir la consulta en las frases SELECT y WHERE. Para realizar este proceso cada valor de búsqueda es asociado a una columna de acuerdo a su cercanía y similitud de tipo de dato. El conjunto de pares de columnas y valores de búsqueda forma la frase WHERE, y el resto de las palabras etiquetadas se consideran parte de la frase SELECT.

La última subcapa realiza la tarea de determinar reuniones implícitas (*joins*) implícitas en la consulta. Esto es necesario ya que para formular una instrucción en SQL, es necesario que el grafo constituido por las tablas identificadas y las condiciones de reunión

(condiciones de búsqueda que involucran una columna de una tabla y una columna de otra tabla) sea un grafo conectado.

Una parte fundamental para el desarrollo de la traducción es el DIS, ya que almacena la información necesaria para que la interfaz pueda hacer la traducción de la consulta. El DIS es una parte vital para el éxito de la traducción, ya que la calidad y cantidad de la información semántica almacenada se ve reflejada en la precisión con la cual la interfaz comprende la consulta en lenguaje natural.

El DIS almacena palabras y frases que hacen referencia a tablas y columnas de la base de datos, así como información del esquema de la BD. La personalización de la interfaz para una BD en particular consiste en llenar el DIS con información semántica relevante de la BD. Este llenado de datos se realiza incluyendo en el proceso: relaciones entre tablas, palabras y frases que hacen referencias a tablas y columnas, valores imprecisos y alias, lo cual, como anteriormente se mencionó, facilita y mejora la eficacia de la interpretación.

Todas estas subcapas, que forman las capas funcionales de la interfaz, se encuentran en el módulo de traducción en capas funcionales que forma parte de la arquitectura general de ILNBD [Aguirre, 2014], como se puede apreciar en la Figura 2.5.

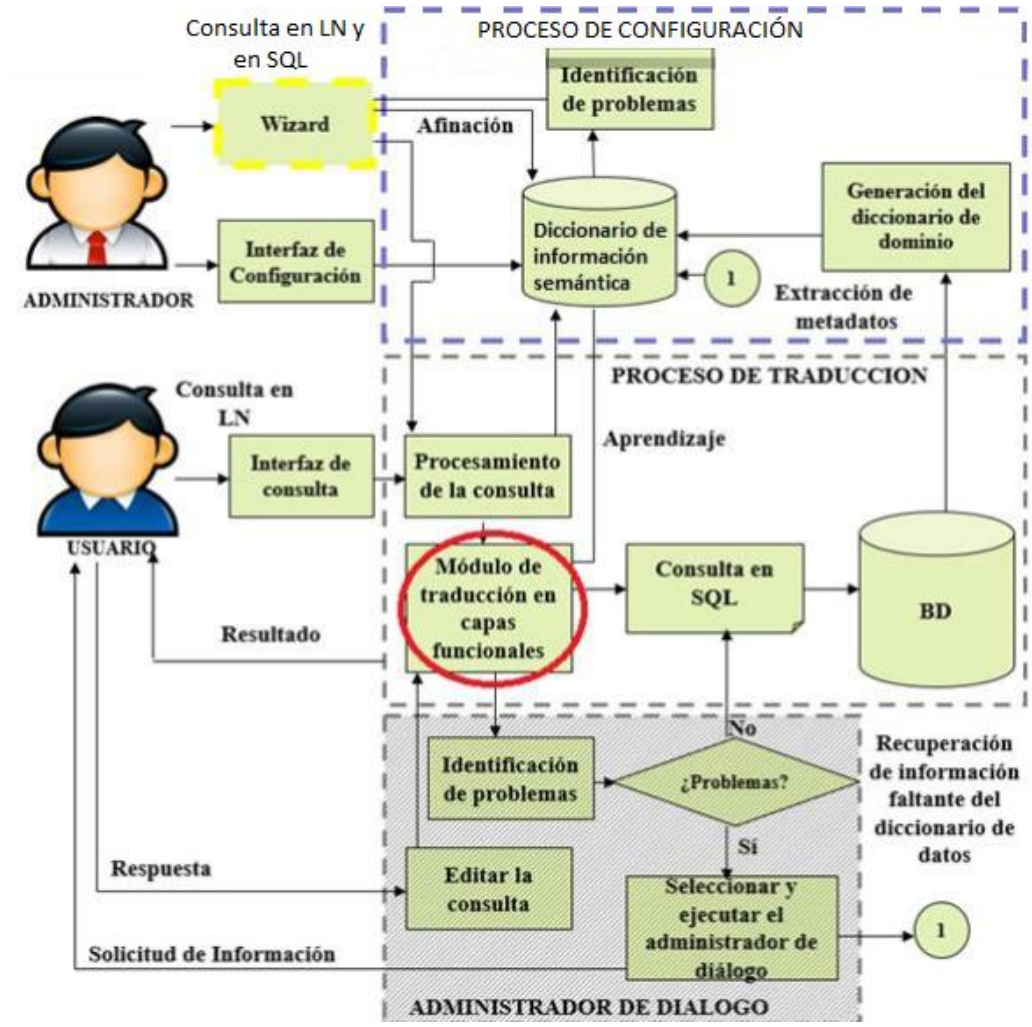

Figura 2.5 Arquitectura general de la ILNBD

CAPÍTULO 3: ANÁLISIS Y SOLUCIÓN DEL PROBLEMA

3.1 Análisis de la interfaz de lenguaje natural para bases de datos

El núcleo de la interfaz de lenguaje natural para bases de datos desarrollada en el ITCM consta de tres capas principales. 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 en que alguna palabra no sea localizada dentro del lexicón se infiere que posiblemente sea 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 consta del análisis semántico, la cual se encarga de identificar tablas y columnas en base a la información almacenada en el diccionario de información semántica. En el análisis semántico 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 es asociado 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.

Por último, se determinan reuniones implícitas, ya que para hacer la consulta en SQL es necesario que el grafo constituido por las tablas y las condiciones de reunión (condiciones de búsqueda constituida 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.

El módulo para el tratamiento de funciones de agregación, agrupamiento y subconsultas inicia después de la identificación de vistas. Su posición dentro del procesamiento de la consulta es esencial, ya que el módulo utiliza la información recabada por la interfaz para realizar el tratamiento de funciones de agregación, agrupamiento y subconsultas. Para el tratamiento de funciones de agregación y la cláusula GROUP BY se creó una tabla en el diccionario de información semántica como se puede apreciar en la Figura 3.1, la cual contiene palabras que hacen referencias a funciones de agregación, agrupamiento y su respectiva equivalencia a SQL. La tabla contiene, aparte de la cláusula GROUP BY, las palabras que se refieren a una función de agregación. Para la detección de la palabra se utiliza el lema, ya que sin importar el tiempo gramatical de la palabra el lema siempre es el mismo. El uso de una tabla que contiene las funciones de agregación ayuda en gran medida a la versatilidad del módulo, ya que palabras nuevas pueden ser agregadas sin necesidad de modificar el procesamiento del módulo.

nombre_bd	palabra	lema_palabra	clase_gram	fa-agrupamiento	columna	tabla
Geobase2.mdb	mayor	mayor	adjetivo	MAX	---	---
Geobase2.mdb	por	por	preposición	GROUP BY	---	---

Figura 3.1 Tabla de información semántica para funciones de agregación

El módulo para el tratamiento de funciones de agregación, agrupamiento y subconsultas 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 análisis semántico (*AnalisisSemantico*) contiene los métodos utilizados para realizar dicho análisis. Dentro de la clase análisis semántico (*AnalisisSemantico)* 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, la cual contiene el módulo completo. 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. Una vez detectadas tablas y columnas se realiza la identificación de frases (*identificacionFrases*), las cuales pueden ser SELECT o WHERE; su etiquetado se realiza en base a las columnas detectadas en el método anterior.

El tercer método identifica problemas en frase SELECT y WHERE llamado (*identificaProblemasFraseSelect_Where),* realiza como su nombre lo indica la identificación de problemas en las columnas etiquetadas como frases SELECT o WHERE, y en caso en que exista un problema, la interfaz presenta un mensaje al usuario indicándole el problema detectado. 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*), el cual 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 sentencia 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, en caso negativo el método para crear la cláusula SELECT (*creaClausulaSelect*) y el método para crear la cláusula WHERE (*creaClausulaWhere*) inician la construcción de la sentencia en SQL. En caso de que la consulta cumpla con el patrón, se realiza la recopilación de elementos para subconsulta (*recopilacion_subconsulta*), la cual se encarga de tomar los componentes léxicos adecuados para formar la sentencia en SQL que representa 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)* que, 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 la cláusula SELECT (*creaClausulaSelect*) y WHERE (*creaClausulaWhere*) son usados para iniciar la construcción de la sentencia en SQL. Al fnalizar, el análisis semántico continúa con su proceso hasta que construye y ejecuta la sentencia en SQL para así retornar la información solicitada. Estos procesos se pueden apreciar en la Figura 3.2.

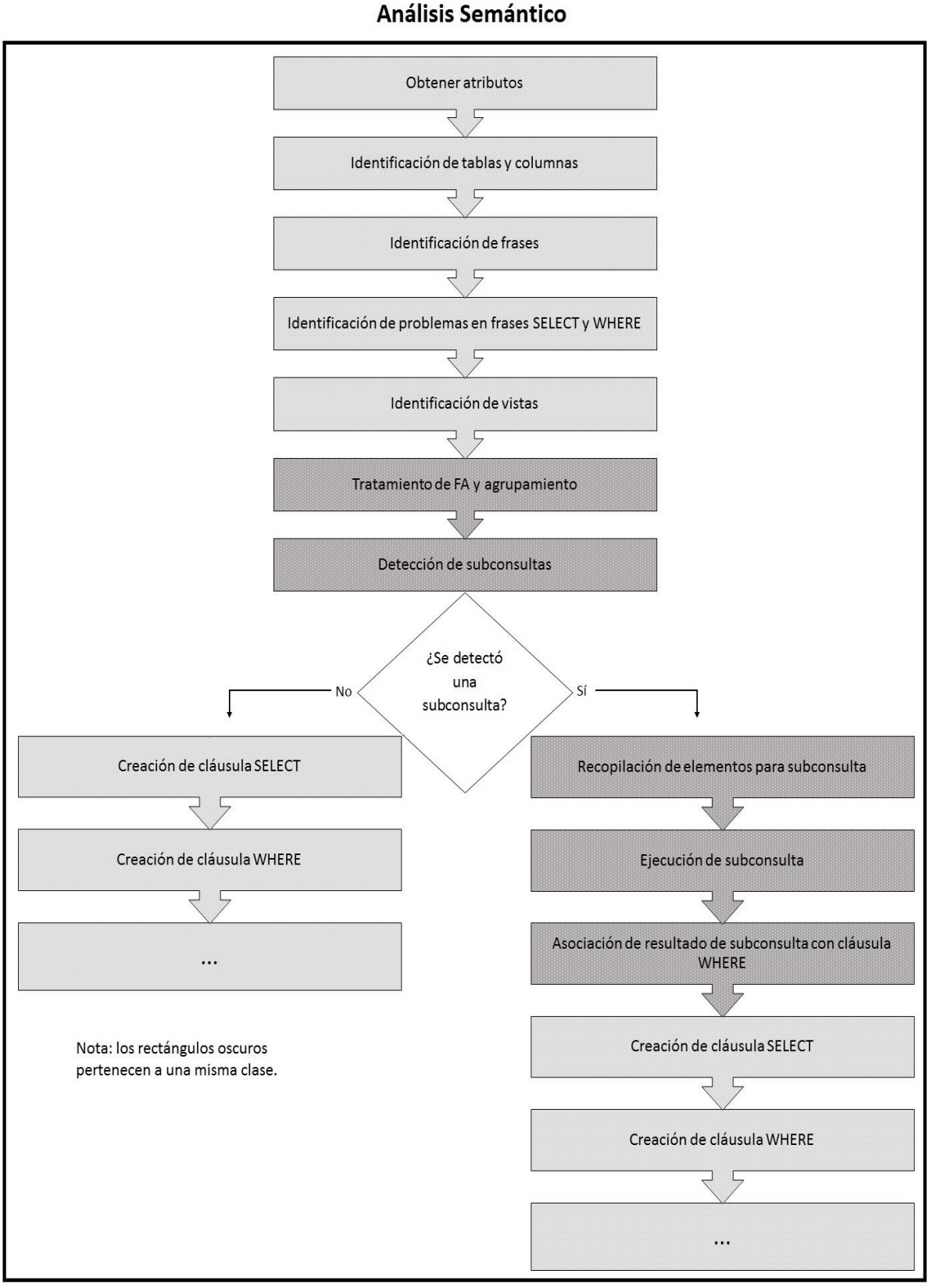

Figura 3.2 Procesos de la interfaz

Con la finalidad de mantener la estructura de la interfaz intacta y no comprometer su funcionalidad, se optó por implementar el módulo para el tratamiento de funciones de agregación, agrupamiento y subconsultas en una clase 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 sus procesos internos.

