Université de Montréal

Una aproximación evolucionista para la generación automática de sentencias SQL a partir de ejemplos

par
Dania Isabel Ahumada Pardo

Département d'informatique et de recherche opérationnelle
Faculté des arts et des sciences

Thèse présentée à la Faculté des arts et des sciences
en vue de l'obtention du grade de Maîtrise (M.Sc.)
enInformatique

19 Mars, 2015

Université de Montréal
Faculté des études supérieures

Cettethèseintitulée:

Una aproximación evolucionista para la generación automática de sentencias SQL a partir de ejemplos

Présentée par
Dania Isabel Ahumada Pardo

A été évaluée par un jury composé des personnes suivantes:

Thèseacceptée le:……....

Resumen

En la actualidad, el uso de las tecnologías ha sido primordial para el avance de las sociedades, estas han permitido que personas sin conocimientos informáticos o usuarios llamados "no expertos" se interesen en su uso, razón por la cual los investigadores científicos se han visto en la necesidad de producir estudios que permitan la adaptación de sistemas, a la problemática existente dentro del ámbito informático.

Una necesidad recurrente de todo usuario de un sistema es la gestión de la información, la cual se puede administrar por medio de una base de datos y lenguaje específico, como lo es el SQL (*StructuredQueryLanguage*), pero esto obliga al usuario sin conocimientos a acudir a un especialista para su diseño y construcción, lo cual se ve reflejado en costos y métodos complejos, entonces se plantea una pregunta ¿qué hacer cuando los proyectos son pequeñas y los recursos y procesos son limitados?

Teniendo como base la investigación realizada por la universidad de Washington[39], donde sintetizan sentencias SQL a partir de ejemplos de entrada y salida, se pretende con esta memoria automatizar el proceso y aplicar una técnica diferente de aprendizaje, para lo cual utiliza una aproximación evolucionista, donde la aplicación de un algoritmo genético adaptado origina sentencias SQL válidas que responden a las condiciones establecidas por los ejemplos de entrada y salida dados por el usuario.

Se obtuvo como resultado de la aproximación, una herramienta denominada EvoSQL que fue validada en este estudio. Sobre los 28 ejercicios empleados por la investigación [39], 23 de los cuales se obtuvieron resultados perfectos y 5 ejercicios sin éxito, esto representa un 82.1% de efectividad. Esta efectividad es superior en un 10.7% al establecido por la herramienta desarrollada en [39]SQLSynthesizer y 75% más alto que la herramienta siguiente más próxima Queryby Output QBO[31].

El promedio obtenido en la ejecución de cada ejercicio fue de 3 minutos y 11 segundos, este tiempo es superior al establecido por SQLSynthesizer; sin embargo, en la medida un algoritmo genético supone la existencia de fases que amplían los rangos de tiempos, por lo cual el tiempo obtenido es aceptable con relación a las aplicaciones de este tipo.

En conclusión y según lo anteriormente expuesto, se obtuvo una herramienta automática con una aproximación evolucionista, con buenos resultados y un proceso simple para el usuario "no experto".

Mots-clés : Programación Genética, Sentencias SQL, Generación a partir de ejemplos, Ingeniería del software.

Résumé

Actuellement l'usage des technologies est primordial pour l'avance de la société, celles-ci ont permis que des personnes sans connaissances informatiques ou des utilisateurs appelés "non expert" s'intéressent à son usage. C'est la raison pour laquelle les enquêteurs scientifiques se sont vus dans la nécessité de produire les études qui permettent l'adaptation des systèmes à la problématique existante à l'intérieur du domaine informatique.

Une nécessité récurrente pour tout utilisateur d'un système est la gestion de l'information, que l'on peut administrer au moyen d'une base de données et de langage spécifique pour celles-ci comme est le SQL (StructuredQueryLanguage), mais qui oblige à l'utilisateur à chercher un spécialiste pour sa conception et sa construction, et qui représente des prix et des méthodes complexes. Une question se pose alors,quoi faire quand les projets sont petites et les ressourceset les processus limités ?

Ayant pour base la recherche de l'université de Washington [39], ce mémoire automatise le processus et applique une différente technique d'apprentissage qui utilise une approche évolutionniste, où l'application d'un algorithme génétique adapté génère des requêtes SQL valides répondant aux conditions établies par les exemples d'entrée et de sortie donnés par l'utilisateur.

On a obtenu comme résultat de l'approche un outil dénomméEvoSQL qui a été validé dans cette étude. Sur les28 exercices employés par la recherche [39], 23 exercices ont été obtenus avec des résultats parfaits et 5 exercices sans succès, ce qui représente 82.1 % d'effectivité. Cette effectivité est supérieure de 10.7 % à celle établie par l'outil développé dans [32]SQLSynthesizer et 75% plus haute que l'outil suivant le plus proche Query by Output QBO [31].

La moyenne obtenue dans l'exécution de chaque exercice a été de 3 min et 11sec, ce qui est supérieur au temps établi par SQlSynthesizer, cependant dans la mesure où un

algorithme génétique suppose que l'existence de phases augmente les rangs des temps, le temps obtenu est acceptable par rapport aux applications de ce type.

Dans une conclusion et selon ce qui a été antérieurement exposé nous avons obtenu un outil automatique, avec uneapproche évolutionniste, avec de bons résultats et un processus simple pour l'utilisateur « non expert »**.**

Mots-clés : Programmation Génétique, Requêtes SQL, Génération à partir d'exemples, Génie Logiciel

Abstract

At present the use of the technologies is basic for the advance of the society; these have allowed that persons without knowledge or so called "non expert" users are interested in this use, is for it that the researchers have seen the need to produce studies that allow the adjustment of the systems the existing at the problematic inside the area of the technology.

A need of every user of a system is the management of the information, which can be manage by a database and specific language for these as the SQL (Structured Query Language), which forces the user to come to a specialist for the design and construction of this one, which represents costs and complex methods, but what to do when they are small investigations where the resources and processes are limited?.

aking as a base the research of the university of Washington [32], this report automates the process and applies a different learning technique, for which uses an evolutionary approach, where the application of a genetic adapted algorithm generates query SQL valid that answer to the conditions established by the given examples of entry and exit given by the user.

There was obtained as a result of the approach a tool named EvoSQL that was validated in the same 28 exercises used by the investigation [32], of which 23 exercises were obtained by ideal results and 5 not successful exercises, which represents 82.1 % of efficiency, superior in 10.7 % to the established one for the tool developed in [32] SQLSynthesizer and 75% higher than the following near tool Query by Output QBO [26].

The average obtained in the execution of every exercise was of 3 min and 11seg that is superior to the time established by SQlSynthesizer, Nevertheless, being a genetic algorithm where the steps existence makes that the ranges of times are extended, the obtained one is acceptable with relation to the applications of this type.

In conclusion et according to previously exposed, we have obtained an automatic tool, with an evolutionary approach, with good results and a simple process for the « not expert » user.

Keywords : Genetic programming, SQL query generation from examples, Software engineering

Tabla de Contenido

Lista de tablas

Lista de figuras

Este trabajo de investigación está dedicado a mi familia, por nunca cortarme las alas, por dejarme crecer y por enseñarme a disfrutar lo bueno de cada circunstancia.

Agradecimientos

Este trabajo de investigación no se hubiera logrado culminar con éxito, sin el constante acompañamiento del profesor Ph. D.HouariSahraoui, su apoyo y ayuda durante toda la maestría, hizo posible alcanzar todos los objetivos propuestos.

AlDépartementd'informatique et de RechercheOpérationnelle et la Faculté des Arts et des Sciencesde la Universidad de Montréaly al grupo GEODES por su apoyo, en especial a IslemBaki, su inquebrantable ayuda permitió mi avance.

Al Gobierno de la República de Colombia, en nombre del Ministerio de Tecnologías de la Información y la Comunicación MINTIC, por el programa TALENTO DIGITAL EXTERIOR,que me brindó los recursos económicos necesarios para realizar mi maestría sin contra tiempos, al ICETEX y al grupo de investigación ANTROPACIFICO por su apoyo y compañía.

CAPÍTULO 1

INTRODUCCIÓN

1.1 Contexto

En un mundo globalizado, el acceso a la tecnología ha sido fundamental para el progreso de la sociedad, en palabras de Henry Ford "El verdadero progreso es el que pone la tecnología al alcance de todos" porque la tecnología ha de ser producida, por y para, cualquier persona.

Durante décadas, se ha visto cómo las innovaciones tecnológicas han mejorado la calidad de vida de los individuos, cómo sus expectativas han marcado el rumbo del progreso, cómo los sistemas y lenguajes de información que en un principio, estaban destinados para el desarrollo y uso exclusivo de usuarios con conocimiento especializado, se han visto en la necesidad de ampliar su cobertura e incluir a todo tipo de usuario, tratando de brindar acceso eficaz, veraz y eficiente a la información, el cual sin lugar a dudas se trata de un componente vital en nuestra sociedad digital.

Se puede ver la información como un conjunto de datos organizados que al procesarlos, constituyen un mensaje sobre un ente en particular. Los datos que se integran generan la información que produce conocimiento, pero no solo incrementando el mismo sino que, proporciona la materia prima fundamental para el desarrollo de soluciones, reglas de evaluación y de decisión, que permiten elegir las acciones a realizar, acciones que entre otras aseguran nuestra existencia.

Con los adelantos tecnológicos obtenidos en la sociedad actual y específicamente en las tecnologías de la información, es casi imposible que una empresa, grupo de investigación o un individuo del común, no haga uso de la información para el desarrollo de sus actividades. El uso de las computadoras como herramienta donde confluyen todos los datos, convierte a los sistemas de información en indispensables, sin discriminar el tipo de usuario ya que, estos sistemas son capaces de ofrecer información de forma rápida, ordenada y concreta, lo que facilita la toma de decisiones.

En vista de la importancia y el interés suscitado hacia la gestión de la información, las ciencias aplicadas han buscado responder proporcionando estructuras lógicas para el almacenamiento de datos; sin embargo, dichas estructuras requieren tanto en su diseño y construcción, usuarios con conocimientos especializados o usuarios llamados "expertos". Estos usuarios "expertos" son los encargados de analizar, diseñar e implementar herramientas software que pueden ser hechas a la medida de una empresa, una persona en particular o ser desarrolladas para un área general, pero por más que el usuario experto se esmere por crear aplicaciones que cubran y faciliten tareas cotidianas, los campos aplicables son infinitos, lo cual hace imposible el cubrimiento a cada necesidad. Es allí, donde las nuevas tecnologías han permitido que personas sin conocimiento informático o usuarios llamados "no expertos", se interesen en su uso debido a sus necesidades específicas, por esta razón la comunidad investigadora se ha interesado por las iniciativas que permiten hacer accesibles estas herramientas informáticas y encontraron en el uso de los ejemplos, una buena práctica donde era posible obtener buenos resultados.

Cada vez es más común utilizar los ejemplos para extraer el conocimiento necesario para la automatización de ciertos procesos, de los cuales no se dispone de conocimientos suficientes. Aquí, la idea es observar las ejecuciones de un proceso o simplemente sus entradas y salidas para aprender los mecanismos genéricos que permitan producir estas salidas a partir de las entradas. Es así como numerosos trabajos permiten la creación de modelos, reglas y estrategias a partir de los ejemplos.

Existen muchos estudios en diversas áreas donde han utilizado ejemplos como método de aprendizaje, investigaciones como por ejemplo [41], donde sus autores recuperan información multimedia mediante el análisis de contenido y el aprendizaje desde los ejemplos; [30] donde los autores utilizan los ejemplos para la derivación de reglas y meta-reglas para la transformación de modelos, [31] y [39] donde los autores parten de los ejemplos dados por el usuario de una base datos para inferir sentencias SQL.

1.2 Problemaespecifico

Tanto usuarios "expertos" como "no expertos" requieren de sistemas que faciliten su investigación por medio de la gestión de datos. Esta gestión se realiza por medio de las bases de datos, estructuras lógicas ordenadas donde se guarda, elimina, modifica y consulta los registros de datos; sin embargo y a pesar de ser bajo la necesidad de los conocimientos y las experiencias del usuario no experto (generalmente), es el usuario experto es quien en ultimas implanta los procesos de definición y manipulación de los datos.

El requisito indispensable de contar con un especialista, dificulta en muchos casos el acceso a la información ya que, los sistemas desarrollados en múltiples casos no cubren a priori todas las necesidades existentes, básicamente porque estas pueden ser infinitas. Además en varias situaciones, consideraciones como el volumen de las transacciones o los medios, entre otras, no permiten disponer de un experto en bases de datos.

Para hacer frente a esta situación, se han desarrollado herramientas para asistir a los usuarios en la construcción de sentencias de definición y de consulta en las bases de datospor ejemplo [33] y [10]. Sin embargo, para utilizar las herramientas, los usuarios necesitan de un conocimiento anterior del lenguaje de consulta, así como del esquema de la base de datos adyacente, es decir, nombre de campos, llaves primarias y foráneas etc. La eficacia de estas herramientas es limitada, ya que los conocimientos requeridos no hacen parte de la sapiencia del usuario promedio.

Por otro lado, otra familia de herramientas utiliza los ejemplos de datos de entrada y salida para inferir sentencias SQL [31]. Un trabajo particularmente interesante es el de SQLSynthesizer[39] que, a partir de conocimientos básicos del usuario(es decir qué datos él tiene y qué datos él quiere), permite encontrar de manera semiautomática un conjunto de sentencias que corresponden a los ejemplos de datos suministrados. Si bien este trabajo constituye un avance innegable en la resolución del problema que compete, presenta numerosas limitaciones. Otro es el aspecto semiautomático de la aproximación, éste está limitado a la vez por el tamaño de los datos y la complejidad de las sentencias a derivar. Además, aunque ha sido aplicada sobre ejemplos simples, la aproximación en varios casosno tuvo éxito encontrando la solución buscada.

1.3 Contribución

El objeto de esta memoria, es la aplicación de una aproximación evolucionista para la generación automática de sentencias SQL a partir de ejemplos de entrada y salida.

Partiendo del trabajo de investigación descrito en [39], se propone una aproximación completamente automática basado en la programación genética, una técnica genérica de optimización que se ha adaptado al aprendizaje de las sentencias SQL. Se propone igualmente, una herramienta software EvoSQL para implantar este enfoque.

Con la automatización, se buscaaumentar el alcancede la herramienta a datos de gran talla y a sentencias complejas, para esta última, la intervención humana de los usuarios no expertos implica una carga cognitiva consecuente. Además, esta automatización permite igualmente facilitar tareas conexas tales como: la generación de código, la validación y la verificación.

Con la programación genética, se busca modelar el aprendizaje de sentencias SQL como un problema de optimización que consiste en explorar el gran espacio de sentencias posibles,

siendo guiado únicamente por la adecuación de sentencias candidatas de los datos suministrados en el ejemplo. Bajo reserva de una buena gestión del espacio de investigación, es posible derivar sentencias complejas a partir de grandes ejemplos sin conocimiento específico como es el caso en [32].

Finalmente, con el objeto de comparar, se validará esta aproximación y la herramienta para los ejemplos utilizados en [39]

1.4 Estructura de la memoria

Esta memoria está estructurada de la siguiente manera:

El capítulo 2 presenta el estado del arte, se introduce con las nociones base del lenguaje SQL y de la programación genética. Enseguida se presentan los trabajos conexos sobre la derivación de sentencias en general y las sentencias SQL en particular. Finalmente, se presentan algunos trabajos sobre la utilización de algoritmos evolucionistas para el aprendizaje de estructuras complejas en ingeniería del software.

En el capítulo 3, se presenta la aproximación propuesta de esta memoria, en donde se discute en particular, las diferentes adaptaciones de la programación genética al programa del aprendizaje de sentencias SQL.

El capítulo 4, presenta la evaluación de la aproximación desarrollada, utilizando los casos propuestos en [39]. Enseñando particularmente, la concepción experimental y la discusión de los resultados obtenidos.

Con el capítulo 5, concluye esta memoria en el cual se presenta un resumen de la aproximación propuesta, los resultados obtenidos así como la discusión, los límites y los mejoramientos a futuro.

CAPÍTULO 2

ESTADO DEL ARTE

Este capítulo tiene por objeto, establecer nociones base, aclarar los lenguajes utilizados, e identificar algunas herramientas software desarrolladas en la generación de sentencias SQL a partir de ejemplos; adicionalmente, dar a conocer algunas de las investigaciones realizadas en los últimos años, la introducción y aplicación en la ingeniería del software.

2.1 Nociones base

2.1.1 SQL

A principios de los años 70's, elcientífico inglés **Edgar Frank Codd**, propuso un modelo relacional y asociado a este, un lenguaje de acceso a los datos, este lenguaje originalmente denominado Sequel, se realizó como parte del proyecto de investigación System R [9], el cual constaba de una base de datos en la que se implementó por primera vez el SQL.

La versión original se desarrolló en el laboratorio de investigación en California (San JoséResearch Center) de IBM,no obstante es Oracle quien lo introduce comercialmente en 1979; años después ANSI (American NationalStandardsInstitute) e ISO (International OrganizationforStandardization)[24]adoptan oficialmente el SQL como lenguaje estándarde base de datos relacionales; desde entonces ha evolucionado a lo que ahora se conoce como SQL (StructuredQueryLanguage, o lenguaje estructurado de consultas).

El lenguaje SQL es un lenguaje muy amplio comprende muchas estructuras y particularidades por lo cual y para comprensión de este trabajo de investigación solo se expondrá aquí un subconjunto de este.

SQL se define como un lenguaje basado en un conjunto de condiciones, proposiciones, restricciones y/o transformaciones con las cuales se construye un enunciado de estructura válida para ser procesada en un motor de base de datos[32].Dentro de una base de datos se puede gestionar diferentes tablas (estructura donde se almacenan los datos) y datos, para lo cual SQL dispone dos tipos de lenguajes: uno para la de definición de datos DDL y el otrode manipulación de datos DML.

2.1.1.1 Lenguaje DDL

El lenguaje de definición de datos (DDL Data DefinitionLanguage) permite crear instrucciones quese utilizan para definir la estructura, las restricciones de integridad, creación, modificación y eliminación de objetos de base de datos [32].

Palabra clave	Descripción
CREATE	Utilizado para crear nuevas tablas, procedimientos almacenados e índices
DROP	Empleado para eliminar tablas, procedimientos almacenados e índices
ALTER	Utilizado para modificar las tablas agregando campos o cambiando la definición de los campos

Tabla 2.2.1 Instrucciones DDL

2.1.1.2 Lenguaje DML

El lenguaje de manipulación de datos (DML Data ManipulatiónLanguage) especifica cómoes un dato ingresado, actualizado, eliminado y recuperado de los objetos o registros definidos en la base de datos [32].

Palabra clave	Descripción
SELECT	Utilizado para consultar registros de la base de datos
INSERT	Utilizado para cargar lotes de datos en la base de datos
DELETE	Utilizado para eliminar los valores de los campos y registros especificados.
UPDATE	Utilizado para modificar registros de una tabla de una base de datos.

Tabla 2.2Instrucciones DML

El SQL permite construir una sentencia estructurada a partir de la cual, se puede realizar una consulta de datos. Por interés de esta investigación se explicará el funcionamiento y estructura de la sentencia de consulta SQL es decir, la cláusula SELECT.

