Universidad de las Ciencias Informáticas

Facultad 2

TRABAJO DE DIPLOMA PARA OPTAR POR EL TÍTULO DE INGENIERO EN CIENCIAS INFORMÁTICAS

Evaluador AR, herramienta para la Evaluación Automatizada de expresiones del lenguaje Álgebra Relacional

Autor(es):

Lidiagnis Fonseca Pérez

Adnier Castellanos Puentes

Tutor(es):

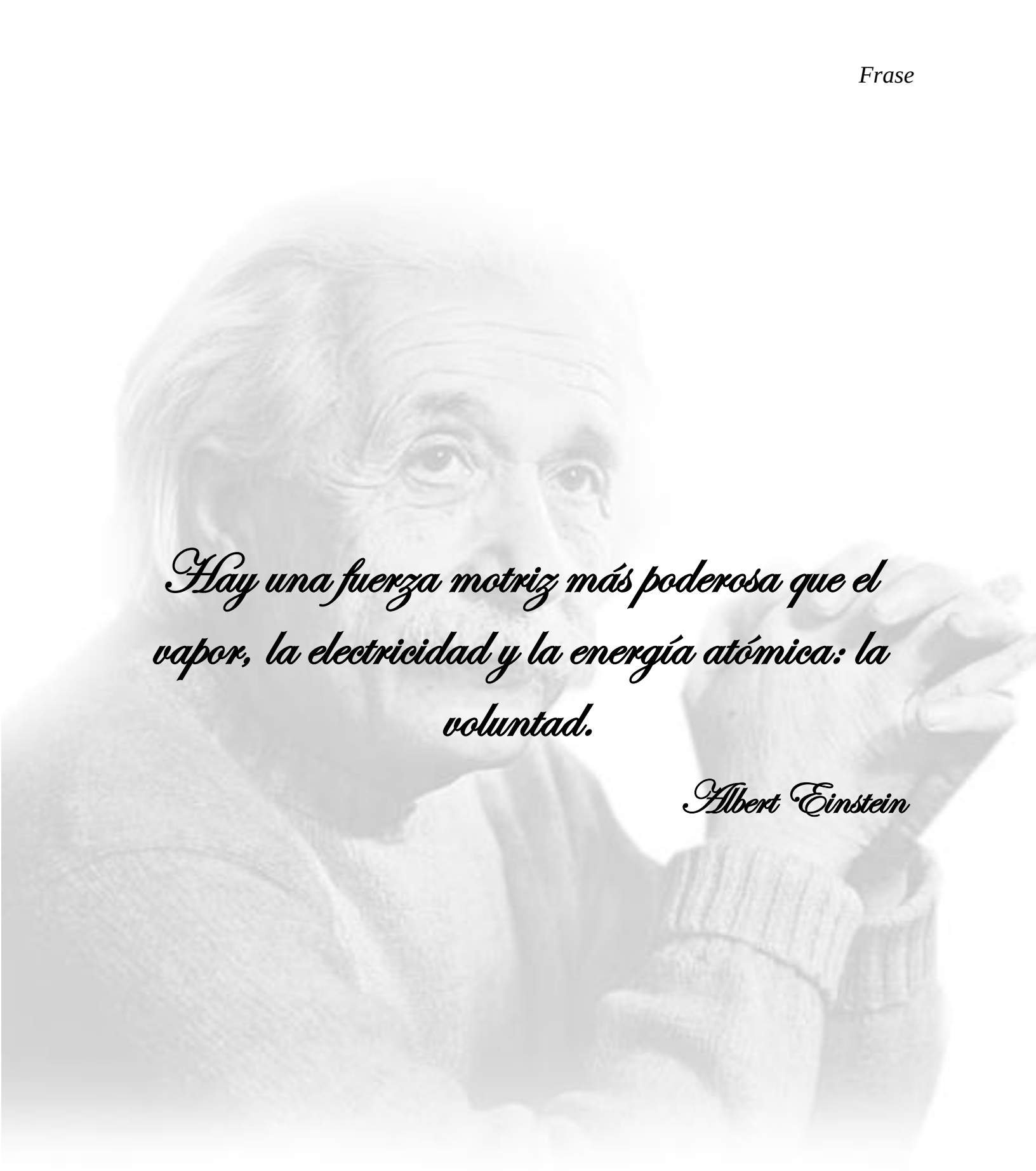
Lic. Yamilka Gómez León.

Msc. Ailec Granda Dihigo

Dr.C. Edistio Yoel Verdecia Martínez

La Habana, Junio 2013

"Año 55 de la Revolución"



Hay una fuerza motriz más poderosa que el vapor, la electricidad y la energía atómica: la voluntad.

Albert Einstein

DECLARACIÓN DE AUTORÍA

Declaro que somos los únicos autores de este trabajo y autorizamos a la Universidad de las Ciencias Informáticas (UCI) y a la Facultad 2 para que hagan el uso que estimen pertinente de este trabajo.

Para que así conste firmamos la presente a los ____ días del mes de junio del 2013.

Autores

Lidiagnis Fonseca Pérez	Adnier Castellanos Puentes

Tutores

Lic. Yamilka Gómez León	Msc. Ailec Granda Dihigo

Dr.C. Edistio Yoel Verdecia Martínez

AGRADECIMIENTOS

Lidiagnis:

A mi compañero de tesis, por ser tan preocupado, y estar presente en cada momento que lo necesité.

A mi Tía Tere, por permitirme ser su sobrina.

A mis primos Yaniuska, Yane, Mailen La Mima, Luisi y al resto de mi familia

A mi novio Janier, por tener tanta paciencia conmigo, y dedicarme tanto amor y cariño en estos tormentosos días, por mostrarme que siempre existe una luz al final del túnel….

A Yudi, el Negro, Ivette, Yoli, Nanita y el Kinde, gracias por hacerme parte de su familia…

A mis amigos Damaysis, Dayna, Yurita, Moreu, Sellés, al insoportable de Boris (con mucho cariño).

A todas mis amistades que han estado conmigo compartiendo todos estos maravillosos años: Grabiel, Leydis, Danima, Kenia, Osmar, Alexei, Dago, Henrry, Orquidia, Karel, Alejandro y a todos en general. A las amistades de Jiguaní Danielito, Jose y demás.

A los profesores que de una forma u otra han contribuido en esta tesis y en mi formación: Ariel, Adrian Maranje, Jorge Amado, Antonio Hernández, Vladimir, Damián

Quiero agradecer especialmente a los tutores por estar pendientes siempre de nuestro trabajo y permitir que saliera bien. A Yamilka, Ailec y a Edistio

Adnier:

A mi compañera de tesis por haber aceptado este reto junto a mí.

A Alexey por las brillantes ideas aportadas al trabajo.

A mis tutores Yamilka, Ailec y Edistio por ser guía fundamental en la realización de este trabajo.

A mis compañeros de estudio, Alain, Jany, Reinier, Yenileidis y en especial a Andrés por toda la ayuda brindada en mis estudios.

A todas las personas con las que compartí durante mis estudios universitarios Layda, Yanersy, Thais, Frank, Yusleidis, Rosalía, Eduardo, Asiel, Mailet, Karel,Orquidia.

A mis profesores de la Facultad No. 2 Imiris Rovira, Abel Vázquez, Vladimir Milián, Lester Rodríguez, Alberto, Iris Reina, Livan Miranda, Ariel Díaz, Adrián Maranje, Damián Ilizastegui.

A mis padres, mi hermana, mi abuelo, mis tíos Tomasa y Jorge…. gracias…. parte de lo que soy se lo debo a ustedes.

DEDICATORIA

Lidiagnis:

La primera persona a quien va dedicada esta tesis es a mi abuela Lidia (Maja), que ya no se encuentra presente físicamente conmigo, pero sé que desde algún lugar me está observando en este momento y sintiéndose orgullosa de la persona en quien me he convertido. La persona que fue mi madre, mi padre, y a quién le debo lo que actualmente soy, es gracias a ella.

A mis abuelos Monguito, Adriano y Adela.

A mi mamá y a mi papá, por ser mi apoyo incondicional en toda mi vida, por estar presente conmigo día a día, soportando mis malcriadeces y tratando de cumplir cada uno de mis sueños. Por confiar en mí sin importar nada, y más que nada, por darme la vida.

A tío Edel, por ser mi segundo padre y apoyarme y cuidarme como si fuera en realidad su hija. Alguien a quien en verdad admiro muchísimo, y hoy, es uno de mis ejemplos a seguir.

A mis queridos tíos: tío Lalo, tío Mami, tío Luis y tío Jorge, por ayudarme en estos 5 años de carrera, por siempre estar pendiente de mí, por sentirse orgullosos de mí.

A mi hermano el niño, por ser una de mis mayores alegrías en esta vida, por dar colorido a todos mis días cuando estoy junto a él.

A mis hermanos Yordan y Yordenis y Leander y Aniel, que aunque estos dos últimos son primos, los considero mis hermanos, por preocuparse siempre por mí.

A mis sobrinos bellos Dayron y la hembrita que viene en camino, que va a ser un rayito de sol en mi vida.

Adnier:

A Inés María Puentes Valdivia, José Enrique Castellanos Delgado y Adnierys Castellanos Puentes, por apoyarme en cada paso dado y ser el sustento de mi vida.

RESUMEN

Como parte del proceso de formación de los estudiantes en la Universidad de las Ciencias Informáticas, se imparten las asignaturas Sistemas de Bases de Datos 1 y Sistemas de Bases de Datos 2, las cuales constituyen un elemento imprescindible para la formación de los ingenieros y para el desarrollo de un software. Dentro de los contenidos que se imparten en la asignatura Sistemas de Bases de Datos 1, se incluye el lenguaje de consulta Álgebra Relacional, que está contenido dentro de la parte manipulativa del Modelo Relacional. La experiencia docente de los profesores en su trabajo con los estudiantes, y los resultados de las evaluaciones sistemáticas, parciales y finales en este tema, evidencia que para los estudiantes resulta difícil reconocer cuándo las consultas expresadas en papel son correctas y responden a los requisitos de información planteados, pues estas pueden formularse de diversas formas y estar correctas. Esta dificultad está determinada porque el Álgebra Relacional es un lenguaje procedural, por lo cual no sólo se necesita saber qué información se precisa, sino las acciones que hay que realizar para obtenerla. Actualmente existen herramientas que trabajan con este lenguaje, pero todas utilizan la notación tradicional, que no es la utilizada en la Universidad. Con el propósito de apoyar el proceso de enseñanza aprendizaje del Álgebra Relacional, se desarrolló una herramienta informática que permite la evaluación automatizada de este lenguaje, utilizando para ello la traducción del lenguaje Álgebra Relacional al SQL. La herramienta garantiza una retroalimentación inmediata de los errores cometidos.

Palabras claves: Álgebra Relacional, evaluación automatizada, retroalimentación, traductor.

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

INTRODUCCIÓN

El mundo experimenta un vertiginoso avance en el campo de las Tecnologías de la Información y las Comunicaciones (TIC). Este avance constituye un motor impulsor en la mejora de la calidad de vida y los servicios, provocando continuas transformaciones en las estructuras económicas, sociales y culturales, e incidiendo en casi todos los aspectos de la vida. El impacto positivo que provoca el uso de las TIC en cualquier sector, hace cada vez más difícil actuar eficientemente prescindiendo de su uso. En el campo educativo las TIC constituyen medios e instrumentos que facilitan el Proceso de Enseñanza Aprendizaje (PEA) y el desarrollo de habilidades. (1)

Uno de los retos más importantes que tiene la sociedad cubana, es diseñar políticas educativas que permitan aprovechar todos los recursos disponibles, e integrar las TIC dentro de los procesos educativos. Estas transformaciones se realizan con el propósito de contar con una sólida estructura educativa que permita adaptarse a los nuevos y rápidos cambios que conlleva la sociedad global. (2) En la actualidad existe la posibilidad de utilizar diversidad de medios tecnológicos; los cuales ayudan a captar la atención de los estudiantes, reducir el tiempo de la asimilación y liberar al profesor de las tareas repetitivas. (3)

Debido a la utilización de las TIC en la educación, las condiciones para la realización del PEA han estado inmersas en un proceso de transformación constante. (4) Uno de los retos lo constituye el desarrollo de la Tecnología Educativa (TE), la cual propicia procesos de enseñanza más interactivos que favorecen la retroalimentación profesor-alumno, y genera nuevos entornos de aprendizaje. (5)

El concepto TE ha sido definido desde diversos puntos de vista y con diferentes alcances. Los criterios de los autores en los últimos años han ido variando y no hay un verdadero consenso acerca del término. De manera general se define la TE como: un conjunto de procedimientos, la aplicación de conocimientos científicos, un campo de teoría y práctica, un proceso complejo, entre otros. (6)

Con la utilización de la TE se propicia en los estudiantes el desarrollo de habilidades como la planificación del aprendizaje y la interactividad. (7) Estas habilidades deben adquirirlas apoyándose en computadoras, multimedia, televisión, Internet, diapositivas, documentales, laboratorios informáticos, laboratorios virtuales, Entornos Virtuales de Enseñanza Aprendizaje (EVEA), Repositorio de Objetos de Aprendizaje[1], entre otros.

[1] Bases de datos o Catálogos creados para ser utilizados en un proceso de enseñanza.

Uno de los niveles educacionales más beneficiados actualmente con las TE y que más requiere su utilización es la educación superior, ya que tiene grandes retos ante una sociedad en constante transformación. Cuba no está exenta de estos retos, por lo que el gobierno ha puesto a disposición de las universidades las tecnologías necesarias para fomentar el pleno desarrollo de los educandos. Esto ha propiciado diversos beneficios que han contribuido a la mejora del PEA en este nivel educacional.

La Universidad de las Ciencias Informáticas (UCI) es una de las universidades cubanas que ha sido beneficiada con esta voluntad del gobierno, al contar con computadoras en las aulas, estudio de televisión, conexión a internet y una red que conecta todas sus aulas, apartamentos, laboratorios y oficinas. La UCI utilizando estos recursos ha obtenido importantes logros no solo en el campo de la producción de software. Contar con esta infraestructura ha permitido durante el proceso docente utilizar diferentes aplicaciones *e-learning*[2], dentro de las que se encuentran el Repositorio de Objetos de Aprendizaje RHODA, la Herramienta de Autor para la Creación de Objetos de Aprendizaje de forma colaborativa CRODA, y el Entorno Virtual de Enseñanza Aprendizaje (EVEA-UCI), una personalización de la plataforma Moodle. (8)

El EVEA-UCI es una plataforma para la teleformación donde se ubican materiales, se abren foros y se realizan evaluaciones como apoyo al PEA de las distintas asignaturas. Dentro de sus ventajas se encuentran que los profesores pueden diseñar diversas actividades e interactuar con los estudiantes mediante los chats, foros y la revisión de las actividades a partir de poner comentarios en los archivos subidos a la plataforma por los estudiantes.

La UCI se dedica a la formación de Ingenieros en Ciencias Informáticas (ICI). Los modos de actuación de este profesional están asociados a los roles que debe ejecutar en las diferentes etapas del ciclo de vida de un *software*, de los cuales se puede mencionar Diseñador de bases de datos y Administrador de bases de datos. (9) El profesional que se forma en la Universidad debe estar preparado para integrarse a equipos de desarrollo de software.

Dentro del desarrollo de los sistemas de información un elemento importante lo constituye garantizar la persistencia de la información, para lograr ello es importante la utilización de las bases de datos. Las bases de datos ocupan un lugar determinante en cualquier área del quehacer humano, comercial, y tecnológico, ya que tienen la capacidad de resolver sus necesidades de información concretas (10) Estas son importantes no solo para el almacenamiento de grandes cantidades de información, también para la recuperación rápida, la

[2] e-learning: permite la interacción del usuario con el material mediante la utilización de diversas herramientas informáticas.

organización y reorganización de la información.

Para sustentar los conocimientos necesarios de los roles antes mencionados y otros, la carrera se estructura en diferentes disciplinas, dentro de las cuales se encuentra la disciplina Ingeniería y Gestión de Software (IGSW). Esta disciplina juega un papel importante en la formación de los futuros ingenieros, al aportar elementos que estos necesitan para desarrollar los roles en los proyectos productivos que se desarrollan en la Universidad y los que desempeñarán en su futura vida laboral. (11)

El estudio de la disciplina IGSW se comienza en segundo año, con la asignatura Sistemas de Bases de Datos 1 (SBD1), seguido de Sistemas de Bases de Datos 2 (SBD2). Estas asignaturas persiguen preparar al estudiante en los conceptos básicos asociados a los Sistemas de Gestión de Bases de datos, donde se incluyen los temas asociados a la utilización de las bases de datos y sus fundamentos. (12)

Entre los temas impartidos en la asignatura SBD1 se encuentra el de Diseño de Bases de Datos Relacionales, que incluye el Modelo Entidad Relación, Patrones de Diseño, etc. Otra temática es la vinculada al Modelo Relacional (MR), que es uno de los elementos más utilizados en la actualidad por los Sistemas Gestores de Bases de Datos (SGBD). El MR es un modelo teórico completo para la representación y manipulación de bases de datos; la parte manipulativa está conformada por los lenguajes de manipulación de datos Cálculo Relacional (CR) y Álgebra Relacional (AR). (13)

Como parte del PEA del AR y del CR, al estudiante se le aplican evaluaciones para comprobar el cumplimiento de los objetivos previamente trazados. Sin embargo este proceso es complejo pues los ejercicios de tipo formulación de expresiones en estos lenguajes pueden tener más de una solución, este proceso hace que los estudiantes deban esperar la revisión que realiza el profesor.

La evaluación es un proceso necesario e inherente al hombre desde hace muchos años. Su definición se ha ido conformando históricamente, dependiendo en gran medida de visiones ideológicas y culturales, opciones políticas, etc. Constituye la acción de apreciar, valorar, fijar el valor de una cosa hecho o fenómeno. Está conformada por procesos de recogida, análisis e interpretación de información que permiten llegar a una decisión que favorezca o no el objeto evaluado cuando se compara con referencias o criterios anteriores. (14)

La automatización del proceso de evaluación es un aspecto deseado y que va acorde con los avances informáticos y las necesidades de las instituciones por alcanzar más altos niveles educativos. Existen actualmente un conjunto de entornos, herramientas y recursos que

permiten la realización de una evaluación automatizada del proceso de aprendizaje de los estudiantes, sin la presencia física del alumno y la revisión exclusiva del profesor, incluyéndose Moodle entre ellas. Sin embargo en temas tan específicos como el AR y el CR se hace necesario desarrollar herramientas que permitan automatizar algunos aspectos de la evaluación, aun a pesar de la existencia de diferentes herramientas que lo hacen.

El AR es un lenguaje de datos que propone un conjunto de operaciones que describen paso a paso cómo automatizar una necesidad de información sobre las relaciones, tal y como estas son definidas en el MR. Este lenguaje es de tipo procedimental, los pasos que forman la consulta describen un procedimiento. Las operaciones que se proponen en el AR se utilizan como una representación intermedia de una consulta a una base de datos y debido a sus propiedades algebraicas, sirven para obtener una versión más optimizada y eficiente de dicha consulta. (15)

El AR es un lenguaje abstracto, permite entender el MR de bases de datos desde la perspectiva matemática, donde una misma consulta se puede formular a partir de diferentes expresiones. Las características mencionadas traen consigo dificultades para su asimilación por parte de los estudiantes, sobre todo en la formulación y verificación de expresiones.

En tareas como la formulación de expresiones en AR el EVEA presenta una dificultad, no ofrece una retroalimentación, pues el entorno no está preparado para ofrecer una retroalimentación que le permita al estudiante observar sus errores cometidos y los pueda corregir. El EVEA contiene además diferentes medios a utilizar en el PEA de otros temas, y no para el AR, como son los Cuestionarios, pero estos sólo permiten trabajar con elementos teóricos y desarrollar las habilidades relacionadas con el tema a un nivel reproductivo, no permite aplicar los contenidos, que es lo que deben lograr los estudiantes.

Una encuesta aplicada a profesores con cuatro (4) o más años de experiencia impartiendo las asignaturas de Bases de Datos, sobre el uso de herramientas de evaluación automatizada en estas asignaturas, arrojó como resultado la necesidad de realizar aplicaciones para el PEA de las asignaturas BD. El tema AR fue uno de los más mencionados por los encuestados para el desarrollo de herramientas que faciliten el estudio y la preparación individual de los estudiantes, a partir de la automatización de la evaluación de las respuestas dadas por los estudiantes.

De una muestra de 17 profesores, el 76.4% (13 profesores), no conocen la existencia de herramientas para el PEA de las BD, e igualmente reconocen la necesidad de crear herramientas específicamente para el tema de AR. De 17 profesores encuestados el 82% (14 profesores), afirman que la planificación de la asignatura SBD1, no es tiempo suficiente para

que los estudiantes desarrollen las habilidades necesarias y asimilen dicho contenido, ya que sólo se dedican 2 frecuencias para su impartición.

Otro elemento es que a partir de su experiencia los profesores han identificado que para los estudiantes es difícil conocer cuándo las consultas expresadas en el papel son correctas y responden a los requisitos de información planteados. También ellos constatan que no existen vías para que los estudiantes puedan saber cuánto han aprendido, cuándo escriben una expresión bien o mal, y qué errores cometen cuando escriben alguna expresión.

Todos estos elementos inciden negativamente en el resultado de las evaluaciones sistemáticas, en las pruebas parciales y las pruebas finales. En entrevistas realizadas a los profesores se confirman cuáles son las deficiencias presentadas por los estudiantes.

Teniendo en cuenta la situación problemática descrita, se identificó como **problema a resolver** *¿Cómo contribuir a la autoevaluación de los contenidos del Álgebra Relacional, con el apoyo de las Tecnologías de la Información y las Comunicaciones?*

El **objeto de estudio** se encuentra enmarcado en la *construcción de herramientas informáticas para la evaluación automatizada de los contenidos relacionados con las bases de datos*. El **campo de acción** está delimitado en *la construcción de herramientas informáticas para la evaluación automatizada de expresiones del Álgebra Relacional*.

Se define como **objetivo general** *desarrollar una herramienta informática para la evaluación automatizada de expresiones del Álgebra Relacional*.

