INSTITUTO TECNOLÓGICO DE CD. MADERO

División de Estudios de Posgrado e Investigación

TESIS

Desarrollo de un Sistema Administrador de Diálogo para una Interfaz de Lenguaje Natural a Base de Datos.

PARA OBTENER EL TÍTULO DE:

Maestro en Ciencias en Ciencias de la Computación

Presenta:

I. S. C. Julio Alejandro Orta Hernández.

Director:

Dr. Juan Javier González Barbosa.

Codirector:

Dr. José Antonio Martínez Flores.

Cd. Madero, Tamaulipas **Mayo de 2013**

Cd. Madero, Tamps; a **14 de Marzo de 2013.**

OFICIO No.: U5.108/13
AREA: DIVISIÓN DE ESTUDIOS
DE POSGRADO E INVESTIGACIÓN
ASUNTO: AUTORIZACIÓN DE IMPRESIÓN
DE TESIS

C. ING. JULIO ALEJANDRO ORTA HERNÁNDEZ
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 en Ciencias de la Computación, se acordó autorizar la impresión de su tesis titulada:

"DESARROLLO DE UN SISTEMA ADMINISTRADOR DE DIÁLOGO PARA UNA INTERFAZ DE LENGUAJE NATURAL A BASE DE DATOS"

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

A T E N T A M E N T E
"Por mi patria y por mi bien"

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

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

c.c.p.- Archivo
Minuta

MYCHC 'NLZO 'jar

Dedicatoria

A dios, por permitirme llegar a este día y por las personas que ha puesto a mi lado.

A mi mama, por todo su apoyo y dedicación, sin ella no sería la persona que soy ni hubiera llegado hasta aquí.

A mi novia Gabriela, por todo su amor y por ser mi motivo de seguir adelante. Mis sueños no estarían completos sin ti.

A mis hermanas, principalmente a Rosalba y mis sobrinos Angélica, Joaquín y Daniela por su apoyo y por estar conmigo siempre.

A mi tía Andrea y toda su familia porque siempre he podido contar con ustedes.

A mi asesor el Dr. Juan Javier Barbosa por su paciencia y sus consejos.

A toda mi familia y amigos por formar parte de mi vida y por todo su apoyo.

Los amo con todo mi corazón.

Julio Alejandro

Agradecimientos

Mi más sincero agradecimiento a todos mis maestros, principalmente a los de mi comité tutorial: Dra. Laura Cruz Reyes, Dra. María Lucila Morales Rodríguez y Dr. Arturo Hernández Ramírez y a mi subdirector de Tesis Dr. José Antonio Martínez Flores por sus valiosos comentarios, críticas y sugerencias.

A mi asesor el Dr. Juan Javier González Barbosa, por su apoyo, sus consejos y por haberme dirigido en esta Tesis.

A mis compañeros de generación: Victoria, Jesús, Alejandro, Raimundo y Daniel, por los consejos, las experiencias compartidas, por su apoyo y por su amistad a lo largo de la Maestría.

Al Consejo Nacional de Ciencia y Tecnología (Conacyt) y al Instituto Tecnológico de Cd. Madero (ITCM) por la beca que me permitió terminar mis estudios de Maestría con éxito.

Resumen

Hasta la fecha es necesario acceder a la información contenida en una base de datos, las ILNBDs (Interfaces de Lenguaje Natural para Bases de Datos) no garantizan una traducción correcta de la consulta en lenguaje natural.

Cuando existe ambigüedad las ILNBDs tratan de identificar a cual columna o tabla hace referencia el usuario y con el propósito de dar una respuesta aceptable a las consultas incluyen información de más en la respuesta que brindan al usuario.

Este trabajo implementa un administrador de dialogo utilizando el modelo de Frames, que permite a usuario elegir las columnas o tablas de las que desea información.

Por medio de reglas se decide si es necesario un dialogo con el usuario y se le presentan las opciones resultado del análisis de la interfaz, modificando la consulta SQL de acuerdo a la respuesta del usuario.

También es importante resolver consultas que devuelvan el número de registros que cumplan con cierta condición y obtener el máximo o mínimo valor de ciertos campos en una tabla, para esto es necesario el uso de las funciones de agregación “Count”, “Max” y “Min”, en este trabajo se implementó por medio del uso de palabras clave el procesamiento de este tipo de consultas.

Se tomó como base la ILNBD desarrollada en el ITCM, se implementó el administrador de dialogo y el procesamiento de las funciones de agregación “Count”, “Max” y “Min”. Se utilizó un corpus de 50 consultas de la base de datos “Northwind” donde se logró un 2% de mejora en consultas contestadas correctamente, sin embargo se logró un incremento del 32% en consultas contestadas sin información de más y un corpus de 50 consultas de la base de datos “Pubs” en la cual se logró un incremento importante del 22% de mejora en consultas contestadas correctamente y un incremento del 44% en consultas contestadas sin información de más.

Summary

To date it is necessary to access the information in a database, the ILNBDs (Natural Language Interfaces for Databases) do not guarantee a correct translation of the natural language query.

When there is ambiguity the ILNBDs try to identify which column or table refers the user and for the purpose of giving an acceptable response to queries include more information in the response they provide to the user.

This work implements a dialogue manager modeled using Frames, which allows user to choose the columns or tables from which you want information.

By means of rules is decided if necessary a dialogue with the user and is presented with the results of the analysis options of the interface, modifying the SQL query according to the user's response.

It is also important to resolve queries that return the number of records that meet a certain condition and get the maximum or minimum value of certain fields in a table, this requires the use of aggregate functions "Count", "Max" and "min ", this work was implemented through the use of keywords processing such queries.

Was taken as a basis the ILNBD developed in the ITCM, was implemented dialogue manager and processing of aggregate functions "Count", "Max" and "Min". We used a corpus of 50 queries the database of "Northwind" which achieved a 2% improvement in queries answered correctly, however achieved a 32% increase in inquiries answered without more information and a corpus of 50 queries database of "Pubs" in which saw a significant increase of 22% improvement in query answered correctly and 44% increase in inquiries answered without more information.

Índice de contenido de la Tesis.

Índice de imágenes.

Índice de Tablas.

Capítulo 1

1.1.- INTRODUCCIÓN.

El diálogo se define según la Real academia española como una “Plática entre dos o más personas, que alternativamente manifiestan sus ideas o afectos”. El habla es la forma más común y natural de comunicarnos entre los seres humanos, la forma en que expresamos nuestras ideas y realizamos acciones cotidianas como: obtener información sobre algún tema, hacer preguntas, hacer la compra de algún objeto, dar y recibir instrucciones.

Actualmente el uso de la computadora se ha difundido en muchas áreas diferentes a la informática e incluso en tareas cotidianas. Muchas personas necesitan utilizar la computadora, ya sea para hacer oficios, procesar información, enviar correos electrónicos, hacer diseños, etc. En la mayoría de las empresas que venden productos o brindan servicios, toda la información generada de compras y ventas se guarda en una base de datos que puede llegar a ser muy grande dependiendo de la empresa.

Para lograr obtener información de una base de datos, es necesario utilizar un lenguaje formal como SQL (Structured Query Language), que nos permite indicarle a la computadora cuál es la información exacta que queremos obtener de dicha base de datos, sin embargo no todas las personas saben utilizar este lenguaje y se requeriría hacer una capacitación para aprenderlo. Lo ideal sería desarrollar aplicaciones con las que los usuarios se pudieran comunicar utilizando el lenguaje natural.

Por esta razón es muy importante el desarrollar aplicaciones reales que faciliten la interacción Hombre-Máquina, ya sea por medio del lenguaje escrito o hablado. Como obtener información de alguna base de datos o darle instrucciones a la computadora y que esta pueda entender lo que le estamos comunicando.

Un sistema de diálogo puede entenderse como un sistema automático que simula el diálogo entre dos personas, en este caso una persona con una máquina, con el objetivo de indicarle a la computadora la ejecución de una tarea específica, que generalmente es proporcionar algún tipo de información.

Actualmente construir un sistema que pueda mantener una conversación con una persona de manera natural es un gran reto, dada la gran cantidad de información que sería necesaria y las limitaciones que se tienen para obtener la información que el usuario necesita. Sin embargo los avances en las investigaciones en sistemas de diálogo, han sido capaces de interactuar con cierto grado de naturalidad, es decir que la computadora también tome la iniciativa en el diálogo, aunque siempre en un dominio o tarea muy específica.

La interfaz del sistema de diálogo puede ser hablada o escrita, dependiendo el entorno en el que será aplicado, como en venta de boletos de avión, consulta de calificaciones de alumnos en una escuela, en sistemas de información meteorológicas, guías de un museo, atención telefónica, servicios bancarios, etc.

En este trabajo se implementará un administrador de diálogo a una interfaz de lenguaje natural en español, que servirá para mejorar el rendimiento de dicha interfaz, iniciando un diálogo cuando la consulta del usuario en lenguaje natural no este del todo clara o cuando se necesite mayor información para generar una mejor respuesta.

En general el sistema solo mostrará un diálogo solo cuando sea necesario, estos casos se explicaran en el capitulo 4 y el capitulo 5 de esta tesis.

1.2.- Antecedentes del problema.

La línea de investigación de este trabajo es la de desarrollar una interfaz de leguaje natural a una base de datos.

Este trabajo se basa en la interfaz desarrollada por el doctor Juan Javier González Barbosa [González J.J., 2005] en su tesis doctoral “Traductor de Lenguaje Natural Español a SQL para un Sistema de Consultas a Bases de Datos” de CENIDET en el año 2005.

Para realizar esta interfaz se tomaron en cuenta algunos trabajos anteriores del ITCM, que fueron una aportación para la construcción de la interfaz, así como también esta interfaz ha generado algunos otros trabajos posteriores para mejorar su rendimiento, como es el caso de esta tesis.

A continuación se mencionarán dichos trabajos de tesis y su principal aportación a la interfaz.

Uno de los primeros trabajos fue la tesis “Construcción de un Preprocesador de Consultas en Lenguaje Natural a una Base de Datos” en noviembre del 2004 [Mendoza, 2002]. El objetivo de esta tesis fue la construcción de un diccionario de dominio, para lo cual se implementó un preprocesador de consultas. Se construyó un diccionario de Metadatos que es generado automáticamente de acuerdo a la base de datos utilizada y un diccionario de sinónimos general para que pudiera ser utilizado en cualquier base de datos. Este preprocesador analiza la consulta en lenguaje natural hecha por el usuario y la etiqueta con información léxica, semántica y sintáctica. La interfaz del ITCM [González J.J., 2005] utiliza la misma arquitectura y el código del preprocesador desarrollado en esta tesis.

Otro trabajo relacionado es la tesis de maestría de Myriam Janeth Rodríguez Martínez, “Diseño de una Técnica para la Traducción de Consultas con Información Implícita a una Base de Datos”, en Octubre de 2005 [Rodríguez, 2005]. Esta tesis

utiliza la clasificación de las consultas del corpus. A partir de esta tesis se consideró la relación de las consultas implícitas que no eran parte de los objetivos de la interfaz del ITCM, pero que dada su importancia y los avances en el diseño de la técnica para resolver estas consultas se agregaron a la tesis doctoral. El diseño desarrollado en esta tesis modela la estructura de una consulta hecha en lenguaje natural utilizando un grafo semántico de la base de datos. También debido a que los valores de tipo cadena aportan información importante para traducir las consultas, utiliza un diccionario de valores.

A finales de ese año el alumno Omar Ulises Silva Domínguez, desarrolló la tesis de maestría “Implementación de un Traductor de Lenguaje Natural Independiente del Dominio”, en Diciembre del 2005 [Silva, 2005]. El objetivo de esta tesis fue la de implementar un traductor de consultas hechas en lenguaje natural en el idioma español a una consulta SQL, de manera independiente del dominio. La arquitectura de esta tesis está compuesta de dos grandes módulos que son el preprocesador y la técnica de traducción, se realizó de esta manera para separar los datos que son forzosamente dependientes del dominio, de los datos que pueden ser procesados de manera general, para lograr la independencia del dominio. Este traductor en el primero en la teoría para el tratamiento de la preposición “de” y de la conjunción “y”, utilizando teoría de conjuntos, realiza una intersección para las preposiciones y una unión para procesar las conjunciones. También incluye la identificación de referencias a valores que no se especifican explícitamente, cuando estos valores se encuentran en la parte de la consulta correspondiente a la cláusula WHERE.

“Estrategia de Mejora para la Independencia del Dominio en una Interfaz de Lenguaje Natural” Tesis de Maestría realizada por la alumna Arodit Margarita González Michel, Enero de 2011 [González A., 2011]. El objetivo de este trabajo es aumentar el porcentaje de éxito en las consultas que se realizan en la ILNDB del ITCM, se revisa el tratamiento de la preposición “de” donde se utiliza la información de la cláusula “WHERE” para dar solución a la petición del usuario.

Este trabajo tomará la interfaz desarrolla en el CENIDET por el Dr. Juan Javier González Barbosa [González J.J., 2005] en su tesis de doctorado, y le implementará un gestor de diálogo para mejorar el porcentaje de éxito en las consultas de los usuarios, cuando la información proporcionada por el usuario no sea suficiente o totalmente clara, se iniciará el diálogo para obtener esa información faltante del usuario que utiliza la interfaz.

1.3.- Planteamiento del problema.

Actualmente la interfaz genera las respuestas sin usar un administrador de diálogo, pero en muchos casos devuelve información adicional que no es requerida por el usuario y que puede hacer confusa dicha respuesta.

Esta información adicional puede estar en el número de columnas que regresa en la consulta, es decir en la parte de la cláusula "**SELECT**", o en las condiciones de dicha consulta, es decir en la parte de la cláusula "**WHERE**".

En este trabajo se abordaran las 3 siguientes problemáticas.

1) Columnas de más en la cláusula "**SELECT**".

2) Columnas de más en la cláusula "**WHERE**".

3) Consultas que utilicen las funciones de agregación "**COUNT**", "**MAX**" o "**MIN**".

Consultas con información de más en la cláusula "SELECT".

En éste tipo de consultas la problemática principal que se aborda es cuando en la respuesta dada por la interfaz se incluyen una o varias columnas que no son necesarias, puesto que no son requeridas por el usuario.

Por ejemplo, se tiene la siguiente consulta de entrada: ***Muéstrame todos los nombres del producto cuya categoría es beverages.***

Respuesta dada por la interfaz original:

```
SELECT Products.ProductName, Categories.CategoryID,
Categories.CategoryName
FROM Categories, Products
WHERE ((Categories.CategoryID LIKE '%beverages%' OR
Products.CategoryID LIKE '%beverages%' OR
Categories.CategoryName LIKE '%beverages%'))
AND Products.CategoryID = Categories.CategoryID
```

Al realizar el análisis de la consulta [***Ver detalle en la sección 4.1***], como se puede observar en el ejemplo se identifican **3** campos en la cláusula "**SELECT**".

En el caso de ejemplo, la respuesta debería estar formada por solo una columna que es: "**Products.ProductName**", por lo que es necesario un diálogo con el usuario para que elija cuál es la columna o columnas que le interesa ver en la respuesta.

La consulta después del diálogo quedaría de la siguiente forma:

```
SELECT Products.ProductName
FROM Categories, Products
WHERE ((Categories.CategoryID LIKE '%beverages%' OR
Products.CategoryID LIKE '%beverages%' OR
Categories.CategoryName LIKE '%beverages%'))
AND Products.CategoryID = Categories.CategoryIDAND
Products.CategoryID = Categories.CategoryID
```

Consultas con información de más en la cláusula "WHERE".

En este tipo de consultas la problemática que se aborda es cuando en la respuesta dada por la interfaz el valor que es proporcionado por el usuario se iguala a distintas columnas para formar la condición, lo que puede ocasionar que se obtengan registros de más que no son requeridos por el usuario.

Por ejemplo, se tiene la siguiente consulta de entrada: ***Muéstrame el empleado con identificador H-B39728F.***

Respuesta dada por la interfaz original:

```
SELECT employee.emp_id, employee.fname, employee.lname
FROM employee
WHERE ((employee.emp_id LIKE '%H-B39728F%'
OR employee.job_id LIKE '%H-B39728F%'
OR employee.pub_id LIKE '%H-B39728F%'))
```

Al realizar el análisis de la consulta [***Ver detalle en la sección 4.1.7***], como se puede observar en el ejemplo se identifican 3 campos en la cláusula "**WHERE**".

En el caso de ejemplo la condición debería estar formada por el valor "**H-B39728F**" igualado a una sola columna que es: "**employee.emp_id**", por lo que por lo que es necesario un diálogo con el usuario para que elija a que columna se refiere con ese valor.

La consulta después del diálogo quedaría de la siguiente forma:

```
SELECT employee.emp_id, employee.fname, employee.lname
FROM employee
WHERE ((employee.emp_id LIKE '%H-B39728F%'))
```

Consultas que utilicen las funciones de agregación "COUNT", "MAX" y "MIN".

Actualmente la interfaz no responde correctamente ninguna consulta que utilice alguna función de agregación. Se agregará a la interfaz la funcionalidad de analizar y responder a consultas que utilicen las funciones de agregación "**COUNT**", "**MAX**" y "**MIN**".

En consultas que utilicen la función de agregación "**COUNT**", se requiere que se devuelva el número de registros que cumplen con determinada condición.

En consultas que utilicen la función de agregación "**MAX**" o "**MIN**", se requiere que se devuelva el valor mayor o menor de una columna, de un grupo de registros que cumplan con una determinada condición.

Por ejemplo, se tiene la siguiente consulta de entrada: ***dame el número de empleados de la editorial Scotney book***.

Respuesta dada por la interfaz:

```
SELECT employee.emp_id, employee.fname, employee.lname,
publishers.pub_id, publishers.pub_name
FROM employee, publishers
WHERE ((publishers.pub_name LIKE 'Scotney') OR
(publishers.pub_name LIKE 'book')) AND employee.pub_id =
publishers.pub_id
```

Al realizar el análisis de la consulta [***Ver detalle en la sección 4.1***], como se puede observar en el ejemplo se identifican 5 campos en la cláusula "**SELECT**".

En la interfaz original no esta implementado el tratamiento de consultas que utilicen funciones de agregación y se implementará en este trabajo. En este caso de ejemplo la interfaz devuelve 5 campos en la cláusula "**SELECT**" y la función de agregación "**COUNT**" solo debe aplicarse a una sola columna que es: "**employee.emp_id**", por lo que se en necesario un diálogo con el usuario para que elija a que columna se aplicará la función "**COUNT**".

La consulta quedaría de la siguiente forma:

```
SELECT COUNT(employee.emp_id)
FROM employee, publishers
WHERE ((publishers.pub_name LIKE 'Scotney') OR
(publishers.pub_name LIKE 'book')) AND employee.pub_id =
publishers.pub_id
```

1.4.- Objetivos.

Objetivo General.

Mejorar el desempeño de una ILNBD implementada en el ITCM [González J.J., 2005] mediante la creación de un Administrador de Diálogo.

Objetivos específicos.

1.- Implementar un Administrador de Diálogo que permita la identificación de la información requerida para mejorar la respuesta a la consulta del usuario.

2.- Implementar las funciones de agregación (count, max y min) al traductor de Lenguaje Natural.

1.5.- Limitaciones del proyecto.

- Las bases de datos utilizadas serán la base de datos Northwind [*Ver anexo A*] y la base de datos Pubs [*Ver anexo B*].
- Los corpus que se van a utilizar son los que se utilizaron para probar la interfaz desarrollada por el Doctor Javier Gonzalez Barbosa [González J.J., 2005].
- El trabajo de esta tesis abordará el problema de cuando la interfaz devuelve columnas innecesarias, permitiendo al usuario por medio de un diálogo elegir las columnas que desea que aparezcan en la respuesta.
- Se agregará a la interfaz el uso de la función “Count”, “Max” y “Min”.
- No se abarcarán otras funciones de agregación SQL, ni la sentencia “GroupBy”.
- Solo se utilizarán los corpus que ya se han utilizado para probar otras interfaces desarrolladas por alumnos del ITCM.

1.6.- Justificación.

Actualmente la mayoría de las empresas utiliza grandes bases de datos para guardar información importante como la de sus clientes, sus proveedores, etc. La mayoría de los empleados, no tiene conocimientos avanzados de computación, ni sabe de comandos SQL para formular las consultas de la información que necesita, por esa razón en muy importante desarrollar interfaces que le permitan, utilizando el lenguaje natural obtener información de las bases de datos.

Tener la base para desarrollar un sistema de diálogo de lenguaje natural que permita a los usuarios comunicarse y obtener datos de fuentes de información como bases de datos de una manera más sencilla al darle la oportunidad de utilizar la forma de comunicación más común que es el lenguaje hablado de manera natural.

1.7 Organización de la Tesis

Capítulo 2: En este capítulo se describen los conceptos relacionados con esta tesis como lenguaje, base de datos, interfaz de lenguaje natural a base de datos, lenguaje de consultas estructurado y generación de lenguaje natural. Así como la definición y clasificación de un gestor de diálogo y las principales funciones de agregación.

Capítulo 3: Se presenta una investigación de los principales trabajos relacionados a este proyecto, es decir las interfaces de lenguaje natural a bases de datos, describiendo su funcionamiento y realizando una comparativa con el proyecto de esta Tesis.

Capítulo 4: En este capítulo se presenta la propuesta de solución, donde se explica el funcionamiento de la interfaz original, se presenta el modelo de frames utilizada para desarrollar este proyecto, las reglas para decidir cuando se interactúa con el usuario por medio de un dialogo y las palabras clave para identificar cuando una consulta utilizará la función de agregación “Count”, “Max” o “Min”.

Capítulo 5: En este capítulo se describe la implementación de la interfaz desarrollada en esta tesis, las modificaciones que se realizaron para que la interfaz funciones en el sistema operativo Windows 7 de 64 bits y se describe como se implementó cada una de las reglas.

Capítulo 6: En este capítulo se presentan los resultados de la experimentación realizada comparando los resultados de la interfaz desarrollada en esta Tesis con los resultados de la tesis realizada por [González A., 2011], las conclusiones obtenidas y los trabajos futuros que pueden abordarse.

Capítulo 2
Marco teórico

2.1 Lenguaje.

Según la real academia española, el lenguaje puede definirse como un conjunto de sonidos articulados con el que el hombre manifiesta lo que piensa y siente, es decir, la manera de expresarse de una persona en particular.

Existen dos tipos de lenguajes bien definidos [Rojas J.C., 2009] que son los lenguajes naturales, que es la forma oral en que se comunican comúnmente las personas y los lenguajes formales, que es un modo de expresión más cuidadoso y preciso que se utiliza para expresarse en un área del conocimiento específico, como podría ser el lenguaje matemático.

2.2 Base de datos.

Una base de datos (**BD**) es un conjunto de datos relacionados entre sí, que tienen un significado implícito.

Un sistema de gestión de bases de datos (**SGBD**, en inglés, database management system: **DMBS**) es un conjunto de programas que permite a los usuarios crear y mantener una base de datos, es decir la construcción, manipulación y consultas en una base de datos.

2.3 Interfaz de lenguaje natural.

Una interfaz de lenguaje natural es un software que nos permite interactuar con una máquina por medio del lenguaje natural. Por lo general, la comunicación se produce en ambos sentidos en forma de pregunta-respuesta.

2.4 Interfaz de Lenguaje Natural a Bases de Datos (ILNBDs).

Una interfaz de lenguaje natural a bases de datos es un sistema que permite que un usuario acceda a la información almacenada en una base de datos, mediante una solicitud expresada en algún tipo de lenguaje natural. Por ejemplo una solicitud en el idioma español o en el idioma inglés [Androutsopoulos et al, 1995].

Este software sirve para hacer mucho más sencilla la interacción entre el usuario y la máquina, de tal forma que el usuario puede realizar una consulta a la base de datos, utilizando el lenguaje natural que utiliza normalmente para comunicarse con otras personas y esta interfaz se encargará de traducir esta expresión a una consulta formal en el lenguaje SQL, dicha consulta se enviará al administrador de la base de datos, que regresará el resultando de la consulta, y se mostrará al usuario como respuesta a su petición realizada en lenguaje natural.

En la forma más sencilla una ILNDB funciona de la siguiente manera:

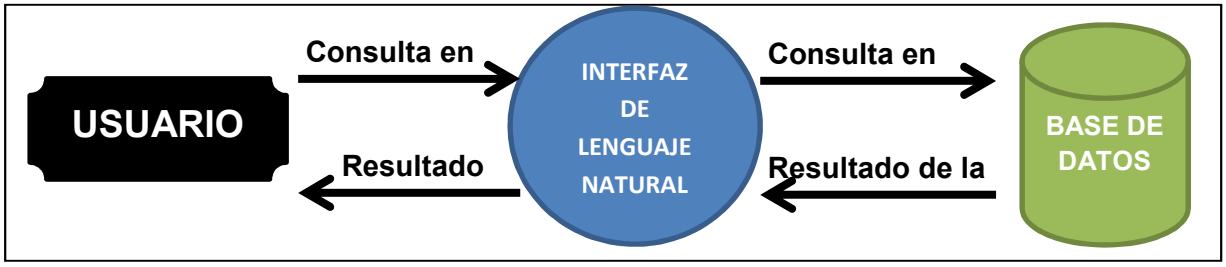


Figura 1.- Flujo de una ILNDB

Existen diversas interfaces que nos permiten acceder a la información almacenada en una base de datos. Los tipos de interfaces más comúnmente utilizadas son las siguientes:

- **Lenguajes Formales de recuperación:** El usuario debe conocer algún lenguaje formal como SQL, para poder hacer sus peticiones a la base de datos, utilizando palabras reservadas para realizar sus consultas.

- **Interfaces gráficas y menús de selección:** El usuario es guiado por menús que le indican cómo formular sus consultas.

- **Interfaces basadas en formularios:** El usuario llena campos de respuestas preestablecidos para realizar su petición a la base de datos.

- **Interfaces de lenguaje natural:** El usuario interactúa con la computadora por medio de un lenguaje natural, ya sea de forma oral o escrita.

2.5 Structured Query Language (SQL).

SQL ó Lenguaje de consultas estructurado, fue desarrollado por IBM, originalmente denominado SEQUEL, como parte del proyecto System R a principios de 1970. Hoy en día numerosos productos son compatibles con el lenguaje SQL, y se ha establecido como el lenguaje estándar para las bases de datos relacionales. La versión más reciente publicada por la ANSI (American National Standards Institute) es SQL: 2008.

SQL es una combinación de constructores del álgebra relacional y del cálculo relacional. Usando SQL es posible, además de definir la estructura de los datos, modificar los datos de la base de datos y especificar restricciones de seguridad [Silberschatz et al, 2006].

En la actualidad el lenguaje SQL, es el estándar mayormente utilizado en los sistemas gestores de bases de datos (SGBD) comerciales. Y aunque existen gran

número de adiciones particulares al lenguaje SQL para distintas implementaciones comerciales, el soporte al estándar SQL-92 es general y muy amplio.

2.6 Generación de Lenguaje natural.

La *Generación de lenguaje natural* (NLG) es el proceso de creación automática de lenguaje natural teniendo como base una representación de la información no lingüística. Por ejemplo la información almacenada en una base de datos.

En general el proceso de generación de lenguaje se puede dividir en tres principales etapas: la planificación del documento, la micro planificación, y la creación de la superficie [Reiter y Dale, 2000]. A continuación se muestra un resumen de las tareas de generación que se realizan en cada etapa.

1.- Planificación del documento.- En esta etapa se toma la petición del usuario en leguaje natural y se transforma en la estructura formal que maneja el sistema de diálogo, y en base a eso se determina la estructura del mensaje de respuesta que se dará al usuario.

2.- Micro planificación.- En esta etapa se agrupan los mensajes en una sola oración, y se decide las palabras específicas del dominio y el orden en que se utilizarán para armar el mensaje, así como el uso de pronombres.

3.- Creación de la superficie. En esta última etapa se verifica la concordancia de las palabras y se aplica el formato de salida que se requiere para ser visualizado por el usuario, por ejemplo código HTML o código en LÁTEX.

2.7 Gestor de diálogo.

El gestor de diálogo es la parte central de un sistema de diálogo pues es el que se encarga de controlar el flujo del diálogo y decidir cuáles son las acciones que se deben llevar a cabo para responder a las intervenciones del usuario en la

conversación. Este interactúa tanto con los componentes que procesan las entradas del sistema como con los que generan la salida del mismo.

El gestor de diálogo debe conocer la semántica utilizada por el usuario en el contexto del diálogo, lograr detectar los errores en los procesos de reconocimiento y comprensión y corregirlos, obtener la información de la fuente de datos, como podría ser una base de datos y generar la respuesta apropiada al contexto actual del diálogo.

Para realizar estas acciones el gestor de diálogo debe guardar el historial de los mensajes generados en todos los turnos de diálogo tanto por el usuario como por la computadora y de alguna herramienta que le ayude conocer el estado actual del diálogo.

Un aspecto importante para la interacción Hombre-Máquina es la comprensión automática del habla, es decir que la computadora pueda entender lo que expresa una persona de manera natural. Para lograr esto se utilizan principalmente dos métodos: Que son los métodos basados en reglas y el desarrollo de metodologías estadísticas.

Actualmente ambos métodos permiten desarrollar aplicaciones capaces de comprender las intervenciones de una persona de manera natural en el diálogo, sin embargo para realizar la gestión de diálogo utilizar estos métodos estadísticos no ha dado buenos resultados debido principalmente a la ausencia de corpus de talla suficientemente grandes. Los sistemas que actualmente realizan la gestión de diálogo se basan en la aplicación de reglas o bien son sistemas mixtos que combinan ambos tipos de métodos.

2.8 Clasificación de Gestores de diálogo.

Los gestores de diálogo tomando en cuenta como representan y realizan el flujo de diálogo se pueden clasificar en cuatro grupos [Allen et al, 2001] [McTear, 2004]. A continuación se mostrarán los diferentes grupos y se describirá su funcionamiento.

SISTEMA DE ESTADOS FINITOS.

En estos sistemas la representación del diálogo se hace por medio de transiciones entre estado en forma de una red, los nodos representan las preguntas que el usuario realiza al sistema y las transiciones representarían los caminos que se pueden seguir en el diálogo. Por lo tanto el diálogo con el usuario se realiza de forma estructurada y el flujo del diálogo puede determinarse con anterioridad. El gestor se desplaza por los estados en donde se decide qué información va a mostrar el sistema para obtener la información que se necesita del usuario para realizar determinada tarea.

Este tipo de gestor se puede utilizar cuando la tarea sea muy sencilla, el diálogo este estructurado y existe un número pequeño de respuestas. Su principal ventaja es la simplicidad de su implementación, sin embargo sus principales desventajas son que no se pueden desarrollar diálogos complejos por su falta de flexibilidad. Los usuarios deben seguir los caminos de diálogo establecido por los estados sin la posibilidad de desviarse.

La creación del gestor de diálogo es una tarea sencilla sin embargo requiere una labor intensiva para determinar los estados en cada dominio, así como la detección y corrección de errores conforme se van detectando al realizar las pruebas, puesto que el flujo de diálogo se realiza de una forma manual. Para realizar el análisis sintáctico se utilizan reglas gramaticales y diversas máquinas de estados, verificando lenguajes regulares.

SISTEMAS BASADOS EN FRAMES.

Un frame puede considerarse como una abstracción de un concepto, que contiene los atributos de dicho concepto y los valores correspondientes a cada atributo.

Este tipo de gestores de diálogo tienen una mayor flexibilidad que los gestores de estados finitos. Tanto los gestores de estado finitos como los basados en frames gestionan una tarea obteniendo un conjunto de datos al usuario para obtener la información de la fuente de datos externa, que comúnmente sería una base de datos.

La diferencia radica en que los sistemas basados en frames no debe seguir un orden establecido para obtener toda la información necesaria, de manera que tanto el sistema como el usuario pueden tomar la iniciativa en el diálogo.

Para que el sistema gestor pueda tener este nivel de flexibilidad debe contar con los siguientes tres componentes.

- Un frame para hacer referencia a los conceptos y atributos definidos para la tarea.
- Una gramática o modelo del lenguaje para tener un más extenso reconocimiento de la información que proporciona el usuario.
- Un algoritmo de control del diálogo que determine las acciones que se realizarán en el sistema basándose en los contenidos del frame.

Existe una variante de los frame denominada E-Form (Electronic form), donde además de la información de los atributos se incluyen preferencias del usuario en forma de valores que deciden cuál concepto tiene más prioridad.

SISTEMAS BASADOS EN PLANES.

Los sistemas basados en planes toman como base que toda comunicación entre seres humanos siempre busca un objetivo. Cada interacción entre el usuario y la

computadora se representan como actos de diálogo, que buscan alcanzar esa meta [Searle, 1969]. El argumento de los gestores de diálogo basado en planes indica que cada acto de diálogo forma parte de un plan para lograr un objetivo global [Allen y Perault, 1980], [Appelt, 1985], [Cohen y Levesque, 1990], que el sistema tiene que identificar y actuar de la manera más adecuada para lograr dicho objetivo. El principal inconveniente de este tipo de gestores es que la tarea de identificar el objetivo global del diálogo puede llegar a ser muy compleja. Un ejemplo de este tipo de gestores de diálogo es la plataforma ATLAS.

SISTEMAS BASADOS EN AGENTES.

En este tipo de sistemas el gestor de diálogo lleva a cabo un razonamiento para determinar el flujo de diálogo, es decir que se dará como respuesta al usuario en cada intervención. Para lograr este razonamiento los agentes utilizan técnicas de inteligencia artificial y el modelado del diálogo puede representarse como una colaboración entre varios agentes para resolver una determinada tarea. Este tipo de gestores se puede utilizar en tareas complejas como resolución de problemas. La comunicación en este tipo de gestores puede considerarse como la colaboración de varios agentes donde cada uno tiene su propio conjunto de acciones a realizar, entre estas acciones pueden estar la detección de errores, definir una estructura para representar el historial del diálogo y permitir que tanto el usuario como el sistema tomen la iniciativa en la conversación. Algunos ejemplos del empleo de esta técnica esta la arquitectura Revenclaw, Queen's Communicator y JASPIS.

2.9 Funciones de agregación en SQL.

Una problemática que se abordará en esta tesis es el resolver consultas que incluyan funciones de agregación. A continuación se mencionan las principales funciones de agregación y se brinda una descripción de cada una de ellas.

Las funciones de agregación son funciones que toman una colección de valores como entrada y producen un único valor de salida. SQL proporciona cinco funciones de agregación primitiva que son las siguientes:

Función.	Descripción.
COUNT	Nos da el número total de filas seleccionadas.
SUM	Suma los valores de una columna.
MIN	Nos da el valor mínimo de una columna.
MAX	Nos da el valor máximo de una columna.
AVG	Calcula el valor medio de una columna.

Tabla 1.- Funciones de agregación primitivas en SQL.

Función de agregación “Max” y “Min”.

La función MAX, devuelve el valor mayor de la columna que se pasa como parámetro. La función MIN, devuelve el valor menor de la columna

La sintaxis es:

```
SELECT MAX / MIN ("nombre_columna")
FROM "nombre_Tabla"
```

Función de agregación “Count”.

La función COUNT permite contar el número de filas en una tabla determinada.

La sintaxis es:

```
SELECT COUNT("nombre_columna")
FROM "nombre_Tabla"
```

Función de agregación “GroupBy”

Dentro de las funciones de agregación esta la función “Sum” que nos podría servir por ejemplo para calcular las ventas totales para todos los negocios. Pero que función se utilizaría para calcular el total de ventas para cada negocio.

Entonces, se necesitan realizar dos cosas: Primero, asegurarnos de que hayamos seleccionado el nombre del negocio así como también las ventas totales. Segundo, asegurarnos de que todas las sumas de las ventas estén GROUP BY (Agrupadas por la columna) negocios.

