SEP **TNM**

INSTITUTO TECNOLÓGICO DE TIJUANA

DIVISIÓN DE ESTUDIOS DE POSGRADO E INVESTIGACIÓN

TRATAMIENTO DE ANOMALÍAS DE DISEÑO EN BASES DE DATOS PARA SU CONSULTA MEDIANTE UNA INTERFAZ DE LENGUAJE NATURAL

TRABAJO DE TESIS

Presentado por
ALAN GABRIEL AGUIRRE LAM

Para Obtener el Grado de
DOCTOR EN CIENCIAS EN COMPUTACIÓN

Director de Tesis
DR. RODOLFO A. PAZOS RANGEL

Co-Director de Tesis
DR. JOSÉ A. MARTÍNEZ FLORES

Tijuana, BC, Agosto, 2021

INSTITUTO TECNOLOGICO DE TIJUANA
POSGRADO EN COMPUTACION

Asunto: Se autoriza impresión de Trabajo de Tesis

Tijuana, B.C., 05 de Agosto del 2021

C. DRA. YAZMIN MALDONADO ROBLES
Jefe de la Div. de Estudios de Posgrado e Investigación
Presente.

En lo referente al trabajo de tesis escrito, con título " **Tratamiento de Anomalías de Diseño en Bases de Datos para Consulta Mediante una Interfaz de Lenguaje Natural"**, presentado por el ***C Alan Gabriel Aguirre Lam.,*** alumno del Doctorado en Ciencias en Computación con número de control **D06071536**, informamos a usted que se autoriza el escrito de tesis y se aprueba en todas sus partes, en virtud de reunir los requisitos de un trabajo de grado de Doctorado y a la vez se autoriza al interesado para que proceda de inmediato a la impresión del mismo y a presentar su examen de grado, ya que cumple con todos los requisitos.

ATENTAMENTE

DRA. ELBA PATRICIA MELIN OLMEDA
PRESIDENTE

DR. OSCAR CASTILLO LOPEZ
SECRETARIO

DRA. DANIELA ADRIANA SANCHEZ VIZCARRA
VOCAL

DR. FEVRIER ADOLFO VALDEZ ACOSTA
VOCAL

DR. JOSE MARIO GARCIA VALDEZ
VOCAL

c.c.p. Oficina de Titulación
c.c.p. División de Estudios de Posgrado e Investigación
c.c.p. Expediente
c.c.p. Interesado

EPMO/*inf

CARTA DECLARACIÓN DE PROPIEDAD INTELECTUAL

Tijuana, BC a 2 de julio del 2021

Yo **ALAN GABRIEL AGUIRRE LAM** reconozco que el Trabajo de Tesis de Doctorado que realice durante mis estudios en el Doctorado en Ciencias en Computación del Instituto Tecnológico de Tijuana fue parte del Proyecto de Investigación Titulado: **Tratamiento de anomalías de diseño en bases de datos para su consulta mediante una interfaz de lenguaje natural** que desarrolla mi director de tesis el **Dr. Rodolfo Abraham Pazos Rangel** y del cual es responsable del proyecto de investigación. Por esta razón, los métodos, modelos, algoritmos, y software realizados, así como datos y resultados obtenidos durante el desarrollo de mi tesis de doctorado son propiedad intelectual de mi Director y Codirector de Tesis, del Tecnológico Nacional de México, Instituto Tecnológico de Tijuana y del Conacyt, y No podré utilizarlos por mi cuenta durante, Ni después de terminar mi beca o estudios, excepto a solicitud escrita para poder utilizarlos bajo una colaboración directa con mi director el cual es responsable del proyecto de investigación. Por tanto, estoy de acuerdo en que No podre utilizar ni tomar modelos, ni datos utilizados en este proyecto de investigación y en el desarrollo de tesis para: presentaciones, publicaciones ni desarrollo de mi propia investigación que pudiera desarrollar una vez concluidos mis estudios**.**

Atentamente

ALAN GABRIEL AGUIRRE LAM

Estudiante de Doctorado en Ciencias en Computación

DECLARACIÓN DE ORIGINALIDAD

Tijuana, BC., 2 de Julio de 2021,

Yo, **Alan Gabriel Aguirre Lam**, estudiante del Doctorado en Ciencias en Computación, en mi calidad de autor manifiesto que este documento de tesis es producto de mi trabajo original y que no infringe los derechos de terceros, tales como derechos de publicación, derechos de autor, patente y similaridad. Por lo tanto, la obra realizada es de mi exclusiva autoría y no infringí en copiar el texto o imágenes, de fuentes de información por lo cual soy responsable del escrito que aquí se presenta.

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

En caso de presentarse cualquier reclamación o acción por parte de terceros en cuanto a los derechos de autor sobre la obra en cuestión, acepto toda la responsabilidad de tal infracción y relevo de esta a mi director de tesis, así como al Tecnológico Nacional de México, al Instituto Tecnológico de Tijuana y a sus respectivas autoridades.

Alan Gabriel Aguirre Lam
Estudiante de Doctorado en Ciencias en Computación

AGRADECIMIENTOS

Expreso mi total agradecimiento al Dr. Rodolfo A. Pazos por su pericia al dirigir el rumbo de este trabajo. Asimismo, agradezco a los Dres. José A. Martínez, Hector J. Fraire, Juan J. González y Juan M. Carpio por los conocimientos brindados a fin de mejorar los resultados de este trabajo. Es oportuno también agradecer al Consejo Nacional de Ciencia y Tecnología (CONACyT), al Instituto Tecnológico de Cd. Madero y al Instituto Tecnológico de Tijuana por el apoyo brindado para la realización de este proyecto. Por último, agradezco a toda mi familia por el apoyo incondicional que me brindaron, en especial a mi esposa Juana Gaspar y a mi hija Hannah J. Aguirre, quienes siempre estuvieron presentes en los retos que se presentaron.

PUBLICACIONES PRODUCTO DE ESTA TESIS

Revista Indizada en el Padrón Conacyt de Revistas Científicas

- Rodolfo A. Pazos, José A. Martínez, Alan G. Aguirre, "Processing Natural Language Queries via a Natural Language Interface to Databases with Design Anomalies", *POLIBITS*, vol. 62, pp. 43-50, 2020.

Capítulos de Libro

- Rodolfo A. Pazos, José A. Martínez, Alan G. Aguirre, Marco A. Aguirre, "Issues in Querying Databases with Design Anomalies Using Natural Language Interfaces", *Fuzzy Logic Augmentation of Neural and Optimization Algorithms: Theoretical Aspects and Real Applications*, Springer, pp. 461-473, 2018.

- Grigori Sidorov, Rodolfo A. Pazos, José A. Martínez, Juan M. Carpio, Alan G. Aguirre, "Configuration Module for Treating Design Anomalies in Databases for a Natural Language Interface to Databases", *Intuitionistic and Type-2 Fuzzy Logic Enhancements in Neural and Optimization Algorithms*, Springer, pp. 703-714, 2020.

RESUMEN

Las interfaces de lenguaje natural a bases de datos facilitan la consulta a bases de datos a usuarios inexpertos, es decir, usuarios que no tienen conocimiento sobre lenguajes formales de consulta a bases de datos. Sin embargo, los desarrolladores de estas herramientas hasta hoy en día diseñan sus interfaces pensando que serán usadas con bases de datos diseñadas correctamente, es decir, sin anomalías de diseño.

En la práctica es común encontrarse con bases de datos cuyo esquema no concuerda con las buenas prácticas de diseño de bases de datos. Esto se debe a que, en algunas ocasiones, así es requerido por las aplicaciones que utilizan estas bases de datos. Esta situación ocasiona que las ILNBDs no funcionen correctamente con BDs que adolecen de este problema.

En este trabajo se diseñaron e implementaron mecanismos que permiten a una interfaz de lenguaje natural a bases de datos dar tratamiento bases de datos que adolezcan de anomalías de diseño. Esto se realiza mediante la configuración de la interfaz para que ésta pueda tener una percepción de la base de datos sin anomalías de diseño y de esta forma pueda funcionar correctamente.

Los resultados obtenidos de esta implementación demuestran la eficacia de los mecanismos mencionados. Asimismo se realizaron comparaciones con otras interfaces que carecen de estos mecanismos, confirmando lo antes mencionado.

CONTENIDO

LISTA DE FIGURAS

LISTA DE TABLAS

Capítulo 1

Introducción

Hoy en día, las bases de datos relacionales se usan ampliamente en la mayoría de las empresas para almacenar información importante. En muchas de las actividades realizadas por dichas empresas se requiere obtener información de manera rápida y confiable. Sin embargo, para que un usuario pueda obtener información contenida en una base de datos (BD), éste debe tener conocimientos sobre un lenguaje formal de consulta a bases de datos, lo cual dificulta a la mayoría de los usuarios la obtención de la información requerida.

Existe una gran cantidad de herramientas de software en el mercado que facilitan la obtención de información de las BDs. Entre las más fáciles de usar por usuarios inexpertos se encuentran las Interfaces de lenguaje natural a bases de datos (ILNBDs). Dichas interfaces permiten a los usuarios formular consultas en su propio lenguaje, facilitando el acceso a información sin requerir conocimientos de un lenguaje formal de consulta a bases de datos o de programación.

Desafortunadamente, a pesar del gran número de ILNBDs (experimentales y comerciales) desarrolladas hasta la fecha, éstas no garantizan la traducción satisfactoria de consultas de lenguaje natural y, por ende, una respuesta confiable [Aguirre, 2014]. Los porcentajes de consultas traducidas correctamente (*recall*), reportados por tres ILNBDs son: 70-80% por ELF [ELF, 2009], 75% por C-PHRASE [Minock, 2010], 80% por PRECISE [Popescu, 2004].

El propósito de este trabajo es mejorar el desempeño de una ILNBD producto de un proyecto denominado Interfaz de Lenguaje Natural Español a Bases de Datos para Usuarios de Internet. Dicho proyecto comenzó a desarrollarse en el CENIDET desde septiembre del 2001, y se ha continuado en el ITCM desde agosto del 2002. Dicha ILNBD reporta un porcentaje de consultas traducidas correctamente de 90% aproximadamente.

1.1 Antecedentes

El presente proyecto tiene como finalidad mejorar el desempeño de una ILNBD desarrollada por Aguirre en el año 2014 [Aguirre, 2014]. Dicha interfaz permite consultar BDs mediante expresiones en lenguaje español no acotado.

Esta ILNBD consta de tres módulos: módulo de configuración, módulo de traducción y un administrador de diálogo. Dichos módulos pueden apreciarse en la arquitectura de la interfaz presentada en la Figura 1.1.

Al igual que en otras ILNBDs, se distinguen dos tipos de usuarios: el administrador de la BD (ABD) y el usuario final. El ABD se encarga de realizar la configuración de la ILNBD, introduciendo información relacionada con el contenido y estructura de la BD en el Diccionario de Información Semántica. El usuario final es la persona que formula consultas en LN a la ILNBD.

El elemento más importante de esta interfaz es el Diccionario de Información Semántica (DIS), el cual contiene la mayor parte de la información relevante para el proceso de traducción de consultas. Inicialmente, dicha información es introducida en el DIS por la misma interfaz por medio de un proceso de configuración automático. Posteriormente, el ABD, mediante el módulo de configuración, puede introducir información sobre términos lingüísticos usados en el dominio de la BD para mejorar el desempeño de la misma.

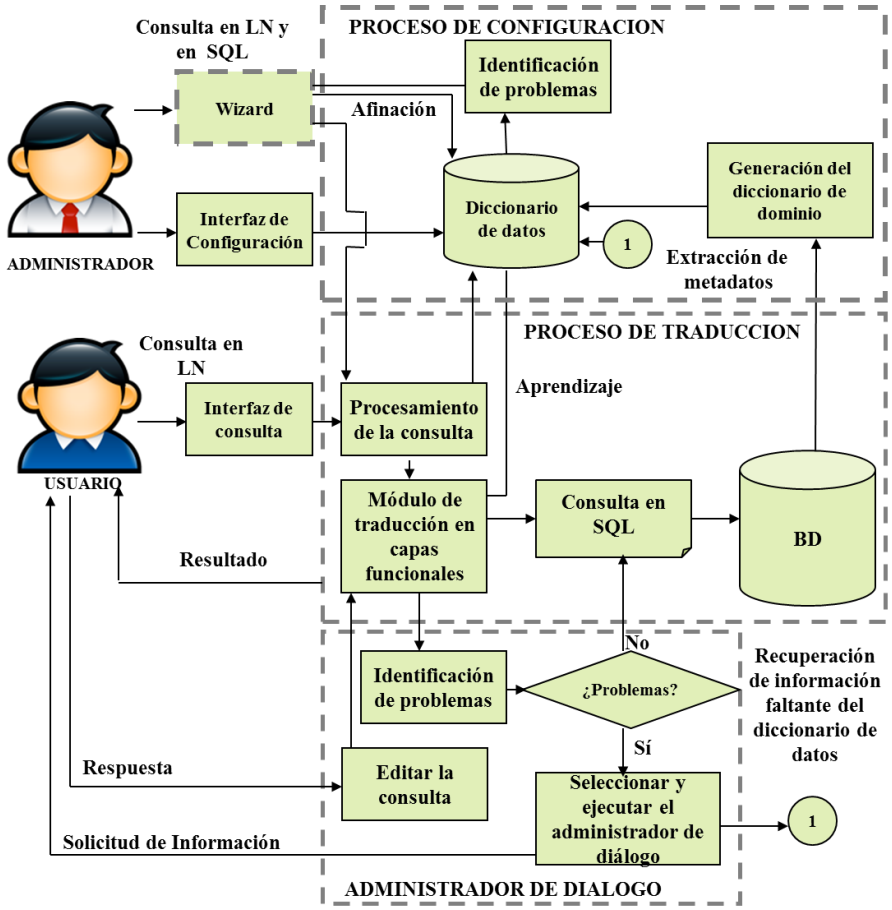


Figura 1.1. Arquitectura de la ILNBD propuesta por Aguirre

El proceso de traducción es realizado por el módulo de traducción en capas de funcionalidad (similar al modelo OSI). Dicho módulo fue diseñado para resolver la mayoría de los problemas encontrados en el proceso de traducción de lenguaje natural a SQL. A continuación, se detallan los aspectos de cada una de las capas de funcionalidad (véase Figura 1.2):

Análisis léxico. Ejecuta un proceso de etiquetado léxico obteniendo de un lexicón la categoría gramatical de cada palabra de la consulta. En esta capa, los errores léxicos deben ser

corregidos, y la ambigüedad en la categoría gramatical y problemas de homografía deben ser resueltos. El resultado obtenido es la consulta etiquetada.

Análisis sintáctico. Haciendo uso de la consulta etiquetada, se construye un árbol sintáctico de la consulta, donde los errores sintácticos son corregidos, se resuelve el problema de elipsis sintáctica y se detectan los problemas de anáfora.

Análisis semántico. A partir de la consulta etiquetada, se construye una representación del significado de la consulta, la cual puede ser usada para traducirla a SQL. Esta capa es la más compleja, debido a que la mayoría de los problemas están relacionados a la comprensión del significado de la consulta.

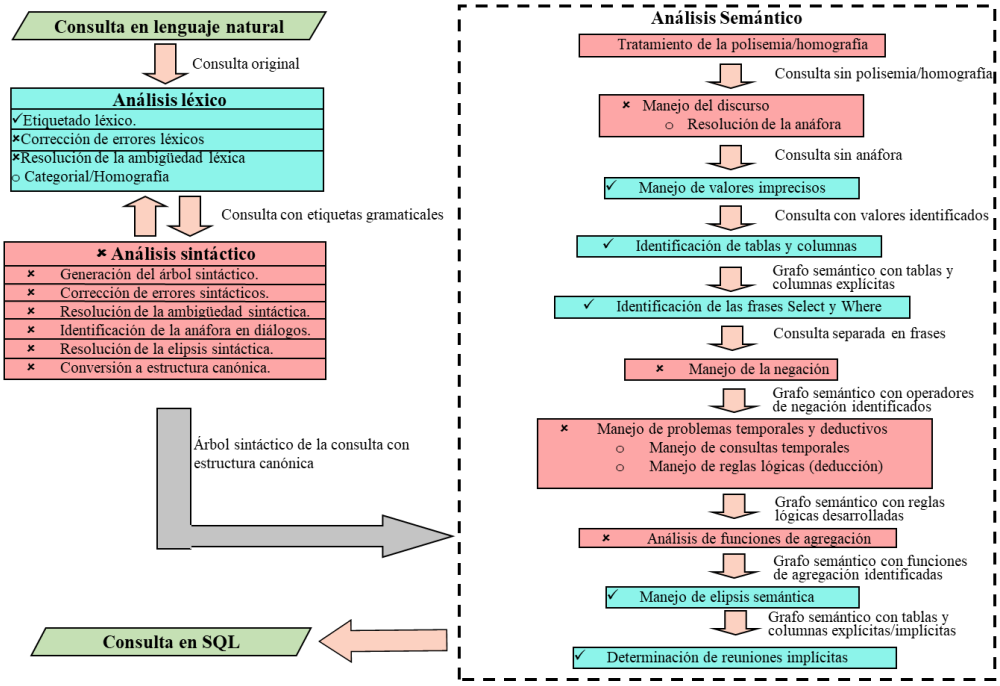


Figura 1.2. Capas de funcionalidad del módulo de traducción

1.2 Descripción del Problema

Aunque han habido muchas expectativas sobre los beneficios de las ILNBDs y, desde los años 60s, se ha desarrollado una larga serie de proyectos para implementar ILNBDs que satisfagan dichas expectativas, llama la atención el uso tan limitado que las ILNBDs tienen en la actualidad. Algunas ILNBDs comerciales ilustran esta situación, por ejemplo: LanguageAccess (desarrollada por IBM) fue descontinuado por IBM, y English Query (desarrollado por Microsoft) fue incluido por última vez en SQL Server ver. 8.0, liberado en el año 2000.

En las ILNBDs de dominio específico se han reportado porcentajes de acierto del orden de 95% en la traducción de consultas, desafortunadamente en las ILNBDs independientes de dominio, el desempeño es sustancialmente inferior. Por ejemplo, los resultados experimentales realizados

con la base de datos ATIS [Linguistic Data Consortium, 1993] (la cual posee una complejidad típica de una BD de regular tamaño que se puede encontrar en aplicaciones reales) arrojan desde un 15.7% de consultas correctamente traducidas con el software ELF hasta un 90% con la ILNBD desarrollada en el ITCM [Aguirre, 2014].

Estos resultados muestran la dificultad que existe para alcanzar un desempeño ideal, es decir, 100% de consultas traducidas correctamente. Dos de los problemas principales por los que las ILNBDs no pueden alcanzar el 100% de acierto son deficiencias en la comprensión de las consultas por parte de la ILNBD y deficiencias en los algoritmos de traducción.

Un problema que dificulta la traducción de consultas es la existencia de anomalías en el diseño del esquema de algunas BDs. Por ejemplo, se pueden encontrar bases de datos de aplicaciones reales con las siguientes anomalías: ausencia de llaves primarias y llaves foráneas en las tablas, falta de normalización del esquema de la BD, llaves primarias sustitutas, y columnas que contienen valores que se pueden calcular con funciones de agregación (COUNT, SUM, AVG, MAX, MIN).

En algunas ocasiones las BDs con anomalías de diseño son diseñadas por usuarios inexpertos que hacen uso de software amigable (p. ej., MS Access, Oracle Workbench, Navicat) para diseñar y hacer uso de las BDs relacionales. En estos casos, dichas BDs son más propensas a tener anomalías de diseño, ya que los usuarios que las crearon no tienen conocimientos formales sobre diseño y administración de BDs.

Una base de datos con anomalías de diseño da lugar a las siguientes situaciones [Rob, 2011]:

- Requiere código SQL más complejo, el cual se ejecuta más lentamente.
- La información se encuentra dispersa en varias tablas. Para realizar un cambio en un valor en una tabla se debe hacer el mismo cambio en varias tablas.
- La información es incongruente o ambigua.
- La base de datos se torna más compleja.
- La base de datos tiene información oculta, por ejemplo, una secuencia de renglones en una tabla.
- Se dificulta el análisis de los índices en la BD resultando en índices redundantes o faltantes. Lo anterior incrementa el tiempo de procesamiento e impacta el desempeño.
- El bajo desempeño incrementa los costos al reducir el ciclo de vida del hardware. Esto es debido al incremento en el uso de hardware para procesar las consultas.

El diseño de la ILNBD desarrollada en el ITCM se basa en el supuesto de que el esquema de la BD a consultar está bien diseñado. Desafortunadamente, existe un número considerable de BDs con anomalías en su diseño [Pedro-de-Jesus, 1999] que son incongruentes con el proceso de traducción de la ILNBD, lo cual ocasiona una reducción del porcentaje de consultas traducidas correctamente.

El propósito de este proyecto de tesis es mejorar el desempeño de la ILNBD descrita en [Aguirre, 2014] para BDs con anomalías en su diseño con el fin de obtener idealmente el mismo desempeño que con BDs diseñadas correctamente.

Las anomalías de diseño mencionadas en la Subsección 2.1.20 pueden relacionarse con diversos problemas que obstaculizan el buen funcionamiento del análisis semántico en las ILNBDs. Tales problemas son los siguientes:

Dificultad de identificación de tablas y columnas de la BD. En la fase de la identificación de los elementos de la consulta en SQL, la ILNBD, al intentar construir las cláusulas *Select* y *Where* de la consulta, pueden suceder dos situaciones: 1) la ILNBD no es capaz de asociar la información contenida en la consulta en LN con la estructura del esquema de la BD, 2) la ILNBD incluye en la consulta información sobre columnas que no son requeridas por el usuario. Esto se debe a que existe una anomalía de diseño en la BD. Por ejemplo, cuando existe una columna repetida en varias tablas, es posible que la ILNBD elija una o varias columnas (de las repetidas) que el usuario no solicitó.

Problemas al establecer reuniones entre tablas. Cuando en el análisis semántico se consideran dos o más tablas diferentes para la construcción de la consulta, es necesario incluir las reuniones entre dichas tablas. Las reuniones son colocadas en la cláusula Where de la consulta; sin embargo, existen casos en los que no se encuentra definida una conexión mediante llaves foráneas entre dos o más tablas.

La estructura de la BD no concuerda con el procesamiento de la ILNBD. En algunas ocasiones, una BD puede tener un diseño diferente con el cual la ILNBD no es capaz de funcionar correctamente.

Obtención de información errónea al consultar la BD. Algunas anomalías de diseño repercuten en los datos almacenados en la BD, por tal motivo, cuando la ILNBD construye una consulta en SQL correctamente, ésta obtiene datos erróneos o incongruentes.

En la Tabla 1.1 se muestra la relación que existe entre las anomalías de diseño y los problemas relacionados con el funcionamiento del análisis semántico. Cabe destacar que la falta de normalización de las tablas en una BD puede dificultar el análisis semántico de consultas de una ILNBD, debido a que la información contenida en la BD no cuenta con la estructura que pudiera requerir el procesamiento de la ILNBD. Por otro lado, se puede observar que la mayoría de las anomalías de diseño ocasiona que sea difícil localizar la información que se solicita en una consulta. Los datos mostrados en la Tabla 1.1 se confirman en el Apéndice C, donde se presentan pruebas funcionales a las ILNBDs ELF y C-Phrase.

Tabla 1.1. Anomalías de diseño y su relación con diversos problemas.

Anomalía de diseño	Identificación de tablas y columnas	Establecer reuniones entre tablas	Información errónea al consultar la BD
Ausencia de PKs y FKs		✓	
Falta de 1FN	✓		✓
Falta de 2FN	✓	✓	
Falta de 3FN	✓	✓	
Uso de llaves primarias sustitutas	✓	✓	

Columnas repetidas	✓		
Columnas con FA			✓
Malos estándares de nombramiento	✓		

Nota:
PK: Llave primaria
FK: Llave foránea

En las Subsección 2.2 se describen diversas ILNBDs; sin embargo, para ninguna de éstas se menciona que tenga la capacidad de analizar la BD en uso y determinar si ésta tiene un esquema libre de anomalías que permita el procesamiento de consultas en LN mediante el método de traducción implementado en la interfaz.

Adicionalmente, existen muy pocas herramientas que permitan realizar el análisis de una BD y encontrar anomalías en el diseño del esquema de la BD.

El modelado y diseño de bases de datos se centra en técnicas para diseño de bases de datos relacionales. Las bases de datos son objetos complejos, son una colección de datos interrelacionados. La creación de una base de datos requiere de un análisis y modelado por etapas que permiten lograr un diseño lógico [Teorey, 2011].

Para ejemplificar lo expuesto en los párrafos anteriores, se realizó una prueba con las ILNBDs ELF y C-Phrase (Tabla 1.2). Dicha prueba consistió en introducir una pequeña muestra de consultas en dichas ILNBDs. Las consultas introducidas requieren que la interfaz utilice parte del esquema de la BD. Esta parte del esquema contiene anomalías de diseño, por lo tanto, la interfaz deberá hacer un tratamiento de las anomalías para poder contestar correctamente la consulta, de lo contrario contestará de manera errónea.