Para dar solución al problema planteado y cumplir con el objetivo general, este se desglosó en los siguientes **objetivos específicos:**

- Determinar las bases teóricas que sustentan la construcción de herramientas informáticas para la evaluación automatizada en las asignaturas de bases de datos.
- Realizar un estudio del estado del arte de las herramientas informáticas existentes para la evaluación automatizada de expresiones del Álgebra Relacional.
- Definir las herramientas, lenguajes, tecnologías de modelado y programación, así como la metodología de desarrollo de *software* a utilizar para la construcción de la Herramienta para la Evaluación Automatizada de expresiones del Álgebra Relacional (HEA-AR), que apoye el PEA del contenido AR a los estudiantes de la UCI.
- Elaborar las actividades que propone la metodología de desarrollo de software seleccionada.
- Implementar la herramienta modelada, y realizar pruebas para garantizar su correcto

funcionamiento.

Los métodos utilizados a lo largo de la investigación fueron:

Como **métodos teóricos:**

- Análisis y Síntesis: Se examinaron bibliografías correspondientes a los contenidos Bases de Datos y Compiladores, con el fin de recolectar y utilizar información que posibiliten la realización de un sistema para la evaluación automatizada de expresiones del lenguaje Álgebra Relacional.
- Histórico lógico: Se utilizó para la confección del estado del arte de las herramientas similares a la propuesta de solución que existen en el mundo, y para las herramientas y tecnologías que se utilizaron para la realización de la aplicación.

Como **métodos empíricos**:

- Análisis Documental: Se utilizó durante la revisión de la bibliografía relacionada con el desarrollo se la aplicación.
- Observación: Se utilizó para adquirir conocimientos sobre el funcionamiento, ventajas y desventajas de otras aplicaciones educativas similares a la que se desarrollará.
- Encuesta: Se realizó una encuesta a los profesores de Bases de Datos de las diferentes facultades de la UCI, para verificar la necesidad de crear herramientas para el lenguaje Álgebra Relacional.
- Entrevista: Es de utilidad para recoger opiniones de los profesores de Sistemas de Bases de Datos 1 en cuanto a las deficiencias observadas en los estudiantes en la asignatura, específicamente en Álgebra Relacional.

Como métodos estadísticos:

- Estadística descriptiva: Se usa para el procesamiento de los datos obtenidos de las encuestas.
- El documento está estructurado en introducción, tres capítulos, conclusiones, recomendaciones y bibliografía. En el **Capítulo 1, Evaluación automatizada de expresiones del Álgebra Relacional, fundamentos teóricos y estado actual***,* se muestra un estudio detallado de la situación actual de las herramientas informáticas para la automatización de la evaluación en el contexto nacional e internacional enfocado en un mejor entendimiento de la solución. También se incluye el análisis de los conceptos relacionados con el tema, la justificación de las herramientas, la metodología y las tecnologías a utilizar para el desarrollo de la aplicación, así como la descripción de

la propuesta de solución. En el **Capítulo 2, Análisis y diseño de la herramienta para la evaluación automatizada de expresiones del Álgebra Relacional**, se incluyen las características del sistema, la especificación de los Requisitos de Software. Se realizan las actividades de análisis y diseño que propone la metodología de desarrollo. También se muestra el prototipo de interfaz de usuario de la herramienta. En el **Capítulo 3, Implementación y pruebas de la HEA-AR**, se describe todo el proceso de implementación del sistema. Se realizan las Tareas de Ingeniería y las Pruebas Unitarias y de Aceptación para la herramienta.

CAPÍTULO 1: Evaluación automatizada de expresiones del Álgebra Relacional, fundamentos teóricos y estado actual

Este capítulo tiene como objetivo abordar los temas relacionados con el uso de las TIC en la evaluación automatizada, así como su contextualización en la UCI y la evaluación automatizada en Bases de Datos. También se analizan conceptos fundamentales asociados a la evaluación automatizada, el Álgebra Relacional y la construcción de traductores entre lenguajes, así como la propuesta de solución y la metodología de desarrollo. Se presentan los resultados del análisis las herramientas existentes en el ámbito nacional e internacional, y de las herramientas, metodología y otros elementos a utilizar para el desarrollo de la aplicación.

1.1. La enseñanza de las bases de datos y el uso de las Tecnologías de la Información y las Comunicaciones en la evaluación automatizada.

Las TIC han tenido un gran impacto en la enseñanza y el proceso de aprendizaje. A raíz de esto ha surgido la necesidad de reflexionar acerca de las posibilidades de integrarlas en los PEA para mejorar la calidad de la educación.

Las nuevas tecnologías pueden emplearse en el sistema educativo de tres maneras distintas: como objeto de aprendizaje, como medio para aprender y como apoyo al aprendizaje. Como objeto de aprendizaje las TIC permiten a los estudiantes familiarizarse con el ordenador y adquirir las competencias necesarias para hacer del mismo un instrumento útil a lo largo de los estudios. (16)

Las tecnologías son utilizadas como un medio de aprendizaje cuando constituye una herramienta al servicio de la formación a distancia, no presencial y del autoaprendizaje; o son ejercicios de repetición, cursos en línea a través de Internet, de videoconferencia, etc. (16)

Como apoyo al aprendizaje, se encuentran integradas en el proceso de aprendizaje, tienen su sitio en el aula, responden a unas necesidades de formación más proactivas y son empleadas de forma cotidiana. (16) Un ejemplo de lo mencionado anteriormente, es el uso de las TIC en el proceso de evaluación de los estudiantes. Estas constituyen un elemento diferenciador respecto a las prácticas evaluativas que hasta ahora se vienen realizando. Esto debe estar relacionado además con el desarrollo de las habilidades vinculadas a los modos de actuación de los profesionales que se forman. Las TIC pueden ayudar en este objetivo de integrar la evaluación con la preparación y el desarrollo de las clases.

Es objetivo del presente epígrafe realizar un análisis de la importancia de los contenidos

relacionados con las bases de datos para los futuros Ingenieros en Ciencias Informáticas y la utilización de las TIC en el proceso de evaluación.

Breve caracterización del PEA de las bases de datos con apoyo en las TIC en la UCI

Dentro de los contenidos más importantes en la carrera de Ingeniería en Ciencias Informáticas están los relacionados con las bases de datos, no sólo para garantizar la persistencia de los datos, sino para lograr su manipulación, lo cual constituye uno de los temas más complejos**.** Dentro de los contenidos que se imparten en la asignatura SBD1 en la UCI está el lenguaje Álgebra Relacional, el cual se incluye dentro de la parte manipulativa del Modelo Relacional.

El AR incluye un conjunto de operadores que se utilizan para manipular las tuplas de las relaciones o tablas en el MR. Cada operador toma una o dos relaciones como entrada y produce una nueva relación como salida. Inicialmente se definieron ocho operadores, cuatro de ellos provienen de la teoría de conjunto (Unión, Intersección, Diferencia y Producto Cartesiano) y el resto son operaciones relacionales especiales (Selección, Proyección, Concatenación y División).

A continuación se definen y ejemplifica cada uno de ellos (17)

Sean A y B dos relaciones distintas, se define entonces como operadores:

- **Unión:** A UNION B es el conjunto de tuplas que pertenecen A o a B (o a ambos). Es una operación asociativa donde A y B tienen el mismo grado y cada uno de los atributos está definido sobre el mismo dominio. En la relación final se eliminan las tuplas repetidas.
- **Intersección:** A INTERSECT B es el conjunto de todas las tuplas que pertenecen A y a B. Es una operación asociativa, donde A y B tienen el mismo grado y cada uno de los atributos está definido sobre el mismo dominio.
- **Diferencia**: Su notación es A MINUS B, es el conjunto de todas las tuplas que pertenecen A y no pertenecen a B. No es una operación asociativa pues el orden varía la solución obtenida. Necesariamente A y B tienen el mismo grado y cada uno de los atributos está definido sobre el mismo dominio.
- **Producto cartesiano:** Su notación es A TIMES B, es el conjunto de tuplas t tales que t es la concatenación de una tupla a ϵ A con todas las tupla b ϵ B. No necesariamente A y B poseen el mismo grado ni tienen que están definidos en el mismo dominio los atributos.
- **Selección:** Su notación es A WHERE condición. La operación selección sobre una relación recupera el subconjunto de tuplas que cumplan con cierta condición. La

condición es definida mediante una expresión lógica, que representa a un predicado. En la relación final se eliminan las tuplas repetidas.

- **Proyección:** Su notación es A {a1, a2,…, an}. Se utiliza para seleccionar atributos específicos de una relación, o para reordenar los mismos en caso que sea necesario. No se pueden especificar dos veces los mismos atributos y en caso de que se proyecten todos los atributos de la misma relación, sería la proyección identidad. En la relación final se eliminan las tuplas repetidas.
- **Concatenación:** Su notación es A JOIN B. Se utiliza para unir dos relaciones bajo la condición de que el último atributo de A debe coincidir con el primer atributo de B, conocido también como join natural. En la relación final se eliminan las tuplas repetidas.
- **División:** Su notación es A DIVIDE BY B. Es utilizada para dividir dos relaciones, donde el grado de A es m+n y el grado de B es n, obteniendo como resultado un cociente de grado m. Los atributos m+i de A deben estar definidos sobre el mismo dominio que el atributo i-ésimo de B. En la relación final se eliminan las tuplas repetidas.

Al ser el AR un lenguaje de tipo procedimental, los pasos que forman la consulta describen un procedimiento. Las operaciones que se proponen en el AR se utilizan como una representación intermedia de una consulta a una base de datos, y debido a sus propiedades algebraicas sirven para obtener una versión optimizada y más eficiente de dicha consulta. Por esta última propiedad es importante que los estudiantes lo manejen.

El AR es un lenguaje abstracto, permite entender el MR de bases de datos desde la perspectiva matemática, donde una misma consulta se puede formular a partir de diferentes expresiones. Como ya se mencionó en la introducción, estas características traen consigo dificultades para su asimilación por parte de los estudiantes, sobre todo en la formulación y verificación de expresiones.

El alto nivel de abstracción del AR dificulta mucho su enseñanza y aprendizaje, pues prácticamente es un lenguaje de consulta teórico; mientras que otros lenguajes de consulta como el SQL, sí permiten realizar pruebas y obtener resultados de forma natural en las bases de datos y por lo tanto son menos abstractos.

Las posibilidades que ofrecen las TIC para el desarrollo de una enseñanza flexible son diversas, pero el desarrollo de sus aplicaciones en los distintos componentes de proceso de enseñanza no ha avanzado por igual. En este sentido, actualmente se puede hablar de una escasa tradición en el uso de las TIC para la evaluación de los procesos de enseñanza. (18) Las TIC ofrecen posibilidades para diseñar múltiples instrumentos para organizar la información

recogida en el proceso de evaluación, lo cual facilita el aprendizaje en los estudiantes.

La evaluación, en una concepción moderna de la enseñanza, ha de impregnar todo el PEA. Pero la evaluación puede contener muchos elementos, tiempos y métodos diferentes, y las TIC se están convirtiendo en un recurso útil en muchos de ellos. (19)

En la UCI se aprovechan diferentes recursos y medios para apoyar el proceso, como es el caso de la utilización del EVEA. Los profesores asignan las notas de los estudiantes siguiendo dos valoraciones: una basada en los trabajos presenciales y otra mediante los trabajos virtuales enviados por correo electrónico y los colocados en el EVEA. Sin embargo el EVEA solo brinda la evaluación automatizada mediante los cuestionarios de cada asignatura, brindando estrategias que serían muy difíciles realizar en papel. Pueden crearse cuestionarios con diferentes tipos de preguntas, generar cuestionarios aleatorios a partir de baterías de preguntas, permitir a los estudiantes tener múltiples intentos y muchas veces poder consultar estos resultados.

Estos cuestionarios se crean para las diferentes asignaturas, entre ellas Sistemas de Bases de Datos 1 y Sistemas de Bases de Datos 2, lo cual constituye una estrategia eficaz para mejorar el aprendizaje de los estudiantes y fomentando, al mismo tiempo, una formación de calidad. El uso de las TIC puede ser una herramienta útil en los aspectos del PEA que necesiten lograr un nivel de desarrollo que vaya a la identificación y a la reproducción, sin embargo cuando se requiere producir o aplicar conocimiento es muy difícil utilizar estas herramientas con un nivel de automatización que libere al profesor de las tareas repetitivas. Aunque se intuye una gran potencialidad en estos medios, siempre es necesario que existan criterios pedagógicos explícitos que guíen su aplicabilidad.

La evaluación automatizada en las asignaturas de bases de datos.

La utilización de las computadoras para ayudar en las tareas de evaluación ha sido un tema de investigación durante décadas, aunque básicamente ha consistido en trasladar los métodos tradicionales de evaluación a un entorno informático, proliferando la evaluación a través de ejercicios tipo test, elección múltiple, etc. (20) Estas formas favorecen una evaluación orientada a lo reproductivo y no a los elementos creativos, aunque es justo destacar que la existencia, por ejemplo de jueces en línea en los concursos de programación ha abierto otras posibilidades. Tal es el caso de los jueces en Línea de la Universidad de Valladolid (UVA) en España, Sphere (SPOJ) de la Universidad de Gdansk en Polonia y PKU (POJ) de la Universidad de Peking en China. Estos jueces en línea permiten evaluar automáticamente los intentos de solución de los programadores. (21)

Con el auge de las TIC, el sistema educativo en general y la evaluación como parte de él se han visto inmersos en un proceso de cambio, hablándose ahora de la evaluación automatizada como la que se realiza a través de un sistema informático. (20)

Soler coincide en que la evaluación automatizada es aquella que se realiza a través de un sistema informático. Con su desarrollo mismo se puede contribuir a mejorar el proceso de aprendizaje de los estudiantes. (20)

En la actualidad existen herramientas para la evaluación automatizada en Bases de Datos que permiten evaluar al estudiante en el laboratorio y no sobre consultas en papel. Con su aplicación se obtiene como resultado que los estudiantes cometan menos errores sintácticos. (22)

La evaluación de las tareas, fundamentalmente las ingenieriles que requieren una aplicación de los conocimientos, es una labor que consume mucho tiempo a los profesores. La evaluación automatizada desde este punto de vista puede aportar beneficios, pues el profesor no tendrá que revisar manualmente este tipo de tareas y asignar la nota sino puede utilizar su tiempo racionalmente. Hay que destacar también que el tiempo dejado de utilizar en la evaluación y calificación puede ser utilizado en el diseño de ejercicios y en conducir el PEA de sus estudiantes para lograr los objetivos propuestos. (23)

Existe otro grupo de aplicaciones informáticas enfocadas al diseño y compilación de los lenguajes de manipulación de datos como el CR y AR. Específicamente de AR las herramientas encontradas solo permiten realizar consultas sobre una BD, pero ninguna proporciona la evaluación de la consulta formulada.

1.2. Análisis de algunas herramientas para la evaluación automatizada de expresiones del Álgebra Relacional

El lenguaje del AR en la actualidad es utilizado en la mayor parte de los casos en entornos formativos. Es por ello que no existen muchas herramientas en el mercado que la utilicen desde otro punto de vista. Durante la revisión documental realizada se detectó que las herramientas que se pueden encontrar están vinculadas a proyectos educativos que han sido desarrollados centrándose solamente en las funcionalidades que se necesitan en dicho entorno. (24)

A continuación se realiza el análisis de algunas de ellas, utilizándose los siguientes indicadores:

I. Fin educativo (Sí/No)
II. Lenguajes de datos con los que trabaja (AR)

III. Utiliza notación matemática habitual (Sí/No)
IV. Realiza traducción de las expresiones al SQL (Sí/No)
V. Ejecuta las consultas (Sí/No)
VI. Permite la gestión de bases de datos (Sí/No)
VII. Incluye opciones de optimización de consultas (Sí/No)
VIII. Utiliza árboles o diagramas para la representación de las consultas (Sí/No)
IX. Permite la conexión a algún SGBD (Sí/No)
X. Detecta errores en el planteamiento de las consultas (Sí/No)
XI. Realiza corrección automatizada de errores en las consultas (Sí/No)
XII. Su distribución es gratuita (Sí/No)
XIII. Es multiplataforma (Sí/No)
XIV. Lenguajes en los que fue desarrollada
XV. Tipo de interfaz (Gráfica/Comandos)

Otros elementos que se tuvieron en cuenta son: lenguaje de programación en que se desarrolló, sistemas operativos para los que se encuentra disponible, tipo de licencia bajo la cual se desarrolló, lugar donde se desarrolló y proyecto al que está vinculada

Análisis individual de las herramientas encontradas:

1. Herramienta para el aprendizaje del AR:

La herramienta fue desarrollada con fines didácticos en la Universidad de Valladolid. Fue concebida para apoyar el PEA del AR en esa institución. Fue desarrollada sobre la plataforma MS Windows, utilizando C++ en su versión 6.0 y la biblioteca Microsoft Foundation Class. Como SGBD se utilizó Access, con su motor de BD Microsoft Jet en su versión 3.0, permitiendo la comunicación con el sistema mediante las funciones de las clases Data Access Objects.

El sistema presenta una interfaz gráfica. Tiene dentro de sus principios mantener al usuario constantemente informado acerca de los errores que comete, para que a partir de ellos pueda ir aprendiendo (25). Utiliza la notación matemática habitual (20). Dentro de sus desventajas está que no incluye opciones de optimización de consultas ni trabaja con diagramas o árboles para la representación de las consultas. Además no corrige de manera automatizada las expresiones ni es multiplataforma.

2. Virtual Tutor for Relational Algebra (Virtura):

Fue desarrollado con fines docentes en la University of Hertfordshire. Constituye un tutor virtual para el trabajo con el AR, y fue desarrollado en el lenguaje Java. (26). Tiene una interfaz visual que permite a los estudiantes desarrollar sintácticamente de manera correcta las expresiones

en AR. Estas expresiones se pueden construir de manera gradual, y pueden evaluarse en cada paso (26). Incluye un intérprete que traduce expresiones del AR al SQL, las cuales ejecuta para mostrar los resultados (20). No permite la utilización de la notación matemática habitual del AR. Su principal limitación es que fue diseñada para trabajar con una única BD previamente definida, desarrollada utilizando Microsoft Access (20).

3. Leap:

Proyecto de fin de carrera de la Universidad de Oxford, desarrollado en el lenguaje C. Posee una Licencia Pública General aunque esta solo permite su utilización en programas con la misma licencia. No cuenta con una interfaz gráfica lo que dificulta el entendimiento del proceso. Debe ser compilado en cualquier plataforma donde se quiera ejecutar aunque solo está disponible para Windows. La documentación adjunta es bastante completa lo cual es un punto muy favorable. Solo está disponible para el idioma Inglés, aunque su código es libre y puede realizarse cambios en el lenguaje. Está desarrollado para que se instale en un servidor y varias copias de la aplicación en diferentes ordenadores. La última versión de la aplicación data del 2005 por lo que el proyecto parece estar abandonado. Está concebida para los operadores tradicionales del álgebra y su distribución es gratuita. (27)

4. RelationalQuery:

Herramienta desarrollada en la Universidad de Sevilla. Posee una interfaz gráfica que permite almacenar/recuperar las consultas. Presenta facilidades en cuanto a su descarga, es multiplataforma ya que su desarrollo se basa en Java, su distribución es gratuita. Permite la traducción al lenguaje SQL. No incluye el operador división. En la actualidad el proyecto se encuentra abandonado. Está concebida para los operadores tradicionales del álgebra. (28)

5. AR y CR con un enfoque de programación:

Fue desarrolla en la Weber State University, con un fin docente. El desarrollo de las expresiones no se concibe en términos matemáticos ni con la notación matemática usual, sino con un enfoque a la programación. Se considera que de esta manera la representación es más familiar para los estudiantes que la representación matemática. Su funcionalidad es exclusivamente mostrar la solución de las consultas que se formulan (20). La manera en que son implementados los lenguajes impide que se pueda utilizar la notación matemática habitual. La concepción para la formulación de las expresiones implica que se pierda la esencia de estos lenguajes.

6. Relational:

Herramienta desarrollada en la Facultad de Ciencias Matemáticas, Físicas y Naturales de la

Universidad de Catania en Italia, dispone de una interfaz gráfica que permite cargar y ejecutar consultas sobre ella y mostrar los resultados. Su implementación se llevó a coba en el lenguaje de programación Phyton y posee Licencia Pública General. No permite la traducción entre Álgebra Relacional y SQL. Disponible para los Sistemas Operativos (SO) Windows, Debian, MacOS. No se publica una documentación detallada y la ayuda al usuario es pobre, lo que hace complicado su utilización. Está concebida para los operadores tradicionales del álgebra. Su distribución es gratuita. (29)

7. Rat:

Herramienta desarrollada en la Universidad Nacional de Costa Rica, posee una interfaz de usuario amigable y entendible. Está disponible es varios idiomas, posee la documentación suficiente para el entendimiento del mismo. Requiere poco recursos de *hardware* (256 Mb RAM). Es de fácil acceso y descarga completamente gratuita. Está concebida para los operadores tradicionales del álgebra. Entre sus funcionalidades se encuentra, en primer lugar, la validación léxica, sintáctica y semántica de las expresiones que se formulen, para garantizar la escritura correcta de las sentencias. Las expresiones son traducidas al SQL y la aplicación es capaz de conectarse a diferentes BD ya creadas en SGBD externos para probarlas. Esta conexión puede realizarse a cualquier SGBD relacional mediante el estándar de comunicación ODBC. (30)

Una de las desventajas que tiene el uso de esta herramienta, es que al ser una aplicación de escritorio, no puede integrarse a otras plataformas que sean utilizadas en diferentes centros educacionales. Esta herramienta tampoco realiza la corrección automatizada de errores.

8. Relational Algebra Interface (RAIN):

Es una herramienta con fines docentes de la University o Stirling. Tiene una interfaz visual atractiva que permite la creación y ejecución de consultas en Álgebra Relacional. Incluye la posibilidad de conectarse con SGBD externos y remotos y utiliza notación en lenguaje natural y notación algebraica. Permite la detección y retroalimentación sobre los errores introducidos en las consultas. También realiza la traducción al SQL de las consultas y tiene un tutorial online. (31) Esta herramienta no incluye opciones para la optimización de consultas ni el trabajo con árboles o diagramas para su visualización. Además no realiza corrección automatizada de errores.