La sintaxis SQL correspondiente es:

```
SELECT "nombre1_columna", SUM("nombre2_columna")
FROM "nombre_tabla"
GROUP BY "nombre1_columna"
```

Consideremos que tenemos la siguiente tabla.

store_name	Sales	Date
Los Ángeles	1500 €	05-Jan-1999
San Diego	250 €	07-Jan-1999
Los Ángeles	300 €	08-Jan-1999
Boston	700 €	08-Jan-1999

Tabla 2.- Ejemplo de tabla de datos.

Por ejemplo, si deseamos encontrar el número de registros en nuestra tabla, ingresamos:

```
SELECT COUNT(store_name)
FROM Store_Information
```

El resultado será: 4

Si deseamos saber las ventas totales para cada negocio, deberíamos ingresar lo siguiente:

```
SELECT store_name, SUM(Sales)
FROM Store_Information
GROUP BY store_name
```

El resultado sería:

store_name	SUM(Sales)
Los Angeles	1800 €
San Diego	250 €
Boston>	700 €

Tabla 3.- Ejemplo de tabla resultado de una consulta.

La palabra clave **GROUP BY** se utiliza cuando se seleccionan columnas múltiples desde una tabla (o tablas) y aparece al menos un operador aritmético en la instrucción SELECT.

Capítulo 3

Estado del arte.

A continuación se mencionarán las interfaces de lenguaje natural a bases de datos (ILNBD) relevantes y se hará una descripción de cada una de ellas.

3.1 TAMIC desarrollada por el Instituto per la Recerca Scientifica e Tecnologica Italy (1996)

La interfaz en lenguaje natural TAMIC (Transparent Access to Multiple Information for the Citizen) [Bagnasco, Bresciani, et al, 1996], tiene como principal finalidad permitir al ciudadano promedio, que no tiene conocimientos en computación, el poder acceder a la información de la administración pública. El sistema contiene una plataforma a través de la cual un administrador público (persona con conocimiento de la base de datos y el sistema) puede interactuar con el sistema.

La interfaz se instala en una computadora personal conectada por medio de una red a un sistema de pensión y a un monitor donde cuálquier ciudadano puede observar y entender el diálogo que se efectúa entre el sistema y el administrador público.

Este monitor permite al ciudadano introducir preguntas en lenguaje natural en base al diálogo observado, que el sistema analiza y las respuestas dadas por el sistema también se muestran en el monitor.

La interfaz utiliza el analizador WEDNESDAY, emplea un diccionario técnico para consultar a la base de datos, y utiliza otros componentes que se han desarrollado en el mismo instituto (IRST). Emplea diálogo para aclarar dudas en la consulta y brindar al usuario una mejor respuesta.

Como la interfaz contiene varias herramientas existen dos niveles en el que el usuario puede interactuar con la interfaz.

El primero es el nivel del usuario común en el cual se le permite utilizar una herramienta a la vez y pasar fácilmente de una herramienta a otra.

El segundo en un nivel más profundo que le permite al usuario tener un enfoque global de la interacción con el sistema, integrando un navegador y acceder a la información mediante un simple diálogo en lenguaje natural.

La interfaz TAMIC realiza las siguientes funciones:

a) Permite acceder a los datos estructurados del dominio en lenguaje natural.
b) Visualizar el dominio mediante una interfaz gráfica.
c) Permite acceder textualmente a las bases de datos
d) Permite acceder al diccionario técnico de la interfaz.
e) Cuenta con herramientas de ayuda para el usuario.

En la siguiente figura se muestran los módulos de TAMIC.

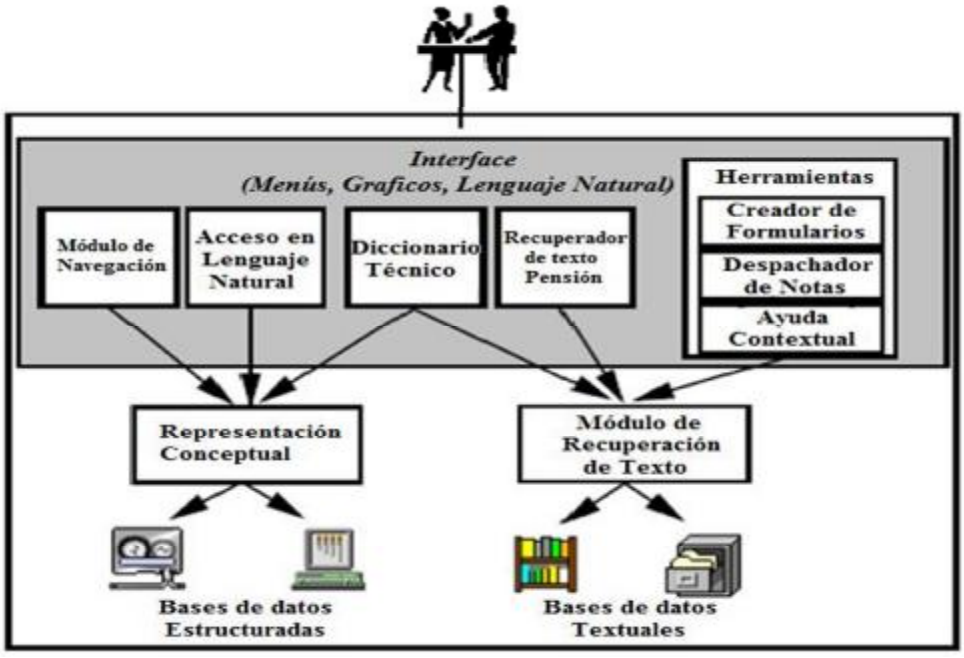


Figura 2.- Módulos de la Interfaz TAMIC.

La arquitectura de TAMIC se basa en un modelo conceptual, que realiza una representación intermedia del conocimiento del mundo que es independiente de la estructura física de la base de datos que se está utilizando.

En las pruebas realizadas en la interfaz TAMIC con el prototipo se accede a información de una sola base de datos, pero se propone el acceder a varias bases de datos a la vez. El último trabajo realizado en esta interfaz fue en el año 2000.

3.2 PRECISE (2003)

La interfaz PRECISE [Popescu, Etzioni and Kautz, 2003], fue desarrollada en la universidad de Washington, y su función es traducir una consulta realizada en lenguaje natural a una sentencia en lenguaje formal de SQL. La primera acción que realiza es determinar si la consulta en lenguaje natural se puede tratar semánticamente, para ello obtiene los elementos léxicos de la sentencia dada como entrada y los compara con elementos de la base de datos a través de restricciones semánticas impuestas, que es resuelto con un algoritmo de flujo máximo, reduciendo el problema de una comparación de grafos. Sin embargo, una de sus principales limitaciones es que el usuario a través de un editor debe realizar el grafo semántico.

La interfaz PRECISE es independiente del dominio, ya que se puede configurar para trabajar con distintas bases de datos. Esta interfaz solo traduce consultas que son semánticamente tratables.

PRECISE se compone de varios módulos a través de los Cuáles se realiza el proceso de traducción, dichos módulos son los siguientes: El analizador sintáctico, el lexicón, el comparador, el tokenizador, el verificador de equivalencias y el generador de consultas.

PRECISE obtiene el lexicón de manera automática en el proceso de configuración, extrayendo los elementos de la base de datos. Después se establece un conjunto de

sinónimos para cada palabra de forma manual, utilizando Word Net, que es una enorme base de datos léxica del idioma inglés que se usa como ontología. De esta forma cada elemento de la base de datos está asociado con una palabra, y cada palabra con un conjunto de sinónimos para ampliar el léxico que los usuarios pueden usar para realizar las consultas. En el léxico existen una serie de restricciones para tokens (Palabras) que sean preposiciones o verbos.

Para realizar el análisis sintáctico PRECISE utiliza Parser Charniak que es un analizador sintáctico estándar y una vez identificados los tokens se extraen relaciones entre ellos. En las palabras del léxico el tokenizador busca los tokens, en los que se encuentra la raíz de cada palabra y se extrae el conjunto de elementos de la base de datos que están relacionados con cada token. Luego el comparador construye un grafo para hacer las relaciones atributo-valor, y por medio de un algoritmo de flujo máximo para verificar si la consulta de entrada satisface las restricciones sintácticas representadas por los tokens ligados.

Una vez hecho esto se prueba si la consulta puede ser traducida a sintaxis SQL, se construyen las diferentes interpretaciones y en caso de que existan más de una, PRECISE solicita al usuario elegir una de ellas para dar una mejor respuesta. Por último se genera la consulta SQL que corresponde con la consulta. Se agregan condiciones de unión a la cláusula WHERE si existen múltiples relaciones.

PRECISE toma como entrada la consulta en lenguaje natural hecha por el usuario y extrae la información necesaria. El proceso de extracción se realiza usando palabras clave obtenidas del grafo semántico construido a partir de la base de datos. Sin embargo debido a que las palabras clave pueden tener diferentes significados dependiendo el dominio, es necesario resolver ambigüedades entre el significado de palabras clave, para realizar esto se utilizan aproximaciones estadísticas que incluyen la comparación de vectores de n-gramas. Esta interfaz puede utilizarse en diferentes dominios, sin embargo depende de la limitada ontología que soporta al dominio y a oraciones cortas y simples.

3.3 INBASE (2003)

InBase [Boldasov et al, 2003], es un proyecto creado en el Russian Research Institute en el año 2003, maneja patrones semánticos para consultar la base de datos, utiliza el Q-language un lenguaje similar a OQL (Object Query Language), utiliza Q-Gen que es un generador de lenguaje natural y para brindar una mejor respuesta en caso de que exista duda utiliza diálogos de aclaración con el usuario.

La interfaz InBase se basa en un enfoque semántico para analizar el texto en lenguaje natural. Utilizando este enfoque asegura una implementación automática, confiable y eficiente al interpretar una consulta en lenguaje natural en una base de datos relacional, la fácil adaptación de esta tecnología a diversos DBMS´s, utilizar varios idiomas y diferentes dominios.

InBase separa el análisis de las consultas en dos aspectos: En la información del dominio en una base de datos específica y la utilización de patrones semánticos que se usan para realizar la consulta a la BD. La estructura de InBase se muestra a continuación:

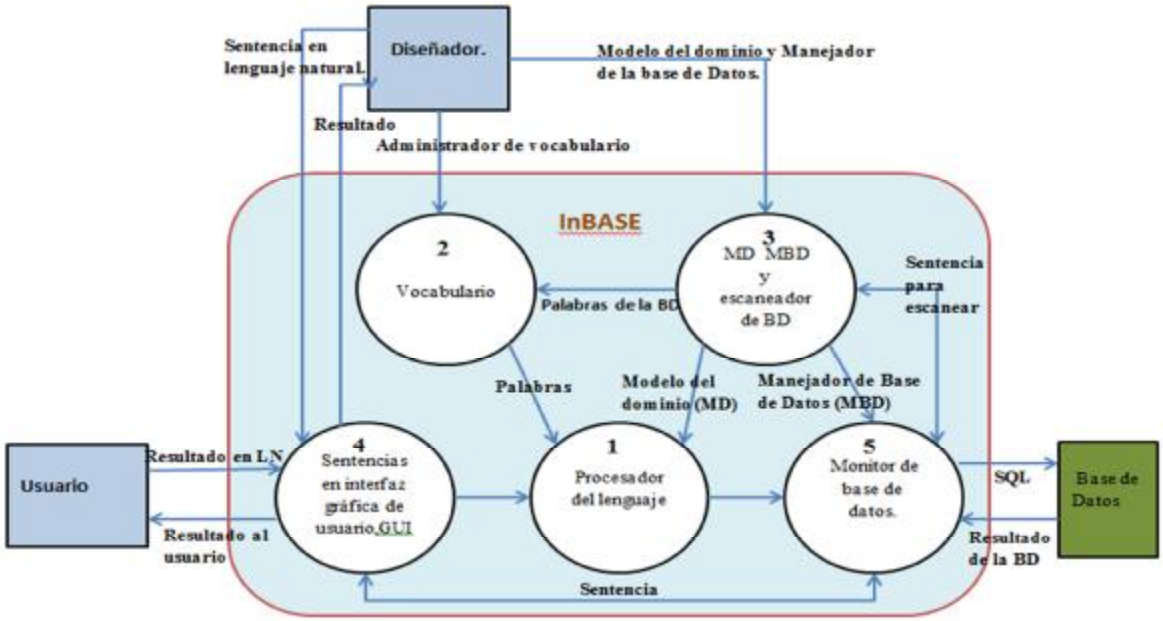


Figura 3.- Estructura de la interfaz InBase.

La interfaz InBase utiliza Q-language para construir una representación formal intermedia. El sistema utiliza un modelo de dominio (DM) sobre la base del diagrama de clases para construir la representación Q-query. La conversión final del lenguaje Q-query a lenguaje SQL o a cuálquier otra representación de la consulta considera el modelo relacional (RM) de la base de datos.

Para convertir la consulta en lenguaje natural a una sentencia SQL, el sistema produce su representación en OQL – Q-query, esta sentencia es transmitida a la entrada del módulo generador el Cualconvierte la representación OQL en una representación en lenguaje natural.

3.4 NLPQC – Concordia University (2005).

La interfaz NLPQC [Stratica, 2005], es un sistema basado en plantillas de dominio, que traduce consultas realizadas en el idioma ingles a consultas SQL para un sistema de base de datos relacional. El análisis sintáctico de las consultas se realiza utilizando un analizador de enlace y el análisis semántico se realiza por medio de plantillas de dominio específico.

El sistema se compone de un preprocesador y un módulo que se ejecuta en tiempo de ejecución. El preprocesador crea una base de conocimiento conceptual del esquema de la base de datos utilizando WordNet. Esta base de conocimiento creada se utiliza en tiempo de ejecución para hacer el análisis semántico de la consulta en lenguaje natural y crear su correspondiente sentencia SQL. Este sistema es de dominio independiente y ha sido probada con la base de datos CINDI que contiene información de una biblioteca virtual.

Esta interfaz cuenta con 4 módulos principales.

- El preprocesador (Pre-processor).
- El análisis sintáctico (Syntactic analysis)

- El análisis semántico (Semantic analysis)
- La construcción del conjunto de consultas (Build set of queries).

En la siguiente figura se muestra la arquitectura utilizada por la interfaz NLPQC, usada con la base de datos CINDI.

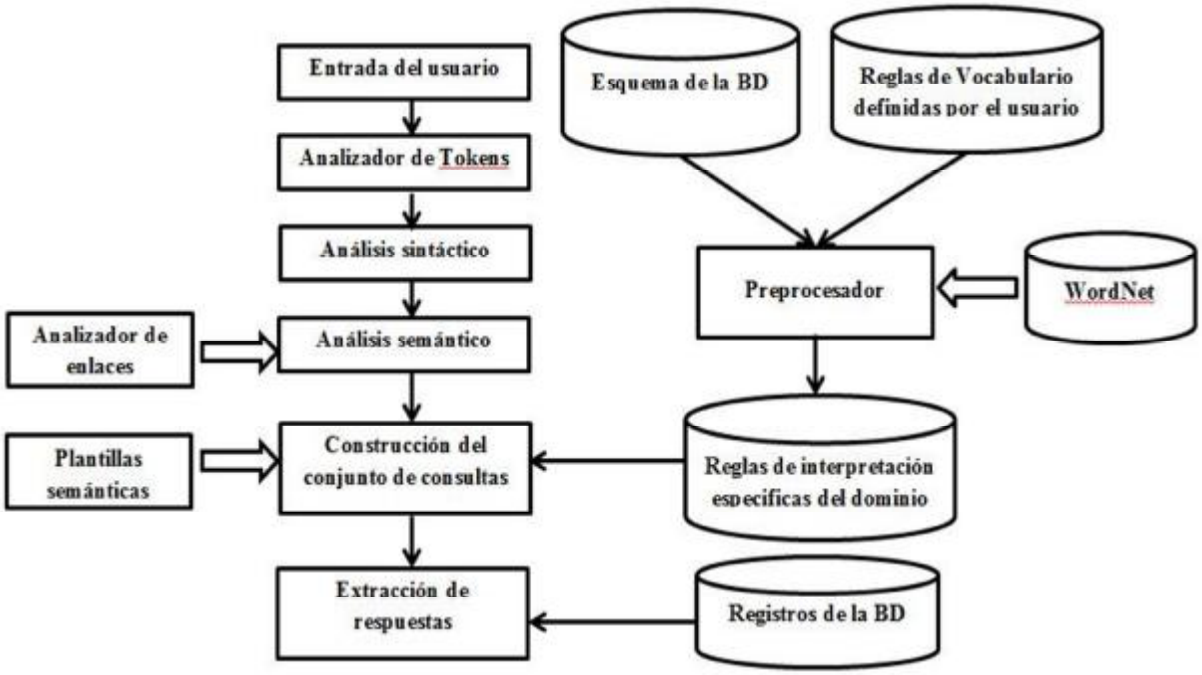


Figura 4.- Arquitectura de la interfaz NLPQC.

Preprocesador.

Mediante la base de conocimiento que se construye a partir del esquema de la base de datos se limita el alcance de la interpretación de la entrada. Utilizando este esquema, las reglas definidas por el usuario y el vocabulario de WordNet se crean reglas de interpretación de dominio específico que forman la base de conocimiento. Con esta la interfaz analiza las variaciones léxicas de las consultas de entrada. El preprocesador primero lee el esquema de la base de datos, identifica los nombres de atributos y las relaciones y crea una lista de sinónimos, hiperónimos e hipónimos para cada relación y nombre de atributo con WordNet.

Análisis Sintáctico

Se usa el analizador LINK para analizar la consulta de entrada. Este analizador devuelve un conjunto de todos los árboles de análisis posibles y los ordena en base a su probabilidad, este analizador es de software libre.

Además este analizador es muy preciso y rápido, estas características son necesarias ya que el análisis de las consultas se realiza en tiempo de ejecución.

Análisis semántico

Ya que la consulta de entrada ha sido analizada sintácticamente, se usan plantillas lograr interpretar si significado.

Las tres plantillas que se usan son las siguientes:

- ***<Atributo> de <objeto>***
- ***<Atributo> de <objeto> de <objeto>***
- ***<Verbo> <objeto> <valor del atributo>***

Del conjunto de árboles generados que están ordenados probabilísticamente, busca hasta encontrar alguno que coincida con alguna de las plantillas mencionadas.

Construcción del conjunto de consultas

Una vez que se ha realizado el análisis semántico, a partir de la plantilla seleccionada se construye la sentencia en lenguaje SQL.

3.5 WYSIWYM – The Open University (2006)

La interfaz WYSIWYM [Hallett, 2006], no requiere al usuario que introduzca las consultas de texto libre, sino que ayuda al usuario a construir la consulta a través de una interfaz de lenguaje natural, por lo que la necesidad del análisis sintáctico y la interpretación semántica quedan eliminadas.

Esta interfaz utiliza un grafo semántico para representar la estructura de la base de datos. Contiene patrones definidos con los que ayuda al usuario a formular sus consultas, por medio de menús que permiten ir construyendo las consultas, donde se muestran las tablas y columnas disponibles.

Se basa principalmente en la creación conceptual de las consultas por medio de la interfaz WYSIWYM, y en la portabilidad del sistema.

Esta interfaz hace uso de un grafo semántico para representar el modelo de la base de datos, y se utiliza esta información como principal medio para generar automáticamente marcos de sentencias.

La interfaz presenta al usuario con un texto en lenguaje natural que corresponde a la consulta incompleta y lo guía hacia la edición de una consulta semánticamente coherente y completa por medio de menús, de manera automáticamente genera las reglas, componentes y recursos que el sistema utiliza para traducir la consulta.

La semántica de una base de datos relacional se especifica como un grafo dirigido donde los nodos representan los elementos y las aristas representan las relaciones entre los elementos.

Cada tabla de la base de datos puede ser visto como un subgrafo, con aristas que representan un tipo especial de relación. Cada nodo tiene que ser descrito en términos de su semántica y en términos de su construcción lingüística. El tipo semántico de un nodo está restringido por el tipo de dato correspondiente a la entidad en la base de datos. Un ejemplo de un grafo semántico utilizado por el sistema es el siguiente:

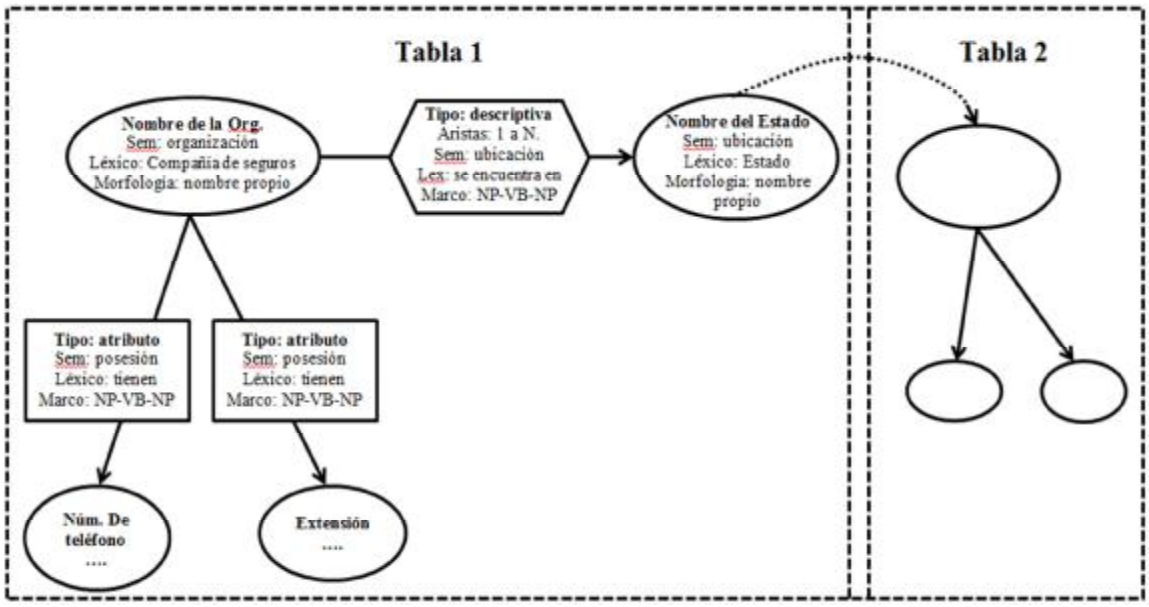


Figura 5.- Grafo semántico de WYSIWYM.

En base a las reglas gramaticales del idioma ingles el sistema genera automáticamente preguntas a partir de las conexiones entre entidades y atributos del grafo semántico, colocando entre corchetes la columna a la que es más probable que se haga referencia. Mediante un menú el usuario elige que pregunta hacer y mediante otro menú la columna a la que se esté haciendo referencia.

Una vez que una consulta se ha construido, se representan internamente como un grafo dirigido acíclico. Además, cada nodo en el grafo puede ser mapeado dentro de un nodo en el grafo semántico de la base de datos.

3.6 C-PHRASE – Umeå University (2008)

La interfaz C-PHRASE [Minock, 2008], modela las consultas utilizando el cálculo de tuplas de Codd y utiliza las gramáticas sincrónicas extendidas con las funciones lambda para representar la gramática semántica. Esta interfaz integra y simplifica las herramientas de su misma autoría, utiliza diálogo como medio de aclaración con el usuario.

Esta interfaz se basa en técnicas de procesamiento de lenguaje natural (PLN), para definir un dominio especifico utiliza definición de equivalencias, dominio léxico, lenguaje de procesamiento natural y la definición de funciones relacionales.

Debido a las dificultades formales de trabajar directamente con las expresiones de SQL, se representan las consultas como expresiones de cálculo de tuplas de Codd. El cálculo de tuplas es un conocido analizador sintáctico desarrollado sobre lógica de primer orden estándar donde las variables son tuplas en las relaciones de bases de datos y no sobre lugares individuales en relaciones n-arias.

Se modelan las gramáticas semánticas como gramática libres de contexto sincrónico aumentada con las expresiones de cálculo lambda (λ-SCFG). Estas reglas λ-SCFG definen dos árboles síncronos de derivados del mismo símbolo de inicio S. El resultado del primer árbol es el lenguaje natural, y el resultado del segundo árbol es un lenguaje formal que expresa la semántica del lenguaje natural, del primer árbol. El requisito de tener las variables dentro de las fórmulas semánticas requiere el uso de expresiones λ y, a su vez (ligeramente) complica la noción de lo que sería el resultado del segundo árbol - los resultados se calculan de abajo hacia arriba e implican conversiones alfa conocidas y las operaciones de reducción beta del cálculo lambda.

De manera formal cada regla tiene la siguiente forma: $A \rightarrow \langle \alpha, \lambda X_1, \ldots, \lambda X_k \beta \rangle$

Donde "A" es un único símbolo no terminal y "α" es una secuencia de símbolos terminales y no terminal, donde los terminales son palabras o secuencias de palabras en lenguaje natural. Consiste en una secuencia de terminales, no terminales y los términos formales de argumentos que componen los argumentos x1, ..., xk. En ocasiones se pasan las funciones lambda como argumentos.

La tarea principal de análisis para las interfaces de lenguaje natural para bases de datos es el análisis de las frases sustantivas, a menudo con cláusulas relativas.

En la notación de λ-SCFG algunas de las reglas generales que orientan su proceso son:

QUERY → <"list the" ・ NP, answer({x|NP(x)})>
NP → <PRE ・ NP,PRE(NP)>
NP → <NP1,NP1>
NP1 → <NP1 ・ POST, POST(NP1)>
NP1 → <HEAD,HEAD>

La primera regla expresa lo que nosotros llamamos un modelo de oración, de la cual hay muchas variantes. Esta regla en particular permite usar el tipo "lista de X" donde X es una frase sustantiva. En total el sistema cuenta con alrededor de 75 patrones manualmente definidos de las oraciones y rara vez nos encontramos con expresiones escritas por los usuarios que no estén cubiertas por estos patrones básicos. Por supuesto, a medida que descubrimos nuevos, nosotros, como diseñadores del sistema, sólo se tienen que añadir a la interfaz. Los últimos cuatro reglas anteriores son más interesantes y permiten analizar frases sustantivas como los de la siguiente figura que corresponde a la consulta " Lista las grandes ciudades en el estado más grande del medio oeste".

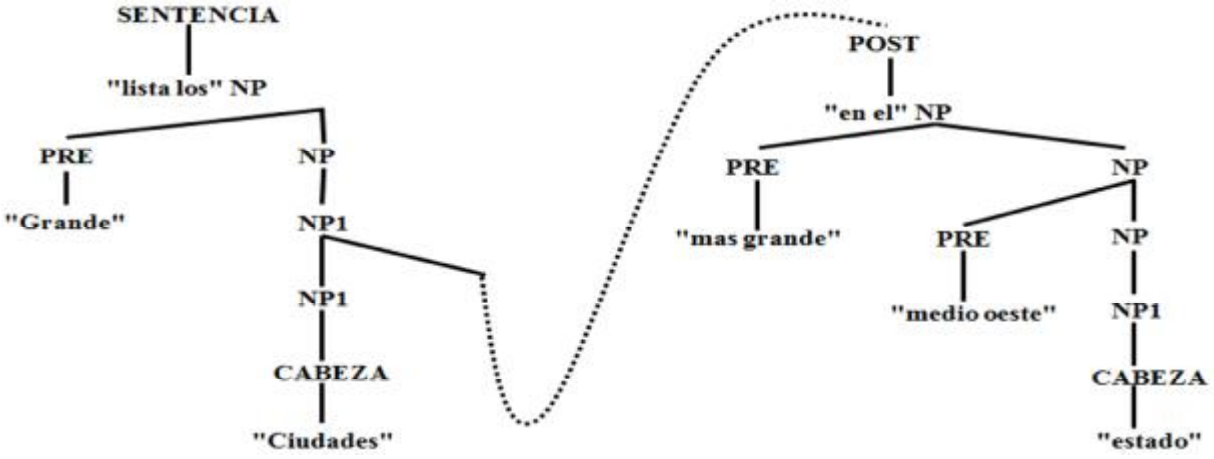


Figura 6.- Análisis de una consulta en C-PHRASE.

3.7 Traductor Rojas. (2009)

Esta interfaz de [Rojas, 2009], es una continuación de la ILNBD desarrollada por el Dr. Javier González Barbosa en CENIDET, fue desarrollada por el MCC. Juan Carlos Rojas Pérez en Junio del 2009, en su tesis doctoral titulada “Administrador de Diálogo para una Interfaz de Lenguaje Natural a Bases de Datos”.

A la interfaz del CENIDET, se le incorporó un administrador de diálogo. Este administrador está conformado por procesos de diálogo que fueron diseñados a partir de una clasificación de los problemas que se presentan al realizar consultas en lenguaje natural, el análisis y la prueba del traductor se realizó con corpus de tres bases de datos que son: Northwind, Pubs y ATIS.

La clasificación de las consultas se realiza de acuerdo a varios tipos de problemas que se presentan al realizar una consulta principalmente el relacionado a la economía de las palabras que incluye mucha información implícita. Esta interfaz posee dos características importantes: La generalidad, que da la posibilidad de poder implementarlo en diversos lenguajes. Y la independencia del dominio es decir que se puede aplicar a diversas bases de datos relacionales. El proceso de diálogo ayuda a brindar mejores respuestas al usuario.

3.8 STK – University of Trento (2010).

La interfaz STK [Giordani 2010] utiliza conjuntos de datos que están formados por pares de consultas en lenguaje natural y consultas en SQL. Se realiza un mapeo a nivel sintáctico de la consulta y luego la aplicación de algoritmos de aprendizaje automático para obtener un traductor automático de las preguntas en lenguaje natural a sus correspondientes consultas SQL. Para ello, se diseñó un conjunto de datos de pares de relaciones que contienen los árboles sintácticos de las preguntas y

consultas y se codifican en máquinas de soporte vectorial por medio de funciones del kernel, para relacionar los dos lenguajes.

Se considera un conjunto de datos de preguntas en lenguaje natural "N" y un conjunto de consultas SQL "S" relacionadas con un dominio y una base de datos específica y se entrena a un clasificador para mapear de forma automática el conjunto de pares. Se asume que los pares son considerados como correctos cuando la consulta SQL responde a la pregunta, e incorrectos en caso contrario. Se entrena un clasificador sobre los pares para seleccionar la sentencia SQL correcta para una pregunta. Entonces, se asignan nuevas preguntas en el conjunto de datos de consultas disponibles, se establece el ranking de esta última por medio de una calificación dada por el clasificador y se selecciona la mayor.

Por ejemplo, cuando se tiene la consulta en lenguaje natural n1: “¿Cuáles estados limitan con Texas?” y las siguientes consultas.

s1: SELECT (Nombre_Estado) FROM Info_Frontera WHERE Limita_con = ‘texas’

s2: SELECT COUNT (Nombre_Estado) FROM Info_Frontera WHERE Limita_con = ‘texas’.

Ya que s1 es una interpretación correcta de la consulta y s2 es incorrecta, el clasificador debe asignar una calificación más alta a S1, de esta manera que la interfaz obtendrá como salida el par <n1, s1>.

En la siguiente figura se muestra un ejemplo de la representación de las consultas por medio de árboles sintácticos.

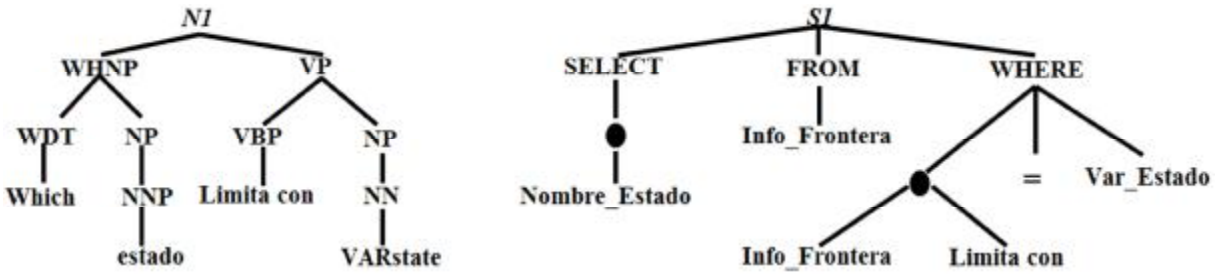


Figura 7.- Pares de consulta en árboles sintácticos.

El objetivo principal del desarrollo de la interfaz STK, es la de relacionar los pares de consulta (Lenguaje natural y SQL) de manera superficial por medio la información semántica. Para representar estas estructuras se utilizan núcleos de árbol (tree kernels) que están basados en máquinas de vector de soporte.

3.9 Tabla comparativa.

A continuación se muestra una tabla comparativa de las interfaces en lenguaje natural a bases de datos que se mencionaron anteriormente.

INTERFAZ	1	2	3	4	5	6	7	8
TAMIC (1996)				✓			✓	
PRECISE (2003)	✓		✓	✓	✓			
InBase (2003)	✓			✓			✓	
NLPQC (2005)	✓			✓	✓			
Traductor CENIDET (2005)	✓	✓	✓	✓		✓		
WYSIWYM (2006)	✓		✓	✓			✓	
C-PHRASE (2008)	✓		✓	✓			✓	
Traductor Rojas (2009)	✓	✓	✓	✓		✓	✓	
STK (2010)	✓			✓				
Trabajo actual (2012)	✓	✓	✓	✓		✓	✓	✓

Tabla 4.- Tabla comparativa de las ILNBD.

Características que se consideran:

1. Independencia del dominio.
2. Utilización del idioma español.
3. Sencillez al configurar la interfaz para que funcione en otro dominio.
4. Manejo de sustantivos para analizar las consultas.
5. Manejo de verbos para analizar las consultas.
6. Importa en el tratamiento de las preposiciones y las conjunciones.
7. Interacción mediante un diálogo con el usuario.
8. Manejo de funciones de agregación y de comparación.

En esta tesis se pretende agregar un gestor de diálogo a la interfaz de CENIDET con el fin de mejorar el porcentaje de respuestas contestadas correctamente por la interfaz de [González A., 2011].

Capítulo 4

Propuesta de solución.

La interfaz desarrollada por el doctor Juan Javier González Barbosa en CENIDET [González J.J., 2005], está compuesta principalmente por dos fases, la construcción del diccionario de dominio y la fase de traducción, este trabajo se centrará en la fase de traducción.

Se analizó la fase de traducción y se añadió el tratamiento de las funciones de agregación “COUNT”, “MAX” y “MIN” y el administrador de diálogo. En la siguiente figura se muestra la arquitectura completa de la interfaz y en azul las fases que fueron agregadas a la interfaz.