Como se mencionó previamente, sólo 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 sentencia 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. Al igual que en el resto de los procesos, el código para el etiquetado de frases sólo fue complementado sin modificar el proceso previamente implementado, lo cual garantiza que la interfaz realice sus procesos tal y como fueron diseñados.

3.2 Problemas en consultas

Al realizar un análisis sistematizado del corpus de Geobase, se detectaron problemas que no habían sido detectados previamente, los cuales aumentan la complejidad del procesamiento de la consulta en lenguaje natural. En esta sección se presentan los problemas detectados que 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 a continuación.

Consultas deductivas: las consultas deductivas están basadas en deducción por uso de inferencia. Por lo general estas consultas son formuladas a bases de datos deductivas [Aguirre, 2014].
Ejemplo: ¿Cuántas ciudades principales hay en Florida?
La palabra *principales* necesita una definición complementaria para determinar cómo obtener esta información de la base de datos.

Consultas con negación: las consultas que involucran negación son fácilmente expresadas en lenguaje natural. Por ejemplo: ¿qué zonas para acampar no requieren de permiso? Sin embargo, pueden poseer algunas dificultades cuando son traducidas de lenguaje natural a SQL [Aguirre, 2014].
Ejemplo: ¿Qué ríos no corren a través de Tennessee?
El uso de la palabra *no* convierte la consulta en una negación.

Consultas con elipsis semántica: es la omisión de elementos que, aunque no son obligatorios sintácticamente, son requeridos para una completa interpretación semántica de un texto [Mchsane, 2004].
Ejemplo: ¿Qué longitud tiene el [río] Mississippi?
El problema ocurre por la omisión de la palabra *río*, la cual se refiere a una tabla (*river*) de la base de datos.

Consultas con anáfora: son consultas que tienen la repetición de un elemento de la oración utilizando pronombres indicativos para referirse a algo mencionado previamente en la oración.
Ejemplo: ¿Cuál es la capital del estado de Arizona y la población de aquel estado?
La palabra *aquel* se refiere a Arizona.

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.
Ejemplo: combina la población de las ciudades del estado de Alabama.

La palabra *población* hace referencia a una columna que existe en las tablas *city* y *state*.

Ambigüedad COUNT/SUM: se presenta cuando hay palabras que no pueden ser discriminadas entre la función de agregación COUNT o SUM.

Ejemplo: ¿Cuántos habitantes tienen las ciudades del estado de Texas?
La palabra *cuántos* genera una ambigüedad entre las funciones de agregación de conteo y suma.

Consultas con otros problemas: son problemas descubiertos que no pueden ser clasificados en una categoría específica. Estos problemas son conversiones, cálculos matemáticos, datos inexistentes.

Ejemplo: ¿Cuántos km cuadrados hay en el estado de Arkansas?

Implica una conversión a kilómetros.

Ejemplo: ¿Cuál es la densidad de población del estado de Texas?

Implica el cálculo de la densidad de población a partir del número de habitantes y el área.

Ejemplo: ¿Cuántas ciudades hay en EUA?

EUA no existe en la base de datos.

Entre los problemas mencionados en esta sección, se resolvieron la ambigüedad en columna y la ambigüedad COUNT/SUM.

3.2.1 Problemas en funciones de agregación

Entre las consultas que contienenen funciones de agregación, se detectaron los siguientes problemas:

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.

Ejemplo:

¿Cuántos estados tienen el área mayor que 1100?

La existencia del par *mayor que* indica una desigualdad, a diferencia del uso de las palabras mayor/menor que se asocian a funciones de agregación.

Elipsis en FA: consiste en la omisión de la columna a la cual se le aplica la función de agregación.

Ejemplo: Cuál es el estado más pequeño [según área o población].
Se omite la columna (*area* o *population*) a la cual aplicar la función de agregación MIN (más pequeño).

Para los problemas mencionados en esta sección, se resolverion la validación de mayor que/menor que y la elipsis en función de agregación.

3.2.2 Problemas en agrupamiento

Los problemas detectados en las consultas con agrupamiento son los siguientes:

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.

Ejemplo: ¿Cuántas velocidades hay por tipo de aeronave?
La concatenación de 3 palabras que se refieren a una columna (*aircraft_type*).

Doble agrupamiento: caso en el cual se presenta un agrupamiento por dos columnas.

Ejemplo: ¿Cuántas montañas hay por estado por altura?

Existe una agrupación con dos columnas (*state_name* y *height*).

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.

Ejemplo: ¿Por cuántos estados corre el río colorado?

La palabra *por* no se refiere a una agrupación.

Elipsis en GROUP BY: la elipsis en GROUP BY se presenta cuando no se especifica una columna para realizar el agrupamiento.

Ejemplo: ¿Cuántos ríos hay por tamaño?

La palabra *tamaño* no se refiere a ninguna columna; por lo tanto, se presenta una elipsis en GROUP BY.

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.
Ejemplo: ¿Qué estado colinda con más estados?
En su traducción a SQL se aplica la condición HAVING una vez que se tienen todos los estados y sus colindancias agrupadas por estado dentro de una subconsulta.

GROUP BY implícito en sentencia 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.
Ejemplo: ¿Qué estado colinda con más estados?
En su traducción a SQL se utiliza un agrupamiento para organizar todos los estados y sus colindancias pero en la consulta en lenguaje natural no explicita un agrupamiento.

Múltiples tablas: se presenta cuando la consulta involucra más de una tabla.
Ejemplo: ¿Cuántos estados hay por población por montañas?
Se utilizan las tablas *state* y *mountain*.

Para los problemas mencionados en esta sección, se resolvieron los problemas de concantenación de palabras en cláusula GROUP BY y doble agrupamiento.

3.2.3 Problemas en subconsultas

Al realizar un análisis sistematizado del corpus de Geobase, se detectó que existen al menos dos patrones en las consultas que involucran subconsultas. Para este proyecto se manejaron sólo consultas con el patrón uno. El patrón para subconsultas es el siguiente: 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 formar 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.

En el caso de consultas con subconsulta se descubrieron los siguientes problemas:

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.
Ejemplo: ¿Cuál es el área del estado más grande?
Aunque la consulta tiene el patrón de subconsultas, su equivalente en SQL no requiere de una subconsulta para obtener la información solicitada.

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 sentencia en SQL.
Ejemplo: ¿Cuál es la capital del estado con el punto más alto?
Contiene 2 subconsultas ya que primero se obtiene la altura del punto más alto, después se obtiene el estado donde esta el punto más alto y a partir de ahí se obtiene su capital.

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.
Ejemplo: ¿Qué estado colinda con más estados?
Se utiliza la condición HAVING al requerirse una subconsulta para obtener la información necesaria como condición de la consulta.

Múltiples tablas: consiste en el uso de más de una tabla en la consulta y subconsulta.
Ejemplo: ¿Qué ríos hay en el estado Alaska?
Se utilizan tres tablas para poder obtener el resultado, una contiene los estados (*state*) otra los ríos (*river*) y existe una tabla intermedia (*riverstate*) que permite conectar ambas tablas.

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.

La complejidad de este problema radica en detectar el operador de comparación en la consulta en lenguaje natural.
Ejemplo: ¿Cuáles estados tienen puntos más altos que el punto más alto en Colorado?
Se utiliza un operador de comparación mayor que (>) para obtener los estados con puntos más altos que el de Colorado.

Para los problemas mencionados en esta sección, se logró resolver el problema raíz de las subconsultas, el cual consiste en detectar la subconsulta, resolverla y asociar su resultado a la consulta principal. Con la utilización de un analizador sintáctico, la capacidad del tratamiento de subconsultas sería mayor.

En general, entre los problemas detectados en consultas se lograron resolver los tres problemas raíz: funciones de agregación, agrupamiento y subconsultas. Para cada problema raíz se resolvieron diversos problemas: ambigüedad en columna, ambigüedad COUNT/SUM, validación mayor que/menor que, elipsis en función de agregación, concatenación de palabras en cláusula GROUP BY y doble agrupamiento. Es importante resaltar que los problemas no aparecen de manera exclusiva en cada consulta; es decir; existen consultas que contienen dos o más problemas de los aquí mencionados, lo cual complica su tratamiento.

3.3 Algoritmos

En esta sección se describe la estructura de datos utilizada por la interfaz. Asi como los algoritmos de propósito general, los cuales se utilizan en el tratamiento de funciones de agregación, agrupamiento y subconsultas. En las siguientes subsecciones se describen los algoritmos que son exclusivos del tratamiento de cada aspecto.

3.3.1 Estructura de datos de la interfaz

La interfaz de lenguaje natural para bases de datos del ITCM utiliza una estructura de datos similiar a una tabla, la cual almacena toda la información relacionada a cada componente

léxico de la consulta. Esta estructura permite analizar de manera aislada cada componente léxico, lo que permite mantener la información organizada. El uso de dicha estrcutura es de vital importancia para el tratamiento de funciones de agregación, agrupamiento y subconsultas.

La estructura de datos, similar a una tabla, se compone de los elementos mostrados en la Figura 3.3. La sección *componente léxico* almacena las palabras que existen en la consulta. La sección *marcado* adquiere el valor booleano de verdadero o falso, donde verdadero indica que el componente léxico es utilizado en la construcción de la consulta en SQL. La sección denominada *clase* representa la clase gramatical de la palabra. La sección *frase* se utiliza cuando existen palabras que constituyen una frase; por ejemplo, número de motores. La sección *identificador de frase* almacena un conjunto de números, donde cada dígito representa la posición de un componente léxico perteneciente a una frase. La sección *tipo* se utiliza para el tipo de frase de cada componente léxico; por ejemplo, SELECT, WHERE, GROUP BY. La sección *etiqueta de columna* se utiliza sólo cuando el componente léxico se refiere a una columna, y en esta sección se almacena la columna para la sentencia de SQL. La sección *etiqueta de tabla* (a semejanza de *etiqueta de columna*) almacena la tabla para la sentencia de SQL. Por último, la sección *etiqueta final* se utiliza para señalar cada palabra que será usada en la traducción, con su equivalente semántico en SQL.

componente léxico	marcado	clase	lema	frase	Identificador de frase	tipo	etiqueta de columna	etiqueta de tabla	etiqueta final

Figura 3.3 Estructura de datos para el análisis semántico

Algoritmo 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
```

El algoritmo para validación de *mayor que* y *menor que* tiene como objetivo discriminar entre las funciones de agregación MAX y MIN y las desigualdades (>, <).
Esto se realiza al detectar si la palabra clave tiene en la siguiente posición la palabra *que*, de ser así se supone que es una desigualdad; en caso en que no contenga la palabra *que*, entonces es etiquetado como una función de agregación.

Algoritmo 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"
```

El algoritmo para búsqueda de frase en SID tiene como finalidad extraer la función de agregación *fa* del diccionario de información semántica. El proceso de extracción consiste en comparar la frase (posible referencia a una función de agregación) contra las frases contenidas en el diccionario de información semántica.

El algoritmo tiene tres posibles resultados. El primer caso es cuando la palabra se refiere a una función de agregación, y por lo tanto, se obtiene la función de agregación a la que hace referencia. El segundo caso ocurre cuando la frase está contenida en una frase más grande; es decir, forma parte de un conjunto de palabras; en este caso el algoritmo indica que la frase necesita más palabras para que sea igual a una contenida en el diccionario de información semántica. Por ultimo, el tercer caso sucede cuando la frase no fue encontrada en las frases extraídas del diccionario de información semántica. Cabe resaltar que el

segundo caso (cuando la frase requiere más de una palabra) puede retornar el mismo resultado una vez que se ha buscado con dos palabras, indicando que la frase que se busca está formada por más de dos palabras, es decir tres o más; esto garantiza que, sin importar el número de palabras contenidas en la frase, el algoritmo podrá manejar la frase hasta que dicha frase contenga el número de palabras necesarias para igualar una frase del diccionario de información semántica (en caso que exista una).