9. Relational Algebra Learning Tool (RALT):

Esta herramienta es resultado del desarrollo de una tesis de grado del Imperial College de Londres. Incluye dentro de sus funcionalidades la creación de consultas en AR usando una

interfaz gráfica interactiva con un enfoque de flujo de datos, sin la necesidad de entrar manualmente las consultas al sistema. Esto evita que se introduzcan errores de sintaxis y que por tanto el trabajo esté enfocado a la comprensión del lenguaje. Las consultas pueden ser ejecutadas y visualizar los resultados paso a paso. Además puede conectarse a BD en SGBD externos e incluye un manual de usuario amplio que facilita su uso. (32)

10. Windows Relational Database Interpreter (WinRDBI):

Fue desarrollada por la Arizona State University con un fin educativo y su distribución es de forma gratuita. Su última versión está desarrollada en el lenguaje de programación Java y utiliza Amzi! Prolog para la interpretación de consultas (29). Es multiplataforma. Tiene una gran cantidad de recursos disponibles para aprender a trabajar con ella, su documentación es completa y bien redactada. (29) WinRDBI es un componente integral utilizado para lograr la comprensión de las capacidades que poseen los lenguajes de consulta para BD relacionales AR, Cálculo Relacional de Tuplas, Cálculo Relacional de Dominios, y el SQL en su versión SQL-92.

Tiene una interfaz de usuario amigable y completa, mediante la que se pueden crear y cargar bases de datos relacionales desde diferentes formatos de archivos, insertar información en ellas y formular y guardar consultas en los leguajes mencionados. La herramienta incluye una retroalimentación inmediata a los estudiantes pues visualiza las soluciones a las consultas propuestas (20).

Una de las desventajas que posee esta herramienta, es que no realiza correcciones automatizadas a las soluciones propuestas, por lo que la retroalimentación no es completa para los estudiantes. Además, no contempla la opción de optimización de consultas ni la manipulación de árboles de expresiones en AR. Adicionalmente, debido a que fue desarrollado sobre la base del establecimiento de tecnología de BD deductivas, que utilizan un lenguaje lógico para consultar las instancias de las BD almacenadas como hechos lógicos, los estudiantes tienen que aprender un nuevo lenguaje (22).

11. Ambiente para el estudio de los lenguajes de consulta relacionales (SQL-EDDI):

Este ambiente para el PEA de los lenguajes de consulta relacionales fue creado en la University of Warwick. Tiene un fin docente, encaminado a comprender la significación de la teoría relacional en el flujo lógico en SQL. Su estrategia está encaminada a combinar una introducción práctica al SQL con lecciones de AR, a partir de un ambiente en el cual los estudiantes pueden explorar la relación entre el uso práctico del estándar SQL y el AR. (20) Para su desarrollo se diseñó e implementó un lenguaje puro de AR nombrado EDDI, y se

desarrollaron traductores para dos variantes del SQL semánticamente consistentes con la teoría relacional: SQLZERO y un subconjunto del estándar SQL. En SQL-EDDI los estudiantes pueden estudiar la evaluación de expresiones algebraicas y relacionarlas con la traducción e interpretación de consultas en SQLZERO y SQL. Incluye representaciones visuales complementarias para que los estudiantes y tutores puedan evaluar de manera informal la calidad de las consultas. (33)

Tiene como desventaja que las expresiones se escriben en una notación diferente a la notación matemática habitual, por lo que se debe aprender un lenguaje adicional para utilizarla. (34) Sus proyecciones estaban encaminadas a incrementar el conjunto de características del SQL que tenían implementadas tales como las definiciones de datos, restricciones de integridad, utilización de valores nulos etc.; la implementación de otros lenguajes de consulta relacionales tales como QUEL; e interfaces para el estudio de la optimización de consultas. (33)

12. Interactive Data Flow Query Language (iDFQL):

Fue desarrollada con fines educativos en la Universidad de Sao Paulo. Constituye una herramienta interactiva que facilita el PEA del AR, mediante la utilización de elementos gráficos para representar las consultas (20). Los operadores del lenguaje se representan como procesos identificados por iconos, y la conexión entre estos iconos produce un diagrama de flujo que representa la consulta. (26)

Las operaciones se pueden ejecutar paso a paso. La representación final de la consulta es un esquema en forma de árbol, en donde las relaciones son los nodos terminales, las operaciones los nodos intermedios, y la raíz contiene la solución final. El estudiante visualiza el resultado final y la sentencia SQL equivalente a la expresión definida. (34)

La herramienta no es una aplicación web. Tiene dentro de sus insuficiencias que no corrige las soluciones incorrectas que se introducen, solamente visualiza el resultado de la ejecución parcial o total del diagrama. (34) No utiliza la notación matemática habitual y tampoco realiza la corrección automatizada de errores.

13. Plataforma para la Evaluación Continua y Mejora de la Enseñanza del AR (ACME-AR):

Se incluye dentro de la plataforma ACME, desarrollada con fines docentes en la Universidad de Girona. El objetivo fundamental de la plataforma está encaminado a la corrección automatizada y online de ejercicios relacionados con la enseñanza y aprendizaje de diferentes materias, incluidas las bases datos. (20)**,** (34)

ACME-AR es un módulo que permite la evaluación de consultas en AR. Además de las funcionalidades generales de la plataforma, este módulo presenta una interfaz gráfica para la entrada de expresiones, con todos los elementos que este lenguaje incluye. Implementa un corrector de expresiones que incluye un analizador sintáctico, un módulo que ejecuta las operaciones para obtener el conjunto resultado, y un módulo comparador para verificar si son correctas o no a partir de la comparación de la solución propuesta y la obtenida. (34)

Este sistema facilita la evaluación formativa pues proporciona una retroalimentación inmediata a los estudiantes. Utiliza la notación matemática usual para este lenguaje. (20)**,** (34) Dentro de sus desventajas se encuentra que no permite la traducción al lenguaje SQL de las expresiones, no incluye opciones para la optimización de consultas, no incluye la representación mediante árboles o diagramas y no permite la conexión con SGBD externos.

14. Herramienta para el aprendizaje del AR y optimización de consultas

Esta es una aplicación resultado de una tesis de grado (29) de la Universidad de Zaragoza. Su principal objetivo es el apoyo a la enseñanza aprendizaje del lenguaje AR. Fue desarrollada en el lenguaje de programación Java. Permite la introducción y manipulación de expresiones en AR y árboles de expresiones en este mismo lenguaje, las cuales se pueden ejecutar de forma automática y paso a paso, ya sea en la propia aplicación o en SGBD externos. La aplicación desarrollada brinda la posibilidad de introducir un conjunto de relaciones de ejemplo sobre las cuales se pueden ejecutar las expresiones introducidas. Estas relaciones pueden crearse desde la propia aplicación, desde BD externas, o a partir de ficheros externos. (29)

Otra de las posibilidades que brinda es la de realizar la optimización de consultas que son introducidas por el usuario. Esto se realiza de forma automática, o paso a paso, aplicando las reglas de transformación del AR así como las estadísticas de las relaciones. Al realizar cada uno de estos procesos se incluyen un conjunto de explicaciones, que permiten a los usuarios una comprensión adecuada. Otra de las funcionalidades que tiene esta aplicación es la traducción de expresiones del AR al lenguaje estándar SQL, lo que permite constatar la relación que existe entre estos dos lenguajes con un modo de ejecución paso a paso. Tiene una interfaz gráfica fácil e intuitiva. (29)

Esta herramienta tiene como desventaja que no permite la utilización de la notación matemática habitual del AR.

15. Herramienta para la creación y ejecución de consultas en Álgebra Relacional:

Herramienta creada con fines educativos en la Universidad de las Ciencias Informáticas para la creación y ejecución de consulta en el lenguaje Álgebra Relacional. La terminología adoptada

para representar los operadores es la usada por los estudiantes en el contexto de la asignatura de bases de datos en la universidad. La herramienta no permite la traducción de consultas al SQL aunque posibilita la ejecución de las consultas del lenguaje Álgebra Relacional.

La aplicación no soporta la gestión de base de datos ni la optimización de consultas. La misma no está apto para conectarse a ningún Sistema Gestor de Bases de Datos. No cuenta con detección de errores, planteamiento de las consultas ni la corrección automatizada de errores en la consulta. No es multiplataforma, ya que fue realizada con tecnología privativa, con el Entrono de Desarrollo Integrado Visual C# 2005 Express Edition y el Framework Microsoft.Net en su versión 2.0. Posee una interfaz gráfica para la interacción con los usuarios. (35)

Debido a que esta herramienta usa una interfaz gráfica interactiva para la definición de las consultas, no se utiliza la notación matemática habitual para el lenguaje. Además no trabaja con la optimización de consultas. Otra de sus desventajas es que no realiza la corrección automatizada de errores.

La Tabla 1.1 muestra el resumen de las herramientas analizadas. Realizando un análisis de la información contenida en ella se llegan a las siguientes conclusiones

- No existen herramientas que estén altamente difundidas y generalizadas, las existentes pertenecen a proyectos desarrollados localmente.
- Algunas aplicaciones de las analizadas anteriormente ejecutan las consultas introducidas por el usuario e incluyen opciones de optimización de consultas, mostrando mediante árboles o diagramas la representación de las consultas. Además realizan la corrección automatizada de errores a dichas consultas.
- La gran mayoría son sistemas multiplataforma, realizan traducción de las expresiones al SQL, permiten además la conexión con algún SGBD y detectan errores en el planteamiento de las consultas.
- Utilizan Java como lenguaje de programación y las gráficas como tipo de interfaz.
- Tienen como desventaja que ninguna incluye todas las funcionalidades y características analizadas.
- En algunos casos definen un nuevo lenguaje que el estudiante debe aprender, lo que puede dificultar el PEA del Álgebra Relacional.

No.	I	II	III	IV	V	VI	VII	VIII	IX	X	XI	XII	XIII	XIV	XV
1	Sí	AR	Sí	Sí	Sí	Sí	No	No	Sí	Sí	No	-	No	Visual C++	Gráfica
2	Sí	AR	No	Sí	Sí	No	No	No	No	Sí	No	-	Sí	Java	Gráfica
3	Sí	AR	No	No	Sí	No	No	Sí	Sí	Sí	No	Sí	Sí	C	Gráfica Comandos
4	Sí	AR CRT	No	Sí	*Sí*	No	No	No	Sí	Sí	No	Sí	Sí	Java	Gráfica
5	Sí	AR CRT	No	No	Sí	No	No	No	-	No	No	-	-	Visual FoxPro	-
6	Sí	AR	Sí	No	No	Sí	Sí	No	No	Sí	No	Sí	Sí	Phyton	Gráfica
7	Sí	AR	Sí	Sí	Sí	No	Sí	No	Sí	Sí	No		Sí	C++	Gráfica
8	Sí	AR	Sí	Sí	Sí	No	No	No	Sí	Sí	No		Sí	Java	Gráfica
9	Sí	AR	No	No	Sí	No	No	Sí	Sí	No	No	-	Sí	Java	Gráfica
10	Sí	AR CRT CRD	No	Sí	Sí	Sí	No	No	Sí	Sí	No	Sí	Sí	Java Amzi! Prolog	Gráfica
11	Sí	AR	No	Sí	Sí	Sí	No	No	No	No	No	-	Sí	-	Gráfica
12	Sí	AR	No	Sí	Sí	No	Sí	Sí	Sí	Sí	No	Sí	-	-	Gráfica
13	Sí	AR	Sí	No	Sí	No	No	No	No	Sí	Sí	-	-	-	Gráfica
14	Sí	AR	Sí	Sí	Sí	Sí	Sí	Sí	Sí	Sí	Sí	Sí	Sí	Java	Gráfica
15	Sí	AR	No	No	Sí	No	No	No	No	No	No	Sí	No	C#	Gráfica

Tabla 1.1 Resumen del análisis de las herramientas existentes

Requisitos fijados por el Departamento Metodológico Central de Ingeniería y Gestión de Software (DMC IGSW)

El proyecto al que está vinculada la presente investigación tiene como cliente al DMC IGSW de la UCI. Conjuntamente con esta aplicación se está desarrollando una similar, pero para el lenguaje Cálculo Relacional. Ambas aplicaciones se integran en una herramienta común, a petición del cliente, por lo que existen tareas que se desarrollan en conjunto con la participación de los autores de la otra aplicación. Los autores de la aplicación para el lenguaje CR son Damaysis Carmona Hernández y Alexei Pupo Ricardo.

El cliente además definió los siguientes requisitos para el sistema:

1. Debe utilizar la notación definida por el Departamento Metodológico Central de Ingeniería y Gestión de Software de la Universidad.
2. Debe permitir definir las características de los ejercicios a resolver por los estudiantes.
3. Debe ejecutarse en múltiples plataformas, considerando el software libre y de código abierto. La Universidad y el país hoy realiza un esfuerzo para introducir el Software

Libre, por lo que la herramienta no debe ser desarrollada utilizando lenguajes y herramientas que limiten su ejecución a plataformas propietarias.

4. Debe traducir del AR al SQL, lo cual se relaciona con que se llevan a cabo otros proyectos, entre ellos el mencionado para el lenguaje CE, y a futuro otro para la evaluación automatizada de expresiones del SQL.
5. Debe funcionar bajo una arquitectura cliente/servidor.
6. Debe contener un núcleo que pueda ser integrado al EVEA UCI.
7. Debe estar disponible 24x7x365 para que los estudiantes la puedan utilizar asincrónicamente.
8. Debe utilizar PostgreSQL como SGBD, al ser el SGBD definido para el proyecto por el departamento, y un SGBD de amplio uso en los proyectos que se desarrollan en la Universidad.
9. Debe contar con una interfaz gráfica que permita retroalimentar al estudiante de los errores cometidos.

Después de un proceso de búsqueda y análisis de la información disponible en la web se puede concluir que no se encontró ninguna herramienta que cumpla con los requisitos dados por el DMC IGSW. Otros elementos que apoyan esta conclusión son:

- Todas las propuestas estudiadas basan su sintaxis en la notación tradicional del AR u otra que no coincide con la que se imparte en la UCI.
- Cuatro (4) de las herramientas analizadas no permiten la traducción de las expresiones al SQL, requisito impuesto para que el motor de verificación sea uno solo.
- La herramienta Leap no cuenta con una interfaz gráfica, el Relational, el RALT, el Leap y el RAIN tienen muy poca documentación, o no tienen manual de usuario, lo cual en alguno de estos casos, dificulta su utilización.

1.3. Propuesta de solución

Luego de un detallado análisis de los problemas que presentan actualmente los estudiantes de segundo año en el lenguaje AR, se concluye la necesidad de realizar una herramienta para la evaluación automatizada de expresiones escritas utilizando el AR. La misma debe garantizar una adecuada retroalimentación,para que el estudiante pueda conocer los errores cometidos y corregirlos. Para dar cumplimiento a estos objetivos la aplicación contará con las siguientes funcionalidades generales:

- Conexión BD (IP servidor, usuario, contraseña, nombre de la BD).
- Gestión de los ejercicios (Insertar, Modificar y Eliminar ejercicios).

- Traducir la solución del lenguaje Álgebra Relacional a una instrucción en el lenguaje SQL.
- Comprobar y Retroalimentar la Solución del estudiante.
- Mostrar resultados de las consultas.

1.4. Herramientas y tecnologías para el desarrollo de la solución

Antes de definir las herramientas y tecnologías para el desarrollo de la solución es necesario profundizar en una de las funcionalidades de la propuesta de solución, en este caso la tercera, que plantea que el sistema debe ser capaz de traducir una expresión escrita utilizando el lenguaje del AR en una expresión equivalente escrita en el lenguaje SQL. Esto conlleva a que el sistema internamente debe tener un traductor capaz de realizar este proceso.

Un traductor es un programa que traduce o convierte un programa escrito en un lenguaje fuente en otro programa escrito en un lenguaje destino, detectando y tratando los posibles errores. (36)

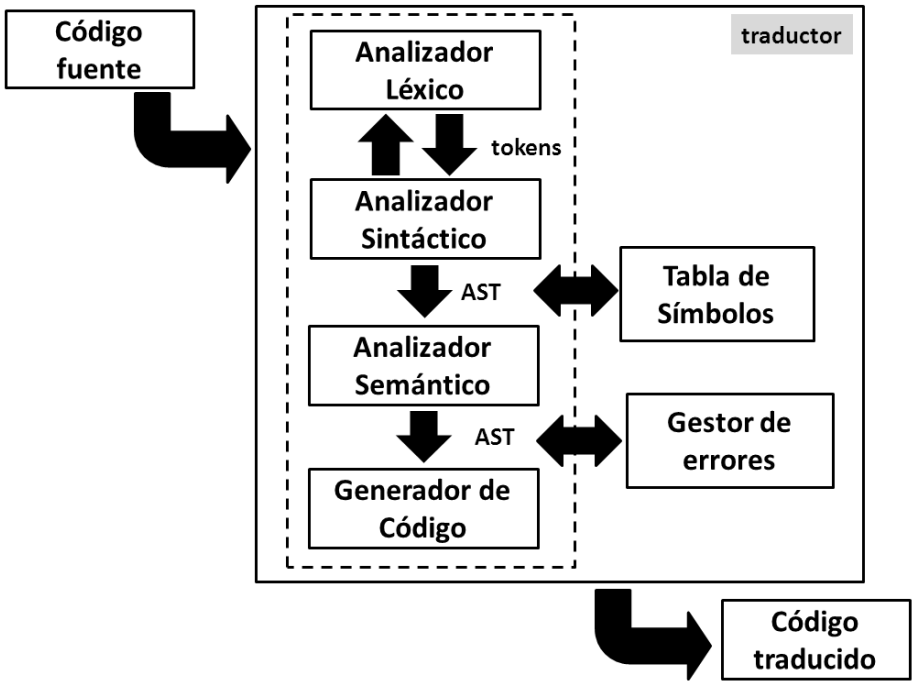


Figura 1.1 Traductor

Las fases de un traductor (36) son cuatro, el análisis léxico, el análisis sintáctico, el análisis semántico y la generación de código.

El **análisis léxico** es la primera fase del traductor. En él se recibe una cadena de caracteres

como entrada, y su principal función es agruparlos en entidades sintácticas o elementales denominados *tokens*[3]. A cada uno de estos *tokens* se le asigna una estructura de la siguiente manera: <tipo_tokens,información>. El primer parámetro determina la categoría (identificador, constante, etc.) y la segunda parte de la tupla proporciona información sobre el *token*, por ejemplo su entrada en la Tabla de Símbolos, el lexema asociado, etc. (36)

Por su parte la fase de **análisis sintáctico** examina una secuencia de *tokens* para determinar si el orden de esa secuencia es correcto de acuerdo a la definición sintáctica del lenguaje. La entrada del analizador sintáctico lo constituye la secuencia de *tokens* generada por el analizador léxico, y la salida es, por un lado, la indicación del cumplimiento de las reglas gramaticales que definen al lenguaje, y por el otro una representación del código que se está traduciendo. Para representar este código muchas veces se utiliza un Árbol de Sintaxis Abstracta (AST). (36)

El **Analizador semántico** se encarga de detectar errores relacionados con la validez del programa, que no pueden ser detectados en las fases anteriores. Verifica que las variables hayan sido declaradas previamente y concuerdan con sus respectivos tipos de datos. Comprueba el tipo de expresiones y modifica al AST incluyendo otra información necesaria para la generación de código. (36)

En la **Generación de código,** la salida del análisis semántico se emplea como entrada para la generación de código. La estructura de datos empleada para intercambiar información entre las dos fases mencionadas es un árbol de sintaxis abstracta decorado. (36)

Este traductor es un software, por lo que su proceso de desarrollo es similar al de otros sistemas de software más complejos. El proceso de desarrollo de un software es definido por Letelier (2003) como un proceso complejo donde: *"las personas desempeñan uno o más roles específicos, utilizan diferentes herramientas para producir artefactos y ejecutan diferentes actividades. El avance del proyecto en el tiempo es controlado mediante hitos que establecen un determinado estado para ciertos artefactos. Los artefactos son resultados obtenidos en cada una de las fases del proceso…"*, esta definición coincide con lo planteado por diferentes autores como Meyer (2006), Jacobson et al (2004) y Pressman (2006) (37).

Es por ello que cuando se decide la construcción de un software hay que seleccionar la metodología de desarrollo, el sistema de gestión de bases de datos, los lenguajes de programación y las herramientas que permitirán desarrollarlo.

[3] Conjunto de lexemas que puede ser tratado como una unidad sintáctica.

Las ***metodologías de desarrollo*** son una guía para los desarrolladores a la hora de la construcción de un software. Estas se agrupan en dos grupos, el primer grupo se orienta al control de los procesos estableciendo rigurosamente las actividades a desarrollar, herramientas a utilizar y notaciones que se usarán. Estas metodologías son llamadas Metodologías Pesadas, Tradicionales o Robustas. El otro grupo se enfoca en la interacción con el cliente y el desarrollo incremental del software, mostrando versiones parcialmente funcionales del software al cliente en intervalos cortos de tiempo, para que pueda evaluar y sugerir cambios en el producto según se va desarrollando (Metodologías Ligeras/Ágiles). (38)

Entre las Metodologías Tradicionales se puede citar algunas: Rational Unified Procces (RUP), Microsoft Solution Framework (MSF) y Win-Win Spiral Model. Entre las Metodologías Ágiles de desarrollo más destacadas se pueden nombrar: Extreme Programming (XP), Scrum, Crystal Clear y Extreme Modeling. (39) (40)

Comparación entre las Metodologías Ágiles y Robustas: (40)

Metodologías Ágiles	Metodologías Pesadas
Basadas en heurísticas provenientes de prácticas de producción de código.	Basadas en normas provenientes de estándares seguidos por el entorno de desarrollo.
Especialmente preparadas para cambios durante el proyecto.	Cierta resistencia a los cambios
Proceso menos controlado, con pocos principios.	Proceso mucho más controlado, con numerosas políticas/normas.
No existe contrato tradicional o al menos es bastante flexible.	Existe un contrato prefijado.
El cliente es parte del equipo de desarrollo.	El cliente interactúa con el equipo de desarrollo mediante reuniones.
Grupos pequeños (menor que 10 integrantes) que trabajan en el mismo sitio.	Grupos grandes y posiblemente distribuidos.
Pocos artefactos.	Más artefactos.
Pocos roles.	Más roles.
Menos énfasis en la arquitectura del software.	La arquitectura del software es esencial y se expresa mediante modelos.