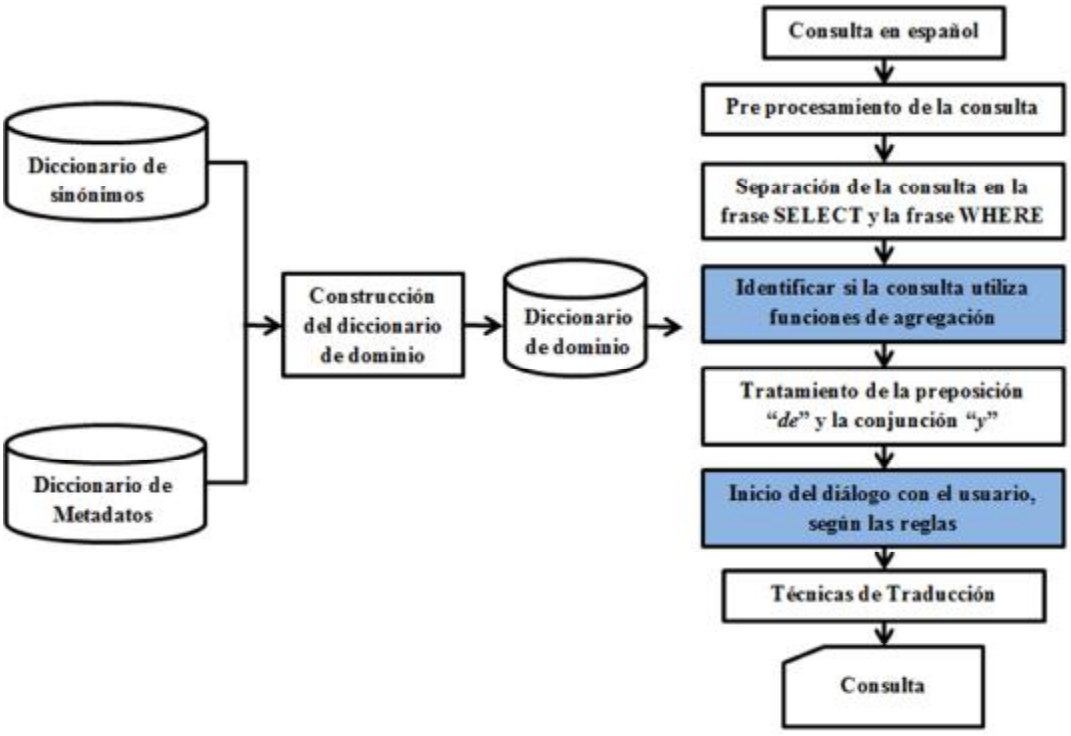


Figura 8.- Arquitectura CENIDET con las problemáticas agregadas.

4.1 Funcionamiento de la Interfaz original.

A continuación se describe el funcionamiento de la interfaz original por medio de un ejemplo.

En la **fase 1**, se separan las palabras en tokens y se eliminan símbolos no deseados en la consulta. En la **fase2**, se identifica el tipo (sustantivo, adjetivo, preposición, etc.) de cada una de las palabras de la consulta. En la **fase 3**, se crean las tablas que guardaran las referencias a tablas o campos encontradas de cada una de las palabras en la base de datos que se consulta. En la **fase 4**, se separa la consulta en lenguaje natural en la cláusula “SELECT” y la cláusula “WHERE”. En la **fase 5**, se realiza el tratamiento de preposiciones y conjunciones, estableciendo una prioridad a cada una de ellas. En la **fase 6**, siguiendo la prioridad de la fase anterior se realiza la intersección o unión de las tablas y campos encontrados en la fase 3. En la **fase 7**, utilizando las palabras que se identificaron como valores se construye la condición de la consulta, es decir la cláusula “WHERE”.

Se tiene la siguiente consulta de entrada:

"***Muestra los nombres de los clientes de CANADÁ***".

4.1.1. Fase 1.- Eliminar símbolos especiales.

Lo primero que se realiza en la interfaz es separar las palabras de la consulta en tokens, un ejemplo de esto se muestra a continuación:

Muestra los nombres de los clientes de CANADÁ							
Muestra	los	nombres	de	los	clientes	de	CANADÁ

Tabla 5.- Separación de la consulta en tokens.

De cada palabra se eliminan símbolos especiales que podrían aparecer al momento que el usuario escribe la consulta y se realizan las acciones dependiendo del símbolo encontrado que se describen en la siguiente tabla:

Símbolo	Acción	Ejemplo de token	Resultado
, (coma)	Se reemplaza por "y"	Pavlova, Tofu	Pavlova y Tofu
$ Inicio	Se reemplaza por "precio " si aparece al inicio	$450	Precio 450
"P1 P2"	Se eliminan comillas y se reemplaza " " por "_"	"Exotic liquids"	Exotic_liquids
"Palabra"	Se eliminan comillas y al inicio se inserta "__"	"Confections"	__Confections
¿	Se eliminan	¿Cuáles	Cuáles
?	Se eliminan	seafood?	seafood
$ Interm	Se eliminan los símbolos de pesos intermedios	800.$00	800.00
Acentos	Se eliminan todos los acentos de las palabras	descripción	descripcion

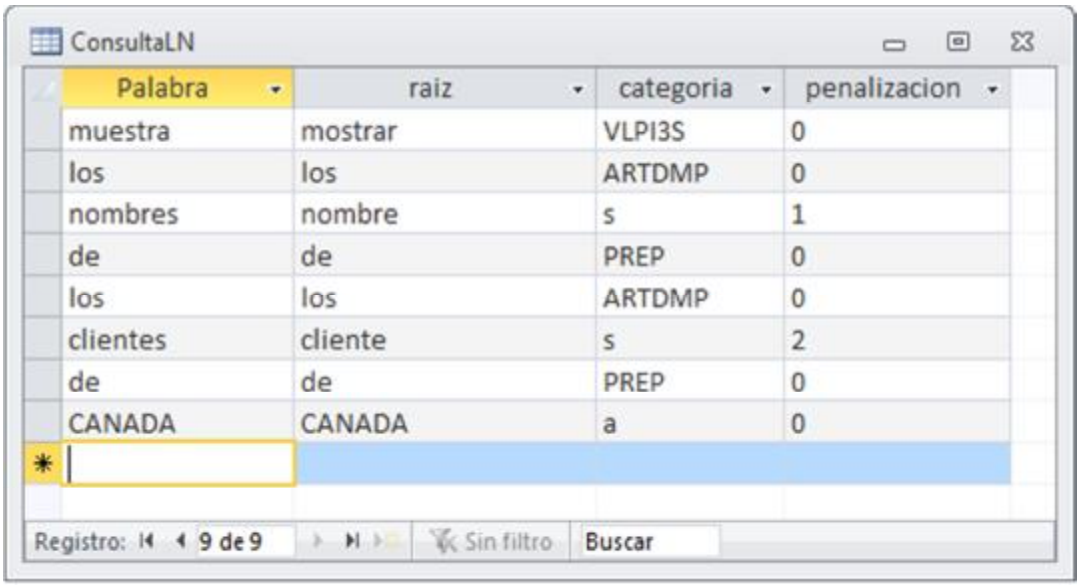
Tabla 6.- Remplazo de símbolos en la consulta.

4.1.2. Fase 2.- Identificación del tipo de palabras y penalizaciones.

Se busca cada palabra en la base de datos "**Lexicon**", se obtiene su palabra raíz y su categoría, que son unas siglas que identifican que tipo de palabra es la actual [***Ver sección 4.4***].

Estos datos se guardan en la tabla "**ConsultaLN**" de la base de datos "**baseconsultas**", donde se le asigna una penalización que indica si tiene el valor **"1"** que la palabra hace referencia a una **columna** y un **"2"** si la palabra hace referencia a una **tabla**.

En la siguiente imagen se muestra el resultado de la tabla "ConsultaLN" para la consulta: "***Muestra los nombres de los clientes de CANADA***".



ConsultaLN

Palabra	raiz	categoria	penalizacion
muestra	mostrar	VLPI3S	0
los	los	ARTDMP	0
nombres	nombre	s	1
de	de	PREP	0
los	los	ARTDMP	0
clientes	cliente	s	2
de	de	PREP	0
CANADA	CANADA	a	0

Registro: 9 de 9 Sin filtro Buscar

Figura 9.- Ejemplo de identificación de palabras en la Tabla "ConsultaLN".

4.1.3. Fase 3.- Creación de tablas auxiliares.

En la fase anterior se identificaron en la tabla "**ConsultaLN**", las palabras que hacen referencia a un campo, que tienen valor "**1**" en el campo "**penalización**" y las palabras que hacen referencia a una tabla, que tienen el valor "**2**" en dicho campo.

Después se extraen las palabras que hagan referencia a una tabla o columna, en este caso las palabras son “**nombres**” y “**clientes**”. La palabra “**nombres**” hace referencia a **columnas** y la palabra “**cliente**” hace referencia a una **tabla**.

Una vez identificadas estas palabras se crean dos tablas en la base de datos “**baseconsultas**” por cada una de estas palabras con los nombres “**SustantivoTabla**” y “**SustantivoCampos**” donde el sustantivo es la palabra identificada.

Para el ejemplo actual las tablas creadas en la base de datos "baseconsultas" son las siguientes.

Palabra que hace referencia a tabla o columna	Nombre de las tablas creadas en la BD baseconsultas
Nombre	NombreCampos NombreTabla
Cliente	ClienteCampos ClienteTabla

Tabla 7.- Tablas creadas en la BD "baseconsultas".

En estas tablas se guardan todas las referencias que existan a tablas o columnas respectivamente de dicha palabra encontrada en la base de datos que se esta consultando (Northwind o Pubs), es decir todas las tablas y columnas que en su descripción contengan la palabra identificada.

Para la consulta que utilizamos de ejemplo "***Muestra los nombres de los clientes de CANADÁ***", los valores de dichas tablas creadas se muestra en la siguiente imagen.

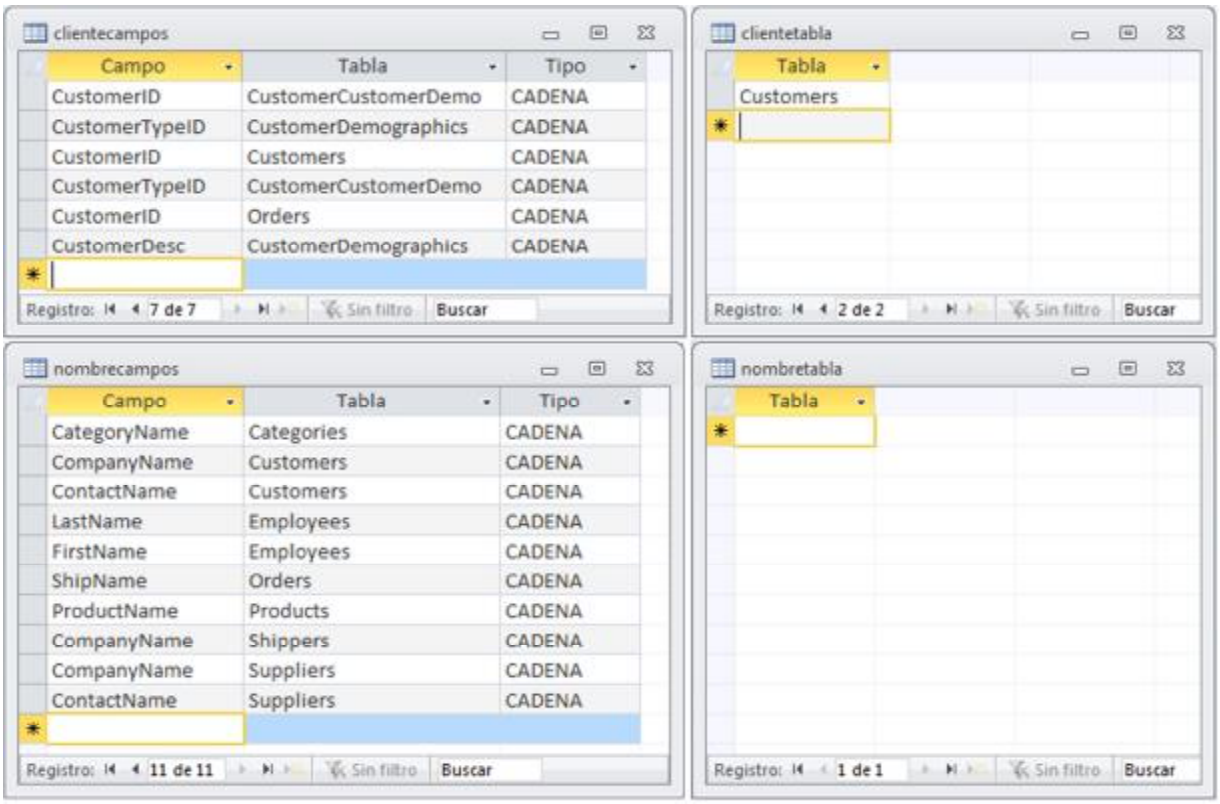


Figura 10.- SustantivoCampos y SustantivoTabla de los sustantivos de la consulta.

En este caso se pueden ver los campos a los que hace referencia “nombre” y “clientes”, también se puede ver que solo la palabra “clientes” hace referencia a una tabla, por lo tanto la tabla “**nombretabla**” no contiene ningún registro.

4.1.4. Fase 4.- Separación de la cláusula SELECT y la cláusula WHERE.

Una vez identificados los campos y tablas, se realiza la separación de la consulta en lo que será la cláusula “**SELECT**” y la cláusula “**WHERE**”, dependiendo la estructura de la consulta de entrada en lenguaje natural.

Para realizar la separación de la consulta se utilizan algunos criterios que toman en cuenta las palabras que hacen referencia a tablas o columnas y los valores existan.

Dichos criterios se describen a continuación:

Primer caso.

Cuando se encuentra un 'valor' y existe un sustantivo previo que hace referencia a una tabla. La oración se corta desde el inicio hasta donde está el sustantivo que hace referencia a la tabla.

Ejemplo.

Consulta:	**Muestra los nombres de los clientes de CANADÁ.**
Preprocesador:	nombre de cliente de CANADÁ
Sustantivo Tabla:	Cliente
Valor:	CANADÁ
SELECT:	nombre de cliente
WHERE:	de CANADÁ

Tabla 8.- Primer caso de separación de consulta.

Segundo caso.

Cuando se encuentra un 'valor' pero no hay sustantivo que haga referencia tabla, en ese caso se corta desde el inicio de la oración hasta donde está el 'valor' y del 'valor' en adelante es la condición.

Ejemplo.

Consulta:	**Dame el identificador de Juan**
Preprocesador:	identificador de Juan
Sustantivo Tabla:	No existe
Valor:	Juan
SELECT:	identificador de
WHERE:	Juan

Tabla 9.- Segundo caso de separación de consulta.

Tercer caso.

Cuando hay dos sustantivos seguidos y el primer sustantivo hace referencia a tablas y el segundo sustantivo hace referencia a columnas, se divide de tal modo que se separan los dos sustantivos.

Ejemplo.

Consulta:	**Dame los nombres de los empleados con fecha de nacimiento 17/09/1958**
Preprocesador:	nombre de empleado fecha de nacimiento 17/09/1958
Sustantivo tabla:	empleado
Sustantivo columna:	fecha
Valor:	17/09/1958
SELECT:	nombre de empleado
WHERE:	fecha de nacimiento 17/09/1958

Tabla 10.- Tercer caso de separación de consulta.

Cuarto caso.

Cuando en una consulta no existe un valor, la consulta no tiene condición y por lo tanto toda la consulta estaría en la cláusula “SELECT”.

Ejemplo.

Consulta:	**Dame los nombres de contactos de los clientes**
Preprocesador:	nombre de contacto de clientes
Sustantivo Tabla:	clientes
Valor:	No existe
SELECT:	nombre de contacto de clientes
WHERE:	No existe

Tabla 11.- Cuarto caso de separación de consulta.

4.1.5. Fase 5.- Tratamiento de preposiciones y conjunciones.

Cuando ya se ha separado la consulta en la cláusula “SELECT” y WHERE”, se continúa con la fase de tratamiento de la preposición “de” y la conjunción “y”.

En esta fase se le asigna un valor numérico a cada preposición y conjunción que corresponde a la prioridad de la misma, recordando que el valor de los sustantivos es 1 si hacen referencia a una columna y 2 si hacen referencia a una tabla.

La forma de estas palabras se identifica de la siguiente manera [n1 de/y n2] donde:

- **N1:** es el valor del sustantivo a la izquierda de la intersección/unión.
- **De/y:** representa una preposición: 'de' o 'del' o una unión: 'y' o 'e'.
- **N2:** es el valor del sustantivo a la derecha de la intersección/unión.

Primero se evalúa toda la expresión y si se encuentra el caso **[2 de 2]** se cambia a **[1 de 2]**, cuando el primer sustantivo aunque hace referencia a una tabla, es una columna que pertenece a la tabla a la Cualhace referencia el segundo sustantivo, de lo contrario se queda igual.

Se asigna un valor número que indica la prioridad de dicha expresión de la forma **[n1 de/y n2],** dichos valores se describen a continuación en la siguiente tabla.

Caso [n1 de/y n2]	**Valor de prioridad asignado.**
[1 de 1]	Prioridad 1.
[1 y 1] o [2 y 2]	Prioridad 2.
[1 de 2]	Prioridad 3.
Cuálquier otro caso	Prioridad 4.

Tabla 12.- Prioridad asignada a preposiciones y conjunciones.

Ejemplo de asignación de prioridades a preposiciones y conjunciones en una consulta.

Consulta a procesar	**dirección**	**y**	**teléfono**	**de**	**empleados**	**y**	**clientes**
Valores de sustantivos	1		1		2		2
Prioridad de prep/conj		2		3		2	

Tabla 13.- Ejemplo de asignación de prioridades.

4.1.6. Fase 6.- Intersección de Tablas.

Después de asignar prioridades a las preposiciones y conjunciones se considera solo la cláusula “SELECT” de la consulta, se obtienen los “SustantivosTabla” y los “SustantivosColumnas” que se guardaron en la fase anterior que contienen las referencias a tablas y columnas de la consulta actual.

Se realiza la intersección entre las tablas siguiendo las prioridades encontradas en la fase anterior para encontrar los elementos en común de las referencias de las palabras de la consulta actual, para la consulta de ejemplo sería:

Primero se intersectan las tablas “SustantivosColumnas”, en este caso serían.

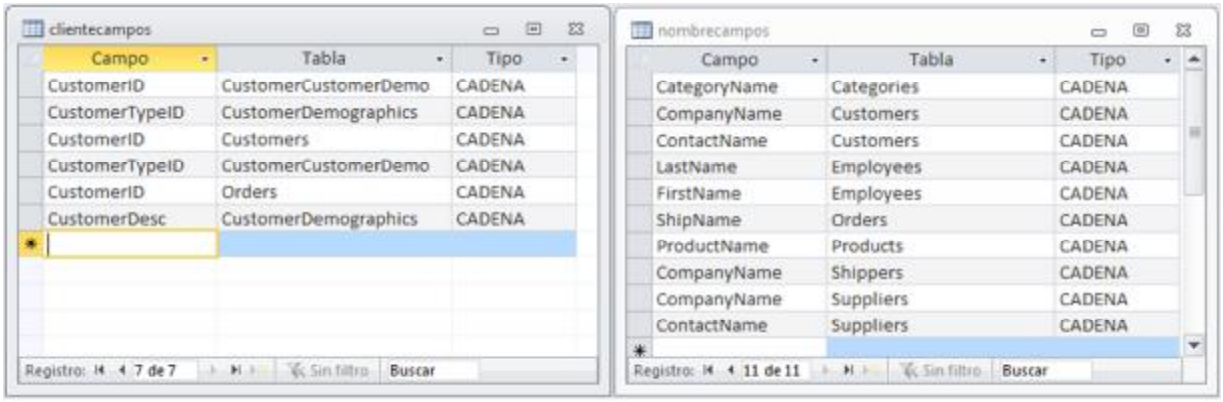


Figura 11.- Intersección entre tablas.

Como se puede ver, no hay campos en común entre las dos tablas, por lo que no regresa ningún resultado.

Después se intersecta la tabla “nombreTabla” con “clienteCampos” y con “nombreCampos” en este caso como “nombreTabla” no contiene ningún registro, tampoco devuelve ningún resultado.

Por último se intersecta la tabla “clienteTabla” con “clienteCampos” y con “nombreCampos” en este caso como “clienteTabla” si contiene a la tabla “Customer”, regresa los siguientes resultados:

Intersección de			**Campos encontrados**	**Tablas encontradas**
clienteTabla	Con	**nombreTabla**	Ninguno	Ninguno
clienteTabla	Con	**clienteCampos**	Ninguno	Ninguno
clienteTabla	Con	**nombreCampos**	CompanyName ContactName	Customer

Tabla 14.- Intersección entre SustantivosColumnas y SustantivosTablas.

Los campos encontrados en estas intersecciones se guardan en la tabla “ResultadoCampos” y las tablas en “ResultadoTablas”, por lo tanto ambas tablas según el ejemplo quedarían de la siguiente manera.

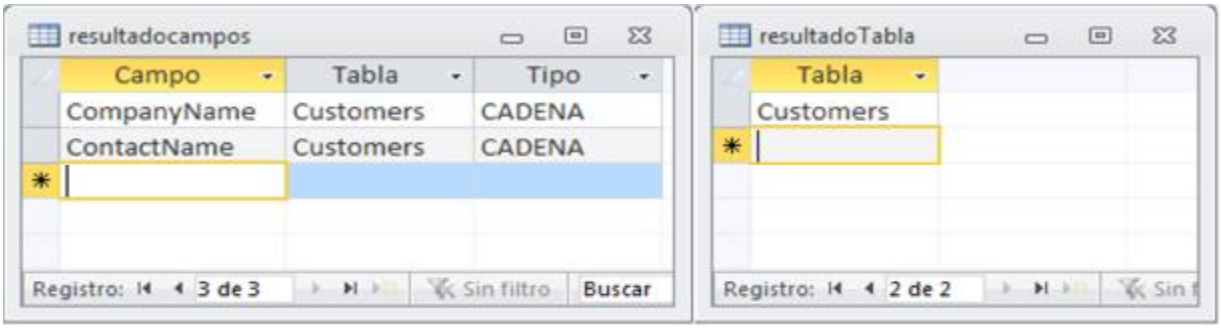


Figura 12.- Tablas resultado del ejemplo de la Regla 1.

Hasta este momento en la cláusula “SELECT”, se tienen identificados dos campos, resultado de las intersecciones de los “sustantivos campos” y los “sustantivos tabla” de la consulta, por lo tanto la cláusula “SELECT” quedaría de la siguiente forma:

```
SELECT Customers.CompanyName, Customers.ContactName
FROM Customers
```

4.1.7. Fase 7.- Análisis de la Cláusula "WHERE".

Una vez que se analizó la cláusula “SELECT” se procede a analizar la cláusula “WHERE”, en donde se toman los “sustantivos campos”, los “sustantivos tabla” y el valor o los valores de la consulta en la cláusula “WHERE”.

En este caso en la cláusula “**WHERE**” solo existe el valor “**CANADÁ**” y ningún sustantivo que haga referencia a algún campo o columna.

Si el sustantivo campo de la consulta es la palabra "**Identificador**", se iguala el valor a cada campo de la tabla que esté relacionada a un identificador y se establece esto como la condición "WHERE".

Si en la cláusula "WHERE" existe un sustantivo que haga referencia a una tabla, se busca el valor encontrado en todas las columnas de dicha tabla.

En el ejemplo actual, como no existe ningún sustantivo que haga referencia a una tabla en la cláusula "WHERE", se busca el valor en las tablas encontradas en la cláusula "SELECT".

Se toma el valor "**CANADÁ**" y se busca en todas las columnas de la tabla que hace referencia el sustantivo tabla de la cláusula "SELECT", es decir en la tabla "**Customers**".

Customers

CustomerID	CompanyName	ContactName	ContactTitle	Address	City	Region	PostalCode	Country	Phone	Fax
ALFKI	Alfreds Futterkiste	Maria Anders	Sales Representative	Obere Str. 57	Berlin		12209	Germany	030-0074321	030-0076545
ANATR	Ana Trujillo Emparedados	Ana Trujillo	Owner	Avda. de la Cons	México D.F.		05021	Mexico	(5) 555-4729	(5) 555-3745
ANTON	Antonio Moreno Taquería	Antonio Moreno	Owner	Mataderos 2312	México D.F.		05023	Mexico	(5) 555-3932	
AROUT	Around the Horn	Thomas Hardy	Sales Representative	120 Hanover Sq.	London		WA1 1DP	UK	(171) 555-7788	(171) 555-6750
BERGS	Berglunds snabbköp	Christina Berglun	Order Administrator	Berguvsvägen 8	Luleå		S-958 22	Sweden	0921-12 34 65	0921-12 34 67
BLAUS	Blauer See Delikatessen	Hanna Moos	Sales Representative	Forsterstr. 57	Mannheim		68306	Germany	0621-08460	0621-08924
BLONP	Blondesddsl père et fils	Frédérique Citea	Marketing Manager	24, place Kléber	Strasbourg		67000	France	88.60.15.31	88.60.15.32
BOLID	Bólido Comidas preparad	Martín Sommer	Owner	C/ Araquil, 67	Madrid		28023	Spain	(91) 555 22 82	(91) 555 91 99
BONAP	Bon app'	Laurence Lebihar	Owner	12, rue des Bouc	Marseille		13008	France	91.24.45.40	91.24.45.41
BOTTM	Bottom-Dollar Markets	Elizabeth Lincoln	Accounting Manager	23 Tsawassen Bl	Tsawassen	BC	T2F 8M4	Canada	(604) 555-4729	(604) 555-3745
BSBEV	B's Beverages	Victoria Ashworth	Sales Representative	Fauntleroy Circu	London		EC2 5NT	UK	(171) 555-1212	

Registro: 10 de 91 | Sin filtro | Buscar

Figura 13.- Tabla "Customers" de la base de datos Northwind.

Como se puede ver la única columna en donde se encuentra el valor "**CANADÁ**" es en "**Country**" por lo tanto es la que se agrega a la tabla "ResultadoCampos" y se construye la condición "**Customers. Country** LIKE '**CANADÁ**".

Por lo tanto se construye la cláusula "WHERE" utilizando esta columna y la consulta final en SQL devuelta por la interfaz seria la siguiente:

```
SELECT Customers.CompanyName, Customers.ContactName
FROM Customers
WHERE ((Customers.Country LIKE 'CANADA'))
```

Este es el funcionamiento general de la interfaz actualmente.

4.2. Modelo del administrador de diálogo.

Para desarrollar el sistema de diálogo se va a utilizar un modelo basado en reglas (También llamado modelo de Frames); un frame puede considerarse como una abstracción de un concepto, que contiene los atributos de dicho concepto y los valores correspondientes a cada atributo [Allen et al, 2001].

Para lograr obtener el valor de estos atributos el administrador de diálogo debe tener reglas que le indiquen cuando iniciar un diálogo con el usuario.

- Se estableció que el primer atributo sean los campos de la cláusula “SELECT”, es decir la interfaz debe encontrar las columnas que desea ver el usuario en la respuesta.
- Se estableció como segundo atributo a los campos que corresponden a la condición “WHERE”, es decir a que columnas hacen referencia los valores que están dados en la consulta del usuario.
- Se estableció como tercer atributo a la función de agregación que se necesita en la consulta, es decir si se requiere la función “COUNT”, “MAX” o “MIN”, o si la consulta actual no utiliza ninguna función de agregación.

Atributo 1: Campos de la cláusula SELECT	Atributo 2: Campos de la condición WHERE	Atributo 3: Campo al que se aplica la función de agregación utilizada.
SELECT Tabla1.campo1, Tabla1.campo2, Tabla2.campo1, Tabla2.campo2, …	WHERE Tabla1.campo1 = VALOR, Tabla1.campo2 = VALOR, Tabla2.campo1 = VALOR, Tabla2.campo2 = VALOR, …	COUNT / MAX / MIN Tabla1.campo1 Solo en caso de que la consulta utilice una función de agregación.

Figura 14.- Frame del administrador de diálogo.

Metodología.

La interfaz separa la consulta introducida por el usuario en la cláusula “SELECT” y la cláusula “WHERE”.

La interfaz toma los sustantivos de la cláusula “**SELECT**” [***Ver sección 4.1.2***] y los busca en la base de datos **"baseconsultas"** para determinar si dichos sustantivos hacen referencia a una tabla o columna y utiliza esta información para identificar a que columnas está haciendo referencia el usuario en la consulta.

La interfaz toma los valores de la cláusula “WHERE” que haya introducido el usuario y busca estos valores en la base de datos tomando las tablas que ya fueron encontradas en la cláusula “SELECT”, en donde además de encontrar a que columnas hace referencia el valor o los valores de la consulta, se agregan columnas a la cláusula “SELECT”.

Se añade también la función de agregación si en la cláusula “SELECT” se encontró una palabra que indique el uso de “COUNT”, “MAX” o “MIN”.

4.3. Reglas para mostrar el diálogo.

Analizando el funcionamiento actual de la interfaz [***Ver sección 4.1***] se han construido 5 reglas que indicarán cuando se debe iniciar un diálogo con el usuario, dichas reglas se describen a continuación.

Regla 1.

Si el número de “sustantivos campo”, es decir sustantivos que hagan referencia a una columna de la base de datos en la consulta es menor al número de campos resultantes encontrados por la interfaz en la cláusula “SELECT”, se inicia un diálogo con el usuario para que decida que columnas desea que aparezcan en la respuesta final.

Regla 2.

Si el número de valores encontrados en la consulta es menor al número de campos resultantes encontrados por la interfaz en la cláusula “WHERE”, se inicia un diálogo con el usuario para que decida a que columna o columnas hace referencia el valor de la consulta.

Regla 3.

En algunos casos después de analizar la cláusula “WHERE”, si existen tablas que se necesitan en la consulta final y que no se encontraron en la cláusula “SELECT” se agregan nuevas columnas a las tablas resultados.

Si el número de columnas encontradas inicialmente en la cláusula “SELECT” es menor al número de columnas en el “SELECT” al agregarse las columnas encontradas después de analizar la cláusula “WHERE”, se inicia un diálogo con el usuario para que decida que columnas desea que aparezcan en la respuesta final.

Regla 4.

Si se encontró que la consulta utiliza una función de agregación y los campos resultantes encontrados por la interfaz son mayores que 1, se inicia un diálogo con el usuario para que decida a cuál de las columnas encontradas se le aplicará la función de agregación.

Regla 5.

Si la consulta actual no contiene “Sustantivos campo” es decir sustantivos en la consulta que hagan referencia a una columna de la base de datos y los campos

resultantes encontrados por la interfaz son mayores que 1, se inicia un diálogo con el usuario para que decida que columnas desea que aparezcan en la respuesta final.

En la fase de traducción se aplicaran estas reglas, de acuerdo al funcionamiento actual de la interfaz [***Ver sección 4.1***], que describen en el diagrama siguiente.

A continuación se muestra un diagrama que muestra cómo se incluyeron estas reglas en la fase de traducción de la interfaz.

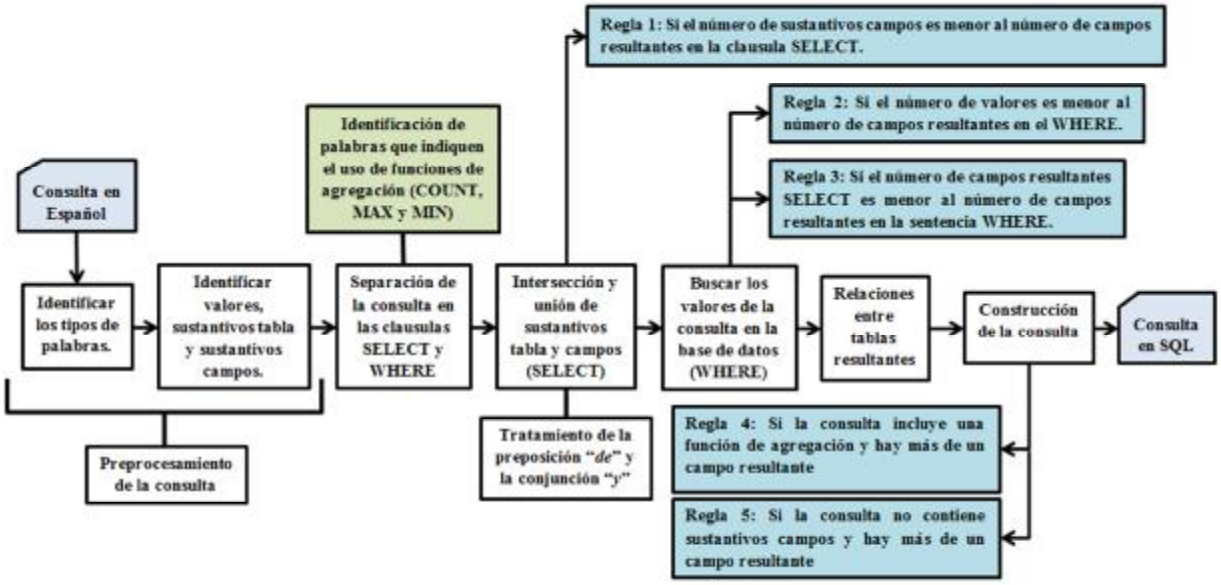


Figura 15.- Detalle de la arquitectura y reglas para mostrar el diálogo.

Cuando se cumple alguna de estas reglas se inicia un diálogo con el usuario, y si una consulta cumple con varias reglas durante la fase de traducción, la interfaz mostrará los diálogos necesarios para aclarar la información y mejorar la calidad de la consulta.

4.4. Tratamiento de consultas que incluyan funciones de agregación.

Cuando se requiere que una consulta devuelva el número de registros que cumplan con cierta condición, el registro con valor máximo o el valor mínimo de alguna consulta, es cuando se utilizan las funciones de agregación “COUNT”, “MAX” y “MIN” respectivamente.

Se analizará la consulta para separar la cláusula “SELECT” de la cláusula “WHERE”, una vez separada la consulta se buscará en la cláusula “SELECT” palabras clave que indiquen que es necesario el uso de alguna de las tres funciones de agregación mencionadas anteriormente.

Para lograr identificar que una consulta utilizará las funciones de agregación “Max”, “Min” y “Count” se utilizarán las palabras clave que a continuación se muestran en la tabla:

Función de Agregación	**Patrón de identificación en el idioma español**	**Palabra Raíz**
Max	máximo, máximos, máxima, máximas, grande, grandes, superlativo, mayor, mayores, extremo, limite, tope, superior, alto, mas	Mayor
Min	mínimo, mínimos, mínima, mínimas, chico, chicos, escaso, menor, minúsculo, menor, menores, pequeño, pequeños, pequeña, pequeñas, reducido, inferior, bajo, menos	Menor
Count	Cantidad, cantidades, cuenta, conjunto, cuantía, medida, tanto, total, numero.	Cantidad

Tabla 15. - Palabras clave para identificar el uso de funciones de agregación.

Se modificó la base de datos “Lexicon” en donde se agregó la tabla llamada “funcAgregacion” que contendrá las palabras clave para identificar la utilización de funciones de agregación y tendrá los siguientes campos:

- **Palabra**: cada una de las palabras clave.
- **Categoría**: Clave que identifica que se trata de una palabra relacionada con la función de agregación “max”, “min” o “count”.
- **Raíz**: Palabra que se utilizará para estandarizar cada palabra.

En el caso de “**max**” y “**min**” se utilizará la clave “**ADJCP**” que significa “*comparative adjective*” y se refiere a un adjetivo que indica el mayor o menor.

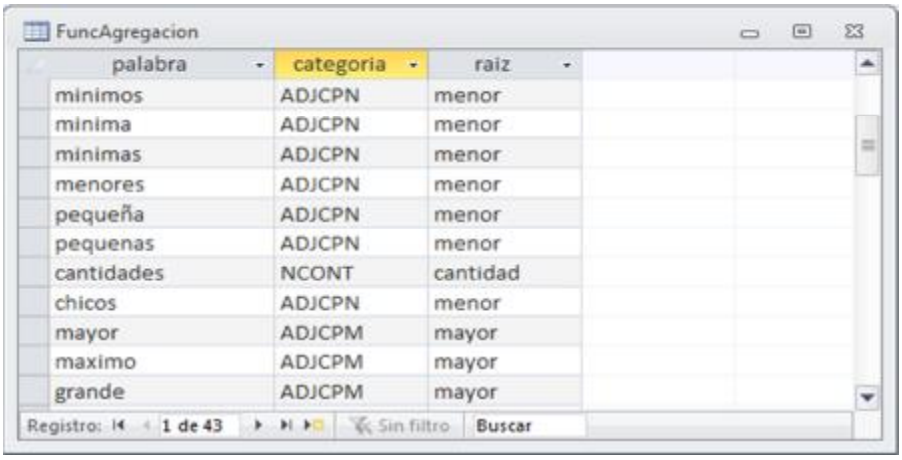
Para distinguir estas palabras clave se agregará a esta una “**M**” cuando haga referencia al mayor y se agregará una “**N**” cuando haga referencia al menor.