2.1.1.3 Clausula SELECT

La estructura simple de una sentencia SQL "SELECT" está dada de la siguiente manera:

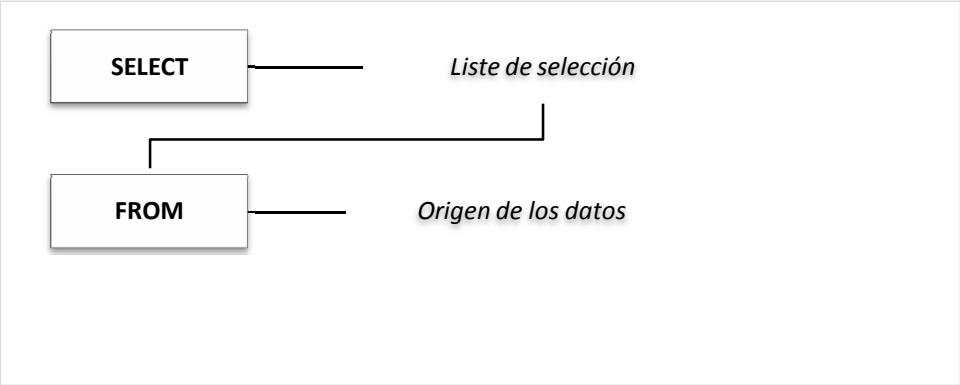

Fig.2.1 Estructura simple de una sentencia SQL

Esta estructura simple, permite que sea tomada la información de una columna (o columnas) en una tabla sin ningún tipo de restricción, lo que necesariamente generará redundancias en los resultados, para minimizar esto, se puede utilizar otras palabras claves (ver tabla 2.2.3), condiciones de unión y funciones agregadas (ver tabla 2.2.4) que permiten

delimitar la búsqueda de la sentencia, creando restricciones que deben ser cumplidas haciendo de ésta una consulta más específica.

Palabra clave	Descripción
WHERE	Especifica una condición que debe cumplirse para que los datos sean devueltos por la consulta
GROUP BY	Especifica la agrupación dada a los datos
HAVING	Especifica una condición que debe cumplirse, similar al *Where* y se utiliza junto al *GroupBy*
ORDER BY	Utilizado para el orden en la presentación final de los resultados, puede expresarse ASC (orden ascendente) o DESC (orden descendente).

Tabla 2.3Instrucciones para delimitaruna consulta de datos.

Y algunas funciones agregadas como:

Palabra clave	Descripción
AVG (exp)	Calcula la media aritmética de un conjunto de valores contenido en un campo específico de la consulta
MIN (exp)	Devuelve el valor mínimo de un conjunto de valores contenido en un campo en específico de la consulta
MAX (exp)	Devuelve el valor máximo de un conjunto de valores contenido en un campo en específico de la consulta
COUNT (exp)	Devuelve el número de registros por una consulta
SUM (exp)	Devuelve la suma del conjunto de valores contenidos en un campo especifico de la consulta
DISTINCT	Devuelve las filas distintas a determinada consulta, puede combinarse con otras palabras claves.

Tabla 2.4Funcionesagregadas

De tal forma que, una sentencia SQL de consulta más completa resultaría con la siguiente estructura (ver fig. 3):

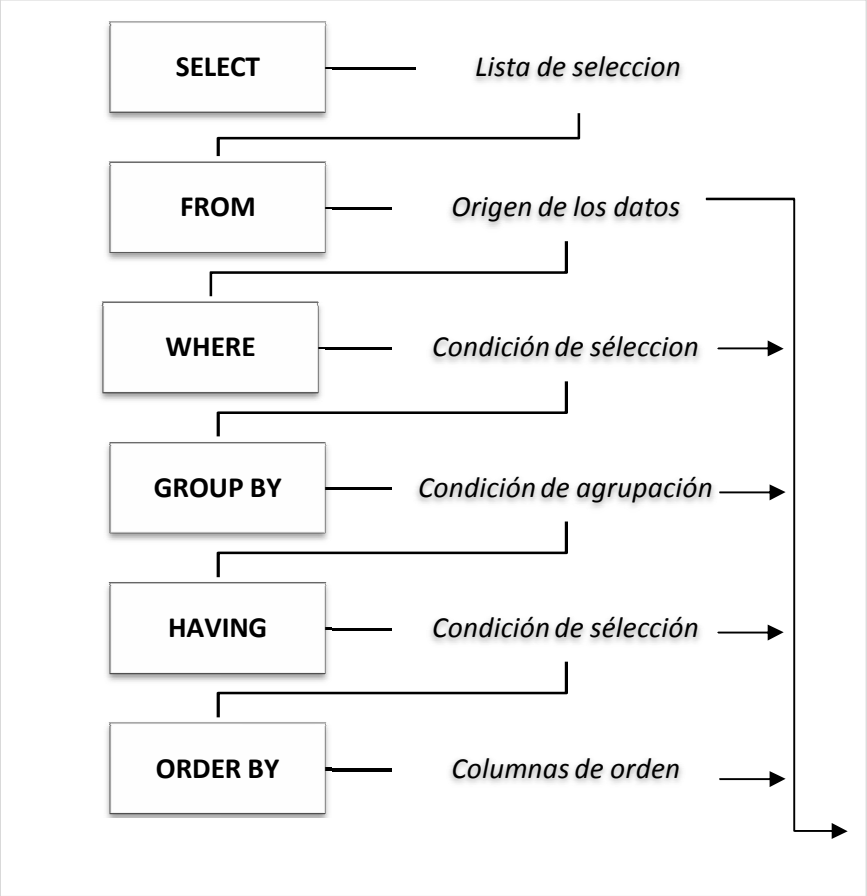

Tabla 2.5 Sentencia SQL agregada.

2.1.1.4 Tipos de datos y operadores

SQL admite una variada gama de tipos de datos para el tratamiento de la información contenida en las tablas, los tipos de datos pueden ser numéricos (con o sin decimales), alfanuméricos, de fecha o booleanos(si o no).Según el gestor de base de datos que se esté utilizando, los tipos de datos pueden variar (ver tabla 2.5).

TIPOS DE DATOS				
Numéricos	**Alpha-Numéricos**	**Fecha**	**Lógicos**	**BLOB**
Integer Numeric Decimal Float	Char(n) Varchar(n, m)	Date DateTime	Bit	Image Text

Tabla 2.6Tipos de datos

Les operadores en SQL se pueden definir como, combinaciones de caracteres que se utilizan para realizar asignaciones como comparaciones de datos.

OPERADORES EN SQL					
Aritméticos	+ - * / ** ^ %	Suma Resta Multiplicación División Elevación Modulo	**Lógicos**	AND OR NOT	Disyunción Union Negación
Relacionales	< <= > >= = <> != !< !>	Menor a Menor o igual a Mayor a Mayor o igual a Igual a Diferente de No Menor a No Superior a	**Encadenamiento**	+	Encadenamiento alpha-numéricos

Tabla 2.7Operadores en SQL

2.1.2 Programacióngenética

El término algoritmo genético, fue por primera vez usado por John Holland [23], en el libro "La Adaptación en los Sistemas Naturales y Artificial" de 1975, el cual contribuyó en la creación de un campo de investigación creciente; varios científicos de distintos orígenes participaron en el desarrollo de ideas similares como Fogel y otros en [20], quienes plantearon la idea *EvolutionaryPrograming* donde modelaba el proceso de evolución como un proceso de optimización, se abstuvieron de modelar el producto final de la evolución y optaron por modelar el proceso de la evolución misma como vehículo para la producción de un comportamiento inteligente.

En sus orígenes, los algoritmos genéticos consistieron en copiar procesos que tenían lugar en la selección natural introducida por Darwin [11]. Este estableció cómo en la naturaleza los individuos de una población competían con otros por recursos para sobrevivir (los individuos que tenían más éxito en la lucha por los recursos, tenían mayores probabilidades de resistir los cambios) y cómo la combinación de las buenas características de diferentes padres, podían originar que su descendencia estuviera incluso mejor adaptados al medio que los mismos padres. Se estableció de esta manera que, las especies evolucionaban adaptándose más y más al entorno a medida que transcurrían las generaciones.

Todo el proceso de evolución opera sobre los cromosomas, y la selección natural es el mecanismo que relaciona los cromosomas con la eficiencia, otorgando a los individuos más adaptados al entorno, un mayor número de oportunidades de reproducirse. En esta etapa de reproducción tienen lugar los procesos evolutivos, siendo los más comunes, el cruce o recombinación, que combina los cromosomas de los padres para reproducir descendenciay por otra parte, las mutaciones causantes de que los cromosomas de la descendencia sean diferentes a los de los padres.

Se puede establecer que la evolución se produce en la naturaleza gracias a:

- Existencia de reproducción entre individuos de una población
- Las características de los individuos afectan su probabilidad de supervivencia
- Existencia de la herencia
- Existencia de recursos finitos que ocasionaban competencia

En la programación genética, se buscan poblaciones de programas que evolucionen transmitiendo su herencia de manera que se adapte mejor al medio. La medida de la calidad de adaptación del individuo, depende del tipo de entorno o problema.

Al igual que se establece en naturaleza, los operadores utilizados por los algoritmos genéticosa fin de hacer evolucionar la población de manera progresiva son:

- La selección: Donde se determina cuáles individuos tienen más probabilidades de obtener mejores resultados a partir de la evolución de la función.

- El cruce: Donde los individuos intercambian parte de su información.

- La mutación: Donde se toma parte de la cadena de información del individuo y se muta o cambia.

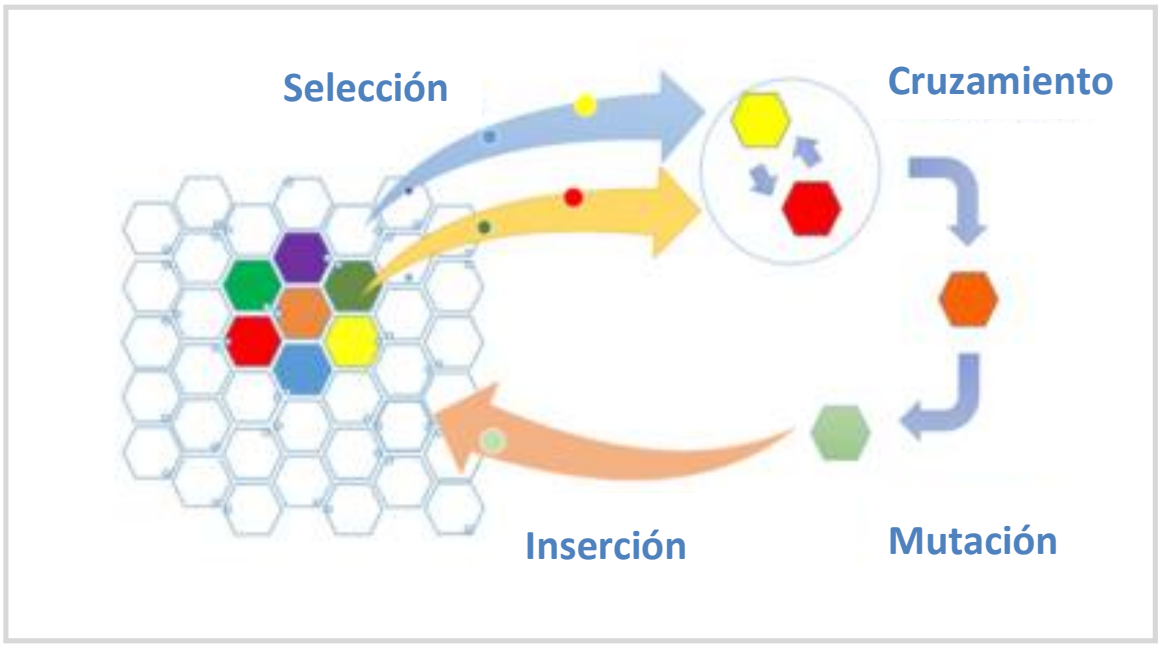

Fig2.2Ciclo de la programación genética

2.1.2.1 Algoritmo genético simple

```
BEGIN /* Algoritmo genético simple */
      1.    Generar una población inicial
      2.    Evaluar la función objetivopara cada individuo
      WHILE NOT Terminado Do
      BEGIN/* Producir una nueva generación  */
      FOR  (tamaño de la población) / 2 DO
      BEGIN/* Ciclo reproductivo  */
         3. Seleccionar dos individuos de la anterior generación para el
            cruce denominados padres(la probabilidad de selección debe ser
            proporcional a la función de evaluación del individuo)
         4. Cruzar dos individuos con cierta probabilidad obteniendo así
            dos descendientes o hijos
         5. Mutar dos descendientes con cierta probabilidad obteniendo así
            dos nuevos hijos
         6. Evaluar la función en los dos nuevos descendientes mutados
         7. Insertar nuevos descendientes mutados quienes serán padres en
            la nueva generación
      END
      IF (la población ha convergido) THEN
      Terminado == TRUE
      END
```

```
    END
END
```

2.1.2.2 Generación de la población inicial

En la generación de la población inicial, un parámetro principal para el algoritmo genético es el tamaño, ya que las poblaciones pequeñas corren el riesgo de no cubrir adecuadamente el espacio de búsqueda, y trabajar con poblaciones de gran tamaño puede acarrear problemas con el costo computacional. Habitualmente, la población inicial se genera aleatoriamente, garantizando que esta población tenga diversidad estructural de las soluciones ya que, su composición puede afectar dramáticamente el comportamiento del algoritmo.

2.1.2.3 Evaluación de la Función Objetivo

La evaluación de la función objetivo,consiste en asignar un valor numérico proporcional a la utilidad o habilidad del individuo evaluado, de tal manera que, entre mejor habilidad tenga el individuo, su función objetivo tendrá un mejor valor que una donde el individuo tenga una menor habilidad; de esta forma, es sancionada su utilidad.

Una buena función objetivo o "fitnessfunction" debe reflejar de cierta manera el valor "real" de la habilidad de un individuo referente a un problema en particular, de esta forma se establecen tipos de fitness asociados a diferentes tipos de problemas, según John Koza [27] 1992

- **Fitness puro**: Donde la medida de ajuste es establecida en la terminología natural del propio problema.

- **Fitness estandarizado:** Donde los valores posibles son mayores e iguales a cero, siendo el mejor el cero.

- **Fitness ajustado:** Donde el mayor valor posible es el 1 y los valores entre el intervalo (0, 1] y se calcula como $= \frac{1}{1+fitnessStandar}$

- **Fitness normalizado:** Donde los valores se encuentran entre [0, 1] pero a diferencia del ajustado, cuanto más próximo esté a 1 no sólo indica que es una buena solución sino que, es destacadamente mejor que las otras.

2.1.2.4 Operadores genéticos

Un operador genético, es una función empleada en los algoritmos genéticos para mantener la diversidad genética de una población.

La variación genética es necesaria para el proceso de evolución. Los operadores genéticos utilizados en los algoritmos genéticos, son análogos a aquellos que ocurren en el mundo natural: la selección equivalente a la supervivencia del más apto en el mundo natural, es el cruzamiento, el cual equivale a la reproducción sexual y la mutación equivale a la mutación biológica.

Para el paso de una generación a la siguiente, se aplican los siguientes operadores genéticos:

- Operadores de Selección
- Operadores de cruzamiento

- Operadores de Mutación

2.1.2.4.1 Operadores de Selección

Los algoritmos de selección de individuos deciden qué individuos van a reproducirse, otorgando más oportunidad de reproducción al individuo con mayor habilidad o más apto, así pues, la selección del individuo estará relacionada con el valor de la función objetivo (Función fitness), sin eliminar por completo las opciones de reproducción menos aptas para evitar homogeneidad.

Se presentan tres cuestiones básicas relacionadas con la fase de selección:

- Espacio de muestreo.
- Mecanismo de muestreo
- Probabilidad de selección.

Cada una de ellas ejerce una influencia significativa en la selección y porconsiguiente, en el comportamiento del algoritmo genético.

2.1.2.4.1.1 Espacio de Muestreo

Para crear una nueva población en una nueva generación, esta se puede realizar incluyendo en el conjunto de todos los padres a los hijos, o solo una parte de ellos, por lo cual el tamaño de muestreo se va a caracterizar por factores como el tamaño y los integrantes (ya sean padres o hijos), de esta forma se establece el espacio de muestreo regular y extendido.

2.1.2.4.1.2 Espacio de Muestreo Regular

Holland [23] estableció en el algoritmo genético original que, los padres serian reemplazados por sus hijos tan pronto como estos nacen, por ser todo al azar los hijos generados podrían ser "peores" que sus padres (menos aptos o fitness bajo) entonces algunos individuos con fitness alto se podrían perder en el proceso de evolución; para solucionar esto se propusieron estrategias de reemplazo. Holland establece como estrategia de reemplazo que cada vez que nace un hijo, este reemplace a un padre de la población elegido de forma aleatoria. Por otra parte Jong [12] propone que, cuando nace un hijo se seleccione un padre para morir y es el más parecido al nuevo hijo, estrategia que se denominó como crowding. Este espacio de muestreo se considera usualmente probabilístico.

2.1.2.4.1.3 Espacio de muestreo extendido

Dentro del espacio de muestreo extendido tanto los padres como los hijos, tienen las mismas oportunidades para competir por la supervivencia. Bäck y otros [4], utilizaron esta estrategia en su algoritmo genético, donde los padres e hijos compiten por la supervivencia y se seleccionan los mejores entre los hijos y los padres como padres de la próxima generación. Este método termina siendo completamente determinístico y una ventaja que presenta es que puede mejorar el performance del algoritmo genético.

2.1.2.4.1.2 Mecanismo de muestreo

Para seleccionar los individuos del espacio de muestreo se puede realizar a través de:

- Selección Estocástica o Probabilística
- Selección Determinística
- Selección Mixta

2.1.2.4.2 Selección Estocástica o Probabilística

Como el nombre lo indica, utiliza el azar para realizar la elección, el método más conocido en esta clasificación es la selección proporcional de Holland o Selección por Ruleta, *roulotte wheel selección.*

La idea de la *selección por ruleta* es determinar la probabilidad de selección o también denominada probabilidad de supervivencia, establece que a cada individuo de la población le sea asignado un valor proporcional a la fitness o función objetivo, de tal forma que, la suma de todos los porcentajes sea la unidad. Para seleccionar, se genera un número aleatorio dentro del intervalo de 0 a 1 y devuelve al individuo en esa posición de la ruleta.

La probabilidad de selección se calcula de la siguiente manera:

$$P(x) = \frac{f(x)}{\sum_{j=1}^{n} f(j)}$$

Dónde:

x: Individuo

n : Tamaño de la población

$P(x)$: Probabilidad del individuo x

$f(x)$: Funciónobjetivo o fitness function en x

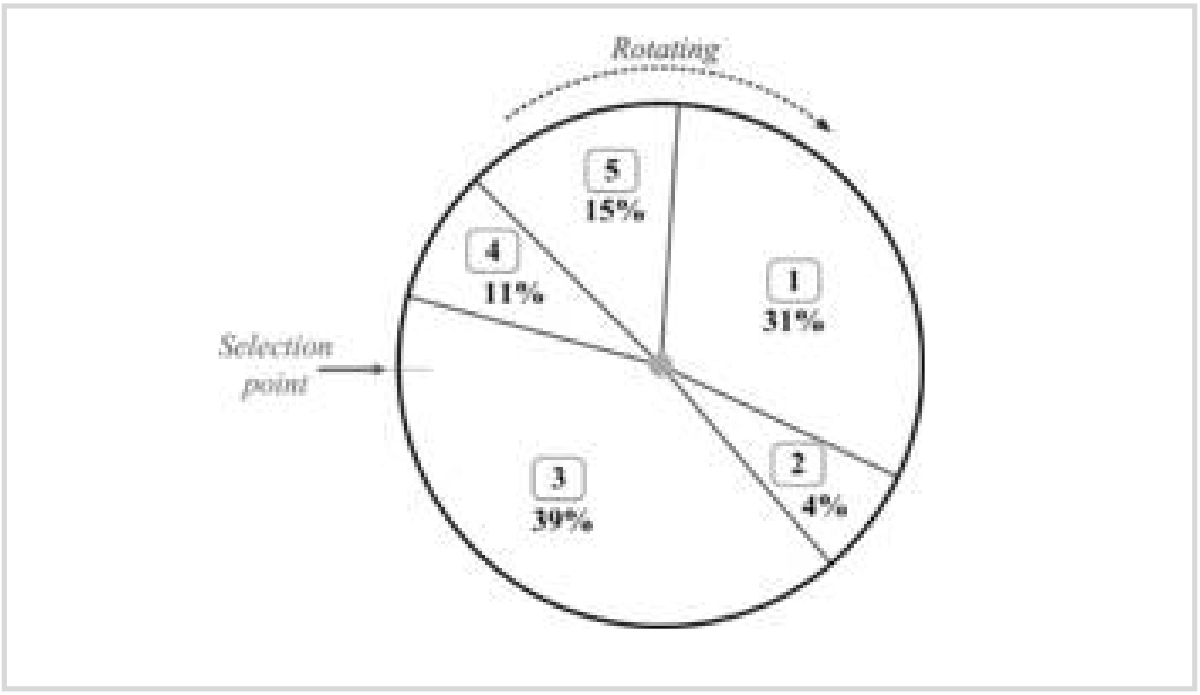

Fig2.3La Ruleta. Se muestran 5 particiones y los porcentajes correspondientes a cada una, dando mayor probabilidad de selección a los valores en la partición 3 y más pequeña a la partición 2.[42]

2.1.2.4.3 Selección Determinística

Como el nombre lo indica, se determina a una característica un valor a priori para realizar la selección, generalmente selecciona a los mejores individuos del espacio de muestreo, este método prohíbe que los individuos duplicados entren en la población.