Tabla 1.2 Diferencia entre Metodologías Ágiles y pesadas.

En la selección de una determina metodología de desarrollo intervienen diversos factores, como la complejidad del software y el tamaño del equipo. Luego de un análisis de los requerimientos del proyecto y de las diferentes metodologías, se selecciona a XP. La herramienta que se va a implementar cuenta con un equipo de desarrollo pequeño, de dos integrantes, es de pequeña envergadura y debe ser realizado en un corto tiempo de duración.

A continuación se realiza una caracterización de la misma que permite dar otros elementos sobre su selección.

XP nace como una nueva forma de enfrentar proyectos de software, proponiendo una metodología basada esencialmente en la simplicidad y agilidad. Esta metodología surge para hacer frente a los grandes procesos planteados por las metodologías tradicionales tildadas en muchos casos de burocráticas. (41)

XP es utilizada generalmente en proyectos de menor envergadura con equipos de desarrollo pequeños y donde se dispone de un corto tiempo de desarrollo (41). Propone la programación en dúo con ambos trabajando juntos en una misma computadora. El producto obtenido es por lo general de mejor calidad que cuando el desarrollo se realiza por programadores individuales.

Entre los principales objetivos trazados por esta metodología se pueden mencionar los siguientes:

- Desarrolladores, gerentes y clientes deben trabajar juntos diariamente, a lo largo del proyecto.
- Construir proyectos alrededor de personas motivadas, dándoles el entorno y soporte que necesitan, y confiando en que realizarán el trabajo.
- Satisfacer al cliente a través de entregas continuas y tempranas es la mayor prioridad.
- Los cambios a los requerimientos son bienvenidos.
- Entregar frecuentemente software que funciona, desde un par de semanas a un par de meses, prefiriendo los periodos más cortos.
- El método más eficiente y efectivo de transmitir información entre un equipo de desarrolladores es la conversación frontal (cara a cara).

Uno de los elementos más importantes a elegir a la hora de planificar el desarrollo de un software, es el lenguaje en que se comunicarán los diferentes miembros del equipo de desarrollo, pues este se convierte en el vehículo que transporta las ideas de una fase a otra y entre los miembros.

El lenguaje unificado de modelado (UML), es un lenguaje para la especificación, visualización, construcción y documentación de los artefactos de un proceso de sistema intensivo. (42) UML se ha convertido en uno de los estándares más utilizados para representar y modelar la información con la que se trabaja en las fases de análisis y especialmente de diseño. Permite representar en mayor o menor medida todas las fases de un proyecto informático desde el análisis con los casos de uso, el diseño con los diagramas de clases, objetos, etc., hasta la implementación.

Habiendo definido el UML como lenguaje para el modelado es necesario tener en cuenta que existen herramientas CASE (Ingeniería de Software Asistida por Computadoras) que permiten la informatización de las fases de análisis y diseño, permitiendo la obtención de diferentes artefactos del proyecto. Dentro de estas herramientas destaca *Visual Paradigm for UML Suite 5.0*. La misma propicia un conjunto de ayudas para el desarrollo de programas informáticos, desde la planificación, pasando por el análisis y el diseño, hasta la generación del código fuente de los programas y la documentación. Es una herramienta para el desarrollo de aplicaciones utilizando UML como lenguaje de modelado. Además proporciona el modelado de procesos de negocio, un generador de mapeo objeto-relacional para Java, .NET y PHP. Visual Paradigm for UML está disponible para Linux y Windows. (43)

Visual Paradigm for UML puede generar código a partir de un diagrama de clases, así como la estructura de una base de datos relacional adecuada para mantener la información contenida en las clases entidad. (43)

Relacionado con el mantenimiento o persistencia de la información, en el presente proyecto el DMC IGSW impuso para el desarrollo de la herramienta como SGBD a PostgreSQL, por lo que se realizó el análisis de las características fundamentales que permitieran aportar elementos para apoyar esta decisión.

Un SGBD es una colección de programas cuyo objetivo es servir de interfaz entre la base de datos, el usuario y las aplicaciones. Se compone por un lenguaje de definición de datos, un lenguaje de manipulación de datos y un lenguaje de consulta. Un SGBD permite definir los datos a distintos niveles de abstracción y manipular dichos datos, garantizando la seguridad e integridad de los mismos. (44)

Un SGBD debe permitir definir una base de datos: especificar tipos, estructuras y restricciones de datos; construir la base de datos: guardar los datos en algún medio controlado por el mismo SGBD; y manipular la base de datos: realizar consultas, actualizarla, generar informes. (44)

PostgreSQL es un SGBD objeto-relacional y con su código fuente disponible libremente. Es uno de los SGBD de código abierto más potentes del mercado. Su desarrollo comenzó hace más de 16 años, funciona muy bien con grandes volúmenes de información y una alta concurrencia de usuarios accediendo simultáneamente al sistema. Este SGBD destaca por su robustez, escalabilidad y cumplimiento de los estándares SQL. Cuenta con versiones para una amplia gama de sistemas operativos. Incluye la mayor parte de los tipos de datos especificados en los estándares SQL92 y SQL99, como: entero, numérico, booleano, char, varchar, fecha, interval o timestamp. Algunos límites generales de PostgreSQL, están incluidos en la siguiente

tabla. (45)

Límite	Valor
Tamaño máximo de la base de datos	Ilimitado (Depende de las capacidades del sistema de almacenamiento)
Tamaño máximo de la tabla	32 TB
Tamaño máximo de una fila	1.6 TB
Tamaño máximo de un campo	1 GB
Número máximo de filas por tabla	Ilimitado
Número máximo de columnas por tabla	250 - 1600 (dependiendo del tipo)
Número máximo de índices por tabla	Ilimitado

Tabla 1.3 Límites generales de PostgreSQL

Algunas de las ventajas de PostgreSQL son:

- Mantiene la misma velocidad tanto en bases de datos pequeñas como grandes.
- Instalación ilimitada: No posee un límite de copias a instalar en los ordenadores.
- Extensible y adaptable: El código fuente está disponible de forma gratuita, para que quien necesite extender o personalizar el programa pueda hacerlo según sus necesidades.
- Multiplataforma: Puede operar sobre distintas plataformas, incluyendo *Linux, Windows, Unix, Solaris y MacOS X,* además de una versión nativa de Windows en estado de prueba.
- Alta escalabilidad ya que es capaz de ajustarse al número de CPUs y a la cantidad de memoria disponible, soportando una mayor cantidad de peticiones simultáneas a la base de datos de forma correcta.

Para el trabajo con el SGBD y con las bases de datos, es bueno contar con herramientas menos complejas o más potentes que las que se incluyen generalmente en su distribución, es por ello que en ocasiones se seleccionan otras con mejores características. Tal es el caso de DBDesigner Fork, un sistema totalmente visual de diseño de bases de datos, que combina características y funciones profesionales con un diseño simple, muy claro y fácil de usar, a fin de ofrecer un método efectivo para gestionar las bases de datos. Permite administrar la base de datos, diseñar tablas, hacer peticiones SQL manuales y mucho más, como ingeniería inversa en MySQL, Oracle, MSSQL y otras bases de datos ODBC, modelos XML y soporte para la función *drag-and-drop*. (46)

DBDesigner Fork se encuentra en capacidad de trabajar en diferentes plataformas informáticas, como es el caso de Windows o Linux, lo que aumenta su versatilidad y permite que el diseñador y administrador de las bases de datos pueda elegir el sistema operativo que desee o que mejor se adapte a las necesidades del proyecto. (47)

Una vez seleccionadas la metodología, el SGBD y las herramientas que permitirán la administración de la base de datos es necesario seleccionar el lenguaje de programación, el entorno de desarrollo, así como el resto de las herramientas que permitirá llevar a cabo el trabajo.

Como se mencionó al inicio del epígrafe dos, uno de los requisitos impuestos al proyecto es que la aplicación sea multiplataforma y que se utilicen herramientas de software libre y de código abierto. Por lo anterior se seleccionó el Lenguaje de programación Java y el entorno integrado de desarrollo NetBeans.

Java es un lenguaje multiplataforma, lo que permite la ejecución de los programas escritos en el en diferentes sistemas operativos. Algunas de sus características más importantes que posee son: (48)

- **Orientado a objetos**: da buen soporte a las técnicas de desarrollo de la Programación Orientada a Objeto y en resumen a la reutilización de componentes de software.
- **Robusto:** fue diseñado para crear software altamente fiable. Para ello proporciona numerosas comprobaciones en compilación y en tiempo de ejecución.
- **Arquitectura neutral:** está diseñado para soportar aplicaciones que serán ejecutadas en los más variados entornos de red y estaciones de trabajo, sobre arquitecturas distintas y con sistemas operativos diversos.
- **Manejo automático de la memoria**: no hay que preocuparse de la recuperación de la memoria, el recolector de basura hace todo ese trabajo.

Para realizar toda la conexión al SGBD se seleccionó Java Database Connectivity (JDBC), un conjunto interfaces de programación que permite el acceso externo a comandos de manipulación y actualización SQL de la base de datos. Ellas permiten la integración de llamadas del SQL en un entorno de programación general proporcionando una biblioteca de rutinas las cuales interactúan con la base de datos. JDBC tiene una rica colección de rutinas que hacen tal interfaz sumamente simple e intuitiva. (49) En general, JDBC permite la ejecución de operaciones sobre bases de datos desde el lenguaje de programación Java. El *driver* JDBC será utilizado para realizar la conexión de la aplicación al SGBD PostgreSQL.

Para desarrollar utilizando a Java y a la biblioteca JDBC, una de las opciones es *NetBeans*,

proyecto exitoso de código abierto con una gran base de usuarios. *NetBeans* es una herramienta de programación que ha sido empaquetada como un programa de aplicación, que consiste en un editor de código, un compilador, un depurador y un diseñador interactivo de interfaz gráfica. Está enfocado al lenguaje de Programación Java, pero soporta además los lenguajes PHP, C/C++, JavaScript, HTML entre otros. (50)

NetBeans viene integrado con servidores de aplicaciones *GlassFish v3, Apache Tomcat* y maneja bases de datos *MySQL y PostgreSQL, entre otros*. (50) Otro aspecto a destacar es que el *Netbeans* es multiplataforma, lo cual permite que funcione en diversos sistemas operativos como *Windows, Mac, Linux o Solaris*. Lo anterior constituye una de las principales razones para optar por este IDE para el desarrollo del sistema.

Otro de los requisitos impuestos por el cliente es que la aplicación cuente con su ayuda. Para su desarrollo se seleccionó la librería *JavaHelp* 2.0.5.

JavaHelp es una librería opcional de *Java* que permite poner ventanas de ayuda a las aplicaciones de forma sencilla. (51) Esta herramienta se utiliza para crear la ayuda del sistema, permitiéndole al usuario esclarecer todas las dudas respecto a la aplicación.

Una vez concluida la aplicación se desarrollarán las pruebas pertinentes para la validación de su funcionamiento. Para las pruebas unitarias, que se realizan sobre el código Java, se utilizará la librería JUnit 4.0. Este tipo de herramienta hace que el esfuerzo y el trabajo en la fase de pruebas se reduzcan, permitiendo que el desarrollador se centre en la verificación de resultados correctos y no escribiendo código extenso para realizar sus pruebas. Además proporciona clases de las cuales se pueden heredar para formar las nuevas clases que serán las que realicen las pruebas unitarias a cada una de las clases que conforman la aplicación. Una prueba puede estar conformada por una serie de datos, utilización y resultados, este último se compara con los datos que en realidad debería mostrar el software, para tener conocimiento de si la aplicación está cumpliendo con la realización de lo solicitado. (52)

Conclusiones Parciales

En este capítulo se abordaron los problemas que motivaron al desarrollo del sistema y se definieron elementos de importancia para la elaboración del mismo. Se analizó el estado del arte de herramientas de este tipo en el contexto nacional e internacional, y de las herramientas y/o tecnologías a utilizar teniendo en cuenta las características definidas para el sistema a desarrollar. Teniendo en cuenta estos aspectos se concluye:

- El PEA en las asignaturas de Bases de Datos se ha ido enriqueciendo con la utilización de las TIC, convirtiéndose estas en herramientas importantes para el aprendizaje de

estos contenidos.

- El EVEA no incluye actividades que permitan evaluar automatizadamente la formulación de expresiones en AR y ofrecer una retroalimentación adecuada.
- La evaluación automatizada de expresiones del AR aporta beneficios en la visualización de las respuestas correctas, garantizando una retroalimentación a estudiantes y profesores que pueden tomar acciones para perfeccionar el PEA en función de mejorar los resultados.
- Las herramientas existentes identificadas y analizadas para la evaluación automatizada de expresiones del AR a nivel internacional y nacional, no cumplen con los requisitos necesarios para su utilización en la UCI.

A partir de las características fijadas por el cliente de la aplicación y con el objetivo de desarrollar un proceso de desarrollo de software exitoso, se seleccionan:

- XP como la metodología de desarrollo pues el equipo de desarrollo es pequeño, es necesario estar en contacto constante con el cliente y el proyecto no es de gran envergadura.
- Visual Paradigm for UML Suite 5.0 como herramienta para el modelado del Árbol de Sintaxis Abstracta.
- PostgreSQL 9.1.2 como SGBD impuesto por el cliente, a partir de los requisitos de otros sistemas que se desarrollan en paralelo. Java en su versión 7 como lenguaje de programación y NetBeans 7.0 como IDE de desarrollo. Las características de Java hacen que se garantice que el sistema pueda ser ejecutado en diferentes plataformas.
- JDBC en su versión 4.0 para realizar la conexión con las bases de datos.
- DBDesigner Fork como aplicación para el diseño de la base de datos y para obtener los scripts de la base de datos modelada.
- JavaHelp 2.0.5 como herramienta para la creación de la ayuda del sistema.
- La librería JUnit para el desarrollo de las pruebas unitarias en la aplicación.

CAPÍTULO 2: Análisis y diseño de la herramienta para la evaluación automatizada de expresiones del Álgebra Relacional

Este capítulo tiene como objetivo presentar los resultados del análisis y diseño de la propuesta del sistema. En el análisis se especifican los requisitos funcionales y no funcionales que el sistema debe cumplir para satisfacer las necesidades del cliente; se identifican las personas relacionadas con el sistema y se describen las Historias de Usuario (HU). En el diseño se describe la arquitectura del sistema, los patrones a utilizar, se realiza el diseño de la base datos, así como las tarjetas CRC.

2.1. Descripción del Sistema

La aplicación HEA-AR se desarrollará como una aplicación de escritorio y se publicará en el EVEA donde estará disponible para su descarga.

El sistema estará compuesto por dos aplicaciones clientes, una que será utilizada por el profesor para la gestión de los ejercicios, y otra para la solución de los ejercicios por parte del estudiante. El profesor será el responsable de la gestión de los ejercicios (adicionar, modificar, eliminar y listar los mismos). Como datos del ejercicio se incluye el nombre, el enunciado, el script de la BD, las relaciones de las tablas con sus atributos, los incisos con sus descripciones, posibles soluciones en AR y SQL, conjuntamente con la complejidad de cada ejercicio. Cada nuevo ejercicio constará con un esquema[4] dentro de la BD, con la información necesaria para su solución.

El estudiante deberá descargar la aplicación del EVEA, ejecutarla y realizar la conexión a la BD mediante una interfaz gráfica. Una vez realizados estos pasos, puede acceder a la interfaz “Responder ejercicio”, seleccionar el ejercicio y el inciso e introducir una posible solución en AR. El estudiante oprimirá la opción “Ejecutar” y el sistema será el encargado de realizar validaciones léxicas, sintácticas y semánticas. Durante este proceso se ofrece una retroalimentación al usuario en caso de existir algún error, indicándosele la naturaleza del mismo.

Para la verificación de la solución la herramienta traduce la respuesta en AR a SQL siempre que no tenga errores léxicos, sintácticos y/o semánticos. Esta traducción se ejecuta en el

[4] Contiene tablas, vistas, procedimientos, etc. Se encuentra dentro de una base de datos, que a su vez está dentro de un servidor.

SGBD conjuntamente con la solución en SQL propuesta por el profesor. Luego se verifica si coinciden los resultados de ambas consultas mediante la implementación de una funcionalidad que compare estos resultados, retroalimentando al usuario en cualquier caso (respuesta correcta o incorrecta) y se visualizan los datos obtenidos en cada consulta en SQL. La comparación de los resultados tiene en cuenta que las filas y las columnas de las tuplas resultantes de la ejecución de ambas expresiones en SQL, pueden tener un orden diferente, y sin embargo se consideran resultados iguales.

A continuación se muestra el flujo de información de la herramienta:

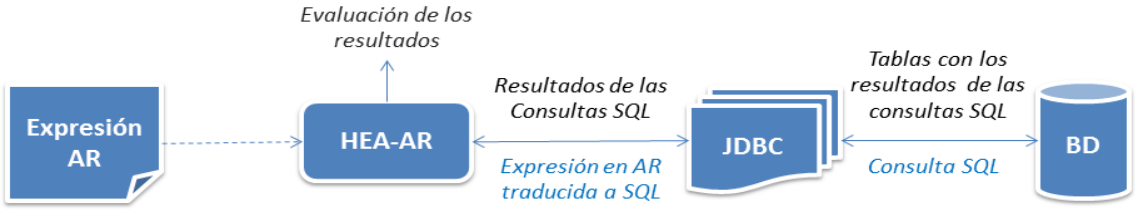


Figura 2.1 Figura Flujo de información de la herramienta

2.2. Requerimientos del Sistema

Los requerimientos de un software son las propiedades o restricciones, determinadas con precisión, que un producto software debe poseer. Los mismos se clasifican en funcionales y no funcionales. Los requisitos funcionales son las condiciones que debe cumplir el sistema, y los requisitos no funcionales son propiedades o cualidades que el producto debe tener.

Requisitos Funcionales

RF1: Conectar BD (IP servidor, usuario, contraseña, nombre de la BD).

RF2: Gestionar ejercicio.

2.1. Insertar ejercicio (nombre, enunciado, relaciones, incisos).

2.1.1. Cargar y ejecutar script.

2.2. Modificar ejercicio (identificador del ejercicio, enunciado, script).

2.2.1 Modificar inciso (descripción, solución_sql, solución_ar)

2.3 Eliminar ejercicio (identificador del ejercicio).

2.3.1 Eliminar inciso (identificador del inciso).

2.3.2 Eliminar relación (identificador de la relación).

2.4 Listar ejercicios.

RF3: Responder ejercicio.

3.1. Seleccionar ejercicio.

3.2. Introducir consulta en Álgebra Relacional.

RF4: Traducir solución del lenguaje Álgebra Relacional a SQL.

4.1. Validar la solución léxicamente.

4.2. Validar la solución sintácticamente.

4.3. Validar la solución semánticamente.

4.4. Traducir consulta.

RF5: Comprobar solución

5.1. Ejecutar consulta en PostgreSQL.

5.2. Comparar resultados de las consultas del profesor y del estudiante.

5.3. Retroalimentar la solución brindada por el estudiante.

5.4. Mostrar resultados de las consultas del profesor y del estudiante.

Requisitos no Funcionales

Apariencia o interfaz externa:

RnF1. El sistema debe contar con una interfaz amigable que permita tanto al profesor como al estudiante interactuar de forma cómoda y que facilite el trabajo con la herramienta.

Legales:

RnF2: Se usarán herramientas de software libre y código abierto, o que funcionen bajo las licencias GNU/GPL, por lo que el sistema será desarrollado también en los términos de la licencia GNU/GPL.

Disponibilidad:

RnF3: El sistema debe estar inicialmente disponible para su descarga y utilización en el EVEA de la universidad, dentro del curso virtual de SBD1.

RnF4: El Servidor de BD estará disponible 24x7x365 para que los estudiantes puedan utilizar la aplicación asincrónicamente

Confiabilidad:

RnF5. Se garantiza un tratamiento adecuado de las excepciones y la validación de las entradas del usuario.

Soporte:

RnF6. Debe poseer un manual de usuario.

Seguridad:

RnF7. Integridad: Se garantiza la integridad de la información que se maneja en el sistema ya que solo podrá ser modificada por las personas autorizadas.

Requisitos de hardware:

RnF8: En el cliente: PC Pentium 3 o superior, CPU 133 MHZ o superior, 256MB RAM mínimo, 512MB RAM recomendada o superior e interfaz de red para la conexión con la BD.

RnF9: En el servidor: Los servidores deberán contar CPU Dual Core 2.0 GHZ o superior, memoria RAM de 2 GB, un disco duro con capacidad disponible de almacenamiento de 10GB o mayor y una interfaz de red para poder brindar los servicios requeridos.

Requisitos de software:

RnF10: En el cliente: Para hacer uso de la aplicación se debe tener instalado previamente la Máquina Virtual de Java en la versión 7 o superior (JRE 7.0).

RnF11: En el servidor: Deberá contar con la instalación del Gestor de Bases de Datos PostgreSQL 9.2.

Personas y aplicaciones relacionadas con el sistema

Personas y aplicaciones	Descripción
Profesor	Será el encargado de publicar los ejercicios con los incisos que el estudiante va a realizar y las BD correspondientes, así como las consultas en AR y SQL que dan respuesta a cada inciso.
Estudiante	Podrá listar los ejercicios publicados y entrar las respuestas correspondientes en el editor para el ejercicio e inciso que seleccione.
Herramienta AR	Será la responsable de realizar la validación y traducción del lenguaje AR al SQL.

Tabla 2.1 Descripción de las personas y aplicaciones relacionadas con el sistema

2.3. Historias de Usuario

Las Historias de Usuario (HU) son formatos en los cuales el cliente describe brevemente las características que el sistema debe poseer, sean requisitos funcionales o no funcionales.