En el caso de “**count**” se utilizará la clave “**NCONT**” que significa “*Countable Noun*” y se refiere a las consultas que requieren devolver la cantidad de registros que cumplen con una condición determinada.

Por lo tanto las claves de categoría que van a utilizar quedarían de la siguiente forma:

- **ADJCPM**: Cuando hace referencia al máximo (Función “**max**”).
- **ADJCPN**: Cuando hace referencia al mínimo (Función “**min**”).
- **NCONT**: Cuando hace referencia a la cantidad (Función “**count**”).

Estos datos se agregaron a la tabla “**funcAgregacion**” de la base de datos “**Lexicon**”, en la siguiente imagen se muestra la vista de dicha tabla:


FuncAgregacion

palabra	categoria	raiz
minimos	ADJCPN	menor
minima	ADJCPN	menor
minimas	ADJCPN	menor
menores	ADJCPN	menor
pequeña	ADJCPN	menor
pequenas	ADJCPN	menor
cantidades	NCONT	cantidad
chicos	ADJCPN	menor
mayor	ADJCPM	mayor
maximo	ADJCPM	mayor
grande	ADJCPM	mayor

Registro: 1 de 43 Sin filtro Buscar

Figura 16.- Tabla funcAgregacion de la BD lexicón.

La interfaz toma la consulta en lenguaje natural como entrada y a cada palabra de dicha consulta se le asigna una etiqueta que indica el tipo de palabra.

La interfaz está desarrollada en lenguaje **JAVA**, y está compuesta de varias clases. La clase en donde se realizan las asignaciones de las etiquetas se llama "**de3.java**", la cual se modificó para que identifique las funciones de agregación mencionadas.

La asignación de etiquetas se realiza en el método "**escribir**" de la clase "**de3.java**". En este código se realiza un ciclo en donde se va tomando de cada una de las palabras en la consulta de entrada. Dentro de este método se consulta la base de datos "**lexicon**".

Primero se busca la palabra en la tabla "**sustantivos**", si se encuentra la palabra se le asigna la etiqueta "**s**".

Si no se encuentra en esta tabla se busca en la tabla "**lexico**", si se encuentra se asigna como etiqueta el campo "**categoría**" del registro encontrado.

Después se incluye la consulta a la tabla "**funcAgregacion**", que tiene la misma estructura de las otras dos tablas, "sustantivos" y "lexico", en donde se buscan las palabras claves descritas anteriormente y se le asigna como etiqueta el campo "**categoría**" del registro encontrado.

Si no se encuentra en ninguna tabla se comprueba si es un número y se etiqueta como "**num**", si no es un número se considera atributo y se etiqueta como "**a**".

A continuación se muestra una comparación entre el algoritmo original del Dr. Juan Javier González Barbosa y el realizado en esta tesis, que se encuentra en el método "**escribir**" de la clase "**de3.java**", en donde se realiza esta asignación de etiquetas a las palabras.

El algoritmo del ciclo principal del método es el siguiente:

Código original	Código Modificado
Ciclo (Por cada palabra de la consulta)	Ciclo (Por cada palabra de la consulta)
Rs=ConsultaTablaSustantivos(Palabra)	Rs=ConsultaTablaSustantivos(Palabra)
Si se encontró la palabra	Si se encontró la palabra
Etiqueta = "s"	Etiqueta="s"
Si no se encontró	Si no se encontró
Rs=ConsultaTablaLexico(Palabra)	Rs=ConsultaTablaLexico(Palabra)
Si se encontró la palabra	Si se encontró la palabra
Etiqueta=Rs.categoria	Etiqueta=Rs.categoria
Si no se encontró en ninguna tabla	**Si no se encontró la palabra**
EsNumero=ComprobarSiEsNumero	
Si es número	**Rs=ConsultaTablaFuncAgregacion(Palabra)**
Etiqueta="num"	Si no se encontró en ninguna tabla
Si no es número.	EsNumero=ComprobarSiEsNumero
Etiqueta="a" //Se considera atributo	Si es número
Fin del ciclo.	Etiqueta="num"
	Si no es número.
	Etiqueta="a" //Se considera atributo
	Fin del ciclo.

Tabla 16.- Comparación del código del método escribir.

Una vez que se identificó que la consulta actual utiliza la función de agregación "COUNT", "MAX" o "MIN", se analizan las cláusulas "SELECT" y "WHERE" para determinar cuáles son los campos que el usuario desea obtener en dicha consulta y si los campos encontrados son más de uno se muestra un diálogo al usuario para que pueda determinar a cuál de las columnas resultantes se debe aplicar la función de agregación, dado que dichas funciones solo se pueden aplicar a una sola columna.

Capítulo 5

Implementación.

La interfaz esta implementada en lenguaje “**JAVA**”, y se compone de 35 clases, de las cuáles se modificó principalmente la clase “**Grafo.java**” en donde se implementaron las reglas y se añadió una nueva clase llamada “**Dialogo.java**” que es la que se encarga de establecer el diálogo con el usuario, la relación entre estas dos clases se muestra en la siguiente imagen.

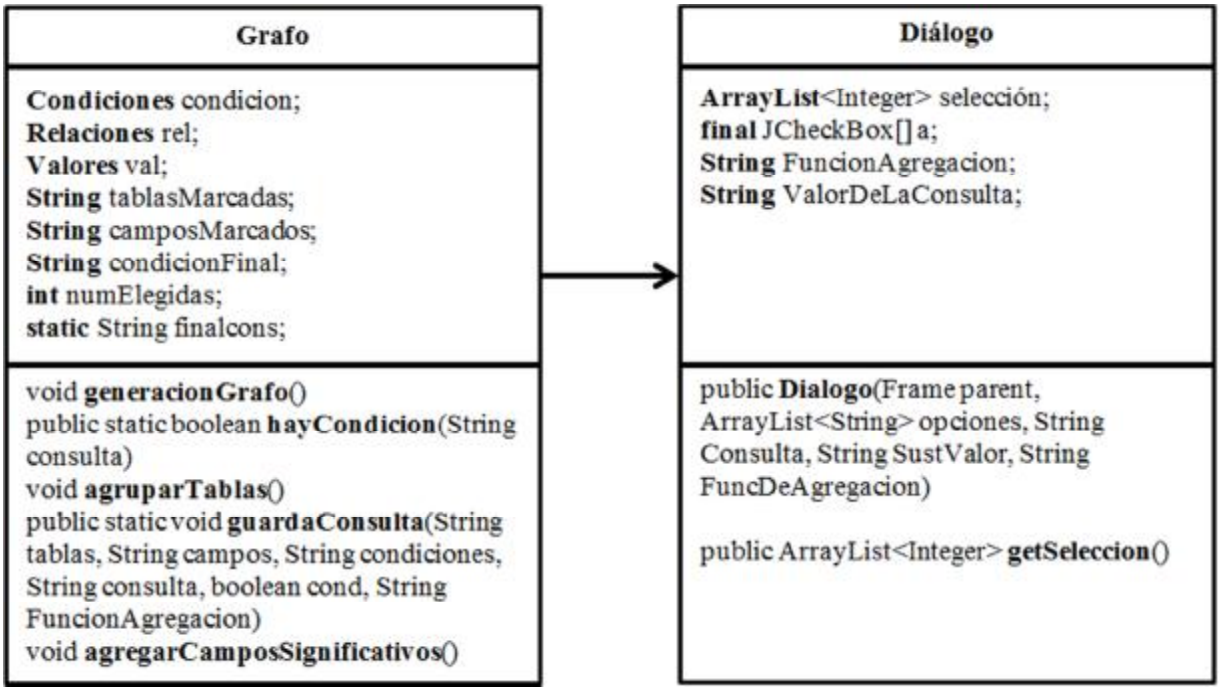


Figura 17.- Diagrama UML de la clase Grafo y la clase diálogo

En la clase “**Grafo.java**” se implementaron las reglas, si una de las reglas se cumple se muestra un diálogo al usuario, llamando a la clase “**Dialogo.java**” que recibe la consulta y las opciones que se mostrarán al usuario y de las cuales elegirá la que le

parezca más adecuada. Esta clase construye el cuadro de diálogo y regresa a la clase "**Grafo.java**" la opción que el usuario eligió.

En base a la respuesta obtenida del usuario se modifica la consulta SQL construida por la interfaz eliminando todas las tablas y columnas que no hayan sido elegidas por el usuario en el diálogo, eliminando la información de más y la ambigüedad en algunos casos.

El código de la clase "**Dialogo.java**" se encuentra en el Anexo F de esta tesis.

La interfaz está diseñada para ejecutarse en el sistema operativo **Windows XP** y directamente desde el **compilador** de "JAVA".

En este trabajo se modificó la interfaz para lograr ejecutarse en **Windows 7** de **64 bits** utilizando el IDE **NetBeans 7.0** o superior.

Entre las modificaciones principales se encuentra el establecimiento de la propiedad "**charSet**" como "**iso-8859-1**", que le indica a la interfaz que se utiliza el lenguaje latinoamericano permitiendo el uso de caracteres como la "**ñ**" y palabras acentuadas.

La interfaz utiliza algunas librerías para su funcionamiento que se encuentran en el archivo "**jdsl.zip**" y que es necesario agregar al proyecto de "NetBeans" [**Ver** Anexo G, sección 2.3].

Es necesario agregar los orígenes de datos que le indican a la interfaz cuáles son las bases de datos que le ayudarán a la traducción de la consulta [**Ver** Anexo G, sección 4].

El proceso completo de instalación y configuración de la interfaz se encuentra en el Anexo G de este trabajo en donde se explica paso a paso el procedimiento para ejecutar la interfaz [**Ver** Anexo G].

Como se explicó en la sección 4.2 y 4.3 de este trabajo, Se construyeron 5 reglas que indican en qué casos, se iniciará un diálogo con el usuario. A continuación se muestran dichas reglas y una breve descripción de cada una de ellas.

Regla	Descripción
1	Si el número de "sustantivos campos" es menor al número de campos resultantes en la cláusula SELECT.
2	Si el número de valores en la consulta es menor al número de campos resultantes en la cláusula WHERE.
3	Si el número de campos resultantes en la cláusula SELECT es menor al número de campos resultantes en la cláusula WHERE.
4	Si la consulta incluye una función de agregación y el total de campos resultantes en la consulta en mayor a 1.
5	Si la consulta no contiene "sustantivos campos" y el total de campos resultantes en la consulta en mayor a 1.

Tabla 17.- Reglas de la interfaz para mostrar el diálogo.

Las primeras 3 reglas son independientes entre sí y al momento de la traducción pueden haber más de un diálogo con el usuario. La regla 4 es excluyente con la regla 1, la regla3 y la regla 5. La regla 5 es excluyente con la regla 1, la regla3 y la regla 4. Ya que si la función contiene una función de agregación o si no existe ningún campo que haga referencia a un campo solo se puede complementar con la regla 2.

A continuación se mostrará un ejemplo de cada una de las reglas, en donde se aplica a una consulta específica, se muestra el funcionamiento de la interfaz, la implementación y en qué parte de la fase de traducción se manda a llamar al diálogo.

Para cada regla se utilizará una consulta de ejemplo para mostrar el proceso de traducción de la interfaz, las tablas de las bases de datos que se utilizan y los diálogos mostrados al usuario en cada uno de los casos.

5.1.- Implementación de la Regla 1.

La regla 1 indica que si el número de palabras de la consulta que hacen referencia a un campo de la base de datos es menor al número de campos encontrados por la interfaz, se mostrara un diálogo al usuario para aclarar la respuesta.

En este caso la consulta de ejemplo es: "***Muestra los nombres de los clientes de CANADÁ***".

Al realizar el análisis de la consulta [**Ver** secciones 4.1.1 – 4.1.6], el tratamiento de preposiciones y las intersecciones, los campos encontrados se guardan en la tabla "**ResultadoCampos**" y la tabla en "**ResultadoTabla**", por lo tanto ambas tablas según el ejemplo quedarían de la siguiente manera.

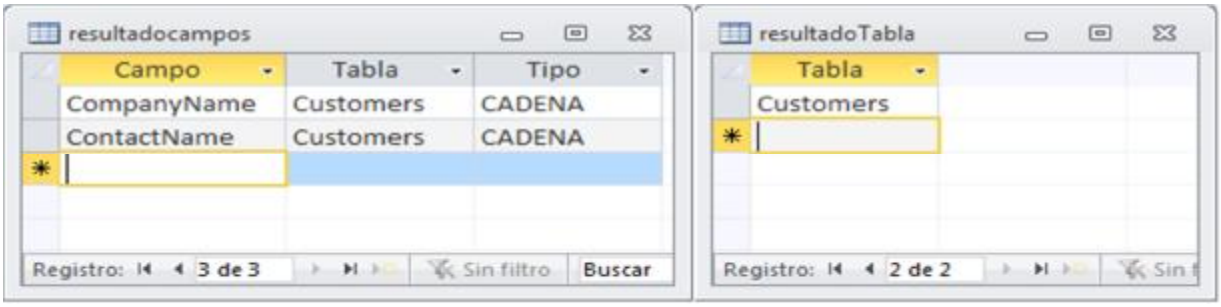


Figura 18.- Tablas resultado del ejemplo de la Regla 1.

Hasta este momento en la cláusula "SELECT", se tienen identificados dos campos, resultado de las intersecciones de los "sustantivos campos" y los "sustantivos tabla" de la consulta, por lo tanto la cláusula "SELECT" quedaría de la siguiente forma:

```
SELECT Customers.CompanyName, Customers.ContactName
FROM Customers
```

En esta fase de la traducción es en donde se implementó la **Regla 1** que se encuentra en la clase "**Grafo.java**", y que realiza las siguientes acciones.

Se realiza una consulta a la tabla "**ResultadosCampos**" para obtener los campos encontrados y se guardan los resultados en un arreglo.

Se realiza una consulta a la tabla "**ConsultaLN**" y se obtiene el número de registros que contengan en el campo "**categoria**" el valor "**s**" (es decir, que son sustantivos) y que tengan en el campo "**penalizacion**" el valor **1**, que significa que la palabra hace referencia a un campo de la base de datos que se consulta.

Se compara el número de campos encontrados y el número de registros que hacen referencia a un campo de la base de datos que se consulta (que debe ser mayor a cero), y si es mayor el número de campos encontrados se cumple la regla 1, como en este ejemplo y por lo tanto se muestra un diálogo al usuario.

Para mostrar el diálogo se utiliza la clase "**Dialogo.java**", que obtiene la consulta original mostrándola al usuario en un cuadro de diálogo y muestra los campos encontrados por la interfaz para permitir al usuario elegir cuál campo o campos desea ver en la consulta final.

El mensaje que se mostrará en el cuadro de diálogo se establece de la siguiente manera:

Se establece en la interfaz que se está evaluando la regla 1, que modifica la cláusula "**SELECT**" de la consulta final.

Por lo tanto el mensaje que muestra al usuario es el siguiente: ***"¿Qué es lo que desea obtener en esta consulta?***" y se muestran a continuación los campos encontrados hasta el momento por la interfaz, permitiendo mediante casillas de verificación elegir los campos que desea obtener.

Por cada campo encontrado por la interfaz se debe obtener su descripción en español, esta información se encuentra en la BD "**Metadatos**" en la tabla del mismo nombre. Por lo tanto se hace una consulta a esta tabla, se extraen las descripciones y se agregan a cada campo.

En este caso en la consulta se busca el nombre de los clientes, por lo tanto la única columna que debe mostrarse es "**Customers.ContactName**".

Como se muestra en este ejemplo se cumple con la regla 1 y el diálogo mostrado al usuario quedaría de la siguiente manera.

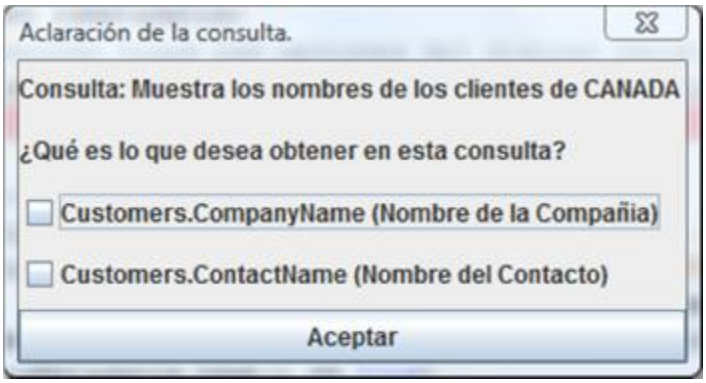


Figura 19.- Diálogo de cláusula SELECT del ejemplo de la Regla 1.

La consulta final quedaría de la siguiente forma.

```
SELECT  Customers.ContactName
FROM Customers
```

Se realiza el análisis de la cláusula "**WHERE**", que en el ejemplo actual es "**de CANADA**", que no contiene ninguna palabra que haga referencia a un campo o una tabla.

Se toma el valor "**CANADÁ**" y se busca en todas las columnas de la tabla que hace referencia el sustantivo tabla de la cláusula "SELECT", es decir en la tabla "**Customers**".

Customers

CustomerID	CompanyName	ContactName	ContactTitle	Address	City	Region	PostalCode	Country	Phone	Fax
ALFKI	Alfreds Futterkiste	Maria Anders	Sales Representative	Obere Str. 57	Berlin		12209	Germany	030-0074321	030-0076545
ANATR	Ana Trujillo Emparedados	Ana Trujillo	Owner	Avda. de la Cons	México D.F.		05021	Mexico	(5) 555-4729	(5) 555-3745
ANTON	Antonio Moreno Taquería	Antonio Moreno	Owner	Mataderos 2312	México D.F.		05023	Mexico	(5) 555-3932	
AROUT	Around the Horn	Thomas Hardy	Sales Representative	120 Hanover Sq.	London		WA1 1DP	UK	(171) 555-7788	(171) 555-6750
BERGS	Berglunds snabbköp	Christina Berglun	Order Administrator	Berguvsvägen 8	Luleå		S-958 22	Sweden	0921-12 34 65	0921-12 34 67
BLAUS	Blauer See Delikatessen	Hanna Moos	Sales Representative	Forsterstr. 57	Mannheim		68306	Germany	0621-08460	0621-08924
BLONP	Blondesddsl père et fils	Frédérique Citea	Marketing Manager	24, place Kléber	Strasbourg		67000	France	88.60.15.31	88.60.15.32
BOLID	Bólido Comidas preparad	Martín Sommer	Owner	C/ Araquil, 67	Madrid		28023	Spain	(91) 555 22 82	(91) 555 91 99
BONAP	Bon app'	Laurence Lebihar	Owner	12, rue des Bouc	Marseille		13008	France	91.24.45.40	91.24.45.41
BOTTM	Bottom-Dollar Markets	Elizabeth Lincoln	Accounting Manager	23 Tsawassen Bl	Tsawassen	BC	T2F 8M4	Canada	(604) 555-4729	(604) 555-3745
BSBEV	B's Beverages	Victoria Ashworth	Sales Representative	Fauntleroy Circu	London		EC2 5NT	UK	(171) 555-1212	

Registro: 10 de 91 | Sin filtro | Buscar

Figura 20.- Tabla "Customers" de la base de datos Northwind para el ejemplo de Regla 1.

Como se puede ver la única columna en donde se encuentra el valor "**CANADÁ**" es en "**Country**" por lo tanto es la que utiliza para construir la condición "**Customers. Country** LIKE '**CANADÁ**".

Como solo se encontró un campo para la condición "**WHERE**" y solo hay un valor en la sentencia introducida por el usuario **no se cumple la Regla 2**.

Ya que en la consulta original si existe al menos una palabra que haga referencia a un campo de la base de datos que se consulta, **no se cumple la Regla 5**.

La consulta actual, no fue identificada por la interfaz como una consulta que utilice las funciones de agregación "Max", "Min" o "Count" [Ver sección 4.4] y por lo tanto **no se cumple la regla 4**.

La tabla en la que se encontró el valor "**CANADA**" de la consulta de ejemplo, es la misma que la encontrada en el análisis de la cláusula "SELECT" y por lo tanto no se agregan campos a la tabla "ResultadosCampo" y **no se cumple la Regla 3**.

Por lo tanto después de realizar el diálogo con el usuario y el análisis de la cláusula "WHERE" la consulta final quedaría de la siguiente forma:

```
SELECT Customers.ContactName
FROM Customers
WHERE ((Customers.Country LIKE 'CANADA'))
```

Y con esto finaliza el análisis de la consulta.

5.2.- Implementación de la Regla 2.

La **regla 2** indica que si el número de palabras de la consulta que se identifican como valores es menor al número de campos encontrados por la interfaz al analizar la cláusula “WHERE”, se mostrara un diálogo al usuario para aclarar la respuesta.

Consulta: ***Muéstrame el empleado con identificador H-B39728F***

Al realizar el análisis de la consulta [**Ver** secciones 4.1.1 – 4.1.6] en la cláusula “SELECT”, el tratamiento de preposiciones y las intersecciones, hasta este momento en la cláusula “SELECT”, se tienen identificados tres campos, resultado de las intersecciones de los “sustantivos campos” y los “sustantivos tabla” de la consulta, por lo tanto la cláusula “SELECT” quedaría de la siguiente forma:

```
SELECT employee.emp_id, employee.fname, employee.lname
FROM employee
```

En este caso, como en la consulta original solo se tiene un “sustantivo campo” y los campos resultantes en la cláusula “SELECT” son tres, se inicia un diálogo con el usuario porque se cumple con la **Regla 1**, del diálogo.

Como se vio en el ejemplo anterior, se construye el diálogo que se mostrara al usuario en la clase “**Dialogo.java**”. Se establece el mensaje que se mostrara, las opciones que son los campos encontrados por la interfaz en la cláusula “**SELECT**” y se obtienen las descripciones en español de cada campo, obteniéndolos de la **BD** “**Metadatos**” consultando la tabla del mismo nombre.

A continuación se muestra un ejemplo del diálogo que muestra la interfaz.

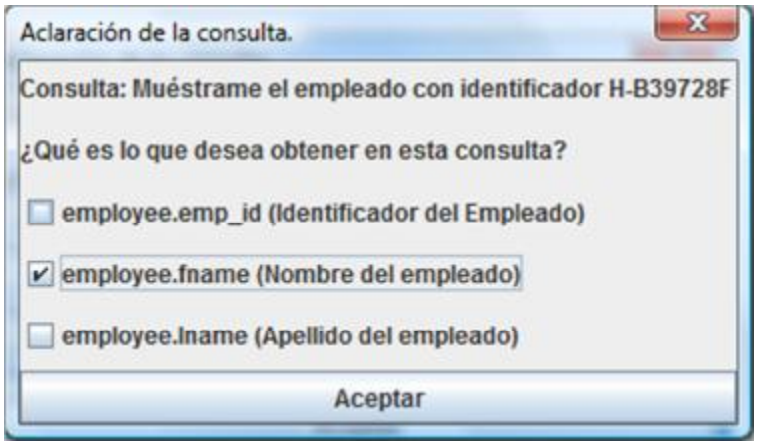


Figura 21.- Diálogo de cláusula SELECT del ejemplo de la Regla 2.

En este caso supongamos que el usuario desea ver el nombre del empleado, por lo tanto se elige el campo "**employee.fname**" y la consulta quedaría de la siguiente forma:

```
SELECT employee.fname
FROM employee
```

Una vez que ya se tienen los campos de la cláusula "SELECT" se procede a analizar la cláusula "WHERE", en donde se toman los "sustantivos campos", los "sustantivos tabla" y el valor o los valores de la consulta.

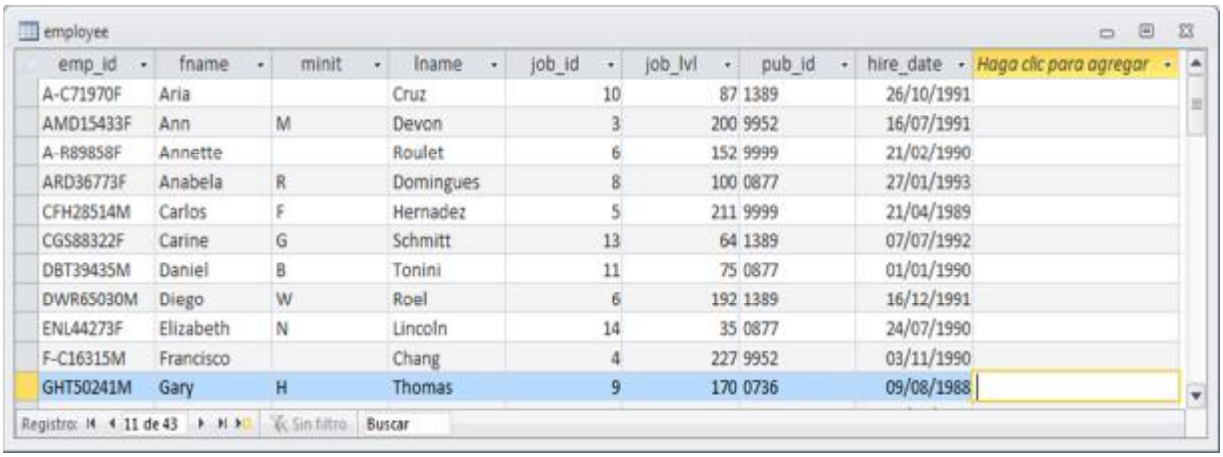
En esta fase de la traducción es en donde se implementó la **Regla 2** que se encuentra en la clase "**Grafo.java**", y que realiza las siguientes acciones.

Se realiza una consulta a la tabla "**ConsultaLN**" y se obtiene el número de registros que contengan en el campo "**categoria**" el valor "**a**, que significa que la palabra hace referencia a un valor dentro de la base de datos que se consulta. En este caso el valor encontrado en la consulta es "**H-B39728F**".

En el ejemplo actual en la cláusula "WHERE", que es "***con identificador H-B39728F***" no existe un sustantivo que haga referencia a una tabla, se obtiene el sustantivo tabla de la cláusula "SELECT", que en este caso es "**Empleados**" y hace referencia a la tabla "**employee**".

También se busca en la cláusula “WHERE” un sustantivo que haga referencia a un campo, en este ejemplo dicha palabra es “**identificador**”.

En la tabla encontrada “**employee**” se busca el “**sustantivo campo**” que es “**Identificador**”, es decir devuelve todas las columnas de la tabla “employee” que estén relacionadas a la palabra “identificador”. A continuación se muestra la tabla “employee”.



employee

emp_id	fname	minit	lname	job_id	job_lvl	pub_id	hire_date	Haga clic para agregar
A-C71970F	Aria		Cruz	10	87	1389	26/10/1991	
AMD15433F	Ann	M	Devon	3	200	9952	16/07/1991	
A-R89858F	Annette		Roulet	6	152	9999	21/02/1990	
ARD36773F	Anabela	R	Domingues	8	100	0877	27/01/1993	
CFH28514M	Carlos	F	Hernadez	5	211	9999	21/04/1989	
CGS88322F	Carine	G	Schmitt	13	64	1389	07/07/1992	
DBT39435M	Daniel	B	Tonini	11	75	0877	01/01/1990	
DWR65030M	Diego	W	Roel	6	192	1389	16/12/1991	
ENL44273F	Elizabeth	N	Lincoln	14	35	0877	24/07/1990	
F-C16315M	Francisco		Chang	4	227	9952	03/11/1990	
GHT50241M	Gary	H	Thomas	9	170	0736	09/08/1988	

Registros: 11 de 43 | Sin filtro | Buscar

Figura 22.- Tabla "employee" de la base de datos Pubs.

Como se puede ver hay tres columnas de la tabla que están relacionadas al identificador que son “**employee.emp_id**”, “**employee.job_id**” y “**employee.pub_id**”.

Por lo tanto el valor “**H-B39728F**” se iguala a estas tres columnas y la consulta queda de la siguiente forma.

```
SELECT employee.fname
FROM employee
WHERE ((employee.emp_id LIKE '%H-B39728F%' OR employee.job_id
LIKE '%H-B39728F%' OR employee.pub_id LIKE '%H-B39728F%'))
```

Se compara el número de campos encontrados al analizar la cláusula “**WHERE**” (que son 3) y el número de **valores** encontrados en la consulta ingresada por el usuario, y si es mayor el número de campos encontrados se cumple la **regla 2**, como en este ejemplo y por lo tanto se muestra un diálogo al usuario.

Para mostrar el diálogo se utiliza la clase "**Dialogo.java**", que obtiene la consulta original mostrándola al usuario en un cuadro de diálogo y muestra los campos encontrados por la interfaz para permitir al usuario elegir cuál campo o campos desea ver en la consulta final.

El mensaje que se mostrará en el cuadro de diálogo se establece de la siguiente manera:

Se establece en la interfaz que se está evaluando la regla 2, que modifica la cláusula "**WHERE**" de la consulta final.

Por lo tanto el mensaje que muestra al usuario es el siguiente: ***"¿A qué columna hace referencia con el valor*** **H-B39728F*?***", que es el encontrado en la consulta y se muestran a continuación los campos encontrados hasta el momento por la interfaz, permitiendo mediante casillas de verificación elegir los campos que desea obtener.

Por cada campo encontrado por la interfaz se debe obtener su descripción en español, esta información se encuentra en la BD "**Metadatos**" en la tabla del mismo nombre. Por lo tanto se hace una consulta a esta tabla, se extraen las descripciones y se agregan a cada campo.

En este caso en la consulta el valor "**H-B39728F"** hace referencia al identificador del empleado, por lo tanto la única columna que debe mostrarse es "**Employee.emp_id**".

Ya que en la consulta original si existe al menos una palabra que haga referencia a un campo de la base de datos que se consulta, **no se cumple la Regla 5**.

La consulta actual, no fue identificada por la interfaz como una consulta que utilice las funciones de agregación "Max", "Min" o "Count" [Ver sección 4.4] y por lo tanto **no se cumple la regla 4**.

La tabla en la que se encontró el valor “**H-B39728F**” de la consulta de ejemplo, es la misma que la encontrada en el análisis de la cláusula “SELECT” y por lo tanto no se agregan campos a la tabla “ResultadosCampo” y **no se cumple la Regla 3**.

Como se muestra en este ejemplo se cumple con la regla 2 y el diálogo mostrado al usuario quedaría de la siguiente manera.

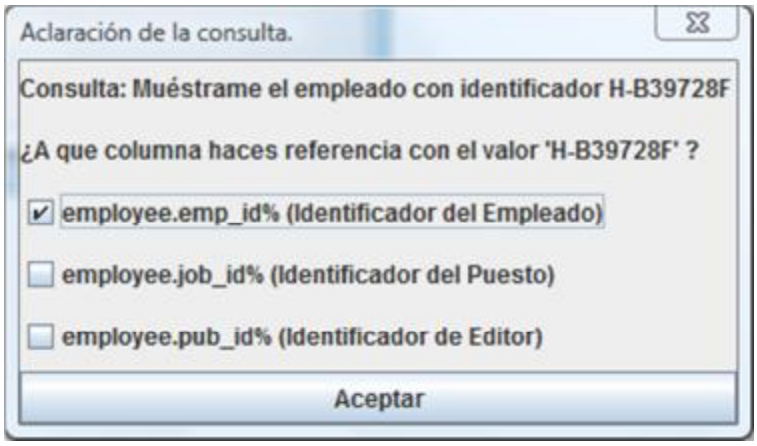


Figura 23.- Diálogo de cláusula WHERE del ejemplo de la Regla 2.

Por lo tanto después de realizar el diálogo con el usuario y el análisis de la cláusula “WHERE” la consulta final quedaría de la siguiente forma:

```
SELECT employee.fname
FROM employee
WHERE ((employee.emp_id LIKE '%H-B39728F%'))
```

De esta forma finaliza el análisis de la consulta.

5.3.- Implementación de la Regla 3.

La tercera regla se aplica cuando el número de campos resultantes en la intersección en la cláusula SELECT es menor al número de campos resultantes al realizar el análisis de la cláusula WHERE.

Consulta: ***muéstrame todos los nombres del producto cuya categoría es beverages***

Al realizar el análisis de la consulta [**Ver** secciones 4.1.1 – 4.1.6] en la cláusula "SELECT", el tratamiento de preposiciones y las intersecciones, hasta este momento en la cláusula "SELECT", se identificó un solo campo, resultado de las intersecciones de los "sustantivos campos" y los "sustantivos tabla" de la consulta, por lo tanto la cláusula "SELECT" quedaría de la siguiente forma:

```
SELECT Products.ProductName
FROM Categories, Products
```

En este caso, en la consulta original solo se tiene un "sustantivo campo" y los campos resultantes en la cláusula "SELECT" también son solo una "Products.ProductName", por lo tanto **no se cumple** con la **Regla 1**, del diálogo.

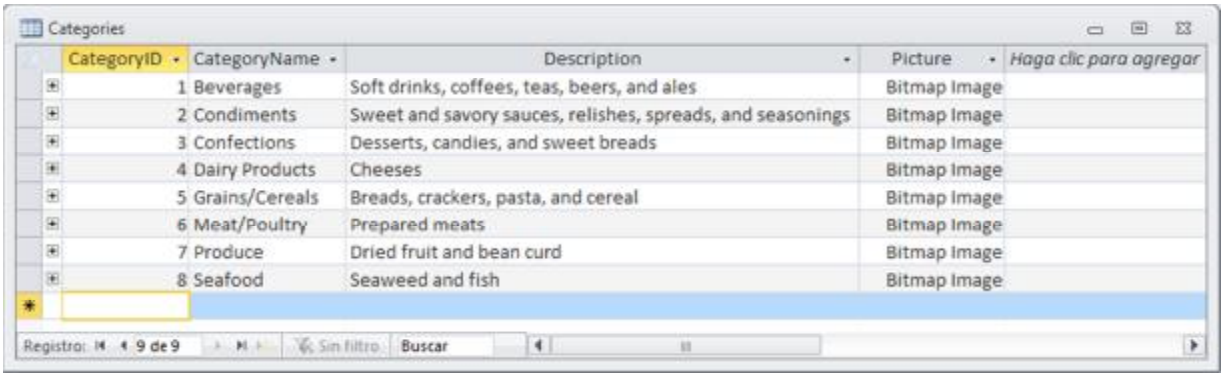
Una vez que se analizó la cláusula "SELECT" se procede a analizar la cláusula "WHERE", en donde se toman los "sustantivos campos", los "sustantivos tabla" y el valor o los valores de la consulta en la cláusula "WHERE".

Se realiza una consulta a la tabla "**ConsultaLN**" y se obtiene el número de registros que contengan en el campo "**categoria**" el valor "**a**, que significa que la palabra hace referencia a un valor dentro de la base de datos que se consulta. En este caso el valor encontrado en la consulta es "**beverages**".

En el ejemplo actual en la cláusula "WHERE", que es "***cuya categoría es beverages***" si existe un sustantivo que haga referencia a una tabla, que en este

caso en “***categoría***” y hace referencia a la tabla “**Categories**”, no hay sustantivos que hagan referencia a un campo.

Se toma el valor “**beverages**” y se busca en todas las columnas de la tabla que hace referencia el sustantivo tabla, es decir en la tabla “**Categories**”.



Categories

CategoryID	CategoryName	Description	Picture	Haga clic para agregar
1	Beverages	Soft drinks, coffees, teas, beers, and ales	Bitmap Image	
2	Condiments	Sweet and savory sauces, relishes, spreads, and seasonings	Bitmap Image	
3	Confections	Desserts, candies, and sweet breads	Bitmap Image	
4	Dairy Products	Cheeses	Bitmap Image	
5	Grains/Cereals	Breads, crackers, pasta, and cereal	Bitmap Image	
6	Meat/Poultry	Prepared meats	Bitmap Image	
7	Produce	Dried fruit and bean curd	Bitmap Image	
8	Seafood	Seaweed and fish	Bitmap Image	

Registro: 9 de 9 Sin filtro Buscar

Figura 24.- Tabla "Categories" de la base de datos Northwind.