Algoritmo 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
```

El algoritmo para solución de ambigüedad resuelve la ambigüedad existente entre las funciones de agregación COUNT y SUM. El algoritmo realiza una búsqueda a través de la estructura que almacena los componentes léxicos para corroborar si existe un componente léxico que se encuentre etiquetado como COUNT/SUM. En caso afirmativo, se activa un diálogo con el usuario para solucionar la ambigüedad mediante la selección de la función de agregación correcta.

Algoritmo 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
```

El algoritmo para solución de elipsis en funciones de agregación tiene como propósito evitar que una consulta sea traducida a SQL sin tener una columna a la cual aplicar la función de agregación. Para resolver este problema se realiza una búsqueda en la estructura que contiene los componentes léxicos buscando algún componente léxico que haya sido marcado con frase SELECT y no contenga una columna detectada; de ser así se activa un diálogo para que el usuario seleccione la columna adecuada a la consulta.

3.3.2 Algoritmo para funciones de agregación

El algoritmo para detección de funciones de agregación y agrupamiento es el algoritmo más importante del tratamiento para funciones de agregación, agrupamiento y subconsultas. Esto debido a que, tanto el agrupamiento como las subconsultas requieren del tratamiento de las funciones de agregación. El algoritmo tiene como objetivo detectar, etiquetar y modificar información de la estructura que contiene a los componentes léxicos para que de este modo la interfaz obtenga la información y construya la consulta en SQL.

Algoritmo 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
```

El funcionamiento del algoritmo es sistemático, ya que analiza cada componente léxico que aún no haya sido identificado en la consulta. Este proceso permite detectar posibles referencias a funciones de agregación o cláusula GROUP BY. Al detectar una posible referencia, entonces invoca al método de búsqueda en frase SID para corroborar si efectivamente se trata de una función de agregación o cláusula GROUP BY.

Una vez hecho esto, el algoritmo analiza el valor obtenido, en caso en que sea una función de agregación o cláusula GROUP BY, el tipo de frase de dicho componente léxico es cambiado a “Frase Select” o ”GROUP BY”. Cuando el componente léxico es parte de

una frase de mayor tamaño el algoritmo detecta el siguiente componente léxico y lo concatena con el actual para así construir una frase que pudiera estar en el diccionario de información semántica. Como se puede apreciar, este algoritmo maneja la parte medular del tratamiento para funciones de agregación, agrupamiento y subconsultas; su diseño permite cierto grado de flexibilidad en cuanto a la detección de palabras claves, esto gracias al uso del diccionario de información semántica que permite agregar o eliminar palabras o frases que se refieran a funciones de agregación o agrupamiento.

Se presenta un ejemplo del funcionamiento del tratamiento de funciones de agregación. De la estructura de datos del análisis semántico, sólo se muestra la información requerida por el módulo para el tratamiento de las funciones de agregación.

Ejemplo:

La siguiente estructura muestra los valores relevantes obtenidos durante el análisis semántico para la consulta *Dame el promedio de las longitudes de los ríos*.

Componente léxico	Dame	el	promedio	de	las	longitudes	de	los	ríos
Lema	dame	el	promedio	de	las	longitud	de	los	río
Tipo						Frase Select			
Etiqueta final						River.length			
Marcado	false	false	false	false	false	true	false	false	false

El proceso inicia verificando que *dame* se encuentre marcado como false. Como éste es el caso, entonces se hace: *frase*←"dame" y después se obtiene *frases_SID*← getFraseDelSID(*frase*), el cual resulta vacío.

frases_SID

Como *frases_SID* está vacío; el ciclo se mueve al siguiente componente léxico. El proceso verifica que *el* se encuentre marcado como false. Como éste es el caso, entonces se hace *frase*←"el" y después se obtiene *frases_SID*← getFraseDelSID(*frase*), el cual resulta vacío.

frases_SID

Como *frases_SID* está vacío; el ciclo se mueve al siguiente componente léxico. El proceso verifica que *promedio* se encuentre marcado como false. Como éste es el caso, entonce se hace *frase*←"promedio" y después se obtiene *frases_SID*← getFraseDelSID(*frase*).

frases_SID	
promedio	AVG

Como *frases_SID* no está vacío; se llama al método busquedaFraseEnSID, el cual buscará *frase* en *frases_SID*. De esta búsqueda se obtiene lo siguiente:

frases_SID	
promedio	AVG

frase = *frases_SID*[0,0]

fa←*frases_SID*[0,1]

Al encontrar *promedio* en la primera fila (fila cero), almacena la función AVG asociada a esta frase en *fa* y retorna la variable al método principal. En el método principal se obtiene *fa* que es el retorno del método busquedaFraseEnSID.

Al ser *fa* distinta de "0" y "+++", se pone *fa* como etiqueta final de ese componente léxico y se define su tipo de frase, tal como se muestra enseguida.

setEtiquetaFinal[Q_2, *fa*]

setMarcado[Q_2, true]

SetTipoFrase[Q_2,"Frase Select"]

Después de este proceso, los valores de la estructura del análisis semántico quedan como se muestra enseguida.

Componente léxico	Dame	el	promedio	de	las	longitudes	de	los	ríos
Lema	dame	el	promedio	de	las	longitud	de	los	río
Tipo			Frase Select			Frase Select			
Etiqueta final			AVG			River.length			
Marcado	false	false	true	false	false	true	false	false	false

El proceso continúa con el componente léxico *de*. Como este componente está marcado como false, se analiza y al no existir en el SID, *frases_SID* tendrá un tamaño de cero. En estas circunstancias, el ciclo avanzará al componente léxico *las* que, de igual manera, está marcado como false, y al analizarlo, se determina que no existe dentro del SID. Al llegar al componente léxico *longitudes*, se determina que se encuentra marcado como true; por lo tanto, se omite su análisis. En tales circunstancias, el proceso continúa con el siguiente componente léxico que está marcado como false, pero no existe en el SID. El proceso continua con los dos últimos componentes léxicos, los cuales tampoco se encuentran en la información de funciones de agregación en el SID, hasta que finalmente llega al final de la consulta.

Inmediatamente el método para resolver ambigüedad es ejecutado, pero no ocurre ninguna modificación, ya que en el atributo etiqueta final no existe la etiqueta "COUNT/SUM"; por lo tanto, no hay ambigüedad. Para concluir el tratamiento, el método para resolución de elipsis en FA localiza el componente léxico de tipo SELECT que, como se puede apreciar, no se encuentra vacío en el campo de etiqueta final; por lo tanto, la columna está especificada y no hay ninguna modificación que realizar (algoritmos 3 y 4).

3.3.3 Algoritmo para agrupamiento

El algoritmo para asociación de columna GROUP BY detecta y modifica la columna que será usada para realizar el agrupamiento en caso de que la consulta requiera agrupamiento. El diseño de este algoritmo asegura que la columna usada para el agrupamiento sea la columna que se encuentra después de la palabra clave requerida para etiquetar dicho componente léxico como agrupamiento.

Algoritmo 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
```

En esta sección se presenta un ejemplo utilizando el pseudocódigo explicado previamente. De la estructura de datos del análisis semántico, sólo se muestra la información requerida por el módulo para el tratamiento de las funciones de agregación/agrupamiento.

Ejemplo:

La siguiente estructura muestra los valores relevantes obtenidos durante el análisis semántico para la consulta *Total de ciudades por estado*.

Componente léxico	Total	de	ciudades	por	estado
Lema	total	de	ciudad	por	estado
Tipo			Frase Select		Frase Select
Etiqueta de columna			city.city_name		city.state_name
Etiqueta final			city.city_name		city.state_name
Marcado	false	false	true	false	true

El proceso inicia verificando que *total* se encuentre marcado como false. Como éste es el caso, entonce se hace *frase*←"total" y después se obtiene *frases_SID*← getFraseDelSID(*frase*).

frases_SID	
total	COUNT/SUM

Como *frases_SID* no está vacío; se llama al método BusquedaFraseEnSID. BusquedaFraseEnSID buscará *frase* en *frases_SID*. De esta búsqueda se obtiene lo siguiente:

frases_SID	
total	COUNT/SUM

frase←*frases_SID*[0,0]
fa←*frases_SID*[0,1]

Al encontrar *total* en la primera fila (fila cero), se almacena la función COUNT/SUM asociada a esta frase en *fa* y retorna la variable al método principal. En el método principal se obtiene *fa* que es el retorno del método BusquedaFraseEnSID.

Al ser *fa* distinta de "0" y "+++", se pone *fa* como etiqueta final de ese componente léxico y se marca como verdadero, tal como se muestra enseguida.
setEtiquetaFinal[Q_0, *fa*]
setMarcado[Q_0, true]

Una vez hecho esto, se determina si el valor es una función de agregación o agrupamiento. Al comprobar que *fa* es "COUNT/SUM", se define su tipo de frase:
setTipoFrase[Q_0, "Frase Select"]

Después de este proceso, los valores de la estructura del análisis semántico quedan como se muestra enseguida.

Componente léxico	Total	de	ciudades	por	estado
Lema	total	de	ciudad	por	estado
Tipo	Frase Select		Frase Select		Frase Select
Etiqueta de columna			city.city_name		city.state_name
Etiqueta final	COUNT/SUM		city.city_name		city.state_name
Marcado	true	false	true	false	true

Una vez analizado el primer componente léxico, el algoritmo continúa con el siguiente componente léxico. Para tal efecto, se verifica que *de* esté marcado como false, y como es así, entonces se hace *frase*←"de" y después se obtiene *frases_SID*← getFraseDelSID(*frase*), el cual resulta vacío.

frases_SID	

Como *frases_SID* está vacío; el ciclo se mueve al siguiente componente léxico.

El proceso verifica que *ciudad* se encuentre marcado como false; al estar marcado como true, el algoritmo omite su análisis y pasa al siguiente componente léxico.

Al encontrar que la unidad léxica *por* se encuentra marcada como false, se hace *frase*←"por" y después se obtiene *frases_SID*←getFraseDelSID(*frase*). De esta búsqueda se obtiene lo siguiente:

frases_SID	
por	GROUP BY

Como *frases_SID* no está vacío; se llama al método BusquedaFraseEnSID, el cual buscará *frase* en *frases_SID*. De esta búsqueda se obtiene lo siguiente:

frases_SID	
por	GROUP BY

Al encontrar *por* en la primera fila (fila cero), almacena la función GROUP BY asociada a esta frase en *fa* y retorna la variable al método principal.

En el método principal se obtiene *fa* que es el retorno del método BusquedaFraseEnSID.

Al ser *fa* distinta de "0" y "+++" se pone *fa* como etiqueta final de ese componente léxico, tal como se muestra enseguida.

setEtiquetaFinal[Q_3, *fa*]

setMarcado[Q_3, true]

Una vez hecho esto, se determina si el valor es una función de agregación o agrupamiento. Al comprobar que *fa* es "GROUP BY", se asigna su tipo de frase como se muestra enseguida.

setTipoFrase[Q_3,"GROUP BY"]

Con el fin de realizar la asociación de columna, la variable *posición* toma el valor de la posición del componente léxico, el cual en este caso es 3 y también la variable *flag* es cambiada a true, ya que se ha detectado un agrupamiento.

Después de este proceso, los valores de la estructura del análisis semántico quedan como se muestra enseguida.

Componente léxico	Total	de	ciudades	por	estado
Lema	total	de	ciudad	por	estado
Tipo	Frase Select		Frase Select	GROUP BY	Frase Select
Etiqueta de columna			city.city_name		city.state_name
Etiqueta final	COUNT/SUM		city.city_name	GROUP BY	city.state_name
Marcado	true	false	true	true	true

Al continuar con el análisis de los componentes léxicos, se observa que el último componente léxico (*estado*) está marcado como true; por lo tanto, no se realiza su análisis.

Una vez finalizada la ejecución del algoritmo principal, se procede con el método de resolución de ambigüedad que modifica la EtiquetaFinal si ésta es "COUNT/SUM". En este caso el algoritmo detecta que el primer componente léxico tiene como etiqueta final "COUNT/SUM"; por lo tanto, se activa un diálogo con el usuario para resolver la ambigüedad.

Como se puede apreciar en el ejemplo, el primer componente léxico tiene ambigüedad, por lo cual se ejecuta el método que cambia la etiqueta final del componente léxico. Después de este proceso, los valores de la estructura del análisis semántico quedan como se muestra enseguida.

Componente léxico	Total	de	ciudades	por	estado
Lema	total	de	ciudad	por	estado
Tipo	Frase Select		Frase Select	GROUP BY	Frase Select
Etiqueta de columna			city.city_name		city.state_name
Etiqueta final	COUNT		city.city_name	GROUP BY	city.state_name
Marcado	true	false	true	true	true

Para concluir el procesamiento del presente ejemplo, se ejecuta el método para asociación de columna para group by.