El tratamiento de las HU es muy dinámico y flexible. Cada una es lo suficientemente comprensible y delimitada para que los programadores puedan implementarla en unas semanas. (53)

Las HU tienen la misma finalidad que la técnica de los casos de uso que propone RUP, pero con algunas diferencias: constan de 3 o 4 líneas escritas por el cliente en un lenguaje no técnico sin hacer mucho hincapié en los detalles. Son usadas para estimar tiempos de desarrollo de la parte de la aplicación que describen. (54)

Para describir el sistema se elaboraron cinco (5) historias de usuarios, relacionadas con la

conexión a la base de datos, la gestión de los ejercicios por parte del profesor, la solución a los ejercicios por parte del estudiante, la tradición entre el AR y el SQL y la comprobación de la solución. Estas historias de usuario responden a los requisitos impuestos por el cliente para la herramienta. A continuación se muestra la HU gestionar ejercicio y la HU Comprobar solución.

Número: 2	**Nombre: Gestionar ejercicio**
Modificación de HU	
Usuario: Profesor	**Iteración Asignada**:1
Prioridad en negocio: Alta	
Riesgo en Desarrollo: Bajo	
Descripción: Se realiza la acción de gestionar ejercicio, dentro de la cual se encuentran las funciones Insertar, Modificar, Eliminar y Listar los ejercicio. Para insertar ejercicios se requieren los parámetros: nombre, enunciado, relaciones e incisos, además de cargar y ejecutar script. Para modificar los ejercicios se requiere el identificador del ejercicio, el enunciado y el script. Dentro de Modificar ejercicio se puede realizar la acción Modificar inciso, el cual incluye la descripción, la solución SQL y la solución en AR. La acción Eliminar ejercicio requiere el identificador del ejercicio. Dentro de Eliminar Ejercicios se pueden Eliminar inciso, que igualmente requiere el identificador del inciso. En la funcionalidad Eliminar relación se requiere el identificador de la relación para realizar dicha acción. La funcionalidad Listar ejercicios muestra información referente a los ejercicios: nombre, enunciado y cantidad de incisos.	

Tabla 2.2 Descripción HU Gestionar ejercicio

Historia de Usuario	
Número: 5	**Nombre:** Comprobar solución
Modificación de HU	
Usuario: Herramienta AR	**Iteración Asignada**: 4
Prioridad en negocio: Alta	
Riesgo en Desarrollo: Bajo	
Descripción: Se ejecutan las consultas del profesor y el estudiante en el SGBD PostgreSQL. Luego de haberse obtenido ambos resultados se comparan, y se realiza la retroalimentación, especificándole al estudiante a través de un mensaje si su resultado coincide con el del profesor o no. En ambos casos se muestran los resultados de ambas consultas.	

Tabla 2.3 Descripción HU Comprobar solución

2.4. Planificación

Durante la planificación, el cliente y el grupo de desarrolladores acuerdan el orden en que deberán implementarse las HU, según las necesidades más inmediatas; y se realiza una estimación del esfuerzo que se necesita para cada una de ellas.

Las **estimaciones de esfuerzo** asociado a la implementación de las HU se establecen utilizando como medida el punto. Un punto, equivale a una semana ideal de programación. Las historias generalmente valen de 1 a 3 puntos. Por otra parte, se mantiene un registro de la "velocidad" de desarrollo, establecida en puntos por iteración, basándose principalmente en la suma de puntos correspondientes a las historias de usuario que fueron terminadas en la última iteración. (55)

A continuación se muestra la estimación de esfuerzo por cada HU:

HU	Historia de Usuario	Puntos de estimación
1	Conectar BD	1
2	Gestionar ejercicio	2
3	Responder ejercicio	2
4	Traducir solución del lenguaje Álgebra Relacional a SQL	4
5	Comprobar solución	2

Tabla 2.4 Estimación de esfuerzo por Historias de Usuario

Como parte del ciclo de vida del software se continúa el proceso con la **planificación de duración de las iteraciones,** la cual tiene como objetivo mostrar la duración de cada iteración y orden en que serán implementadas las HU en cada una de ellas. El desarrollo del sistema se dividió en cuatro (4) iteraciones, estas se definen de la siguiente forma:

- **Iteración 1:** En esta iteración se implementaron las HU #1 y HU #2, ambas tienen una prioridad alta, obteniéndose así la versión 0.1 del producto dándole la posibilidad al usuario de probar dichas funcionalidades.
- **Iteración 2:** El objetivo de esta iteración es la implementación de la HU #3, que se encarga de visualizar los ejercicios a resolver para su posterior solución. Una vez concluida la iteración el estudiante tiene la posibilidad de introducir las respuestas a los ejercicios que la aplicación ofrece. Se obtiene la versión 0.2 del sistema.
- **Iteración 3:** Durante el transcurso de esta iteración se implementó la HU #4, en la cual se desarrolló el analizador léxico, sintáctico y semántico, brindando la información pertinente en caso de error y la traducción del lenguaje AR al SQL. Se obtiene la versión 0.3 del sistema.

- **Iteración 4:** Se realizó la HU #5 donde se comparan los resultados de las consultas del profesor y del estudiante, y se brinda la retroalimentación, mostrando los resultados de la comparación al estudiante. Se obtiene la versión 1.0 del sistema.

A continuación se muestra el orden de desarrollo de las HU en cada iteración en dependencia de la prioridad que tengan y el tiempo de duración de las mismas:

Iteración	Orden de implementación de las HU	Duración de la Iteración
Iteración 1	1. Conectar BD 2. Gestionar ejercicio	3 Semanas
Iteración 2	1. Responder ejercicio	2 Semanas
Iteración 3	1. Traducir solución del lenguaje AR a SQL	4 Semanas
Iteración 4	1. Comprobar solución	2 Semanas

Tabla 2.5 Planificación de duración de iteraciones

La **planificación de entregas de las iteraciones** establece qué HU serán agrupadas para conformar una entrega y el orden de las mismas; y se realiza en base a las estimaciones de tiempos de desarrollo realizadas por los desarrolladores. (41) A continuación se presenta la planificación de entregas para la fase de implementación, donde se realizaron versiones entregables del sistema al finalizar cada una de las iteraciones.

Sistema	Fin iteración 1	Fin iteración 2	Fin iteración 3	Fin iteración 4
Herramienta para la evaluación automatizada de expresiones del lenguaje Álgebra Relacional".	25/3/2013 Versión 0.1	7/4/2013 Versión 0.2	6/5/2013 Versión 0.3	22/5/2013 Versión 1.0

Tabla 2.6 Planificación de entregas de las iteraciones.

2.5. Diseño

En el ciclo de vida de un software, el diseño del proyecto es el proceso de elaboración de la propuesta de trabajo de acuerdo a pautas y procedimientos sistemáticos. (56)

XP no define una técnica específica de modelado, pueden utilizarse sencillos esquemas, tarjetas CRC (Clase, Responsabilidad, Colaboración) o diagramas de clase utilizando UML, siempre que sean útiles y no requieran mucho tiempo en su creación. (41)

Arquitectura Cliente/Servidor

Es una arquitectura distribuida que permite a los usuarios finales obtener acceso a la información en forma transparente aún en entornos multiplataforma. En el modelo cliente

servidor, el cliente envía un mensaje solicitando un determinado servicio a un servidor (hace una petición), y este envía uno o varios mensajes con la respuesta (provee el servicio). En un sistema distribuido cada máquina puede cumplir el rol de servidor para algunas tareas y el rol de cliente para otras. (57) Esta arquitectura se utiliza para realizar la conexión con la base de datos.

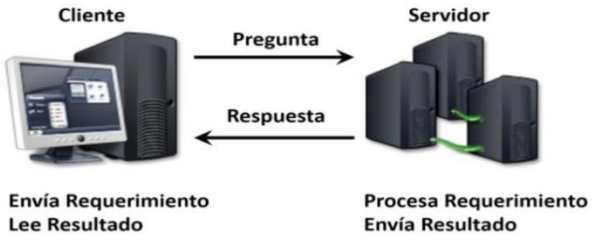


Figura 2.2 Modelo Cliente/Servidor

En el lado del cliente se definió **la arquitectura tres capas.** El estilo en capas constituye una organización jerárquica, de manera que cada capa proporciona servicios a la capa inmediatamente superior y se sirve de las prestaciones que le brinda la inmediata inferior. (58) La arquitectura en tres capas permitió distribuir el trabajo de creación de la aplicación por niveles.

La capa de Presentación es la interfaz de usuario. Esta capa se comunica con la capa de Negocio, que es donde reside el traductor y donde se encuentra la información referente a la gestión de los ejercicios. Esta capa se comunica con la capa de Presentación, para recibir las solicitudes y presentar los resultados, y con la capa de Acceso Datos, para solicitar al gestor de base de datos almacenar la información. En la capa de Acceso Datos se encuentra la información para acceder a la base de datos del sistema.

Figura 2.3 Arquitectura tres capas

Patrones de Diseño

Un **patrón de diseño** es una descripción de clases y objetos comunicándose entre sí, adaptada para resolver un problema de diseño general en un contexto particular. Contribuyen a reutilizar diseño gráfico, identificando aspectos claves de la estructura de un diseño que puede ser aplicado en una gran cantidad de situaciones. Mejoran la flexibilidad, modularidad y extensibilidad, factores internos e íntimamente relacionados con la calidad percibida por el usuario. (59)

La utilización de patrones de diseño, permite ahorrar grandes cantidades de tiempo en la construcción de software.

Los patrones utilizados en la solución son los Patrones GOF (Gang of Four), los cuales se clasifican según su ámbito y según su propósito como sigue: (59)

	Creación	Estructural	Comportamiento
Clase	Factory Method	Adapter	Interpreter Template Method
Objeto	Abstract Factory Builder Prototype Singlenton	Adapter Bridge Composite Decorator Facade Flyweight Proxy	Chain of Responsability Command Iterator Mediator Memento Observer State Strategy Visitor

Tabla 2.7 Clasificación Patrones GOF

En el sistema se usarán dentro de los Patrones Estructurales el *Composite* (Compisición) y el *Facade* (Fachada), y dentro de los Patrones de Comportamiento el *Chain of responsibility* (Cadena de responsabilidades), *Interpreter* (Intérprete) y el *Visitor* (Visitante).

Patrón Composite (Composición): Compone objetos en estructuras de árboles para representar jerarquías parte-todo. Permite a los clientes tratar uniformemente a los objetos simples y compuestos de una estructura jerárquica recursiva. En el sistema se usa en el Árbol de Sintaxis Abstracta (AST). (59) El AST se puede observar en el Epígrafe 3.1.2.

Patrón Chain of Responsability (Cadena de Responsabilidades): Proporciona a más de un objeto la capacidad de atender una petición, para así evitar el acoplamiento con el objeto que hace la petición. Se forma con estos objetos una cadena, en la cual cada objeto o satisface la

petición o la pasa al siguiente. (59)

Este patrón está relacionado con el patrón Composite. Cuando se utilizan juntos, los hijos tienen un enlace al padre que les permite acceder a información sin conocer la clase que las contiene, así como propagar la ejecución. (59) Se puede ver a la hora de manejar las responsabilidades en las clases del análisis léxico, sintáctico y semántico.

Patrón *Interpreter* (Intérprete): El patrón *Interpreter* dado cierto lenguaje, define una representación para su gramática, y usa la representación para interpretar las sentencias del lenguaje. (59) Es decir, este patrón busca definir un intérprete para dicho lenguaje, para el cual define una gramática y un intérprete de la misma para poder resolver los problemas. La gramática definida para el lenguaje AR se encuentra en el Epígrafe 3.1.2.

Patrón *Visitor* (Visitante): El patrón *Visitor* representa una operación a realizar sobre los elementos de una estructura de objetos y a su vez permite definir una nueva operación sin cambiar las clases de elementos sobre las que opera. (59) Este patrón es muy utilizado en compiladores, intérpretes y analizadores de código por las estructuras jerárquicas (árboles) que en ellos se implementa.

A continuación se muestra un fragmento del AST con el patrón Visitor:

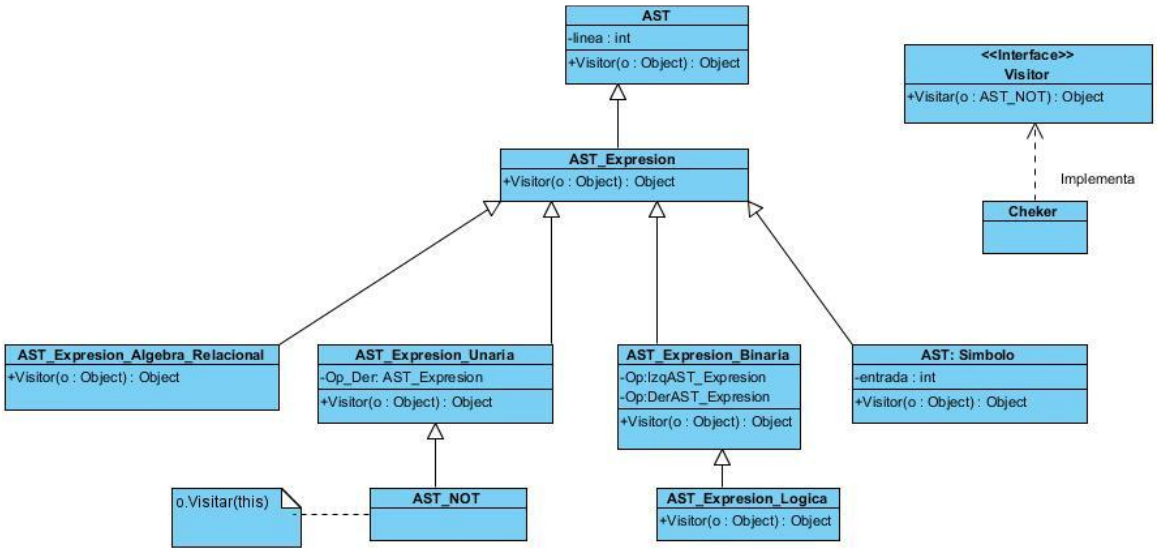


Figura 2.4 Representación de Patrón Visitor

Un **Modelo Entidad Relación (MER)** es un "*Modelo de datos basado en una percepción del mundo real que consiste en un conjunto de objetos básicos llamados entidades y relaciones entre estos objetos*". (60) Los elementos fundamentales que componen el MER son las *entidades*, los *atributos* y las *relaciones* entre las entidades.

Las **entidades** son objetos del mundo real sobre los que se queire almacenar información. Las entidades están compuestas de *atributos* que son los datos que definen el objeto. De entre los atributos habrá uno o un conjunto de ellos que no se repite; a este atributo o conjunto de atributos se le llama *clave* de la entidad. En toda entidad siempre hay al menos una clave que en el peor de los casos estará formada por todos los atributos de la tabla. (60)

Las relaciones son las asociaciones entre entidades, sin existencia propia en el mundo real que se está modelando, pero necesarias para reflejar las interacciones existentes entre entidades. Las relaciones pueden ser de tres tipos: Relaciones 1-1, Relaciones 1-n y las Relaciones n-n. (60)

A partir del MER y aplicando reglas establecidas, se obtiene el Modelo Relacional de la base de datos, que se muestra en la Figura 2.5. Este modelo conforma la BD inicial de la herramienta, creada en el esquema público; y que permitirá la gestión de la información propia que maneja la herramienta. Adicionalmente, en la BD se creará un nuevo esquema con las tablas y datos definidos para cada uno de los ejercicio que se adicionen, sobre los cuales se ejecutarán las consultas SQL para la comprobación de la solución.

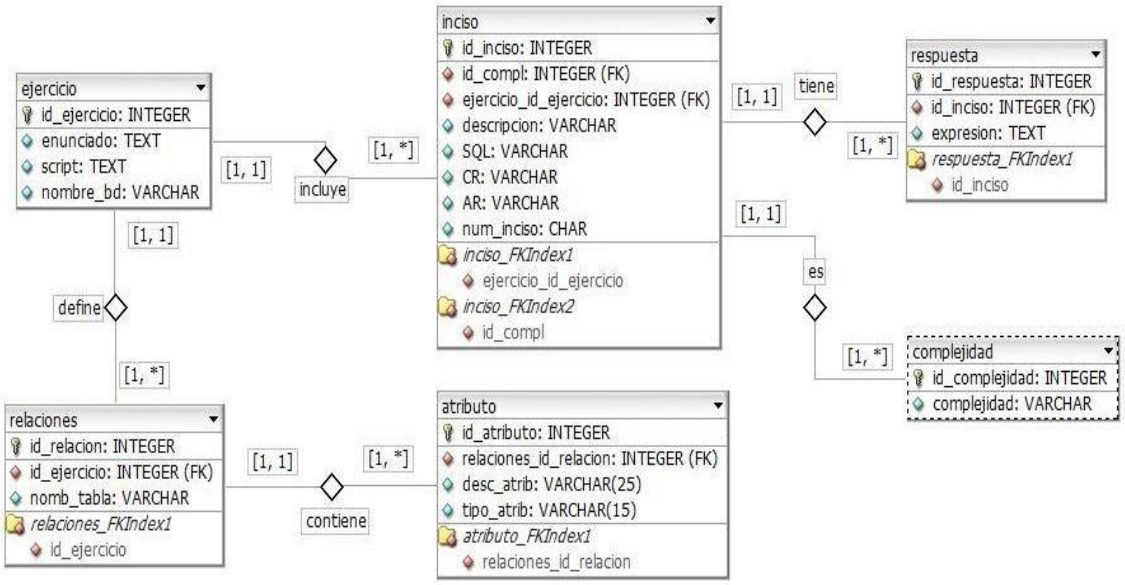


Figura 2.5 Diseño de la base de datos de la herramienta

Tarjetas CRC

El uso de las tarjetas CRC permiten al programador centrarse y apreciar el desarrollo orientado a objetos olvidándose de los malos hábitos de la programación procedural clásica. Las tarjetas CRC representan objetos; la clase a la que pertenece el objeto se escriben en la parte de arriba de la tarjeta, en una columna a la izquierda se escriben las responsabilidades u

objetivos que debe cumplir el objeto y a la derecha, las clases que colaboran con cada responsabilidad.

Clase Traductor	
Descripción: clase encargada de validar y traducir las consultas.	
Responsabilidad	**Colaborador**
Validar y traducir las consultas introducidas en el sistema.	Traductor. Conexión.

Tabla 2.8 Tarjeta CRC clase Traductor

Un **prototipo de interfaz de usuario** *"…es un modelo del comportamiento del sistema que puede ser usado para entenderlo completamente o ciertos aspectos de él y así clarificar los requerimientos… un prototipo es una representación de un sistema, aunque no es un sistema completo, posee las características del sistema final o parte de ellas"* (61)

El prototipo de interfaz de usuario permite comprobar cuestiones tales como la navegación, visualización de los atributos definidos así como la ejecución de todos los servicios. A continuación se muestra el prototipo de Interfaz de Usuario Responder ejercicio.

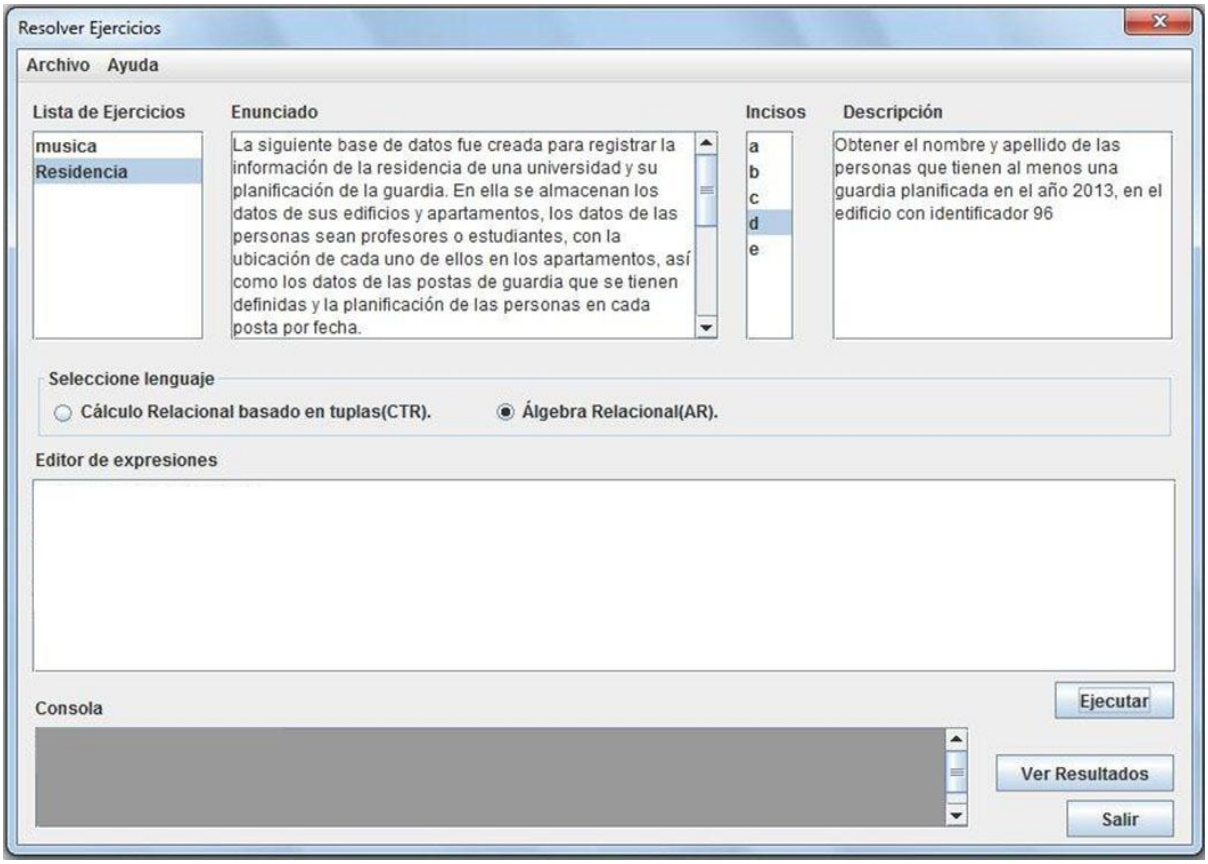


Figura 2.6 Prototipo de Interfaz de Usuario Responder ejercicio

Conclusiones Parciales

En el presente capítulo se realizó el análisis y diseño del sistema. A partir de la realización de este proceso se concluye que:

- La descripción realizada del sistema permitió al cliente intervenir en su desarrollo y adquirir un mayor conocimiento de la aplicación. En esta descripción se definieron los requisitos funcionales y no funcionales.
- Se definieron cinco (5) HU, con cuatro (4) iteraciones definidas para cada una con su correspondiente entrega.
- Se definió la duración de cada iteración y el orden de la implementación de las HU, para una correcta planificación de las iteraciones y de entrega del sistema con el objetivo de garantizar un control de las versiones a entregar.
- La planificación de entregas permitió definir una fecha límite aproximada de once (11) semanas para la culminación del producto final, realizando entregas al finalizar cada iteración.
- La arquitectura seleccionada proporciona un marco de referencia necesario para guiar la construcción del sistema.
- La utilización de los Patrones de Diseño GOF permitieron una mayor organización, mejorar la flexibilidad, modularidad y extensibilidad de la aplicación, para garantizar una mayor calidad del producto.
- Se definieron las Tarjetas CRC asociadas a la herramienta, permitiendo ello al programador realizar un inventario de las clases existentes en el sistema para facilitar una programación más organizada.