Como primer paso, se ordenan los individuos utilizando el valor de la fitness, de esta manera selecciona los mejores individuos definiendo un umbral *T*. Se selecciona el *T%* conlos

mejores individuos que serán cruzados con otros (dependiendo del tipo de selección en particular que se desee aplicar), por ejemplo:

- En la selección denominada *Vasconcelos* [1], se cruzan los mejores individuos con los individuos menos aptos.

- En la selección *Elitista,* se asegura que el mejor individuo se mantenga a través de las generaciones (bajo ciertas condiciones muy generales), se garantiza la convergencia teórica al óptimo global y en la práctica, mejora la velocidad de convergencia de los algoritmos genéticos.

- En el *Reemplazo Generacional* [36], se reemplaza a los individuos menos aptos con hijos. Los individuos se seleccionan en función de su probabilidad de supervivencia, los individuos menos aptos que la media, tienen más posibilidades de ser seleccionados para desaparecer.

2.1.2.4.4 Selección Mixta

Esta selección combina las características de las selecciones anteriores, posee tanto particularidades estocásticas como determinísticas. Un ejemplo de este tipo, es la selección por torneo presentada por Goldbert [21], donde selecciona un conjunto de individuos de forma aleatoria y se escoge el mejor del conjunto de reproducción. El número de individuos común del torneo es 2.

La selección de torneo estocástica fue sugerido por Wetzel [35], donde la probabilidad de selección se calcula de la forma usual y se muestrean pares consecutivos de individuos usando

la selección por ruleta, luego de sortear un par, los individuos con alto fitness se ingresan en la nueva población.

2.1.2.5 Probabilidad de Selección

La probabilidad de selección de cada individuo posee algunas propiedades no deseables por ejemplo:

- En generaciones tempranas hay tendencia a que super individuos dominen el proceso de selección.
- En generaciones finales (cuando la población ha convergido), la competencia entre individuos es menos fuerte y se producirá un comportamiento aleatorio de búsqueda, donde la selección no concede mayor ventaja a uno u otro individuo.
- En una población infinita, el proceso siempre es imperfecto y puede favorecer más de lo correspondiente a individuos ocasionalmente más aptos, debido a que la población es finita y puede acabar expulsando de la población a los restantes haciendo que, el algoritmo genético converja prematuramente hacia tal individuo.

Para mitigar las anteriores propiedades no deseables, existen propuestas de método de *Escalamiento* o *Ranking,* donde se transforman los valores de la función objetivo y la probabilidad de supervivencia de cada individuo para:

- Mantener una diferencia razonable entre los niveles de fitness relativos de los individuos.
- Prevenir un monopolio muy rápido de algunos super individuos para reunir los requerimientos de limitar la competencia temprana y estimular la tardía.

Bäck y otros en [5] introduce la noción de selección por ranking, donde ordena la población de mejor a peor y asigna la probabilidad de selección de cada individuo, acorde al ranking pero ya no al fitness.

2.1.2.4.2 Operadores de cruzamiento

El cruzamiento o crossover, es el principal operador genético empleado para la recombinación o reproducción para la descendencia, que será insertada en la siguiente generación.

Este proceso trabaja sobre dos individuos a la vez, y genera hijos al recombinar sus características; para lo cual, se toma dos individuos correctamente adaptados al medio y se obtiene una descendencia que comparte genes de ambos, existiendo la posibilidad que los genes heredados sean precisamente los causantes de lo "bueno" en los padres.

Una forma simple de realizar el crossover consiste en elegir de forma aleatoria, un punto de cruce y generar el hijo, combinando el segmento de un padre a la izquierda del punto de cruce y generar el hijo combinándolo a la derecha del otro padre.

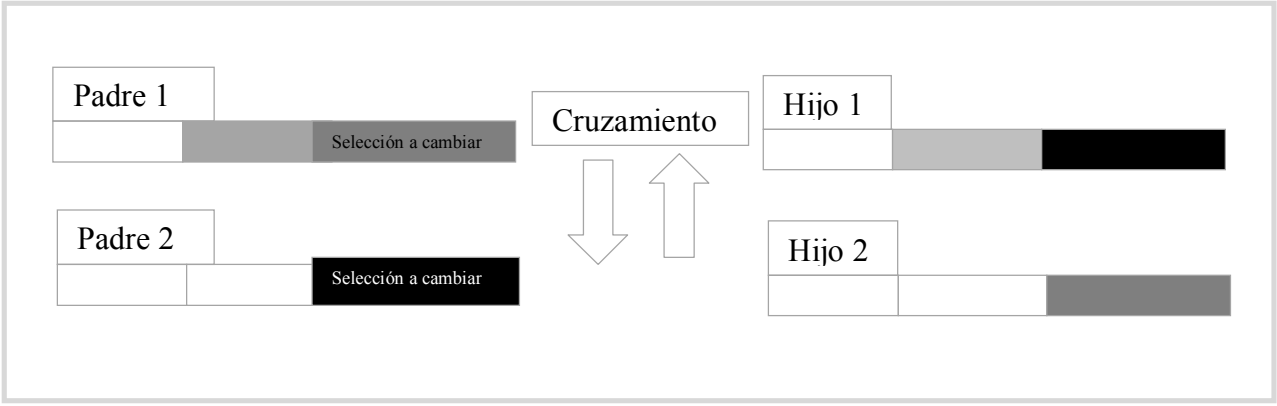

Fig2.4Cruzamiento o Crossover

La probabilidad de *Cruzamiento*, se define como la relación entre el número de hijos producidos en cada generación y el tamaño de la población. Esta probabilidad controla el número de individuos que se someterán a la operación del crossover; una alta probabilidad de crossover permite una mayor exploración del espacio de soluciones, reduciendo las oportunidades de establecer en un óptimo falso, por el contrario una probabilidad muy alta, provoca gran desperdicio de tiempo en la exploración de regiones no prometedoras del espacio de soluciones. Igualmente, existen cruzamientos que utilizan múltiples puntos de cruce, todo depende del algoritmo a implementar ya que, cuando se escoge un punto de cruce al azar es posible que algunas de las sub-cadenas sean iguales, debido a esto, al hacer el cruzamiento no se generarían nuevos individuos, lo que se traduciría en pérdida de eficiencia ocasionados por la reevaluación de espacios ya explorados.

2.1.2.4.3 Operadores de mutación

La *Mutación,* es un operador que produce cambios espontáneos en la población, lo que introduce una variabilidad extra que permite la exploración de espacios nuevos de solución, esta técnica puede insertar nuevos genes aún inexistentes en las generaciones precedentes a fin de aumentar la diversidad y evita la convergencia prematura del algoritmo.

Una forma simple de realizar la mutación es alterando uno o más genes del individuo aleatoriamente (ver fig. 2.5).

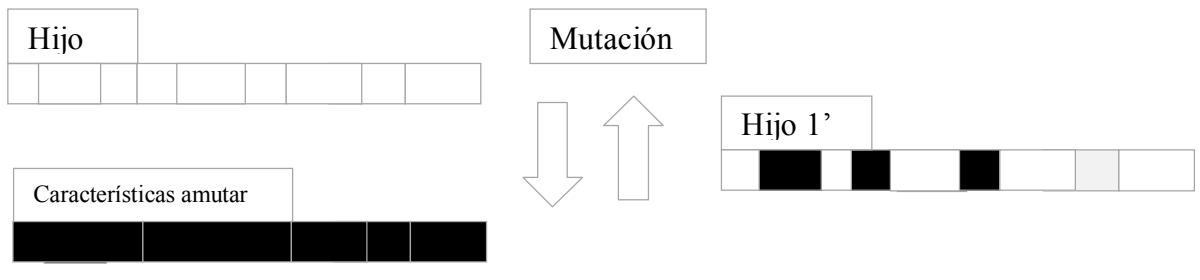

Fig2.5Mutación.

Esta operación es controlada por el porcentaje o tasa de mutación, definida como el número total de individuos de la población que deben ser mutados, si la tasa es muy baja, muchos individuos que podrían haber sido producidos nunca se prueban y si es muy alta, habrá mucha perturbación aleatoria; es decir, los hijos comenzarán a perder su parecido a los padres y el algoritmo perderá la habilidad de aprender del historial de búsqueda.

Mientras que la mutación en un algoritmo genético sirve como un operador parareintroducir características perdidas en la población, las cuales no podrían ser recuperadasnuevamente por medio de la recombinación, el crossovercombina segmentossignificativos de diferentes padres para producir un nuevo individuo, que sebeneficiará con combinaciones de características ventajosas de ambos padres. Las mutaciones pueden darse por un reemplazo aleatorio de un "parte" o múltiples "partes".

2.1.2.5 Criterios de finalización

La finalización de una ejecución del algoritmo genético, puede ser establecida por la designación de un resultado enel algoritmo genético, esa designaciónnormalmente es la del mejor individuo que aparece en alguna generación de la población como el resultado de una corrida del algoritmo; sin embargo, se encuentra frecuentemente un número máximo de generaciones o se presenta cuando el programa ha convergido a algún valor.

Las condiciones de finalizaciónmás comunes son:

- Encuentro de una solución que satisfaga los criterios
- Número fijo de generaciones que alcanzó
- Presupuesto asignado alcanzado (memoria, tiempo)
- Idoneidad de la solución de más alto rango está alcanzando o ha alcanzado un nivel tal que, las sucesivas iteraciones ya no producen mejores resultados

2.2 Trabajosconexos

El uso de los ejemplos para generar conocimiento ha sido utilizado en diferentes investigaciones, entre la literatura existente, encontramos metodologías como el aprendizaje por el ejemplo, el cual se describe a continuación.

2.2.2 Aprendizajepor el ejemplo

El aprendizajemediante el uso de ejemplosha sido una técnica utilizada durante muchos años, este medio permite inferir una serie de operaciones que posibilita el aprendizaje en un sistema determinado.

Esta técnica consiste en recibir ejemplos del sistema y a partir de estos, definir comportamientos, generar estructuras, procedimientos, órdenes etc. que al ser evaluadas en el mismo sistema, se convierten en pruebas de su funcionamiento, permitiendo en casos ser generalizadas para adaptarse enescenarios arbitrarios, Un ejemplo de esta técnicas es la *Consultapor el Ejemplo*.

2.2.3 Consulta por el ejemplo

A principios de los años setenta, se desarrolló en el centro T.J Watson de IBM por ZloofMoshé[40], esta técnica de consulta por el ejemplo o en inglés QuerybyExemple(QBE)es un método de consulta en bases de datos.

QBE está basado en la idea de DRC (Domainrelationalcalculus), que es un cálculo que fue introducido por Michel Lacroix y Alain Pirotte como un lenguaje de consultas declarativo de base de datos para el modelo relacional.

Aunque QBE fue originalmente textual, las siguientes implementaciones como la de Access, ofrecen una interfaz gráfica para la expresión de las consultas.Este lenguaje consiste que el usuario llene un formulario con los datos que requiere y el sistema responde con datos reales que fueron especificados en el formulario; de esta forma, QBE consigue que el usuario sea parte de la consulta, ya que el origen de la búsqueda se encuentra en la mente de éste. Esto ayuda a minimizar los errores de búsqueda ya que, es el propio usuario quien facilita la información. QBE suministra una manera para que, un usuario de software realice consultas sin necesidad de conocer un lenguaje específico de consulta como SQL.

Posee dos características distintivas:

1. A diferencia de muchos lenguajes de consulta y de programación, QBE presenta una sintaxis bidimensional. Una consulta en el lenguaje unidimensional (como SQL) se puede formular en una línea posiblemente larga. Un lenguaje bidimensional necesita dos dimensiones para la formulación de consultas (Las consultas parecen tablas).

2. Las consultas en QBE se expresan "mediante un ejemplo". En lugar de incluir un procedimiento para obtener la respuesta deseada, se usa un ejemplo de lo deseado. El sistema generaliza este ejemplo para obtener la respuesta a la consulta.

Se puedendistinguirvariostipos de consultas:

- **Consulta por selección básica:** Este tipo de consulta permite extraer datos de la base de datos con el criterio que imponga el usuario. En su versión más simple, una consulta de selección mostraría todos los campos de todas las filas de una tabla. Se puede definir tanto los campos a mostrar como las filas que cumplan una determinada condición.

- **Consulta por resumen:** Este tipo de consultas calculan información resumida como totales y medias de un campo en concreto (el cálculo es por columnas). Generalmente, se aplican a campos numéricos y con fines estadísticos. Además de los totales y las medias, también es posible calcular el recuento de filas, y los valores máximos y mínimos de un campo. En estos casos se puede aplicar también a otros tipos de datos, como campos textuales.

- **Consulta por tabla de referencia cruzada:** Este tipo de consultas permite generar columnas que no existen en una determinada tabla a partir de los datos que aparecen en las filas. Son útiles cuando, por ejemplo, deseamos generar columnas con fechas o intervalos de tiempo (como un año) que contengan totales (como total de gastos). Estas consultas son difíciles de expresar en el lenguaje SQL estándar (que se verá en el siguiente tema), pero Access contiene una instrucción especial para ellas.

- **Consulta por parámetros:**Las consultas de parámetros permiten que el usuario proporcione datos que determinen el comportamiento de estas consultas. Estos datos particularizan las consultas, de forma que se adaptan a las necesidades concretas del usuario.

- **Consulta Por Acción:**Las consultas de acción permiten eliminar, añadir y modificar filas de tablas.

Entre las ventajas de QBE se encuentra, la sencillez y rapidez; por el contrario entre sus desventajas está que solo sirve para consultas simples en la cual haya que encontrar un match en los atributos y presenta limitaciones en los datos que se puedan pasar para encontrarun match.

Debido a que no requiere conocimientos de programación, QBE es adecuado para los usuarios finales, ya que utiliza una interfaz fácil de usar para obtener datos de bases de datos, por el contrario, el lenguaje SQL, tiene una sintaxis compleja que toma años de dominar, comandos SQL como crear, borrar consultar y modificar las estructuras de base de datos y los registros de seguridad del usuario, así como crear informes ad hoc requieren de conocimientos no solo del lenguaje en particular sino de análisis del modelo en particular, características especiales que dificultan su aprendizaje en el usuario final. QBE produce SQL y lo aísla al usuario, simplificando su trabajo; sin embargo, presenta limitaciones por lo cual y a pesar de lo complejo, SQL se convierte en el lenguaje dominante de base de datos, aunque existen muchas herramientas software de reportes inspirados en QBE.

2.2.4 SQL por el ejemplo

La mayoría de literatura existente sobre la generación de sentencias SQL, parte con una base de datos diseñada, establecida y destinada para personal especializado con conocimiento en bases de datos.

Los estudios [14], [2] desarrollan herramientas que cumplen tareas específicas, permiten generar sentencias a partir de la estructura de la base de datos y de información, como llaves primariasy foráneas que son la fuente para crear condiciones de unión, pero necesitan de un usuario con conocimientos específicos.

Existen otros como [9], [34] que utilizan los ejemplos como base de partida para su generación, pero son muy limitadas por que realizan únicamente el proceso en una sola tabla, lo cual no es realmente funcional ya que, en general las pequeñas estructuras de bases de datos utilizan más de 2 tablas.

En [31] se desarrolló *Queryby Output QBO,* una herramienta para inferir consultas SQL (es una de las más recientes técnicas), requiere un par de entrada-salida y usa el árbol de decisión [26] para inferir sentencias, pero presenta desventajas ya que, solo puede unir dos tablas y requiere que los usuarios especifiquen las columnas de salida, puede inferir solo consultas simples por que no soporta muchas de las características del SQL como GROUP BY et HAVING.

Sin embargo, la investigación realizada por un grupo de investigación de la universidad de Washington [39] amplía significativamente el subconjunto de SQL soportado por las anteriores herramientas. Ellos parten con la premisa de generar sentencias SQL a partir de ejemplos de entrada y salida, para lo cual toman ejemplos del sistema y generan tablas virtuales de datos(tablas de entrada ver(fig. 2.6) y salida ver (fig2.7)), estas tablas luego son condensadas en una base de datos y a partir del conocimiento de reglas dentro del lenguaje SQL y reglas propias, establecen la estructura válida para una sentencia SQL.

El subconjunto de SQL cubierto en la investigación [39], toma las 10 principales características de SQL más utilizadas (ver fig. 2.6); sin embargo, Zhang y Sun dentro de su investigación aclaran que, la cláusula IN no es tomada dentro de este grupo, dándole entrada en el grupo seleccionado a la cláusula HAVING, por ser más comúnmente usada junto a la cláusula GROUP BY y descartan cláusulas como LEFT JOIN y RIGHT JOIN, por considerar que son menos utilizadas por los usuarios finales no expertos, y otras están diseñadas para hacer la consulta SQL más fácil de escribir y por lo tanto, su eliminación no afecta la expresividad del lenguaje.

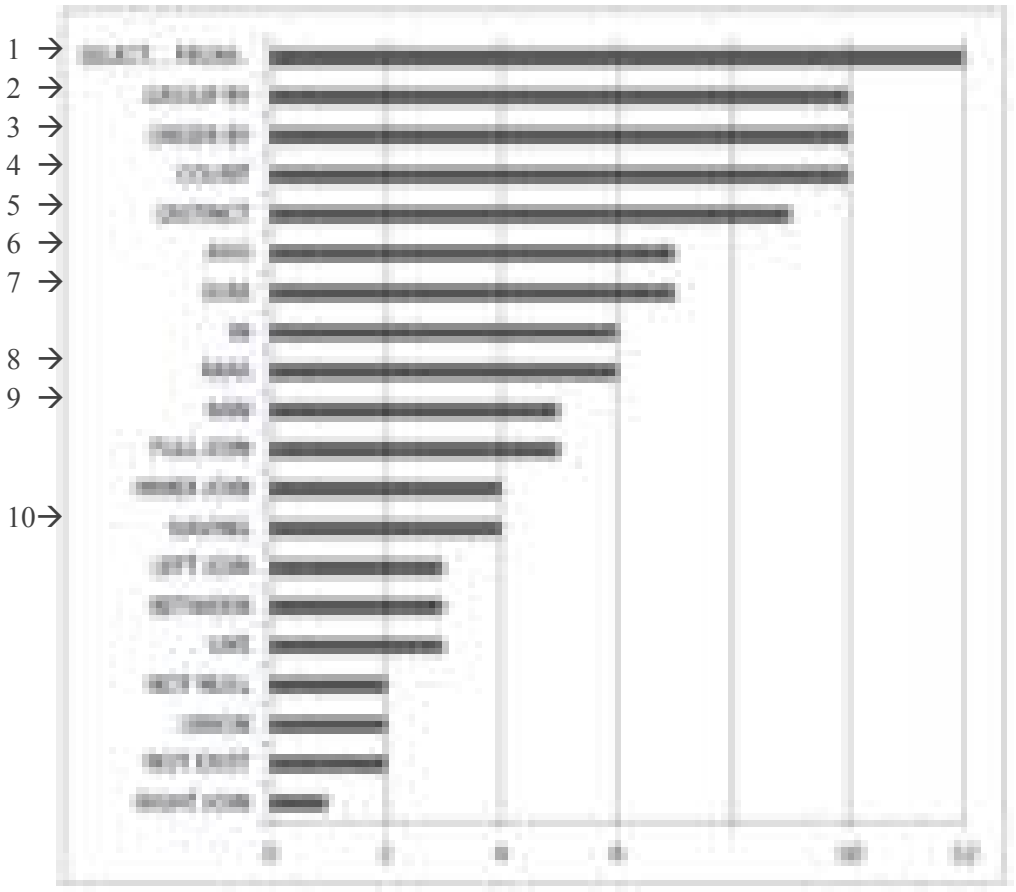

Fig2.6Cláusulas SQL más utilizadas. Esta gráfica muestra el resultado de la encuesta realizada en [39], donde establece las 10 cláusulas SQL más utilizadas por 12 programadores indagados.