Tabla 1.2. Consultas de prueba para ELF y C-Phrase

Consulta en LN	BD	Anomalía de diseño	Respuesta ELF	Respuesta C-Phrase
1. How many rivers does Colorado have?	Geobase (sin FKs)	Ausencia de llaves foráneas	×	✓
2. Give me the population of the capital of the state of Alabama	Geobase	Ausencia de llaves foráneas	×	×
3. Give me the state with the highest point	Geobase	Columnas repetidas en múltiples tablas	×	×
4. Give me the population of Illinois	Geobase	Columnas con FA	×	×
5. Name the rivers in Alabama	Geobase	Uso de llaves sustitutas	×	×

Como se puede observar en la Tabla 1.2, ELF no pudo responder correctamente ninguna de las consultas, mientras que C-Phrase respondió correctamente la consulta 1. Los resultados mostrados en la Tabla 1.2 se explican en el Apéndice C.

1.3 Objetivos

Para resolver el problema propuesto en este proyecto, se plantea el siguiente objetivo general (OG):

OG) Mejorar el desempeño de la ILNBD desarrollada en el ITCM para BDs con anomalías en el diseño de su esquema, de tal forma que su desempeño sea lo más cercano posible al que obtiene para BDs diseñadas correctamente, y además que sea superior al desempeño de ELF y C-Phrase para BDs con anomalías de diseño.

Para alcanzar el objetivo general (OG), se requiere alcanzar los siguientes objetivos específicos:

OE.1) Realizar un análisis de varias BDs con el fin de encontrar las anomalías más comunes en el diseño de BDs.
OE.2) Analizar los problemas relacionados con la presencia de anomalías de diseño en BDs.
OE.3) Diseñar algoritmos para el tratamiento de las anomalías de diseño más comunes en BDs.
OE.4) Diseñar una vista semántica con el propósito de proporcionar a la ILNBD una representación del esquema de la BD sin anomalías de diseño.
OE.5) Realizar pruebas de la nueva versión de la ILNBD para determinar su desempeño con BDs que poseen anomalías de diseño.
OE.6) Realizar pruebas funcionales con las ILNBDs ELF y C-Phrase para comprobar si éstas cuentan con un mecanismo para el tratamiento de anomalías de diseño en BDs.
OE.7) Realizar pruebas de la nueva versión de la ILNBD, ELF y C-phrase para comparar su desempeño con BDs que poseen anomalías de diseño.

1.4 Hipótesis

La hipótesis de investigación considerada para este proyecto de tesis es la siguiente:

H.1) Es posible definir para la ILNBD [Aguirre, 2014] un nivel de abstracción sobre una base de datos con anomalías de diseño, de tal manera que el traductor de consultas actual de la ILNBD perciba la base de datos exenta de anomalías. Nota: esto se debe manifestar como un desempeño de la nueva versión semejante al de la versión actual con una base de datos exenta de anomalías de diseño.

1.5 Justificación y Beneficios

Las ILNBDs son sistemas inteligentes para BDs que permiten obtener información de una BD, mediante lenguaje natural. Además, las ILNBDs tienen las siguientes ventajas [Sujatha, 2012]:

- Cualquier usuario sin conocimientos sobre un lenguaje de consulta formal puede hacer uso de ellas. Los lenguajes de consulta formal son difíciles de aprender, al menos por usuarios inexpertos (usuarios cuya área de especialización no es la informática).
- Las interfaces gráficas y basadas en formularios son más fáciles de usar por usuarios casuales; sin embargo, el uso de formularios, ventanas, selecciones restringidas por menús, etc. constituyen lenguajes de comunicación artificial que deben ser aprendidos por el usuario. Por otro lado, una ILNBD permite formular consultas en el lenguaje nativo del usuario, por lo que una ILNBD es más apropiada para un usuario casual.
- Requiere un entrenamiento mínimo para que un usuario pueda hacer uso de una ILNBD.

La mayoría de las ILNBDs con buen desempeño (alrededor de 95% de consultas traducidas correctamente) son aquéllas que son dependientes de dominio, es decir, aquéllas que sólo pueden ser usadas para consultar una BD. Como estas ILNBDs han sido implementadas para un dominio específico, su implementación/configuración está adaptada a las anomalías que tenga la BD a consultar.

Por otra parte, las ILNBDs independientes de dominio han tenido un gran número de deficiencias y limitaciones, las cuales se ven reflejadas en el desempeño de éstas. Lo anterior se debe a que dichas interfaces deben tener una estructura genérica que pueda procesar consultas en LN para cualquier BD. Además, como se ejemplifica en la Subsección 2.2 de este documento, dichas ILNBDs no tienen prevista su operación con BDs que contengan anomalías de diseño. Por lo tanto, estas ILNBDs tienden a fallar en el proceso de traducción de consultas en LN con estas BDs.

Es importante destacar que el proceso de traducción de LN a SQL de la ILNBD desarrollada en el ITCM [Aguirre, 2014] se basa en el supuesto de que la BD a consultar está bien diseñada. Desafortunadamente, existen muchas aplicaciones que usan BDs que tienen anomalías de diseño [Pivert, 2010][Pedro-de-Jesus, 1999][Howard, 2013][Mfourga, 1997]. Por lo tanto, esta ILNBD no se desempeñaría correctamente con un gran número de BDs que adolecen de este problema.

El objetivo del trabajo descrito en este documento fue brindar a la ILNBD desarrollada en el ITCM la capacidad de procesar consultas en LN formuladas a BDs con anomalías de diseño, de tal forma que el desempeño de la ILNBD sea lo más cercano posible al que obtiene para BDs diseñadas correctamente.

El beneficio específico de este trabajo consiste en que la ILNBD propuesta pueda trabajar con un mayor número de BDs, al poder procesar consultas a BDs con anomalías de diseño.

1.6 Alcance y Limitaciones

Los alcances de este proyecto de tesis se definen a continuación.

1. El módulo para tratamiento de anomalías de diseño en BDs fue implementado en la ILNBD del ITCM [Aguirre, 2014].
2. Mediante el módulo para tratamiento de anomalías de diseño en BDs, se efectúa el tratamiento para las siguientes anomalías de diseño:
    a. Ausencia de llaves primarias y foráneas.
    b. Falta de 2da forma normal.
    c. Falta de 3ra forma normal.
    d. Uso de llaves primarias sustitutas
    e. Presencia de columnas repetidas en múltiples tablas.
    f. Columnas con valores que se pueden calcular con funciones de agregación (FA).

Las limitaciones de este proyecto de tesis se definen en los siguientes puntos:

1. Las consultas en SQL construidas por la ILNBD no se transformarán a sus equivalentes optimizadas.
2. La ILNBD sólo maneja un subconjunto de las consultas de SQL de la versión ISO/IEC 9075:1989(E) (SQL 1). Entre las funciones que no están implementadas se encuentran Order by, Desc, Asc y todas las funciones que se relacionen con operaciones de inserción, borrado o actualización de información.
3. El módulo para tratamiento de anomalías de diseño en BDs fue diseñado para usarse en conjunto con la ILNBD del ITCM [Aguirre, 2014].
4. El módulo para tratamiento de anomalías de diseño no contempla su uso con BDs deductivas.
5. El módulo para tratamiento de anomalías de diseño no da tratamiento a las siguientes anomalías de diseño:
    a. Falta de 1ra. forma normal.
    b. Malos estándares de nombramiento.
6. El módulo para tratamiento de anomalías de diseño no trata anomalías de BDs que tengan un esquema en estado crítico. Es decir, no se considera el uso del módulo para BDs con un diseño extremadamente defectuoso.
7. El módulo para tratamiento de anomalías de diseño no efectúa tratamiento para los problemas relacionados con BDs sucias.

Capítulo 2

Marco Teórico y Estado del Arte

Este capítulo presenta varios términos y definiciones que son necesarios para entender los temas involucrados en esta tesis, además, contiene los temas más relevantes del estado del arte.

2.1 Marco Teórico

2.1.1 Procesamiento de Lenguaje Natural

A fin de entender mejor el tema de procesamiento de lenguaje natural, a continuación se presentan definiciones básicas sobre dos tipos de lenguajes involucrados en el tema, y posteriormente se presenta la definición de procesamiento de lenguaje natural.

Lenguaje

Sistema que asocia contenidos de pensamiento y significación a manifestaciones simbólicas tanto orales como escritas. Cuando se habla de lenguajes se pueden diferenciar dos clases muy bien definidas [Polanco, 2000]:

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

Lenguaje formal

En matemáticas, lógica y ciencias de la computación, un lenguaje formal es un lenguaje cuyos símbolos primitivos y reglas para unir esos símbolos están formalmente especificados [Mellema, 2009].

Lenguaje natural

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

- Entendimiento del lenguaje escrito que utiliza el conocimiento léxico, sintáctico y semántico del lenguaje, unido a la información o conocimiento del dominio.
- Entendimiento del lenguaje oral que comprende todo lo del campo anterior junto con toda la fonología.

Procesamiento de lenguaje natural

El procesamiento de lenguaje natural (PLN o NLP en inglés) es un conjunto de técnicas computacionales para analizar y representar naturalmente textos en uno o más niveles de análisis lingüístico con el fin de llevar a cabo el procesamiento del lenguaje como un humano para un rango de tareas y aplicaciones [Liddy, 1998].

2.1.2 Base de Datos

Una base de datos (BD) es un conjunto de datos relacionados entre sí. Los datos se refieren a hechos conocidos que pueden registrarse y que tienen un significado implícito. Una base de datos tiene las siguientes propiedades implícitas [Elmasri 1997]:

- Una base de datos representa algún aspecto del mundo real, en ocasiones llamado minimundo o universo de discurso. Las modificaciones del minimundo se reflejan en la base de datos.
- Una base de datos es un conjunto de datos lógicamente coherente, con cierto significado inherente. Una colección aleatoria de datos no puede considerarse propiamente una base de datos.

Toda base de datos se diseña, construye y puebla con datos para un propósito específico. Está dirigida a un grupo de usuarios y tiene ciertas aplicaciones preconcebidas que interesan a dichos usuarios.

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

2.1.3 Interfaz de Lenguaje Natural

El término interfaz de lenguaje natural (ILN) ha sido usado para referirse a los sistemas de interacción en lenguaje natural, donde las solicitudes del usuario son procesadas más o menos como oraciones aisladas, a menudo empleando un análisis lingüístico profundo [Mitkov, 2005]. La arquitectura general de una ILN se muestra en la Figura 2.1.

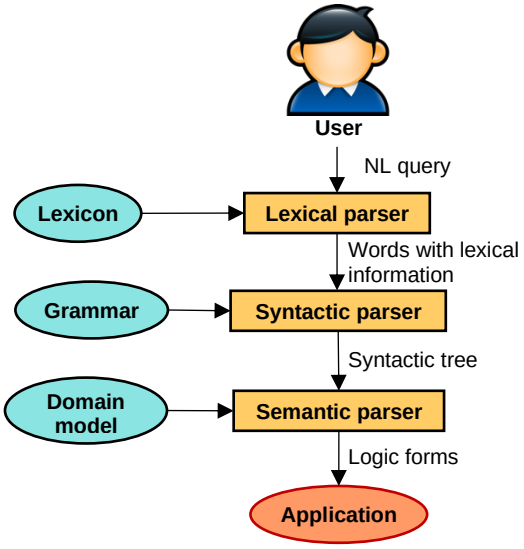


Figura 2.1. Arquitectura general de una ILN

2.1.4 Interfaz de Lenguaje Natural a Bases de Datos

Una ILNBD es un sistema que permite al usuario acceder a la información almacenada en una base de datos formulando una solicitud en lenguaje natural [Androutsopoulos, 1995].

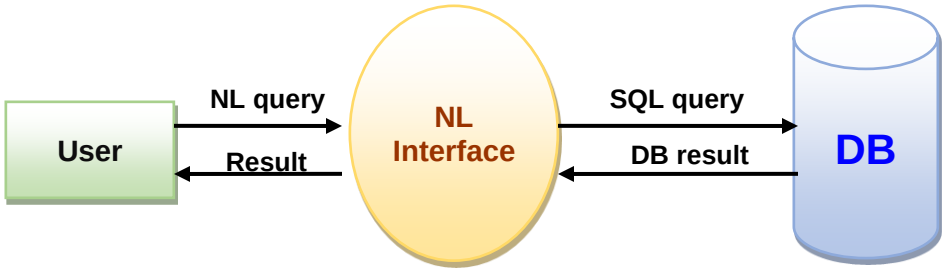


Figura 2.2. Flujo de una ILNBD

En el flujo de una ILNBD (mostrado en la Figura 2.2), generalmente el resultado obtenido es presentado de dos maneras: como instrucción en SQL o como una respuesta en lenguaje natural. En este proyecto de tesis los resultados corresponderán a los retornados por la ejecución de una instrucción en el lenguaje SQL como se observa en el siguiente ejemplo:

Consulta en lenguaje natural:
Obtener para cada libro parejas de autores.
Consulta en SQL:
SELECT *A.Clave, A.Autor, B. Autor*
FROM *Autores A, Autores B*
WHERE *A.Clave = B.Clave*

Resultado:

Clave	Autor	Autor
B0012	1. Francisco López	1. Francisco López
B0012	1. Francisco López	2. Javier Alonso
B0012	1. Francisco López	3. Marta Rebolledo
B0012	2. Javier Alonso	2. Javier Alonso
B0012	2. Javier Alonso	1. Francisco López
B0012	2. Javier Alonso	3. Marta Rebolledo
B0012	3. Marta Rebolledo	3. Marta Rebolledo

El resultado de la consulta muestra la información mediante columnas (*Clave*, *Autor*) organizada en columnas y renglones (registros encontrados).

2.1.5 Métricas de Evaluación de Desempeño para ILNBDs

Las métricas usadas para evaluar el desempeño en una ILNBD son las siguientes [Pazos, 2013]:

Precisión. Porcentaje de consultas respondidas correctamente en relación al número de consultas traducidas. Esta métrica puede ser calculada mediante la expresión (1).

Recall. Porcentaje de consultas respondidas correctamente en relación al número total de consultas introducidas en la interfaz. Esta métrica puede ser calculada mediante la expresión (2).

$$precisión = \frac{Número\ total\ de\ consultas\ correctas}{Número\ total\ de\ consultas\ traducidas} \times 100 \quad (1)$$

$$recall = \frac{Número\ total\ de\ consultas\ correctas}{Número\ total\ de\ consultas} \times 100 \quad (2)$$

2.1.6 Vista Semántica

Entre los diferentes tipos de problemas que ocurren en las consultas, existe uno que puede estar presente cuando se consultan bases de datos complejas. Este problema está relacionado con el significado semántico de una consulta. Cuando se hace referencia a una entidad, otra entidad (o entidades) se encuentra semánticamente relacionada con la primera entidad, donde el usuario generalmente ignora la relación entre dichas entidades.

El propósito de una vista semántica es integrar en una entidad virtual dos o más entidades que están relacionadas mediante llaves foráneas. Esto es con el fin de hacer explícita alguna relación o información que de otra forma sería difícil de inferir a partir de la información semántica de las entidades individuales [Aguirre, 2014].

Teniendo en cuenta el concepto de vista en SQL (véase la Subsección 2.1.9), una vista semántica es una vista en SQL a la cual se añade información semántica, a semejanza de la

definición de tablas y columnas de la BD junto con su información semántica, todo lo cual es almacenado en un diccionario de información semántica.

2.1.7 Modelo Relacional

El modelo relacional se ocupa de tres aspectos principales de la información: la estructura de datos, la integridad de los datos y la manipulación de datos [Date, 2004].

1. Aspecto estructural. El usuario percibe la información de la base de datos como tablas y nada más que tablas.
2. Aspecto de integridad. Estas tablas satisfacen ciertas restricciones de integridad.
3. Aspecto de manipulación. Los operadores disponibles para que el usuario manipule estas tablas (por ejemplo, para fines de recuperación de datos), operadores que derivan tablas a partir de tablas. En particular, tres de estos operadores son importantes: restringir, proyectar y reunir (este último operador también es conocido como combinar).

El modelo relacional consta de los siguientes cinco componentes [Date, 2004]:

1. Un conjunto abierto de tipos escalares (incluyendo en particular el tipo lógico o valor verdadero).
2. Un generador de tipos de relación y una interpretación propuesta de dichos tipos.
3. Herramientas para definir variables de relación de dichos tipos de relación generados.
4. Un operador de asignación relacional para asignar valores de relación a las variables de relación mencionadas.
5. Un conjunto abierto de operadores relacionales genéricos para derivar valores de relación a partir de otros valores de relación.

2.1.8 Relación

Dados los conjuntos S_1, S_2, …, S_n (no necesariamente distintos), R es una relación de n conjuntos si es un conjunto de n-tuplas, cada una de las cuales tiene su primer elemento a partir de S_1, su segundo elemento de S_2 y así sucesivamente [Codd, 1970]. R tiene grado n. Las relaciones de grado 1 son llamadas unarias, de grado 2 binarias, de grado 3 ternarias y de grado n n-arias [Codd, 1970].

Una relación n-aria R se implementa en una base de datos como una tabla. A continuación se describe la correspondencia entre los conceptos del modelo relacional y la terminología de bases de datos, así como algunas propiedades de las tablas.

1. Una relación R se representa en una base de datos como una tabla.
2. Para cada n-tupla de R debe existir un renglón en la tabla.
3. El ordenamiento de los renglones no importa.
4. Todos los renglones son distintos.
5. El ordenamiento de las columnas de la tabla es importante y corresponde al ordenamiento S_1, S_2, …, S_n de los dominios en los cuales R está definida.

6. El significado de cada columna es parcialmente portado al etiquetar la columna con el nombre del dominio correspondiente.
7. Se le denomina cabecera al conjunto de dominios S_1, S_2, …, S_n de *R*, así como al conjunto correspondiente de columnas de una tabla

En la Figura 2.3 se muestra un ejemplo de una relación con *k* tuplas.

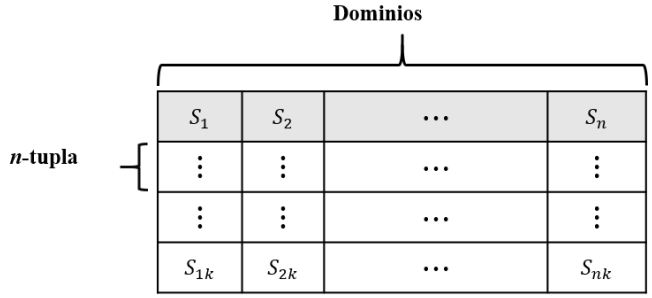


Figura 2.3. Ejemplo de una relación con *k* tuplas

2.1.9 Vista

Una vista en una relación derivada que es virtual, no real. El valor de una vista dada en un tiempo determinado es el resultado de evaluar una expresión relacional al momento en cuestión. Nótese que la expresión que define la vista debe mencionar al menos una relación, de otra manera no podrá existir la vista [Date, 2008].

En SQL es posible crear vistas mediante la siguiente expresión:

CREATE VIEW *viewname* **AS** SELECT *query*

CREATE VIEW es una instrucción que almacena la especificación de una consulta. La subexpresión SELECT *query* se usa para generar la tabla virtual (relación virtual), donde *query* representa una consulta en SQL.

Por ejemplo, en la primera instrucción de la Figura 2.4 se muestra la sintaxis para crear una vista llamada *PRICEGT50*. Esta vista contiene tres columnas (*P_DESCRIPT*, *P_QOH* y *P_PRICE*) e incluye sólo renglones en los cuales el precio es mayor que $50. La segunda instrucción en SQL permite mostrar los renglones que satisfacen la cláusula Where de la vista [Rob, 2011]. Después de esta instrucción, la figura muestra el resultado de su ejecución.

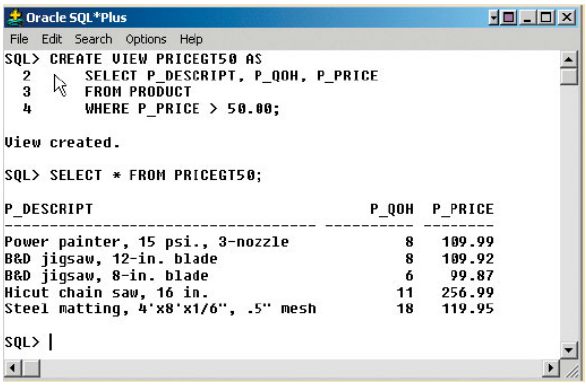


Figura 2.4. Creación de una vista

2.1.10 Llave Candidata

Es un identificador único. Más precisamente, sea *K* un subconjunto de la cabecera de una relación *R*; entonces *K* es una llave candidata de *R*, si y sólo si se satisfacen las siguientes condiciones:

a) No existe un valor posible de *K* tal que *R* contenga dos tuplas distintas con el mismo valor de *K*.
b) Lo mismo no puede ser dicho para cualquier subconjunto propio de *K*.

Nótese que toda relación tiene al menos una llave. Nótese también que, por definición, las llaves son conjuntos de atributos (y los valores llaves son, por lo tanto, tuplas); sin embargo, si el conjunto de atributos que constituye una llave *K* contiene sólo un atributo *A*, entonces informalmente hablando *A* por sí misma es una llave [Date, 2008].

2.1.11 Llave Primaria

Una llave primaria es una llave candidata que se ha seleccionado como llave principal de una relación. Considérese un ejemplo de una base de datos, la cual incluye relaciones correspondientes a partes, proyectos y proveedores. Una relación llamada *parte* se define de acuerdo con los siguientes dominios:

1. Número de parte.
2. Nombre de la parte.
3. Color de la parte.
4. Peso de la parte.
5. Cantidad en existencia.
6. Cantidad en demanda.

En el ejemplo, *Número de parte* sería una llave primaria. Una llave primaria es no redundante, si es de sólo un dominio (no una combinación) o una combinación tal que ninguno de

los dominios participantes sea superfluo en identificar únicamente cada elemento. Una relación puede contener más de una llave primaria no redundante.

2.1.12 Llave Foránea

Sea R_2 una relación. Entonces una llave foránea (*FK*) en R_2 se define como un conjunto de columnas de R_2 tal que se satisfagan las siguientes condiciones:

a. Existe una relación R_1 (R_1 y R_2 no son necesariamente distintas) con una llave candidata *CK*.
b. Es posible renombrar algún subconjunto de las columnas de *FK*, de tal forma que *FK* se convierte en *FK'*, y *FK'* y *CK* sean del mismo tipo, es decir, que haya una correspondencia de la *i*-ésima columna de *FK'* con la *i*-ésima columna de *CK*.
c. Cada valor de *FK* en su valor actual de R_2 tiene un valor para *FK'* que es idéntico al valor de *CK* en algún renglón del valor actual de R_1.

Para comprender mejor este concepto, a continuación se define el concepto de integridad referencial: ninguna tupla referente puede existir, si la tupla referenciada correspondiente no existe. Más precisamente, sea *FK* una llave foránea en una relación R_2, sea *K* la llave primaria correspondiente en la relación correspondiente R_1, y sea *K'* derivada de *K* como se explica en la definición de llave foránea. Entonces la regla de integridad referencial requiere que no exista ocasión alguna en la que exista un valor de *FK* en R_2 que no sea el valor de *K'* para alguna tupla (necesariamente única) en R_1 [Date, 2008].

Almacén			
Proveedor	**Parte**	**Proyecto**	**Cantidad**
1	2	5	17
1	3	5	23
2	3	7	9
2	7	5	4
4	1	1	12

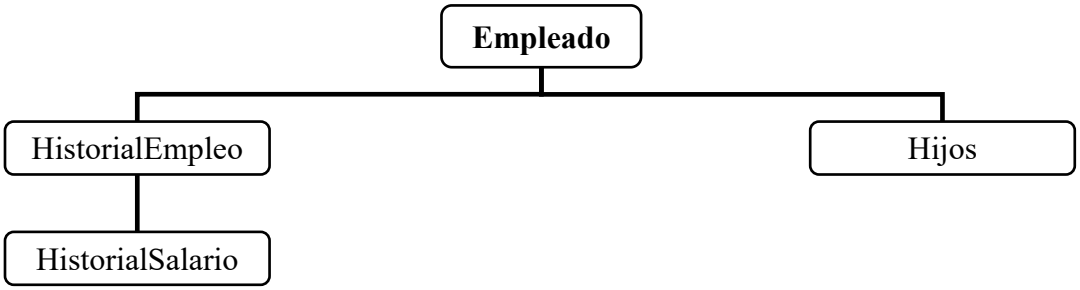
Figura 2.5. Ejemplo de una relación

En la Figura 2.5, la combinación de los dominios *Proveedor*, *Parte* y *Proyecto* forman una llave primaria, mientras que cada uno de esos dominios tomados por separado son llaves foráneas.

2.1.13 Primera Forma Normal

Una relación se encuentra en primera forma normal (1FN), si la relación no incluye dominios no simples, es decir, dominios que son compuestos de dos dominios o más [Codd, 1970].