Como se puede ver la única columna en donde se encuentra el valor “**Beverages**” es en “**CategoryName**” por lo tanto es la que se agrega a la tabla “**ResultadoCampos**” y se construye la condición “***Categories.CategoryName LIKE ‘beverages’***”.

Se compara el número de campos encontrados al analizar la cláusula “**WHERE**” (que solo es 1) y el número de **valores** encontrados en la consulta ingresada por el usuario que también es 1, por lo tanto en este ejemplo **no cumple la regla 2**, que indica que se mostrara un mensaje al usuario si el número de valores es mayor a número de campos encontrados al analizar la cláusula “WHERE”.

Una vez analizada la cláusula “**WHERE**” y como existen varias tablas en “ResultadoTabla” se obtienen estas tablas y también se obtienen las tablas de “Resultadocampos”, se comparan y se obtienen las tablas que si están en “ResultadoTabla” y que no se encuentran en “ResultadoCampos”, que en este caso sería la tabla “**Categories**”.

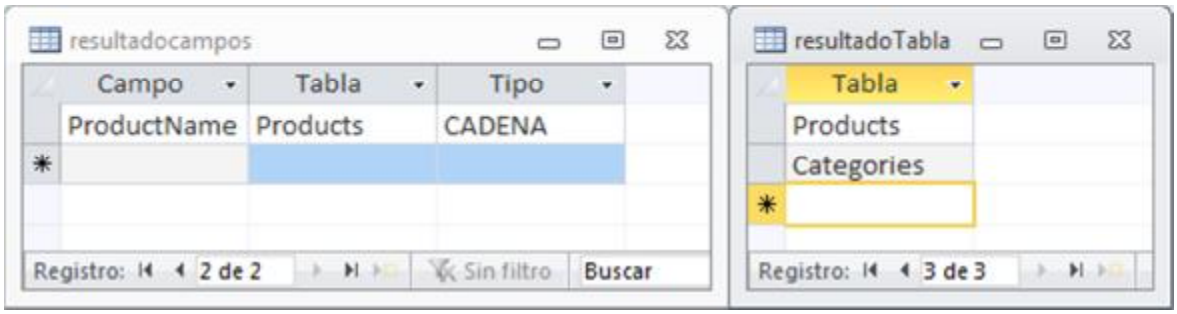


Figura 25.- Tablas resultado del ejemplo de la regla 3.

Haciendo una consulta a la base de datos actual, en la tabla "**__infoTablas**", se obtiene la descripción de la tabla que en este caso es "**Categories**" que en el campo descripción tiene "**Categoria**" y esta palabra se busca en la columna "**DESCRIPCOLUM**" de la tabla "**Metadatos**", es decir se obtienen todos los registros que en su descripción contengan la palabra "**Categoría**" y donde la columna "**TABLENAME**" coincida con "**Categories**".

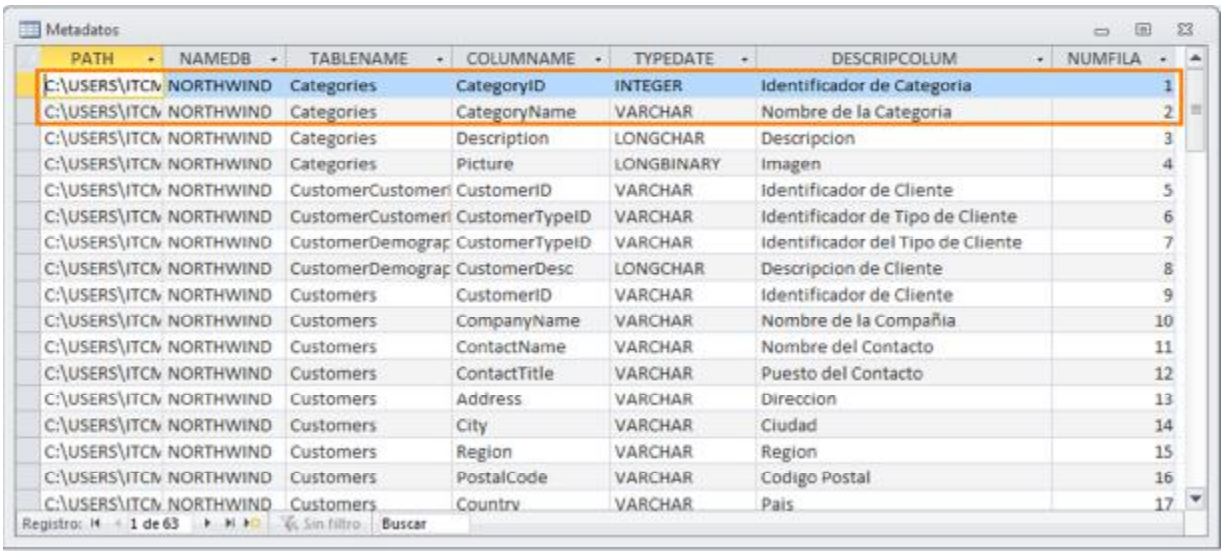


Figura 26.- Tabla "Metadatos" de la base de datos del mismo nombre.

Como se observa en la tabla, se encuentran las columnas "**Categories.CategoryID**" y "**Categories.CategoryName**" que cumplen con estos criterios, por lo tanto estas

columnas se agregan a los campos resultado, por lo tanto los campos resultados de las cláusulas “**SELECT**” y “**WHERE**” quedan de la siguiente forma:

Columnas elegidas en el análisis de la cláusula SELECT	Columnas elegidas en el análisis de la cláusula WHERE
Products.ProductName	Products.ProductName
	Categories.CategoryID
	Categories.CategoryName

Tabla 18.- Columnas elegidas en la cláusula SELECT y WHERE.

En esta fase de la traducción es en donde se implementó la **Regla 3** que se encuentra en la clase “**Grafo.java**”, y que realiza las siguientes acciones.

Se obtiene el número de campos elegidos al analizar la cláusula “**WHERE**”, que como se explicó en este ejemplo son 3.

Se realiza una consulta a la tabla “**ResultadosCampos**” para obtener los campos encontrados y se guardan los resultados en un arreglo, en este ejemplo al analizar la cláusula “**SELECT**” solo se encontró un campo.

Se compara el número de campos encontrados al analizar la cláusula “**WHERE**” (que son 3) y el número de campos encontrados al analizar la cláusula “**SELECT**” (que solo es 1), y si es mayor el número de campos encontrados en la cláusula “WHERE” que en la cláusula “SELECT” se cumple la **regla 3**, como en este ejemplo y por lo tanto se muestra un diálogo al usuario.

Para mostrar el diálogo se utiliza la clase “**Dialogo.java**”, que obtiene la consulta original mostrándola al usuario en un cuadro de diálogo y muestra los campos encontrados por la interfaz para permitir al usuario elegir cuál campo o campos desea ver en la consulta final.

El mensaje que se mostrará en el cuadro de diálogo se establece de la siguiente manera:

Se establece en la interfaz que se está evaluando la regla 3, que modifica la cláusula "**SELECT**" de la consulta final.

Por lo tanto el mensaje que muestra al usuario es el siguiente: ***"¿Qué es lo que desea obtener en esta consulta?***" y se muestran a continuación los campos encontrados hasta el momento por la interfaz, permitiendo mediante casillas de verificación elegir los campos que desea obtener.

Por cada campo encontrado por la interfaz se debe obtener su descripción en español, esta información se encuentra en la BD "**Metadatos**" en la tabla del mismo nombre. Por lo tanto se hace una consulta a esta tabla, se extraen las descripciones y se agregan a cada campo.

Ya que en la consulta original si existe al menos una palabra que haga referencia a un campo de la base de datos que se consulta, **no se cumple la Regla 5**.

La consulta actual, no fue identificada por la interfaz como una consulta que utilice las funciones de agregación "**Max**", "**Min**" o "**Count**" [Ver sección 4.4] y por lo tanto **no se cumple la regla 4**.

Como se muestra en este ejemplo se cumple con la regla 3 y el diálogo mostrado al usuario quedaría de la siguiente manera.

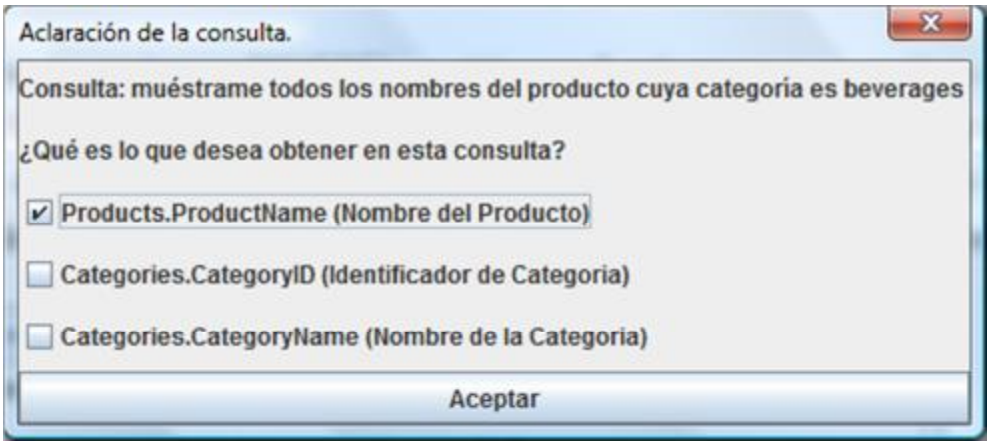


Figura 27.- Diálogo de clausula SELECT del ejemplo de la Regla 2.

En esta consulta se desea obtener los nombres del producto, por lo tanto el usuario seleccionará la columna correspondiente que es "**Products.ProductName**", y la consulta final dada por la interfaz considerando las relaciones de las tablas quedaría de la siguiente forma.

```
SELECT Products.ProductName
FROM Categories, Products
WHERE ((Categories.CategoryName LIKE 'beverages')) AND
Products.CategoryID = Categories.CategoryID
```

De esta forma finaliza el análisis de la consulta.

5.4.- Implementación de la Regla 4.

La cuarta regla se aplica cuando el número de campos resultantes final del análisis de la cláusula “SELECT” y la cláusula “WHERE” son mayores que 1 y se identificó que en la consulta es necesario utilizar alguna función de agregación, ya sea “COUNT”, “MAX” o “MIN”.

Consulta: ***dame la cantidad de autores de la ciudad de Berkeley***

Al realizar el análisis de la consulta [**Ver** secciones 4.1.1 – 4.1.6] en la cláusula “SELECT”, el tratamiento de preposiciones y las intersecciones, hasta este momento en la cláusula “SELECT”, se identificó un solo campo, resultado de las intersecciones de los “sustantivos campos” y los “sustantivos tabla” de la consulta, por lo tanto la cláusula “SELECT” quedaría de la siguiente forma:

```
SELECT Authors.au_id, Authors.au_lname, Authors.au_fname,
titleAuthor. au_id
FROM Authors, titleAuthor
```

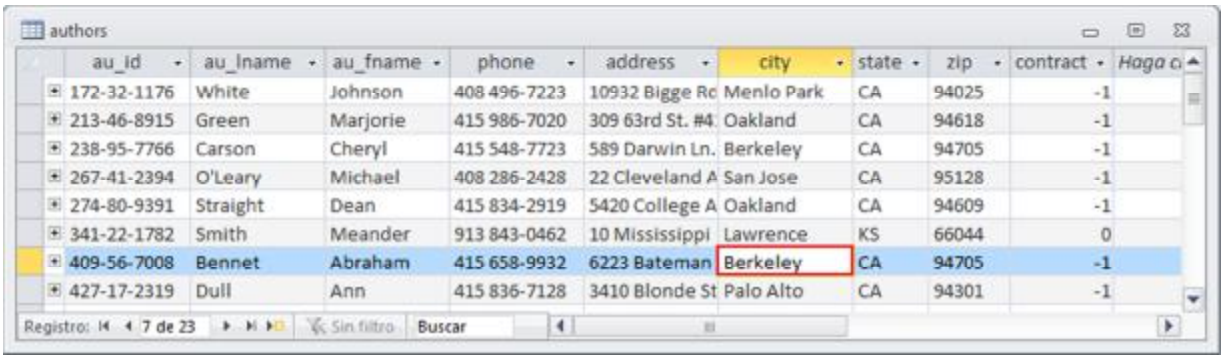
En este caso, en la consulta original solo se tiene un “sustantivo campo” y los campos resultantes al analizar la cláusula “SELECT” son 4, por lo tanto **si se cumple** con la **Regla 1**, del diálogo, sin embargo como esta regla es **excluyente** a la **regla 4 no se muestra** ningún **diálogo** al usuario.

Una vez que se analizó la cláusula “SELECT” se procede a analizar la cláusula “WHERE”, en donde se toman los “sustantivos campos”, los “sustantivos tabla” y el valor o los valores de la consulta en la cláusula “WHERE”.

Se realiza una consulta a la tabla “**ConsultaLN**” y se obtiene el número de registros que contengan en el campo “**categoria**” el valor “**a**, que significa que la palabra hace referencia a un valor dentro de la base de datos que se consulta. En este caso el valor encontrado en la consulta es “***Berkeley***”.

En el ejemplo actual en la cláusula “WHERE”, que es “***de la ciudad de Berkeley***” no existe un sustantivo que haga referencia a una tabla, se obtienen el sustantivo tabla de la cláusula “SELECT”, que en este caso es “**autores**” y hace referencia a la tabla “**Authors**”.

Se toma el valor “***Berkeley***” y se busca en todas las columnas de la tabla que hace referencia el sustantivo tabla, es decir en la tabla “**Authors**”.



authors

au_id	au_lname	au_fname	phone	address	city	state	zip	contract	Haga c
172-32-1176	White	Johnson	408 496-7223	10932 Bigge Rc	Menlo Park	CA	94025	-1	
213-46-8915	Green	Marjorie	415 986-7020	309 63rd St. #4	Oakland	CA	94618	-1	
238-95-7766	Carson	Cheryl	415 548-7723	589 Darwin Ln.	Berkeley	CA	94705	-1	
267-41-2394	O'Leary	Michael	408 286-2428	22 Cleveland A	San Jose	CA	95128	-1	
274-80-9391	Straight	Dean	415 834-2919	5420 College A	Oakland	CA	94609	-1	
341-22-1782	Smith	Meander	913 843-0462	10 Mississippi	Lawrence	KS	66044	0	
409-56-7008	Bennet	Abraham	415 658-9932	6223 Bateman	Berkeley	CA	94705	-1	
427-17-2319	Dull	Ann	415 836-7128	3410 Blonde St	Palo Alto	CA	94301	-1	

Registro: 7 de 23 Sin filtro Buscar

Figura 28.- Tabla "authors" de la base de datos Pubs.

Como se puede ver la única columna en donde se encuentra el valor “***Berkeley***” es en “**city**” por lo tanto se construye la condición “***authors.city LIKE '%Berkeley%'***”.

Se compara el número de campos encontrados al analizar la cláusula “**WHERE**” (que solo es 1) y el número de **valores** encontrados en la consulta ingresada por el usuario que también es 1, por lo tanto en este ejemplo **no cumple la regla 2,** que **no es excluyente** con la **regla 4.**

Una vez analizada la cláusula “**WHERE**” y como existen varias tablas en “ResultadoTabla” se obtienen estas tablas y también se obtienen las tablas de “Resultadocampos”, se comparan y se obtienen las tablas que si están en “ResultadoTabla” y que no se encuentran en “ResultadoCampos”, que en este caso como se puede observar en las tabla resultado no existe ninguna.

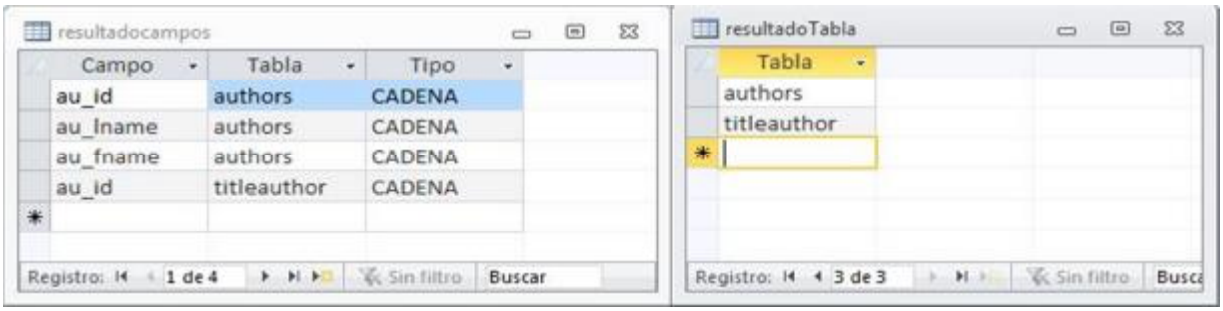


Figura 29.- Tablas resultado del ejemplo de la regla 4.

Como en esta comparación no se encontró ninguna tabla, no se añaden campos a la tabla “resultadoscampos”, y ademas como la **regla 3 es excluyente** con la **regla 4**, no se muestra ningún diálogo con el usuario.

En esta fase de la traducción es en donde se implementó la **Regla 4** que se encuentra en la clase “**Grafo.java**”, y que realiza las siguientes acciones.

En esta consulta se analiza la cláusula “SELECT” que corresponde a la frase “dame la cantidad de autores” y se identifica el uso de la función de agregación por medio de la palabra “Cantidad”, ya que se busca esta palabra en la tabla “**FuncAgregacion**” de la base de datos “**Lexicon**”, la tabla se muestra a continuación.

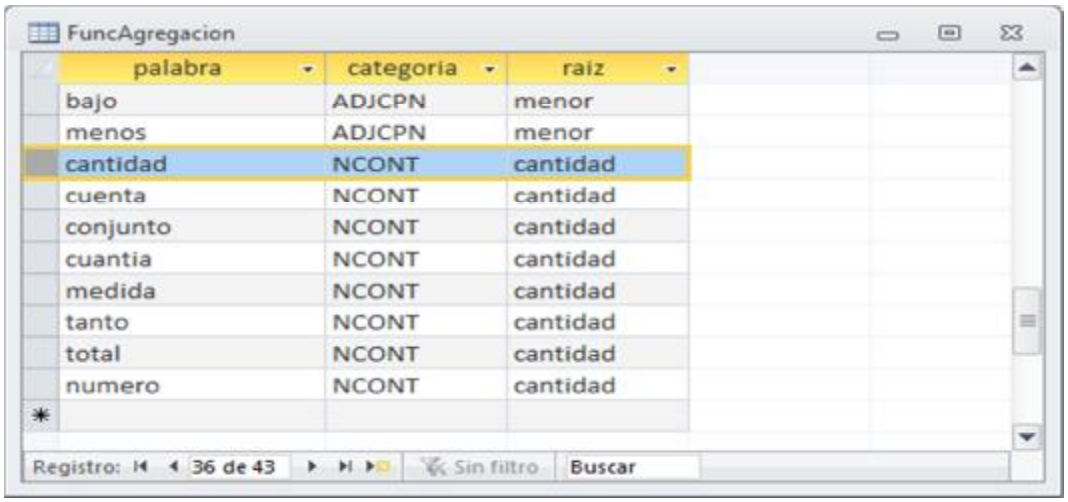


Figura 30.- Tabla "FuncAgregacion" de la base de datos Lexicon.

En este caso la palabra “**cantidad**” tiene la categoría “**NCONT**”, que indica que la consulta actual utiliza la función “**COUNT**” para resolverse correctamente.

Las demás categorías serían las siguientes:

- **ADJCPM**: Cuando hace referencia al máximo (Función “**max**”).
- **ADJCPN**: Cuando hace referencia al mínimo (Función “**min**”).
- **NCONT**: Cuando hace referencia a la cantidad (Función “**count**”).

Se realiza una consulta a la tabla “**ResultadosCampos**” para obtener los campos encontrados al analizar la cláusula “SELECT” y la cláusula “WHERE” y estos campos se guardan en un arreglo.

Las funciones de agregación “**COUNT**”, “**MAX**” y “**MIN**” solo pueden aplicarse a un solo campo, por lo tanto si el número de campos encontrados por la interfaz al analizar la consulta es mayor a 1, se cumple con la regla 4, como en este ejemplo y por lo tanto se muestra un diálogo al usuario.

Para mostrar el diálogo se utiliza la clase “**Dialogo.java**”, que obtiene la consulta original mostrándola al usuario en un cuadro de diálogo, también muestra los campos encontrados por la interfaz para permitir al usuario elegir cuál campo o campos desea ver en la consulta final y devuelve la opción elegida del usuario a la interfaz para modificar dicha consulta.

El mensaje que se mostrará en el cuadro de diálogo se establece de la siguiente manera:

Se establece en la interfaz que se está evaluando la regla 4, indica la utilización de una de las funciones de agregación “COUNT”, “MAX” o “MIN”.

El mensaje que se muestra al usuario depende de la función de agregación que se esté utilizando, los mensajes son los siguientes.

- Función MAX: ***¿De qué columna desea obtener el mayor?***
- Función MIN: ***¿De qué columna desea obtener el menor?***
- Función COUNT: ***¿De qué columna deseas obtener la cantidad de registros?***

Por lo tanto como se utiliza la función “COUNT” el mensaje que muestra al usuario es el siguiente: ***“¿De qué columna deseas obtener la cantidad de registros??***” y se muestran a continuación los campos encontrados hasta el momento por la interfaz, permitiendo mediante casillas de verificación elegir los campos que desea obtener.

Por cada campo encontrado por la interfaz se debe obtener su descripción en español, esta información se encuentra en la BD “**Metadatos**” en la tabla del mismo nombre. Por lo tanto se hace una consulta a esta tabla, se extraen las descripciones y se agregan a cada campo.

Como se muestra en este ejemplo se cumple con la regla 4 y el diálogo mostrado al usuario quedaría de la siguiente manera.

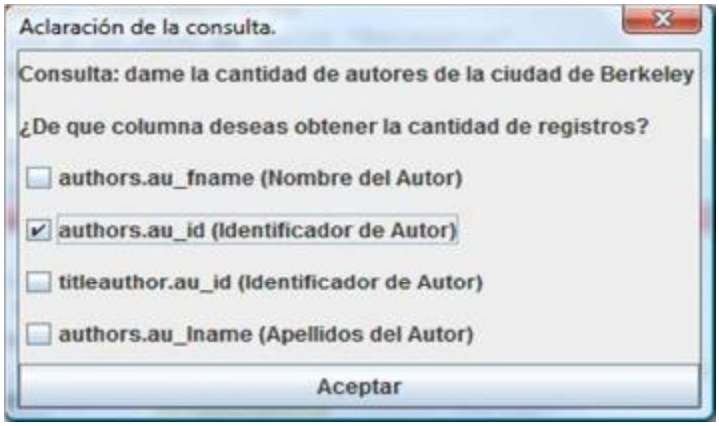


Figura 31.- Diálogo del ejemplo de la Regla 4.

En este caso se pregunta al usuario cuál es la columna que desea contabilizar con la función “**COUNT**”, en este ejemplo en particular se selecciona el campo “**Authors.au_id**”.

Ya que en la consulta original si existe al menos una palabra que haga referencia a un campo de la base de datos que se consulta, **no se cumple la Regla 5**. Además que la **regla 4 es excluyente** con la **regla 5**.

Por lo tanto después de realizar el diálogo con el usuario y el análisis de la cláusula "WHERE" la consulta final quedaría de la siguiente forma:

```
SELECT COUNT(authors.au_id)
FROM authors
WHERE ((authors.city LIKE '%Berkeley%'))
```

Cuando se utiliza una función de agregación solo debe asignarse a dicha función una sola columna, mediante el diálogo se evita el conflicto de tener varias columnas resultado y permite al usuario elegir cuál es la mejor opción para construir la consulta deseada.

5.5.- Implementación de la Regla 5.

La quinta regla se aplica cuando en la cláusula “SELECT” no existe ningún sustantivo que haga referencia al campo de una tabla, en este caso si el número de campos resultantes final del análisis de la cláusula “SELECT” y la cláusula “WHERE” es mayor que 1, se mostrará un diálogo al usuario para mejorar la calidad de la consulta.

Consulta: ***dame la categoría de la pasta***

“SELECT”, el tratamiento de preposiciones y las intersecciones, hasta este momento en la cláusula “SELECT”, se identificó un solo campo, resultado de las intersecciones de los “sustantivos campos” y los “sustantivos tabla” de la consulta, por lo tanto la cláusula “SELECT” quedaría de la siguiente forma:

```
SELECT Categories.CategoryID, Categories.CategoryName
FROM Categories
```

En este caso, en la consulta original no se tiene ningún “sustantivo campo” y los campos resultantes al analizar la cláusula “SELECT” son 2, pero como no se tiene ningún “sustantivo campo” en la consulta no se evalúa la **regla 1**, además esta regla es **excluyente** a la **regla 4** por lo tanto **no se muestra** ningún **diálogo** al usuario.

Una vez que se analizó la cláusula “SELECT” se procede a analizar la cláusula “WHERE”, en donde se toman los “sustantivos campos”, los “sustantivos tabla” y el valor o los valores de la consulta en la cláusula “WHERE”.

Se realiza una consulta a la tabla “**ConsultaLN**” y se obtiene el número de registros que contengan en el campo “**categoria**” el valor “**a**, que significa que la palabra hace referencia a un valor dentro de la base de datos que se consulta. En este caso el valor encontrado en la consulta es “***pasta***”.

En el ejemplo actual en la cláusula “WHERE”, que es “***de la pasta***” no existe un sustantivo que haga referencia a una tabla, se obtienen el sustantivo tabla de la

cláusula “SELECT”, que en este caso es “***categoría***” y hace referencia a la tabla “**categories**”.

Se toma el valor “***pasta***” y se busca en todas las columnas de la tabla que hace referencia el sustantivo tabla, es decir en la tabla “**categories**”.

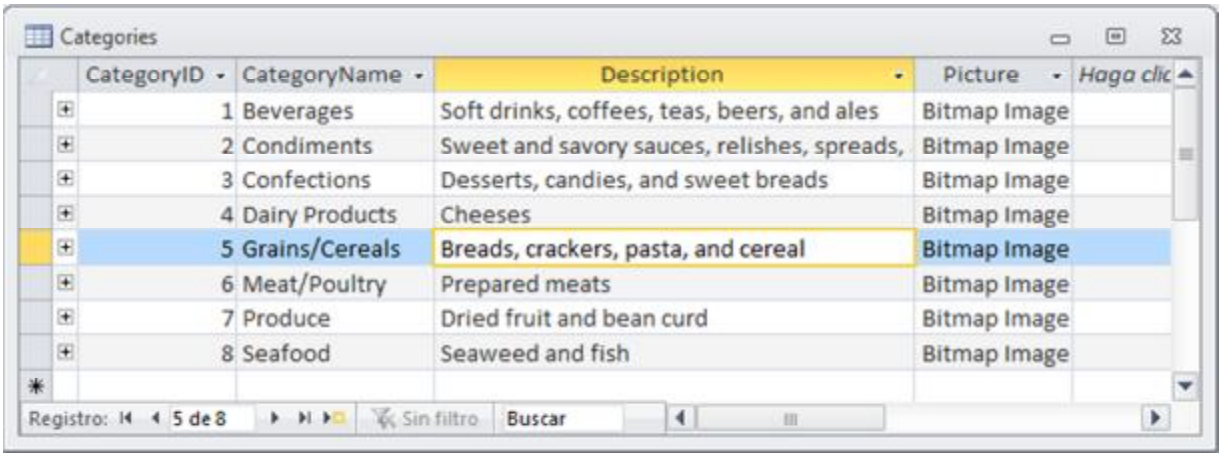


Figura 32. Tabla Categories de la base de datos Northwind.

Como se puede ver la única columna en donde se encuentra el valor “***pasta***” es en “**Description**” por lo tanto se construye la condición “***Categories.Description LIKE '%pasta%'***”.

Se compara el número de campos encontrados al analizar la cláusula “**WHERE**” (que solo es 1) y el número de **valores** encontrados en la consulta ingresada por el usuario que también es 1, por lo tanto en este ejemplo **no cumple la regla 2,** que **no es excluyente** con la **regla 5.**

Una vez analizada la cláusula “**WHERE**” y como en la tabla en la que se encontró el valor “**pasta**” de la consulta de ejemplo, es la misma que la encontrada en el análisis de la cláusula “SELECT” y por lo tanto no se agregan campos a la tabla “ResultadosCampo” y **no se cumple la Regla 3**, además que es excluyente con la regla 5.

Las tablas resultado quedaría de la siguiente manera.

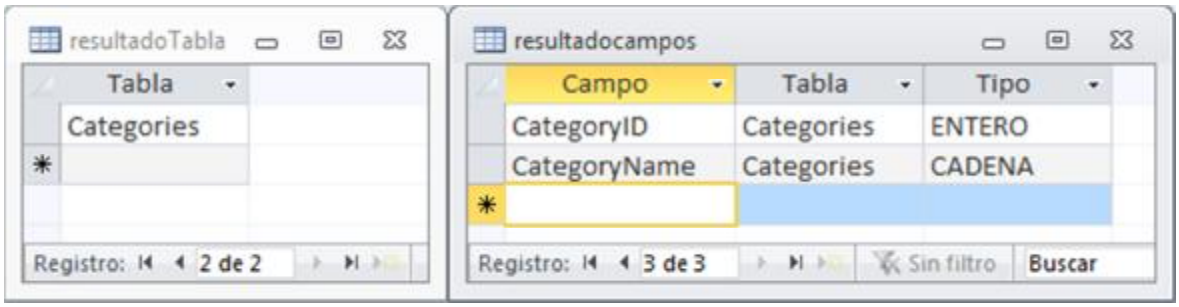


Figura 33.- Tablas resultado del ejemplo de la regla 4.

En esta fase de la traducción es en donde se implementó la **Regla 5** que se encuentra en la clase "**Grafo.java**", y que realiza las siguientes acciones.

Se realiza una consulta a la tabla "**ResultadosCampos**" para obtener los campos encontrados y se guardan los resultados en un arreglo, en este ejemplo al analizar la consulta se encontraron 2 campos.

Como en la consulta original no existe ningún "sustantivo campo" y los campos encontrados son mayores a 1, se cumple la **regla 5**, como en este ejemplo y por lo tanto se muestra un diálogo al usuario.

Para mostrar el diálogo se utiliza la clase "**Dialogo.java**", que obtiene la consulta original mostrándola al usuario en un cuadro de diálogo y muestra los campos encontrados por la interfaz para permitir al usuario elegir cuál campo o campos desea ver en la consulta final.

El mensaje que se mostrará en el cuadro de diálogo se establece de la siguiente manera:

Se establece en la interfaz que se está evaluando la regla 5, que modifica la cláusula "**SELECT**" de la consulta final.

Por lo tanto el mensaje que muestra al usuario es el siguiente: ***"¿Qué es lo que desea obtener en esta consulta?***" y se muestran a continuación los campos

encontrados hasta el momento por la interfaz, permitiendo mediante casillas de verificación elegir los campos que desea obtener.

Por cada campo encontrado por la interfaz se debe obtener su descripción en español, esta información se encuentra en la BD "**Metadatos**" en la tabla del mismo nombre. Por lo tanto se hace una consulta a esta tabla, se extraen las descripciones y se agregan a cada campo.

La consulta actual, no fue identificada por la interfaz como una consulta que utilice las funciones de agregación "**Max**", "**Min**" o "**Count**" [Ver sección 4.4] y por lo tanto **no se cumple la regla 4**.

Como se muestra en este ejemplo se cumple con la regla 5 y el diálogo mostrado al usuario quedaría de la siguiente manera.

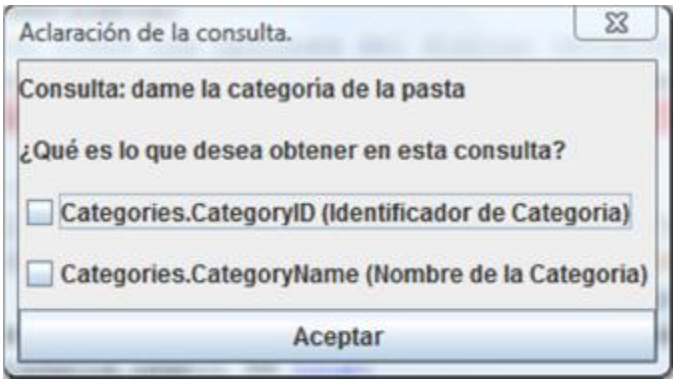


Figura 34.- Diálogo de clausula SELECT del ejemplo de la Regla 5.

En esta consulta se desea obtener los nombres de la categoría, por lo tanto el usuario seleccionará la columna correspondiente que es "Categories.CategoryName" y la consulta final dada por la interfaz quedaría de la siguiente forma.

```
SELECT Categories.CategoryName
FROM Categories
WHERE ((Categories.Description LIKE '%pasta%'))
```

De esta forma se logra mejorar la calidad de la consulta, eliminando las columnas innecesarias en el resultado final.

Capítulo 6

Resultados y conclusiones.

Para probar la interfaz se utilizaron los corpus utilizados en la tesis de [González A., 2011], que se compone de 50 consultas de la base de datos de Northwind [*Ver anexo C*] y 50 consultas de la base de datos de Pubs [*Ver anexo E*].

Cada consulta se compone de una forma imperativa y una forma interrogativa, que representan dos formas distintas de realizar la misma consulta.

Por ejemplo:

- **Consulta imperativa:** *dame el apellido de los empleados con fecha de contratación mayor a 01/01/1992.*
- **Consulta interrogativa:** *¿Cuál es el apellido de los empleados con fecha de contratación mayor a 01/01/1992?*

El análisis de una consulta en forma imperativa y una forma interrogativa es la misma, solo cambia el procesamiento inicial de dicha consulta.

Es decir que en una consulta interrogativa se eliminan símbolos especiales como los signos de interrogación “¿” y “?”, pero el análisis de la consulta es el mismo, por lo tanto los resultados devueltos por la interfaz para una consulta imperativa y una consulta interrogativa son los mismos.

A continuación se muestran los resultados en una tabla y una gráfica, donde se comparan con los resultados obtenidos en la experimentación con los resultados obtenidos en la tesis de [González A., 2011].

6.1 Resultado comparativos de la base de datos de Northwind.

En la siguiente tabla comparativa, se muestras los resultados de las experimentaciones realizadas por Arodith [González A., 2011] comparados con los resultados obtenidos en la experimentación de la interfaz desarrollada en esta tesis para la base de datos Northwind.

La primer columna "**Total**" indica el número total de consultas en este caso son 50, la segunda columna "**Correctas**" indica el número de consultas contestadas correctamente, la tercer columna "**Incorrectas**" indica el número de consultas contestadas incorrectamente, la quinta columna "**correctas sin inf. de más**" indica cuantas de las consultas contestadas correctamente no tienen información de más, y la sexta y última columna "**Correctas con inf. de más**" indica cuantas de las consultas contestadas correctamente si tienen información de más.

	Total de consultas		Correctas		Incorrectas		Correctas sin inf. de más		Correctas con inf. de más	
[Este trabajo]	50	100%	47	94%	3	6%	42	84%	5	10%
[González A., 2011]	50	100%	46	92%	4	8%	26	52%	20	40%

Tabla 19.- Resultados de la base de datos de Northwind.