CAPÍTULO 3: Implementación y pruebas de la HEA-AR

En el presente capítulo se documentan las actividades que se generan durante el transcurso de la implementación y prueba según lo propuesto en la metodología de desarrollo XP. Se desglosan las HU en tareas de ingeniería con el objetivo de facilitar el trabajo de los desarrolladores. Se muestran los resultados obtenidos en la implementación de la herramienta y en la realización de las pruebas unitarias y de aceptación realizadas a la aplicación.

3.1. Tareas de Ingeniería

Las Tareas de Ingeniería (TI) se realizan para llevar a cabo la correcta implementación de las HU descritas por el cliente, que se realizan en cada una de las iteraciones. Las TI permiten a los desarrolladores obtener un nivel de detalle más avanzado que el que propicia las HU.

El desarrollo del software se planificó en cuatro iteraciones de trabajo. Para la HU Gestionar ejercicio\Insertar ejercicio se definió la TI que se muestra a continuación:

Tarea de Ingeniería para la Iteración #1:

HU #2: Gestionar ejercicio\Insertar ejercicio.

Tarea de Ingeniería		
Número de Tarea: 2	**Número de la HU:** 2	
Nombre de Tarea: Insertar ejercicio		
Tipo de Tarea: Desarrollo	**Fecha Inicio:** 11/3/2013	**Fecha Fin:** 15/3/2013
Programador Responsable: Adnier Castellanos Puentes Alexei Pupo Ricardo		
Descripción: El sistema muestra la opción para que el usuario (profesor) inserte los ejercicios en el sistema y ejecute el script.		

Tabla 3.1 Tarea de Ingeniería Insertar ejercicio

3.2. Implementación y validación del traductor.

Se procede a la implementación del sistema partiendo de los requisitos funcionales y siguiendo la línea de la arquitectura seleccionada para el lado del cliente. En la implementación, XP propone tener en cuenta una serie de aspectos como son la disponibilidad del cliente y el desarrollo en pareja para lograr mayores resultados en el desarrollo del software. Se puede ver como un proceso de fabricación, en el que se pone énfasis en la gestión de los recursos y el control de las operaciones para optimizar los costes, la planificación y la calidad del producto final.

En el sistema se desarrolló el Analizador Léxico, el Analizador Sintáctico el Analizador

Semántico y la Generación de Código (traducción del lenguaje AR al SQL). Dentro del Analizador Sintáctico se realizó la Gramática LL (1), y el Árbol de Sintaxis Abstracta (AST). Una vez terminado el Sistema se realizó la conexión con la base de datos utilizando el Driver JDBC.

Analizador Léxico

En esta fase se tomó la secuencia de símbolos entrada y se agruparon en entidades sintácticas simples o elementales denominadas *tokens*. Los *tokens* fueron creados mediante esquemas reconocedores de cadenas (Autómatas), obtenidos de las expresiones regulares. También se construyó la Tabla de símbolos, que se utiliza para almacenar estos *tokens* y la información pertinente al resto de las fases. Los tokens que representan constantes o identificadores se almacenan en la tabla de símbolos a medida que van apareciendo en el programa fuente.

Ejemplo de expresión en AR:

((Persona {id_edificio, id_apto} JOIN Apartamento) WHERE idedificio = idedificio) {idedificio}

Tokens reconocidos en la expresión anterior:

tk_parentesis_abierto, tk_parentesis_abierto, tk_id, tk_llave_abierta, tk_id, tk_coma, tk_id, tk_llave_cerrada, tk_operador_join, tk_id, tk_parentesis_cerrado, tk_operador_where, tk_id, tk_operador_igual, tk_id, tk_parentesis_cerrado, tk_llave_abierta, tk_id, tk_llave_cerrada.

Expresiones Regulares del lenguaje AR:

letra	→	a|…|z|A|…|Z
dígito	→	1|2|3|4|5|6|7|8|9
entero	→	(dígito)(dígito |0)*|0
parte_decimal	→	(dígito|0)*(dígito)|0
real	→	entero.parte_decimal
hora	→	‘dígito dígito : dígito dígito : dígito dígito’
fecha	→	‘dígito dígito /dígito dígito / dígito dígito dígito dígito’
cadena	→	“(letra |’ ‘| _ | dígito)*”
identificador	→	letra (letra | dígito)* letra

Tabla 3.2 Expresiones Regulares.

Ejemplo Autómata Finito Determinista para la expresión regular Identificador

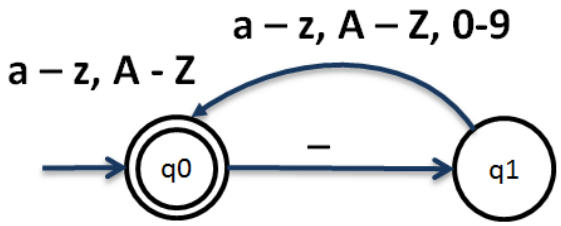


Figura 3.1 Autómata Finito Determinista para los identificadores.

```
private Token ID() throws IOException
{
    StringBuilder buf = new StringBuilder();
    do {
        buf.append(caracter_actual);
        Consumir();
    } while (esLetra());

    String lex = buf.toString();
    if (lex.charAt(0) == '_' || lex.charAt(lex.length() - 1) == '_')
    {
        lista_erores.add(new Error_lexico(entrada.getLinea_actual(), "Error Léxico: "
                + "Los ID no deben empezar ni terminar con _ " + caracter_actual));
        Token tk_Error = new Token(Tipo_token.tk_error, lex, entrada.getLinea_actual());
        Consumir();
        return tk_Error;
    }

    Tipo_token reservada = palabras_reservadas.get(lex);
    if (reservada != null)
    {
        return new Token(reservada, lex, entrada.getLinea_actual(),tabla_simbolos.Adicionar(lex, reservada));
    }
    return new Token(Tipo_token.tk_id, lex, entrada.getLinea_actual(), tabla_simbolos.Adicionar(lex, Tipo_token.
}
```

Figura 3.2 Método para reconocer los identificadores en la fase Analizador Léxico.

Analizador Sintáctico

En el análisis sintáctico se examinaron la secuencia de *tokens* generados por el Analizador Léxico, para determinar si el orden de esa secuencia es correcto de acuerdo a las reglas gramaticales definidas para el lenguaje AR. Para la correcta verificación de la sintaxis del lenguaje se implementó un analizador sintáctico descendente predictivo y recursivo que utiliza una gramática de tipo LL(1).

En las fases posteriores al análisis sintáctico es necesario conocer la estructura sintáctica de la secuencia de *tokens*. Para ello como salida del Analizador Sintáctico se produce un Árbol de Sintaxis Abstracta, que almacena los elementos significativos de la estructura del programa, así como su relación sintáctica.

Gramática Álgebra Relacional

Una gramática es un ente o modelo matemático que permite especificar un lenguaje, es decir, es el conjunto de reglas capaces de generar todas las posibilidades combinatorias de ese lenguaje, y sólo las de dicho lenguaje, ya sea este un lenguaje formal o un lenguaje natural. (36) Para describir el lenguaje de las expresiones validas en el AR se definió la gramática **GAR** = {**N**, **Σ**, **P**, **S**}, donde **N** es el conjunto de los símbolos no terminales o alfabeto de variables del lenguaje, **Σ** conjunto de símbolos terminales o alfabeto de constantes, **P** el conjunto de reglas de producción y **S** el símbolo inicial de la gramática, por donde comienza la generación de sentencias. La gramática que se muestra continuación fue producto de convertir la gramática original del lenguaje a una gramática LL(1), la que permite implementar un analizador sintáctico descendente predictivo y recursivo.

Definición de la gramática para el Álgebra Relacional

G= {N, ∑, P, S}

donde:

N = {<Programa_Algebra_Relacional>, <Declaracion_Relacion>,<Relacion_A>, <Tipo_Dato> <Más_Relación>,<ListaA_Declaracion_relacion>, <Más_Atributos>, <Declaracion_Atributo>, <Expresion_Algebra>, <ExpresiónAlgebra_1>, <Acople_División>, <Acople_División_1>, <Proyeccion>, <Proyección_1>, <Selección_1>, <Relacion>, <Star_again>, <Lista_Atributos >, <Seleccion>, <Exp>, <Expresion>, <Mas_Expresion>, <Comp_Oper>, <Valores>, <Operaciones>, <Para_Comparar>, <Sexo>, <Valores_Booleanos>, <Op_comparacion>, <Op_aritmeticos>, <Op_logicos>, <Exp_aritmetica>, <Mas_Exp_aritmetica>, <Termino>, <Mas_Termino>, <Factor>**}**

∑= {, (,), {, }, :, + ,-, *, /, =, <, >, <>, <=, >=, AND, OR, NOT, ‘, _, , True, False, M, F, UNION, INTERSECT, WHERE, DIVIDE BY, TIMES, MINUS, JOIN **}**

S = <Programa_Algebra_Relacional>

<Programa_Algebra_Relacional> → <Declaracion_Relacion> <Expresion_Algebra>

<Declaracion_Relacion> → <Relacion_A><Más_Relacion>

<Más_Relacion> → , <Declaracion_Relacion> | e

<Relacion_A> → **RELATION Id IS** (<ListaA_Declaracion_relacion>) ;

<ListaA_Declaracion_relacion> → <Declaracion_Atributo><Más_Atributos>

<Más_Atributos> → ,<ListaA_Declaracion_relacion >| e

<Declaracion_Atributo> → **Id :** <Tipo_Dato>

<Tipo_Dato> → **Int | float | String | Boolean | Char | Date | Time**

<ExpresionAlgebra> <Acople_Division> <ExpresiónAlgebra_1>

<ExpresiónAlgebra_1> → **UNION** <Acople_Division> <ExpresiónAlgebra_1> |
INTERSECT <Acople_Division> <ExpresionAlgebra_1> |
MINUS <Acople_Division> ExpresionAlgebra_1> |
TIMES <Acople_Division> ExpresionAlgebra_1> | e

<Acople_Division> → <Proyección> <Acople_Division_1>

<Acople_División_1> → **JOIN** <Proyección Acople_Division_1> |
DIVIDE BY< Proyeccion Acople_Division_1> | e

<Proyeccion> → <Seleccion> <Proyección_1>

<Proyeccion_1> → {<lista_atributos> } <Star_again> | e

<Seleccion> → <Relacion Selección_1>

<Selección_1> → **WHERE** <Expresion> <Star_again> | e

<Relacion> → **Id** | (<ExpresiónAlgebra>) | <ExpresionAlgebra>

<Star_again> → <ExpresiónAlgebra_1> | <Acople_División_1> | <Proyección_1> |
<Seleccion_1> | e

<lista_atributos> → **,** <Parámetros> | e

<Expresion> → <Comp_Oper> <Mas_Expresion> |
(<Comp_Oper>)<Mas_Expresion>

<Mas_Expresion> → <Op_logicos> <Expresion> <Mas_Expresion> | e

<Comp_Oper> → <Valores> | **NOT** (<Valores>)

<Valores> → <Exp_aritmetica> <Op_comparación> <Operaciones> <Exp>

<Operaciones> → <Exp_Aritmetica> | <Para_Comparar>

<Exp> → <Op_comparacion><Valores>|e

<Para_Comparar> → **hora | fecha |**<Valores_Booleanos>| **Literal_Cadena** | <Sexo>

<Sexo> → **M** | **F**

<Valores_Booleanos> → **True | False**

<Op_comparacion> → **= | < | > | <> | <= | >=**

<Op_aritmeticos> → **+ | - | * | /**

<Op_logicos> → **AND | OR**

<Exp_aritmetica> → <Termino> <Mas_ Exp_aritmetica >

<Mas_Exp_aritmetica> → **+** <Termino> <Mas_Exp_aritmetica > |
- <Termino> <Mas_Exp_aritmetica > | e

<Termino> → <Factor> <Mas_Termino>

<Mas_Termino> → ***** <Factor> <Mas_Termino> | **/** <Factor> <Mas_Termino> | e

<Factor> → (<Exp_aritmetica>) **| Id | Literal_entero | Literal_real |**

Tabla 3.3 Gramática LL (1) para el lenguaje AR

A continuación se muestra un ejemplo de la regla y su correspondiente implementación en el analizador sintáctico.

<Expresion_Algebra>→ <Expresion_Simple> | <Expresion_Compuesta>

```
//<Expresion_Algebra> → <Expresion_Simple> | <Expresion_Compuesta>
public AST_Expresion Expresion_Algebra_Relacional() throws IOException
{
    AST_Expresion solucion;
    if (actual.getTipo() == Tipo_token.tk_id) {
        solucion = Expresion_Simple();
        return solucion;
    } else {
        if (actual.getTipo() == Tipo_token.tk_parentesis_abierto) {
            solucion = Expresion_Compuesta();
            return solucion;
        }
    }

    return null;
}
```

Figura 3.3 Implementación de una regla de la gramática.

Fragmento de jerarquía del AST para la representación del lenguaje

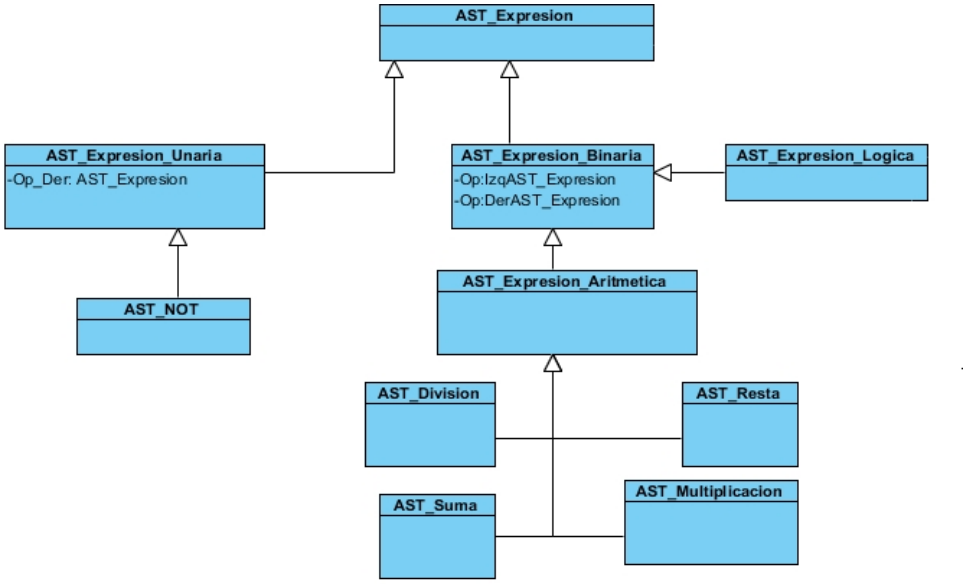


Figura 3.4 Fragmento del AST para representar las estructuras del lenguaje

Analizador Semántico

En esta fase se toma como entrada el AST generado por el Analizador Sintáctico, con la información relativa a la organización jerárquica-gramatical de los *tokens* en la instrucción que se analiza. A través del patrón de diseño *Visitor* se realiza el recorrido a cada nodo del árbol detectando las expresiones semánticamente incorrectas. Como resultado de esta fase se obtiene un AST decorado con la información adicional obtenida durante el análisis semántico y necesaria para la fase de Generación de Código.

```
public Object Visitar(AST_Programa_Algebra o) {
    try {

        relacion = o.getDeclaracion_relacion();
        if (relacion.isEmpty()) {
            reporte_errores.add(new Error_semantico(o.getLinea(), "Lista de la relacion vacía"));
        }
        for (int i = 0; i < relacion.size(); i++) {
            relacion.get(i).Visitar(this);
        }
        if (o.getExpresion_algebra_relacional() != null) {
            o.getExpresion_algebra_relacional().Visitar(this);
        } else {
            reporte_errores.add(new Error_semantico(o.getLinea(), "Debe introducir una consulta"));
        }
    }
    catch (Exception a)
    {

    }
    return null;
```

Figura 3.5 Implementación del Patrón Visitor.

Generación de código

El árbol decorado por el Analizador Semántico se utilizó para generar la traducción del programa fuente y de esta forma lograr la interpretación de las acciones asociadas a las sentencias en curso.

A continuación se muestra un ejemplo de la traducción de una expresión del AR a SQL y la estructura de la traducción de expresiones simples del lenguaje AR al SQL.

*(*Paciente ***WHERE*** *mcpio_p = "Guanabo" {id_p}* ***JOIN*** *Operacion* ***WHERE*** *causa_op = "Urgencia" {id_p, id_m}* ***JOIN*** *Medico) {id_m, nomb_m, edad_m, id_e}*
SELECT Medico.* **FROM** Medico **INNER JOIN** Operacion **USING** (id_m) **INNER JOIN** Paciente **USING** (id_p) **WHERE** mcpio_p = "Guanabo" and causa_op = "Urgencia"

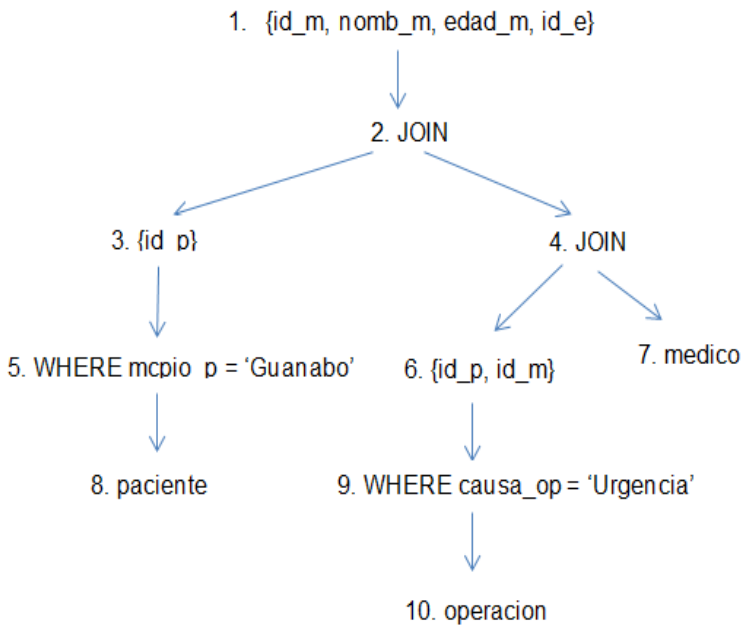
Tabla 3.4 Ejemplo traducción de expresión del AR al SQL.

RELACIÓN	
R	SELECT * FROM R
SELECCIÓN	
R WHERE (condición)	SELECT * FROM R WHERE condición
PROYECCIÓN	
R {at1, ..., atn}	SELECT R.at1, ... , R.atn FROM R
PRODUCTO CARTESIANO	
R1 TIMES R2	SELECT * FROM R1, R2
UNIÓN	
R1 UNION R2	SELECT * FROM R1 UNION (SELECT * FROM R2)
INTERSECCIÓN	
R1 INTERSECT R2	SELECT * FROM R1 INTERSECT (SELECT * FROM R2)
DIFERENCIA	
R1 MINUS R2	SELECT * FROM R1 EXCEPT (SELECT * FROM R2)
DIVISIÓN	
R1 DIVIDE BY R2	tabla1 {at1, ..., atm} Minus ((tabla1 {at1, ..., atm} Times tabla2) Minus tabla1)) {at1, ..., atm}
JUNTA NATURAL	
R1 JOIN R2	SELECT * FROM R1 INNER JOIN R2 ON (R1.atn=R2.at1)

Tabla 3.5 Modelo de la traducción de expresiones simples del AR al SQL

El recorrido utilizado para esta traducción fue el entreorden[5]. Seguidamente se muestra un ejemplo para este recorrido en la traducción.

[5] Recorrido sobre un árbol, primero se visita el nodo hijo izquierdo, luego el padre y por último el nodo hijo derecho.



1. Para 10. Operacion
2. Para 9: **SELECT** * **FROM** Operacion **WHERE** Operacion.causa_op = 'Urgencia' (se guarda en una tabla temporal tabla1)
3. Para 8: Paciente
4. Para 7: Medico
5. Para 6: **SELECT** tabla1.id_p, tabla1.id_m **FROM** tabla 1(se guarda en tabla2)
6. Para 5: **SELECT** * **FROM** paciente **WHERE** Paciente.mcpio_p = 'Guanabo' (se guarda en tabla3)
7. Para 4: **SELECT** * **FROM** tabla2 **INNER JOIN** medico **ON** (tabla2.id_m = medico.id_m) (se guarda en tabla4)
8. Para 3: **SELECT** tabla3.id_p **FROM** Paciente (se guarda en tabla5)
9. Paso 2: **SELECT** * **FROM** tabla 5 **INNER JOIN** tabla4 **ON** (tabla5.id_p = tabla4.id_p) (se guarda en tabla6)
10. Paso 1: **SELECT** tabla6.id_m, tabla6.nomb_m, tabla6.edad_m, tabla6.id_e **FROM** tabla6

Cómo realizar la traducción

Con el objetivo de realizar la traducción del lenguaje AR a SQL se hace uso del patrón *Visitor* mencionado con anterioridad y de la cláusula WITH del SQL. Esta cláusula permite reusar una

consulta *SELECT* cuando hay que utilizarla más de una vez en una sentencia o consulta SQL compleja. Los resultados de la consulta definida en el comando *WITH* son almacenados en una tabla temporal, logrando de esta forma mejorar el rendimiento de la sentencia principal. Este comando admite la definición de múltiples consultas con sólo separarlas por comas, dicho nombre será visible para todas las consultas definidas posteriormente dentro del mismo WITH. Obviamente, también será visible para la sentencia o consulta principal. (62)

Para una comprensión de este procedimiento se explica a continuación la secuencia de pasos a seguir según los operadores del Álgebra Relacional.