Al comprobar que la variable *flag* es verdadera, sabemos que existe un agrupamiento; por lo tanto, se se inicia un ciclo for a partir de la variable *posición*+1:

posición ← 3+1

Al validar que

TipoFrase[Q_4] = "Frase Select"

EtiquetaColumna[Q_4] <> ∅

EtiquetaFinal[Q_4] <> ∅

Sabemos que existe una columna ya identificada; y por lo tanto, se supone que es la columna para agrupamiento; en consecuencia:

TipoFrase(Q_4,"GROUP BY")

Finalmente, los valores de la estructura del análisis semántico quedan como se muestra enseguida.

Componente léxico	Total	de	ciudades	por	estado
Lema	total	de	ciudad	por	estado
Tipo	Frase Select		Frase Select	GROUP BY	GROUP BY
Etiqueta de columna			city.city_name		city.state_name
Etiqueta final	count		city.city_name	GROUP BY	city.state_name
Marcado	true	false	true	true	true

3.3.4 Algoritmo para subconsultas

Como se mencionó previamente, al realizar un análisis sistematizado del corpus de Geobase se detectó que existen al menos dos patrones dentro de las consultas que involucran subconsultas. Para este proyecto se manejaron sólo consultas con el patrón uno (ver Subsección 3.2.3). El patrón para subconsultas es el siguiente: columna | columna interna | FA. 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.

Algoritmo 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
```

El algoritmo inicia localizando la función de agregación, esto con el fin de ubicarse en el componente léxico correcto que le permitirá identificar si la consulta cumple con el

patrón uno para ser procesada. A partir de la posición del componente léxico que se refiere a la función de agregación, se busca en el componente léxico de la posición anterior para verificar que sea una columna, al corroborar esta información la etiqueta queda almacenada en una variable *auxiliar*. Para comprobar que la columna sea de tipo numérico se ejecuta un algoritmo externo que permite validar dicha información. En caso de que la columna no sea de tipo numérico y no tenga la función de agregación COUNT, se rechaza la consulta como subconsulta, ya que, aunque es probable que cumpla con el patrón, no puede ser procesada por la incongruencia entre función de agregación y tipo de dato de la columna. Una vez que se valida el tipo de dato de la columna y se cumplen las condiciones para verificar el patrón el proceso continúa con la recopilación de los elementos para la subconsulta, cuyo seudocódigo se presenta enseguida.

Algoritmo 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
29 //subconsulta y se almacena su valor en la variable resultado
```

El algoritmo para recopilar elementos de la subconsulta tiene como función detectar los elementos necesarios para construir la subconsulta en SQL. El algoritmo recorre todos los componentes léxicos en busca de alguna función de agregación que ya haya sido detectada. Una vez que es localizada, se almacena en una variable para posteriormente utilizarla en la construcción de la sentencia. Ya que es un elemento requerido únicamente en la subconsulta, sus etiquetas son modificadas a fin de que la interfaz no lo utilice en la construcción de la consulta en SQL.

Una vez detectada la función de agregación, se procede a localizar la columna interna. Esta columna, al igual que la función de agregación, ya ha sido detectada y etiquetada por el módulo de análisis semántico.

El proceso, al detectar la columna interna, modifica su tipo de frase a "Frase Where". Esto es necesario, ya que, al momento de construir la consulta en SQL, la interfaz la asociará con un valor de búsqueda (resultado de la subconsulta). Una vez localizada la columna interna y la función de agregación, sólo resta obtener la tabla que está incluida en la columna interna mediante el uso del método split (existente en Java). Al tener los tres elementos necesarios, ahora es posible construir la subconsulta y ejecutarla. Finalmente, al obtener el resultado de la subconsulta, el algoritmo termina su proceso, y el algoritmo de asociación de cláusula WHERE puede iniciar su proceso.