Zhang y Sun desarrollaron una herramienta con base en su investigación denominada SQLSynthesizer y dentro de su diseño ellos consideran 3 etapas de construcción de una sentencia SQL (ver fig 2.7):

- Creación del esqueleto
- Terminación de la sentencia
- Candidato de selección

El algoritmo en alto nivel utilizado se establece de la siguiente manera

Toma como elementos de entra y espera en salida:

Entrada: ***TI***ejemplo de las tablas de entrada,

T0 un ejemplo de la tabla de salida

Salida: Una lista "rankeada" de sentencias SQL

(*TI*,*T0*):Las parejas de sentencias SQLSynthesizer

1: queryList← una lista de sentencias vacía

2: skeletons← creación del esqueleto de la Sentencia (TI , T0)

3: **F0R**cada esqueleto se hace

4: conds← infierir las restricciones de (TI , T0, skeleton)

5: aggs← buscar los funciones agregadas (TI , T0, skeleton, conds)

6: columns← buscar la función orderby (T0, skeleton, aggs)

7: queries← Construye las sentencias con las anteriores variables encontradas (skeleton, conds, aggs, columns)

8: **F0R**cada sentencia se hace

9: **if**satisface los Ejemplos (query, TI , T0) ***then***

10: queryList.add(query) adiciona en la lista de sentencias

11: **end if**

12: **end for**

13: **end for**

14: rankQueries(queryList) "rankea" las lista de las sentencias

15: **returnqueryList**devuelve la lista de sentencias.

La línea 2 corresponde a la creación del esqueleto de la sentencia o “SkeletonCreation”los pasos del 3 al 13 corresponde a completar la sentencia o “QueryCompletion” y la línea 14 corresponde a “rankear” los candidatos o “CandidateRaking”

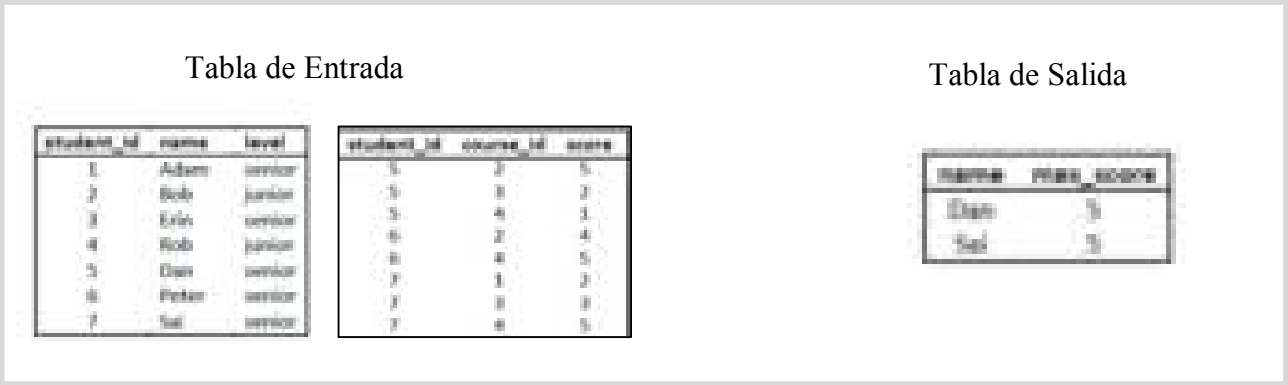

Fig2.7Tablas de entrada y salida [39].

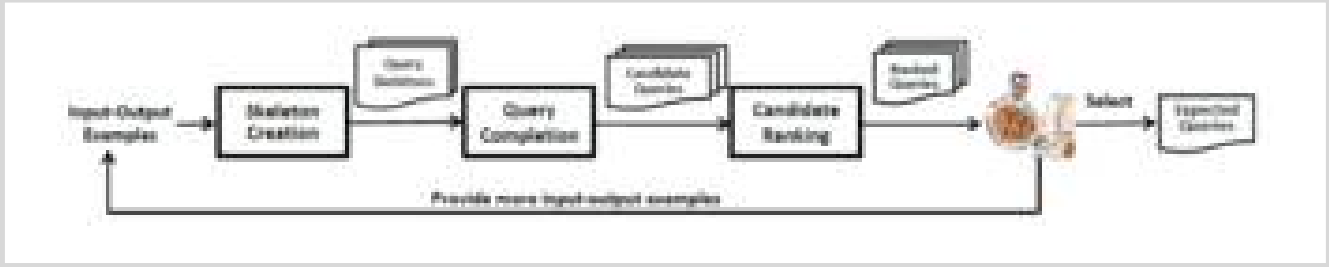

Fig2.8Ilustracióndelflujo de trabajo de SQLSynthesizer[39].

- **Skeletoncreation o Creación del esqueleto:** El programa escanea los ejemplos de entrada y salida dados, y heurísticamente determina el producto cartesiano de las tablas y columnas de resultados, posteriormente con la información recogida, se crea un esqueleto de una consulta SQL,la cual se encuentra incompleta y que se espera completar en la siguiente etapa.

- **Querycompletion o Terminación de la sentencia:**En esta etapa seutiliza un algoritmo de aprendizaje automático para inferir reglas precisas y expresivas, compuesto de múltiples ciclos deverificación. Dentro de las características particulares en esta etapa, se encuentra la agregación de funciones (count, max, min, avg, distint, sum), las cuales son dotadas de un costo o valor numérico (dependiendo de las características de cada sentencia al ser evaluada, verificando los ejemplo de entrada y de salida), por medio de este valor , logran decidir qué función es la más recomendable para completar la sentencia SQL y al final, obtener una lista de sentencias SQL candidatas a ser seleccionadas.

- **Candidate ranking o Candidato de selección:** En esta etapa se toma la lista de sentencias SQL candidatas y a partir de una tabla de reglas de inferencia, se realiza un filtro de tal manera que, si varias sentencias SQL satisfacen los ejemplos, se emplea el principio de la navaja de Occam para clasificar las consultas. Este principio plantea que, entre más simple la sentencia, la posición es más alta en el listado de sentencias de salida. La simplicidad de la sentencia, se define en base a la exigencia de condiciones y/o expresiones contenidas en ella, a cada sentencia se le da un nuevo valor, dependiendo de la cantidad de condiciones, y este valor se aproxima a la longitud de cada sentencia SQL generada.

Este proceso, es realizado varias veces para encontrar la sentencia SQL que satisfaga los ejemplos dados, ya que no siempre de la primera ejecución del programa se obtiene una solución óptima.

En la discusión de su trabajo Zhan y Sun en [39] plantean que, su herramienta no es sólida ni completa ya que, utiliza la heurística para inferir sentencias, condiciones y proyecciones de

columnas, y no se puede garantizar la sentencia correcta para cada caso; sin embargo, encuentran que es útil en la síntesis de una amplia variedad de sentencias. Para la evaluación de su trabajo toman 23 ejercicios de un libro de texto de bases de datos [32] y 5 ejercicios de foros [18] logrando cubrir un amplio margen de tipos de requetés SQL.

Analizando esta investigación, surgió la idea de generar automáticamente sentencias SQL por un medio de aprendizaje diferente al utilizado; buscando en la literatura, se encontró la programación genética como medio de aprendizaje, obteniendo buenos resultados en algunas investigaciones, lo que llevó a emplearla en esta investigación.

2.2.5 Limitaciones del aprendizaje por el ejemplo

Esta técnica de aprendizaje por el ejemplo presenta características especiales para su buena ejecución, si estas características no son cumplidas se convierten en limitaciones.

En nuestro caso se debe contar con muy buenos ejemplos de entrada y salida que representen el comportamiento de los demás elementos que conforman el sistema, estos ejemplos deben ser variados tratando de cubrir varias áreas que permitan sacar buenas conclusiones del sistema, de igual manera la cantidad de estos debe ser suficiente, de no contar los ejemplos con estas características,se obtendrá un conocimientoequivocado del sistema, sacando conclusiones erróneas que no contribuyan en el proceso de aprendizaje.

.

2.3 Aplicación de la programacióngenética.

Existen en la literatura numerosas aplicaciones de la programación genética en diferentes áreas del conocimiento, salud, informática, biología, matemática etc. Su aplicación en particular utilizando ejemplos dentro de la Ingeniería del software se puede ver claramente en estudios como:

- *Generatingmodeltransformation rules fromexamplesusinganevolutionaryalgorithm*[16], donde los autores proponen un enfoque evolutivo para generar automáticamente reglas de transformación de modelos, a partir de un conjunto de ejemplos

- *Genetic-programmingapproach to learn model transformation rulesfromexamples*. [17], donde los autores proponen un enfoque basado en programación genética para aprender automáticamente reglas para transformación de modelo a partir de transformación de parejas de modelos fuente-destino usadas como ejemplos

- *Automaticallysearchingformetamodelwell-fomedness rules in examples and counter-examples*[15], donde sus autores proponen a partir de modelos válidos y no válidos para recuperar automáticamente reglas OCL usando programación genetica

- *Learningimplicit and explicit control in Modeltransformationbyexamples*[7], donde los autores proponen mediante un enfoque evolutivo además de aprender de las reglas paratransformacion de modelos a partir de ejemplos, capturar el control implícito y explícito sobre las reglas transformación.

- *Une approchegénétiquepour la dérivation de règles et métarègles de transformation de modelés à partir d'exemples*[30], donde sus autores proponen el mejoramiento de una aproximación anteriormente presentada donde utilizan ejemplos para la generaciónautomática de reglas paratransformación de modelos

- *OntheApplication of GeneticProgrammingfor Software EngineeringPredictiveModeling: A SystematicReview*[3], donde sus autores investigan la evidencia de la regresión simbólica utilizando la programación genética.

Son muchas las investigaciones donde es utilizada la programación genética, las anteriores son una muestra de la diversidad de campos en la cual es utilizada.Sin embargo, ningún enfoque presenta la generación automática de sentencias SQL desde ejemplos utilizando la programación genética, excepto la que actualmente se presenta.

2.3.1 Ventajas y desventajas de la programación genetica

En general se pueden resumir las siguientes ventajas de los algoritmos genéticos:

- No necesitan conocimientos específicos sobre el problema que intentan resolver.

- Operan de forma simultánea con varias soluciones, en vez de trabajar de forma secuencial como las técnicas tradicionales.

- Cuando los algoritmos genéticos se utilizan para problemas de optimización, maximizar una función objetivo resulta menos afectada por los máximos locales (falsas soluciones) que las técnicas tradicionales.

- Resultan muy fáciles de ejecutarse en las modernas arquitecturas masivamente paralelas.

- Utilizan operadores probabilísticos, en vez de los típicos operadoresdeterminísticos de las otras técnicas.

- Pueden tardar mucho en converger, o no converger en absoluto, dependiendo en cierta medida de los parámetros que se utilicen como tamaño de la población, número de generaciones, etc.

- Pueden converger prematuramente debido a una serie de problemas de diversa índole.

- La repetida evaluación de la función objetivo ya que, en problemas de gran envergadura la evaluación de una única función puede requerir horas o días deestimación

- Su eficiencia computacional puede ser pobre, puede demandar grandes recursos de cómputo para problemas complejos.

- Lo interesante de los algoritmos genéticos, proviene del hecho de tratarse de una técnica robusta, que puede tratar con éxito una gran variedad de problemas provenientes de diferentes áreas, incluyendo aquellos en los que otros métodos encuentran dificultades.

- No se garantiza que un algoritmo genético encuentre la solución óptima del problema, pero existe evidencia empíricaque encuentra soluciones de un nivel aceptable, en un tiempo competitivo con el resto de algoritmos de optimización combinatoria.

- En el caso de existir técnicas especializadas para resolver un determinado problema, lo más probable es que superen al algoritmo genético, tanto en rapidez como en eficacia.

- El gran campo de aplicación de los algoritmos genéticos, se relaciona con aquellos problemas para los cuales no existen técnicas especializadas. Incluso en el caso en que dichas técnicas existan y funcionen bien, pueden efectuarse mejoras de las mismas, combinándolas con los algoritmos genéticos.

CAPÍTULO 3

APROXIMACIÓN

En este capítulo se verá la aproximaciónrealizada, la forma en cómo se realizó la codificación del algoritmo genético, así como una presentación inicial del mismo, el principio aplicado y cómo la teoría explicada en los capítulos precedentes fueutilizada y adaptada.

3.1 Problema general

El objetivo es proporcionar sentencias SQL perfectas a las necesidades de usuarios no expertos, utilizando ejemplos de entrada y salida como información base para el sistema.

Se busca ofrecer una opción a los usuarios sin conocimiento, automatizando la generación de sentencias, para acceder a la información sobre una base de datos a partir de la información que el posee (como ejemplos de datos del sistema). Si bien existen aproximaciones que abordan el problema bajo diferentes metodologías de aprendizaje, estas requieren de condiciones y características especiales para su ejecución, por ejemplo: se necesita que los usuarios tengan conocimientos específicos (como la estructura de la base de datos) o la cantidad de cláusulas del lenguaje de consultas soportado es muy limitada, o los sistemas requieren de la intervención continua de un usuario en diferentes etapas el proceso, lo cual minimiza su autonomía y aplicabilidad en el campo real.

De igual forma, se busca que las sentencias sean automáticamente generadas por el sistema informático sin ninguna intervención del usuario, usando como método de aprendizaje la programación genética, metodología que nos permita encontrar al interior de un gran espacio

de combinaciones posibles (basada en las características propias y heredadas) una solución perfecta para cada ejemplo.

Cada ejemplo está formado por datos de entrada (la información que el sistema tiene) (ver fig. 3.1) y datos de salida (la información que el usuario quiere obtener del sistema) (ver fig. 3.2). Esta información puede ser fácilmente suministrada por cualquier usuario perteneciente a un sistema. El usuario, debe suministrar algunos ejemplos de datos del sistema y debe decir el ejemplo de cómo espera que la información sea retornada, no es necesario que el usuario suministre el listado completo de la información, ya que el sistema debe funcionar como un patrón de valores que se generaliza para el resto de la información.

Una vez actualizados los datos de entrada y salida, se deben crear sentencias SQL válidas hasta encontrar al menos una sentencia SQL que retorne los mismos datos de salida dados en el ejemplo(ver fig. 3.2). Se utilizó el lenguaje de consulta SQL por ser el lenguaje más utilizado por los diferentes motores de bases de datos hasta el momento.

Datos de Entrada (DE)

Realizador

Id_Realizador	Nombre	Nación
1	Tim Burton	Estados-Unidos
2	Jeunet Jean-Pierre	Francia
3	DahanOlivier	Francia
4	Spielberg Steven	Estados-Unidos

Genero

Id_Genero	Genero
1	Ciencia-Ficción
2	Drama
3	Acción
4	Terror

Filme

Id_Filme	Id_Realizador	Id_Genero	Titulo
1	2	2	Amélie
2	4	2	Cheval de Guerre
3	3	2	La Vie en Rose
	4	3	Indiana Jones

Fig3.1Ejemplo de datos de entrada (DE).

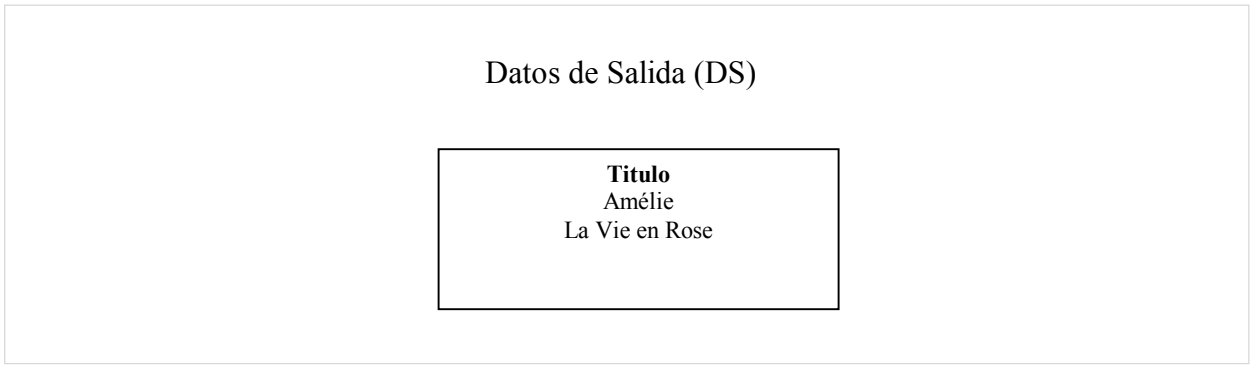

Fig3.2Ejemplo de datos de Salida (DS).

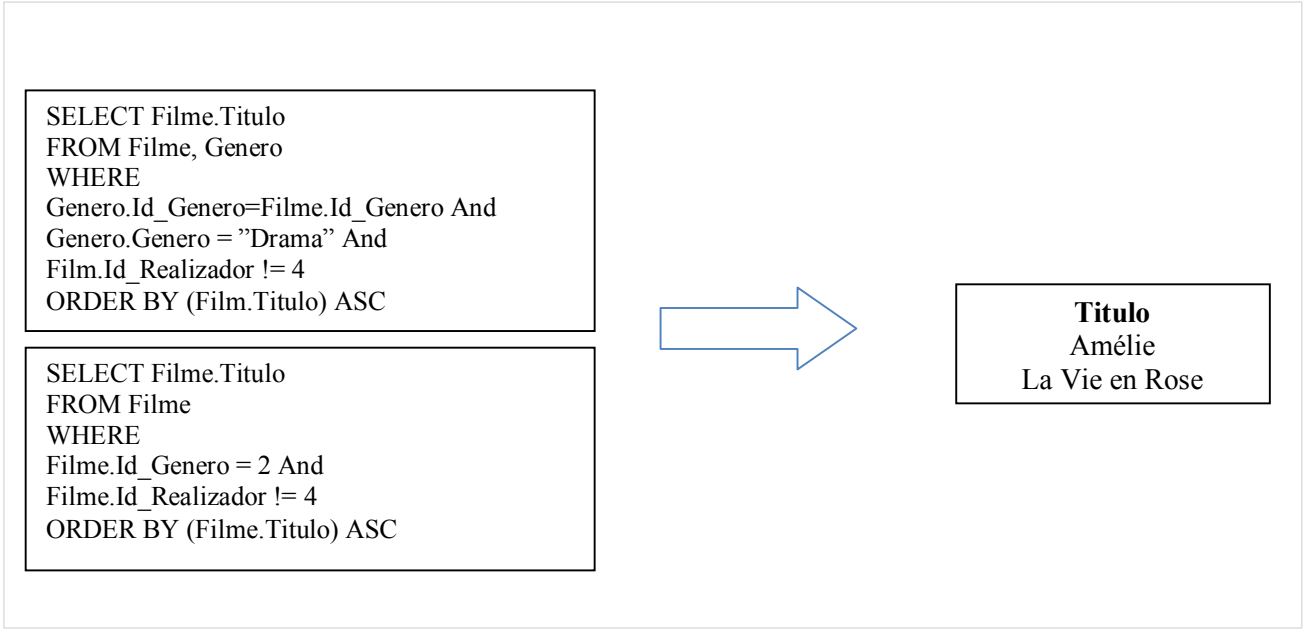

Fig3.3Posibles sentencias SQL se deben genera automáticamente para encontrar una respuesta perfecta.