En el caso de los operadores de conjunto **UNION, INTERSECCION Y DIFERENCIA** el algoritmo a seguir seria el siguiente:

Paso No. 1 - Visitar la expresión izquierda de los operadores y en caso de ser una expresión simple (Empleado), se construiría una sentencia en SQL de esta forma: *SELECT* * *FROM* Empleado.

```
if (o.getOpizquierdo() instanceof AST_Ident_Referencia_Expresion_Algebra)
{
   cadena_CPI =  o.getOpizquierdo().Interfaz_Generadora(this, n,gDistinct);
}
```

Paso No. 2 - De no ser verdadera la condición del paso anterior se visita la expresión izquierda de los operadores, se crea un nombre para la tabla temporal, se obtienen los datos de la visita a la expresión conformándose una consulta similar a esta

Nombre_tabla *AS* (*SELECT* aspectos a seleccionar *FROM* expresión_izquierda)

```
tripa_der = o.getOpderecho().Interfaz_Generadora(this, n,gDistinct);
tempder = "TablaT"+ vistas_temporales++;
cadena_CPD = tempder + " AS " + " ("+ tripa_der + ") ";
temp_tripa_der = tempder;
```

Paso No.3 - Repetir los pasos 1 y 2 para la expresión derecha de estos operadores.

Paso No.4 - De ser una expresión simple en ambas expresiones (izquierda y derecha) se procede a formar una expresión en SQL similar a esta.

SELECT * FROM expresión izquierda OPERADOR[6] (Select * FROM expresión derecha)

```
if(o.getOpderecho() instanceof  AST_Ident_Referencia_Expresion_Algebra &&
   o.getOpizquierdo() instanceof  AST_Ident_Referencia_Expresion_Algebra)
{
     return "(" + cadena_CPI +")" + " OPERADOR " +"("+ cadena_CPD + ")";
}
```

[6] Está en dependencia del operador que se utilice.

Paso No.5 - De no cumplirse la condición anterior se verifica para qué miembro del operador se cumple, se crea un nombre para la tabla temporal y se forma una expresión similar a esta Nombre_tabla *AS* (*SELECT* aspectos a seleccionar *FROM* expresión)

```
if((o.getOpizquierdo() instanceof AST_Ident_Referencia_Expresion_Algebra))
{
    temp_tripa_iz = "TablaT"+ vistas_temporales++;
    cadena_CPI = temp_tripa_iz + " AS" + "("+cadena_CPI + ")";
}
if((o.getOpderecho() instanceof  AST_Ident_Referencia_Expresion_Algebra))
{
    temp_tripa_der = "TablaT"+ vistas_temporales++;
    cadena_CPD = temp_tripa_der + " AS " + "("+ cadena_CPD + ")";
}
```

Paso No.6 Devolver el resultados de las consultas formadas con anterioridad

```
solucion = "WITH " + cadena_CPI + "," + cadena_CPD + "(" + " SELECT "+ Distinc(gDistinct)+" * FROM " +
   temp_tripa_iz + ") " + " OPERADOR " + " (  SELECT"+ Distinc(gDistinct)+" * FROM " +temp_tripa_der + ")";

    return  solucion;
```

Traducción para el operador **Proyección**

Paso No.1- Verificar si el operador izquierdo de la proyección es una expresión simple, de ser afirmativo solo se obtendría el nombre de la expresión simple (nombre de la entidad en la BD) sin llamar al patrón Visitor

```
if (o.getOp_izquierdo() instanceof AST_Ident_Referencia_Expresion_Algebra)
{
    var1 = ((AST_Ident_Referencia_Expresion_Algebra) o.getOp_izquierdo()).getLexema();
}
```

Paso No.2 **-** De no cumplirse la condición anterior se llama al *Visitor*, se obtiene el SQL correspondiente, se genera un nombre para la tabla temporal, se obtiene la lista de atributos proyectados y se genera una sentencia con la cláusula *WITH*

```
temp1 = o.getOp_izquierdo().Interfaz_Generadora(this, n ,gDistinct);
var1 = "tablaT"+vistas_temporales++;
cadena = "WITH "+var1+" AS ("+temp1+")";
```

Paso No.3 - Formar la sentencia SQL final con los atributos proyectados y la expresión obtenida en el paso anterior.

```
String solucion = cadena+ " SELECT "+ Distinc(gDistinct) +" "+ lista_atrib +" FROM " + var1;
return solucion;
```

Traducción para el operador **Selección**

Paso No.1- Verificar si el operador izquierdo de la selección es una expresión simple, de ser afirmativo solo se obtendría el nombre de la expresión simple (nombre de la entidad en la BD) si la llamada al *Visitor*

```
if (o.getOp_izquierdo() instanceof AST_Ident_Referencia_Expresion_Algebra)
{
    var1 = ((AST_Ident_Referencia_Expresion_Algebra) o.getOp_izquierdo()).getLexema();
}
```

Paso No.2- De no cumplirse la condición anterior, llamar al patrón *Visitor*, obtener la consulta correspondiente a la expresión izquierda, generar un nombre para la tabla temporal y formar la consulta con la cláusula *WITH*.

```
temp1 = o.getOpizquierdo().Interfaz_Generadora(this, n,gDistinct);
var1 = "tablaT"+vistas_temporales++;
cadena = "WITH "+var1+ " AS " + "("+temp1+")";
```

Paso No.3 - Obtener la expresión derecha de la selección

```
String expresion_derecha = o.getOpderecho().Interfaz_Generadora(this, n,true);
```

Paso No.4- Formar la consulta correspondiente con la expresión izquierda y derecha del operador.

```
String result = cadena+"SELECT "+Distinc(gDistinct)+" "+lista_atrib+" FROM "+var1+" WHERE "+expresion_derecha;
return result;
```

Traducción para el operador **Producto Cartesiano**

Paso No.1- Verificar si el operador izquierdo de la proyección es una expresión simple, de ser afirmativo solo se obtendría el nombre de la expresión simple (nombre de la entidad en la BD) sin llamar al patrón.

```
if (o.getOp_izquierdo() instanceof AST_Ident_Referencia_Expresion_Algebra)
{
    var1 = ((AST_Ident_Referencia_Expresion_Algebra) o.getOp_izquierdo()).getLexema();
}
```

Paso No.2 - De no cumplirse el paso anterior llamar al patrón *Visitor*, generar un nombre a la tabla temporal y formar la cadena correspondiente con la llamada al patrón.

```
tempizq = o.getOpizquierdo().Interfaz_Generadora(this, n,gDistinct);
nomobre_izq = "TablasT"+ vistas_temporales++;
cadena_CPI = nomobre_izq + " AS " + "(" + tempizq +")" ;
nombre_tripa_izquierda = nomobre_izq;
```

Paso No.3- Repetir los pasos 1 y 2 para expresión derecha al operador.

Paso No.4- Si ambas expresiones son simples, se construye la expresión en SQL correspondiente *SELECT * FROM* expresión_ izquierda, expresión derecha.

Paso No.5- Si la condición anterior no se cumple, entonces se verifica para qué expresión (izquierda o derecha) se cumple, se genera un nombre para la tabla temporal y se construye la expresión equivalente.

```
if(o.getOpizquierdo() instanceof AST_Ident_Referencia_Expresion_Algebra )
 {
 tempizq = o.getOpizquierdo().Interfaz_Generadora(this, n,gDistinct);
 nombre_tripa_izquierda = "TablasT"+ vistas_temporales++;
 cadena_CPI = nombre_tripa_izquierda + " AS " + " "+ "(" + tempizq + ")" + "" ;
 }
```

Paso No.6- Construir la expresión final con las tablas temporales obtenidas en los pasos anteriores y la cláusula *WITH*.

```
solucion = "WITH "+cadena_CPI+","+ cadena_CPD+" SELECT "+Distinc(gDistinct)+"* "+" FROM "
       + nombre_tripa_izquierda + "," + nombre_tripa_derecha;
return solucion;
```

Traducción para el operador **JOIN**

Paso No.1- Se verifica si la expresión izquierda es simple, de ser así no se llama al *Visitor,* se obtiene el nombre de la expresión, el último atributo de la relación y se guarda el resultado de la consulta.

```
if (o.getOpizquierdo() instanceof AST_Ident_Referencia_Expresion_Algebra)
{
    cadena_CPI = ((AST_Ident_Referencia_Expresion_Algebra) o.getOpizquierdo()).getLexema();
    nombre_izq =  cadena_CPI;
}
```

Paso No.2 – De no cumplirse la condición anterior se llama el Visitor, almacenar el resultado de la consulta, se genera un nombre para la tabla temporal, se obtiene el último atributo de la relación y se construye la expresión en SQL.

```
temp1 = o.getOpizquierdo().Interfaz_Generadora(this, n,gDistinct);
nomb_izq = "tablaT" + vistas_temporales++;
cadena_CPI = nomb_izq+ " AS ("+temp1+")";
```

Paso No.3 - Repetir los pasos 1 y 2 para la expresión derecha del operador, con la excepción de tomar el primer atributo de la relación y no el último.

Paso No.4- Si las expresiones (izquierda y derecha) son simples se unen ambas expresiones en una consulta bajo las condiciones propias del operador.

```
if (o.getOpizquierdo() instanceof AST_Ident_Referencia_Expresion_Algebra &&
        o.getOpderecho() instanceof AST_Ident_Referencia_Expresion_Algebra)
{
solucion = "SELECT "+Distinc(gDistinct)+atributos_join + " FROM "+ cadena_CPI+" INNER JOIN " + cadena_CPD +
        " ON ( " + cadena_CPI + "." + ultimo_izq +" = " + nomb_der + "." + primero_der + " )";
}
```

Paso No.5- Sino se cumple el paso anterior se genera la consulta SQL con la cláusula final y las cadena formadas en cada uno de los pasos anteriores.

```
solucion = "WITH "+cadena_CPI+", "+cadena_CPD+" SELECT "+Distinc(gDistinct)+ atributos_join + " FROM "
+ nomb_izq +" INNER JOIN "+ nomb_der+ " ON ( "+nomb_izq+"."+ultimo_izq +" = "+nomb_der+"."+primero_der+" )";
```

Traducción para el operador **DIVIDE BY**

Para la traducción de este operador se recomienda la equivalencia de esta expresión

tabla1 DIVIDE BY tabla2

Expresión equivalente:

tabla1 {at1,..., atm} Minus ((tabla1 {at1,..., atm} Times tabla2) Minus tabla1)) {at1,..., atm}

Paso No.1- Obtener la lista de atributos de la expresión izquierda del operador. DIVIDE BY (Lista$_1$)

Paso No.2- Obtener la cantidad de atributos del paso anterior (cant$_2$)

Paso No.3- Obtener la cantidad de atributos de la expresión derecha del operador DIVIDE BY (cant$_3$).

Paso No.4- Obtener los atributos de la expresión derecha del operador DIVIDE BY (lista$_4$)

Paso No.5- Resta la cantidad de atributos de la izquierda con la cantidad de atributos de la derecha de las expresiones del operador DIVIDE BY (cant$_2$ - cant$_3$)

Paso No.6- Obtener la lista de atributos del operador DIVIDE BY (lista$_6$)

Paso No.7- Adicionar a una nueva lista, la obtenida en el paso anterior (lista$_7$)

Paso No.8- Adicionar a la lista creada en el paso anterior, la lista de atributos de la expresión derecha del operador DIVIDE BY (lista$_7$ + lista$_4$)

Paso No. 9- Crear una nueva lista con la n-m atributos del resultado del paso 1(lista9)

Paso No. 10- Crear un nuevo AST Expresión de tipo AST Proyección$_1$ asignándole como expresión izquierda, la expresión izquierda del operador DIVIDE BY

Paso No.11- Crear un nuevo AST Expresión de tipo AST TIMES, asignándole como expresión izquierda la expresión obtenida en el paso anterior (AST Proyección$_1$) y como expresión derecha la expresión derecha del operador DIVIDE BY.

Paso No.12- Crear un nuevo AST Expresión de tipo AST MINUS$_1$, insertándole como expresión izquierda, la expresión izquierda del operador DIVIDE BY y como expresión derecha el AST resultante en el paso anterior (AST TIMES).

Paso No.13- Crear un nuevo AST Expresión de tipo Proyección$_2$ asignándole como expresión izquierda el AST obtenido en el paso anterior (AST MINUS$_1$) y como lista de atributos proyectados la obtenida en el paso 9.

Paso No.14- Crear un nuevo AST Expresión de tipo AST MINUS$_2$ asignándole como expresión derecha el resultante del paso anterior (AST Proyección$_2$) y como expresión izquierda el resultado del paso 12 (AST MINUS$_1$).

Paso No.15- Asignar al AST obtenido en el paso No. 10(AST Proyección$_1$) la lista de atributos obtenida en la paso No.6

Paso No.16 Asignar al AST obtenido en el paso 13 la lista de atributos obtenida en el paso No.6.

Paso No.17- Asignar al AST obtenido en el paso 11 (AST TIMES) la lista de atributos resultante del paso No.8

Paso No.18 Asignar al AST derivado del paso del paso No. 12 (AST MINUS$_1$), la lista de atributos de la entidad obtenida de la expresión izquierda del operador DIVIDE BY

Paso No.19- Establecerle al AST obtenido en el paso No. 14 (AST MINUS$_2$) la lista de atributos obtenido en el paso No.6

3.3. Validación de la propuesta.

Pruebas Unitarias

Las pruebas unitarias son las encargadas de verificar el código y son diseñadas por los programadores. Cada uno de los desarrolladores tiene que ir probando constantemente lo que va obteniendo en el transcurso de la implementación de un sistema, para garantizar que las funcionalidades exigidas por el cliente estén siendo implementadas correctamente. (63) Las pruebas unitarias se fueron desarrollando a medida que se terminaba de implementar alguna funcionalidad, probándola directamente en el entorno real, para lo cual se utilizó la librería JUnit.

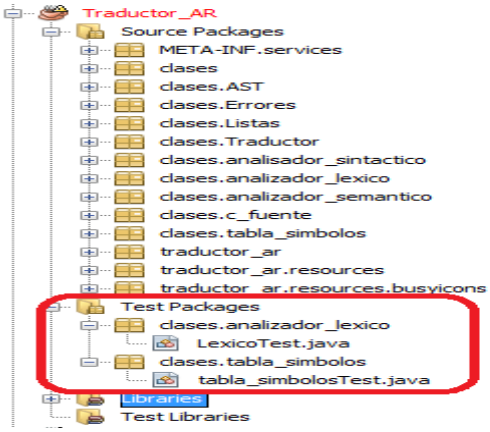


Figura 3.6 Pruebas Unitarias a la Herramienta

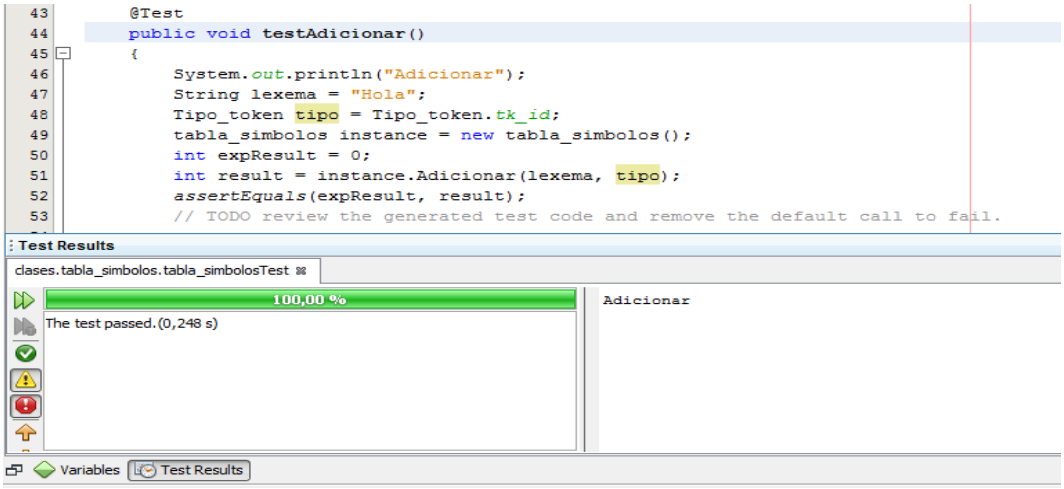


Figura 3.7 Pruebas Unitarias Método Adicionar de la Tabla de Símbolos

Pruebas de Aceptación

Las pruebas de aceptación se elaboran a lo largo de la iteración, en paralelo con el desarrollo del sistema, y adaptándose a los cambios que el sistema sufra. (63) Para el desarrollo de las mismas se utilizó el tipo de pruebas funcionales, las cuales prueban todas las capas de la aplicación. Dentro de este tipo de prueba, se utilizó el método de caja negra y de forma más específica, la técnica de partición equivalente

Las pruebas caja negra se concentran en los requisitos funcionales de un software y se llevan a cabo sobre su interfaz, obviando el comportamiento interno y la estructura del programa. Los casos de prueba de la caja negra tratan encontrar errores de la siguiente categoría: (37)

1) funciones incorrectas o faltantes

2) errores de interfaz

3) errores en estructura de datos o en acceso a bases de datos alternas,

4) errores de comportamiento o desempeño

5) errores de inicialización y término.

La partición equivalente divide el dominio de entrada de un programa en clases de datos a partir de los cuales pueden derivarse casos de pruebas. Cada una de estas clases de equivalencia representa a un conjunto de estados válidos o inválidos para las condiciones de entrada. (37)

Las pruebas de aceptación se registran en tablas, las cuales están divididas en las siguientes secciones:

- ✓ **Clases Válidas**: descripción de cada uno de los pasos seguidos durante el desarrollo de la prueba, se tendrá en cuenta cada una de las entradas válidas que hace el usuario con el objetivo de ver si se obtiene el resultado esperado.
- ✓ **Clases Inválidas**: descripción de cada uno de los pasos seguidos durante el desarrollo de la prueba, se tendrá en cuenta cada una de las posibles entradas inválidas que hace el usuario con el objetivo de ver si se obtiene el resultado esperado y cómo responde el sistema.
- ✓ **Resultado Esperado**: breve descripción del resultado que se espera ya sea para entradas válidas o entradas inválidas.
- ✓ **Resultado de la Prueba**: breve descripción del resultado que se obtiene.
- ✓ **Observaciones**: algún señalamiento o advertencia que sea necesario hacerle a la sección que se está probando.

Resultados de las pruebas de aceptación para cada Iteración de prueba

- A las HU pertenecientes a la iteración 1 se les desarrollaron 3 iteraciones de pruebas y como resultado se detectaron 2 no conformidades para la primera iteración, 1 para la segunda, y en la tercera, todas estas no conformidades fueron mitigadas.
- A la HU correspondiente a la iteración 2 se le desarrolló 2 iteraciones de pruebas y como resultado se detectó 1 no conformidad que fue resuelta en la iteración 2.
- A la HU perteneciente a la iteración 3 se le desarrolló 3 iteraciones de pruebas y como resultado se detectaron 2 no conformidades para la primera iteración, 1 para la segunda, y en la tercera, todas estas no conformidades fueron eliminadas.
- A la HU de la iteración 4 se le desarrolló 2 iteraciones de prueba y como resultado se detectaron 2 no conformidades que fueron resueltas en la iteración 2.

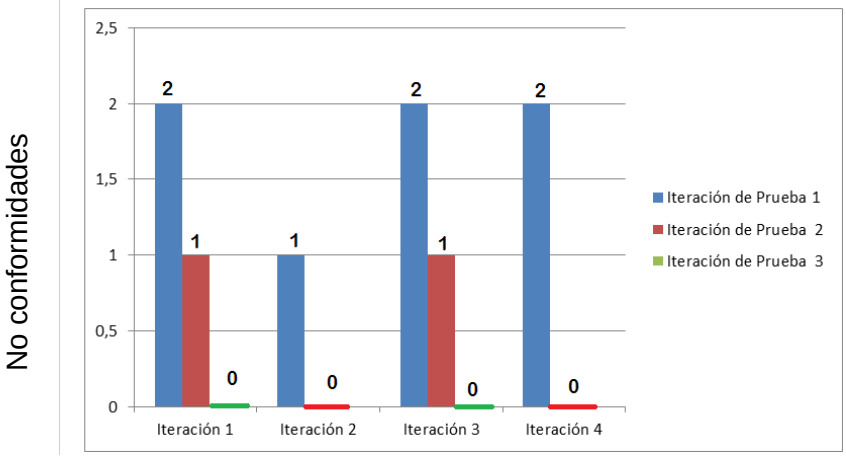


Figura 3.8 Resultados de las iteraciones de pruebas.