Algoritmo 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 SENTENCIA EN SQL CON EL VALOR WHERE QUE
13//REPRESENTA EL RESULTADO DE LA SUBCONSULTA
```

El algoritmo de asociación de cláusula WHERE tiene como finalidad localizar la columna que será usada para asociar el valor obtenido por la subconsulta. El valor obtenido de la ejecución de la subconsulta se integra a la estructura de los componentes léxicos una posición más delante de la columna etiquetada como WHERE. Este mecanismo permite modificar las características del componente léxico para que pueda ser usado por la interfaz en la construcción de la consulta en SQL. De esta forma el resultado de la subconsulta queda como un valor de búsqueda dentro de la estructura que contiene los componentes léxicos, lo cual permite que el proceso de la subconsulta sea transparente para el núcleo de la interfaz.

En esta sección se presenta un ejemplo utilizando el pseudocódigo explicado previamente. De la estructura de datos del análisis semántico, sólo se muestra la información requerida por el módulo para el tratamiento de las subconsultas.

Componente léxico	cuál	es	el	nombre	del	estado	con	el	área	más pequeña
Tipo						Frase Select			Frase Select	Frase Select
Etiqueta de columna						State.state_name			lake.area, state.area	
Etiqueta final						State.state_name			state.area	MIN
Marcado	false	false	false	false	false	true	false	false	true	true

El procesamiento de subconsultas inicia una vez finalizado el tratamiento para funciones de agregación. El proceso inicia recorriendo desde el último componente léxico hasta el primer componente léxico. Al detectar que el componente léxico *más pequeña* tiene como etiqueta final “MIN”, sabemos que se ha localizado una función de agregación. Por lo tanto, la variable *fa* toma el valor de esa etiqueta final, el *contador* aumenta en uno, el componente léxico es marcado como true y su tipo de frase cambia a “SUBCONSULTA”, tal como se muestra enseguida.

fa←"MIN"
contador=0
contador←*contador*+1

Después de este proceso, los valores de la estructura del análisis semántico quedan como se muestra enseguida.

Componente léxico	cuál	es	el	nombre	del	estado	con	el	área	más pequeña
Tipo						Frase Select			Frase Select	SUBCONSULTA
Etiqueta de columna						State.state_name			lake.area, state.area	
Etiqueta final						State.state_name			state.area	MIN
Marcado	false	false	False	false	false	true	false	false	true	true

La segunda condición verifica si es una columna, al comprobar que no lo es, se finaliza esa condición. La tercera condición verifica si el contador es igual a dos, en este caso el contador es igual a uno por lo cual el ciclo continúa.

Al pasar al penúltimo componente léxico, la primera condición no es cumplida; por lo tanto, pasa a la segunda condición. En esta condición se verifica que el componente léxico *área* contenga una etiqueta final, una etiqueta de columna, un tipo de frase "Frase Select" y esté marcada como true. Por lo tanto, se supone que es la columna necesaria para la subconsulta de acuerdo al patrón. La variable *subcolumna* almacena la etiqueta final, *contador* aumenta en uno y el tipo de frase es cambiado a "Frase Where", tal como se muestra enseguida.

subcolumna←"state.area"
contador=1
contador←*contador*+1

Después de este proceso, los valores de la estructura del análisis semántico quedan como se muestra enseguida.

Componente léxico	cuál	es	el	nombre	del	estado	con	el	área	más pequeña
Tipo						Frase Select			Frase Where	SUBCONSULTA
Etiqueta de columna						State.state_name			lake.area, state.area	
Etiqueta final						State.state_name			state.area	MIN
Marcado	false	false	False	false	false	true	false	false	true	true

La tercera condición verifica si la variable *contador* es igual a dos. En este caso ya hemos obtenido la función de agregación y la subcolumna; por lo tanto, *contador* es igual a dos. En tales circunstancias, el ciclo es detenido y se ejecuta el método split (existente en Java), el cual permite obtener la tabla de la variable *subcolumna*, tal como se muestra enseguida.

subtabla←"state"=split(*subcolumna*)

subcolumna="area"

Una vez separada la subtabla de la subcolumna, se construye la subconsulta en SQL con la información recopilada, y el resultado es almacenado en la variable *resultado*.

resultado← SELECT *fa* (*subcolumna*) FROM *subtabla*

resultado← SELECT MIN (area) FROM state

resultado←1100

Al obtener el resultado de la subconsulta, se termina el método de recopilación de elementos para subconsulta.

Una vez finalizada la recopilación de elementos para la subconsulta y haber obtenido el resultado, se aplica el método para asociación de cláusula WHERE. Este método tiene como finalidad asociar el valor de búsqueda obtenido por la subconsulta con

la consulta principal. El proceso inicia con un ciclo que recorre los componentes léxicos de la consulta con el fin de encontrar un componente léxico que satisfaga la condición de tener su tipo de frase = "Frase Where", su etiqueta de columna no este vacía y se encuentre marcado como true. Tal como se muestra enseguida, el componente léxico 8 satisface estas condiciones.

Tipo[Q_8]="Frase Where"

ColumnTag[Q_8]<>" "

Marked[Q_8]=true

Cuando se cumplen las condiciones, se agrega una nueva columna a la estructura de datos del análisis semántico en la siguiente posición del componente léxico actual (Q_8), con las etiquetas que se muestran enseguida; donde *resultado* es la variable que contiene el resultado de la subconsulta, la cual se asigna como etiqueta final.

addToken[Q_9,"resultado_subconsulta"]

setTipoFrase[Q_9,"Frase Where"]

setEtiquetaFinal[Q_9,"*resultado*"]

setMarcado[Q_9, true]

Después de este proceso, los valores de la estructura de datos del análisis semántico quedan como se muestra enseguida.

Componente léxico	cual	es	el	nombre	del	estado	con	el	área	resultado_subconsulta	más pequeña
Tipo						Frase Select			Frase Where	Frase Where	SUBCONSULTA
Etiqueta de columna						State.state_name			lake.area, state.area		
Etiqueta final						State.state_name			state.area	1100	MIN
Marcado	false	false	false	false	false	true	false	false	true	true	true

Una vez finalizado el proceso del método de asociación de cláusula WHERE, el análisis semántico procede normalmente, tomando la información que contiene la estructura modificada y construyendo la sentencia en SQL que permite obtener el resultado final de la consulta.

CAPÍTULO 4: EXPERIMENTACIÓN

4.1 Descripción del escenario de pruebas

A continuación se presentan las características técnicas utilizadas para realizar las pruebas del módulo para el tratamiento de funciones de agregación, agrupamiento y subconsultas. También se describen las características del corpus de pruebas y se detalla su constitución.

4.1.2 Hardware y software

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

Tabla 4.1 Características del hardware

Característica	**Especificación**
Procesador	AMD E-450 @ 1.66 GHz
Memoria	3 GB
Sistema operativo	Windows 8 pro

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

Tabla 4.2 Características del software

Característica	**Especificación**
Entorno	Netbeans IDE 8.0.2
Lenguaje	Java 1.7

4.1.3 Corpus de consultas para prueba

Para la realización de las pruebas del módulo para funciones de agregación, agrupamiento y subconsultas, se elaboró un corpus de consultas a partir de las bases de datos de Geobase y ATIS. El corpus de pruebas está compuesto por al menos 10 consultas por caso de prueba de los problemas resueltos que fueron detectados en la tipificación, dando un total de 100 consultas.

La composición del corpus de pruebas, según cada base de datos, se encuentra representada en la tabla 4.3. La tabla 4.4 muestra la composición del corpus según los casos de prueba tratados. Por último, la tabla 4.5 presenta la composición del corpus según el número de problemas involucrados.

Tabla 4.3 Composición del corpus de pruebas según la base de datos

Base de Datos	**Número de Consultas**	**Porcentaje de Consultas**
ATIS	42	42 %
GEOBASE	58	58 %
TOTAL	100	100 %

Tabla 4.4 Composición del corpus de pruebas según los casos de prueba

Tipos de Problema	**Número de Consultas**
Función de agregación (sin agrupamiento)	47
Agrupamiento	33
Ambigüedad	45
Ambigüedad en columna	16
Elipsis en función de agregación	10
Doble agrupamiento	11
Subconsulta	26

Tabla 4.5 Composición del corpus de pruebas según el número de problemas involucrados

Número de Problemas Involucrados	Número de Consultas	Tipos de Problemas
1	36	FA, agrupamiento y subconsulta.
2	44	FA, ambigüedad en columna, ambigüedad count/sum, agrupamiento, ambigüedad en columna.
3	16	FA, Doble agrupamiento, ambigüedad count/sum, agrupamiento, ambigüedad en columna, elipsis en función de agregación.
4	3	Agrupamiento, ambigüedad count/sum, ambigüedad en columna, doble agrupamiento.

4.2 Resultados

Las pruebas realizadas constan de tres partes: pruebas funcionales, de desempeño y comparativas. Las pruebas funcionales califican si la consulta fue traducida correctamente a su equivalente semántica en SQL. Las pruebas de desempeño se refieren a qué tanto falla el proceso de traducción cuando retorna el resultado; es decir, si el resultado obtenido es la información pedida por el usuario. Finalmente, las pruebas comparativas, como su nombre lo indica, comparan el funcionamiento de C-Phrase con el módulo desarrollado en este proyecto.

Las pruebas funcionales califican si la consulta fue traducida correctamente a su equivalente semántico en SQL. De las 100 consultas que integran el corpus de pruebas, 92

consultas fueron traducidas correctamente a SQL, lo que equivale al 92 %. En algunas consultas que no fueron traducidas correctamente, se presentó un error en la palabra *río*, debido a una heurística implementada en la versión original del analizador semántico para resolver casos en que un componente léxico tiene más de una categoría gramatical; en estas consultas, la palabra *río* es detectada erróneamente como un verbo (*reír*) en lugar de un sustantivo. En las otras consultas que no fueron traducidas correctamente se presenta una elipsis en función de agregación que impide que la consulta contenga el patrón necesario para ser detectado como una subconsulta.

Las pruebas de desempeño evalúan qué tanto falla el proceso de traducción cuando retorna el resultado. Si se descartan las consultas mencionadas anteriormente, entonces la metodología desarrollada en este proyecto para el tratamiento de funciones de agregación, agrupamiento y subconsultas tuvo un desempeño del 100 %.

Para las pruebas comparativas con C-Phrase, no fue posible utilizar todo el corpus: solamente las consultas para la base de datos de Geobase, ya que C-Phrase está configurado sólo para esta base de datos. C-Phrase obtuvo 11 consultas correctas de 58, lo cual corresponde al 18 % de las consultas de Geobase. Sin embargo, los resultados obtenidos por tres consultas se consideran parcialmente correctos debido a que, aunque no proporcionaron el resultado esperado, su resultado implica el esperado. A continuación se presenta un ejemplo para clarificar.
Al ingresar la consulta: ¿Cuántas ciudades hay en el estado de Montana?
C-Phrase arroja como resultado: Greatfalls y Billings.
Respuesta esperada: dos.

La información obtenida por C-Phrase es correcta, pero no es exactamente la información que se pidió, ya que la consulta está pidiendo un número, que es el conteo de las ciudades, no el nombre de ellas.

Por otra parte, se observó que C-Phrase no detecta ni resuelve la elipsis en función de agregación, lo que le afecta en gran medida. En el caso de las consultas con agrupamiento, C-Phrase no detecta la cláusula GROUP BY; por lo tanto, no realiza el agrupamiento de la información solicitada. En lo que respecta a consultas que involucran

subconsultas, sólo logró responder correctamente una consulta de las 10 ingresadas. Tomando en cuenta las consultas respondidas correctamente y las que se consideran parcialmente correctas, el porcentaje de desempeño de C-Phrase es del 24 %. En comparación, para el módulo para tratamiento de funciones de agregación, agrupamiento y subconsultas (producto de este proyecto), el desempeño es del 86 % (sólo con el corpus de Geobase).

CAPÍTULO 5 CONCLUSIONES Y TRABAJOS FUTUROS

5.1 Conclusiones

Este trabajo tuvo como resultado un módulo que da tratamiento a funciones de agregación, agrupamiento y subconsultas. El módulo está integrado dentro de la interfaz de lenguaje natural para bases de datos desarrollada en el ITCM, lo cual aumenta su funcionalidad y mejora su desempeño, ya que antes no respondía este tipo de consultas. Por lo tanto, se concluye que se cumplió con el objetivo general de este proyecto.