Si bien existen muchas sentencias SQL que satisfacen el ejemplo dado, no es menester de esta investigación, analizar la intención del usuario al dar los ejemplos, es decir, en el ejemplo de la fig. 3.1 y la fig 3.2, una posibilidad es que el usuario desee obtener el listado de Films donde el director sea de nacionalidad francesa, pero como se puede ver en la fig. 3.3, en ninguna de las sentencias generadas de ejemplo, se toma como condición la nacionalidad del director; sin embargo, generan como respuesta los mismos datos dados en el ejemplo como datos de salida, lo que la convierte para este caso en una solución perfecta para el ejemplo.

3.2 Aproximación

Se propone una aproximación evolucionista para resolver el problema, por el cual a partir de un algoritmo genético, se crea un programa generador automático de sentencias SQL a partir de los ejemplos de entrada y salida.

El algoritmo genético básico se puede ver de la siguiente manera (fig.3.4):

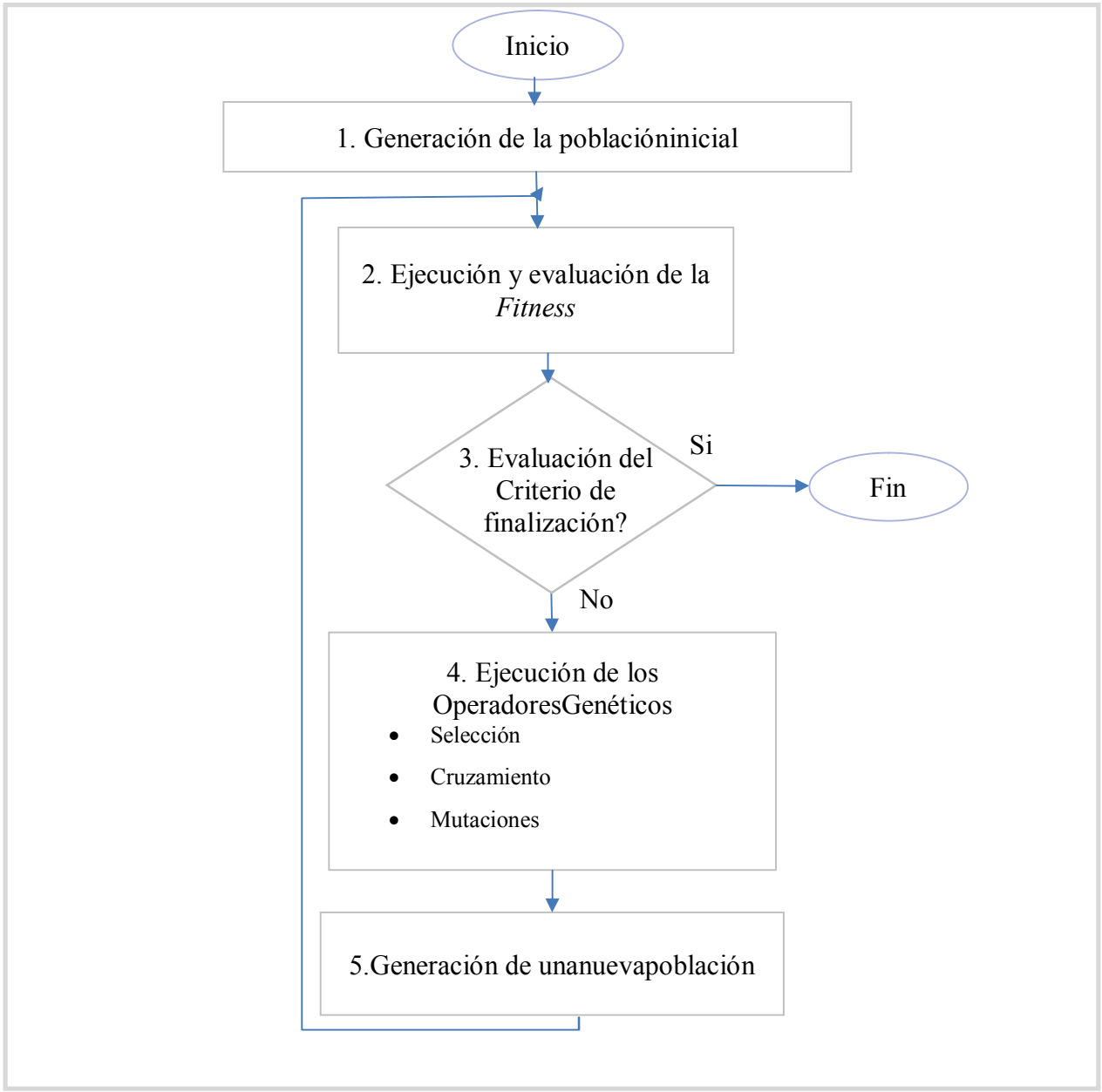

Fig3.4 Diagrama de flujo del algoritmo genético.

1. **Generación de la población inicial:** En esta etapa se crea la primera población de sentencias SQL, estas son generadas aleatoriamente partiendo de la información de los ejemplos dados por el usuario y el tamaño de la población es parametrizado por el mismo.

2. **Ejecución y evaluación de la función objetivo:** En esta etapa, las sentencias SQL son ejecutadas en una base de datos y es evaluada su función objetivo, según la similaridad léxica lograda, atribuyendo un valor numérico a cada sentencia.

3. **Evaluación del criterio de finalización:** Después de realizar la evaluación individual de la población, se verifica si existe un criterio de finalización. Pueden existir múltiples criterios de finalización, por ejemplo, por la cantidad de generaciones o de iteraciones, por el criterio obligado cuando encuentre un valor de función objetivo máximo o cuando se detecte que el programa ha convergido entre otros.

4. **Ejecución de los operadores genéticos:** Una vez verificado el criterio de finalización y este no ha sido positivo, el algoritmo utiliza los operadores genéticos: de selección, de cruzamiento y de mutación, las cuales generan una nueva población de sentencias SQL.

5. **Generación de una nueva población:** Esta nueva población tiene las mismas características que la inicial, en cuanto a la cantidad de sentencias que la conforman, una vez esté completa la población se devuelve a la etapa 2 donde se ejecuta, se evalúa y recomienza el ciclo hasta satisfacer un criterio de finalización.

6. **Fin:** Llegar a esta etapa significa que ha satisfecho algún criterio de finalización y el algoritmo termina.

En cada iteración o generación obtenida o realizada, los individuos evolucionan progresivamente, debido a los aperadores genéticos quienes se van adaptando en pro de resolver el problema, al igual que las especies al medio ambiente.

3.2.1 Codificación del problema

Cada ejemplo dado formado por DE y DS, puede verse como un conjunto de tablas virtuales (siendo DE una o múltiplex tablas y DS una sola tabla). Esta estructura de tablas simple (es decir sin relaciones) posibilita la codificación dentro de cualquier base de datos (ver fig. 3.5), permitiendo de esta forma que, la sentencia SQL generada sea ejecutada por cualquier motor.

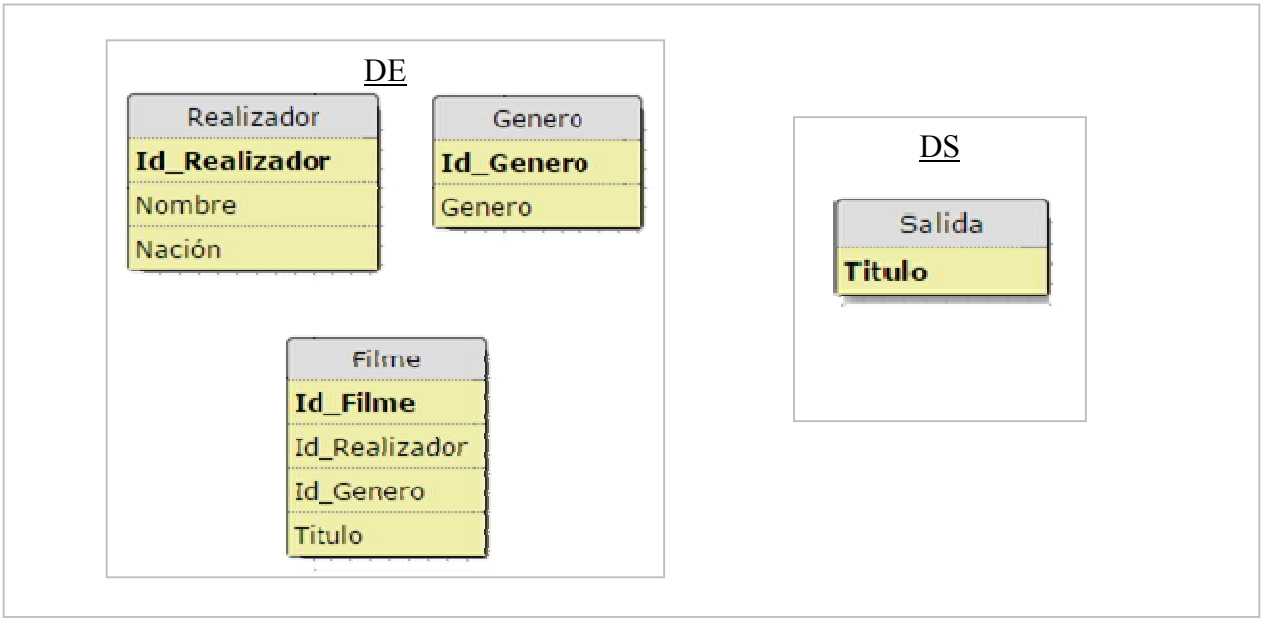

Fig3.5El diagrama presenta el esquema de base de datosaplicado al ejemplo de la Fig. 3.1 y Fig. 3.2, estableciendo las tablas, columnas y registros(datos) a utilizar. (Notar que las tablas se encuentran sin relaciones).

El algoritmo genético produce un programa que está compuesto por un conjunto ***A*** de ***n***sentencias SQL validas ***Ri***(para ***i =1..n***). ("Válida" significa que la sentencia puede ser ejecutada en un motor de bases de datos y no genere error). En la herramienta software, estas sentencias están agrupadas en un arreglo sistemático de objetoso Array

A=

R_1	R_2	R_3	…	R_n

Cada una de estas sentencias ***Ri*** está formada por duplas (***DE,DS***) de características de los ejemplos delos datos de entrada ***DE***(es decir múltiples tablas que pertenecen a **DE**)y datos de salida **DS** (Es decir, una única tabla T que pertenece a **DS**). En la herramienta software, esta información está agrupada en una matriz o arreglo bidimensional.

Ri=

DE_i	DS_i
T_1	T_S
T_2	
T_3	
• • •	
T_x	

Asu vez, cada ***Ti***y***Ts***son matrices compuestas por campos ***Cm***y registros ***Ez***

Ti =

C_{11}	E_{12}	E_{13}	…	E_{1z}
C_{21}	E_{22}	E_{23}	…	E_{2z}
C_{31}	E_{32}	E_{33}	…	E_{3z}
•••	•••	•••	⋱	•••
C_{m1}	E_{m2}	E_{m3}	…	E_{mz}

Cuando una sentencia SQL es ejecutada en una base de datos, produce un conjunto y registros guardados***Si;***conformando una matriz, la cual es comparada con la matriz***Ts*** en ***DS*** para concluir el éxito de la sentencia válida (*j* = número de campos de la tabla T en DS)

Si =

S_{11}	S_{12}	S_{13}	…	S_{1j}
S_{21}	S_{22}	S_{23}	…	S_{2j}
•••	•••	•••	⋮	•••
S_{y}	S_{y2}	S_{y3}	…	S_{yj}

Ts =

C_{11}	E_{12}	E_{13}	…	E_{1j}
C_{21}	E_{22}	E_{23}	…	E_{2j}
•••	•••	•••	⋮	•••
C_{j1}	E_{j2}	E_{j33}	…	E_{yj}

Se busca una sentencia SQL que genere como respuesta una matriz ***Si*** igual a la matriz***Ts.***

3.2.2 Creación de los programas

Como se pudo observar en el capítulo anterior, una sentencia SQL de consulta puede tener la siguiente forma:

```
1.SELECT <champ> [{,<champ>}]
2.FROM<tableau  [{,<tableau>}]
3.[WHERE<condition> [{ AND|OR<condition>}]]
4.[GROUP BY <champ> [{,<champ>}]]
5.[HAVING<condition>[{ AND|OR<condition>}]]
6.[ORDER BY<champ>[ASC | DESC][{,<champ> [ASC | DESC ]}]]
```

Fig3.6Construcción de una sentencia SQL

Partiendo de la información por los ejemplos dados DE y DS (ver fig. 3.1 y fig. 3.2), con la estructura de la base de datos establecida (ver fig. 3.5), las sentencias SQL son construidas de la siguiente forma:

1. Para definir los campos que se van a seleccionar en la sentencia SQL, seempleael nombre de las columnas, el tipo de dato y la cantidad de datos en ***DS*** (Es en esta fase, la única vez en donde la información de ***DS*** es utilizada) y estos datos son comparados con la información de los atributos en las clases ***DE***(ver fig. 3.7). La comparación realizada es léxica y es asignado un valor 0 (cero) a cada match encontrado.

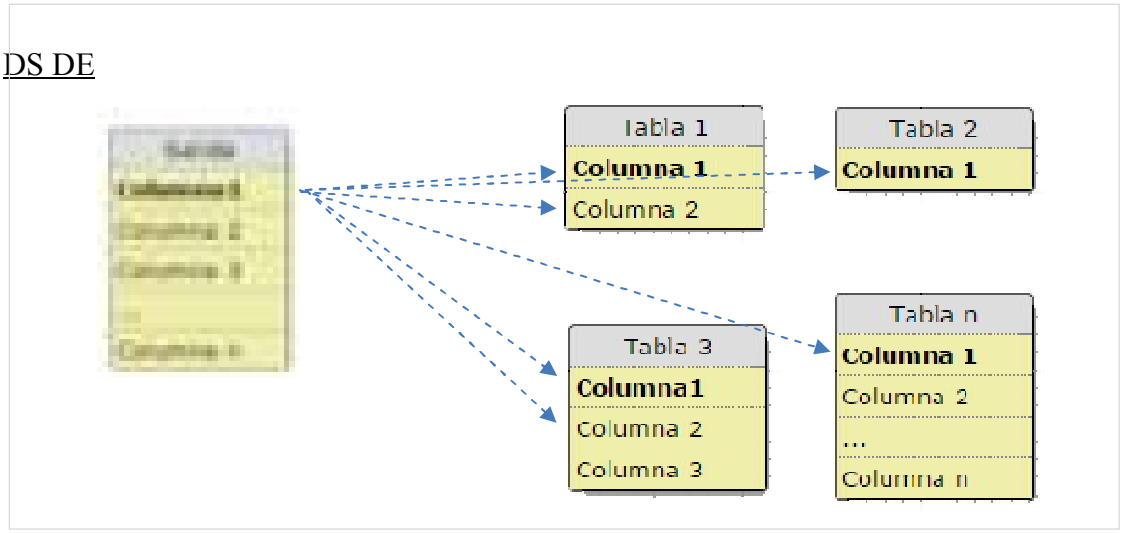

Fig3.7Comparación de atributos.

Cada vez que exista un match, el atributo encontrado es anexado en una lista con la siguiente estructura:

Tabla.Columna // Estos nombres pertenecen a la tabla y columna ***DE***

Se establecen dos listas de similaridad de los campos:

- En la primera lista, se guardan las columnas donde la comparación léxica es igual 0 (cero) es decir, donde los campos comparados en ***DS*** y ***DE*** sean idénticos.

- En la segunda lista, se guardan las columnas con el mismo tipo de datos y de cierta correspondencia léxica, previendo un posible error humano de escritura y así, no excluir tan rápidamente estas columnas.

Cuando estas dos listas son establecidas, se selecciona aleatoriamente un elemento de la primera lista, de estar vacía (es decir, de no haber match) entonces se utiliza la segunda lista y de igual manera es seleccionado un elemento aleatoriamente.

- Una tercera lista es realizada para almacenar las columnas agregadas, es decir, un listado para las columnas que existen en ***DS*** sin que existan coincidencias léxicas idénticamente iguales en ***DE***; por medio del tipo de dato de la columna agregada (int, varchar) se genera un vínculo y se combina con una función(ver Tabla 2.4), esto es realizado por que el nuevo campo construido puede ser el campo que falta por coincidencia en ***DS.*** Su estructura es la siguiente:

Función(Tabla.Columna)	// Estos nombres pertenecen a la tabla y columna ***DE*** // *Función* seleccionada aleatoriamente de la Tabla 2.2.4 dependiendo del tipo de dato faltante en ***DS***

Una vez establecidos los elementos se crea la primer parte de la sentencia SQL

SELECT *Tabla.Columna, Función(Tabla.Columna),* ***…*** *,Tabla**n.**Columna* ***n***

La cantidad y el orden de campos seleccionados aleatoriamente, son controlados por el número de campos y el orden existente en ***DS***.

En la fig. 3.5 se puede ver un ejemplo de cómo el procedimiento funciona, en el ejemplo se encuentra una única coincidencia léxica en el campo *Titulo*y pertenece a la tabla *Filme,* de este modose escribirá:

SELECT*Filme.Titulo*

Como la tabla*Salida*sólo tiene un atributo, entonces sólo se selecciona un único elemento, y como hay comparación léxica 0 y no existen otras listas, entonces *Film.Titre* es el único atributo posible de elección.

2. Para definir los elementos que acompañan la cláusula ***FROM*** (Donde figuran las tablas de procedencia de los datos); en una primera fase de experimentación, se establecieron a partir del nombre de los campos del SELECT, ya que tienen el nombre de la tabla concatenado, (para el ejemplo, la tabla a escoger es *Film);* sin embargo, como se presentan campos generados a partir de la agrupación de las tablas, es decir, campos que no se pueden prever. En una segunda fase de experimentación, la selección de las clases se realizó aleatoriamente. La estructura es la siguiente:

FROM*Tabla, Tabla2…Tablan*

Con los anteriores elementos podría ser generada ya una sentencia SQL. Las cláusulas del 3 al 6 (verfig. 3.6) son utilizadas para especificar una sentencia, siendo seleccionadas y agregadas aleatoriamente.

3. Para formar las cláusulas WHERE y HAVING se utilizan las condiciones generadas aleatoriamente a partir de los atributos, el tipo de dato, operadores aleatorios adecuados (ver tabla 2.2.4.2) y los datos almacenados de las tablas en ***DE***; pero teniendo en cuenta que, las columnas deberían pertenecer a las tablas incluidas en el FROM de la sentencia. Su estructura es la siguiente:

Tabla.Columna(operadores{ < | = |!=| >..}) *{**Tabla.Columna** | **Dato** | **Registro**}*

Para el ejemplo (ver fig. 3.1):

- *Filme.Id_Genero = 2*
- *Filme.Id_Genero> 2*
- *Genero.Id_Genero = Filme.Id_Genero*
- *Genero.Genero = "Drama" ... etc*

Estas condicionesson seleccionadas aleatoriamente y componen la parte del WHERE y HAVING, utilizando AND y OR como separadores entre las condiciones de ser necesario. Como por ejemplo:

WHERE*Filme.Id_Genero = 2 AND Genero.Id_Genero=Filme.Id_Genero*

HAVING*(Filme.Id_Genero> 2)*

4. Para formar las cláusulas GROUP BY y ORDER BY se utilizan las columnas del SELECT y columnas aleatorias de las tablas establecidas en el FROM. Su estructura, es la siguiente:

GROUP BY(Tabla.Columna)

ORDER BY(Tabla.Columna) {*ASC|DESC*} //Palabras claves *ASC oDESC* para orden ascendente o descendente

Como por ejemplo:

GROUP BY *(Filme.Id_Genero)*

ORDER BY (Filme.Titulo) ASC

De la forma anterior, se construyen las diferentes partes de cada una de las sentencias que forman la población inicial.

3.2.3 Creación de la población inicial

- En la población inicial, cada individuo es una sentencia SQL válida, es decir, correctamente construida y ejecutada en una base de datos.

- Esta sentencia debe cumplir los requerimientos establecidos por los ejemplos dados DE y DS como se observó anteriormente.