A continuación se muestra la gráfica correspondiente a estos resultados:

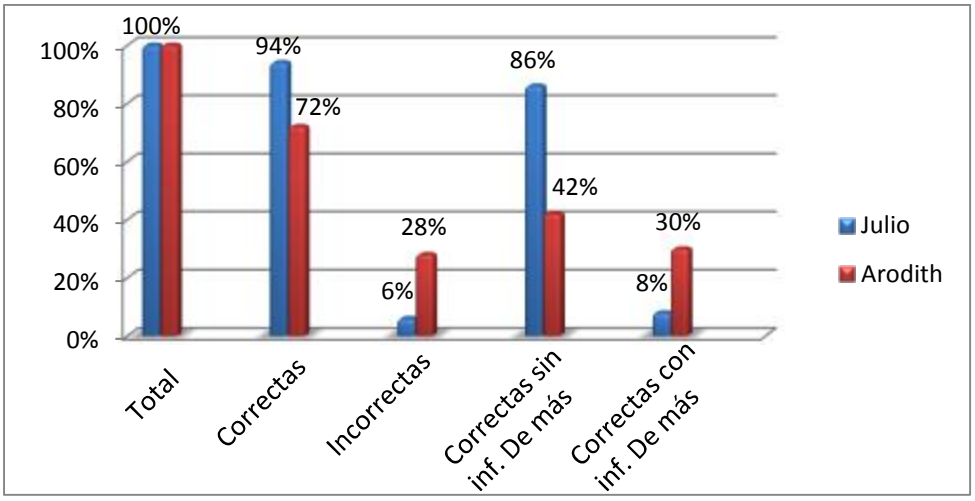


Figura 35.- Resultados gráficos de las pruebas de la BD Northwind.

En este corpus de la base de datos Northwind solo se logró un 2% de mejora en consultas contestadas correctamente, sin embargo se logró un incremento del 32% en consultas contestadas sin información de más.

El bajo incremento de consultas contestadas correctamente, se debe a que en el corpus solo existe una consulta que utiliza la función de agregación “COUNT”.

6.2 Resultados comparativos de la base de datos Pubs.

En la siguiente tabla comparativa, se muestras los resultados de las experimentaciones realizadas por Arodith [González A., 2011] comparados con los resultados obtenidos en la experimentación de la interfaz desarrollada en esta tesis para la base de datos **Pubs**.

La primer columna “**Total**” indica el número total de consultas en este caso son 50, la segunda columna “**Correctas**” indica el número de consultas contestadas correctamente, la tercer columna “**Incorrectas**” indica el número de consultas contestadas incorrectamente, la quinta columna “**Sin inf. de más**” indica cuantas de las consultas contestadas correctamente no tienen información de más, y la sexta y última columna “**Con inf. de más**” indica cuantas de las consultas contestadas correctamente si tienen información de más.

	Total de consultas		**Correctas**		**Incorrectas**		**Correctas sin inf. de más**		**Correctas con inf. de más**	
[Este trabajo]	50	100%	47	94%	3	6%	43	86%	4	8%
[González A., 2011]	50	100%	36	72%	14	28%	21	42%	15	30%

Tabla 20.- Resultados de la base de datos de Pubs.

A continuación se muestra la gráfica correspondiente a estos resultados:

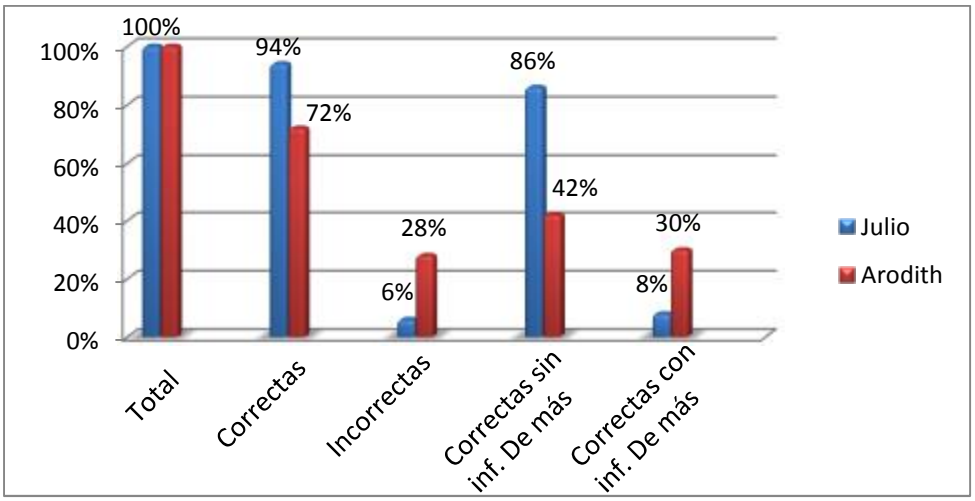


Figura 36.- Resultados gráficos de las pruebas de la BD Pubs.

En este corpus de la base de datos Pubs si se logró un incremento importante del 22% de mejora en consultas contestadas correctamente y un incremento del 44% en consultas contestadas sin información de más.

El alto incremento de consultas contestadas correctamente se debe a que en este corpus existe un gran número de consultas que utilizan funciones de agregación, en particular la función “COUNT” y el procesamiento de este tipo de consultas se desarrollo en este trabajo de tesis.

6.3 Conclusiones.

El uso del modelo de reglas para mostrar el diálogo obtuvo buenos resultados.

El diálogo con el usuario permite obtener mejores resultados y una mayor interacción de la interfaz con el usuario ya que cuando existe ambigüedad en la consulta introducida por el usuario es más factible preguntar al usuario que tratar de adivinar lo que desea en dicha consulta.

La posibilidad de resolver consultas que incluyen las funciones de agregación “count”, “max” y “min”, permite resolver un mayor número de consultas.

6.4 Trabajos futuros.

Como trabajos futuro se considera resolver consultas que incluyan la función “Group by”, ya que actualmente la interfaz no puede contestarlas y la inclusión de otras funciones de agregación como “SUM”, “AVG”, y otras.

Por ejemplo si se desea saber cuantos libros se vendieron en una fecha determinada se puede utilizar la consulta: “***dame el número de libros vendidos el 13/09/1994***” y la interfaz respondería correctamente. Sin embargo si se desea saber el número de libros vendidos en una fecha por cada editorial la consulta se puede formular de la siguiente forma: “***dame el numero de libros vendidos el 13/09/1994 por cada editorial***” pero la interfaz al no utilizar la función “**Group by**” no puede resolver esta consulta.

Por ejemplo si se desea obtener el monto total de las ventas de libros en determinada fecha la consulta se puede formular de la siguiente forma “***dame la suma del monto de los libros vendidos el 13/09/1994***” pero la interfaz al no utilizar la función “**Sum**” no puede resolver esta consulta.

Este tipo de consultas son muy comunes actualmente y por lo tanto es importante que en futuros trabajos se aborde el resolver consultas con las características antes mencionadas.

También se deben consideran el resolver consultas complejas que incluyan anidación de consultas.

Por ejemplo si se desean obtener los productos cuyo precio sea mayor al de todos los productos con categoría “Dairy Products”, primero se debe obtener el producto con mayor precio de la categoría “Dairy Products” y después obtener todos los productos que tengan un precio mayor al encontrado en dicha categoría. Para resolver esta consulta se necesitan consultas anidadas por lo tanto la interfaz actualmente no puede resolverlas.

Para realizar el análisis de la consulta se consideran sustantivos, preposiciones y conjunciones, pero se podrían considerar otro tipo de palabras como verbos, adjetivos, adverbios y otros elementos para lograr una mayor calidad en las consultas.

Por lo tanto es muy importante abordar estas problemáticas en trabajos futuros para aumentar el porcentaje de consultas contestadas correctamente.

Referencias.

[Allen et al, 2001] Allen, J., D. Byron, M. Dzikovska, G. Ferguson, L. Galescu, y A. Stent. 2001. Towards conversational human-computer interaction. En AI Magazine,2001, páginas 27-37.

[Allen y Perault, 1980] Allen, J.F. y C.R. Perault. 1980. Analyzing intentions in dialogues. En Artificial Intelligence, volumen 15(3), páginas 143-178.

[Androutsopoulos et al, 1995] Androutsopoulos I., Ritchie G.D., Thanisch P. Natural Language Interfaces to Databases - An Introduction. Natural Language Engineering, Vol 1:29-81, 1995.

[Appelt, 1985] Appelt, D.E. 1985. Planning English Sentences. En Cambridge, University Press.

[Bagnasco, Bresciani, et al, 1996] Clara Bagnasco, Paolo Bresciani, Bernardo Magnini, y Carlo Strapparava, ”Natural Language Interpretation for Public Administration Database Querying in the TAMIC Demonstrator”, In the Proceedings of the Second International Workshop on Applications of Natural Language to Information Systems, Amsterdam, Netherlands, 1996.

[Boldasov et al, 2003] Michael V. Boldasov, Elena G. Sokolova, y Michael G. Malkovsky, “User Query Understanding by the InBASE System as a Source for a Multilingual NL Generation Module”, Text, Speech and Dialogue, 5th International Conference, TSD 2002, pp.33-40, Sep. 2002.

[Cohen y Levesque, 1990] Cohen, P.R. y H.J. Levesque. 1990. Rational interaction as the basis for communication. En P. R. Cohen, J. Morgan, and M. E. Pollack, editors, Intentions in Communication. MIT Press.

[Giordani 2010] GIORDANI A. y MOSCHITTI A..Semantic Mapping Between Natural Language Questions and SQL Queries via Syntactic Pairing";. In: Proceedings of International Conference on Applications of Natural Language to Information Systems, 2009.

[González A., 2011] González, Arodit M. Estrategia de Mejora para la Independencia del Dominio en una Interfaz de Lenguaje Natural. Tesis Maestría. Instituto tecnológico de Cd. Madero, Tamaulipas, México, 2011.

[González J.J., 2005] Gonzalez, J.Javier. Traductor de Lenguaje Natural Español a SQL para un Sistema de Consultas a Bases de Datos. PhD dissertation, Centro Nacional de Investigación y Desarrollo Tecnológico, Cuernavaca, México, 2005.

[Hallett, 2006] Hallett, C. Proceedings of the Fourth International Natural Language Generation Conference. Association for Computational Linguistics. pp.95-102, 2006. (WYSIWYM)

[McTear, 2004] McTear, Michael F., 2004. Spoken Dialogue Technology: Towards the Conversational User Interface. Springer.

[Mendoza, 2002] Mendoza, Alejandro; “Desarrollo de un Módulo de Predicados Lógicos para Consultas a la Base de Datos de Alumnos del ITCM”. Tesis de licenciatura. Instituto Tecnológico de Ciudad Madero. 2002.

[Minock, 2008] MINOCK M., OLOFSSON P. y NÄSLUND A. Towards building robust natural language interfaces to databases. In: NLDB '08: Proceedings of the 13th international conference on Natural Language and Information Systems, Berlin, Heidelberg, 2008. (C-PHRASE).

[Popescu, Etzioni and Kautz, 2003] Ana M. Popescu , Oren Etzioni, Henry Kautz,"Towards a theory of natural language interfaces to databases," University of Washington, D.C. (PRECISE).

[Reiter and Dale, 2000] E. Reiter and R. Dale. Building Applied Natural Language Generation Systems. Cambridge University Press, Cambridge, 2000.

[Rodríguez, 2005] Rodríguez, Myriam; "Diseño de una Técnica para la Traducción de Consultas con Información Implícita a una Base de Datos". Tesis de Maestría. Instituto Tecnológico de Cd. Madero. 2005.

[Rojas J.C., 2009] Rojas J.C. 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.

[Searle, 1969] Searle, John. 1969. Speech acts: An essay in the philosophy of language. En Cambridge University Press, Cambridge (Reino Unido).

[Silberschatz et al, 2006] Silberschatz S., Korth, y Sudarshan, Fundamentos de Bases de Datos, 5ta edición, McGraw- Hill, 2006.

[Silva, 2005] Silva, Omar; "Implementación de un Traductor de Lenguaje Natural Independiente de Dominio". Tesis de Maestría. Instituto Tecnológico de Cd. Madero. 2005.

[Stratica, 2005] STRATICA N. Using semantic templates for a natural language interface to the CINDI virtual library . Data & Knowledge Engineering, Vol 55:4-19, Elsevier Science Publishers, The Netherlands, 2005. (NLPQC)

[Torres F., 2006] Torres, Francisco; "Sistemas de diálogo basados en modelos estocásticos". Tesis de Doctorado, Departamento de Sistemas Informáticos y Computación, Universidad Politécnica de Valencia, España, 2006.

Anexos.

Anexo A.- Esquema de la base de datos Northwind, que es una BD de ejemplo de Microsoft.

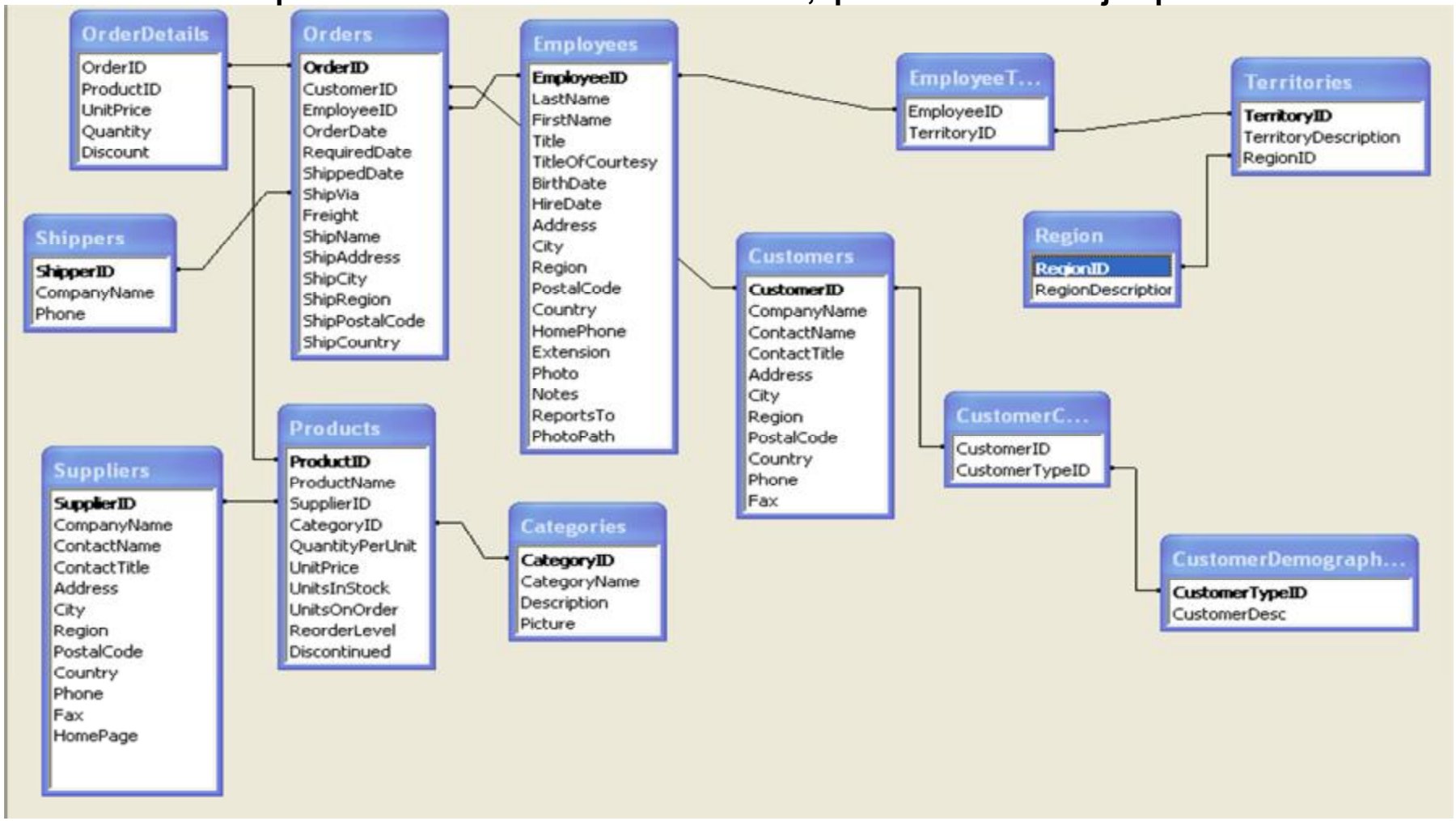


Figura 37.- Esquema de la base de datos Northwind.

Anexo B.- Esquema de la base de datos Pubs, que es una BD de ejemplo de Microsoft.

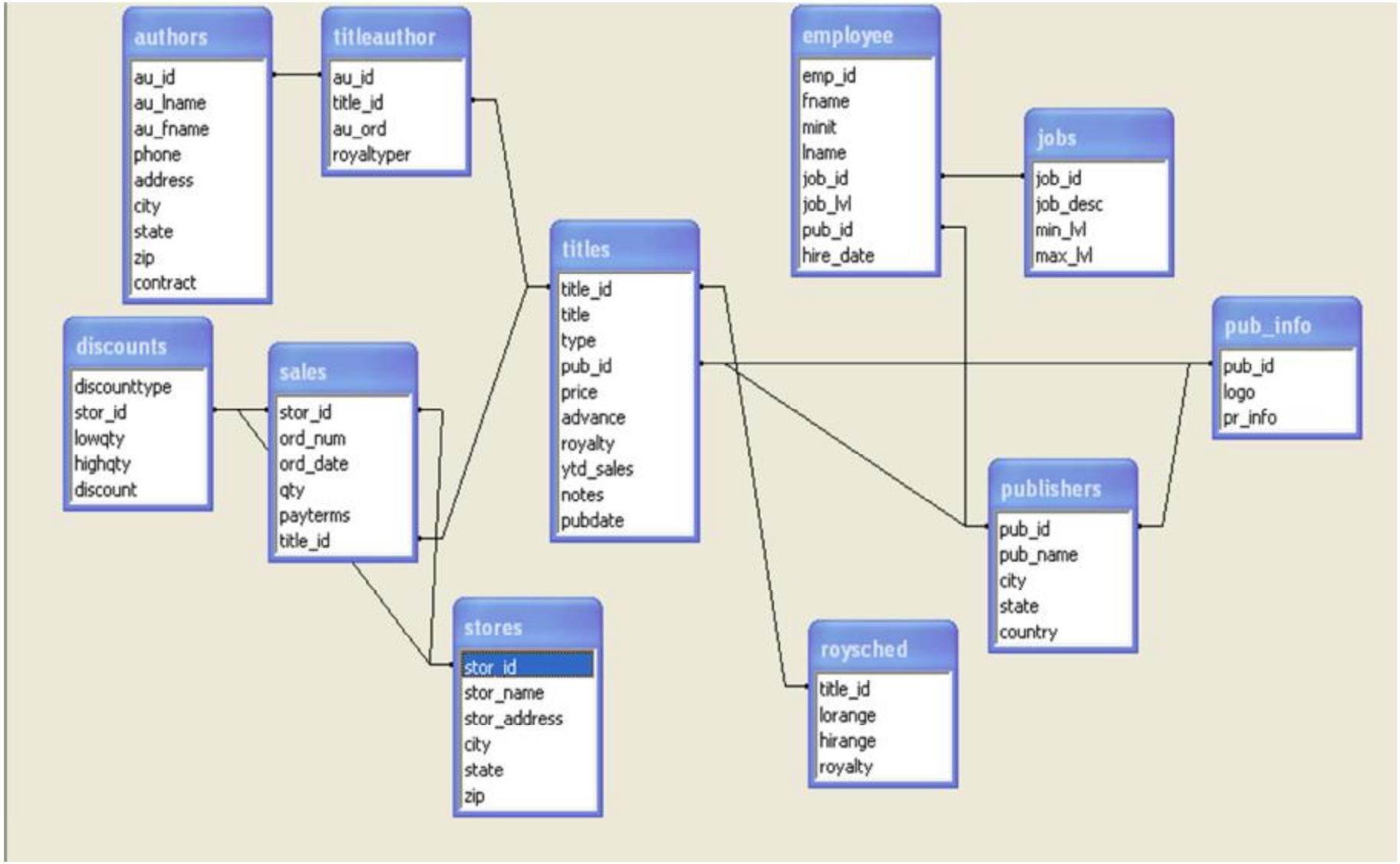


Figura 38.- Esquema de la base de datos Pubs.

Anexo C.- Corpus de la base de datos de Northwind.

No.	Consulta imperativa	Consulta interrogativa
1	dame el nombre de todas las categorías	¿Cuál es el nombre de todas las categorías?
2	dame el apellido de los empleados con fecha de contratación mayor a 01/01/1992	¿Cuál es el apellido de los empleados con fecha de contratación mayor a 01/01/1992?
3	dame el nombre de las compañías de clientes de la ciudad de México	¿Cuál es el nombre de las compañías de clientes de la ciudad de México?
4	dame el flete de la orden con fecha de envío 09/07/1996	¿Cuál es el flete de la orden con fecha de envío 09/07/1996?
5	dame el apellido del empleado con fecha de contratación 23/07/1996	¿Cuál es el apellido del empleado con fecha de contratación 23/07/1996?
6	dame el identificador de la orden cuyo monto total sea menor de 20	¿Cuál es el identificador de la orden cuyo monto total es menor de 20?
7	dame el identificador del empleado con descripción de territorio Philadelphia	¿Cuál es el identificador del empleado con descripción de territorio Philadelphia?
8	dame la descripción que tiene el nombre de la categoría Dairy Products	¿Qué descripción tiene el nombre de la categoría Dairy Products?
9	dame la descripción de la categoría que tiene el identificador de la categoría número 4	¿Cuál es la descripción de la categoría que tiene identificador de la categoría número 4?
10	dame las categorías de los productos	¿Cuáles son las categorías de los productos?
11	dame el nombre de la compañía de los clientes con identificador ANTON	¿Cuál es la nombre de la compañía de los clientes con identificador ANTON?
12	muéstrame todos los nombres del producto cuya categoría es beverages	¿Cuáles son los nombres de los productos cuya categoría es beverages?
13	dame los nombres de los productos de la categoría 2	¿Qué nombres de productos son de la categoría 2?
14	lista los nombres de los proveedores de Germany	Cuál es el nombre de los proveedores de Germany
15	dame los precios por unidad de los productos con identificador de producto 3 y 6	¿Cuál es el precio por unidad de los productos con identificador de producto 3 y 6?
16	dame la descripción de la categoría condiments	¿Cuál es la descripción de la categoría condiments?

17	dame el precio y la cantidad del producto con identificador de producto 42	¿Cuál es el precio y la cantidad del producto con identificador de producto 42?
18	dame el precio unitario de la orden con identificador 10249	¿Cuál es el precio unitario de la orden con identificador 10249?
19	dame la descripción de la categoría de Dairy	¿Qué descripción tiene la categoría Dairy Products?
20	dame el precio de la orden con identificador 10248	¿Cuál es el precio de la orden con identificador 10248?
21	dame los proveedores de la ciudad de Tokio	¿Quiénes son los proveedores de la ciudad de Tokio?
22	un listado de todos los clientes que sean Sales Representative	¿Qué clientes son Sales Representative?
23	dame una lista de los clientes que son de London	¿Qué clientes son de London?
24	dame la categoría de sweet	¿Cuál es la categoría sweet?
25	dame la categoría de la pasta	¿Cuál es la categoría de la pasta?
26	muéstrame todos los productos con la categoría "produce" e identificador de producto 39	¿Qué productos tienen nombre de categoría “produce” e identificador de producto 39?
27	Dame el número de productos con precio mayor a 10	¿Qué cantidad de productos tienen precio mayor a 10?
28	dame los nombres de la compañía de clientes	¿Cuáles son los nombres de las compañías de clientes?
29	muéstrame el teléfono del cliente cuyo nombre de compañía es United Package	¿Cuál es el teléfono del cliente cuyo nombre de compañía es United Package?
30	dame el nombre del navio que tiene como país de envió Brazil	Cuál es el nombre del navío que tiene como país de envío Brazil
31	dame el apellido del empleado que tiene el nombre Nancy	¿Cuál es el apellido del empleado que tiene el nombre Nancy?
32	dame el puesto del contacto del cliente que tiene la dirección Obere Str.57	¿Cuál es el puesto del contacto del cliente que tiene la dirección Obere Str.57?
33	Dame la ciudad donde se encuentra el proveedor exotic liquids	¿En qué ciudad se encuentra el proveedor exotic liquids?

34	dame el nombre del contacto del cliente que tiene la fecha de orden 30/07/1996	¿Cuál es el nombre del contacto del cliente que tiene la fecha de orden 30/07/1996?
35	dame el teléfono del fletador speedy express	¿Cuál es el teléfono del fletador speedy express?
36	dame los nombres de las compañías de clientes que están situadas en México D.F.	¿Cuáles son los nombres de las compañías de clientes que están situadas en México D.F.?
37	dame la fecha de contratación del empleado Margaret Peacock	¿Cuál es la fecha de contratación del empleado Margaret Peacock?
38	dame el identificador del fletador United Package	¿Cuál es el identificador del fletador United Package?
39	dame el codigo postal y la ciudad del proveedor Exotic Liquids	¿Cuál es el código postal y la ciudad del proveedor Exotic Liquids?
40	dame la dirección del cliente Antonio moreno	¿Cuál es la dirección del cliente Antonio moreno?
41	dame la fecha de nacimiento del empleado andrew fuller	¿Cuál es la fecha de nacimiento del empleado andrew fuller?
42	dame el número de teléfono del fletador federal shipping	¿Cuál es el número de teléfono del fletador federal shipping?
43	dame el título de cortesía de la empleada Nancy Davolio	¿Cuál es el título de cortesía de Nancy Davolio?
44	dame el nombre del proveedor con identificador 16	¿Cuál es el nombre del proveedor con identificador 16?
45	dame el puesto del contacto del cliente Comércio Mineiro	¿Cuál es el puesto del contacto del cliente Comércio Mineiro?
46	dame la descripción de la categoría de producto Pavlova	¿Cuál es la descripción de la categoría de producto Pavlova?
47	dame los teléfonos de las fletadoras united package y speedy express	¿Cuáles son los teléfonos de las fletadoras united package y speedy express?
48	dame las unidades de reserva de los productos de la categoría seafood	¿Cuántas unidades de reserva de los productos tiene la categoría seafood?
49	Muestra la dirección de Laura Callahan	¿Cuál es la dirección de Laura Callahan?
50	lista los clientes que compraron fletes mayores a 50	¿Qué clientes me compraron fletes mayores a 50?

Anexo D.- Resultados de la interfaz con el corpus de Northwind.

	Julio Orta	
No.	Imperativa	Interrogativa
1	SELECT Categories.CategoryName FROM Categories	SELECT Categories.CategoryName FROM Categories
2	SELECT Employees.FirstName, Employees.LastName FROM Employees WHERE ((Employees.HireDate > #1992/01/01#))	SELECT Employees.FirstName, Employees.LastName FROM Employees WHERE ((Employees.HireDate > #1992/01/01#))
3	SELECT Customers.CompanyName FROM Customers WHERE ((Customers.City LIKE '%Mexico%'))	SELECT Customers.CompanyName FROM Customers WHERE ((Customers.City LIKE '%Mexico%'))
4	SELECT Orders.Freight FROM Orders WHERE ((Orders.ShippedDate = #1996/07/09#))	SELECT Orders.Freight FROM Orders WHERE ((Orders.ShippedDate = #1996/07/09#))
5	SELECT Employees.FirstName, Employees.LastName FROM Employees WHERE ((Employees.HireDate = #1996/07/23#))	SELECT Employees.FirstName, Employees.LastName FROM Employees WHERE ((Employees.HireDate = #1996/07/23#))
6	SELECT Orders.OrderID FROM Orders WHERE ((Orders.Freight < 20))	SELECT Orders.OrderID FROM Orders WHERE ((Orders.Freight < 20))
7	SELECT EmployeeTerritories.EmployeeID FROM EmployeeTerritories WHERE ((EmployeeTerritories.TerritoryID LIKE '%Philadelphia%'))	SELECT EmployeeTerritories.EmployeeID FROM EmployeeTerritories WHERE ((EmployeeTerritories.TerritoryID LIKE '%Philadelphia%'))
8	SELECT Categories.Description FROM Categories WHERE ((Categories.CategoryName LIKE '%Dairy%') OR (Categories.CategoryName LIKE '%Products%'))	SELECT Categories.Description FROM Categories WHERE ((Categories.CategoryName LIKE '%Dairy%') OR (Categories.CategoryName LIKE '%Products%'))
9	SELECT Categories.Description FROM Categories WHERE ((Categories.CategoryID = 4))	SELECT Categories.Description FROM Categories WHERE ((Categories.CategoryID = 4))
10	SELECT Categories.CategoryName FROM Categories, Products WHERE Products.CategoryID = Categories.CategoryID	SELECT Categories.CategoryName FROM Categories, Products WHERE Products.CategoryID = Categories.CategoryID
11	SELECT Customers.CompanyName FROM Customers WHERE ((Customers.CustomerID LIKE '%ANTON%'))	SELECT Customers.CompanyName FROM Customers WHERE ((Customers.CustomerID LIKE '%ANTON%'))
12	SELECT Products.ProductName FROM Categories, Products WHERE ((Categories.CategoryName LIKE 'beverages')) AND Products.CategoryID = Categories.CategoryID	SELECT Products.ProductName FROM Categories, Products WHERE ((Categories.CategoryName LIKE 'beverages')) AND Products.CategoryID = Categories.CategoryID
13	SELECT Products.ProductName FROM Categories, Products WHERE ((Products.ProductName = '2')) AND Products.CategoryID = Categories.CategoryID	SELECT Products.ProductName FROM Categories, Products WHERE ((Products.ProductName = '2')) AND Products.CategoryID = Categories.CategoryID
14	SELECT Suppliers.ContactName FROM Suppliers WHERE ((Suppliers.Country LIKE 'Germany'))	SELECT Suppliers.ContactName FROM Suppliers WHERE ((Suppliers.Country LIKE 'Germany'))
15	SELECT Products.UnitPrice FROM Products WHERE ((Products.ProductID = 3) OR (Products.ProductID = 6))	SELECT Products.UnitPrice FROM Products WHERE ((Products.ProductID = 3) OR (Products.ProductID = 6))
16	SELECT Categories.Description FROM Categories WHERE ((Categories.CategoryName LIKE 'condiments'))	SELECT Categories.Description FROM Categories WHERE ((Categories.CategoryName LIKE 'condiments'))
17	SELECT Products.UnitPrice FROM Products WHERE ((Products.ProductID = 42))	SELECT Products.UnitPrice FROM Products WHERE ((Products.ProductID = 42))
18	SELECT OrderDetails.UnitPrice	SELECT OrderDetails.UnitPrice

	FROM OrderDetails WHERE ((OrderDetails.OrderID = 10249))	FROM OrderDetails WHERE ((OrderDetails.OrderID = 10249))
19	SELECT Categories.Description FROM Categories WHERE ((Categories.CategoryName LIKE '%Dairy%'))	SELECT Categories.Description FROM Categories WHERE ((Categories.CategoryName LIKE '%Dairy%') OR (Categories.CategoryName LIKE '%Products%'))
20	SELECT Orders.Freight FROM Orders WHERE ((Orders.OrderID = 10248))	SELECT Orders.Freight FROM Orders WHERE ((Orders.OrderID = 10248))
21	SELECT Suppliers.SupplierID FROM Suppliers WHERE ((Suppliers.City LIKE '%Tokio%'))	SELECT Suppliers.SupplierID FROM Suppliers WHERE ((Suppliers.City LIKE '%Tokio%'))
22	SELECT Customers.CustomerID FROM Customers WHERE ((Customers.ContactTitle LIKE '%Sales%') OR (Customers.ContactTitle LIKE '%Representative%')	SELECT Customers.CustomerID FROM Customers WHERE ((Customers.ContactTitle LIKE '%Sales%') OR (Customers.ContactTitle LIKE '%Representative%'))
23	SELECT Customers.CustomerID FROM Customers WHERE ((Customers.City LIKE 'London'))	SELECT Customers.CustomerID FROM Customers WHERE ((Customers.City LIKE 'London'))
24	SELECT Categories.CategoryName FROM Categories WHERE ((Categories.Description LIKE '%sweet%'))	SELECT Categories.CategoryName FROM Categories WHERE ((Categories.Description LIKE '%sweet%'))
25	SELECT Categories.CategoryName FROM Categories WHERE ((Categories.Description LIKE '%pasta%'))	SELECT Categories.CategoryName FROM Categories WHERE ((Categories.Description LIKE '%pasta%'))
26	SELECT Categories.CategoryName FROM Categories, Products WHERE ((Categories.CategoryName LIKE 'produce') AND (Products.ProductID = 39)) AND Products.CategoryID = Categories.CategoryID	SELECT Products.ProductName FROM Categories, Products WHERE ((Categories.CategoryName LIKE '%?produce?%') AND (Products.ProductID = 39)) AND Products.CategoryID = Categories.CategoryID
27	SELECT COUNT(Products.ProductName) FROM Products WHERE ((Products.UnitPrice > 10))	SELECT COUNT(Products.ProductName) FROM Products WHERE ((Products.UnitPrice > 10))
28	SELECT Customers.CompanyName FROM Customers	SELECT Customers.CompanyName FROM Customers
29	SELECT Customers.Phone FROM Customers WHERE ((Customers.CompanyName LIKE '%United%') OR (Customers.CompanyName LIKE '%Package%'))	SELECT Customers.Phone FROM Customers WHERE ((Customers.CompanyName LIKE '%United%') OR (Customers.CompanyName LIKE '%Package%'))
30	SELECT Orders.ShipName FROM Orders WHERE ((Orders.ShipCountry LIKE 'Brazil'))	SELECT Orders.ShipName FROM Orders WHERE ((Orders.ShipCountry LIKE 'Brazil'))
31	SELECT Employees.FirstName, Employees.LastName FROM Employees WHERE ((Employees.FirstName LIKE '%Nancy%'))	SELECT Employees.FirstName, Employees.LastName FROM Employees WHERE ((Employees.FirstName LIKE '%Nancy%'))
32	SELECT Customers.ContactTitle FROM Customers WHERE ((Customers.Address LIKE '%Obere%') OR (Customers.Address LIKE '%Str.57%'))	SELECT Customers.ContactTitle FROM Customers WHERE ((Customers.Address LIKE '%Obere%') OR (Customers.Address LIKE '%Str.57%'))
33	SELECT Suppliers.City FROM Suppliers WHERE ((Suppliers.CompanyName LIKE '%exotic%') OR (Suppliers.CompanyName LIKE '%liquids%'))	SELECT Suppliers.City FROM Suppliers WHERE ((Suppliers.CompanyName LIKE '%exotic%') OR (Suppliers.CompanyName LIKE '%liquids%'))
34	SELECT Customers.ContactName FROM Customers, Orders WHERE ((Orders.OrderDate = #1996/07/30#)) AND Customers.CustomerID = Orders.CustomerID	SELECT Customers.ContactName FROM Customers, Orders WHERE ((Orders.OrderDate = #1996/07/30#)) AND Customers.CustomerID = Orders.CustomerID
35	SELECT Shippers.Phone FROM Shippers WHERE ((Shippers.CompanyName LIKE '%speedy%') OR (Shippers.CompanyName LIKE '%express%'))	SELECT Shippers.Phone FROM Shippers WHERE ((Shippers.CompanyName LIKE '%speedy%') OR (Shippers.CompanyName LIKE '%express%'))

36	SELECT Customers.CompanyName FROM Customers WHERE ((Customers.Country LIKE 'Mexico') OR (Customers.City LIKE '%D.F.%'))	SELECT Customers.CompanyName FROM Customers WHERE ((Customers.Country LIKE 'Mexico') OR (Customers.City LIKE '%D.F.%'))
37	SELECT Employees.HireDate FROM Employees WHERE ((Employees.FirstName LIKE 'Margaret') OR (Employees.LastName LIKE 'Peacock'))	SELECT Employees.HireDate FROM Employees WHERE ((Employees.FirstName LIKE 'Margaret') OR (Employees.LastName LIKE 'Peacock'))
38	SELECT Shippers.ShipperID FROM Shippers WHERE ((Shippers.CompanyName LIKE '%United%') OR (Shippers.CompanyName LIKE '%Package%'))	SELECT Shippers.ShipperID FROM Shippers WHERE ((Shippers.CompanyName LIKE '%United%') OR (Shippers.CompanyName LIKE '%Package%'))
39	SELECT Suppliers.City, Suppliers.PostalCode FROM Suppliers WHERE ((Suppliers.CompanyName LIKE '%Exotic%') OR (Suppliers.CompanyName LIKE '%Liquids%'))	SELECT Suppliers.City, Suppliers.PostalCode FROM Suppliers WHERE ((Suppliers.CompanyName LIKE '%Exotic%') OR (Suppliers.CompanyName LIKE '%Liquids%'))
40	SELECT Customers.Address FROM Customers WHERE ((Customers.CompanyName LIKE '%Antonio%' OR Customers.ContactName LIKE '%Antonio%') OR (Customers.CompanyName LIKE '%moreno%' OR Customers.ContactName LIKE '%moreno%'))	SELECT Customers.Address FROM Customers WHERE ((Customers.CompanyName LIKE '%Antonio%' OR Customers.ContactName LIKE '%Antonio%') OR (Customers.CompanyName LIKE '%moreno%' OR Customers.ContactName LIKE '%moreno%'))
41	SELECT Employees.BirthDate FROM Employees WHERE ((Employees.FirstName LIKE 'andrew') OR (Employees.LastName LIKE 'fuller'))	SELECT Employees.BirthDate FROM Employees WHERE ((Employees.FirstName LIKE 'andrew') OR (Employees.LastName LIKE 'fuller'))
42	SELECT Shippers.Phone FROM Shippers WHERE ((Shippers.CompanyName LIKE '%shipping%'))	SELECT Shippers.Phone FROM Shippers WHERE ((Shippers.CompanyName LIKE '%shipping%'))
43	SELECT Employees.TitleOfCourtesy FROM Employees WHERE ((Employees.FirstName LIKE 'Nancy') OR (Employees.LastName LIKE 'Davolio'))	SELECT Employees.TitleOfCourtesy FROM Employees WHERE ((Employees.FirstName LIKE 'Nancy') OR (Employees.LastName LIKE 'Davolio'))
44	SELECT Suppliers.CompanyName, Suppliers.ContactName FROM Suppliers WHERE ((Suppliers.SupplierID = 16))	SELECT Suppliers.CompanyName, Suppliers.ContactName FROM Suppliers WHERE ((Suppliers.SupplierID = 16))
45	SELECT Customers.ContactTitle FROM Customers WHERE ((Customers.CompanyName LIKE 'comercio') OR (Customers.Fax LIKE 'Mineiro'))	SELECT Customers.ContactTitle FROM Customers WHERE ((Customers.CompanyName LIKE 'comercio') OR (Customers.Fax LIKE 'Mineiro'))
46	SELECT Categories.Description FROM Categories, Products WHERE ((Products.ProductName LIKE 'Pavlova')) AND Products.CategoryID = Categories.CategoryID	SELECT Categories.Description FROM Categories, Products WHERE ((Products.ProductName LIKE 'Pavlova')) AND Products.CategoryID = Categories.CategoryID
47	SELECT Shippers.Phone FROM Shippers WHERE ((Shippers.CompanyName LIKE '%united%') OR (Shippers.CompanyName LIKE '%package%') OR (Shippers.CompanyName LIKE '%speedy%') OR (Shippers.CompanyName LIKE '%express%'))	SELECT Shippers.Phone FROM Shippers WHERE ((Shippers.CompanyName LIKE '%united%') OR (Shippers.CompanyName LIKE '%package%') OR (Shippers.CompanyName LIKE '%speedy%') OR (Shippers.CompanyName LIKE '%express%'))
48	SELECT Products.UnitsInStock FROM Categories, Products WHERE ((Categories.CategoryName LIKE 'seafood')) AND Products.CategoryID = Categories.CategoryID	SELECT Products.UnitsInStock FROM Categories, Products WHERE ((Categories.CategoryName LIKE 'seafood')) AND Products.CategoryID = Categories.CategoryID
49	SELECT Employees.Address FROM Employees WHERE ((Employees.FirstName LIKE 'Laura') OR (Employees.LastName LIKE 'Callahan'))	SELECT Employees.Address FROM Employees WHERE ((Employees.FirstName LIKE 'Laura') OR (Employees.LastName LIKE 'Callahan'))
50	SELECT Customers.CustomerID FROM Customers WHERE ((Customers.Fax > '50'))	SELECT Customers.CustomerID FROM Customers WHERE ((Customers.Fax > '50'))

Anexo E.- Corpus de la base de datos Pubs.