Se puede asegurar el cumplimiento del objetivo, tomando en cuenta el desempeño del módulo, el cual fue de un 92 % de consultas correctamente contestadas de un corpus de 100 consultas. En las pruebas funcionales, 2 de 100 consultas utilizadas en las pruebas fueron erróneamente traducidas, debido a causas que no son atribuíbles al módulo desarollado y 6 consultas presentaron un error debido a la presencia de elipsis en función de agregación dentro de subconsultas. Por otra parte, al realizar las pruebas de desempeño, se observa un 100 % de certeza, si se descartan las consultas erróneamente traducidas.

Además, se realizaron pruebas comparativas con C-Phrase utilizando la base de datos de Geobase. C-Phrase obtuvo un 24 % de consultas contestadas correctamente (incluyendo respuestas parcialmente correctas), a diferencia del 86 % de recall obtenido utilizando la interfaz de lenguaje natural para bases del ITCM (sólo con el corpus de Geobase).

Por último, una contribución importante producto del desarrollo de este proyecto fue la tipificación de problemas en funciones de agregación, agrupamiento y subconsultas. Esta tipificación deja un precedente para futuros trabajos, ya que, a partir de dicha tipificación, es posible ir resolviendo los problemas de manera sistemática.

Para concluir, la implementación del módulo para tratamiento de funciones de agregación, agrupamiento y subconsultas mejoró el desempeño de la interfaz de lenguaje natural para bases de datos desarrollada en el ITCM, permitiendo dar una mayor funcionalidad y exactitud al usuario.

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.

ANEXO A CORPUS DE CONSULTAS

En la tabla A.1 se presenta el corpus de consultas de pruebas utilizado durante la experimentación. La tabla se encuentra organizada por tipo de problema y base de datos.

Además, se presentan enseguida los esquemas de las bases de datos utilizadas: Geobase y ATIS. Nota: las consultas marcadas con un asterisco fueron creadas; es decir, no existían en los corpus de consultas originales para estas bases de datos.

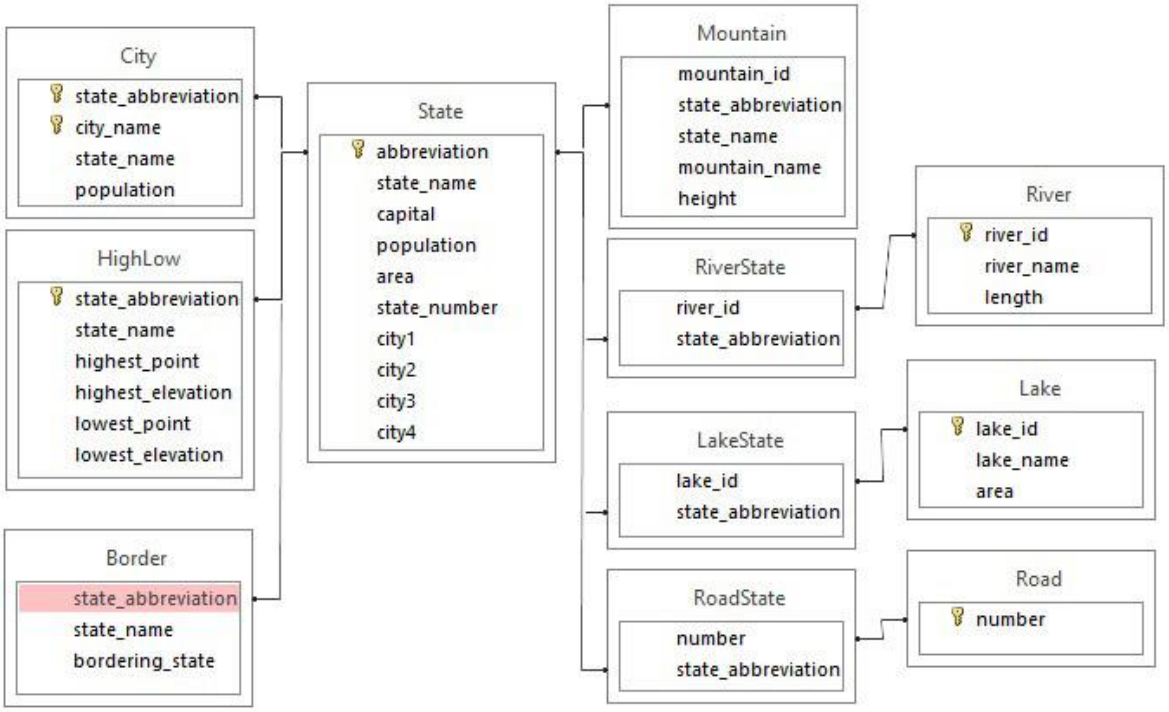

Figura A.1 Esquema de Geobase

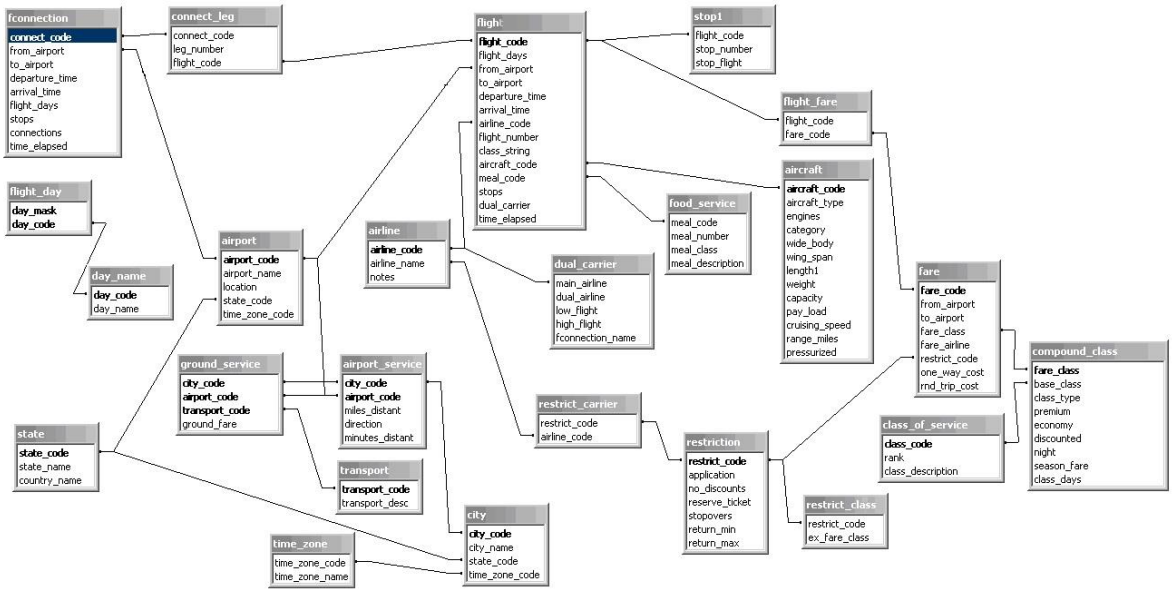

Figura A.2 Esquema de ATIS

Tabla A.1 Corpus de consultas para pruebas

No. de Consulta	Consulta	Consulta en inglés	Base de Datos	Problemas Involucrados
1*	Cuántos ríos corren a través del estado de Texas	How many rivers run through the state of Texas	Geobase	FA, Ambigüedad COUNT/SUM
2*	Combina la población de las ciudades del estado de Alabama	Combine the population of the cities from the state of Alabama	Geobase	FA, Ambigüedad en columna
3*	Suma la altura de las montañas del estado de Washington	Sum the height of the mountains from the state of Washington	Geobase	FA
4*	Cuál es la media de la longitud de los ríos en el estado de Texas	What is the average length of the rivers in the state of Texas	Geobase	FA
5*	Cuál es el promedio de la altura de las montañas del estado de Alabama	What is the average height of the mountains in the state of Alabama	Geobase	FA
6*	Cuál es la longitud del río más largo del estado de Texas	What is the length of the longest river in the state of Texas	Geobase	FA
7	Cuál es el área del estado más grande	what is the area of the	Geobase	FA

		largest state		
8*	Cuál es la longitud del río más pequeño en el estado de Oregon	What is the length of the smallest river in the state of Oregon	Geobase	FA
9	Cuál es el área del estado más pequeño	what is the area of the smallest state	Geobase	FA
10*	Cuál es la longitud del río más grande del estado de Oregon	What is the length of the longest river in the state of Oregon	Geobase	FA
11*	Dame la altura de la montaña más grande del estado de Colorado	Give me the height of the biggest mountain in the state of Colorado	Geobase	FA
12*	Cuántas ciudades hay en el estado de California	How many cities are in the state of California	Geobase	FA, Ambigüedad COUNT/SUM
13*	Cuántos estados tienen colindancia con el estado de Florida	How many states border the state of Florida	Geobase	FA, Ambigüedad COUNT/SUM
14*	Dame la altura de la montaña más grande del estado de Alaska	Give me the height of the biggest mountain in the state of Alaska	Geobase	FA
15*	Cuántos lagos hay en el estado de California	How many lakes are in the state of California	Geobase	FA, Ambigüedad COUNT/SUM
16*	Suma las montañas del estado de	Sum the mountains	Geobase	FA, Elipsis en FA

	Washington	from the state of Washington		
17	Cuál es el río más grande del estado de Oregon	What is the biggest river in the state of Oregon	Geobase	FA, Elipsis en FA, ambigüedad en río
18	Cuál es el estado más pequeño	which state is the smallest	Geobase	FA, Elipsis en FA, Subconsulta
19*	Cuál es la media de los ríos en el estado de Texas	What is the average of the rivers in the state of Texas	Geobase	FA, Elipsis en FA
20*	Cuál es el promedio de las montañas del estado de Colorado	Average of the mountains in the state of Colorado	Geobase	FA, Elipsis en FA
21*	Cuál es el promedio de la longitud de los ríos del estado de Minnesota	Average length of the rivers in the state of Minnesota	Geobase	FA
22	Cuántas ciudades hay en el estado de Montana	How many cities are in the state of Montana	Geobase	FA, Ambigüedad COUNT/SUM
23	Cuántos lagos hay en el estado de New York	How many lakes are in the state of New York	Geobase	FA, Ambigüedad COUNT/SUM
24*	Suma la longitud de los ríos del estado de Iowa	Sum the length of the rivers in the state of Iowa	Geobase	FA
25*	Cuántos ríos tiene el estado de North Dakota	How many rivers are in the state of North Dakota	Geobase	FA, Ambigüedad COUNT/SUM
26	Cuál es la ciudad más grande	What's the largest city	Geobase	FA, Elipsis en FA, Subconsulta

27*	Cuál es el lago más pequeño	What is the smallest lake	Geobase	FA, Elipsis en FA, Subconsulta
28*	Cuál es el lago más grande	What is the largest lake	Geobase	FA, Elipsis en FA, Subconsulta
29	Cuál es la ciudad más pequeña	What is the smallest city	Geobase	FA, Elipsis en FA, Subconsulta
30*	Suma las montañas del estado de Alaska	Sum the mountains from the state of Alaska	Geobase	FA, Elipsis en FA
31*	Cuál es la población de la ciudad más pequeña del estado de Texas	What is the population of the smallest city in the state of Texas	Geobase	FA, Ambigüedad en columna, Subconsulta
32*	Total de ciudades por estado	Total cities by state	Geobase	Agrupamiento, Ambigüedad COUNT/SUM
33*	Cuántas montañas hay por estado	How many mountains there are by state	Geobase	Agrupamiento, Ambigüedad COUNT/SUM
34*	Total de ciudades por población	Total cities by population	Geobase	Agrupamiento, Ambigüedad COUNT/SUM
35*	Total de puntos más altos por estado	Total heighest points by state	Geobase	Agrupamiento, Ambigüedad COUNT/SUM
36*	Total de puntos más bajos por estado	Total lowest points by state	Geobase	Agrupamiento, Ambigüedad COUNT/SUM
37*	Cuántos lagos hay por área	How many lakes are there by area	Geobase	Agrupamiento, Ambigüedad COUNT/SUM
38*	Suma la población de las ciudades por estado	Sum the population of the cities by state	Geobase	Agrupamiento
39*	Cuántos ríos hay	How many	Geobase	Agrupamiento,

	por longitud	rivers are there by length		Ambigüedad COUNT/SUM
40*	Cuántas ciudades hay por población por estado	How many cities by population by state	Geobase	Agrupamiento, Ambigüedad COUNT/SUM, Ambigüedad en columna, Doble agrupamiento
41*	Cuántas montañas hay por estado por altura	How many mountains by state by height	Geobase	Agrupamiento, Ambigüedad COUNT/SUM, Doble agrupamiento
42*	Cuántas ciudades hay por estado	How many cities by state	Geobase	Agrupamiento, Ambigüedad COUNT/SUM
43*	Cuántas montañas hay por altura	How many mountains by height	Geobase	Agrupamiento, Ambigüedad COUNT/SUM
44*	Cuántas ciudades hay por población	How many cities by population	Geobase	Agrupamiento, Ambigüedad COUNT/SUM
45*	Cuántos ríos por longitud	How many states by length	Geobase	Agrupamiento, Ambigüedad COUNT/SUM
46*	Cuántos estados hay por población	How many states by population	Geobase	Agrupamiento, Ambigüedad COUNT/SUM
47*	Cuántos estados hay por población por capital	How many states by population by capital	Geobase	Agrupamiento, Ambigüedad COUNT/SUM, Doble agrupamiento
48*	Cuántos estados hay por población por área	How many states by population by area	Geobase	Agrupamiento, Ambigüedad COUNT/SUM, Doble agrupamiento
49*	Dame el estado con la población más grande	What is the state with the biggest population	Geobase	Subconsulta
50	Cuál es la capital con la población más grande	what is the largest capital in population	Geobase	Subconsulta
51	Dame el estado con el área más pequeña	what is the state with the smallest	Geobase	Subconsulta

		area		
52*	Cuál es la montaña con la altura más pequeña	What is the mountain with the smallest height	Geobase	Subconsulta
53*	Cuál es el estado con la población más pequeña	What is the state with the smallest population	Geobase	Subconsulta
54	Dame la ciudad con la población más pequeña	what is the city with the smallest population	Geobase	Subconsulta
55*	Cuál es el río con la longitud más grande	What is the river with the largest length	Geobase	Subconsulta
56*	Cuál es la ciudad con la población más grande	What is the city with the biggest population	Geobase	Subconsulta
57*	Cuál es la montaña con la altura más grande	What is the mountain with the biggest height	Geobase	Subconsulta
58*	Cuál es el estado con el área más grande	What is the state with the biggest area	Geobase	Subconsulta
59*	Cuántos vuelos hay con origen en ATL y destino en BOS	How many flights are there from ATL to BOS	ATIS	FA, Ambigüedad COUNT/SUM
60*	Cuántas aerolíneas hay	How many airlines are there	ATIS	FA, Ambigüedad COUNT/SUM
61*	Suma el costo del vuelo 148	Sum the cost of the flight 148	ATIS	FA, Ambigüedad en columna
62*	Suma el número de escalas del vuelo 106	Sum the number of stops from flight 106	ATIS	FA, Ambigüedad en columna
63*	Cuál es la tarifa		ATIS	FA, Ambigüedad en

	más cara con origen en ATL y destino en BOS	What is the most expensive fare from ATL to BOS		columna
64*	Cuál es la más alta velocidad de todos los tipos de aeronave	What is the highest speed of all the aircraft types	ATIS	FA
65*	Cuál es la tarifa más barata con origen en ATL y destino en BOS	What is the cheapest fare from ATL to BOS	ATIS	FA, Ambigüedad en columna
66*	Cuál es la más baja velocidad de todos los tipos de aeronave	What is the lowest speed from all the aircraft types	ATIS	FA
67*	Cuál es la media de las tarifas con origen en ATL y destino en BOS	What is the average fare from ATL to BOS	ATIS	FA, Ambigüedad en columna
68*	Promedio de costo del vuelo 148	Average cost from the flight 148	ATIS	FA, Ambigüedad en columna
69*	Cuántos códigos de transporte hay	How many transports codes are there?	ATIS	FA, Ambigüedad COUNT/SUM, Ambigüedad de columna
70*	Suma el tiempo de viaje del vuelo 148	Total time of flight 148	ATIS	FA, Ambigüedad en columna
71*	Cuántos aeropuertos hay	How many airports are there	ATIS	FA, Ambigüedad COUNT/SUM
72*	Cuántos tipos de aeronave hay	How many aircraft types are there	ATIS	FA, Ambigüedad COUNT/SUM
73*	Cuántos códigos de aerolínea hay con	How many airline	ATIS	FA, Ambigüedad COUNT/SUM,

	origen en ATL y destino en BOS	codes are there from ATL to BOS		Ambigüedad en columna
74*	Promedio del peso del tipo de aeronave BOEING 737-300	Average weight of the aircraft type BOEING 737-300	ATIS	FA
75*	Cuántas aerolíneas hay por notas	How many aircraft by notes are there	ATIS	Agrupamiento, Ambigüedad COUNT/SUM
76*	Cuántas velocidades hay por tipo de aeronave	How many speeds by aircraft types are there	ATIS	Agrupamiento, Ambigüedad COUNT/SUM
77*	Cuántos tipos de aeronave hay por peso	How many aircraft types by weight are there	ATIS	Agrupamiento, Ambigüedad COUNT/SUM, Doble agrupamiento
78*	Cuántos aeropuertos hay por ubicación	How many airports by location are there	ATIS	Agrupamiento, Ambigüedad COUNT/SUM
79*	Cuántos tipos de aeronave hay por categoría de aeroplano	How many aircraft types by aircraft category are there	ATIS	Agrupamiento, Ambigüedad COUNT/SUM
80*	Cuántos tipos de aeronave hay por número de motores	How many aircraft types by engines are there	ATIS	Agrupamiento, Ambigüedad COUNT/SUM
81*	Cuántos tipos de aeronave hay por peso por velocidad	How many aircraft types by weight by speed are there	ATIS	Agrupamiento, Ambigüedad COUNT/SUM
82*	Cuántas escalas	How many	ATIS	Agrupamiento,