Considérese, por ejemplo, una colección de relaciones mostrada en la Figura 2.6(a). *HistorialEmpleo* e *Hijos* son dominios no simples de la relación *Empleado*; además, *HistorialSalario* es un dominio no simple de la relación *HistorialEmpleo*. El árbol de la Figura 2.6(a) muestra las interrelaciones de los dominios no simples.



Empleado (NumEmpleado, Nombre, FechaNacimiento, HistorialEmpleo, Hijos)
HistorialEmpleo (FechaEmpleo, Titulo, HistorialSalario)
HistorialSalario (FechaSalario, Salario)
Hijo (NombreHijo, AñoNacimiento)

Figura 2.6(a). Conjunto sin normalizar

Empleado (numEmpleado, nombre, fechaNacimiento)
HistorialEmpleo (numEmpleado, fechaEmpleo, titulo)
HistorialSalario (numEmpleado, fechaEmpleo, fechaSalario, salario)
Hijo (numEmpleado, nombreHijo, añoNacimiento)

Figura 2.6(b). Conjunto normalizado

El resultado de normalizar la colección de relaciones de la Figura 2.6(a) es la colección de relaciones de la Figura 2.6(b).

2.1.14 Dependencia Funcional

Sean *X* y *Y* subconjuntos de la cabecera de una relación *R*; entonces la dependencia funcional (DF) $X \rightarrow Y$ se mantiene en *R*, si y sólo si dos tuplas cualesquiera concuerdan en *X* y también concuerdan en *Y*. *X* y *Y* son el determinante y el dependiente respectivamente, y la DF puede ser leída como "*X* determina funcionalmente a *Y*" o "*Y* es funcionalmente dependiente de *X*" [Date, 2012].

Por ejemplo, la DF $\{CITY\} \rightarrow \{STATUS\}$ se mantiene en la relación *S* mostrada en la Figura 2.7.

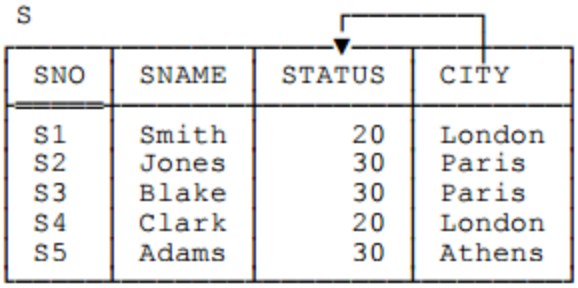


Figura 2.7. Dependencia funcional en la relación *S*

2.1.15 Segunda Forma Normal

Una relación se encuentra en la segunda forma normal (2FN), si se encuentra en primera forma normal, y todos los atributos que no son llaves son dependientes de todos los atributos de la llave. De acuerdo con esta definición, si la relación tiene un solo atributo como su llave, entonces automáticamente está en la segunda forma normal [Kroenke, 2003].

Considérese la relación *Actividad* en la Figura 2.8. Si eliminamos o borramos la tupla para *EID* 175, perderemos el hecho de que Squash cuesta $50. Además, no podemos registrar una actividad hasta que un estudiante se inscriba en ella. El problema con esta relación es que tiene una dependencia que involucra sólo una parte de la llave que es la combinación (*EID*, *Actividad*); pero la relación contiene una dependencia, *Actividad–Cuota*. En este caso, *Cuota* es parcialmente dependiente de la llave. Para eliminar la anomalía debemos separar la relación en dos.

EID	Actividad	Cuota
100	Esquí	200
100	Golf	65
150	Natación	50
175	Squash	50
175	Natación	50
200	Natación	50
200	Golf	65

Figura 2.8. Relación Actividad

2.1.16 Tercera Forma Normal

Los atributos no deben tener dependencia transitiva con otros atributos que no sean la llave primaria, y la tabla debe estar en la primera y segunda forma normal [Date, 2004]. Un ejemplo simple de la violación de la tercera forma normal (3FN) es una tabla de ganadores de un torneo, donde se tienen cuatro columnas: *Torneo*, *Año*, *Ganador*, *Cumpleaños de ganador*. Al tener como llave candidata a {*Torneo*, *Año*}, la violación se presenta al tener una dependencia de la columna *Cumpleaños de ganador* con la llave candidata sólo a través de la columna *Ganador*, la cual no es una llave candidata.

2.1.17 Lenguaje de Consultas Estructurado

SQL (Structured Query Language, por sus siglas en inglés) 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, el cual 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:2016. 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, 2006].

2.1.18 Lenguaje de Definición de Datos

Cuando se implementa una base de datos relacional, primero se debe definir la estructura de la misma con un SMBD. Para tal efecto, los programadores usan un lenguaje de definición de datos (DDL, por sus siglas en inglés).

Haciendo uso del DDL se especifican los nombres de las tablas en la base de datos, se nombran y describen las columnas de esas tablas, se definen los índices y se describen otras estructuras tales como restricciones y restricciones de seguridad [Kroenke, 2003].

En SQL se hace uso de una colección de verbos imperativos cuyo fin es modificar el esquema de la BD añadiendo, cambiando o borrando tablas. Las instrucciones del DDL pueden ser usadas junto con otras instrucciones en SQL.

Para ejemplificar el uso del DLL considérese la siguiente expresión en SQL:

```
CREATE TABLE airport (
   airport_code      VARCHAR(50) PRIMARY KEY,
   airport_name      VARCHAR(50) NOT NULL,
   location          VARCHAR(50) NOT NULL,
   state_code        VARCHAR(50) NOT NULL,
   time_zone_code    VARCHAR(50) NOT NULL
)
```

La expresión mencionada hace uso de la instrucción Create perteneciente al DDL. Dicha instrucción permite crear una tabla con cinco columnas dentro del esquema de la BD; además, se puede hacer uso de las instrucciones Drop, Alter y Rename para destruir un objeto, modificarlo o renombrarlo respectivamente.

2.1.19 Lenguaje de Manipulación de Datos

El lenguaje de manipulación de datos (DML, por sus siglas en inglés) permite la manipulación o procesamiento de los objetos definidos mediante el DDL [Date, 2004].

SQL permite hacer uso del DML para obtener y manipular información en una base de datos relacional. Las instrucciones en SQL que se pueden usar son las siguientes: Select, Insert into, Update y Delete from.

La instrucción usada principalmente en este proyecto para realizar las consultas a una BD es la instrucción Select, la cual sirve para obtener un conjunto de resultados de una o más tablas. Considérese el siguiente ejemplo para ilustrar la sintaxis de dicha instrucción:

SELECT $T_i.C_j$ FROM T_i WHERE $T_i.C_k$ = <valor>

Donde $T_i.C_j$ y $T_i.C_k$ son columnas de la base de datos, en la cláusula From T_i es una tabla perteneciente a la base de datos, a partir de la cual se obtendrá información. Por último, en la cláusula Where se especifican condiciones para restringir los resultados. Dicha condición se especifica asociando un valor a una columna $T_i.C_k$ de la BD.

2.1.20 Anomalías de Diseño en BDs

El diseño adecuado de una base de datos es crucial para el buen funcionamiento de la misma, es decir, el desarrollo, aplicación y desempeño posterior presentarán muy pocos problemas. Algunas de las anomalías que ocurren en el diseño de bases de datos son las siguientes:

Ausencia de llaves primarias y foráneas. La razón más importante para tener llaves primarias y foráneas es la identificación de renglones únicos en cada tabla de la base de datos. Una función igual de importante para las llaves primarias y foráneas es la conexión entre diversas tablas de la base de datos. La omisión de las llaves primarias dificulta mantener los datos organizados y su consulta de manera precisa y simple. Por otra parte, la omisión de las llaves foráneas repercute directamente en la conexión entre tablas y su proceso de consulta mediante una ILNBD (ver Apéndice C).

Falta de la primera forma normal. Una tabla no se encuentra en 1FN, si al menos una de sus columnas no es atómica. Una columna atómica es aquélla que no puede ser dividida. Tal columna se dice que muestra atomicidad [Rob, 2011]. Por ejemplo, en la Figura 2.9, la columna *statestringlist* en la tabla *river* no es atómica, porque la tabla *river* puede ser dividida en dos tablas: *river* y *river_state*. Con esto se gana flexibilidad al consultar la BD. En general los diseñadores prefieren usar columnas con valores simples.

river

name	length	statestringlist
St. Francis	684	Missouri, Arkansas
Tombigbee	658	Mississipi, Alabama
Washita	805	Texas, Oklahoma
Wateree Catawba	636	North Carolina, South Carolina

river

name	Length
St. Francis	684
Tombigbee	658
Washita	805
Wateree Catawba	636

river_state

river_name	state_name
St. Francis	Missouri
St. Francis	Arkansas
Tombigbee	Mississipi
Tombigbee	Alabama
Washita	Texas
Washita	Oklahoma
Wateree Carawba	North Carolina
Wateree Carawba	South Carolina

state

name	abbreviation	…
Missouri	mo	…
Mississipi	ms	…
Oklahoma	ok	…
Texas	tx	…

Figura 2.9. Descomposición de una columna no atómica

Falta de la segunda forma normal. Una tabla no se encuentra en segunda forma normal, si al menos una de las columnas de ésta posee una dependencia funcional con una columna que no sea la llave primaria.

Falta de la tercera forma normal. Una tabla no se encuentra en tercera forma normal, si al menos una columna tiene dependencia transitiva con otra columna que no es la llave primaria [Date, 2004]. En la Figura 2.10 se presenta la dependencia funcional {*CITY*}→{*STATUS*}. Dado que {*CITY*} depende funcionalmente de {*STATUS*}, y ésta no es una llave candidata, la tabla *S* no está en 3NF (aunque sí se encuentra en 2NF).

Llave primaria

Dependencia funcional {CITY}→{STATUS}

S

S#	SNAME	STATUS	CITY
S1	Smith	20	London
S2	Jones	10	Paris
S3	Blake	30	Paris
S4	Clark	20	London
S5	Adams	30	Athens

Figura 2.10. Ejemplo de una tabla sin 3FN

Uso de llaves primarias sustitutas. Una llave primaria sustituta es una columna simple que consta de la siguiente propiedad: sus valores sirven solamente como sustitutos para las entidades que representan, es decir, sirven para representar el hecho de que las entidades correspondientes existen y no conllevan información adicional alguna [Date, 2008]. El uso de llaves primarias sustitutas puede crear redundancia de datos en una tabla tal como se muestra en la Figura 2.11.

Llave primaria sustituta

Id	Bin	Wine	Producer	Year	Bottles	Ready
1	2	Chardonnay	Buena Vista	2001	1	2003
2	3	Chardonnay	Geyser Peak	2001	5	2003
3	6	Joh. Riesling	Jekel	2002	1	2003
4	12	Fumé Blanc	Ch. St. Jean	2001	4	2003
5	12	Fumé Blanc	Ch. St. Jean	2001	4	2003

Tuplas repetidas

Figura 2.11. Ejemplo de redundancia de datos al usar llaves primarias sustitutas

Presencia de columnas repetidas en múltiples tablas. Este problema está relacionado con la redundancia de información, ya que estas columnas no son llaves foráneas ni llaves primarias en las tablas donde se encuentran localizadas. Un ejemplo de este tipo de columnas se encuentra en la base de datos Geobase, donde la columna *state_name* aparece repetida en múltiples tablas. En el ámbito de las ILNBDs, este problema ocasiona que una ILNBD al trabajar con una BD con dicho problema no obtenga la información requerida por el usuario, ya que existen numerosas ocurrencias de la misma columna en diferentes tablas, haciendo que la ILNBD relacione la consulta con una columna errónea. Un ejemplo de este tipo de anomalía se muestra en la Figura 2.12.

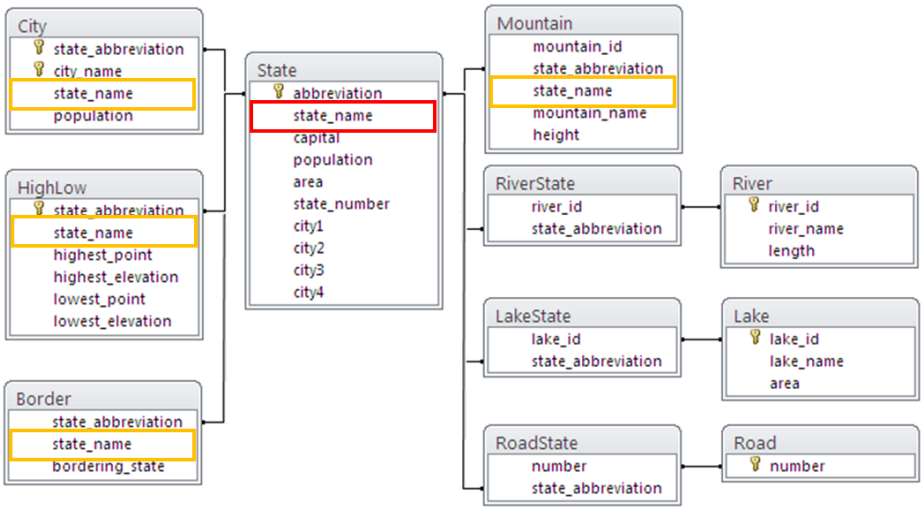


Figura 2.12. Esquema de BD con columnas repetidas en múltiples tablas

Columnas con valores que se pueden calcular con funciones de agregación. La presencia de columnas con valores calculados anula la finalidad de las funciones de agregación. Las columnas cuyos valores puedan ser obtenidos mediante la aplicación de una función de agregación podrían ser innecesarias e incorrectas. Lo anterior se debe a que los valores calculados de dichas columnas solamente podrían ser correctos para BDs estáticas, ya que en este tipo de BDs no se realizan inserciones de nuevos renglones además de los que ya existen. Por otra parte, en una BD dinámica se tendrían que actualizar los valores de la columna con valores calculados por funciones de agregación cada vez que se inserte un renglón nuevo.

Malos estándares de nombramiento. La congruencia en los descriptores es uno de los puntos más importantes. El correcto nombramiento de los descriptores, tablas, columnas y objetos permite no sólo a los usuarios, sino a otros programadores y desarrolladores comprender y manipular la base de datos de manera más eficiente. Es necesario evitar descriptores confusos o poco comunes que sólo el diseñador de la base de datos puede comprender. En algunas ILNBDs (como ELF) la eficacia del proceso de traducción depende de que los nombres de las tablas y columnas describan adecuadamente el significado de éstas.

2.1.21 Bases de Datos Sucias

Una base de datos sucia es aquélla que contiene información sucia, es decir, información incompleta, inválida o incorrecta [Chu, 2004].

Por ejemplo, en algunas ocasiones, los usuarios de las aplicaciones de BDs pueden cometer errores al introducir información en la BD causando información errónea e incongruente en la BD. Además, si la configuración de las columnas de la BD lo permite, la información contenida en éstas puede tener campos vacíos, es decir, información faltante en los renglones.

La tabla *cliente* en la Figura 2.13 es un ejemplo de una tabla en una BD sucia.

name	mktsegmt	nation	address
Mary	building	USA	Jones Ave
Mary	banking	USA	Jones Ave
Marion	banking	USA	Jones ave
John	building	America	Arrow
John S.	building	USA	Arrow
John	banking	Canada	Baldwin

Figura 2.13. Tabla *cliente*

2.2 Estado del Arte

Varias ILNBDs han sido desarrolladas desde la década de los 70s tales como LUNAR [Woods, 1972], RENDEZVOUS [Codd, 1974], LADDER [Hendrix, 1978], entre otras. Estas interfaces, a pesar de haber sido diseñadas para una BD particular, también podían ser configuradas para trabajar con otras BDs; sin embargo, dicha tarea era muy difícil debido a las limitaciones técnicas de ese entonces.

En [Pazos, 2018] se consideran algunas ILNBDs que son relevantes para este trabajo. Dichas ILNBDs son relevantes por dos razones: Primero, existe un prototipo o interfaz comercial de la ILNBD con la cual se pueden hacer pruebas y así indagar si cuentan con algún mecanismo para dar tratamiento a las anomalías de diseño. Segundo, son ILNBDs desarrolladas recientemente.

Cabe mencionar que también existen herramientas de software para diseño y análisis de BDs; algunos ejemplos de éstas son Visual Paradigm [Visual-Paradigm, 2020], Vertabelo [Vertabelo, 2020], DbSchema [Wise Coders, 2020], Toad Data Modeler [Quest, 2020] entre otros.

Las herramientas mencionadas, además de permitir a los usuarios diseñar fácilmente BDs a partir de diagramas entidad-relación, también cuentan con herramientas de análisis sobre la estructura de la BD diseñada. Sin embargo, la mayoría de las herramientas de análisis y corrección de errores con las que cuentan estos softwares son para corregir errores relacionados con BDs sucias.

Además, las anomalías de diseño consideradas en este trabajo requieren de información semántica que únicamente el ABD puede proporcionar. Desafortunadamente, ninguno de los softwares mencionados considera este tipo de problema.

A pesar de que ningún trabajo mencionado hasta ahora ha abordado el problema de las anomalías de diseño en BDs, se incluyen en este análisis del estado del arte algunas interfaces con las cuales se pueden hacer pruebas tales como C-PHRASE [Minock, 2010] y ELF [Conlon, 2004], y otras que han sido desarrolladas recientemente tales como NADAQ [Xu, 2019], Cross-domain NLI [Wang, 2019], DBPal [Utama, 2018], nQuery [Sukthankar, 2017], NALI [Mvumbi, 2016], Query Builder Interface Using Dependency Parsing [Kokare, 2015], Aneesah [O'Shea, 2015] y NaLIR [Li, 2014]. En las primeras se verifica el desempeño que tienen con BDs con anomalías de diseño.

2.2.1 ELF (2004)

ELF es una interfaz comercial diseñada para BDs relacionales por la compañía Elsoft [ELF, 2009]. Antes de comenzar a utilizar la interfaz, ésta debe realizar una configuración automática de acuerdo con la base de datos usada.

En la configuración automática, ELF obtiene información sobre el esquema de la BD y la almacena en su diccionario de datos. De esta forma construye un índice con el contenido de los renglones de la BD. Dicho índice permite a esta ILNBD buscar palabras referentes a valores de búsqueda que aparezcan en las consultas en LN. De esta forma identifica los valores de búsqueda, así como las columnas de la BD relacionadas con los valores identificados.

En las pruebas presentadas en [Conlon, 2004], la interfaz fue configurada por un experto, y los usuarios eran profesionales en computación. El porcentaje de éxito (*recall*) fue de 70-80%.

Actualmente, se cuenta con una versión comercial de la ILNBD ELF, por lo cual fue posible realizar pruebas con dicha interfaz. Dichas pruebas tienen como finalidad demostrar que la ILNBD ELF no es capaz de procesar consultas a BDs con anomalías de diseño (véase Apéndice C).

2.2.2 C-Phrase (2010)

C-Phrase [Minock, 2010] es una ILNBD basada en tecnología de web para procesar consultas en lenguaje inglés. Dicha interfaz trabaja con bases de datos relacionales. Las consultas procesadas por esta interfaz son representadas como expresiones de cálculo de tuplas.

Para realizar el procesamiento de sintagmas nominales, la interfaz representa los sintagmas nominales (NP) como premodificadores (PRE) y postmodificadores (POST). Algunas reglas generales para la configuración de C-Phrase se muestran a continuación.

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 define un patrón de enunciado del cual existen muchas variantes. Dicha regla permite a los usuarios escribir consultas del tipo "Lista los X", "dame el X", donde X es un sintagma nominal. El sistema cuenta con alrededor de 75 patrones de enunciado. El proceso de creación de la consulta en SQL provee elementos léxicos que definen premodificadores, cabeceras y postmodificadores de sintagmas nominales para una base de datos. Las cuatro reglas restantes permiten procesar sintagmas nominales con premodificadores y postmodificadores.

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

2.2.3 NaLIR (2014)

NaLIR [Li, 2014] es una interfaz de consulta en lenguaje natural interactiva que trabaja con bases de datos relacionales. Dicha interfaz puede interpretar consultas complejas en lenguaje natural de manera genérica y es independiente de dominio. Las consultas que puede contestar esta interfaz pueden incluir funciones de agregación, consultas anidadas y varios tipos de reuniones.

Esta ILNBD consta de tres componentes principales. El primer componente transforma una consulta en LN a un árbol de consulta. El segundo componente verifica la transformación interactivamente con el usuario. Para tal efecto, la interfaz muestra el árbol de consulta al usuario para que éste verifique, si la información contenida en el árbol es la información que le interesa conocer de la BD. Por último, un tercer componente traduce el árbol de consulta a una consulta en SQL.

Para realizar el proceso de traducción de una consulta en LN, la interfaz utiliza un analizador de dependencias, el cual usa un analizador sintáctico desarrollado en la Universidad de Stanford [Marneffe, 2006] para generar un árbol sintáctico a partir de la consulta en lenguaje natural. Después, la interfaz mapea los nodos del árbol sintáctico que se pueden usar como

componentes en SQL y los agrupa por categorías. En algunos casos, los nodos mapeados se pueden clasificar en diferentes categorías, por lo cual puede darse ambigüedad al momento de interpretar estos nodos. Por cada uno de estos nodos, la interfaz obtiene el mejor mapeo y reporta los mapeos candidatos al usuario.

Una vez que el usuario verifica el árbol de consulta, el traductor utiliza su estructura para generar la estructura apropiada en la expresión en SQL y completa las reuniones (*joins*) correspondientes. La instrucción en SQL resultante puede contener funciones de agregación, subconsultas multinivel y varios tipos de reuniones, entre otras cosas. Finalmente, la interfaz evalúa la instrucción en SQL usando el manejador de BDs y muestra el resultado al usuario.

2.2.4 Aneesah (2015)

Aneesah [O'Shea, 2015] es una ILNBD con habilidades conversacionales, la cual provee un ambiente interactivo y amigable para facilitar a los usuarios la obtención de información almacenada en una base de datos relacional. Dicha interfaz es independiente de dominio, ya que se puede usar con cualquier BD relacional.

La arquitectura de Aneesah se desarrolló adoptando el enfoque de coincidencia de patrones. Esta arquitectura consiste de tres componentes principales, los cuales se detallan a continuación:

Componente 1. Comprende un gestor de conversación, interfaz de usuario, memoria temporal y componentes del agente conversacional. Este último permite a la interfaz conversar con el usuario para guiarlo en el proceso de obtención de información.

Componente 2. Consiste de una base de conocimientos que sirve como cerebro para la interfaz, la cual contiene información para que la interfaz funcione con cualquier BD (se requiere una configuración previa por parte del usuario).

Componente 3. El tercer componente comprende un motor de SQL. Éste, a partir de la información/sintaxis recibida del componente 1, formula la consulta en SQL basándose en las decisiones del usuario.

2.2.5 Query Builder Based on Dependency Parsing (2015)

El objetivo principal de esta ILNBD es proporcionar comunicación entre un usuario y una computadora sin hacer uso de ninguna sintaxis de consulta a BDs [Kokare, 2015].

Análisis de constituyentes y análisis de dependencias son dos técnicas ampliamente usadas en procesamiento de lenguaje natural. En el análisis de constituyentes, el árbol sintáctico descompone un enunciado en subfrases no terminales. En el árbol sintáctico existen tres tipos de frases, las terminales son las palabras del enunciado, y las aristas en el análisis de constituyentes no están etiquetadas.

Por otro lado, el análisis de dependencias conecta palabras de acuerdo a sus relaciones. Cada vértice en el árbol representa una palabra, los nodos hijos son palabras que son dependientes del padre, y las aristas están etiquetadas por la relación.

Sin embargo, el análisis de constituyentes necesita mucho tiempo para analizar las palabras del enunciado. Por lo tanto, es necesario desarrollar una NLQBI que extraiga *Parts of Speech* (POS, unidades léxicas en español) y procese la consulta en menos tiempo. El analizador de la ILNBD se desarrolló usando el análisis de dependencias, donde las consultas que ya han sido dadas como entrada al sistema no se procesarán de nuevo. La ILNBD guarda todas las consultas y su instrucción en SQL correspondiente que fue previamente generada.

A continuación se explica el funcionamiento de la ILNBD. Primero, se ejecuta un análisis morfológico, la consulta introducida es revisada en busca de palabras clave y signos de puntuación, los cuales son removidos. Después, las unidades léxicas (*tokens*) son separadas.

El siguiente paso es un análisis sintáctico en el cual la consulta es convertida en una estructura en forma de árbol usando análisis de dependencias. Los sustantivos, adjetivos, etc. se relacionan en forma de enlaces asimétricos binarios. Estos enlaces forman una estructura de dependencias, generando el árbol del análisis de dependencias. Este árbol de análisis se verifica para encontrar los significados en la fase del análisis semántico. Entonces, se convierte a una consulta lógica. La consulta se mapea con las tablas y columnas de la BD usando las unidades léxicas más significativas extraídas en la primera fase. Después de esto, la consulta se traduce a una consulta en SQL reemplazando las unidades léxicas con nombres de tablas o nombres de columnas, la cual se envía a la BD para proporcionar al usuario el resultado exacto.

Esta ILNBD reporta una exactitud de 91.66%; sin embargo, no se menciona el corpus de prueba usado.