No conformidades

No.	No Conformidad	Ubicación	Estado
1	No se muestra un mensaje indicando que se ha realizado la conexión con la BD	Prueba de Aceptación Iteración #1, HU Conectar BD.	Resuelta
2	No se elimina el ejercicio seleccionado de la BD	Prueba de Aceptación Iteración #1, HU Gestionar ejercicios\ Eliminar ejercicio.	Resuelta
3	No se muestra un mensaje indicando que no existen ejercicios en la BD.	Prueba de Aceptación Iteración #1, HU Gestionar ejercicios\ Listar ejercicios.	Resuelta
4	No se muestra un mensaje indicando que no seleccionó el inciso a resolver	Prueba de Aceptación Iteración #2, HU Responder ejercicio	Resuelta
5	No se muestran los errores léxicos.	Prueba de Aceptación Iteración # 3, HU Traducir solución del lenguaje Álgebra Relacional a SQL	Resuelta
6	No se muestran los errores semánticos.	Prueba de Aceptación Iteración # 3, HU Traducir solución del lenguaje Álgebra Relacional a SQL	Resuelta
7	No traduce el operador JOIN	Prueba de Aceptación Iteración # 3, HU Traducir solución del lenguaje Álgebra Relacional a SQL	Resuelta
8	No realiza la comparación entre los resultados de la consulta	Prueba de Aceptación Iteración # 4, HU Comprobar Solución.	Resuelta
9	No se muestra un mensaje especificando que los resultados de las consultas no coinciden	Prueba de Aceptación Iteración # 4, HU Comprobar Solución.	Resuelta

Tabla 3.6 No conformidades detectadas

A continuación se muestran los resultados de aplicar las Pruebas de Aceptación en la HU Gestionar ejercicio\Eliminar ejercicio correspondiente a la Iteración #1

Pruebas de Aceptación Iteración #1: HU Gestionar ejercicio\Eliminar ejercicio

Clases Válidas	Clases Inválidas	Resultado Esperado	Resultado de la Prueba	Observaciones
El usuario Profesor accede a la interfaz Gestionar ejercicio\ Eliminar ejercicio y selecciona el ejercicio que desee eliminar y presiona el botón Eliminar.		El sistema verifica que el ejercicio esté seleccionado y lo elimina de la BD.	El sistema verifica que el ejercicio esté seleccionado y lo elimina de la BD.	
	El usuario Profesor accede a la interfaz Gestionar ejercicio\ Eliminar ejercicio y no selecciona el ejercicio que desee eliminar y presiona el botón Eliminar.	El sistema muestra un cartel indicando que debe seleccionar el ejercicio que desee eliminar.	El sistema muestra un cartel indicando que debe seleccionar el ejercicio que desee eliminar.	

Tabla 3.7 Prueba Aceptación Eliminar ejercicio.

Conclusiones Parciales

En este capítulo se realizó un análisis del desarrollo de la herramienta, se realizaron las tareas de ingeniería que dieron solución a todas las HU logrando una mayor organización y rapidez en el desarrollo del sistema. Se realizaron las pruebas Unitarias y las pruebas de Aceptación a las Historias de Usuarios implementadas en cada una de las iteraciones, mostrándose un resultado de ambas pruebas.

- A partir de la gramática definida con la notación establecida por el Departamento Docente Metodológico de Ingeniería de Software para el Álgebra Relacional, se identificaron las expresiones regulares para cada *token*. Esta gramática se transformó en una gramática LL(1), ello permitió implementar la correcta traducción del lenguaje Álgebra Relacional al SQL.
- Las quince (15) Tareas de Ingeniería realizadas permitieron obtener un nivel de detalle más avanzado al brindado por las HU, garantizándose su implementación.
- La realización de las pruebas Unitarias al código, permitió a los programadores una inmediata retroalimentación de su trabajo y eliminar los errores.
- De manera general, durante la realización de las pruebas de aceptación realizadas a la herramienta, se detectaron nueve no conformidades en las cuatro iteraciones de desarrollo, todas las no conformidades fueron resueltas satisfactoriamente.

CONCLUSIONES

Una vez concluida la investigación e implementación del sistema: Evaluador AR: Herramienta para la Evaluación Automatizada de Expresiones del lenguaje Álgebra Relacional, se da cumplimiento a la problemática planteada y al objetivo general y específicos arribando a las siguientes conclusiones:

- En el desarrollo de la presente investigación, a través de la revisión documental realizada, se evidenció el importante papel que cumplen las TIC en la educación, lo cual propicia el desarrollo de las TE que generan nuevos entornos de aprendizaje en la evaluación automatizada.
- Se asume que la evaluación automatizada es aquella que se realiza a través de un sistema informático. A partir de su aplicación se puede contribuir a mejorar el proceso de aprendizaje de los estudiantes, brindándoles respuestas de una manera más inmediata y favoreciendo la interactividad. A partir de la introducción de las TIC en la educación, su utilización constituye un elemento diferenciador respecto a las prácticas evaluativas convencionales.
- El estudio realizado de las herramientas informáticas existentes para la evaluación automatizada de expresiones del Álgebra Relacional, demostró la necesidad de crear una aplicación de este tipo, esto se fundamenta en que las herramientas analizadas no cumplen con los requisitos impuestos por el Departamento Metodológico Central de Ingeniería de Software. Los requisitos más importantes en este sentido son utilizar la notación definida por el departamento y realizar la traducción al SQL.
- Siguiendo las especificaciones del Departamento, y a partir del análisis del proceso de desarrollo de software, se seleccionaron: XP como metodología de desarrollo de software, Visual Paradigm for UML y DBDesigner Fork, como herramientas de modelado, PostgreSQL como SGBD, Java como lenguaje de programación, JDBC4 como librería para establecer la comunicación con la BD, NetBeans como IDE, Adobe Robohelp para la creación y edición de la ayuda y JUnit para la realización de las pruebas unitarias.
- Se realizó el proceso de desarrollo del software siguiendo como guía la metodología XP, paro lo cual se desarrollaron cinco historias de usuario para describir el sistema en el lenguaje del cliente; se realizó la estimación de esfuerzo, que arrojó un tiempo de desarrollo de once semanas y la planificación de cuatro entregas. El sistema cuenta una

arquitectura Cliente-Servidor y para el lado del cliente una arquitectura en tres capas. Para su implementación se utilizaron los Patrones GOF.

- Se logró la implementación de un traductor que realiza el análisis de las expresiones en Álgebra Relacional y las traduce a SQL. Este traductor puede ser integrado a otras herramientas.
- La solución final cuenta con dos aplicaciones, una de ellas para la gestión de los ejercicios por parte de los profesores y la otra permite al estudiante resolverlos, retroalimentándolo con los errores cometidos y/o con los resultados de la consulta. Ambas aplicaciones son de escritorio y cuentan con una interfaz gráfica que facilita el entendimiento del proceso.
- Se realizaron pruebas unitarias para validar la implementación del código y de aceptación para validar las Historias de Usuario definidas. En este proceso se detectaron nueve no conformidades las cuales se logró corregir y/o validar correctamente. Lo anterior garantizó que los requerimientos de software se cumplieran, validándose la herramienta como un producto de software.
- Se obtuvo como principal resultado el desarrollo de una herramienta que permite a los estudiantes autoevaluar su aprendizaje del contenido Álgebra Relacional en la asignatura Sistemas de Bases de Datos 1, sin depender de la presencia del profesor o de que este realice la corrección de sus ejercicios, esto facilita el proceso de enseñanza aprendizaje, en particular el desarrollo de la habilidad de formular expresiones en Álgebra Relacional para recuperar información.

RECOMENDACIONES

Una vez desarrollado el sistema y expuestas las conclusiones del trabajo se recomienda que:

- Adicionar las operaciones SEMIJOIN, SEMIMINUS, EXTEND y SUMARIZE, utilizando para su implementación la estrategia empleada en el operador DIVIDE BY.
- Integrar la herramienta al Entorno Virtual de Enseñanza Aprendizaje mediante un *applet* de java con el núcleo del traductor, para utilizar las funcionalidades que brinda el EVEA para la gestión de usuarios, gestión de reportes, gestión de calificaciones, etc.
- Realizar la validación de incisos cuyo conjunto de resultado es vacío y en los cuales no se puede comparar los conjuntos de datos de la consulta construida a partir de la expresión del álgebra y de la solución proporcionada por el profesor.
- Realizar una implementación para validar los resultados a partir de la equivalencia entre dos AST que representen a la misma solución.
- Mostrar recomendaciones a los estudiantes sobre los elementos que hacen que sus respuestas no sean correctas, para fortalecer la retroalimentación.
- Utilizar técnicas de Inteligencia Artificial, para validar las soluciones del estudiante, a partir de la solución introducida por el profesor en Álgebra Relacional.
- Incluir en la herramienta del profesor la posibilidad de realizar análisis sobre los resultados de un determinado ejercicio.

REFERENCIAS BIBLIOGRÁFICAS

1. **Sánchez Rodríguez, Alejandro , Oseguera Ríos, Emmanuel and Galeana de la O , Lourdes.** Universidad de Colima. *Sitio web del Centro Interactivo de Aprendizaje Multimedia.* [Online] CEUPROMED, 2006. [Cited: Noviembre 3, 2012.] http://ciam.ucol.mx/articulo_red.php.
2. **Fernandez, A.** Sociedad de la Información. *Sitio web de la Universitat Oberta de Catalunya.* [Online] Noviembre 26, 2088. [Cited: Noviembre 3, 2012.] http://ictconsequences.net/uoc/sociedadinformacion/2008/11/26/importancia-de-las-tic-en-la-educacion/.
3. **Cueva Carrión, Samanta Patricia, et al., et al.** Colombia Digital. *Sitio web de Colombia.* [Online] Enero 2009. [Cited: Noviembre 3, 2012.] http://www.colombiadigital.net/newcd/dmdocuments/Doc%2010196%20%28Tecnolog%C3%ADas%20de%20Informaci%C3%B3n%20y%20Comunicaci%C3%B3n%20%28TIC%C2%B4s%29%20en%20la%20.pdf.
4. **Suárez Guerrero, Cristóbal.** campus.usal. *Sitio web de la Universidad de Salamanca.* [Online] Ediciones Universidad de Salamanca, 2012. [Cited: Noviembre 3, 2012.] http://campus.usal.es/~teoriaeducacion/rev_numero_04/n4_art_suarez.htm.
5. **Ortega Escalante, Elodia.** Procesos de Formación de Espacios Virtuales. *Sitio web de la Universidad de Salamanca.* [Online] [Cited: Noviembre 6, 2012.] http://noesis.usal.es/Documentos/ARTICULOS%20EDUCARE%202003/Qu%E9%20es%20Tecnolog%EDa%20Educativa.pdf.
6. **Ballester Gouraige, Andres and Gómez García, Alexis.** Revista Edusol. *Sitio web de la Universidad de Ciencias Pedagógicas "Raúl Gómez".* [Online] 2002. [Cited: Noviembre 6, 2012.] http://www.revistaedusol.rimed.cu/articulos/vol_3_2002/art_ballester.pdf.
7. **Farell Vázque, Lic. Guillermo E.** Infomed. *Sitio web de la Red de salud de Cuba.* [Online] Octubre 18, 2004. [Cited: Noviembre 6, 2012.] http://www.sld.cu/galerias/pdf/sitios/prevemi/cursovirtual.pdf.
8. **Arias Guerra, Yuliane and Arias Gerra, Yusel.** Informática Jurídica. *Sitio web de la Revista Informática Jurídica.* [Online] 2012. [Cited: Noviembre 7, 2012.] http://www.informatica-juridica.com/trabajos/Conceptualizacion_de_una_red_social_Educativa.pdf.
9. **Diaz Sardiñas, Adolfo.** *Modelo del Profesional y Objetivos de la carrera de Ingeniría en Ciencias Informáticas.* La Habana : s.n., 2010.

10. **Hernández Saldaña, Jesús Ernesto .** Zona Porxim@. *Sitio web de la Universidad Regiomontana de Mexico.* [Online] Noviembre 5, 2001. [Cited: Noviembre 10, 2012.] http://www.ur.mx/ligas/zonaproxima/articulos/publicacion01/art-03.htm.
11. **Departamento de Ingeniería de Software.** Intranet. *Sitio web Universidad de las Ciencias Informáticas.* [Online] 2012. [Cited: Noviembre 9, 2012.] http://intranet2.uci.cu/node/86/ingenieria-software.
12. **Date, C.J.** *Introducción a los Sistemas de Bases de Datos.* México : Pearson Educación, 2001.
13. **Pons Capote, Olga.** *Introducción a las Bases de Datos.* Madrid : Thompson Editores Spain, 2005.
14. *La evaluación del aprendizaje. Tendencias y reflexión crítica.* **González, Miriam.** 1, La Habana : s.n., 2000, Revista Cubana de Educación Superior, Vol. XX, pp. 47-62.
15. **Bases de Datos.** Bases de Datos. [Online] Noviembre 4, 2009. [Cited: Noviembre 9, 2012.] http://unefabasededatos2009.blogspot.com/2009/04/conceptos-basicos-modelo-relacional.html..
16. **Gómez Pérez, José Ramón.** Las TIC en Educación. . *sitio web de Gómez Pérez, José Ramón: Las TIC en la educación.* [Online] 2013. [Cited: Marzo 25, 2013.] http://boj.pntic.mec.es/jgomez46/ticedu.htm..
17. **EVEA.** EVEA. *Sitio web EVEA.* [Online] 2012. [Cited: Noviembre 27, 2012.] http://eva.uci.cu/file.php/180/2._Clases/Tema_2/Materiales_basicos/6.Algebra_Relacional.pdf.
18. **FERNÁNDEZ MORANTE, M. CARMEN and CEBREIRO LÓPEZ, BEATRIZ .** Universidad de Sevilla. *Sitio Web de la Universidad de Santiago de Compostela.* [Online] 2012. [Cited: Noviembre 25, 2012.] http://www.sav.us.es/pixelbit/pixelbit/articulos/n21/n21art/art2107.htm.
19. **Gras Martí, Albert , et al., et al.** agm.cat. [Online] [Cited: Noviembre 26, 2012.] http://agm.cat/recerca-divulgacio/Evaluacion_ej_TIC_EnsCC_Exp_M-12ComPedag2003.pdf.
20. **Soler Masó, Josep .** *Entorno Virtual para el Aprendizaje y la Evaluación Automática en Bases de Datos.* Girona, España : s.n., 2010. Tesis doctoral. ISBN: 978-84-694-0260-3.
21. **González Vallejo, Leandro , et al., et al.** *MÓDULO PARA LA DETECCIÓN DE PLAGIO EN EL JUEZ EN LÍNEA CARIBEÑO.* Universidad de las Ciencias Informáticas. La Habana : s.n. ISBN: 978-959-7213-02-4.
22. **Garrido Picaso, Piedad, Fuentes Muñoz, Gabriel and Tramullas Saz, Jesús.** *Una Herramienta para la evaluación automatizada de la disciplina de Bases de Datos.* Zaragoza, España : s.n.

23. *Towards automated assessment of engineering assigments.* **Tong-Seng, Jon, et al., et al.** Atlanta : s.n., 2009. Proceedings of International Joint Conference on Neural Networks.
24. **Yus Peirote, Roberto and Ilarri Artigas, Sergio.** UZanjuán. *Sitio web de la Universidad de Zaragoza.* [Online] 2010. [Cited: Diciembre 2, 2012.] http://zaguan.unizar.es/record/4695#.
25. **Hernández, Carmen, et al., et al.** Departamento de Informática de la Universidad de Valladoid. *sitio web de Departamento de Informática de la Universidad de Valladoid. España.* [Online] 2002. [Cited: Noviembre 15, 2012.] http://www.giro.infor.uva.es/oldsite/docpub/JenuiDefinitivo.pdf.
26. **Appel, Ana P. y Traina, Caetano Jr.** LBD27. *sitio web de LBD27.* [Online] 2003. [Cited: Noviembre 15, 2012.] http://www.lbd.dcc.ufmg.br/colecoes/weirjes/2004/006.pdf..
27. **LEAP RDBMS.** LEAP RDBMS. [Online] 2012. [Cited: Diciembre 3, 2012.] http://leap.sourceforge.net/.
28. **Gómez Gauchia, Héctor.** *Mapas Conceptuales como primera fase del Diseño de Bases de Datos.* Madrid : s.n., 2012.
29. **Yus Peirote, Roberto.** *Herramienta para el aprendizaje del álgebra relacional y optimización de consultas.* Departamento de Informática e Ingeniería de Sistemas, Universidad de Zaragoza. Zaragoza, España : s.n., 2010. Tesis de grado.
30. **Unciencia.** Unciencia. *Sitio Web de la Universidad Nacional de Costa Rica.* [Online] 2012. [Cited: Diciembre 5, 2012.] http://www.revistas.una.ac.cr/index.php/uniciencia/index.
31. **Wallace Harris, Steven. RAIN.** Computing Science and Mathematics. *sito web de Computing Science and Mathematics, University of Stirling.* [Online] 2006. [Cited: Noviembre 17, 2012.] www.cs.stir.ac.uk/.../Harris.RTF.
32. **Mitra, Pritam.** *Relational Algebra Learning Tool.* Department of Computing, Imperial College. Londres : s.n., 2009. Tesis de grado.
33. **Beynon, Meuring.** *A computer-based environment for the study of relational query language.* Coventry, UK : s.n., 2003.
34. **Soler, J.** *Entorno virtual para el aprendizaje y evaluación automática de bases de datos.* Girona, España : s.n., 2010. Tesis doctoral. ISBN: 978-84-694-0260-3.
35. **Perdomo Velázquez , Yubismel.** *Herramienta para la creación y ejecución de consultas en Álgebra Relacional.* Universidad de las Ciencias Informáticas. La Habana : s.n., 2013. Serie Científica. ISSN: 2306-2495 | RNPS: 2343.
36. **Acevedo Martíınez, Dr. Liesner and Osorio Ramíırez, Msc. Karel .** *Técnicas de Compilación: Manual Práctico para estudiantes de In-formática.* La Habana : s.n., 2011.

37. **S. Pressman, Roger.** *Software Engineering. A Practitioner's Approach.* New York : s.n., 2006. ISBN 978-0-07-337597-7.
38. **Carrillo Pérez, Isaías, Pérez González, Rodrigo and Rodríguez Martín, Aureliano David.** [Online] Octubre 15, 2008. [Cited: Diciembre 10, 2012.]
39. **Brito Acuña, Karenny.** eumed.net. *BIBLIOTECA VIRTUAL de Derecho, Economía y Ciencias Sociales.* [Online] 2012. [Cited: Diciembre 10, 2012.] http://www.eumed.net/libros-gratis/2009c/584/Metodologias%20tradicionales%20y%20metodologias%20agiles.htm.
40. **Canós, José H., Letelier, Patricio and Penadés, María del Carmen.** *Metodologías Agiles de Desarrollo de Software.* Valencia : s.n.
41. **Joskowicz, Ing. Jose.** *Reglas y Prácticas en eXtreme Programming.* España : s.n., 2008.
42. **Hernández Orallo, Enrique.** *El Lenguaje Unificado de Modelado (UML).* España : s.n., 2013.
43. **Visual Paradigm for UML.** Visual Paradigm for UML. *sitio web de Visual Paradigm for UML.* [Online] 2012. [Cited: Noviembre 25, 2012.] http://www.visual-paradigm.com/product/vpuml/.
44. **PETKOVIĆ, DUŠAN.** *Microsoft SQL Server 2005: a beginner's guide.* 2005.
45. **More Palacios, Cristopher .** SlideShare. [Online] Junio 15, 2012. [Cited: Enero 19, 2013.] http://www.slideshare.net/crisge21/sistemas-gestores-de-bases-de-datos-13334896.
46. **Softonic.** Softonic. *sitio web de Softonic.* [Online] 2013. [Cited: Abril 27, 2013.] http://dbdesigner.softonic.com/.
47. **Mixprogramas.** Mixprogramas. *sitio web de Mixprogramas.* [Online] 2013. [Cited: Abril 27, 2013.] http://www.mixprogramas.com/db-designer-fork-1-3a/ .
48. Departamento de Informática. *Sitio web de la Universidad de Valladolid.* [Online] 2013. [Cited: Enero 19, 2013.] http://www.infor.uva.es/~jmrr/TAD2003/Sesiones/TADONJava/JAVA.html.
49. **Universidad del Valle.** Universidad del Valle. *Sitio web de la Universidad del Valle de Colombia.* [Online] 2013. [Cited: Abril 25, 2013.] http://eisc.univalle.edu.co/materias/BD/IntroduccionAlJDBC.pdf.
50. **Redes Zone.** Redes Zone. [Online] 2013. [Cited: Enero 19, 2013.] http://www.redeszone.net/2012/07/27/netbeans-7-2-nueva-version-de-este-popular-ide/.
51. **Java.net.** Java.net. *sitio web de The Source for Java Technology Collaboration.* [Online] 2013. [Cited: Junio 6, 2013.] https://javahelp.java.net/.
52. **Ramos, Rafael Aldo.** compujuy. [Online] 2013. [Cited: Abril 10, 2013.] http://www.compujuy.com.ar/.

53. *Las Metodologías de Desarrollo Ágil como una Oportunidad para la Ingeniería del Software Educativo.* **Orjuela Duarte, Ailin and Rojas C., Mauricio.** 2, Colombia : ISSN, 2008, Vol. 5.

54. **Programación Extrema.** Programación Extrema. [Online] 2013. [Cited: Marzo 2, 2013.] http://programacionextrema.tripod.com/fases.htm.

55. **oness.sourceforge.** oness.sourceforge. [Online] 2013. [Cited: Marzo 2, 2013.] http://oness.sourceforge.net/docbook/planificacion_entrega.html.

56. **webs.uvigo.** webs.uvigo. *Sitio Web de la Universidad de Vigo.* [Online] 2013. [Cited: Marzo 6, 2013.] http://webs.uvigo.es/pmayobre/06/arch/profesorado/maria_mendez/2_guia_practica.pdf.

57. **catarina.udlap.** catarina.udlap. [Online] 2013. [Cited: Marzo 6, 2013.] http://catarina.udlap.mx/u_dl_a/tales/documentos/lis/marquez_a_bm/capitulo5.pdf.

58. **Reynoso, Carlos and Kiccillof, Nicolás.** *Estilos y Patrones en la Estrategia de Arquitectura de Microsft.* Buenos Aires : s.n., 2004.

59. **Gamma, Erich, et al., et al.** *Design patterns: elements of reusable object-oriented software.* Boston, USA : Addison-Wesley Longman Publishing Co., 1995.

60. **Silberschatz, Abraham, et al., et al.** *Fundamentos de Bases de Datos.* 1998.

61. **Ingenieria en Sistemas.** Ingenieria en Sistemas. [Online] 2013. [Cited: Marzo 10, 2013.] http://sistemas2009unl.wordpress.com/prototipos-informaticos/ .

62. **PostgreSql.** PostgreSql. *sitio web de PostgreSql.* [Online] The PostgreSQL Global Development Group, 2013. [Cited: Junio 5, 2013.] http://www.postgresql.org/docs/9.1/interactive/queries-with.html.

63. **Pérez, Isaías Carrillo.** *METODOLOGIA DE DESARROLLO DE SOFTWARE.* 2005.