	hay por conexiones	stops by conecction are there		Ambigüedad COUNT/SUM
83*	Cuántos nombre de estado hay por nombre de país	How many state names are by country	ATIS	Agrupamiento, Ambigüedad COUNT/SUM
84*	Cuántas categorías de aeroplano hay por longitud de vuelo por número de motores	How many categories of aircraft are by range by engines	ATIS	Agrupamiento, Ambigüedad COUNT/SUM, Doble agrupamiento
85*	Cuántos vuelos hay por origen por destino	How many flights are by origin by destination	ATIS	Agrupamiento, Ambigüedad COUNT/SUM, Doble agrupamiento
86*	Cuántos código de aerolínea hay por aerolínea por notas	How many airline codes are by airline by notes	ATIS	Agrupamiento, Ambigüedad COUNT/SUM, Ambigüedad en columna, Doble agrupamiento
87*	Cuántos días hay por origen por destino	How many days are by origin by destination	ATIS	Agrupamiento, Ambigüedad COUNT/SUM, Doble agrupamiento
88*	Cuántos tipos de aeronave hay por peso por número de asientos	How many aircraft types are by weight by capacity	ATIS	Agrupamiento, Ambigüedad COUNT/SUM, Doble agrupamiento
89*	Cuántas llegadas hay por origen por código de aerolínea	How many arrivals are by origin by airline code	ATIS	Agrupamiento, Ambigüedad COUNT/SUM, Ambigüedad en columna, Doble agrupamiento
90*	Cuenta los vuelos por código de aerolínea	Count the flights by airline code	ATIS	Agrupamiento, Ambigüedad COUNT/SUM
91*	Cuál es el tipo de aeronave con el peso más grande	What is the type of aircraft with the	ATIS	Subconsulta

		highest weight		
92*	Cuál es la tarifa de aerolínea con tarifa de viaje redondo más cara	What is the fare of the airline with the most expensive round trip cost	ATIS	Subconsulta
93*	Cuál es la tarifa de aerolínea con tarifa de viaje sencillo más barata	What is the fare of the airline with the cheapest one way cost	ATIS	Subconsulta
94*	Cuál es la categoría de aeroplano con el número de asientos más grande	What is the aircraft category with the biggest capacity	ATIS	Subconsulta
95*	Cuál es el tamaño de equipo con la carga más grande	What is the length with the biggest pay load	ATIS	Subconsulta
96*	Cuál es el vuelo con la salida más temprana	What is the flight with the earliest departure time	ATIS	Subconsulta
97*	Cuál es el tipo de aeronave con la longitud de vuelo más grande	What is the aircrat type with the biggest range	ATIS	Subconsulta
98*	Cuál es el tipo de aeronave con el número de motores más alto	What is the aircraft type with the heighest number of engines	ATIS	Subconsulta
99*	Cuál es el tipo de aeronave con el peso más pequeño	What is the aircraft type with the smallest weight	ATIS	Subconsulta

100*	Cuál es el vuelo con la llegada más tarde	What is the flight with the latest arrival time	ATIS	Subconsulta

ANEXO B CORPUS DE CONSULTAS EN SQL

En la tabla B.1 se presenta el corpus de consultas de pruebas utilizado durante la experimentación. Esta tabla contiene la traducción de cada consulta a SQL y si el resultado fue correcto en las pruebas con C-Phrase y con el módulo desarrollado en esta tesis, en donde ✓ indica un resultado correcto y ✗ indica un resultado erróneo.

Tabla B.1 Corpus de consultas para pruebas con traducción a SQL

No. de Consulta	**Consulta**	**Consulta en SQL**	**Prueba con C-Phrase**	**Prueba con módulo**
1	Cuántos ríos corren a través del estado de Texas	SELECT COUNT (River.river_name) FROM River, State, RiverState WHERE River.river_id = RiverState.river_id AND RiverState.state_abbreviation = State.abbreviation AND State.state_name LIKE 'Texas';	✓	✓
2	Combina la población de las ciudades del estado de Alabama	SELECT SUM (City.population) FROM City, State WHERE City.state_abbreviation = State.abbreviation AND State.state_name LIKE 'Alabama';	✗	✓
3	Suma la altura de las montañas del estado de Washington	SELECT SUM (Mountain.height) FROM Mountain, State WHERE Mountain.state_abbreviation = State.abbreviation AND State.state_name LIKE 'Washington';	✗	✓
4	Cuál es la media de la longitud de los ríos en el estado de Texas	SELECT AVG (River.length) FROM River, State, RiverState WHERE River.river_id = RiverState.river_id AND RiverState.state_abbreviation = State.abbreviation AND State.state_name LIKE 'Texas';	✗	✓

5	Cuál es el promedio de la altura de las montañas del estado de Alabama	SELECT AVG (Mountain.height) FROM Mountain, State WHERE Mountain.state_abbreviation = State.abbreviation AND State.state_name LIKE 'Alabama';	✗	✓
6	Cuál es la longitud del río más largo del estado de Texas	SELECT MAX (River.length) FROM River, State, RiverState WHERE River.river_id = RiverState.river_id AND RiverState.state_abbreviation = State.abbreviation AND State.state_name LIKE 'Texas';	✗	✓
7	Cuál es el área del estado más grande	SELECT MAX (State.area) FROM State;	✗	✓
8	Cuál es la longitud del río más pequeño en el estado de Oregon	SELECT MIN (River.length) FROM River, State, RiverState WHERE River.river_id = RiverState.river_id AND RiverState.state_abbreviation = State.abbreviation AND State.state_name LIKE 'Oregon';	✗	✓
9	Cuál es el área del estado más pequeño	SELECT MIN (State.area) FROM State;	✓	✓
10	Cuál es la longitud del río más grande del estado de Oregon	SELECT MAX (River.length) FROM River, State, RiverState WHERE River.river_id = RiverState.river_id AND RiverState.state_abbreviation = State.abbreviation AND State.state_name LIKE 'Oregon';	✗	✓
11	Dame la altura de la montaña más grande del estado de Colorado	SELECT MAX (Mountain.height) FROM Mountain, State WHERE Mountain.state_abbreviation = State.abbreviation AND State.state_name LIKE 'Colorado';	✗	✓
12	Cuántas ciudades hay en el estado de California	SELECT COUNT (City.city_name) FROM City, State WHERE City.state_abbreviation = State.abbreviation AND State.state_name LIKE 'California';	✗	✓

13	Cuántos estados tienen colindancia con el estado de Florida	SELECT COUNT (State.state_name) FROM State, Border WHERE State.abbreviation = Border.state_abbreviation AND State.state_name LIKE 'Florida';	✗	✓
14	Dame la altura de la montaña más grande del estado de Alaska	SELECT MAX (Mountain.height) FROM Mountain, State WHERE Mountain.state_abbreviation = State.abbreviation AND State.state_name LIKE 'Alaska';	✗	✓
15	Cuántos lagos hay en el estado de California	SELECT COUNT (Lake.lake_name) FROM Lake, State, LakeState WHERE Lake.lake_id = LakeState.lake_id AND LakeState.state_abbreviation = State.abbreviation AND State.state_name LIKE 'California';	Parcial	✓
16	Suma las montañas del estado de Washington	SELECT SUM (Mountain.height) FROM Mountain, State WHERE Mountain.state_abbreviation = State.abbreviation AND State.state_name LIKE 'Washington';	✓	✓
17	Cuál es el río más grande del estado de Oregon	Ambigüedad en la palabra rio de reír y río cuerpo de agua.	✓	✗
18	Cuál es el estado más pequeño	SELECT MIN (State.population) FROM State; SELECT MIN (State.area) FROM State; Debido a elipsis en función de agregación no puede construir la subconsulta.	✓	✗
19	Cuál es la media de los ríos en el estado de Texas	SELECT AVG (River.length) FROM River, State, RiverState WHERE River.river_id = RiverState.river_id AND RiverState.state_abbreviation = State.abbreviation AND State.state_name LIKE 'Texas';	✗	✓
20	Cuál es el promedio de	SELECT AVG (Mountain.height) FROM Mountain, State	✗	✓

	las montañas del estado de Colorado	WHERE Mountain.state_abbreviation = State.abbreviation AND State.state_name LIKE 'Colorado';		
21	Cuál es el promedio de la longitud de los ríos del estado de Minnesota	SELECT AVG (River.length) FROM River, State, RiverState WHERE River.river_id = RiverState.river_id AND RiverState.state_abbreviation = State.abbreviation AND State.state_name LIKE 'Minnesota';	✗	✓
22	Cuántas ciudades hay en el estado de Montana	SELECT COUNT (City.city_name) FROM City, State WHERE City.state_abbreviation = State.abbreviation AND State.state_name LIKE 'Montana';	Parcial	✓
23	Cuántos lagos hay en el estado de New York	SELECT COUNT (Lake.lake_name) FROM Lake, State, LakeState WHERE Lake.lake_id = LakeState.lake_id AND LakeState.state_abbreviation = State.abbreviation AND State.state_name LIKE 'New York';	Parcial	✓
24	Suma la longitud de los ríos del estado de Iowa	SELECT SUM (River.length) FROM River, State, RiverState WHERE River.river_id = RiverState.river_id AND RiverState.state_abbreviation = State.abbreviation AND State.state_name LIKE 'Iowa';	✗	✓
25	Cuántos ríos tiene el estado de North Dakota	SELECT COUNT (River.river_name) FROM River, State, RiverState WHERE River.river_id = RiverState.river_id AND RiverState.state_abbreviation = State.abbreviation AND State.state_name LIKE 'North Dakota';	✓	✓
26	Cuál es la ciudad más grande	SELECT MAX (City.population) FROM City; SELECT MAX (City.area) FROM City; Debido a elipsis en función de	✓	✗

		agregación no puede construir la subconsulta.		
27	Cuál es el lago más pequeño	SELECT MIN (Lake.area) FROM Lake; Debido a elipsis en función de agregación no puede construir la subconsulta.	✗	✗
28	Cuál es el lago más grande	SELECT MAX (Lake.area) FROM Lake; Debido a elipsis en función de agregación no puede construir la subconsulta.	✗	✗
29	Cuál es la ciudad más pequeña	SELECT MIN (City.population) FROM City; SELECT MIN (City.area) FROM City; Debido a elipsis en función de agregación no puede construir la subconsulta.	✓	✗
30	Suma las montañas del estado de Alaska	SELECT SUM (Mountain.height) FROM Mountain, State WHERE Mountain.state_abbreviation = State.abbreviation AND State.state_name LIKE 'Alaska';	✓	✓
31	Cuál es la población de la ciudad más pequeña del estado de Texas	SELECT MIN (City.population) FROM City, State WHERE City.state_abbreviation = State.abbreviation AND State.state_name LIKE 'Texas'; Debido a elipsis en función de agregación no puede construir la subconsulta.	✓	✗
32	Total de ciudades por estado	SELECT COUNT (City.city_name), state_name FROM City GROUP BY City.state_name ;	✗	✓
33	Cuántas montañas hay por estado	SELECT COUNT (Mountain.mountain_name), Mountain.state_name FROM Mountain GROUP BY Mountain.state_name ;	✗	✓
34	Total de ciudades por	SELECT COUNT (City.city_name), City.population	✗	✓

	población	FROM City GROUP BY City.population ;		
35	Total de puntos más altos por estado	SELECT COUNT (HighLow.highest_point), HighLow .state_name FROM HighLow GROUP BY HighLow.state_name ;	✗	✓
36	Total de puntos más bajos por estado	SELECT COUNT (HighLow.lowest_point), HighLow .state_name FROM HighLow GROUP BY HighLow.state_name ;	✗	✓
37	Cuántos lagos hay por área	SELECT COUNT (Lake.lake_name), Lake.area FROM Lake GROUP BY Lake.area ;	✗	✓
38	Suma la población de las ciudades por estado	SELECT SUM (City.population), City.state_name FROM City GROUP BY City.state_name ;	✗	✓
39	Cuántos ríos hay por longitud	SELECT COUNT (River.river_name), River.length FROM River GROUP BY River.length ;	✗	✓
40	Cuántas ciudades hay por población por estado	SELECT COUNT (City.city_name), City.population , City.state_name FROM City GROUP BY City.population, City.state_name ;	✗	✓
41	Cuántas montañas hay por estado por altura	SELECT COUNT (Mountain.mountain_name), Mountain.state_name , Mountain.height FROM Mountain GROUP BY Mountain.state_name, Mountain.height ;	✗	✓
42	Cuántas ciudades hay por estado	SELECT COUNT (City.city_name), City.state_name FROM City GROUP BY City.state_name ;	✗	✓
43	Cuántas montañas hay por altura	SELECT COUNT (Mountain.mountain_name), Mountain.height FROM Mountain GROUP BY Mountain.height ;	✗	✓
44	Cuántas ciudades hay por población	SELECT COUNT (City.city_name), City.population FROM City GROUP BY City.population ;	✗	✓

45	Cuántos ríos por longitud	SELECT COUNT (River.river_name), River.length FROM River GROUP BY River.length ;	✗	✓
46	Cuántos estados hay por población	SELECT COUNT (State.state_name), State.population FROM State GROUP BY State.population ;	✗	✓
47	Cuántos estados hay por población por capital	SELECT COUNT (State.state_name), State.population, State.capital FROM State GROUP BY State.population, State.capital ;	✗	✓
48	Cuántos estados hay por población por área	SELECT COUNT (State.state_name), State.population, State.area FROM State GROUP BY State.population, State.area ;	✗	✓
49	Dame el estado con la población más grande	SELECT State.state_name, State.population FROM State WHERE State.population = (SELECT MAX (State.population) FROM State);	✗	✓
50	Cuál es la capital con la población más grande	SELECT State.capital, State.population FROM State WHERE State.population = (SELECT MAX(State.population) FROM State);	✗	✓
51	Dame el estado con el área más pequeña	SELECT State.state_name, State.area FROM State WHERE State.area = (SELECT MIN(State.area) FROM State);	✓	✓