2.2.6 NALI (2016)

Esta ILNBD propone métodos para simplificar su afinación sin sacrificar cobertura o exactitud [Mvumbi, 2016]. Para tal efecto, utiliza dos ambientes de autoría para reducir el trabajo requerido para usar la ILNBD con varios dominios.

El primer ambiente de autoría se llama descendente, el cual supone la existencia de un corpus de consultas de ejemplo en LN sin etiquetar usadas para obtener términos léxicos para simplificar la configuración de la ILNBD. Este enfoque reduce el esfuerzo al afinar la ILNBD al incluir automáticamente la información semántica para los verbos en forma negativa, los adjetivos comparativos y superlativos en el modelo de configuración.

El segundo ambiente de autoría se llama ascendente, el cual explora la posibilidad de construir un modelo de configuración sin afinación manual usando la información del esquema de BD y un diccionario.

Esta ILNBD utiliza SQL como lenguaje de consulta formal y el lenguaje inglés como lenguaje de entrada. Cabe mencionar que las consultas en LN de entrada que recibe esta ILNBD deben estar libres de errores gramaticales y ortográficos. El sistema de análisis está basado en un enfoque simbólico para consultas en LN.

Durante el proceso de consulta, la ILNBD utiliza un etiquetador POS y un analizador sintáctico desarrollados en la universidad de Stanford. Por lo tanto, la arquitectura de esta ILNBD está basada en sintaxis y utiliza el analizador de dependencias de Stanford.

La traducción de las consultas de LN a SQL se realiza en cuatro pasos: análisis léxico, análisis sintáctico, análisis semántico y traducción a SQL

Esta ILNBD fue probada con Geoquery y obtuvo 77.4% de precisión y 74.5% de *recall*.

2.2.7 nQuery (2017)

nQuery [Sukthankar, 2017] es una ILNBD independiente de dominio que se basa en un enfoque que incorpora solicitudes complejas en LN (operaciones de manipulación de información y preguntas) con solicitudes simples. La interfaz permite procesar consultas que involucran funciones de agregación, condiciones múltiples en la cláusula Where y cláusulas como Order by, Group by y Having. Este sistema fue desarrollado para el sistema manejador de BD MySQL.

nQuery traduce consultas en LN a consultas en SQL antes de obtener la información de la BD. Este sistema se enfoca en obtener la información, pero también permite la traducción de otras operaciones de manipulación de información (Insert, Delete y Update). Sin embargo, la interfaz traduce consultas que pueden ser procesadas por MySQL para reducir la complejidad de las consultas a la BD.

2.2.8 DB-Pal (2018)

DBPal [Utama, 2018] aprovecha los avances en modelos *deep learning* para hacer más robusta la comprensión de consultas de las siguientes maneras: Primero, DBPal usa un modelo *deep* para traducir instrucciones en lenguaje natural a SQL, haciendo el proceso de traducción más robusto para parafraseo y otras variaciones lingüísticas. Segundo, DBPal proporciona un modelo de autocompletamiento aprendido que sugiere extensiones parciales de consultas a los usuarios durante la formulación de consultas, ayudándolos así a escribir consultas complejas.

DBPal tiene dos características principales basadas en modelos de redes neuronales:

1. Robust Query Translation. Propone un método de traducción basado en un modelo de red neuronal recurrente de secuencia a secuencia. La robustez del modelo de traducción radica en mapear las expresiones que varían lingüísticamente a operaciones de bases de datos relacionales finitas predefinidas. Un punto clave en este tipo de ILNBDs es diseñar un conjunto de entrenamiento para el modelo. En este trabajo se implementó un método

de generación sintético que utiliza únicamente el esquema de BD con anotaciones mínimas como entrada y genera una gran colección de pares de consultas en LN y sus instrucciones en SQL correspondientes. Para generar un conjunto de entrenamiento, primero se ejecuta el paso de generación de datos (llamado Generator), donde se usa el esquema de BD junto con un conjunto de plantillas base que describen los pares NL-SQL y los diccionarios *slot-filling* para generar un conjunto de entrenamiento de 1 a 2 millones de pares. En el segundo paso, se aumenta automáticamente el conjunto inicial de pares NL-SQL aprovechando los modelos de lenguaje existentes para variar automáticamente la parte en LN de cada par usando diferentes variaciones lingüísticas (llamado *Augmentation*).

2. Interactive Auto-Completion. DBPal proporciona una herramienta de autocompletamiento en tiempo real y sugerencias de consultas para ayudar a los usuarios que no están familiarizados con el esquema de la BD.

Los autores de DBPal realizaron pruebas utilizando el corpus de Geoquery que ya ha sido usado para evaluar otras ILNBDs; además, para probar las variantes lingüísticas, diseñaron otro corpus llamado Patients, el cual es una BD de estudios médicos que cuenta con una tabla y consiste de 290 consultas en LN y SQL. Asimismo hicieron comparaciones con NaLIR y NSP.

Los resultados obtenidos por DBPal con los corpus mencionados son 75.93% de exactitud para Patients y 48.9% de exactitud para Geo.

2.2.9 Cross-domain NLI (2019)

Es una ILN que aplica una estrategia de etiquetado de consultas de propósito general y un modelo de traducción neuronal multilenguaje. Con el etiquetado de consulta, cada dominio se maneja de igual forma con un modelo de traducción neuronal multilenguaje [Wang, 2019].

Propone un modelo donde tipos de consultas diferentes y dominios diferentes comparten los mismos componentes. Para tal efecto, la ILNBD ejecuta un preprocesamiento, el cual consiste en separar la información de dominio específico de la consulta.

Dado un par (pregunta, consulta SQL), la metodología principal consiste en insertar símbolos prediseñados para agrupar elementos mencionados en la consulta en LN para manejar cada muestra de manera uniforme.

El procesamiento realizado por la ILNBD es el siguiente:

1. Cuenta con un clasificador binario BC que detecta elementos para predecir, si un elemento *e* aparece en una pregunta *q* que corresponde a una consulta en SQL *p* por el significado semántico de la pregunta, la cual toma *q* y *e* como entradas sin hacer referencia a *p*.

2. La ILNBD busca la frase con más influencia en la consulta usando métodos llamados texto adversarial basado en gradiente.

3. Se insertan símbolos en *q* para agrupar las frases que describen los elementos, denotados como *q'*.

4. Se construye un modelo seq2seq para traducir *q'* a *p'*, y *p'* es una consulta donde los elementos son reemplazados por símbolos insertados en *q*.

5. Los símbolos insertados son reemplazados con elementos para formar la consulta original (es decir, convertir *p'* a *p*).

Esta interfaz utiliza un codificador basado en una red neuronal recurrente bidireccional multicapa (RNN). Utiliza un símbolo prefijo (p. ej. <*SQL*>) para manejar diferentes tipos de consultas y para que cada tipo de consulta sea tratada uniformemente.

Esta ILNBD reporta experimentos utilizando los corpus WikiSQL, OVERNIGHT y Geo880; las métricas de evaluación fueron exactitud de coincidencia de consulta Acc_{qm}. También se comparan los resultados de ejecución, denotados como Acc_{ex}. Para WikiSQL se obtuvo 74.5% de Acc_{qm} y 82.7% de Acc_{ex}, para OVERNIGHT 76.8% de Acc_{qm} y para Geo880 84.1% de Acc_{qm}.

2.2.10 NADAQ (2019)

NADAQ es una ILNBD que combina *deep learning* y técnicas de análisis de SQL [Xu, 2019]. Para tal fin, el sistema añade neuronas y dimensiones de compresión de esquemas a la fase de decodificación. Estas neuronas son controladas por un autómata que supervise los estados gramaticales del decodificador. Además, la ILNBD incluye una técnica que permite a la red neuronal rechazar las consultas que son irrelevantes para el dominio de la BD y sugiere consultas candidatas en LN.

NADAQ está constituido en base a tres módulos, los cuales se describen a continuación:

1. Almacenamiento de información. Este módulo incluye MySQL como sistema manejador de BD. Extrae metadatos de las tablas para entrenar el modelo de traducción, mismo que sirve para procesar instrucciones en SQL para presentar los resultados a los usuarios.
2. Gestor de modelos. Este módulo constituye el núcleo de la ILNBD, el cual utiliza varios modelos para traducción bidireccional entre LN y SQL y también utiliza modelos para rechazar consultas irrelevantes. El módulo provee información al módulo de Interfaz de usuario.
3. Interfaz de usuario. Este módulo consta de las interfaces para interacción con el usuario.

NADAQ fue evaluada con tres BDs: Microsoft Academic Search (MAS), IMDb y Geobase. Las pruebas involucraron la comparación de tres métodos: máquina de red neuronal convolucional, traducción computacional de secuencia a secuencia basada en atención y modelo de análisis

semántico con retroalimentación. La ILNBD obtuvo una puntuación F1 de 83.9% para Geobase y 80% para IMDb.

2.2.11 Conclusión

Es importante mencionar que en este análisis se omitieron las ILNBDs de dominio específico, ya que éstas están diseñadas para funcionar con las anomalías de diseño que tiene la BD para la que fueron diseñadas.

Tabla 2.1. ILNBDs del estado del arte

ILNBD	Año	Tratamiento de Anomalías de Diseño
NADAQ	2019	No
Cross-domain NLI	2019	No
DB-Pal	2018	No
nQuery	2017	No
NALI	2016	No
Query builder	2015	No
Aneesah	2015	No
NaLIR	2014	No
C-Phrase	2010	Parcial
ELF	2004	Parcial

Tanto las ILNBDs desarrolladas más recientemente como las ILNBDs comerciales no han considerado las anomalías de diseño en BDs como un problema que deba ser solucionado para poder traducir correctamente consultas en LN a consultas en SQL. Debido a esto, no se puede asegurar que el funcionamiento de la mayoría de las ILNBDs listadas sea correcto al utilizar BDs con anomalías de diseño.

Capítulo 3

Anomalías de Diseño de Solución Simple

3.1 Ausencia de Llaves Primarias y Foráneas

Esta anomalía afecta el desempeño de la ILNBD, ya que, antes de poder ser usada por el usuario final, ésta debe ser configurada. Lo anterior se realiza mediante una configuración inicial que se ejecuta automáticamente al momento de seleccionar una BD que no ha sido usada anteriormente en la ILNBD.

Al momento de realizar la configuración automática, se registra en el DIS la información sobre las tablas, columnas y relaciones de la BD. Al momento de registrar las relaciones existentes entre tablas, la ILNBD verifica las llaves foráneas definidas en las tablas; si no hay llaves foráneas definidas en las tablas, entonces la ILNBD es incapaz de registrar las relaciones entre éstas. Las relaciones registradas en el proceso descrito son usadas en el proceso de traducción de la consulta en LN a una consulta en SQL.

Para solventar este problema, se creó un módulo para configurar el DIS de la ILNBD para definir las relaciones faltantes en el esquema de BD (Figura 3.1).

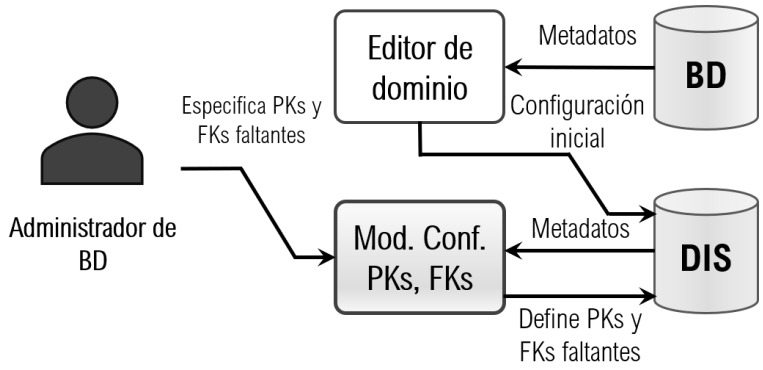


Figura 3.1. Flujo del módulo para falta de PKs y FKs

En dicho módulo, el ABD puede definir las llaves primarias faltantes en el DIS. Para tal fin, primero debe identificar las columnas a las que les hagan falta llaves primarias. Posteriormente, mediante el Módulo de configuración, el administrador puede indicar una por una, las columnas que desea que sean llaves primarias (Figura 3.2), donde PK simboliza llave primaria y FK, llave foránea.

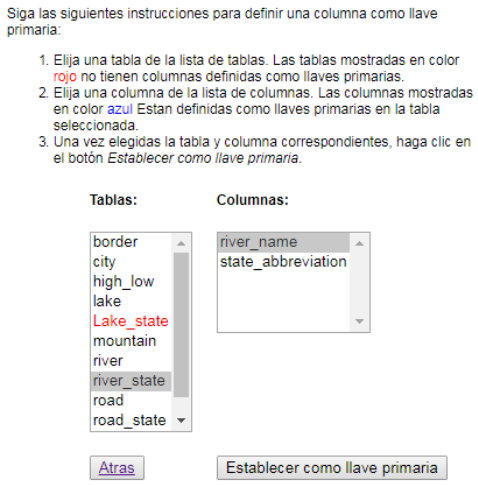


Figura 3.2. Módulo para definir llaves primarias

El finalizar este proceso, en el DIS quedarán registradas como llaves primarias las columnas indicadas por el administrador de BD. Cabe mencionar que es posible definir llaves primarias compuestas por dos o más columnas en el DIS mediante este método.

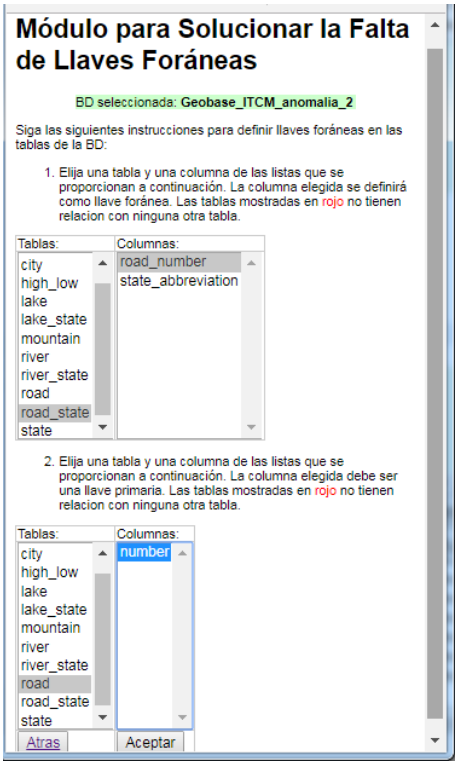


Figura 3.3. Módulo para definir llaves foráneas

Asimismo, para solucionar la falta de llaves foráneas, el administrador de BD debe identificar la columna que desea declarar como llave foránea (columna referente) y la columna con la cual desea enlazar la llave foránea (columna referida), donde ésta a su vez es una llave primaria. Posteriormente, mediante la Interfaz de Configuración (Figura 3.3), el administrador puede indicar dichas columnas, y automáticamente la ILNBD definirá la relación entre dichas columnas en la

tabla *relaciones* del DIS. Una vez terminado este proceso, la ILNBD podrá usar la relación definida en el proceso de traducción de consultas.

3.2 Uso de Llaves Primarias Sustitutas

El uso de llaves primarias sustitutas afecta el desempeño de la ILNBD, ya que esta anomalía puede causar redundancia de datos y falta de relaciones entre tablas. La redundancia de datos que puede ocurrir con esta anomalía afecta en el resultado obtenido por la ILNBD, ya que algunos de los renglones retornados por la ILNBD en estos casos pueden ser confusos para el usuario final; mientras que la falta de relaciones afecta a la definición de reuniones en la consulta en SQL y, por lo tanto, al resultado final.

Para solventar este problema, se desarrolló un módulo de configuración para el DIS [Sidorov, 2020] que permite al ABD indicar la columna (o combinación de columnas) que constituya una posible llave primaria de una tabla con el fin de verificar que ésta no tenga dos o más renglones con los mismos valores en la llave primaria. Si la tabla no posee una columna o combinación de columnas que satisfagan esta condición, entonces el problema no puede ser resuelto, ya que la base de datos no satisface una condición esencial de una base de datos correctamente diseñada: para toda tabla debe poderse definir una llave primaria con una o varias de sus columnas (excluyendo la llave sustituta). En este caso, la base de datos tendría que ser reparada para satisfacer esta condición.

Considérese una tabla T_P que tiene una llave primaria sustituta K_S; para resolver este problema, es necesario que T_P cuente con una llave candidata. En tales circunstancias, el problema se resuelve utilizando la Interfaz de Configuración del DIS para cambiar la definición de K_S de llave primaria a llave sustituta y definir la nueva llave primaria K_P de T_P.

Este problema se complica cuando existe otra tabla T_R que tiene una relación de integridad referencial con T_P. En este problema se presentan los siguientes casos:

1. Que en el esquema de la BD se encuentre definida una llave foránea de T_R a T_P mediante la llave sustituta K_S y que exista una columna (o combinación de columnas) C_{IFK} que constituya una llave foránea implícita que haga referencia a la llave primaria natural K_P de T_P.

2. Que en el esquema de la BD se encuentre definida una llave foránea de T_R a T_P mediante la llave sustituta K_S y que no exista una columna (o combinación de columnas) C_{IFK} que constituya una llave foránea implícita que haga referencia a la llave primaria natural K_P.

3. Que en el esquema de la BD no se encuentre definida una llave foránea de T_R a T_P mediante la llave sustituta K_S y que exista una columna (o combinación de columnas) C_{IFK} que constituya una llave foránea implícita que haga referencia a la llave primaria natural K_P.

Para cada uno de los casos anteriores, primero es necesario redefinir la llave primaria para T_P tal como se describió anteriormente. Después, para el primer y tercer casos, es necesario utilizar

el módulo de configuración para indicar la columna (o combinación de columnas) C_{IFK} que constituya una posible llave foránea de la tabla T_R con el fin de verificar que se satisfaga la restricción referencial entre la tabla T_R y la tabla T_P. Si la tabla T_R no posee una columna o combinación de columnas que satisfagan esta condición, entonces el problema no puede ser resuelto, y la base de datos tendría que ser reparada para satisfacerla.

El primer caso se puede resolver usando el módulo de configuración del DIS para definir que C_{IFK} sea una llave foránea implícita de T_R, lo cual puede servir para algún análisis semántico que pudiera necesitarse para otros propósitos. Sin embargo, para el análisis semántico actual (para generación de reuniones y subconsultas) podría usarse la llave foránea definida en el esquema de la BD.

El segundo caso puede resolverse como se mencionó anteriormente, al reparar la BD para que ésta tenga una columna o combinación de columnas C_{IFK} que satisfaga una restricción referencial entre T_R y T_P. Una vez reparada la BD, se debe implementar un mecanismo para verificar la integridad referencial existente entre C_{IFK} y la llave primaria natural K_P de T_P. De igual forma que en el primer caso, para el análisis semántico actual podría usarse la llave foránea definida en el esquema de la BD.

El último caso puede resolverse usando el módulo de configuración del DIS para definir que C_{IFK} sea una llave foránea de T_R. A diferencia del primer caso, en esta ocasión se podría usar C_{IFK} en el análisis semántico para generar reuniones y subconsultas.

El proceso para solventar el problema relacionado con esta anomalía se realiza en dos etapas. Primero, se define la llave primaria sustituta en el DIS (Figura 3.4).

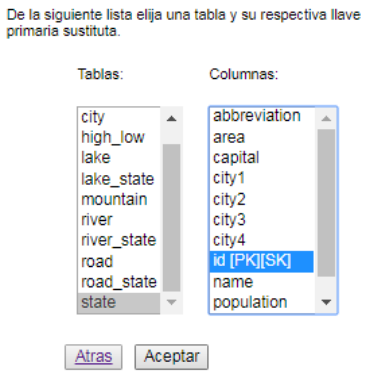


Figura 3.4. Definición de una llave primaria sustituta en el DIS

Posteriormente, en caso de que la tabla que contiene la llave primaria sustituta pueda tener una relación mediante otras columnas que no sean la llave primaria sustituta (llave primaria natural), se define dicha relación seleccionando la columna referida y después la columna referente (Figura 3.5). Cabe mencionar que, si la llave primaria natural está compuesta de dos o más columnas, es posible definir una relación de este tipo al efectuar el proceso descrito por cada columna que conforme la llave primaria natural.

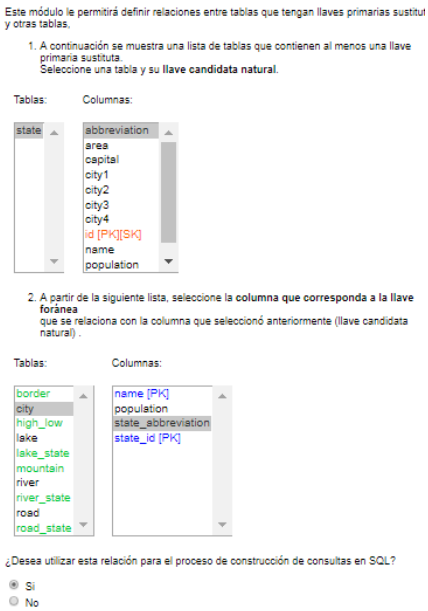


Figura 3.5. Definición de una relación

Por último, el administrador de BD puede decidir si desea utilizar la relación definida para que sea usada por la ILNBD para realizar el proceso de traducción.

Cabe mencionar que con la inclusión de este proceso, la ILNBD ahora puede considerar el uso de llaves primarias compuestas, es decir, llaves primarias compuestas de dos o más columnas. Esta situación no estaba contemplada en el diseño de la ILNBD anterior.

Al configurar el DIS mediante este módulo de configuración, la ILNBD es capaz de utilizar en el proceso de traducción las relaciones definidas mediante llaves primarias naturales e ignorar las relaciones definidas mediante las llaves primarias sustitutas.

Por ejemplo, considere la siguiente consulta en LN:

¿En cuál estado se encuentra el lago Ontario?
In which state is Lake Ontario?

Las tablas involucradas en la consulta son *State*, *LakeState* y *Lake*. En la Figura 3.6 se puede observar el fragmento del esquema de la BD con la anomalía de diseño, donde *Lake.lake_id* es una llave primaria sustituta, *LakeState.lake_id* es una llave foránea que conecta con la llave primaria sustituta, y podría existir una relación natural entre *Lake.lake_name* y *LakeState.lake_name*; sin embargo, debido al uso de la llave primaria sustituta, no se definió dicha relación en el esquema de BD.

Mediante el módulo de configuración se define la llave primaria sustituta *Lake.lake_id* y se define la llave foránea (*LakeState.lake_name* REFERENCES *Lake.lake_name*). Al momento de definir dicha llave foránea, la ILNBD marca la llave foránea (*LakeState.lake_id* REFERENCES *Lake.lake_id*) como nula para que ésta sea ignorada en el proceso de traducción de las consultas.

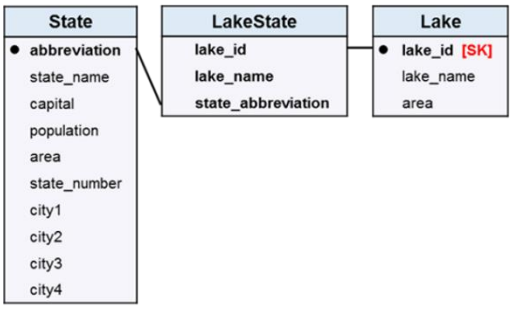


Figura 3.6. Fragmento de la BD Geobase con una llave primaria sustituta

Al procesar la consulta mediante la ILNBD sin dar tratamiento a la anomalía de diseño, se obtendría la siguiente consulta en SQL:

```
1: SELECT State.state_name
2: FROM State, LakeState, Lake
3: WHERE Lake.lake_name LIKE 'Ontario'
4: AND State.abbreviation=LakeState.state_abbreviation
5: AND LakeState.lake_id = Lake.lake_id;
```

mientras que la consulta obtenida al dar tratamiento a la anomalía de diseño mediante la interfaz de configuración es la siguiente:

```
1: SELECT State.state_name
2: FROM State, LakeState, Lake
3: WHERE Lake.lake_name LIKE 'Ontario'
4: AND State.abbreviation = LakeState.state_abbreviation
5: AND LakeState.lake_name = Lake.lake_name;
```

La primera consulta hace uso de las reuniones y la llave primaria sustituta, mientras que la segunda consulta utiliza una reunión natural mediante los nombres de los lagos.

3.3 Columnas Cuyos Valores se Pueden Calcular con FAs

Las columnas cuyos valores se pueden calcular con FA ocasionan que la ILNBD pueda obtener resultados erróneos, ya que los valores obtenidos también pueden ser calculados mediante una FA aplicada a otra columna de otra tabla.

Por tal motivo, se desarrolló un módulo de configuración que permite al ABD definir en el DIS las columnas que se pueden calcular con FA, y asignarles una FA asociada a una columna para que ésta pueda ser usada en lugar de la columna con anomalía.