No.	Consulta imperativa	Consulta interrogativa
1	dame los títulos de los libros	¿Cuál es el título de todos los libros?
2	visualiza los tipos de descuentos	¿Cuál son los tipos de descuentos?
3	dame la dirección y la ciudad de las editoriales	¿Cuál es la dirección y la ciudad de las editoriales?
4	dame el título del libro con identificador TC4203	¿Cuál es el título del libro con identificador TC4203?
5	selecciona el título donde el precio sea igual a $19.99 y el tipo sea business	¿Cuál es el título donde el precio sea igual a $19.99 y el tipo sea business?
6	lista los empleados con su respectivo cargo	¿Cuáles son los empleados y su respectivo rango?
7	dame el puesto que ocupa cada empleado	¿Cuál es el puesto que ocupa cada empleado?
8	Dame el almacén al que pertenece la siguiente dirección 679 carson st.	¿Cuál es el almacén al que pertenece la siguiente dirección 679 carson st.?
9	muéstrame el empleado con identificador H-B39728F	¿Cuál es el empleado con identificador H-B39728F?
10	lista los autores que viven en la ciudad de Oakland	¿Cuáles son los autores que viven en la ciudad de Oakland?
11	dame el empleado que tiene como nivel de trabajo 227	¿Cuál es el empleado que tiene como nivel de trabajo 227?
12	dame el empleado que tiene como fecha de contratación como 13/02/1991	¿Cuál es el empleado que tiene como fecha de contratación 13/02/1991?
13	selecciona todos los libros en donde su adelanto sea mayor a 5000	¿Cuáles son todos los libros en donde cuyo adelanto sea mayor a 5000?
14	mostrar los libros cuyo precio es mayor a $19.99 y son de tipo business	¿Cuáles son los libros cuyo precio es mayor a $19.99 y son de tipo business?
15	muestra la editorial que se encuentra en Germany	¿Cuál es la editorial que se encuentra en Germany?

16	dame el autor del libro the busy	¿Quién es el autor del libro the busy?
17	selecciona todos los libros del autor Smith	¿Cuáles son los libros del autor Smith?
18	dame los libros del autor green	¿Cuáles son los libros del autor green?
19	dame el puesto que tiene el empleado Francisco Chang	¿Cuál es el puesto que tiene el empleado Francisco Chang?
20	dame el puesto y fecha de contratación que tiene el empleado Paolo	¿Cuál es el puesto y fecha de contratación que tiene el empleado Paolo?
21	dame la dirección del almacén Barnum´s	¿Cuál es la dirección del almacén Barnum´s?
22	dame el número de teléfono del autor cheryl	¿Cuál es el número de teléfono del autor cheryl?
23	dame la ciudad de la editorial New Moon Books	¿Cuál es la ciudad de la editorial New Moon Books?
24	dame el nivel del empleado de philip cramer	¿Cuál es el nivel del empleado de philip cramer?
25	dame el precio del libro con identificador de editorial 1389	¿Cuál es el precio del libro con identificador de editorial 1389?
26	lista los libros de la editorial GGG&G	¿Cuáles son los libros de la editorial GGG&G?
27	dame el identificador y el precio del libro you can	¿Cuál es el identificador y el precio del libro you can?
28	dame la dirección y el teléfono del autor del libro you can	¿Cuál es la dirección y el teléfono del autor del libro you can?
29	dame la ciudad en donde se encuentra el autor Johnson White	¿Cuál es la ciudad en donde se encuentra el autor Johnson White?
30	dame el apellido que tiene el empleado Pedro	¿Cuál es el apellido que tiene el empleado Pedro?
31	dame la ciudad en donde se encuentra el almacen Bookbeat	¿Cuál es la ciudad en donde se encuentra el almacen Bookbeat?
32	dame el estado en donde se encuentra la editorial Ramona publishers	¿Cuál es el estado en donde se encuentra la editorial Ramona publishers?
33	título de los libros cuyos editores se encuentran en TX	¿Cuáles son los título de los libros cuyos editores se encuentran en TX?
34	dame el nombre y dirección del empleado que trabaja para la editorial GGG&G	¿Cuál es el nombre y la dirección del empleado que trabaja para la editorial

		GGG&G?
35	dame la descripción del puesto del empleado VPA30890F	¿Cuál es la descripción del puesto del empleado VPA30890F
36	obtener el nombre del almacén donde se encuentra el libro cooking with	¿Cuál es el nombre del almacén donde se encuentra el libro cooking with?
37	dame la fecha en que se realizó el contrato del empleado PTC11962M	¿Cuál es la fecha en que se realizó el contrato del empleado PTC11962M?
38	dame la cantidad de autores de la ciudad de Berkeley	¿Cuál es la cantidad de autores de la ciudad de Berkeley?
39	dame el número de empleados de la editorial Scotney book	¿Cuál es el número de empleados de la editorial Scotney book?
40	dame el número de ventas realizadas el 14/09/1994	¿Cuál es el número de ventas realizadas el 14/09/1994?
41	dame el número de libros vendidos el 13/09/1994	¿Cuál es el número de libros vendidos el 13/09/1994?
42	lista los identificadores que tienen los libros	¿Cuáles son los identificadores que tienen los libros?
43	dame la ciudad a la que pertenece el código postal 89076	¿Cuál es la ciudad a la que pertenece el código postal 89076?
44	dame la fecha de contratación de pedro	¿Cuál es la fecha de contratación de pedro?
45	dame los libros que contiene cada editorial	¿Cuáles son los libros que contiene cada editorial?
46	muestra la cantidad de ventas de sillicon valley	¿Cuál es la cantidad de ventas de sillicon valley?
47	dame la editorial en donde trabaja victoria ashworth	¿Cuál es la editorial en donde trabaja victoria ashworth?
48	nombre del almacén donde se encuentra the busy	¿Cuál es nombre del almacén donde se encuentra the busy?
49	obtén los números de ejemplares que tiene el libro the busy	¿Cuál es el números de ejemplares que tiene el libro the busy?
50	dame la dirección del almacén Barnum's	¿Cuál es la dirección del almacén Barnum's?

Anexo F.- Resultados de la interfaz con el corpus de Pubs.

	Julio Orta	
No.	Imperativa	Interrogativa
1	SELECT titles.title FROM titles	SELECT titles.title FROM titles
2	SELECT discounts.discounttype FROM discounts	SELECT discounts.discounttype FROM discounts
3	SELECT publishers.city FROM publishers	SELECT publishers.city FROM publishers
4	SELECT titles.title FROM titles WHERE ((titles.title_id LIKE '%TC4203%'))	SELECT titles.title FROM titles WHERE ((titles.title_id LIKE '%TC4203%'))
5	SELECT titles.title FROM titles WHERE ((titles.price = 19.99) AND (titles.type LIKE '%business%'))	SELECT titles.title FROM titles WHERE ((titles.price = 19.99) AND (titles.type LIKE '%business%'))
6	SELECT employee.emp_id, employee.fname, employee.lname FROM employee, jobs WHERE employee.job_id = jobs.job_id	SELECT employee.emp_id, employee.fname, employee.lname FROM employee, jobs WHERE employee.job_id = jobs.job_id
7	SELECT employee.emp_id, employee.fname, jobs.job_desc, employee.job_id, jobs.job_id, employee.lname FROM employee, jobs WHERE employee.job_id = jobs.job_id	SELECT employee.emp_id, employee.fname, jobs.job_desc, employee.job_id, jobs.job_id, employee.lname FROM employee, jobs WHERE employee.job_id = jobs.job_id
8	SELECT stores.stor_name FROM stores WHERE ((stores.stor_address = '679') OR (stores.stor_address LIKE '%carson%') OR (stores.stor_address LIKE '%st.%'))	SELECT stores.stor_name FROM stores WHERE ((stores.stor_address = '679') OR (stores.stor_address LIKE '%carson%') OR (stores.stor_address LIKE '%st.%'))
9	SELECT employee.fname FROM employee WHERE ((employee.emp_id LIKE '%H-B39728F%'))	SELECT employee.fname FROM employee WHERE ((employee.emp_id LIKE '%H-B39728F%'))
10	SELECT authors.au_fname, authors.au_id FROM authors WHERE ((authors.city LIKE '%Oakland%'))	SELECT authors.au_fname, authors.au_id FROM authors WHERE ((authors.city LIKE '%Oakland%'))
11	SELECT employee.emp_id, employee.fname FROM employee, jobs WHERE ((jobs.max_lvl = 227)) AND employee.job_id = jobs.job_id	SELECT employee.emp_id, employee.fname FROM employee, jobs WHERE ((jobs.max_lvl = 227)) AND employee.job_id = jobs.job_id
12	SELECT employee.emp_id FROM employee WHERE ((employee.hire_date = #1991/02/13#))	SELECT employee.emp_id FROM employee WHERE ((employee.hire_date = #1991/02/13#))
13	SELECT titles.title FROM titles WHERE ((titles.advance > 5000))	SELECT titles.title FROM titles WHERE ((titles.advance > 5000))
14	SELECT titles.title, titles.title_id FROM titles WHERE ((titles.price > 19.99) AND (titles.type LIKE '%business%'))	SELECT titles.title, titles.title_id FROM titles WHERE ((titles.price > 19.99) AND (titles.type LIKE '%business%'))
15	SELECT publishers.pub_name FROM publishers WHERE ((publishers.country LIKE 'Germany'))	SELECT publishers.pub_name FROM publishers WHERE ((publishers.country LIKE 'Germany'))
16	SELECT titleauthor.au_id FROM titleauthor, titles WHERE ((titles.title LIKE '%the%') OR (titles.title LIKE '%busy%')) AND titleauthor.title_id = titles.title_id	SELECT titleauthor.au_id FROM titleauthor, titles WHERE ((titles.title LIKE '%the%') OR (titles.title LIKE '%busy%')) AND titleauthor.title_id = titles.title_id
17	SELECT titleauthor.title_id FROM authors, titleauthor WHERE ((authors.au_lname LIKE 'Smith')) AND titleauthor.au_id = authors.au_id	SELECT titleauthor.title_id FROM authors, titleauthor WHERE ((authors.au_lname LIKE 'Smith')) AND titleauthor.au_id = authors.au_id

18	SELECT titleauthor.title_id FROM authors, titleauthor WHERE ((authors.au_lname LIKE 'green')) AND titleauthor.au_id = authors.au_id	SELECT titleauthor.title_id FROM authors, titleauthor WHERE ((authors.au_lname LIKE 'green')) AND titleauthor.au_id = authors.au_id
19	SELECT employee.job_id FROM employee, jobs WHERE ((employee.fname LIKE 'Francisco') OR (employee.lname LIKE 'Chang')) AND employee.job_id = jobs.job_id	SELECT employee.job_id FROM employee, jobs WHERE ((employee.fname LIKE 'Francisco') OR (employee.lname LIKE 'Chang')) AND employee.job_id = jobs.job_id
20	SELECT employee.hire_date, employee.job_id FROM employee, jobs WHERE ((employee.fname LIKE 'Paolo')) AND employee.job_id = jobs.job_id	SELECT employee.hire_date, employee.job_id FROM employee, jobs WHERE ((employee.fname LIKE 'Paolo')) AND employee.job_id = jobs.job_id
21	SELECT stores.stor_address FROM stores WHERE ((stores.stor_address LIKE 'Barnum´s'))	SELECT stores.stor_address FROM stores WHERE ((stores.stor_address LIKE 'Barnum´s'))
22	SELECT authors.phone FROM authors WHERE ((authors.au_fname LIKE 'cheryl'))	SELECT authors.phone FROM authors WHERE ((authors.au_fname LIKE 'cheryl'))
23	SELECT publishers.city FROM publishers WHERE ((publishers.pub_name LIKE '%New%') OR (publishers.pub_name LIKE '%Moon%') OR (publishers.pub_name LIKE '%Books%'))	SELECT publishers.city FROM publishers WHERE ((publishers.pub_name LIKE '%New%') OR (publishers.pub_name LIKE '%Moon%') OR (publishers.pub_name LIKE '%Books%'))
24	SELECT employee.job_lvl FROM employee WHERE ((employee.fname LIKE 'philip') OR (employee.lname LIKE 'cramer'))	SELECT employee.job_lvl FROM employee WHERE ((employee.fname LIKE 'philip') OR (employee.lname LIKE 'cramer'))
25	SELECT titles.price FROM publishers, titleauthor, titles WHERE ((publishers.pub_id = '1389')) AND titles.pub_id = publishers.pub_id AND titleauthor.title_id = titles.title_id	SELECT titles.price FROM publishers, titleauthor, titles WHERE ((publishers.pub_id = '1389')) AND titles.pub_id = publishers.pub_id AND titleauthor.title_id = titles.title_id
26	SELECT titles.pub_id FROM publishers, titleauthor, titles WHERE ((publishers.pub_name LIKE 'GGG&G')) AND titles.pub_id = publishers.pub_id AND titleauthor.title_id = titles.title_id	SELECT titles.pub_id FROM publishers, titleauthor, titles WHERE ((publishers.pub_name LIKE 'GGG&G')) AND titles.pub_id = publishers.pub_id AND titleauthor.title_id = titles.title_id
27	SELECT titles.price, titles.title_id FROM titles WHERE ((titles.title LIKE '%you%') OR (titles.title LIKE '%can%'))	SELECT titles.price, titles.title_id FROM titles WHERE ((titles.title LIKE '%you%') OR (titles.title LIKE '%can%'))
28	SELECT authors.address, authors.phone FROM authors, titleauthor, titles WHERE ((titles.title LIKE '%you%') OR (titles.title LIKE '%can%')) AND titleauthor.title_id = titles.title_id AND titleauthor.au_id = authors.au_id	SELECT authors.address, authors.phone FROM authors, titleauthor, titles WHERE ((titles.title LIKE '%you%') OR (titles.title LIKE '%can%')) AND titleauthor.title_id = titles.title_id AND titleauthor.au_id = authors.au_id
29	SELECT authors.city FROM authors WHERE ((authors.au_fname LIKE 'Johnson') OR (authors.au_lname LIKE 'White'))	SELECT authors.city FROM authors WHERE ((authors.au_fname LIKE 'Johnson') OR (authors.au_lname LIKE 'White'))
30	SELECT employee.lname FROM employee WHERE ((employee.fname LIKE 'Pedro'))	SELECT employee.lname FROM employee WHERE ((employee.fname LIKE 'Pedro'))
31	SELECT stores.city FROM stores WHERE ((stores.stor_name LIKE 'Bookbeat'))	SELECT stores.city FROM stores WHERE ((stores.stor_name LIKE 'Bookbeat'))
32	SELECT publishers.state FROM publishers WHERE ((publishers.pub_name LIKE '%Ramona%') OR (publishers.pub_name LIKE '%publishers%'))	SELECT publishers.state FROM publishers WHERE ((publishers.pub_name LIKE '%Ramona%') OR (publishers.pub_name LIKE '%publishers%'))
33	SELECT titles.title FROM publishers, titles	SELECT titles.title FROM publishers, titles

	WHERE ((publishers.state LIKE 'TX')) AND titles.pub_id = publishers.pub_id	WHERE ((publishers.state LIKE 'TX')) AND titles.pub_id = publishers.pub_id
34	SELECT employee.fname, employee.lname FROM employee, publishers WHERE ((publishers.pub_name LIKE 'GGG&G')) AND employee.pub_id = publishers.pub_id	SELECT employee.fname, employee.lname FROM employee, publishers WHERE ((publishers.pub_name LIKE 'GGG&G')) AND employee.pub_id = publishers.pub_id
35	SELECT jobs.job_desc FROM employee, jobs WHERE ((employee.emp_id LIKE 'VPA30890F')) AND employee.job_id = jobs.job_id	SELECT jobs.job_desc FROM employee, jobs WHERE ((employee.emp_id LIKE 'VPA30890F')) AND employee.job_id = jobs.job_id
36	SELECT stores.stor_name FROM stores, titles, sales WHERE ((titles.title LIKE '%cooking%') OR (titles.title LIKE '%with%')) AND sales.title_id = titles.title_id AND sales.stor_id = stores.stor_id	SELECT stores.stor_name FROM stores, titles, sales WHERE ((titles.title LIKE '%cooking%') OR (titles.title LIKE '%with%')) AND sales.title_id = titles.title_id AND sales.stor_id = stores.stor_id
37	SELECT employee.hire_date FROM employee WHERE ((employee.emp_id LIKE 'PTC11962M'))	SELECT employee.hire_date FROM employee WHERE ((employee.emp_id LIKE 'PTC11962M'))
38	SELECT COUNT(authors.au_id) FROM authors WHERE ((authors.city LIKE '%Berkeley%'))	SELECT COUNT(authors.au_id) FROM authors WHERE ((authors.city LIKE '%Berkeley%'))
39	SELECT COUNT(employee.emp_id) FROM employee, publishers WHERE ((publishers.pub_name LIKE 'Scotney') OR (publishers.pub_name LIKE 'book')) AND employee.pub_id = publishers.pub_id	SELECT COUNT(employee.emp_id) FROM employee, publishers WHERE ((publishers.pub_name LIKE 'Scotney') OR (publishers.pub_name LIKE 'book')) AND employee.pub_id = publishers.pub_id
40	SELECT COUNT(sales.qty) FROM sales WHERE ((sales.ord_date = #1994/09/14#))	SELECT COUNT(sales.qty) FROM sales WHERE ((sales.ord_date = #1994/09/14#))
41	SELECT COUNT(titles.title_id) FROM titles WHERE ((titles.pubdate = #1994/09/13#))	SELECT COUNT(titles.title_id) FROM titles WHERE ((titles.pubdate = #1994/09/13#))
42	SELECT titleauthor.au_id, titles.pub_id, titles.title, titleauthor.title_id, titles.title_id FROM titleauthor, titles WHERE titleauthor.title_id = titles.title_id	SELECT titleauthor.au_id, titles.pub_id, titles.title, titleauthor.title_id, titles.title_id FROM titleauthor, titles WHERE titleauthor.title_id = titles.title_id
43	SELECT authors.city FROM authors WHERE ((authors.zip = '89076'))	SELECT authors.city FROM authors WHERE ((authors.zip = '89076'))
44	SELECT employee.hire_date FROM employee WHERE ((employee.fname LIKE 'Pedro'))	SELECT employee.hire_date FROM employee WHERE ((employee.fname LIKE 'Pedro'))
45	SELECT pub_info.pub_id, publishers.pub_id, titles.pub_id, publishers.pub_name, titles.title, titleauthor.title_id, titles.title_id FROM pub_info, publishers, titleauthor, titles WHERE titleauthor.title_id = titles.title_id AND titles.pub_id = publishers.pub_id AND publishers.pub_id = pub_info.pub_id	SELECT pub_info.pub_id, publishers.pub_id, titles.pub_id, publishers.pub_name, titles.title, titleauthor.title_id, titles.title_id FROM pub_info, publishers, titleauthor, titles WHERE titleauthor.title_id = titles.title_id AND titles.pub_id = publishers.pub_id AND publishers.pub_id = pub_info.pub_id
46	SELECT COUNT(sales.qty) FROM sales, titles WHERE ((titles.title LIKE '%sillicon%') OR (titles.title LIKE '%valley%')) AND sales.title_id = titles.title_id	SELECT COUNT(sales.qty) FROM sales, titles WHERE ((titles.title LIKE '%sillicon%') OR (titles.title LIKE '%valley%')) AND sales.title_id = titles.title_id
47	SELECT publishers.pub_name FROM employee, pub_info, publishers WHERE ((employee.fname LIKE 'victoria') OR (employee.lname LIKE 'ashworth')) AND employee.pub_id = publishers.pub_id AND publishers.pub_id = pub_info.pub_id	SELECT publishers.pub_name FROM employee, pub_info, publishers WHERE ((employee.fname LIKE 'victoria') OR (employee.lname LIKE 'ashworth')) AND employee.pub_id = publishers.pub_id AND publishers.pub_id = pub_info.pub_id
48	SELECT stores.stor_name FROM stores, titles, sales WHERE ((stores.stor_name LIKE '%the%') OR (titles.title LIKE '%busy%')) AND sales.title_id = titles.title_id AND sales.stor_id = stores.stor_id	SELECT stores.stor_name FROM stores, titles, sales WHERE ((stores.stor_name LIKE '%the%') OR (titles.title LIKE '%busy%')) AND sales.title_id = titles.title_id AND sales.stor_id = stores.stor_id

49	SELECT CustomerCustomerDemo.CustomerTypeID FROM titles WHERE ((titles.title LIKE '%the%') OR (titles.title LIKE '%busy%'))	SELECT CustomerCustomerDemo.CustomerTypeID FROM titles WHERE ((titles.title LIKE '%the%') OR (titles.title LIKE '%busy%'))
50	SELECT stores.stor_address FROM stores WHERE ((stores.stor_name LIKE 'Barnum''s'))	SELECT stores.stor_address FROM stores WHERE ((stores.stor_name LIKE 'Barnum''s'))

.

Anexo G.- Instalación y configuración de la interfaz en Windows 7 de 64 bits.

1.- Requerimientos.

Tener instalado el JDK 1.6.0_25 y el JRE 1.6.0_25. Para 32 bits (X86) ó superior.

Tener instalado el Netbeans 7.0 ó superior.

2.- Instalación de la interfaz.

2.1.- Copiar la carpeta “Interfaz” que contiene el sistema a la ubicación donde por default Netbeans crea lo nuevos proyectos que en general es: “C:\Users\ITCM\Documents\NetBeansProjects”, es decir en “Documentos” del usuario que está utilizando el CPU dentro de la carpeta “NetBeansProjects”.

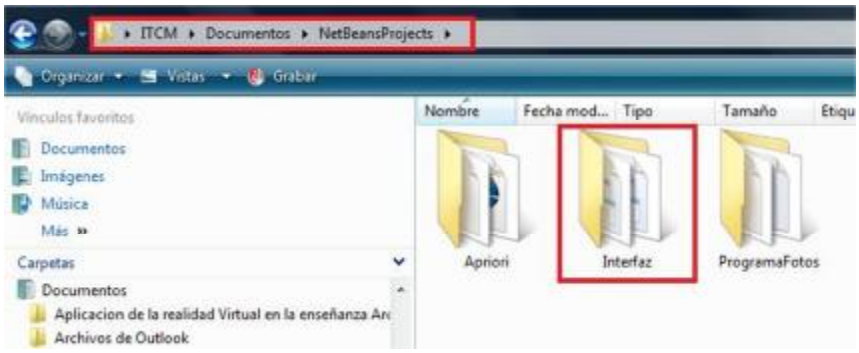


Figura 39.- Carpeta de proyectos de NetBeans.

2.2.- Desde Netbeans abrir el proyecto.

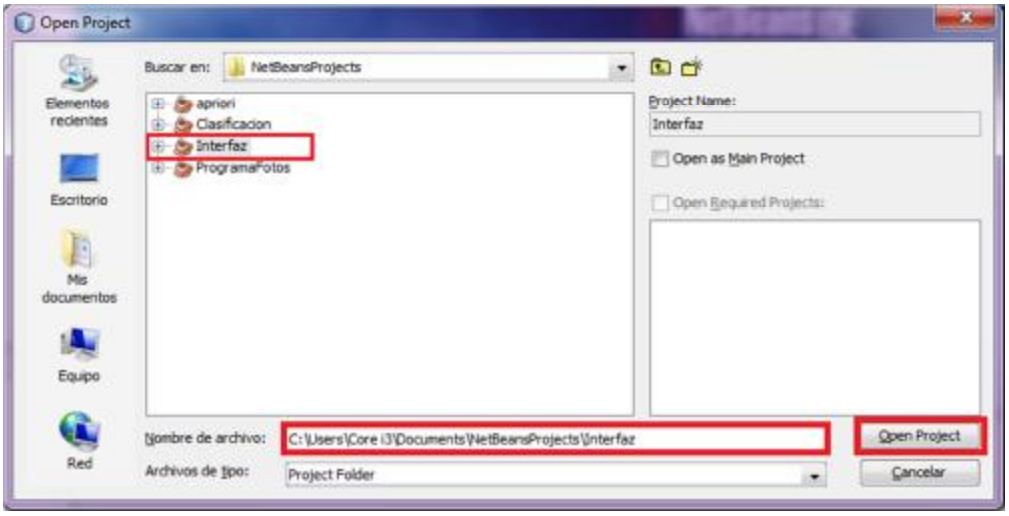


Figura 40.- Abrir el proyecto "Interfaz".

2.3.- El sistema de la interfaz utiliza ciertas librerías que no están disponibles al momento de abrirla y marcará varios errores, por lo tanto hay que agregar dichas librerías que se encuentran en el archivo "jdsl.zip" que está en la misma carpeta del sistema "Interfaz".

2.4.- Hacer clic derecho con el mouse sobre la carpeta "Libraries" del proyecto "Interfaz"y seleccionar la opción "Add JAR/Folder…"

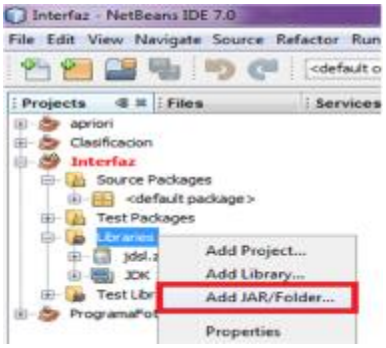


Figura 41.- Añadir librería al proyecto.

2.5.- Se abre un cuadro de diálogo, en el campo "Buscar en:" colocarse en la carpeta "Interfaz" del sistema, seleccionar el archivo "jdsl.zip" y hacer clic en el botón "Abrir".

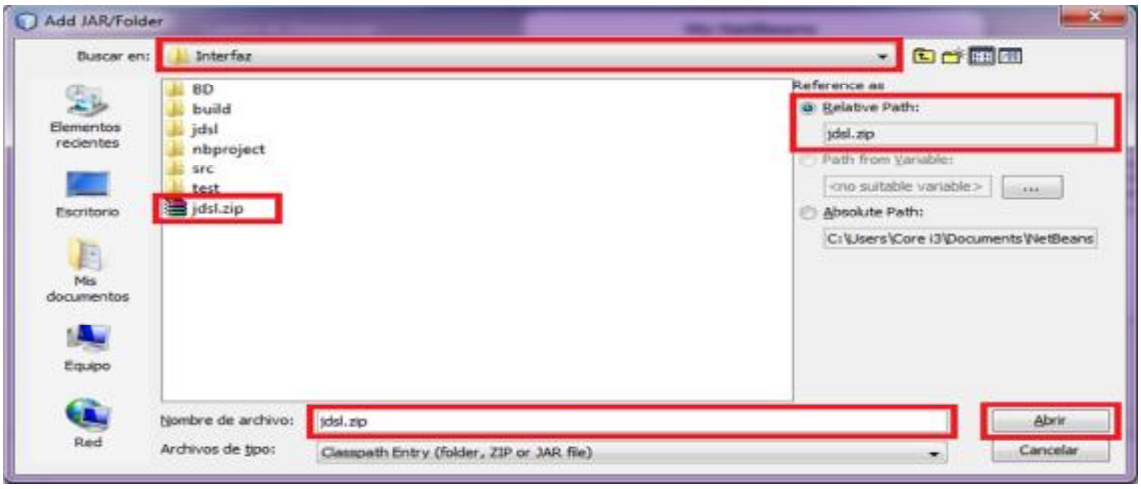


Figura 42.- seleccionar el archivo "jdsl.zip".

2.6.- Al agregar las librerías ya no marcará errores en ninguna clase.

3.- Cambiar el Path de los orígenes de datos ODBC.

3.1. El controlador de Windows para datos ODBC se llama "odbcad.exe", primero hay que verificar que este archivo se encuentre en el directorio "c:\WINDOWS\System32\odbcad.exe", para aplicaciones de 64 bits y en "c:\WINDOWS\SysWOW64\odbcad.exe", para aplicaciones de 32 bits.

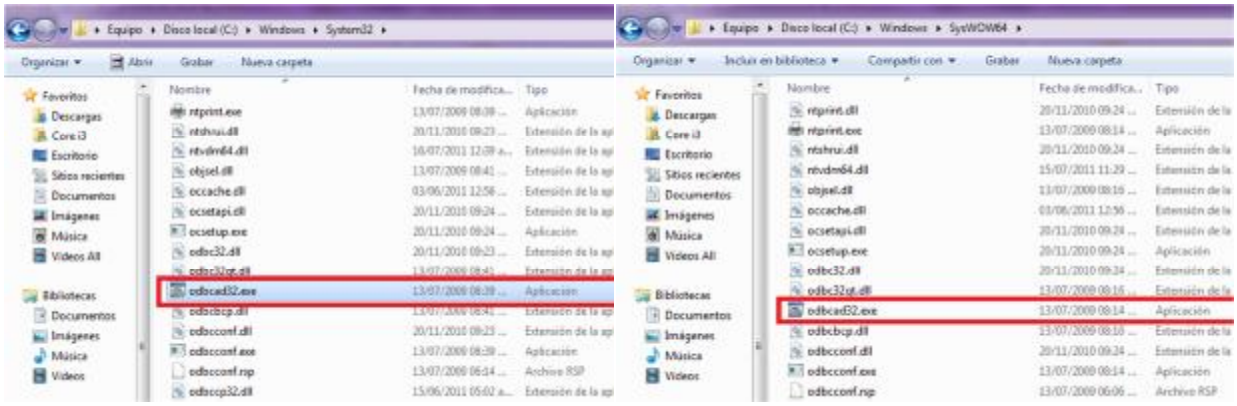


Figura 43.- Archivo "odbcad.exe" de origenes de datos ODBC.

3.2.- Ahora se tiene que cambiar el path de los orígenes de datos ODBC. Entrar al Panel de control, a "Sistema y seguridad" y después a "Herramientas administrativas".

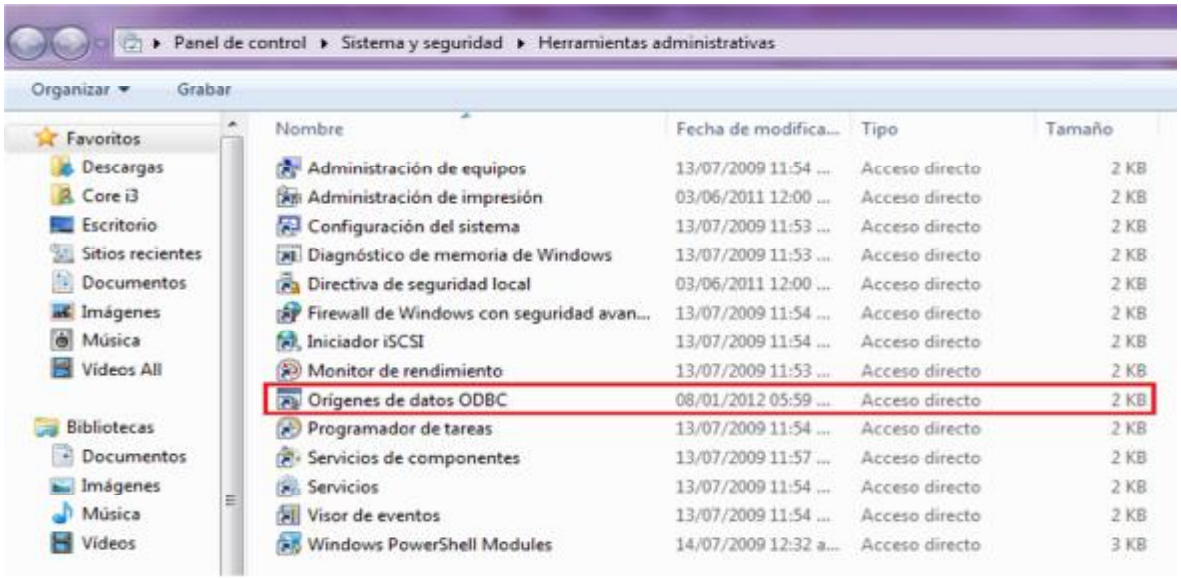


Figura 44.- Cambiar el path de los orígenes de datos ODBC.