- El tamaño de la población inicial es determinado por un parámetro N, este número es la cantidad de sentencias SQL válidas a generar, cada sentencia SQL es construida con componentes aleatorios en cada una de sus etapas; sin embargo, se mantiene la

coherencia de la estructura para que al ser ejecutada en la base de datos no genere error.

3.2.4 Evaluación de la función objetivo o *FitnessFunction*

La evaluación de cada programa es un proceso que permite calcular y atribuir un valor con respecto a la función de aptitud, de fitnessu objetivo en cada sentencia SQL por cada generación.

En la programación genética, la función objetivo mide la diferencia o la tasa de error entre el resultado esperado y lo que se ha producido. Aquí, semide la diferencia entre los elementos solución y el patrón, en este caso los elementos de salida, en otras palabras nos permite conocer que tan "buena" es la solución encontrada con respecto a los datos de ejemplo de salida DS.

La función objetivo,se emplea para regular la aplicación de los operadores genéticos, es decir, permitirá controlar el número de selecciones, los cruces y mutaciones que se llevarán a cabo en la generación.

Las sentencias SQL existentes, son ejecutadas en la base de datos para obtener un resultado, este resultado es un conjunto de términos o elementos finito, que puede estar vacío, o compuesto por caracteres y/o números.

Este conjunto de elementos se pueden encontrar lógicamente como un vector o matriz, que contiene a los elementos ordenados en filas y columnas, que posee un número *N* determinado de elementos y cada uno de estos, es referenciado por la posición que ocupa dentro de la matriz (como se observó en la codificación del problema). Este conjunto de

elementos resultantes será comparado con los elementos que hacen parte del ejemplo de salida dado ***DS.***

En cada comparación, se evalúa el elemento del conjunto resultado basado en 4 factores:

- Cantidad de elementos retornados
- Existencia del elemento
- Posición del elemento
- Posición del elemento con respecto al siguiente (orden)

Se establecieron dos matrices: La matriz Si de elementos de salidaDS y la matriz Ts de ejemplos de resultado (ver fig. 3.8).

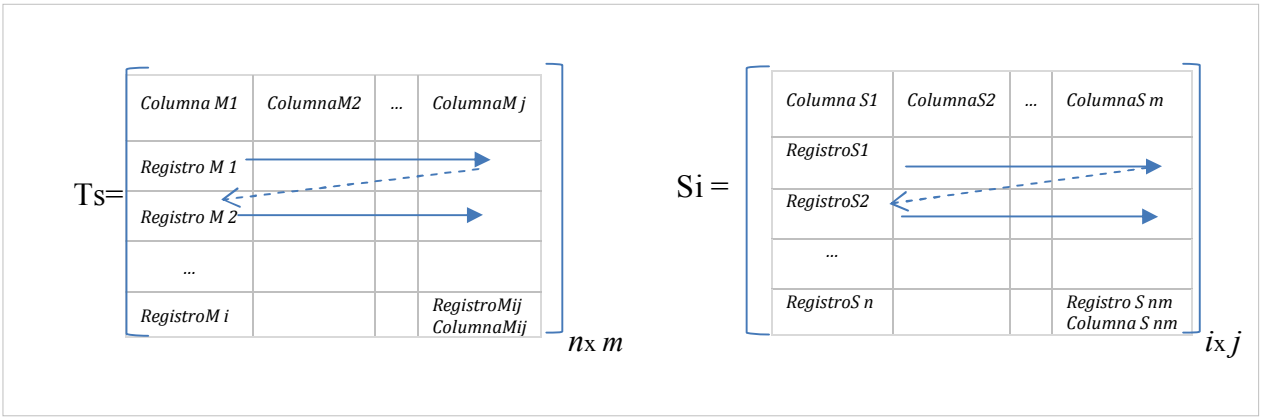

Fig3.8Matrices de elementos de resultado***Si*** y los elementosde salida***Ts***

Para realizar la comparación, se ejecutó registro por registro y se realizó en dirección como muestran las flechas de la fig. 3.8. Se establecieron 4 etapas las cuales se definieron después de haber probado otras combinaciones de patrones a tener en cuenta y se encontraron mejores resultados con la siguiente combinación:

1. Se compara el tamaño de las matrices. El tamaño de la matriz $\boldsymbol{Ts}_{n \times m}$debe ser igual a la matriz$\boldsymbol{Si}_{i \times j}$. El tamaño se almacenó en la variable *L*

2. Se verificó la existencia de cada registro ***Si*** en la matriz ***Ts***, la cual se almacenó en la variable C.

3. Se obtuvo la posición en la matriz***Si***en la que se encontró el registro y se comparó conla posición establecida en***Ts***. La posición se almacenó en la variable *P*.

4. Se comparó el orden, el registro x de la matriz ***Ts***yel registro x+1 correspondes a los elementos en la misma posición en la matriz ***Si***, entonces se considera que la matriz está ordenada. El orden se almacenó en la variable *O*.

Según la cantidad de aciertos en cada fila de registros, se promedia y se asigna un valor, el cual podría ser máximo 1. Al final se realiza un promedio general basado en los 4 aspectos abordados.

$$FO = \frac{\mathrm{L} + \mathrm{C} + \mathrm{P} + \mathrm{O}}{4}$$

Calculado de la siguiente forma:

$$L = \begin{cases} 1 & Si\ los\ tamaños\ son\ iguales \\ 0 & Si\ no \end{cases}$$

$$C = \frac{\sum_{x=1}^{i*j} r\,x}{i*j}\ r = Registro\ \boldsymbol{Si} = \begin{cases} 1 & si\ \ existe \\ 0 & si\ no \end{cases}$$
$$i*j = tamaño\ de\ la\ matriz$$

$$P = \frac{\sum_{x=1}^{i*j} r\,x}{i*j}\ r = Registro\ \boldsymbol{Si} = \begin{cases} 1 si\ la\ posición\ es\ la\ correctas \\ 0 \qquad\qquad si\ no \end{cases}$$
$$i*j = tamaño\ de\ la\ matriz$$

$$O = \begin{cases} 1 & si\ la\ posición\ es\ la\ correcta\ con\ respecto\ a\ la\ siguiente \\ 0 & si\ no \end{cases}$$

Para ejemplarizar como se calcula la función objetivo planteamos 2 sentencias SQL generadas apartir dela información de la figura 3.1 y figura 3.2 :

1. `SELECT filme.titulo FROM filme WHERE 1 GROUP BY (filme.titulo).`

TABLA RESULTADOS TABLA DE SALIDA (DS)

Ts=

Titulo
Amelie
Cheval de guerre
Indiana jones
La vie en rose

Si =

Titulo
Amelie
La vie en rose

Calculamos la función objetivo como sigue:

$$FO = \frac{\mathrm{L} + \mathrm{C} + \mathrm{P} + \mathrm{O}}{4}$$

1. Para encontrar ***L***comparamos el tamaño de las 2 matrices,
   - ***Ts*** de tamaño 4 y ***Si*** de tamaño 2, al ser diferente y según las especificaciones antes vistas entonces ***L = 0***

2. Para encontrar C verificamos la existencia de cada registro ***Si*** en ***TS i***

$$C = \frac{\sum_{x=1}^{i*j} r\,x}{i*j} \qquad r = Registro\ \boldsymbol{Si} = \begin{cases} 1 & si\ \ existe \\ 0 & si\ no \end{cases}$$

$$C = \frac{\sum_{x=1}^{2} r\,x}{2} \quad \begin{matrix} r1 = amelie \\ r2 = la\ vie\ en\ rose \end{matrix}$$

$$C = \frac{1+1}{2} \quad \begin{matrix} r1 = amelie = 1\ porque\ si\ existe\ en\ Ts \\ r2 = la\ vie\ en\ rose = 1\ porque\ si\ existe\ en\ Ts \end{matrix}$$

$$C = 1$$

3. Para encontrar P verificamos que la posición del elemento en ***Si*** sea la misma en ***Ts***

TABLA RESULTADOS

Ts=

Titulo	***posición***
Amelie	1
Cheval de guerre	2
Indiana jones	3
La vie en rose	4

TABLA DE SALIDA (DS)

Si=

Titulo	***posición***
Amelie	1
La vie en rose	2

$$P = \frac{\sum_{x=1}^{i*j} r\,x}{i*j} \quad \begin{matrix} r = Registro\ \boldsymbol{Si} = \begin{cases} 1 si\ la\ posición\ es\ la\ correctas \\ 0 \quad si\ no \end{cases} \\ i*j = tamaño\ de\ la\ matriz \end{matrix}$$

$$P = \frac{\sum_{x=1}^{2} r\,x}{2} \quad \begin{matrix} r1 = amelie\ posición\ 1 \\ r1 = la\ vie\ en\ rose\ posición\ 2 \\ i*j = 2 \end{matrix}$$

$$P = \frac{1+0}{2} \quad \begin{matrix} r1 = amelie\ posición\ 1 = 1\ porque\ la\ posicion\ es\ la\ correcta\ en\ \boldsymbol{Ts} \\ r1 = la\ vie\ en\ rose\ posición\ 2 = 0\ porque\ la\ posición\ NO\ es\ correcta\ en\ Ts \end{matrix}$$

$$P = \frac{1}{2} = 0{,}5$$

4. Para encontrar ***O*** verificamos el orden de los registros y como se puede ver la posición del primer elemento es la correcta pero la del siguiente elemento de la matriz ***Ts*** no es la misma con respecto a la matriz ***Si*** por lo cual ***O=0***

5. Para finalizar se calcula la función objetivo

$$FO = \frac{\mathrm{L} + \mathrm{C} + \mathrm{P} + \mathrm{O}}{4} = \begin{matrix} L = 0 \\ C = 1 \\ P = 0{,}5 \\ O = 0 \end{matrix}$$

$$FO = \frac{1{,}5}{4} \rightarrow FO = 0{,}375$$

2. `SELECT filme.titulo FROM filme WHERE filme.titulo='la vie en rose' OR filme.id_realizador=2.`

`TABLA RESULTADOS` `TABLA DE SALIDA (DS)`

Ts=

Titulo
Amelie
La vie en rose

Si =

Titulo
Amelie
La vie en rose

Calculamos la función objetivo como sigue:

$$FO = \frac{\mathrm{L} + \mathrm{C} + \mathrm{P} + \mathrm{O}}{4}$$

1. Para encontrar ***L***comparamos el tamaño de las 2 matrices por lo cual ***L=1*** ya que son iguales.
2. Para encontrar ***C***verificamos la existencia de los registros de ***Si*** en ***Ts***, por lo cual ***C=1*** ya que los 2 registros existen en ***Ts***
3. Para encontrar ***P***verificamos la posición de los registros de ***Si*** en ***Ts***, por lo cual ***P=1***

4. Para encontrar O verificamos el orden de la matriz ***Ts*** con respecto a ***Si,*** por lo cual ***O=1.***

5. Se calcula la función objetivo:

$$FO = \frac{\mathrm{L} + \mathrm{C} + \mathrm{P} + \mathrm{O}}{4} = \frac{1 + 1 + 1 + 1}{4} \quad \rightarrow FO = 1$$

3.2.5 Operadoresgenéticos

Como se observó en un capítulo precedente, para evolucionar poblaciones se utilizan los operadores, este enfoque utiliza operadores genéticos de selección, de cruzamiento y mutación. Por supuesto, estos operadores garantizan la validez de los programas evolucionados.

3.2.5.1 Selección

Se realiza una selección en la población para determinar qué individuos se verán afectados. Para esto se utiliza el principio de la ruleta o *Roulotte Wheel Selection,* donde se da más oportunidad de evolucionar programas con una buena evaluación de aptitud o función objetivo, dejando una pequeña posibilidad a las más bajas evaluaciones. Esta ruleta genera un par de elementos aleatorios los cuales se denominaron como padres.

Otro operador utilizado es la selección elitista, la cual fuerza a que mejores individuos de la población sean seleccionados, la cantidad de estos, está definida por un valor parametrizable y de esta forma, se asegura la presencia de individuos con las mejores evaluaciones de aptitud ya que, estos se agregan directamente en la siguiente generación.

Normalmente, el valor utilizado en la selección elitista está dado por un porcentaje de la población, por ejemplo, para un valor de elitismo del 5 % en una generación de 1000 individuos, aseguraría que los 50 más aptos de su generación pasen a la siguiente. Para el 95% restante se utiliza la ruleta, esta genera los padres correspondientes a la proporción, utilizando el 100% de individuos de la población y ellos estarían sujetos al cruce y la mutación en esa generación.

3.2.5.2 Cruzamiento

El cruzamiento permite crear nuevas sentencias SQL a partir de los padres *P* seleccionados. Estos padres se cruzan y algunas características son cambiadas buscando mejorar la adaptación al medio, en nuestro caso mejorar el valor de la función objetivo.

El punto de cruce en las sentencias SQL padres, se seleccionó aleatoriamente, se establecieron puntos en los cuales se podría generar un cruce para que la sentencia SQL semánticamente fuera válida:

12 Puntosposibles de cruce (verTabla 3.1):

WHERE	**ORDER BY**	**GROUP BY**	**HAVING**	COUNT	**DISTINCT**	**AVG**	**SUM**	MAX	**MIN**	**AND**	**OR**

Tabla 3.1Puntosposibles de cruzamiento

El punto de cruce se establece aleatoriamente y se realiza el mismo punto en cada padre *P,* después se cambia la parte izquierda al punto elegido, generando por cada par de

padres, 2 nuevas sentencias SQL denominadas hijos *H,* el cruce se efectúa bajo cierta probabilidad que indica la frecuencia con la que se producen cruces.

Por ejemplo, se establece un arreglo con las 12 cláusulas posibles y después se selecciona un número aleatorio, en este caso y para el ejemplo, el número 6, entonces se busca la cláusula en la posición 6 = "AVG", después se busca en los padres P1 y P2 la cláusula "AVG". Cada padre es dividido en la cláusula seleccionada y se guarda en los arreglos hijos H1 y H2 (ver fig. 3.9)

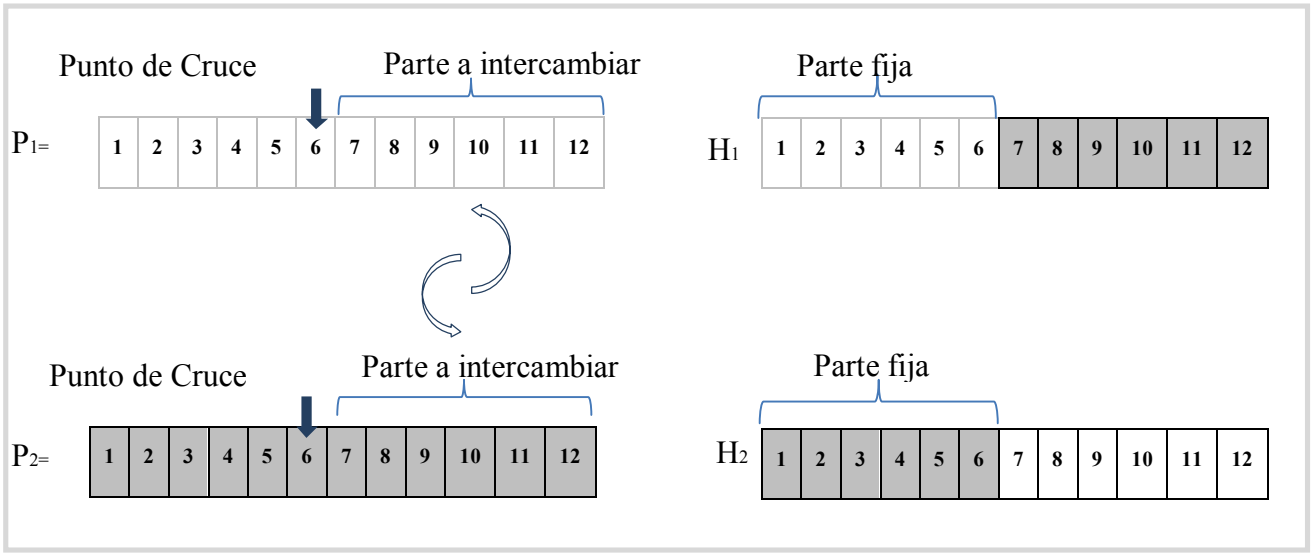

Fig3.9Cruzamiento

De no existir en ninguno de los padres el punto de cruce seleccionado, se vuelve a generar un nuevo punto de corte aleatorio hasta que al menos uno de los dos padres pueda generar un cruce.

Para ejemplarizar este proceso se plantea 2 sentencias SQL generadas a partir de la información de la figura 3.1 y figura 3.2 a quienes denominamos padres P1 y P2

1. `SELECTfilme.titulo FROM filme WHERE filme.titulo='la vie en rose' ANDfilme.id_genero=2.`
2. `SELECT filme.titulo FROM filme WHERE filme.id_filme=1 AND filme.id_genero=2 OR filme.id_realizador=2.`

Se selecciona aleatoriamente un punto de cruzamiento de los 12 posibles establecidos en la tabla 3.1. en este caso el punto de cruce la palabra "AND";

P1. `SELECT filme.titulo FROM filme WHERE filme.titulo='la vie en rose' ANDfilme.id_genero=2`

P2. `SELECT filme.titulo FROM filme WHERE filme.id_filme=1 ANDfilme.id_genero=2 OR filme.id_realizador=2`

Una vez establecido el punto de cruce se intercambia la parte a la siguiente del punto de corte, para el ejemplo la parte resaltada, y se generan los hijos, H1 y H2

H1. `SELECT filme.titulo FROM filme WHERE filme.titulo='la vie en rose'ANDfilme.id_genero=2 OR filme.id_realizador=2`

H2. `SELECT filme.titulo FROM filme WHERE filme.id_filme=1ANDfilme.id_genero=2`

3.2.5.3 Mutación

En la mutación, se busca crear nuevo material genético para las sentencias SQL permitiendo así alcanzar zonas de espacio de búsqueda que no estaban cubiertas por la población actual.

La mutación es utilizada de manera conjunta con el cruzamiento, una vez el cruce tenga éxito entonces uno a uno los hijos obtenidos se mutan bajo cierta probabilidad; esta probabilidad suele ser muy baja, de la misma forma cuando se habla en términos evolutivos, la mutación se manifiesta de forma extraordinaria (nada común), por lo tanto, la mutación modifica ciertas partes de la sentencia SQL de forma aleatoria, atendiendo a la probabilidad establecida.

Los puntos posibles de la mutación (ver fig. 3.10)

WHERE	ORDER BY	GROUP BY	HAVING	COUNT	DISTINCT	MAX
SUM	MIN	AVG	OR	AND	DESC	ASC
COUNT(DISTINCT	>	>=	<	<=	!=	=

Fig.3.10Puntos posibles de mutación

Se establecen puntos posibles de mutación y aleatoriamente se selecciona uno de estos, la cláusula correspondiente a este punto se busca dentro de la sentencia SQL, si esta no se encuentra, se selecciona otro punto aleatorio hasta encontrar un término que exista en la sentencia SQL. Una vez encontrado el término, se selecciona aleatoriamente al interior de un subconjunto los posibles cambios que podrían ser realizados, por ejemplo:

- Si en la primera selección aleatoria surge el termino: `">"`

  El subconjunto de mutación será :`{"<", ">=", "<=", "=", "!="}`

- Si en la primera selección aleatoria surge el termino: `"ASC"`

  El subconjunto de mutación será :`{"DESC"}`

- Si en la primera selección aleatoria surge el termino: `"AND"`

  El subconjunto de mutación será :{`"OR"`}

- Si en la primera selección aleatoria surge el termino: `"MAX"`

  El subconjunto de mutación será :{`"AVG", "SUM", "MIN", "COUNT"`}

En el caso de existir términos repetidos dentro de la sentencia SQL analizada, aleatoriamente se decide cuántos y cuáles deben ser mutados, (este es el caso del término AND et OR ya que por ser conectores de condiciones se encontraba en múltiples partes).

Por ejemplo, se selecciona el número aleatorio 6= "MAX" y del respectivo subconjunto el número aleatorio 2 ="SUM", después se realiza la mutación (ver fig. 3.11).