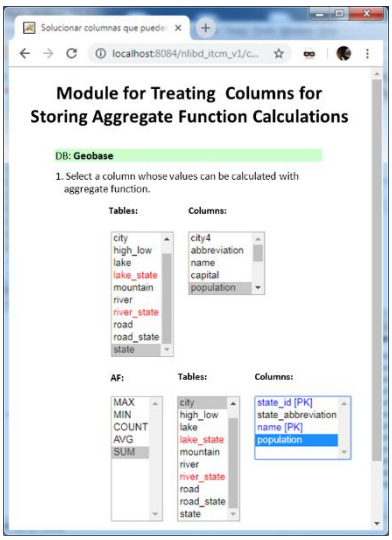


Figura 3.7. Módulo para tratar columnas cuyos valores se pueden calcular con FA

El ABD puede utilizar el módulo mostrado en la Figura 3.7, el cual muestra una lista de todas las tablas de la BD y sus columnas. A partir de esta lista se debe seleccionar la columna que tenga la anomalía. Posteriormente, se debe indicar la FA y la columna a la cual se le va a aplicar la FA.

Asimismo se desarrolló un algoritmo que modifica el funcionamiento interno de la ILNBD de tal forma que ésta pueda utilizar las FAs asociadas a este tipo de columnas.

El Algoritmo 3.1 muestra el pseudocódigo que describe el funcionamiento del tratamiento de columnas que pueden ser calculadas con FA. En el pseudocódigo, *Q* es la consulta introducida por el usuario, Q_i es un *token* de la consulta *Q*, *n* es el número total de *tokens* en *Q*, y *FAColVal* es una variable usada para almacenar la etiqueta final que tendrá el *token* que hace referencia a la columna con la anomalía. En la línea 3, por cada *token* de la consulta *Q*, se verifica si la columna a la cual hace referencia el *token* Q_i es una columna que se puede calcular con FA, de ser así, en la línea 4 se almacena una expresión en *FAColVal* a partir de la FA almacenada en el DIS para dicha columna y la columna a la cual se le aplicará la FA. Por último, se actualiza la etiqueta final del *token* Q_i con la información de *FAColVal*.

Algoritmo 3.1. Pseudocódigo para procesar columnas con FA

```
1:  procedure AFColumnsProcess(Q,n)
2:      for i=0 to n-1 do
3:          if isAFColumn(Qi) then
4:            AFColVal = getAF(Qi) + "("+getAFColumn(Qi)+")"
5:              setFinalTag(Qi, AFColVal)
6:          endif
7:      endfor
```

Para ejemplificar el funcionamiento del Algoritmo 1, considere la siguiente consulta en LN:

¿Cuál es la población del estado de Mississippi?
What is the population of the state of Mississippi

La población de un estado podría calcularse de dos maneras: mediante la columna *State.population* o sumando las poblaciones de las ciudades del estado. El ABD debe decidir si desea utilizar la columna con la anomalía o en su lugar utilizar una FA.

Al procesar dicha consulta sin dar tratamiento a la anomalía de diseño, la ILNBD construiría una consulta en SQL con la columna *State.population* en la cláusula Select como sigue:

```
1: SELECT State.population
2: FROM State
3: WHERE State.name LIKE 'Mississippi'
```

Por otra parte, al definir en el DIS la columna *State.population* como una columna que se puede calcular con FA y asignar en su lugar la FA SUM(*City.population*), la ILNBD detectará que hay dos tablas involucradas en la consulta: *City* en la cláusula Select y *State* en la cláusula Where. Por lo tanto, la ILNBD generará la siguiente consulta en SQL con sus respectivas reuniones:

```
1: SELECT SUM(City.population)
2: FROM State, City
3: WHERE State.name LIKE 'Mississippi' AND
4: State.abbreviation = City.state_abbreviation
```

Es importante mencionar que el ABD puede elegir entre utilizar la columna con la anomalía de diseño o utilizar la FA para que la ILNBD la utilice en el procesamiento de consultas.

3.4 Columnas Repetidas en Múltiples Tablas

Esta anomalía afecta al análisis semántico de la ILNBD, ya que pueden existir columnas repetidas que no son llaves foráneas, las cuales pueden contener información inconsistente y, además, hacen referencia a otra columna que contiene la misma información (columna con la información más confiable).

La solución a este problema es que la ILNBD, en lugar de utilizar este tipo de columnas, utilice la columna que contiene la información más confiable. Para tal efecto, se implementó un módulo en el Editor de Dominio que permite configurar el DIS para realizar lo mencionado (Figura 3.8).

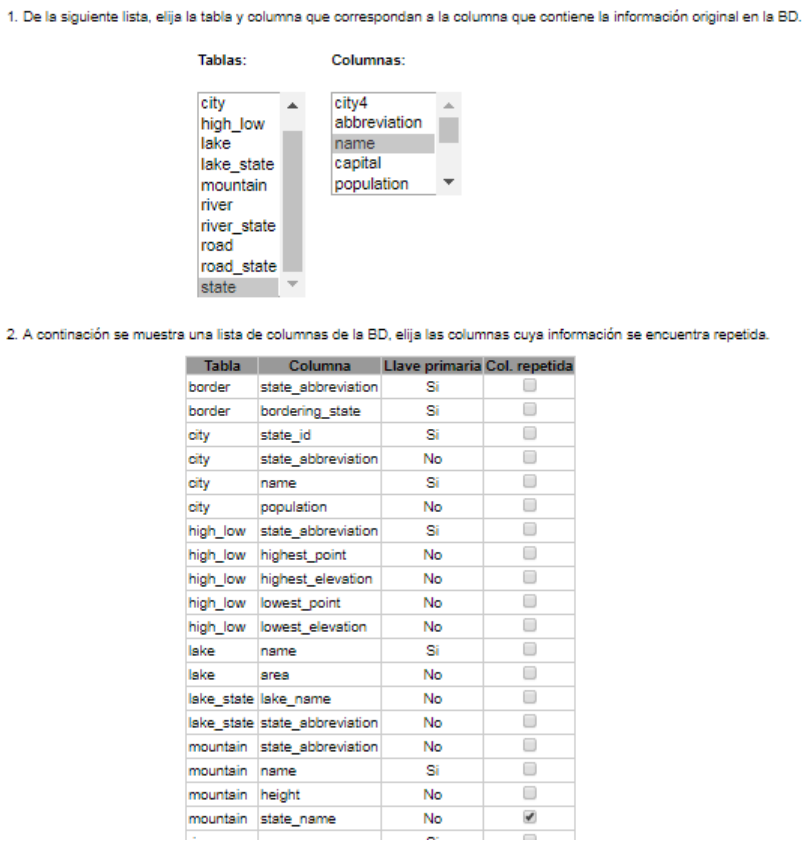


Figura 3.8. Módulo para tratamiento de columnas repetidas

La presencia de columnas repetidas en múltiples tablas es una anomalía que crea conflicto en el procesamiento de la ILNBD al momento de elegir una columna de la BD que tiene varias ocurrencias en diferentes tablas.

El DIS de la ILNBD contiene la información necesaria para asociar las palabras presentes en una consulta en LN con las columnas de una BD. Sin embargo, cuando en una BD existen varias tablas que tienen columnas repetidas, la ILNBD no es capaz de saber cuál es la columna que contiene la información a la que el usuario se refiere. La mayoría de las veces, las columnas que adolecen de este problema pueden contener información incorrecta debido a errores de inserción. Por lo tanto, cuando la ILNBD haga referencia a dicha información, ésta estará equivocada.

Para solventar los problemas relacionados con esta anomalía, el administrador de BD debe identificar la columna que contiene la información correcta (usualmente una columna que corresponde a un atributo de una entidad fuerte). Después, el administrador de BD debe indicar las columnas que están repetidas mediante el módulo de configuración implementado (Figura 3.8). Una vez hecho lo anterior, el DIS contará con la información necesaria para realizar el procesamiento de consultas que involucran este tipo de columnas.

Aunado a lo anterior, se desarrolló un algoritmo para permitir a la ILNBD identificar estas columnas y así poder procesar correctamente las consultas (Algoritmo 3.2).

Algoritmo 3.2. Pseudocódigo para procesar columnas repetidas

```
1:    procedure TreatmentOfRepColumns(Q,n)
2:      for i=0 to n-1 do
3:        if isColumn(Qi) then
4:          repCols = getRepeatedCols()
5:          repColsTags = getRepeatedColsTags()
6:          for j=0 to sizeOf(repCols)
7:            if getColumnTag(Qi) == repColsj
8:              setFinalTag(Qi, repColsTagsj)
9:            endif
10:         endfor
11:       endif
12:     endfor
```

A continuación se explica el funcionamiento del Algoritmo 2. En la línea 3, por cada unidad léxica de la consulta *Q*, se identifican las que hacen referencia a una columna. Posteriormente, en las líneas 4 y 5 se obtienen las columnas que se encuentran repetidas en la BD con sus respectivas etiquetas (columnas con información más confiable). Esta información de las columnas se obtiene del DIS. En la línea 7 se realiza una verificación para identificar la unidad léxica que haga referencia a alguna columna repetida; en caso de identificar alguna, en la línea 8 se etiqueta la unidad léxica con la columna correcta.

Para ejemplificar el funcionamiento del pseudocódigo mencionado, considere la siguiente consulta:

¿Qué montañas están en el estado de Alaska?
Which mountains are in the state of Alaska?

La consulta anterior, al ser procesada por la ILNBD, detecta para la cláusula Select la columna *Mountain.mountain_name* y en la cláusula Where la columna *Mountain.state_name* con su valor de búsqueda *Alaska*. La columna *state_name* se encuentra en 5 tablas de la BD (*City*, *State*, *Mountain*, *HighLow*, *Border*). Sin embargo, la columna que tiene la información más confiable es *State.state_name*, las demás columnas contienen información repetida y poco confiable. Al ejecutar el Algoritmo 2, ILNBD evita el uso de la columna *Mountain.state_name*, ya que no contiene información confiable, y en su lugar utiliza la columna *State.state_name*. Una vez hecho esto, la ILNBD procede a identificar las reuniones implícitas entre tablas y, por último, obtiene la siguiente consulta en SQL:

```
1:  SELECT Mountain.mountain_name
2:  FROM State, Mountain
3:  WHERE State.state_name LIKE 'Alaska' AND
4:  State.abbreviation = Mountain.state_abbreviation
```

3.5 Evaluación Experimental

En esta sección se describen pruebas experimentales realizadas a la interfaz de configuración para tratamiento de anomalías de diseño en la ILNBD, así como sus resultados.

El objetivo principal de este experimento fue probar que la configuración efectuada en el DIS al usar el módulo de configuración permite que la ILNBD responda correctamente consultas en LN con BDs con anomalías de diseño. Asimismo se evaluó la usabilidad de la interfaz de configuración. Los recursos empleados para realizar las pruebas se describen en la Tabla 3.1.

Tabla 3.1. Recursos empleados para la experimentación

Concepto	Descripción
Usuarios	11 alumnos de maestría en ciencias en computación.
Servidor de web	1 computadora Dell Latitude E6320. 12GB RAM, Procesador Intel i5 2.8ghz, 256 GB SSD.
Equipos para usuarios	11 computadoras con navegador de web y soporte para cookies habilitado.
Documentos y papelería	Manual de usuario para interfaz de configuración. Esquemas de las BDs ATIS y Geobase. Documento con casos de prueba para anomalías.

Con el fin de que los usuarios estuvieran familiarizados con la operación de la interfaz de configuración y con las BDs que se usarían en las pruebas (para simular que cada usuario fuera un administrador de BD), se entregó un manual y los esquemas de las BDs ATIS y Geobase a cada usuario tres días antes del día de la prueba.

Al realizar las pruebas se entregó a cada uno de los usuarios un documento que describe las anomalías de diseño para las que deberían modificar el DIS mediante la Interfaz de Configuración para Anomalías. Dicho documento tiene las siguientes características:

- Contiene 20 casos de prueba separados en grupos de 5 (1 grupo por cada tipo de anomalía).
- En cada grupo de 5 casos, 3 casos pertenecen a la BD Geobase y 2 casos a la BD ATIS.
- Por cada caso de prueba se incluye una consulta en LN que involucra un fragmento de la BD que contiene la anomalía y un esquema de BD que tiene la anomalía de diseño.

Las pruebas se realizaron en una sesión de 2 horas y 30 minutos. En dicha sesión se indicó a los usuarios comenzar a realizar la configuración del DIS mediante la interfaz de configuración en el siguiente orden: primero los casos 16 a 20 (columnas repetidas en múltiples tablas), segundo los casos 11 a 15 (columnas que se pueden calcular con FA), tercero los casos 6 a 10 (uso de llaves primarias sustitutas) y por último los casos 1 a 5 (ausencia de llaves foráneas). Esta planificación se realizó, ya que los usuarios nunca habían utilizado la interfaz de configuración y no estaban muy familiarizados con las BDs empleadas. De esta manera se presentaron los casos más sencillos al comienzo para que no invirtieran demasiado tiempo en la primera parte de las pruebas y, posteriormente, en los casos más complejos, no tuvieran problema para resolverlos.

Es importante mencionar que la interfaz de configuración utilizada por los usuarios incluye un software supervisor para contabilizar diversas operaciones e indicarles si han solucionado correctamente los casos de prueba. El proceso del software supervisor se describe en la Figura 3.9.

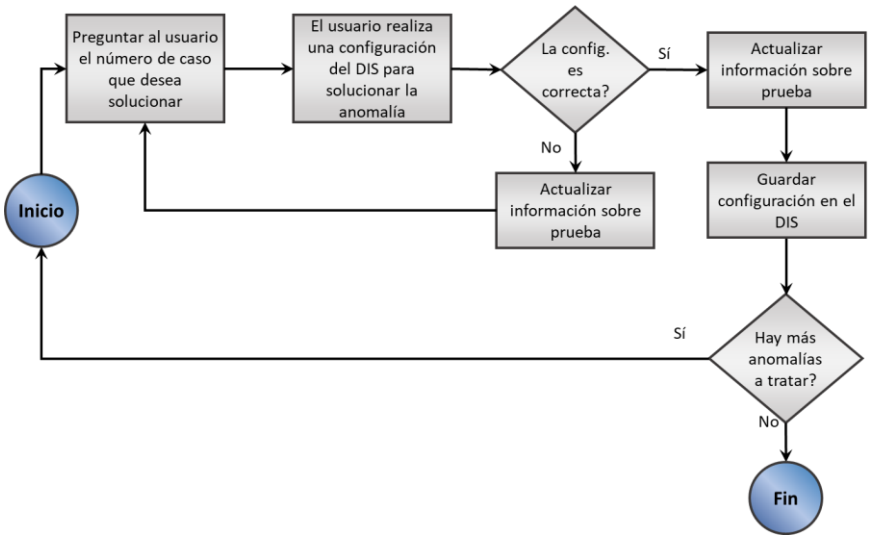


Figura 3.9. Flujo de las pruebas para anomalías de solución simple

3.6 Resultados y Conclusiones

Para recolectar datos de las pruebas, se implementó un software supervisor para medir los siguientes parámetros:

- Número de intentos realizados por cada usuario antes de solucionar cada caso de prueba.
- Tiempo empleado por cada usuario en solucionar cada caso de prueba.
- Número de casos de prueba solucionados correctamente.

En la tabla 3.2 se muestran los resultados de las pruebas en promedio de todos los casos de prueba, donde se puede observar que en promedio 19.1 de los 20 casos de prueba fueron solucionados correctamente por los usuarios. Los usuarios emplearon un promedio de 4.6 intentos para solucionar correctamente un caso de prueba, y el tiempo promedio para solucionar un caso fue de 4.12 minutos.

Tabla 3.2. Resultados de las pruebas para anomalías de solución simple

Total de casos	No. config. correctas	Promedio intentos	Tiempo promedio (segs.)
20	19.09	4.6	246.57

En la Figura 3.10 se puede observar que la mayoría de los usuarios pudieron configurar correctamente el DIS para la mayoría de los casos de prueba. Los únicos casos donde un usuario no configuró correctamente el DIS fueron los casos 6, 7, 8, 9, 10 y 17. En el caso 18 dos usuarios

no configuraron correctamente el DIS, lo cual se debe a que la consulta de este caso requería que los usuarios tuvieran conocimientos más específicos sobre la BD ATIS, por lo cual no pudieron deducir cuáles tablas estaban involucradas en la consulta.

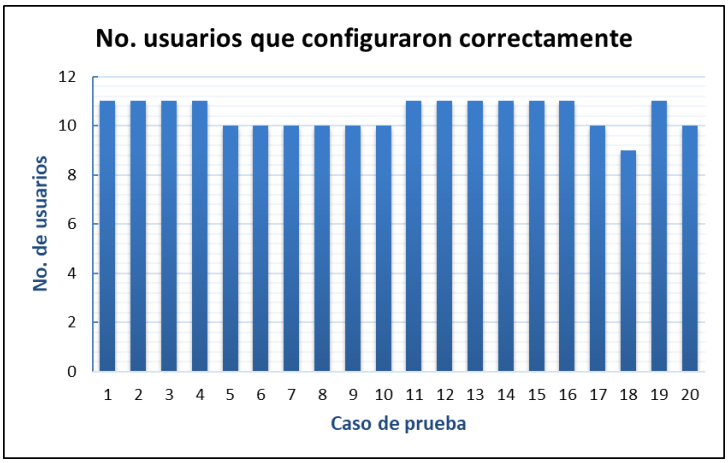


Figura 3.10. Número de usuarios que configuraron correctamente los casos de prueba

En la Figura 3.11 se muestra el tiempo promedio que los usuarios emplearon para configurar correctamente cada caso de prueba. Se puede observar que en el caso 16 los usuarios tardaron más tiempo configurando el DIS. No obstante que el caso 16 (columnas repetidas en múltiples tablas) es fácil de solucionar, los usuarios tuvieron dificultades, debido a que éste fue el primer caso que intentaron solucionar y no tenían experiencia previa en el uso de la interfaz de configuración. De igual manera, en el caso 6 (uso de llaves primarias sustitutas) los usuarios emplearon más tiempo de lo normal, debido a que este caso fue el primero en resolver para este tipo de anomalía que se considera la más difícil de configurar de los cuatro tipos considerados en este documento.

En [Pazos, 2016] se presenta un experimento similar en el cual se solicitó a un grupo de 28 estudiantes de ingeniería en computación que afinaran la configuración de esta ILNBD con el Editor de Dominio y el software ELF [ELF, 2009] con el editor del Diccionario. Los resultados (Tabla 3.3) indican que tardaron 71.3 minutos en promedio para configurar ELF para una consulta, mientras que la configuración de la ILNBD para una consulta fue de 5.8 minutos. Como se puede observar, el tiempo de configuración para una consulta de ELF es 17.30 veces mayor que el tiempo de reconfiguración del DIS para una anomalía. Por otra parte, el tiempo de reconfiguración promedio para una anomalía (4.12 minutos) es 28.96% menor en comparación al tiempo empleado para afinar la configuración para una consulta mediante el Editor de Dominio de la ILNBD (5.8 minutos). En cuanto al porcentaje de éxito, la reconfiguración del DIS obtuvo un mejor valor (95.45%) en comparación con la afinación de la configuración realizada por los implementadores (90%).

Tabla 3.3. Comparación de resultados de experimentos de ELF y la ILNBD

ILNBD	No. casos de prueba	No. casos correctos	% casos correctos	Tiempo promedio
ELF	70 consultas	8.73	12.48	71.3
ILNBD (Editor Dominio)	70 consultas	40.15	57.35	5.8
ILNBD (Interfaz Conf.)	20 anomalías	19.09	95.45	4.12

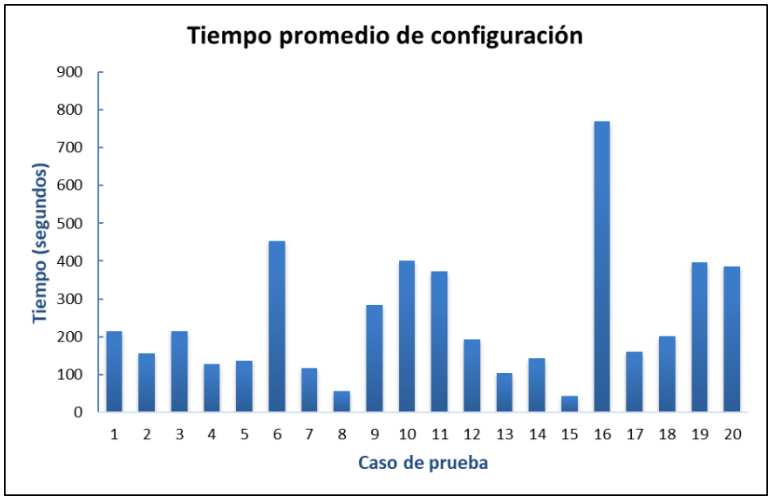


Figura 3.11. Tiempo promedio de configuración por cada caso

En los casos 6, 10 y 16 los usuarios emplearon más intentos que en otros casos, debido a que éstos son los primeros de cada grupo de casos (Figura 3.12), es decir, son los primeros casos de cada tipo de anomalía. Por lo tanto, los usuarios tuvieron que intentar más veces, ya que no habían usado la interfaz para solucionar por primera vez ese tipo de anomalías.

Es importante mencionar que los usuarios podían solicitar asesoría del implementador de la ILNBD para realizar las configuraciones correctamente. Dicha asesoría consistía en explicar parte del esquema de BD para un caso en particular, de tal manera que el usuario pudiera aprender a configurar los demás casos. Los casos en que los usuarios solicitaron más asesorías fueron los casos 19 y 20, ya que estos casos, al pertenecer a la BD ATIS, requieren un mayor conocimiento sobre el dominio de dicha BD, y los usuarios no estaban familiarizados con ésta. Además, los dos casos forman parte del primer grupo de casos que debían resolver los usuarios. Por lo tanto, no tenían tanto entrenamiento con la interfaz de configuración.

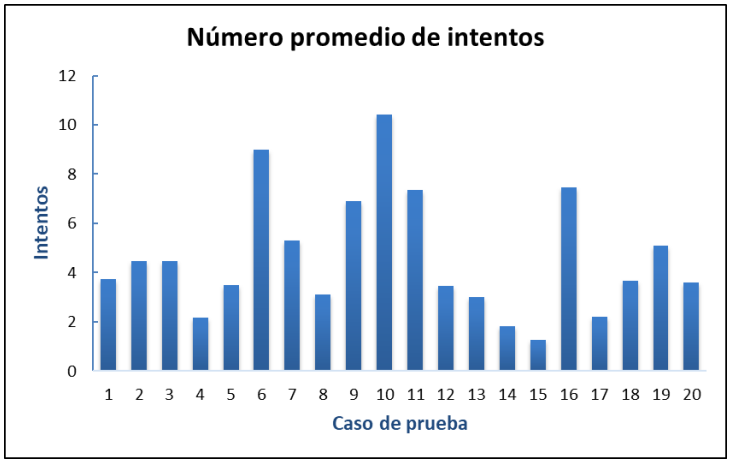


Figura 3.12. Número de intentos por cada caso

Las pruebas presentadas en esta sección demuestran que, con poco entrenamiento sobre la interfaz de configuración y un conocimiento general de la BD en uso, un ABD puede configurar fácilmente el DIS para dar tratamiento a las anomalías de diseño consideradas. En promedio un

usuario empleó alrededor de 4 minutos por anomalía en la reconfiguración del DIS; sin embargo, para la mayoría de las anomalías la reconfiguración tardó menos de 3 minutos, debido a que dichas anomalías no requerían de conocimiento extenso sobre el dominio de la BD en uso.

Asimismo los resultados mostrados en la Tabla 3.3 demuestran que la configuración de la ILNBD mediante la Interfaz de Configuración para Anomalías es más efectiva que la configuración mediante el Editor de Dominio y que también se obtienen mejores resultados que los obtenidos al configurar la ILNBD ELF.

Capítulo 4

Anomalías de Diseño de Solución Compleja

El objetivo de este trabajo fue mejorar el desempeño de la ILNBD desarrollada en el ITCM para BDs que adolecen de anomalías de diseño en su esquema. Para tal fin, fue necesario modificar el DIS para que por medio de éste la ILNBD pudiera tener una representación del esquema de BD sin anomalías de diseño y, de esta manera poder procesar correctamente las consultas formuladas a BDs con falta de 2FN y 3FN.

4.1 Modificaciones al DIS

Con el propósito de dar a la ILNBD una representación del esquema de BD debidamente normalizado, se añadieron tablas al DIS. La estructura de estas tablas permite contener la definición de las metavistas que sirven para representar la descomposición de las tablas con falta de 2FN o 3FN. Nota: se le llama metavista a una percepción de una base de datos, en donde la BD incluye tablas base y vistas.

A continuación se describen las cuatro tablas incorporadas al DIS:

- *falta_2fn_3fn*. Indica qué tablas no están en 2FN o 3FN.
- *metavista*. Sirve para almacenar las metavistas que contendrá una BD.
- *metavista_columnas*. Sirve para almacenar las columnas que contendrá la metavista.
- *metavista_relaciones*. Sirve para almacenar las relaciones que existen entre las metavistas y otras tablas de la BD.

En las Tablas 4.1, 4.2, 4.3 y 4.4 se describen las columnas de las tablas mencionadas.