3.3.- Hacer clic derecho con el mouse sobre "Orígenes de datos ODBC" y seleccionar "propiedades" y se abrirá un cuadro de diálogo.

3.4.- En el cuadro de diálogo irse a la pestaña "Acceso directo" y cambiar el campo "Destino:".

Cambiar: %windir%\system32\odbcad32.exe

Por: %windir%\SysWOW64\odbcad32.exe

Una vez hecho esto los orígenes de datos ODBC, utilizarán la aplicación para 32 bits en lugar de la aplicación para 64 bits.

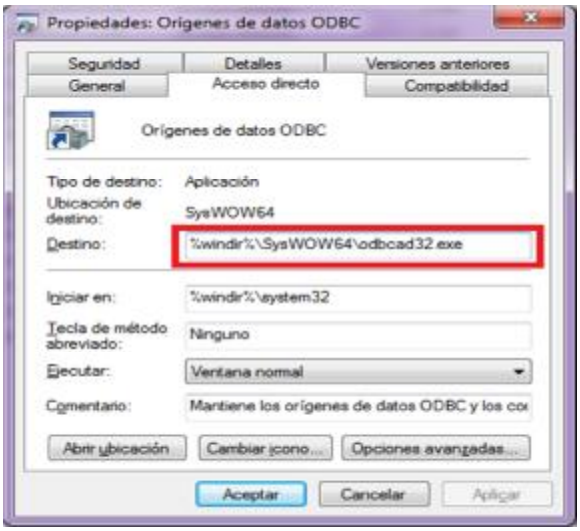


Figura 45.- Propiedades de orígenes de datos ODBC.

4.- Dar de alta en Windows los orígenes de datos que utilizará el sistema.

4.1 En la misma ubicación del panel de control, es decir en “Sistema y seguridad” dentro de “Herramientas administrativas”, hacer doble clic sobre el acceso directo “Orígenes de datos ODBC”.

En el paso anterior ya se modificó para que tenga acceso a la aplicación para 32 bits, para que se puedan dar de alta los orígenes de datos, de lo contrario no se podría ejecutar el programa.

Se abre un cuadro de diálogo, nos posicionamos en la pestaña “DNS de usuario”.

4.2.- En el cuadro de diálogo “Administrador de orígenes de datos ODBC”, hacer clic en el botón agregar. Se abre un nuevo cuadro de diálogo llamado “Crear nuevo origen de datos”, en donde se selecciona la opción “Driver do Microsoft Access (*.mdb)” y se hace clic en el botón de Finalizar.

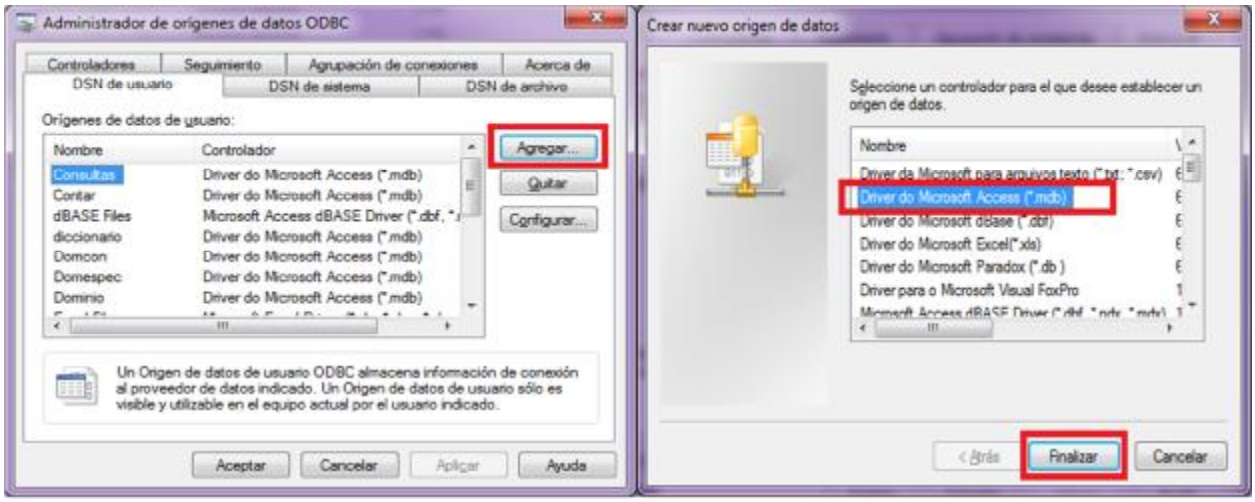


Figura 46.- Crear nuevo origen de datos de Microsoft Access.

4.3.- Se abre un cuadro de diálogo en donde se establece el “Nombre del origen de datos”, alguna descripción, y se selecciona la base de datos de Access que estará vinculado al origen de datos.

En este caso en Nombre se escribirá “Consultas”, en descripción “Consultas de la base de datos” y se hace clic en el botón “Seleccionar” de la sección de base de datos.

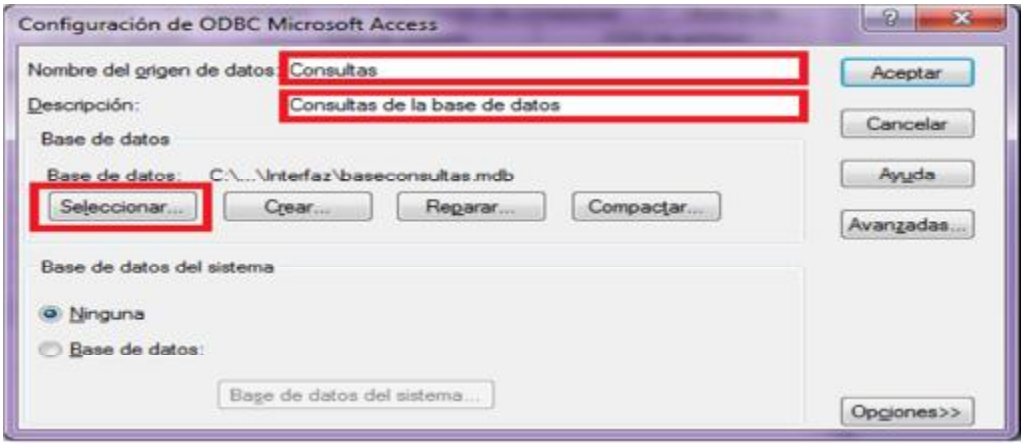


Figura 47.- Configuración del origen de datos.

4.4.- Al dar clic en seleccionar se abre el cuadro de diálogo. En la sección de “Directorios” nos ubicamos en la carpeta “Interfaz” donde se encuentra la aplicación y las bases de datos.

Del lado izquierdo aparecen todas las bases de datos disponibles, en este caso seleccionamos el archivo “baseconsultas.mbd” y damos “Aceptar”.

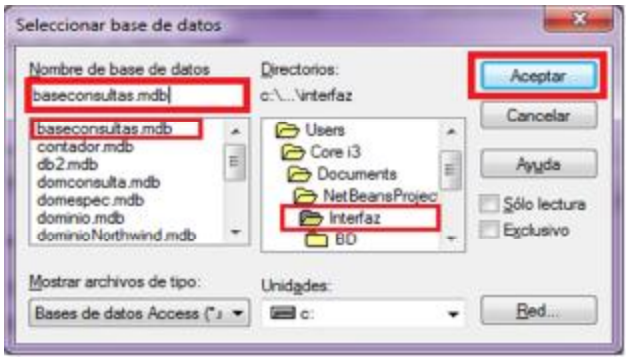


Figura 48.- Seleccionar BD del origen de datos.

4.5.- Se regresa automáticamente al cuadro de diálogo inicial que es “Administrador de orígenes de datos ODBC”, donde en la sección “Orígenes de datos de usuario:” ya se agregó el nuevo origen de “Consultas”.

Siguiendo el mismo procedimiento se agregan todos los orígenes de datos que requiere el sistema que se muestran en la siguiente Tabla.

Nombre del Origen	Base de datos.
Consultas	baseconsultas.mdb
Contar	contador.mdb
Diccionario	db2.mdb
Domcon	domconsulta.mdb
Domespec	domespec.mdb
Dominio	dominio.mdb
Grafo	Grafo.mdb
Lexicon	lexicon.mdb
Metadatos	metadatos.mdb
Northwind	Northwind.mdb
Palabra	Palabras.mdb
Pubs	Pubs.mdb
Sinonimo	db2.mdb

Tabla 21.- Orígenes de datos utilizados por la interfaz.

Una vez agregados todos los orígenes de datos necesarios, dar clic en el botón “Aceptar” del cuadro de diálogo “Administrador de orígenes de datos ODBC”.

5.- Agregar en NetBeans la referencia al JDK de 32 bits que se tiene instalado.

5.1.- NetBeans por default utiliza el JDK que se encuentre en “Archivos de programa”, es decir el JDK de 64 bits. Por lo tanto hay que agregar la referencia al JDK de 32 bits e indicarle a la interfaz de NetBeans que la utilice.

5.2.- Para agregar la referencia al JDK de 32 bits, en NetBeans ir al menú “Tools” y seleccionar “Java Platforms”, donde se agregarán estas referencias.

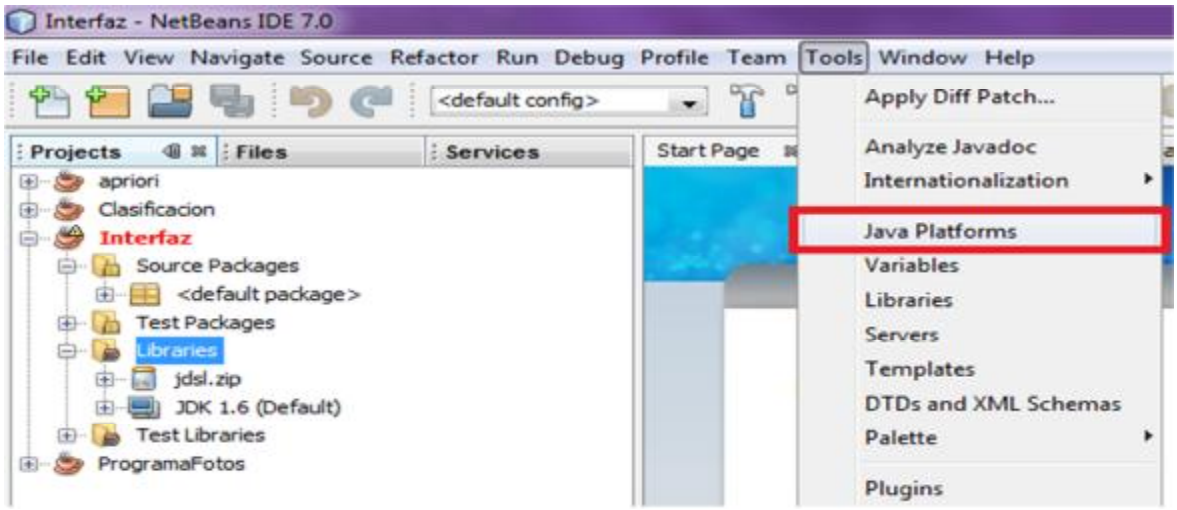


Figura 49.- Opción "Java Platforms" del menú "Tools".

5.3.- Se abre el cuadro de diálogo “Java Platform Manager”, donde nos muestra las referencias a los JDK instalados y Cuál esta seleccionado por Default. Hacer clic en el botón “Add Platform…”.

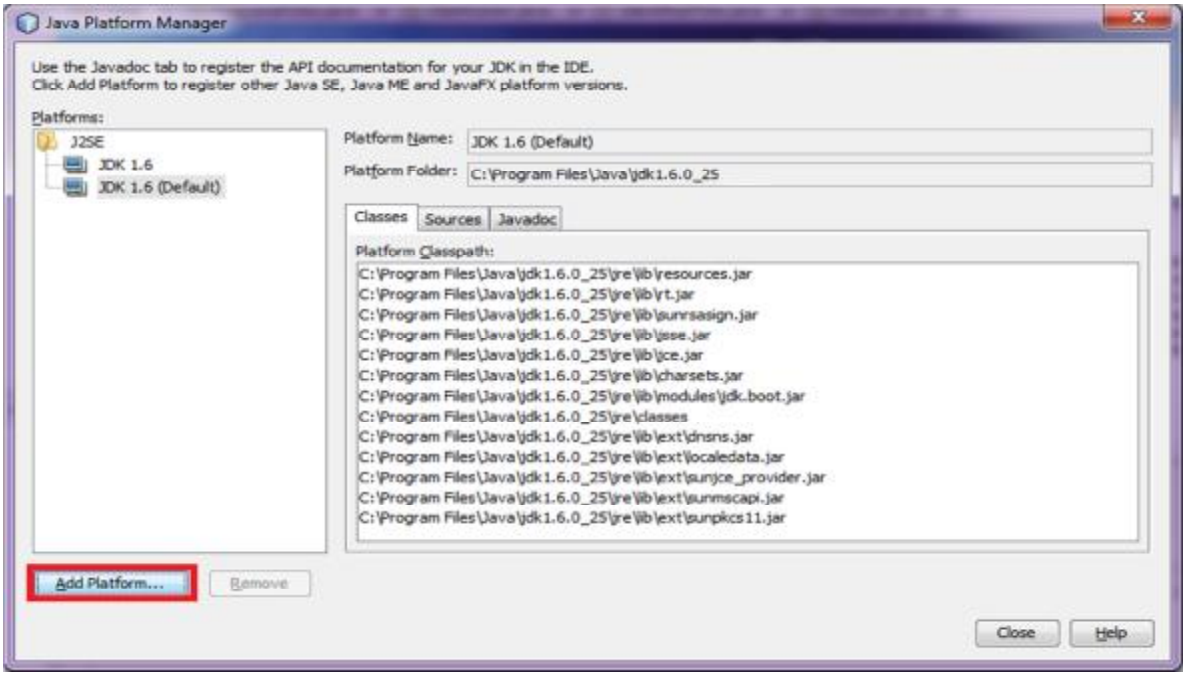


Figura 50.- Ventana para agregar plataforma el NetBeans

5.4.- Se abre un nuevo cuadro de diálogo llamado “Add Java Platform”, donde en la sección “Buscar en:” debemos ubicarnos primero en “c:\” y después en la carpeta “Archivos de programa (x86)”, una vez dentro de la carpeta entramos a la carpeta “Java” y seleccionamos la carpeta del JDK en este caso “jdk1.6.0_25” y hacemos un clic en el botón “Next”.

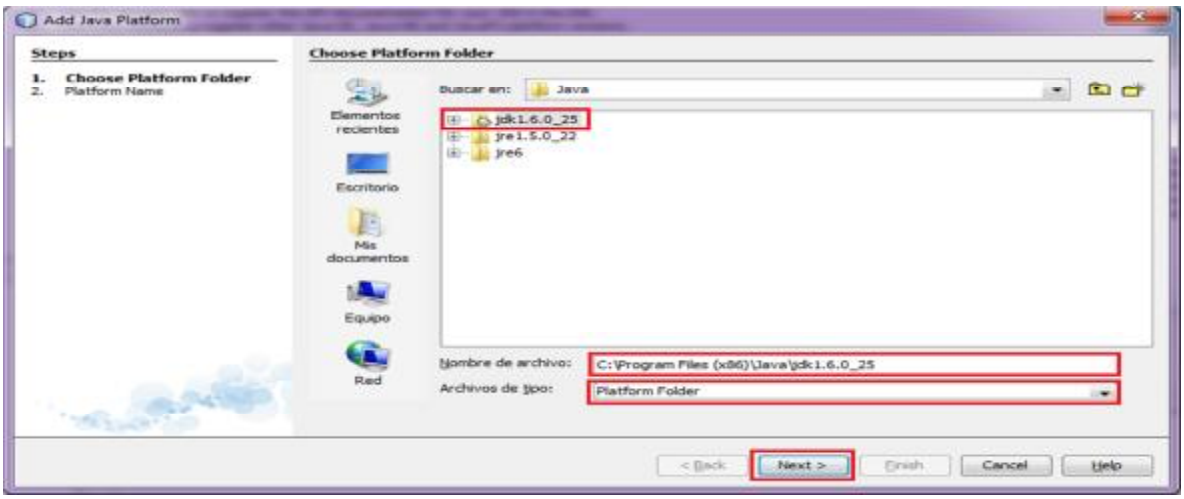


Figura 51.- Elegir la maquina virtual (JDK) que utilizará NetBeans.

5.5.- En el siguiente cuadro de diálogo podemos especificar un nombre a la referencia de la plataforma, y damos clic en finalizar, es decir en el botón “Finish”. Nos regresa al cuadro de diálogo inicial, se agrega la nueva referencia a la plataforma que está en el campo “Platform Folder” y nos muestra las ubicaciones de cada una de las clases.

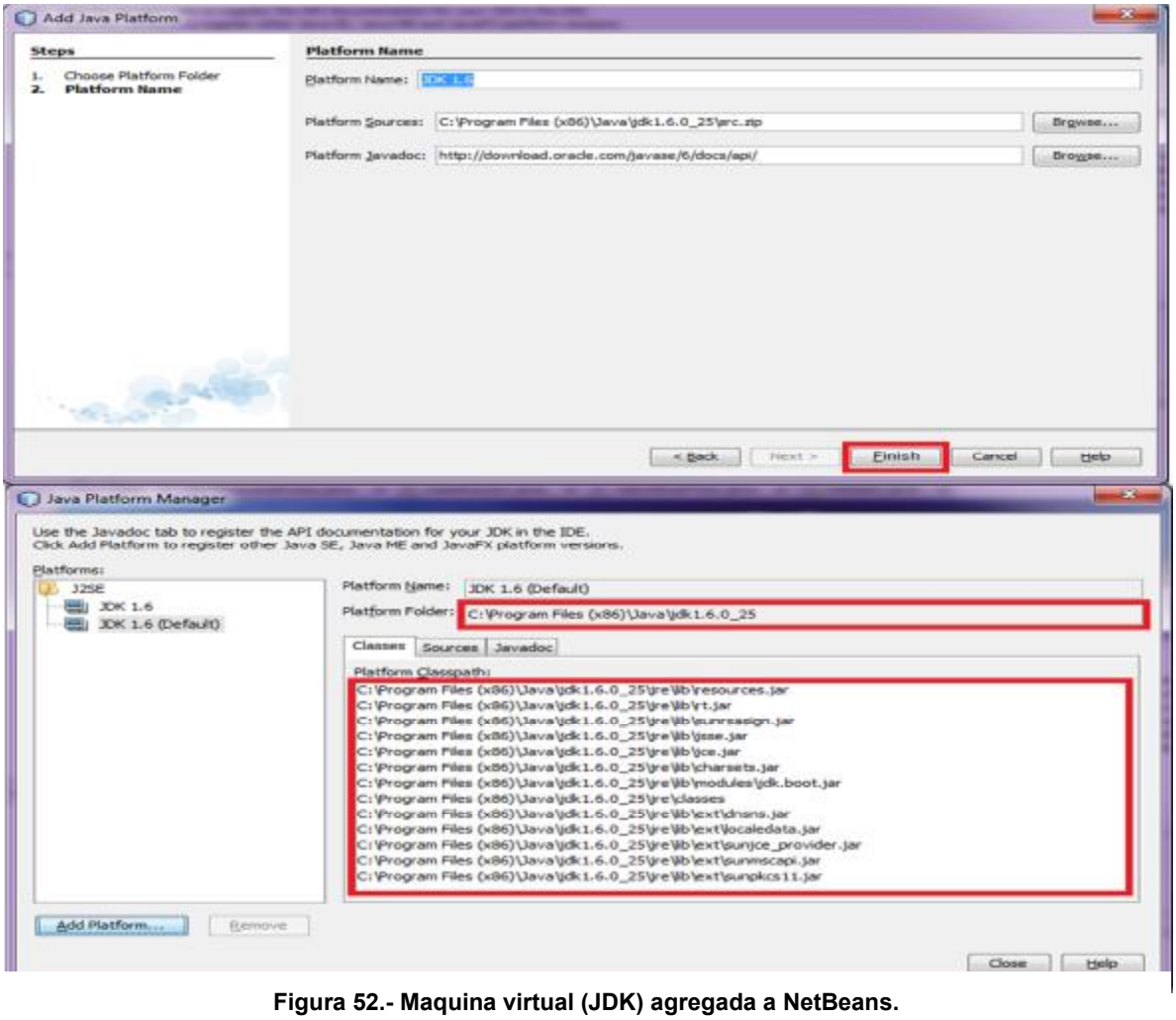


Figura 52.- Maquina virtual (JDK) agregada a NetBeans.

6.- Cambiar la dirección de inicio de la interfaz de NetBeans al JDK de 32 bits.

6.1.- Ahora ya se tiene instalada la referencia al JDK de 32 bits, sin embargo NetBeans, seguirá utilizando el JDK de 64 bits porque así lo tiene configurado. Debemos cambiar esta configuración.

6.2.- Con el explorador de Windows nos ubicamos en “C:\” luego en “Archivos de programa”, entramos a la carpeta “NetBeans 7.0” y después a la carpeta “etc”.

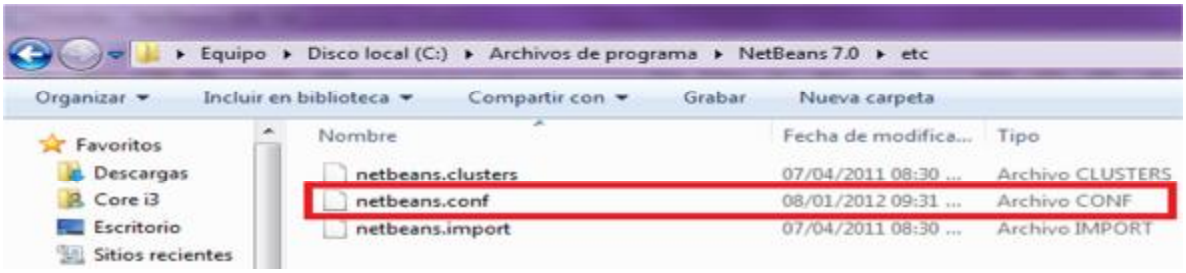


Figura 53.- Archivo de configuración de NetBeans.

6.3.- En esta carpeta existe un archivo llamado “netbeans.conf”, abrimos este archivo. En el cuadro de diálogo nos preguntará con que programa deseamos abrirlo, seleccionamos “WordPad” y damos clic en el botón “Aceptar”.

Al abrirlo veremos varias líneas de texto en inglés, donde las líneas que comienzan con el símbolo “#” son solo comentarios que no se toman en cuenta en la configuración.

6.4.- En este archivo debemos localizar la línea que comienza con la palabra “netbeans_jdkhome=” y que tiene una dirección por ejemplo “C:\Program Files\Java\jdk1.6.0_25”, comentar la línea con el símbolo “#” y agregar una nueva cambiando la ruta al JDK de 32 bits que se tenga instalado.

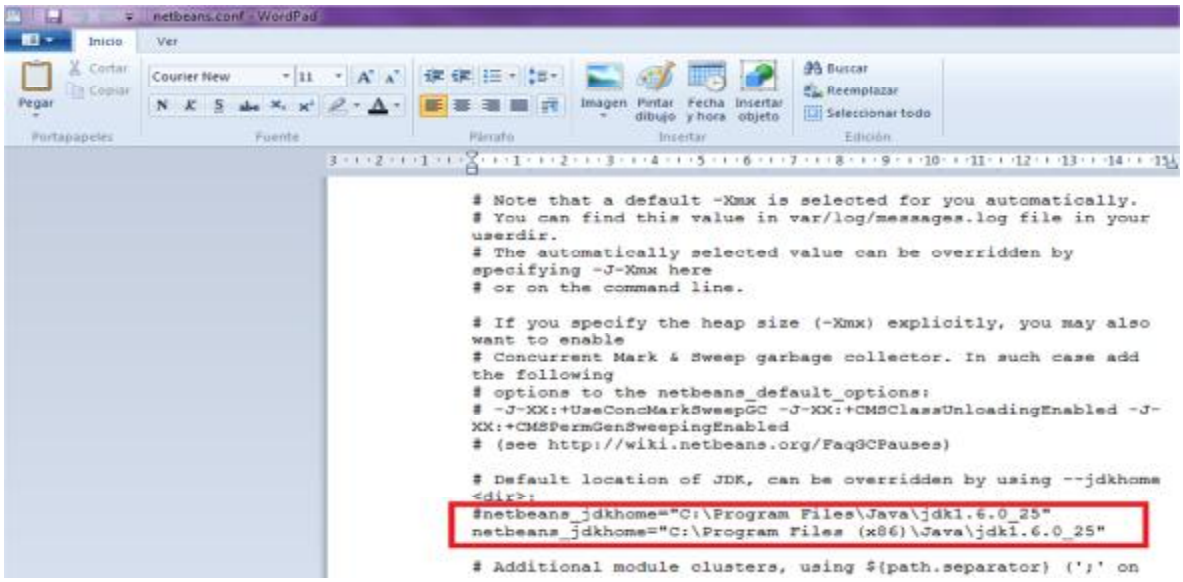


Figura 54.- Contenido del archivo de configuración de NetBeans.

6.5.- Una vez hecho esto, debemos cerrar NetBeans y volverlo a abrir para que ya inicie utilizando el JDK de 32 bits, y se debe establecer como proyecto principal a “Interfaz”.

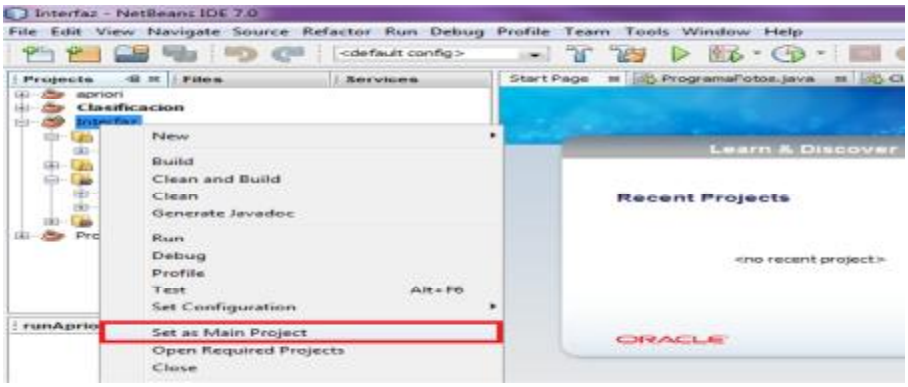


Figura 55.- Establecer el proyecto principal.

7.- Habilitar permisos para crear archivos en “C:\” y evitar el problema de escritura de archivos.

7.1.- Con los pasos anteriores ya debería funcionar adecuadamente el sistema de la interfaz, sin embargo en algunas PC. Windows tiene bloqueada la escritura en el Directorio “C:\” y esto ocasionaría una excepción de escritura al ejecutar el programa. Para evitar este error quitaremos las notificaciones del control de cuentas de usuario.

7.2.- Entrar al panel de control, luego a “Sistema y seguridad” y después a “Centro de actividades”. Del lado izquierdo hacer clic en “Cambiar configuración de control de cuentas de usuario.

Aparecerá un cuadro de diálogo donde se permite elegir cuando se desea recibir notificaciones acerca de cambios en el equipo, donde se recorrerá la barra hasta lo más bajo que es “No notificarme nunca” y dar clic en el botón “Aceptar”

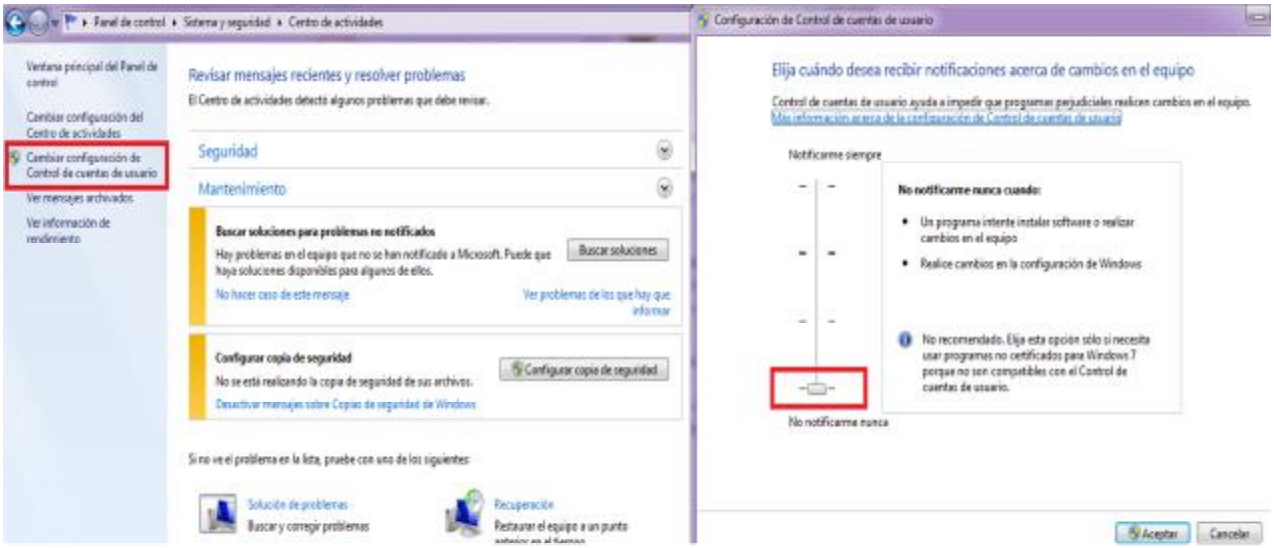


Figura 56.- Control de cuentas de usuario.

Ahora solo se ejecutará el programa y debe funcionar correctamente.

Anexo H.- Código de la Clase Dialogo.

```
import java.awt.Frame;
import java.awt.GridLayout;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.sql.*;
import java.util.ArrayList;
import java.util.Properties;
import javax.swing.JButton;
import javax.swing.JCheckBox;
import javax.swing.JDialog;
import javax.swing.JLabel;
import javax.swing.JOptionPane;

public class Dialogo extends JDialog implements ActionListener {

   ArrayList<Integer> seleccion = new ArrayList();
   final JCheckBox[] a;
   String FuncionAgregacion="";
   String ValorDeLaConsulta="";

   public Dialogo(Frame parent, ArrayList<String> opciones, String Consulta, String SustValor, String
FuncDeAgregacion) throws SQLException {
      super(parent,"Aclaración de la consulta.",true );
      setDefaultCloseOperation(javax.swing.WindowConstants.DISPOSE_ON_CLOSE);
      super.setResizable(false);
      GridLayout contenedor = new GridLayout(0, 1);
      contenedor.setVgap(5);
      contenedor.setHgap(10);
      getContentPane().setLayout(contenedor);
      JLabel lblTitle=new JLabel("Consulta: " + Consulta);
      add(lblTitle);
      ValorDeLaConsulta = SustValor;
      FuncionAgregacion = FuncDeAgregacion;
      String PalabraAgregacion = "";
      if (FuncDeAgregacion.equals("MAX"))
      {
         PalabraAgregacion = "el Mayor";
      }
      else if (FuncDeAgregacion.equals("MIN"))
      {
         PalabraAgregacion = "el Menor";
      }
      else if (FuncDeAgregacion.equals("COUNT"))
      {
         PalabraAgregacion = "la cantidad de registros";
```

```
}

int NumDeValores = ValorDeLaConsulta.trim().split(" ").length;
JLabel lblDes =new JLabel("");
if (SustValor.equals(""))
{
   if (FuncDeAgregacion.equals(""))
   {
      lblDes.setText("¿Qué es lo que desea obtener en esta consulta?");
   }
   else
   {
      lblDes.setText("¿De que columna deseas obtener " + PalabraAgregacion + "?");
   }

}
else
   if(NumDeValores==1)
   {
      lblDes.setText("¿A que columna haces referencia con el valor " + SustValor + "?");
   }
   else
   {
      lblDes.setText("¿A que columnas haces referencia con los valores " + SustValor + "?");
   }

add(lblDes);

//Se editan los nombres de las columnas para ver las opciones en español.

//Se indica que el CharSet es "iso-8859-1", es decir latinoamericano.
Properties PropLatino = new Properties();
PropLatino.put ("charSet", "iso-8859-1");
//Se realiza la conexion a la base de datos "Metadatos".
Connection conMetadatos = DriverManager.getConnection("jdbc:odbc:metadatos", PropLatino);
Statement stmMetadatos = conMetadatos.createStatement();
ResultSet rsMetadatos;
//Se recorren todas las opciones del dialogo recibidas.
ArrayList<String> OpcionesDescripcion = new ArrayList();
for(int j=0; j< opciones.size();j++)
{
   //Se divide la tabla y el campo.
   String[] Temporal = opciones.get(j).split("\\.+");
   String ConsultaActual = "SELECT DESCRIPCOLUM FROM Metadatos " +
                           "WHERE TABLENAME = '" + Temporal[0] + "' AND
                           COLUMNAME = '" + Temporal[1].replace("%", "") + "'";
   rsMetadatos = stmMetadatos.executeQuery(ConsultaActual);
   if(rsMetadatos.next() == true)
```

```
            {
                String Descripcion = rsMetadatos.getString("DESCRIPCOLUM").trim();
                OpcionesDescripcion.add(opciones.get(j) + " (" + Descripcion +")");
            }
            else
            {
                OpcionesDescripcion.add(opciones.get(j));
            }
        }

        a = new JCheckBox[OpcionesDescripcion.size()];
        for (int i = 0; i < a.length; i++) {
            a[i] = new JCheckBox(OpcionesDescripcion.get(i));
            add(a[i]);
        }
        JButton boton = new JButton("Aceptar");
        boton.addActionListener(this);
        add(boton);
        super.setLocationRelativeTo(null);
        super.pack();
        super.setVisible(true);

    }

    @Override
    public void actionPerformed(ActionEvent e) {
        seleccion.clear();
        boolean EsVacio = true;
        for (int i = 0; i < a.length; i++) {
            if (a[i].isSelected()) {
                seleccion.add(i);
                EsVacio=false;
            }
        }

        //Se pregunta si el arreglo de seleccion no es vacio.
        if (!EsVacio)
        {
            //Si no es vacio se pregunta si existe una funcion de agregacion, si no existe se cierra el dialogo.
            if (FuncionAgregacion.equals(""))
            {
                setVisible(false);
                dispose();
            }
            else
            {
                //Si existe una funcion de agregacion se pregunta si se seleccionaron mas de dos campos.
                if (seleccion.size()>1 && ValorDeLaConsulta.equals(""))
```

```
            {
               //Como es una funcion de agregacion no se pueden seleccionar varios campos.
               JOptionPane.showMessageDialog(this, "La consulta utiliza una funcion de agregación y
solo puede elegir un solo campo.");
            }
            else
            {
               //Si solo se seleccionó un campo se cierra el dialogo.
               setVisible(false);
               dispose();
            }
         }

      }
      else
      {
         JOptionPane.showMessageDialog(this, "Debe seleccionar al menos una opción.");
      }

   }
   public ArrayList<Integer> getSeleccion(){
      return seleccion;
   }
}
```