Fig3.11Ejemplo de la Mutación

Para ejemplarizar este proceso se plantea 1 sentencias SQL generadas a partir de la información de la figura 3.1 y figura 3.2 a quienes denominamos hijos H1 y H2

H1.`SELECTfilme.titulo FROM filme WHERE filme.titulo='la vie en rose' AND filme.id_genero=2.`

Se selecciona aleatoriamente un punto de mutaciónpara H1 de los posibles establecidos en la tabla 3.2, en este caso el punto de mutación la palabra "AND" y se selecciona del subconjunto de mutación posible un elemento en este caso el único posible es "OR"

H1.`SELECTfilme.titulo FROM filme WHERE filme.titulo='la vie en rose' ANDfilme.id_genero=2.`

Una vez establecida la palabras a mutar y su subconjunto se realiza la mutación en H1 y nace un nuevo hijo al que se denomina H1'.

H1'.`SELECTfilme.titulo FROM filme WHERE filme.titulo='la vie en rose' ORfilme.id_genero=2.`

Y para finalizar, se estableció como criterios de finalización:

- El valor máximo de la función objetivo se establecio en : 1
- La cantidad de generaciones máximas:*10*

CAPÍTULO 4

EVALUACIÓN

Este trabajo de investigación, nace con la iniciativa de aprendersentencias SQL a partir de ejemplos, dentro de la literatura estudiada se encontró un trabajo de investigación realizado por un grupo de la universidad de Washington [39], que evaluó su investigación en 23 ejercicios de un libro de base de datos [32] y en 5 ejercicios de foros de preguntas de internet [18], utilizaron estos ejercicios ya que están diseñados con el propósito de ser un reto y a menudo diseñados para cubrir una amplia gama de características de SQL, incluyendo casos poco realistas dentro de su uso.

La evaluación de la aproximación en la síntesis de sentencias SQL se realizó bajo 4 aspectos:

1. Porcentaje de éxito.
2. Tiempo necesario.
3. Esfuerzo humano.
4. Eficacia comparada en técnicas de inferencia de sentencias SQL existentes.

En este trabajo de investigación, se cubrirán estos aspectos y se validará la aproximación propuesta, con el mismo grupo cubierto de cláusulas SQL y el mismo juego de datos, en este caso los 28 ejercicios establecidos.

4.1 Implementación

La implementación de la aproximación propuesta, es una herramienta software desarrollada en JAVA [25] denominada EvoSQL. El framework se ocupa de la construcción y la aplicación de cada sentencia SQL. La ejecución de cada sentencia SQL generada, es realizada sobre el motor de base de datos Mysql [29], el cual devuelve el conjunto de datos solución. Las tablas utilizadas soportan enteros y caracteres comotipos de datos de entrada.

Se establecieron bases de datos individuales para cada ejercicio las cuales constan de una única tabla de salida y múltiples tablas de entrada, en el caso de los ejercicios evaluados el número máximo de tablas ingresadas por cada uno fue de 4. Los nombres de las tablas, el conjunto de los campos y los datos, fueron escritos de forma idéntica a los expuestos y se encuentran disponibles en la página web prevista porel grupo de investigación en [13].

Las pruebas fueron realizadas en un equipo con 2.5GHZ Intel Core i5 con 4GB de memoria física bajo Windows 7 profesional edición.

4.2 Resultadose interpretación

Se ejecutaron 28 ejercicios sobre la herramienta software EvoSQL, todas las pruebas se realizaron bajo las siguientes condiciones:

- Una población de 1000 sentencias válidas SQL

- 10 generaciones como límite máximo de reproducción, ya que en las pruebas anteriores en SQLSynthesizer[39] utilizaron este mismo criterio de finalización.
- La proporción considerada en la selección por Elitismo fue del 10%.

- El porcentaje considerado en el cruzamiento del 60%.
- El porcentaje considerado en la Mutación del 20%.

Se obtuvieron los siguientes resultados:

No	*Ejercicio*		*EvoSQL*			*Éxito*		
	Número	Tamaño Número de tablas	Costo del tiempo (minutos)	Fitness Max	#Generación	QuerybyOutput QBO	SQLSynthesizer	**EvoSQL**
1	**T 5.1.1**	4	4:02	1	1	Si	Si	Si
2	**T 5.1.2**	4	0:25	1	0	No	No	Si
3	**T 5.1.3**	2	5:40	1	9	No	Si	Si
4	**T 5.1.4**	3	6:15	0,8484	10	No	No	No
5	**T 5.1.5**	2	0:43	1	0	No	Si	Si
6	**T 5.1.6**	3	1:07	1	1	No	Si	Si
7	**T 5.1.7**	1	2:05	1	1	No	Si	Si
8	**T 5.1.8**	1	2:08	1	1	No	Si	Si
9	**T 5.1.9**	2	0:28	1	1	No	Si	Si
10	**T 5.1.10**	2	1:50	1	1	Si	Si	Si
11	**T 5.1.11**	4	8:29	1	6	No	No	Si
12	**T 5.1.12**	4	1:58	1	1	No	No	Si
13	**T 5.2.1**	2	0:28	1	0	No	Si	Si
14	**T 5.2.2**	3	1:46	1	1	No	Si	Si
15	**T 5.2.3**	3	1:22	1	0	No	No	Si
16	**T 5.2.4**	3	8:56	1	6	No	Si	Si
17	**T 5.2.5**	3	5:28	0.8518	10	No	No	No
18	**T 5.2.6**	3	5:52	0.9148	10	No	Si	No

19	**T 5.2.7**	3	1:40	1	1	No	Si	Si
20	**T 5.2.8**	3	0:52	1	0	No	No	Si
21	**T 5.2.9**	3	4:07	1	5	No	Si	Si
22	**T 5.2.10**	3	10:45	0.8260	10	No	Si	No
23	**T 5.2.11**	3	10:15	0.8333	10	No	No	No
24	**Fórum 1**	1	0:17	1	0	No	Si	Si
25	**Forum.2**	1	1:27	1	1	No	Si	Si
26	**Fórum 3**	1	0:36	1	0	No	Si	Si
27	**Fórum 4**	4	0:11	1	0	No	Si	Si
28	**Fórum 5**	3	0:28	1	0	No	Si	Si
	Moyenne		3:11	0.9760	3.071	0.07142	0.7142	0.8214

Tabla 4.1Resultados de la ejecución de 28 ejercicios sobre EvoSQL y comparación de éxito con las herramientas software más cercanas en características a EvoSQL (QBO, SQLSynthesizer).

4.2.1 Porcentaje de éxito

Al inicio de esta investigación, se consideró la dificultad que representaba el conseguir buenos resultados al utilizar valores aleatorios para la creación de Sentencias SQL, debido en gran parte a la rigurosidad que presentaba la construcción de una sentencia y más aún, con la cantidad de valores posibles de combinaciones y el gran campo de exploración establecido; sin embargo, se encontró en cuanto al porcentaje de éxito, muy buenos resultados de la herramienta EvoSQL basada en la aproximación presentada en esta memoria ya que, en la tabla 4.1 de resultados, se puede ver claramente que la herramienta EvoSQL, obtuvo 82% de éxito para los casos de estudio evaluados. En los ejercicios donde EvoSQL no tuvo éxito, se identificó que para encontrar solución perfecta, era necesario utilizar cláusulascomplementarias como la cláusula IN, la cual no fue objeto en esta investigación y

aunque se obtuvieron valores muy cercanos al óptimo en la función objetivo, no se logró encontrar una combinación perfecta de cláusulas.

Evaluación de los Ejercicios	*EvoSQL*	*SQLSynthesizer*
Éxito	23 exercices	20 exercices
SinÉxito	5 exercices	8 exercices
	4, 17, 18, 22, 23	2, 4, 11, 12, 15, 17, 20, 23
	18,22	Si
	Si	2, 11, 12, 15, 30
Comunes SinÉxito	4, 17, 23	

Tabla 4.2Evaluación de éxito para los 28 ejercicios, comparación entre EvoSQL y SQLSynthesizer

EvoSQL logró encontrar la solución a 5 ejercicios (2, 11, 12, 15, 20) donde SQLSynthesizer no tuvo éxito, clasificándolos como: “Sin solución” con la combinación de cláusulas cubierta (ver tabla. 4.1); sin embargo, se encontró una combinación satisfactoria a estos ejercicios. Por contraste, EvoSQL no tuvo éxito en 2 ejercicios (18, 22) en los cuales SQLSynthesizer si lo obtuvo, y en 3 ejercicios (4, 17, 23) conjuntamente, no se encontró solución perfecta ni con EvoSQL ni con SQLSynthesizer (ver tabla 4.2).

Por ejemplo en el ejercicio 4 y 17 donde fue común no encontrar solución ni en EvoSQL y SQLSynthesizer la sentencia necesaria para encontrar la perfecta solución al caso de estudio, requiere de dos sentencias anidadas, ya que se necesita de los datos de una sub-consulta para realizar una restricción específica, al no utilizar sentencias anidadas nuestra aproximación no se encontrara esta solución.

Para los ejercicios 18, 22 y 23 consideramos que no se logró construir la combinación correcta de restricciones en la consulta, tal vez debido a la cantidad y lo especifico de estas, el azar no permite definir lógicamente cual sentencia falta, o cual sobra para formar la sentencia SQL correcta.

4.2.2 El tiempo necesario.

El tiempo en promedio obtenido en la evaluación de los ejercicios es de 3 minutos 11 segundos. Hay que considerar que, se genera una población de 1000 sentencias válidas partiendo de valores aleatorios, lo que representa un tiempo base en su ejecución por lo cual, el tiempo promedio obtenido para programas de este tipo (es decir algoritmos genéticos) es aceptable.

Los ejemplos utilizados se consideran de tamaño pequeño, no mayor a 4 tablas en la base de datos, en cada ejemplo para la generación de una determinada población requiere de un tiempo base como tiempo de ejecución, el mayor consumo de tiempo en EvoSQL es realizado en la generación de la población inicial, ya que el primer problema a enfrentar fue generar una sentencia válida a partir de elementos aleatorios debido a las características del lenguaje SQL, ya que al ser estructurado deja muy poco al azar, todo está anidado y tiene un seguimiento lógico que muy difícilmente se puede conservar con elementos de este tipo(azar), lo que significa que se podían presentar muchos ciclos de evaluación hasta encontrar una

combinación válida donde la sentencia SQL no generara errordurante su ejecución; la generación de la población inicial ronda alrededor de 1 minuto, en algunos casos se obtuvo en la población inicial una solución perfecta en17 segundos por tiempo de ejecución; sin embargo, por las características propias de los algoritmos genéticos el tiempo de ejecución no es una de las ventajas obtenidas por el tipo de metodología de aprendizaje utilizado.

4.2.3 Esfuerzo humano.

El esfuerzo humano establecido en la escritura de los ejercicios (que se basa en el llenado las bases de datos) sigue siendo el mismo que en SQLSyntesizer[39], pero a diferencia de este, EvoSQL se diseñó con la variante de aceptar en parte, el error humano en la escritura de los campos (al menos con algunos caracteres de diferencia), lo cual no es permitido por SQLSynthesizer y genera una característica favorable en EvoSQL.

Cabe resaltar que, los resultados obtenidos en la tabla 4.1 se deben a la realización de una única prueba con múltiples generaciones, es decir, el usuario interviene una única vez en el desarrollo del programa, lo cual lo convierte en totalmente automático, mientras que en SQLSynthesizer[39] plantean varias iteraciones para encontrar la perfecta solución, establecen 2.3 rounds de interacción del usuario con el programa en promedio que van entre mínimo 1 minuto y 5 minutos como máximo, lo que implica que sus tiempos de ejecución y el esfuerzo humano se incrementan y aunque EvoSQL continua con mayores tiempos de ejecución, se diferencia en el tipo de programación utilizada (Es decir programación genética versus la programación tradicional que debería ser en teoría mucho más rápida), lo cual se considera representa un punto a favor de EvoSQL.

4.2.4 Eficacia comparada en técnicas de inferencia de sentencias SQL existentes.

Se comparóEvoSQL con SQLSynthesizer[39] y Queryby Output QBO [31] porque son dos herramientas que infieren sentencias SQL más próximas a esta investigación, donde se utilizan ejemplos para la generación de sentencias.

Mientras que EvoSQL y SQLSynthesizer cubren el mismo campo de cláusulas SQL, QBO no soporta características como GROUP BY y HAVING, lo que en primera instancia lo deja en desventaja frente a las 2 aproximaciones, de igual manera, fueron evaluados los mismos ejercicios en esta herramienta, en donde se logró obtener 2 resultados de éxito.

Se puede observar en la fig. 4.2 la gráfica comparativa del porcentaje de éxito, obtenido en las pruebas donde se obtuvo resultado exitoso para 23 ejercicios y sin éxito en 5 ejercicios, esto representa que EvoSQL logra obtener un 82,1% de efectividad. Esta efectividad es superior en 10.7% a la establecida por la herramienta desarrollada en [39]SQLSynthesizer y 75% más alta que la herramienta siguiente más próxima Queryby Output QBO [31], lo cual convierte a EvoSQL en la herramienta con más éxito en la generación de sentencias SQL a partir de ejemplos hasta el momento y con una aproximación evolucionista.

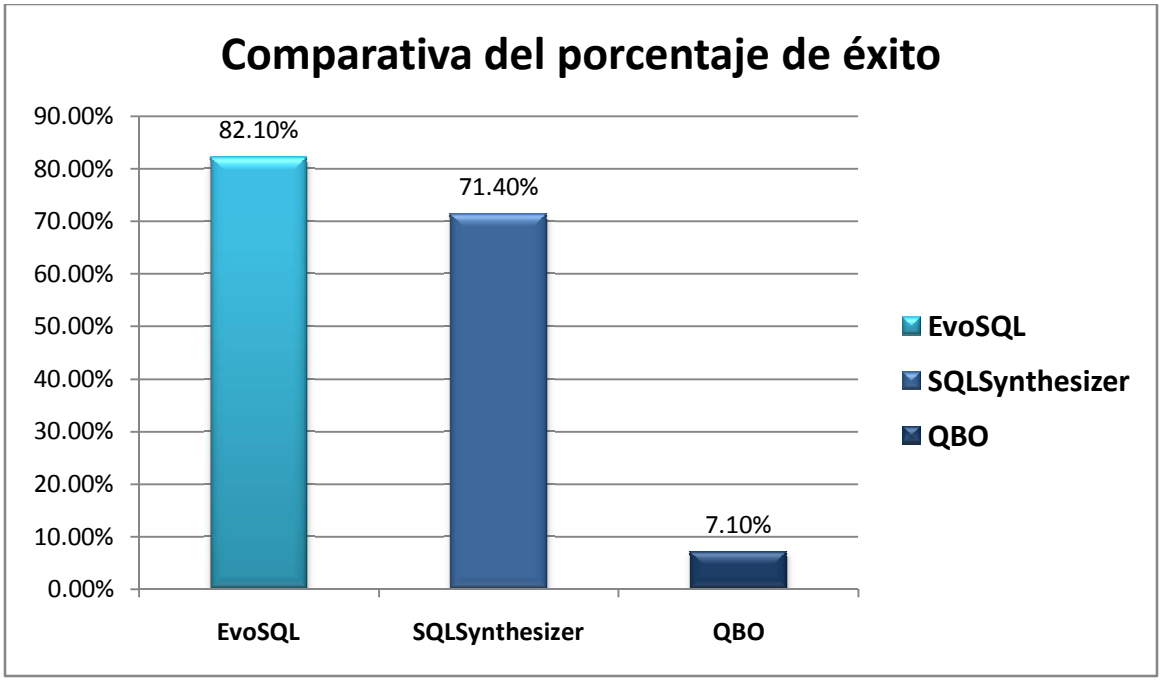

Fig4.1Comparativa del porcentaje de éxito

4.2.5 Adaptación

En la siguiente gráfica (ver fig4.2), se pueden observar los valores de la función objetivoaplicada al ejemplo de la fig 3.1 y fig 3.2, en la generación 0 a la última generación (en este caso la 9). Igualmente, se puede observar que la generación 0 inicia con valores base alrededor de 0.18 y encuentra sus mejores valores en ciertos picos alrededor de 0.55, teniendo un promedio de 0.26para sus individuos, mientras que en la generación 9 se toma como valor mínimo en 0.019 y maneja un constante incremento encontrando el valor máximo 1, estableciendo un promedio de 0.57para su población, esto muestra claramente la adaptabilidad de los programas con base en la función objetivo prevista a través de las generaciones.

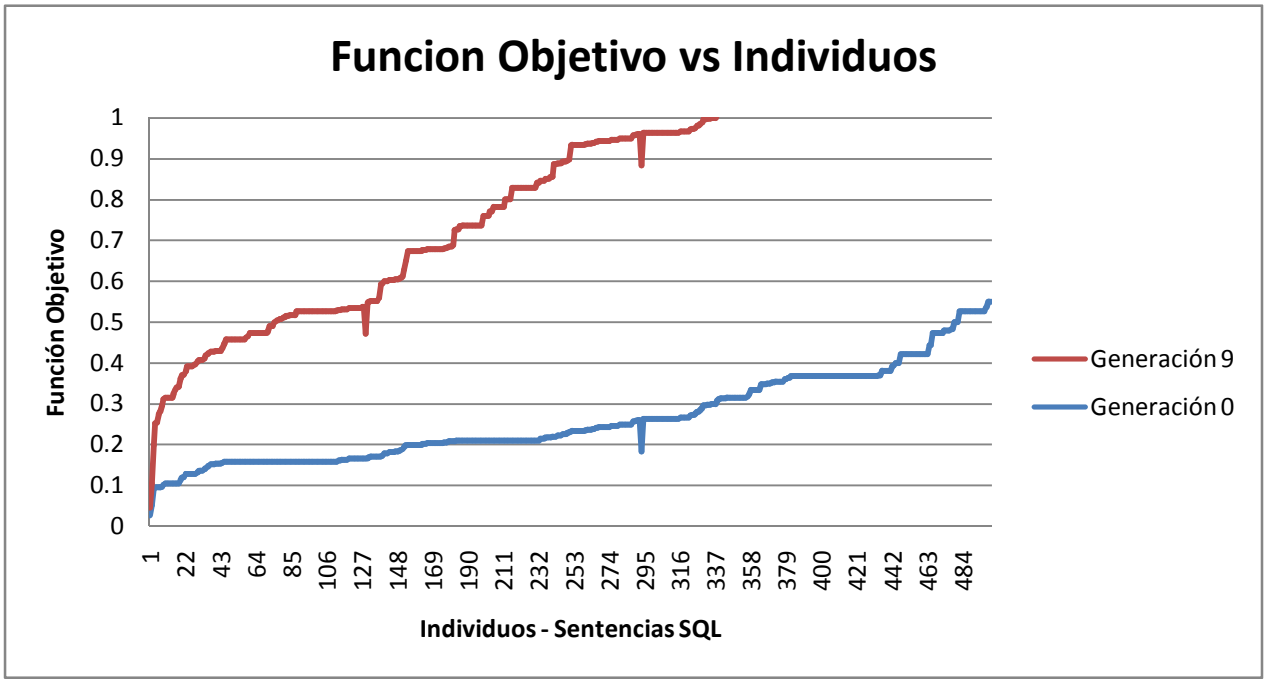

Fig4.2Adaptación de la población

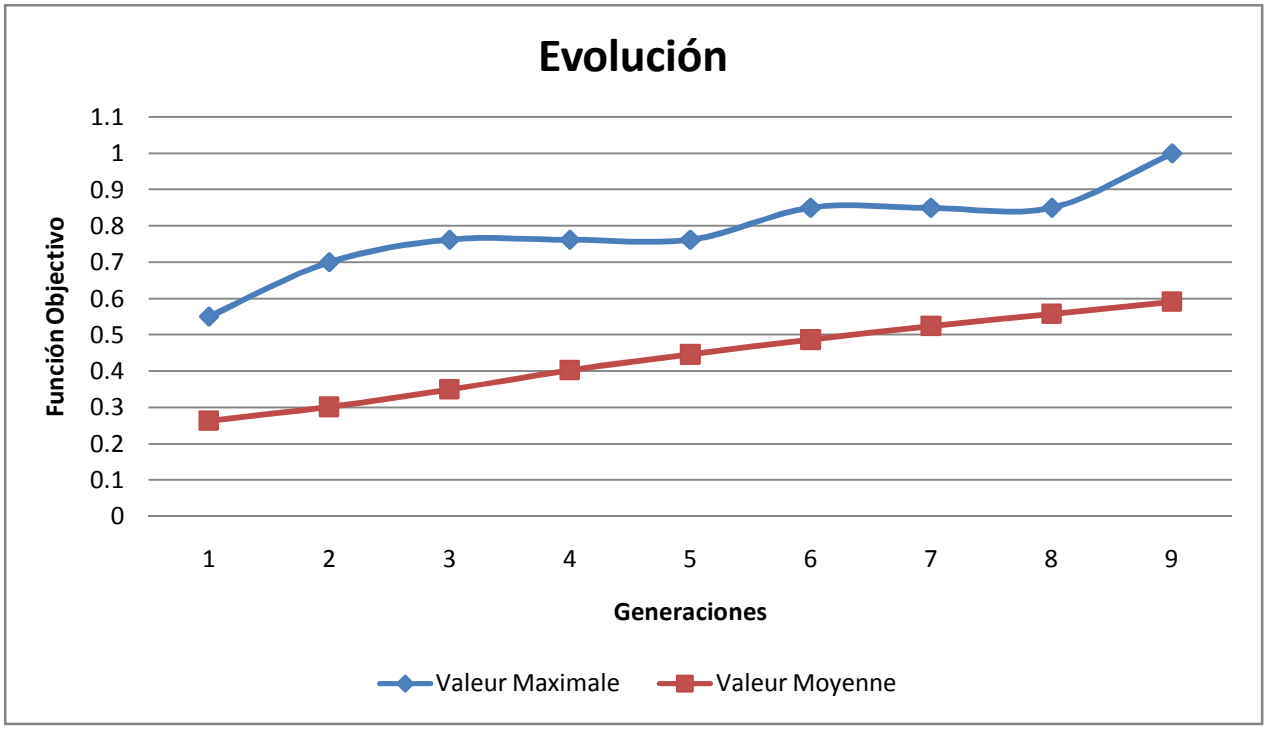

Fig4.3Promedio de evolución de los programas.