Tabla 4.1. Columnas de la tabla *falta_2fn_3fn*

Columna	Descripción
nombre_bd	Nombre de la BD.
tabla_base	Nombre de la tabla base que no está en 2FN o 3FN.
tipo	Los posibles valores de esta columna son *2fn* o *3fn*, los cuales indican el tipo de anomalía de que adolece la tabla base.

Tabla 4.2. Columnas de la tabla *metavista*

Columna	Descripción
nombre_bd	Nombre de la BD.
metavista	Nombre de la metavista.
tabla_base	Nombre de la tabla base a la cual hace referencia la metavista.

Tabla 4.3. Columnas de la tabla *metavista_columnas*

Columna	Descripción
nombre_bd	Nombre de la BD.
metavista	Nombre de la metavista.
columna	Nombre de la columna que pertenece a la metavista.
columna_base	Nombre de la columna base a la cual hace referencia la columna de la metavista.

Tabla 4.4. Columnas de la tabla *metavista_relaciones*

Columna	Descripción
nombre_bd	Nombre de la BD.
entidad_referente	Nombre de la tabla base o metavista que hace referencia a otra tabla o metavista.
columna_referente	Nombre de la columna perteneciente a la entidad referente.
entidad_referida	Nombre de la tabla base o metavista que es referida por la entidad referente.
columna_referida	Nombre de la columna que pertenece a la entidad referida.

4.2 Tratamiento de Falta de Segunda Forma Normal

De acuerdo con la literatura, una tabla no se encuentra en 2da forma normal, si al menos una columna no es dependiente de la llave primaria natural de la tabla [Date, 2004]. Para solucionar este problema, normalmente se realiza una descomposición de la tabla para normalizarla tal como se muestra en la Figura 4.1 [Databasedev, 2020].

Tomando el ejemplo de la Figura 4.1, considere una tabla T con las siguientes características: tiene una llave primaria compuesta de dos columnas (C_{PK1}, C_{PK2}), donde C_{PK2} define funcionalmente a un conjunto de columnas $C_{DEP.}$ Por lo tanto, C_{DEP} está definida parcialmente por la llave primaria. Además, T tiene un conjunto C_{IND} de columnas que dependen funcionalmente de la llave primaria (C_{PK1}, C_{PK2}), pero que no dependan funcionalmente de C_{PK2}. Además, a partir de la tabla T, se crean las vistas V_1 y V_2.

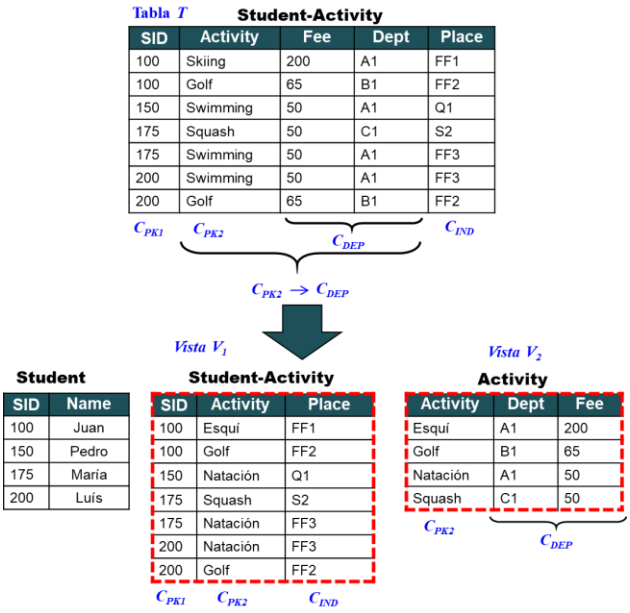


Figura 4.1. Descomposición de una tabla sin 2da forma normal

Ampliando el ejemplo anterior, en la Figura 4.2 se muestra un esquema de BD, el cual consta de 4 tablas, donde la tabla *Student-Activity* no se encuentra en 2FN.

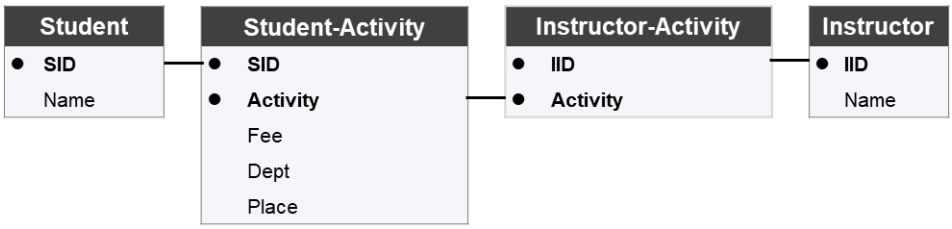


Figura 4.2. Ejemplo de falta de 2FN

Considere la siguiente consulta en LN: ***Dame las cuotas que maneja el instructor Juan***. Esta consulta, al ser procesada por la ILNBD sin dar tratamiento a anomalías de diseño, identifica que la unidad léxica *cuotas* hace referencia a la columna *Fee* de la tabla *Student-Activity* y la asocia a la cláusula Select. Además, la unidad léxica *instructor* hace referencia a la columna *Instructor.Name* y la asocia a la cláusula Where. Posteriormente, la ILNBD detecta que *Juan* es un valor de búsqueda, el cual asocia a la columna *Instructor.Name*.

Por último, la ILNBD construye una consulta en SQL que es la representación de la consulta en LN en SQL (Instrucción 1).

Instrucción 1:

```
1:  SELECT Student-Activity.Fee
2:  FROM Student-Activity, Instructor-Activity, Instructor
3:  WHERE Instructor.Name LIKE 'Juan' AND
4:          Instructor-Activity.IID = Instructor.IID AND
5:          Student-Activity.Activity = Instructor-Activity.Activity
```

La Instrucción 1 obtiene como resultado una tabla, cuyos renglones se encuentran repetidos, ya que un instructor tiene varias actividades con diferentes estudiantes. Por lo tanto, la tabla de resultado contiene las cuotas repetidas por cada alumno que el instructor tenga por actividad, mientras que lo solicitado por la consulta en LN son las cuotas de las actividades que imparte el instructor.

Para evitar este tipo de errores, se modificó la estructura del DIS para dar tratamiento a esta anomalía de diseño al representar normalizada la tabla que tiene la anomalía (en este caso, *Estudiante-Actividad*).

Esta sección describe las modificaciones realizadas al procesamiento de consultas en LN para dar tratamiento a la falta de 2FN y 3FN. A continuación se explica el tratamiento de falta de 2FN y el procesamiento de consultas que involucran tablas con falta de 2FN.

Cuando una tabla no se encuentra en 2da forma normal, normalmente se realiza una descomposición de la tabla para normalizarla tal como se muestra en la Figura 4.1.

Primero, se deben identificar las columnas C_{PK2} y $C_{DEP.}$

Posteriormente, se define una vista V_1, la cual está conformada por las columnas C_{PK1}, C_{PK2} y C_{IND}, donde C_{PK2} sería una llave foránea que relaciona V_1 con la vista V_2 cuya construcción se explica más adelante.

```
1: CREATE VIEW V1 (CPK1, CPK2, CIND) AS
2: SELECT CPK1, CPK2, CIND FROM T
```

Por último, se define una vista V_2 que contiene las columnas C_{PK2} y C_{DEP}, donde C_{PK2} sería la llave primaria natural de V_2. Para construir la vista se podría utilizar la siguiente expresión en SQL:

```
1: CREATE VIEW V2 (CPK2, CDEP) AS
2: SELECT DISTINCT CPK2, CDEP
3: FROM T
```

Cabe mencionar que las consultas en LN que soliciten información sobre una tabla con esta anomalía de diseño serán traducidas a SQL de la siguiente manera: se realizará el análisis semántico para identificar los elementos de las vistas que se requerirán incluir en la consulta; después, se construirá la consulta a partir de la estructura de las vistas generadas; y por último se generará una instrucción en SQL basada en las tablas base que sea semánticamente equivalente a la consulta generada a partir de las vistas.

Para tal efecto, mediante un módulo de configuración, se definen en el DIS la metavista *Actividad* y la metavista *Estudiante-Actividad* (Figura 4.3). Mismas que son incluidas en la configuración del DIS de la ILNBD.

Considerando el ejemplo y su esquema de BD (Figura 4.2), la tabla *Student-Activity* no está en 2FN. Por lo tanto, el Administrador de BD deberá hacer uso de un módulo de configuración en la ILNBD para corregir esta anomalía de diseño. La configuración de las tablas *falta_2fn_3fn*, *metavista*, *metavista_columnas* y *metavista_relaciones* queda como se muestran en las Tablas 4.5, 4.6, 4.7 y 4.8 respectivamente.

Tabla 4.5. Información de la tabla *falta_2fn_3fn* para el ejemplo de falta de 2FN

nombre_bd	tabla_base	tipo
Students	Student-Activity	2fn

En la tabla *falta_2fn_3fn* se registra la tabla base que adolece de falta de 2FN, que es en este caso *Student-Activity*, así como el tipo de anomalía (2fn).

Tabla 4.6. Información de la tabla *metavista* para el ejemplo de falta de 2FN

nombre_bd	metavista	tabla_base
Students	Student-Activity	Student-Activity
Students	Activity	Student-Activity

En la tabla *metavista* se especifican los nombres de las metavistas (*Student-Activity* y *Activity*) en las cuales se descompondrá la tabla base que adolezca de falta de 2FN (*Student-Activity*).

Tabla 4.7. Información de la tabla *metavista_columnas* para el ejemplo de falta de 2FN

nombre_bd	metavista	columna	columna_base
Students	Student-Activity	SID	SID
Students	Student-Activity	Activity	Activity
Students	Student-Activity	Place	Place
Students	Activity	Activity	Activity
Students	Activity	Fee	Fee
Students	Activity	Dept	Dept

En la tabla *metavista_columnas* se especifican las columnas que tendrán las metavistas (en este caso, las columnas de las metavistas *Student-Activity* y *Activity*) y se relacionan a las columnas de las tablas base.

Tabla 4.8. Información de la tabla *metavista_relaciones* para el ejemplo de falta de 2FN

nombre_bd	entidad_referente	columna_referente	entidad_referida	columna_referida
Students	Student-Activity	Activity	Activity	Activity
Students	Student-Activity	SID	Student	SID
Students	Instructor-Activity	Activity	Activity	Activity

En la tabla *metavista_relaciones* se especifican las relaciones existentes entre las metavistas y otras metavistas o tablas base de la BD (p. ej., *Student-Activity* REFERENCES *Activity*).

Una vez realizada la configuración del DIS, la ILNBD puede utilizar esta información en el proceso de traducción. Por lo tanto, el esquema de BD que estaría percibiendo la ILNBD es el que se muestra en la Figura 4.3.

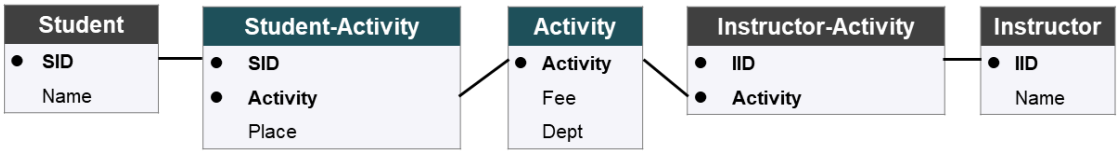


Figura 4.3. Esquema de BD normalizado para el ejemplo de falta de 2FN

4.3 Tratamiento de Falta de Tercera Forma Normal

De acuerdo con la literatura, una tabla no está en tercera forma normal (3FN) cuando existen columnas que tienen dependencia transitiva con otras columnas que no son la llave primaria [Date, 2004]. En la Figura 4.4 se muestra un ejemplo simple de la violación de la tercera forma normal [Rob, 2011], donde la tabla *employee* no se encuentra en 3FN, debido a que la columna *chg_hour* no depende funcionalmente de la llave primaria *emp_num*; además, *chg_hour* depende funcionalmente de *job_class* (dependencia transitiva).

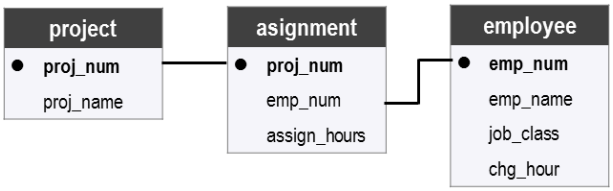


Figura 4.4. Ejemplo de falta de 3FN

Considere la siguiente consulta en LN: ***Dame los puestos y sueldos de los empleados***. Esta consulta, al ser procesada mediante la ILNBD previamente configurada utilizando el esquema de la Figura 4.4, identifica que las unidades léxicas *puestos* y *sueldos* hacen referencia a las columnas *job_class*, *chg_hour* de la tabla *employee*. En esta consulta no se detectan valores de búsqueda.

La ILNBD construye una consulta en SQL que es la representación de la consulta en LN en SQL (Instrucción 2).

Instrucción 2:

1: **SELECT** *employee.job_class, employee.chg_hour*
2: **FROM** *employee*

La Instrucción 2 obtiene como resultado una tabla cuyos renglones se encuentran repetidos, ya que la información requerida se refiere a una entidad (en este caso, *job*) que se encuentra dentro de la tabla *employee*. Por lo tanto, la información de dicha entidad se encuentra repetida por cada empleado, es decir, por cada renglón de la tabla *employee*.

Esta anomalía puede resolverse realizando una descomposición de la tabla. Dicha descomposición sirve para eliminar las dependencias transitivas y se explica a continuación.

Considere una tabla *T* cuyas columnas son C_{PK}, C_1, …, C_n, C_{det} y C_{dep}, donde C_{PK} es la llave primaria natural de *T*, y $C_{det} \rightarrow C_{dep}$ (C_{dep} es funcionalmente dependiente de C_{det}). Por lo tanto, C_{dep} no depende de la llave primaria C_{PK}.

Primero se crea una vista V_1 que contiene las columnas que tienen dependencia transitiva $C_{det} \rightarrow C_{dep}$, siendo C_{det} la llave primaria de V_1.

```
1: CREATE VIEW V1 (Cdet, Cdep) AS
2: SELECT DISTINCT Cdet, Cdep FROM T
```

Posteriormente, se crea una vista V_2 que contiene la llave primaria natural C_{PK} de la tabla *T*, las columnas C_1, …, C_n y una llave foránea C_{det} que hace referencia a V_1. Cabe mencionar que V_2 no contiene la columna C_{dep}.

```
1: CREATE VIEW V2 (CPK, C1, …, Cn, Cdet) AS
2: SELECT Cdet, C1, …, Cn FROM T
```

Considerando el ejemplo y su esquema de BD (Figura 4.4), la tabla *employee* no está en 3FN. Por lo tanto, el Administrador de BD deberá hacer uso de un módulo de configuración en la ILNBD para corregir esta anomalía de diseño. La configuración de las tablas *falta_2fn_3fn*, *metavista*, *metavista_columnas* y *metavista_relaciones* queda como se muestran en las Tablas 4.9, 4.10, 4.11 y 4.12 respectivamente.

Tabla 4.9. Información de la tabla *falta_2fn_3fn* para el ejemplo 1

nombre_bd	tabla_base	tipo
ConstructCo	employee	3fn

En la tabla *falta_2fn_3fn* se especifica la tabla base que tiene la falta de 3FN (*employee*) y el tipo de anomalía (3fn).

Tabla 4.10. Información de la tabla *metavista* para el ejemplo 1

nombre_bd	metavista	tabla_base
ConstructCo	employee	employee
ConstructCo	job	employee

En la tabla *metavista* el ABD especifica los nombres de las metavistas (*employee* y *job*) en las cuales se descompondrá la tabla base que adolece de falta de 3FN (*employee*).

Tabla 4.11. Información de la tabla *metavista_columnas* para el ejemplo 1

nombre_bd	metavista	columna	columna_base
ConstructCo	employee	emp_num	emp_num

ConstructCo	employee	emp_name	emp_name
ConstructCo	employee	job_class	job_class
ConstructCo	job	job_class	job_class
ConstructCo	job	chg_hour	chg_hour

En la tabla *metavista_columnas* se especifican las columnas que tendrán las metavistas (en este caso, las columnas de las metavistas *employee* y *job*) y se relacionan a las columnas de las tablas base.

Tabla 4.12. Información de la tabla *metavista_relaciones* para el ejemplo 1

nombre_bd	entidad_referente	columna_referente	entidad_referida	columna_referida
ConstructCo	asignment	emp_num	employee	emp_num
ConstructCo	employee	job_class	job	job_class

En la tabla *metavista_relaciones* se especifican las relaciones existentes entre las metavistas y otras metavistas o tablas base de la BD (p. ej., *asignment* REFERENCES *employee*).

Una vez realizada la configuración del DIS, la ILNBD puede utilizar esta información en el proceso de traducción, por lo que el esquema de BD que estaría percibiendo la ILNBD es el que se muestra en la Figura 4.5.

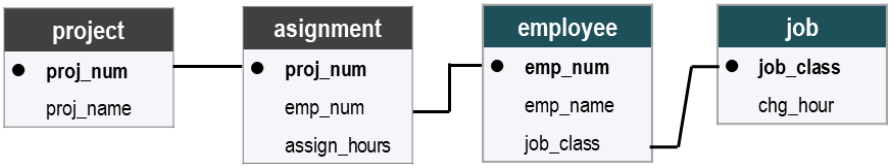


Figura 4.5. Esquema de BD normalizado para el ejemplo de falta de 2FN

4.4 Procesamiento de Consultas que Involucran Falta de 2FN y 3FN

Cabe mencionar que las consultas en LN que soliciten información sobre una tabla con falta de 2FN o 3FN serán traducidas a SQL de la siguiente manera: se realizará el análisis semántico para identificar los elementos de las vistas que se requerirán incluir en la consulta; después, se construirá la consulta a partir de la estructura de las vistas generadas; y por último se generará una instrucción en SQL basada en las tablas base que sea semánticamente equivalente a la consulta generada a partir de las vistas.

A continuación se presentan dos algoritmos, los cuales tienen como objetivo describir el proceso que realiza la ILNBD para procesar consultas que involucran tablas con falta de 2FN o 3FN. El Algoritmo 4.1 es ejecutado por la ILNBD después de construir la consulta en SQL basada en metavistas y tablas base (Q_M). El algoritmo 2 es ejecutado por la ILNBD para construir una consulta basada en tablas base. Los procesos descritos a continuación se realizan en la fase final del análisis semántico ejecutado por la ILNBD (Figura 4.6).

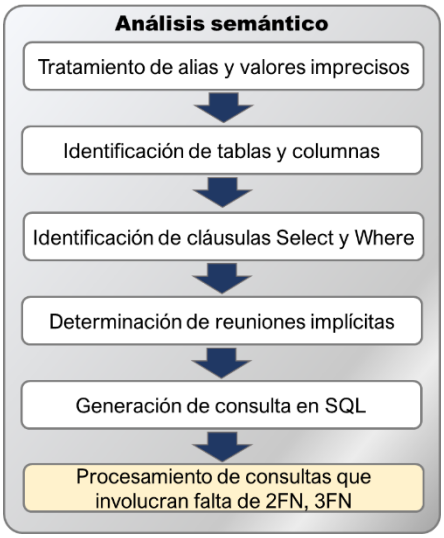


Figura 4.6. Estructura del análisis semántico de la ILNBD

A continuación se describe el Algoritmo 4.1. Primero, entre las tablas involucradas en la consulta se realiza una búsqueda para encontrar las tablas que tengan falta de 2FN o 3FN (líneas 2 a 7). En caso de encontrar tablas con dicha característica, se guardan en *T* (línea 5). Posteriormente, si alguna tabla con falta de 2FN o 3FN está involucrada en la consulta, entonces la ILNBD deberá generar una consulta sobre tablas base únicamente (línea 9), ya que la consulta generada mediante el proceso normal de traducción (consulta que puede involucrar metavistas y tablas base Q_M) puede hacer referencia a alguna metavista para tratar la falta de 2FN o 3FN.

Algoritmo 4.1. Pseudocódigo para procesar consultas que involucran tablas con falta de 2FN o 3FN

columns // conjunto de columnas involucradas en la consulta
n // número de columnas involucradas en la consulta
VB // conjunto que contiene valores de búsqueda
2FN3FNTables // conjunto de tablas involucradas en la consulta con falta de 2FN o 3FN
Q_M // consulta que involucra metavistas y tablas base
Q_B // consulta que involucra tablas base

```
1:  procedure process2FN3FN
2:  for i = 0 to n-1
    // Buscar las columnas que pertenezcan a tablas con falta de 2FN o 3FN
3:      table ← getTable(columns[i])
4:      if noncoformity2FN3FN(table) then
            // La columna pertenece a una tabla sin 2FN o 3FN
5:          add table to 2FN3FNTables
6:      end if
7:  end for
8:  if 2FN3FNTables is not empty then
        // Al menos una tabla con falta de 2FN o 3 FN está involucrada
9:      Q_B ← generateBaseTablesQuery(Q_M)
10: end if
11: return Q_B
```

El Algoritmo 4.2 describe el proceso realizado por la ILNBD para generar una consulta en SQL sobre tablas base a partir de una consulta sobre metavistas y tablas base. Para tal fin, primero separa la consulta que involucra metavistas y tablas base (Q_M) en tres cláusulas: Select, From, Where y obtiene sus columnas, tablas y condiciones respectivamente (líneas 2, 3 y 4). Después, la ILNBD genera los componentes para construir las cláusulas Select, From y Where de la consulta sobre tablas base Q_B.

Para generar los componentes de la cláusula Select de Q_B (líneas 5 a 8), se convierten las columnas de la cláusula Select de Q_M (*selectColumnsM*) a columnas de sus respectivas tablas base con ayuda del DIS (tabla *metavista_columnas*).

Para generar los componentes de la cláusula From de Q_B (líneas 9 a 12), se convierten las tablas que aparecen en la cláusula From de Q_M (*fromTablesM*) a sus respectivas tablas base con ayuda del DIS (tabla *metavista*).

Para generar los componentes de la cláusula Where de Q_B (líneas 13 a 27), primero se convierten las columnas de los valores de búsqueda a sus respectivas columnas de tablas base (línea 15). Posteriormente, se evalúan las reuniones de Q_M; para tal efecto, si una reunión se efectúa entre dos metavistas (línea 19 y 20), ésta es ignorada y no se incluye en la cláusula Where de Q_B. Además, si la reunión involucra una metavista y una tabla base, entonces se convierten las columnas pertenecientes a metavistas a su equivalente en tablas base (línea 23) mediante el DIS (tabla *metavista_relaciones*).

Por último, se genera una instrucción Select-From-Where (línea 28) con los componentes de *selectColumnsB*, *fromTablesB* y *whereRestrictionsB*, dando como resultado una consulta basada en tablas base (Q_B).

Algoritmo 4.2. Pseudocódigo para generar una consulta sobre tablas base a partir de una consulta sobre metavistas y tablas base

```
Q_M // consulta que involucra metavistas y tablas base
Q_B // consulta que involucra tablas base
selectColumnsM // lista de columnas de la cláusula Select de la instrucción Q_M
fromTablesM // lista de tablas de la cláusula From de la instrucción Q_M
whereRestrictionsM // lista de condiciones de la cláusula Where de la instrucción Q_M
m // número de columnas de la cláusula Select de la instrucción Q_M
n // número de tablas de la cláusula From de la instrucción Q_M
p // número de condiciones de la cláusula Where de la instrucción Q_M
selectColumnsB // lista de columnas de la cláusula Select de la instrucción Q_B
fromTablesB // lista de tablas de la cláusula From de la instrucción Q_B
whereRestrictionsB // lista de condiciones de la cláusula Where de la instrucción Q_B
1:  procedure generateBaseTablesQuery(Q_M)
2:  selectColumnsM ← obtenerSelect(Q_M)
3:  fromTablesM ← obtenerFrom(Q_M)
4:  whereRestrictionsM ← obtenerWhere(Q_M)
    // generar cláusula Select de la instrucción Q_B
```

```
5:   for i = 0 to m-1 do
6:       column ← get base table and column of selectMColumns[i]
7:       add column to selectColumnsB
8:   end for
     // generar cláusula From de la instrucción Q_B
9:   for i = 0 to n-1 do
10:      table ← get base table of fromTablesM[i]
11:      add table to fromTablesB
12:  end for
     // generar cláusula Where de la instrucción Q_B
13:  for i = 0 to p-1 do
14:      if isSearchValue(whereRestrictionsM[i])
15:          columnValue ← getBaseColumn(whereRestrictionsM[i])
16:          add columnValue to whereRestrictionsB
17:      end if
18:      if isJoin(WhereRestrictionsM[i]) then
19:          if isMMJoin(whereRestrictionsM[i]) then
20:                  ignore join //La reunión es entre metavistas, por lo tanto, se ignora
21:          end if
22:          if esReunionMTB(whereRestrictionsM[i])
     // La reunión es de metavista a tabla base
23:                  join← sustituir por las columnas de tabla base correspondientes
24:                  add join to whereRestrictionsB
25:          end if
26:      end if
27:  end for
28:  Q_B ← "SELECT DISTINCT " + selectColumnsB + " FROM "+ fromTablesB
           + " WHERE " + whereRestrictionsB
29:  return Q_B
```

4.5 Evaluación Experimental

Con la finalidad de verificar el buen funcionamiento de la ILNBD al procesar consultas que involucran tablas que no están en 2FN o 3FN, se realizaron pruebas funcionales con la ILNBD.