52	Cuál es la montaña con la altura más pequeña	SELECT Mountain.mountain_name, Mountain.height FROM Mountain WHERE Mountain.height = (SELECT MIN(Mountain.height) FROM Mountain);	✗	✓
53	Cuál es el estado con la población más pequeña	SELECT State.state_name, State.population FROM State WHERE State.population = (SELECT MIN(State.population) FROM State);	✗	✓
54	Dame la	SELECT City.city_name,	✗	✓

	ciudad con la población más pequeña	City.population FROM City WHERE City.population = (SELECT MIN(City.population) FROM City);		
55	Cuál es el río con la longitud más grande	Ambigüedad en la palabra río no permite identificar la columna river_name. Al usar la palabra ríos arroja el valor correcto.	✗	✗
56	Cuál es la ciudad con la población más grande	SELECT City.city_name, City.population FROM City WHERE City.population = (SELECT MAX(City.population) FROM City);	✗	✓
57	Cuál es la montaña con la altura más grande	SELECT Mountain.mountain_name, Mountain.height FROM Mountain WHERE Mountain.height = (SELECT MAX(Mountain.height) FROM Mountain);	✗	✓
58	Cuál es el estado con el área más grande	SELECT State.state_name, State.area FROM State WHERE State.area = (SELECT MAX(State.area) FROM State);	✗	✓
59	Cuántos vuelos hay con origen en ATL y destino en BOS	SELECT COUNT (flight.flight_number) FROM flight WHERE flight.to_airport LIKE 'BOS' AND flight.from_airport LIKE 'ATL';	No lo soporta	✓
60	Cuántas aerolíneas hay	SELECT COUNT (airline.airline_name) FROM airline;	No lo soporta	✓
61	Suma el costo del vuelo 148	SELECT SUM (fare.one_way_cost) FROM fare, flight, flight_fare WHERE fare.fare_code = flight_fare.fare_code AND flight_fare.flight_code = flight.flight_code AND flight.flight_number = 148; SELECT SUM (fare.rnd_trip_cost) FROM fare, flight, flight_fare WHERE fare.fare_code =	No lo soporta	✓

		flight_fare.fare_code AND flight_fare.flight_code = flight.flight_code AND flight.flight_number = 148;		
62	Suma el número de escalas del vuelo 106	SELECT SUM (flight.stops) FROM flight WHERE flight.flight_number = 106;	No lo soporta	✓
63	Cuál es la tarifa más cara con origen en ATL y destino en BOS	SELECT MAX (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.to_airport LIKE 'BOS' AND flight.from_airport LIKE 'ATL'; SELECT MAX (fare.one_way_cost) FROM fare, flight, flight_fare WHERE fare.fare_code = flight_fare.fare_code AND flight_fare.flight_code = flight.flight_code AND flight.to_airport LIKE 'BOS' AND flight.from_airport LIKE 'ATL';	No lo soporta	✓
64	Cuál es la más alta velocidad de todos los tipos de aeronave	SELECT MAX (aircraft.cruising_speed) FROM aircraft;	No lo soporta	✓
65	Cuál es la tarifa más barata con origen en ATL y destino en BOS	SELECT MIN (fare.one_way_cost) FROM fare, flight, flight_fare WHERE fare.fare_code = flight_fare.fare_code AND flight_fare.flight_code = flight.flight_code AND flight.to_airport LIKE 'BOS' AND flight.from_airport LIKE 'ATL'; SELECT MIN (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.to_airport LIKE 'BOS'	No lo soporta	✓

		AND flight.from_airport LIKE 'ATL';		
66	Cuál es la más baja velocidad de todos los tipos de aeronave	SELECT MIN (aircraft.cruising_speed) FROM aircraft;	No lo soporta	✓
67	Cuál es la media de las tarifas con origen en ATL y destino en BOS	SELECT AVG (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.to_airport LIKE 'BOS' AND flight.from_airport LIKE 'ATL'; SELECT AVG (fare.one_way_cost) FROM fare, flight, flight_fare WHERE fare.fare_code = flight_fare.fare_code AND flight_fare.flight_code = flight.flight_code AND flight.to_airport LIKE 'BOS' AND flight.from_airport LIKE 'ATL';	No lo soporta	✓
68	Promedio de costo del vuelo 148	SELECT AVG (fare.one_way_cost) FROM fare, flight, flight_fare WHERE fare.fare_code = flight_fare.fare_code AND flight_fare.flight_code = flight.flight_code AND flight.flight_number = 148; SELECT AVG (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 = 148;	No lo soporta	✓
69	Cuántos códigos de transporte hay	SELECT COUNT (ground_service.transport_code) FROM ground_service;	No lo soporta	✓
70	Suma el tiempo de viaje del vuelo 148	SELECT SUM (flight.time_elapsed) FROM flight WHERE flight.flight_number = 148;	No lo soporta	✓
71	Cuántos	SELECT COUNT	No lo	✓

	aeropuertos hay	(airport.airport_name) FROM airport;	soporta	
72	Cuántos tipos de aeronave hay	SELECT COUNT (aircraft.aircraft_type) FROM aircraft;	No lo soporta	✓
73	Cuántos códigos de aerolínea hay con origen en ATL y destino en BOS	SELECT COUNT (flight.airline_code) FROM flight WHERE flight.to_airport LIKE 'BOS' AND flight.from_airport LIKE 'ATL';	No lo soporta	✓
74	Promedio del peso del tipo de aeronave BOEING 737-300	SELECT AVG (aircraft.weight) FROM aircraft WHERE aircraft.aircraft_type LIKE 'BOEING 737-300';	No lo soporta	✓
75	Cuántas aerolíneas hay por notas	SELECT COUNT (airline.airline_name), airline.notes FROM airline GROUP BY airline.notes ;	No lo soporta	✓
76	Cuántas velocidades hay por tipo de aeronave	SELECT COUNT (aircraft.cruising_speed), aircraft.aircraft_type FROM aircraft GROUP BY aircraft.aircraft_type ;	No lo soporta	✓
77	Cuántos tipos de aeronave hay por peso	SELECT COUNT (aircraft.aircraft_type), aircraft.weight FROM aircraft GROUP BY aircraft.weight ;	No lo soporta	✓
78	Cuántos aeropuertos hay por ubicación	SELECT COUNT (airport.airport_name), airport.location FROM airport GROUP BY airport.location ;	No lo soporta	✓
79	Cuántos tipos de aeronave hay por categoría de aeroplano	SELECT COUNT (aircraft.aircraft_type), aircraft.category FROM aircraft GROUP BY aircraft.category ;	No lo soporta	✓
80	Cuántos tipos de aeronave hay por número de motores	SELECT COUNT (aircraft.aircraft_type), aircraft.engines FROM aircraft GROUP BY aircraft.engines ;	No lo soporta	✓
81	Cuántos tipos de aeronave hay por peso por velocidad	SELECT COUNT (aircraft.aircraft_type), aircraft.weight , aircraft.cruising_speed FROM aircraft	No lo soporta	✓

		GROUP BY aircraft.weight, aircraft.cruising_speed ;		
82	Cuántas escalas hay por conexiones	SELECT COUNT (fconnection.stops), fconnection.connections FROM fconnection GROUP BY fconnection.connections ;	No lo soporta	✓
83	Cuántos nombre de estado hay por nombre de país	SELECT COUNT (state.state_name), state.country_name FROM state GROUP BY state.country_name ;	No lo soporta	✓
84	Cuántas categorías de aeroplano hay por longitud de vuelo por número de motores	SELECT COUNT (aircraft.category), aircraft.range_miles, aircraft.engines FROM aircraft GROUP BY aircraft.range_miles, aircraft.engines;	No lo soporta	✓
85	Cuántos vuelos hay por origen por destino	SELECT COUNT (flight.flight_number), flight.from_airport, flight.to_airport FROM flight GROUP BY flight.from_airport, flight.to_airport;	No lo soporta	✓
86	Cuántos código de aerolínea hay por aerolínea por notas	SELECT COUNT (airline.airline_code), airline.airline_name, airline.notes FROM airline GROUP BY airline.airline_name, airline.notes ;	No lo soporta	✓
87	Cuántos días hay por origen por destino	SELECT COUNT (flight.flight_days), flight.from_airport, flight.to_airport FROM flight GROUP BY flight.from_airport, flight.to_airport ;	No lo soporta	✓
88	Cuántos tipos de aeronave hay por peso por número de asientos	SELECT COUNT (aircraft.aircraft_type), aircraft.weight, aircraft.capacity FROM aircraft GROUP BY aircraft.weight, aircraft.capacity ;	No lo soporta	✓
89	Cuántas llegadas hay por origen por código de aerolínea	SELECT COUNT (flight.arrival_time), flight.from_airport, flight.airline_code FROM flight GROUP BY flight.from_airport, flight.airline_code;	No lo soporta	✓
90	Cuenta los	SELECT COUNT	No lo	✓

	vuelos por código de aerolínea	(flight.flight_number), flight.airline_code FROM flight GROUP BY flight.airline_code ;	soporta	
91	Cuál es el tipo de aeronave con el peso más grande	SELECT aircraft.aircraft_type, aircraft.aircraft.weight FROM aircraft WHERE aircraft.weight = (SELECT MAX(aircraft.weight) FROM aircraft);	No lo soporta	✓
92	Cuál es la tarifa de aerolínea con tarifa de viaje redondo más cara	SELECT fare.fare_airline, fare.rnd_trip_cost FROM fare WHERE fare.rnd_trip_cost = (SELECT MAX(fare.rnd_trip_cost) FROM fare);	No lo soporta	✓
93	Cuál es la tarifa de aerolínea con tarifa de viaje sencillo más barata	SELECT fare.fare_airline, fare.one_way_cost FROM fare WHERE fare.one_way_cost = (SELECT MIN(fare.one_way_cost) FROM fare);	No lo soporta	✓
94	Cuál es la categoría de aeroplano con el número de asientos más grande	SELECT aircraft.category, aircraft.capacity FROM aircraft WHERE aircraft.capacity = (SELECT MAX(aircraft.capacity) FROM aircraft);	No lo soporta	✓
95	Cuál es el tamaño de equipo con la carga más grande	SELECT aircraft.length1, aircraft.pay_load FROM aircraft WHERE aircraft.pay_load = (SELECT MAX(aircraft.pay_load) FROM aircraft);	No lo soporta	✓
96	Cuál es el vuelo con la salida más temprana	SELECT flight.flight_number, flight.departure_time FROM flight WHERE flight.departure_time = (SELECT MIN(flight.departure_time) FROM flight);	No lo soporta	✓
97	Cuál es el tipo de aeronave con la longitud de vuelo más grande	SELECT aircraft.aircraft_type, aircraft.range_miles FROM aircraft WHERE aircraft.range_miles = (SELECT MAX(aircraft.range_miles) FROM aircraft);	No lo soporta	✓
98	Cuál es el	SELECT aircraft.aircraft_type,	No lo	✓

	tipo de aeronave con el número de motores más alto	aircraft.engines FROM aircraft WHERE aircraft.engines = (SELECT MAX(aircraft.engines) FROM aircraft);	soporta	
99	Cuál es el tipo de aeronave con el peso más pequeño	SELECT aircraft.aircraft_type, aircraft.weight FROM aircraft WHERE aircraft.weight = (SELECT MIN(aircraft.weight) FROM aircraft);	No lo soporta	✓
100	Cuál es el vuelo con la llegada más tarde	SELECT flight.flight_number, flight.arrival_time FROM flight WHERE flight.arrival_time = (SELECT MAX(flight.arrival_time) FROM flight);	No lo soporta	✓

GLOSARIO

Interfaz de lenguaje natural: Mecanismos de comunicación entre una persona y una máquina a través del lenguaje natural. En la mayoría de las ocasiones esta comunicación es bidireccional, siendo de tipo pregunta-respuesta [Aguirre, 2014].

Structured query language (SQL): SQL (lenguaje de consultas estructurado) fue desarrollado por IBM. SQL es una combinación de constructores de álgebra relacional y del cálculo racional. Usando SQL se puede definir la estructura de los datos, además de modificar los datos de la base de datos y especificar restricciones de seguridad. Actualmente SQL es el estándar de uso en la inmensa mayoría de los sistemas de gestión de bases de datos comerciales [Aguirre, 2014].

Lenguaje de definición de datos: Permite gestionar las "estructuras" de los objetos [Gabillaud, 2010]:

- Crear, modificar, eliminar objetos.
- Autorizar o prohibir el acceso a los datos.
- Activar o desactivar auditoría.
- Añadir comentarios al diccionario de datos.

Las instrucciones que existen en el lenguaje de definición de datos son: CREATE, ALTER, DROP, GRANT, REVOKE, AUDIT, NOAUDIT, ANALYZE, RENAME, TRUNCATE, COMMENT, FLASHBACK y PURGE.

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 Cd. Madero, Cd. Madero, México, 2014.

[Androutsopoulos, 1993]

I. Androutsopoulos, G. Ritchie y P. Thanisch, "MASQUE/SQL, *An Efficient and Portable Natural Language Query Interface for Relational Databases*", 1993.

[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, México, 2014.

[Date, 1989] C.J. Date, C. White, *A guide to SQL/DS*. Addison-Wesley Publishing Company, 1989.

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

[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.

[Gabillaud, 2010]

J. Gabillaud, O. Heurtel, *Recursos Informáticos Oracle 11g - SQL, PL/SQL, SQL*Plus*, Ediciones ENI, 2010.

[González, 2005] J.J. González, *Traductor de Lenguaje Natural Español a SQL para un Sistema de Consultas a Bases de Datos*, tesis de doctorado, Depto. de Ciencias Computacionales, Centro Nacional de Investigación y Desarrollo Tecnológico, Cuernavaca, México, 2005.

[Mcshane,2004] M. Mcshane, S. Beale y S. Niremburg, "OntoSem Methods for Processing Semantic Ellipsis", *CLS '04 Proceedings of the HLT-NAACL Workshop on Computational Lexical Semantics*. 2004.

[Ortega, 2014] R. Ortega, *Implementación de un administrador de diálogo para una interfaz de lenguaje natural para consultas a bases de datos*, tesis de maestría, Depto. de Posgrado e Investigación, Instituto Tecnológico de Cd. Madero, CD. Madero, México, 2014.

[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. 28

[Quintana, 2008] G. Quintana, M. Márques, J.I. Aliaga, M.J. Aramburu, *Aprende SQL*. Publicaciones Universitat Jaume, 2008.

[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, México, 2009.