En la fig. 4.3 se establecen los valores máximo y promedio de cada generaciónpara el ejemplo de la fig 3.1 y fig 3.2, se puede ver cómo se presenta un incremento constante y una evolución de los programas con respecto al valor de la función objetivo en cada generación evaluada.

4.3 Discusión

Al iniciar esta investigación, se planteó que una de las características a tener en cuenta para la validación de nuestra aproximación, era mantener las mismas características cubiertas en la investigación [39], uno de los factores a tener en cuenta es la cobertura de cláusulas SQL que esta aproximación abarca, ya que los ejercicios sin éxito necesitan de cláusulas extra que en este momento no son compatibles con EvoSQL ni con SQLSynthesizer[39]; sin embargo, se obtuvieron mejores resultados de éxito en EvoSQL conservando las mismas características SQL y de casos de estudio para validación.

A diferencia del SQLSynthesizer[39], que requiere que los usuarios suministren ejemplos sin ningún tipo de error, EvoSQL acepta errores en la digitación de los datos de ejemplo, tomando un poco más de tiempo de ejecución, pero se pueden obtener buenos resultados bajo este tipo de errores, no siendo esta un limitante para su ejecución.

SQLSynthesizer[39] establece que no proporciona información con respecto a cuándo el usuario debe renunciar y asumir que, no se puede sintetizar una sentencia SQL perfecta para el ejemplo en ejecución; por su parte EvoSQL al establecer la cantidad de generaciones máxima brinda un límite al usuario y el algoritmo siempre termina convergiendo a un valor ya sea la cota máxima o al valor máximo alcanzado en su generación.

Se pudo establecer que, a mayor número de población menor número de generaciones y cuando la población es muy pequeña, posiblemente no se encuentre una solución deseada dado a lo limitado del espacio.

Como se estableció EvoSQL y SQLSynthesizer asumen que el usuario tiene conocimiento del sistema y en especial de la base de datos. La sentencia SQL generada no representa la intención del usuario al brindar los ejemplos.

Se establece como una amenaza a la validez, el hecho de que a pesar de la cantidad de casos estudios y de su naturaleza variada, abordados por los ejercicios del texto de base de datos [34], no se puede aún generalizar de resultados a cualquier campo.

El utilizar un algoritmo genético que utiliza elementos aleatorios, implica que no siempre se va a encontrar la misma solución a determinado ejercicio, en igualdad de tiempos y demás particularidades debido al factor de azar manejado. Sin embargo, los resultados obtenidos con la aproximación en la versión EvoSQL genera confianza; cabe resaltar que los ejercicios del manual [34] son poco realistas, llenos de redundancias en nombres y valores, diseñados específicamente para desconcertar, confundir, enredar y desafiar el trabajo del algoritmo; en comparación con los ejercicios del foro[18] que resultaron ser problemas realizados con datos reales EvoSQL, ésta herramienta tuvo éxito en todos además de excelentes tiempos de ejecución en cada uno.

Se puede decir que,EvoSQL se convierte en una opción como herramienta para el usuario no experto para la generación de sentencias SQL a partir de ejemplos y que la programación genética es una buena técnica con la cual se pueden obtener buenos resultados.

CAPÍTULO 5

CONCLUSION

En este capítulo,se realiza una recopilación de la contribución de los resultados obtenidos, las limitaciones de la aproximación y los trabajos futuros.

5.1 Resumen de la contribución

Este estudio ha sido realizado con el fin de ofrecer a los usuarios no expertos, una opción para la generación de sentencias SQL a partir de ejemplos. Estos usuarios tienen la necesidad de desarrollar sentencias SQL en pequeñas bases de datos para sus investigaciones y no es viable la intervención de un experto en la materia, debido a diferentes factores como el tamaño de la investigación o capital etc.

Actualmente, para mitigar este problema existen herramientas como SQLSynthesizer[39] y QBO[31] que ofrecen una opción al usuario, sin embargo, estas herramientas presentan algunos limitantes como por ejemplo, la práctica en QBO es muy limitada, por el número de tablas y la cobertura de cláusulas SQL, y a pesar de que SQLSynthesizer cubre una amplia gama de cláusulas, la talla de sus aplicaciones aún es pequeña y su proceso para sintetizar una solución perfecta, continúa solicitando del usuario no experto una serie de conocimientos y participación dentro del proceso más del deseado. Por lo cual, se estableció la idea de realizar una aproximación con un método de aprendizaje diferente para la síntesis de sentencias SQL y así establecer la posibilidad de obtener mejores

resultados , bajo una herramienta completamente automática y con un proceso simple y sencillo de desarrollo, donde la intervención del usuario fuera mínima.

Esta investigación, se realizó bajo la premisa de utilizar una aproximación evolucionista, con una técnica de aprendizaje como la programación genética y realizando la validación de nuestros resultados bajo las mismas características utilizadas en SQLSynthesizer, tales como:

- Las 10 cláusulas más utilizadas en la declaración de una sentencia SQL {"SELECT…FROM","WHERE","ORDER BY", "GROUP BY", "HAVING", "COUNT", DISTINCT", "AVG", "SUM", "MAX", "MIN"}

- 28 casos de estudio, 23 de los cuales pertenecen al manual de bases de datos [32] y 5 ejercicios expuestos en foros de internet[18].

- Unnúmeromáximo de 10 iteraciones, comolímiteparaencontrarunasolución perfecta.

5.2 Resumen de los resultadosobtenidos

Para la validación de la aproximación presentada en esta memoria, en su versión EvoSQL, se emplearon los mismos casos de estudio de SQLSynthesizer, donde fueron ejecutados 28 ejercicios, 23 pertenecientes a un manual de bases de datos y 5 ejercicios propuestos en foros de internet.

De esta evaluación se puede concluir:

  - EvoSQL tiene 82.1% de éxito para encontrar una solución perfecta para sentencias SQL.

- El 17.9% restante sin éxito, ha sido distribuido en:

    - 10.8% pertenecen a los ejemplos donde es necesario utilizar la cláusula IN entre otras para encontrar una solución, cláusulas no cubiertas por esta investigación.
    - 7.1% restante considerado como "el precio del azar" por utilizar elementos aleatorios; sin embargo, se obtiene un alto valor objetivo, esto significa que el algoritmo no está totalmente perdido en su búsqueda.

- Con respecto al esfuerzo humano :

    - En la escritura de los ejercicios : Se mantuvo igual en EvoSQL y SQLSynthesizer

    - Durante la interacción con la herramienta en la búsqueda de una solución perfecta:

        - SQLSynthesizer establece interacciones múltiples del usuario, debido a las iteraciones múltiples a realizar.
        - EvoSQL tiene una única intervención del usuario al inicio del programa para establecer parámetros (la población, las generaciones, etc.)

    - El tratamiento de errores humanos en la digitación:

        - SQLSynthesizer no soporta el error en la escritura de los campos del ejercicio.

- EvoSQL soporta este tipo de errores gracias al tipo de comparación realizada.

- Con respecto a los tiempos de ejecución :

  - SQLSynthesizer establece un promedio de 9 segundosde tiempo de ejecución, pero obvia de 1-5 minutos por cada iteración del usuario y establece una media de 2.5 rounds para encontrar una solución perfecta.
  - EvoSQL establece como promedio total de ejecución 3.11 minutos y 3 generaciones para encontrar una solución perfecta.

- Comparación de éxito:

  - EvoSQL tiene 82.1% de efectividad. Esta efectividad es superior a la establecida por SQLSynthesizer y 75% más alta que la herramienta más próxima Queryby Output QBO.

- Con respecto a la adaptación:

  - El algoritmo genético funciona bien en este tipo de problemas, donde el espacio solución es considerablemente pequeño. Este establece tiempos aceptables de ejecución y simplicidad en el proceso, y es quien hace a la programación genética, una buena técnica de aprendizaje.

5.3 Limitaciones

Las limitaciones existentes son las inherentes al algoritmo genético, es decir:

- El espacio de búsqueda sólo puede trabajar con poblaciones finitas no muy grandes ya que, de lo contrario se convierte en un espacio infinito de combinaciones.

- Es un algoritmo de búsqueda ciego, es decir, solamente guiado por la aptitud de los individuos y no incorpora conocimientos específicos del problema en cuestión, ya que no hace parte del proceso evaluar la intención del usuario en la búsqueda de una solución perfecta.

- El algoritmo realiza la búsqueda de los mejores puntajes, utilizando únicamente el valor objetivo de las sentencias, pero a veces la información proporcionada es insuficiente para orientar correctamente el algoritmo en su búsqueda de la solución perfecta, la cual se considera como "el precio del azar"; sin embargo, si se desorienta no necesariamente se pierde del todo.

- Para que el algoritmo funcione correctamente, necesariamente se debe suministrar una cantidad mínima de información, es decir, una amplia población y buenos ejemplos de entrada y salida. Si no se suministra esta mínima información, el mecanismo no funciona con propiedad, y este evolucionará incorrectamente.

5.4 Trabajos futuros

Después de realizar este trabajo de investigación, surgen varias líneas por la cuales se puede continuar y se proponen como trabajos futuros los siguientes:

- Una aproximación que incluya cláusulas de SQL que en este estudio no fueron evaluados, clausulas como INNER JOIN,LEFT JOIN y RIGHT JOINla inclusión de sentencias anidadas utilizando IN, esto permitiría generar sentencias más complejas y ampliaría el campo para encontrar soluciones que no fueron encontradas en este trabajo.

- Realizar una adaptación al algoritmo genético en la etapa de mutación, para generar nuevas características no tomadas en cuenta en esta investigación. En esta etapa se puede generar mutaciones no solo de cláusulas o palabras reservadas como "AND" o "GROUP BY" sino de condiciones completas por ejemplo "AND(Filme.titulo="Amelie")", o mutar el valor de los campos "AND (Filme.titulo="la vie en rose")" cosas con las que actualmente no cuenta nuestra aproximación.

- Se recomienda ampliar los casos de estudio evaluados para generalizar los resultados sobre un campo arbitrario.

Como conclusión general, se puede decir que se ha obtenido una herramientacon una aproximación evolucionista, que se presenta como una opción, con buenos resultados y un proceso simple para un usuario no experto.

Bibliografía

[1] A. Kuri A. Kuri. "PatternRecognitionvia Vasconcelos' GeneticAlgorithm". Lecture Notes in Computer Science. Vol. 3287, pp. 328-335. 2004.

[2] ABDUL Khalek, Shadi; KHURSHID, Sarfraz. Automated SQL query generation for systematic testing of database engines. En Proceedings of the IEEE/ACM international conference on Automated software engineering. ACM, 2010. p. 329-332.

[3] AFZAL Wasif, Richard Torkar. On the Application of Genetic Programming for Software Engineering Predictive Modeling: A Systematic Review. Expert System with Applications 2011.

[4] BÄCK, J., Reducing Bias and Inefficiency in the Selection Algorithm, en Grefenstette, J. (editor),Proceedings of the First International Conference on Genetic Algorithms, Lawrence Erlbaum Associates, pag. 14-21, Hillsdale, NJ, 1985.

[5] BÄCK, T., Selective Pressure in Evolutionay Algorithms: a Characterisation of Selection Mechanisms, en Fogel, D. (editor), Proceedings of the First IEEE Conference on Evolutionary Computation, IEEE Press, pag. 57-62, 1995.

[6] BARATE, Renaud. Apprentissage de fonctions visuelles pour un robot mobile par programmation génétique. 2008. Tesis Doctoral. Université Pierre et Marie Curie-Paris VI.conference on Automated software engineering. ACM, 2010. p. 329-332.

[7] BAKI,Islem, Sarahoui, Houari, Masson, Philippe, Faunes, Martin,et al. Learning Implicit and Explicit Control in Model Transformations by Example

[8] BOURGOIN, Brice. Construction de caractéristiques par programmation génétique pour un système de reconnaissance multiclasse. 2007. Tesis Doctoral. École de technologie supérieure.

[9] CHAMBERLIN, Donald D., et al. A history and evaluation of System R. Communications of the ACM, 1981, vol. 24, no 10, p. 632-646.

[10] CHEUNG, Alvin; SOLAR-LEZAMA, Armando; MADDEN, Samuel. Optimizing database-backed applications with query synthesis. ACM SIGPLAN Notices, 2013, vol. 48, no 6, p. 3-14.

[11] DARWIN Charles, On the origin of species, New York: D. Appleton and Co., 1859. http://www.biodiversitylibrary.org/bibliography/28875

[12] De Jong, K., An Analysis of the Behaviour of a Class of Genetic Adaptive Systems, Ph.D. Thesis, University of Michigan, Ann Arbor, 1975.

[13] DATAS ofSQLSynthesizerhttp://sqlsynthesizer.googlecode.com/

[14] EGENHOFER, Max J.. . Spatial SQL: A query and presentation language.Knowledge and Data Engineering, IEEE Transactions on, 1994, vol. 6, no 1, p. 86-95.

[15] FAUNES, Martin, Cadavidjuan, Baudry Benoit, et al Automatically searching for métamodèle well-formedness rules in examples and counter-examples. In :Model-Driven Engineering Languages and systems. Springer Berlin heldelberg. 2013 p. 187-202

[16] FAUNES, M., Sahraoui, H., Boukadoum, M.: Generating model transformation rules from examples using an evolutionary algorithm. In: Automated Software Engineering, pp. 1–4 (2012)

[17] FAUNES, M., Sahraoui, H., Boukadoum, M.: Genetic-programming approach to learn model transformation rules from examples. In: Duddy, K., Kappel, G. (eds.) ICMB2013. LNCS, vol. 7909, pp. 17–32. Springer, Heidelberg (2013)

[18] FORUM question Pag Web http://forums.tutorialized.com/sql-basics-113/finding-list-of- duplicate-records-with- more-than-two-duplicates-379845.html, http://forums.tutorialized.com/sql-basics-113/sql-query-379547.html, http://forums.tutorialized.com/sql-basics-113/need-help-in-wiring-a-query-380750.html,http://forums.tutorialized.com/sql-basics-113/inner-join-with-multiple-conditions-378384.html 2014.

[19] FLORES, Perfecto Malaquias Quintero, et al. Extracción de reglas lingüísticas difusas por la programación genética. En LFA'10: Lógica Difusa y Aplicaciones. 2010

[20] FOGEL, L.J., Owens, A.J., Walsh, M.J. (1966), Artificial Intelligence through Simulated Evolution, John Wiley

[21] GOLDBERG, D., Korb, B., y Deb, K., A Comparative Analysis of Selection Schemes Used in Genetic Algorithms, en Rawlings, G. (editor), Foundations of Genetic Algorithms, Morgan Kaufmann Publishers, pag. 69-93, San Mateo, CA, 1991

[22] GUARDA, Alvaro. Apprentissage génétique de règles de reconnaissance visuelle. Application à la reconnaissance d'éléments du visage. 1998. Tesis Doctoral.

[23] HOLLAND, J.H., Adaptation in Natural and Artificial Systems, University of Michigan Press, Ann Arbor, 1975.

[24] ISO/IEC 9075 defines Structured Query Language (SQL) Information technology -- Database languages – SQL. 2011

[25] JAVA http://www.oracle.com/technetwork/es/java/javase/downloads/index.html

[26] J. R. Quinlan. Induction of decision trees. Mach. Learn., 1(1):81–106,March 1986.

[27] KOZA, John R. Genetic programming: on the programming of computers by means of natural selection. Vol. 1 MIT press, 1992.

[28] MITCHELL, Melanie, "Chapter 3: Genetic Algorithms in Scientific Models", An Introduction to Genetic Algorithms, The MIT Press, Cambridge, MA, (1996), pp. 85-108

[29] MYSQL. http://www.mysql.com.

[30] QUENTIN, Cobraert; Mason, Philippe. Une approche génétique pour la dérivation de règles et méta-règles de transformation de modèles à partir d'exemples. 2013.

[31] QUOC Trung Tran, Chee-Yong Chan, and Srinivasan Parthasarathy. Query by output. In SIGMOD, pages 535–548, 2009

[32] RAGHU Ramakrishnan and Johannes Gehrke. Database Management Systems. Addison- Wesley (3rd Edition), 2007.

[33] TIFFANY, Rob. Data Manipulation Language. En SQL Server CE Database Development with the. NET Compact Framework. Apress, 2003. p. 139-179.

[34] WELLS, Garth. Data Definition Language. En Code Centric: T-SQL Programming with Stored Procedures and Triggers. Apress, 2001. p. 35-70.

[35] WETZEL, A., Evaluation of the Effectiveness of Genetic Algorithms, en Combinatorial Optimization, Technical Report, University of Pittsburgh, 1983.

[36] WHITLEY, D., Genitor: a Different Genetic Algorithm, en Proceedings of the Rocky Mountain Conference on Artificial Intelligence, Denver, 1989

[37] YEN, MY-M.; Scamell, Richard W.. . A human factors experimental comparison of SQL and QBE. Software Engineering, IEEE Transactions on, 1993, vol. 19, no 4, p. 390-409.

[38] YUROVITSKY, Michael. Programmation génétique pour le jeu Tetris.

[39] ZHANG, Sai; SUN, Yuyin. Automatically synthesizing SQL queries from input-output examples. En Automated Software Engineering (ASE), 2013 IEEE/ACM 28th International Conference on. IEEE, 2013. p. 224-234.

[40] ZLOOF, Moshe M.. . QBE/OBE: a language for office and business automation. Computer, 1981, vol. 14, no 5, p. 13-22

[41] ZHIBIN Lei; Ganapathy, S. Kicha; Safranek, Robert J. Miracle: multimedia information retrieval by analysing content and learning from examples. En Applications of Computer Vision, 1998. WACV'98. Proceedings., Fourth IEEE Workshop on. IEEE, 1998. p. 272-273.

[42] ZHI-NING Xia et al. Chemical components determination via terahertz spectroscopic statistical analysis using microgenetic algorithm. Optical Engineering, 2011, vol. 50, no 3, p. 034401-034401-12.