Estas pruebas consistieron en diseñar 10 casos de prueba: 5 casos de prueba se utilizaron para probar la ILNBD con tablas que involucran falta de 2FN, y 5 casos de prueba involucran tablas con falta de 3FN.

Cada caso de prueba consta de los siguientes elementos:

- BD que contiene una tabla con falta de 2FN o 3FN según sea el caso.
- DIS configurado mediante la configuración inicial automática realizada por la ILNBD.

- Consulta en LN que involucra una tabla con falta de 2FN o 3FN según sea el caso.

El resultado de cada caso de prueba consta de los siguientes elementos:

- DIS afinado para dar tratamiento a las faltas de 2FN y 3FN.
- Consulta en SQL producto de la traducción de la consulta en LN.

Para las pruebas de falta de 2FN se utilizaron dos esquemas de BD. En las Figuras 4.7 y 4.8 se muestran las tablas y columnas de estos esquemas.

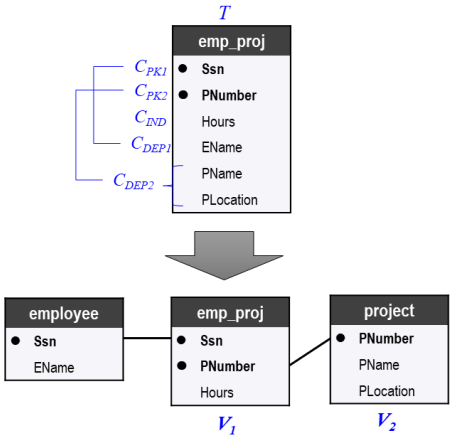


Figura 4.7. Esquema y descomposición de la tabla *emp_proj*

Como se puede observar en la Figura 4.7, la tabla *emp_proj* no está en 2FN, debido a que existe una dependencia funcional entre las columnas *PNumber*, *PName* y *PLocation*; dichas columnas corresponden a una entidad llamada *project.*

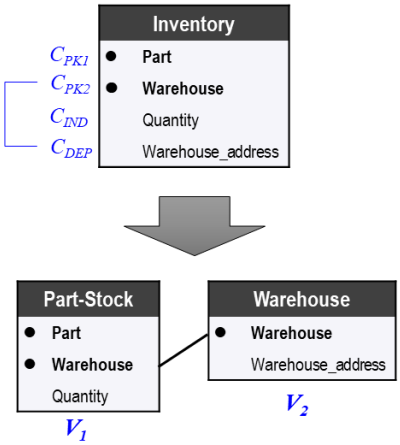


Figura 4.8. Esquema y descomposición de la tabla *Inventory*

En la Figura 4.8 se muestra la tabla *Inventory* que no está en 2FN. Existe una dependencia parcial que involucra a la columna *Warehouse* con *Warehouse_address*.

Con el objetivo de comparar el funcionamiento de otras ILNBDs y nuestra ILNBD al procesar consultas con BDs con falta de 2FN y 3FN, se formularon consultas en la interfaz ELF.

Las consultas empleadas para estas pruebas se muestran en la Tabla 4.15. Asimismo se muestran los resultados obtenidos de las pruebas.

Por otra parte, para comprobar el buen funcionamiento de la ILNBD al procesar consultas que involucran tablas con falta de 3FN, se formularon 5 consultas utilizando el esquema de BD mostrado en la Figura 4.9.

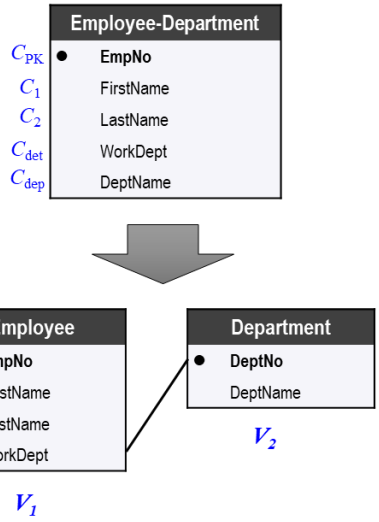


Figura 4.9. Esquema y descomposición de la tabla *Employee-Department*

La tabla *Employee-Department* mostrada en la Figura 4.9 no está en 3FN, debido a que *WorkDept* define funcionalmente a *DeptName*, y *DeptName* no está en función de la llave primaria *EmpNo*. Al igual que el experimento realizado para probar la falta de 2FN, se hizo una comparación de la ILNBD ELF con nuestra ILNBD.

4.6 Resultados y Conclusiones

Las pruebas realizadas consistieron en diseñar 5 consultas para probar el funcionamiento de las ILNBDs con BDs con falta de 2FN y 5 consultas para probar el funcionamiento con BDs con falta de 3FN, dando un total de 10 consultas. Asimismo, para nuestra ILNBD, se realizó la configuración del DIS mediante las tablas *falta_2fn_3fn*, *metavista*, *metavista_columnas* y *metavista_relaciones*, cuyos contenidos se muestran en las tablas 4.13, 4.14 y 4.15.

Tabla 4.13. Información de la tabla *falta_2fn_3fn* para las pruebas

nombre_bd	tabla_base	tipo
Ej1Falta2FN	emp_proj	2fn
Ej2Falta2FN	Inventory	2fn
Ej1Falta3FN	Employee-Department	3fn

Tabla 4.14. Información de la tabla *metavista* para las pruebas

nombre_bd	metavista	tabla_base
Ej1Falta2FN	employee	emp_proj
Ej1Falta2FN	emp_proj	emp_proj
Ej1Falta2FN	project	emp_proj
Ej2Falta2FN	Part-Stock	Inventory
Ej2Falta2FN	Warehouse	Inventory
Ej1Falta3FN	Employee	Employee-Department
Ej1Falta3FN	Department	Employee-Department

Tabla 4.15. Información de la tabla *metavista_columnas* para las pruebas

nombre_bd	metavista	columna	columna_base
Ej1Falta2FN	employee	Ssn	Ssn
Ej1Falta2FN	employee	EName	EName
Ej1Falta2FN	emp_proj	Ssn	Ssn
Ej1Falta2FN	emp_proj	PNumber	PNumber
Ej1Falta2FN	emp_proj	Hours	Hours
Ej1Falta2FN	project	PNumber	PNumber
Ej1Falta2FN	project	PName	PName
Ej1Falta2FN	project	PLocation	PLocation
Ej2Falta2FN	Part-Stock	Part	Part
Ej2Falta2FN	Part-Stock	Warehouse	Warehouse
Ej2Falta2FN	Part-Stock	Warehouse_address	Warehouse_address
Ej1Falta3FN	Employee	EmpNo	EmpNo
Ej1Falta3FN	Employee	FirstName	FirstName
Ej1Falta3FN	Employee	LastName	LastName
Ej1Falta3FN	Employee	WorkDept	WorkDept
Ej1Falta3FN	Department	DeptNo	DeptNo
Ej1Falta3FN	Department	DeptName	DeptName

Tabla 4.16. Información de la tabla *metavista_relaciones* para las pruebas

nombre_bd	entidad_referente	columna_referente	entidad_referida	columna_referida
Ej1Falta2FN	emp_proj	Ssn	employee	Ssn
Ej1Falta2FN	emp_proj	PNumber	project	PNumber
Ej2Falta2FN	Part-Stock	Warehouse	Warehouse	Warehouse
Ej1Falta3FN	Employee	WorkDept	Department	DeptNo

En la tabla 4.17 se muestran los resultados obtenidos de las 10 consultas formuladas a nuestra ILNBD y la interfaz ELF.

Tabla 4.17. Resultados de las pruebas

Consulta en LN	Tabla con anomalía	Nuestra ILNBD	ELF
Consultas con tablas que no están en 2FN			
1. ¿Cuál es la ubicación del proyecto Bird?	emp_proj	✓	✓
2. Dame el Número de seguro de los empleados que trabajen menos de 5 horas.	emp_proj	✓	×
3. Dame la dirección del almacén B.	Inventory	✓	×
4. Dame las partes contenidas en el almacén C.	Inventory	✓	×
5. Dame los almacenes que tengan cantidades mayores de 100.	Inventory	✓	✓
Consultas con tablas que no están en 3FN			
6. Dame el nombre de departamento del departamento número E11.	Employee-Department	✓	×
7. Dame los nombres de los empleados del departamento de Operaciones.	Employee-Department	✓	✓
8. ¿Cuáles son los empleados que trabajan en el departamento E20?	Employee-Department	×	×
9. ¿En qué departamento trabaja el empleado John?	Employee-Department	✓	×
10. ¿Cuáles son los apellidos de los empleados el departamento de Software Support?	Employee-Department	✓	✓

La ILNBD pudo contestar correctamente 9 de 10 consultas. La consulta 8 *¿Cuáles son los empleados que trabajan en el departamento E20?* fue la única consulta que contestó mal, ya que a la unidad léxica *departamento*, al ser configurada en la ILNBD, se le asignó la columna *DeptName*. Sin embargo, a lo que se refiere la consulta es al número de departamento. Por lo tanto, la ILNBD asocia la unidad léxica *departamento* a la columna *DeptName*, y su valor de búsqueda fue *E20*; pero la respuesta correcta debía ser *DeptNo* = E20.

La ILNBD ELF contestó 4 de 10 consultas correctamente. Cabe mencionar que estas consultas solicitan información que se puede obtener a partir de la tabla con falta de 2FN o 3FN sin necesidad de dar tratamiento a dichas anomalías. Las consultas restantes no fueron respondidas por ELF, porque en algunos casos no comprendía algunas palabras y en otros casos daba respuestas equivocadas.

Capítulo 5

Conclusiones y Trabajo Futuro

5.1 Conclusiones

El esquema de BD es un aspecto muy importante a considerar en el proceso de traducción de consultas en LN a SQL. En el proceso de traducción de una ILNBD intervienen varios factores tales como el método de traducción, el esquema de la BD en uso, las características de la consulta en LN, entre otros.

En lo que respecta a la ILNBD desarrollada en el ITCM, es importante mencionar que ésta fue desarrollada bajo el supuesto de que sería utilizada con BDs correctamente diseñadas, es decir, BDs sin anomalías de diseño. Desafortunadamente, en la práctica, podemos verificar que existen muchas BDs que adolecen de este problema. Debido a lo anterior, la funcionalidad de esta ILNBD se ve reducida al ser usada con BDs con anomalías de diseño, ya que el uso de dichas BDs en la ILNBD ocasiona que las consultas en LN formuladas a la interfaz den como resultado consultas en SQL con información errónea (objetivo específico OE2).

Después de realizar una búsqueda sobre las anomalías de diseño más comunes en BDs, se realizó una clasificación de las anomalías de diseño, ya que algunas eran más complejas que otras (por lo cual se cumple el objetivo específico OE1). Por lo tanto, se clasificaron en dos grupos:

- Anomalías de solución simple.
  - Ausencia de llaves primarias y foráneas.
  - Uso de llaves primarias sustitutas.
  - Columnas cuyos valores se pueden calcular con funciones de agregación.
  - Columnas repetidas en múltiples tablas.
- Anomalías de solución compleja.
  - Falta de segunda forma normal.
  - Falta de tercera forma normal.

Para verificar la factibilidad de este trabajo, se hicieron pruebas funcionales con las ILNBDs ELF y C-Phrase (Apéndice C). En dichas pruebas se puede apreciar que ELF y C-Phrase no cuentan con mecanismos para dar tratamiento a anomalías de diseño en BDs y, por lo tanto, no pueden contestar correctamente consultas que involucren tablas que adolecen de este problema. Asimismo en la literatura no se mencionan ILNBDs que cuenten con mecanismos con estas características, por lo tanto, se cumple el OE 6.

Para solventar el problema mencionado, en este trabajo se diseñaron e implementaron módulos de configuración del DIS y algoritmos para dar tratamiento a las anomalías ya mencionadas (ver Sección 2.1.20).

El tratamiento de las anomalías de diseño consistió en modificar y añadir tablas al DIS de la ILNBD para que ésta tuviera información sobre las tablas que tienen anomalías de diseño, y así poder identificarlas. Posteriormente, dicho tratamiento incluyó el almacenamiento en las tablas del DIS de una representación de la BD sin las anomalías de diseño. Sin embargo, para que la ILNBD pudiera utilizar las tablas modificadas o añadidas al DIS, fue necesario modificar el proceso de traducción al implementar diversos algoritmos (Algoritmos 3.1, 3.2, 4.1 y 4.2), por lo que el objetivo específico OE3 queda cubierto.

Para dar tratamiento a las anomalías de falta de 2FN y 3FN, se añadieron al DIS las tablas *falta_2fn_3fn*, *metavista*, *metavista_columnas* y *metavista_relaciones*. Estas tablas permiten a la ILNBD tener una metavista, es decir, una percepción de una base de datos, donde la BD incluye tablas base y vistas (por lo que el objetivo específico OE4 se cumple).

En este trabajo se presentaron 2 conjuntos de pruebas experimentales.

Las primeras pruebas fueron diseñadas con el objetivo de verificar el funcionamiento correcto de la ILNBD, al contestar consultas que involucran anomalías de diseño de solución simple (Capítulo 3), y al mismo tiempo verificar la usabilidad del módulo de configuración desarrollado para dar tratamiento a estas anomalías. Los resultados de estas pruebas arrojaron que los usuarios de la interfaz de configuración implementada emplearon aproximadamente 4.12 minutos para tratar una anomalía de diseño; además, obtuvo un promedio de 19.09 casos de prueba correctamente configurados, lo que equivale a un 95.45% de 20 casos de prueba.

El segundo conjunto de pruebas se realizó con el objetivo de verificar el funcionamiento de la ILNBD, al contestar consultas que involucran anomalías de diseño de solución compleja (falta de 2FN y falta de 3FN), y a su vez comparar con el funcionamiento de ELF con BDs que adolecen de estas anomalías (se cumplen OE5 y OE7). Para tal fin, se diseñó un corpus de 10 consultas distribuidas para su formulación en tres esquemas de BD; dos de estos esquemas adolecían de falta de 2FN y uno, de falta de 3FN.

La ILNBD fue capaz de contestar correctamente 9 de 10 consultas, donde la contestación errónea se debió a un problema de configuración de la ILNBD. La ILNBD ELF contestó correctamente 4 consultas de 10. En estas pruebas se puede observar que ELF no contesta correctamente, porque no tiene un mecanismo para tratar las anomalías de diseño, pero sólo contesta correctamente las consultas cuyos resultados se pueden obtener directamente de la tabla con anomalías.

Los módulos de configuración para la ILNBD fueron implementados utilizando Java Enterprise Edition (JEE), el cual permitió reutilizar el código de la versión anterior de la ILNBD implementada en Java Standard Edition (JSE), y de esta forma añadirle funcionalidad mediante conexión a internet con un navegador de web. Esta implementación permitió la realización de las pruebas para las anomalías de diseño de solución sencilla, ya que varios usuarios pudieron conectarse al servidor de la ILNBD.

5.2 Trabajo Futuro

Aún existen problemas relacionados con BDs con diseños defectuosos. Por lo cual se proponen los siguientes trabajos:

1. Diseño e implementación de un módulo para dar tratamiento a la falta de primera forma normal en la ILNBD.
2. Mejora del módulo para dar tratamiento a la ausencia de llaves primarias y foráneas, añadiendo una interfaz gráfica amigable para el ABD.
3. Mejora del tratamiento de columnas cuyos valores se pueden calcular con FAs, mediante el desarrollo de un método que permita aplicar agrupamientos y otras FAs.
4. Búsqueda de otras anomalías de diseño que pudieran afectar el desempeño de la ILNBD.
5. Implementación de herramientas para realizar pruebas en la ILNBD.
6. Mejora en el módulo de configuración inicial de la ILNBD para que pueda dar tratamiento a anomalías de diseño desde la configuración inicial automática.

Apéndices

Apéndice A. Descripción de la Base de Datos ATIS

ATIS (Air Travel Information Service) es una base de datos relacional que almacena información sobre vuelos, y es usada como BD de pruebas para procesamiento de lenguaje natural hablado y escrito. Este apéndice presenta un diagrama del esquema de BD.

El esquema de BD de ATIS describe las tablas y relaciones que existen entre tablas (Figura A.1).

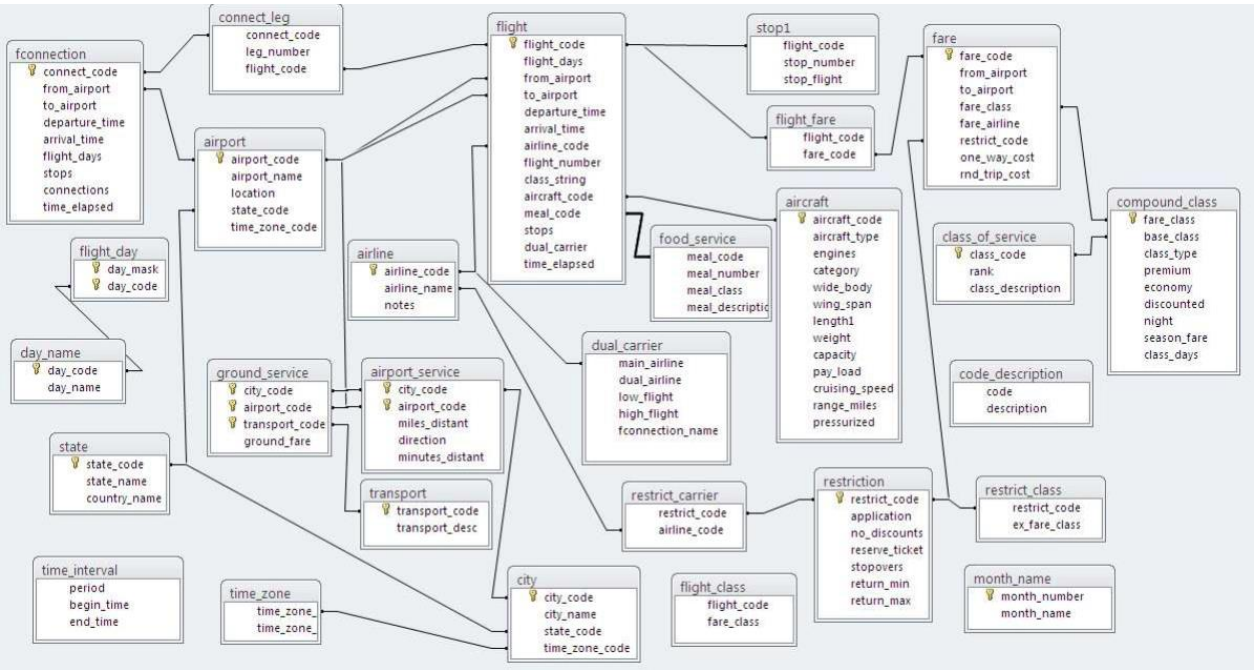


Figura A.1. Esquema de BD de ATIS

Apéndice B. Descripción de la Base de Datos Geobase

Geobase es una BD que fue presentada por Borland International en el año 1969 como una BD de ejemplo para su consulta mediante LN a través de una interfaz de lenguaje natural diseñada en Turbo Prolog.

Dicha base de datos contiene información sobre datos geográficos de los Estados Unidos de América.

La distribución de dicha BD con su respectiva ILN diseñada en Prolog consta de los siguientes archivos:

1. Archivo ***GEOBASE.PRO***

Contiene:
- Los predicados que se usan para realizar consultas por la ILN definida en *GEOBASE.LAN*.
- El esquema de la BD (tablas, relaciones, etc.).
- El script en prolog para acceder a la BD.

2. Archivo ***GEOBASE.DBA***

Contiene:
- Los datos de la BD.

3. Archivo ***GEOBASE.INC***

Contiene:
- Las cláusulas de soporte para los menús del sistema en LN.
- Mecanismo para evaluación de consultas para el sistema en LN.

4. Archivo ***GEOBASE.LAN***

Contiene:
- Expresiones en LN para el procesamiento del sistema en LN.

5. Archivo ***GEOBASE.HLP***

Contiene:
- Descripción sobre la BD.
- Ejemplos de consultas en LN que puede procesar el sistema en LN.
- Ayuda para usar el sistema en LN.

En la Figura B.1 se muestra el esquema de Geobase en su versión de Prolog.

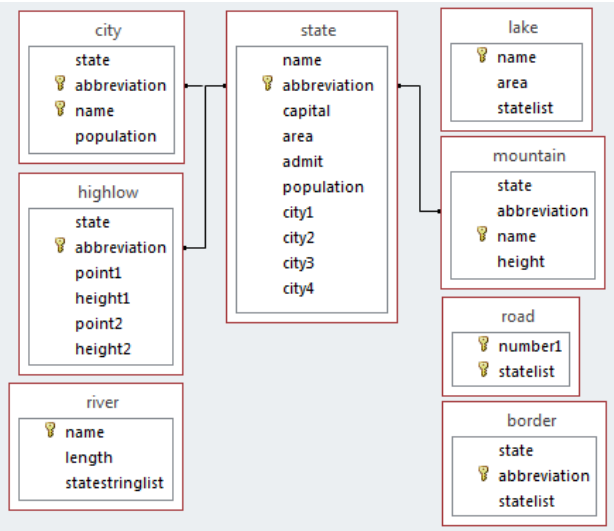


Figura B.1. Esquema de Geobase en Prolog

Aparte de la versión en Prolog, varios autores han adaptado esta BD a un esquema relacional con el fin de realizar pruebas con ésta. En la Figura B.2 se muestra la versión de Geobase usada por C-Phrase para realizar pruebas. De igual manera, en la Figura B.3 se muestra la versión de Geobase usada para realizar pruebas en la ILNBD desarrollada en el ITCM. Por último, en la Figura B.4 se muestra el esquema de Geobase usado por Mooney.

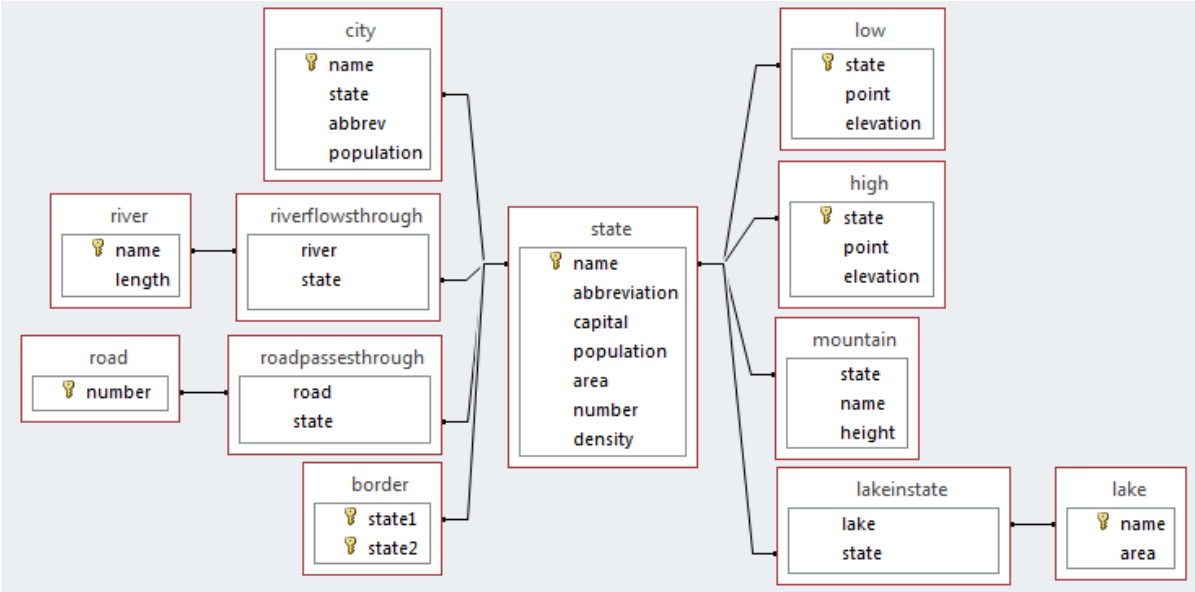


Figura B.2. Esquema de Geobase usado por C-Phrase

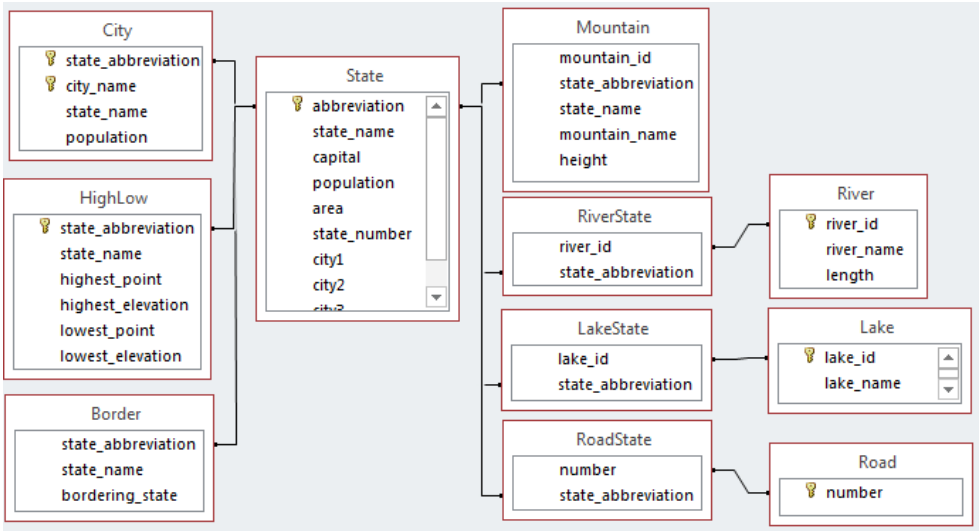


Figura B.3. Esquema de BD de Geobase usado por la ILNBD del ITCM

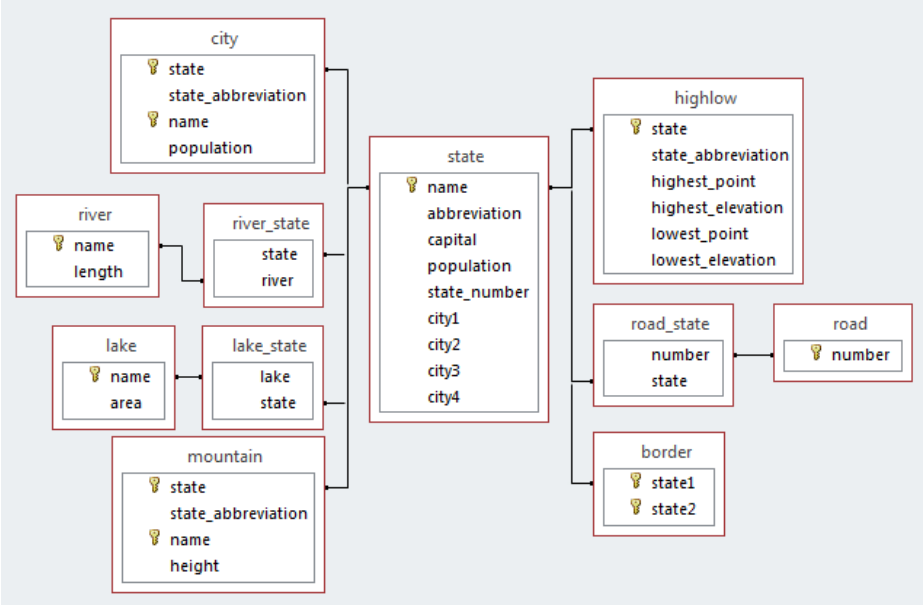


Figura B.4. Versión del esquema de BD de Geobase de Mooney

Apéndice C. Pruebas con ELF y C-Phrase con BDs con Anomalías de Diseño

Consulta 1. *¿Cuántos ríos tiene Colorado?*
How many rivers does Colorado have?

Al introducir la consulta anterior en la interfaz ELF haciendo uso de la BD Geobase con llaves primarias y foráneas debidamente definidas, ELF obtiene como resultado una respuesta correcta tal como se puede apreciar en la Figura C.1.

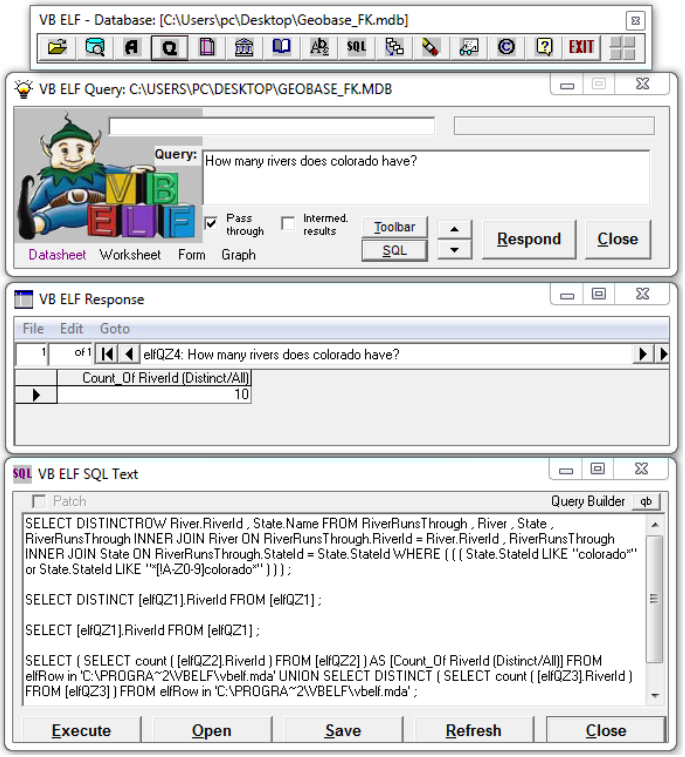


Figura C.1. Resultado obtenido por ELF con una BD con PKs y FKs

Sin embargo, al hacer uso de una versión de la BD Geobase sin llaves foráneas definidas, ELF detecta la omisión de llaves foráneas y alerta al usuario sobre este problema (véase Figura C.2).

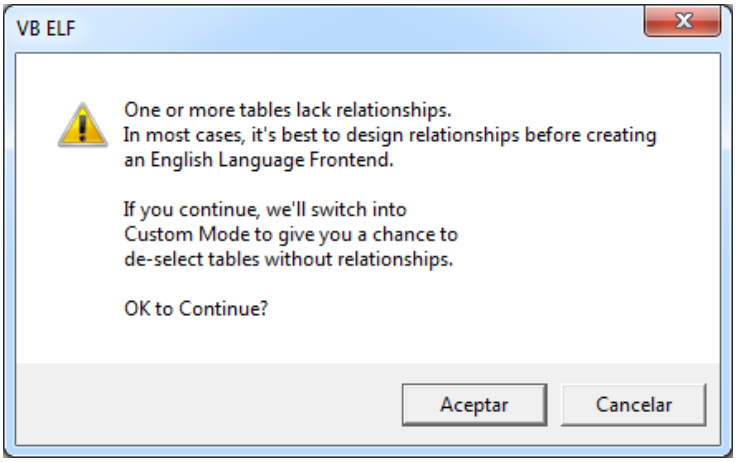


Figura C.2. Aviso de ELF sobre tablas sin llaves foráneas

Posteriormente, al ejecutar el análisis de la BD para crear el diccionario de información de la misma, ELF muestra una ventana (Figura C.3), la cual solicita al usuario seleccionar las tablas que éste desee tomar en cuenta al momento de realizar la traducción de consultas en LN.

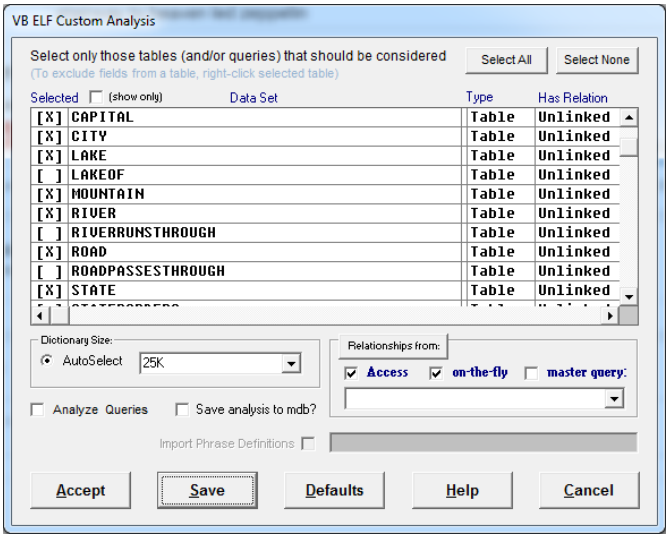


Figura C.3. Selección de tablas a considerar para la traducción

Una vez que el usuario haya definido el conjunto de tablas a considerar para la traducción de la consulta, ELF se encuentra listo para responder. Sin embargo, al introducir la consulta *How many rivers does Colorado have?*, la interfaz es incapaz de traducir la consulta, debido a que no reconoce algunas palabras presentes en la consulta tal como se muestra en la Figura C.4.

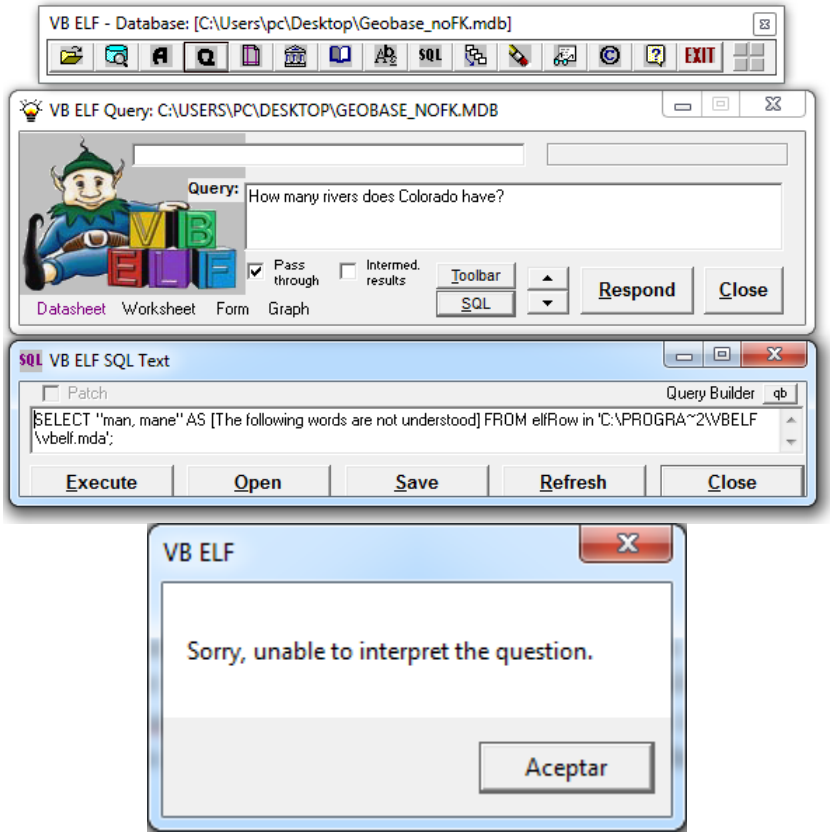


Figura C.4. Resultado obtenido por ELF con una BD sin FKs

Por lo anterior se puede apreciar que ELF no es capaz de traducir consultas a BDs con anomalías de diseño, en este caso, BDs con omisión de llaves foráneas.

Por otra parte, C-Phrase contesta correctamente la consulta 1.

Consulta 2. *Dame la población de la capital del estado de Alabama.*
Give me the population of the capital of the state of Alabama.

Para la consulta 2, ELF contesta incorrectamente, ya que el resultado que obtiene es la población de *Alabama* en lugar de la población de su capital (*Montgomery*). Esto se debe a que el esquema original de Geobase no tiene una llave foránea que relacione *state.capital* con *city.city_name*. Por lo tanto, no es posible construir la reunión entre las tablas *state* y *city* para obtener la población de *Montgomery*.

La respuesta obtenida por C-Phrase para la consulta 2 es errónea. Aunque detecta dos posibles traducciones para la consulta, las dos son incorrectas, porque obtiene la población de *Alabama*, en lugar de *Montgomery*, al igual que ELF.

Consulta 3. *Dame el estado con el punto más alto.*
Give me the state with the highest point.

Al ingresar la consulta anterior, ELF no es capaz de contestar correctamente, ya que la columna que se necesita conocer es *state.state_name*, y dicha columna se encuentra repetida en numerosas tablas: *mountain.state_name*, *border.state_name*, *highlow.state_name*, *city.state_name*.

Por tal motivo, ELF toma la columna *highlow.state_name* en lugar de *state.state_name*, lo cual ocasiona que el proceso de traducción no se efectúe correctamente.

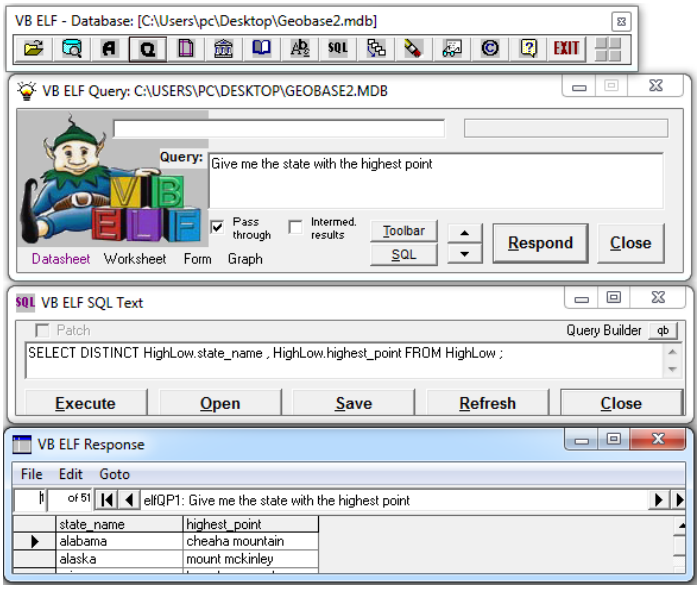


Figura C.5. Resultado obtenido por ELF para la consulta 3

La respuesta dada por C-Phrase para la consulta 3 es incorrecta, dado que ignora el hecho de que es necesario encontrar primero el punto más alto de todos los estados de la tabla *highlow*.

Consulta 4. *Dame la población de Illinois.*
Give me population of Illinois.

Para la consulta 4 (*Give me population of Illinois*), las respuestas obtenidas por ELF y C-Phrase son parcialmente correctas, porque hay dos alternativas para calcular la población: obtener la población almacenada en la tabla *state* para *Illinois*, o sumando las poblaciones de las ciudades almacenadas en la tabla *city* para el estado de *Illinois*. En este caso no está claro cuál alternativa es la correcta; esto depende de la BD y de su aplicación.

Consulta 5. *Nombra los ríos de Alabama.*
Name the rivers in Alabama.

Se utilizó una versión modificada de Geobase con llaves primarias sustitutas (Figura E.3). La respuesta obtenida por ELF es incorrecta, porque la cláusula Where incluye la tabla s*tateLoPoint*, y la información obtenida incluye información redundante.

La respuesta dada por C-Phrase para la consulta 5, aunque es correcta, tiene información redundante, debido a que incluye renglones duplicados de la tabla *state* (como se muestra en la Figura C.6).

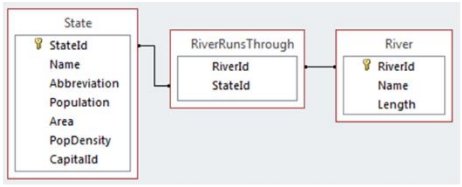


Tabla: state

StateId	Name	Abbreviatio	Population	Area	PopDensity	CapitalId
alabama	alabama	al	51700	3894000	0,013276836	montgomeryA
alabama2	alabama	al	51700	3894000	0,013276836	montgomeryA
alaska	alaska	ak	591000	401800	1,470881	juneauAk
arizona	arizona	az	114000	2718000	0,041942604	phoenixAz
arkansas	arkansas	ar	53200	2286000	0,023272092	littleRockAr

Tabla: riverrunsthrough

RiverId	StateId
tombigbee2	alabama
tennessee2	alabama
tombigbee2	alabama2
tennessee2	alabama2
colorado2	arizona
gila2	arizona

Figura C.6. Llaves primarias sustitutas y redundancia de datos en Geobase

Referencias

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

[Andrade, 1997] G. Andrade, *Supresión de Ambigüedades Léxicas en Galena mediante Métodos Estadísticos*, tesis de licenciatura, Universidade da Coruña, 1997.

[Androutsopoulos, 1995] I. Androutsopoulos, G. Ritchie y P. Thanisch, "Natural Language Interface to Database: An Introduction", *Journal of Natural Language Engineering*, pp. 29-81, 1995.

[Baik, 2019] C. Baik, H.V. Jagadish y Y. Li, "Bridging the Semantic Gap with SQL Query Logs in Natural Language Interfaces to Databases", *IEEE International Conference on Data Engineering (ICDE)*, 2019.

[Chu, 2004] M. Chu, *Blissful Data: Wisdom and Strategies for Providing Meaningful, Useful, and Accessible Data for All Employees*, American Management Association, 2004.

[Codd, 1970] E. F. Codd, *A Relational Model of Data for Large Shared Data Banks*, reporte de investigación, IBM Research Laboratory, San Jose, California, 1970.

[Codd, 1974] E. F. Codd, "Seven Steps to Rendezvous with the Casual User", *IFIP Working Conference Data Base Management*, pp. 179-200, 1974.

[Conlon, 2004] S. Conlon, J. Conlon y T. James, "The Economics of Natural Language Interfaces: Natural Language Processing Technology as a Scarce Resource", *Decision Support Systems*, vol.38, no. 1, pp. 141–159, oct. 2004.

[Databasedev, 2020] Databasedev, "Relational Database Table Design Tips", http://www.databasedev.co.uk/table_design_tips.html, 2020.

[Date, 2004] C. J. Date, *Introducción a los Sistemas de Bases de Datos*, 8va edición, Pearson Educación, 2004.

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

[Date, 2012] C. J. Date. *Database Design and Relational Theory*, O'Reilly Media, Inc., 2012.

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

[Elmasri, 1997] R. Elmasri y S. Navathe, *Fundamentos de Sistemas de Bases de Datos*, Addison-Wesley Iberoamerica, 1997.

[Hendrix, 1978] G. Hendrix, E. Sacerdoti, D. Sagalowicz y J. S. Locum, "Developing a Natural Language Interface to Complex Data", *ACM Transactions on Database Systems*, vol. 3, no. 2, pp. 105-147 1978.

[Howard, 2013] J. Howard y J. Tiret, "Seven Deadly Sins of Database Design, Embarcadero Developer Network", https://esj.com/articles/2010/02/16/database-design-sins.aspx, 2013.

[Kokare, 2015] R. Kokare y K. Wanjale, "A Natural Language Query Builder Interface for Structured Databases Using Dependency Parsing", *International Journal of Mathematical Sciences and Computing*, vol. 1, no. 4, pp. 11–20., doi:10.5815/ijmsc.2015.04.02, 2015.

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

[Li, 2014] F. Li y H.V. Jagadish, "Constructing an Interactive Natural Language Interface for Relational Databases", *Proc. of the VLBD Endowment*, vol. 8, No. 1, 2014.

[Liddy, 1998] E. Liddy, "Natural Language Processing for Information Retrieval and Knowledge Discovery", *Visualizing Subject Access for 21st Century Information Resources*, pp. 137-147, 1998.

[Linguistic Data Consortium, 1993] Linguistic Data Consortium, "The 2884 ATIS0 Speaker-dependent Training Prompts", https://catalog.ldc.upenn.edu/LDC93S4B-3, 1993.

[Marneffe, 2006] M. C. de Marneffe, B. MacCartney y C. D. Manning, "Generating Typed Dependency Parses from Phrase Structure Parses", *International Conference on Language Ressources and Evaluation*, pp. 449-454, 2006.

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

[Mfourga, 1997] N. Mfourga, "Extracting Entity-Relationship Schemas from Relational Databases: A Form-Driven Approach", *Proc. of the 4th Working Conference on Reverse Engineering*, IEEE, 1997.

[Minock, 2010] M. Minock, "C-phrase: A System for Building Robust Natural Language Interfaces to Databases", Data & Knowledge Engineering, vol. 69, no. 3, pp. 290-302, 2010.

[Mitkov, 2005] R. Mitkov, *The Oxford Handbook of Computational Linguistics*, Oxford University Press, p. 130, 2005.

[Mvumbi, 2016] T. Mvumbi, *Natural Language Interface to Relational Database: A Simplified Customization Approach*, Master thesis, University of Cape Town, Cape Town, South Africa, 2016.

[O'Shea,] J. D. O'Shea, K. Shabaz y K. A. Crockett, "Aneesah: A Conversational Natural Language Interface to Databases", *World Congress on Engineering 2015*, vol. 1, Londres, 2015.

[Pazos, 2013] R. A. Pazos, J. J. González, M. A. Aguirre, J. A. Martínez y H. J. Fraire, "Natural Language Interfaces to Databases: An Analysis of the State of the Art", *Recent Advances on Hybrid Intelligent Systems*, *Studies in Computational Intelligence*, *Springer Berlin Heidelberg*, no. 451, pp. 463-480, 2013.

[Pazos, 2016] R. A. Pazos, M. A. Aguirre, J. J. González, J. A. Martínez, J. Pérez y A. A. Verástegui, "Comparative Study on the Customization of Natural Language Interfaces to Databases", *SpringerPlus*, DOI: 10.1186/s40064-016-2164-y, 2016.

[Pazos, 2018] R. A. Pazos R., J. A. Martínez F., A. G. Aguirre L., M. A. Aguirre L., "Issues in Querying Databases with Design Anomalies Using Natural Language Interfaces", *Fuzzy Logic Augmentation of Neural and Optimization Algorithms: Theoretical Aspects and Real Applications. Studies in Computational Intelligence*, vol 749, pp. 461-473, 2018.

[Pedro-de-Jesus, 1999] M. L. Pedro de Jesus y P.M.A. Sousa, "Selection of Reverse Engineering Methods for Relational Databases", *Proc. European Conference on Software Maintenance and Reengineering*, 1999.

[Pivert, 2010] O. Pivert y H. Prade, "Handling Dirty Databases: From User Warning to Data Cleaning Towards an Interactive Approach", *Fourth International Conference on Scalable Uncertainty Management*, Francia, 2010.

[Popescu, 2004] A. M. Popescu, A. Armanasu, O. Etzioni, D. Ko y A. Yates, "Modern Natural Language Interfaces to Databases: Composing Statistical Parsing with Semantic Tractability", *Proc. of the 20th International Conference on Computational Linguistics*, pp. 141-147, 2004.

[Quest, 2020] Quest Software Inc., "Toad Data Modeler. Easy-to-use, Multi-platform Database Modeling", https://www.toadworld.com/products/toad-data-modeler, 2020.

[Polanco, 2000] D. F. Polanco, *Evaluación y Mejora de un Sistema Automático de Análisis Sintagmático*, tesis de licenciatura, Ingeniería Electrónica de la Escuela Técnica Superior de Ingenieros de Telecomunicación de Madrid de la Universidad Politécnica de Madrid, España, 2000.

[Rob, 2011] P. Rob y C. Coronel, *Database Systems: Design, Implementation, and Management*, 9a edición, Course Technology, 2011.

[Sidorov, 2020] G. Sidorov, R. A. Pazos, J. A. Martínez, J. M. Carpio y A. G. Aguirre, "Configuration Module for Treating Design Anomalies in Databases for a Natural Language Interface to Databases", *Intuitionistic and Type-2 Fuzzy Logic Enhancements in Neural and Optimization Algorithms: Theory and Applications*, vol. 862, Springer, 2020.

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

[Sujatha, 2012] B. Sujatha, S. Viswanadha y H. Shasiya, "A Study of the Various Architectures for Natural Language Interfaces to DBs", *International Journal of Computer Science and Network (IJCSN)*, 2012.

[Sukthankar, 2017] N. Sukthankar, S. Maharnawar, P. Deshmukh, Y. Haribhakta y V. Kamble, "nQuery - A Natural Language Statement to SQL Query Generator", *Proc. of the 55th Annual Meeting of the Association for Computational Linguistics*, Student Research Workshop, pp. 17-23, 2017.

[Teorey, 2011] T. Teorey, S. Lighstone, T. Nadeau y H.V. Jagadish, *Database Modeling and Design*, Fifth Edition, Elsevier Inc., 2011.

[Utama, 2018] P. Utama, N. Weir, F. Basık, C. Binnig, U. Cetintemel, B. Hättasch y A. Usta, "DBPal: An End-to-end Neural Natural Language Interface for Databases", https://arxiv.org/abs/1804.00401, 2018.

[Visual-Paradigm, 2020] Visual Paradigm, "Visual Paradigm Features", https://www.visual-paradigm.com/features/, 2020.

[Vertabelo, 2020] Vertabelo SA, "Vertabelo Database Modeler" https://www.vertabelo.com/features, 2020.

[Wang, 2019] W. Wang, “A Cross-domain Natural Language Interface to Databases Using Adversarial Text Method”, *Proc. VLDB 2019 PhD Workshop*, 2019.

[Wise Coders, 2020] Wise Coders GmbH, “DBSchema: The Best Database Designer & Admin GUI Tool”, 2020.

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

[Xu,2019] B. Xu, R. Cai, Z. Zhang, X. Yang, Z. Hao, Z. Li y Z. Liang, “NADAQ: Natural Language Database Querying Based on Deep Learning”, *IEEE Access*, vol. 7, pp. 35012-35017, doi:10.1109/access.2019.2904720, 2019.